Szittya Ottó

Digitális és Analóg Technika

Informatikusoknak

II.

Dr. Szittya Ottó

DIGITÁLIS ÉS ANALÓG TECHNIKA

INFORMATIKUSOKNAK

II. kötet

Nyitott rendszerű képzés
– távoktatás – oktatási segédlete
Felsőoktatási tankönyv

Gábor Dénes Főiskola
Budapest, 2001

Készült az Oktatási Minisztérium támogatásával

Lektorálta: **Dr. Kóczy T. László**
a műszaki tudományok doktora
egyetemi tanár

Dr. Flesch István
egyetemi adjunktus

Dr. Adamis Gusztáv
egyetemi adjunktus

ISBN 963 577 261 0 ö
ISBN 963 577 263 7 II.

Kiadó: **LSI Oktatóközpont**
Felelős vezető: **Dr. Kovács Magda**
Témafelelős: **Flier István**
Tördelés: **HEXACO GNH Kft.**

LIGATURA KFT – TEXT KFT

A könyv megrendelhető vagy megvásárolható:
LSI Oktatóközpont, 1037 Budapest, Bécsi út 324.
Tel./fax: 250-60-13

TARTALOMJEGYZÉK

9. KÜLÖNLEGES DIGITÁLIS HÁLÓZATOK

Az eddigiekben tárgyalt főként BOOLE–SHEFFER–PEIRCE axiómákon felépített digitális hálózatokon túlmenően különféle, más elveket felhasználó eszközök és hálózatok is kialakultak. Ezek karrierje – a mikroelektronika eszközök fejlődésének, és a felhasználói elvárásoknak függvényében – változatosan alakult az idők folyamán. Jó példa az eszközfejlődés és a felhasználás kapcsolatára a BOOLE elven működő áramlogikás *híd-típusú* hálózatok esete (5-4. ábra). A klasszikus jelfogás-hálózatok korszakában a *soros–párhuzamos* struktúrának *híd-struktúrába* való áttranszformálása hasznos törekvés volt, mivel így lényegesen egyszerűbb áramlogikás hálózathoz jutottak. Az elsősorban feszültséglogikákat felhasználó bipoláris félvezetők egyeduralma idején a híd-típusú hálózatok feledésbe merültek, majd a *MOS transzferelemek* megjelenése után a VLSI technikában újra szerephez jutottak, lényegesen magasabb műszaki színvonalon.

A fejezet keretében néhány fontosabbnak tekinthető, részben teljesen új, részben BOOLE-elveket hasznosító különleges digitális hálózatot mutatunk be az alábbiakban.

9.1. Szimmetrikus-, pozicionális- és híd-hálózatok

Ez a hálózatcsoport BOOLE-elveken alapul, és elsősorban áramlogikás vonatkozásban mutat különleges tulajdonságokat.

!

9.1.1. Szimmetrikus függvények és hálózatok

9.1.1.1. Szimmetrikus BOOLE-függvények

A szimmetrikus hálózatok a logikai függvények egy speciális osztályán alapulnak.

Valamely $F^n(X_1 \ldots X_n)$ logikai függvényt *szimmetrikusnak* (totál-szimmetrikusnak) nevezzük, ha független változóinak tetszőleges permutációjára invariáns tulajdonságot mutat.

Az $F^n(X_1 \ldots X_n)$ függvényt *részben szimmetrikusnak* (parciálisan szimmetrikusnak) nevezzük $\{X_i \ldots X_K\}$ változókra ($\{X_i \ldots X_K\} \subset \{X_1 \ldots X_n\}$) akkor és csak akkor, ha az $X_i \ldots X_K$ változók tetszés szerinti összecserélése esetén invariáns tulajdonságot mutat.

Azokat a változókat, melyekre nézve a függvény szimmetrikus tulajdonságot mutat, *szimmetria-változók*nak nevezzük. Például:

a) $Y_k = AB+AC+BC$ – totál szimmetrikus
b) $Y_L = A\,\overline{B}C+A\overline{B}\,\overline{C}$ – parciálisan szimmetrikus A és C-re
c) $Y_M = ABC+\overline{A}BC+\overline{A}\,\overline{B}\,\overline{C}$ – A, B és $\overline{C}$-re szimmetrikus.

A szimmetrikus függvényekhez jellegzetes számok, az ún. β *skalár jellemzők* rendelhetők. A függvény szimmetrikus voltának szükséges és elégséges feltétele, hogy meghatározható egy $\beta_1, \beta_2 \ldots \beta_K$ számcsoport olymódon, hogy ha pontosan β_i számú $i = 1, 2 \ldots k$ független változó értéke „1" (vagy „0"), akkor és csak akkor a függvény értéke is „1" (vagy „0").

A $0 \le \beta_i \le n$ skalár jellemzők úgy is értelmezhetők, mint a szimmetrikus függvény „1"-es értékeihez tartozó term-ek *„súlya"*, ugyanis valamely term súlyán a behelyettesített értékkészletében szereplő „1"-esek számát értjük.

Például az a) eset $Y_K = F(A, B, C)$ függvényre $\beta = 2$ és 3, mert akkor és csak akkor vesz fel „1" értéket, ha független változói közül egyidejűleg 2 vagy 3 vesz fel „1" értéket. Ezt a továbbiak során a következőképpen jelöljük: $F^3_{(2)(3)}$.

A szimmetrikus függvényt *„elemi szimmetrikus függvény"*-nek nevezzük, ha csak egyetlen β skalár jellemzője van. (9.1) és (9.2) alapján belátható, hogy $(2n + 2)$ darab elemi szimmetrikus függvény létezik.

$$\left.\begin{aligned} &F^n_{(0)} = \overline{X}_1 \cdot \overline{X}_2 \ldots \overline{X}_n = E^n_0 \\ &F^n_{(1)} = (\overline{X}_1 \cdot \overline{X}_2 \ldots X_n) + \ldots + (X_1 \cdot \overline{X}_2 \ldots \overline{X}_n) = \\ &= \text{az egy darab „1"-est tartalmazó } E^n_i \text{-k összege} \\ &\vdots \\ &F^n_{(n)} = X_1 \cdot X_2 \ldots X_n = E^n_{2^n-1} \end{aligned}\right\} \quad (9.1)$$

továbbá: $i = 0, 1 \ldots n$-re $G^n_{(i)} = \overline{F^n_{(n-i)}}$ (9.2)

Bebizonyítható, hogy bármely szimmetrikus függvény előállítható elemi szimmetrikus függvények összegeként (szorzataként).

A következőkben (korlátozott számú változóra felírt alakban) összefoglalunk néhány, szimmetrikus függvényekre vonatkozó fontosabb tételt, melyek transzformációval bármelyik más bázis-rendszerben felírhatók és „n" változóra is általánosíthatók:

$$\left.\begin{aligned} F^3_{(2)(3)}(X_1,X_2,X_3) &= \overline{F}^3_{(0)(1)}(X_1,X_2,X_3) \\ F^3_{(2)(3)}(X_1,X_2,X_3) &= F^3_{(0)(1)}(\overline{X}_1,\overline{X}_2,\overline{X}_3) \\ F^5_{(0)(2)(4)(5)} &= F^5_{(0)(2)(5)} + F^5_{(2)(4)} \\ F^5_{(2)(3)} &= F^5_{(0)(2)(3)} \cdot F^5_{(1)(2)(3)(4)} \end{aligned}\right\} \qquad (9.3)$$

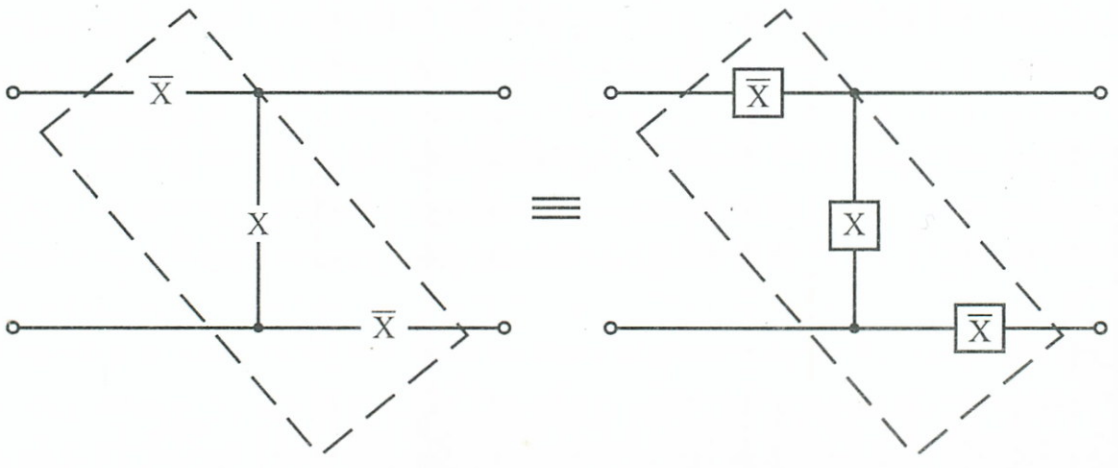
9.1.1.2. Szimmetrikus függvényeket realizáló hálózatok

A szimmetrikus logikai függvények klasszikus realizálási területét az ún. reiteratív struktúrájú hálózatok képezik. Ezek jellegzetessége, hogy felépítésük visszavezethető a 9-1. ábra szerinti ág-típusú alapcellák – vagy ezek negatívjának – *ismétlődő* rendszerére. A reiteratív hálózatokat az LSI MOS technikában alkalmazzák újabban, korábban a sokérintkezős jelfogás hálózatokban játszottak szerepet.

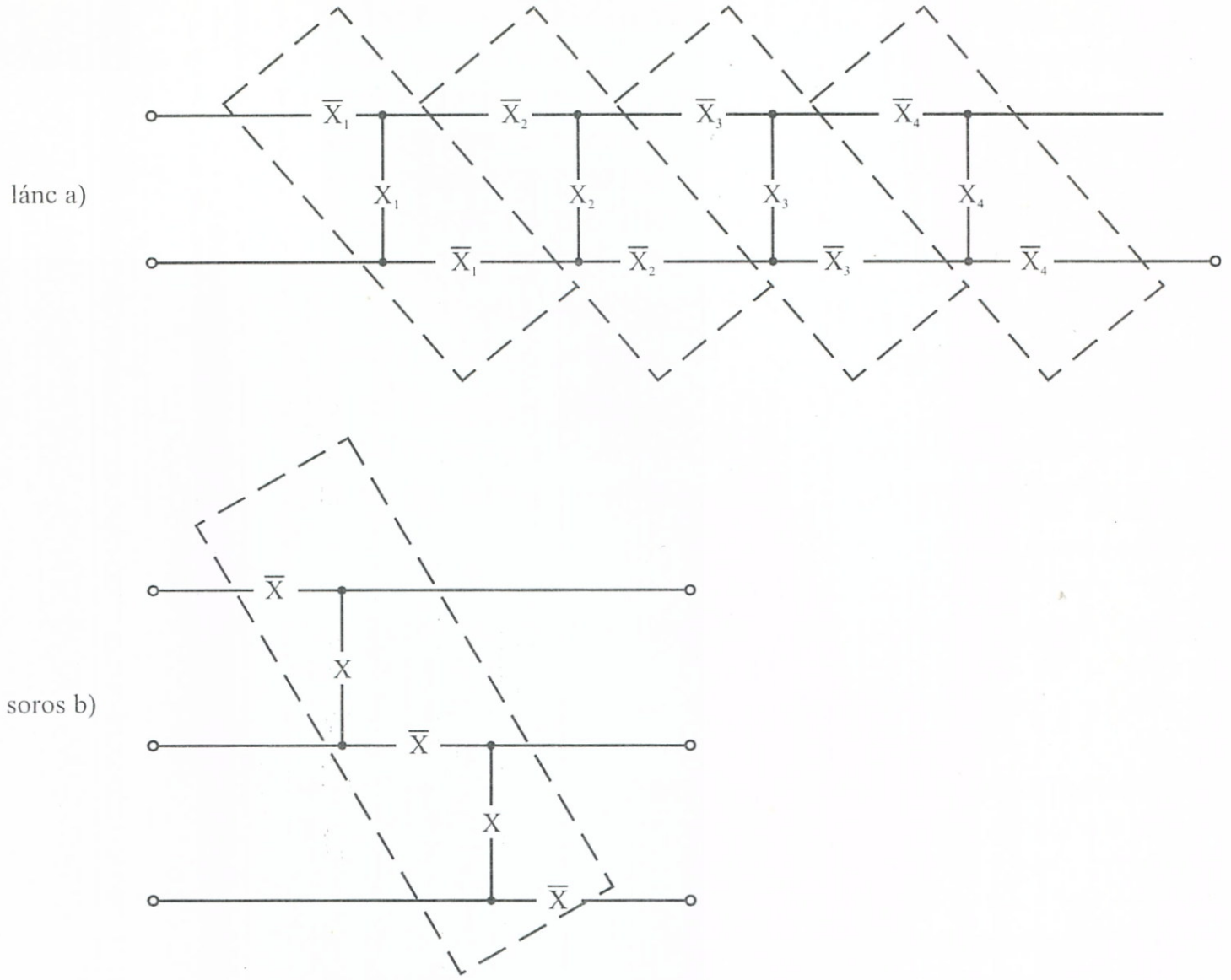
Az alapcellák lánc (9-2a. ábra) és soros (9-2b. ábra) kombinációiból olyan *általános tulajdonságú hálózatok* építhetők fel, melyből meghatározott részek elhagyása után különféle feladatok realizálása válik lehetővé. Egy ilyen általános 5-változós hálózatot mutat példaként a 9-3. ábra. A hálózat természetesen „n" változósra is bővíthető.

Ha a 9-3. ábra szerinti 5-változós általános hálózatból csak a 9-4. ábra szerinti részt hasznosítjuk, akkor egy olyan hálózatot kapunk, amelyik az

$$F^4_{(0)},\quad F^4_{(1)},\quad F^4_{(2)},\quad F^4_{(3)},\quad F^4_{(4)}$$



9-1. ábra *Ág-típusú alapcella*



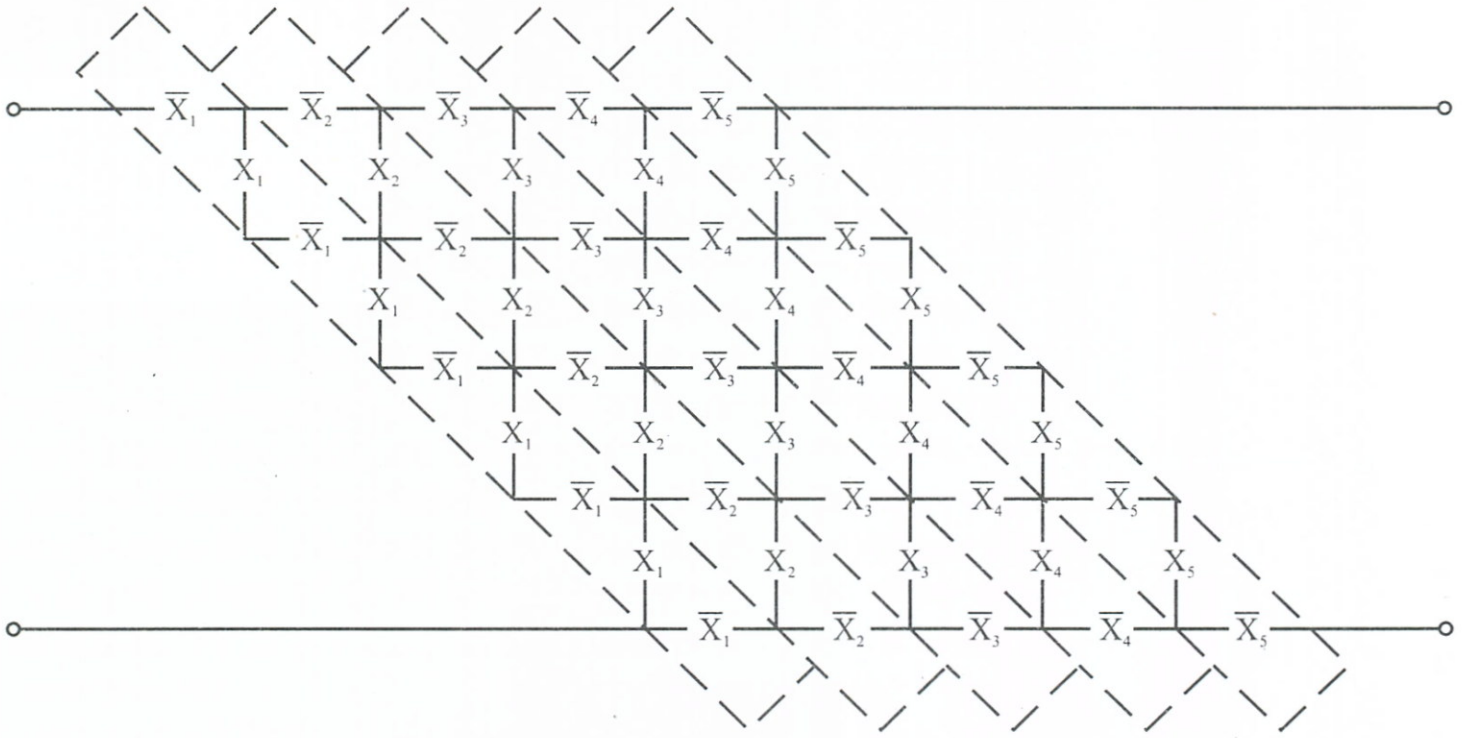
9-2. ábra *Ág-típusú alapcellák lánc-és soros kapcsolatai*

szimmetrikus függvények, vagy ezek diszjunkcióinak előállítására alkalmas. Az ilyen hálózatokat ág-típusú *szimmetrikus hálózat*oknak nevezzük. Mint látni fogjuk, konkrét megtervezésük során többféle topológiai egyszerűsítési szabály definiálható velük kapcsolatosan. Vizsgáljuk meg ezeket egy adott példa keretében.

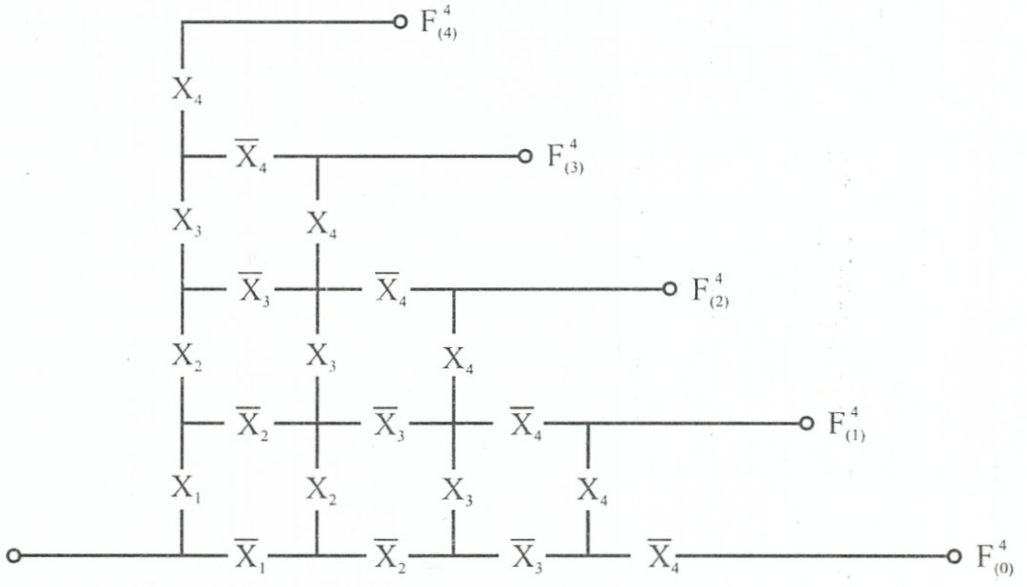
9.1. példa: Legyen a feladat egy olyan kombinációs hálózat tervezése, melynek egy kimenete van, és a kimeneti függvény „1" logikai értéket vesz fel minden olyan esetben, mikor a négy független változó közül egyidejűleg *páros számú* vesz fel „1"-es értéket.

Ha a nulla darabszámot is párosnak tekintjük, akkor a feladatot egy

$$F^{4}_{(0)(2)(4)}$$



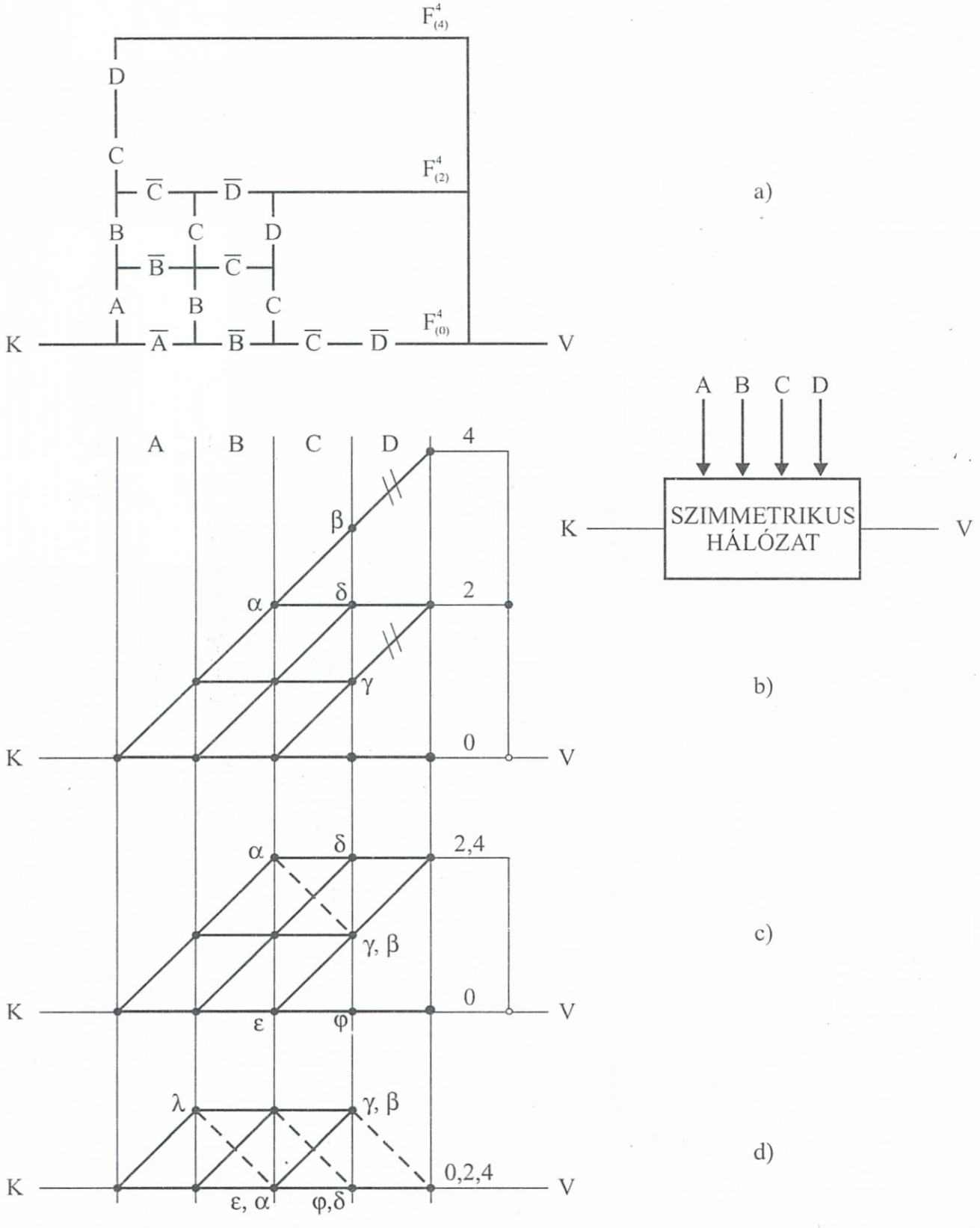
9-3. ábra *5-változós reiteratív struktúra*



9-4. ábra *Szimmetrikus hálózat leszármaztatása reiteratív struktúrából*

típusú szimmetrikus függvénnyel lehet leírni. A példát realizáló hálózatot kialakíthatjuk a 9-4. ábrából, ha ott a felesleges $F^4_{(1)}$ és $F^4_{(3)}$ kimeneteket elhagyjuk és a (9.3) összefüggések értelmében példánkra felírható:

$$F^4_{(0)(2)(4)} = F^4_{(0)} + F^4_{(2)} + F^4_{(4)}$$



9-5. ábra *Reiteratív hálózat egyszerűsítése gráfos módszerrel*

formula alapján az $F^4_{(0)} - F^4_{(2)} - F^4_{(4)}$ kimeneteket párhuzamosan VAGY-ba kapcsoljuk, azaz összekötjük. Az eredmény a 9-5a. ábrán látható hálózat lesz.

Ezt a hálózatot már meg lehetne építeni például *MOS transzfer–gate*-ekkel, vagy *jelfogó–érintkezős* áramlogikás hálózattal, melynél a kapcsolók száma 16 darab lenne. Ez elég nagy számnak mutatkozik, ezért kíséreljük meg a hálózat egyszerűsítését (esetleges többszörösen kihasználható áramutak keresésével). Ennek szisztematikus lebonyolítására vezessünk be egy gráfos jelölésmódot, mely a tervezés további lépéseit meg fogja könnyíteni. A gráfot úgy alakítsuk ki, hogy minden X_i *záró*-kapcsolót egy *ferde* vonal, minden X_i *bontó*-kapcsolót egy *vízszintes* vonal jelképezzen. Ezek után a 9-5a. ábrát helyettesítő „szimbolikus gráf" a 9-5b. ábra szerint fog alakulni. A gráfnál azt is feltételezzük, hogy az *A, B, C, D* intervallumokban egymás felett elhelyezkedő, gráf-rúddal helyettesített kapcsolók egyszerre kapnak működtetési hatást. (Például jelfogós realizációnál ugyanaz a jelfogó működteti az egymás feletti rudaknak megfelelő, egyszerre aktivizálódó érintkezőket.)

Az a) vagy b) ábra-változatot jól megfigyelve, megállapítható, hogy például $D = 1$ bekövetkezésekor a *K* és *V* pontok között a kettős áthúzással megjelölt β–4 és γ–2 ágakon, tehát egyszerre két úton is létesülhet kapcsolat, mivel a két ág egybevágóan működik. Kézenfekvő a gondolat, hogy a két egybevágó ágat egyetlen, kétszeresen kihasznált ággal próbáljuk helyettesíteni. Erre lehetőséget az ún. *„lecsúsztatási"* művelet ad.

Lecsúsztatásnál a β–4 ágat a γ–2 ággal fedésbe hozzuk és a γ pontba „becsúszott" β-nek korábbi α-val fennálló összeköttetését a 9-5c. ábra ferde, ugyancsak záró érintkezőt jelképező szaggatott vonalával pótoljuk. Ezzel a művelettel, tehát egy ág kétszeres kihasználásával a hálózat ágainak számát eggyel csökkenteni tudtuk.

A csúsztatási művelettel kapcsolatban általános szabályként bizonyítás nélkül megjegyezzük a következőket:

- Lecsúsztatni csak *összefüggő ágat,* vagy ágakkal határolt parallelogrammát lehet.
- Csúsztatás csak akkor végezhető, ha az összekötött „emeletek" sorszámai (példánkban: 0, 2, 4) *számtani* haladványt alkotnak, vagy minden emeletet beleértve, vagy a valamelyiktől felfelé következőket. Az utóbbi esetben a csúsztatás csak attól az emelettől felfelé végezhető, melytől a haladvány kezdődik.
- Hamis eredményű a csúsztatás, ha a számtani haladvány utolsó tagja után következő n+1-edik tag nem nagyobb a független változók számánál.

A szimmetrikus hálózatok topológiai egyszerűsítésének egy másik módszere a *„forgatás"*. Ez leginkább akkor vetődik fel, ha az emeletek közt a „0"-ás szerepel.

Forgatásra választott példánkban is kínálkozik lehetőség. Elvégzése úgy történik, hogy a 9-5c. ábra λ–γ, β vízszintes ága, mint „forgástengely" körül a felső hálózatrészt „leforgatjuk", ezáltal az α–ε-nal, δ–φ-vel stb. kerül fedésbe. A keletkező új hálózatot a 9-5d. ábra mutatja, melyen az „átfordult" ágakat szaggatottan jelöltük (pl. λ–$\varepsilon\alpha$). Érdekes megjegyezni, hogy a korábbi lecsúsztatás következtében kapott a 9-5c. ábrabeli α–γ, δ ág „beolvadt" az ε, α–γ ágba.

A 9-5b. kiinduló-gráfot összehasonlítva a kapott, egyszerűsített 9-5d. gráffal megállapíthatjuk, hogy az eredeti 16 darab gráfrúd helyett 12 darab is elegendő a feladat megoldására, tehát érdemes volt az egyszerűsítést megkísérelni.

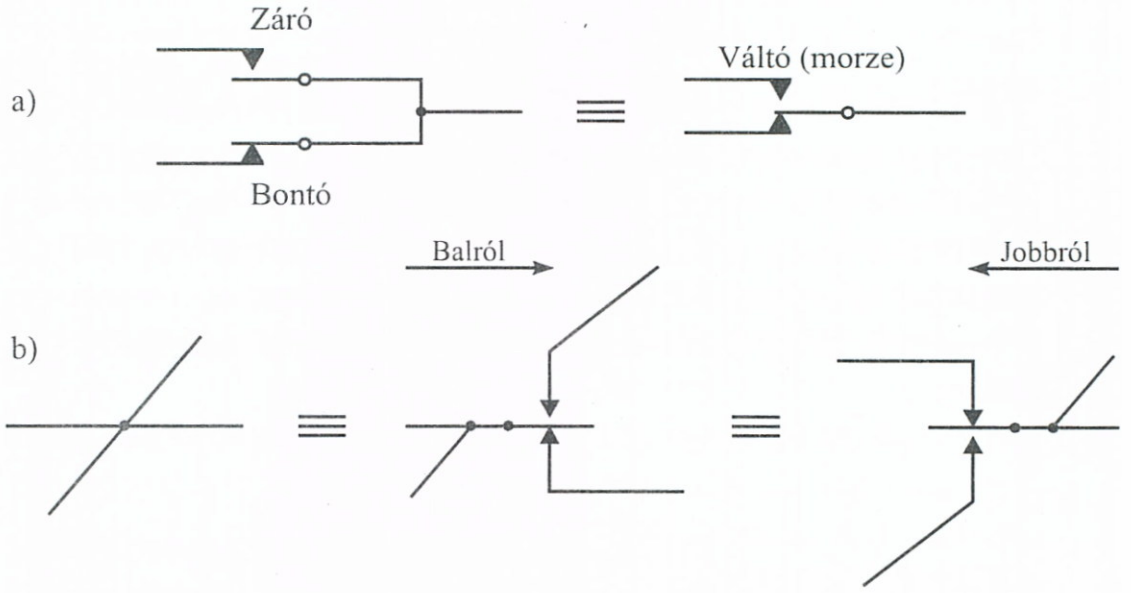
A szimbolikus rács alapján „visszaállíthatjuk" az egyszerűsített kapcsolóhálózatot, melyre itt egy klasszikus jelfogó-érintkezős megoldást mutatunk be.

A gráfos ábrázolásra történő áttéréskor bevezetett megállapodás szerint:

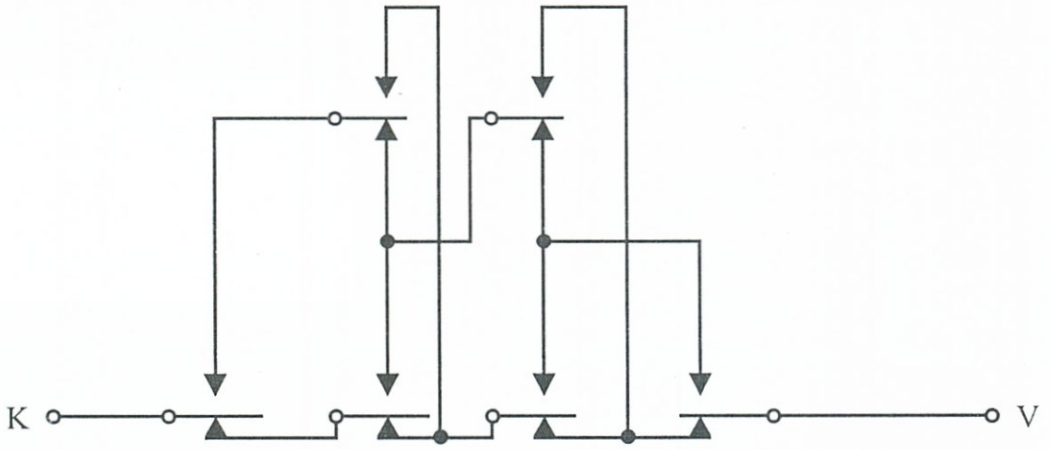
ferde gráfrúd	: záró-kapcsoló
vízszintes gráfrúd	: bontó-kapcsoló

jelentéssel bírt. Most, a visszatranszformáláskor is ezt kell érvényesítenünk.

A klasszikus jelfogós realizáláskor még kihasználható egy további egyszerűsítési lehetőség az ún. *váltó* (morze) érintkező-típus alkalmazása révén, amit a 9-6a. ábra szemléltet. A



9-6. ábra *Érintkezők és a szimbolikus gráf kapcsolata*



9-7. ábra *Szimmetrikus hálózat realizációja klasszikus jelfogóérintkezős realizációval*

gráf → kapcsoló hálózat

transzformáció „szótárát" a 9-6b. ábra mutatja.

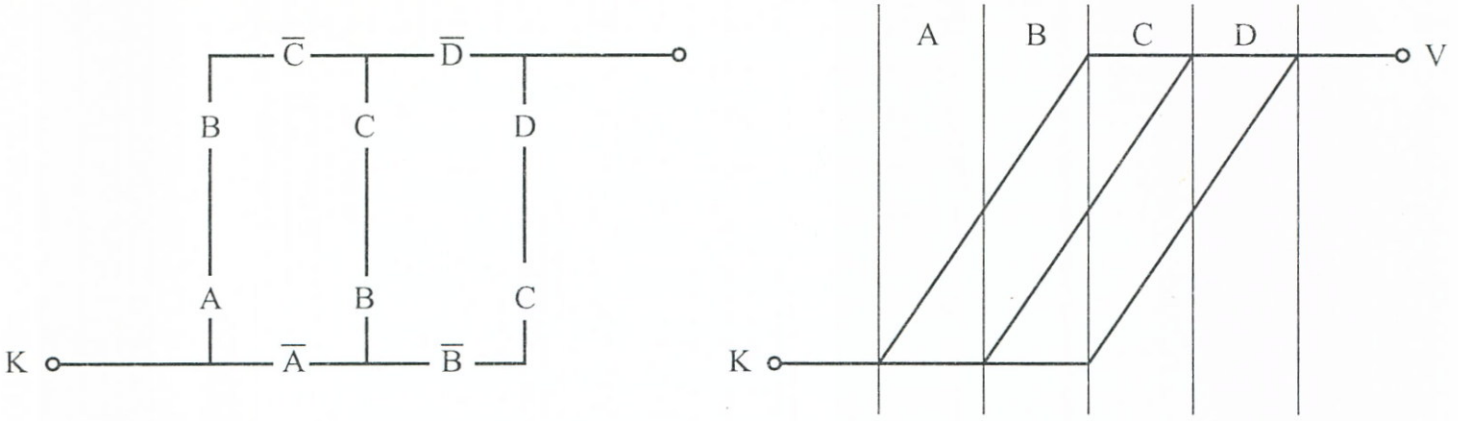
Az elmondottak értelmében a 9-7. ábrán látható morze érintkezős hálózattá rajzolhatjuk át a kapott eredményt. A kiindulás akár K, akár V oldalról történhet az átrajzolásnál, a kapott eredmény a 9-8. ábrán látható.

Megjegyezzük, hogy ha a feladatot nem reiteratív struktúrában oldottuk volna meg, hanem hagyományos módon, a logikai táblázatból kiindulva felírt algebrai formulákat realizáló ágak rendszeréből építettük volna össze, akkor a kapcsolók száma a jelenleginek többszöröse lett volna.

9.1.2. Pozicionális hálózatok

A reiteratív hálózatokkal kapcsolatosan megemlítünk még egy hálózattípust, mely ugyan szigorúan véve nem szimmetrikus függvényeket realizál, de rokonságban áll az előzőkben részletesen tárgyalt változattal. E hálózattípust *pozicionális* ág típusú hálózatnak nevezzük. Származtatása ugyancsak a 9-1. és 9-3. ábrák általános hálózatából kiindulva történhet. Ha ezt oly módon redukáljuk, hogy az egyes sejtekben a közbenső *vízszintes átmeneteket* elhagyjuk, akkor egy hálózat-fajtához jutunk, melynél a bemeneti változók egymáshoz viszonyított *„helyzete"* (pozíciója) határozza meg a kimeneti függvény természetét.

9.2. példa: A 9-8. ábra hálózatát megvizsgálva, megállapítható, hogy $Y_K^4 = 1$ érték olyan esetekben lép fel, midőn a független változók kö-



9-8. ábra *Példa pozicionális hálózatra*

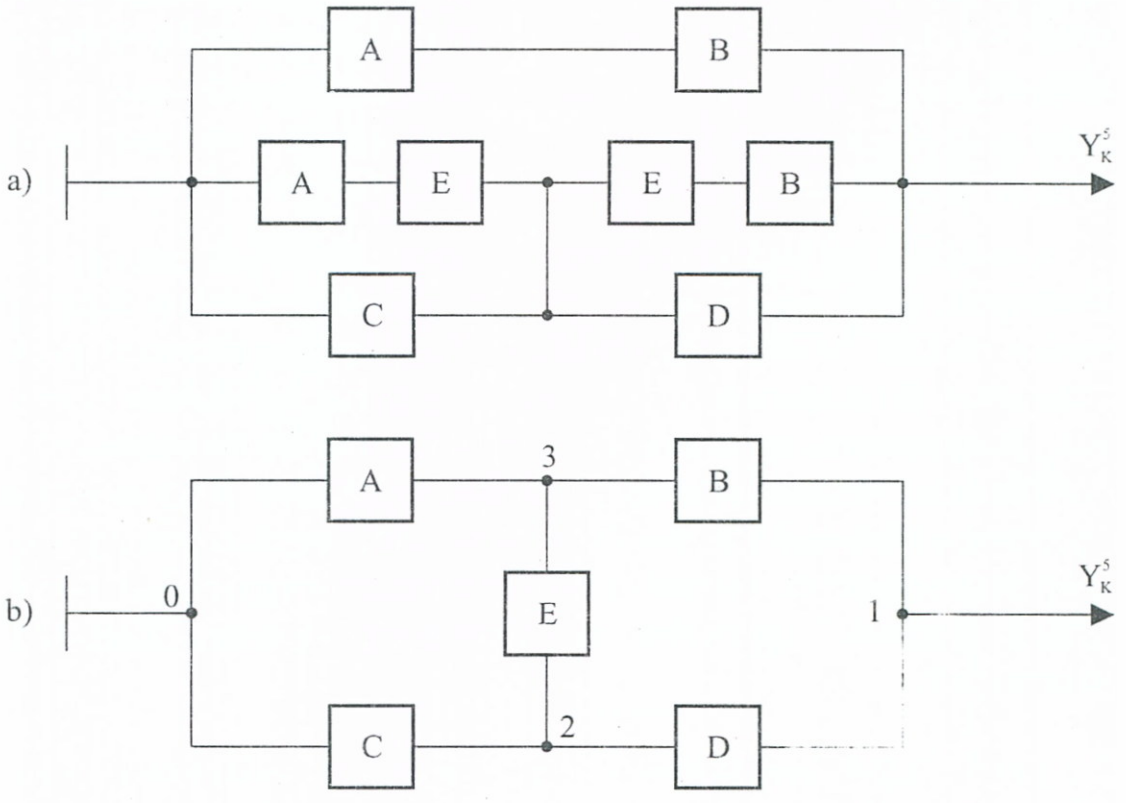
zül valamelyik két, egymás melletti pozíciójú (szomszédos) szerepel „1"-es értékkel.

Az iterációsan realizált hálózat itt is lényegesen egyszerűbb a hagyományos úton előállítottnál.

9.1.3. Híd-struktúrák

A híd-struktúrájú hálózatok az ág-típusú hálózatokon belül, egy jellegzetes felépítésű csoportot képeznek.

Kiindulásul vizsgáljuk meg a 9-9. ábrán látható két ág-típusú logikai vázlatot.



9-9. ábra *Példa híd struktúrára*

Az a) ábra szerinti megszokottabb, ún. soros, párhuzamos típusú és a b) ábra szerinti híd-típusú hálózatok mindegyike ugyanazzal az algebrai függvényalakkal jellemezhető:

$$Y^5 = AB + AED + CD + CEB$$

Szembetűnő viszont, hogy míg a soros–párhuzamos struktúránál 8, a híd-struktúránál csak 5 kapcsolóelem szükséges a hálózat felépítéséhez, mivel ez utóbbinál az *A, B* és *D* kapcsolók többszörösen is ki vannak használva.

a) Közvetlen összeköttetési táblázat

A két struktúra topológiai eltérését a szokásos algebrai módszerekkel nem tudjuk egyértelműen kifejezni, ezért célszerű olyan leírásmódot keresni, mely a topológiát is tartalmazza. Ilyen adottságokkal rendelkezik az ún. *közvetlen összeköttetési táblázat,* melyet a következő kritériumok alapján képezünk:

a) a hálózat minden csomópontjához a táblázaton egy sort és egy oszlopot rendelünk,

b) a táblázat minden eleméhez egy, a sorához, ill. oszlopához tartozó csomópontok összeköttetéseit jellemző adatot rendelünk, így:

- rövidzárral csatlakozó két csomópont táblázatbeli sor-oszlop találkozásainál elhelyezkedő elembe „1"-et írunk;
- tetszés szerinti K kapcsolóval összekötött két csomópont táblázatbeli sor-oszlop találkozásaihoz K-t írunk;
- ha két csomópont közt csak további csomóponton vagy csomópontokon keresztül lehet összeköttetést létesíteni, akkor a megfelelő táblázatelembe „0" kerül.

c) A táblázat tükrös a főátlóra, ha a hálózat kapcsoló-elemei bilaterálisok (pl. jelfogó kontaktusok) és nem tükrös, ha az elemek közt *unilaterálisok* (pl. dióda) is szerepelnek.

Ezek után, a 9-9b. ábra hálózatát logikailag és topológiailag is jellemző közvetlen összeköttetési táblázat a következőképpen írható fel:

	0	1	2	3
0	1	0	C	A
1	0	1	D	B
2	C	D	1	E
3	A	B	E	1

A híd-típusú hálózatok szintézise a közvetlen összeköttetési táblázaton alapul. Az ezzel kapcsolatos szisztematikus eljárásokkal itt nem foglalkozunk. Csupán megemlítjük, hogy *nem minden* ág-típusú hálózat képezhető le híd-struktúrába és az ezzel kapcsolatos identifikációra is szisztematikus eljárásokat dolgoztak ki.

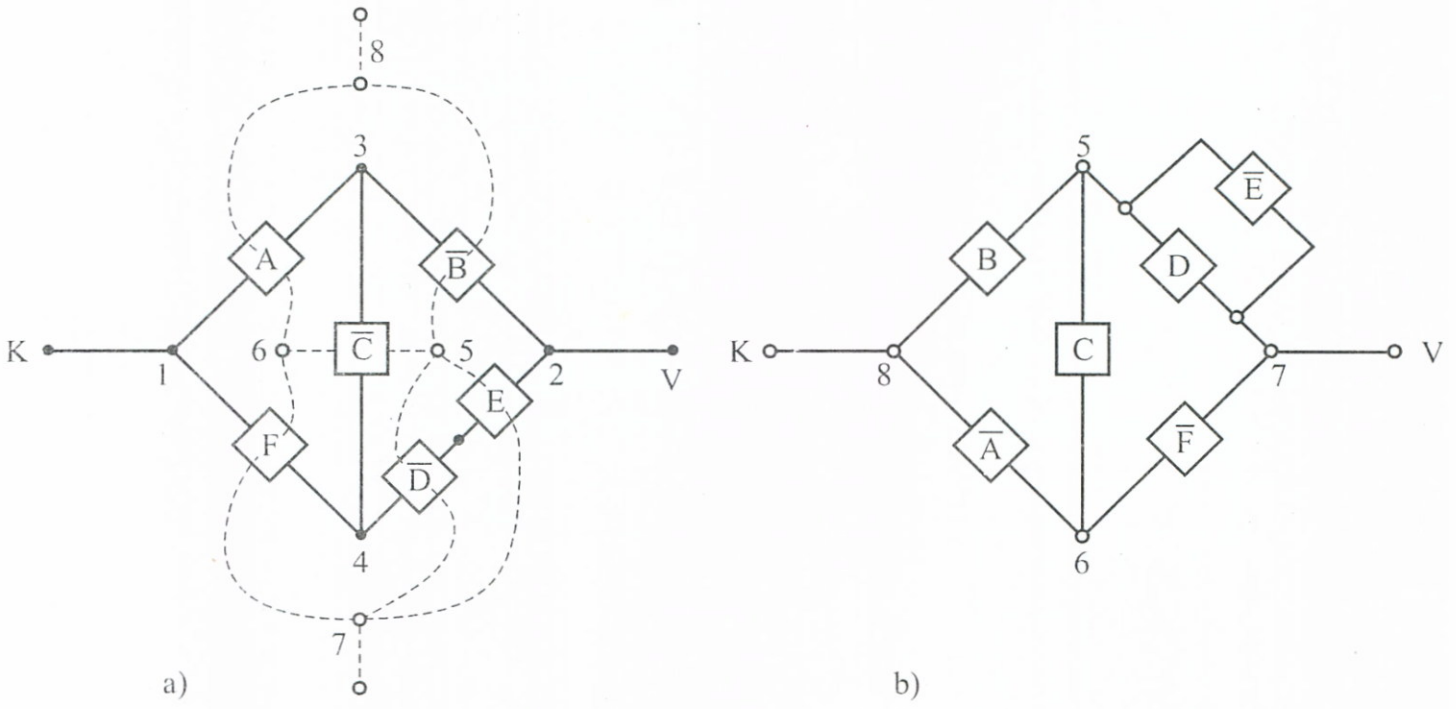
b) Topológiai átalakítások

Az alábbiakban befejezésül röviden bemutatjuk egy példa kapcsán, valamely híd-struktúrában realizált logikai függvény negáltjának topológiai előállítását. Az eljárás a hálózatelméleti „duál-hálózat szerkesztés" módszerén alapul. Eszerint minden soros jellegnek – parallel minden parallelnek – soros, minden huroknak csomópont és minden csomópontnak hurok felel meg az új hálózatban. Továbbmenőleg minden ág-elemnek is ellentétjét (záró-kapcsolónak bontó, bontónak-záró) kell venni a negált hálózatban.

9.3. példa: Adva a 9-10a. ábrán látható híd-struktúra.

I. lépés: Vegyünk fel minden hurokban egy-egy csomópontot és jelöljük meg őket. Itt huroknak számítanak a be- és kimeneti „felezővonal" feletti és alatti térrészek is (9-10a. ábra).

II. lépés: Kössük össze a szomszédos csomópontokat úgy, hogy az összekötő vonalak mindig egy-egy ág-elemen halad-



9-10. ábra *Grafikus topológiai hálózatátalakítás*

janak át. Az így kapott háló lesz a negált hálózat „ágainak" elrendezési képe (9-10a. ábra).

III. lépés: Az új elrendezési kép alapján, rajzoljuk meg úgy a végleges negatív hálózatot, hogy minden ág-elem helyett ellentettjét rajzoljuk be: 9-10b. ábra. Mint látható, a híd-ágakban levő *soros* kapcsolat *párhuzamossá* (és viszont) transzformálódik.

9.2. Küszöb-logikás hálózatok

Az eddigi fejezetek során tárgyalt problémakör a 2. fejezetben bevezetett logikai függvény-definíció köré csoportosult. Ennek szellemében, műszaki feladatainkat elemi-logikai műveletek rendszerében írtuk le és az elemi-logikai műveleteket realizáló alapáramkörök hálózatával realizáltuk. Az így kapott hálózatok – különösen nagyobb változó szám esetén – meglehetősen sok összetevőjük miatt terjengőssé válhattak. Felvetődik a kérdés, vajon lehetne-e olyan absztrakt kifejezésmódot találni, melynél az elemi műveletek koncentráltabb egységekben jelennének meg, és ily módon – miután az ezeket realizáló alapáramkörök is összetettebb részfeladatokat realizálnak – a teljes hálózat kevesebb összetevőből lenne felépíthető. Ilyen koncentráltabb matematikai leírás lehetőségét rejti magában az ún. *„küszöb-elemeken"* alapuló *„küszöb-logikás hálózatok"* bevezetése (threshold logic).

9.2.1. Küszöb-elem, küszöbfüggvény

Egy küszöb-elem elvi felépítését és továbbiakban általunk használandó szimbolikus jelölését a 9-11. ábra mutatja. Az irodalomban másféle jelölésmódok is szerepelnek.

Minden küszöb-elem egy egyszerű küszöbfüggvényt határoz meg.

Az $Y \in \{0, 1\}$ és $X = \{X_1 \dots X_i \dots X_n\}$ (ahol: $X_i \in \{0, 1\}$ és $i = 1, 2 \dots n$) változók közt fennálló alábbi függvénykapcsolatot *egyszerű küszöbfüggvény*nek nevezzük:

$$Y^n = \begin{cases} 1 & \text{ha} \quad \sum_{i=1}^{n} W_i \cdot X_i \geq T \\ 0 & \text{ha} \quad \sum_{i=1}^{n} W_i \cdot X_i < T \end{cases} \qquad (9.4)$$

ahol: – W_i az ún. *„súlytényező”* valós pozitív vagy negatív szám;
– T az ún. *„küszöbérték”* valós pozitív vagy negatív szám,
továbbá: a $\sum$ jel, ill. a $W_i \cdot X_i$ közti „·” jel hagyományos *aritmetikai* összeadási, illetve szorzási műveletek.

Egyszerűsített jelölésmódaraban a továbbiakban írható:

$$Y^n = \left\langle \sum_{i=1}^{n} W_i \cdot X_i \right\rangle_T = \langle W_i \cdot X_i \ldots W_n \cdot X_n \rangle_T$$

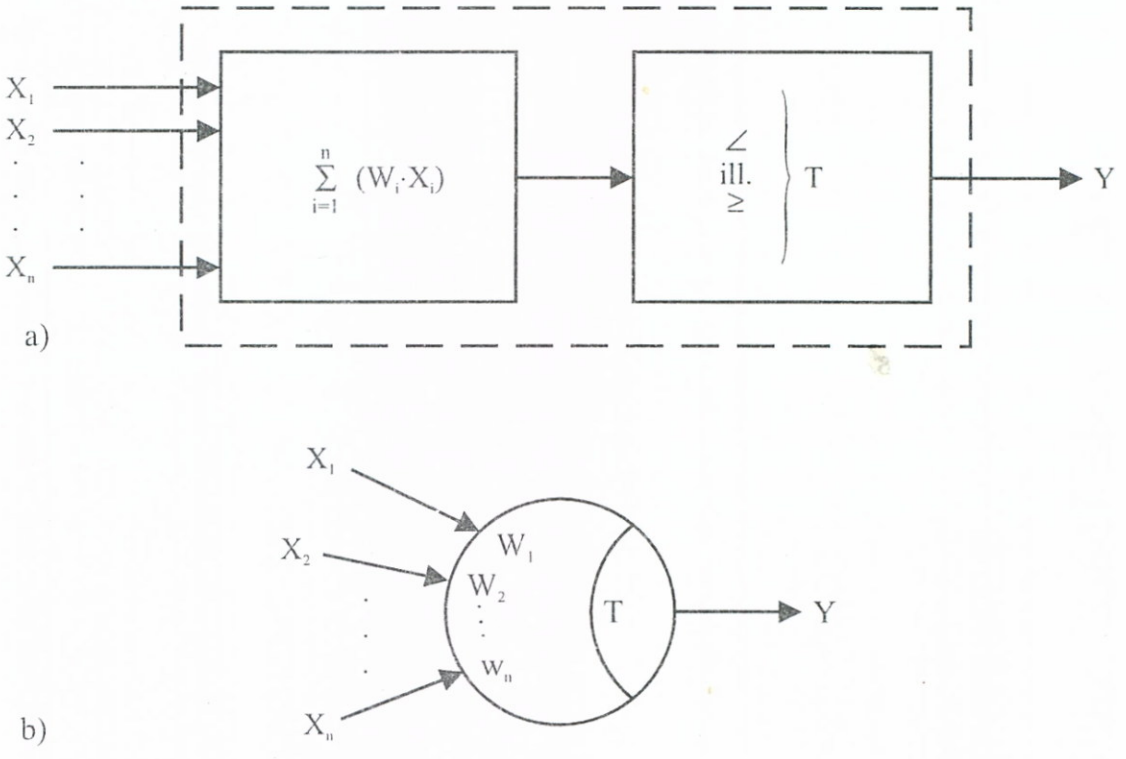
A 9-11. ábra szerinti küszöb-elem „működése” ezek után a következőképpen értelmezhető:

Az X_i független-változók értékei a hozzájuk tartozó i-edik bemeneten a küszöb-elemre jellemző W_i súlytényezőkkel összeszorozva „súlyozódnak”, majd a súlyozott tagok összeadódnak. Az összeadás eredménye összehasonlításra kerül (a küszöb-elemre ugyancsak jellemző) „T” küszöbértékkel. Az összehasonlítás eredménye a (9.4) definíció szerinti egyenlőtlenség értelmében előállítja „Y” függővváltozó konkrét értékét.

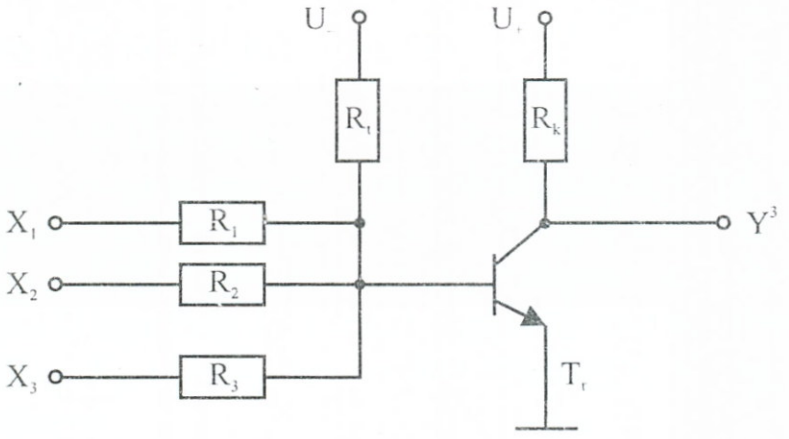
A küszöbfüggvények (ill. elemek) egyszerűsített jellemzésére célszerű bevezetni az alábbi

$$v = \{W_1, \ldots W_{nj}, T\} \tag{9.5}$$

ún. *„súly-küszöbvektor”*-t.



9-11. ábra *Küszöb függvény definíciója és jelölése*



9-12. ábra *Egy realizált küszöb áramkör*

A küszöb-elem előbb vázolt, bonyolultnak tűnő működése áramkörileg meglepő egyszerűen realizálható, többféle fizikai elven is. Egy integrált áramkörös megoldású küszöb-áramkör kapcsolását mutatja a 9-12. ábra.

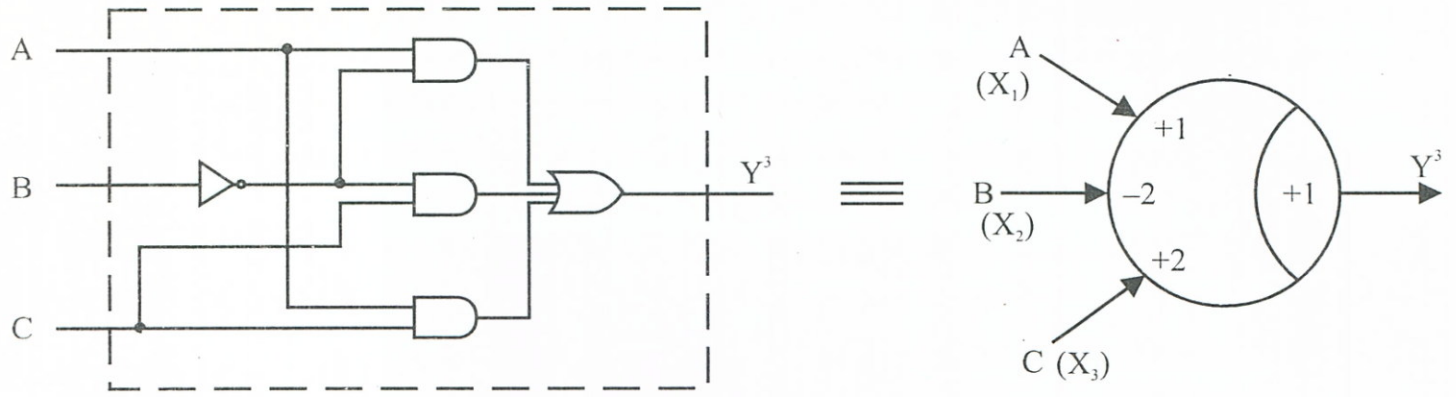
A kapcsolásban, a bemenetek soros R_1, R_2, R_3 óhmos ellenállás-értékei a W_1, W_2, W_3 súlytényezőket, R_T ellenállás és a „$-U_T$" feszültség együttesen a T küszöbértéket határozzák meg. A kapcsolás valójában egy lineáris analóg feszültség összegezést végez a súlyozott bemenetekre, majd egy komparálás után zárja, ill. nyitja „$<, \geq$" relációknak megfelelően *Tr.* tranzisztort.

9.4. példa: Vizsgáljuk meg a következő súly–küszöb vektorral megadott küszöbfüggvényt:

$$v = \{+1, -2, +2; +1\}.$$

A (9.4) definíció alapján felírható az alábbi táblázat.

X_1 W_1=+1	X_2 W_2=–2	X_3 W_3=+2	$W_1 X_1$= $=X_1-2X_2+2X_3$	T	Y_K^n
0	0	0	0		0
0	0	1	+2		1
0	1	0	–2		0
0	1	1	0		0
1	0	0	+1	+1	1
1	0	1	+3		1
1	1	0	–1		0
1	1	1	+1		1



9-13. ábra *BOOLE és küszöbrealizációk összehasonlítása*

A táblázatból megállapítható, hogy küszöbfüggvényünk ekvivalens a következő logikai (BOOLE) függvénnyel:

$$Y^3 = \sum^{3}(1,4,5,7) = \langle 1 \cdot X_1 - 2X_2 + 2 \cdot X_3 \rangle_{T=1}$$

A 9-13. ábrán felrajzoltuk a küszöbfüggvény szimbolikus vázlatát és az ekvivalens logikai függvény algebrailag egyszerűsített alakjának logikai vázlatát is.

Már a 9-13. ábra is érzékelteti, hogy egyetlen küszöb-elem a logikai műveletek *igen nagy koncentrációját* tudja megvalósítani.

9.2.2. Egyszerű és összetett küszöbfüggvények

Egyetlen küszöb-elem egy *egyszerű küszöbfüggvényt* határoz meg. Ha egy $Y^n = F^n(X_1 \ldots X_n)$ logikai függvény jellemzése csak több küszöb-elem együttesével végezhető el, akkor a leírás ún. *összetett küszöbfüggvénnyel* történik. Az összetett küszöbfüggvényt egyszerű küszöbfüggvények rendszere alkotja az alábbiak szerint:

$$\left.\begin{aligned} Y^n &= F^n\left[U_1^n(X_1 \ldots X_n) \ldots U_m^n(X_1 \ldots X_n)\right] \\ Y^m &= F^m(U_1^n \ldots U_m^n) = \left\langle \sum_{j=1}^{m} W_j U_j \right\rangle_T \end{aligned}\right\} \quad (9.6)$$

ahol:
$$U_j^n(X_1 \ldots X_n) = \left\langle \sum_{i=1}^{n} W_{ji} X_i \right\rangle_{T_j}$$

Az alábbiakban néhány később felhasználandó tételt foglalunk össze, bizonyítás nélkül.

a) Ha egy F^n logikai függvény egy $v' = \{W_1', \ldots W_j', \ldots W_n', \ldots T'\}$ súlyküszöb-vektorú egyszerű küszöbfüggvénnyel kifejezhető, akkor kifejezhető a $v = \{W_1 \ldots -W_j, \ldots W_n; (T'-W_j)\}$ vektorú egyszerű küszöbfüggvénnyel. Itt: $T = T'-W_j$ vagy általánosítva több negatív W-re: $T = T'-\sum$ (összes negatív W–k).

b) Az *a)* tételből következik, hogy valamely egyszerű küszöbfüggvénnyel kifejezhető F^n logikai függvény mindig konvertálható egy kizárólag $W_k' > 0$ súly-tényezőket tartalmazó Φ^n egyszerű küszöbfüggvénybe.

c) Valamely F^n logikai függvény és negáltjának súlyküszöb-vektora között a következő rokonság állítható fel:

$$F^n\text{-nél: } v = \{W_1 \ldots W_j, \ldots W_{nj}; T\}$$

$$\overline{F^n}\text{-nél: } v' = \{-W_1 \ldots -W_j, \ldots W_{nj}; -T\}$$

d) Egy egyszerű küszöbfüggvény negáltja ugyancsak egyszerű küszöbfüggvény.

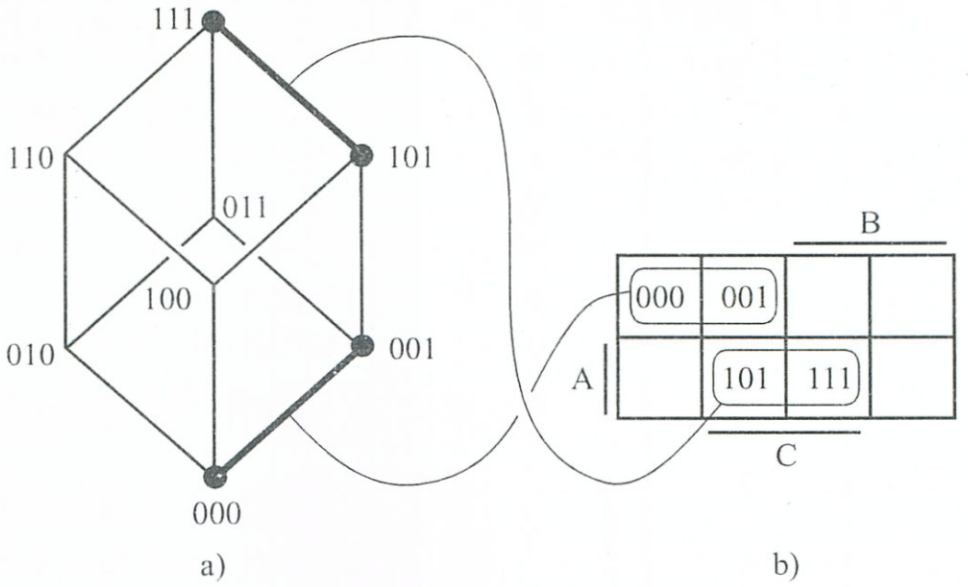
e) Egy logikai függvényt „UNATE"-nak nevezzük, ha diszjunktív irredundáns normál alakjában (tovább már nem egyszerűsíthető normál alakjában) a változók közül egyik sem szerepel negált, illetve nem negált alakban egyidejűleg. *Negatív unate* a függvény, ha az összes független változó negált, *pozitív unate,* ha valamennyi nem negált.

Például az $Y^3 = X_1 \overline{X}_2 + X_2 \overline{X}_3$ függvény X_1-*re* nézve pozitív unate, X_3-*ra* negatív unate, X_2-*re* pedig nem unate. Az $Y_L^3 = \overline{X}_1 + \overline{X}_2 \overline{X}_3$ függvény negatív unate.

f) Minden egyszerű küszöbfüggény unate. Ez a tétel fordítva nem érvényes.

9.2.3. Geometriai ábrázolás és lineáris szeparálhatóság

Az F^n logikai függvényt geometriailag úgy is ábrázolhatjuk, hogy a független változók értékkombinációit egy „n" dimenziós-kocka csúcsaihoz rendeljük és megjelöljük, hogy melyik csúcs jellemez az adott függvénynél $Y = 1$ értékhez tartozó független változó kombinációt. Az „n" dimenziós kockában a szomszédos csúcsokon az értékkombinációk úgy vannak elhelyezve, hogy egymástól csak egy független változó ellentétesre változása révén különböznek. (A Hamming-távolság $H_D = 1$.) Például ilyen elrendezés látható a háromváltozós: $F^3 = \overline{A}\,\overline{B} + AC$ függvényre a 9-14a. ábrán. Az $Y = F^3(A, B, C) = 1$ értékekhez tartozó



9-14. ábra *Geometriai ábrázolás*

csúcsokat bekarikáztuk. A 9-14b. ábra a megfelelő *V–K*-táblát mutatja. Mint látható, itt egy-egy 2-sejtes *tömbnek* a kockán egy-egy *él* ún. *szubkocka* felel meg.

A továbbiak érdekében célszerű bevezetni egy *parciális rendezési relációt,* amelynek alapján az „*n*" dimenziós kocka egyes sarkaihoz rendelt független változók behelyettesítési értékkombinációit „nagyság szerint" rendezhetjük, amennyiben ezek egymással összehasonlíthatók. Eszerint:

$$(a_1, a_2 \ldots a_n) \leq (b_1, b_2 \ldots b_n) \tag{9.7}$$

akkor és csak akkor, ha minden „*i*"-re áll:

$$a_i \leq b_i$$

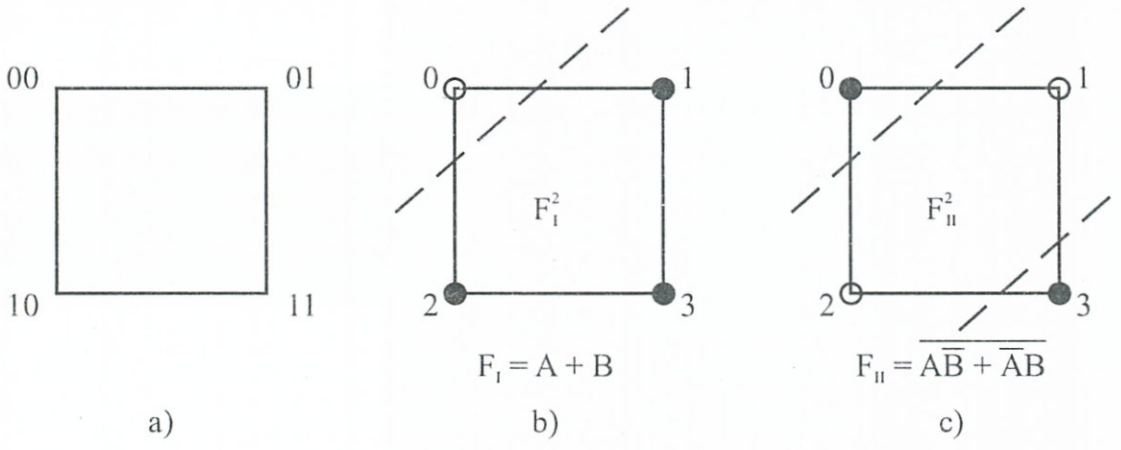
Például a 9-14. ábrán a legkisebb háromváltozós csúcs (0, 0, 0), a legnagyobb (1, 1, 1), továbbá (0, 0, 1) < (1, 0, 1).

Nem összehasonlíthatók egymással a (0, 1, 0) és (1, 0, 0) csúcsok.

Egy F_k^n logikai függvény *unate,* akkor és csak akkor, ha nem tautologia (minden változó kombinációnál „1") és a (9.7) definíció szerinti parciális rendezés létezik olyan formában a csúcsokhoz rendelt minden értékkombináció párra, hogy ha $(a_1 \ldots a_n)$ egy „1"-es csúcs, akkor $(b_1 \ldots b_n) \geq (a_1 \ldots a_n)$ esetén $(b_1 \ldots b_n)$ is egy „1"-es csúcs az F^n függvény „*n*" dimenziós kockán ábrázolt képén.

Ha létezik egy (*n*–1) dimenziós:

$$W_1 X_1 + \ldots + W_n X_n = T$$



9-15. ábra *Hipersík és egyszerű ill. összetett esetek*

ún. *hipersík,* amely az „n" dimenziós kockán átfektetve szétválasztja a kockán ábrázolt F^n függvény „1" értékű csúcsait a „0"-ásoktól, akkor az F^n függvény *lineárisan szeparálható,* és *egyszerű küszöbfüggvénnyel* leírható.

9.5. példa: Egy kétdimenziós kocka látható a 9-15a. ábrán. Két konkrét függvényt ábrázoltunk a *b)* és *c)* ábrákon:

$$F_I^2 = E_1^2 + E_2^2 + E_3^2 = A + B \qquad \text{(9-15b. ábra)}$$

$$F_{II}^2 = E_0^2 + E_3^2 = \overline{A}\overline{B} + AB \qquad \text{(9-15c. ábra)}$$

Mint látható, a *b)* ábra esetében átfektethető egyetlen elválasztó, $n-1 = 1$ dimenziós „hipersík" a befeketített „1" tényezőjű és az üres körrel jelölt „0" tényezőjű csúcsok között, míg a *c)* ábránál ezt a szeparálást egyetlen síkkal nem lehet elvégezni. Következésképpen a *b)* eset egyszerű küszöbfüggvénnyel, a *c)* eset csak összetett küszöbfüggvénnyel írható le. Minthogy pedig a fenti függvények irredundáns diszjunktív normál alakok, látható, hogy F_I^2 unate, míg F_{II}^2 nem unate.

9.2.4. Identifikációs algoritmus

Eddigi eredményeink alapján már megadható egy olyan eljárásnak az algoritmusa, amelynek segítségével felírhatunk egy adott $F^n(X_1 \ldots X_n)$ logikai függvénynek megfelelő elemi küszöbfüggvényt, amennyiben ilyen létezik. Az eljárás lépései a következők:

I. lépés: $F^n(X_1 \ldots X_n)$ diszjunktív irredundáns alakjának előállítása.

II. lépés: Unate-vizsgálat. Ha a függvény nem unate, akkor a függvény lineárisan nem szeparálható.

III. lépés: Ha a függvény unate, akkor a diszjunktív irredundáns alakot konvertáljuk egy minden változójában pozitív Φ^n függvénybe, melynek minden súlyozási tényezőjére áll, hogy: $W_i' > 0$. Ezt a konvertálást a későbbi lépések érdekében végezzük el.

IV. lépés: Megkeressük a hipersík helyét, mely az $F^n = 1$ és $F^n = 0$ értékekkel jellemzett csúcsok mezői között helyezkedik el az „n" dimenziós kockában. Ehhez kikeressük a (9.7)-re és az *e)* tételre támaszkodva a minimális „1"-es és a maximális „0"-ás csúcsokat, mivel a hipersík ezek közt kell, hogy elhelyezkedjen. A talált „p" darab legnagyobb „1"-es és „q" darab legkisebb „0"-ás csúcs $p \times q$ darab egyenlőtlenséget eredményez, melyekből Φ_K^n kizárólag *pozitív* előjelű W_i' tényezői és T' küszöbértéke meghatározhatók, így $\nu\Phi$ felírható.

V. lépés: Φ_K^4 visszakonvertálása (az ismert W_i' és T' értékek birtokában) F^n-be az *a)* és *b)* tételek alapján. A kapott eredmény már a keresett elemi küszöbfüggvény ν_F vektora.

9.6. példa: Vizsgáljuk meg az alábbi logikai függvényt, és ha elemi küszöbfüggvénnyel leírható, határozzunk meg egy megfelelő ν vektort.

$$Y^4 = F_K^4(A,B,C,D) = \sum^4(0,1,3,4,5,6,7,12,13)$$

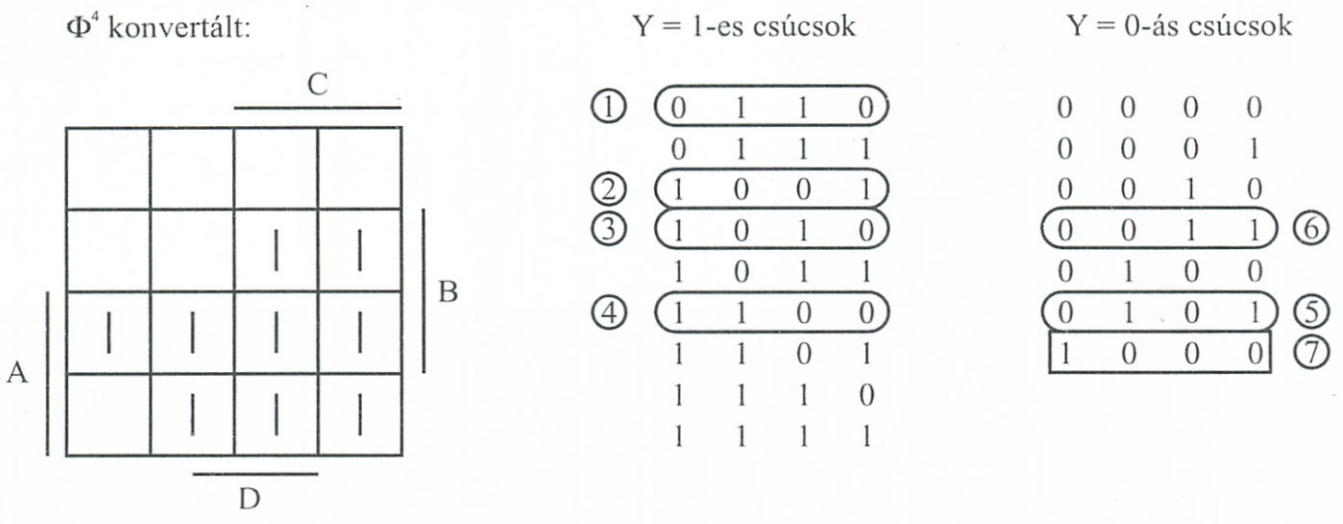
I. lépés: *V–K* táblán minimalizálva a diszjunktív irredundáns alak:

$$Y_K^4 = \overline{A}B + \overline{A}D + B\overline{C} + \overline{A}\,\overline{C}$$

II. lépés: A függvény unate mindegyik változóra leolvashatóan (A és C mindenhol negált, B és D csak eredeti formában) szerepel:

III. lépés: Konvertálás csupán negálatlan változókat tartalmazó Φ_K^4 függvénybe:

Mivel A és C negáltak, ezeknél kell változtatni, így a konvertált függvény:



9-16. ábra *„1"-es és „0"-ás csúcsok szétválasztása*

$$\Phi_K^4 = AB + AD + BC + AC$$

IV. lépés: Hipersík helyének megkeresése. Ehhez megkeressük az „1"-es csúcsok és a „0"-ás csúcsok halmazai közt elhelyezkedő határzónát, az n-1 = 4-1 = 3 dimenziós hipersíknak ebben kell elhelyezkedni. A határzóna széleit a (9.7) szerinti parciális rendezésnek megfelelően a legkisebb „1"-es csúcsok és a legnagyobb „0"-ás csúcsok képezik. A kisebb-nagyobb összehasonlítást a (9.7) egyenlőtlenséggel végezzük. Az össze nem hasonlítható csúcsokat külön felsoroljuk. Példánknál a 9-16. ábrán a *V–K* táblából felírtuk az „1"-es és „0"-ás csúcsokat.

A *legkisebb „1"-es csúcsok* azok lesznek az $Y = 1$-esek közül, melyekben a legkevesebb „1"-es van:

$$\rightarrow \underset{①}{(0,1,1,0)}\ \underset{②}{(1,0,0,1)}\ \underset{③}{(1,0,1,0)}\ \underset{④}{(1,1,0,0)}$$

A *legnagyobb „0"-ás csúcsok* azok lesznek az $Y = 0$-ások között, amelyekben a legtöbb „1"-es van, illetve ide soroljuk az össze nem hasonlíthatókat is. Ez utóbbiak azok lesznek, amelyek a többi 0-ástól $D > 1$ Hamming-távolságra vannak:

$$\Rightarrow \begin{cases} \underset{⑤}{(0,1,0,1)}\ \underset{⑥}{(0,0,1,1)} & \text{– legtöbb 1-es} \\ \underset{⑦}{(1,0,0,0)} & \text{– össze nem hasonlítható egyik } Y = 0\text{-ással sem} \end{cases}$$

A $p = 4$ darab $\rightarrow$ és $q = 3$ darab $\Rightarrow$ jelű csúcsot összehasonlítva, $p \times q = 12$ darab egyenlőtlenség írható fel: Az ①-esen például (0, 1, 1, 0) a B és C változók helyén van 1-es, így ez a csúcs a W_B' és W_C'-re ad tájékoztatást. Azaz, írható

$$\left.\begin{array}{ll} \rightarrow ① & W_B' + W_C' \\ ② & W_A' + W_D' \\ ③ & W_A' + W_C' \\ ④ & W_A' + W_B' \end{array}\right\} > \left\{\begin{array}{ll} W_B' + W_D' & ⑤ \Leftarrow \\ W_C' + W_D' & ⑥ \\ W_A' & ⑦ \end{array}\right.$$

Ezekből közvetlenül felírhatók a következők:

$$\left.\begin{array}{lll} W_A' > W_D' & W_C' > W_D' & W_B' > 0 \\ W_A' > W_B' & W_B' > W_D' & W_C' > 0 \\ W_A' > W_C' & & W_D' > 0 \end{array}\right\} \qquad (9.8)$$

(9.8)-ból W_A', W_B', W_C', W_D'-re sok megoldás adható meg. A realizáció szempontjából általában a *legkisebb egész* értékekre célszerű törekedni. (9.8)-ból látható, hogy W_D', a legkisebb, ezt vegyük fel $W_D' = 1$-nek, akkor felírhatók a konvertált Φ^4 függvény súlytényezői:

$$W_D' = 1. \quad W_C' = W_B' = 2 \quad W_A' = 3$$

A Φ^4-hez tartozó T' küszöbérték meghatározásához behelyettesítjük a súlyértékeket $\rightarrow$ és $\Rightarrow$ „legkisebb", ill. „legnagyobb" csúcsokba, a „T'" érték ezek súlyozott összege közt kell, hogy elhelyezkedjen:

$\rightarrow$ ① csúcs: (0, 1, 1, 0) súlyozott összege: $W_B + W_C = 2 + 2 = 4$

$\Rightarrow$ ⑦ csúcs: (1, 0, 0, 0) súlyozott összege: $W_A = 3$

Innen: $4 \geq T' > 3$

Választunk: $T' = 3{,}5$-öt

Végül felírható Φ^4 súly-küszöb vektora:

$$v_\Phi = \{3, 2, 2, 1; 3, 5\}$$

V. lépés: Φ^4 visszakonvertálása F^4-be *a)* és *b)* tételek alapján. A *III. lépés*ben X_i-ből X_i-be konvertált változók súlytényezőit itt a visszakonvertálásnál negatív előjellel látjuk el (példánknál A és C változók), a többiek pedig változatlanul maradnak. Így F^4 súlytényezői:

$$W_D = 1, \; W_C = -2, \; W_B = 2, \; W_A = -3$$

A konvertált küszöbérték az 1. tétel szerint (W_l-ekkel jelöljük a negatív súlytényezőket).

$$T = T' - \sum W_l = 3{,}5 + W_A + W_C = 3{,}5 - 2 - 3 = -1{,}5$$

A kiinduló F^4 függvényt leíró elemi küszöbfüggvény *súly-küszöb vektora* végül:

$$v_F = \{-3, 2, -2, 1; -1{,}5\}$$

azaz:

$$F^4 = \sum^{4} (0, 1, 3, 4, 5, 6, 7, 12, 13) = < 3A + 2B - 2C + D >_{-1{,}5}$$

9.2.5. Összetett küszöbfüggvények szintézise

Egy tetszés szerinti F^n logikai függvény küszöb-logikás leírása, mint említettük, egyszerű vagy összetett küszöbfüggvényekkel lehetséges. Az első esetre vonatkozóan már az előző pontban megismertünk egy algoritmust. Most az összetett küszöbfüggvényes leírás lehetőségét vizsgáljuk meg a *V–K* táblák felhasználásával.

A cél a minimális számú küszöb-elemet tartalmazó megoldás keresése.

Az eljárás során valójában az F^n függvény küszöb-elemes dekompozíciójára kerül sor, mely a következő tényekre épül:

A küszöbfüggvények eddig megismert tulajdonságai alapján bebizonyítható, hogy a *V–K* táblákon definiálhatók olyan *megengedett alakzatok, melyek elemi küszöbfüggvényekkel leírhatók.*

a) $n = 3$-változós esetre a megengedett alakzatok típusait a 9-17a. ábra foglalja össze, egy tipikus $n = 4$-változós alakzat látható a 9-17b. ábrán.

b) Mindegyik alakzat bármely pozícióban elhelyezkedhet a táblán, ha alapvető topológiai struktúráját megtartja.

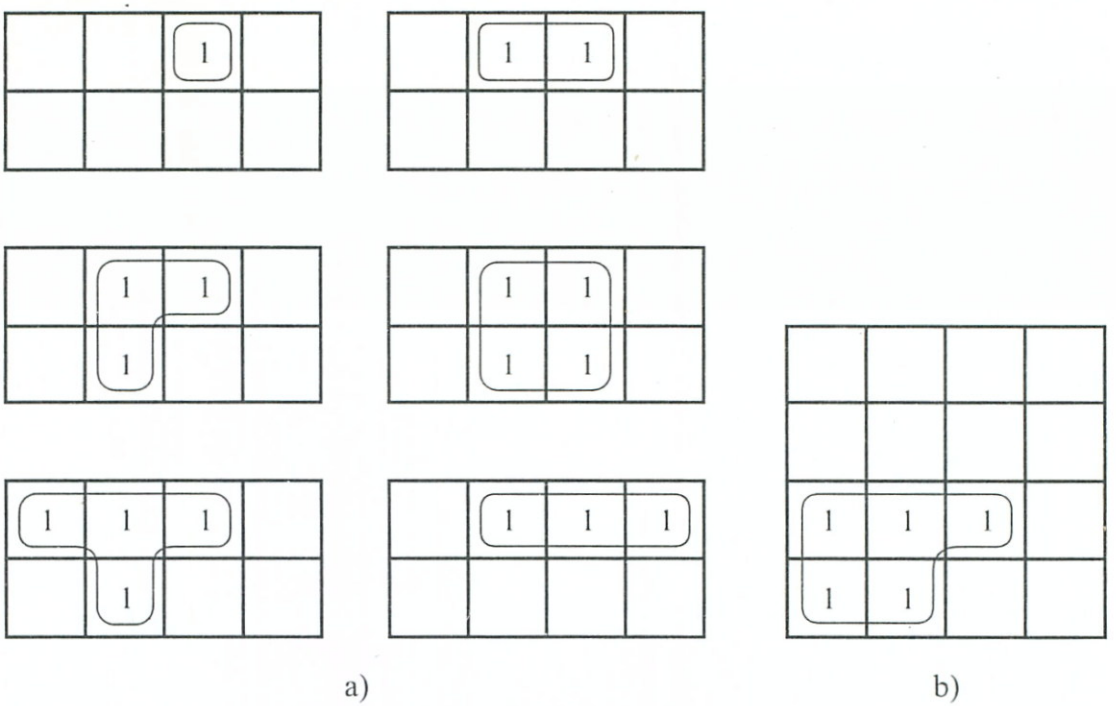
c) Alacsonyabb változó számhoz rendelt, megengedett alakzatok magasabb változószám esetén is megengedett alakzatok.

d) Mivel egy elemi küszöbfüggvény negáltja is elemi küszöbfüggvény, így a „0" cellák által képzett alakzatok is megengedettek.

Az eljárás fő lépései ezek után a következők:

I. lépés: *Diszjunktív irredundáns alak előállítása.*

II. lépés: A kiinduló F^n BOOLE-függvény unate vizsgálata. Ha a függvény unate, úgy egyszerű küszöbfüggvény előállítása következik és a 9.2.4. pontban ismertetett eljárást alkalmazzuk. Ha a függvény nem unate, akkor következik a

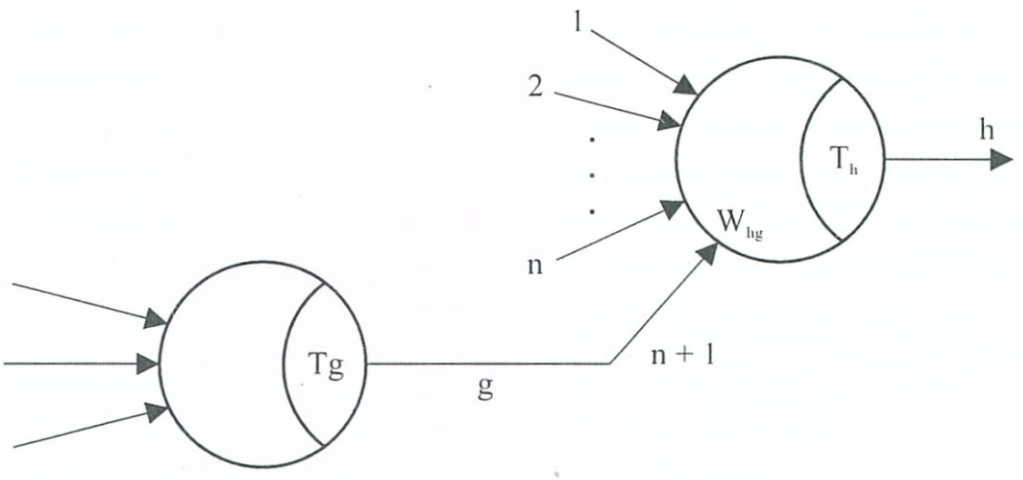


9-17. ábra *Megengedett alakzatok*

III. lépés: A függvényt a *V–K* táblán megengedett alakzatokkal fedjük le, és majd ezen alakzatokhoz tartozó elemi küszöbfüggvények kaszkád rendszerével írjuk le (9-18. ábra).

IV. lépés: A megengedett alakokhoz tartozó „*n*" változós elemi küszöbfüggvények felírása (9.2.4. pont).

V. lépés: A kaszkád elrendezés „meghajtott" „*h*"-elemének a „meghajtó" „*g*"-elemmel csatlakozó, (*n* + 1)-edikként felvett bemenetéhez tartozó W_{hg} súlytényező megállapítása. Ezt olyan nagyra kell választani, hogy ha a „*g*"-elem kimenete „1" értékű lesz, ennek hatása feltét-



9-18. ábra *Kaszkád összetett küszöbfüggvény*

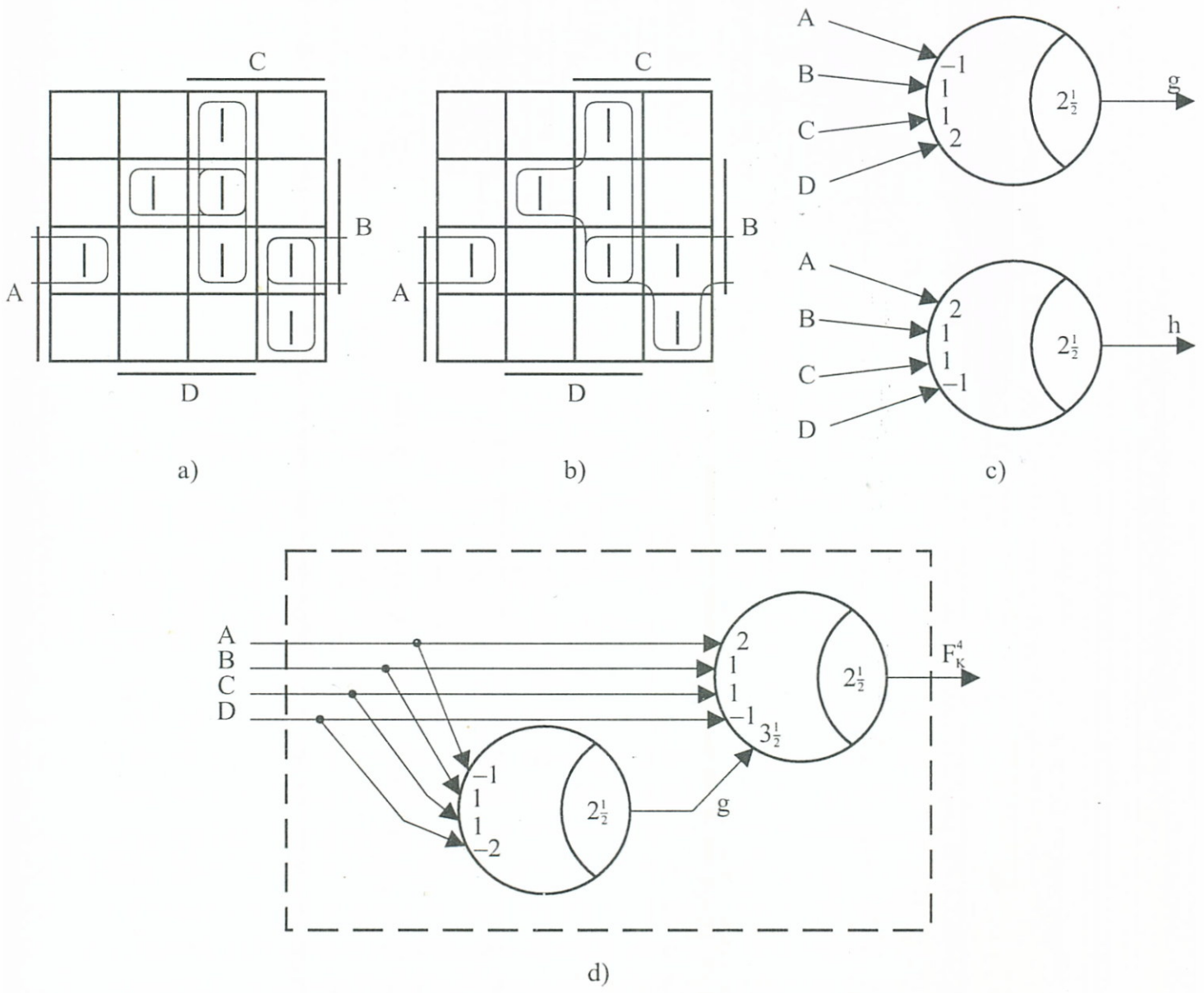
lenül átjuthasson a „h" elem kimenetéhez. Ezért, ha „W_h" $_{\text{negatív}}$-val jelöljük a „h" elem negatív súlytényezőinek abszolút értékeit, első közelítésben felírható:

$$W_{hg} = T_h + \sum W_{h\ \text{negatív}} \tag{9.9}$$

9.7. példa: Vizsgáljuk meg a következő logikai függvényt, és írjuk fel küszöblogikás megfelelőjét:

$$Y^4 = F^4(A,B,C,D) = \sum^{4} (3, 5, 7, 10, 12, 14, 15)$$

I. lépés: *Diszjunktív irredundáns alak előállítása. A függvény V–K* tábláját a 9-19a. ábra mutatja.



9-19. ábra *Összetett küszöbfüggvény realizációja*

II. lépés: Unate-vizsgálat. A diszjunktív irredundáns alakból látható, hogy a függvény nem unate, mivel „A" és „D" is szerepelnek negáltan és nem negáltan is:

$$Y^A = AB\overline{D} + \overline{A}BD + \overline{A}CD + BCD + AC\overline{D}$$

III. lépés: Lefedés megengedett alakzatokkal. A lefedés 2 darab, a 3-változós táblán már definiált alakzattal is elvégezhető, ezek itt is megengedettnek számítanak (9-19b. ábra).

IV. lépés: A két megengedett alakhoz tartozó egyszerű küszöbfüggvényeket a 9.2.4. pont algoritmusa szerint tudjuk meghatározni. Ezt itt nem részletezve a kapott eredmény:

$$\nu_g = \left\{-1, 1, 1, 2; \quad 2\frac{1}{2}\right\}$$

$$\nu_h = \left\{2, 1, 1, -1; \quad W_{hg}; \quad 2\frac{1}{2}\right\}$$

A küszöb-elemeket a 9-19c. ábra mutatja.

V. lépés: W_{hg} meghatározása. Előző pontból: $T_h = 2\frac{1}{2}$, $\sum W_{h\,\text{negatív}} = |-1| = 1$. Behelyettesítve (9.9)-be kapjuk:

$$W_{hg} = T_h + \sum W_{h\,\text{negatív}} = 2\frac{1}{2} + 1 = 3\frac{1}{2}$$

Ezt beírva az előző lépés ν_h-jába már megkapjuk a keresett *összetett küszöbfüggvényt,* mely a 9-19d. ábrán látható.

9.2.6. Összefoglalás

Az elmondottak alapján belátható, hogy a küszöblogikás kifejezésmód a műveletek nagyobb koncentrációját teszi lehetővé, így végeredményben egyszerűbb hálózat felépítést eredményez. Ebben rejlik elsősorban e rendszerek előnyös volta. Elterjedésük mégsem olyan gyors és általános, mint várható lenne, ennek oka a jelenleg még fennálló számos, hátrányos kötöttség.

Az egyik fő problémát a realizációs áramkörökkel szemben támasztott fokozott pontossági igény jelenti, mivel a kapcsolások fokozottabban érzékenyek az áramköri paraméterek megváltozására. Így például a 9-12. ábránál az R_1, R_2, R_3 ellenállások szórása a súlyténye-

zők megváltozását, az R_T, $-U_T$ és a T_r tranzisztor bemenő paramétereinek elvándorlása a küszöbpont elcsúszását eredményezheti. Problémák jelentkeznek nagyobb bemenetszámnál is.

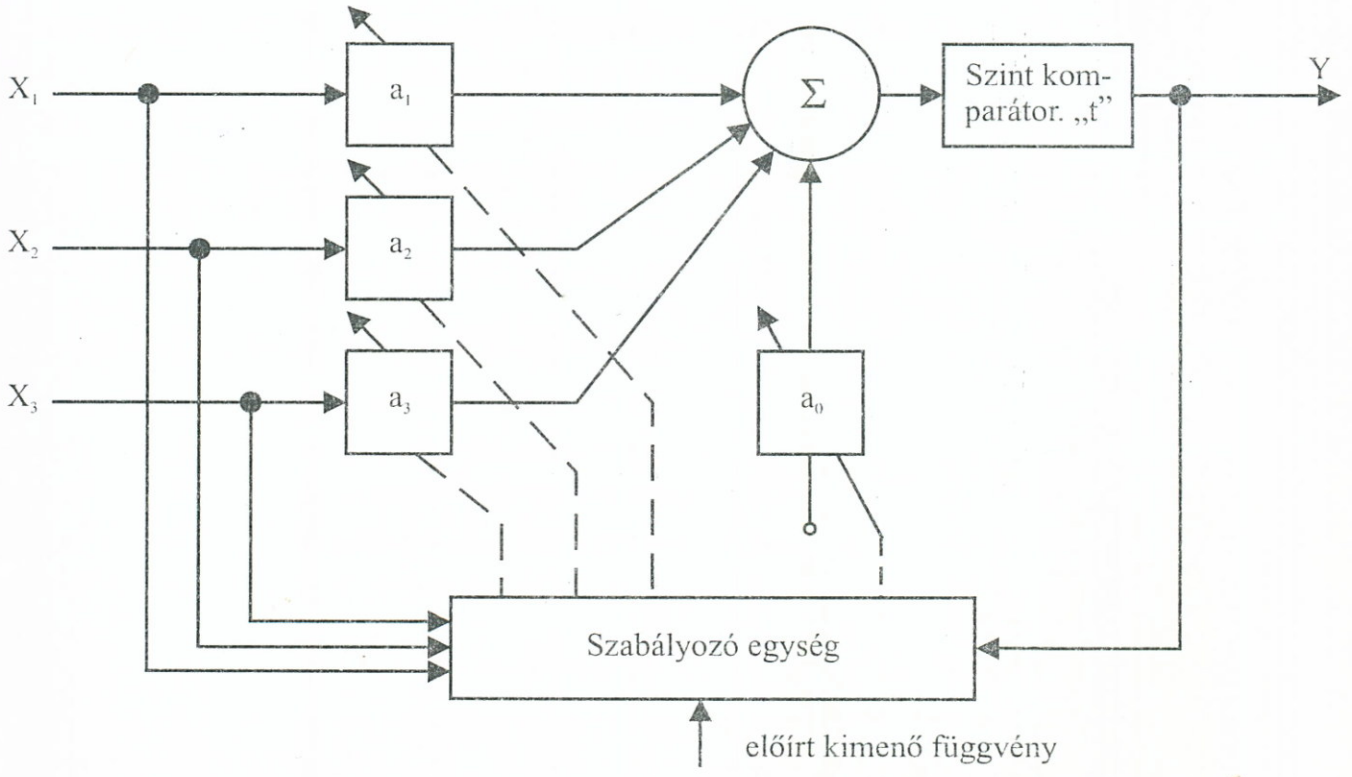
Másik probléma, a küszöblogikák tervezéséhez a hatékony szintézis-eljárások jelenleg matematikailag még nem minden részében tisztázott volta.

Napjainkban mindkét alapproblémával kapcsolatban intenzív kutatás folyik.

IC-s megoldásban már léteznek küszöblogikás rendszerek. MOS-tranzisztoros felépítéssel kidolgoztak olyan megoldásokat, ahol a W_i súlytényezők és a T' küszöbérték külső vezérléssel változtathatók, így egy küszöb-elem többféle logikai függvényt is realizálhat. Egy kapacitív analóg súlyozó tényezőkkel működő megoldást mutat a 9-20. ábra.

Egy szabályozó-áramkör, az előírt kimeneti függvénynek megfelelően „a_i" kapacitív súlyozó tényezők feszültségét állítja be egy meghatározott adaptációs időszak alatt. Az „a_0" *tényező a* $T = t - a_0$ küszöb beállítására szolgál. Az előállított küszöbfüggvény:

$$Y = \begin{cases} 1 & \text{ha: } a_1 X_1 + \ldots a_3 X_3 \geq t - a_0 \\ 0 & \text{ha: } a_1 X_1 + \ldots a_3 X_3 < t - a_0 \end{cases}$$



9-20. ábra *Programozható MOS-IC küszöbfüggvények*

9.3. Majoritás és minoritás-logikák

A küszöblogikák speciális osztályának tekinthetők a *majoritás* (más néven: többségi) és *minoritás* (más néven: kisebbségi) logikák, melyek majoritás-, ill. minoritás-kapukból épülnek fel.

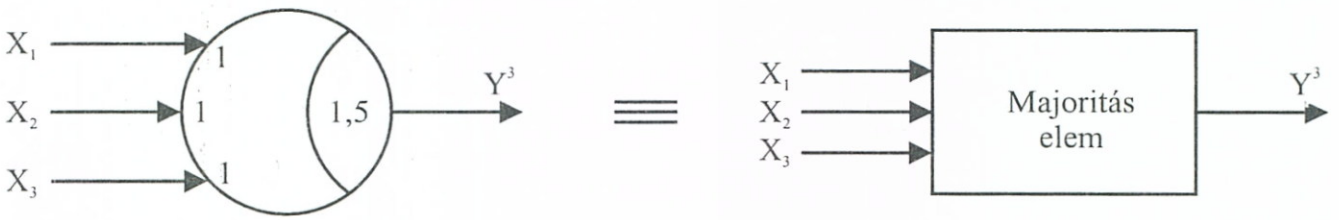
A majoritás (minoritás) kapu egy olyan egyszerű logikai hálózat, melynél a kimeneti „*Y*" akkor „1" értékű, ha a bemeneti X_1 ... X_n-eknél a többség (kisebbség) „1"-es értéket vesz fel.

Mivel „többség" (kisebbség) legalább három résztvevő esetén definiálható használhatóan, ezért a majoritás (minoritás)-logikákat igen gyakran 3-bemenetű elemi kapukból építik fel.

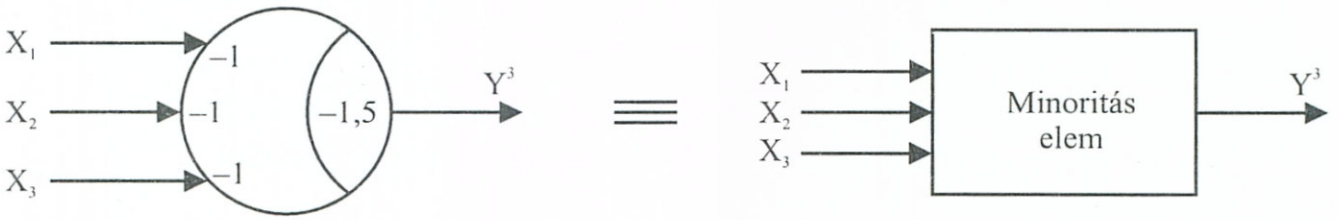
A majoritás, ill. minoritás-kapukhoz önálló logikai műveleteket definiálhatunk, és segítségükkel funkcionálisan teljes rendszerek is felépíthetők. Egyes irodalmak külön algebrákat is tárgyalnak.

9.8. példa: Vizsgáljuk meg a 9-21a. ábrán látható logikai táblázatot majoritás és minoritás szempontjából.

a)



b)



c)

X_1	X_2	X_3	F^3_{Ma}	F^3_{Mi}
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

9-21. ábra *Majoritás és minoritás logikák*

Megállapítható, hogy

$$v_{Ma} = \{1, 1, 1; 1{,}5\}$$
$$v_{Mi} = \{-1, -1, -1; -1{,}5\}$$

súlyküszöb-vektorok mellett a 9-21b., ill. c. ábrák küszöb-elemei adhatók meg, melyek egy majoritás, ill. minoritás-elemnek felelnek meg.

Viselkedés szempontjából hasonló tulajdonságú kapukat kaphattunk volna az:

$$Y = F^3_{(2)(3)}(X_1, X_2, X_3) = F^3_{Ma}$$
$$Y = F^3_{(0)(1)}(X_1, X_2, X_3) = F^3_{Mi}$$

szimmetrikus függvények realizációja révén is.

9.4. Többértékű logikák

A kettőnél több állapotú kapcsoló-eszközökből felépíthető rendszerek kérdéskörével már évtizedek óta foglalkoznak és velük kapcsolatosan – különösen elméleti vonatkozásban – figyelemreméltó eredményeket értek el. Miután a komplemens elem meghatározása többféle értelmezésben megközelíthető, ezért többféle algebra is kialakult az idők folyamán (Post, Lukasiewitz stb.). E fejezet keretében (elsősorban bevezető célzattal) csak néhány, műszaki alkalmazás szempontjából számításba jöhető alapfogalmat tárgyalunk.

9.4.1. A POST függvény fogalma és a vele kapcsolatos műveletek

A 2. fejezetben megismert kétértékű logikai függvény fogalmának egy általánosított változatához jutunk a többértékű változókon felépülő, ún. *„POST-függvény"* bevezetésével.

Az $Y = F^n(X_1 \ldots X_i \ldots X_n)$ „n"-változós *Post-függvény* a $\{0, 1 \ldots (R-1)\}$ halmaz „n"-szeres Descartes-szorzatát egyértelműen a $\{0, 1, \ldots (R-1)\}$ halmazba képezi le. Fennáll:

$$X_i \in \{0, 1 \ldots (R-1)\} \; (i = 1, 2 \ldots n)$$
$$Y \in \{0, 1 \ldots (R-1)\}$$

A képezhető POST-függvények száma: $N = R^{R^n}$

Mint látható, a POST-függvények „X_i", „Y" változói több lehetséges diszkrét értéket vehetnek fel, ily módon az ún. *„többértékű logikák"* leírására szolgálnak.

A POST-függvények $R = 2$ esetén a 2. fejezetbeli logikai függvényekbe (BOOLE-függvényekbe) mennek át.

Operátorok, komplemens, algebrai összefüggések

A többértékű függvényekben szereplő többértékű mennyiségek algebrája eltér a már tárgyalt bináris algebráktól. A fő problémát a *komplemens* kérdésköre jelenti, miután ez „R" érték esetén többféleképpen is definiálható. A többféle lehetséges definíciónak megfelelően az idők folyamán különféle algebrák alakulhattak ki, mint ezt a bevezetőben már említettük.

a) Operációk és tulajdonságaik

Az $A, B \ldots K$ ($A, B, \ldots K \in \{0, 1 \ldots (R-1)\}$ többértékű mennyiségekre definiálhatók a következő operációk:

$$\begin{aligned} \text{Min}\,(A, B \ldots K) &= A \cdot B \cdot \ldots K \\ \text{Max}\,(A, B \ldots K) &= A + B + \ldots + K \end{aligned} \qquad (9.10)$$

9.9. példa: A 9-22. ábrán szemléltetésül megadtuk $R = 3$ értékű két mennyiség esetére a Min (A, B) és Max (A, B) operátorokat.

A	B	Min (A, B)	Max (A, B)
0	0	0	0
0	1	0	1
0	2	0	2
1	0	0	1
1	1	1	1
1	2	1	2
2	0	0	2
2	1	1	2
2	2	2	2

9-22. ábra *3-értékű igazságtáblázat Min és Max műveletekre*

A (9.10) definícióval kapcsolatban fennállnak a következő tulajdonságok:

$$
\begin{array}{lllll}
\text{I.} & A\cdot B & = B\cdot A & \text{Kommutativitás} & \\
 & A+B & = B+A & & \\
\text{II.} & A\cdot B\cdot C & = A\cdot(B\cdot C) & \text{Asszociativitás} & \\
 & A+B+C & = A+(B+C) & & \\
\text{III.} & A\cdot(B+C) & = A\cdot B+A\cdot C & \text{Disztributivitás} & \\
 & A+(BC) & = (A+B)\,(A+C) & & \\
\text{IV.} & A\cdot(R-1) & = A & \text{Egység} & \text{elem} \\
 & A+0 & = A & \text{Nulla} & \text{létezése}
\end{array}
\qquad (9.11)
$$

b) Komplemensek

A komplemens fogalmára több definíciót adunk meg.

1. Komplemens:

$$\overline{A} = (R-1) - A \qquad (9.12)$$

ahol: a mínusz jel *aritmetikai* kivonást jelent és: $A \in \{0, 1 \ldots (R-1)\}$

2. Komplemens:

Vezessük be a következő „unáris operátort"

$$\begin{matrix} a, b \\ X \end{matrix} = \begin{cases} R-1 & \text{ha: } a \le X \le b \\ 0 & \text{egyébként} \end{cases} \qquad (9.13)$$

ahol: $X, a, b \in \{0, 1 \ldots (R-1)\}$
és: $a \le b$ fix konstans-értékek.
Ezután a komplemens:

$$\overline{\begin{matrix} a, b \\ X \end{matrix}} = \begin{cases} 0 & \text{ha: } a \le X \le b \\ R-1 & \text{egyébként} \end{cases} \qquad (9.14)$$

További, az irodalomban tárgyalt komplemens képzésnél használható műveletek, pl. a POST-féle „ciklikus tagadás" és „forgatás" műveletek, melyek segítségével külön algebrák építhetők fel. Jelen vizsgálódásainkhoz ezek nem szükségesek, így velük részletesebben nem foglalkozunk.

A következőkben fenti két komplemens felhasználásával építjük fel algebrai rendszerünket, mégpedig oly módon, hogy az 1. típust *konstans*-értékekre, míg a 2. típust *változókra* definiáljuk.

A 2. típussal kapcsolatban bevezetett $\begin{matrix} a,b \\ X \end{matrix}$ valójában *kétértékű* (bináris) változó, annak ellenére, hogy maga „X" „R"-értékű változó. Ily módon az $\begin{matrix} a,b \\ X \end{matrix}$ kétértékű változóra felhasználhatjuk a *kétértékű BOOLE-algebra* már megismert tételeit is a továbbiakban.

További összefüggéseket foglaltunk össze a 9-23. ábrán.

a	$\overline{X^{a,b}} = X^{0,a-1} + X^{b+1,R-1}$ ahol: $X^{0,a-1} = 0$, ha: $a = 0$ $X^{b+1,R-1} = 0$, ha: $b = R-1$
b	$X^{0,R-1} = R-1$
c	$X^{a,b} + X^{\overline{a,b}} = R-1$
d	$X^{a,w} + X^{y,b} = \begin{cases} X^{a,b} & \text{ha: } a \le y \le w \le b \\ X^{y,b} & \text{ha: } y \le a \le w \le b \\ X^{w+1,y-1} & \text{ha: } a \le w \le y \le b \end{cases}$
e	$X^{a,w} + X^{y,b} = \begin{cases} 0 & \text{ha: } a \le w \le y \le b \\ X^{y,w} & \text{ha: } a \le y \le w \le b \\ X^{y,b} & \text{ha: } a \le y \le b \le w \end{cases}$
f	DE MORGAN: $\overline{C \cdot X_1^{a_1,b_1} \cdot X_2^{a_2,b_2} \ldots X_n^{a_n,b_n}} = \overline{C} + X_1^{\overline{a_1,b_1}} + X_2^{\overline{a_2,b_2}} + \ldots + X_n^{\overline{a_n,b_n}}$ $\overline{C + X_1^{a_1,b_1} + X_2^{a_2,b_2} + \ldots + X_n^{a_n,b_n}} = \overline{C} \cdot X_1^{\overline{a_1,b_1}} \cdot X_2^{\overline{a_2,b_2}} \cdot \ldots \cdot X_n^{\overline{a_n,b_n}}$
g	$\text{Min}(X_1, X_2 \ldots X_n) = X_1 \cdot X_2 \ldots X_n$ $\text{Max}(X_1, X_2 \ldots X_n) = X_1 + X_2 + \ldots + X_n$ a, b, y, w, X_i, C $\in \{1, 2 \ldots R-1\}$

9-23. ábra *Az unáris operátorral kapcsolatos műveletek*

Σ és Π ALAKOK. TERM TÁBLÁK

a) Teljesen meghatározott függvények

Minden többértékű logikai függvény előállítható a következő kanonikus alakban:

$$Y = \sum_K \alpha_K \cdot X_1^{a_{K_1}, a_{K_1}} \cdot X_2^{a_{K_2}, a_{K_2}} \ldots X_n^{a_{K_n}, a_{K_n}} = \sum_K \alpha_K \cdot E_K^n \qquad (9.15)$$

ahol: $\alpha_K \in \{0, 1 \ldots (R-1)\}$ konstans

E_K^n – egy n változós „Min-term".

Komplemense pedig felírható az alábbi módon:

$$\overline{Y} = \prod_K (\overline{\alpha}_K + X_1^{\overline{a_{K_1}, a_{K_1}}} + \ldots + X_n^{\overline{a_{K_n}, a_{K_n}}}) \qquad (9.16)$$

Itt: $\overline{\alpha}_K$ – a (9.12) szerint: $\overline{\alpha}_K = (R-1) - \alpha_K$

$X^{\overline{a_K, a_K}}$ – a (9.14) szerint képezhető.

(9.15) formula alapján felírható a következő alak is:

$$Y = F_1 + F_2 + \ldots F_{R-1} = \sum_{i=1}^{R-1} F_i \qquad (9.17)$$

ahol: $F_i = i \cdot \sum_{j=1}^{q_i} E_j^n, \quad i \in \{0, 1 \ldots (R-1)\}$

E_j^n – azokat a *„Min-term"*-eket jelöli, ahol $Y = i$

q_i – azoknak a „term"-eknek száma, ahol $Y = i$

A (9.17) alak $R = 2$ esetén a (9.15) alakba megy át.

9.10. példa: Írjuk fel a 9-24. ábra logikai táblázatával megadott $n = 1$ kétváltozós $R = 3$-értékű logikai függvényt, ill. komplemensét Σ és Π alakjaiban.

A többértékű elemeket rendszerező logikai táblázat felépítését az ábrán tanulmányozhatjuk. A sorok száma: $R^n = 3^2 = 9$. Az I. és II. oszlopban az X_1, X_2 független változók minden lehetséges értékvariációja szerepel. A III. oszlopban az Y-értékeknek a megfelelő sorba történő beírásával adjuk meg a konkrét függvényt. A IV. oszlop értékei már számítás eredményeképpen adódtak. Mivel a táblabeli értékek konstansok, így a komplemenset a (9.12) módon képezzük sorról sorra.

I.	II.	III.	IV.
X_1	X_2	Y	$\overline{Y}$
0	0	2	0
0	1	0	2
0	2	0	2
1	0	2	1
1	1	0	2
1	2	2	0
2	0	1	1
2	1	0	2
2	2	0	2

9-24. ábra *A 9.10. példa kiinduló igazságtáblázata*

Pl. $Y_x = 2$ esetén $\overline{Y}_x = (R-1) - Y_x = (3-1) - 2 = 0$.

– A *(9.15) szerinti kanonikus alakot felírva:*

$$Y = 2 \cdot X_1^{0,0} \cdot X_2^{0,0} + 0 \cdot X_1^{0,0} \cdot X_2^{1,1} + 0 \cdot X_1^{0,0} \cdot X_2^{2,2} +$$
$$+ 1 \cdot X_1^{1,1} \cdot X_2^{0,0} + 0 \cdot X_1^{1,1} \cdot X_2^{1,1} + 2 \cdot X_1^{1,1} \cdot X_2^{2,2} +$$
$$+ 1 \cdot X_1^{2,2} \cdot X_2^{0,0} + 0 \cdot X_1^{2,2} \cdot X_2^{1,1} + 0 \cdot X_1^{2,2} \cdot X_2^{2,2} =$$
$$= 2 \cdot X_1^{0,0} \cdot X_2^{0,0} + 1 \cdot X_1^{1,1} \cdot X_2^{0,0} + 2 \cdot X_1^{1,1} \cdot X_2^{2,2} + 1 \cdot X_1^{2,2} \cdot X_2^{0,0}$$

Mint látható, az α_K koefficienseket Y-nak a megfelelő sorhoz tartozó táblabeli értékei képezik.

– Innen átrendezéssel kapható a *(9.17) szerinti* alak:

$$Y = 1 \cdot \underbrace{(X_1^{1,1} \cdot X_2^{0,0} + X_1^{2,2} \cdot X_2^{0,0})}_{F_1} + \underbrace{2 \cdot (X_1^{0,0} \cdot X_2^{0,0} + X_1^{1,1} \cdot X_2^{2,2})}_{F_2} = F_1 + F_2$$

– A *komplemensek* Π alakban felírva:

$$\overline{Y} = (0 + X_1 + X_2)^{\overline{0,0}\ \overline{0,0}} \cdot (2 + X_1 + X_2)^{\overline{0,0}\ \overline{1,1}} \cdot (2 + X_1 + X_2)^{\overline{0,0}\ \overline{2,2}} \cdot$$
$$\cdot (1 + X_1 + X_2)^{\overline{1,1}\ \overline{0,0}} \cdot (2 + X_1 + X_2)^{\overline{1,1}\ \overline{1,1}} \cdot (0 + X_1 + X_2)^{\overline{1,1}\ \overline{2,2}} \cdot$$
$$\cdot (1 + X_1 + X_2)^{\overline{2,2}\ \overline{0,0}} \cdot (2 + X_1 + X_2)^{\overline{2,2}\ \overline{1,1}} \cdot (2 + X_1 + X_2)^{\overline{2,2}\ \overline{2,2}}$$

b) Részben meghatározott függvények

A többértékű függvényekkel kapcsolatban is adódhatnak olyan feladatok, amelyek részben határozott függvényekre vezetnek. Itt a h_i határozatlan értékekhez a $\{0, 1 \ldots (R-1)\}$ halmaz bármelyik értékét hozzárendelhetjük. A részben határozatlan függvények (9.15), ill. (9.16) típusú kanonikus alakjaiban a határozatlan eseteknél az α_K koefficiensek helyébe h_i kerül.

Ha egy $Y = F^n(X_1 \ldots X_n)$ R értékű függvény részben meghatározott függvény, akkor felírható a következőképpen, analógiában (9.17)-tel:

$$Y = \left(\sum_{i=1}^{R-1} F_1 \right) + F_h \qquad (9.18)$$

ahol: a zárójeles tag (9.17) szerint értelmezendő, továbbá:

$$F_h = \sum_{k=1}^{q_h} h_k \cdot E_K^n$$

E_K^n = azokat a „Min-term"-eket jelöli, ahol

$$Y = h_k \ (h_k \in \{0, 1 \ldots (R-1)\})$$

q_h = azoknak a term-eknek száma, ahol

$$Y = h_K.$$

c) Grafikus leírásmód, term-tábla

A többértékű függvények grafikus leírása analógiába állítható a kétértékű BOOLE-függvnyek *V–K* táblás leírásával. Így a kanonikus alak egy-egy termjének itt is a *term-tábla* egy-egy sejtje felel meg és a függvény annyi sejtet fed le, ahány nem nulla értékű, különböző term a kanonikus alakban szerepel.

Mivel a *term-tábla* sejtjeinek száma itt: R^n, ezért a grafikus leírás „R", ill. „n" növekedésével rohamosan terjengőssé válik.

X_1	X_2	Y
0	0	0
0	1	1
0	2	h_1
0	3	0
1	0	3
1	1	0
1	2	h_2
1	3	0
2	0	0
2	1	0
2	2	1
2	3	2
3	0	h_3
3	1	0
3	2	0
3	3	1

a)

X_2 \ X_1	0	1	2	3
0	0	3	0	h_3
1	1	0	0	0
2	h_1	h_2	1	0
3	0	0	2	1

b)

9-25. ábra *Logikai értéktáblázat határozatlan esetekkel*

9.11. példa: Adott a 9-25a. ábra logikai táblázatával az $Y = F^2(X_1, X_2)$, $R = 4$ értékű részben határozott függvény. Írjuk fel (9.15), (9.18) szerinti $\sum$ alakjait és ábrázoljuk *term-táblán* is.

– A *kanonikus* alak:

$$Y = 1 \cdot X_1^{0,0} \cdot X_2^{1,1} + h_1 \cdot X_2^{0,0} \cdot X_2^{2,2} + 3 \cdot X_1^{1,1} \cdot X_2^{0,0} + h_2 \cdot X_1^{1,1} \cdot X_2^{2,2} +$$
$$+ 1 \cdot X_1^{2,2} \cdot X_2^{2,2} + 2 \cdot X_1^{2,2} \cdot X_2^{3,3} + h_3 \cdot X_1^{3,3} \cdot X_2^{0,0} + 1 \cdot X_1^{3,3} \cdot X_2^{3,3}$$

– Az F_i *alak* (9.17) szerint:

$$Y = \underbrace{1 \cdot (X_1^{0,0} \cdot X_2^{1,1} + X_1^{2,2} \cdot X_2^{2,2} + X_1^{3,3} \cdot X_2^{3,3})}_{F_1} + \underbrace{2 \cdot X_1^{2,2} \cdot X_2^{3,3}}_{F_2} +$$
$$+ \underbrace{3 \cdot X_1^{1,1} \cdot X_2^{0,0}}_{F_3} + \underbrace{h_1 \cdot X_1^{0,0} \cdot X_2^{2,2} + h_2 \cdot X_1^{1,1} \cdot X_2^{2,2} + h_3 \cdot X_1^{3,3} \cdot X_2^{0,0}}_{F_h}$$

– A *term-táblát* a 9-25b. ábrán rajzoltuk fel.

9.4.2. Többértékű függvények minimalizálása

A többértékű logikai függvények minimalizálása a korábbi 2. fejezetben és a 7. fejezet 7.1. pontban bevezetett általános érvényű fogalmakon alapul. A 9-23. ábra operátorainak felhasználásával többértékű függvényeink és komplemenseik kétszintes minimál alakban adhatók meg. A többértékű logikák minimalizálásra formuláris, grafikus és táblázatos módszerek is kidolgozásra kerültek. A táblázatos módszerek közt szerepelnek a McCluskey eljárással rokon felépítésű eljárások is. A továbbiakban – helyszűke miatt – csak a formuláris és grafikus módszereket tárgyaljuk vázlatosan.

9.4.2.1. Formuláris minimalizálási módszerek

A formuláris minimalizálás a (9.11)-ben felsorolt tulajdonságoknak, a 9-23. ábra összefüggéseinek és a BOOLE-algebra itt is érvényes szabályainak felhasználásán alapul.

A minimalizálást célszerű a következő főbb lépésekben végezni:

I. lépés: A (9.17) vagy (9.18) szerinti alakok összetevőinek előállítása a célból, hogy „F_i"-ket egymás után minimalizálhassuk.

II. lépés: Először az $i = R–1$ esetekhez tartozó F_{R-1}-et egyszerűsítjük. Ha „F_h" is van, akkor az egyszerűsítésnél figyelembe vesszük a „h_k"-értékű don't care term-eket is, a h_1, h_2 … stb. értékeinek legcélszerűbb megválasztásával.

III. lépés: Ezután $i = R–2$ eset következik F_{R-2} egyszerűsítésével. Itt felhasználjuk a Min (A, B) operátor tulajdonságai alapján közvetlenül belátható alábbi tételt:

Ha ugyanazt a term-et egyszerre F_{R-l} és F_{R-l+1} összetevők is előállítják, akkor írható:

$$\text{Min}\,[F_{R-l}, F_{R-l+1}] = F_{R-l}, \tag{9.19}$$

melyből az következik, hogy F_{R-l}-nél az F_{R-l+1}-hez tartozó termeket is határozatlannak tekintjük az eredeti „h_k"-k mellett.

IV: lépés: Folytatjuk az $i = R–3, R–4$ … stb. esetek egyszerűsítését a *III. lépés* szabályai szerint egészen $i = 1$ esetig, F_1 minimalizálásának befejezéséig.

V. lépés: Minimálalak felírása.

9.12. példa: Írjuk fel a következő $R = 4$ értékű teljesen meghatározott $n = 2$ változós függvény minimál alakját:

$$Y = 2 \cdot X_1 \cdot X_2 + 2 \cdot X_1 \cdot X_2 + 2 \cdot X_1 \cdot X_2 + 2 \cdot X_1 \cdot X_2$$

Mint látható, a függvény nem kanonikus alakban van megadva. A kanonikus alak $R^n = 4^2 = 16$ összetevőt, a logikai táblázat ugyanannyi sort tartalmazna, ha felírnánk. Megvizsgálva a függvényt, megállapítható, hogy itt:

$$Y = F_2,$$

azaz a minimalizálás egyszerű algebrai egyszerűsítésre redukálódik, mivel $F_h = 0$ a teljes határozottság miatt. Ezek után a *disztributív* tulajdonság (9.11) és pl. a 9-23. ábra *a)* szabályának felhasználásával kihozható a minimálalak:

$$Y = 2 \cdot X_1 \cdot (X_2 + X_2) + 2 \cdot X_1 \cdot (X_2 + X_2) =$$
$$= 2 \cdot X_1 \cdot \overline{X_2} + 2 \cdot X_1 \cdot \overline{X_2} =$$
$$= 2 \cdot \overline{X_2}(X_1 + X_1) =$$
$$= 2 \cdot \overline{X_2} \cdot \overline{X_1}$$

X_1	X_2	Y
0	0	h_1
0	1	1
0	2	2
1	0	0
1	1	2
1	2	h_1
2	0	1
2	1	1
2	2	2

9-26. ábra *Logikai táblázat a 9.13. példához*

9.13. példa: Adva a 9-26. ábra logikai táblázatával egy $R = 3$ állapotú, $n = 2$-változós, részben határozott többértékű függvény.

A minimalizált alakot az előzőleg vázolt algoritmus segítségével állíthatjuk elő:

I. lépés: (9.18) Σ alak összetevőinek előállítása:

$$Y = F_1 + F_2 + F_h$$

Az $Y = 1$-hez tartozó sorokból felírható:

$$F_1 = 1 \cdot (X_1 \cdot X_2 + X_1 \cdot X_2 + X_1 \cdot X_2)$$

Az $Y = 2$ sorokból:

$$F_2 = 2 \cdot (X_1 \cdot X_2 + X_1 \cdot X_2 + X_1 \cdot X_2)$$

végül az $Y = h_i$ sorokból:

$$F_h = h_1 \overset{0,0}{X_1} \cdot \overset{0,0}{X_2} + h_2 \overset{1,1}{X_1} \cdot \overset{2,2}{X_2}$$

II. lépés: Ezután írhatjuk az $i = R - 1 = 3 - 1 = 2$ esetre:

$$F_2 + F_h = \overset{2,2}{X_2}(2 \cdot \overset{0,0}{X_1} + 2 \cdot \overset{2,2}{X_1} + h_2 \overset{1,1}{X_1}) +$$
$$+ \overset{1,1}{X_1}(2 \cdot \overset{1,1}{X_2} + h_2 \overset{2,2}{X_2}) + h_1 \cdot \overset{0,0}{X_1} \cdot \overset{0,0}{X_2}$$

felvéve: $h_2 = 2$ és $h_1 = 0$ értékeket kapjuk:

$$F_2 + F_h = 2 \cdot \overset{2,2}{X_2}(\overset{0,0}{X_1} + \overset{1,1}{X_1} + \overset{2,2}{X_1}) + 2 \cdot \overset{1,1}{X_1}(\overset{1,1}{X_2} + \overset{2,2}{X_2}) =$$
$$= 2 \cdot \overset{2,2}{X_2} \cdot \overset{0,2}{X_1} + 2 \cdot \overset{1,1}{X_1} \cdot \overset{1,2}{X_2}$$

mivel: $\overset{0,2}{X_1} = \overset{0,R-1}{X_1} = \mathrm{R} - 1 = 3 - 1 = 2$ (b. szabály)

$$F_2 + F_h = 2(\overset{2,2}{X_2} \cdot 2 + \overset{1,1}{X_1} \cdot \overset{1,2}{X_2}) = 2(\overset{2,2}{X_2} + \overset{1,1}{X_1} \cdot \overset{1,2}{X_2})$$

III. lépés: A következő lépés az $i = R - 2 = 3 - 2 = 1$ eset, ahol az összes 2-es tényezőt h-nak tekintjük a *-gal jelzett $F_2 + F_h$ alakban, majd írjuk:

$$F_1 + (F_2 + F_h)_{2=h} = 1(\overset{0,0}{X_1} \cdot \overset{1,1}{X_2} + \overset{2,2}{X_1} \cdot \overset{0,0}{X_2} + \overset{2,2}{X_1} \cdot \overset{1,1}{X_2}) +$$
$$+ \overset{2,2}{X_2}(h \cdot \overset{0,0}{X_1} + h \cdot \overset{2,2}{X_1} + h_2 \cdot \overset{1,1}{X_1}) +$$
$$+ \overset{1,1}{X_1}(h \cdot \overset{1,1}{X_2} + h_2 \cdot \overset{2,2}{X_2}) + h_1 \cdot \overset{0,0}{X_1} \cdot \overset{0,0}{X_2} =$$
$$= \overset{0,0}{X_1}(1 \cdot \overset{1,1}{X_2} + h \cdot \overset{2,2}{X_2} + h_1 \cdot \overset{0,0}{X_2}) + \overset{2,2}{X_1}(1 \cdot \overset{0,0}{X_2} + 1 \cdot \overset{2,2}{X_2} + h \cdot \overset{1,1}{X_2}) +$$
$$+ \overset{1,1}{X_1} \cdot (h_2 \cdot \overset{2,2}{X_2} + h \cdot \overset{1,1}{X_2} + h_2 \cdot \overset{2,2}{X_2})$$

$h = h_1 = 1$-et választva az első két tagban, $h = h_2 = 0$-át a harmadikban:

$$F_1 + (F_2 + F_h)_{2=h} = 1 \cdot \overset{0,0}{X_1} \cdot \overset{0,2}{X_2} + 1 \cdot \overset{2,2}{X_1} \cdot \overset{0,2}{X_2} =$$

$$= 1 \cdot \overset{0,2}{X_2} (\overset{0,0}{X_1} + \overset{2,2}{X_1}) = 1 \cdot \overset{0,2}{X_2} \cdot \overset{1,1}{\overline{1}} = 1 \cdot \overset{1,1}{\overline{X_1}}$$

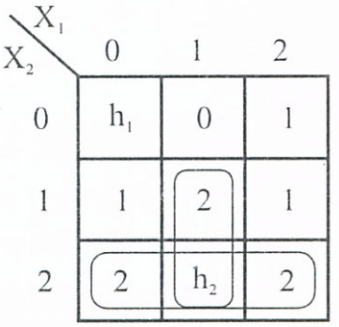
IV. lépés-re nincs szükség, miután már a *III. lépés*ben elértük F_1-et.

V. lépés: A minimálalak:

$$Y = 2 \cdot (\overset{2,2}{X_2} + \overset{1,1}{X_1} \cdot \overset{1,2}{X_2}) + 1 \cdot \overset{1,1}{\overline{X_1}}$$

9.4.2.2. Grafikus minimalizálási módszerek

a) A term-tábla sejtjeiből itt is alkothatók *tömbök,* melyek egy lefedő implikánsnak felelnek meg. A törekvés úgyszintén a *„legnagyobb tömbök"* előállítására irányul, miután a lefedő implikánsok így egyszerűbbek lesznek. (9.17), (9.18) és (9.19) értelmében – hasonlóan a formuláris módszerekhez – itt is az F_i, $i = R–1$ esettel kezdjük a tömbösítést, majd az itteni sejteket „h"-nak tekintve, folytatjuk az $i = R–2$ esettel.



a) $i = R - 1 = 2$

$$F_2 + F_h = 2(\overset{2.2}{X_2} + \overset{1.1}{X_1} \cdot \overset{1.2}{X_2})$$

b) $i = R - 1 = 2$

$$F_1 + (F_2 + F_h)_{2=h} = 1 \cdot \overset{1.1}{\overline{X_1}}$$

$$Y = 2(\overset{2.2}{X_2} + \overset{1.1}{X_1} \cdot \overset{1.2}{X_2}) + 1 \cdot \overset{1.1}{\overline{X_1}}$$

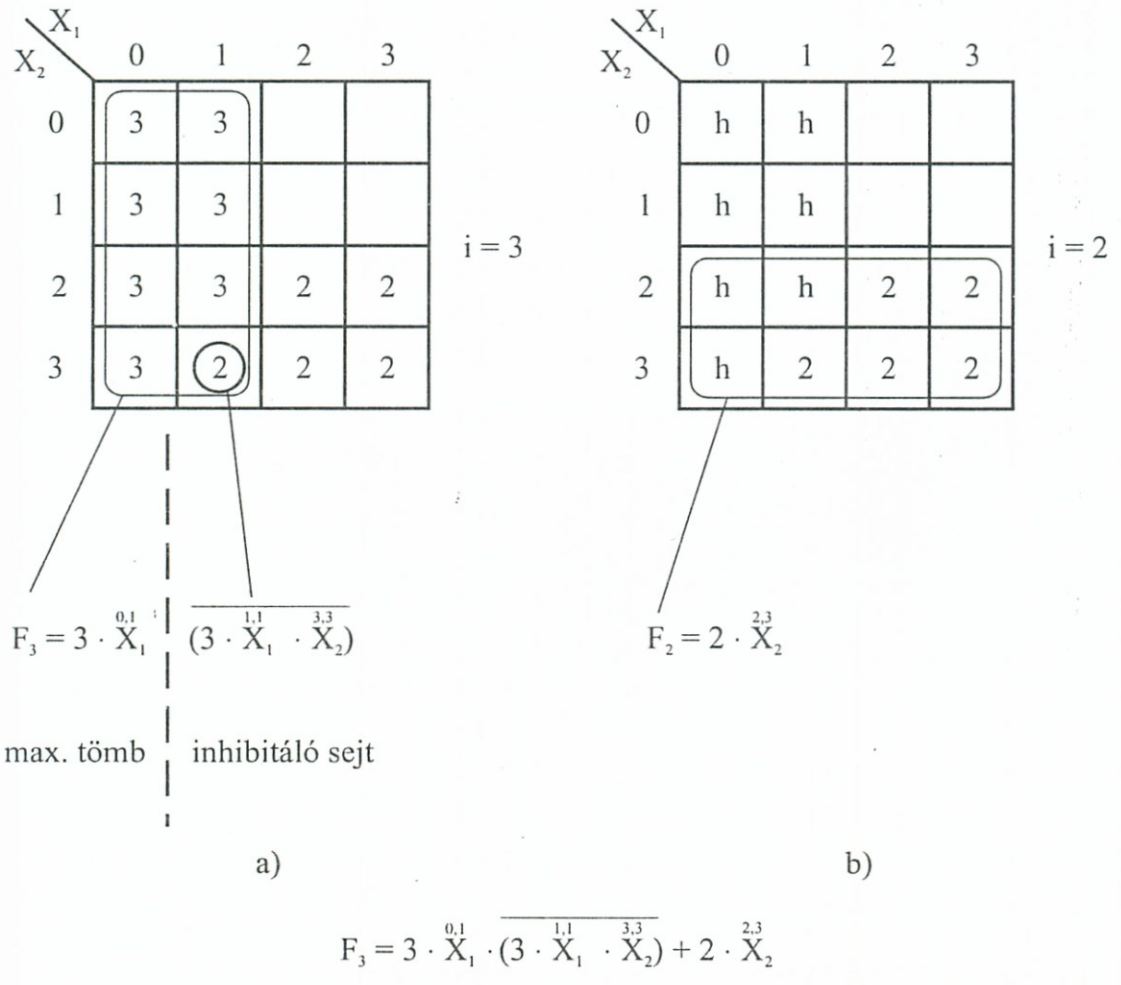
9-27. ábra *A 9.13. példa grafikus megoldása*

b) A gátlási elv alapgondolata hasonló a BOOLE-függvényeknél gyakran alkalmazott módszerrel. Eszerint úgy képzünk maximális tömböket, hogy – ha ez előnyös – olyan sejteket is befoglalunk a tömbökbe, melyek nem tartoznak F_i-hez. Ezután maximális tömbjeinket az F_i-hez nem tartozó sejtek term-jeivel *inhibitáljuk,* oly módon, hogy ha $h_K \cdot E_K$ az inhibitáló term alakja, akkor $h_K = R-1$ értéket adunk az eredeti érték helyett. Mint látható, itt is minden kimeneti értékhez külön minimalizáljuk, mint a formuláris módszereknél is tettük.

9.14. példa: Minimalizáljuk grafikusan a *9.13. példá*ban tárgyalt függvényt:

A 9-27a. ábrán látható term-tábla tömbösítése megfelel a *9.13. példa* $i = 2$ lépésének, a 9-27b. ábra pedig az $i = 1$-nek. A kapott eredmények összehasonlíthatók.

9.15. példa: Minimalizáljuk gátlási elv felhasználásával a 9-28a. ábrán term-táblájával megadott függvényt.



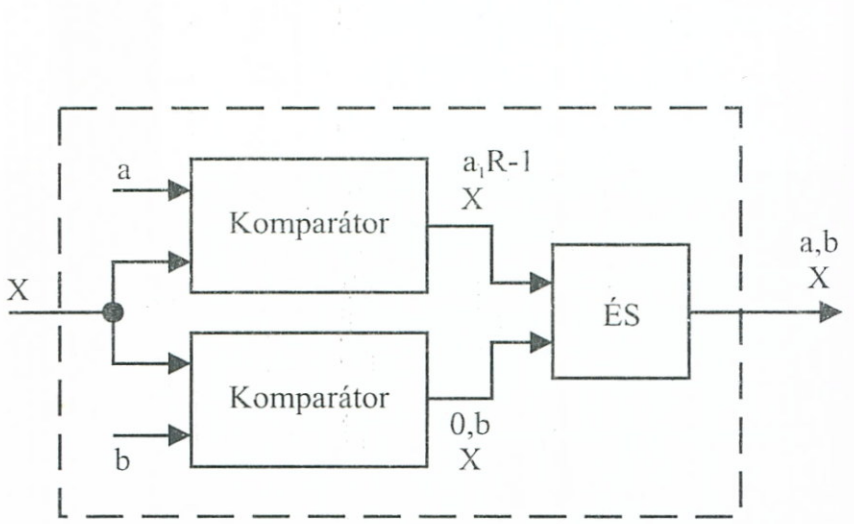
$$F_3 = 3 \cdot \overset{0,1}{X_1} \cdot \overline{(3 \cdot \overset{1,1}{X_1} \cdot \overset{3,3}{X_2})} + 2 \cdot \overset{2,3}{X_2}$$

9-28. ábra *Gátlási elv grafikus alkalmazása*

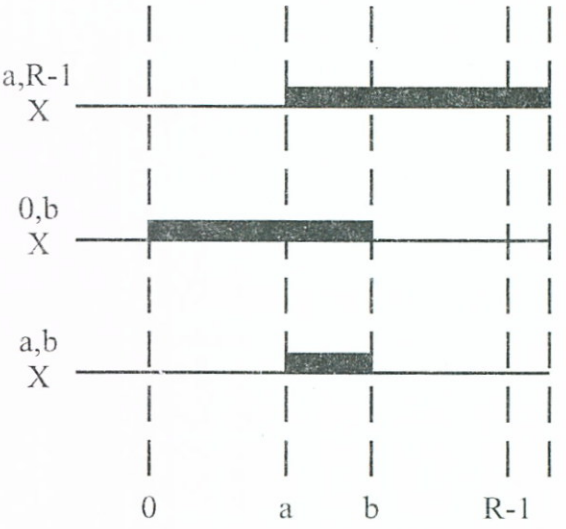
A 9-28a. ábrán $i = 3$ esetre látható a gátlási elv alkalmazása F_3-nál. A 9-28b. ábrán $i = 2$-re F_2-höz egy nagy tömb képezhető minden 3-as sejt „h"-nak vételével. Az $i = 1$-es esetre nem kerül sor, mivel „1"-es sejt a függvényben nincs.

9.4.3. Megjegyzések, realizációs kérdések

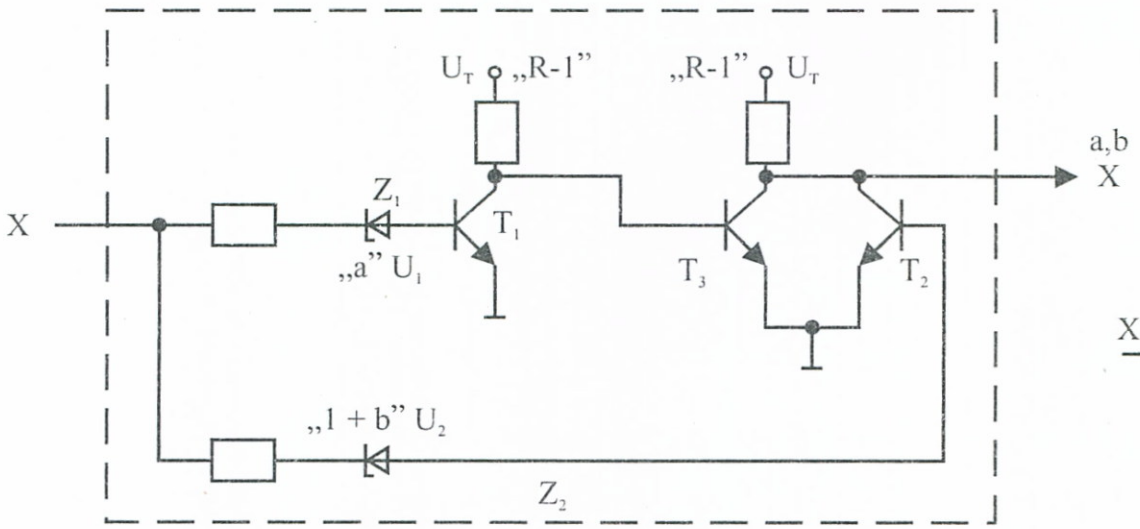
A többértékű logikák a jelenségek tömörebb leírását teszik lehetővé. (Például négy kétállapotú elemet helyettesíthet két négyállapotú stb.) Elterjedésüknek jelenleg – hasonlóan a küszöblogikákkal kapcsolatban a 9.2.6. pontban elmondottakhoz – elsősorban realizációs problémák (pl. áramköri építőelemekkel szemben támasztott, fokozott pon-



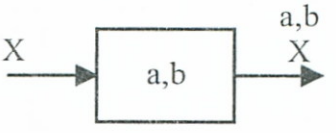
a)



b)



c)



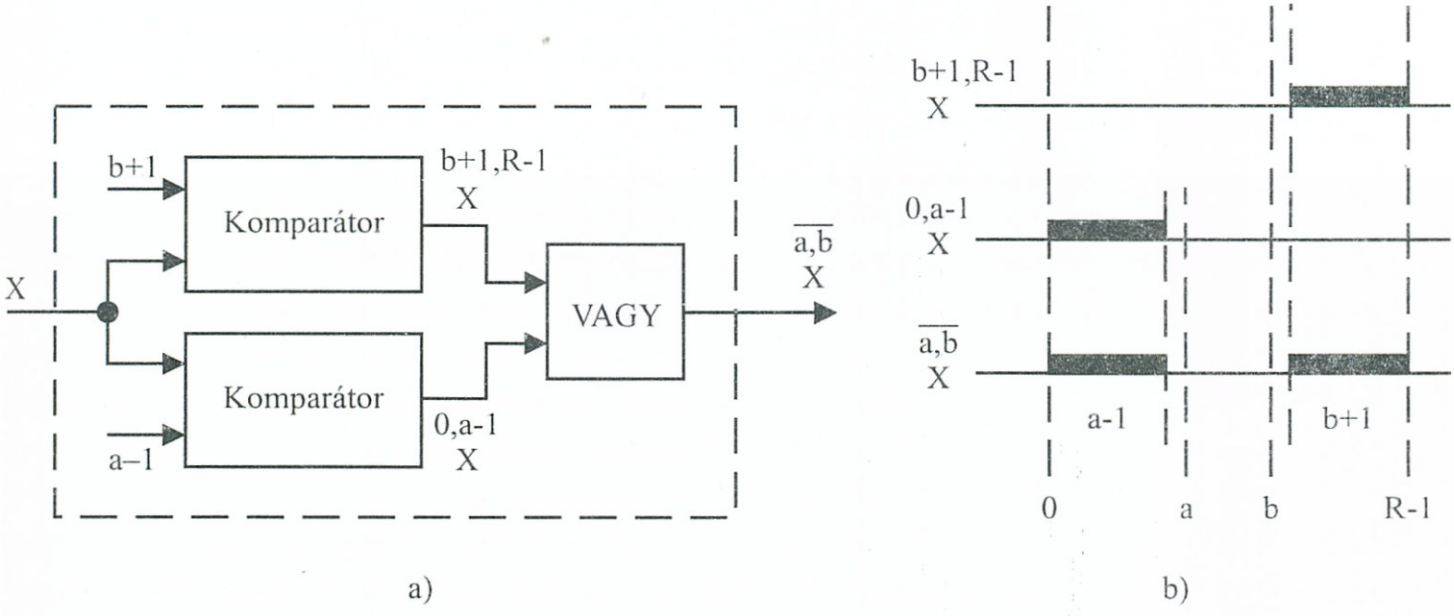
d)

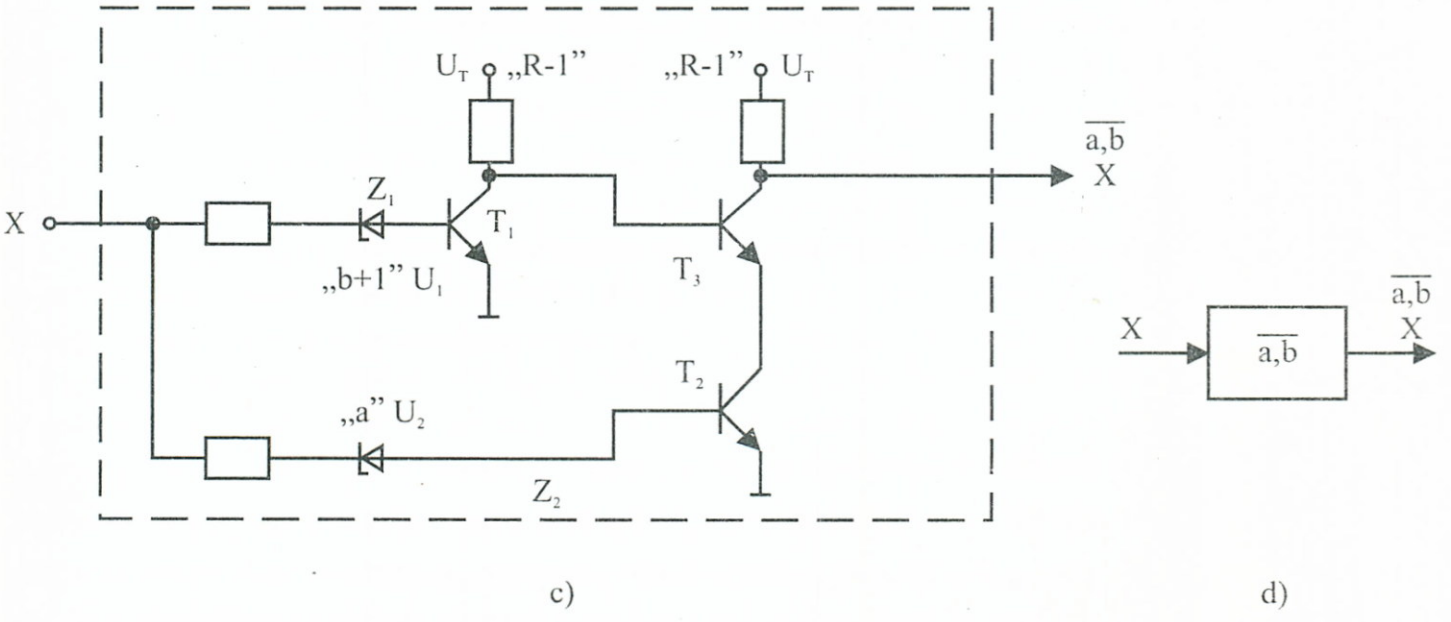
9-29. ábra *Az $\overset{a,b}{X}$ operátor egy lehetséges realizációja*

tossági igény, költségesebb elemek stb.), másrészt a szisztematikus matematikai módszerekkel kapcsolatosan még fennálló problémák szabnak határt. Az utóbbi években, a technikai lehetőségek fejlődésével párhuzamosan a témakör irodalma igen kiterjedtté vált.

A most tárgyalt többértékű logikai rendszer, mint láttuk, olyan felépítésű, hogy a *klasszikus kétértékű* ÉS, VAGY alapáramkörök benne alkalmazásra kerülhetnek, a Min, ill. Max műveleteknek megfelelően.

Az $\overset{a,b}{X}$ unáris operátor realizálásának egy elvi vázlatát a 9-29a. ábrán láthatjuk. A 9-29b. ábrából leolvasható, hogy az „a, (R–1)" és „0, b" intervallum-határok beállítását komparátorok végzik. A komparálási műveletek eredményeiből egy ÉS művelettel „vágjuk ki" a ki-





9-30. ábra *Az* $\overset{\overline{a,b}}{X}$ *komplemens operátor egy lehetséges realizációja*

ván „*a, b*" intervallumot. Egy konkrét áramköri realizáció látható a 9-29c., szimbolikus jelölése a 9-29d. ábrán. Itt U_1, U_2 a T_1, T_2 tranzisztorok nyitó feszültségei, a határértékek beállítását a választott Zener-feszültségű Zener-diódák végzik.

Hasonló elven értelmezhető az $\overline{X}_{a,b}$ komplemens realizálásának 9-30a., b, c, d. ábrákon bemutatott példája is.

9.5. Fuzzy-logikák

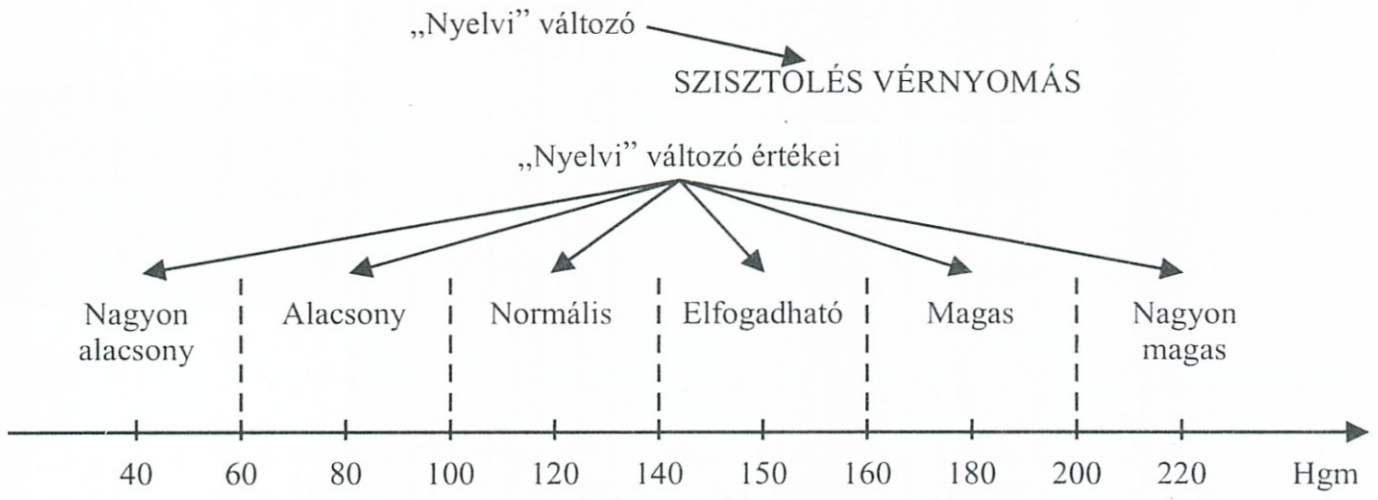
Az elektronikus rendszerek fejlődése során mind bonyolultabb rendszerek alakulnak ki, melyek egyre több összetevőt tartalmaznak és ezekkel – a lehető legrövidebb idő alatt – nagyon összetett feladatokat kell megoldani. Az ilyen, nagyon összetett rendszerek működésének leírása a hagyományos, minden részletet figyelembevevő, „precíz" matematikai eszközökkel egyre nehézkesebbé válik, és ez kihat a „jó" működés feltételeire is.

A 60-as évek közepén L. A. Zadeh által bevezetett, ún. Fuzzy-halmazok a műszaki feladatoknak egy új koncepció szerinti megfogalmazását és tárgyalását tették lehetővé. Az angol „fuzzy" szó számos jelentése közül itt az „elmosódott körvonalú", „életlen", „homályos" fordítások kaptak szerepet, amelyek érzékeltetik a dolgok „nem részletekbe menő" jellemzését, melyekre Zadeh matematikai hátteret keresett. Zadeh, abból a megállapításból indult ki, hogy az emberi nyelv szavaival és mondataival történő jellemzés egyszerűbb, tömörebb és gyorsabban kiértékelhető lehet, mint a számokon alapuló, részletekbe kényszerített, hagyományos matematikai jellemzés. Fentiek érdekében – többek között – célszerűnek látta bevezetni az emberi kifejezésmódhoz jobban illeszkedő, ún. *nyelvi változó* fogalmát.

Vizsgáljuk meg az ezzel kapcsolatos problémákat egy példán keresztül. A 9–31. ábrán az emberi szisztolés vérnyomás gyógyítás szempontjából érdekes értékeit egy 40-től 220 Hgmm-es skálán ábrázoltuk. Egy konkrét mérés esetén többféle módon járhatunk el:

a) Követhetjük a klasszikus „precíz" módszert, ahol az ábrázolt 180 szám közül feljegyezzük az aktuális mért értéket (pl. 147 Hgmm), és ezt az adatot használjuk fel a további gyógyítási folyamatban.

b) Megoldhatjuk a problémát úgy is, hogy (korábbi tapasztalatokat is figyelembevéve) a 180 elemből álló halmaz elemeit a 9–31. ábra szerinti módon (Nagyon alacsony, ..., Normális, ..., Nagyon magas)



9-31. ábra *Nyelvi változó értékeinek értelmezése*

szubhalmazokba csoportosítjuk a gyorsabb kiértékelhetőség érdekében, és a továbbiakban a mérések során csak azt figyeljük, hogy az aktuális adat melyik alhalmazba került és ennek alapján *minősítjük* a beteg állapotát.

Összehasonlítva a két változatot, megállapíthatjuk, hogy az a) eset klasszikusan precíz vérnyomás-eredményt szolgáltat 1 Hgmm-es pontossággal, viszont itt 180-féle számértékkel kell foglalkoznunk és ez a pontosság a legtöbb esetben nem is szükséges. A b) változatnál a vérnyomást „nyelvi változó" értékekkel jellemezzük, melyekre a példában itt 6-féle értéket vezettünk be, így a kiértékeléskor csak ezekkel kell foglalkoznunk. Igaz, hogy az a) változathoz viszonyítottan ez utóbbi jellemzés kevésbé részletezett, de az orvosi gyakorlat számára – adott esetben – kielégítő információt szolgáltathat.

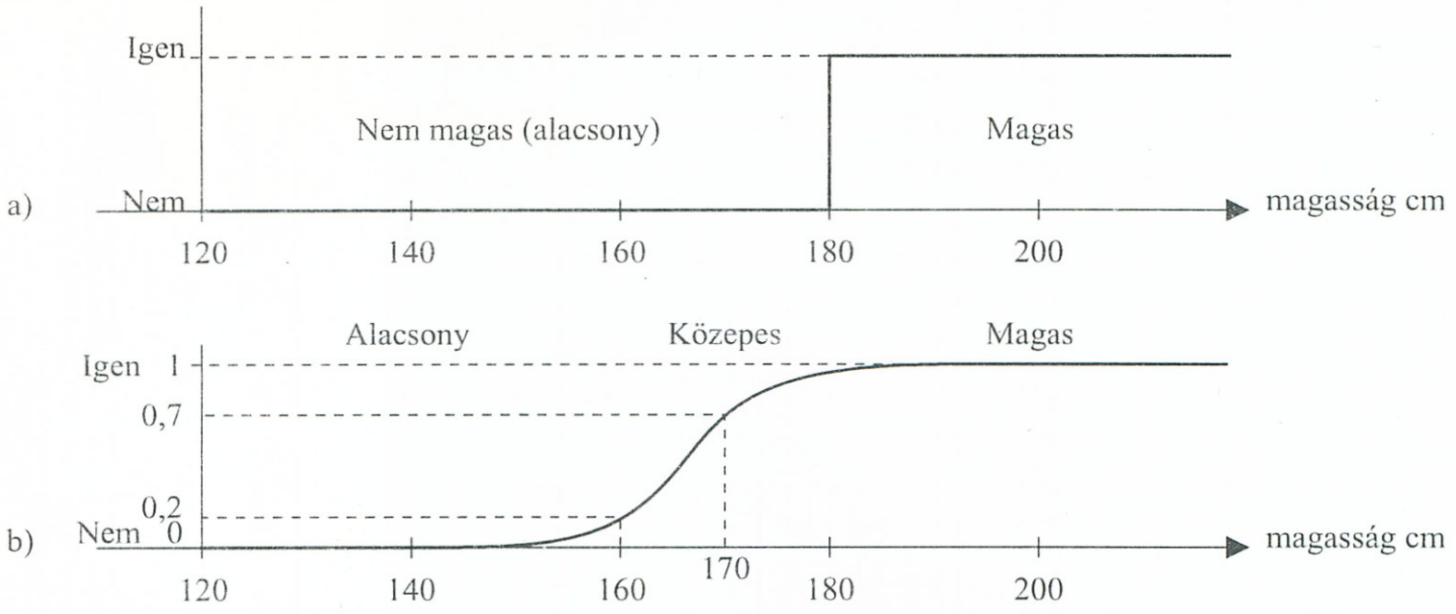
A példával kapcsolatosan még további két, szélső esetet is megfogalmazhatunk:

c) Szélső esetnek tekinthető az a helyzet, midőn a vérnyomást például egy analóg mutatós mérőeszközzel mérjük, azaz a mért értékek skálája a 40–220-as intervallumban *folytonos* lesz, és bármely valós értéket felvehet.

d) Ugyancsak szélső esetnek tekinthető, ha a 40–220-as halmazt csupán két szubhalmazra osztjuk fel (pl. 160-nál elválasztva) „nem magas" és „magas" nyelvi változó értékek bevezetésével, amely például a magas vérnyomásúak kiszűrésére használható.

Egy másik, ugyancsak Zadeh által tárgyalt problémát, a minősítő osztályok és átmenetek kérdéskörét egy újabb példa keretében célszerű megvizsgálnunk.

A 9–32a. ábrán az embereket magasságuk alapján két:



9-32. ábra *Testmagasság kategóriák*

– magas,
– nem magas (alacsony)

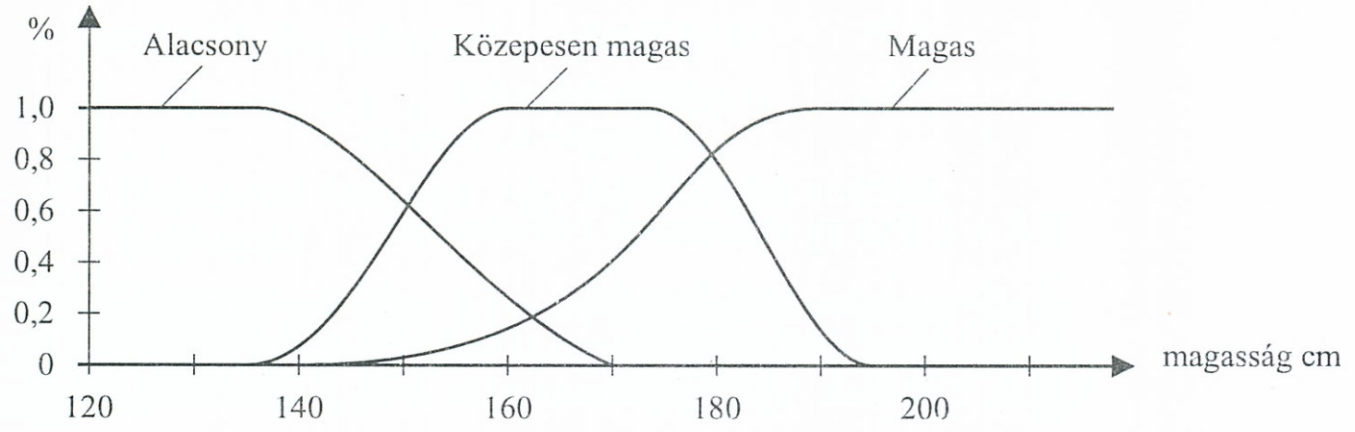
csoportba soroltuk. A magasság mértékét kifejező ugrásfüggvény a két csoportot egyértelműen elkülöníti olymértékben, hogy közös elemük nincsen. Ez a leírásmód már ismerős a bináris logikai halmazok témaköréből és az ott tárgyaltak alapján az előbbieket úgy is megfogalmazhatjuk, hogy az emberek egy adott vizsgálati csoportját (egy ún. X „univerzumot") két elem (0 – nem magas, 1 – magas) készletére képeztük le, azaz:

$$X \rightarrow \{0, 1\}$$

A 9–32b. ábra egy más csoportosítási elvet szemléltet. A magasság mértékét kifejező görbe nem jelöl ki egyértelmű elkülönítést. Itt az X univerzumot nem kizárólag két elem készletére képezzük le, hanem bármely (diszkrét vagy folytonos skála szerinti) értékre a 0 – nem magas, 1 – magas értékek között. Ily módon például a 170 cm „magas" emberekre, azt is mondhatjuk, hogy kb. 70%-osan magasak, a 160 cm „magas"-akra pedig, hogy kb. 20%-osan magasak – a görbéről leolvashatóan, azaz a magasságok %-os mértékét egy szorzófaktorral jellemezve is megadhatjuk.

Ez a jellemzésmód nem jelöl ki szigorúan elkülönített osztályokat. A „magasság"-ot például tovább tagolhatjuk:

nem is olyan magas (közepes),
egyáltalán nem magas (alacsony)



9-33. ábra *Testmagasság és fuzzy halmazok*

osztályokra, melyeknél (miután határaik meglehetősen szubjektívek is lehetnek a megfogalmazások alapján) az is előfordulhat, hogy két osztály közös elemeket is tartalmaz. Ily módon például a 160 cm magasságú embert úgy is jellemezhetjük, hogy „magas is, meg alacsony is", de azt is mondhatjuk, hogy „közepesen magas".

A fentiekben elmondottakra valóban jellemzőek az „életlen", „homályos", „elmosódott körvonalú" meghatározások, melyeket a bevezetőben a Fuzzy szó kapcsán már említettünk. A testmagasságokra vonatkozó példánkkal kapcsolatos, ilyen fuzzy-halmazokat szemléltet a 9–33. ábra.

9.5.1. Fuzzy-halmazok és műveletek

9.5.1.1. A fontosabb jelölések összefoglalása

Miután a fuzzy-halmazok tanulmányozásánál számos halmazelméleti jelölést kell használnunk, ezért ezek közül a leggyakoribbakat előzetesen összefoglaljuk:

$\alpha \in \{0, 1\}$	α értéke: 0, ill. 1 lehet
$\alpha \in [0, 1]$	α értéke: lehet 0, ill. 1, ill. 0,1 között bármi diszkrét vagy folytonos skálaérték
$a \in A$	a eleme A-nak
$A \subset B$	A benne foglaltatik B-ben
$A \subseteq B$	A benne foglaltatik B-ben, vagy vele egyenlő
$\forall\, a$	valamennyi a
$\exists\, a$	létezik olyan a
$A \Rightarrow B$	ha A, akkor B (implikáció)
$A \Leftrightarrow B$	A ekvivalens B-vel (ekvivalencia)
$f\colon A \rightarrow \mid B$	A leképezése B-re f alapján
$\{a \mid \ldots\}$	azon a elemek halmaza, melyekre … teljesül

$\neg$	komplemens műveleti jel
$\cup$	unió, egyesítés
$\cap$	interszekció, metszet

9.5.1.2. Fuzzy-halmazok definíciója

*Fuzzy-halmaz*nak nevezzük azt a halmazt, melynek minden X „univerzum"-beli eleméhez egy 0 és 1 közé eső valós számot rendelünk. Egy adott A fuzzy-halmaz esetén, mely X-nek alhalmaza írható:

$$\mu_A : X \rightarrow [0, 1]$$

itt μ_A az A fuzzy-halmaz ún. *tagsági* függvénye.

Amennyiben a fenti [0, 1] helyett {0, 1} szerepel, azaz a halmaz-elemekhez 0 vagy 1-et rendelünk, akkor a fuzzy-halmaz egy határozott halmazba megy át.

Fuzzy-halmazokra felírhatók az alábbiak, ha x_i egy halmazelem és $\mu_i(x_i)$ a vele kapcsolatos tagsági függvényérték:

Diszkrét elemeknél

$$A = \sum_{i=1}^{h} \mu_i \,/\, x_i \qquad (9.20a)$$

Folytonos elemeknél

$$A = \int \mu(x)/x \qquad (9.20b)$$

A formulákban a μ/x az adott x és a μ tagsági függvény összetartozását érzékelteti.

9.16. példa: Tételezzük fel, hogy a 9–33. ábra alapján a 9–34. ábra táblázatában szereplő diszkrét fuzzy-halmaz értékek állnak rendelkezésünkre. A táblázatból (9.20) alapján a KÖZEPESEN MAGAS (KM) fuzzy-halmaz a következőképpen írható le:

testmagasság cm		120	130	140	150	160	170	180	190	200	210
ALACSONY (A)	μ_A	1	1	0,9	0,5	0,2	0	0	0	0	0
KÖZEPESEN MAGAS (KM)	μ_{KM}	0	0	0,1	0,5	1	1	0,6	0,1	0	0
MAGAS (M)	μ_M	0	0	0	0,1	0,3	0,7	0,9	1	1	1

9–34. *Fuzzy-halmazok diszkrét elemekkel*

$$KM = 0/120 + 0/130 + 0{,}1/140 + 0{,}5/150 + 1/160 +$$
$$+ 1/170 + 0{,}6/180 + 0{,}1/190 + 0/200 + 0/210$$

9.5.1.3. Fuzzy-halmazokkal kapcsolatos fontosabb definíciók

a) *Mag*-nak (core) nevezzük az X univerzumon értelmezett fuzzy-halmazt, mely az A halmaz valamennyi $\mu_A = 1$ értékű elemét tartalmazza:

$$\text{core}\,(A) = \{x \in X \mid \mu_A(x) = 1\} \tag{9.21}$$

b) *Hordozó*nak (support) nevezzük azt a fuzzy-halmazt, mely az A-halmaz valamennyi $\mu_A \geq 0$ értékű elemét tartalmazza:

$$\sup(A) = \{x \in X \mid \mu_A(x) > 0\} \tag{9.22}$$

c) *Magasság*nak (height) nevezzük az A halmazban lévő legnagyobb tagsági függvényértéket:

$$\text{height}\,(A) = \max_x(\mu_A(x)) \tag{9.23}$$

d) *Normalizált*nak nevezzük az A halmazt, ha magassága:

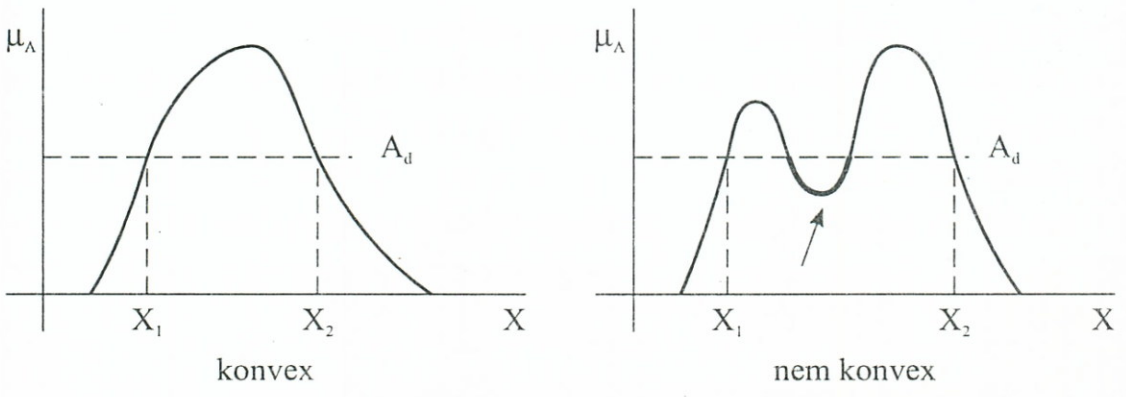
$$\text{height}\,(A) = 1$$

e) *α-vágat*nak nevezzük az A halmaz hordozójának (support) azt a részhalmazát, melynek elemeihez rendelt $\mu_A(x)$ tagsági függvényérték nem kisebb az α valós számnál

$$A_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\} \tag{9.24}$$

f) *Konvex*nek nevezzük az A halmazt, ha valamennyi α-vágata konvex. A konvexitást grafikusan a 9–35. ábrán szemléltettük.

$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \geq \min[\mu_A(x_1), \mu_A(x_2)], \forall x_1 x_2 \in X, \lambda \in [0,1]$$



9-35. ábra *Konvex és nem-konvex fuzzy halmazok*

g) *Számosság*nak (kardinalitás) nevezzük az A halmazt alkotó elemek μ_A értékeinek összegét.

$$|A| = \sum_{x \in X} \mu_A(x) \tag{9.25}$$

h) *Részhalmaz*nak tekintjük az A halmazt B halmazra vonatkozóan, ha valamennyi univerzum-beli elemére fennáll, hogy az A halmaz tagsági függvényértékei nem nagyobbak a megfelelő B halmazbeli tagsági függvényértékeknél.

$$A \subseteq B : \mu_A(x) \leq \mu_B(x) \quad \forall x \in X \tag{9.26}$$

9.17. példa: Vizsgáljuk meg az előzőkben bevezetett fogalmakat a 9–34. ábra KÖZEPESEN MAGAS fuzzy-halmazának esetében:

– KM = 0/120 + 0/130 + 0,1/140 + 0,5/150 + 1/160 + 1/170 + 0,6/180 + 0,1/190 + 0/200 + 0/201
– mag : core (KM) = {160, 170}
– hordozó : sup (KM) = {140, 150, 160, 170, 180, 190}
– magasság : height (KM) = 1
– a = 0,5 vágat : $(KM)_{\alpha = 0,5}$ = 150, 160, 170, 180
– konvexitás : KM → konvex (lásd: 9–33. ábra)
– számosság : |KM| = 0,1 + 0,5 + 1 + 1 + 0,6 + 0,1 = 3,3

9.5.1.4. Műveletek fuzzy halmazokon

A fuzzy halmazokkal kapcsolatosan eddig elmondottak alapján felmerülhet a gondolat, hogy a fuzzy problémakör rokonságba hozható a 9.4. pontban bemutatott többértékű logikák problémakörével. Ez a rokonság különösen a diszkrét elem-skálájú fuzzy halmazoknál szembetűnő, és azt sugallja, hogy az ott alkalmazott műveleti, számítási elvek a fuzzy halmazoknál is felhasználhatók. Zadeh is innen indult ki, és az általa bevezetett műveleteket a POST–LUKASIEWITZ többértékű műveletrendszer – bizonyos értelemben vett – általánosításának tekintette. Később a Zadeh által bevezetett műveleteket több szerző (többféle szempontot előtérbe helyezve) többféle irányban általánosította és a jelenlegi változatok „kínálatából", azt a változatot célszerű alkalmazni, amelyik az adott feladat leírásához a legjobban illeszkedik.

α) A Zadeh-féle standard műveletek:

a) Unió (max-művelet), halmazok egyesítése

Az A és B fuzzy halmazok egyesített halmazának nevezzük a következő halmazt:

$$A \cup B = \{(x, \mu_{A\cup B}(x)) \mid x \in X, \mu_{A\cup B} = \max [\mu_A(x), \mu_B(x)]\} \quad (9.27)$$

azaz az egyesítési tagsági függvény a komponens tagsági függvények *max* kapcsolatából származik.

b) Metszet (min-művelet), halmazok metszete

Az A és B fuzzy halmazok metszet halmazának nevezzük a következő halmazt:

$$A \cap B = \{(x, \mu_{A\cap B}(x) \mid x \in X, \mu_{A\cap B} = \min [\mu_A(x), \mu_B(x)]\} \quad (9.28)$$

azaz a komponens tagsági függvények *min* kapcsolatát kell képezni.

c) Komplemens

Az A fuzzy halmaz komplemens halmaza:

$$A^c = \{(x, \mu_{A^c}(x)) \mid x \in X, \mu_{A^c}(x) = 1 - \mu_A(x)\} \quad (9.29)$$

azaz a komplemens tagsági értékek előállításakor a megfelelő eredeti értékeket 1-ből ki kell vonni.

9.18. példa: Végezzük el a Zadeh-féle standard műveleteket a 9–34. ábra KÖZEPES (KM) és MAGAS (M) fuzzy halmazain.

A *max* műveletnél a μ_{KM}, μ_M párok közül mindig a nagyobbat kell választani az eredményhez:

$$KM \cup M = 0/120 + 0/130 + 0{,}1/140 + 0{,}5/150 + 1/160 + \\ + 1/170 + 0{,}9/180 + 1/190 + 1/200 + 1/210$$

A *min* műveletnél a μ_{KM}, μ_M párok közül mindig a kisebbet kell választani az eredményhez:

$$KM \cap M = 0/120 + 0/130 + 0/140 + 0{,}1/150 + 0{,}3/160 + \\ + 0{,}7/170 + 0{,}6/180 + 0{,}1/190 + 0/200 + 0/210$$

A komplementálásnál az 1-ből kivont μ értéket kell beírni az eredménybe:

$$KM^c = 1/120 + 1/130 + 0{,}9/140 + 0{,}5/150 + 0/160 + \\ + 0/170 + 0{,}4/180 + 0{,}9/190 + 1/200 + 1/210$$

A példa eredményeiből is látható, hogy itt a formális logika szabályai már nem érvényesek, azaz:

$$A \cup A^c \neq X$$

$$A \cap A^c \neq 0$$

β) További műveletek

A Zadeh-féle három alapműveleten túlmenően a következő további műveleteket definiálhatjuk:

d) Algebrai összeg

$$A + B = \{(x, \mu_{A+B}(x) \mid x \in X, \mu_{A+B}(x) = \\ = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)\} \tag{9.30}$$

e) Algebrai szorzat

$$A \cdot B = \{(x, \mu_{A \cdot B}(x) \mid x \in X, \mu_{A \cdot B}(x) = \mu_A(x) \cdot \mu_B(x)\} \tag{9.31}$$

f) Korlátozott összeg

$$A \oplus B = \{(x, \mu_{A \oplus B}(x) \mid x \in X, \mu_{A \oplus B}(x) = \\ = \min[1, \mu_A(x) + \mu_B(x)]\} \tag{9.32}$$

g) Korlátozott szorzat

$$A \odot B = \{(x, \mu_{A \odot B}(x) \mid x \in X, \mu_{A \odot B}(x) = \\ = \max[0, \mu_A(x) - \mu_B(x)]\} \tag{9.33}$$

γ) A három alapművelet általánosítása

Az alapművelet általánosításának több elvi változata is elképzelhető különféle speciális függvények és paraméterek bevezetésével, így az eltelt néhány évtized folyamán számos műveletcsalád alakult ki. A családok száma napjainkban sem tekinthető lezártnak. Könyvünk keretei meghaladják ezek részletesebb bemutatását, ezért a továbbiakban inkább általánosságokra szorítkozhatunk, esetenként szemléltetésül bemutatva az egyes koncepciókból eredő eltéréseket.

a) *s-norma* (unió, t-conorma)

Bevezethető egy „*u*" (union) függvény, mely eleget tesz a következő előírásoknak:

$$u : [0, 1] \times [0, 1] \rightarrow [0, 1],$$

ekkor:

$$\mu_{A\cup B}(x) = u[\mu_A(x), \mu_B(x)] \quad \forall x \in X \qquad (9.34)$$

u-ra a következők adhatók meg:

- $u\ (0, 0) = 0$
  $u\ (0, 1) = \mathrm{u}\ (1, 0) = u\ (1, 1) = 1$
- $u\ (a, b) = u\ (b, a)$
- ha: $a \leq a'$ és $b \leq b'$
  akkor: $u\ (a, b) \leq u\ (a', b')$
- $u\ (u\ (a, b)\ c) = u\ (a, u\ (b, c))$,

tehát érvényes rá a: kommutativitás, monotonitás és aszociativitás

Példák változatokra:
Yager – család s-norma:

$$u_w^y(a,b) = \min[1, (a^w + b^w)^{1/w}] \qquad w \in (0, \infty)$$

- az esetben, ha: $w \to \infty$ felé tart, határesetben átmegy a Zadeh-féle: max (a, b) műveletbe
- az esetben, ha: $w = 1$, átmegy az alábbi „korlátozott összeg" kifejezésbe:

$$u_1^y(a,b) = \min\ [1, (a + b)]$$

Hamacher – család s-norma:

$$u_\gamma^H(a,b) = \frac{a + b - (2 - \gamma) \cdot a \cdot b}{1 - (1 - \gamma) \cdot a \cdot b} \qquad \gamma \in (0, \infty)$$

- az esetben, ha: $\gamma = 1$, átmegy az *algebrai unió* kifejezésbe:

$$u_1^H(a,b) = \mathrm{a} + \mathrm{b} - \mathrm{a} \cdot \mathrm{b}$$

b) *t-norma* (metszet, intersection)

Bevezethető egy „i" (intersection) függvény, mely eleget tesz a következő előírásoknak:

$$i\colon [0, 1] \times [0, 1] \to [0, 1]$$

ekkor:

$$\mu_A \cap_B(x) = i\ [\mu_A(x), \mu_B(x)] \quad \forall\ x \in X \qquad (9.35)$$

i-re a következők adhatók meg:

- $i\,(1, 1) = 1$
  $i\,(0, 1) = i\,(1, 0) = i\,(0, 0) = 0$
- továbbá, hasonlóan az s-normához, itt is érvényes a:
  - kommutativitás
  - monotonitás
  - aszociativitás

Példák változatokra:
Yager – család t-norma:

$$i_w^y(a,b) = 1 - \min[1, ((1-a)^w + (1-b)^w)^{1/w}] \qquad w \in (0, \infty)$$

- az esetben, ha: $w \to \infty$ felé tart, határesetben átmegy a Zadeh-féle: min (a, b) műveletbe.

Hamacher – család t-norma:

$$i_\gamma^H(a,b) = \frac{a \cdot b}{\gamma + (1-\gamma) \cdot (a + b - a - b)} \qquad \gamma \in (0, \infty)$$

- az esetben, ha $\gamma = 1$, átmegy a geometriai t-normába:

$$i_1^H(a,b) = a \cdot b$$

c) *Komplemens* (complement)

Bevezethető egy „n" (non) függvény, mely eleget tesz a következő előírásoknak:

$$n\colon [0, 1] \to [0, 1],$$

ekkor:

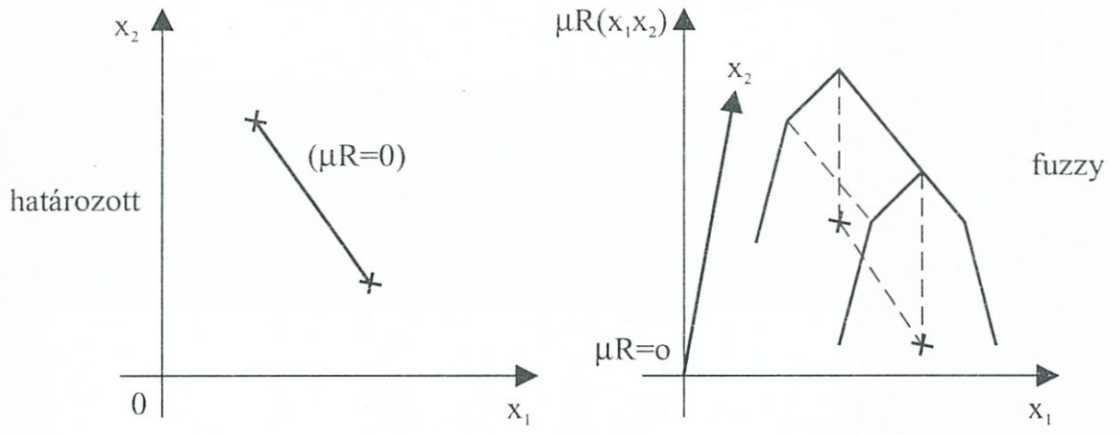
$$\mu_{\overline{A}}(x) = n\,(\mu_A(x)) \qquad \forall\, x \in X \tag{9.36}$$

n-re a következők adhatók meg:

- $n\,(0) = 1$
  $n\,(1) = 0$
- $\forall\, a, b \in [0, 1]$, ha $a < b$, akkor $n\,(a) \geq n\,(b)$
  tehát *nem monoton* növekvő.

9.5.2. Fuzzy relációk

A *relációk* a halmazelemek közötti kapcsolatok meglétéről tájékoztatnak. A *fuzzy relációk* a halmazok elemeinek összerendeltségi *mértékét* határozzák meg. Egy fuzzy reláció, melyet n darab halmaz



9-36. ábra *Határozott és fuzzy relációk*

között értelmezünk, az n dimenziós tér pontjaihoz rendel μ_R tagsági függvényeket a 9–36. ábrának megfelelően.

Amennyiben $X_1, X_2, \ldots, X_n$ halmazok referencia-osztályai az $A_1, A_2, \ldots, A_n$ fuzzy halmazoknak, akkor az $A_1, A_2, \ldots, A_n$ halmazok *fuzzy-relációja* a következő:

$$R(A_1, \ldots A_n) = \{((x_1, \ldots, x_n, \mu_{R(A1, \ldots An)}(x_1, \ldots, x_n)) \mid \mid (x_1, \ldots, x_n) \in X_1 \times \ldots \times X_n\} \quad (9.37)$$

ahol: $\mu_{R(A1, \ldots An)} x_1, \ldots, x_n = \min(\mu_A(x_1), \ldots \mu_A(x_n))$
és: $X = X_1 \times X_2 \times \ldots \times X_n$ a direkt (Descartes) szorzat.

Két, tetszőleges halmaz relációját *bináris relációnak* nevezzük.

Diszkrét, véges elemszámú fuzzy halmazoknál a relációt gyakran a μ_i tagsági függvények *mátrixával* adják meg.

9.19. példa: Legyen adva: $X_1 = X_2 = \{1, 2, 3\}$, tovább: $A = 0{,}5/1 + 1/2 + 0{,}6/3$, ill. $B = 1/1 + 0{,}6/2 + 0/3$, ekkor az $R(A, B)$ reláció $\mu_{R(A, B)}$ tagsági függvény mátrixa:

$\mu_{R(A,B)}$	b_1	b_2	b_3
a_1	0,5	0,5	0
a_2	1	0,6	0
a_3	0,6	0,6	0

9.5.2.1. Fuzzy relációkkal kapcsolatos fontosabb definíciók

a) Az $X_1 \times X_2 \times \ldots \times X_3$ univerzum elemeit az x *sorozatvektor*-ral jelölhetjük:

$$x = (x_1, x_2, \ldots x_n)$$

b) Ha egy y sorozat az x sorozat elemeinek csak egy részét tartalmazza, akkor y az x-nek *részsorozata.* Jelölése:

$$y \; \pi \; x$$

c) Valamely $R(X_1, X_2, \ldots X_n)$ fuzzy-reláció Y fuzzy halmazra (relációra) vett *vetület*ének (projekciójának) nevezzük azt az $[R{\downarrow}Y]$ fuzzy relációt, melynek tagsági függvénye:

$$\mu_{[R\downarrow Y]}(y) = \max_{x,\,y} \mu_R(x)$$

ahol: $y \; \pi \; x$ fennáll

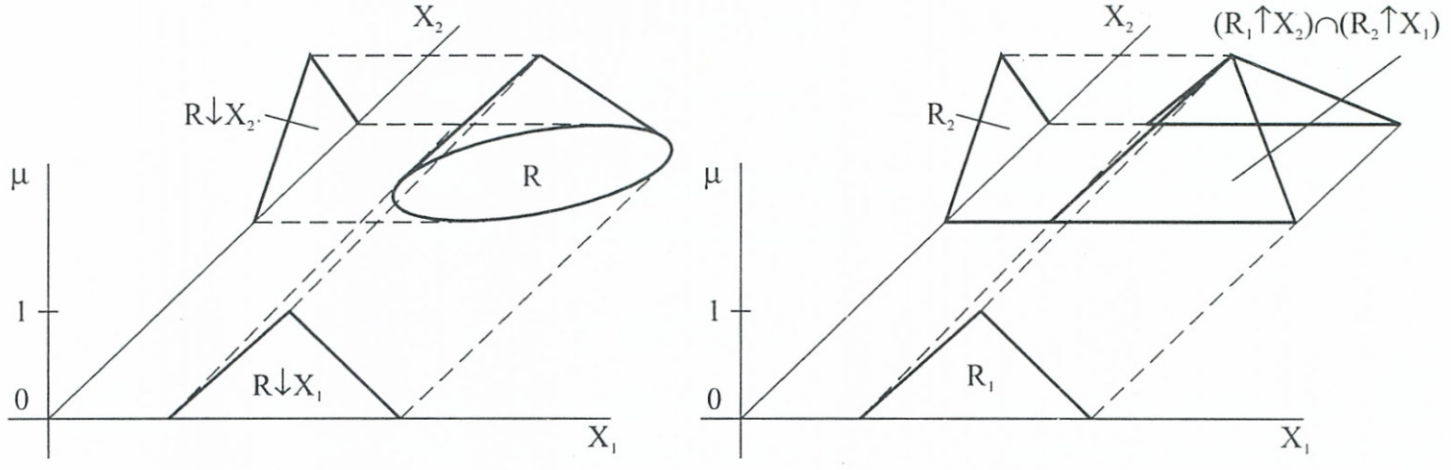
d) Az $R(Y_1, Y_2, \ldots Y_n)$ fuzzy reláció X-re vett *hengeres kiterjesztés*ének nevezzük azt az $[R{\uparrow}X{-}Y]$ fuzzy relációt, melynek tagsági függvénye:

$$\mu_{[R\uparrow X-Y]}(x) = \mu_R(y)$$

valamennyi x-re, ahol fennáll: $y \; \pi \; x$

Ezzel a kiterjesztéssel olyan $X{-}Y$ dimenziókra is kiterjeszthetjük a relációt, ahol eddig nem volt definiálva.

e) Valamely R fuzzy relációt közelíteni lehet az egyes dimenziókra vett vetületeinek hengeres kiterjesztésének metszetével, azok ún. *hengeres lezárt*jával:



9-37. ábra *R közelítése hengeres lezártjával*

$$\mu_{cyl\{R_i\}}(x) = \min_{i \in I} \mu_{[R_i \uparrow X - X_i]}(x)$$

Itt *cyl* (cylinder), ahol $cyl\{R_i\}$ az R_i relációnak $\{ R_i \mid i \in I\}$ vetületeken alapuló hengeres lezártja.

f) Ha egy A fuzzy halmazt (relációt) annak összes lehetséges α vágatával (A_α) adunk meg, akkor ezt *felbontási alak*nak nevezzük.

$$\mu_A(y) = \sup_{\alpha \in (0,1]} \min(\alpha, \mu_{A_\alpha}(x))$$

itt: $\mu_{A\alpha}(x) = 1$, ha $x \in A_\alpha$
$\mu_{A\alpha}(x) = 0$ egyéb esetben

g) Igazolható, hogy az A fuzzy halmaz előállítható az α-vágatok (αA_α) uniójaként, azaz, ha az összes α vágatot ismerjük, a halmaz megadható a tagsági függvények nélkül is. Ez az ún. *megjelenítési tétel.* Például $\alpha = 0{,}5$ esetre a 9–38. ábrán látható.

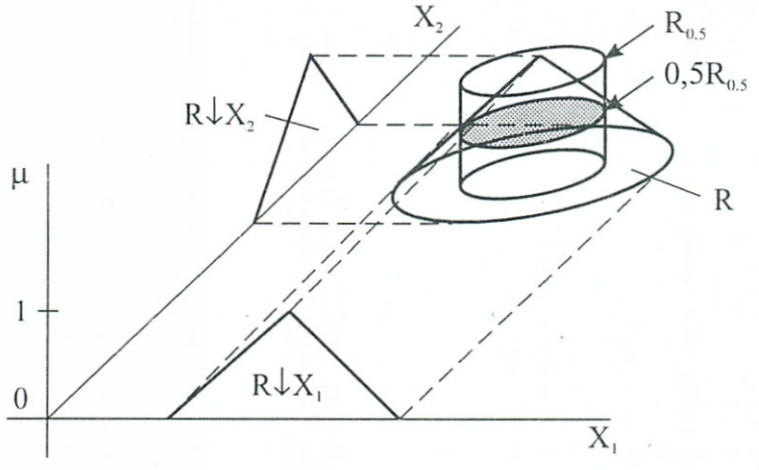
h) Ha a $P(X, Y)$ és $Q(Y, Z)$ két relációt ugyanazon az Y halmazon értelmezzük, akkor a két reláció ún. *kompozíció*ját az alábbi $R(X, Z)$ relációként fogalmazhatjuk meg:

$$R(X, Z) = P(X, Y) \circ Q(Y, Z) \qquad (9.38)$$

ahol: $R(X, Z)$ az $X \times Z$ univerzumon értelmezett reláció, és $(x, z) \in R$, ha létezik legalább egy $y \in Y$, hogy $(x, y) \in P$, és $(y, z) \in Q$.

A kompozícióra felírhatók a következő tulajdonságok:

$$P \circ Q \neq Q \circ P$$
$$(P \circ Q)^{-1} = Q^{-1} \circ P^{-1}$$
$$(P \circ Q) \circ R = P \circ (Q \circ R)$$



9-38. ábra *Az R reláció α-vágata (α=0,5)*

Két fuzzy reláció kompozíciója következtében egy újabb fuzzy reláció adódik, melyhez különböző meggondolások alapján rendelhetünk μ_R tagsági függvényeket. Egy gyakran alkalmazott változat az ún. Zadeh-féle „max–min" kompozíció, melynél a kompozíciós műveletet max [min ...] műveletként értelmezzük.

h1.) *9.20. példa:* A dolgok jobb megvilágítása érdekében vizsgáljunk meg egy példát, amely a 9–39. ábrával kapcsolatos.

Az ábrán látható *X, Y, Z* halmazok között a *P(X, Y)*, *Q (Y, Z)*, *R (X, Z)* fuzzy relációk érvényesek, és az x_i–y_j, y_k–z_l halmazelem-párok közötti μ_P, μ_Q kapcsolatok számszerűen is adva vannak. Az összetevők jelölése:

$$x_i - y_k \text{ kapcsolatnál: } p_{i,k}$$
$$y_k - z_j \text{ kapcsolatnál: } q_{k,j}$$
$$x_i - z_j \text{ kapcsolatnál: } r_{i,j}$$

ahol:
$$r_{i,j} = \max_k [\min(p_{i,k}, q_{k,j})]$$

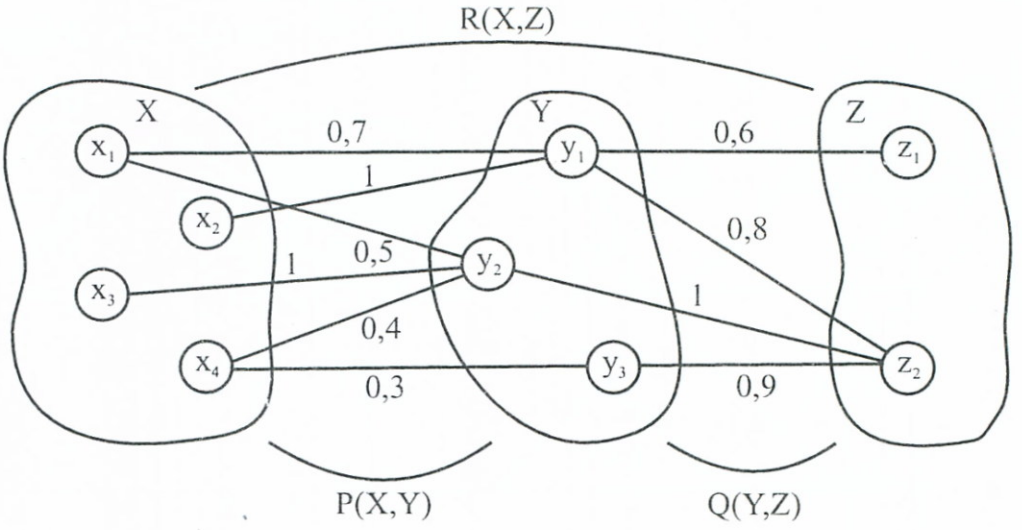
néhány esetet kiszámítva:

$$r_{1,2} = \max [\min (0{,}7,\ 0{,}8),\ \min (0{,}5,\ 1{,}0)] = \max [0{,}7,\ 0{,}5] = 0{,}7$$

$$(x_1 \nearrow y_1 \searrow z_2 = x_1 \rightarrow z_2)$$
$$\searrow y_2 \nearrow$$

$$r_{2,1} = \max [\min (1{,}0,\ 0{,}6)] = 0{,}6$$

$$(x_2 \rightarrow y_1 \rightarrow z_1 = x_2 \rightarrow z_1)$$



9-39. ábra *Bináris fuzzy relációk szemléltetése*

$r_{4,2} = \max [\min (0{,}4,\ 1{,}0),\ \min (0{,}3,\ 0{,}9)] = \max [0{,}4,\ 0{,}3] = 0{,}4$

Amennyiben a komponenseket *mátrixokba* foglaljuk, általánosított formában felírhatjuk a következőket:

$$\mathbf{M}_P = [p_{i,k}],\ \mathbf{M}_Q = [q_{k,j}],\ \mathbf{M}_R = [r_{i,j}]$$
$$R = P \circ Q : [r_{i,j}] = [p_{i,k}] \circ [q_{k,j}] \qquad (9.39)$$

h.2.) A „max-min" típusú kompozíciót Zadeh az ún. „bizonytalan logikai következtetések" kezelésére is alkalmasnak tartotta. A *klasszikus logikában* a HA ..., AKKOR típusú implikációk segítségével vontunk le következtetéseket (lásd: 7.5.5.1. pont). Ezeket a továbbiakban „biztos" logikai következtetéseknek nevezhetjük és jellemzőjük az volt, hogy az ítélet *megegyezett* az implikáció bal oldalával:

pl.: Ítélet : Füst van (U)
Implikáció : HA füst van (U), AKKOR ég a pajta (V))
Következtetés : Ég a pajta (V)

Fentieket a MODUS PONENS következésfajtába soroltuk. Formulárisan így adható meg:

Ítélet : U
Implikáció : HA U, akkor V
Következtetés : V

A formulában U, V *határozott* halmazok.

A *fuzzy logikában* (mivel „elmosódottan", „bizonytalanul" definiált fogalmakkal kell számolnunk) az implikáció úgy módosul, hogy az ítélet *csak hasonlít* az implikáció bal oldalához:

pl. Ítélet : Mintha füstöt látnék (U*)
Implikáció : HA füst van (U), AKKOR ég a pajta (V)
Következtetés : Előfordulhat, hogy ég a pajta (V*)

Ilyenkor a következtetés *bizonytalanságot* takar. Formulárisan megadva:

Ítélet : U*
(hasonlít)
Implikáció : Ha U, akkor V
Következtetés : V*
(bizonytalan)

Ezt a formulát általánosított következtetés-fajtának GENARALISED MODUS PONENS (GMP)-nek nevezzük, melynél az U, U*, V, V*

fuzzy halmazok. A formulákban az U-t *antecedens*nek, a V-t *konzekvens*nek is szokták nevezni.

Az elmondottakkal kapcsolatosak a következő meggondolások: Legyen egy A fuzzy halmaz X-en, egy B pedig Y-on értelmezve. Továbbá legyen egy R fuzzy reláció X, Y-on értelmezve, azaz felírhatók a következők:

$$A = \{(x, m_A(x)) \mid x \in X, m_A(x) : A \rightarrow [0,1]$$

$$B = \{(y, m_B(x)) \mid y \in Y, m_B(x) : B \rightarrow [0,1]$$

$$R = \{(x, y, m_R(x, y) \mid x, y \in X, Y, m_R(x, y) = \min (m_A(x), m_B(y))\}$$

Ezután tételezzük fel, hogy olyan a feladat, hogy az A, R halmazokat ismerjük és keressük B-t. A megoldásnál $B = A \circ R$ kompozíciót alkalmazva, ennek max-min változatával felírható a keresett tagsági függvény:

$$\mu_B(y) = \max [\min (m_A(x), m_R(x, y))] \qquad (9.40)$$

A kapott összefüggés felhasználható „bizonytalan" logikai következtetések tárgyalásánál.

i) Az $R(X, Y)$ fuzzy reláció *értelmezési tartományának* (domain, dom $R(X, Y)$) nevezzük azt az X-en értelmezett fuzzy halmazt, melynek tagsági függvénye:

$$\mu_{\text{dom}R}(x) = \max_{y \in Y} \mu_R(x, y) \quad \forall x \in X$$

j) Az $R(X, Y)$ fuzzy reláció *értékkészletének* (range, ran $R(X, Y)$) nevezzük azt az Y-on értelmezett fuzzy halmazt, melynek tagsági függvénye:

$$\mu_{\text{ran}R}(y) = \max_{x \in X} \mu_R(x, y) \quad \forall y \in Y$$

k) Az $R(X, Y)$ fuzzy reláció *magasság*a (height) az a $h \rightarrow$ valós érték, amely a reláció legmagasabb tagsági foka:

$$h(R) = \max_{y \in Y} \max_{x \in X} \mu_R(x, y)$$

továbbá érvényes:

$$h(R) = h(\text{dom } R) = h(\text{ran } R)$$

normális az R reláció : ha $h(R) = 1$

szubnormális : ha $h(R) = 1$ nem teljesül

l) Az X, Y fuzzy halmazokon értelmezett $R(X, Y)$ bináris relációt *függvény*nek nevezzük, ha nem létezik olyan értelmezési tartománybeli eleme, melyhez a reláció két értékkészletbeli elemet rendelne, azaz:

$$\forall x \in X\text{-hez nem létezik } y_1, y_2 \in Y, \text{ hogy}$$

$$R(x, y_1) > 0 \text{ és } R(x, y_2) > 0, \text{ ahol: } y_1 \neq y_2$$

m) Az $R(X, Y)$ bináris fuzzy reláció *inverzének* nevezzük azt az $R^{-1}(X, Y)$-t, melyre fennáll:

$$\mu_{R^{-1}}(y, x) = \mu_R(x, y) \qquad \forall (x, y) \in X \times Y$$

továbbá felírhatók:

$$\left.\begin{aligned} (R^{-1})^{-1} &= R \\ \text{dom } R(X, Y) &= \text{ran } R^{-1}(X, Y) \\ \text{dom } R^{-1}(X, Y) &= \text{ran } R^{-1}(X, Y) \end{aligned}\right\} \qquad (9.41)$$

9.5.2.2. Fuzzy halmazok geometriai reprezentációja

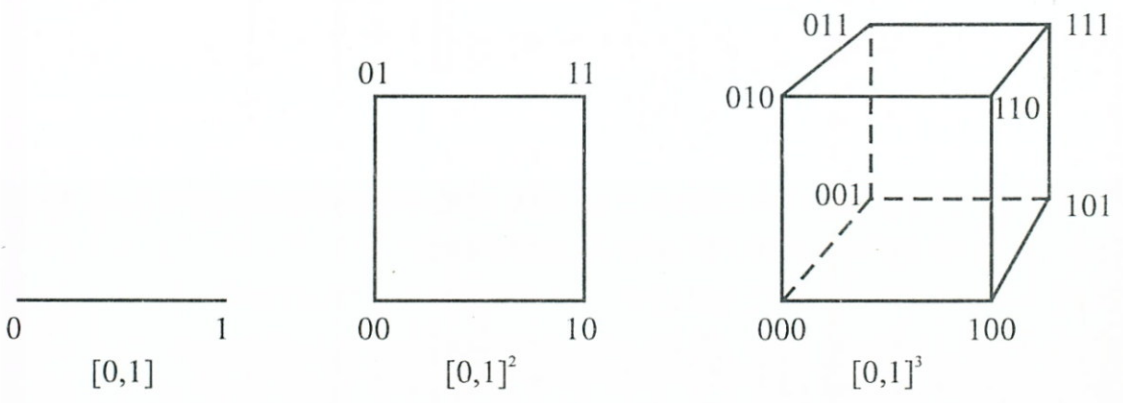
A fuzzy halmazok grafikus ábrázolása egységkockák (hiperkockák) segítségével történik. A határozott halmazok a kockák csúcsain és élein, a fuzzy halmazok a kockák belső pontjain helyezkednek el. Egységkockákat szemléltet a 9–40. ábra

Ha A egy fuzzy halmaz

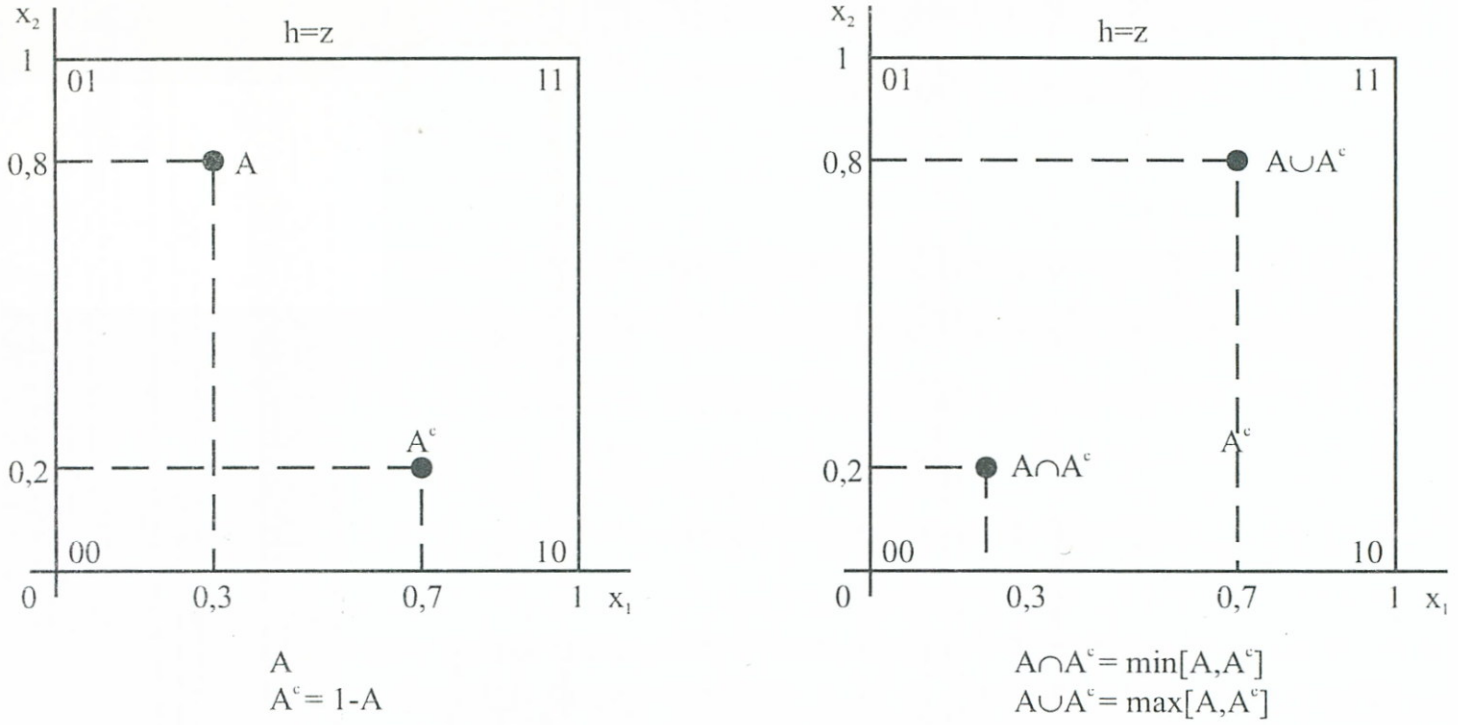
$$A = \{(x, m_A(x)) \mid x \in X, m_A(x) : X \rightarrow [0,1]\}$$

akkor *illesztő vektor*nak nevezzük az alábbi vektort:

$$A(m_A(x_1), \ldots m_A(x_n)) : X \rightarrow [0,1]^n$$



9-40. ábra *$(X,[0,1]^n)$ egységkockák $n = 1,2,3$ esetén*



9-41. ábra *A, A^c, $A \cap A^c$, $A \cup A^c$ grafikus ábrázolása egységkockán*

9.21. példa: Vizsgáljuk meg az alábbi esetek grafikus képét az egységkockán: A, A^c, $A \cup A^c$, $A \cap A^c$
Legyen az illesztő vektor adva az alábbiakban:

$$A(0{,}3\ (x_1),\ 0{,}8\ (x_2)),\ X \rightarrow [0,1]^2$$

A 9–41. ábrán egy egységkockán feltüntettük a fenti kérdéses eseteket.

