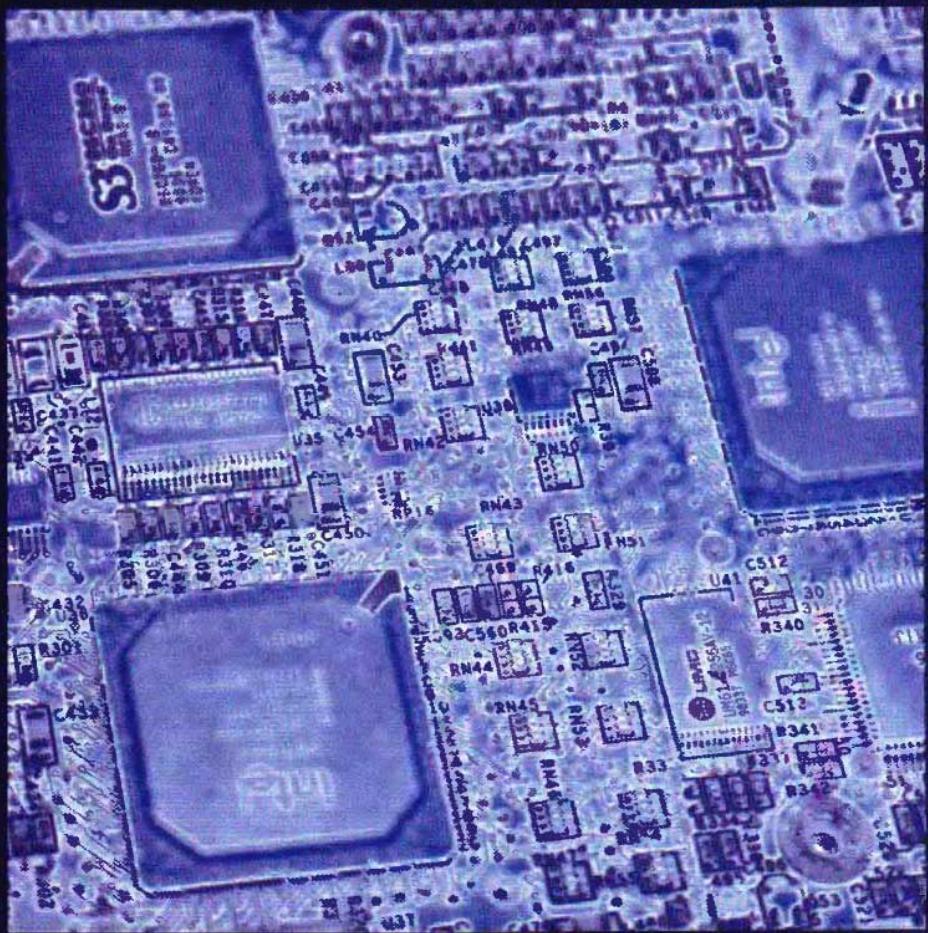


Zombori Béla

# Digitális elektronika



VILLAMOS SZAKMÁK



Nemzeti Tankönyvkiadó

TANKÖNYVMESTER KIADÓ

## Tartalomjegyzék

Előszó .....	5
1. DIGITÁLIS TECHNIKA ALAPJAI .....	7
1.1. A digitális jelek fogalma és jellemzői .....	7
1.2. Számrendszerök .....	8
1.3. Kódolás .....	11
2. LOGIKAI ALGEBRA .....	19
2.1. A logikai algebra alapvető összefüggései .....	19
2.2. Logikai függvények grafikus egyszerűsítése .....	24
3. A LOGIKAI HÁLÓZATOK ALAPELEMEI .....	36
3.1. Kapuáramkörök .....	37
4. TÁROLÓK .....	46
5. LOGIKAI HÁLÓZATOK ANALÍZISE ÉS REALIZÁLÁSA .....	54
5.1. Kombinációs hálózatok .....	54
5.2. Sorrendi hálózatok .....	67
6. FUNKCIIONÁLIS ÁRAMKÖRÖK .....	78
6.1. Multiplexerek .....	78
6.2. Demultiplexerek, dekódolók .....	81
6.3. Aritmetikai áramkörök .....	84
6.4. Regiszterek .....	91
6.5. Számlálók .....	96
6.6. Gyűrűs számlálók .....	109
7. MIKROPROCESSZOROK .....	113
7.1. Pipe-line elv .....	114
7.2. Lebegőpontos aritmetika, szuper-skalár felépítés .....	116
7.3. Cache alkalmazása .....	117
7.4. Virtuális tárkezelés .....	118
7.5. Multitask rendszer .....	118

## Előszó

Ez a **Digitális elektronika** c. könyv, amit kezében tart a kedves Olvasó, a Tankönyvmester Kiadó villamos ipari és rokon szakmák számára készített tankönyvcsaládjának egyik alapozó tankönyve, és a szakképzésben résztvevő tanulók számára foglalja össze a digitális áramkörök elméleti alapjaival, tervciszével, a fontosabb áramkörcsaládokkal és jellegzetes digitális áramkörökkel, valamint a modern mikroprocesszorok néhány szervezési elvével kapcsolatos legfontosabb ismereteket.

A könyv **Az elektronika** c. tankönyv szerves folytatása, az önálló kötetben való megjelentetést főleg az indokolja, hogy sok iskolában párhuzamosan tanítják az Eléktrotechnika és Digitális elektronika c. tantárgyakat. A könyv a rövid informatikai bevezető (számrendszerök, kódolás) és a logikai algebra alapjainak összefoglalása után ismerteti a TTL és CMOS áramkörcsaládok jellemzőit, alkalmazási területeit, a kombinációs és sorrendi hálózatok analizisének és realizálásának módszereit, a tárolóelemeket, valamint a digitális rendszerek felépítésénél alapvető fontosságú funkcionális áramköröket (multiplexerek, demultiplexerek, dekódolók, aritmetikai áramkörök, regiszterek, számlálók).

A könyv legnagyobb előnye, hogy a digitális technikát rendkívül tömören, sok ábrával illusztrálva, rendszertechnikai szemlélet alapján foglalja össze, és az elméleti ismeretek esetében szükséges alapfogalmak magyarázatára szorítkoznak. A könyv jól előkészíti a későbbi tanulmányok szakmaspecifikus tananyagát.

A könyv elméleti ismereteit jól kiegészítik még a

**TM–11002 Elektrotechnika,**

**TM–11004 Elektronika,**

című tankönyvek, a gyakorlati munkát jól segítik a

**TM–11008 A villamos mérések alapjai,**

**TM–11209 Villamos mériși feladatok,**

**TM–11009 A villamos gyakorlatok alapjai**

című tankönyvek és az áramkörök gyakorlati megvalósításához nyújt segítséget a

**TM–11005 Villamos anyagismeret és technológia**

című könyv.

Eredményes tanulást és szakmai sikereket kíván minden kedves Olvasójának a

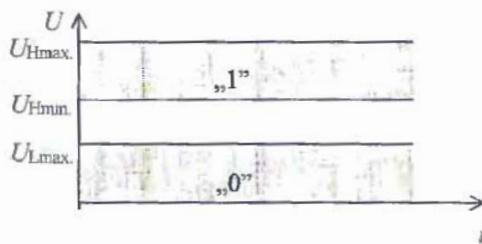
# 1. A DIGITÁLIS TECHNIKA ALAPJAI

A következő fejezetek olyan információkat tartalmaznak, amelyek megalapozzák a digitális áramkörök felhasználását, a számítógép-technikában, a vezérlés- és szabályozástechnikában, a műszertechnikában, híradástechnikában és az elektronika egyéb ágaiban.

## 1.1. A digitális jelek fogalma és jellemzői

Az Elektronika tankönyvben megismert alkatrészek és áramkörök közös jellemzője, hogy analóg jeleket dolgoztak fel. **Az analóg jelek folyamatosan változó jelek, változási tartományukban tetszőleges értéket vehetnek fel.** A következőkben olyan alkatrészeket és áramköröket ismerünk meg, amelyek digitális jeleket dolgoznak fel. **A digitális jelek csak meghatározott értéket vehetnek fel, az egyik értékükönél a másikra ugrásszerűen változnak meg.** Fizikailag ezek az értékek legtöbbször feszültségszintek.

A digitális alkatrészek és áramkörök két feszültségszintű digitális jelet dolgoznak fel. A két feszültségszint egy lehetséges elrendezését szemlélteti a 1.1. ábra.



1.1. ábra. A digitális jel feszültségszintjei

Az egyik feszültségszint 0 V közelébe esik, kisebb, mint egy előre meghatározott  $U_{L\max}$  feszültség. A jelölésben szereplő  $L$  betű az angol *Low* – alacsony szó rövidítése. Az ebbe a tartományba eső feszültségszint a digitális jel  **$U_L$  alacsony logikai szintje**. A másik feszültségszint meghatározott  $U_{H\min}$  és  $U_{H\max}$  feszültségek közé esik. A  $H$  betű a *High* – magas rövidítése. Az ebbe a tartományba eső feszültség a digitális jel  **$U_H$  magas logikai szintje**.

Az 1.1. ábra feszültségelrendezése a **pozitív feszültségrendszer-pozitív logika**. Pozitív a feszültségrendszer, mert az  $U_L$  és az  $U_H$  szint is a pozitív feszültségek tartományába esik, és pozitív a logika, mert a kisebb feszültséghez tartozik az alacsony logikai szint, a nagyobb feszültséghez a magas logikai szint.

## 1.2. Számrendszer

A digitális jelek matematikai leírására a kettes számrendszer a legalkalmasabb, hiszen a két feszültségszinthez egyértelműen hozzárendelhető a **kettes számrendszer két számértéke, a 0 és az 1**, amint azt az 1.1. ábra is mutatja. Ezért a számrendszer gyakran bináris (kétértékű) számrendszernek nevezzük.

A bináris számrendszer **helyiértékes**, tehát a számok nagyságrendje attól függ, hogy melyik helyiértéken helyezkednek el. Pl. az 1011 bináris szám által hordozott érték:

$$1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0.$$

Az egyes helyiértékek nagyságrendjét kettő hatványai határozzák meg. A legkisebb helyiértéktől indulva a helyiértékek:

$$2^0 = 1; 2^1 = 2; 2^2 = 4; 2^3 = 8; 2^4 = 16; 2^5 = 32; 2^6 = 64 \text{ stb.}$$

A bináris számok egy helyiértékén szereplő számjegyet bináris digitnek, vagy röviden **bitnek** nevezzük.

A digitális jelek leírására használatos kettes számrendszer és az általunk egyéb téren megszokott tízes számrendszer között az átalakítást egy-egy példán keresztül mutatjuk be.

Kettes számrendszerből tízesbe való áttérésnél az egyes bináris helyiértékeket keletkező számértékeket összegezve a tízes számrendszerbeli értéket kapjuk. Pl. egy négy helyiértékes bináris szám esetén:

$$1011 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 11,$$

a szokásos jelölésekkel:  $1011_2 = 11_{10}$ .

Kettes számrendszerbeli törtek átalakításánál is hasonló módszert alkalmazunk. Pl.

$$0.01101_2 = 0 \cdot \frac{1}{2^1} + 1 \cdot \frac{1}{2^2} + 1 \cdot \frac{1}{2^3} + 0 \cdot \frac{1}{2^4} + 1 \cdot \frac{1}{2^5},$$

A kettes számrendszerbeli pontot kettedes (bináris) pontnak nevezzük.

A tízes számrendszerből kettesbe való áttérésnél az átalakítandó számot kettő hatványaiból kell összerakni. Az erre használt módszer szerint a kettővel való

osztásnál keletkezett egész hányadot és a maradékot – amely csak 0, vagy 1 lehet – feljegyezzük, majd a hányadost újra osztjuk kettővel, a maradékot feljegyezzük stb. Példaként oldjuk meg a következő feladatot!

$$183_{10} = \dots \dots \dots_2$$

$$\begin{array}{r}
 \text{maradék} \\
 \downarrow \\
 183 \mid 1 \\
 \text{hányados} \Rightarrow \quad 91 \mid 1 \\
 \quad 45 \mid 1 \\
 \quad 22 \mid 0 \\
 \quad 11 \mid 1 \quad \uparrow \text{ összeolvasás irányába} \\
 \quad \quad 5 \mid 1 \\
 \quad \quad 2 \mid 0 \\
 \quad \quad 1 \mid 1 \\
 \quad \quad 0 \mid
 \end{array}$$

A feljegyzett maradékokat alulról felfelé összeolvasva megkapjuk a decimális szám bináris megfelelőjét:

$$183_{10} = 10110111_2.$$

A tizedes törtek átalakításánál, a törtet kettővel szorozzuk, a szorzat egész részét feljegyezzük, majd a tört részt újra szorozzuk kettővel stb. Pl.

$$0,36_{10} = \dots \dots \dots_2.$$

$$\begin{array}{r}
 \text{a szorzat egész része} \\
 \downarrow \\
 0,36 \\
 \text{a szorzat törtrésze} \Rightarrow \quad 0,72 \mid 0 \\
 \quad 0,44 \mid 1 \\
 \quad 0,88 \mid 0 \quad \downarrow \text{az összeolvasás irányába} \\
 \quad 0,76 \mid 1 \\
 \quad 0,52 \mid 1 \\
 \quad 0,04 \mid 1 \\
 \quad \text{stb.}
 \end{array}$$

$$0,36_{10} = 0,010111\dots_2.$$

A digitális jelek leírására és az ezeket feldolgozó áramkörök működésének jellemzésére a bináris számrendszer igen jól használható. Kiolvasni és kimondani egy sokbites számit azonban igen nehézkes, ezért erre a célra a tömörebb leírást és jobb kimondhatóságot biztosító tizenhatos, vagy másképpen hexadecimális számrendszer használjuk.

**A hexadecimális számrendszerben** a mennyiségek leírására 16 számjegyet használunk:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Ebben a számsorban a betűk is számértéket jelentenek: A = 10; B = 11; C = 12; D = 13; E = 14; F = 15. Bevezetésükre azért volt szükség, mert egy helyiértéken nem állhat két számjegy. Ilyen jelölésrendszert használva egy hexadecimális szám pl. a 39C4F.

A tizenhatos számrendszer is helyiértékes, ezért igaz, hogy  $39C4F = 3 \cdot 16^4 + 9 \cdot 16^3 + C \cdot 16^2 + 4 \cdot 16^1 + F \cdot 16^0$ . Összegzve az egyes helyiértékek számértékeit, kiszámíthatjuk a hexadecimális szám tízes számrendszerbeli megfelelőjét:

$$39C4F_{16} = 3 \cdot 65536 + 9 \cdot 4096 + 12 \cdot 256 + 4 \cdot 16 + 15 \cdot 1 = 236623_{10}$$

Tízes számrendszerből a tizenhatosba való átszámitás a bináris számrendszerrel már megismert módszer szerint lehetséges, de az osztószám 16. Az átalakítás módja jól követhető a következő példán.

48526| E  
3032| 8  
189| D  
11| B  
0|

Tehát:  $48526_{10} = BD8E_{16}$

Szokásos, az indexbe írt 16 helyett a hexadecimális mennyiségek jelölésére a H betűt használni: BD8E H (vagy BD8EH formában).

A kettes és a tizenhatos számrendszer közötti átalakítás egyszerűen elvégezhető, mert a két számrendszer alapszámai közötti összefüggés:  $16 = 2^4$ . A 4-es kitevő miatt a bináris számot 4 bites csoportokra kell osztani és ezeket a csoportokat kell egy hexadecimális számmal leírni.

Példaként alakítsunk át egy 12 bites bináris számot hexadecimálissá!

$$110001111001_2 = \dots \text{...} \cdot 16 \cdot \dots$$

A megoldás első lépése a bináris szám számjegycinek 4 bites csoportokra való felosztása:

$$1100 \ 0111 \ 1001.$$

Második lépésként a csoportokat átírjuk hexadecimálisba:

$$1100_2 = 12_{10} = C_{16}$$

$$0111_2 = 7_{10} = 7_{16}$$

$$1001_2 = 9_{10} = 9_{16}$$

$$\text{Tehát: } 110001111001 = C79 \ H$$

A hexadecimálisból binárisba való áttérés az előzőek megfordításából adódik. Pl.

$$6H4A \ H = \dots\dots\dots_2 \text{ esetén}$$

$$6_{16} = 0110_2$$

$$B_{16} = 1011_2$$

$$4_{16} = 0100_2$$

$$A_{16} = 1010_2$$

Tehát egy hexadecimális helyértéket minden négy bites bináris számmal írunk le!

A feladat eredménye így:  $6B4A \ H = 0110101101001010_2$

### 1.3. Kódolás

Az információ leírását valamelyen egyezményes jelrendszerben kódolásnak nevezzük. Kódolás például a tízes számrendszerbeli szám átírása bináris számrendszerbe. Ebben az esetben a tízes számrendszerbeli szám az információ, a bináris számrendszer az egyezményes jelrendszer, az átírás folyamata pedig a kódolás. Természetesen a bináris szám visszairható a tízes számrendszerbe, ilyenkor a kódolási folyamatot visszafelé hajtjuk végre. Ezt nevezzük **dekkódolásnak**. A kódolás, ill. dekkódolás elnevezés csak attól függ, hogy melyik egyezményes jelrendszert tekintjük kiindulási információnak. Ezért általánosan fogalmazva a **kódolás** (ill. a dekkódolás) **áttérés egyik egyezményes jelrendszerből a másik egyezményes jelrendszerre**. A jelrendszeret **kódrendszernek** nevezzük, ezen egyes elemei a **kódszavak**. Ilyen értelemben pl. a tízes és a kettes számrendszer is kódrendszer, az egyes számok pedig a kódszavak.

A kódokat kódolt tartalmuk szerint a numerikus és az alfanumerikus kódok csoportjába soroljuk.

A **numerikus kódrendszerök** csak számokat kódolnak. A leggyakrabban alkalmazottak a következők:

- Bináris kód.** A számokat a kettes számrendszerben megismertek szerint kódoljuk.
- Hexadecimális kód.** A számok hexadecimális számrendszerben való leírása.
- Komplemens kód.** A komplement kiegészítőt jelent. A bináris kódok esetében a kiegészítést egyre, ill. kettőre lehet elvégezni.

Az **egyes komplement kódban** a bináris szám bitjeit egyre egészítjük ki. Pl. az 101011 bináris szám egyes komplemente: 010100 (vegyük észre, hogy egyszerűen a nullák helyett egyest, az egyesek helyett nullát kell írni).

**Kettes komplement kódban** a bináris számot előbb átírjuk egyes komplement kódba, majd hozzáadunk egyet. Pl. az előző szám kettes komplemente:

$$\begin{array}{r} 010100 \\ + \quad \quad 1 \\ \hline 010101 \end{array}$$

(Az összeadás szabályai:  $0+0=0$ ;  $0+1=1$ ;  $1+0=1$ ;  $1+1=0$ , marad 1, amit az egygel magasabb helyiértéken figyelembe veszünk).

- BCD kód.** A sokféle BCD kód közös jellemzője, hogy a decimális számokat helyi értékenként 0-tól 9-ig 4 biten binárisan kódoljuk. Innen ered a kódrendszer elnevezése: Binary Coded Decimal – binárisan kódolt decimális.

a) **8-4-2-1** súlyozású BCD kód (vagy természetes BCD kód, normál BCD kód, N-BCD kód). A súlyozás az egyes helyi értékeken álló bitek decimális értékét jelenti. A kódolás menetét jól mutatja a következő példa:

$$359_{10} = \dots \text{BCD} \cdot$$

Minden helyi értéken álló decimális számot 4 biten binárisan kódolunk:

$$3_{10} = 0011_2; 5_{10} = 0101_2; 9_{10} = 1001_2 \cdot$$

Tehát a 359 tízes számrendszerbeli szám BCD kódja:

$$359_{10} = 001101011001_{BCD}$$

A dekódolás úgy történik, hogy a BCD kódszót 4 bites csoportokra bontjuk a legkisebb helyi értéktől indulva és ezeket átírjuk decimálisba. Pl.

$$1001000100000111_{BCD} = \dots \cdot 10 \cdot$$

A legkisebb helyi értéktől 4 bites csoportokat képezve:

$$0111_2 = 7_{10}; 0000_2 = 0_{10}; 0001_2 = 1_{10}; 1001_2 = 9_{10},$$

tehát a dekódolt szám:  $9107_{10}$ .

- b) **Háromtöbbletes kód** (Excess-3 kód, XS-3 kód). A decimális számokhoz a náluk hárommal nagyobb szám bináris megfelelőjét rendeljük hozzá, ahogyan azt az 1.1. táblázat tartalmazza.
- A háromtöbbletes kód sajátossága, hogy
- minden kódszó tartalmaz legalább egy darab egyest,
  - a kód önkölcsönöző. A kódot leíró táblázat középvonalára (4. és 5. kódelem között) nézve az azonos távolságra elhelyezkedő kódszavak egymásnak bitenként egyes komplementeisei.
- c) Az **Aiken kód** olyan négybites kódrendszer, amelyben az egyes bitek súlyozása: 2-4-2-1.
- A súlyozásból következik, hogy a rendszerben kódolható legnagyobb szám a decimális 9 és a kódszavak 0-4 tartományban azonosak az NBCD kódszavakkal. A kód önkölcsönöző a rendszer 4. és 5. kódeleme köztengelyre nézve.
- d) **Gray-kód.** Alapváltozata 4 biten a decimális számjegyeket kódolja úgy, hogy az egymást követő kódszavak csak egy bitben térjenek el egymástól. A további Gray-kód változatuktól való megkülönböztetés miatt gyakran N-Gray- (normál-Gray-) kódnak nevezzük.

A Gray-kód és a decimális számok egymáshoz rendelését az 1.1. táblázat tartalmazza. A Gray-kód és az NBCD kód közötti átszámítás úgy végezhető, hogy balról indulva az NBCD első bitjét változatlanul leírjuk, majd az egymás melletti biteket összeadva megkapjuk a Gray-kódszót:

1 0 0 1

/ \ / \ /

1 1 0 1

A Gray-kód NBCD kóddá alakításakor úgy járunk el, hogy a Gray-kódszó első bitjét változatlanul leírjuk, majd a további biteket az előzőleg kiszámított NBCD bit és a következő átszámítandó Gray-bit összegzésével képezzük:

0 1 0 1

↓ ↓ ↓ ↓

0 → 1 → 1 → 0

BCD kódok. 1.1. táblázat

Decimális szám	N-BCD	XS-3	Aiken	N-Gray-kód
0	0000	0011	0000	0000
1	0001	0100	0001	0001
2	0010	0101	0010	0011
3	0011	0110	0011	0010
4	0100	<u>0111</u>	<u>0100</u>	0110
5	0101	1000	1011	0111
6	0110	1001	1100	0101
7	0111	1010	1101	0100
8	1000	1011	1110	<b>1100</b>
9	1001	1100	1111	1101

5. **Egylépéses kódok.** Az egylépéses kódok közös jellemzője, hogy az egymást követő kódszavak csak egy bitben térnek el egymástól. A ciklikus egylépéses kódok utolsó és első kódszavára is érvényes az egybites eltérés.

Egylépéses kódok. 1.2. táblázat

Decimális szám	Johnson-kód	Gray-kód
0	0000	0000
1	0001	0001
2	0011	0011
3	0111	0010
4	1111	0110
5	1110	0111
6	1100	0101
7	1000	0100
8		1100
9		1101
10		1111
11		1110
12		1010
13		1011
14		1001
15		1000

- a) Johnson-kód. Helyi értékes kód, ahol a 0 értékű kód után a legkisebb helyi értéktől haladva feltöltődik 1 bitekkel, majd a csupa egyest tartalmazó kódszó után 0 bitekkel. Ez a szabály bármilyen bitszám esetén igaz. Az **1.2.** táblázat a 4 bites Johnson-kódot mutatja. A táblázatból jól látható a ciklikusság, valamint az a sajátosság, hogy a 4 biten lehetséges 16 kódszóból a Johnson-kód csak 8 kódszót használ fel. Ez általánosságban is igaz más bitszámú Johnson-kódra is.
- b) Gray-kód. Az előző szakaszban megismert N-Gray-kód kiterjesztett változata, amely tartalmaz minden olyan kódszót, amit a bitszám lehetővé tesz. Az **1.2.** táblázat szerint a Gray-kód is ciklikus.

Az egylépéses kódokat elsősorban a vezérlés- és szabályozástechnikában használjuk, mert a kódszavak közötti egybites eltérés jól illeszkedik az elmozdulás, vagy előfordulás kódolási követelményeihez, ugyanakkor az esetleges pontatlanságból erődő kódszótévesztés nem okoz megengedhetetlenül nagy eltérést.

**6. Hibafelderítő és hibajavító kódok.** A kódrendszerben lévő kódszavak valamilyen információt hordoznak. Ezeknek az információknak az egyik áramköri egységből a másikba való átvitele során – különösen, ha az átvitel nagy távolságra történik – hiba keletkezhet. Hasonlóképpen megváltoztathatja a helyes információt egy meghibásodott áramkör is. Ha a kódrendszer valamennyi kódszavához információt rendeltünk, akkor a hiba egy másik, egyébként érvényes és megengedett, de jelen esetben nem helyes kódszót hoz létre.

Hibafelderítés akkor lehetséges, ha a kódrendszerbe olyan kódszavakat is beiktatunk, amelyekhez nem rendelünk információt, ezért ezek csak akkor fordulhatnak elő, ha valamelyik információt hordozó kódszó hibássá vált. Az így létrehozott kódrendszerben tehát lesznek információtartalommal bíró megengetett kódszavak, és hibát jelző tiltott kódszavak. A tiltott kódszavak miatt a kódrendszer **redundáns**: az információ szempontjából többletet tartalmazó.

A **bináris kód** használatánál a redundancia a **Hamming-távolsággal** jellemezhető, amely azt mutatja, hogy a rendszerben lévő bármely két kódszó között hány bitben van eltérés. A kódrendszer Hamming-távolsága a kódszavak közötti legkisebb távolság.

Ha egy kódrendszerben minden kódszóhoz rendelünk információt, akkor belátható, hogy a Hamming-távolság egységnyi, az ilyen kódrendszer az előzőek szerint nem redundáns.

a) **Paritásbittel kiegészített kód.** A bináris kód kódszavait egy redundanciát létrehozó paritásbittel egészítjük ki. Egy  $n$  bites bináris kód esetén az információt hordozó kódszavak száma  $2^n$ , a paritásbittel kiegészített kód kódszavainak száma  $2^{n+1}$ , tehát összesen kétszer annyi kódszó van, mint amennyi az információ kódolásához szükséges. A kódrendszer Hamming-távolsága 2.

A paritásbittel való kiegészítés kétféleképpen lehetséges:

- páros paritású kódrendszerben az információs kódszóban lévő egyesek számát a paritásbit páros darabszámról egészíti ki. Pl. ha az információ 1000110, akkor a paritásbit 1, mert eredetileg a kódszóban 3 db, tehát páratlan számú egyes volt, így a kiegészített kódszó: 1000110 1.
- páratlan paritású kódrendszerben az információs kódszóban lévő egyesek számának kiegészítése páratlan darabszárra történik. Az előző információ paritásbitje pl. 0, mert páratlan darabszámu egyes van eredetileg a kódszóban. A kiegészített kódszó: 1000110 0.

A paritásbittel kiegészített kódrendszerben a kódszavakban lévő egyesek számát folyamatosan figyelve, a hiba akkor deríthető fel, ha páratlan darabszámu bit változott hibásra. Csak ilyenkor változik meg ugyanis a kódszó paritása ellenkezőjére. A hibafelderítés tehát korlátozott mértékű. Javítható a hibafelderítés aránya a redundancia növelésével.

- b) **Hamming-kód.** Olyan redundáns kód, amelyben a Hamming-távolság három, így a hibafelderítésen kívül hibajavításra is lehetőséget ad, páratlan számú bit hibásra változása esetén. A kódrendszer négy információsbit és három paritásbit esetén az **1.3. táblázat** szerinti.

Hamming-kód. **1.3. táblázat**

Jelölés:	P <sub>1</sub>	P <sub>2</sub>	I <sub>3</sub>	P <sub>4</sub>	I <sub>5</sub>	I <sub>6</sub>	I <sub>7</sub>
súlyozás:	0	0	8	0	4	2	1
	0	0	0	0	0	0	0
	1	1	0	1	0	0	0
	0	1	0	1	0	1	0
	1	0	0	0	0	1	1
	1	0	0	1	1	0	0
	0	1	0	0	1	0	1
	1	1	0	0	1	1	0
	0	0	0	1	1	1	1
	1	1	1	0	0	0	0
	0	0	1	1	0	0	1

Az alkalmazott paritáskód minden hármon paritásbit esetén páros paritású:

- a P<sub>1</sub> paritásbit az I<sub>3</sub>, I<sub>5</sub>, I<sub>7</sub> információs biteket egészíti ki páros paritásúra,
- a P<sub>2</sub> paritásbit az I<sub>3</sub>, P<sub>4</sub>, I<sub>5</sub> biteket egészíti ki páros paritásra,
- a P<sub>4</sub> paritásbit az I<sub>5</sub>, I<sub>6</sub>, I<sub>7</sub> információs biteket egészíti ki páros paritásra.

A hibajavítás elve a következő:

- az ellenőrzendő kódszóból 4 bites csoportokat képezünk:  
 $H_1 \Rightarrow P_1, I_3, I_5, I_7$ ,  
 $H_2 \Rightarrow P_2, I_3, I_6, I_7$ ,  
 $H_4 \Rightarrow P_4, I_5, I_6, I_7$ .
- az egyes csoportokon páratlan paritást ellenőrzünk: ha a bitcsoport páratlan egyest tartalmaz, akkor az ellenőrzés eredménye 1, páros számú egyes esetén 0. A  $H_1 = 1$ , ha az ellenőrzés eredménye 1, egyébként  $H_1 = 0$ . A  $H_2 = 2$ , ha az ellenőrzés eredménye 1, egyébként  $H_2 = 0$ . A  $H_4 = 4$ , ha az ellenőrzés eredménye 1, egyébként  $H_4 = 0$ .
- a kiszámított H értékeket összeadva megkapjuk a hibás bit sorszámát.

5. **N-ből 1 kód.** Olyan 0 és 1 számjegyekből álló kód, amelyben N db bit van, de ezek közül minden kódszóban csak 1 db egyes van. Így a kódrendszer n darab kódszóból áll. Pl. 4-ből 1 kódrendszerben a tízes számrendszerbeli számok így kódolhatók.

#### Decimális szám 4-ből 1 kód

0	0001
1	0010
2	0100
3	1000

Az **alfanumerikus kódok** betűket, számjegyeket, írásjeleket és egyéb speciális jeleket kódolnak. A leggyakrabban használt alfanumerikus kód az ASCII-kód. A rövidítés az American Standard Code for Information Interchange. A kódrendszerben az egyes jelekhez 8 bites bináris kódot rendel egy egyezményes és nemzetközileg is elfogadott kódtáblázat. Ezt szemlélteti az **1.4.** táblázat (18. oldal).

### Ellenőrző kérdések

- Milyen összefüggés van a digitális jelek és a kettes számrendszer között?
- Hogyan végezhető el az átalakítás az egyes számrendszerök között?
- Soroljuk fel a BCD kódokat!
- Ismertessük az egylépéses kódokat!
- Ismertessük a hibafelderítő és hibajavító kódok kialakításának módszerét!

## ASCII-kódok. 1.4. táblázat

dec	hex	char	dec	hex	char	dec	hex	char	dec	hex	char	dec	hex	char	dec	hex	char	dec	hex	char
0	00	NUL	32	20	space	64	40	@	96	60	'	128	80	€	160	A0	á	192	C0	Ł
1	01	SOH	33	21	!	65	41	À	97	61	à	129	81	ő	161	A1	í	193	C1	ł
2	02	STX	34	22	"	66	42	฿	98	62	฿	130	82	é	162	A2	ó	194	C2	ł
3	03	ETX	35	23	#	67	43	฿	99	63	฿	131	83	â	163	A4	ú	195	C3	ł
4	04	EOT	36	24	\$	68	44	฿	100	64	฿	132	84	ä	164	A5	ñ	196	C4	-
5	05	ENQ	37	25	%	69	45	฿	101	65	฿	133	85	à	165	A6	ñ	197	C5	+
6	06	ACK	38	26	&	70	46	฿	102	66	฿	134	86	å	166	A7	°	198	C6	·
7	07	BEL	39	27	,	71	47	฿	103	67	฿	135	87	ç	167	A8	º/ó*	199	C7	‡
8	08	BS	40	28	(	72	48	฿	104	68	฿	136	88	è	168	A9	í	200	C8	‡
9	09	TAB	41	29	)	73	49	฿	105	69	฿	137	89	ë	169	AA	¬	201	C9	‡
10	0A	LF	42	2A	*	74	4A	฿	106	6A	฿	138	8A	è	170	AA	—	202	CA	‡
11	0B	VT	43	2B	+	75	4B	฿	107	6B	฿	139	8B	฿	171	AB	—	203	CB	‡
12	0C	FF	44	2C	,	76	4C	฿	108	6C	฿	140	8C	฿	172	AC	...	204	CC	‡
13	0D	CR	45	2D	-	77	4D	฿	109	6D	฿	141	8D	฿	173	AD	,	205	CD	—
14	0E	SO	46	2E	.	78	4E	฿	110	6E	฿	142	8E	฿	174	AE	»	206	CB	‡
15	0F	SI	47	2F	/	79	4F	฿	111	6F	฿	143	8F	฿	175	AF	«	207	CF	‡
16	10	DLE	48	30	0	80	50	฿	112	70	฿	144	90	฿	176	B0	‡	208	D0	‡
17	11	DC1	49	31	1	81	51	฿	113	71	฿	145	91	฿	177	B1	‡	209	D1	‡
18	12	DC2	50	32	2	82	52	฿	114	72	฿	146	92	฿	178	B2	‡	210	D2	‡
19	13	DC3	51	33	3	83	53	฿	115	73	฿	147	93	฿	179	B3	—	211	D3	‡
20	14	DC4	52	34	4	84	54	฿	116	74	฿	148	94	฿	180	B4	—	212	D4	—
21	15	NAK	53	35	5	85	55	฿	117	75	฿	149	95	฿	181	B5	—	213	D5	—
22	16	SYN	54	36	6	86	56	฿	118	76	฿	150	96	฿	182	B6	—	214	D6	—
23	17	ETB	55	37	7	87	57	฿	119	77	฿	151	97	฿	183	B7	—	215	D7	—
24	18	CAN	56	38	8	88	58	฿	120	78	฿	152	98	฿	184	B8	—	216	D8	—
25	19	EM	57	39	9	89	59	฿	121	79	฿	153	99	฿	185	B9	—	217	D9	—
26	IA	→ SUB	58	3A	:	90	5A	฿	122	7A	฿	154	9A	฿	186	BA	—	218	DA	—
27	IB	← ESC	59	3B	;	91	5B	฿	123	7B	฿	155	9B	฿	187	BB	—	219	DB	—
28	IC	Ł FS	60	3C	<	92	5C	฿	124	7C	฿	156	9C	฿	188	BC	—	220	DC	—
29	ID	↔ GS	61	3D	=	93	5D	฿	125	7D	฿	157	9D	฿	189	BD	—	221	DD	—
30	IB	↑ RS	62	3E	>	94	5E	^	126	7E	~	158	9E	฿	190	BE	—	222	DE	—
31	IF	▼ US	63	3F	?	95	5F	—	127	7F	□	159	9F	฿	191	BF	—	223	DF	—

\* CWI kód

## 2. LOGIKAI ALGEBRA

### 2.1. A logikai algebra alapvető összefüggései

A logikai algebra segítségével a szóban megfogalmazott feladatokat logikai függvény formájában írhatjuk fel. A logikai függvényeket a logikai algebra törvényei és alaptételei által lehet a legegyszerűbb alakra hozni. Az egyszerűsítés célja az, hogy a feladatot megvalósító áramkör a legkevesebb alkatrészből legyen felépíthető. A logikai algebrát másképpen Boole-algebrának nevezzük.

Pl. legyen a feladat egy gépkocsiriasztó készítése, amely úgy működik, hogy:

- akkor riaszt, ha az ajtót, a motorháztetőt vagy a csomagtartó tetejét kinyitják,
- akkor riaszt, ha a tulajdonos nem kapcsolta ki a riasztót.

Ezt az egyszerű feladatot úgy oldjuk meg, hogy a megfogalmazott eseményeket **feltételeknek** tekintjük, amelyeknek **függvénye** a feladatban meghatározott összefüggések szerint a riasztás bekövetkezése. Tehát, történjen riasztás, ha **vagy** az ajtót, **vagy** a motorháztetőt, **vagy** a csomagtartót kinyitották **és** a tulajdonos **nem** kapcsolta ki a riasztót. A szövegben szereplő ÉS, VAGY, NEM a feltételek között teremt függvénykapcsolatot, meghatározva ezzel a függvényt.

Ezek a függvénykapcsolatok a logikai algebra alapfüggvényei. A logikai algebra az egyszerűbb leírás érdekében egy jelölésrendszeret használ a függvények leírására:

- a feltételeket az ABC nagybetűvel jelöljük és független változónak hívjuk,
- a függvényt leggyakrabban F betűvel jelöljük,
- az ÉS kapcsolat jelölésére a szorzás jelét ( $\cdot$ ) használjuk, pl. A·B,
- a VAGY kapcsolat jelölésére az összeadás jelét (+) használjuk, pl. A+B,
- a NEM (tagadás) jelölésére a felülvonást használjuk, pl. A'.

