AJTONYI ISTVÁN

DIGITÁLIS RENDSZEREK

MISKOLCI EGYETEM UNIVERSITY OF MISKOLC UNIVERSITÄT MISKOLC МИШКОЛЬЦСКИЙ УНИВЕРСИТЕТ L'U
DE MISKOLC MISKOLCI EGYETEM UNIVERSITY OF MISKOLC UNIVERSITÄT MISKOLC МИШКОЛЬЦСКИЙ УНИВ
T L'UNIVERSITÉ DE MISKOLC MISKOLCI EGYETEM UNIVERSITY OF MISKOLC UNIVERSITÄT MISKOLC МИШКО
РСИТЕТ L'UNIVERSITÉ DE MISKOLC MISKOLCI EGYETEM UNIVERSITY OF MISKOLC UNIVERSITÄT MISKOLC
МИШКОЛЬЦСКИЙ УНИВЕРСИТЕТ L'UNIVERSITÉ DE MISKOLC MISKOLCI EGYETEM UNIVERSITY OF MISKOLC
T MISKOLC МИШКОЛЬЦСКИЙ УНИВЕРСИТЕТ L'UNIVERSITÉ DE MISKOLC MISKOLCI EGYETEM UNIVERSITY O
UNIVERSITÄT MISKOLC МИШКОЛЬЦСКИЙ УНИВЕРСИТЕТ L'UNIVERSITÉ DE MISKOLC MISKOLCI EGYETEM U

MISKOLCI EGYETEM
2002

MISKOLCI EGYETEM
VILLAMOSMÉRNÖKI INTÉZET
AUTOMATIZÁLÁSI TANSZÉK

AJTONYI ISTVÁN

DIGITÁLIS RENDSZEREK

MISKOLCI EGYETEM
2002

Készült a PHARE

támogatásával

Írta:
Dr. AJTONYI István
a műszaki tudomány kandidátusa

Lektorálta:
Dr. CSOPAKI Gyula
a műszaki tudomány kandidátusa

A kiadásért felelős:
a Miskolci Egyetem rektora

Példányszám: 400 db.
Szövegszerkesztés, tördelés: Jancsurák Sándorné

Nyomdai munkák:
EURO TEAM
Design & Press
e-mail: euroteam@chello.hu

Utánnyomás
ISBN 963 661 399 5

ELŐSZÓ

Ezen tankönyv a Miskolci Egyetem műszaki karain mérnöki tanulmányokat folytató hallgatók számára készült. Kiemelten javasolt a villamos ill. informatikus hallgatók részére a jegyzettel azonos című tantárgy oktatásához. Ezen túlmenően ajánlható valamennyi technológus ill. villamos mérnökhallgató részére. Az anyag összeválogatásánál egyrészt a már klasszikusnak számító alapozó ismeretek illetve az erre alapuló naprakész ismeretek összefoglalása volt a cél. A tankönyv *8 fejezetből* áll.

Az *első fejezet* bevezető jellegű rendszerezést nyújt a téma feldolgozásához.

A *második fejezet* a **bináris számrendszerbeni műveletekkel**, illetve a **kódolás, adattovábbítás** alapjaival foglalkozik.

A *harmadik fejezet* a **logikai tervezés módszereit** mutatja be számos példa kapcsán.

A *negyedik fejezet* célja a **digitális áramkörök** ismertetése. E fejezet a tranzisztor kapcsoló üzemétől a TTL, CMOS áramkörök ismertetésén át jut el az I^2L, GaAs áramkörök bemutatásáig.

A **kombinációs hálózatok** alkalmazásával kapcsolatos ismereteket az *ötödik fejezet* tartalmazza.

A **sorrendi hálózatok** ismertetését a *hatodik fejezet* tartalmazza. E fejezet összefoglalja a flip-flopok, számlálók, regiszterek, memóriák ismereteit számos technikai adattal szemléltetve.

A jegyzet legbővebb fejezete a *hetedik fejezet*, amely a **mikroprocesszorok, mikroszámítógépek** legfontosabb ismereteit tartalmazza. Megtalálható a fejezetben a számítógépek alapvető funkcióin, a 8 bites, 16 bites mikroprocesszorok összefoglaló ismertetésén túl a RISC, SPARC processzorok és a PENTIUM funkcionális ismertetése is.

A jegyzet *nyolcadik fejezete* a **programozható logikai elemek** (PLA, FPLA, PAL, GAL, ispLSI, FPGA) átfogó ismertetését és ezen elemekkel való tervezés szempontjait tartalmazza.

A szerző ezúton köszöni Dr. Csopaki Gyula egyetemi docensnek (BME) a lektorálás során tett értékes megjegyzéseit.
Ugyancsak köszönet illeti a tankönyv gépelési, szerkesztési munkálatainak elvégzéséért Jancsurák Sándorné előadót.
A szerző köszönettel vesz minden hasznos, a tankönyvvel kapcsolatos észrevételt.

Miskolc, 1998. november

1. BEVEZETÉS

A berendezésekben a fizikai mennyiségeket jelek reprezentálják. A jelek értelmezési tartománybeli értékeitől függően két nagy csoportba sorolhatók, úgymint **analóg jelek**, illetve **digitális jelek**. Az olyan jeleket, amelyek egy tartományon belül tetszőleges értéket vehetnek fel, s a különböző értékhez különböző információ tartalmat rendelünk, **analóg** jeleknek nevezzük.

Az olyan jelfeldolgozó elem, amelynek bemeneti, illetve kimeneti jele egyaránt analóg (pl. erősítő, csillapító stb.), kimeneti jelének és bemeneti jelének kapcsolatát a bemeneti jel egy meghatározott értéktartományán belül **folytonos függvény** írja (pl. $U_{ki}=KU_{be}$).

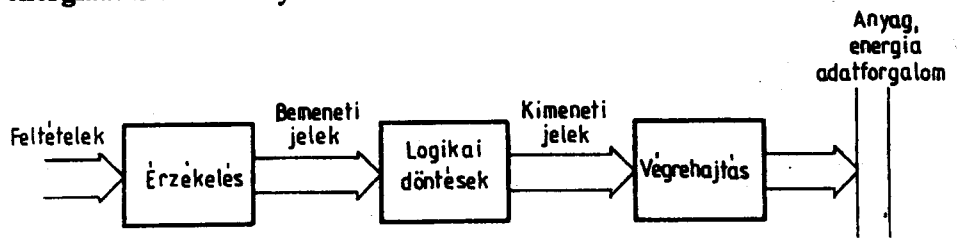
A **digitális jel** csak végesszámú, diszkrét értéket vehet fel és ezen értékek valamely egység (kvantum) egész számú többszörösei. A digitális jelek az információt **számjegyes** formában fejezik ki. Amennyiben a digitális jel értéktartománya két értékre (0, 1) korlátozódik, kétértékű (binér, bináris) jelről beszélünk.

Az olyan jelfeldolgozó egység, amelynek bemeneti és kimeneti jelei egyaránt digitálisak, kimenetének és bemenetének kapcsolatát úgy definiálhatjuk, ha felsoroljuk a bemeneti jelek összes lehetséges értékét és megadjuk ezek mindegyikéhez a kimeneti jelek diszkrét értékét. Ebben a megfogalmazásban a bemeneti jelek a kimeneti jelek keletkezésének feltételét jelentik. A kimeneti jelek tehát úgy tekinthetők, mint a bemeneti jelek **következményei.** Ilyen esetben a kimenetek és a bemenetek közötti kapcsolatot **logikai kapcsolattal** adhatjuk meg.

Digitális jelfeldolgozás esetén az említett **logikai** kapcsolat három tevékenység alapján hozható létre.

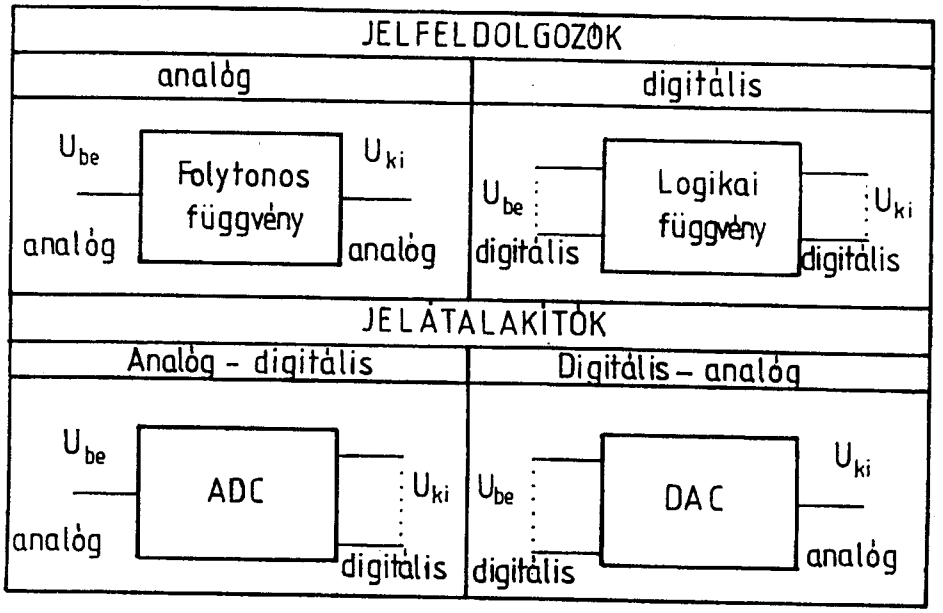
a) A mindenkori **bemeneti feltételek** megismerése,
b) A mindenkori bemeneti feltételekhez a feladat ismeretében a megfelelő kimeneti esemény előidézéséhez szükséges **logikai döntés** meghozatala,
c) A **logikai döntés érvényesítése** a megfelelő kimeneten.

A b, jelű logikai döntést a berendezésekben **logikai hálózat** valósítja meg. Mindhárom tevékenység (a, b, c) együttes megvalósítása esetén **vezérlésről** beszélhetünk. A vezérlésen az irányításnak azon **nyílt hatásláncú ágát** értjük, amelynek révén az **anyag- és adat-feldolgozási**, valamint az **energiaátalakulási** folyamatok automatizálhatók.



1.1. ábra

A vezérlési folyamat hatáslánca az 1.1. ábrán látható. A vezérlő berendezésben az 1.1. ábrán látható funkciókat **szervek** valósítják meg, úgymint érzékelő szerv, logikai döntést végző szerv, beavatkozó/végrehajtó szerv. Az utóbbi két évtizedben a logikai döntést végző szervek, azaz a logikai hálózatok sokkal dinamikusabban fejlődtek, mint a másik két szerv. Rendszertechnikai okból megemlítjük a jelátalakító elemeket, úgymint: **analóg-digitális átalakító** (ADC), **digitális-analóg átalakító** (DAC). A **jelfeldolgozó és jelátalakító** egységek funkcionális sémáját az 1.2. ábrán adtuk meg.



1.2. ábra

1.1. A logikai hálózatok csoportosítása

A logikai hálózatok a megoldandó vezérlési feladattól függően két nagy csoportba sorolhatók, úgymint kombinációs és sorrendi logikai hálózatok. A továbbiakban csak a kétértékű logikai jelekkel működő logikai hálózatokkal foglalkozunk. A **kombációs hálózatok** bemenetén fellépő **ugyanazon jelkombinációkhoz a kimeneten ugyanazon kimeneti jelkombináció** tartozik. Ugyanakkor azonos kimeneti jelkombináció több bemeneti jelkombinációhoz is tartozhat. Itt jegyezzük meg, hogy kombinatorikai szempontból a "kombináció" elnevezés helytelen. A kombinatorikai szempontból helyes elnevezés: variáció. Történelmileg először a kombinációt használták és máig ez maradt fenn. **Sorrendi** logikai hálózatról akkor beszélünk, ha **ugyanazon bemeneti kombinációhoz más-más kimeneti esemény** tartozhat attól függően, hogy ezen bemeneti kombinációk milyen **sorrendben** kerülnek a hálózat bemenetére. A sorrendi hálózatok **belső memóriával** képesek a bemeneti kombinációk sorrendjében a helyes kimeneti eseményt létrehozni.

1.2. Logikai rendszerek

A digitális berendezésekben a bonyolult logikai feladatok realizálását leggyakrabban egyszerű logikai hálózatok rendszerbe foglalásával oldják meg. A logikai hálózatoknak az így kialakuló rendszere alapján bevezethetjük a **logikai rendszer** fogalmát.

A logikai rendszerek az információt hordozó segédenergiát illetően **villamos, pneumatikus, hidraulikus** rendszerek lehetnek. E tárgy keretében főként csak a villamos rendszerekkel foglalkozunk. Az utóbbi években a villamos rendszerek fejlődtek a legdinamikusabban. A villamos rendszerekben alkalmazott kapcsoló elemek alapján megkülönböztetünk **érintkezős és érintkezőmentes** logikai hálózatokat. Az érintkezőmentes hálózatok fejlődése az elektronikai alkatrészektől és azok integráltsági fokától függ. Történelmileg az alábbi érintkezőmentes logikai rendszerek jöttek létre:

- diszkrét félvezetős rendszerek (második generációs rendszerek)
- kis- és közepes mértékben integrált áramkörös rendszerek (SSI-MSI) (harmadik generációs rendszerek)
- nagymértékben integrált áramkörös (LSI) rendszerek (negyedik generációs rendszerek)
- igen nagy mértékben integrált áramkörök (VLSI) rendszerek (ötödik generációs rendszerek).

A bemenetek és kimenetek közötti logikai kapcsolat realizálása alapján a logikai rendszerek két nagy csoportba sorolhatók: **huzalozott logikájú rendszerek** ill. **programozható logikájú rendszerek**. Történelmileg elsőként a huzalozott logikájú rendszerek terjedtek el (érintkezős, diszkrét félvezetős ill. kis és közepes mértékben integrált áramkörök). A huzalozott logikai rendszereket a programozható logikai rendszerek követték. Napjainkban a programozható logikai rendszerek is két jól elkülöníthető csoportba sorolhatók, úgymint **időosztásos elven** működő rendszerek ún. **letöltő programon** alapuló rendszerek. Az időosztásos logikai rendszerek jellemzője, hogy a végrehajtó program végrehajtását egy kristályoszcillátor ütemezi, azaz a program időben állandóan „fut" a rendszeren. Ilyen eszközök például a mikroprocesszorok, számítógépek, programozható vezérlők. Az ún. letöltő programos logikai rendszerek tulajdonképpen a huzalozott logikai struktúrát követik, a letöltő program a chipen belüli összeköttetéseket határozza meg. Ezen eszközöknél a program tehát nem fut, hanem a program funkciója a nagyszámú programozható elem és összekötés konfigurálása. A program letöltése után az áramkör úgy működik, mint a huzalozott logikai rendszer.

2. SZÁMRENDSZEREK ÉS KÓDRENDSZEREK

Ebben a fejezetben a digitális technikában használatos számrendszereket és kódrendszereket ismertetjük.

2.1. Számrendszerek

A következőkben a számok felírását, konvertálását és a bináris számrendszerbeni műveleteket mutatjuk be.

2.1.1. Számok felírása a különböző számrendszerekben

Valamely N szám (numerus) az R alapú (radixú) számrendszerben definíciószerűen

$$N_R = \pm \sum_{k=-h}^{n-1} A_k \cdot R^k \tag{2-1}$$

alakban adható meg. Itt

$$N_{\text{egé sz}} = A_{n-1}R^{n-1} + \ldots + A_1 R + A_0 \tag{2-2}$$

az egész rész, és

$$N_{\text{tört}} = A_{-1}R^{-1} + \ldots A_{-h+1}R^{-h+1} + A_{-h}R^{-h} \tag{2-3}$$

tört rész.

Az N szám az R alapú számrendszerben rövidítve,

$$N_R = A_{n-1}\cdots A_1 A_0 \quad , \quad A_{-1}\cdots A_{-h+1}A_{-h}(R) \tag{2-4}$$

↑

RADIX VESSZŐ

alakban adható meg, ahol az R alsó index a számrendszerre utal.

Pl.: $N = 968,52_{10}$.

A fenti megadás az ún. **additív számrendszerekre** - mint pl. a római számok - nem vonatkozik.

A legfontosabb számrendszereket, számjegyeiket és elnevezésüket a 2.1. táblázatban találjuk.

2.1. táblázat

Megnevezés	Alap (R)	Számjegyek
Bináris (duális, kettes)	$R = 2_{10}$	0,1
Ternális (hármas)	$R = 3_{10}$	0,1,2,
Kvintális (ötös)	$R = 5_{10}$	0,1,2,3,4
Oktális (nyolcas)	$R = 8_{10}$	0,1,2,3,4,5,6,7
Decimális (tizes)	$R = 10_{10}$	0,1,2,3,4,5,6,7,8,9
Duodecimális (12-es)	$R = 12_{10}$	0,1,2,3,4,5,6,7,8,9,a,b
Hexadecimális (16-os	$R = 16_{10}$	0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F

A könnyebb átszámíthatóság kedvéért az említett számrendszerekben kifejezett első 16 számot a 2.2. táblázatban adtuk meg.

2.2. táblázat

Bináris	Ternális	Kvintális	Oktális	Decimális	Duodeci-mális	Hexadeci-mális
0	0	0	0	0	0	0
1	1	1	1	1	1	1
10	2	2	2	2	2	2
11	10	3	3	3	3	3
100	11	4	4	4	4	4
101	12	10	5	5	5	5
110	20	11	6	6	6	6
111	21	12	7	7	7	7
1000	22	13	10	8	8	8
1001	100	14	11	9	9	9
1010	101	20	12	10	a	A
1011	102	21	13	11	b	B
1100	110	22	14	12	10	C
1101	111	23	15	13	11	D
1110	112	24	16	14	12	E
1111	120	30	17	15	13	F
10000	121	31	20	16	14	10

A táblázatból kiolvasható, hogy pl. a 14_{10} a különféle számrendszerekbeli megfelelője:

$$14_{10} = 1110_2 = 112_3 = 24_5 = 16_8 = 12_{12} = E_{16}.$$

Itt jegyezzük meg, hogy a 2-es számrendszerben megadott számot a szám után írt B, a tizes számot D, az oktális számot O ill. Q, a hexadecimális számot H betűvel is szokás jelölni. Eszerint

$$14D = 1110B = 16Q = EH.$$

A digitális technikában a **bináris számrendszer** bír kiemelt jelentőséggel. Valamely N szám bináris számrendszerbeli alakja:

$$N_{(2)} = \pm \sum_{k=-h}^{n-1} A_k \cdot 2^k. \qquad (2\text{-}5)$$

azaz

$$N_2 = \underbrace{A_{n-1} \cdot 2^{n-1} + \ldots + A_1 \cdot 2 + A_0}_{\text{egész rész}} , \underbrace{A_1 \cdot 2^{-1} + \ldots + A_{-h+1} \cdot + 2^{-h+1} + A_{-h} \cdot 2^{-h}}_{\text{tört egész}}. \qquad (2\text{-}6)$$

↑

bináris vessző

Mivel a számok ábrázolása a **helyiértékes** felíráson alapul, az egyszerűsített felírásnál csak az **együtthatókat** adjuk meg.

Pl.: $11011,01_2 = 27,25_{10}$

Ugyanis: $1.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2} = 27,25_{10}$.

A bináris számrendszer egy-egy helyiértéke egy bit információt jelent. Eszerint egy 4 bites bináris szám egy 4 helyiértékes bináris számot jelent. Később ismertetendő célszerűségi okokból a bináris szám **legnagyobb** súlyú bitjének helyét **MSB**-vel (Most Singificant Bit), a **legkisebb** helyiértékű bitjét **LSB**-vel (Least Significant Bit) jelölik:

$$\begin{array}{cccccccc} 2^7 & & & & & & & 2^0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & \\ \uparrow & & & & & & & \uparrow \\ \text{MSB} & & & & & & & \text{LSB.} \end{array}$$

Valamely N szám kettes számrendszerbeli alakját a szám **bináris kódjának is** nevezik.

2.1.2. Számok konvertálása

Konvertálás alatt valamely R_1 alapú számrendszerben megadott szám **átszámítását** értjük egy R_2 alapú számrendszerbe. Mivel a köznapi életben a

leggyakrabban használt számrendszer a 10-es számrendszer, ezért a leggyakoribb feladat a decimális számrendszerbe és viszont történő konvertálás. Egy R≠10 alapú számrendszerből a tizes számrendszerbe történő átszámítás a **definíciós képlet** (2-1) alapján könnyen elvégezhető. A megfordított művelet bonyolultabb.

Az **egész számok** átszámítása **radix-szal** történő **sorozatos osztással** végezhető el és a maradékok adják az új számrendszer jegyeit. A **tört számok** átszámítása radix-szal történő sorozatos **szorzással** kapható meg. A következőkben a fontosabb konvertálási algoritmusokat ismertetjük.

2.1. 2.1. Decimális-bináris konverzió (D → B)

A **decimális-bináris konvertálás** algoritmusa a következő. A kérdéses decimális szám **egész részét elosztjuk kettővel,** majd a kapott eredményt újból és újból elosztjuk kettővel. Az eredményt a szám alá, a maradékot pedig a szám jobb oldalára húzott vonal mellé írjuk. Az osztást addig folytatjuk, amíg 0-t nem kapunk eredményül. A maradékot **alulról fölfelé olvasva** az adott szám egész részének bináris megfelelőjét kapjuk.

A kérdéses **decimális szám tört részének bináris** megfelelőjét **ismételt kétszerezéssel** kapjuk. A tizedesvessző helyén húzott vonal bal oldalára 0-t írunk, ha a kétszerezéssel kapott eredmény 1-nél kisebb; és 1-et, ha nagyobb. Az utóbbi esetben következő kétszerezés csak a szám tört részére vonatkozik. Az eljárást addig folytatjuk, amíg a tizedesvessző jobb oldalán csupa 0-ás decimális jegyet nem kapunk. A leolvasás **felülről lefelé** történik. Pl. . 497, $3515625_{10} = ?_2$.

Egész rész			Tört rész		
497	1	↑	0,	3515625	x2
248	0		0	7031250	x2
124	0		1	4062500	x2
62	0	leolvasási irány	0	8125000	x2
31	1		1	6250000	x2
15	1		1	2500000	x2
7	1		0	5000000	x2
3	1		1	0000000	x2
1	1				
0					

leolvasási irány ↓

Tehát $497, 3515625_{10} = 111110001, 0101101_2$.

Ha a végeredmény **véges tört,** akkor előbb-utóbb (1) 000-át kapunk a szorzás eredményeként, **ha végtelen tört,** akkor addig szorozgatunk 2-vel, amíg elegendő jegyet nem kapunk (véges tizedes tört bináris megfelelője végtelen tört is lehet!).

2.1.2.2. Bináris-decimális konverzió (B → D)

A **bináris-decimális konverzió** legegyszerűbben a definíciós képlet alkalmazásával végezgető el.

Pl.: $11101110,000111_2 = ?_{10}$

Megoldás:

Egész rész: $0.2^0 + 1.2^1 + 1.2^2 + 1.2^3 + 0.2^4 + 1.2^5 + 1.2^6 + 1.2^7 =$

$2 + 4 + 8 + 32 + 64 + 128 = 238_{10}.$

Tört rész: $0.2^{-1} + 0.2^{-2} + 0.2^{-3} + 1.2^{-4} + 1.2^{-5} + 1.2^{-6} =$

$0{,}21875_{10}.$

Tehát: $11101110,0001111_2 = 238{,}21875_{10}.$

2.1.2.3. Bináris-oktális konverzió (B → O)

Algoritmus: a bináris számjegyeket a bináris vesszőtől jobbra és balra 3-as csoportokra osztjuk, s az így kapott **triádok** adják az oktális szám egy-egy helyiértékét.

Pl.: $1101011,01011_2 = ?_8$

Megoldás:

$$\underbrace{1}\ \underbrace{101}\ \underbrace{011}\ ,\ \underbrace{010}\ \underbrace{11}_{2}$$

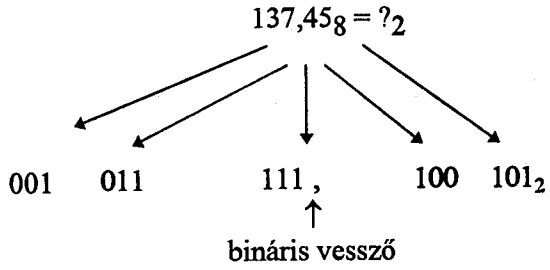
$$\downarrow\ \downarrow\quad \downarrow\quad \downarrow\quad \downarrow\ .$$

$$1\quad 5\quad 3\ ,\ 2_8\quad 3_8$$

2.1.2.4. Oktális-bináris konverzió (O → B)

Az **oktális és bináris** számrendszer radix-ai közötti összefüggés miatt az oktális-bináris konverzió az előző algoritmushoz hasonlóan igen egyszerűen végezhető el. Az oktális szám helyiértékeit az oktális vesszőtől jobbra és balra (egészek és törtek) egy-egy háromjegyű **triáddal** adjuk meg.

Pl.:



2.1.2.5. Decimális-oktális konverzió (D → O)

A decimális-oktális konverzió a 2.1.2.1-ben leírtak analógiájára **sorozatos 8-as osztással** (egész rész) ill. sorozatos **8-as szorzással** (tört rész) végezhető el. Ez a közvetlen módszer.

Példa: $833,372_{10} = ?_8$

	Maradék		
$833_{10} : 8_{10}$	1_8	$8_{10}.0,372_{10} = 2,976_{10}$	2_8
$104_{10} : 8_{10}$	0_8	$8_{10}.0,976_{10} = 7,808_{10}$	7_8
$13_{10} : 8_{10}$	5_8	$8_{10}.0,808_{10} = 6,464_{10}$	6_8
1_{10}	1_8	$8_{10}.0,464_{10} = 3,712_{10}$	3_8

Végeredmény: $833,372_{10} = 1501,2763..._8$

A decimális-oktális konverzió **közvetett módon** a **decimális-bináris** ill. **bináris-oktális konverzióval két lépésben** is elvégezhető.

2.1.2.6. Oktális-decimális konverzió (O → D)

Az oktális-decimális konverzió a definíciós képlet alapján végezhető el:

Pl.: $172,5_8 = ?_{10}$.

Megoldás: $2.8^0 + 7.8^1 + 1.8^2 + 5.8^{-1} = 2 + 56 + 64 + 0,625 = 122,625_{10}$.

2.1.2.7. Bináris-hexadecimális konverzió (B → H)

A bináris és a hexadecimális számrendszer radixai közötti kapcsolat révén ez a konverzió a B → O átalakításhoz hasonlóan végezhető el.

Algoritmus: a bináris szám számjegyeit a bináris vesszőtől indulva jobbra és balra négyes csoportokra **(tetrádokra)** osztjuk, majd ezen tetrádokat **hexadecimális** számmá alakítjuk.

Pl.: $\underbrace{10}\ \underbrace{1101}\ \underbrace{1011}\ ,\ \underbrace{0011}\ \underbrace{11}_{2} = ?_{16}$

Megoldás: 2 D B , 3 C

Tehát: $1011011011,001111_2 = 2DB,3C_{16}$.

2.1.2.8. Hexadecimális-bináris konverzió (H → B)

A H → B konverzió **tetrád képzéssel** végezhető el.

Pl.: $3FA5, CD3_{16} = ?_2$.

Megoldás:

3	F	A	5	,	C	D	3
11	1111	1010	0101	,	1100	1101	0011_2.

2.1.2.9. Oktális-hexadecimális konverzió (O → H)

Az O → H konverzió legegyszerűbben két lépésben végezhető el:
a) O → B konverzió a 2.1.2.4 szerint.
b) B → H konverzió a 2.1.2.7 szerint.

2.1.2.10. Hexadecimális-oktális konverzió (H → O)

A H → O konverzió legegyszerűbben két lépésben végezhető el:
a) H → B konverzió a 2.1.2.8. szerint.
b) B → O konverzió a 2.1.2.3. szerint.

2.1.2.11. Decimális-hexadecimális konverzió (D → H)

A D → H konverzió a 2.1.2.1-ben leírtak analógiájára közvetlen módon 16-tal történő sorozatos osztással (egész rész) ill. szorzással (tört rész) vagy **közvetett módon** két lépésben D → B majd B → H átalakítással végezhető el.

Példa: $152_{10} = ?_{16}$
Megoldás:

$152_{10} : 16_{10}$	8 vagy	a) $152_{10} = ?_2 = \underbrace{1001}\underbrace{1000}_2$	
9 ·			
		b)	98_{16}

Tehát: $152_{10} = 98_{16}$

2.1.2.12. Hexadecimális-decimális konverzió (H → D)

A H → D konverzió a definíciós képlet felhasználásával könnyen elvégezhető.

Példa: $3B_{16} = ?_{10}$
Megoldás: $3.16^1 + B = 48 + 11 = 59_{10}$.

2.1.3. Komplemens számok

Komplemens számon kiegészítő számot értünk. A komplemens képzés általános alapelve (radix komplemens):

$$N + N_K = R^n. \qquad (2\text{-}7)$$

ahol: N a kérdéses számot, N_K a komplemensét, R a számrendszer alapját jelenti, az n pedig egy előre rögzített érték, de minimálisan és külön megjegyzés hiányában a N jegyeinek száma.

A leggyakrabban használt komplemens típusok:

10-es komplemens:	$N_{K10} = 10^n - N$
2-es komplemens:	$N_{K2} = 2^n - N$
9-es komplemens:	$N_{K9} = 10^n - 1 - N$
1-es komplemens:	$N_{K1} = 2^n - 1 - N$

Belátható, hogy a komplemensek száma az előre rögzített n értékétől függ. Példaként adjuk meg a 14_{10} szám 10-es és 9-es komplemensét két legközelebbi hatványra.

		n = 2	n = 3		n = 2	n = 3
N	→	14	014		14	014
N_{K10}	→	86	986	N_{K9} →	85	985
		100	1000		99	999

2.1.2.3.1. Egyes komplemens

A kettes számrendszerbeni komplemensek különleges jelentőséggel bírnak az aritmetikai műveletek végzése miatt.