Az egyes számításokat részletezve a számszerű eredmények:

Ha:	$A =$	: (0,3, 0,8)
akkor:	$A^c = 1 - A$	: (0,7, 0,2)
	$A \cap A^c = \min [A, A^c]$	: (0,3, 0,2)
	$A \cup A^c = \max [A, A^c]$	: (0,7, 0,8)

9.5.2.3. A „fuzzy entropy" fogalma

A 9–41. ábra példájának tapasztalatai alapján elgondolkodtatónak látszik az az eset, ha A (0,5, 0,5) kiindulással oldanánk meg a feladatot, azaz még mélyebbre hatolnánk az egységkocka belsejébe. A 9–41. ábra a.) és b.) eseteiből szemléletesen látható, hogy az A, Ac, A ∩ Ac, A U Ac pontok közelednének egymáshoz és az említett A (0,5, 0,5) elérésekor a hagyományos logikákban eleve elképzelhetetlen:

$$A = A^c = A \cap A^c = A \cup A^c$$

eset adódna. Természetesen hagyományos logikákban (a {0,1} feltétel miatt) csak az egységkocka élein mozoghatunk, így az említett paradoxon nem állhat elő. Más a helyzet a fuzzy esetnél, ahol a [0,1] feltétel miatt bejuthatunk a kocka belsejébe, így elérhetjük a kocka közepét is. A fuzzy esetben azonban – mint láttuk – az A = Ac eset nem elképzelhetetlen, mivel a „félig igaz – félig nem igaz" esettel is számolni tudunk.

Az előzőleg elmondottak alapján tulajdonképpen megfogalmazhatunk egy olyan felismerést, mely szerint minél inkább távolodunk befelé a kocka éleitől, annál inkább nő a fuzzy halmaz bizonytalansága. Ennek számszerű felbecsülésére célszerű bevezetni egy mutatót, melyet *fuzzy entrópiá*nak (fuzzy entropy) fogunk nevezni.

Fuzzy entrópia a következő mennyiség egy adott, A fuzzy halmaz esetén:

$$E(A) = \frac{L_k}{L_t}$$

ahol az egységkocka belsejében mérve:

L_k = „A" távolsága a legközelebbi kockaéltől
L_t = „A" távolsága a legtávolabbi kockaéltől.

Bebizonyítható, hogy fennáll:

$$E(A) = \frac{L_k}{L_t} = \frac{\left|A \cap A^c\right|}{\left|A \cup A^c\right|} \tag{9.42}$$

Az $E(A)$ adat segítségével jellemezhető valamely halmaz bizonytalanságának mértéke. Az összefüggés például határozott halmazok esetén is használható. Ilyenkor, mint tudjuk, a halmaz a hiperkocka valamelyik élén található, ekkor $L_k = 0 \quad (A \cap A^c = 0)$, $L_t = A \cup A^c = 1$, tehát:

$$E_{\text{határozott}} = \frac{\left|A \cap A^c\right|}{\left|A \cup A^c\right|} = \frac{0}{1} = 0$$

tehát *nincs* bizonytalanság.

9.5.3. FUZZY flip-flopok elvi működése és realizációja

A fuzzy tárolóelemek kialakításánál célszerű a kétértékű logikáknál megismert flip-flopok általánosításából kiindulnunk. Az irodalomban gyakran találkozunk az F^3 (Fuzzy–Flip-Flop) jelöléssel is.

Kiindulásul válasszuk a „legintelligensebb" klasszikus kétértékű *J–K* tárolóelemet, melyet minimál konjunktív, ill. diszjunktív alakjaiban a következőképpen írhatunk fel a BOOLE ÉS, VAGY, NEM függvényekkel:

$$Q' = (J+Q).(\overline{K}+\overline{Q})$$

$$Q' = J\cdot\overline{Q}+\overline{K}\cdot Q \qquad \text{(ahol: } Q' = Q_{t+1},\ Q = Q_t\text{)}$$

Ezek a kifejezések a BOOLE algebra értelmében egymással ekvivalensek.

9.5.3.1. A J–K flip-flop kiterjesztése F^3-ra

Formálisan átírva az előzőket, fuzzy ($\cap$,$\cup$, KOMPLEMENS) (9.27), (9.28), (9.29) műveletekbe kapjuk:

$$Q'_S = (J \cup Q) \cap ((1-K) \cup (1-Q)) \tag{9.43}$$

$$Q'_R = (J \cap (1-Q) \cup ((1-K) \cap Q) \tag{9.44}$$

Bebizonyítható, hogy a fuzzy logikában a két kifejezésre az alábbi reláció áll fenn:

$$Q'_S \geq Q'_R, \tag{9.45}$$

azaz – ellentétben a BOOLE változattal – ezek nem mindig ekvivalensek. Emiatt célszerű volt őket az *S*-set, *R*-reset indexeléssel megkülönböztetnünk.

		BOOLE	FUZZY	
J	K	Q'	Q'_R	Q'_S
0	0	Q	Q	Q
0	1	0	0	$Q\cap(1-Q)$
1	0	1	$(1-Q)\cup Q$	1
1	1	$\overline{Q}$	$1-Q$	$1-Q$

9–42. ábra *J–K flip-flop BOOLE és FUZZY definíciós táblái*

A *J–K* flip-flop definíciós tábláját BOOLE és FUZZY esetekre a 9–42. ábra szerint rajzolhatjuk fel:

A fuzzy esetre feltételezve, hogy $Q \in [0,1]$ az előbbi Q'_S, Q'_R kifejezéseinket a következő formába is átírhatjuk:

$$Q' = \begin{cases} (J \cup Q) \cap ((1-K) \cup (1-Q)) \text{ ha: } J \geq K \\ (J \cap (1-Q)) \cup ((1-K) \cap Q) \text{ ha: } J \leq K \end{cases} \quad (9.46)$$

9.5.3.2. Az algebrai F^3

Az előző (9.43), (9.44) formuláinkat átírhatjuk a (9.30), (9.31) műveletekkel képzett formába is, az

$$A \cap B \rightarrow A \cdot B$$

$$A \cup B \rightarrow A + B$$

transzformációk feltételezésével. Ezt a flip-flop változatot *algebrai* F^3-nak (Q^A) fogjuk a továbbiakban nevezni. Így felírva Q_S és Q_R-t:

$$Q_S^A = F_S = J + Q - J \cdot K - J \cdot K \cdot Q - K \cdot Q^2 + J \cdot K \cdot Q^2 \quad (9.47)$$

$$Q_R^A = F_A = J + Q - 2 \cdot J \cdot Q - K \cdot Q + J \cdot Q^2 + J \cdot K \cdot Q - J \cdot K \cdot Q^2 \quad (9.48)$$

A kapott kifejezések egyelőre nehezen lennének kezelhetők, ezért a 9–23. ábra *Q'* tábla-oszlopa alapján célszerű $F(J, K, Q)$-ra a következő kikötéseket tennünk, $Q \rightarrow 1-Q$ helyettesítéssel (9–43. ábra).

	$F(J, K, Q)$	F
*k*1.	$F(0, 0, Q)$	Q
*k*2.	$F(0, 1, Q)$	0
*k*3.	$F(1, 0, Q)$	1
*k*4.	$F(1, 1, Q)$	$1-Q$
*k*5.	$F(0{,}5, 0{,}5, Q)$	0,5

9–43. ábra *Kikötések az algebrai F^3 függvényeinek megoldásához*

A 9–43. ábra *k*1....*k*4. kikötése származik a 9–42. ábrából, míg a *k*5.-öt abból a meggondolásból vettük fel, hogy egy abszolút fuzzy bemenet (maximális entrópia esetén) hasonló fuzzy kimenetet célszerű, hogy eredményezzen.

Ezek után felírhatunk egy *általános paraméteres egyenletet* a (9.47), (9.48) egyenletek megoldása érdekében:

$$F = (\alpha_0 + \alpha_1 J + \alpha_2 K + \alpha_3 JK) + (\beta_0 + \beta_1 J + \beta_2 K + \beta_3 JK)\, Q + \\ + (\gamma_0 + \gamma_1 J + \gamma_2 K + \gamma_3 JK)\, Q^2$$

A 9–43. ábra kikötéseit F-re alkalmazva kapjuk a következőket:

$k1$: $\alpha_0 + \beta_0 Q + \gamma_0 Q^2 = Q$
innen: $\alpha_0 = 0, \beta_0 = 1, \gamma_0 = 0$
$k2$: $\alpha_2 + (1 + \beta_2)\, Q + \gamma_2 Q^2 = 0$
innen: $\alpha_2 = 0, \beta_2 = -1, \gamma_2 = 0$
$k3$: $\alpha_1 + (1 + \beta_1)\, Q + \gamma_1 Q^2 = 1$
innen: $\alpha_1 = 1, \beta_1 = -1, \gamma_1 = 0$
$k4$: $(1 + \alpha_3) + (1-1-1 + \beta_3)\, Q + \gamma_3 Q^2 = 1-Q$
innen: $\alpha_3 = 0, \beta_3 = 0, \gamma_3 = 0$

Fentiek alapján már felírható:

$$F = J + (1-J-K)\, Q = J + Q - J \cdot Q - K \cdot Q = J + Q - (J + K)\, Q \qquad (9.49a)$$

Végül:

$$F = J \cdot (1-Q) + Q \cdot (1-K) \qquad (9.49b)$$

Mivel behelyettesítéssel $k5$ is teljesül, így nem részletezetten igazolható, hogy (9.49a) egyetlen megoldása a (9.47) és (9.48)-nak.

Az F algebrai fuzzy-flip-flopra a következő tulajdonságokat foglalhatjuk össze:

$t1$: F mindig > 0, ha: $J, K, Q \neq 0,1$
$t2$: F mindig < 1, ha: $J, K, Q \neq 0,1$
$t3$: $F(J_1) > F(J_2)$, ha: $J_1 > J_2$ és $Q \neq 1$
$t4$: $F(K_1) < F(K_2)$, ha: $K_1 > K_2$ és $Q \neq 0$
$t5$: A „*D*”-*flip-flop* esete:
$F(D, 1-D, Q) = D$, ha: $J = D, K = 1-D$
$t6$: A „*T*”-*flip-flop* esete:
$F(T, T, Q) = 1-Q$, ha: $J = K = T = 1$
$F(T, T, Q) = Q$, ha: $J = K = T = 0$

9.5.3.3. F^3 általánosított alakban

Amennyiben alkalmazzuk a 9.5.1.2. pont γ.) alpontjának általánosításait a (9.49a) összefüggés átírható a következőképpen:

$$F = J + Q - JQ - KQ = JsQ - KtQ$$

vagy *megfelelő* műveletcsalád megválasztásával felírható még az alábbi formában is:

$$F = \max(J, Q) - \min(K, Q) \qquad (9.50)$$

9.5.3.4. Fuzzy flip-flopok működésének ábrázolása egységkockákkal

A fuzzy flip-flopok működése lényegesen elüt a kétállapotú rendszerek tapasztalataitól. Amiatt, hogy a bemeneti J, K vezérlések és a jelenlegi állapotot képviselő Q „0" és „1" között tetszés szerinti értékekre beállhatnak (esetleg folytonos skálán is) a következő állapotot képviselő Q'-t csak többdimenziós térben tudjuk ábrázolni. A $Q' = f(J, K, Q)$ háromváltozós függvényt rendszerint úgy ábrázolják, hogy fix Q_i értékeknél egységkockán ábrázolják a

$$Q' = f(J, K)_{Q_i}$$

függvényeket, és i-t leggyakrabban: $i = \{0; 0{,}25; 0{,}5; 0{,}75; 1\}$ skála szerint lépcsőzik.

Az előzőkben levezetett: kiterjesztett; algebrai és általánosított F^3-ak mindegyikét más-más formulák képviselik, így (annak ellenére, hogy J–K flip-flopokról van szó) működésük más-más grafikus képet mutat. A számos változat közül:

kiterjesztett $Q'R$-t a 9–44. ábrán,
algebrai Q_S^A-t a 9–45. ábrán,
általánosított F-t a 9–46. ábrán

ábrázoltuk grafikusan.

9.5.3.5. Fuzzy flip-flopok realizációja

A fuzzy flip-flopok lényegesebben összetettebb áramkörök, mint a BOOLE típusú kétállapotú változatok. Ennek egyik oka a fuzzy változók többértékű (gyakran folytonos) értékskálája, másik oka a megoldandó műveletek sokrétűsége. A realizációs hálózatok gyakran keverten tartalmaznak digitális és analóg összetevőket is. Az áramkörök viszonylagos összetettsége a korszerű IC-technológiák korszakában már nem tekinthető akadálynak.

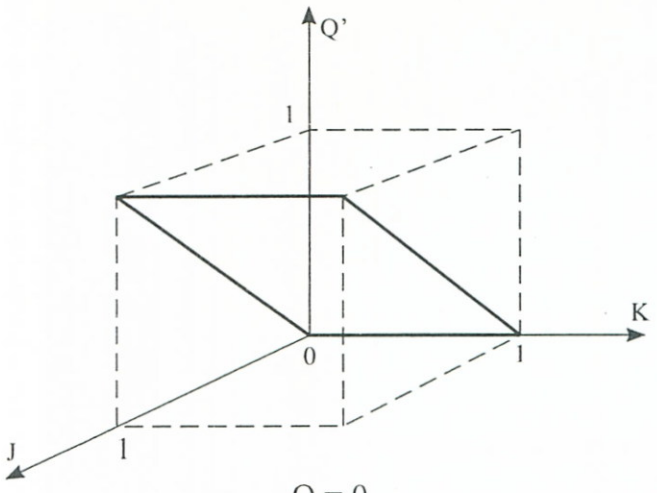
A következőkben az *algebrai fuzzy flip-flop* két realizációs változatát mutatjuk be.

a) Diszkrét módú algebrai fuzzy flip-flop (F^3)

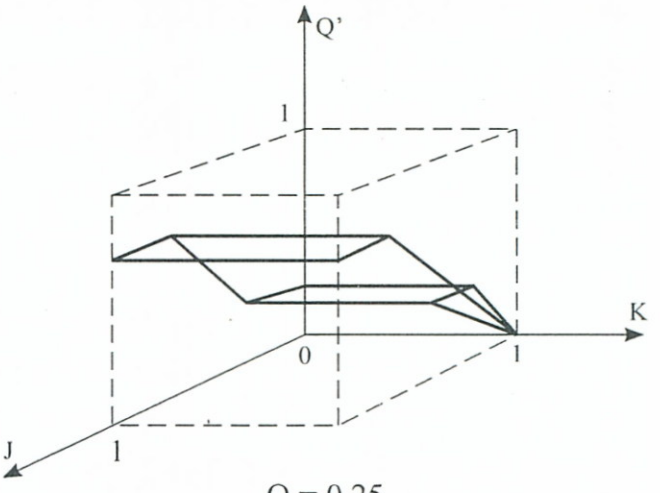
A 9–47. ábrán látható diszkrét módú F^3 a (9.49b) összefüggés alapján lett kialakítva:

$$Q^A = F = J + Q - JQ - KQ$$

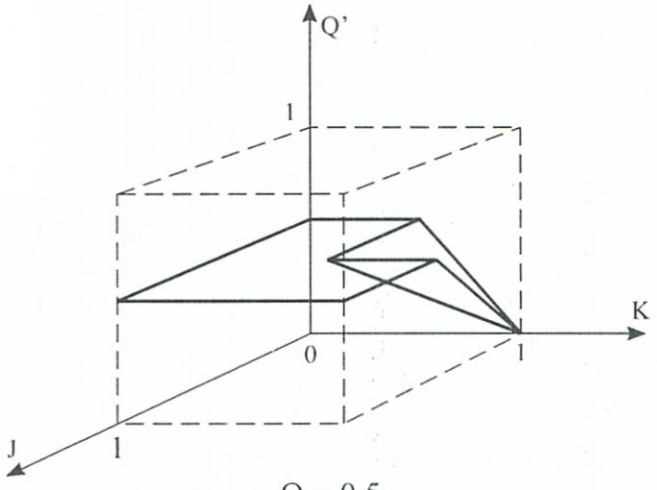
Itt a [0,1]-beli μ(X) tagsági függvényértékek 4 bites formában vannak digitalizálva {0000, 0001, ...1111}. A berendezés két 4×4 bites szor-



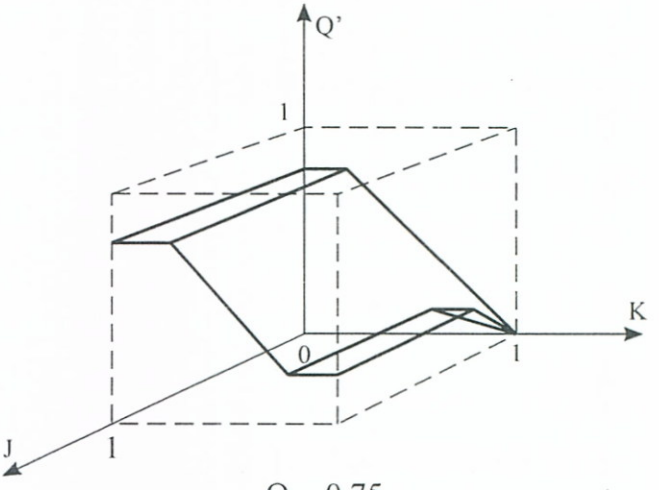
Q = 0



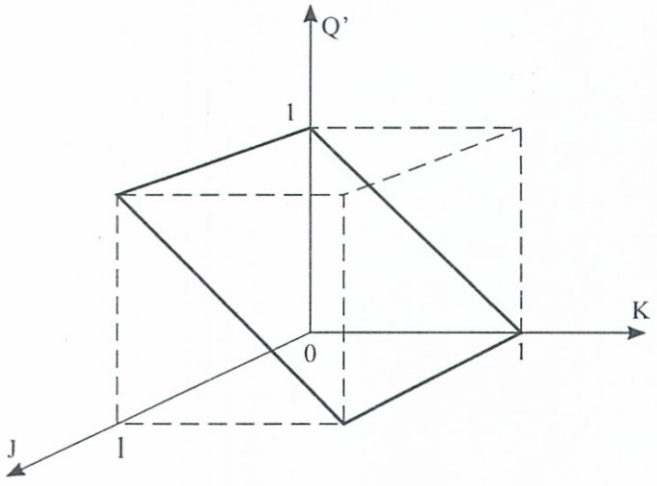
Q = 0,25



Q = 0,5



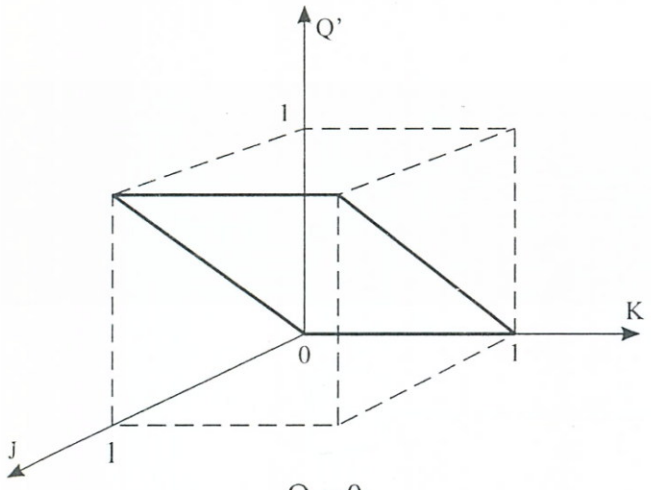
Q = 0,75



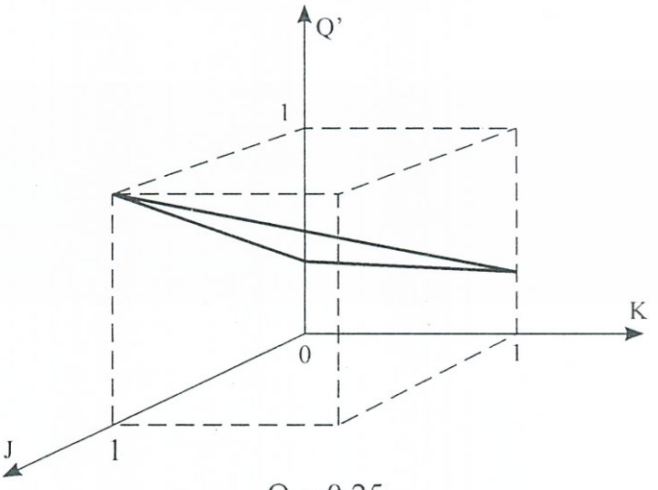
Q =1,0

$Q' = (J \cap (1-Q)) \cup ((1-k) \cap Q)$

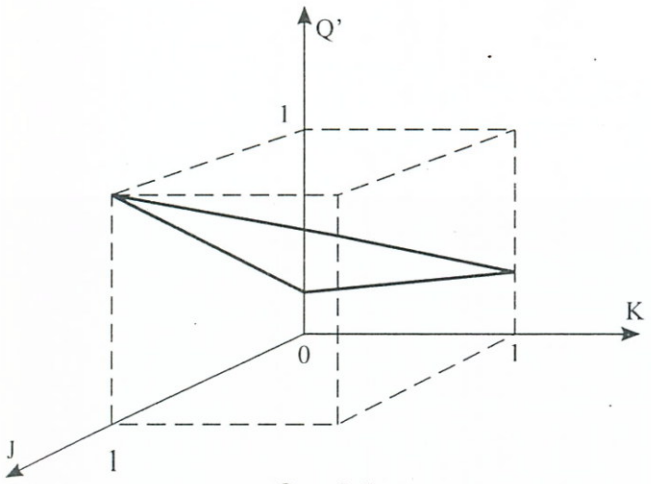
9-44. ábra *Kiterjesztett Q_R Reset típusú fuzzy flip-flop működésének ábrázolása egységkockán*



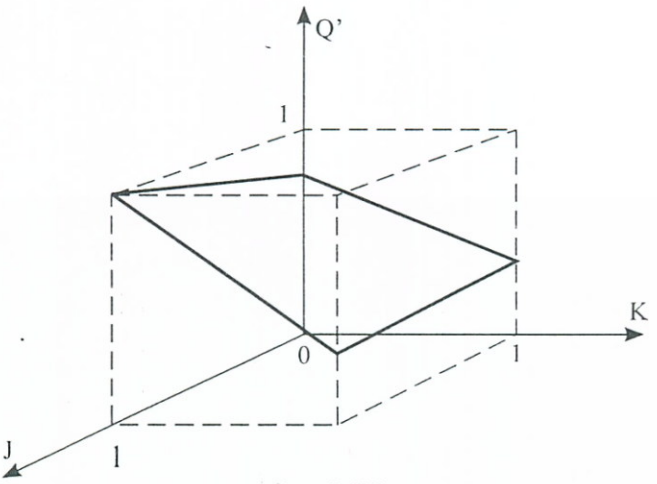
Q = 0



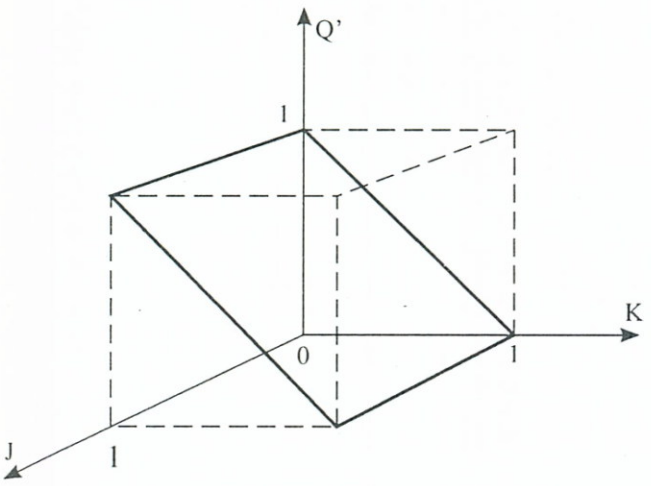
Q = 0,25



Q = 0,5



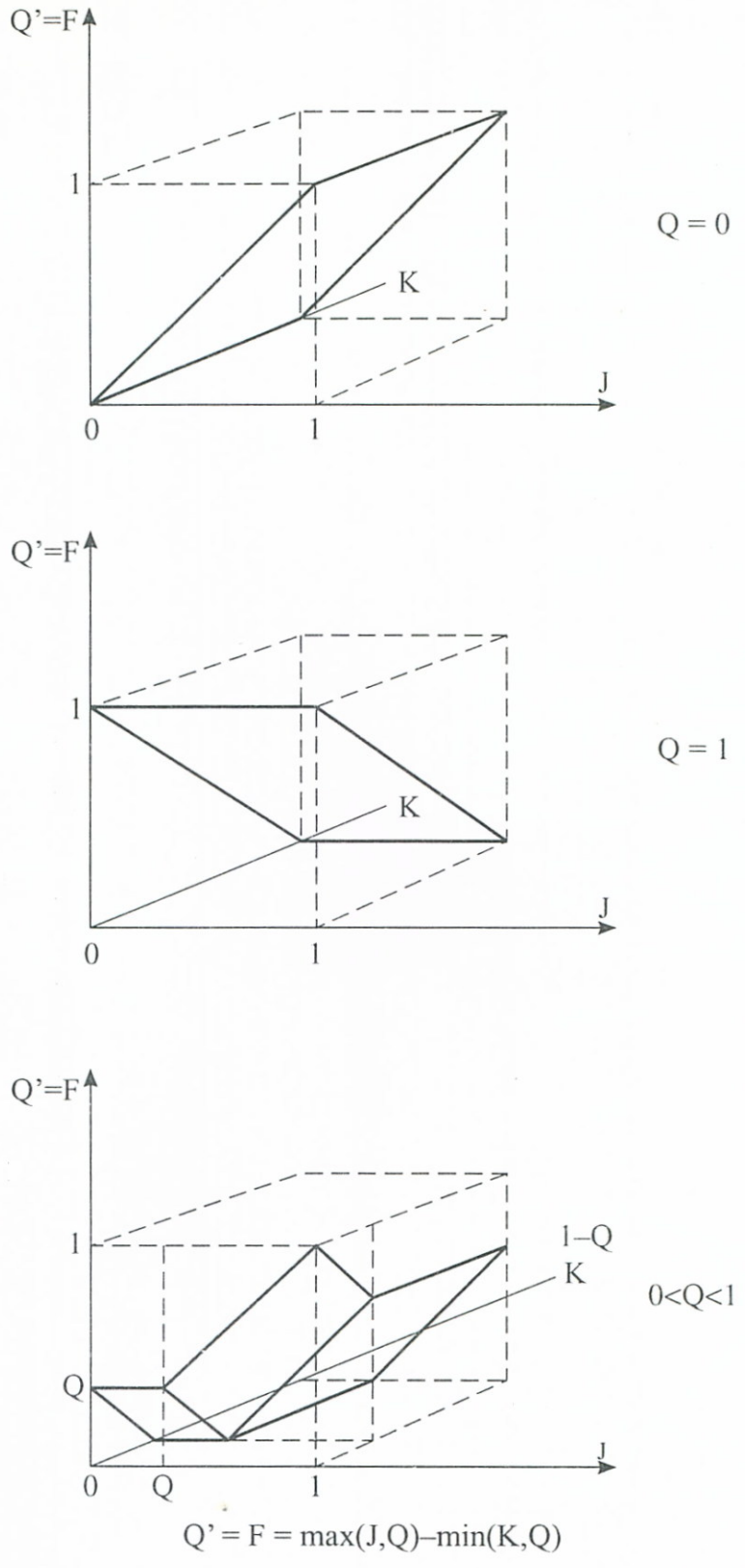
Q = 0,75



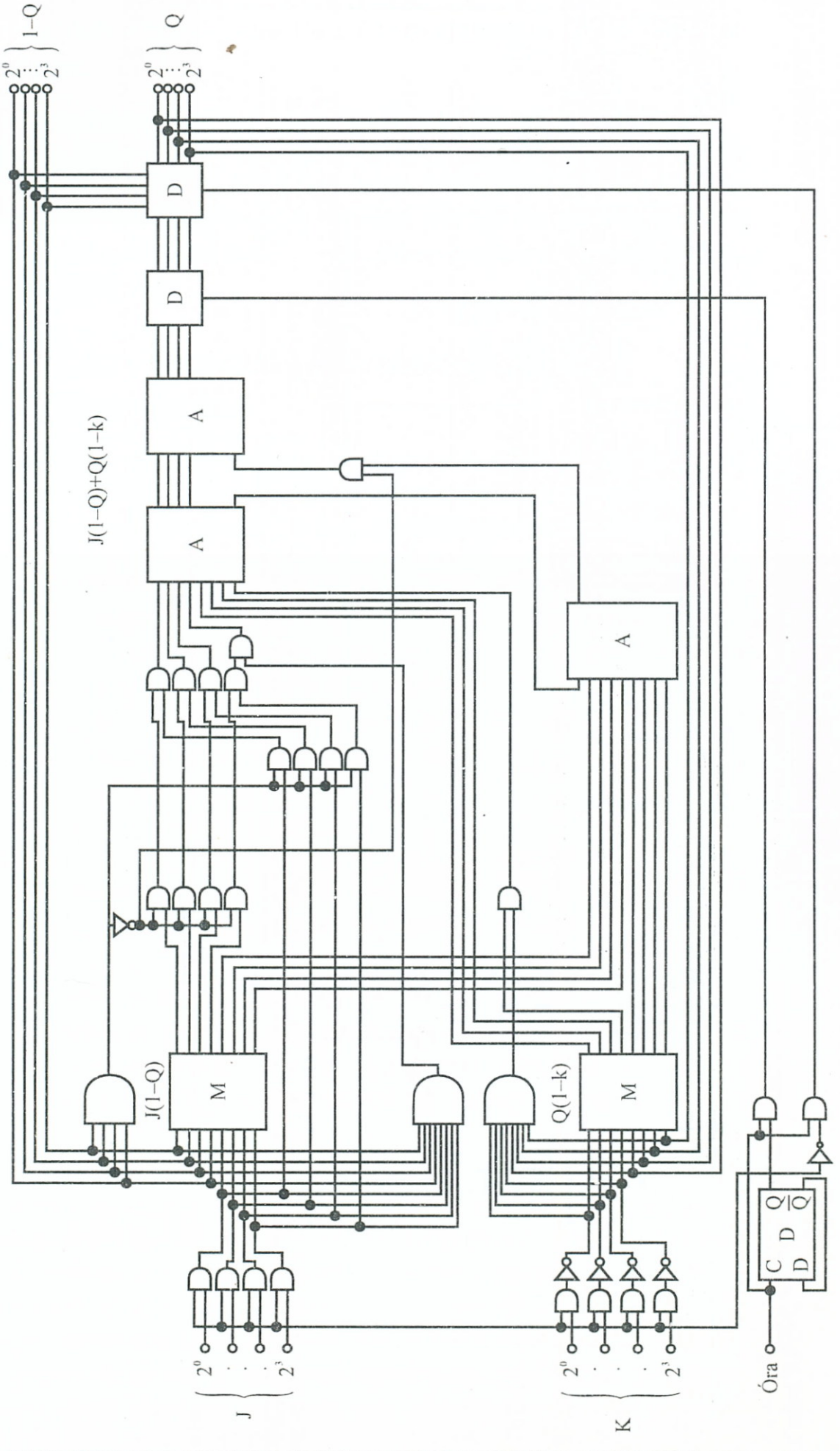
Q =1,0

Q' = J + Q – JQ – KQ

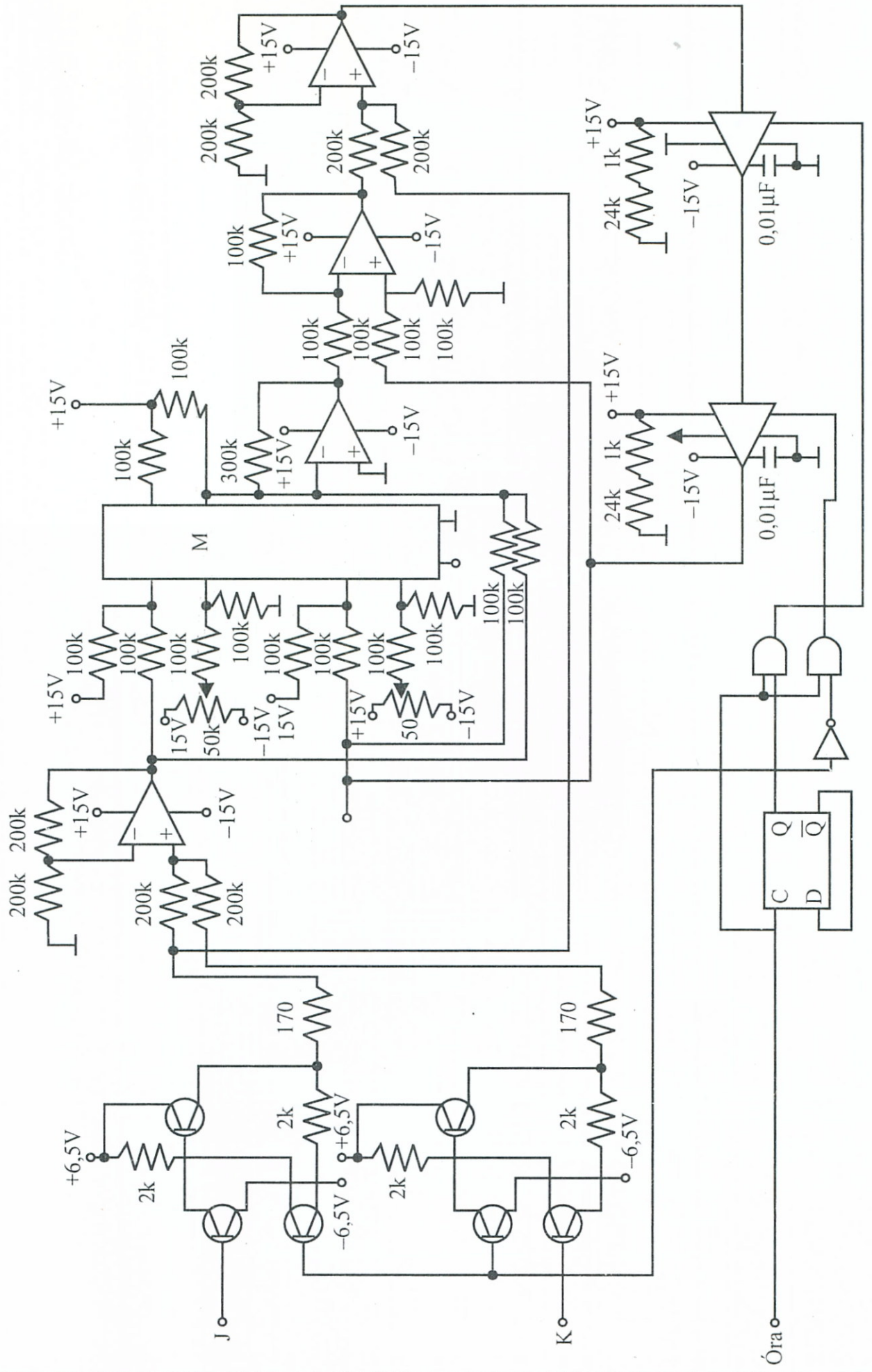
9-45. ábra *Algebrai Q_s Set típusú fuzzy flip-flop működésének ábrázolása egységkockán*



9-46. ábra *Általánosított alakú fuzzy flip-flop működésének ábrázolása egységkockán*



9-47. ábra *Diszkrét módú algebrai fuzzy flip-flop egy digitális realizációja*



9-48. ábra *Folytonos-módú algebrai fuzzy flip-flop egy katalógusból kimásolt analóg realizációja*

zóáramkört (M), 4 bites összeadókat (A) és klasszikus D flip-flopokat (D) tartalmaz.

b) Folytonos módú algebrai fuzzy flip-flop (F^3)

A 9–48. ábrán látható folytonos módú F^3 a (9.49a) összefüggés alapján lett kialakítva. A [0,1]-beli fuzzy értékeket folytonos analóg feszültség [0V, 5V] képviseli. A formula szerinti összeadási, szorzási- és kivonási műveleteket a 4.3. fejezetben megismert analóg műveletvégzők oldják meg.

9.5.4. Fuzzy logikák irányítástechnikai alkalmazása

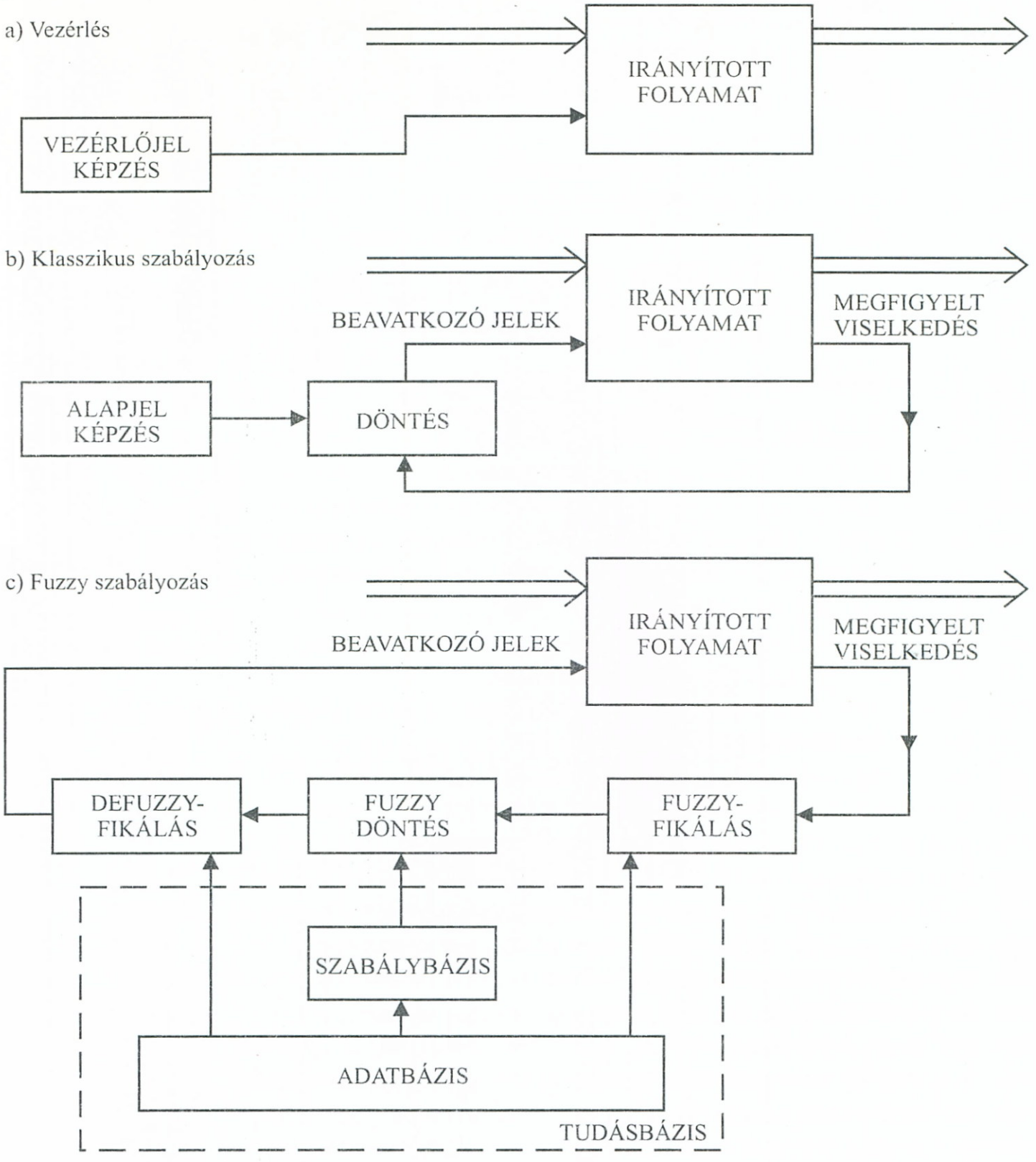
Napjainkban a fuzzy logikák legfontosabb alkalmazási területét az *irányítástechnikai* berendezések képezik. Szerepük e vonatkozásban folyamatosan növekszik. A terjedés különösen olyan esetekben látványos, ahol a klasszikus módszerek a folyamat matematikai modelljének hiányában nehézkesen alkalmazhatók, mivel a fuzzy irányítórendszerekben az irányítás tapasztalatokból és megfigyelésekből nyert szabálybázisának kialakítása nem okvetlenül kívánja meg a matematikai modell ismeretét. A fuzzy irányítórendszer jobban utánozza az emberi következtetési tevékenységet és szillogizmusok (lásd: 7.5.5. pont) formájában oldja meg a feladatokat.

Klasszikus osztályozás szerint az irányító-berendezéseknek két alaptípusát szokták elkülöníteni, ezek elvi vázlatát a 9–49a. és b. ábrákon hasonlíthatjuk össze.

*Vezérlés*nél az irányított folyamat kívánt lefolyás szerinti alakulása a beadott *vezérlő* jelek (vezérlő program) hatására történik.

*Klasszikus szabályozás*nál az irányított folyamat kívánt lefolyás szerinti alakulása lényegében az ún. *alapjelek* szerint történik, de itt a rendszer *megfigyeli* a folyamat tényleges (kimeneti) viselkedését, és ha ez (például valamilyen külső nem várt körülmény hatására) eltér az elvárt viselkedéstől, akkor egy *döntési logika* gondoskodik a *beavatkozó jelek* értelemszerű korrekciót eredményező módosításáról. Mint látható, a szabályozás *visszacsatolt* rendszerstruktúrában valósítható meg.

*Fuzzy-szabályozás*nál (9–49c. ábra) a visszacsatolt rendszerstruktúra ugyancsak megvalósul, de a folyamat (tényleges) *megfigyelt* viselkedésére utaló információkat egy fuzzy transzformációnak *(fuzzyfikálás)* vetjük alá. A transzformációt követően kihasználhatjuk a fuzzy-rendszerek minden, korábban megismert előnyét, így: az emberivel rokon „gon-



9-49. ábra *Az irányítás különféle változatai*

dolkodásmódot", az említett, klasszikus matematikai modell hiányában is létező feladatmegoldó képességet, a tapasztalatok és korábbi megfigyelések hasznosítását és – igen gyakran – a nagyobb feladatmegoldó sebességet. A szükséges korrekciós módosításokat eredményező *döntések* egy ugyancsak fuzzy-orientált *tudásbázis* alapulvéte-

lével hozhatók meg. Mivel e döntések is fuzzy-jellegűek, ezért – végezetül – ezeket még vissza kell transzformálnunk *(defuzzyfikálás)* az irányított folyamat számára „megérthető" *beavatkozó jelekké.*

9.5.4.1. A fuzzy irányítórendszer összetevői

A továbbiakban vizsgáljuk meg kissé részletesebben a fuzzy irányítórendszer fő összetevőit.

a) A FUZZYFIKÁLÁS-t végző „fuzzyfikáló egység" valójában interfészként működik, és feladata a *megfigyelt értékek* (rendszerint konkrét számértékek) fuzzy számokká történő átalakítása. Az egység gyakran adatkonverziót is végez, melynek során az *adatbázis*ban meghatározott alaphalmazra képezi le a megfigyelt értékeket.

*Fuzzyfikáló operátor*nak tekintjük a

$$\text{nem fuzzy} \xrightarrow[op]{} \text{fuzzy}$$

operátort, mely a két halmaz közötti transzformációt megoldja.

Egy gyakran alkalmazott módszer, hogy háromszög alakú μ tagsági függvényt rendelünk az átalakítandó adathoz, oly módon, hogy a várható érték felel meg a csúcsnak és az alap mérete arányos a bizonytalansági intervallum szélességével.

Ha bizonytalanságtól mentes értéket akarunk átalakítani fuzzy halmazzá, akkor érdemes a következő transzformációt választani:

x_0 fuzzyfikálandó érték esetén álljon fenn:

$$\mu_A(x) = 1 \quad \text{ha: } x_0 = 0$$

$$\mu_A(x) = 0 \quad \text{ha: } x_0 \neq 0$$

$$\text{és } \forall x \in X$$

b) A ADATBÁZIS feladata, hogy mind a *megfigyelési,* mind a *következtetési* univerzumok számára tartalmazza a nyelvi változók jellemző adatait, melyek a megfigyelések fuzzyfikálásánál és a következtetések „kiszámításánál" kerülnek felhasználásra.

Mivel a *megfigyelt értékek* folytonosak is lehetnek, ezért ilyenkor szükségessé válhat azok kvantálása. Ez például kisebb volumenű feladatoknál lehet előnyös, ahol a nem túl nagyszámú *megfigyelési* kombinációhoz hozzárendelendő *következtetést* egyetlen táblázatban lehet tárolni megfigyelési–következtetés-párok formájában. Ilyenkor a *döntési* művelet egyszerű táblázatkeresésre egyszerűsödhet.

Ugyancsak az adatbázissal kapcsolatos a *nyelvi változók* definiálása, melynek során a *megfigyelési* és *-következtetési* univerzumok-

hoz tartozó nyelvi változók értékeit és ezek μ tagsági függvényét kell előállítani. A nyelvi értékekre történő felbontást *fuzzy-felosztási* műveletnek nevezzük. A nyelvi értékek száma már meghatározza a maximálisan kialakítható *fuzzy-szabályok számát* is. Amennyiben a *fuzzy-döntés*sel kapcsolatos következtetést *kompozícióval* kívánjuk megoldani, ügyelni kell a „sűrű" lefedést biztosító szabályhalmaz kialakítására.

A nyelvi értékeket *diszkrét* esetben $\mu_i(x_i)/X_i$ elemeket tartozó vektorral, *folytonos* esetben a nyelvi értékek számának megfelelő számú tagsági függvénnyel szokták megadni.

c) A SZABÁLYBÁZIS-nál alkalmazott fuzzy következtetési módszerek elsősorban kompozíciós (9.5.2.1. h.2) *általánosított modus ponens (GMP)* elvre épülnek, ezért a szabálybázis: HA U, AKKOR $V = U \rightarrow V$ típusú szabályokat tárol. Ezekben a szabályokban mind az U (antecedens), mind a V (konzekvens) helyén olyan nyelvi értékek találhatók, melyeket előzőleg a *tudásbázis*ban már bevezettünk. Az antecedens helyén a konkrét *megfigyelések,* a konzekvens helyén a megfigyelésekhez tartozó *következtetések* szerepelnek.

A szabálybázis feladata, hogy az összes fuzzy-szabályhoz tartalmazza a szabályt leíró valamennyi *megfigyelési* és *-következtetési* nyelvi értéket.

A szabálybázisban szereplő szabályok *felállításának* többféle módszere ismeretes:

– *Szakértői tudás* felhasználása:

Miután a nyelvi változók használata jól illeszkedik a GMP szabályaihoz, ezért például egy emberi közreműködéssel már irányított berendezés automatizálásakor jól fuzzyfikálható lehet egy működési leírás, melyből a szabálybázis már összeállítható.

– *A berendezés kezelőjének megfigyelése*

Ilyenre lehet példa egy ember által működtetett berendezéssel kapcsolatos kezelési mozzanatok és tevékenységek összegyűjtése és ezek rendszerezése alapján történő szabálybázis összeállítása.

– *Adott folyamat fuzzy modellje*

A modell alapján összegyűjthetők azok a *bemenő*-jelek, melyek a folyamat meghatározott típusú állapotváltozásait eredményezik.

– *Tanulás*

Az ilyen elven működő, ún. *adaptív* rendszerek, melyek kísérletezésen (a működő rendszer folyamatos megfigyelésén) alapulnak és egy rendszerint már meglévő szabálybázis kiterjesztésére, illetve tökéletesítésére szolgálnak.

c) A FUZZY-DÖNTÉS-eket az ún. „döntési logikai egység" valósítja meg. Ennek feladata, hogy az *adatbázis* adataival és a *szabálybázis* szabályaival kialakítsa a *megfigyelés*hez tartozó fuzzy következtetéseket, melyek lehetnek fuzzy *halmazok,* vagy fuzzy *relációk* is. Fentiekkel kapcsolatosan összefoglalhatók a következők, melyek a „döntési logika" működésénél kerülnek felhasználásra.
Egy szabály (R_i) formátuma a szabálybázisban:

R_i : HA $x = A_i$, AKKOR $y = B_i = A_i \rightarrow B_i$
ahol: $x = x_1 \times x_2 \times \ldots \times x_n$: megfigyelés
$y = y_1 \times y_2 \times \ldots \times ym$: következtetés
$i \in [1, r]$: szabály sorszáma
$A_i = A_{1,i} \times A_{2,i} \times \ldots \times A_{n,i}$: antecedens
$B_i = B_{1,i} \times B_{2,i} \times \ldots \times B_{m,i}$: konzekvens

A teljes szabálybázis (R) az R_i szabályok diszjunkciójaként értelmezhető, azaz felírható a következőképpen:

$$R = A \rightarrow B = \sum_{i=1}^{r} R_i = \sum_{i=1}^{r} (A_i \rightarrow B_i)$$

A döntéssel kapcsolatos következtetés – mint már utaltunk rá – a *megfigyelési* fuzzy halmaz és a *szabálybázis-reláció* kompozíciójaként fogható fel, azaz felírható:

$$y = x \circ R$$

ahol: $x \in X$ megfigyelés fuzzy halmaz
$y \in Y$ következtetés fuzzy halmaz
R a szabálybázis reláció

Ezután például a 9.5.2.1.h. pontban bevezetett Zadeh-féle max-min kompozíció-változat felhasználásával felírhatók a következők is:

$$y = x \circ R$$

$$\mu_{x \circ R}(y) = \max_{x \in X} \min[\mu_x(x), \mu_R(x, y)]$$

és bebizonyítható az a tétel, hogy:

a szabály-relációk unióján számított következtetés megegyezik a szabály relációkon külön-külön számított következtetések uniójával.

d) A DEFUZZYFIKÁLÁS műveletét végző „defuzzyfikáló egység" interfészként működik, csak a „fuzzyfikáló egységhez" viszonyítottan ellenkező értelemben. Feladata, hogy a „döntési logiká"-ból érkező fuzzy halmazokat *visszaalakítsa* a *beavatkozó jelek* határozott értékeivé, melyek az irányított folyamatot a kívánt értelemben befolyásolják.

A defuzzykálásra többféle elven kínálkozik lehetőség:

– *Maximális tagsági értékű elem keresése:*
Itt azt az alaphalmazbeli elemet keressük, melyhez a maximális tagsági érték maximális (y_M):

$$y_M: \quad y_M \in Y, \quad \mu_y(y_M) = \max_{y \in Y} \mu_y(y)$$

A látszólag kézenfekvő módszer hibája, hogy több alaphalmazbeli elemnek lehet ugyanolyan maximális tagsági értéke, így ezek megkülönböztetése nem egyértelmű. Ezért itt vagy úgy döntenek, hogy véletlenszerűen választanak, vagy úgy, hogy az elsőként felismert esetet tekintik a megoldásnak.

– *Maximumok átlagolása:*
Itt nem a maximális tagsági értékű elemet, hanem azok átlagát választjuk a fuzzy halmaz legjellemzőbb elemének.

$$y_M: \quad \{y_M^i \in Y | \mu_y(y_M^i) = \max_{y \in Y} \mu_y(y)\}$$

$$y_M = \sum_{i=1}^{n} \left(\frac{y_M^i}{n} \right)$$

ahol: n = maximális tagsági értékű elemek száma
E módszer hibája lehet például, hogy a vele kiválasztott eset nem is tagja a fuzzy halmaznak (μ értéke = 0), vagy még az univerzumban sem szerepel.

– *Súlypont-keresési módszer*
Itt a maximumok átlagolását a halmaz valamennyi elemére kiterjesztjük. A legjellemzőbb elemet a következő formula adja meg:

$$y_c = \frac{\sum_{y \in Y} y \cdot \mu_y(y)}{\sum_{y \in Y} \mu_y(y)}$$

Itt is még előfordulhat az előbbi módszernél említett hiba eset.

– *Defuzzyfikálás korlátozással*
Ennél a módszernél az előző két módszer hibáját, úgy küszöbölhetjük ki, hogy a fenti kritikus eseteket nem értékeljük (kitiltjuk), hanem a hozzájuk legközelebb eső már érvényes, nem kitiltott elemet értelmezzük.

A fentiekben bemutatott fuzzy irányító-berendezést végleges üzembe helyezése előtt még *„behangolják"*, azaz a paramétereket szimulált, vagy a már megépített berendezésen iterációs módszerrel a kívánt mértékig „pontosítják".

9.E.Ellenőrző kérdések

E.9.1. Mit nevezünk *szimmetrikus* BOOLE-függvénynek, és milyen változatai léteznek?
E.9.2. Mik azok a *skalár-jellemzők* szimmetrikus függvényeknél?
E.9.3. Milyen összefüggéseket tud felsorolni a szimmetrikus függvények átalakításával kapcsolatosan?
E.9.4. Mik azok a *reiteratív* struktúrák?
E.9.5. Milyen módszereket ismer reiteratív hálózatok egyszerűsítésére?
E.9.6. Mik a *pozicionális* hálózatok?
E.9.7. Mit tud a *híd*-struktúrákról?
E.9.8. Mi a *közvetlen összeköttetési* táblázat?
E.9.9. Milyen topológiai lehetőségeket ismer a híd-struktúrák átalakítására?
E.9.10. Milyen elven működik egy *küszöb*-elem?
E.9.11. Mi a *súlyküszöb-vektor*, és mit jelentenek a *súlytényezők*?
E.9.12. Milyen előnyei vannak egy küszöb-elemnek a BOOLE-kapus megoldással szemben?
E.9.13. Mik a hátrányai a küszöb-elemes realizációnak?
E.9.14. Mit jelent a UNATE tulajdonság?
E.9.15. Mi a lineáris szeparálhatóság?

E.9.16. Milyen lépésekből áll egy küszöbfüggvény identifikációs algoritmusa?
E.9.17. Miként állíthatók elő *összetett* küszöbfüggvények?
E.9.18. Mik a *majoritás-* és *minoritás* logikák?
E.9.19. Milyen rokonság van a küszöblogikák és a majoritás-minoritás logikák között?
E.9.20. Mit nevezünk *többértékű* logikának?
E.9.21. Mit tud a POST-függvényről?
E.9.22. Miért jelent problémát a *komplemens* meghatározása többértékű logikáknál?
E.9.23. Milyen általános műveletek definiálhatók többértékű logikák esetén?
E.9.24. Milyen algebrai axiómák foglalhatók össze többértékű logikáknál?
E.9.25. Miként definiálható a komplemens *konstans*ok esetén?
E.9.26. Mi az *unáris* operátor?
E.9.27. Milyen összefüggések érvényesek az unáris operátorra?
E.9.28. Miként adható meg a komplemens *változók esetén*?
E.9.29. Írjon fel egy kanonikus alakot többértékű függvényre.
E.9.30. Milyen kezelést igényelnek a részben meghatározott függvények többértékű logikáknál?
E.9.31. Milyen egy *term-tábla* többértékű logikáknál?
E.9.32. Milyen lépésekbe foglalható össze a többértékű függvények minimalizálása *algebrai* jellemzésnél?
E.9.33. Milyen lépéseket kell elvégezni *grafikus* minimalizálásnál többértékű függvények esetén?
E.9.34. Mutasson be egy *realizációs esetet* az unáris operátorra.
E.9.35. Esetünkben mit takar a *fuzzy* szó?
E.9.36. Mit értünk *nyelvi változó* alatt, mutassuk be egy egyszerű példán.
E.9.37. Mi a különbség a klasszikus és fuzzy-logikák között?
E.9.38. Mi a tagsági függvény?
E.9.39. Fuzzy esetben mi a *core, sup, height?*
E.9.40. Fuzzy esetben mi az a *kardinalitás?*
E.9.41. Mik a *klasszikus fuzzy* műveletek?
E.9.42. Milyen *további fuzzy* műveleteket ismer?
E.9.43. Mi az az *„s" norma, „t" norma?*
E.9.44. Mit értünk fuzzy *reláció* alatt?
E.9.45. Milyen a max-min *kompozíció?*
E.9.46. Mi a különbség a hagyományos és az általánosított MODUS PONENS-ek között?

E.9.47. Hogy jelöljük egy fuzzy reláció *értékkészletét* és *értelmezési tartományát?*
E.9.48. Mit tud a fuzzy halmazok *geometriai* ábrázolásmódjáról?
E.9.49. Mi a *fuzzy entropy?*
E.9.50. Milyen fuzzy flip-flop alapváltozatokat ismer?
E.9.51. Elképzelhető-e *analóg* áramkörökből felépített fuzzy flip-flop?
E.9.52. Melyik technikai területen előnyös a fuzzy logikák alkalmazása napjainkban?
E.9.53. Milyen irányítórendszer változatokat ismer?
E.9.54. Milyen összetevőkből épül fel egy fuzzy szabályozórendszer?
E.9.55. Mi a „fuzzyfikálás", ill. „defuzzyfikálás"?
E.9.56. Mi az adatbázis és szabálybázis feladata fuzzy irányító-rendszerekben?
E.9.57. Mit tud a fuzzy döntéseket megoldó egységről?
E.9.58. Milyen szerepe van az általánosított modus ponens-nek (GMP) a fuzzy szabályozásnál?
E.9.59. Milyen defuzzyfikációs eljárásokat ismer?
E.9.60. Mit értünk a fuzzy irányítórendszer „behangolásán"?

10. FUNKCIONÁLIS EGYSÉGEK

A funkcionális egységek valamely komplex feladatra kialakított, rendszerint moduláris szempontokat is figyelembevevő összetett elektronikus hálózatok.

A funkcionális egységek az előző fejezetekben megismert alkatrészekből, analóg és digitális alapáramkörökből stb. épülnek fel, de egészükben egy magasabb hierarchiájú rendszeralkotási szintet képviselnek, mivel önmagukban is bonyolultabb feladatok (funkciók) megoldására (pl. számlálás, memorizálás, összeadás stb.) és nagyobb mennyiségű információ kezelésére alkalmasak.

Funkcionális egységek alkalmazásával a tervező e funkciókkal építkezve oldja meg feladatát, aminek során a rendszer matematikai modelljét a korábban megismert aritmetikai-logikai műveletek helyett, egy más elvek szerint felépített, a funkcionális egységek által képviselt műveletrendszerre képezi le.

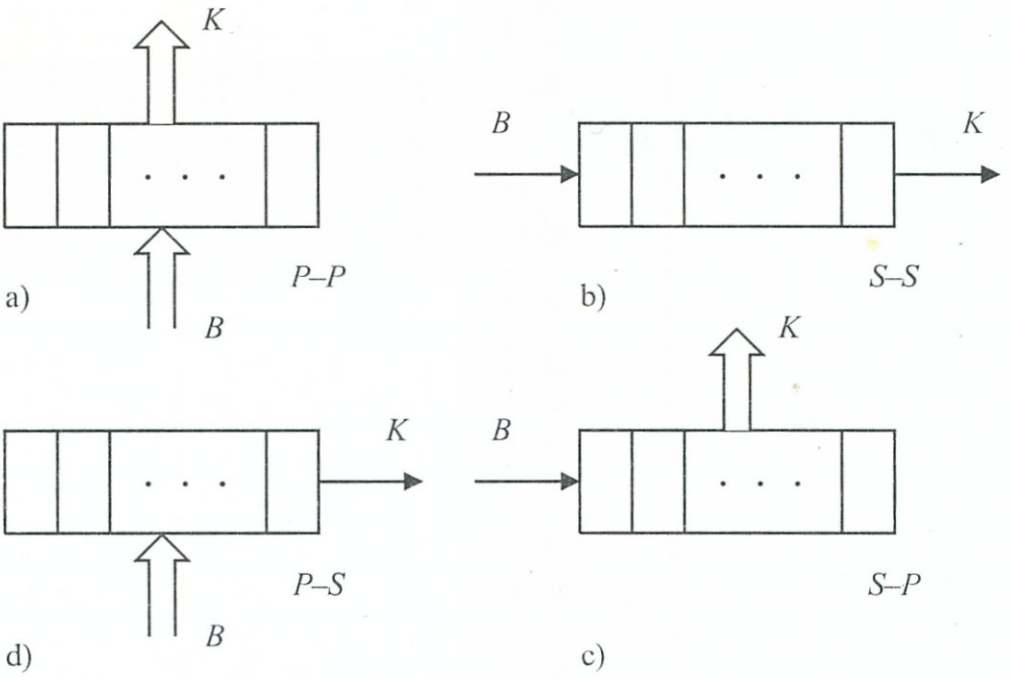
A funkcionális egységek választéka a gyakorlat követelményei alapján alakult ki, és elterjedésüket az integrált áramköri technika fejlődése nagyban elősegítette.

A következő pontokban az elektronikai rendszerek gyakorlatában alkalmazott fontosabb funkcionális egységek tárgyalásával foglalkozunk, súlyozottan kezelve az informatikai alkalmazás vonatkozásában előtérben levő digitális egységeket.

10.1. Regiszterek

A regiszterek sorrendi hálózatból adott típusfeladatra kialakított funkcionális egységek.

Alapvető tulajdonságuk, hogy a beléjük sorosan vagy párhuzamosan bevitt információt meghatározott ideig *tárolni* tudják, esetleg a tárolt információval a regiszter elemi tárolóiban *relatív helyváltoztatást,* ún. *léptetést* („shift"-elést) végezni képesek, végül pedig a tárolt információ soros, ill. párhuzamos kiadására alkalmasak.



10–1. ábra *Regiszterek alapváltozatai*

Alapváltozatok

Az információ-bevitelnek, ill. kivitelnek megfelelően négy alapváltozatot különíthetünk el a 10-1. ábra szerinti értelmezésben.

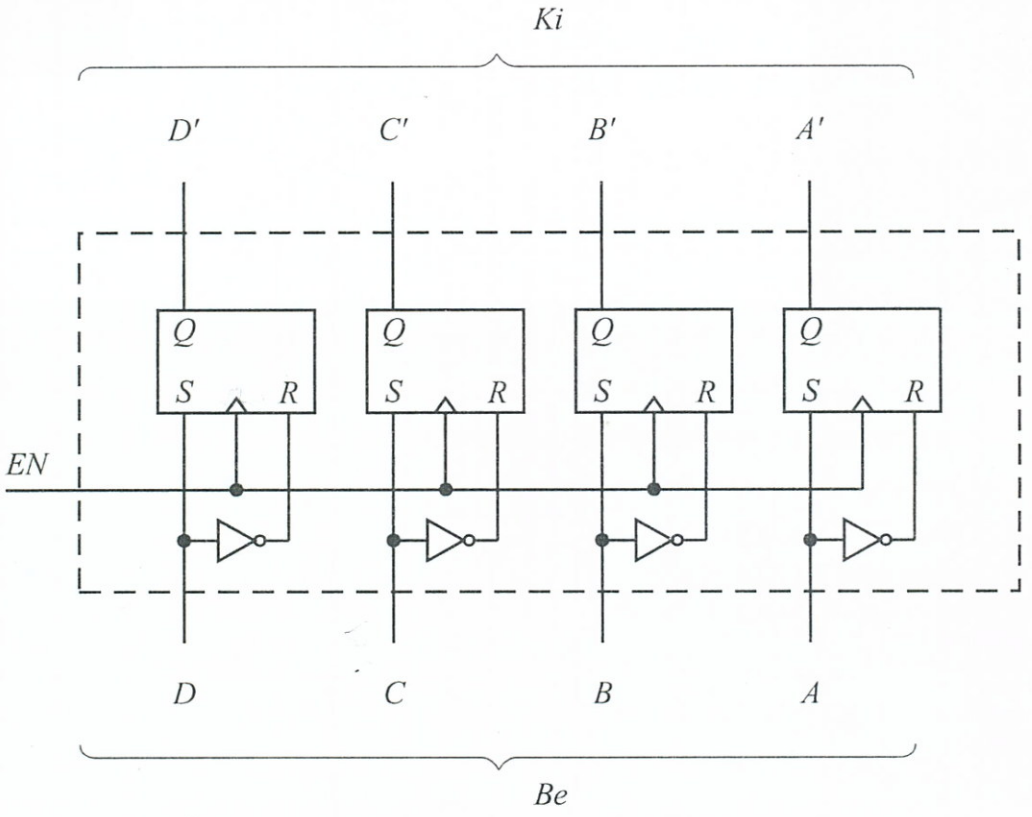
a) *P-P változat:* A párhuzamos beírást információ-tárolási fázis követi. A tárolt információt ugyancsak párhuzamosan olvassuk ki a működés által megkívánt időpontban.

b) *S-S változat:* A bemenetre időben sorosan érkező bemeneti állapotsorozatot a regiszter „lépésenként feltöltődve" („shift"-elve) tárolja, majd szükség esetén (esetleg más léptetési frekvenciával) a kimenetén egy kimeneti állapotsorozat formájában kiadja.

c) *S-P változat:* A bemeneti soros állapotsorozatot shift-elés, majd tárolás után, a *párhuzamos* kimeneteken egyidejűleg megjelenő kódelemekből álló *kódszó* formájában adja ki.

d) *P-S változat:* Párhuzamos kódszavas beírást követő tárolás után a kimeneten a shift-elés következtében egy kimeneti állapotsorozat jelenik meg.

– A 10-2. ábrán egy szinkron S–R tárolóelemekből felépülő 4 bit-es P-P típusú regisztert rajzoltunk fel, melynél az adatbeírás egyidejű engedélyezését az órabemenetekre csatlakozó EN engedélyező jel biztosítja. A P-P típusú regisztereket gyakran *tároló* vagy *memória*-regiszternek (STORAGE) nevezik.



10–2. ábra *4 bit-es P–P regiszter*

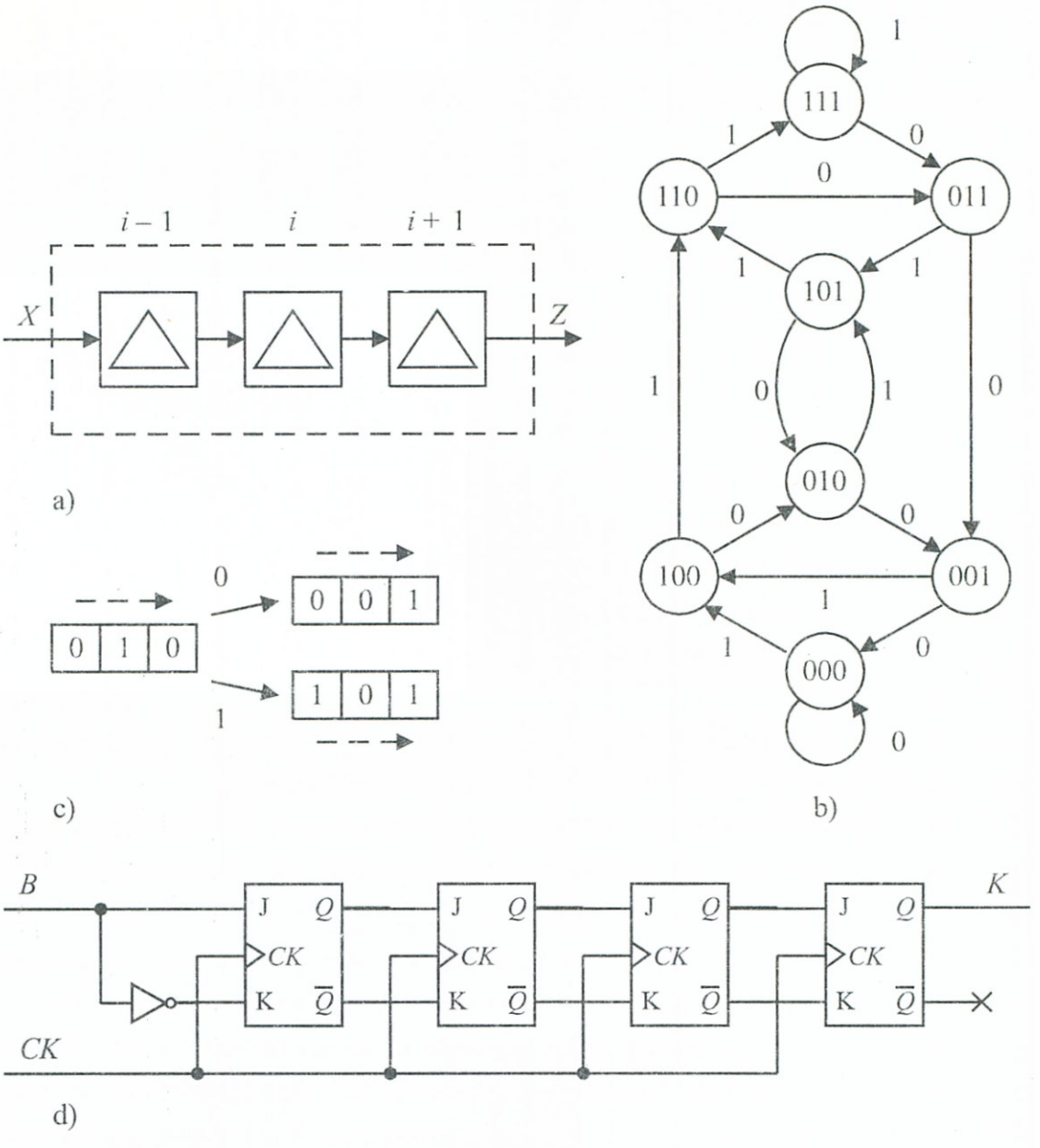
A 10-1. b), c), d) változatok mindegyikénél szükség van a léptetés (shift-elés) műveletére, melynek révén a regiszter tároló-celláiban tárolt információt „bit"-enként jobbra, ill. balra mozgathatjuk. Egy ilyen, ún. *léptető regiszter* elvileg több *késleltető* cella-kaszkád kapcsolásával állítható elő (lásd 10-3. ábra).

Egy hármas „szóhosszúságú" léptetőregiszter elvi vázlatát a 10-3a., állapotdiagramját a b. ábra mutatja. Valamely regiszter állapotegyenlete a következőképpen írható fel:

$$q_{i(T+1)} = q_{(i-1)T} \qquad (10.1)$$

A léptetés mechanizmusát a 10-3c. ábra értelmezi egy olyan esetre, mikor a léptetés a 010 tárolt kódszóból kiindulva 0 vagy 1 újabb bemeneti állapot hatására *eggyel* jobbra halad.

– A 10-3d. ábrán egy szinkron J–K tárolóelemekből felépülő négybites S-S típusú jobbra léptető regiszter logikai vázlata látható. A *B* bemenetre a *CK* órajel ütemében érkező logikai értékek (0, 1) ütemenként eggyel jobbra ugranak, és a *K* kimenethez érkezve sorosan, ütemenként kilépnek. A kapcsolást figyelmesebben megvizsgálva, megállapíthatjuk, hogy az ellenütemű bemenetekkel rendelkező J–K

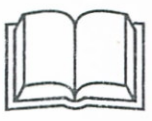


10–3. ábra *Léptetőregiszter működési elve és egy realizációja*

tárolók valójában *D-tárolót* valósítanak meg, mely *szinkronizált késleltetésnek* is tekinthető, így párhuzamba állítható a 10-3a. ábrával.

10.2. Kódolás, kódátalakítók

10.2.1. Információelméleti alapfogalmak



A digitális berendezéseket – általánosított értelmezésben – olyan rendszereknek tekinthetjük, melyeknél az összetevő egységek között *üzenet*-továbbítás folyik. Az üzenetek *információt* hordoznak és információelméleti módszerekkel bebizonyítható, hogy valamely üzenet *információtartalma* (más néven: a benne rejlő információ mennyisé-

ge) attól függ, hogy mekkora *tájékozatlanságot* szüntet meg az üzenet, továbbá, hogy mekkora az üzenetet alkotó szimbólumok bekövetkezésének *valószínűsége*.

– Ennek alapján az *információ egységét* a következőképpen definiáljuk: Egységnyi információt hordoz az az üzenet, mely két szimbólumból álló szimbólumkészlettel rendelkezik és a szimbólumok bekövetkezésének valószínűsége azonos (tehát 50–50%). Az információnak ezt az egységét *bit*nek nevezzük.

Fenti meghatározás más formában úgy is megfogalmazható, hogy két azonos valószínűséggel előforduló szimbólum egyértelmű jellemzéséhez 1 bitnyi információ (pl. egy igen–nem állásfoglalás) szükséges.

Példa: A *bit* fogalmának szemléletesebbé tételére vizsgáljunk meg egy négy, azonos valószínűségű szimbólumból álló szimbólumkészletet, és állapítsuk meg, hogy hány igen–nem döntés (hány bit-információ) szükséges e négy szimbólum valamelyikének egyértelmű elhatárolásához.

Látható, hogy a feladatnál hasonló módszert kell követni, mint a divatos társasjátéknál, ahol a valamit megtudni kívánó kérdező csak igen–nem feleletek alapján tájékozódhat.

Feleljen meg az érdekesség kedvéért a négy szimbólumnak a régi arab mese négy, kívülről teljesen egyforma zsákja, melyek közül háromban homok volt, míg a negyedikben arany. Utóbbit az nyerhette el, aki *legkevesebb* igen–nem választ kívánó kérdés feltevése alapján tudta azt, a négy közül kiválasztani. Az *első* pályázó – nem lévén céltudatos – azt az utat választotta, hogy a zsákokra egyenként rákérdezett: „ebben van-e az arany?" Előfordulhatott volna véletlenül, hogy elsőre el is találja a keresettet, de miután nem volt szerencséje, csak utolsó, harmadik kérdése után tudta a kiválasztást egyértelműen elvégezni. A *második* pályázó módszeresebben állt a dologhoz. A zsákokat gondolatban két kettes csoportba osztályozta, majd első kérdése így szólt: „a bal oldali csoportban van-e az arany?" A kapott akár igen, akár nem kimenetelű válasz alapján már tudhatta, hogy melyik csoport tartalmazza a keresett zsákot és ezután egy második, erre a már csak két zsákot tartalmazó csoportra vonatkozó kérdésével a kiválasztást egyértelműen el tudta végezni. A második pályázónak láthatóan csak *két* kérdést kellett feltennie az első pályázó *három* kérdésével szemben.

Alkalmazzuk a továbbiakban mi is a második pályázó ügyes módszerét tetszés szerinti N szimbólumból álló készlet egyes szimbólu-

mainak egyértelmű elhatárolására. Mint láttuk $N = 4$ szimbólumnál két igen–nem döntés, azaz 2 bit/szimbólumnyi információ szükséges a jellemzéshez.

Általánosságban ezután kimondható, hogy N azonos valószínűségű szimbólum közül egynek a kiválasztásához

$$H = {}^2\log N \quad \text{bit/szimbólum} \tag{10.2}$$

szükséges.

Entrópia:

Általánosságban nem mindig áll fenn, hogy a szimbólumkészlet elemei azonos valószínűséggel szerepelnek (pl. az élő beszédnél nem azonos gyakorisággal használunk minden betűt). Ez esetben, a szimbólumonkénti információ meghatározásához a (10.2.) formulát nem használhatjuk. Matematikailag igazolható, hogy ilyenkor: – valamely üzenet egy szimbólumra vonatkoztatott *átlagos információ tartalma,* ún. *entrópiája* a következő formulával számítható:

$$H = -\sum_{i=1}^{N} P_i \cdot {}^2\log P_i \ \text{bit/szimbólum}, \tag{10.3}$$

ahol: N – a szimbólumok száma

P_i – az i-edik szimbólum előfordulási valószínűsége.

Az entrópia akkor és csak akkor zérus, ha valamelyik szimbólum bekövetkezése biztos.

Az entrópia akkor maximális, ha valamennyi szimbólum azonos valószínűségű, ilyenkor:

$$P_i = P = \frac{1}{N}$$

$$H_{\max} = -\sum_{i=1}^{N} \frac{1}{N} \cdot {}^2\log \frac{1}{N} = -{}^2\log \frac{1}{N} = {}^2\log N \tag{10.4}$$

és mint látható, a (10.2) formulához jutunk.

Redundancia

Valamely üzenetforrás ki nem használt információ mennyiségét *redundanciá*nak nevezzük. Kiszámítása a következő képletekkel történik

$$\Delta H = H_{max} - H \tag{10.5}$$

Ha a redundanciát a maximális entrópiához viszonyítjuk, az ún. *relatív redundanciát* kapjuk:

$$r = \frac{\Delta H}{H_{max}} = 1 - \frac{H}{H_{max}} \tag{10.6}$$

A redundancia *csökkentése* növeli az üzenetforrás szimbólumkészletének kihasználtságát, viszont csökkenti az információátvitel biztonságát, a redundancia *növelésével* ez a viszonylat megfordul.

Memória nélküli és Markov-források

Ha egy tetszés szerint kiragadott üzenet kibocsátása *független* a korábban kibocsátott üzenetektől, akkor a forrást *memória nélküli forrás*nak (zérus memóriájú forrásnak) nevezzük.

Ha egy tetszés szerint kiragadott üzenet kibocsátása a korábban kibocsátott üzenetektől is függ, akkor az ún. *Markov-forrás*ról beszélünk. A korábban kibocsátott üzenetek sorozatát a *forrás állapotának* nevezzük, ez az állapot egy újabb üzenet kibocsátásakor megváltozik. Markov-forrással modellezhető például az értelmes beszéd, ahol az egymást követő hangok statisztikusan összefüggenek.

10.2.2. Kódolási alapok

Általánosságban *kód*-nak nevezzük valamely információ szimbólumokkal történő kifejezését, *kódolás*-nak pedig két szimbólumhalmaz egymáshoz rendelését. A kódolás ellentétes művelete (visszaképezés a kiindulóhalmazra) a *dekódolás*.

Elektronikus rendszerekben leggyakrabban két értékű bináris szimbólumokat (0,1–L, H) használunk a kódolásnál, melyekkel többnyire számokat és betűszimbólumokat fejezünk ki.

– Valamely szimbólumhalmas az elemeiből, értelmes üzenet előállítása érdekében képzett szimbólum-sorozatot *kód-szó*nak nevezünk.

– A kódokat a kódszavak képzéséhez felhasznált szimbólumkészlet alapján két csoportba soroljuk:

bináris a kód, ha felépítése két szimbólum útján,

nem bináris, ha kettőnél több szimbólum felhasználása révén történik.

– A kódszavak hossza alapján megkülönböztetünk:
kötött szóhosszúságú és
változó szóhosszúságú
kódokat.

– A kódoknak csatornán történő átvitele elvileg kétféle üzemmódban lehetséges.

Soros átvitel-ről beszélünk, ha a kódszó elemeit egyetlen csatornán, *időben sorba* állítva továbbítjuk.

Párhuzamos átvitel esetén a kódszó elemeket több csatornán *egyidejűleg* továbbítjuk.

Gyakori a két elv részleges kombinálása is, ekkor *vegyes átvitel*-ről beszélünk.

A hibakorlátozás elvi szempontjai:

Az információ-átvitel célja a következő egyenlőség lehető legjobb biztosítása:

küldött üzenet = érkező üzenet.

Ideális átvitelnél ez az egyenlőség zavartalanul teljesül.

A valóságban az átvitel során fellépő zavarok (pl. hibás áramköri működés) zajos átviteli csatornát teremtenek, csatornaveszteség keletkezik, így az *átvitt információ* csökken.

A jelentkező hibák kiküszöbölésére a rendszerben alkalmazott „bőbeszédűség", azaz a *rendszer redundanciájának növelése* ad lehetőséget. A rendszer redundanciáját a rendszert alkotó áramkörök megfelelő kialakításával, vagy *hibakorlátozó kódrendszer* alkalmazásával növelhetjük.

A hibakorlátozó kódrendszereket két csoportba soroljuk, ezek:

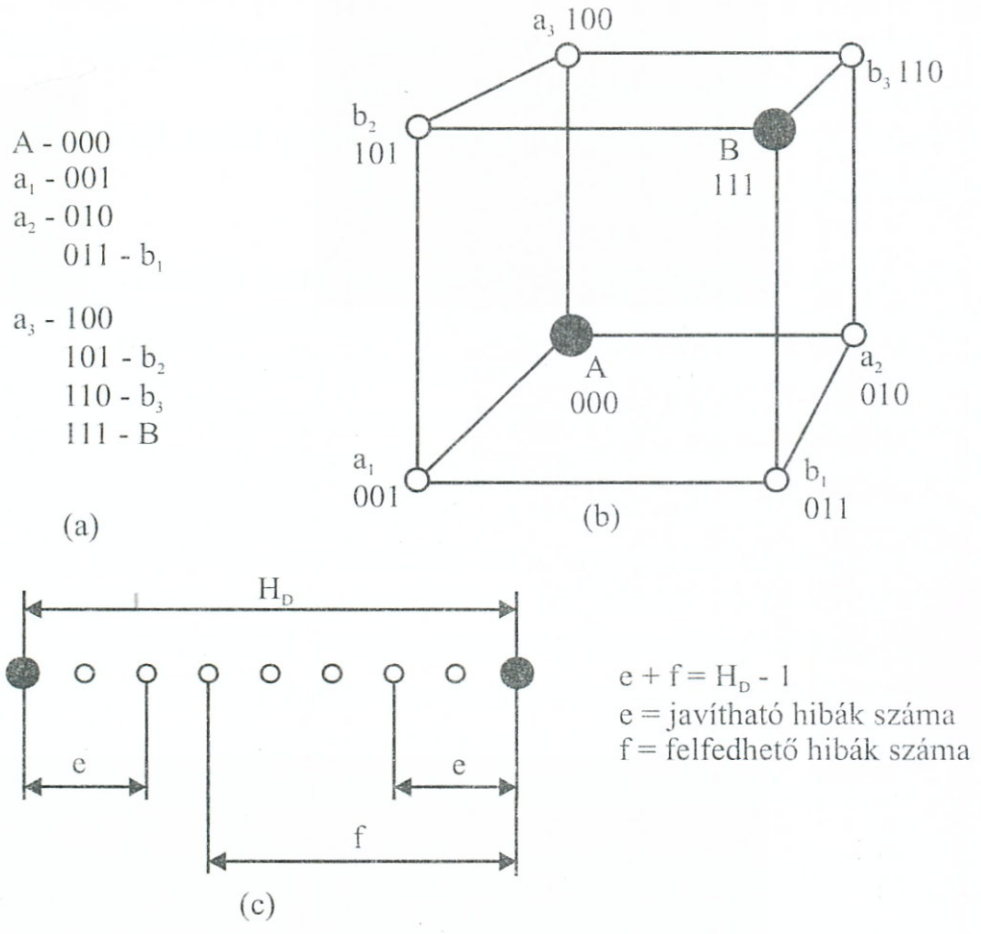
– hiba-felfedő vagy ED (ERROR DETECTING),
– hibajavító vagy EC (ERROR CORRECTING)

kódok.

A következőkben ezeket tesszük vizsgálat tárgyává. A vizsgálatot leszűkítjük a logikai rendszerekben elsősorban érdekes: kötött szóhosszúságú bináris kódokra.

A hiba-korlátozó kódok minőségi jellemzésére bevezetjük az ún. Hamming-távolság fogalmát, melyet H_D-vel jelölünk a továbbiakban.

– Két *kódszó között* annyi a *Hamming-távolság,* ahány kódelemet ellenkezőjére kell változtatni az egyik szóban avégett, hogy a másik kódszóval megegyezővé váljon.



10 - 4. ábra. *Hibafelfedés és hibajavítás*

– Egy *kód* Hamming-távolságán, kódszó készletének elemei közt észlelhető *legkisebb Hamming-távolságot* értjük.

Vizsgáljuk meg egy *szemléltető példa* keretében a hiba-felfedésnek és hiba-javításnak Hamming távolsággal kapcsolatos feltételeit.

Példa: Legyen egy három-elemes csoport-kódunk, melynek kódszókészlete 8 szót tartalmazzon a 10.4a. ábrának megfelelően:

Helyezzük el az a) ábra egyes kódszavait a b) ábrán látható kocka csúcsain úgy, hogy a csúcsok $H_D = 1$ Hamming-távolságra legyenek egymástól. Bináris kódoknál a „zajos" átvitel következtében keletkező hibák úgy jelennek meg, hogy az egyes kódelemek ellenkezőjükre ($0 \rightarrow 1$, $1 \rightarrow 0$) változnak. Ennek alapján a következő megállapításokat érdemes megtennünk:

a) Ha a hasznos üzenetünk szótárába olyan kódszavakat válogattunk a 10.4a. ábra 8 lehetősége közül, melyek *azonos kockaél* végpontjain helyezkednek el (tehát rájuk nézve $H_D = 1$), akkor egyetlen fellépő hiba esetén, a beérkező és valójában hibás kódszót az üzenet–vevő, egy másik hasznos szóként félreértheti. Ilyen eset lép fel például a_2 - 010 és b_3 - 011 hasznos kódszavaknál, mikor a küldött a_2

utolsó elemének zaj következtében történő megváltozása miatt, a vevő b_3-t fog hasznos üzenetként elkönyvelni.

Nyilvánvaló: $H_D = 1$ esetén arra rájönni, hogy hibás átvitel történt, nincs lehetőség.

b) Ha a hasznos üzenetek szótárába most csak olyan szavakat engedünk bekerülni, melyek egymástól *lap-átló,* vagy ennél nagyobb távolságban vannak, $H_D \geq 2$, akkor egy hiba fellépése esetén az üzenetvevő azt tapasztalja, hogy szótárában nem szereplő kódszó érkezett be. A szótártól való eltérésből már következtetni lehet a hiba előfordulás tényére. Például a_1 - 001 és a_2 - 010 hasznos kódok relációjában a_1 utolsó elemének hibája egy nem hasznos A - 000 kódot eredményez.

c) Ha végül hasznos kódszavaknak *test-átlón* elhelyezkedőket választunk $H_D = 3$, akkor még két hiba fellépése sem eredményez másik hasznos üzenettel történő túlfedést, ezért két hiba felfedésére is lehetőség nyílik.

A-t és B-t tekintve például hasznosnak, A-nak két hibája esetén B-től még mindig $H_D = 1$ távolságra leszünk.

d) A *test-átlón* elhelyezkedő hasznos kódszavak már a *hiba-javítás* lehetőségét is magukban rejtik. Ugyanis pl. A-t és B-t tekintve ismét hasznosnak, A-nak egy hibája esetén: a_1, a_2 vagy a_3 szótáron kívüli üzenet keletkezik (és ez fog fennállni B-b_1, b_2, b_3 vonatkozásában is). Mivel a_i-k távolsága A-tól csak 1, B-től pedig 2 (és b_i-kre pedig ez fordítva áll), kialakíthatjuk a dekódoló berendezést olyanra, hogy az a_i-ket A-vá, a b_i-ket B-vé alakítsa az üzenet-vevő számára.

A rendszer *redundanciájának növekedését* tanulságos számszerűleg is megvizsgálni:

Az a) esetben mind a nyolc kódszó felhasználható hasznos üzenetként. A b) esetben már csak négyet tudunk úgy kiválasztani, hogy a $H_D \geq 2$ feltétel teljesüljön, végül a c) esetben már csak két kódszó hasznosítása esetén biztosítható a $H_D = 3$ feltétel.

Az a) és c) szélső eseteket összehasonlítva a következők írhatók fel:

$$H_a = H_{\max} = {}^2\log 8 = 3 \text{ bit/szimbólum}$$

$$H_c = {}^2\log 2 = 1 \text{ bit/szimbólum}$$

A redundanciák:

$$r_a = 0$$

$$r_c = \frac{H_{\max} - H_c}{H_{\max}} = \frac{3-1}{3} = 0{,}666$$

Mint látható, a hibajavítási képesség „ára" a magas redundancia.

Az előző példa alapján általános következtetéseket vonhatunk le. Ezek szerint:

– *hibafelfedés feltétele:* $H_D \geq 2$
– *hibajavítás feltétele:* $H_D \geq 3$ (10.7)

Általánosságban az együttesen felfedhető és javítható hibák számát a 10.4c. ábrában megfogalmazott összefüggésekkel jellemezhetjük.

10.2.3. Alkalmazott kódrendszerek

A kódokat felhasználói oldalról többféle szempont szerint csoportosíthatjuk, aszerint, hogy milyen egymáshoz rendelést valósítanak meg. Az elterjedten használt kódokat nemzetközi szabványokban rögzítik az egységesség érdekében.

A következőkben csupán néhány, digitális technikában gyakran előforduló kódot tekintünk át a 10-5. ábra táblázatának segítségével, melynél a bemutatott kódok a decimális számok szemszögéből lettek csoportosítva.

Hexadecimális kód: Az egyes szimbólumok a megfelelő decimális számot 16-os *radixú* számrendszerben jelölik a konvencionális, szám és betű szimbólumokkal.

Bináris kód: Az egyes kódszavak a megfelelő decimális szám bináris átszámítottjaként állnak elő. Az egyes kód-elemek helyértékenként kettő hatványaival *súlyozottak.* A kódrendszer *additív* tulajdonságú, azaz az összes kódszó „jelentése", mely az összerendelést képviseli, aritmetikai műveletekkel határozható meg.

BCD kód: A BCD számábrázolás szerint a többhelyértékes decimális számok jellemzése helyértékenként történik 0-tól 9-ig terjedő bináris számokkal:

EXCESS 3 kód: (Háromtöbbletes kód) Az egyes kódszavak előállítása az

EXCESS KÓDSZÓ = BIN SZÁM + 3

Dec.	Hexa	BINÉR 8 4 2 1	10^1 B C D 10^0 8421 8421	EXCESS 3 DEC=BIN+3	GRAY	JOHNSON	5-ből 2 7 4 2 1 0
0	0	0 0 0 0	0000 0000	0 0 1 1	0 0 0 0	0 0 0 0 0	1 1 0 0 0
1	1	0 0 0 1	0000 0001	0 1 0 0	0 0 0 1	0 0 0 0 1	0 0 0 1 1
2	2	0 0 1 0	0000 0010	0 1 0 1	0 0 1 1	0 0 0 1 1	0 0 1 0 1
3	3	0 0 1 1	0000 0011	0 1 1 0	0 0 1 0	0 0 1 1 1	0 0 1 1 0
4	4	0 1 0 0	0000 0100	0 1 1 1	0 1 1 0	0 1 1 1 1	0 1 0 0 1
5	5	0 1 0 1	0000 0101	1 0 0 0	0 1 1 1	1 1 1 1 1	0 1 0 1 0
6	6	0 1 1 0	0000 0110	1 0 0 1	0 1 0 1	1 1 1 1 0	0 1 1 0 0
7	7	0 1 1 1	0000 0111	1 0 1 0	0 1 0 0	1 1 1 0 0	1 0 0 0 1
8	8	1 0 0 0	0000 1000	1 0 1 1	1 1 0 0	1 1 0 0 0	1 0 0 1 0
9	9	1 0 0 1	0000 1001	1 1 0 0	1 1 0 1	1 0 0 0 0	1 0 1 0 0
10	A	1 0 1 0	0001 0000	.	1 1 1 1	.	
11	B	1 0 1 1	0001 0001		1 1 1 0	.	
12	C	1 1 0 0	0001 0010		1 0 1 0		
13	D	1 1 0 1	0001 0011		1 0 1 1		
14	E	1 1 1 0	0001 0100		1 0 0 1		
15	F	1 1 1 1	0001 0101		1 0 0 0		
.	.		.		.		
.	.		.		.		
.	.		.		.		

10–5. ábra *Néhány kódrendszer*

formula alapján történik a 10 alatti tartományban. A kód jellegzetessége, hogy a 4-es és 5-ös számok közt egy elválasztó vonal húzható, melyre tükrözve a táblázatot az egymással fedésbe kerülő kódszavaknál 0-val mindig 1, illetve 1-gyel mindig 0 találkozik.

GRAY-kód: Legjellemzőbb tulajdonsága, hogy két „sorrendszomszédos" kódszó mindig csak egyetlen kód-elemnél térhet el egymástól. A szókészlet tetszés szerint bővíthető, betartva az ábrán látható „1-es mintázat" törvényszerűségeit.

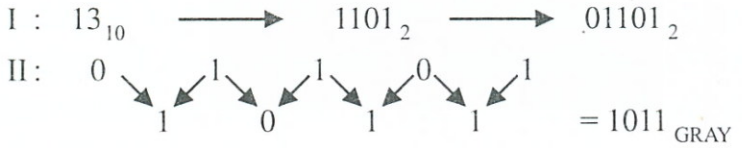
Az eddigi első három kód átalakítása decimálisba, vagy viszont a definíciók alapján nem okoz nehézséget az additív tulajdonságok jóvoltából. A GRAY-kód konverziói valamivel bonyolultabbak, a következő két algoritmus ezeket szemlélteti:

a) DEC-BIN-GRAY átalakítás:

I. lépés: DEC/BIN átalakítás, és legelőre egy 0-át írunk.

II. lépés: A kapott BIN kód két szomszédos elemét úgy párosítjuk, hogy *azonos* szomszédoknál az eredmény: 0, *eltérő* szomszédoknál pedig: 1.

Pl.:



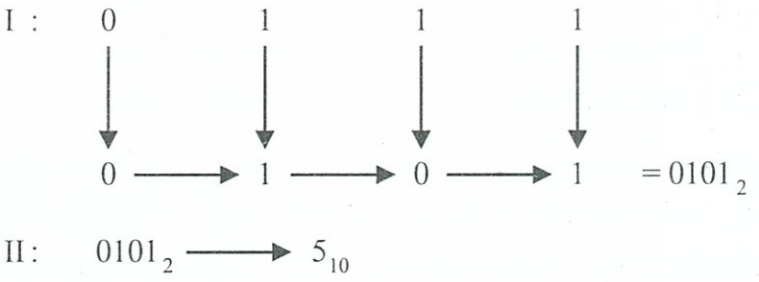
(10.8)

b) GRAY-BIN-DEC átalakítás:

I. lépés: A kiinduló GRAY alak legnagyobb helyértéke megegyezik az új BIN legnagyobb helyértékével, majd az eggyel alacsonyabb helyértékek meghatározásánál a nagyobb helyértéknél kapott eredményt párosítjuk az aktuális GRAY elemmel. *Egyforma* párnál az eredmény: 0, *eltérőnél:* 1.

II. lépés: BIN/DEC átalakítás

Pl.: Adva: 0111_{GRAY}



(10.9)

JOHNSON kód: A kód a táblázatban látható szabályosságot tartalmazza, így előnyös hardver megvalósításokra nyújt lehetőséget. A szókészlet bővíthető a következő formula szerint:

szavak száma = 2× kód-elemek száma

5-ből 2 kód: Jellegzetessége, hogy minden szóban csak két 1-es szerepel. Kedvező hibafelfedő tulajdonságokkal rendelkezik a szavakban található szabályosság miatt.

Hibafelismerés.

A bináris jelek továbbítása során, átviteli zavar következtében a 0–1 jelek felcserélődhetnek. Ez hibát jelent, melynek kivédésére *hibafelismerő* (sőt *hibajavító*) eljárásokat is kifejlesztettek, melyekkel később foglalkozunk. Legegyszerűbb ezek közül a *paritás-elemes* eljárás, amivel bármely kódrendszer alkalmassá tehető hibafelismerés-

DEC	Kiegészített				
	Eredeti BIN				Paritás
0	0	0	0	0	0
.			.		.
.			.		.
.			.		.
10	1	0	1	0	0
11	1	0	1	1	1
.			.		.
.			.		.
.			.		.
15	1	1	1	1	0

10–6. ábra *Párosra kiegészítés paritás elemmel*

re, amennyiben minden kódszót egy ún. *paritás bit*-tel kiegészítünk. A paritás elemet:

- páros (EVEN)
- páratlan (ODD)

kiegészítőként alkalmazhatjuk, aszerint, hogy a kiegészített kódszóban szereplő 1-esekre (0-ásokra) a páros vagy páratlan szabályosság lesz a jellemző.

A 10-6. ábra ilyen *páros 1-esre* kiegészítő bináris kódot mutat.

A kódolási hibafelismerő-képesség ezúton történő növelésének természetesen ára van. A 10-6. ábra szerinti bináris kód csak 16-féle eset (szimbólum) megkülönböztetésére képes. Valójában pedig 5 elemet használ, mellyel elméletileg $N = 2^5 = 32$ szimbólum megkülönböztetése lenne lehetséges. A kódrendszerben valójában információfelesleg keletkezik, azzal, hogy az elméleti 32 esetből elhagyjuk a páratlan 1-eseket tartalmazókat, azaz tulajdonképpen nem hasznosítjuk a lehetséges szimbólumok 50%-át.

Ha megvizsgálnánk a 10-6. ábra táblázatának valamennyi kódszavát, akkor tapasztalhatnánk, hogy bármelyik két kódszót egymással összehasonlítva kielégülve a (10.7)-ben megfogalmazott hibafelismerési feltétel, azaz $H_D < 2$ Hamming-távolságú párosítást nem találnánk.

Többdimenziós paritás-elemes kód

A paritás-elemes kódok többdimenziós általánosításával a hibakorlátozó képesség növekszik. Pl. a (10.10) táblázatbeli kódnál, ahol a sorok és oszlopok is paritás-elemeket tartalmaznak, egy hiba javításának lehetősége áll fenn. A sorok és oszlopok paritás-ellenőrzéseinek kijelölése révén a hibás elem megtalálható és ellenkező értékükre változtatva javítható.

Információt hordozó kód-elemek	Sorparitás ellenőrzés
0 0 0 1 1	0
1 0 0 0 0	1
0 1 0 0 1	0
1 1 0 1 0	1
1 1 0 1 1	0
1 1 1 1 1	1
0 0 1 0 0	1
Oszlopparitás ellenőrzése	Paritás-elemek ellenőrzése

(10.10)

Hamming-féle hibajavító kód

A Hamming-kódoknál, több paritás-elem alkalmazásával részkódszavak paritásvizsgálatát végezzük el és ezek eredményéből következtetünk a hiba helyére, melyen levő kódszó-elemet ellenkezőjére változtatva, a hibát kijavítjuk.

A hibajavítás elvi menetét egy hibát javító változatnál a 10.7. ábrán követhetjük.

Az egy-egy paritás-elemet (X_1, X_2, X_4) tartalmazó, célszerűen összeállított rész-kódszavak paritásellenőrzését, az E_1, E_2, E_4 blokkokban végezzük el. Paritáshiba esetén a megfelelő blokk vagy blokkok „1”-es elemet jelentetnek meg kimenetükön, hibamentességnél pedig „0”-t. A három blokk kimeneti elemkombinációit az F dekódolóba vezetve, ennek hét kimenete közül azon jelenik meg „1”-es, amellyel egyező betűjelű bemenőcsatornán a hiba jelentkezett. Az így megkeresett hibás csatorna-elemet a megfelelő J javító blokk ellenkezőjére változtatja. Amelyik J blokkba az F-ből „0” érkezik, ott a csatornán beérkező kódszóelemek változatlanul maradnak.

Szemléltetésül vizsgáljuk meg azt az esetet, mikor a hasznos információt hordozó A, B, C, D és a kiegészítő paritás-elemeket hordozó X_1, X_2, X_4 csatornákon a helyes:

X_1	X_2	A	X_4	B	C	D
1	0	1	1	0	1	0

(10.11)

kódszó helyett az alábbi hibás kódszó érkezik:

X_1	X_2	A	X_4	B	C	D
1	0	0	1	0	1	0
		↑ (hiba)				

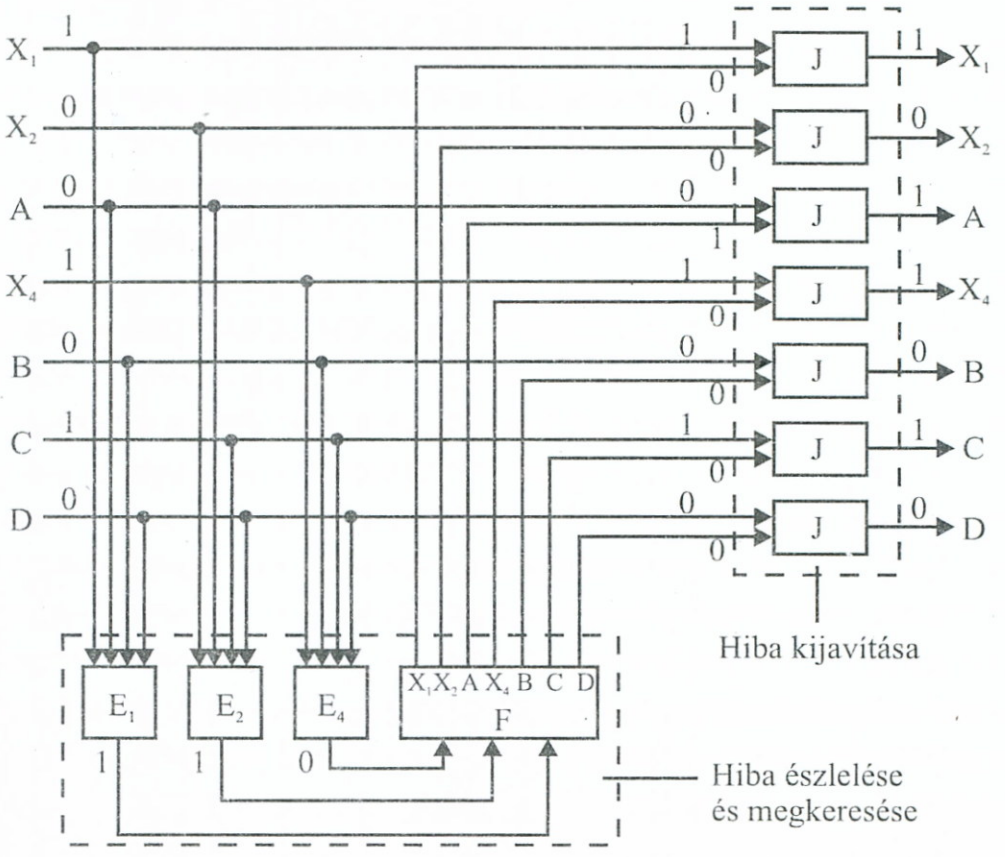
(10.12)

A paritásellenőrzött részkódszavak a következők:

E_1				E_2				E_4			
X_1	A	B	D	X_2	A	C	D	X_4	B	C	D
1	0	0	0	0	0	1	0	1	0	1	0

A paritásellenőrzés eredménye:

E_1	E_2	E_4
1	1	0



10 - 7. ábra

Az F blokkban történő hibakeresés elemi okoskodásra vezethető vissza: Miután E_4-nél nincs hiba (0), ezért (csak egy hiba lehetőségét feltételezve) X_4, B, C és D csatornákon biztos jó elem érkezett. A hiba ezért csak olyan csatornánál lehet, amelyik mindkét hibát jelző blokkban ($E_1 = 1$ és $E_2 = 1$) szerepel, de E_4-ben nem. Ennek a megszorításnak csak az „A" csatorna tesz eleget, tehát a hibás elemnek itt kellett beérkezni. Ez a kiinduló (10.11) és (10.12) összehasonlításával valóban ellenőrizhető is.

A J_A javító blokk – miután F-től jelet kap – „A" csatorna beérkező jelét ellentétesre változtatja, így az összes J-k kimenetének együttese már a javított, helyes kódszót:

$$1\ 0\ 1\ 1\ 0\ 1\ 0$$

szolgáltatja.

A J_i javító blokkok valójában egy egyszerű antivalencia kapuval realizálhatók. Ha ugyanis megállapítottuk, hogy az adott i" helyen nincs hiba, azaz a J_i alsó bemenetén $J_{ialsó} = 0$ van, akkor az antivalencia kapu a $J_{ifelső}$ bemeneti jelét

$$0 \oplus J_{ifelső} = J_{ifelső}$$

összefüggés értelmében változtatás nélkül átengedi.

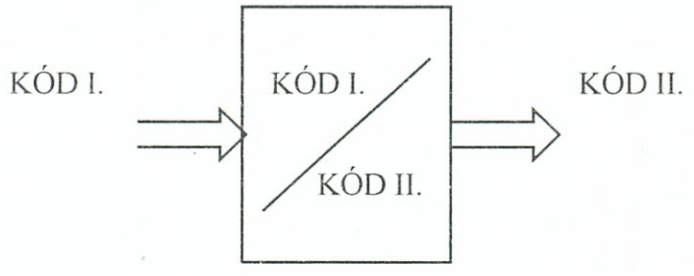
Ha a J_i alsó bemenetére $J_{ialsó} = 1$ érkezik, akkor itt *hiba van,* amit ki kell javítani. A javítás a J_i felső bemenetére érkezett jel invertálásával történik, amit az antivalencia kapu az

$$1 \oplus J_{ifelső} = \bar{J}_{ifelső}$$

összefüggés értelmében el is végez.

10.2.4. Kódátalakítók

A kódátalakítók elvi működése a 10-8. ábra alapján fogalmazható meg, mely szerint a bemenetre érkező KÓD I. kódrendszer-beli kódszavakat a KÓDÁTALAKÍTÓ funkcionális egység a KÓD II. kód-



10–8. ábra *Kódátalakító elvi vázlata*

	KÓD I./KÓD II.	Formulák
a)	BIN/GRAY	$G_{n-1} = B_{n-1}$ $G_i = B_i \oplus B_{i+1}$
b)	GRAY/BIN	$B_{n-1} = G_{n-1}$ $B_i = G_i \oplus B_{i+1}$

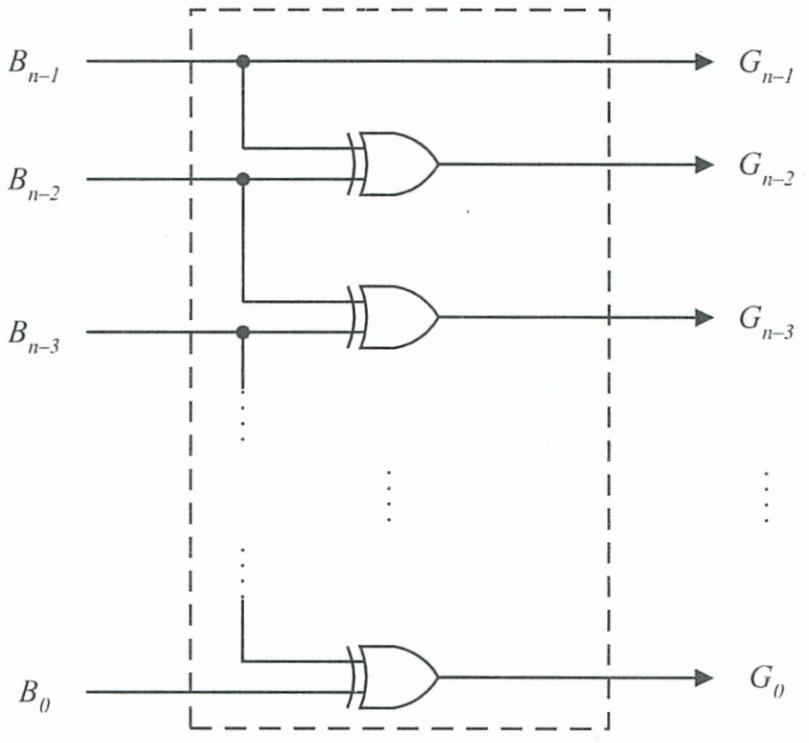
10–9. ábra *Konverziók a BIN ↔ GRAY kódok között*

rendszer-beli, a bemenetivel *azonos információt* kifejező kódszavakká alakítja át.

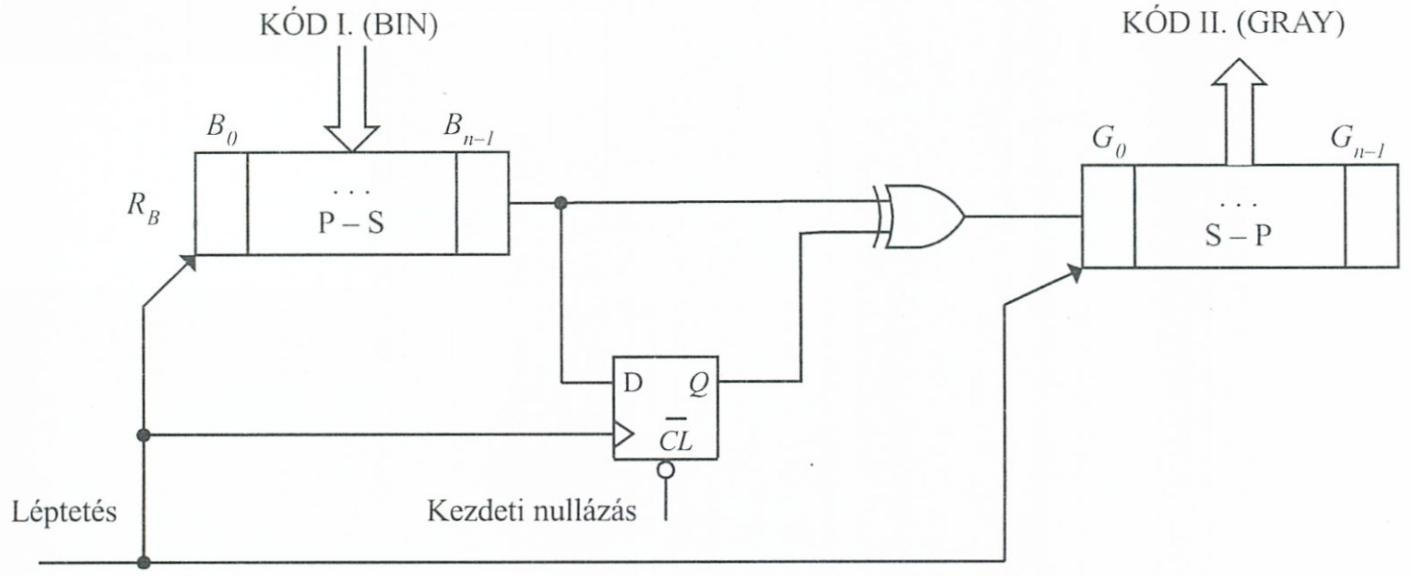
A kódátalakítás történhet párhuzamosan, például kombinációs hálózattal, vagy időben sorbaállítottan, például sorrendi hálózattal.

Párhuzamos esetre példaként a 10-10. ábrán felrajzoltunk egy BINÉR/GRAY átkódolót, mely az előzőkben bemutatott átalakítási számításokból levezethető és a 10-9. ábra táblázatában összefoglalt formulákon épül fel.

Soros esetre egy ugyancsak BINÉR/GRAY megoldást mutat a 10-11. ábra. Itt első lépésként betöltjük a BIN kódot, párhuzamosan a



10–10. ábra *Párhuzamos BIN/GRAY kódátalakító*



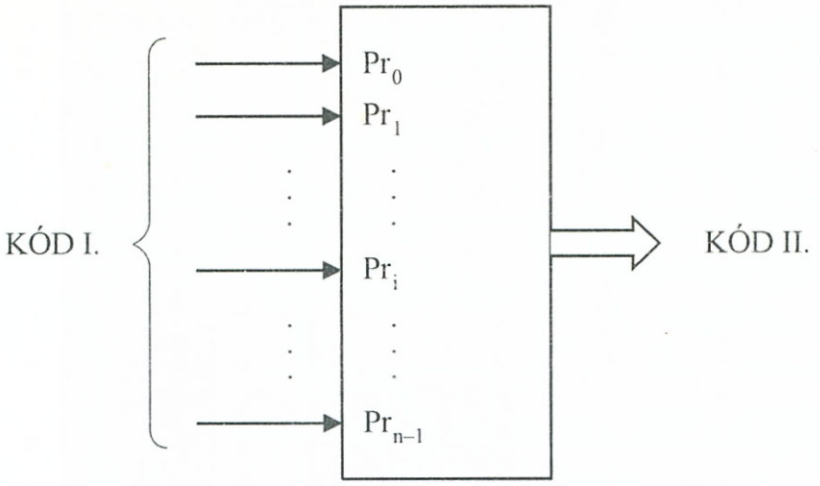
10–11. ábra *SOROS - BIN/GRAY kódátalakító*

P-S típusú regiszterbe úgy, hogy a legmagasabb bináris helyiérték lépjen majd ki legelőször. A D-típusú tárolóelem késleltető feladatot lát el a 10-9. ábra formuláiban szereplő *i* helyiérték pozíciós indexek (itt időbeli) eltolódásának biztosítása érdekében. Indítás előtt a D-tárolóelembe 0 információt írunk be a $\overline{CL}$ bemenet révén. Ezután megindulhat a léptetés, melynek során a P-S-ből, és egy ütemnyi eltolással a D tárolóból kilépő bit-ek a 10-9a. ábra formulái szerint az antivalencia kapun kapcsolatba lépnek. A kiszámított bit-ek lépésenként tárolódnak az S-P regiszterben, ahonnan az összes bit átléptetése után az eredmény párhuzamosan kiolvasható lesz.

Priorításos kódátalakítók

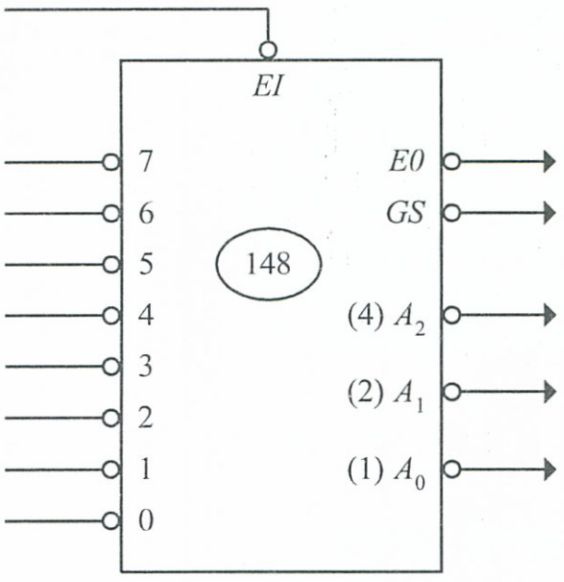
A priorításos kódátalakítók jellemzője, hogy a bemeneti KÓD I. kódszavainak bit-jei egymáshoz viszonyítottan előre megadott, megkülönböztető priorítással rendelkeznek, és a KÓD I. kódszó legmagasabb priorítású bit-je határozza meg a kimeneti KÓD II. rendszer-beli kódszót (10-12. ábra).

Szemléltetésül vizsgáljuk meg a 10-13a. ábrán látható 148 típusú 8/3-as *Priorításos kódoló* működését, melynél a bevezetőben megadott definíció úgy érvényesül, hogy itt a *decimálisan magasabb* értékű bemenet rendelkezik magasabb priorítással és a kimeneten ennek BINÉR kódja kell, hogy megjelenjen. Az ilyen alapon kitöltött működési táblázatot a 10-13b. ábrán rajzoltuk fel, aktív 0 (L) jellemzőjű be-, és kimenetek feltételezésével. Az EI bemenet-engedélyező hatású, E0 azt jelzi, hogy engedélyezés van, de a 0, 1, ... 7 bemenetek



Pr_i – prioritási tényező (pl. $Pr_i > Pr_j$)

10–12. ábra *Prioritásos kódátalakító elvi vázlata*



a)

BE									KI				
0	1	2	3	4	5	6	7	*EI*	A_2	A_1	A_0	*E0*	*GS*
X	X	X	X	X	X	X	X	H	H	H	H	H	H
H	H	H	H	H	H	H	H	L	H	H	H	L	H
X	X	X	X	X	X	X	L	L	L	L	L	H	L
X	X	X	X	X	X	L	H	L	L	L	H	H	L
X	X	X	X	X	L	H	H	L	L	H	L	H	L
X	X	X	X	L	H	H	H	L	L	H	H	H	L
X	X	X	L	H	H	H	H	L	H	L	L	H	L
X	X	L	H	H	H	H	H	L	H	L	H	H	L
X	L	H	H	H	H	H	H	L	H	H	L	H	L
L	H	H	H	H	H	H	H	L	H	H	H	H	L

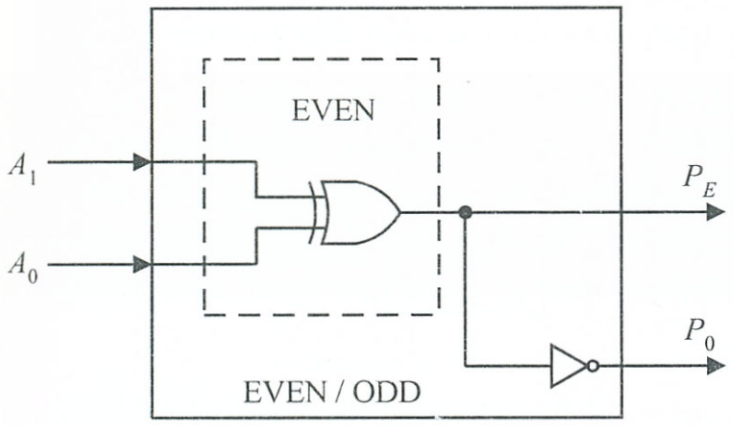
b)

10–13. ábra *8/3-as prioritásos kódoló működése*

passzívak. GS az engedélyezés esetén megjelenő aktív bemenet jelenlétét jelzi.

Hibafelismerés paritásképzéssel

A paritásképzés elvével a 10-6. ábra kapcsán már foglalkoztunk. A 10-14a. ábrán 2 bit-es Páros (EVEN), illetve Páratlan (ODD) típusú paritásképzőt tanulmányozhatunk, melynek működését a 10-14b. táblázat definiálja. A táblázatból felírhatók a kimeneti függvények:



A_1	A_0	P_E	P_0
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

a) b)

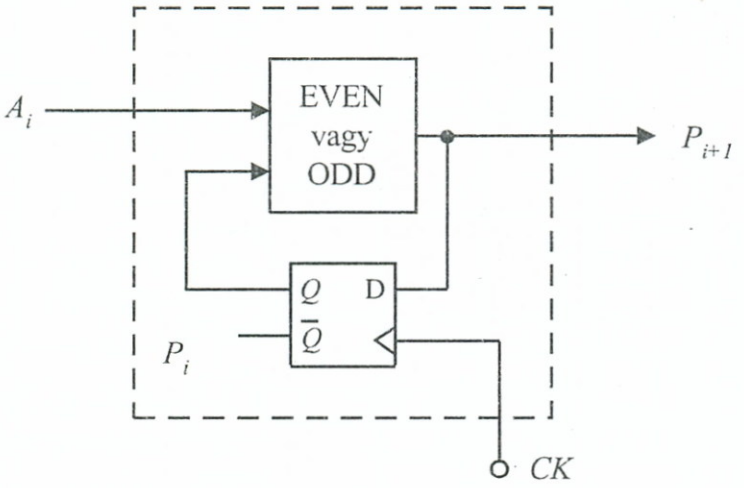
10–14. ábra *Két bit-es paritásképző működési elve*

$$\left.\begin{array}{ll} \text{Páros:} & P_{EVEN} = A_0 \oplus A_1 \\ \text{Páratlan:} & P_{ODD} = A_0 \equiv A_1 = \overline{P}_{EVEN} \end{array}\right\} \qquad (10.13)$$

A több bit-es paritásképzés *soros* és *párhuzamos* szervezésű lehet.

Soros paritásképzés elvi vázlatát mutatja a 10-15. ábra, ahol az időben korábbi i–1 bit és az érkező i bit kapcsolatát visszacsatolással állítjuk elő.

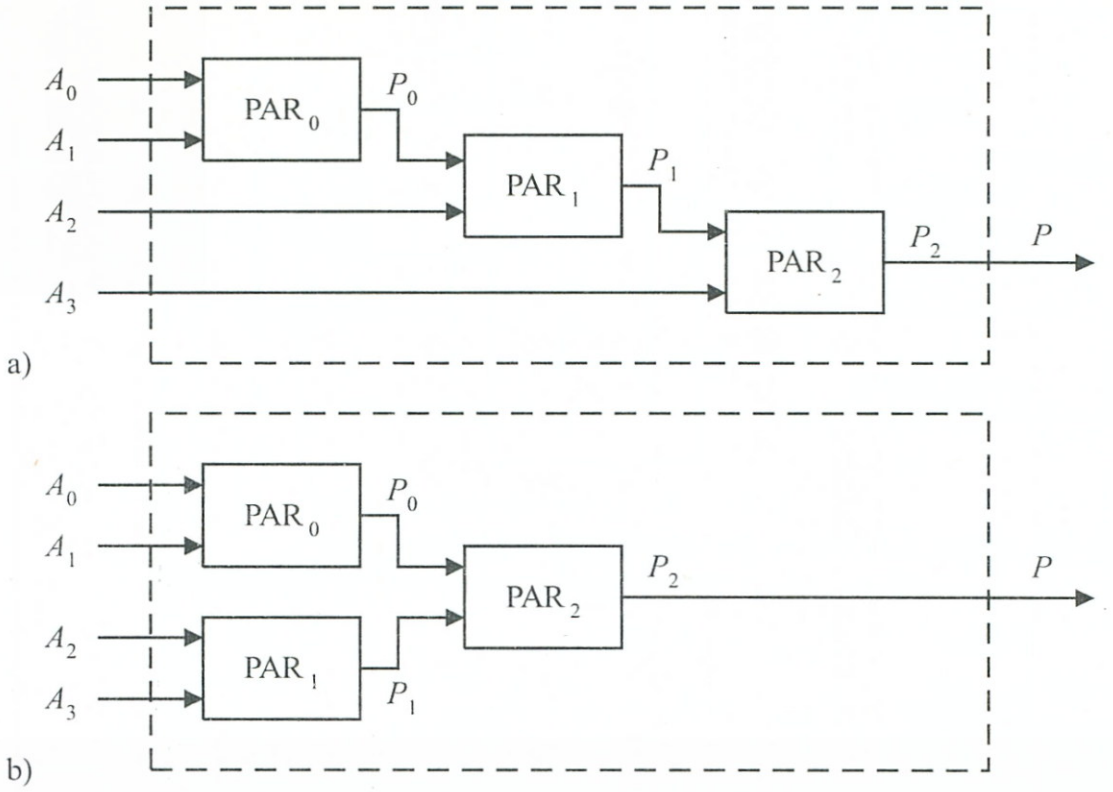
Párhuzamos paritásképzésnél kétféle felépítési struktúra alakítható ki.



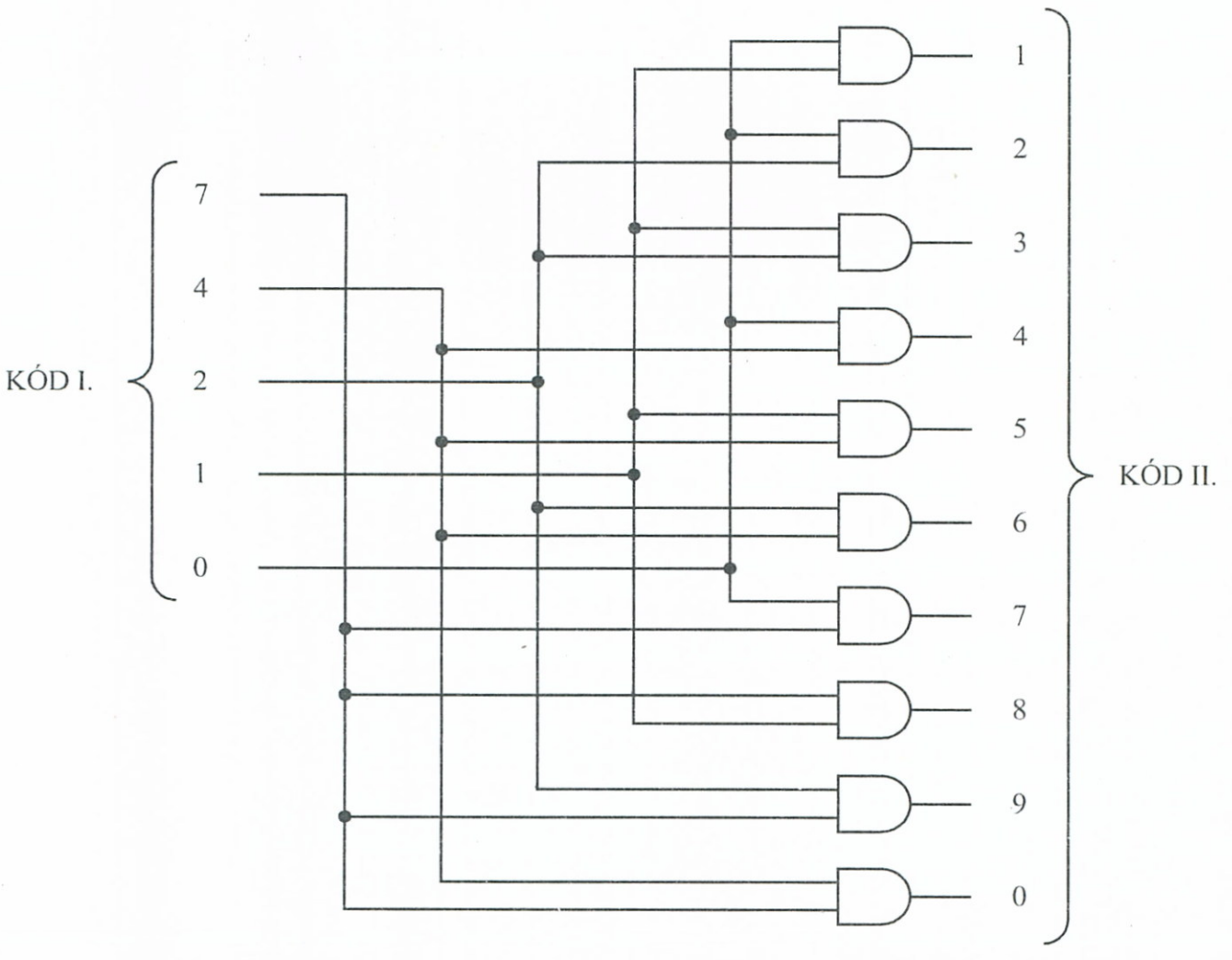
10–15. ábra *Soros paritásképző cella felépítése*

a) *Kaszkád* struktúrára utal a 10-16a. ábra, mely két-két bit láncolt szukcesszív összehasonlításán épül fel,

b) *Fa* struktúrát láthatunk a 10-16b. ábrán, ahol egyidejűleg több bit-csoport fa-elrendezésű összehasonlítása történik.



10–16. ábra *Kaszkád ill. Fa struktúrájú párhuzamos paritásképzés*



10–17. ábra *Párhuzamos (5-ből 2) / DECIMÁLIS átkódoló*

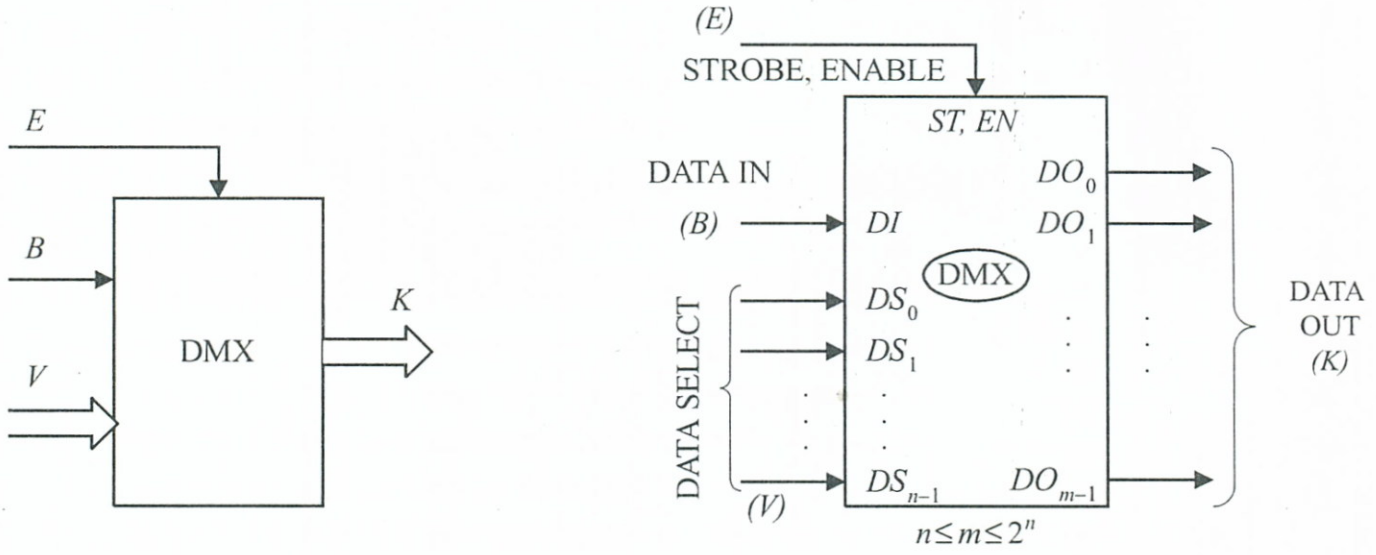
Hibafelismerés a kódrendszer célszerű megválasztásával

A 10-17. ábrán egy (5-ből 2)/DECIMÁL párhuzamos elven működő kódátalakítót rajzoltunk fel. A határozatlan term-ek száma: $2^5 - 10 = 22$ miatt a hálózat meglehetősen leegyszerűsödött. Ha a bemenetre érkező kód-szónál az 5-ből 2 szabályosság felborul, ez abban is jelentkezik, hogy egyidejűleg több kimenet is aktívvá válik.

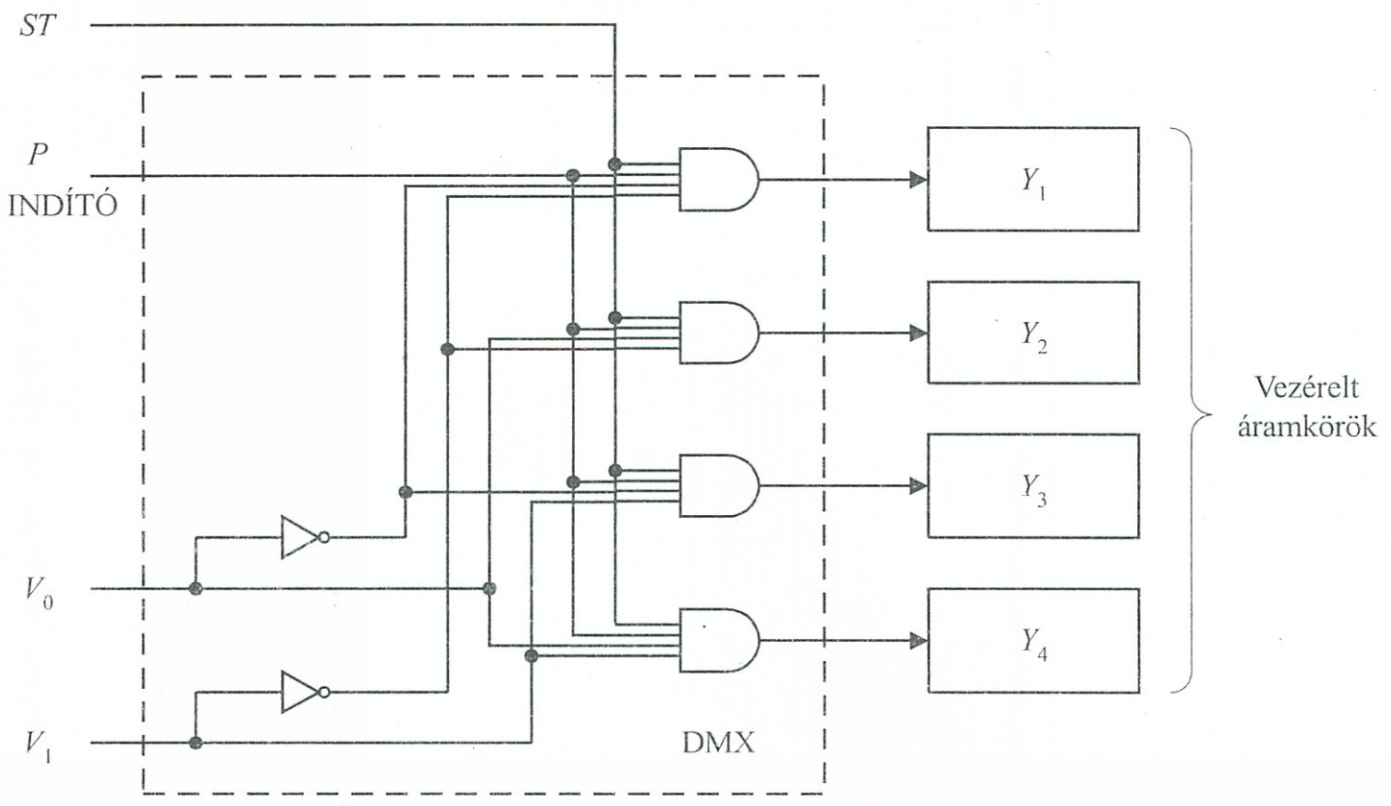
10.3. Demultiplexerek

Egy demultiplexer működését a 10-18a. ábrából kiindulva tanulmányozhatjuk. Funkciója szerint, a *B* bemenetre érkező jeleket kell átirányítani a *K* kimenetcsoport valamelyik K_i kimenetére, attól függően, hogy a *V* kiválasztó-vezérlő bemenetek ezek közül melyiket jelölik ki. A *K* kimenetek aktivizálását egy *E* engedélyező bemenet teszi lehetővé. A 10-18b. kissé részletezettebb ábrán feltüntettük a katalógusokban található gyakoribb jelöléseket és elnevezéseket, valamint a választó bemenetek és a kimenetek száma közt fennálló összefüggést is, ahol n a választó bemenetek, m a kimenetek száma.

A DMX többféle feladat elvégzésére is felhasználható:



10–18. ábra *DEMULTIPLEXER elvi vázlata*



10–19. ábra *Vezérlendő ÁK $_i$ kiválasztása*

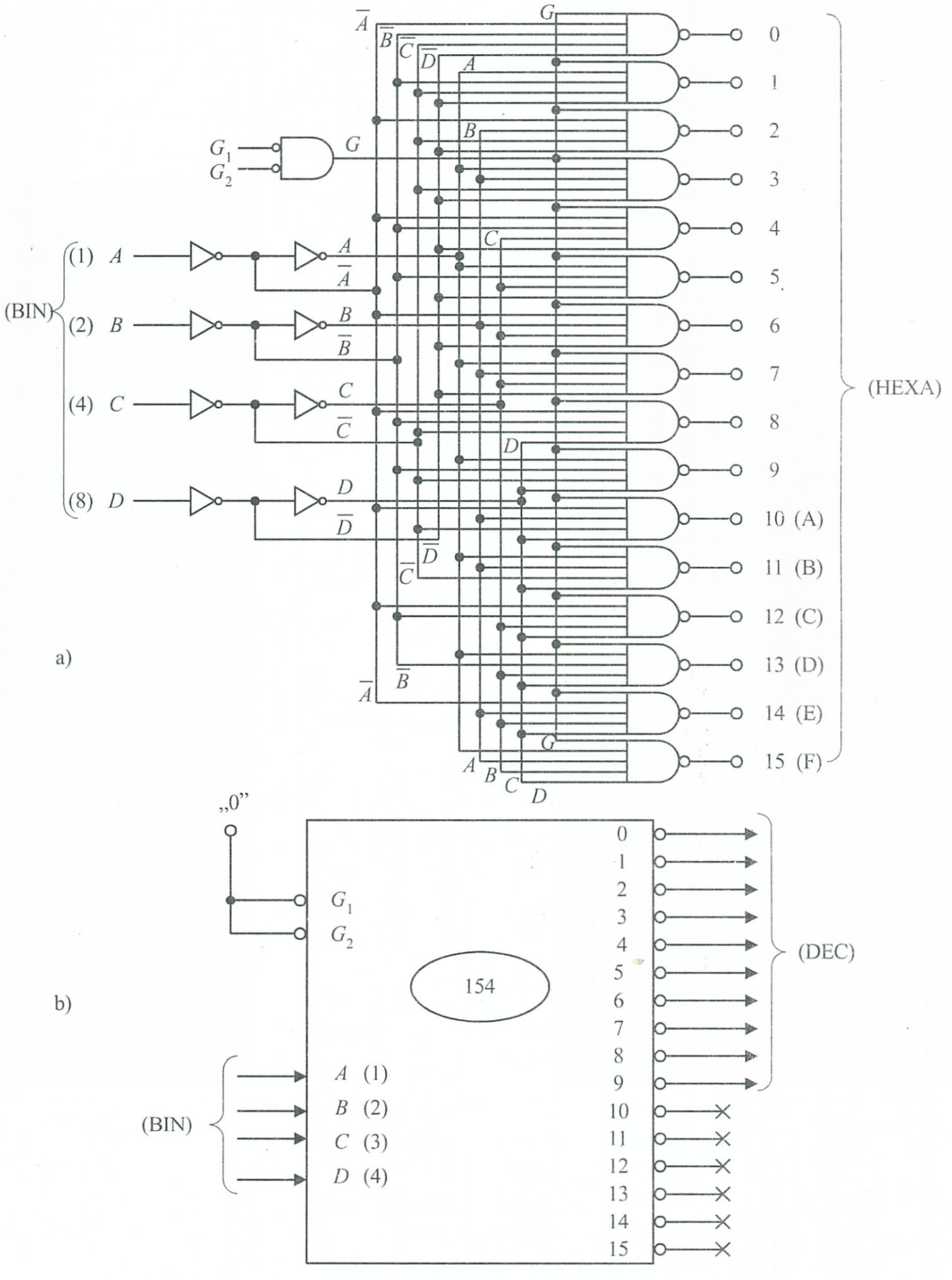
10.3.1. Adatelosztás

Egy adatelosztási feladatmegoldást szemléltet a 10-19. ábra példája, melynél egy *P* indító jellel mindig a DMX által (V_1, V_0 kombinációival) kiválasztott aktuális Y_i áramkört tudjuk elindítani, ha erre az ST engedélyező jel lehetőséget ad.

10.3.2. Kódátalakítás

A DMX-ek felhasználhatók kódátalakítási feladatokra is. Ilyenkor a bemeneti (átalakítandó) KÓD I. kódszó bit-jei kerülnek a *V* választóbemenetekre, a kimeneti KÓD II. pedig a *K* kimeneteken jelenik meg. A *B,* ill. ST bemeneteket vagy olyan konstans logikai szintre kötjük, mely a (*V*) — (*K*) kapcsolatú művelet zavartalanságát biztosítja, vagy kapuzási-engedélyezési feladatok ellátására használjuk fel őket.

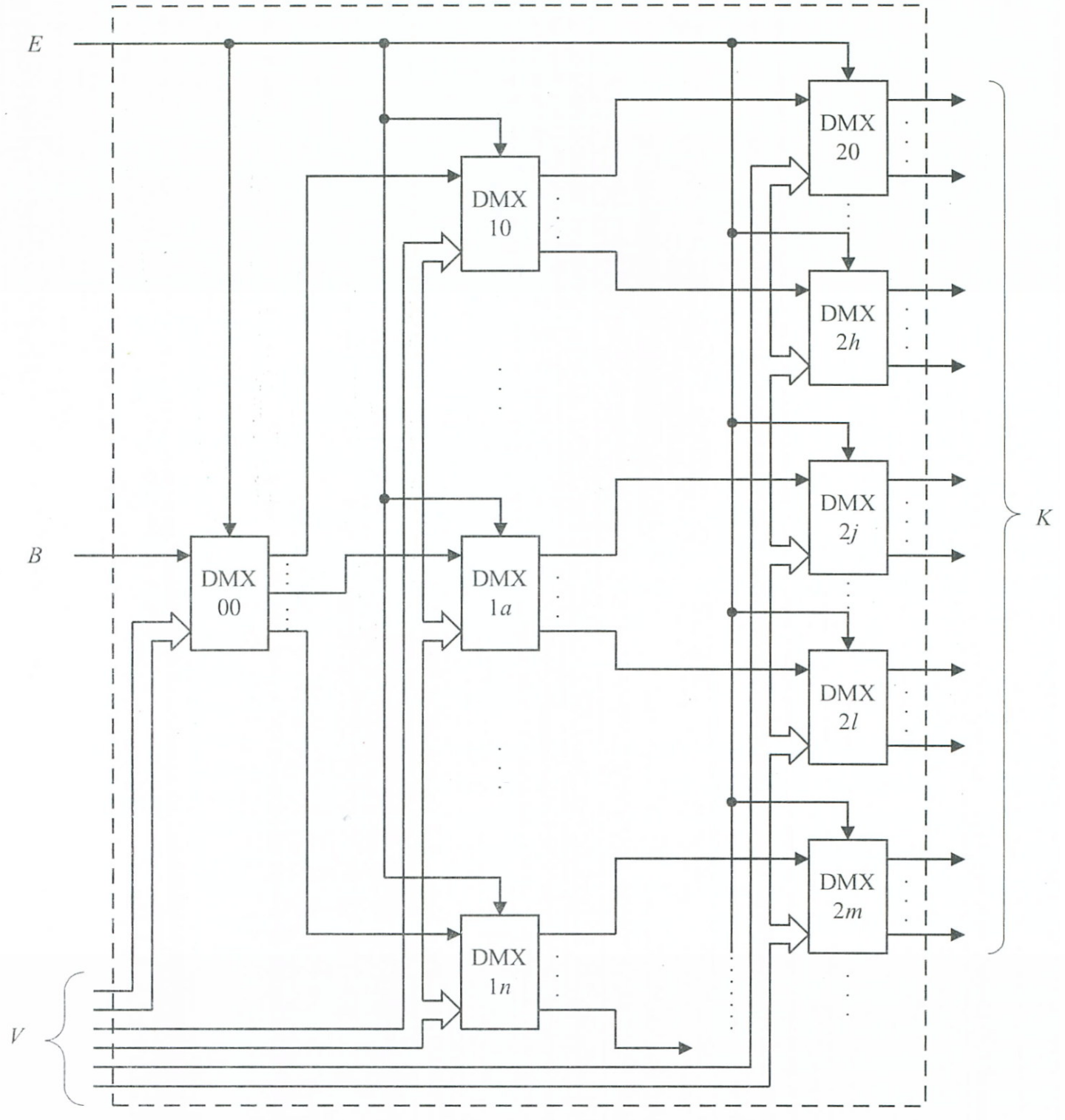
A 10-20a. ábrán egy DMX-el megoldott BINÉR/HEXA kódátalakítót rajzoltunk fel. A DCBA (8, 4, 2, 1) súlyozású bemenetekre érkező négyelemes BINÉR kódszó átszámított értékének megfelelő HEXA kimeneten jelenik meg egy *aktív 0* szint, míg a többi kimenetek *passzív 1* szinten maradnak. (Pl.: BE: 1101 esetén KI: *D*) Az ilyen kó-


a)
(BIN)
(1) A
(2) B
(4) C
(8) D
G_1
G_2
G
(HEXA)
0
1
2
3
4
5
6
7
8
9
10 (A)
11 (B)
12 (C)
13 (D)
14 (E)
15 (F)
b)
„0”
154
(BIN)
A (1)
B (2)
C (3)
D (4)
(DEC)

10–20. ábra

dolást „n-ből egy" típusnak nevezik. A G_1G_2 vezérlőbemenetekkel valamennyi kimenet letiltható.

Ha ugyanezt az áramkört úgy üzemeltetjük, hogy a bemenetekre csak BCD kódszavakat adunk (10-20b. ábra), akkor egy BINÉR/DECIMÁL átkódolóhoz jutunk, ahol a 10 ... 15 kimeneteket nem használjuk fel.



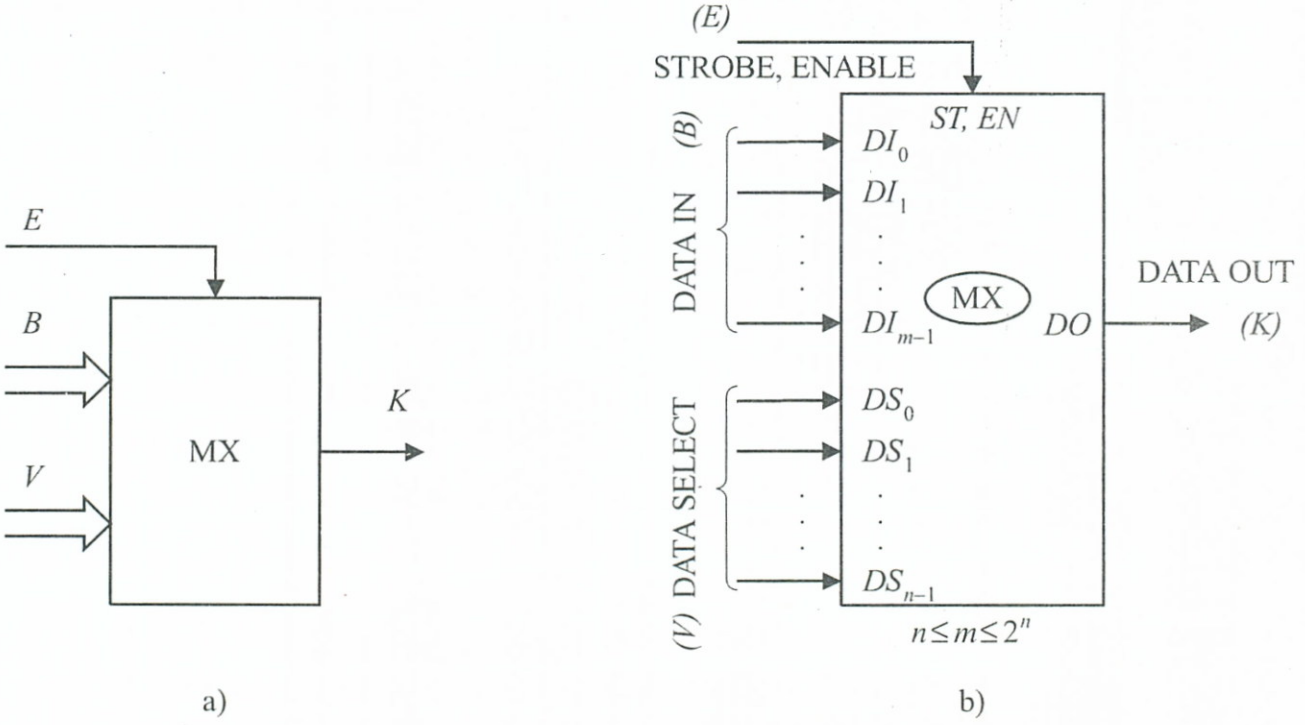
10–21. ábra *DMX bővítés elvi vázlata*

10.3.3. Demultiplexerek összekapcsolása

A demultiplexereket gyakran a „Bemenet-Választóbemenet-Kimenet" (B/V/K) számcsoporttal jellemezzük. Például a 10-19. ábrabeli DMX jellemzője: 1/2/4. A kereskedelemben forgalmazott DMX-eknél egy-egy IC áramköri tokokban található egységek kimeneteinek száma gyakran kisebb, mint amennyi az adott műszaki feladathoz szükséges lenne. Ilyen esetekben – ha nem kívánunk az adott feladathoz egy speciális integrált áramkört (BOÁK) tervezni – a feladatot több DMX összekapcsolásával, ún. *bővítésével* oldjuk meg. A bővítés elvét a 10-21. ábra szemlélteti.

10.4. Multiplexerek

Egy multiplexer működését a 10-22a. ábrából kiindulva követhetjük. Funkciója szerint a *B* bemenetcsoportra érkező jelek valamelyikét kell átirányítania a K kimenetre, attól függően, hogy a *V* kiválasztó-vezérlő bemenetek melyik B_i bemenetet jelölik ki. Hasonlóan a DMX-ekhez, itt is szerepel egy *E* engedélyező bemenet, továbbá itt is felrajzoltunk egy részletesebb ábrát (10-22b. ábra), mely az elterjedtebb jelöléseket és összefüggéseket szemlélteti.



10–22. ábra *MULTIPLEXER elvi vázlata*

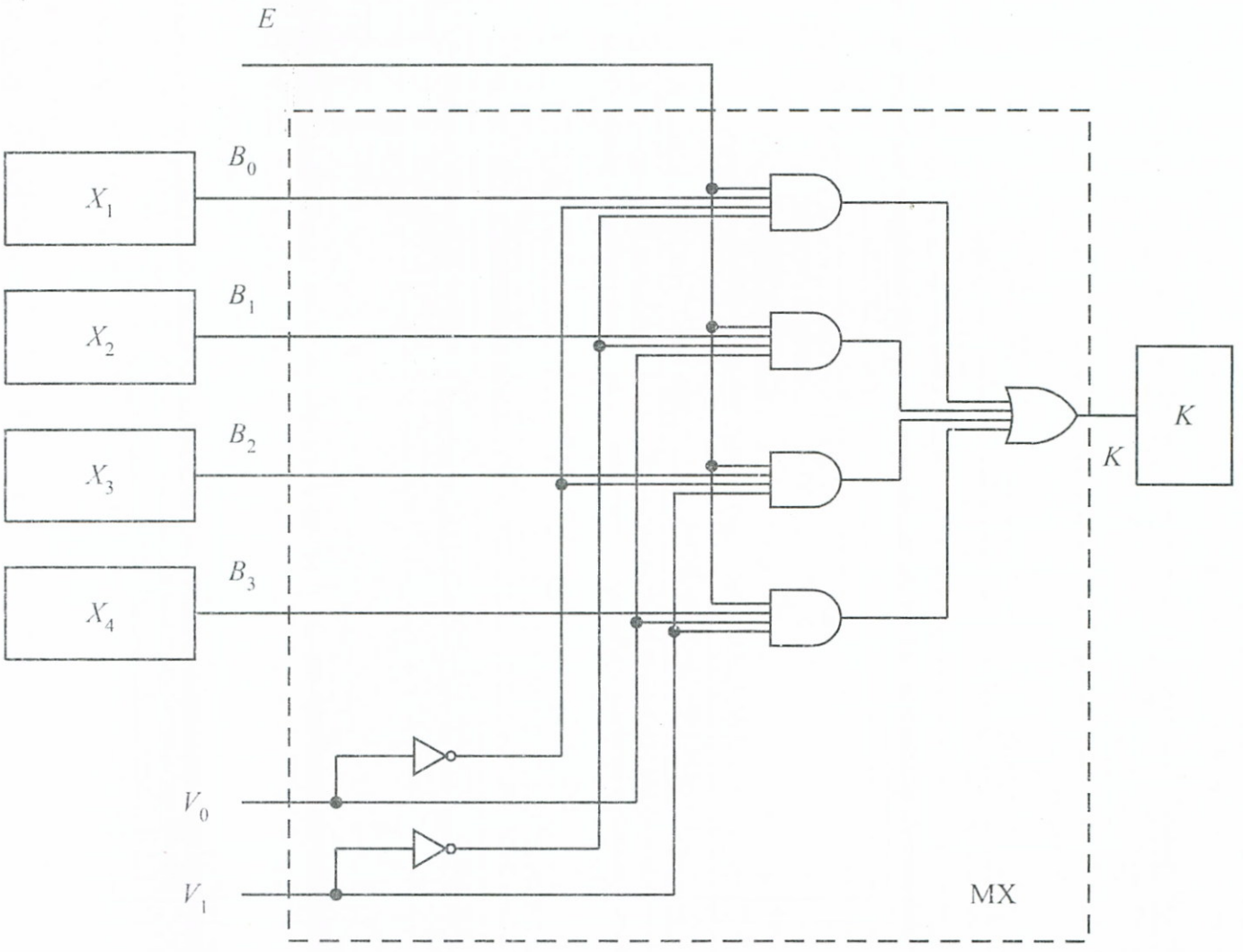
10.4.1. Adatkiválasztás

Az MX-ek egyik fontos alkalmazási területe a különböző helyekről érkező adat-, ill. parancs-funkciójú jelek közül a kívántnak kiválasztása és a kimenet felé továbbítása.

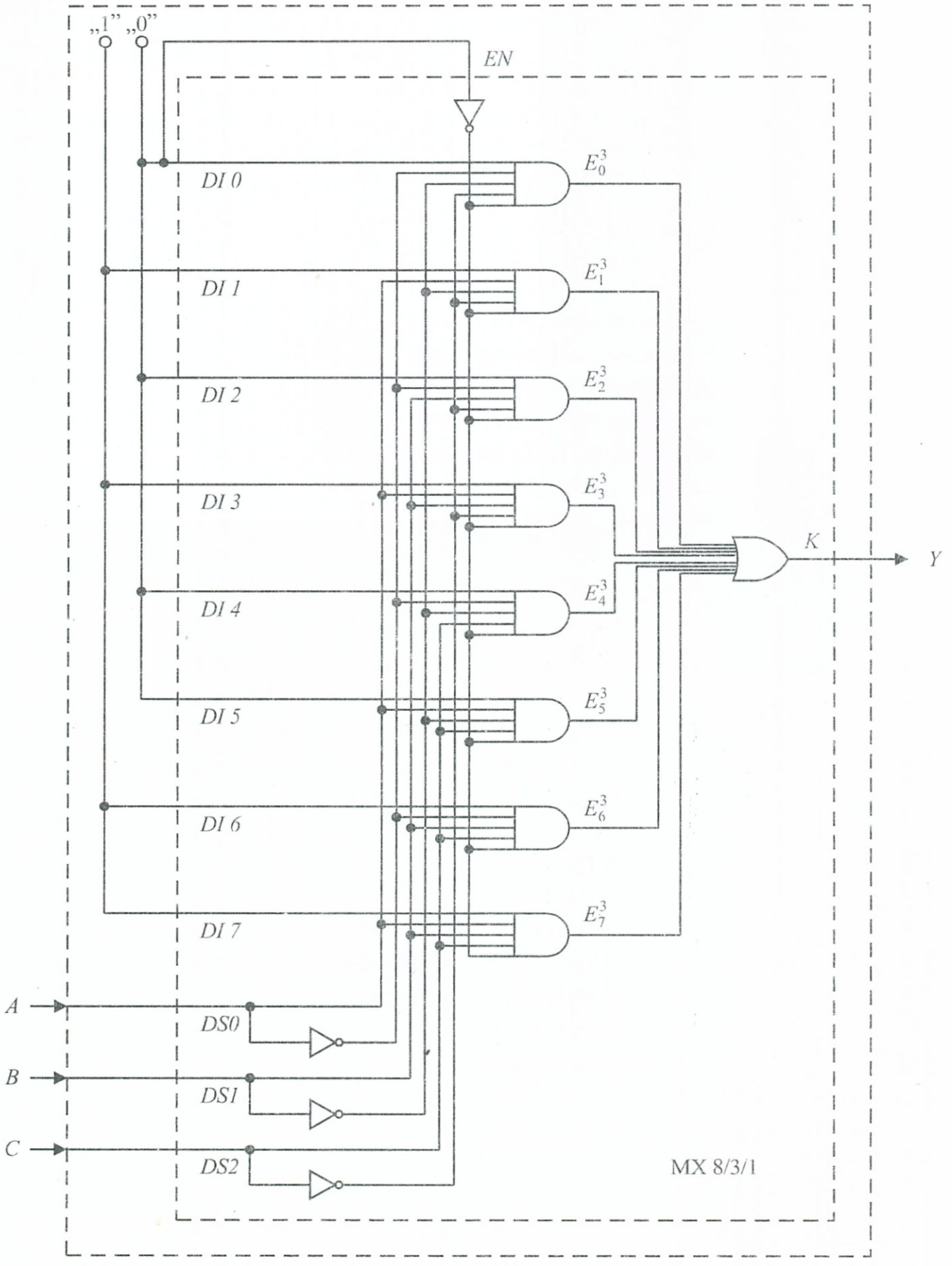
Egy adatkiválasztási feladat megoldására vonatkozó példát láthatunk a 10-23. ábrán, ahol az X_i mérőáramkörökről érkező mérési adatokat a V_0, V_1 választó jelek különböző kódjainak segítségével „kapcsoljuk rá" a K kimeneten elhelyezkedő egyetlen kiértékelő berendezésre, alkalmanként. Az MX-eket is – hasonlóan a DMX-ekhez – jellemezhetjük a (B/V/K) számcsoporttal. Ez példánknál: 4/2/1

10.4.2. Logikai feladatmegoldás

Az MX áramkörök segítségével tetszés szerinti logikai függvényt realizálhatunk, amennyiben az MX olyan felépítésű, hogy a DS_0,



10–23. ábra *Különféle mérési helyek rákapcsolása a K kiértékelő berendezésre*



10–24. ábra *Az* $Y = \sum^{3}(1, 3, 6, 7)$ *függvény realizálása MX 8/3/1-gyel*

$DS_1 \ldots DS_{n-1}$ választó bemenetei az áramkör belsejében az összes lehetséges E_i^n ÉS-term-et megvalósítják, és fennáll az: $m = 2^{n-1}$ összefüggés.

Ez esetben ugyanis felírható a 10-22b. ábra jelöléseit is felhasználva a következő kifejezés, mely valójában a K kimeneten értelmezett valamely $Y = F^n$ logikai függvény diszjunktív szabályos alakja:

$$K = Y = F^n = DI_0 \cdot E_0^n + DI_1 \cdot E_1^n + \ldots + DI_{m-1} \cdot E_{2^{n-1}-1}^n \qquad (10.14)$$

A (10.14) kifejezést felhasználva, ezután egy adott $Y = F^n(X_1 \ldots X_n)$ függvényt úgy realizálhatunk valamely MX-szel, hogy a $DS_i(V)$ csatlakozásokat tekintjük a bemeneti változóknak, a K kimenetet Y-nak, míg az egyes DI_i (B) bemeneteket pedig olyan 0–1 konstans logikai értékekre kötjük, melyeket a realizálandó Y függvény igazságtáblázatának megfelelő sorai (ill. a diszjunktív alak α_i tényezői) az Y-ra előírnak. Az E bemenetet rendszerint „szabad" állást biztosító értékre kötjük le.

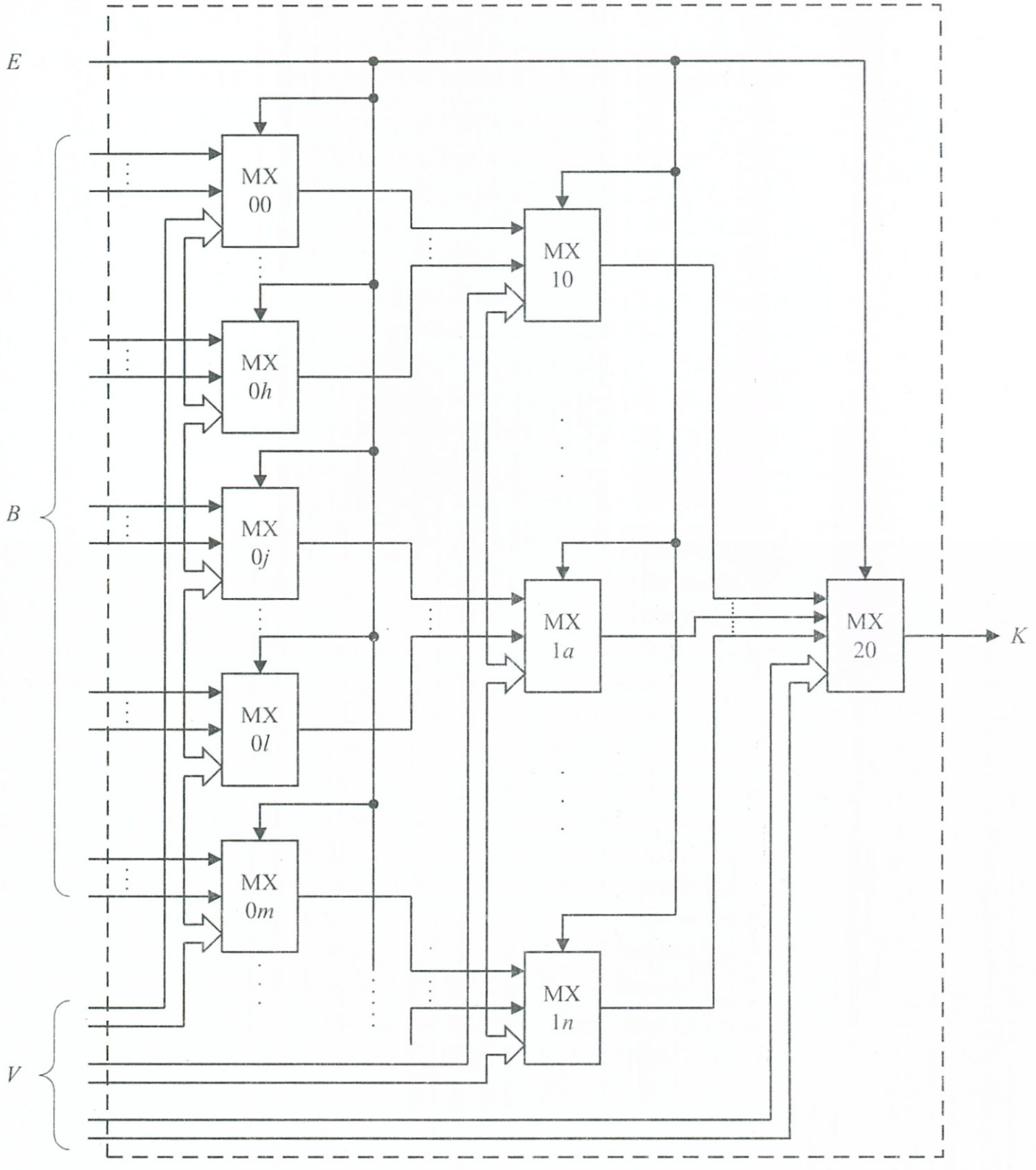
Példaként realizáljuk a következő függvényt MX segítségével:

$$Y = F(C, B, A) = \sum^{3}(1,3,6,7) \qquad (10.15)$$

A realizálásnál 8/3/1 MX-et választottuk, mivel egy háromváltozós függvény teljes diszjunktív alakjában 8 tag szerepel, ezért ez az MX típus látszik a realizációra legmegfelelőbbnek. A (10.14) összefüggés értelmében, a 10-24. ábrában a példa kiinduló alakjában szereplő term-ek $\alpha_1 = \alpha_3 = \alpha_6 = \alpha_7$ tényezőit az 1, a többieket: $\alpha_0 = \alpha_2 = \alpha_4 = \alpha_5$-öt pedig a 0 logikai értékeknek megfelelő tápfeszültség-pontokra kötöttük, ugyanakkor a C, B, A független változókat a DS2, DS1, DS0 bemenetekre vezetjük. Mivel az EN engedélyező bemenet aktív 0-ára kapcsol „szabadra", ezért ezt is a 0 konstans értékre csatlakoztatjuk. Az ily módon „felprogramozott" MX biztosítja a (10.15) formulával megadott kiinduló függvény áramköri realizálását.

10.4.3. Multiplexerek összekapcsolása

Hasonlóan a DMX-eknél elmondottakhoz, itt is szükségessé válhat a bővítés, melynek elvi vázlatát MX esetre a 10-25. ábra mutatja.



10–25. ábra *MX bővítés elvi vázlata*

10.5. Számlálók

A számlálók sorrendi áramkörökből felépülő funkcionális egységek, melyek a bennük tárolt információ értékét a bejövő jel hatására 1-gyel növelik vagy csökkentik. A definíciónak megfelelő két alapváltozatot a 10-26. ábra elvi vázlatán láthatjuk.

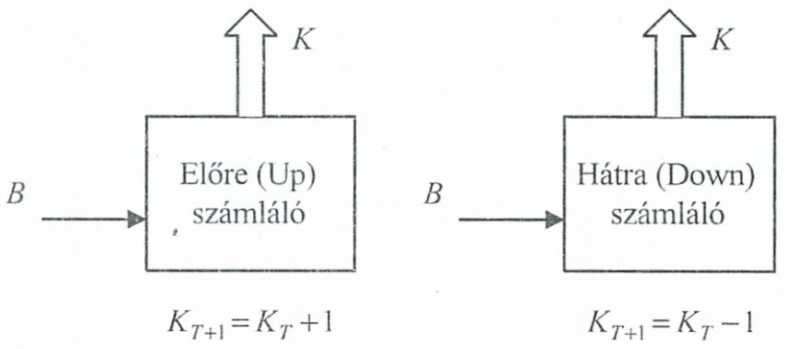
A számlálók – mint sorrendi hálózatok – a számlált információt a mindenkori belső állapotukkal fejezik ki, melyet kódolt formában jelenítenek meg a kimeneten. Emiatt a számlálók automata-elméleti leírásánál elsősorban a Moore-modell kerül előtérbe.

A számlálók „hasznos" belső állapotainak számát, ami egyben azt a számot jelenti, mely a számláló által megszámlálható maximális érték: *moduló*-nak (M) nevezzük, és a számláló fontos jellemzőjének tekintjük.

A számlálókat többféle szempont szerint osztályozhatjuk:

- a számlálót felépítő tárolóelemek kapcsolatai alapján:
  - soros
  - párhuzamos
- a számlálót felépítő sorrendi hálózat jellege alapján:
  - szinkron
  - aszinkron
- a számlálás iránya alapján:
  - előre (inkrementáló)
  - hátra (dekrementáló)
  - kétirányú (reverzíbilis)
- a számlálandó információt kifejező kódrendszer alapján:
  - Binér
  - BCD
  - JOHNSON

  stb.



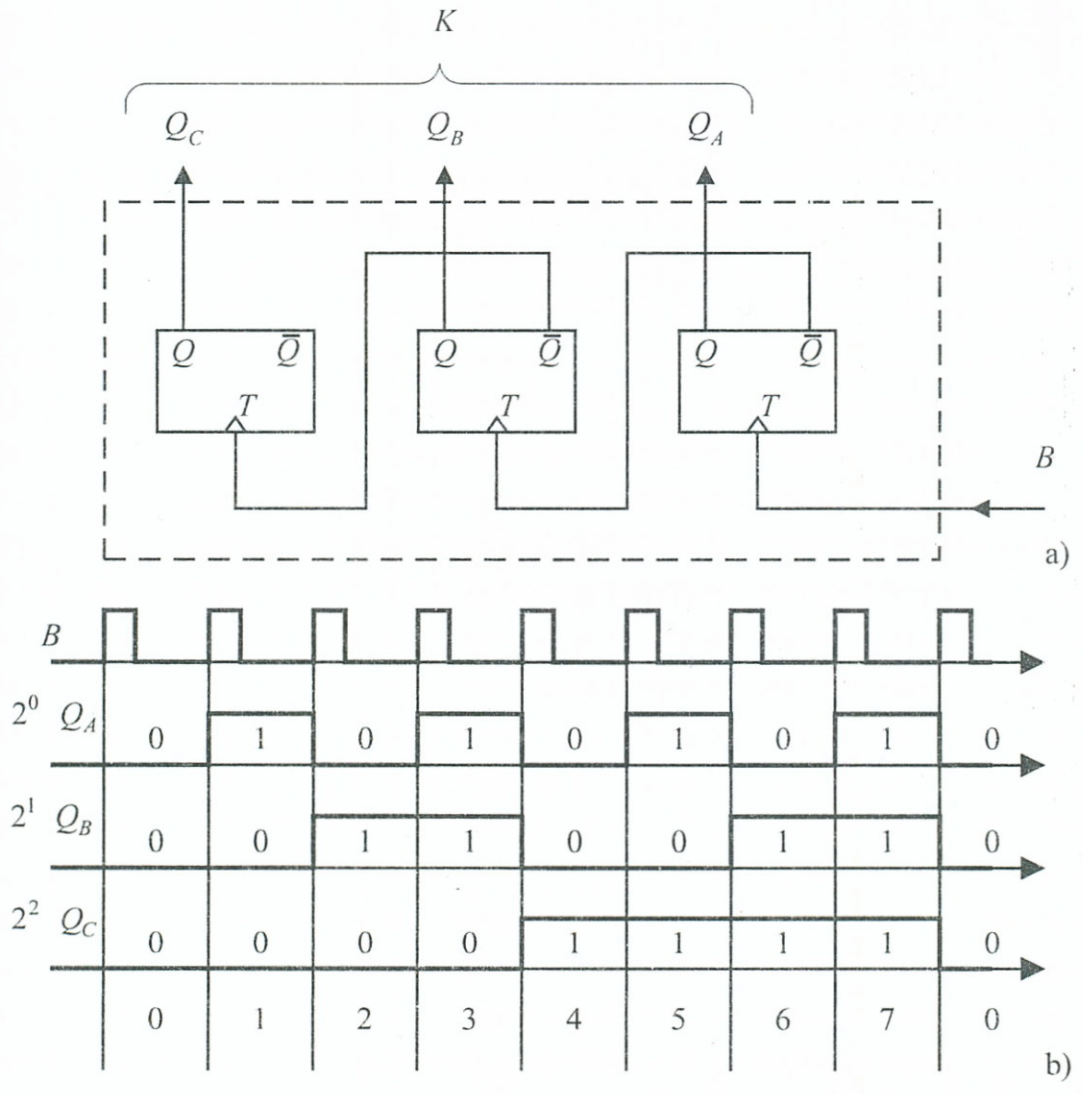
10–26. ábra *Előre- és hátra számlálók elvi vázlata*

típusú számlálókat különböztetünk meg, melyek egy adott számláló elnevezésében is megjelennek.

10.5.1. Aszinkron előreszámlálók

A 10-27a. ábrán egy számlálásra alkalmas sorrendi hálózat látható. Vizsgáljuk meg ennek működését, állapítsuk meg az „*M*" modulót és adjuk meg jellemző megnevezését is.

A hálózatot – mint látható – felfutó élre működő *T* tárolók soros kapcsolata alkotja. Ha lépésről lépésre felrajzoljuk a 10-27b. ábrán látható idődiagramot, mellyel figyelembevesszük, hogy a tárolók egymásközti kapcsolata a $\overline{Q} \rightarrow T$ összeköttetéseken zajlik le, akkor megállapíthatjuk, hogy a 10-27b. ábra: 0, 1,... 7 ütemsorrendjében az egyes tárolók kimeneti értékeiből alkotott kódszavak binér rendszert



10–27. ábra *M = 8-as –aszinkron–soros–binér–előre számlánc*

alkotnak, és az ütemeket jellemző decimális sorszámok bináris megfelelőjét adják meg. A 7. ütemet követően ismét 0-ával kezdődő ciklus következik. A példa kérdésére válaszolva, megállapíthatjuk, hogy a kapcsolás egy „moduló 8-as – aszinkron – soros – binér – előre" típusú számlánc.

Az aszinkron számlálók lényegesen hátrányos tulajdonsága, hogy a tárolóelemek állapotváltozása a terjedési időkkel eltoltan, láncolódva történik, így az értékelendő állapotváltozások közé több tranziens állapot-kód is beiktatódhat. Ez mind a működési sebesség, mind a számláló által meghajtott további áramkörök egyértelmű vezérlése szempontjából kedvezőtlen lehet.

10.5.2. Szinkron előreszámlálók

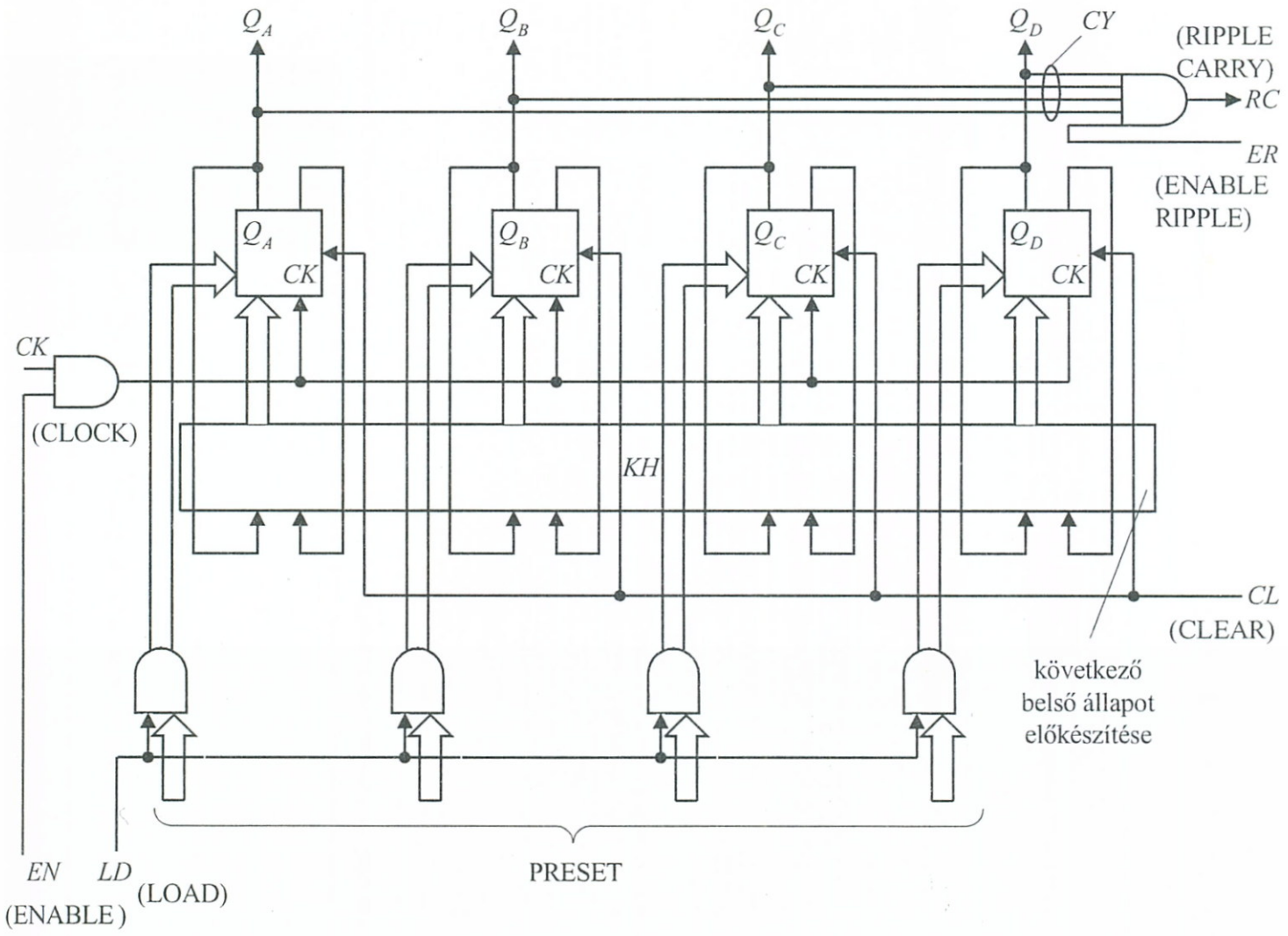
Szinkron számlálóknál az órajelek működtető hatása minden tárolóelemnél közel azonos időben érvényesül, ezért az előzőkben említett tranziens-kódok veszélye lecsökken, és így a működési sebesség is kedvezőbben alakul. A tárolóelemek vezérlését egy kombinációs hálózat végzi, melynek bemeneteit a vezérelt tárolóelemek előző időütemben felvett logikai állapotai alkotják a 10-28. ábra elvi rajzának megfelelően.

A számlálót el szokták látni többféle, sokoldalúságot biztosító további bemenetekkel is, így a 10-28. ábrán feltüntettük a számláló léptetését (CLOCK) engedélyező (ENABLE), az általános törlő (CLEAR) és a számláncot adott pozícióba bármikor beállító (PRESET) jeleket kapuzó (LOAD) bemenettel. A később tárgyalandó bővítések érdekében gyakran kivezetik a számláló legmagasabb értékű kódját kapuzottan összefoglaló átvitel (RIPPLE CARRY) és az ezt kapuzó (RIPPLE ENABLE) csatlakozásokat is.

A különféle csatlakozások aktív 0, vagy aktív 1 tulajdonsággal rendelkezhetnek. A LOAD és CLEAR csatlakozások *szinkron* (tehát az órajellel együtt hatásos), vagy *aszinkron* (órajeltől függetlenül hatásos) jellegűek is lehetnek a működésében szinkronnak tekintett számláncnál.

Egy ilyen, fent részletezett kialakítású hálózat elvi felépítését 4 bit-re a 10-28. ábra szemlélteti.

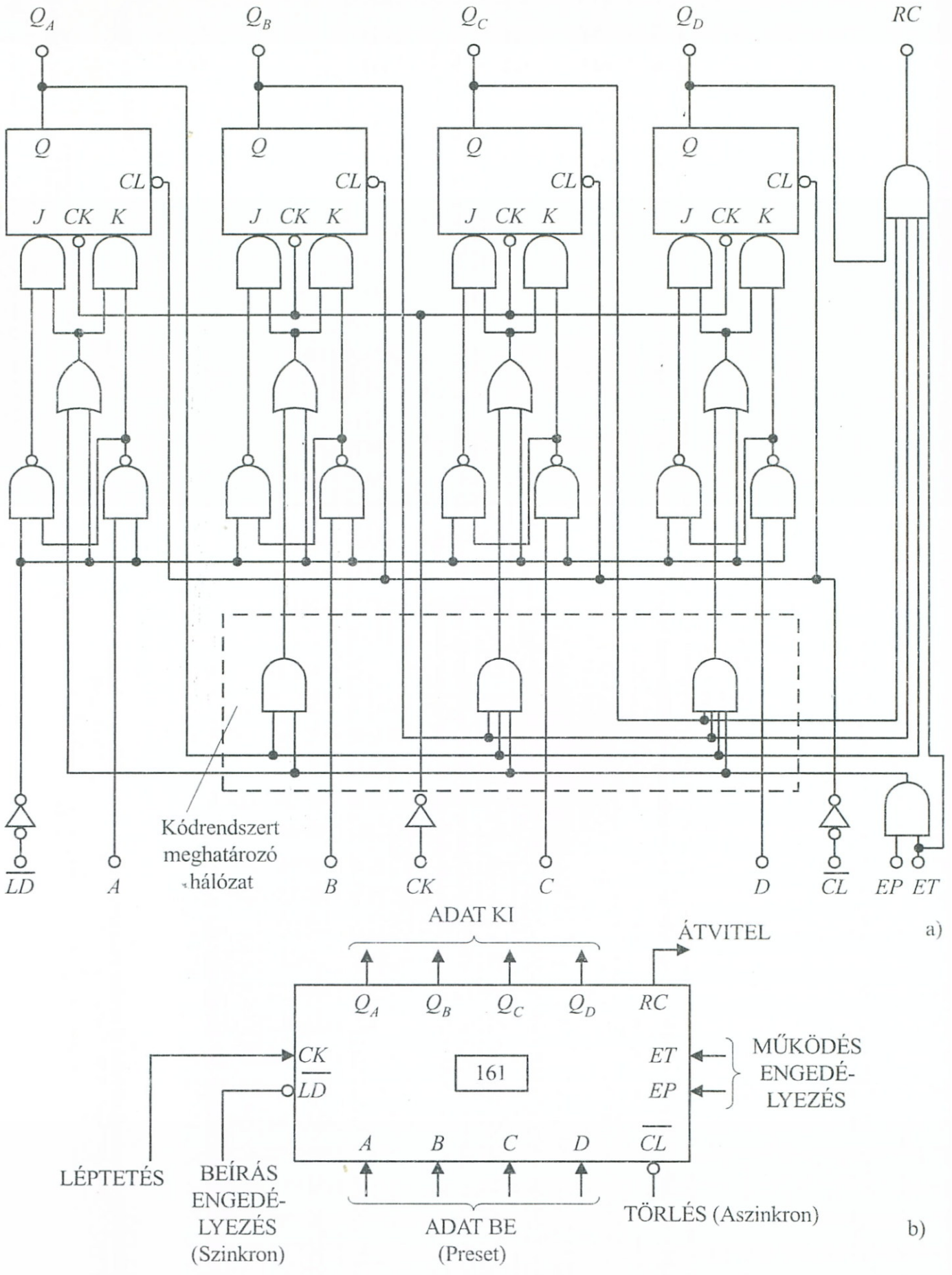
Szemléltetésül a 10-29a. ábrán egy „161" típusú M = 16 – szinkron – párhuzamos – binér előreszámláló modult rajzoltunk fel némi egyszerűsítéssel. A hálózaton belüli szaggatottan körülhatárolt rész biztosítja az egyes bináris helyértékek közötti „párhuzamos" átvitelt olyan elven, hogy a magasabb helyértéken elhelyezkedő tárolóele-



10–28. ábra *Párhuzamos szinkron számlánc elvi vázlata*

mek csak akkor tudják megváltoztatni állapotukat, ha már valamennyi alacsonyabb helyérték tárolóeleme 1 (H–High) állapotba került. A kapcsolás preszetelhető, azaz a számlálás kezdeti állapotát az A, B, C, D bemenetekre (1, 2, 4, 8 súlyozás) adott bináris számmal beállíthatjuk. Az előzetesen előkészített beírandó értéket az LD (LOAD) bemenettel aktualizálhatjuk és a tárolókba való tényleges beírás a következő órajel (CK–CLOCK) megérkezésekor következik be, azaz *szinkron* módon történik. A hálózat valamennyi tárolójának $Q_i = 0$-ba történő állítását a CL (CLEAR) bemenetre adott aktív „0" szinttel végezhetjük *aszinkron* módon. Végül a számlánc el van látva egy bővítési lehetőséget elősegítő, DCBA – 1111 állapotnál aktivizálódó RC (RIPPLE CARRY) átviteli kimenettel, melyet az ET (ENABLE) bemenettel lehet engedélyezni. Az ET és EP engedélyezések EP·ET = 0 esetén a számláló működését tudják leállítani a tárolóelemek vezérlő pontjainak inhibitálásával.

A számlánc csatlakozási vázlatát a 10-29b. ábrán adtuk meg.



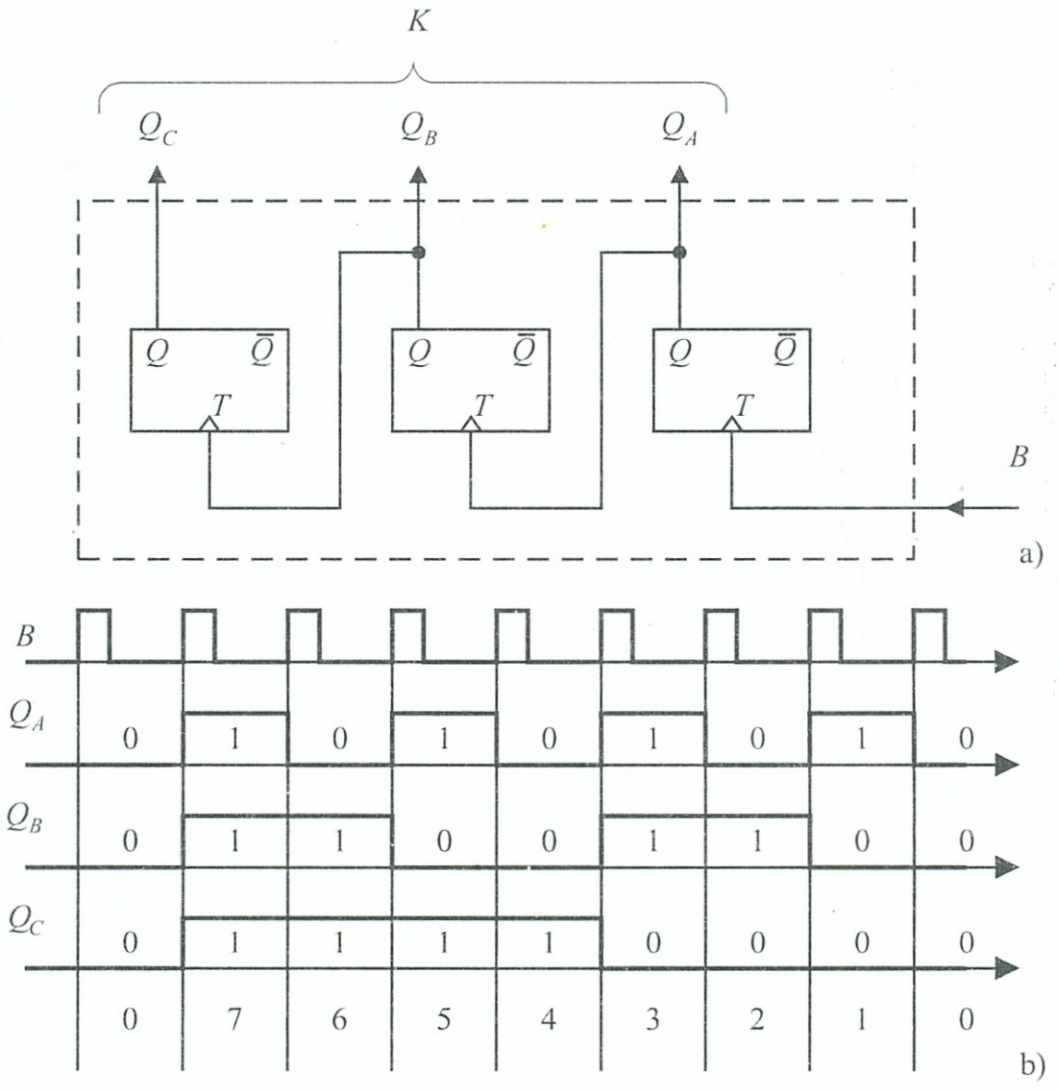
10–29. ábra *M = 16 – szinkron-párhuzamos-binér-előre számlánc*

10.5.3. „Hátra"-számlálók

A „hátra"-számlálók elvileg csak annyiban különböznek az „előre"-számlálóktól, hogy a lépésenként egymást követő, számláló állapotát kifejező kódszavak *dekrementálódva* követik egymást.

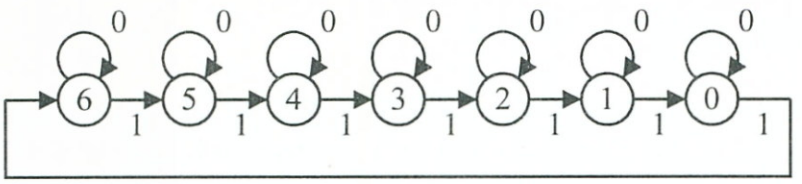
Az aszinkron *hátra-számlálók* származtatása az előre irányúakból áramkörileg gyakran rendkívül egyszerűen oldható meg. Ez a lehetőség az előre-kódok és a hátra-kódok közötti komplemens természetű kapcsolatban rejlik. Vizsgáljuk meg ennek érvényesülését a következő példán keresztül:

A 10-30a. ábrán látható aszinkron hálózat csak annyiban különbözik a 10-27a. ábrabelitől, hogy a $Q_A - Q_B - Q_C$ fokozatok közti kapcsolat az előzőhöz képest invertáltan, tehát $Q_i \rightarrow T$ összeköttetéssel van megoldva. Ennek következménye viszont a 10-30b. ábra szerinti



10–30. ábra *M = 8-as –aszinkron–soros–binér–hátra számlánc*

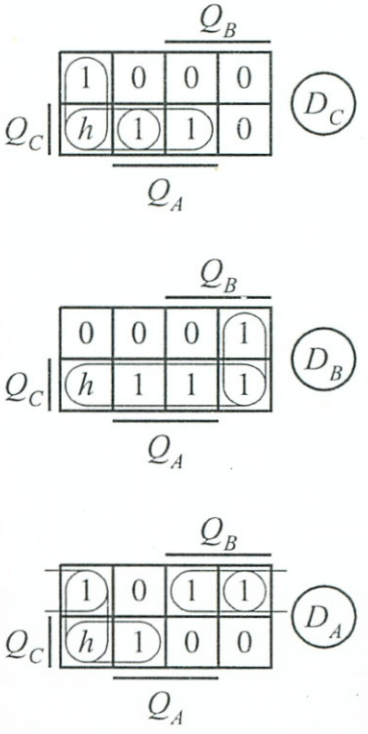
a)



b)

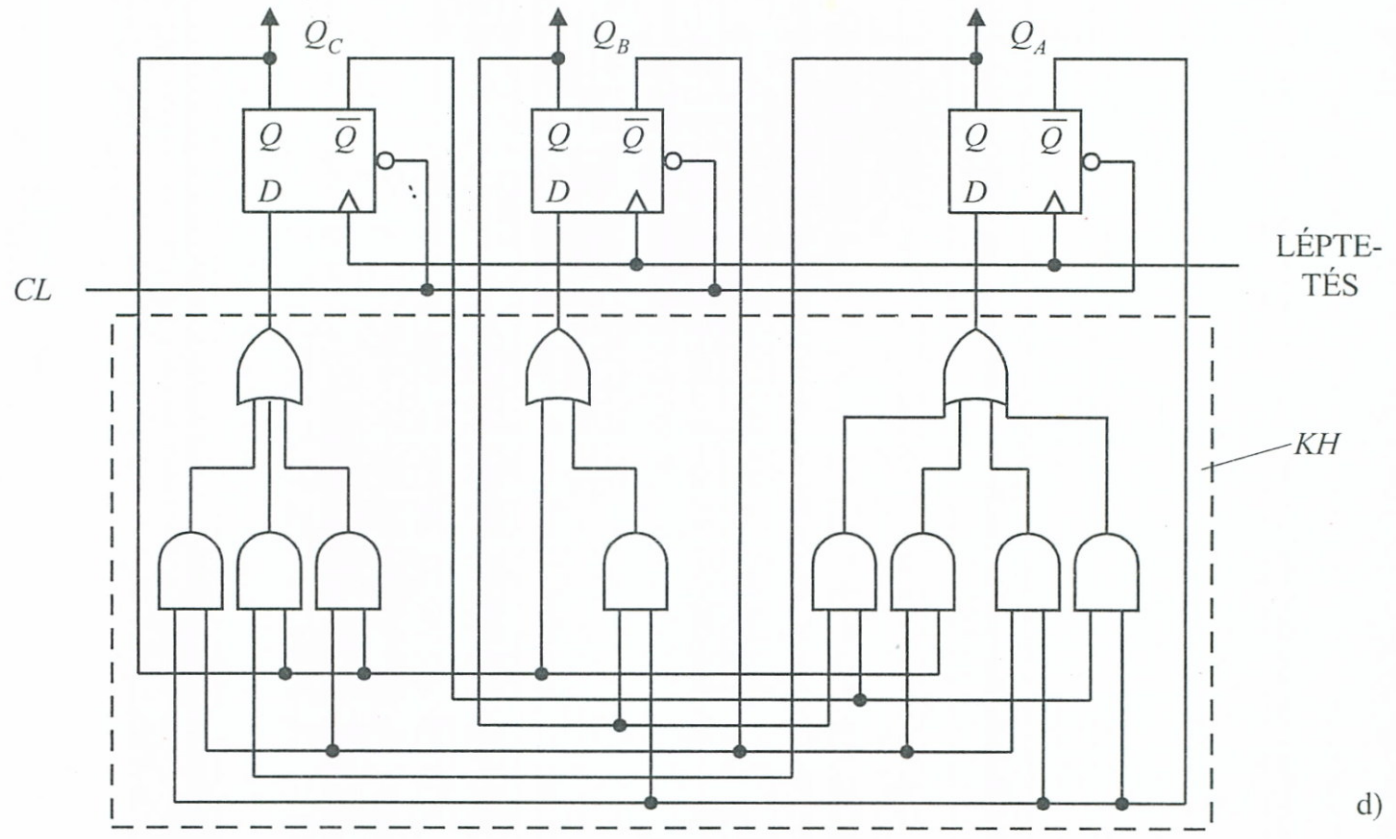
	KORÁBBI			ÚJABB			VEZÉRLÉS		
	T			$T+1$			T		
	Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	D_C	D_B	D_A
0	0	0	0	1	0	1	1	0	1
1	0	0	1	0	0	0	0	0	0
2	0	1	1	0	0	1	0	0	1
3	0	1	0	0	1	1	0	1	1
4	1	1	0	0	1	0	0	1	0
5	1	1	1	1	1	0	1	1	0
6	1	0	1	1	1	1	1	1	1

c)



$$D_C = \overline{Q}_A\overline{Q}_B + Q_AQ_C + \overline{Q}_BQ_C$$
$$D_B = Q_C + \overline{Q}_AQ_B$$
$$D_A = Q_B\overline{Q}_C + \overline{Q}_A\overline{Q}_C + \overline{Q}_A\overline{Q}_B + \overline{Q}_BQ_C$$

d)



10–31. ábra *M = 7 – szinkron-párhuzamos-GRAY-hátra számlánc*

idődiagramon alapuló működés, mely 0, 7, 6, ..., 1 ütemsorrendre, azaz dekrementáltan lépkedő: „M = 8 – aszinkron – soros – binér – hátra" típusú számlálóra utal.

A *szinkron hátra-számlálók* elvi rajza sem különbözik lényegében a 10-28. ábrán felrajzolttól. A 10-26. ábra értelmében hátra-számláláskor, a hálózat minden léptetéskor itt is egy *dekrementálást* végez, abban a kódrendszerben kifejezve, melyben a számláló működik. Áramköri kialakítás tekintetében ez a 10-28. ábránál azt jelenti, hogy a következő belső állapotok előkészítését végző KH hálózat tervezésekor a „visszafelé" sorrendet kell értelemszerűen biztosítani. Ezt a következő 10-31. ábra-beli példánál részletesebben illusztráljuk. A „lefelé" átvitelt pedig a BORROW-BW (kölcsön vagy maradék) betűkkel jelöljük az előző RC helyett.

Szemléltető példaként alakítsunk ki egy moduló M = 7-es szinkron–párhuzamos–GRAY–hátra számláncot. A realizációhoz használjunk „D" típusú tárolóelemeket.

Mivel itt nem áll rendelkezésre egy kereskedelemben kapható modul, a feladat valójában egy szinkron szekvenciális hálózat tervezését jelenti, ami – mint látni fogjuk – adott esetben nem lesz nehéz feladat. A számláló–szekvenciális–hálózat gráfját kiindulásként a 10-31a. ábra mutatja, ahol az M = 7-nek megfelelő 7 darab belső állapot a hét számlálási pozíciót fogja képviselni. Ugyancsak nem jelent gondot az állapotok kódolása sem, mivel a GRAY kód itt elegendően szükséges 3 bit-es változata a már tárgyalt 10-5. ábrából felírható.

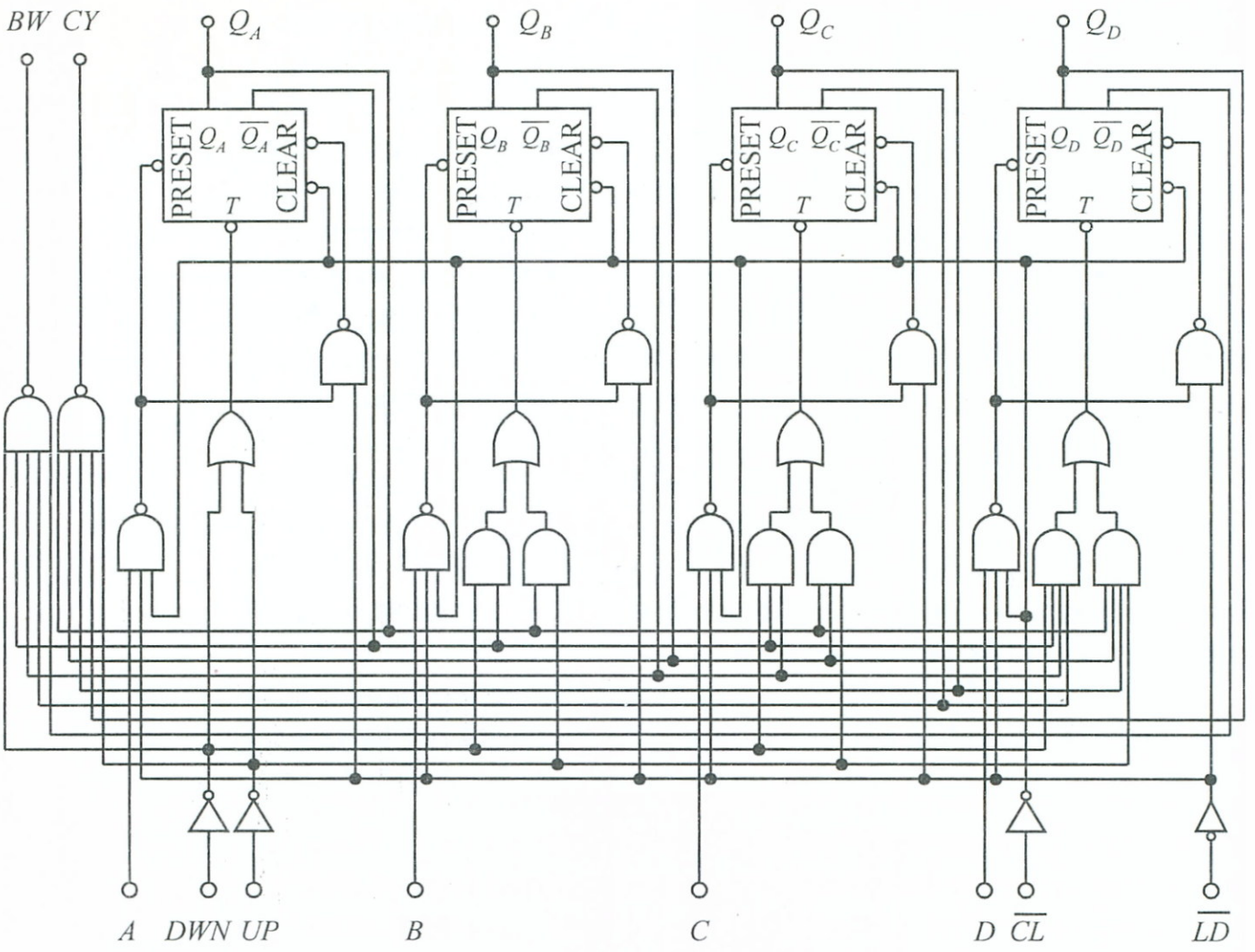
Kitölthető ezután a 10-31b. ábrán látható rovatos állapottábla és ebből a 2. fejezet 2-25. ábráján D tárolóra bevezetett

$$D_T = Q_{T+1} \qquad (10.16)$$

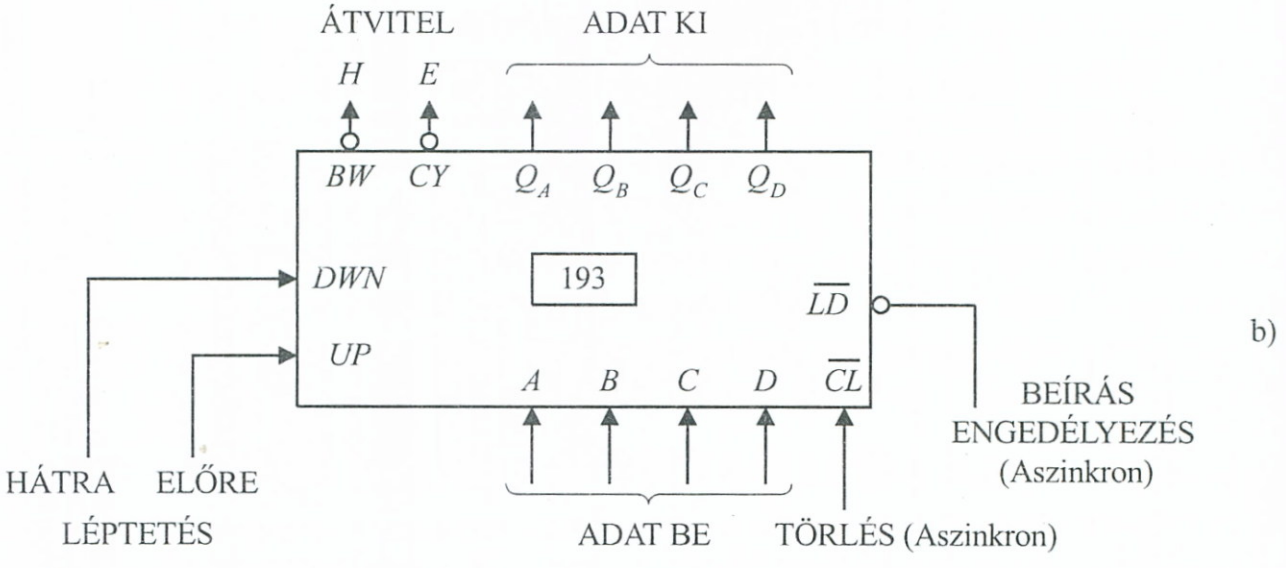
összefüggés felhasználásával a tárolóelemek $D_i = f(Q_A, Q_B, Q_C)$ függvényei is felírhatók V–K táblán egyszerűsítve a 10-31c. ábra szerint. Végül a 10-31d. ábrán felrajzoltuk a számláló részletes logikai vázlatát, melyet elláttunk aszinkron törlési (CL) lehetőséggel is.