Ha a példában az ajtó kinyitását A-val, a csomagtartóét B-vel, a motorháztetőt C-vel jelöljük, riasztó kikapcsolását pedig D-vel, akkor a logikai függvény:

$$F^4 = (C + B + A) \cdot \overline{D}.$$

Ez a logikai függvény **algebrai alakban** való leírása. Az F felső indexében a szám azt jelöli, hogy a függvény független változóinak száma négy. A független változók **igazak** vagy **hamisak** lehetnek, így a függvény is lehet igaz vagy hamis értékű: igaz, ha bekövetkezik, hamis, ha nem. Ehhez a két értékhez jól illeszkedik a kettes

számsrendszer. A hamis értékét 0-val, az igazat 1-gel jelölve egyszerűen leírhatók a **logikai alapfüggvények** egy olyan táblázattal, amelyből változók bármilyen értéke mellett kiolvasható a függvény értéke. Az ilyen táblázatot **igazságtablázatnak** nevezzük. Kétváltozós (A, B) függvényekre az igazságtablázatok:

**ÉS függvény:**

B	A	F
0	0	0
0	1	0
1	0	0
1	1	1

**VAGY függvény:**

B	A	F
0	0	0
0	1	1
1	0	1
1	1	1

**NEM függvény:**

A	F
0	1
1	0

Ugyanezek a függvények algebrai alakban:

$$F^2 = B \cdot A$$

$$F^2 = B + A$$

$$F = \overline{A}$$

Gyakran az alapfüggvények angol elnevezését használjuk:

AND

OR

NOT

Másképpen:

konjunkció

diszjunkció

negáció

Szavakban megfogalmazva:

- az ÉS függvény akkor igaz, ha az A és a B változó is igaz, tehát valamennyi változó igaz,
- a VAGY függvény akkor igaz, ha **bármelyik** változó igaz.

Az alapsüggvényekből adódik néhány sokszor használt egyszerű függvény.

A NEM-ÉS, másképpen NAND függvény az ÉS függvény tagadása. Algebrai alakja:

$$F^2 = \overline{B \cdot A}$$

A **NAND függvény** igazságtablázata:

B	A	F
0	0	1
0	1	0
1	0	0
1	1	0

A **NEM-VAGY**, másképpen NOR függvény, a VAGY függvény tagadása.

$$F^2 = \overline{B + A}$$

B	A	F
0	0	1
0	1	0
1	0	0
1	1	0

**A kizáró-VAGY**, másnéven XOR (Exclusive OR) függvény:

$$F^2 = B \cdot \overline{A} + \overline{B} \cdot A$$

Egyszerűbb leírására külön műveleti jelet alkalmaz a logikai algebra, ez a „körkereszt”. Ezzel:

$F^2 = B \oplus A$	B	A		F
	0	0		0
	0	1		1
	1	0		1
	1	1		0

Az igazságáblázat alapján a függvénykapcsolat úgy fogalmazható meg, hogy a kizáró-VAGY kapcsolat a változók azonosságát zárja ki és csak akkor igaz, ha az egyik változó igaz, a másik pedig hamis. A XOR függvényt ezért szokás **antivalencia függvénynek** is nevezni.

A **megengedő-ÉS függvény** vagy másnépen ekvivalencia a változók azonossága csetén igaz.

$F^2 = \overline{B} \cdot \overline{A} + B \cdot A$	B	A		F
	0	0		1
	0	1		0
	1	0		0
	1	1		1

A kétváltozós függvényekhez hasonlóan a logikai függvények több változóval is felírhatók. Pl. egy négyváltozós NAND kapcsolat:

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

Egy bonyolult gyakorlati feladat logikai függvényének felírásánál gyakran előfordul, hogy nem a legegyszerűbb függvény adódik. A függvények egyszerűsítését a logikai algebra törvényei és alaptételei teszik lehetővé.

**A logikai algebra törvényei** a kommutativitás (felcserélhetőség), az asszociativitás (csoportosíthatóság) és a disztributivitás (szétválaszthatóság, szétoszthatóság).

**A kommutativitás** az ÉS, ill. VAGY műveletben szereplő változók felcserélhetőségét jelenti:

$$B + A = A + B,$$

$$B \cdot A = A \cdot B.$$

Az **asszociativitás** lehetővé teszi, hogy a függvényben szereplő azonos műveleteket tetszőleges sorrendben hajtsuk végre:

$$(C + B) + A = C + (B + A),$$

$$C \cdot (B \cdot A) = (C \cdot B) \cdot A.$$

A **disztributivitás** törvénye szerint mindenkor, hogy előbb a VAGY és azután az ÉS kapcsolatát végezzük-e el, vagy fordítva:

$$(C + B) \cdot A = (B \cdot A) + (C \cdot A),$$

$$(C \cdot B) + A = (B + A)(C + A).$$

**A logikai algebra alaptételei** egyszerű logikai azonosságokat írnak le:

$$(1) A \cdot \overline{A} = 0;$$

$$(2) A + \overline{A} = 1;$$

$$(3) \overline{\overline{A}} = A;$$

$$(4) A + A = A;$$

$$(5) A \cdot A = A;$$

$$(6) A \cdot 0 = 0;$$

$$(7) A + 0 = A;$$

$$(8) A \cdot 1 = A;$$

$$(9) A + 1 = 1.$$

A felsoroltak mellett alaptételként alkalmazzuk a De-Morgan-tételt:

$$\overline{B \cdot A} = \overline{B} + \overline{A};$$

$$\overline{B + A} = \overline{B} \cdot \overline{A}.$$

A törvények és az alaptételek alkalmazása jól követhető a következő példákon.

**1. feladat**

Egyszerűsítsük a logikai függvényeket!

$$a) F^3 = B \cdot A + C \cdot B \cdot A,$$

$$b) F^3 = C \cdot B \cdot A + \bar{C} \cdot A + C \cdot \bar{B} \cdot A,$$

$$c) F^3 = \overline{B \cdot \bar{A} + \bar{C} \cdot B}.$$

**Az 1. feladat megoldása**

Az a) feladat megoldása:

$$F^3 = B \cdot A + C \cdot B \cdot A.$$

A disztributivitást felhasználva kiemelhető  $B \cdot A$ :

$$F^3 = B \cdot A \cdot (1 + C).$$

A (9) téTEL miatt:

$$F^3 = B \cdot A \cdot 1.$$

A (8) téTEL szerint:

$$F^3 = B \cdot A.$$

A b) feladat megoldása:

$$F^3 = C \cdot B \cdot A + \bar{C} \cdot A + C \cdot \bar{B} \cdot A.$$

A disztributivitást alkalmazva az első és a harmadik változócsoportra:

$$F^3 = C \cdot A \cdot (B + \bar{B}) + \bar{C} \cdot A.$$

A (2) téTEL miatt:

$$F^3 = C \cdot A \cdot 1 + \bar{C} \cdot A.$$

A (8) téTEL miatt:

$$F^3 = C \cdot A + \bar{C} \cdot A.$$

Újra a disztributivitást használva:

$$F^3 = A \cdot (C + \bar{C}).$$

A (2) téTEL miatt:

$$F^3 = A.$$

A c) feladat megoldása:

$$F^3 = \overline{B \cdot \bar{A} + \bar{C} \cdot B}.$$

A De-Morgan-tételt használva:

$$F^3 = \overline{\overline{B \cdot \bar{A}} \cdot \overline{\bar{C} \cdot B}}.$$

Újra alkalmazva a két változócsoporthoz:

$$F^3 = (\overline{B} + A) \cdot (C + \overline{B}).$$

A disztributivitás miatt:

$$F^3 = \overline{B} \cdot A + \overline{B} \cdot \overline{B} + C \cdot A + C \cdot \overline{B}.$$

Az (5) tételel miatt:

$$F^3 = \overline{B} \cdot A + \overline{B} + C \cdot A + C \cdot \overline{B}.$$

Az első, a második és a negyedik változócsoporthoz kiemelhető  $\overline{B}$  (disztributivitás):

$$F^3 = \overline{B} \cdot (A + 1 + C) + C \cdot A.$$

A (9) tételel szerint:

$$F^3 = \overline{B} \cdot 1 + C \cdot A.$$

A (8) tételel alkalmazva:

$$F^3 = \overline{B} + C \cdot A.$$

## 2.2. Logikai függvények grafikus egyszerűsítése

Az algebrai úton elvégzett egyszerűsítésekkel kitűnik, hogy akkor lesz egyszerűbb a függvény, ha a kiemelések (disztributivitás) révén a zárójelben maradó kifejezés valamelyik alaptételnek felelnek meg. A kiemelés akkor végezhető el, ha van legalább két olyan változócsoporthoz, amelyek csak egy változó értékében térnek el egymástól. Ezt a tényt használja fel a **grafikus egyszerűsítési eljárás**.

**A grafikus egyszerűsítés szabályos alakú függvények egyszerűsítésére alkalmas.**

**A szabályos alakú függvények** (másképpen: normál alakú függvények) termekből állnak. A **term** olyan változócsoporthoz, amelyben minden változó szerepel igaz vagy hamis (más elnevezéssel: ponált vagy negált) értékkel és ezeket a változókat azonos függvénykapcsolatok kötik össze. Egy háromváltozós függvény esetében term pl. a  $C \cdot \overline{B} \cdot A$  vagy term pl. a  $\overline{C} + B + \overline{A}$  változócsoporthoz. A példákból is látható, hogy kétféle term lehetséges. A **minterm** olyan term, amelyben a változókat ÉS kapcsolat köti össze. Ilyen a  $C \cdot \overline{B} \cdot A$ . A **maxterm** olyan term, amelyben a változók között VAGY kapcsolat van. Az  $\overline{C} + B + \overline{A}$  változócsoporthoz pl. maxterm. A kétféle termből kétféle szabályos függvényalak hozható létre.

**A diszjunktív szabályos alakú függvény** mintermek VAGY kapcsolatából áll. Ilyen pl. az  $F^3 = \overline{C} \cdot B \cdot \overline{A} + C \cdot \overline{B} \cdot A + C \cdot B \cdot \overline{A}$  függvény.

**A konjunktív szabályos alakú függvény** maxtermek ÉS kapcsolata. Ilyen pl. az  $F^3 = (C + \overline{B} + \overline{A}) \cdot (\overline{C} + \overline{B} + \overline{A})$  függvény.

A logikai függvények igazságtáblázattal és szabályos alakkal való megadása közötti összefüggést a következő példa mutatja egy háromváltozós függvényre. Legyen a függvény igazságtáblázata:

C	B	A	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

A függvény  $F^3 = 1$  igaz értékű az igazságtáblázat alapján, ha C nem igaz és B sem igaz és A sem igaz, vagy még akkor is igaz a függvény, ha C nem igaz és B nem igaz és A igaz, vagy még akkor is igaz, ha... A szöveges leírás helyett egyszerűbb az algebrai alakban való leírás:

$$F^3 = \overline{C} \cdot \overline{B} \cdot \overline{A} + \overline{C} \cdot \overline{B} \cdot A + C \cdot \overline{B} \cdot \overline{A} + C \cdot \overline{B} \cdot A.$$

Az eredményül kapott függvény diszjunktív szabályos alakú függvény. Levonható tchát az a következtetés, hogy a **diszjunktív szabályos alakú függvény a logikai függvény igaz értékét meghatározó termeket írja le**.

Az igazságtáblázattal adott függvény  $F^3 = 0$  hamis értékű, ha az  $F^3 = 0$  sorokhoz tartozó változócsoportok hamis logikai értéket adnak eredményül. Ez pl. a harmadik sorban azt jelenti, hogy a C változó igaz, vagy a B változó hamis, vagy az A változó igaz akkor a függvény hamis. A negyedik sorban: hamis a függvény, ha a C változó igaz, vagy a B változó hamis, vagy az A változó hamis. A hetedik és a nyolcadik sorban hasonlóképpen határozható meg a függvény hamis értéke. Ez alapján a függvény algebrai alakja:

$$F^3 = (C + \overline{B} + A) \cdot (C + \overline{B} + \overline{A}) \cdot (\overline{C} + \overline{B} + A) \cdot (\overline{C} + \overline{B} + \overline{A}).$$

Az utóbbi gondolatmenet azt mutatja, hogy a **konjunktív szabályos alakú függvény a logikai függvény hamis értékét meghatározó termeket írja le**.

A szabályos alakú függvények az algebrai, vagy az igazságtáblázattal történő megadásnál egyszerűbben is megadhatók. Erre az ad lehetőséget, hogy a termekben minden változó szerepel, ezért a változóknak helyi értékkel tulajdoníthatunk, amiből kiszámítható a termek értéke. Ezt az értéket a term **sorszámanak** nevezzük. A termen belüli változók helyi értékei: A –  $2^0$ , B –  $2^1$ , C –  $2^3$ , D –  $2^4$  stb. A termek sorszámanak számításánál a negált változó zérus értékű, a ponált pedig a helyi értéknek megfelelő számértékű. A minttermek és a maxtermek sorszámait és szokásos jelöléseiket, példaként háromváltozós függvényre, a **2.1** táblázat tartalmazza.

A háromváltozós függvények termjei. **2.1.** táblázat

Mintermek	Maxtermek	Sorszám
$m_0^3 : \bar{C} \cdot \bar{B} \cdot \bar{A}$	$M_0^3 : (\bar{C} + \bar{B} + \bar{A})$	0
$m_1^3 : C \cdot \bar{B} \cdot A$	$M_1^3 : (\bar{C} + \bar{B} + A)$	1
$m_2^3 : \bar{C} \cdot B \cdot \bar{A}$	$M_2^3 : (\bar{C} + B + \bar{A})$	2
$m_3^3 : \bar{C} \cdot B \cdot A$	$M_3^3 : (\bar{C} + B + A)$	3
$m_4^3 : C \cdot \bar{B} \cdot \bar{A}$	$M_4^3 : (C + \bar{B} + \bar{A})$	4
$m_5^3 : C \cdot \bar{B} \cdot A$	$M_5^3 : (C + \bar{B} + A)$	5
$m_6^3 : C \cdot B \cdot \bar{A}$	$M_6^3 : (C + B + \bar{A})$	6
$m_7^3 : C \cdot B \cdot A$	$M_7^3 : (C + B + A)$	7

Hasonlóképpen számíthatók ki bármilyen változószámú függvény termjeinek sorszámai.

Egy  $n$  változós függvény  $i$ -edik termjével felírható a két szabályos alakú függvény általános alakja:

diszjunktív szabályos függvény sorszámos alakja:

$$F^n = \sum^n m_i^n,$$

konjunktív szabályos függvény sorszámos alakja:

$$F^n = \prod^n M_i^n.$$

A  $\Sigma$  (szumma, összeg) jelölés arra utal, hogy a mintermeket VAGY kapcsolatba kell hozni egymással, a  $\Pi$  (produktum, szorzat) jelölés pedig a maxteremek ÉS kapcsolatát jelöli.

## 2. feladat

Irjuk fel a függvények sorszámos szabályos alakját!

a)  $F^4 = \bar{D} \cdot \bar{C} \cdot B \cdot \bar{A} + D \cdot \bar{C} \cdot \bar{B} \cdot A + D \cdot C \cdot B \cdot A + D \cdot \bar{C} \cdot \bar{B} \cdot \bar{A},$

b)  $F^4 = (D + C + \bar{B} + \bar{A}) \cdot (\bar{D} + \bar{C} + \bar{B} + \bar{A}) \cdot (\bar{D} + C + \bar{B} + A) \cdot (D + C + B + A).$

### A 2. feladat megoldása

a) a mintermek sorszámai:

$$\bar{D} \cdot \bar{C} \cdot B \cdot \bar{A} \Rightarrow 2; \quad D \cdot \bar{C} \cdot \bar{B} \cdot A \Rightarrow 9; \quad D \cdot C \cdot B \cdot A \Rightarrow 15; \quad D \cdot \bar{C} \cdot \bar{B} \cdot \bar{A} \Rightarrow 8.$$

A függvény sorszámos alakja:

$$F^4 = \Sigma^4(2,8,9,15).$$

b) a maxtermek sorszámai:

$$(D + C + \bar{B} + \bar{A}) \Rightarrow 12; \quad (\bar{D} + \bar{C} + \bar{B} + \bar{A}) \Rightarrow 0; \quad (\bar{D} + C + \bar{B} + A) \Rightarrow 5;$$

$$(D + C + B + A) \Rightarrow 15.$$

A függvény sorszámos alakja:

$$F^4 = \Pi^4(0,5,12,15).$$

A nem szabályos alakú függvényeket szabályos alakra az átalakítandó függvény bővítésével lehet átírni. A bővítés során a függvény értéke nem változhat, ezért diszjunktív alakra hozásnál ÉS kapcsolatba hozzuk 1 értékkel, konjuktív alakra hozásnál pedig VAGY kapcsolatba hozzuk 0 értékkel. Az 1 és 0 értéket az alaptételek szerint a hiányzó változó és negáltja VAGY kapcsolatából, ill. ÉS kapcsolatából hozzuk létre. Pl. az  $F^3 = \bar{C} \cdot B + \bar{B} \cdot A$  függvény első tagját  $A + \bar{A} = 1$  értékkel, a második tagját pedig  $C + \bar{C} = 1$  értékkel kell bővíteni a diszjunktív alakra hozáshoz:

$$F^3 = \bar{C} \cdot B \cdot (A + \bar{A}) + (C + \bar{C}) \cdot \bar{B} \cdot A = \bar{C} \cdot B \cdot A + \bar{C} \cdot B \cdot \bar{A} + C \cdot \bar{B} \cdot A + \bar{C} \cdot \bar{B} \cdot A.$$

A konjuktív alakra való átírás pl. az  $F^3 = (\bar{C} + \bar{A}) \cdot (C + B)$  függvényen követhető.

A függvény első tagját  $B \cdot \bar{B} = 0$  értékkel, a második tagját  $A \cdot \bar{A} = 0$  értékkel kell bővíteni:

$$\begin{aligned} F^3 &= [B \cdot \bar{B} + (\bar{C} + \bar{A})] \cdot [A \cdot \bar{A} + (C + B)] = \\ &= (\bar{C} + B + \bar{A}) (\bar{C} + \bar{B} + \bar{A})(C + B + A)(C + B + \bar{A}). \end{aligned}$$

A két szabályos függvényalak között szükség esetén elvégzhető az átalakítás. **A szabályos alakú függvények közötti átalakítást** az átalakítandó függvény kétszeres negálásával lehet elvégezni. Az egyik negálás a De-Morgan-tétel szerint a függvényben szereplő logikai függvénykapcsolatokat változtatja meg: az ÉS kapcsolatból VAGY kapcsolat lesz, a VAGY kapcsolatból pedig ÉS, így a mintermkból maxtermek, a maxtermekből pedig mintermek. A másik negálás pedig visszaállítja az eredeti függvényt, hiszen az  $\bar{\bar{A}} = A$  alaptétel szerint csak a kétszeres negálás nem változtatja meg a függvény értékét. A kétszeres negálást úgy végezzük el, hogy

- első lépésben az átalakítandó függvény sorszámos alakjából kiírjuk azokat a termeket, amelyek nincsenek a függvényben. Mivel a függvény eredetileg azokat a termeket tartalmazza amelyek mellett igaz a függvény, ezért a nem szereplő termek a hamis függvényt határozzák meg,
- második lépésben a negált függvényben szereplő termek komplementét végezzük. A komplementképzést (kiegészítést) minden a függvény maximális term-sorszámrára végezzük el. A komplementképzés eredménye az átalakított függvény.

Ezt az átalakítási módszert algebrai átalakításnak nevezzük.

Az átalakítás két lépése jól azonosítható a következő példán.

### 3. feladat

Alakitsuk át a következő függvényeket!

- a)  $F^3 = \Pi^3(0,3,6,7)$ .
- b)  $F^4 = \Sigma^4(1,2,5,8,10,13)$ .

### A 3. feladat megoldása

Az a) feladat megoldása

- első lépésben az átalakítandó függvényből meghatározzuk a benne nem szereplő termeket:  
 $\bar{F}^3 = \Pi^3(1,2,4,5)$ ,
- második lépésben a negált függvény termjeinek komplementét képezzük. Háromváltozós függvénynél a max. sorszám 7, ezért a kiegészítés 7-re történik:  
 $F^3 = \Sigma^3(6,5,3,2)$ .

A termeket sorrendbe írva:

$$F^3 = \Sigma^3(2,3,5,6).$$

A b) feladat megoldása

A negált függvény:

$$\bar{F}^4 = \Sigma^4(0,3,4,6,7,9,11,12,14,15).$$

Komplemensképzés és sorarendezés után:

$$F^4 = \Pi^4(0,1,3,4,6,8,9,11,12,15).$$

**A logikai függvények grafikus egyszerűsítése** kettő-, három- és négyváltozós függvényknél használatos. Az egyváltozós függvényeknél értelmetlen, az öt- és ennek több változó esetén lehetséges, de túlzottan bonyolult. A grafikus egyszerűsítési eljárást Veitch és Karnaugh dolgozta ki. A két eljárás között szinte csak a jelölérendszerben van különbség, ezért az eljárást Veitch–Karnaugh-módszernek, röviden V–K módszernek nevezzük, a felhasznált táblázatokat pedig V–K tábláknak.

Az egyszerűsítéshez a szabályos alakú függvény termjeit táblázatba foglaljuk, úgy kiállítva a táblázatot, hogy az **egymás mellé kerülő termek csak egy változóban térjenek el egymástól**. Ezt a szabályt betartva olyan termek kerülnek egymás mellé, amelyek ha benne vannak az egyszerűsítendő függvényben, akkor felhasználhatók az egyszerűsítéshez. Ezt tapasztaltuk az algebrai egyszerűsítés során is. Pl. háromváltozós függvénynél csak egy változóban térnek el egymástól a  $\bar{C} \cdot B \cdot A$  és a  $C \cdot B \cdot \bar{A}$  termek, ezért összevonhatók és így egyszerűsithetők:

$$(C + \bar{C}) \cdot B \cdot \bar{A} = B \cdot \bar{A}.$$

A függvény diszjunktív szabályos alakjának egyszerűsítéséhez a **minterm táblázatokat**, a konjunktív függvényalakjának egyszerűsítéséhez pedig a **maxterm táblázatokat**.

**tokat** használjuk. A kettő-, három- és négyváltozós függvények minterm táblázatait a 2.1. ábra, a maxterm táblázatokat pedig a 2.2. ábra tartalmazza.

		B					
		B					
		B					
A							
B		0	1				
		2	3				
	C	0	1	3	2		
		4	5	7	6		
	D						
		12	13	15	14		
		8	9	11	10		
	A						

2.1. ábra. Minterm táblázatok

		B					
		B					
		B					
A							
B		3	2				
		1	0				
	C	7	6	4	5		
		3	2	0	1		
	D						
		15	14	12	13		
		11	10	8	9		
		3	2	0	1		
		7	6	4	5		
	A						

2.2. ábra. Maxterm táblázatok

A táblázatok cellákból állnak, minden cellában szerepel a cellához tartozó term sorszáma. A sorok és az oszlopok mellett vonalak jelzik, hogy melyik változó igaz a je llt sorokban, ill. oszlopokban. Az egyes cellákba beírva a termek algebrai alakjait ellenörizhető a leírt jelölésrendszer helyessége valamint az, hogy az egymás melletti termek csak egy változóban térnek el egymástól. Jól látható, hogy ez a szabály minden irányban igaz, sőt ilyen értelemben egymás mellett lévőnek számítanak:

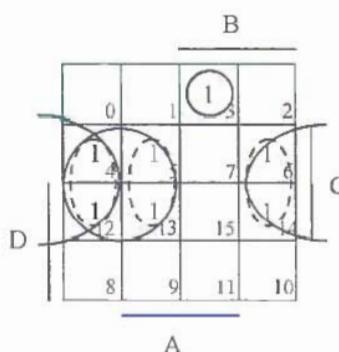
- az első és az utolsó oszlop cellái páronként, pl. a mintermtáblákban a 4 és 6 cellák,
- az első és az utolsó sor cellái páronként, pl. a négyváltozós mintermtáblában a 3 és 11 cellák,
- a sarkokban lévő négy cella: 0, 2, 8, 10.

(A cellák ismertetett elrendezése egy lehetséges változat, a szomszédos cellákra vonatkozóan megismert szabályt betartva más elrendezésű táblázatok is létrehozhatók.)

**A diszjunktív szabályos alakban adott függvény úgy egyszerűsíthető,** hogy a függvény változóinak száma szerint kiválasztott minterm táblázatban megjelöljük a függvényben szereplő termeket. A jelölés általában a megfelelő cellába írt 1. A táblázat szomszédos celláiba kerülő, ezért összevonható termeket kettesével egy hurokba összevonjuk. Ha két hurok egymás mellé kerül, akkor ezeket négyes hurokká vonjuk össze, két szomszédos négyes hurkot nyolcas hurokká stb. Ha van olyan term, amely nem vonható össze más termekkel, akkor ez a term egyes hurkot képezi vagyis nem egyszerűsíthető. Akkor lesz a függvény a legegyszerűbb alakú, ha a lehető legnagyobb hurkokat képezzük, de ügyelünk arra, hogy a hurkok száma a lehető legkevesebb legyen. Miután valamennyi megjelölt termet sikerült bevonni valamelyik hurokba, a táblázat szélein található jelöléseket felhasználva meghatározzuk azokat a változókat, amelyek a hurok teljes területén belül azonos értékűek. Mivel az eredeti függvény mintermeiből állt, ezért a hurok területén belül azonos értékű változók ÉS kapcsolatban vannak egymással, és az így kialakított változócsoportokat pedig VAGY kapcsolatok kötik össze. A szabályokat betartva a legegyszerűbb függvényhez jutunk.

A grafikus egyszerűsítés megismert szabályainak gyakorlati alkalmazását egy feladat megoldásával tekintjük át. Egyszerűsítük ezért lépésenként az  $F^4 = \Sigma^4(3,4,5,6,12,13,14)$  diszjunktív szabályos alakú függvényt!

Mivel a függvény négyváltozós diszjunktív függvény, ezért a négyváltozós mintermtáblát használjuk. A táblázatban bejelöljük az egyszerűsítendő függvényben szereplő mintermeket a 4, 5, 6 stb. sorszámú cellákban, amint azt a 2.3. ábra mutatja.



2.3. ábra. A négyváltozós függvény mintermtáblája

Az ábrán láthatóan a 3 sorszámú mintermnek nincs szomszéda, ez egyes hurkot jelent, tehát nem egyszerűsíthető. Kettes hurok képezhető a 4-12, az 5-13 és a 6-14 párok ból, mert szomszédos termek. Ezt a hurkolási lehetőséget mutatja a szagatott vonal. A 4-12 és az 5-13 hurkok azonban szintén szomszédok, ezért összevonhatók, négyes hurkot hozva létre. Hasonlóképpen szomszédos hurkoknak számítanak.

nak a 4-12 és 6-14 hurkok, tehát ezek is összevonhatók négyes hurokba. Így kialakultak a folyamatos vonallal jelölt, lehető legnagyobb hurkok, és mivel valamennyi termet már belefoglaltuk valamelyik hurokba, így a hurkolást befejeztük. Érdemes megfigyelni, hogy a 4 és 12 sorszámú termeket két hurok kialakításánál is figyelembe vettük. Ez megengedhető, mert a függvény értéke azzal nem változik meg, ha ugyanazokat a változókat többször szerepeltek jön a függvényben (l. az  $A \cdot A \cdot A \cdot \dots = A$  alaptételt).

A hurkolás után meghatározzuk minden hurokra külön-külön azokat a változókat, amelyek a hurok teljes területén belül azonos értékűek. A 4-5-12-13 hurokban a C változó igaz értékű, a B változó pedig hamis értékű, ezért ezt a hurkot a  $C \cdot \bar{B}$  változócsoporthoz jellemzi. A D és az A változó a hurok felében igaz, a másik felében hamis, így ezek nincsenek a kiolvasott változócsoporthoz. A 4-12-6-14 hurkon belül a C igaz értékű, az A hamis értékű. A hurkot leíró változócsoporthoz:  $C \cdot \bar{A}$ . Az egyes hurokban szerepel valamennyi változó:  $\bar{D} \cdot \bar{C} \cdot B \cdot A$ .

A változócsoporthoz a változókat ÉS kapcsolatok kötik össze, mert mintermek összevonásából jöttek létre. Mivel diszjunktív alakból indultunk ki, a változócsoporthoz VAGY kapcsolata az egyszerűsített függvény:  $F^4 = \bar{D} \cdot \bar{C} \cdot B \cdot A + C \cdot \bar{A} + C \cdot \bar{B}$ .

A függvény konjuktív alakjával is elvégezhető az egyszerűsítés. **A konjuktív szabályos alakú függvényt** a megismert szabályok szerint lehet egyszerűsíteni, azzal az eltéréssel, hogy az egyszerűsítésre a maxtermtáblát használjuk és a táblázatban szereplő hamis termeket hurkoljuk. Ez az eltérés annak az előző megállapításunknak a következménye, hogy a konjuktív szabályos alak a függvény hamis értékét meghatározó termeket írja le.

A maxtermtábla kétféleképpen tölthető ki:

- az igazságátlábatból az  $F = 0$  függvény értékekhez tartozó mintermek sorrendjét megállapítjuk és beírjuk a maxtermtáblába. Példánkban az igazságátlábat első sorának első oszlopában szereplő maxterm:  $(D + C + B + A)$ , sorszáma 15. A második oszlopban szereplő maxterm:  $(D + C + B + \bar{A})$ , sorszáma 14 stb.
- ha rendelkezésükre áll a függvény mintermtáblája, akkor ezt tükrözziük a maxtermtáblába: a mintermtábla egyeseteit a maxtermtábla azonos pozíciójú cellájába írjuk. Az üresen maradt cellákat megjelöljük 0-val. Példánkban ez a 2.3. ábra mintermtáblázatának felhasználásával végezhető el.

Bármelyik módszert is használjuk, egy olyan maxtermtáblához jutunk, amelyben a termeket 0 értékekkel jelölik. Ezekre alkalmazzuk a hurkolásnál és a kiolvasásnál megismert szabályokat. A kiolvasásnál figyelembe vesszük, hogy maxtermes alakból indultunk ki, ezért a hurkokban azonos értékű változók VAGY kapcsolatban vannak, és a hurkot jellemző változócsoporthoz között ÉS kapcsolat van.

Szemléltetésként az előző feladat függvényének konjuktív alakjával is végezzük el az egyszerűsítést! A kitöltött maxtermtáblát a 2.4. ábra szemlélteti.

	B			
D	0 15 14	1 12 13		C
	1 11 10	0 8 9		
	1 3 2	0 6 1	1 1	
	0 7 6	0 4 5	0 5	C
A			A	

2.4. ábra. A négyváltozós függvény maxtermtáblája

A táblázat alapján négycs hurok képezhető a 7-6-4-5, valamint a 15-14-7-6 termekből, kettes hurokba foglalható a 0-8 term. A 13 sorszámú term miatt még egy négyes hurkot kell kialakítani a négy sarokból. Ez a 15-13-7-5 termeket tartalmazza. A megismert szabályt követve kiolvashatók a hurkokra jellemző változóscsoportok és ezek ÉS kapcsolatával felírható az egyszerűsített függvény:

$$F^4 = (\bar{D} + C) \cdot (C + B) \cdot (\bar{C} + \bar{B} + \bar{A}) \cdot (C + A).$$

Egy adott gyakorlati feladatot leíró logikai függvénynél gyakran előfordul, hogy vannak olyan bemeneti változókombinációk (termek), amelyek a feladat szempontjából nem meghatározottak, mert vagy nem fordulnak elő, vagy ha előfordulnak is, a hozzájuk tartozó függvényérték közömbös. Ezeket a kimeneti állapotokat **határozatlan állapotoknak** nevezzük. A határozatlan termek a függvény sorszámos alakjában úgy adhatók meg, hogy h betűvel jelölve külön felsoroljuk ezeket. Pl.  $F^4 = \Sigma^4(4,6,7,13,15, h: 1,5,10)$ . A függvények egyszerűsítésénél a határozatlan termek szabadon felhasználhatók a hurkok képzésénél, ha használatukkal a függvény egyszerűbb lesz. A felirt függvényt grafikusan egyszerűsítve a mintermtáblával a 2.5. ábrán látható eredményre jutunk.

	B			
D	0 X 1 3 2			C
	1 4 X 5 1 6			
	12 13 1 15 14			
	8 9 11 X 10			
A			A	

2.5. ábra. A határozatlan termek felhasználása

A mintermtáblából látható, hogy az 5 sorszámú, határozatlan termet felhasználva két négyes hurkot lehet kialakítani. Ha az 5 term nem lenne, akkor csak három kettes hurkot lehetne képezni, ami bonyolultabb függvényt eredményezne. Az 1 és 10 határozatlan termekre nincs szükség. Az egyszerűsített függvény  $F^4 = \overline{D} \cdot C + C \cdot A$ . A határozatlan term nélkül  $F^4 = \overline{D} \cdot C \cdot A + \overline{D} \cdot C \cdot B + D \cdot C \cdot A$ .

A feladatok megoldása során megismertük egy függvény minterm és maxterm táblázata közötti összefüggést. Ezt felhasználhatjuk a két sorszámos függvényalak közötti átalakításra is. Ez a **grafikus átalakítási módszer** jóval egyszerűbb, mint a korábban megismert algebrai módszer: az átalakítandó sorszámos függvényt ábrázoljuk saját táblázatában, majd áttükrözük a másik függvényalak táblázatába. Innen kiolvashatók az átalakított függvény sorszámai. Ezt a módszert mutatja be a 2.6. ábrán látható két táblázat.

		B					
		7	0	6	4	0	5
		3	2		0	0	1
		A			A		A

		B							
		1	0	0	1	1	3	0	2
		1	4	1	5	0	7	1	6
		A				A			A

2.6. ábra. A függvények grafikus átalakítása

A maxtermtábla az átalakítandó  $F^3 = \Pi^3(0,5,6)$  függvényt ábrázolja. Ezt átírva a mintermtáblába, kiolvashatók a diszjunktív függvény sorszámai:  $F^3 = \Pi^3(0,3,4,5,6)$ .

A V-K táblák használatával jelentősen egyszerűsíthető a nem szabályos alakú függvények szabályos alakúvá alakítása. Megkülönböztetésül az algebrai úton végzett szabályos alakra hozástól a módszert **grafikus szabályos alakra hozásnak nevezzük**.

Legyen a szabályos alakra hozandó függvény  $F^3 = \overline{C} \cdot A + B \cdot \overline{A}$ ! Mivel a változócsoportokban a változók ÉS kapcsolatban vannak, a mintermtáblát használjuk.

A 2.7. ábra mintermtáblájában azokat a cellákat kell megjelölni, amelyek a függvényben szereplő változócsoportokkal leírhatók.

		B					
		0	1	1	3	1	2
C	A	4	5	7	1	6	

		B					
		7	0	6	4	5	
C	A	3	0	2	0	0	1

2.7. ábra. A függvények szabályos alakra hozása

A  $\overline{C} \cdot A$  változócsoporttal jellemzhető cellák keresése: a C változó hamis értéke a táblázat felső sorára jellemző. Ezen belül az A változó az 1 és 3 cellákban igaz. Ezt a két cellát megjelöljük. A  $B \cdot \overline{A}$  celláinak keresése: a B változó a jobb oldali két oszlopban igaz, ezen belül az A változó a 2 és 6 cellákban hamis, ezért ezt a két cellát is megjelöljük. Végül a megjelölt cellák sorszámait kiolvasva megkapjuk a szabályos alakú függvényt:  $F^3 = \Sigma^3(1,2,3,6)$ .

Hasonlóképpen járunk el akkor is, ha olyan függvényt kell szabályos alakra hozni, amelyben a változók VAGY kapcsolatba vannak egymással, de most a maxtermtáblát használjuk. Az  $F^3 = (B + \overline{A}) \cdot (\overline{C} + \overline{A})$  függvény szabályos alakra hozását a 2.7. ábra maxterm táblája mutatja. A  $(B + \overline{A})$  váltoozócsoport a 2 és 6 cellákat, a  $(\overline{C} + \overline{A})$  változócsoport pedig a 0 és 2 cellákat írja le. Ezért ezeket kell megjelölni 0-val, hiszen most konjunktív szabályos alakúra hozzuk a függvényt. A jelölt cellák sorszámait kiolvasva kapjuk a szabályos alakú függvényt:  $F^3 = \Pi^3(0,2,6)$ .

**Összefoglalva** a logikai algebrával kapcsolatban leírtakat, megállapíthatjuk, hogy az ismeretek a következő gondolatmenetre épülnek.

A logikai függvények események bekövetkezésének feltételeit írják le logikai változókkal és a közöttük lévő logikai függvénykapcsolatokkal. A három logikai alapfüggvény a tagadás, az ÉS valamint a VAGY kapcsolat.

A logikai függvényeket a minimális alkatelemekkel való megvalósíthatóság miatt a lehető legegyszerűbb alakra kell hozni. Az egyszerűsítés algebrai és grafikus módszerrel végezhető. Az algebrai egyszerűsítés a Boole-algebra törvényeinek és alaptételeinek felhasználásával végezhető. A grafikus egyszerűsítéshez a logikai függvények szabályos alakjára van szükség. A nem szabályos alakú függvények szintén algebrai és grafikus módszerrel hozhatók szabályos alakra.

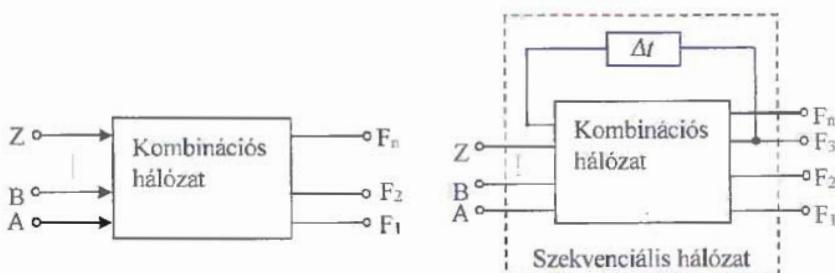
Egy logikai függvény grafikus egyszerűsítése elvégezhető a függvény diszjunktív és konjunktív szabályos alakjával. A diszjunktív szabályos alak egyszerűsítéséhez a minterm V-K táblázatot, a konjunktív szabályos alak egyszerűsítéséhez pedig a maxterm táblát használjuk. A két V-K táblával a szabályos alakok egymásba átalakíthatók.

**Ellenőrző kérdések**

1. Melyek a logikai algebra törvényei és alaptételei?
2. Mit jelent a szabályos alakú függvény, milyen fajtait ismerjük?
3. Értelmezzük a term, minterm, maxterm és a szabályos alakú függvények fogalmát!
4. Milyen elven épülnek fel a V-K táblák?
5. Ismertessük a grafikus egyszerűsítés szabályait!

### 3. A LOGIKAI HÁLÓZATOK ALAPELEMEI

A logikai függvényekkel megfogalmazott feladatok áramkori megvalósítása logikai hálózatokkal történik. A logikai hálózatok felépítésük és működésük szerint két csoportba sorolhatók: kombinációs logikai hálózatok és sorrendi logikai hálózatok. A két hálózat tömbvázlatát a 3.1. ábra mutatja.