Az **egyes** komplemens képzése:

a) **Képlettel:**

$$N_{K1} = 2^n - 1 - N. \qquad (2\text{-}8)$$

b) **Algoritmussal:** valamely pozitív bináris szám 1-es komplemensét a bináris helyiértékek 1 → 0 ill. 0 → 1 cseréjével kaphatjuk.

Pl.: $N = 1110_2 = 14_{10}$.

Mivel a szám 4 helyiértékes, így $n_{min} = 4$.

a) $N_{K1} = 2^4 - 1 - N = 15 - 14 = 1_{10} = 0001_2$

$N = 1\ 1\ 1\ 0_2$

b) ↓ ↓ ↓ ↓ komplementálás

$N_{KI} = 0\ 0\ 0\ 1_2$

n = 5 esetén:

a) $N_{K1} = 2^5 - 1 - N = 31 - 14 = 17_{10}$

$N = 0\ 1\ 1\ 1\ 0_2$

b) $\downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow$ komplementálás

$N_{KI} = 1\ 0\ 0\ 0\ 1_2$

2.1.3.2. Kettes komplemens

A **kettes komplemens** az eredeti számot a kettes számrendszerbeli „kerek" számra egészíti ki.

Képzése:

a) **Képlettel:**

$$N_{K1} = 2^n - N. \qquad (2\text{-}9)$$

b) **Soros algoritmussal**: a kérdéses pozitív egész szám (N) legkisebb helyiértéke (LSB) felől, a legnagyobb helyiértéke (MSB) felé haladva a nullákat és az első 1-es jegyet is változatlanul hagyjuk, de az összes többi helyiértéken 0 → 1 ill. 1 → 0 cserét végzünk.

c) **Párhuzamos algoritmussal**: első lépésben előállítjuk a kérdéses szám (N) egyes komplemensét (N_{K1}), majd ehhez hozzáadunk egyet:

$$N_{K2} = N_{K1} + 1. \qquad (2\text{-}10)$$

Pl.: $N = 14_{10} = 1110_2$; $n = 4$.

a) $N_{K2} = 2^4 - 14 = 16 - 14 = 2_{10}$

b) $N = 1110_2$

csere ←

első egyes változatlan

$N_{K2} = 0010_2$

c) $N = 1\ 1\ 1\ 0_2$

$N_{K1} = 0\ 0\ 0\ 1_2$

$+ \quad 1_2$

$N_{K2} = 0\ 0\ 1\ 0_2.$

2.1.4. Bináris számábrázolás digitális berendezésekben

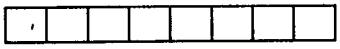
A következőkben a nagyságrend és az előjel ábrázolását mutatjuk be.

2.1.4.1. Nagyságrend ábrázolása

A nagyságrend ábrázolására a **fixpontos** és a **lebegőpontos ábrázolásmód** terjedt el.

2.1.4.1.1. Fixpontos számábrázolás

A **fixpontos ábrázolás** esetén a tizedesvessző (radix vessző) a számot tartalmazó regiszterben előzetes megállapodás szerint rögzített helyen, rendszerint az első értékes jegy előtt van, vagyis minden szám csak 1-nél kisebb lehet (2.1. ábra).

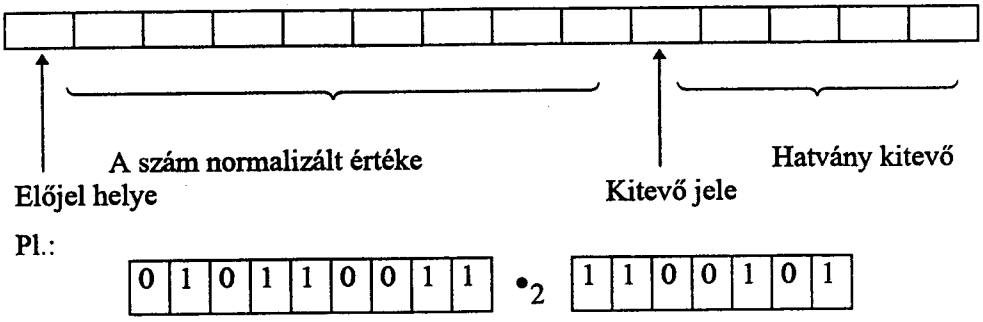


2.1. ábra

Mérőműszerekben, adatfeldolgozó berendezésekben a mértékegységet úgy kell megválasztani (parametrizálni), hogy a mérőszám 1-nél kisebb legyen.

2.1.4.1.2. Lebegőpontos számábrázolás

Lebegőpontos ábrázolásmód esetén a számok **normalizált, féllogaritmus** alakban vannak. A regiszter tartalmazza a szám normalizált értékét, mely a fixpontoshoz hasonlóan, valamilyen 1-nél kisebb szám, ezt követően tartalmazza a szorzó 10 ill. 2 hatványkitevőjének előjel bitjét és a hatványkitevő értékét (2.2. ábra).



2.2. ábra

2.1.4.2. Előjel ábrázolása

Az előjeles számok háromféle szokásos ábrázolási lehetősége:

a) Előjelnagyság (előjel-abszolút érték)
b) 1-es komplemensű
c) 2-es komplemensű.

2.1.4.2.1. „Előjelnagyság" ábrázolás

A radix vessző előtti első helyiérték az **előjel** ábrázolásra szolgál. Az előjelnagyság szerinti ábrázolásban az azonos abszolút értékű, de különböző előjelű számok a képzeletbeli radix vesszőtől jobbra eső részben teljesen megegyeznek, csak az előjelbit különbözik.

Pl.: $+29_{10} = 0{,}11101_2$

$-29_{10} = 1{,}11101_2$

2.1.4.2.2. Előjeles 1-es komplemensű ábrázolás

A **pozitív számokat** az előzőhöz hasonlóan az előjelbittel a „tiszta bináris formában", az egyenes kódban ábrázoljuk. A **negatív számokat egyes komplemensükkel** ábrázoljuk és az előjelbit 1-es. Az azonos abszolút értékű negatív és pozitív számok tehát egymásnak komplemensei, mert valamennyi helyen a 0-ák és 1-ek fel vannak cserélve. 32 16 8 4 2 1

Pl.: $+29_{10} = 0{,}11101_2$

$-29_{10} = 1{,}00010_2$.

2.1.4.2.3. Előjeles 2-es komplemensű ábrázolás

A pozitív számok ábrázolása azonos az előbbi két módszerrel (0 előjel és egyenes bináris kód). A **negatív számok 1-es előjelbittel** és **2-es komplemensükkel** vannak ábrázolva.

Pl.: $+29_{10} = 0{,}11101_2$

$-29_{10} = 1{,}00011_2$.

A 2.3. táblázatban összehasonlításul a +8 ($.2^{-3}$) és -8 ($.2^{-3}$) közötti számokat tüntettük fel háromféle ábrázolásban ill. a ritkán használatos **„offset binary"** (eltolt bináris) kódban.

2.1.5. Összeadás és kivonás komplemens számábrázolással

A bináris számrendszerben a legkisebb helyiértéken végzett összeadás **(félösszeadó)** ill. kivonás **(félkivonó)** műveleti szabályait a 2-4. táblázatban foglaltuk össze. Amennyiben **átvitel** vagy **áthozat** keletkezik, a helyiértékenkénti eredményt a fenti szabályok ismételt alkalmazásával kapjuk. Az i-edik helyiértékre vonatkozó műveleti szabályokat a 2-5. táblázatban

foglaltuk össze. Azokat az egységeket, amelyek ezen műveleteket elvégezni képesek **teljes összeadónak** ill. **teljes kivonónak** nevezzük.

2.3. táblázat

$.2^{-3}$	Előjelnagyság	1-es komplemensű	2-es komplemensű	Offset binary
8				
7	0,111	0,111	0,111	1,111
6	0,110	0,110	0,110	1,110
5	0,101	0,101	0,101	1,101
4	0,100	0,100	0,100	1,100
3	0,011	0,011	0,011	1,011
2	0,010	0,010	0,010	1,010
1	0,001	0,001	0,001	1,001
0	0,000	0,000	0,000	1,000
-1	1,001	1,110	1,111	0,111
-2	1,010	1,101	1,110	0,110
-3	1,011	1,100	1,101	0,101
-4	1,100	1,011	1,100	0,100
-5	1,101	1,010	1,011	0,011
-6	1,110	1,001	1,010	0,010
-7	1,111	1,000	1,001	0,001
-8			1,000	0,000

Az összeadás pozitív számok esetén mindhárom előzőekben leírt számábrázolásnál azonos: a megfelelő helyiértékeket összeadjuk, az előző helyiértékben keletkező átvitel figyelembevételével, a fenti táblázat szerint. Az **összeg nem érheti el az 1-et**, különben a regiszter **túlcsordul. A kivonás ill. negatív számok összevonása** szintén helyiértékenként összeadással történik az illető számábrázolás sajátos szabályainak megfelelően.
A következőkben az 1-es ill. 2-es komplemensű számok áramkörileg legegyszerűbben megvalósítható összevonási algoritmusát ismertetjük. Az áramköri megvalósításra az 5.2.1.2.pontban visszatérünk.

2-4. táblázat

Bemenetek		Félösszeadó		Félkivonó	
		X + Y		X - Y	
X	Y	Összeg S	Átvitel (C)	Különbség D	Áthozat (B)
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	0	1	0
1	1	0	1	0	0

2-5. táblázat

Bemenetek			Teljes összeadó		Teljes kivonó	
			$X_i + Y_i + C_{i-1}$		$X_i - (Y_i+B_{i-1})$	
X	Y	C_{i-1} / B_{i-1}	S_i	C_i (Pi)	D_i	B_i
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1

2.1.5.1. Az 1-es komplemenssel adott számok összevonása

Algoritmus:

1. lépés: Az összegzendő számokat n + 1 helyiértékűre egészítjük ki, ahol n a nagyobbik szám helyiértékeit jelenti.

2. lépés: Az előjelbitet számbitként kezelve elvégezzük az összeadást.

3. lépés: Az MSB biten keletkező **végső átvitelt (körülátvitel**: EAC) hozzáadjuk az összeg legkisebb helyiértékű bitjéhez (LSB).

4. lépés: Ha az eredmény előjele 1, akkor az összeg negatív és 1-es komplemens alakú. Ha az eredmény előjele 0, akkor az összeg pozitív.

Példa:

$+14_{10} : 0{,}01110_2$ $\quad -14_{10} : 1{,}10001_2$

$\underline{+12_{10}} : \underline{0{,}01100_2}$ $\quad \underline{-12_{10}} : \underline{1{,}10011_2}$

$\underline{+26_{10}} : 0{,}11010_2$ $\quad \underline{-26_{10}}$ $1{,}00100_2$

EAC

→ 1

Előjeles eredmény $\quad 1{,}00101_2$

$+14_{10} : 0{,}01110_2$ $\quad -14_{10} : 1{,}10001_2$

$-12_{10} : 1{,}10011_2$ $\quad +12_{10} : 0{,}01100_2$

$+2_{10}$ $0{,}000001_2$ $\quad -2_{10} : 1{,}11101_2$

EAC

$\rightarrow 1_2$

$0{,}00010_2$

2.1.5.2. A kettes komplemenssel adott számok összevonása

Algoritmus:

1. lépés: Az összegzendő számokat **n + 1 helyiértékűre egészítjük** ki, ahol n a nagyobbik szám helyiértékeit jelenti.

2. lépés: Az **előjelbitet számbitként** kezelve elvégezzük az összeadást.

3. lépés: Ha az eredmény előjelbitje 0, akkor ez azt jelenti, hogy az összeg pozitív, ha pedig 1, akkor az eredmény negatív és kettes komplemens alakú.

A 2-es komplemensű aritmetikában a **végső átviteltől eltekintünk.**
Példa:

$+14_{10} : 0{,}01110_2$ $\quad -14_{10} : 1{,}10010_2$

$+12_{10} : 0{,}01100_2$ $\quad -12_{10} : 1{,}10100_2$

$+26_{10} : 0{,}11010_2$ $\quad -26_{10}$ $1{,}00110_2$

$+14_{10} : 0{,}01110_2$ $\quad -14_{10} : 1{,}10010_2$

$-12_{10} : 1{,}10100_2$ $\quad +12_{10} : 0{,}01100_2$

$+2_{10}$ $0{,}00010_2$ $\quad -2_{10} : 1{,}11110_2$

Az 1-es és 2-es komplemensű aritmetika algoritmusát összehasonlítva megállapíthatjuk, hogy a **2-es komplemensű gyorsabb,** mert az eredményt minden esetben **egyetlen összeadással** képezi az EAC-tól függetlenül.

Megjegyzés: a példákban az öt értékes jegyet tartalmazó bináris számok mellől elhagytuk a 2^5 szorzót.

2.2. Kódrendszerek

Ebben a fejezetben a kódolással kapcsolatos fogalmakat, definíciókat ismertetjük.

2.2.1. Kódolási alapfogalmak

Kódnak két szimbólumhalmaz egyértelmű egymáshoz rendelését nevezzük. Az egymáshoz rendelési művelet meghatározott szempontok szerinti végrehajtását **kódolásnak** mondjuk. A kódolás ellentétes művelete - visszatérés a kiinduló halmazra - a **dekódolás.**

A **jelkészlet** azon jeleknek az összessége, melyeket meghatározott szabályok szerint a kódolási művelet során kódszó alkotásra felhasználhatunk. A kétértékü (bináris) kódok **szimbólumkészlete** két elemből áll. Ezek a "0" és az "1". A továbbiakban a **kétértékű** kódokkal foglalkozunk.

A jelkészlet elemeiből meghatározott szabályok szerint képzett értelmes üzenetet jelentő egybefüggő szimbólum sorozatot **kódszónak** nevezzük.

A **kódszó készlet** azoknak a kódszavaknak az összessége, melyek az adott rendszeren belül kódolásra felhasználhatók. Lehetnek olyan kódszavak is, melyeket a meghatározott szabályok szerint képeztünk, de nem tartoznak a kódkészletbe, ezek a **tiltott kódszavak.**

Az információ egysége a **bit. Egységnyi információt hordoz az az üzenet, mely két szimbólumból álló készlettel rendelkezik, és ezen szimbólumok bekövetkezési valószínűsége 0,5-0,5.** Más szavakkal: két azonos valószínűséggel előforduló szimbólum közül az egyik bekövetkezése 1 bit/szimbólum információt jelent. Eszerint egy bináris szám i-edik helyiértékén szereplő érték 1 bit információt jelent.

Ha az információ forrásnak N **egyenlő valószínűséggel** előforduló szimbóluma van, akkor az információ mennyisége:

$$\mathbf{H = ld\ N\ bit/szimbólum.} \qquad (2\text{-}11)$$

Eszerint a decimális számrendszer - melynek 10 egyenlő valószínűséggel előforduló szimbóluma van. - bármely helyiértékén előforduló értéke 3,32 bit információt hordoz.

Redundanciának valamely üzenetforrás **ki nem használt információ mennyiségét** nevezzük:

$$\mathbf{R = H_{max} - H\ bit/szimbólum.} \qquad (2\text{-}12)$$

Gyakran a relatív redundanciát adják meg:

$$\mathbf{R_{rel} = \frac{H_{max} - H}{H_{max}}}. \qquad (2\text{-}13)$$

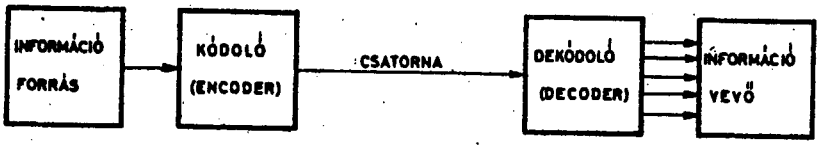
Kódszavak jellemzésére használják a **Hamming-távolság (D)** fogalmát. A Hamming-távolság két kódszó között annyi, ahány kódelemet kell az ellenkezőjére változtatni az egyik kódszóban ahhoz, hogy a másik kódszót kapjuk.

Pl.:	A	=	0010	A	=	11011
	B	=	1100	B	=	0101
	D_{AB}	=	3	D_{AB}	=	4.

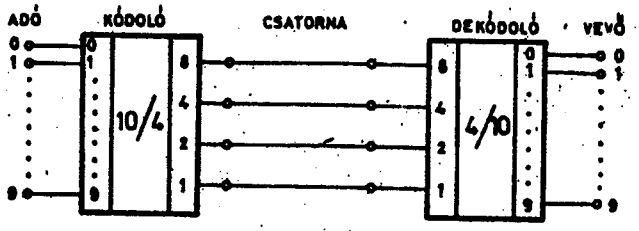
Egy kód Hamming-távolságán a kódszó készletének elemei között észlelt **legkisebb** Hamming-távolságot értjük.

2.2.2. A kódolt információ átvitele

A digitális technikában igen gyakori feladat az, hogy a berendezés egyik részéből a másik részébe, vagy egy távol lévő másik berendezésbe kell információt átvinni. Ez azt jelenti, hogy a kódolt jeleket meghatározott sebességgel egy ADÓ-ból a VEVŐ-be kell juttatni. A digitális információtovábbító rendszer általános sémája a 2.3 ábra szerinti. Eszerint az ADÓ jelét célszerű az átvitel szempontjából előnyösebb kóddá alakítani, (kódolni), majd a VEVŐ-ben visszaalakítani (dekódolni). Az átvitel a **csatornán** történik, amely .lehet villamos kábel, rádiófrekvenciás összeköttetés, fényvezető szál, pneumatikus vagy hidraulikus csőrendszer stb.
Az előzőek megvilágítására rajzoltuk meg a 2.4 ábrát, amelyen a 0...9-ig terjedő decimális számokat bináris csatornán visszük át. **Itt a kódolás célja a kevesebb érpár.** Az átvitel alapvetően kétféle lehet: **párhuzamos** és **soros.**

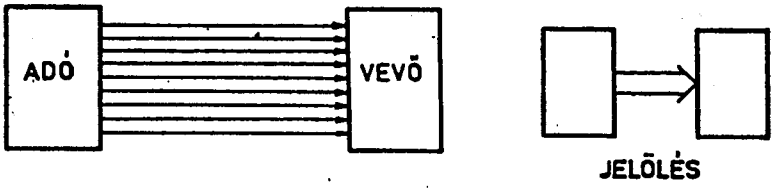


2.3. ábra



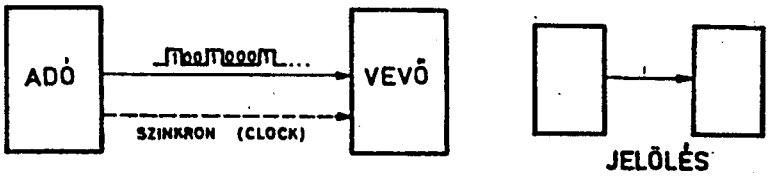
2.4. ábra

Párhuzamos átvitel esetén egy-egy kódszó (karakter) bitjeit **egyidejűleg, külön csatornán** továbbítjuk. Így az összes bit gyakorlatilag **egyidőben érkezik** a vevőhöz. Az egyes kódszavak, karakterek sorban, egymás után kerülnek továbbításra (párhuzamos bit, soros karakter átvitel, 2.5. ábra).

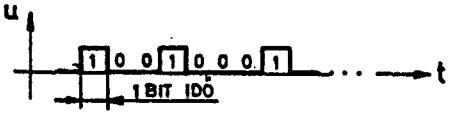


2.5. ábra

Soros (soros bit, soros karakter) átvitelkor (2.6. ábra) az információ bitjeit egyenként, egymás után továbbítjuk **„egy" · vezetéken** (a gyakorlatban egyetlen vezetékpáron, ahol az egyik vezeték rendszerint földelt).
A legegyszerűbb esetben az összekötő vezetéken a binárisan kódolt jel a „0" és „1"-nek megfelelő egymás utáni feszültség vagy áramszintek formájában van jelen (2.7. ábra). A soros átvitelnél fontos kérdés, hogy meddig tart az egyik bit és mikor kezdődik a másik, illetve meddig tart az egyik kódszó és mikor kezdődik a másik. A vevőnek a megfelelő pillanatban kell érzékelnie a 0-át vagy az 1-et akkor is, amikor egymás után több 1-es vagy több 0 van, és a biteket nem választja el egymástól 1 → 0 vagy 0 → 1 átmenet. Szükség van tehát **bitszinkronizációra** és **karakterszinkronizációra**.



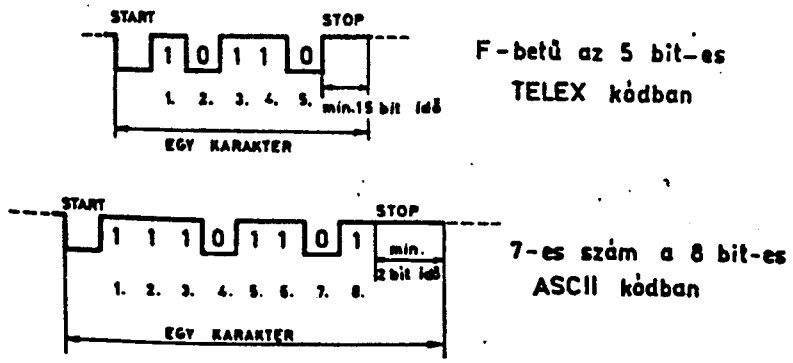
2.6. ábra



2.7. ábra

Ha több karakter blokkba rendezve kerül továbbításra, el kell különíteni a blokkokat egymástól. Ez a **blokkszinkronizáció** feladata. További feladat a teljes üzenet elejének és végének ismerete az **üzenetszinkronizáció** révén. A szinkronizáció legegyszerűbb módja külön **szinkron vezeték** kiépítése az adó és a vevő között. Párhuzamos átvitelnél ez szokásos is, sorosnál csak akkor, ha az adó és a vevő közel van és nem jelent jelentős költséget a szinkron vezeték kiépítése.
Az időzítés szempontjából a soros adatátvitel lehet **aszinkron** és **szinkron.**
Az **aszinkron** átvitelt **START-STOP üzeműnek** is nevezik. Az információ szakaszosan, karakterenként, kódszavanként kerül továbbításra a szavak között szünettel. Az "adás" mindig egy **START jellel kezdődik** és egy **STOP** (szünet) jellel fejeződik be. A **szünet** feltétlenül **hosszabb egy bit** időnél. Így történik

(aszinkron áramjelek) az átvitel TELEX-géppel vagy TELETYPE-pal való kommunikáció esetén (2. 8 ábra).



2. 8 ábra

Egy billentyű leütésekor a gép először egy START impulzust ad ki, amely "0" szintű, majd a hasznos biteket és végül az "1"-es STOP bitet, amely 1,5 ill. 2 bit ideig tart minimum, valójában amíg a következő billentyűt le nem ütjük. A START ill. STOP bitek alapján így a kódszavak jól elkülöníthetők (**karakterszinkronizáció**).

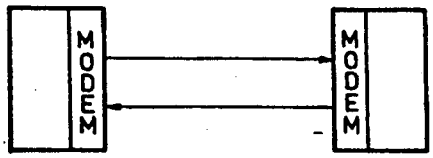
A **bitszinkronizáció** a START jellel indított, az adó bitfrekvenciájával azonos frekvenciájú ÓRA jellel valósítható meg a vevőben. Itt a vevő oszcillátornak pontosságától függ az átvitel hibamentessége. Ugyanis, ha az órafrekvencia elcsúszása az utolsó biteknél elér egy teljes bitidőt, akkor a vett információ hamis lesz. Az elcsúszás veszélye annál nagyobb, minél hosszabb karaktereket küldünk. Távgépíróknál és más elektromechanikus perifériális készülékeknél a **szinkronizálás mechanikus** úton van megvalósítva. Az aszinkron átvitel nagy előnye, hogy a berendezések bizonyos mértékig egyszerűbbek és olcsóbbak lehetnek, mivel az információt nem kell tárolni, azaz **buffer memóriára nincs szükség.** Ugyanakkor hátrányos, hogy az átvivő csatorna nincs kellően **kihasználva** a hosszú szünetek (karakter keresés) miatt.

A **szinkron átvitel a lehető legjobb csatorna kihasználást** teszi lehetővé, mivel ennél általában nem tartanak szüneteket. A kódszavak, karakterek egymás után START és STOP bitek nélkül áramlanak. A karakter sorozat **blokkba** van rendezve. A vevőnek és az adónak igen **precíz szinkronban** kell lennie a blokk egész hosszában. Nemcsak a blokk elejét és végét kell elcsúszásmentesen érzékelni, hanem azt is, hogy hol van a blokk eleje és a blokkban hol kezdődnek és végződnek az egyes kódszavak. A blokk hossza a rendszertől függően a legkülönbözőbb lehet. A **blokk szinkronizálást** a blokk elején elhelyezett a **szinkronkarakterekkel** biztosítják. Egyes rendszereknél az egyes blokkok között szünetet tartanak, az aszinkron átvitelhez hasonlóan (pl. mágneses adatrögzítők). A szinkron átvitelű berendezések általában drágábbak a járulékos **buffer tárolók** miatt. Főleg olyan helyeken alkalmazzák a szinkron adatátvitelt, ahol egyetlen csatornára **időosztásos** üzemmódban **több berendezés**

kapcsolódik, így a csatorna jobban kihasználható. Egy blokk pl. a 2.9. ábra szerinti lehet.

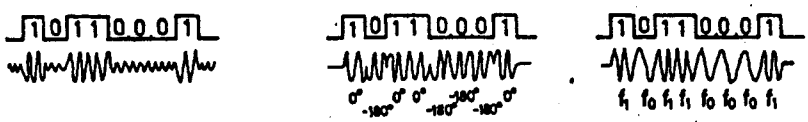
2.9. ábra

Távolsági adatátvitelnél csatornánként gyakran a telefon vonalat vagy rádióösszeköttetést használják. Ilyenkor a csatornára kerülő digitális jelet át kell alakítani, **modulálni** kell. Mivel a csatorna mindkét végén lévő készülék lehet egyszerre adó és vevő is, ezért mindkét oldalon szükséges egy szabványos modulátor és demodulátor: **MODEM** (2.10. ábra).



2.10. ábra

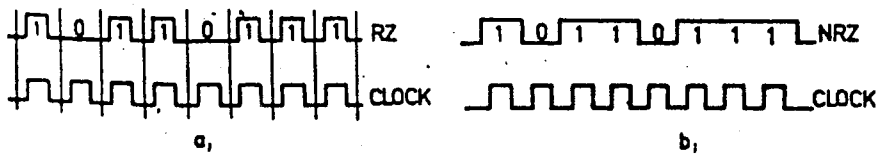
A moduláció fajtáit mutatja a 2.11. ábra.



2.11. ábra

Amplitúdó moduláció (AM): a 0 ill. 1 értéket a nagyfrekvenciás vivő két különböző **amplitúdó értéke** reprezentálja.
Frekvencia moduláció (FM, ill. FSK - Frequency Shift Key): a két bit értékét a vevőhullám **kétféle frekvencia értéke** reprezentálja.
Fázismoduláció: a két bit értékét a vivőhullám **kétféle fázishelyzete** jelenti.
A modulálatlan digitális jel sem mindig egyszerűen 0-1 jelszintekből álló jelfolyam. A következőkben ezekből bemutatunk néhányat.

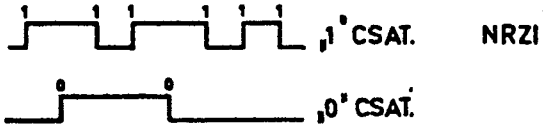
a) **RZ** (Return to Zero = nullára visszatérő): jellegzetessége, hogy minden bit után visszatér 0-ra, azaz ha egymás után több egyes van, akkor ezeket 0-ás szakaszok tagolják 2.12.a) ábra.



2.12. ábra

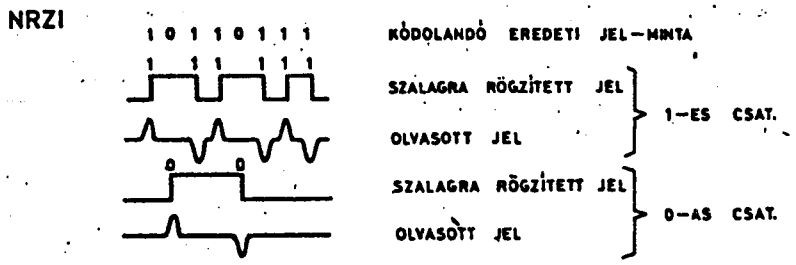
b) **NRZ** (Non Return to Zero = nullára vissza nem térő): a teljes bit ideig a jel bit értéke változatlan, tehát több egymás utáni 1-es nem különül el (2.12.b) ábra).

c) **Kétcsatornás NRZI** (Non Return to Zero Interrupt = nullára visszatérő, megszakadó): érdekessége, hogy a bit értékeket nem a pillanatnyi jelszint határozza meg, hanem a **jelátmenetek.** Az egyik csatorna az 1-eseket hordozza: ahol egyes van a karakterben, ott egy-egy 1 → 0 vagy 0 → 1 átmenet van. A másik csatornán akkor van jel átmenet, ha az illető bit értéke 0. Eszerint egyszerre csak az egyik csatornán van jel átmenet (hiszen egy bit vagy 1 vagy 0), de minden bit helyén van valamelyik csatornán jelátmenet (2.13.a) ábra).



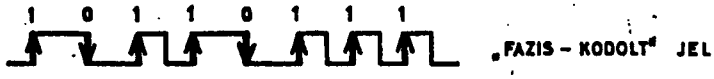
2.13.a) ábra

Az NRZI jelfajta előnye, hogy szinkronizáló jel nem szükséges hozzá, hiszen a bitek helyét az ugrások egyértelműen kijelölik. Ez a jelátvitel különösen a mágneses szalagon történő adatrögzítésnél előnyös, mivel az indukció törvény értelmében csak a jelátmeneteket lehet rögzíteni (2.13.b) ábra).