10.5.4. Reverzíbilis számlálók

A reverzíbilis számlálók elvi felépítése főként abban tér el például a 10-28. ábrán felrajzolttól, hogy itt az előre, ill. hátra-irányok működtetéséhez két: $KH_{előre}$ ill. $KH_{hátra}$ kombinációs hálózatot kell alkalmaznunk, melyek aktivizálása az *előre*–választó (UP), ill. a *hátra*–választó (DOWN) vezérlő bemenetekkel történik, kizárásos ala-



a)



b)

10–32. ábra *M = 16 – szinkron-párhuzamos-binér-reverzibilis számlánc*

pon. A kétféle működési irány miatt mindkét átvitel típus (CARRY, BORROW) külön kimenetekkel rendelkezhet.

Szemléltető példaként a 10-32a. ábrán egy „193" típusú M = 16 – szinkron – párhuzamos – binér – reverzíbilis számláló modult rajzoltunk fel némi egyszerűsítéssel. A kapcsoláson belül jól elkülöníthető az UP-COUNT (UP), illetve a DOWN-COUNT (DWN) bemenetekkel aktivizálható $KH_{előre}$ ill. $KH_{hátra}$ kombinációs hálózatrész és megtalálhatók az előre átvitel (CY), ill. hátra átvitel (BW) csatlakozások.

A 10-32b. ábrán felrajzoltuk a számlánc csatlakozási vázlatát is.

10.5.5. Számlálók kialakítása adott feladatra

A számlálók konkrét feladatnál történő alkalmazására két út kínálkozik. Az *első* változatnál a kívánt kapacitásra (modulusra) egyedileg tervezünk számláló hálózatot a szekvenciális hálózattervezés módszereivel. A *második* változatnál felhasználjuk a kereskedelmi forgalomban kapható modulokat és ezeket „módosítjuk" az adott feladatnak megfelelően. Itt két probléma adódhat a felhasználandó számlánc modulusa (M_s) és az adott feladat által kívánt modulus (M_k) alapján:

$$\left.\begin{array}{ll} \text{a) } M_s > M_k \text{ esetén:} & \text{ciklus rövidítés} \\ \text{b) } M_s < M_k \text{ esetén:} & \text{bővítés} \end{array}\right\} \qquad (10.17)$$

szükséges. Vizsgáljuk meg mindkét esetet:

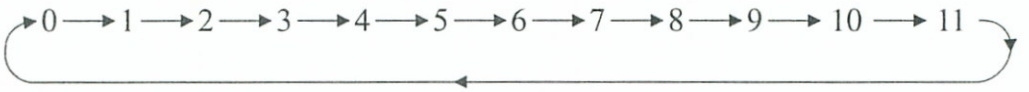
a) Ciklus-rövidítés

A számláló ciklusának rövidítése valójában a modulus csökkentését jelenti. Ezt legtöbbször *logikai visszacsatolás* alkalmazásával valósítjuk meg, oly módon, hogy a visszacsatoló ágat a CL, ill. LD csatlakozásokra visszavezetve, a számlálót nem hagyjuk végigmenni az elvileg lehetséges teljes cikluson. A visszacsatoló ág egy olyan ÉS feltételből indul, mely az új M_k modulusnak felel meg. Itt külön kell figyelnünk arra, hogy a CL, ill. LD bemenetek szinkron vagy aszinkron természetűek-e, ugyanis a visszacsatolási *ÉS feltétel kódja:*

$$\left.\begin{array}{ll} \text{a) szinkron CL ill. LD esetén:} & N_{vs} = K_u \\ \text{b) aszinkron CL ill. LD esetén:} & N_{va} = K_u + 1 \end{array}\right\} \qquad (10.18)$$

kell, hogy legyen, ahol K_u a ciklusból való kiugrás kódértéke.

Vizsgáljuk meg a 10-29b. ábra moduljából kiindulva a visszacsatolás kialakítását $M_s = 16$ modulusú – binér előreszámlálót feltételezve, olyan esetre, midőn a kívánt ciklus:



Vizsgáljuk meg a szinkron és aszinkron változatot is.

A kívánt modulus a ciklus alapján: $M_k = 12$. Miután $M_s > M_k$, ezért *ciklus rövidítést* kell alkalmaznunk. Mivel a ciklus tartalmazza a 0 – 0000 állapotot is, ezért a visszacsatolást a CL bemenetre vezethetjük vissza. (Ha a 0 nem volna a ciklusban, akkor az LD bemenetet kellene értelemszerűen használnunk.) A szinkron, ill. aszinkron működésű CL-et feltételező két változatot a 10-33. ábrán rajzoltuk fel az ÉS feltétel kódjának fent részletezett beállításával, $K_u = 11$ behelyettesítésével:

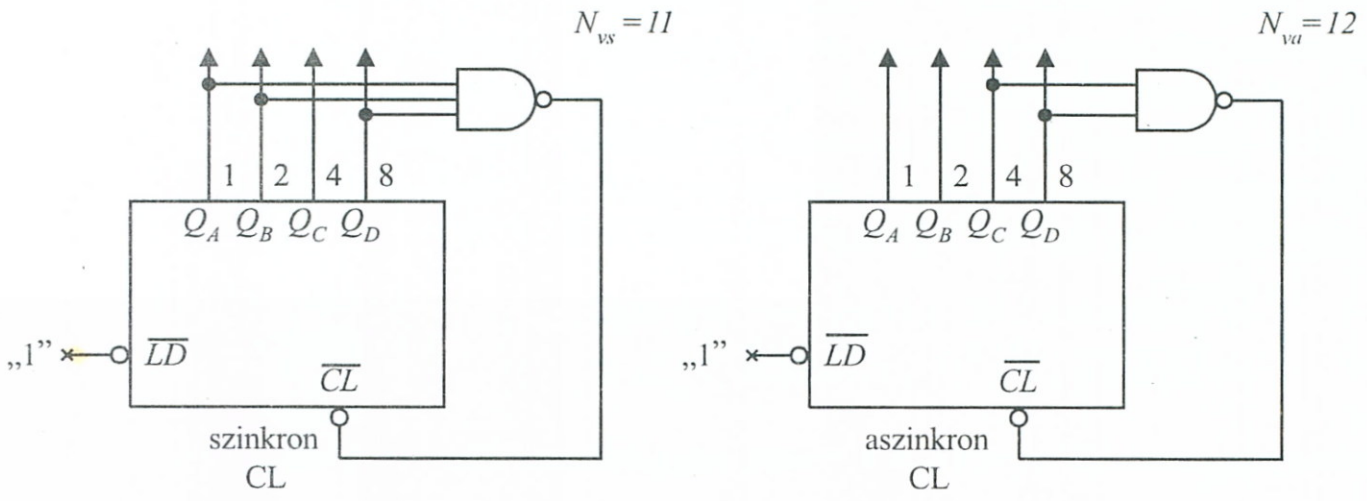
$$N_{vs} = 11$$

$$N_{va} = 11 + 1 = 12$$

A nem használt bemeneteket közömbös állapotba kell állítanunk. (Pl.: $\overline{LD} = 1$ (H))

b) Ciklus-bővítés

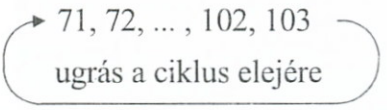
A számláló ciklusának bővítése a modulus növelését jelenti, azaz a feladatot *több modul-elem* összerakásával kell megoldanunk. Itt felhasználásra kerülhetnek az adott modul áramkör átvitellel kapcsola-



10–33. ábra *Ciklusrövidítés szinkron ill. aszinkron CL bemenetnél*

tos csatlakozásai is (pl. RC, EP, ET stb.). Vizsgáljuk meg a bővítést egy összetettebb példa keretében.

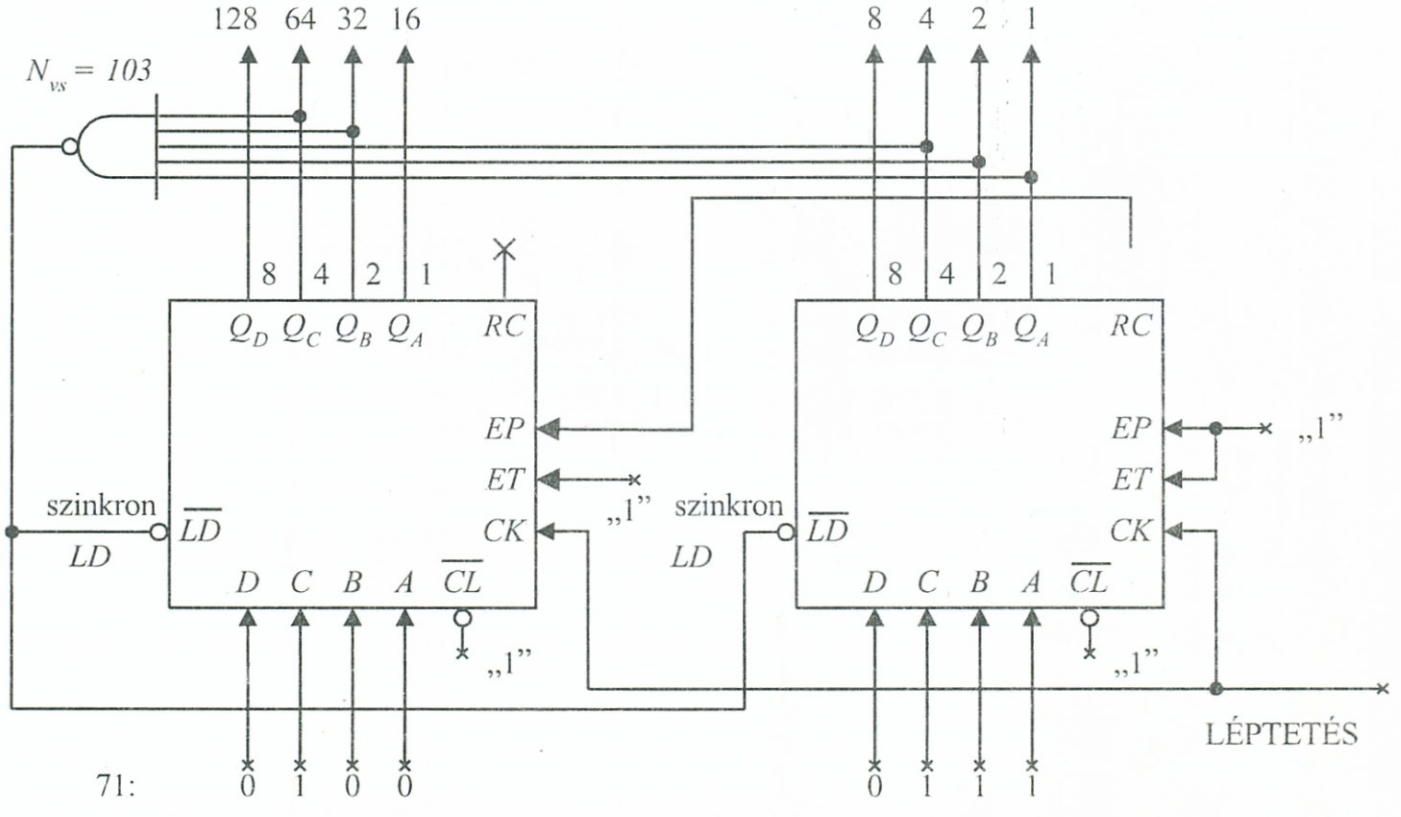
Gyakorlásul készítsünk egy olyan szinkron számlálóművet, amely az alábbi ciklusban dolgozik binér kódban:



Mint látható, a 0 nem szerepel a ciklusban, így a $\overline{CL} = 1$ (H) bemenet-közömbösítést kell biztosítanunk. Mivel az egyes állapotokban felvett értékek kódolására egy négybites számlánc nem elegendő (a max. 103 érték miatt), két négybites binér láncot kell bővített formában összekapcsolnunk (10-34. ábra). A ciklus 71-ből indulva lépked előre 103-ig, majd ennek elérését követően ugrik vissza 71-re. A visszacsatolási ÉS feltétel kódja binárisan 8 bit-en:

$$N_{vs} = 103_{10} = 01100111_2$$

Ezt a kikapuzott értéket vezetjük vissza, mint párhuzamos beírás engedélyező jelet a feltételezetten szinkron típusú LD bemenetre, mely nyitáskor a nyolc párhuzamos Preset bemenetre kötött konstans:



10–34. ábra *Ciklus bővítés szinkron LD bemenetnél*

$$71_{10} = 01000111_2$$

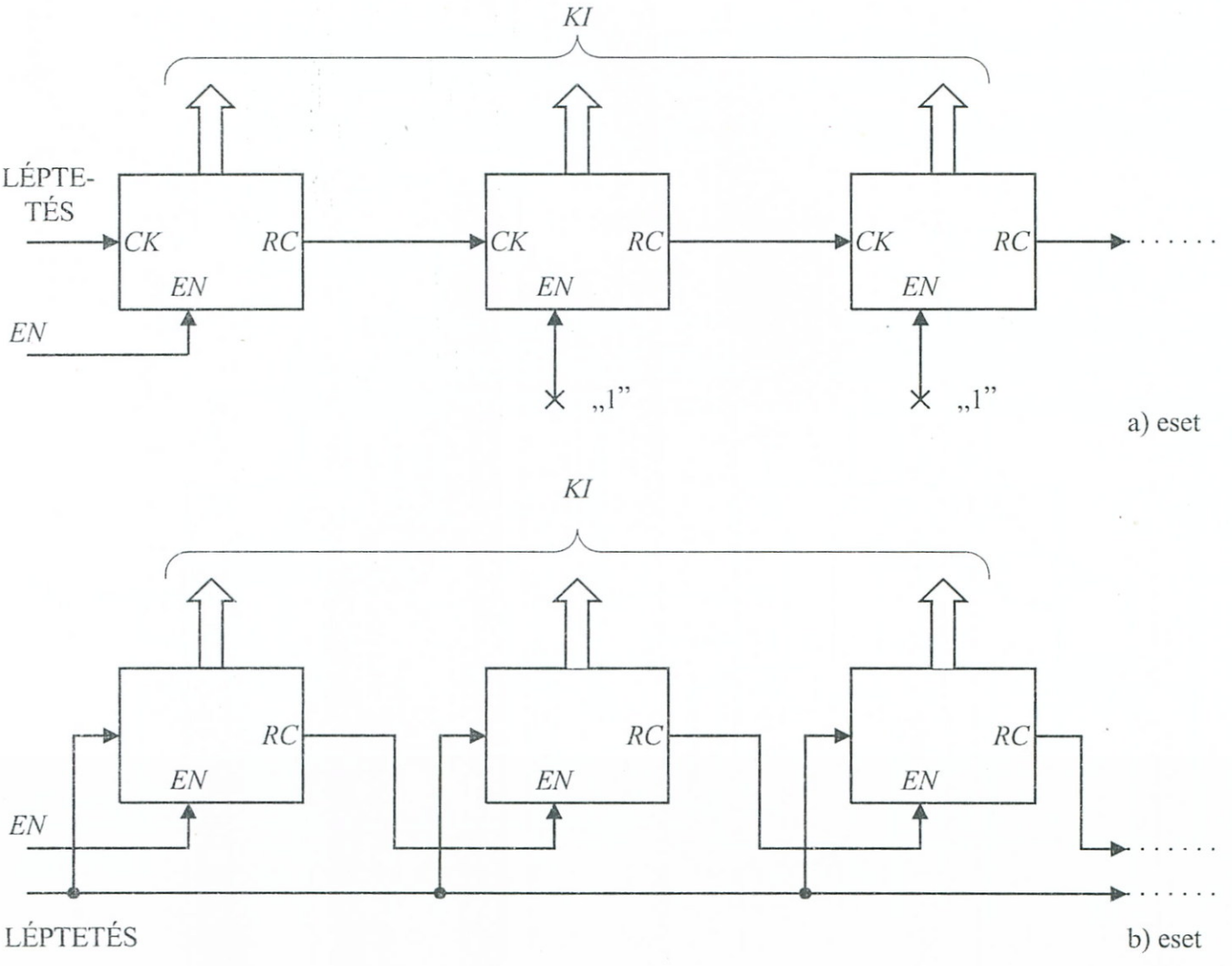
ciklusindító kódot íratja be a bővített számláló tárolóiba. A következő léptető jelnél a működés innen folytatódik tovább.

c) Bővítési változatok

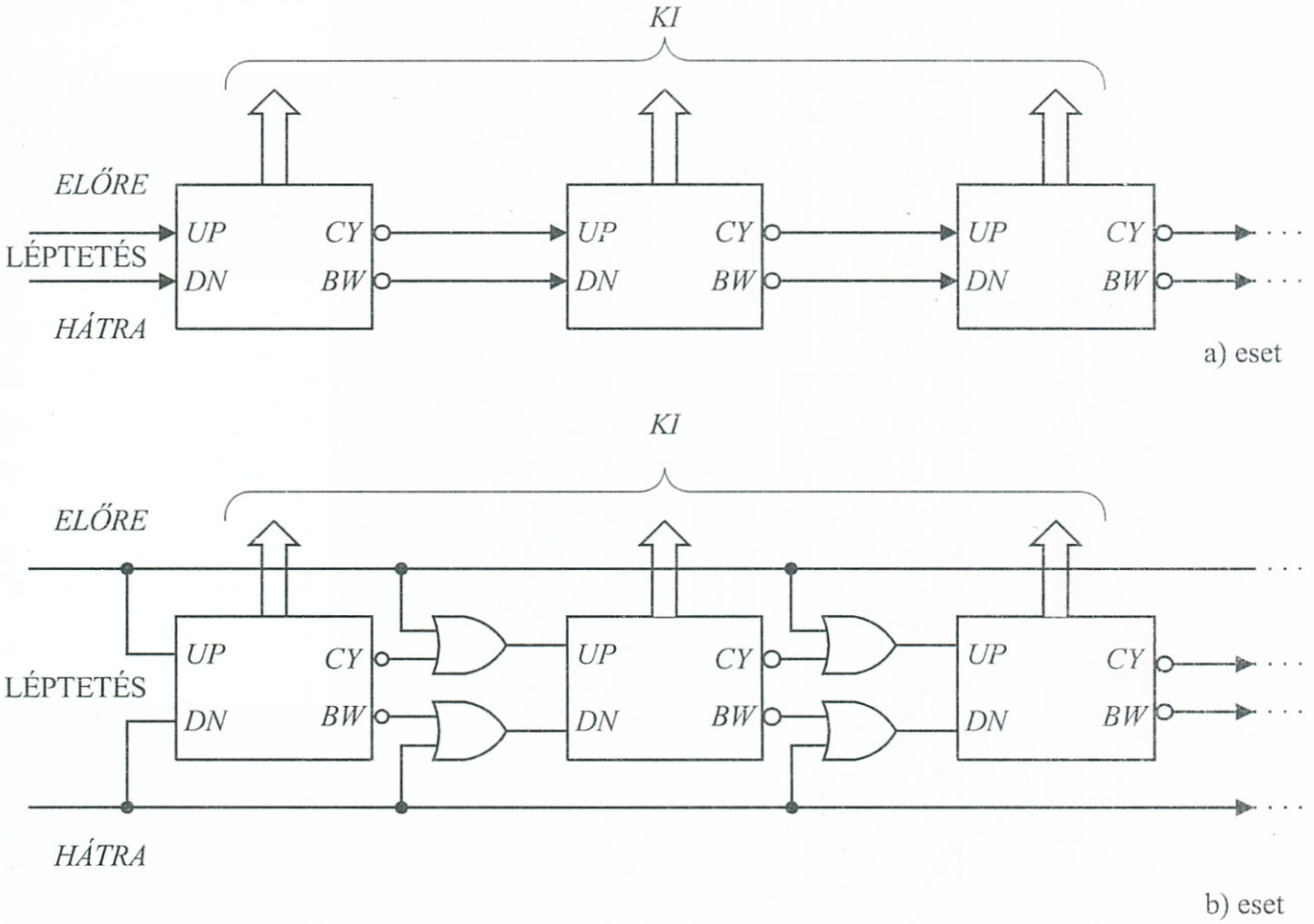
Az előző pontban bemutatott számlánc bővítésnek több elvi változata is elképzelhető:

I. Az *egyirányban* működő szinkron számlánc típusú eseteknél a 10-35. ábra szerinti elvi bővítési változatok képzelhetők el. Itt az

a) eset: az egyes modulok között *aszinkron* kapcsolatot létesít, az egyes bővítő fokozatok órajelét az előző modul átvitele szolgáltatja,



10–35. ábra *Egyirányú számlánc elvi bővítési változatok*



10–36. ábra *Reverzibilis számlánc elvi bővítési változatok*

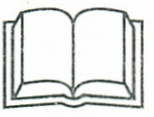
b) eset: az egyes modulok egyszerre kapják órajeleiket, míg a következő fokozat engedélyezését az előző fokozat *átvitele* szolgáltatja.

II. A *reverzíbilis* számlálókra bemutatott példabeli számlánc-típusnál a 10-36. ábra típusú bővítéseket alkalmazhatjuk. Itt az

a) eset: az egyes modulok *soros* láncolatát jelenti és rokonságban áll az előbbi I.a. esettel.

b) eset: itt ugyancsak az *átvitel* hozza létre a modulok közötti *kapcsolatot,* hasonlóan az I.b. esethez.

10.5.6. Számlálókkal felépített frekvencia-osztók



Digitális rendszerek kialakítása során gyakori feladat, hogy valamely adott frekvenciájú impulzus-sorozatot valamilyen alacsonyabb frekvenciájúra osszunk le. Ilyen feladat adódhat például nagypontosságú kvarc oszcillátorok viszonylag nagy szaporaságú impulzus-sorozatának kívánt lassúságú órajellé konvertálása esetén stb. A szám-

láló hálózatok segítségével ez a „leosztás" nagy pontossággal és viszonylag egyszerűen elvégezhető.

Ha f_k = kimeneti-, f_b = bemeneti frekvencia és M = a számláló modulusa, akkor felírható:

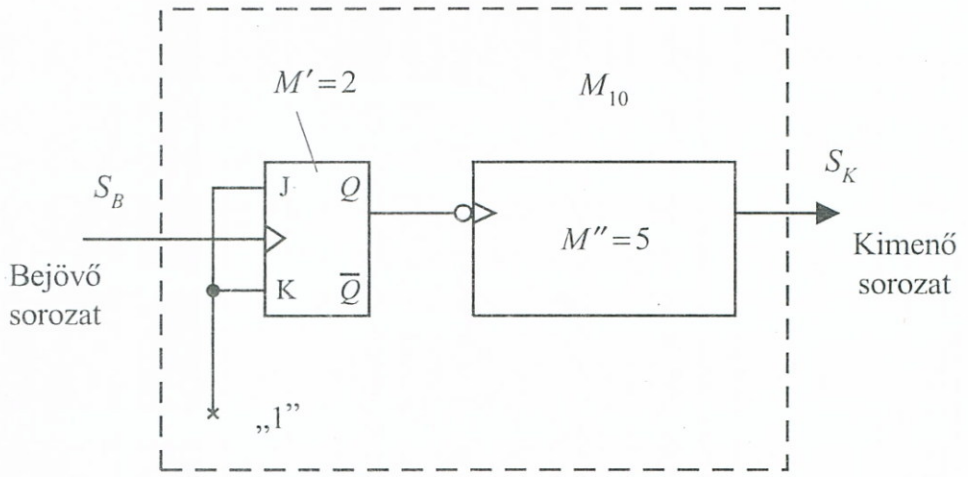
$$f_k = \frac{f_b}{M} \tag{10.19}$$

Már a tárolóelemek korábbi tárgyalásakor láthattuk, hogy egy T-tárolóelemnél (mivel a működtető órajelnek csak valamelyik jelhomlokára billent) a Q kimeneten jelentkező 1-esekből álló impulzussorozat frekvenciája a *fele* volt a T bemenetre érkezőének. A T-tárolóelemet ezért a legegyszerűbb (1:2 arányú) frekvencia-osztónak tekinthetjük.

A különféle osztásarányok előállításához felhasználhatók a számlálók ciklus-rövidítésével és bővítésével kapcsolatosan korábban elmondottak.

Általános szabály, hogy az osztási tényezőt *törzstényezőkre* kell bontani és az egyes törzstényezőknek megfelelő osztóláncokat *sorba* kell kapcsolni, ekkor valójában a modulusok szorzata állítja elő az osztási tényezőt. Ilyen esetre mutat példát a 10-37. ábrán bemutatott $M_{10} = M_2 \times M_5$ felépítésű számlánc.

Az ábrán az első fokozat egy J–K tárolóból kialakított T tároló, ez mint említettük, egy felező osztó, a második fokozat valamely $M'' = 5$ modulusú számlánc.



Eredő modulus: $M = 2 \times 5 = 10$

Kimenő frekvencia: $f_K = \frac{f_B}{M} = \frac{f_B}{10}$

10–37. ábra

A frekvenciaosztók kialakításánál, külön feladatot jelent a kimeneti (leosztott frekvenciájú) impulzus-sorozat 1-0 (Jel–Szünet) viszonya, mivel ezek általában nem egyformák. Ha az egyenlőség igény, akkor például egy gyakran alkalmazott megoldásnál a sorbakapcsolásokat – lehetőség szerint – úgy csoportosítják, hogy a kimenetre csatlakozó fokozat egy $M = 2$-es osztó legyen.

A 10-37. ábra-beli példánál, ha a két fokozat sorrendjét felcseréljük, akkor a kimenő frekvencia nem változik, viszont az S_k sorozat jel–szünet viszonya 1:1-re alakul.

Bonyolultabb esetként vizsgáljuk meg azt az osztási esetet, midőn az osztási tényező olyan, hogy ugyan nem hatványa 2-nek, de eggyel csökkentve már egy 2^n értékkel lesz egyenlő.

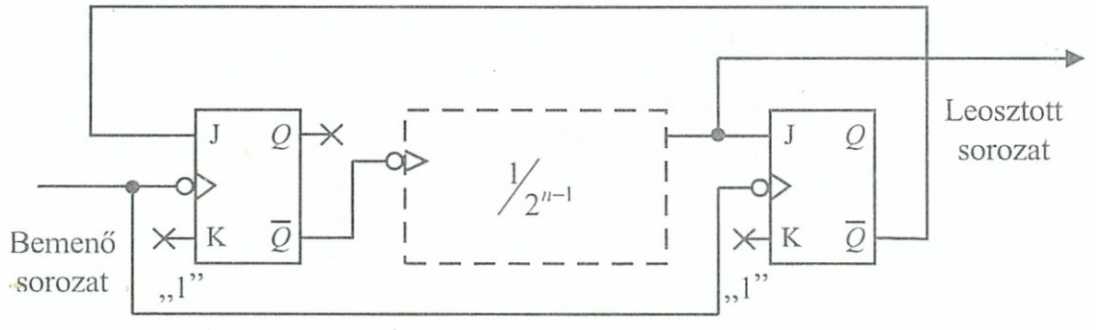
Ilyen esetben jól alkalmazható a 10-38. ábra elvi vázlata, melynél logikai visszacsatolást is beiktattunk.

Végezetül visszautalunk a 10-27. ábra binér számláncára, mint frekvencia-osztóként értelmezett példára. Ennek a bemenethez legközelebbi tárolójának kimenetén fél-, a második tárolón negyed-, a harmadikon nyolcad... (stb.) frekvenciájú jelek jelennek meg a B bemenethez viszonyítottan, azaz:

$$f_{QA} = \frac{f_B}{2}$$

$$f_{QB} = \frac{f_B}{4}$$

$$f_{QC} = \frac{f_B}{8}$$

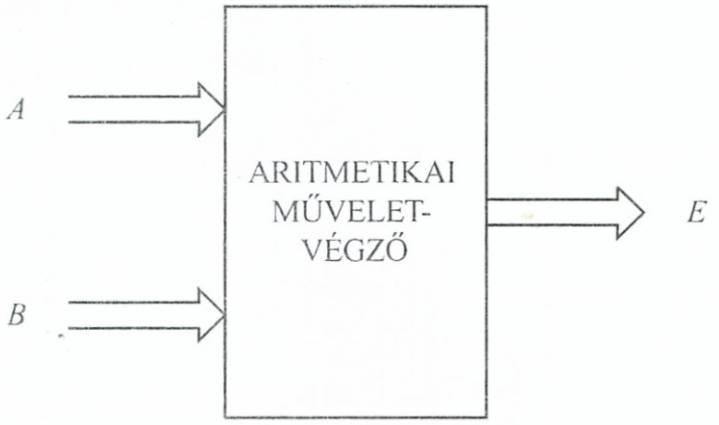


10–38. ábra *1:2ⁿ+1 frekvenciaosztó elvi vázlata*

10.6. Aritmetikai műveletvégzők

Aritmetikai műveletvégzés mind digitális, mind analóg elven megvalósítható. Ebben a fejezetben a digitális aritmetika alapeseteivel foglalkozunk. Analóg műveletvégzőkkel a 4. fejezetben a *műveleti erősítők*nél találkoztunk.

Informatikai rendszerekben elsősorban digitális aritmetikai feladatok fordulnak elő. A digitális aritmetikai műveletvégzőket a 10-39. ábra kapcsán tanulmányozhatjuk:



10–39. ábra *Aritmetikai műveletvégző elvi vázlata*

A digitális műveletvégző az *A* és *B* bemenetekre érkező, valamilyen kódban kifejezett számokkal aritmetikai műveleteket végez (pl. összeadás-kivonás, szorzás, osztás stb.), és a kapott eredményt az *E* kimeneten jeleníti meg, ugyancsak valamilyen kód formájában. Az elvi ábra elsősorban *két operandusos* műveletekre utal, természetesen elképzelhetők több operandusos esetek is.

Az aritmetikai műveletek rendkívül sokfélék lehetnek, és az őket realizáló hardver egységek választéka is igen nagy, melyek tanulmányozásával a digitális rendszertechnika tématerület foglalkozik. Ebben a könyvben bevezető célzattal a digitális összeadás, -kivonás, -szorzás és -osztás alapjait mutatjuk be néhány példa keretében.

10.6.1. Számábrázolás

A digitális számokkal végzett műveletek elvégzésének módozatai a számábrázolás módjától függenek. A bináris számokat a digitális rendszereknél:

– fixpontos és
– lebegőpontos

elv szerint szokták ábrázolni.

10.6.1.1. Bináris fixpontos számábrázolás

A fixpontos ábrázolás azt jelenti, hogy a szám egész- és tört- részeit elválasztó „pont" (tízedes pont, bináris pont) a műveletek során állandó (fix) pozícióban marad. A fixpontos ábrázoláson belül az

– előjel – abszolútértékes és a
– komplemenses

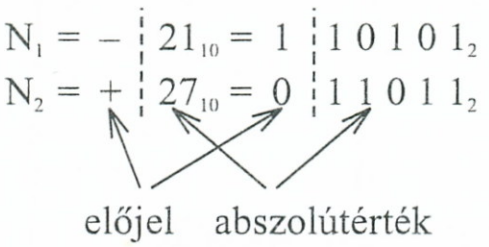
ábrázolásmódokat vizsgáljuk meg a továbbiakban.

a) Bináris „előjel-abszolútértékes" ábrázolás

Bináris számok esetén – miután a számot alkotó szimbólumok csak a 0, 1 értékekre korlátozódnak, továbbá a számok ábrázolására felhasznált fizikai eszközök túlnyomó többségükben kétállapotúak – az előjel ábrázolását is célszerű a 0, 1 szimbólumokkal elvégezni egy újabb „előjel-helyérték" bevezetésével, melynél általában a következő összerendelést alkalmazzuk:

0 – pozitív
1 – negatív

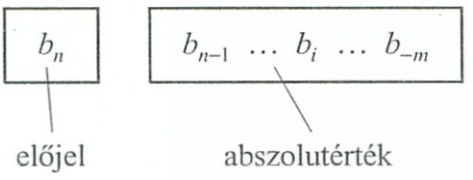
Például, írjuk fel az $N_1 = -21_{10}$ és $N_2 = +27_{10}$ decimális számokat és a megfelelő bináris számokat előjel-abszolútértékes ábrázolás-módban:



Általánosságban egy fixpontos bináris szám előjel-abszolútértékes ábrázolásban a következő formulával adható meg:

$$N = -b_n \cdot 2^n + \sum_{i=-m}^{n-1} b_i \cdot 2^i \qquad (10.20)$$

ahol: m – a tört rész számjegyeinek száma
n – az egész rész számjegyeinek száma
b_i – a számjegy szimbólum, melyre fennáll:
$0 \le b_i \le 1$
továbbá: $b_n = 0$ pozitív számnál
$b_n = 1$ negatív számnál.



10–40. ábra *Előjel-abszolutértékes ábrázolás*

Ez megfelel a (már kissé a későbbi realizációra is utaló) 10-40. ábrának.

Az előjel-abszolútértékes ábrázolásnál fel kell hívnunk a figyelmet az ún. „kétféle nulla" problémájára, mely a (10.20) formulánál akkor áll elő, ha zérus abszolútérték mellett az előjel helyértéknél b_n helyébe vagy 0-át, vagy 1-et helyettesítünk. Az így fellépő pozitív, ill. negatív nulla a digitális rendszerek működésénél zavart okozhat, melyre alkalmanként külön figyelmet kell fordítani.

Összeadás és kivonás előjel-abszolútértékes ábrázolásnál

A bináris összeadás és -kivonás egy-helyértékes elvégzésének szabályait a 10-41. ábra művelettáblázataival adhatjuk meg.

A kivonási művelettáblázatoknál jól látszik, hogy olyankor, mikor a kisebbítendő eleve kisebb, mint a kivonandó, a kivonás csak úgy végezhető el, ha a kisebbítendőt a megelőző, eggyel magasabb helyér-

Összeadási esetek:

	0		1		0		1	OP 1
+	0	+	0	+	1	+	1	OP 2
	0		1		1		0	ÖSSZEG

átvitel (CARRY) a magasabb helyérték felé

Kivonási esetek::

áthozat (kölcsönv. BORROW) a magasabb helyérték felől

	0		1		1		0	OP 1
–	0	–	0	–	1	–	1	OP 2
	0		1		0		1	KÜLÖNBSÉG

10–41. ábra *Művelettáblázatok bináris egyhelyértékes esetre*

tékről „kölcsönvett" értékkel megnöveljük. Ezt a „kölcsönzős" módszert az iskolából már jól ismerjük, a tanult papír-ceruza módszeres kivonást annak idején lényegében így végeztük el. A gépi kivonást – különösen, ha a több helyértéknél fellépő „láncolódó kölcsönzésekre" gondolunk – nagyon nehézkessé tenné e módszer követése. Többek között emiatt és az előjel-abszlútértékes ábrázolásnál felvetődő más gondok (pl. az említett kétféle nulla stb.) elkerülésére vezetjük be a *komplemens*-en alapuló ábrázolást, mellyel a következő pontokban foglalkozunk.

b) Ábrázolás „komplemens"-sel

Kiindulási példaként legyen adva egy fogyasztásmérő „villanyóránk", mely mondjuk $M = 10\,000$ kWó-ig tud mérni. Nevezzük ezt a határértéket *M-modulus*nak. Ha óránkon a fogyasztásérték jelenleg $f = 3700$, akkor az óra még:

$$k = M - f = 10\,000 - 3700 = 6300\text{-at}$$

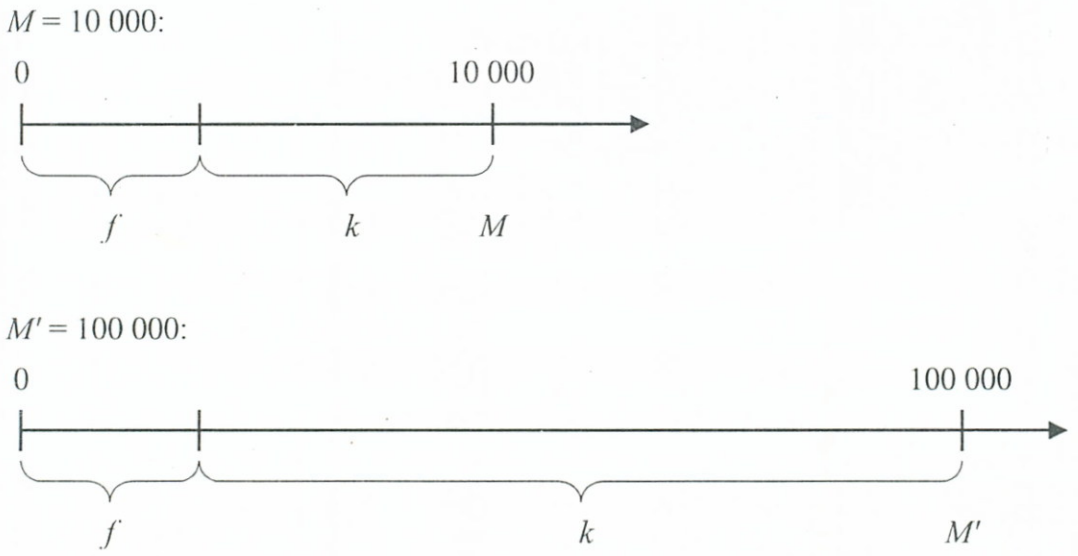
tud számolni. A k értéket az f kiegészítőjének – *komplemensének* nevezzük. Ügyelni kell arra, hogy a komplemens nem csak *f*-től, hanem *M*-től is függ:

$$k = k(f, M)$$

Például: $M' = 100\,000$ *modulusú óránál ugyanarra az* $f = 3700$-ra:

$$k' = M' - f = 100\,000 - 3700 = 96\,300$$

A komplemenset számegyenesen is szemléltethetjük mindkét példabeli esetre a 10-42. ábra szerinti módon.



10–42. ábra *Komplemens a számegyenesen*

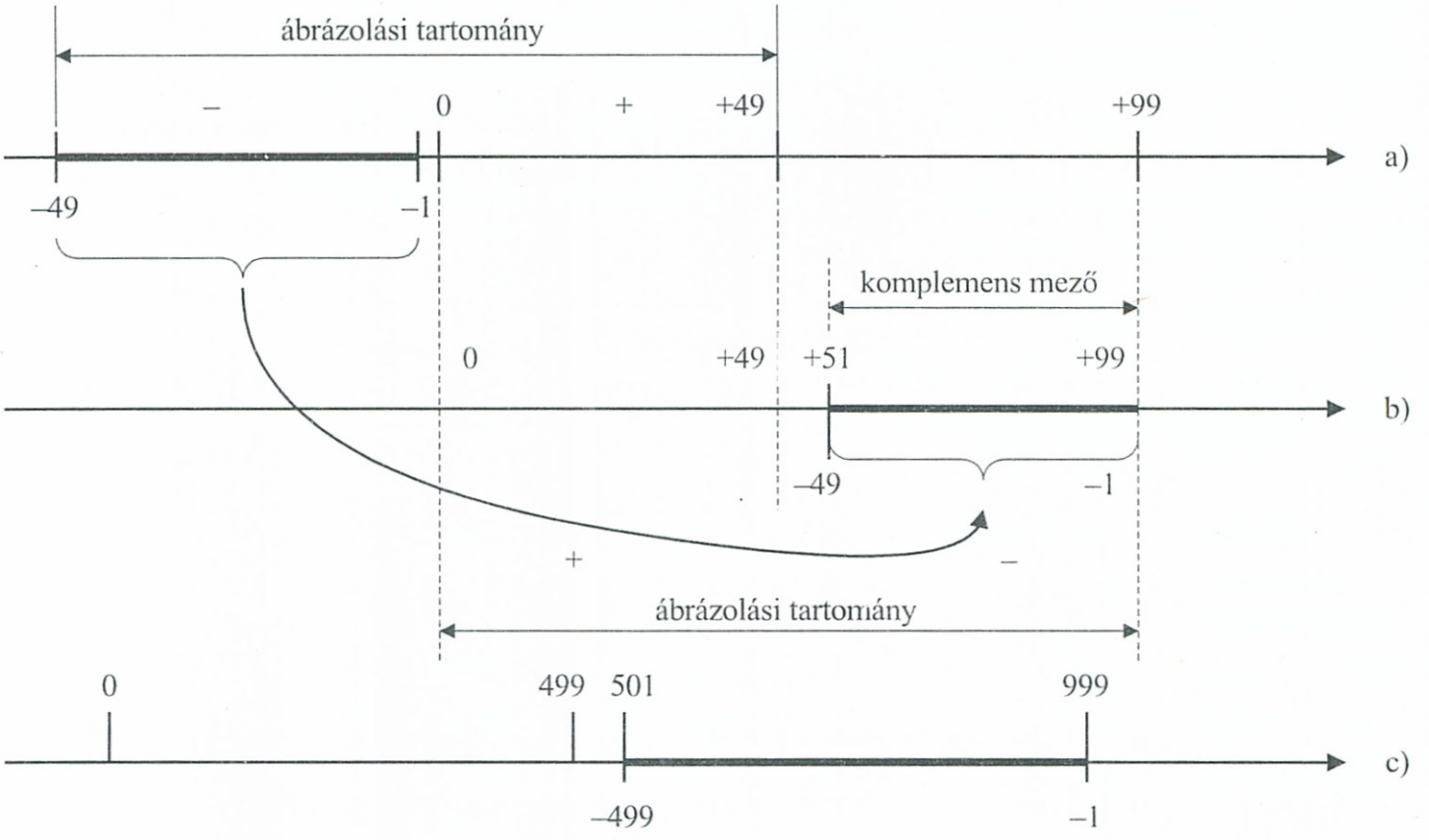
A komplemens felhasználása negatív számok ábrázolására

Példaként tegyük fel, hogy kéthelyértékes számokat akarunk ábrázolni, csak

–49-től +49-ig

a számegyenesen. Ez a megszokott előjel-abszolútértékes ábrázolásmódban a 10-43a. ábra szerint alakul. Már láttuk, hogy itt a 0-tól balra, ill. jobbra lévő zónákat csak külön beiktatott *előjel*-lel tudjuk megkülönböztetni. Ennek a „kényelmetlen"előjelnek kiküszöbölésére alkalom adódik, ha a –49-től –1-ig terjedő negatív tartományt, a 10-43b. ábra szerinti módon áthelyezzük a +51-től +99-ig terjedő komplemens tartományba oly módon, hogy pl. –49-nek: +51, ... –1-nek: +99, azaz az ábrázolási tartomány szélességére jellemző modulusnak megfelelő komplemenset feleltetünk meg. Ezzel a leképezéssel az ábrázolási tartományt teljes egészében a pozitív mezőbe helyeztük át, így az előjelekkel már nem is kell közvetlenül foglalkoznunk. Ebben az ábrázolásban pl. a 73-as szám láttán (az $M = 100$-as modulus hallgatólagos tudtával) automatikusan a –27-re (100 – 73)-ra gondolunk.

A példa kapcsán természetesen felvetődik a kérdés, mi lesz a komplemens mezővel lefedett +50 és e feletti kétjegyű számok sorsa.



10–43. ábra *Komplemens felhasználása negatív szám ábrázolására*

A probléma úgy hidalható át egyszerűen, ha az előbbinél nagyobb modulust választunk (pl. $M = 1000$-ret), azaz az ábrázolást kiterjesztjük három helyi értékre. Ilyenkor például a +65 ábrázolásakor (mely $M = 100$-nál $100 - 65 = -35$-öt jelentene) 065-öt kell írnunk. A három helyi értékes ábrázolásnál az ábrázolható számtartományt –499-től +499-ig tudjuk kiterjeszteni a 10-43c. ábrának megfelelően.

A példák általánosított tapasztalatai alapján felírhatók a komplemensekre vonatkozó alábbi összefüggések:

$$\left.\begin{array}{l}\text{pozitívszám: } N_K = N \\ \text{negatívszám: } N_K = M - /N/\end{array}\right\} \qquad (10.21)$$

Bináris számábrázolás komplemens felhasználásával

A (10.20) összefüggést bináris előjel-nélküli *egész* számra alkalmazva felírható:

$$N = \sum_{i=0}^{n-1} b_i \cdot 2^i \qquad (10.22)$$

Ha *inverz bináris szám*-nak nevezzük azt a számot, melyet az eredetiből úgy kaptunk, hogy minden $\overline{b}_i$ eleme helyett annak ellentétjét (0 helyett 1-et és fordítva) -et helyettesítünk be, akkor az inverzre felírható:

$$N_I = \sum_{i=0}^{n-1} \overline{b}_i \cdot 2^i \qquad (10.23)$$

Amennyiben az $N + N_I$ összeget képezzük, akkor megkapjuk az adott helyérték-számmal elérhető legnagyobb számot, mely bináris esetünkben n darab 1-esből áll:

$$N + N_I = \sum_{i=0}^{n-1} b_i \cdot 2^i + \sum_{i=0}^{n-1} \overline{b}_i \cdot 2^i = 2^n - 1$$

Például, ha: $N = 101101_2$ (45_{10}), $N_I = 010010$

$$N + N_I = 101101 + 010010 = 111111_2$$

Ha az utóbbi összefüggést a (10.24) összefüggéssé alakítjuk át, megkapjuk a *bináris komplemens* kiszámítására alkalmas formulát:

$$-N = 2^n + \sum_{i=0}^{n-1} \overline{b}_i \cdot 2^i + 1 = -2^n + \underbrace{N_I + 1} \qquad (10.24)$$

Itt, mint látható, az *előjel bit* (2^n) negatív súlyozású. Egyidejűleg leolvasható a komplemens számérték kiszámításának algoritmusa is:

$$\begin{aligned} &\text{I. lépés: képezzük az inverzet:} && N \rightarrow N_I \\ &\text{II. lépés: hozzáadunk 1-et:} && N_I + 1 \end{aligned} \qquad (10.25)$$

Példaként írjuk fel $N = 25_{10}$ bináris megfelelőjének komplemensét, amely (amennyiben a negatív számot a komplemenssel fejezzük ki), egyben -25_{10} bináris megfelelője is.

A 25_{10}-hez számértékileg elegendő lenne 5 bináris helyi érték, mivel az ezen ábrázolható legnagyobb szám: $11111_2 = 31_{10} > 25_{10}$. Ebbe a modulusba azonban a komplemens ábrázolás már nem férne bele, ezért legalább 6 helyi értékkel kell ábrázolnunk. Itt:

$N = 25_{10} =$	0	1	1	0	0	1_2	Eredeti szám
	2^5	2^4	2^3	2^2	2^1	2^0	
	32	16	8	4	2	1	
I. lépés: N_1	1	0	0	1	1	0_2	Inverz
II. lépés: +1	+0	0	0	0	0	1_2	
$N_K = -N$	1	0	0	1	1	1_2	Komplemens

előjel helyérték

Mint látjuk, az előjel helyérték alapján, az eredmény valóban negatív. Ezt különben számszerűen is ellenőrizhetjük, ha a (10.24) formula negatív súlyozású legnagyobb előjel-helyértékét figyelembevéve, újra decimálisba számítjuk át a kapott N_K eredményt:

$$\begin{aligned} N_K &= -1\cdot 2^5 + 0\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 20 = \\ &= -32 + 0 + 0 + 4 + 2 + 1 = -25 \end{aligned}$$

A példánál azt is szemléltethetjük, hogy N-hez N_K-t hozzáadva egy helyértékkel nagyobb számhoz jutunk, mely jelen esetben a modulusra is utal:

$N =$	0	1	1	0	0	1_2
$N_K =$	1	0	0	1	1	1_2
	1	0	0	0	0	0

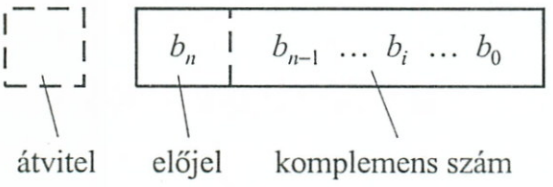
Kivonás visszavezetése összeadásra

Fentiek alapján a továbbiakban két pozitív szám kivonását, mint egy pozitív és egy negatív szám összeadását értelmezzük és a komplemens alkalmazásával a feladatot kizárólag pozitív előjelű mennyiségek tartományába visszük át.

Bináris összeadás-kivonás komplemens ábrázolásmódban, túlcsordulás

Az előzőekben már láttuk, hogy komplemens ábrázolásmódnál – lényegében – csak összeadással kell foglalkoznunk. Vezessük be az ábrázoláshoz a 10-44. ábrát, mely első látásra hasonlít ugyan az előjel-abszolútértékes ábrázolás 10-40. ábrájához, de értelmezése valójában teljesen eltér attól, mivel a (10.21) összefüggéseken és a (10.24) formulán alapul. Az átvitel ábrázolása csak azért történt, hogy a következő szemléltető-példákkal jobban összhangban lehessünk.

További meggondolások érdekében tekintsünk adottnak két operandust (pl. $N_1 = 3_{10}$ és $N_2 = 6_{10}$), végezzünk velük összeadást mind a négy lehetséges előjelkombináció feltételezésével, a 10-44. ábra szerinti ábrázolásmódban:



10–44. ábra *Komplemens bináris számábrázolás*

I. eset: N_1 és N_2 egyaránt pozitív

$N_1 = +3_{10}$:	0	$0\ 0\ 1\ 1_2$
$N_2 = +6_{10}$:	0	$0\ 1\ 1\ 0_2$
Összeg $= +9_{10}$	0	$1\ 0\ 0\ 1_2$

pozitív az eredmény

II. eset: N_1 negatív és N_2 pozitív

$N_1 = -3_{10}$:		1	1	1	0	1_2	→ komplemens, a negatív előjel
$N_2 = +6_{10}$:		0	0	1	1	0_2	miatt: 0 0 0 1 1
Összeg $= +3_{10}$:	1	0	0	0	1	1_2	1 1 1 0 0
	↓	↓					+0 0 0 0 1
	most érdektelen átvitel	pozitív eredmény					1 1 1 0 1

III. eset: N_1 pozitív és N_2 negatív

$N_1 = +3_{10}$:	0	0	0	1	1_2	
$N_2 = -6_{10}$:	1	1	0	1	0_2	→ komplemens ismét a negatív
Összeg $= -3_{10}$:	1	1	1	0	1_2	előjel miatt: 0 0 1 1 0
	↓					1 1 0 0 1
	az eredmény negatív!					+0 0 0 0 1
	helyességét visszakomple mentálással ellenőrizhetjük:					1 1 0 1 0

1 1 1 0 1
0 0 0 1 0
+0 0 0 0 1
0 0 0 1 1 → tehát 3, mely természetesen negatív 3

IV. eset: N_1 és N_2 egyaránt negatív

$N_1 = -3_{10}$:		1	1	1	0	1_2 →	} mindkét komplemenst
$N_2 = -6_{10}$:		1	1	0	1	0_2 →	} fenn már kiszámítottuk
Összeg $= -9_{10}$:	1	1	0	1	1	1_2	
	↓	↓					
	most érdektelen átvitel	az eredmény negatív!					
		Ellenőrző visszakomplementálás:					

1 0 1 1 1
0 1 0 0 0
+0 0 0 0 1
0 1 0 0 1 → negatív 9-nek felel meg

Végezetül egy példa, amely az ún. *túlcsordulás* (overflow) jelenségére hívja fel a figyelmet. Ez a jelenség valójában a kicsinek választott modulus következménye:

Legyen: $N_1 = 8_{10}$, $N_2 = 9_{10}$

N_1 =	$+8_{10}$:	0 :	1 0 0 0_2
N_2 =	$+9_{10}$:	0 :	1 0 0 1_2
Összeg =	$+17_{10}$:	1 :	0 0 0 1_2

számszerűen rossz!

látszólag negatív, ami téves, mert két pozitívot adtunk össze!

Modulus (helyértékszám) növeléssel:

N_1 =	$+8_{10}$:	0 :	0 1 0 0 0_2
N_2 =	$+9_{10}$:	0 :	0 1 0 0 1_2
Összeg =	$+17_{10}$:	0 :	1 0 0 0 1_2

Eredmény pozitív, és számszerűen is helyes!

10.6.1.2. Lebegőpontos számábrázolás

A lebegőpontos számábrázolás nagyobb tartományok lefedését és nagyobb számítási pontosság elérését teszi lehetővé. A számábrázolás itt a (10.26) formula alapján történik:

$$N = s, m \cdot r^k \qquad (10.26)$$

ahol: N – az ábrázolt szám
s – előjel
m – mantissza
r – radix, a számrendszer alapszáma
k – karakterisztika (előjeles kitevő).

Mint látható, itt s, m révén a mantisszánál előjel-abszolútértékes ábrázolásmóddal találkozunk.

Például: a -519_{10}-es decimális számot megadhatjuk a következő módon (10.26) alapján:

$$-5, 19 \cdot 10^{+2},$$

ahol: N = 519_{10}
s = negatív (–)
m = 5, 19 (egész és törtrészekkel)
r = 10
k = +2

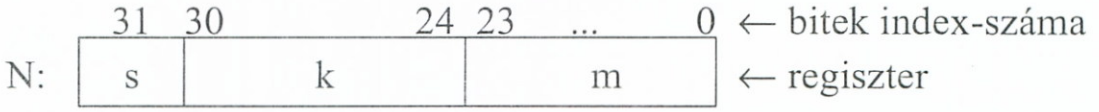
A digitális berendezéseknél (főként a számítógépeknél) célszerű a mantissza egész-, és tört részét összevonni a könnyebb kezelhetőség szempontjából. Példabeli esetünkben ennek érdekében a következő átalakítást végezhetjük:

$$-5{,}19 \cdot 10^{+2} = -0{,}519 \cdot 10^{+3}$$

Mint látható, ily módon az „egész" rész eltűnik (mindig „0" lesz) és a mantissza egy tizedes törtté alakul, melynél a tizedespont után közvetlenül egy nullától eltérő számjegy következik.

Az ilyen alakokat *normalizált* alakoknak nevezik és a számítástechnikában elterjedten alkalmazzák.

A digitális berendezéseknél természetesen *bináris* számokat használunk és ezeket *regiszterekben* helyezzük el. A lebegőpontos számokat kifejező (10.26) formulát ilyenkor közvetlenül nem alkalmazzuk. A problémát úgy oldjuk meg, hogy a számot tartalmazó regisztereknél az egyes szám-komponensek részére szektorokat alakítanak ki, melyek a számítások során, mindig ugyanazon elrendezésben maradnak a regiszterben. Egy ilyen szektoros regiszterbeli ábrázolást mutat példaként a 10.45. ábra:



itt: s – előjel : 1 bites (0 – pozitív, 1 – negatív)
m – mantissza : 24 bites
r – radix : r = 2 (kettes számrendszer)
k – karakterisztika : 7 bites

10-45. ábra *Egy lebegőpontos regiszter-formátum*

A 10.45. ábrán látható mantissza szektor 24 bites. Ez 25 bitesre tovább bővíthető azzal a meggondolással, hogy a mantissza bináris pont utáni *első* bitje eredendően, mindig „1"-es értékű. Ha ezt tudjuk, akkor ez a bit formálisan el is hagyható, így helyét egy további mantissza bittel feltölthetjük. Az ilyen ábrázolásmódot *implicit bit*es-nek nevezzük.

Végül oldjunk meg egy szemléltető példát a 10.45. ábra szerinti formátum felhasználásával. Ábrázoljuk az: $N = -1011_2$ számot.

$N = -1011_2 = -0{,}1011 \cdot 2^4$
$s = 1$ (negatív)
$m = 1011_2$
$k = 4 = 0000100_2$

A regiszter tartalom:

	s	k		m					
N:	1	000	0100	1011	0000	0000	0000	0000	0000
		8	4	B	0	0	0	0	0

A rendkívül hosszú bináris kifejezéseket nehéz kezelni, ezért ezeket gyakran tetrádokra (négyes csoportokra) bontjuk és az egyes tetrádokat hexadecimális számjegyenként értelmezve a lebegőpontos számot egy HEXA számmal fejezzük ki. Esetünkben a kiindulószám hexa formátumú lebegőpontos alakja:

$$N = 84B00000_{16} \text{ Hexa}$$

10.6.2. Bináris összeadók és kivonók

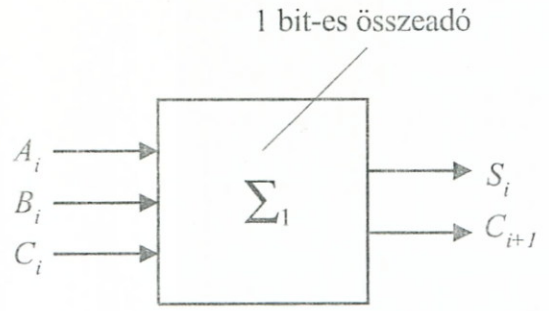
A következőkben az előző pontban összefoglaltakat hasznosító áramköröket tesszük vizsgálat tárgyává.

Egybites bináris összeadó

Vizsgáljuk meg a 10-46a. ábrán elvi vázlatával megadott *1 bit-es összeadó* felépítését. Amennyiben ezt modul elemnek tekintjük, fel kell tételeznünk, hogy az A_i, B_i *operandusok* mellett a bemenetre érkezik egy alacsonyabb helyértéken átvitelként keletkezett C_i *áthozat,* továbbá a kimeneten az S_i *összeg* mellett egy C_{i+1} következő fokozat felé irányuló *átvitel* is megjelenik.

A hálózatra jellemző két (S_i, C_{i+1}) kimeneti függvény logikai értéktáblázatát a 10-46b. ábrán adtuk meg a bináris összeadás 10-41. ábrán összefoglalt szabályainak felhasználásával. A táblázatból felírhatók az összeg és az átvitel függvényei a 10-46c. szerint, és egy realizációs logikai vázlat a 10-46d. ábrán került felrajzolásra.

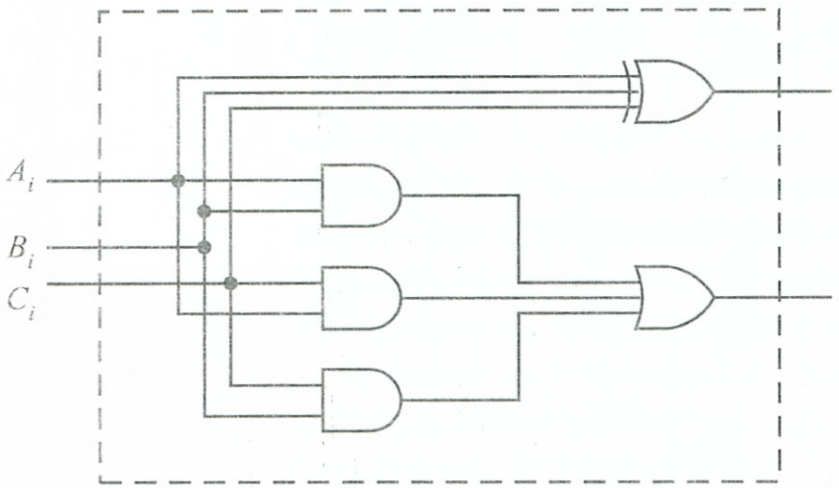
A Σ_1 1-bites összeadóból leszármaztatható az u.n. $\Sigma_{1/2}$ *fél-összeadó,* mely úgy keletkezik, hogy feltételezzük, hogy nincs C_i áthozat



a)

A_i	B_i	C_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

b)



$S_i = A_i \oplus B_i \oplus C_i$

$C_{i+1} = A_iB_i + A_iC_i + B_iC_i$

d)

c)

10–46. ábra *1 bit-es összeadó elve és működése*

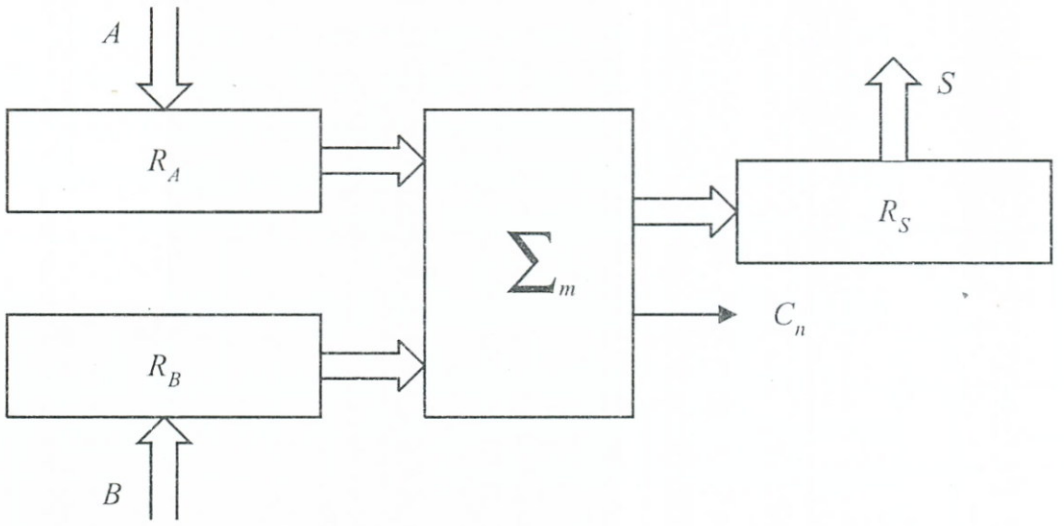
azaz ezt a bemenetet elhagyhatjuk a 10-46a. ábrából. Ezzel a módosítással a 10-46b,c,d ábrák is értelemszerűen egyszerűsödnek.

A *több-bites összeadás* megvalósítására soros- és párhuzamos elven felépített műveletvégzőket alkalmazunk.

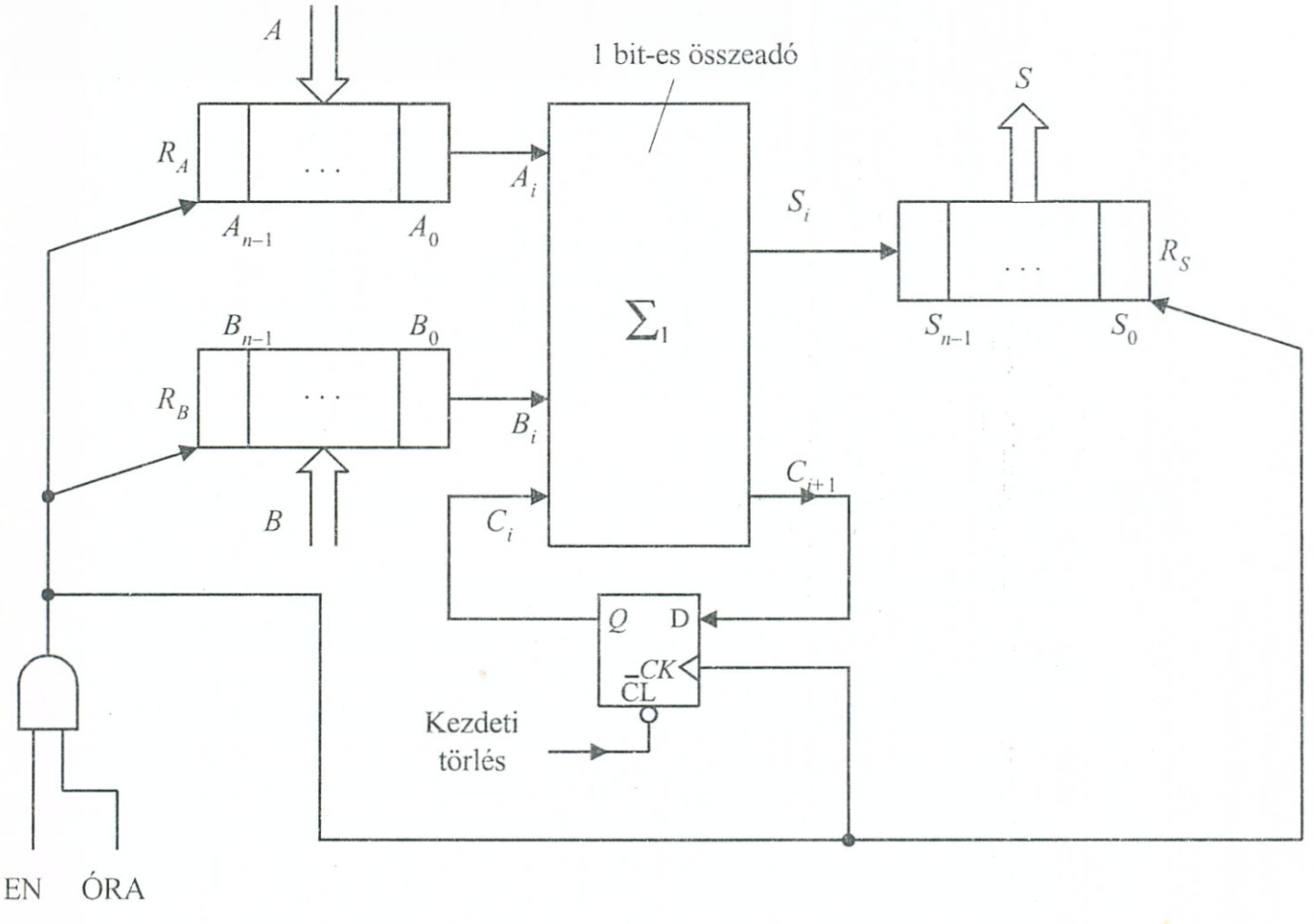
10.6.2.1. Soros elven működő összeadók

Soros összeadóknál a művelet az „A" és „B" kódszavak egyes szakaszaira *időben eltolva* történik. (10-47. ábra) Ezáltal a hosszú operandusok összeadásakor nem szükséges nagyobb kapacitású („n" bit-es összeadót) alkalmazni. Természetesen az összeadás soros elvégzése hosszabb időt vesz igénybe.

Az operandusokat 1 bitenként sorbaállító, 10-48. ábrán példaként felrajzolt kapcsolásnál az R_A, R_B regiszterekbe előzetesen betöltjük az *A, B* operandusokat, melyek *legkisebb helyértéke* van az összeadóhoz legközelebb.



10–47. ábra *Soros elven működő összeadó vázlata*



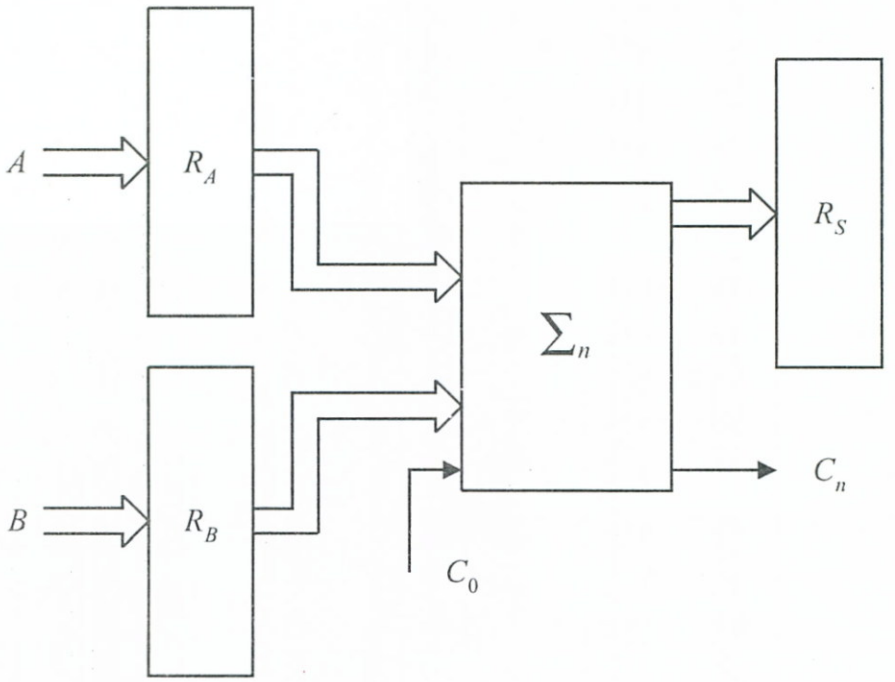
10–48. ábra *Soros elven működő n-bit-es összeadó*

Ezután az átvitelt előállító, visszacsatoló ág-beli *D*-tárolót töröljük a kezdeti $C_0 = 0$ átvitel biztosítása céljából. Az összeadás az $EN = 1$ engedélyezéssel indul, és minden órajel hatására a regiszterek jobbra lépnek, mialatt 1-1 bit összeadása történik meg. Az S_i összeg-bit az R_S

regiszterbe lép be. Az n-edik lépést követően az $EN = 0$ feltétellel a léptetést leállítjuk, az S összeg az R_S regiszterből kiolvasható lesz, míg a C_n átvitel a visszacsatoló hurokról vehető le. Az egy helyértékkel eltolt C_i áthozat itt időben szükséges 1 bit-nyi eltolódását a szinkron késleltetésként üzemelő D-tároló biztosítja.

10.6.2.2. Párhuzamos elvű összeadók

Párhuzamos összeadásnál a művelet az A, B operandusok valamennyi bitjére egyidejűleg történik, így a műveleti idő lényegesen megrövidülhet (10.49. ábra).



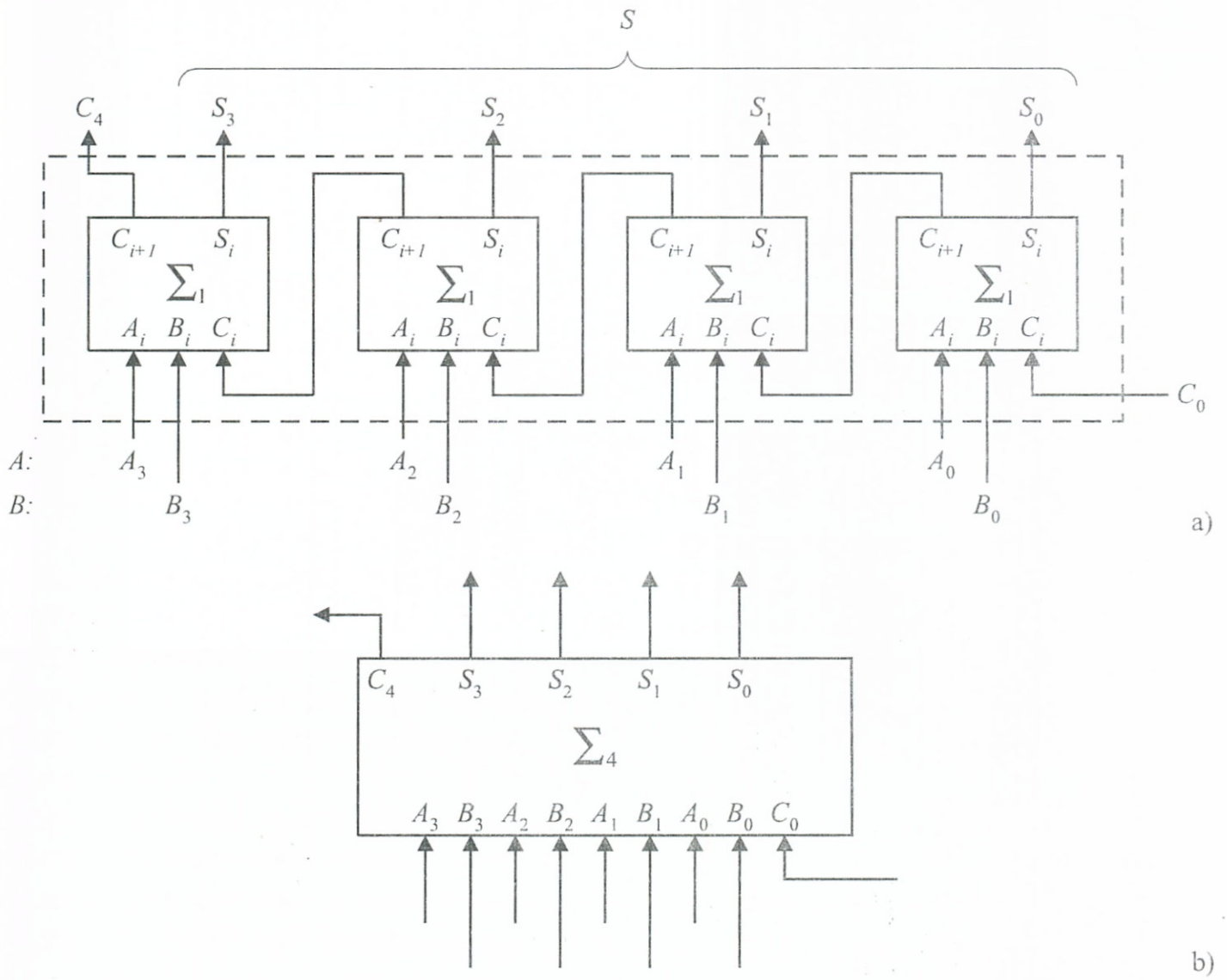
10–49. ábra *Párhuzamos elven működő összeadó vázlata*

a) RIPPLE–CARRY (sorosan terjedő) átvitelű összeadók

E változat működését a 10.50. ábrán bemutatott 4-bites párhuzamos összeadón tanulmányozhatjuk, melyet a 10.46. ábra 1 bites moduljaiból építettünk fel.

b) LOOK–AHEAD–CARRY elvű összeadók

A 10.50a. ábrán látható kapcsolás hátránya, hogy az átvitel „végigfutásához" számottevő terjedési idő kell, így az összeadás eredménye csak a tranziensek lezajlása után értékelhető. A soros átvitelből eredő hátrányt gyakran úgy kerülik meg, hogy az átvitelképzést kiemelik a modulokból és egy külön kombinációs hálózattal oldják



10–50. ábra *1 bit-es modulokból felépített 4 bit-es össszeadó*

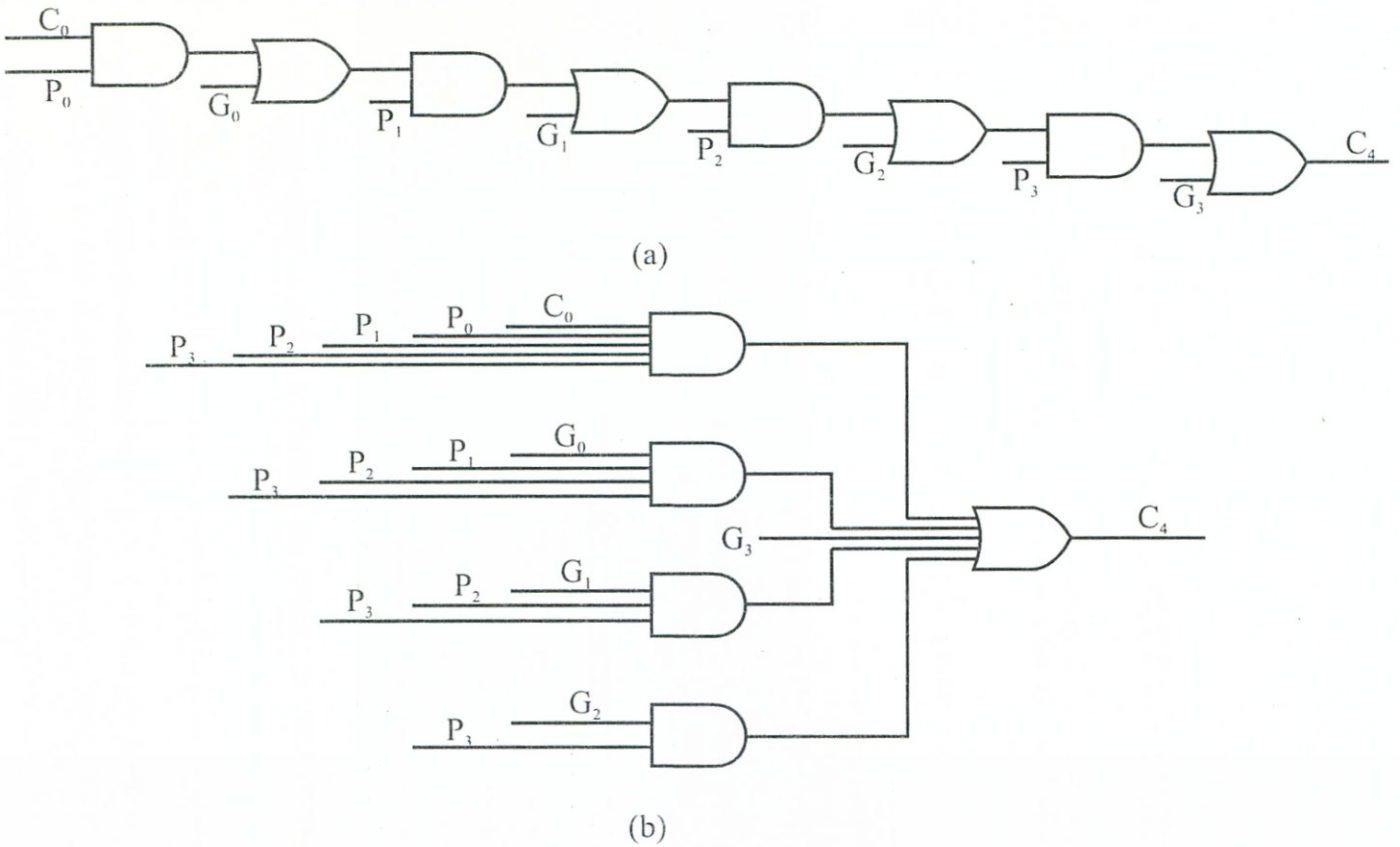
meg. Az ilyen elven működő összeadókat LOOK–AHEAD– CARRY-típusú műveletvégzőknek nevezik.

Ez utóbbi típus szemléltetése érdekében írjuk fel újra a 10.46c. ábra átvitelre vonatkozó összefüggését és alakítsuk át az alábbi módon:

$$C_{i+1} = A_iB_i + A_iC_i + B_iC_i$$

$$C_{i+1} = A_iB_i + (A_i + B_i)\,C_i = G_i + P_iC_i \qquad (10.27)$$

Itt a $G_1 = A_iB_i$ függvényt *átvitelkeletkezési,* a $P_i = A_i + B_i$ függvényt pedig *átvitelterjedési* feltételnek nevezik, ugyanis $G_i = 1$ esetben az adott helyi értéken keletkezik átvitel, $P_i = 1$ esetben pedig az alacsonyabb helyi értékről továbbterjed. $G_i + P_i = 0$ esetben egyáltalán nem lesz átvitel, még akkor sem, ha az alacsonyabb helyi értéken



10 - 51. ábra. *Ripple carry és look - ahead típusú átvitelképzés*

ilyen volt. A (10.27) képlet rekurzív alkalmazásával s némi ügyes átrendezéssel kapjuk az alábbi összefüggést:

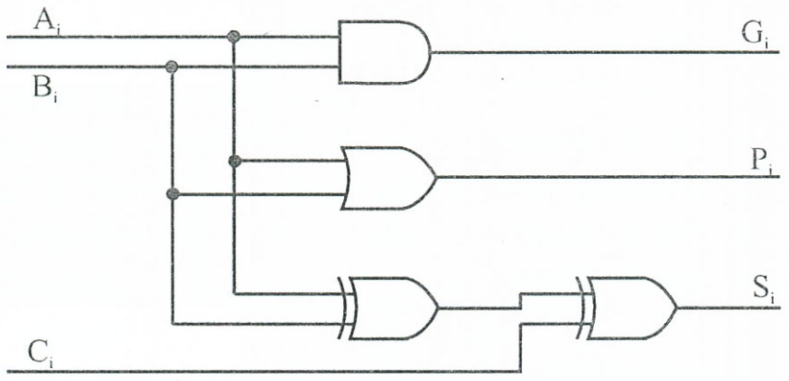
$$C_{i+1} = G_i + P_iG_{i-1} + P_iP_{i-1}G_{i-2} + ... + P_iP_{i-1} ... P_0C_0 \quad (10.28)$$

Ez a formula alkotja az alapját a LOOK–AHEAD–CARRYs megvalósításnak, ahol a párhuzamos összeadó egység átvitel kimenete igen gyorsan előállítja az átvitelt. A 10.51. ábra a. kapcsolása a *ripple carry* (terjedő) rendszerű soros (lassú), b. vázlata pedig a *look-ahead* típusú (gyors) átvitel előállítását szemlélteti 4 bites összeadandók esetében.

Az ilyen gyors átvitelképzésű megoldásoknál az egészösszeadót is úgy érdemes kialakítani, hogy az az összeg mellett ne az átvitelt, hanem közvetlenül az átvitelkeletkezési és -terjedési függvényeket állítsa elő például a 10.52. ábra szerinti elrendezésben.

Egy négybites párhuzamos összeadó kialakítása tehát 4 db, a 10.52. ábrán látható gyors átvitelképzésre alkalmas egészösszeadóból és a megfelelő, a 10.51b. ábra elve szerinti elrendezésű átvitelképző részáramkörökből történik.

Technológiai szempontból természetesen az ilyenfajta sokbites párhuzamos összeadó megvalósítása nehézségeket is hordoz magá-



10 - 52. ábra. *Átvitelkeletkezési-, és terjedési - függvények közvetlen előállítása*

ban. A bitszám növelésével ugyanis egyrészt egyre nő az átvitel előállításához szükséges logikai kapuk bemeneteinek a száma, s ami ennél még nagyobb probléma: egyre magasabb fan-outot kell az áramkörön belül megvalósítani. Ezért egy modul bitszáma általában erősen korlátozott, s a nagyon sokbites összeadók esetében még mindig jelentkezik a terjedési késleltetés problémája. (10.27) és (10.28) tanulmányozásával azonban rájöhetünk arra, hogy semmi akadálya az átvitelkeletkezés és -terjedés elvének tetszőleges méretű modulra történő általánosításának. A modul legfelső helyiértéken keletkező átvitel helyett (vagy mellett) tehát a több-bites összeadó modul rendelkezik egy modul–átvitel–keletkezés és modul–átvitel–terjedés kimenettel is (G és P), melyek felhasználásával a modulokból, a look–ahead–carry elvét felhasználva, további logikai kapukkal nagyobb egységek alakíthatók ki. Egy ilyen több-bites összeadó modul blokkja látható a 10.53. ábrán.

Az átvitellel kapcsolatos kimenetek közötti összefüggés a következő:

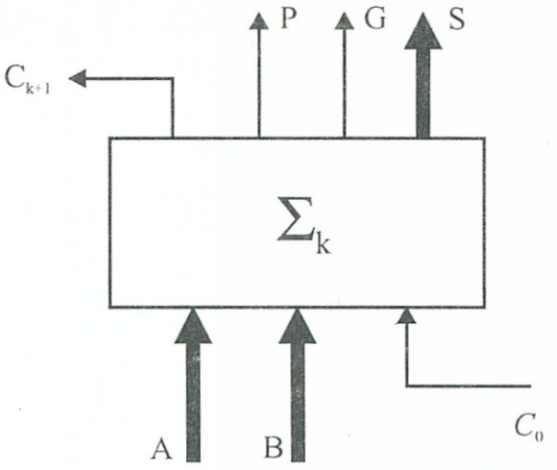
$$C_{k+1} = G + PC_0.$$

G és P belső előállítása pedig az alábbi függvények alapján történik:

$$G = G_k + G_{k-1}P_k + G_{k-2}P_kP_{k-1} + \ldots + G_0P_kP_{k-1}\ldots P_1$$

$$P = P_0P_1\ldots P_k \qquad (10.29)$$

Megemlítjük még, hogy léteznek olyan integrált áramkörök is, amelyek a gyors átvitelképzést a fokozatok közötti szinteken is megkönnyítik.



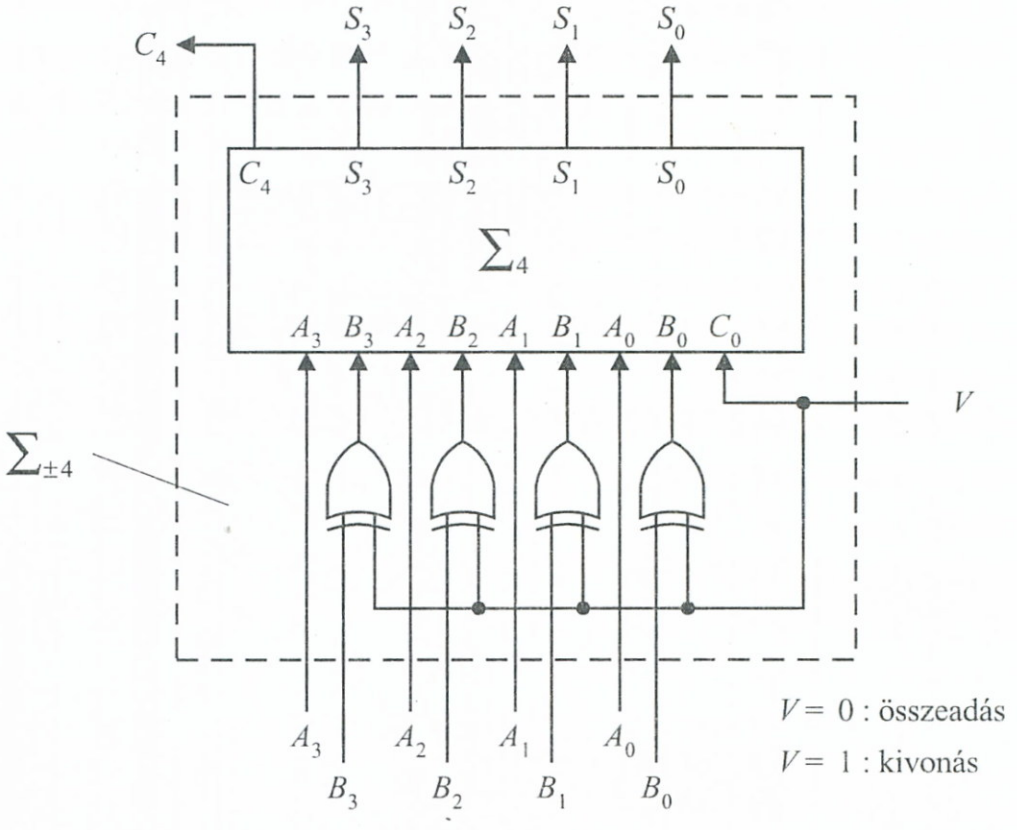
10 - 53. ábra. *Look - ahead elvű, több - bites összeadó modul*

10.6.2.3. Párhuzamos elvű kivonók

Már az előző pontban bemutattuk, hogy a kivonást miként lehet – komplemens képzéssel – összeadásra visszavezetni.

Miután a komplemens előállítása viszonylag egyszerűen megoldható a következő lépésekben:

– először: operandus negáltjának előállítása;
– másodszor: +1 hozzáadása,



10–54. ábra *4 bit-es összeadó-kivonó kialakítása összeadóból*

ezért az összeadó áramkört a fentieket elvégző „előtét" beiktatásával kivonó áramkörré alakíthatjuk át.

Például alakítsuk át a 10-50. ábra összeadóját kivonó áramköri kiegészítést is tartalmazó *összeadó-kivonó* modullá. A megoldás a 10-54. ábrán látható. A B operandus bemenetei elé egy antivalencia kaput helyeztünk el, melyet a C_0-ra is rákötött V vezérlő jel kapcsol az alábbiak szerint.

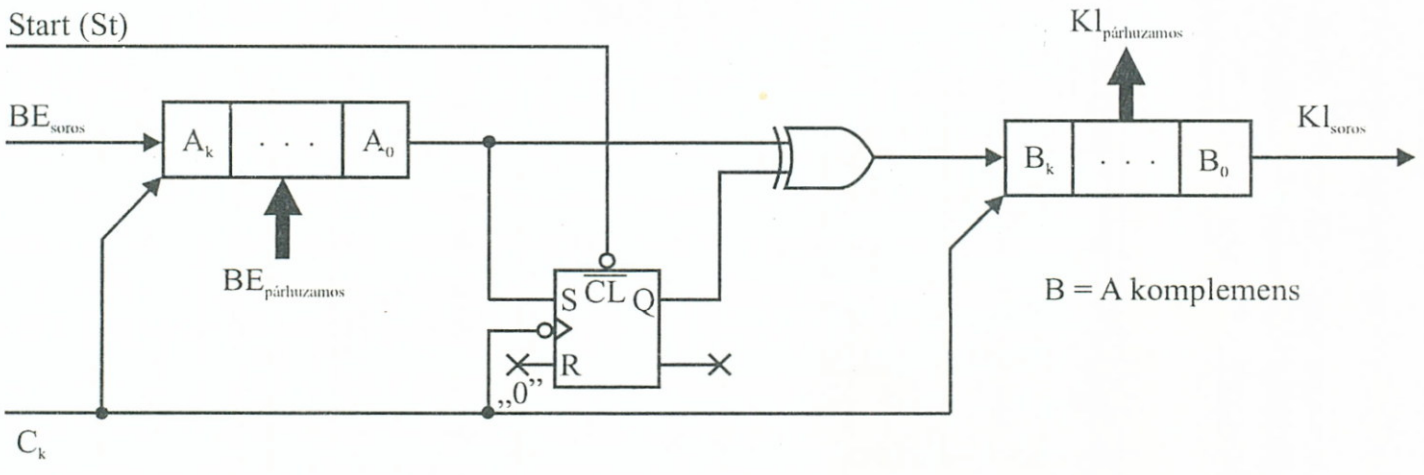
$V = 0$ esetén: az antivalencia kapu *ismételő* elemként működik, és a B_i-ket változatlanul átengedi magán, továbbá a $V = C_0 = 0$ értékű átvitel is biztosítva van, így az eredő áramkör *összeadóként* működhet.

$V = 1$ esetén: az antivalencia *inverterként* üzemel, és előállítja B operandus negáltját, majd ehhez a $V = C_0 = 1$ mint áthozat révén +1 hozzáadódik, így lényegében *„B" komplemense* keletkezik a bemeneten. Ezzel elvégezve az összeadást – mint láttuk – valójában kivonást végzünk.

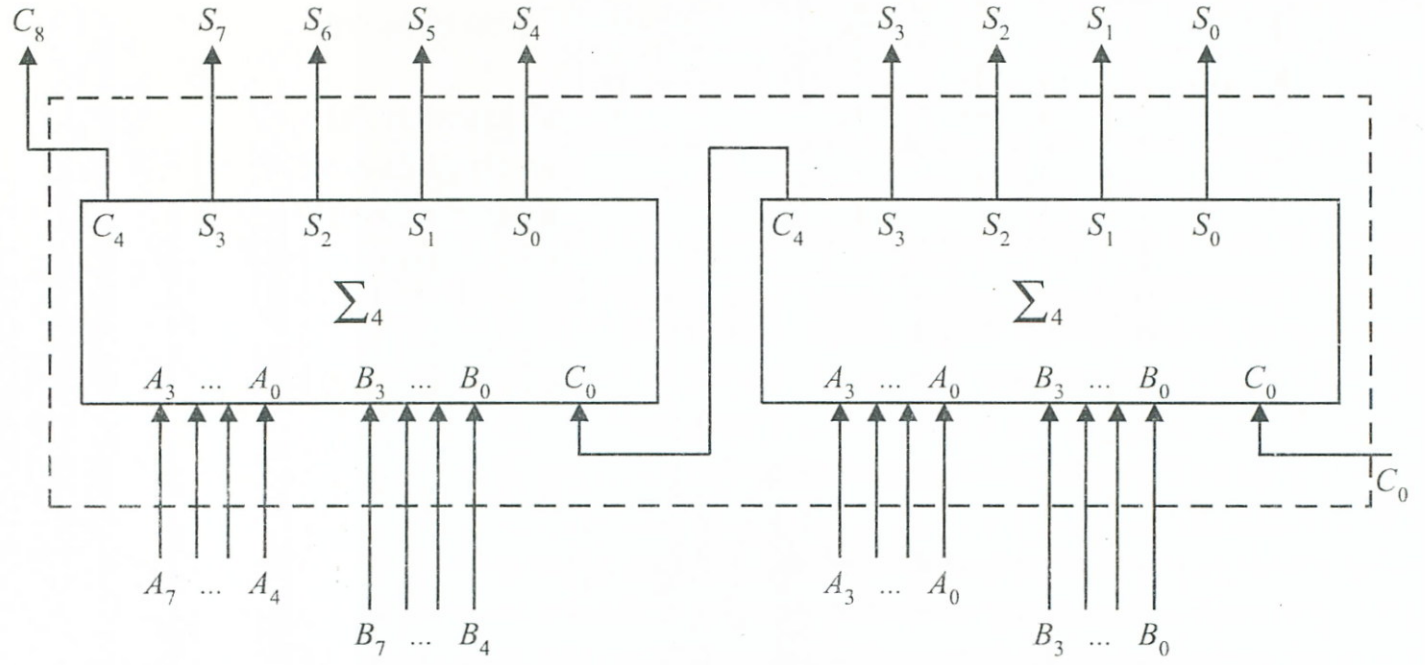
10.6.2.4. Soros elvű kivonók

A kivonásnál itt is a komplemenset használhatjuk fel, de ezt most „sorosan" kell előállítanunk. Itt soros vonalon érkeznek a bitek a legalsó helyiértéktől kezdve és mindaddig változatlanul lépnek ki, amíg az első 1-es meg nem jelent. A következő bittől kezdve minden negálódik. Egy lehetséges kialakítás látható a 10-55. ábrán.

Az indítás az St bemenetre adott törlő impulzussal valósul meg, amely a master–slave S-R tárolót alaphelyzetbe hozza. Ezután indulhat a bitek beléptetése, s a tároló mindaddig nem változtatja meg az állapotát, amíg egy 1-es nem jelentkezik a bemenetén. Ekkor az órajel lefutó élére átbillen 1-be és itt is marad a működés további idejére. Így az első 1-es még változatlanul halad át az antivalencia-kapun, az ezt követő többi jegy azonban negálódik.



10 - 55. ábra. *Soros komplemensképző áramkör*



10–56. ábra *8 bit-es összeadó kialakítása bővítéssel*

A 10.55. ábrabeli soros komplemensképzőt például a 10.48. ábrán bemutatott soros összeadóval kombinálva már előállítható a soros kivonó áramkör.

10.6.2.5. Modulok összekapcsolása

A műveletvégző modulokat úgy alakítják ki a gyártó cégek, hogy magasabb bit-számra is bővíthetők legyenek. Egy rendkívül egyszerű példaként a 10-56. ábrán felrajzoltunk egy bővítéssel létrehozott 8 bites összeadót, melyet a 10-50. ábra áramkörtípusából építettünk fel.

Bonyolultabb modulokkal történő bővítésnél már több csatlakozási pontot kell összekapcsolni, ezért ilyenkor a katalógusok előírásait kell követnünk.

10.6.3. BCD számábrázolás és összeadás

!

A BCD (Binary Coded Decimal) kóddal már a 10.5. ábrán találkoztunk. Ennek alapján megállapítható volt, hogy decimális számoknak bináris elemekkel történő kifejezésekor valamely decimális helyérték (dekád) ábrázolásához négy bináris helyértékre (egy „tetrád"-ra) van szükség.

A BCD számok összeadásakor többféle úton is célhoz juthatunk:

a) Ha bináris összeadó áll rendelkezésünkre, akkor először BCD–BIN átkódolást végzünk, ezután az összeadást kettes szám-

rendszerben végezzük el, majd egy BIN–BCD visszakódolás után kapjuk a decimális eredményt.

b) Közvetlenül BCD összeadót alkalmazunk, mely rendszerint dekádos modulokból épül fel és annyi modult tartalmaz, ahány helyértékű decimális számokkal kell az összeadást elvégeznünk.

A két változat közötti választásnál az átkódolási késleltetések, a BCD összeadás sebessége, és a környező hardver-rendszerhez való illeszkedés, amiket elsősorban mérlegelni kell.

A BCD számokkal végzett műveleteknél gyakran problémát jelent az, hogy a decimális számjegyek csak tíz-féle (0, 1, ... 9) kombinációt értelmeznek a 4 bittel kifejezhető 16-féle (Hexa) lehetőség közül. A műveletek (pl. összeadás) során előadódhat viszont olyan eset, hogy a részeredmények „átlógnak" a nem értelmezett mezőbe. Valójában itt az átvitelképzés problémái jelentkeznek, melyek megoldására korrekciós faktorok (leggyakrabban a 6-os korrekció) bevezetésére van szükség.

Fentiek megvilágítására vizsgáljuk meg a 10.57. ábrában táblázatosan megoldott BCD összeadási példát:

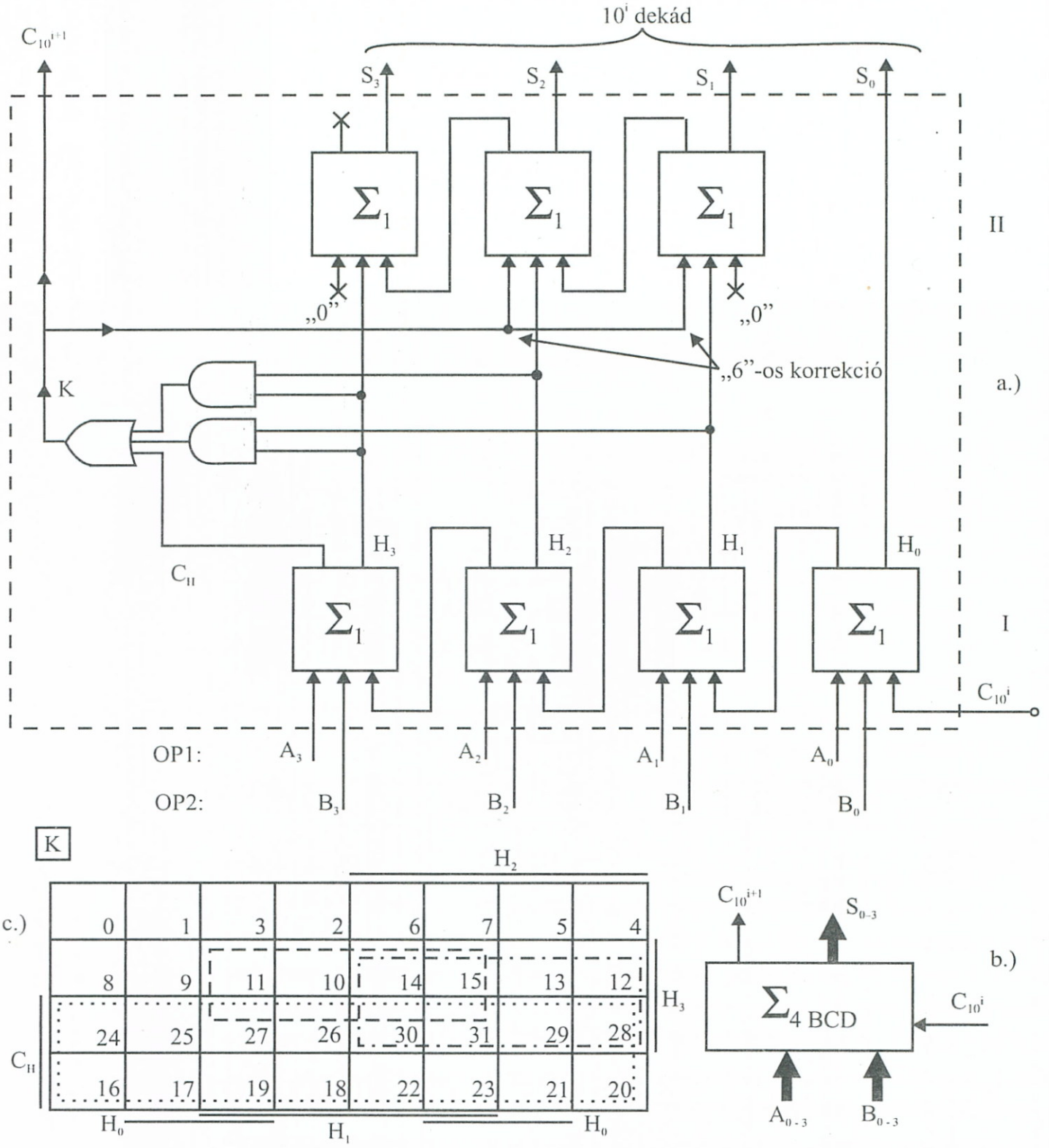
	Decimális	BCD			
		10^3	10^2	10^1	10^0
OP1	$5\ 8\ 2\ 8_{10}$	0101	1000	0010	1000_{BCD}
+ OP2	$3\ 9\ 1\ 5_{10}$	0011	1001	0001	0101_{BCD}
Hamis összeg	9 1 3 D ⇐	1001	0001 (átvitel) ⇑	0011	1101 (Hexa) ⇑
	↑ ↑				
+ 6-os korrekció:	0 6 0 6	0000	0110	0000	0110
Helyes összeg:	$9\ 7\ 4\ 3_{10}$	1001	0111	0100	0011_{BCD}

10–57. ábra *BCD összeadás 6-os korrekcióval*

A BCD rovatban elhelyezkedő négy decimális helyértékű, $4 \times 4 = 16$ bites operandusokat bináris helyértékenként összeadva, már az első tetrádnál HEXA-szám adódik, azaz „kilógunk" a decimális szimbólum-halmazból, mivel 9 fölé kerültünk. A 10^2 helyérték összeadásánál pedig ($8 + 9 = 17$) a Hexa (4-bites) szimbólumkészletből is „kilógunk" és átvitelt kellett képeznünk a 10^3 helyérték irányában. A 9 és 15 határértékek túllépéséből eredő átviteli gondok miatt az ábrán nyíllal megjelölt, 10^0 és 10^2, példabeli problémás helyeken 6-os kor-

rekciót kellett alkalmaznunk, mellyel a „Hamis összeg" eltéréseit kiküszöbölhettük.

Fenti 6-os korrekciós elv hardver formában történő alkalmazását láthatjuk a 10.58a. ábra *egydekádos* BCD összeadóján. A I. 4-bites BIN összeadó megegyezik a 10.50. ábrán már bemutatottal és kimenetén itt a „hamis összeg" jelenik meg, mivel az OP1 + OP2 össze-



10 - 58. ábra. *BCD összeadó dekád 6-os korrekcióval*

adást végzi el. Az 1-dekádos BCD összeadó elvi vázlatát a 10.58b. ábrán rajzoltuk fel. Mint a 10.57. ábra példájánál tapasztaltuk, 6-os korrekcióra – végső fokon –, akkor van szükség, ha a 9-et túlléptük a dekád-összeggel. A korrekciós áramkör logikai vázlatának felrajzolása érdekében célszerű előállítani a 10.58c. ábrán látható 5-változós minterm táblát, melynek független változói azok a kimenetek, melyek a 6-os korrekció előállításában szerepet kapnak. A tábla azon helyein kell korrekciót alkalmazni, melyek minterm-indexeik túllépik a 9-et. Ezeket tömbösítéssel minimalizálva kapjuk a K-korrekciós függvényt:

$$K = C_H + H_3 \cdot H_1 + H_3 \cdot H_2$$

A K-kimenetet kötjük be a I.-hez hasonló felépítésű II. összeadó 0110_2 (6_{10})-nak megfelelő bemeneteire. Így a felső összeadó a szükségessé váló korrekciós összeadásokat el tudja végezni.

10.6.4. Bináris szorzás

Ha egy léptetőregiszterben tárolt bináris számot *balra* léptetünk, akkor a szám minden számjegye eggyel magasabb bináris helyérték pozícióba kerül. Ezt, a kiinduló szám szempontjából úgy tekinthetjük, hogy azt kettővel (a radix-szal) megszoroztuk. A léptetőregiszterrel tehát a bináris szorzás „legegyszerűbb" esetét valósíthatjuk meg, ún. *soros* szorzással.

10.6.4.1. Szorzás előjel abszolútértékes ábrázolás esetén

Előjel-abszolútértékes szorzásnál a szorzat előjelének és abszolút értékének előállítása külön-külön feladatot igényel.

Előjel-számolás esetén, azt az „iskolás" szabályt alkalmazhatjuk, mely szerint a szorzat előjele:

– sp: pozitív – ha a szorzandók előjele (s_1, s_2) egyforma,
– sp: negatív – ha a szorzandók előjele (s_1, s_2) eltérő.

Az így megfogalmazott feladatot *logikai* értelmezésben is átgondolva egy *antivalencia* függvényre ismerhetünk, amennyiben a negatív jelet (a korábbiakkal egyezően) „1"-gyel jelöljük.

$$s_p = s_1 \oplus s_2 \qquad (10.30)$$

Abszolútérték-kiszámításnál ugyancsak a „klasszikus, iskolás" szabályt alkalmazhatjuk, mely szerint a szorzatot úgy kapjuk, ha: „a szorzandó értékét a szorzó minden egyes bitjével külön-külön meg-

szorozva, az így nyert részszorzatokat a szorzó bit helyértékének megfelelő helyértékre eltolva összeadjuk".

Szemléltetésül vizsgáljuk meg a 10.59a. ábrában megoldott „iskolás" szorzást, melynél a műveletet a szorzó legkisebb helyértékén kezdtük. A példa menetéből látható, hogy rész-szorzatonként a „0"-val történő szorzás esetén csupa „0", míg „1"-gyel történő szorzás esetén a szorzandó „megismétlése" az eredmény, a kívánt helyérték-eltolódás figyelembevételével.

A = szorzandó: $+7_{10} = 0\,|\,111_2$

B = szorzó: $-6_{10} = 1\,|\,110_2$

előjel (s_i)

abszolutérték

Előjel kiszámítása:

$s_p = s_A \oplus s_B = 0 \oplus 1 = 1$

s_p = negatív

Abszolutérték kiszámítása:

	2^2	2^1	2^0	2^2	2^1	2^0	
	A = 1	1	1	1	1	0	=B
		0	0	0			
+		0	0	0			
		0	0	0			
+		1	1	1			
		1	1	1	0		
+	1	1	1				
	1	0	1	0	1	0	=|P|=szorzat
	2^5	2^4	2^3	2^2	2^1	2^0	abs. é.

Eredmény:

$$P = \text{szorzat} = s_p, |P| = -101010_2 = 42_{10}$$

10–59. ábra *Bináris, klasszikus iskolás szorzás*

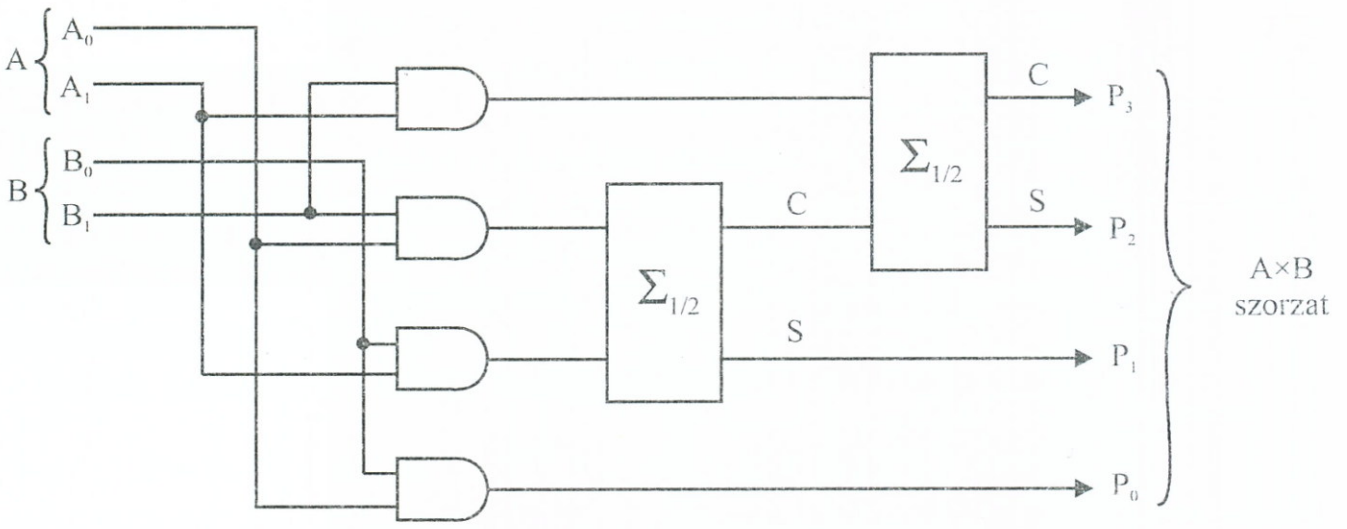
A példában bemutatott szorzás áramköri realizálására többféle megoldás is kínálkozik:

a) PÁRHUZAMOS STRUKTÚRA szerinti realizálással nagyon gyors szorzóáramkörök építhetők, mivel a részösszegek összeadói már néhány kapuáramköri késleltetést követően elérhetők.

A megoldásnak persze hátránya is van: igen sok kapuáramkört és fél-, illetve egész összeadót igényel, tehát meglehetősen költséges.

Egy nagyon egyszerű eset, a 2×2 bites párhuzamos szorzó kapcsolási rajza látható a 10.60. ábrán.

A rajzból megérthető a szorzó működési elve, de a kevés összeadandó részszorzat miatt itt még igen egyszerű a megoldás, valójában egyetlen egész-összeadóra sincs szükség. Hasonlítsuk ezt az egyszerű kapcsolást össze a 4×4 bites szorzó vázlatával (10.61. ábra).



10 - 60. ábra. *2× 2-bites párhuzamos szorzó*

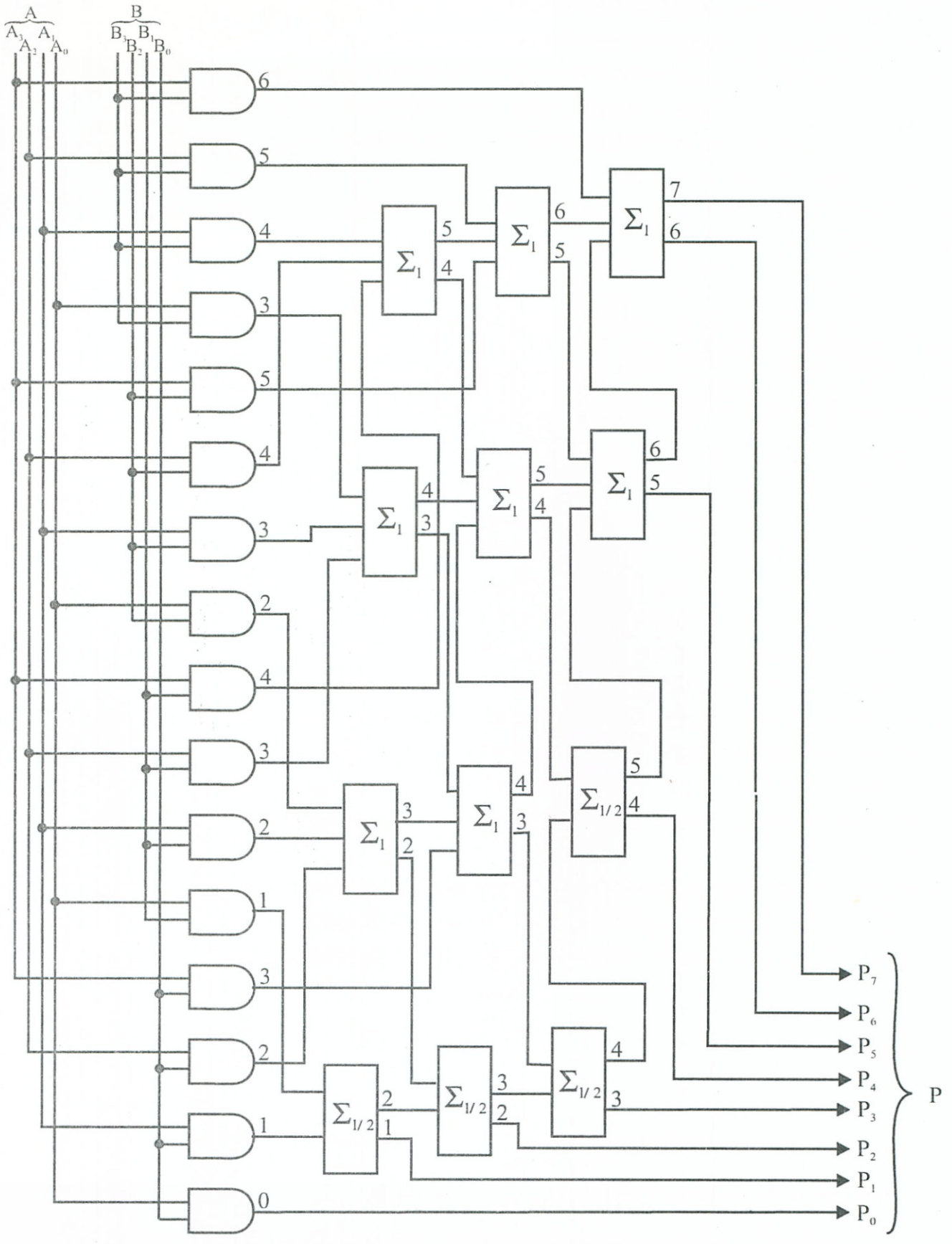
A könnyebb eligazodás érdekében minden egyes ÉS-kapu, fél- és egészösszeadó (S- és C-) kimenet mellett feltüntettük a helyiérték sorszámát, amelyen az adott bit képződik. Így nyomon követhető, hogyan adódnak össze az egyes részszorzatok bitjei, illetve az átvitelbitek. Az ábra bonyolultsága miatt az egyes részletszorzatokat képző áramkörcsoportokat nem helyeztük el tagoltan, az egyes helyiértékek éppen a kimenetek melletti indexek révén követhetők nyomon.

Ez az elrendezés már nagyságrenddel bonyolultabb, s gondoljuk meg, hogy a gyakorlatban általában 8×8, 16×16, 32×32, vagy 64×64 méretű szorzásokat kell megoldani. A szorzandók méretével a szorzó terjedelme és költsége exponenciálisan nő, tehát egy bizonyos méret felett ez az elv – még VLSI realizáció esetén is – nehezen kezelhetővé válik.

b) SOROS STRUKTÚRA szerinti realizálásnál az egyes helyértékeknél adódó műveleteket időben egymás után sorba állítva végezzük el. Emiatt a fázisonként kapott eredményeket – a későbbi felhasználás érdekében – regiszterekben tárolnunk kell. Az időbeli sorba állítás miatt a szorzás végrehajtásának ideje – természetesen – meghosszabbodik, viszont a realizáló hálózat felépítése lényegesen egyszerűbbé válhat.

Vizsgáljuk meg a továbbiakban a soros működés kapcsán felvetődő kérdéseket, mégegyszer átgondolva a 10.59. ábra példájának tapasztalatait is.

A példából leolvasható, hogy a szorzat bitjeinek száma (hossza) nagyobb, mint az operandusok hossza (szélső esetben azok duplája). A soros műveletvégzés során tárolni kell majd az operandusokat, a részösszegeket és magát a szorzatot is. A rész-összeadásokat minden



10 - 61. ábra. *4×4 bites párhuzamos szorzó*

esetben a helyértékenként balra léptetett szorzandóval kell fázisonként elvégezni és ehhez az összeadó is dupla hosszúságú kellene, hogy legyen.

A felsorolásban szereplő kívánságlista enyhíthető a szükséges regiszterek számának csökkentésével és – ami különösen lényeges – a dupla hosszúságú összeadó szimpla hosszúságúra csökkentésével, ha a 10.59. ábra „iskolás" algoritmusát, néhány matematikai átalakítás felhasználásával módosítjuk.

Felírva a szorzatot a szorzó komponenseivel:

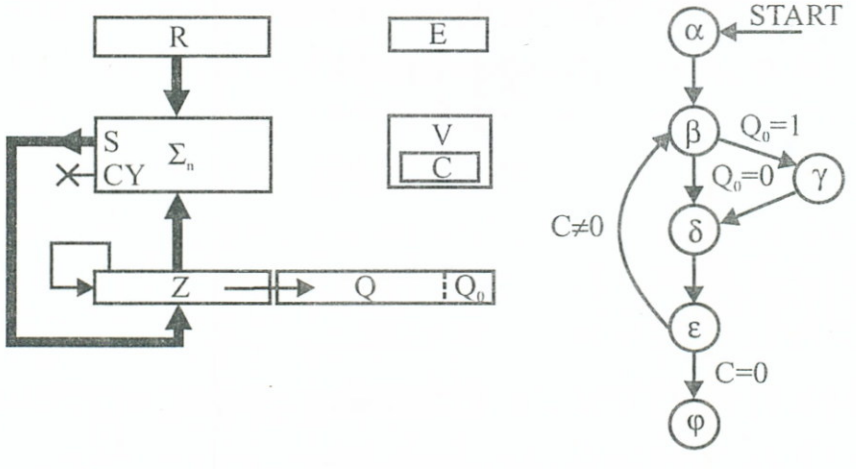
$$P = A \times B = A \times (b_{n-1} \cdot 2^{n-1} + \ldots + b_1 \cdot 2^1 + b_0 \cdot 2^0) \qquad (10.31)$$

Végezzünk el egy rekurzív átalakítást, mely szerint:

$$A \times B = \{\ldots[(2^n \cdot A \cdot b_0 \cdot 2^{-1} + 2^n \cdot A \cdot b_1) \cdot 2^{-1} + 2^n \cdot A \cdot b_2] \cdot 2^{-1} + \\ + \ldots + 2^n \cdot A \cdot b_{n-1}\} \cdot 2^{-1} \qquad (10.32)$$

Ha a (10.32) rekurzív formulát értelmezzük, akkor megállapítható, hogy a mindenütt szereplő $2^n \cdot A$ szorzat – ami n-szeres *balra léptetés*nek felel meg – azt jelenti, hogy a szorzandó azonos helyértékű bitjétől a P szorzat legkisebb helyértékű bitje n-bittel jobbra fog elhelyezkedni. A szorzást – hasonlóan a 10.59. ábra példájához – a B szorzó legalacsonyabb (b_0) bitjével kezdve, minden részletösszeadás után a 2^{-1} szorzó miatt *eggyel jobbra* kell majd léptetnünk és miután a szóhossz itt n, ezért az algoritmus n-léptetés után fog befejeződni.

A (10.32) rekurzív formula által értelmezett algoritmus végrehajtására használható fel a 10.62a. ábrán látható blokksémával jellemzett hálózat, melynek működését a 10.62b. ábra állapotgráfja szerint dolgozó, valamennyi regiszterrel és az összeadóval is kapcsolatot tartó V vezérlőegység irányítja.



(a)

(b)

10 - 62. ábra. *Bináris szorzó 1-bit figyelésű algoritmussal*

A b. ábrabeli állapotgráf a szorzási algoritmus időbeli végrehajtásának lépéseit tartalmazza és alapjául szolgálhat a V vezérlőegység, mint sorrendi hálózat megtervezéséhez is.

Vizsgáljuk meg a rekurzív formula alapján dolgozó algoritmus lépéseit.

α állapot: Indulás előtt fel kell tölteni az egyes regisztereket:

SZORZANDÓ:	A → R
SZORZÓ:	B → Q
TÖRLÉS:	0 → Z
CIKLUSSZÁM:	n → C

C egy preszetelhető *hátra-számláló,* melybe beírjuk a B szorzó bit hosszát (n–t) és a működés során később, minden egyes helyértékenkénti szorzás befejezése után majd 1-gyel dekrementálni fogjuk. A szorzási folyamat befejezését az fogja jelezni, hogy C = 0 lesz, azaz az utolsó részösszeg képzési ciklus is befejeződött.

A START indító parancs után a szorzási algoritmus beindul.

β állapot: A működés a *β* állapotba tevődik át, itt a berendezés megvizsgálja, hogy a Q_0 bit értéke milyen értékű:

$Q_0 = 0$-nál: nincs részösszeg-képzés, ezért a hálózat átlép *δ*-ba.

γ állapot:

$Q_1 = 1$-nél: R + S összeadás történik, majd továbblépés *δ*-ba.

δ állapot: Itt a Z, Q regiszterpár (a 2^{-1} szorzó miatt) 1 bittel *jobbra* léptetődik együttesen, és ezzel az aktuális b_i szorzóbittel kapcsolatos részszorzat teendői befejeződnek.

Itt dekrementáljuk a C hátraszámlálót is.

ε állapot: Megvizsgáljuk, hogy nem értünk-e el a szorzási folyamat végére, azaz nem végeztük-e el már az összes szorzóbittel történő szorzást. Ezt az előzők értelmében C állapota mutatja.

$C \neq 0$-nál: még *újabb* részműveleti ciklust kell lefolytatni, ezért visszaugrás *β*-ba, és a *β*...*ε* ciklus ismétlődik.

$C = 0$-nál: ez volt az *utolsó* ciklus, a folyamat leáll *β*-ben.

φ állapot: Ebben az állapotban az egyes regiszterek – melyek tartalma időközben folyamatosan változott – a szorzás végeredményét tartalmazzák:

Z, Q = SZORZAT (A × B = P)
R = SZORZANDÓ (A)
C = 0

Mint látható, az algoritmus során elvégzett teendőknél fontos szerepe van a Qo bitnek, mely ciklusonként a b_i aktuális szorzóbitet tartalmazza. Ezért a bemutatott algoritmust: *1-bit figyelésű szorzási algoritmusnak* is nevezik.

Miután az operandusok itt is előjel-abszolútértékes ábrázolásnak, ezért végezetül a 10.59. ábrabeli példához hasonló módon ez E előjelképző blokkal még a szorzat előjelét is meg kell határozni.

A bemutatott *soros szorzóberendezés* lassúbb, mint a 10.60. ábra elvén felépülő párhuzamos megoldás, mivel a:

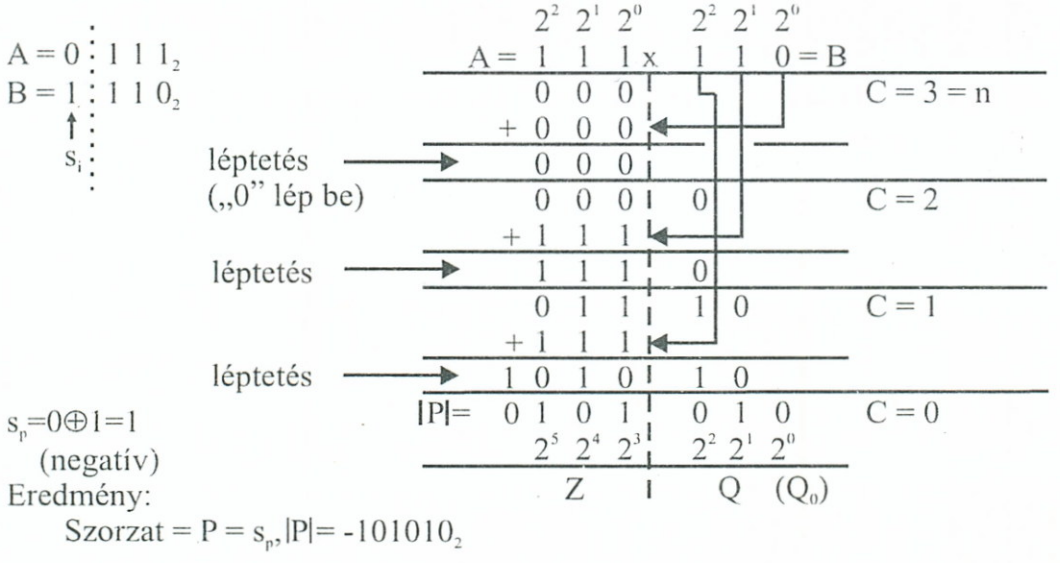
$$\text{hasznos szorzási idő} \approx n \times t_{CK},$$

ahol: n = az operandusok bit-száma
t_{CK} = 1 részműveleti ciklus időtartama.

A szorzóművet vezérlő V vezérlőegység, a megadott állapotgráfból kiindulva akár fázisregiszteres, akár mikroprogramozott változatban is kialakítható a sorrendi hálózatokkal foglalkozó 8. fejezet 8.4. pontjában megismert módszerek segítségével.

A realizált hálózat (elég nagy „n" esetén) lényegesen egyszerűbb lehet, mint a 10.61. ábrabeli változat.

Végezetül, a 10.59. ábrabeli példában szereplő számpéldát odjuk meg még egyszer a most megismert 1-bit figyelésű algoritmus felhasználásával. A megoldás egyes lépéseit a 10.63. ábrán követhetjük.



10 - 63. ábra. *Bináris, 1-bit figyelésű szorzási példa*

10.6.4.2. Szorzás komplemens kódban

A korábban már bevezetett „komplemenses" számábrázolásnál – mint tapasztaltuk – nem kellett az előjelekkel külön foglalkoznunk. Ezért a komplemens-kódban ábrázolt számok szorzására több eljárást is kidolgoztak, melyek közül az elterjedt BOOTH-féle szorzási algoritmust mutatjuk be az alábbiakban.

Induljunk ki ismét a szorzat (10.31)-ben már alkalmazott komponensekkel történő ábrázolásából, de itt ügyelnünk kell arra, hogy a komplemens kódú jellemzésnél a legmagasabb bit negatív súlyozású (10.24) értelmében. Emiatt kiinduló-formulánk (10.33) szerint fog alakulni:

$$A \times B = A \times (-b_{n-1} \cdot 2^{n-1} + \ldots + b_1 \cdot 2^1 + b_0 \cdot 2^0) \qquad (10.33)$$

Használjuk fel az alábbi lehetséges formális átalakítást:

$$b_i \cdot 2^i = b_i \cdot 2^{i+1} - b_i \cdot 2^i, \qquad (10.34)$$

melyet (10.33)-ba behelyettesítve (10.35)-öt kapjuk:

$$A \times B = A \times [(b_{n-2} - b_{n-1}) \cdot 2^{n-1} + \ldots + (b_0 - b_1) \cdot 2^1 + (b_{-1} - b_0) \cdot 2^0] \qquad (10.35)$$

Ezután bevezetve az alábbi jelölést:

$$d_i = b_{i-1} - b_i,$$

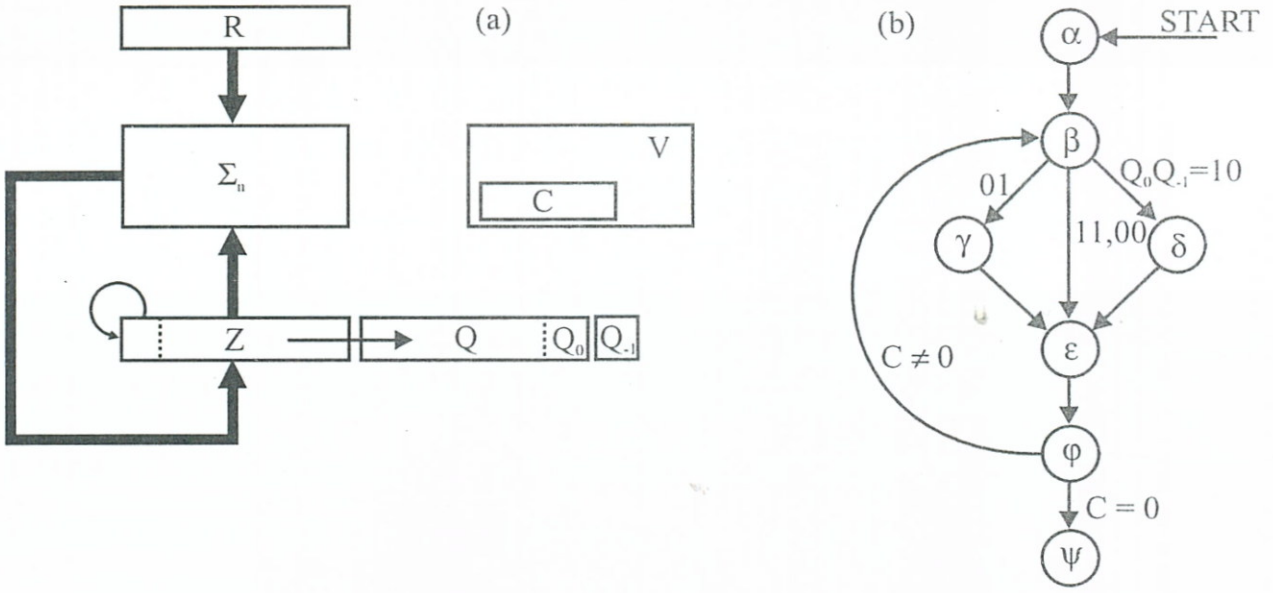
majd behelyettesítve (10.35)-be és némileg átrendezve a következő rekurzív formulához (10.36) jutunk:

$$A \times B \{\ldots [(2^n \cdot A \cdot d_0 \cdot 2^{-1} + 2^n \cdot A \cdot d_1) \cdot 2^{-1} + 2^n \cdot A \cdot d_2] \cdot 2^{-1} + \ldots + 2^n \cdot A \cdot d_{n-1}\} \cdot 2^{-1} \qquad (10.36)$$

A formulák levezetése során keletkezett 2^{-1} súlyozású b_{-1} szám értéke egész számok esetén $b_{-1} = 0$. A d_i bitek –1, +1 és 0 értékeket vehetnek fel és emiatt a részletszorzatokat néha kivonjuk vagy hozzáadjuk az előző részösszeghez, vagy 0-esetén nem kell összevonási műveleteket végeznünk a (10.37) táblázatban összefoglaltaknak megfelelően:

b_i	b_{i-1}	d_i	Művelet
0	0	0	NINCS
0	1	1	ÖSSZEADÁS
1	0	–1	KIVONÁS
1	1	0	NINCS

(10.37)



10 - 64. ábra. *Bináris szorzó, komplemenses, 2-bit figyelésű algoritmussal*

A (10.36) rekurzív formulából leolvashatók a szorzási algoritmussal kapcsolatos teendők is, a (10.32)-nél tapasztaltakhoz hasonlóan: A szorzandót itt is n-szer *balra* kell léptetni ($2^n \cdot A$) és a szorzást a b_0-lal kell kezdeni, továbbá az algoritmus n lépésből áll. A lényeges eltérés, hogy itt *két bitet* (b_i, b_{i-1}) kell figyelni és a szükséges műveletek a (10.37) táblázat szerint alakulnak, melyek után *mindig léptetés* következik. Mivel erre az algoritmusra jellemző, hogy két bitet kell figyelni, ezért az irodalomban „két bit figyelésű algoritmus"-nak is nevezik.

Az elmondottak végrehajtására használható fel a 10.64a. ábra blokksémájával jellemzett hálózat, melynek működését a V vezérlőegység irányítja a 10.64b. ábra gráfjával értelmezett és az alábbiakban részletezett algoritmus szerint:

α állapot: Indulás előtt fel kell tölteni az egyes regisztereket:

SZORZANDÓ : $A \rightarrow R$
SZORZÓ : $B \rightarrow Q$ (Q_0 a b_0-t tárolja)
TÖRLÉS : $0 \rightarrow Q_{-1}$ (Q_{-1} a b_{-1}-t tárolja)
TÖRLÉS : $0 \rightarrow Z$
CIKLUSSZÁM : $n \rightarrow C$

A START indító-parancs után a szorzási algoritmus beindul.

β állapot: Ebben az állapotban Q_0Q_{-1} két bit vizsgálata történik a (10.37)-nek megfelelően.

γ állapot: $Q_0Q_{-1} = 01$ esetén ide ugrunk a β állapotból és itt, a részösszegek *összeadását* kell elvégezni.

δ állapot: $Q_0Q_{-1} = 10$ esetén ide ugrunk a β állapotból és itt a részösszegek *kivonását* kell elvégezni.

ε állapot: $Q_0Q_{-1} = 00$, ill. $Q_0Q_{-1} = 11$ esetén közvetlenül ide ugrunk a ß állapotból. Itt a Z, Q regiszterpár eggyel együttesen jobbra shiftelődik és az aktuális részszorzat ciklus teendői befejeződnek, emiatt itt dekrementáljuk a C hátraszámlálót.

φ állapot: Megvizsgáljuk C belső állapotát,

C ≠ 0-nál: további ciklusnak kell jönnie, ezért visszaugrunk β-ba és a β ... φ ciklus megismétlődik;

C = 0-nál: ez volt az utolsó ciklus és a folyamat leáll Ψ állapotban.

Ψ állapot: Ebben az állapotban az egyes regiszterek – melyek tartalma időközben folyamatosan változott – már a szorzás végeredményét tartalmazzák az *előjellel együtt:*

Z, Q = SZORZAT (A × B = P)
R = SZORZANDÓ (A)
C = 0

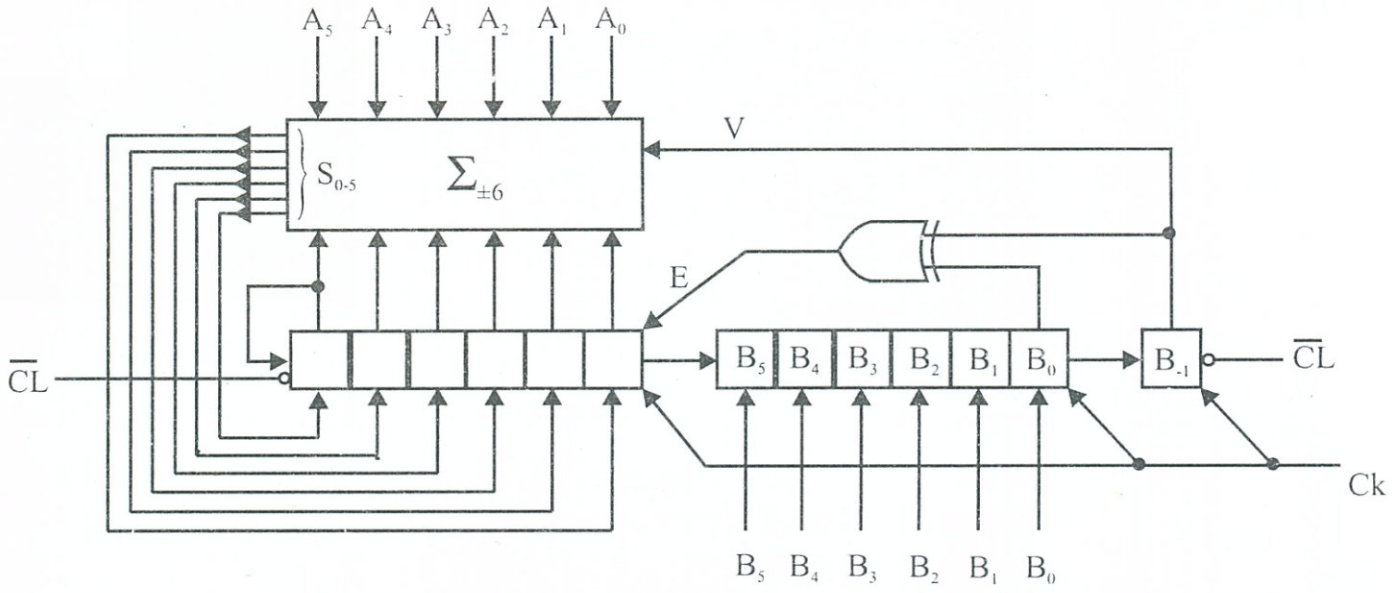
Miután itt is – bizonyos értelemben – SOROS szorzásról volt szó, a szorzási műveleti időre hasonló meggondolások érvényesek, mint az egy bit-figyelésű algoritmus esetén.

Szemléltetésül itt is bemutatunk egy részletezett számpéldát, melynek lépéseit a 10.65. ábrán követhetjük:

	-3_{10}		$+2_{10}$		
	A = 1 1 0 1$_2$	x	0 0 1 0$_2$	= B	
	→ 0 0 0 0		0 0 1 0	0	léptetés C = 4
Kivonás komplemens összeadással	0 0 0 0		0 0 0	1 0	kivonás majd
	A_k +0 0 1 1				léptetés
	→ 0 0 1 1		0 0 0	1 1 0	C = 3
Normál összeadás	0 0 0 1		1 0 0	0 1	összeadás majd
	+1 1 0 1				léptetés
	→ 1 1 1 0		1 0 0	0 1	C = 2
	→ 1 1 1 1		0 1 0	0 0	léptetés C = 1
	→ 1 1 1 1		1 0 1	0 0	C = 0
	Z		Q	Q_0 Q_{-1}	

$P = A \times B = (-3_{10}) \times (+2)_{10} = (0011_2)_k \times (0010_2) = 1101_2 \times 0010_2$
$P = 11111010_2 = -6_{10}$

10-65. ábra. *Bináris komplemenses szorzási példa 2-bit figyelésű algoritmusra*



10 - 66. ábra. *Egy 6-bites Booth szorzó realizációja*

Végezetül a 10.66. ábrán felrajzoltuk egy 6-bites kiépítésű szorzóáramkör részletezettebb sémáját, mely a Booth-algoritmus szerint működik.

Az ábrán a V vezeték választja ki az üzemmódot: 1 érték esetén összeadás, 0-nál pedig kivonás történik:

Az elrendezés lényeges eleme a tulajdonképpen 13 bites léptetőregiszter, melynek legalsó helyiértékét (B-1) indítás előtt – mint láttuk – nullázni kell. A regiszter felső helyiértékeit szintén törölni kell, a későbbiekben itt fog keletkezni a szorzat, melynek jobbra léptetésekor a mindenkori előjel-bit lép be balról. A regiszter felső 6 helyiértékén lehetőség van a párhuzamos beírásra. Ennek engedélyezését az E bemenet végzi, E = 0 esetben nincs beírás.

Vizsgáljuk most meg, hogyan működik ez a szorzóáramkör. Induláskor B_{-1}-ben 0 van. Ha a B_0 bit értéke 0, nem történik semmi, és a következő órajelre a B regiszter felső fele (0-k sorozata) eggyel jobbra lép. Ha B_1 értéke 1 volt, a Booth-féle algoritmus szerint kivonást kell végezni. Ekkor a V bemenetre 0 érkezik, tehát a $\Sigma_{\pm 6}$ áramkör valóban kivonást végez, azaz A 2-es komplemensét adja hozzá a B-ben található 0-hoz, s ez az összeg beíródik B felső részébe, mivel E ilyenkor 1 (B_0 és B_{-1} különbözőek). Az ezt követő léptetés hatására B_{-1}-be a korábbi B_0, B_0 helyére pedig B_1 kerül, s az összeadás (kivonás) folytatódik. Összeadásra mindig akkor fog sor kerülni, ha B_{-1}-ben 1 található, B_0-ban pedig 0. Ha e két bit azonos, a részeredmény egyszerűen tovább lép jobbra és nem változik az értéke. Mikor a B regiszterben a szorzó legmagasabb helyiértékű bitje (B_5) éppen kilépett

(azaz belépett B_{-1}-be), akkor a B regiszter 12 bitjében a szorzat 2-es komplemens kódú alakja lesz található.

A $\Sigma_{\pm 6}$ (6-bites összeadó/kivonó) áramkör hasonló módon működik, mint a 10.54. ábrán megismert kapcsolás.

10.6.5. BCD szorzás

E pontban a BCD szorzás egy gyakran alkalmazott változatát mutatjuk be, mely az ún. „duplázó–felező" algoritmuson alapul. A szorzási műveletet itt összeadási művelettel kombinált duplázó–felező lépésekre vezetjük vissza.

Az *algoritmus* az alábbi matematikai átalakítások szisztematikus felhasználásán alapul:

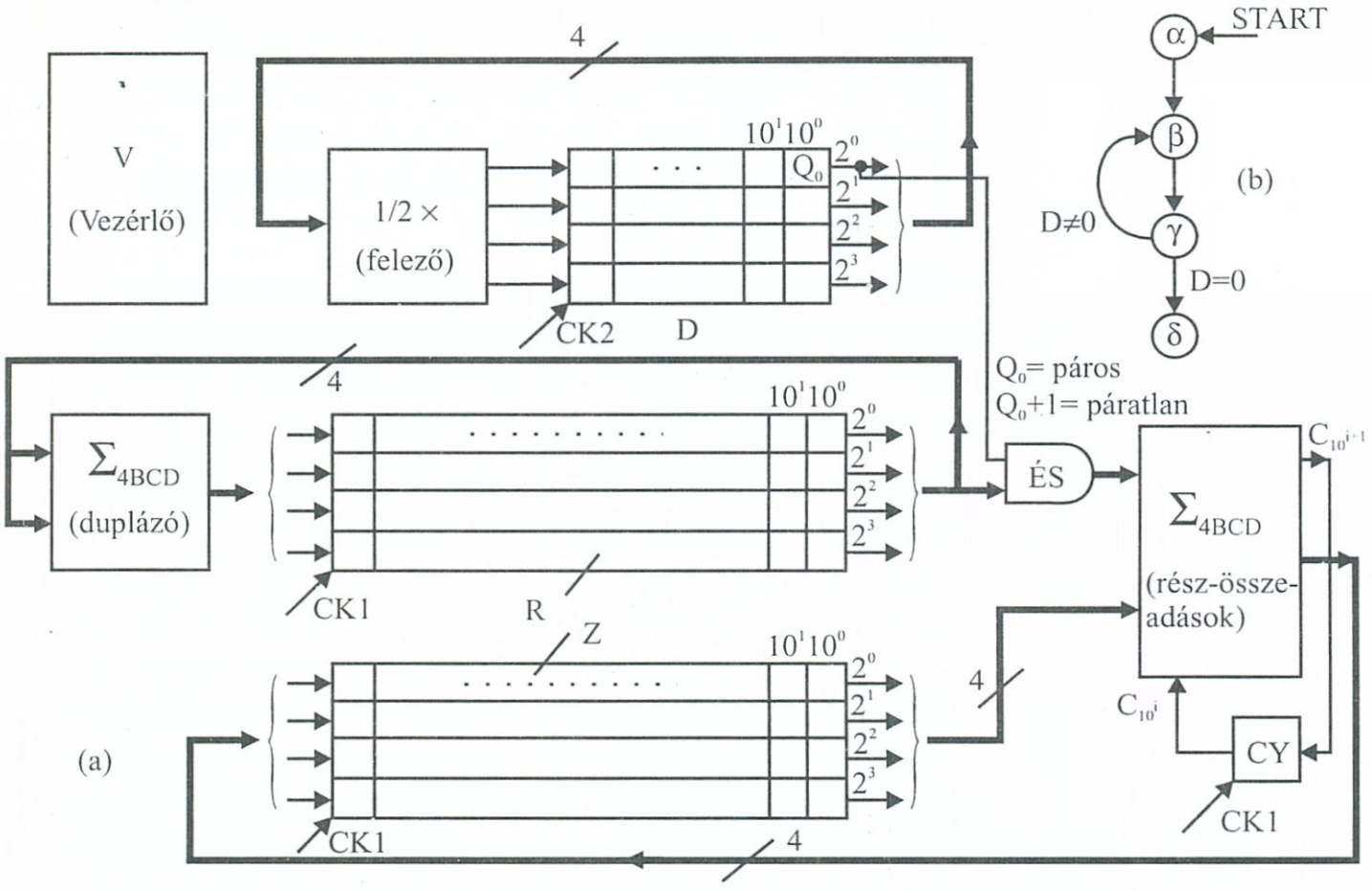
$$\left.\begin{aligned} A \times B &= 2A \cdot \frac{B-1}{2} + A \qquad \text{(ha B páratlan szám)} \\ A \times B &= 2A \cdot \frac{B}{2} \qquad \text{(ha B páros szám} \end{aligned}\right\} \quad (10.38)$$

ahol: A = szorzandó
B = szorzó

A (10.38) formulákból látszik, hogy ezek felhasználásakor a szorzandó *duplázásával* egyidőben a szorzó *felezésére* is sor kerül. Innen ered az algoritmus elnevezése is. A szorzási művelet szisztematikus lépései során a szorzandó duplázását és a szorzó felezését mindaddig folytatni kell, amíg a feleződő szorzó a „0"-t el nem éri. A lépések során keletkező *páratlan* értékű szorzók esetén az aktuális szorzandót mindig *hozzá kell adni* a megelőző fázisokban keletkezett szorzandók összegéhez. Ez a mozzanat a (10.38) első összefüggéséből adódik. *Páros* értékű szorzónál – (10.38) második összefüggésének megfelelően – „0"-t kell hozzáadni a megelőző fázisokban keletkezett szorzandók összegéhez.

A fent összefoglalt algoritmus végrehajtására használható fel. a 10.67a. blokkvázlattal jellemzett hálózat, melynek működését a V vezérlőegység irányítja az előzőleg röviden összefoglalt szorzási algoritmushoz illeszkedő 10.67b. ábra állapotgráfnak megfelelően.

A D, R, Z regisztertömbökben 4-4 párhuzamosan lépkedő soros regiszter működik együtt. Egy-egy függőleges oszlopban szereplő, egymás alatti négy bit egy bináris tetráddal kifejezett dekádot képvisel, és a dekádok decimális súlyozása jobbról–balra növekszik. A dekádok az órajelek hatására sorosan jobbra lépkednek.



10 -67. ábra. *BCD szorzóberendezés duplázó - felező algoritmussal*

A *B osztó* a D regisztertömbben van tárolva, melynek hossza az ismételt felezgetések miatt n tetrád.

Az *A szorzó* az R tömbben van tárolva, melynek hossza az ismételt duplázások miatt 2n tetrád.

Végül az *A×B-szorzat* a Z tömbben fog kialakulni, így ennek hossza is 2n tetrádot igényel.

Az algoritmus lépései a 10.67b. állapotgráf ütemezésében a következőkben foglalhatók össze:

α állapot: Indulás előtt fel kell tölteni az egyes regisztertömböket és az átviteli CY tárolót, mely rendszerint egy D-flip-flop.

SZORZANDÓ : A → R
SZORZÓ : B → D
TÖRLÉS : 0 → Z
TÖRLÉS : 0 → CY

A START indítás után a szorzási algoritmus beindul.

β állapot: Az osztandó duplázása és az osztó felezése után *páros* osztó ($Q_0 = 0$) esetén 0-át (ilyenkor az ÉS-kapu lezár), *páratlan* osztó ($Q_0 = 1$) esetén (az ÉS-kapu kinyit) az aktuális osztandó értékét adjuk hozzá a Z tartalmához, melyben az egyes fázisokban keletkezett szorzandók összege halmozottan tárolódik.

γ állapot: Itt meg kell vizsgálni a felezések során egyre csökkenő B_i osztót tartalmazó D regiszter tartalmát, amennyiben ez még nem 0, akkor újabb ciklus következik, azaz *β*-t megismételjük. D = 0 esetén a szorzás befejeződik és átlépünk a *δ* állapotba.

δ állapot: Itt a ciklusok során változó tartalmú regiszterek már a szorzás végének megfelelő állapotban vannak.

$Z = \text{SZORZAT}\ (A \times B)$
$D = 0$

10.6.6. Bináris osztás

Az osztást, mint műveletet általánosságban a (10.39) összefüggésekkel jellemezhetjük:

$$\left.\begin{aligned} A:B = \frac{A}{B} &= Q + \frac{R}{B} \\ A &= Q \cdot B + R \end{aligned}\right\} \qquad (10.39)$$

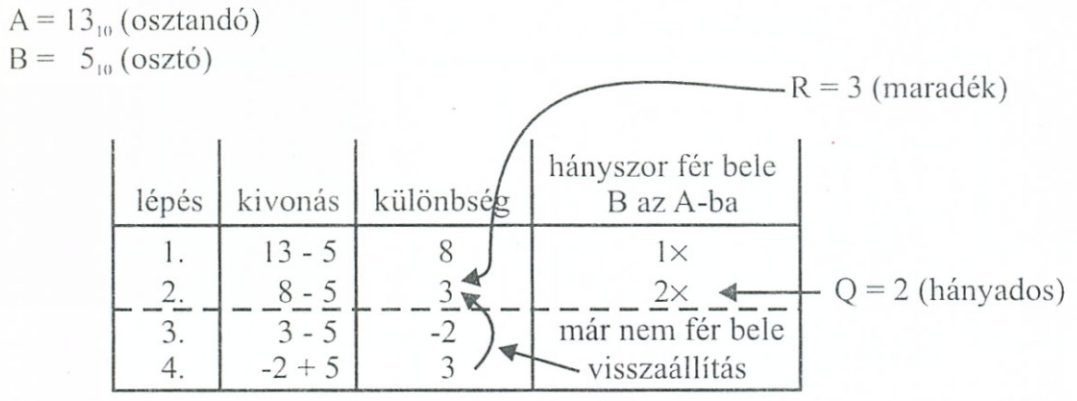
ahol: A – osztandó
B – osztó
Q – hányados
R – maradék

és a *törtvonal* fejezi ki a másként A:B módon jelölt *osztási* műveletet. Normális körülményeket feltételezve még az is kimondható, hogy:

$$B \neq 0$$

10.6.6.1. Osztás visszavezetése kivonásra

Az osztást visszavezethetjük megismételt kivonási és összehasonlítási műveletek együttesére, melyet egy előzetes, decimális számokat tartalmazó példán szemléltetünk. A lépéseket a 10.68. ábrán követhetjük. A cél az, hogy megállapítsuk, hogy a B osztó hányszor „fér bele" *egész-számszor* az A osztandóba (ez fogja ugyanis megadni a Q hányadost), továbbá, meg kell még állapítanunk az R maradékot is. A

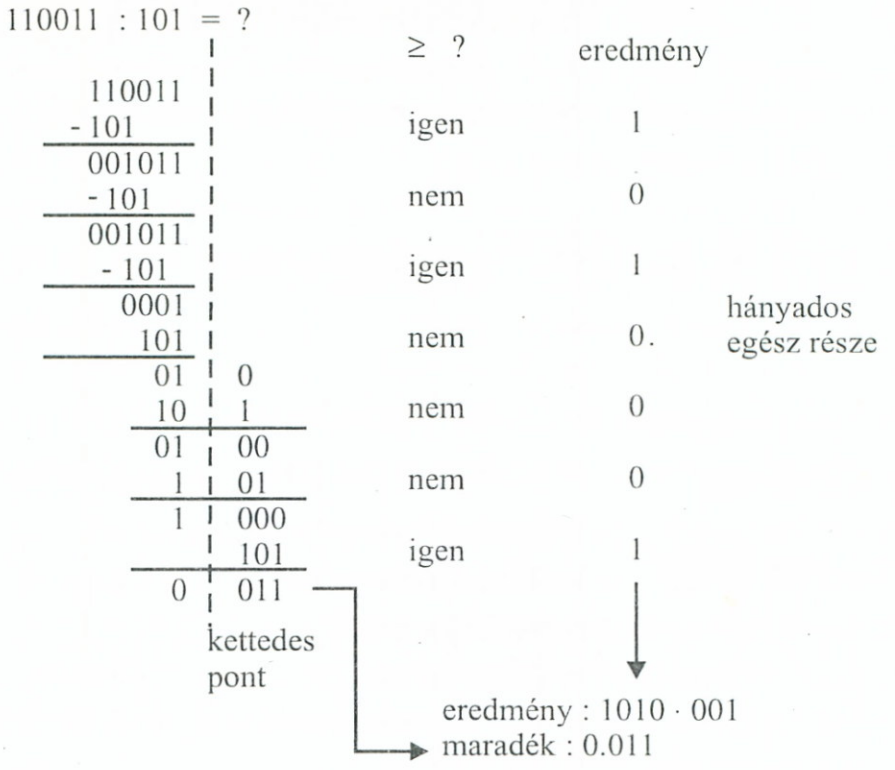


10 - 68. ábra. *Osztás visszavezetése kivonásra és összehasonlítási műveletre*

10.68. ábrán a 3. lépés kivonása negatív eredményt hoz, ebből kiderül, hogy a B 2× fért bele az A-ba, tehát a hányados Q = Z, míg a maradék az utolsó *nem negatív* eredményű kivonás (2. lépés) eredményéből adódik, azaz R = 3. Fentiek felismeréséhez tulajdonképpen el kellett végeznünk a végeredmény szempontjából „felesleges" 3. lépés kivonását is, mivel itt derült ki, hogy már a 2. lépésben az algoritmus leállítandó. Soros algoritmusok esetében ezt a „túlfutást" általában kompenzálni kell, hogy visszaléphessünk az algoritmus „utolsó hasznos" fázisába. Ezt a műveletet *visszaállítás*-nak nevezzük és egyszerű példánkban ez történik a 4. lépésben, ahol egy kompenzáló összeadással valójában visszaállítjuk az algoritmust a 2. lépésbe.

10.6.6.2. Bináris osztás kivonással

A bináris osztás is visszavezethető kivonásra és összehasonlításra az előzők alapján. Egy elterjedt algoritmus szerint az osztót először úgy kell elhelyezni az osztandóhoz képest, hogy a legmagasabb helyértékek *egymás alá* kerüljenek. Ha ekkor az osztandó nagyobb az osztónál, az osztót (az adott, eltolt helyérték szerint) ki kell vonni, majd az osztandót eggyel balra léptetve, ezt az eljárást kell megismételni a kivonás eredményével. Az osztandó léptetését, összehasonlítását, kivonását mindaddig folytatni kell, míg az osztó a valódi helyérték pozíciójába nem kerül és a felsorolt lépéseket itt is el nem végeztük. A hányados eközben úgy keletkezik, hogy valahányszor az osztandót az osztónál nem kisebbnek találtuk, 1-et, valahányszor pedig az osztó bizonyult nagyobbnak, 0-t írunk le. A fenti eljárással megkapjuk a hányados egész részét. Amennyiben kettedestört jegyekre is szükség van, az eljárás befejezése után az eredményben leírjuk a kettedespontot, majd folytatjuk az osztó jobbra csúsztatását, s



10 - 69. ábra. *Bináris osztás ismételt kivonással*

közben az osztandót 0 értékű kettedes jegyekkel egészítjük ki. Az osztás folyamatát illusztrálja a 10.69. ábra.

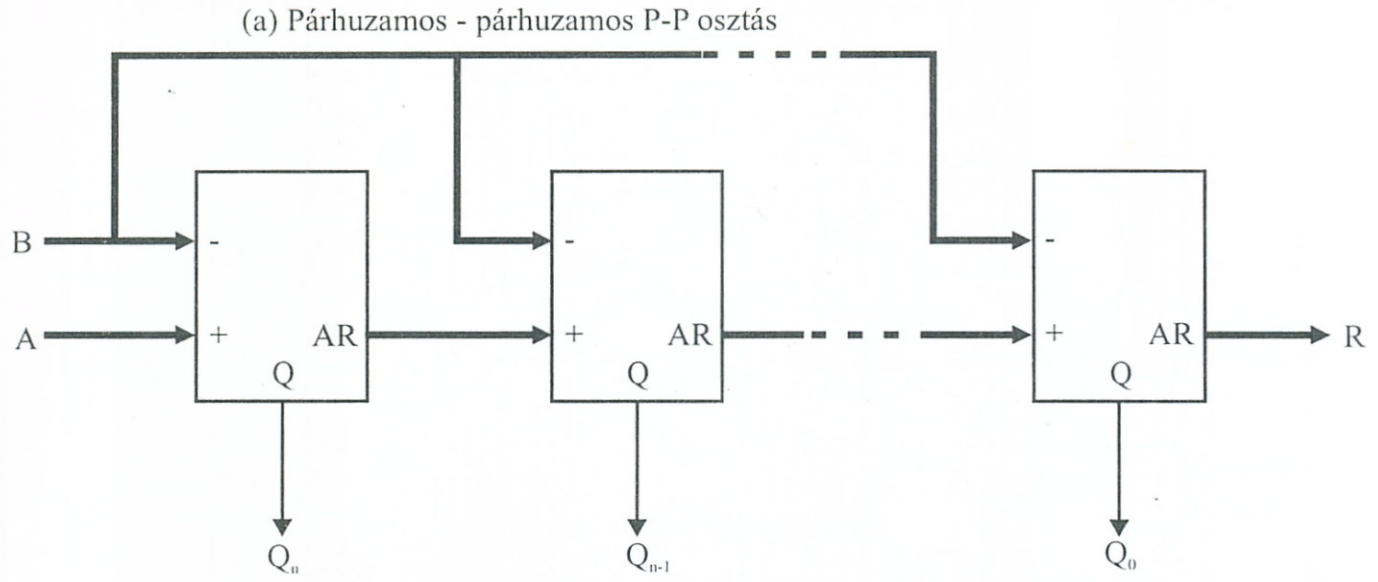
10.6.6.3. Kivonásos algoritmussal dolgozó osztóegységek

A 10.69. ábra bináris osztási algoritmusán alapulnak a következőkben bemutatandó osztó áramkörök, némi módosulásokkal.

Ezeknél is az osztandóból az osztót először 2 olyan magas hatványával szorozva (vagyis annyi helyiértékkel balra tolva) kíséreljük meg kivonni, hogy az osztandó legmagasabb helyiértékű bitje és az osztó legmagasabb helyiértékű bitje egymás alá kerüljenek. Ha a kivonás sikerült, vagyis az osztó ilyen módon még mindig kisebb az osztandónál, a hányados legmagasabb helyiértékű bitje 1 lesz, s a különbséggel fogjuk folytatni az előbbi eljárást oly módon, hogy az osztót most egy pozícióval jobbra léptetjük. Ha azonban a különbség negatív volt, a hányados legmagasabb helyiértékű bitje 0, s az osztót eggyel jobbra léptetve, az eredeti osztandóval folytatjuk az eljárást. Ezt az elvet követik az alábbi osztóberendezések is.

a) Párhuzamos–párhuzamos P–P osztás

A 10.70. ábra hálózatánál egy-egy blokk egy több-bites kivonó–döntő áramkört tartalmaz, melynek „+” bemenetére az osztandó,

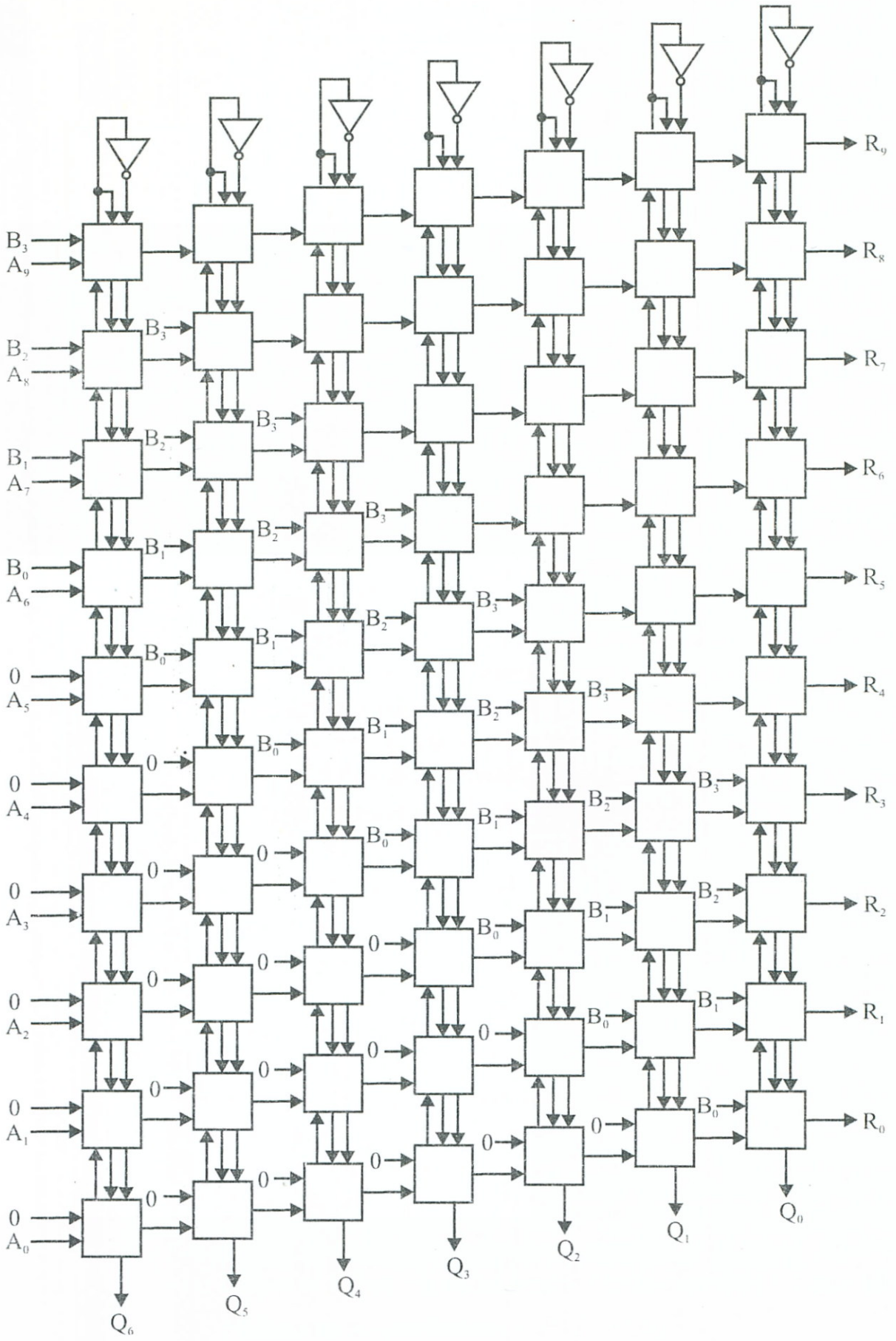


10 - 70. ábra. *Párhuzamos - párhuzamos bináris osztó*

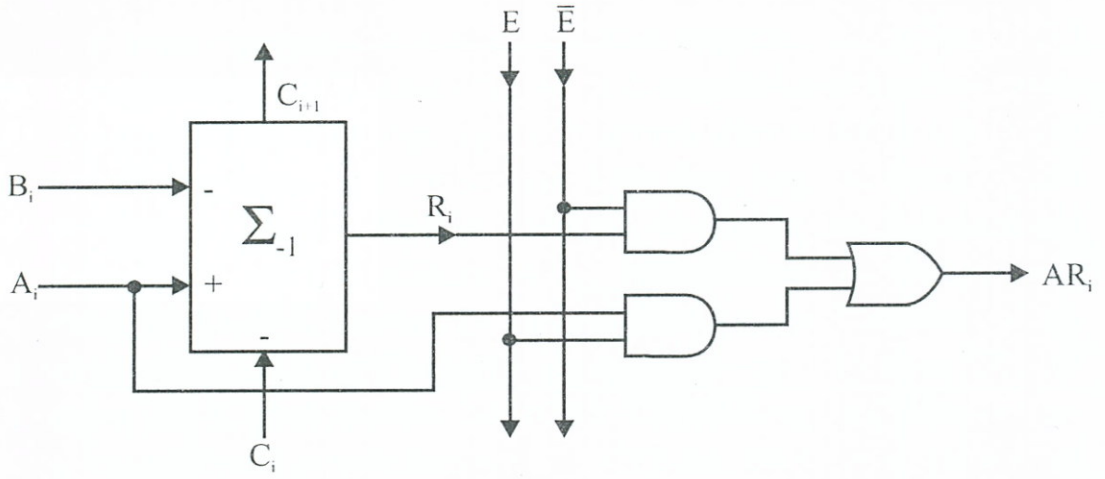
„–" bemenetére az osztó kerül. Az AR kimeneten vagy a változatlan osztandó (1 értékű áthozat esetén), vagy a maradék (0 értékű áthozat esetén) jelenik meg, a Q bit pedig a hányados, mely nem más, mint az áthozat negáltja. Az osztó kapcsolás részletezve látható a 10.71. ábrán, mely 10 bites osztandó és 4 bites osztó esetén működik. A hányados így 7 biten keletkezik, természetesen lehetséges, hogy a legmagasabb helyiértéken 0 van.

Egy-egy cella a 10.72. ábrának megfelelő kapcsolású.

A működés a következő: A Σ_{-1} jelű 1-bites egészkivonó A_i bemenetére az osztandó, B_i bemenetére az osztó megfelelő helyiértékű bitje kerül. A C_i bemeneten az előző helyiértékről érkezett áthozat, a C_{i+1} kimeneten pedig a következő helyiértékre továbbítandó áthozat található. E és $\overline{E}$ a legmagasabb helyiértékű áthozattól függő engedélyező bemenetek, melyek az összes helyiértéken végigfutnak, ha az áthozat 1, az AR_i kimeneten Ai változatlanul jelenik meg, ha viszont az áthozat 0, AR_i-re R_i kerül. Az ilyen cellák együttesét csupán az engedélyező jelet előállító inverterrel kell kiegészíteni, s helyiértékenként a negált áthozat adja ki a Q_i hányadosbitet is. A 10.71. ábra rajzán természetesen mindazok az egészkivonók félkivonóra cserélhetők ki, amelyeknek egyik bemenetére biztosan 0 kerül. Ennek az áramkörnek a hátránya, hogy igen költséges, n bites osztandó és m bites osztó esetén $n \cdot (n - m + 1)$ kivonó-döntő áramkört és $n - m + 1$ invertert tartalmaz.



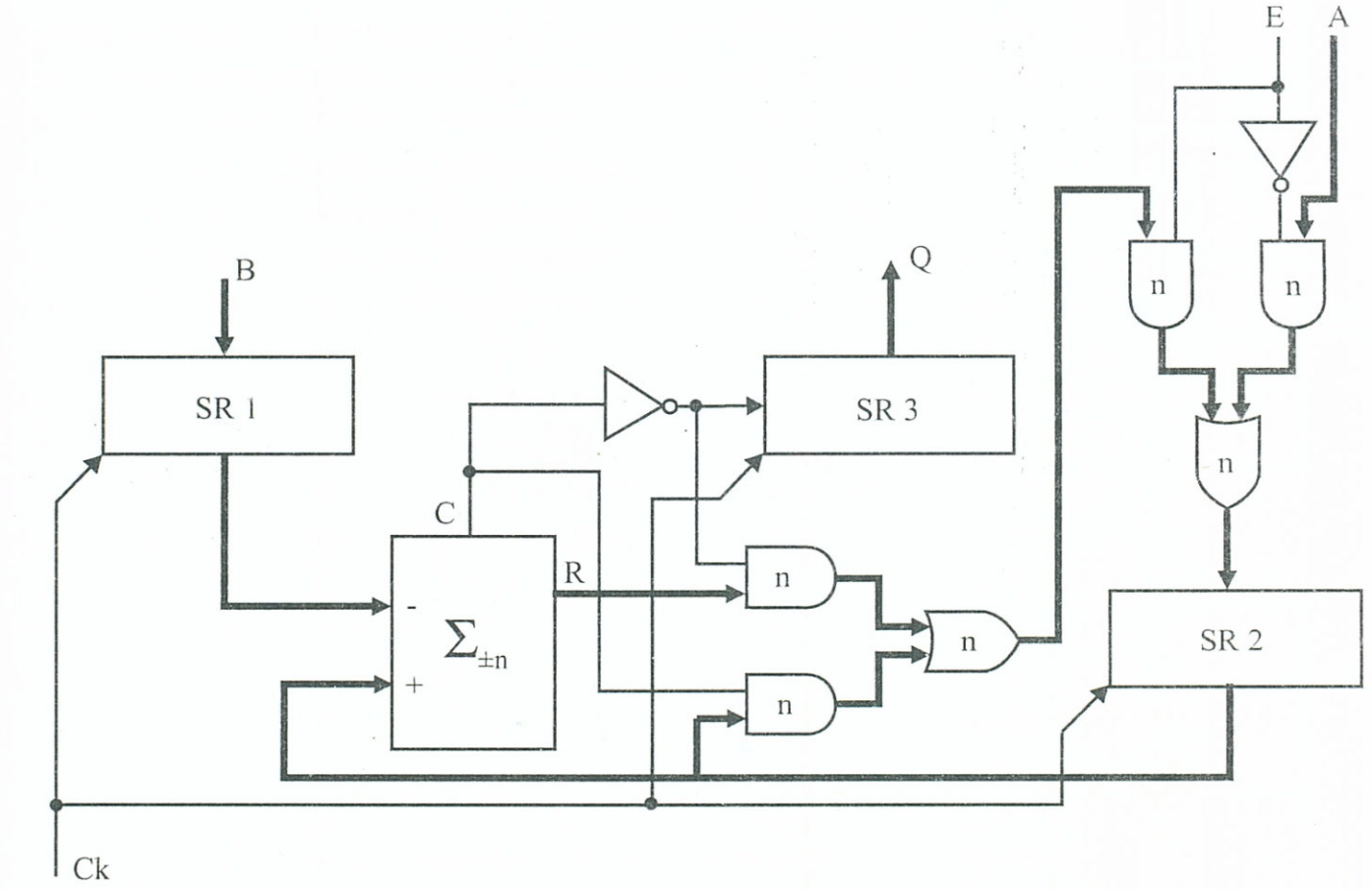
10 - 71. ábra. *Párhuzamos - párhuzamos bináris osztó cellastruktúrája*



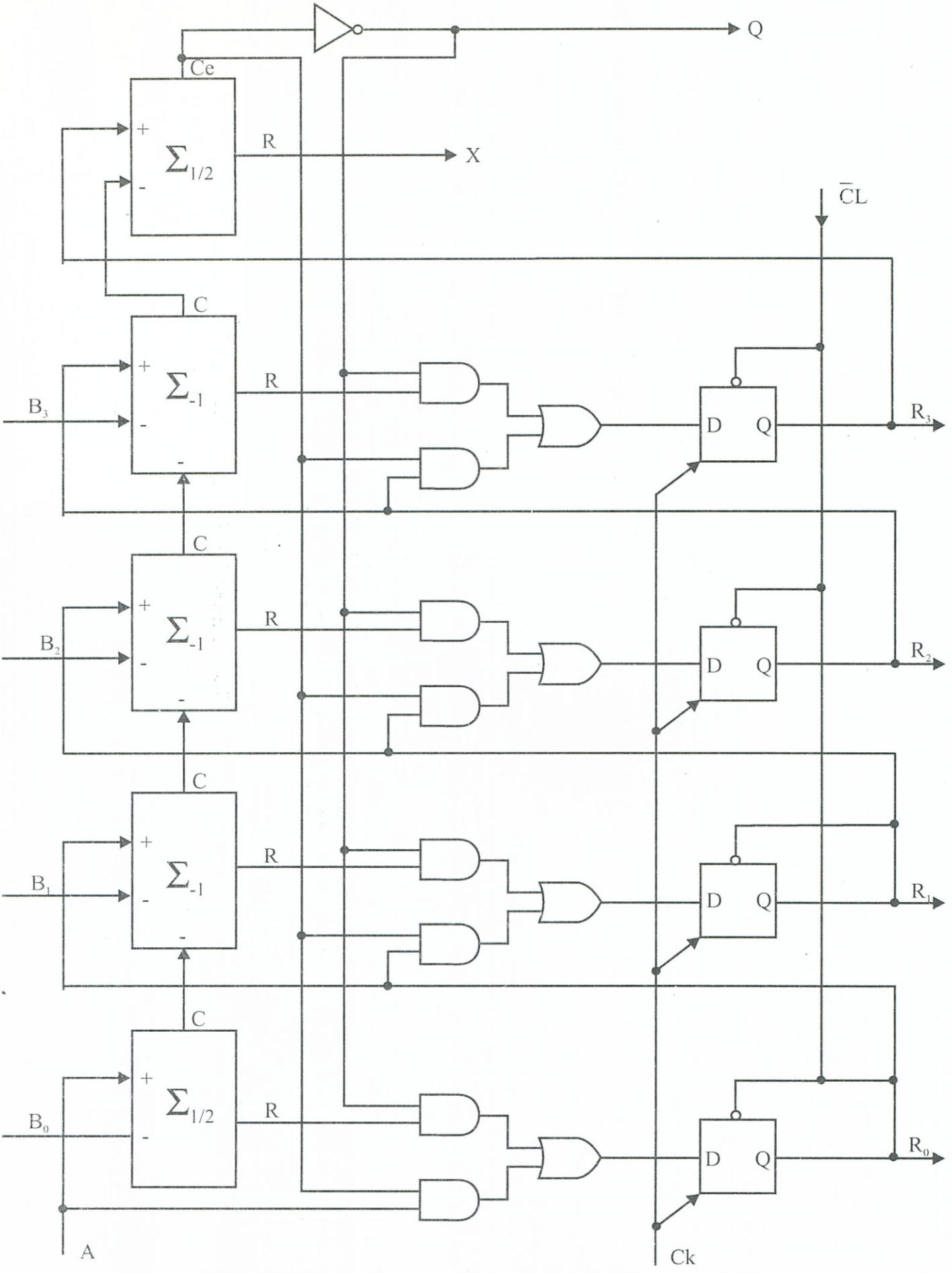
10 - 72. ábra. *Egy elemi cella felépítése*

b) Párhuzamos–soros osztás (P-S)

Ugyanezen elv alapján sorosan is végezhető osztás, ekkor természetesen az áramkör költsége lényegesen lecsökken, hiszen jóval kevesebb elemből épül fel, ennek ára azonban az lesz, hogy a sebesség viszont szintén jóval alacsonyabb lesz. A párhuzamos operandusok soros elven történő osztását végző áramkör a 10.73. ábrán látható.



10 - 73. ábra. *Párhuzamos - soros bináris osztó*



10 - 74. ábra. *Soros - párhuzamos bináris osztó*

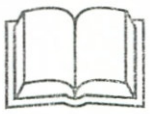
Az n-bites osztandó az A bemeneteken érkezik és az n-bites SR2 léptetőregiszterbe kerül, amennyiben az E engedélyező bemenet értéke 0. Az m-bites osztót a legmagasabb helyiértékre feltolva a B bemeneteken át az n-bites SR1 regiszterbe írjuk úgy, hogy az alacsonyabb helyiértékekre 0 kerül. Ezután indulhat az osztás, az E bemenetet azonban előbb 1-re kell állítani. A $\Sigma_{\pm n}$ jelű áramkör n-bites kivonó, mely megfelel a 10-71. ábra egy oszlopának. Az eredő áthozat a C kimeneten képződik, ennek negáltja lép be az (n – m + 1)-bites SR3 regiszterbe. Az R kimeneteken képződik a különbség, a C áthozat negáltja engedélyezi ennek beírását az SR2-be, ha C = 1, az eredeti kisebbítendő visszaíródik. Az egyszerűbb ábra érdekében n db ÉS-kaput, illetve n db VAGY-kaput egyetlen szimbólummal jelöltünk.

c) Soros–párhuzamos osztás (S–P)

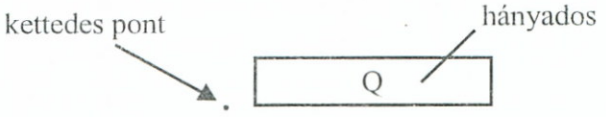
Ugyanezen elven alapuló újabb módosított realizáció valósítja meg a soros osztandó és párhuzamos osztó esetében működő áramkört (10-74. ábra).

A példában 4-bites osztó a B_i bemeneteken áll rendelkezésre, a tetszőleges hosszúságú osztandó pedig a legmagasabb helyiértéktől kezdve az A bemeneten érkezik. Közben a Q kimeneten jelennek meg a hányados bitjei sorosan, szintén a legmagasabb helyiértéktől kezdve. Az osztás megkezdése előtt a D-tárolókat a CL bemenet segítségével törölni kell, így mindaddig, amíg az osztandó bitjei közül annyi be nem lépett, hogy az így a D-tárolókból képzett regiszterben tárolt szám (legalsó helyiértéknek tekintve az éppen A-n található bitet) már eléri vagy meghaladja a B osztót, a legmagasabb helyiértéken keletkező C_e áthozat mindig 1 lesz, azaz a hányados bitjei 0-k. Amíg az áthozat 1, az ÉS-kapuk mindig az eggyel alacsonyabb helyiértékről érkezett bit változatlan beírását fogják engedélyezni a D-tárolókba, így a 4 D-tároló egyszerűen léptető regiszterként fog működni. Mihelyst $C_e = 0$ keletkezik, megjelenik az első 0-tól különböző Q bit és ugyanakkor az ÉS-kapuk a kivonókból érkező R maradékok beírását engedélyezik, tehát az ilyen lépések eredményeképpen a D-tárolók alkotta regiszterben léptetés helyett a különbség bitjei jelennek meg. Természetesen ezek tovább fognak felfelé lépkedni (valójában az osztó lépked lefelé hozzájuk képest), s alulról mindig az osztandó újabb, alacsonyabb helyiértékű bi-jei lépnek be folytatólagosan. A legfelső helyiértéken soha nem keletkezhet 1 értékű maradék, ezért ezt a kimenetet (X) nem használjuk fel. Az osztás befejezése után a maradék az Ri kimeneteken olvasható ki párhuzamosan. Az ábrán Σ_{-1} az egész kivonó, $\Sigma_{-1/2}$ pedig a fél kivonó jele.

d) Tört hányadost előállító osztóberendezés



Ez az osztóberendezés ugyancsak az ismételt kivonás elvén működik és tört hányados (A:B, ha A ≤ B) kiszámítására alkalmas előjel-abszolútértékes ábrázolást feltételezve. Az eredményt ún. „kettedes-tört" alakban kapjuk a 10-75. ábra szerinti értelmezésben:



10 - 75. ábra. *Kettedes - tört - es ábrázolás*

Az algoritmus az abszolútértékek hányadosát állítja elő és rokonságban van a korábbi osztó algoritmussal. Az ismételten képződő részmaradékokat a következő formulával lehet megadni:

$$R_i = rR_{i-1} - Q_i \cdot B \qquad (10.40)$$

ahol: R_i – aktuális rész-maradék
r – radix (kettes számrendszer: r = 2)
R_{i-1} – előző rész-maradék
Q_i – részhányados
B – osztó

A (10.40) formula érvényes más számrendszerben is, csak r helyett a megfelelő radixot kell beírni.

Az osztási algoritmus előzetesen a következőkben foglalható össze, figyelembevéve a 10.68. ábra példájánál tapasztaltakat is:

- Az első részmaradék maga az *osztandó* $A = R_0$.
- A (10.40) formula szerint az előző részmaradékot 2-szerezni kell (ez történhet *balra* léptetéssel, majd ki kell vonnunk a kétszerezés eredményéből az osztót.
- Ha keletkezett átvitel (CY = 1), azaz az új részmaradék *pozitív,* akkor a kivonás elvégezhető volt, ilyenkor $Q_i = 1$ lesz, és felírható, hogy $Q_i = CY = 1$.
- Ha nem keletkezett átvitel (CY = 0), azaz az új részmaradék *negatív,* akkor a kivonást nem lett volna szabad elvégezni, ezért $CY = Q_i = 0$ lesz és az állapotot *vissza kell állítani* a „felesleges" kivonás előtti értékre. A visszaállítás az osztó hozzáadásával történik. (Lásd: 10-68. ábra példája)
- A folyamatot mindaddig folytatni kell, amíg a részmaradék képzést minden i-re el nem végeztük, n-bites szóhossz esetén a lépések száma: n.

Az algoritmust matematikailag is alátámaszthatjuk a következő rövid levezetéssel. Ha az egyes mozzanatokra sorba felírjuk a (10.40) összefüggést:

$$R_0 = A$$
$$R_1 = 2R_0 - Q_1B = 2A - Q_1B$$
$$R_2 = 2R_1 - Q_2B = 2(2A - Q_1B) - Q_2B$$
$$\vdots$$
$$R_n = 2^n \cdot A - 2^{n-1} \cdot Q_1B - 2^{n-2} \cdot Q_2B - \ldots - 2^0 \cdot Q_nB$$

R_n átrendezése, 2^{-n}-el történő végigszorzás, és a kiemelések elvégzése után végül írható:

$$A = B \cdot \sum_{i=1}^{n} Q_i \cdot 2^i + R_n \cdot 2^{-n} \qquad (10.41)$$

vagy B-vel végig osztva kapjuk a kiinduló-formulával erősen rokon alábbi kifejezést:

$$\frac{A}{B} = \sum_{i=1}^{n} Q_i \cdot 2^i + \frac{R_n \cdot 2^{-n}}{B} \qquad (10.42)$$

ahol: – az első tag a *hányados*
– a második tag számlálója pedig a maradék.

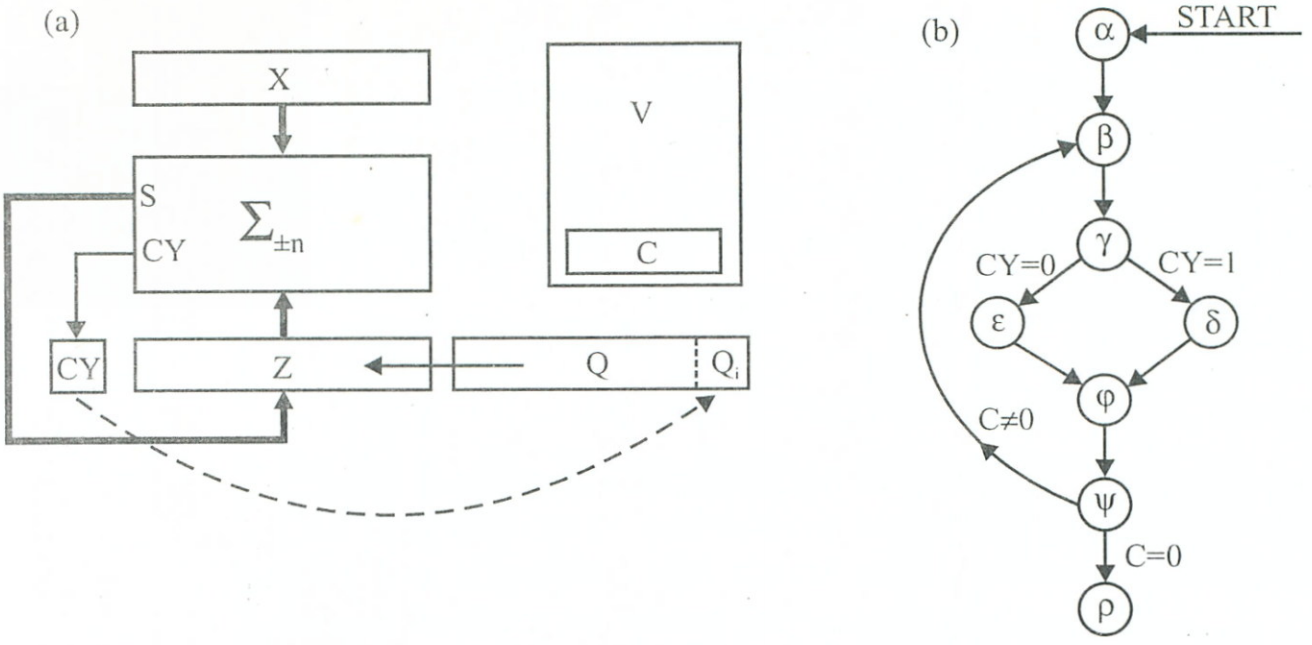
Mindkettőről megállapítható, hogy valóban tört számot képviselnek.

Az elmondottak végrehajtására használható fel a 10-76a. ábrán blokksémával jellemzett hálózat, melynek működését (a szorzási realizációknál tapasztaltakhoz hasonlóan) egy V vezérlőegység irányítja a 10-76b. ábra gráfjával értelmezett algoritmus szerint.

Az első, ami szembetűnik [és (10.40) alapján várható is volt] a 10-76a. ábra blokkvázlatánál, hogy itt a Z, Q munkaregiszterek *balra* lépkednek, és hogy az összeadó–kivonó áramkör átviteli kimenete is felhasználásra kerül a fent elmondottak értelmében.

A berendezés adatokkal feltöltése előtt célszerű meggyőződni arról, hogy az OSZTÓ $\neq 0$ ($B \neq 0$) és az algoritmus által megkívánt „törthányados" ($A \leq B$) feltételek teljesülnek-e. Ezután, az előzetesen már összefoglalt *osztási algoritmus* egyes lépéseit képviselő 10-76b. ábrabeli gráfállapotok-beli teendők a következőképpen kell, hogy alakuljanak:

α állapot: Indulás előtt fel kell tölteni az egyes regisztereket és az átvitel bitet nullázni kell:



10 - 76. ábra. *Bináris osztóberendezés algoritmizált vezérlővel*

OSZTANDÓ : $|A| \rightarrow Z$
OSZTÓ : $|B| \rightarrow X$
TÖRLÉS : $0 \rightarrow Q$
TÖRLÉS : $0 \rightarrow CY$
CIKLUSSZÁM : $n \rightarrow C$

A START indító parancs után az osztási folyamat beindul.

β állapot: Itt a Z, Q regiszterpárt *balra* kell léptetni (2-vel való szorzás), majd el kell végezni a részmaradék számításához a *kivonási* műveletet is.

γ állapot: Ha ez sikeres, akkor CY = 1 és elugrunk a *δ* állapotba. Ha „nem kellett volna" elvégezni a kivonást, akkor CY = 0, ekkor az *ε* állapotba ugrunk.

δ állapot: A *sikeres* kivonásnak megfelelően a CY = 1 értéket beírjuk a Q regiszter pillanatnyilag *legutolsó* bitjébe: $CY = 1 \rightarrow Q_i$, majd továbblépünk *φ* állapotba.

ε állapot: A *sikertelen* kivonás következtében $CY = 0 \rightarrow Q_i$ beírás történik és a rendszer-állapotot *vissza kell állítani* (összeadással) a kivonás előtti értékre, majd továbblépünk a *φ* állapotba.

φ állapot: Miután az aktuális részmaradékkal kapcsolatos teendők befejeződtek, itt dekrementáljuk a C hátrafelé lépő ciklusszámlálót.

Ψ állapot: Ha $C \neq 0$, akkor még újabb részmaradék számítás kell, hogy következzen, ezért visszaugrunk a *β* állapotba.

Ha C = 0, akkor ez volt az utolsó ciklus és a műveletsorozat leáll a δ állapotban.

ρ állapot: Ebben az állapotban az egyes regiszterek (melyeknek tartalma túlnyomórészt folyamatosan változott az algoritmus során) már az osztás végeredményét tartalmazzák a következők szerint:

Q = HÁNYADOS (Q)
Z = MARADÉK (R)
X = OSZTÓ (B)
C = CIKLUS VÉGE (0)

Miután itt előjel-abszolútértékes számábrázolás van, a fentiek csak az eredmény abszolútértékét érintik. Az előjelszámítás itt is az osztásra vonatkozó „iskolás" szabály szerint történik, melynél, ha az osztó és osztandó előjele azonos, akkor pozitív, ha eltérő, akkor negatív előjelű az osztás eredménye. Ez, a konvenciók szerint az alábbi formulával számítható ki:

$$s_E = s_{osztó} \oplus s_{osztandó} \tag{10.43}$$

ahol: s_E – eredmény előjele
$s_{osztó}$ – osztó előjele
$s_{osztandó}$ – osztandó előjele

és a konvenció: + = 0, - = 1.

Végül, szemléltető célzattal itt is bemutatunk egy részletezett számpéldát, melynek lépései a 10-77. ábrán követhetők. A kivonásokat a B_k komplemens hozzáadásával képezzük, a visszaállítást B hozzáadással:

OSZTANDÓ: $|A| = |-3_{10}| = 0011_2$ (Z-be betöltve) n = 4
OSZTÓ : $|B| = |+5_{10}| = 0101_2$ (X-be betöltve), $B_k = 1011$

A példa ellenőrző-próbáját a kiinduló (10.38) összefüggésbe történő behelyettesítéssel végezhetjük:

$$\frac{|A|}{|B|} = \frac{3}{5} = Q + \frac{R}{B} = \frac{9}{16} + \frac{3/16}{5} = \frac{45}{80} + \frac{3}{80} = \frac{48}{80} = \frac{3}{5}$$

Nyilván felvetődik a kérdés, hogy a bemutatott *törthányadosú* osztás elvei alapján vajon kialakítható-e egy *egészhányadosú* algoritmus. A probléma nem nagy ráfordítással megoldható. A módszer részleteit a Feladatgyűjtemény vonatkozó részében egy ugyancsak részletesen kidolgozott példa keretében mutatjuk be.

Osztandó: $|A| = |-3_{10}| = 0011_2$ (Z-be betöltve) $n = 4$
Osztó : $|B| = |+5_{10}| = 0101_2$ (X-be betöltve) $B_k = 1011$

	CY	Z	Q	Q_i	C	C
$A=R_0$	0	0011	000	0	4	
$2R_0$		0110				←
$+B_k$		1011				
R_1	1	0001	000	1	3	CY→Q_1=1
$2R_1$		0010	001			←
$+B_k$		1011				
	0	1101		0		Vissza-
+B		0101				állítás
R_2	↖	0010	001	0	2	
$2R_2$		0100	010			←
$+B_k$		1011				
	0	1111			0	Vissza-
+B		0101				állítás
R_3	↖	0100	010	0	1	
$2R_3$		1000	100			←
$+B_k$		1011				
R_4	1	0011	100	1	0	CY→Q_4=1

Kettedes vessző — Maradék: $2^{-1}2^{-2}2^{-3}2^{-4}$ — Hányados: $2^{-1}2^{-2}2^{-3}2^{-4}$

MARADÉK : $R = 0,0011_2 = 1/8+1/16 = 3/16$
HÁNYADOS : $Q = 0,1001_2 = 1/2+1/16 = 9/16$
ELŐJEL : $s_E = s_{osztó} \oplus s_{osztandó} = 1 \oplus 0 = 1 \rightarrow$ Negatív

10-77. ábra. *Bináris törthányadosú osztás példája*

10.6.7. Műveletek lebegőpontos bináris számokkal

A lebegőpontos bináris számokkal végzett műveleteknél a (10.26) definíciós formulából kell kiindulnunk:

$$N_i = s_i, m_i \cdot 2^{k_i} \qquad (10.44)$$

A formulából látható, hogy minden számhoz háromféle jellemző tartozik, ezek

s_i = előjel
m_i = mantissza
k_i = karakterisztika (előjeles kitevő)

Ha a lebegőpontos számokkal műveleteket végzünk, *mindhárom jellemző* kiszámítását *külön-külön* el kell végeznünk az adott művelet

által meghatározott szabályok szerint, melyek műveletfajtánként (esetleg erősen) eltérők lehetnek.

a) *Szorzás*-nál pl. két operandusra a következők írhatók fel:

$$N_P = N_i \times N_j = (s_i, m_i \cdot 2^{k_i}) \times (s_j, m_j \cdot 2^{k_j}) \qquad (10.45)$$

A szorzat *előjelének* (s_P) kiszámításánál azt az „iskolás" szabályt kell alkalmaznunk, mely szerint, ha a szorzandók előjele:

megegyezik, akkor: + (0)
eltérő, akkor: – (1)

lesz az eredmény. A számításnál – mint már láttuk – jól felhasználható az *antivalencia* függvény, ha fenntartjuk a: – = 1, + = 0 konvenciót. Ilyenkor érvényes:

$$s_P = s_i \oplus s_j \qquad (10.46)$$

A szorzat *abszolút-értékének* ($m_P \cdot 2^{k_P}$) kiszámításához el kell végeznünk a két operandus exponenciális függvény összeszorzását:

$$|N_P| = m_P \cdot 2^{k_P} = (m_i \cdot 2^{k_i}) \times (m_j \cdot 2^{k_j}) = (m_1 \times m_j) \cdot 2^{k_i + k_j}$$

melyből az alábbiakat kapjuk:

$$m_P = m_i \times m_j$$

$$k_P = k_i + k_j \qquad (10.47)$$

Végeredményben megállapítható, hogy a szorzat három jellemzőjénél mindegyikhez más- és másfajta műveletvégző áramkört kell használnunk.

Így

s_P-hez : logikai ekvivalencia képzőt
m_P-hez : aritmetikai szorzót
k_P-hez : aritmetikai összeadót

A problémát tovább bonyolíthatja, hogy normalizált alakok esetén a szorzat normalizálásáról is gondoskodni kell.

b) *Osztásnál* ismét két operandust választva, a következők írhatók fel:

$$N_D = N_i : N_j = (s_i, m_i \cdot 2^{k_i}) : (s_j, m_j \cdot 2^{k_j}) \qquad (10.48)$$

A hányados *előjelének* (s_D) kiszámításánál az iskolai szabály „szerencsére" megegyezik a szorzatra vonatkozóval, így alkalmazható az előbbi antivalencia függvény:

$$s_D = s_i \oplus s_j \qquad (10.49)$$

A hányados abszolút-értékéhez ($m_D \cdot 2^{k_D}$) el kell osztani az osztandót az osztóval:

$$|N_D| = m_D \cdot 2^{k_D} = \frac{m_i \cdot 2^{k_i}}{m_j \cdot 2^{k_j}} = \frac{m_i}{m_j} \cdot 2^{k_i - k_j}$$

melyből az alábbiak adódnak:

$$m_D = \frac{m_1}{m_2}$$
$$k_D = k_i - k_j \qquad (10.50)$$

azaz, végül az osztásnál a következőkre van szükség:

s_D-hez : logikai antivalencia képző
m_D-hez : aritmetikai osztó
k_D-hez : aritmetikai kivonó

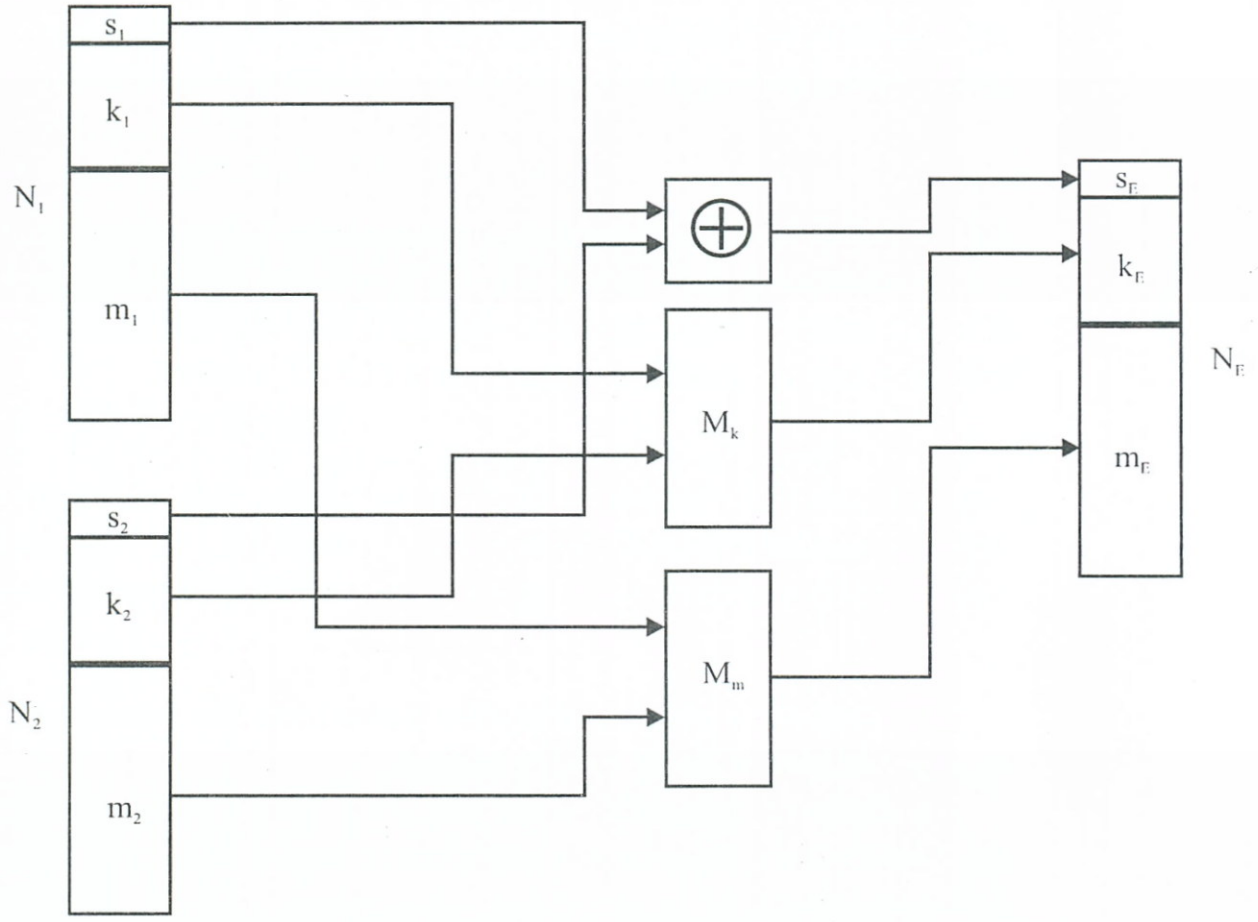
Az előzőkhöz hasonlóan, itt is ügyelni kell a normalizált alak biztosítására.

A szorzással és osztással kapcsolatosan elvégzett fenti meggondolások alapján (felhasználva a párhuzamosságokat) már felrajzolható egy blokkséma, melyből a lebegőpontos szorzás- és osztás elvégzésére alkalmas áramkör működési elve követhető.

A 10-78. ábra blokksémában az N_i és N_j operandusok egy-egy regiszterben, normalizált szektor elrendezésben helyezkednek el, és a megfelelő szektor-párok a nekik megfelelő két-operandusos műveletvégzők bemenetére csatlakoznak:

- s_i, s_j mind szorzásnál, mind osztásnál egy antivalencia kapura
- k_1, k_2 az M_K műveletvégzőkre, mely:
  szorzásnál: aritmetikai *összeadó*
  osztásnál: aritmetikai *kivonó*
  áramkört képvisel
- m_1, m_2 az M_m műveletvégzőre, mely:
  szorzásnál: aritmetikai *szorzó*
  osztásnál: aritmetikai *osztó*
  áramkörnek felel meg.

A három műveletvégző által kiszámított értékek egy kimeneti, a szorzatot normalizált szektorhelyes elrendezésben tartalmazó eredményregiszterbe kerülnek.



$N_E = N_P$ szorzásnál: M_k: összeadó, M_m: szorzó, $s_E=s_P$, $k_E=k_P$, $m_E=m_P$
$N_E = N_D$ osztásnál: M_k: kivonó, M_m: osztó, $s_E=s_D$, $k_E=k_D$, $m_E=m_D$

10- 78. ábra. *Lebegőpontos szorzó, -ill. osztó műveletvégző áramkör blokkvázlata*

c) *Összeadás- és kivonásnál* a helyzet lényegesen bonyolultabbá válik, mivel a

$$N_E = N_i + N_j = (s_i, m_i \cdot 2^{k_i}) + (s_j, m_j \cdot 2^{k_j}) \qquad (10.51)$$

összeadás, ill. kivonás csak akkor végezhető el, ha olyan matematikai átalakításokat végzünk, melyekkel a két operandusban azonos kitevőjűekké válnak a hatványfaktorok. Ezek az átalakítások a mantisszákra is kihatnak, ezért az áramkörök lényegesen bonyolultabbakká válnak, valamint a normalizált alakok kezelése is több odafigyelést igényel.

További problémát okoz, hogy az eredmény (összeg-, vagy különbség) előjel-megállapításánál *nem szimmetrikus* nagyság-relációkat kell alapul venni.

Felírhatók a következők az összevonásra:

$$N_{ÖV} = N_i \pm N_j = (s_i, m_i \cdot 2^{k_i}) + (s_j, m_j \cdot 2^{k_j})$$

Az *eredmény előjele* az abszolútértékben *nagyobbik* operandus előjelével fog megegyezni.