3.1. ábra. Logikai hálózatok

**A kombinációs logikai hálózat** kimeneteinek állapotát a bemenetekre adott változók értékei határozzák meg. A kimeneti függvényeket olyan bemeneti változók határozzák meg, amelyek értéke nem függ az időtől, kizárolag csak az adott pillanatban érvényes bemeneti feltételektől. Az ilyen hálózatok logikai függvényeit megvalósító áramkori elemek a *logikai kapuáramkörök*, vagy szokásos rövidebb elnevezéssel *kapuk*.

**A sorrendi logikai hálózatok** (szekvenciális hálózatok) kimeneti függvényeit a bemeneti változók értékén kívül a kimenet előző állapota is meghatározza. Az áramkori felépítés szempontjából ez azt jelenti, hogy a  $t_n$  időpontban érvényes **előző kimeneti állapotot** tárolni kell és visszacsatolás kialakításával a bemenetre kell vezetni. Így a hálózat  $t_{n+1}$  időpontban létrejövő **következő kimeneti állapotát** az előző állapot is képes befolyásolni. Ugyanolyan A, B, C stb. bemeneti változó értékek mellett tchát a kimenetek állapota különböző lehet az áramkör előző állapotától függően.

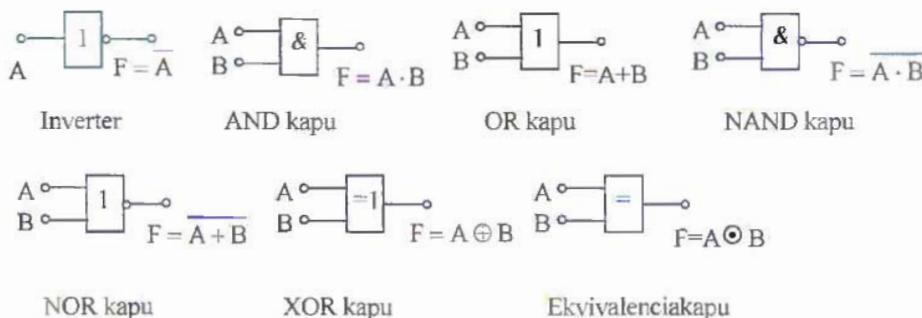
A szekvenciális hálózatok felépítéséhez tárolókat használunk, amelyek kapuáramkörökből kialakított visszacsatolt hálózatok.

### 3.1. Kapuáramkörök

A kapuáramkörök a logikai alapfüggvényeket megvalósító digitális integrált áramkörök. A megvalósított függvény és a hozzátartozó kapuk elnevezése:

Függvény	Kapuáramkör
negáció	inverter
ÉS	AND
VAGY	OR
NEM-ÉS	NAND
NEM-VAGY	NOR
kizáró VAGY	XOR (antivalencia)
megengedő ÉS	ekvivalencia

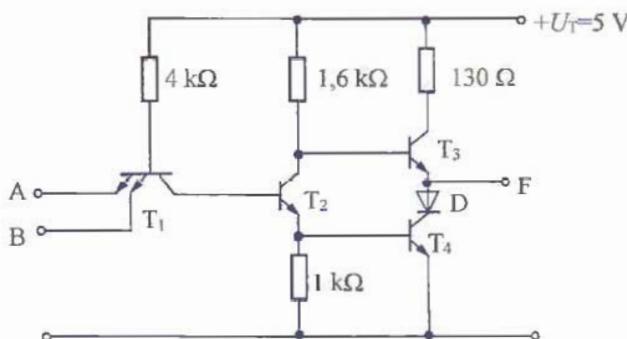
Az AND, OR, NAND, NOR kapuk bemenetcinek száma típusonként változó. Nem alapfüggvényt, de gyakran előforduló függvénykombinációt megvalósító kapuáramkör az AND-OR-INVERTER kapu. A kapuáramkörököt kapcsolási rajzokon jölképi jelölésekkel ábrázoljuk. Ezeket foglalja össze a 3.2. ábra.



**3.2. ábra.** A kapuáramkörök jelképi jelölései

A digitális integrált kapuáramkörök belső áramköreit bipoláris vagy MOS tervezérlé-  
sű tranzisztorokból építik fel. A bipoláris tranzisztorból felépített kapuk a TTL  
(Tranzisztor-Tranzisztor-Logika) áramkörök. A MOS tranzisztorokból több külön-  
böző kapcsolástechnikával létrehozott kapuáramkör ismert. Ezek közül gyakorlati je-  
lentősége szinte kizárolag a CMOS (komplementer-MOS) kapuáramköröknek van.

**A TTL kapuáramkörök** belső kapcsolástechnikájának legjellemzőbb áramköre a NAND kapu. A 3.3. ábra a kétbemenetű NAND kapu kapcsolási rajzát mutatja.



3.3. ábra. TTL NAND kapu belső felépítése

A  $T_1$  többemitteres (multiemitteres) tranzisztor ÉS kapcsolatot valósít meg emitterei, tchát a kapu bemeneti között. Ha az emitterek valamelyikét vagy minden kettőt 0 V feszültségre (közös potenciál) kötjük kívülről, akkor a tranzisztor kinyit, mert a bázisán lévő pozitív feszültséghoz képest az emitterre negatívvabb feszültséget kap. Csak akkor zár le a tranzisztor, ha minden emitterére a tápfeszültség (vagy azt megközelítő feszültség) kerül. A 0 V-ot logikai 0-nak, a tápfeszültséget pedig logikai 1 szintnek tekintve, a multiemitteres tranzisztor működését úgy is megfogalmazhatjuk, hogy a tranzisztor kinyit, ha az A-B bemeneteire 0-0, 0-1, 1-0 logikai értékek jutnak. Csak akkor zár le a tranzisztor, ha minden bemenetére 1 szint kerül. A tranzisztor zárt állapota tehát a bemeneti változók ÉS kapcsolata szerint jön létre.

A  $T_2$  tranzisztor zárt állapotban van, ha a  $T_1$  tranzisztor nyitott, mert a bázisára ilyenkor a nyitott  $T_1$   $U_{CE}$  feszültsége (a tranzisztor szaturációs feszültsége) kerül. A zárt  $T_2$  tranzisztoron nem folyik áram, ezért az emitter-ellenállásán nem esik feszültség, a  $T_4$  zárt állapotban lesz. A  $T_2$  kollektor-ellenállásán sem csökken feszültség, ezért a  $T_3$  nyitott állapotban van, mert a bázisa tápfeszültségre kapcsolódik. Végeredményben tehát a kimeneten a lezárt  $T_4$  és a nyitott  $T_3$  miatt közel tápfeszültség, vagyis 1 szint van. A bemeneti kombinációkat tekintve ez a kimeneti 1 szint a bemeneti 0-0, 0-1, 1-0 értékekhez tartozik.

A  $T_2$  tranzisztor nyitott állapotban van, ha a  $T_1$  zárt állapotú. A  $T_2$  tranzisztoron ezért áram folyik, ami az emitter-ellenálláson akkora feszültséget hoz létre, hogy a  $T_4$  kinyisson. A kollektor-ellenálláson eső feszültség viszont lezárja a  $T_3$  tranzisztorát. A kimeneten így a  $T_4$  szaturációs feszültsége mérhető, ami logikai 0 szintet jelent. Ezt a kimeneti állapotot tehát, a bemeneten lévő 1-1 állapot hozta létre.

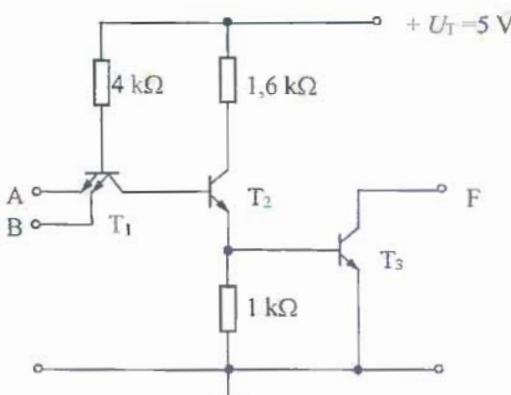
Áttekintve a bemeneti állapotokhoz tartozó kimeneti állapotokat, felismerhető a NAND kapcsolat.

A kapcsolásban szereplő D dióda a  $T_3$  tranzisztor biztos zárasát segíti. Amikor a  $T_2$  és a  $T_4$  nyitottak, akkor hozzávetőlegesen igaz, hogy  $U_{E2} = U_{BE4} = 0,6$  V,

$U_{C2} = U_{E2} + U_{S2} = 0,8 \text{ V}$  és  $U_{C4} = U_{S4} = 0,2 \text{ V}$ . Dióda nélkül a  $T_3$  bázisa és emittere közé így  $U_{c2} - U_{c4} = 0,6 \text{ V}$  kerül. Ez éppen a nyitás határa. A diódával együtt a  $T_3$  nyitásához kb. 1,2 V-ra van szükség, vagyis a 0,6 V biztosan nem tudja kinyitni a tranzisztor.

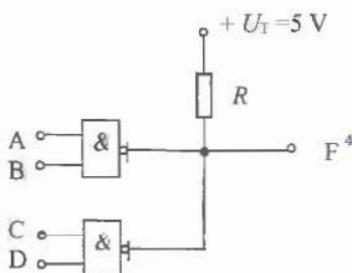
A  $T_3$ ,  $T_4$  tranzisztorokból és a D diódából álló kimeneti fokozat neve: **totem-pole** (totem oszlop) **kimenet**. Az ilyen kimeneti megoldást az indokolta, hogy ezzel csökkenhető legjobban a kaput terhelő kapacitások jelkiseleltető hatása. Egy hálózatban a kapu kimenetére újabb kapuk csatlakoznak, amelyek bemeneti kapacitásait a lehető leggyorsabban fel kell tölteni 1 szintre, amikor a kimenet magas szintre vált és ki kell sütni, szintén a lehető leggyorsabban, amikor a kimenet 0-ra vált (l. az Elektronika tankönyv 5.3.1. pontjában a kapcsolási időről leírtakat). A kapuáramkörök bemeneti kapacitásait elsősorban a többemitteres tranzisztor és a  $T_2$  elektrodakapacitásai határozzák meg. Nagyságrendileg ez a kapacitás néhány szor 10 pF. A terhelő-kapacitás töltése és kisütése akkor a gyors, ha a lehető legkisebb ellenálláson keresztül történik. A totem-pole kimenet ennek a követelménynek tesz eleget, hiszen az 1 szintet (tápfeszültség) a nyitott  $T_3$  tranzisztoron, a 0 szintet pedig szintén a nyitott  $T_4$  tranzisztoron keresztül kapcsolja a terhelésre és ezzel együtt a kapacitásra. A totem-pole kimenet tehát gyors működést tesz lehetővé.

A kapuáramkörökből kialakított hálózatokban előfordul, hogy két kapukimenetet össze kell kötni. Ebben az esetben a totem-pole kimenettel rendelkező kapuáramkör nem használható, hiszen ha a kapuk kimenetét ellentétes szintre vezéreljük, akkor valamelyik kapu valószínűleg tönkremegy. Ilyenkor nyitott kollektoros kimenetű kapukat kell használni. A nyitott kollektoros kapu kapcsolási rajza és jelképi jelölése a 3.4. ábrán látható.



3.4. ábra. Nyitott kollektoros TTL kapu

A kapu csak külső ellenállással kiegészítve működik, ezen keresztül kapcsolódik a  $T_3$  tranzisztor a tápfeszültségre. Ugyanerre az ellenállásra kapcsolható a másik (többi) nyitott kollektoros kapu is. Ezt mutatja a 3.5. ábra. Az ábrán megfigyelhető a nyitott kollektoros kapuk jelképi jelölése is.

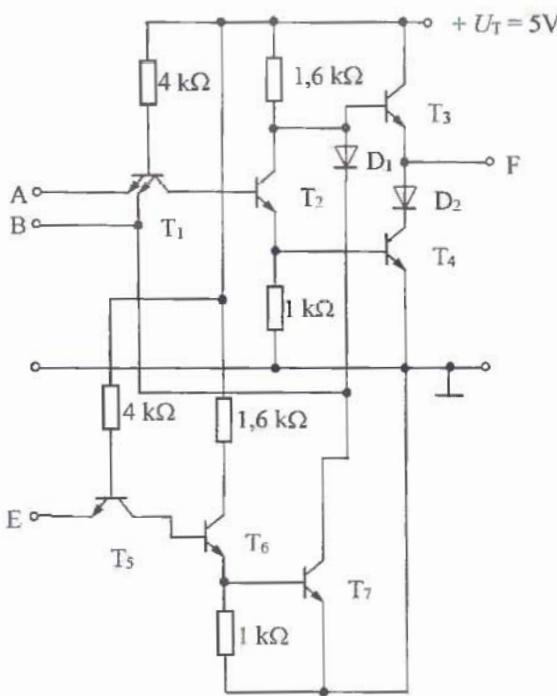


3.5. ábra. A nyitott kollektoros kapuk összekötése

A kapuáramkörök összefüggésével egy új logikai függvénykapcsolat jön létre, amit huzalozott függvénykapcsolatnak nevezünk: bármelyik kapuáramkör kimenete 0 szintre vált, a közös kimenet is 0 lesz. Csak ha minden kimenet 1 szintű, akkor lesz a közös kimenet is 1. Ha NAND kapukat huzalozunk, akkor a kimenetek  $A \cdot B$ , ill.  $C \cdot D$  függvények szerint igazak, ezért a huzalozott függvénykapcsolat  $F^4 = A \cdot B \cdot C \cdot D$ . Az összekötetőség előnye mellett hátrány, hogy jelentősen megnövekszik a jelkésletetési idő, mert a terhelő kapacitások töltése 1 szinten az ellenálláson keresztül megy végbe.

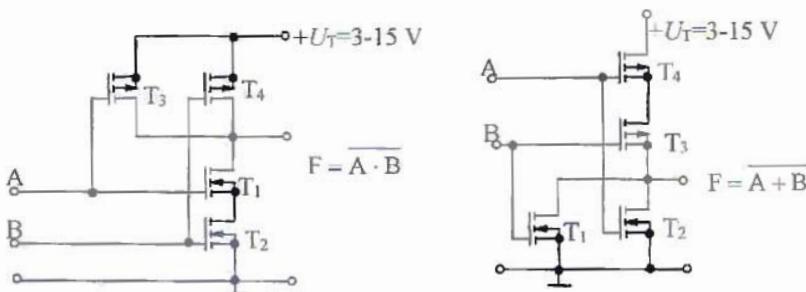
Az előzőekben megismert totem-pole és nyitott kollektoros kimeneti megoldás mellett készítenek háromállapotú (tri-state) kimeneti megoldással ellátott kapuáramköröket. A 0 és 1 logikai állapotok mellett a harmadik állapot a közel árammentes, nagyimpedanciás állapot. Ilyen állapotban tehát a kimenet gyakorlatilag lekapcsolódik a terhelésről. A belső áramköri kialakítást a 3.6. ábra mutatja.

A  $T_5$ ,  $T_6$ ,  $T_7$  tranzisztorokból álló tiltó áramkör E (Enable – engedélyezés) bemetelére 1 szintet (tápfeszültség) adva, lezárja a  $T_5$  és kinyitja a  $T_6$  és a  $T_7$  tranzisztor. A nyitott  $T_7$  közvetlenül lezárja  $T_3$ -at és a  $T_1$  nyitásán keresztül a  $T_4$  tranzisztort is. A kimeneten tehát csak a kimeneti tranzisztorok szaturációs áramai folynak. Ha  $T_5$  emitterére 0 szint kerül, akkor  $T_5$  nyitva,  $T_6$  és  $T_7$  pedig zárva van, ezért a tiltó áramkör hatástanlan és a kapu úgy működik, mint egy szokásos totem-pole kimenetű áramkör. A tri-state kimenetű áramköröket sín kialakítására használjuk. A sín egy vezeték, amelyre kapuk kimenetei és más kapuk bemenetei kapcsolódnak. Bármelyik kapu kimenet összefüggésbe kerülhet a sínén keresztül a kapubemenetekkel, ha a kimenetét engedélyezzük. Természetesen alaphelyzetben valamennyi kapukimenet harmadik állapotban van.



3.6. ábra. A háromállapotú kimeneti megoldás kapcsolási rajza

A **CMOS kapuáramkörök** n és p csatornás MOS tranzisztorokból épülnek fel, amelyek páronként komplementer kapcsolásban működnek. Felépítésük igen egyszerű, amint azt a 3.7. ábra is mutatja NAND és NOR kapuk esetén.



3.7. ábra. CMOS NAND és NOR kapuk

A NAND kapu n csatornás tranzisztorai a gate-elektródájukra adott 0 V feszültségre, tehát logikai 0 szintre lezárnak, a p csatornások pedig kinyitnak. Ha 0 szintet kapcsolunk pl. az A bemenetre, akkor a T<sub>1</sub> tranzisztor lezár, a T<sub>3</sub> viszont kinyit. Ezért a B bemenet vezérlésétől függetlenül a kimeneten közelítőleg tápfeszültség, tehát logikai 1 szint van. Hasonló a helyzet, ha a B bemenetet kapcsolunk 0 szintet. A mindenbemenetre adott 0 szint szintén 1 kimeneti állapotot hoz létre. A T<sub>1</sub> és T<sub>2</sub> soros kapcsolása miatt csak akkor lehet a kimeneten 0 szint, ha A is és B is 1 szintre kapcsolódik. Ilyenkor T<sub>1</sub> és T<sub>2</sub> nyit, T<sub>3</sub> és T<sub>4</sub> lezár. Érdemes megfigyelni, hogy a kimeneti szinteket nyitott tranzisztorok kapcsolják a kimenetre, ezért a kapacitív terhelések töltése-kisütése szempontjából ugyanolyan kedvező az áramkör kialakítása, mint a totem-pole.

A NOR kapu kimenetén 0 szint jelenik meg, ha akár az A, akár a B bemenetre 1 szint kerül, mert vagy T<sub>1</sub>, vagy T<sub>2</sub> nyit. Ha mindenbemenetet adott 0 szinttel, akkor T<sub>3</sub> és T<sub>4</sub> egyszerre lesz nyitva, ezért a kimeneten logikai 1 szint jelenik meg.

A kapuáramkörök típusjelölése meglehetősen egységes. A TTL áramkörök jelölése az SN betükkel kezdődik. Az ezt követő két szám azt a környezeti hőmérsékleti tartományt jelöli, amelyen belül az áramkör – a gyártó által megadott jellemzőkkel – használható:

SN 74..., SN 49..., SN 75...: 0 °C-tól +70 °C-ig,

SN 84...: -25 °C-tól +85 °C-ig,

SN 54...: -55 °C-tól +125 °C-ig.

A hőmérsékletet jelző két szám után következő számok az áramkört azonosítják. Pl. SN 7400 áramkör négy két bemenetű NAND kaput tartalmaz. Az SN sorozat nem csak kapuáramköröket, hanem más funkciót ellátó digitális áramköröket is tartalmaz. Az SN sorozattal csereszabatos (kompatibilis) CMOS áramköröket is készítenek. A TTL-től való megkülönböztetésül a sorozatot jelölő betűk és a hőmérsékleti tartományt jelölő két szám után egy C betű szerepel a típusjelölésben. Pl. SN 74C00 négy darab kétbemenetű CMOS NAND kapu.

A CMOS áramkörök közül a legszélesebb áramkörválasztékkal a CD 40... sorozat rendelkezik. A 40-et követő számok az áramkört azonosítják, az ezután következő újabb betűk közül az első a működési (környezeti) hőmérsékleti tartományt mutatja, a második pedig a tokozás fajtáját:

első betű M -55 °C-tól +125 °C-ig,

C -40 °C-tól +85 °C-ig,

második betű D dual-in-line

K flat-pack (a tokozások megnevezésének értelmezését a Tankönyvmester Kiadó Villamos anyagismeret technológia technológiá c. tankönyve ismerteti).

A kapuáramkörök, mint integrált alkatrészek a következő katalógusadatokkal jellemezhetők:

- tápfeszültség.** A jelenleg használatos digitális integrált kapuáramkörök pozitív tápfeszültségről működnek. A TTL áramkörök tápfeszültsége egységesen  $5\text{ V} \pm 5\%$ . A CMOS áramkörök tápfeszültségét a felhasználó  $3\text{ V}$  és  $15\text{ V}$  között megválaszthatja. A tápfeszültséggel azonban az áramkör valamennyi egyéb jellemzője is megváltozik,
- logikai szintek.** A TTL áramkörök logikai szintjei az állandó tápfeszültség következtében jól meghatározott, a gyártó által garantált értékek, külön az áramkörök bemenetére és külön a kimenetére.

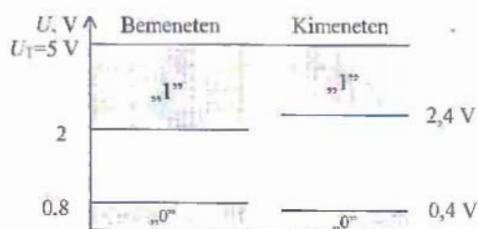
A bemeneten a logikai 0 szint feszültségtartománya:  $0\text{ V} - 0,8\text{ V}$ ,

a logikai 1 szint feszültségtartománya:  $2\text{ V} - 5\text{ V}$ .

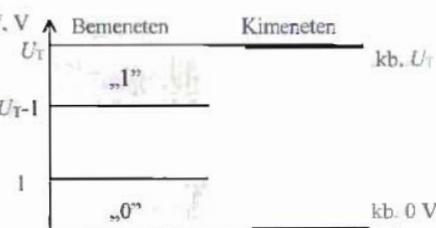
A kimeneti logikai 0 szint feszültségtartománya:  $0\text{ V} - 0,4\text{ V}$ ,

logikai 1 szint feszültségtartománya:  $2,4\text{ V} - 5\text{ V}$ .

Ezek az adatok határértékek. A TTL áramkörök logikai szintjeinek határértékeit a **3.8. ábra** foglalja össze.



**3.8. ábra.** A TTL áramkörök logikai szintjei



**3.9. ábra.** A CMOS áramkörök logikai szintjei

A CMOS áramkörök logikai 1 szintjeit az alkalmazott  $U_T$  tápfeszültség határozza meg:

A bemeneten a logikai 0 szint feszültségtartománya:  $0\text{ V}$ -tól  $1\text{ V}$ -ig,  
a logikai 1 szint feszültségtartománya:  $(U_T - 1\text{ V})$ -tól  $-U_T$ -ig.

A kimeneti logikai 0 szint:  $0\text{ V}$ ,  
logikai 1 szint:  $U_T$ .

A CMOS áramkörök logikai szintjei a **3.9. ábrán** láthatók.

- **zajtartalék vagy zajérzéketlenség.** Ezen a feszültségtartományon belüli feszültségváltozás nem változtatja meg az áramkör logikai állapotát. Ez egy áramkör kimcnete és az azt követő áramkör bemenete között értelmezett mennyiség. A TTL áramkörökönél a 3.8. ábrából következően 0 és 1 szinten is 0,4 V az értéke, mert  $0,8 - 0,4 = 0,4$  V, ill.  $2,4 - 2 = 0,4$  V. A CMOS áramkörök-nél a 3.9. ábra alapján 0 és 1 szinten is kb. 1 V.
- **kimeneti terhelhetőség, fan out.** Az áramkörök max. kimeneti árama határozza meg, hogy mekkora az a terhelés, amely mellett az áramkör még helyesen működik. A digitális áramkörök kimcnete általában újabb digitális áramkörbemeneteket hajt meg, ezért a max. kimeneti áramnál jobban jellemzi a kimeneti terhelhetőséget az, hogy hány darab áramkörbemenetet képes a kimenet meghajtani. Egységherhelésnek egy kapuáramkör bemenete számít.

TTL áramkörökönél a fan out értéke általában 10.

A CMOS áramkörök fan out értéke elvileg igen nagy lehetne, hiszen a MOS tranzisztorok vezérléséhez nincs szükség áramra, így a bemenetek nem terhelnék a kimenetet. A MOS tranzisztorok viszonylag nagy bemeneti kapacitása a kapcsolás pilanatában jelentősen terheli a meghajtó áramkört, ez korlátozza a kimenetre kapcsolható bemenetek számát.

Készülnek kifejczetten nagyáramú kimenettel rendelkező áramkörök, amelyek fan out értéke általában 30.

- **teljesítménydisszipáció,  $P_D$ .** Az áramkör tápfeszültsége és tápáram felvételle határozza meg. A TTL kapuk normál kivitel esetén  $P_D = 10$  mW. Készítenek alacsony teljesítményfelvételű TTL áramköröket úgy, hogy megnövelik az áramkörben lévő ellenállások értékeit. Ezzel elérhető a  $P_D = 1$  mW teljesítményfelvétel, de romlik az áramkör működési sebessége. A normál tipustól való megkülönböztetésül az ilyen áramköröket L betűvel jelölik, pl. SN 74L00.

A CMOS áramkörök legnagyobb előnye – az egyszerűsége mellett – az igen kicsi teljesítményfelvétel. A működésből következik, hogy a komplementer tranzisztorok közül az egyik minden le van zárva, ezért csak a szaturációs áram folyik rajta. A CMOS kapuk teljesítményfelvétele ugyanakkor erősen frekvenciafüggő. Pl. egy CMOS kapu teljesítményfelvétele 1 kHz frekvencián kb. 10 nW, 1 MHz frékvencián már kb. 1 mW.

- **jelkésleltetési, vagy jelterjedési idő,  $t_{pd}$  (propagation delay time).** A bemeneten történő változás és a hatására létrejövő kimeneti változás között eltelt idő. Szoros kapcsolatban van az áramkört felépítő tranzisztorok jelkésleltetési idejével, amit – mint már láttuk – elsősorban a tranzisztorok kapacitásai okozzák. A jelkésleltetési idő nagysága általában különböző a 0–1 és az 1–0 átmenetnél. Az áramkörök adatlapjai a két idő átlagát, az átlagos jelkésleltetési időt adják meg.

A TTL áramkörökknél alapesetben a jelkésleltetési idő  $t_{pd} = 10$  ns. A belső ellenállások értékének csökkentésével készítenek nagysebességű áramköröket, amelyeknél  $t_{pd} = 6$  ns. Ezzel együtt viszont megnövekszik a disszipációs teljesítmény  $P_D = 23$  mW-ra. Az ilyen áramkört H betűvel jelölik, pl. SN 74H00. Az áramkörön belül az ellenállások megváltoztatása tehát vagy a teljesítményfelvételt csökkenti de növeli a jclkésleltetést (L típus:  $P_D = 1$  mW, de  $t_{pd} = 33$  ns), vagy a jelkésleltetést csökkenti de növeli a teljesítményfelvételt (H típus).

Nagy működési sebesség és mérsékelt teljesítményfelvétel növekedés jellemzi a Schottky-diódás telítésgátlással ellátott áramkört. A jelteredmény idő  $t_{pd} = 3$  ns, a disszipáció pedig  $P_D = 19$  mW. A jelölés S, pl. SN 74S00.

Egyszerre csökken a disszipáció és a jelkésleltetés az Schottky-diódás telítésgátlással ellátott, L típusú áramkörökknél. Ezek betűjele LS, pl. SN 74LS00.

A CMOS áramkörök jelkésleltetési ideje meglehetősen nagy, 20 ns és 200 ns közötti érték. Ez számít a CMOS áramkörök legnagyobb hátrányának.

### Ellenőrző kérdések

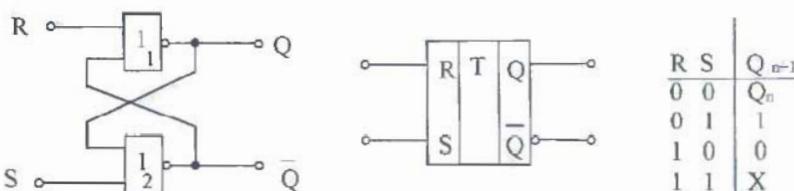
1. Ismertessük az alapvető kapuáramköröket!
2. Ismertessük a TTL NAND kapu felépítését és működését!
3. Ismertessük a többcmitteres tranzisztoros bemeneti fokozat működését!
4. Ismertessük a totem-pole és tri-state kimeneti fokozatok felépítését és működését!
5. Milyen módon kapcsolhatók össze a kapuáramkörök kimenetei?
6. Értelmezzük a kapuáramkörök jellemzőit!
7. Csoportositsuk működési sebesség és disszipáció szempontjából a TTL áramkör családokat!

## 4. Tárolók

A tárolóáramkörök, más néven flip-flopok, kétállapotú áramkörök. Két kimeneti állapotuknak megfelelően 1 bit információ tárolására alkalmasak. Az információ a tároló megfelelő vezérlésével beírható a tárolóba, ill. a vezérlés megváltoztatásával kitörölhető. Az általuk megvalósított funkciótól függően a tárolókat a következő logikai csoporthoz soroljuk.

- R-S típusú tárolók,
- inverz R-S, vagy másképpen ( $\bar{R} - \bar{S}$ ) tárolók,
- J-K tárolók,
- T (trigger) tárolók,
- D (delay) tárolók.

Az **R-S tárolók** S (set – írás) bemenete a beírásra, vagyis egy olyan kimeneti állapot létrehozására való, amikor a Q kimeneten logikai 1 szint jön létre. Ez jelzi az 1 információs bit tárolását. Az R (reset – törlés) bemenetre adott vezérlés a Q kimeneten 0 állapotot hoz létre, a flip-flop törlődik, tehát a továbbiakban 0 információs bitet tárol. Az R-S tárolók belső felépítését, jelképi jelölését és vezérlési táblázatát a 4.1. ábra mutatja.

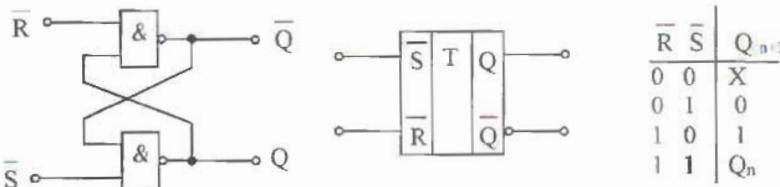


4.1. ábra. R-S tárolók belső felépítése, jelképi jelölése és vezérlési táblázata

A kapcsolási rajzon jól látható, hogy a kapuk invertálása miatt a két kimenet egymásnak negáltja. Az áramkör működésének elemzéséhez tételezzük fel, hogy a Q kimeneten 1 állapot van (ez az előző állapot) és az R-S bemenetekre 0-0 logikai szintet adunk. A NOR függvénynek megfelelően az 1. kapu kimenetén 1 szint lesz (marad), a 2. kapu kimenetén 0 szint lesz (marad). Ezt úgy is megfogalmazhatjuk, hogy az R-S flip-flop 0-0 vezérlés hatására megtartja előző állapotát. Tchát a következő állapot megegyezik az előző állapottal.

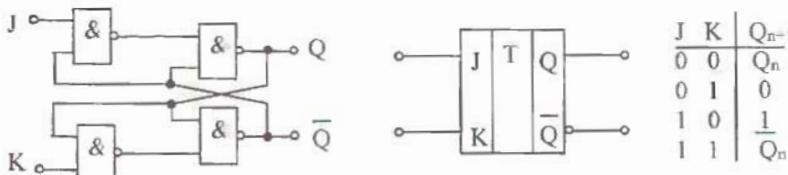
Az R-S bemenetre 0-1 vezérlést adva az 1. NOR kapu kimenete 1 szintre vált, mert  $Q = \overline{\overline{Q} + R} = \overline{0 + 0} = 1$ . Tehát az S bemenetre adott 1 beírja a tárolót. Megfordítva a vezérlést ( $R = 1, S = 0$ ) ellentétes változást hoz létre, a Q kimenet 0 szintű lesz. Ezzel a vezérléssel töröltük a tárolót. Érdekes helyzetet idéz elő az  $R = 1, S = 1$  vezérlés. A NOR függvény szerint ilyenkor minden két kapu 0 kimeneti szintet adna a kimenetén, ami ebben a kapcsolásban nem lehetséges. Ilyenkor véletlenszerűen alakul a kimeneti állapot, ezért ez a vezérlési állapot nem megengedett. Logikailag sem értelmezhető az 1-1 vezérlés, mert egyszerre jelentné a beírás és a törlés vezérlését. A tároló vezérlésénél lehetséges eseteit és a vezérlések hatására létrejövő kimeneti állapotokat a **vezérlési táblázat** (igazságátlábazat) foglalja össze a 4.1. ábrán. A jelképi jelölésben szereplő T betű az áramkör funkciója. Jelentése: tároló.

Az **inverz R-S ( $\overline{R} - \overline{S}$ ) tároló** minden tekintetben fordított működésű az R-S tárolóhoz képest. A 4.2. ábra kapcsolási rajzán kívül a jelképi jelölését és a vezérlési táblázatát mutatja. A tároló beírása és törlése 0 szinttel lehetséges, a tiltott vezérlés pedig a 0-0.



4.2. ábra. Az  $\overline{R} - \overline{S}$  flip-flop belsejének felépítése, jelképi jelölése és vezérlési táblázata

A **J-K tároló** belsejű áramköri kialakítása olyan, hogy nincs tiltott vezérlési állapota. A J bemeneten keresztül a tároló beírható, a K bemeneten keresztül törölhető. Kapcsolási rajzát, jelképi jelölését és vezérlési táblázatát a 4.3. ábra mutatja.

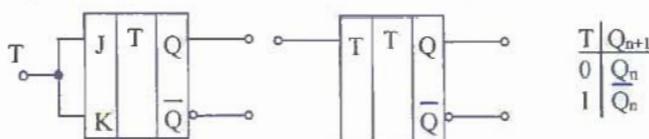


4.3. ábra. A J-K tároló felépítése, jelképi jelölése és vezérlési táblázata

A tároló bemenetén lévő két NAND kapunak az a feladata, hogy az inverz R-S tároló 0-0 tiltott vezérlési állapotát kiküszöbölje. Ha a J-K bemenetekre 0-0 vezérlés kerül, akkor a NAND kapuk másik bemenetére kerülő vezérléstől függetlenül a kimenetükön 1 szint jelenik meg. Ez a NAND kapukat követő inverz R-S tároló számára

nem tiltott vezérlés, hatására megtartja előző kimeneti állapotát. A J-K bemenetekre adott 1-1 vezérlés hatására a NAND kapuk invertálják az éppen aktuális  $Q - \bar{Q}$  kimeneti állapotokat, ezért az inverz R-S tároló kimeneti állapota megváltozik. Ezt úgy fogalmazzuk meg, hogy a J-K tároló az 1-1 vezérlés hatására billen (triggerrel). A beírás a J bemenetre adott 1 szinttel, a törlés pedig a K bemenetre adott 1 szinttel valósítható meg.

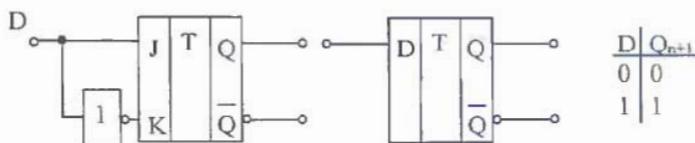
**A T típusú tároló** egy összekötött bemenetű J-K tároló, amint azt a 4.4. ábra szemlélteti, a jelképi jelöléssel és a vezérlési táblázattal együtt.



4.4. ábra. A T tároló felépítése, jelképi jelölése és vezérlési táblázata

A T tároló bemenetét (közösített bemenet) T triggerbemenetnek nevezzük. A tároló vezérlési táblázata tulajdonképpen a J-K tároló első és utolsó sora. A T bemenetet 0 szinttel vezérelve a kimenet megtartja az előző állapotát. A bemeneti 1 vezérlésre a tároló triggerrel, az előző állapot ellenére változik.

A **D flip-flop** kimenetén olyan logikai érték jelenik meg, amilyet a bemenetérc kapcsoltunk. Ezt a funkciót egy olyan J-K tároló képes ellátni amelynek bemenetei ellentétes vezérlést kapnak. Ezt a két bemenet közé kapcsolt inverter teszi lehetővé. Elemezve az áramkör működését látszik, hogy a D tároló a J-K igazságátlábazatának a középső két sorát valósítja meg. A D tároló felépítését, jelképi jelölését és a vezérlési táblázatot a 4.5. ábra mutatja.



4.5. ábra. A D tároló felépítése, jelképi jelölése és vezérlési táblázata

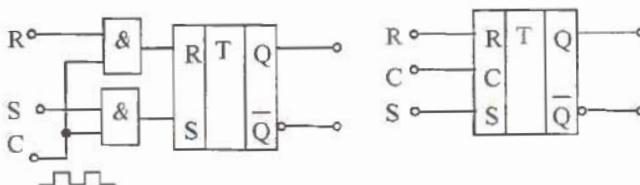
A tárolókat a vezérlésük jellegétől függően is csoportosíthatjuk. **A vezérlés jellegétől függő csoportok:**

- sztatikus vezérlésű tárolók,
- dinamikus vezérlésű tárolók. Ezen belül
  - kapuzott tárolók,
  - mester-szolga tárolók,
  - élvezérelt tárolók.

**A sztatikus vezérlésű tárolók** kimenetén a bemeneti vezérlés hatására a kimeneti állapot azonnal megjelenik. Ilyen tárolók voltak az előzőekben megismertek. A sztatikus vezérlésű tárolók hátránya, hogy a bemenetükre érkező esetleges zavarok szintén megváltoztatják a kimenet állapotát: a sztatikus tároló átlátszó a zavarok szempontjából.

**A dinamikus vezérlésű tárolók** kimcneti állapotának megváltozását **órajel** (Clock) ütömezi.

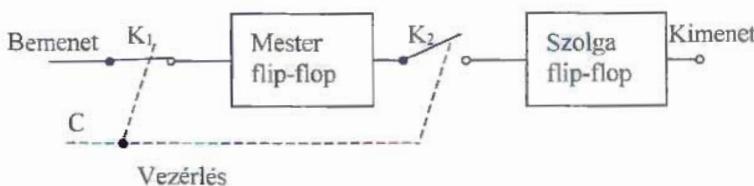
A legegyszerűbb dinamikus tároló a **kapuzott tároló**. Bármelyik tárolóból kialakítható kapuzott tároló, ha a sztatikus bemeneteit ÉS kapukon keresztül vezéreljük és a kapukat órajellel tiltjuk vagy engedélyezzük. A 4.6. ábra példaként egy R-S tároló kapuzását mutatja.