2.13.b) ábra

d) **Fáziskódolt** (Phase-encoded = PE): ennél is a **jelátmenet,** az ugrás reprezentálja a biteket attól függően, hogy az átmenet milyen irányú 0→ 1 vagy 1 → 0. Ha több azonos értékű bit követi egymást, akkor a jelnek a két bit között fél időben vissza kell ugrania, hogy a következő bit pillanatában ugyanolyan előjelű ugrás legyen (2.14. ábra). Detektáláskor természetesen csak az egyszeres frekvenciának megfelelő időpillanatokban lévő átmeneteket kell figyelembe venni. Mivel az információt itt is **jelátmenetek** reprezentálják, ezért ezt is a **mágneses adatrögzítésnél** használják.



2.14. ábra

2.2.3. Kódtípusok

A hagyományos vezérléstechnikában a számkódok, az intelligens vezérléseknél, adatátvitelnél, az alfanumerikus kódok bírnak jelentőséggel.

2.2.3.1. Számkódok (numerikus kódok)

A következőkben a száminformációk továbbításánál használt kódtípusokat ismertetjük.

2.2.3.1.1. Súlyozott (pozícionális) kódok

Pozícionális (súlyozott) kódról beszélünk, ha a kódszavak helyiértékeihez valamilyen **valós számot (súly)** rendelünk. Ilyenkor a kódszó információtartalmát (N_{10}) rendszerint a kódszó egyeseinek megfelelő súlyok összege adja (additív kód):

$$N_{10} = \sum_{i=1}^{k} W_i S_i . \qquad (2\text{-}14)$$

ahol W_1 az i-edik súlyozás, S_i 0 vagy 1, k pedig a kódszó elemeinek száma. Valamely szám **bináris (duális)** kódja a **szám bináris megfelelője**. A bináris kódhoz hasonlóan az oktális kód a szám oktális, a hexadecimális kód a szám hexadecimális megfelelője.

2.2.3.1.2. BCD kódok

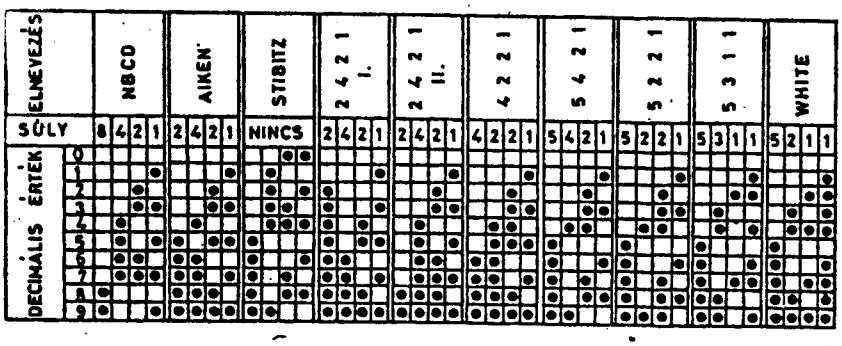
A száminformációk továbbítására leggyakrabban **a binárisan kódolt decimális** kódokat (BCD) használják. Tíz számjegy kettes számrendszerbeni ábrázolásához minimálisan 4 helyiértékre van szükség. Viszont a négy változó lehetséges 16 kombinációjából csak 10-et használunk ki. Megemlítjük, hogy a bináris kombinációk és a decimális számok közötti összes egymáshoz rendelési lehetőség alapján:

$$N = \frac{16!}{(16-10)!} \sim 2{,}9.10^{10} .$$

féle 4 bites BCD kódot lehet lehet készíteni. A gyakoribb BCD kódok táblázatát a 2.15. ábrán láthatjuk. A táblázatban az 1 értéket fekete négyzet jelzi. A decimális számok feldolgozásánál leginkább a **normál BCD kódot (NBCD)** alkalmazzák. **Súlyozása 8421.**
Képzése: a decimális szám egy-egy helyiértékét egy 8421 súlyozású **tetráddal** adjuk meg.

Példaként adjuk meg a 472_{10} szám NBCD kódbeli megfelelőjét.

Tehát: $472_{10} = 0100\ 0111\ 0010_{NBCD}$.



2.15. ábra

NBCD szám átírása a decimális alakba a következőképpen történik: az NBCD kódban adott számot a tizedesponttól jobbra és balra **tetrádokra** (négyes csoportokra) osztjuk, és minden tetrádot a neki megfelelő decimális számjeggyel helyettesítjük. Figyeljük meg, hogy NBCD kódban az **utolsó hat kombináció nem fordul elő.** Megjegyezzük, hogy az NBCD kódot igen gyakran csak BCD-nek mondják.

Az **AIKEN** kód súlyozása a 2421. Képzése (helyiértékenként!):

$$N_{AIKEN} = \begin{cases} N_{bináris}, & \text{ha } N < 5 \\ (N+6)_{bináris}, & \text{ha } N \geq 5 \end{cases}$$

Pl.: 472_{10} = | 0100 | 1101 | 0010_{AIKEN} |

4 7+6 2

Az AIKEN-kód kódszó készlete a 4 bites bináris kód első és utolsó öt kombinációját tartalmazza, a középső hat kombináció nem fordul elő.

Az Aiken kód előnyei:

- az első helyiérték alapján eldönthető, hogy a szám 5-nél kisebb vagy nem,
- a kód helyértékenkénti 0 → 1, 1 → 0 cseréjével a szám 9-es komplemensét kapjuk.

Pl.: $3_{10} = 0011_{AIKEN}$

$\underline{6_{10}} = \underline{1100_{AIKEN}}$

$9_{10} = 1111_{AIKEN}$

Az ilyen kódot, melynek inverze a szám 9-es komplemensét adja **önkomplementáló kódnak** nevezzük. Az önkomplementáló kódok előnye a kivonásnál domborodik ki.

2.2.3.1.3. Excess kódok

Az **excess kódok** közvetetten súlyozott kódok. A szám képzése az

$$N_{10} = \frac{\sum_{i=1}^{k} W_i S_i - e}{q}. \qquad (2\text{-}15)$$

képlet alapján történik. A súlyok (W_i) kettő nem negatív hatványai. A leggyakoribb excess kód **a 3-fölösleg kód.** Szokásos elnevezése **még STIBITZ-kód, 3-többletes kód**. A 2-15 képletből q = 1, e = 3 helyettesítéssel kapjuk. Egy decimális szám STIBITZ-kódbeli megfelelőjét az

$$N_{STIBITZ} = (N + 3) \text{ bináris}$$

képlet alapján képezhetjük **helyiértékenként**:
Visszaalakítás:

$$N_{10} = \frac{\sum_{i=1}^{k} W_i \cdot S_i - 3}{i = 1}. \qquad (2\text{-}16)$$

ahol $W_i = 8,4,2,1$.

Példaként adjuk meg a 472_{10} STIBITZ kódbeli megfelelőjét:

$$472_{10} = \underset{4+3}{0111} \mid \underset{7+3}{1010} \mid \underset{2+3}{0101}_{STIBITZ}.$$

A 3-többletes kód szintén **önkomplementáló.** A STIBITZ-kód önkomplementáló tulajdonságáról győződjünk meg egy példa kapcsán:

$$3_{10} = 0110_{STIBITZ}$$

$$\frac{6_{10}}{9_{10}} = 1001_{STIBITZ}.$$

2.2.3.1.4. Hibafelfedő és javító kódok

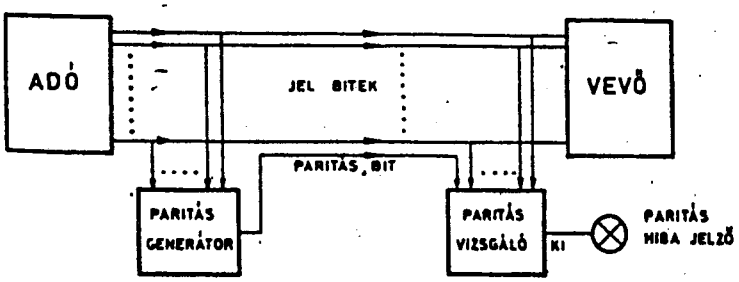
Az eddigiekben az információ továbbítását **„zajmentes" csatorna** feltételezésével végeztük. Zajos csatorna esetén (pl. hibás áramköri működés) a küldött kódszó megváltozhat, azaz kódszóban 0 → 1 vagy 1 → 0 csere jöhet létre. A gyakorlatban annak a legnagyobb a valószínűsége, hogy csak egy helyen változik meg a kódszó. Egyetlen hiba ún. paritásos ellenőrzéssel felfedhető. A **paritásos ellenőrzésnek** alapelve, hogy minden kódszóval még egy ún. **paritásos** ellenőrzésnek alapelve, hogy minden kódszóval még egy ellenőrző helyiértéket (paritás bit) is átviszünk a csatornán. Ennek értékét úgy választjuk meg, hogy a teljes kódszó - beleértve az ellenőrző helyiértéket is -

páros vagy páratlan 0-kat ill. 1-eseket tartalmazzon. Ezt a járulékos helyiértéket **paritás bitnek** szokás nevezni.

Az ilyen párosság ellenőrzéssel ellátott BCD kódok továbbításához párhuzamos átvitelnél **öt vezetékre** van szükség. A relatív redundancia tehát

$$Rrel = \frac{5-1d10}{5} = 0{,}336.$$

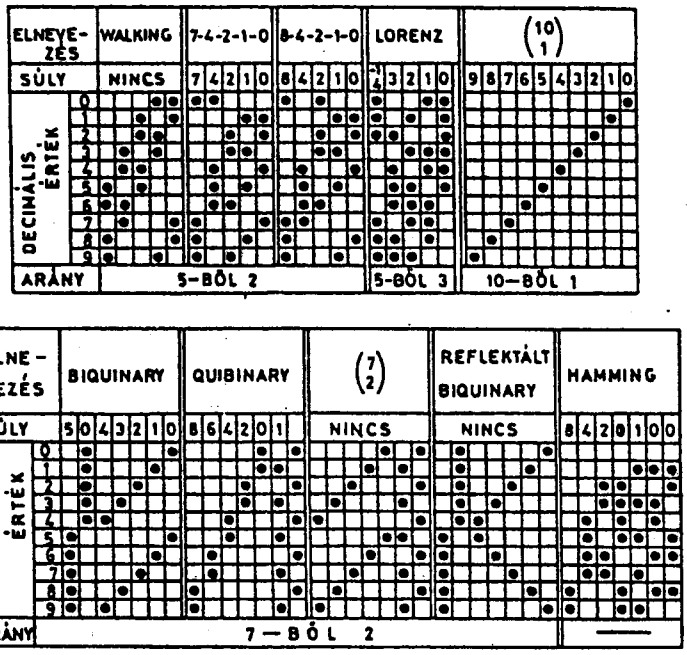
A hibafelfedést tehát a **redundancia növelésével** értük el. TTL rendszerben a paritásbit előállítása, ill. ellenőrzése az SN 74180 típusú bővíthető 8 bites áramkörrel könnyen elvégezhető. A paritásos ellenőrzést igen gyakran alkalmazzák, mert így **tetszőleges kódtípus** felruházható hibafelfedő képességgel. Paritás hibajelzéssel megvalósított **párhuzamos kódátvitel** vázlatát mutatja a 2.16. ábra. A **paritás generátor** figyeli az adó kimeneteit és előállítja a paritás bitet, amivel kiegészíti az átviendő jelet. A vevő oldalon a **paritás vizsgáló** figyeli a kiegészített információt és ha nem megfelelő számú 1-est észlelt, akkor jelzést ad.



2.16. ábra

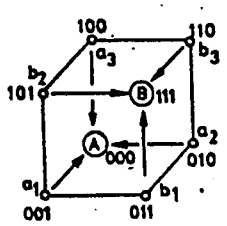
Egyszerűbb hibafelfedő áramkört igényelnek az **aránykódok.** Ezek alapvető jellemzője, hogy az egyes kódszavakban szereplő **egyesek és nullák aránya állandó.** A leggyakrabban használt aránykód a 7-4-2-1-0 súlyozású „5-ből 2" kód. A **biquinary kód** 9-re oly módon komplemens, hogy a bi és a quinér részeket fordítva kell leírni. A gyakoribb aránykódok táblázatait a 2.17. ábrán adtuk meg.

A **hibafelfedés feltétele, hogy a Hamming-távolság egynél nagyobb legyen (D ≥ 2).** Ha ugyanis a két küldött kódszó között a Hamming-távolság 1, akkor egy hiba a másik kódszót is eredményezheti. **Maximálisan D-1 hiba fedhető fel.** Az eddigiekben csak a hiba létének kimutatásával foglalkoztunk. A hiba **javításához a hiba helyét** is ismerni kell. A hiba javításának lehetőségét egy olyan példa kapcsán mutatjuk be, amikor három helyértékű bináris kóddal csak két üzenetet továbbítunk (A = 000, B = 111). A 8 különböző kombinációt ábrázoljuk egy kocka segítségével (2.18. ábra).



2.17. ábra

Az üzenetek közötti Hamming-távolság tehát 3. Az eddigi ismereteink szerint lehetőség volna D - 1 = 2 hiba felfedésére. Ehelyett lehetőség kínálkozik 1 hiba javítására. Ha ugyanis egy hiba keletkezik, akkor az A üzenetből a_i, ill. a B üzenetből b_i lesz, (ahol i = 1,2,3). Amennyiben olyan dekódoló áramkört készítünk, amely az a-kat A-nak, a b-ket B-nek veszi, akkor ez az áramkör 1 hibát ki tud javítani.



X	Y	Z	A	B
0	0	0	1	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	0	1

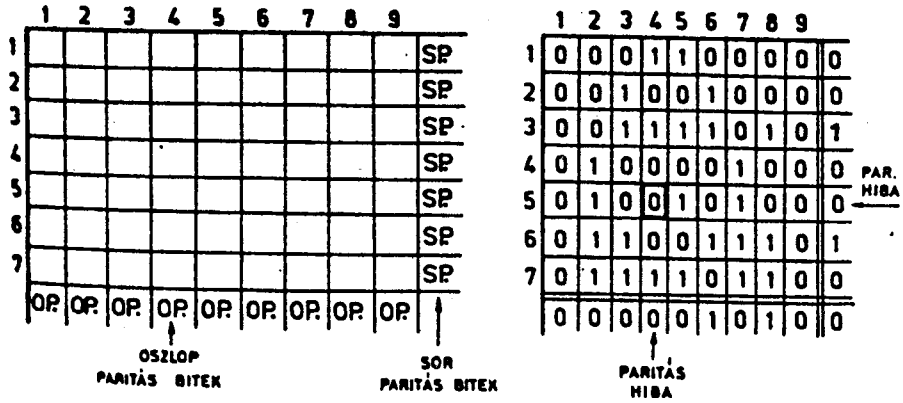
2.18. ábra

Általánosságban **m információs bithez k ellenőrző** értéket a (2-17) szerint kell rendelni, hogy egy hibát ki tudjunk javítani.

$$2^k \geq m + k - 1. \qquad (2\text{-}17)$$

Eszerint a decimális számok (m = 4) egy hiba korrigálására alkalmas továbbításához k = 3 **ellenőrző** helyiérték szükséges. Ilyen tulajdonságú a **Hamming-kód,** melynek kódtáblázata a 2.17. ábrán megtalálható. A Hamming-féle hibajavítás ismertetésére az 5.2.2.4. pontban visszatérünk.

A hibajavítást blokkrendszerű adatátvitel esetén **sor-** és **oszlop paritás** ellenőrzésével is elvégezhetjük. Ha az adó kódszavait szisztematikusan egymás alá írva képzeljük el, akkor egy-egy paritás bitet minden sorhoz és minden oszlophoz rendelhetünk (2.19. ábra). Ily módon egyetlen hiba a hibás sor és oszlop metszéspontjában van. Így tudjuk a hiba helyét, tehát értékcserével (0 → 1 vagy 1 → 0) a hiba javítható.



2.19. ábra

Az újabb rendszertechnikai ismereteket igénylő **egyátmenetű kódokkal** az 5.2.3. pontban foglalkozunk.

2.2.3.2. Alfanumerikus kódok

Alfanumerikus információk átvitelénél leggyakrabban az ASC II, az EBCDIC és a telex kódot használják.

2.2.3.2.1. Az ASC II kód

Ezt a **8 bites** amerikai szabvány kódot használja igen sok perifériás berendezés. A 8. bit **paritásbit,** amely a 7 hasznos karakterben szereplő **1-esek számát párosra** egészíti ki. A maradék 7 bit (2^7 kombináció) szolgál az információ kódolására, a 6. és 7. bit azt határozza meg, hogy a legkisebb 5 bit kisbetű, nagybetű, szám vagy írásjel-e. Figyeljük meg a 2.20. ábrán, hogy a betűkarakter bináris kódja megfelel a szóban forgó betű ABC szerinti „helyezésével". Pl. az „E" betű 5. sz. ABC-ben, így a kódjának utolsó 5 bitje: 00101.

ASC. II. KÓD																
7. 6. bit				5.	4.	3.	2.	1.	7. 6. bit				5.	4.	3.	2.	1.
00	01	10	11						00	01	10	11					
NULL	<I	@	\						DLE	0	P	p	●				
SOK	!	A	a					●	DC1	1	Q	q	●				●
STX	"	B	b				●		DC2	2	R	r	●			●	
ETX	#	C	c				●	●	DC3	3	S	s	●			●	●
EOT	S	D	d			●			DC4	4	T	t	●		●		
ENQ	%	E	e			●		●	NAK	5	U	u	●		●		●
ACK	&	F	f			●	●		SYN	6	V	v	●		●	●	
BELL	'	G	g			●	●	●	ETB	7	W	w	●		●	●	●
BS	(	H	h		●				CAN	8	X	x	●	●			
HT	)	I	i		●			●	EM	9	Y	y	●	●			●
LF	*	J	j		●		●		SVB	:	Z	z	●	●		●	
VT	+	K	k		●		●	●	ESC	;	[	{	●	●		●	●
FF	,	L	l		●	●			FS	<	\	:	●	●	●		
CR	-	M	m		●	●		●	GS	=	]	}	●	●	●		●
SO	.	N	n		●	●	●		RS	>	^	~	●	●	●	●	
SI	/	O	o		●	●	●	●	US	?	—	DEL	●	●	●	●	●

2.20. ábra

ASCII kód szerint történik a kommunikáció a központi egysége és a klaviatúra illetve a display között soros átvitel formájában.

2.2.3.2.2. A telex kód

A telexkód 5 bites alfanumerikus kód. Mivel az 5 sáv csak $2^5 = 32$ variációs lehetőséget nyújt, ezért külön kell gondoskodni a betűk és a többi jel kódolásáról. Az öt lyuk ill. annak megfelelő áramimpulzus azt jelenti, hogy utána betűk következnek (B ill. A....Z), míg 4 lyuk azt jelenti, hogy utána egyéb jelek (E, ill. -....+) következnek. Tehát az 11111 kódú betűváltó után érkező valamennyi kódszó betűt jelent egészen addig, amíg az 11011 kódú számváltó meg nem érkezik. Az ezután érkező kódszavak számot vagy írásjelet jelentenek. 6 kódszó a telexgép vezérlésére szolgál.

Felhasznált irodalom

Ajtonyi I: Vezérléstechnika I. J 14-1417 Tankönyvkiadó, Budapest, 1987.
Ajtonyi I: Vezérléstechnika II. J 14-1418 Tankönyvkiadó, Budapest, 1987.
Gál T: Digitális rendszerek I. BME jegyzet 1995.
Jancvics - Tóth: A logikai tervezés módszerei Műszaki Könyvkiadó 1971.

3. A LOGIKAI TERVEZÉS ALAPJAI

A fejezet célja: a logikai hálózatok analízisénél és szintézisénél használatos tervezési módszerek bemutatása és begyakorlása.

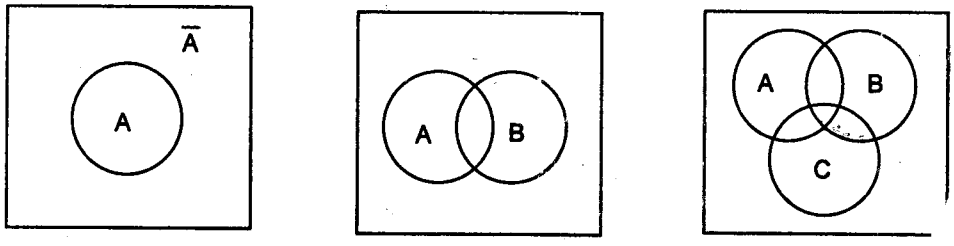
3.1. Logikai változók

A spekulatív úton kialakult formális logika törvényszerűségeinek matematikai leírása **George Boole** és **Augustus De Morgan** nevéhez fűződik. A matematikai logikának a kapcsolóelemek leírására használható részét **Boole algebrának (kapcsolás algebra, logikai algebra)** szokás nevezni. A Boole algebra változóit **kétértékű** eseményeknek tekintjük, s ezek lehetnek **igaz**, vagy **hamis** értékűek attól függően, hogy bekövetkeznek-e, vagy sem. Ezen eseményeket az ABC betűivel jelöljük és **logikai változóknak (Boole változók)** nevezzük. A számításoknál a változók valóságos eseménytartalmától eltekintünk, s csak a puszta matematikai formát vizsgáljuk. Ezért a betűk által szimbolizált eseményeknek igazságértéket tulajdonítunk. A szóban forgó esemény szempontjából logikailag két eset lehetséges: vagy bekövetkezik, vagy nem. Ha bekövetkezik, akkor logikai értéke **igaz**, amit röviden 1 jellel jelképezünk, ha pedig nem következik be, akkor a logikai értéke **hamis**, amit a 0 szimbolizál. Ily módon egy kapcsoló eszköz, digitális érzékelő, jelzőlámpa stb. be- ill. kikapcsolt állapotához hozzárendelhetjük az 1, illetve 0 szimbólumot pl. az alábbiak szerint: **bekapcsolt állapot: 1, kikapcsolt állapot: 0.** Hangsúlyozni kell, hogy az 1, illetve 0 a logikában nem számjegyek, csak célszerű **szimbólumok**, melyekhez az állítás igaz, vagy hamis értéke tartozik.

3.1.1. Logikai változók szemléltetése

A logikai változók szemléltetésére az alábbi módszerek használatosak: Venn diagram, Veitch diagram, idődiagram, KV tábla, folyamatábra.

a) A **Venn diagramok** a logikai változókhoz egy-egy **síkba leképzett ponthalmazt** rendelnek. A ponthalmazok a diagramon tetszőleges síkidommal határolt területeket képeznek. A 3.1.a) ábrán az idom kerületén belüli terület felel meg a változó A = 1, míg a kívüleső terület pedig a változó A = 0 értékének.

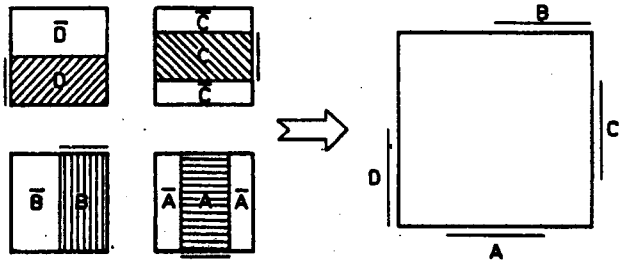


a) *b)* *c)*

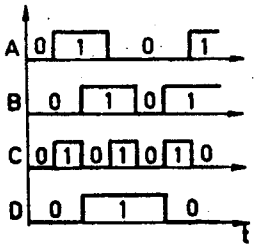
3.1. ábra

Az ábrán a két, illetve három változós Venn diagramot is megrajzoltuk. A Venn diagramok legfeljebb **három változóig** használatosak.

b) **A Veitch diagramok** olyan módosított Venn diagramok, amelyek a változó **igenleges**, illetve **nemleges** értékét - minthogy azok bekövetkezési valószínűsége 50-50 % - egyenlő területrésszel ábrázolják (3.2. ábra). A Veitch diagram peremén vonallal és betűvel jelöljük, hogy a változó **igenleges (ponált)**, illetve **nemleges (negált)** értéke mely területrészen található. **Síkbeli Veitch diagramon maximum 4, térbelin maximum 6 változó** ábrázolható egyidejűleg szemléletesen. A Veitch diagramok a logikai változók szemléltetésén túl a logikai kapcsolatok meghatározására is használatosak.



3.2. ábra



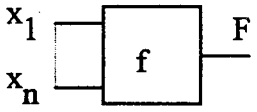
3.3. ábra

c) A logikai események **idődiagramon** is jól szemléltethetők. Ennél az ábrázolási módnál a kétértékű eseményeket az **idő függvényében** ábrázoljuk, így az események időbeli lefolyását is követhetjük. A 3.3. ábra szerinti idődiagram 4 változót szemléltet. Idődiagramon tetszőleges számú változó ábrázolható egyidejűleg.

3.2. Logikai függvények

A **logikai (kapcsolási) függvény** a kimeneti és bemeneti események közti **logikai kapcsolatot** adja meg. A logikai függvények sajátsága, hogy mind a függő (kimeneti), mind a független (bemeneti) változók csak két értéket vehetnek fel: 0-át, illetve 1-et.

A logikai függvények általános jelölése a 3.4. ábra alapján:



3.4. ábra

$$F = f(A, B, C, \ldots N). \quad (3\text{-}1)$$

$$F = f(X_1, X_2 \ldots Xn). \quad (3\text{-}2)$$

ahol A, B, C, ... N, illetve $X_1, X_2, ..., X_n$ **a független Boole változók, f pedig a függvénykapcsolatot** (funkció) jelöli.

Mivel mind a bemeneti, mind a kimeneti változók kétértékűek, ezért a független változók számától (n) függően előre meghatározható a képezhető kapcsolási függvények száma (K):

$$K = 2^{2^n}. \qquad (3\text{-}3)$$

A lehetséges függvények számát a változószám függvényében a 3.1. táblázat mutatja.

3.1. táblázat

n	$K = 2^{2^n}$
1	4
2	16
3	256
4	65 536
5	4967 296

Itt jegyezzük meg, hogy a kimeneti változókat szokás még Y_i, illetve Z_i betűkkel, az időtől függő változókat Q_i-vel jelölni. A jelölésnél gyakran a **fizikai modellhez alkalmazkodó betűket** használjuk a be- és kimenetek megkülönböztetésére.

3.2.1. Egyváltozós logikai függvények

Ha a kimeneti esemény egyetlen bemeneti változótól függ, **egyváltozós logikai függvényről** beszélünk. A 3.5. ábrán feltüntettük az **A** bemeneti (független) változó értékétől függően az F függő változó milyen értéket vehet fel. A képezhető függvények száma 4.

A 3.5. ábrán látható táblázatot, mely a független változók összes lehetséges kombinációit és a függvény által meghatározott kimeneti eseményt feltünteti **értéktáblázatnak (kombinációs, vagy igazságtáblázatnak)** nevezzük.

A logikai függvény jelölésében a felső index a bemeneti változók számát, az alsó pedig a **függvény sorszámát adja meg**. Ezt a **decimális sorszámot a függvényértékekből alkotott bináris számból** képezzük. Az egyváltozós függvények közül az F_1^1 és az F_2^1 bír kiemelt jelentőséggel.

	A	$F_0^1(A) = 0$	$F_1^1(A) = \overline{A}$	$F_2^1(A) = A$	$F_3^1(A) = 1$
2^0	0	0	1	0	1
2^1	1	0	0	1	1
Elnevezés		Konstans "0"	Tagadás	Jelmásolás	Konstans "1"

3.5. ábra

3.2.1.1. A NEM kapcsolat: $F = \overline{A}$

A NEM kapcsolat táblázata

A	$F_1^1 = \overline{A}$
0	1
1	0

A továbbiakban a változó **tagadott (negált)** értékét a betűjele fölé húzott vízszintessel jelöljük (olv. A negált). A NEM kapcsolat szokásos elnevezései: **tagadás, jelfordítás, negáció, inverzió**. A tagadást fizikailag realizáló logikai áramkört **invertern**ek nevezik. Az inverter a bemenetén fellépő jelváltást (**1 ⟶ 0** illetve **0 ⟶ 1**) **időkéséssel** valósítja meg, így a fenti igazságtáblázat csak állandósult állapotban igaz. Szimbolikus jelöléseknél a tagadást **kis kör** jelzi. Az SN 7404 típusú integrált áramkör 6 db invertert tartalmaz. **Az inverter kimenő jele a bemenő jellel ellentétes logikai értékű.**

Relés hálózatoknál az alábbi hozzárendelést szokás alkalmazni:

relé tekercs	gerjesztve:	1
	gerjesztetlen:	0
relé érintkező	zárt (rövidzár):	1
	nyitott (szakadás):	0

A fenti hozzárendelés szerint a tagadás műveletét érintkezős hálózatokban **bontó** érintkezővel realizálhatjuk.

A tagadás szabályai: $\overline{1} = 0$ $\quad \overline{\overline{A}} = A$ (3-4)

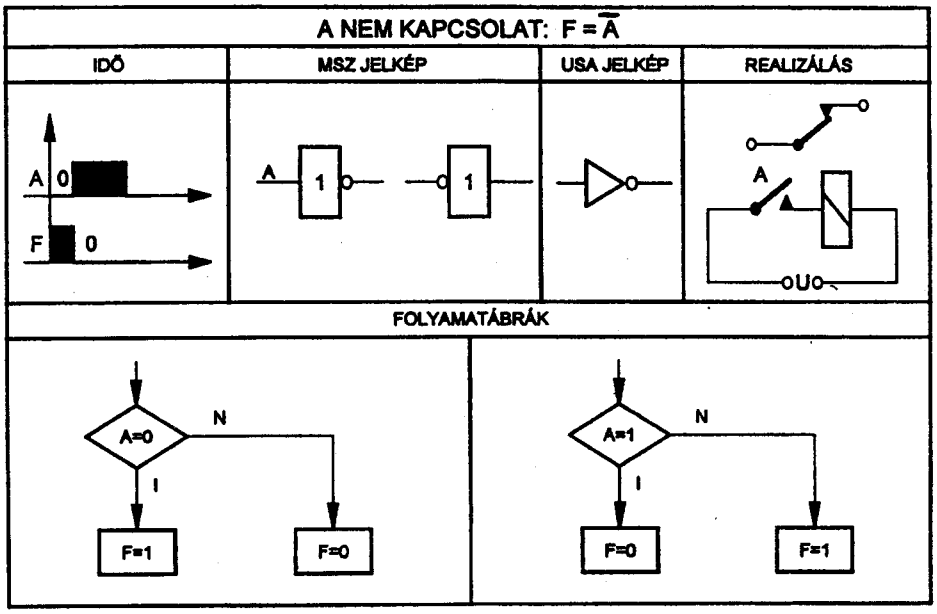
$\overline{0} = 1$ $\quad \overline{\overline{A}} = \overline{A}$.