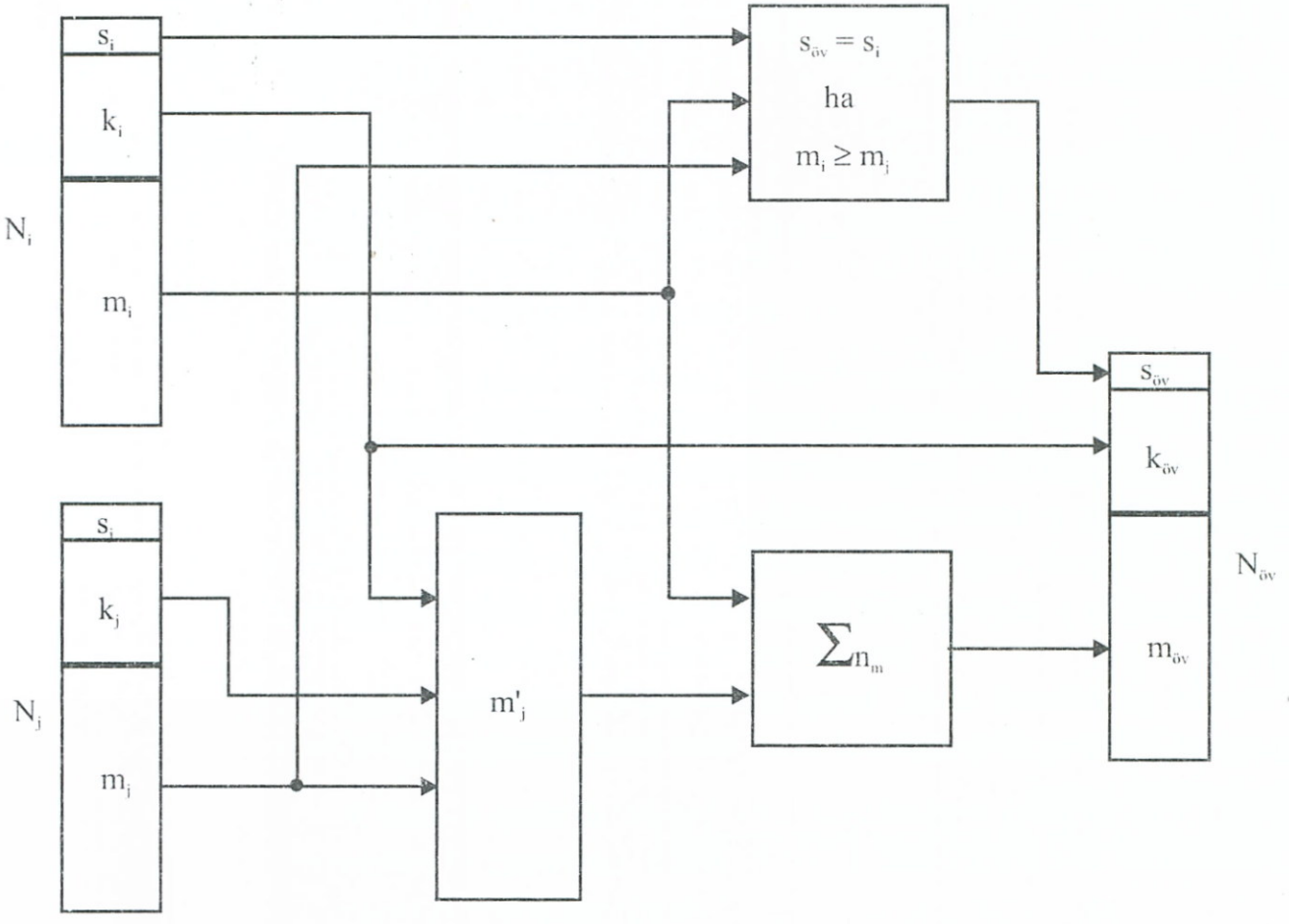
$$s_{öv} = s_i \quad \text{ha: } |N_i| \geq |N_j| \tag{10.52}$$

Az *eredmény abszolútértéké*nek kiszámításánál – mint fent említettük – az összevonást csak akkor lehet közvetlenül elvégezni, ha $k_i = k_j$. A $k_i \neq k_j$ esetben átalakítást kell végezni a hatványfaktorok azonos kitevőjűvé transzformálása érdekében a (10.53) szerint:

$$\left.\begin{aligned} &|N_{Ö,K}| = m_i \cdot 2^{k_i} \pm m \cdot 2^{k_j} = (m_i \pm m'_j) \cdot 2^{k_i} \\ &\text{ahol: } m'_j \cdot 2^{k_i} = m_j \cdot 2^{k_j} \\ &\text{és: } \qquad m'_j = m_j \cdot 2^{k_j - k_i} \end{aligned}\right\} \tag{10.53}$$

Mint látható, a közös k_i kitevőbe transzformálás kihatással van az m_2 mantisszára is, mely m'_2-vé alakult.

Az elvégzett meggondolások alapján a 10-79. ábrán felrajzoltunk egy blokksémát, melyen a lebegőpontos összevonás elve követhető. Az operandusok és az eredmény – hasonlóan a 10-78. ábrához – re-



10 - 79. ábra. *Lebegőpontos összevonás elvi sémája*

giszterekben helyezkednek el. Itt különös figyelmet kell fordítani az eredmény normalizált alakjának előállítására a (10.53)-beli transzformáció miatt.

10.7. Digitális komparátorok, összehasonlítók

10.7.1. Elvi működés

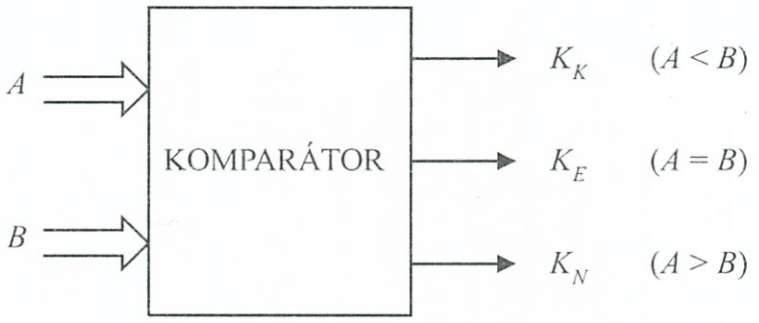
A komparátor elvi működését a 10-80. ábrán követhetjük. A komparátor az „A", ill. „B" bemenetekre érkező kódszavakat *mennyiségi tartalmuk* szemszögéből összehasonlítja, és az összehasonlítás eredményét az

$$\left.\begin{aligned} A < B &= K_K \\ A = B &= K_E \\ A > B &= K_N \end{aligned}\right\} \qquad (10.54)$$

relációknak megfelelően a három kimenet valamelyikén jelzi.

Komparálási feladatok megoldása mind digitális, mind analóg elven megvalósítható. Ebben a fejezetben a digitális komparálás kérdéseivel foglalkozunk, az analóg összehasonlítókat már a 4. fejezet 4.2. és 4.3. pontjaiban tanulmányoztuk.

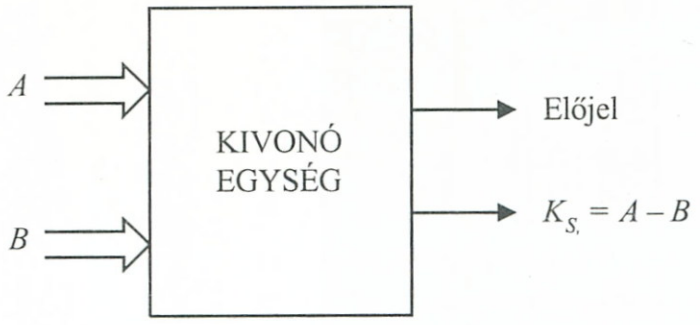
A (10.54) relációk vizsgálatánál kétféle lehetőségünk is adódik:



10–80. ábra *Komparátor elvi vázlata*

a.) Aritmetikai összehasonlítás

Itt az A és B közötti mennyiségi viszonylat megállapítására aritmetikai műveleteket alkalmazunk. Például a 10-81a. ábrán látható *kivonó egység* bemenetére adva A-t és B-t, a kivonás elvégzése után



a)

$K_{S.} = 0$	Előjel	Reláció
IGEN	Érdek-telen	$A = B$
NEM	+ (0)	$A > B$
	− (1)	$A < B$

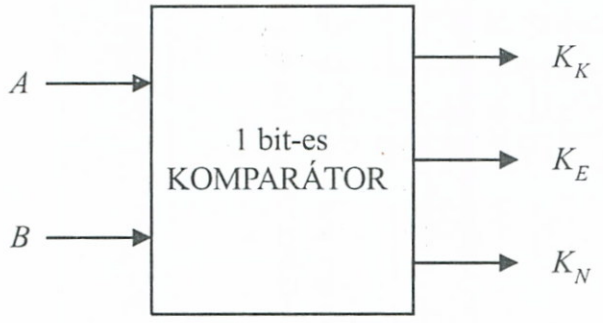
b)

10–81. ábra *Összehasonlítás aritmetikai úton*

(lásd 10.6. pont) a 10-81b. táblázat szerinti kiértékelést elvégezve tudjuk megállapítani, hogy melyik reláció érvényes.

b.) Logikai összehasonlítás

Itt a relációkkal kapcsolatosan meg kell gondolnunk, hogy amíg a K_E előállításához elegendő a hagyományos logikai szemlélet, az egyenlőségnek *ekvivalencia*-függvénnyel történő kifejezésével, addig a K_K, K_N „kisebb"-„nagyobb" értelmezéséhez további konvenciókra van szükség. Miután a kisebb-nagyobb relációk *mennyiségi* tar-



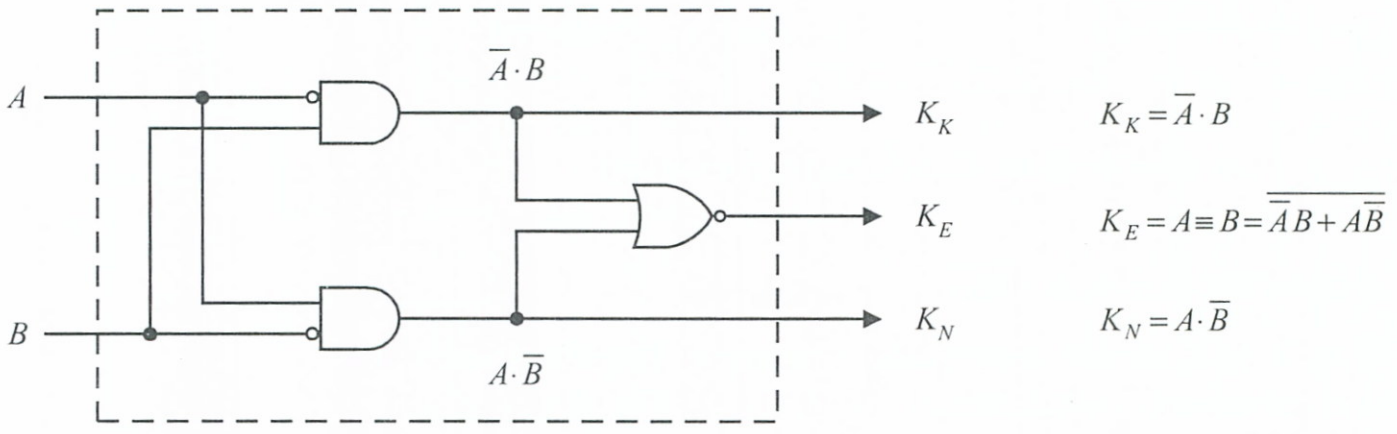
a)

A	B	K_K	K_E	K_N
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

b)

c)

d)



10–82. ábra *1 bit-es komparátor*

talmúak, ezért ilyen eseteknél mennyiségi megkülönböztetést kell tennünk a hagyományos logikai hálózatoknál eddig ilyen értelemben nem megkülönböztetett 0 és 1 logikai értékek, illetve az ezekből felépülő kódszavak között is.

A 10-82a. ábrán egy *1 bites komparátor* elvi vázlata szerepel, összhangban a 10-80. ábra szerintivel. Itt az A, ill. B kódszavak csupán *1 bit* hosszúságúak. A hálózat logikai vázlatának felrajzolásához meg kell állapodnunk előzetesen a logikai értékekre vonatkozó – már említett – mennyiségi összerendelésben. Ez a gyakorlatban, az esetek túlnyomó részében az alábbi szerint alakul:

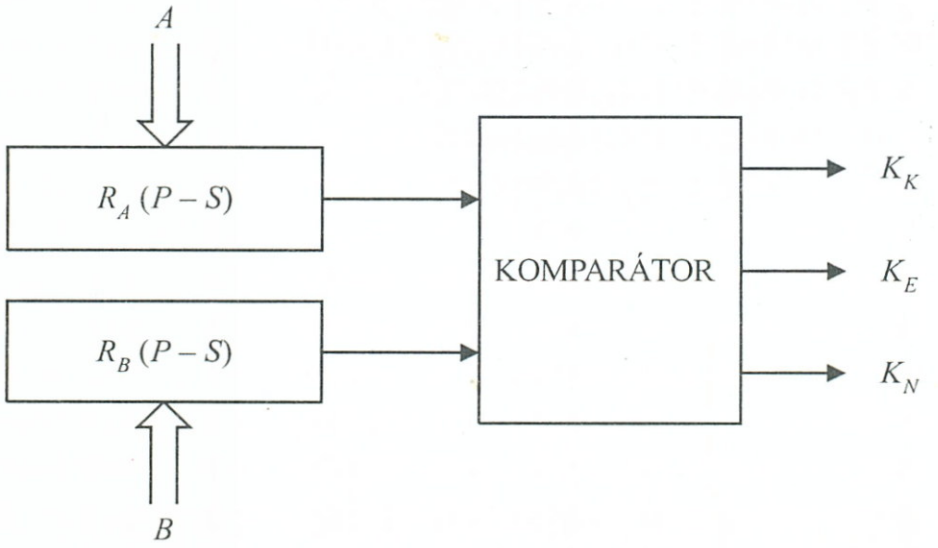
$$1 > 0$$

Konvenciónk alapján kitölthető a 10-82b. ábra szerinti logikai táblázat a háromféle kimeneti függvényre, melyeket a táblázatból algebrailag is felírhatunk a 10-82c. ábra szerinti módon. Végül megadhattuk a realizációs hálózat logikai vázlatát is a 10-82d. ábrának megfelelően.

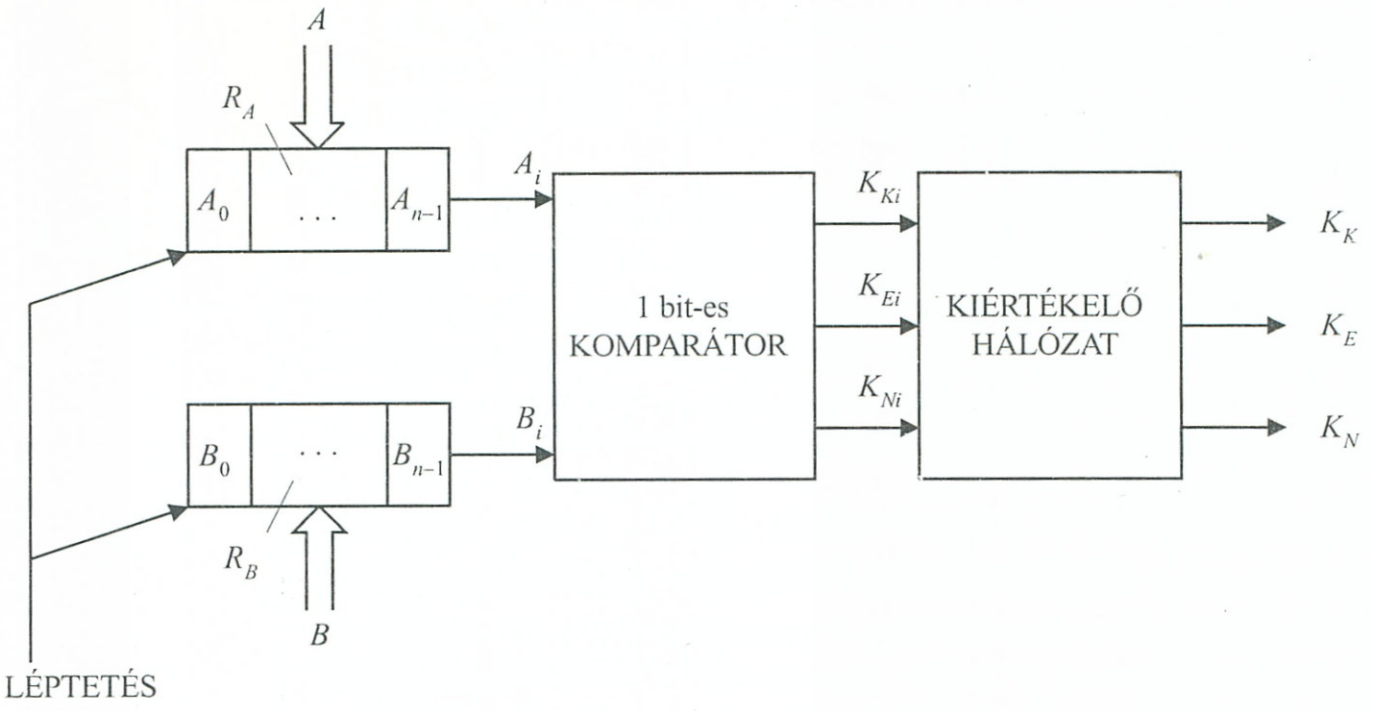
A *több-bites logikai* összehasonlítás megvalósítására *soros* és *párhuzamos* elven működő komparátorok alakultak ki.

10.7.2. Soros elven működő komparátorok

A soros elven működő komparátoroknál az összehasonlítás az *A* és *B* kódszavak egyes szakaszaira *időben eltolva* történik. (10-83. ábra) Ezáltal *hosszú kódszavak* összehasonlításakor nem szükséges nagykapacitású összehasonlító hálózatot alkalmazni. Természetesen az összehasonlítás így kialakuló többütemű folyamata hosszabb idő-



10–83. ábra *Soros elven működő komparátor*



10–84 ábra *n bit-es összehasonlítás 1 bit-es komparátorral*

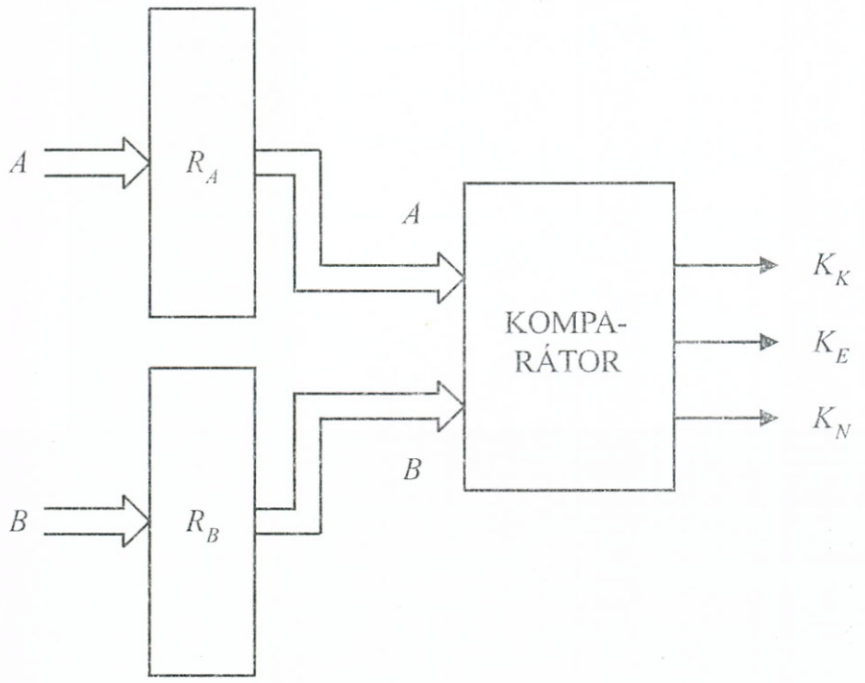
tartam alatt fog lezajlani, és a megoldás sorrendi hálózatot igényel. Vizsgáljunk meg egy fentiekre utaló példát.

A 10-84. ábrán egy, az előző példában tárgyalt *egybites* komparátort felhasználó, *n-bites összehasonlító egységet* tanulmányozhatunk. Az összehasonlítandó A és B kódszavak az R_A és R_B regiszterekben vannak tárolva az összehasonlítást megelőzően, mégpedig olyan elhelyezésben, hogy a *legmagasabb helyiérték* fog először az egy bit-es komparátorba belépni. A *CK* órajel engedélyezését követően a jobbra lépő regiszterek léptetése megindul, és *bit-helyiértékenként* megtörténik az összehasonlítás.

A lépésenként csökkenő helyiértékű A_i–B_i bit-párok összehasonlításakor azonnal eldől az $A \lessgtr B$ eredmény, amint valamelyik i-nél $A_i \lessgtr B_i$ tapasztalunk, mivel a magasabb helyértékre vonatkozó döntés már megszabja a továbbiak sorsát is a „kisebb", ill. „nagyobb" vonatkozásában. $A_i = B_i$ esetén mindig tovább kell lépnünk, és az $A = B$ relációt csak akkor mondhatjuk ki, ha az utolsó helyértéket elérve, itt is $A_0 = B_0$ tapasztaltunk. Ez utóbbi gondolatmenetet realizálandó, a hálózatot még ki kell egészítenünk egy itt nem részletezett „KIÉRTÉKELŐ" hálózattal is, melynek kimenetén a teljes kódszóra vonatkozó eredmény jelenik meg.

10.7.3. Párhuzamos elven működő komparátorok

A párhuzamos elven működő komparátoroknál az összehasonlítás az A és B kódszavak valamennyi bit-jére *egyidejűleg* történik (10-85. ábra).

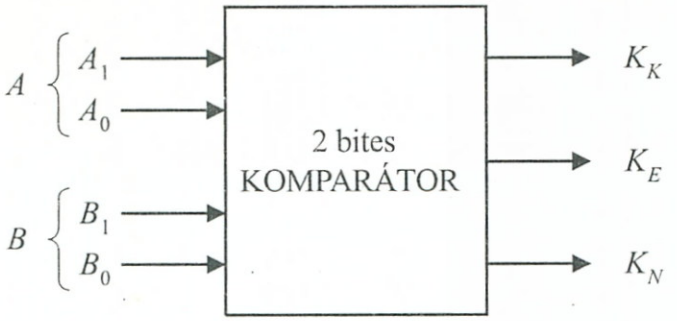


10–85. ábra *Párhuzamos elven működő komparátor*

A párhuzamosság miatt a műveletvégrehajtás – a soros változathoz képest – gyorsabb lesz, viszont a komparálást végző hálózatrész bonyolultabbá válik. A megváltozott viszonyok szemléltetése céljából vizsgáljuk meg a 10-86a. ábrán látható elvi vázlatot, melyen egy *2 bites párhuzamos* komparátor elvi vázlata szerepel, azaz az *A,* ill. *B* kódszavak 2-2 bit hosszúságúak. A logikai hálózat működését jellemző függvények felírásánál (a 2 bit miatt) több feltételt is figyelembe kell vennünk a korábbi *egybites* esethez viszonyítva:

Így előzetesen kimondható, hogy $A = B$ csak akkor állhat fenn, ha minden egyes bit egyenlőségének ÉS kapcsolata teljesül, továbbá az $A < B$, ill. $A > B$ esetnél, helyértékenként felülről indulva, mindig az *először* megjelenő $A_i < B_i$ vagy $A_i > B_i$ egyenlőtlenség határozza meg a kérdéses relációt.

Fenti feltételek alapján a kimeneti függvények 2-bites esetünkre a 10-86b. ábra szerint alakulnak, melyek alapján a 10-86c. hálózat rajzolható fel.



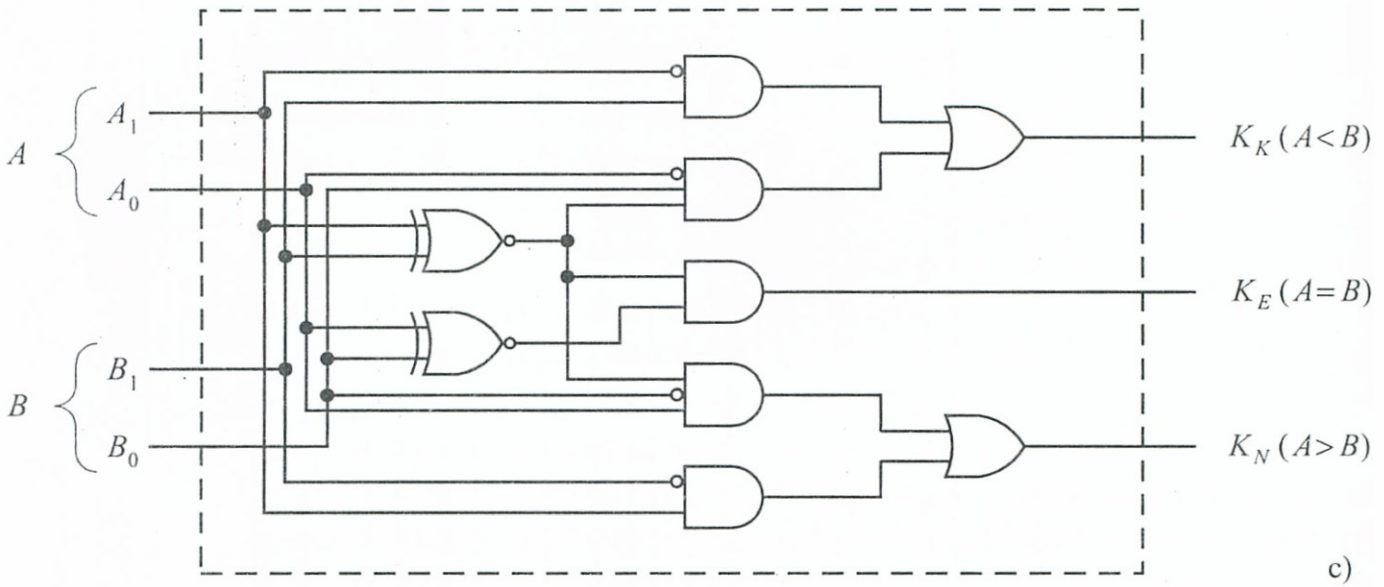
a)

$$K_K = \overline{A}_1 \cdot B_1 + (A_1 \equiv B_1) \cdot \overline{A}_0 \cdot B_0$$

$$K_N = A_1 \cdot \overline{B}_1 + (A_1 \equiv B_1) \cdot A_0 \cdot \overline{B}_0$$

$$K_E = (A_1 \equiv B_1) \cdot (A_0 \equiv B_0)$$

b)

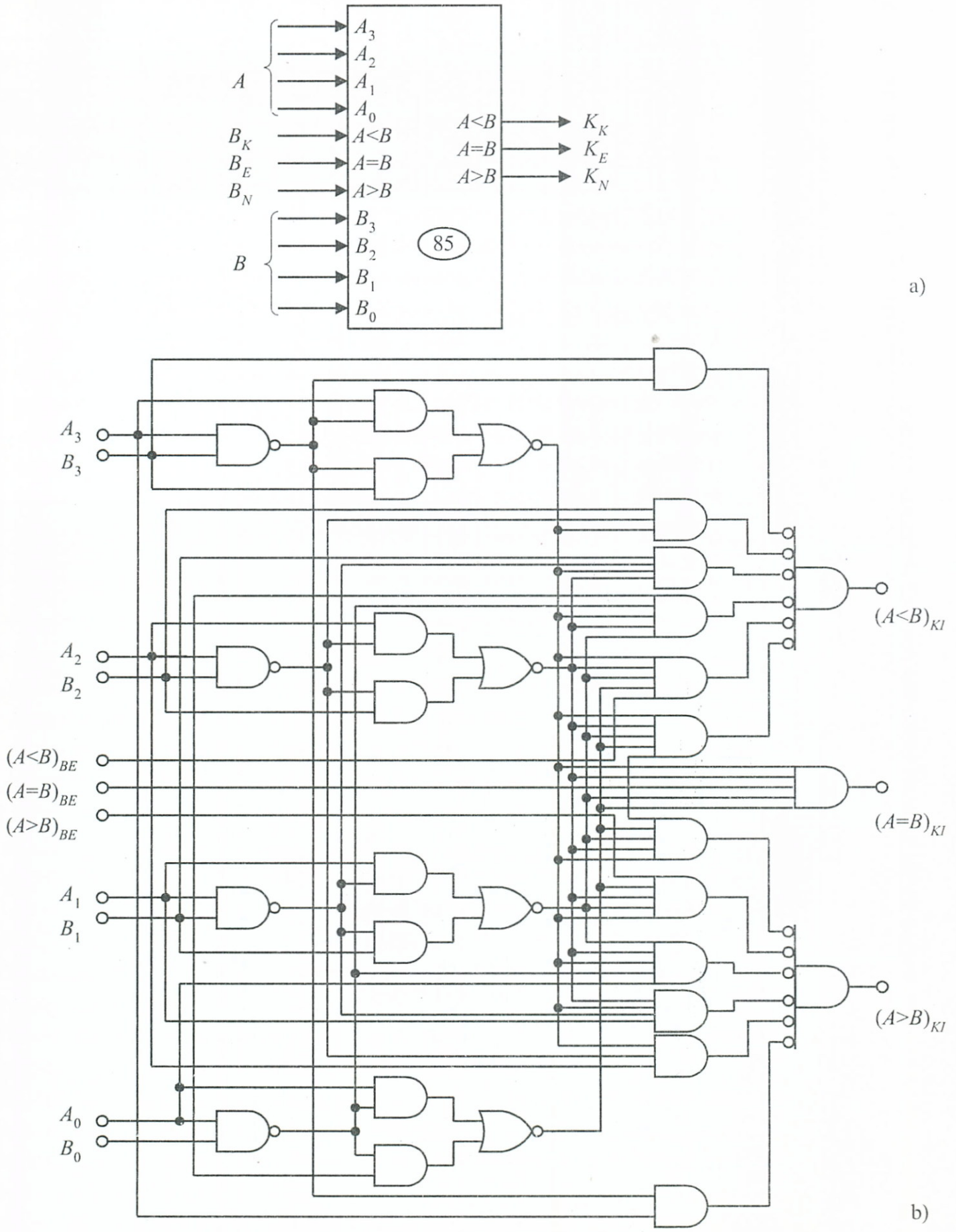


c)

10–86. ábra *2 bit-es párhuzamos komparátor*

10.7.4. Komparátorok összekapcsolása

A gyártó cégek által előállított, egy IC tokba beépített komparátorok kapacitása behatárolt, ezért összetettebb összehasonlítási feladatoknál szükségessé válik a több IC tok szisztematikus összekapcsolásán alapuló bővítés. Ennek érdekében – hasonlóan a más funkcionális egységeknél tapasztaltakhoz – a gyártók további kiegészítő csatlakozásokkal látják el az áramköri egységeket. Vizsgáljunk meg egy ilyen alapon kialakított hálózatot.



10–87. ábra *4 bit-es párhuzamos komparátor*

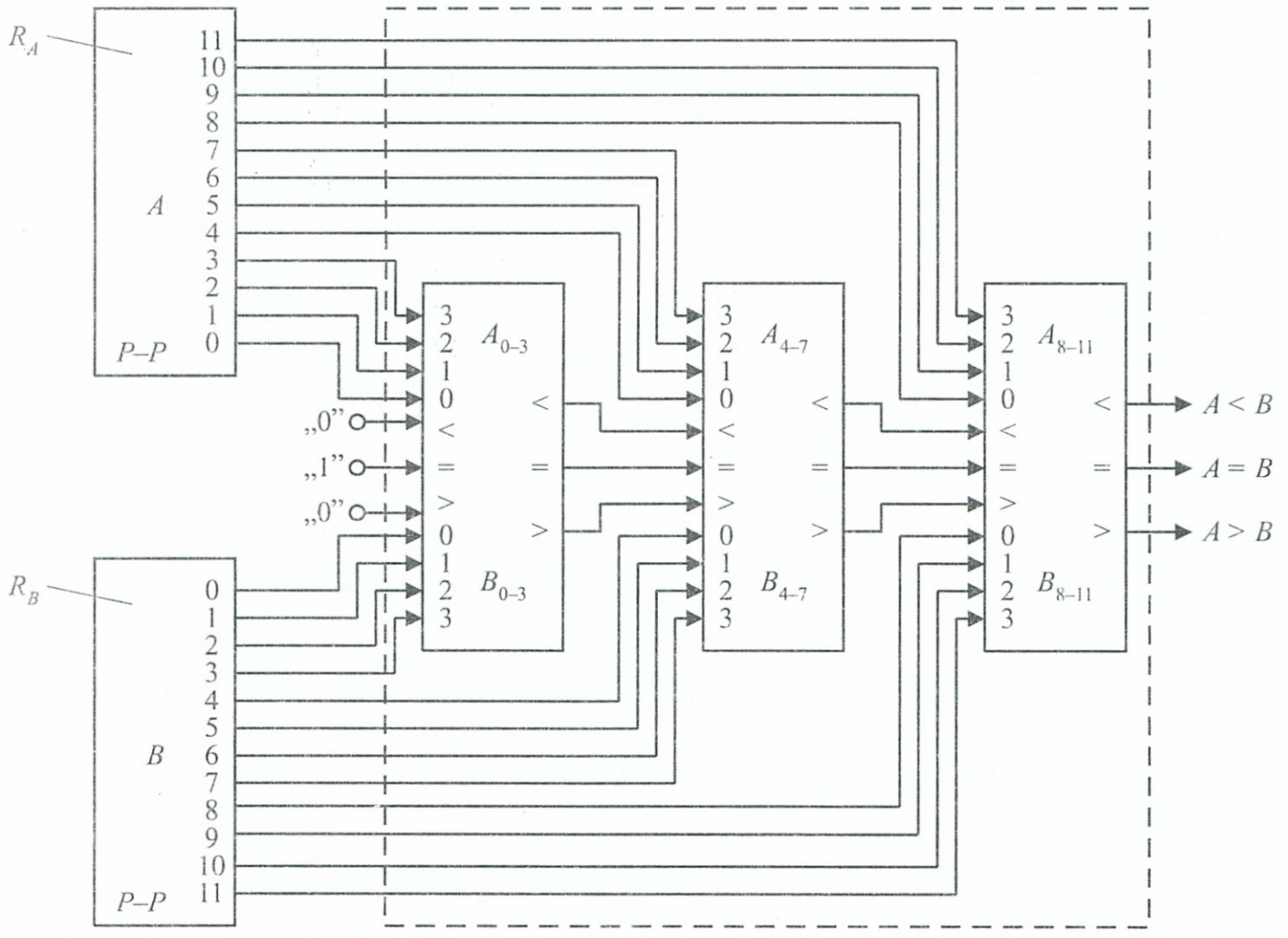
A 10-87a. ábrán egy bővítési feladatok megoldására is alkalmasan kialakított 4 bites komparátor elvi vázlata látható, melyen az

$$A\colon\ A_3A_2A_1A_0$$

$$B\colon\ B_3B_2B_1B_0$$

összehasonlítandó bemenetek mellett $(A < B)_{BE}$, $(A = B)_{BE}$, $(A > B)_{BE}$ jelű bővítési célzatú bemeneteket is találunk. Ezek a bemenetek az alacsonyabb helyérték-pozíciókon elvégzett összehasonlítások eredményeit képviselik, és továbbítják „felfelé".

Ha a K_K, K_E, K_N működést jellemző függvényeket itt, a 4 bites esetre is felírnánk, közel hasonló jellegű alakokat kapnánk ($A_3 \ldots A_0$, $B_3 \ldots B_0$ változókra), mint amilyeneket a 2 bites esetre a 10-86b. ábrán már felírtunk. A különbséget az új $(A < B)_{BE}$, $(A = B)_{BE}$, és $(A > B)_{BE}$ bemenetek jelentik, melyeket úgy kell a függvényekben tekintenünk, mint az A_0, B_0 biteknél eggyel alacsonyabb helyértéket. A függvények felírását ezúttal nem részletezve a 10-87b. ábrán felrajzoltuk



10–88. ábra *12 bit-es párhuzamos komparátor kialakítása 4 bit-es modulokból*

egy gyári kapcsolás teljes logikai vázlatát, melynél a 10-86c. ábrához viszonyítva néhány célszerű algebrai átalakítás is felhasználásra került.

Végezetül szemléltessük az eddig elmondottakat egy bővítést ténylegesen igénylő példán keresztül:

Alakítsunk ki 12 bites párhuzamos elven felépülő komparátort az előzőleg megismert 4 bites modul-elem felhasználásával.

A megoldás (a 12 bemenet miatt) 3 modul-elem összekapcsolását igényli. Tételezzük fel, hogy az összehasonlítandó 12 bites A, B kódokat R_A, R_B P-P típusú regiszterekben tároljuk, és innen küldjük őket a 10-88. ábrán felrajzolt módon bővített komparátorba, melynek eredő kimenetein kapjuk az összehasonlítás eredményét.

10.8. Memóriák

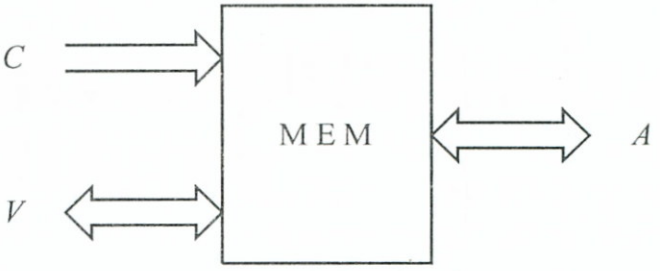
10.8.1. Elvi működés, főbb jellemzők, csoportosítás

A memóriák nagyobb mennyiségű információ átmeneti vagy tartós tárolására szolgáló egységek. Néhány bit-, vagy kisebb bitcsoport (ún. „szó") tárolására alkalmas áramkörökkel (pl. tárolóelemek, regiszterek) az előző fejezetekben már foglalkoztunk. A nagyobb tárolókapacitású memóriák – lényegében – e tárolóelemekből – regiszterekből épülnek fel, de az összetevők nagy száma, a tárolt információ célszerű kezelése az együttműködés magasabb szervezeti formáját és ennek kapcsán különféle kisegítő-vezérlő áramköri egységek alkalmazását vonja maga után.

A memóriák egy jellegzetes csoportját (ROM) a „Kapcsoló-mátrixok" tanulmányozásakor (6-41. ábra) már érintettük, most az ott megismerteket beilleszthetjük a memóriákkal kapcsolatosan itt elmondandók körébe.

A memóriák működését első közelítésben a 10-89. ábra sémáján tanulmányozhatjuk.

A C CIM bemenetre érkező információval jelöljük ki az A ADAT csatlakozásra érkező (beírással tárolandó), vagy innen távozó (kiolvasandó) jelek memórián belüli helyét. A V VEZÉRLÉS csatlakozásokon keresztül adhatunk utasításokat (pl. beírás, kiolvasás stb.) vagy nyerhetünk információkat (pl. foglaltság stb.) a memória működésére vonatkozóan.



10–89. ábra *MEMÓRIA elvi vázlata*

A memóriák gyors fejlődése elsősorban a számítógépek által felvetett igényeknek a következménye, melyeknek megfelelően rendkívül sokféle változatuk alakult ki. Ezek részletesebb ismertetése a számítástecnika témakörébe nyúlik át, ezért jelen könyvben a memóriákat főként általános digitális rendszertechnikai szemléletmódban tárgyaljuk és csak az elektronikus felépítésű memóriákra szorítkozunk.

A memóriákat sokféle szempont szerint csoportosíthatjuk, foglaljuk össze ezek közül a néhány leglényegesebbet.

a) *Információtárolás rendszere*
   a1. Változtatható tartalom
      – gyors
      – lassú
   a2. Rögzített tartalom
      – gyártáskor
      – felhasználó által
b) *Tárolás időbeli módja*
   b1. Statikus
   b2. Dinamikus
   b3. Kvázisztatikus
c) *Címzés alaprendszere*
   c1. Tetszőleges v. közvetlen eléréssel
   c2. Asszociatív eléréssel
   c3. Soros eléréssel
d) *Gyártástechnológia alapján*
   d1. Bipoláris
   d2. MOS
e) *Adattárolás szervezési változatai*
   e1. Bit szervezés
   e2. Szó szervezés

A memóriákkal kapcsolatosan számos elnevezés és fogalom szerepel az irodalomban, melyek közül a gyakoribbakat az alábbiakban foglaljuk össze:

- RWM (Read Write Memory): olvasható és írható (változtatható tartalmú) memória.
- RAM (Random Access Memory): tetszőleges elérésű tároló, többnyire gyors változtatható tartalommal.
- ROM (Read Only Memory): csak kiolvasható memória rögzített tartalommal, mely többnyire ugyancsak tetszőleges elérésű.
- PROM (Programmable Read Only Memory): Programozható (rendszerint egy alkalommal) ROM.
- EPROM (Electrically Programmable Read Only Memory): Elektromosan programozható ROM.
- EEROM (Electrically Erasable ROM): Elektromosan törölhető ROM.
- EAROM (Electrically Alterable ROM): Elektromosan módosítható tartalmú ROM. Felhasználható például lassan változtatható tartalmú memóriaként.
- REPROM (Reprogrammable ROM): Újra (többször is) programozható ROM.
- CAM (Content Addressable Memory): Tartalom alapján címezhető, más néven, *asszociatív* elérésű memória.
- SAM (Serial Access Memory): Soros elérésű memória, legtöbbször változtatható tartalommal.

A *tárolás időbeli módja,* mint memória jellemző, elsősorban az RWM (olvasható és írható) memóriákkal összefüggésben vetődik fel, és szoros kapcsolatban áll a gyártástechnológiával.

Bipoláris felépítés esetén főként a *sztatikus* elv érvényesül, mivel a memóriát felépítő, bistabil tárolóelem-cellák a beírt információt mindaddig megőrzik, míg ezt újabb beírással nem módosítjuk.

MOS felépítésnél mind a sztatikus, mind a dinamikus elv érvényesül. A *sztatikus* elven működő tárolóelem-cellák elvi felépítése hasonló a bipoláris rendszernél használtakhoz, csak MOS tranzisztorokra alkalmazva. A *dinamikus* elven működő tárolóelem-cellák a MOS tranzisztorok saját kapacitásainak töltéstároló-képességét hasznosítják (lásd: 5.3.3. pont: dinamikus- és előtöltéses inverterek: 5-34. és 5-35. ábrák). Mivel ezek a kapacitások csak rövid ideig képesek *dinamikusan* tárolni a „beléjük írt" információt, ezért tartalmuk periódikus felfrissítése szükséges, mielőtt töltésüket elveszítenék. Ez

a magyarázata, hogy a dinamikus memóriák blokkvázlata kiegészül egy ún. frissítő egységgel, mely a fent említett kapacitás-töltés reprodukálásáról gondoskodik.

Az ún. *kvázisztatikus* (pszeudo-sztatikus) memóriáknál a dinamikus és sztatikus elvek előnyös kombinációját alkalmazzák.

A *memóriák specifikálásakor* szereplő jellemzők közül megemlítjük a:

memória-kapacitást, mely a memóriában tárolható információ mennyiségét jellemzi. Ezt vagy *bit-kapacitás*ban, azaz a memóriában tárolható bitek számában, vagy *szó-kapacitás*ban, azaz a memóriában tárolható (rendszerint szokványos hosszúságú: 8 bit, 16 bit, 32 bit, 64 bit) összetartozó bit-csoportok, más néven *szó*-k maximális számában adják meg. Ez utóbbi esetben gyakori elnevezések: 1 byte = 8 bit, 1 szó = dupla byte = 16 bit, dupla szó = 32 bit.

A memóriakapacitás kapcsán szereplő további mennyiségek: 1 kbit = 1024 bit (2^{10}), 1 kbyte = 1024 byte, 64 kbyte = 65536 byte = 524288 bit, 1 Mbit = 1.048576 bit stb.

Például: egy 4 kbyte-os ROM tárolóról részletesebben a következőket mondhatjuk el:

szó-hossz: 8 bit (ha egy byte alatt 8 bit-es szót értünk)

szókapacítás. 4096 byte (mivel 1 kbyte ~ 1024 byte)

bitkapacítás: 4096×8 = 32 768 bit ~ 32 kbit

A byte-ban, szó-ban, vagy bit-ben kifejezett értékek, az információnak a memóriában történő elhelyezkedésének és elérésének *szervezésére* is utalnak.

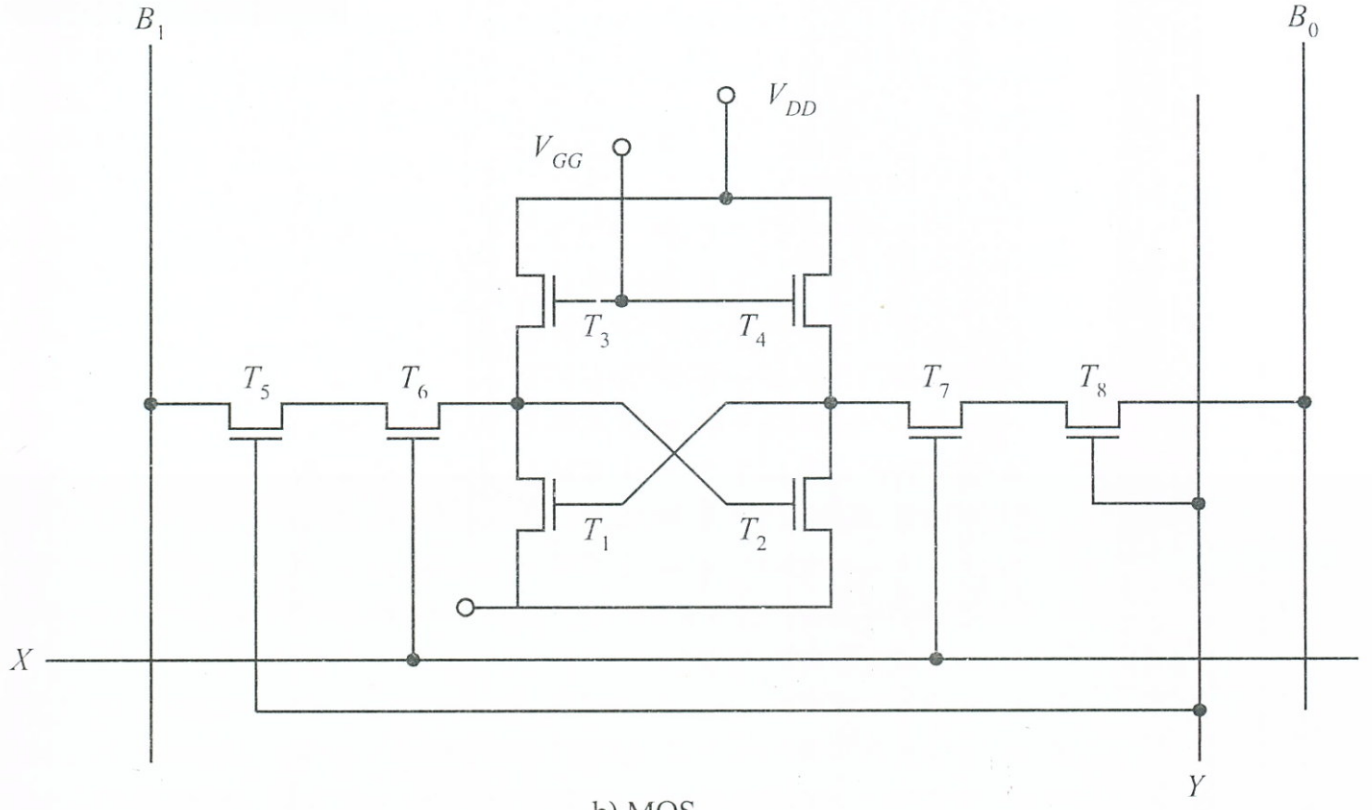
*elérési idő*vel jellemezzük a memóriák működési sebességét, mely azt az időtartamot jelzi, mely a CIM kiadása és a végleges (stabilizálódott) ADAT íráskor vagy olvasáskor észlelt megérkezése között eltelik.

10.8.2. Memória-cellák felépítése és működése

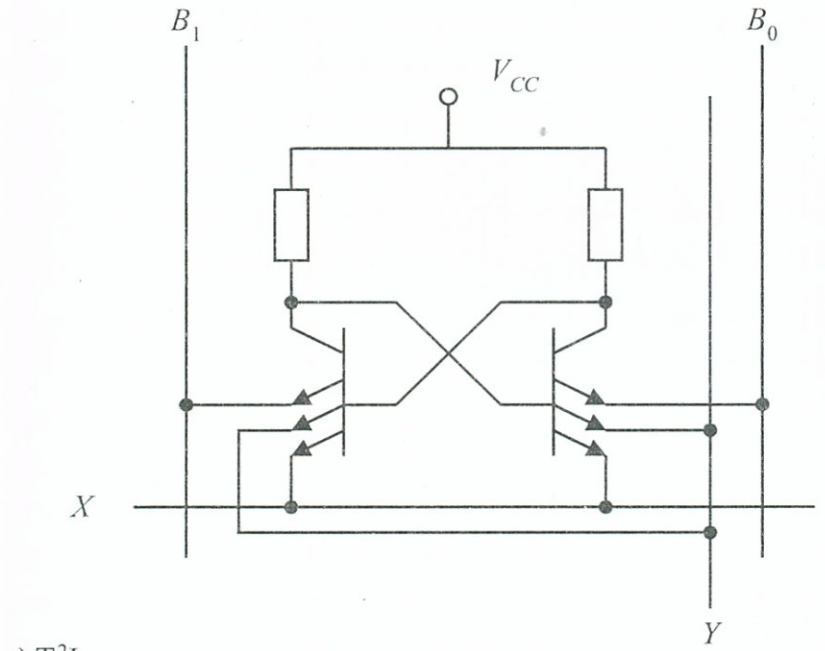
A memória-cellák 1 bit tárolását elvégző tárolóelemek, melyekből a memóriák „emlékező egysége" felépül.

ROM jellegű memóriáknál – mint az 6. fejezet 6.4.1. és 6.4.2. pontjában már tárgyaltuk – a kapcsoló-elemek irreverzibilisen vagy reverzibilisen programozva fixre beállított TTL, esetleg MOS, ill. FAMOS áramkörök, melyekből szervezett *kapcsolómátrix* alkotja a memória emlékező-egységét.

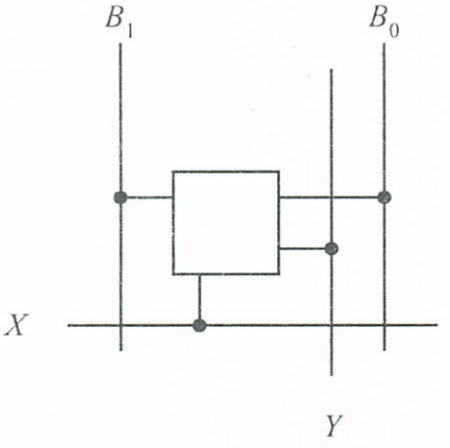
RWM jellegű memóriáknál, a sztatikus és dinamikus tárolási módnak megfelelően különbséget kell tennünk sztatikus és dinamikus működésű írható és olvasható cellák között.



b) MOS



a) T^2L



c)

10–90. ábra *Sztatikus memória-cella T^2L, MOS változatban és elvi jelölése*

a) Sztatikus memória-cellák

Működési elvük az 5. fejezetben bevezetett tárolóelemek működésével rokon, azzal a módosítással, hogy az emlékező-egységbe jól szervezetten (pl. mátrix elrendezésben) beépíthetők legyenek.

– *TTL felépítésű,* sztatikus tároló cellára mutat példát a 10-90a. ábra, mely két keresztbe visszacsatolt 3 emitteres, bipoláris tranzisztorból épül fel. A 3 emitter valójában egy ÉS kapcsolást realizál, mely az *adat-beíró-kiolvasó* vezetékek, valamint az X és Y mátrixpont *kijelölő cím-vezetékek* között létesít logikai kapcsolatot. *Tárolási* üzemmódban az X és Y vezetékek alacsony szintűek, ezért a tranzisztorok árama nem a B_1–B_0 irányba folyik. *Olvasási* üzemmódban X és Y magas szintre kerülnek, így a kiválasztott cellára a B_1–B_0 adatvezetékek aktivizálódnak és logikai állapotuk leolvasható lesz. *Írási* üzemmódban ugyancsak X–Y magasra állításával aktivizáljuk a cellát, majd a B_1B_0 vezetékpárra olyan polaritású jelet küldünk, amelyik oldalra a tárolócellát be akarjuk billenteni.

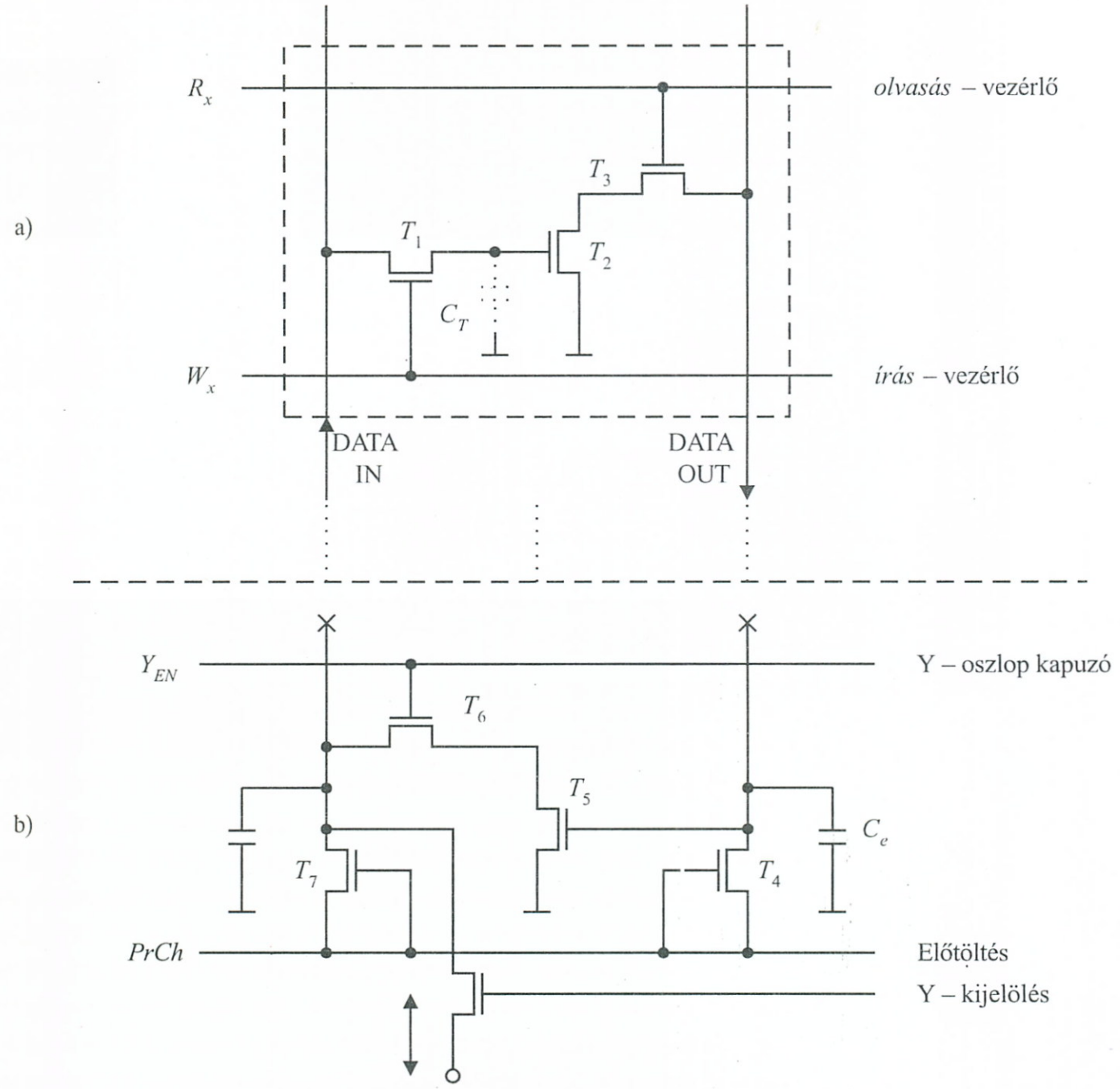
MOS felépítésű sztatikus, ún. 8 tranzisztoros cellára láthatunk példát a 10-90b. ábrán. A cella hasonló szervezésű, mint az előző TTL változat. Az emlékező-funkciót a két keresztbecsatolt T_1–T_2 MOS tranzisztor biztosítja passzívált T_3–T_4 „munka-ellenállásaival", míg a bal, ill. jobb oldali *adat-beíró-kiolvasó* vezetékek és az X, ill. Y *cím-vezetékek* logikai kapcsolatát a T_5–T_6, ill. T_7–T_8 soros ÉS kapcsolók biztosítják.

Mind a két bemutatott változat működése a korábban már megismert S–R tárolóelem működésén alapul, és elvi jelölésüket a 10-90c. ábrán láthatjuk. Természetesen fenti példákon túlmenően a sztatikus cellák számos más realizációs változata is kialakult.

b) Dinamikus memória-cellák

A dinamikus cellák működése a MOS elemek 5.3.3. pontban tárgyalt dinamikus tulajdonságain alapul (5-34., 5-35. ábrák).

– *3 tranzisztoros* dinamikus cellára mutat példát a 10-91a. ábra, melynél a tárolási funkciót nem az eddig „megszokott" visszacsatolással érjük el, hanem az ún. tároló tranzisztor (T_2) saját (C_T) kapacitására bízzuk. *Beíráskor* az *írás-vezérlő* jelével nyitjuk a T_1 kapuzó tranzisztort és beírandó 1 esetén feltöltjük a C_T kapacitást, mely ezt a töltést (bizonyos ideig) tárolja. *Kiolvasáskor* az olvasás-vezérlő jellel nyitjuk a T_3 kapcsolót. Ha C_T = 1-ben volt, akkor a T_2 rövidzárként viselkedik és a kimeneti pontra kapcsolódó – előzőleg „előtöltéssel" feltöltött – C_e kapacitást kisüti, így az olvasó adatvezeték 0 értékre



10–91. ábra *3-tranzisztoros dinamikus tárolócella (a), kiegészítő „oszlop-áramkörrel" (b)*

kerül. Ha C_T = 0-ban volt, akkor a T_2 szakadásként viselkedik és C_e nem tud kisülni, így az olvasó adatvezeték 1-ben marad. Látható, hogy a cella kiolvasáskor invertál, ezért a kiértékeléskor ezt figyelembe kell venni. Megjegyezzük, hogy a MOS-oknál gyakran előforduló negatív feszültségek miatt az áramkör negatív logikában is működhet.

A C_T kapacitásnak működés során bekövetkező kisülése miatt, ennek töltését automatikusan regenerálni („frissíteni") kell. Az erre, valamint az előtöltésre, továbbá az Input/Output csatlakoztatásra szolgáló fokozatot a 10-91b. jelű ábrarész tartalmazza. Itt a T_5–T_6 tranzisztorok egy inverter kapcsoló részét képezik és a destruktívan kiolvasott jeleket negáltan visszajuttatják az íróoldali adatvezetékre, így a kiolvasást mindig frissítés követheti. Természetesen frissítésre a nem kiolvasásból eredő lassú töltéskisülés miatt is szükség van, ezt a dinamikus memóriáknál külön *frissítési ciklusok* beiktatásával oldják meg.

A dinamikus működésű cellák áramköri kialakítására az irodalomban nagyszámú megoldás szerepel, így pl. megoldott az *1 tranzisztoros* dinamikus tárolás is, mely az IC lapkákon további helyfoglalás-csökkenést, közvetve az egy lapkára koncentrálható memória-kapacitás növelését eredményezi. Az áramköri megoldások részletesebb ismertetésével jelen könyvben nem foglalkozunk.

10.8.3. Memória-egységek felépítése

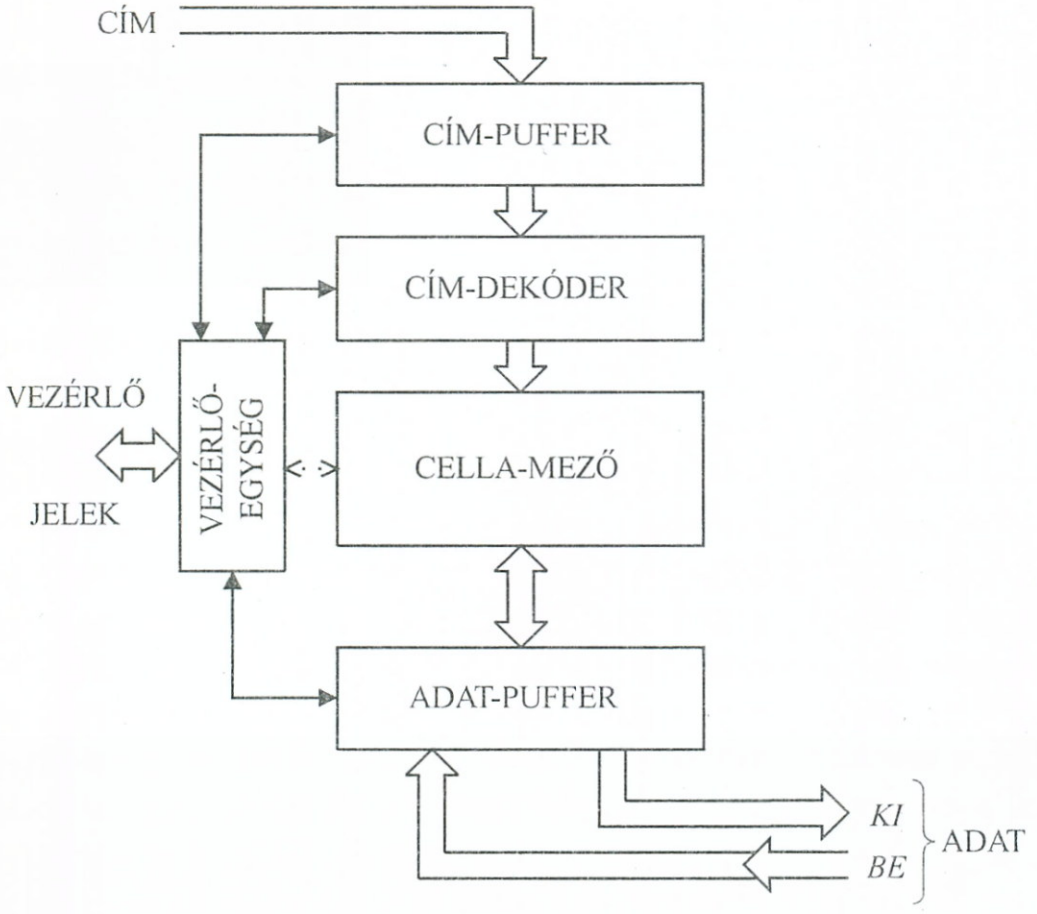
A memóriák több részegységből állnak.

– A *cella mező* (vagy cella mátrix) az információ-tárolási feladatokat látja el, és szerkezetileg, egységnyi információt tároló elemi cellák szervezett együtteséből épül fel. A cellák különféle szervezési struktúrákban kapcsolódhatnak egymáshoz, melyekre később még részletesebben kitérünk. A cella-mező és a külvilág kapcsolatát különféle kisegítő áramkörök biztosítják (10-92. ábra).

– A *Címzési információk* rendszerint egy *cím-pufferbe* (cím regiszterbe) érkeznek. A cím puffer és a cella-mező címző csatlakozásai között általában egy *cím dekóder* helyezkedik el, mely lehetővé teszi, hogy a cím-bemenetek számát csökkentsük, és „n" bemenő vezetékkel $\leq 2^n$ helyen végezhessünk kijelölést.

– Az *adat információk* gyakran egy *adat pufferen* keresztül kerülnek kapcsolatba a cella-mezővel. Ez a kapcsolat RWM memóriák esetén BE/KI irányú, míg a ROM családnál csak KI-felé irányul.
A BE/KI kapcsolatnál gyakori, hogy külön-külön adatvonalak szolgálnak a BE, ill. KI irányú jelek átvitelére, de előfordul, hogy kétirányú vonalakon áramlik az információ.

– A *vezérlési információk* feldolgozását egy *vezérlő áramkör* látja el. Ez fogadja a memóriához érkező (→) különféle működtetési információkat, illetve tájékoztatást küld a csatlakozó digitális rendszer felé (←) a memória állapotáról. Fentiek értelmében kapuzási, enge-



10–92. ábra *Memória felépítése egységekből*

délyezési, sorbaállítási feladatokat lát el, és dinamikus memória esetén részt vesz a frissítési folyamatban is. A digitális rendszerrel fenntartott kapcsolat vezérlési információból az alábbiakban felsoroltunk néhányat szemléltető célzattal. A felsorolásban zárójelben feltüntettük a gyakrabban alkalmazott rövidítő jelölést, valamint a memória oldaláról értelmezett irányítottságot is.

→ *kiválasztás* (CS), ezzel jelöljük ki az aktivizálandó memória áramkört hordozó IC lapkát (Chip Select)
→ *engedélyezés* (ME), ezzel aktivizálhatjuk a memóriát
→ *olvasás* (RD), innen tudja meg a memória, hogy adatot akarunk *kiolvasni* belőle
→ *írás* (WR), innen tudja meg a memória, hogy adatot akarunk *beírni*
→ *érvényes adat* (DA), mely beírásra készen áll a memória részére
→ *érvényes cím* (VA), a cím vezetékeken
→ *külső frissítés* (RFRSH), dinamikus memóriáknál
← *érvényes kiolvasott adat* (DR) van az adatvezetékeken
← *foglaltsági jelzés* (BUSY), mely arról tudósít, hogy a memória pillanatnyilag nem alkalmas a külső kommunikációra.

10.8.4. Memóriák szervezése

A memóriák emlékezési feladatát ellátó cella-mező kialakítása többféle szervezési változatban képzelhető el.

a) *Bit szervezésű* memóriák címzésekor az egyes tároló-cellák egyedenként (bitenként) kerülnek kijelölésre. Ha az elemi tároló-cellákat egy két-dimenziós mátrix struktúrában elhelyezkedőknek képzeljük, akkor a címzés egy „*ij*" paraméterű mátrix-elem megjelölését jelenti. Egy bit-szervezésű 16 bites R/W memória-mátrix látható a 10-93a. ábrán.

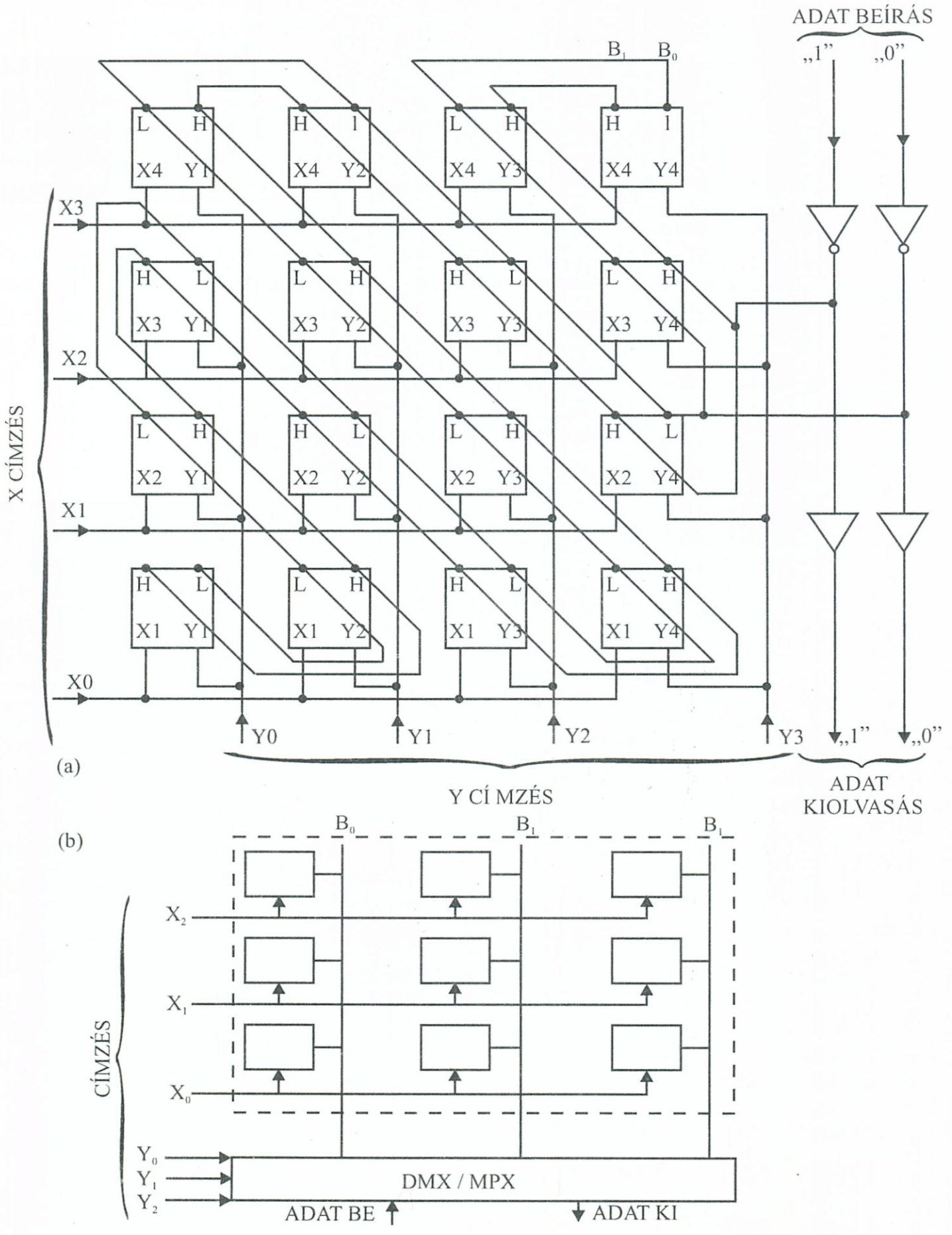
Itt a memória-cellák kapcsolási rajza megegyezik a 10-90a. ábrabeli TTL celláéval, mivel mind *X*, mind *Y* irányú kijelölésre szükség van. Az X_i–Y_j cella kijelölésekor csak az ij keresztpontban teljesül a két emitter ÉS típusú kapcsolata, így kizárólag ez az egy cella aktivizálódik. *Kiolvasáskor* ennek a cellának B_0–B_1 kimeneti jelei jelennek meg a memória *adatkimenetein. Beíráskor* pedig a *adatbemenetekre* adott jelek csak erre az aktivizált cellára lesznek hatásosak, annak ellenére, hogy valamennyi cella rákapcsolódik a be-, ill. kimeneti vezetékekre.

Mint látható, ez a memória típus „meztelen", azaz sem puffer, sem dekóder áramkörök nincsenek beépítve és vezérlésről is csak olyan mértékben beszélhetünk, hogy külön adatbeíró és külön adatkiolvasó csatlakozásokkal rendelkezik. A kiolvasó erősítő olyan felépítésű, hogy lehetővé teszi a bővítést is.

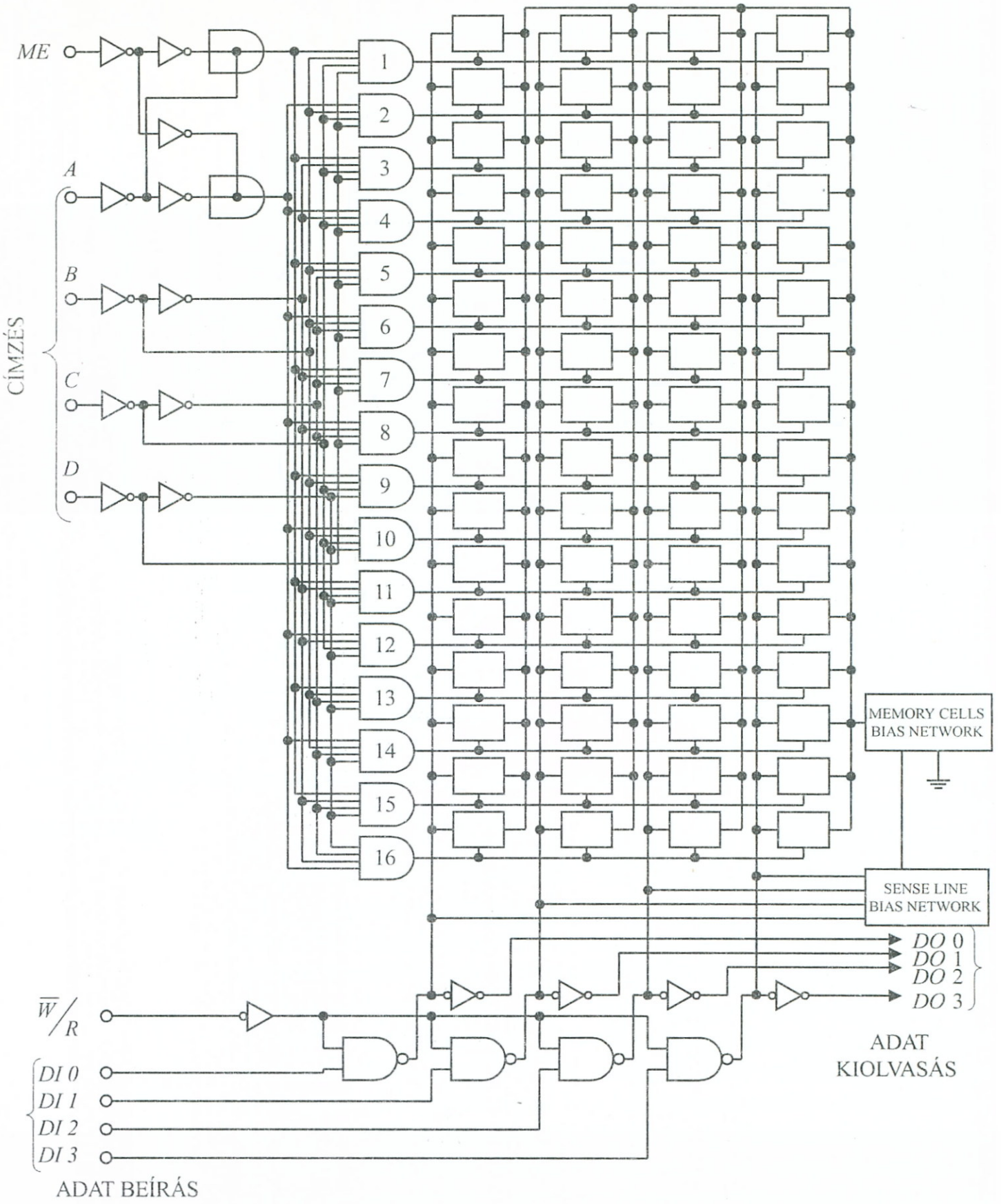
A 10-93a. ábra valójában csak egy „memória sík"-nak tekinthető, egy-egy sík a szavaknak csak egy-egy bitjét tartalmazza, emiatt egy komplett memória annyi „sík"-ból áll, ahány bites szavakat tartalmaz. Valamely szó megcímzése a kívánt X_i–Y_j kijelölésével történik, melynek hatására *valamennyi* összetartozó síknál *ugyanaz* a keresztpont aktivizálódik. Az így kijelölt keresztpontok bitjei együtt alkotják a megcímzett *szó*-t.

Elsősorban technológiai okokból a bit-szervezés struktúráját célszerű volt módosítani. Egy ilyen, ún. *módosított bit-szervezésű* elrendezés elvét mutatja a 10-93b. ábra. Itt a kiolvasás hasonlít a szó-szervezésű esethez, mivel a sorkijelölés X_i-vel történik, viszont az oszlopokra elhelyezett MPX/DMX áramkör csak annak az *oszlop-bit*-nek értékét engedi aktualizálódni, melyet az Yj kijelöl. Beíráskor a DMX, olvasáskor az MPX működik.

b) *Szó szervezés*nél az elemi tároló-cellákat ugyancsak két-dimenziós mátrix struktúrában elhelyezkedőnek képzelve, *címzéskor* egy-



10 -93. ábra. *(a) bit szervezésű memória - sík, (b) módosított bit - szervezés*



10–94. ábra *SZÓ-szervezésű, 4 bit szóhosszú, 16 szavas R/W memória*

szerre egy *teljes mátrix sor* kijelölését végezzük el. Egy-egy sor alkot egy *szó*-t, melynek a bitjeit tároló cellasort egyszerre írjuk be RWM tároló esetén, illetve olvassuk ki mind RWM, mind ROM tárolónál.

Szemléltetésül egy szó-szervezésű, 4 bit szóhosszúságú, 16 szókapacitású ($16 \times 4 = 64$ bit kapacitású), olvasható/írható memóriát rajzoltunk fel a 10-94. ábrán.

A memória RAM típusú és elkülönített ADAT BE, ill. ADAT KI csatlakozásokkal rendelkezik, melyeket a (WRITE/READ) vezérlő bemenettel választunk ki, ahol „0"-írást, „1"-olvasást jelent. A memória külön adat-, ill. címpufferrel nem rendelkezik. A CÍMZÉS bemeneteket a 16 ÉS kapuból álló CÍM-DEKÓDER „sokszorozza" meg, melynek letiltását egy HUZALOZOTT kapcsolással csatlakoztatott *ME* (MEMORY ENABLE) bemenettel lehet vezérelni. A memória-cellák hasonló felépítésűek, mint a 10-90a. ábrán bemutatott bipoláris cella, azzal a különbséggel, hogy a cella keresztbecsatolt tranzisztorai itt csak két emitteresek, mivel az Y címzésre – a *szó-szervezés* miatt – nincs szükség. A dobozzal jelölt „BIAS" nevet hordozó hálózatrészek áramköri-működési feladatokat látnak el. Az adatkimenetek OPEN COLLECTOR-os megoldásúak az esetleges *memória-bővítés* érdekében. (lásd. később)

Az előbbinél nagyobb kapacitású közös adat be/kimenetű, szószervezésű R/W memória blokkvázlatát rajzoltuk fel a 10.95a. ábrán.

Az $\overline{\text{ME}}$ aktív nullás vezérlőbemenet itt is az *egész egység* engedélyezését vezérli és több modulból álló összetett memória-egység esetén csak azok a modulok működnek, melyek az adatforgalom szempontjából éppen aktuálisak. Ez a szervezési mód azért is nagyon előnyös, mert a teljes egység energia-fogyasztása és hődisszipációja minimálisra csökkenthető.

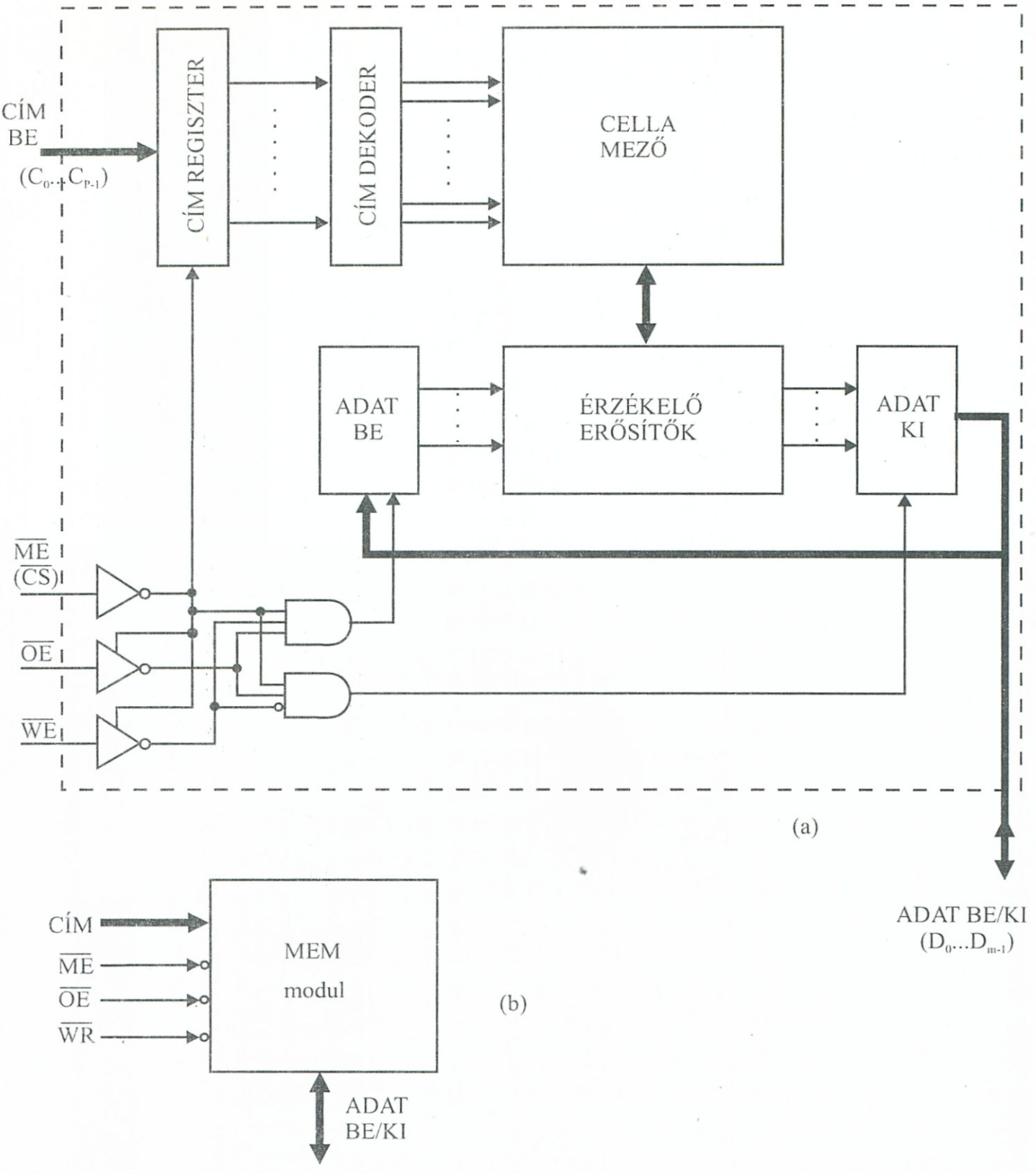
Ennél a változatnál az írás és olvasás vezérlése két elkülönített bemenettel történik.

A $\overline{\text{WE}}$ =WRITE ENABLE) aktív nullás *beírás-engedélyezés* is csak az aktív $\overline{\text{ME}}$ alatt történhet.

Az $\overline{\text{OE}}$ (OUTPUT ENABLE) ugyancsak aktív nullás *olvasás-engedélyezés* szintén függ az $\overline{\text{ME}}$-től és olvasás esetén működteti a háromállapotú (THREE STATE) kimeneti meghajtókat, ami különösen előnyös az elterjedt busz-szervezésű digitális rendszereknél.

A memória-modul egyszerűsített jelölését 10-95b. ábrán rajzoltuk fel.

c) *Soros-üzemű cella-szervezés*nél (SAM) a cellák relatív elrendeződésének szemléltetésére a mátrix struktúra már nem alkalmas. Itt a 10-96. ábra elvi vázlatából kell kiindulnunk, melynél a tároló-mező



10 - 95. ábra. *R/W közös adat ki/be-menetű memória belső blokkvázlata*

celláiban tárolt információt egy visszacsatoló-hurok mentén folyamatosan áramlónak képzeljük. Mivel így a tárolt információ állandó mozgásban van, ezért e memória típust *dinamikus memóriának* is nevezik. A cella-mező szervezésénél még megkülönböztethetünk:

10–96. ábra *Soros-üzemű cella-szervezés*

– szó-soros
– bit-soros

változatokat, ahol az első esetben a szavak bitjei egymással *párhuzamosan,* egyidejűleg küldik át bit-jeiket a következő cellacsoportba, míg a második esetben a szavak bitjei is sorosan, egymás után láncolódnak és így illeszkednek bele az információ folytonos áramlásába.

Fenti tárolási elv nemcsak az elektronikus memóriák területén kerül felhasználásra, ennél általánosabb érvényű. Például, így működnek a forgó mágneses diszkek, vagy a lézer-letapogatású forgólemezes memóriák is.

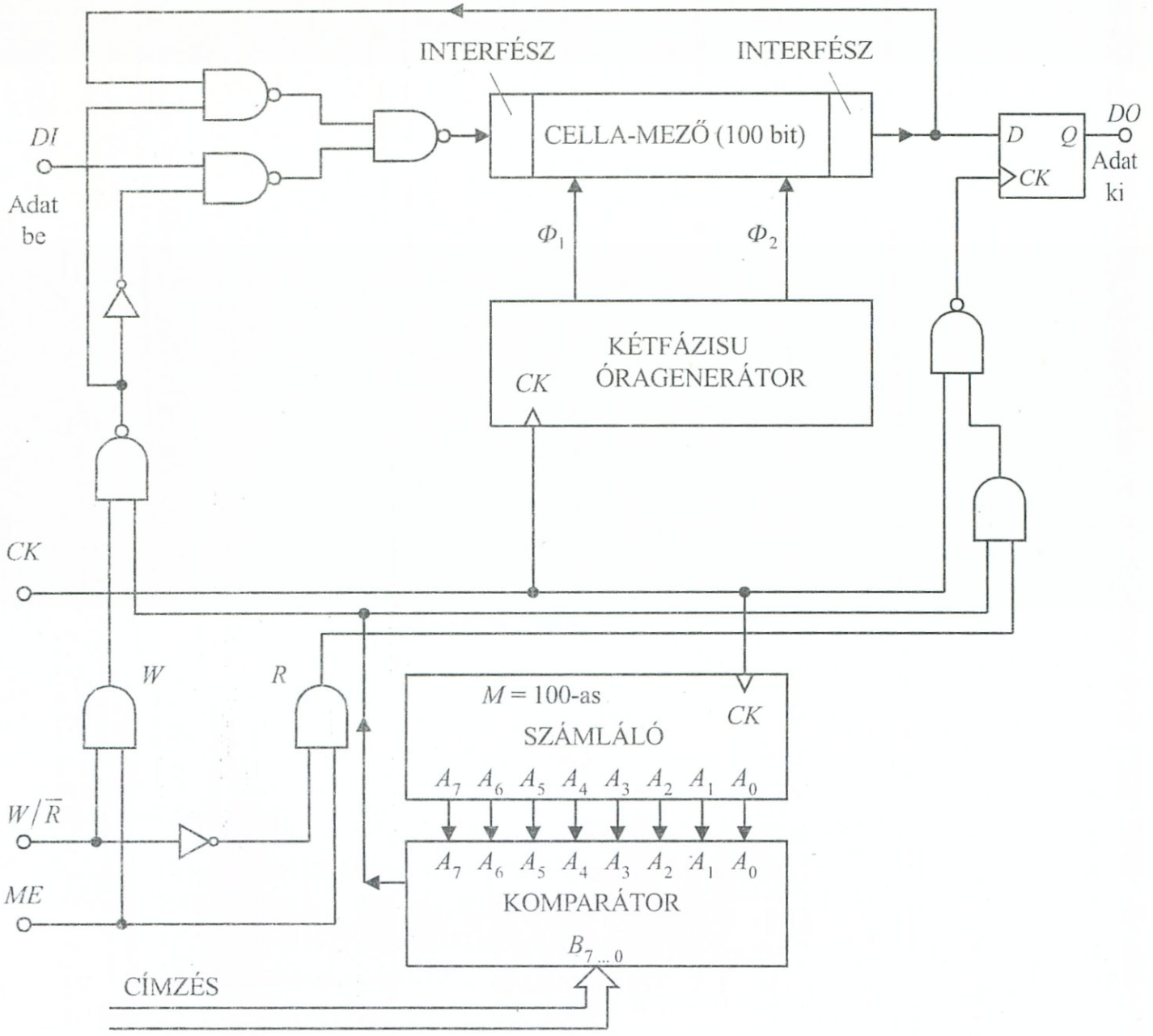
Elektronikus memóriáknál történő felhasználásukkor a cella-mezőt léptető-regiszterek alkotják, túlnyomórészt MOS realizációban.

A soros-üzemű memóriák címzése, adatbeírása, adatkiolvasása, tárolása különféle üzemmód állapotokat igényel és eltér az eddig bemutatott esetektől.

A 10-97. ábrán egy gyárilag előállított MOS és bipoláris elemekből felépülő soros-üzemű memóriát rajzoltunk fel szemléltető célzattal.

A memória-cella mezejét egy dinamikus léptető-regiszter képezi, melynek bit-soros kapacitása 100 bit. Ez a regiszter interfész elemeken keresztül csatlakozik a TTL kiszolgáló rendszerhez. A 100 bit megcímzéséhez felhasznált 8 címző bit-et CÍMZÉSKOR egy 8 bit-es komparátor $B_{0\text{-}7}$ bit-jeire adjuk rá. A komparátor $A_{0\text{-}7}$ bit-jei egy 8 kimenetű $M = 100$ modulusú számlálóra csatlakoznak, mely számláló azonos ütemben lépked a cella-mezőt képviselő kétfázisú ($\Phi_1\Phi_2$) regiszterrel.

Ha a B_{0-7} CÍM-kód megegyezik a cella-mező pillanatnyi helyzetét jellemző A_{0-7} számláló-kóddal, akkor a *DI* adatbemenetre adott, illetve a *D0* adatkimeneten kiolvasandó információ beíródik, illetve kiolvasódik, attól függően, hogy $W/\overline{R} = 1$, illetve $W/\overline{R} = 0$ parancsot adtunk a memóriának $ME = 1$ vezérlés esetén. $ME = 0$-nál a memória TÁROLÁSI üzemmódban áll be, azaz a benne levő információt keringteti a következő „hozzáfordulási" műveletig.

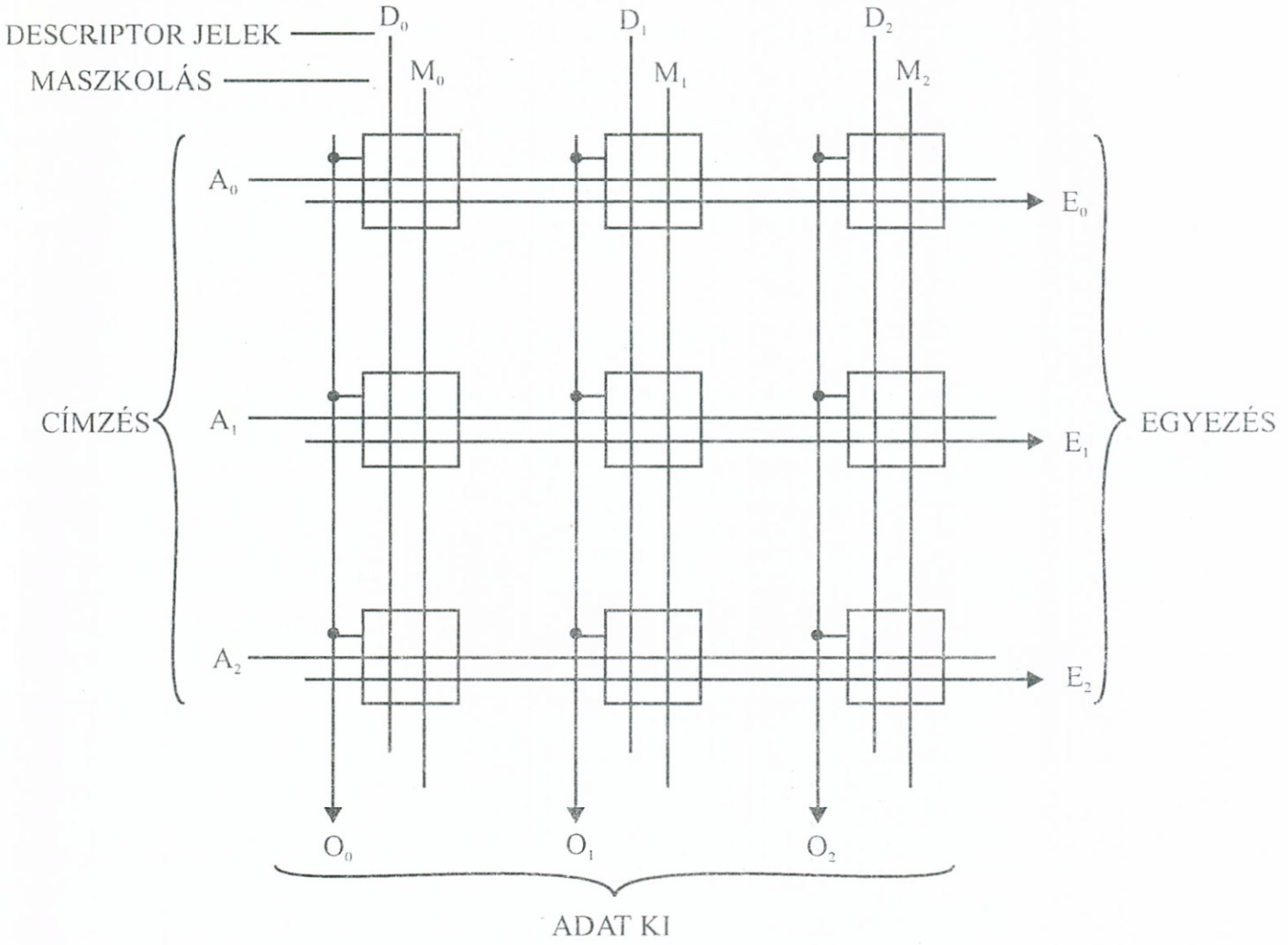


10–97. ábra *100 bit-es szó-soros dinamikus memória*

10.8.5. Aszociatív memóriák

Aszociatív memóriáknál a tartalom tekinthető körülbelül ismertnek, és az ehhez tartozó címet kell megtalálnunk (CAM). Ilyen feladat adódhat például, amikor egy árukészletből, megadott feltételnek eleget tevő cikket kell kikeresnünk. A keresési folyamat gyorsabb lesz, ha a feltétel paraméterekhez tartozó cikkeket keressük ki, és ezek közül választunk ki egyet.

A „hagyományos" memóriákhoz képest tehát e memóriák még összehasonlító elemeket is tartalmaznak, melyek gyakran már a tároló-cellákba vannak beépítve, így egy ilyen cella többféle csatlakozás-



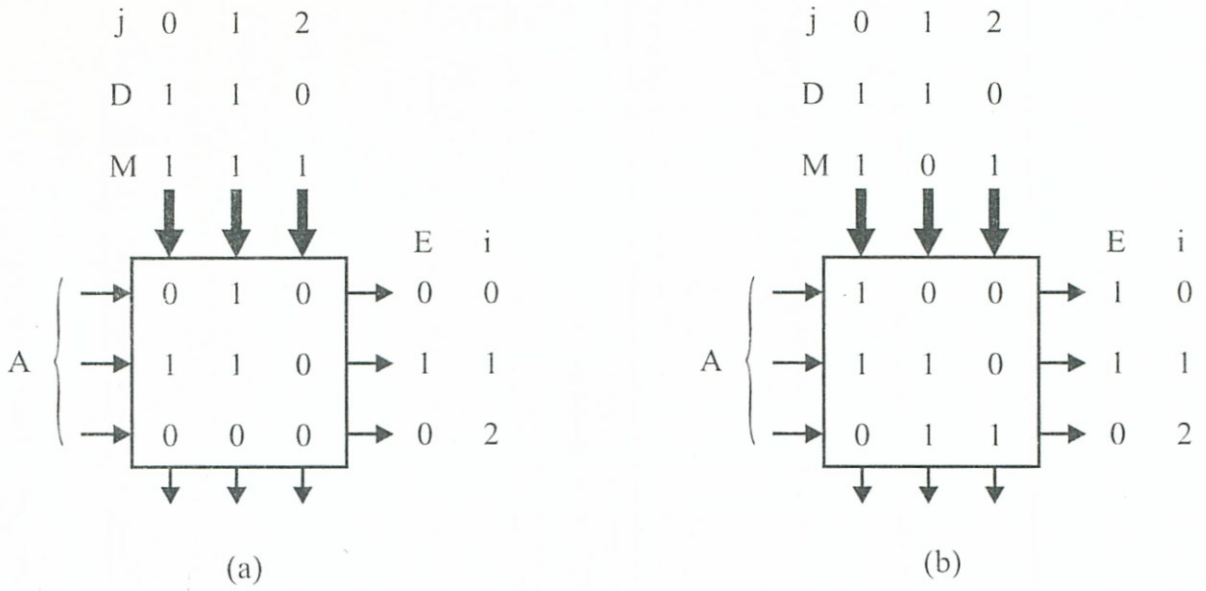
10 - 98. ábra. *CAM memória cella - mátrixa*

sal is rendelkezhet a 10.98. ábrának megfelelően. Az itt felrajzolt cella-mátrixnál

- „RAM"-üzemmódban az „A_i" vezetékekkel címezzük meg az egyes sorokat és a „D_j" vezetékeken beadott jelek kerülnek beírásra, illetve a kiolvasáskor a „O_j" kimeneteken jelennek meg.
- „CAM"-üzemmódban kissé más a helyzet.

Itt is az „A_i" vezetékekkel címezzük meg az egyes sorokat, viszont a „D_j" (Descriptor) bemenetre érkező jeleket összehasonlítjuk az aktuális cellában tárolt tartalommal. Eredményül azokon az „E_i" (Egyezés) vezetékeken fogunk kimenő jeleket kapni, mely esetekben a tárolt információ megegyezik a „D"-n beadott értékkel. A memória még rendelkezik „M_j" (Maszkoló) bemenetekkel is, melyek a bitenkénti engedélyezés feladatát látják el. Csak azokon a helyeken értékelődik az összehasonlítás, melyeket a maszkolás engedélyez.

Az összehasonlítás és maszkolás működését szemlélteti a 10.99. ábra példája. Az *a) esetben* minden „j" bit engedélyezett ($M_j = 1$), és



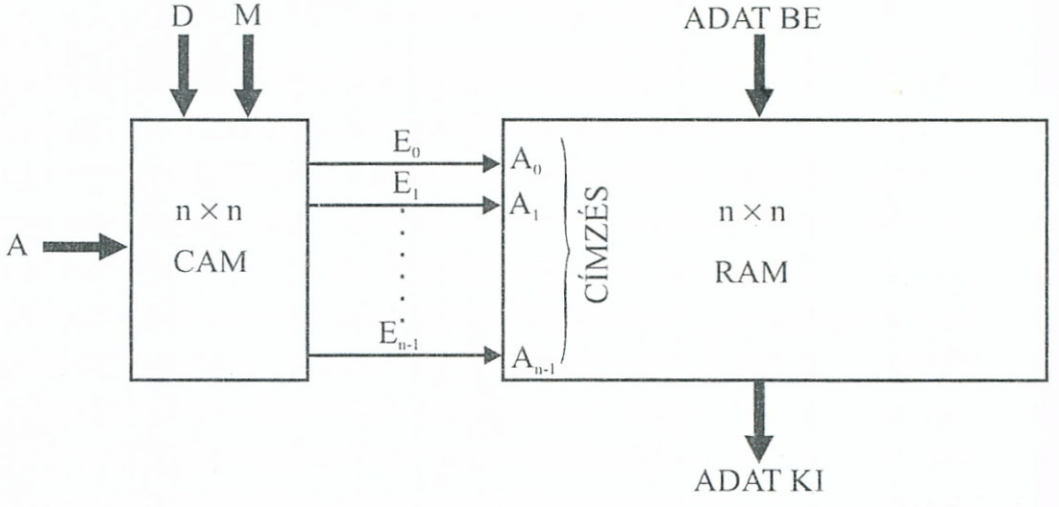
10 - 99. ábra. *DESCRIPTOR - MASZKOLÁS - EGYEZÉS kapcsolata*

így valamennyinek részt kell vennie az összehasonlításban. Mint látható, ez esetben az i = 1 helyen találtunk teljes egyezést ($E_1 = 1$). A *b) esetnél* a j = 1 eset tiltva van ($M_1 = 0$), ezért az összehasonlítást a j = 0 és j = 2 helyeke nézve kell elvégeznünk. Ennek megfelelően az i = 0 és i = 1 helyeken találunk egyezést ($E_0 = 1$, $E_1 = 1$).

A CAM-okat sokféle feladat megoldásánál alkalmazzák. Egy gyakori alkalmazási területre utal a következő példa.

Vizsgáljuk meg a 10-100. ábrán látható elv szerint működő *gyorscímzésű RWM*-egység működését.

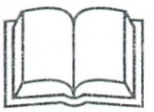
Mint látható, a tényleges RWM feladatot egy RAM egység látja el, melynek címző vezetékei csatlakoznak egy CAM egyezéstípusú kimeneteihez. Tételezzük fel, hogy a RAM-ban olyan adatokat tárolunk, melyeket valamilyen kiemelt szempontból „kigyűjtöttünk" egy nagyobb, tárolt adatkészletből. A címzéskor a CAM gyorsan el tudja



10 - 100. ábra. *Gyorscímzésű CAM - RAM rendszer elvi vázlata*

dönteni, hogy a címzendő adat a RAM-ban van-e, avagy más helyen kell keresnünk, ily módon a keresés meggyorsul. Az elv bővíthető oly módon, hogy például valamely nagykapacitású memóriát több CAM-mal címzett szegmensre tagolunk és az adatkeresést ezekkel végeztetjük. A fenti elven így rendkívül meggyorsíthatjuk egy nagymennyiségű adathalmaz cél-orientált átvizsgálását.

10.8.6. Memóriák összekapcsolása



Amennyiben a műszaki feladat által igényelt memória-kapacitás meghaladja a rendelkezésre álló memória-modul jellemzőit, ilyenkor a realizálandó memóriát több összetevő modulból kell összeraknunk, azaz kisebb méretű memóriákat kell összekapcsolnunk.

Az összekapcsolás két alapváltozaton alapul:

– szóhossz bővítésen, és
– kapacitás-bővítésen.

Általános esetben a két alapváltozatot kombináltan kell alkalmazni.

a) *Szóhossz bővítésnél* az összetevő memória-modul szóhossza *rövidebb* a szükségesnél, ezért több modult kell összekapcsolnunk a hosszabb szóhossz érdekében.

Példaként egy ROM-típusú memória szóhossz-bővítését mutatjuk be a 10.101b. ábrán, ahol feltételeztük, hogy a benne szereplő modulok a 10.101a. ábra szerint épülnek fel és az Y_j adatkimenetek háromállapotúak, továbbá az ME engedélyező bemenettel aktíválhatók.

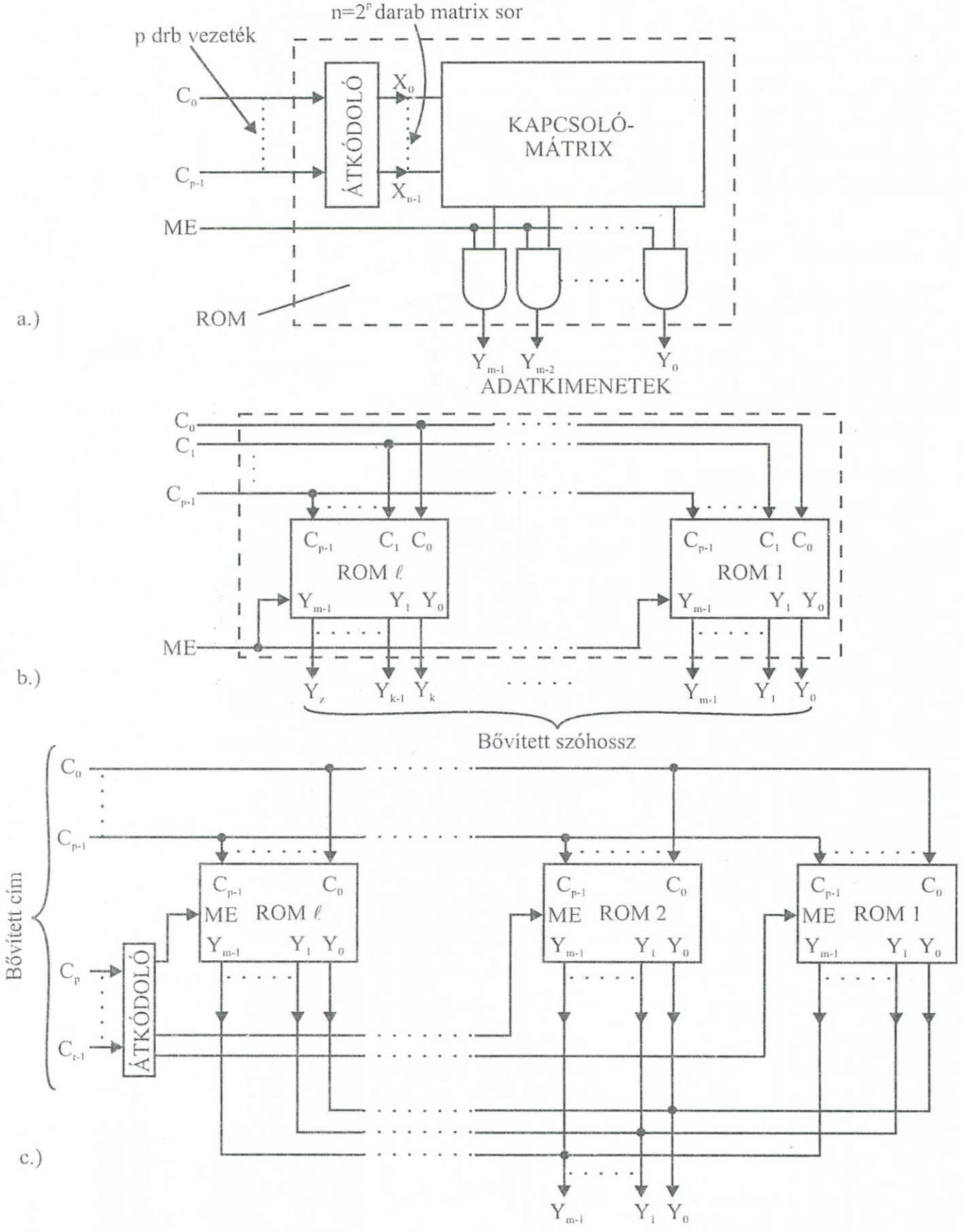
b) *Kapacitás-bővítésnél* az összetevő memória-modulban levő szavak darabszáma *kevesebb* a szükségesnél, és most emiatt kell több modult összekapcsolnunk.

Példaként itt is a 10.101a. ábra modulját vesszük alapul, és ezzel mutatunk be egy kapacitás-bővítési esetet a 10.101c. ábrán. Az egyes modulok szükség szerinti bekapcsolását az *ME*-pontokon keresztül az *ÁK* átkódoló áramkör végzi, melynek bemeneteivel a címtartomány kiegészül.

Természetesen a két bővítés kombináltan is előfordulhat egy adott feladatnál. Ilyenkor célszerű először szóhossz-bővítéssel „közbenső" modulokat képezni, majd a kapacitás-bővítést e közbenső modulokkal végrehajtani.

További memória-bővítési példamegoldásokkal a FELADATGYŰJTEMÉNY aktuális részében találkozhatunk.

A bemutatott bővítési elvek, mind ROM, mind RWM memóriák esetén alkalmazhatók.



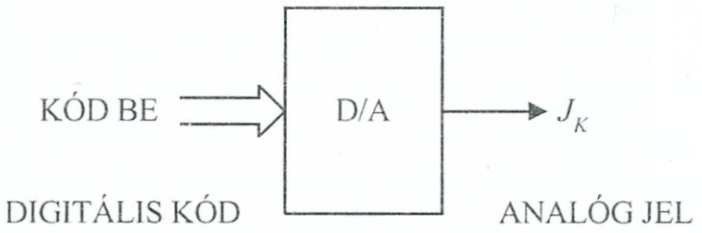
10 - 101. ábra. *Szóhossz-, illetve kapacitásbővítési elv szemléltetése adott ROM - memória modul felhasználásával*

A *SAM típusú memóriáknál* a kapacitás-bővítés a regiszterhossz növelését jelenti. Szóhossz bővítésnél: *szó-soros* esetben a párhuzamos regiszterek számának növelése, míg *bit-soros* esetben a regiszterhossz szóhosszból eredő növelése szükséges.

CAM-memóriáknál a bővítés az elmondottakhoz hasonlóan történik, de továbbmenőleg kiterjed a kulcs, maszk és egyezés csatlakozásokra is.

10.9. Digitál-Analóg átalakítók

A digitál-analóg átalakítók (D/A konverterek) a *digitális* típusú jelek *analóg* típusú alakítására szolgálnak. Ez áramkörileg a kódolt számadatoknak, velük (legtöbbször) arányos elektromos feszültséggé (vagy árammá) konvertálását jelenti a gyakorlati esetek többségében (10-102. ábra).



10–102. ábra *D/A átalakító elvi vázlata*

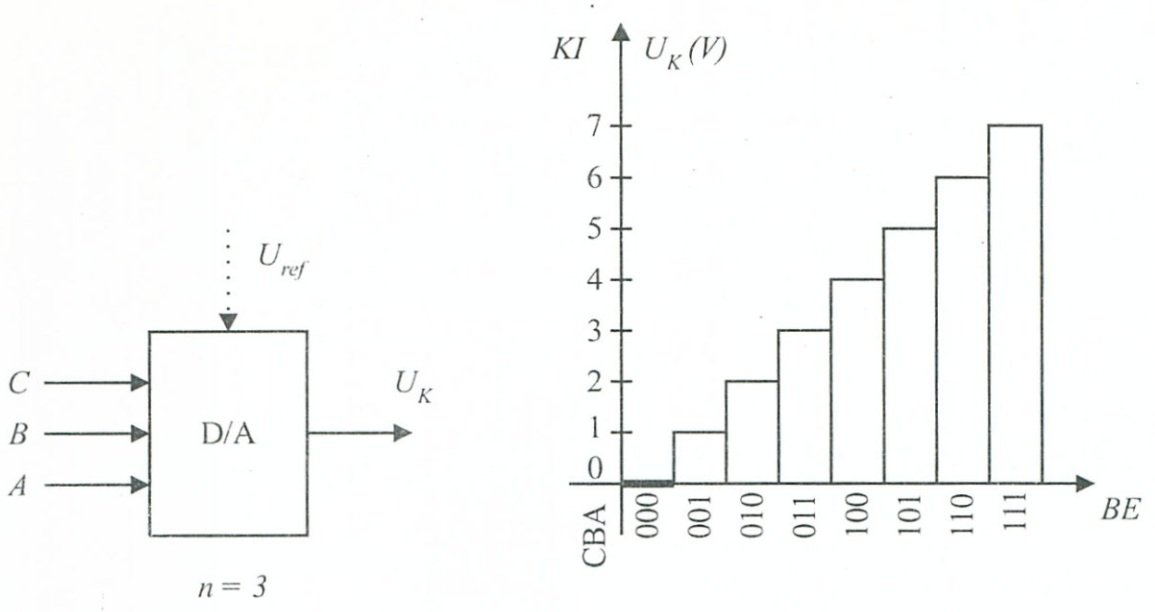
10.9.1. Elvi működés, általános blokkvázlat

A D/A átalakító minden lehetséges kódkombinációhoz egy-egy diszkrét J_K értéket rendel hozzá, melyek száma: N. A digitális bemenetek szempontjából nem közömbös, hogy az átalakító milyen kódrendszerben működik. Például:

- „N-ből 1" típusú kódnál a szükséges bemenetek száma: $n = N$
- Bináris kódnál pedig: $n = \text{ld}\, N$

Természetesen a kialakításkor más szempontokat is figyelembe vesznek.

Szemléltetésül a 10-103. ábrán felrajzoltuk egy bináris kóddal dolgozó D/A átalakító átviteli karakterisztikáját. Mint látható, az U_K kimeneti feszültség értékkészlete egy N lépcsős *kvantált* halmaz,



10–103. ábra *D/A konverter átviteli karakterisztikája*

mely elvileg tetszés szerinti sűrűségűre finomítható a bemeneti kódelemek n számának növelésével.

Az ábrán bejelölt U_{ref} *referenciafeszültség* a kimenő jelekhez támpontul szolgáló nagypontosságú és stabilitású egyenfeszültség, ami üzemi jelnek tekinthető.

Az elterjedten használt bináris D/A átalakítók bemeneti bitszámai és a hozzájuk tartozó lépcsők száma az alábbi:

n	N
8	256
10	1024
12	4096
16	65536

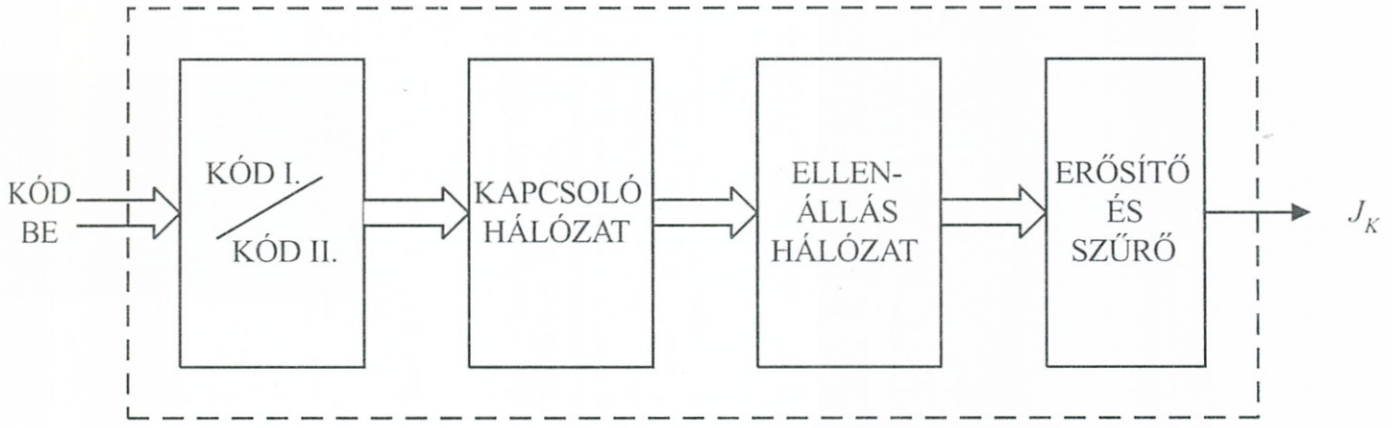
ezek közül a 10 és 12 bites változatok a leggyakoribbak.

Az úgynevezett *beállási idők,* melyek BE—KI relációban értendők *ns* és *μs* értéktartományokba esnek.

A D/A konverterek működése szempontjából – mint láttuk – nem mindegy az analóg jellé alakuló kód típusa, ezért gyakran a bemeneten a megfelelő kódot szolgáltató *átkódolót* helyeznek el.

Az egyes kódokhoz tartozó feszültségszintek előállítása aktív vagy passzív *ellenállások osztóláncaival* történik, melyek kívánt kombinációinak beállítását egy *kapcsolóhálózat* végzi el.

Végül a kimeneten *általában erősítő és szűrő* elemek biztosítják a kívánt analóg kimeneti jel megfelelő paramétereit.



10–104. ábra *D/A átalakító összetevői*

Egy D/A átalakító főbb egységeit összefoglaló blokksémát a 10-104. ábrán rajzoltuk fel. Konkrét megoldásoknál egyes egységek elmaradhatnak, vagy egyetlen fokozatban egyesítve realizálódnak.

10.9.2. Kapcsoló- és ellenálláshálózatok

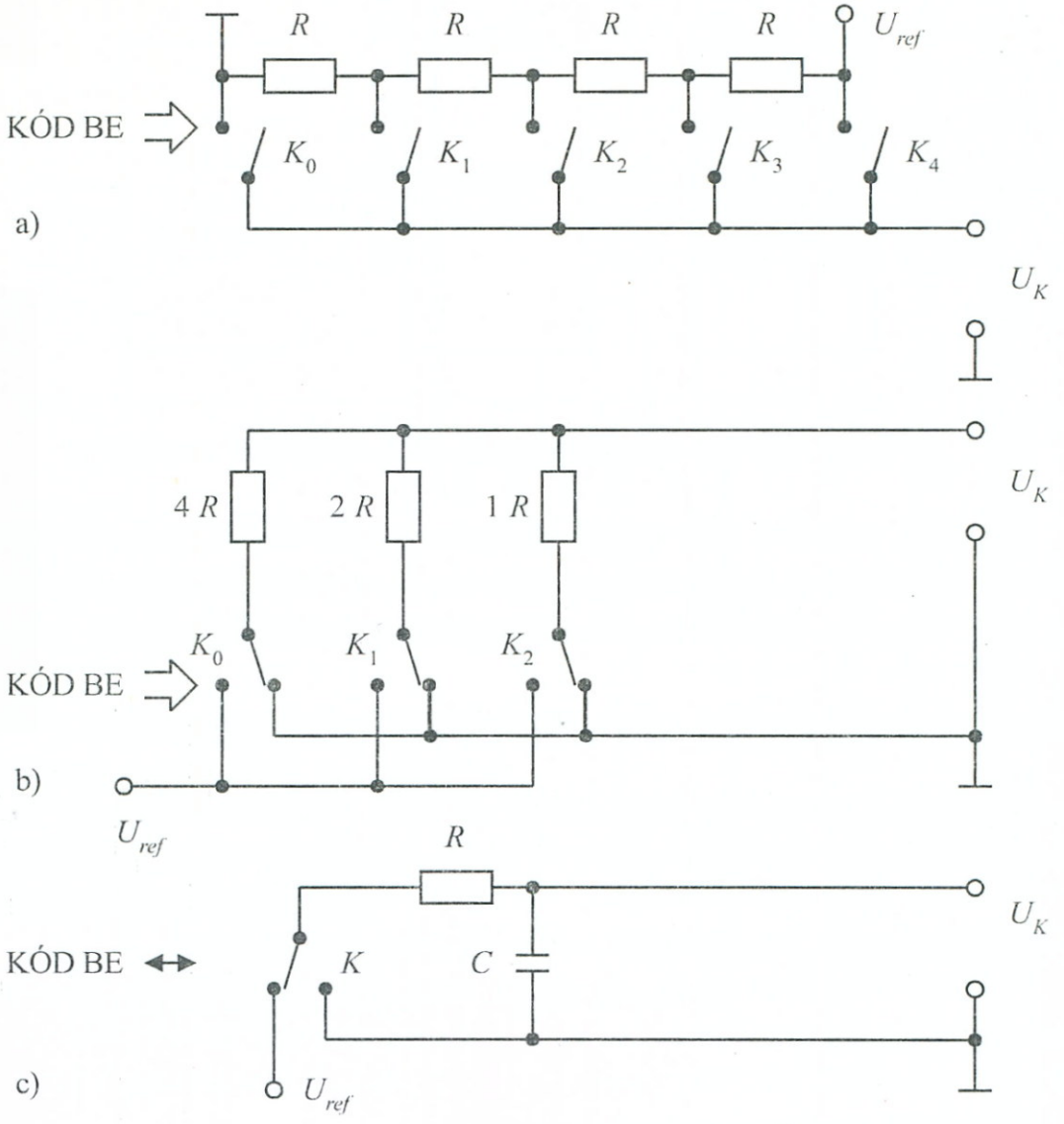
A 10-104. ábra általános blokkvázlatán szereplő KAPCSOLÓ-HÁLÓZAT és ELLENÁLLÁS-HÁLÓZAT együttese valósítja meg a tényleges átalakítási feladatot. Az együttes kialakítására három elvi alaptípus alakult ki, melyeket a 10-105. ábra foglal össze.

A 10-105a. ábrán az ún. *direkt* változat elvi rajza látható. Itt a bemeneti kód a már említett „N-ből 1"-es kód, mellyel az ellenállásosztó azon emeletén levő kapcsolót zárjuk, melynek feszültsége a kívánt U_K kimeneti analóg feszültségérték.

A 10-105b. ábrán a *súlyozott* változat szerepel. Ennél a hálózatnál az egyes ellenállások leggyakrabban bináris helyérték szerint vannak súlyozva és a kapcsolók száma is megegyezik a helyértékek számával.

A 10-105c. ábrán a *számlánccal vezérelt* változatot rajzoltuk fel. Itt csak egyetlen periodikusan működő kapcsoló található, mely egy *R–C* tagra dolgozik. A *K* kapcsoló BE/KI kapcsolt állapotainak arányát jellemző időkitöltési tényezőt az *SZ* számlálóval lehet beállítani, mégpedig úgy, hogy a *C* kapacitás sarkain mérhető kimeneti feszültség számtani középértéke a kívánt U_K értéknek feleljen meg.

A felsoroltak közül leggyakrabban alkalmazott alaptípus a *súlyozott* változat, mivel a direkt változat sok kapcsolót és sokelemes ellenállásosztót igényel, az egyszerűsége miatt kedvezőnek látszó számláncos változat pedig az RC tag időállandója miatt lassú beállási időjellemzővel rendelkezik.

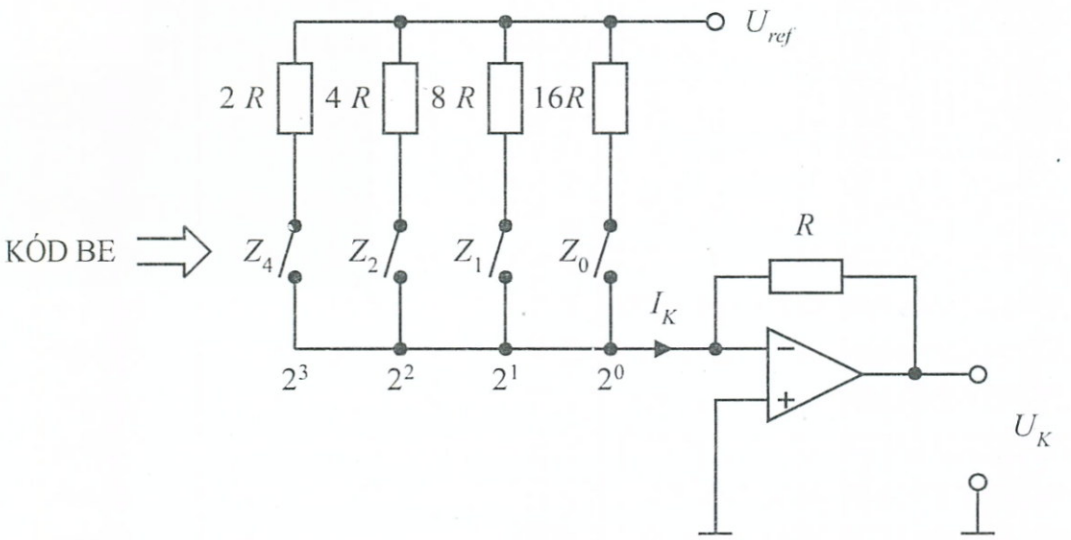


10–105. ábra *D/A kapcsoló- és ellenálláshálózatok*

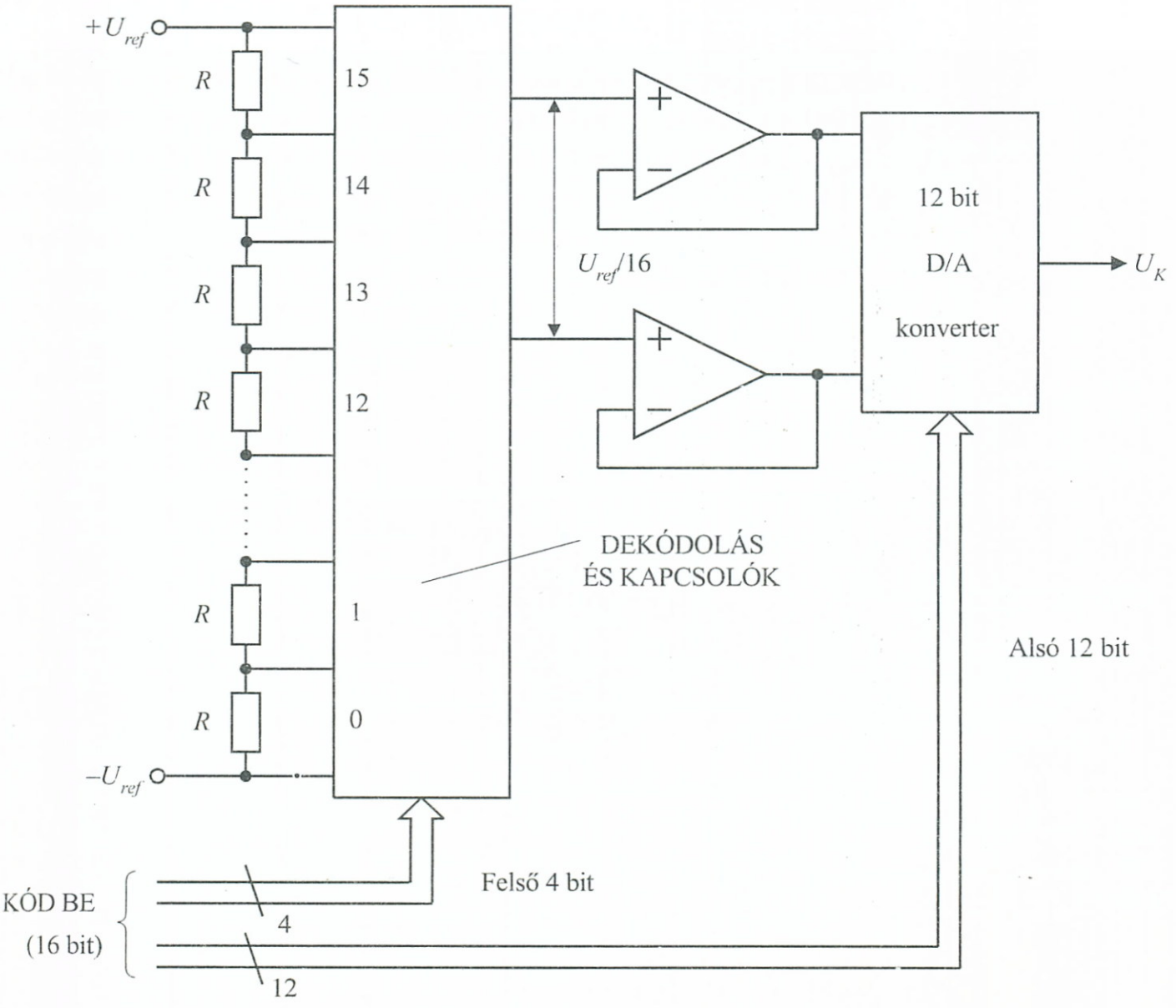
10.9.3. Jellegzetes megvalósítási esetek

a) *Áramok súlyozott összeadásán* alapul a 10-106. ábrán felrajzolt 4-bites D/A átalakító. Az egyes „*i*" helyértékek ágain zárt Z_i kapcsolóállásnál folyó I_i áram értéke a helyérték bináris súlytényezőjének megfelelő értékre van beállítva az R_i ellenállással. A Z_i kapcsolók akkor záródnak, amikor a bemenet i helyű kód-bitje 1 értékű. Mivel a műveleti erősítő R értékű visszacsatoló ellenállása miatt az összegezőponton mindig 0 feszültség lesz, az egyes részáramok egymást nem fogják befolyásolni. Ezért a kimeneti analóg U_K feszültségre felírható:

$$U_K = -U_{ref} \frac{Z}{Z_{\max} + 1} \qquad (10.55)$$



10–106. ábra *D/A konverter súlyozott áramokkal*

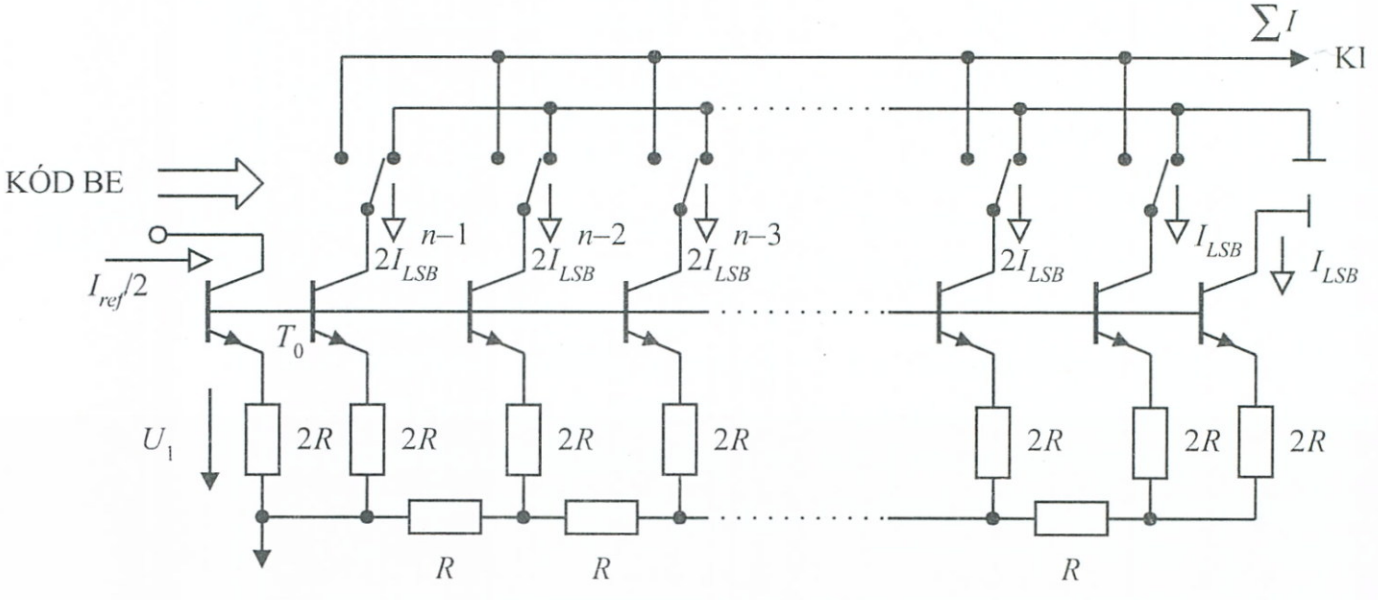


10–107. ábra *Szegmentált D/A konverter*

Z_{max}-nál minden kapcsoló zárva van és az eredő súlyozás ekkor: 15, Z a pillanatnyi bekapcsolt súlyozásokból számítható.

b) Egy nagypontosságú, ún. *szegmentált 16 bites D/A konverter* látható a 10-107. ábrán. Itt a konvertálás két lépcsőben valósul meg. A bemeneti 16 bites kód felső 4 bitje az *első fokozatra* kerül, ahol a „16-ból 1"-es kódban felépített direkt változatú ellenállásosztó aktuális emeletét választja ki, ezáltal előállítja az U_{ref}-nek azt az 1/16-od részét kitevő tartományt, amelyen belül az előállítandó analóg szint van. Ennek a szegmensnek alsó és felső határa fogja képezni a *második fokozat* 12 bites D/A konverterének U_{ref} tartományát. Az így előállított 16-bites D/A konverter igen jó paraméterekkel rendelkezik.

c) *Kapcsolt áramgenerátoros* D/A konverter látható a 10-108. ábrán. Itt az I_K kimenő áram (a 4. fejezet 4.1. és 4.3. pontjaiban már megismert) állandó áramú bipoláris áramgenerátorok áramának eredőjeként alakul ki. Az összetevők helyértékenkénti súlyozásuak és a bemeneti bináris kódtól függően aktivizálódnak. A működés során a referencia $I_{ref}/2$ értékű áram a T_0 tranzisztor $2R$ emitter ellenállásán $U_1 = RI_{ref}$ referenciafeszültséget hoz létre. T_0 bázispontja össze van kötve az egyes ágak tranzisztorainak bázisával, és ha feltételezzük, hogy mindegyik tranzisztornál az U_{BE} bázis-emitterfeszültség egyenlő, akkor az R–$2R$ létrahálózat $2R$ ágaiban folyó emitteráramok egy binárisan súlyozott sorozatot képeznek. Az analóg kimenőjelet képviselő I_K áram a kollektorkörbe beiktatott kapcsolóhálózat kódjainak függvényében állítódik elő.

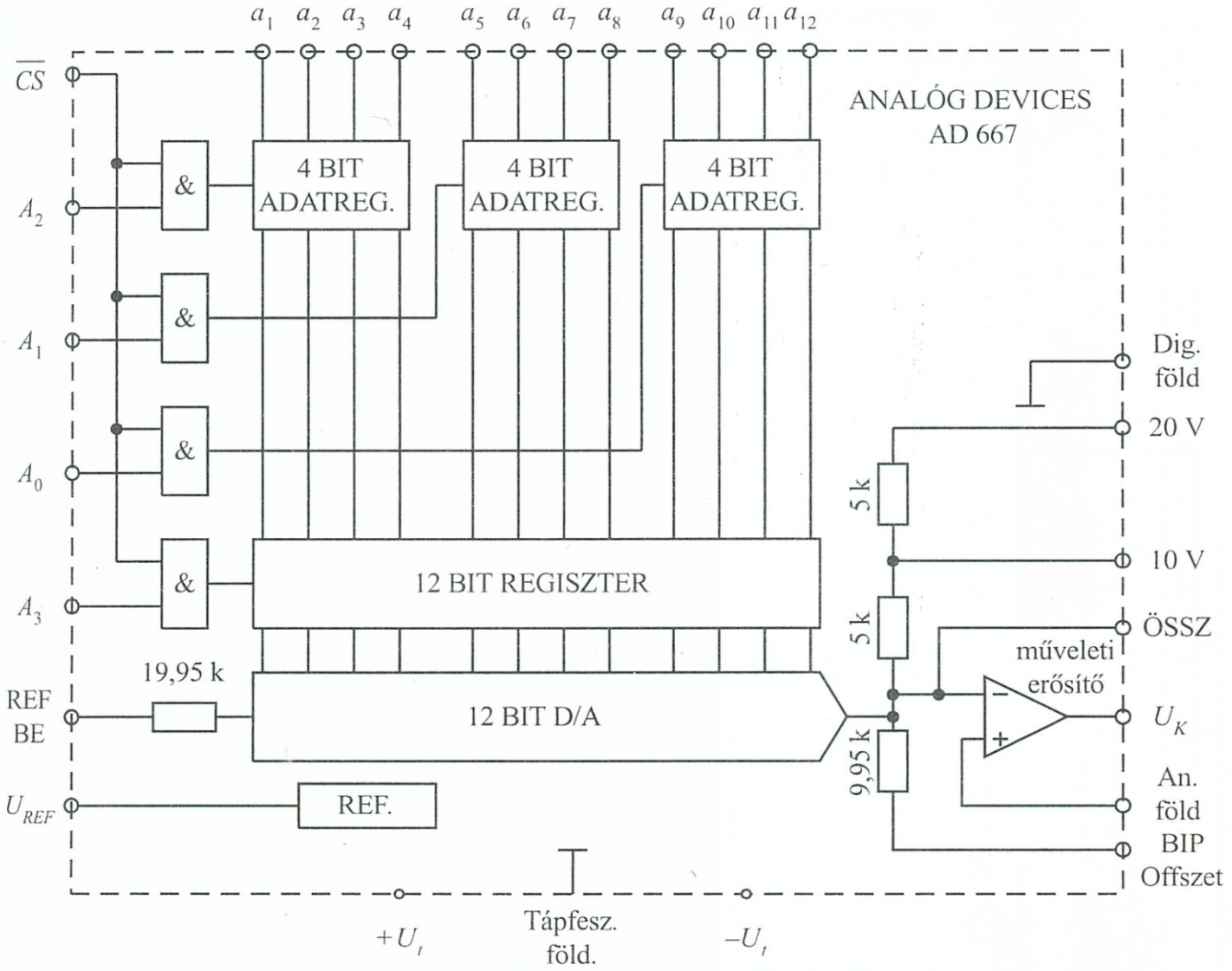


10–108. ábra *Kapcsolt áramgenerátoros D/A konverter*

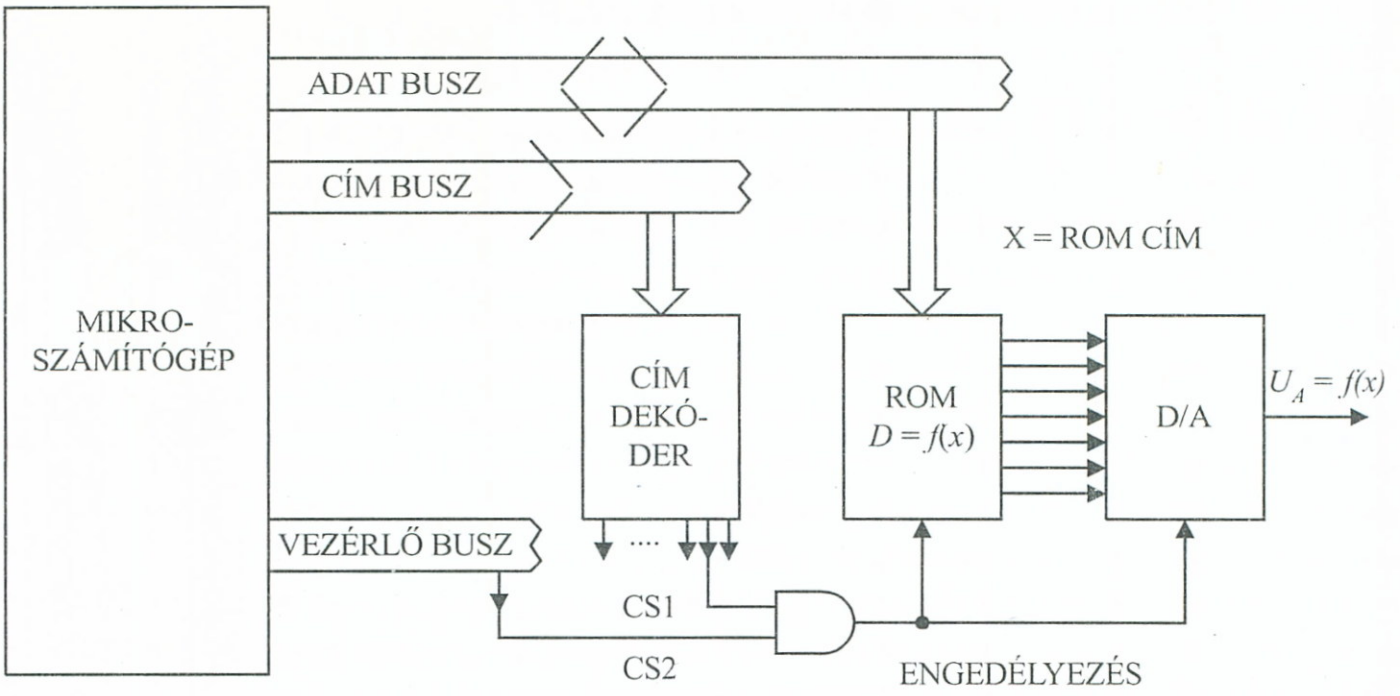
Az ábrán a I_{LSB} a legalacsonyabb súlyozású kódelemhez tartozó áram értéke (LSB = Least Significant Bit).

10.9.4. Alkalmazások

a) A 10-108. ábrán bemutatott kapcsolt áramgenerátoros D/A átalakító, egy ANALÓG DEVICES integrált áramkörbe beépített 12 bites változatát szemlélteti a 10-109. ábra. A chipbe beépítették az U_{ref} forrást, a 12 bites D/A konvertert és a kimeneti műveleti erősítőt, valamint a különféle beállításokhoz és az ofszeteléshez szükséges ellenállásokat. A chipbe beépítettek továbbá egy digitális jelek átmeneti tárolására felhasználandó regiszter-tömböt is, melynél az egyes regiszterek kiválasztására a chip-select ($\overline{CS}$) jellel kapuzott $A_0 \ldots A_3$ választó bemenetek szolgálnak.



10–109. ábra *D/A átalakítós IC kisegítő áramkörökkel*



10–110. ábra *U = f(x) függvény előállítása D/A konverterrel*

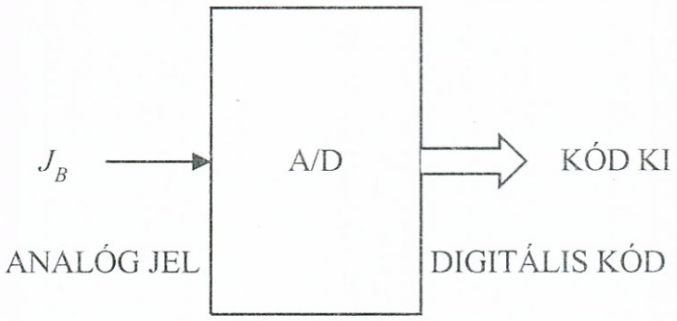
b) Szemléltető alkalmazási feladatként a 10-110. ábrán felrajzoltuk egy számítógéphez csatlakozó D/A átalakítót tartalmazó hálózat elvi vázlatát, mely tetszés szerinti

$$y = f(x)$$

függvény kiválasztott értékeinek előállítására szolgál. A hálózat aktíválása a rendszer–CIM–BUSZ-ra csatlakozó CÍMDEKÓDER és VEZÉRLŐBUSZ csatlakozásokkal kapuzott CS1, CS2 CHIP-SELECT jelekkel történik. Az $f(x)$ függvény (pl. logaritmus-függvény értéktáblázatát a ROM memória tartalmazza, melynek megcímzését a rendszer ADAT-BUSZ-on keresztül végezhetjük. A ROM adatkimeneti vezetékei a D/A konvertert vezérlik, melynek kimenetén az $U_K = f(x)$ analóg feszültség formájában megjelenik.

10.10. Analóg-Digitál átalakítók

Az analóg-digitál átalakítók (A/D konverterek) az analóg típusú jelek digitális típusúvá alakítására szolgálnak. Ez áramkörileg a gyakorlati esetek többségében az elektromos feszültség (vagy áram) értékeknek velük (legtöbbször arányosan) összefüggő kódolt számadatokká való átalakítását jelenti (10-111. ábra).

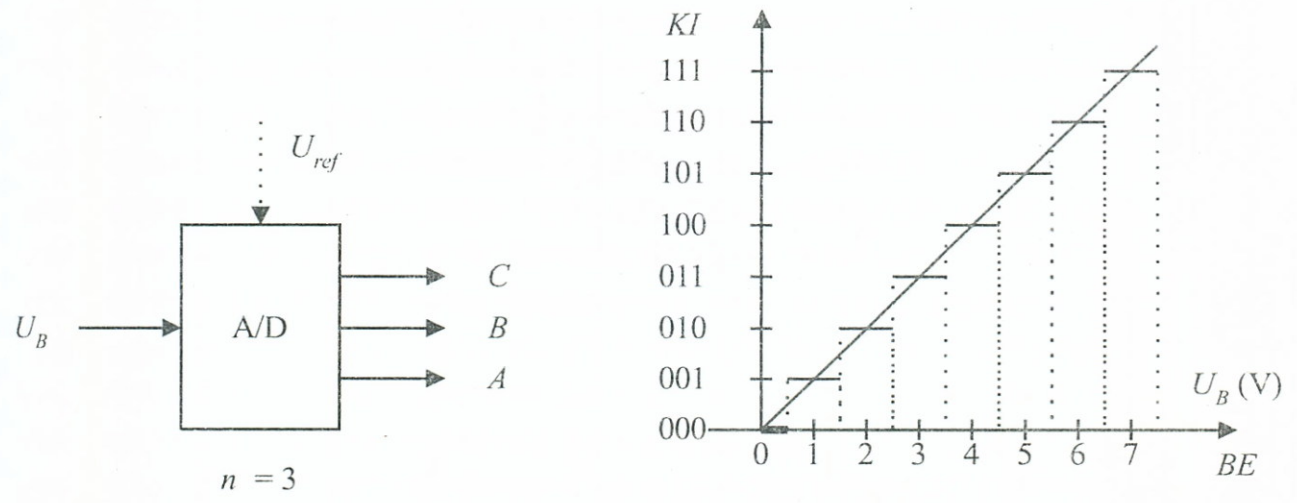


10–111. ábra *A/D átalakító elvi vázlata*

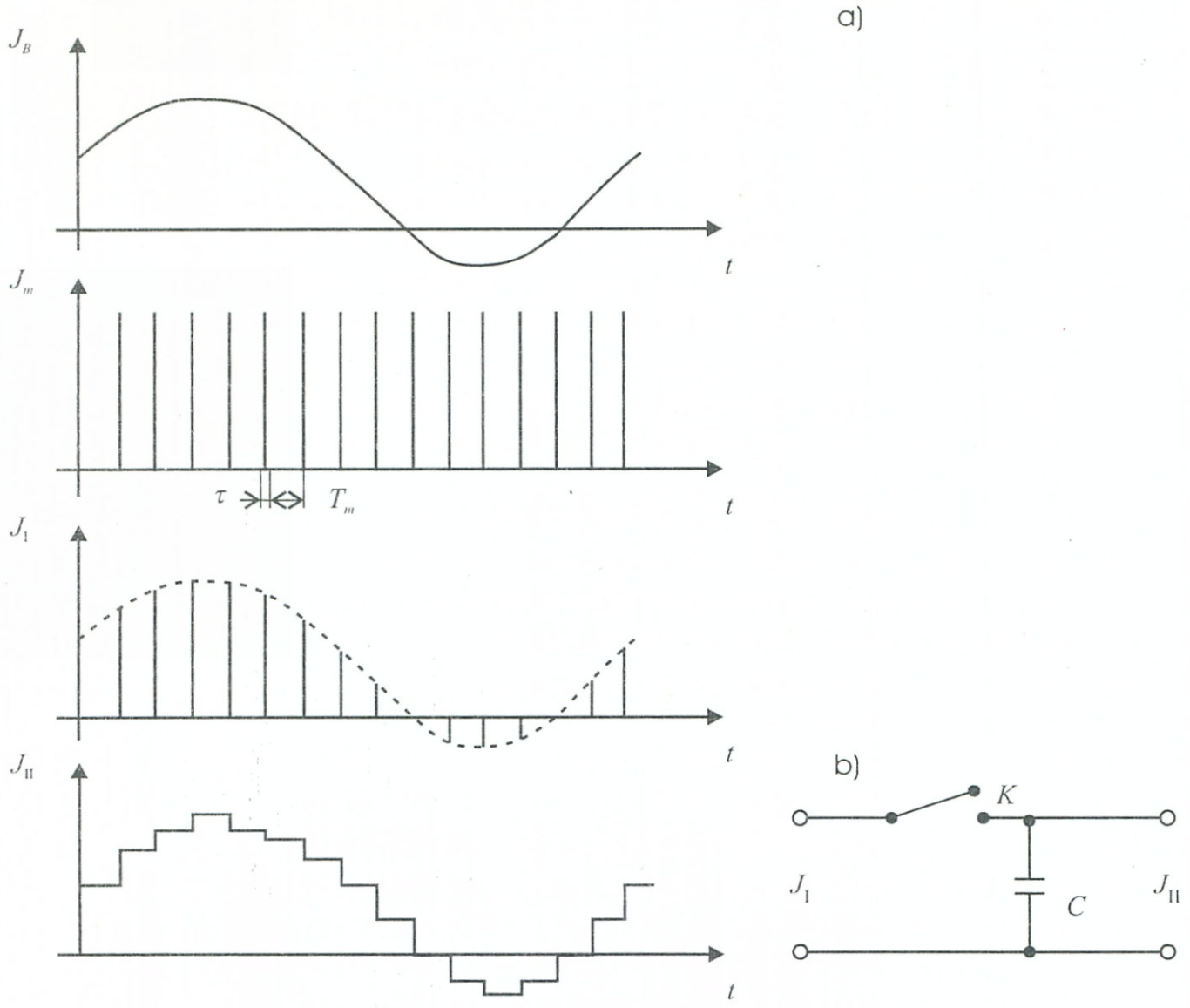
10.10.1. Elvi működés, mintavételezés

A 10-112. ábrán felrajzoltuk egy A/D átalakító átviteli karakterisztikáját, melynél a példaként választott $n = 3$ érték miatt, a bemenő analóg jel (itt feszültség) tartományát 8 részre kellett osztani. Itt a 0-7 V tartományba eső, beérkező analóg jelet az 1 V szélességű szakaszon belül az átalakító ugyanazzal a kóddal adja meg, ami azt jelenti, hogy pl. 101 kimenetet észlelve, a bemenetre 4,5 és 5,5 V között bármilyen feszültség érkezhetett. Ez legkedvezőtlenebb esetben itt ± 0,5 V átviteli hibát jelenthet.

Az A/D átalakító bemenetére érkező J_B jel „digitalizálásához" több feladatot kell az átalakítónak megoldania: a $J_B = J_B(t)$ időfüggő jelet bizonyos időpontokban meg kell mérni, a mérés eredményét a diszkrét értékskála adataival össze kell hasonlítani, majd valamelyik (rendszerint a legközelebbi) fokozathoz be kell sorolni, végül a karakterisztika szerinti digitális kódot ki kell küldeni.



10–112. ábra *A/D konverter átviteli karakterisztikája*



10-113. ábra *Mintavételezés és tartás*

A periodikusan ismétlődő teendők első lépése a mintavételezési eljárás, melynek folyamata a 10-113a. ábrán követhető. Az első ábra a $J_B = J_B(t)$ analóg bemeneti időfüggvényt mutatja. A periódikus vizsgálat a T_m mintavételezési periódusokkal jellemzett τ szélességű mintavételező impulzus-sorozattal, a J_m *mintavételező* jel közreműködésével történik, melynek révén előállítódik a J_I *mintavételezett jel.* Ez utóbbi is egy impulzus-sorozat, de – ellentétben az állandó amplitúdójú impulzusokból álló mintavételező jellel – itt az impulzusok amplitúdója megegyezik a J_B függvény adott időpontban mért amplitúdójával. A mintavételezett amplitúdók már összehasonlíthatók az A/D konverter-karakterisztika diszkrét értékskála adataival és az adott időpontban érvényes digitális kód kiküldése lehetővé válik.

A mintavételezés technikája mélyebb matematikai háttérrel rendelkezik, mellyel itt nem foglalkozunk. Röviden, annyit szükséges

megemlíteni, hogy a mintavételezés jóságára a mintavételező impulzusok szélessége és szaporasága is befolyással van. Ideális körülményeket feltételezve, az A/D átalakító Be-, és Kimeneti oldalain levő információknak azonosaknak kellene lenniük. Ez az azonosság úgy lenne megállapítható, ha a kimeneti oldalon levő jelet eredeti formájába „visszaállítva" (például egy *ideális* aluláteresztő szűrőn keresztül) a bemeneti függvénnyel való azonosságot tapasztalnánk. A valóságban a jelvisszaállítást az ún. *tartó kapcsolások* végzik. Ezek egyik legegyszerűbb változata az ún. *nulladrendű tartó,* melyet a 10-113b. ábrán is felrajzoltunk. A tartó bemenetére ráadott J_I mintavételezett függvény „visszaállítottja" megjelenik a kimeneten, esetünkben a J_{II} időfüggvény formájában.

A J_I és J_{II} függvények amplitúdói közötti eltérés főként abból adódik, hogy a J_{II} függvény amplitúdó értékeinek megállapításakor figyelembe kellett venni az A/D konverter rögzített N diszkrét lépcsőből álló, az egyes kódszavakhoz rendelt (10-112. ábra) skálájához történő „kerekítéseket" is. Emiatt torzítás jelentkezik, amit *kvantálási zaj*nak neveznek. Értéke a 10-113a. ábra jelöléseivel

$$\Delta J = J_{II} - J_I \qquad (10.56)$$

A kvantálási zaj miatt az eredeti analóg jelet már nem tudjuk torzításmentesen visszaállítani.

10.10.2. A/D átalakítók alaptípusai

Az A/D konvertereket három, jól elkülöníthető alaptípus köré lehet csoportosítani.

a) *Direkt* változat esetén a bemeneti analóg feszültséget az N fokozatnak megfelelő N darab referenciaértékkel hasonlítjuk össze és megállapítjuk, hogy milyen intervallumba esik. A kapott fokozathoz így „direkt" eljutunk. A változat fő hátránya, hogy fokozatonként egy-egy, összesen N (vagy N–1) komparátort igényel. A módszert *word at time* módszernek is nevezi az irodalom.

b) *Fokozatos megközelítés*es változatnál az átalakítást „fokozatosan" végezzük el. Először a legnagyobb helyérték referenciafeszültségével hasonlítjuk össze a beérkezett feszültséget, majd a maradékot összehasonlítjuk a következő helyértékkel, és ezt folytatjuk ciklikusan a végéig.

A szükséges összehasonlítások, és a referenciafeszültségek száma megegyezik a helyérték számával, azaz n = ld N. A módszert *digit at*

time vagy *szukcesszív aproximációs módszer*nek is nevezi az irodalom.

c) *Számlálós* változatnál, azt kell megszámolni, hogy hány alkalommal kell a legkisebb helyértékű referenciafeszültséget összeadni ahhoz, hogy a bemeneti feszültség kiadódjon. A módszert *level at time* módszerként is nevezik.

A módszer rendkívül egyszerű áramköröket igényel, de a sok összeadási lépés miatt viszonylag időigényes.

10.10.3. Jellegzetes megvalósítási esetek

a) Direkt változat prioritásos dekódolóval

A 10-114. ábrán egy $N = 8$ fokozatú értékskálával rendelkező, direkt A/D kapcsolás látható. A 7 darab referencia-értéket egy ellenálláslánc állítja elő az ábrán feltüntetett értékarányokkal.

Az ábrán az U_{LSB} a *legalacsonyabb súlyozású* kódelemhez tartozó feszültségérték.

Ha a beérkező feszültség értékére fennáll:

$$U_{ref\,i} \leq U_B < U_{ref\,i+1}$$

akkor a komparátor-kimenetekre felírható:

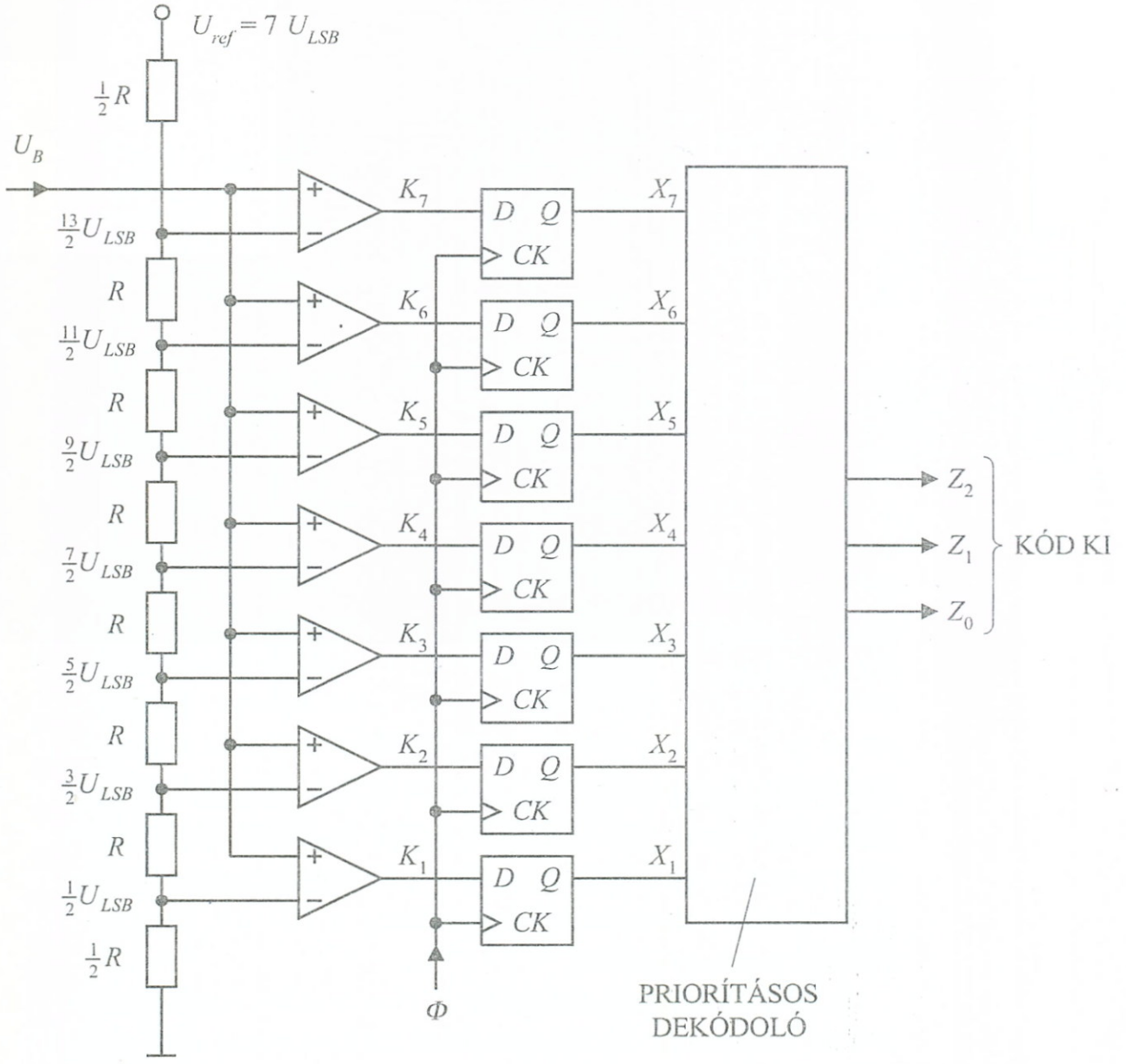
$$k \leq i = 1$$
$$k > i = 0$$

A $k_1 \ldots k_7$ értékekből képzett kódszavakat végül is egy prioritásos kódolóra vezetve (lásd. 10. fejezet 10-13. ábra), megkapjuk az i-edik „feszültség-emelet" bináris kimenő kódját.

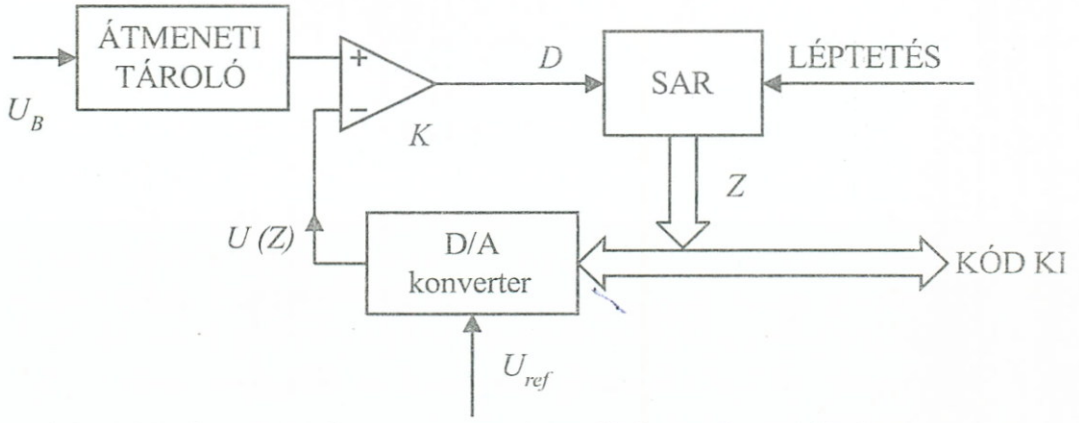
A kapcsolás bemeneti tranzienseinek kivédése céljából a k_i értékeket rendszerint egy szinkron D-tárolós P-P regiszterbe vezetik, melynek bemenetét a Φ ciklikusan érkező kapuzó-jel csak akkor nyitja ki, ha a bemenet már stabilizálódott és a kód a Prioritásos kódolóra már ráengedhető. Ezt a megoldást mutatja a 10-114. ábra is.

b) Fokozatos megközelítéses megoldás elvi vázlata látható a 10-115. ábrán.

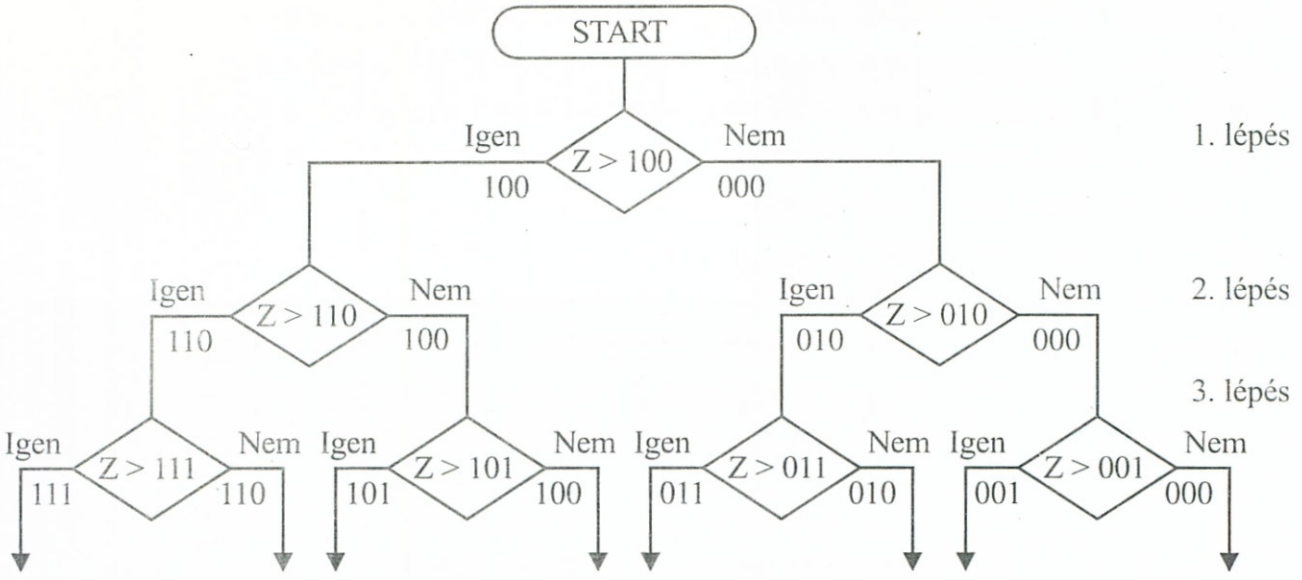
A bemeneti U_B feszültség egy ÁTMENETI TÁROLÓ-ba kerül, így az A/D átalakítás ideje alatt, mint állandó adat áll rendelkezésre. A hálózat tartalmaz egy D/A átalakítót is, melynek feszültségével kerül az U_B összehasonlításra. A D/A konverter bemenetére érkező Z súlyozási kód induláskor nullázódik, majd ezután a legnagyobb súly-



10–114. ábra *Direkt A/D konverter priorításos dekódolóval*



10–115. ábra *Fokozatos megközelítéses elvü A/D konverter*



10–116. ábra

értékű kódelem, az MSB (MSB = Most Significant Bit) 1-be áll be, és a *K* komparátor elvégzi az

$$U_B > U(Z)$$

összehasonlítást. „Igaz" esetén az 1-es megmarad, ellenkező esetben átáll 0-ára. Hasonló összehasonlítás történik ciklikusan az összes kódelemre, fogyó súlyú irányban, LSB-ig bezárólag. Az összehasonlítás döntési folyamatát 3 helyértékig a 10-116. ábra fa-struktúrája mutatja és magyarázza.

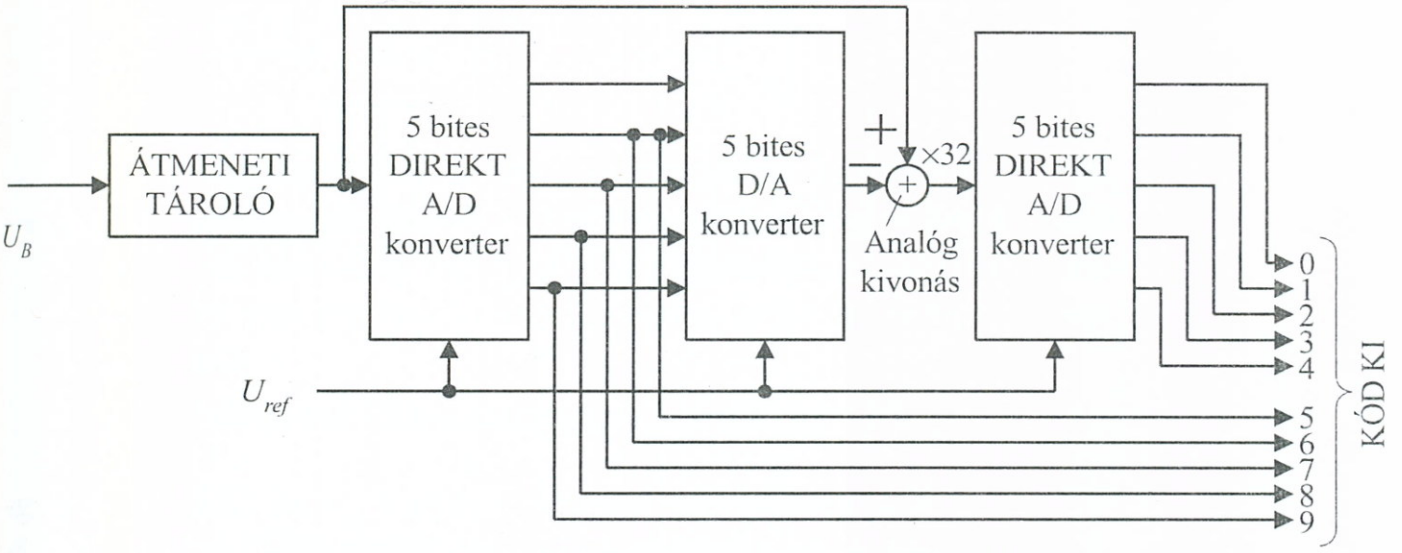
A 10-115. ábrán szereplő SAR (Successive Approximation Register) az ismertetett összehasonlítási ciklussorozat alatt shiftelődve tárolja az összehasonlítások eredményeit és LSB értékének meghatározását követően:

$$Z = (Z_{\max} + 1)\frac{U_B}{U_{ref}} \tag{10.57}$$

információval rendelkezik.

A SAR lényegét tekintve egy S–P típusú regiszter (lásd 10. fejezet 10.1. pont), de több kiegészítő áramköri részt is tartalmaz az ismertetett működés kiszolgálása érdekében. A SAR elemeket IC tokozott állapotban is forgalmazzák.

c) A kaszkád megoldású A/D átalakítókat a direkt- és fokozatos megközelítésű változatok kombinációjának is tekinthetjük. Ez a megoldás kompromisszumosan egyesíti az előnyöket. Elvi vázlata a 10-117. ábrán látható.



10–117. ábra *Kaszkád megoldásu A/D konverter*

Az ábrán látható 10 bit-es A/D átalakítónál a biteket két 5-ös csoportba ($Z_9 \ldots Z_5$), ($Z_u \ldots Z_0$) soroljuk. Az első fokozatban egy 5-bites direkt A/D átalakítóval előállítjuk a felső 5 bitet, ami az U_B egy durva lépcsőzésű kvantált értékét adja meg. A részeredményt egy 5-bites D/A átalakítóval visszakonvertáljuk újra analóggá, és analóg kivonással kivonjuk az U_B-ből. Végül az analóg módban levő maradékot egy második 5-bites direkt A/D-vel digitalizáljuk. A kapott eredmény:

$$Z = Z_{\max} \frac{U_B}{U_{ref}} \tag{10.58}$$

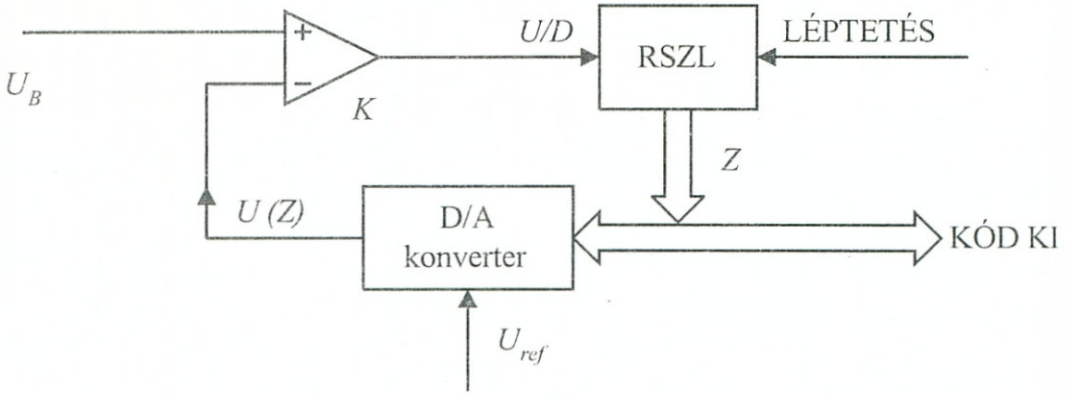
10 bit esetén

$$Z = 1023 \frac{U_B}{U_{ref}}$$

d) Számlálós megoldások

Nem igényes feladatokra alkalmazzák őket, mivel olcsó megoldásnak tekinthetők. Az átalakítás időállandója viszont elérheti a másodperces nagyságrendet is. Több változatuk terjedt el, ezek közül kettőt mutatunk be:

d1) Kompenzációs változat elvi vázlata a 10-118. ábrán látható, mely bizonyos mértékig hasonlít a 10-118. ábra „Fokozatos megközelítéses" megoldásához, csak itt egy egyszerű reverzíbilis számláló (RSZL) helyettesíti a SAR-t.



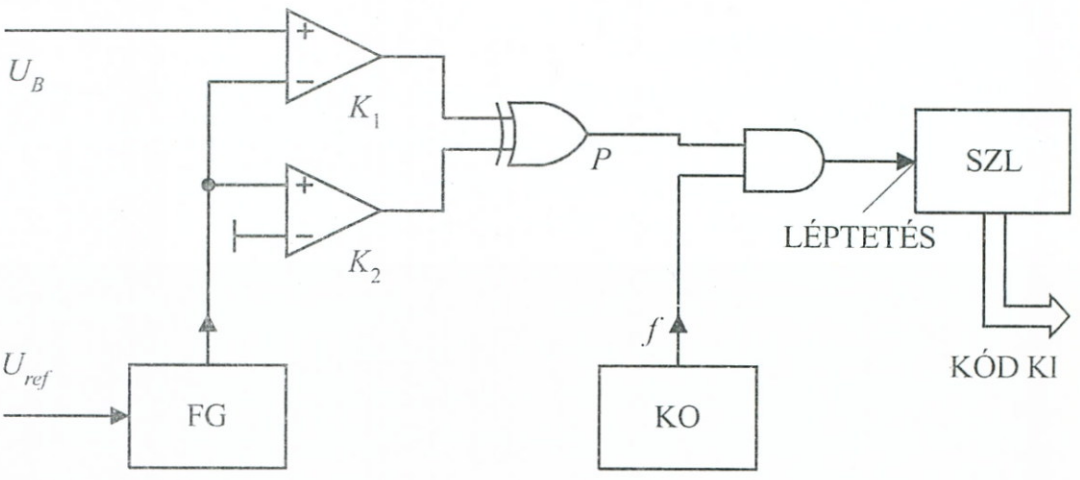
10–118. ábra *Kompenzációs A/D konverter*

A K komparátor összehasonlítja U_B-t az $U(Z)$ feszültséggel, és ha a különbség *pozitív,* ekkor RSZL-t ELŐRE, különben pedig HÁTRA lépteti.
Emiatt az $U(Z)$ feszültség előbb-utóbb közel egyenlő lesz az U_B-vel és e körül ingadozik, így a Z kimeneten:

$$Z = (Z_{\max} + 1)\frac{U_B}{U_{ref}} \tag{10.59}$$

kód jelenik meg.
Szükség szerint külön leállító áramkört is csatlakoztatnak, melyet egy ablakkomparátor vezérel, enélkül az áramkör ugyanis állandóan működik.
Az áramkört TRACKING A/D konverternek is szokták nevezni.

d2. Fűrészfogjeles A/D átalakító elvi vázlata látható a 10-119. ábrán. Az FG fűrészfoggenerátor pozitív iránytangensű lineáris feszültségfüggvényt állít elő:



10–119. ábra *Fűrészjellel működő A/D átalakító*

$$U = \frac{U_{ref}}{\tau_0} t - U_0$$

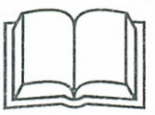
A kapcsolásban szereplő antivalencia kapuból és K_1, K_2 komparátorokból kialakított ablakkomparátor jóvoltából a P ponton addig van 1-es, míg a fűrészfeszültség 0 és U_B között van. Az előző összefüggésből ez az időtartam kifejezhető:

$$\Delta t = \frac{\tau}{U_{ref}} U_B$$

A Δt idő nagyságát a *K0* kvarcoszcillátor jeleinek megszámlálásával állapítjuk meg az *SZL* számláló segítségével, amely mindig a kezdeti 0 értékről indulva, beérkezett impulzusainak számával jellemzi az eltelt időt, és ezt kódolt formában fejezi ki a *KI* kimeneten. Ez az érték

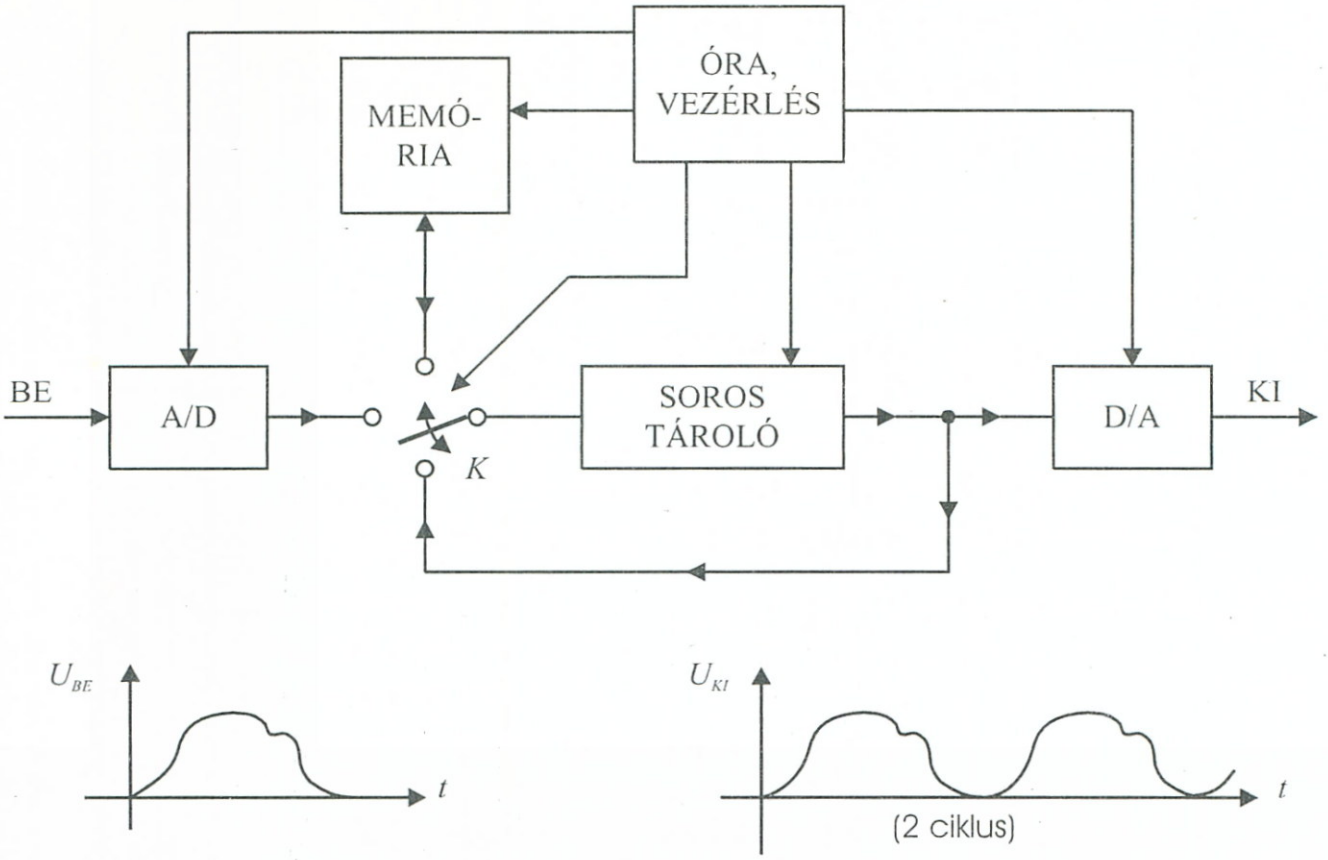
$$Z = \frac{\Delta t}{T} = \frac{\tau f}{U_{ref}} U_B \qquad (10.60)$$

10.10.4. Egy összetett alkalmazási példa



A/D és D/A átalakítók egy összetett alkalmazását mutatja a 10-120. ábra, melynél az a feladat, hogy egy bejövő analóg jelmintát tároljanak, illetve a jelmintából periódikusan többszörözött analóg jelcsoportot állítsanak elő.

Az ún. feltöltési szakaszban az U_{BE} jelet A/D digitalizálás után a SOROS TÁROLÓ-ba töltik, majd a K kapcsoló átkapcsolásával itt cirkuláltatják a visszacsatoló hurokban. Egyidejűleg a SOROS TÁROLÓ kimeneti pontjáról, a D/A-n keresztüljutva a *Ki* kimeneten megjelenik a betárolt jelminta, mégpedig annyiszor periódikusan ismétlődve, ahányszor a visszacsatoló-hurokban a körforgás megismétlődik. A *K* kapcsoló felső állásában a SOROS TÁROLÓBÓL a jelminta egy MEMÓRIÁ-ba elmenthető, ily módon ott különféle jeltípusokból egy könyvtár hozható létre, melyből az alkalmilag szükséges jelminta lehívható.



10-120. ábra *Jelminták tárolása és többszörözése*

10.E. Ellenőrző kérdések és feladatok

E.10.1. Mi a funkcionális egység?
E.10.2. Mit nevezünk regiszternek, milyen alapváltozatok léteznek?
E.10.3. Rajzoljon fel egy P-P és egy S-S-regisztert.
E.10.4. Írja fel a léptetőregiszter állapotegyenletét.
E.10.5. Mitől függ valamely üzenet információtartalma?
E.10.6. Mi az információ egysége?
E.10.7. Miként írható fel az *entropia*?
E.10.8. Mekkora az entropia, ha valamelyik szimbólum bekövetkezése teljesen biztos?
E.10.9. Miként írható fel a *redundancia*?
E.10.10. Mi a Markov-forrás?
E.10.11. Mi a kód?
E.10.12. Mi a kódolás és dekódolás?
E.10.13. Mit tud a *hiba-felfedő* kódolásról?
E.10.14. Mi a Hamming-távolság?
E.10.15. Mit tud a *hibajavító* kódolásról?
E.10.16. Soroljon fel néhány fontosabb kódrendszert.

E.10.17. Mutassa be a BIN–GRAY kódátalakítást.
E.10.18. Mit tud a *paritás bit*-tel kiegészített kódokról?
E.10.19. Hogy működnek a paritásképzők?
E.10.20. Hogy működik a Hamming hibajavító kódolás?
E.10.21. Mi a DEMULTIPLEXER? Nevezze meg két fontos felhasználási területét.
E.10.22. Mi a MULTIPLEXER, és mire használható?
E.10.23. Hogy lehet MULTIPLEXER-rel logikai függvényt realizálni?
E.10.24. Hogy lehet DEMULTIPLEXER-t, ill. MULTIPLEXER-t bővíteni?
E.10.25. Mi a számláló definíciója, milyen alapon csoportosíthatók a számlálók?
E.10.26. Ismertessen egy aszinkron előre-számlálót.
E.10.27. Milyen elvi vázlattal jellemezhető egy szinkron előre-számláló?
E.10.28. Mutasson be egy aszinkron hátra-számlálót.
E.10.29. Miben tér el egy szinkron hátra-számláló elvi vázlata az E.10.27.-ben szereplőtől?
E.10.30. Ismertessen egy reverzíbilis számlálót.
E.10.31. Mi a *ciklusrövidítés* és a *ciklusbővítés* számlálók esetén?
E.10.32. Hogy lehet digitális *frekvencia-osztót* csinálni?
E.10.33. Foglalja össze, miként lehet számlálót kialakítani SSI összetevőkből.
E.10.34. Mi jellemző az *aritmetikai* műveletvégzőkre?
E.10.35. Mi a *fix-pontos* számábrázolás?
E.10.36. Milyen a bináris előjel-abszolútértékes számábrázolás?
E.10.37. Milyen a bináris *komplemens*es számábrázolás?
E.10.38. Hogyan vezethető vissza a kivonás összeadásra?
E.10.39. Mi a túlcsordulás?
E.10.40. Végezze el különböző előjelű bináris számok összeadását.
E.10.41. Mi a *lebegőpontos* számábrázolás?
E.10.42. Mi a *normalizált* alak?
E.10.43. Mi az *implicit* bit?
E.10.44. Rajzoljon fel egy 1-bites bináris összeadó modult.
E.10.45. Mi a különbség az *egész-összeadó* és a *fél-összeadó* között?
E.10.46. Mutasson be egy *soros* bináris összeadót „n" bit-re.
E.10.47. Mutasson be egy *párhuzamos* bináris összeadót „n" bit-re.

E.10.48. Hogy működnek a LOOK–AHEAD–CARRY elvű összeadók, miben térnek el a RIPPLE–CARRY-elvű összeadóktól?
E.10.49. Rajzoljon fel egy *párhuzamos* bináris kivonót „n" bit-re.
E.10.50. Rajzoljon fel egy *soros* bináris kivonót „n" bit-re.
E.10.51. Miként bővíthetők az összeadó vagy kivonó modulok?
E.10.52. Mit tud a BCD számábrázolásról és összeadásról?
E.10.53. Mutasson be egy 6-os korrekcióval működő BCD-összeadó dekádot.
E.10.54. Milyen a legegyszerűbb bináris szorzás?
E.10.55. Milyen az „iskolás" szorzás?
E.10.56. Mutasson be egy párhuzamos szorzót 2×2 bites esetre.
E.10.57. Hány bitből áll a szorzat a tényezőkhöz viszonyítottan?
E.10.58. Mi lesz a szorzat *előjele* előjel-abszolutértékes ábrázolásnál?
E.10.59. Milyen az *1-bit figyelésű* szorzó algoritmus? Ismertesse működését egy blokkséma kíséretében.
E.10.60. Oldjon meg egy számpéldát az 1-bit figyelésű szorzással.
E.10.61. Milyen a *2-bit figyelésű* szorzó algoritmus? Ismertesse működését egy blokkséma keretében.
E.10.62. Oldjon meg egy számpéldát a 2-bit figyelésű szorzással.
E.10.63. Mutasson be egy realizációt a *Booth-féle* szorzásra.
E.10.64. Milyen formulákon alapul a *duplázó-felező* szorzás?
E.10.65. Mutasson be egy BCD szorzóberendezést.
E.10.66. Visszavezethető-e az osztás kivonásra? Mutassa be egy számpéldán keresztül.
E.10.67. Oldjon meg egy bináris osztási számpéldát ismételt kivonásos módszerrel.
E.10.68. Mutasson be egy P-P-osztót.
E.10.69. Mutasson be egy P-S-osztót.
E.10.70. Mutasson be egy S-P-osztót.
E.10.71. Mi a *tört-hányados*, és mi az *egész-hányados*?
E.10.72. Milyen elven működik egy *tört-hányados*t előállító osztóberendezés?
E.10.73. Oldjon meg egy számpéldát tört-hányadosú bináris osztásra.
E.10.74. Milyen elven működik egy *lebegőpontos* bináris szorzóberendezés?
E.10.75. Milyen elven működik egy lebegőpontos bináris osztóberendezés?
E.10.76. Milyen elven működik egy lebegőpontos bináris összeadó/kivonó berendezés?

E.10.77. Mi a *komparátor,* milyen elveken építhető fel?
E.10.78. Milyen az *aritmetikai* összehasonlítás?
E.10.79. Milyen a *logikai* összehasonlítás?
E.10.80. Mutasson be egy soros bináris komparátort.
E.10.81. Mutasson be egy 2-bites bináris komparátort.
E.10.82. Mit tud a komparátorok modulokból történő összekapcsolásáról?
E.10.83. Rajzolja fel egy memória elvi vázlatát.
E.10.84. Milyen szempontok szerint csoportosíthatók a memóriák?
E.10.85. Mit jelentenek a következők: RWM, RAM, SAM, CAM, ROM, EPROM?
E.10.86. Milyenek a sztatikus- és a dinamikus memóriák, és milyen cellákból épülnek fel?
E.10.87. Milyen egységekből áll egy memória?
E.10.88. Mit jelent a *bit*-szervezés és a *szó*-szervezés?
E.10.89. Rajzolja fel egy R/W *szószervezésű* memória blokkvázlatát.
E.10.90. Mit tud a *soros üzemű* cella-szervezésű memóriákról?
E.10.91. Mire használhatók az *aszociatív* memóriák?
E.10.92. Rajzoljon fel egy aszociatív memóriákra jellemző blokkvázlatot.
E.10.93. Milyen elveken bővíthetők a modulokból felépített memóriák?
E.10.94. Milyen eltérésekkel találkozunk a ROM és RWM memóriák bővítésekor?
E.10.95. Mi a *Digital-Analóg* konverter, és milyen az átviteli karakterisztikája?
E.10.96. Milyen összetevőkből épül fel egy D/A konverter?
E.10.97. Milyen kapcsoló- és ellenálláshálózat alapváltozatokat alkalmaznak D/A átalakítóknál?
E.10.98. Ismertessen egy jellegzetes D/A átalakító-megoldást!
E.10.99. Mi az *Analóg-Digitál* konverter, és milyen az átviteli karakterisztikája?
E.10.100. Ismertesse az A/D átalakítással kapcsolatos jeltípusokat!
E.10.101. Mi az a *mintavételezés* és a *kvantálási zaj*?
E.10.102. Milyen alaptípusok alapján csoportosíthatók az A/D átalakítók?
E.10.103. Ismertessen egy jellegzetes A/D átalakító-megoldást!
E.10.104. Mi az a *szukcesszív approximáció,* A/D esetén?

11. SZOFTVER ÉS HARDVER VEGYES ALKALMAZÁSÁN ALAPULÓ DIGITÁLIS RENDSZEREK

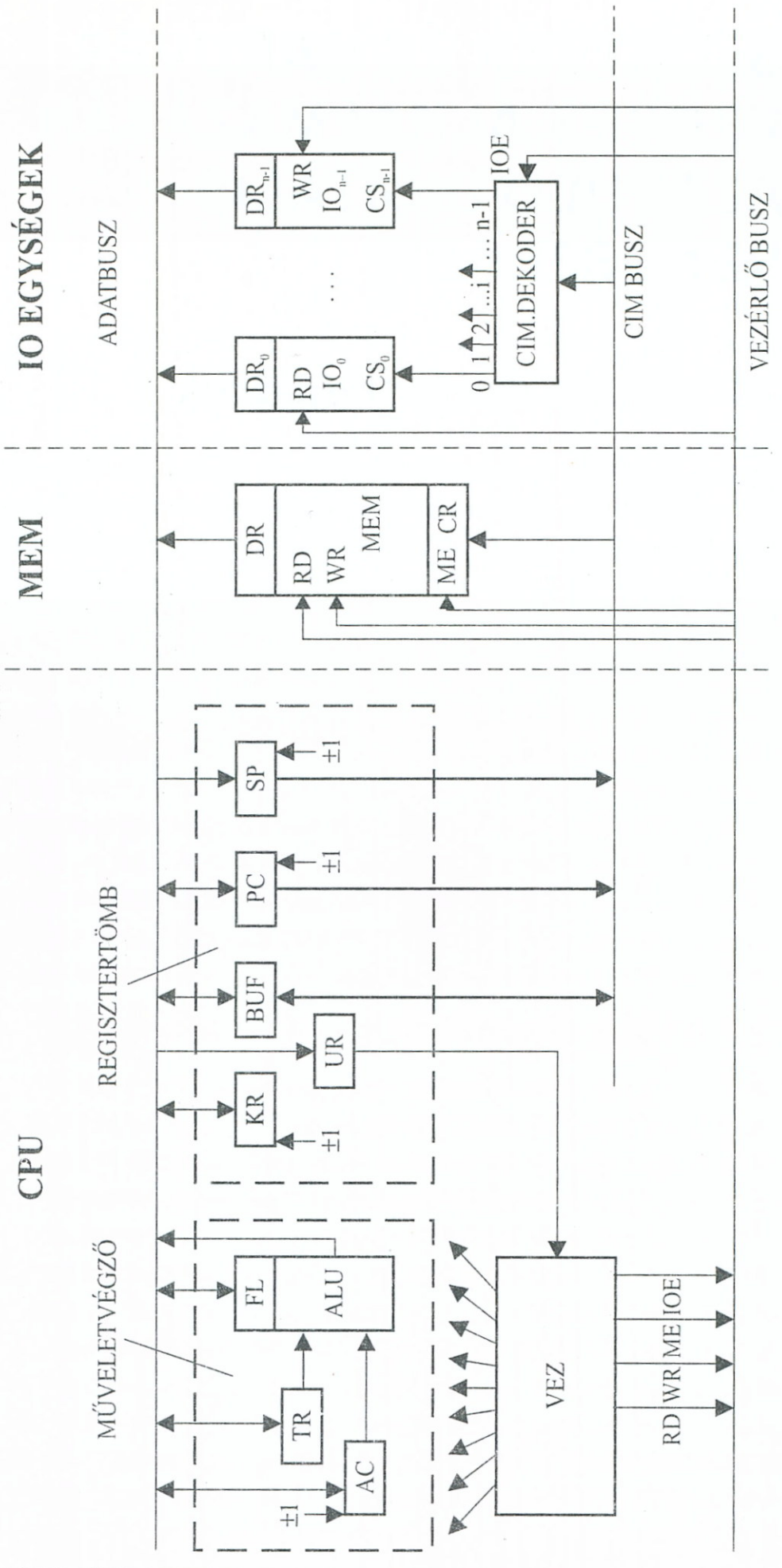
11.1. Általános meggondolások

A korszerű digitális rendszerekben szoftver- és hardver összetevők együttesen kerülnek alkalmazásra. Az ilyen rendszerek működése a 2–2b. ábra elvi vázlatán alapul, ahol a működés módját meghatározó *szoftver*-információk és a *hardver*-eszközök hatékony együttműködését a rendszerben szereplő *vezérlőegység* biztosítja. A vezérlőegység működését *program* irányítja, mely érkezhet külső információforrásból, vagy elhelyezkedhet a rendszer memória egységének egy elkülönített részében.

A klasszikus berendezésekben a „működtetési programot" közvetlen *emberi irányítás* szolgáltatta, például a kezelői kiképzéskor megtanult és saját memóriájában tárolt ismeretek révén, vagy egy „kezelési utasítás"-ban tároltak alkalomszerű közvetítésével. Később, a működési programokat már *külső adathordozók*ra rögzítették (pl. lyukszalag, mágneses stb. tárolóeszközök) és ezek beolvasásával jutottak az aktuális feladat megoldásához szükséges program információihoz.

A korszerű digitális berendezéseknél (pl. digitális, univerzális számítógépekben, mikroprocesszorokban) az ún. NEUMANN-elv értelmében, a program-információkat a berendezés memória egységének egy erre a célra elkülönített részében (a programmezőben) helyezik el, miáltal a rendszer számára magasabb automatizálási szintet, jobb flexibilitást és a nagyobb működési sebességet lehet biztosítani.

A feladatonkénti más-más program követelményeihez feladatonként átszervezhető *univerzális hardver* elrendezésre van szükség, melyet busz-szervezésű (5–5c. ábra) struktúrákkal lehet biztosítani. Az egyes hardver egységek alkalomszerű buszra, vagy egymáshoz történő csatlakoztatását a *vezérlőegység* végzi el, összhangban a *szoftver-program* lépéseivel. Az előző fejezetek tapasztalatai azt mutatták, hogy a digitális berendezések főként: logikai, aritmetikai,



11-1. ábra *Egy szoftver-hardver vegyes alkalmazásán alapuló busz-szervezésű* ***univerzális digitális rendszer*** *blokkvázlata*

adatmozgatási, adatátalakítási műveletek keretében oldják meg feladataikat. Ennek megfelelően egy univerzális hardver egységnek elsősorban ilyen funkcionális egységekből kell felépülnie.

Egy ilyen univerzális, digitális hardver elrendezést rajzoltunk fel a 11–1. ábrán. A berendezés – mint látható – buszos szervezésű és a 8., ill. 10. fejezetekben megismert vezérlő, regiszter, műveletvégző, memória stb. funkcionális egységekből épül fel. A funkcionális egységeket, itt betöltött feladataik alapján:

A) CPU (CENTRAL PROCESSING UNIT)
B) MEMORIA (MEM)
C) INPUT–OUTPUT EGYSÉGEK (IO)

csoportokba sorolhatjuk. Az egyes egységek közötti összeköttetést *feladatok szerint* elkülönített, háromféle busz biztosítja. A buszok, a hozzájuk csatlakozó egységek alkalomszerű fel/le-kapcsolhatóságának érdekében általában *három állapotú* (5.1.2. pont) csatlakozásokkal rendelkeznek.

Az ADATBUSZ, a MEMORIA, illetve az INPUT–OUTPUT egységek kiolvasott vagy beírandó információit továbbítja a CPU–MEM–IO egységek között.

A CIMBUSZ, a MEM adott rekeszéhez, vagy az aktuálisan kiválasztandó IO_i megcímzéséhez szükséges információkat továbbítja.

A VEZÉRLŐ BUSZ a rendszer működéséhez nélkülözhetetlen vezérlő információkat hordozza. [Például RD-kiolvasás (READ), WR-beírás (WRITE), ME-memóriára vonatkozó, IO–Input–Output-ra vonatkozó stb.] Ezek az információk a VEZÉRLŐ egységből érkeznek.

11.2. A rendszer hardver összetevői

Vizsgáljuk meg a következőkben a 11–1. ábra hardver összetevőinek feladatkörét és működését.

11.2.1. *CPU–Central Processing Unit* (Központi feladatmegoldó egység)

több összetevő alegységből épül fel, melyek eltérő funkciókkal rendelkeznek.

A.1. A VEZ-jelű VEZÉRLŐ egység a 11–1. ábrán látható valamennyi funkcionális egységgel összeköttetésben áll. (Az egyes egy-

ségekhez tartó vezetékeket részleteiben nem ábrázoltuk.) Ennek az „idegrendszernek" a segítségével az egyes blokkokat, a memóriában elhelyezkedő program egyes *utasítás*ainak megfelelően (mely utasítások az UR–*utasításregiszter*en keresztül jutnak a vezérlőbe), adott időpontban, a kívánt üzemmódban működteti, ill. a buszokra rákapcsolja vagy kiiktatja őket. A vezérlő egységekkel a 8. fejezet 8.4. pontjában részletesen foglalkoztunk és már ott láttuk, hogy ezek *mikroprogram-tár* egységében egy adott feladatvégrehajtás állapotgráfjának információi tárolhatók. Ez egy adott CPU esetében azt jelenti, hogy vezérlőegységében előre tárolva vannak a CPU által végrehajtható összes utasítás algoritmusai, és bármelyik utasítás aktuálissá válásakor elegendő a vezérlőegységgel az adott utasításra jellemző, ún. *műveleti kód*ot (MK, vagy OC–Operation Code) közölni, aminek hatására a vezérlő egység automatikusan elvégezteti a kijelölt művelet végrehajtását. (Pl. kiolvassa az adatot a memóriából és összeadatja az AC-regiszter tartalmával stb.)

A.2. A MŰVELETVÉGZŐ EGYSÉG több alegységből tevődik össze:

a.2.1. Az *ALU Aritmetikai, logikai műveletvégző* egyesíti magában a 10. fejezetben bemutatott műveletvégzők képességeit és itt általában *két-operandusos* műveletvégző változatban kerül megvalósításra. Nevének megfelelően alkalmas például: összeadási-, kivonási- (esetleg: szorzási-, osztási-, ill. bonyolultabb) *aritmetikai műveletek,* továbbá rendszerint ÉS-, VAGY-, ANTIVALENCIA-, NEGÁCIÓ *logikai műveletek* végrehajtására, de képes lehet még például *komparátor* funkciók ellátására is. Amennyiben a műveleteknél *járulékos* információk (pl. átvitel [CY], előjel [S], paritásbit [P], túlcsordulás [OVF], hiba [ERR], zéró eredmény [Z] stb.) ún. FLAG-bitek is megjelennek, akkor ezek az ALU-hoz kapcsolódó *FLAG-regiszterben* kerülnek tárolásra.

Egy FLAG-regiszter bit-elrendezését mutatja a 11–2. ábra:

FL:	S	Z	...	OVF	P	ERR	...	CY

11–2. ábra *FLAG-regiszter bitjei*

a.2.2. Az *AC Akkumulátor*-regiszter egy univerzális képességű regiszter, mely rendszerint P–P, S–S, S–P és P–S üzemmódban is tud működni, így (szükség esetén) *shift*elési feladatok ellátására is alkalmas. Funkcionálisan legfontosabb szerepe az ALU egyik operandu-

sának a művelet előtti átmeneti tárolása és egyszerűbb CPU-kban (mint az ábrán is látható) a művelet utáni eredmény átmeneti tárolása. Az AC-regiszter – mint látni fogjuk – számos szoftver utasítással is elérhető.

Az AC akkumulátor és az FL flag-regiszter együttesét gyakran PROGRAM ÁLLAPOT SZÓ (PSW–PROGRAM STATUS WORD)-nak is nevezik, mely a műveletsorozat adott pillanatbeli legfontosabb információit tárolja.

a.2.3. A *TR Operandus-regiszter* az ALU másik operandusának átmeneti tárolására szolgál.

A.3. REGISZTER TÖMB összetevői a következők:

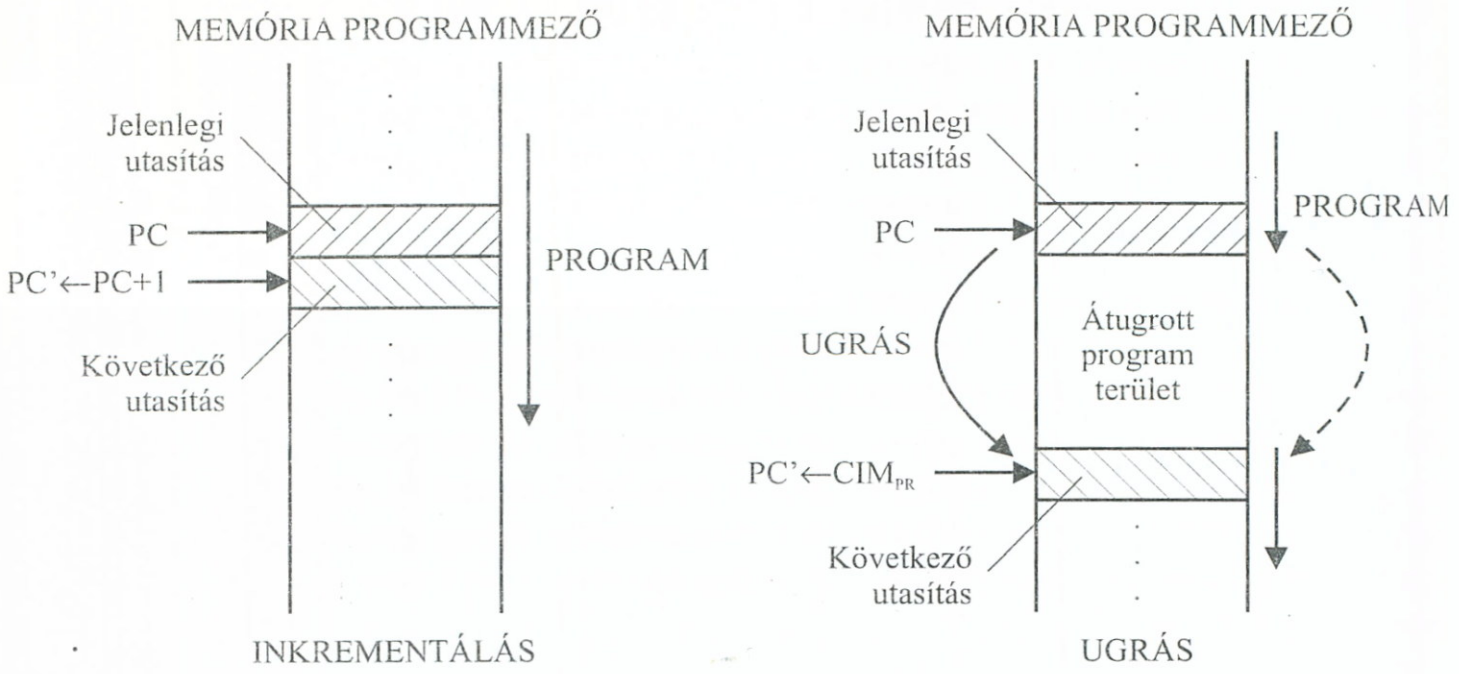
a.3.1. Az *UR Utasítás-regiszter*ben kerül átmeneti tárolásra az *M–Memória* programmezejéből kiolvasott aktuális utasítás, melynek MK műveleti kódját kell az UR-ből a VEZ vezérlőbe küldenünk, hogy a soron következő feladatot a VEZ felismerje, majd ezt követően a rendszerrel végre tudja hajtatni.

a.3.2. A *KR Kiegészítő regiszter* az utasításvégrehajtás címzési feladataiban kap szerepet, továbbá átmeneti tárolási feladatokat lát el, és tartalma általában inkrementálható, ill. dekrementálható.