4.6. ábra. Kapuzott R-S tároló és jelképi jelölése

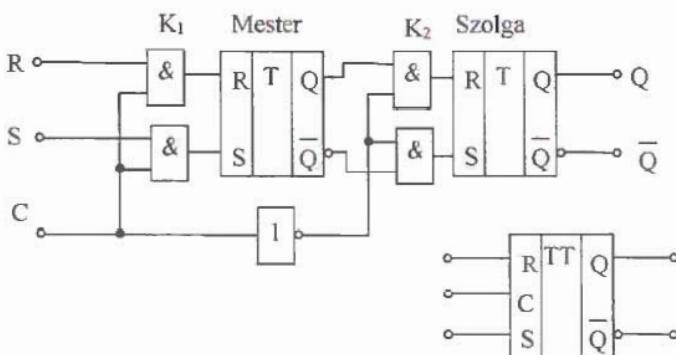
Az órajel 0 szintje mellett az ÉS kapuk kimenetén az R-S vezérléstől függetlenül 0 szint van. Ha az órajel 1 szintű, akkor az ÉS kapu kimenetén megjelenhet az R-S vezérlés. A tároló átlátszósága az órajel időtartamára csökken.

A **mester-szolga (master-slave)** **tároló** kétütemű tároló, amelynek elvi felépítését a 4.7. ábra szemlélteti.



4.7. ábra. Mester-szolga elv

Az első ütemben a  $K_1$  kapcsoló zárt, a  $K_2$  pedig nyitott állapotban van. A bemenetre adott vezérlés a mester tárolóra hatásos, de a kimeneten nem jelenik meg a vezérlés következménye a nyitott  $K_2$  miatt. A második ütemben a  $K_1$  nyitott a  $K_2$  zárt, ezért a mesterben lévő információ átirányít a szolgába és megjelenik a kimeneten. Ez a felépítés elvileg megakadályozza, hogy a bemenetre kerülő zavarok hatása megjelenjen a kimeneten. Az elv megvalósítható bármilyen tárolóval. A 4.8. ábra példaként a mester-szolga R-S tároló felépítését és jelképi jelölését mutatja.

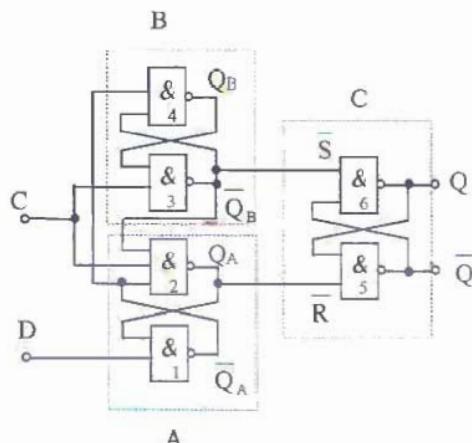


4.8. ábra. Mester-szolga R-S tároló

A kapcsolók ÉS kapuk, amelyeket az inverter miatt ellenütemben vezérlünk az órajellel. A mester-szolga tároló elvileg nem átlátszó. Az órajel azonban nem ideális négyzetöggel, ezért a fel- és lefutási ideje alatt egy rövid időre a bemeneten lévő kapuk és a két tároló közötti kapuk is nyitva vannak. Ez alatt a rövid idő alatt a bemenetet érő zavarok hatása megjelenik a kimeneten.

**Az élvezérelt tárolók** vezérlési lehetősége az órajel változásának időpontjához kötődik. Ez vagy az órajel  $0 \rightarrow 1$  átmenete, vagy az  $1 \rightarrow 0$  átmenet. Az első esetben fel-futó ére működő **pozitív élvezérelt** a flip-flop, a második esetben pedig lefutó ére működő **negatív élvezérelt**.

A legegyszerűbb belső felépítésű élvezérelt tároló a D tároló, ezért ennek működését elemezzük, pozitív élvczérlés esetén, a 4.9. ábra alapján.

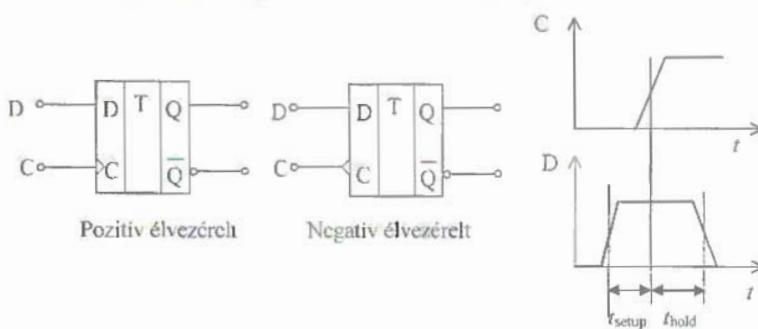


4.9. ábra. Élvezérelt D tároló

A tároló az A, B és C inverz R-S flip-flopokból áll. Alaphelyzetben, amikor az órajel 0 szintű, akkor ez a 2. és 3. NAND kapu kimenetét letiltja, ezért a C flip-flop  $\bar{R}_C - \bar{S}_C$  bemenetei 1-1 vezérlést kapnak. Az ilyen vezérlés hatására a kimenet megtartja előző állapotát az órajel 0 szintje mellett, tehát nem lehet hatásos a D bemenet vezérlése. A D bemenet állapota viszont előkészíthet egy billenést az órajel felfutó élénk megérkezése előtt:

- ha  $D = 0$ , akkor a  $Q_A$  kimeneten 1 szint, a  $Q_B$  kimeneten 0 szint van. A  $Q_A$  kimenet 1 szintje előkészíti az A flip-flopot a billcnésre, ami az órajel felfutó élénél bekövetkezik, mert a 2. kapu valamennyi bemenetén 1 szint lesz. A B tároló nem billen, mert a  $Q_B$  kimenet 0 szintje miatt az órajel felfutó éle hatástalan. A C tároló bemenetein tchát az órajel felfutásakor  $R_C = 0, S_C = 1$  vezérlés lesz, ezért a tároló törlődik. Végeredményben a D bemenetre 0-át adva a felfutó él hatására a D tároló kimenetén 0 jelent meg. Ugyanakkor a 2. kapu kimenetén a billenéskor megjelenő 0 szint a továbbiakban tiltja az 1. kaput, ezért a D bemenet esetleges változása már hatástalan lesz.
- ha az órajel felfutó élénck megérkezése előtt  $D = 1$  vezérlést adunk a tárolónak, akkor a  $Q_A$  kimenetn 0 szint lesz, mert az órajel  $C = 0$  miatt a 2. kapu kimenete 1 szintű. A  $Q_A$  0 szintje a  $Q_B$  kimenetet 1 szintre állítja. Az órajel felfutó élénk megérkezéskor ezért a 3. kapu 1-1 vezérlést kap, kimenete 0 szintre vált. A C tároló vezérlése tehát  $S_C = 0, R_C = 1$ . Ennek hatására a tároló 1-be íródik. Végeredményben tchát a D bemenetre 1-et adva, a tároló az órajel felfutásakor beíródik. Ugyanakkor a 3. kapu billenés utáni kimeneti 0 szintje a 2. kapu bemenetére kerül és letiltja az A tárolót. Ilyenkor már a D bemenet esetleges változása hatástalan lenne.

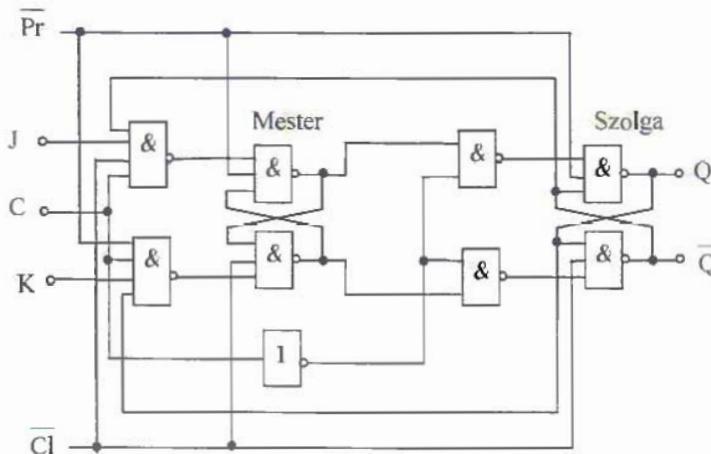
A tárolók valamennyi logikai típusából készítenek elvészérelt tárolókat, mert ez a vezérlési mód teszi lehetővé a legrövidebb idcijű átlátszóságot. Külön előnye, hogy az ilyen tárolókkal felépített rendszerekben a változások jól meghatározott időpontokban (fel- vagy lefutó él) következnak be. A helyes működés feltétele a gyártók által megadott vezérlési idők biztosítása. A 4.10. ábra az elvészérelt tárolók jelképi jelölését és a helyes működéshez szükséges időtartamokat mutatja.



**4.10. ábra.** Az elvészérelt tárolók jelképi jelölése és a vezérlési idők

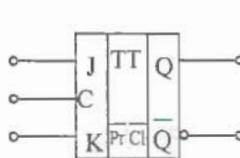
A  $t_{\text{setup}}$  az előkészítési idő. Legalább ennyi idővel az órajel élének megérkezése előtt a bemeneten kell lennie a vezérlésnek, és a  $t_{\text{hold}}$  tartási ideig még ott kell maradnia az órajel felfutása után. A  $t_{\text{setup}}$  és  $t_{\text{hold}}$  időkön kívül a katalógusok megadják az órajel legnagyobb megengedett felfutási idejét is. Ennél nagyobb felfutási idő bizonytalanná teszi az áramkör működését.

A dinamikus vezérlésű tárolókat gyakran úgy készítik el, hogy rendelkezzenek sztatikus vezérlő bemenetekkel is. Ezek az órajeltől független töröl és beíró bemenetek a tárolókból felépített áramkörök alapállapotát állítják be. A 4.11. ábra egy sztatikus bemenetekkel ellátott mester-szolga J-K tároló belső felépítését mutatja.



4.11. ábra. Sztatikus bemenetekkel ellátott J-K tároló

Az ábrán jelölt  $\overline{\text{Pr}}$  (preset – beírás) és  $\overline{\text{Cl}}$  (clear – törles) bemenetek az órajeltől függetlenül, közvetlenül hatnak a belső  $\overline{\text{R}} - \overline{\text{S}}$  tárolókra, így tetszőleges időpontban beállítható a kimeneti állapot. A tároló jelképi jelölését és működési táblázatát a 4.12. ábra mutatja.



$\overline{\text{Pr}}$	$\overline{\text{Cl}}$	J	K	$\overline{\text{Q}_{n+1}}$
0	0	x	x	x
0	1	x	x	1
1	0	x	x	0
1	1	x	x	$\overline{\text{Q}_n}$
1	1	0	0	$\overline{\text{Q}_n}$
1	1	0	1	0
1	1	1	0	1
1	1	1	1	$\overline{\text{Q}_n}$

4.12. ábra. Sztatikus bemenettel is rendelkező tároló és működési táblázata

Gyakran előfordul, hogy a sztatikus bemenetek 0 szintre hatásosak, tehát egy-egy beépített inverteren keresztül vezérlik a tárolót. A bemenetek jelölése ebben az esetben:  $\bar{Cl}$  és  $\bar{Pr}$ . A negált vezérlés a zavarvédelettség szempontjából kedvező. A zavarok mindenkor feszültségimpulzusok, amelyek amplitúdója elérheti az 1 szintnek megfelelő feszültségtartományt. Az 1 szintre hatásos bemeneteken ez téves vezérlést okozhat. A negált szintű, 0-ra hatásos bemenetek viszont érzéketlenek a zavarokra.

### **Ellenőrző kérdések**

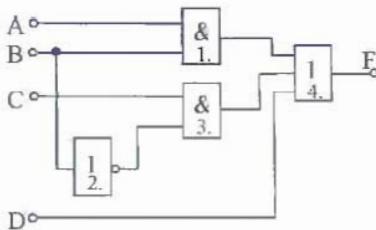
1. Csoportositsa a tárolókat!
2. Ismertessük a NOR kapukból felépített R-S tároló működését és vezérlési táblázatát!
3. Ismertessük a J-K tároló működését és vezérlési táblázatát!
4. Ismertessük az élvezérlés fogalmát, megvalósítási módját, alkalmazási területeit!
5. Ismertessük a mester-szolga elv lényegét és megvalósítási módját!

## 5. LOGIKAI HÁLÓZATOK ANALÍZISE ÉS REALIZÁLÁSA

A logikai hálózatok kapuáramkörökből felépített kombinációs hálózatok, ill. kapuáramkörökből és tárolókból álló szekvenciális hálózatok. Egy hálózat működésének elemzését analízisnek, míg a hálózat tervezését és megvalósítását realizálásnak nevezzük.

### 5.1. Kombinációs hálózatok

A kombinációs hálózatok analizálása a logikai algebra függvénycinek, alaptételeinek és törvényeinek felhasználásával történik. Az analízis célja a hálózat kimeneti függvényének meghatározása. Az 5.1. ábra egy NEM-ÉS-VAGY kapukból álló kombinációs hálózatot mutat. Az ilyen felépítésű hálózatot röviden NÉV rendszerű hálózatnak nevezzük.



5.1. ábra. NÉV rendszerű hálózat

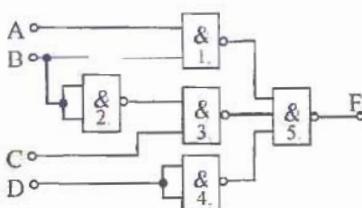
A hálózat a kapuk által megvalósított függvények felírásával elemezhető a bementerektől a kimenet felé haladva. Az 1. sorszámú ÉS kapu kimenetén lévő függvény:  $F_1 = B \cdot A$ .

A 2. inverter kimenetén:  $F_2 = \bar{B}$ .

A 3. ÉS kapu kimenetén:  $F_3 = C \cdot F_2 = C \cdot \bar{B}$ .

A 4. VAGY kapu kimenetén:  $F^4 = F_1 + F_3 + D = B \cdot A + C \cdot \bar{B} + D$ .

A NAND kapukból felépített hálózatok elemzésének módszert az 5.2. ábra hálózatán végezzük.



**5.2.** ábra. NAND kapukból felépített kombinációs hálózat

Az egyes kapuk kimenetein megvalósított függvények:

$$F_1 = \overline{B \cdot A},$$

$F_2 = \overline{B \cdot B} = \overline{B}$ , a NAND kapu tehát összekötött bemenetekkel inverter,

$$F_3 = \overline{C \cdot F_2} = \overline{C \cdot \overline{B}},$$

$$F_4 = \overline{D \cdot D} = \overline{D},$$

$$F = \overline{F_1 \cdot F_3 \cdot F_4} = \overline{\overline{B \cdot A} \cdot \overline{C \cdot \overline{B}}} \cdot \overline{D}.$$

A De-Morgan-tételt kétszer alkalmazva:  $F^4 = B \cdot A + C \cdot \overline{B} + D$ .

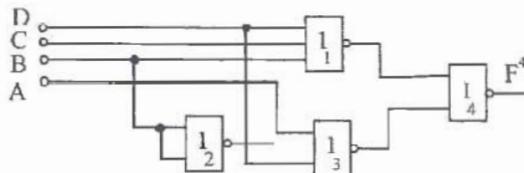
A csak NAND kapuból felépített hálózat is megvalósítja tehát ugyanazt a függvényt, amit a NÉV rendszer. A hálózatot és az általa megvalósított függvényt összehasonlítva megállapíthatjuk, hogy

- az 5. kapu – annak ellenére, hogy NAND kapu – a változócsoporthoz között tulajdonképpen VAGY kapcsolatot valósít meg,
- az 1. és 3. kapuk – amelyek szintén NAND kapuk – a függvény szerint tulajdonképpen ÉS kapcsolatot hoznak létre a bemeneteikre kerülő változók között.

Általánosan is megfogalmazhatjuk ezeket a következtetéseket, ha a hálózatot egy olyan többszintű hálózatnak tekintjük, amelyben az 1. szint a kimenethez legközelebb eső. Így az **5.2.** ábra hálózatában az 5. kapu az 1. szinten, az 1., 3., 4. kapuk a 2. szinten és a 2. kapu pedig a 3. szinten van. Az előzőeket a szintek segítségével megfogalmazva: páratlan szinten a NAND kapu VAGY kapcsolatot valósít meg a kimeneti függvényben, páros szinten pedig ÉS kapcsolatot.

Összefüggés található a változók értéke és bevezetésük szintje között is. A páros szinten bevezetett változókat (pl. A, B, C, D a 2. szinten) a hálózat nem változtatja meg, a páratlan szinten bevezetett változókat (B változó a 3. szinten) viszont negálja.

Az **5.3.** ábra áramköre NOR kapuból áll. A hálózat analízise az előzőkhöz hasonlóan történik.



**5.3.** ábra. NOR kapukból álló kombinációs hálózat

$$F_1 = \overline{D + C + B},$$

$$F_2 = \overline{B},$$

$$F_3 = \overline{D + F_2 + A} = \overline{D + \overline{B} + A},$$

$$F_4 = \overline{F_1 + F_3} = \overline{\overline{D + C + B} + \overline{D + \overline{B} + A}}.$$

A De-Morgan-tételt alkalmazva:  $F^4 = (D + C + B) \cdot (D + B + A)$ .

Ha az eredményül kapott függvényt az összehasonlíthatóság miatt szabályos alakra hozzuk és átalakítjuk, akkor ugyanazt a függvényt kapjuk, amit a NÉV és NAND rendszernél.

A hálózatot itt is színtezve a megismertek szerint, a függvény és a hálózat összehasonlításából levonható következtetés:

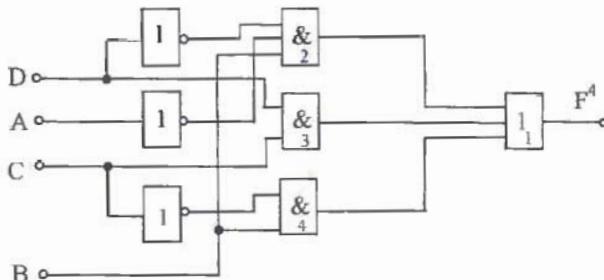
- páratlan szinten a NOR kapu ÉS kapcsolatot valósít meg, páros szinten pedig VAGY kapcsolatot,
- a páros szinten bevezetett változókat a hálózat nem változtatja meg, a páratlan szinten bevezetetteket negálja.

Az előző három feladat az elemzés módszerének megismerését célozta. Ezen túlmenően azonban a feladatok egy fontos következtetés levonására is lehetőséget adtak, ugyanaz a függvény megvalósítható NÉV, NAND és NOR hálózattal is. Ez azt is jelenti, hogy bármelyik függvény megvalósítható csak NÉV kapuk, vagy csak NAND kapuk, vagy csak NOR kapuk felhasználásával. Azokat a rendszereket, amelyekkel bármilyen függvény megvalósítható, **funkcionálisan teljes rendszernek** nevezük. Így tehát a feladatokban használt rendszerök funkcionálisan teljes rendszerek.

A **kombinációs hálózatok realizálása** egy logikai függvény megvalósítását jelenti kapuáramkörök felhasználásával. A realizálás során törkendni kell a lehető legkevesebb kapuáramkör felhasználására, amit úgy érhetünk el, ha a megvalósítandó függvényt a megismert módszerek valamelyikével egyszerűsítjük. A minimalizált függvényt az egyik funkcionálisan teljes rendszerrel valósítjuk meg.

A legegyszerűbb a **NÉV rendszerben való realizálás**, hátránya azonban, hogy három különböző kaputípust igényel, és ezért a különböző integrált áramköri tokok kihasználása valószínűleg nem lesz optimális. A realizálás módszerét az

$F^4 = \overline{D} \cdot B \cdot \overline{A} + D \cdot C + \overline{C} \cdot B$  függvény megvalósításával ismerjük meg, az eredményül kapott hálózatot pedig az 5.4. ábrán rajzoljuk meg.



5.4. ábra. Logikai függvény realizálása NÉV rendszerben

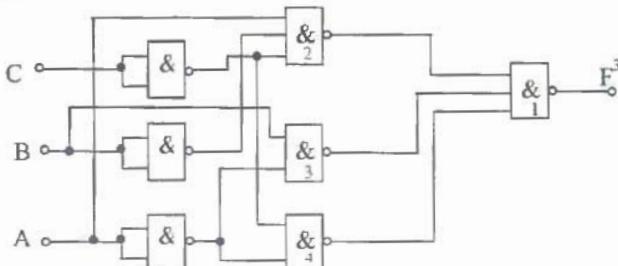
A függvény három változócsoporthálózatból áll, amelyek VAGY kapcsolatban vannak egymással. Ennek a megvalósításához egy hárombemenetű VAGY kapura van szükség. Ez a rajzon az 1. kapu. A három bemenetre külön-külön elő kell állítani a változócsoporthálózatokban lévő függvénykapcsolatokat. Az egyik bemenetre ehhez egy hárombemenetű ÉS kaput kell kötni (2. kapu), a másik bemenetre egy kétbemenetűt (3. kapu), a harmadik bemenetre szintén kétbemenetűt (4. kapu). A 2. kapu bemenetére a  $D$  és az  $A$  változókat, a 4. kapu bemenetére pedig a  $C$  változót egy-egy inverteren keresztül kell vezetni. A többi változó közvetlenül az ÉS kapuk bemeneteire kapcsolódik. Hasonló módszerrel realizálható bármilyen logikai függvény NÉV rendszerben.

A logikai függvények NAND kapus, funkcionálisan teljes rendszerrel is realizálhatók.

A **NAND kapus realizálás** módszere azokat a szabályokat használja fel, amelyeket az 5.2. ábra hálózatának elemzése során megismertünk. Pl. az

$F^3 = \overline{C} \cdot \overline{B} \cdot A + B \cdot \overline{A} + \overline{C} \cdot \overline{A}$  függvény realizálása a következőképpen történik:

- a függvény három változócsoporthálózat közötti VAGY kapcsolatot ír le. Ezt az első szinten egy hárombemenetű NAND kapuval lehet megvalósítani, mert a NAND kapu páratlan szinten VAGY kapcsolatot valósít meg. Az 5.5. ábrán ez az 1. kapu.



5.5. ábra. Függvényrealizálás NAND kapukkal

- az első szinten elhelyezett NAND kapu bemeneteire a függvény szerinti változók ÉS kapcsolatát kell megvalósítani a következő, tehát páros szinten. Páros szinten a NAND kapu ÉS kapcsolatot valósít meg, ezért a 2. kapu a  $\bar{C} \cdot \bar{B} \cdot A$ , a 3. kapu a  $B \cdot \bar{A}$ , a 4. kapu pedig a  $\bar{C} \cdot A$  függvényt valósítja meg. Több ÉS, ill. VAGY kapcsolat nincs a függvényben, ezért több szintre nincs szükség.
- a kimeneten megjelenő függvényben a 2. kapu C és B változói negált értékkel, az A változó pedig ponált értékkel szerepel. Páros szinten lévő kapuba vezetjük be a változókat, ezért a realizálási szabály szerint olyan értékkel kell ezeket bevezetni, amilyen értékkel a függvényben szerepelnek. A C és a B változót negálva, az A változót pedig ponálva. A negált változókat inverterrel állítjuk elő. A 3. kapu B és A változóját ponálva, a 4. kapu C és A változóit pedig negálva vezetjük be a hálózatba. A változók negálását természetesen NAND kapukból készített inverterrel végezzük el, hiszen más kaput a NAND rendszerben nem használhatunk.

A bemeneten elhelyezett inverterek szintjét nem tekintjük 3. szintnek, mert a digitális rendszerben általában rendelkezésünkre áll a változók ponált és negált értéke is, tehát nem szükséges inverterekkel előállítani. A megvalósított hálózatot ezért kétszintű hálózatnak tekinthetjük. Általánosságban is igaz, hogy bármilyen logikai függvény NAND rendszerben kétszintű hálózattal megvalósítható. Ez egyszerűen belátható, hiszen egy függvényben – a negációt leszámítva – csak VAGY és ÉS kapcsolat lehet, amelyek az 1. és a 2. szinten megvalósíthatók. Ez a megállapítás akkor nem általánosítható, ha a realizáláshoz csak kétbemenetű kapukat használhatunk. (A leggyakrabban használt SN 7400 típusú integrált áramkör 4 db kétbemenetű NAND kaput tartalmaz.)

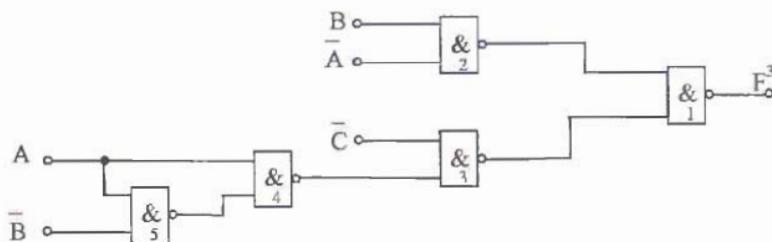
**A kétbemenetű NAND kapukkal való realizálás** módszerét a függvény határozza meg. Ha lehetséges, a függvényt átalakítjuk kétszintű realizáláshoz alkalmas formába, ha nem, akkor az inverterekkel való szinteltolás módszerét használjuk.

Az átalakítás minden olyan kiemelés, amelynek eredményeként csak két változó vagy változócsoport között kell függvénykapcsolatot megvalósítani. Az előző feladat függvényén elvégezhető ez az átalakítás:

$$F^3 = \bar{C} \cdot \bar{B} \cdot A + B \cdot \bar{A} + \bar{C} \cdot A = \bar{C} \cdot (\bar{B} \cdot A + \bar{A}) + B \cdot \bar{A}.$$

Az átalakított függvény realizálásához csak kétbemenetű kapura van szükség:

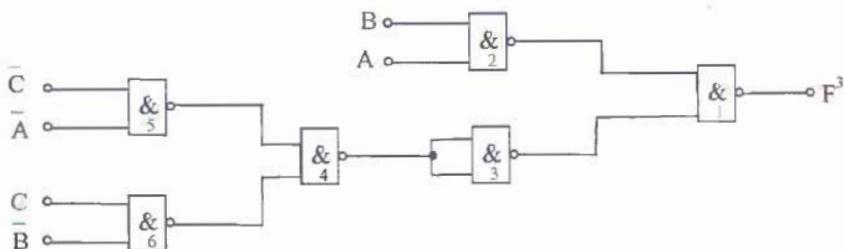
- az első szinten a kétbemenetű NAND kapu a  $B \cdot \bar{A}$ , valamint az első csoporthoz kötött VAGY kapcsolatot valósítja meg, amint azt az **5.6.** ábra mutatja.



5.6. ábra. Függvényrealizálás körbemenetű NAND kapukkal

- a második szinten a  $\bar{C}$ , valamint a zárójeles kifejezés és a  $B$ ,  $\bar{A}$  változók közötti ÉS kapcsolat valósul meg,
- a harmadik szinten NAND kapuja VAGY kapcsolatot realizál:  $\bar{B} \cdot A + \bar{A}$ ,
- a negyedik szinten a  $\bar{B} \cdot A$  függvénykapcsolat valósul meg,
- a változók bevezetése páros szinten olyan értékkel történik, amilyennel a függvényben szerepelnek: 2. és 3. kapu  $B, \bar{A}, \bar{C}$ , 5. kapu  $A, \bar{B}$ . Páratlan szinten a függvényhöz képest negált értékkel: 4. kapu,  $A$  változó.

A megvalósítandó függvény nem minden esetben alakítható át kiemeléssel. Ilyen függvény pl. az  $F^3 = B \cdot A + \bar{C} \cdot \bar{A} + C \cdot \bar{B}$ , amelynek megvalósítását mutatja az 5.7. ábra.



5.7. ábra. Függvényrealizálás szinteltolással

A függvényben kiemelés nem végezhető, de az asszociativitást felhasználva a változócsoporthatákat tetszés szerint csoportosíthatjuk. Egy lehetőség a csoportosításra:

$$F^4 = B \cdot A + (\bar{C} \cdot \bar{A} + C \cdot \bar{B}),$$

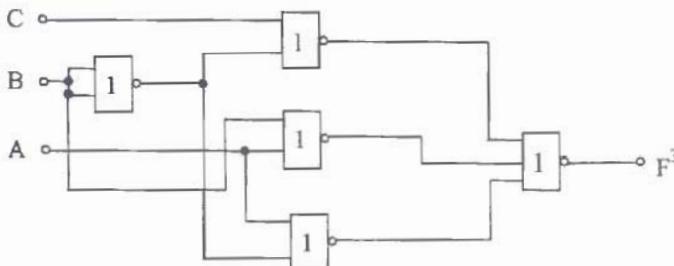
- az első szinten lévő NAND kapu a  $B \cdot A$ , valamint a zárójeles kifejezés közötti NAND kapcsolatot realizálja,
- a második szinten a NAND kapu ÉS kapcsolatot valósítana meg, de a függvényben nem erre van szükség. Ezért ebből a NAND kapuból invertert csinálnunk, ezzel a megvalósítást a következő szintről toljuk. Ezen a harmadik szinten már megvalósítható a zárójelben lévő VAGY kapcsolat,
- a negyedik szinten a NAND kapuk az ÉS kapcsolatokat realizálják.

Az eljárás helyessége analízissel igazolható.

A megoldott feladatokból látható, hogy a kapubemenetek számának korlátozásával növekedett a szintek száma. Az alkalmazások egy részében nem engedhető meg a többszintű hálózat, mert a szintek számának növekedésével nő a hálózat be- meneti és kimenete közötti jelkésleltetési idő, tehát a hálózat lassúbb lesz. Az eredő jelkésleltetés a realizált hálózat legtöbb kaput tartalmazó ágában lévő kapuk késleltetési idejének összegc. Példaként hasonlítsuk össze az 5.5. és az 5.6. ábra áramköreit feltételezve, hogy a kapuk késleltetési ideje  $t_{pd} = 10$  ns. Az 5.5. ábra kétszintű hálózatánál az eredő jelkésleltetés:  $t_{pde} = 20$  ns. Az 5.6. ábra áramkörénél:  $t_{pde} = t_{pd5} + t_{pd4} + t_{pd3} + t_{pd1} = 40$  ns.

**A NOR kapukkal való függvényrealizálás** az 5.3. ábra áramkörének analíziséből levont következtetések alapján történik. Az  $F^4 = (C + \bar{B}) \cdot (B + A) \cdot (\bar{B} + A)$ ,

- az első szinten – páratlan szint – a hárombemenetű NOR kapu ÉS kapcsolatot valósít meg a zárójelben lévő mennyiségek között,
- a második szinten – páros szint – a NOR kapuk a zárójelben lévő VAGY kapcsolatokat valósítják meg,
- a változók bevezetésére páros szinten kerül sor, ezért olyan értékkel kell bevezetni mindegyiket, amilyen értékkel a függvényben szerepelnek. A megvalósított hálózatot az 5.8. ábra mutatja.



5.8. ábra. NOR kapus realizálás

A kétbemenetű NOR kapukkal való realizálás a NAND rendszernél megismert módszerrel végezhető.

A feladatokból látható, hogy **NAND kapukkal** az olyan függvényalakok realizálhatók egyszerűen, amelyekben a változócsoporthak VAGY kapcsolatban vannak egymással, a változócsoporthakban lévő változókat pedig ÉS kapcsolatok kötik össze. Az ilyen függvényhez **diszjunktív szabályos alakú függvények** egyszerűsítésével jutunk.

Hasonló gondolatmenettel látható be, hogy **NOR rendszerrel a konjuktív normál alakból** egyszerűsített függvények realizálhatók közvetlenül.

A NÉV rendszerrel bármilyen függvényalak realizálható.

A függvényegyszerűsítés és átalakítás, valamint a funkcionálisan teljes rendszerekkel való függvényrealizálás összefoglalására oldjuk meg a következő feladatot!

#### 4. feladat

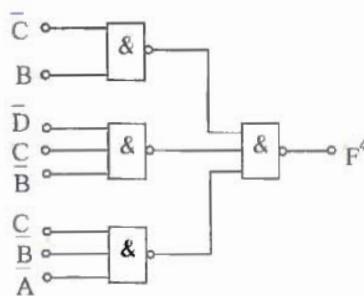
Realizáljuk az  $F^4 = \Sigma^4(2,3,4,5,10,11,12)$  függvényt NAND, NOR és NÉV rendszerben!

#### A 4. feladat megoldása

A NAND kapus realizáláshoz a megadott diszjunktív normál alakot kell egyszerűsíteni. Az egyszerűsítést grafikusan végezzük a négyváltozós minterm táblát használva. Ezt mutatja az 5.9. ábra.

				B			
				1	3	1	2
				0	1	4	5
D							
				7	6		
				12	13	15	14
				8	9	11	10

5.9. ábra. A 4. feladat minterm táblája



5.10. ábra. A 4. feladat NAND hálózata

Kiolvasva a bejelölt hurkokat, adódik az egyszerűsített függvény:

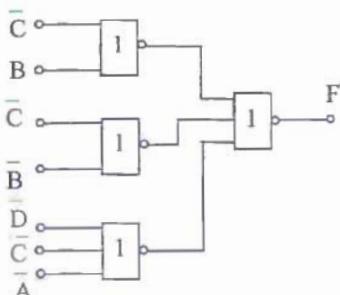
$$F^4 = \bar{C} \cdot B + \bar{D} \cdot C \cdot \bar{B} + C \cdot \bar{B} \cdot \bar{A}$$

A függvényt megvalósító hálózatot az 5.10. ábra mutatja.

A NOR kapus realizáláshoz a diszjunktív függvényt át kell alakítani konjuktív szabályos alakra. A grafikus átalakításhoz szükséges maxtermtábla az 5.11. ábrán látható.

				B		
				0	3	2
				0	1	4
D						
				5	0	6
				12	0	13
				8	9	11

5.11. ábra. A 4. feladat maxtermtáblája



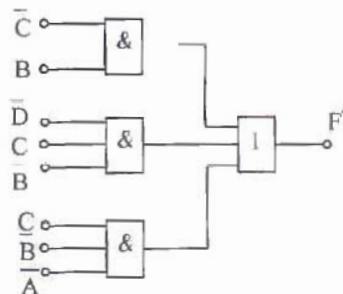
5.12. ábra. A 4. feladat NOR hálózata

A minterm táblából átirrt 0 termeket hurkolva és kiolvasva az egyszerűsített függvényt:

$$F^4 = (C + B) \cdot (\bar{C} + \bar{B}) \cdot (\bar{D} + \bar{C} + \bar{A}).$$

A NOR kapukkal realizált hálózatot az 5.12. ábra mutatja.

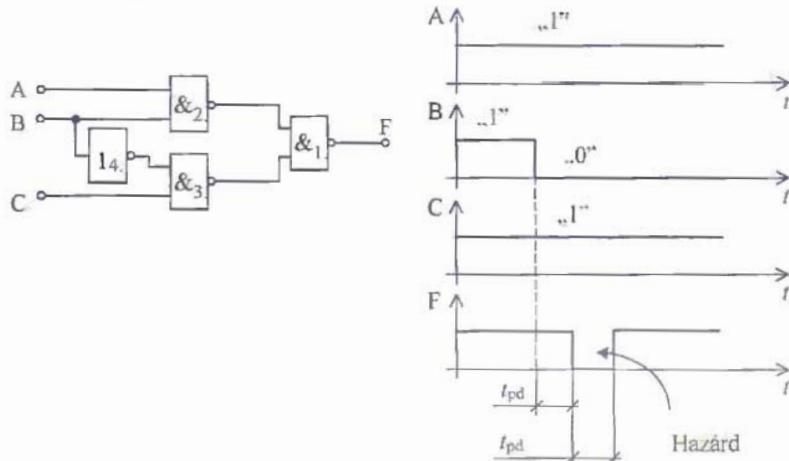
A NÉV rendszerttel való realizáláshoz bármelyik egyszerűsített függvényalak felhasználható. Az 5.13. ábrán a diszjunktív alakból egyszerűsített függvényt realizáló hálózat látható.



5.13. ábra. A 4. feladat NÉV áramköre

A kombinációs hálózatok realizálásánál nem vettük figyelembe azt a tényt, hogy kapuáramkörök késleltetési idővel is rendelkeznek, ezért a bemeneti változók értékének megváltozása a kimeneten csak időkésessel jelentkezik. Az időkésés egyes esetekben átmenetileg hibás kimeneti állapotot is eredményezhet, amit **hazardna** nevezünk. Egyszerűbb esetben **sztatikus hazard** lép fel, de elsősorban bonyolultabb hálózatoknál előfordulhat **dinamikus hazard** is.

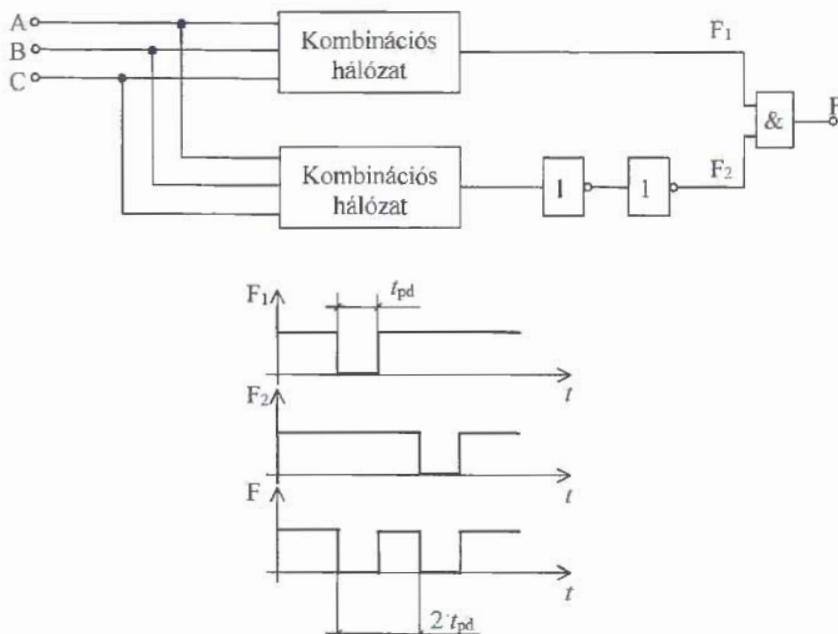
A sztatikus hazard keletkezése az 5.14. ábrán látható.