Szavakban : **páros számú tagadás állítást, páratlan számú tagadás tagadást jelent.**

A NEM kapcsolat idődiagramját, érintkezős realizációját, MSZ jelképét a 3.6. ábrán találjuk. Egy vezérlőberendezésen belül az invertert a logikai funkción túl **időzítési, jelgenerálási, vagy jelregenerálási** feladatok megoldására is használhatjuk.

A NEM művelet bájtokra is kiterjeszthető. Ez esetben az invertálást a bájt (vagy szó) valamennyi bitjére el kell végezni. Például a CMA utasítás az akkumulátor tartalmának komplementálását végzi.

Legyen az A=2B hex, így a CMA művelet elvégzése után az A tartalma D4 hex lesz, ami egyben az eredeti tartalom **egyes komplemense.**



3.6. ábra

3.2.1.2. A jelmásolás (jelkövetés)

A **jelmásoló** a bemenő jel logikai értékét jeleníti meg a kimeneten:

A	$F_2^1(A) = A$
0	0
1	1

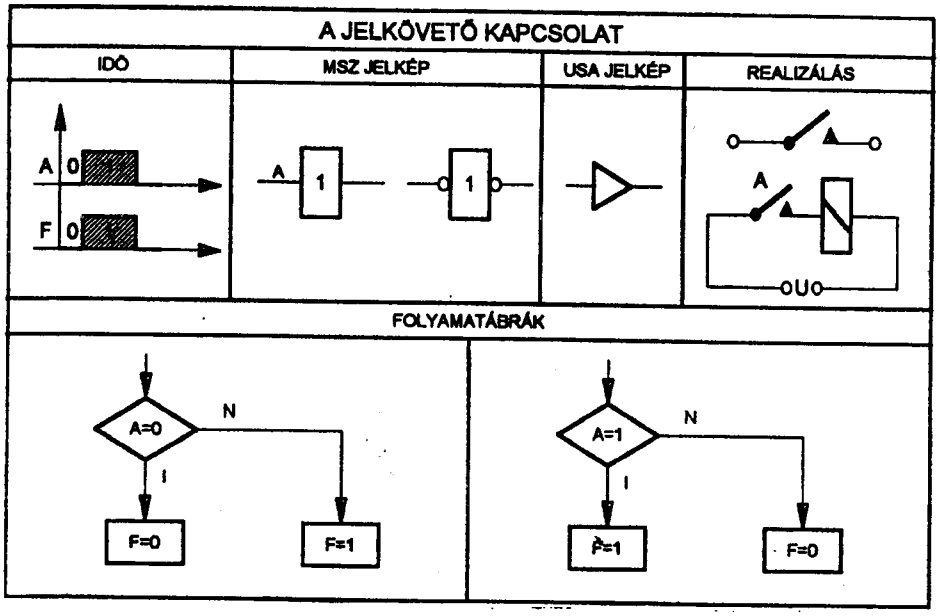
A másolási funkciót megvalósító eszköz a bemeneten fellépő jelváltást **időkéséssel** követi. Az előzőekben bevezetett hozzárendeléssel a relés hálózatokban a jelmásolás **záró** érintkezővel valósítható meg.

Egy vezérlő berendezésen belül a jelmásoló alkalmazásának oka lehet:

- érintkezők multiplikálása
- terhelési problémák megoldása (jelregenerálás)
- teljesítményerősítés (kapcsoló erősítő)
- időzítési problémák megoldása (jelkésleltetés).

A jelmásoló idődiagramja, MSz jelképe és relés realizálása a 3.7. ábrán látható.

A **direkt áramkörrel** működtetett relé **jelmásolási** feladatot lát el. A "konstans 0", illetve "konstans 1" a bemeneti változóktól nem függ, ezért ezekkel nem foglalkozunk.



3.7. ábra

Az SN 7406 típusú integrált áramkör 6 db jelmásolót tartalmaz.

Megjegyzés: a jelmásolást, illetve tagadást fizikailag realizáló áramkörök meghibásodása gyakran "konstans 0", illetve "konstans 1" függvényt eredményez.

3.2.2. Kétváltozós logikai függvények

Ha a kimeneti esemény két bemeneti változó értékétől függ, kétváltozós logikai függvényről beszélünk. Tizenhat ilyen függvény lehetséges (3.8. ábra).

	2^1 A	2^0 B	F_0^2	F_1^2	F_2^2	F_3^2	F_4^2	F_5^2	F_6^2	F_7^2	F_8^2	F_9^2	F_{10}^2	F_{11}^2	F_{12}^2	F_{13}^2	F_{14}^2	F_{15}^2
0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
2	1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
3	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

3.8. ábra

Megfigyelhető, hogy a táblázat tartalmazza az egyváltozós függvényeket is Ezek: $F_0^2, F_3^2, F_5^2, F_{10}^2, F_{12}^2, F_{15}^2$.

A továbbiakban csak a speciális kétváltozós függvényekkel foglalkozunk.

3.2.2.1. ÉS kapcsolat: $\mathbf{F_8^2(A,B) = A \wedge B = A \,\&\, B = A \cdot B = AB}$

Az olyan logikai kapcsolatot, amely **akkor és csak akkor** igaz, ha valamennyi bemenő változó értéke egyidejűleg igaz, **ÉS kapcsolatnak** nevezzük. További szokásos elnevezések: **AND művelet, konjunkció, logikai szorzás.** Utóbbira utal az ÉS művelet jelölése, a változók közé írt szorzásjel (•), amit gyakran el szokás hagyni. Az alsó decimális index képzése: (ld. 3.8. ábra bináris hozzárendelését)

$$1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 2^3 = 8_{10}.$$

A konjunkció műveletének értéktáblázata két változó esetén:

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

3.9. ábra

Az értéktáblázatból is látható, hogy a konjunkció művelete minden olyan esetben 0-s értékű, amikor legalább **egy bemeneti változó értéke 0.** A konjunkció - halmaz elméletéből vett analógia alapján - grafikus szemléltetésben **közös rész képzésnek (interszekció)** felel meg. Érintkezős hálózatokban a konjukció **záró érintkezők soros** kapcsolásával valósítható meg. Az ÉS kapcsolatot fizikailag realizáló áramkört **ÉS kapunak (AND gate)** nevezzük. Az SN 7408 típusú integrált áramkör **4 db kétbemenetű ÉS kaput** tartalmaz. Az ÉS kapu kimenetén tehát egyetlen esetben van 1 jel, ha valamennyi bemenetén 1 jel van, és minden olyan esetben 0 jel van a kimenetén, ha legalább egy bemenetén 0 jel van. Eszerint az **ÉS kapu bemenetén a 0 jel a meghatározó.**

Az ÉS művelet szabályai:

$$\begin{aligned}
&A \cdot 1 = A && \overline{A} \cdot 1 = \overline{A} && 0 \cdot 0 = 0 \\
&A \cdot 0 = 0 && \overline{A} \cdot 0 = 0 && 0 \cdot 1 = 0 \\
&A \cdot A = A && \overline{A} \cdot \overline{A} = \overline{A} && 1 \cdot 0 = 0 \\
&A \cdot \overline{A} = 0 && \overline{A} \cdot A = 0 && 1 \cdot 1 = 1 \qquad (3\text{-}5) \\
&A \cdot B = B \cdot A && && \rightarrow \textbf{kommutatív} \\
&A(B \cdot C) = (A \cdot B)C = ABC && && \rightarrow \textbf{asszociatív.}
\end{aligned}$$

A 3-5 szabályok igazságtáblázattal, vagy érintkezős analógiával könnyen beláthatók. A kommutatív tulajdonság a sorbakapcsolt érintkezők felcserélhetőségét, az asszociatív pedig kiemelhetőségét jelenti.
Az ÉS kapcsolat **tetszőleges számú (n)** változóra általánosítható:

$$F = X_1 \wedge X_2 \wedge .... \wedge X_n . \qquad (3\text{-}6)$$

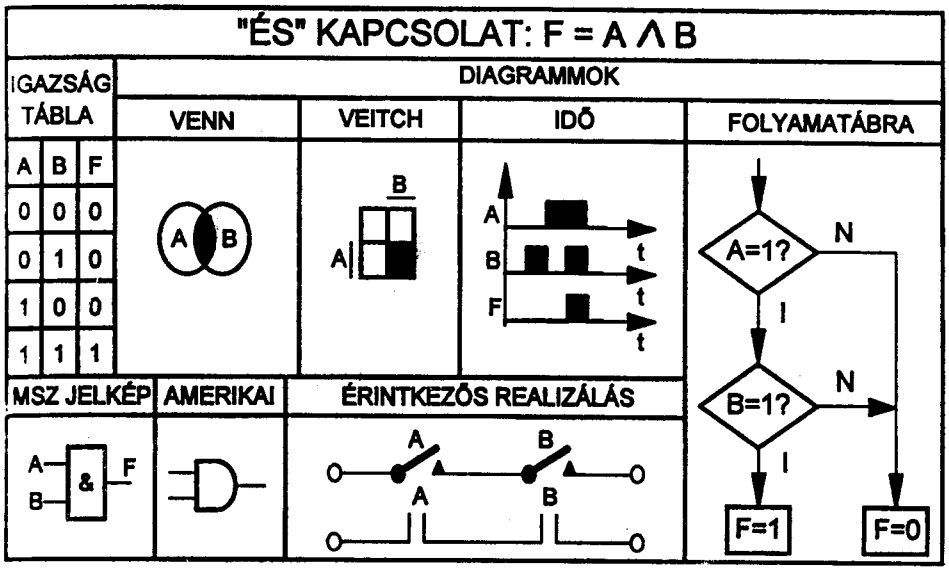
Az ÉS művelet bájtokra (szavakra) is kiterjeszthető. Ez esetben az ÉS műveletet a bájt (vagy szó) valamennyi azonos pozíciójú bitjére el kell végezni. Az ANA B utasítás az akkumulátor A és a B regiszter tartalma közötti ÉS műveletet az alábbiak szerint végzi:

Legyen $A = 3D \text{ hex} = 0011\ 1101_2$

$B = 97 \text{ hex} = \underline{1001\ 0111_2}$

Az ANA B utasítás után: $0001\ 0101_2 = 15$ hex.

Az ÉS művelettel kapcsolatos további tudnivalókat a 3.10. ábrán foglaltuk össze.



3.10. ábra

3.2.2.2. VAGY kapcsolat: $F_{14}^2(A,B) = A \vee B$

A VAGY függvény értéke egyetlen esetben **0**, ha valamennyi bemeneti változó értéke egyidejűleg **0**. Szokásos elnevezés még: **diszkjunkció, OR művelet, logikai összeadás.** Utóbbira utal korábbi írásjele (+), valamint hogy grafikus ábrázolásnál **területek egyesítését jelenti (unió).** Tehát a diszkjunció műveletének eredménye akkor igaz (1-es), ha a benne szereplő logikai változók közül **legalább egynek** az értéke igaz (1-es).

Az alsó decimális index képzése: $1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 14_{10}$.

A VAGY kapcsolat értéktáblázata kétváltozó esetén:

A	B	A∨B
0	0	0
0	1	1
1	0	1
1	1	1

A diszkjunkciót érintkezős hálózatokban **záró érintkezők párhuzamos** kapcsolásával lehet megvalósítani. Elektronikus logikai hálózatokban a diszkjunkció műveletét **VAGY kapu (OR gate)** realizálja. A VAGY művelet tetszőleges számú (n) változóra értelmezhető:

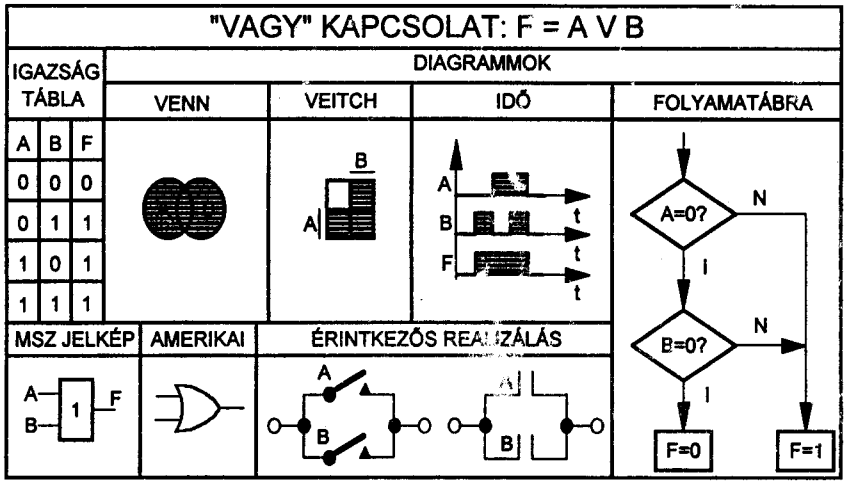
$$F = X_1 \vee X_2 \vee \ldots \vee X_n . \qquad (3\text{-}7)$$

A VAGY kapu kimenetén tehát akkor van nulla, ha valamennyi bemeneten egyidejűleg 0 jel van és akkor van 1, ha legalább egy bemeneten 1 jel van. Eszerint a **VAGY kapu bemenetén az 1 jel a domináló.**

A diszkjunkció műveleti szabályai:

$$A \vee 1 = 1 \qquad \overline{A} \vee 1 = 1 \qquad 0 \vee 0 = 0$$
$$A \vee 0 = A \qquad \overline{A} \vee 0 = \overline{A} \qquad 0 \vee 1 = 1$$
$$A \vee A = A \qquad \overline{A} \vee \overline{A} = \overline{A} \qquad 1 \vee 0 = 1$$
$$A \vee \overline{A} = 1 \qquad \overline{A} \vee A = 1 \qquad 1 \vee 1 = 1$$
$$A \vee B = B \vee A \qquad \rightarrow \textbf{kommutatív} \qquad (3\text{-}8)$$
$$A \vee (B \vee C) = (A \vee B) \vee C = A \vee B \vee C \qquad \rightarrow \textbf{asszociatív.}$$

A 3-8 szabályok igazságtáblázattal, vagy relés realizálással könnyen beláthatók. A VAGY művelettel kapcsolatos további ismereteket a 3.11. ábrán foglaltuk össze.



3.11. ábra

A VAGY művelet bájtokra (szavakra) is kiterjeszthető. Ez esetben a VAGY műveletet a bájt (vagy szó) **valamennyi azonos pozíciójú bitjére** el kell végezni. Az ORA C utasítás az akkumulátor (A) és a C regiszter tartalma közötti VAGY műveletet az alábbiak szerint végzi:

Legyen $A = 61$ hex $= 0110\,0001_2$

$C = C3$ hex $= \underline{1100\,0011_2}$

Az ORA C utasítás után: $1110\,0011_2 = E3$ hex.

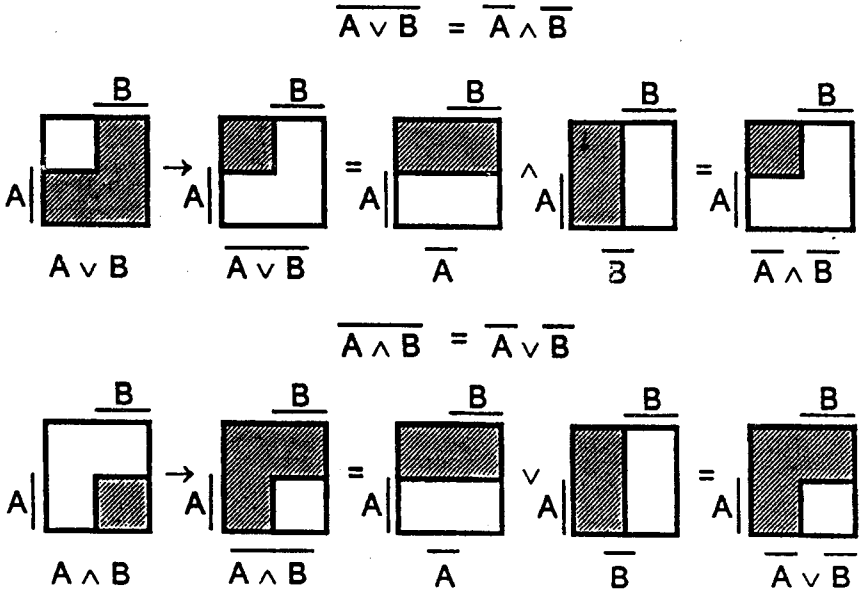
Láttuk az eddigiekben, hogy érintkezős realizálási módnál a változó igaz értékének záró érintkező, a változó negált értékének bontó érintkező, a konjukciónak soros kapcsolás, a diszkjunkciónak pedig párhuzamos kapcsolás felel meg. Itt említjük meg, hogy a NEM, ÉS, VAGY kapcsolatot logikai alapműveletnek tekintjük, mert velük valamennyi további logikai függvény megvalósítható (NÉV rendszer).

De Morgan szabályok

Az ÉS, illetve VAGY műveletek között a **De Morgan szabályok** teremtenek kapcsolatot a **NEM függvény** révén.

$$\overline{X_1 \vee X_2 \vee \ldots \vee X_n} = \overline{X_1} \wedge \overline{X_2} \wedge \overline{X_n}$$
$$\overline{X_1 \wedge X_2 \wedge \ldots \wedge X_n} = \overline{X_1} \vee \overline{X_2} \vee \overline{X_n}. \qquad (3\text{-}9)$$

Szavakban: logikai szorzat tagadottja egyenlő a változók negáltjának összegével, illetve összeg tagadottja egyenlő a változók negáltjának logikai szorzatával. A De Morgan szabályok könnyen beláthatók értéktáblázattal, vagy grafikus szemléltetéssel. (3.12. ábra).



3.12. ábra

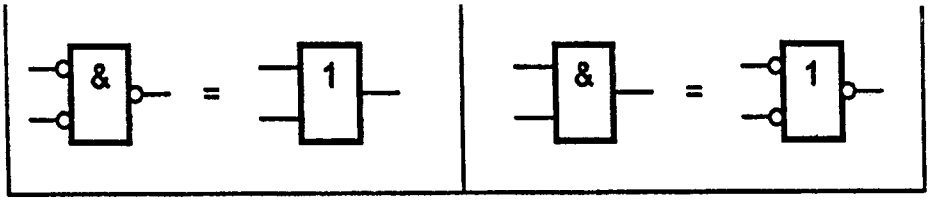
Két változó esetén:

$$\overline{A \wedge B} = \overline{A} \vee \overline{B}$$
$$\overline{A \vee B} = \overline{A} \wedge \overline{B}. \qquad (3\text{-}10)$$

A De Morgan szabályok alapján magyarázhatók a 3.13. ábrán megrajzolt ekvivalens MSZ jelképek.

A (3-10) összefüggések táblázatos igazolása.

A	B	$\overline{A}$	$\overline{B}$	AB	$\overline{A \wedge B}$	$\overline{A} \vee \overline{B}$	$A \vee B$	$\overline{A \vee B}$	$\overline{A} \wedge$
0	0	1	1	0	1	1	0	1	1
0	1	1	0	0	1	1	1	0	0
1	0	0	1	0	1	1	1	0	0
1	1	0	0	1	0	0	1	0	0



3.13. ábra

3.2.2.3. Antivalencia kapcsolat: $F_6^2(A,B) = \overline{A}B \vee A\overline{B} = A \forall B$

Ez a kétváltozós függvény akkor igaz, ha vagy csak az A, vagy csak a B igaz, vagyis amikor a bemeneti változók **ellentétes** értékűek. Használt elnevezések még: **kizáró VAGY, exclusív OR.**
Az antivalencia értéktáblázata:

A	B	$A \forall B$
0	0	0
0	1	1
1	0	1
1	1	0

Műveleti szabályai:

$$A \forall A = A\overline{A} \vee \overline{A}A = 0 \qquad 0 \forall 0 = 0$$
$$A \forall 1 = A0 \vee \overline{A}1 = \overline{A} \qquad 0 \forall 1 = 1$$
$$A \forall \overline{A} = AA \vee \overline{A}\,\overline{A} = 1 \qquad 1 \forall 0 = 1 \qquad (3\text{-}11)$$
$$A \forall 0 = A1 \vee \overline{A}0 = A \qquad 1 \forall 1 = 0$$
$$\overline{A} \forall 0 = \overline{A}1 \vee A0 = \overline{A}.$$

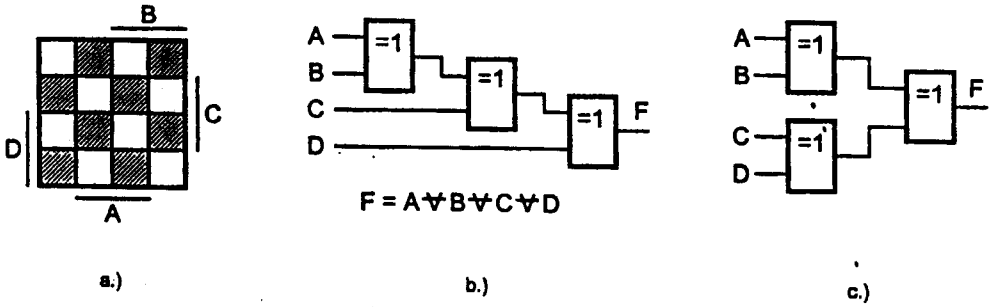
$$A \forall B = B \forall A$$

$$(A \forall B) \forall C = A \forall (B \forall C) = A \forall B \forall C$$

$$A \forall B = A\overline{B} \vee \overline{A}B = (A \vee B) \wedge (\overline{A} \vee \overline{B}) = (A \vee B)\overline{A \wedge B}$$

$$\overline{A \forall B} = \overline{A} \forall B = A \forall \overline{B}.$$

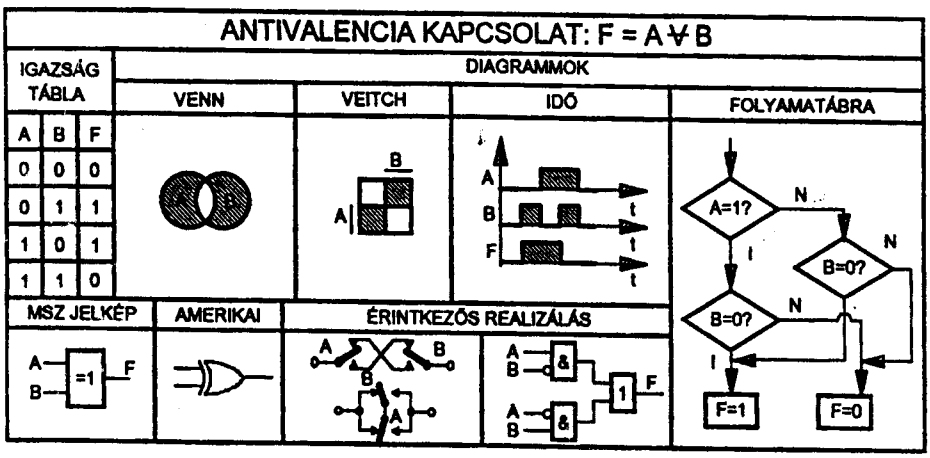
Az antivalencia művelet több változóra is értelmezhető. Példaként egy 4 változós antivalencia kétféle bővítési sémáját b), c) rajzoltuk meg a 3.14. ábrán.



a.) b.) c.)

3.14. ábra

Az antivalencia kapcsolatra vonatkozó további ismereteket a 3.15. ábrán találjuk.



3.15. ábra

A 7486 típusú integrált áramkör 4 db kétbemenetű ANTIVALENCIA kaput tartalmaz.

Programozható áramkörökben az ANTIVALENCIA függvényt realizáló utasítás szokásos elnevezései: XOR, XRA, EXOR.

Az ANTIVALENCIA kapcsolat bájtokra (szavakra) is kiterjeszthető. Ez esetben a XOR műveletet a bájt (vagy szó) valamennyi azonos pozíciójú bitjére el kell végezni. Az XRA D utasítás az akkumulátor (A) és a D regiszter tartalma közötti XOR műveletet az alábbiak szerint végzi:

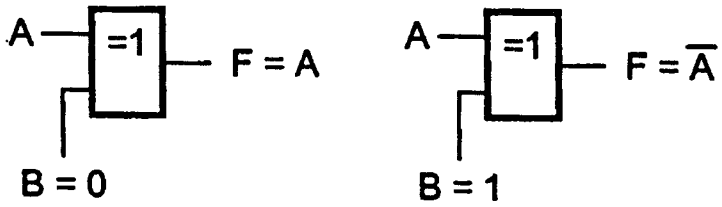
Legyen $A = B0 \text{ hex} = 1011\,0000_2$

$D = 67 \text{ hex} = \underline{0110\,0111_2}$

Az XRA D utasítás után: $1101\,0111_2 = D7 \text{ hex}.$

Itt említjük meg, hogy az erősáramú technikában az ANTIVALENCIA függvényt realizáló érintkezős kapcsolást **alternáló kapcsolásnak** nevezik és például lépcsőház világításnál alkalmazzák.

Az antivalencia kapu egy olyan vezérlő áramkörnek is felfogható, amelynél az egyik változó értékétől függően vagy jelmásol, vagy negál.(3.16. ábra).



3.16. ábra

Ezt a kapcsolást leggyakrabban a **programozható áramkörökben** használják.

3.2.2.4. Ekvivalencia kapcsolat: $F_9^2(A,B) = (\overline{A} \wedge \overline{B}) \vee (A \wedge B) = A \text{ A} B$

A logikai egyenletből látható, hogy a függvény a bemeneti változók **egyenértékűsége** esetén 1 értékű. Egymással egyenértékű minden igaz ítélet, illetve minden 1-es értékű változó és egymással egyenértékű minden hamis értékű ítélet, illetve minden 0-ás értékű változó. Az irodalomban **koincidencia, exclusiv NOR (XNOR, EXNOR)** elnevezés is használatos.

Az ekvivalencia értéktáblázata 2 változó esetén:

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

Műveleti szabályai:

$$A \wedge 0 = (\overline{A} \wedge 1) \vee (A \wedge 0) = \overline{A} \qquad 0 \wedge 0 = 1$$

$$A \wedge 1 = (\overline{A} \wedge 0) \vee (A \wedge 1) = A \qquad 0 \wedge 1 = 0$$

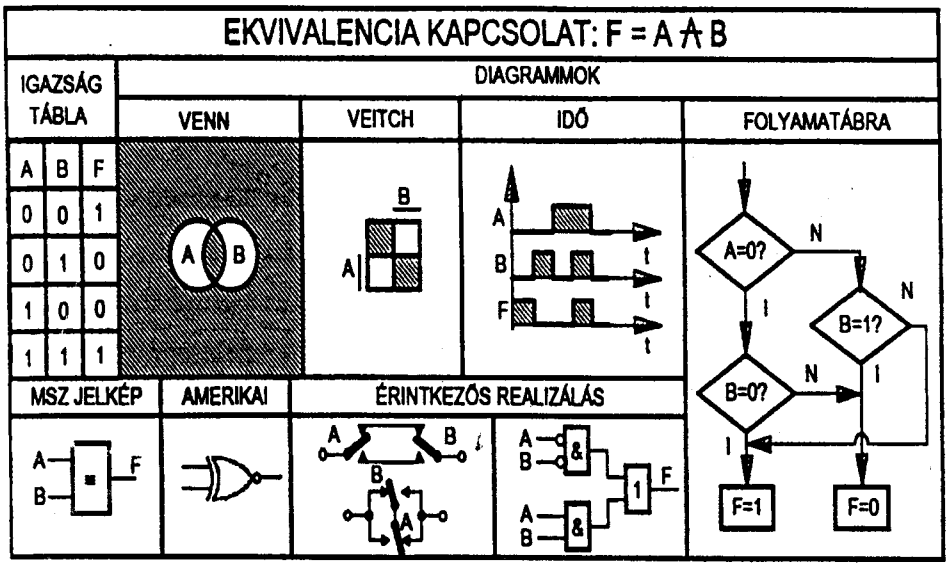
$$A \wedge A = (A \wedge A) \vee (\overline{A} \wedge \overline{A}) = 1 \qquad 1 \wedge 0 = 0$$

$$A \wedge \overline{A} = (\overline{A} \wedge A) \vee (A \wedge \overline{A}) = 0 \qquad 1 \wedge 1 = 1$$

$$A \wedge B = (\overline{A} \wedge \overline{B}) \vee (A \wedge B) = (\overline{A} \vee B) \wedge (A \vee \overline{B}) \qquad (3\text{-}12)$$

$$A \wedge B = B \wedge A \qquad \overline{A \forall B} = A \wedge B.$$

Az ekvivalencia függvénnyel kapcsolatos további ismereteket a 3.17. ábrán találhatjuk.



3.17. ábra

3.2.2.5. NEM - ÉS kapcsolat: $F_7^2(A,B) = \overline{A \wedge B} = \overline{A} \vee \overline{B}$

Az olyan logikai kapcsolatot, **amely akkor és csak akkor hamis, ha valamennyi független változó egyidőben igenleges értékű, NEM-ÉS függvénynek** (NOT AND röviden NAND kapcsolat) nevezzük.