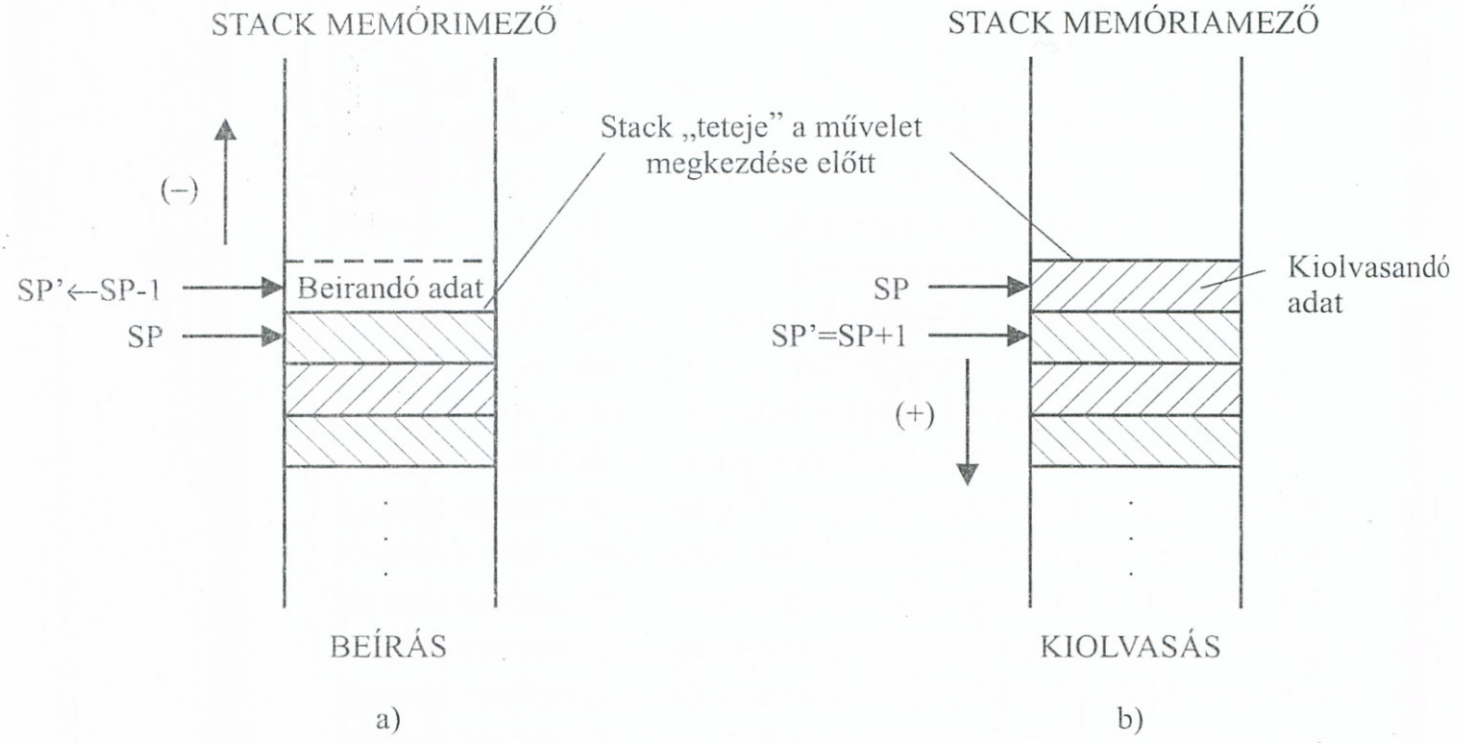
a.3.3. A *BUF Átmeneti tároló* (BUFFER) regiszter az egyes utasítások végrehajtási folyamatában kap szerepet, és jelen CPU példában átviteli hidat képez az ADAT- és CÍM-buszok között is.

a.3.4. A *PC Programszámláló* (Program Counter, egyes cégek leírásaiban IP–Instruction Pointer) valójában egy párhuzamosan preszetelhető és inkrementálható számláló, melynek mindenkori állása mindig annak a programmező-beli utasításnak memóriacímét tartalmazza, amelyet a következő lépésben meg kell majd kezdenünk. Ha a PC-t inkrementáljuk, akkor a programban a sorrendben következő utasítás kerül végrehajtásra. Ha a PC tartalmát párhuzamosan preszetelve „kívülről" átírjuk CIMPR-re, akkor a programban „ugrás" történik és a *preszetelés*sel (CIMPR-rel) megcímzett utasítás kerül végrehajtásra, így a program innen folytatódik tovább. Az átugrott programterület ilyenkor nem kerül felhasználásra. Az elmondottak a 11–3. ábrán követhetők.

a.3.5. Az *SP Stack Pointer* működési elve hasonló a PC-hez, annyiban, hogy szintén egy elkülönített memória-terület megcímzésére szolgál. Ezt az elkülönített memória-területet STACK MEMORIA-nak nevezzük. A stack memóriát különféle információk (címek, adatok) nem túl hosszú időtartamra történő tárolására használjuk. A tárolás meghatározott sorrendben történik, mintha egy függőleges „ve-



11-3. ábra *A következő utasítás megcímzése*



11-4. ábra *STACK beírási és kiolvasási művelet*

rem"-be (stack = verem) egymás fölé helyeznénk el a tárolandó adatokat. A 11–4. ábráról leolvasható a működési elv: *Beíráskor* az SP-t dekrementáljuk és erre az új című STACK-helyre (az eddigi adatok fölé) tároljuk az új információt. *Kiolvasáskor* először kiolvassuk a legfelső adatot, majd az SP-t inkrementálva állítjuk be az újonnan „legfelsővé" vált adat címét.

11.2.2. MEMORIA-egység

A memóriák, mint funkcionális egységek felépítésével és működésével a 10. fejezet 10.8. pontjában részletesen foglalkoztunk. A 11–1. ábrán látható szoftver- és hardver összetevőket egyaránt alkalmazó digitális berendezésnél egy *szó-szervezésű* memória szerepel, mely általános esetben mind RAM, mind ROM cellamezőket tartalmaz. A memória elméleti szókapacitása:

$$N = 2^n \text{ szó} \qquad (11.1)$$

ahol: n – a CR címregiszter bitben számított hossza.

A memória számos, felhasználási szempontból elkülönített tárolóterületet foglal magában. Ezekre utal a 11–5. ábra, mely az eddigiekben megemlített memóriamezőket szemlélteti.

11–5. ábra *Különféle feladatok céljaira elkülönített memória-területek*

A memóriának a teljes rendszerhez történő csatlakoztatása jól követhető a 11–1. ábrán. A kiszemelt memóriarekesz címe a CÍM Busz-on keresztül érkezik a CR CÍMREGISZTER-re, mely aktíválja a megcímzett helyet. Attól függően, hogy a VEZÉRLŐ-egységről *RD olvasási,* vagy WR írási utasítás érkezik a DR ADATREGISZTER (DATAREGISZTER) kifelé-, vagy befelé irányban kapcsolódik az ADAT BUSZ-ra.

A memória címregisztere csak akkor aktivizálódik, ha a VEZÉRLŐ egységről ME (MEMORY ENABLE) engedélyező jel érkezik. Ily módon a CÍM BUSZ memóriára, illetve IO EGYSÉG-ekre vonatkozó cím információit a vezérlőegység szét tudja választani.

11.2.3. IO – INPUT–OUTPUT egységek

Az ebbe a csoportba tartozó egységek többirányú feladatokat láthatnak el. Ezek közül kiemelhetők a következők:

- A CPU–MEM együttes „külvilággal" való kapcsolatainak szervezése. Például: külső jeladó áramkörök (pl. nyomógomb) és kijelzők (pl. egy lámpa) csatlakoztatása, S–P, P–S, A/D, D/A átalakítási feladatok, jel-átviteli időzítések, szabványos jel-átviteli rendszerekhez történő illesztés, aszinkron/szinkron működési feltételek biztosítása stb.
- A CPU működési adottságainak szélesítése. Például: a CPU alapkiépítésben korlátozott műveletkészletének kibővítése, áttérés más számábrázolásban történő számolásra stb.
- A MEM alapegység kapacitásának növelése, *háttértárak* képzése. Ezekkel kapcsolatos további feladatot jelenthet a MEM és IO-egységek közvetlen kapcsolatának megszervezése (a CPU közreműködésének kihagyásával), mely gyorsabb információ-átvitelt tesz lehetővé. Az ily módon létrehozott *közvetlen memóriaelérés*t DMA-nak (Direct Memory Access) is nevezik.
- INTERFÉSZ (INTERFACE) feladatok ellátása, melyek keretében a különféle paraméter-jellemzőkkel leíró PERIFERIÁLIS EGYSÉGEKnek a CPU jellemzőkhöz való illeszkedését biztosítják.

A 11–1. ábrán szemléltettük az IO-egységeknek a teljes rendszerhez történő csatlakoztatását. Az egyes IO_i egységek CS (CHIP SELECT) lábaira adott jelekkel aktivizálható valamely kiválasztott IO-egység. A kiválasztás a CÍM BUSZ-on keresztül történik egy CÍMDEKÓDER áramkör segítségével (mely pl. egy 10. fejezetben bevezetett DMX-hez hasonló felépítésű lehet). A megcímzett IO_i egységnek még rendszerint kell kapnia egy *RD-olvasás*-engedélyezési, vagy egy *WR-írás*-engedélyezési parancsot is, mely után a DR_i (DATA REGISZTER) az ADATBUSZ-szal megfelelő irányítottsággal kapcsolatba kerül. A rajzon példaként felrajzolt IO_0-ból *olvasni,* az IO_{n-1}-be pedig írni lehet, természetesen léteznek írható és olvasható típusú IO-egységek is, melyeknél mind RD, mind WR-vezérlőjel előfordul. A CÍMDEKÓDER még el van látva egy IOE-input/output engedélyezési kapuzójellel, mely biztosítja a memória címzési jelek és az IO címzési jelek szétválasztását a VEZÉRLŐ-egység révén.

11.3. A rendszer szoftver összetevői

Mint a bevezetőben említettük, a rendszer szoftver összetevői a PROGRAM-hoz kötődnek, mely a memória PROGRAMMEZŐ-ben van tárolva a működés folyamán. Előfordulhat, hogy a program állandóan a memóriában van, de olyan megoldás is alkalmazható, hogy csak felhasználás előtt olvassuk be valamely „háttér"-tárolóból, mivel ez esetben a memória ún. OPERATÍV MEMÓRIA részét gazdaságosabban ki lehet használni.

11.3.1. Program és utasítások

A PROGRAM-ot UTASÍTÁS-ok egymásutáni sorozatából építjük fel, egy-egy utasítás valamely *műveletet* végeztet el a berendezéssel a VEZÉRLŐ-egység közreműködésével. A műveleteket:

- adatátviteli,
- aritmetikai,
- logikai,
- összehasonlítási,
- ugrási,
- léptetési,
- vezérlési

típuscsoportokba sorolhatjuk. A művelet elvégzéséhez kialakítandó utasítás szerkezete (formátuma) többféle lehet. Például egy *két-operandusos* műveletnél elméletileg megadható a 11–6. ábra szerinti, ún. *négycímes utasítás formátum,* mely a programbeli adott műveletel kapcsolatos összes információt tartalmazza.

A 11–6. ábrabeli utasítás öt szektorból áll. Az első szektor MK (műveleti kód) vagy OC (Operation Code) *kódolt* formában megadja az elvégzendő művelet típusát, amit az OP1 és OP2 *operandusok* között el kell végezni. A második és harmadik szektor megadja azoknak a helyeknek a *címét* (CÍM 1, CÍM 2), ahol az operandusok megtalálhatók. Ezek a helyek lehetnek a *memóriá*ban, de lehetnek például va-

1.	2.	3.	4.	5.
MŰVELET TÍPUSA OC, MK	1. OPERANDUS CÍME CÍM 1	2. OPERANDUS CÍME CÍM 2	EREDMÉNY CÍME CÍM R	A KÖVETKEZŐ UTASÍTÁS CÍME A PROGRAMBAN CÍM KU

11–6. ábra *Négycímes utasítás-formátum*

lamelyik regiszterben (pl. AC-akkumulátor) előre „odakészítve" is. A negyedik szektor a művelet után kapott *eredmény* elhelyezési címét (CÍM R) tartalmazza (R-Result), mely utalhat ismét memóriára, de pl. lehet ismét egy regisztercím is. Az ötödik szektorbeli információ a memória *programmező* egy címe (CÍM KU), konkrétan az ezután megoldandó feladathoz tartozó utasításé. Ha ez a cím számszerint a most megoldott utasítás címe után következik, akkor *inkrementálás*ról, ha egy távolabbi címről van szó, akkor *ugrás*ról beszélünk, mint ezt már korábban a 11–3. ábrában bemutattuk.

A négycímes utasítás a gyakorlatban nehezen kezelhető lenne, ezért egyszerűbb utasítás-formátumokat alkalmaznak. Elterjedt (különösen mikroprocesszoros realizációknál) az ún. *egycímes* utasításformátum, melyet a 11–7. ábrán szemléltetünk. Mint látható, ez csak két szektort tartalmaz. A változatlan MK műveleti kód mellett a második szektorban egy *többcélú* mennyiséget az ún. DISPLACEMENT-et vezettük be, melybe *különféle típusú* információk (pl. Operandus címe, maga az operandus, programmező-beli cím stb.) helyettesíthetők be. Ily módon egy lényegesen rövidebb és rendkívül rugalmasan alkalmazható utasítás-formátumhoz jutottunk. A dolognak – természetesen – „ára van", mivel azt a feladatot, amit a négycímes utasítással egyetlen lépéssel meg tudunk oldani, az egycímes utasítás típussal csak több lépésben érhetjük el. Ennek ellenére a korszerű univerzális, digitális berendezésekben az egycímes utasítások rendkívül elterjedtek.

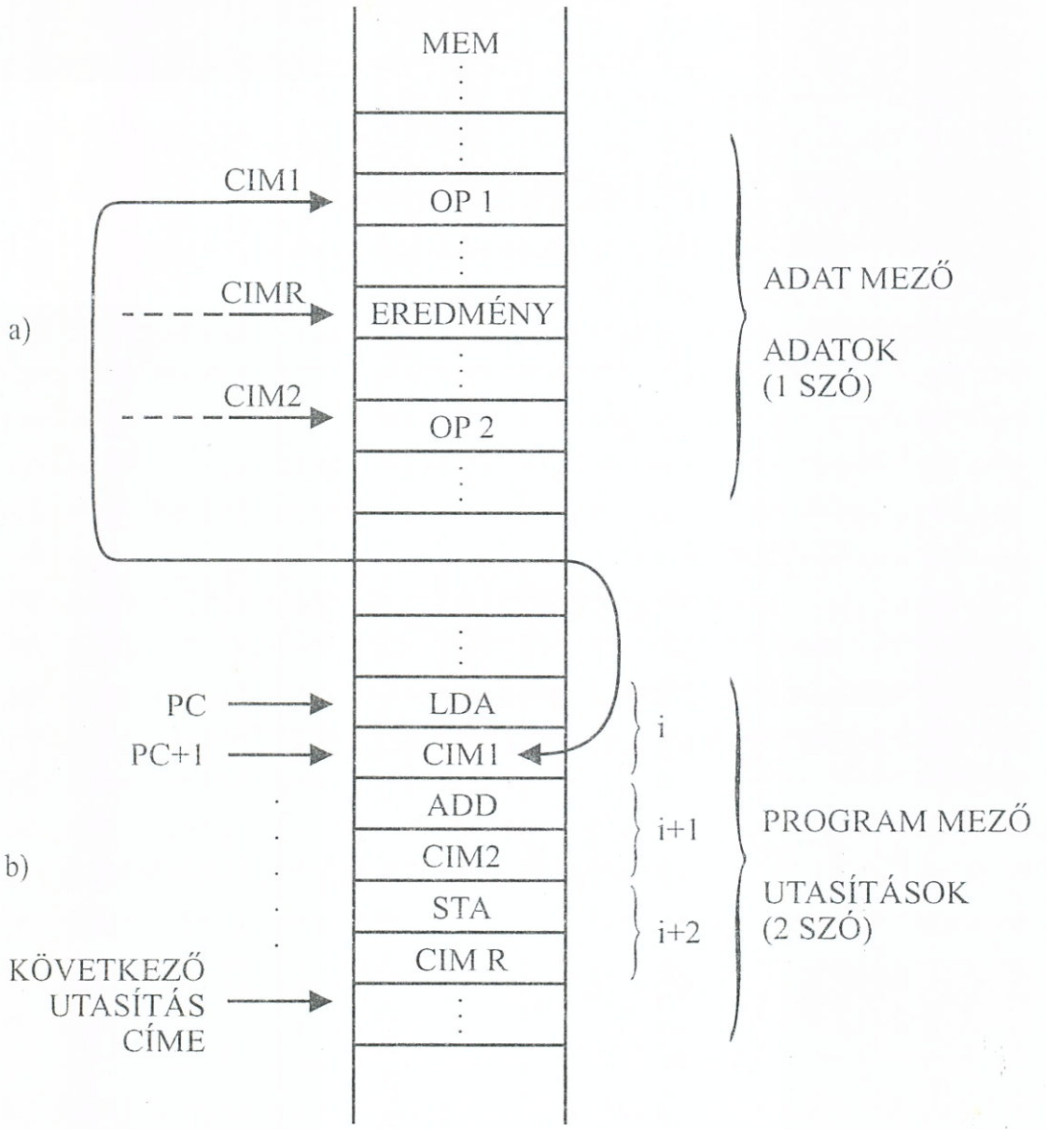
1.	2.
MŰVELET TÍPUSA MK	DISPLACEMENT D

11–7. ábra *Egycímes utasításformátum*

11.3.2. Az utasításvégrehajtás hardver lépései és az utasításkészlet

A következőkben vizsgáljuk meg, hogy milyen lépések történnek a hardverben egy-egy szoftver utasítás végrehajtása során.

A konkrétabb tárgyalhatóság érdekében, tételezzük fel, hogy a 11–1. ábra univerzális, digitális berendezésében 16-bites szóhosszúságú: valamennyi regiszter, a memória, az IO-egységek, az ADAT BUSZ és a CÍM-BUSZ, továbbá 16-bites operandusokkal dolgozik



11-8. ábra *Adatok és utasítások a memoriában*

az ALU is. Az utasításformátum legyen *egycímes,* olymódon, hogy mind az MK rész, mind a D rész külön-külön 16-bites méretű. Miután a programot a MEM *program* mezejében kell elhelyezni – és mint előbb rögzítettük – egy memória szó hossza 16-bites, ezért az MK+D = 32 bit hosszú utasítást csak a 11–8b. ábra szerinti módon tudjuk a MEM *programmező*ben „összegyűrve" tárolni.

Az adatok csak 16-bitesek, ezért ezek egy-egy memória szóban tárolhatók a MEM *adat*mezőben, a 11–8a. ábrarész szerinti módon.

Az utasítások műveleti kódjának megnevezésére célszerű bevezetni egy egyszerű, *ember által* is jól „követhető" szimbolikus nemzetközi nyelvet (ASSEMBLY-nyelv), mely az elvégzendő műveletek angol szavainak rövidített változataiból lett összeállítva.

Például a 11–8b. ábra i-jelű utasításánál az a feladat, hogy a CÍM 1-gyel megcímzett OP 1 adatot vigyük át az AC akkumulátorba. Itt az MK műveleti kód ASSEMBLY nyelven lehet például a következő:

LDA (LoaD to Accumulator)

Természetesen a mi digitális berendezésünk ezt egy *bináris kód* formájában találja az MK helyén. Az

LDA ↔ bináris kód

összerendelést pedig eredetileg a berendezés *tervezésekor* állapították meg, amikor a gép utasításkészletét összeállították. Az utasítások műveleti kódjaira vonatkozó bináris táblázatot *gépi kódtáblázat*nak nevezik és az univerzális, digitális berendezés (például egy számítógép) műszaki specifikációjában megadják rendszerint HEXA formában. Például a 11–1. ábra példabeli gépére vonatkozóan (16 bit) az LDA kódja legyen:

$$\begin{array}{lcccccc} \text{LDA} : & 0010 & 1101 & 1110 & 1111_{\text{BIN}} & = & 2\text{DEF}_{\text{HEXA}} \\ & 2 & D & E & F & & \end{array}$$

Az ASSEMBLY-program klasszikus módon történő papír–ceruza írásakor (az OP1 címének az ugyancsak HEXA-ban kifejezett, pl. CÍM 1 = $43\text{B}8_{\text{HEXA}}$ feltételezésével), ahol a D-displacement helyére most a CÍM 1 kerül, így a kérdéses teljes utasítás a következőt takarja:

LDA CÍM 1 : 2DEF, $43\text{B}8_{\text{HEXA}}$

Ezt, a VEZÉRLŐ-egység logikai áramkörei egy HEXA/BINÁRIS átkódolás után, például komparátor (lásd: 10. fejezet 10.7. pont) közreműködésével már azonosítani tudják.

A program felhasználó által történő írása során természetesen az ASSEMBLY nyelvet használhatjuk, és az ASSEMBLY utasításokat manapság egy FORDÍTÓ-program konvertálja a gép számára is érthető bináris kódokká. Napjainkban néha már az ASSEMBLY-ben való programírás is túl részletezettnek bizonyulhat, ezért kidolgozták a magasabb szintű nyelvekben megírt programoknak ASSEMBLY, ill. közvetlen GÉPI KÓDBA történő konvertálását.

Térjünk vissza ezután a 11–8. ábrán szereplő feladathoz, ahol egy ASSEMBLY-ben megírt programrészletet találunk (i, i +1, i +2 utasítások) a velük kapcsolatos memória ADATMEZŐ-részlettel. Az ASSEMBLY programot olvasva a következők derülnek ki:

i) lépés: LDA CÍM1 (LoaD to Accumulator, CÍM1).
Vigyük át a CÍM1-gyel megcímzett adatot (OP1) az AC-akkumulátorba.

i +1) lépés: ADD CÍM2 (ADDITION, CÍM2).
Adjuk hozzá az AC pillanatnyi tartalmához (OP1) a CÍM2-vel megcímzett adatot (OP2), és az eredményt írjuk be AC-be.
i +2) lépés: STA CÍMR (STore Accumulator, CÍMR).
Tároljuk be a memória adatmezejében a CÍMR címre az akkumulátor pillanatnyi tartalmát.

A példát összehasonlítva egy olyan esettel, ahol *négycímes* utasítást használhattunk volna, megállapítható, hogy ott egyetlen utasítás magában foglalta volna mindazokat a tevékenységeket, melyeket itt három *egycímes* utasítással oldottunk meg. Látható az is, hogy az egycímes esetben esetenkénti *átmeneti* tárolásokra volt szükség az AC felhasználásával, és ennek során az AC-ben az eredménnyel néha korábbi adatokat is felülírtunk. A másik fontos megjegyeznivaló az, hogy a „négycímes" CÍM KU (*következő* utasítás címe) meghatározását itt egy külön HW-elem a PC végzi, olymódon, hogy a műveletek során újra és újra inkrementálódva mindig a *következő* utasítást címezi meg.

Vizsgáljuk meg ezután, hogy a 11–1. ábrán látható univerzális, digitális berendezés egyes HW-összetevői miként vesznek részt a különféle utasítások végrehajtásában.

a) *LDA CÍM1* utasítás – mint láttuk – az OP1 adatot viszi át az AC-be. Ezt a továbbiakban az alábbi módon jelöljük:

$$AC \leftarrow OP1$$

Ezt az adatátviteli műveletet a 11–1. ábra hardverje több elemi lépésben hajtja végre. Elsőként ki kell olvasni a programmezőből (11–8. ábra) az MK műveleti kódot (amiről most induláskor még nem tudjuk, hogy mi a jelentése), és az UR-en keresztül eljuttatva a VEZ-be, ott *komparálással* fel kell ismernünk, hogy MK = LDA. Ezután ki kell olvasni CÍM1-et, majd ezzel a CR címregiszteren keresztül a MEM-et megcímezve, onnan ki kell olvasni az OP1-et, mely végül a DR-en keresztül az AC-be kerülhet.

Az eddig elmondottakat egzaktul az ún. HARDVER LEÍRÓ NYELV-ek segítségével lehet követni, olyan részletességgel, hogy a GÉP működésének minden mozzanatát szemléltetni tudjuk. Hardver leíró nyelveket használnak például a számítógépek tervezési folyamatában szigorú matematikai algoritmizált feltételek között. Számunkra most ez nem szükséges, ezért csak néhány „műveletet" veze-

CIKLUS	ELEMI HARDVER MŰVELET	MAGYARÁZAT
1. (FETCH)	CR ← PC	– Az MK programmező-beli címét átírjuk CR-be
	INR PC	– A PC-t INKREMENTÁLÁSsal előkészítjük a következő rekeszbeli CÍM1 kiolvasásához (INR PC = PC + 1)
	DR ← (CR)	– Kiolvassuk MK-t a programmezőből
	UR ← DR	– Átvisszük MK-t az UR-be
	VEZ ← UR	– Betöltjük MK-t a VEZ-be, ahol a vezérlő *komparálással* felismeri, hogy MK = LDA, ezután a gép *már tudja,* hogy milyen típusú utasítást hajt végre. Az eddig lezajlottakat UTASÍTÁS-LEHÍVÁSI CIKLUSnak, más néven: FETCH-nek nevezzük. Minden a FETCH-csel kezdődik.
2.	CR ← PC	– CÍM1-et tartalmazó rekesz címét átírjuk CR-be
	INR PC	– PC-t előkészítjük a soron következő utasítás MK-jának kiolvasásához
	DR ← (CR)	– Kiolvassuk CÍM1-et a programmezőből, mivel CÍM1 mutatja meg OP1 helyét az adatmezőben, ezért:
	BUF ← DR CR ← BUF	– DR-t a BUF-regiszter segítségével áttöltjük CR-be, így előkészítettük az OP1 kiolvasását
3.	DR ← (CR)	– OP1 kiolvasása az adatmezőből
	AC ← DR	– OP1 áttöltése AC-be és ezzel az AC ← OP1 művelet befejeződött

11–9. ábra *Az LDA CÍM1 utasítás végrehajtásának részletezett mozzanatai a HW-egységeknél*

CIKLUS	ELEMI HW MŰVELET	MAGYARÁZAT
1. (FETCH)	FETCH	– Mindig egyforma
2.	⋮	– A 2. ciklus is megegyezik, csak itt CÍM2 szerepel
3.	DR ← (CR)	– OP2 kiolvasása az adatmezőből
	TR ← DR	– OP2 áttöltése TR-be
	AC ← AC+TR	– AC ← OP1 + OP2 összeadás elvégzése, az összeg tárolva marad AC-ben

11–10. ábra *Az ADD CÍM2 utasítás végrehajtásának részletezése*

tünk be az alábbiakban, amikkel az információáramlást a berendezés egységei között szemléltetni tudjuk. Ezek a következők:

Átvitel:	X-ből Y-ba	: $Y \leftarrow X$
Kiolvasás:	X címről a MEM-ből és a kiolvasott információ átvitele Y-ba	: $Y \leftarrow (X)$
Beírás:	Y-ról érkező információ beírása MEM-be az X címre	: $(X) \leftarrow Y$

Alkalmazva a bevezetett műveleteket a LDA CÍM1 utasításnak a 11–1. ábra berendezésén történő végrehajtásának követésére a 11–9. ábrát adhatjuk meg, mely a részműveletek mellett párhuzamosan tartalmazza a magyarázatokat is.

b) *ADD CÍM2.* Itt a feladat:

$$AC \leftarrow OP1 + OP2$$

A megoldás során ügyelni kell arra, hogy az előző utasításban OP1-et már betöltöttük az AC-be, ezért az OP2-t csak a TR-be vihetjük át. A kapott eredményt az AC-ben fogjuk tárolni, így az eredetileg abban található OP1 itt felülíródik. A részműveleteket a 11–10. ábra mutatja.

c) *STA CÍM R.* Itt a feladat:

$$EREDMÉNY \leftarrow AC,$$

azaz a OP1 + OP2 összeg betárolása a MEM-beli EREDMÉNY-nek kijelölt helyre. A részletezett lépések a 11–11. ábrán láthatók

CIKLUS	ELEMI HW MŰVELET	MAGYARÁZAT
1. (FETCH)	FETCH	– Mindig egyforma
2.	⋮	– A 2. ciklus is megegyezik, csak itt CÍM R szerepel, mellyel az összeg betárolását készítjük elő
3.	DR ← AC (CR) ← DR	– A korábbi OP1 + OP2 összeg áttöltése DR-be – Az összeg beírása a MEM CÍM R-rel címzett EREDMÉNY rekeszébe

11–11. ábra *Az STA CÍM R utasítás végrehajtásának részletezése*

Vizsgáljunk meg ezután néhány további utasításvégrehajtást, néhány további fogalom bevezetésével kapcsolatosan.

d) *MOV B, A.* A feladat: regiszteradat-mozgatás (MOVE):

$$BUF \leftarrow AC$$

Ez az utasítás egy *regiszter-regiszter* átvitelre vonatkozik, és nincs kapcsolatban a MEM adatmezővel, sőt memóriaprogram-mezejében is csak egy memória szót foglal el, mivel valójában csak az MK-ból áll. Itt a részműveletek a 11–12. ábrán láthatók.

CIKLUS	ELEMI HW MŰVELET	MAGYARÁZAT
1. (FETCH)	FETCH	– Mindig egyforma
2.	BUF ← AC	– AC áttöltése BUF-ba

11–12. ábra *A MOV B, A utasítás végrehajtásának részletezése*

e) *MVI B, DATA.* A feladat: egy számszerű adat (DATA) betöltése (MOVE) a kijelölt regiszterbe (BUF):

$$BUF \leftarrow DATA$$

Az ilyen jellegű műveleteket IMMEDIATE műveletnek nevezzük, erre utal az MK-ban szereplő betű is (MoVe Immediate). Ez az utasítás kétszavas, mivel MK mellett, meg kell adni a kívánt ADAT-ot is. A műveletek a 11–13. ábrán láthatók.

CIKLUS	ELEMI HW MŰVELET	MAGYARÁZAT
1. (FETCH)	FETCH	– Mindig egyforma
2.	CR ← PC	– A DATA adatot tartalmazó rekesz címét átírjuk CR-be
	INR PC	– PC-t előkészítjük a soron következő utasítás MK-jának kiolvasásához
	DR ← (CR)	– Kiolvassuk az utasításban megadott DATA-t a programmezőből
	BUF ← DR	– DATA-t áttöltjük BUF-ba

11–13. ábra *Az MVI B, DATA utasítás végrehajtásának részletezése*

f) *JMP CÍM U* Itt a feladat: elugrás a programmező egy kijelölt című helyére (CÍM U) és a program ezután innen folytatódik. Ezzel

már foglalkoztunk a 11–3. ábrában, az ott szerepelt CÍM PR megegyezik a mostani CÍM U-val, tehát a teendő:

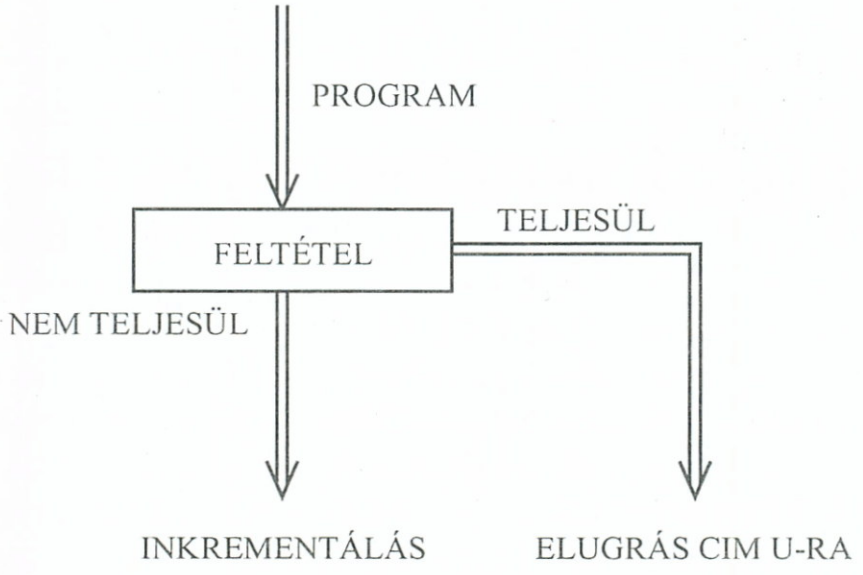
PC ← CÍM U

A részletezett műveletek a 11–14. ábrán láthatók. Ez az utasítás is kétszavas méretű.

CIKLUS	ELEMI HW MŰVELET	MAGYARÁZAT
1. (FETCH)	FETCH	– Mindig egyforma
2.	CR ← PC	– A pogrammező-beli CÍM U-nak *címét* átírjuk CR-be
	INR PC	– PC-t előkészítjük a soron következő utasítás MK-jának kiolvasásához
	DR ← (CR)	– Kiolvassuk az utasításbeli CÍM U-t a programmezőből
	PC ← DR	– A PC ← CÍM U átállítással végrehajtjuk az ugrást

11–14. ábra *A JMP CÍM U utasítás végrehajtásának részletezése*

g) *JC CÍM U.* Ez egy ún. *feltételes ugrás,* amelynél csak akkor valósulhat meg az ugrás, ha egy adott feltétel teljesül, ha nem teljesül, akkor a program inkrementálással halad tovább a 11–15. ábrának megfelelően:



11-15. ábra *A feltételes ugrás elve*

A FELTÉTEL jelei érkezhetnek közvetve a külső HARDVER-egységektől, de a feltételes ugró utasításokba rendszerint az FL FLAG REGISZTER (11–2. ábra) FLAG–BITjeit szokták beleépíteni. Jelen

példánkban a feltétel az átvitel CARRY–CY, melyet a JC műveleti kódban a C betű (Jump Carry) érzékeltet. Esetünkben tehát a feladat:

ha : CY = 1 UGRÁS CÍM U-ra
ha : CY = 0 NINCS UGRÁS, *inkrementálás*sal jöhet a következő utasítás végrehajtása

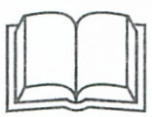
A részletek a 11–16. ábrán követhetők.

CIKLUS	ELEMI HW MŰVELET	MAGYARÁZAT
1. (FETCH)	FETCH	– Ugyanaz, mint a 11–9. ábra FETCH-ciklusa, csak itt a VEZ még megvizsgálja a CY = ? értéket is
2a.	CY = 0 esetén	– Mivel nincs ugrás, a *következő utasítás* kezdődik inkrementálás révén
2b.	CY = 1 esetén	– Ugyanaz történik, mint a JMP *feltétel nélküli* ugrás 2. ciklusában (11–14. ábra)

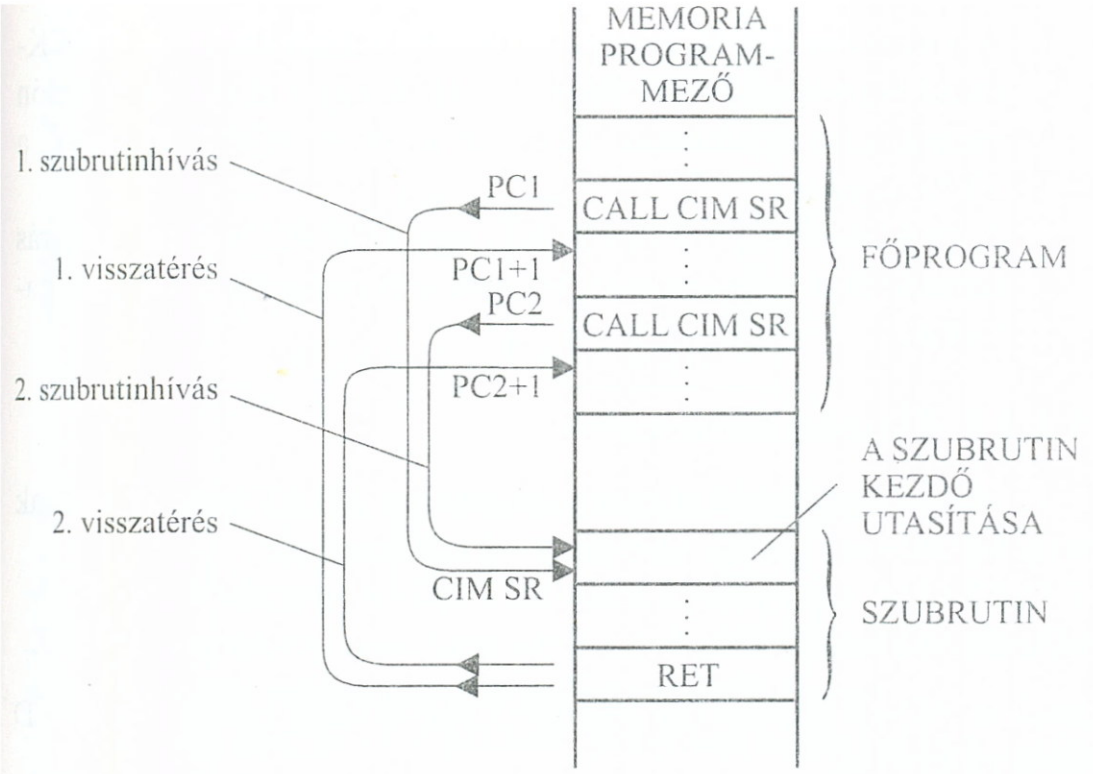
11–16. ábra *A JC CÍM U utasítás végrehajtásának részletei*

Az eddig bemutatott példák jól szemléltetik a szoftver- és hardver találkozását és együttműködését az univerzális, digitális berendezéseknél. Egy ilyen berendezés – a tárgyaltakon túlmenően – általában még igen nagyszámú és sokféle utasítással rendelkezik, melyek lefedik a 11.3.1. pontban felsorolt műveletek feladatkörét. Valamely univerzális, digitális berendezés utasításainak összességét UTASÍTÁSKÉSZLET-nek nevezzük. Vizsgáljunk meg néhány további utasítást.

11.3.2.1. A szubrutin és szubrutinhívás



Az univerzális, digitális berendezések programjaiban gyakran adódnak olyan azonos *rutin*feladatok, melyek többször is ismétlődnek. Például a gyökvonási rutinfeladatokat (melyek lépései minden alkalommal *ugyanazok,* és csupán a behelyettesítési értékeikben térnek el egymástól) előre meg lehet írni, a memória programmezejében tároltan el lehet helyezni és mikor szükségessé válnak, onnan előhívhatók. Az ilyen rutinfeladatokat megoldó programokat „szubrutin"-nak nevezik. A szubrutin használatára utal a 11–17. ábra.
Az ábrát tanulmányozva, első gondolatként felmerülhet, hogy a főprogramból egy JMP CÍM SR utasítással *ugorjunk* ki a szubrutinhoz. Ez a módszer a szubrutin lefutásának befejeződésekor szükségessé váló visszaugráskor jelentene problémát. Egyrészt amiatt, hogy a



11-17. ábra *Többszörös szubrutinhívás*

JMP kiugrási művelet alkalmával (PC ← CÍM SR miatt) szükségszerűen felülíródna az elugrás helye, ezáltal a gép „elfelejtené" a PC + 1 visszatérési címet. Másrészt *ismételt* szubrutinhíváskor a szubrutin végén levő visszaugrási utasításnál, mindig más- és más visszatérési címet kellene behelyettesíteni. Fentiek kiküszöbölésére két újabb utasítás bevezetése vált szükségessé, melyek formátumai:

CALL CÍM SR	(CALL – hívás)
RET	(RETURN – visszatérés)

A CALL utasítás két lépésből áll:

1. A visszatérési hely címének (PC_i + 1) *tárolása* rendszerint a STACK-memória mezőben. Ezt a lépést az utasításkészletekben önálló STACK *beírási* műveletként is értelmezik, ilyenkor a neve PUSH-művelet. („Belenyomás" a STACK-be.) E művelet részletes lépéseit már a 14–4a. ábrán végiggondoltuk. Itt a beírandó adat a PC_i + 1 cím értéke.
2. Elugrás a szubrutin kezdőcímére, egy automatikusan végrehajtott JMP CÍM SR-rel, ahonnan a szubrutin végrehajtása megindul.

A szubrutin futásának befejeződésekor elérjük a szubrutin utolsó utasításaként található RET-utasítást, mely szintén két lépésből áll:

1. A visszatérési hely címének (PC_i + 1) *kiolvasása* a STACK-mező legfelső helyéről (lásd: 14–4b. ábra). Ezt is szokás külön utasításként értelmezni, ilyenkor a neve: POP („Kilövés" a STACK-ból).
2. A kiolvasott PC_i + 1 birtokában egy JMP PC_i + 1 visszaugrás automatikus végrehajtása, miáltal a főprogram futása folytatódhat.

11.3.2.2. Címzési módozatok

Az eddigi utasításokkal kapcsolatosan az ADAT megtalálásának (megcímzésének) kétféle módozatával találkoztunk:

a) *Direkt (közvetlen) címzés:*

Ennél az ADAT címe (CÍM-effektív) közvetlenül megtalálható a D displacementben:

$$C_{eff} = D \tag{11.2}$$

b) *IMMEDIATE eset:*

Ilyennel az e) alpontbeli MVI B, DATA típusú utasításnál találkoztunk, ahol maga az adat (DATA) benne volt a D displacementben, így valójában nem is kellett címezni:

$$ADAT = D \tag{11.3}$$

Fentieken túlmenően az ADAT megcímzésének további módozatai is elképzelhetők. Ezek közül két további esetet tárgyalunk az alábbiakban.

c) *RELATÍV ÉS INDEXELT címzés*

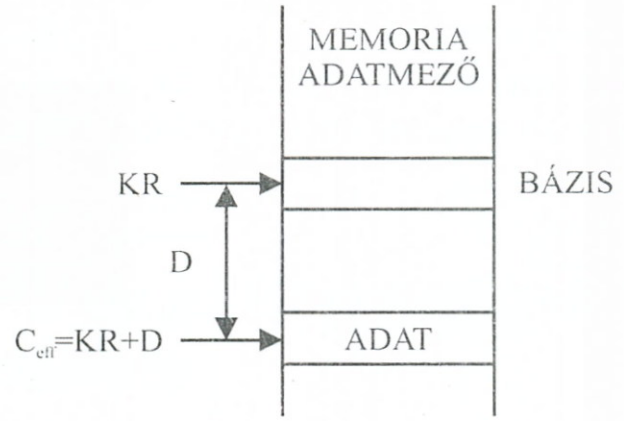
Ennél az esetnél a 11–1. példabeli berendezés KR kiegészítő regiszterét hívhatjuk segítségül és a *relatív címet* az alábbi formulával számítjuk ki:

$$C_{eff} = D + KR \tag{11.4}$$

A relatív címzés memóriabeli viszonyait szemlélteti a 11–18. ábra, melynél látható, hogy a KR-rel megadott „BÁZIS"-hoz viszonyítottan a D displacementtel tudunk egy KR + D címet meghatározni.

Ha a KR-regiszter inkrementálhatóságát, ill. dekrementálhatóságát is hasznosítjuk, akkor az ún. *indexelt címzés*hez jutunk, itt írható:

$$C_{eff} = D + KR \pm 1 \tag{11.5}$$



11-18. ábra *Relatív címzés egy esete*

Az *indexelt* címzés például előnyösen használható ADATTÁBLÁZATOK soronkénti megcímzésénél, ahol a következő sor megcímzéséhez csak egy inkrementálási műveletet kell végezni.

A *relatív* címzés elve például előnyösen használható a STACK MEMÓRIA olyan rekeszeinek megcímzésénél, melyek valahol a verem „mélyén" vannak. Ilyenkor a címzés relatív bázisa a STACK POINTER, és KR megadja, hogy a tetőhöz képest milyen mélyen van a címzett rekesz:

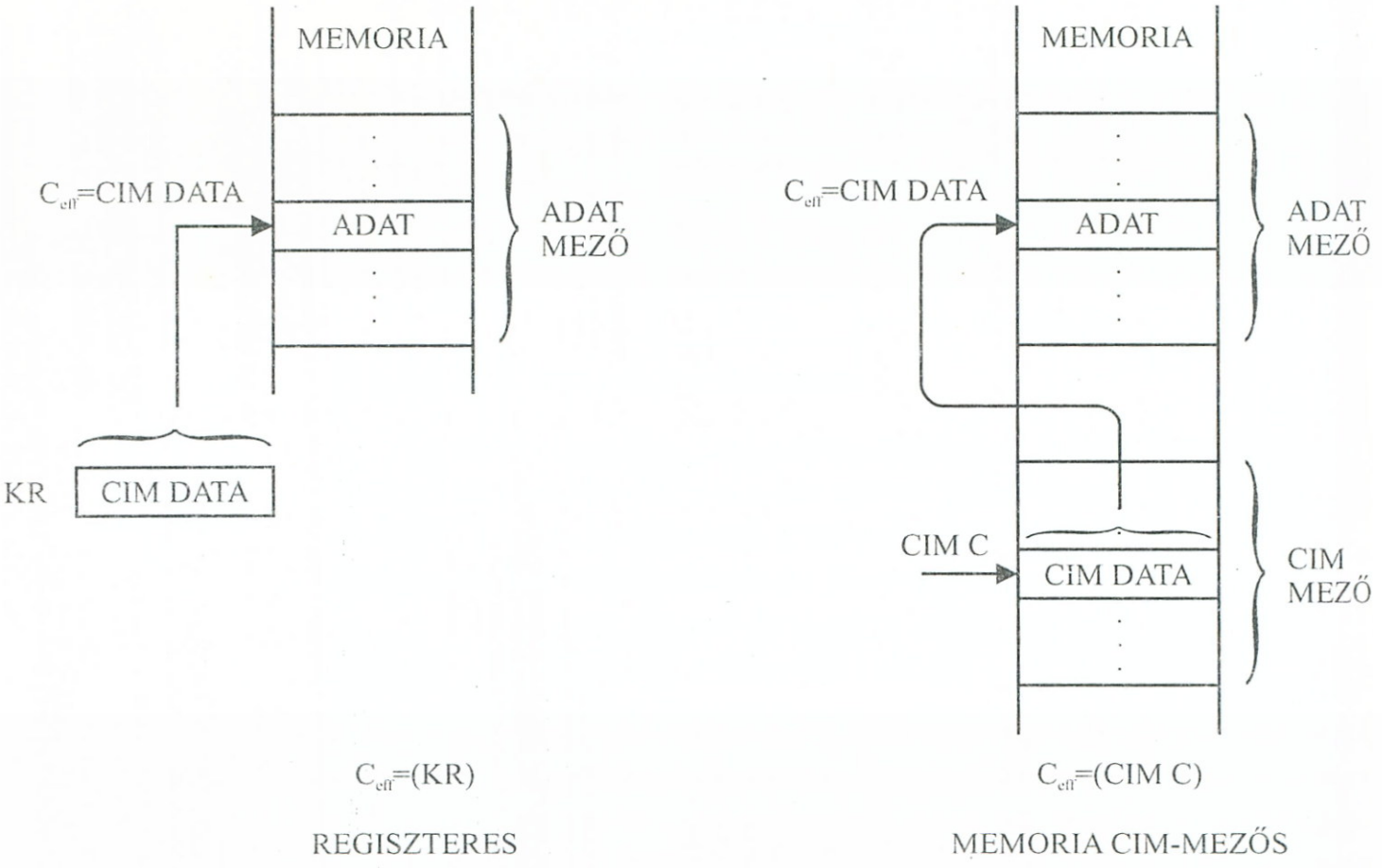
$$C_{eff} = SP + KR$$

d) *Indirekt (közvetett) címzés:*

A közvetett címzés azzal a történettel érzékeltethető, melyben valaki úgy adta meg a lakáscímét, hogy megnevezte azt az odvas fát, melynek odvában egy papírra a kérdéses lakáscím fel van írva. Az odvas fa esetünkben lehet például a KR kiegészítő-regiszter, de lehet a memória egy erre a célra elkülönített CÍM–MEZŐ-je, melynek CÍM C helyén levő rekeszében a CÍM DATA adatcím megtalálható. Indirekt címzésekre mutat példákat a 11–19. ábra.

Természetesen a használat előtt az „odvas fákba" (KR, CÍM-MEZŐ) előzetesen be kell írni a tényleges CÍM DATA értékeket. A közvetett címzéssel az utasításkészlet választéka bővíthető. Például a már bemutatott: LDA CÍM 1 (AC ← MEM) utasítás helyett választhatjuk a következőket (KR-t K-val jelölve):

```
MVI   K, CÍM DATA        (KR ← CÍM DATA)
  ⋮
MOV   A, M               (AC ← (KR)),
```



11-19. ábra *Indirekt címzési változatok*

ahol: az első utasítás ugyanolyan, mint a 11.3.2. pontban bemutatott IMMEDIATE e.) utasítás. A második utasítás részletezett végrehajtását pedig a 11–20. ábra mutatja:

CIKLUS	ELEMI HW MŰVELET	MAGYARÁZAT
1. (FETCH)	FETCH	– Megegyezik a korábbiakkal és itt a VEZ még azt is felismeri, hogy INDIREKT címzés van és CÍM DATA a KR-ben van tárolva
2.	BUF ← KR CR ← BUF DR ← (CR) AC ← DR	– CÍM DATA kiolvasása KR-ből és BUF-on keresztül áttöltése a CR-be – ADAT kiolvasása a MEM-ből – ADAT átvitele AC-be, és ezzel a művelet befejeződött

11–20. ábra *Az INDIREKT címzésű MOV A, M utasítás részletezése*

Mivel itt két utasítást használtunk az LDA helyett, látszólag ez a változat előnytelenebb. Viszont olyan esetekben, amikor a CÍM DATA helyről több ízben kell az adatot kiolvasni, az utóbbi változat lesz

gazdaságosabb, azáltal, hogy a MOV A, M csak egysoros utasítás. Ha még figyelembe vesszük azt is, hogy KR inkrementálható is, akkor egy nagyobb memória adatmező kiolvasásakor ez a gazdaságosság még növekszik.

11.4. A CPU és az IO-egységek kapcsolata

11.4.1. IO-egységek címzése

A CPU és az IO-egységek kapcsolata kétféle *címzési* elven is szervezhető.

- Az első változatnál az IO-egységek címzését teljesen azonos módon értelmezzük, mint a MEM-rekeszek címzését, ilyenkor az IO-egységek címei „alhalmaz"-át képezik a memória teljes (N) címkapacitásának. Az ilyen címzést MEMORY MAPPED (memóriában ágyazott) címzésnek nevezik.
- Elképzelhető egy olyan változat is, melynél az IO-egységek címmezeje függetlenített a memóriáétól. Ez esetben különösen fontossá válnak a 11–1. ábrán látható univerzális, digitális berendezés VEZ-egységének ME, IOE csatlakozásai, melyek révén ez a szétválasztás megvalósíthatóvá válik.

A CPU és IO-egységek adatforgalmában általában szerepel egy kitüntetett CPU-regiszter, mely az IO-egységek DR_i adatbuszaival tartja a kapcsolatot. Ez gyakran egy erre a feladatra elkülönített *IO-regiszter,* de – különösen kisebb gépeknél – gyakran ezt a feladatot is az univerzális AC akkumulátor-regiszter végzi el.

Tételezzük fel, hogy példaként bevezetett 11–1. ábrabeli univerzális, digitális berendezésünknél is ez a helyzet, és ekkor bevezethetünk további két (rendkívül egyszerű felépítésű) IO-utasítást:

IN CÍM IO_i: Ennek az utasításnak feladata az IO címtartományba eső: CÍM IO_i című IO_i egység DR_i adatregiszterében lévő információnak az AC akkumulátor-regiszterbe történő beolvasása:

$$AC \leftarrow DR_i$$

Az IO művelet okozta – korábbiakhoz viszonyított – eltéréseket a részletezett, 11–21. ábrabeli utasítás-végrehajtáson tanulmányozhatjuk. Az utasítás két szót foglal el a programmezőben.

CIKLUS	ELEMI HW MŰVELET	MAGYARÁZAT
1. (FETCH)	FETCH	– Ugyanaz, mint a 11–9. ábrabeli FETCH, csak itt a VEZ-egység a RD és IOE vezérlő kimeneteit aktivizálja a VEZÉRLŐ BUSZ-on. A RD (olvasás) az AC (befelé: IN) irányába kapcsolja az ADATBUSZT, IO pedig a CÍMDEKÓDERT aktivizálja.
2.	CR ← PC INR PC DR ← (CR) BUF ← DR CÍMDEK ← BUF AC ← DR_i	– CÍM IO_i-t tartalmazó rekesz címét átírjuk CR-be – PC-t előkészítjük a soron következő utasítás MK-jának kiolvasásához – Kiolvassuk CÍM IO_i-t a programmezőből, és mivel ez mutatja meg az IO egységek közül az aktualizálandót, ezért: – DR-t a BUF-regiszter segítségével áttöltjük a CÍMDEKÓDER-be, így előkészítettük a DR_i adatregiszter tartalmának kiolvasását – DR_i tartalmának áttöltése AC-be. Ezzel a AC ← DR_i művelet befejeződött

11–21. ábra *Az IN CÍM IO_i utasítás végrehajtásának részletezése*

j) *OUT CÍM IO_i.* Ez az utasítás az előzőkben bemutatottnak fordítottja. Feladata:

$$DR_i \leftarrow AC$$

művelet végrehajtása. A megoldási részletek a RD–WR és az adatátviteli irány felcserélésével teljesen hasonlóak a 11–21. ábrán bemutatottakhoz.

11.4.2. Az INTERRUPT. Program-megszakítás

Az univerzális, digitális berendezések folyamatos üzemelésekor előadódhatnak váratlan időpontban bekövetkező események (pl. egy STOP vagy ALARM gomb megnyomása), melyek miatt a program futását le kell állítanunk, a váratlan esemény ügyében intézkedni kell, majd ennek befejezését követően a programot újra-indítva, folytathatjuk a megszakított üzemelést. Az ilyen eseménysorozatot nevezzük MEGSZAKÍTÁS-nak, vagy INTERRUPT-nak, rövidebben: IT-nek.

A fent elmondottakkal kapcsolatosan némi hasonlóságot fedezhetünk fel a szubrutinnal foglalkozó 11.3.2.1. ponttal. Ott is megállítottuk

a főprogram futását, elugrottunk (egy részfeladat megoldása érdekében) a szubrutinra, majd ennek befejezésekor visszatérve a főprogramba, folytatjuk az előzőleg abbahagyott tevékenységet.

A szubrutinhívás és -megszakítás között mégis lényeges különbség van. A szubrutinhívást előre beterveztük, a kívánt helyen előre elhelyeztünk egy CALL utasítást, amely tartalmazta a szubrutin ugráscímét is. A megszakítás bekövetkezése váratlan történés, és ha az ezzel kapcsolatos eseményeket egy szubrutin keretében kívánnánk megoldani, akkor nem állna rendelkezésre a megszakítási rutin kezdőcíme sem.

Mint már sejthető, fentiek érdekében a 11–1. ábrabeli példaberendezésünk képességeit ki kell terjesztenünk, hogy az interrupt feladatok ellátására is képes legyen, továbbá az interrupt rendszerbe illeszkedő IO-egységeket is alkalmassá kell tennünk arra, hogy az új feltételeknek megfeleljenek.

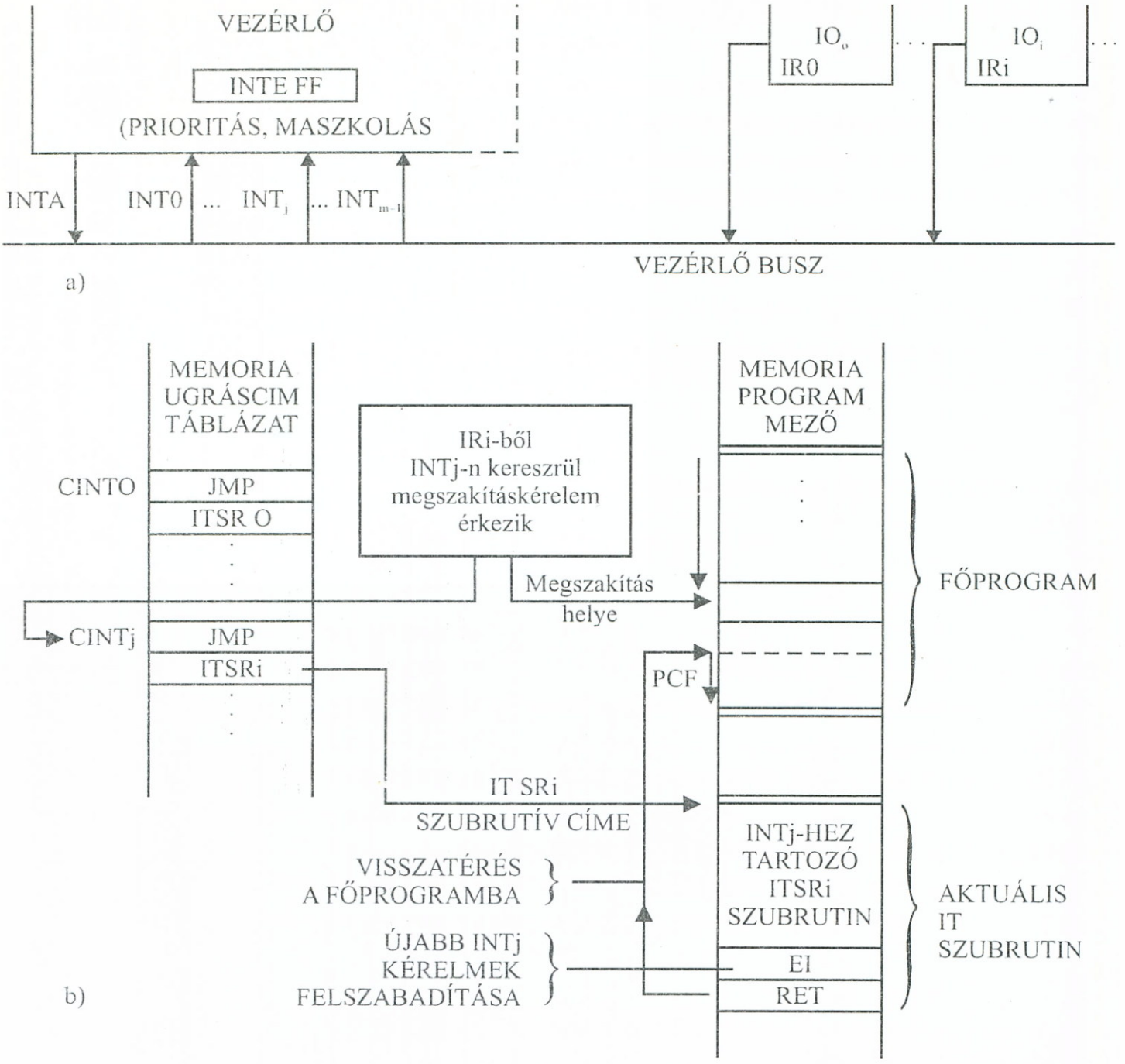
A 11–22. ábrán felrajzoltuk azokat a részleteket, melyekkel az eredeti 11–1. ábrát ki kell bővítenünk. Mint a 11–22a. ábrából látható, a VEZ vezérlőegység INT_0 ... INT_j ... INT_{m-1} (Interrupt Request) interrupt „kérelmező" bemenetekkel együtt, melyek a VEZÉRLŐ BUSZ-on keresztül az IO egységeken kialakított IR_0 ... IR_i ... IR_{n-1} interrupt kezdeményező csatlakozásokkal vannak kapcsolatban.

A megszakítási folyamat érdekében a MEM egységben elkülönítettünk egy UGRÁSCÍM TÁBLÁZAT-ot (lásd: 11-22b. ábra), melyben minden egyes INT_j vezérlő bemenethez hozzárendelünk $CINT_j$ jelű *címzéssel* néhány írható–olvasható táblázati memóriarekeszt. E rekeszek helye a memóriában fixen rögzített, és ha például az INT_j INTERRUPT KÉRŐ BEMENETRE „váratlanul" hardver jel érkezik, akkor a gép automatikusan:

$$PC \leftarrow CINT_j$$

típusú szoftver ugrást hajt végre, miközben (legtöbbször) elmenti a STACK-be a PC korábbi értékét – a későbbi visszatérés érdekében.

Az ugráscím táblázat üres rovataiba – mintha a programmező egy része lenne – általában egy ugró-utasítás keretében beírják annak a PROGRAMMEZŐ-beli helynek a címét (JMP $ITSR_i$), mely $ITSR_i$ címen a kérelmező IO_i egységéhez tartozó INTERRUPT SZUBRUTIN kezdődik. Az UGRÁSCÍM TÁBLÁZAT-ból tehát, azonnal továbbugorhatunk a programmező most már tetszés szerinti helyére megírható INTERRUPT SZUBRUTIN-ra. Mint az elmondottakból látható, a berendezés meglehetősen bonyolult módon tudja csak elő-



11-22. ábra *Az INTERRUPT folyamat szervezése univerzális digitális berendezésnél*

állítani az INT_j hardver-bemenet *áramkörbeli pozíciója* alapján a $ITSR_i$ hardver szubrutin szoftver címét.

A VEZ vezérlőegység INT_j interrupt-kérő bemeneteinek hardver környezetében egy INTE FF (INTERRUPT ENABLE FLIP-FLOP) helyezkedik el. Ez alapállásban „0" logikai állapotban van, és ilyenkor az INT_j bemenetek bármelyikére érkező interrupt kérelmet a VEZ egység elfogadja. Meg kell jegyezni, hogy az INT_j bemenetek között egymáshoz képest PRIORITÁSI SORREND is előfordulhat, így, ha *egyidejűleg* több kérelem érkezik, ezek közül a *magasabb prioritású* lesz a nyerő. Ugyancsak megjegyzendő, hogy az egyes INT_j bemene-

teket *előválasztással* le is lehet tiltani, mely műveletet MASZKOLÁS-nak nevezzük. Így például egy alkalmilag LEMASZKOLT (letiltott) magasabb prioritású bemenet már nem lesz „nyerő" a nem letiltott alacsonyabb prioritású bemenettel szemben.

Visszatérve az INTE flip-flopra, ez – mint említettük – INTE = 0 logikai állapotban van alapállásban, amikor IT folyamat nem zajlik. Egy INTj kérelem beérkezése és elfogadása esetén az INTE = 1 logikai állapot következik be, melynek révén az elfogadott INT_j bemenet kivételével a többi, hardver úton inhibitálódik. Ez a letiltás mindaddig fennáll, amíg az INTE flip-flopot vissza nem billentjük. Az INTE FF *szoftver úton* történő *billentésére* két utasítást szoktak bevezetni:

engedélyezés: EI (ENABLE INTERRUPT) (INTE ← ∅)
letiltás: DI (DISABLE INTERRUPT) (INTE ← 1).

A CPU-val együtt-dolgozó rendszer számára fontos információ, hogy a megszakítási kérelem elfogadásra került-e vagy sem. Emiatt a VEZ-egység az interrupt kérelem elfogadását követően egy INTA–INTERRUPT ACKNOWLEDGE (INTERRUPT NYUGTÁZVA) jelet küld vissza a digitális rendszerbe.

Az elmondottak alapján pontokba foglalhatjuk és a 11–22. ábrán követhetjük egy INPUT–OUTPUT-egység interrupt kérésének és végrehajtásának vegyes HW–SW folyamatát:

1. FŐPROGRAM fut, és egy „váratlan" pillanatban:
2. IR_i – HW interrupt kérés érkezik az IO_i egységtől
3. Ez a VEZÉRLŐ BUSZ-on keresztül az INTj interrupt kérési VEZ-bemenetre továbbítódik
4. A VEZ vezérlő egység *maszkolási* és *prioritási* szempontból megvizsgálja a kérelmet, továbbá megvizsgálja, hogy: INTE = 0 állapot fennáll-e. Ha igen, akkor elfogadja a kérelmet: INTE = 1 állapotot állít be, és INTA jelet küld a VEZÉRLŐ BUSZ-on keresztül a rendszer felé.
5. A FŐPROGRAM futása megszakad és a pillanatnyi PCF érték (a szubrutinhívásnál látottakhoz hasonlóan) a STACK-ba elmentődik.
6. Elugrás az UGRÁSCÍM TÁBLÁZAT-ba (PC ← $CINT_j$), majd innen tovább az IO_i-hez már korábban megírt INTERRUPT SZUBRUTIN $ITSR_i$ kezdőcímére.
7. Megkezdődik az INTERRUPT SZUBRUTIN végrehajtása.
8. Az IT SZUBRUTIN érdemi részének végrehajtása után, vissza kell állítani az INTE = 0 alapállapotot, ezért a rutinból történő

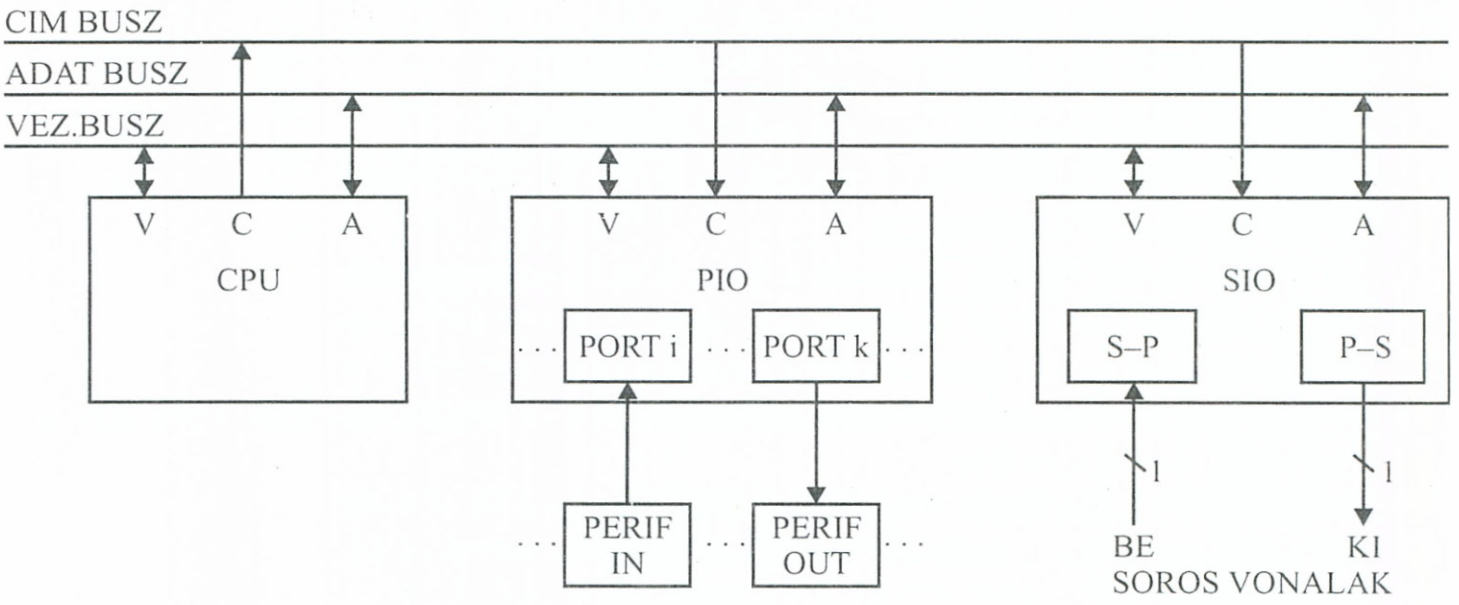
visszatérés előtt ki kell adni az EI SW törlő (INTE ← 0) utasítást.

9. RET utasítással az IT szubrutin befejeződik, és a STACK-ből kiolvassuk a főprogram megszakításkor elmentett PCF értékét, majd:
10. A főprogram folytatja megszakított futását.

11.4.3. IO INTERFÉSZ feladatok

Az IO-egységek többsége a „külső" feladatokat megoldó (PERIFERIÁLIS) egységek felé INTERFÉSZ feladatokat lát el, melyek során a perifériák működési paramétereit *illeszti* a CPU paramétereihez, továbbá az *adatátvitel* különféle változataihoz nyújt támogatást. Az adatátvitel *párhuzamos* vagy *soros* változatokban szervezhető és az INTERFÉSZ feladatok már bizonyos mértékben alkalmazkodnak az adatátvitel *szabványosított* módozataihoz és berendezéseihez (pl. MODEM-ek) is.

A párhuzamos IO feladatokat megoldó berendezéseket PIO egységeknek, míg a Soros IO feladatokat megoldókat SIO egységeknek fogjuk a továbbiakban nevezni. A CPU–IO rendszerben elfoglalt helyzetüket a 11–23. ábrán vázoltuk fel.



11-23. ábra *IO INTERFÉSZ-ek elhelyezkedése a rendszerben*

11.4.3.1. PIO egységek

Egy PIO egység blokkvázlatát a 11–24. ábrán tanulmányozhatjuk. Mint látható, egy PIO-ra gyakran több PERIFÉRIA is rákapcsolható, ilyenkor a PIO-ban perifériánként egy-egy ún. PORT áramköri egy-

séget építenek be. A PIO tartalmaz egy PIOVEZ vezérlőegységet, mely az egyes $PORT_i$-ket irányítja.

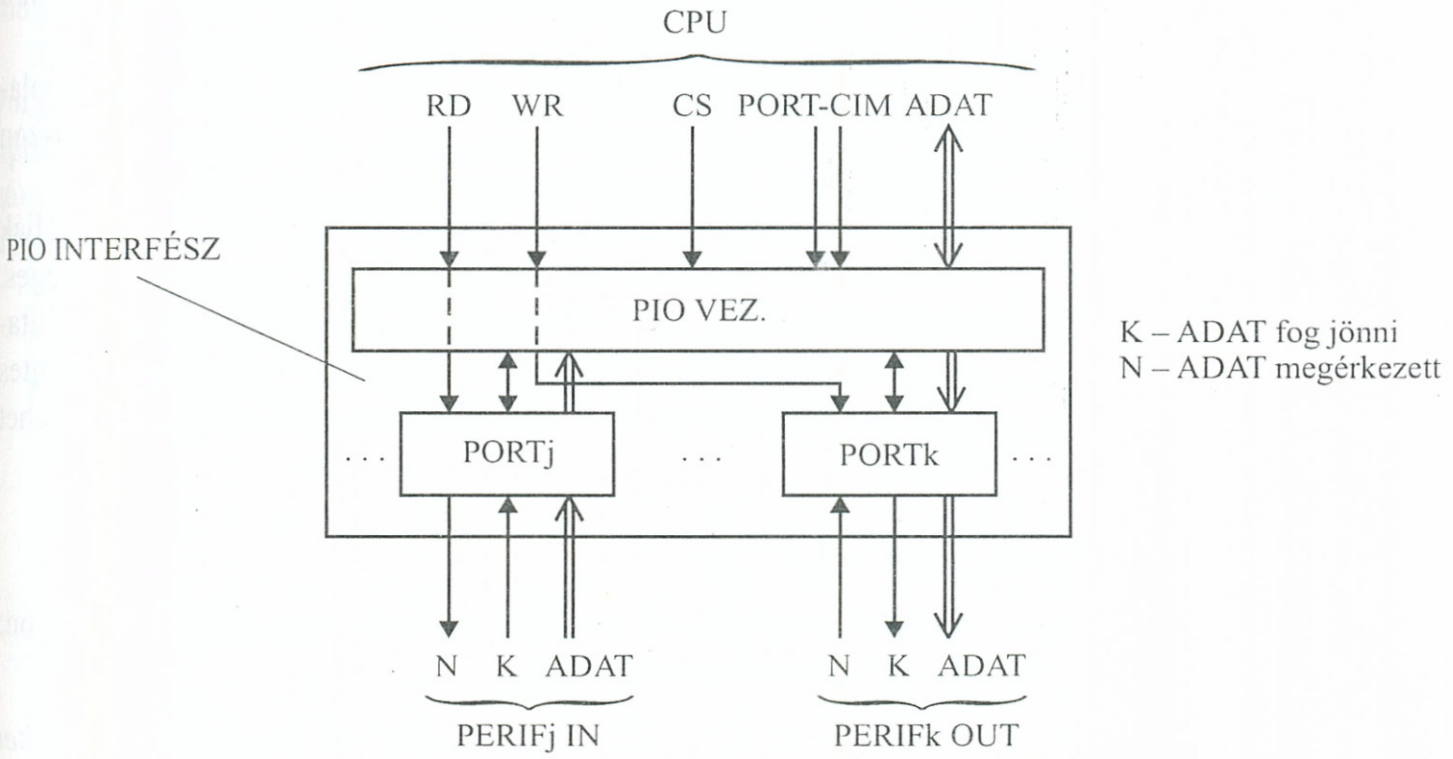
A PIOVEZ a CPU-val áll közvetlen kapcsolatban és onnan

- ADATBUSZ-ról: – ADAT-jeleket kap (OUT) vagy oda küld (IN)
- CÍMBUSZ-ról: – PIO kiválasztó jelet (CS–CHIP SELECT)
  – $PORT_i$ kiválasztó jelet ($PORT_i$ vagy PIO VEZ címe)
- VEZÉRLŐ BUSZ-ról: – WR (Kimenő ADAT)
  – RD (Bejövő ADAT)

jeleket kap.

Tartalmaz továbbá $PORT_i$ egységeket, melyek az egyes $PERIF_i$ perifériákat szolgálják ki. Az átvitel megbízhatóságának növelésére gyakran alkalmazzák az ún. HAND SHAKING átvitelt, mely a következő sorrend szerinti lépésekben dolgozik:

1. lépés: KEZDŐ jel (K) az *adatforrás felől:* „ADATOT fogok küldeni"
2. lépés: Párhuzamos ADAT-SZÓ (A) átvitele az ADATBUSZ-on
3. lépés: NYUGTÁZÓ jel (N) a *célállomás*tól: „Megjött az ADAT"



11-24. ábra *PIO INTERFÉSZ és csatlakozásai*

A 11–24. ábra PIO egységénél a „j"-edik $PERIF_j$ IN egy *adatforrás.* Ez küldi először a (K)-t, majd az INPUT (A)-t, végül a PORTj vissza $PERIF_j$-nek (N)-et.

Ugyanennél a PIO-nál a „k"-adik $PERIF_k$ OUT egy *adatvevő.* Ide először $PORT_k$ küldi a (K)-t, majd szintén ő küldi az OUTPUT (A)-t, végül a $PERIF_k$ *vevő*-állomás vissza az (N)-et.

11.4.3.2. SIO egységek

Egy SIO egység blokkvázlata a 11–25. ábrán látható. Általában egy SIO-ban egy SOROS ADÓ-t és egy SOROS VEVŐ-t építenek be. A SIO tartalmaz egy SIO VEZ vezérlőegységet, amely vezérli az ADÓ-t, a VEVŐ-t és közvetve (az ugyancsak SIO-ba beépített MODEM–VEZ egységet is. A MODEM (a MODULÁTOR–DEMODULÁTOR szavak összevonásából keletkezett) az aktuális aszinkron, vagy szinkron adatátvitelhez alkalmazkodva, a nemzetközi adatátviteli előírásoknak megfelelően módosítja (kódolja, rendezi) az átviteli paramétereket.

a) A *SIO VEZ* a CPU felől kapja a *vezérlő* jeleket.

A CÍM-BUSZ-ról:

- CS (Chip Select) jel a SIO-t aktualizálja
- C/D (Control/Data) jel az ADATBUSZ-on áramló információ *fajtáját* választja meg, és leggyakrabban a legalacsonyabb helyértékű címvezetékre (A_0) csatlakoztatják.
  Ha $A_0 = 0$, akkor az adatbuszon a *tényleges ADAT*-ok áramlanak be-, vagy ki irányban. Ha $A_0 = 1$, akkor az adatbuszon VEZÉRLÉSI információk áramlanak.
  Fontos észrevenni, hogy az ADATBUSZ-t itt felhasználják VEZÉRLÉSI információk szállítására is. Ez úgy lehetséges, hogy *páros*-CÍM($A_0 = 0$) esetén az adatbuszon mást szállítanak, mint *páratlan*-CÍM($A_0 = 1$) esetén. Ily módon a 8-bites adatbuszon (elvben) $2^8 = 256$-féle vezérlési információt lehet közölni.

A VEZÉRLŐ-BUSZ-ról:

- *WR aktív* esetén *kifelé* halad az információ az ADATBUSZ-on;
- *RD aktív* esetén *befelé* halad.

α) Amennyiben: $A_0 = 0$ KI-, ill. BE irányban *tényleges adatátvitel* történik.

β) Amennyiben: $A_0 = 1$

- *WR aktív* esetén különféle utasítás információkat küldhetünk a SIO-nak, ilyenek például:
  MÓD-utasítások:
  - szinkron/aszinkron üzemmód
  - adatátvitel sebessége
  - átviendő soros karakterek hossza bitben
  - legyen-e paritás ellenőrző bit stb.

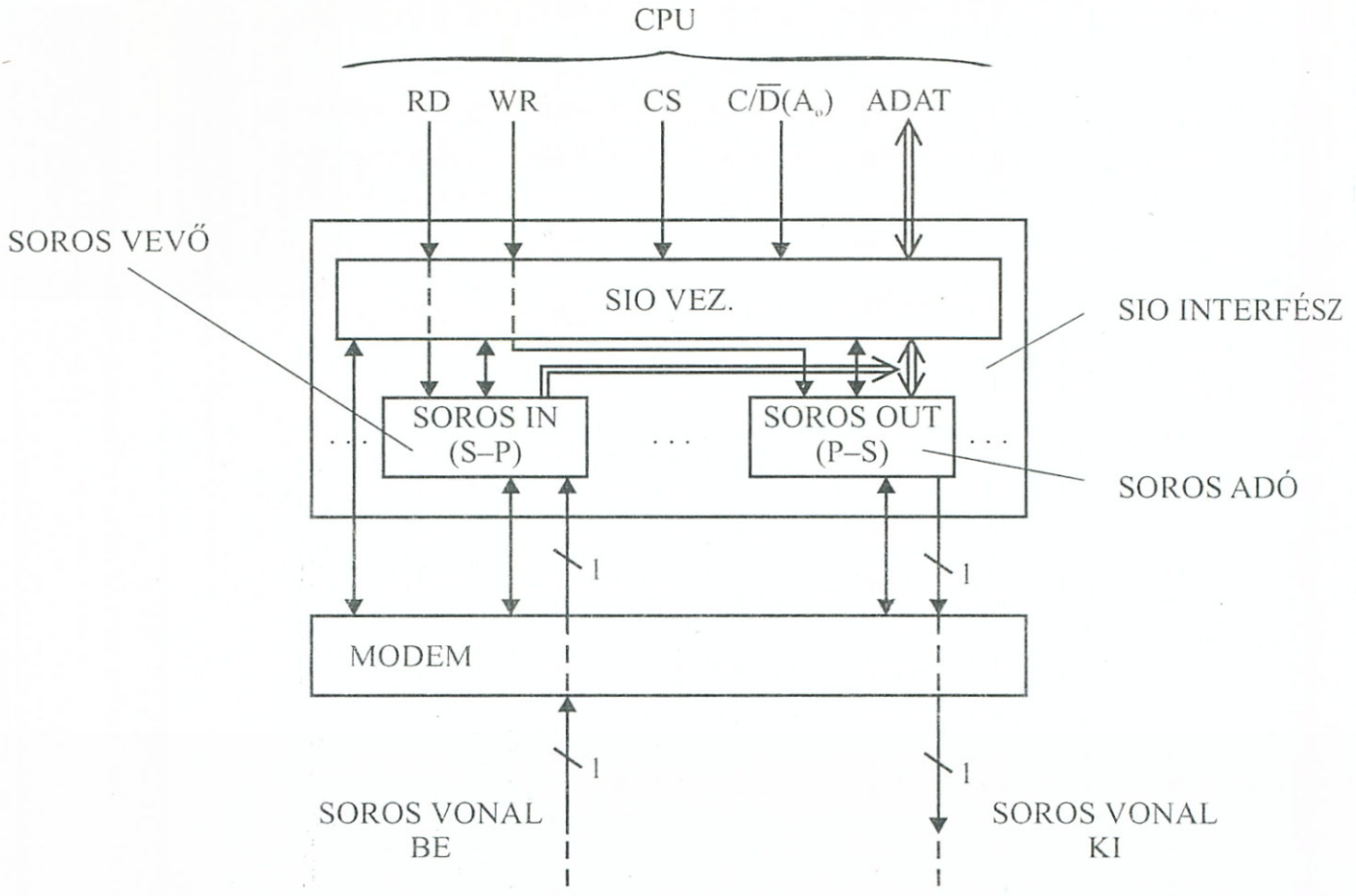
  Parancs-utasítások:
  - adás/vétel engedélyezése, letiltása
  - törlési utasítások stb.
- *RD aktív* esetén lekérdezhetjük a SIO-t, hogy milyen állapotban (státuszban) van. Ilyen *státusz* információk például a következők:
  - ADÓ-, ill. VEVŐ-egységek készenléti állapota
  - MODEM egység működési állapota
  - Paritás-, túlcsordulás-, adatformátum hibák stb.

b) A *SOROS OUT* (ADÓ) egység – lényegében PÁRHUZAMOS jelekből csinál időben sorbaállított soros kimenő jelsorozatot. Legfontosabb eleme a P–S kimeneti regiszter.

c) A *SOROS IN* (VEVŐ) egység SOROS-an beérkező jelekből, egy S–P bemeneti regiszterrel csinál egy PÁRHUZAMOS adat-szót, mely végül a CPU felé menő ADATBUSZ-ra kerül.

d) A *MODEM* egység maga nem része a SIO-nak, mivel a rendkívül sokféle adatátviteli variációhoz alkalmazkodva, nagyon sokféle típust gyártanak az erre szakosodott cégek. A SIO VEZ egység néhány uniformizált összeköttetéssel áll kapcsolatban a MODEM-mel, ilyenek pl.:

- MODEM állapotellenőrzés
- Adattovábbítás engedélyezése
- Adáskérés stb.



11-25. ábra *SIO INTERFÉSZ és csatlakozásai*

11.5. A MEMORIA és az IO-egységek kapcsolata

A MEMORIA és az IO-egységek közötti információ-átvitel két alapvető módszerét az alábbiakban foglalhatjuk össze.

11.5.1. Kapcsolat a CPU közreműködésével

Ennél az információ-átviteli változatnál az adatok mindig útbaejtik a CPU-t mindkét irányú átvitelnél:

$$\text{MEM} \leftrightarrow \text{CPU} \leftrightarrow \text{IO}$$

azaz a MEM ↔ IO közvetlen kapcsolat itt nem létezik. A 11–1. ábrán látható példabeli berendezésnél a CPU-beli AC Akkumulátor-regiszter látja el az átviteli közbenső állomás szerepét.
Például: a MEM ← IO irányú átvitel egy változata a következő program szerint zajlik:

IO MEM IN CIM IO_i – A CIM IO_i című IOi INPUT EGYSÉG DR_i adatregiszterének tartalmát átvisszük a CPU AC-be:

$$AC \leftarrow DR_i$$

STA CIM_k – Az AC tartalmának beírása a MEM CIM_k című rekeszébe:

$$(CIM_k) \leftarrow AC$$

Fordított irányban: IO ← MEM átvitel egy változata:

MEM IO LDA CIM_k – A CIM_k című MEM-rekeszt kiolvassuk a CPU AC-be:

$$AC \leftarrow (CIM_k)$$

OUT CIM IO_i – Az AC tartalmát átvisszük az IO_i OUTPUT EGYSÉG DR_i adatregiszterébe:

$$DR_i \leftarrow AC$$

Ez a tisztán szoftver utasításokon alapuló módszer – különösen nagyszámú elemet tartalmazó adatblokkok átvitelekor – „vontatottnak" bizonyulhat, ezért HW–SW kombináción alapuló gyorsabb átviteli módszereket alakítottak ki.

11.5.2. Közvetlen MEM–IO-kapcsolat

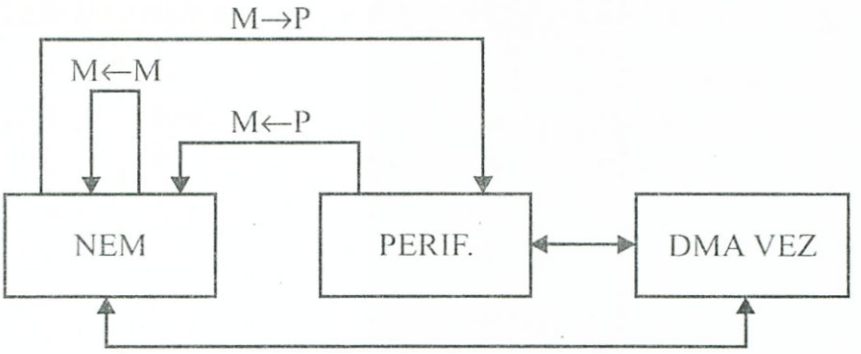
A közvetlen MEM–IO-kapcsolat (Direct Memory Acces [DMA]) kialakításánál első feladat a CPU semlegesítése a rendszerben. Emiatt a 11–1. ábrán szereplő, és a 11–22a. ábrán már módosított VEZ-egységünket tovább kell tökéletesítenünk. A „tökéletesítést" itt a HOLD (tartás), és HLDA (HOLD nyugtázása) nevű csatlakozások bevezetése jelenti, mely jelek a CPU rendszerbuszokról történő *leválasztásával* kapcsolatosak.

HOLD = 1 jel CPU-ba küldése esetén (némi belső késleltetéssel) a CPU ADAT, CIM és számos VEZÉRLŐ BUSZ csatlakozása *nagy-impedanciájú* (harmadik állapotba) kapcsolja át magát, majd erről a rendszert:

HLDA = 1 kimeneti jellel értesíti. Mindaddig, amíg a HOLD = 1 bemeneti jel fennáll, a CPU ilyen nagy-impedanciájú állapotban marad, és csak HOLD = 0 bekövetkezésekor kapcsolódik vissza a rendszerhez.

A CPU HOLD-dal történő kiiktatása csak első lépése a MEM–IO közvetlen kapcsolat létrehozásának.

A semlegessé vált CPU helyett célszerű bevezetni egy olyan *cél-hardver* egységet, mely a direkt MEM–IO-kapcsolat működteté-



11-26. ábra *Közvetlen adatátviteli változatok DMA-val*

séhez szükséges vezérlési teendőket magára vállalja. Ezt a cél-hardver egységet DMA VEZÉRLŐ EGYSÉG-nek (DMA CONTROLLER) nevezzük.

A CPU kiiktatása után fennmaradó:

MEM – PER – DMVEZ

rendszer a 11–26. elvi ábra szerinti közvetlen adatátviteli változatokat tudja megvalósítani.

A DMA folyamatban résztvevő egységes részletezettebb kapcsolatait és magát a DMA folyamat lebonyolódását a 11–27. ábrára támaszkodva tanulmányozhatjuk.

Egy korszerű DMA VEZ egységhez több PERIF egység is csatlakoztatható, és az egyes perifériák csatlakozódási pontjai között PRIORITÁSI sorrend állítható be. Legtöbbször e csatlakozódási pontok SW úton *lemaszkolhatók.*

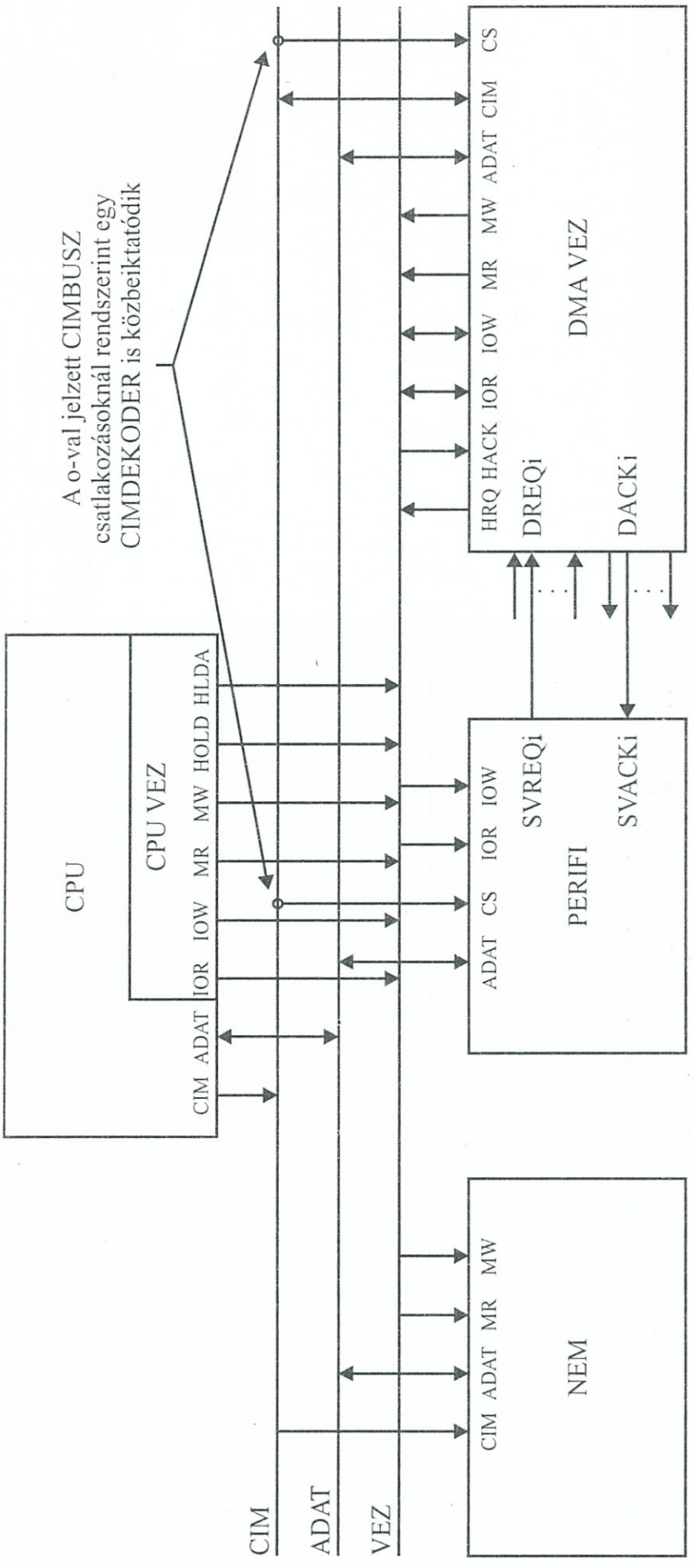
A DMA átvitel beindítása előtt a DMA VEZ egység *regisztereiben* be kell állítani a következő információkat:

- MEM-ba vagy MEM-ből áramló adatblokk kezdő címe
- átviendő karakterek száma
- DMA átvitel típusa (1-szavas, blokkos, periféria-vezérelt stb.)
- periféria bemenetek prioritási sorrendje.

Fenti *inicializálás* után a DMA *figyelő* állapotba kerül. Ezután – szemléltetésül – vizsgáljunk meg egyet részletesen a 11–26. ábra DMA változatai közül. A jelölések a 11–27. részletezett ábrán követhetők:

P → M átvitel.

- A DMA VEZ-re csatlakozó perifériális egységek közül $PERIF_i$ DMA átvitelt kezdeményez SVREQ (SERVICE REQUEST)–$DREQ_i$ (DMA REQUEST) jelkapcsolattal. Ezt a DMA VEZ



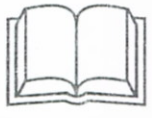
11-27. ábra *DMA VEZÉRLŐ-vel támogatott PER-MEM adatátviteli rendszer*

prioritási és *maszkolási* szempontból megvizsgálja, és ha minden rendben:

- HRQ (HOLD Request) – HOLD jelet küld a CPU-nak, felkérve azt, hogy menjen *nagy-impedanciájú* állapotba.
- CPU lekapcsolódik a rendszerről és HLDA (HOLD ACKNOWLEDGE) – HACK úton erről informálja a DMA VEZ egységet.
- DMA VEZ megüzeni PERIFi-nek, hogy kérését elfogadták: DACK – SVACK (SERVICE ACKNOWLEDGE).
- DMA VEZ átveszi a teljes rendszer irányítását és az előzetesen felprogramozott MEM-adatblokk *kezdőcímét* kiküldi a memóriának (tehát mostantól ő címezi a MEM-et).
- DMA VEZ ezután kiküldi az IOR (PERIFÉRIA OLVASÁSA) és a MW (MEM ÍRÁSA) vezérlőjeleket, majd az ADATBUSZ-on keresztül az *adatátvitel* megindul.
- Minden karakter átvitele után a DMA VEZ *dekrementálja* saját *karakter-számlálóját,* mikor ez eléri a nulla értéket, azaz a teljes blokk átvitelre került, az adatátvitel leáll.
- Ezután DMA VEZ megszünteti az eddig fennálló HRQ–HOLD-jelet, minek következtében CPU *visszatér* normál állapotba és megszünteti a HLDA–HACK jeleket, majd folytatja korábbi működését.
- DMA VEZ megszünteti DACK–SVACK-ot, így a $PERIF_i$ is megtudja, hogy a folyamat befejeződött.

Az $M \rightarrow P$ *átvitel*nél fentiekhez hasonlóan zajlik minden, csak az adatáramlás lesz ellentétes irányú. $M \rightarrow M$ *átvitel*nél SW úton indítjuk a folyamatot és az $M_1 \rightarrow M_2$ memória-helyek közötti átvitelnél nélkülözhetetlen *átmeneti tárolás* feladatát a DMA VEZ ún. AR (átmeneti regisztere) látja el: $M_1 \rightarrow AR \rightarrow M_2$ adatátviteli úttal. Természetesen itt foglalkozni kell az M_1 forrás és az M_2 célállomási helyek megcímzésével is.

11.6. Egy professzionális, univerzális, digitális berendezés szemléltető bemutatása



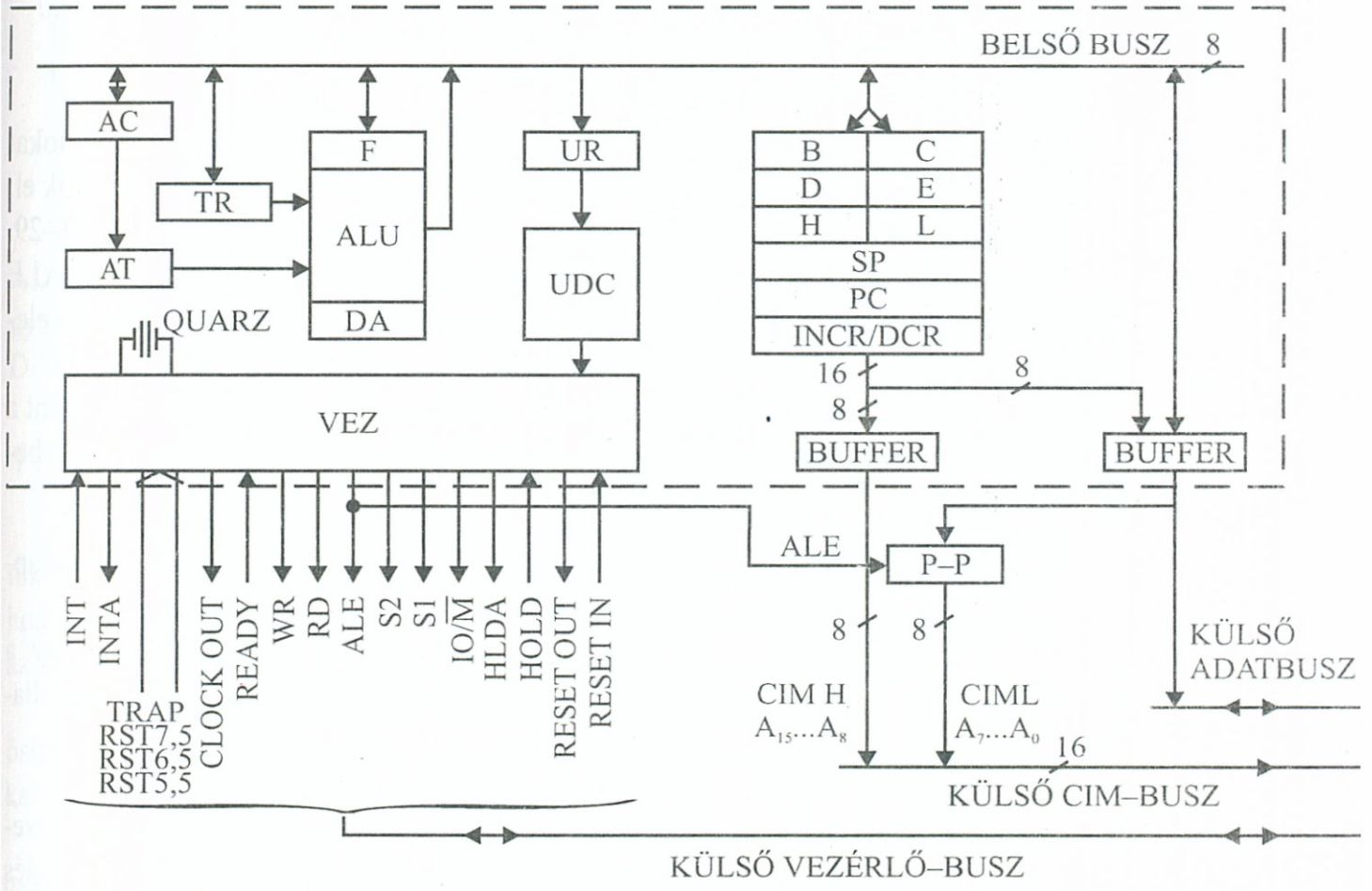
A fejezetben eddig elmondottak alkalmazásának szemléltetésére a következőkben alkalmazási példák keretében röviden bemutatjuk az INTEL cég egy *8 bit-bázisú* univerzális, digitális berendezését (mikroprocesszor), melynél a katalógus-változathoz képest néhány (didaktikai szempontból célszerű) kisebb egyszerűsítést végeztünk.

11.6.1. A HW felépítése

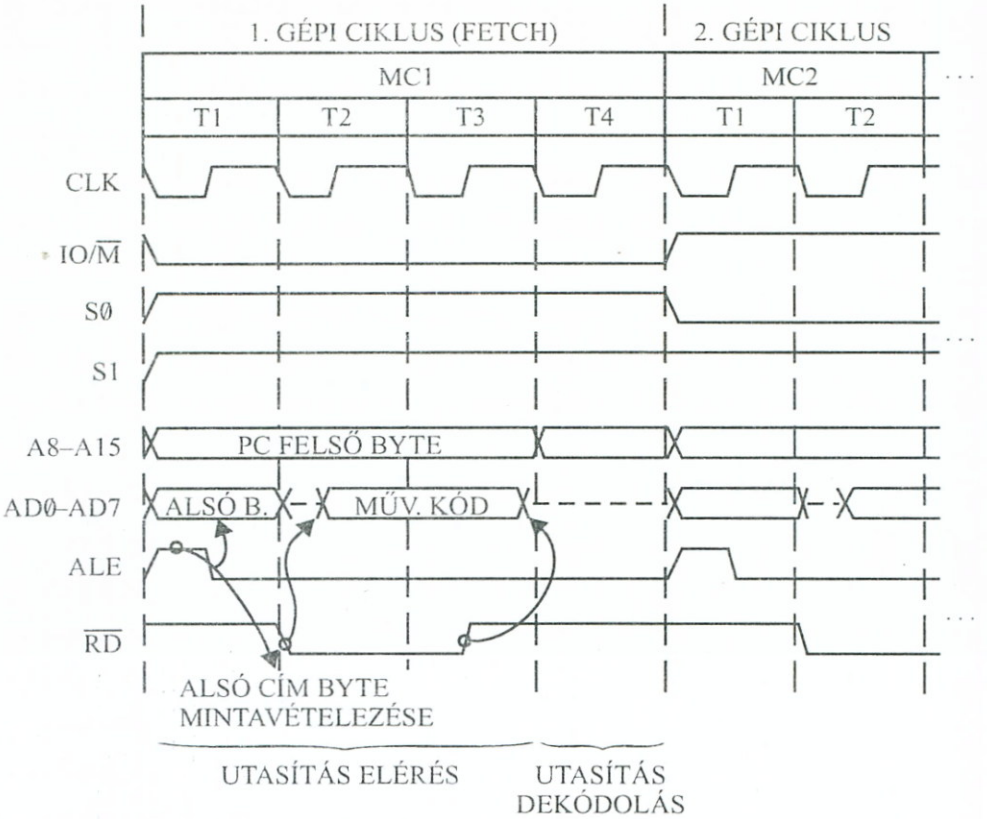
A HW-rendszer felépítése rokonságban áll a 11–1. ábrán bemutatott, elvi alapokat szemléltető rendszerrel. Lényeges különbség az, hogy az *adatbusz* és a MEMORIA szóhossza csupán 8 bit (1 Byte) hosszúságú, míg a *címbusz* 16 bites, így elvileg közvetlen címzéssel $2^{16} = 64$ kByte memóriahely érhető el. Ez – mint az utasításkészlet tanulmányozásánál majd tapasztalni fogjuk – kihat az utasításkészlet felépítésére is.

A *CPU felépítése* (11–28. ábra) is tükrözi a cím- és adatbuszhosszak eltérését. Célszerűségi szempontok alapján a *regisztertömb*-ben több munkaregisztert helyeztek el. Ezek egyedileg 8 bit hosszúságúak, de párokba is csoportosíthatók, melyek 16 bit hosszúságúak.

A CPU IC tok PIN lábainak korlátozott száma miatt a CÍM-busz 16 bitjéből az alsó nyolcat $CÍM_L$ (LOW): A_7, …, A_1, A_0 közös kimeneti lábra kötötték az ADAT-busz DATA: D_7, …, D_1, D_0 nyolc vezetékével. Emiatt ezeken a csatlakozásokon egy bizonyos időtartamig *CÍM-információk,* majd később *ADAT-információk* jelennek meg. A CÍM-információk időszakáról a VEZÉRLŐ-egység ALE (Address



11-28. ábra *INTEL-alapú CPU felépítése és csatlakozásai*



11-29. ábra *FETCH-CIKLUS idődiagramja a 11-28 ábra-beli CPU-nál*

Latch Enable) csatlakozása ad jelzést. Ilyenkor a CÍM-információkat (a 11–28. ábrán látható módon) egy külső P–P-regiszterbe mentik el, így a CÍML A_7 ... A_0 alsó byte később is rendelkezésre áll. A 11–29. ábrán felrajzoltuk a FETCH ciklus idődiagramját is, melyről az ALE időzítésviszonyok, továbbá a kiolvasás alatt álló utasítás MK műveleti kódjának időfázisa is kiolvasható.

A VEZ vezérlőegység itt több csatlakozással rendelkezik, mint a korábbi általános modellnél (11–1., 11–22. és 11–27. ábrák) már bevezetettek. Ezek a következők:

RESET IN:	Kívülről érkező HW-jel, mely a CPU-t alaphelyzetbe állítja.
RESET OUT:	A RESET IN-t követő kimenő jel, mely a CPU-hoz csatlakozó teljes rendszert alapállapotba hozza.
HOLD, HLDA:	A már ismert DMA jelek.
IO/M:	IO–input-output esetén = 1, M – memória esetén = 0 értékű jel, mely az IO vagy M kijelölés *előválasztására* szolgál.

S_0, S_1: STATUSZ jelek, melyek az IO/M-mel együtt kódolva jelzik a CPU pillanatnyi gépi ciklusát. (Például a FETCH kódja: IO/M–S_1–S_0 = 011)

ALE: A 11–29. ábra idődiagramján szereplő kapuzó jel, mely a CÍM–ADAT kimenetek szétválasztására szolgál (11–28. ábra).

RD, WR: OLVASÁS, ÍRÁS vezérlő-jelek. IO/M-mel kombinálva: IOR, IOW (INPUT/OUTPUT RD/WR), illetve MR, MW (MEM RD/WR), azaz *perifériára,* ill. *memóriára* aktualizálhatók.

READY: Kívülről érkező HW jel, mely jelzi, hogy a rendszer együttműködésre kész állapotban van.

CLK OUT: A rendszert ütemező órajel.

TRAP; RST 7,5; RST 6,5; RST 5,5: INTERRUPT-kérő bejövő jelek, melyek a 11–22. ábrán vázoltak szerint működnek, és prioritási sorrendjükben a TRAP a legmagasabb. Az egyes IT-kérő bemenetekhez rendelt ugrási táblázat-címeket és prioritást a 11–32. ábrán foglaltuk össze.

INTA: INTERRUPT kérést jóváhagyó, nyugtázó jel.

INT: INTERRUPT-kérő bejövő jel, melyre egy sokféle IT fogadására alkalmas speciális IT-vezérlőegység csatlakoztatható.

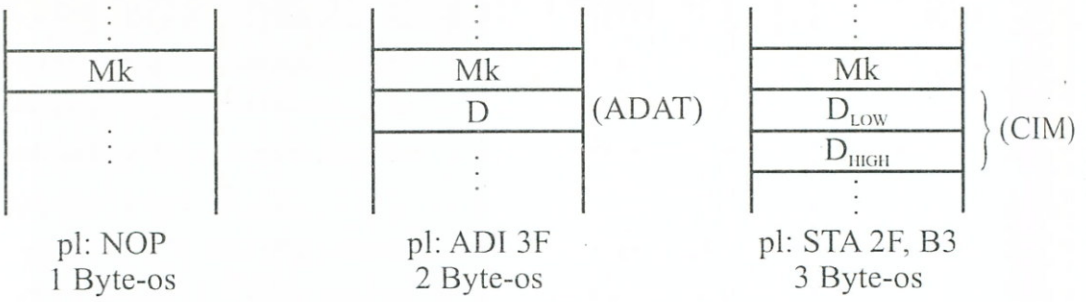
Végül megemlítjük, hogy a CPU el van látva egy DA (DECIMAL ADJUST) egységgel is, amely a (10. fejezet 10.6.3. pontjában tárgyalt) BCD 6-os korrekciót végzi el az utasításkészletben található DAA szoftver-parancs hatására.

11.6.2. Az utasításkészlet

A 8-bites adatok és a 16-bites cím információk – mint már említettük – befolyásolják az utasításkészletet is. A memória 8-bites szóhossza miatt háromféle (1, 2 és 3 Byte-os) utasítások szerepelhetnek az utasításkészletben a 11–30. ábra szerinti formátumokban.

A teljes utasításkészletet a 11–31a. ábra táblázataiban foglaltuk össze, a 11–31b. ábra pedig magyarázó megjegyzéseket és direktívákat tartalmaz.

Az utasítás-táblázat első oszlopa az *assembly* utasítást, a második oszlop az utasításnak hardver-leírónyelven leírt hatását mutatja, míg

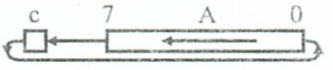
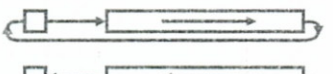


11-30. ábra *Utasításváltozatok a tanulmányozott berendezésnél*

a harmadik oszlop a FLAG regiszter azon flag-jeit sorolja fel, melyekre az adott utasítás hatást gyakorolt.

- Jellegzetesség, hogy ha valamelyik *utasítás-memória*-ban az X betű szerepel, akkor az utasítás *regiszter-párral* kapcsolatos. Ez alól kivételt a logikai XOR utasítások képeznek.
- Mint látható, a táblázatokból a *regiszter-párok* között a H,L kitüntetett szerepet is kap, hasonlóan a 11–1. ábrán szereplő KR-regiszterhez, amely az *indirekt címzésnél* az operandus címének tárolását végzi.
- A *STACK-műveleteknél* a SP Stack-pointer kétszer inkrementálódik/dekrementálódik a 16-bites CÍM információknak a 8-bites szélességű STACK-ban szükséges „kétemeletesre összegyűrt" helyfoglalása miatt.
- Az *összehasonlító* COMPARE utasítások *aritmetikai* úton (kivonással) kerülnek végrehajtásra, ezért „egyenlőség" esetén a *Z–Zero flag* aktivizálódik.
- A *PC-be* közvetlenül a H,L-en keresztül lehet valamit beírni.
- Az *aritmetikai* utasítások között átvitelt is figyelembevevő, ill. anélküli összeadások (kivonások) is szerepelnek.
- A *logikai* utasítások jellegzetessége, hogy az *átviteli flag*-okat „0"-ba állítják.
- A ROTATE (forgató) utasítások valójában SHIFT-elést végeznek az *accumulátor*nál, de a kilépő bitet mindig visszacsatolják a belépő oldalra: vagy közvetlenül, vagy a CY átvitel-biten keresztül.
- A CONTROL utasítások között kerültek felsorolásra az IO és az INTERRUPT flip-flopot billentő utasítások (EI, DI).
- Ugyancsak a CONTROL utasítások közt találhatók az ún. *RST d3 utasítás*ok, melyek a CALL utasításokkal és az IT-vel állnak rokonságban. Ugyanis első lépésben a PC-t a STACK-be mentik, viszont az elugrás itt már egy UGRÁSCÍM TÁBLÁZAT-ba történik, és innen a folyamat hasonlóan zajlik, mint a

SOURCE FORM		EFFECT	FLAGS AFFECTED
		Data transfer	
MOV	r_1,r_2	$r_1 \leftarrow r_2$	
MOV	r,M	r ← (HL)	
MOV	M,r	(HL) ← r	
MVI	r,d8	r ← d8	
MVI	M,d8	(HL) ← d8	
LXI	rp,d16	rp ← d16	
LXI	SP,d16	SP ← d16	
LDA	addr	A ← (addr)	
LHLD	addr	HL ← (addr)	
LDAX	B	A ← (BC)	
LDAX	D	A ← (DE)	
STA	addr	(addr) ← A	
SHLD	addr	(addr) ← HL	
STAX	B	(BC) ← A	
STAX	D	(DE) ← A	
XCHG		HL ↔ DE	
		Stack	
PUSH	rp	SP ← SP–2; (SP) ← rp	
PUSH	PSW	SP ← SP–2; (SP) ← A,F	
POP	rp	rp ← (SP); SP ← SP + 2	
POP	PSW	A,F ← (SP); SP ← SP+2	szapc
SPHL		SP ← HL	
XTHL		HL ↔ (SP)	
		Compare unsigned	
CMP	r	A–r	szapc
CMP	M	A–(HL)	szapc
CPI	d8	A–d8	szapc
		Jump	
JMP	addr	PC←addr	
JZ	addr	if z then PC←addr	
JNZ	addr	if z then PC←addr	
JC	addr	if c then PC←addr	
JNC	addr	if c then PC←addr	
JPE	addr	if p then PC←addr	
JPO	addr	if p then PC←addr	
JM	addr	if s then PC←addr	
JP	addr	if s then PC←addr	
PCHL		PC←HL	
		Call	
CALL	addr	SP ← SP–2; (SP) ← PC; PC ← addr	
CZ	addr	if z then CALL	
CNZ	addr	if z then CALL	
CC	addr	if c then CALL	
CNC	addr	if c then CALL	
CPE	addr	if p then CALL	
CPO	addr	if p then CALL	
CM	addr	if s then CALL	
CP	addr	if s then CALL	
		Return	
RET		PC ← (SP); SP ← SP+2	
RZ		if z then RET	
RNZ		if z then RET	
RC		if c then RET	
RNC		if c then RET	
RPE		if p then RET	
RPO		if p then RET	
RM		if s then RET	
RP		if s then RET	

SOURCE FORM		EFFECT	FLAGS AFFECTED
		Arithmetic	
ADD	r	A ← A + r	szapc
ADD	M	A ← A + (HL)	szapc
ADC	r	A ← A + r + c	szapc
ADC	M	A ← A + (HL) + c	szapc
ADI	d8	A ← A + d8	szapc
ACI	d8	A ← A + d8 + c	szapc
DAD	rp	HL ← HL + rp	c
DAD	SP	HL ← HL + SP	c
SUB	r	A ← A–r	szapc
SUB	M	A ← A–(HL)	szapc
SBB	r	A ← A–r–c	szapc
SBB	M	A ← A–(HL)–c	szapc
SUI	d8	A ← A–d8	szapc
SBI	d8	A ← A–d8–c	szapc
DAA		if $A_{3..0}$>9 or a then A ← A+6 if $A_{7..4}$>9 or c then A ← A+60H	szapc
		Logic	
ANA	r	A ← A AND r	sz0p0
ANA	M	A ← A AND (HL)	sz0p0
ANI	d8	A ← A AND d8	sz0p0
ORA	r	A ← A OR r	sz0p0
ORA	M	A ← A OR (HL)	sz0p0
ORI	d8	A ← A OR d8	sz0p0
XRA	r	A ← A XOR r	sz0p0
XRA	M	A ← A XOR (HL)	sz0p0
XRI	d8	A ← A XOR d8	sz0p0
CMA		A ← $\overline{A}$	
STC		c ← 1	1
CMC		c ← $\overline{c}$	c
		Increment	
INR	r	r ← r + 1	szap
INR	M	(HL) ← (HL) + 1	szap
INX	rp	rp ← rp + 1	
INX	SP	SP ← SP + 1	
		Decrement	
DCR	r	r ← r–1	szap
DCR	M	(HL) ← (HL)–1	szap
DCX	rp	rp ← rp–1	
DCX	SP	SP ← SP–1	
		Rotate	
RAL			c
RAR			c
RLC			c
RRC			c
		Control	
IN	d8	A ← portd8	
OUT	d8	portd8 ← A	
EI		INTE ← 1	
DI		INTE ← 0	
RST	d8	(SP–2) ← PC; SP ← SP–2; PC ← d3×8	
NOP		no operation	
HLT		halt until interrupt	

11–31a. ábra *Egy INTEL-alapú 8-bites mikroprocesszor utasításkészlete*

NOTATION

```
r       :Any 8 bit register: A B C D E H L
rp      :Any 16 bit register pair
          Write B for BC
                D for DE
                H for HL
PC      :Program Counter
SP      :Stack Pointer
M       :Memory byte referenced by HL
F       :Flag bites: s z 0 a 0 p 1 c
                     | | | | | | |
                     | | | | | |  carry bit
                     | | | | |  parity bit
                     | | | |  auxiliary carry bit
                     | |  zero bit
                     |  sign bit
INTE    :Interrupt Enable flag
PSW     :Program Status Word: A,F
addr    :memory address, 16 bits   } or expression
d3      :3 bits of data            } evaluted
d8      :8 bits of data            } to the given
d16     :16 bits of data           } length
(item)  :memory area referenced by „item"
port    :Input/Output port
$       :present value of location counter
[ ]     :optional firld, may be omitted
```

DIRECTIVES

```
        ORG     d16          :Location Counter ← d16
        END     [start addr] :end assembly
name    EQU     d16          :define name of value d16
name    SET     d16          :assign value d16 to name
[name]  DS      d16          :define storage of length d16
[name]  DB      d8[,d8]...   :define byte(s) with initial
                             value(s) given with
                             expression(s) or string(s)
[name]  DW      d16[,d16]... :define word(s) with initial
                             value(s) given with
                             expression(s)
```

11-31b. ábra. *Kiegészítések az a) ábrabeli utasításkészlethez*

11–22b. ábra esetében. Az ugráscím-táblázat kezdetét ennél a processzornál a MEMORIA 00,00 címére helyezték. Valamely RST_i utasításhoz rendelt ugráscím-táblázathely kezdőcíme az alábbi formulával számítható ki:

$$\text{CÍM UT}_{\text{KEZDŐ}} = d3 \times 8 \qquad (11.6)$$

Például az RST 5 utasítás először elmenti a pillanatnyi PC-értéket, majd elugrik a $5 \times 8 = 40 \rightarrow 00{,}40$ címre.

A részletezett ugrási táblázatot a 11–32. ábrán rajzoltuk fel. Ebbe belerajzoltuk az *SW:RST d3 utasítások* mellett, a CPU INTERRUPT csatlakozási pontjai közül az (RST_i-khez rokon) ún. *HW:RST bemenetek*hez rendelt fix ugrási táblázatcímeket is (melyekre már a 11–22. ábrán utaltunk).

- Megemlítendő még, hogy a program futása az ún. HLT (HALT) utasítással leállítható, de a folytatás ilyenkor csak HW *interrupt* úton valósítható meg.
- A NOTATION táblázatban a FLAG-bitek közt található „*a*"-AUXILIARY CARRY-bit az egy Byte-ban *tetrádosan* ábrázolt két BCD szám közötti *átvitelt* jelenti.
- A DIRECTIVES táblázatban szereplő

ORG d16

RST_i	MEM CÍM	IT PRIORITÁS
RST 0	00,00	
RST 1	00,08	
RST 2	00,16	
RST 3	00,24	
RST 4	00,32	
TRAP	00,36	1 (legmagasabb)
RST 5	00,40	
RST 5,5	00,44	4
RST 6	00,48	
RST 6,5	00,52	3
RST 7	00,56	
RST 7,5	00,60	2

11–32. ábra *RST_i ugrási táblázat a 11–28. ábrabeli mikroprocesszorhoz*

utasítás az utána következő *assembly* program programmezőbeli kezdőcímét jelöli ki.

- Az END [start addr] utasítás az ORG-gal kezdett program végét határozza meg.

11.6.3. Szemléltető feladatok

11.1. példa: Vizsgáljuk meg az AC akkumulátor tartalmát az alábbi programrészlet lefutása végén:

```
        ⋮
MVI     B, Ø5 H
MOV     A, B
MVI     C, 12 H
INR     A
ANA     C
HLT
```

A 11–31a. ábra utasításkészlete alapján a következők írhatók fel.

assembly	hatás	bináris tartalom	
MVI B, ∅ 5 H	B ← ∅ 5 H	B: 00000101_2	
MOV A, B	A ← B	A: 00000101_2	
MVI C, 12 H	C ← 12 H	C: 00010010_2	} és
INR A	A ← A + 1	A: 00000110_2	}
ANA C	A ← A és C	A: 00000010_2	

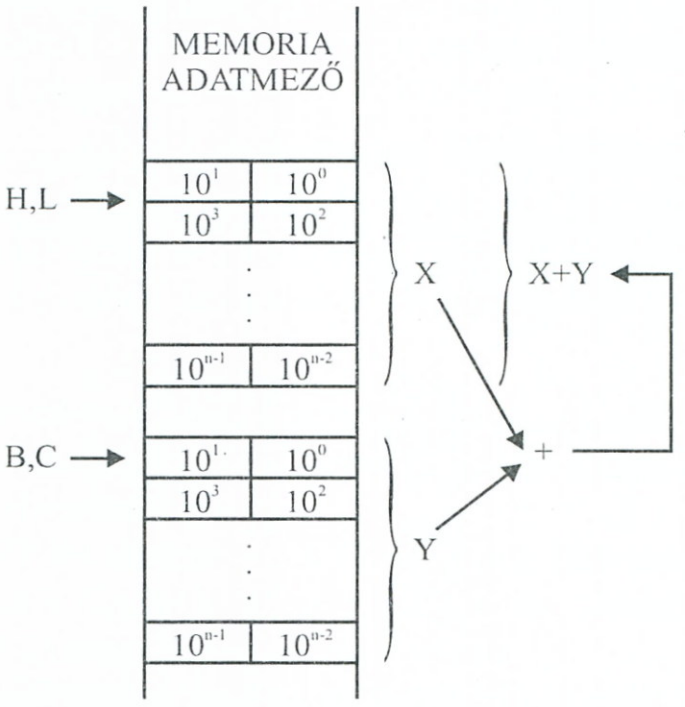
Az AC tartalma: AC = ∅2H

11.2. példa: Legyenek adottak az X, Y decimális számok, melyek n számjegyből állnak, és a memóriában a 11–33. ábra szerint BCD-ben vannak ábrázolva.

Írjunk assembly szubrutint, mely elvégzi az X + Y összeadást, és az eredményt az X helyén (azt felülírva) tárolja.

Mint látható, az X, ill. Y számjegyei helyérték szerint alulról kezdődően rendszerezve találhatók H, L, ill. B, C regiszterpárokkal címezve az adatmezőben. A megírandó programhoz be kell vezetnünk egy POINTERT, mely az egyes Byte-onként elvégzett rész-összeadásokat majd számolni fogja. Válasszuk e számláló pointernek a még le nem foglalt D-regisztert, mely maximum n = 256 Byte-ig (2^8) képes számolni. Az összeadható számjegyek száma (N) ennek kétszerese lesz

$$N = 2n,$$



11-33. ábra *BCD összeadás a 11.2 Példa szerint*

mivel egy Byte-ban két számjegy van tárolva.

Szemléltessük a program felépítésének lépéseit egy folyamatábrán, mely mellett egyidejűleg párhuzamosan feltüntetjük az assembly utasításokat, továbbá kommentáltuk az egyes lépéseket is (11–34. ábra).

A 11–34. ábrához még célszerű néhány kiegészítő megjegyzést tenni.

PROGRAM-MEZŐ CIMEK	ASSEMBLY PROGRAM		FOLYAMATÁBRA	MAGYARÁZATOK
	ORG 2FOOH			– Program elhelyezése a programmezőben
			BCDÖ	– BCDÖ: Rutin elnevezése
2F 00 H	BCDÖ	MVI D,n	D ← n	– D-beállítása n Byte-ra
2F 01		ANA A	CY ← ∅	– CY-nullázása
			CIKL	
2F 02	CIKL	LDAX B	AC ← (BC)	– Y első Byte lehozása AC-be
2F 03		ADC M	AC ←AC+(H,L)+CY	– Y+X+CY összeadás
2F 04		DAA	DECIMAL ADJUST	– BCD 6-os korrekció
2F 05		MOV M,A	(H,L)←AC	– Összeg aktuális Byte-jának tárolása
2F 06		INX H	H,L INKREMENTÁLÁS	– Következő Byte-ok megcímzése
2F 07		INX B	B,C INKREMENTÁLÁS	
2F 08		DCR D	D DEKREMENTÁLÁS	– POINTER tartalmának csökkenése
2F 09		JNZ CIKL	D = ∅	– Utolsó Byte volt?
			NEM / IGEN	
2FOC		RET	RET	
		END 2F00		– Itt a program vége a memoriában

11-34. ábra *BCD összeadás assemblyprogramja és folyamatábrája*

- A BCDÖ a szubrutin hivatkozási neve az assembly programban *kezdőcímke*ként szerepel. A címke egyben konkrét programbeli címet is képviselhet, melynek értéke esetünkben: BCDÖ = 2F∅∅H.
- A CY nullázása (mivel közvetlen ilyen utasítás nincs) lehetett volna: STC CY ← 1

  CMC CY ← CY

  módon is, de ez két utasítást igényelt volna, ezért hasznos volt kihasználni a logikai utasítások CY-t nullázó tulajdonságát.
- A CIKL ugráshelyi címke itt a 2F∅2 H címet képviseli.
- A „carry-s" összeadásnál az első bytok összeadási ciklusában a CY még nulla a kezdeti nullázás miatt. Később az előző összeadás átvitelét adja hozzá az aktuális Xi+Yi-hez.
- A JN7 CIKL valójában egy 3 Byte-os utasítás:

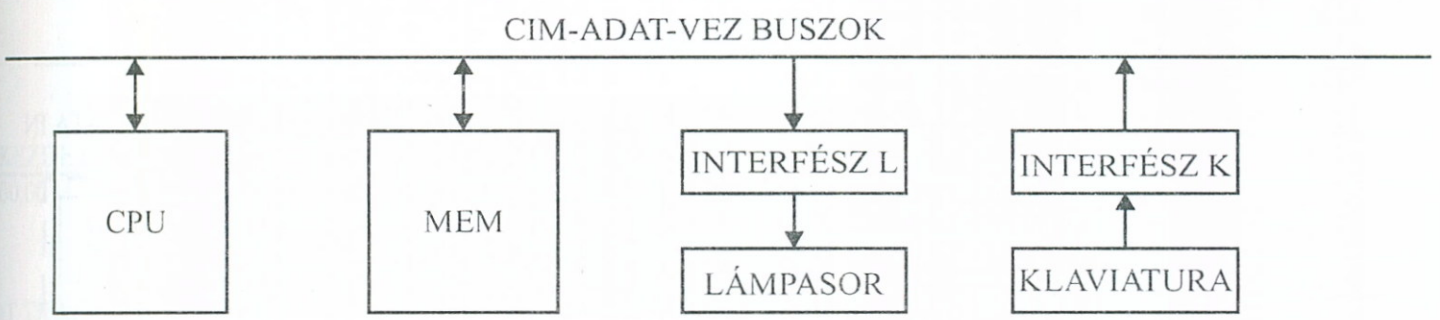
$$\begin{array}{ll} \text{MK}: & \text{JNZ} \\ \left.\begin{array}{l}\text{CIM}_\text{L}\\ \text{CIM}_\text{H}\end{array}\right\} & \text{CIKL} \end{array}$$

  miatt, ezért az utána következő utasításcím 3-mal több. Előzőleg minden utasítás 1 Byte-os volt, ezért ott a címek csak 1-gyel növekedtek.

11.3. példa: Legyen adva egy maximum 255 db billentyűt tartalmazó klaviatúra és egy 8 lámpát tartalmazó kijelző lámpasor. Valamely billentyűt megnyomva egy (a billentyűre jellemző) 8-elemes bináris kód jelenik meg a billentyű csatlakozási pontjain. Alakítsunk ki egy olyan berendezést, melynél egy billentyű megnyomása esetén a lámpasoron az a bináris kód jelenik meg, amely a megnyomott billentyűre jellemző, továbbá a billentyű kódja feljegyzésre kerüljön a memória egy inkrementálódva címzett adat-táblázatában.

A feladatot korábbi „klasszikus digit" ismereteinkkel is meg lehetne oldani. Ennek ellenére, most oldjuk meg egy SW–HW kombinációs elem működő, univerzális, digitális berendezés felhasználásával. Válasszuk a 11–28. ábrán látható mikroprocesszort.

A feladat jellegéből látható, hogy itt tisztán párhuzamos adatátvitelt célszerű alkalmazni. A billentyűzet és a lámpasor közvetlenül nem, hanem csak *INTERFÉSZ egységen* keresztül illeszthető a mikroprocesszoros rendszerhez. Erre felhasználhatnánk a 11–24. ábrán megismert, kereskedelmi forgalomban beszerezhető PIO egységet is,



11-35. ábra *CPU-MEM-LÁMPA-KLAVIATURA rendszer*

de részben a feladat egyszerűsége, részben didaktikai szempontból most az INTERFÉSZ-eket is magunk fogjuk kialakítani.

A feladat első közelítésű blokkvázlata a 11–35. ábrán látható:

a) A berendezés hardverje

A működés algoritmusának első lépése a klaviatúra valamelyik K_i gombjának megnyomása. Ez a gomb egy 8-bites (rá jellemző) $KÓD_i$ karaktert fog az INTERFÉSZ K-n keresztül az *adatbusz*ra küldeni, melyről előzetesen a CPU-t informálni kell. Ennek érdekében (miután a gombnyomás váratlan jelenség) célszerű *interrupt*-os üzemmódot választani, tehát feltételezzük, hogy bármelyik nyomógomb megnyomásakor egy IRQ (INTERRUPT REQUEST) IT-kérő jel jelenik meg a klaviatúra erre a célra kialakított HW-kimenetén. Ezt csatlakoztatnunk kell a CPU valamelyik IT-kérő bemenetéhez. Válasszuk példánkban a TRAP bemenetet erre a célra.

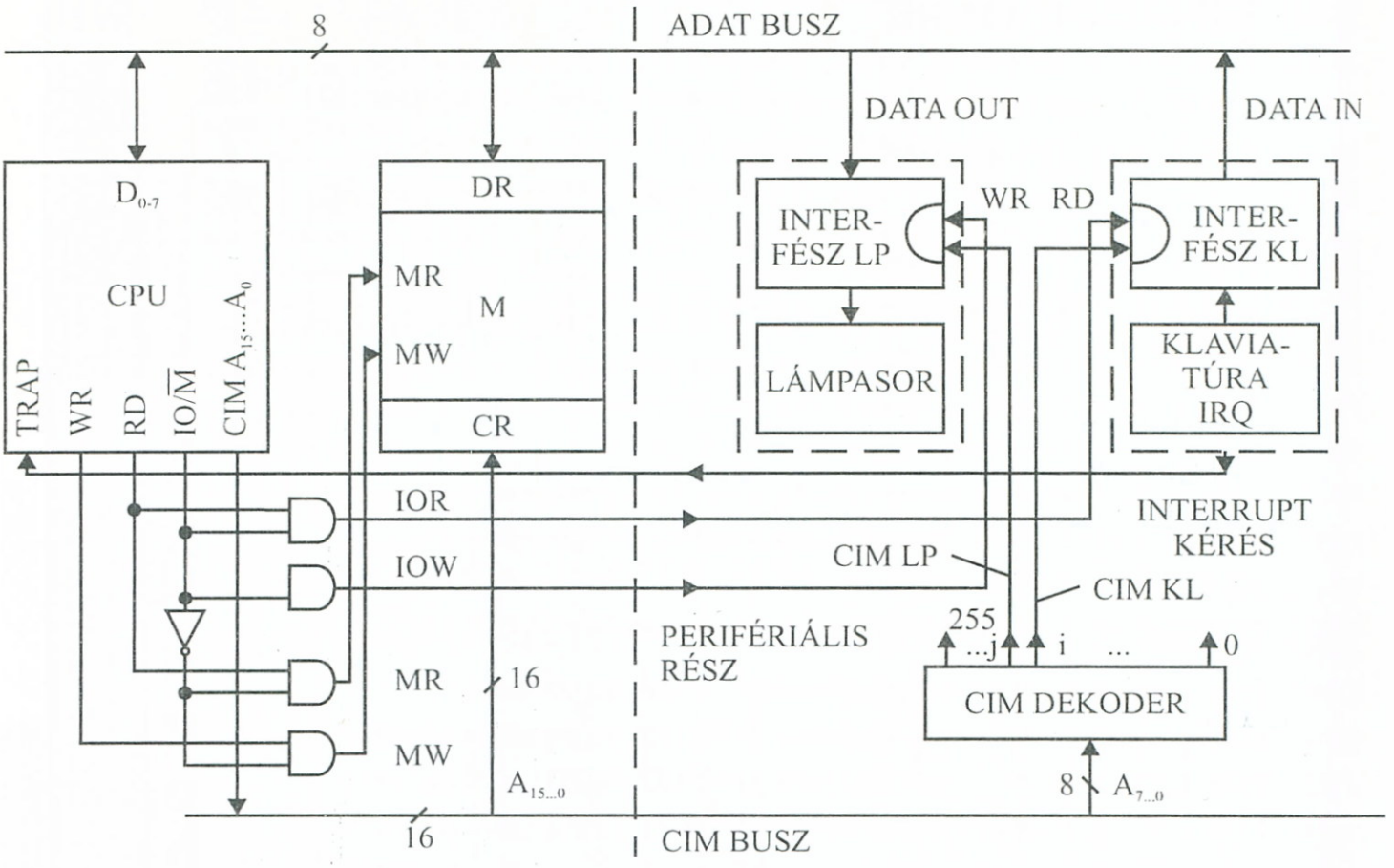
A klaviatúra INTERFÉSZ K-ja kialakítható egy kapuzott P–P regiszterből, melybe a gomb kódja megnyomásakor *közvetlenül* átíródik, és az *adatbusz* felé eső háromállapotú kimenete akkor fog aktivizálódni, amikor az IT-kezelő szubrutin majd parancsot ad.

A lámpasor INTERFÉSZ L-jével hasonló a helyzet, az *adatbusz* felől érkező adatbeírásra engedélyt az IT szubrutin fog adni, és az INTERFÉSZ L P–P-regiszter flip-flopjai „közvetlenül" gyújtják ki a kijelző lámpákat.

Mindkét interfész megcímzése egy CÍMDEKÓDER-en keresztül történik.

A címzéseknél különbséget kell tennünk a MEM és IO-címzések között, ezek szétválogatását a CPU IO/$\overline{M}$, RD, WR csatlakozásaival vezérelt kapukkal oldottuk meg a 11–36. ábrán. A *memória*-címek 16-bitesek, az IO-címek ennél a mikroprocesszornál (lásd: utasításkészlet) 8-bitesek.

A rendszer-hardver részletezett vázlatát a 11–36. ábrán rajzoltuk fel.



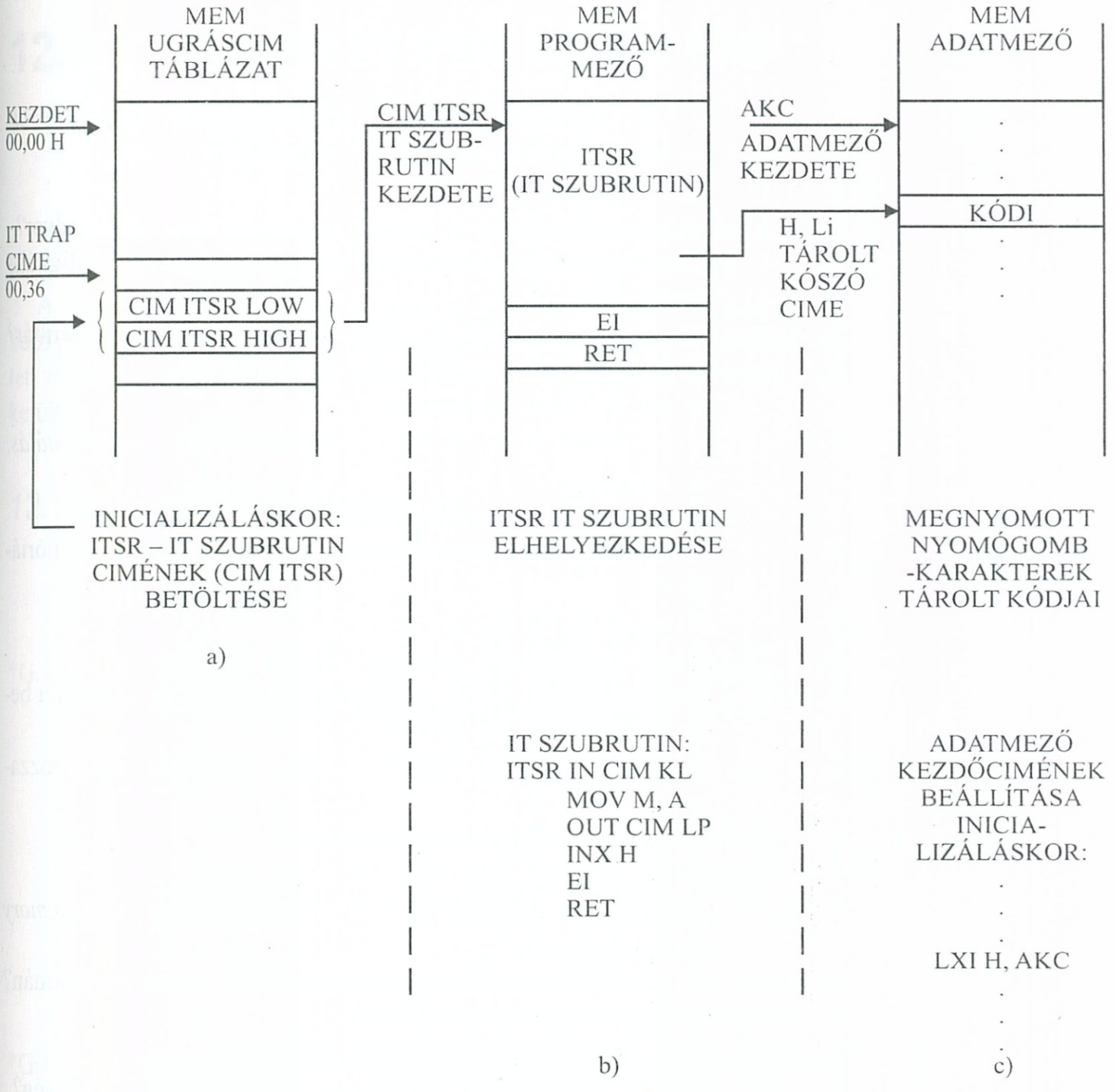
11-36. ábra *KLAVIATÚRA–LÁMPASOR–MEM–CPU rendszer részletezett hardverje*

b) A berendezés szoftverje

Miután a szoftver interruptot kezel, szubrutint működtet és IO, ill. MEM-kezelési feladatokat is ellát, ezért a jobb áttekintés érdekében az információ-viszonyokat a 11–37. ábrán részletesen szemléltettük.

Az IRQ IT-kérést a HW összeállításakor a TRAP IT-bemenetre vezettük, ezért az IT-jel ideérkezésekor a CPU elugrik a (11–22b. és 11–2. ábráknál elmondottak értelmében) memória MEM UGRÁS-CÍM-TÁBLÁZAT-ának TRAP-hoz tartozó címére, mely a 11–32. ábrából kiolvashatóan: 00, 36H értékű. Ide – az üzemelés megkezdése előtt (INICIALIZÁLÁSKOR) – be kellett töltenünk azt a CÍM ITSR *IT-szubrutincímet,* melytől kezdve a MEM–PROGRAMMEZŐ-ben az ITSR interrupt szubrutin elhelyezkedik (11–37a. ábra).

A szubrutinban szerepelni fog a kimeneti INTERFÉSZ L periféria címén (CÍM LP) kívül a MEM H, Li címe is, melyhez tartozó MEM-rekeszbe a megnyomott gomb kódját (KÓDi) – a feladat értelmében – majd betároljuk. A MEM címzését célszerű indirekt módon a H, L utasítással végezni, mivel a táblázatos címzésnél ilyenkor elegendő H, L inkrementálást alkalmazni az ismételten beérkező $KÓD_i$ karakterek esetén.



11-37. ábra *A 11-36. ábrabeli HW-hez tartozó SW tevékenységek és programok*

A 11–37c. ábra alsó felén látható, hogy inicializáláskor miként lehet beállítani az adatmező kezdő címét (H, L ← AKC).

A 11–37b. ábra alsó felén található az assemblyben megírt ITSR *IT szubrutin.* Mint említettük, ennek a végén szerepelnie kell az EI (INTE ← 1) utasításnak, hogy a CPU IT letiltása megszűnjön és a RET után fogadni tudja az újabb IT-hívásokat.

11.E. Ellenőrző kérdések

?

E.11.1. Milyen elvi vázlaton alapulnak a szoftver- és hardver *együttes alkalmazásán* alapuló digitális rendszerek?
E.11.2. Mi a NEUMANN-elv?
E.11.3. Milyen szervezési elvet igényel egy *univerzális* hardver?
E.11.4. Milyen *összetevőkből* épül fel egy univerzális, digitális rendszer?
E.11.5. Mit tud a *vezérlőegységről*, mit tárol a mikroprogram-tár?
E.11.6. Milyen műveleteket old meg egy ALU?
E.11.7. Mire jó a FLAG-regiszter?
E.11.8. Mit eredményez a programszámlálónál az *inkrementálás*, ill. *preszetelés?*
E.11.9. Hogy történik a kiolvasási művelet a STACK-nel?
E.11.10. Milyen fontosabb szektorokat (mezőket) ismer a memóriában?
E.11.11. Mire jók az IO-egységek?
E.11.12. Mi a *displacement*, milyen funkciói lehetnek?
E.11.13. Mutassa be egy LDA CÍM típusú utasítás lefolyását a berendezés HW összetevőin.
E.11.14. Ismertesse a CALL CIMS utasítás végrehajtásának mozzanatait.
E.11.15. Mutasson be egy *indexelt* címzést.
E.11.16. Mi az az IMMEDIATE?
E.11.17. Milyen elven címezhetők az IO-egységek, és mi az a *memory mapped* címzés?
E.11.18. Milyen lépések történnek az IN CIM IO utasítás folyamán?
E.11.19. Mire jó az *ugráscím-táblázat?*
E.11.20. Mire jó az EI utasítás és a DI utasítás?
E.11.21. Milyen feladatokat old meg egy párhuzamos IO-egység?
E.11.22. Milyen feladatokat old meg egy soros IO-egység?
E.11.23. Mi a különbség a *parancs* és a *státusz* között?
E.11.24. Milyen közvetlen kapcsolat szervezhető a memória és az IO-egységek között?
E.11.25. Mi az a HOLD és a HLDA?
E.11.26. Milyen lépésekből áll egy direkt PER → MEM átvitel?
E.11.27. Mi az a TRAP egy CPU-nál?
E.11.28. Milyen egy ROTATE utasítás?
E.11.29. Milyen *prioritás-viszonyokat* ismer RST_i utasításoknál?

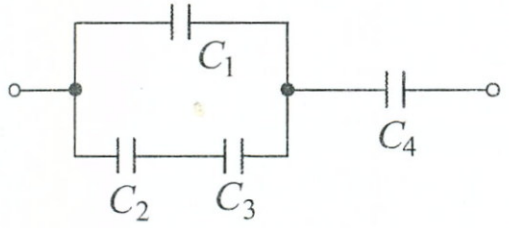
12. GYAKORLÓ FELADATOK ÉS MEGOLDÁSAIK

Az itt szereplő feladatok az egyes fejezetek tematikáihoz alkalmazkodó csoportosításban és sorrendben lettek összeállítva.

A *-gal jelölt *G. i. j. számozású feladatok megoldásai a rákövetkező pont F. i. j. jelű helyén találhatók. Néhány egyszerűbbnek tekintett esettől eltekintve, valamennyi feladat megoldása kidolgozásra került.

12.1.1. Gyakorló feladatok az 1. „Villamosságtani alapok" c. fejezethez

*G.1.1. Határozzuk meg a 12-1. ábra kapacitás-együttesének eredőjét, valamint az egyes kapacitások feszültségeit és töltéseit:



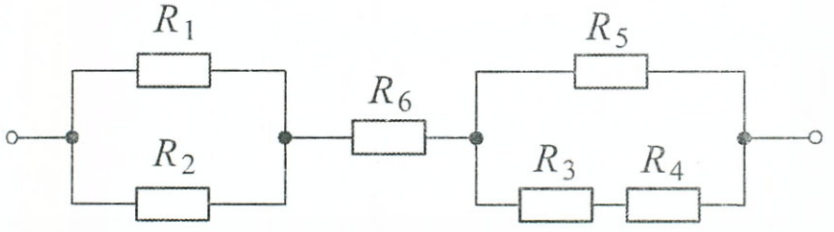
$U = 500$ V

$C_1 = 1\ \mu F$ $C_3 = 6\ \mu F$

$C_2 = 4\ \mu F$ $C_4 = 2\ \mu F$

12–1. ábra *Kapacitív hálózat*

*G.1.2. Mekkora ellenállással helyettesíthető a 12-2. ábra-beli kapcsolás?



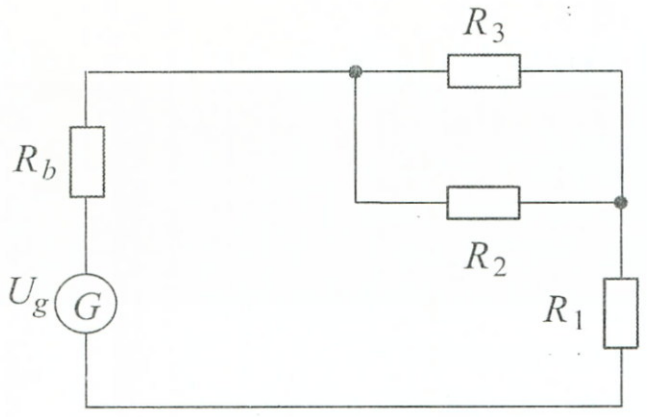
$R_1 = 4\ k\Omega$ $R_4 = 7\ k\Omega$

$R_2 = 4\ k\Omega$ $R_5 = 2\ k\Omega$

$R_3 = 1\ k\Omega$ $R_6 = 3\ k\Omega$

12–2. ábra *Ellenállás hálózattal*

*G.1.3. Határozzuk meg a 12-3. ábra hálózatában az R_3 ellenálláson átfolyó áram nagyságát.



$U_g = 120$ V $\quad R_1 = 12\ \Omega$
$R_b = 0{,}5\ \Omega \quad R_2 = 20\ \Omega$
$\quad\quad\quad\quad\quad\quad R_3 = 40\ \Omega$

12–3. ábra *Ellenállásos áramkör*

*G.1.4. Írjuk fel a G.1.3. példa hálózatának teljesítményviszonyait.

*G.1.5. Mekkora a tárolt energia egy 10 μF-os kondenzátorban, ha 6 kV feszültséggel töltjük?

*G.1.6. Mekkora a tárolt energia egy 0,6 *H* induktivitású tekercsben, ha a rajta átfolyó áram 800 mA?

*G.1.7. Mekkora az átfolyó áram és az egyes elemeken eső feszültség egy soros *RC* kapcsolásnál, ha $R = 12$ kΩ, $C = 0{,}2$ μF, és a tápfeszültség $U = 110$ V, $f = 50$ Hz?

*G.1.8. Milyenek a teljesítményviszonyok egy soros *RL* áramkörnél, ha 220 V-os 50 Hz-es hálózatra kapcsoljuk? $R = 50\ \Omega$, az induktív reaktancia: $X_L = 100\ \Omega$. Állapítsuk meg L értékét, és végül írjuk fel a komplex impedanciát is.

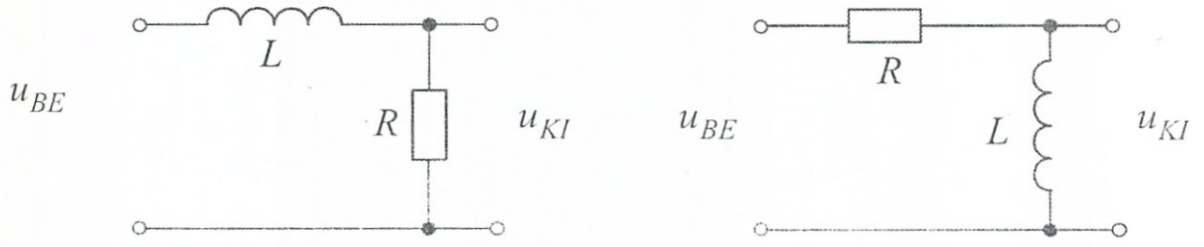
*G.1.9. Rajzoljuk fel a vektorábrákat.

a) soros RL
b) soros RC

kétpólusok esetén, színuszos jellel feltételezve.

*G.1.10. Miként alakul az U_2 feszültség az 1-54a. ábra négypólusánál, ha bekapcsoláskor a bemenetre U_1 nagyságú ugrásfüggvény érkezik?

*G.1.11. Vizsgáljuk meg a 12-4. ábra négypólusait abból a szempontból, hogy melyikük használható alul, ill. felüláteresztőként, periódikus jelek esetén. Adjuk meg a frekvencia jelleggörbéket.



a.) b.)

12–4. ábra *LR, RL négypólusok*

*G.1.12. Miként viselkednek a 12-4. ábra négypólusai, ha a bemenetükre U_0 nagyságú ugrásfüggvényt adunk?

12.1.2. Feladatmegoldások az 1. fejezethez

F.1.1. a) Az eredő kapacitás
Az 1-7a., és 1-7b. ábrákon összefoglalt soros- és párhuzamos eredőket jellemző formulák alapján írható C2 és C3 soros eredőjére:

$$\frac{1}{C_{2,3}} = \frac{1}{C_2} + \frac{1}{C_3}$$

ezt rendezve és behelyettesítve:

$$C_{2,3} = \frac{C_2 \cdot C_3}{C_2 + C_3} = \frac{4 \cdot 6}{4 + 6} = 2{,}4\ \mu\text{F}$$

C_1 és $C_{2,3}$ párhuzamos eredője:

$$C_{1,2,3} = C_1 + C_{2,3} = 1 + 2{,}4 = 3{,}4\ \mu\text{F}$$

végül a teljes eredő kapacitásra felírva:

$$\frac{1}{C} = \frac{1}{C_{1,2,3}} + \frac{1}{C_4}$$

$$C = \frac{C_{1,2,3} \cdot C_4}{C_{1,2,3} + C_4} = \frac{3{,}4 \cdot 2}{3{,}4 + 2} = 1{,}26\ \mu\text{F} = 1{,}26 \cdot 10^{-6}\ \text{F}$$

b) Az egyes kapacitásokon ébredő feszültséghez először kiszámítjuk az eredő töltést (1.8) alapján:

$$Q = C \cdot U = 1{,}26 \cdot 10^{-6} \cdot 500 = 6{,}3 \cdot 10^{-4} \text{ Coulomb}$$

Ezután C_4 és $C_{1,2,3}$ feszültségei:

$$U_4 = \frac{Q}{U} = \frac{6{,}3 \cdot 10^{-4}}{2 \cdot 10^{-6}} = 315 \text{ V}$$

$$U_{1,2,3} = 500 - 315 = 185 \text{ V}$$

Ez a feszültség lép fel az ábra bal oldali párhuzamos tagjának sarkain. Ezért

$$Q_1 = C_1 \cdot U_{1,2,3} = 1 \cdot 10^{-6} \cdot 185 = 185 \cdot 10^{-6} \text{ Coulomb}$$

$$Q_{2,3} = C_{2,3} \cdot U_{1,2,3} = 2{,}4 \cdot 10^{-6} \cdot 185 = 445 \cdot 10^{-6} \text{ Coulomb}$$

Végül C_2 és C_3-ra felírva:

$$U_2 = \frac{Q_{2,3}}{C_2} = \frac{445 \cdot 10^{-6}}{4 \cdot 10^{-6}} = 111 \text{ V}$$

$$U_3 = U_{1,2,3} - U_2 = 185 - 111 = 74 \text{ V}$$

F.1.2. Az eredő kiszámításához az 1-9. ábra formuláit használhatjuk fel. $R_{1,2}$-nél:

$$\frac{1}{R_{1,2}} = \frac{1}{R_1} + \frac{1}{R_2}$$

Rendezve és behelyettesítve kapjuk.

$$R_{1,2} = \frac{R_1 \cdot R_2}{R_1 + R_2} = \frac{4 \cdot 10^3 \cdot 4 \cdot 10^3}{4 \cdot 10^3 + 4 \cdot 10^3} = \frac{16 \cdot 10^6}{8 \cdot 10^3} = 2 \cdot 10^3 \ \Omega = 2 \text{ k}\Omega$$

Továbbá $R_{3,4}$-nél:

$$R_{3,4} = R_3 + R_4 = 1 \cdot 10^3 + 7 \cdot 10^3 = 8 \cdot 10^3 \ \Omega = 8 \text{ k}\Omega$$

$R_{3,4,5}$-nél:

$$\frac{1}{R_{3,4,5}} = \frac{1}{R_{3,4}} + \frac{1}{R_5}$$

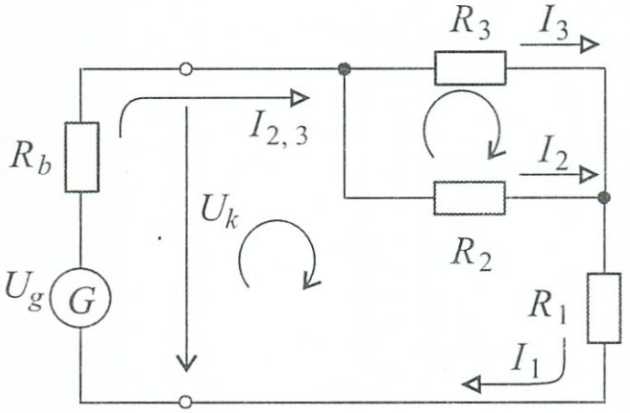
A rendezés után:

$$R_{3,4,5} = \frac{R_{3,4} \cdot R_5}{R_{3,4} + R_5} = \frac{8 \cdot 10^3 \cdot 2 \cdot 10^3}{8 \cdot 10^3 + 2 \cdot 10^3} = \frac{16 \cdot 10^6}{10 \cdot 10^3} = 1{,}6\ \text{k}\Omega$$

Végül az eredő:

$$R = R_{1,2} + R_6 + R_{3,4,5} = 2 + 3 + 1{,}6 = 6{,}6\ \text{kW}$$

F.1.3. a) A megoldás első lépéseként fel kell venni az áramok és feszültségek mérőirányait, ez a 12-5. ábrán látható:



12–5. ábra *Mérőirányok*

b) Az I_1, I_2, I_3 ismeretlenek száma három, megoldásukhoz a 1-12., és 1-13. ábrák csomóponti és hurok-törvényeit használhatjuk fel. (Kirchhoff-törvények) A felvett mérőirányok alapján felírható a következő egyenletrendszer:

$$I_2 + I_3 + I_1 = 0 \qquad (I_1 = I_{2,3})$$

$$I_3 \cdot R_3 - I_2 R_2 = 0 \qquad (I_2 \text{ szembe folyik a felvett hurokiránnyal})$$

$$-U_g + I_1 \cdot R_b + I_2 \cdot R_2 + I_1 \cdot R_1 = 0 \qquad (I_{2,3} = I_1)$$

c) Mivel a feladat I_3 értékének kiszámítása erre, mint ismeretlenre megoldjuk az egyenletrendszert és kifejezzük belőle I_3-at. Áramkörünk esetében (a részletezett lépéseket mellőzve) és a behelyettesítéseket elvégezve adódik:

$$I_3 = \frac{R_2 \cdot U_g}{(R_1 + R_b)(R_2 + R_3) + R_2 \cdot R_3} = \frac{20 \cdot 120}{(12 + 0{,}5)(20 + 40) + 20 \cdot 40} = 1{,}55\ \text{A}$$

Természetesen ugyanez az eredmény adódott volna, ha az egyes eredő ellenállásokkal és az Ohm-törvénnyel számolunk.

F.1.4. A teljesítményviszonyok jellemzéséhez az (1.34) formulákat használhatjuk fel. Például az R3 ellenálláson leadott teljesítmény:

$$P_3 = I_3^{\,2} \cdot R_3 = 1{,}55^2 \cdot 40 = 96{,}1\,\mathrm{W}$$

A többi ellenálláshoz tartozó teljesítmények hasonló módon számíthatók.

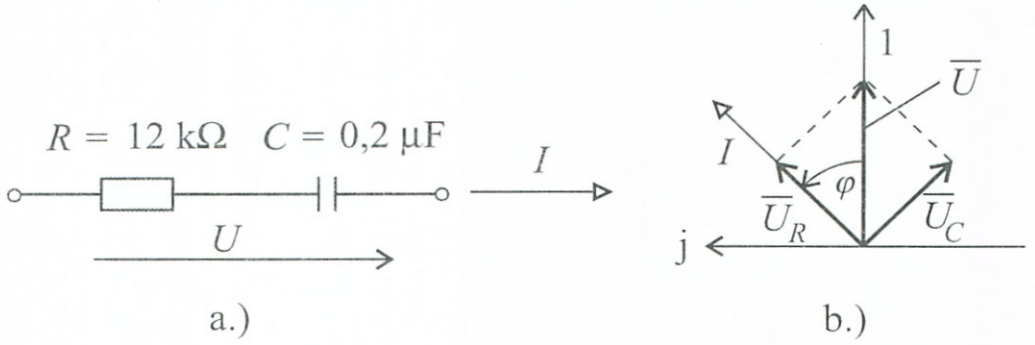
F.1.5. A kondenzátor tárolt energiája az (1.13) összefüggésekkel számolható. Esetünkben:

$$W = \frac{1}{2} C U^2 = \frac{1}{2} 10 \cdot 10^{-6} \cdot \left(6 \cdot 10^3\right)^2 = 180\,\mathrm{Ws}$$

F.1.6. A tekercsben tárolt energia az (1.64) összefüggéssel számolható. Esetünkben:

$$W = \frac{1}{2} L I^2 = \frac{1}{2} \cdot 0{,}6 \left(800 \cdot 10^{-3}\right)^2 = 0{,}192\,\mathrm{Ws}$$

F.1.7. A soros R–C kapcsolást felrajzoltuk a 12-6a. ábrán:



12–6. ábra *RC kapcsolás*

A kapcsoláson mérhető feltételezetten színuszos feszültséggel kapcsolatos összefüggéseket az (1.104) és (1.105) kifejezésekből olvashatjuk ki. Ezekből az eredő impedancia ($\omega = 2\pi f$ helyettesítéssel):

$$\underline{Z} = R + \frac{1}{j\omega C} = 12 \cdot 10^3 + \frac{1}{j2\pi \cdot 50 \cdot 0{,}2 \cdot 10^{-6}}\ \Omega$$

A komplex számokra összefoglalt 1.4.4. fejezet-beli összefüggések felhasználásával, továbbá a második tagban a számlálót és nevezőt $-j$-vel megszorozva, a számolások elvégzése után kapjuk az eredő impedancia komplex vektorát kanonikus alakban:

$$\underline{Z} = a + jb = R + jX_c = 12 \cdot 10^3 - j15{,}9 \cdot 10^3\ \Omega$$

És mint ismeretes, egy komplex vektor abszolút értéke az 1-33. ábra formuláiból kiolvashatóan:

$$Z = |\underline{Z}| = \sqrt{a^2 + b^2} = \sqrt{12^2 + (-15{,}9)^2} \cdot 10^3 = 20 \cdot 10^3\ \Omega$$

A fázisszög ugyancsak az 1-33. ábra segítségével:

$$\varphi = \operatorname{arctg} \frac{b}{a} = \operatorname{arctg} \frac{-15{,}9}{12} = -55°$$

Ennyivel „siet" az áramerősség a feszültséghez képest. Az áramerősség effektív értéke (1.105) figyelembevételével:

$$I = \frac{U}{Z} = \frac{110}{20 \cdot 10^3} = 5{,}5 \cdot 10^{-3}\ \text{A} = 5{,}5\,\text{mA}$$

Az R ellenálláson fellépő feszültség:

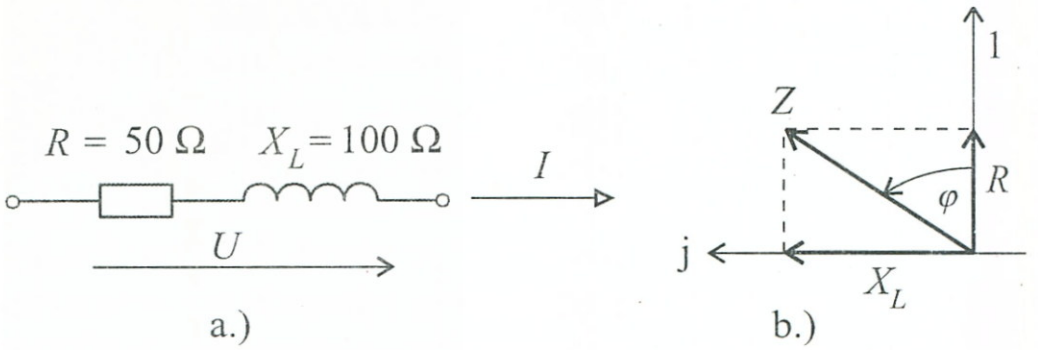
$$U_R = RI = 12 \cdot 10^3 \cdot 5 \cdot 5 \cdot 10^{-3} = 66\,\text{V}$$

Ez az árammal fázisban van és az U kapocsfeszültséghez képest ez is „siet" 55°-kal.
A C kondenzátoron fellépő feszültség:

$$U_C = X_C \cdot I = 15{,}9 \cdot 10^3 \cdot 5{,}5 \cdot 10^{-3} = 87{,}5\,\text{V}$$

Ez az áramhoz képest 90°-kal „késik", így a kapocsfeszültséghez képest 90°–55° = 35°-kal „késik".
Az elmondottak jobb szemléltetésére a 12-6b. ábrán (nem léptékhelyesen) felrajzoltuk a hálózat vektordiagramját is.

F.1.8. A soros R–L kapcsolást felrajzoltuk a 12-7a. ábrán:
A teljesítményviszonyokra az (1.110), (1.113), (1.114) formulák az irányadók színuszos jelek esetén.



12–7. ábra *RL kapcsolás*

Először kiszámítjuk az eredő impedanciát:

$$\overline{Z} = R + jX_L = 50 + j100\ \Omega$$

Az abszolútérték:

$$Z = \sqrt{R^2 + X_L^2} = \sqrt{50^2 + 100^2} \approx 112\ \Omega$$

Az áram:

$$I = \frac{U}{Z} = \frac{220}{112} = 1{,}97\ \text{A}$$

A cos φ értéke:

$$\cos\varphi = \frac{R}{Z} = \frac{50}{112} \approx 0{,}447\ (\varphi = 63°30')$$

$$\sin\varphi = \frac{X_L}{Z} \approx 0{,}894$$

Ezekből az egyes teljesítmények:
Látszólagos:

$$S = U \cdot I = 220 \cdot 1{,}97 = 433\,\text{VA}$$

Hatásos:

$$P = U \cdot I \cdot \cos\varphi = S \cdot \cos\varphi = 433 \cdot 0{,}447 = 194\ \text{W}$$

Meddő:

$$Q = U \cdot I \cdot \sin\varphi = S \cdot \sin\varphi = 433 \cdot 0{,}89 = 387\ \text{W}$$

Végezetül számítsuk ki az L értékét.
Mivel: $X_L = \omega L = 2\pi f L$

$$L = \frac{X_L}{2\pi f} = \frac{100}{2\pi \cdot 50} = \frac{1}{\pi} H$$

Az impedancia-vektorok a 12-7b. ábrán tanulmányozhatók.

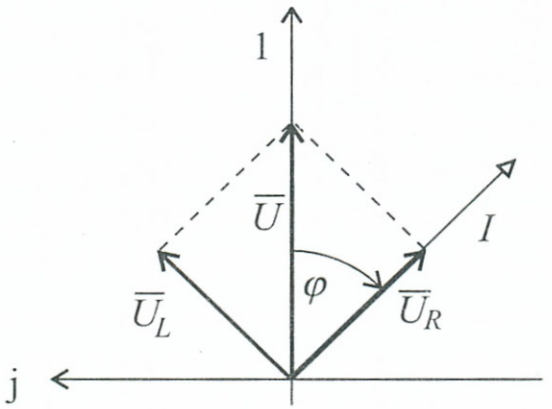
F.1.9. a) Soros R–L hálózatoknál az impedancia:

$$\overline{Z} = R + jX_L = R + j\omega L$$

a feszültségek:

$$\overline{U} = \overline{Z} \cdot \overline{I} = R\overline{I} + j\omega L \cdot \overline{I} = \overline{U}_R + \overline{U}_L$$

Itt $\overline{U}_R$ fázisban van az árammal, vektora egybeesik az áram irányával. $\overline{U}_L$ az áramhoz képest 90°-kal siet (az áram a feszültséghez képest késik). $\overline{U}_R$ és $\overline{U}_L$ eredője $\overline{U}$ 90°-nál kisebb szöggel siet az áramhoz képest.
Az elmondottaknak megfelelő vektor ábrát a 12-8. ábra mutatja.



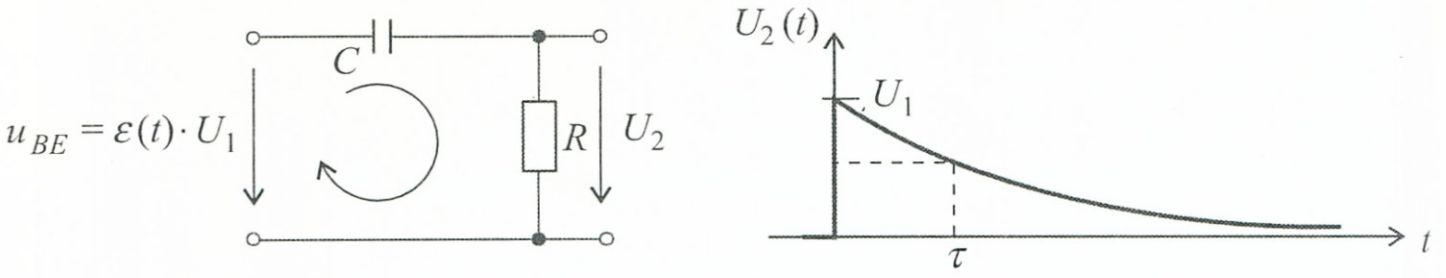
12–8. ábra *Vektorábra*

b) A soros RC viszonyait a 12-6b. ábrán már felrajzoltuk.

F.1.10. A példában vizsgálandó négypólust a 12-9a. ábrán rajzoltuk fel, feltüntetve a bemeneti ugrásfüggvényt is:
Az (1.79), (1.82) összefüggések felhasználásával felírható egy hurokegyenlet:

$$\frac{1}{C}\int_0^t i(v)dv + Ri - \varepsilon(t)U_1 = 0$$

(A felírásnál töltetlen kondenzátort tételeztünk fel, $Q_o = 0$)



a.) b.)

12–9. ábra *CR tag*

A kapott integro-differenciálegyenletet klasszikus úton megoldva megkaphatjuk az áram időbeli változását leíró függvényt.
Ennek érdekében mindkét oldalt differenciálva, az integráljel eltűnésével egyidejűleg kapjuk:

$$\frac{1}{C} i + R \frac{di}{dt} = 0$$

A változók szétválasztása után, ezt átrendezve, $RC = \tau$ helyettesítéssel:

$$\frac{di}{i} = -\frac{1}{\tau} dt$$

Mindkét oldalt integrálva és figyelembevéve a matematikából ismert, itt felmerülő alapintegrálokat, felírható:

$$\int \frac{1}{i} di = -\frac{1}{\tau} \int 1 \cdot dt$$

$$\ln i = -\frac{1}{\tau} t + k$$

azaz:

$$i(t) = e^{-\frac{1}{\tau} t + k} = e^k \cdot e^{-\frac{1}{\tau} \cdot t}$$

A határozatlan integrálból eredő k konstans meghatározására felhasználhatjuk az alábbi kezdeti feltételt, $t = +0$ pillanatra, $t = 0$ behelyettesítésével:

$$I_o = \frac{U_1}{R} = i(+0) = e^k \cdot e^{-\frac{1}{\tau}\cdot 0} = e^k \cdot 1 = e^k$$

ezt visszahelyettesítve az $i(t)$ függvénybe:

$$i(t) = \frac{U_1}{R} \cdot e^{-\frac{t}{\tau}}$$

A példában kérdezett U_2 kimeneti feszültség az R ellenállás sarkain mérhető:

$$U_2 = u_R = R \cdot i(t)$$

Ide behelyettesítve az áramot a kérdéses kimeneti feszültség:

$$u_{ki}(t) = U_2 = u_R = R \cdot \frac{U_1}{R} \cdot e^{-\frac{t}{\tau}} = U_1 \cdot e^{-\frac{t}{\tau}}$$

A kimeneti feszültség időbeli alakulását a 12-9b. ábrán rajzoltuk fel.

A matematikai úton nyert eredményt egyszerű meggondolásokkal ellenőrizhetjük: A $t = +0$ pillanatban a töltetlen kondenzátorba akadálytalanul áramolhatnak a töltések, a kondenzátor „rövidzárként" viselkedik, így az R ellenállás sarkain a teljes bemenő feszültség megjelenik. A töltődés során a kondenzátor egyre jobban „ellenáll" a töltésáramlásnak, nő a rajta eső feszültség, miközben az R ellenállás sarkain folyamatosan csökken a 12-9b. ábrán ellenőrizhető módon.

A feladat rokonságban áll az 1.16. példával.

F.1.11. A feladatok megoldása azonos lépésekben történik, mint az 1.22. Példa, ill. az 1.23. Példa történései, így a további részletektől eltekintünk. A lényeges különbség, hogy az induktív reaktancia:

$$X_L = \omega L$$

növekvő frekvencia esetén növekvő-, csökkenő frekvencia esetén csökkenő, egyenáramnál pedig nulla-ellenállásként viselkedik.

Emiatt az RL tag felüláteresztő, az LR tag aluláteresztő jellegű.

F.1.12. A feladatmegoldásnál támpontul szolgálhatnak az 1.24. Példa, ill. 1.25. Példa lépései, ezért itt nem bocsátkozunk részletekbe. A lényeges különbség az operátoros impedanciánál van, mivel itt induktivitás szerepel a kapacitás helyett, ezért:

$$Z(s) = R + sL$$

továbbá: $\tau = \frac{L}{R}$ az időállandó

E változások alapján számolva, végül kiadódik, hogy az

RL differenciáló	$(\tau < 1)$
LR integráló	$(\tau > 1)$

négypólusként viselkedik.

12.2.1. Gyakorló feladatok a 2. „Logikai rendszerleírás alapjai" c. fejezethez

*G.2.1. Rajzoljunk fel példákat analóg jelekre.

*G.2.2. Rajzoljunk fel példákat digitális jelekre.

*G.2.3. Adva a következő igazságtáblázat:

a) Írjuk fel az függvényt

– diszjunktív és
– konjunktív

kanonikus alakban.

b) Ábrázoljuk a diszjunktív alakot VENN diagramon és V–K táblán.

c) Adjuk meg a konjunktív alak duálját.

d) Írjuk fel az függvényt a két kanonikus alakban.

e) Számítsuk ki a függvény K sorszámát.

C	B	A	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

*G.2.4. Bizonyítsuk be, hogy az:

ANTIVALENCIA - ÉS

függvénycsoport funkcionálisan teljes. A bizonyításnál felhasználhatjuk, hogy a NEM-ÉS-VAGY rendszer bizonyítottan funkcionálisan teljes.

G.2.5. Igazoljuk a tanult módszerek bármelyikével, hogy fennállnak a következő algebrai összefüggések:

a) $X \oplus 0 = X$

b) $X \oplus 1 = \overline{X}$

c) $X \oplus X = 0$

d) $X \oplus \overline{X} = 1$

e) $X \oplus Y = \overline{X \equiv Y}$

f) $(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$

g) $\underline{X \cdot (Y} \oplus Z) = X \cdot Y \oplus X \cdot Z$

h) $X \oplus Y = X \oplus \overline{Y} = \overline{X} \oplus Y = X \cdot Y + \overline{X} \cdot \overline{Y}$

i) $X \oplus Y = X \cdot \overline{Y} + \overline{X} \cdot Y$

$j)\ X \equiv Y = (X + \overline{Y}) \cdot (\overline{X} + Y)$
$k)\ X \oplus Y = (X + Y) \cdot (\overline{X} + \overline{Y})$
$l)\ X \oplus Y = (X|\overline{Y})|(\overline{X}|Y)$
$m)\ X \oplus Y = (X\|Y)\|(\overline{X}\|\overline{Y})$
$n)\ X \equiv Y = (X|Y)|(\overline{X}|\overline{Y})$
$o)\ X \equiv Y = (X\|\overline{Y})\|(\overline{X}\|Y)$

*G.2.6. Igazoljuk a következő BOOLE algebrai azonosság helyességét:
a) - táblázatosan
b) VENN diagrammal

$$XY + \overline{X}Z + YZ = XY + \overline{X}Z$$

*G.2.7. Keressük meg az alábbi függvény

$$Y = ABC + DEF$$

SHEFFER (NAND) szimbólumokkal megadható logikai vázlatát.

*G.2.8. Adva a következő ÉS–VAGY struktúrájú függvény:

$$Y = (A + B) \cdot (\overline{A} + C)$$

a) Adjuk meg logikai vázlatát.
b) Alakítsuk át VAGY–ÉS struktúrájúvá és adjuk meg ennek az alaknak is a logikai vázlatát.
c) Állítsuk elő az igazságtáblázatot.

G.2.9. Az előző példabeli függvénynél:
a) Adjuk meg a logikai vázlatot kizárólag NAND elemek segítségével.
b) Adjuk meg a logikai vázlatot kizárólag NOR elemek segítségével.

G.2.10. Adva a következő logikai függvény:

$$Y = \sum^{4}(1,4,5,8,9,12,13)$$

a) Keressük meg a diszjunktív minimál alakot.
b) Állítsuk elő a konjunktív minimál alakot.

*G.2.11. Adva a következő logikai függvény:

$$Y = \prod^{4}(2^h, 4^h, 5,6,7,9,11,12,15)$$

a) Írjuk fel a minimál alakokat.
b) Soroljuk fel a konjunktív minimál alaknál felhasznált implikánsokat.

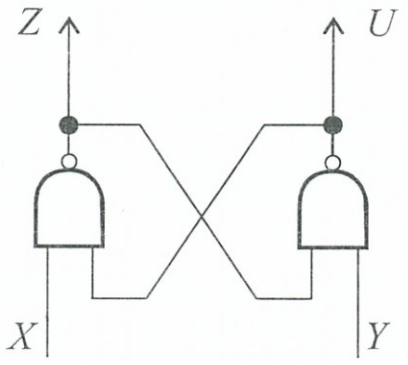
*G.2.12. Határozzuk meg a következő két függvény ekvivalenciájaként előállított eredő függvényt:

$$Y_1 = F_I^4(D,C,B,A) = \sum^{4}(1,3,5,7,9,11,12,14)$$

$$Y_2 = F_{II}^4(D,C,B,A) = \sum^{4}(2,3,4,5,8,9,12,13)$$

majd írjuk fel a legegyszerűbb algebrai alakban.

*G.2.13. Vizsgáljuk meg, hogy realizálható-e egy inverz SR tárolóelem a 12-10. ábra-beli hálózattal:



12–10. ábra *Keresztbecsatolt NAND hálózatok*

*G.2.14. Vizsgáljuk meg, miként lehetne *J–K* tárolóból:
a) *D* - típusú
b) *T* - típusú
c) *S–R* - típusú
tárolóelemet előállítani.

12.2.2. Feladatmegoldások a 2. fejezethez

F.2.1. Hasonló ábrák láthatók a 2-1a. ábrán.

F.2.2. Hasonló ábrák láthatók a 2-1b. ábrán.

F.2.3. ad. a.) A (2.7) megállapításból kiindulva az Y = 1-es sorokat ÉS term-ként értelmezve, ezek összege adja a diszjunktív alakot:

$$Y = \overline{C} \cdot \overline{B} \cdot A + \overline{C} \cdot B \cdot \overline{A} + C \cdot \overline{B} \cdot \overline{A} + C \cdot \overline{B} \cdot A + C \cdot B \cdot A =$$
$$= E_1^3 + E_2^3 + E_4^3 + E_5^3 + E_7^3 =$$
$$= \sum^3 (1,2,4,5,7)$$

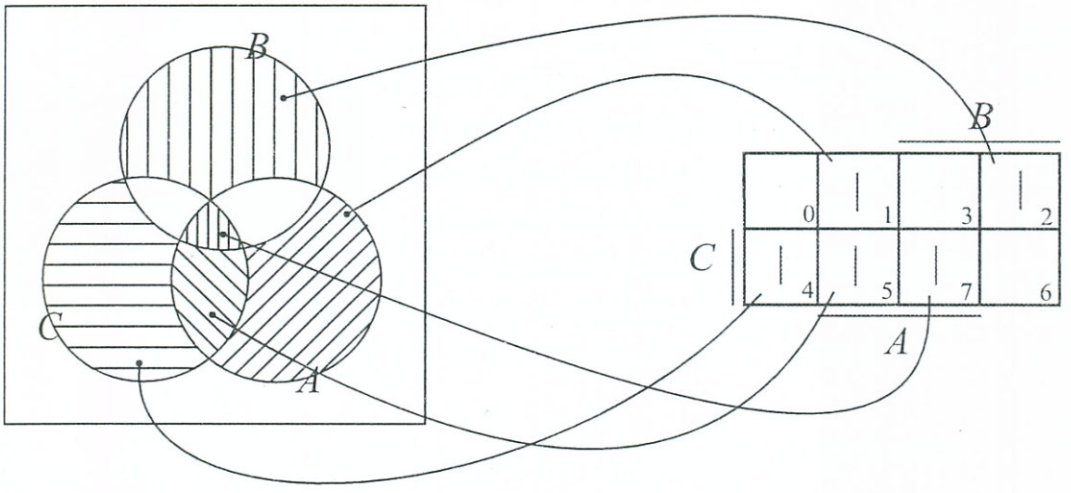
A konjunktív alakot a táblázatból közvetlenül a (2.8) megállapítás alapján írhatjuk fel, az Y = 0-ás soroknál a „0"-ákkal értelmezett VAGY term-ek szorzataként.

$$Y = (C + B + A)(C + \overline{B} + \overline{A})(\overline{C} + \overline{B} + A) =$$
$$= V_7^3 \cdot V_4^3 \cdot V_1^3 = \prod^3 (1,4,7)$$

ad. b) A VENN, ill. V–K táblákat a 12-11a., ill. 12-11b. ábrák mutatják.
Az előzőleg felírt konjunktív alakot a maxterm táblából (12-11c. ábra) ellenőrizhetjük, ezt a (2.11) megállapítás alapján rajzoltuk fel.

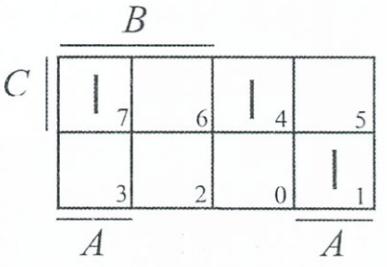
ad. c) A konjunktív alak *duál*-ját a (2.6) összefüggés alapján írjuk fel:

$$F_K^D = \left[(C + B + A) \cdot (C + \overline{B} + \overline{A}) \cdot (\overline{C} + \overline{B} + A)\right]^D =$$
$$= \overline{(\overline{C} + \overline{B} + \overline{A}) \cdot (\overline{C} + B + A) \cdot (C + B + \overline{A})}$$
$$= C \cdot B \cdot A + C \cdot \overline{B} \cdot \overline{A} + \overline{C} \cdot \overline{B} \cdot A$$



a.) b.) *minterm*

c.) *maxterm*



12–11. ábra *VENN és V – K ábrázolás*

ad. d) A két kanonikus alak negáltját mind az igazságtáblázatból, mind a V–K táblák 0-ás term-jeinek eredőjeként felírhatjuk.

$$\overline{Y}_{DISZJ.} = \sum^{3}(0,3,6)$$

$$\overline{Y}_{KONJ.} = \prod^{3}(0,2,3,5,6)$$

ad. e) A függvény index-számát a 2.5. Példa tapasztalatai alapján számíthatjuk ki az igazságtáblázat Y értékeiből képzett bináris számból:

$$K = 10110110_2 = 1\cdot2^7 + 1\cdot2^5 + 1\cdot2^4 + 1\cdot2^2 + 1\cdot2^1 =$$

$$= 128 + 32 + 16 + 4 + 2 =$$

$$= 182_{10}$$

$$Y = F^{3}_{182}(C, B, A)$$

F.2.4. A bizonyításnál feltételezzük, hogy a NEM-ÉS-VAGY rendszer funkcionálisan teljes. Ezután, ha ki tudjuk fejezni a fentieket ANTIVALENCIA-ÉS rendszerben, akkor ezt a rendszert is funkcionálisan teljesnek tekinthetjük.

- ÉS művelet: már eleve adva van az új rendszerben
- NEM művelet: Ez az ANTIVALENCIÁ-val kifejezhető, de csak akkor, ha még hozzávesszük az F_3^1 (2-8. ábra) „1" forrásfüggvényt: $F_3^1 = 1$, ekkor:

$$\overline{X} = 1 \oplus X$$

- VAGY művelet: Ezt részletezni már nem is szükséges, mivel a VAGY a DE MORGAN azonosság (2-12a. ábra 9. formula) segítségével kifejezhető az ÉS, ill. NEM műveletekkel.

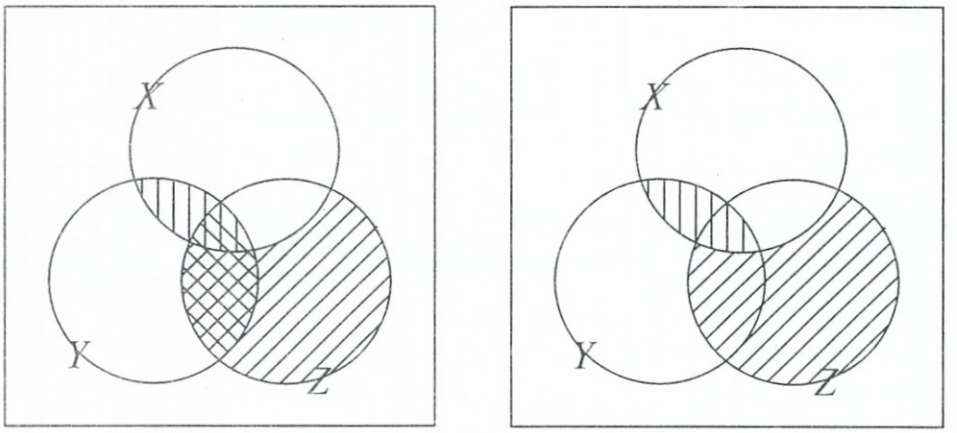
A végső konklúzió: funkcionálisan teljes az:

ANTIVALENCIA-ÉS-"1"

rendszer.

X	Y	Z	$\overline{X}$	XY	$\overline{X}Z$	YZ	*Bal oldal*	*Jobb oldal*
0	0	0	1	0	0	0	0	0
0	0	1	1	0	1	0	1	1
0	1	0	1	0	0	0	0	0
0	1	1	1	0	1	1	1	1
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	1	0	0	1	1
1	1	1	0	1	0	1	1	1

a.)



b.)

Bal oldal *Jobb oldal*

$XY + \overline{X}Z + YZ = XY + \overline{X}Z$ igazolása

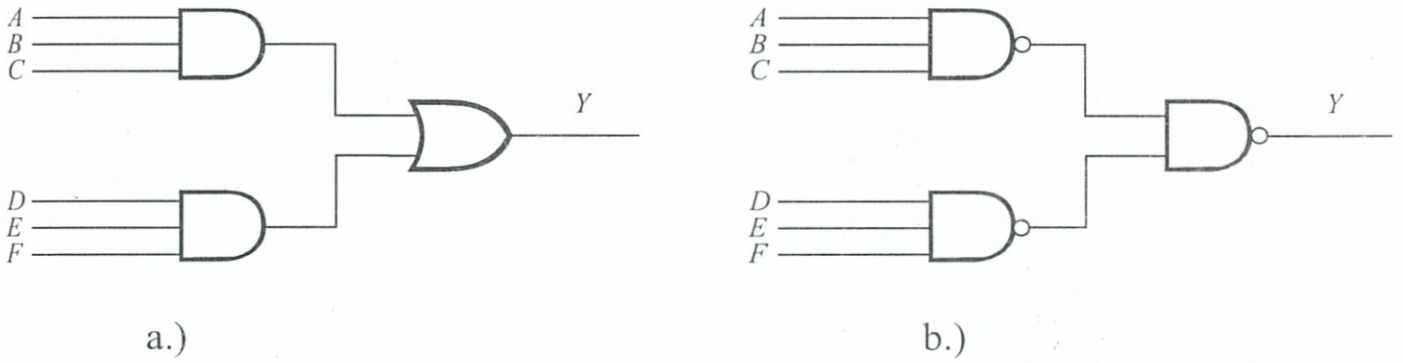
12–12. ábra

F.2.6. A táblázatos, ill. grafikus igazolást lépésenként a 12-12a., ill. 12-12b. ábrán követhetjük, külön elvégezve a műveleteket az azonosság bal, ill. jobb oldalán.
A VENN diagramból jól látható, hogy a baloldalon az $X \cdot Y \cdot Z$ jelű középső mező és a mellette lévő $\overline{X} \cdot Y \cdot Z$ *mező kettőzötten túlfedett, melyet az* $Y \cdot Z$ szorzat okoz, így az elhagyható. A szabályt fenti tulajdonság miatt *abszorpciós* szabálynak is nevezik.

F.2.7. Felírva és alkalmazva a 2-12b. ábra 24a. formuláját:

$$X \cdot Y \cdot Z + U \cdot V \cdot W = (X|Y|Z)|(U|V|W)$$

közvetlenül felrajzolható a 12-13a. ábrával jellegében hasonló struktúrát mutató 12-13b. ábra szerinti megoldás



a.) b.)

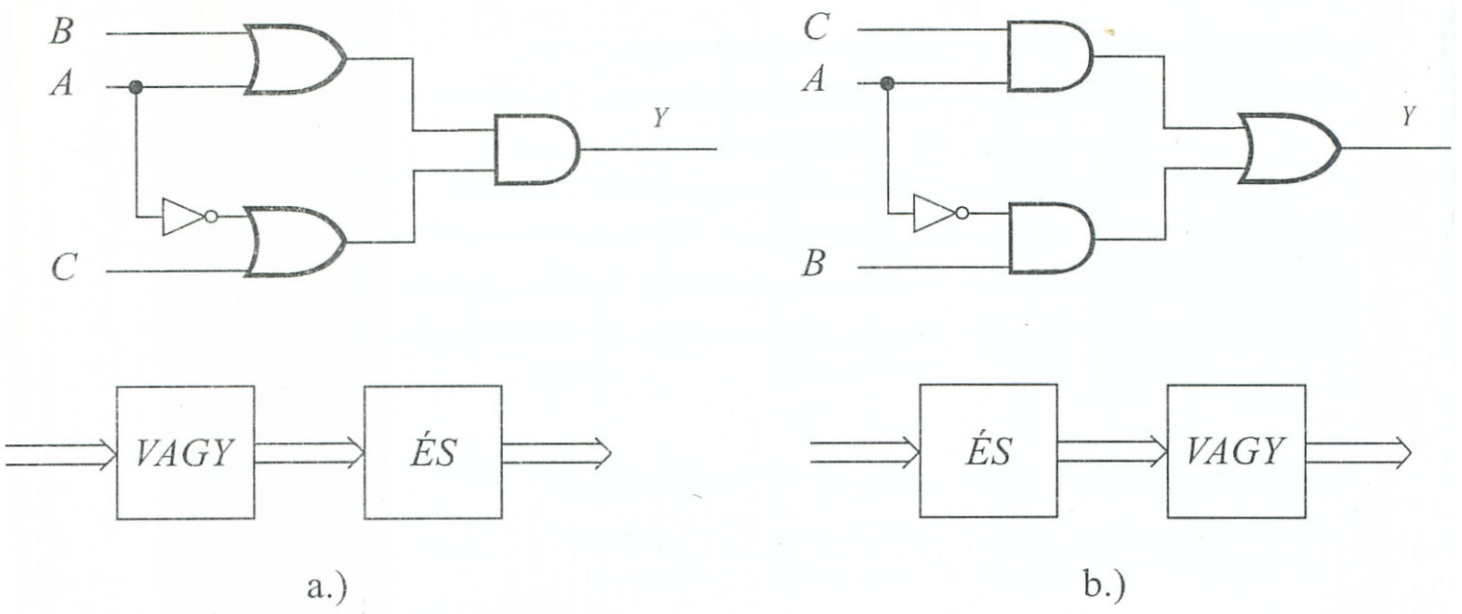
12–13. ábra *ÉS - VAGY és NAND - NAND rokonság*

F.2.8. Alkalmazva a 2-12a. ábra 12. formuláját, felírható:

$$(A + B)(\overline{A} + C) = A \cdot C + \overline{A} \cdot B$$

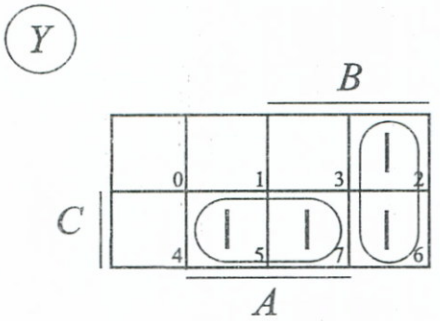
itt a baloldal ÉS-VAGY struktúrájú, a jobboldal VAGY-ÉS struktúrájú. A logikai vázlatok a 12-14. ábrán láthatók.
Az igazságtáblázatot a diszjunktívvá alakított formula tömbjeinek minterm táblába történő berajzolása után oldhatjuk meg. A minterm táblát a 12-15a. ábrán, a belőle kiolvasható logikai táblázatot a 12-15b. ábrán rajzoltuk fel:

F.2.11. Felrajzolva a maxterm táblát a kiinduló konjunktív alak segítségével, majd a 2.2.4. pont felhasználásával a minterm táblát, ezekből - tömbösítés után - leolvashatók a minimál



a.) b.)

12–14. ábra *Struktúraváltás*



m_i	C	B	A	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

a.) b.)

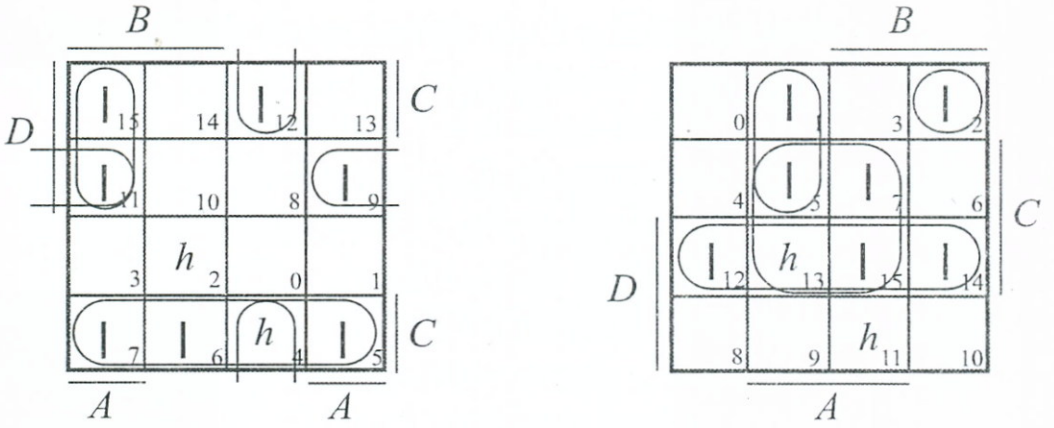
12–15. ábra *V – K tábla és igazságtáblázat*

alakok, melyek összetevői egyben a kérdéses implikánsok. (12-16. ábra)

F.2.12. A megoldást *négyváltozós V–K* táblákkal kell elvégezni, a 2. fejezet 2.7. Példa e.) változatával megegyező módon.

F.2.13. Írjuk fel például a Z kimenetet algebrai alakban az ábra alapján:

$$Z' = X|U = X|(Y|Z) = \overline{X \cdot (\overline{Y \cdot Z})}$$



Konjunktív *Diszjunktív*

$$Y_{KONJ.} = (\overline{D} + C) \cdot (D + B + A) \cdot (C + \overline{B} + \overline{A}) \ (D + \overline{C} + A)$$

$$Y_{DISZJ.} = D \cdot C + C \cdot A + \overline{D} \cdot \overline{B} \cdot A + \overline{D} \cdot \overline{C} \cdot B \cdot \overline{A}$$

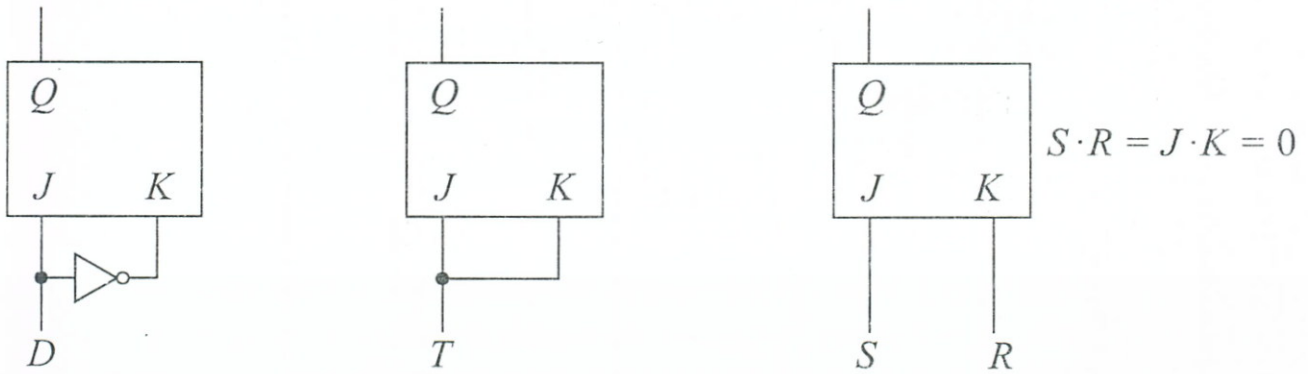
12–16. ábra *Minimál alakok*

$$Z' = \overline{X} + Y \cdot Z$$

Ez összhangban van a 2-26. ábrával, tehát az inverz *S-R* egy realizációs esetének tekinthető.

F.2.14. Összehasonlítva a 2-25. ábrán látható *J–K* definíciós táblázatot:

a) esetben a D tárolóéval, észrevehető, hogy: $J = \overline{K} = D$ megvalósításával a *J–K* táblázat 2. és 3. sorát állítottuk be, ami egyezeik a D táblázattal.



a.) b.) c.)

12–17. ábra *D, T, S–R leszármaztatása J–K-ból*

b) esetben a T tárolóéval, észrevehető, hogy: $J = K = T$ megvalósításával a *J–K* táblázat 1. és 4. sorát állítottuk be, ami egyezik a T táblázattal.

c) esetben az S–R tárolóéval, észrevehető, hogy csak le kell tiltani az $S = R = 1$ esetet és akkor a *J–K*, *S–R*-ré válik.

Az elmondottakat áramkörileg a 12-17. ábra szemlélteti

12.3.1. Gyakorló feladatok a 3. „Elektronikus áramkörök építőelemei" c. fejezethez

*G.3.1. Egy D = 10 mm átmérőjű porcelán hengerre, nem bifilárisan l = 40 mm hosszon, N = 100 menetnyi ellenálláshuzal van feltekercselve. Az ellenálláshuzal átmérője d = 0,25 mm, fajlagos ellenállása ρ = 0,5 ohm mm^2/m. Vizsgáljuk meg, hogy mekkora az ellenállás ennél a *huzalellenállásnál* egyenáram, ill. 50 Hz-es váltóáram esetén.

*G.3.2. Egy ohmos ellenállásnál felismerhetetlenné vált a külső gyűrűs jelölés. Bemérés után R = 187,3 kΩ ellenállás-érték adódott. Mi lehetett az eredeti színkódos jelölés?

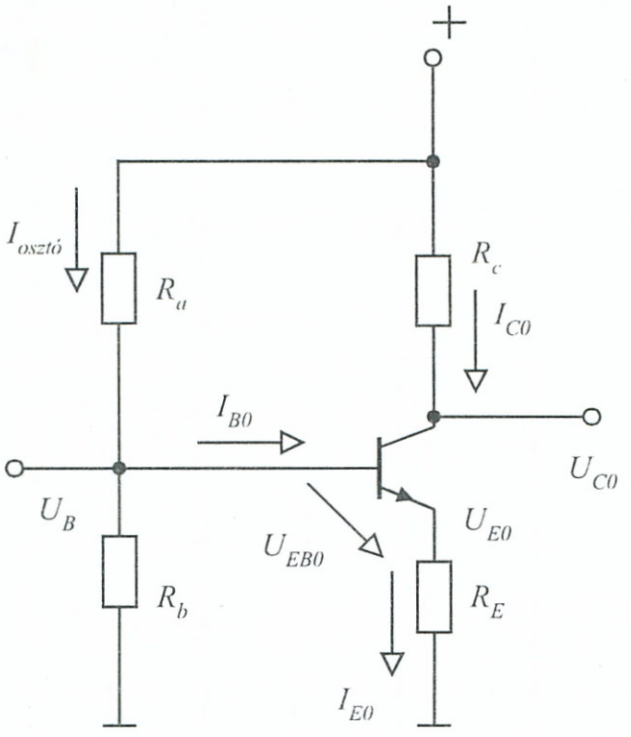
*G.3.3. Adva egy C_{max} = 300 pF-es forgókondenzátor, mely 1:10 arányban változtatható. Milyen v kapacitásviszony állítható be, ha vele C_F = 300 pF-es fix kondenzátort kapcsolunk:
a) párhuzamosan
b) sorosan

*G.3.4. Alakítsunk ki olyan diódás hálózati egyenirányítót, amely a 3-33. ábra egyenirányítójánál annyival jobb, hogy mindkét színusz félhullámot hasznosítja.

*G.3.5. A Zener-diódákkal foglalkozó fejezet (3.20) összefüggésével kapcsolatosan láttuk, hogy $|U_{ZJ}|$ = 5,7 V feszültség alatt a hőfokténvező negatív, felette pozitív viselkedésű. Vizsgáljuk meg, hasznosítható-e ez a tény hőstabilizálásra.

*G.3.6. Egy földelt emitteres kapcsolású bipoláris tranzisztor kollektor ellenállásaként egy L induktivitású tekercs van beiktatva, vizsgáljuk meg, hogy kapcsolóüzemmód esetén, itt milyen problémák lépnek fel.

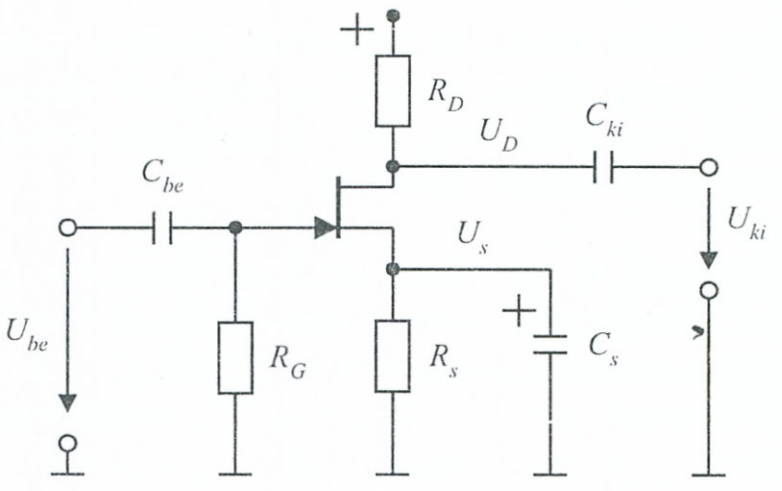
*G.3.7. Végezzük el egy földelt emitteres kapcsolású bázisosztóval, és emitterköri soros visszacsatolással rendelkező bipoláris tranzisztor munkapont-beállítását. (12-18. ábra) Kiindulóadatok: β = 100, U_{EBO} = 0,5 V, beállítandó kollektor áram $I_{co} \approx 1\,\text{mA}$, tápfeszültség: $U_+ = 12\,\text{V}$ és az emitterpont feszültsége U_{EO} = 1V körül legyen. A bázisosztó áramát $I_o \geq 10 I_{Bo}$-ra állítsuk be.



12–18. ábra *Munkapontbeállítás bipoláris kapcsolásnál*

*G.3.8. Számítsuk ki egy földelt kollektoros fokozat bemenő- és kimenő ellenállását, ha $\beta = 300$, $R_E = 3$ kΩ a meghajtó generátor ellenállása: $Rg = 40$ kΩ és $I_c = 2$ mA.

*G.3.9. Végezzük el a 12-19. ábra-beli kapcsolás munkapont-beállítását $I_D = 3$ mA áramra, ha feltételezzük, hogy az adott FET-nél $U_p = -3$ V, $I_{DS} = 10$ mA.



12–19. ábra *Munkapontbeállítás FET-nél*

*G.3.10. A tirisztorok fázis-levágásos elvű üzemeltetése (lásd pl. a 3.10. példát) radikális jeltorzulásokkal jár együtt, melyek (lásd 1. fejezet Fourier közelítés) sok felharmonikust okozva zavarjeleket hoznak létre. Minél nagyobb a vezérelt teljesítmény, annál nagyobb zavarás éri a környezetet. Keressünk megoldást, mellyel a zavarokat csökkenteni lehet.

12.3.2. Feladatmegoldások a 3. fejezethez

F.3.1. Az ellenálláshuzal *nem bifilárisan* van tekercselve, ezért *nem egyenáramú esetben* légmagos tekercsnek is tekinthető. Emiatt az eredő impedanciát soros *R–L* kapcsolásként kell kezelnünk, és hasonlóan kell számolnunk, mint az 1. fejezet G.1.8. példája esetében. Egyenáramu esetben csak az ohmos összetevővel kell számolnunk. Az *ohmos ellenállású R* összetevőt az 1. fejezet (1.21) formulájával írhatjuk fel:

$$R = \rho_H \cdot \frac{l_H}{A_H}$$

ahol a *H* indexek a huzal adataira utalnak:

$$A_H = \frac{d^2\pi}{4}$$

a huzal hosszát az egy tekercsmenet hosszának és a menetszámnak szorzata adja:

$$l_H \approx D \cdot \pi \cdot N$$

Az *önindukciós tényezőt* az 1. fejezet (1.59) formulájából kiindulva számíthatjuk ki:

$$L = \mu \cdot \frac{A_T}{l_T} \cdot N^2$$

ahol a *T* indexek a tekercs adataira utalnak:

$$A_T = \frac{D^2\pi}{4}$$

$l_T = 40$ mm adva van.