5.14. ábra. A sztatikus hazard keletkezése

A sztatikus hazárd akkor következik bc, ha a függvény egy változója két, jelkésletetés szempontjából különböző, ágon jut el a kimenetről. Az ábrán szereplő hálózat B változója ilyen, ezért a 2. kapun keresztül egy késleltetési idővel hamarabb jut az 1. kapura, mint a másik ágon. Ez egy kapu késleltetése idejére 0 szintre állítja a kimenetet.

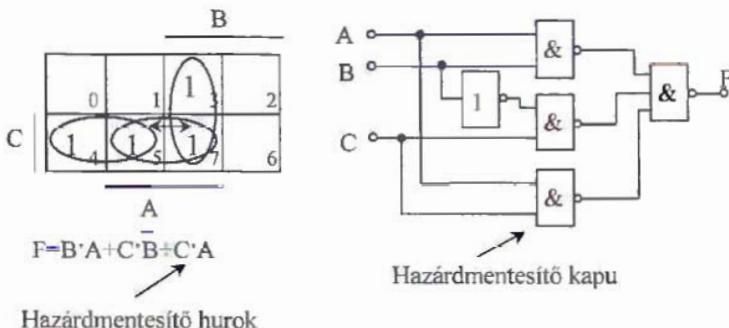
A dinamikus hazárd keletkezését az **5.15.** ábra szemlélteti.



**5.15.** ábra. A dinamikus hazárd keletkezése

Két sztatikus hazárral rendelkező hálózat – pl. az **5.14.** ábra hálózata – egyikének kimeneti jele még két másik kapun keresztül jut el a kimeneti ÉS kapura. Az  $F$  kimeneten így két kapu késleltetésnél megfelelő idővel később is megjelenik egy hazárd. Bonyolultabb hálózatnál akár sorozatos hazárd is felléphet. Ez az egymás után többször megjelenő hibás kimenneti szint a dinamikus hazárd. Az ábrából az is látható, hogy ha a kombinációs hálózatok nem rendelkeznek sztatikus hazárral akkor dinamikus hazárd sem lép fel.

A sztatikus hazárd megszüntetését az **5.16.** ábra alapján követhetjük.



5.16. ábra. A sztatikus hazárd megszüntetése

A V–K táblában ábrázolva a hazárdos hálózat függvényének hurkait, megállapíthatjuk, hogy akkor keletkezik a hazárd, amikor a B változó értéket vált, vagyis átlép egyik hurokból a másikba. Ezt a váltást mutatja a nyíl a V–K táblában. Ez általában is igaz: ha a V–K táblában két hurok hasonló elrendezésben érintkezik, akkor a két hurok között értéket váltó változó hazárdot okoz. Ezt felismerve a hazárdmentesítés úgy történik, hogy a két érintkező hurkot egy újabb hurokkal összekötjük. Ezzel látyszólag egy felesleges hurkot hozunk létre, tehát bonyolultabban realizáljuk a függvényt mint kellenc, de a hazárdot megszüntetjük.

A funkcionálisan teljes rendszerekkel való megvalósítás mellett **programozható logikai eszközökkel** is végezhető függvényrealizálás. Az eszközök rövidített neve PLD (*Programmable Logic Devices* – programozható logikai eszköz). A függvényrealizálásra alkalmas PLD-k nagyszámú kapuáramkört tartalmaznak, amelyek között az eszköz programozásával lehet a realizáláshoz szükséges összeköttetéseket létrehozni. A programozás, tehát a feladatnak megfelelő összéköttesek kialakítása, tulajdonképpen a valamennyi kapu között a gyártás során létrehozott összeköttetések közül a nem szükségesek megszüntetését jelenti. A programozó készülékkel az összeköttetés az áramkör meghatározott belső pontjain kiéhető, az áramkör tehát a továbbiakban már csak arra a feladatra használható, amire felprogramozták, más feladatra való átalakítása nem lehetséges.

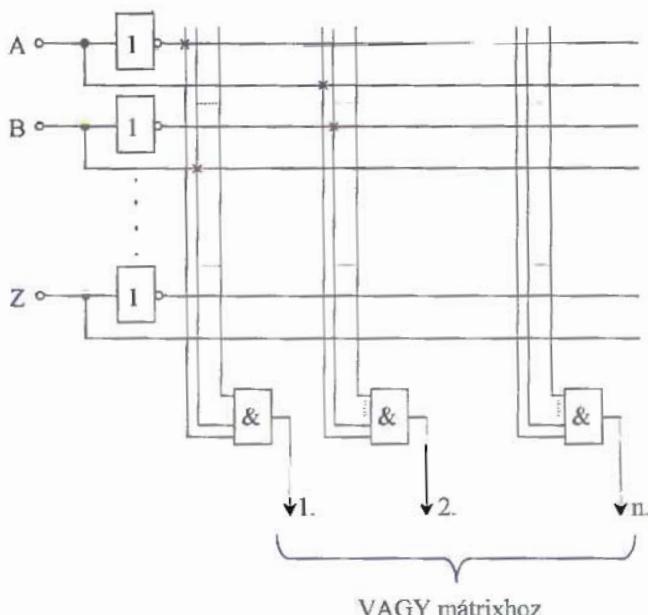
A PLD-k függvényrealizálásra alkalmas fajtái:

- PLA (*Programmable Logic Array* – programozható logikai elrendezés),
- PAL (*Programmable Array Logic* – programozható logikai elrendezés),
- PGA (*Programmable Gate Array* – programozható kapuelrendezés).

(Elvileg a később megismertő PLS áramkör is felhasználható függvényrealizálásra, azonban elsődleges felhasználási területe a szekvenciális hálózatok realizálása. Felépítését és használatát ezért a szekvenciális hálózatok megvalósításánál ismerjük meg.)

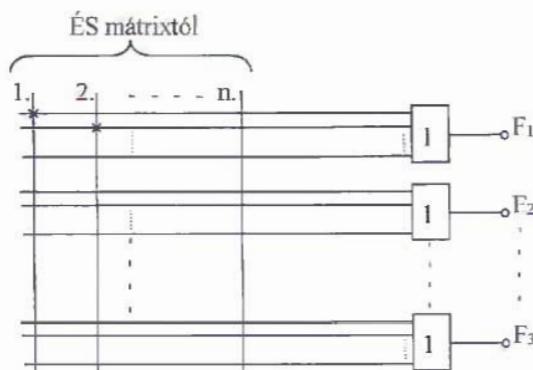
A belső felépítés elve valamennyi áramkörnél azonos, eltérés néhány kiegészítő áramkörben van az egyes típusok között. A legegyszerűbb belső felépítéssel a PLA rendelkezik, így ezen keresztül ismerjük meg a PLD-k használatát.

A PLA áramkör belső felépítését tekintve NÉV rendszerű hálózat, amelyben az inverterek és az ÉS kapuk, ill. a VAGY kapuk külön mátrixkapcsolást alkotnak. Az ÉS mátrix kapcsolási rajzát az 5.17. ábra tartalmazza.



5.17. ábra. A PLA ÉS mátrixa

Az inverterek vezetékei és az ÉS kapuk bemenetei között a gyártáskor létrehozott összeköttetések a programozás során megszüntethetők. A rajzon minden a **megmaradt összeköttetéseket jelöljük**. Az 5.17. ábrán pl. a jelölt összeköttetések miatt az ÉS mátrix 1. kimenetén a  $B \cdot \bar{A}$ , a 2. kimenetén pedig a  $\bar{B} \cdot A$  függvény jelenik meg. Az ÉS mátrixszal megalósított függvények a VAGY mátrixon keresztül jutnak a kiemeneti VAGY kapukra, amint azt az 5.18. ábra is mutatja.



5.18. ábra. A PLA VAGY mátrixa

A VAGY mátrix is programozható, ezért az ÉS mátrix függvényei bármelyik VAGY kapu bemeneteire kapcsolhatók. Az ábrán jelölt összeköttetésekkel csak egy függvényt valósítunk meg:  $F_1 = \overline{B} \cdot A + B \cdot \overline{A}$ .

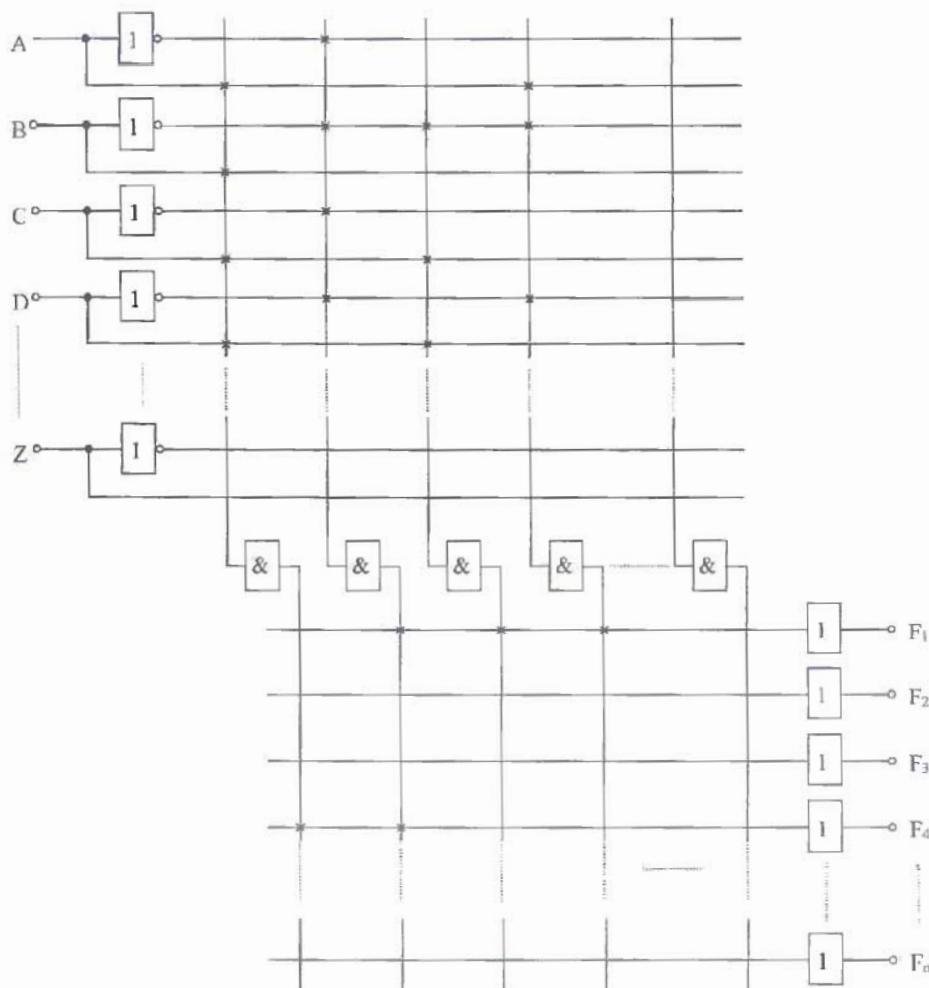
A PLA programozásánál az összeköttetések, tehát a függvények realizálása egy lépésben hozhatók létre, a belső áramkörök száma pedig nyilvánvalóan nem csökkenhető, tehát nincs jelentősége annak, hogy realizálás előtt a függvényt egyszerűsítettük-e vagy nem.

A PLA belső felépítésének egyszerűbb áttekinthetősége miatt az ÉS és a VAGY mátrixot sematikusan (nem minden részletre kiterjedően) szokás ábrázolni. Ezt mutatja az 5.19. ábra.

Az 5.19. ábrán jelölt összeköttetésekkel példaként két függvényt valósítottunk meg:

$$F_1^4 = \overline{D} \cdot \overline{C} \cdot \overline{B} \cdot A + D \cdot C \cdot \overline{B} + \overline{D} \cdot \overline{B} \cdot A,$$

$$F_4^4 = D \cdot C \cdot B \cdot A + \overline{D} \cdot \overline{C} \cdot \overline{B} \cdot \overline{A}$$

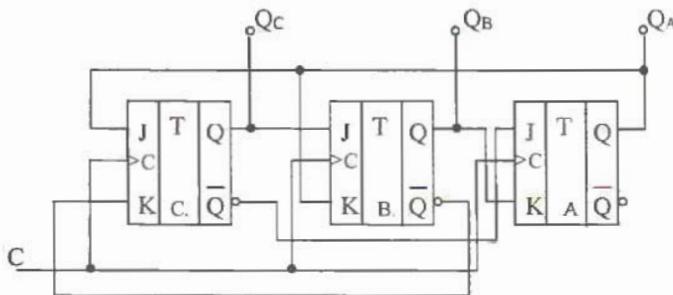


5.19. ábra. A PLA belső felépítése

## 5.2. Sorrendi hálózatok

A sorrendi vagy szekvenciális hálózatok olyan flip-flopokból felépülő, többkimennetű hálózatok, amelyek kimeneteiken az órajelek hatására előre meghatározott állapotok sorozatát veszik fel. Felépítésük jellemzője, hogy tárolókból – általában J-K flip-flopokból – épülnek fel és a sorrendiség (szekvencia) érdckében a következő kimeneti állapot függ az előző kimeneti állapottól. Az 5.20. ábra példaként egy három

kimenettel rendelkező szekvenciális hálózatot mutat. A tárolók a közös órajel miatt egyszerre működnek, ezért a hálózat **szinkron szekvenciális hálózat**. Érdemes megfigyelni, hogy a tárolók elvezéreltek, ezért a változás időpontja jól meghatározott.



**5.20.** ábra. Szinkron sorrendi hálózat

A hálózat működésének elemzéséhez (sorrendi hálózat analízise) írjuk fel a tárolók **vezérlési függvényeit** a kapcsolási rajz alapján!

$$J_C = Q_A; J_B = Q_C; J_A = \bar{Q}_C;$$

$$K_C = \bar{Q}_B; K_B = Q_A; K_A = Q_B.$$

A további vizsgálathoz tételezzük fel, hogy a kimenetek éppen  $Q_A = Q_B = Q_C = 0$  állapotban vannak. Ilyenkor az egyes tárolók bemenetén lévő vezérlések a vezérlési függvények alapján a következők:

$$J_C = 0; J_B = 0; J_A = 1;$$

$$K_C = 1; K_B = 0; K_A = 0.$$

Ezek a vezérlések a J-K tároló vezérlési táblázata (**4.2.** ábra) alapján a következő kimeneti állapotokat hozzák létre:

- a C jelű tároló a vezérlés hatására törlődik (most 0-ban marad a kimenete),  $Q_C = 0$ ,
- a B tároló megtartja előző állapotát, tehát  $Q_B = 0$ ,
- az A tároló beíródik,  $Q_A = 1$ .

Igy a hálózat új kimeneti állapota:

$$Q_C = 0; Q_B = 0; Q_A = 1.$$

Ez az állapot új vezérléseket jelent a tárolóknak:

$$J_C = 1; J_B = 0; J_A = 1;$$

$$K_C = 1; K_B = 1; K_A = 0.$$

Így, ha érkezik egy órajel, akkor a tárolók új kimeneti állapota:

- a C tároló megváltoztatja az előző állapotát,  $Q_C = 1$ ,
- a B tároló törlődik,  $Q_B = 0$ ,
- az A tároló beíródik,  $Q_A = 1$ .

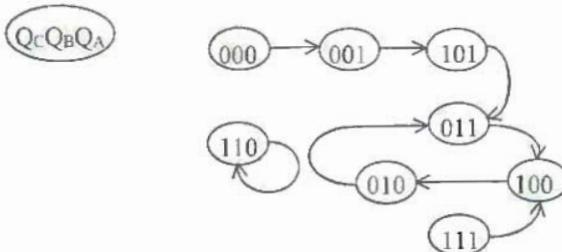
Az új kimeneti állapot tehát:  $Q_C = 1; Q_B = 0; Q_A = 1$ .

Ez újabb vezérlést jelent a tárolóknak, ezért az újabb órajel hatására ismét megváltozik a kimeneti állapotuk stb. Az analízist addig kell végezni, amíg az elvileg lehetséges összes kimeneti állapotot megvizsgáljuk. Az előzőekben bemutatott szöveges elemzés helyett áttekinthetőbb, ha az eredményeket táblázatba foglaljuk.

C	$J_C$	$K_C$	$J_B$	$K_B$	$J_A$	$K_A$	$Q_C$	$Q_B$	$Q_A$
							0	0	0
1.	0	1	0	0	1	0	0	0	1
2.	1	1	0	1	1	0	1	0	1
3.	1	1	1	1	0	0	0	1	1
4.	1	0	0	1	1	1	1	0	0
5.	0	1	1	0	0	0	0	1	0
6.	0	0	0	0	1	1	0	1	1
							1	1	1
1.	1	0	1	1	0	1	1	0	0
							1	1	0
1.	0	0	1	0	0	1	1	1	0

↔ már előfordult,  
 ↔ feltételezett új kimeneti állapot,  
 ↔ már előfordult,  
 ↔ feltételezett új kimeneti állapot,  
 ↔ már előfordult.

Az eredmények megadhatók az állapotok bináris alakjának felsorolásával táblázatosan, úgy ahogyan az előző táblázat utolsó három oszlopa mutatja. Sokkal áttekinthetőbb azonban az állapotdiagrammal való megadás. Ezt szemlélteti a megoldott példa kimeneti állapotait ábrázolva az 5.21. ábra.



5.21. ábra. Állapotdiagram

Az állapotdiagram nyilai azt mutatják, hogyan követik egymást a kimeneti állapotok. Az eredményt értelmezve látható, hogy

- ha a hálózat kimenete éppen 110 állapotban van, akkor ez az órajelek hatására nem változik meg. Az 110 állapot egy önmagában záródó hurkot alkot (reťszelő állapot),
- bármilyen más kimeneti állapotból a hálózat az órajelek hatására a 011-100-010-011... ciklusba kerül és a továbbiakban ebben a ciklusban marad.

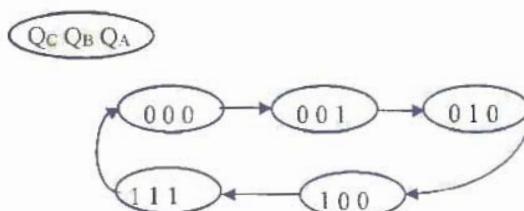
Az analízis során megismert hálózat szinkron sorrendi hálózat volt, mert a tárolók egyszerre kapták az órajelét, tehát a kimenetükön az állapotváltozás is egyidőben következett be. Aszinkron működésű sorrendi hálózatokkal tanulmányaink során csak a számlálóknál találkozunk, ezért működésüket is ott tekintjük át.

### A szinkron sorrendi hálózatok realizálása az állapotdiagram meghatározásával kezdődik.

Az állapotdiagram egy gyakorlati feladat egymást követő lépéseinak kódolt formája. Pl. egy közlekedési lámpa, amely a busz sávot külön engedélyezi, folyamatosan a piros → piros-buszsáv zöld → piros-sárga → zöld → sárga → piros → stb. sorrend szerint működik (tekintve most az egyes lámpaváltások közötti időzítésektől). A lámpák bekapcsolását állapotoknak tekintve kódolhatjuk az egyes állapotokat, pl. a következőképpen.

Állapotok	Állapotkódok
piros	000
piros-buszsáv zöld	001
piros-sárga	010
zöld	100
sárga	111
piros	000

Az állapotok hárombites bináris kódval kódolhatók, mert az öt egymástól különböző állapot kódolására két bit kevés, négy bit pedig feleslegesen sok lenne. Az állapotok és a hárombites kódok egymáshoz való rendelése tetszőleges. Az előző kódolás szerinti állapotdiagramot az 5.22. ábra mutatja.



5.22. ábra. A közlekedési lámpa állapotdiagramja

A realizáláshoz J-K tárolókat használunk, mert ennek a tárolótípusnak nincsenek tiltott vezérlési állapotai, ugyanakkor, ha a feladat úgy kívánja, készíthető belőle T, ill. D tároló is. Fogalmazhatunk úgy is, hogy a J-K tároló a legrugalmásabban alkalmazható flip-flop. A vezérlés szempontjából az élvezérelt tároló a legmegfelelőbb, mert így biztosítható legjobban a kimeneti állapotuk megváltozásának szinkronitása.

A bináris kód egyes bitjeit a J-K tárolók állítják elő kimeneteiken, ezért a feladatban található hárombites kóhoz három flip-flopra van szükség.

A tervezéshez két segédletet használunk munkánk megkönnyítésérc. Az egyik a J-K tárolók állapotátmeneti táblázata (fordított vezérlési táblázat):

$Q_n$	$Q_{n+1}$	$J$	$K$
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Az állapotátmeneti tábla első két oszlopa tartalmazza az összes Ichetséges állapotátmenetet, amely egy feladat megoldása során előfordulhat. A J és K oszlopa pedig azokat a szükséges vezérléseket adja meg, amelyek az első két oszlopan levő állapotátmenetekhez szükségesek. A táblázat egyes sorainak magyarázata:

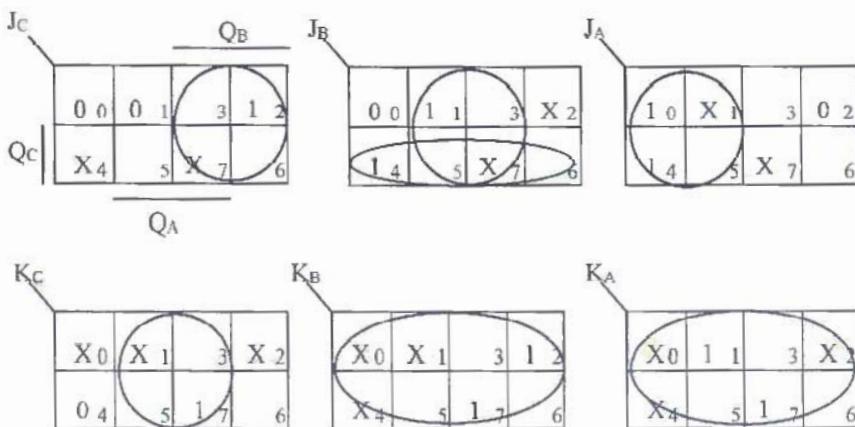
- első sor: ahhoz, hogy a J-K tároló kimenetén  $0 \rightarrow 0$  átmennet jöjjön létre az órajel hatására, nem szabad a tárolót beírni, tehát  $J = 0$ . A K bemenetre adott vezérlés közömbös, mert, ha  $K = 0$ , akkor a tároló megtartja előző állapotát, ha  $K = 1$ , akkor törlődik. Vagyis a flip-flop kimeneti állapota mindenkorban 0 lesz (l. a J-K tároló vezérlési táblázatát a 4.2. ábrán),
- második sor: ahhoz, hogy a tároló kimenetén 1 legyen,  $J = 1$  vezérléssel beírást kell végezni. A K bemenet vezérlése közömbös, mert  $K = 0$ -ra beírás törtenik,  $K = 1$  vezérlésre pedig a flip-flop megváltoztatja előző állapotát ( $J = 1, K = 1 \rightarrow Q_{n+1} = Q_n$ ),
- harmadik sor: a tárolónak törlődni kell, chhez  $K = 1$  vezérlés kell. A J bemenet állapota közömbös, aminek magyarázata az előzőekből már következik,
- negyedik sor: nem szabad a tárolót törölni, tehát  $K = 0$ .

A másik segédlet egy olyan táblázat, amely egymás alatt, sorrendben tartalmazza a kimenetek állapotait, feltüntetve az egyes állapotok kódjának decimális megfelelőit is, valamint az adott állapotban szükséges J és K vezérléseket. A három tároló kimeneteit jelöljük  $Q_C, Q_B, Q_A$ -val. Így a feladathoz tartozó táblázat:

$Q_C$	$Q_B$	$Q_A$	Dec. érték	$J_C$	$K_C$	$J_B$	$K_B$	$J_A$	$K_A$
0	0	0	0	0	x	0	x	1	x
0	0	1	1	0	x	1	x	x	1
0	1	0	2	1	x	x	1	0	x
1	0	0	4	x	0	1	x	1	x
1	1	1	7	x	1	x	1	x	1
0	0	0	0						

stb.

A tervezés célja a J-K tárolók vezérlési függvényeinek meghatározása, amelyek biztosítják, hogy a hálózat az órajelek hatására az előírt állapotokat vegye fel. A vezérlési függvények logikai függvények, amelyek legegyszerűbb alakja, grafikus egyszerűsítéssel, a V-K minterm táblákból olvasható ki. minden tárolóbemenethez tartozik egy függvény, így egy V-K tábla is. A táblázatok változói a  $Q_C$ ,  $Q_B$ ,  $Q_A$  kimenetek előző állapotban felvett értékei, hiszen a következő állapotot az határozza meg, hogy előzőleg mi történt a hálózatban (sárga után jön mindenki a zöld, a zöld után mindenki piros stb.). Az egyes bemenetek V-K tábláit az 5.23. ábra mutatja.



5.23. ábra. A vezérlési függvények táblázatai

A táblázatra vonatkozó hurkolási szabályok:

- annyi hurkot kell képezni, hogy valamennyi 1 hurokban legyen,
- az X közömbös termeket használhatjuk a hurkoláshoz, ha az egyszerűsíti a függvényt, tchát nagyobb hurkot tesz lehetővé,
- a V-K táblák üres cellái azt jelentik, hogy a hozzájuk tartozó termeket nem használtuk fel az állapotok kódolására. Ezek a termek tehát nem fordulhatnak

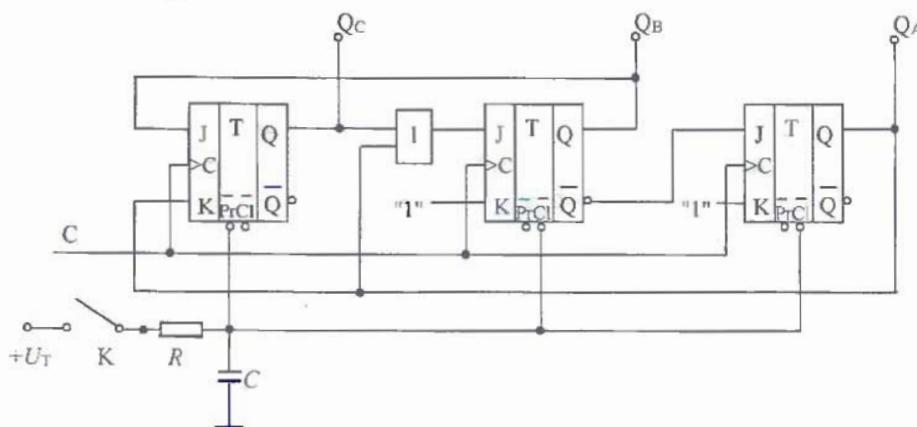
elő a működés során, kivéve a bekapcsolás pillanatát. Ha megfelelő kapcsolástechnikai megoldással gondoskodunk arról, hogy bekapcsoláskor valamelyik felhasznált állapotba kerüljön a rendszer, akkor valóban semmilyen körülmenyek között nem fordulhatnak elő az üres cellák termjei. Ezt feltételezve az üres cellákat úgy tekinthetjük, mintha közömbösek lennének, ezért ezeket szükség szerint felhasználhatjuk a hurkolásnál. A tervezési folyamat végén azonban gondoskodni kell arról, hogy a tárolók egy meghatározott kezdő állapotba kerüljenek.

Ezeket a szabályokat figyelembe véve a feladat vezérlési függvényei:

$$J_C = Q_B; \quad J_B = Q_C + Q_A; \quad J_A = \bar{Q}_B,$$

$$K_C = Q_A; \quad K_B = 1; \quad K_A = 1.$$

A vezérlési függvények alapján a sorrendi hálózat realizálható. A három tárolót közös órajelről vezéreljük, így lesz szinkronműködésű a hálózat. A vezérlési függvényeket bármelyik rendszerben realizálhatjuk. Példánkban a realizáláshoz csak egy VAGY kapura van szükség a  $J_B$  bemenet vezérléséhez. A megvalósított hálózatot az 5.24. ábra mutatja.



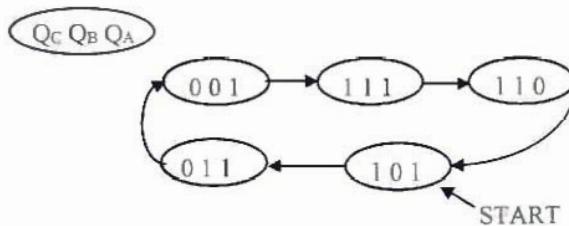
5.24. ábra. A közlekedési lámpa vezérlő áramköre

A bekapcsoláskor szükséges beállítást (START állapot) a flip-flopok sztatikus bemenetein keresztül végezzük el. Használunk negált sztatikus bemenetű tárolókat és végezzük el úgy a beállítást, hogy bekapcsoláskor minden tároló zöld jelzéssel induljon a működés. Ehhez a tárolókat  $Q_C = 1$ ,  $Q_B = 0$ ,  $Q_A = 0$  állapotba kell hozni bekapcsoláskor. Ezt a kezdeti értékbeállítást a bekapcsolás pillanatában az  $RC$  tag végzi el, ami egy differenciáló áramkör, ezért egy 0 szintű tüske impulzust ad a bekötés szerint a C tároló preset, a B és A tároló clear bemenetére. A tárolók kimenetei ennek hatására a kívánt állapotot veszik fel. Mivel a sztatikus vezérlés csak igen rövid ideig

hat, a továbbiakban – a következő órajeltől – a működés már a vezérlési függvények szerinti.

### 5. feladat

Realizáljuk szinkron sorrendi hálózattal az 5.25. ábrán adott állapotdiagramot! Bekapcsoláskor a hálózat a jelölt állapotból induljon!



5.25. ábra. Az 5. feladat állapotdiagramja

### Az 5. feladat megoldása

A realizáláshoz 3 db élvezérelt J-K tároló szükséges. A kimeneteiken megjelenő állapotSORozat:

$Q_C$	$Q_B$	$Q_A$	Dec. érték
0	0	1	1
1	1	1	7
1	1	0	6
1	0	1	5
0	1	1	3
0	0	1	1
stb.			

A bemenetek kitöltött vezérlési táblázatait az 5.26. ábra mutatja.

$J_C$	$Q_B$				$J_B$	$Q_B$				$J_A$	$Q_B$			
$Q_C$	0	1	2	3	$Q_C$	0	1	2	3	$Q_C$	0	1	2	3
	4	X	5	X		4	1	5	X		4	X	5	X
	Q_A					Q_A					Q_A			
K_C	0	X	1	X	Q_C	0	1	X	3	Q_C	0	X	1	X
Q_C	4	1	5	X		4	1	5	X		4	X	5	X
	Q_A					Q_A					Q_A			
K_B	0	X	1	X	Q_C	0	X	1	3	Q_C	0	X	1	X
Q_C	4	1	5	X		4	X	5	1		4	X	5	X
	Q_A					Q_A					Q_A			
K_A	0	0	0	0	Q_C	0	0	0	0	Q_C	0	0	0	0
Q_C	4	0	5	1		4	0	5	1		4	0	5	1
	Q_A					Q_A					Q_A			

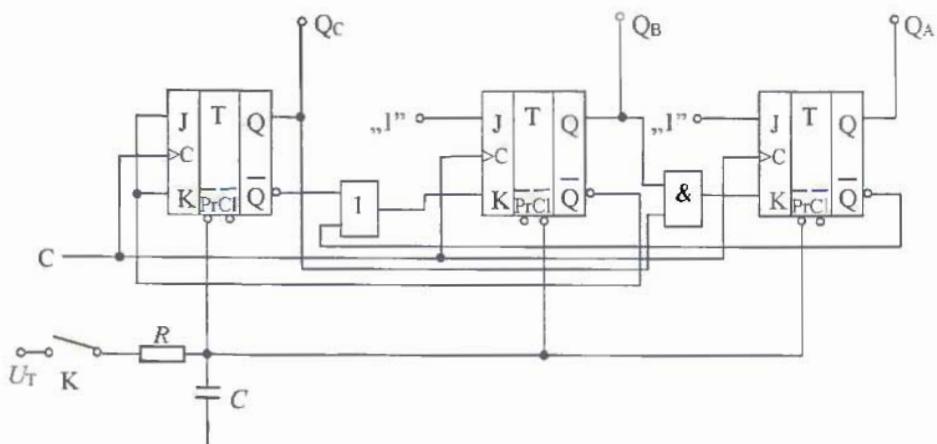
5.26. ábra. Az 5. feladat vezérlési táblázatai

A vezérlési táblázatokból kiolvasható vezérlési függvények:

$$J_C = \overline{Q_B}; \quad J_C = 1; \quad J_A = 1,$$

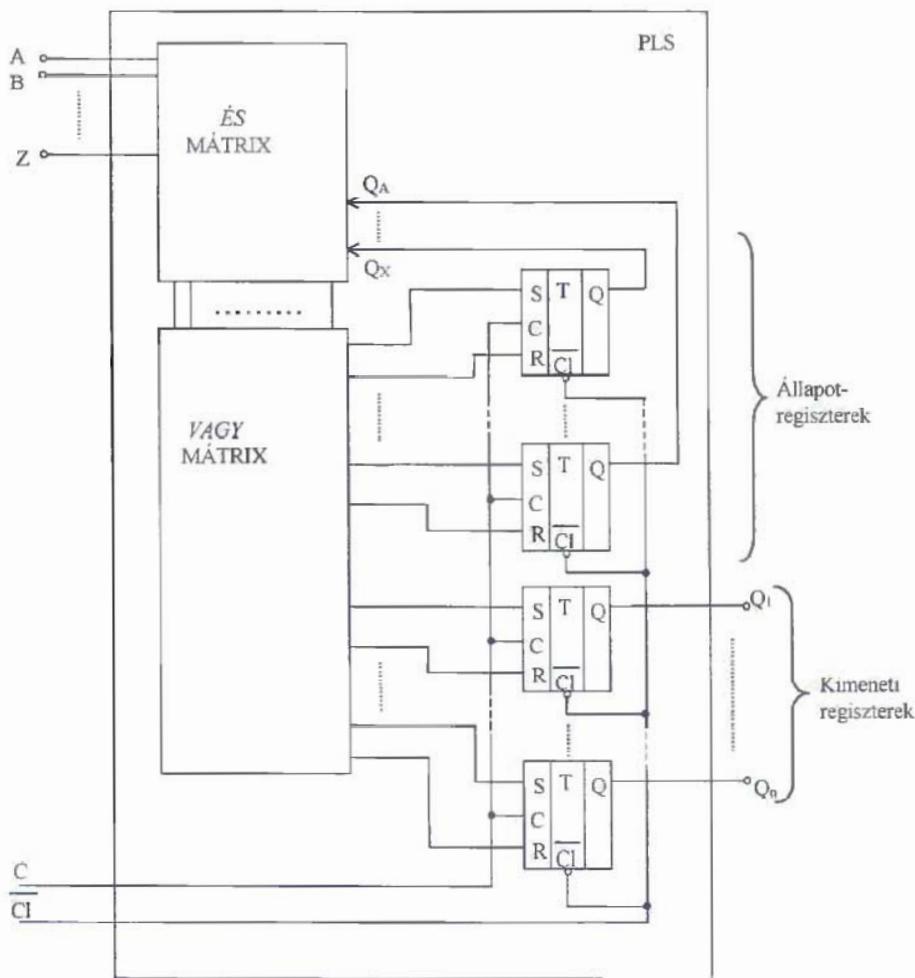
$$K_C = \overline{Q_B}; \quad K_B = \overline{Q_C} + \overline{Q_A}; \quad K_A = Q_C \cdot Q_B.$$

A megvalósított hálózatot az **5.27.** ábra mutatja.



**5.27.** ábra. Az 5. feladat áramköre

A szinkron sorrendi hálózatok realizálhatók **PLS** (*Programmable Logic Sequencer* – programozható logikai sorrendképző) **áramkör** felhasználásával is. Az áramkör belső felépítése az **5.28.** ábrán látható.



5.28. ábra. A PLS áramkör belső felépítése

A PLS áramkörben lévő ÉS mátrix és a VAGY mátrix felépítése pontosan megegyezik a PLA áramkörnél megismert hasonló mátrixokkal. Ebben az áramkörben az állapotregiszterekkel való visszacsatolás miatt lehetőség van arra, hogy a két mátrix a regiszterek vezérlési függvényeit állítsa elő, úgy, ahogyan a programozást elvégezzük. Az állapotregiszterek kimenetei nincsenek kivezetve, hanem a megvalósított sorrendi hálózat a kimeneti regisztereken keresztül válnak hozzáférhetővé. A realizáláshoz használható regiszterek R-S típusúak, amit a vezérlési függvények meghatározásánál figyelembe kell venni.

A PLS áramkörök úgy használhatók fel, hogy a megismert módon meghatározzuk a megvalósítandó sorrendi hálózat vezérlési függvényeit R-S tárolókra. Ezeket a vezérlési függvényeket a  $Q_A \dots Q_X$  változókkal írjuk fel. (A hagyományos módszernél ezek voltak a  $Q_A, Q_B$  stb. változók a mintermtáblában). A vezérlési függvényeket a mátrixokkal beprogramozzuk az állapotregiszterek bemeneteire. Az így megvalósított sorrendi hálózat kimenetei a  $Q_A \dots Q_X$  kimenetek lennének, azonban közvetlenül nem hozzáférhetők. Ezért a mátrixok programozásával arról is gondoskodni kell, hogy ezek a kimeneti regiszterek bemenetére kerüljenek. Így végül is a sorrendi hálózat kimenetei a  $Q_1 \dots Q_n$  kimenetek lesznek.

### Ellenőrző kérdések

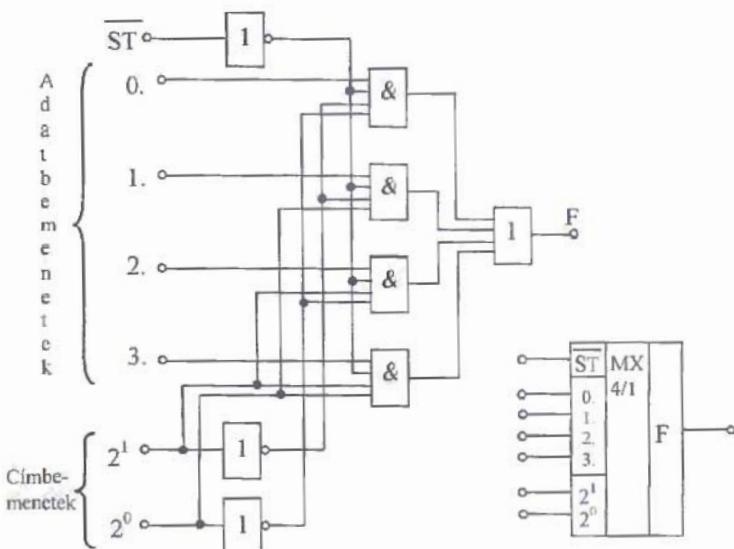
1. Mi a funkcionálisan teljes rendszer fogalma?
2. Ismertesse a NAND és NOR kapuval történő realizálás szabályait!
3. Mit jelent a hazárd, milyen formái vannak, hogyan küszöbölhető ki?
4. Rajzolja fel a PLA belső felépítését!
5. Milyen elemekből épülnek fel a sorrendi hálózatok?
6. Ismertessük a szinkron sorrendi hálózatok tervezésének és realizálásának módszereit!
7. Hogyan állíthatók a sorrendi hálózatok alapállapotba?