A NEM-ÉS (röviden NÉS) függvény értéktáblázata a két változó esetén:

A	B	$\overline{A \wedge B}$	$A \wedge B$
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

A NÉS függvény értéktáblázatát a konjunkció értéktáblázatával összevetve kiderül, hogy **az ÉS, illetve NÉS kapcsolat egymás negáltja.**

A NÉS műveletet fizikailag realizáló áramkör neve NÉS kapu (NAND gate). **A NAND kapu kimenetén tehát csak akkor van 0 jel, ha valamennyi bemenetén 1 jel van.** Eszerint **a NAND kapu bemenetén a 0 jel a meghatározó.**

A NÉS művelet N változó esetén:

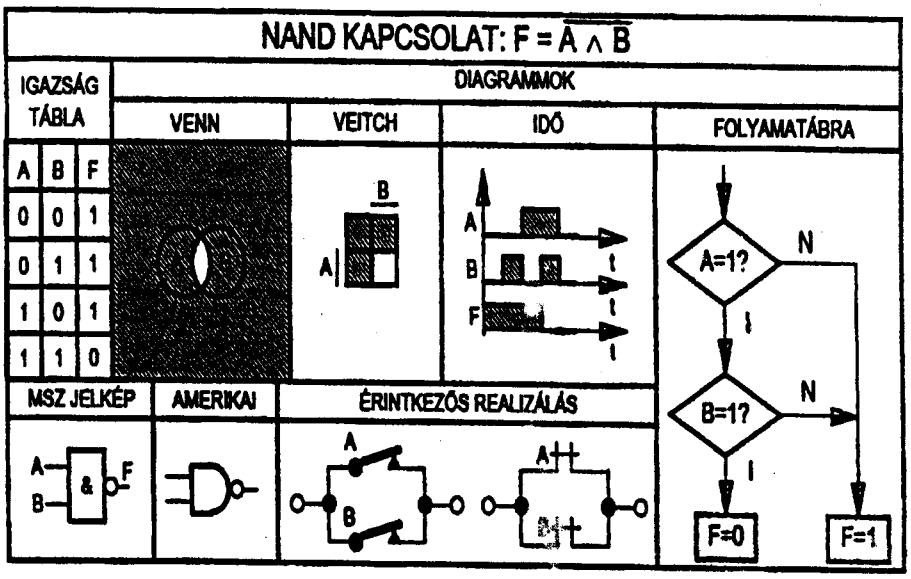
$$F = \overline{X_1 X_2 \ldots X_n} = \overline{X_1} \vee \overline{X_2} \vee \ldots \vee \overline{X_n} . \qquad (3\text{-}13)$$

A NÉS művelet szabályait az ÉS kapcsolatnál leírtak negálásával kaphatjuk.

Pl.: $\overline{A \wedge 0} = 1$

$\overline{A \wedge 1} = \overline{A}$.

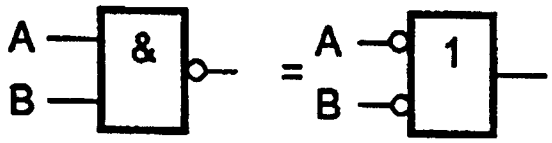
A NÉS művelettel kapcsolatos további ismereteket a 3.18. ábrán foglaltuk össze.



3.18. ábra

A 3.19. ábrán vázolt ekvivalens jelölések a De Morgan szabályok ismeretében követhetők.

A 7400 típusú integrált áramkör 4 db kétbemenetű NAND kaput tartalmaz.



3.19. ábra

3.2.2.6. NEM-VAGY kapcsolat: $F_1^2(A,B) = \overline{A \vee B} = \overline{A} \wedge \overline{B}$

A NEM-VAGY (röv. NVAGY) művelet a diszkjunkció műveletének a negáltja. A kimenet tehát csak akkor lesz 1-es értékű, ha valamennyi bemeneti változó értéke egyidejűleg 0. Gyakran használt angol elnevezése a NOT OR rövidítése: **NOR művelet.**

A NEM-VAGY művelet értéktáblázata két változó esetén:

A	B	$\overline{A \vee B}$	$A \vee B$
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

Az OR és NOR művelet tehát egymás negáltja.

A NOR művelet n változó esetén:

$$F = \overline{X_1 \vee X_2 \vee \ldots X_n} = \overline{X_1} \wedge \overline{X_2} \ldots \wedge \overline{X_n} . \qquad (3\text{-}14)$$

A NOR kapcsolat szabályait a diszkjunkciónál leírtak negálásával kaphatjuk.

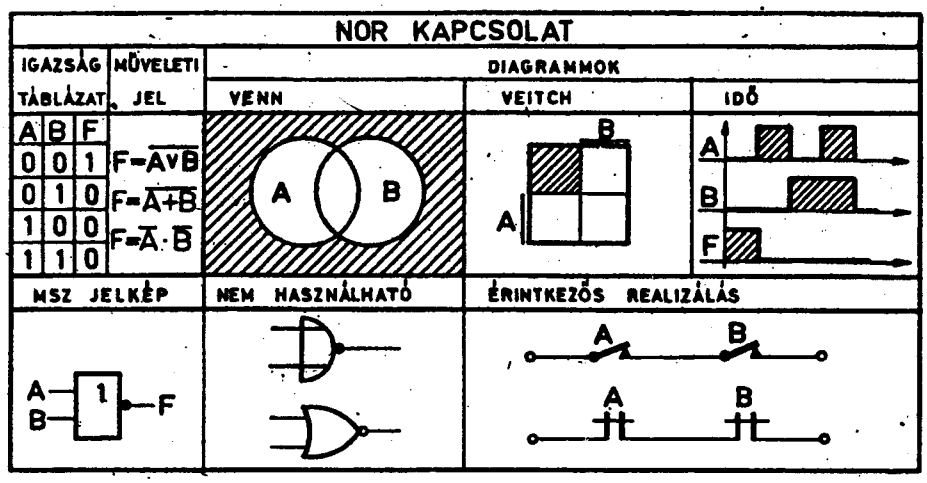
Pl.: $\overline{A \vee 0} = \overline{A}$

$\overline{A \vee 1} = 0$.

Az NVAGY műveletet fizikailag realizáló áramkört **NVAGY kapunak mondjuk (NOR gate).** A NOR kapu bemenetén az **1 jel a meghatározó.**

A 7402 típusú integrált áramkör 4 db kétbemenetű NOR kaput tartalmaz.

A NOR művelettel kapcsolatos további tudnivalókat a 3.20. ábrán foglaltuk össze.



3.20. ábra

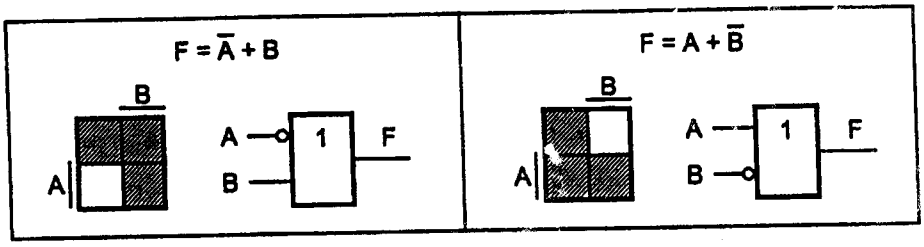
A NAND és a NOR műveleteket **univerzális műveleteknek** is nevezzük, mert bármelyikükkel **valamennyi logikai kapcsolat megvalósítható.**

3.2.2.7. Az implikáció: $F_{11}^{2}(A,B) = \overline{A} \vee B$

Az **implikáció** az eddigiektől eltérően **nem kommutatív**. Az implikáció csak akkor hamis (0), ha az előtag (A) igaz és az utótag (B) hamis.

Inverz implikáció: $F_{13}^{2}(A,B) = A \vee \overline{B}$.

Az implikáció Veitch diagramja és jelképe a 3.21. ábra szerinti. Az EKVIVALENCIA függvény a két implikáció ÉS kapcsolata.



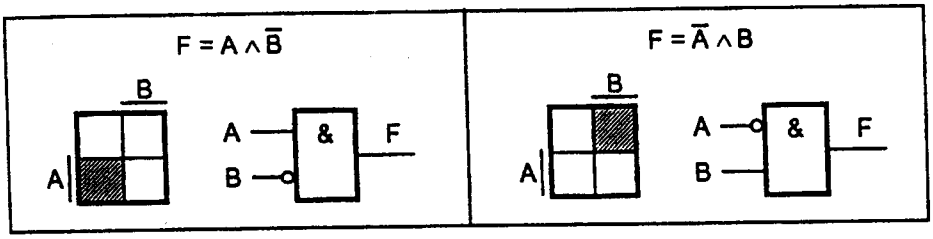
3.21. ábra

3.2.2.8. A inhibició: $F_{4}^{2}(A,B) = A \wedge \overline{B}$

Az inhibició (tiltás) művelete lényegében az implikáció negáltja. Így ez sem kommutatív. A művelet eredménye csak abban az esetben igaz, ha az előtag egyedül, önmagában igaz. Az utótag (B) igenleges értéke letiltja a kimenetet.

Az inverz inhibició: $F_2^2(A,B) = \bar{A} \wedge B$.

Az inhibició Veitch diagramja és jelképe a 3.22. ábra szerinti. Az ANTIVALENCIA függvény a két inhibició VAGY kapcsolata.



3.22. ábra

Megjegyzések:

a) Minden függvényhez tartozik egy másik **duális függvény**, amely csak abban tér el, hogy a benne az ÉS, illetve VAGY műveletek fel vannak cserélve.

b) Két függvény akkor **inverze** egymásnak, ha azonos bemeneti változó értékéhez tartozó függvény értékek egymás ellentettjei:

Duális páros		Inverz párok	
$A \wedge B$	$A \vee B$	$A \wedge B$	$\overline{A \wedge B}$
$\overline{A \vee B}$	$\overline{A \wedge B}$	$A \vee B$	$\overline{A \vee B}$
$\bar{A} \wedge B$	$\bar{A} \vee B$	$A \wedge \bar{B}$	$\bar{A} \vee B$
$A \wedge \bar{B}$	$A \vee \bar{B}$	$\bar{A} \wedge B$	$A \vee \bar{B}$
$A \forall B$	$A \wedge B$	$A \forall B$	$A \wedge B$.

3.2.3. Többváltozós logikai függvények megadási módjai

A képezhető kapcsolási függvények száma a független változók számával rohamosan nő (3-3). Nem célszerű tehát a 2-nél több bemeneti változót tartalmazó függvényeket egyenként tárgyalni, ugyanis bármely **n > 2 változós függvény kétváltozós függvényekből** felépíthető.

A kapcsolási függvények megadására a következő módszereket ismertetjük:

a) kombinációs táblázat
b) index-számos alak
c) grafikus módszer
d) kanonikus (normál) alakok.

3.2.3.1. Értéktáblázat

a) Egy n változós függvényt **táblázatos** módszerrel úgy adhatunk meg, hogy az értelmezési tartomány minden helyén - jelen esetben a bináris sorrendben felsorolt 2^n bemeneti kombinációhoz - megadjuk a függvény értékét. Példaként két n = 3 változós függvényt a 3.2. táblázatban adtuk meg.

3.2. táblázat

i	2^2	2^1	2^0	$F^3_{203}(A,B,C)$	$F^3(A,B,C)$
	A	B	C	(α_i)	(α_i)
0	0	0	0	1	0
1	0	0	1	1	1
2	0	1	0	0	x
3	0	1	1	1	1
4	1	0	0	0	x
5	1	0	1	0	0
6	1	1	0	1	1
7	1	1	1	1	x

Teljesen specifikáltnak (határozottnak) mondjuk a logikai függvényt, ha a bemeneti változók 00...0 és 11...1 közé eső 2^n kombinációhoz 1, illetve 0 értékek tartoznak. Ilyen esetben elegendő vagy csak az 1, vagy csak a 0 helyeket felsorolni. 2^{2^n} db ilyen függvény van. Erre láthatunk példát a 3.2. táblázatban megadott $F^2_{203}(A,B,C)$ függvény esetében.

Amennyiben a független változók 00....0 és 11...1 közé eső 2^n kombináció közül csak **m** kombinációhoz írják elő, amelyekben legyen F értéke 1, illetve 0, a függvényt **nem teljesen specifikáltnak mondjuk.**

A (2^n-m) kombinációban a függvény értéke tetszőlegesen 0-nak, vagy 1-nek vehető. Az ilyen kombinációkat **közömbös (érvénytelen, redundáns, don't care) kombinációknak** nevezzük. 2^m db ilyen függvény van. A nem teljesen határozott logikai függvény bemeneti kombinációit tehát három csoportba sorolhatjuk:

- a függvény 1 értékeihez tartozó kombinációk
- a függvény 0 értékeihez tartozó kombinációk
- közömbös kombinációk.

Erre láthatunk példát a 3.2. táblázat utolsó oszlopában.

A közömbös kombinációkat a logikai függvények egyszerűsítésénél lehet előnyösen felhasználni.

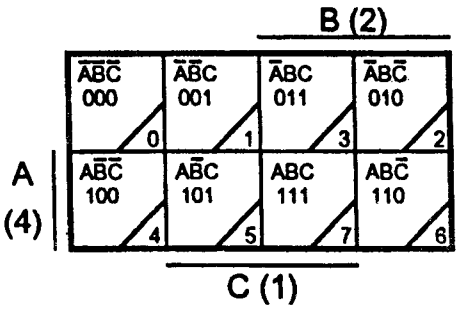
3.2.3.2. Index számos alak

Láttuk, hogy az egy és kétváltozós függvényeket meg lehet adni egy **decimális index** felhasználásával. Valamely n változós függvényt index számos alakban úgy adhatunk meg, hogy a függvényt jelentő F betű mellé felső indexbe a változók számát, alsó indexbe pedig a függvény értékekből képzett 2^n helyértékű bináris szám decimális megfelelőjét írjuk. Hangsúlyozzuk, hogy az így megadott függvény csak akkor egyértelmű, ha a **bemeneti változók sorrendjét** rögzítjük. A 3.2. táblázatban adott példabeli függvény indexe tehát:

$$1\cdot 2^7+1\cdot 2^6+0\cdot 2^5+0\cdot 2^4+1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+1\cdot 2^0=203.$$

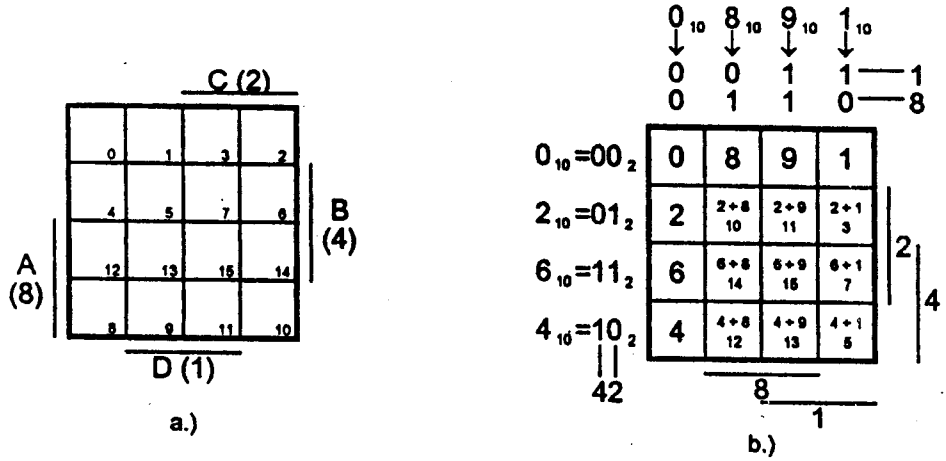
3.2.3.3. Grafikus megadás

Minél több a bemeneti változók száma, annál több munkát igényel a kombinációs táblázat kitöltése. Ilyenkor célszerűbb a Karnaugh-Veitch (továbbiakban: KV) táblák alkalmazása. A **KV tábla** tulajdonképpen módosított Veitch diagram, másszóval a kombinációs táblázat négyzethálós elrendezése. A táblázat **egy-egy cellája egy-egy bemeneti kombinációnak felel meg**. Eszerint egy **n változós KV táblának 2^n cellája van**. Az egyes cellákhoz a bemeneti kombinációkat sokféleképpen hozzá lehet rendelni, de - a későbbiekben ismertetett okok miatt - az olyan elrendezések használatosak, amelyeknél egymás mellett **szomszédos bemeneti kombinációk** vannak. **Két bemeneti kombináció szomszédos, ha közöttük csak egy változó értékben van eltérés**. A 3.23. ábrán egy háromváltozós KV tábla szerkesztését láthatjuk.



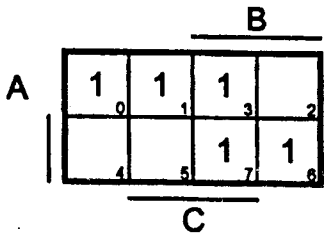
3.23. ábra

Célszerűen a Veitch diagramból indulunk ki, s minden cellába beírjuk a képviselt független változó kombinációt. Ha a változókhoz a kombinációs táblázatban rögzített súlyozást rendeljük, akkor az elemi kombinációk egy-egy **decimális számmal** jellemezhetők. A 3.24. ábrán a **leggyakrabban használt peremezésű négyváltozós** KV tábla, a b) ábrán pedig tetszőleges peremezésű KV tábla másik szerkesztési módja látható.



3.24. ábra

Figyeljük meg, hogy a decimális indexek a súlyozásokhoz vannak rendelve. Tehát egy F(DCBA) sorrendű megadás esetén a D változó lesz a 8-as súlyozású, így az kerül a 8-as peremezéshez. Tetszőleges n változós függvény **grafikus megadása** azt jelenti, hogy egy n változós KV tábla celláiba beírjuk a függvény 1,0 és közömbös (x) értékeit. Az $F^{3}_{203}(A,B,C)$ függvényt grafikus módszerrel a 3.25. ábrán adtuk meg.



3.25. ábra

3.2.3.4. Teljes diszjunktív normál alak

Az előzőekben láttuk, hogy a függvény megadható 1, illetve 0 helyeinek feltüntetésével. Ha előírt tulajdonságú függvényt akarunk előállítani képlettel, kétváltozós függvényekből felépített kifejezéssel, akkor egyaránt kiindulhatunk a függvény 1, illetve 0 helyeiből.

Ismeretes, hogy a **logikai szorzat** (ÉS kapcsolat) egyetlen egy esetben igaz, amikor minden tényezője egyidejűleg igenleges értékű. Ezért az előállítandó függvény minden 1 helyéhez hozzárendelünk egy logikai szorzatot, amely az előírt helyen 1 értékű, s ezeket a szorzatokat összeadjuk. (Természetesen

szorzás alatt konjunkciót, összeadás alatt diszkjunkciót értünk). Pl. a 3.2. táblázat alapján:

$$F^3_{203}(A,B,C)=\overline{A}\,\overline{B}\,\overline{C}\vee\overline{A}\,\overline{B}C\vee\overline{A}B\overline{C}\vee A\overline{B}\overline{C}\vee ABC.$$

Normál alakúnak mondjuk a függvényt, ha a változók **szorzatainak összegéből** (diszkjunktív normál alak), vagy **összegeinek szorzatából** (konjuktív normál alak) áll. A függvény **teljes normál alakjának** nevezzük azt a normál alakot, amelynek minden tagjában (diszkjunktív alak), illetve tényezőjében (konjuktív alak) szerepel a függvény összes változója igenleges (ponált), vagy nemleges (negált) alakban.

A teljes diszjunktív alakban szereplő logikai szorzatokat (ÉS kapcsolatokat) **mintermeknek** nevezzük. Az elnevezés onnan származik, hogy ezek a tagok a KV táblán egy-egy cellát (minimális termet) foglalnak el. Figyeljük meg, hogy a KV tábla egy-egy cellája a beírt decimális index számú mintermnek felel meg. Ezért a KV táblát **minterm táblának** is szokás nevezni.

A teljes diszkjunktív normál alak képzési szabálya: ha a függvény szóban forgó 1 helyén a változó értéke $x_i = 0$, akkor $\overline{x}_i$-et, ha pedig $x_i = 1$, akkor x_i-t írunk. Az összes változóból szorzatot képezünk, majd a függvény valamennyi 1 helyére ily módon képzett szorzatokat összeadjuk. A minterm fogalmának bevezetésével a teljes diszjunktív normál alakú függvény a 3.17. szerinti alakban írható.

$$F(X_1X_2\ldots X_n)=\sum_{i=0}^{2^n-1}\alpha_i^n m_i^n. \qquad (3\text{-}15)$$

Ahol α_i^n a **karakterisztikus együttható** értéke 1, vagy 0 attól függően, hogy az i-edik mintermet tartalmazza-e a függvény, vagy nem.

Pl.:

$$F^3_{203}(A,B,C)=\alpha_0^3 m_0^3\vee\alpha_1^3 m_2^3\vee\alpha_3^3 m_3^3\vee\alpha_4^3 m_4^3\vee\alpha_5^3 m_5^3\vee\alpha_6^3 m_6^3\vee\alpha_7^3 m_7^3$$

$$\alpha_0^3=\alpha_1^3=\alpha_3^3\vee\alpha_6^3=\alpha_7^3=1$$

$$\alpha_2^3=\alpha_4^3=\alpha_5^3=0$$

$$F^3_{203}(A,B,C)=m_0^3\vee m_1^3\vee m_3^3\vee m_6^3\vee m_7^3.$$

Egyszerűbben **(mintermes alakban):**

$$F^3_{203}(A,B,C)=\Sigma(0,1,3,6,7).$$

A nem teljesen specifikált függvényeknél a redundáns mintermeket is fel kell tüntetni. Például a 3.2. táblázatban adott függvény:

$$F^3(A,B,C) = \Sigma(1,3,6) \vee \Sigma_X(2,4,7).$$

3.2.3.5. Teljes konjunktív normál alak

A **teljes konjunktív** normál alak felírásához a függvény 0 helyeiből kell kiindulni. Az F^3_{203} (A,B,C) három esetben 0 értékű. Írjuk fel ezeket, mint az $\overline{F}$ 1 helyeit diszjunktív alakban:

$$\overline{F^3_{203}}(A,B,C) = \bar{A}B\bar{C} \vee A\bar{B}\bar{C} \vee A\bar{B}C.$$

Alkalmazzunk a De Morgan szabályt:

$$F^3_{203} = \overline{\overline{F^3_{203}}} = \overline{\bar{A}B\bar{C} \vee A\bar{B}\bar{C} \vee A\bar{B}C} = \overline{\bar{A}B\bar{C}} \& \overline{A\bar{B}\bar{C}} \& \overline{A\bar{B}C}$$

$$F^3_{203}(A,B,C) = (A \vee \bar{B} \vee C)(\bar{A} \vee B \vee C)(\bar{A} \vee B \vee \bar{C}).$$

Az ilyen alakban adott függvény - amely tehát **összegek szorzatából** áll és az összeg minden tagja valamennyi független változót tartalmazza ponált, vagy negált értékével - **teljes konjuktív normál** alakúnak mondjuk. E függvény elemi összegeit **maxtermeknek** $\left(M^n_i\right)$ nevezzük.

Egy logikai függvényt a fentiek alapján előállíthatunk oly módon, hogy a 0 helyeit valósítjuk meg. Ismeretes, hogy egy **logikai összeg** egyetlen egy esetben 0, amikor minden tagja egyidejűleg 0. Tehát az előállítandó függvény minden 0 helyéhez hozzárendelünk egy logikai összeget, amely a **szóban forgó** helyen **0**. Ezt az összeget úgy képezzük, hogy ahol a kérdéses változó $x_i = 0$, ott x_i-t, ahol $x_i = 1$, ott $\bar{x}_i$ -et írunk.

Végül a függvény valamennyi 0 helyére ily módon képzett összeget összeszorozzuk. A maxterm forgalmának bevezetésével valamely függvény konjunktív normál alakban a 3-16. szerint adható meg.

$$F(X_1X_2\ldots X_n) = \prod_{i=0}^{2^n-1} \left(\alpha^n_i \vee M^n_{2^n-1-i}\right). \qquad (3\text{-}16)$$

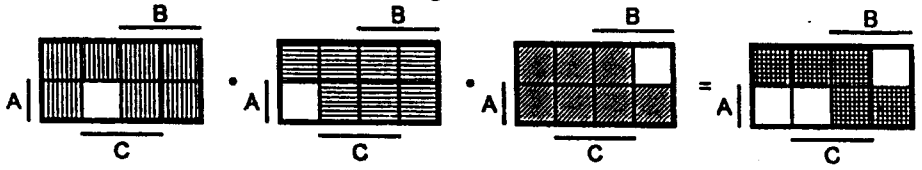
Pl.:

$$F^3_{203}(A,B,C) = (1 \vee M^3_7)(1 \vee M^3_6)(0 \vee M^3_5)(1 \vee M^3_4)(0 \vee M^3_3)(0 \vee M^3_2)(1 \vee M^3_1)(1 \vee M^3_0) = M^3_5 \wedge M^3_3 \wedge M^3_2.$$

Egyszerűbben **(maxtermes alakban)**:

$$F^3_{203}(A,B,C) = \prod(2,3,5).$$

Ennek megvilágítására tekintsük meg a 3.26. ábrát.



3.26. ábra

A mintermek és maxtermek közötti kapcsolat megállapítása céljából írjuk fel a háromváltozós mintermeket és maxtermeket (3.3. táblázat).

A táblázat alapján beláthatók a 3-17, illetve 3-18. összefüggések.

$$\overline{m_i^n} = M_{2^n-1-i}^n \cdot \qquad (3\text{-}17)$$

$$\overline{M_i^n} = m_{2^n-1-i}^n \cdot \qquad (3\text{-}18)$$

Ha n független változó van, akkor 2^n különböző m_i^n és ugyanannyi M_i^n létezik.

Könnyen beláthatók a 3-19 szerinti összefüggések is.

$$m_j^n \wedge m_k^n = 0, \text{ ha } j \neq k \qquad (3\text{-}19)$$

$$M_s^n \vee M_p^n = 1, \text{ ha } s \neq p.$$

3.3. táblázat

1	2^2 A	2^1 B	2^0 C	m_i^3	M_i^3
0	0	0	0	$m_0^3 = \overline{A}\,\overline{B}\,\overline{C}$	$M_0^3 = \overline{A} \vee \overline{B} \vee \overline{C}$
1	1	1	1	$m_1^3 = \overline{A}\,\overline{B}C$	$M_1^3 = \overline{A} \vee \overline{B} \vee C$
2	0	1	0	$m_2^3 = \overline{A}B\overline{C}$	$M_2^3 = \overline{A} \vee B \vee \overline{C}$
3	0	1	1	$m_3^3 = \overline{A}BC$	$M_3^3 = \overline{A} \vee B \vee C$
4	1	0	0	$m_4^3 = A\overline{B}\,\overline{C}$	$M_4^3 = A \vee \overline{B} \vee \overline{C}$
5	1	0	1	$m_5^3 = A\overline{B}C$	$M_5^3 = A \vee \overline{B} \vee C$
6	1	1	0	$m_6^3 = AB\overline{C}$	$M_6^3 = A \vee B \vee \overline{C}$
7	1	1	1	$m_7^3 = ABC$	$M_7^3 = A \vee B \vee C$

Negált függvény kanonikus alakjainak képzése a 3-20, 3-21. szerint történik.

$$\overline{F}(X_1X_2\ldots X_n) = \sum_{i=0}^{2^n-1}\left(\overline{\propto_i^n} \wedge m_i^n\right) \qquad (3\text{-}20)$$

$$\overline{F}(X_1X_2\ldots X_n) = \prod_{i=0}^{2^n-1}\left(\overline{\propto_i^n} \vee M_{2^n-1-i}^n\right) \qquad (3\text{-}21)$$

A negált függvényben ugyanazon indexű termek közömbösek, mint a ponált függvényben. Megjegyezzük, hogy a logikai függvények különböző megadási módjainak megvannak a maga előnyei és ugyanakkor hátrányai is. Például a szöveges feladat megadása legkönnyebben kombinációs táblázat segítségével történhet, míg a minimalizáláshoz (ld. később) a grafikus megadás a legcélszerűbb.

3.3. A BOOLE algebra összefüggései

A Boole algebra szabályait az egyes műveletek kapcsán ismertettük. Az áttekinthetőség céljából a fontosabb összefüggéseket a 3.4. táblázatban foglaltuk össze. A táblázat tartalmazza a disztributivitás szabályait is:

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C).$$

Speciális esetet képeznek az alábbi összefüggések:

$$A \wedge (A \vee B) = AA \vee AB = A(1 \vee B) = A$$

$$A \wedge (\overline{A} \vee B) = A \wedge B$$

$$A \vee (\overline{A} \wedge B) = A \vee B$$

$$(A \vee B) \wedge (A \vee C) = A \vee (B \wedge C)$$

Ezen összefüggések mind értéktáblázattal, mind Veitch diagrammal igazolhatók.

3.4. táblázat

$(\vee)$	$(\wedge)$	Megjegyzés
$A = 0$, ha $A \neq 1$ $\overline{0} = 1$	$A = 1$, ha $A \neq 0$ $\overline{1} = 0$	A mennyiségek kétértékűek
$0 \vee 0 = 0$ $0 \vee 1 = 1 \vee 1 = 1 \vee 0 = 1$	$0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0$ $1 \wedge 1 = 1$	Mennyiséggel végzett műveletek szabályai

$A \vee 0 = 0 \vee A = A$ $A \vee 1 = 1 \vee A = 1$ $A \vee A = A$ $A \vee \overline{A} = 1$	$A \wedge 0 = 0 \wedge A = 0$ $A \wedge 1 = 1 \wedge A = A$ $A \wedge \overline{A} = 0$ $A \wedge A = A$	Egy változóval végzett műveletek szabályai
$AB \vee \overline{A}B = (A \vee \overline{A})B = B$ $A \vee B = B \vee A$ $A \vee AB = A$ $A \vee \overline{A}B = A \vee B$	$(A \vee B)(\overline{A} \vee B) = 0 \vee \overline{A}B \vee AB = B$ $A \wedge B = B \wedge A$ $A(A \vee B) = A$ $A(\overline{A} \vee B) = AB$	Két változóval végzett műveletek szabályai
$(A \vee B) \vee C = A \vee (B \vee C)$ $= A \vee B \vee C$ $AB \vee AC = A \vee (B \vee C)$ $\overline{A \vee B \vee C} = \overline{A} \wedge \overline{B} \wedge \overline{C}$	$(A \wedge B)C = A(B \wedge C) = A \wedge B \wedge C$ $(A \vee B)(A \vee C) = A \vee BC$ $\overline{A \wedge B \wedge C} = \overline{A} \vee \overline{B} \vee \overline{C}$	Három változóval végzett műveletek szabályai De Morgan tételek

3.4. Logikai függvények minimalizálása

Egy digitális berendezés ára a beépített elemek számával nő, ezért fontos azon módszerek ismerete, amelyek alapján az adott működésű kapcsoló rendszert a lehető legolcsóbban, a legkevesebb építőelem felhasználásával tudunk elkészíteni. Ez indokolja a kanonikus alakú logikai függvények egyszerűsítési (minimalizálási) módszereinek megismerését.