μ = légmagos esetre kell behelyettesíteni.
A *reaktáns összetevő:*

$$X_L = \omega L = 2\pi f \cdot L$$

ahol f = 50 Hz adva van a kiinduló adatok között.

F.3.2. A megoldásnál a tűréstartományra is gondolni kell: R = 187,3 kΩ = $(187 \cdot 10^3 + 0{,}3 \cdot 10^3)$ ohm
A lehetséges színkód:

1	8	7	10^3	2%-ba belefér
barna	szürke	ibolya	narancs	vörös

F.3.3. Az *a) párhuzamos* esetben a kapacitások összeadódnak
az alsó határ: $C_{min} + C_F = 330$ pF
a felső határ: $C_{max} + C_F = 600$ pF
A kapacitásviszony:

$$V_P = \frac{600}{330} = 1{,}82$$

A *b) soros* esetben a reciprokok adódnak össze

az alsó határ: $C_{\min} \times C_F = \dfrac{30 \cdot 300}{30 + 300} = 27{,}8$ pF

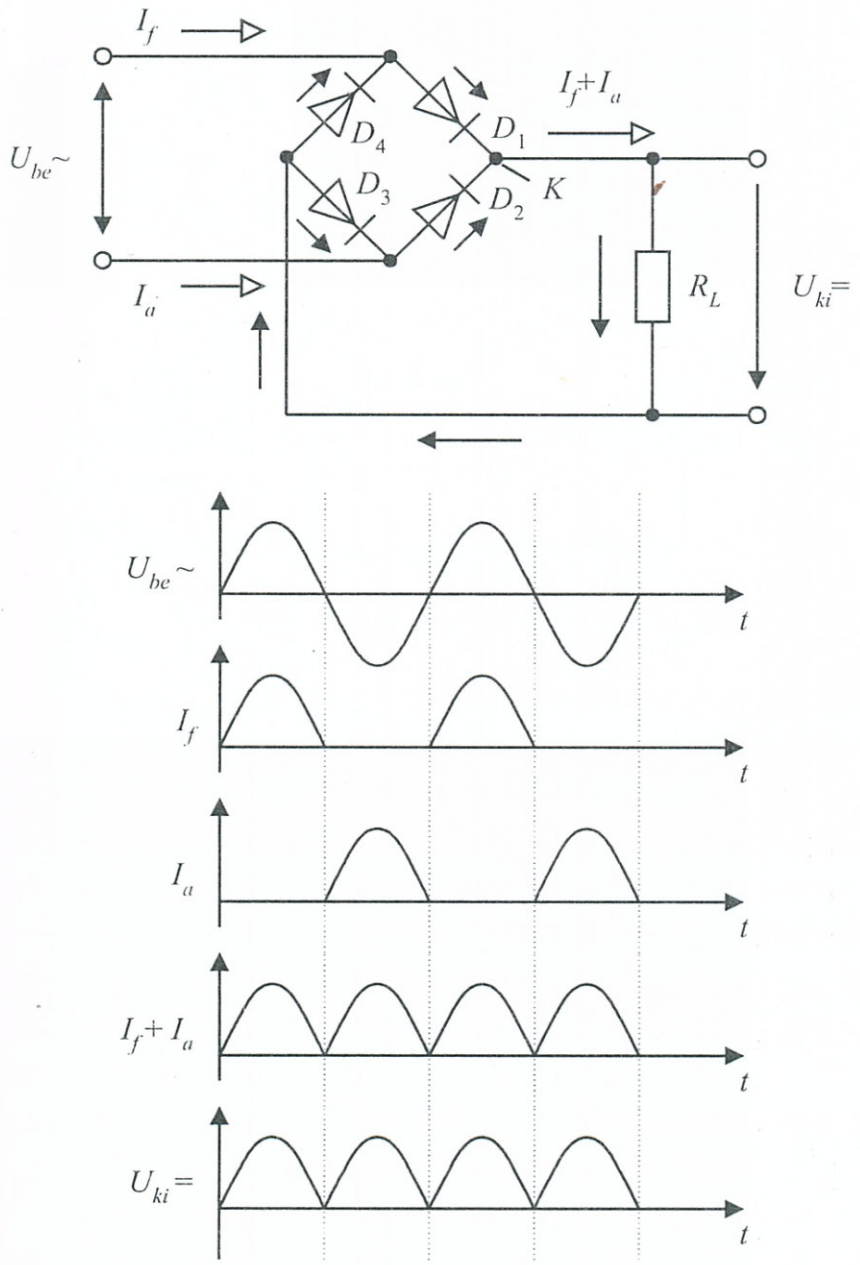
a felső határ: $C_{\max} \times C_F = \dfrac{300 \cdot 300}{300 + 300} = 150$ pF

a kapacitásviszony:

$$V_s = \frac{150}{27{,}8} = 5{,}5$$

F.3.4. Egy lehetséges megoldás az ún. *Graetz-kapcsolás* révén adódhat, melyet a 12-20. ábrán rajzoltunk fel a be- és kimeneti jelalakok feltüntetésével. A hídba kapcsolt négy dióda mind a pozitív, mind a negatív színuszfélhullámot a K kimeneti pontra tereli, így ezek az idődiagram szerint egyetlen lüktető félhullám-sorba rendeződnek.

F.3.5. A megoldás egyszerűen adódik. Például U_Z = 14 V-os Zenerfeszültség-igény esetén az egyetlen U_{ZJ} = 14 V-os



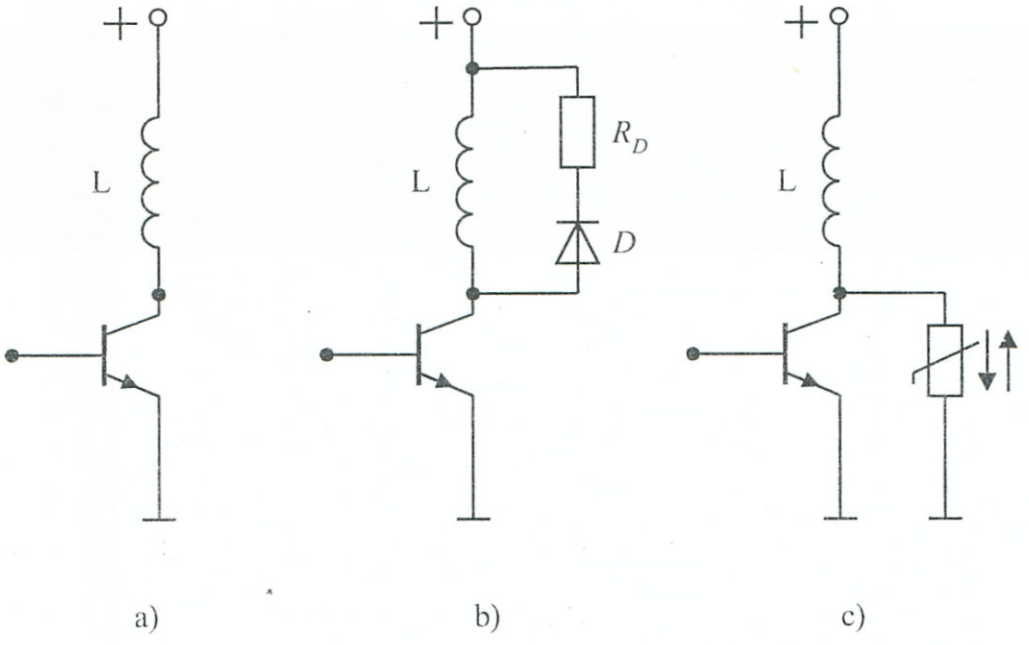
12–20. ábra *Graetz-kapcsolás és a jel-viszonyok*

Z-dióda helyett a hőstabilizálás érdekében célszerűbb beépíteni egy *Za–Zb* diódákból sorosan kapcsolt együttest:

$$U_{ZJ} = U_{ZJa} + U_{ZJb} = 10 + 4 = 14 \text{ Volt}$$

ahol $U_{ZJa}(U_{ZJa} > 5{,}7V)$ pozitív, U_{ZJb} negatív $(U_{ZJb} < 5{,}7V)$ hőmérsékleti tényezői egymást kompenzálják.

F.3.6. A 12-21a. ábrán látható, kollektorkörében tekercset tartalmazó tranzisztor záró irányba történő kapcsolásakor a rendkívül meredek változás miatt nagy feszültség indukálódik, mely meghaladhatja a tranzisztor megengedett legnagyobb



12–21. ábra *Túlfeszültség védelem induktív trehelésnél*

feszültségét és így a tranzisztor tönkremehet. Az indukált feszültség ellenkező irányítottságú a Lenz-törvény értelmében, ezért ezt figyelembe kell venni a védőáramkör kialakításakor.

A 12-21b. ábrán látható megoldásnál egy *D*-diódát kapcsoltunk a tekercscsel párhuzamosan. Normál működéskor a dióda zárva van, így nem sok vizet zavar. Az áram megszakításakor indukálódó feszültség (mivel az előzőhöz képest fordított polaritású) nyitja a diódát, mely kis ellenállású szakaszán a tekercs sarkait „rövidre zárván" a kollektorponti feszültséget nem engedi veszélyesen magasra nőni. Az R_D ellenállás úgy választják meg, hogy - ha szükséges - korlátozza a dióda tranziens áramát és így védje a diódát, de a kollektorponti feszültséget se engedje magasra nőni. Gyakran R_D el is maradhat.

A 12-21c. ábrán egy *VDR* ellenállás van a kapcsoló tranzisztorral párhuzamosan kapcsolva. A *VDR* ellenállás a növekvő feszültséggel egyidőben csökkenti ellenállását, így korlátozza a kollektorponton megengedettnél nagyobb feszülség kialakulását.

F.3.7. A kiinduló feltételeket figyelembevéve, meg kell határoznunk a 12-18. ábrán feltüntetett jellemzők értékeit. A meg-

oldás során felhasználjuk a 3.6.3.1. pontban elmondottakat is. Az R_E *emitterköri visszacsatoló ellenállás* a megadott $U_{Eo} \approx 1\text{V}$, $I_{co} \approx 1\text{mA}$ $\beta = 100$ értékekkel, mivel:

$$I_{Bo} = \frac{I_{co}}{\beta} = \frac{1}{100} = 0{,}01\text{mA} = 10\,\mu\text{A}$$

és: $I_{Eo} = I_{co} + I_{Bo} \approx I_{co} = 1\text{mA}$

innen: $R_E = \frac{U_{EO}}{I_{co}} = \frac{1}{1 \cdot 10^{-3}} = 1000\,\Omega = 1\text{k}\Omega$

A szükséges U_B *osztóponti feszültség:*

$$U_B = U_{EBO} + U_{EO} = 0{,}5 + 1 = 1{,}5\text{V}$$

A *bázisosztó árama* a feladatban előírt 10-es szorzóval:

$$I_o \geq 10 \cdot I_{Bo} \approx 10 \cdot 10 = 100\,\mu\text{A}$$

Az R_a *felső osztóellenállás:* a 12-18. ábrából felírva:

$$R_a = \frac{U_+ - U_B}{I_o} = \frac{12 - 1{,}5}{100 \cdot 10^{-6}} = 10{,}5 \cdot 10^4 = 105\,\text{k}\Omega$$

Az R_b *alsó osztóellenállás,* ha az osztó alsó ágának árama az ábrából felírva I_o–I_{Bo}, akkor:

$$R_b = \frac{U_B}{I_o - I_{Bo}} = \frac{1{,}5}{(100 - 10) \cdot 10^{-6}} = 16{,}5 \cdot 10^3 = 16{,}5\,\text{k}\Omega$$

A kollektorellenállás meghatározásánál jó lenne ismerni a meghajtott fokozat bemeneti terhelését. Mivel ez nincs megadva, ezért csak a kivezérlési szempontokat vesszük figyelembe, ennek alapján előnyös, ha

$$U_{co} = \frac{U_+}{2} = \frac{12}{2} = 6\text{V}$$

Ezt felhasználva, és az ábrából felírva az R_c *kollektorellenállás:*

$$R_c = \frac{U_t - U_o}{I_{co}} = \frac{12 - 6}{1 \cdot 10^{-3}} = 6 \cdot 10^3 = 6\,\text{k}\Omega$$

A kapcsolatos alkatrész paramétereket ezzel mind felírtuk. Következhetne a kapott ellenállásértékek kerekítése a legközelebbi szabványos táblázat-beli értékre.
Végezetül megadjuk a kollektor munkaponti feszültségének *drift*-jét, melyet, mint tudjuk, az R_E visszacsatoló-ellenállás befolyásol:

$$\frac{\Delta U}{\Delta T} = \left(-2\frac{\text{mV}}{\text{°K}}\right)\cdot\frac{R_c}{R_E} = -2\cdot\frac{6}{1} = -12\frac{\text{mV}}{\text{°K}}$$

F.3.8. A példa megadott adataival és a 3.6.3.3. pontban bevezetett összefüggésekkel:
A *bemenőellenállás:*

$$r_{Be} \approx \beta \cdot R_E = 300 \cdot 3 \cdot 10^3 = 900\,\text{k}\Omega$$

A *kimenőellenállás:*

$$r_{ki} = \left(\frac{1}{S} + \frac{Rg}{\beta}\right) \times R_E$$

mivel 3.26 értelmében:

$$S = \frac{I_c}{U_T} = \frac{2\,\text{mA}}{26\,\text{mV}}$$

Ezért:

$$rki = \left(\frac{26\cdot10^{-3}}{2\cdot10^{-3}} + \frac{40\cdot10^3}{300}\right)\times 3\cdot10^3 = 140\,\Omega$$

F.3.9. A példában megadott kiindulóadatok mellé vegyük még fel a *drain áramot* $I_D = 3$ mA-re.
A (3.48) összefüggést tegyük explicitté U_{GS}-re:

$$U_{GS} = U_P\left(1 - \sqrt{\frac{I_D}{I_{DS}}}\right)$$

Ide behelyettesítve az eddigieket kapjuk:

$$U_{GS} = -3\left(1 - \sqrt{\frac{3\cdot10^{-3}}{10\cdot10^{-3}}}\right) = -1{,}36\,\text{V}$$

Ez teljesíti a 3.7.6.1. pontbeli feltételt, miszerint:

$$0 < |U_{GS}| < |U_p|$$

Innen az R_S *source ellenállás* az ábra alapján:

$$R_S = \frac{U_{GS}}{I_D} = \frac{1{,}36}{3 \cdot 10^{-3}} = 450\,\Omega$$

A munkaponti U_{Do} *drainfeszültséget* úgy kell felvenni, hogy a maximális $\Delta U_{D\max}$ kivezérlésnél se kerüljön U_{DS} az U_k küszöbfeszültség alá, ezt felírva:

$$U_{Do} > U_S + |\Delta U_{D\max}| + U_k$$

Felvéve: $\Delta U_{D\max} = \pm 2$ V-ot; a 3.7.3. pont alapján: $U_K = U_{GS} - U_p$ továbbá kb. 2 V ráhagyást véve biztonsági tartalékként, adódik:

$$U_{Do} \approx 7\,\text{V}$$

Az ábrából felírható az R_D *drain ellenállás* a választott $U_+ = +15$V tápfeszültség, a munkaponti drain feszültség és a drain áram behelyettesítésével:

$$R_D = \frac{U_+ - U_{Do}}{I_D} = \frac{15 - 7}{3 \cdot 10^{-3}} = 2{,}7\,\text{k}\Omega$$

A munkaponti meredekség (3.50) szerint:

$$S = \frac{2}{U_p}\sqrt{I_{DS} \cdot I_D} = \frac{2}{3}\sqrt{(10 \cdot 10^{-3}) \cdot (3 \cdot 10^{-3})} = 3{,}7\,\text{mA / V}$$

A közelítő feszültségerősítés $r_{DS} > R_D$ feltételezésével, az alsó határfrekvencia környezetében:

$$A \approx -S \cdot R_D = (-3{,}7) \cdot 2{,}7 \approx -10$$

A 12-19. ábra-beli kapcsolásnál a három kapacitás három felüláteresztő szűrőt képez, hasonlóan a 3.6.3.1. pontbeli kapcsoláshoz.
Esetünkben szűrőnként $f_a = 20$ Hz-es alsó eredő határfrekvenciát feltételezve, az egyes szűrők fai határfrekvenciája ($n = 3$ esetén):

$$f_{ai} = \frac{f_a}{\sqrt{n}} = \frac{20}{\sqrt{3}} \approx 10\,\text{Hz}$$

Ezzel az értékkel kiszámíthatjuk például a drain körbeli C_S kapacitás értékét, mely a váltakozó komponensekre nézve rövidre zárja az R_S ellenállást (lásd 3-55. ábrával kapcsolatos formulák):

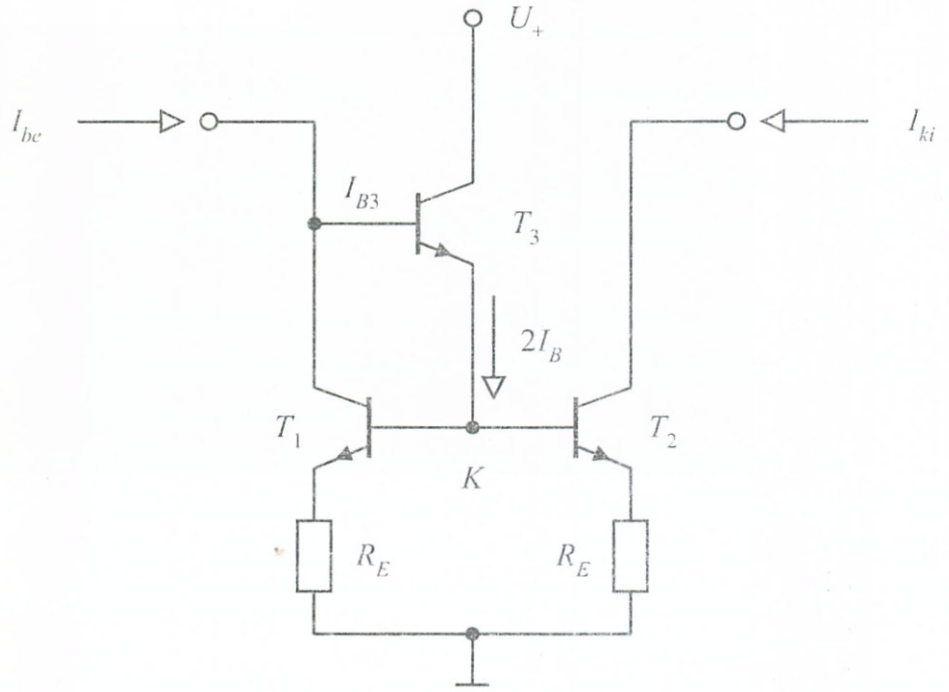
$$C_s = \frac{S}{2\pi fai} = \frac{3{,}7}{2\pi \cdot 10} \approx 51\,\mu\text{F}$$

F.3.10. Mint a feladat is említi, a zavarjelek kialakulásában fő bűnös a jeltorzulás, mely a színusz-jelek „felszabdalásából" keletkezik.

A helyzet lényegesen javul, ha épen hagyjuk a félhullámokat, de nem gyújtjuk őket be minden egyes periódusban, hanem csak meghatározott kihagyásonként. Ilyenkor a szűrő-integráló tagokon átvezetett jelek területének adott időtartamra eső összegéből képzett kimenő jel ugyancsak arányos lesz a φ szöggel, de az összetevők mindegyike - igaz kihagyásokkal - egy-egy szabályos félhullám. Ez az ún. *félhullámvezérlés* sokkal kevesebb felharmonikust tartalmaz, így lényegesen kisebb a zavarösszetevő mértéke.

12.4.1. Gyakorló feladatok a 4. „Jellegzetes elektronikus áramkörök" c. fejezethez

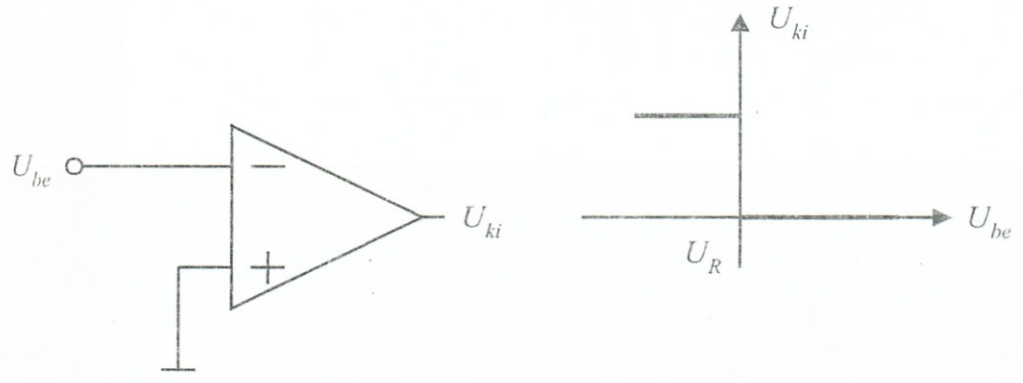
*G.4.1. Vizsgáljuk meg a 12-22. ábrán látható kapcsolás működését.



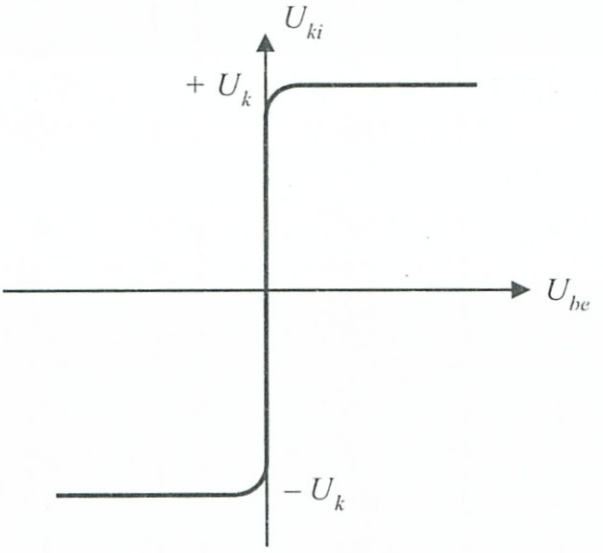
12–22. ábra *Vizsgált kapcsolás*

*G.4.2. Vizsgáljuk meg, hogy a 4-14. ábra ofszet-feszültségkiegyenlítési módozatain túlmenően kínálkozik-e egyéb lehetőség is.

*G.4.3. Miként módosítható a 12-23. ábrán látható *analóg komparátor*nál az eredetileg 0-értékű komparálási szint. A transz-



12–23. ábra *Analog komparátor*



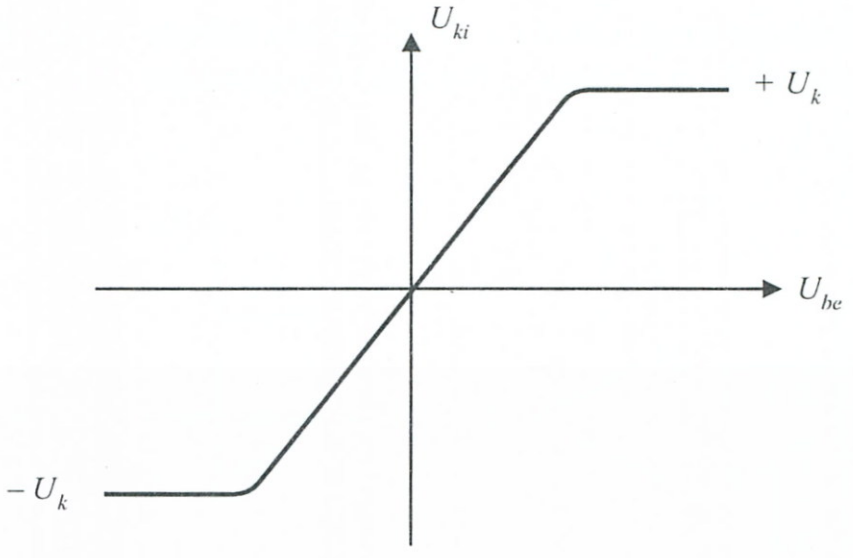
12–24. ábra *Ugrási transzfer karakterisztika*

fer karakterisztikát jó közelítéssel ugrásfüggvénynek tekinthetjük.

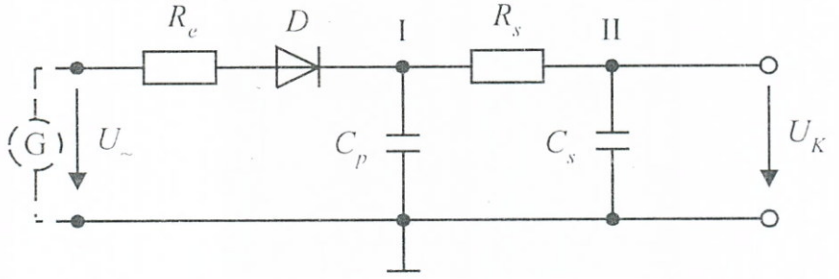
*G.4.4. Miként állítható elő a 12-24. ábra szerinti transzfer karakterisztika műveleti erősítővel?

*G.4.5. Állítsunk elő műveleti erősítővel a 12-25. ábra szerinti transzfer karakterisztikát.

*G.4.6. Állítsunk elő műveleti erősítővel logaritmikus transzfer karakterisztikát.



12–25. ábra *Ferde átmeneti transzfer karakterisztika*



12–26. ábra *Szűrőtagos egyenirányító*

*G.4.7. Rajzoljuk fel a 12-26. ábra *szűrőtaggal* kiegészített egyutas, hálózati egyenirányítójának jelalakjait az I. és II. pontokon.

*G.4.8. Rajzoljunk fel áramköri megoldásokat olyan esetekre, amikor mechanikus kontaktusok jelét kell elektronikus áramkörre csatlakoztatni.

*G.4.9. Alakítsunk ki normál kapukból és alkatrészekből egy beállítható küszöbszintekkel rendelkező, engedélyező bemenettel rendelkező Schmitt triggert.

*G.4.10. Alakítsunk ki olyan impulzus adót Schmitt trigger segítségével, mely átkapcsolhatóan az egyik üzemmódban *lefutó* jelhomlokra „1" impulzust, a másik üzemmódban *felfutó* jelhomlokra „0" impulzust állít elő.

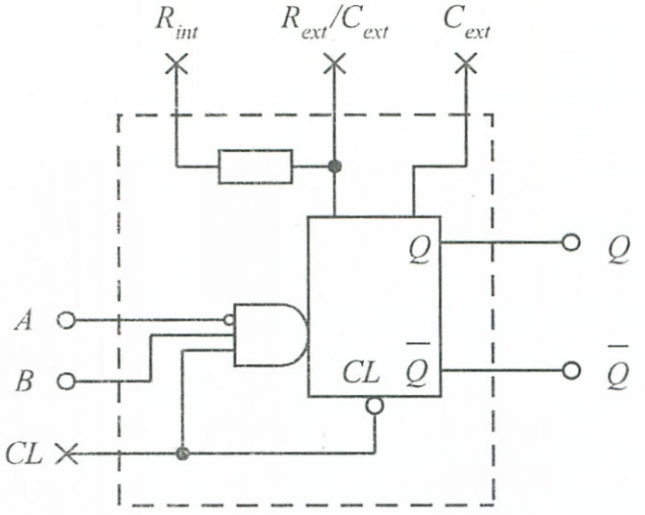
*G.4.11. Vizsgáljuk meg a 12-27. ábrán látható 123 típusú monostabil multivibrátor működését.

*G.4.12. Alakítsunk ki NOR kapukkal monostabil multivibrátort.

*G.4.13. A 12-28. ábrán egy 121 típusú hangolható monostabil multivibrátor katalógusrajza látható. Vizsgáljuk meg, miként működtethető az áramkör, melynek, mint látható, két negatív és egy pozitív élre indító bemenete is van. A bemenetek Schmitt trigger közbeiktatásával fejtik ki hatásukat.

*G.4.14. Adva a 12-29. ábrán látható Schmitt triggerrel kombinált NAND kapus *IC*. Alakítsunk ki segítségével *RC* oszcillátort.

*G.4.15. Alakítsunk ki *IC* inverterekkel kvarcvezérlésű *RC* oszcillátort.



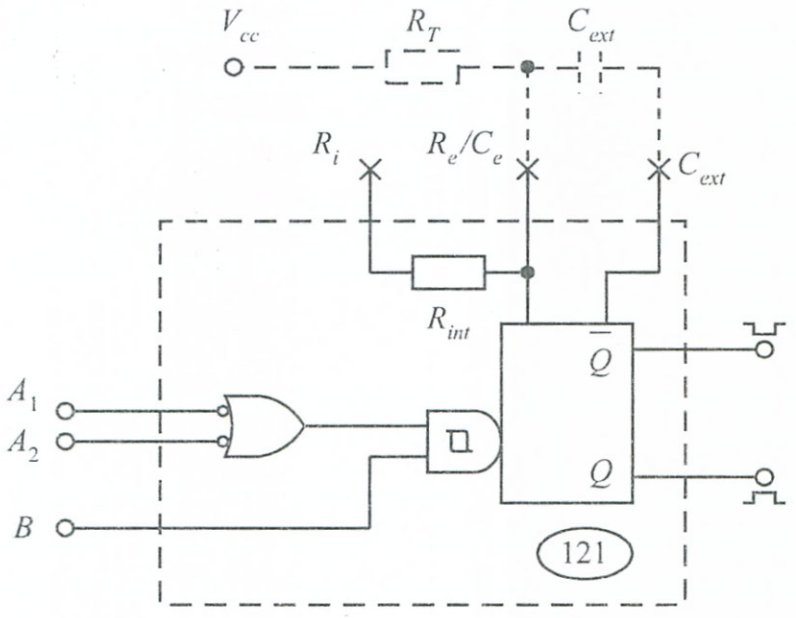
a)

CL	A	B	Q	$\overline{Q}$
L	X	X	L	H
X	H	X	L	H
X	X	L	L	H
H	L	↑	⎍	⎍
H	↓	H	⎍	⎍
↑	L	H	⎍	⎍

↑ = felugrás
↓ = leugrás
X = közömbös

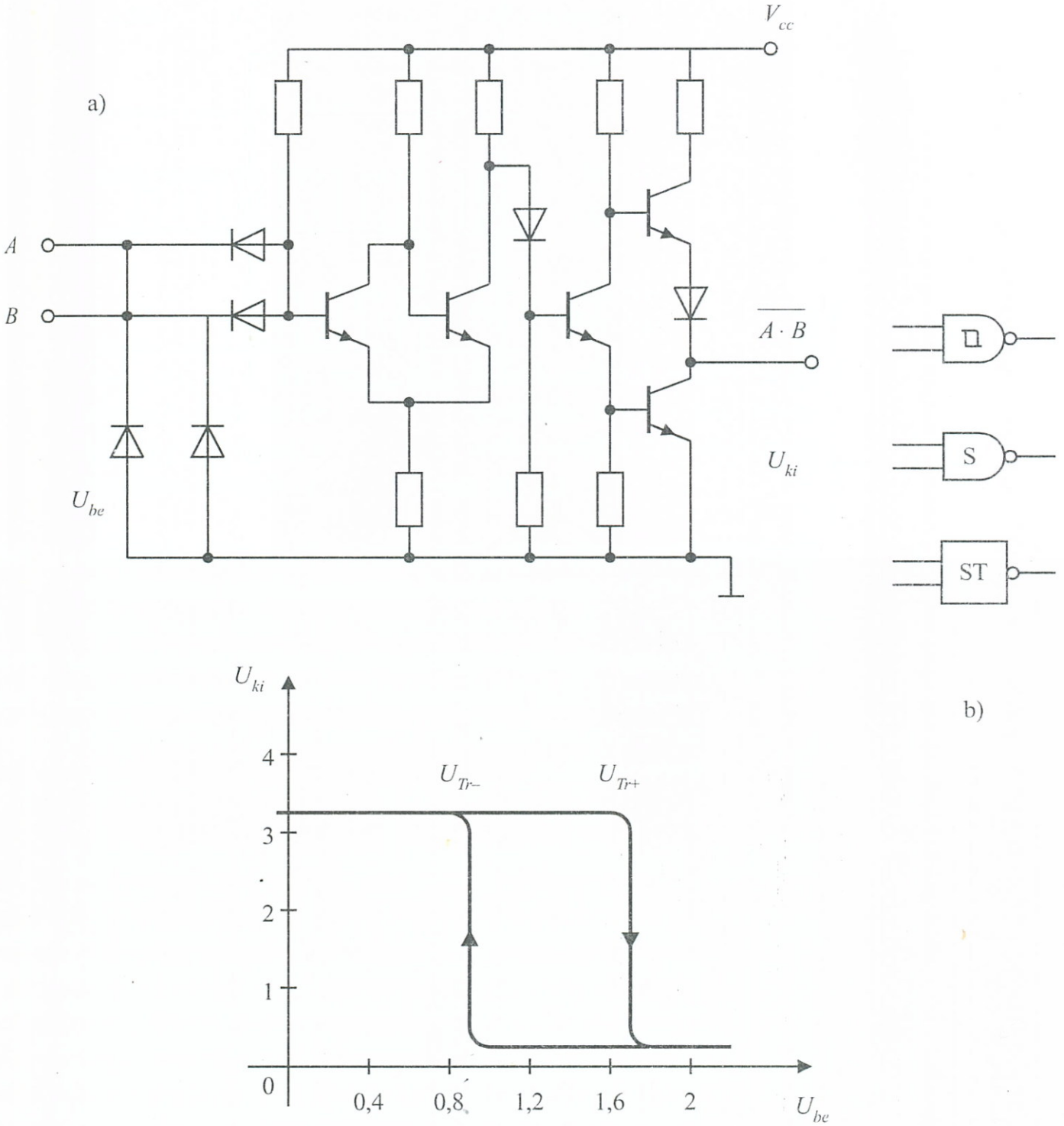
b)

12–27. ábra *Monostabil IC*



12–28. ábra *Schmitt triggeres monostabil multivibrátor*

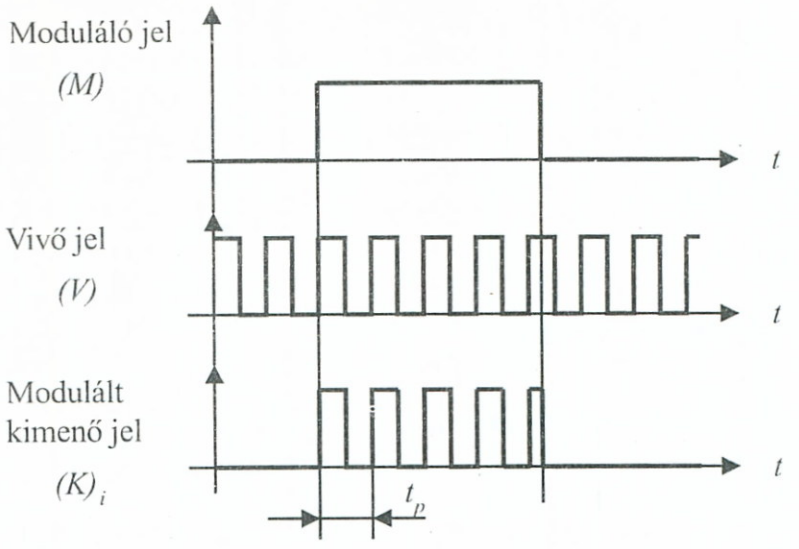
*G.4.16. Alakítsunk ki CMOS inverterekkel felépített *RC* oszcillátort, melynek frekvenciája kapcsolókkal lépcsőkben hangolható.



12–29. ábra *NAND kapu Schmitt triggerrel*

*G.4.17. Állítsunk össze impulzus-sorozat frekvenciát megkettőző kapcsolást, mely 0, illetve1 impulzusokra átkapcsolható.

*G.4.18. Az előző példa kapcsolásából rakjunk össze frekvenciasok-szorozót.



12–30. ábra *Modulált jelek*

*G.4.19. Kíséreljük meg egy modulátor kapcsolás kialakítását, mely a 12-30. ábra szerinti idő-feltételekkel működik.

*G.4.20. Tételezzük fel, hogy az előző példa modulátora egy átvivő vonalra küldi jeleit, és a VEVŐ oldalon szükségessé válik az eredeti moduláló jel leválasztása. Alakítsunk ki egy ilyen demodulátort.

12.4.2. Feladatmegoldások a 4. fejezethez

F.4.1. A megvizsgálandó 12-22. ábra erősen hasonlít a 4. fejezet 4-8. ábrán megismert *áramtükör*höz. A lényeges különbség az, hogy a T_1 tranzisztor kollektorpontja itt nem közvetlenül csatlakozik a $T_1 - T_2$ közös K bázisponthoz, hanem a T_3 tranzisztor emitterkörén keresztül. Ez azonban az I_{be} áram számára lényegesen kisebb elágazó áramot jelent a K közös pont irányában.

A 4-8. ábra-beli áramtükörnél ez az áram:

$$I_{elágazó} = 2I_B$$

és, mint láttuk a kimenőáram:

$$I_{ki} = \frac{I_{be}}{1 + \frac{2}{\beta}}$$

Példánknál a T_3 áramerősítése miatt:

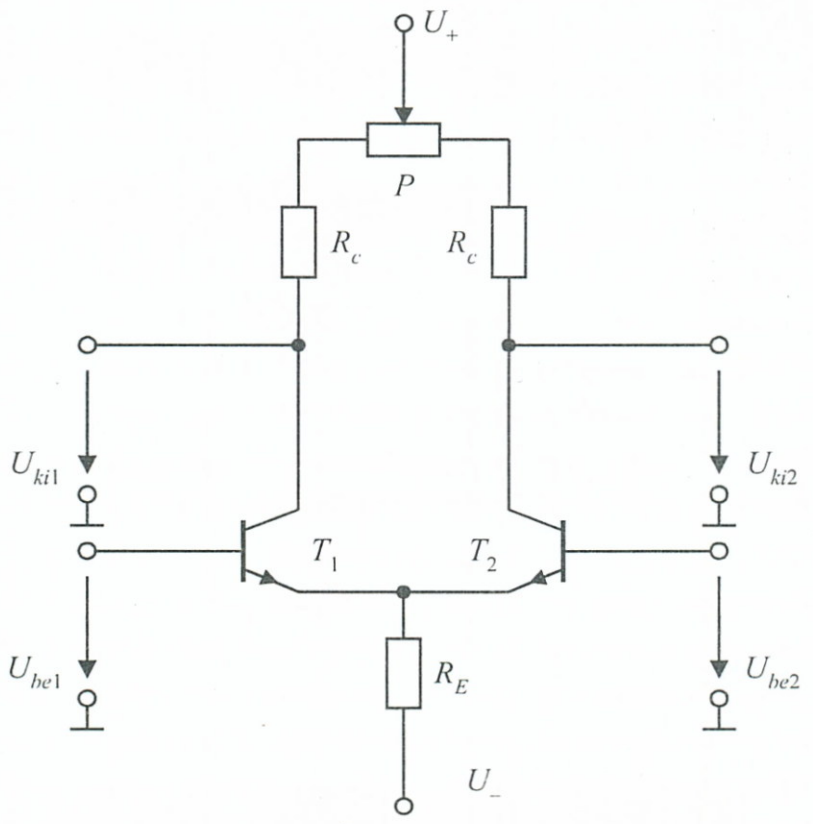
$$I_{elágazó} = \frac{2 \cdot I_B}{1+\beta}$$

ezzel viszont a kimenőáram:

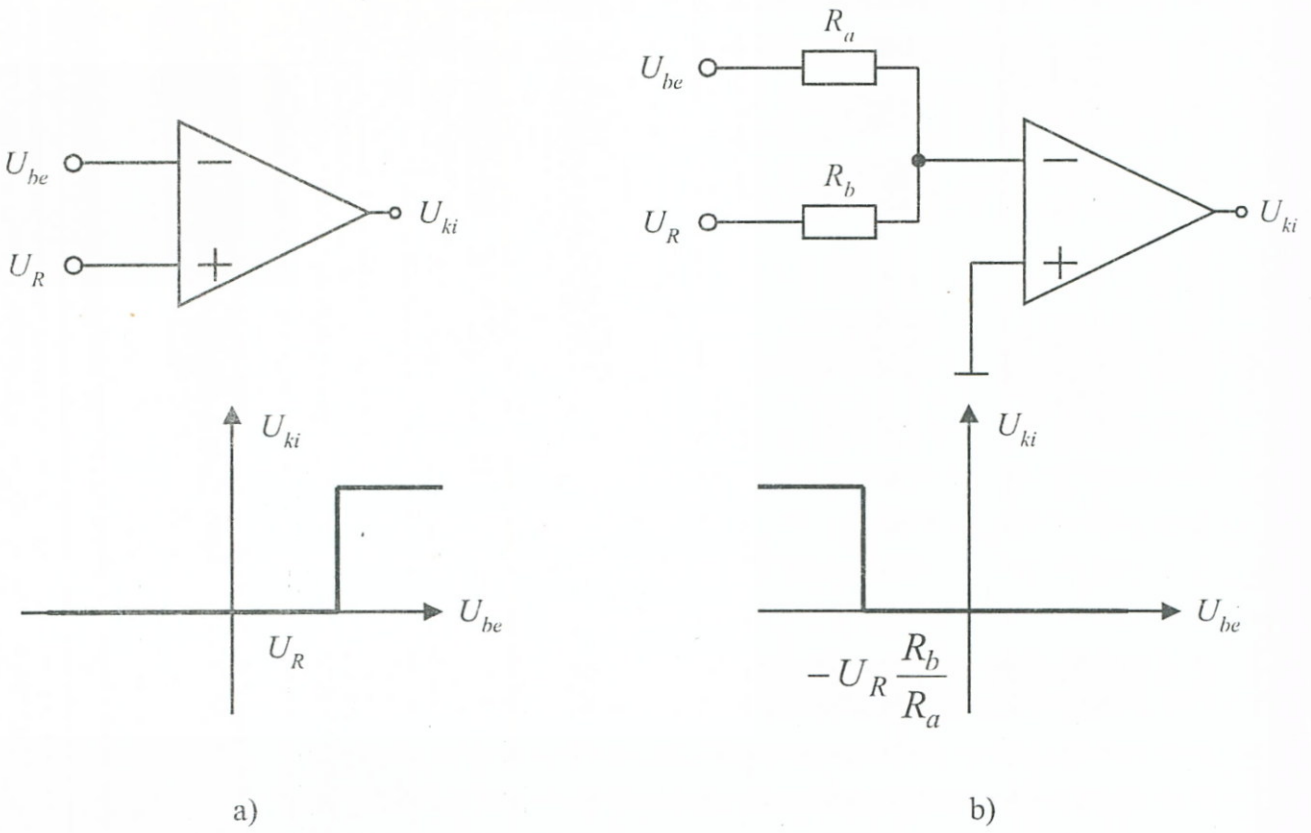
$$I_{ki} = \frac{I_{be}}{1+\frac{2}{\beta(1+\beta)}}$$

Vagyis I_{ki} és I_{be} eltérése sokkal kisebb, ami azt jelenti, hogy kapcsolásunk sokkal „jobb" áramtükör, mint a 4-8. ábrabeli. Az R_E ellenállások pedig tovább javítják a hőstabilitási paramétereket is.

F.4.2. A 4-14. ábra a., b. kapcsolásai részben az emitterkörbe, részben az egyik bemeneti körbe avatkoztak be. A 12-31. ábrán most felrajzolt kapcsolásunk a kollektor ellenállások értékének egymáshoz képesti eltolásával végzi el a korrekciót a P potenciométer segítségével.



12–31. ábra *Ofszetfeszültség korrekciós kapcsolás*



a) b)

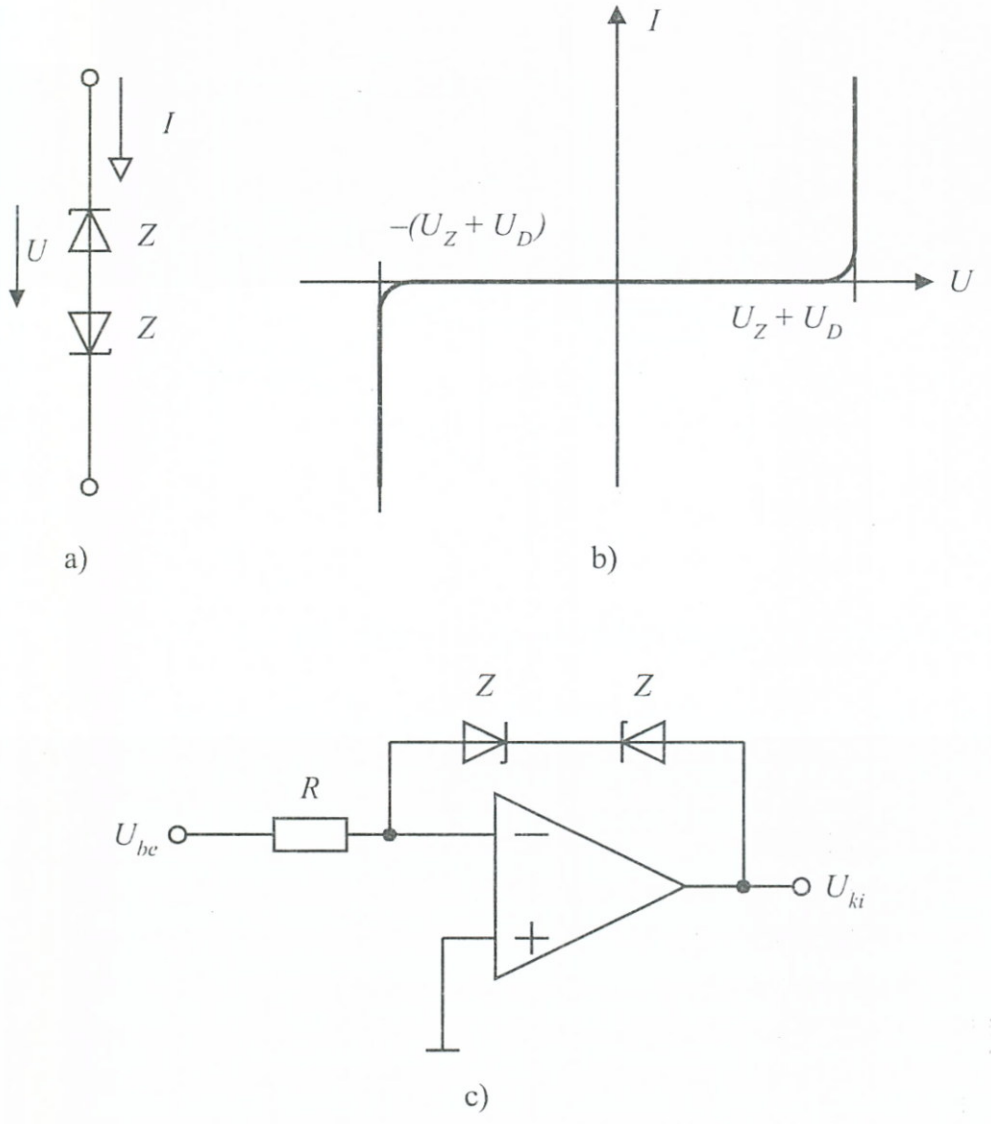
12–32. ábra *Komparálási szint eltolása*

F.4.3. A komparálási szintet, a műveleti erősítő mindkét bemenetét igénybevéve, az U_R referenciafeszültség beiktatásával módosíthatjuk.

a) Pozitív irányban a 12-32a. ábra szerinti kapcsolással
b) Negatív irányban a 12-32b. ábra szerinti ellenállásosztó beiktatásával.

F.4.4. A 12-24. ábrán látható transzfer karakterisztika egyik lehetséges megvalósítása a 12-33a. ábrán látható Zener-diódás osztón alapul. Itt az egyik dióda Zener üzemben dolgozik, a másik ugyanakkor nyitó irányban van, így az osztó karakterisztikája a 12-33b. ábra szerinti lesz, az U_Z letörési feszültséghez mindig hozzáadódik a nyitott dióda U_D feszültségejtése is.

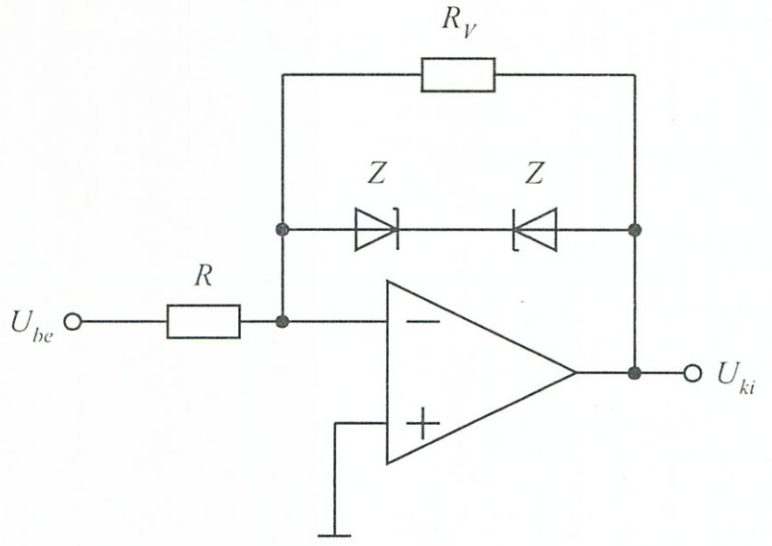
Amennyiben a Zeneres osztót egy műveleti erősítő visszacsatoló hurkába iktatjuk a 12-33c. ábra szerinti módon, akkor a kapcsolás kimenő karakterisztikája megegyezik az előállítani kívánt 12-24. ábrán megadott karakterisztikához.



12–33. ábra *Visszacsatolás Zener osztóval*

F.4.5. A 12-25. ábra karakterisztikája az előző példában kialakított kapcsolás továbbfejlesztésével megvalósítható. Ha a visszacsatoló hurokban a Zeneres osztót egy R_V ellenállással áthidaljuk a 12-34. ábra szerinti módon, akkor az új kapcsolás a kívánt karakterisztika szerint működik.

F.4.6. Már a 4. fejezet 4-24. ábráján felrajzoltunk egy exponenciális függvényt generáló kapcsolást. Hasonló úton haladva, a 12-35a. ábrán felrajzoltunk egy kapcsolást, mely a visszacsatoló ágban levő dióda exponenciális jellegű karakterisztikájának lényegében az inverzét képezi. Mivel a korábbiak alapján (3.18) ismert, hogy:



a)

12–34. ábra *Visszacsatolás áthidalt Zener osztóval*

$$I_D = I_S \cdot \exp\frac{U_D}{U_T} = I_S e^{\frac{U_D}{U_T}}$$

és a P pont virtuális föld, írható:

$$I = \frac{U_{be}}{R}$$

Az ideális műveleti erősítőre elmondottak alapján kimondható:

$$I = I_D$$

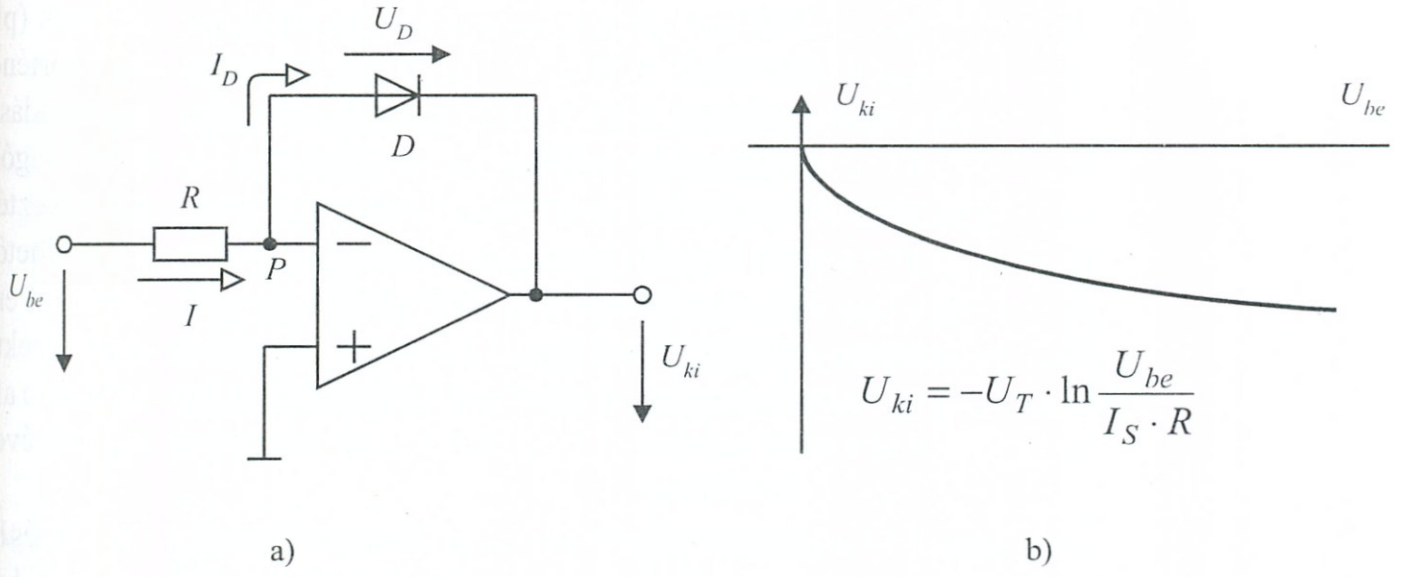
azaz írható, mivel $U_D = -U_{ki}$:

$$\frac{U_{be}}{R} = I_S \cdot e^{\frac{-U_{ki}}{U_T}}$$

innen U_{ki}-t explicitté téve kapjuk a logaritmusos kifejezést:

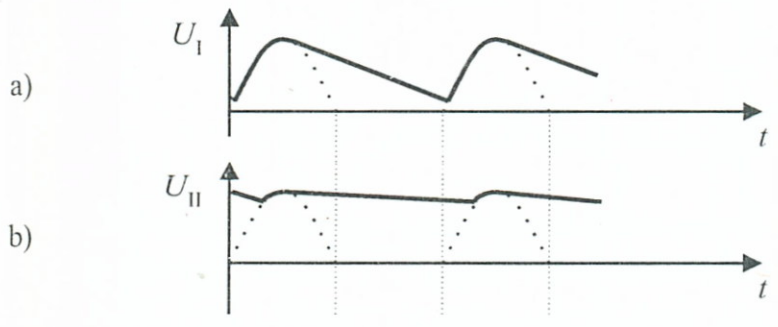
$$U_{ki} = -U_T \ln\frac{U_{be}}{R \cdot I_S}$$

A logaritmikus karakterisztikát a 12-35b. ábrán rajzoltuk fel.



12–35. ábra *Logarítmikus függvénygenerátorRV*

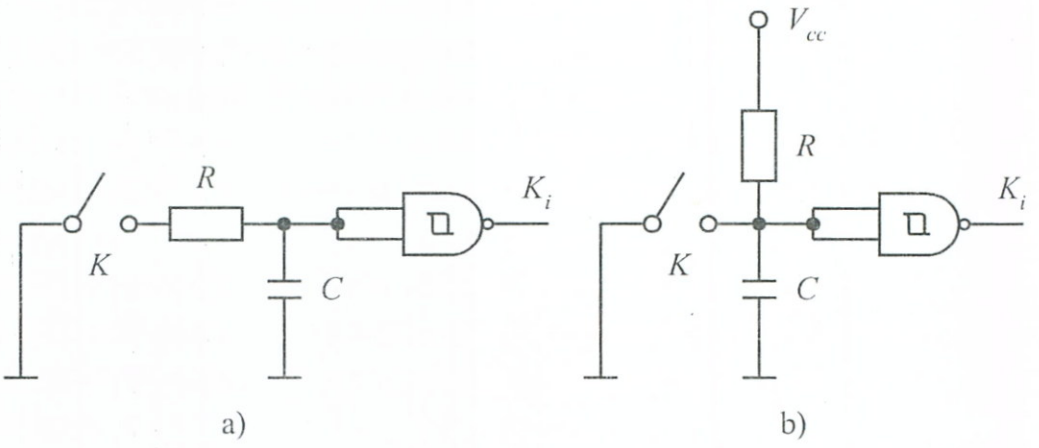
F.4.7. Az $R_S - C_S$ nélküli kapcsolás működési viszonyait a 4. fejezet 4.32. és 4-33. ábrái kapcsán már megvizsgáltuk. A 4-33. ábra $U(t)$ jelalakja megfelel az I. pont feszültségének. Az $R_S - C_S$ beiktatása valójában az I. pont és a földpont között egy váltóáramú feszültség-osztót eredményez, ahol a C_S sarkain mért II. kimeneti feszültség váltófeszültségű összetevője számára az X_{CS} reaktancia nagyon kis ellenállást jelent, ezért $U_{II\sim}$ lesöntölődik. Az egyenáramú összetevőre a leosztás nincs hatással, mivel számára a kondenzátor végtelen nagy ellenállásként mutatkozik. A kimenő jelalakra a 12-26. ábrán nem szereplő külső terhelés (R_T) van még hatással, ha R_T kisebb a szűrt jel hullámossága növekszik. A 12-36a. ábrán a kissé eltúlzott hullámosságú U_I-et, a b. ábrán a szűrt kimeneti $U_{II} = U_k$ jelet rajzoltuk fel. A $C_P - R_S - C_S$ együttes nélkül az I. ponton színusz pozitív félhullámok lennének mérhetők.



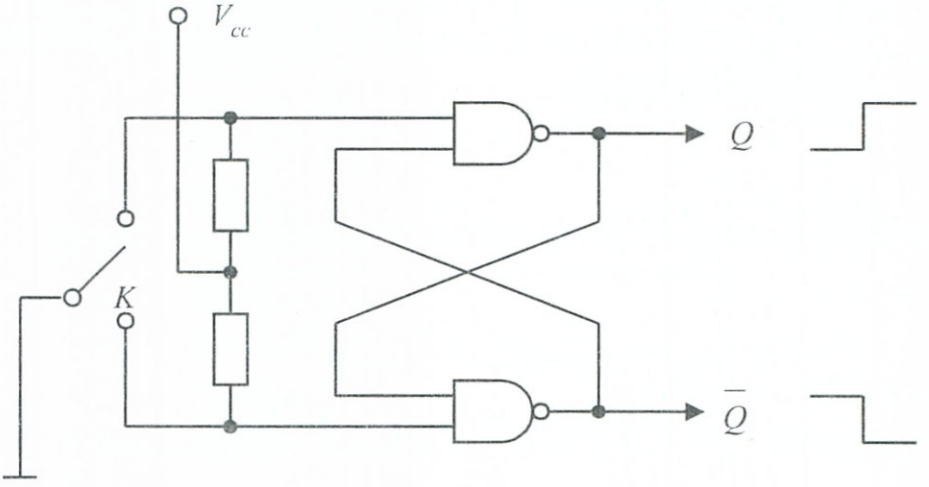
12–36. ábra *Szűrés hatása egyenirányításnál*

F.4.8. Az elektromechanikus kapcsolókat tartalmazó elemek (pl. nyomógombok, relék) elektronikus rendszerekhez történő csatlakozásakor zavart okozhat az érintkezők összezáródása folyamán egymásról többször visszapattanó érintkező rugók okozta, ún. pergési jelenség. A pergési jelenség következtében a várt feszültség-ugrás helyett egy nehezen kezelhető, rossz jelalak-jellemzőkkel bíró rövid impulzus-sorozat érkezne az elektromechanikus kapcsolóhoz csatlakozó elektronikus hálózatra. A jelenség kiküszöbölésének gyakran alkalmazott módja a zavarjeleknek kondenzátor segítségével történő kiszűrése.

RC aluláteresztő-szűrős megoldást mutatnak a 12-37a. és b. ábrák kapcsolásai, melyeknél a *C* kapacitás jel „elkenő" hatását ellensúlyozandó egy Schmitt triggerrel alakíthatjuk megfelelő meredekségűvé a kimenő jelet.



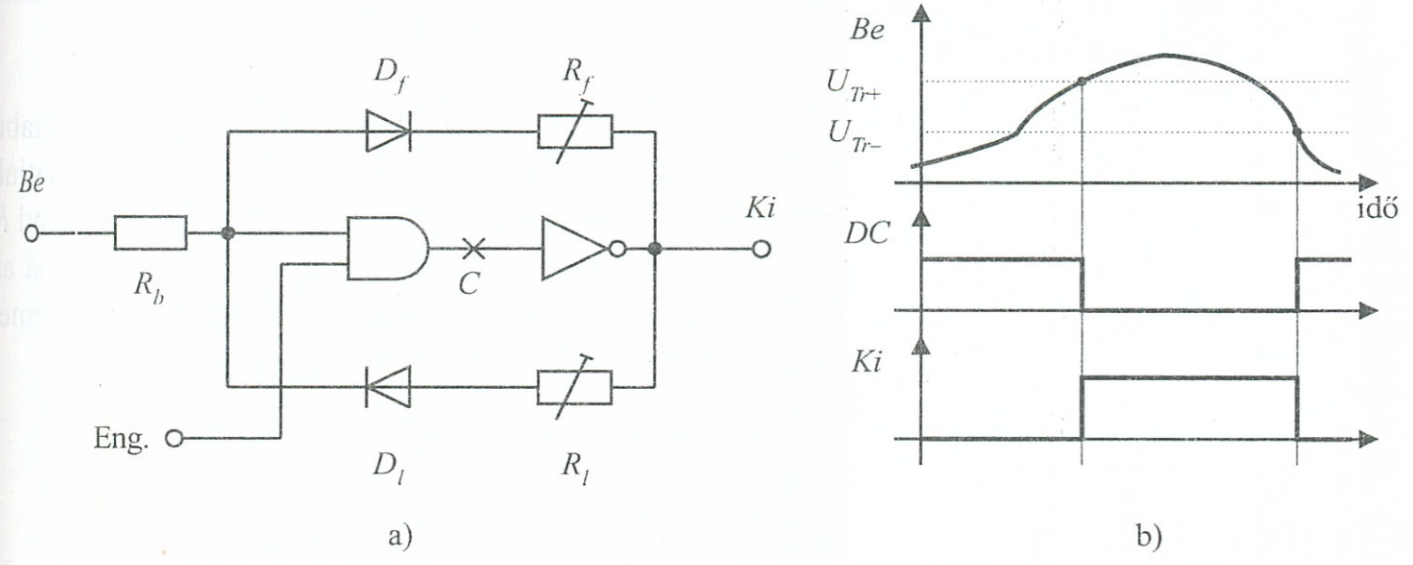
12–37. ábra *Pergés-védelem Schmitt trigerrel*



12–38. ábra *Pergés-védelem S–R tárolóval*

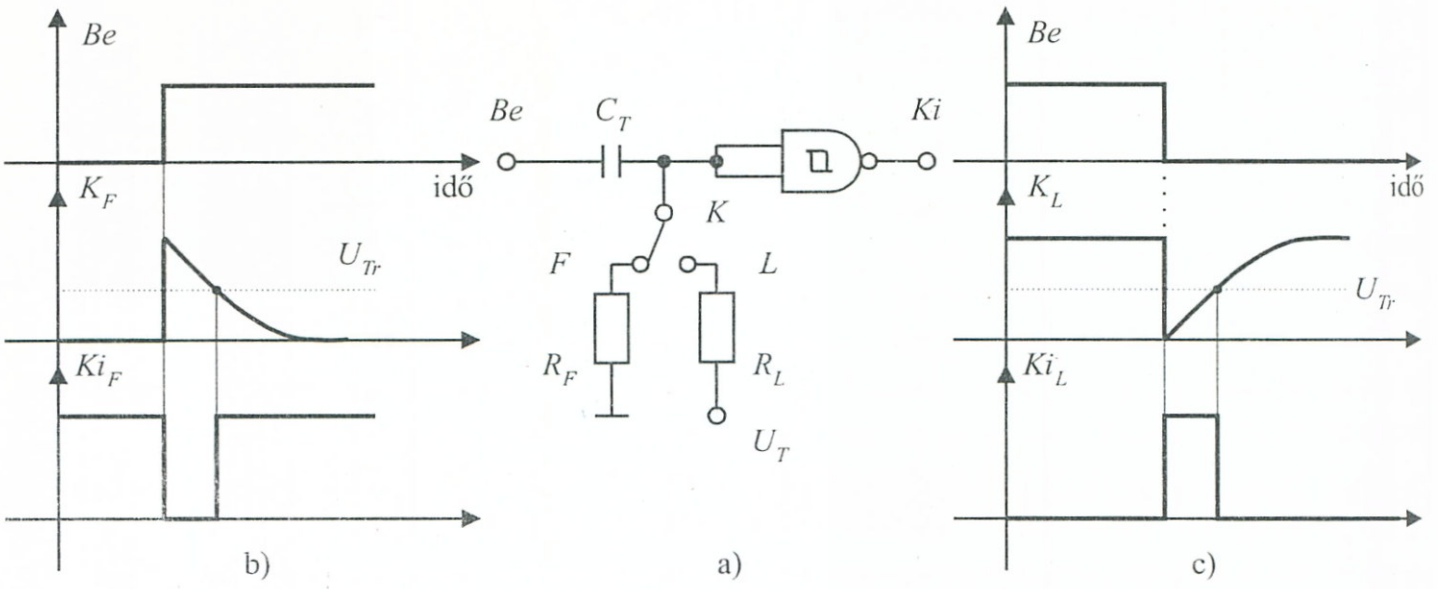
S-R tárolós megoldás látható a 12-38. ábrán, melyet *morze érintkező* esetén alkalmazhatuk. Átkapcsoláskor az első érintkezési impulzus a tárolóelemet a megfelelő oldalra átbillenti és a következő pergési jelsorozat már nem változtatja meg a tároló állapotát, így a tárolóelem átbillenési kimenő jele képviseli a mechanikus átváltást. Ellenkező irányú átkapcsoláskor a folyamat ellenkező értelemben zajlik le. Miután ennél a megoldásnál nincs kondenzátor töltési/kisülési időállandó, ezért a működési sebesség is kedvezőbb lesz.

F.4.9. A G.4.9. feladatnak megfelelő kapcsolás és a hozzátartozó idődiagramok a 12-39. ábrán láthatók. Az engedélyezést az első fokozat NAND kapujának második bemenete teszi lehetővé. (Eng.) A pozitív visszacsatolás itt nem emitterköri, hanem az *R-D* ellenállás-dióda ágak valamelyikén keresztül valósul meg, attól függően, hogy a D_f, ill. D_l diódák közül, melyik kerül vezető állapotba.
Ki pont feszültségszintjének hatására. Az R_f ellenállással a felfutási U_{Tr+}, az R_l ellenállással a lefutási U_{Tr-} küszöbszintek állíthatók be.



12–39. ábra *Állítható küszöbü Schmitt trigger*

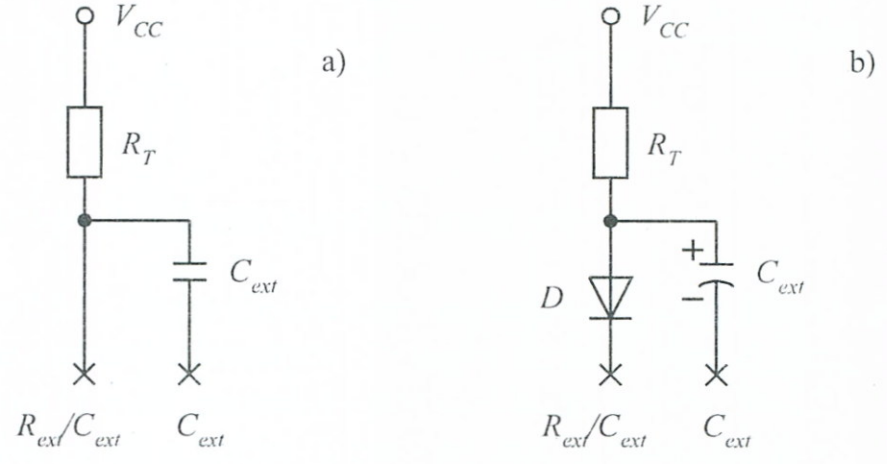
F.4.10. A feladat egy lehetséges megoldását egy *R-C* differenciáló tag és egy Schmitt trigger kombinációja szolgáltathatja a 12-40. ábra szerinti összekapcsolásban. Ha a 12-40a. ábrán a *K* kapcsolót *F* állásba tesszük, akkor a Be jel *felfutó* jelhomlokára a differenciáló tag *K* oldali kimenetén a K_F vá-



12–40. ábra *Felfutásra és lefutásra kapcsolható impulzus adó*

laszjel keletkezik. Ez a Schmitt trigger U_{Tr} küszöbszint-értékét kétszer is átlépi, minek következtében a kimeneten a $Ki_F = 0$ impulzus jelenik meg. (12-40b. ábra)
A K kapcsoló L helyzetében az R_L ellenállásra U_T feszültség kapcsolódik, és a Be jel *lefutó* jelhomlokának hatására a 12-40c. ábra szerinti diagramok rajzolhatók fel.
A Ki_F és Ki_L impulzusok szélessége az R_F, ill. R_L ellenállásokkal állítható.

F.4.11. A 12-27a. ábrán látható kiinduló 123 típusú monostabil multivibrátor működését a 12-27b. táblázatból olvashatjuk le. Mint látható, a tervezők jóvoltából mind L (A), mind H szintű (B) bemenetek is rendelkezésre állnak, továbbá az áramkör rendelkezik egy CL (CLEAR) visszaállító beme-



12–41. ábra *Időzítőelemek csatlakoztatása a 7-27a áramkörhöz*

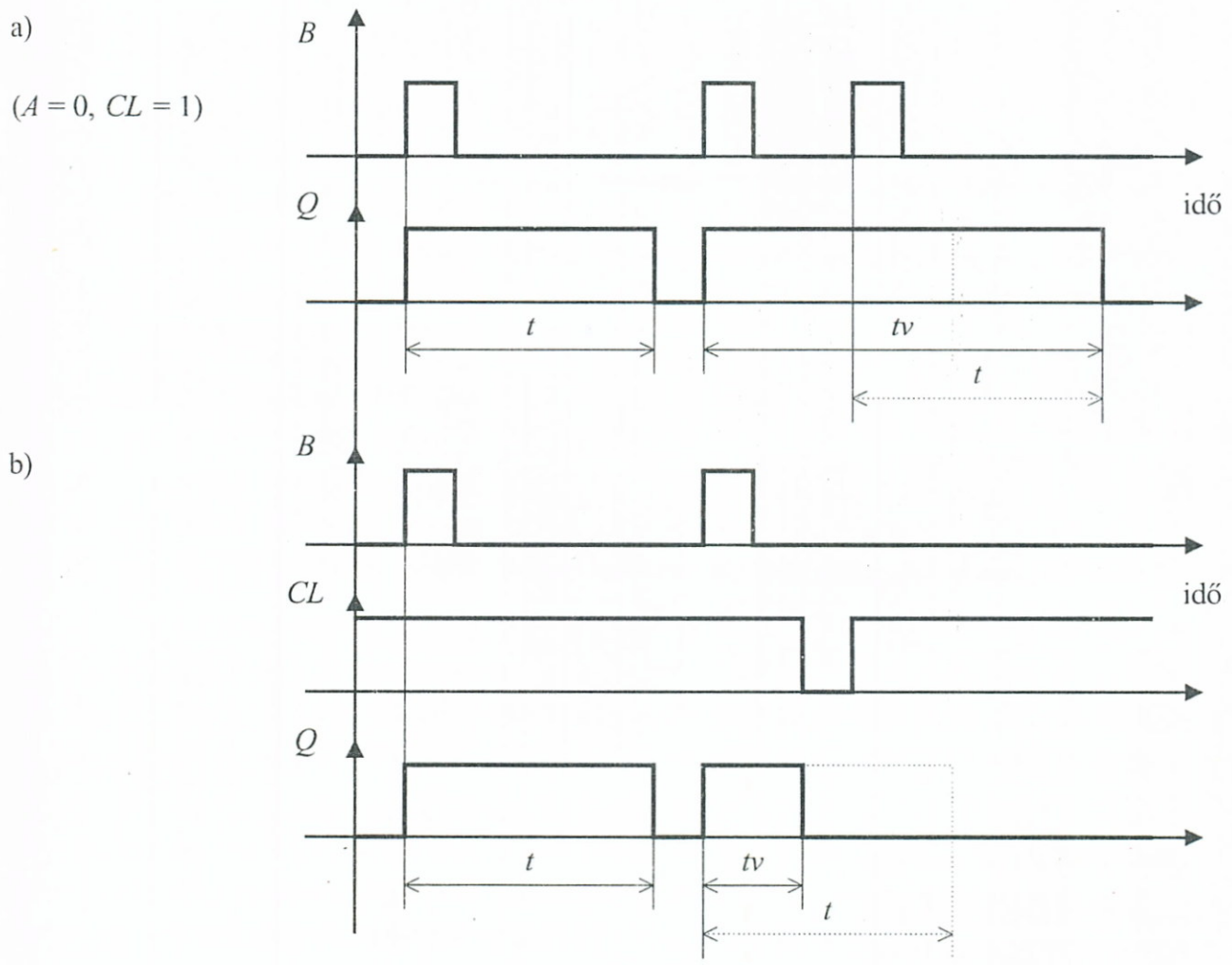
nettel, mellyel a pillanatnyilag futó kimeneti impulzus megszakíthatóvá válik. A *CL* bemenet aktivizálódásakor az A és B hatása letiltódik.

A kimeneti impulzusok (Q, $\overline{Q}$) szélességét a 12-41. ábra kapcsolásai segítségével lehet beállítani. Az a.) változat a szokványos esetekben, míg a b.) változat olyan esetekben kerül alkalmazásra, amikor C_{ext} elektrolit kondenzátor, ill. ha $C_{ext} \geq 1$ nF és a *CL* bemenet is felhasználásra kerül. Ha *CL*-t nem használjuk, akkor $CL = 1$ állandó állapotba kell állítani a többi bemenetek zavartalan működtethetősége érdekében.

A kimeneti impulzus szélességét a két esetre, a következő formulákkal számíthatjuk:

$$t_a = 0{,}32 C_e \, (R_t + 700 \text{ ohm})$$

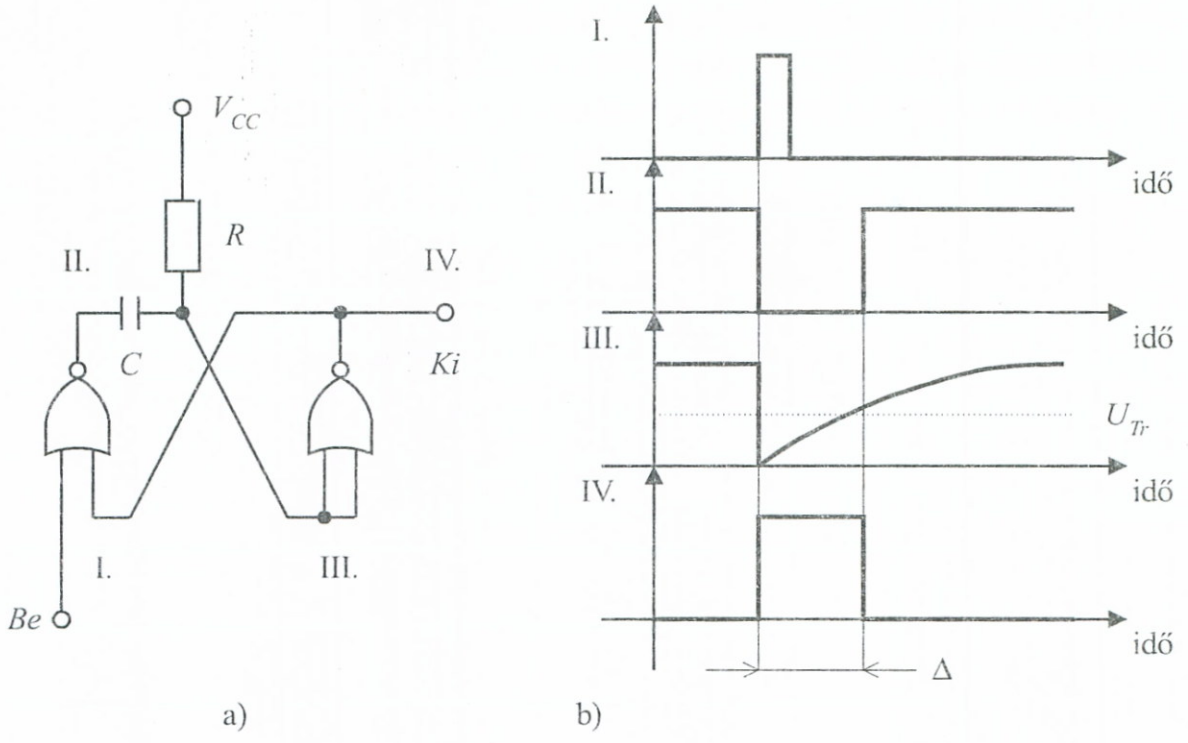
$$t_b = 0{,}28 C_e \, (R_t + 700 \text{ ohm})$$



12–42. ábra *Időviszonyok különféle vezérlésnél egy 123 típusú monostabil multivibrátornál*

A monostabil multivibrátor működésének időviszonyait mérlegelhetjük a 12-42. ábra alapján, ahol az első indító impulzus hatására mind az a.), mind a b.) ábra eseteiben a jel-futás zavartalan. A második indító impulzus után az:
a) esetnél *CL inaktív* és futó impulzus közben érkezett egy újabb indítás. A b.) esetnél futó impulzus közben egy *aktív CL* impulzus érkezik. Mint az ábrákból látható, a kimeneti impulzus időtartama mindkét esetben befolyásolódik.

F.4.12. A 12-43a. ábrán látható két NOR kapuból álló hálózatnál az I. bemeneti pontra egy „rövid" impulzust adva, ennek *felfutó* élére a 12-43b. I. idődiagramjának megfelelően a II., III., IV. pontokon észlelhető jelek a további megfelelő jelű idődiagramokon követhetők. Mint megállapítható, a kimeneti (IV.) impulzus szélessége az *R-C* elemek időállandójának és a kimeneti NOR kapu bemeneti komparációs szintjének (U_{Tr}) a függvénye.



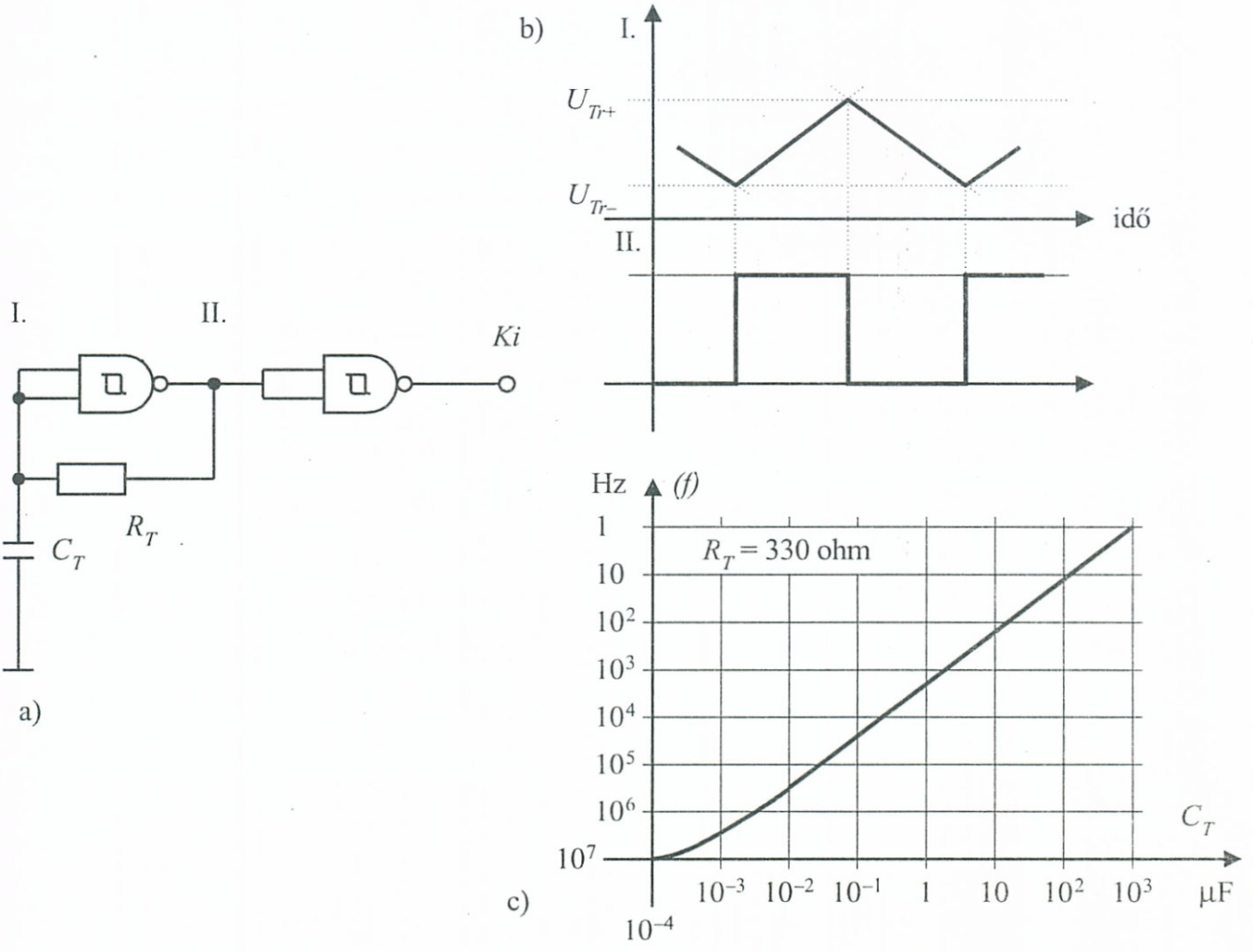
12–43. ábra *Monostabil multivibrátor működése*

F.4.13. A 12-28. ábra monostabil multivibrátoránál, mind a Q, mind a $\overline{Q}$ kimenetek hasznosíthatók. A kimeneti impulzusok szélességének beállítására külső hangoló-elemek (R_T, C_{ext}) csatlakoztathatók, melyekkel a szélességi időértékek: t_W = 40 nsec-től 28 sec-ig terjedő széles értéksávban beállíthatók.

Az áramkör egy R_{int} beépített ellenállással is rendelkezik, melyet (R_i csatlakozásának a V_{cc}-re kötésével) aktivizálva, 35 ns impulzus-szélesség jelenik meg a kimeneteken. A kívánt impulzus-szélességhez tartozó $R_T - C_{ext}$ értékek megválasztására a katalógusok nomogrammokat közölnek. A szélső értékek: C_{ext}-nél 10 pF-tól 10 uF-ig, ill. R_T-néál 2 kohm-tól 40 kohm-ig terjednek. Az impulzus-szélesség hozzávetőleges meghatározására a katalógus a következő formulát ajánlja:

$$t_w = C_{ext} \cdot R_T \cdot 0{,}7$$

F.4.14. A 12-29. ábra áramkörének felhasználásával a 12-44a. ábra szerinti *RC* oszcillátor állítható össze:
A periódikus jelsorozatot valójában az első fokozat már előállítja az R_T visszacsatoló ellenállás és az időzítést beállító C_T kapacitás révén a 12-44b. ábra idődiagramjának megfelelően. Ebből látható, hogy a II. pontbeli jel 1 állapot-

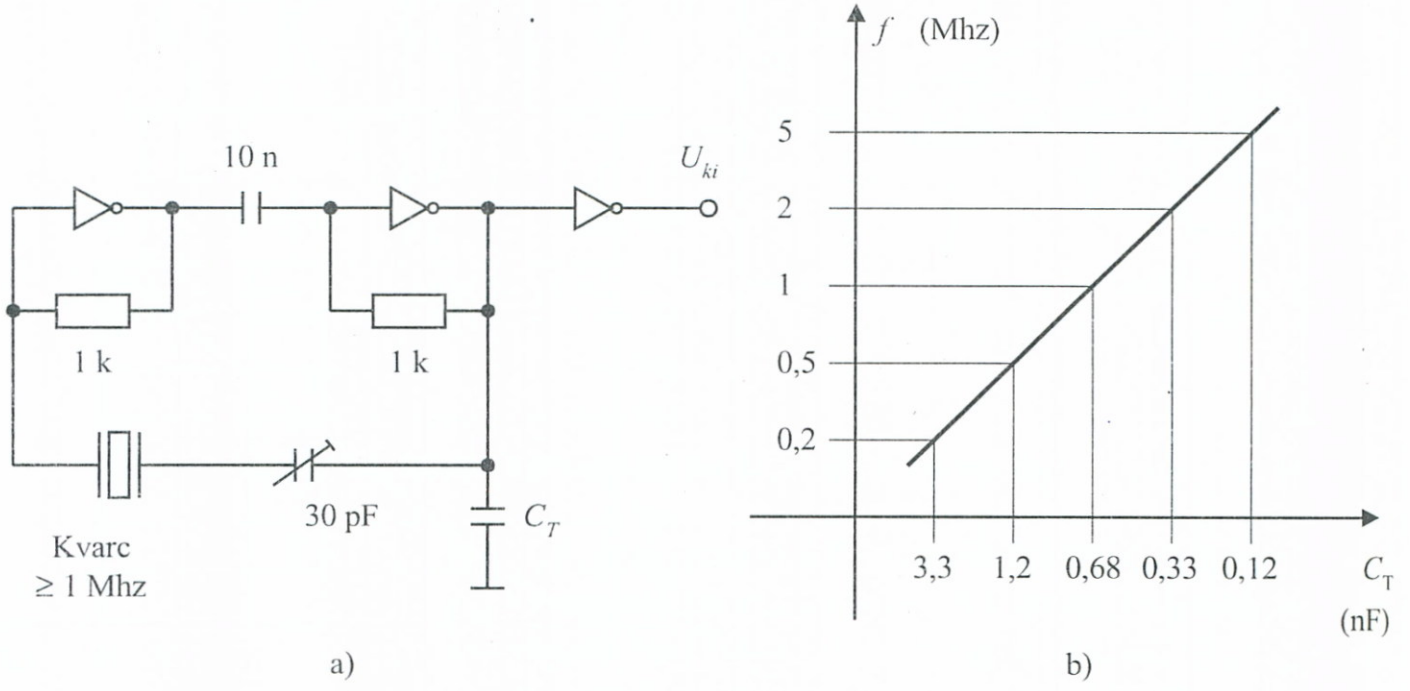


12–44. ábra *RC oszcillátor visszacsatolt Schmitt triggerrel*

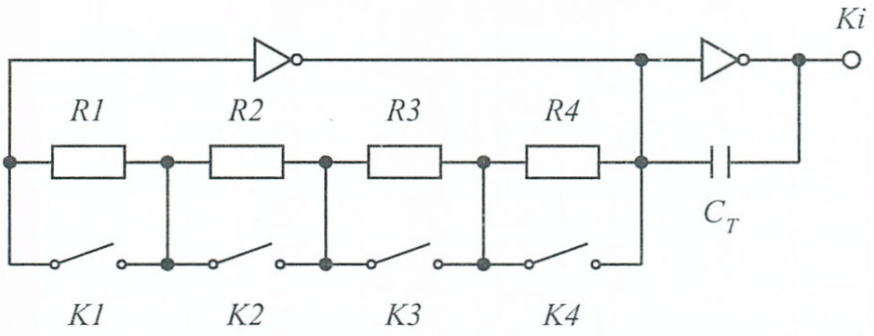
ba kerülését követően a C_T ondenzátor lassan feltöltődik és midőn ezáltal az I. pont feszültsége eléri a Schmitt trigger U_{Tr+} küszöbfeszültségét, az átbillen és a kimenet 0 állapotba kerül. Ezt követően a C_T kondenzátor kisülése indul meg, melynek hatására az I. pont feszültsége a Schmitt trigger U_{Tr-} küszöbfeszültsége felé csökken, majd ezt elérve, újabb átbillenés következik, és a folyamat ciklikusan folytatódik. A II. pontra csatlakozó második Schmitt triggeres NAND kapu valójában csak kimeneti jelformálásra és leválasztásra szolgál. A 12-44c. ábrán látható nomogramból a kívánt működési frekvenciához beforrasztandó C_T értékek olvashatók le.

F.4.15. A 12-45a. ábrán egy T^2L inverterekből felépített kvarc-vezérlésű oszcillátort láthatunk, melynél a kvarckristállyal sorosan elhelyezkedő kondenzátor hangoló trimmer szerepet lát el. A C_T kondenzátorral állítható kimeneti frekvencia alakulását a 12-45b. ábrán követhetjük.

F.4.16. A G.4.16. feladatnak megfelelően a 12-46. ábrán rajzoltunk fel egy visszacsatolt, CMOS invertereket tartalmazó *RC* oszcillátort. Itt a frekvenciát a C_T kondenzátorból és az $R_1...R_4$ ellenálláslánc megfelelő részéből álló R_T ellenállás időállandója határozza meg. Az ellenálláslánc egy adott R_i



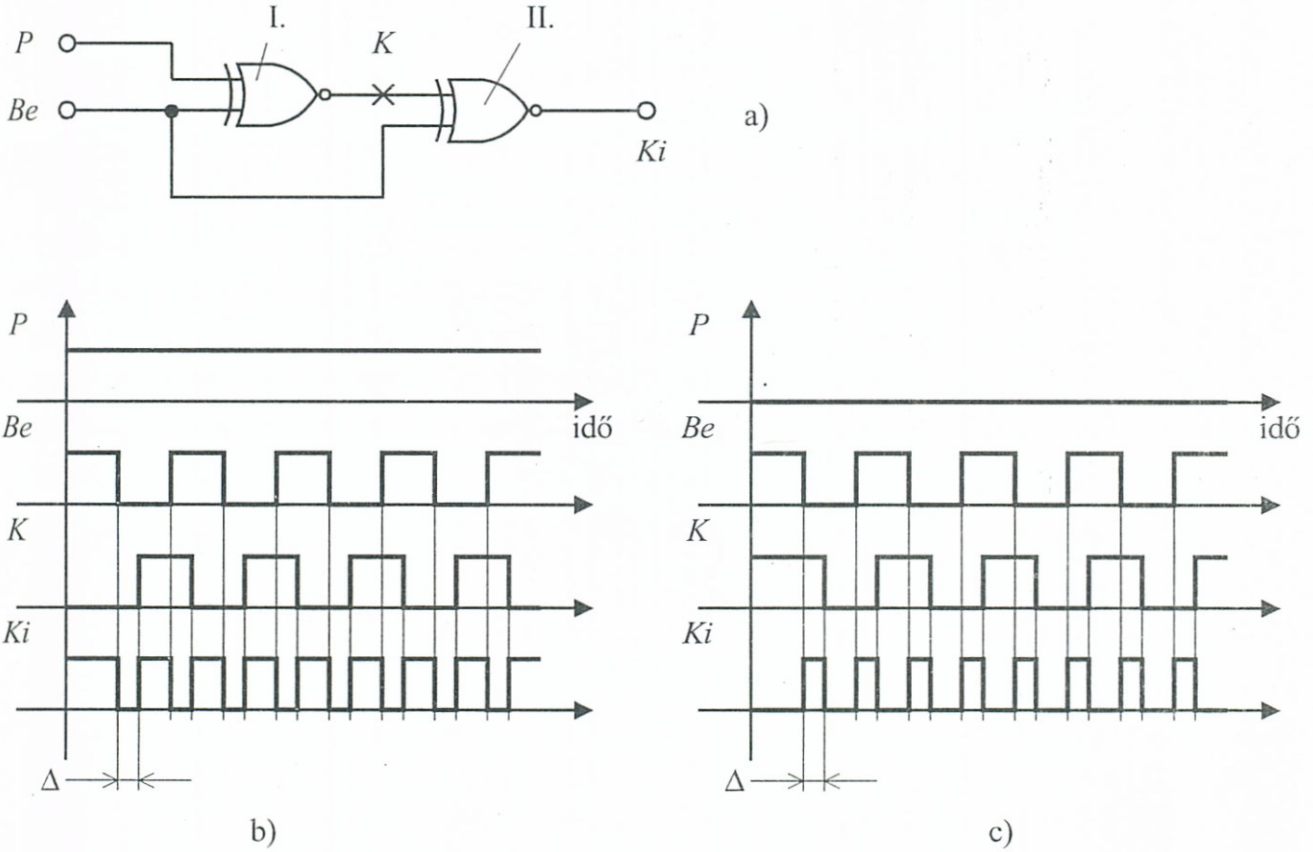
12–45. ábra *Kvarcvezérelt RC oszcillátor inverterrel*



12–46. ábra *Programozható CMOS oszcillátor*

elemének kiiktatása a *Ki* kapcsoló révén történhet, mely kapcsolók is lehetnek MOS elemek. A *Ki* kapcsolók kombinálása esetén a különféle frekvencia-féleségek felső határa 16.

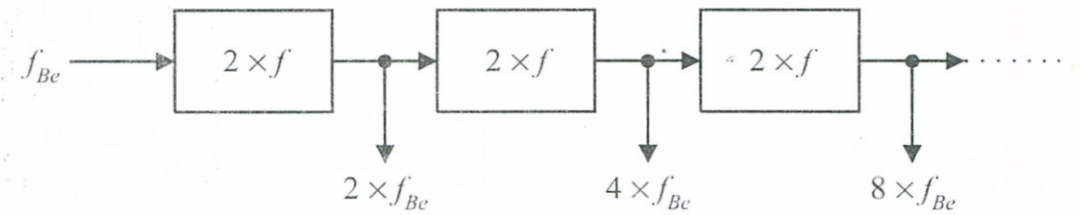
F.4.17. A 12-47a. ábrán antivalencia kapukból felépített kapcsolás látható. Az I. kapu *P* impulzus-polaritás vezérlő bemenetével lehet a kimeneti impulzus-sorozat polaritását befolyásolni, oly módon, hogy $P = 0$ esetén az I. kapu ISMÉTELŐ-ként, $P = 1$ esetén INVERTER-ként viselkedik a *Be* jelű pont je-



12–47. ábra *Polaritásvezérelt frekvenciakétszerező hálózat*

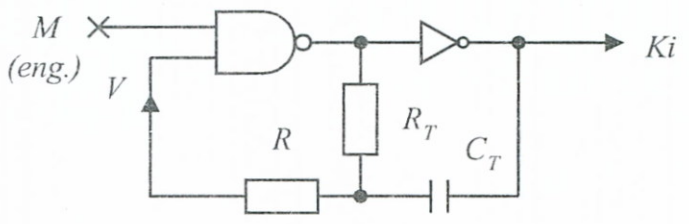
lei számára. Az áramköri rajz *K* pontjára érkező jelek az I. ANTIVALENCIA kapu késleltetésével eltolva jelennek meg és a II. ANTIVALENCIA kapunál találkoznak a *Be* jelű pont impulzusaival. Az eredő kimeneti jelek a II. kimenetén jelentkeznek.
A 12-47b. és c. ábrákon felrajzolt idődiagramok jól szemléltetik a $P = 1$, ill. $P = 0$ esetekre a *Ki* ponton megjelenő megkettőzött frekvenciájú impulzus-sorozat kialakulását.

F.4.18. A frekvencia sokszorozó kapcsolások kialakításának egyik egyszerű módja a kisebb szorzófaktorú egységek kaszkádosítása. Ilyen esetre utaló hálózatot rajzoltunk fel a 12-48. ábrán, ahol modulként pl. az előző példában bemutatott kétszerező kapcsolás is felhasználható. A kérdéses négyszerezés, ill. nyolcszorozás a megfelelő leágazások kivezetésével érhető el, de nincs akadálya további 2^n típusú többszörözésnek a láncolat bővítése révén.



12–48. ábra *Frekvencia többszörözés kétszerezők kaszkádosításával*

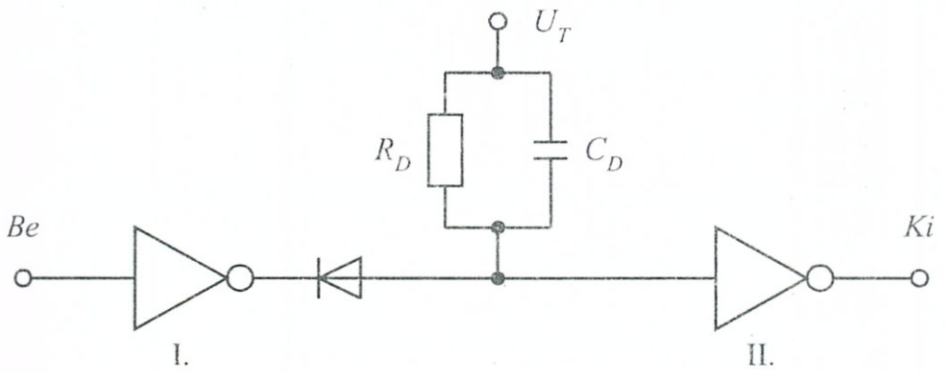
F.4.19. A 12-30. ábrán kiindulásként megadott tp periódusidejű modulált jelsorozat előállítását (többek között) megoldhatjuk egy indított oszcillátor segítségével. A 12-49. ábrán, egy engedélyező bemenettel ellátott $R_T C_T$ elemekkel hangolható oszcillátort rajzoltunk fel, melynél az áramkör megfelelő pontjain fellépő jeleket a kiinduló 12-30. ábra jeleivel azonos módon jelöltük. Mint látható, az „*M*" moduláló je-



12–49. ábra *Modulált R-C oszcillátor*

let az *RC* oszcillátor engedélyező pontjára vezetjük, mely ÉS bemenetként működik.

F.4.20. A kérdéses *demodulátor áramkör*-re rajzoltunk fel egy példát a 12-50. ábrán az irodalom alapján. Itt a bemenetre érkező jelsorozat „rövid" impulzusait az I. invertert követő $R_D C_D$ tag elnyeli, viszonylag nagy időállandója jóvoltából. Ha az eredeti M moduláló jelnél jelszünet ($M = 0$) jelentkezik, akkor - mivel ez mindig hosszabb ideig tart - az $R_D C_D$ tag időállandója ki tudja fejteni hatását, és elérvén a kimeneti II. inverter küszöbértékét, azt át tudja kapcsolni. Ily módon a *Ki* kimeneti ponton újra kialakulhat az eredeti *M* moduláló jel. A berendezés akkor működik használhatóan, ha az $R_D \cdot C_D > t_p$ feltétel teljesül. Itt t_p a 12-30. kiinduló-ábrán szereplő periódusidő.



12–50. ábra *Demodulátor*

12.5.1. Gyakorló feladatok az 5. „Digitális áramkörök" c. fejezethez

*G.5.1. Jellemezzük a feszültség és áramlogikák közötti összefüggést.

*G.5.2. Rajzoljunk fel egy antivalencia áramkört kapcsolókkal.

*G.5.3. Rajzoljuk fel az alábbi függvényt realizáló áramlogikás és feszültséglogikás kapcsolóhálózatot.

$$Y = A \cdot \overline{B} + (\overline{C} + D) \cdot \overline{A}$$

*G.5.4. Rajzoljunk fel egy Schottky-diódás VAGY-kaput és vizsgáljuk meg működését.

*G.5.5. Feltételezve, hogy a G.5.3. példabeli hálózat pozitív logikában működik, adjuk meg, hogy negatív logikában mit valósít meg.

*G.5.6. Rajzoljunk fel egy MOS-TRANSFER kaput tartalmazó áramkört.

*G.5.7. Rajzoljunk fel egy CMOS kapcsolást az

$$Y = \overline{A}(B + \overline{C})$$

függvény realizálására.

*G.5.8. Valósítsuk meg a G.5.3.-beli függvényt T^2L HUZALOZOTT (wired) kapcsolással.

*G.5.9. Rajzoljunk fel egy háromállapotú kimenettel rendelkező invertert, ahol a kimeneti állapot beállítása *nem soros* kapcsolóval történik.

*G.5.10. Rajzoljunk fel egy ECL kapcsolást és ismertessük működését.

*G.5.11. Rajzoljunk fel *dinamikus* MOS invertert és ismertessük működését.

*G.5.12. Rajzoljunk fel egy *előtöltéses* MOS invertert és ismertessük működését.

*G.5.13. Rajzoljuk fel egy szinkron-inverz *S–R* tárolóelem logikai vázlatát.

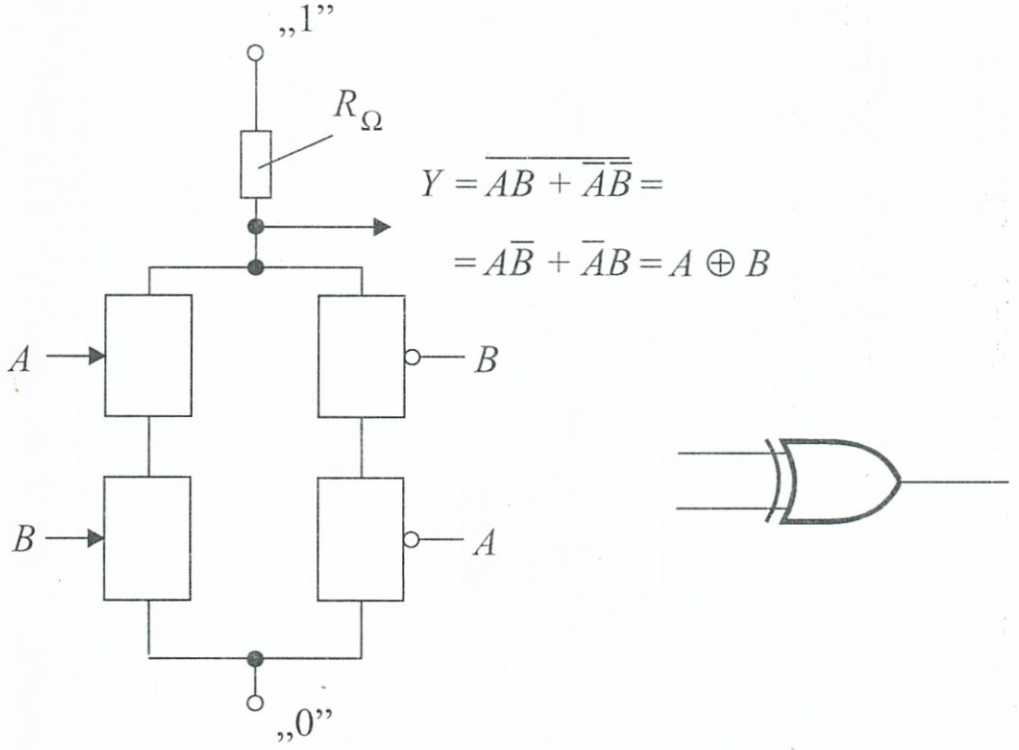
*G.5.14. Rajzoljuk fel egy *élvezérelt D*-tárolóelem logikai vázlatát.

*G.5.15. Rajzoljunk fel DATA LOCK OUT típusú élvezérelt Master–Slave *D*-tárolóelem logikai vázlatát.

12.5.2. Feladatmegoldások az 5. fejezethez

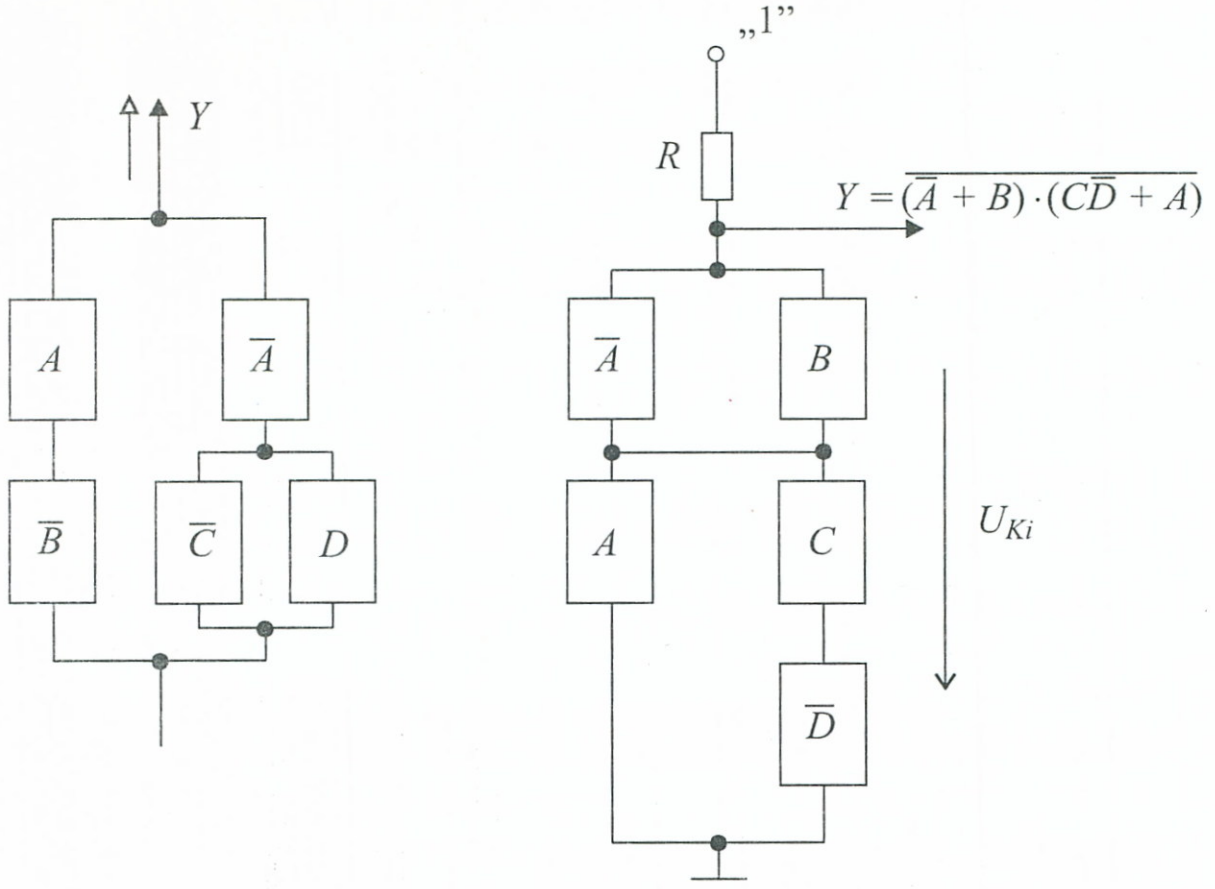
F.5.1. Az összefüggéseket az 5. fejezet (5.1) formulák és az 5–9. ábra adja meg.

F.5.2. A megoldás a 12–51. ábrán látható.



12–51. ábra *Antivalencia kapu kapcsolókkal realizálva*

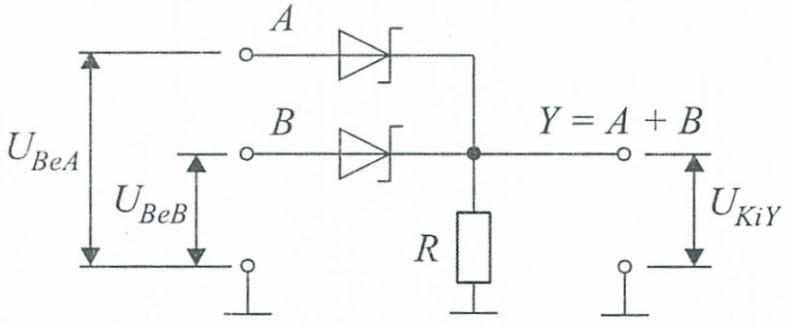
F.5.3. Az áramlogikás hálózat a 12–52a. ábrán a feszültséglogikás 12–52b. ábrán látható. Ez utóbbi felrajzolásához először az $\overline{Y}$-at állítottuk elő az 5–9b. ábra értelmében. A negált függvény:



12–52. ábra $Y = A\overline{B} + (\overline{C} + D)\,\overline{A}$ *realizációi*

$$\overline{Y} = (\overline{A} + B)\cdot(C \cdot \overline{D} + A)$$

F.5.4. Pozitív logikás megoldást rajzoltunk fel a 12–53a. ábrán. Itt bármelyik bemenetre *H* szint érkezik az illető dióda vezetni



A	B	Y
L	L	L
L	H	H
H	L	H
H	H	H

b.)

12–53. ábra *VAGY kapcsolás Schottky diódákkal*

kezd, így a kimeneten is *H* szint jelenik meg a 12–53b. táblázatnak megfelelően.

F.5.5. Mint láttuk az 5–12b. ábránál, itt a duálfüggvényt kell felírnunk. Ez a (2.6) összefüggés segítségével a következő lesz:

$$Y^D = [A\overline{B} + (\overline{C} + D)\overline{A}]^D = \overline{\overline{A} \cdot B + (C + \overline{D}) \cdot A} = (A + \overline{B}) \cdot (\overline{C} \cdot D + \overline{A})$$

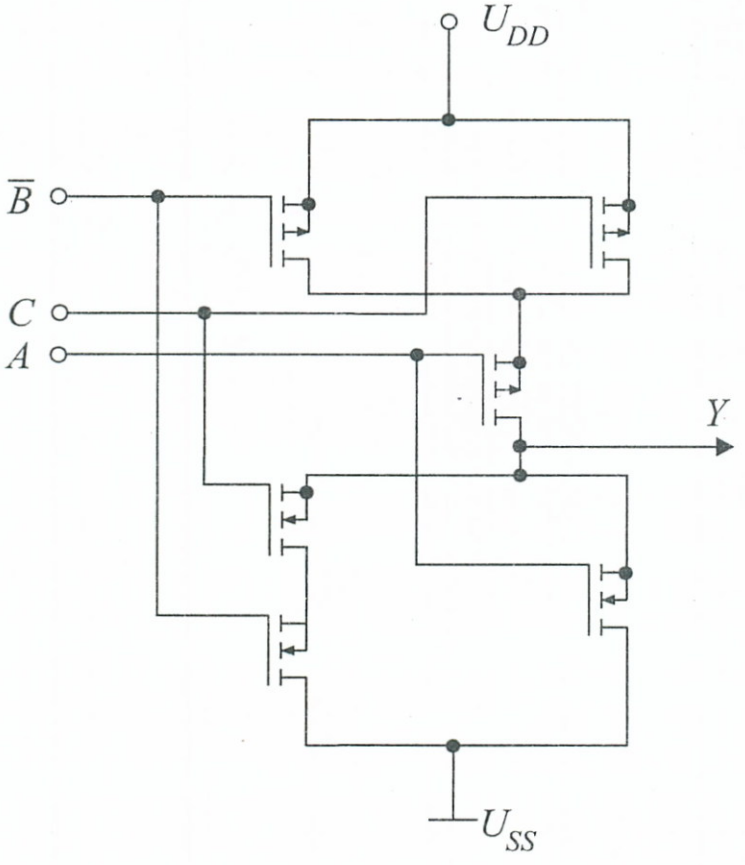
F.5.6. Ilyen megoldás látható az 5–29. ábrán.

F.5.7. A hálózat felrajzolásához az 5–31a. ábra szolgálhat kiindulásul. Eszerint képeznünk kell az *Y* kiindulófüggvény negáltját (H_A) és ennek duálját az n- és p-típusú hálózatrészekhez, mivel:

$$Y = Y_{Ki} = \overline{H}_A(Be)$$

$$HA = \overline{Y} = \overline{\overline{A}(B + \overline{C})} = A + \overline{B} \cdot C$$

$$H_A^{DUÁL} = \overline{\overline{A} + B \cdot \overline{C}} = A \cdot (\overline{B} + C)$$



12–54. ábra *CMOS realizálási példa*

A végleges hálózat a 12–54. ábrán látható, általánosított MOS jelölésekkel.

F.5.8. A *huzalozott* (WIRED) kapcsolások elvét az 5–21. ábra hálózata mutatja, melynek ún. *szabadkollektoros* (OPEN COLLECTOR) NAND elemét az 5–20c. ábrán tanulmányozhatjuk.
A megoldáshoz a kérdéses függvényt *negált diszjunktív* jellegű alakra kell hozni első lépésben, majd ez ennek megfelelő, 5–21. ábrához hasonló kapcsolást kell felrajzolni.

a) Megoldási változat:
A kiindulófüggvényt az alábbi módon átalakítjuk:

$$Y = A \cdot \overline{B} + (\overline{C} + D) \cdot \overline{A} = A\overline{B} + \overline{C}\,\overline{A} + D\overline{A} = \overline{\overline{A\overline{B}} + \overline{\overline{C}\,\overline{A}} + \overline{D\overline{A}}}$$

Az a) változat realizációját a 12–55a. ábra mutatja:

b) Megoldási változat:
Előállítjuk a kiinduló függvény *negáltját* minimál diszjunktív alakban és erre közvetlenül alkalmazzuk az *eleve negált típusú* alakot megvalósító 5–21. ábrabeli kapcsolást. A negálást a 12–55c. ábrán végeztük el a „0"-ás term-ek tömbösítésével, a végleges hálózatot itt a 12–55b. ábra mutatja. Mint látható, ez esetben a második változat az egyszerűbb.

F.5.9. A megoldásnál az 5–7b. ábra támpontul szolgál, de most az ÉS-funkciót előre hoztuk a kapcsolók elé (12–56. ábra).

F.5.10. A megoldásnál támpontul szolgálhatnak az 5–25. ábrán látható kapcsolások.

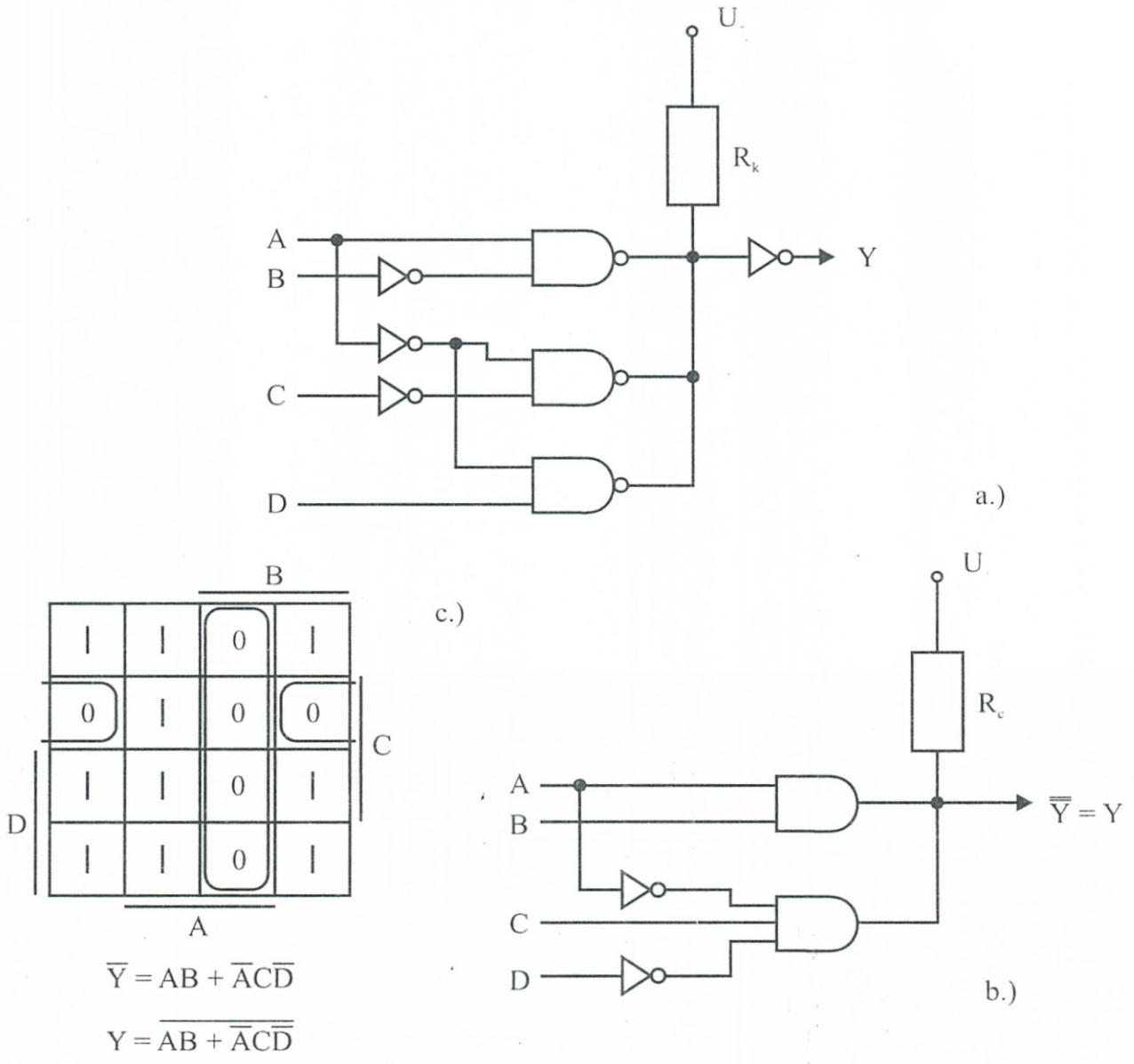
F.5.11. A megoldás az 5–34. ábrán látható.

F.5.12. A megoldás az 5–35. ábrán látható.

F.5.13. A megoldásnál támpontul szolgálnak az 5–37. ábrán látható kapcsolások.

F.5.14. Egy lehetséges megoldás az 5–42. ábrán látható.

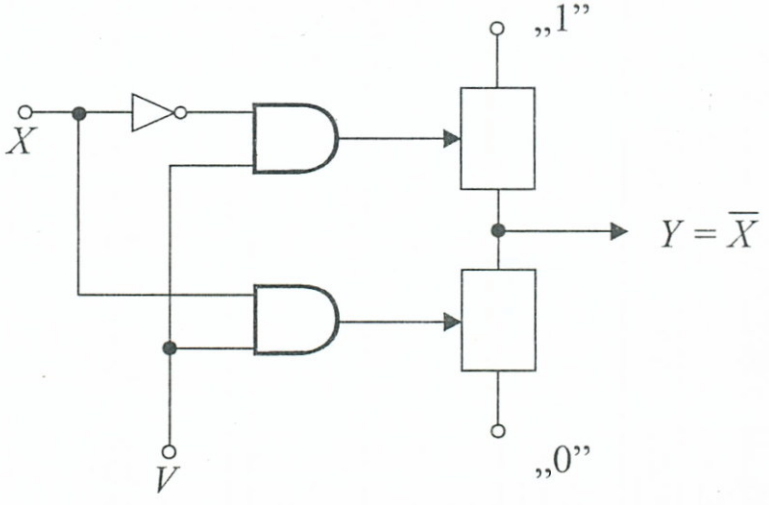
F.5.15. Egy megoldási változat az 5–43. ábrán látható.



$$\overline{Y} = AB + \overline{A}C\overline{D}$$

$$Y = \overline{AB + \overline{A}C\overline{D}}$$

12-55. ábra *A G.5.7. feladat realizációs változatai*



12–56. ábra *Háromállapotú inverter*

12.6.1. Gyakorló feladatok a 6. „Digitális hálózatok" c. fejezethez

G.6.1. A digitális hálózatok alapfogalmainak elmélyítése céljából gondoljuk végig mégegyszer a 6.1. Definíciók című pontban összefoglaltakat.

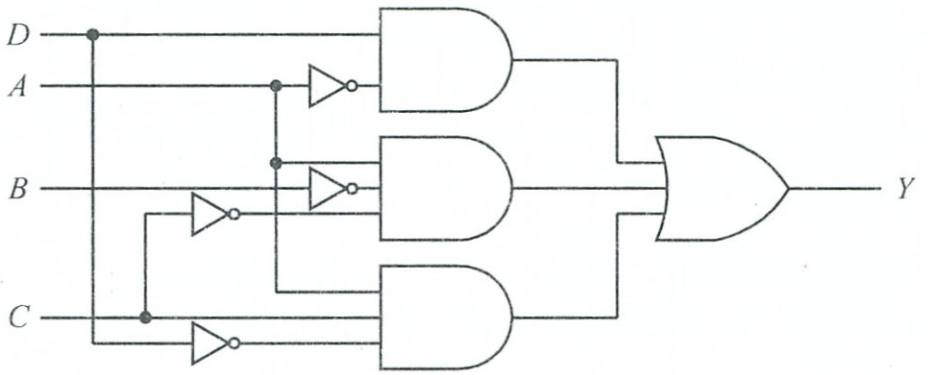
G.6.2. Az automataelméleti jellemzés elmélyítése érdekében ismételjük át a 6.1. Példát.

*G.6.3. Adva az alábbi függvény:

$$Y = \prod^{3}(0, 1, 2, 6, 7)$$

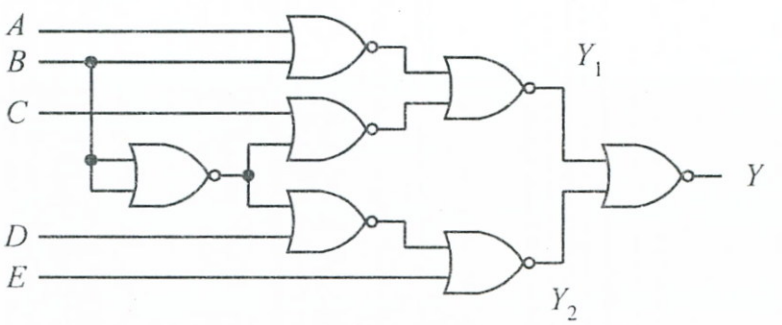
Rajzoljuk fel a megvalósítható legegyszerűbb, hazardmentes hálózatot.

*G.6.4. Vizsgáljuk meg a 12–57. ábra szerinti hálózatot hazard szempontjából. Ha találunk hazardot, küszöböljük ki:

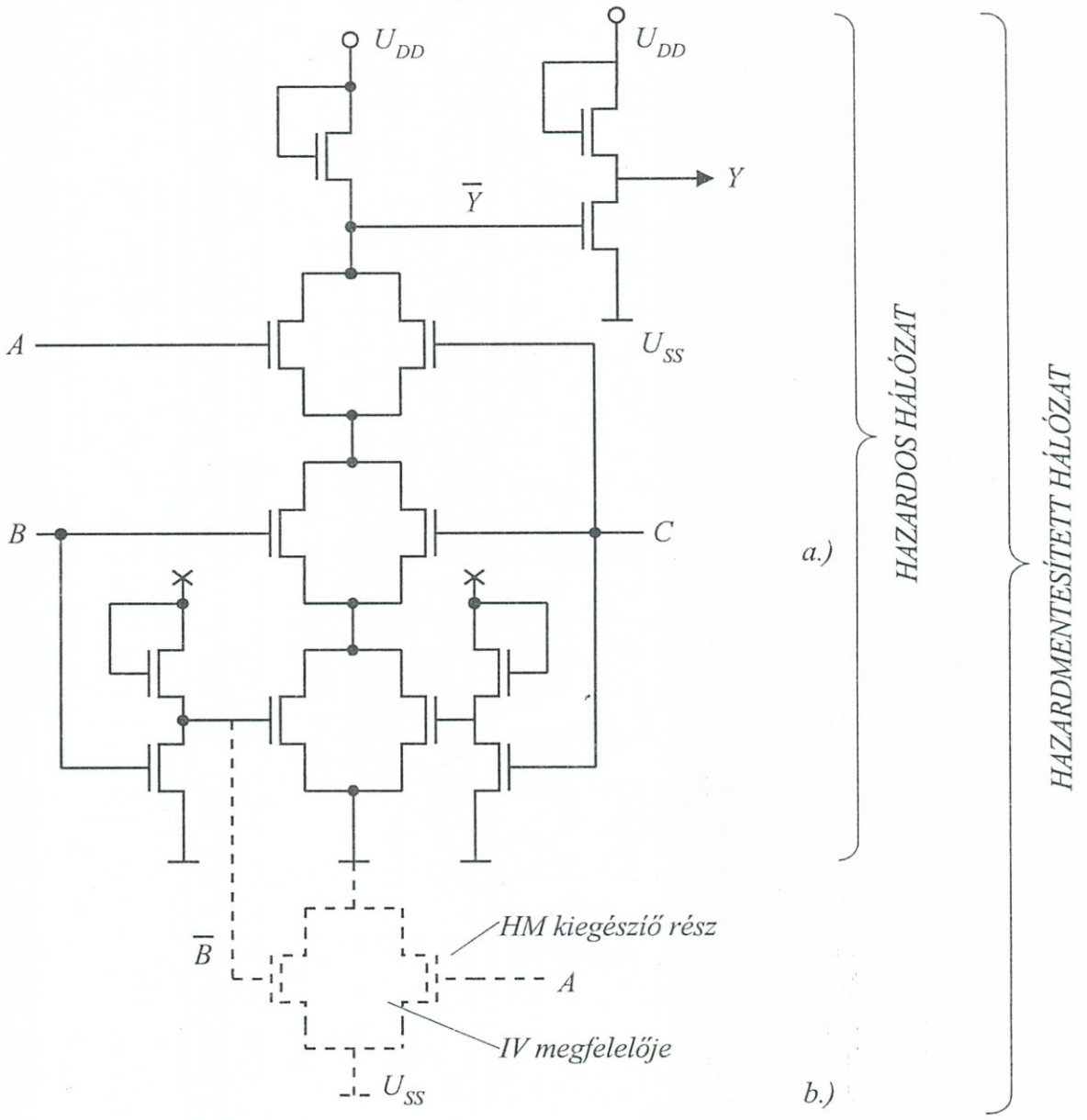


12-57. ábra *Kindulás a G.6.4. példához*

*G.6.5. Felléphet-e dinamikus hazard a 12–58. ábra hálózatánál? Ha igen, adjuk meg, hogy milyen bemeneti vezérlési feltétel esetén.



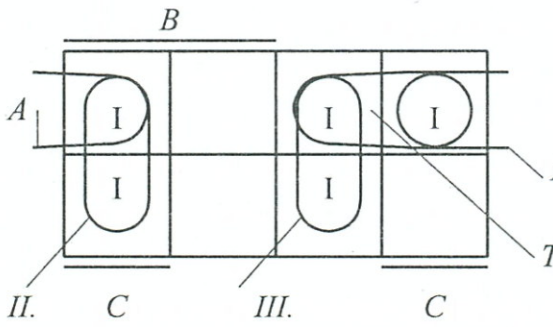
12-58. ábra *Kindulás a G.6.5. példához*



KIMETI FÜGGVÉNY: $Y=\overline{(\overline{Y})}=(A+C)(B+C)(\overline{B}+\overline{C})$

I. II. III.

MAXTERM TÁBLA:



I.–III. között hazard veszély!

I.

TÚLFEDŐ TÖMB(IV)

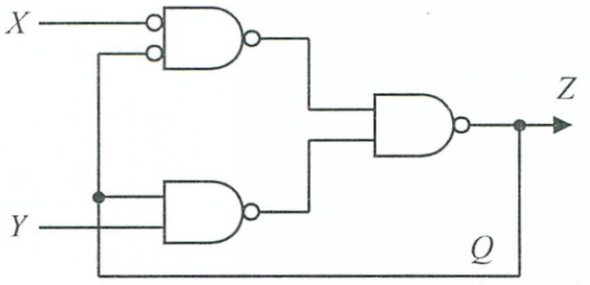
HAZARDMENTESÍTETT FÜGGVÉNY: $Y=(A+C)(B+C)(\overline{B}+\overline{C})(A+\overline{B})$

I. II. III. IV.

12-59. ábra *Hazardmentesítés MOS hálózatnál a G.6.6. példával kapcsolatban*

*G.6.6. Vizsgáljuk meg a 12–59a. ábrán látható MOS áramkört HAZARD szempontjából.

*G.6.7. Van-e gerjedési veszély, és hol a 12–60. ábrabeli visszacsatolt hálózatban, és lehet-e versenyfutás?



12-60. ábra *Visszacsatolt kombinációs hálózat a G.6.7. példához*

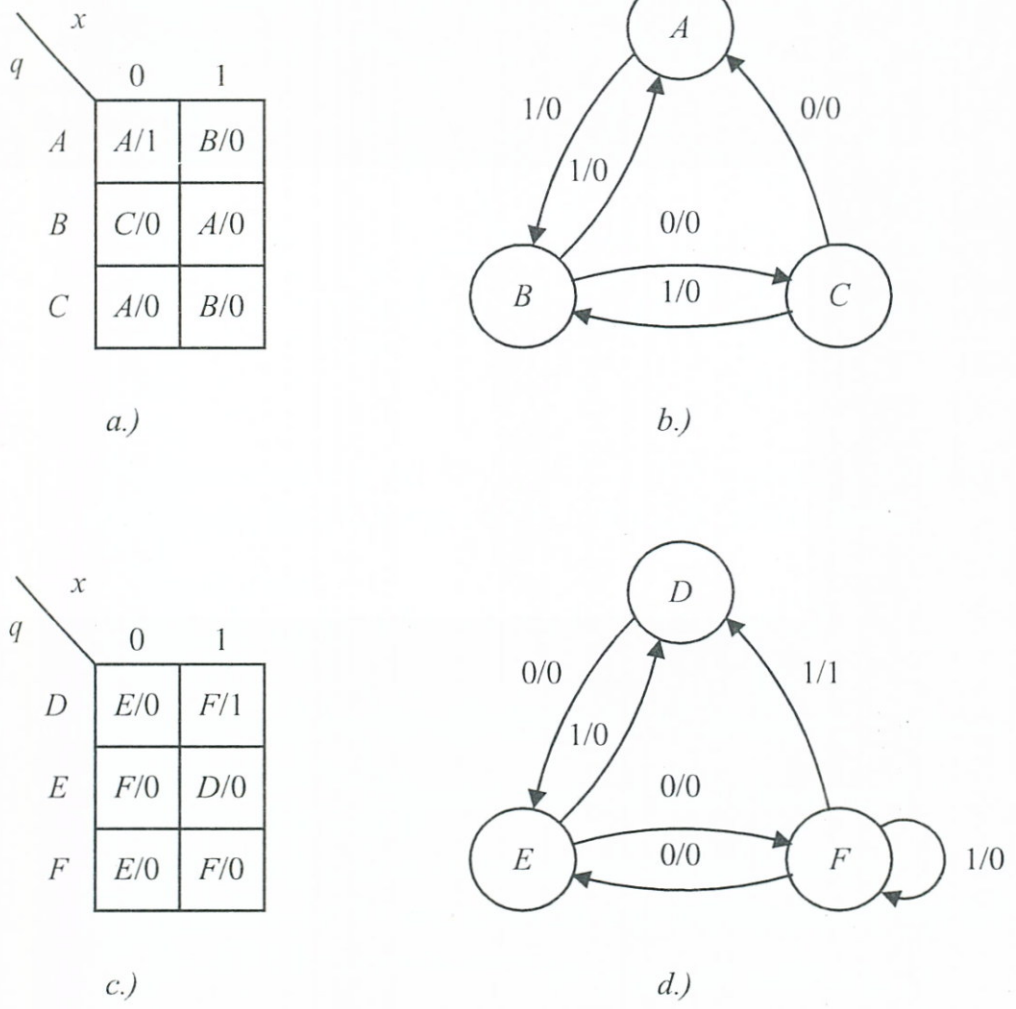
*G.6.8. Állapítsuk meg, milyen az a hálózat a 6–23a. ábra üres blokkjában, amely a 6–23d., e., f. tranziens-viselkedést mutatja?

*G.6.9. Vizsgáljuk meg a 2–24. ábra viselkedését a belső késleltetések figyelembevételével.

*G.6.10. Rajzoljunk fel állapotgráfokat a következő esetek szemléltetésére:

– iniciálisan összefüggő
– abszorbens
– izolált.

*G.6.11. Legyen adva két összetevő automata (SH_I és SH_{II}) a 12–61a, c. ábrán látható Huffman-táblákkal, és a 12–61b, d. állapotgráfokkal jellemezve. Az ábrán egyszerűsített jelöléseket vezettünk be, hogy az indexekkel történő bajlódást megtakarítsuk. Határozzuk meg a két összetevő *parallel kapcsolásának* eredőjét. Az eredő kimenetét előállító *KHφ kombinációs automata φ leképező-függvénye jellemezzen egy egyszerű diszjunkciót.* Legyen mindkét összetevőnk *iniciális* automata, és a kezdeti állapotok legyenek: SH_I-nél: „A", SH_{II}-nél: „*D*".

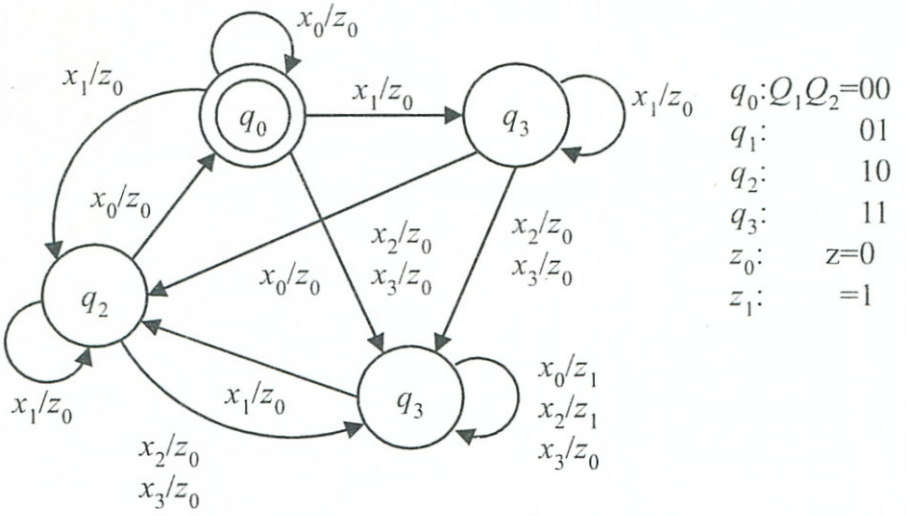


12-61. ábra *Komponens automaták a G.6.11. példához*

*G.6.12. Határozzuk meg a G.6.11. példabeli komponensek *soros* (kaszkád) összekapcsolásából származó eredő automatát. Legyenek itt is iniciális állapotok kijelölve, így SH_I-nél „A", SH_{II}-nél pedig „*E*"-ből történjék a kiindulás.

*G.6.13. Tekintsük a *T* flip-flop-ot egy komponens automatának. Kapcsoljunk kaszkádba a *T* flip-flopokból kettőt, és vizsgáljuk meg az eredő automata működését.

*G.6.14. Adva a 12–62. ábrán látható állapotgráf. Vizsgáljuk meg ütemdiagramok segítségével a gráffal jellemzett sorrendi hálózat működését a $q_0 : Q_1Q_2 = 00$ állapotból indulva



12-62. ábra *Kiinduló állapotgráf a G.6.14. példához*

$$s_x^7 = x_0 - x_1 - x_3 - x_2 - x_0 - x_1 - x_0 =$$

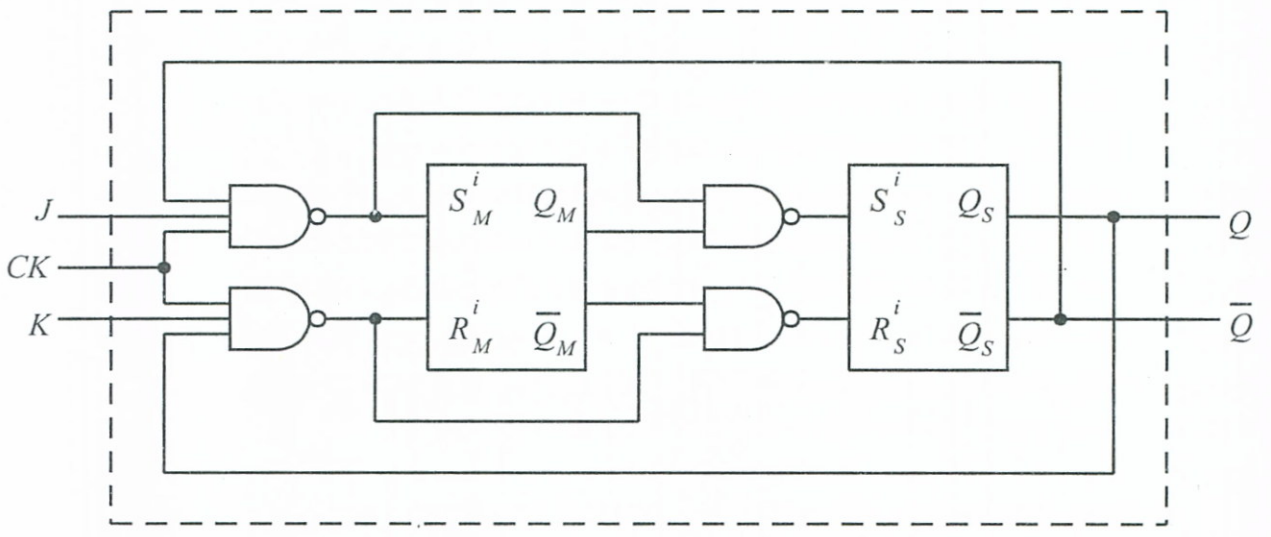
$$\overset{AB}{} = 00 - 01 - 11 - 10 - 00 - 01 - 00$$

bemeneti állapotsorozat hatására:

– először aszinkron,
– majd szinkron

működést feltételezve.

*G.6.15. Vizsgáljuk meg a 12–63. ábrán látható közbenső tárolós kapcsolás viselkedését a CK órajel egyes fázisaiban.



12-63. ábra *Kiinduló ábra a G.6.15. példához*

*G.6.16. Alakítsunk ki *J–K* tárolóelemet *D*-típusú elemből kiindulva.

*G.6.17. Alakítsunk ki az alábbi függvényegyüttesből kombinációs hálózatot a) ROM, b) PLA, c) PAL realizációs változatokban.

$$Y_1 = C\overline{B} + B\overline{A}$$

$$Y_2 = CA + \overline{C}B\overline{A}$$

$$Y_3 = C\cdot(A+B) + \overline{C}\cdot\overline{B}\cdot\overline{A}$$

$$Y_4 = \overline{C}\cdot\overline{B}\cdot\overline{A}$$

*G.6.18. Programozzunk fel egy MAX–LAB egységet a következő felhasználásra:

– tárolóelem
– egyedi órajel

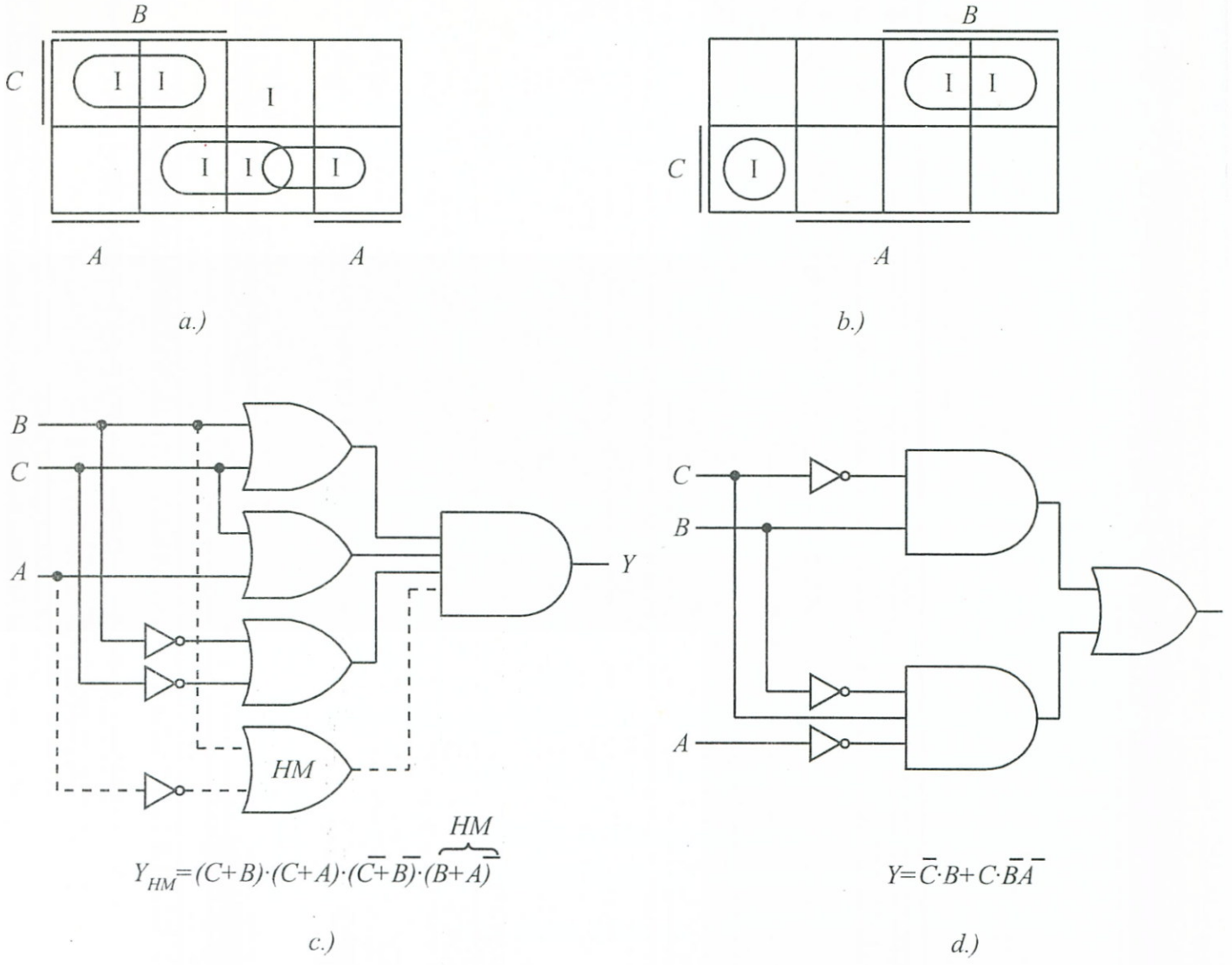
*G.6.19. Programozzunk fel egy XILINX–CLB egységet a következő felhasználásba:
– aktív tárolóelem
– tároló kimenete visszacsatolva
– a tárolóelem visszacsatoló ága kapuzottan egy 4-bemenetű KH-ra csatlakozzon.

12.6.2. Feladatmegoldások a 6. fejezethez

F.6.3. Felrajzolva a kiinduló konjunktív alakhoz tartozó *maxterm*-táblát a 12–64a. ábrán látható, hogy az élszomszédos tömbök miatt hazardveszély áll fenn, melyet a $(B+A)$ lefedő implikánssal lehet közömbösíteni (12–64c. ábra). Lényegesen egyszerűbb eredményhez jutunk, ha a *minterm*-táblára térünk át (12–64b. ábra). Itt nincsenek élszomszédos tömbök és a kapuk száma is csökkenthető, mint ez a 12–64d. ábrából látható.

F.6.4. A megoldáshoz fel kell írni a kimenetet:

$$Y = \underset{a}{D\cdot\overline{A}} + \underset{b}{\overline{C}\cdot\overline{B}\cdot A} + \underset{c}{C\overline{D}A}$$



12-64. ábra *Hazardmentesítés a G.6.3. példához*

A megoldást itt először a 6.2.1.1. pontban megfogalmazott hazard keletkezési feltételek alapján végezzük el, mely szerint *diszjunktív alak esetén a hazard feltétele* a következő:

– legyen legalább két alábbi típusú implikáns:

$$\left.\begin{aligned} I'_D &= X_i \cdot E' \\ I''_D &= \overline{X}_i \cdot E'' \end{aligned}\right\}$$ ahol: E', E" az ÉS-típusú implikánsok X_i, $\overline{X}_i$ nélküli részei

– és fennáll:

$$E'\cdot E'' \neq 0$$

– továbbá:

E', E" nem szerepel a diszjunktív alakban.

A példához már felírt Y függvénynél a, b, c-vel megjelöltük a szereplő implikánsokat és a fenti hazardfeltétel alapján páronként megvizsgáljuk őket:

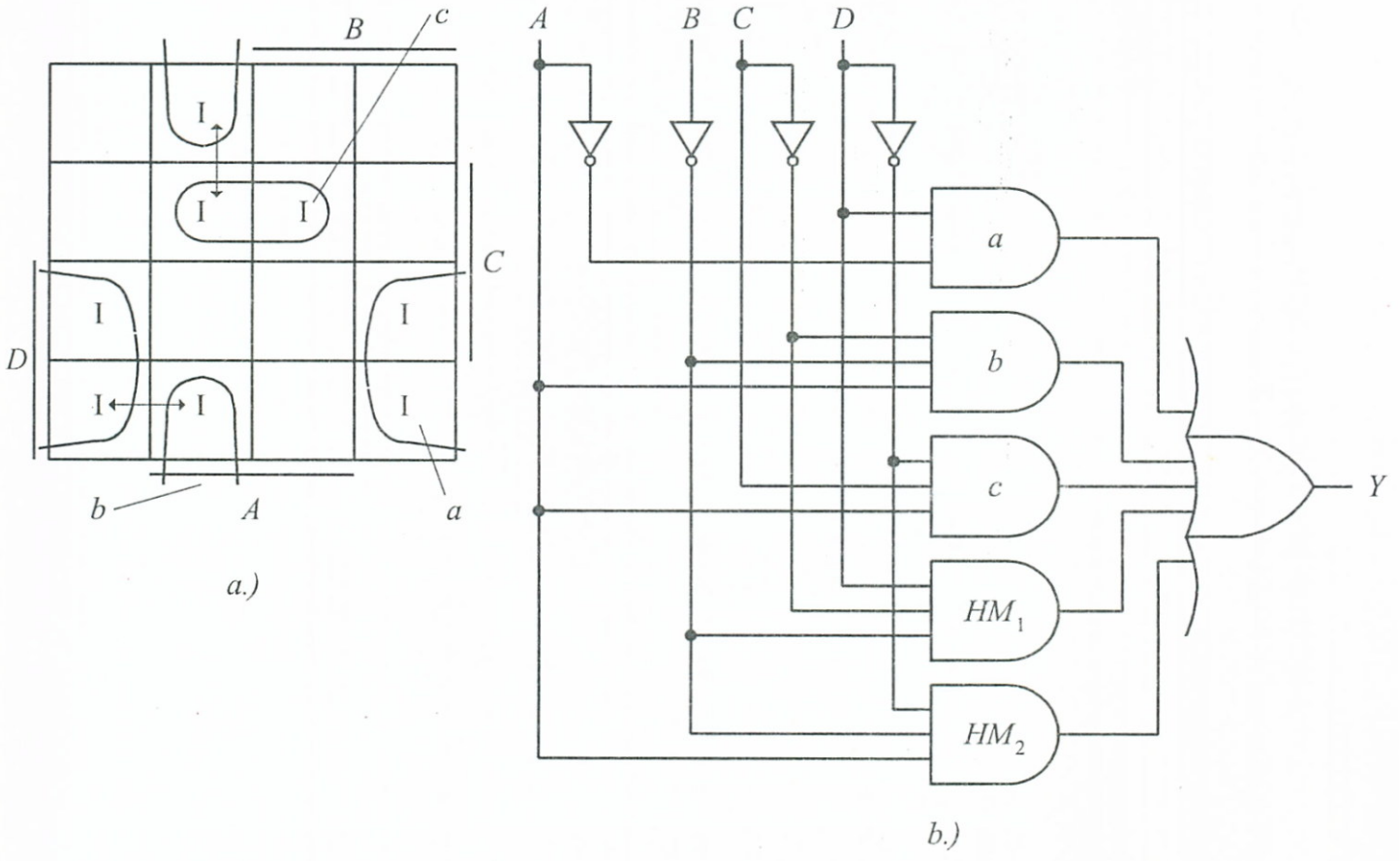
a–b: $D\cdot\overline{A}$ $\overline{C}\,\overline{B}\cdot A$ és $D\cdot\overline{C}\,\overline{B}\neq 0$ hazard lehet
(X_i)

a–c: $D\overline{A}$ $\overline{D}CA$ de $D\cdot\overline{D}C=0$ hazard nem lehet
(X_i)

b–c: $\overline{C}\,\overline{B}\,A$ $\overline{D}CA$ és $\overline{D}\cdot\overline{B}\cdot A\neq 0$ hazard lehet
(X_i)

A vizsgálat azt mutatja, hogy két esetben keletkezhet 1-es típusú sztatikus hazard. Azért 1-es típusú, mert a diszjunktív alak prim-implikánsai közötti átmenetkor keletkezhet, azaz $Y=1$ kimeneti érték közben.

Ha a $D\,\overline{C}\,\overline{B}$ és a $\overline{D}\,\overline{B}\,A$ prim-implikánsokat is hozzáadjuk a függvényhez, akkor megszüntetjük a hazard keletkezésének lehetőségét. Ugyanis pl. az a és b prim-implikánsokat realizáló kapuk közötti vezérlés-átadás közben a $D\,\overline{C}\,\overline{B}$-t megvalósító hazard-mentesítő kapu biztosítja az $Y=1$ érté-



12-65. ábra *Sztatikus – hazardoktól mentesített hálózat*

ket. Hasonlóan, a *b–c* esetben a $\overline{D}\,\overline{B}\,A$ kapu lesz a hazard-mentesítő.
Az elmondottakat grafikusan is követhetjük a 12–65a. ábra *élszomszédos* tömbjeinek keresésével.
Végül a hazard-mentesített hálózatot a 12–65b. ábrán rajzoltuk fel.

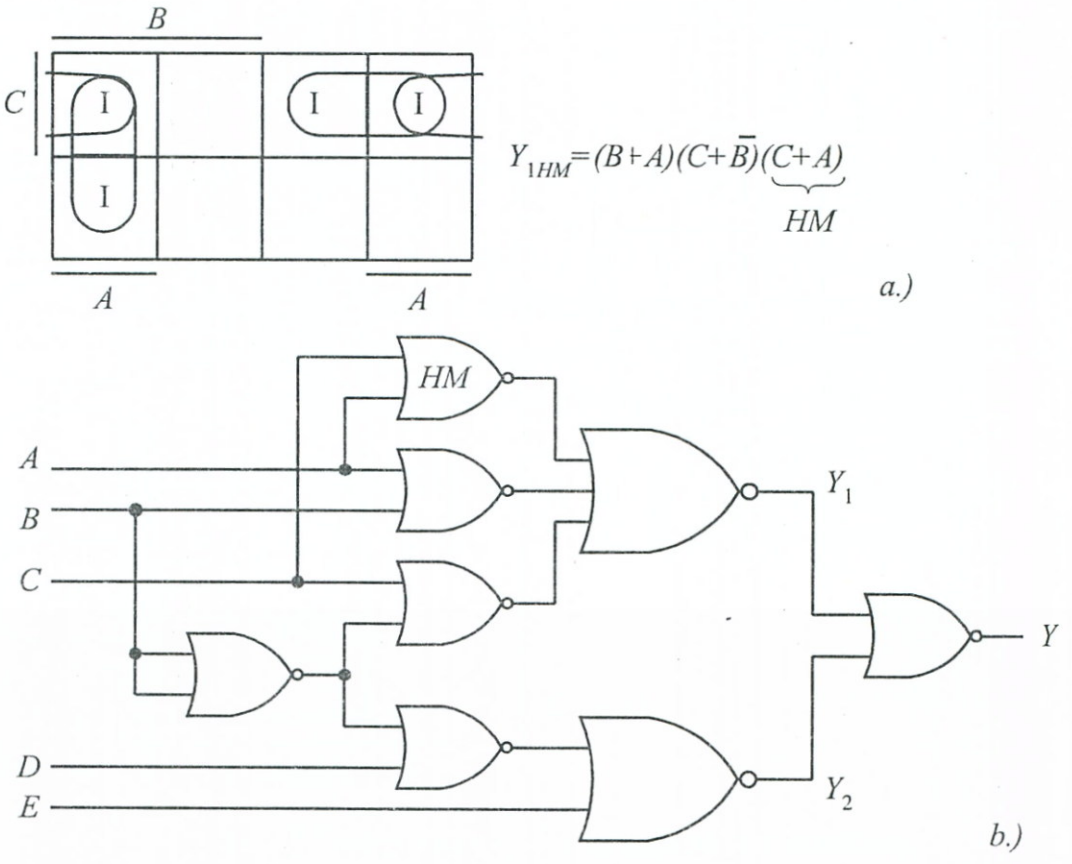
F.6.5. Dinamikus hazard mindig csak olyankor jelentkezhet, ha sztatikus hazard is keletkezett. Felírva az áramkör jellegzetes pontjainak függvényeit a NOR–NOR → NEM–ÉS–VAGY transzformációt nem részletezve (lásd: 2. fejezet 2–12b. ábra):

$$Y = \overline{Y}_1 \cdot \overline{Y}_2$$

$$Y_1 = (B + A) \cdot (C + \overline{B})$$ és $C + A \neq 1$ hazard lehet („0”-típusú hazard)

$$Y_2 = (D + \overline{B}) \cdot \overline{E}$$

ahol a hazard-vizsgálatot most a 6.2.1.1. pontbeli *konjunktív alakra* megfogalmazott hazard-feltételek szerint végeztük. Végül a 12–66a. ábra Y_1 V–K tábláján is bejelöltük a



12-66. ábra *Dinamikus hazardtól mentesített hálózas*

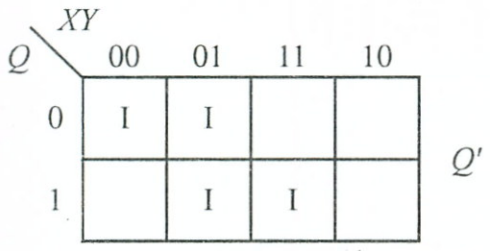
hazard-veszélyes részt, melyet a 12–66b. logikai vázlaton berajzolt HM-jelű kapuval közömbösítünk.

F.6.6. Az újdonságot ennél a feladatnál a MOS kapcsoló-elemekből felépülő *áramlogikás* hálózat jelenti. A 12–59. ábrán felírtuk az a kiinduló a) ábrarészből leolvasott konjunktív függvényt, melyet maxterm táblán ábrázolva felismerhető a hazard-veszély. A szükséges *túlfedő* tömböt képviselő hálózatrészt a b) ábrarésznél csatlakoztattuk a kiinduló hálózathoz.

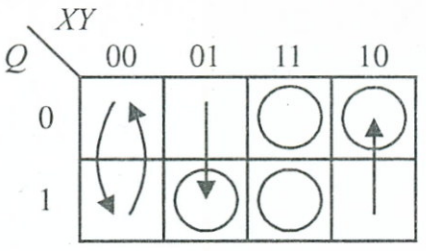
F.6.7. A jellemző függvények közül csak a belső állapotfüggvényt kell felírni, és mivel a $Q = Z$, így a kimeneti tábla is azonos lesz a gerjesztési táblával. A hálózatból felírható a *gerjesztési függvény,* esetünkre alkalmazva a NAND–NAND–ÉS–VAGY transzformációs formulát is (2–12. ábra, 24a. képlet):

$$Z = Q' = (\overline{X}|\overline{Q}) \mid (Y|Q) = \overline{X}\cdot\overline{Q} + Y\cdot Q$$

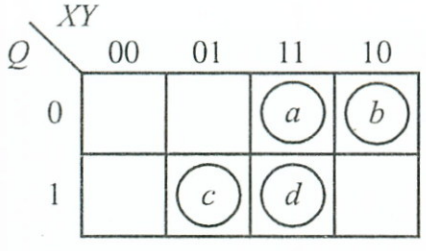
Ezzel felrajzolható a 12–67a. ábra szerinti gerjesztési tábla. A gerjesztési tábláról a $Q' = Q$ stabil, $Q' \neq Q$ instabil feltételek segítségével átjelölhetjük a 12–67b. átmeneti táblára az instabil és a bekarikázott stabil helyeket, továbbá az instabil állapotokból kiinduló belső átmeneteket. Mint látha-



a.) Gerjesztési tábla



b.) Átmeneti tábla



c.) Állapot tábla

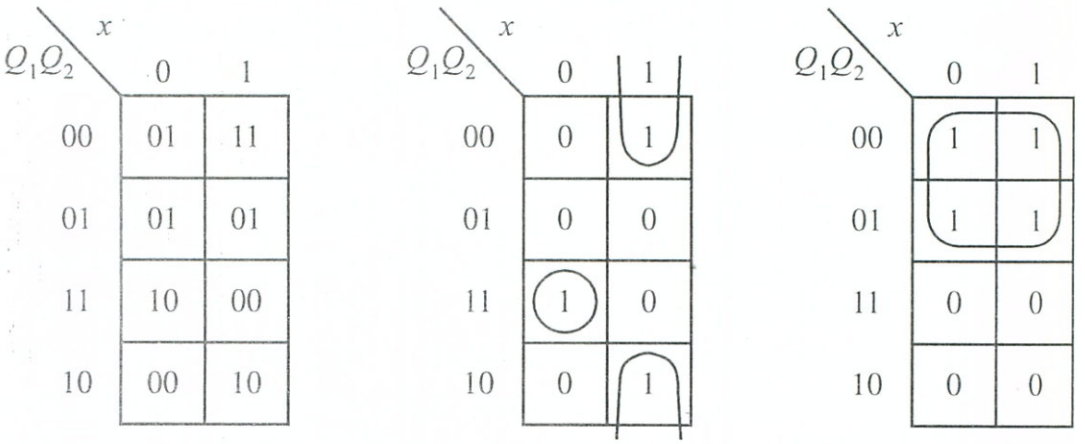
12-67. ábra *Jellemző táblák*

tó, az $XY = 00$ vezérlésnél gerjedés lép fel. Teljesség kedvéért a 12–67c. ábrán megadtuk az állapottáblát is.

F.6.8. A dobozban egy visszacsatolt rendszerbe illeszkedő KH van, mely három bemenettel (X, Q_1, Q_2) és két kimenettel (Q'_1, Q'_2) rendelkezik. Q'_1 és Q'_2 felírhatók a 6–23b. gerjesztési táblából, ha azt Q'_1–Q'_2 V–K táblákra szétválasztjuk a 12–68. ábra szerint. Innen a minimalizált függvények a 12–68b. és 12–68c. ábrákból:

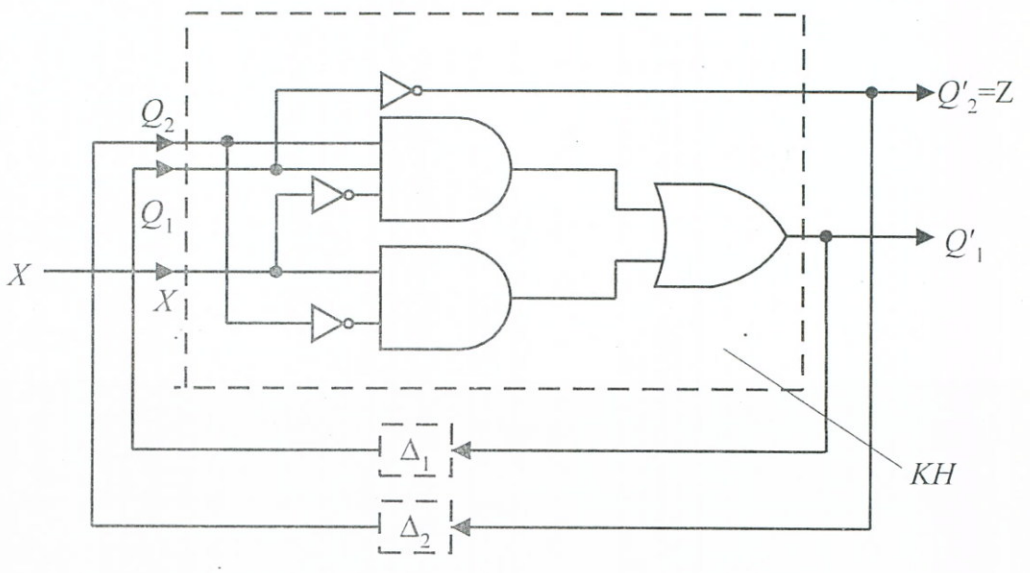
$$Q'_1 = X\cdot\overline{Q}_2 + \overline{X}\cdot Q_1\cdot Q_2$$
$$Q'_2 = \overline{Q}_1$$

A kérdéses KH a 12–68d. ábrán látható.



Gerjesztési tábla

a.) *b.)* *c.)*

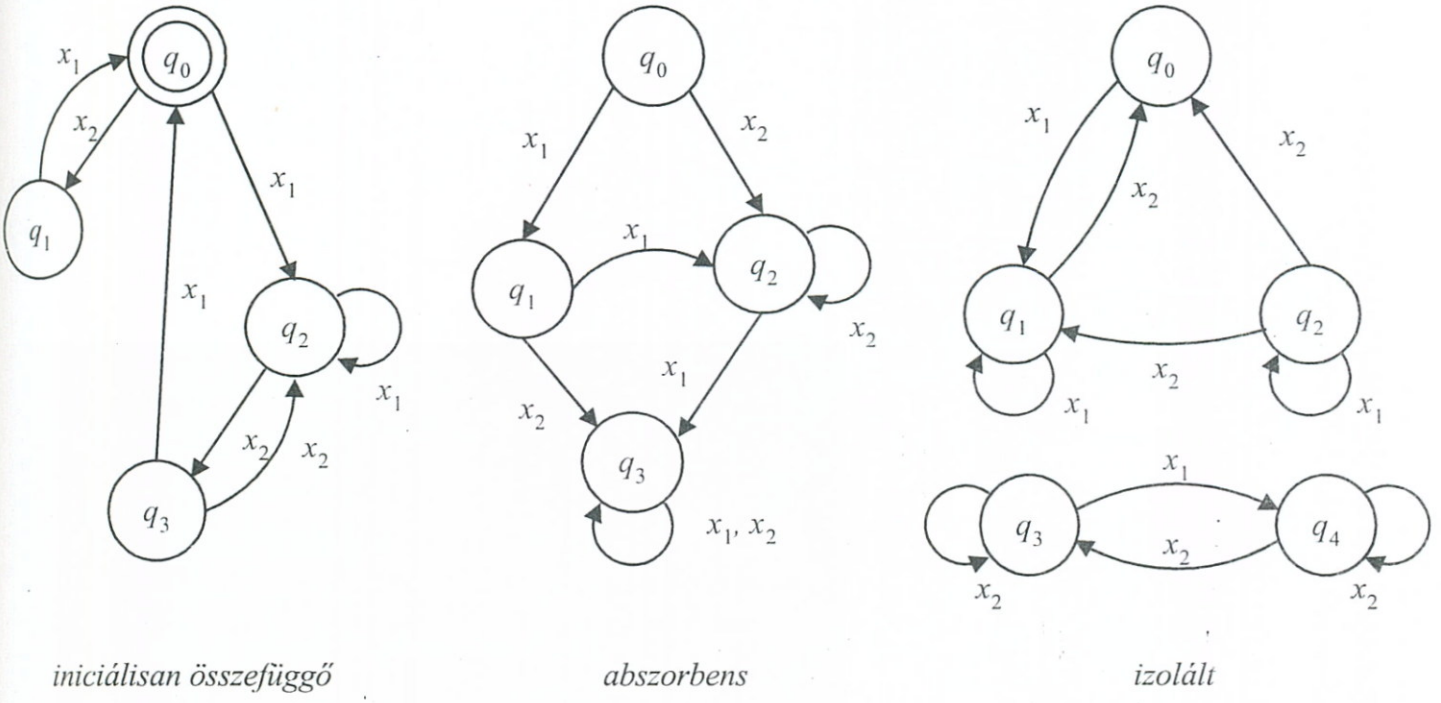


d.)

12-68. ábra *Tranziens viselkedésű visszacsatolt KH G.6.8. példához*

F.6.9. A feladatmegoldás menete teljesen hasonló a 6.8. példánál követetthez. Kiinduláskor a kiinduló 2–24. ábrát át kell rajzolni a 6–24b. ábra szerinti elrendezéssé, majd ennek figyelembevételével kell kitölteni a gerjesztési–átmeneti és állapottáblát – hasonlóan a 6–24c, d, e. ábrákhoz.

F.6.10. A 12–69. ábrán felrajzolt gráfok megfelelnek a feladatban felsoroltaknak.



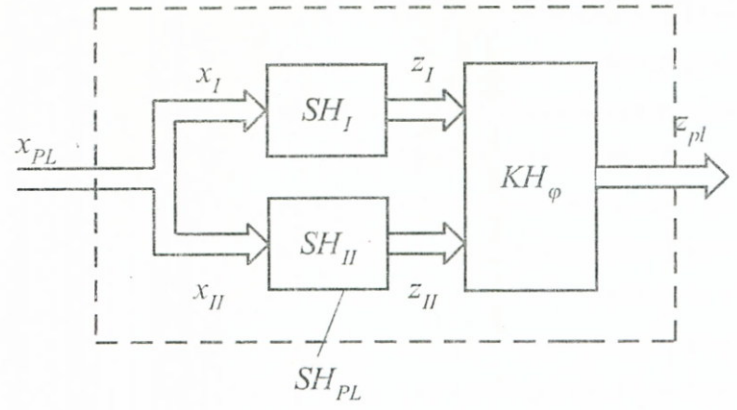
iniciálisan összefüggő *abszorbens* *izolált*

12-69. ábra *Gráfok a G.6.10. példához*

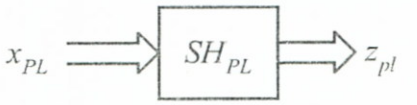
F.6.11. Mielőtt a példamegoldáshoz hozzáfogunk, célszerű az automaták összekapcsolásához néhány meggondolást fűznünk.

Több automatának, mint komponenseknek összekapcsolása újabb automatát eredményez, melynél az összetevők többféle struktúra szerint lehetnek összekapcsolva. Vizsgáljunk meg a lehetőségek közül a példabeli eseteket.

a) Párhuzamos összekapcsolás két komponenst tartalmazó változata látható a 12–70a. ábrán. Az ábráról leolvasható az eredő automata jellemzőinek adott esetben Mealy-modell szerinti előállítása is. Mint látható, a bemeneti állapothalmaz a komponensekre és az eredőre ugyanaz. Az eredő belső állapothalmazát az összetevők belső állapothalmazainak Descartes szorzata adja meg. Az eredő kimenetét a két komponens egy KHφ *kombinációs* automata révén állítja

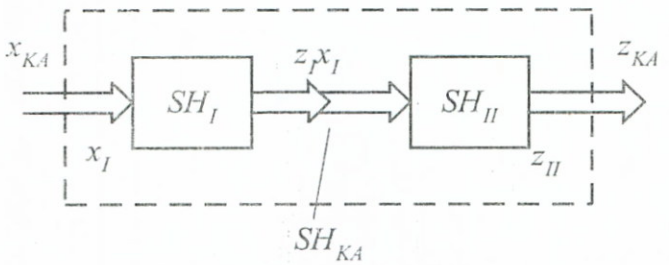


$SH_I=\{x_I, z_I, q_I \rho_I, \varphi_I\}$
$SH_{II}=\{x_{II}, z_{II}, q_{II} \rho_{II}, \varphi_{II}\}$
$SH_{PL}=\{x_{PL}, z_{PL}, q_{PL}, \rho_{PL}, \varphi_{PL}\}$

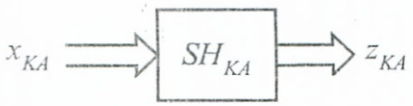


$x_{PL}=x_I=x_{II}$ $\quad q_{PL}=q_I \times q_{II}$
$q'_{PL}=\rho_{PL}(x_{PL}, q_{PL})=\rho_{PL}(x_{PL}, q_I \times q_{II})$
$z_{PL}=\psi_{PL}(x_{PL}, q_{PL})=\psi_{PL}(x_{PL}, q_I \times q_{II})$
$z_{PL}=\varphi(z_I, z_{II})$

a.) Párhuzamos



$SH_I=\{x_I, z_I, q_I, \rho_I, \psi_I\}$
$SH_{II}=\{x_{II}, z_{II}, q_{II}, \rho_{II}, \psi_{II}\}$
$SH_{KA}=\{x_{KA}, z_{KA}, q_{KA}, \rho_{KA}, \psi_{KA}\}$



$x_{KA}=x_I$ $\quad q_{KA}=q_I \times q_{II}$
$z_I=x_{II}$
$q'_{KA}=\rho_{KA}(x_{KA}, q_{KA})=\rho_{KA}(x_I, q_I \times q_{II})$
$z_{KA}=z_{II}$
$z_{KA}=\rho_{KA}(x_{KA}, q_{KA})=\rho_{KA}(x_I, q_I \times q_{II})$

b.) Soros (Kaszkád)

12-70. ábra *Autmaták párhuzamos és soros eredője*

elő. Ennek φ leképező függvénye mindig az adott konkrét feladat előírásai szerint alakul. (Pl.: egyszerű ÉS-kapcsolat stb.)

b) Soros (kaszkád) összekapcsolás két komponenst tartalmazó változata látható a 12–70b. ábrán. Az automatát itt is Mealy-modell-lel jellemezve, felírtuk az eredő és a komponensek közti kapcsolatokat. Az ábráról közvetlenül leolvasható, hogy az eredő bemenete megegyezik az SH_I komponens bemenetével, kimenete pedig az SH_{II} kimenetével stb.

Ezek után a G.6.11. példára visszatérve, mivel a két összetevő automata iniciális, azaz kijelölt kezdő működési álla-

$x_1=x_{I1}=x_{III}=x_{PL1}=0$ $\quad q_{I1}=A,\quad q_{I2}=B,\quad q_{I3}=C$

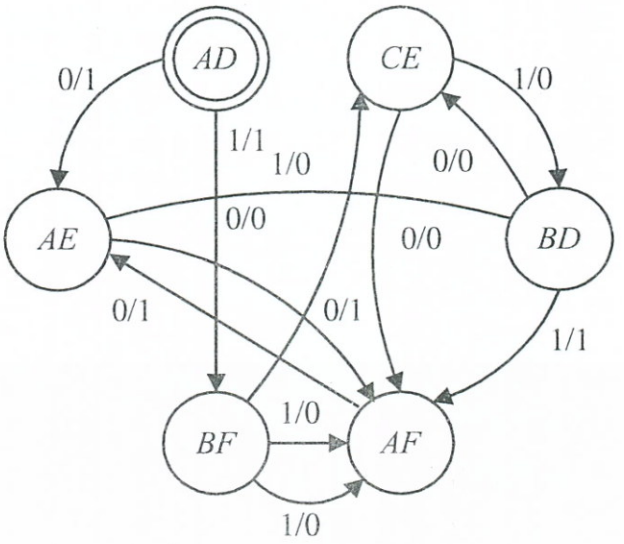
$x_2=x_{I2}=x_{III2}=x_{PL2}=0$ $\quad q_{III}=D,\quad q_{III2}=E,\quad q_{III3}=F$

$z_{I1}=0,\quad z_{I2}=1,\quad z_{PL1}=0$

$z_{III}=0,\quad z_{III2}=1,\quad z_{PL2}=1$ $\qquad z_{PL}=z_I+z_{II}$

q \ x	0	1
AD	AE/1	BF/1
AE	AF/1	BD/0
BF	CE/0	AF/0
AF	AE/1	BF/0
BD	CE/0	AF/1
CE	AF/0	BD/0

a.)



b.)

12-71. ábra *Két komponens automata párhuzamos eredője*

pottal rendelkezik, ennek megfelelően az *eredő* is egy iniciális automata, melynek kezdeti állapota „*AD*" lesz. Az eredő automata Huffman táblájának és állapotgráfjának előállítása céljából haladjunk végig a működésen. Például a kezdő állapotból kiindulva adjunk a bemenetre: $x_{PL1} = x_{I1} = x_{III} = x_1 = 0$ állapotot. Ekkor SH_I új állapota ismét „*A*", SH_{II}-é pedig a tábla alapján „*E*" lesz. Az eredő ennek megfelelően $x_1 = 0$ hatására „*AE*" állapotot fog felvenni. A kimeneti állapotnál $x_1 = 0$ hatására SH_I változatlanul „1", SH_{II} változatlanul „0" lesz, és a kettő „φ" szerinti diszjunkciója „1", ami az eredő SH_{PL}-nek $x_1 = 0$ hatására bekövetkező új állapotát adja meg. Hasonló próbálgatásokkal lépésről lépésre felépíthető az eredő automata 12–71a. és b. ábra szerinti Huffman-táblája és állapotgráfja. Mivel a kezdő álla-

potokat kikötöttük, jelen példabeli automata nem fogja a Descartes szorzat minden lehetséges párosítását felvenni, így a 12–71. ábra sem tartalmaz minden elvileg elképzelhető cellát, csomópontot és átmenetet. Ha a kezdő állapot nem lenne kötött, akkor az ábra bővülhetne.

F.6.12. A kiinduló komponensek itt is iniciálisak (SH_I: *A*, SH_{II}:*E*), az eredő kiinduló állapota tehát: „*AE*". Adjunk a bemenetre $x_{KA1} = x_{I1} = 0$ állapotot. Ekkor SH_I „*A*"-ban marad, és kimenete: $z_{I1} = 0$ lesz. Az SH_{II} bemenetét $z_{I1} = 0 = x_{II1}$ képezi ennek hatására a második komponens „*E*"-ből „*D*"-be megy, és kimenete $z_{II1} = 0$ lesz. Ily módon az eredő SH_{KA} automata az „*AD*" állapotot veszi fel, és $z_{II} = z_{KA}$ egyenlőség miatt, ak-

$x_{KA1}=x_{I1}=0$ $\quad z_{II1}=x_{KA1}=0$

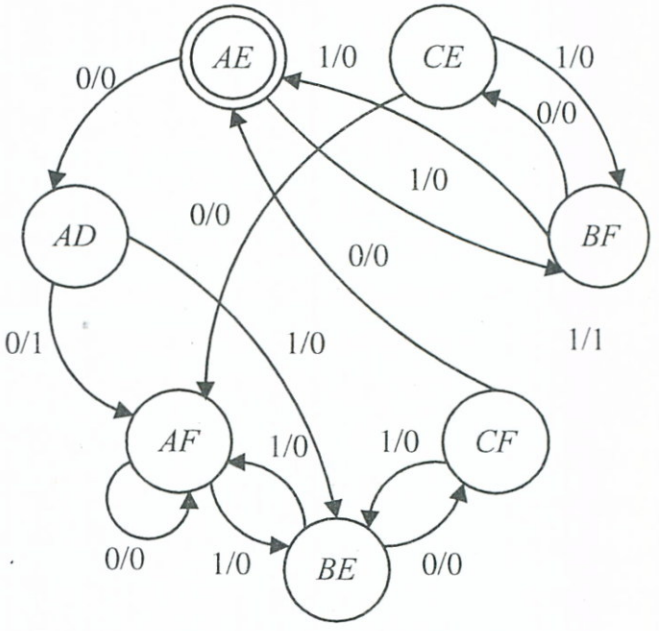
$x_{KA2}=x_{I2}=1$ $\quad x_{II2}=x_{KA2}=1$

$x_{II1}=z_{I1}=0$

$x_{II2}=z_{I2}=1$

q \ x	0	1
AE	*AD*/0	*BF*/0
AD	*AF*/1	*BE*/0
AF	*AF*/0	*BE*/0
BE	*CF*/0	*AF*/0
CF	*AE*/0	*BE*/0
BF	*CE*/0	*AE*/0
CE	*AF*/0	*BF*/0

a.)

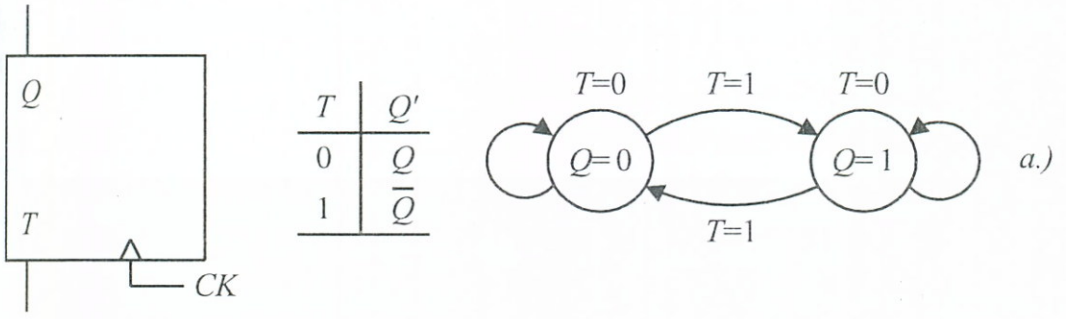


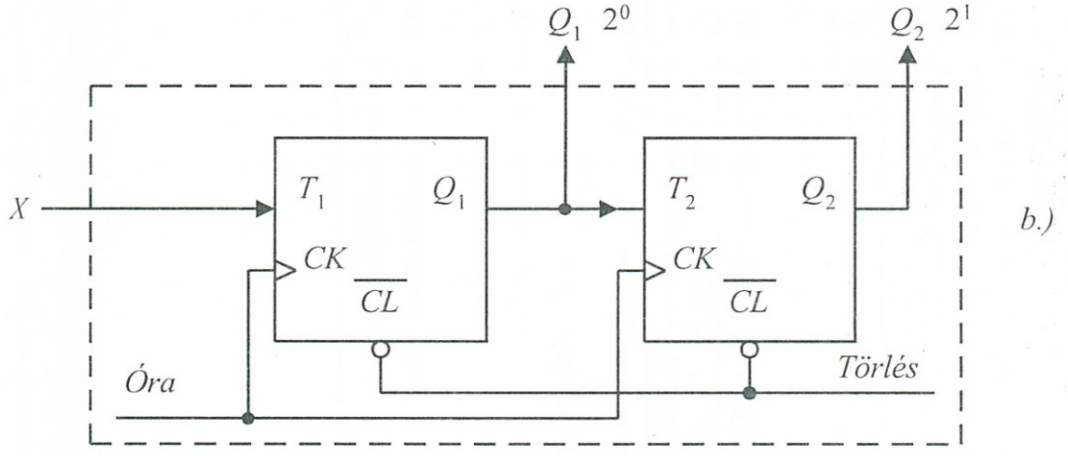
b.)

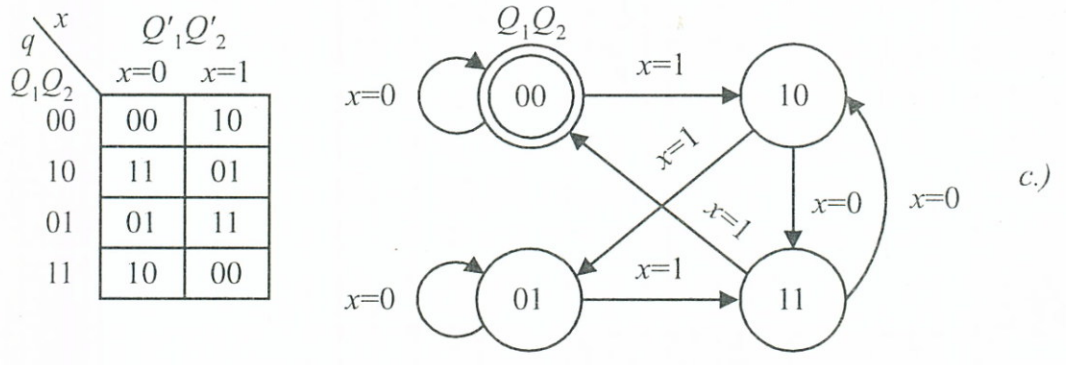
12-72. ábra *Két komponens automata soros eredője*

tuális kimenete: $z_{KA1} = x_{III} = 0$ lesz. Hasonló további lépésekkel a teljes Huffman-tábla és állapotgráf felépíthető (12–72. ábra). Az iniciális jelleg miatt itt sem lesz minden eset meghatározott.

F.6.13. Ez a feladat az előző feladatnak egy konkrétabb, gyakorlati változata. Tételezzük fel, hogy szinkron T flip-flopjaink vannak, melyekre mint komponens automatákra felrajzolható a 12–73a. ábra állapottáblája és állapotgráfja a 2–25. ábránál már bevezetetteknek értelmében. A kaszkád kapcsolás elrendezését a 12–73b. ábrán rajzoltuk fel.





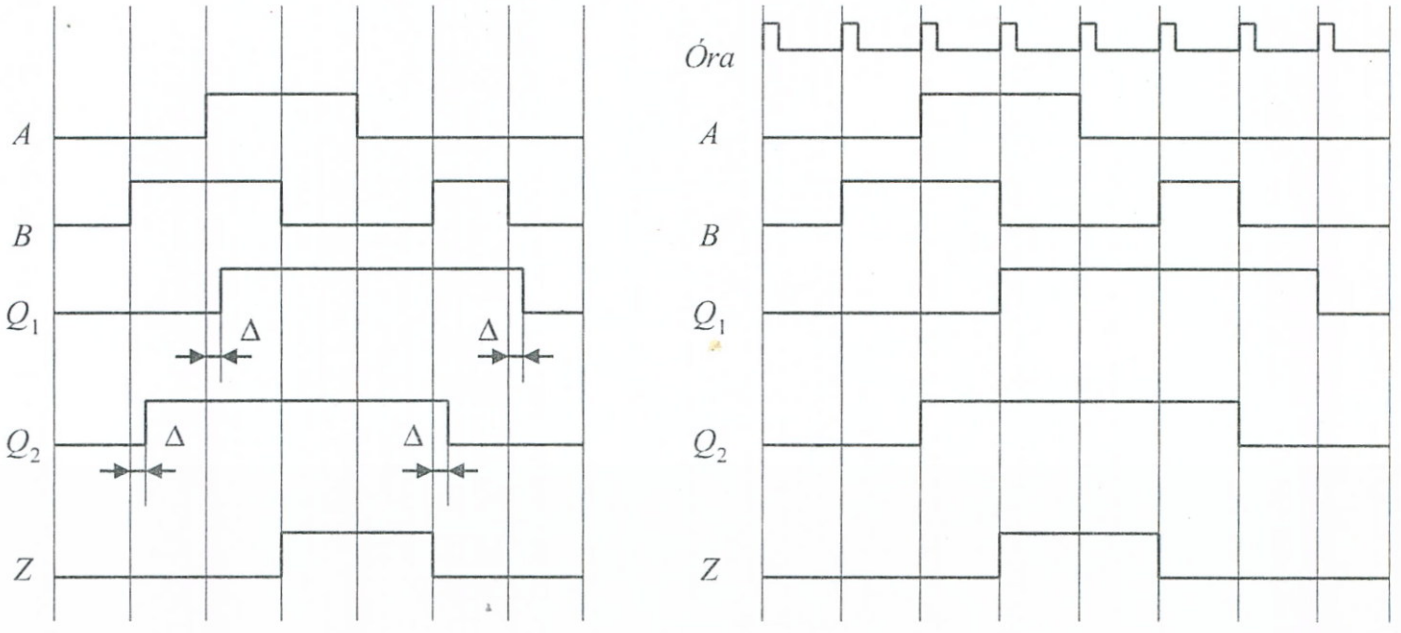


12-73. ábra *Két T tároló soros eredő hálózata*

Tételezzük fel, hogy Q_1Q_2–00 iniciális állapotból indítjuk a vizsgálatot, pl. a $CL = 0$ törlés után. Az összetevő T-tárolók közül az fog állapotot változtatni, amelyiknek bemenetére a CK órajel előtt $T_i = 1$ érkezett. A $T_i = 0$ esetet követő órajelre a tároló marad korábbi állapotában. E meggondolásokkal kitölthető a 12–73c. ábrán látható eredő állapottábla és állapotgráf. Az így kapott automata például állandó $X = 1$ esetén. Úgy működik, mint a 10. fejezetben megismert *soros binér számlánc* szinkron változata, amely a CK órajeleket számlálja, ha a Q_1Q_2 kimenetek bináris súlyozása megegyezik a 12–73b. ábrán látható bejelöléssel.

F.6.14. Az *aszinkron* esetnél a Q_1, Q_2 szekunder (belső) változók csak Δ *áramköri késleltetéssel* követik a bemeneti A, B változók változásait (12–74a. ábra).
Szinkron esetben minden változás a szinkron jel fellépésekor jelentkezik (példánknál feltételezetten felfutáskor) a 12–74b. ábrának megfelelően.

F.6.15. A vizsgálathoz a *gerjesztési* és az *átmeneti* táblát kell előállítanunk, melynél a CK órajelet is bemeneti változónak tekintjük. Az átmeneti táblát megelőző *gerjesztési* tábla felrajzolásához az inverz S–R tárolóelemre vonatkozó működési függvények felhasználásával esetünkre felírhatók a következők:



a.) aszinkron *b.) szinkron*

12-74. ábra *Aszinkron és szinkron működés összehasonlítása*

$$Q = Q_s' = \overline{S_s^i} + R_s^i \cdot Q_s =$$

$$= \overline{(J \cdot CK \cdot \overline{Q_s}) \cdot Q_M} + \overline{\overline{Q}_M \cdot \overline{(K \cdot CK \cdot Q_s)}} \cdot Q_s =$$

$$= \overline{J} \cdot Q_M + \overline{CK} \cdot Q_M + Q_s \cdot Q_M + K \cdot CK \cdot Q_s$$

$$Q_M' = (J \cdot CK \cdot \overline{Q}_s) + \overline{(K \cdot CK \cdot Q_s)} \cdot Q_M =$$

$$= J \cdot CK \cdot \overline{Q}_s + \overline{K} \cdot Q_M + \overline{CK} \cdot Q_M + \overline{Q}_s \cdot Q_M$$

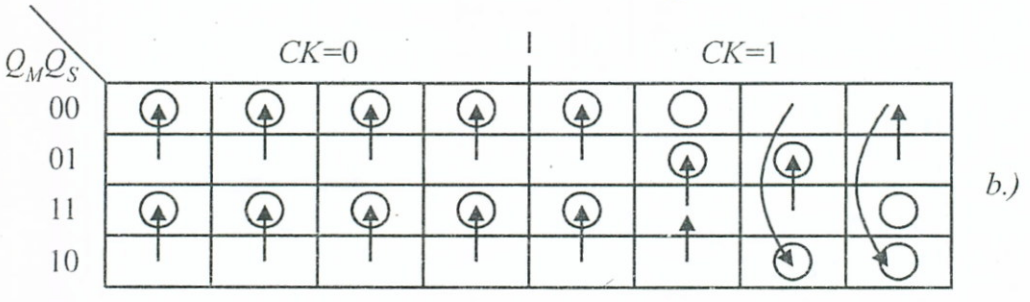
A gerjesztési tábla ezután a 12–75a. ábra szerint felrajzolható, ebből pedig már adódik az átmeneti tábla a 12–75b. ábrának megfelelően.

Az átmeneti táblát kiértékelve megállapíthatjuk, hogy amíg az órajel: $CK = 0$, addig J és K-nak nincs hatása Q_M és Q_s-re. Az órajel *felfutásakor* ($CK = 0 \rightarrow 1$) Q_M a J, K szerint változik, független tőlük. Állandó magas órajel: $CK = 1$ idején J, K mindkét változásnak hatásával van, míg lefutáskor ($CK = 1 \rightarrow 0$) Q_s felveszi Q_M értéket.

A tapasztalt működés megfelel az 5–38. ábrával kapcsolatosan bevezetett Master–Slave típusú működésnek. Példabeli hálózatunknál ügyelni kell arra, hogy $CK = 1$ állapotban J, K közül egyik sem változzon, mert ez megzavarhatja a Master–Slave üzemet.

Q_MQ_S \ CK,J,K	000	001	011	010	100	101	111	110
00	00	00	00	00	00	00	10	10
01	00	00	00	00	00	01	01	00
11	11	11	11	11	11	01	01	11
10	11	11	11	11	11	11	10	10

a.)



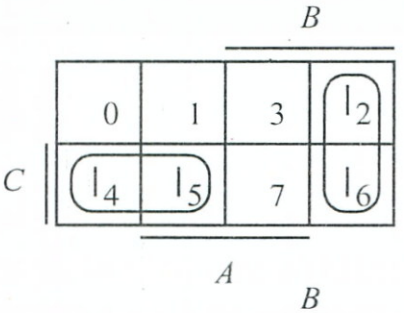
b.)

12-75. ábra *Gerjesztési- és átmeneti táblák*

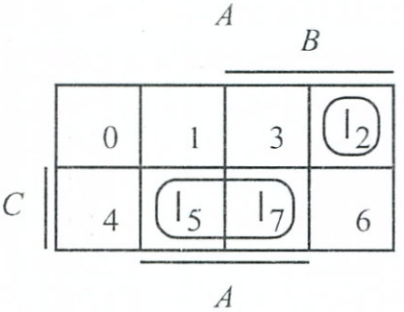
F.6.16. A feladatmegoldás menete teljesen megegyezik a 6.12. Példában tapasztaltakkal, csak itt a kiindulás a D-tárolóelemből történik és az eredményül kapott hálózatnak a J–K tároló definíciós táblázata szerint kell működnie.

F.6.17. Vizsgáljuk meg sorban a felsorolt eseteket:

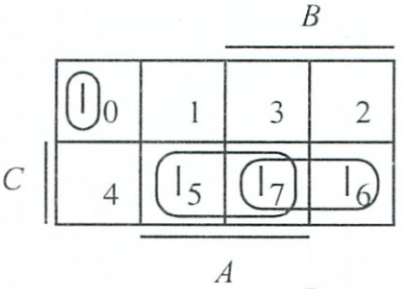
a) *ROM-realizáció:*
A 6–43. ábra értelmében a PLD *ÉS-mátrixa* ROM realizációnál *fix* (előregyártott) és tartalmazza az összes *minterm*-et, melyek esetünkben 3-változósak. Az ÉS-mátrix tehát példánkban az m_0^3, ... m_7^3 minterm-eknek megfelelően 8 kimenetű.
Első feladatunk, hogy a megadott kiinduló függvényekről megállapítsuk, hogy a teljes diszjunktív alakjuk, melyik



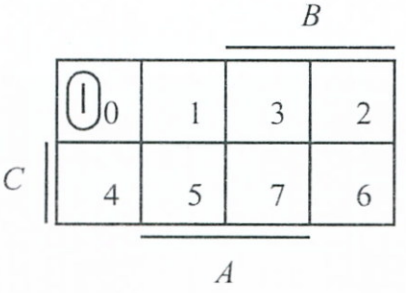
$$Y_1 = C\overline{B} + B\overline{A} = \sum^{3}(2,4,5,6)$$



$$Y_2 = CA + \overline{C}B\overline{A} = \sum^{3}(2,5,7)$$



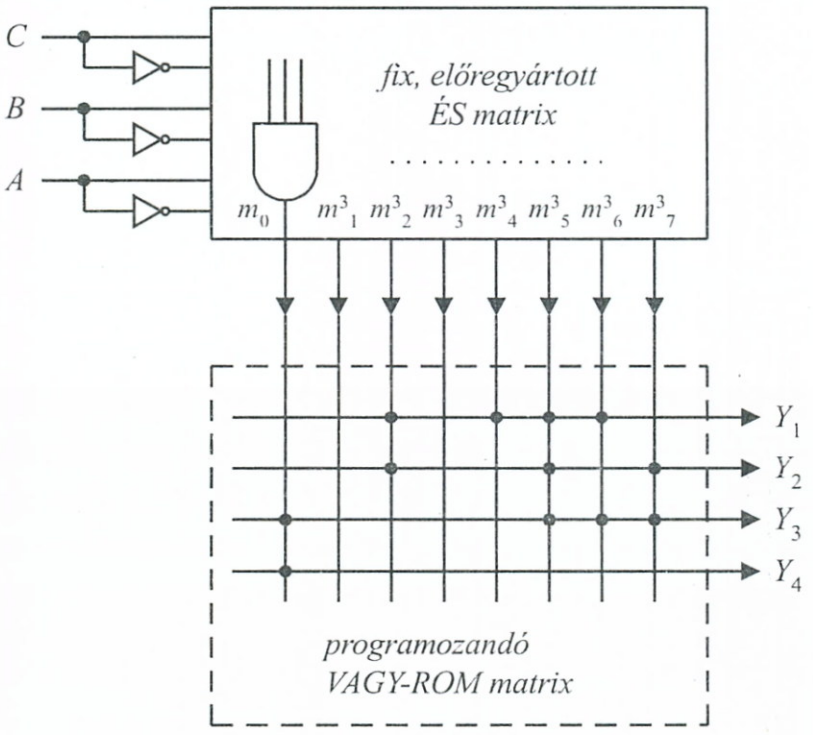
$$Y_3 = C(A+B) + \overline{C}B\overline{A} = CA + CB + \overline{C}B\overline{A} =$$
$$= \sum^{3}(0,5,6,7)$$



$$Y_4 = \overline{C}B\overline{A} = \sum^{3}(0)$$

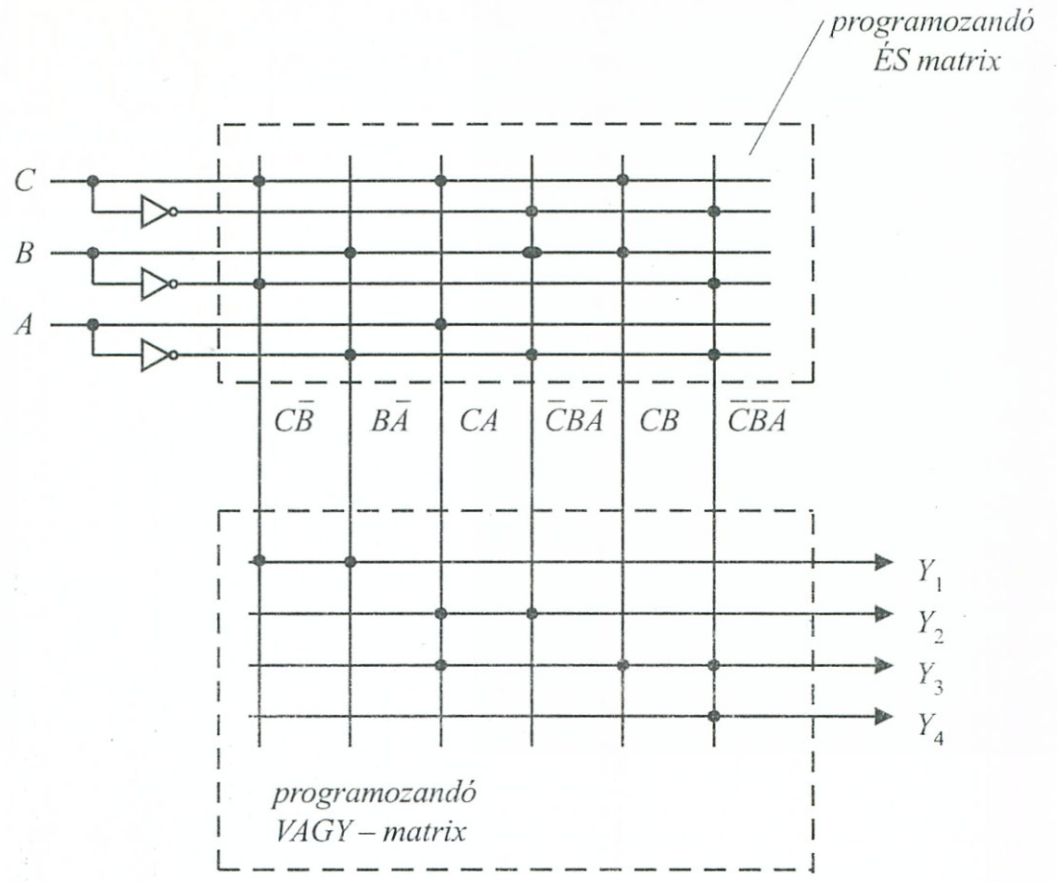
12-76. ábra *Átalakítások a G.6.17. feledat mmegoldásához*

minterm-ekből épül fel. A grafikus utat választva, a 12–76. ábrán megadtuk mind a négy függvény V–K tábláit, melyekről a benne foglalt minterm-ek index-száma leolvasható lesz.
Ezután már felrajzolható az esetünkre alkalmazott kapcsolómátrix-struktúra, melynél a *VAGY-mátrixot* kellett programoznunk és ROM-ba beégetnünk a 12–77. ábra szerint.

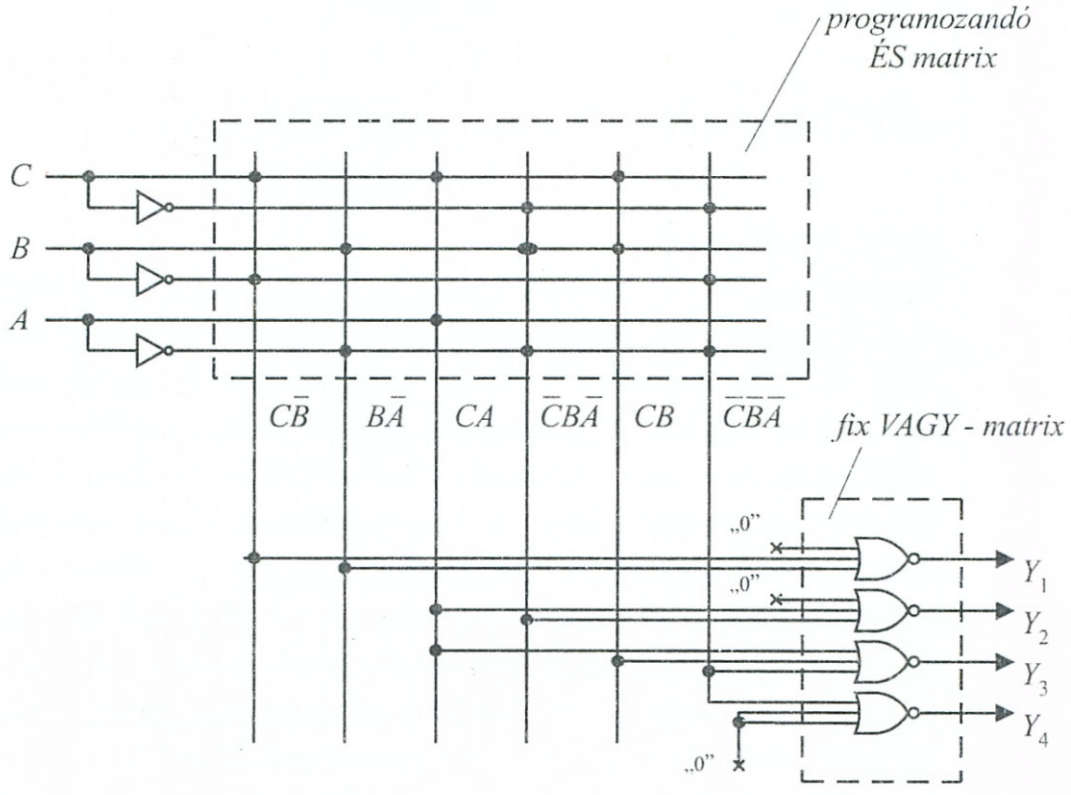


12-77. ábra *A G.6.17. feladat ROM realizációja*

b) *PLA realizáció:*
Ugyancsak a 6–43. ábrából kiindulva megállapíthatjuk, hogy itt a PLD mindkét mátrixa programozható. Ennek megfelelően előtérbe kerülhet a függvények minimál-alakjainak előállítása: legkevesebb számú és legrövidebb *implikánsok* előállítása. Az előző változatnál szereplő 12–76. ábra tábláiról már leolvasható, hogy a kiinduló alakok eleve minimál alakok voltak, így külön minimalizálásra nincs szükség. Ily módon a PLA realizációt ábrázoló 12–78. ábrabeli hálózat *ÉS-mátrixa* az ÉS-kapcsolatokat hordozó, összes előforduló implikánst, míg a *VAGY-mátrix* az egyes függvényekben szereplő aktuális implikánsok kapcsolatait fejezi ki.
A 12–78. ábrából látható, hogy az ÉS-mátrix oszlopainak száma csökkent a ROM realizációhoz viszonyítottan.