## 6. Funkcionális áramkörök

A funkcionális áramkörök olyan digitális integrált áramkörök, amelyek egy adott feladat ellátására készültek. Közös jellemzőjük, hogy kapukból és tárolókból épülnek fel és az áramkörök tokból csak a funkció ellátásához szükséges kivezetések vannak kivezetve. A digitális áramkörökben ezeket alkatrészként használjuk, jellemzőik, működésüket leíró táblázataik a katalógusokban megtalálhatók.

### 6.1. Multiplexerek

A multiplexer adatválasztó áramkör, amely a bemeneti közül a címzéssel kiválasztottat kapcsolja össze a kimenettel. Így a kiválasztott bemeneten lévő adat jut a kimenetre. A 6.1. ábra egy olyan multiplexert szemléltet, amelyik négy **adatbemenetet** és az ezeket közül egyet kiválasztó két **címzőbemenettel** rendelkezik. Az áramkör elnevezése ezért 4/1 (négyből egy) multiplexer.

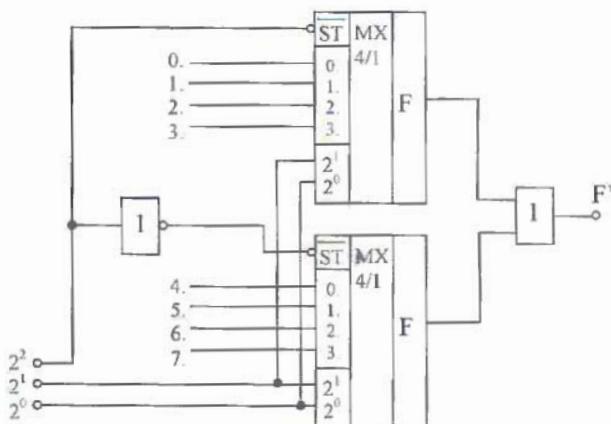


6.1. ábra. A 4/1 multiplexer belső felépítése és jelképi jelölése

A címzőbemenetekre adott bináris cím bitjei közvetlenül vagy invertoren keresztül úgy kapcsolódnak az ÉS kapuk bemeneteire, hogy azok egyike a bináris címcik megfelelő decimális sorszámú bemenetet engedélyezze a kimenetre jutni. Pl. ha a  $2^0$  címbemenetre 1, a  $2^1$  bemenetre pedig 0 címzö bit jut, akkor az utóbbi inverteren keresztül jut az 1. sorszámú kapura. A kapu két címzőbemenetére így 1 szint kerül, engedélyezve az 1. adatbemenetet.

A minden ÉS kapura bevezetett ST (*Strobe* – kapuzás) bemenet az egész áramkört (integrált áramköri tokot) engedélyezi, vagy tiltja. Általában negált szintű bemenet. A jelképi jelölésen a be- és kimenetek, az engedélyezés jelölése található, valamint a multiplexer funkciója: MX.

Az SN sorozatban pl. a 74151 típus egy 8/1, a 74150 típus egy 16/1 multiplexer. A CD sorozatból a 4512B egy 8/1, a 4539B típus pedig két 4/1 multiplexer egy tokban. A multiplexerek bemenetcinek száma **kaszkádosítással** (bövítéssel) növelhető. A 6.2. ábra a módszer bemutatására két 4/1 multiplexer felhasználásával egy 8/1 multiplexer kialakítását mutatja.

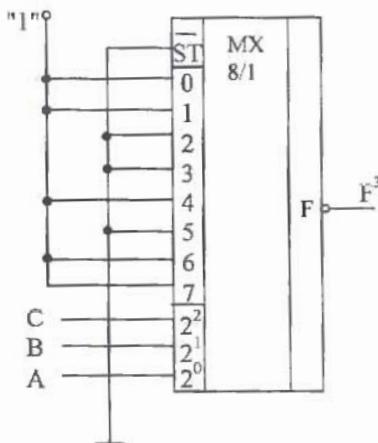


6.2. ábra. Multiplexerek kaszkádosítása

A két 4/1 multiplexer címzőbemeneteit párhuzamosítjuk. A legnagyobb helyi értékű címzőbemenetet az ST kapuzó bemenetekből alakítjuk ki. Ez választ a két tok közzött. A közös címek csak az egyik tokra hatásosak. A kimeneteket egy VAGY kapu fogja össze egyetlen kimenettel.

A multiplexereket alapfunkciójuknak megfelelően általában adatválasztási céllra használjuk. További felhasználási lehetőségek a függvényrealizálásra való alkalmazás és párhuzamos-soros átalakító készítése.

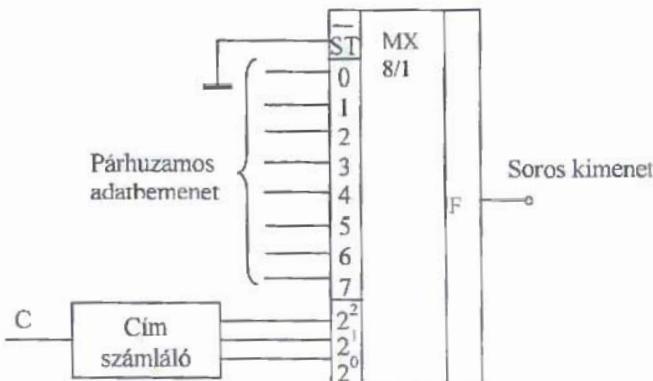
A multiplexerrel diszjunktív szabályos alakú függvényt lehet realizálni. Erre mutat példát a 6.3. ábra. Az áramkör az  $F^3 = \Sigma^3 (0,1,4,6,7)$  függvényt realizálja.



**6.3.** ábra. Függvényrealizálás multiplexerrel

Olyan multiplexert kell választani a realizáláshoz, amelyiknek annyi címző bemenelete van, ahány változós a függvény. A változókkal megcímzett bemenetekre 1 szintet kapcsolunk akkor, ha a változókból képzett term szerepel a függvényben. Ellenkező esetben 0 kerül az adatbemenetre. Pl. a  $\bar{C} \cdot \bar{B} \cdot A$  minterm sorszáma 1, ezért ha ez kerül a címzöbemenetre, akkor az 1. adatbemenetet címzi meg. Ezen a bemeneten 1 szintnek kell lenni, hiszen a függvény erre a termre igaz értéket ad.

Párhuzamos-soros átalakító a **6.4.** ábra szerint alakítható ki multiplexerből.

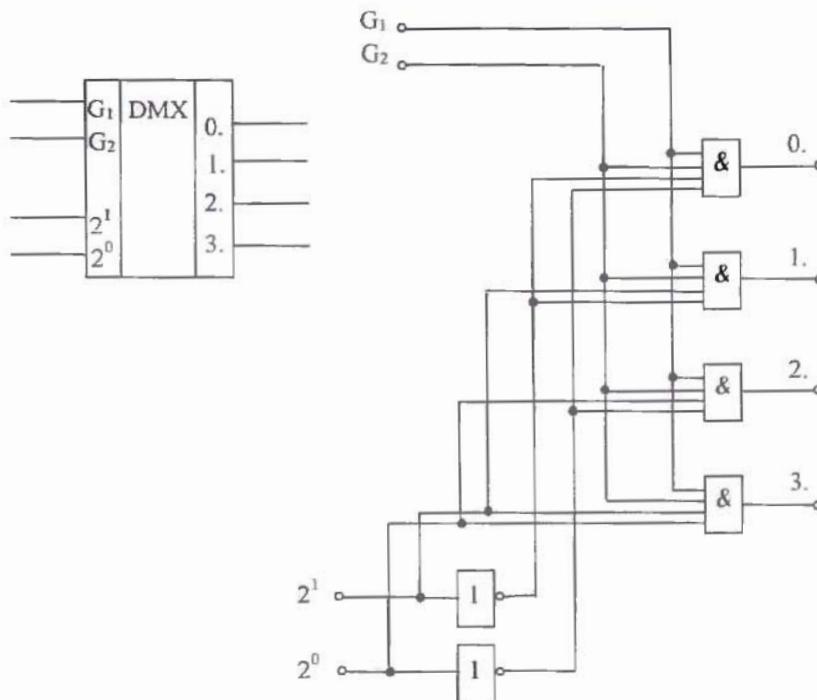


**6.4.** ábra. Párhuzamos-soros átalakító multiplexerrel

A címzöbemenetekre olyan áramkör kerül, amelyik a 000 címtől kezdődően az 111 címig sorban végig címzi az adatbemeneteket. Az adatbemeneten lévő párhuzamos adatbíték sorban egymás után a kimenetre kerülnek. A címek váltása, tehát az adatok kimenetre kapcsolásának időpontja az órajellel ütemezhető.

## 6.2. Demultiplexerek, dekódolók

A demultiplexerek a multiplexerekhez képest ellenkező jellegű funkciót látnak el: az egy vonalon érkező információt több kimenet között osztják szét. A kimenetek közül címzéssel választanak. A demultiplexerek belső felépítésének elvét a 6.5. ábra mutatja, jelképi jelölésével együtt, 1/4 demultiplexer esetén. A tényleges belső felépítés ettől eltérhet, mert másképpen történik a konkrét áramköri kialakítás TTL, és másképpen a MOS áramkörökönél. Bármilyen is a tényleges áramkör, a megvalósított funkció az ábrán láthatónak felel meg.



6.5. ábra. A demultiplexerek belső felépítése

A címzés áramkörc megegyezik a multiplexernél megismert áramkörrel. Az adatbemenet egy engedélyezést/tiltást végző ÉS kapun keresztül kapcsolódik a kimenetre. Az engedélyezés/tiltás a G (gate – kapu) bemeneten keresztül lehetséges. Szokásos jelölése még: E (enable – engedélyezés). A felhasználás szempontjából ez a bemenet tulajdonképpen az egész tokot engedélyczi/tiltja. A kapcsolásból látszik, hogy az adatbemenet és az engedélyező bemenet felcserélhető. A katalógusok ezért jelölés-

ben nem is tesznek különbséget közöttük, G, vagy E betűvel jelölik ezeket a bemeneteket.

Az SN sorozatban pl. 1/8 demultiplexer a 74LS138, vagy egy tokban két 1/4 demultiplexer a 74156 típus. A CD sorozatban egy tokban két 1/4 demultiplexer a 4555 típus.

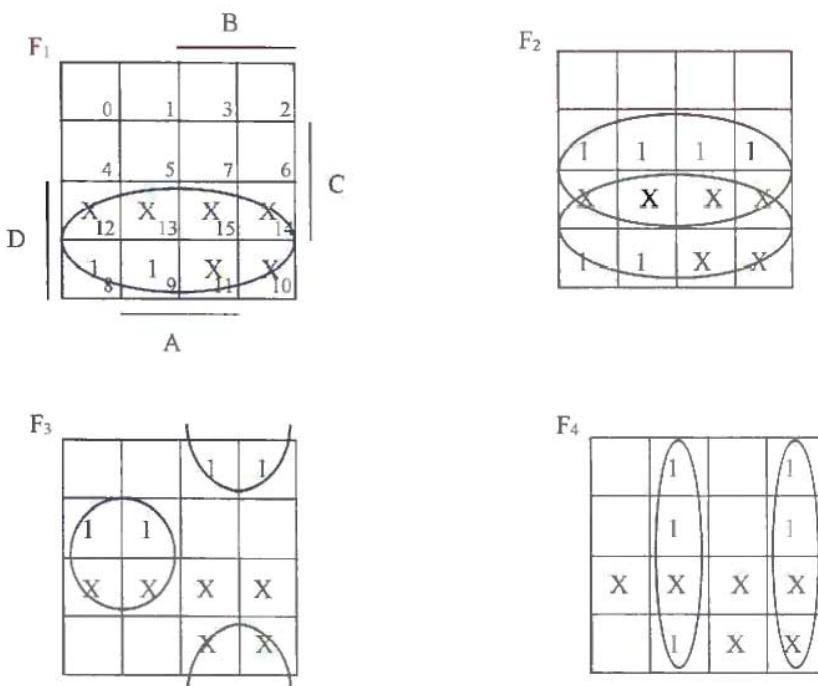
Az adatszétoztáson kívül a demultiplexerek dekódolóként is alkalmazhatók. Az **6.5.** ábra áramköre pl. egy kétbites bináris számot dekódol decimális számmá. Ha az engedélyező bemenetekre 1 szintet kapcsolunk, akkor a címzésre használt kétbites bináris szám decimális megfelelőjével sorszámozott kimenetet jelenik meg 1 szint. Tehát a bináris szám *megjelöli* a decimális értékével sorszámozott kimenetet. Ebben az alkalmazásban a demultiplexer egy 2/4 dekódoló. Az 1/8 demultiplexerek 3/8 bináris-decimális dekódolók, az 1/16 DMX 4/16 bináris-decimális dekódoló stb. A demultiplexerek felépítésükön következően csak bináris-decimális dekódolást képesek elvégezni. Készülünk igen szűk választékkal kifejezetten dekódolási funkciót ellátó áramkörök is, pl. BCD-decimális, háromtöbbletes-decimális.

A nagyon kevés integrált áramköri formában kapható dekódoló miatt gyakran kénytelenünk arra, hogy kapuáramkörökből készítsük dekódolót. A dekódoló egy többkimenetű kombinációs hálózat, ezért a kombinációs hálózatok realizálásánál megismert módszer itt is alkalmazható. A tervezés menetét egy BCD Gray-dekódér tervezésével ismerjük meg. Az alkalmazott módszer bármilyen dekódoló tervezéséhez felhasználható.

Első lépésként a két kódrendszer egyes elemeit egymáshoz rendeljük, tehát minden bemeneti kódszó kimeneti megfelelőjét külön-külön megadjuk:

BCD kód (bemenetek)				Gray-kód (kimenetek)			
D	C	B	A	F1	F2	F3	F4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1

Második lépésként – a kódtáblázatot egy többkimenetű kombinációs hálózat igazságtablázatának tekintve az F<sub>1</sub>, F<sub>2</sub>, F<sub>3</sub>, F<sub>4</sub> függvényeket egyszerűsítjük, célszerűen grafikus módszerrel. A függvények V-K minterm tábláit a **6.6.** ábra mutatja.

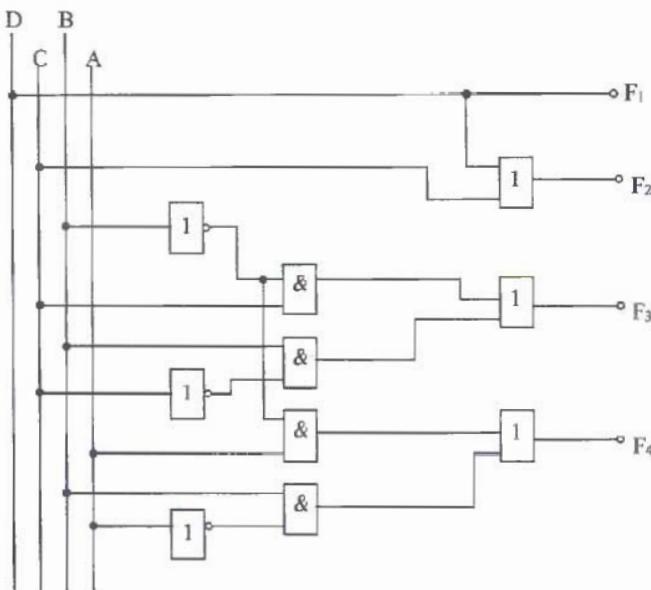


6.6. ábra. BCD-Gray-dekódoló V-K táblái

Mivel a bemeneti kód BCD kód, ezért a 9-nél nagyobb sorszámú termek nem fordulhatnak elő. A függvények egyszerűsítésénél ezek a termek határozatlannak tekinthetők. A dekódoló függvényei a V-K táblákból:

$$F_1 = D; \quad F_2 = D + C; \quad F_3 = C \cdot \bar{B} + \bar{C} \cdot B; \quad F_4 = \bar{B} \cdot A + B \cdot \bar{A}.$$

A NÉV rendszerben megvalósított dekódoló kapcsolási rajzát a 6.7. ábra mutatja.



6.7. ábra. BCD-Gray-dekódoló

### 6.3. Aritmetikai áramkörök

Az aritmetikai áramkörök bináris számrendszerben dolgozó műveletvégző áramkörök. Az áramkör bemeneti adatait operandusoknak nevezzük, amelyek legtöbbször bináris vagy BCD kódú bináris számok. Ennek megfelelően a műveletvégzés eredménye is bináris vagy BCD kódú bináris szám.

Bizonyítható, hogy bármilyen művelet (pl. szorzás, gyökvonás, integrálás stb.) visszavezethető komplementeképzések, összeadások és léptetések sorozatára. A bináris számok bitjeinek egyik helyi értékéről a másikra való léptetése a 7.4.4. pont részleteivel végezhető el. A bináris összeadás az erre a céllra készített hálózattal, az összeadó áramkörrel végezhető el. Ezek az áramkörök a bináris összeadás szabályai szerint működnek:

- összeadás mindenkorának csak két operandus között végezhető,
- az operandusok bitjeit helyi értékenként kell összegezni, a legkisebb helyi értéktől kezdve. minden helyi értéken képezni kell az **S** (szumma) összegbitet és a **C** (carry) átvitelbitet. Az átvitel a bináris számrendszerben ugyanaz a mennyiség, mint amit a tízes számrendszerben használunk. Az összeg bit és az átvitel bit képzésének szabályát a legkisebb helyi értékre (a 0. helyi érték) a következő táblázat mutatja:

B <sub>0</sub>	A <sub>0</sub>	S <sub>0</sub>	C <sub>1</sub>
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Magyarázatot csak az utolsó sor igényel:  $1 + 1 = 10_2$ , ezért az eredmény 0 és marad 1 (átvitel).

- az egyes helyi értékeken az összeadásnál figyelembe kell venni az előző helyi értékről jövő átvitelt. A táblázatban az átvitel indexe is erre utal: a 0. helyi értéken keletkező átvitel az 1. helyi értéken kell figyelembe venni,
- a legnagyobb helyi értéken keletkező átvitel az összeg legnagyobb helyi értékű bitje. Az eredmény tehát egy bittel hosszabb lehet, mint az operandusok.  
Pl. négy bites operandusoknál:

$$\begin{array}{r} 1010 \quad (\text{decimálisan: } 10 + 13 = 23) \\ + 1101 \\ \hline 10111. \end{array}$$

A legkisebb helyi értékre felírt táblázat egy olyan egy bites összeadó igazságtáblázata, amely nem veszi figyelembe az előző helyi értékről jövő átvitelt. Az ilyen összeadót **félösszeadónak** (HS, half szummatör – félösszeadó) nevezzük. Az igazságtáblázat alapján felírható S és C függvények:

$$S = B \cdot \overline{A} + \overline{B} \cdot A, \quad C = B \cdot A.$$

Az összeg antivalencia függvény, az átvitel pedig ÉS függvény. A függvényeket megvalósító félösszeadó kapcsolás a 6.8. ábrán látható.



6.8. ábra. Félösszeadó áramkör

Az egy helyi értéken teljes összeadást végző áramkör figyelembe veszi az előző helyi értékről jövő átvitelt is. Ezt a működést leíró igazságtáblázat:

$B_n$	$A_n$	$C_n$	$S_n$	$C_{n+1}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

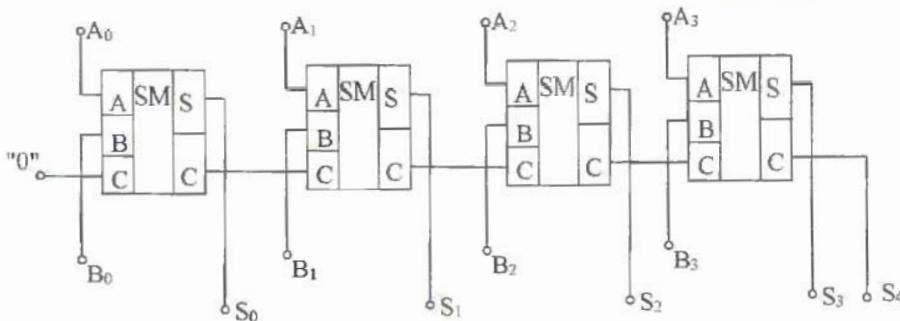
Az igazságtablázat alapján felírható a teljes összeadó összeg- és átvitelfüggvénye. Az összegfüggvény nem egyszerűsíthető, az átvitelfüggvény igen. Egyszerűsítés után a teljes összeadó függvényei:

$$S_n = C_n \cdot \bar{B}_n \cdot \bar{A}_n + \bar{C}_n \cdot B_n \cdot \bar{A}_n + \bar{C}_n \cdot \bar{B}_n \cdot A_n + C_n \cdot B_n \cdot A_n,$$

$$C_{n+1} = B_n \cdot A_n + C_n \cdot A_n + C_n \cdot B_n.$$

Ezeket a logikai függvényeket vagy célszerűen csoportosított változatukat valósítják meg kombinációs hálózattal az integrált összeadó áramkörök. A teljes hálózat felrajzolásától eltekintünk.

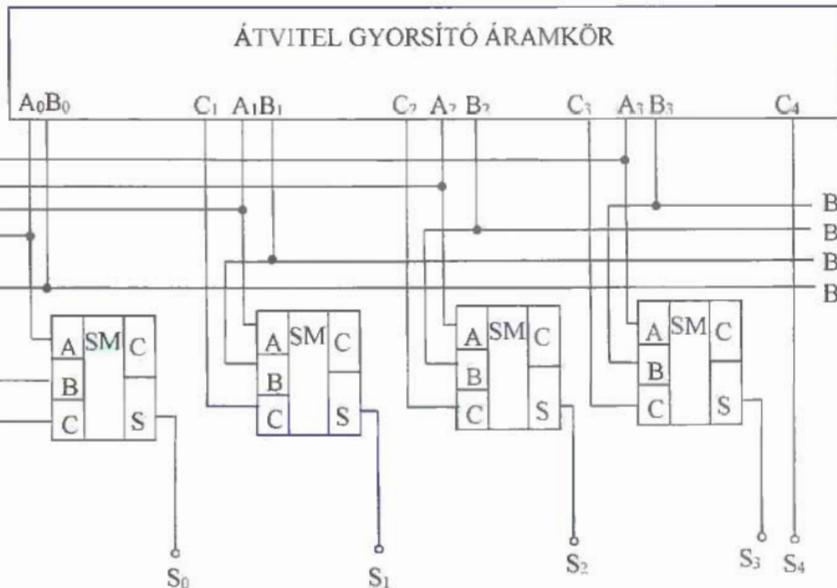
Az egy helyi értékes teljes összeadó felhasználásával készíthetünk több bites összeadót is úgy, hogy egymással párhuzamosan annyi egy helyi értékes összeadót alkalmazunk, ahány bitesek az operandusok. A 6.9. ábra négy bites teljes összeadót mutat.



6.9. ábra. Négy bites teljes összeadó

A legkisebb helyi érték  $S_0$  átvitel bemenetét a használat során 0 szintre kötjük, hiszen itt nincs előző helyi értékről jövő átvitel. A legnagyobb helyi értéken keletkező  $C_3$  átvitel az összeg legnagyobb helyi értékű  $S_4$  bitje. Az egy helyi értékes összeadók között az átvitel sorosan terjed, mert egy következő helyi értéknek minden meg kell várnia az előző helyi értéken való összeadás befejezését. A gyors összeadást tehát az

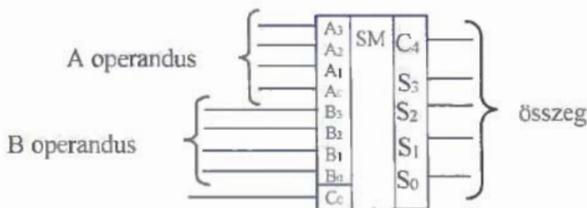
átvitel soros terjedésc akadályozza. Készülnek ezért olyan összeadók, amelyek **átvitelgyorsító áramkört tartalmaznak**. A bonyolult felépítésű áramkörök részletes ismertetésre nincs mód, működési elvük azonban egyszerű: az operandusok egyes bitjeiből egy kombinációs hálózat meghatározza az egyes helyi értékeken keletkező átviteleket, és ezeket párhuzamosan, az összeadók bemenetére juttatja. Az egyes helyi értékek összeadói így szinte egyidőben kapják a két operandus bitét és az átvitelbitet. Az ilyen elven működő összeadó belső felépítését mutatja a 6.10. ábra.



6.10. ábra. Összeadó gyorsított átvitelképzéssel

Az SN sorozat átvitelgyorsítás nélküli összeadói pl. a 7482, 7483 típusok, átvitelgyorsítással működik a 74283. A CD sorozat 4008 típusú összeadója átvitelgyorsítás nélküli.

A belső felépítéstől eltérően az összeadókat kapcsolási rajzokon jelképi jelölésükkel ábrázoljuk. Négybites összeadó jelképi jelölését mutatja a 6.11. ábra.



6.11. ábra. Négybites összeadó jelképi jelölése

Nem készítnek integrált kivitelben bináris kivonó áramköröket. Szükség esetén azonban bináris összeadóból egyszerűen készíthető kivonó áramkör. A kivonás műveletét is – mint minden egyéb műveletet – összeadásra vezetjük vissza:

$$S = A - B = A + (-B).$$

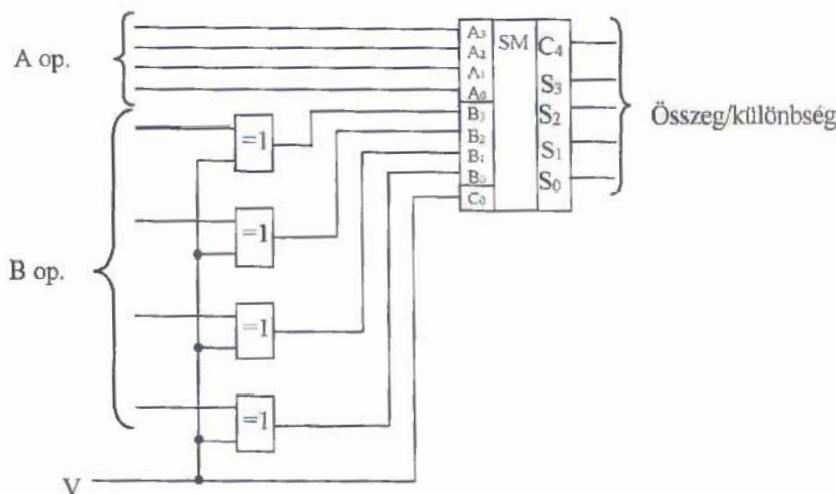
A negatív számokat kettes komplementus kódjukkal ábrázoljuk a számításokban. Ezért az előző azonosság így fogalmazható meg: két operandus különbségét úgy képezzük, hogy a kisebbítendőhöz hozzáadjuk a kivonandó kettes komplementusát. A kettes komplementus képzését már ismerjük: a szám bitenkénti negálásával előállítjuk az egyes komplementest és hozzáadunk egyet.

A pozitív és a negatív számokat előjellel különböztetjük meg egymástól a bináris számrendszerben is. A pozitív számok előjele 0, a negatív számoké 1. Kettes komplementus kódban ábrázolva a negatív számokat, a negatív számot jelölő 1 előjel azt is jelenti, hogy az előjel után következő bitsor egy bináris szám kettes komplemente. Pl.

$$+12_{10} = 0\ 1100_2; \quad -12_{10} = 1\ 0100_2.$$

A kivonás tehát így irható le:  $S = A + B_2$ .

Ezen az elven működő négy bites összeadó/kivonó áramkört mutat a 6.12. ábra.



**6.12. ábra.** Bináris összeadó kivonó áramkör

Az összeadó B bemenetein lévő antivalencia kapuk vezérelhető inverterek. Az igazságtablázatban az egyik bemeneti változót vezérőjelnek tekintve a táblázat sorait a következőképpen értelmezhetjük:

## B V F

- 0 0 0 a vezérlőjel 0, a kimeneten a B változó jelenik meg,  
 0 1 1 a vezérlőjel 1, a kimeneten a B változó negáltja jelenik meg,  
 1 0 1 a vezérlőjel 0, a kimeneten a B változó jelenik meg,  
 1 1 0 a vezérlőjel 1, a kimeneten a B változó negáltja jelenik meg.

**Összefoglalva:** az antivalencia kapu a vezérlőjel 0 értéke mellett átmásolja a kimenetére a bemenetén lévő változót, a vezérlőjel 1 értéke mellett pedig invertálja a bemeneti változót.

A 6.12. ábra antivalencia kapui  $V = 1$  vezérlésre negálják a B operendus egyes bitjeit, tehát a kivonandó egyes komplementensét képezik. A vezérlőjel az  $C_0$  bemenetre is be van kötve, ezért az összeadó a legkisebb helyi értéken az egyes komplementens-hez hozzáad az összegzés során egyet. Így egyszerre történik meg az egyes komplementensből a kettes komplementens képzése és a kivonásnak számító összeadás. Ha  $V = 0$ , akkor nincs komplementens képzés, az áramkör összeadóként működik.

Az előzőekben bináris összeadó és kivonó áramköröket ismertünk meg. Egyes alkalmazásokban előfordul, hogy BCD kódú bináris számokat kell összeadni vagy kivonni.

**BCD kódú összeadó** készítéséhez is a bináris összeadókat használjuk kiegészítő áramkörrel. A kiegészítés azért szükséges, mert a a BCD összeadás és a bináris összeadás eredménye nem egyezik meg, ha az összeg nagyobb mint 9. Két bináris és két BCD szám összeadásakor keletkező eredmények lehetséges értékeit mutatja a 6.1. táblázat.

Bináris és BCD összegek. 6.1. táblázat

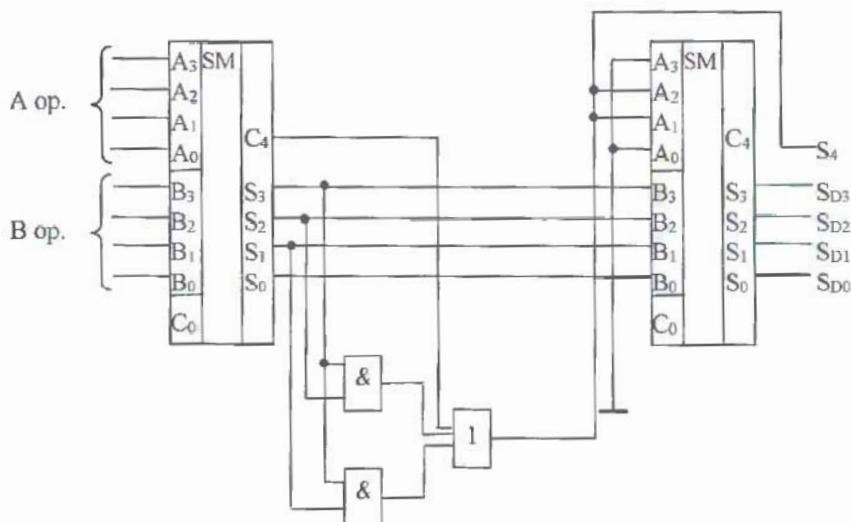
Decimális összeg	Bináris összeg					BCD összeg				
	$C_4$	$S_3$	$S_2$	$S_1$	$S_0$	$C_4$	$S_3$	$S_2$	$S_1$	$S_0$
0 (pl. 0+0)	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5 (pl. 2+3)	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9 (pl. 7+2)	0	1	0	0	1	0	1	0	0	1
<b>10</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>11</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>
<b>12</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>
<b>13 (pl. 1+12)</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>
<b>14</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>
<b>15</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>
<b>16</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>
<b>17</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>
<b>18 (pl. 9+9)</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>19 (pl. 9+9+átvitel)</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>

A táblázat elemzéséből kiderül, hogy a bináris és a BCD összegek 0-tól 9-ig megegyeznek. Ha azonban az összeg nagyobb, mint 9, akkor a **BCD összeg hattal nagyobb, mint a bináris összeg**. Ezért a bináris összeadót egy olyan áramkörrel kell kiegészíteni, amelyik figyeli a bináris összeget, és ha nagyobb, mint 9, akkor az eredményt korrigálja. A hat hozzáadását BCD korrekciójának nevezzük.

Ahhoz, hogy az összeg nagyobb legyen mint 9, a táblázat alapján az szükséges, hogy a bináris összeg  $C_4$  bitje 1 legyen, vagy az  $S_3$  bit 1 értéke mellett még az  $S_2$  vagy az  $S_1$  bit is 1 legyen. A korrekció szükségességét figyelő függvény ezért:

$$F_K = C_4 + S_3(S_2 + S_1) = C_4 + S_3 \cdot S_2 + S_3 \cdot S_1.$$

A függvényt megvalósító kombinációs hálózattal felépített BCD összeadó kapcsolási rajza a 6.13. ábrán látható.



6.13. ábra. A BCD összeadó kapcsolási rajza

Az első összeadó kimenetén a bináris összeg jelenik meg. Ha ennek értéke nagyobb, mint 9, akkor a kombinációs hálózat kimenetén 1 szint lesz. A második összeadó A bemenetein ilyenkor bináris 6 jelenik meg:  $A_0 = 0, A_1 = 1, A_2 = 1, A_3 = 0$ . A második összeadó kimenetein a korrigált BCD összeg jelenik meg. Ha a bináris összeg kisebb, mint 10, akkor a kombinációs hálózat kimenetén 0 szint jelenik meg, vagyis a második összeadó nullát ad a bináris összeghez.

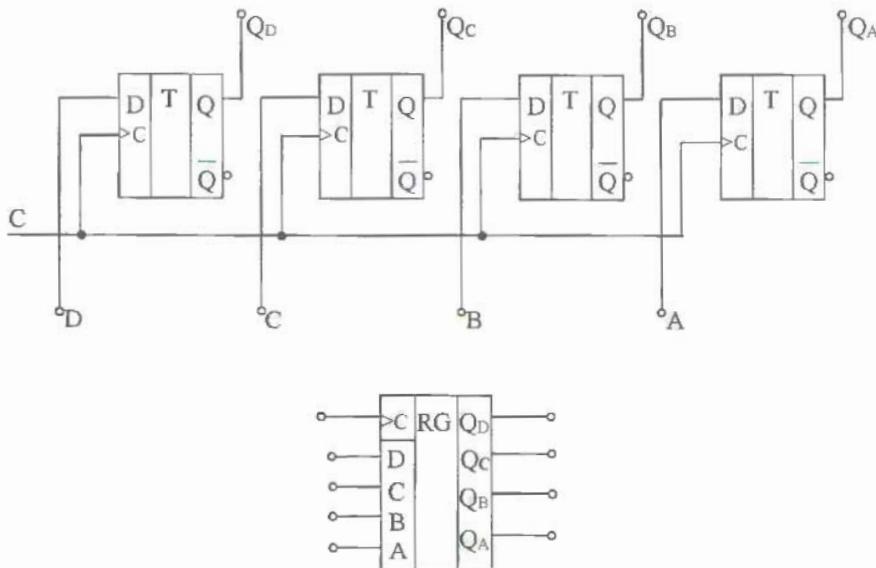
A BCD kivonó áramkör hasonló elven készíthető.

## 6.4. Regiszterek

A regiszterek **több bit egyidejű tárolására** alkalmas áramkörök. A tárolási funkciót D flip-flopok látják el, amiket az integrált áramkörön belül gyakran más tárolótípusokból (pl. R-S) alakítanak ki. A regiszterek egyes típusai a tárolási feladaton kívül a tárolt információ **léptetésére** is alkalmasak. Így a regiszterek két típusa:

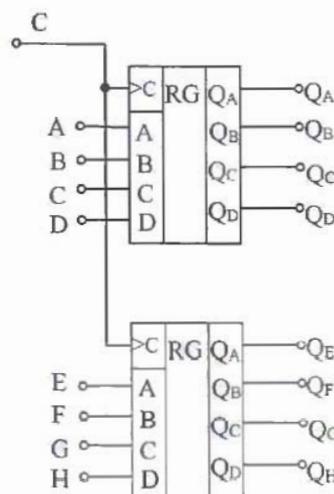
- átmenneti tárolók,
- léptetőregiszterek.

Az **átmenneti tárolók** (pufferregiszterek, latch tárolók) minden flip-flopjába egyszerre (párhuzamosan) történik az információ beírása. A beírást az órajel vezéri. A tárolt információ a regiszter kimenetén hozzáérhető. A **6.14.** ábra példaként egy **4 bites pufferregiszter** belső felépítését és jelképi mutatja.



**6.14.** ábra. Pufferregiszter

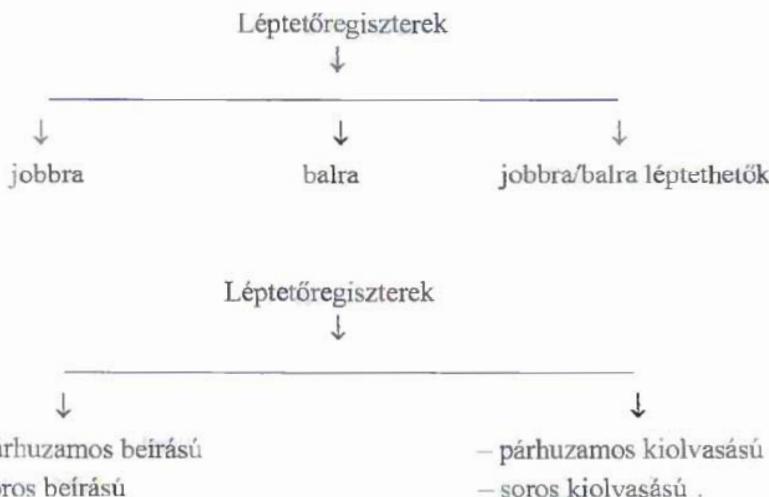
Ha a tárolandó információ több bitet tartalmaz, mint amennyit egy regiszter képes tárolni, akkor több regiszterből készítünk nagyobb kapacitású (tárolóképességű) kaszkádossással. Két négybitcsból összeállított 8 bites regisztert mutat a **6.15.** ábra.