A minimalizálás célja: alkatrészek (érintkezők, diódák, tranzisztorok, integrált áramkörök), illetve utasítások (memóriakapacitás, végrehajtási idő) megtakarítása.

Néhány minimalizálási szempont:

a) legegyszerűbb algebrai alak (minimálalak)
b) legkevesebb elemi áramkör
c) legkevesebb IC tok
d) azonos áramköri elemek
e) legkedvezőbb tranziens viselkedés
f) minimális áramköri fokozatok száma.

A minimalizálási eljárások alapja a 3-22. szerinti két összefüggés.

$$\mathbf{AB \vee A\overline{B} = A\left(B \vee \overline{B}\right) = A \wedge 1 = A}$$
$$\mathbf{\left(A \vee B\right)\left(A \vee \overline{B}\right) = AA \vee AB \vee A\overline{B} \vee 0 = A.} \qquad (3\text{-}22)$$

Egy logikai függvény algebrai alakját **minimál alaknak** nevezzük akkor, ha azonos átalakításokkal nem hozható kevesebb változót tartalmazó alakra.

A kapcsolási függvények minimalizálására a következő módszereket ismertetjük:

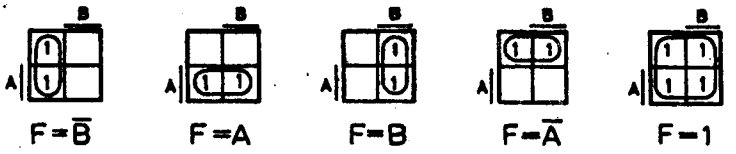
a) algebrai
b) grafikus
c) numerikus.

3.4.1. Algebrai módszer

A logikai függvények algebrai módszerrel történő egyszerűsítésén a Boole algebra szabályainak ismételt alkalmazását értjük. A módszert a grafikus eljárással párhuzamosan példák kapcsán mutatjuk be.

3.4.2. Grafikus minimalizálás

Ha valamely függvény diszjunktív kanonikus alakjában két olyan minterm fordul elő, melyek egymással **szomszédosak**, akkor ezek egyetlen szorzattá vonhatók össze. Ismeretes, hogy a célszerűen megválasztott peremezésű KV táblán az **algebrailag szomszédos** kombinációk **helyileg is szomszédosak**. Ez ad lehetőséget a grafikus minimalizálásra. Két 1-gyel jelölt szomszédos minterm egyetlen tömbbé vonható össze. A 3.27. ábrán a kétváltozós függvényeknél lehetséges összevonásokat tüntettük fel.



3.27. ábra

Végezzük el az egyszerűsítéseket algebrai módszerrel is.

$$F^2(A,B)=\Sigma(0,2)=\bar{A}\,\bar{B}\vee A\bar{B}=\bar{B}(A\vee\bar{A})=\bar{B}$$

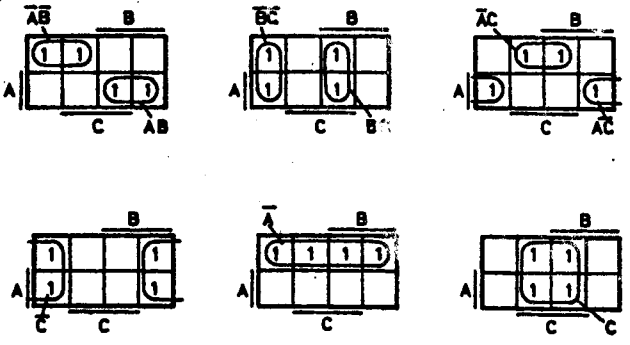
$$F^2(A,B)=\Sigma(2,3)=A\bar{B}\vee AB=A(\bar{B}\vee B)=A$$

$$F^2(A,B)=\Sigma(1,3)=\bar{A}B\vee AB=B(\bar{A}\vee A)=B$$

$$F^2(A,B)=\Sigma(0,1)=\bar{A}\,\bar{B}\vee\bar{A}B=\bar{A}(\bar{B}\vee B)=\bar{A}$$

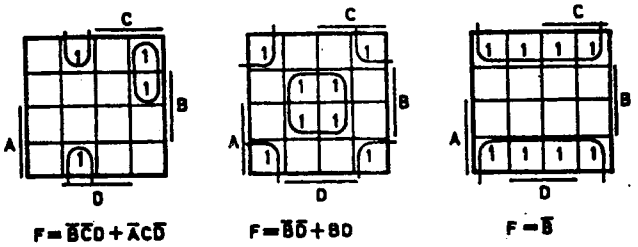
$$F^2(A,B)=\Sigma(0,1,2,3)=\bar{A}\,\bar{B}\vee\bar{A}B\vee A\bar{B}\vee AB=\bar{A}(\bar{B}\vee B)\vee A(\bar{B}\vee B)=\bar{A}\vee A=1.$$

A 3.28. ábrán a három változós KV táblák néhány jellegzetes kettős, illetve négyes tömbjeit szemléltetik.



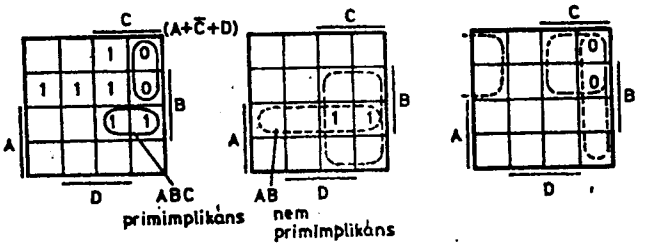
3.28. ábra

A 3.29. ábrán a négyváltozós függvények tömbösítésére láthatunk példákat.



3.29. ábra

A legnagyobb tömbök létrehozásakor a függvény primimplikánsait keressük meg. Az F függvény **primimplikánsának** nevezzük a változóknak (ponáltak, vagy negáltak), olyan szorzatát (diszjunktív alak), vagy **összegét** (konjuktív alak), melyet az F függvény még **tartalmaz**, de bármely változóját elhagyva a függvény már nem tartalmazza. Jól szemlélteti ezt a 3.30. ábra, ahol az ABC, illetve $\left(A \vee \overline{C} \vee D\right)$ a függvény primimplikánsai, de bármely változót elhagyva (pl.: AB) a függvény már nem tartalmazza.



3.30. ábra

A logikai függvények minimalizálási eljárása a primimplikánsok megkereséséből, majd pedig a szükséges primimplikánsok kiválasztásából áll. A primimplikánsok keresésének lépései grafikus módszernél a következők:

a) **Primimplikánsok keresése**:

1. Ábrázoljuk a függvényt KV táblán,
2. A 2^i számú szimmetrikusan elhelyezkedő szomszédos 1-gyel jelölt cellát egy tömbbé vonunk össze, amikor is i db. változó esik ki, ahol $i = 0, 1, 2 \ldots n$,
3. Mindig a lehető **legnagyobb tömböt** célszerű kialakítani,
4. Valamennyi 1-gyel jelölt cellának **legalább** egy tömbben szerepelnie kell,
5. Ugyanazon cella **több** tömbnek is eleme lehet,
6. A KV táblák négy változóig széleiken összefüggőnek tekinthetők.

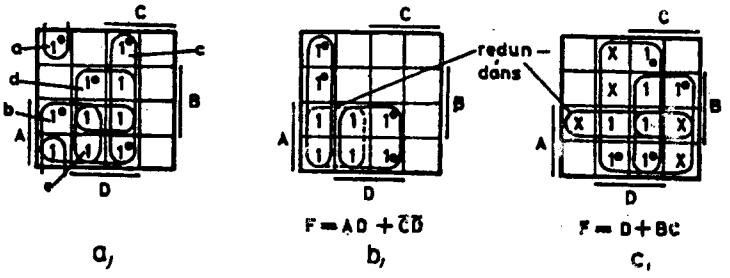
b) **A szükséges primimplikánsok kiválasztása**

Valamely primimplikáns **lényeges**, ha tartalmaz olyan m_i^n mintermet, amelyet minden más primimplikáns már nem tartalmaz. Az ilyenek tehát a függvény megvalósításához **nélkülözhetetlenek**. Azon tömbök lesznek a minimalizált függvény **szükséges** primimplikánsai, amelyek a függvény valamennyi 1-gyel jelölt cellájának egyszeri lefedéséhez **elengedhetetlenül** szükségesek.

A szükséges primimplikánsok kiválasztásának lépései:

1. Jelöljük meg egy-egy ponttal azon mintermeket, amelyeken csak egy hurok megy keresztül. Ezen tömbök lesznek a **nélkülözhetetlen primimplikánsok.**
2. Vonalkázzuk be a nélkülözhetetlen primimplikánsok által lefedett mintermeket.
3. Maradt-e olyan 1-gyel jelölt minterm, amelyet a nélkülözhetetlen primimplikánsok nem fednek le?
4. A fennmaradó 1-ek lefedésére válasszuk a legkevesebb és legnagyobb tömböt.

A primimplikánsok meghatározása - azaz a tömbök leolvasása a KV táblákból - úgy történik, hogy a tömböknek megfelelő szorzatokban csak azok a betűk szerepelnek, (pontáltan, vagy negáltan), amelyek a tömbök valamennyi cellájában szerepelnek. A 3.31. a) ábrán valamennyi primimplikáns lényeges, a b) ábrán viszont a fentiek értelmében az $A\overline{C}$ tömb redundáns.

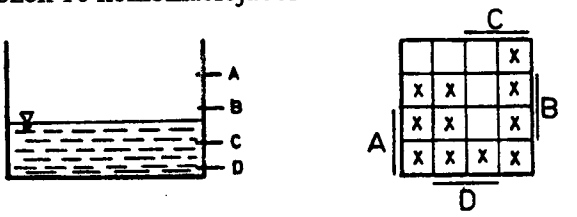


3.31. ábra

Ha a logikai függvény **nem teljesen specifikált**, akkor a közömbös mintermeket x-szel jelöljük és célszerűen azon x-eket, amelyek **nagyobb tömbök** létrehozását teszik lehetővé x = 1-nek, a többieket x = 0-nak vesszük. A **primimplikánsok kiválasztásánál csak a 1-gyel jelölt mintermeket** kell lefedni.

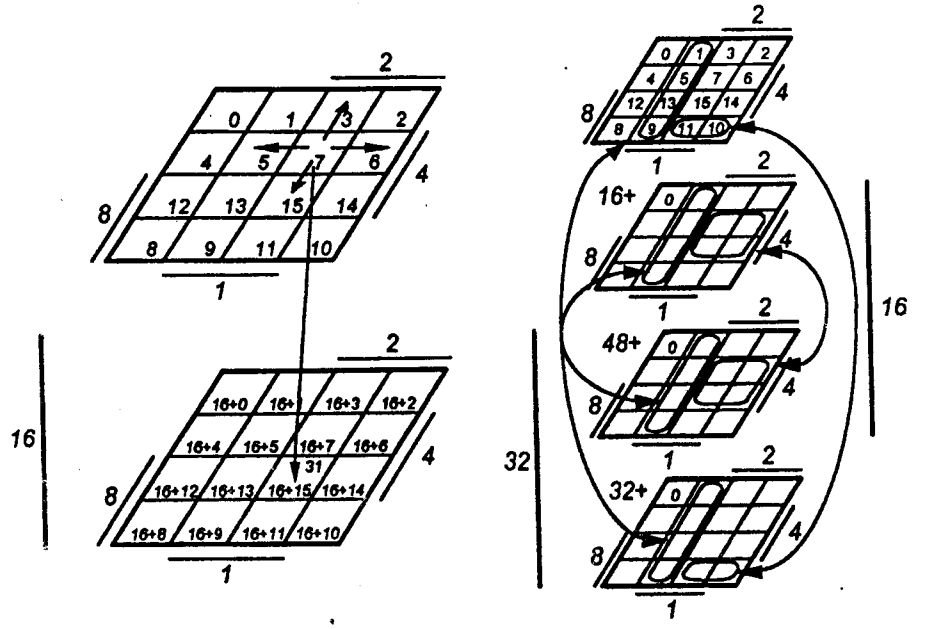
Tisztán x-eket tartalmazó tömb létrehozása értelmetlen. Közömbös kombinációkkal a gyakorlatban akkor találkozunk, ha a feladat fizikai jellegénél fogva ezek üzemszerűen nem fordulnak elő. Például tartályszint

vezérlésnél (3.32. ábra) a szintérzékelést A, B, C, D érzékelők végzik, melyek akkor adnak 1 jelet, ha a folyadék a kérdéses érzékelőt ellepte. Belátható, hogy a független változók 16 kombinációjából csak 5 fordulhat elő.



3.32. ábra

A grafikus eljárás **szemléletessége** egyben a módszer alkalmazásának gátját is jelenti. Ugyanis n változó esetén valamennyi 2^n kombinációnak n darab szomszédja létezik. Mivel egy négyzet oldalaihoz csak négy szomszédos négyzet rajzolható, így a síkban ábrázolt KV táblán csak n = 4 változó esetén kerülnek helyileg egymás mellé az összevonható kombinációk. Tehát négynél több változó esetén a módszer csak nagy körültekintéssel végezhető. Az öt és hatváltozós függvények **térbeli KV táblán** megfelelő gyakorlattal ábrázolhatók és egyszerűsíthetők. Az öt és hatváltozós KV tábla szerkesztését a 3.33. ábrán láthatjuk.



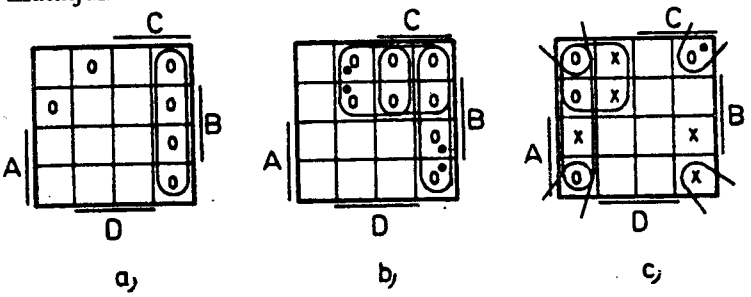
3.33. ábra

Az ábra szerinti elrendezésnél a minterm számok könnyen megállapíthatók. Néhány jellegzetes összevonható mintermet (tömböt) is feltüntettünk az ábrán. A tömbösítést először az egyes négyváltozós táblákon célszerű elvégezni, majd

a táblák közötti összevonási lehetőségeket megkeresni.

Az F függvény **minimalizált konjuktív** alakját úgy határozhatjuk meg, hogy a fentiekben leírt módon előállítjuk az $\overline{F}$ diszjunktív minimál alakját, majd az ilyen alakban felírt $\overline{F}$ -et negáljuk.

A 3.31. ábrán bemutatott függvények konjuktív alakjának meghatározását a 3.34. ábrán találjuk. A logikai függvények grafikus minimalizálását példák kapcsán mutatjuk be.



3.34. ábra

1. Példa:

Egyszerűsítsük az

$$F(A,B,C,D) = \Sigma(0,2,3,4,8,10,11,14,15) + \Sigma_X(1,12,13)$$

függvényt diszjunktív, illetve konjuktív alakban.

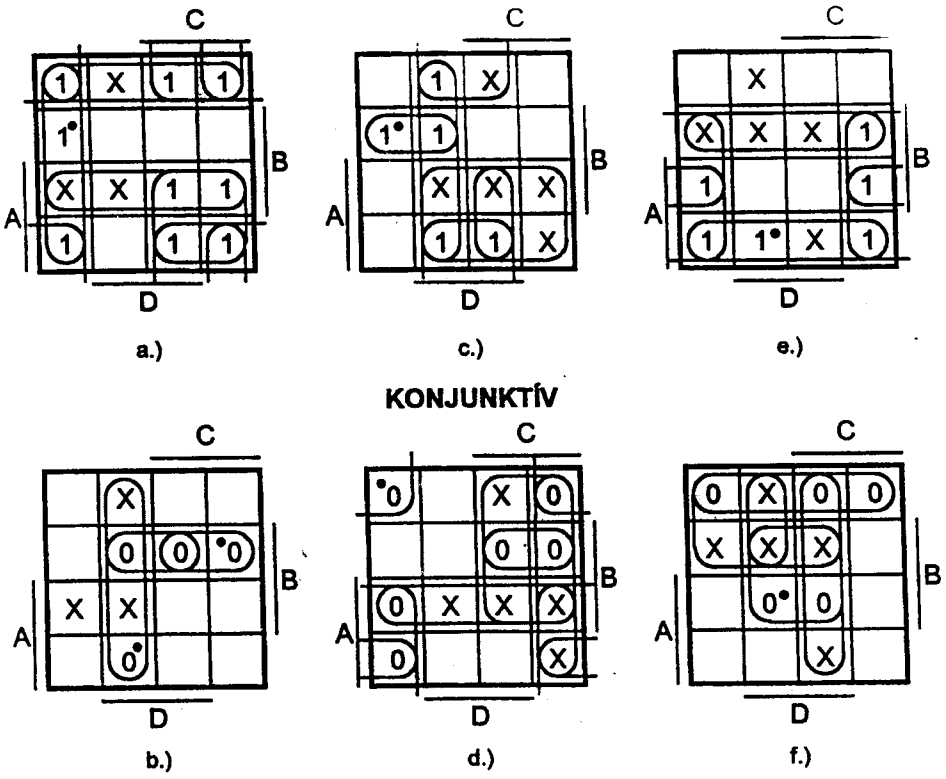
Megoldás:

a) **Diszjunktív alak**

1. Ábrázoljuk a függvényt KV táblán (3.35.a) ábra).
2. Végezzük el a tömbösítést (5 db 4-es tömb van).
3. Jelöljük meg ponttal azon mintermeket, melyeken csak egy hurok megy keresztül (m_4).

Nélkülözhetetlen primimplikáns: $\overline{C} \wedge \overline{D}$.
A nélkülözhetetlen pi által megvalósított mintermek: m_0, m_4, m_8.
A fennmaradó mintermek lehetséges lefedései:

$$\overline{B}C \vee AC \text{ ill. } \overline{B}C \vee AB \text{ ill. } \overline{A}\,\overline{B} \vee AC.$$



3.35. ábra

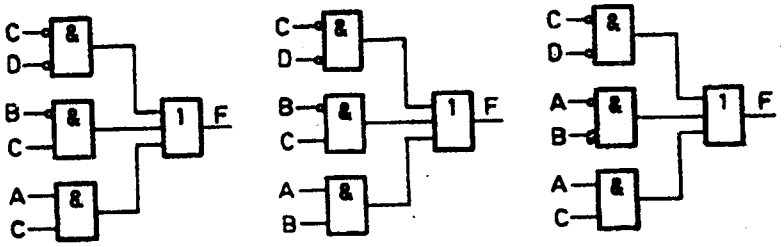
Tehát az ekvivalens diszkjunktív minimál alakok:

$$F = \overline{C}\,\overline{D} \vee \overline{B}C \vee AC$$

$$F = \overline{C}\,\overline{D} \vee \overline{B}C \vee AB$$

$$F = \overline{C}\,\overline{D} \vee \overline{A}\,\overline{B} \vee AC.$$

A függvényeket realizáló hálózatok MSz jelképe a 3.36. ábrán látható.



3.36. ábra

b) **Konjunktív alak** 3.35.b) ábra.

A konjunktív minimál alak előállításánál a függvény 0 helyeiből is kiindulhatunk. Ilyenkor a 0-ákat tömbösítjük a diszjunktív alaknál leírt szabályok szerint. **A tömbök leolvasásánál viszont mindig az oldalt feltüntetett változó negált értékét** olvassuk le, s ezekből **logikai összegeket** képezünk.

Nélkülözhetetlen primimplikánsok:

$$\left(C \vee \overline{D}\right) - \text{ az } m_9 \text{ miatt}$$

és

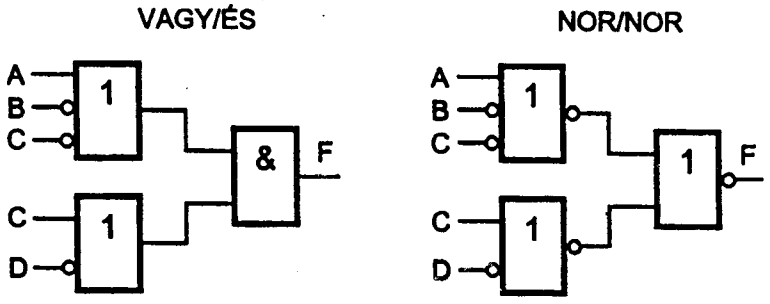
$$\left(A \vee \overline{B} \vee \overline{C}\right) - \text{ az } m_6 \text{ miatt}.$$

Tehát: $F = \left(C \vee \overline{D}\right)\left(A \vee \overline{B} \vee \overline{C}\right)$.

A NOR/NOR alakbani megoldáshoz induljunk ki a függvény **kétszer tagadott** alakjából, majd alkalmazzuk a De Morgan szabályt az alsó tagadásjel felbontására.

$$F = \overline{\overline{\left(C \vee \overline{D}\right)\left(A \vee \overline{B} \vee \overline{C}\right)}} = \overline{\overline{\left(C \vee \overline{D}\right)} \vee \overline{\left(A \vee \overline{B} \vee \overline{C}\right)}}.$$

A VAGY/ÉS, illetve NOR/NOR hálózat a 3.37. ábrán látható.



3.37. ábra

2. Példa:

Egyszerűsítsük az

$$F(A,B,C,D) = \Sigma(1,4,5,9,11) + \Sigma_X(3,10,13,14,15)$$

függvényt diszkjunktív és konjunktív alakban.

Megoldás:

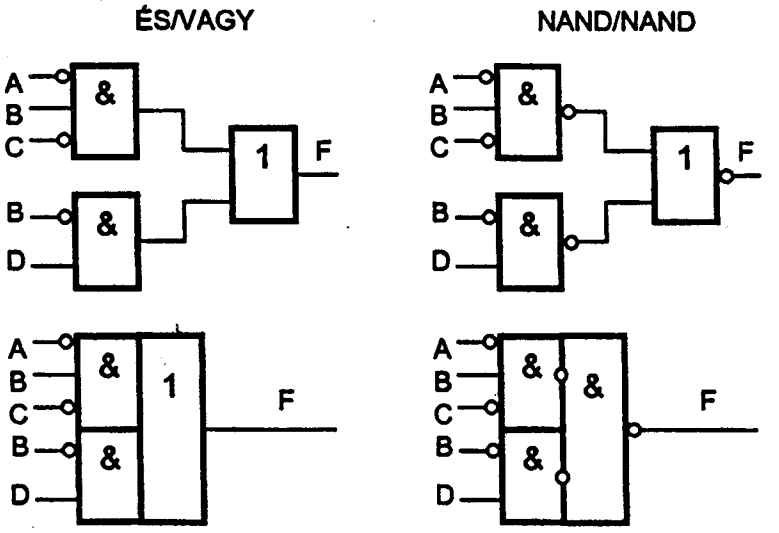
a) **Diszkjunktív alak** 3.35.c) ábra.

Képezhető tömbök száma: 1 db kettes, 4 db négyes.

Nélkülözhetetlen pi: - az m_4 miatt: $\overline{A}B\overline{C}$.

A fennmaradó mintermek: m_1, m_9, m_{11}.

A függvény diszjunktív minimál alakja: $F = \overline{A}B\overline{C} \vee \overline{B}D = \overline{\overline{A}B\overline{C} \wedge \overline{B}D}$.



3.38. ábra

A diszjunktív függvényt realizáló hálózat MSz jelképes rajzát a 3.38. ábrán adtuk meg.

b) **Konjunktív alak** 3.35.d) ábra.

Képezhető tömbök száma: 6 db négyes.

Nélkülözhetetlen pi: $(B \vee D)$ - az m_0 miatt.

Ekvivalens megoldások:

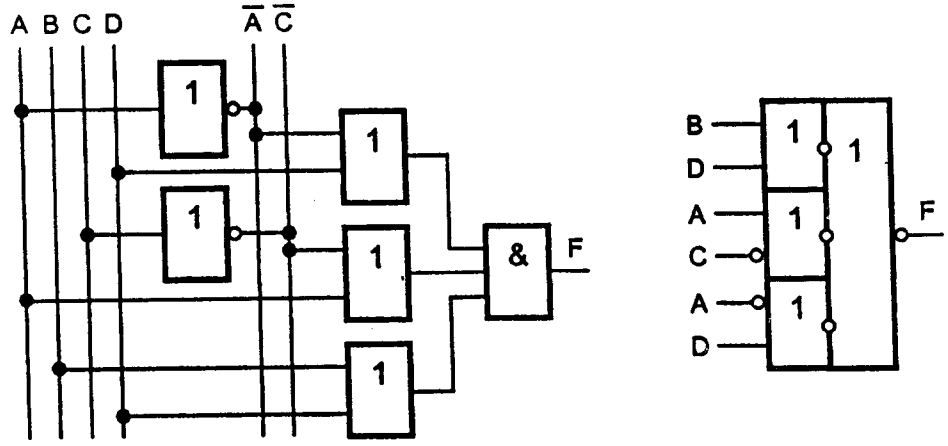
$$F = (B \vee D)(A \vee \overline{C})(\overline{A} \vee \overline{B}) = \overline{\overline{(B \vee D)} \vee \overline{(A \vee \overline{C})} \vee \overline{(\overline{A} \vee \overline{B})}}$$

$$F = (B \vee D)(A \vee \overline{C})(\overline{A} \vee D) = \overline{\overline{(B \vee D)} \vee \overline{(\quad \overline{C})} \vee \overline{(\overline{A} \vee D)}}$$

$$F = (B \vee D)(\overline{B} \vee \overline{C})(\overline{A} \vee \overline{B}) = \overline{\overline{(B} \quad \overline{C})} \vee \overline{(\overline{A} \vee \overline{B})}$$

$$F = (B \vee D)(\overline{B} \vee \overline{C})(\overline{A} \vee \overline{\quad}) \quad , \vee \overline{(\overline{B} \vee \overline{C})} \vee \overline{(\overline{A} \vee \overline{D})}.$$

A konjunktív függvényt realizáló hálózat Msz jelképes rajzát a 3.39. ábrán adtuk meg.



3.39. ábra

3. Példa:

Egyszerűsítsük az

$$F(A,B,C,D) = \Sigma(6,8,9,10,12,14) + \Sigma_X(1,4,5,7,11)$$

függvényt diszkjunktív és konjunktív alakban.

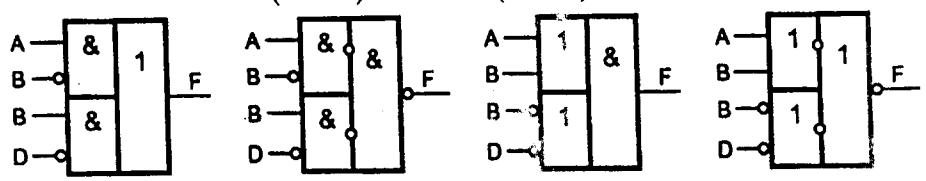
Megoldás:

a) **Diszkjunktív alak**: 3.35.e) ábra.

$$F = A\overline{B} \vee B\overline{D} = \overline{\overline{A\overline{B}} \wedge \overline{B\overline{D}}}.$$

b) **Konjunktív alak**: 3.35.f) ábra.

$$F = (\overline{B} \vee \overline{D})(A \vee B) = \overline{\overline{(\overline{B} \vee \overline{D})} \vee \overline{(A \vee B)}}.$$



3.40. ábra

A függvényt realizáló hálózatot a 3.40. ábrán rajzoltuk meg MSz jelképekkel.

4. Példa:

Minimalizáljuk az

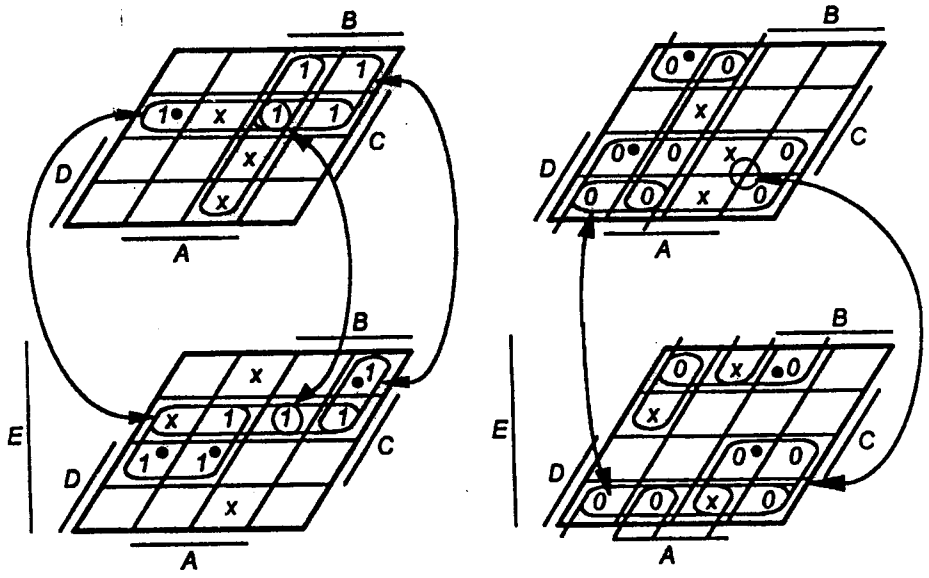
$$F(E,D,C,B,A) = \Sigma(2,3,4,6,7,18,21,22,23.28,29) + \Sigma_X(5,11,15,17,20,27)$$

függvényt diszkjunktív és konjunktív alakban.

Megoldás:

a) **Diszkjunktív alak**:

1. Ábrázoljuk a függvényt térbeli ötváltozós KV táblán.
2. Képezzük a lehető legnagyobb tömböket előbb a síkbeli táblákon, majd a táblák között.
3. Jelöljük meg ponttal a nélkülözhetetlen pi-ket. (3.41. ábra).