12-78. ábra *A G.6.17. feladat PLA realizációja*

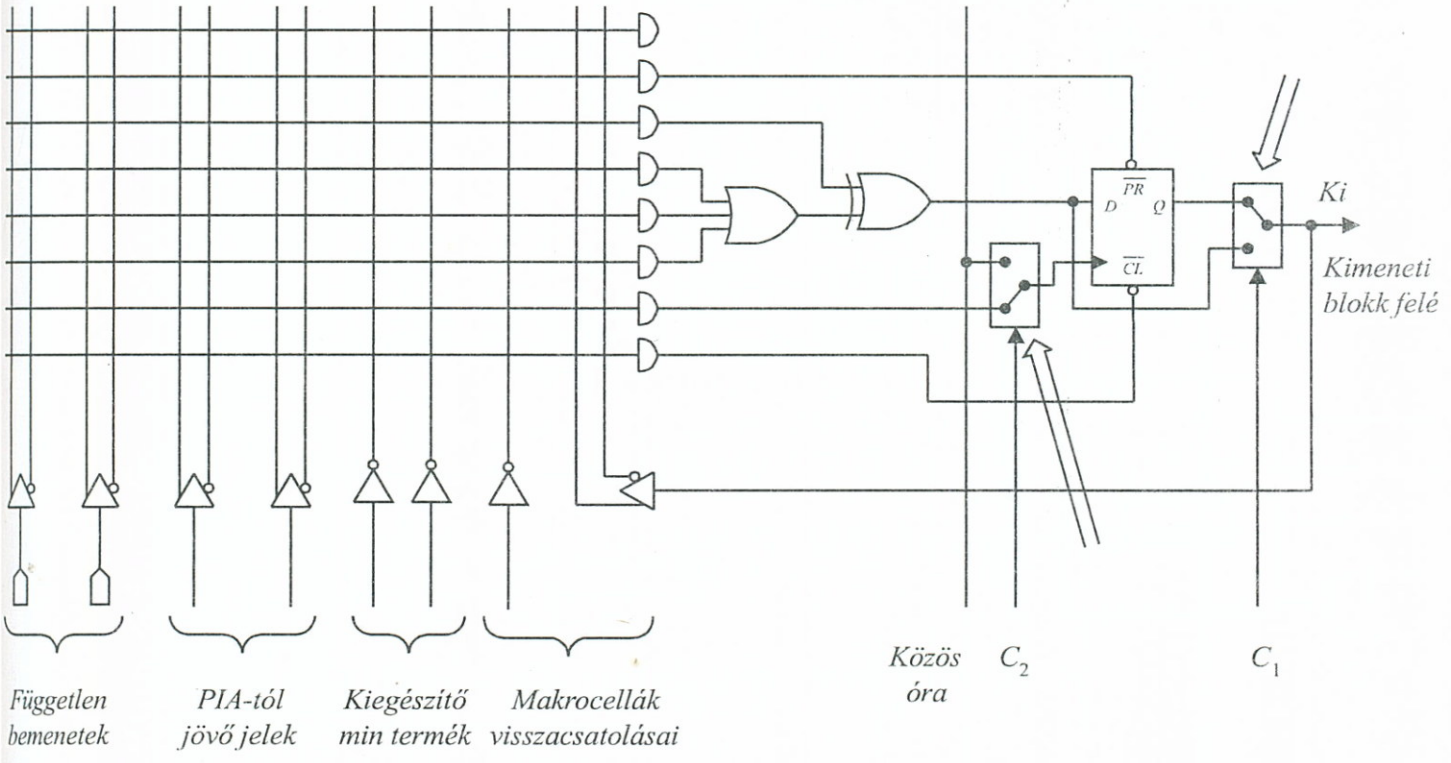


12-79. ábra *A G.6.17. feladat PAL realizációja*

c) *PAL realizáció:*
Ismét a 6–43. ábrából kiindulva látható, hogy itt a PLD *VAGY-mátrixa a fix* és az ÉS-mátrixot kell programozni. Az ÉS-mátrixban szerepelhetnek az a) változat mintermjei, vagy a b) változat implikánsai, attól függően, hogy melyik eredményez egyszerűbb megoldást. A PAL típusú PLD-áramkörök típusskálája rendkívül sokoldalú, így a katalógusokból számos kombináció kiválasztható. A 12–79. ábrán egy 3-bemenetű VAGY-okból álló PAL-ra alapozott megoldást rajzoltunk fel az implikánsos ÉS-mátrixból kiindulva:

F.6.18. A feladatmegoldást a 6–45a. ábrán látható MAX–LAB *Makrocella C_1, C_2 programozó kapcsolóinak megfelelő beállításával kell elvégezni. A 12–80. ábrán mégegyszer felrajzoltuk a Makrocellát:*

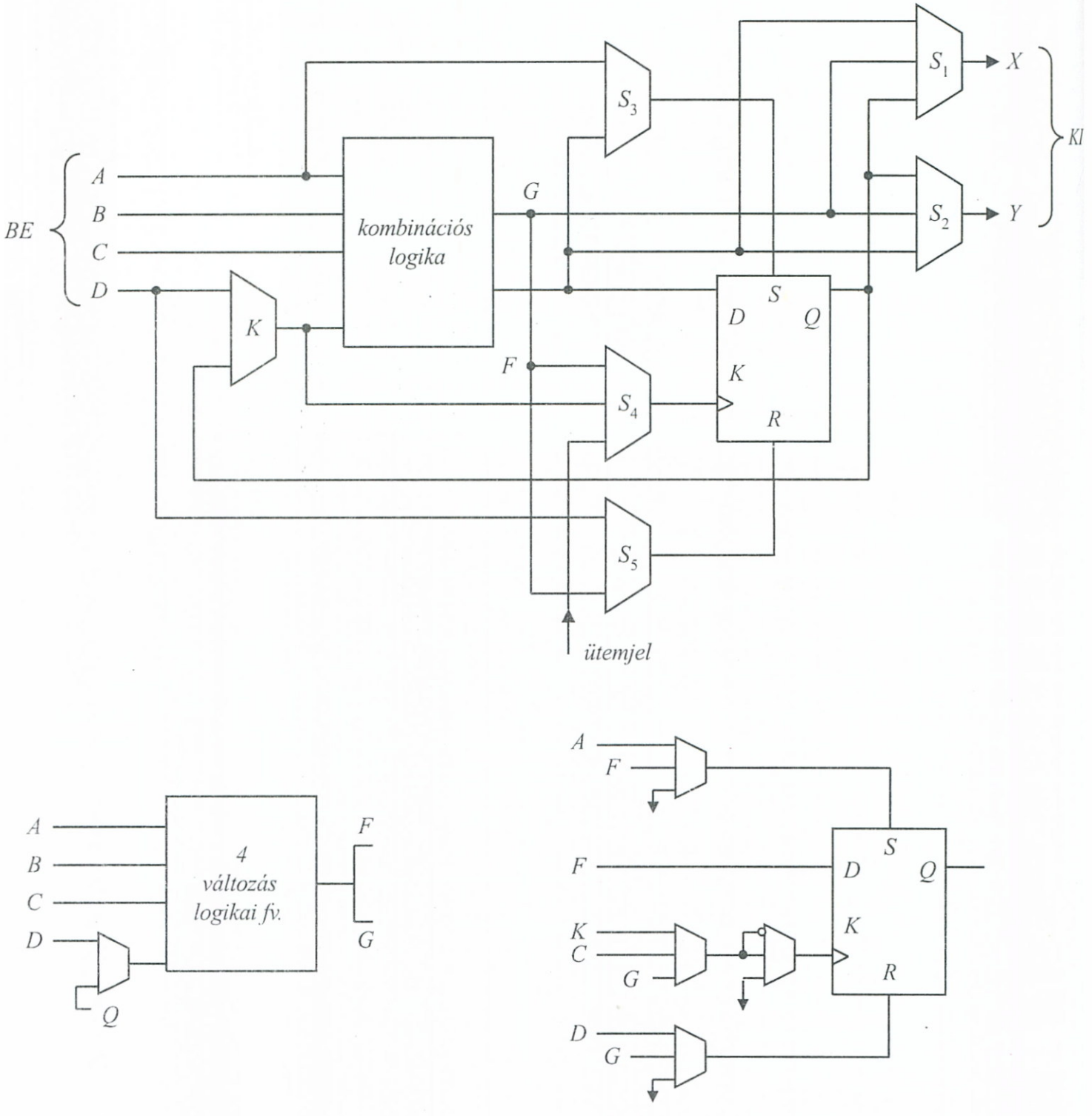
Mint látható, a C_1 kapcsoló *felső* állásba állítása után a tárolóelem kimenete közvetlenül a kimenetre csatlakozik, azaz a *tárolóelem aktivizálódik,* ugyanakkor a tárolót rövidzárral kiiktatni képes vezeték ág megszakad.



12-80. ábra *MAX-LAB makrocella programozása a G.6.18. feledat által előírt üzemmódra*

Az egyedi órajelet a C_2 kapcsoló alsó állásba állításával aktivizálhatjuk, ilyenkor a Makrocella a közös órajelről leválasztódik.

F.6.19. A feladatmegoldást a 6–48. ábrán bemutatott XILINX CLB eszköznek a kijelölt működési módra történő felprogramo-



12-81. ábra *XILINX-CLB beállítása a G.6.19. feladathoz*

zásával végezhetjük el az S_i és K kapcsolók értelemszerű beállításával. A 12–81. ábrán mégegyszer felrajzoltuk a CLB-t a példához aktuális részek kiemelésével.

Mint látható, a D-tároló kimenetre kapcsolását az S_1, S_2 kapcsolók biztosítják. Az S_4 az órajelre, S_3, S_5 a SET, ill. RESET beállításra vannak befolyással. A K kapcsoló teszi lehetővé a visszacsatoló ág kívánt aktivizálását.

12.7.1. Gyakorló feladatok a 7. „Kombinációs hálózatok kialakítási kérdései" c. fejezethez

*G.7.1. Minimalizáljuk a következő, részben meghatározott függvénnyel jellemzett kombinációs hálózatot grafikusan.

$$Y_K^6 = \sum^{6} (0, 1^h, 4, 5^h, 25^h, 36, 37, 38, 39, 44, 45, 46, 47^h, 48)$$

(a p^h-term-ek határozatlanok).

*G.7.2. Minimalizáljuk a G.7.1. példabeli hálózatot QUINE–Mc CLUSKEY táblázatos módszerrel.

*G.7.3. Tervezzünk grafikus úton közös részhálózatos kombinációs hálózatot a megadott kimeneti függvényekkel:

$$Y_1 = \sum^{4} (0^h, 2, 8, 10^h, 12^h, 14)$$

$$Y_2 = \sum^{4} (7^h, 8^h, 10, 12, 14, 15)$$

$$Y_3 = \sum^{4} (0, 2^h, 7, 12, 14^h, 15^h)$$

*G.7.4. NOR-kapus hálózattal alakítsuk ki – lehetőleg közös részhálózatos megoldásban – az alábbi függvényrendszerrel jellemzett kombinációs hálózatot:

$$Y_\alpha^4 = \prod^{4} (3, 4, 7, 12, 14)$$

$$Y_\beta^4 = \prod^{4} (1, 2, 3, 4, 12, 13, 15)$$

$$Y_\gamma^4 = \prod^{4} (2, 3, 4, 11, 12, 13)$$

*G.7.5. Kíséreljük meg diszjunkt dekompozíciós hálózattal realizálni az alábbi függvénnyel jellemzett kombinációs hálózatot:

$$Y = \sum^{5} (3, 4, 5, 7, 9, 10^h, 13^h, 14^h, 15, 17, 18^h, 21, 22, 23^h, 27^h, 28, 29, 30^h, 31^h)$$

*G.7.6. Tervezzük meg a következő függvénnyel megadott kombinációs hálózatot – ha lehet – dekompozíció alkalmazásával.

$$Y = \sum^{4} (0, 2, 5, 7, 9, 10, 12, 15)$$

*G.7.7. Hasonlítsuk össze a G.7.6. példa eredményét a kétszintes ÉS–VAGY realizációban kialakított hálózattal.

*G.7.8. Vizsgáljuk meg a 7–15. ábra SSI realizációs változatait olyan szempontból, hogy melyik igényel legkevesebb IC tokot a megvalósításhoz. Tételezzük fel, hogy az SSI IC tokokba beépíthető kapuáramkörök száma a 12–82. ábra táblázatával van jellemezve.

Kapubemenet száma	*Beépített elemek száma*
1 – *INVERTER*	6
2) *ÉS, VAGY,*	4
3) *NAND, NOR,*	3
4) *ANTIVALENCIA,*	2
5) *EKVIVALENCIA*	1

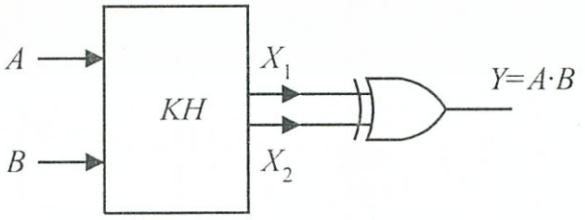
12-82. ábra *SSI IC tokokban szereplő kapuk száma*

*G.7.9. Mit kell a 12–83. ábrán látható KH-nak tartalmaznia ahhoz, hogy az *Y*-kimenet teljesüljön?

*G.7.10. Oldjuk meg a következő logikai egyenlőtlenség rendszert:

$$\overline{B}(\overline{X}+A) \geq \overline{X}A + \overline{B}\,\overline{A}$$

$$\overline{A} \leq \overline{B}\,\overline{A} + X$$



12-83. ábra *Előtét KH áramkör tervezése*

*G.7.11. Egy betöréssel kapcsolatosan nyomozás folyik. Kinn járt már egy nyomozó és volt helyszíni szemle is. A begyűjtött információkat rendezve próbáljuk kideríteni mi az igazság.
A *nyomozó* a következő tényállást rögzítette:
a) Ha a tettes férfi, akkor kistermetű.
b) Ha kistermetű, az ablakon mászott be.
c) Férfi a tettes, vagy legalábbis férfi ruhát hordott.
d) Ha férfi ruhát hordott, akkor feltéve, hogy hiteles a „szemtanú" vallomása, az ablakon mászott be.
A *helyszíni szemle* utólag kiderítette, hogy:
e) Nem az ablakon mászott be.

12.7.2. Feladatmegoldások a 7. fejezethez

F.7.1. A grafikus minimalizáláshoz fel kellene rajzolnunk egy 6-változós minterm-táblát 64 mezejével. Ez a 2–16. ábrák további kiterjesztésével történhetne. További problémákkal kerülnénk szembe a legnagyobb tömbök kialakítása során, mivel a logikailag szomszédos tömbrészek grafikusan (látszólag) nem mindig lennének szomszédosak, így kezelésük rendkívül nehézkessé, és bizonyos mértékig „szubjektívvá" válna a nehezebb áttekinthetőség miatt. Ezért ilyen esetekben – mint a 7.1. pontban már említettük – a táblázatos módszerekhez fordulunk. Ezért a probléma megoldása érdekében oldjuk meg a G.7.2. feladatot, melynek kiindulása ugyanaz a függvény, mint amit a G.7.1.-ben felírtunk.

F.7.2. A feladatmegoldásnál – mivel a kiindulás határozatlan termeket is tartalmaz – a 7.1.2. pontban összefoglaltakat is figyelembe kell venni. Továbbá, ha a változók száma nagy, a terjengős értéktáblázatok kezelése nehézkessé válik. Ez esetben célszerű a QUINE–Mc CLUSKEY módszer, ún.

„index-számos" változatát alkalmazni. A következő példában ezt mutatjuk be, megjegyezzük, hogy az index-számos változat mind részben határozott, mind teljesen meghatározott esetre alkalmazható.

Súly	Index	F E D C B A							
0	0	0 0 0 0 0 0	✓	0, 1 (1)	✓	0, 1, 4, 5 (1, 4)	b	36, 37, 38, 39, 44, 45, 46, 47 (1, 2, 8)	f
1	1^h	0 0 0 0 0 1	✓	0, 4 (4)	✓	0, 4, 1, 5 (4, 1)	c	36, 37, 44, 45, 38, 39, 45, 47 (1, 8, 2)	g
	4	0 0 0 1 0 0	✓	1, 5 (4)	✓	4, 5, 36, 37 (1, 32)	d	36, 38, 37, 39, 44, 46, 45, 47 (2, 1, 8)	h
2	5^h	0 0 0 1 0 1	✓	4, 5 (1)	✓	4, 36, 5, 37 (32, 1)	e	36, 38, 44, 46, 37, 39, 45, 47 (2, 8, 1)	i
	36	1 0 0 1 0 0	✓	4, 36 (32)	✓	36, 37, 38, 39 (1, 2)	✓	36, 44, 37, 45, 38, 46, 39, 47 (8, 1, 2)	j
	48	1 1 0 0 0 0	a	5, 37 (32)	✓	36, 37, 44, 45 (1, 8)	✓	36, 44, 38, 46, 37, 45, 39, 47 (8, 2, 1)	k
3	25^h	0 1 1 0 0 1	✓	36, 37 (1)	✓	36, 38, 37, 39 (2, 1)	✓		
	37	1 0 0 1 0 1	✓	36, 38 (2)	✓	36, 38, 44, 46 (2, 8)	✓		
	38	1 0 0 1 1 0	✓	36, 44 (8)	✓	36, 44, 37, 45 (8, 1)	✓		
	44	1 0 1 1 0 0	✓	37, 39 (2)	✓	36, 44, 38, 46 (8, 2)	✓		
4	39	1 0 0 1 1 1	✓	37, 45 (8)	✓	36, 39, 45, 47 (2, 8)	✓		
	45	1 0 1 1 0 1	✓	38, 39 (1)	✓	44, 46, 45, 47 (2, 1)	✓		
	46	1 0 1 1 1 0	✓	38, 46 (8)	✓	37, 45, 39, 47 (8, 2)	✓		
5	47^h	1 0 1 1 1 1	✓	44, 45 (1)	✓	38, 46, 39, 47 (8, 1)	✓		
				44, 46 (2)	✓	38, 39, 46, 47 (1, 8)	✓		
				39, 47 (8)	✓	44, 45, 46, 47 (1, 2)	✓		
				45, 47 (2)	✓				
				46, 47 (1)	✓				
①				②		③		④	

12-84. ábra *DON'T CARE eseteket is tartalmazó G./.1. példa táblázatos megoldása, összehasonlító táblázat.*

I. lépés: Összehasonlító táblázat első rovata.

Az összehasonlító táblázat első rovata hasonlóan épül fel, mint a 7.1.1. pont 7.1. példájánál, de itt, mivel don't care függvényről van szó, a határozatlan term-eket is felsoroljuk (12–84. ábra ①-es rovat).

II. lépés: Prim-implikánsok keresése.

Az index-számos változat e lépésnél annyiban tér el az eredetitől, hogy az összehasonlítást nem a 0, 1 értékkel kifejezett term-ekkel, hanem közvetlenül a term-ek decimális index-számainak felhasználásával végezzük el. Itt is érvényes az az előző példában alkalmazott szabály, hogy két olyan term vonatható be az egyszerűsítésbe, melyek egymástól D = 1 Hamming-távolságban különböznek, azaz „összehasonlíthatók". Az index-számos módszernél az összehasonlíthatóság a következő szabállyal fogalmazható meg:

Két, egymástól egy súlyértékkel eltérő term akkor összehasonlítható, ha a term-ek decimális index-számainak különbsége kettőnek egész számú hatványa, és a nagyobb indexű egyben nagyobb súlyú is.

Ha a termek változóihoz hátulról növekvő sorrendben a kettes számrendszer helyértékeit rendeljük, akkor az előző index-szám különbség ugyancsak kettő hatványaként felírva, kijelöli azt a változót is, melyet az összehasonlítás eredményeképpen eliminálhattunk. Például az ① rovat 3-as súlyú „37" term-jét összehasonlítva, a 2-es súlyú „5" term-mel a következőket olvashatjuk le:

		2^5	2^4	2^3	2^2	2^1	2^0
Súly	Index	F	E	D	C	B	A
2	5	0	0	0	1	0	1
3	37	1	0	0	1	0	1

– 37 nagyobb „súlyú" eggyel, mint 5
– A különbség: $37 - 5 = 32 = 2^5$
– Az eredmény:

	F	E	D	C	B	A
(37,5)	–	0	0	1	0	1

Tehát „*F*"-et (a 2^5 helyértékű változót) elimin——áltuk.

Hasonló módon végezve és megjelölve a 12–84. ábra ① rovatában a lehetséges összehasonlításokat a ② rovatot nyerjük. Példánknál az ① rovatban nem lelt párra a „48"-as és „25^{h}"-ös term. Ezek közül a „25^{h}"-ös határozatlan, viszont a „48"-as határozott, ezért prim-implikánsnak tekintjük, és „a" betűvel meg is jelöljük.

A ② rovat első oszlopában az összehasonlított term-párok vesszővel elválasztott index-számai szerepelnek súlyaik szerinti csoportokban, utánuk zárójelben a különbség.

Itt is folytatni kell az összehasonlításokat. Csak azokat a sorokat hasonlítjuk össze, ahol egyrészt a korábbi index-szám különbség ugyanakkora (ugyanis ezeknél ugyanazt a változót eliminálták), másrészt ha mindkét sornál az első szám (legkisebb index) különbsége kettő egész számú hatványa és a kisebb indexű sor van a kisebb súlyú csoportban.

Az így nyert ③ rovatnál konzekvensen alkalmazva az elmondottakat, nyerhetjük a ④-es és esetleg magasabb számú rovatokat. Jelen példánál a ④ rovat minden sora prim-implikáns, így ez az utolsó rovat.

III. lépés: Prim-implikáns táblázat.

A betűkkel megjelölt prim-implikánsokat egy táblázatba rendezzük, melynek oszlop-fejrovataiban csak a határozott értékű term-eket soroljuk fel. A term-ek és implikánsok kapcsolatát *-gal jelöljük. Példánknál a 12–85a. ábra képezi a prim-implikáns táblázatot.

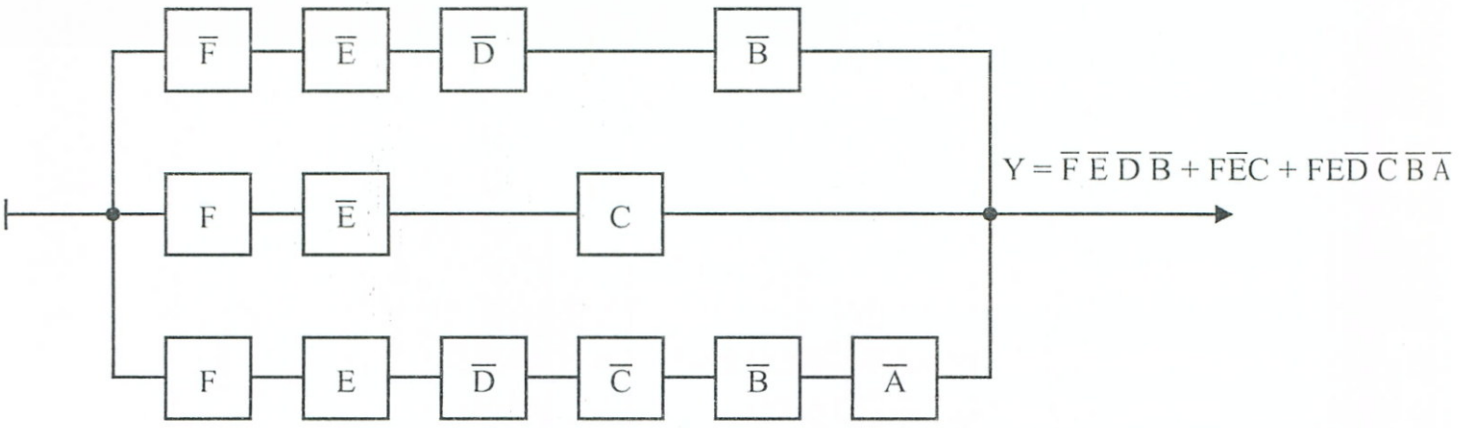
IV. lépés: Irredundáns megoldások kiválasztása.

A szelekciós függvény igen leegyszerűsödik, miután a tényezők ismétlődnek. Az egyetlen lényeges prim-implikáns „*a*", mivel ennek kihagyása nem lényegesen egyszerűsíti a számítást, bennhagyjuk a szelekciós függvényben, mely a $b = c,\ d = e,\ f = g = h = i = j = k$ egyenlőségek felismerése után az alábbi lesz:

$$b\cdot(b+d)\cdot(d+f)^2\cdot f^5\cdot a = b\cdot f\cdot a$$

PRIM IMPLIKÁNSOK			0	4	36	37	38	39	44	45	46	48
48		a										⊛
0, 1, 4, 5	(1, 4)	b	*	*								
0, 4, 1, 5	(4, 1)	c	*	*								
4, 5, 36, 37	(1, 32)	d		*	*	*						
4, 36, 5, 37	(32, 1)	e		*	*	*						
36, 37, 38, 39, 44, 45, 46, 47	(1, 2, 8)	f			*	*	*	*	*	*	*	
36, 37, 44, 45, 38, 39, 46, 47	(1, 8, 2)	g			*	*	*	*	*	*	*	
36, 38, 37, 39, 44, 46, 45, 47	(2, 1, 8)	h			*	*	*	*	*	*	*	
36, 38, 44, 46, 37, 39, 45, 47	(2, 8, 1)	i			*	*	*	*	*	*	*	
36, 44, 37, 45, 38, 46, 39, 47	(8, 1, 2)	j			*	*	*	*	*	*	*	
36, 44, 38, 46, 37, 45, 39, 47	(8, 2, 1)	k			*	*	*	*	*	*	*	

a)



b)

12-85. ábra *DON'T CARE eseteket tartalmazó példa G.7.1. példa prímimplikáns táblázata és áramlogikás realizációja*

A megmaradt „*b f a*" *visszatranszformálása* a következők szerint történik:

Írjuk fel a prim-implikánsban szereplő legkisebb index-számú term-et, majd hagyjuk el belőle azokat a változókat, melyek helyértékét a term-nél zárójelben feltüntetett maradék számok kijelölik.

Példánknál:

		2^5	2^4	2^3	2^2	2^1	2^0	
		32	16	8	4	2	1	
b	E_0^6	$\overline{F}$	$\overline{E}$	$\overline{D}$	$\overline{C}$	$\overline{B}$	$\overline{A}$	(1,4)
f	E_{36}^6	F	$\overline{E}$	$\overline{D}$	C	$\overline{B}$	$\overline{A}$	(1,2,8)
a	E_{48}^6	F	E	$\overline{D}$	$\overline{C}$	$\overline{B}$	$\overline{A}$	semmi

A minimál alak:

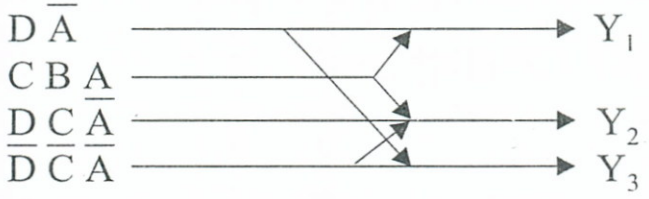
$$Y_{K\,\min} = \overline{F}\,\overline{E}\,\overline{D}\,\overline{B} + F\,\overline{E}\,C + F\,E\,\overline{D}\,\overline{C}\,\overline{B}\,\overline{A}$$

V. lépés: Kombinációs hálózat felrajzolása. Realizációs áramkörnek egy áramlogikás kapcsolóhálózatot választottunk, mely a 12–85b. ábrán látható.

F.7.3. Hasonló feladatot már a 7.2. példa kapcsán megoldottunk, de a jelenlegi annál kissé bonyolultabb, mivel itt 3 kimenetű a hálózat.
A függvények és a közösített V–K-táblák a 12–86a. ábrán láthatók A táblákról leolvasott prim-implikánsokat a 12–86b. táblázatban foglaltuk össze. A szelekciós függvény megoldása után (csak a legrövidebb tagokat felírva) a következőket kapjuk:

$$S = \ldots\ nijg + nijl + nijk + nijo + \ldots$$

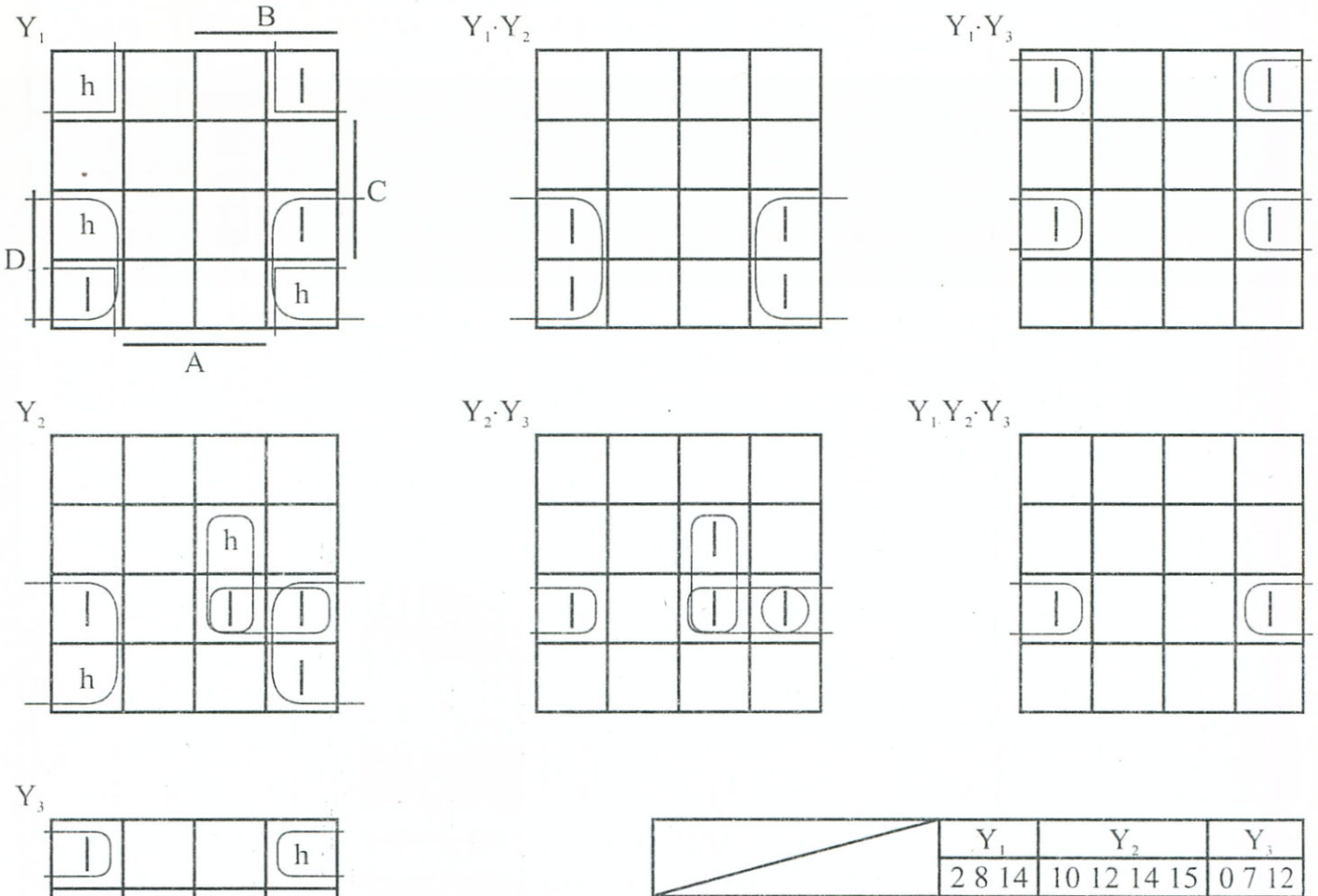
és mivel a táblázatból látható: $g = l = k = o$, írhatjuk:



Ennek alapján már a logikai vázlat felrajzolható lenne, ha felhasználjuk az említett 7.2. példa tapasztalatait.

F.7.4. A táblázatos módszer példánkra történő alkalmazása előtt célszerű néhány további meggondolást tennünk:

Az 5.2. pontban és az előző példánál grafikus módszerrel már tárgyaltuk ezt a problémát. A cél itt is hasonló: többszörösen kihasználható részhálózatok képzése közös implikánsok fellelése révén, melyekkel a végleges hálózat – esetleg – egyszerűbbé tehető.



a)

b)

			Y_1	Y_2	Y_3
			2 8 14	10 12 14 15	0 7 12
Y_1	$D\overline{A}$	a	x x		
	$\overline{C}\,\overline{A}$	b	x x		
Y_2	$D\overline{A}$	c		x x x	
	DCA	d		x x	
	DCA	e		x	
Y_3	$\overline{D}\,\overline{C}\,\overline{A}$	f			x
	$DC\overline{A}$	g			x
	CBA	h			x
$Y_1 \cdot Y_2$	$D\overline{A}$	i	x x	x x x	
$Y_1 \cdot Y_3$	$D\overline{C}\,\overline{A}$	j	x		x
	$DC\overline{A}$	k	x		x
$Y_2 \cdot Y_3$	$DC\overline{A}$	l		x x	x
	DCB	m		x x	
	CBA	n		x	x
$Y_1Y_2Y_3$	$DC\overline{A}$	o	x	x x	x

12-86. ábra *Közös részhálózat előállítása 3 kimenetnél*

A QUINE–Mc CLUSKEY módszert, néhány további előírás beiktatásával, alkalmassá tehetjük közös implikánsok keresésére. Az így módosított eljárás a következőképpen alakul.

I. lépés: Összehasonlító táblázat első rovata.

Az ① rovatban *valamennyi kimenethez tartozó* logikai függvény term-jeit felsoroljuk „súlyuk" és index-számuk szerinti csoportosításban. A term-ek mellett feltüntetjük annak a kimeneti függvénynek betűjelét is, melyben szerepel.

II. lépés: Prim-implikánsok keresése.

Az egyes term-ek összehasonlítási szabályai itt a következők:

- Az egyes rovatokban *azonos* csoportokban szereplő sorok akkor hasonlíthatók össze, ha index-számaik és a zárójeles különbségek megegyeznek, de *különböző* kimeneti függvényekhez tartoznak. Ezek az összevonások adják a közös implikánsokat.
- Az egyes rovatok két *szomszédos* „súlycsoportjában" előforduló sorok akkor hasonlíthatók össze, ha *ugyanahhoz* a kimeneti függvényhez tartoznak, a legkisebb indexeik különbsége kettő egész számú hatványa, és a nagyobbik index nagyobb súlycsoportban van, továbbá zárójeles különbségeik megegyeznek.

III. lépés: Prim-implikáns táblázat.

Ez hasonló felépítésű mint az előző táblázatok, az eltérés az, hogy a term-eket a kimeneti függvények szerint csoportosítjuk.

IV. lépés: Irredundáns megoldások kiválasztása.

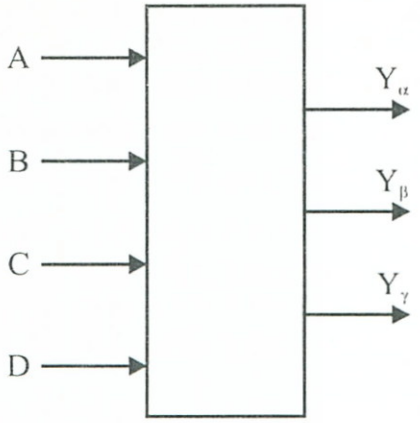
Ezeket vagy közvetlenül kiolvassuk a prim-implikáns táblázatból, vagy a szelekciós függvény segítségével kaphatjuk meg a prim-implikánsoknak szimultán függvényenként, ill. függvénycsoportonként történő *szétválogatása* után.

V. lépés: Realizált hálózat felrajzolása.

Ezután már réterhetünk a G.7.4. példa konkrét megoldására:

I. lépés: Összehasonlító táblázat első rovata.

A függvény VAGY-term-ekkel, konjunktív kanonikus alakban van megadva, de ez a módszerben semmi változást nem



a)

	D C B A			
1	0 0 0 1	1	β	✓
	0 0 1 0	2	β	✓
	0 0 1 0	2	γ	✓
	0 1 0 0	4	α	✓
	0 1 0 0	4	β	✓
	0 1 0 0	4	γ	✓
2	0 0 1 1	3	α	✓
	0 0 1 1	3	β	✓
	0 0 1 1	3	γ	✓
	1 1 0 0	12	α	✓
	1 1 0 0	12	β	✓
	1 1 0 0	12	γ	✓
3	0 1 1 1	7	α	✓
	1 0 1 1	11	γ	✓
	1 1 0 1	13	β	✓
	1 1 0 1	13	γ	✓
	1 1 1 0	14	α	✓
4	1 1 1 1	15	β	✓

①

1	2	0	β, γ	✓
	4	0	α, β	✓
	4	0	α, γ	✓
	4	0	β, γ	✓
	1, 3	2	β	a
	2, 3	1	β	✓
	2,3	1	γ	✓
	4, 12	8	α	✓
	4, 12	8	β	✓
	4, 12	8	γ	✓
2	3	0	α, β	✓
	3	0	α, γ	✓
	3	0	β, γ	✓
	12	0	α, β	✓
	12	0	α, γ	✓
	12	0	β, γ	✓
	3, 7	4	α	b
	3, 11	8	γ	c
	12, 14	2	α	d
	12, 13	1	β	✓
	12, 13	1	γ	✓
	13	0	β, γ	✓
	13, 15	2	β	e

②

1	4	0	α, β, γ	✓
	2, 3	1	β, γ	f
	4, 12	8	α, β, γ	✓
	4, 12	8	α, β	✓
	4, 12	8	α, γ	✓
	4, 12	8	β, γ	✓
2	3	0	α, β, γ	g
	12	0	α, β, γ	✓
	12, 13	1	β, γ	h

③

1	4, 12	8	α, β, γ	i

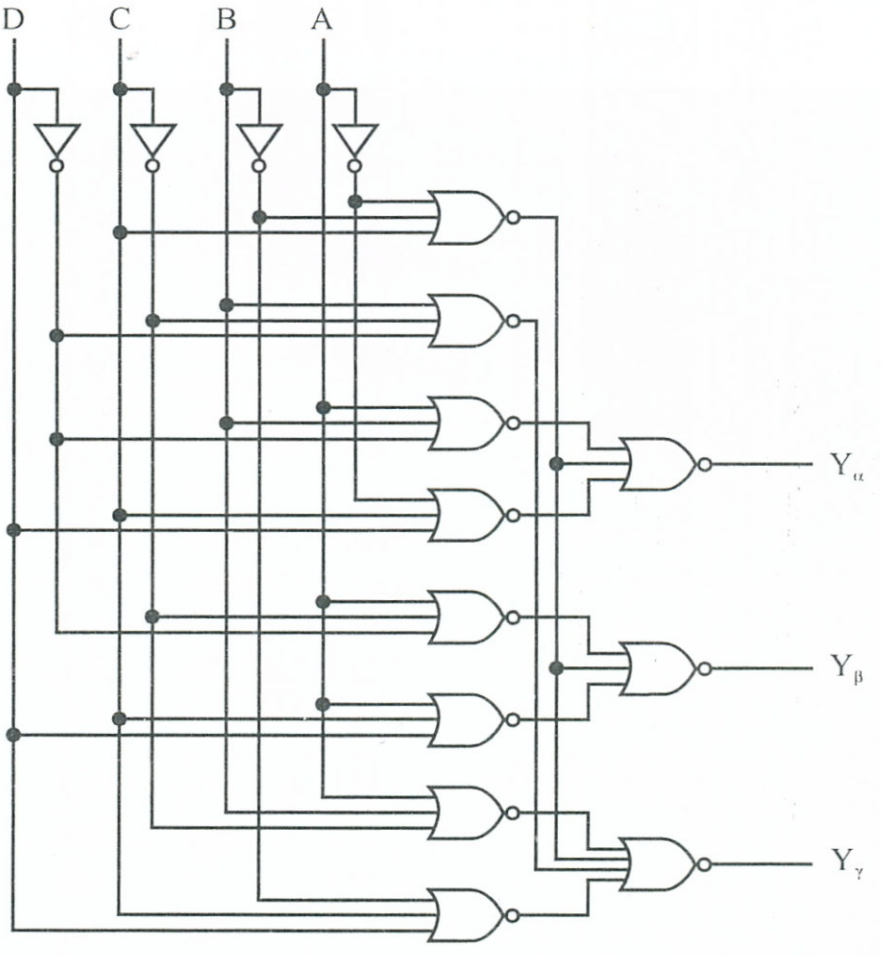
④

b)

12-87. ábra *Közös részhálózatok keresése*

				Y_α					Y_β							Y_γ					
				3	4	7	12	14	1	2	3	4	12	13	15	2	3	4	11	12	13
1, 3	2	β	a						*		*										
3, 7	4	α	b	*		*															
3, 11	8	γ	c														*		*		
12, 14	2	α	d				*	*													
13, 15	2	β	e											*	*						
2, 3	1	β, γ	f							*	*					*	*				
3	0	α, β, γ	g	*							*						*				
12, 13	1	β, γ	h										*	*						*	*
4, 12	8	α, β, γ	i		*		*					*	*					*		*	

a)



b)

12-88. ábra *Implikáns táblázat és realizáció a G.7.4. példához*

okoz. Az ① rovat, az előzők értelmében a 12–87b. ábra szerint alakul.

II. lépés: Prim-implikánsok keresése.

Az előzőleg összefoglalt szabályoknak megfelelő összehasonlítások eredményei a 12–87b. ábra ②, ③, ④ oszlopaiban szerepelnek.

III. lépés: Prim-implikáns táblázat.

Ez a 12–88a. ábrán látható.

IV. lépés: Irredundáns megoldások kiválasztása.

A 12–88a. táblázatban „g" kivételével minden prim-implikáns lényeges és lefedik az összes term-oszlopot, ezért „g" elhagyható:
A minimalizált függvények a következők:

$F_\alpha^4 : b,d,i \qquad Y_\alpha = (\overline{A}+C+D)\cdot(A+B+\overline{D})\cdot(B+\overline{C}+\overline{D})$

$F_\beta^4 : a,e,f,i \qquad Y_\beta = (\overline{A}+\overline{B}+D)\cdot(A+B+D)\cdot(\overline{A}+\overline{B}+C)\cdot(A+\overline{C}+\overline{D})$

$F_\gamma^4 : c,f,h,i \qquad Y_\gamma = (\overline{B}+C+D)\cdot(\overline{A}+\overline{B}+C)\cdot(A+B+\overline{C})\cdot(B+\overline{C}+\overline{D})$

V. lépés: Logikai vázlat.

A kívánt NOR-kapus realizáláshoz Y_α, Y_β, Y_γ függvényeket a SHEFFER–PEIRCE algebra 24b. azonosságának felhasználásával homogén Peirce műveleti rendszerbe transzformálhatjuk és így kapjuk a 12–88b. ábrán látható logikai vázlatot.

F.7.5. A megoldást a következő lépésekbe foglalhatjuk:

I. lépés: A 7–11a. és b. ábrán látható ötváltozós dekompozíciós táblákból kiindulva, a határozatlan term-eket vízszintes áthúzással megkülönböztetve és gondolatban minden táblára elvégezve a bekarikázásokat is, végül a 12–89a. ábra szerinti tábla-változaton találtunk dekompozíciós lehetőséget.

II. lépés: Az egzisztencia-vizsgálat valójában már megtörtént, a határozatlan term-ek 12–89a. táblán szereplő (dekompozíció elősegítésének megfelelő) kijelölésénél. Mint

látjuk $\nu = 2$, azaz csak kétféle oszlop létezik. A partíciók: $P_\alpha = \{A, B, D\}$, $P_\beta = \{C, E\}$

III. lépés: Jelen példánál nem érdekes.

IV. lépés: F_S szubfüggvény meghatározása. A 12-89a. táblán az első sor $F_{\mathrm{I}}^3 = 0$, a harmadik $F_{\mathrm{III}}^3 = 1$ függvényfaktort jelöl $\{C, E\}$ változó kombinációk mellett, F_S-nek választható vagy a 2. vagy a 4. sorhoz tartozó, egymásnak komplemensét képező valamelyik $F_j^3(A, B, D)$. Választva F_{II}^3-at (nyíllal megjelölt sor) kapjuk:

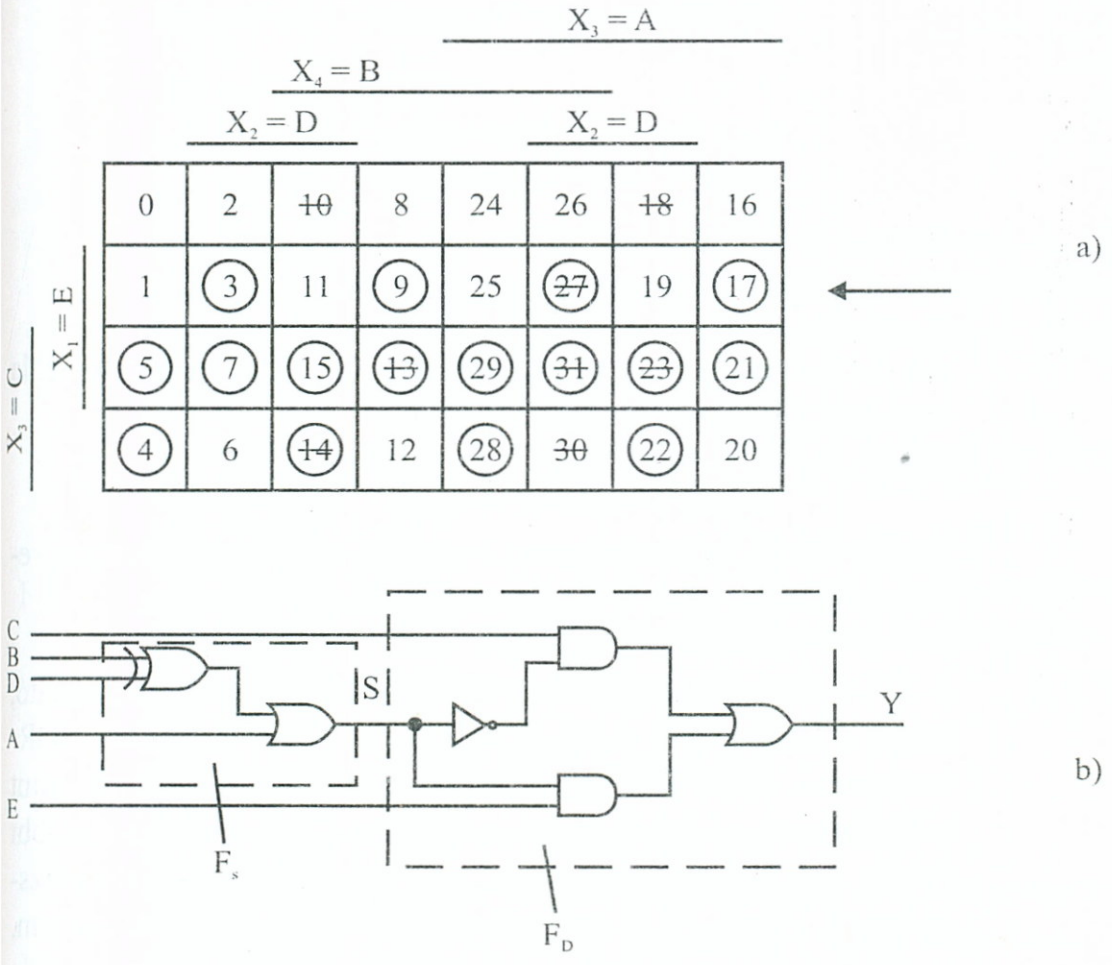
$$S = F_{\mathrm{II}}^3(A, B, D) = \overline{A}\,\overline{B}\,D + \overline{A}\,B\,\overline{D} + A\,B\,D + A\,\overline{B}\,\overline{D}$$

Ez algebrailag átírható a következő alakba:

$$S = A \oplus (B \oplus D)$$

V. lépés: A dekompozíciós alak:

$$Y = F_D(S, C, E) = S\cdot\overline{C}\cdot E + C\cdot E + \overline{S}\cdot C\cdot\overline{E} = \overline{S}\cdot C + S\cdot E$$



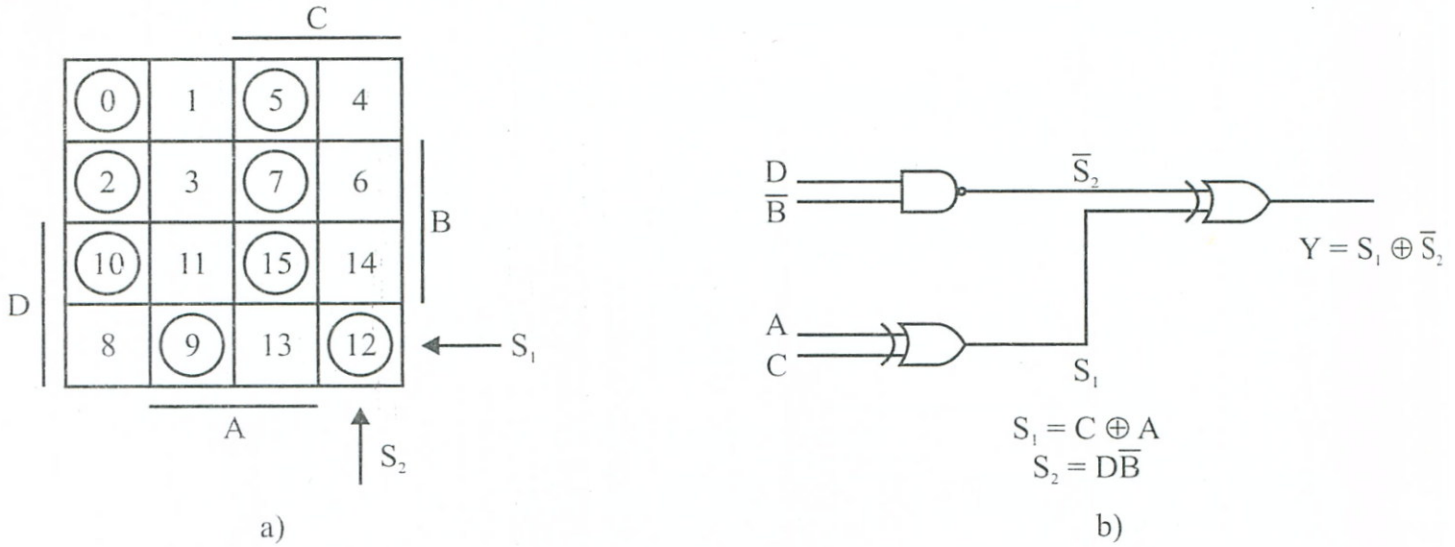
12-89. ábra *Ötváltozós dekompozíciós hálózat*

vagy S-be behelyettesítve:

$$Y = \overline{[A \oplus (B \oplus D)]}C + [A \oplus (B \oplus D)]E$$

Mint látható, itt rendkívül tömör alakot kaptunk, a dekompozíció nagyon hatékony volt. A realizációs logikai vázlat a 12–89b. ábrán látható.

F.7.6. A 4-változós dekompozíciós táblák kitöltése után a 12–90a. ábra tábláján két irányból is képezhettünk szubfüggvényt. A kapott rendkívül egyszerű hálózat a 12–90b. ábrán látható. Érdemes összehasonlításul még felrajzolni a normál tömbösítéssel kapott megoldást is.



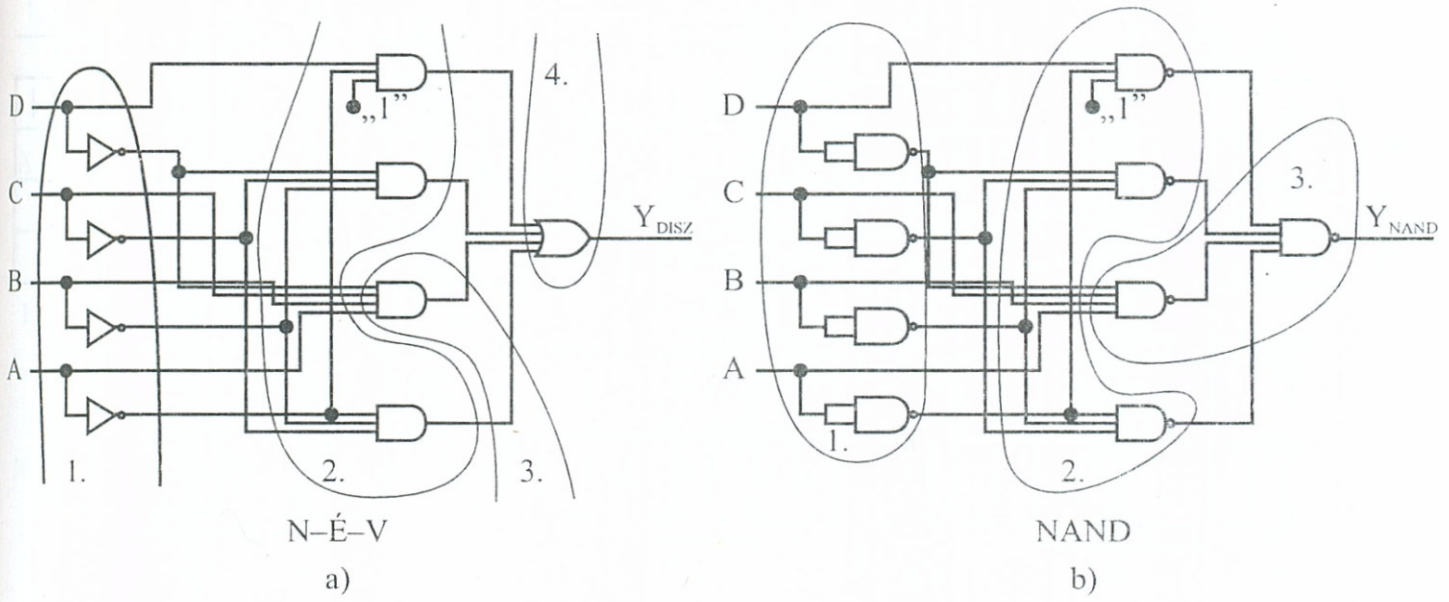
12-90. ábra *Összetett diszjunkt dekompozíció*

F.7.7. Az összehasonlításhoz képezzük a 12–90a. minterm-tábla legnagyobb tömbös minimál alakját:

$$Y = \overline{D}\,\overline{C}\,\overline{A} + B\,\overline{C}\,\overline{A} + D\,C\,A + B\,C\,A + D\,\overline{C}\,\overline{B}\,A + D\,C\,\overline{B}\,\overline{A}$$

Mint látható, ez lényegesen bonyolultabb megoldáshoz vezet, mint a 10–90b. ábra dekompozíciós változata.

F.7.8. A 7–15. ábra SSI realizációs változatait megvizsgálva látható, hogy a konjunktív alakból származó N–V–É, NOR–NOR-változatok ennél az esetnél lényegesebben terjengősek, mint a diszjunktív alakból származó esetek. Ezért csak az utóbbi kettőt kell a legkevesebb IC-tok szempontjából megvizsgálnunk. Az N–É–V alaknál eleve többféle kapu-típus van, melyek kihasználtsági mértéküktől függetlenül más-más IC-tokot igényelnek. A 12–91. ábrán újra felrajzoltuk a kér-



12-91. ábra *Az egy IC tokba gyűjthető kapu-csoportok*

déses változatokat és (a 12–82. ábra „katalógusát" figyelembevéve) elhatároltuk az ún. „amőba-vázlattal" az egy-egy IC-tokba gyűjthető kapu-csoportokat.

A 12–91a. ábrabeli esetnél – mint látható – 4 db IC-t kellett használni és kihasználatlan maradt az

1.-nél: 2 inverter
3.-nál: 1 4-bemenetű ÉS-kapu
4.-nél: 1 4-bemenetű VAGY-kapu.

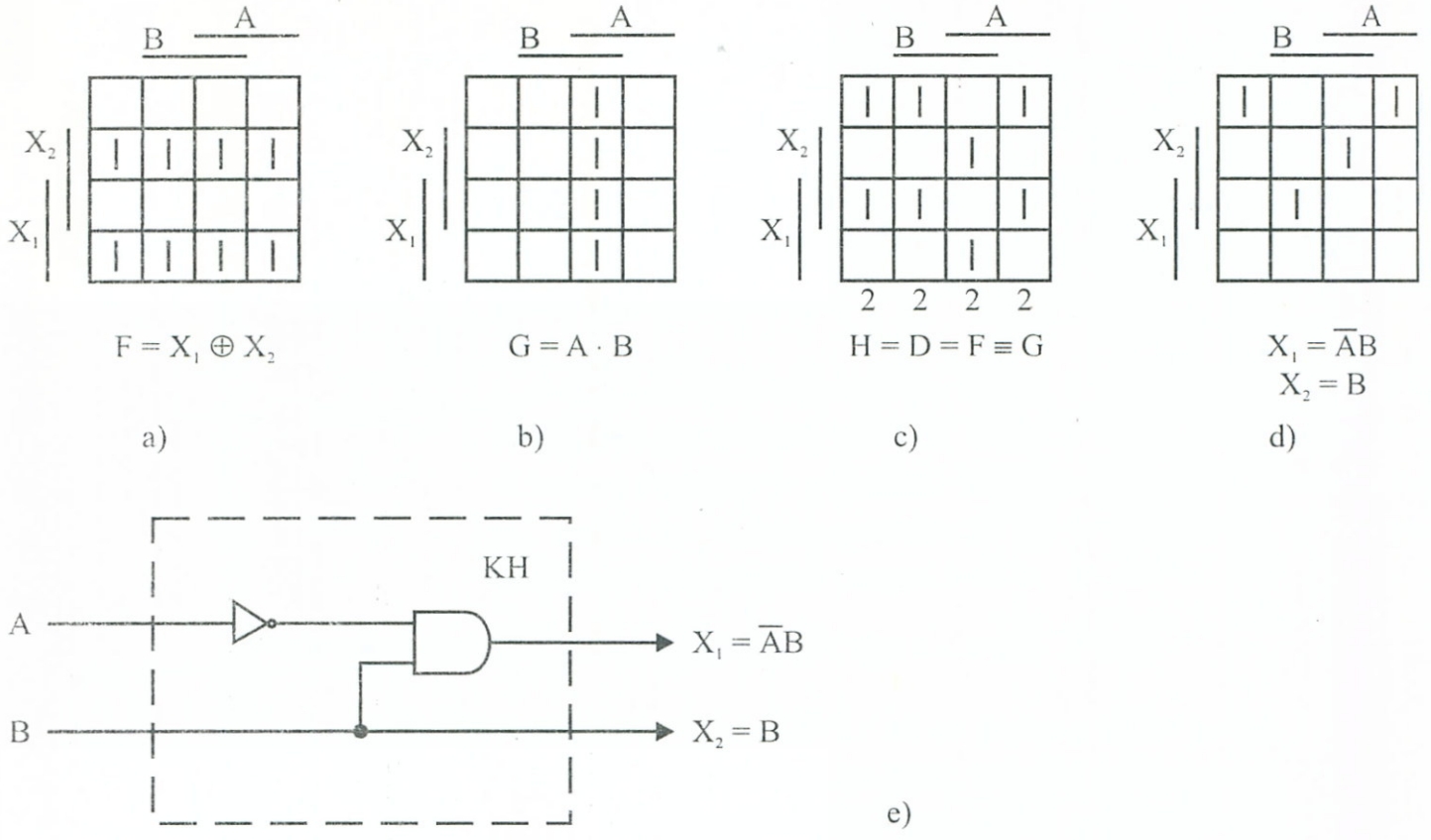
Természetesen egy összetettebb áramkör esetén ezek a „feleslegek" – esetleg más részeknél – még felhasználhatók lehetnek, viszont a távolabbi összeköttetések kiépítése a hordozó nyomtatott áramkört bonyolítja.

A 12–91b. ábrabeli eset csak 3 db IC-t igényel és ezek mindegyike 100%-os kihasználtságú.

F.7.9. Az előtét KH tervezése – mint a 7. fejezet 7.5. pontjában már tapasztaltuk – egyenletmegoldásra vezethető vissza. Felírható a 12–83. ábra alapján:

$$\underbrace{X_1 \oplus X_2}_{F} = \underbrace{A \cdot B}_{G} = Y$$

Grafikus megoldást választva és követve a 7.12. példa tapasztalatait, a 10–92. ábrán foglalhatók össze a lépések.



12-92. ábra *Előtét-hálózat tervezése egyenletmegoldással*

Az a) és b) ábrákon az egyenlet bal- (F), ill. jobb oldalát (G) ábrázoltuk. A (7.10) összefüggés alapján képzett diszkrimináns a c) ábrán látható. Innen megállapítható, hogy összesen $2\times2\times2\times2 = 16$-féle megoldása van az egyenletnek. Ezek közül „találomra" kiválasztva a d) tábla szerinti esetet, a megoldások leolvashatók. Az előtét-hálózat egyik lehetséges realizációját az e) ábra mutatja.

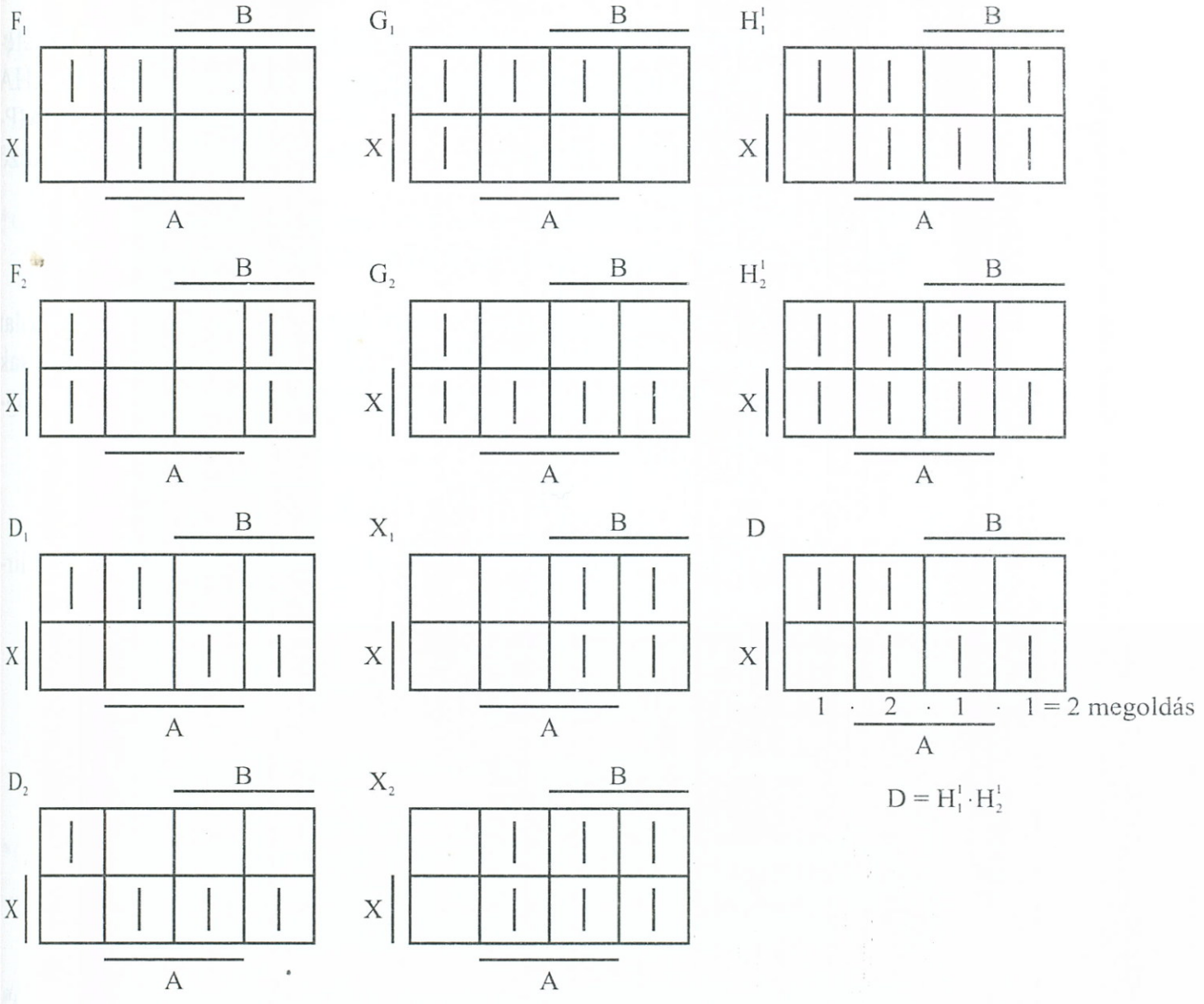
F.7.10. Az egyenlőtlenség megoldását a (7.11) összefüggések alapján mindig egyenletmegoldásra vezetjük vissza.

Első lépésben a 7.5.1. pont összefüggései alapján írhatjuk:

$$F_1 = \overline{B}(\overline{X} + \overline{A}) \qquad F_2 = \overline{A}$$
$$G_1 = \overline{X}A + \overline{B}\,\overline{A} \qquad G_2 = \overline{B}\,\overline{A} + X$$
$$H_1^1 = F_1 + \overline{G_1} \qquad H_2^1 = \overline{F_2} + G_2$$

A grafikus megoldás lépései a 12–93. ábrán követhetők, az előzőkben összefoglalt algoritmus-lépések alapján.

A diszkrimináns: $D = H_1^1 \cdot H_2^1$, mint látható, két megoldást is kaptunk. Ezek:



12-93. ábra *Egyenlőtlenség-rendszer grafikus megoldása*

$$X_1 = B \qquad X_2 = B + A$$

F.7.11. A példa hasonlít a 7.5.5.3. pont 7.14. példájához. A G.7.11. kiindulás ítéleteinek és következtetéseinek alapján kiszűrve a „változókat", célszerű ezekre a következő jelöléseket bevezetni:

1) F – férfi a tettes
2) K – kistermetű a tettes
3) A – az ablakon mászott be
4) R – férfi ruhát hordott
5) H – hiteles a szemtanú vallomása

Ezután a kiinduló ítéletekről és következtetésekről feltételezzük, hogy külön-külön és együtt is igazak, majd a „HA U, AKKOR V" kifejezéseket: $U \rightarrow V$-vel jelölve (lásd IMPLIKÁCIÓ-s függvény: 2–9. ábra) felírható a következő egyenlet:

$$(F \rightarrow K) \cdot (K \rightarrow A) \cdot (F + R) \cdot [R \rightarrow (H \rightarrow A)] \cdot \overline{A} = \text{igaz}$$

A kifejezés csak akkor lehet igaz (1), ha az ÉS-kapcsolat minden összetevője is igaz (1). Az implikációra ugyancsak a 2–9. ábra táblázata alapján felírható az alábbi összefüggés:

$$U \rightarrow V = \overline{U} + V$$

Végül az előbbi hosszú ÉS-es kifejezés összetevőire felírhatók a következők:

1) $F \rightarrow K = \overline{F} + K = 1$

2) $K \rightarrow A = \overline{K} + A = 1$

3) $F\ V\ R = F + R = 1$

4) $R \rightarrow (H \rightarrow A) = \overline{R} + (\overline{H} + A) = 1$

5) $\overline{A} = 1$

Fentiek alapján „kiszámíthatók" a következők:

5)-ből: $A = 0$

2)-ből: $\overline{K} + 0 = 1 \Rightarrow K = 0$

1)-ből: $\overline{F} + 0 = 1 \Rightarrow F = 0$

3)-ból: $0 + R = 1 \Rightarrow R = 1$

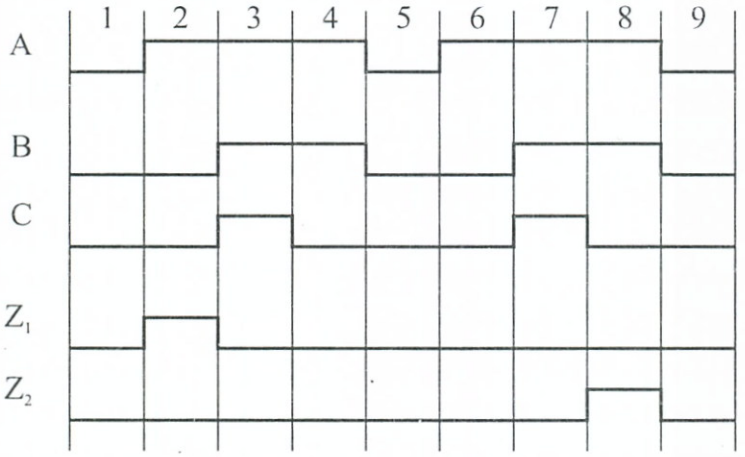
4)-ből: $\overline{1} + (\overline{H} + 0) = 1 \Rightarrow 0 + \overline{H} = 1 \Rightarrow H = 0$

A számszerű eredményeket „emberi nyelvre" visszafordítva a következő megállapításokat tehetjük:

$F = 0$ nem férfi a tettes
$K = 0$ nem kistermetű
$R = 1$ férfi ruhát hordott
$H = 0$ nem hiteles a szemtanú vallomása

12.8.1. Gyakorló feladatok a 8. „Sorrendi hálózatok kialakítási kérdései" c. fejezethez

*G.8.1. Tervezzük meg legkevesebb tárolóelemet igénylő megoldással a 12–94. ábrán látható ütemdiagram szerint működő hálózatot.



12-94. ábra *Kiinduló ütemdiagram a G.8.1. példához*

*G.8.2. Kövessük végig az előző példa S–R tárolós megoldási változatának ütemdiagram szerinti működését az állapotgráfon, megjelölve az ütemszámokat az átmeneti éleken.

*G.8.3. Tervezzük meg S–R tárolós realizálással a következő táblázattal megadott szekvenciális hálózatot.

A	*B*	*C*	*T* időszak	*Z*
			idő ↓	
0	0	0	1	0
0	0	1	2	0
0	0	0	3	1
0	1	0	4	0
0	1	1	5	0
0	1	0	6	1
1	1	0	7	0
1	0	0	8	0
1	0	1	9	0
1	0	0	10	1
0	0	0	11	0

*G.8.4. Ellenőrizzük le „Lépcsős táblázatos" módszerrel, hogy a 8.6. Példában „csoportokra bontással" egyszerűsített állapottábla valóban a legegyszerűbb-e?

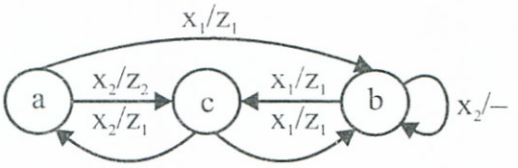
*G.8.5. Adva a 12–95. ábra előzőleg egyszerűsített állapottáblája. Keressünk hozzá állapotkódolást „szomszédossági elv"-ből kiindulva.

q_i \ x	q_{i+1} x=0	q_{i+1} x=1	z x=0	z x=1
a	b	b	0	0
b	d	e	0	0
d	h	i	0	0
e	i	i	0	0
h	a	a	0	0
i	a	a	0	1

12-95. ábra *Kiindulás a G.8.5. példához*

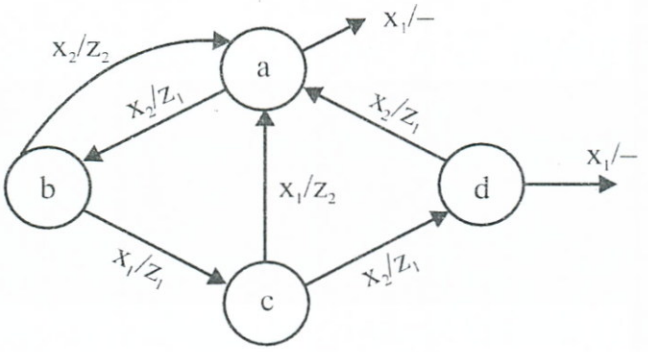
*G.8.6. Alakítsunk ki *bináris sorozatdetektort* olyan esetre, ahol $Z = 1$, ha a beérkező utolsó 4 bit: 1001.

*G.8.7. Tegyünk kísérletet a 12–96. ábrán látható automatagráf egyszerűsítésére



12-96. ábra *Kiindulás a G.8.7. példához*

*G.8.8. Vizsgáljuk meg a 8.20a. ábrán megadott állapottáblát olyan szempontból, hogy a 8.11. példa keretében felderített: P_1, P_2, P_3 helyettesítési tulajdonságú partíciók mellett találunk-e még további esetet.

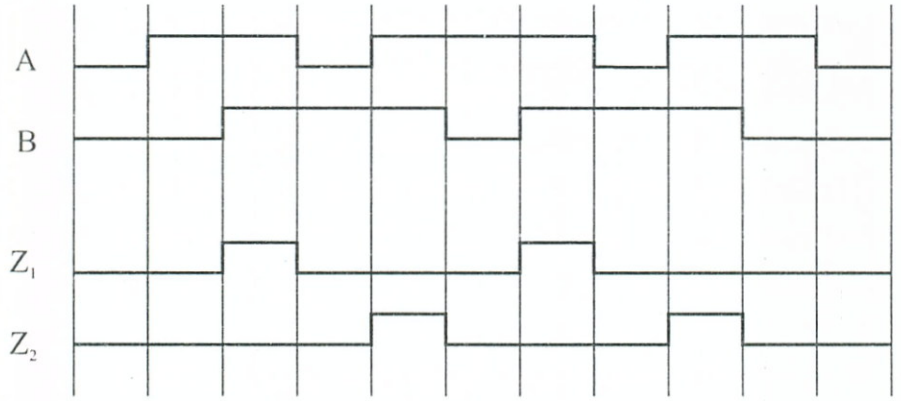


12-97. ábra *Kiindulás a G.8.9. példához*

*G.8.9. Vizsgáljuk meg a 12–97. ábra gráfjával megadott automatát helyettesítési tulajdonságú partíciók szempontjából, és értékeljük, miként kaphatjuk a legegyszerűbb hálózatot.

*G.8.10. Tervezzük meg és ROM-mal realizáljuk a 8–14f. ábra állapottáblával kiindulásként megadott sorrendi hálózatot.

*G.8.11. Tervezzük meg a 12–98. ábra idődiagramjával jellemzett aszinkron sorrendi hálózatot S–R tárolótípussal.

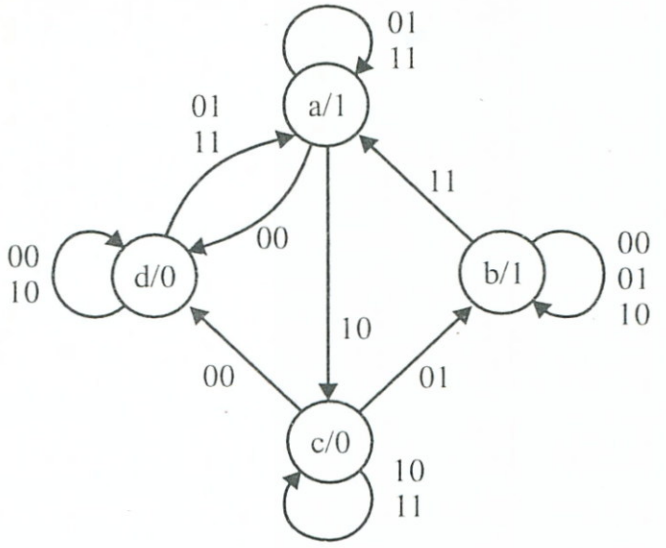


12-98. ábra *Kiinduló idődiagram a G.8.11. feladathoz*

*G.8.12. Folytassuk a 12–99. ábrán megadott egyszerűsített állapottáblából kiindulva a sorrendi hálózat tervezését.

q \ AB	00	01	11	10
I	(a)	b	c	(a)
II	c	(b)	(b)	(b)
III	(c)	(c)	(c)	a

12-99. ábra *Kiinduló állapottábla a G.8.12. feladathoz*



12-100. ábra *Kiinduló állapotgráf a G.8.13. feladathoz*

*G.8.13. Tervezzük tovább a 12–100. ábrán látható gráffal jellemzett aszinkron sorrendi hálózatot.

*G.8.14. Állítsunk elő kritikus versenyhelyzetmentes állapotkódolást a 12–101. ábra állapottáblájával jellemzett aszinkron sorrendi hálózatnál.

q \ AB	00	01	11	10
a	b /0	–/–	c/–	ⓐ/0
b	ⓑ/0	ⓑ/0	ⓑ/0	a/0
c	–/–	b/–	ⓒ/1	d/1
d	b/–	–/–	b/–	ⓓ/1

12-101. ábra *Kiinduló állapottábla a G.8.14. feladathoz*

*G.8.15. Tervezzük meg a 10–62. ábrán bemutatott szorzóberendezés *V* vezérlőegységét *fázisregiszteres* felépítéssel.

*G.8.16. Tervezzük meg a 10–76. ábrán bemutatott osztóberendezés *V* vezérlőegységét *mikroprogramozott* elvű felépítéssel.

G.8.17. Állítsuk elő a G.8.15. feladatot realizáló hálózat logikai vázlatát.

G.8.18. Állítsuk elő a G.8.16. feladatot realizáló hálózatot.

12.8.2. Feladatmegoldások a 8. fejezethez

F.8.1. A megoldást az intuitív módszerrel végezzük el. Az algoritmus lépései:

I. lépés: Ütemdiagram adva van.

II. lépés: A megkülönböztetendő ütemek (12–102a. ábra).

↓ – ② – ⑥

~~⓪~~ – ④ – ⑧

III. lépés: Elvi felépítésként válasszunk tárolós változatot.

IV. lépés: A tárolóelem típusát előre nem tudjuk, sőt éppen olyan típust keresünk, mely egyszerűbb megoldást ad. Próbálkozzunk *először S–R-tárolóval*. Az első, amit fel kell ismernünk az, hogy ennél a típusnál az idővonal indítása után nem kell annak fenntartásáról gondoskodni, mivel a tároló bebillent állapotában stabilan megmarad mindaddig, míg vissza nem billentjük. Ez akkor szükséges éppen, amikor az idővonalat meg kívánjuk szakítani.

V. lépés: Az idővonal berajzolása során, többszöri kísérletezés után megállapíthatjuk, hogy a kimenetek szempontjából szükséges megkülönböztetést elvégzi ugyan egyetlen szekunder-változóhoz rendelt idővonal (például Q_1), de ennek indításához, ill. megszakításához további megkülönböztetésekre van szükség, ami miatt egy második (például Q_2) szekunder-változót is fel kell vennünk (12–102a. ábra). Erre az esetre szemléltetésül felrajzoltuk az *állapotábrát* is (12–102b. ábra).

VI. lépés: A tárolók működtető függvényeit közvetlenül a diagramból írjuk fel egyszerűsítetten:

1. tároló: $S_1 = C \cdot \overline{Q_2}$

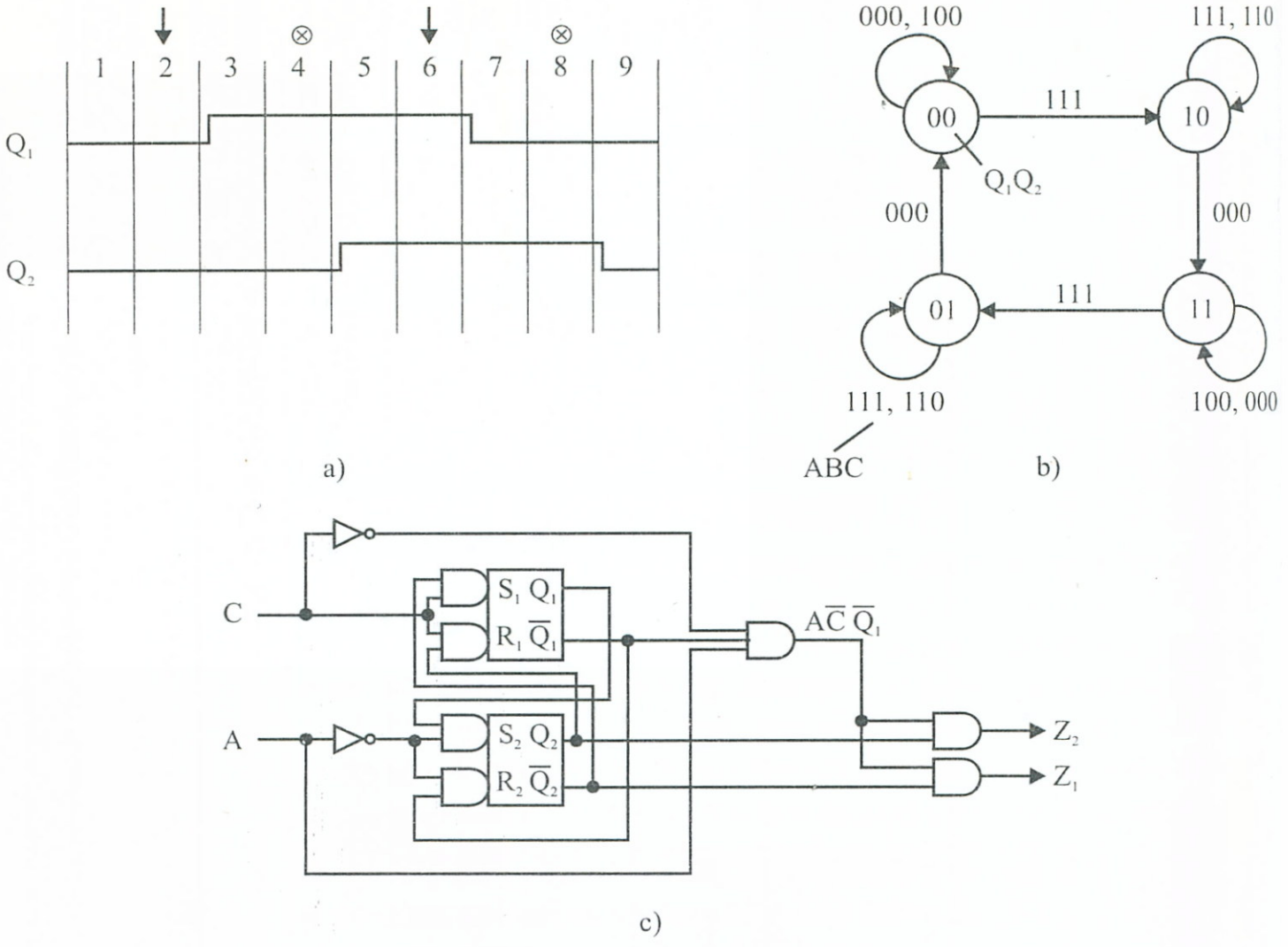
$$R_1 = C \cdot Q_2$$

2. tároló: $S_2 = \overline{A} \cdot Q_1$

$$R_2 = \overline{A} \cdot \overline{Q_1}$$

Az S–R tárolókra teljesülni kell az alábbi feltételnek is:

$$S \cdot R = 0$$



12-102. ábra *Megoldás S-R tárolóval*

Ezt mindig ellenőriznünk kell a példamegoldások során. Jelenleg:

$$S_1 \cdot R_1 = (C \cdot \overline{Q}_2) \cdot (C \cdot Q_2) = 0$$

$$S_2 \cdot R_2 = (\overline{A} \cdot Q_1) \cdot (\overline{A} \cdot \overline{Q}_1) = 0$$

mindkét esetben teljesül. Megjegyezzük, előfordulhat, hogy az eredmény nem „0", hanem egy *határozatlan kombinációt* eredményez. A feltételt ilyenkor is teljesültnek tekinthetjük, mivel a határozatlan kombinációknak tetszés szerint tulajdoníthatunk „0", vagy „1" értéket.

VII. lépés: A kimeneti függvények egyszerűsített alakban:

$$Z_1 = A \cdot \overline{C} \cdot \overline{Q}_1 \cdot \overline{Q}_2$$

$$Z_2 = A \cdot \overline{C} \cdot \overline{Q}_1 \cdot Q_2$$

Más alakot is felírhattunk volna, de ezek nem tartalmaznak „B”-t ugyanúgy, mint a *VI. lépés* „S” és „R” függvényei, ily módon a „B” független-változót eliminálhattuk.

VIII. lépés: A logikai vázlatot a 12–102c. ábrán rajzoltuk fel.

IV/A. lépés: Próbálkozzunk ezután „T” tárolós megoldással. Ez azt jelenti, hogy az eljárást a *IV. lépéstől* kezdve újra le kell folytatnunk.

V/A. lépés: Elegendő egy idővonal felvétele, mivel „C”-vel felváltva be-, ill. kibillenthetjük a tárolót (12–103a. ábra).

VI/A. lépés: A működtető függvény rendkívül egyszerű:

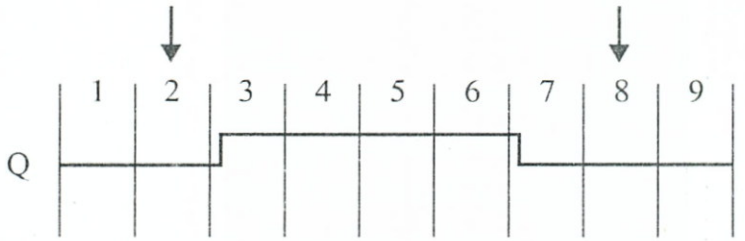
$$T = C$$

VII/A. lépés: A kimeneti függvények a következők lesznek:

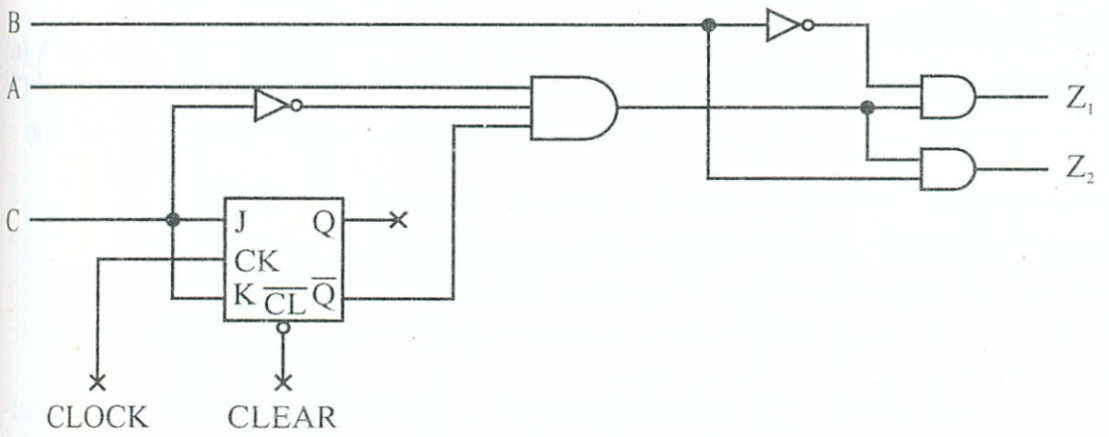
$$Z_1 = A \cdot \overline{B} \cdot \overline{C} \cdot \overline{Q}$$

$$Z_2 = A \cdot B \cdot \overline{C} \cdot \overline{Q}$$

Megjegyezzük, hogy például Z_1-ből B-et elhagyva, a 2-es ütem elején „Q” megjelenésének kezdetéig, Δ hosszúságú



a)



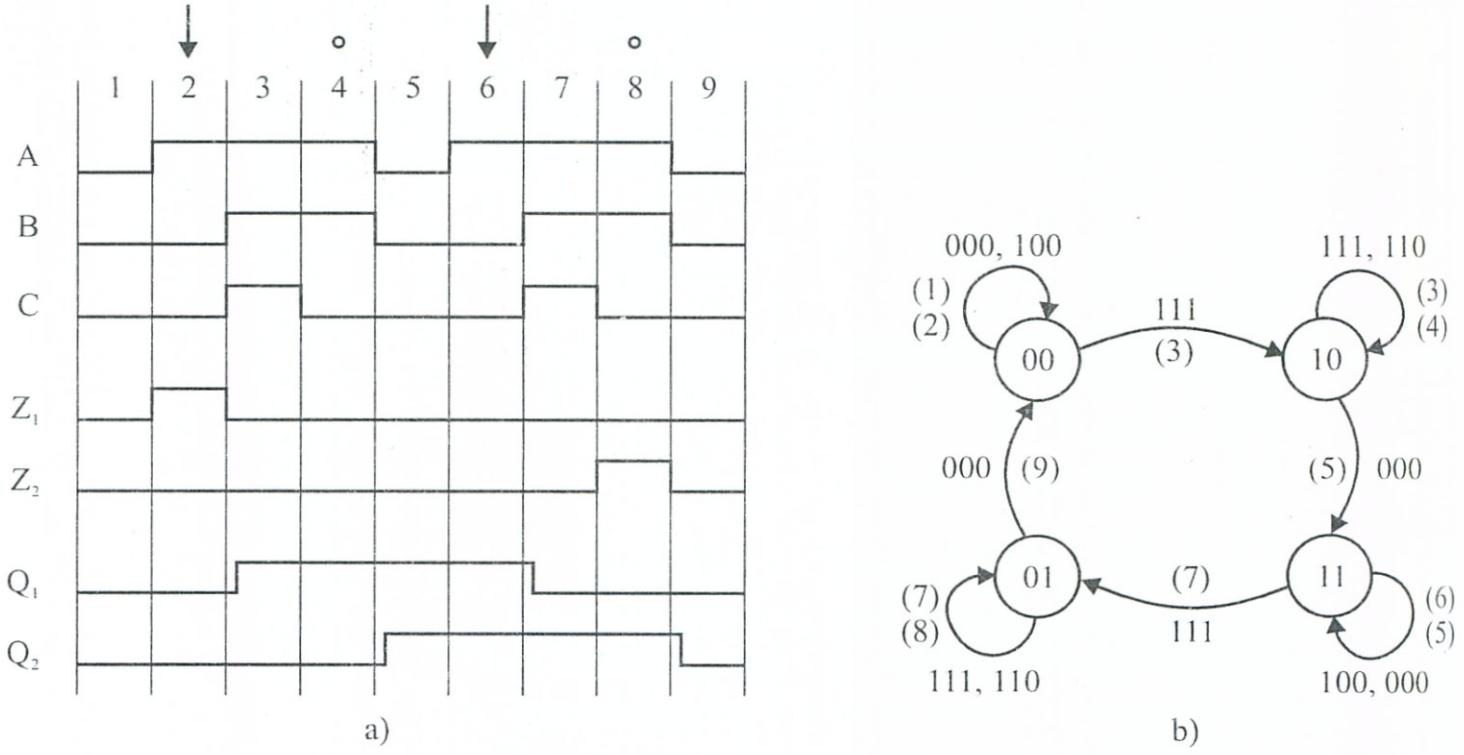
b)

12-103. ábra *Megoldás T tárolóval*

tranziens keletkezik. Néha ez megengedhető, ekkor az egyszerűsítés ily módon is tovább vihető.

VIII/A. lépés: A hálózat a 12–103b. ábrán látható, a T-tárolót, az integrált áramköröknél gyakrabban előforduló J–K tároló két bemenetének összekötésével alakítottuk ki. A T-tárolót a működési ciklus előtt törölni kell a $\overline{CL}$–CLEAR bemenetnél. A berendezés, elvileg mind szinkron, mind aszinkron üzemmódban működőképes, ha a tárolóelemet a megfelelő változatban építjük be.

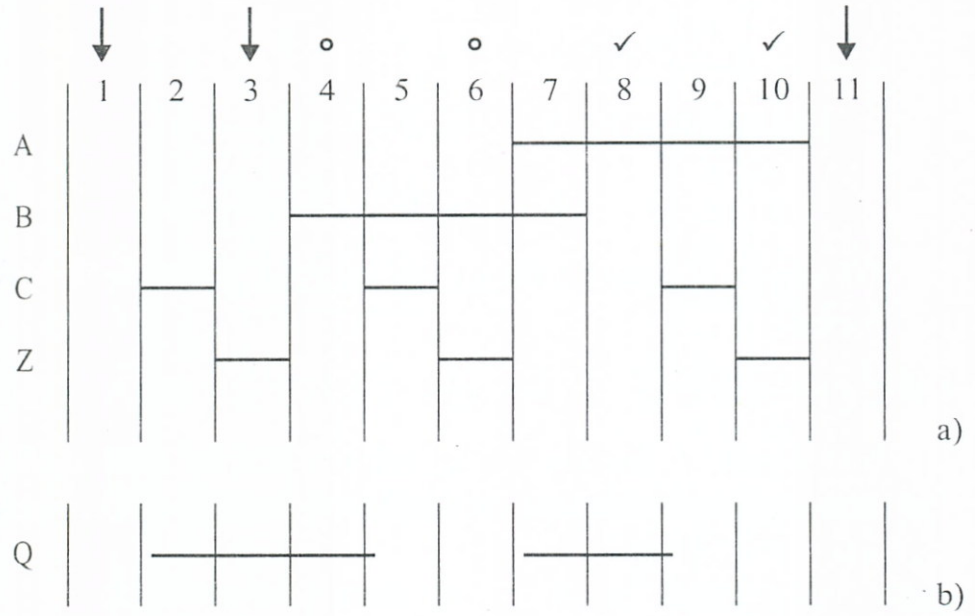
F.8.2. A 12–104a. ábrán az S–R tárolós megoldáshoz tartozó két szekunder-változós teljes ütemdiagramot rajzoltuk fel.
A 12–104b. ábra állapotgráfján zárójelezett számokkal tüntettük fel a megfelelő ütemek sorszámait. A Z_1Z_2 kimeneteket a jobb áttekinthetőség miatt nem jelöltük. Az állapotgráfról a két részre esett időütemekhez tartozó instabil állapotok is leolvashatók.



12-104. ábra *Az ütemdiagram és az állapotgráf kapcsolata*

F.8.3. A megoldást a 8.1.1. pont algoritmusa szerint végezzük:

I. lépés: Az ütemdiagramot a 12–105a. ábra szerint rajzolhatjuk fel a kiinduló táblázatból.



12-105. ábra *Ütemdiagram a G.8.3. példához*

II. lépés: A megkülönböztetendő ütemek:

↓ ① , ⑪ — ③

○ ④ — ⑥

✓ ⑧ — ⑩

III. lépés: Az elvi felépítést a feladatmegadás meghatározza.

IV–V. lépés: A szekunder-változó kijelölési művelet nagy figyelmet igényel. Miután több kombinációpárt kell megkülönböztetni, első látásra úgy érezzük, hogy több szekunder-változó bevezetésére is szükség lesz. Figyelmes vizsgálat után azonban belátható, hogy eredményre jutunk egyetlen szekunder-változó többszöri felhasználása révén is, ha az idővonalat a 12–105b. ábra szerint vesszük fel, és az ábrán (a gyakran alkalmazott ábrázolásmód szerint) csak a *magas szintű* változó állapotokat tüntettük fel.

VI. lépés: A kétszeres indítás a tároló bemeneti függvényeinek felírásánál két összetevőt eredményez, ezeket szemlélet alapján a diagramból mindjárt egyszerűsítjük is, ha lehetséges:

Indítás: 2. ütem: $S' = \overline{A} \cdot \overline{B} \cdot C$

7. ütem: $S'' = AB\overline{C} \rightarrow AB$

$$S = S' + S'' = \overline{A} \cdot \overline{B} \cdot C + A \cdot B$$

Visszabillentés: 5. ütem: $R' = \overline{A}BC \rightarrow BC$

9. ütem: $R'' = A\overline{B}C \rightarrow AC$

VII. lépés: Kimeneti függvény felírása közvetlenül a diagramból:

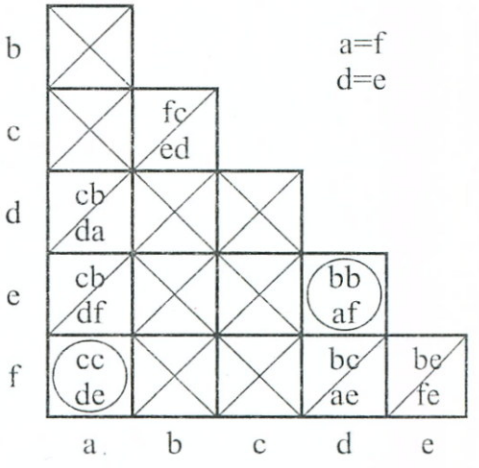
$$Z = \overline{A}\,\overline{B}\,\overline{C}\,Q + \overline{A}\,B\,\overline{C}\,\overline{Q} + A\,\overline{B}\,\overline{C}\,\overline{Q} = \overline{C}\,[\overline{A}\,(\overline{B}\,Q + B\,\overline{Q})] + A\,\overline{B}\,\overline{Q}$$

VIII. lépés: A logikai vázlat felrajzolásától ezúttal eltekintünk, mivel az előző példa alapján ez már nem okoz problémát.

F.8.4. A 12–106. ábrán egymás mellett felrajzoltuk a kiinduló 8–15a. ábrát és a lépcsős táblázatot, melyen elvégeztük az algoritmus szerinti egyszerűsítést. Mint látható, itt a korábbi ekvivalencia adódott.

q_{t+1} / q_t	x=0	x=1	z x=0	z x=1
a	c	d	1	1
b	f	e	0	0
c	c	d	0	0
d	b	a	1	1
e	b	f	1	1
f	c	e	1	1

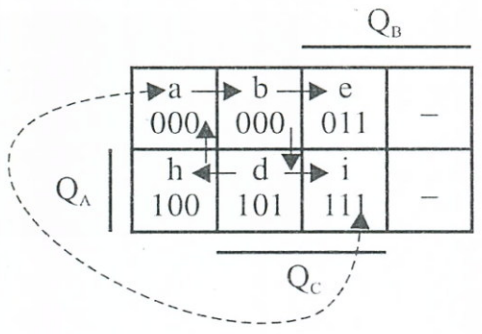
a)



b)

12-106. ábra *Állapotösszevonás lépcsős táblázattal*

F.8.5. A feladatmegoldásnál a 8–10. példában tárgyalt elveket hasznosítjuk. Miután a kiinduló, kódolandó tábla 6 belső állapotot tartalmaz, ezek kódolásához 3 db szekunder-változó (Q_A, Q_B, Q_C) elegendő. A szomszédos (H_e = 1 Hamming-távolságú) Q_A, Q_B, Q_C kombinációkat a 12–107a. ábra V–K táblájáról lehet „összeszedni." például az ott bejelölt *a, b, c, ... f* állapot-elhelyezésekkel. Természetesen itt is adódnak nem át-



a)

q_i	Q_A	Q_B	Q_C
a	0	0	0
b	0	0	1
d	1	0	1
e	0	1	1
h	1	0	0
i	1	1	1
–	0	1	0
–	1	1	0

b)

12-107. ábra *Állapotkódolás szomszédossági kódokkal*

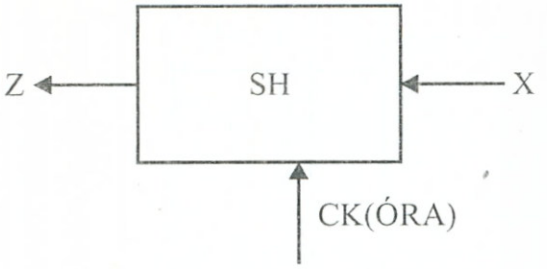
hidalható helyzetek (pl. „*i*" és „*a*" között), de a többiek jól teljesítik a szomszédossági feltételt.
Az állapotkódolási táblázat a 107b. ábrán látható.

F.8.6. A kérdéses sorozatdetektor tervezéséhez támpontul szolgál a korábbi 8.13. Példa. Jelen példánknál a kiindulási feltétel alapján felrajzolható a 12–108. ábrán látható *állapotgráf* és *állapottábla,* melyek egyszerűsítését *csoportokra bontással* elvégezve, a 12-109a. ábra szerinti egyszerűsített állapottáblát rajzolhatjuk fel. Itt az ekvivalens csoportok:

$$\begin{array}{cccc} a & b=d=f=h & c=g & e \\ \alpha & \beta & \gamma & \delta \end{array}$$

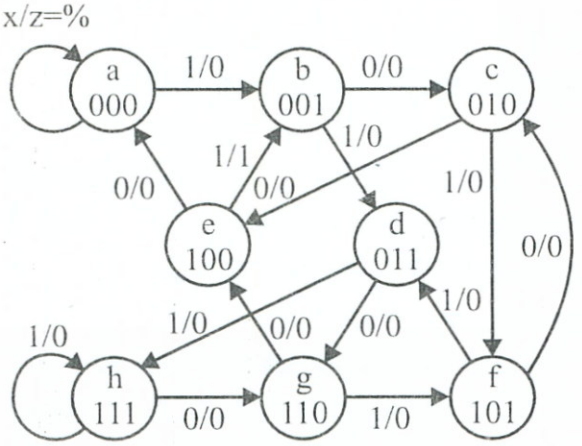
Mint látható, az egyszerűsítés nagyon hatékony volt, így a realizáláshoz 2 tárolóelem elegendő lesz (Q_1Q_2).
Az itt is választott szomszédossági kódolás folyamata a 12–109b. ábrán követhető. A kódolt állapottábla a 12–109c. és e. ábrákon, a minimalizált függvények az f. ábrán, a realizációs hálózat a g) ábrán látható. A realizációnál szinkron S–R-t használtunk, melynek vezérlési tábláját ugyancsak felrajzoltuk a 12–109d. ábrán a számítások megkönnyítése céljából.

F.8.7. A részben határozott automatát leíró gráf alapján felrajzolható állapottábla a 12–110a., a kompatibilitási osztályok zárt halmazának meghatározása céljából felrajzolt lépcsős tábla a 12–110b., az egyszerűsített állapottábla a 12–110c. ábrán, míg a kérdéses gráf a 12–110d. ábrán látható.



Z=1
ha S_x^4=1001

a)



b) állapot gráf

q_t	q_{t+1}/z_t x=0	x=1
a	a/0	b/0
b	c/0	d/0
c	e/0	f/0
d	g/0	h/0
e	a/0	b/1
f	c/0	d/0
g	e/0	f/0
h	g/0	h/0

c) állapottábla

Egyszerűsítés *csoportokra bontással*:
A kimenet alapján indulva:

I.

1	2						
e	a	b	c	d	f	g	h
22	22	22	12	22	22	12	22

II.

1	2		3				
e	c	g	a	b	d	f	h
33	13	13	33	23	23	23	23

III.

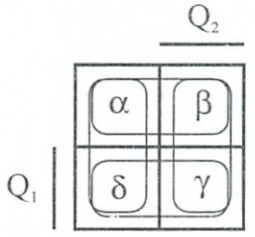
1	2		3	4			
e	c	g	a	b	d	f	h
34	14	14	34	24	24	24	24

12-108. ábra *Sorozatdetektor állapot-egyszerűsítése*

F.8.8. Megvizsgálva újra a 8–20b. lépcsős táblát, próbáljuk meg a vizsgálatot az (af)-ből indítva lefolytatni. A lépéseket a 8.11. Példa szerinti módon elvégezve, karikázással rájelöltük a lépéssorozatot a lépcsős táblára a 12–111a. ábra szerint, majd ezt követően elvégezzük a helyettesítési tulajdonságú partíció-keresést a 12–111b. ábrán (hasonlóan a 8–21. ábra lépéseihez). Mint látható, végül is a

q_t	q_{t+1} x=0	x=1
α	α/0	β/0
β	γ/0	β/0
γ	δ/0	β/0
δ	α/0	β/1

a)



Q_1Q_2
$\alpha = 00$
$\beta = 01$
$\gamma = 11$
$\delta = 10$

b) szomszédos állapotkódolás

	t Q_1Q_2	t+t Q_1Q_2 x=0	x=1	z x=0	x=1
α	00	00	01	0	0
β	01	11	01	0	0
γ	11	10	01	0	0
δ	10	00	01	0	1

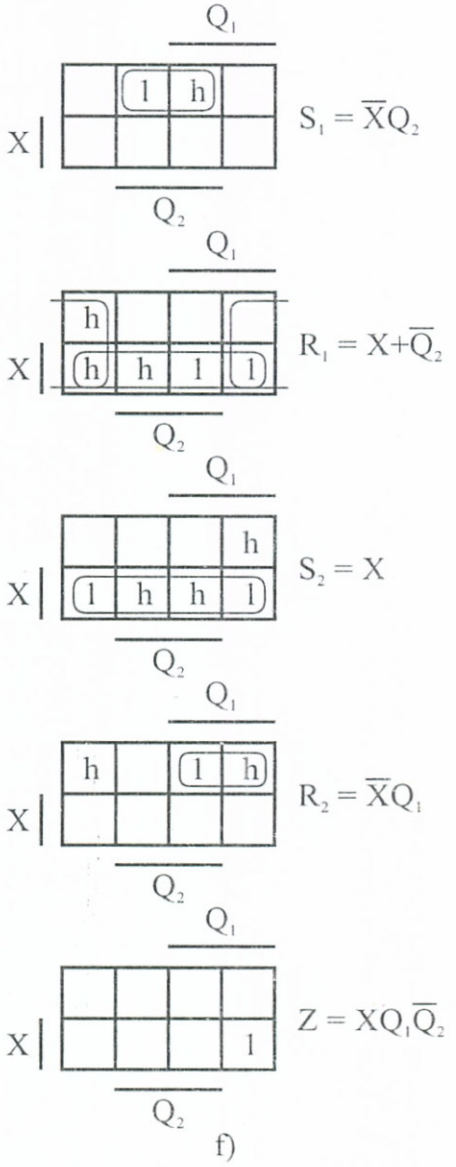
c)

$Q_t \rightarrow Q_{t+1}$		SR
0	0	0h
0	1	10
1	0	01
1	1	h0

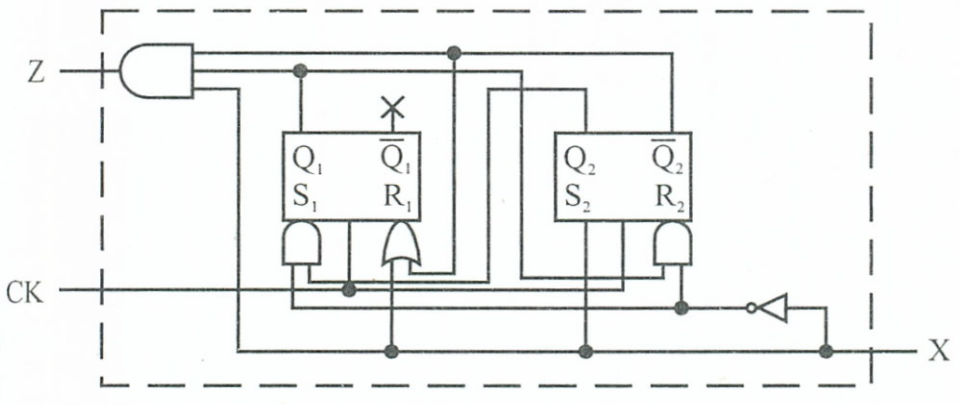
d)

t X	Q_1	Q_2	t+1 Q_1	Q_2	S_1	R_1	S_2	R_2	Z
0	0	0	0	0	0	h	0	h	0
0	0	1	1	1	1	0	h	0	0
0	1	0	0	0	0	1	0	h	0
0	1	1	1	0	h	0	0	1	0
1	0	0	0	1	0	h	1	0	0
1	0	1	0	1	0	h	h	0	0
1	1	0	0	1	0	1	1	0	1
1	1	1	0	1	0	1	h	0	0

e)

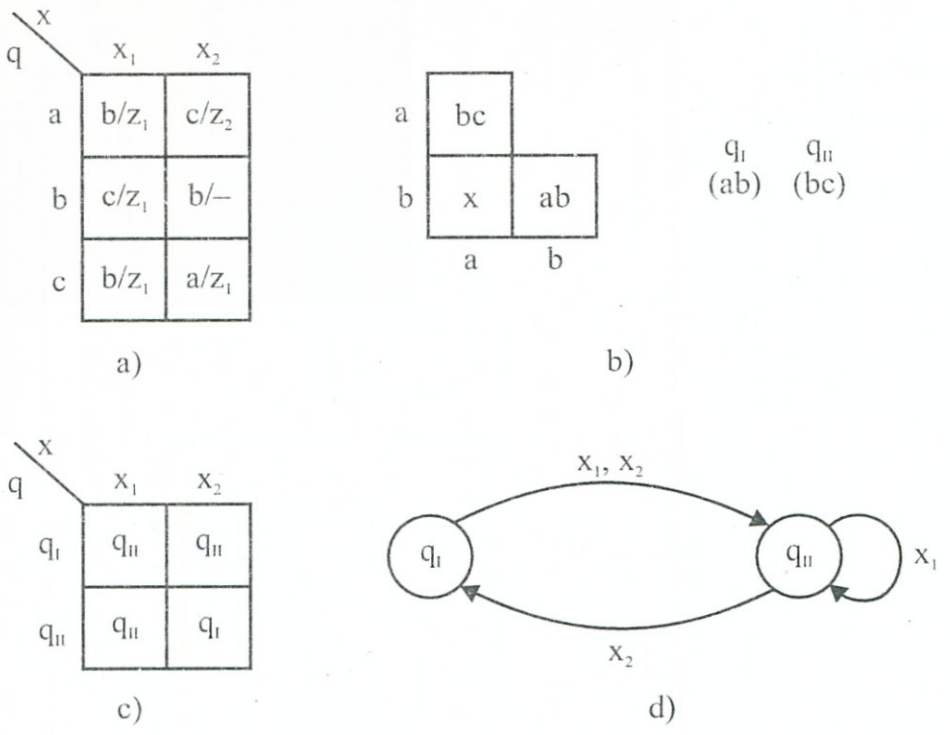


f)

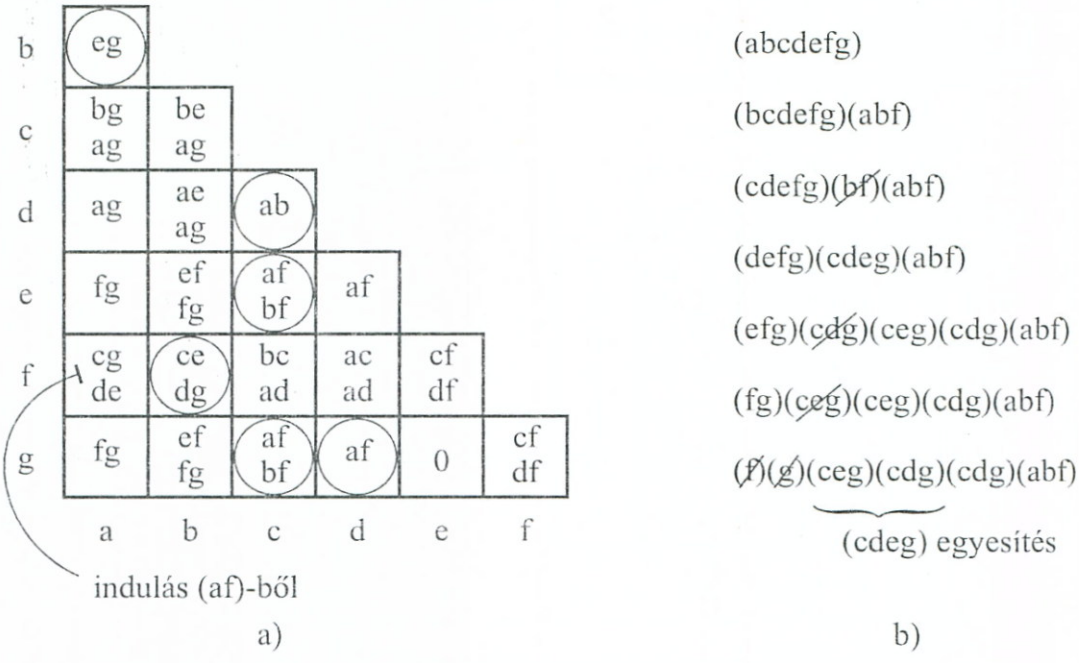


g)

12-109. ábra *Sorozatdetektor $S_x^4 = 1001$-re*



12-110. ábra *Állapotösszevonás az F.8.7. példánál*



12-111. ábra *Helyettesítési tulajdonságú partíció keresése (af)-ből kiindulva*

$$P_4 = \{(abf)\ (cdeg)\}$$

újabb, előnyös partícióhoz jutottunk, melyet a 8–22. ábra szerinti összehasonlításhoz hasonlóan kiértékelve: $m = 3$

($m_1 = 1, m_2 = 2$) a szükséges szekunder-változók száma, ami egyenértékű a korábban legelőnyösebbnek minősített P_3-mal.

F.8.9. A 12–97. ábra állapotgráfja alapján felrajzolhatjuk a 12–112a. ábra szerinti állapottáblát, majd a 12–112b. ábra szerinti lépcsős táblát. Ebből közvetlenül leolvasható a következő helyettesítési tulajdonságú partíció:

$$P_1 = \{(a)\ (c)\ (bd)\}$$

Tovább keresve és (bd)-ből továbblépve a lépcsős táblázat segítségével adódik (ac) partíció is, így egy újabb lehetőség:

$$P_2 = \{(ac)\ (bd)\}$$

A szekunder-változók szükséges számának felmérése után (12–112c. ábra), azt állapíthatjuk meg, hogy P_1-hez: $m = 3$, P_2-höz: $m = 2$ szekunder-változó szükséges.
Folytassuk tovább a feladatmegoldást, és mindkét esetre írjuk fel a vezérlő és kimeneti függvényeket. Mindkettőhöz válasszuk a következő összerendelést:

$$x = \{x_1, x_2\} = \{0, 1\} \qquad z = \{z_1, z_2\} = \{0, 1\}$$

q_T	x_1	x_2
a	–/–	b/z_1
b	c/z_1	a/z_2
c	a/z_2	d/z_1
d	–/–	a/z_1

a)

b	ab		
c	(bd)	ac ad	
d	ab	0	ad
	a	b	c

b)

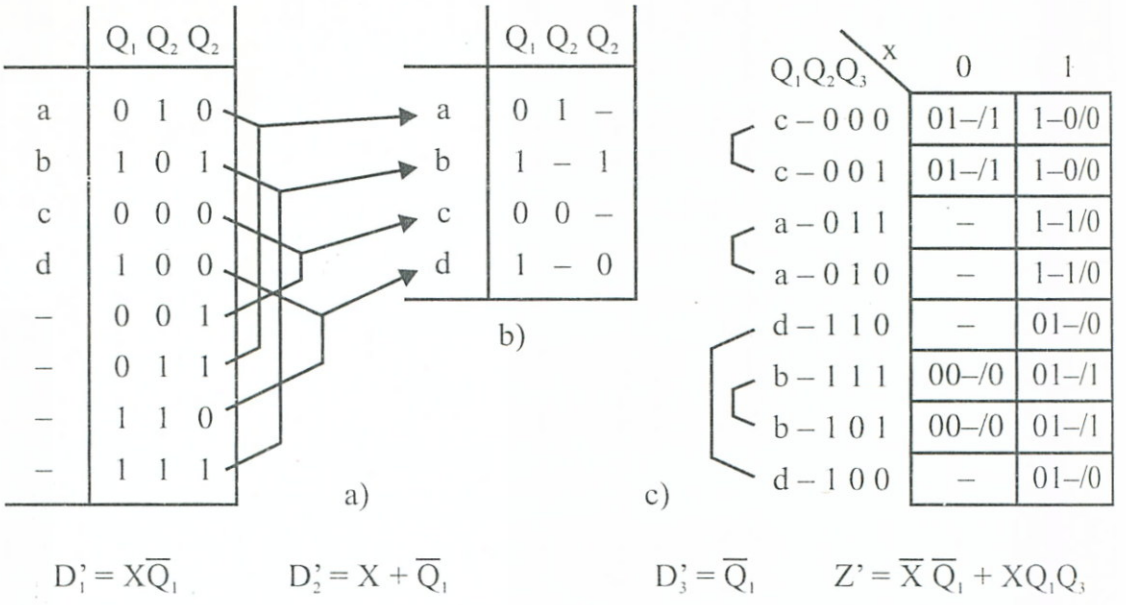
$P_1 = \{(a)(c)(bd)\}$

$P_2 = \{(ac)(bd)\}$

p_i	c_i	p_i	m_{1i}	m_{2i}	m
p_1	3	2	2	1	3
p_2	2	2	1	1	2

c)

12-112. ábra *Helyettesítési tulajdonságú partíciók F.8.9.-nél*

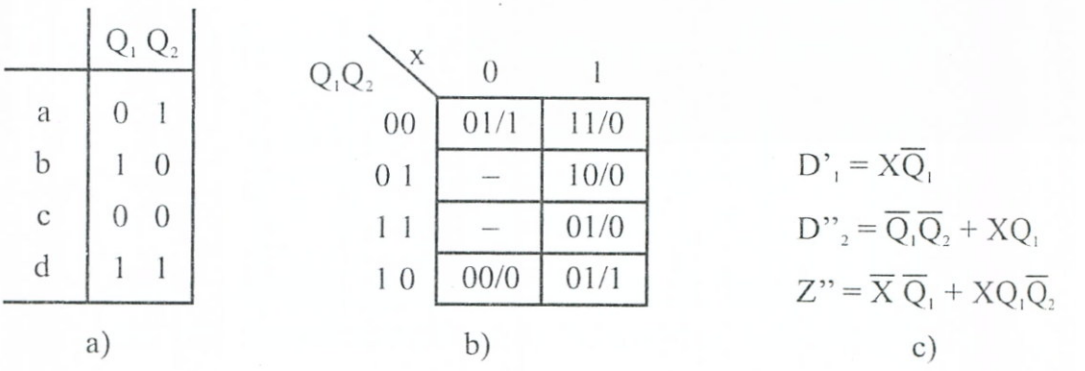


12-113. ábra *Az F.8.9. feladatmegoldás I. változata*

A P_1 esetre a szekunder-változókat kódoljuk a 12–113a. ábra szerint, ahol Q_1Q_2 a csoportok megkülönböztetését, Q_3 a csoportokon belüli állapotok megkülönböztetését végzi. Itt a fel nem használt esetek jóvoltából a 12–113b. ábra szerinti összevonást végezhetjük el, végül a 12–113c. kódolt állapottábla alapján felírhatók a D_1', D_2', D_3', és Z' függvények.

A P2 esetre a szekunder-változó kódolást a 12–114a. ábra szerint elvégezve (Q_1 csoport, Q_2 állapot megkülönböztetés) a 12–114b. kódolt állapottáblából felírhatók a D_1'', D_2'', Z'' függvények.

A P_1 alapján kapott I. változatot a P_2 alapján kapott II-kal összehasonlítva azt találjuk, hogy a kombinációs hálózatrész kapubemeneteinek száma P_1 esetén: 11, míg P_2 esetén: 15, azaz a II. változat bonyolultabb.



12-114. ábra *Az F.8.9. feladatmegoldás II. változata*

Összefoglalva a két változat tapasztalatait adott példára, megállapíthatjuk:

– P_1 megoldásánál a kombinációs hálózatrész egyszerűbb,
– P_2 megoldáshoz kevesebb számú tárolóelem szükséges.

A példa felhívja a figyelmet arra, hogy a különféle egyszerűsítések elvégzése után, olyan egymás ellen dolgozó eredmények is keletkezhetnek, melyek miatt a végleges realizációhoz további mérlegeléseket is kell tennünk. Esetünkben például, ha PLA realizáció mellett döntenénk, akkor a kevesebb szekunder-változót jelentő P_2 megoldás lenne a kedvezőbb, mivel – mint már utaltunk rá – PLA-nál a kombinációs hálózatrész egyszerűsödésének nincs lényeges szerepe.

F.8.10. Kiindulva az újra felrajzolt 12–115a. ábrából, szomszédossági alapon választjuk a 12–115b. szerinti párosítást, majd kitölthetjük a 12–115c. kódolt állapottáblát. J–K tárolóelemet választva, a 12-115c. alapján 12–115d. ábra szerinti vezérlési és kimeneti függvények írhatók fel, végül a realizációs hálózatot a 12–115e. ábra mutatja. A vezérlési és kimeneti függvények a ROM-ba vannak beégetve a 6–41. ábra és a 8–26. ábra elveinek megfelelően.

F.8.11. A feladatmegoldás hasonló lépésekben történik, mint a 8.15. példánál.

I–II. lépés: Induláskor megbetűzzük a kiinduló idődiagram ütemeit, melyeket stabil állapotoknak tekintünk, majd felrajzolhatjuk a 12–116a. ábrán látható primitív állapottáblát.

III/A. lépés: Primitív állapottáblánkat a kompatibilitási feltételek alapján megvizsgálva a 12–116b. ábra következtetéseire juthatunk, melyeket a primitív állapottáblára rávezetve, majd az állapot-összevonásokat elvégezve a 12–116c. ábra egyszerűsödött táblázatát kapjuk.

III/B. lépés: A felrajzolható 12–116d. egyesítési gráf értelmében viszont a sorok egyesítésével még további egyszerűsítési lehetőségek is kínálkoznak, így végül is a kétsoros 12–116e. ábra szerinti állapottábla lesz az eredmény.

q_t	q_{t+1} $x_0=0$	q_{t+1} $x_1=1$	Z $x_0=0$	Z $x_1=1$
a	a	b	0	0
b	c	b	0	0
c	e	b	0	0
e	a	b	0	1

a)

t+1	t
a	a, e
b	a, b, c, e
c	b
d	c

a: 00
e: 01
b: 10
c: 11

b)

q_t	t Q_1Q_2	t+1 0	t+1 1
a	0 0	00/0	10/0
e	0 1	00/0	10/1
c	1 1	01/0	10/0
b	1 0	11/0	10/0

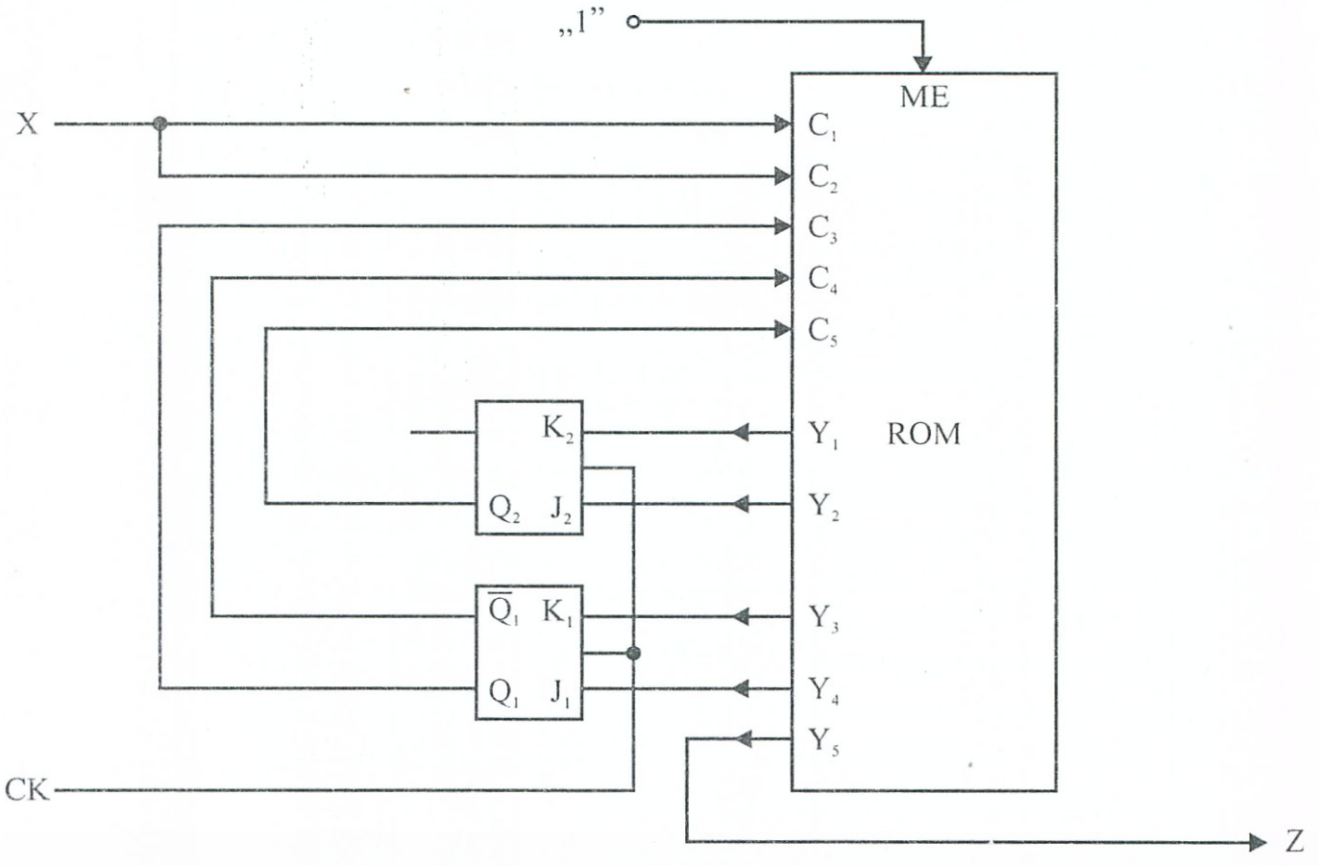
qt + 1/z

c)

$J_1 = X$
$K_1 = \overline{X}Q_2$
$J_2 = \overline{X}Q_1$
$K_2 = X+\overline{Q}_1$

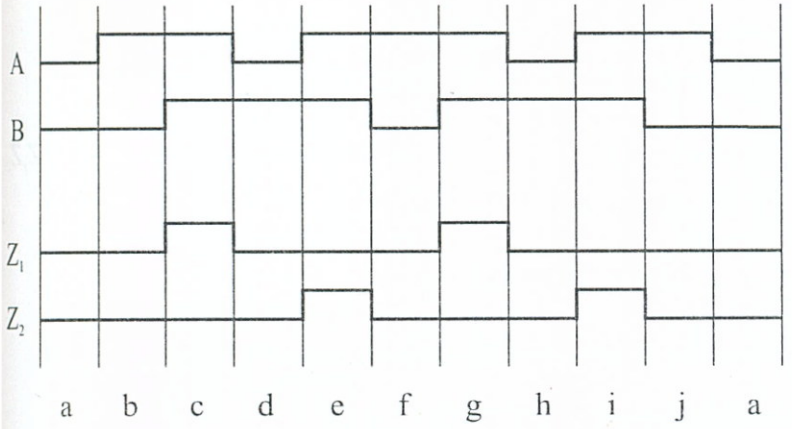
$Z = X\overline{Q}_1Q_2$

d)



e)

12-115. ábra *Az F.8.10. példa megoldása*



a)

q \ AB	00	01	11	10	Z_1	Z_2
I	(a)	–	–	b	0	0
II	–	–	c	(b)	0	0
III	–	d	(c)	–	1	0
IV	–	(d)	e	–	0	0
V	–	–	(e)	f	0	1
VI	–	–	g	(f)	0	0
VII	–	h	(g)	–	1	0
VIII	–	(h)	i	–	0	0
IX	–	–	(i)	j	0	1
X	a	–	–	(j)	0	0

b ~ f ha: c ~ g II.–VI.

c ~ g ha: d ~ h III.–VII.

d ~ h ha: e ~ i IV.–VIII.

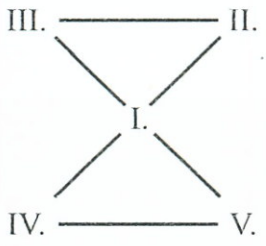
e ~ i ha: f ~ j V.–IX.

f ~ j feltétel nélkül VI.–X.

b)

q \ AB	00	01	11	10	Z_1	Z_2
I	(a)	–	–	b	0	0
II	–	–	c	(b)	0	0
III	–	d	(c)	–	1	0
IV	–	(d)	e	–	0	0
V	–	–	(e)	b	0	1

c)

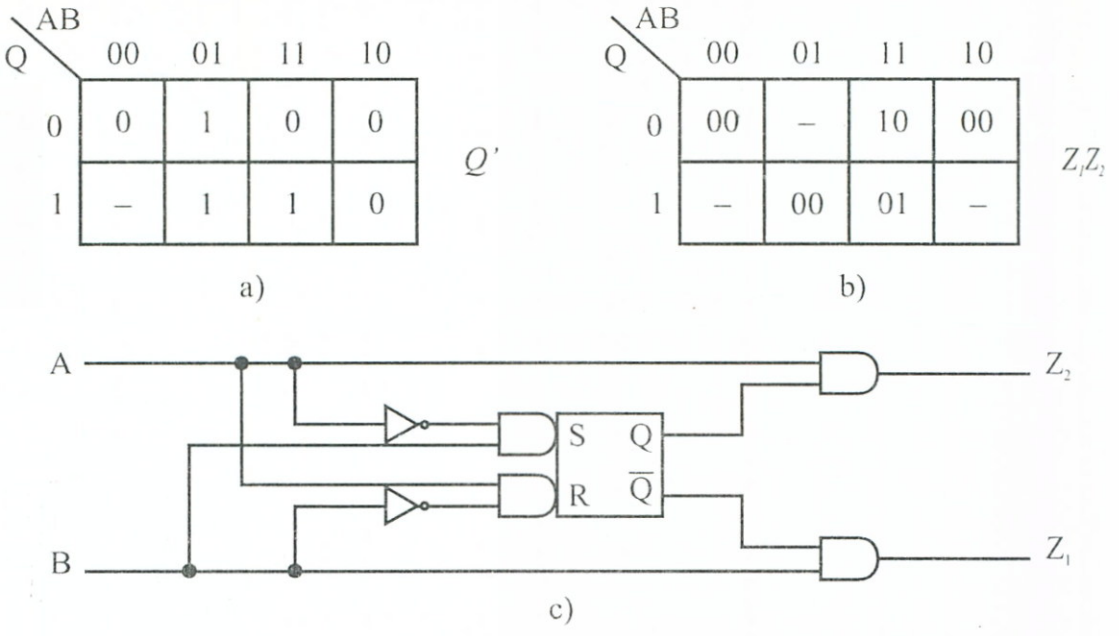


d)

q \ AB	00	01	11	10	
I.–II.–III.	(a)/00	d	(c)/10	(b)/00	Q = 0
IV.–V.	–	(d)/00	(e)/01	b	Q = 1

e)

12-116. ábra *Az F.8.11. feladatmegoldás lépései*



12-117. ábra *Az F. 8.11. feladat realizációja*

IV. lépés: Az állapotkódolást problémamentesen végrehajtva a kódolt belső állapottábla és a kimeneti tábla a 12–117a. és b. ábrák szerint alakul.

V. lépés: A feladat S–R-tárolós megoldást ír elő. Az $S = f_S(A, B, Q)$ és $R = f_R(A, B, Q)$ működtető függvényeket – a korábbiakban már alkalmazott módszerrel – a 6–34. ábra működési táblázatának segítségével írhatjuk fel és végül kapjuk:

$$S = \overline{A}B$$
$$R = A\overline{B}$$

VI. lépés: A kimeneti függvényeket a kimeneti táblából felírva:

$$Z_1 = B \cdot \overline{Q}$$
$$Z_2 = A \cdot Q$$

VII. lépés: Az S–R tárolóelemes realizációs hálózatot végül a 12–117c. ábrán rajzoltuk fel.

F.8.12. A kiinduló 12–59. ábra alapján felrajzolható az átmeneti gráf a 12–118a. ábra szerint. A három belső állapot kódolásához legalább két szekunder-változó szükséges. Az így keletkező négysoros V–K-táblán nem tudjuk úgy elhelyezni a gráfon látható (mindegyik mindegyikkel kapcsolatban álló)

három állapotot, hogy mindenütt közvetlen *szomszédos* átmenet keletkezzen. Pl. a 12–118b. ábrán látható elhelyezésnél a kódok:

AB : 10 → 11 változásnál :

q : ⓐ → c → ⓒ

Q_1Q_2 : ⓪⓪ → 00 → ⑪

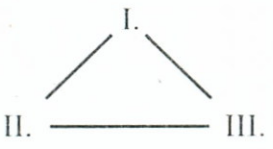
AB : 11 → 10 változásnál :

q : ⓒ → a → ⓐ

Q_1Q_2 : ⑪ → 11 → ⓪⓪

tehát ezekben az esetekben a szomszédosság nem teljesül.

Amennyiben a határozatlan eseteket tartalmazó: Q_1Q_2:01 sorban az AB:11, ill. 10 oszlopnál „c", ill. „a" instabil állapotokat, a „00", ill. „11" sorban pedig értelemszerűen „IV" instabil állapotokat iktatunk be, az előbbi átmenet-sorozat



a)

Q_1Q_2	q \ AB	00	01	11	10
0 0	I.	(a)	b	c	(a)
0 1	IV.	–	–	–	–
1 1	III.	(c)	(c)	(c)	a
1 0	II.	c	(b)	(b)	(b)

b)

Q_1Q_2 \ AB	00	01	11	10
00	00	10	01	00
01	–	–	01	00
11	11	11	11	01
10	11	10	10	10

d)

Q_1Q_2	q \ AB	00	01	11	10
0 0	I.	(a)	b	IV	(a)
0 1	IV.	–	–	c	a
1 1	III.	(c)	(c)	(c)	IV
1 0	II.	c	(b)	(b)	(b)

c)

12-118. ábra *Állapotkódolás az instabil állapotok módosításával*

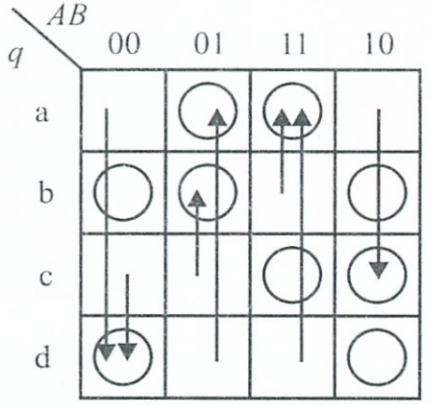
ugyan közvetetté válik és a hálózat elveszti a normál típusú jellegét, viszont a kérdéses helyeken a kritikus versenyfutás veszélye megszűnik. A módosított állapottáblát a 12–118c. ábrán, a végleges kódolt állapottáblát a 12–118d. ábrán rajzoltuk fel.

F.8.13. A 12–100. ábrából látható, hogy egy MOORE-modell kínálkozik közvetlen leírásként, mivel a kimenetek a belső állapotokhoz vannak rendelve a bemeneti állapotoktól függetlenül. A gráfnak megfelelő állapottáblát, illetve átmeneti táblát a 12–119a., illetve a 12–119b. ábrán rajzoltuk fel. Mind a kiindulógráf, mind a kapott táblák tanulmányozása során hamarosan kiderül, hogy szomszédos kódolást ilyen formában nem tudunk megvalósítani. Amennyiben viszont bevezetjük az „1" és „2" jelű közbenső állapotokat a 12–119c. ábra szerint módosított állapotgráfnak és a 12–119d. ábra szerinti állapotkódolásnak megfelelően, akkor a kritikus versenyfutás veszélye megszűnik, mivel a szomszédosság így már teljesül. Az új viszonyoknak megfelelő átmeneti tábla-módosulást a 12–119e. ábrán tüntettük fel. Itt a kihangsúlyozás érdekében a 12–119b. ábrához viszonyítva változatlan átmeneteket nem tüntettünk fel. Az aktuálisan kitöltött kódolt állapottáblát a 12–119f. ábra mutatja. A ki nem használt esetek kitöltésénél ügyelni kell a korábban már említett esetleges tranziensek elkerülésére.

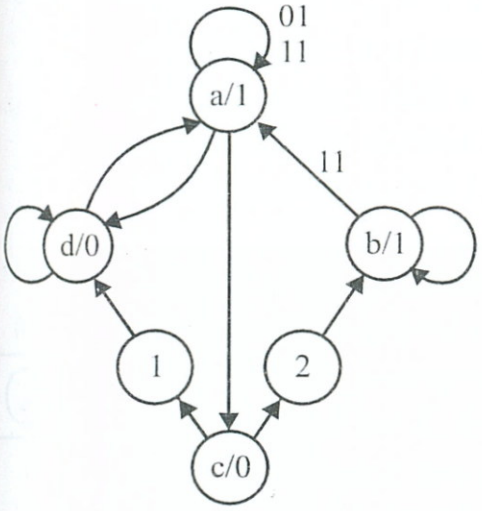
F.8.14. A kiinduló 12–101. ábrán látható állapottábla alapján felrajzoltuk a 12–120a. ábra szerinti állapotgráfot, melynek tanulmányozása nyomán kiderül, hogy a közvetlen kapcsolatban álló: *a-b-c* és *b-c-d* gráfrészletek eleve lehetetlenné teszik a szomszédossági feltételek teljesítését, így kritikus versenyfutás veszélyét rejtik magukban. A szomszédosság biztosítására választhatnánk például az előző feladatban alkalmazott közbenső instabil állapotok beiktatásának módszerét is, de jelen példánál megtakarítható új állapotok bevezetése, ha a 12–120b. ábra szerinti kódolás feltételezésével a 12–120a. ábra állapotgráfján a közvetlen ⓒ → ⓑ átmenetet megszüntetjük és a ⓒ → d → ⓑ közvetett átmeneten keresztül biztosítjuk a kapcsolatot (12–120c. ábra).
Ez a változás a 12–120d. és e. ábrák eredeti, ill. a módosításokat mutató átmeneti tábláinak összehasonlításakor szem-

q \ AB	00	01	11	10	Z
a	d	(a)	(a)	c	1
b	(b)	(b)	a	(b)	1
c	d	b	(c)	(c)	0
d	(d)	a	a	(d)	0

a)



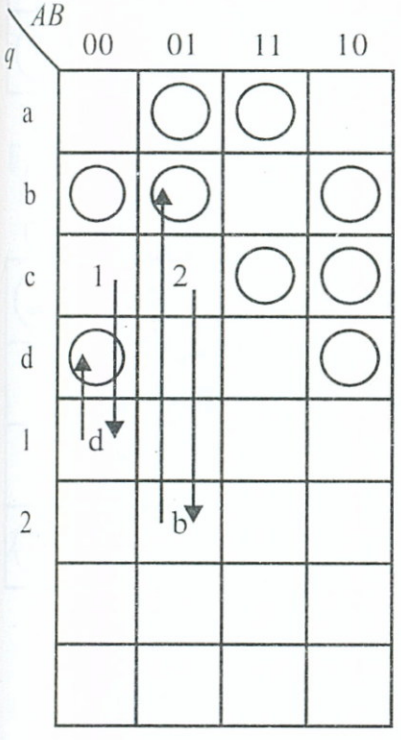
b)



c)

	Q_1	Q_2	Q_2
a	0	1	0
b	1	1	0
c	0	0	0
d	0	1	1
1	0	0	1
2	1	0	0

d)

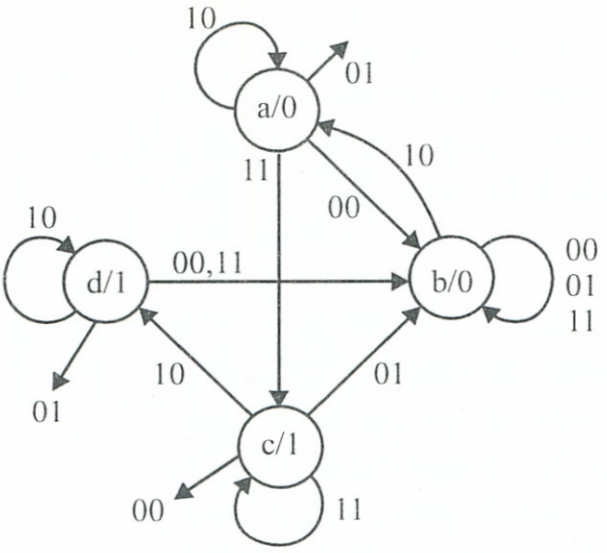


e)

$Q_1Q_2Q_3$ \ AB	00	01	11	10	Z
0 1 0	011	010	010	000	1
1 1 0	110	110	010	110	1
0 0 0	001	101	000	000	0
0 1 1	011	010	010	011	0
0 0 1	011	–	–	–	0
1 0 0	–	110	–	–	–
1 0 1	–	–	–	–	–
1 1 1	–	–	–	–	–

f)

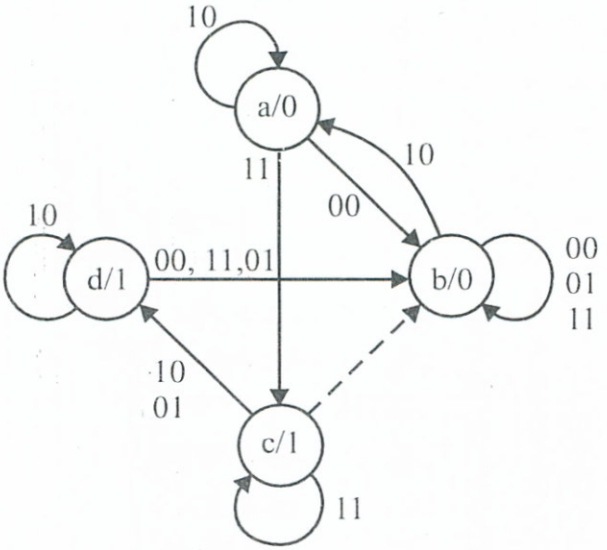
12-119. ábra *Közbenső állapotok felvétele az F.8.13. feladatnál*



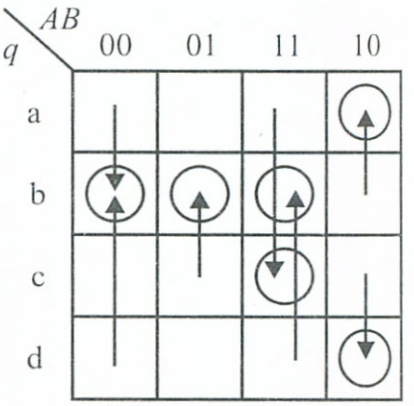
a)

	Q_1 Q_2
a	0 1
b	0 0
c	1 1
d	1 0

b)



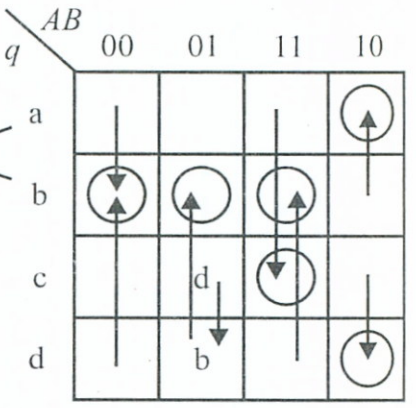
c)



d)

Q_1Q_2 \ AB	00	01	11	10	Z
b 00	(00)	(00)	(00)	01	0
a 01	00	–	11	(01)	0
c 11	–	10	(11)	10	1
d 10	00	00	00	(10)	1

f)



e)

12-120. ábra *Állapotkódolás instabil esetek módosításával*

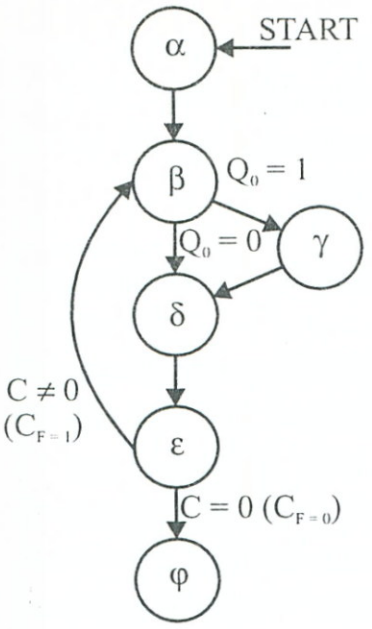
betűnő. A végleges, kritikus versenymentes kódolt állapottábla a 12–120f. ábrán látható. Itt ügyeljünk arra, hogy a V–K tábla kötött kód-elrendezése miatt az első két sornál „*a*" és „*b*" állapotok helye felcserélődött.

F.8.15. A feladatmegoldásnál felhasználjuk a 8.20. Példa tapasztalatait.
A 12–121a. ábrán felrajzoltuk a kiinduló-állapotgráfot, ebből a 12–121b. előzetes állapottábla előállítható. Itt a 10–62. ábrabeli *C* számláló *üres,* ill. *még nem üres* állapotait egy C_F flag $C_F = 0$, $C_F = 1$ állapotaival jelképeztük. Az egyszerűség kedvéért most nem törekedve szisztematikus állapotkódolásra, bevezetjük a c. ábra q_t rovatában ábrázolt egyszerű bináris kódot, mellyel kitöltjük a c. teljes kódolt állapottáblát, ahol a választott J–K tárolók vezérlő bemenetei is szerepelnek. A kérdéses J_i, K_i értékeket a d. ábrában újra-felrajzolt J–K-ra vonatkozó vezérlési táblázat segítségével számítottuk ki, figyelembevéve a feltételes átmeneteknél szereplő Q_0, C_F feltételek szerepét is. Végül az e. ábrarészen felírtuk a kiszámított J_i és K_i értékeket, melyekkel a 8–61. ábrán szereplő hálózathoz hasonló, vezérlő egységünk már felrajzolható. A jelenlegi példa és a 8–61. ábra realizációs vázlatai között eltérés csupán a 8–61. ábra KOMBINÁCIÓS HÁLÓZAT nevű egységénél jelentkezik az e. ábrán kiszámított függvényeknek megfelelően.

F.8.16. A feladatmegoldásnál felhasználjuk a 8.19. Példa tapasztalatait. A 12–122a. ábrán felrajzoltuk a kiinduló állapotgráfot. Ennek alapján a ROM-ba beégetendő és az aktuális feladathoz illeszkedő mikroprogram már összeállítható. Az esetünkre alkalmazott mikroprogramot a 12–122b. ábrán rajzoltuk fel. A feltételes ugrásoknál az alábbi értelmezést használtuk:

$JC_x =$ *INC* ha: $X = 0$ (INKREMENTÁLÁS)
JMP ha: $X = 1$ (UGRÁS A MEGADOTT ROM-CÍMRE)

A feladat további lépéseit már az említett 8.19. Példa alapján követhetjük. Az ott felrajzolt 8–59. ábra realizációs hálózata teljesen hasonló a jelen példa itt fel nem rajzolt realizációjához. Eltérés két helyen van:



a)

q_t	q_{t+1} Felt. nélk.	$Q_0=0$	$Q_0=1$	$C_F=0$	$C_F=1$
α	β	–	–	–	–
β	–	δ	γ	–	–
γ	δ	–	–	–	–
δ	ε	–	–	–	–
ε	–	–	–	φ	β
φ	–	–	–	–	–

c)

q_t	$Q_1Q_2Q_3$	Felt. nélk.	q_{t+1} $(Q_1Q_2Q_3)_{t+1}$ $Q_0=0$	$Q_0=1$	$C_F=0$	$C_F=1$	Flip-flop vezérlések J_1K_1	J_2K_2	J_3K_3
α	0 0 0	0 0 1	–	–	–	–	0 h	0 h	0 h
β	0 0 1	–	0 1 1	0 1 0	–	–	0 h	1 h	h 0 / h 1 } Q_0
γ	0 1 0	0 1 1	–	–	–	–	0 h	h 0	1 h
δ	0 1 1	1 0 0	–	–	–	–	1 h	h 1	h 1
ε	1 0 0	–	–	–	1 0 1	0 0 1	h 0 / h 1 } C_F	0 h	1 h
φ	1 0 1	–	–	–	–	–	h h	h h	h h

c)

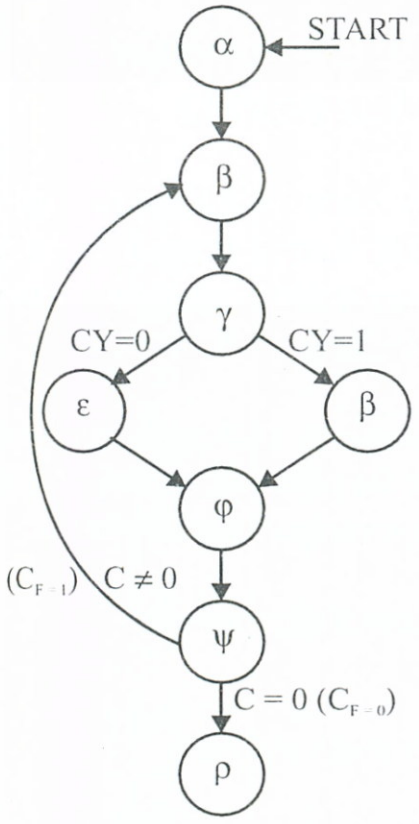
$Q_t \rightarrow Q_{t+1}$		J K
0	0	0 h
0	1	1 h
1	0	h 1
1	1	h 0

d)

$J_1 = \delta$
$K_1 = \varepsilon \cdot C_F$
$J_2 = \beta$
$K_2 = \delta$
$J_3 = 1$
$K_3 = \delta + \beta \cdot Q_0$

e)

12-121. ábra *Fázisregiszteres vezérlés tervezése szorzóberendezésekhez*



a)

RELATÍV ROM CÍM	ÁLLAPOT	CÍMZÉSI MÓD	VEZÉRLŐ KIMENET V_i	ROM UGRÁS CÍM
0	α	INC	V_α	–
1	β	INC	V_β	–
2	γ	JC_{CY}	V_γ	CIMδ(7)
3	ε	INC	V_ε	–
4	φ	INC	V_φ	–
5	ψ	JC	V_ψ	CIMβ(1)
6	ρ	–	V_ρ	–
7	δ	JMP	V_δ	CIMφ(4)

b)

12-122. ábra *Mikroprogramozott vezérlés tervezése osztóberendezéshez*

1) A ROM-ba beégetendő kódolt mikroprogramnál, mely esetünkben a 12–122b. ábra szerint írható fel.

2) Az ún. DÖNTÉSI LOGIKÁ-nál, melyben a kombinációs hálózatot mindig a mikroprogramban előforduló ugrások alapján kell megtervezni.

12.9.1. Gyakorló feladatok a 9. „Különleges digitális hálózatok" c. fejezethez

*G.9.1. Tervezzük meg MOS reiteratív kapcsolóhálózattal azt a négy nyomógombos berendezést, amely lámpát gyújt ki azokban az esetekben, ha egy vagy kettő, vagy három-nyomógombot nyomunk meg egyidejűleg.

*G.9.2. Alakítsunk ki olyan 6-nyomógombos hálózatot, melynek kimenetén P piros lámpa világít, ha páratlan számú, Z zöld lámpa világít, ha páros számú nyomógombot nyomunk.

*G.9.3. Adottak a következő szimmetrikus függvények:

$$Y_1 = F^5_{(0)(4)(5)}$$
$$Y_2 = F^5_{(1)(2)(3)(5)}$$
$$Y_3 = F^5_{(2)(3)}$$

a) Milyen eredő függvények keletkeznek az alábbi esetekben?

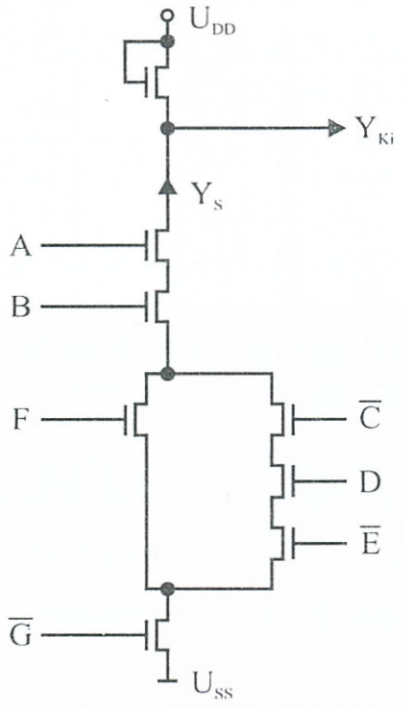
$$Y_\alpha = Y_1 + Y_2$$
$$Y_\beta = Y_1 + Y_3$$
$$Y_\gamma = Y_1 \cdot Y_2$$
$$Y_\delta = Y_1 \cdot Y_3$$

b) Miként fejezhető ki kevesebb skalár jellemzős alakban az Y_2?
c) Miként írható fel negált változókkal az Y_3?
d) Miként írható fel negált alakban az Y_3?

*G.9.4. Próbáljuk MOS–TRANSFER GATE-t felhasználó HÍD-típusú hálózattal realizálni az alábbi függvényt:

$$Z = \overline{Y} = \overline{\overline{E} \cdot (\overline{D} \cdot C \cdot \overline{B} \cdot A + D \cdot \overline{C} \cdot B \cdot A)}$$

*G.9.5. A 12–123. ábrán látható MOS-hálózat hibája, hogy sok sorbakapcsolt tranzisztort tartalmaz, ez pedig a vezérlő feszültségek szempontjából kedvezőtlen. Alakítsuk át úgy a



12-123. ábra *Kiindulás a G.9.5. feladathoz*

kapcsolást – ha lehet – úgy, hogy a sorbakapcsolt tranzisztorok száma ne haladja meg a hármat.

*G.9.6. Keressünk a felsorolt egyszerű Boole-függvényekhez küszöblogikás realizációkat:

$$Y_\alpha = A \cdot B \qquad Y_\delta = A + B$$
$$Y_\beta = A \cdot \overline{B} \qquad Y_\varepsilon = \overline{A + B}$$
$$Y_\gamma = \overline{A} \qquad Y_\varphi = A \oplus B$$

*G.9.7. Milyen küszöb-hálózattal realizálható a következő Boole-függvény?

$$Y = A \cdot \overline{B} \cdot C + AB$$

*G.9.8. Állítsuk elő a 9–26. ábra logikai táblázatával megadott 3-értékű logikás, kétváltozós függvény komplemensét.

*G.9.9. Állítsuk elő és ábrázoljuk a 9–34. ábrán szereplő „ALACSONY" (A) fuzzy halmaz komplemensét.

*G.9.10. Írjuk fel az előző feladat A^c-re a következőket: core, sup, height, $\alpha = 0{,}6$ vágat, kardinalitás.

*G.9.11. Állítsuk elő a 9–34. ábra ALACSONY (A) és MAGAS (M) halmazoknál az alábbiakat.

$$A \cup M, \quad A \cap M, \quad M^c$$

*G.9.12. Készítsünk példát az általánosított GMP-re.

*G.9.13. Milyen összefüggések írhatók fel az R fuzzy reláció és ennek ínverze között?

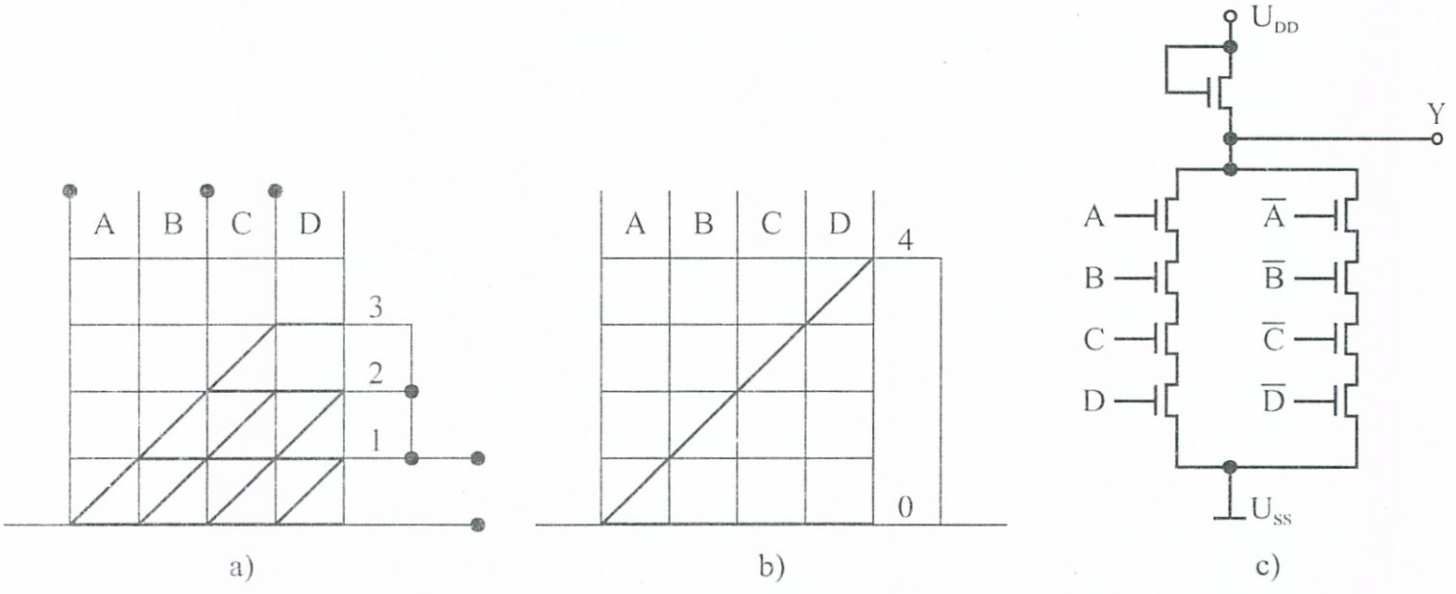
*G.9.14. Ábrázoljuk egységkockán a következőket:

a.) $A(0{,}7(x_1))$ valamint: A^c
b.) $B(0{,}9(x_1), 0{,}2(x_2))$ valamint: B^c

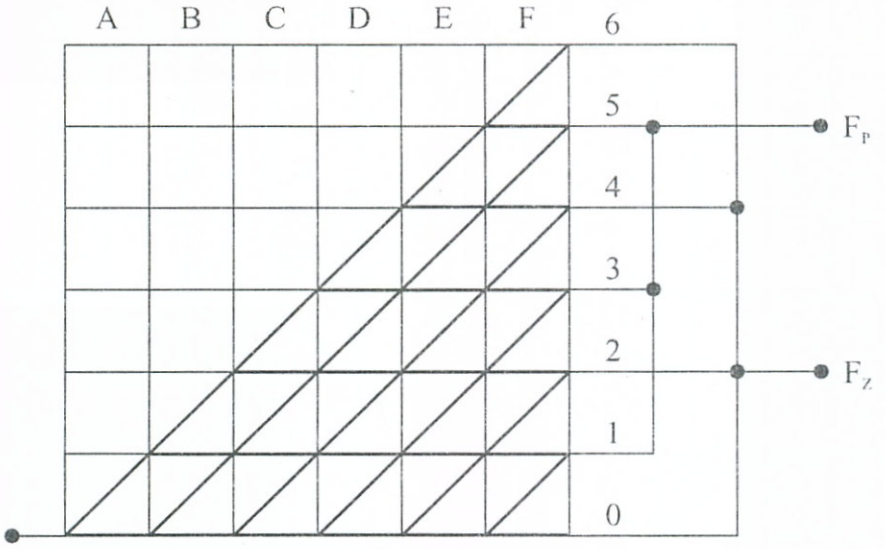
12.9.2. Feladatmegoldások a 9. fejezethez

F.9.1. Szimmetrikus hálózatról van szó, melynek példa szöveg alapján felrajzolt gráfja a 12–124a. ábrán látható. Mivel a MOS realizáció miatt úgy is a negált kimenet szükséges, célszerű a negált gráfot képezni (12–124b. ábra), melynek alapján a realizációs áramkör a 12–124c. szerint alakul.

F.9.2. Ez a feladat egy tipikus reiteratív hálózatra való eset, és szimmetrikus függvényekkel jól leírható. A szöveg alapján felírhatjuk, ha a „0" db gomb megnyomását is *páros* esetnek tekintjük:



12-124. ábra *MOS szimmetrikus hálózat a G.9.1. feladathoz*



12-125. ábra *Kiindulás az F.9.2. feladatmegoldáshoz*

$$Y_P = F^6_{(1)(3)(5)} = F^6_P(A,B,C,D,E,F)$$

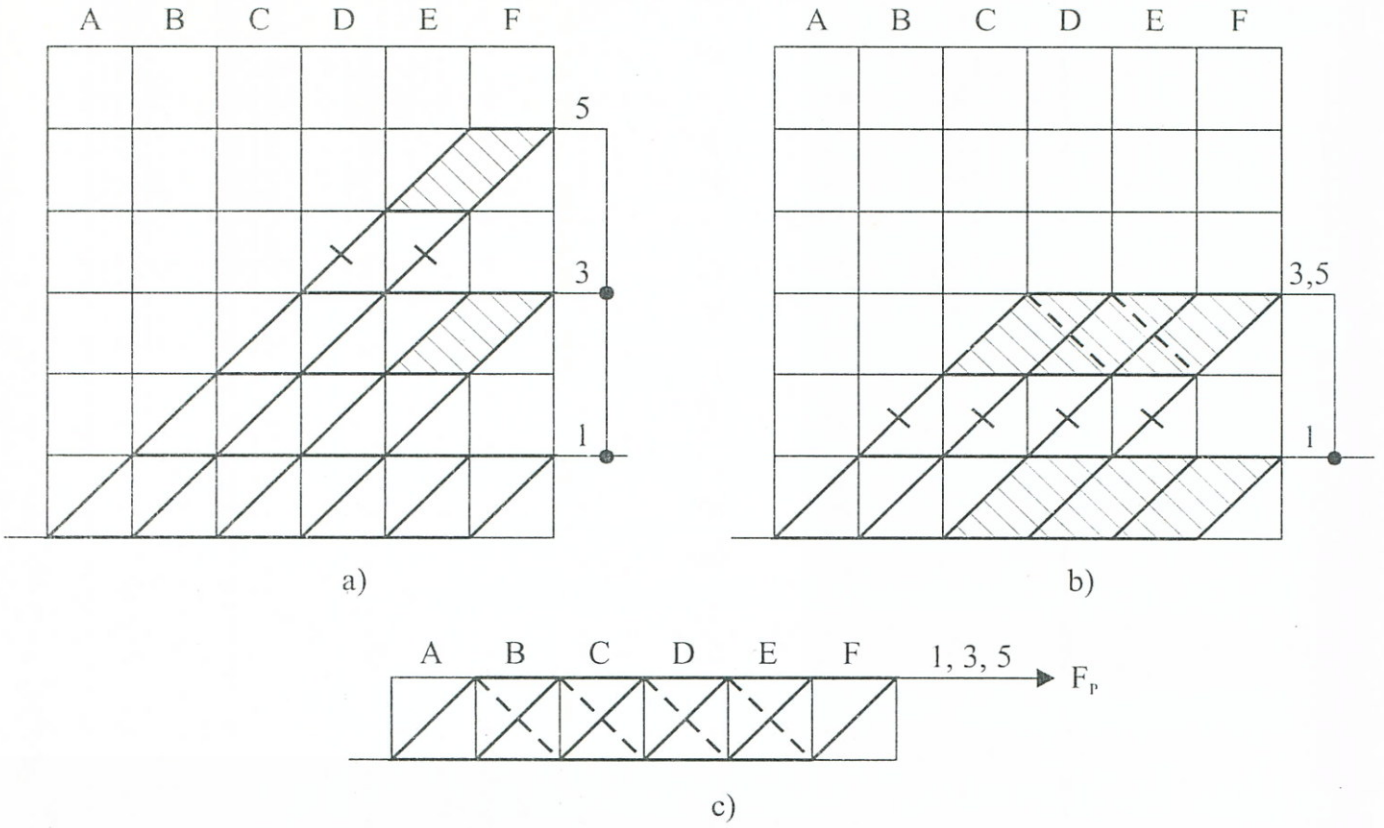
$$Y_Z = F^6_{(0)(2)(4)(6)} = F^6_Z(A,B,C,D,E,F)$$

A tervezést a 8.1.2. pontban bemutatottak alapján végezhetjük. Kiinduláskor a két függvényt egymástól függetlennek is tekinthetjük, de a 6-változós szimbólikus gráf felrajzolása után már látható, hogy a két realizációs hálózatnak lesznek közös részhálózatai (12–125. ábra). Különválasztva az F_P és F_Z megoldást, mindkét esetben alkalmazhatjuk a 9.1. Példában megismert „lecsúsztatási" és „forgatási" műveleteket.

F_P-nél a 12–126a. ábrán a sraffozott területeket egymásba *csúsztathattuk,* ezáltal az „5" és „3" kimenetek túlfedésbe kerültek. A 12–126b. ábrán az előző eredményt tovább alakíthattuk az itteni sraffozott területek újabb egymásba *csúsztatásával,* így végül a 12–126c. ábrabeli, a kiindulónál lényegesen egyszerűbb gráfhoz jutottunk, melynél a rácsrudak száma sokkal kevesebb, mint a kiindulógráfnál, azaz a kapcsolóhálózat kevesebb elemmel megépíthetővé vált.

F_Z-nél a 12–127a., b. ábrákon *csúsztatás,* míg a c.) d.) ábrákon egy *forgatás* tette lehetővé az egyszerűsítést.

A 12–126c. és 12–127d. ábrák összehasonlításából látható, hogy az F kimeneti fokozat kivételével egybevágóak, így végül a 12–127e. ábra szerint egyesíthetők.



12-126. ábra *Az F_P függvény egyszerűsítése szimbolikus gráf segítségével*

A hálózat realizációja történhet MOS vagy érintkező hálózattal. A hálózattá transzformálásnál – mint korábban láttuk – a gráf ferde vonalai *záró-*, a vízszintes vonalak pedig *bontó* kapcsolót jelképeznek.

F.9.3. A feladatok megoldásánál a 9. fejezetben lèvezetett (9.3) összefüggéscsoportot használjuk fel.

a) $Y_\alpha = Y_1 + Y_2 = F^5_{(0)(4)(5)} + F^5_{(1)(2)(3)(5)} =$

$= F_{(0)\,(1)\,(2)\,(3)\,(4)\,(5)} = 1$

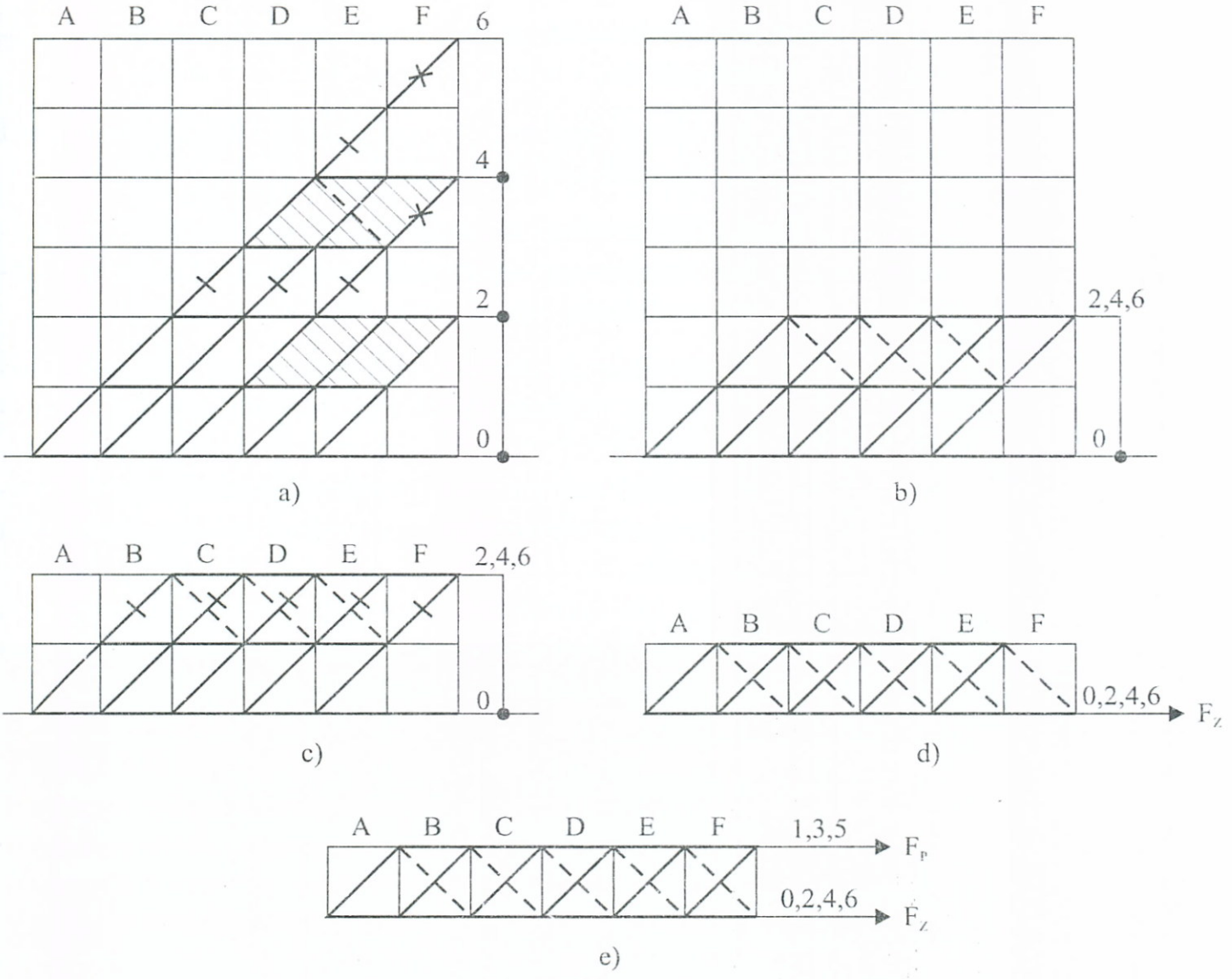
$Y_\beta = Y_1 + Y_3 = F^5_{(0)(4)(5)} + F^5_{(2)(3)} =$

$= F^5_{(0)(2)(3)(4)(5)} = \overline{F^5_{(1)}}$

$Y_\gamma = Y_1 \cdot Y_2 = F^5_{(0)(4)(5)} \cdot F^5_{(1)(2)(3)(5)} = F^5_{(5)}$

$Y_\delta = Y_1 \cdot Y_3 = F^5_{(0)(4)(5)} \cdot F^5_{(2)(3)} = 0$

b) $Y_2 = F^5_{(1)(2)(3)(5)} = \overline{F^5_{(0)(4)}}$



12-127. ábra *Az F_Z függvény egyszerűsítése szimbolikus gráf segítségével*

c) $Y_3 = F^5_{(2)(3)}(X_1, X_2, X_3, X_4, X_5) =$

$= F^5_{(0)(1)(4)(5)}(\overline{X}_1, \overline{X}_2, \overline{X}_3, \overline{X}_4, \overline{X}_5)$

d) $\overline{Y}_3 = \overline{F^5_{(2)(3)}} = F^5_{(0)(1)(4)(5)}$

F.9.4. A feladatmegoldásnál a 9.1.3a. pontban elmondottakból indulunk ki, de előzetesen célszerű néhány további meggondolást is tennünk.

A híd és a soros–párhuzamos struktúra topológiai eltérését a szokásos algebrai módszerekkel nem tudjuk egyértelműen kifejezni, ezért célszerű olyan leírásmódot keresni, mely a topológiát is tartalmazza. Ilyen adottságokkal rendelkezik – mint a 9.1.3. pontban már utaltunk rá – az ún. *közvetlen*

összeköttetési táblázat, melyet a következő kritériumok alapján képezünk:

a) A hálózat minden csomópontjához a táblázaton egy sort és egy oszlopot rendelünk.

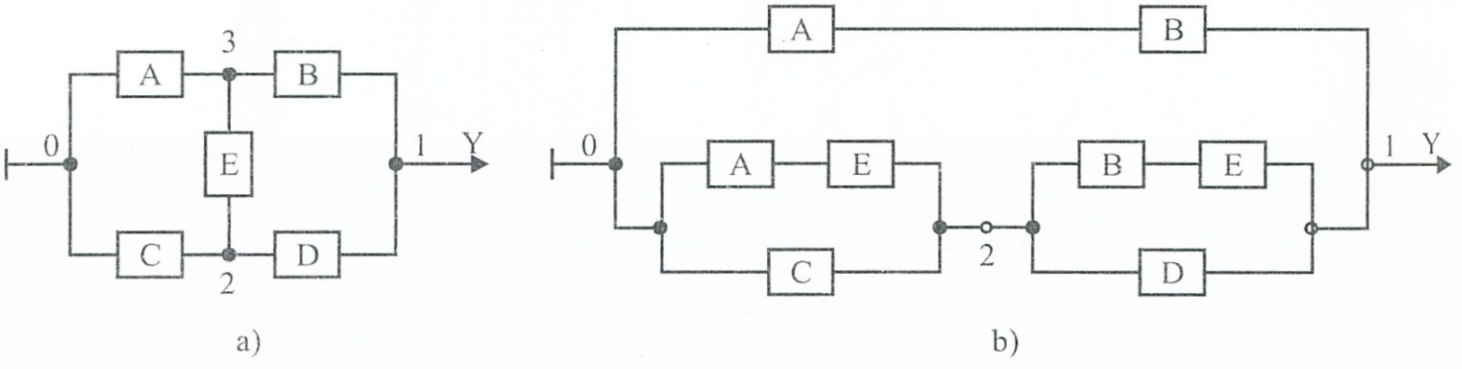
b) A táblázat minden eleméhez egy, a sorához, ill. oszlopához tartozó csomópontok összeköttetéseit jellemző adatot rendelünk, így:

- rövidzárral csatlakozó két csomópont táblázatbeli sor-oszlop találkozásainál elhelyezkedő elembe „1"-et írunk;
- tetszés szerinti K-kapcsolóval összekötött két csomópont táblázatbeli sor-oszlop találkozásaihoz K-t írunk;
- ha két csomópont közt csak további csomóponton vagy csomópontokon keresztül lehet összeköttetést létesíteni, akkor a megfelelő táblázatelembe „0" kerül.

c) A táblázat tükrös a főátlóra, ha a hálózat kapcsoló elemei bilaterálisok (pl. MOS-tranzisztor) és nem tükrös, ha az elemek közt *unilaterálisok* (pl. dióda) is szerepelnek.

Az elmondottak figyelembevételével például a 12–128a. ábra hálózatát logikailag és topológiailag is jellemző közvetlen összeköttetési táblázat a 12–129a. ábra szerinti lesz.

Az összeköttetési táblázat tulajdonságainak további tanulmányozása céljából „szüntessük meg" a 12–128a. ábrán a 3-as csomópontot, természetesen olyan kikötéssel, hogy az így keletkező hálózat tulajdonságai „kívülről nézve" a logi-



12-128. ábra *Csomópont megszüntetési lépés*

kai működés szempontjából változatlanok maradjanak. Ez akkor lesz lehetséges, ha azokat a „kerülő utakat", melyek a 0-csomópontból kiindulva 3-on keresztül 1-be vezettek, és melyek 3-as kiiktatásával megszűnnek, új, közvetlen összeköttetéssel pótoljuk. Ily módon a 12–128b. hálózat keletkezik, melyhez tartozó újabb összeköttetési táblázat a csomópont-szám csökkenése miatt egy rendszámmal alacsonyabb lesz és a 12–129b. ábra szerint alakul.

	0	1	2	3
0	1	0	C	A
1	0	1	D	B
2	C	D	1	E
3	A	B	E	1

a.)

	0	1	2
0	1	$0 + AB$	$C + AE$
1	$0 + AB$	1	$D + BE$
2	$C + AE$	$D + BE$	1

b.)

12–129. ábra *Összeköttetési táblázat*

Az új táblázat elemeit szándékosan írtuk $u + v \cdot w$ alakban, ugyanis így az ábrák és a hozzájuk tartozó összeköttetési táblázatok között felismerhető az alábbi általános törvényszerűség:

Valamely „i" csomópont megszüntetésekor a hálózat logikai funkciója akkor nem változik meg, ha a megszűnő ágakat egyidejűleg új, közvetlen összeköttetéssel pótoljuk. Ez az összeköttetési táblázatban úgy jut kifejezésre, hogy:

a) a megszűnt csomópontnak megfelelő „i" sort és oszlopot az új táblázatban már elhagyjuk;

b) a megszűnő elemek sor-oszlop találkozásában a régi összeköttetési táblázat f''_{jk} elemét a következőképpen „bővítjük" az új táblázat f'_{jk} elemévé:

$$f'_{jk} = f''_{jk} + f''_{ji} \cdot f''_{ik}$$

Az összefüggésben f''_{jk} a „j" és „k" csomópontok közt már korábban is fennálló összeköttetést jelképezi, míg $(f''_{ji} \cdot f''_{ik})$

az újonnan létesített közvetlen összeköttetést fejezi ki a redukált hálózatnál.

Fenti formulánkat a 12–128. és 12–129. ábrák esetére alkalmazva, az $i = 3$ megszüntetése után például $j = 2$ és $k = 1$ csomópontok közötti összeköttetést eredetileg megadó

$$f''_{jk} = f''_{21} = D \text{ elem helyébe}$$

$$\left.\begin{aligned} f''_{ji} &= f''_{23} = E \\ f''_{ik} &= f''_{31} = B \end{aligned}\right\} \text{ révén}$$

az új, alacsonyabb rendű táblázatba bekerülő elem:

$$f'_{jk} = f'_{21} = f'_{12} = f''_{21} + f''_{23} \cdot f''_{31} = D + E \cdot B$$

ami valóban egyezik 12–129b. ábra megfelelő elemével.

Az előzők alapján fordított irányban gondolkodva összerakhatók egy

soros–párhuzamos → híd

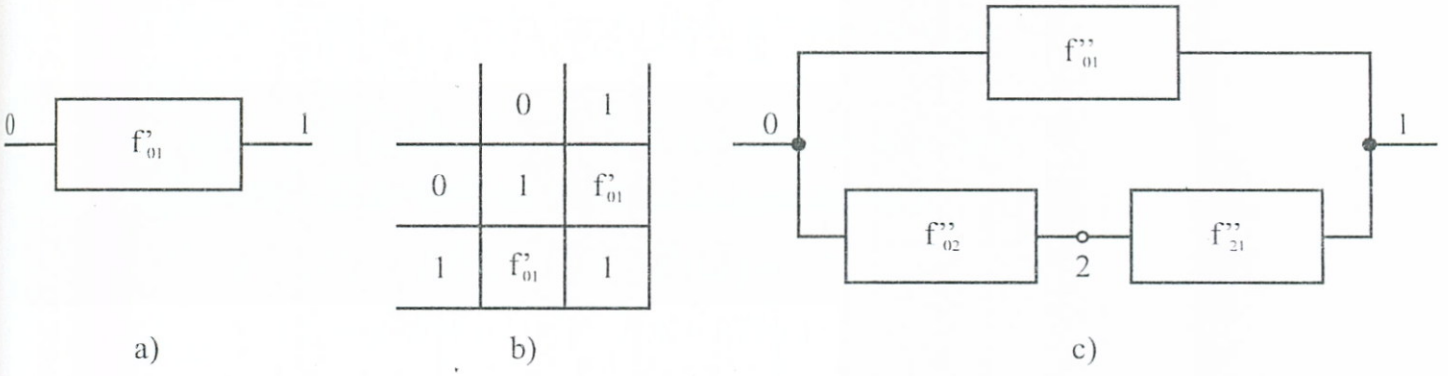
típusú szintézis főbb lépései, mellyel – várhatóan – egyszerűbb kombinációs kapcsolóhálózatot sikerül kialakítanunk.

Kiinduláskor szükséges lehet a híd-struktúra képzésének feltétel-vizsgálata. Mivel nem minden függvény „hidasítható", ezért meg kell vizsgálnunk a következőket:

a) Létezni kell a diszjunktív alakban négy olyan tagnak, melyben két, célszerűen megválasztott „kezdő" és két „végző" elemnek minden lehetséges kombinációja szerepel, kivéve a kezdő–kezdő és végző–végző kombinációs eseteket.

b) A kezdő és végző elemek szorzóként történő kiemelése folytán keletkező két tényezőben, az azonos változók csak meghatározott kombinációkban szerepelhetnek: $(u + v \cdot w)$, ill. $(w + u \cdot v)$ formában.

Például a 12–128a. ábránál a kezdő elemek A és C, a végző elemek B és D, a szorzatkombinációk: AB, CD, $(AD)E$, $(BC)E$ és nem szerepel AC, ill. BD alkotókat is tartalmazó



a) b) c)

12-130. ábra *Csomópont bővítés*

tag. Továbbá, ha a 12–128a. ábrán szereplő Y alaknál A-t és C-t kiemeljük, kapjuk a b) pontnak megfelelően:

$$Y_k = A(B+ED) + C(D+EB)$$

Valamely kiinduló függvényalak esetén a kezdő- és végelemek nem mindig találhatók meg közvetlen felismeréssel, emiatt ezek keresésére több szisztematikus eljárást is kidolgoztak, melyekkel e könyvben nem foglalkozunk.

A híd kialakítás lépései a következők:

I. lépés: Primitív (kiinduló) összeköttetési táblázat felállítása, majd szisztematikus bővítése. A primitív összeköttetési táblánál:

$$F_k^n = f_{01}^{'} = f_{10}^{'} \qquad *$$

A tábla a 12–130b. ábra szerinti lesz.

II. lépés: Összeköttetési táblázat-bővítés egy újabb csomópont felvételével. Ehhez az F_k^n függvényt a következő alakra rendezzük:

$$F_k^n = f_{01}^{'} = f_{01}^{''} + f_{02}^{''} \cdot f_{21}^{''} \qquad **$$

Ebből a bővített újabb összeköttetési táblázat kitölthető, és megfelel a 12–130c. ábra hálózatának.

III.-(N-1). lépések: Újabb és újabb bővítések a II. lépéshez hasonlóan, amíg van lehetőség.

N. lépés: Végleges hálózat felrajzolása.

Az előző algoritmus megismerése után már nekifoghatunk a G.9.4. feladat tényleges megoldásának. Újra felírva:

$$Z = \overline{Y} = \overline{\overline{E}(\overline{D}C\overline{B}A + D\overline{C}B\overline{A})}$$
$$= \overline{\overline{E}\,\overline{D}C\overline{B}A + \overline{E}D\overline{C}B\overline{A}}$$

A MOS realizáció miatt az *Y*-ra végezzük el a hidasítást.

	0	1
0	1	$\overline{E}\,\overline{D}C\overline{B}A + \overline{E}D\overline{C}B\overline{A}$
1	$\overline{E}\,\overline{D}C\overline{B}A + \overline{E}D\overline{C}B\overline{A}$	1

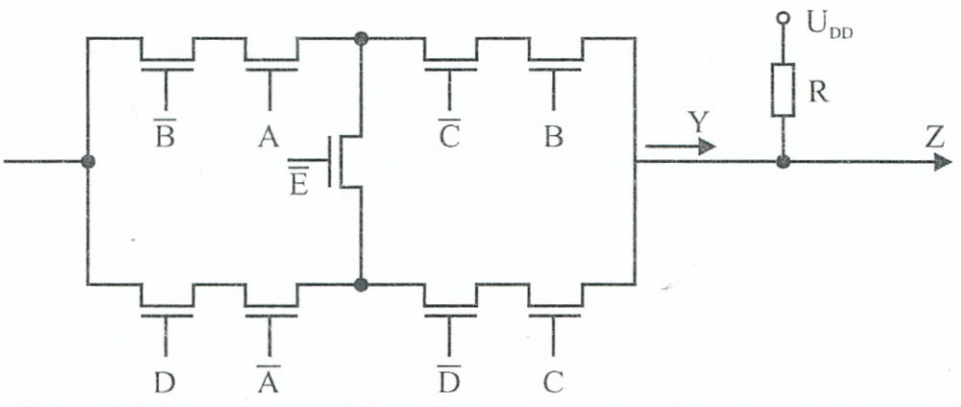
a)

	0	1	2
0	1	0	$\overline{B}A + \overline{E}D\overline{A}$
1	0	1	$\overline{C}B + \overline{E}\,\overline{D}C$
2	$\overline{B}A + \overline{E}D\overline{A}$	$\overline{C}B + \overline{E}\,\overline{D}\,C$	1

b)

	0	1	2	3
0	1	0	$\overline{B}A$	$D\overline{A}$
1	0	1	$\overline{C}B$	$\overline{D}C$
2	$\overline{B}A$	$\overline{C}B$	1	$\overline{E}$
3	$D\overline{A}$	$\overline{D}C$	$\overline{E}$	1

c)



d)

12-131. ábra *Híd struktúra kialakítása szisztematikus módszerrel*

I. lépés: Primitív összeköttetési tábla a * összefüggés alapján (12–131a. ábra).

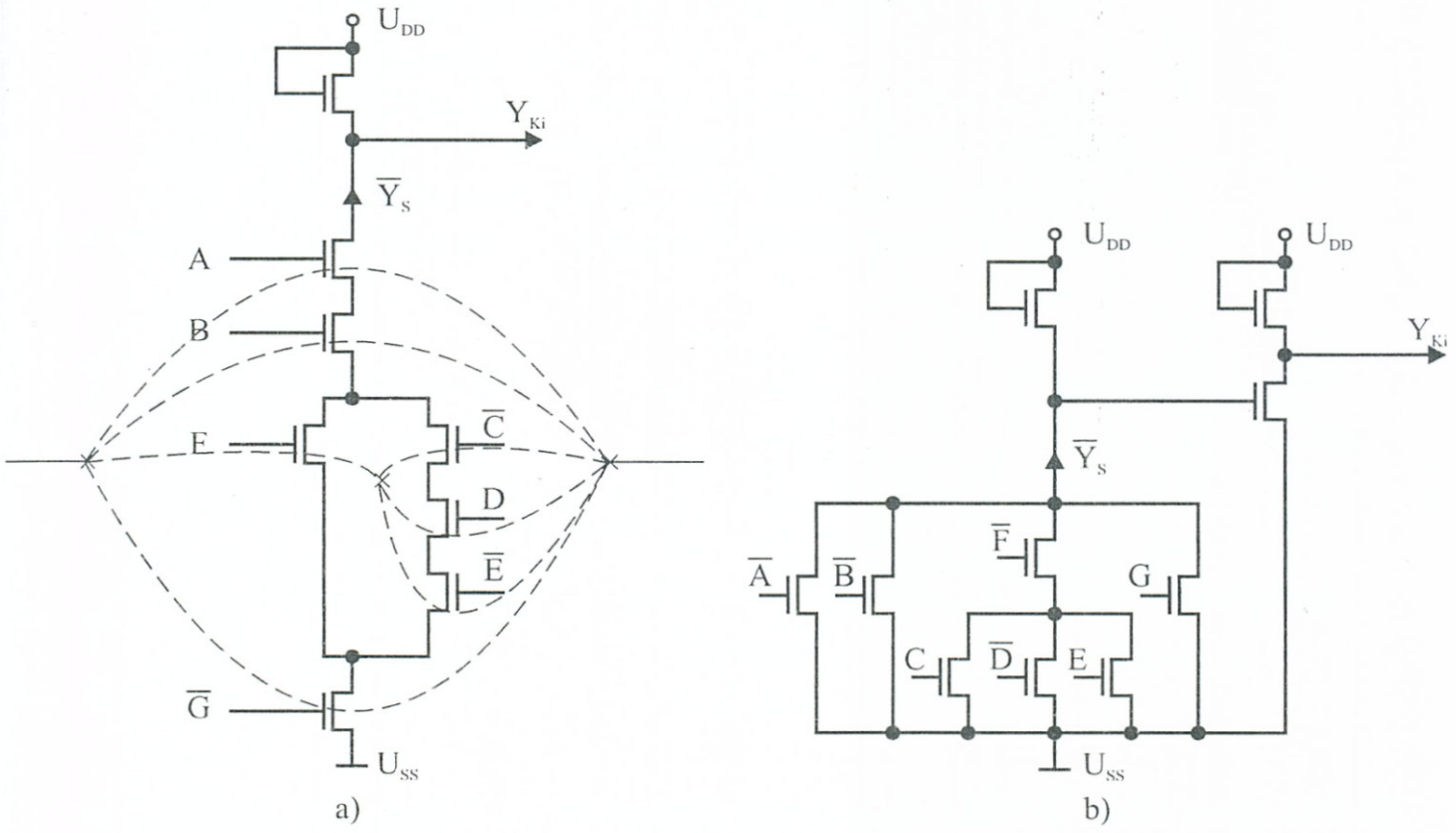
II. lépés: A függvényt olyan alakra kell hoznunk, hogy hasonlítson a ** összefüggéshez.

$$Y = (\overline{C}B + \overline{E}\overline{D}C)\cdot(\overline{B}A + \overline{E}D\overline{A}) =$$
$$= 0 + (\overline{C}B + \overline{E}\overline{D}C)\cdot(\overline{B}A + \overline{E}D\overline{A}) =$$
$$= f_{01}^{'} + f_{02}^{''}\cdot f_{21}^{''}$$

Ennek alapján az első bővített összeköttetési tábla a 12–131b. ábra szerint alakul.

III. lépés: Az $f_{02}^{''}$ és $f_{21}^{''}$ elemeket úgy is tovább fejthetjük, hogy ügyelünk arra, hogy az új, 4. sorba, illetve oszlopba a „tükrös" helyekre azonos kifejezések kerüljenek. Ilyen alapon felrajzolható a tovább már nem bővíthető 12–131c. ábra.

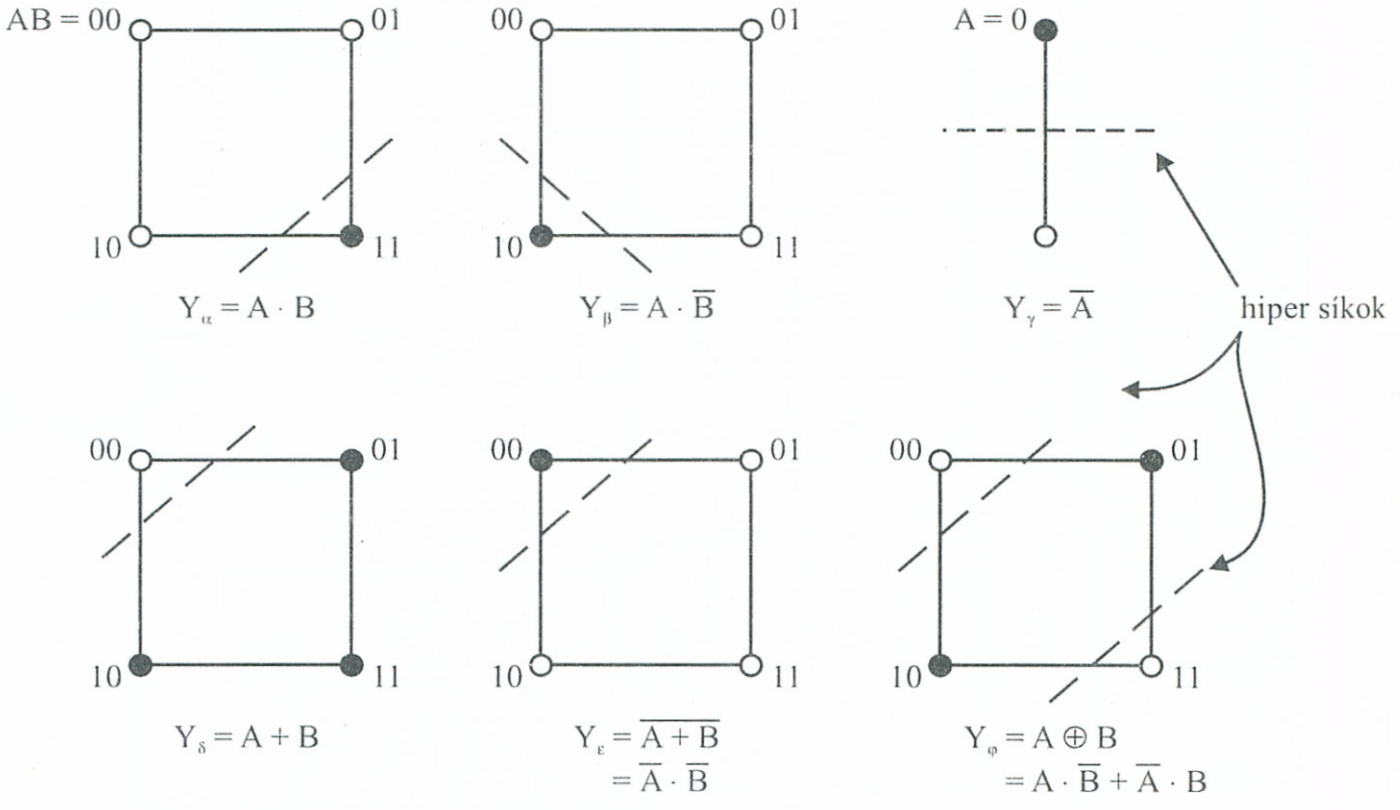
IV. lépés: A végleges híd-hálózat felrajzolása a 12–131d. ábrán történt.



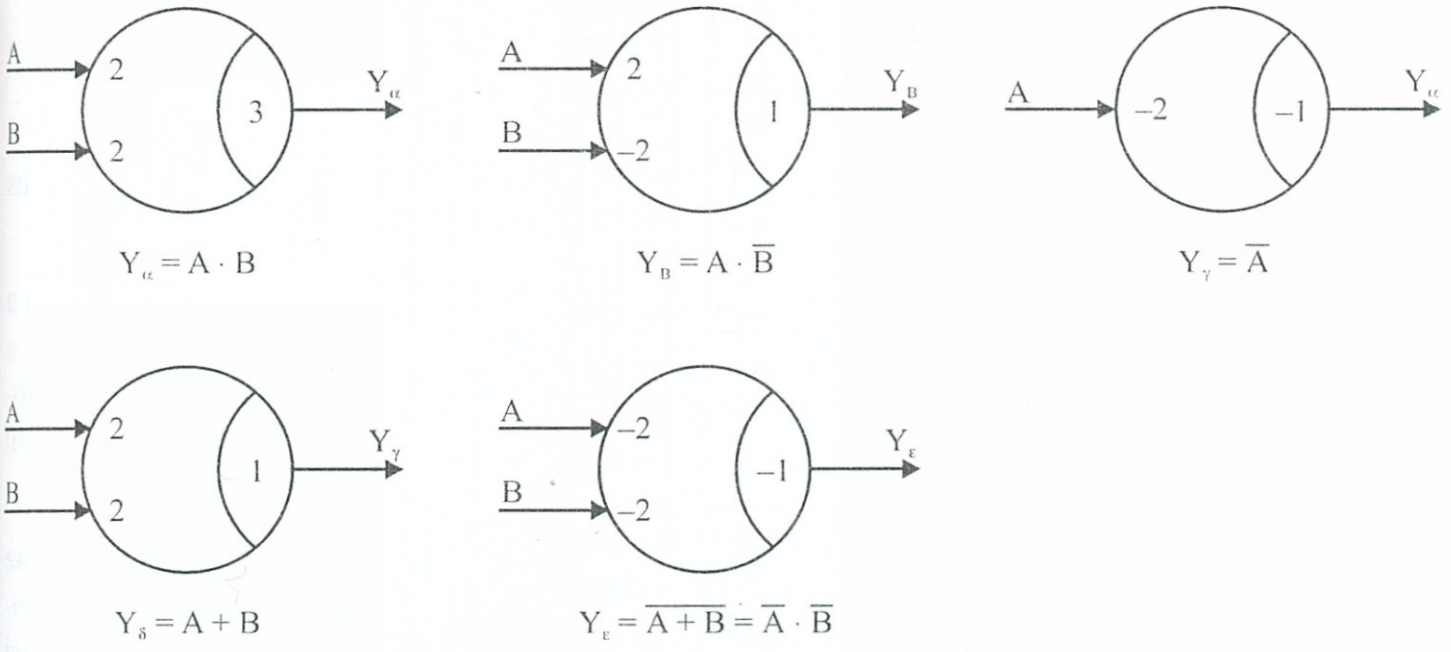
12-132. ábra *Áttérés kevesebb sorbakapcsolt elemre*

F.9.5. A megoldáshoz a 9–10. ábrán bemutatott grafikus eljárás segítségével juthatunk. A 12–132a. ábrán berajzoltuk a „duál" szerkesztési vonalakat, melyek segítségével megrajzolhattuk a 12–132b. ábrát, ahol a sorbakapcsolt aktív tranzisztorok száma csak kettő, így a példabeli célkitűzést még túl is teljesítettük. A kimenetre még egy invertert is kellett tenni, mivel a szerkesztés során a dualitás miatt $Y_s \rightarrow Y_s$ negált képződött és ezt vissza kellett fordítanunk.

F.9.6. Első lépésként meg kell vizsgálnunk, hogy az adott Boole-függvény *lineárisan szeparálható-e*? Ha igen: akkor egyetlen-, ha nem: akkor csak összetett küszöbelemmel lesz realizálható (a 9.5. Példa értelmében). Ezt a vizsgálatot esetünkre a 12–133. ábrán látható n = 2 és n = 1 dimenziós „kockákkal" végeztük el. Itt a kérdéses függvények „1"-es minterm-jeit a kocka *befeketített* csúcsain tüntettük fel, a „0"-ás minterm-eket *üres*-körös csúcsok jelzik. Amennyiben a befeketített és üres csúcsok egymástól egyetlen „hipersíkkal" elválaszthatók, akkor egyetlen küszöbelem elegendő a realizáláshoz, több hiper-síknál több elemre van szük-



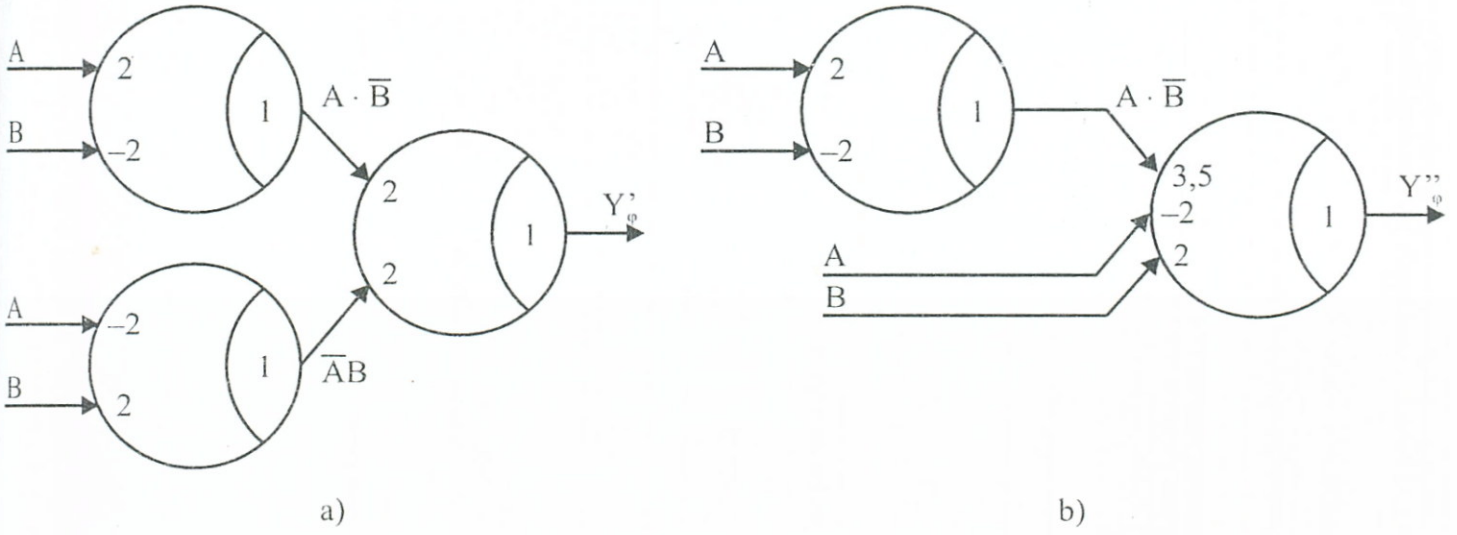
12-133. ábra *Elemi BOOLE függvények ábrázolása „n" dimenziós „kockán" és a hiper-síkok*



12-134. ábra *Lineárisan szeparálható elemi BOOLE függvények küszöbelemes realizációi*

ség. Az egyes ábrák kiértékelése után megállapítható, hogy az ANTIVALENCIA-függvénynél két hiper-síkra volt szükség, azaz összetett küszöbhálózatot igényel, míg a többiek egy-egy küszöbelemmel realizálhatók. Ez utóbbiaknál (mivel a függvények rendkívül egyszerűek) a megoldások *intuitív úton* is előállíthatók. Lehetséges megoldási eseteket foglal össze a 12–134. ábra.

Az ANTIVALENCIA realizációt két elv szerint is megkísérelhetjük, a 12–135. ábrának megfelelően. Az a) változatnál a külön-külön megvalósított $A\overline{B}$, $\overline{A}B$ összetevőket egy VAGY-elemmel kapcsoljuk össze. A b) változatnál az egyi-



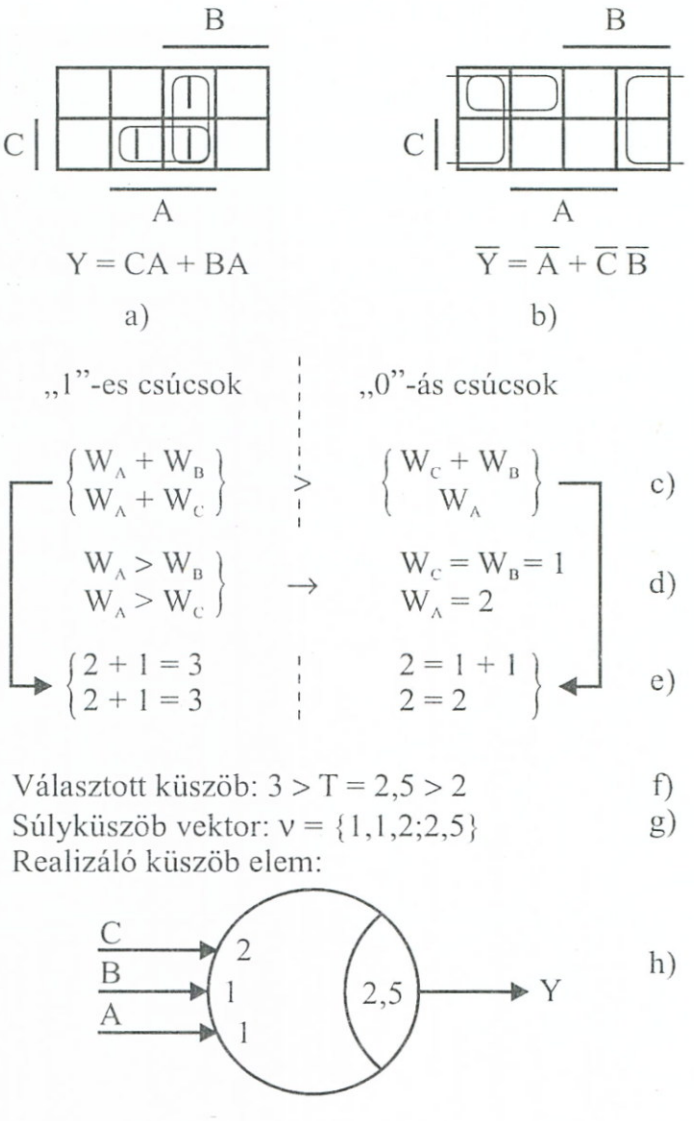
12-135. ábra *Összetett küszöbfüggvények realizációs változatai ANTIVALENCIA esetén*

ket (pl. $A\overline{B}$-et) „alárendeljük" hierarchikusan a másiknak a 9–18. ábra kaszkádosítási elvének megfelelően.