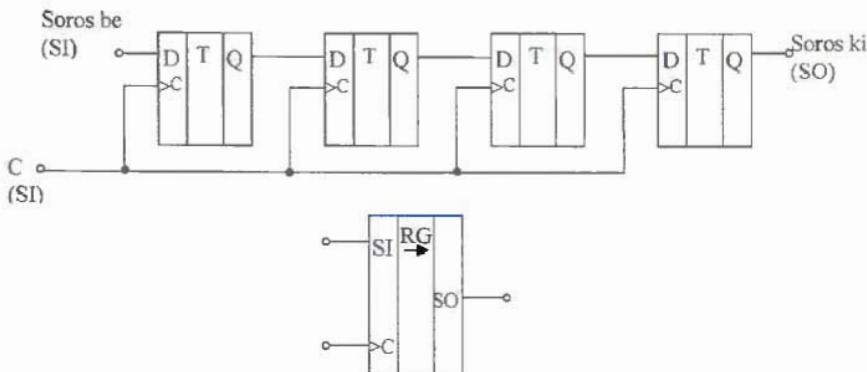
**6.15.** ábra. A regiszterek kapacitásának bővítése

Az SN sorozatban igen sok pufferregiszter van, pl. 7475, 74100. A CD sorozat egyik pufferregisztere pl. a 4076 típus.

A **léptetőregiszterek** a léptetés iránya, valamint az információ beírása és kiolvasása szempontjából csoportosíthatók:



**A 6.16. ábra** egy 4 bites, **jobbra léptethető, soros beírású, soros kiolvasású** léptetőregisztert mutat.

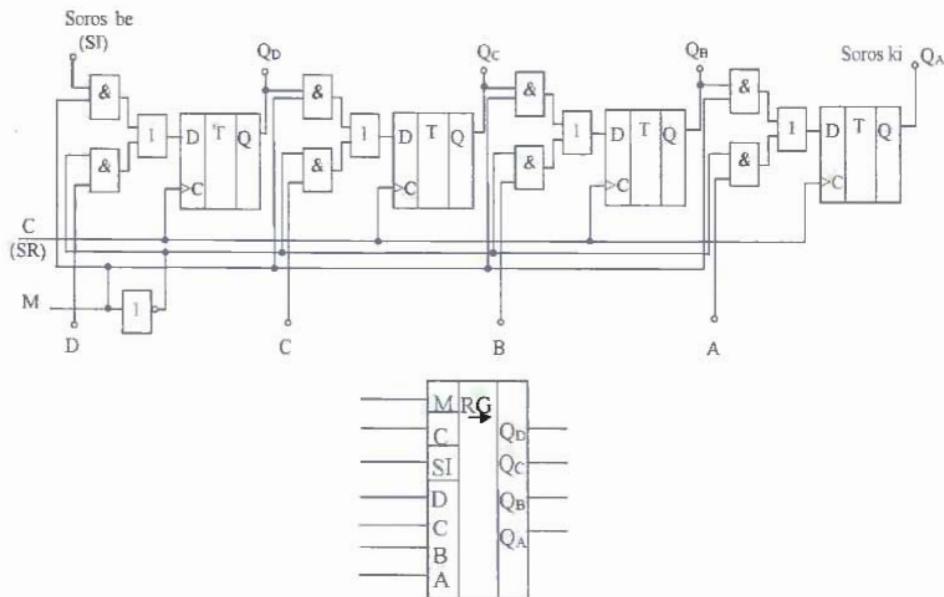


**6.16. ábra.** Jobbra léptethető, soros be/soros ki regiszter

Az információ egyes bitjeinek beírása az SI (*serial input* – soros bemenet) bemeneten át történik és a beírt bitek léptethetők az órajellel. Ez tehát a jobbra léptetést vezérlő SR (*shift right* – léptetés jobbra) bemenet. A regiszteren való végiglćptetés után az információ bitjei sorasan az SO (*serial output*) soros kimeneten jelennek meg.

Az SN sorozatban jobbra léptethető soros be/soros ki regiszter a 7491, A CD sorozatban a 4006 típus.

**Párhuzamos és soros beírású, párhuzamos és soros kiolvasású, jobbra léptethető** regisztert mutat a **6.17. ábra**.

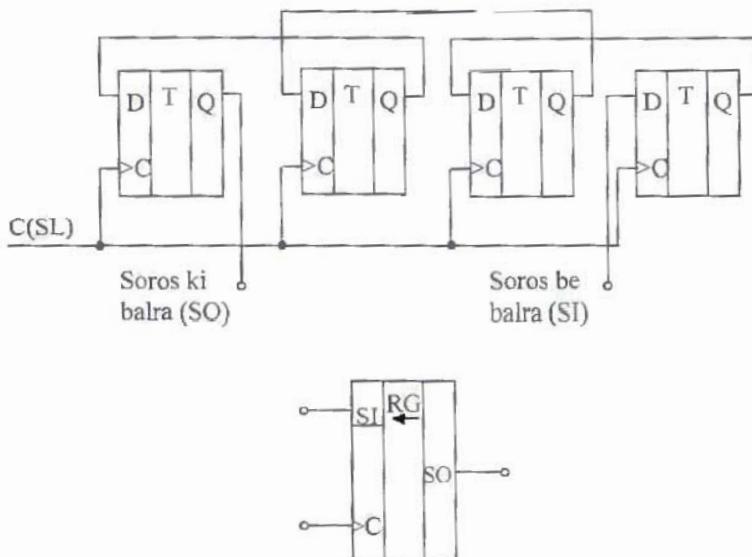


**6.17. ábra.** Jobbra léptethető, párhuzamos/soros be, párhuzamos/soros ki regiszter

A flip-flopok között lévő kombinációs hálózat az M (mode) bemenetre adott vezérléstől függően a párhuzamos beírást vagy a léptetést engedélyezi: M = 0-ra léptetés, M = 1-re párhuzamos beírás. Párhuzamos beírás és léptetés tehát egyszerre nem lehetséges. A léptetés az órajellel történik. A beírt és léptetett információhoz párhuzamosan a Q kimeneteknél lehet hozzáérni. A regisztert lehet kombináltan is használni, mert az SI bemeneten soros beírás is lehetséges. Ha csak a Q<sub>A</sub> kimenetet használjuk, akkor a regiszter soros kiolvasású. Ilyen regiszter pl. az SN 7495.

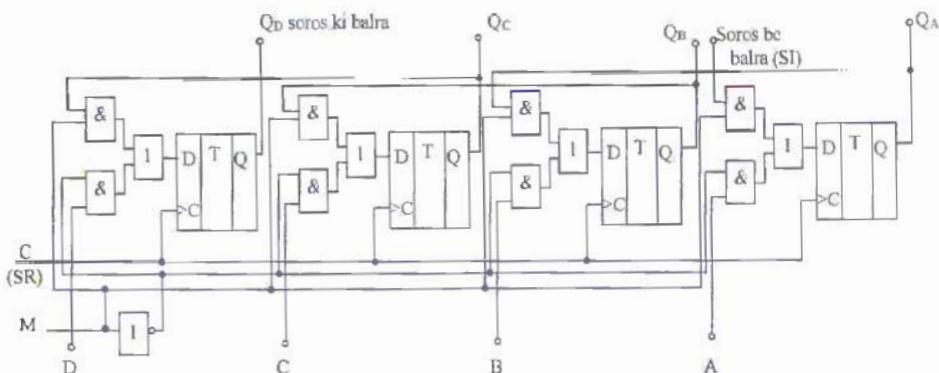
Ha a 6.17. ábrán látható kapcsolásból, a Q<sub>A</sub> kimenetet kivéve, elhagyjuk a párhuzamos kimeneteket, akkor egy **soros/párhuzamos beírású, soros kiolvasású jobbra léptethető** regiszterhez jutunk. Ilyen regisztert tartalmaz pl. az SN 7494, a CD 4014.

**Balra léptethető, soros beírású, soros kiolvasású** regisztert mutat a 6.18. ábra. Ebben az esetben az órajel balra lépteti az A flip-flop bemenetére kerülő biteket. A regiszteren végigléptetett bitek a D flip-flop kimenetén lépnek ki. A léptetés az SL (Shift left) bemenetre adott órajelekkel lehetséges.



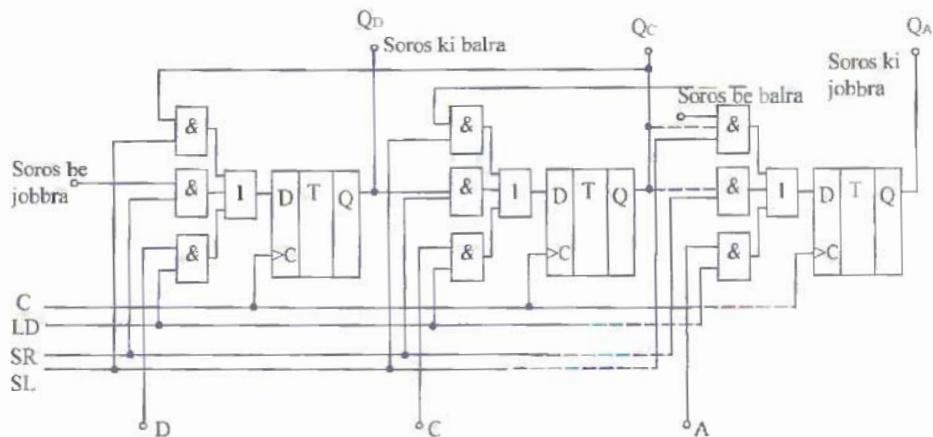
6.18. ábra. Balra léptethető soros be/soros ki regiszter

**A balra léptethető, párhuzamos/soros beírású és párhuzamos/soros kiolvasású** regiszter kapcsolási rajza a 6.19. ábra szerinti. A be- és kimenetek funkciója és az áramkör működése az előzőek alapján azonosítható.



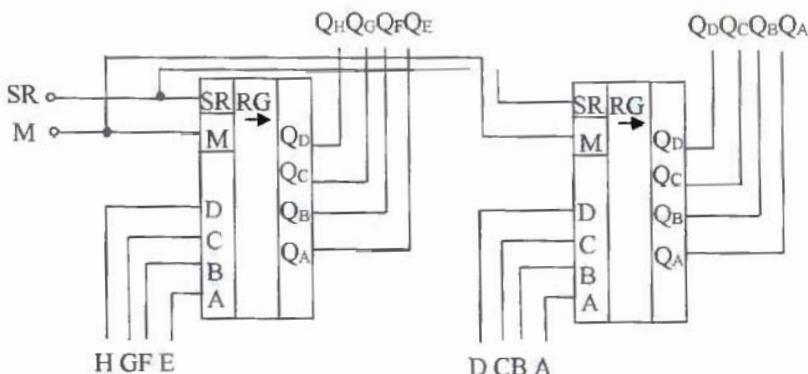
6.19. ábra. Balra léptethető, párhuzamos/soros bc, párhuzamos/soros ki regiszter

Valamennyi eddig megismert funkciót egyesíti a **jobbra-balra léptethető, soros-párhuzamos beírású, soros-párhuzamos kiolvasású** léptetőregiszter. Kapcsolási rajzának egy részlete a 6.20. ábrán látható. A párhuzamos beírást az  $Ld$  (*Load* – töltés) bemenet engedélyezi.



6.20. ábra. Jobbra/balra léptethető, soros/párhuzamos be, soros/párhuzamos ki regiszter

**A léptetőregiszterek kaszkádosítása** a vezérlőbemenetek párhuzamosításával és az információ bitjeinck az egyik tokból a másikba való átlépést biztosító összckötéssel valósítható meg. A 6.21. ábra példaként két négybites, párhuzamos beírású és kiolvasású, jobbra léptethető regiszter kaszkádosítását mutatja nyolc bitre.



6.21. ábra. A léptetőregiszterek kaszkádosítása

**A regiszterek felhasználási területe** igen széleskörű, szinte a leggyakrabban használt digitális áramkörök. A pufferregisztereket minden olyan helyen használjuk, ahol kevés adatot kell rövid ideig tárolni. Pl. a kijelző adat tárolására a kijelzés időtartamára, vagy műveletvégző egységekben a műveletben felhasznált adatok tárolása a műveletvégzés idejére.

A léptetőregiszterek típusik alkalmazása a soros-párhuzamos, ill. a párhuzamos-soros átalakítóként való alkalmazás. Az első esetben a soros bemenetre érkező biteket egyenként beléptetjük a regiszterbe, majd párhuzamos kimeneteken keresztül valamennyi bithez egyszerre hozzáférhetünk.

A párhuzamos-soros átalakítónál párhuzamos beíró bemenetekkel és soros kimenettel rendelkező léptetőregisztert használunk. A párhuzamos bemeneteken egy lépéssel birt információt a léptetőjel ütemében kiléptetjük a soros kimeneten.

## 6.5. Számlálók

A számlálók feladata az órajel bemenetükre érkező impulzusokat megszámlálása és a számlálás eredményének kijelzése. A kijelzés minden a bemenetre érkező impuluszszámot kódoló bináris szám formájában történik. A kijelzhető mennyiség meghatározza a megszámolható impulzusok maximális számát. A számláló egyik jellemző adata ezért a **számláló modulusa**: a számláló kimenetén az egymástól megkülönböztethető állapotok száma. Pl. a két kimenettel rendelkező számlálók kimeneti állapotai:

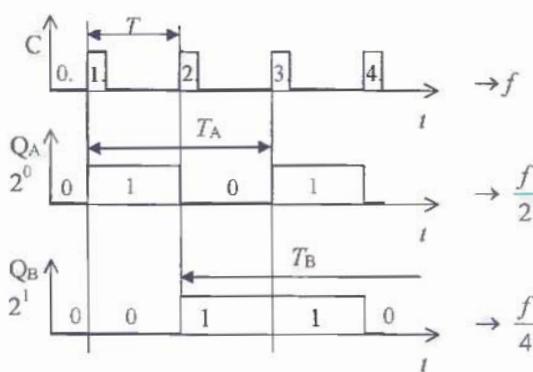
Impulzusok száma	Kimeneti állapot	
C	$Q_B(2^1)$	$Q_A(2^0)$
0	0	0 alapállapot,
1	0	1
2	1	0
3	1	1 végállapot.

A számláló tehát 0-tól 3-ig számol és jelez ki, a kimenetén megkülönböztethető állapotok száma és ezért a számláló modulusa 4.

A kimeneti állapotok előző felsorolásából következik a **számlálás elve** is:

- $Q_A$  kimenet, vagyis a legkisebb helyi értékű kimenet, minden megszámlálálandó impulzusra megváltoztatja előző állapotát,
- a  $Q_B$  kimenet, vagyis a következő helyi értékű kimenet, minden második impulzusra változtatja meg kimeneti állapotát. Ez az előző helyi érték állapotával megfogalmazva azt is jelenti, hogy az előző helyi érték  $1 \rightarrow 0$  átmenete hozza létre a következő helyi érték új állapotát,
- a számláló a modulusának megfelelő végállapot elérése után érkező további impulzusok hatására újra alapállapotból számol tovább. Működése tehát ciklikus.

A 6.22. ábrán a számlálás elve idődiagramon követhető.



6.22. ábra. A számlálás elve

Az idődiagram is az előző megállapításainkat támasztja alá: a  $Q_A$  kimenet minden órajelre állapotot vált, a  $Q_B$  kimenet állapotának változása a  $Q_A$  lefutásához kötődik. Az ábrából az is látható, hogy a legkisebb helyi értékű kimeneten megjelenő impulzussorozat periódusideje kétszer akkora, a következő helyi értéké pedig négy-szer akkora, mint az órajel periódusideje.

Ez a kimeneti jelek frekvenciáját tekintve azt jelenti, hogy az órajel frekvenciához képest a legkisebb helyi értéken  $f/2$ , a következő helyi értéken  $f/4$  frekvenciájú négy-szög impulzussorozat jelenik meg. A kimenetek egymáshoz viszonyított frekvenciaaránya: a nagyobb helyi értéken fele akkora frekvenciájú a kimeneti impulzussorozat, mint az előző.

Mindkét megközelítésből nyilvánvalóan adódik, hogy a következtetések általánosíthatók: a következő helyi értékű kimenet akkor változik, amikor az előző állapotot  $1 \rightarrow 0$  állapotváltása bekövetkezik. A kimenetek frekvenciában mért osztásviszonya kettő hatványai szerint növekszik a helyi érték növekedésével. A következő helyi érték minden felezi az előző helyi érték kimeneti jelének frekvenciáját.

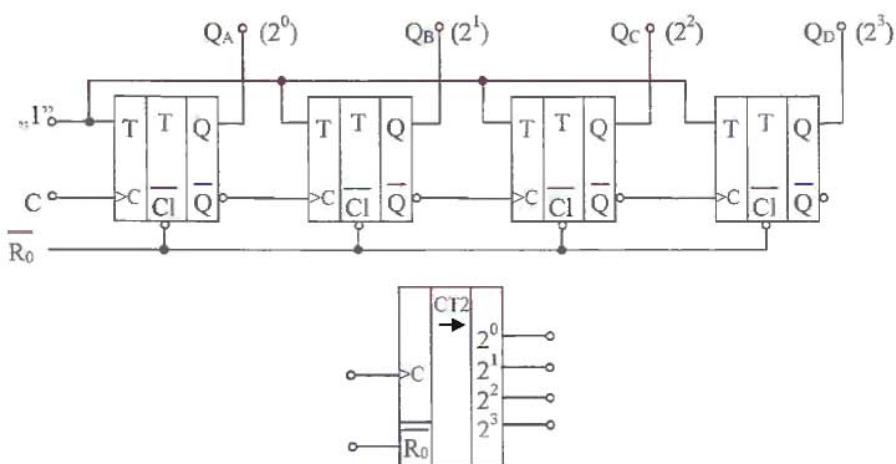
A tárolók vezérlési táblázatain végigtekeintve megállapíthatjuk, hogy az elvezérelt T típusú tároló a T bemenetére adott 1 vezérlés mellett minden órajel hatására megváltoztatja kimeneti állapotát. A számláló minden helyi értékén elvezérelt T flip-flopot alkalmazva a számlálás alapelve szerint működő áramkör adódik. **A számlálók alapeleme ezért az elvezérelt T tároló.** A digitális integrált számlálók egyes típusaiban nem közvetlenül T tárolót használnak, hanem J-K, vagy R-S tárolókhól alakítanak ki a T tárolóval megegyező funkciójú flip-flopokat.

A számlálók sokféle típusát a következők szerint csoportosíthatjuk:

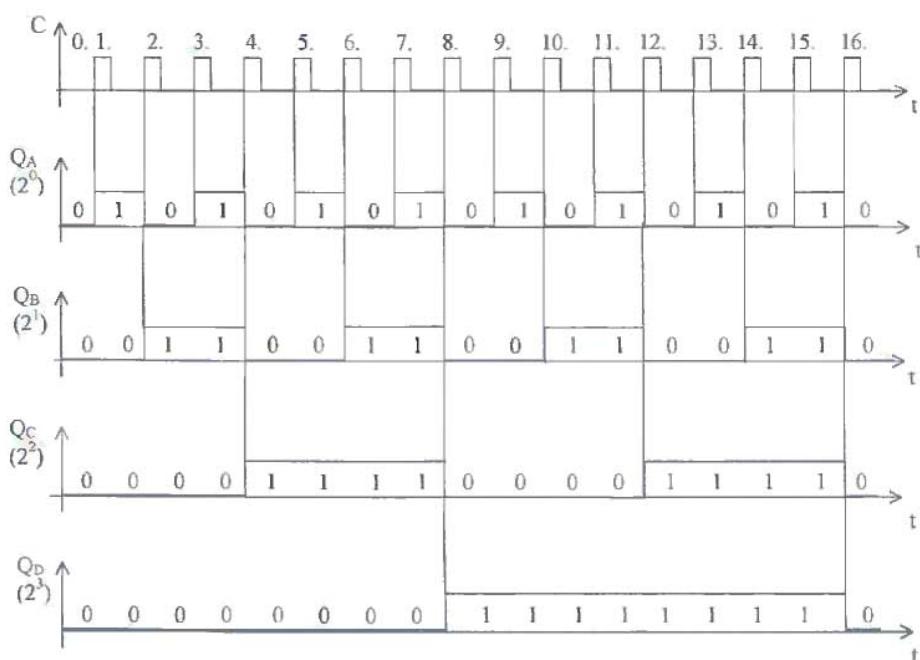
a belső felépítés szerint	a kijelzés kódja szerint	a számlálás irányá szerint	a kezdeti érték beállítása szerint
– aszinkron	- bináris	– előre (felfelé)	– párhuzamos beírású, sztatikus törlésű
	- BCD	– hátra (lefelé) – reverzibilis (fel/le)	– sztatikus törlésű

A leggyakrabban használt belső felépítésű számláló az **aszinkron, bináris, előreszámító, sztatikus törlőbemenettel**. A 6.23. ábra négy bites számlálót mutat, jelképi jelölésével együtt. A jelképi jelölésben szereplő CT (*Counter* – számláló) a számláló funkciójel. A mellette szereplő 2 bináris számlálót jelöl.

A T flip-flopok pozitív elvezéreltek, ezért az első órajel bemérésére közvetlenül csatlakozik az órajelbemenet. Erre kerülnek a megszámlálálandó impulzusok. A pozitív elvezérlés miatt, következő tárolót az előző  $\bar{Q}$  negált kimenete vezéri, mert a 6.22. ábrából következően, amikor a Q kimenet lefut, akkor kell a következő tárolónak billenni. A számláló működése aszinkron mert csak az első flip-flopot vezérl az órajel, a további tárolók egymástól kapják a vezérlést. A számlálás idődiagramját a 6.24. ábra mutatja a belső késleltetéseket figyelmen kívül hagyva.



6.23. ábra. Négybites aszinkron előrc számláló



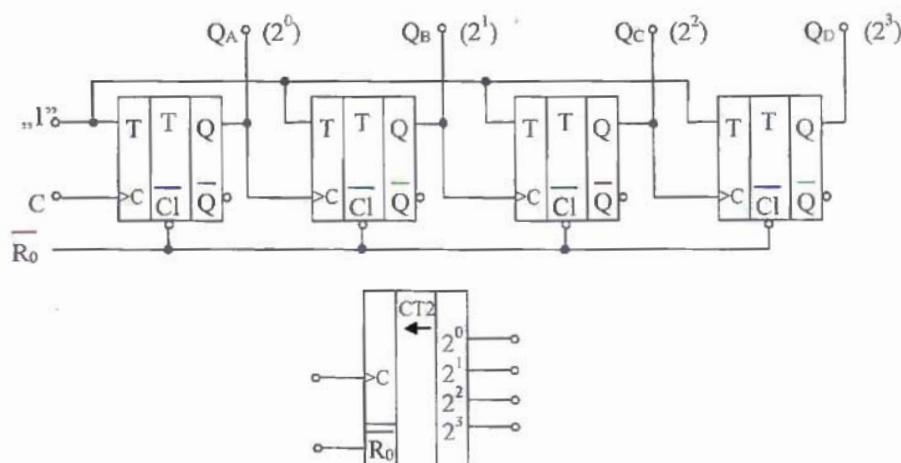
6.24. ábra. A négybites előre számláló idődiagramja

Az  $\bar{R}_0$  sztatikus törlőbemenet a tárolók  $\bar{Cl}$  sztatikus törlőbemenetére csatlakozik, tehát az órajeltől függetlenül egyszerre töri a flip-flopokat. Ez azt jelenti, hogy

számlálás közben bármikor törölhető, alap állapotbaállítható a számláló. Az órajel-től független sztatikus törlést **aszinkron sztatikus törlésnek** nevezzük.

Az ismertetetthez hasonló típus az SN 7493.

A visszaszámítók az alapállapotból az első órajel hatására végállapotba kerülnek, majd a további órajelek hatására csökkentik a kimenetükön megjelenő bináris értéket. A 6.25. ábrán látható kapcsolás egy **négybites aszinkron visszaszámító, aszinkron sztatikus törlőbemenettel**.



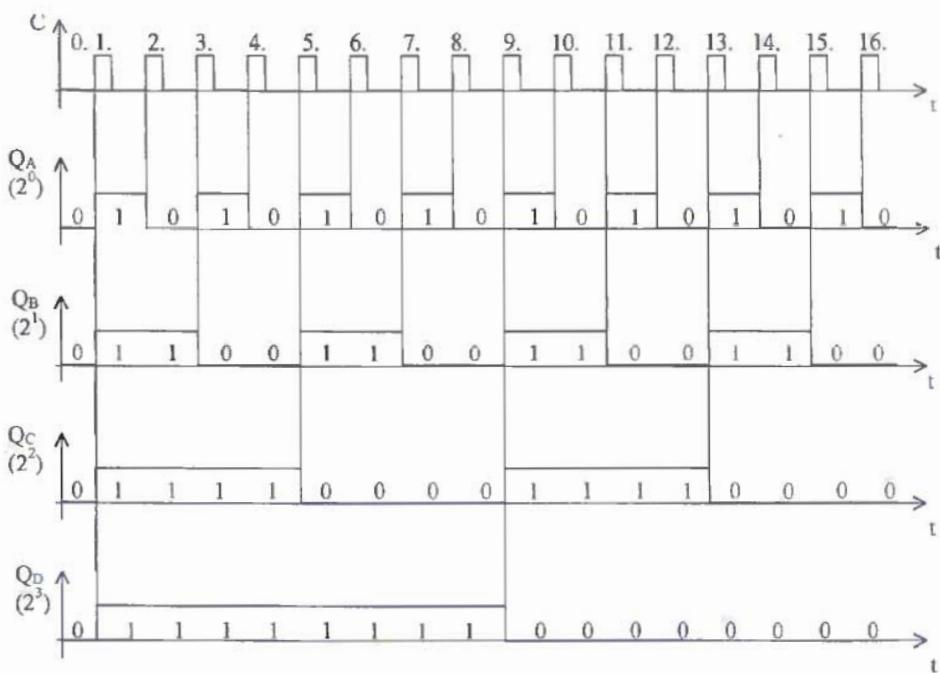
6.25. ábra. Négybites bináris visszaszámító

A pozitív élvezérelt T flip-flopok az előző Q kimenetről kapják vezérlésüket, ezért az első órajel hatására valamennyi tároló billen, az órajel felfutó éle végigfut a számlálón. Ezt és a további órajelekre bekövetkező változásokat mutatja a 6.26. ábra.

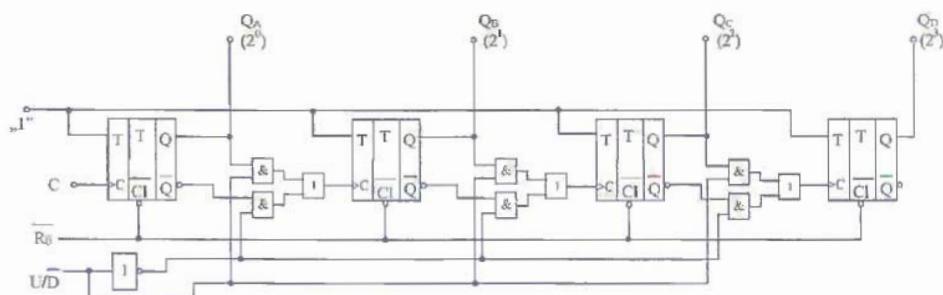
Az aszinkron számlálók felépíthetők negatív élvezérelt tárolókból is. Ilyen felépítés mellett az első tároló invertálva kapja az órajelet és a további tárolók vezérlése a Q kimenetekről történik.

**Négybites reverzibilis** (kétirányú) **aszinkron számláló, sztatikus aszinkron törlőbemenettel** látható a 6.27. ábrán. Szokásos elnevezések még: fel/le (*up/down*) számláló, oda/vissza számláló.

A tárolók között lévő kombinációs hálózat a számlálási irányt vezérlő U/ $\bar{D}$  bemenet állapotától függően engedélyezi az órajel bemenetek vezérlését az előző tároló  $\bar{Q}$  vagy Q kimenetről.



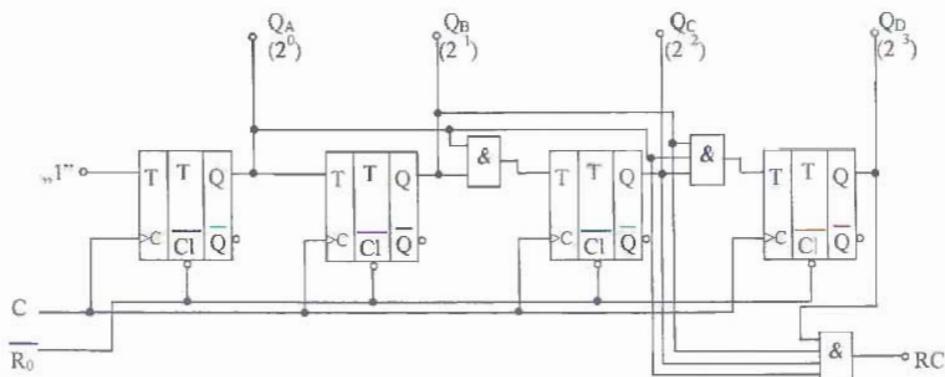
6.26. ábra. Bináris visszaszámító idődiagramja



6.27. ábra. Négy bites reverzibilis számláló

A **szinkron számlálók** közös jellemzője, hogy a flip-flopok egyszerre kapják meg a számlálálandó órajelet. Így tehát minden szinkron számláló egy szinkron szekvenciális hálózat, amelynek állapotdiagramja a számlálási ciklust tartalmazza. A szinkron számlálók a 7.3.2. pontban megismert tervezési módszerrel tervezhetők. A részletes tervezéstől itt eltekintünk, a továbbiakban csak a kész áramkört vizsgáljuk.

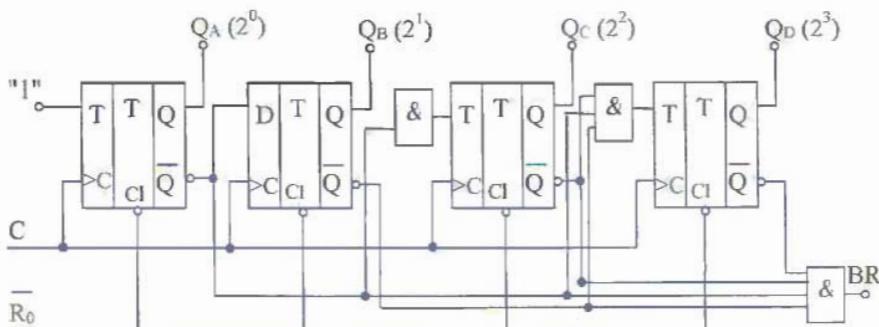
A 6.28. ábrán egy négy bites szinkron előreszámító látható sztatikus aszinkron törlö bemenetekkel.



6.28. ábra. Négy bites szinkron előrc számláló

A T bemenetek vezérlését ellátó kapuáramkörök tulajdonképpen a szckvenciális hálózat tervezése során adott vezérlési függvényeket valósítják meg. Szóban úgy fogalmazható meg a tervezés eredménye, hogy az órajel hatására a következő tárolónak akkor kell billennie, ha az előző tárolók kimeneteinek mindegyike már 1 szinten van (egy következő tároló csak akkor billen, ha az előzőök már betelték). Ez a vezérlés biztosítja a 6.22. ábra szerinti számlálást. Az RC kimenettel rendelkező ÉS kapu a szinkron számlálók kaszkádosságát segíti, mert előállítja a következő – már másik tokban lévő – tároló vezérlését.

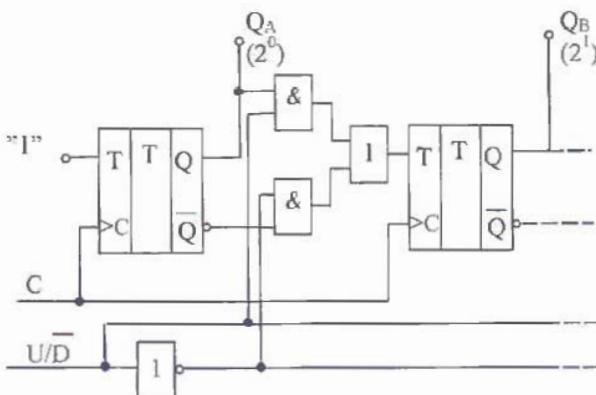
**Szinkron bináris visszaszámító** hasonló elven készíthető, de a T bemeneteket vezérő ÉS kapukat az előző tárolók negált kimenetéről kell vezérelni, mert visszaszámításkor akkor kell egy tárolónak billeni, amikor már az összes előző tároló kimenete 0 szintű. A 6.29. ábra egy szinkron bináris visszaszámítót mutat.



6.29. ábra. Szinkron bináris visszaszámító

A BR (*Borrow* – áthozat, szó szerint: kölcsön) kimenetű ÉS kapu a kaszkádosításhoz szükséges: előállítja a következő tokban lévő első tároló számára a T bemenetet vezérlő jelet.

**Szinkron reverzibilis számláló** T tárolónak vezérlése a számlálási irány kijelölésétől függően az előző tároló Q vagy  $\bar{Q}$  kimenetéről történik. A bonyolult teljes kapcsolás helyett a **6.30.** ábra példaként a  $2^1$  helyi értéken lévő tároló vezérlését mutatja.

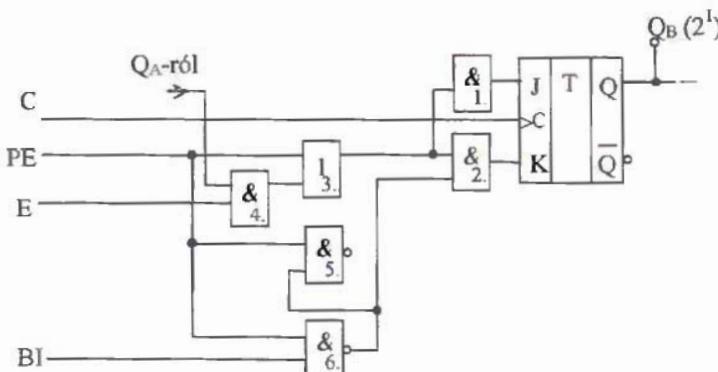


**6.30.** Szinkron reverzibilis számláló tárolójának vezérlése

A szinkron reverzibilis számlálók is rendelkeznek olyan kimenettel, amely a kaszkádosítást segíti. Ez a tokon belül a RC és BR kimenetek VAGY kapuval való közössítésével jön létre. Ez a kimenet tchát 1 szintet ad akkor is, amikor előre számlálásnál a szinkron számláló elérte a végállapotot, és akkor is, amikor hátra számlálásnál alapállapotba került. A közösített kimenetet általában M/m (Maximum/minimum) jelöléssel látják el.

A szinkron számlálók nagy része rendelkezik párhuzamos beíró bemenetekkel is. Ezekben a bemeneteken keresztül, a párhuzamos beírás engedélyezése mellett, a tárolók tartalma tetszőleges értékre állítható. A számlálás a következő órajel hatására erről az értékről folytatódik. A párhuzamos beírás egyes számlálótípusoknál az órajellel szinkronban lehetséges, más típusoknál az órajeltől függetlenül, aszinkron módon. A párhuzamos beírás megkönnyítésére a T tárolókat J-K tárolókból célszerű kialakítani, amelyek bemenetei számláláskor összekapcsolódnak, így T tárolóként viselkednek, beíráskor viszont J-K típusúak.

A **6.31.** ábra egy szinkron párhuzamos beírású, szinkron előreszámító  $2^1$  helyi értékén lévő tároló vezérlését mutatja.



6.31. ábra. A párhuzamos beírás elve

Az ábra szerinti megoldásban a párhuzamos beírás szinkron típusú, mert a J-K bemenetekre hat, amelyek viszont csak órajelre veszik figyelembe a vezérlést.

A PE (*parallel enable*) bemenet a párhuzamos engedélyező bemenet, az E (*enable*) bemenet a számlálót engedélyezi, a BI (*B-input*) a B tároló párhuzamos beíró bemenete. Számláláskor az E bemenet engedélyezett:  $E = 1$ , a párhuzamos beírás nem:  $PE = 0$ . Az 5. és 6. kapu tiltva van, kimenetük 1 szintű, ezért az 1. és 2. kapu engedélyezett. Az  $E = 1$  engedélyezi a 4. kaput, ezért a  $Q_A$  kimenetről jövő felfutó él a 4. és 3. kapun keresztül az engedélyezett 1. és 2. kapuk kimenetére jut. A J és a K bemenetet tehát együtt vezérli a  $Q_A$  kimenet. Ilyenkor tehát a J-K tároló egy T flip-flopnak számít. Párhuzamos beírás úgy lehetséges, ha  $PE = 1$ . Ez a vezérlés a 3. VAGY kapun keresztül engedélyezi az 1. és 2. kapukat, lehetővé téve a J és K bemenetek vezérlését és engedélyezi az 5. és 6. kapukat is.

Ha a BI bemeneten pl. 1-et akarunk beírni, akkor a 6. NAND kapu kimenetén az invertálás miatt 0 szint lesz, ami a K bemenetre jut. Az 5. NAND kapún keresztül, újabb invertálás után, a J bemenet 1 szintet kap. Végeredményben tehát a tároló vezérlése:  $J = 1$ ,  $K = 0$ , ezért a tároló beíródik. A következő órajel már erről a beírt értékről billenti a tárolót.

Ha a J-K tárolóba 0-t akarunk beírni, akkor a 6. kapu miatt  $K = 1$  vezérlést kap, az 5. kapun keresztül a J bemenet pedig 0-t. A tároló tehát törlődik (beíródik 0).

Párhuzamos beírási lehetőséggel rendelkezik az igen gyakran használt SN 74161 és SN 74163 bináris számláló. A párhuzamos beírás a 6.31. ábrának megfelelő. A CD sorozatban párhuzamos beírású számlálója pl. a CD 40161 és a CD 40163.