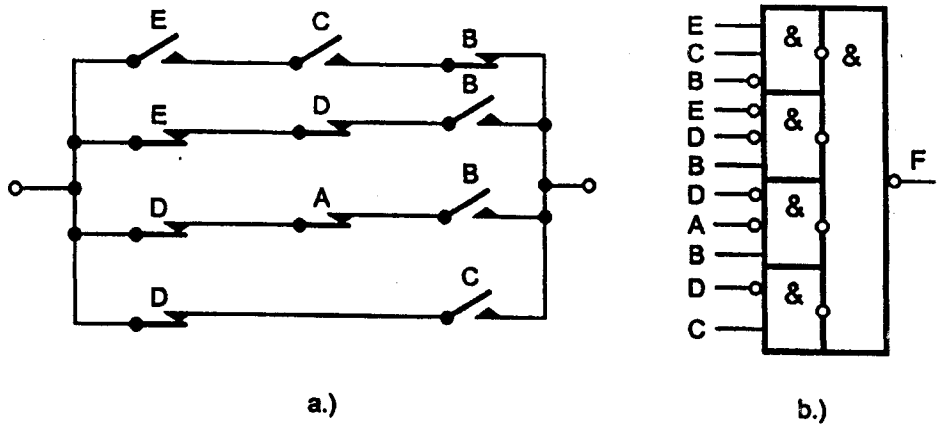
3.41. ábra

A függvény:

$$F = \dot{E}C\overline{B} \vee \overline{D}C \vee \overline{D}B\overline{A} \vee \overline{E}\,\overline{D}\,B = \overline{\overline{ECB} \wedge \overline{DC} \wedge \overline{DBA} \wedge \overline{E}\,\overline{DB}}$$

$$F = EC\overline{B} \vee \overline{D}C \vee \overline{D}B\overline{A} \vee \overline{E}BA = \overline{\overline{ECB} \wedge \overline{DC} \wedge \overline{DBA} \wedge \overline{E}BA}.$$

Az érintkezős ÉS/VAGY, illetve NÉS/NÉS alakbani realizálását a 3.42. ábra szemlélteti.

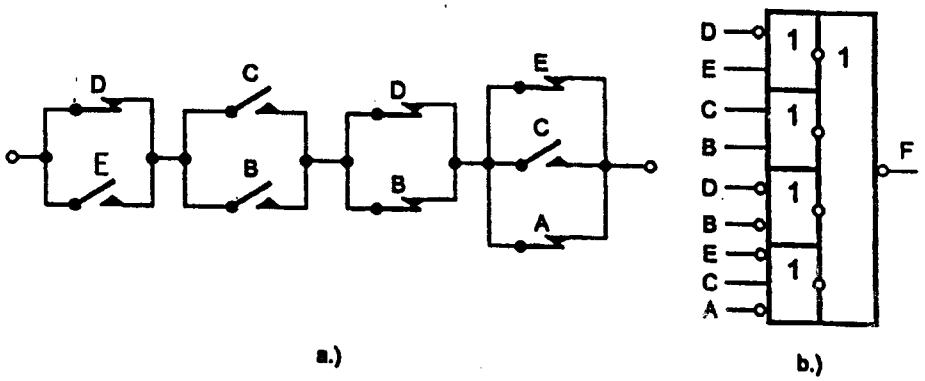


3.42. ábra

b) Konjunktív alak:

$$F = (E \vee \overline{D})(C \vee B)(\overline{D} \vee \overline{B})(\overline{E} \vee C \vee \overline{A}) = \overline{\overline{(E \vee \overline{D})} \vee \overline{(C \vee B)} \vee \overline{(\overline{D} \vee \overline{B})} \vee \overline{(\overline{E} \vee C \vee \overline{A})}}.$$

Az érintkezős, illetve NOR/NOR alakbani realizációt a 3.43.ábrán találjuk.



3.43. ábra

3.4.3. Numerikus minimalizálás

Láttuk, hogy a grafikus eljárás max. 6 változóig használható. Szükség van tehát egy olyan algoritmusra, amely a **változók számától** függetlenül alkalmazható.

A **Quine - Mc Cluskey-féle numerikus** egyszerűsítési módszer olyan szisztematikus eljárás, amely tetszőleges számú független változó esetén alkalmazható.

A diszkjunktív normál alakban adott függvény egyszerűsítésének lépései:

1) A függvényben szereplő mintermek bináris kódjainak felírása és a **mintermek csoportokba** sorolása a kódjaikban előforduló **1-esek száma szerint**, az egyesek száma szerint növekvő sorrendben. A csoporton belül a mintermek nagysága szerint **növekvő** sorrendben következnek.

5. Példa:

$$F(X_1, X_2, \ldots . X_6) = \Sigma(0,2,6,7,8,10,12,14,15,41).$$

A) Két **csoport szomszédos**, ha a két csoporthoz tartozó 1-esek számának különbsége 1. Szomszédos mintermek ezért csak szomszédos csoportokban lehetnek, ami az egyszerűsítést megkönnyíti.

B) Ha két minterm szomszédos, akkor a decimális megfelelőiknek különbsége 2-nek nem negatív egész kitevőjű hatványa. Fordítva, ha két szomszédos csoportból való minterm decimális megfelelőinek különbsége 2 nem negatív egész kitevőjű hatványa, és a több egyest tartalmazó csoportból való minterm decimális megfelelője a nagyobbik, akkor a két minterm szomszédos.

A minterm bináris kódja		1-esek száma	Csoportosítás
0	000000	0	0
2	000010	1	2
6	000110	2	8
7	000111	3	6
8	001000	1	10
10	001010	2	12
12	001100	2	7
14	001110	3	14
15	001111	4	41
41	101001	3	15

2) Az összehasonlítást a **legelső** elemmel kezdjük. Ezt csak a rákövetkező **szomszédos** csoport elemeivel kell összehasonlítani. Ha találunk olyan számpárt, amelyek B)-nek megfelelő értelemben összevonhatók, akkor mindkettőt oldalt + jellel megjelöljük, és a számpár elemeit növekvő sorrendben egy **új oszlopban** egymás mellé írjuk, majd zárójelben megjelöljük a különbségüket is. Ez jelöli ugyanis az **elhagyható változó**

helyiértékét. Az egyszerűsítést az első és második csoport elemeinek összehasonlításával kezdjük. Ezt követően elvégezzük az összehasonlítást az első oszlop második és harmadik csoportjának elemei között. A keletkező számpárok a második oszlop második csoportját fogják alkotni stb.

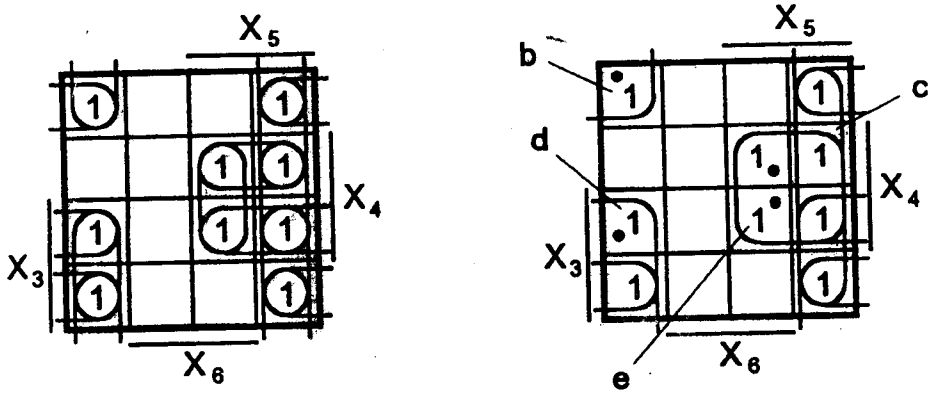
Az összehasonlítást az első oszlop összes szomszédos csoportja között elvégezzük.

I.oszlop	II.oszlop	
0 +	0,2	(2)
2 +	0,8	(8)
8 +	2,6	(4)
6 +	2,10	(8)
10 +	8,10	(2)
12 +	8,12	(4)
7 +	6,7	(1)
14 +	6,14	(8)
41	10,14	(4)
15 +	12,14	(2)
	7,15	(8)
	14,15	(1)

3) A második oszlopból a harmadik oszlopot a 2-ben leírt módon képezzük, de az összevonás feltétele az, hogy a **zárójelben lévő összes szám megegyezzen** (ugyanazon változók hiányozzanak mindkét csoportból) és az **első decimális számok különbsége 2 pozitív egész kitevőjű hatványa** legyen, és a **hátrább álló csoportból való decimális szám legyen a nagyobbik**. Ha nem végezhető el az összevonás, akkor az egyes oszlopokban + jellel nem megjelölt szorzatok a primimplikánsok.

I.oszlop	II.oszlop		III.oszlop
0	0,2	(2)	0, 2, 8, 10 (2,8) b
2	0,8	(8)	2, 6, 10, 14 (4,8) c
8	2,6	(4)	8,10, 12, 14 (2,4) d
6	2,10	(8)	6, 7, 14, 15 (1,8) e
10	8,10	(2)	
12	8,12	(4)	
7	6,7	(1)	
14	6,14	(8)	
41...a	10,14	(4)	
15	12,14	(2)	
	7,15	(8)	
	14,15	(1)	

Az összehasonlításokat szemlélteti a 3.44. ábra.



3.44. ábra

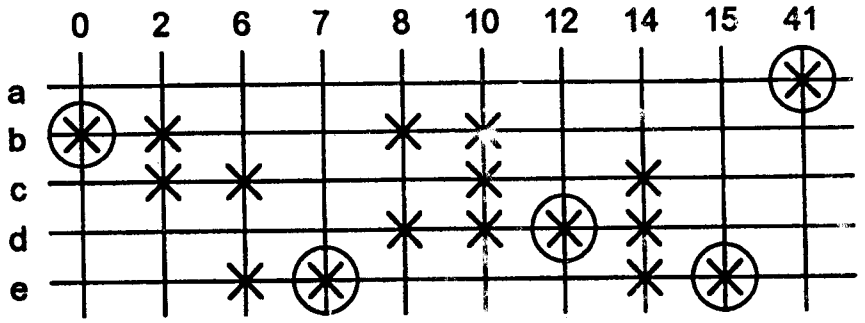
Figyeljük meg, hogy az első összehasonlítás a 2-es tömböket eredményezi, a második összehasonlítás a 4-es tömböket, stb. Mivel ua. 4-es tömb ugyanazon mintermekből kétféleképpen képezhető, az egyiket elhagyjuk (Pl.: 0, 2, 8, 10, illetve 0, 8 , 2 10).

Az eddigiekben végzett művelet a grafikus eljárásnál a **tömbösítésnek** felel meg.

4) A szükséges primimplikánsok kiválasztása a **primimplikáns táblázattal** történik. A táblázat sorainak száma megegyezik a primimplikánsok számával. A táblázat oszlopainak száma megegyezik a függvényt alkotó mintermek számával. Minden primimplikáns sorában megjelöljük azon mintermek oszlopait, melyeket az illető primimplikáns tartalmaz.

   A függvényt alkotó primimplikánsokat úgy kell kiválasztani, hogy együttesen valamennyi mintermeket tartalmazzák. E célból karikázzuk be azon jeleket, melyek **egyedül állnak** oszlopunkban (3.45. ábra). Azon primimplikánsokat, melyek sorában bekarikázott jel van, nélkülözhetetlen primimplikánsok, mert olyan mintermeket is tartalmaznak, melyek más primimplikánsban nem fordulnak elő. A **nélkülözhetetlen primimplikánsoknak** a függvényben feltétlenül szerepelniük kell. Példánkban: a, b, d, e.
   Ezután '+' jellel jelöljük meg a táblázat fejlécén azon mintermeket, melyeket a nélkülözhetetlen primiplikánsok tartalmaznak (példánkban valamennyit). Amennyiben maradnak olyan mintermek, amelyeket a függvény tartalmaz, de a lényeges primiplikánsok nem tartalmaznak, úgy azokat célszerűen kiválasztott primiplikánsokkal kell megvalósítani. Az egyszerűsített példabeli függvény:

$$F = a \vee b \vee d \vee e = X_1 \overline{X_2} X_3 \overline{X_4} \overline{X_5} X_6 \vee \overline{X_1} X_2 \overline{X_4} \overline{X_6} \vee \overline{X_1} \overline{X_2} X_3 X_6 \vee X_1 X_2 X_4 X_5$$



3.45. ábra

A leírt numerikus módszer a Quine módszernek ábrázolási módjában egyszerűsített változata. A Quine eljárásnál a mintermeket **bináris**, a numerikus eljárásnál pedig **decimális** számként kezeljük. A Quine módszernél

a "kiesett változót "-" jellel jelöljük. A két módszert összehasonlítva mutatjuk be az

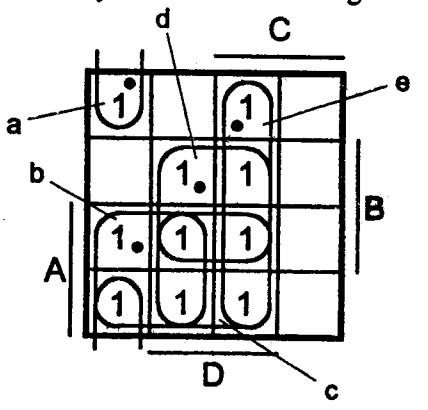
$$F^4(A,B,C,D) = \Sigma(0,3,5,7,8,9,11,12,13,15).$$

függvény egyszerűsítése kapcsán.

a) Quine módszer:

I.oszlop		II.oszlop		III.oszlop	
	ABCD		ABCD		ABCD
0	0000 +	0,8	-000..a	8,9,12,13	1-0-...b
8	1000 +	8,9	100- +	3,7,11,15	--11...c
3	0011 +	8,12	1-00 +	5,7,13,15	-1-1...d
5	0101 +	3,7	0-11 +	9,11,13,15	1--1...e
9	1001 +	3,11	-011 +		
12	1100 +	5,7	01-1 +		
7	0111 +	5,13	-101 +		
11	1011 +	9,11	10-1 +		
13	1101 +	9,13	1-01 +		
15	1111 +	12,13	110- +		
		7,15	-111 +		
		11,15	1-11 +		
		1315	11-1 +		

Hasonlítsuk össze az eredményt 3.46. ábra szerinti grafikus megoldással.



3.46. ábra

b) Quine-Mc Cluskey módszer:

I.oszlop	II.oszlop		III.oszlop	
0	0,8	(8)...a	8,9,12,13	(1,4)...b
8	8,9	(1) +	3,7,11,15	(4,8)...c
3	8,12	(4) +	5,7,13,15	(2,8)...d
5	3,7	(4) +	9,11,13,15	(2,4)...e
9	3,11	(8) +		
12	5,7	(2) +		
7	5,13	(8) +		
11	9,11	(2) +		
13	9,13	(4) +		
15	12,13	(1) +		
	7,15	(8) +		
	11,15	(4) +		
	13,15	(2) +		

Írjuk fel a primimplikánsokat:

a - $\overline{B}\,\overline{C}\,\overline{D}$ (a 8-as értékű változót hagyjuk el): -000)

b - $A\,\overline{C}$ (az 1-es, 4-es értékű változókat hagyjuk el): 1-0-)

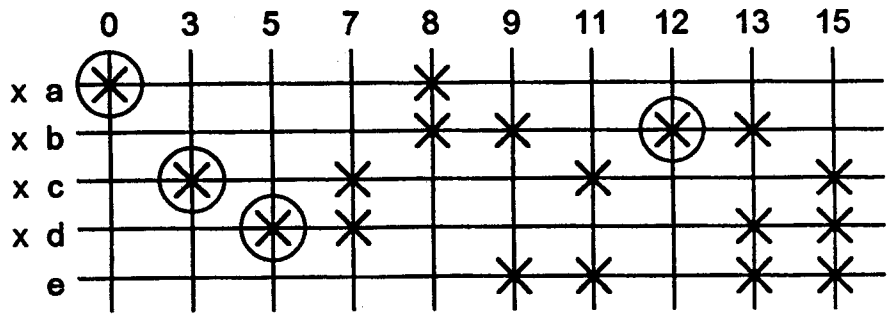
c - CD (a 4-es, 8-as értékű változókat hagyjuk el): --11)

d - BD (a 2-es, 8-as változókat hagyjuk el: -1-1)

e - AD (a 2-es, 4-es változókat hagyjuk el: 1--1).

A logikai szorzatok visszaállításánál numerikus módszernél a **zárójelben** lévő változókat, Quine módszernél pedig a "-" jellel jelölt változókat hagyjuk el. A **megmaradó 0-ához a változó negáit, az 1-hez pedig a változó ponált** értékét írjuk.

A primimplikáns táblázat a 3.47. ábrán látható.



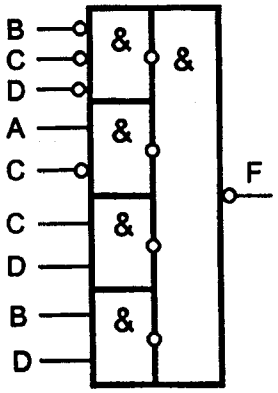
3.47. ábra

Lényeges primimplikánsok:

$$F = a \vee b \vee c \vee d$$

$$F = \overline{B}\,\overline{C}\,\overline{D} \vee A\,\overline{C} \vee CD \vee BD.$$

A függvényt grafikus módszerrel a 3.46. ábrán egyszerűsítettük. A grafikus eljárásnál azon tömbök lesznek a nélkülözhetetlen primimplikánsok, amelyek - valamennyi lehetséges tömb berajzolása esetén - olyan mintermeket tartalmaznak amelyeken csak egy hurok megy keresztül. Logikai sémája a 3.48. ábra szerinti.



3.48. ábra

Megjegyzések:

1) Ha a logikai függvény nem teljesen specifikált, akkor a **közömbös mintermeket** is felvesszük a **kiindulási oszlopba** (ezáltal nő az összevonás lehetősége), de a primimplikánsokkal csak azokat a mintermeket fedjük le, amelyekre a logikai függvény 1 értékére van specializálva. Tehát a primimplikáns táblázat fejlécén a **közömbös mintermeket nem tüntetjük fel.**

2) A numerikus módszer minden változtatás nélkül alkalmazható, ha a függvény konjunktív normál alakban adott, mivel a Boole algebrában

$$(A \vee B)(A \vee \overline{B}) = A.$$

3) A kapcsolási függvények egyszerűsítésénél gyakran **ekvivalens megoldások** is adódnak. Ilyenkor célszerű valamennyi megoldást feltüntetni. Ezek a **pótlólagos** egyszerűsítéseknél előnyösen felhasználhatók.

4) Gyakran van szükség azonos bemeneti változóktól függő **többkimenetű** hálózat tervezésére. Az m kimenetű hálózat működése m db. logikai függvényből álló függvényrendszerrel jellemezhető. A hálózat tervezésének egyik lehetősége, hogy **valamennyi logikai függvényt külön-külön** minimalizáljuk. Egyszerűbb eredményre vezethet a többkimenetű hálózat **közös implikánsainak** megkeresése.

5) Olyan esettel gyakran találkozunk, amikor a primimplikánsok kiválasztása nem végezhető el ilyen egyszerűen, ugyanis valamennyi 1-gyel jelölt mintermen legalább **két hurok** megy keresztül. Ilyen esetben a kiválasztást a segédfüggvény felírásával végezhetjük el. A **segédfüggvény** alkalmazását két példa kapcsán mutatjuk be.

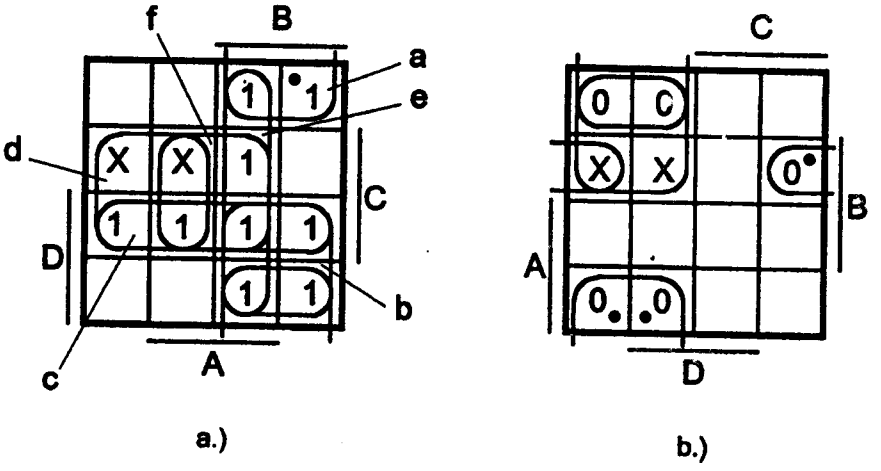
1. Példa:

Minimalizáljuk az

$$F(D,C,B,A) = \Sigma(2,3,7,10,11,12,13,14,15) + \Sigma_X(4,5).$$

függvényt diszjunktív és konjunktív alakban. A tömbösítés a 3.49. ábrán látható.

Nélkülözhetetlen primiplikáns: a = $\overline{C}B$. A nélkülözhetetlen pi által megvalósított mintermek m_2, m_3, m_{10}, m_{11}.

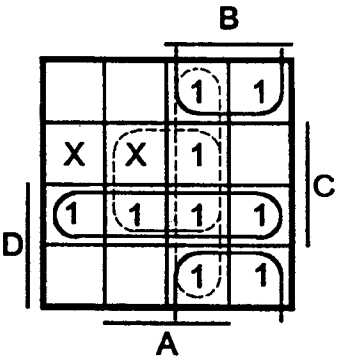


3.49. ábra

Fennmaradó egyessel jelölt mintermek: m_7, m_{12}, m_{13}, m_{14}, m_{15}, melyeken minimum két hurok megy keresztül. Az optimális megoldás kiválasztásához készítsük el a primiplikáns táblázatot (3.50. ábra).

p_i \ m_i	7	12	13	14	15
b (10 11 14 15)				●	●
c (12 13 14 15)		●	●	●	●
d (4 5 12 13)		●	●		
e (3 7 11 15)	●				●
f (5 7 13 15)	●		●		●

a.)



b.)

3.50 ábra

Írjuk fel egyenként, hogy a fennmaradó mintermeket, mely primiplikánsokkal valósíthatjuk meg.

$$m_7 - (e \vee f)$$

$$m_{12} - (c \vee d)$$

$$m_{13} - (c \vee d \vee f)$$

$$m_{14} - (b \vee c)$$

$$m_{15} - (b \vee c \vee e \vee f).$$

A végső megoldásban tehát szerepelni kell az (e ∨ f) ÉS (c ∨ d) ÉS (c ∨ d ∨ f) ÉS (b ∨ c) ÉS (b ∨ c ∨ e ∨ f) jelű primimplikánsoknak.
A segédfüggvény (G) tehát:

$$G = (e \vee f) \wedge (c \vee d) \wedge (c \vee d \vee f) \wedge (b \vee c) \wedge (b \vee c \vee e \vee f).$$

Végezzük el a kijelölt műveletet:

$$G = (ecv\ fcv\ edv\ fd)(cv\ dv\ f) \wedge (bv\ c)(bv\ cv\ ev\ f)$$

$$G = (ecv\ fcv\ ecdv\ fcdv\ ecdv\ fcdv\ edv\ fdv\ ecfv\ fcv\ edfv\ fd)$$

$$G = (ecv\ fcv\ edv\ fd)(bv\ c)(bv\ cv\ ev\ f)$$

$$G = (ecbv\ fcbv\ edbv\ fdbv\ ecv\ fcv\ edcv\ fdc)$$

$$G = (ecv\ fcv\ edbv\ fdb)(bv\ cv\ ev\ f)$$

$$G = (ecbv\ fbcv\ edbv\ fdbv\ \underline{ec} \vee \underline{fc} \vee edbcv\ fdbcv....$$

$$... \quad ecv\ efcv\ edbv\ efdbv\ ecfv\ fcv\ edfbv\ fdb).$$

Figyeljük meg, hogy a végeredményül kapott primimplikáns csoportok mind egy-egy megoldást adnak. Pl. az e, c, b primimplikánsokkal az m_7, m_{12}, m_{13}, m_{14}, m_{15} mintermek lefedhetők. Mely pi csoportot válasszuk? Nyilvánvalóan azt a csoportot kell választani, amelyben a **legkevesebb számú és legkevesebb változót tartalmazó** (legnagyobb tömb) primimplikánsok vannak.

A választás feltétele tehát:

- a legkevesebb betűt tartalmazó szorzat (példánkban: ec ill. fc).
- kevesebb változót tartalmazó primimplikáns (példánkban ekvivalens, mert mind az e, mind az f egyaránt négyes tömbök).

A függvény diszkjunktív alakja tehát:

$$F = av\ cv\ e = B\overline{C} \vee CDv\ AB = \overline{\overline{B\overline{C}} \wedge \overline{CD} \wedge \overline{AB}}$$

$$F = av\ cv\ f = B\overline{C} \vee CDv\ AC = \overline{\overline{B\overline{C}} \wedge \overline{CD} \wedge \overline{AC}}.$$

A választott tömbösítést mutatja a 3.50.b) ábra alapján:

A konjunktív alak a 3.49.b) ábra:

$$F = (Cv\ B) \wedge (Dv\ \overline{C} \vee A) = \overline{\overline{(Cv\ B)} \vee \overline{(Dv\ \overline{C} \vee A)}}.$$

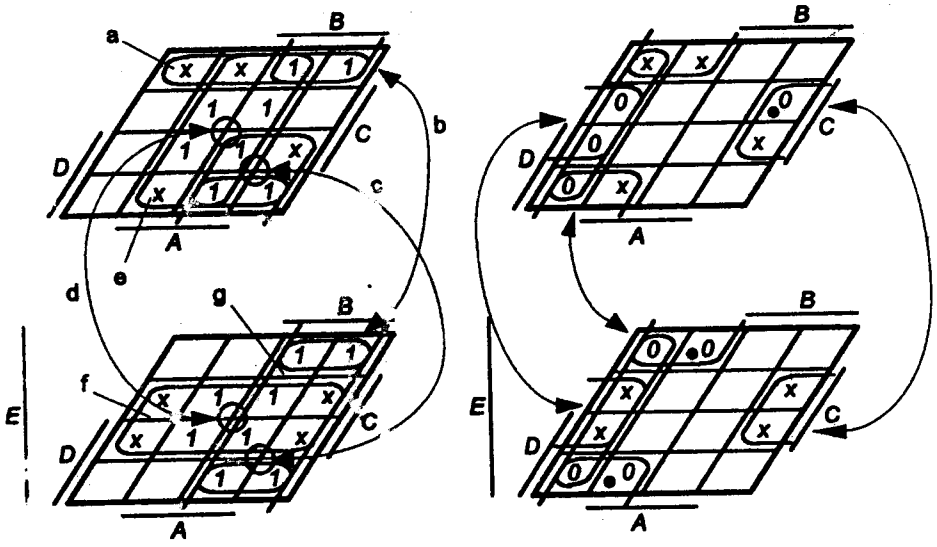
2. Példa:

Egyszerűsítsük az

$$F(E,D,C,B,A) = \Sigma(2,3,5,7,10,11,12,13,15,18,19,21,2[illegible]26,27,29,31) + \Sigma_x(0,1,9,14,20,22,28,30)$$

függvényt diszjunktív és konjunktív alakban.

A tömbösítés a 3.51. ábrán látható. Nélkülözhetetlen primimplikáns nincs, valamennyi mintermen egynél több hurok megy át. Írjuk fel a primimplikáns táblázat (3.52. ábra) alapján a segédfüggvényt.



3.51. ábra

A műveleteket elvégezve optimális megoldásul a bd primimplikáns csoportot kapjuk.

Tehát a függvény:

$$F = b \vee d = \overline{C}B \vee AC = \overline{\overline{\overline{C}B} \wedge \overline{AC}}.$$

A konjunktív alak:

$$F = (C \vee A)(C \vee B) = \overline{\overline{(C \vee A)} \vee \overline{(C \vee B)}}.$$

p_i \ m_i	2	3	5	7	10	11	13	15	18	19	21	23	26	27	29	31
a	●	●														
b	●	●			●	●			●	●			●	●		
c					●	●		●					●	●		●
d			●	●			●	●			●	●			●	●
e		●	●	●		●	●	●								
f											●	●			●	●
g									●	●		●	●	●		●

3.52. ábra

3.5. Szimmetrikus függvények

Valamely F(X1,X2,...Xn) logikai függvény szimmetrikus, ha a változók tetszőleges páronkénti felcserélésekor a függvény értéke változatlan.

Pl.: $F(X_1,X_2,X_3) = X_1 \wedge X_2 \wedge \overline{X_3} \vee X_1 \wedge \overline{X_2} \wedge X_3 \vee X_1 \wedge X_2 \wedge X_3$.

Egyszerűsítve:

$$F(X_1,X_2,X_3) = X_1 \wedge X_2 \vee X_1 \wedge X_3 \vee X_2 \wedge X_3 .$$

Jelölése:

$S^{\mathbf{n}}_{\mathbf{0,1...k}}$ ← változó számok
← szimmertira számok,

ahol $0 \leq k \leq n$.

A példabeli függvényre alkalmazva:

$$S^3_{2,3}(X_1,X_2,X_3).$$

A szimmetria számok értelmezésére figyeljük meg, hogy a függvény akkor 1 értékű, - ld. diszjunktív normál alak - ha bármely kettő, illetve ha három változó igenleges értékű.

A szimmetrikus függvényekre vonatkozó néhány tétel:

1) Két ugyanazon változókon értelmezett szimmetrikus függvény **összege** is szimmetrikus, az összegfüggvény szimmetria számai a komponens függvények szimmetria számainak egyesítése.

Pl.: $S^4_{1,3} \vee S^4_{2,3} = S^4_{1,2,3}$.

2) Két szimmetrikus függvény **szorzata** szimmetrikus függvény, ha van a komponens függvényeknek közös szimmetria száma.

Pl.: $S^5_{1,2,4} \wedge S^5_{2,3,4} = S^5_{2,4}$.