F.9.7. A küszöbhálózat előállítását a 9.2.4. pontbeli algoritmus alapján végezhetjük, és a 12–136. ábrán követhetjük.

I., II., III. lépés: Diszjunktív irredundáns alak előállítása a 12–136a. ábra minterm-táblájából adódik. Látható, hogy a függvény UNATE, így van remény az egyetlen küszöbelemmel történő realizálhatóságra, sőt nem kell az $Y \rightarrow \Phi$ konverzió sem, miután az összes változó ponált.

IV., V. lépés: Hiper-sík helyének megkeresése. Ez itt már az „n-dimenziós kockákkal" nehézkes lenne, ezért a 9–16. ábránál már alkalmazott egyenlőtlenségrendszer-megoldási



12-136. ábra *Küszöb elem identifikációja adott BOOLE függvényhez*

módszert fogjuk használni. Ez a művelet két lépésre tagolódik:

– W_i-súlytényezők megkeresése és
– T-küszöbérték beállítása.

A *súlytényezők* meghatározásához megkeressük a „legkisebb" „1"-es („befeketített") csúcsokat, valamint a „legnagyobb" „0"-ás („üres") csúcsokat. Itt a „hiper-sík" grafikus berajzolását egy egyenlőtlenség-rendszer megoldása helyettesíti. Az egyenlőtlenségek felírását megkönnyíti, ha az „1"-es csúcsokhoz a 12–136a., a „0"-ás csúcsokhoz a 12–136b. minimál alakokat felírjuk, melyekből kigyűjthető a c.) ábrabeli egyenlőtlenség-rendszer a súlytényezőkre. Mivel rendkívül egyszerű esettel van dolgunk, a d.) ábrán szereplő megoldás közvetlenül adódik. Itt lehetőleg *alacsony* súlyértékekre törekedve, és a kapott relációkat tiszteletben tartva felvettük a $W_C = W_B = 1$ és $W_A = 2$ értékeket.

Mellékesen megjegyezzük, hogy ha az egyenlőtlenség-rendszer megoldásánál ellentmondások adódtak volna, akkor ez azt jelentette volna, hogy *nem elegendő egyetlen* elválasztó hiper-sík, azaz összetett küszöbfüggvényes realizációra kellett volna áttérni.
A *küszöbérték* meghatározásához ki kell számítani a c.) ábrabeli egyenlőtlenségébe behelyettesített összegeket az e.) ábra szerint, és a T-küszöbértéket ezek közé kell beállítani az f.) ábrának megfelelően.
E feladatnál, mivel a kiinduló függvény eleve *pozitív-UNATE* volt, elmarad a $\Phi \rightarrow Y$ visszakonvertálás is, így a végleges súlyküszöb-vektor a g.) ábrán, a küszöbelem pedig a h.) ábrán felrajzolható.

F.9.8. Itt is felbontjuk (9.18) szerint a függvényt és $i = R-1$, $i = R-2$ ... esetekre felrajzoljuk a term-táblákat. Mivel általánosságban írható (9.18), majd De Morgan alapján (9–23. ábra f.):

$$F = F_1 + F_2 + \ldots + F_{R-1} + F_h$$

$$\overline{F} = \overline{F_1 + F_2 + \ldots + F_{R-1} + F_h}$$

$$\overline{F} = \overline{F}_1 \cdot \overline{F}_2 \cdot \ldots \cdot \overline{F}_{R-1} \cdot \overline{F}_h$$

a)

F_2

X_2 \ X_1	0	1	2
0	h_2	0	0
1	0	2	0
2	2	h_2	2

$\overline{F}_2$

X_2 \ X	0	1	2
0	2	2	2
1	2	0	2
2	0	0	0

(i = 2)

b)

F_1

X_2 \ X_1	0	1	2
0	h_1	0	1
1	1	h	1
2	h	h_2	h

$\overline{F}_1$

X_2 \ X_1	0	1	2
0	1	2	1
1	1	2	1
2	1	2	1

(i = 1)

c)

X_2 \ X_1	0	1	2
0	1	2	1
1	1	0	1
2	0	0	0

$\overline{F} = \text{Min}\,(F_1, F_2) = F_1 \cdot F_2$

12-137. ábra *Komplemens előállítása grafikus módszerrel*

Az egyes sejtekben szereplő konstans-értékek komplemensét pedig (9.12) alapján állítjuk elő.

A logikai szorzást a Min (*A, B*) operátor 9–22. ábra szerinti táblázata alapján végezzük el.

Az elmondottak követhetők a 12-137a., b., c. ábrákon.

F.9.9. Az eredeti A fuzzy halmaz a 9–34. ábrából:

$$A = 1/120 + 1/130 + 0{,}9/140 +$$
$$+0{,}5/150 + 0{,}2/160 + 0/170 +$$
$$+0/180 + 0/190 + 0/200 + 0/210$$

Felhasználva a (9.29) összefüggést, a komplemensre kapjuk:

$$A^c = 0/120 + 0/130 + 0{,}1/140 +$$
$$+0{,}5/150 + 0{,}8/160 + 1/170 +$$
$$+1/180 + 1/190 + 1/200 + 1/210$$

F.9.10. A (9.21), (9.22), (9.23), (9.24), (9.25) felhasználásával:

$$\text{core}(A^c) = \{170, 180, 190, 200, 210\}$$
$$\sup(A^c) = \{140, 150, 160, 170, 180, 190, 200, 210\}$$
$$\text{height}(A^c) = 1$$
$$(A^c)_{\alpha=0{,}6} = \{160, 170, 180, 190, 200, 210\}$$
$$|A^c| = 0{,}1 + 0{,}5 + 0{,}8 + 1 + 1 + 1 + 1 + 1 = 6{,}4$$

F.9.11. A feladat teljesen hasonlít a 9.18 Példához, így a műveleteket hasonlóan kell elvégezni.

F.9.12. Az elkészítendő példához forrásul szolgálhatnak a 9.5.2.1. pont h2.) alpontjában elmondottak.

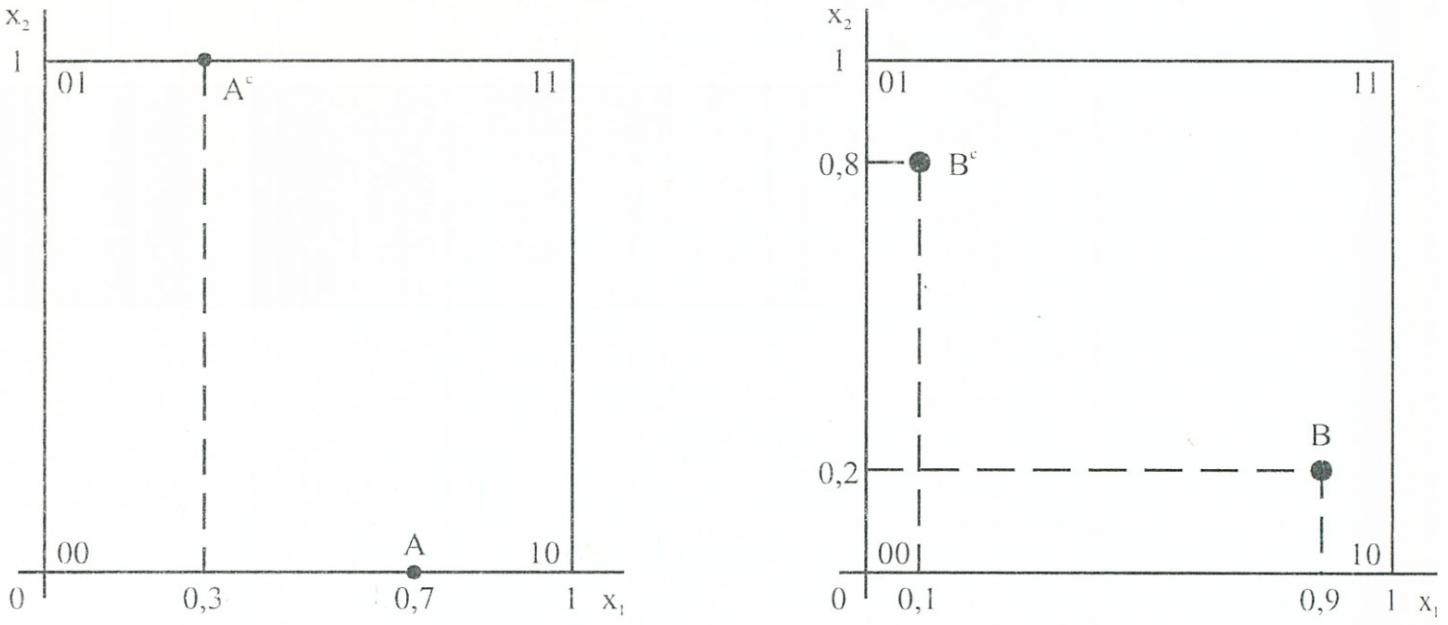
F.9.13. A (9.41) összefüggés értelmében felírhatók:

$$(R^{-1})^{-1} = R$$
$$\text{dom}R(X, Y) = \text{ran}R^{-1}(X, Y)$$
$$\text{dom}R^{-1}(X, Y) = \text{ran}R^{-1}(X, Y)$$

F.9.14. Az *a.) esetnél* látható, hogy: $\mu_A(x_2) = 0$ tehát az illesztő vektor így is írható:

$A(0{,}7(x_1),\ 0(x_2))$, továbbá:
$A^c(0{,}3(x_1), 1(x_2))$

Az A, A^c halmazokat a 11–138a. ábra mutatja:



12-138. ábra *A G.9.14. feladat halmazainak grafikus ábrázolása egységkockán*

$B(0{,}9(x_1),\ 0{,}2(x_2))$, továbbá:
$B^c(0{,}1(x_1), 0{,}8(x_2))$

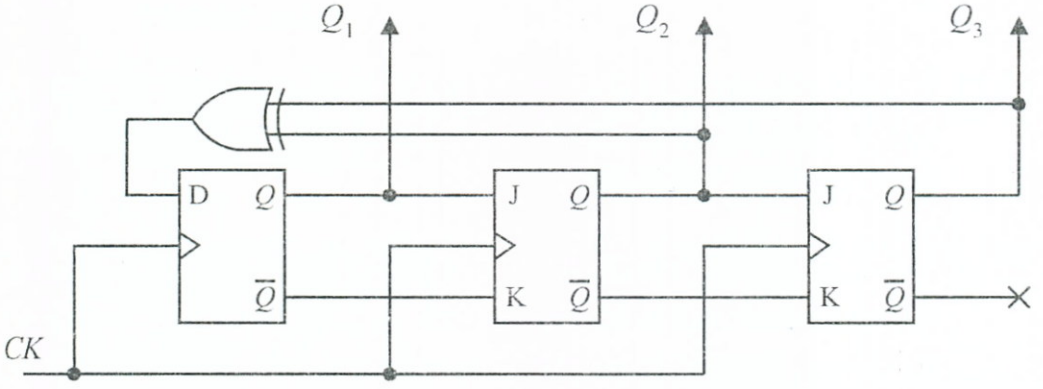
Az B, B^c halmazokat a 11–138b. ábra mutatja

12.10.1. Gyakorló feladatok a 10. „Funkcionális egységek" c. fejezethez

*G.10.1. Rajzoljunk fel egy *S–P* regiszter kapcsolást.

*G.10.2. Rajzoljunk fel egy *P–S* regiszter kapcsolást.

*G.10.3. Vizsgáljuk meg a 12–139. ábrán látható kapcsolás működését, és azt is, hogy milyen célra lenne hasznosítható.



12–139. ábra *Kiinduló hálózat a G.10.3. példához*

G.10.4. Gondoljuk át, miként módosul (10.2), ha nem azonosak a valószínűségek.

*G.10.5. Alakítsunk át egy GRAY-kódszót BIN kódrendszerbeli, soros ábrázolással.

G.10.6. Gondoljuk át, hogy miként alkalmas egy többdimenziós, paritás-elemes kódrendszer hibajavításra.

*G.10.7. Alakítsunk ki soros paritásképző áramkört n-bitre.

*G.10.8. Alakítsunk ki egy 1/3/8 DMX-et áramlogikás MOS-kapcsolók felhasználásával.

*G.10.9. Realizáljuk az alább megadott logikai függvényt:

a) 16/4/1 MPX
b) 8/3/1 MPX

típusú modul-elemek felhasználásával

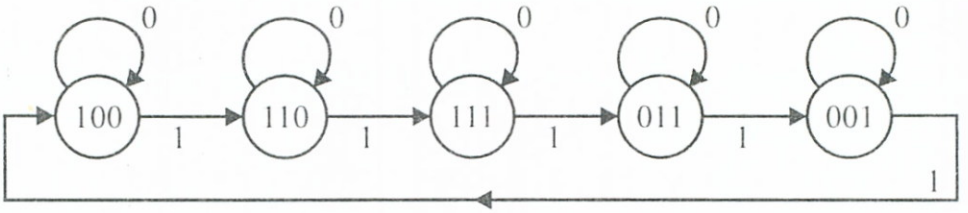
$$Y = \prod^{4}(0,3,5,9,11,12)$$

*G.10.10. Gondoljuk át, hogy miként oldható meg DMX és MPX segítségével 8 mérési állomás jeleinek átvitele nagyobb távolságra egyetlen soros átviteli vonal birtokában.

*G.10.11. Tervezzünk kétirányú 3-bites S–S típusú léptető-regisztert. A kapcsolást lássuk el alaphelyzet-beállító CL-bemenettel is.

*G.10.12. Vizsgáljuk meg a 10–29a. ábrán látható szinkron számláncnál, hogy a bekeretezett hálózatrész valóban biztosítja-e helyes, *bináris* működést.

*G.10.13. Tervezzünk egy módosított JOHNSON-számlálót a 12–140. ábra szerinti működési ciklussal. A hálózatot úgy alakítsuk ki, hogy bekapcsolás után maximum két fázist követően már „beletaláljon" az üzemszerű működési ciklusba.



12-140. ábra *Módosított JOHNSON számláló gráfja*

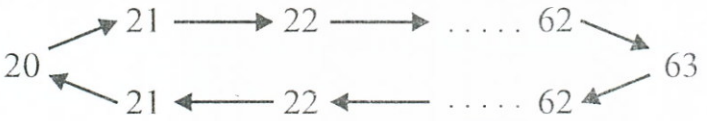
*G.10.14. Alakítsunk ki egy 1/31-es *frekvenciaosztót* számlálók felhasználásával.

*G.10.15. Alakítsunk ki az alábbi ciklus realizálására egy számlálót a 10–32. ábrán bemutatott modul felhasználásával:

$$6 \rightarrow 5 \rightarrow 4 \rightarrow 0 \rightarrow 1 \rightarrow 2$$

*G.10.16. Adva a 12–141. ábrán látható reverzíbilis számláló-hálózat. Vizsgáljuk meg működését és bővítési adottságait.

*G.10.17. Alakítsunk ki a 12–141. ábrán látható számláló kapcsolás felhasználásával egy alábbi reverzáló ciklusban működő hálózatot:



G.10.18. Végezzük el két felvett szám összeszorzását a tanult *duplázó–felező* módszerrel.

G.10.19. Végezzük el két felvett bináris szám összeszorzását a Booth-algoritmus szerint.

*G.10.20. Vizsgáljuk meg, miként lehetne *négyzetre emelő* áramkört kialakítani.

*G.10.21. Gondoljuk át, miként lehetne egyszerű *hatványozó* áramkört előállítani.

G.10.22. Végezzük el két felvett szám osztását *kivonásra* történő visszavezetéssel.

*G.10.23. Tegyünk kísérletet a megismert *törthányadosú* osztóberendezés hasznosítására egészhányadosú osztásnál.

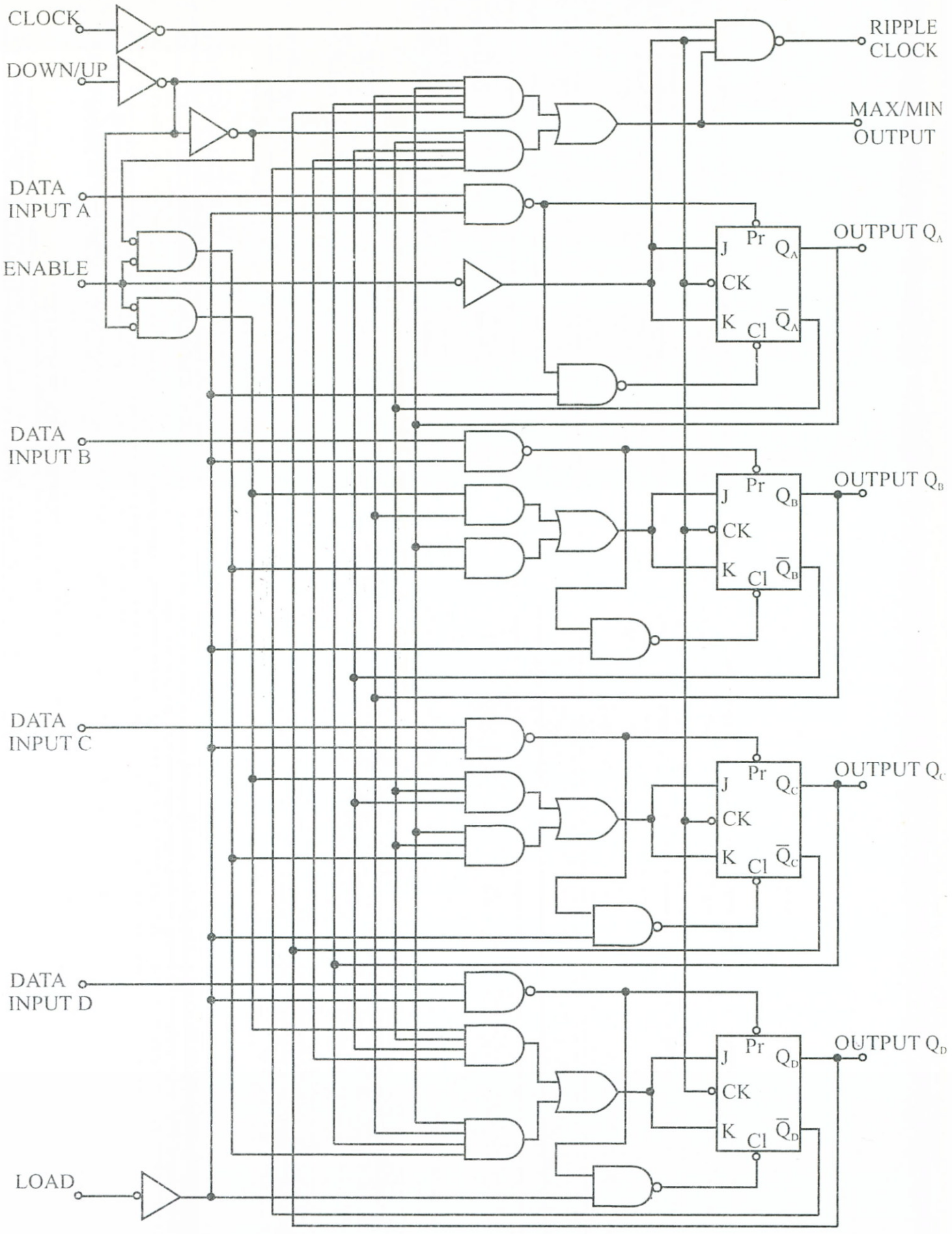
*G.10.24. Vizsgáljuk meg, hogy milyen belső felépítéssel realizálható a 10.84. ábra soros komparátorának KIÉRTÉKELŐ HÁLÓZAT-a.

*G.10.25. Tervezzünk RAM felhasználásával egy sorrendi hálózatot, mely egy X bemenettel és egy Z kimenettel rendelkezik és X minden negyedik állapotváltozásakor történik Z-nél állapotváltozás.

*G.10.26. Vizsgáljuk meg egy CAM–RAM összetevőkből felépülő *gyorsmemória* címzési viszonyait olyan esetben, amikor adott „D" esetén több egyezési kimenet is adódik és ki kell alakítanunk ezekkel egy címzési stratégiát.

*G.10.27. Alakítsunk ki 64 x 1 bit kapacitású, bit-szervezésű RAM-memóriát a 10.93. ábrán bemutatott 4 x 4 = 16 x 1 bit kapacitású modulból, bővítéssel.

*G.10.28. Tekintsük most *kiinduló modul*nak az előző példa 12-165. ábrabeli eredményeként kapott hálózatot. Vizsgáljuk meg,



12 - 141. ábra. *Binér - reverzibilis számlánc a G.10.16 példához*

miként állítható elő ilyen modulokból egy 64 x 8 bites szószervezésű memória.

*G.10.29. Alakítsunk ki a 12-167a. ábrán látható szószervezésű RAM-memória modul felhasználásával egy 12 kByte-os RAM-memóriát.

*G.10.30. Kialakítandó egy digitális berendezés, mely többek között egy 8 bites szóhosszúságú ROM-memóriát, egy ugyancsak 8 bites SZ számlálót és egy ÖH összehasonlító áramkört tartalmaz. Az SZ számláló egy START jelre indul és mindaddig lépked, amíg a kimenetén jelentkező érték meg nem egyezik a ROM kimeneti Byte-jával. Egyezés esetén, az ÖH összehasonlító E kimenetén található lámpa kigyullad és a számláló leáll. A léptetés ideje alatt az összehasonlító K kimenetén található lámpa ad jelet minden léptetéskor, egészen az E lámpa kigyulladásáig. Ha valamilyen okból a számláló nem állt meg a kívánt helyen és túllép, akkor az összehasonlító N kimenetén levő lámpa ad hibajelet. A 0-tól 255-ös nagyság szerint tárolt ROM-szavakat, melyek a számláló lépésszámának felelnek meg, a ROM címző bemenetén keresztül lehet előválasztani. A realizációnál tételezzük fel, hogy a kiinduló MSI-modulkészlet minden eleme 4 bites adatcsatlakozásokkal rendelkezik és a ROM címbemeneteinek száma: 8.

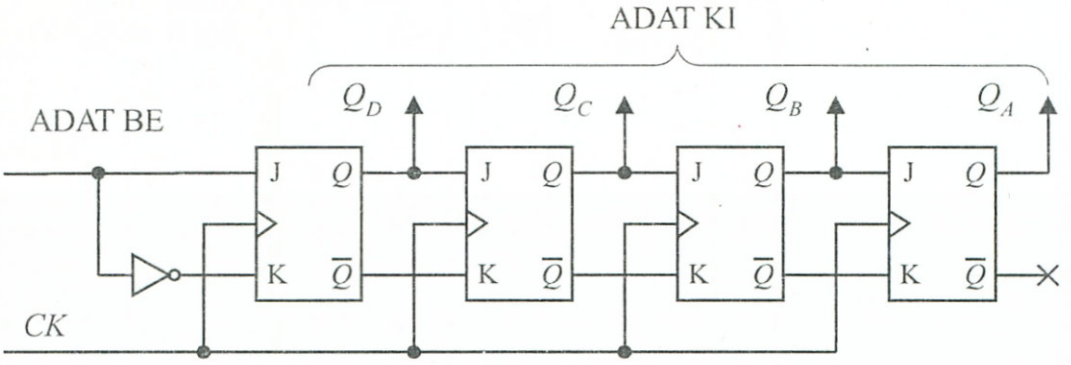
*G.10.31. Mi történik egy *D/A* átalakítónál, ha az *Uref* referenciafeszültség széles tartományban változtatható?

*G.10.32. Milyenek az analóg tartó áramkörök?

12.10.2. Feladatmegoldásoka 10. fejezethez

F.10.1. *S–P* regisztert alakíthatunk ki a 10–3. ábrán felrajzolt *S–S* változatból, ha valamennyi tárolócella Q_{ic} kimeneti pontját is kivezetjük a 12–142. ábra szerinti módon.

F.10.2. *P–S* regisztert alakíthatunk ki ugyancsak a 10–3. ábrabeli *S–S* változatból, ha az egyes flip-flopokat ellátjuk az aszinkron párhuzamos beírást lehetővé tevő *PR* (Preset) és *CL*



12–142. ábra *S-P tároló J-K flip-flopokkal*

(Clear) bemenetekkel, és ezeket használjuk fel párhuzamos adatbemenetekként.

F.10.3. A kiinduló 12–139. ábrán látható hálózat egy visszacsatolt *D*-tárolós léptetőregiszterből és antivalencia kapuból tevődik össze. Az ábra alapján felírhatók a tárolóelemekre jellemző működési függvények, ill. ezek alapján a kódolt rovatos állapottábla (12–143a.). Ha végül az állapotgráfot is

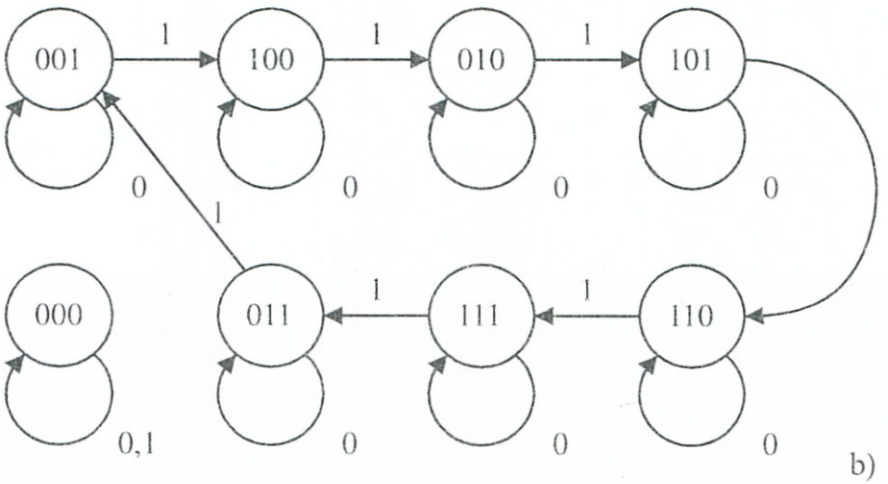
$$Q_{1(T+1)} = Q_{2T} \oplus Q_{3T} = D_{1T}$$

$$Q_{2(T+1)} = Q_{1T} \quad = D_{2T}$$

$$Q_{3(T+1)} = Q_{2T} \quad = D_{3T}$$

Q_{1T}	Q_{2T}	Q_{3T}	$Q_{1(T+1)}$	$Q_{2(T+1)}$	$Q_{3(T+1)}$
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	1	1
0	0	0	0	0	0

a)

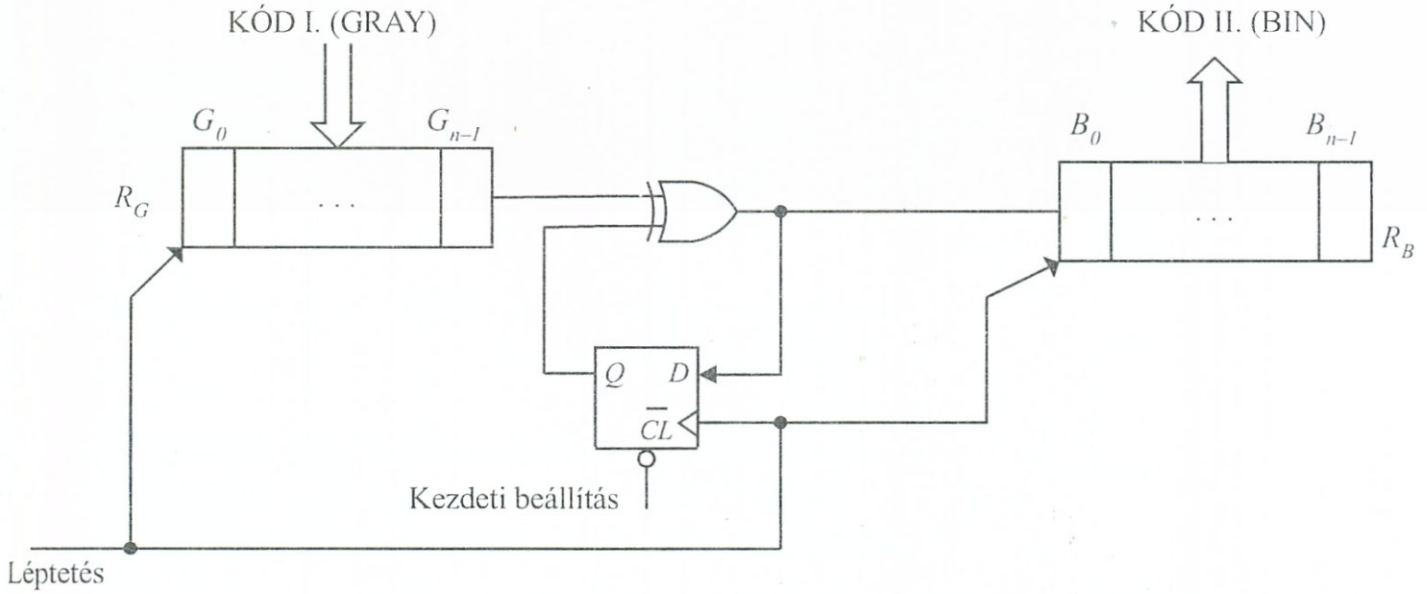


b)

12–143. ábra *Visszacsatolt léptető-regiszter, mint karaktergenerátor*

felrajzoljuk a 12–143b. ábra szerint, ebből megállapítható, hogy egy *M = 7-es modulusú számlálóhoz* jutottunk, melynél a $Q_1Q_2Q_3$-000 állapot *izolált,* azaz amennyiben az áramkör valami módon (pl. áramellátás bekapcsolásakor) ide bekerül, akkor működésképtelenné válik. A hálózat felhasználható például a táblázat kódszavait adott sorrendben kibocsátó, ún. *„karakter generátor"*-ként.

F.10.5. A megoldást a 10–9. ábra összefüggései alapján rajzoltuk fel a 12–144. ábrán.



12–144. ábra *SOROS - GRAY/BIN kódátalakító*

F.10.7. Soros paritásképzésnél a (10.13) összefüggésekből indulunk ki, azzal a meggondolással, hogy az $A_0 \rightarrow A_1$ bitek a soros esetben egymáshoz képest időeltolással jelentkeznek, a 10–15. ábra elvi rajzának megfelelően. Ha ezt a (10.13) összefüggésre alkalmazzuk, akkor felírhatók az alábbiak:

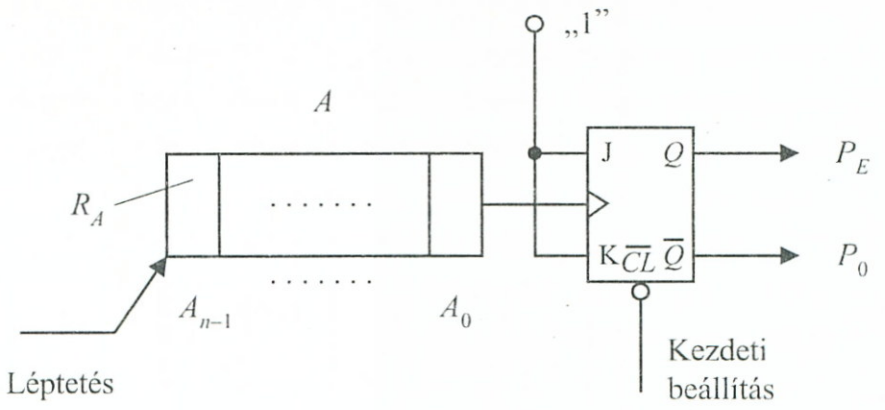
$$P_{E(T+1)} = A_T \oplus P_{ET}$$

$$P_{O(T+1)} = A_T \equiv P_{OT}$$

Ha az összefüggések közül például a PÁROS esetet kiragadjuk, ezt formálisan azonosnak találjuk a T-tárolóelemre a 2. fejezet 2–25. ábráján bevezetett működési függvénnyel:

$$Q_{T+1} = T_T \oplus Q_T$$

$$P_{E(T+1)} = A_T \oplus P_{ET}$$



12–145. ábra *SOROS n-bites paritásképző*

A felismerés alapján $P_E = Q \cdot T = A$ helyettesítéssel a 12–141. ábra szerinti kapcsolás rajzolható fel megoldásként, melynél a

$$\overline{P}_E = P_O$$

összefüggést is figyelembe vettük, és a *T*-tárolóelemet *J–K* tárolóelemből állítottuk elő.

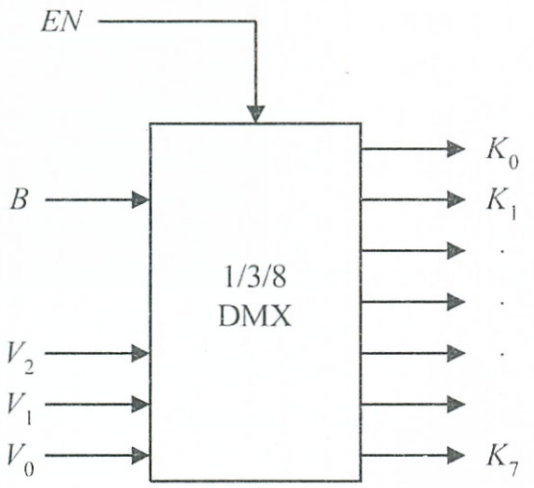
F.10.8. Az 1/3/8 DMX elvi vázlata és működési vázlata a 12–146a. és b. ábrán látható. A táblázat egy-egy sora egy-egy ÉS-kapcsolatnak felel meg az alábbi formula szerint:

$$K_i = B \cdot E_i^3(V_0, V_1, V_2)$$

Az ÉS-kapcsolatot MOS-áramlogikában sorbakapcsolt MOS tranzisztorokkal valósíthatjuk meg. Az ily módon felépített, ún. FA-struktúrájú megoldást a 12–146c. ábrán rajzoltuk fel.

F.10.9. Az MPX-szel történő függvényrealizálással már foglalkoztunk a 10.4.2. pontban. Jelenlegi példánkban ugyanannak a függvénynek a realizálását 3-féle változatban is bemutatjuk. A kiinduló-függvényt első lépésben diszjunktív alakra kell hoznunk, a (10.14) összefüggés felírhatósága érdekében. Az átalakítást V–K táblával elvégezve, a 12–147a. ábrán megkaptuk a diszjunktív szabályos alakot.

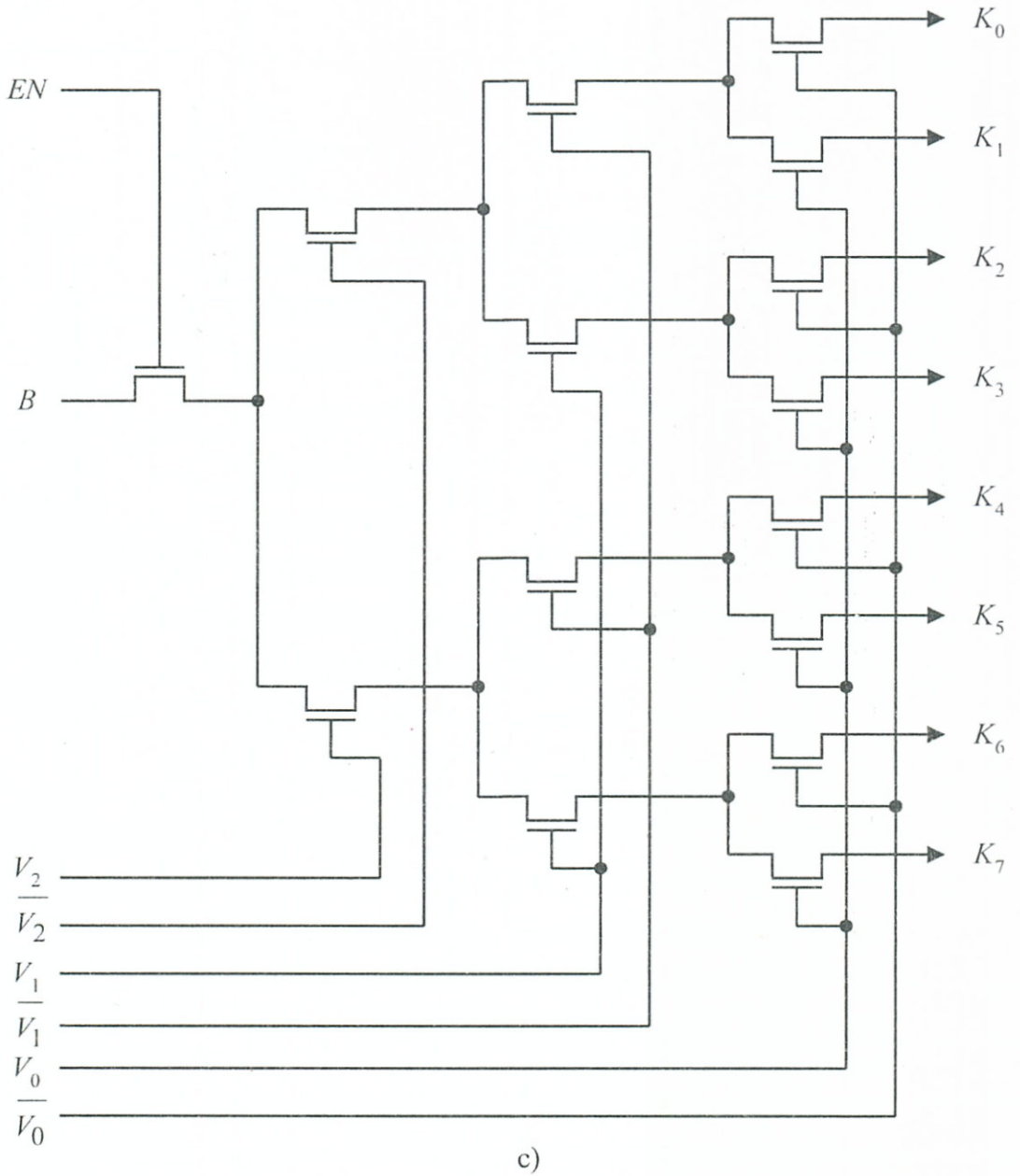
a) változat: Az n=4 változós függvényt egy 16/4/1-es MPX-szel közvetlenül realizálhatjuk. A 10–24. ábrán bemutatott megoldáshoz hasonlóan eljárva, a kiinduló függvényünk-



a)

V_2	V_1	V_0	B	EN	$K_i = 1$
0	0	0			K_0
0	0	1			K_1
0	1	0			K_2
0	1	1	1	1	K_3
1	0	0			K_4
1	0	1			K_5
1	1	0			K_6
1	1	1			K_7

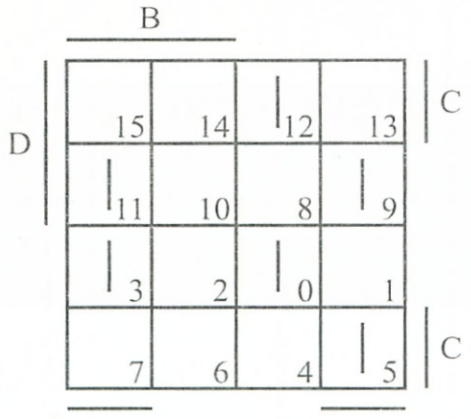
b)



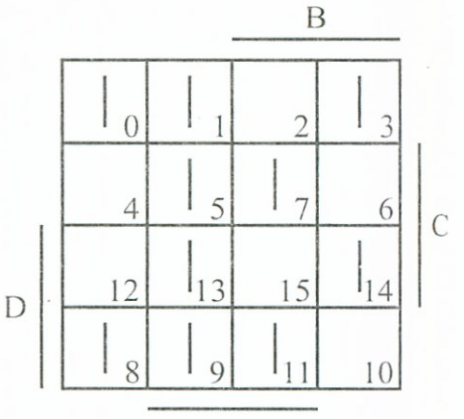
c)

12–146. ábra *1/3/8 DMX MOS-FA struktúra*

a)



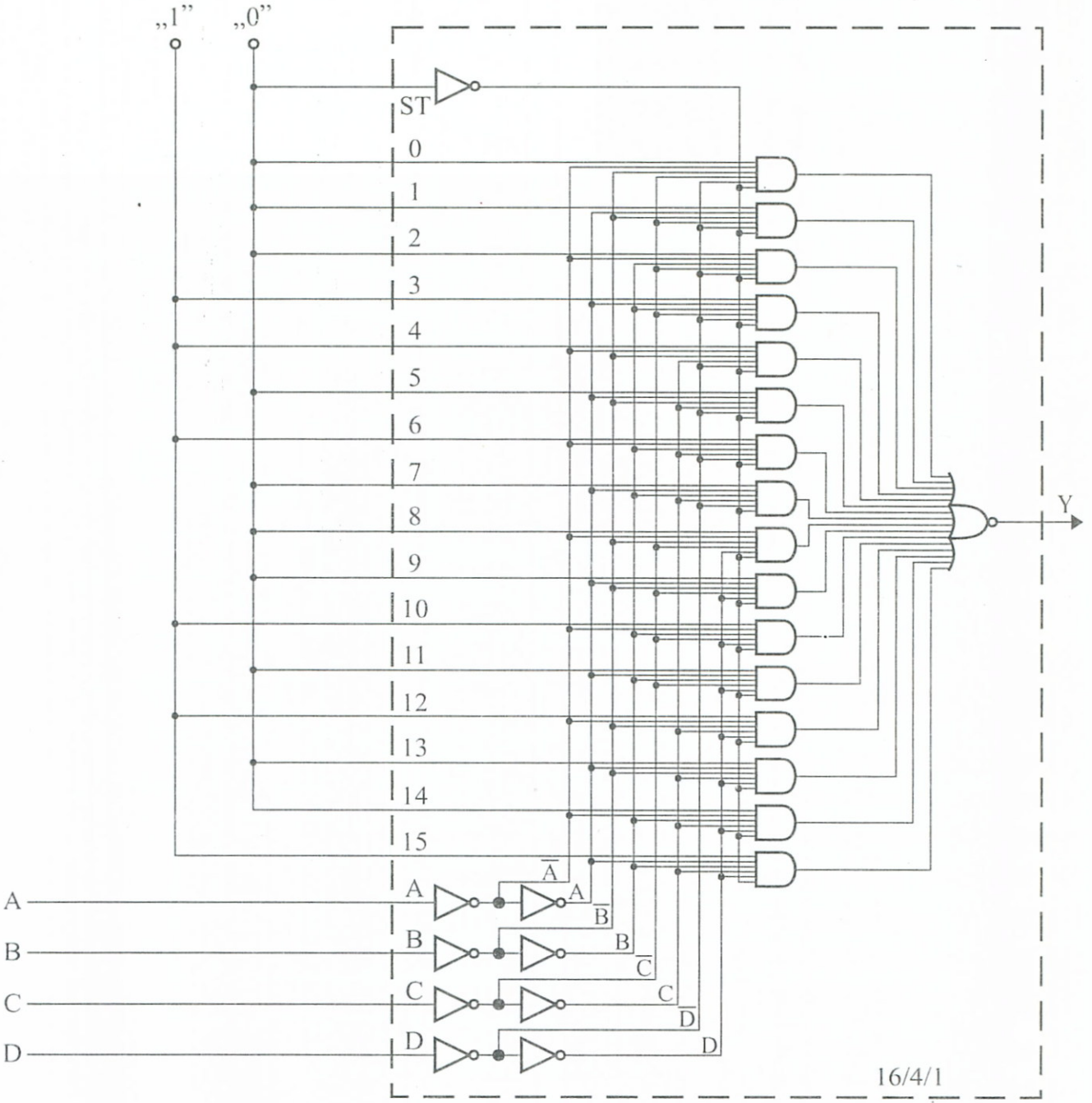
$Y = \overset{4}{\pi}(0, 3, 5, 9, 11, 12)$



$Y = \overset{4}{\Sigma}(0, 1, 2, 5, 7, 8, 9, 11, 13, 14)$

$\overline{Y} = \overset{4}{\Sigma}(3, 4, 6, 10, 12, 15)$

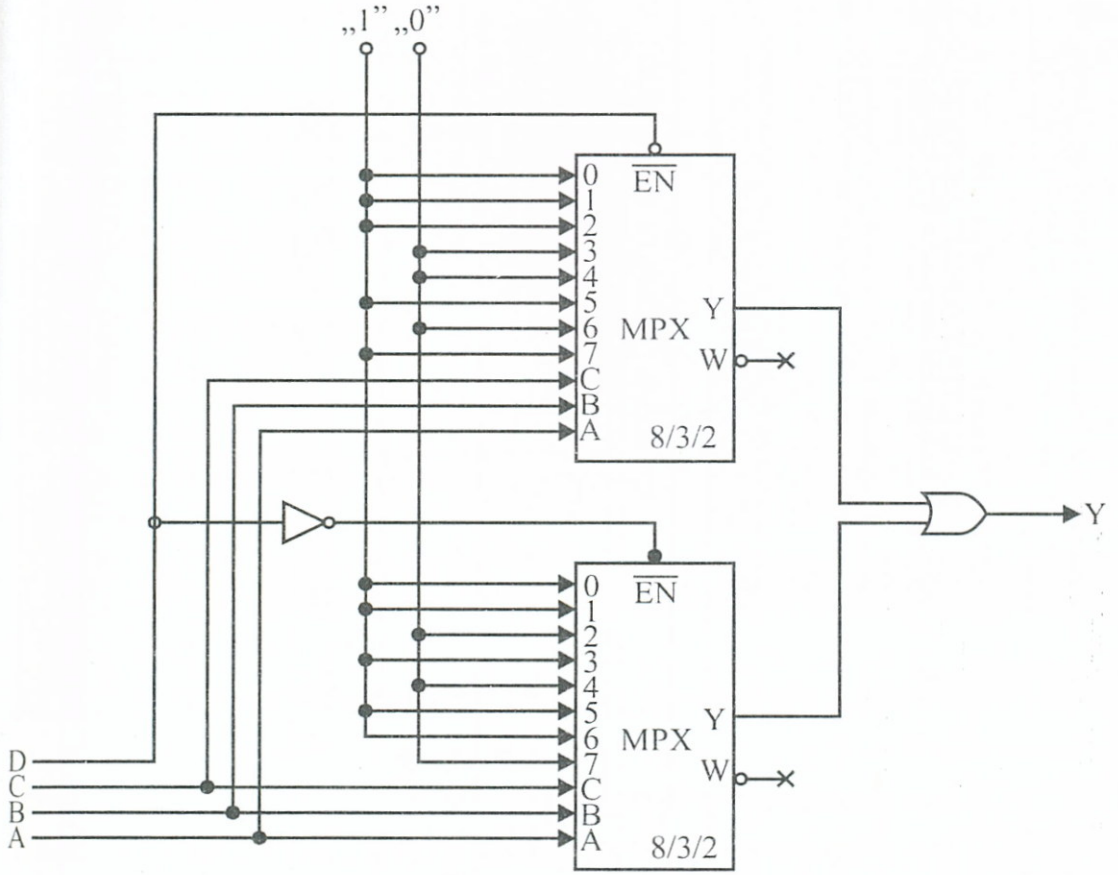
b)



12-147. ábra *Függvényrealizálás 16/4/1 MPX-szel*

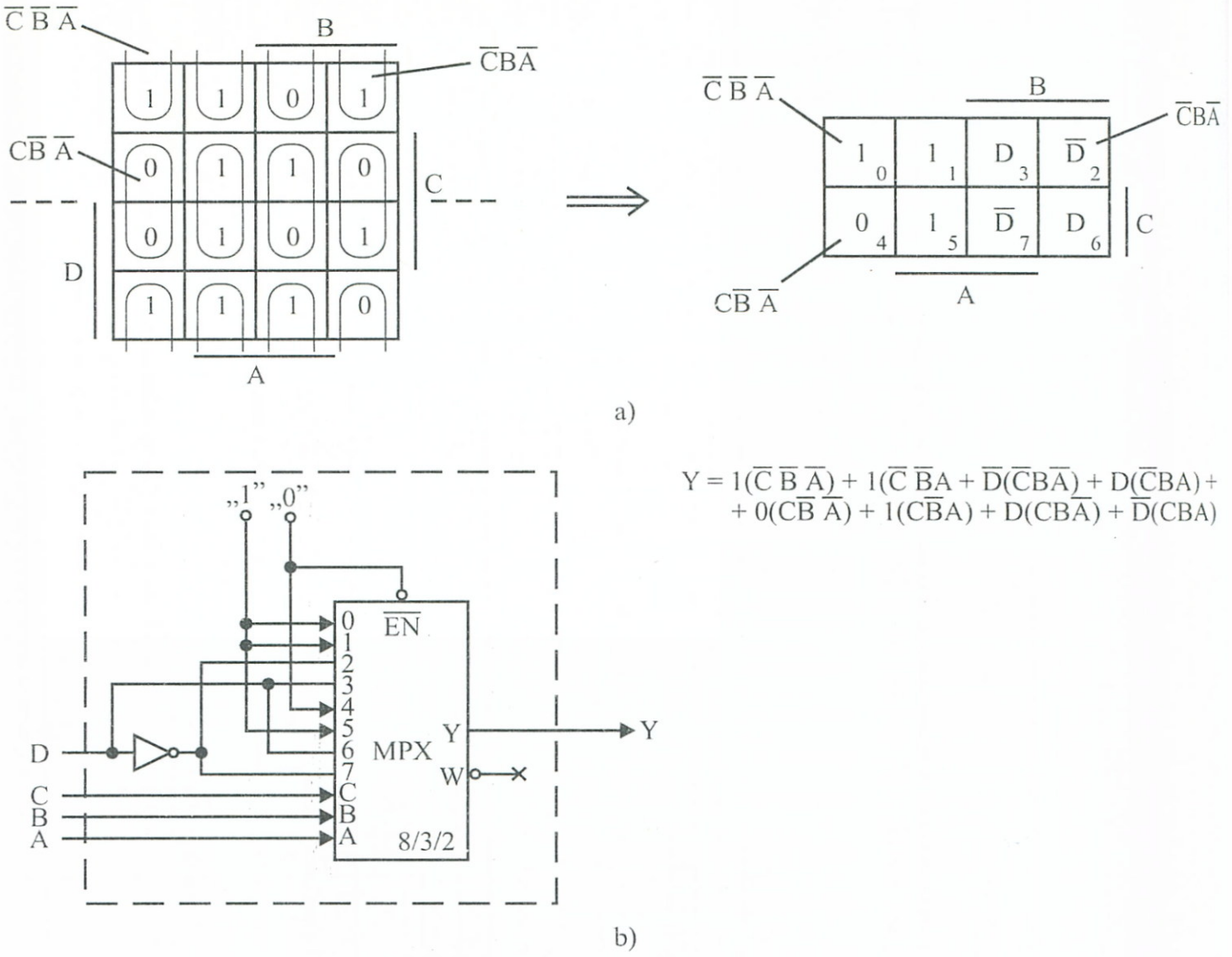
nek megfelelő eredményt a 12–147b. ábrán rajzoltuk fel. Mivel az áramkör a kimeneten negált (aktív „0") eredményt szolgáltat, célszerű volt a függvény negáltjának diszjunktív alakjából kiindulni.

b) változat: A 8/3/1 típusnál, mikor az MPX „EN" bemenettel is rendelkezik, alkalmazhatunk bővítést, melyhez a 12–148. ábrán látható 8/3/2-es modul típust használjuk fel. Itt nem szükséges a függvény negáltját realizálnunk, miután a modul mind ponált, mind negált kivezetéssel rendelkezik.



12-148. ábra *Függvényrealizálás MPX bővítéssel*

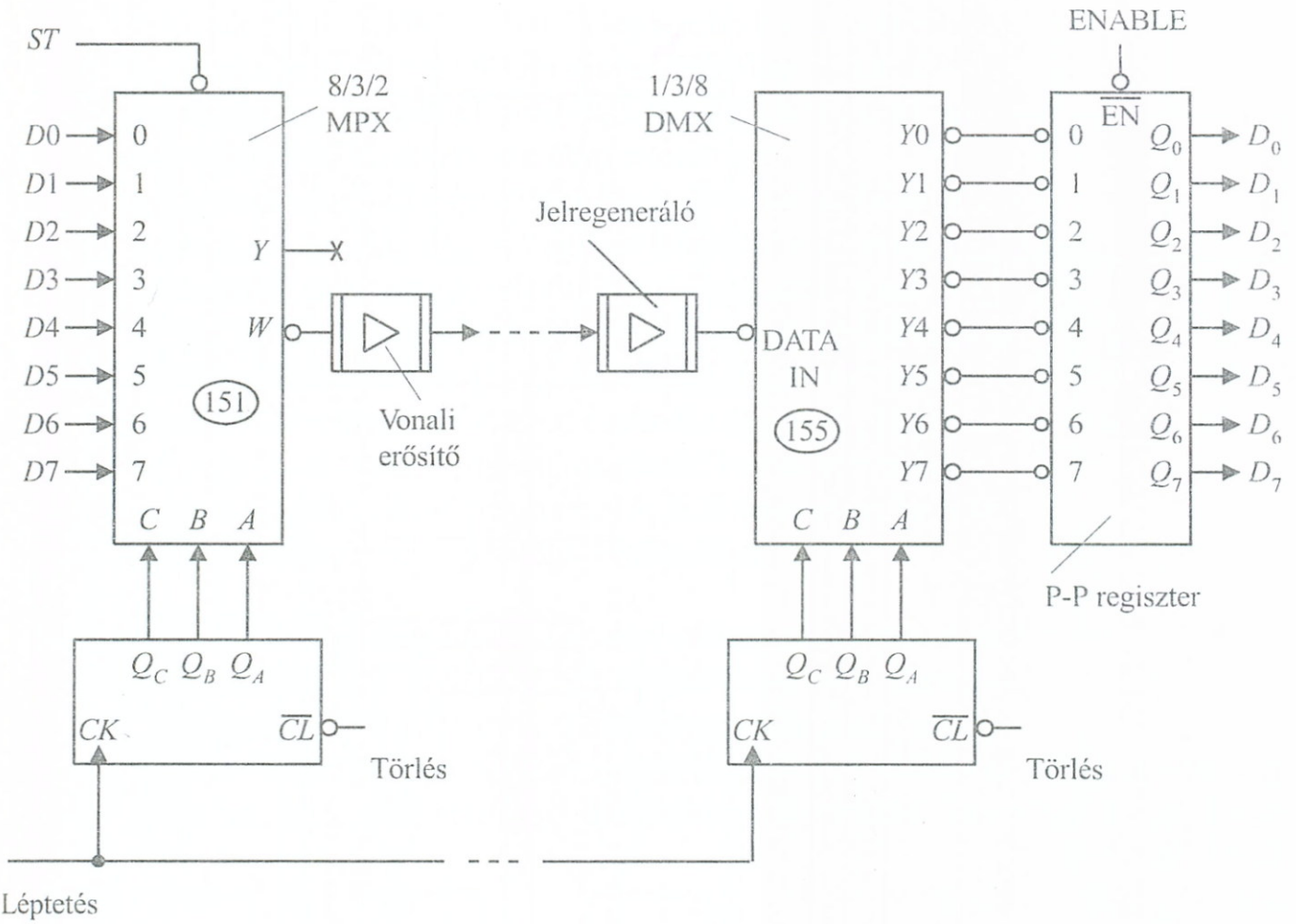
c) változat: A feladat megoldható egyetlen 8/3/1 típusú MPX-szel is, ez esetben az engedélyező EN bemenet nem kell, hogy aktív szerepet kapjon. Ha megvizsgáljuk a 12–149. ábra minterm-tábláját, belátható, hogy a 12–149a. ábra értelmében a 4-változós táblát átrajzolhatjuk egy 3-változóssá, mely alapul szolgál az egyetlen 8/3/1 MX-szel történő realizációhoz. Ismét a 12–148. ábra áramkörét választva a realizációhoz a 12–149b. ábrán rajzoltuk fel a megoldást.



$$Y = 1(\overline{C}\,\overline{B}\,\overline{A}) + 1(\overline{C}\,\overline{B}A + \overline{D}(\overline{C}B\overline{A}) + D(\overline{C}BA) + \\ + 0(C\overline{B}\,\overline{A}) + 1(C\overline{B}A) + D(CB\overline{A}) + \overline{D}(CBA)$$

12-149. ábra *Négyváltozós függvény realizálása 8/3/2 MX-szel*

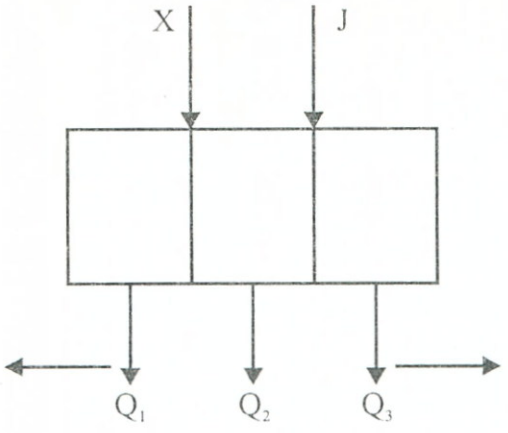
F.10.10. A feladat egy adatátviteli problémát vet fel, melynél csak egyetlen adat-vonallal rendelkezünk. Az elvi sémára a 12–150. ábrán rajzoltuk fel, hogy 8/3/2 típusú MPX és egy DMX 1/3/8-ra bővített változatának felhasználásával. Az átvitel úgy történik, hogy az MPX-nél a D_0–D_7 adatbemeneteket a választó bemenetekre csatlakozó (pl. BINÉR ELŐRE) számlánc léptetésével egyenként (időben eltolva) juttatjuk a W-kimenetre, ahonnan az információ az átviteli adatvonalon *sorosan* továbbítódik a DMX DATA IN bemenetére. A DMX Yi adat-kimeneteit is egy előbbivel megegyező, azzal szinkronban lépkedő számlánc léptetései választják ki. Amennyiben a vevő oldalon az Y_0–Y_7 kimeneteket egyidejűleg, *párhuzamosan* akarjuk felhasználni, akkor a DMX után még egy P–P típusú regisztert kell csatlakoztatnunk,



12–150. ábra *8 független mérési adat távolsági átvitele egy soros adatvonalon*

melyet a soros átviteli ciklus befejeződése után olvasunk ki az ENABLE kapuzó jel megfelelő időzítésével.
Amennyiben az ADÓ–VEVŐ távolság nagyobb, szükség esetén az MPX-oldalon *erősítőt,* a DMX oldalon *jelregenerálót* is be kell iktatni.

F.10.11. A kétirányú soros léptetőregiszter tervezését a szinkron sorrendi hálózatok tervezésénél megismert módszerek alapján végezhetjük el (8. fejezet). A hálózat elvi rajzát a 12–151a. ábrán tanulmányozhatjuk, itt „*X*" az adatbemenet, „*J*" a léptetés irányának kijelölése. *S–S* üzem esetén a soros kilépőjelek *jobbra lépésnél* a Q_3 kimeneten, *balra lépésnél* a Q_1 kimeneten jelennek meg a léptetés sorrendjében. Amennyiben a Q_2 cellát is kivezetjük, az áramkör *S–P* üzemben is használható lesz. Az ily módon körülírt hálózatnak megfelelő kódolt rovatos állapottáblát a 12–151b. ábrán töltöttük ki. Amennyiben *D*-tárolós realizációt választunk, a



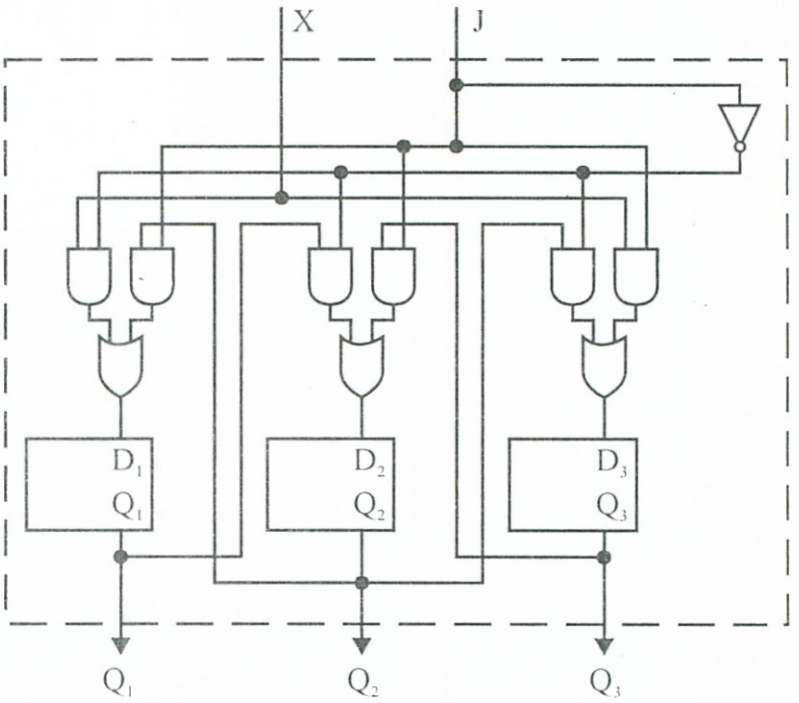
a)

t			J = 0						J = 1					
Q_1	Q_2	Q_3	x = 0			x = 1			x = 0			x = 1		
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	0	1	0	0	0	1	0	0	1	1
0	1	0	0	0	0	1	0	1	1	0	0	1	0	1
0	1	1	0	0	1	1	0	1	1	1	0	1	1	1
1	0	0	0	1	0	1	1	0	0	0	0	0	0	1
1	0	1	0	1	0	1	1	0	0	1	0	0	1	1
1	1	0	0	1	1	1	1	1	1	0	0	1	0	1
1	1	1	0	1	1	1	1	1	1	1	0	1	1	1

b)

$$D_1 = \bar{J}X + J \cdot Q_2$$
$$D_2 = \bar{J} \cdot Q_1 + J \cdot Q_3$$
$$D_3 = \bar{J} \cdot Q_2 + J \cdot X$$

c)



d)

12-151. ábra *Kétirányú léptetőregiszter kialakítása*

$$Q_{i(t+1)} = D_{it}$$

összefüggés értelmében felírható a tárolók működtetési függvénye (12–151c. ábra), a realizációs hálózat pedig a 12–151d. ábra szerinti lesz.

F.10.12. A 10–29. ábrabeli számláncnál az előreszámlálást biztosító hálózatrész működését a 12–152. ábra táblázatán ellenőrizhetjük. A számlánc Q_A, Q_B, Q_C, Q_D *J–K* típusú tárolóelemei számlálási üzemmódjukban *T*-tárolóelemként üzemelnek (*Ji* = *Ki*). A számlálási léptetést az EP.ET = ENABLE kapuzó-jel engedélyezi.

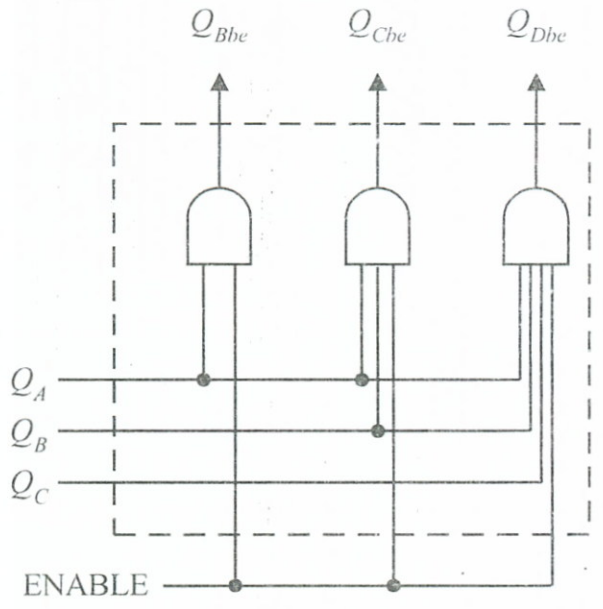
A Q_A tárolóelem képviseli a 2^n legkisebb binér helyértéket, így ennek minden órajel-léptetés esetén állapotot kell váltania függetlenül a többi tárolótól. Ezért a Q_A bemeneti vezérlése a jelölt hálózatrészben nem is szerepel.
A Q_B tárolóelem állapotváltásai (az a. ábra táblázatából ellenőrizhetően) mindig a $Q_A = 1$ *ütemet követően* kell, hogy megtörténjenek, ezért a Q_{Bhe} jelet a $Q_A = 1$ fogja képviselni. A Q_C, ill. Q_D tárolóelemek állapotváltozásai (ugyancsak a táblázatból ellenőrizhetően) a Q_AQ_B–11, ill. $Q_AQ_BQ_C$–111 *ütemeket követően* következnek be, ezért ezeknél a $Q_{Cbe} = Q_A \cdot Q_B$, $Q_{Dbe} = Q_A \cdot Q_B \cdot Q_C$ feltételek adják a megfelelő bemeneti vezérlést.
A kapu áramkörökbe még az ENABLE feltételt is bevezették az előzőkben elmondottak értelmében.

F.10.13. A kiindulásként megadott működési ciklusra, ha közvetlenül megterveznénk a hálózatot, ez a *ciklusban nem szereplő:* $Q_1Q_2Q_3$:000, 010, 101 elvileg lehetséges állapotok valamelyikébe bekerülve bizonytalan működésű lehetne. Ezért célszerű az ún. „self starting" kialakításra törekedni, mely

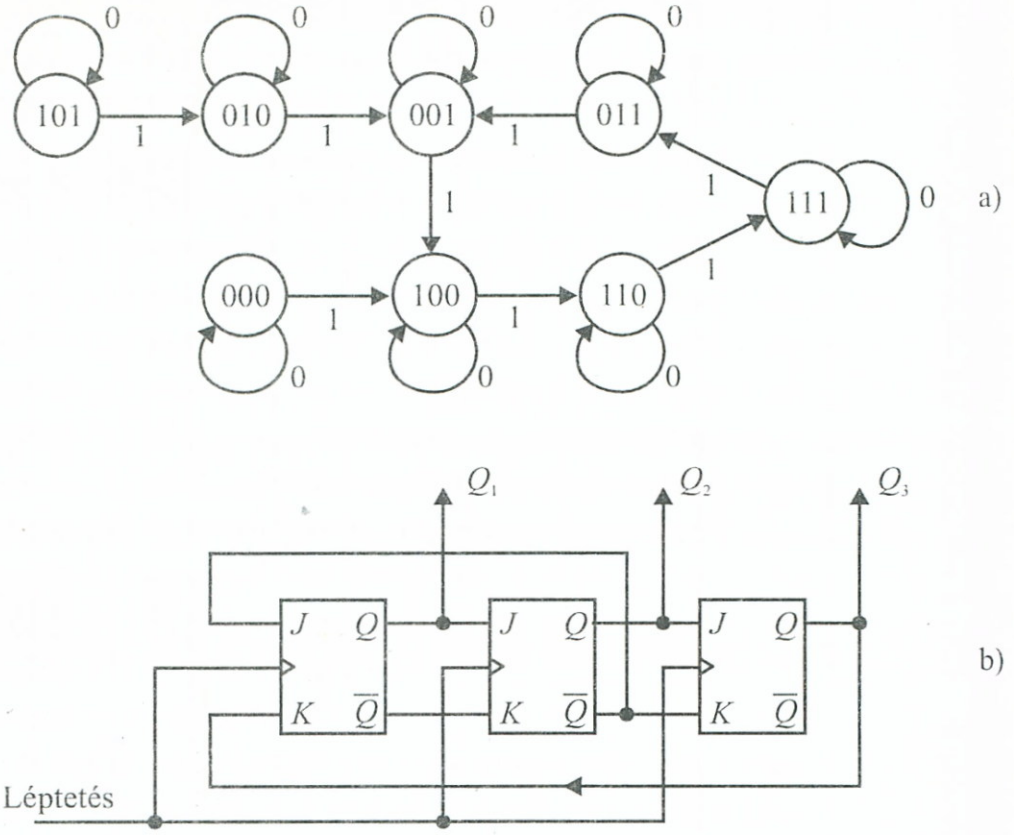
a)

Q_D	Q_C	Q_B	Q_A	
0	0	0	0	
0	0	0	1	← Q_A
0	0	1	0	
0	0	1	1	← Q_AQ_B
0	1	0	0	
0	1	0	1	← Q_A
0	1	1	0	
0	1	1	1	← $Q_AQ_BQ_C$
1	0	0	0	
1	0	0	1	← Q_A
1	0	1	0	
1	0	1	1	← Q_AQ_B
1	1	0	0	
1	1	0	1	← Q_A
1	1	1	0	
1	1	1	1	← $Q_AQ_BQ_C$

b)



12–152. ábra *A 10-29. ábrabeli számláló léptetés vezérlése*

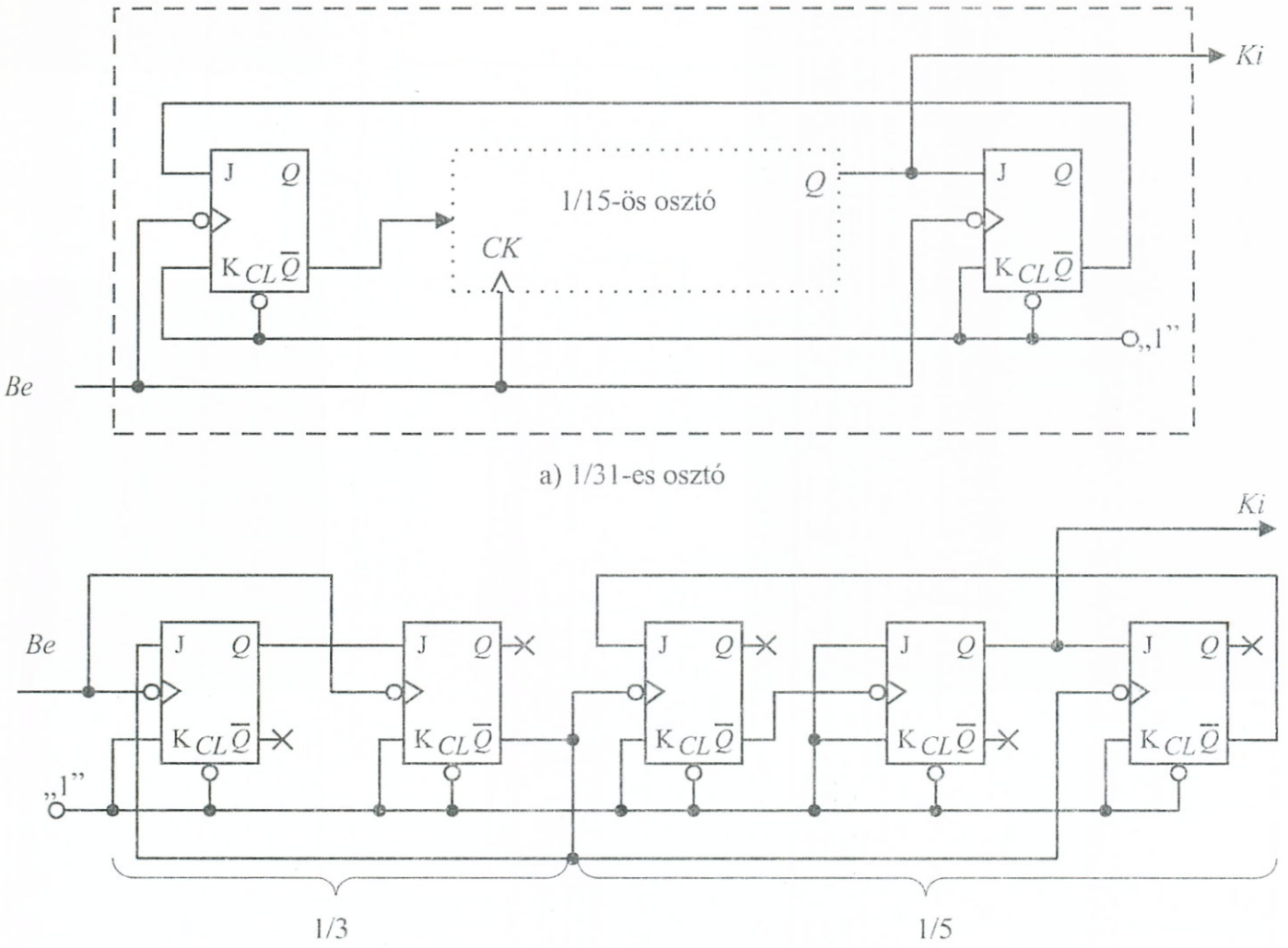


12-153. ábra *M = 5-ös modulusú JOHNSON számláló „self starting" kialakítással*

olyan megoldást jelent, mely (pl. bekapcsoláskor) bármelyik állapotból indulva biztosan beletalál a kívánt üzemszerű ciklusba. Esetünkben a kódolt állapotgráfot, ha a 12–153a. ábra szerint vesszük fel, akkor teljesülhet a fenti kívánalom. A megoldáshoz J–K tárolóelemeket választva és a tervezés lépéseit itt nem részletezve végül a 12–153b. hálózatot kapjuk. Mint látható, itt a J–K tárolók egy léptetőregisztert alkotnak, melynél a visszacsatolás egy invertálás beiktatásával szerepel (a $\overline{Q}_2$ révén).

F.10.14. A frekvenciaosztókkal a 10.5.6. pontban foglalkoztunk. A feladatban szereplő 1/31-es osztót a 10–38. ábra elvi kapcsolása alapján tervezhetjük meg. Ennek értelmében a 12–154a. ábra teljesíti a 31-es osztást. A benne szereplő 1/15-ös osztót pedig egy 1/3 és 1/5 osztó páros, soros kapcsolásával állíthatjuk elő a 12–154b. ábra szerinti kapcsolással.

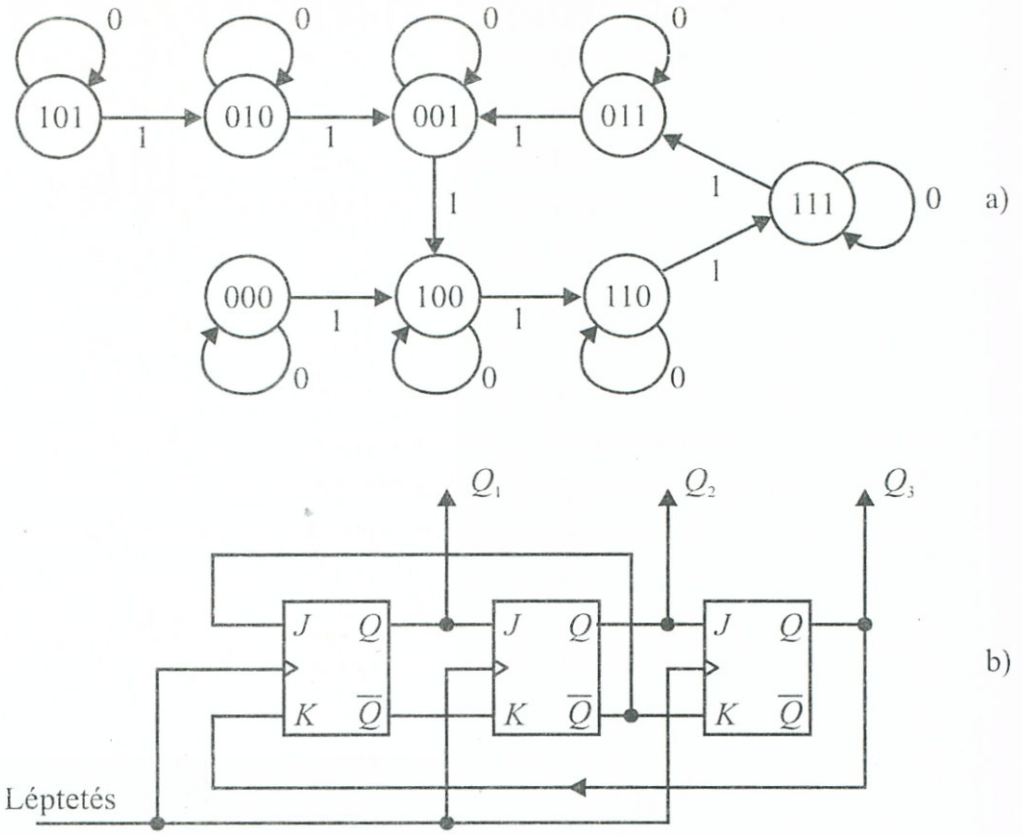
F.10.15. A feladat rokonságban van a 10–33. ábrán bemutatott kapcsolásokkal (mivel itt is ciklusrövidítés a feladat), de esetünkben *reverzíbilis* számlálót kell alkalmaznunk. Továbbá,



12–154. ábra *1/31-es frekvenciaosztó egy változata*

gondoskodni kell arról, hogy az ELŐRE, illetve HÁTRA irányokat kijelölő jelet, amíg a számlálási ciklus megkívánja, tároljuk. Végül ügyelni kell arra, hogy a megadott típusú modul *aszinkron* CLEAR és LOAD vezérléssel rendelkezik, amit (a 10–33. ábrán bemutatott módon) figyelembe kell venni a reverzálási kapujelek értékének megállapításakor. Példánknál ezért kellett az ELŐRE irányú léptetés végén 2 helyett 3-at, a HÁTRA irányú léptetés végén 4 helyett 3-at kapuzni. A realizációs hálózatot a 12–155. ábrán rajzoltuk fel.

F.10.16. A 12–141. ábrán megadott hálózat egy $M = 16$ – szinkron–párhuzamos–binér–reverzíbilis számláló, melynél az előre-számlálást a DOWN/UP = 0, a hátra-számlálást a DOWN/UP = 1 előválasztással lehet beállítani. A számlánc

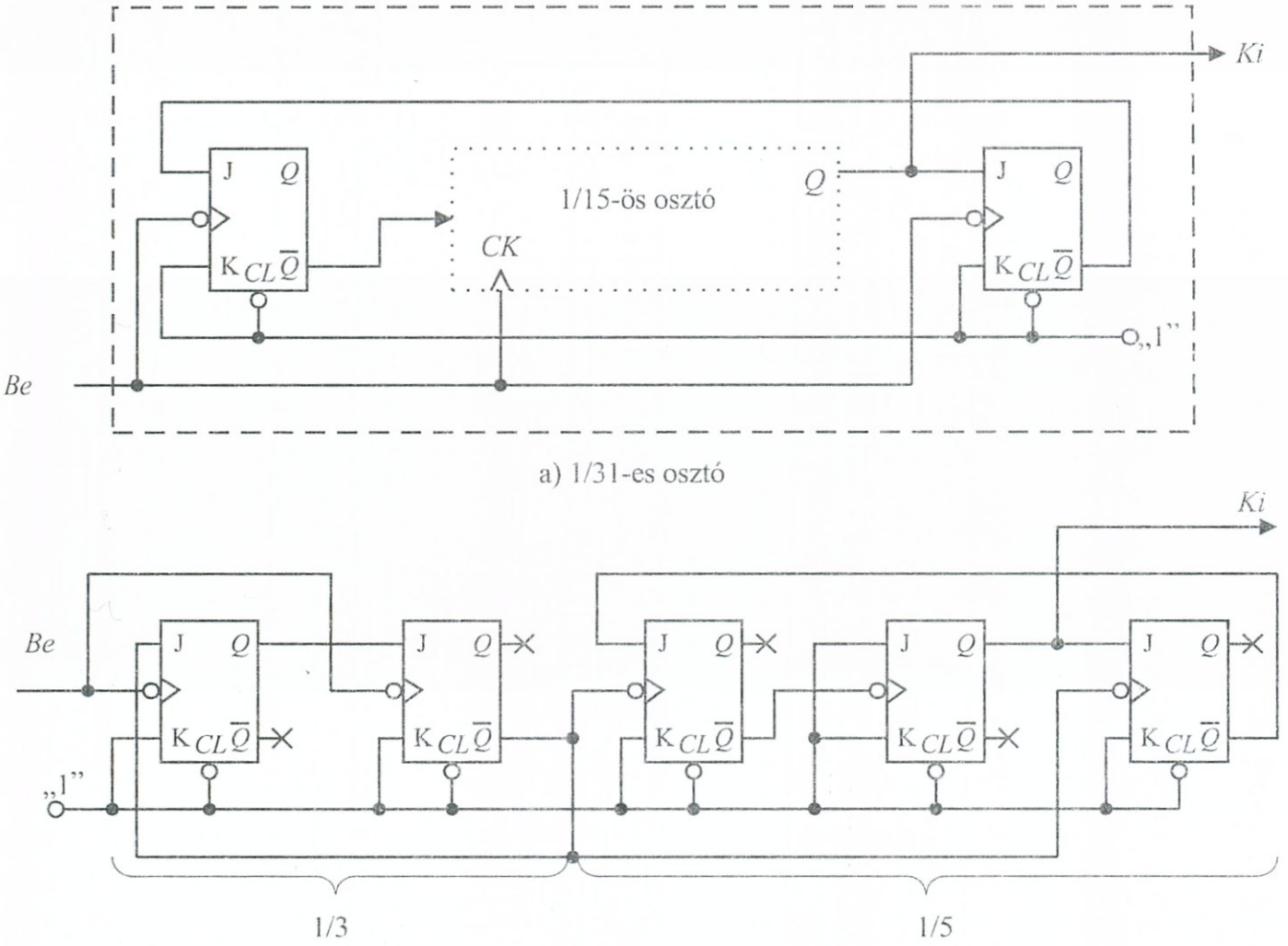


12-153. ábra *M = 5-ös modulusú JOHNSON számláló „self starting" kialakítással*

olyan megoldást jelent, mely (pl. bekapcsoláskor) bármelyik állapotból indulva biztosan beletalál a kívánt üzemszerű ciklusba. Esetünkben a kódolt állapotgráfot, ha a 12–153a. ábra szerint vesszük fel, akkor teljesülhet a fenti kívánalom. A megoldáshoz J–K tárolóelemeket választva és a tervezés lépéseit itt nem részletezve végül a 12–153b. hálózatot kapjuk. Mint látható, itt a J–K tárolók egy léptetőregisztert alkotnak, melynél a visszacsatolás egy invertálás beiktatásával szerepel (a $\overline{Q_2}$ révén).

F.10.14. A frekvenciaosztókkal a 10.5.6. pontban foglalkoztunk. A feladatban szereplő 1/31-es osztót a 10–38. ábra elvi kapcsolása alapján tervezhetjük meg. Ennek értelmében a 12–154a. ábra teljesíti a 31-es osztást. A benne szereplő 1/15-ös osztót pedig egy 1/3 és 1/5 osztó páros, soros kapcsolásával állíthatjuk elő a 12–154b. ábra szerinti kapcsolással.

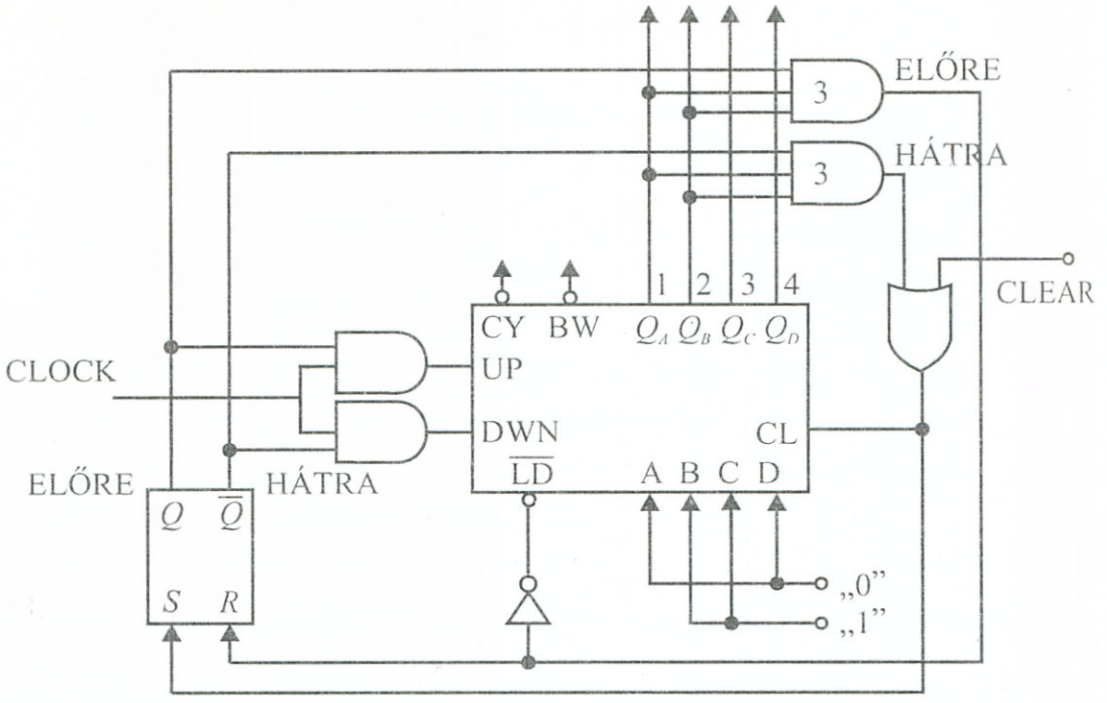
F.10.15. A feladat rokonságban van a 10–33. ábrán bemutatott kapcsolásokkal (mivel itt is ciklusrövidítés a feladat), de esetünkben *reverzíbilis* számlálót kell alkalmaznunk. Továbbá,



12–154. ábra *1/31-es frekvenciaosztó egy változata*

gondoskodni kell arról, hogy az ELŐRE, illetve HÁTRA irányokat kijelölő jelet, amíg a számlálási ciklus megkívánja, tároljuk. Végül ügyelni kell arra, hogy a megadott típusú modul *aszinkron* CLEAR és LOAD vezérléssel rendelkezik, amit (a 10–33. ábrán bemutatott módon) figyelembe kell venni a reverzálási kapujelek értékének megállapításakor. Példánknál ezért kellett az ELŐRE irányú léptetés végén 2 helyett 3-at, a HÁTRA irányú léptetés végén 4 helyett 3-at kapuzni. A realizációs hálózatot a 12–155. ábrán rajzoltuk fel.

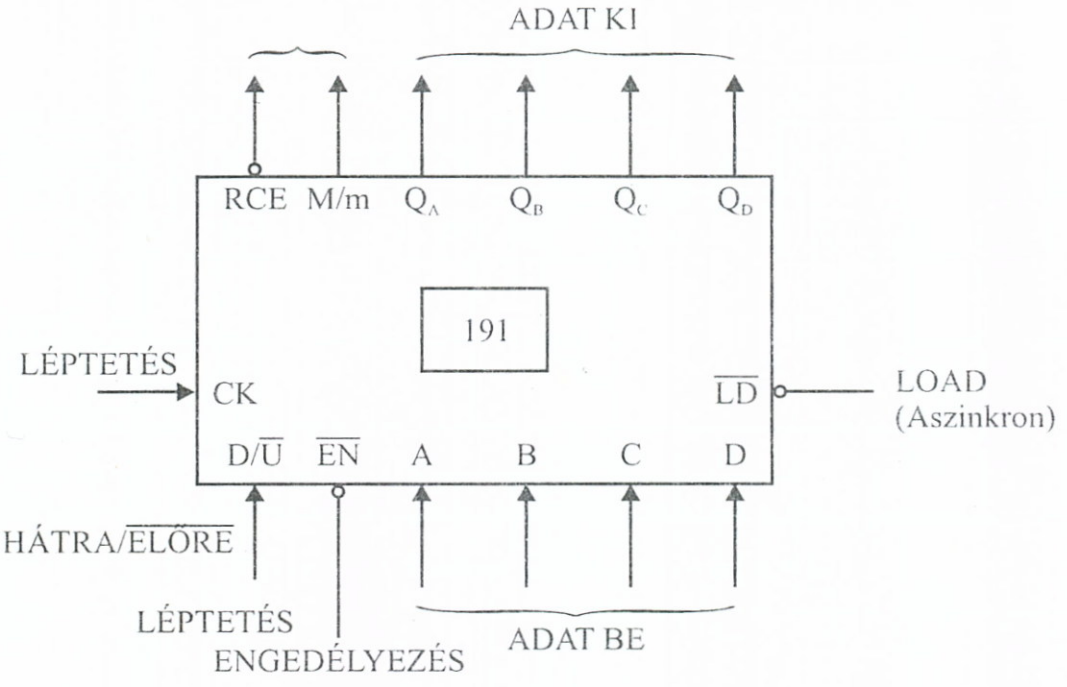
F.10.16. A 12–141. ábrán megadott hálózat egy $M = 16$ – szinkron–párhuzamos–binér–reverzíbilis számláló, melynél az előre-számlálást a DOWN/UP = 0, a hátra-számlálást a DOWN/UP = 1 előválasztással lehet beállítani. A számlánc



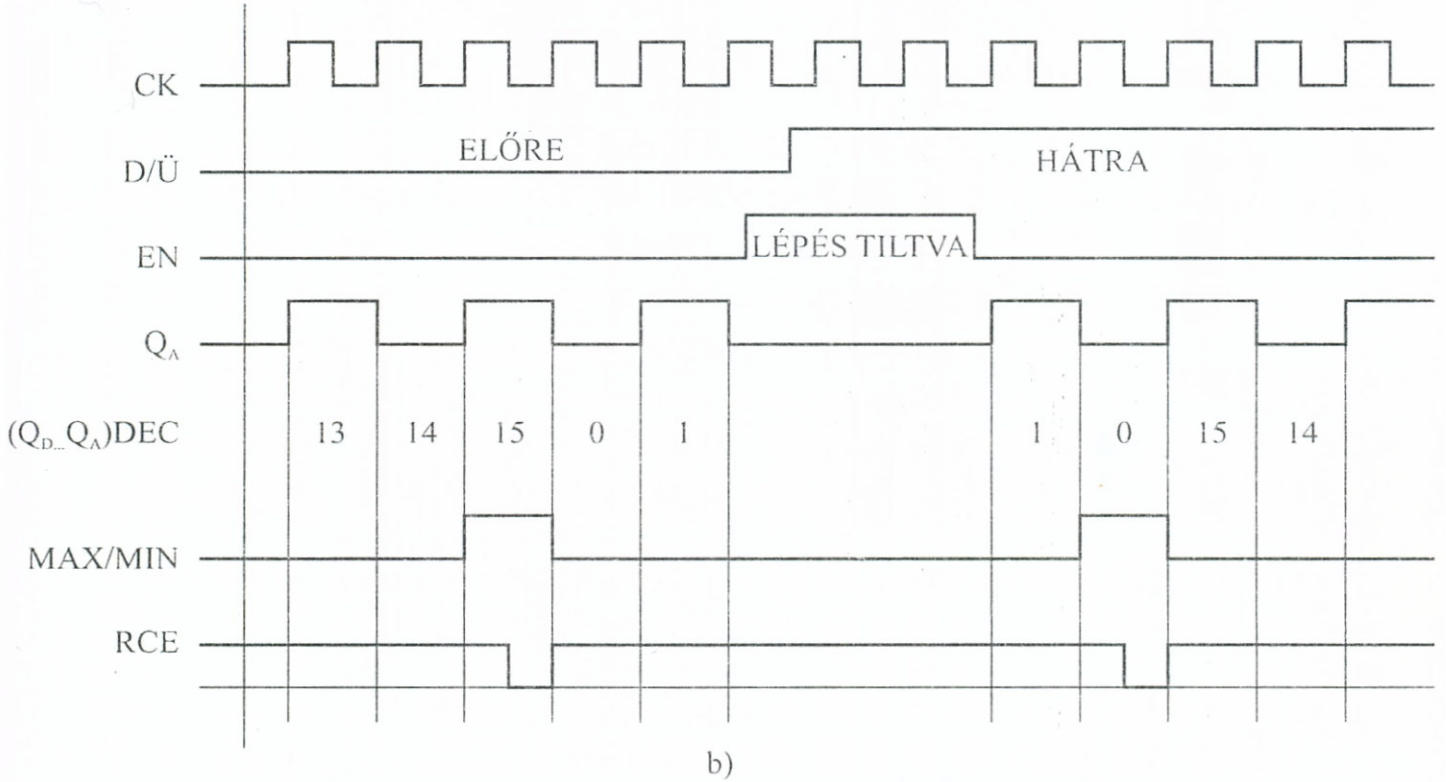
12-155. ábra *Reverzált számlálási ciklus megvalósítása a G.10.15. példánál*

rendelkezik a DATA INPUT-i bemenetek révén párhuzamos preszetelhetőséggel, melyet az aszinkron tulajdonságú LOAD (LD = 0) értékkel engedélyezhetünk. A léptetést az ENABLE (EN = 1) bemenő jellel tilthatjuk le. Bővítés céljára a RIPPLE COUNT (RC) és a MAX/MIN (M/m) kivezetések szolgálnak (12–156a. ábra). E kivezetések jelviszonyait a 12–15b. ábra idődiagramján tanulmányozhatjuk.
A bővítésre szolgáló RC és M/m kimenetek kombinálásával a számláncot háromféleképpen lehet bővíteni a 12–157. ábrán látható változatoknak megfelelően.

F.10.17. Az előző példában részletesen tárgyalt reverzíbilis szinkron számlánc M = 16 modulusú, így a megadott ciklus realizálására két fokozatúra bővített megoldás lesz csak alkalmas. Válasszuk a 12–157b. ábra szerinti bővítést és az előre/hátra irány ciklus alatti tárolására alkalmazzunk egy NAND kapupárból kialakított inverz S–R tárolót. Annak biztosítására, hogy a számláló biztosan a ciklusban induljon, biztosítanunk kell egy cikluson belüli értékre történő kezdeti beállítást is. Ezt az $\overline{LD} = 0$ időszakos beállításával biztosítjuk. A PRESET bemenetekre a 21-es értéknek megfelelő bináris kódot kötöttük be konstans szintekkel. Ily módon a kezdeti irányt külön nem is kell megadnunk, mivel 21-ről indulva



a)

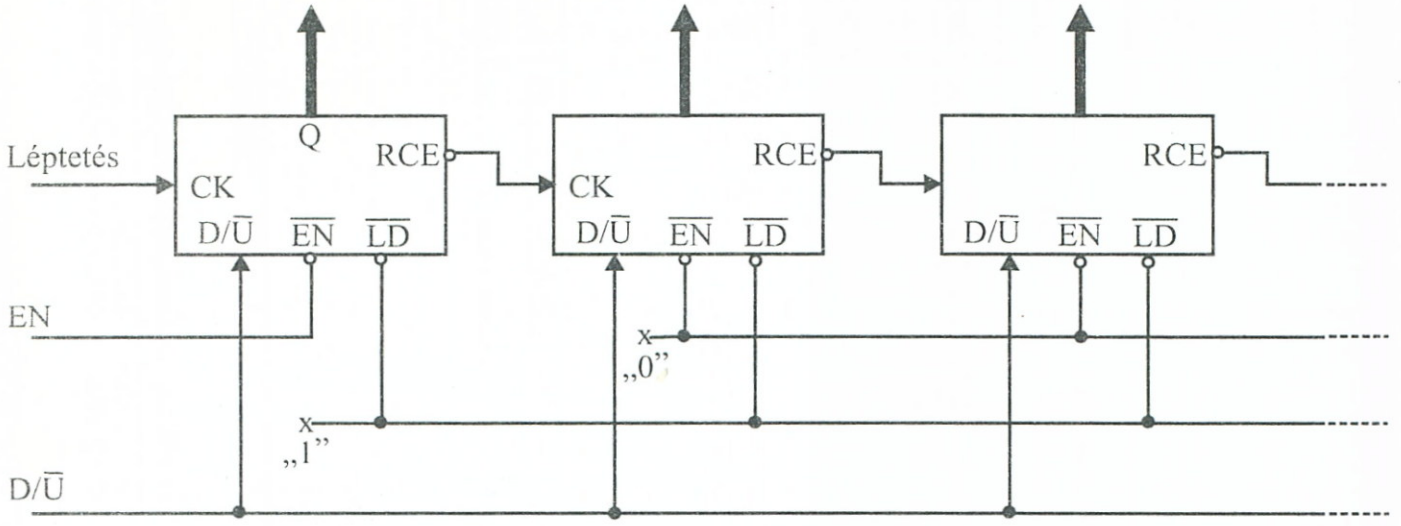


b)

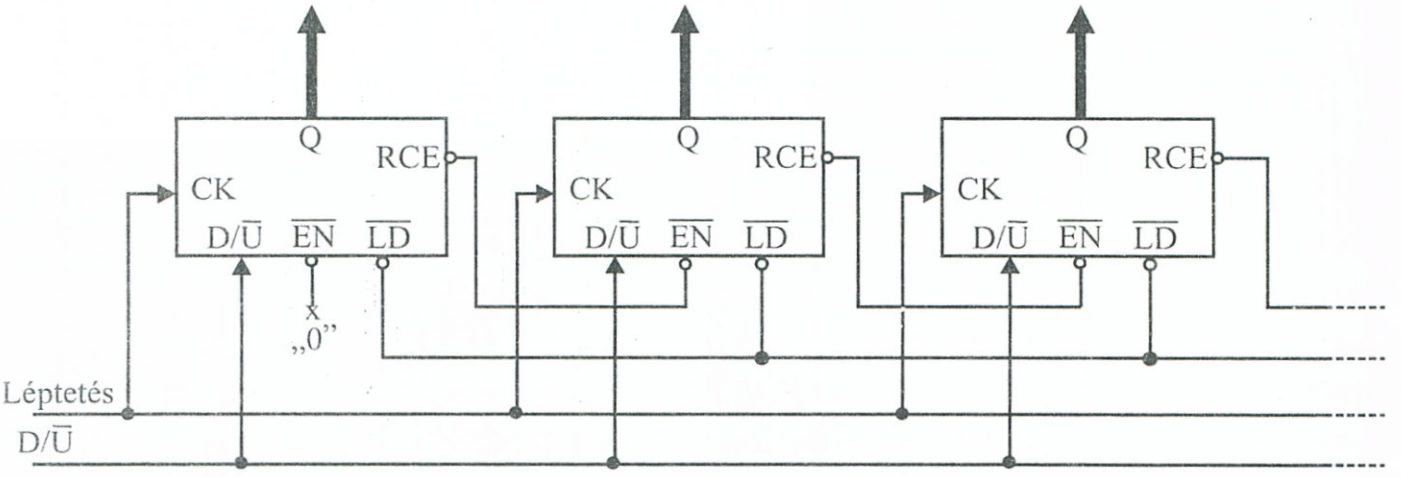
12-156. ábra *Egy 191 típusú reverzibilis számlánc elvi vázlata és működési időviszonyai*

UP vagy DOWN esetén is a cikluson belül maradunk. A teljes hálózat a 12–158. ábrán látható.

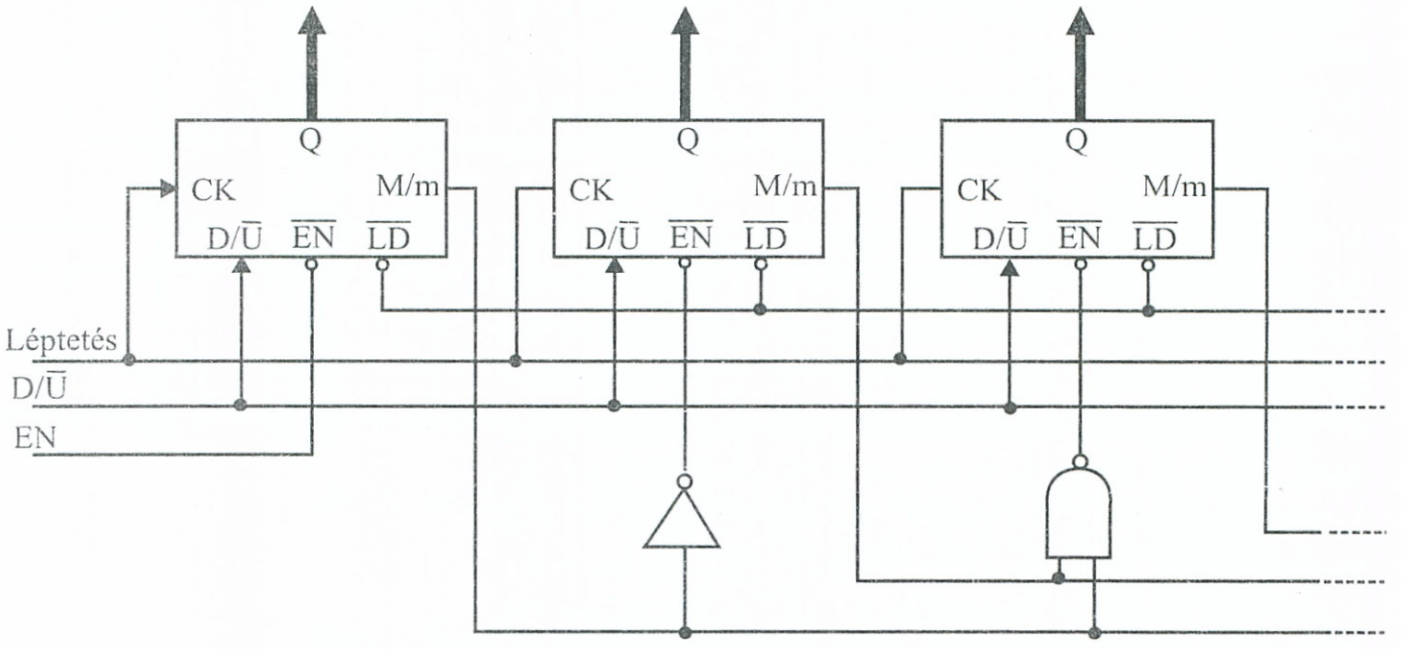
F.10.20. Négyzetre emelésnél első ötletként önként adódik, hogy egy szorzóáramkör mindkét bemenetére ugyanazt a számot vezessük s így a szorzat a szám négyzete lesz. Ha azonban



(a) Aszinkron kapcsolat, kaszkád elrendezés

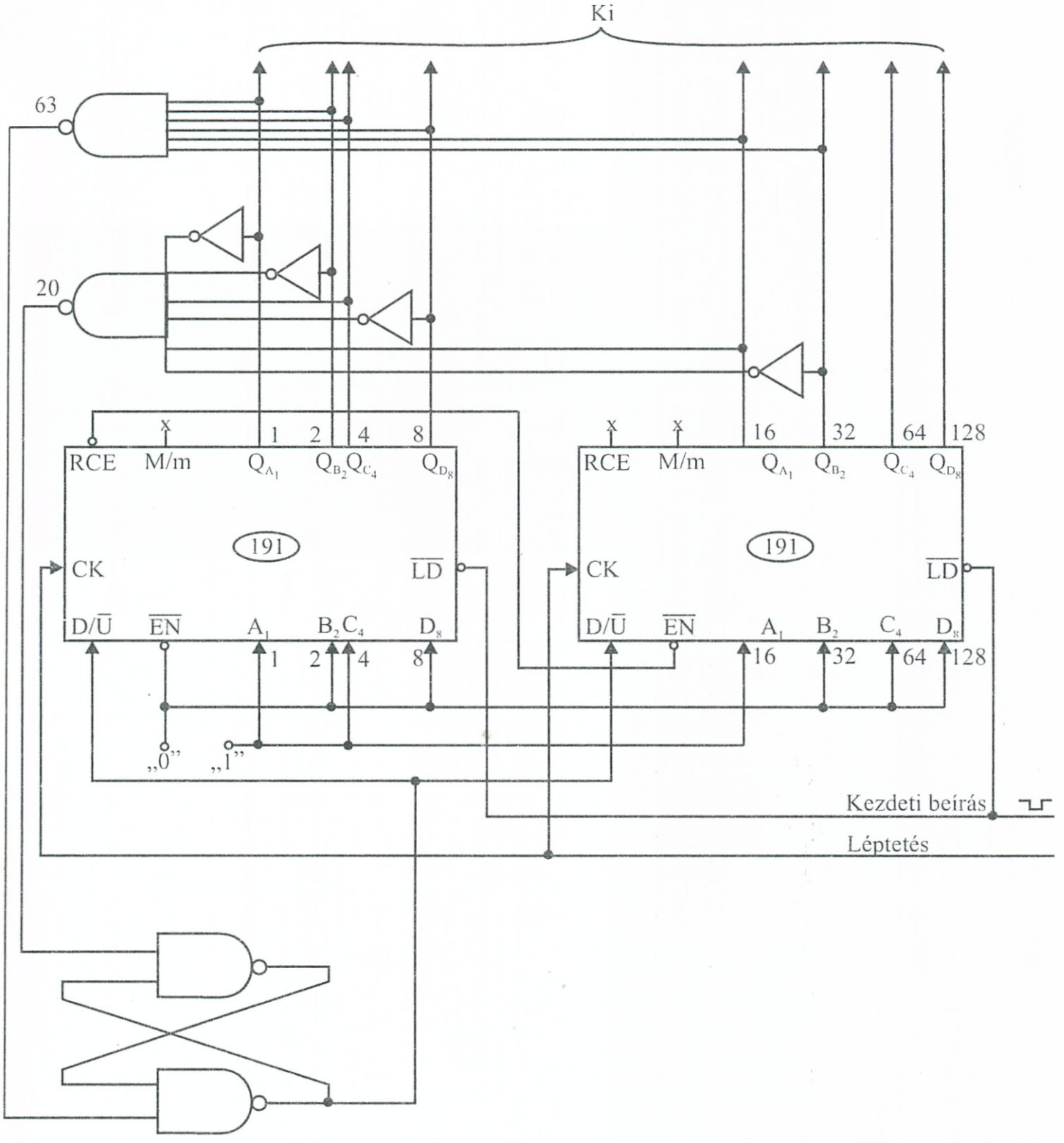


(b) Szinkron kapcsolat, kaszkád elrendezés



(c) Szinkron kapcsolat, párhuzamos elrendezés

12 - 157. ábra. *Bővítési változatok*



12 - 158. ábra. *Számlálás 20 ↔ 63 ciklusban*

kifejezetten a négyzetre emelés a cél, jóval egyszerűbb áramkörrel is célhoz érhetünk. Tekintsünk például egy 4-bites számot: $A_3A_2A_1A_0$. Állítsuk elő a négyzetre emelés részletszorzatait s vizsgáljuk meg ezeket, illetve az összeadásuk során keletkező jegyeket:

				A_3	A_2	A_1	A_0	$xA_3A_2A_1A_0$
				A_3A_0	A_2A_0	A_1A_0	A_0A_0	
			A_3A_1	A_2A_1	A_1A_1	A_0A_1		
		A_3A_2	A_2A_2	A_1A_2	A_0A_2			
	A_3A_3	A_2A_3	A_1A_3	A_0A_3				
S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0	

Figyelembevéve a logikai ÉS idempotenciáját és kommutativitását, ez a szorzat kissé egyszerűbb alakban is előállítható. Némi átrendezés után kapjuk:

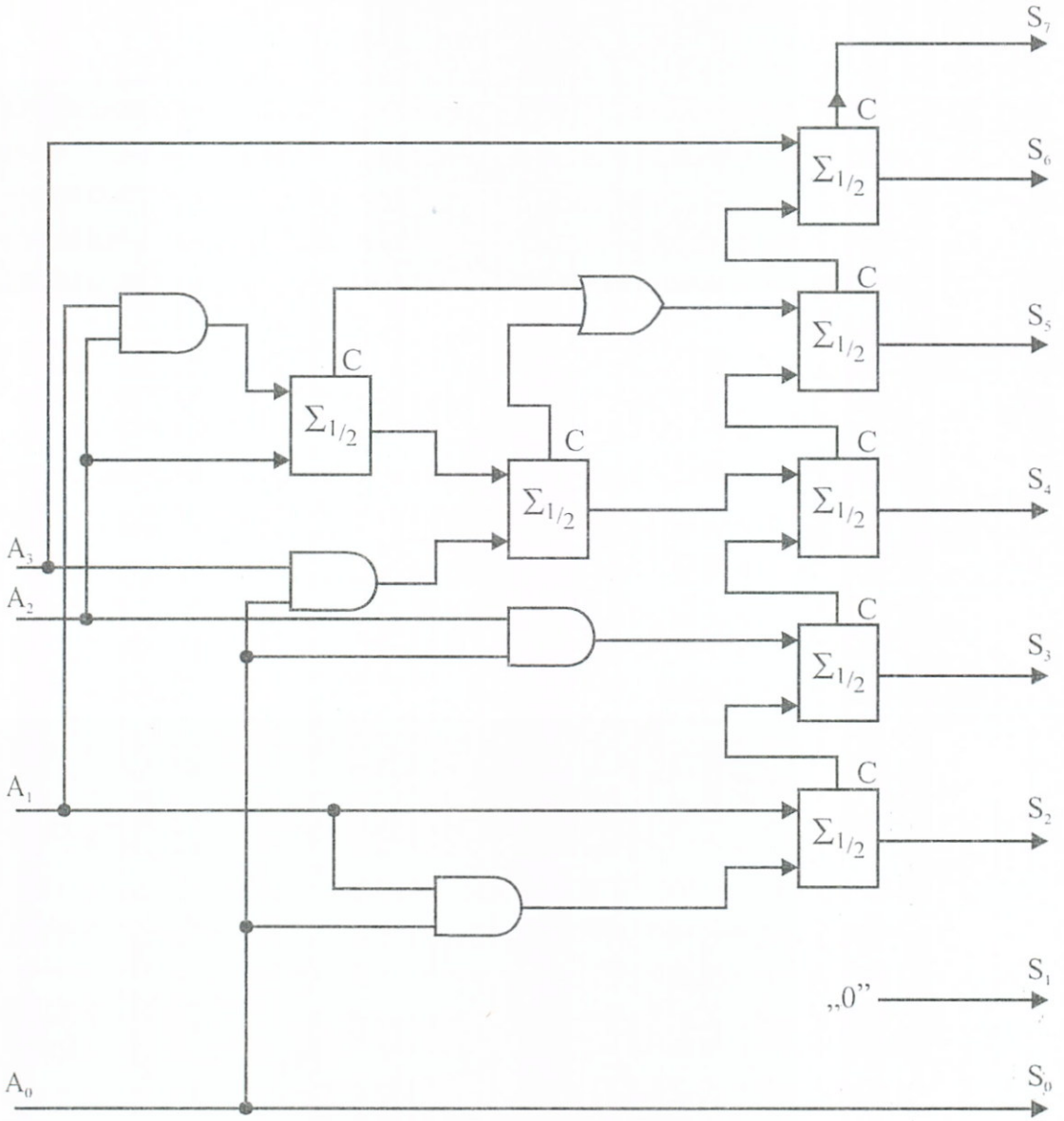
				A_3	A_2	A_1	A_0	$xA_3A_2A_1A_0$
				A_0A_3	A_0A_2	A_0A_1	A_0	
			A_1A_3	A_1A_2	A_1	A_0A_1		
		A_2A_3	A_2	A_1A_2	A_0A_2			
	A_3	A_2A_3	A_1A_3	A_0A_3				
S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0	

Az S_i bitekre a következők fognak teljesülni:

$$S_0 = A_0$$
$$S_1 = 0$$
$$S_2 = A_1 + A_0A_1$$
$$S_3 = A_0A_2 + C_3$$
$$S_4 = A_2 + A_0A_3 + A_1A_2 + C_4$$
$$S_5 = C_5$$
$$S_6 = A_3 + C_6$$
$$S_7 = C_6$$

A fentiek alapján már felrajzolhatjuk a 12–159. ábrán látható 4-bites négyzetre emelő áramkört.

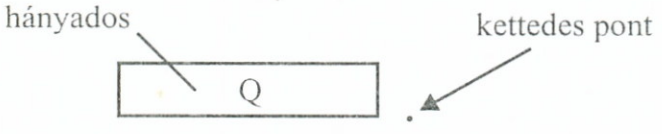
F.10.21. Bináris számok egész hatványának kiszámítására (hasonlóan a négyzetre emeléshez), a legegyszerűbb módszer az ismételt szorzás. Természetesen magasabb hatványnál ez meglehetősen költséges megoldás, hiszen nagyon áramkörigényes. Ügyesebb áramkör, amelyet a következő elven készítünk: Tegyük sorossá a szorzások elvégzését, s használjunk csupán egy szorzóáramkört, amelynek a szélessége elegendő



12-159. ábra *4-bites négyzetreemelő áramkör*

az $(n-1)$-ik hatvány méretű szorzandó fogadására is. Ezután megfelelő vezérlővel, a kiindulási operandust tekintve szorzónak, előállítjuk a négyzetet, majd ezt visszaírjuk a szorzandó helyére s ezt újra megszorozzuk az eredeti operandussal stb. Ily módon n-lépésben előáll az n-edik hatvány.

F.10.23. A feladat tehát az, hogy megvizsgáljuk, miként használható fel a 10–76a. ábrában szereplő berendezés az $A : B$, ha $A \geq B$ ún. *egészhányadosú* osztásnál. Ilyenkor a 10–75. ábrabeli helyzet a 12–160. ábrabelivé változik meg:



12-160. ábra. *Hányados egész-hányadosu osztásnál*

Ha a (10.41)-re vezető levezetés kiindulásánál *n-lépéssel jobbra* léptetjük el az osztandót, akkor „látszólag" *kisebbé tesszük* az osztónál, azaz hasonló feltételeket teremtünk, mint a „tört-hányadosú" osztás algoritmusánál fennálltak. Ezután – mint látni fogjuk – az algoritmus egész-hányadosú osztásra is használható lesz. Az elmondottakat formulárisan is elvégezve, itt

$$R_o = \mathrm{A}\cdot 2^{-\mathrm{n}}$$

a korábban levezetett részmaradék formulába a mostani R_o-t behelyettesítve és a számításokat elvégezve kapjuk.

$$R_n = A - B(2^{n-1}\cdot Q_1 + 2^{n-2}\cdot Q_2 + \ldots + Q_n)$$

innen átrendezéssel végül kapjuk:

$$\frac{A}{B} = \sum_{i=1}^{n} Q_i \cdot 2^{n-1} + \frac{R_n}{B}$$

ahol a hányados – mint a formulákból látható – nem tört. A szükséges blokkséma azonos lehet a 10–76a. ábrabelivel, viszont a 10–76b. állapotgráf egyes állapotainak tartalma módosul. Így a *-os sorok változásaival:

α állapot: Indulás előtti feltöltésnél:

OSZTANDÓ	$: A\cdot 2^{-\mathrm{n}}$	$\rightarrow Q$	*
OSZTÓ	$: B$	$\rightarrow X$	
TÖRLÉS	$: 0$	$\rightarrow Z$	*
TÖRLÉS	$: 0$	$\rightarrow CY$	
CIKLUSSZÁM	$: n$	$\rightarrow C$	

és az osztás után:

ρ állapot:

Q = HÁNYADOS	(Q)
Z = MARADÉK	(R)
X = OSZTÓ	(B)
C = CIKLUS VÉGE	(0)

Osztandó: $|A| = |-9_{10}| = 1001_2$ (Q-ba betöltve) n=4
Osztó : $|B| = |+4_{10}| = 0100_2$ (X-be betöltve) B_k=1100

	CY	Z	Q	Q_i	C	
$A \cdot 2^{-4} = R_0$	0	0000	100	1	4	
$2R_0$		0001	001			←
$+B_k$		1100				
	0	1101		0		Vissza-
$+B$		0100				állítás
R_1	↯	0001	001	0	3	
$2R_1$		0010	010			←
$+B_k$		1100				
	0	1110		0		Vissza-
$+B$		0100				állítás
R_2	↯	0010	010	0	2	
$2R_2$		0100	100			←
$+B_k$		1100				
R_3	1	0000	100	1	1	
$2R_3$		0001	001	0		←
$+B_k$		1100				
	0	1101		0		Vissza-
$+B$		0100				állítás
R_4	↯	0001	001	0	0	
		Maradék	Hányados			
Kettedes vessző		$2^{-1}2^{-2}2^{-3}2^{-4}$	$2^{-1}2^{-2}2^{-3}2^{-4}$			

Maradék: $R = 0001{,}0_2 = 1_{10}$
Hányados: $Q = 0010{,}0_2 = 2_{10}$
Előjel: $S_E = S_{\text{osztó}} \oplus S_{\text{osztandó}} = 0 \oplus 1 = 1 \longrightarrow$ NEGATÍV

12 - 161 ábra. *Bináris osztás „egészhányadossal”*

Mivel itt is előjel-abszolútértékes ábrázolás van, az előjelet a (10.43) összefüggéssel számíthatjuk.

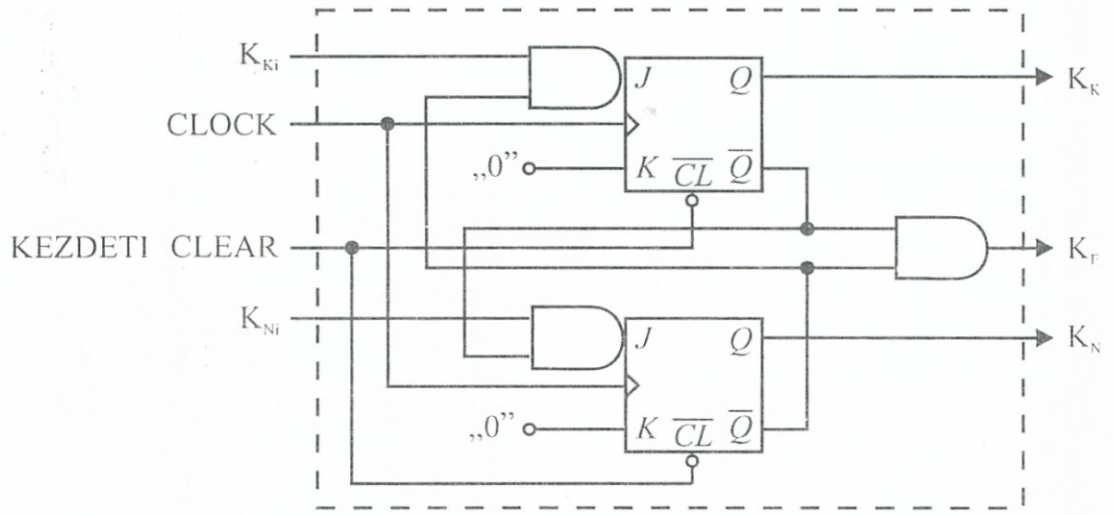
Szemléltetésül oldjuk meg a következő példát, melynek lépéseit a 12–161. ábrán követhetjük.

$$(-9) : (+4) = ?$$

Az *ellenőrző-próba* itt is a (10.38) alapján történhet:

$$\frac{|A|}{|B|} = \frac{9}{4} = Q + \frac{R}{B} = 2 + \frac{1}{4} = \frac{8}{4} + \frac{1}{4} = \frac{9}{4}$$

F.10.24. A 10–84. ábra soros komparátoránál szereplő KIÉRTÉKELŐ HÁLÓZAT-ra azért van szükség, mert a K_K, K_E, K_N kimeneteken megjelenő <, =, > eredménynek a *teljes, „n"-bites kódszóra* kell érvényesnek lenni. A KIÉRTÉKELŐ bemenetére érkező K_{Ki}, K_{Ei}, K_{Ni} értékek a léptetés során, minden bit-pozícióban változ(hat)nak. Az A, B operandusok nagyobb, ill. kisebb relációja akkor dől el egyértelműleg, mikor a léptetés során valamelyik bit-pozíciónál az $A_i \gtrless B_i$ reláció kialakul. Ezután a léptetést már nem kellene folytatni, mivel az újabb (alacsonyabb helyértékű) bit-pozíciós összehasonlítások „felülírnák" a tényleges eredményt. Ennek elkerülésére az elsőként bekövetkezett $A_i \gtrless B_i$ esetet tárolni kell, mivel ez lesz a végleges eredmény. Ha az A, B kódszavak egyenlőek, akkor a teljes „n" hosszra le kell folytatni a bit összehasonlításokat és csak az utolsó A_0, B_0 összehasonlítását követően mondhatjuk ki az egyenlőséget. Egy ilyen kiértékelő hálózat realizációt rajzoltunk fel a 12–162. ábrán.



12-162. ábra *KIÉRTÉKELŐ HÁLÓZAT soros komparátorhoz*

F.10.25. A feladat valójában egy 1/4-es frekvenciaosztó-hálózat, de az érdekesség a RAM-mal történő realizálás igényében rejlik. A korszerű digitális berendezésekben gyakran alkalmazzák az ún. átprogramozható (vagy újraprogramozható) logikákat, melyeknél a RWM-típusú memóriákba töltik be a megoldandó feladat *struktúráját* is a paramétereken túlmenően. Egy másik struktúra betöltésével valójában egy *másik áramkört* személyesítünk meg, miáltal az ún. többcélú digitális rendszerek felé teszünk egy lépést. Tulajdon-

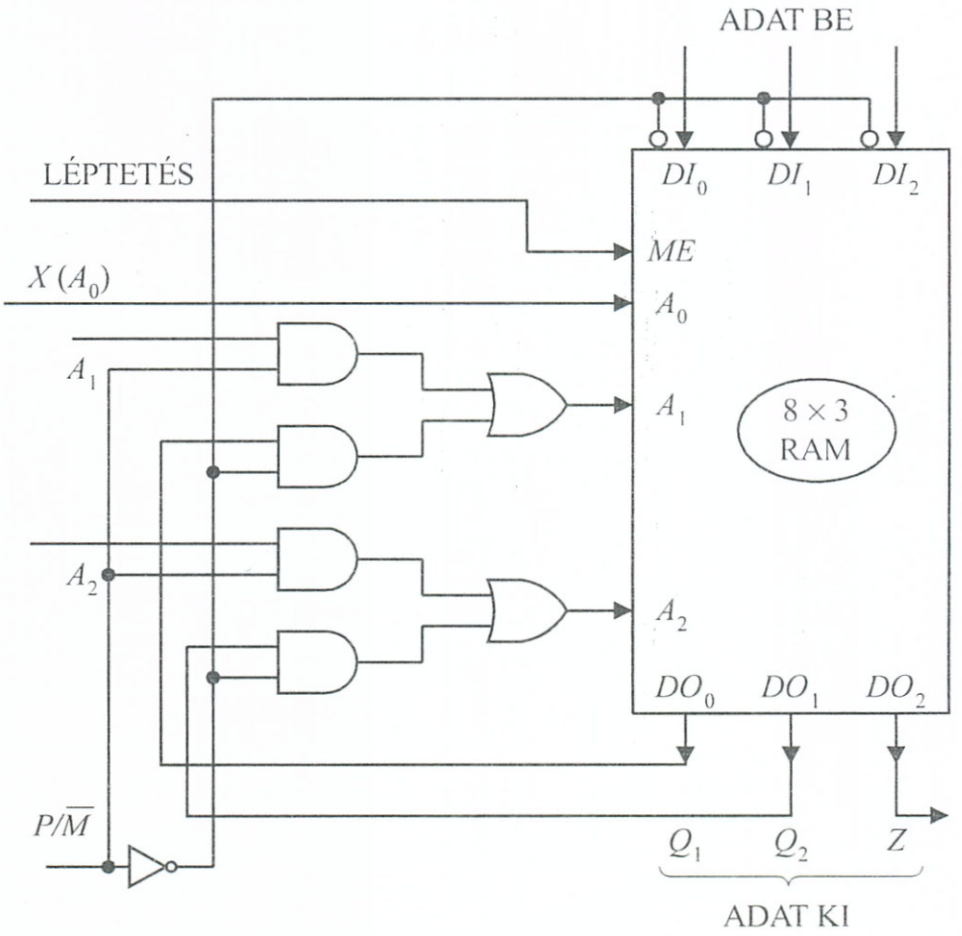
képpen maguk az univerzális digitális számítógépek is ilyen *többcélú* rendszerek.

Példánkra visszatérve, a 12–163a. ábrán kitöltöttünk egy kódolt táblázatot, mely az előírt működésnél kiindulásul szolgálhat. Ez a $T \rightarrow (T + 1)$ ütemeket kapcsolatba állító MEALY-modellen alapuló táblázat (melyet a sorrendi hálózattervezésnél gyakran használtunk), itt úgy is felfogható, mint a realizációnál felhasználandó RAM CÍM–ADAT-táblázata a működési ciklus esetén.

Miután a működési ciklus állapotainak száma 8, ezért a szükséges memóriakapacitás 8 szó és a változók száma alapján a minimálisan szükséges szóhossz 3 bit. A 12–163b. ábrán felrajzoltunk egy visszacsatolt sorrendi hálózatot, melynek révén a $(T + 1)$-edik időütemek címzésére az előzőleg kiolvasott adatoknak is lehetőséget adtunk. A memória működésének az X bemenet változásaival történő összehangolását (pl. a tran-

T			$T+1$		
X	Q_1	Q_2	Q_1	Q_2	Z
0	0	0	0	0	0
1	0	0	0	1	0
0	0	1	0	1	0
1	0	1	1	0	0
0	1	0	1	0	1
1	1	0	1	1	1
0	1	1	1	1	1
1	1	1	0	0	1
A_0	A_1	A_2	DO_0	DO_1	DO_2
CÍM			ADAT KI		

a)

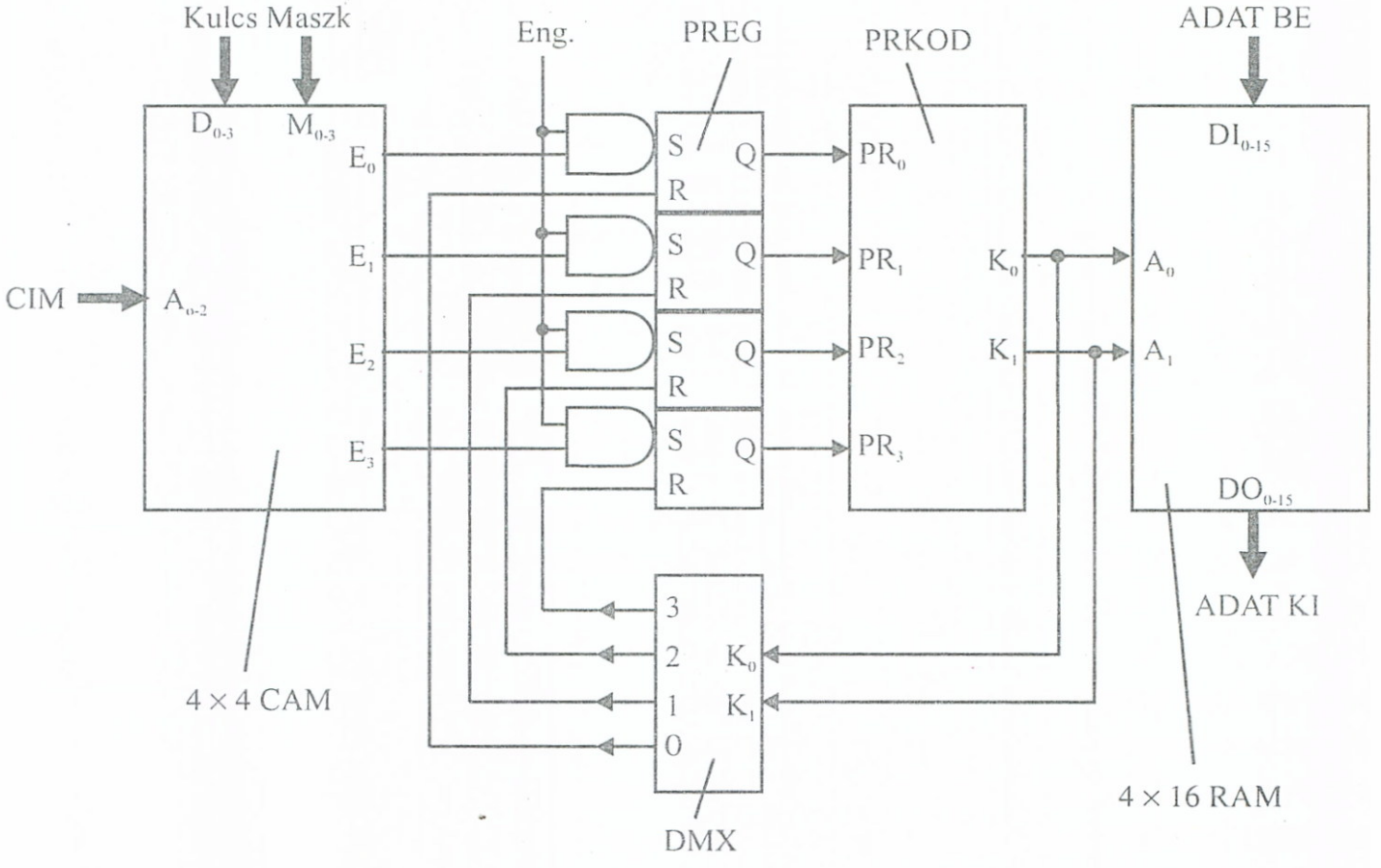


b)

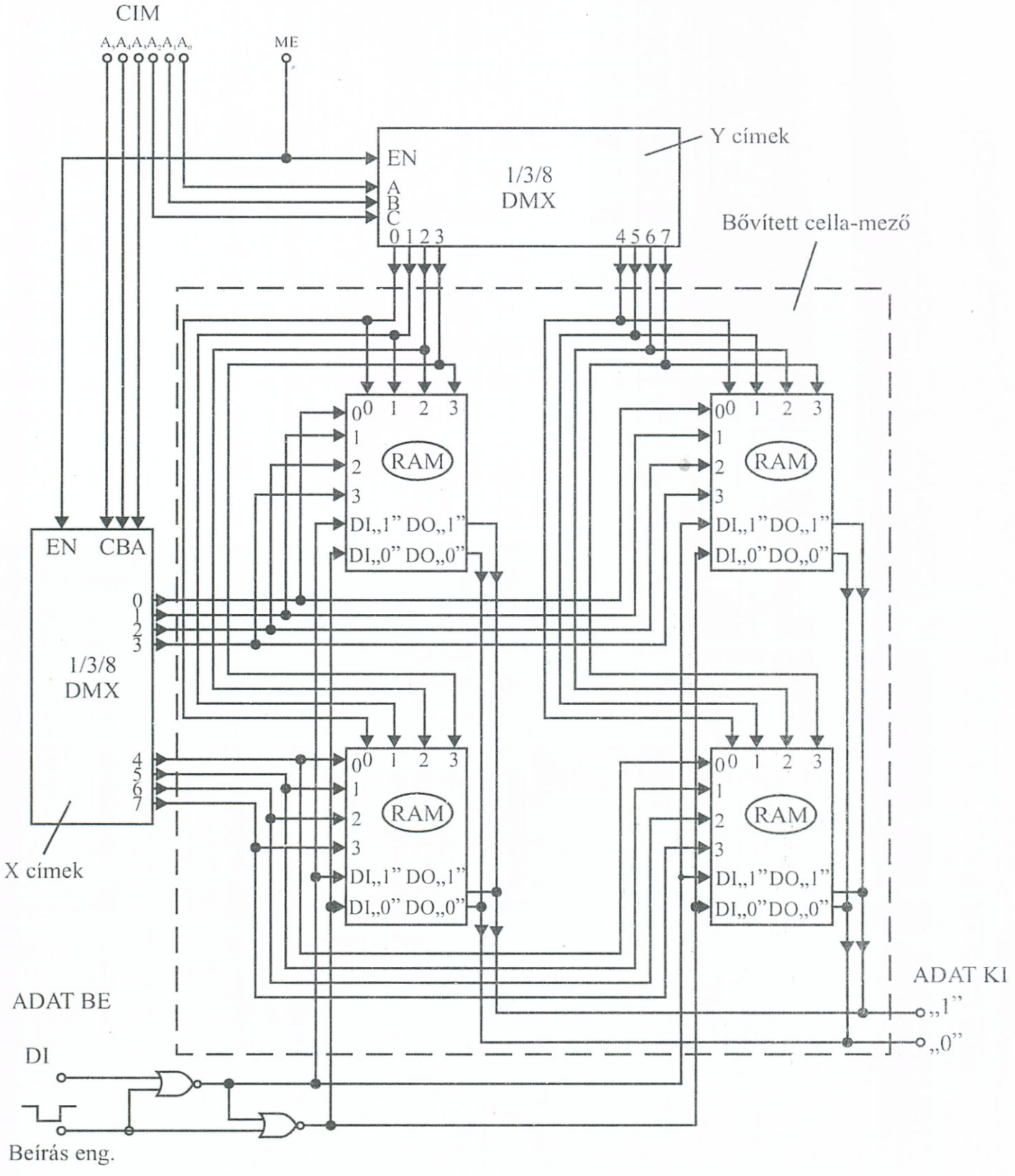
12–163. ábra *1/4-es frekvenciaosztó kialakítása RAM-mal*

ziensek hatásának kiküszöbölése céljából) az *ME* bemenetre adott léptetéssel is elősegíthetjük. Az áramkörnél feltüntettünk előzetes beíró áramköröket is, melyek révén az üzemszerű használat előtt a RAM adatokkal feltölthető az $A_0, A_1 A_2$ CÍM vezetékek és a $\overline{DI_0}, DI_1, DI_2$ adatbeíró vezetékek révén. A feltöltést a $P/\overline{M}$ üzemmód választóbemenet $P/\overline{M} = 1$ állapotával engedélyezzük, mikor is a visszacsatoló hurok inhibitálódik. $P/\overline{M} = 0$ esetén az A_1, A_2 és az adatbemenetek leválasztódnak, a visszacsatoló-hurok aktivizálódhat, valamint A_0 X bemenetté alakul át.

F.10.26. A feladatmegoldásnál a 10–100. ábrán vázolt gyorsított címzési elvet alkalmazhatjuk. Mivel itt több „*E*" alapján kell döntenünk, célszerű ezek között prioritási sorrendet kialakítanunk. Ennek érvényesítésére a 12–164. ábrán a CAM kimenetén megjelenő E_i-ket átmenetileg egy P–P típusú regiszterben tároltuk (PREG), mely egy *prioritásos kódátalakítóhoz* csatlakozik (PRKÓD). Az összekötés a kimenetekhez rendelt PR_i prioritások figyelembevételével történt. A PRKÓD kimenetei képezik a csatlakozó RAM *címző* beme-



12 - 164.ábra. *CAM - RAM gyorsmemória prioritásos címzéssel*



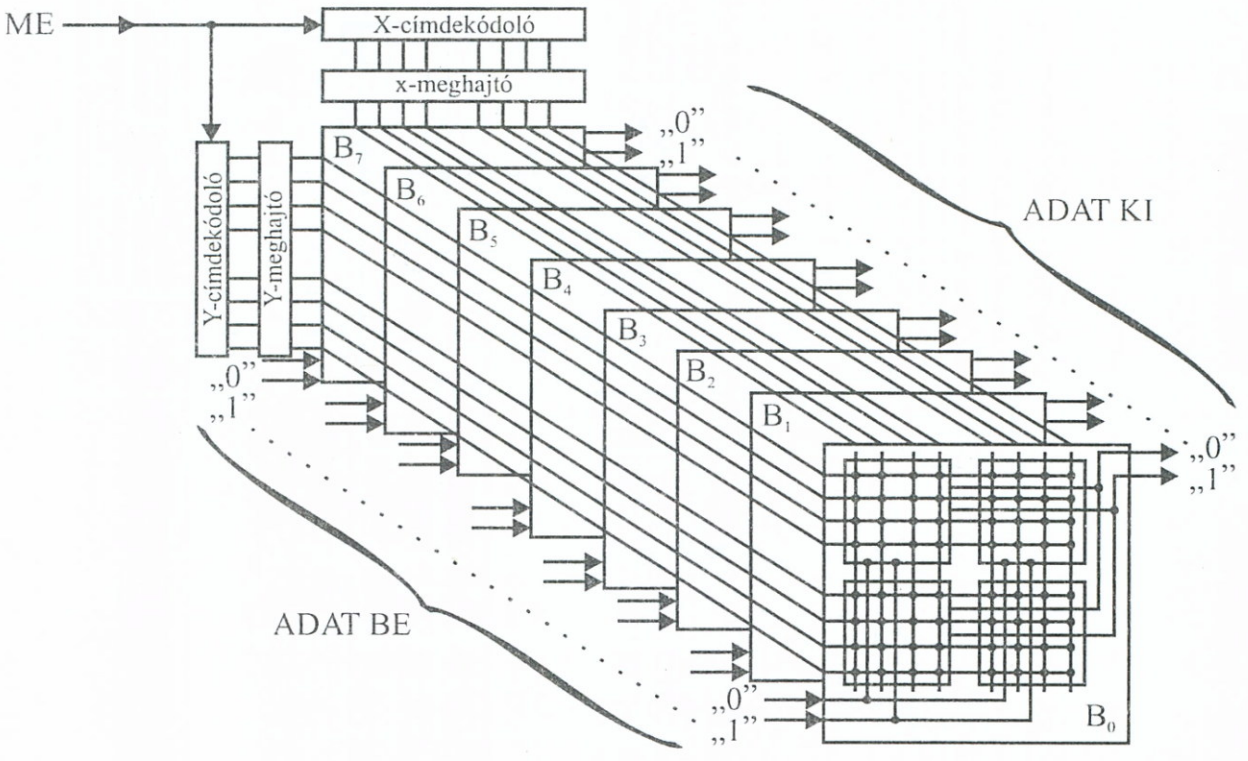
12-165. ábra *64×1 bites bitszervezésű RAM*

neteit, melyeken mindig a legmagasabb prioritás érvényesül. Egy adott PR_i-hez tartozó címzés befejezése után a „használt" cím-kódot egy kódátalakítóként működtetett DMX-re vezetjük, melynek az így aktivizált kimenete törli

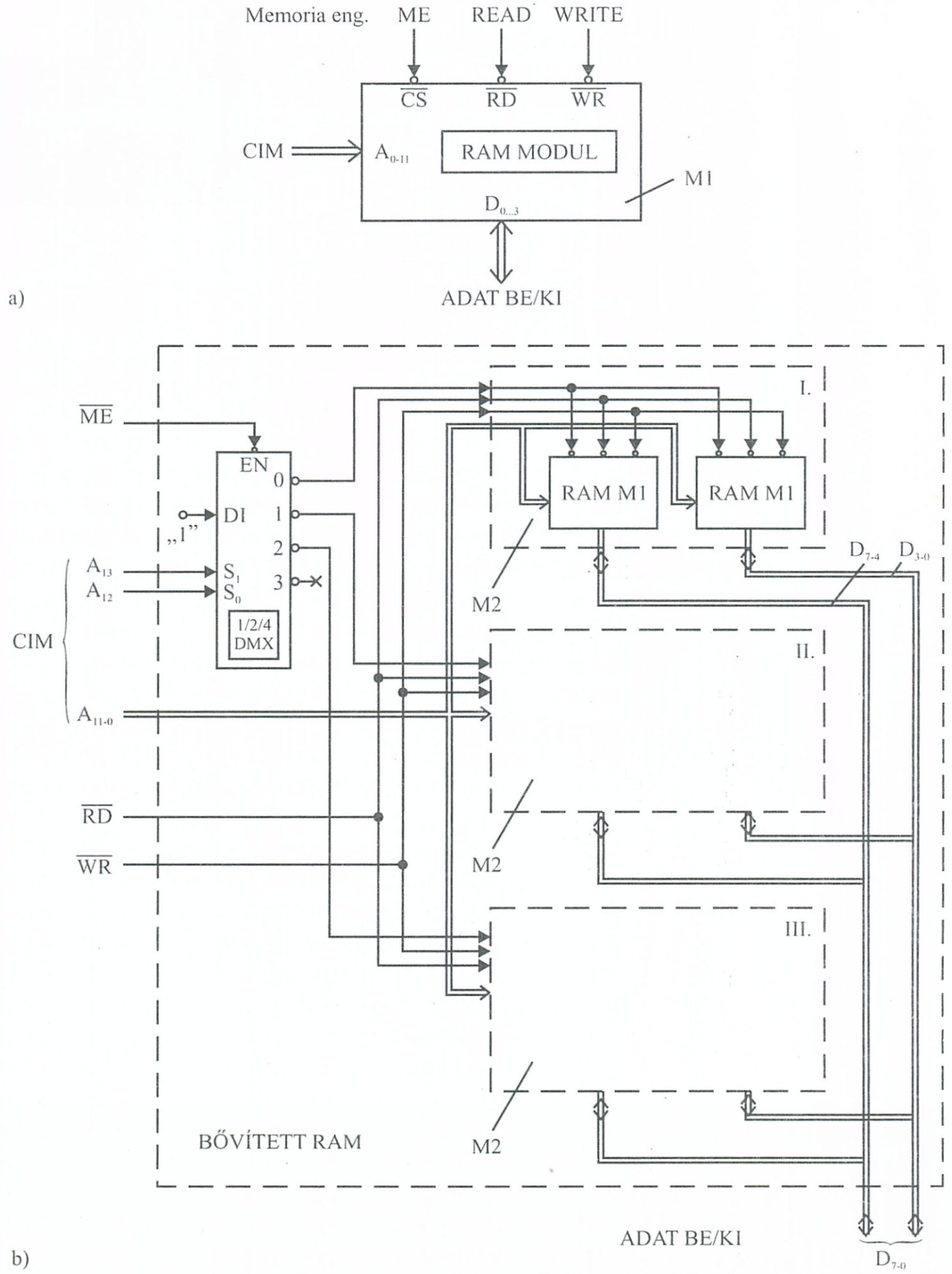
a PR_i címzés tárolóját, miáltal a PR_{i+1} prioritás lesz a legmagasabb és ez fogja meghatározni a következő RAM-cím kijelölését.

F.10.27. Esetünkben – mivel a szóhossz 1 bit – csupán *kapacitás-bővítésre* van szükség, és ehhez 64:16 = 4 db modulra lesz szükségünk. Tételezzük fel, hogy az adatkimeneti csatlakozások három-állapotúak, így másik modullal összeköthetők. A 64×1-bit *X–Y* irányú címzéséhez 3–3 bit ($2^6 = 64$) szükséges, melyet 1/3/8-as DMX-ek CÍMDEKÓDER-kénti felhasználásával biztosíthatunk. A DMX-ek EN bemeneteinek összekötésével előállíthatjuk a memória *ME* engedélyező bemenetét. A teljes hálózatot a 12–165. ábrán rajzoltuk fel.

F.10.28. A kiinduló modult a 12–166. ábrában látható négyzetek képviselik, melyek 64×1-bit kapacitásúak, ennek megfelelően 8×8 db tárolócellás mátrixot alkotnak. A rajzon minden pont egy-egy tárolócellát jelöl. Ebben a példában címkapacitás-bővítést nem kell alkalmaznunk, a kívánt 8-as szóhossz miatt viszont szóhossz-bővítés szükséges. Ezt például a 12–166. ábrán látható elvi elrendezéssel oldhatjuk meg, melynél az X–Y CÍMDEKÓDEREK megnövekedett



12-166. ábra *64×8-as szószervezésű RAM kialakítása bővítéssel*



12-167. ábra *RAM szóhossz-, és kapacitásbővítés*

áramköri terhelése miatt szükségessé váló MEGHAJTÓ fokozatokat is feltüntettük.

F.10.29. A 12-167a. ábrán látható megadott RAM-modult megvizsgálva, megállapítható:

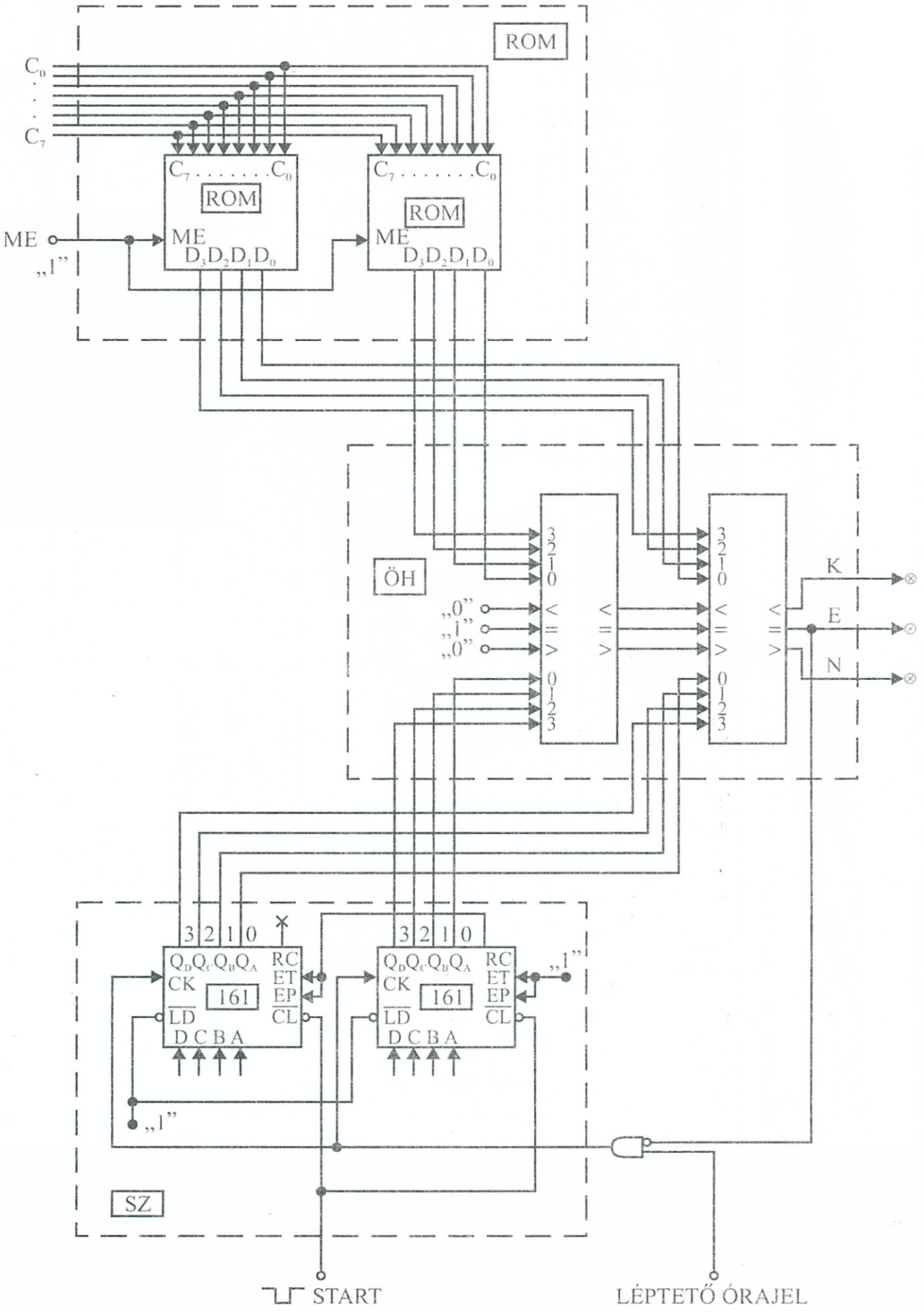
szóhossz:	4 bit	(D_{0-3}-ból)
kapacitás:	4 kszó	(A_{0-11}-ből)

A kívánt 12 kByte-os memória kialakításához szükséges paraméterek:

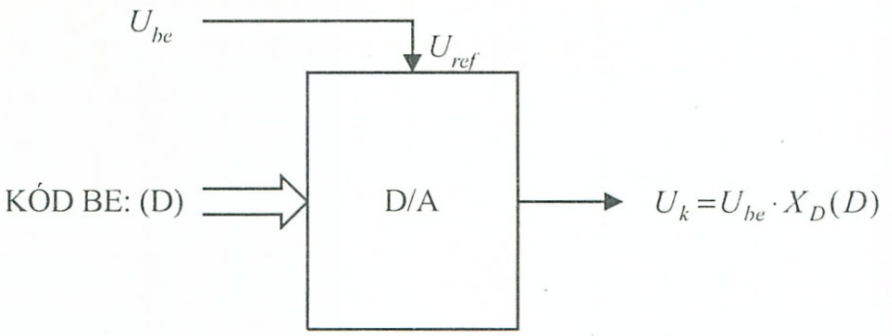
szóhossz:	8 bit	(1 Byte = 8 bit)
kapacitás:	12 KByte	

Azaz, mind szóhosszbővítésre, mind kapacitásbővítésre szükség van. A feladatot két lépésben oldjuk meg. Először szóhosszbővítést végzünk 8 bitre, melyhez 2 kiinduló M1-es modulból kialakítunk egy M2-es modult, majd ez utóbbiból annyit kapcsolunk össze, hogy a kívánt 12 kByte-os kapacitás kiadódjék. Az I, II. és III. modulegyüttes megcímzéséhez további *címzőbit*ek is szükségesek, melyeket egy 1/2/4 DMX-esn keresztül csatlakoztatunk az M2 modulok $\overline{CS}$ bemenetére. Az olvasás/írás vezérléshez a modulok $\overline{RD}/\overline{WR}$ bemeneteit is össze kell kapcsolni. (12-167b. ábra)

F.10.30. A feladatmegoldáshoz felhasználandó 4 bites modulokhoz tételezzük fel *számlálónak* a 10-29. ábra „161"-es modulját, *komparátornak* a 10-87. ábra „85"-ös modulját, és *ROM*-nak a 10-10a. ábra „ROM"-moduljának egy 4 bites változatát. Mivel a példa előírásai szerint, a kialakítandó berendezésben 8 bites adatok szerepelnek, ezért mindegyik összetevőnél *bővítést* kell alkalmaznunk.
A *számláló-bővítés* a 10-34. ábrán bemutatott elvek szerint történhet, a *komparátor-bővítés*hez a 10-88. ábra szolgálhat kiindulásul. A *ROM-bővítésnél* (a 8. címbemenet miatt) kapacitás-bővítés nem kell, így csupán szóhossz-bővítést kell végeznünk a 10-101b. ábrán bemutatott módon. A teljes hálózat a 12-168. ábrán került felrajzolásra. Az indító START jel a számlálót nullázza, majd ez az órajelek hatására mindaddig előre lépked, amíg a komparátor E egyenlőség jele – a léptető órajel letiltásával – a működést le nem állítja.



12-168. ábra *Összetett bővítési példa*



12–169. ábra *Szorzó D/A átalakító*

F.10.31. Mindazon *D/A* átalakítókat, amelyek széles tartományban változtatható referenciafeszültséggel működtethetők, szorzó (multiplying) átalakítóknak nevezik. Használatos a digitális, vagy digitálisan vezérelt potenciométer elnevezés is. Ha felírjuk egy átalakító kimeneti feszültségét (12–169. ábra).

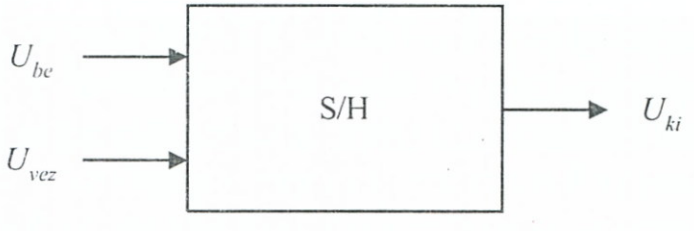
$$U_K = U_{ref} \cdot X_D(D) = U_{b \cdot e} X_D(D)$$

akkor ebből azonnal látható, hogy U_A két bemeneti mennyiség szorzata, amelyek közül az egyik analóg feszültség U_{ref}, a másik pedig egy digitális kódtól (D) függő szám, amelynek értéktartománya:

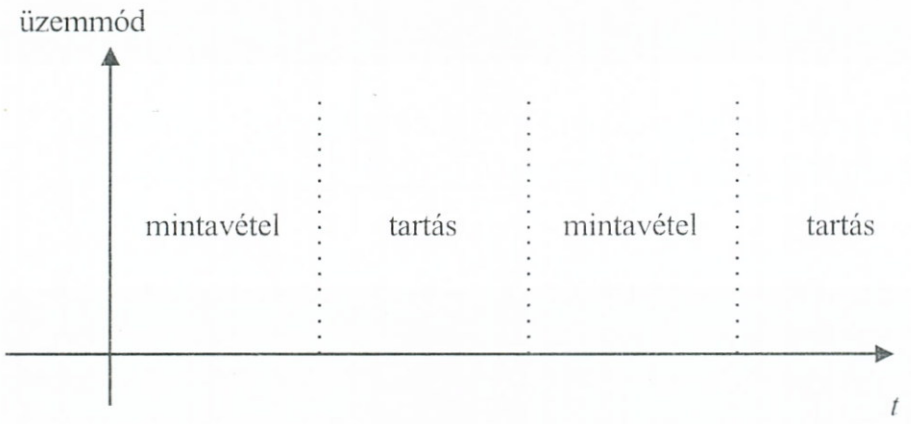
$$0 < X_D < 1$$

A felírt összefüggés unipoláris esetre vonatkozik. Ha U_{ref} kétpolaritású, akkor a szorzó két síknegyedes.

F.10.32. Az *S/H* ún. mintavevő-tartó (SAMPLE TRACK/HOLD) áramköröket mintavevő (követő)-tartó áramköröknek is nevezik, és a 12–170. blokksémával ábrázolhatók.
Az ábrán U_{be} a jelbemenet, U_{vez} a vezérlőfeszültség, U_{ki} pedig a jelkimenet. Működésükre jellemző a mintavételi-tartási periódusok időbeli váltakozása, ahol a periódusok eltérő hosszúságúak is lehetnek (12–171. ábra).

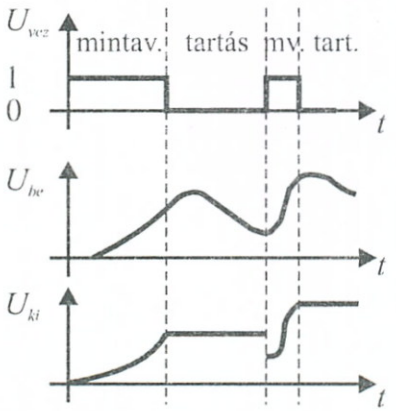


12–170. ábra *Mintavevő-tartó áramkör*



12–171. ábra

A mintavétel és -tartás problémáit már a 10.10. pontban a 10-113. ábrával kapcsolatosan érintettük, ott szerepelt egy nulladrendű tartókapcsolás is. A jelalakokat egy nulladrendű tartó esetére a 12–172. ábra mutatja.



12-172. ábra *Mintavételező-tartó működése*

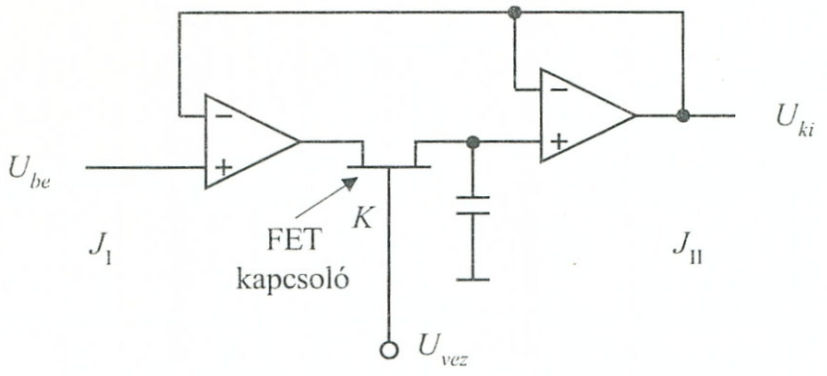
Az analóg jel az U_{be} bemenetre érkezik, az U_{vez} vezérlőjel egy kapcsolójel, mely $U_{vez} = 1$ esetén mintát vesz és ez alatt az idő alatt a berendezés követi az analóg jelet. $U_{vez} = 0$-ára kapcsolásakor az utolsó jelérték tárolódik, a berendezés *tartó* állapotba áll be.

A *nulladrendű* tartó egy nagyon egyszerű elvi áramköri változata a 10-113b. ábrán látható, ott

$$U_{be} = J_I$$

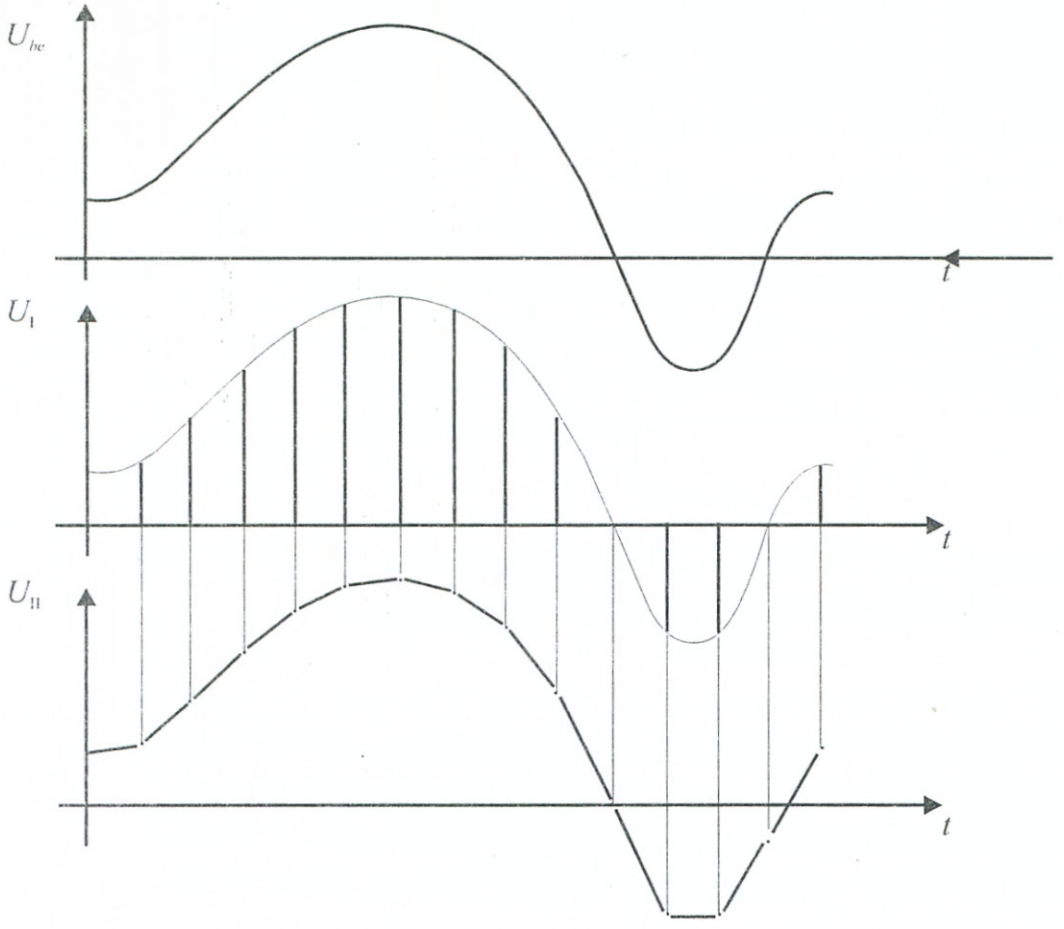
$$U_{vez} = K$$

$$U_{ki} = J_{II}$$

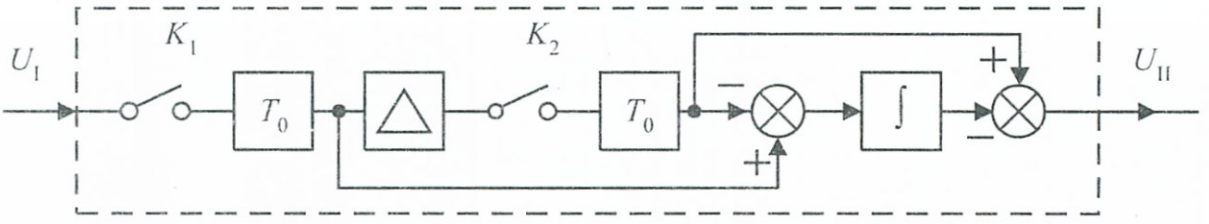


12–173. ábra *Egy alkalmazott S/H kapcsolás*

a)



b)



12–174. ábra *Elsőrendű interpoláló tartó jelei és blokkvázlata*

Egy gyakorlatban használatos megoldást rajzoltunk fel a 12–173. ábrán.

Az áramkör visszacsatolás révén biztosítja azt, hogy mintavételezéskor C töltése vagy kisütése kis impedanciáról történjék.

Az *elsőrendű* tartóknál a mintavételezések közötti tartások nem vízszintes görbeszakaszok, hanem interpolálással vagy extrapolálással előállított ferde-törtvonalas közelítések. Egy elsőrendű interpoláló tartó blokkvázlata látható a 12–174b. ábrán, a jelviszonyokat a 12–174a. görbék mutatják. A blokksémán látható K_1, K_2 kapcsolók mintavételkor „nagyon gyorsan" záródnak és feltöltik a T_0 tartókban levő kondenzátorokat a kapcsoló feszültségére. A Δ blokk időkésleltető áramkör. A kapcsolás ezeken kívül még tartalmaz kivonó és integráló tagokat is, az interpoláció végrehajtása érdekében.

12.11.1. Gyakorló feladatok a 11. „Szoftver és hardver vegyes alkalmazásán alapuló digitális rendszerek" c. fejezethez

*G.11.1. Kövessük végig a NOP-utasítás végrehajtásának mozzanatait a 12-1. ábra hardverjén.

*G.11.2. Kövessük végig a CALL CIMS utasítás végrehajtásának mozzanatait a 12-1. ábra hardverjén.

*G.11.3. Mi lesz az *AC-ben* az alábbi programrészlet végrehajtásának végén?

```
    ⋮
XRA   A
ORI   05H
MVI   D,11
DCR   D
ADD   D
HLT
```

*G.11.4. Mi lesz a *C regiszterben* az alábbi programrészlet HLT utasítása idején?

```
    ⋮
STC
MVI   A, A4H
RAR
ANI   FFH
MOV   C, A
HLT
```

*G.11.5. A MEM memória *adatmezejében* egymás mellett van egy PERC és egy ÓRA-rekesz. Írjuk meg azt az IDŐMÉRŐ szubrutint, amelyet pontosan minden percben kívülről meghívnak és amely percenként felfrissíti a PERC és az ÓRA-rekeszekben a pontos időt.

G.11.6. Terjesszük ki a G.11.5. feladatot. MÁSODPERC, PERC, ÓRA, NAP, HÓNAP, ÉV adatokkal dolgozó időmérővé.

*G.11.7. Adva a MEM memória adatmezejében egy CIMK adott kezdőcímű adatcsoport, mely L = 100 db szót tartalmaz. Írjunk assembly szubrutint, amely az L = 100 szóban levő számadatok közül kikeresi a *legnagyobbat* és a *legkisebbet,* majd a *legnagyobbat* a 12-28. ábrabeli CPU-nak B regiszterében, a *legkisebbet* pedig a C regiszterben tárolja.

*G.11.8. Vázoljuk fel, hogy miként alakul a 11.3. Példa hardver berendezése, ha INTERFÉSZ-ként PIO egységet alkalmazunk.

12.11.2. Feladatmegoldások a 11. fejezethez

F.11.1. A NOP utasítást a program „lyukas" helyeinek „töltelékeként" szokták alkalmazni, rendszerint valamilyen későbbi feladat érdekében. A NOP utasításnál, a nevének megfelelően (NOP = No OPeration) érdemleges művelet nem történik, csupán azt kell biztosítani, hogy a program rajta tovább juthasson. Ehhez elegendő, hogy a PC programszámláló inkrementálódjon. Miután ez magában a FETCH-ben megtörténik, így további teendőkre már nincs szükség.

F.11.2. Itt a 11.3.2.1. pontban elmondottakból indulunk ki, melyek szerint a CALL CIMSR egy
– PC STACK-be mentési (PUSH) és egy
– JMP CIMSR ugrási műveletből áll.
A részletezett lépéseket – a korábbi esetekhez hasonlóan – táblázatosan foglaltuk össze a 12-175. ábrán.

CIKLUS	ELEMI HARDVER MŰVELET	MAGYARÁZAT
1. FETCH	FETCH	– Mindig egyforma
2.	$SP \leftarrow SP-1$	– PC tárolási helyének kijelölése a STACK-ben
	$CR \leftarrow SP$	– STACK megcímzése
	$PC \leftarrow PC+1$	– A visszatérési PC beállítása
	$DR \leftarrow PC$	– PC tárolásának előkészítése
PUSH	$(CR) \leftarrow DR$	– PC tárolása a STACK-ban
	$PC \leftarrow PC-1$	– PC visszaállítása CIM SR rekeszéhez
3.	$CR \leftarrow PC$	– CIM SR rekeszének megcímzése
	$DR \leftarrow (CR)$	– CIM SR értékének kiolvasása
UGRÁS SZUBRUTIN-RA	$PC \leftarrow DR$	– CIM SR áttöltése PC-be

12–175. ábra *CALL CIMSR utasítás végrehajtásának mozzanatai a 12-1. ábra HW-énél*

F.11.3. A 12-31a. ábra utasításkészletének alapján felírhatók a következők (12–176. ábra).

ASSEMBLY	HATÁS	BINÁRIS KÓD		
XRA A	A ← A⊕A	A	0000	0000
ORI Ø5 H	A ← A or Ø5H	A	0000	0000
		Ø5H	0000	0101
		A ←	0000	0101
MVI D, 11H	D ← 11H	D	0001	0001
DCR D	D ← D – 1	D	0001	0001
		–1	– 0000	0001
		D ←	0001	0000
ADD D	A ← A + D	A	0000	0101
		D	+ 0001	0000
		A ←	0001	0101
HLT				

Végül AC tartalma: $0\,0\,0\,1\,0\,1\,0\,1_2 = 15H$

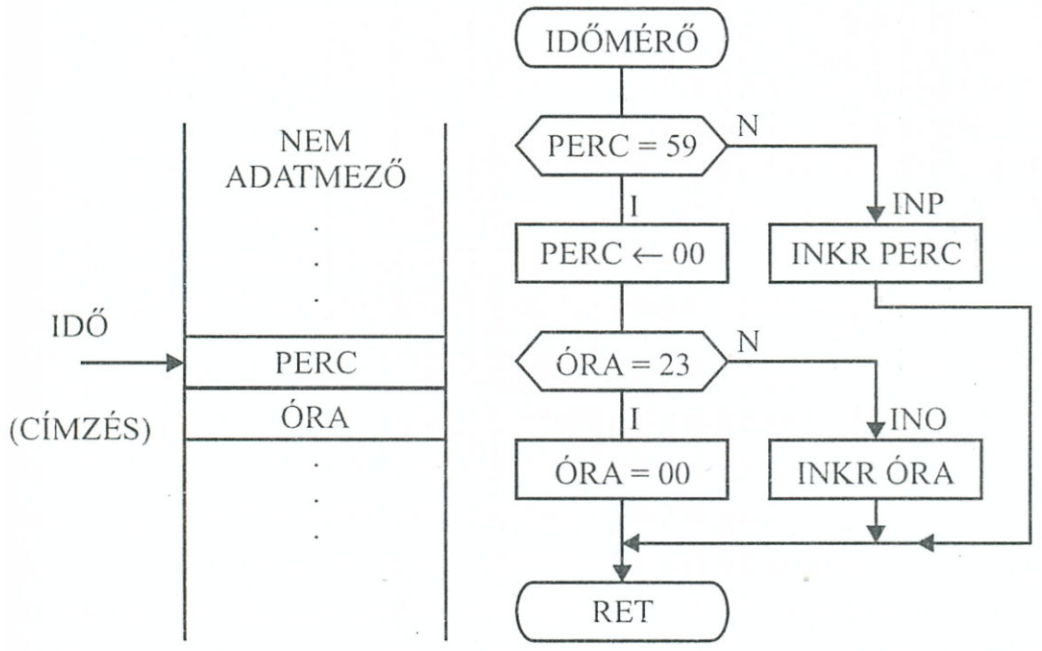
12–176. ábra *Az F.11.3. példa-megoldás*

F.11.4. Hasonlóan az előző példához, felírható a 12–177. ábra

ASSEMBLY	HATÁS	BINÁRIS KÓD		
STC	CY ← 1	CY = 1		
MVI A,A4H	A ← A4H	A	1010	0100
RAR	CY → A	CY $\xrightarrow{1}$ A	1101	0010
		0 ◄		
ANI FFH	A ← A and FFH	A	1101	0010
		FF	1111	1111
		A ←	1101	0010
MOV C, A	C ← A	C ←	1101	0010

12–177. ábra *Az F.11.4. példa-megoldás*

F.11.5. A PERC és ÓRA-rekeszek helyét a memóriában a 12–178a. ábra mutatja az IDŐ nevű címmel. A folyamatábra a 12–178b. ábrán látható, míg az assembly programot a 12–178c. ábrában írtuk meg.

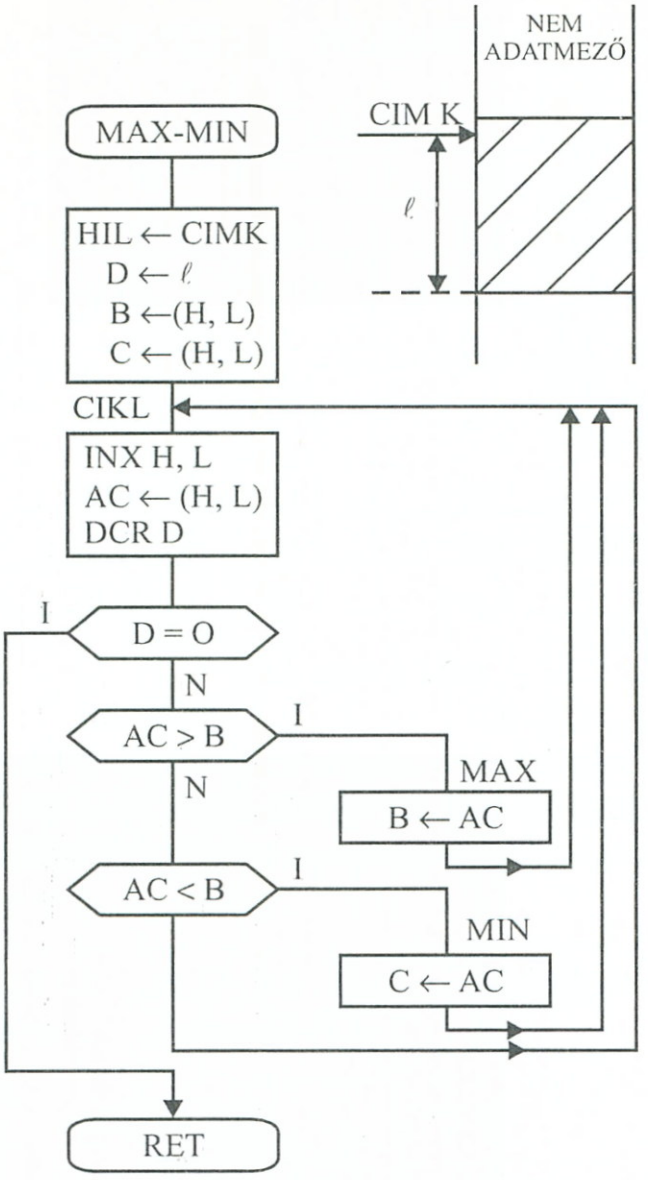



```
IDŐMÉRŐ  LXI   H, IDŐ
         MOV   A, M
         CPI   59
         JNZ   INP
         MVI   M, ΦΦH
         INX   H
         MOV   A, M
         CPI   23
         JNZ   INO
         MVI   M, ΦΦH
         RET
INO      INR   M
         RET
INP      INR   M
         RET
```

a) b) c)

12-178. ábra *IDŐMÉRŐ program az F.11.5. példamegoldáshoz*

F.11.7. A részletezett folyamatábrát a 12–179a. ábra mutatja. Az assembly programot a 12–179b. ábrán írtuk meg.



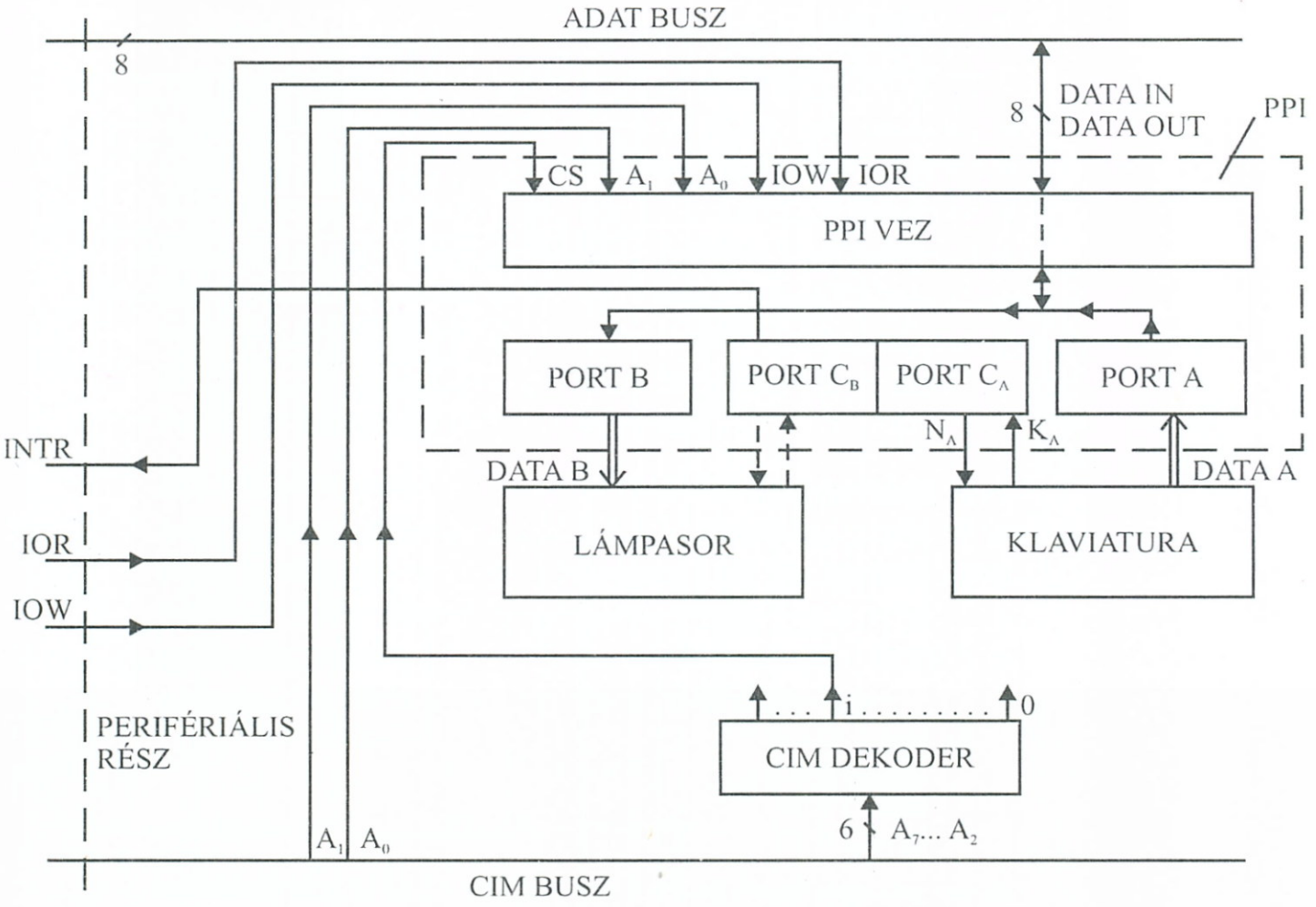
a)

```
MAX-MIN  LXI   H, CIMK
         MVI   D, ℓ
         MOV   B, M
         MOV   C, M
CIKL     INX   H
         MOV   A, M
         DCR   D
         RZ
         CMP   B
         JP    MAX
         CMP   C
         JM    MIN
         JMP   CIKL
MAX      MOV   B, A
         JMP   CIKL
MIN      MOV   C, A
         JMP   CIKL
```

b)

12-179. ábra *Folyamatábra és assembly program az F.11.7. feladatmegoldáshoz*

F.11.8. A PIO egységeket, a gyártó cégek (INTEL, MOTOROLA stb.) többféle változatban állítják elő. A működési leírásokat, paramétereket, a felhasználási mintapéldákat *katalógusokban* bocsátják a felhasználók rendelkezésére. Ezek részletes ismerete nélkül a 11.3. Példa PIO-s változatát csak „körvonalazottan" tudjuk megoldani. A PIO-egység általános vázlatával kapcsolatosan már említettük, hogy egy PIO egységben általában több PORTi-t is beépítenek. Induljunk ki egy, az INTEL cég által forgalmazott PIO egységből (ezt ott PPI – Parallel-Peripherial-Interface-nek is nevezik), mely 3 db PA, PB, PC port-ot tartalmaz. A 12–180. ábrán felraj-



12-180. ábra *KLAVIATURA-LÁMPASOR illesztése PPI-vel a 11-36. ábra rendszeréhez*

zoltunk egy olyan blokkvázlatot, amely egy ilyen PPI-t használ fel, és amely blokkvázlattal a 12–36. ábra szaggatott vonaltól jobbra eső – „perifériális rész" helyettesíthető – a feladatmegoldásnak megfelelően.

A PPI-ben található PORTi-ket *külön-külön* meg lehet címezni. Egy gyakran alkalmazott megoldásként válasszuk a PORT-címzésre a címbusz legalsó (A1A0) bitjeit, melyekkel pl. a 12–181a. ábra táblázata szerint címezhetjük meg PA, PB, PC-t és lehetőségünk van még a PIO VEZ blokk külön megcímzésére is. Ez utóbbi azért fontos, mert a működtetés érdekében a PPI-t többféle *működési mód*ba kell majd beprogramoznunk ún. CONTROL WORD (CW) *vezérlőszavak* felhasználásával. A vezérlőszavak (hasonlóan, mint a SIO-nál már láttuk) az *adatbuszon* érkeznek a CPU felől, 8 bites, párhuzamos formában. Ha a PA, PB, (PC) blokkok vannak megcímezve, akkor az adatbuszon ADAT

PORT címzése:

A_1 A_0	PORT és VEZ
0 0	PA
0 1	PB
1 0	PC
1 1	PPI VEZ

a)

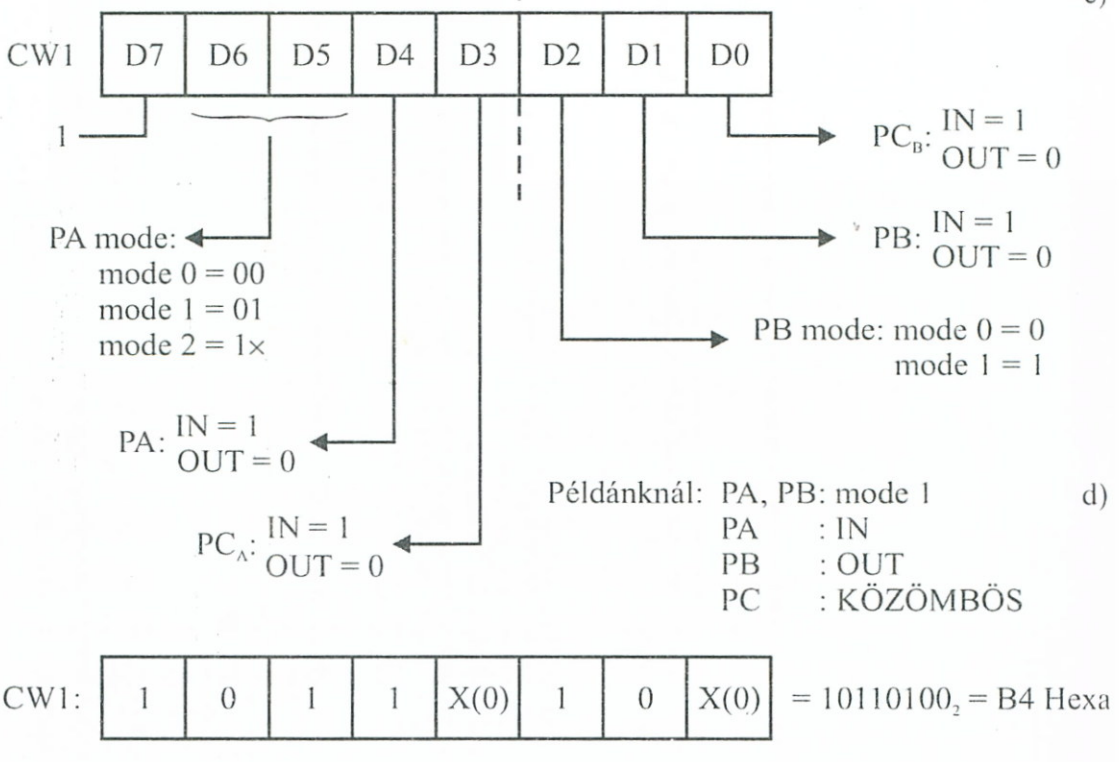
mode változatok:

mode	JELLEMZŐK
mode 0	PA, PB, PC független PORT-ok
mode 1	PA, PB, független PORT-ok PC vezérlő feladatokat lát el hand shaking vezérlés
mode 2	PA, PB, független PORT-ok PA kétirányú adatátvitelre alkalmas PB egyirányúra PC vezérlő feladatokat lát el hand shaking vezérlés

b)

CONTROL WORD:

c)



Példánknál: PA, PB: mode 1 d)
PA : IN
PB : OUT
PC : KÖZÖMBÖS

CW1:	1	0	1	1	X(0)	1	0	X(0)

$= 10110100_2 = B4$ Hexa

CIMZÉSI VISZONYOK:

A_7 A_6 A_5 A_4 A_3 A_2	A_1 A_0	CIM $PORT_i$	CIM HEXÁBAN
	0 0	PORT A	4 C
0 1 0 0 0 0	0 1	PORT B	4 D
(például felvéve)	1 0	PORT C	4 E
	1 1	PPI VEZ	4 F
CS			

e)

12-181. ábra *PPI működési paraméterek beállítása*

(DATA) információ áramlik, ha a PIO VEZ-t címezzük, akkor az adatbuszon VEZÉRLŐ SZÓ érkezik a CPU felől.
A *vezérlő szó* tartalmazza az egyes PORT-ok adatátviteli irányítottságát (IN, OUT), továbbá a PPI működési módját. A lehetséges működési módokat a 12–181 b. ábrán soroltuk fel. Egy vezérlő szó bitenkénti felépítését tanulmányozhatjuk a 12–181c. ábrán. A vezérlő szót – adott felhasználási helyzetnek megfelelően – bitenként *ki kell tölteni,* és az így keletkező bináris számot HEXA formában már, mint egy *átvitelre jellemző konstanst* tekintjük.

Jelen feladatunknál – az átvitel megbízhatóságának javítása érdekében – válasszuk a 11.4.3.1. pontban már említett *hand shaking* üzemmódot. Miután a 12–180. ábrából láthatóan mind PA, mind PB egyirányú adatátvitelt kell hogy végezzenek, ezért a PPI-t a 12–181b. ábra alapján: mode 1-be kell majd beállítanunk. Az adatátviteli irányoknál:

PA IN (befelé)

PB OUT (kifelé)

értelemben kell majd programoznunk. A hand shaking üzemmódban a PC port csatlakozó vezetékei a 11.4.3.1. pontban ugyancsak említett:

K – „adatot fogok küldeni" (kezdő)

N – „adat megjött" (nyugtázó)

jelek feladatait látják el, mégpedig úgy, hogy a PC_H (felső négy bit) a PA-t, a PC_L (alsó négy bit) a PB-t szolgálja ki az átvitel során. (PC szabadon maradt bitjeit – egyes PPI-típusoknál még ki lehet pl. adatátvitelre használni.)

A körvonalazott működési feltételek alapján már kitölthető a 12–180 ábrában szereplő PPI-re az aktuális CW *control word,* melyet a 12–181d. ábrán mutattunk be részleteiben.
A PPI-k egyes típusait alkalmassá tették INTERRUPT üzemmódra is, ezt a 12–180. ábrán ugyancsak feltüntettük.
A PPI CONTROL WORD valójában lehet:

– COMMAND WORD (Parancs típusú), melynél a CPU parancsokat küld a PPI-nek;

– STATUS WORD (Lekérdező típusú), melynél a CPU lekérdezi a PPI-t az egyes kimeneti csatlakozások jel-állapotaira vonatkozóan.

A 11–36. ábrabeli HW PPI-vel történő módosításának kihatása van a *szoftverre is*. A berendezés áram alá helyezése után, itt is be kell írni a PPI VEZ-be a CONTROL WORD-öt, azért, hogy a PPI a 12–180. ábra szerinti működésre legyen *felprogramozva*. Ez az *inicializálás*, amely a HW építését „helyettesíti". Az inicializáló program:

```
⋮
MVI     A, B4 H     AC ← CW 1      } lásd:
OUT     4F          PPI VEZ ← AC   } 12–181e. ábra
⋮
```

Ezután a PPI működésre kész.
Valamelyik gomb megnyomása után, itt is IT-vel indulhat a működés, majd ráugrunk az IT szubrutinra. A PPI-s változatnál az ITSR-ben

CIM KL helyébe: CIM PORT A (4CH)
CIM LP helyébe: CIM PORT B (4DH)

kerülnek.

Az INTERFÉSZ egységek részletesebb tárgyalásával az alkalmazás orientáltabb irodalmi források és a MIKROSZÁMÍTÓGÉP-es szakirodalom foglalkoznak.

NÉHÁNY TÁJÉKOZTATÓ IRODALMI FORRÁS

Dr. Simonyi Károly: Elméleti villamosságtan. Tankönyvkiadó, 1960.

Dr. Fodor György: A Laplace transzformáció műszaki alkalmazása. Műszaki Könyvkiadó, 1962.

J. Millman–A. Grabel: Microelectronics. Mc Graw-Hill, 1988.

Hollós–Dr. Vágó: Villamosságtan I–III. LSI Kiadás, 1990.

Dr. Komarik József: Analóg elektronika. LSI Kiadás, 1990.

Dr. Szittya Ottó: Logikai kapcsolástan. Tankönyvkiadó, 1971.

Zvi Kohavi: Switching and Finite Automata Theory. Mc Graw–Hill, 1980.

Dr. Szittya Ottó: Logikai rendszerek és szekvenciális automaták. Tankönyvkiadó, 1986.

Dr. Flesch István: Logikai rendszerek tervezése. Példatár. Tankönyvkiadó, 1987.

Dr. Arató Péter: Logikai rendszerek tervezése. Tankönyvkiadó, 1988.

U. Tietze–Ch. Schenk: Analóg és digitális áramkörök. Műszaki Könyvkiadó, 1990.

M. A. Harrison: Introduction to the Switching and Automata Theory. Mc Graw–Hill, 1970.

Theisz–Gruber–Jagudits: Digitális technika I. Tankönyvkiadó, 1977.

Bohus–Horváth: Digitális számítógépek. Tankönyvkiadó, 1982.

Dr. Szittya Ottó: Digitális elektronika I–V. kötet. LSI Kiadás, 1990.

Gerdai Gábor: Programozható áramkörök. Távközlési Kiadó, 1992.

Dr. Szittya Ottó: Bevezetés az elektronikába. LSI Kiadás, 1996.

C. J. Coates: Threshold Logic. John Wiley, 1970.

D-C. Rine: Computer science and multiple-valued logic. North Holland, New York, 1977.

R. H. Katz: Contemporary logic design. The Benjamin Cummings, 1994.

L. A. Zadeh: Fuzzy sets. Information and Control. 1965.

Dr. Bagyinszki János: Fuzzy logic. Kandó Kálmán Polytechnics. 1997.

Cox Earl: The Fuzzy system handbook. AP-Professional, 1994.

Vörös Gábor: Bevezetés a neurális számítástechnikába. LSI Kiadás, 1997.

Dr. Kóczy T. László: Fuzzy Logic I. Tempus Jep 07759-94-Modify, 1997.

Dr. Kóczy–Kovács Sz: Fuzzy Logic II. Tempus Jep 07759-94-Modify, 1997.

Dr. Szittya Ottó: Fuzzy Flip-Flops, Functional Units, Arithmetics Tempus. Budapest–Barcelona, 1997.

John Uffenbeck: Microcomputers and Microprocessors. Prentice Hall Inc., 1991.

TÁRGYMUTATÓ

TARTALOMJEGYZÉK

GÁBOR DÉNES FŐISKOLA

MŰKÖDÉSI HELYEI

Budapest **Informatikai Rendszerek Intézete:**
Vezető: *Dr. Kovács Magda*
1037 Budapest, Bécsi út 324.
Információ
Tel.:06-1-436-65-19; Fax:06-1-436-65-28
Honlap: www.gdf.hu

Informatikai Alkalmazások Intézete:
Vezető: *Dr. Zárda Sarolta*
1115 Budapest, Etele út 68.
Tanulmányi Osztály
Tel.: 06-1-203-02-83; 06-1-203-03-04/8900
E-mail: : tanoszt@gdf.hu Honlap: www.gdf.hu

Ajka Vezető: *Kovács Sándor*
Tel.: 06-88-453-948, 06-30-3902-576;

Baja Vezető: *Búcsú Lajos*
6500 Baja, Oltványi u. 14.
Tel./Fax: 06-79-426-427; E-mail: oktatas@fit.hu
Honlap: www.fit.hu

Balatonboglár Vezető: *Lakos István*
8630 Balatonboglár, Szabadság u. 41.
Tel.: 06-85-351-633; 06-30-2544-995;
Tel./Fax: 06-85-351-316 E-mail: lakist@mathiasz.sulinet.hu
Honlap: www.extra.hu/balatonboglar

Békéscsaba Vezető: *Kovács Zoltánné*
5600 Békéscsaba, Andrássy u. 73/1.
Tel.: 06-66-448-385; 06-30-3552-039;
Fax: 06-66-448-385; E-mail: gdf-bkk@nap-szam.hu
Honlap: www.gdf-bkk-03.nap-szam.hu

Cegléd Vezető: *Sági Ferenc*
2700 Cegléd, Kossuth F.u. 32.
Tel.: 06-20-9575-958;
Tel./Fax: 06-53-311-695; E-mail: sagif@infotars.hu
Honlap: www.extra.hu/gdf-cegled

Debrecen Vezető: *Kovács Jánosné*
4029 Debrecen, Maróthy Gy. u. 5-7.
Tel./Fax: 06-52-418-660; E-mail: gdf@cts.hu
Honlap: www.cts.hu

Dunaújváros **Szervezése Székesfehérváron**
Vezető: *Róth Péter*
8000 Székesfehérvár, Mátyás király krt. 5.
Tel./Fax: 06-22-348-542; E-mail: roker@mail.axelero.hu

Eger Vezető: *Csathó Csaba*
3300 Eger, Széchenyi u. 58.
Tel.: 06-30-9581-822; Tel./Fax: 06-36-411-811;
E-mail: sprinter@mail.agria.hu Honlap: http://sprinter.agria.hu

Esztergom Vezető: *Fűrész Tiborné*
2500 Esztergom, Bánomi út 8.
Tel./Fax: 06-1-230-43-76; 06-20-3518-601;
E-mail: fureszt@axelero.hu Honlap: web.axelero.hu/fureszt

Érd Vezető: *Budai Csaba*
2030 Érd, Széchenyi tér 1.
Tel./Fax: 06-20-9344-770; 06-1-283-79-79;
E-mail: budacsa@axelero.hu

Gyöngyös Vezető: *Gerják István*
3200 Gyöngyös Kőrösi Csoma S. u. 9.
Tel./Fax: 06-37-303-683; 06-30-9585-488
E-mail: gerjak@mail.datanet.hu, trivium@tvnetwork.hu

Győr Vezető: *Domonkos Gyuláné*
9021 Győr, Árpád u. 2.
Tel.: 06-96-550-200; Fax: 06-96-550-201
E-mail: etelka@gdfgyor.axelero.net Honlap: www.gdfgyor.axelero.net

Hódmezővásárhely Vezető: *Dr.Gál József*
6800 Hódmezővásárhely, Szántó K. J. u. 64.
Tel.: 06-62-535-536; 06-30-3135-873;
Fax: 06-62-535-530; E-mail: imsuli@delfin.hu
Honlap: www.gdf.vasarhely.hu

Isaszeg Vezető: *Keresztúri Jusztina*
2117 Isaszeg, Madách u. 1/a
Tel./Fax: 06-28-496-206; E-mail: gdfisa@mail.sziszi.hu

Kaposvár Vezető: *dr.Paál Jenő*
7400 Kaposvár, Guba S. u. 36-40.
Tel.: 06-82-314-155/147; 06-82-424-339;
Fax: 06-82-320-757; E-mail: posane@atk.kaposvar.pate.hu

Kecskemét Vezető: *Pósfayné Bakota Éva*
6000 Kecskemét, Hunyadi János tér 2.
Tel.: 06-76-411-494; 06-20-9762-100; 06-20-9935-735;
Fax: 06-76-411-041; E-mail: gdfkecsk@axelero.hu

Kisvárda Vezető: *Meskó Marianna*
4600 Kisvárda, Szent László út 27.
Tel.: 06-45-500-290; Fax: 06-45-500-291;
E-mail: info@felveteliiroda.hu Honlap:www.felveteliiroda.hu

Keszthely Vezető: *dr Vargáné Dugonics Rita*
8360 Keszthely, Deák F. u. 57.
Tel.: 06-83-312-330/262; 06-83-312-330/290; 06-30-2165-583
Fax: 06-83-314-334; E-mail: vdr@georgikon.hu

Mátészalka Vezető: *Pethő Mária Terézia*
4700 Mátészalka, Seregély u. 17/A.
Tel.: 06-44-417-889; 06-20-9743-125; 06-20-3663-669
Tel./Fax: 06-44-417-889;
E-mail: fic@okito.hu, info@okito.hu Honlap: www.okito.hu

Miskolc Vezető: *Dr Czap László*
3535 Miskolc, Árpád u. 2.
Tel.:06-20-3132-869; Tel./Fax: 06-46-414-429;
E-mail: czap@mazsola.iit.uni-miskolc.hu
Honlap: http://gdfm.homeip.net

Nagykanizsa Vezető: *Németh Zoltán*
8800 Nagykanizsa, Ady Endre u. 74/a
Tel.: 06-93-312-383; 06-93-313-010/730; 06-30-9370-605;
Fax: 06-93-310-107; E-mail: nemethz.gdf@chello.hu

Nyíregyháza Vezető: *Méhész János*
4400 Nyíregyháza, Vasvári Pál u. 1.
Tel.: 06-42-407-027; 06-42-406-850; 06-20-3559-390;
Fax: 06-42-406-848; E-mail: szuvnyirseg@chello.hu
Honlap: www.freeweb.hu/gdf-nyhz

Pápa Vezető: *Gerics Erzsébet*
8500 Pápa, Budai Nagy Antal u. 1.
Tel.: 06-20-9583-058; Tel/Fax: 06-89-311-297;
E-mail: gerics@gege-comp.hu Honlap: www.gege-comp.hu

Pécs Vezető: *Kedves Vera*
7624 Pécs, Tiborc u. 38/c
Tel.: 06-72-213-412; Fax: 06-72-310-259;
E-mail: gdfpecs@axelero.hu

Pilisvörösvár **Szervezése Esztergomban**
Vezető: *Fűrész Tiborné*
2500 Esztergom, Bánomi út. 8.
Tel./Fax: 06-1-230-43-76; 06-20-3518-601;
E-mail: fureszt@axelero.hu Honlap: web.axelero.hu/fureszt

Salgótarján Vezető: *dr. Agócs József*
3100 Salgótarján, Kossuth u. 8.
Tel.: 06-32-416-833; Fax: 06-32-317-420;
E-mail: jozsef.agocs@ncsszi.hu

Sátoraljaújhely **Szervezése Mátészalkán**
Vezető: *Pethő Mária Terézia*
4700 Mátészalka, Seregély u. 17/A.
Tel.: 06-20-9743-125; 06-20-3663-669;
Tel./Fax: 06-44-417-889; E-mail: info@okito.hu

Siófok **Szervezése Székesfehérváron**
Vezető: *Róth Péter*
8000 Székesfehérvár, Mátyás király krt. 5.
Tel./Fax: 06-22-348-542; E-mail: roker@mail.axelero.hu

Sopron Vezető: *Dr. Molnár László*
9400 Sopron, Bajcsy-Zsilinszky u. 4.
Tel.:06-99-518-182; 06-30-4112-941;
Fax: 06-99-518-259; E-mail: molnarl@fmk.nyme.hu

Szeged Vezető: *Malincsák János*
6720 Szeged, Kígyó u. 4.
Tel: 06-20-9245-756; Tel./Fax: 06-62-423-258,
E-mail: malincsa@tvnetwork.hu, gdfszeged@tvnetwork.hu
Honlap: www.gdfszeged.hu

Szekszárd Vezető: *Doszkocs László*
7100 Szekszárd, Rákóczi u. 70.
Tel./Fax: 06-74-413-435; E-mail: gdf@kvantum.hu
Honlap: http://gdf.kvantum.hu/

Székesfehérvár Vezető: *Róth Péter*
8000 Székesfehérvár, Mátyás király krt. 5.
Tel./Fax: 06-22-348-542; E-mail: roker@mail.axelero.hu

Szolnok Vezető: *Nyáry László*
5000 Szolnok, Arany J. u. 4.
Tel.: 06-56-375-122; Fax: 06-56-511-262;
E-mail: titgdf@externet.hu

Szombathely Nyílt nap helye:
POTE Szhelyi Képzési Kp. Jókai Mór u. 14.
Vezető: *Csicseri Andrea*
9700 Szombathely, Kisfaludy u. 51.
Tel.: 06-94-501-894; 06-20-9910-357; Fax: 06-94-501-899;
E-mail: gdf_okt@flagnet.hu Honlap:www.flagnet.hu

Tatabánya Vezető: *Gimesi Lászlóné*
2800 Tatabánya, Vértanuk tere 13.
Tel./Fax: 06-34-510-460; 06-20-9625-047;
E-mail: gnkft@axelero.hu

Vác Vezető: *Dr. Molnár Lajos*
2601 Vác, Németh László u.4-6.
Tel.: 06-27-317-077; 06-27-317-886; Fax: 06-27-315-093;
E-mail: boronkay@vac.hu

Veszprém Vezető: *Nagyné László Mária*
8201 Veszprém, Egyetem u. 10.
Tel.: 06-88-422-022/4191; 06-20-5636-777;
Fax: 06-88-422-022/4452; E-mail: laszlo@almos.vein.hu

Zalaegerszeg Vezető: *Dr. Sárváryné Kiss Katalin*
8900 Zalaegerszeg, Rákóczi u. 4-8.
Tel.: 06-92-311-229; 06-20-9225-546; Fax: 06-92-510-590;
E-mail: zeggdf@szam.hu Honlap: www.szam.hu

Határon túli központok:

Románia

Erdélyi központok szervezése Kolozsváron (Cluj-Napoca)
Vezető: *Dr. Selinger Sándor*
România,3400 Cluj-Napoca str.: I. Budai Deleanu 64.
Levélcím: OFP 5. CP 737
Tel./Fax: 00-40/2-64-431-841; E-mail: gdf@gdf.ro
Honlap: www.gdf.ro

Csíkszereda (Miercurea Ciuc)
Kolozsvár (Cluj-Napoca)
Marosvásárhely (Târgu-Mureş)
Nagyvárad (Oradea)
Sepsiszentgyörgy (Sfântu-Gheorghe)
Szatmárnémeti (Satu-Mare)
Székelyudvarhely (Odorheiu-Secuiesc)

Szerbia-Montenegro

Szabadka (Subotica) Vezető: *Bóni László*
24000 Szabadka, Szegedi út 74. Szerbia-Montenegro
Tel.:00-381-24-546-067; 00-381-24-546-463;
Fax: 00-381-24-546-021; E-mail: imc@magnotron.co.yu

Szlovákia

Kassa (Košice) Vezető: *Dr. Kiss Imre*
04001 Košice, Moyzesova 58., Slovenská Republika,
Tel.: 00-421-55-727-4435, Fax: 00-421-55-727-4429;
E-mail: kiss@teledom.sk Honlap: www.teledom.sk

Diószeg Sládkovičovo Vezető: *Ladislav Spalek*
92521 Sládkovičovo, Fučikova ul. 269.
Slovenská Republika
Tel.: 00-421-31-788-1733; 00-421-31-788-1712;
Fax: 00-421-31-788-1710;
E-mail: ladislav.spalek@ivosr.sk, takacsatti@hotmail.com