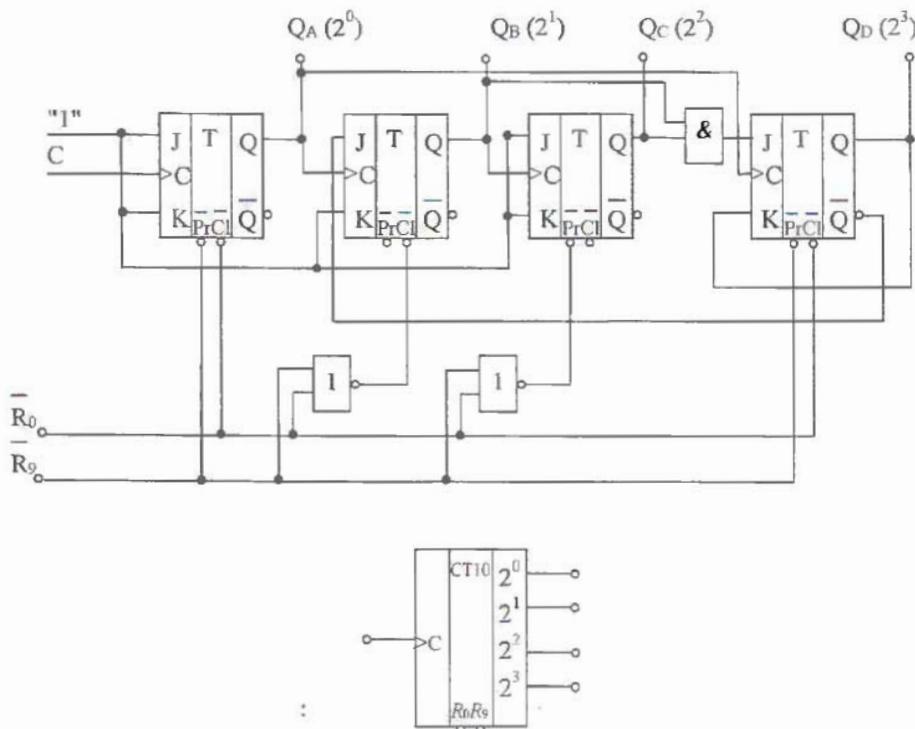
A **sztatikus törlő bemenetek** feladatát a korábbi kapcsolásokban már láttuk: a számlálás közben alapállapotba állítható a számláló. Általában az integrált áramkörökbén az előző ábrákon pl. 6.23., 6.25. ábrák szereplő megoldást alkalmazzák,

tehát az  $R_0$  bemenetre adott vezérlés a sztatikus clear bemenetekre hat. Ezek órajel-től függetlenek, ezért a sztatikus törlés is aszinkron. Létezik néhány olyan számláló – pl. SN 74163, CD 40163 –, amelynél a sztatikus törlés is csak az órajellel együtt hatásos. Az ilyen számlálók a **szinkron sztatikus törlésű számlálók**.

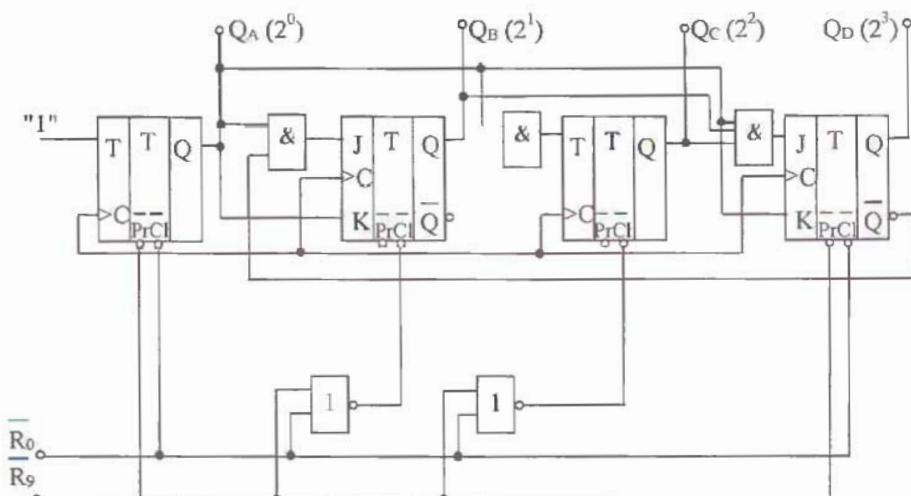
A **BCD kódú számlálók** modulusa 10, a megszámolt impulzusokat 0-tól 9-ig képe-sek megszámolni és négy biten, tehát BCD kódban kijelezni. Gyakran dekádzámlálónak nevezzük, mert több BCD számlálót egymás után sorolva (kaszkádosítva) minden számláló a teljes érték egy dekádját (BCD helyi értékét) számolja meg és jelzi ki.

A BCD számlálók úgy készülnek, hogy egy négybites bináris számláló számlálási ciklusát lerövidítik úgy, hogy max. csak 9-ig számoljanak. A BCD számlálók funk-ciójele CT 10.

A 6.32. ábra egy aszinkron, a 6.33. ábra pedig egy szinkron BCD számláló belső felépítését mutatja. Az áramkörök működése a kapcsolások analízisével határoz-ható meg.



6.32. ábra. Aszinkron BCD számláló



6.33. ábra. Szinkron BCD számláló

A 6.32. ábra kétféle sztatikus beállító bemenetet mutat. Az R<sub>0</sub> a már ismert törlőbemenet, amely az összes tárolót törli. Az R<sub>9</sub> bemenet csak BCD számlálóknál létezik. A tárolókat a Pr sztatikus beíró bemeneteit felhasználva az 1001 állapotba állítja.

A szinkron és az aszinkron számlálók működését összehasonlíthatjuk, hogy a szinkron számlálóknál valamennyi kimenet egyidőben veszi fel az új értéket, az aszinkron számlálók kimenetei viszont egymás után, időkéséssel. Az órajel hatásos éléhez képest a szinkron számláló összes kimenetén annyi idő múlva jelenik meg az új érték, amennyi egy tároló és a bemeneteit vezérlő kombinációs hálózat együttes jelkészletettségi ideje. Pl. a 6.28. ábra szinkron számlálójánál a max. jelkészletetés egy kapu és egy tároló jelkészletetési idejének összege. Ahhoz, hogy a számláló helyesen működjön, ez idő alatt nemérkezhet újabb órajel a bemenetre, hiszen még az előzőnek megfelelő értéket sem vette fel a számláló kimenete. A kapu készletettségét pl. 10 ns-nak, a tároló készletettségt 15 ns-nak véve, az órajel periódusidejét 25 ns-nál nagyobbra kell választani. Ez frekvenciára átszámítva:

$$f_{\max} = \frac{1}{2.5 \cdot 10^{-8}} = 4 \cdot 10^7, \quad f_{\max} = 40 \text{ MHz}.$$

Az aszinkron számlálók az órajel hatásos éle után csak akkor veszik fel a kimeneteiken az új értéket, amikor már valamennyi tárolón végigfutott a vezérlés. Ez pl. a 6.23. ábra aszinkron számlálója esetén a négy tároló készletetési idejének összege. Itt is 15 ns-mal számolva, az órajel periódusideje minimálisan 60 ns. Az órajel maximális frekvenciája így:

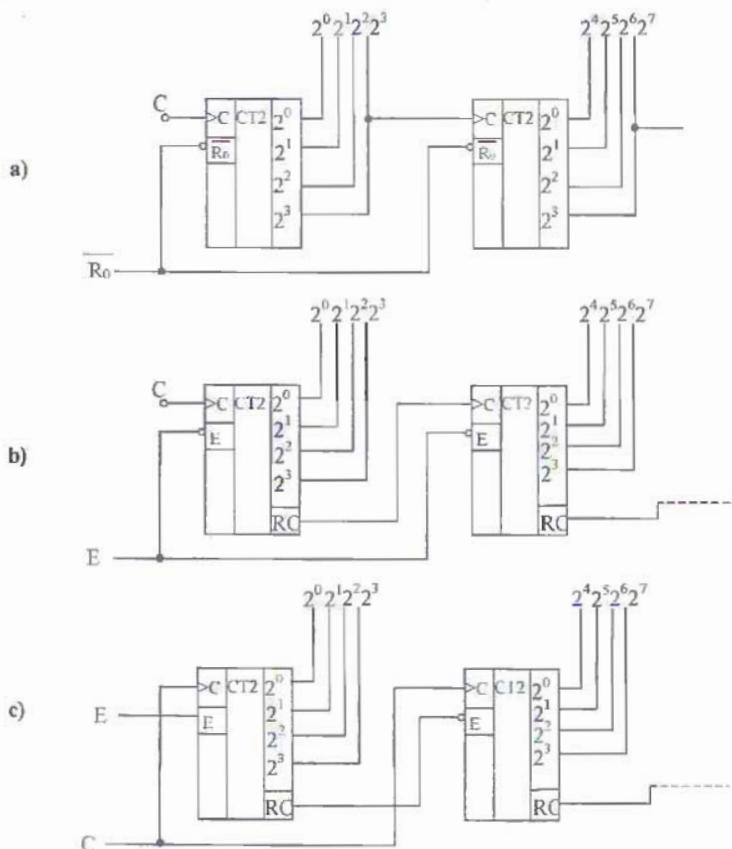
$$f_{\max} = \frac{1}{6 \cdot 10^{-8}} = 1,66 \cdot 10^7, \quad f_{\max} = 16,6 \text{ MHz.}$$

A feladatból levonható az a következtetés, hogy **az aszinkron számlálók kisebb frekvenciáig használhatók, mint a szinkron számlálók.**

A számlálók eredeti számlálási ciklusának módosítása két szempontból merülhet fel:

- egy számláló integrált áramkör (tok) modulusa kisebb, mint az adott feladat szerint szükséges impulzusszám. Ebben az esetben több számlálót kell összekapcsolni, kaszkádosítani,
- a számláló vagy a kaszkádosított számlálólánc modulusát kisebbre kell beállítani, mint az eredeti érték.

A kaszkádosítás lehetőségeit mutatja a 6.34. ábra aszinkron és szinkron előre számlálóknál.



6.34. ábra. A számlálók kaszkádosítása

- a) Aszinkron számlálók kaszkádosítása,    b) Szinkron számlálók kaszkádosítása,  
c) Szinkron számlálók szinkron kaszkádosítása

A **kaszkádosítás** elve az, hogy a tokok között lehetőleg olyan legyen a kapcsolat, mint a tokon belüli tárolók között. Ez az elv az aszinkron számlálóknál úgy valósítható meg, hogy a legnagyobb helyi értékű kimenetet továbbvezetjük a következő tok órajel bemenetére.

A szinkron számlálók kaszkádosítása elvégzhető aszinkron és szinkron módon. Mindkét esetben a számlálók végállapotát jelző RC kimenetet használjuk. Az aszinkron kaszkádosítás kedvezőtlen megoldás, mert megbontja a tokon belüli szinkron működést. A kedvező megoldás a szinkron módon való kaszkádosítás, mert így a tokok is egyszerre kapják az órajelét, csakúgy, mint a számlálókon belül a tárolók.

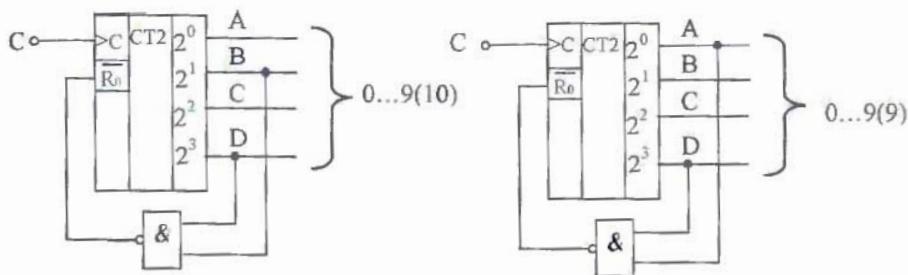
A számlálók **ciklusrövidítése** úgy valósítható meg, hogy egy áramkörrel figyeljük az előírt végállapotot és amikor a kimenet ezt elérte, töröljük a számlálót a sztatikus törlőbemeneten keresztül. A törlés módja a számláló típusától függően lehet aszinkron és szinkron sztatikus törlés.

Az aszinkron törlésű számlálóknál a kimeneteket figyelő áramkörnek az előírt végállapotnál egygel nagyobb kimeneti értéket kell figyelni. Ha a végértéket figyelnénk, akkor ez nem jelenne meg a kimeneten egy teljes órajel periódusra, mert az aszinkron törlés azonnal hatásos.

Az egygel nagyobb érték figyelése esetén a végállapot megmarad egy teljes periódusra a kimeneten, de a következő is megjelenik egy igen rövid időre, a törlés idejére. Ha ez a rövid téves impulzus zavaró a kimeneten, akkor csak szinkron törlésű számlálót szabad alkalmazni.

A szinkron törlésű számlálóknál az előírt végértéket kell figyelni. Ennek megjelenése előkészíti a kimeneteket figyelő áramkörön keresztül a törlést, de ez csak a következő órajelnél lesz hatásos.

Az leírt két módszer követhető a 6.35. ábrán. A feladat az, hogy egy négybites bináris számlálóból kell dekadikus számlálót készíteni. A kimeneteket figyelő áramkör egy kombinációs hálózat, amelynek akkor és csak akkor kell a kimenetén a törléshez szükséges 0 szintet előállítani, amikor a figyelt érték megjelenik. Ez aszinkron törlésű számlálónál 1010 (tchát decimális 10), szinkron törlésűnél 1001 (decimális 9).



6.35. ábra. A számlálók ciklusrövidítése

a) Aszinkron törlés

b) Szinkron törlés

A szinkron törlésnél a kombinációs hálózatnak azt kell figyelni, hogy mikor lesz a D és B kimeneten 1 szint:  $F_{10} = D \cdot B$ . Szinkron törlésnél a D és a A kimeneteket kell figyelni:  $F_9 = D \cdot A$ . Ezek a függvények ÉS kapuval megvalósíthatók, mivel azonban a törlés negált szinten történik, NAND kaput használunk.

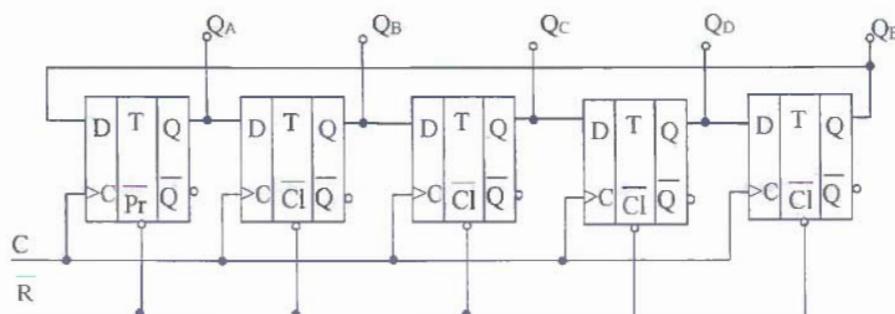
A számlálókat sok esetben **frekvenciaosztóként** használjuk. Erre az ad lehetőséget, hogy a számlálók egy-egy kimenetén az órajel leosztott értéke jelenik meg, ahogyan azt a 6.22. ábrán már láttuk. A frekvenciaosztók bemeneti jele tehát az órajel, kimenete pedig az a számlálókimenet, amelyen a szükséges frekvenciájú jel megjelenik. Pl. ha egy  $f = 8$  kHz frekvenciájú négyzetösszegjelből kell 125 Hz frekvenciájuk létrehozni, akkor egy  $8000/125 = 64$  osztásviszonyú osztóra van szükség. Erre a céllra felhasználhatjuk a 6.34. ábra bármelyik kaszkádósított számlálóját úgy, hogy az órajelbemenetre vezetjük a 8 kHz frekvenciájú négyzetösszegjelet és kimenetként a  $2^5$  helyi értékű számlálókimenetet használjuk. Ezen éppen 125 Hz-es négyzetösszegjel-sorozat jelenik meg.

Amennyiben a kettő hatványai szerinti osztásviszony nem megfelelő, akkor ciklusrövidítést alkalmazunk. A rövidített ciklusú számláló kimeneteket figyelő áramkörének kimeneti jele felhasználható osztott kimenetként. Pl. a 6.35. ábra áramköre 10-es osztóként használható.

## 6.6. Gyűrűs számlálók

A gyűrűs számlálók léptetőregiszterek visszacsatolásával kialakított számlálóáramkörök.

A **moduló-N számláló** a legegyszerűbb felépítésű gyűrűs számláló. Az N mindenkor egymástól megkülönböztethető kimeneti állapotainak számát, tehát a számláló modulusát jelenti. A 6.36. ábra példaként egy moduló-5 számlálót mutat.



6.36. ábra. Moduló-5 számláló

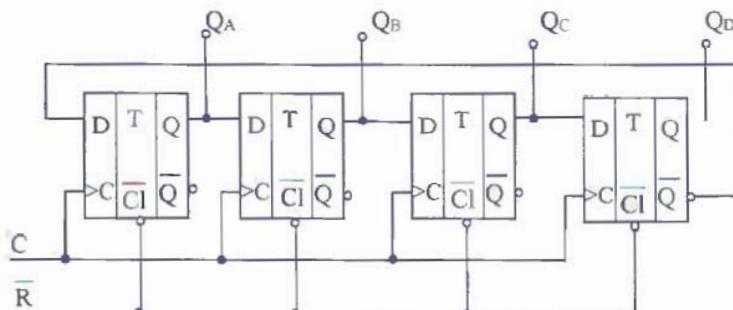
A számláló alapállapota az 10000 állapot, amit a törlő bemenettel lehet beállítani. Az órajelek hatására az első tárolóban lévő 1 végiglép a tárolón és a visszacsatolás miatt a működés ciklikusan ismétlődik. Az áramkör tchát a kimenetein az N-ből egy kódot állítja elő.

Ez a működés azonban megfelel egy tetszőleges számrendszerben való számlálásnak is. Az ábra moduló-5 számlálójának kimeneteit 0-tól 4-ig sorszámozva azt mondhatjuk, hogy a számláló ötös számrendszerben számol, minden léptetésnél ki-jelölve a lépésszámnak megfelelő sorszámú kimenetet.

A tárolók számának változtatásával egyszerűen készíthetünk bármilyen számrendszerbeli számlálót anélkül, hogy dekódolót kellene alkalmaznunk a kimeneten. (A számláló áramkörökkel is lehet bármilyen számrendszerben számláló áramkört készíteni, de a bináris kimeneten ilyenkor egy dekódolót kell alkalmazni).

A moduló-N számláló egy gyors számláló, mert szinkron működésű és mert nem szükséges dekódolót alkalmazni. Hátránya azonban, hogy a modulus növekedésével növekszik a tárolók száma is.

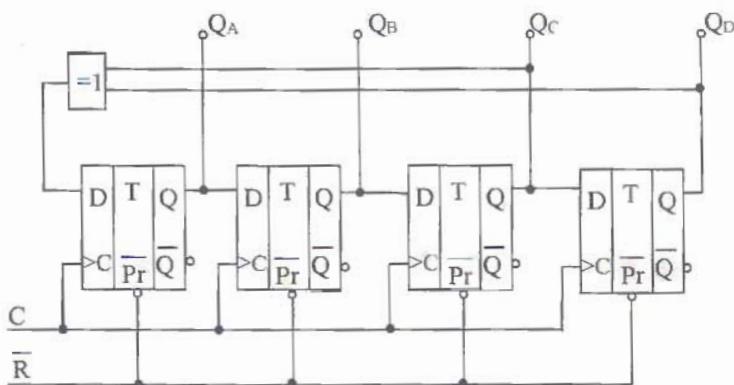
**A Johnson-számláló** egy léptetőregiszter négált soros kimenetének soros bemenet-re való visszacsatolásával hozható létre, amint azt a **6.37.** ábra mutatja.



6.37. ábra. Négy bites Johnson-számláló

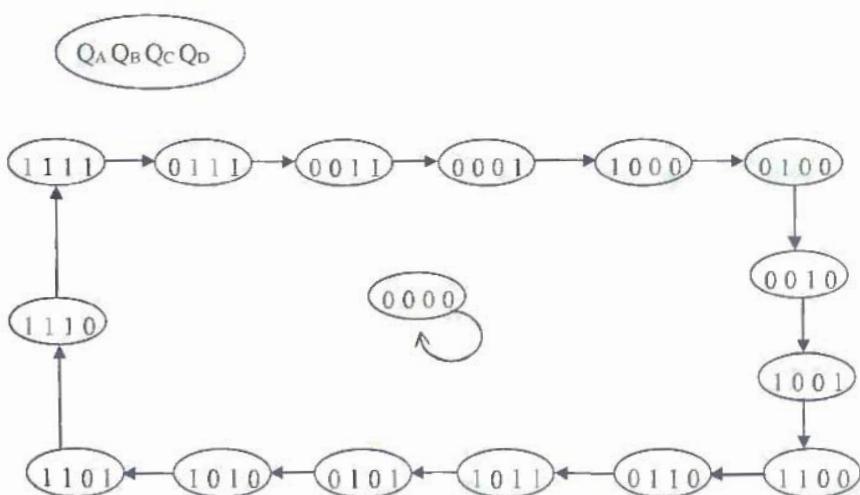
Törlés után az órajelek hatására a számláló a Johnson-kódot állítja elő a kimenetein: a ciklus első részében az A tárolótól indulva feltölti a tárolókat egyesekkel, majd a ciklus második részében nullákkal.

A **maximális ciklusú számláló** antivalencia kapuval visszacsatolt léptetőregiszter. A **6.38.** ábrán látható négy bites számlálónál visszacsatolás a léptetőregiszter C és D kimeneteiről történik.



6.38. ábra. Maximális ciklusú számláló

A számláló elnevezése abból adódik, hogy a 16 lehetséges állapotból az órajelek hatására a kimeneten 15 állapot megjelenik. A ciklusból hiányzó állapot a 0000 állapot, mert ilyen kimeneti állapotok csetén az antivalencia kapu 0-át csatol vissza a bemenetre, tehát nem változik a léptetés utáni állapot. A számláló alapállapota ezért nem lehet ez az állapot. Az ábrán az 1111 kiindulási állapot beállítása lehetséges a törlőbemenetben. A hálózat működésének analizálásával a 6.39. ábrán látható ciklus adódik.



6.39. ábra. A maximális ciklusú számláló állapotdiagramja

A ciklusban az egymás utáni kódszavak látszólag rendszertelenül követik egymást. Ez lehetőséget ad arra, hogy a számlálási funkció kívül az áramkört – főleg nagyobb bitszám esetén – álvéletlen generátorként használjuk.

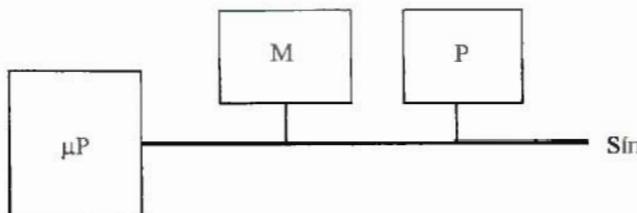
### **Ellenőrző kérdések**

1. Rajzoljuk fel a multiplexer és a demultiplexer belső felépítését!
2. Hogyan történik a dekódolók tervezése?
3. Mi a félösszeadó és a teljes összeadó közötti különbség?
4. Írjuk fel a teljes összeadó igazságtablázatát!
5. Hogyan lehet bináris összeadóból decimális összeadót készíteni?
6. Csoportosítsuk a regisztereket és ismertessünk minden csoportból egy jellegzetes típust!
7. Csoportosítsuk a számlálókat és ismertessünk minden csoportból egy jellegzetes típust!
8. Hogyan módosítható a számlálási ciklus?
9. Ismertessük a gyűrűs számlálók típusait!

## 7. MIKROPROCESSZOROK

A mikroprocesszorok VLSI (*Very Large Scale Integration*: nagyon nagy bonyolultságú) integrálási technológiával készült digitális integrált áramkörök. Az alkalmazás céljától függően készülnek általános célú mikroprocesszorok (szokásos rövidítéssel:  $\mu P$ ) és célprocesszorok. Az általános célú mikroprocesszorokat általában a számítógépek központi vezérlőegységeként használják. A célprocesszorok egy adott feladat önálló megoldásra alkalmas vezérlők, pl. billentyűzetillesztő vagy mikrovezérlő, amely egy teljes mikroprocesszoros rendszert tartalmaz egy integrált áramkörben.

A mikroprocesszorokon belül a 6. fejezetben megismert funkcionális digitális áramköröket találjuk. A regiszterek, számlálók, kapukból felépített logikai áramkörök, multiplexerek, demultiplexerek és összeadók, speciális logikai feladatokat megvalósító rendszertechnikai csoportokat alkotnak.

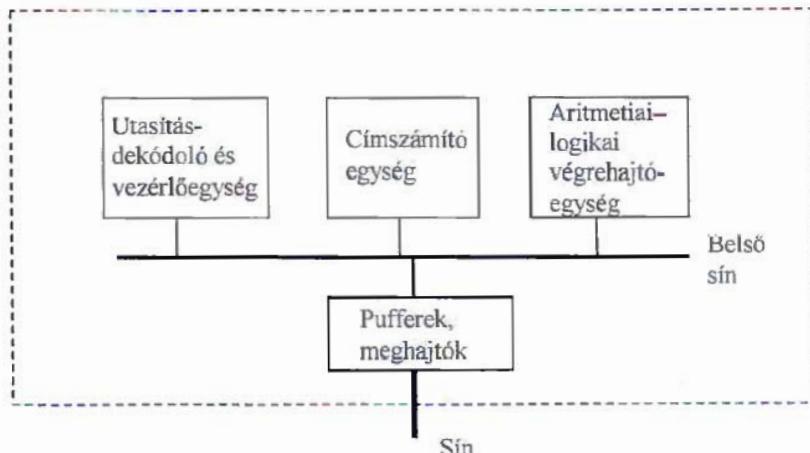


7.1. ábra. Mikroprocesszoros rendszer

A processzor felépítése és működése csak áramköri környezetében ismeretében vizsgálható. A 7.1. ábrán egy mikroprocesszoros rendszer látható. Az M memória sorszámozott, vagyis címezhető rekeszekben tárolja a programok utasításait és az adatokat, bináris számok (bitminta) formájában. A P perifériák a rendszer és a felhasználók közötti kapcsolatot teremtik meg, pl. monitor, billentyűzet stb. Az egységek kapcsolata azonos típusú információt szállító vezetékcsoportokkal valósul meg. Ezeket a vezetékcsoportokat síneknek nevezzük. A sínek vezetékeire több egység is kapcsolódik, ezért a rendszer áramkörei three-state kimenetük. Ebben a rendszerben a központi vezérlőegység feladata:

- a memória- és perifériacímek kialakítása az utasítások alapján,
- a címnek megfelelő egység kimenet-engedélyezése az adatátvitel megvalósítására,
- az utasításban előírt aritmetikai vagy logikai művelet végrehajtása.

A 7.2. ábrán a mikroprocesszor belső felépítése látható. Az egyes rendszertchnikai egységek önálló feladatokat látnak el.



7.2. ábra. A mikroprocesszor belső felépítése

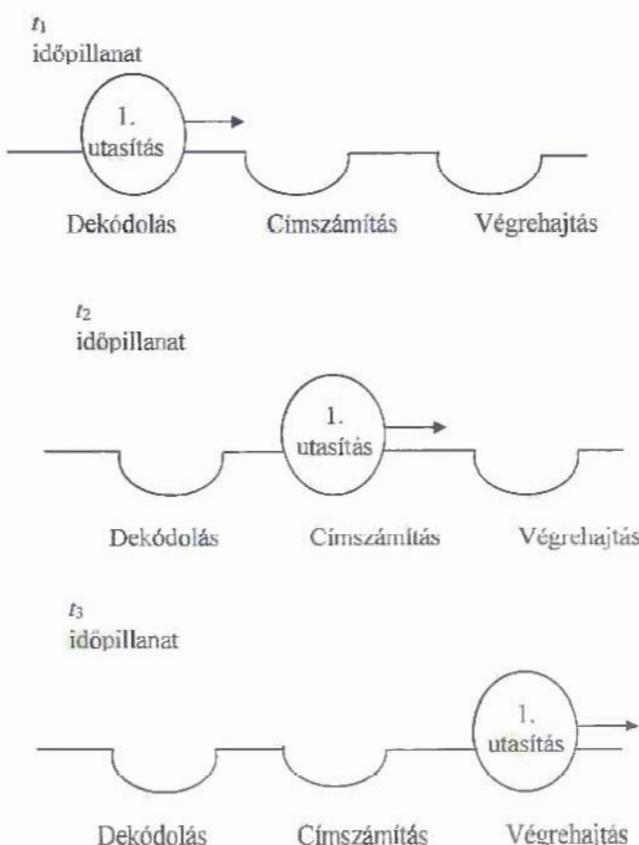
**Az utasításdekódoló és vezérlőegység** a program kezdőcímétől folyamatosan olvassa az utasításokat a memóriából, értelmezi azokat, majd a végrehajtáshoz szükséges áramköröket a megfelelő sorrendben működteti a vezérlőjelek segítségével.

**A címszámító egység** a memória és perifériacímek kialakítását végzi.

**A végrehajtóegység** a bináris adatokon matematikai vagy logikai műveleteket végez. A felsorolt egységek minden mikroprocesszor-típusban megtalálhatók. Az egyes processzortípusok a belső egységek együtteskódésénél szervezésében, a sebesség és a teljesítménynövelését szolgáló rendszertchnikai kialakításban térnek el egymástól. A mikroprocesszorok teljesítménye az elmúlt 10 évben 900%-kal növekedett. Ezt a hihetetlen teljesítménynövekedést egyrészt az órajel frekvenciájának emelése, másrészről a teljesítményt növelő rendszertchnikai változások eredményezték. A processzorok teljesítményét növelő és a jelenleg korszerűnek számító processzorokban alkalmazott rendszertchnikai megoldások a következők:

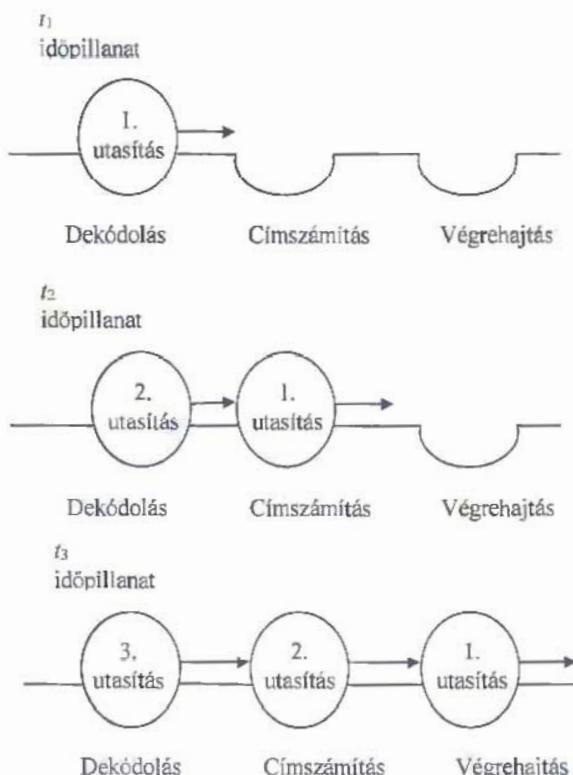
## 7.1. Pipe-line elv

Egy utasítás feldolgozása a 7.2. ábra funkcionális egységeinek időben egymást követő használatát igényli. Ez azt jelenti, hogy az utasítás egy adott feldolgozási stádiumában csupán egy egység működik, a többi nem. Az utasítás pipe-line nélküli feldolgozását mutatja a 7.3. ábra, az utasítás végrehajtásának három különböző fázisában.



7.3. ábra. Utasítás végrehajtás pipe-line nélkül

A pipe-line (csővonalszerű) feldolgozás lényege, hogy minden funkcionális egység, minden időpillanatban működjön. A processzoron belül egyszerre több utasítás található a feldolgozás különböző fázisában. A 7.4. ábra szerinti utasítás-végrehajtásnál a belső funkcionális egységek időben egyszerre, párhuzamosan dolgoznak, ami nyilvánvaló teljesítménynövekedést eredményez. A mikroprocesszorok fejlesztése során a pipe-line állomások számát egyre növelik (szuper pipe-line működésű processzorok).



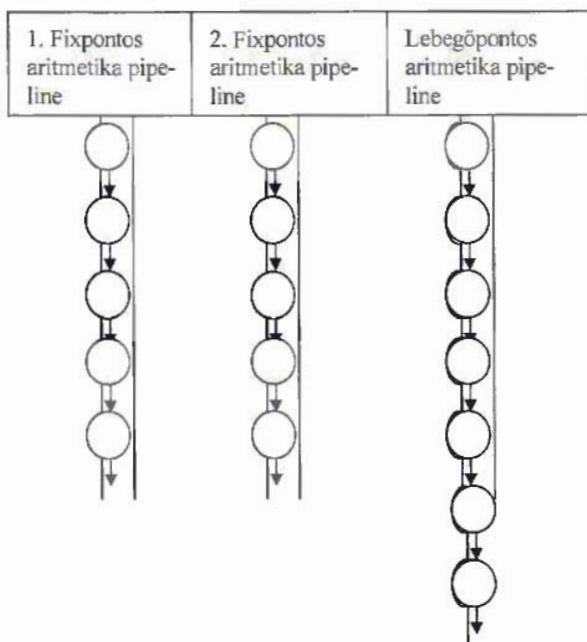
7.4. ábra. Pipe-line utasítás-véghajtás

## 7.2. Lebegőpontos aritmetika, szuper-skalár felépítés

A processzorok által végzett aritmetikai műveletek operandusait a memóriában két-féle formátumban tárolhatjuk. A fixpontos formátumban meghatározott hosszságú, adott számú bitból álló számok tárolhatók. A rendelkezésre álló korlátozott hosszúság meghatározza a szám maximális értékét. Sokkal nagyobb számok tárolhatók lebegőpontos formátumban. Ez a tízes számrendszerben megszokott normál alak (pl.  $1,54 \cdot 10^{23}$ ) kettes számrendszerbeli megfelelője. A lebegőpontos formátum is fixpontos részekből áll, a számértékből és a nagyságrendjét megadó kitevőből. Ahogyan tízes számrendszerben külön végezzük műveletet a számértékkel és külön a kitevővel, a processzorok aritmetikai egysége is külön végez műveletet a lebegőpontos operandus két részével. Ez lassú műveletvégzést eredményez, mert csak több lépésben végezhető el. A sebesség növelés érdekében a korszerű mikroprocesszorok lebegőpon-

tos aritmetikai egységet is tartalmaznak a fixpontos aritmetika mellett. Az ilyen aritmetikai egység igen nagy sebességgel hajtja végre a lebegőpontos műveleteket. A processzoron belül így két feldolgozó egység működik párhuzamosan.

A Pentium osztályú processzoroknál tovább növelték az időben egyszerre működő feldolgozó egységek számát, ezért nevezzük az ilyen mikroprocesszorokat szuper-skalár felépítésűnek. A feldolgozóegységek egymástól függetlenül pipe-line működéssel hajtják végre az utasításokat, így ezek a processzorok szuper-skalár, szuper pipe-line struktúrájúak. A teljesítménynövekedés jól látható a 7.5. ábrán.



7.5. ábra Szuper-skalár, szuper pipe-line felépítés

### 7.3. Cache alkalmazása

A mikroprocesszoros rendszer egyes egységeinek sebessége erősen eltérő. A processzor a leggyorsabb, a memória adatainak elérése azonban hosszú időt igényel. A rendszer teljesítményét tehát visszafogja a lassú memória. A mai processzorokba ezért relatív kisméretű, igen gyors hozzáférésű memóriát integrálnak, a hozzá tartozó vezérlővel együtt. Ezt nevezzük cache memóriának. A cache minden időpillanatban a központi memória egy részének pontos mását tartalmazza. Megfelelő szervezéssel elérhető, hogy a processzor a működési idő nagy részében a gyors cache-ből olvassa az információt, ne a lassú memoriából. Ez a szervezés is sebességnövekedést eredményez.

## 7.4. Virtuális tárkezelés

Az előző teljesítménynövelő módszerek alkalmazása lehetővé tette, hogy rövid idő alatt nagy méretű programok nagy mennyiséggű adatot dolgozzanak fel. A memória mérete azonban korlátozza a programok és az adatok mennyiségett. Ezért a korszerű mikroprocesszorok – megfelelő regiszterek és egyéb áramkörök segítségével – támogatják azokat az operációs rendszereket, melyek memóriacím alapján önállóan, a háttértárolóról a memóriába olvassák a feldolgozáshoz éppen szükséges információt. Ez a virtuális tárkezelés. Eredményeképpen a felhasználónak úgy tűnik, mintha a processzor közvetlenül hozzáférne a háttértárolón lévő összes információhoz. Így a memória mérete már nem korlátozza a programok méretét és az egyszerre feldolgozható adatmennyiséget.

## 7.5. Multitask rendszer

A multitask rendszer azt jelenti, hogy a processzor *egyidőben* több feladatot (taskot) hajt végre. Az előző megoldásoknak köszönhetően elért igen nagy schességnövekedés lehetővé teszi, hogy a processzor ciklikusan egy rövid időtartományban az egyik programot, a következő időtartományban egy másik programot stb. futtasson, majd amikor valamennyi taskból végrehajtott egy olyan kis részt, amely belefert az adott időtartományba, akkor visszatér a legelsőhöz, és ott folytatja, ahol éppen abbahagyta. Ez a folyamat ismétlődik folyamatosan. Az egyes programokhoz tartozó időtartományok olyan rövidek, hogy a felhasználó úgy érzékeli, a processzor egyszerre több programot futtat. Az ilyen multitask működés lebonyolításához szükséges áramköri egységeket tartalmazzák a processzorok, lehetséget adva a felhasználó számára, hogy kihasználja ennek a működési módnak az előnyeit. A processzor multitask működtetése azonban csak megfelelő operációs rendszerrel lehetséges.

A mikroprocesszoros rendszerek fejlődése olyan dinamikus, hogy a felépítésükben és működésükben bekövetkező változások csak a napi szakirodalom tanulmányozásával követhetők nyomon.

### Ellenőrző kérdések

1. Milyen elemekből épül fel a mikroprocesszoros rendszer?
2. Ismertessük a mikroprocesszor feladatát!
3. Milyen fontosabb egységeket tartalmaz a mikroprocesszor?
4. Soroljuk fel a processzorok teljesítményét növclő megoldásokat?