3) A szimmetrikus függvény **tagadottja** is szimmetrikus függvény, amelynek szimmetria számai az eredeti függvény szimmetria számainak **komplemens** halmazának elemei.

Pl.: $\overline{S^6_{1,2,3,6}} = S^6_{0,4,5}$.

4) **Kanonikus** az a szimmetrikus függvény, amelynek csak egy szimmetria száma van. Tetszőleges szimmetrikus függvény felépíthető kanonikus szimmetrikus függvényekből.

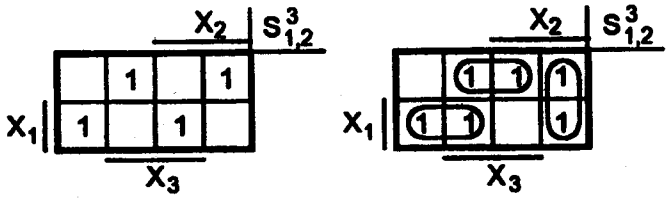
$$\text{Pl.: } S^4_{1,3,4} = S^4_1 \vee S^4_3 \vee S^4_4.$$

5) Minden szimmetrikus függvény azonos azzal a szimmetrikus függvénnyel, amelyben a ponált változókat negáltakkal - és fordítva - helyettesítjük, a szimmetria számokat pedig (n-k)-ra változtatjuk.

Pl.:

$$S^4_{2,3}(A,B,C,D) = S^4_{4-2}(\overline{A},\overline{B},\overline{C},\overline{D}) + S^4_{4-3}(\overline{A},\overline{B},\overline{C},\overline{D}) = S^4_{1,2}(\overline{A},\overline{B},\overline{C},\overline{D}).$$

Megjegyzés:
Kétszintű realizációban csak a **szomszédos szimmetria számú** szimmetrikus függvények vonhatók össze (3.53. ábra).
A szimmetrikus, illetve részben szimmetrikus függvények realizálásánál előnyösen használhatók az ANTI- ill. EKVIVALENCIA elemek.



3.53. ábra

Ezek az elemek szimmetrikus függvények:

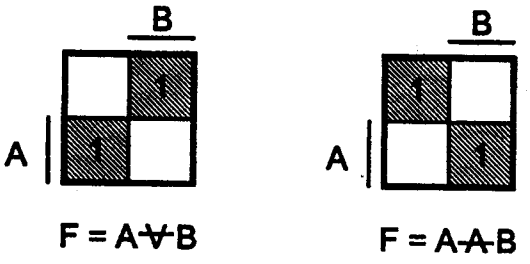
$$\text{Antivalencia: } F^2_6(A,B) = A \vee B = S^2_1(A,B)$$

$$\text{Ekvivalencia: } F^2_9(A,B) = A \wedge B = S^3_{0,2}(A,B).$$

Itt jegyezzük meg, hogy mivel az ANTIVALENCIA elemmel a NEM kapcsolat realizálható, így bármely függvény ÉS/VAGY/ANTIVALENCIA rendszerben megvalósítható.

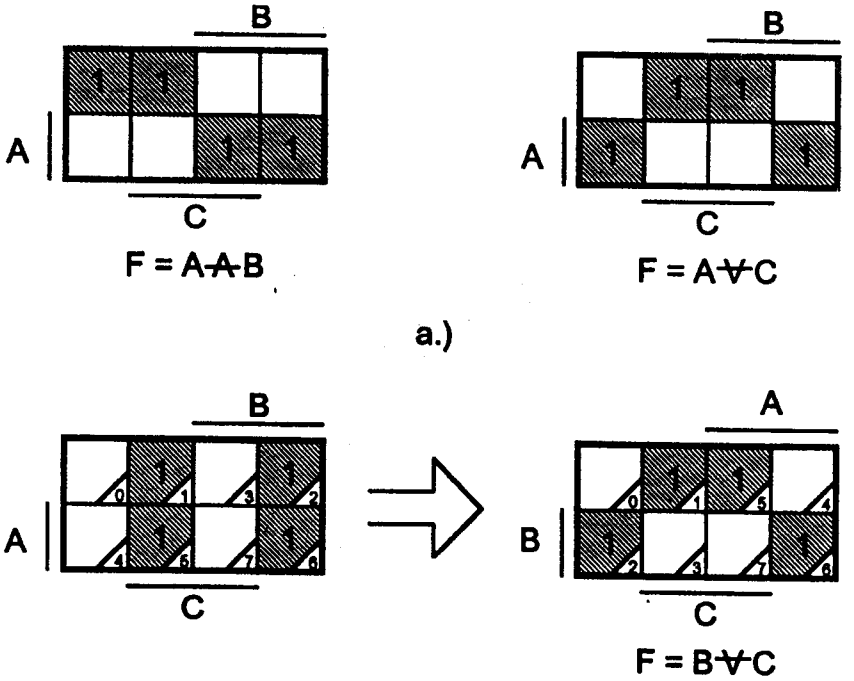
Kiindulásul tekintsük a fenti kapcsolatok KV tábláit (3.54. ábra).

Figyeljük meg, hogy a KV táblán jelölt mintermek a "csúcsokon szomszédosak".



3.54. ábra

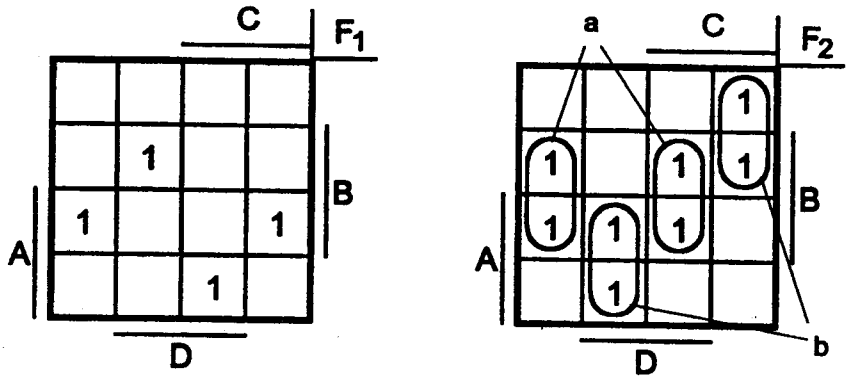
Ez a szomszédosság a hagyományos értelemben vett tömbökre is érvényes. Erre láthatunk példát a 3.55. ábrán. Az a, ábrán ez közvetlenül látható, a b, ábrán viszont csak átrajzolás után.



3.55. ábra

Természetesen az ÉS, illetve VAGY kapcsolatra vonatkozó eddigi szabályok (területek közös része, illetve összege) továbbra is érvényesek.

Erre láthatunk példát a 3.56. ábrán.

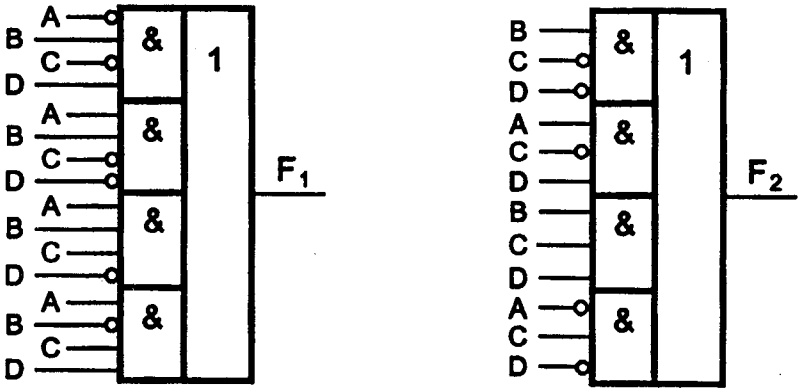


3.56. ábra

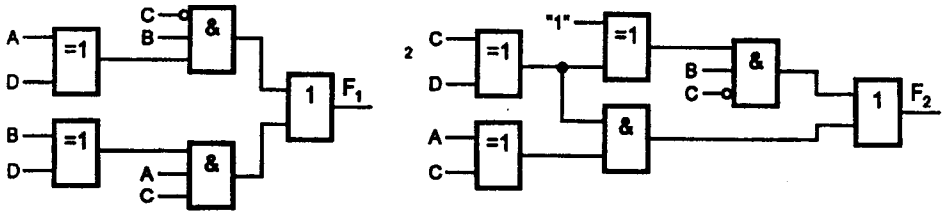
$$F_1 = B\bar{C}(A \forall D) \vee AC(B \forall D)$$

$$F_2 = a \vee b = B(CaD) \vee (A \forall C)(C \forall D).$$

A 3.57. ábrán a kétszintű ÉS/VAGY, A 3.58. ábrán a többszintű (3) ÉS/VAGY/ANTIVALENCIA kapcsolást is bemutattunk.



3.57. ábra



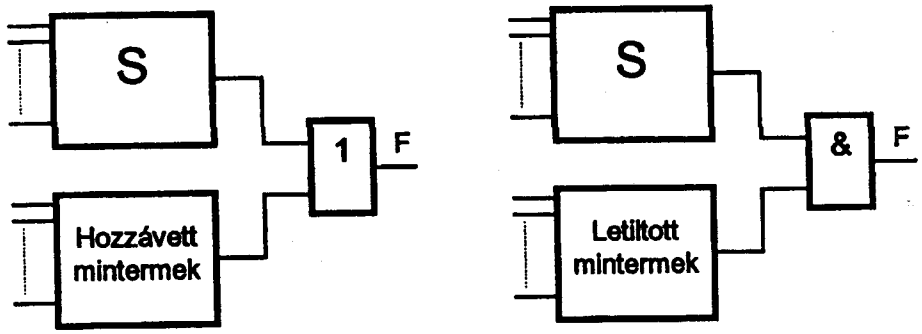
3.58. ábra

Az F_1 függvény esetében a szükséges kapuk száma:

Kapu	ÉS/VAGY	ÉS/VAGY/ANTIVALENCIA
Inverter	4	-
ÉS	4X4 be	2x3 be
VAGY	1x4 be	1x2 be
ANTIVALENCIA	-	2x2 be
Összes bemenetek száma	24	12

Figyeljük meg, hogy a VALENCIA kapcsolatba összevont tömböknél **változó nem esik ki**, mint a hagyományos minterm összevonásoknál.

Az ANTIVALENCIA elemekkel - mint láttuk - a függvény **szimmetrikus** részét célszerű realizálni. Az általános logikai függvényt ezek szerint kétféleképpen valósíthatjuk meg (3.59. ábra).



3.59. ábra

A 3.59. ábra szerint a részben szimmetrikus logikai függvényt mintermek hozzáadásával, vagy letiltásával tehetjük szimmetrikussá.

Az egyszerűsítés menete a következő:

a) Meghatározzuk a függvény hagyományos (ÉS, illetve VAGY), valamint a VALENCIA primimplikánsait.
b) A primimplikánsok kiválasztását valamennyi pi-re megvizsgáljuk és a legelőnyösebb megoldást választjuk.

Megjegyzés:

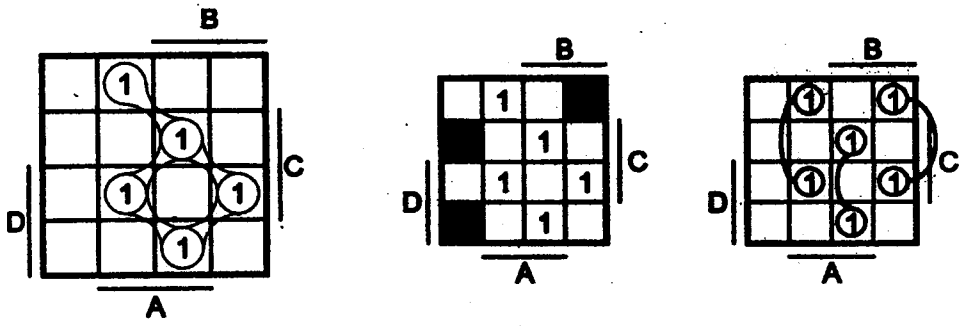
A fenti módszer csak részben tekinthető szisztematikusnak, de kellő gyakorlattal igen rövid idő alatt meghatározható az optimális megoldás a hasonló célú hálózatfelbontási eljárásokhoz viszonyítva, és ugyanakkor minimális matematikai apparátust igényel. A módszert egy további példa kapcsán mutatjuk be.

Példa:

Egyszerűsítsük és realizáljuk a következő függvényt:

$$F(D, C, B, A) = \sum(1,7,11,13,14).$$

Megoldás: (3.60. ábra).



3.60. ábra

a) Kétszintű NAND/NAND hálózat (nem tömbösíthető):

4 db INVERTER
5 db 4 bemenetű NAND
1 db 5 bemenetű NAND.

b) Bontsuk fel a függvényt az alábbiak szerint:

$$F(D,C,B,A) = S_3^4 \vee m_1^4$$

$$S_3^4 = \overline{D}CBA \vee D\overline{C}BA \vee DC\overline{B}A \vee DCB\overline{A}$$

$$S_3^4 = BA(C \forall D) \vee DC(A \forall B)$$

$$F = BA(C \forall D) \vee DC(A \forall B) \vee \overline{D}\,\overline{C}\,\overline{B}\,A.$$

c) Vonjuk össze az m_1^4 mintermet az m_7^4 -tel:

$$\overline{D}\,\overline{C}\,\overline{B}\,A \vee \overline{D}\,C\,B\,A = \overline{D}A(C \text{A} B)$$

$$S_3^4 = CB(D \forall A) \vee DA(C \forall B)$$

$$F = CB(D \forall A) \vee DA(C \forall B) \vee \overline{D}A(\overline{C \forall B}).$$

d) Adjuk hozzá a függvényhez az m_2^4, m_4^4, m_8^4 mintermeket, majd tiltsuk le ugyanazokat:

$$F = S_{1,3}^4 \wedge \overline{m_2^4} \wedge \overline{m_4^4} \wedge \overline{m_8^4}$$

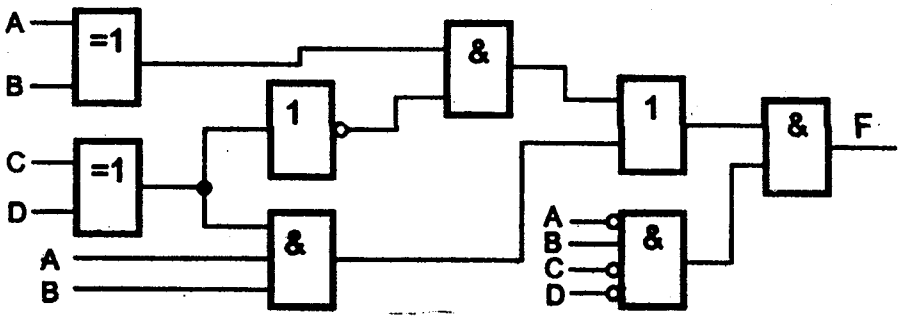
$$F = (A \vee B \vee C \vee D) \wedge \overline{m_2^4} \wedge \overline{m_4^4} \wedge \overline{m_8^4}.$$

e) Csak azt m_2^4 – t adjuk hozzá a függvényhez, majd tiltsuk le:

$$F = [A\bar{B}(C \wedge D) \vee \bar{A}B(C \wedge D) \vee AB(C \vee D)] \wedge \overline{m_2^4}$$

$$F = [A(C \vee B) \wedge B(C \wedge D) \vee AB(C \vee D)] \wedge \overline{m_2^4}.$$

Utóbbi megoldást rajzoltuk meg a 3.61. ábrán.

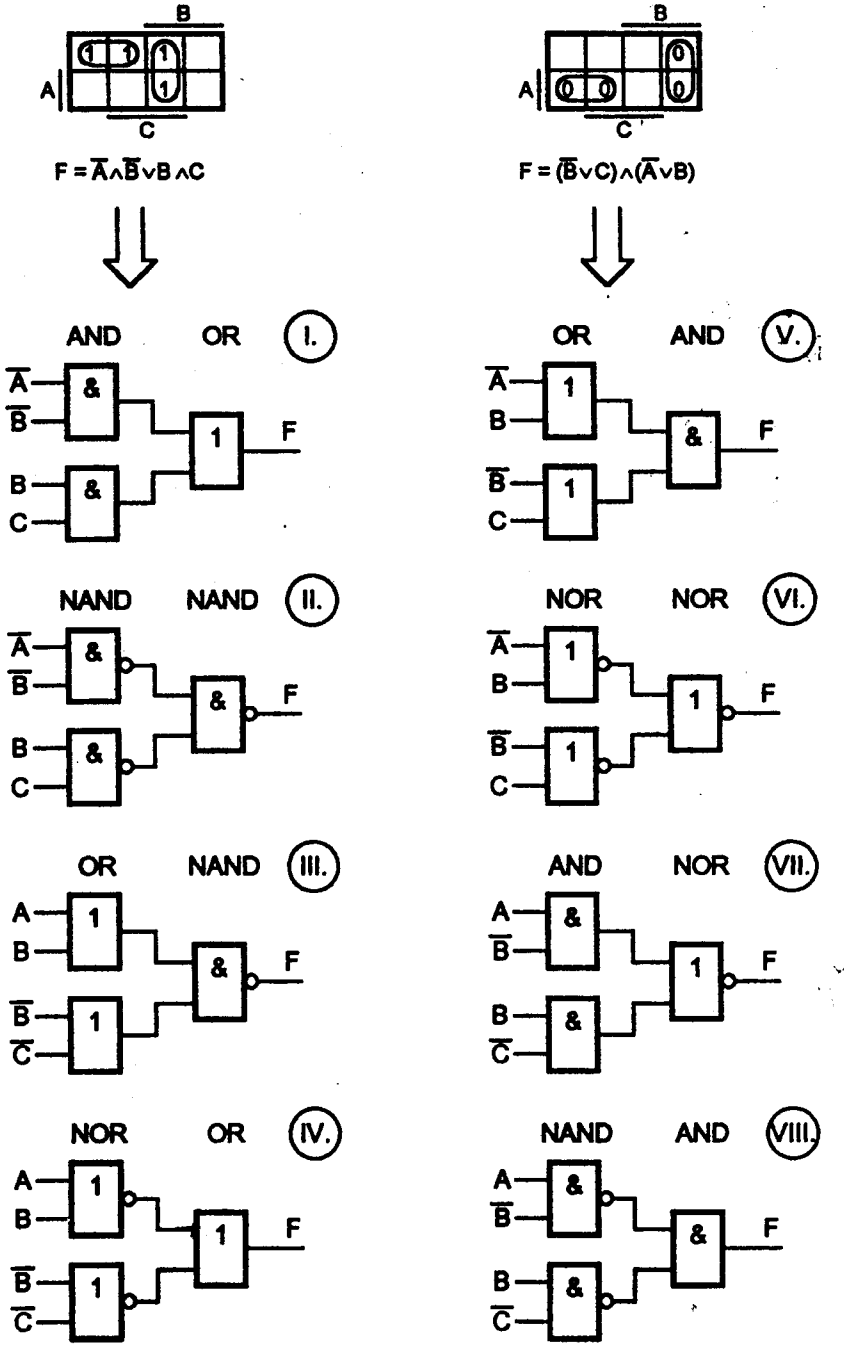


3.61. ábra

3.6. Logikai függvények realizálása

3.6.1. Kétfokozatú (kétszintű) hálózatok

Bármelyik logikai függvény 8 féle **kétfokozatú** alakban realizálható. Példaként az $F^3(A,B,C) = \Sigma(0,1,3,7)$ függvény 8 féle megvalósítását mutatjuk be a 3.62. ábrán.

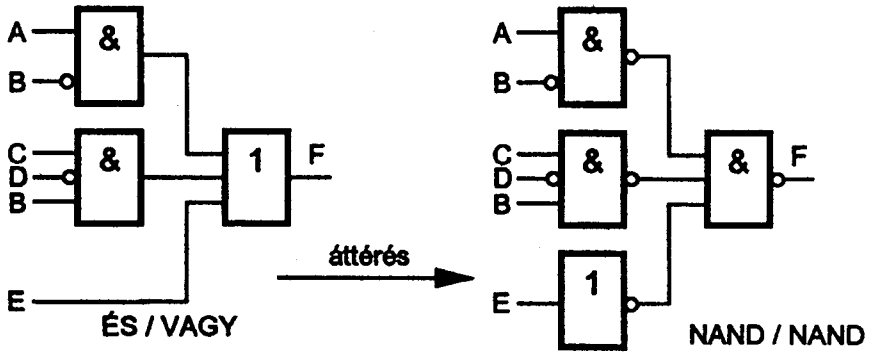

B
A
C
F = $\overline{A} \wedge \overline{B} \vee B \wedge C$
F = $(\overline{B} \vee C) \wedge (\overline{A} \vee B)$
AND OR I.
NAND NAND II.
OR NAND III.
NOR OR IV.
OR AND V.
NOR NOR VI.
AND NOR VII.
NAND AND VIII.
F

3.62. ábra

Az egyes alakzatokat a **diszjunktív**, illetve **konjunktív** alakból kiindulva kapjuk az $F = \overline{\overline{F}}$, és a De Morgan szabály háromszori ismételt alkalmazása esetén. Megjegyezzük, hogy a DTL rendszerbeni realizáció szempontjából az I. és V., a TTL rendszer szempontjából a II., VI., VII. alakzat a legfontosabb.

Az ábrából látható, hogy a **diszjunktív alakról** közvetlenül át lehet térni a **NAND elemes** realizációra. Áttéréskor minden **ÉS és VAGY kaput NAND kapuval** kell helyettesíteni, az összekötések nem változnak. Hasonlóképpen: a **konjunktív alakról** közvetlenül át lehet térni a **NOR elemes** realizációra. Áttéréskor minden **VAGY** és **ÉS kaput NOR kapuval** kell helyettesíteni, az **összekötések nem változnak.**

Felhívjuk a figyelmet, hogy az áttérésnél mindig célszerű alkalmazni a De Morgan szabályt. Ha ugyanis az egyszerűsítés **egyedül álló tagból** (diszjunktív alakban), vagy tényezőből (konjunktív alakban) áll, akkor a gépies átrajzolás hibás működést eredményez. Erre hívja fel a figyelmet a 3.63. ábra.



3.63. ábra

3.6.2. Többfokozatú (többszintű) hálózatok

Diódás kapuáramkörökből a nem kívánatos feszültségeltolódások miatt kettőnél többfokozatú hálózatokat nem szokás építeni. Az eddigiekben ismertetett minimalizálási eljárások kétszintű hálózatokat eredményeznek {kivétel a "VALENCIA" módszer). A NAND, ill. NOR elemeknél a feszültség eltolódások elhanyagolhatók. Ez azt jelenti, hogy **univerzális elemekkel** többszintű hálózatok is készíthetők.

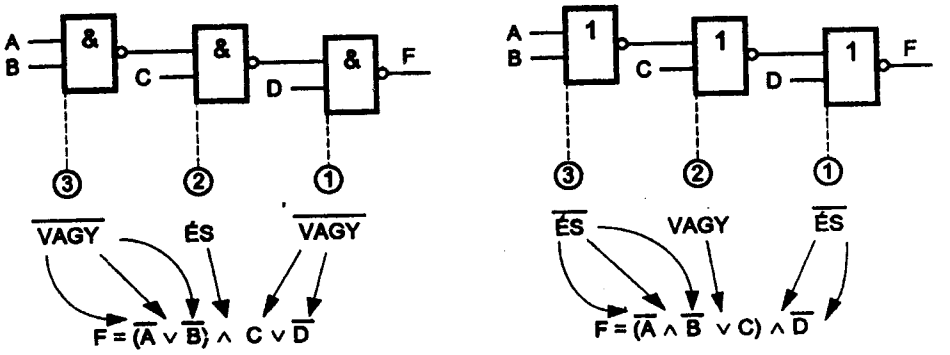
A többszintű hálózatok kimenő függvényének felírása (analízise) a **De Morgan** szabályok ismételt alkalmazásával lehetséges, de a szintek számával egyre nehezebb. Ezért adott kapcsolás analíziséhez a hálózatot a 3.64. ábra szerinti szintekre osztjuk.

Az analízis szabályai homogén NAND elemes hálózatnál:

1) A többszintű NAND hálózat minden **páratlan szinten** bevezetett változót **komplementál.**
2) **A páratlan szinten** bevezetett kapuk **VAGY** műveletet, a **páros szinten** bevezetett kapuk ÉS műveletet realizálnak a logikai függvényben.

Az analízis szabályai homogén NOR elemes hálózatnál:

1) A többszintű NOR hálózat minden **páratlan szinten** bevezetett változót komplementál.
2) A **páratlan szinten** lévő kapuk **ÉS**, a **páros szinten** lévő kapuk **VAGY műveletet** realizálnak a logikai függvényben.



3.64. ábra

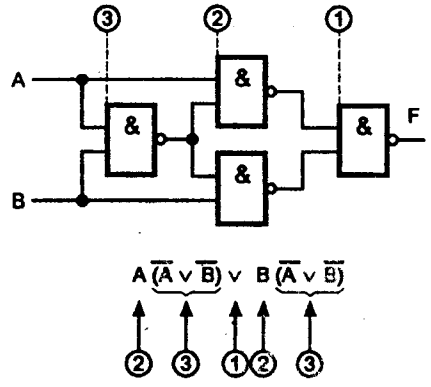
Valamely logikai függvény többfokozatú NAND hálózattal történő realizálásánál (szintézis) az a cél, hogy a függvényt - a Boole algebra szabályainak megfelelően - olyan alakra hozzuk, hogy realizálható legyen ÉS, illetve VAGY kapukkal, amelyek sorrendje az egymást követő szinteken:

... VAGY - ÉS - VAGY - ÉS ...

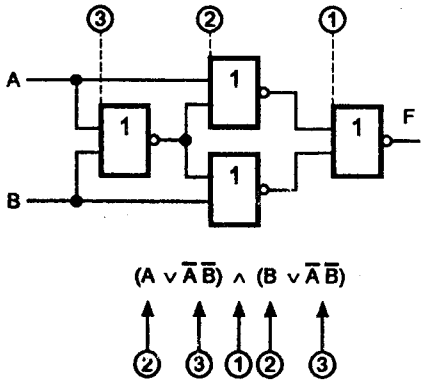
Célszerű továbbá a **negált változókat a páratlan, a ponált** változókat a **páros szinten** bevezetni. Példaként realizáljuk az antivalencia függvényt 3 szintű NAND hálózattal.

$$F = \overline{A}B \vee A\overline{B} = \overline{A}B \vee B\overline{B} \vee A\overline{B} \vee A\overline{A} = B(\overline{A} \vee \overline{B}) \vee A(\overline{A} \vee \overline{B}).$$

A megoldás a 3.65.a) ábrán látható. Hasonló átalakítással kapható az ekvivalencia 3 szintű NOR realizációja is (b) ábra.

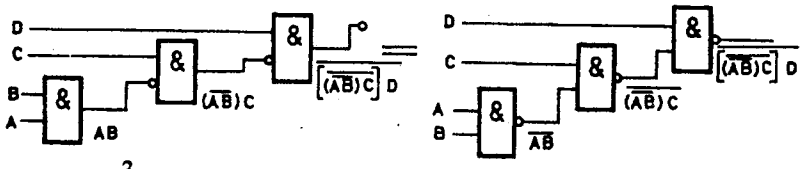


3.65.a) ábra



3.65.b) ábra

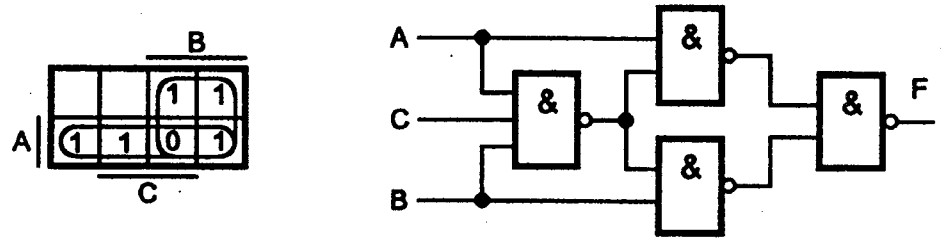
Valamely függvény többszintű NAND realizációja az **inhibíció** elv ismételt alkalmazásával grafikus módszerrel is meghatározható. Az inhibició és a NAND függvény közötti kapcsolatot a 3.66. ábra szemlélteti. Mint látható, **minden egyes NAND kapu tulajdonképpen inhibíciós műveletet** valósít meg.



Példaként az $F^3(A,B,C) = \Sigma(2,3,4,5,6)$ függvényt a 3.67. ábrán grafikusan is megadjuk. Az ismert eljárással egyszerűsítve, a következő ekvivalens megoldásokat kapjuk.

$$F = \bar{A}B \vee A\bar{B} \vee B\bar{C}$$

$$F = \bar{A}B \vee A\bar{B} \vee A\bar{C}.$$



a.) b.)

3.67. ábra

Mindkét megoldás kétfokozatú megvalósítása 3 db invertert, 3db kétbemenetű és 1 db hárombemenetű NAND kaputigényel. A 3.66. ábrán bejelölt tömbösítéssel:

$$F = (A \vee B)\overline{(ABC)} = A(\overline{A} \vee \overline{B} \vee \overline{C}) \vee B(\overline{A} \vee \overline{B} \vee \overline{C}).$$

A b) ábrán feltüntetett megoldás invertert nem tartalmaz.

Realizáláskor **páros szinten** elhelyezett kapukkal valósítjuk meg azon tömböket, melyek **1-eseket**, míg páratlanokon azokat, amelyek **0-kat** tartalmaznak.

A többszintű hálózatokkal történő realizálás szisztematikus módszerének az ún. **hálózat dekompozíció módszere** tekinthető. Ezzel kapcsolatosan terjedelmi okból az irodalomra utalunk.

Felhasznált irodalom

Ajtonyi I: Vezérléstechnika I. J 14-1417 Tankönyvkiadó, Budapest, 1987.
Ajtonyi I: Vezérléstechnika II. J 14-1417 Tankönyvkiadó, Budapest, 1987.
Ajtonyi I: Vezérléstechnika példatár J-1419 Tankönyvkiadó, Budapest, 1987.
Arató-Kondorosi-Risztics: Logikai hálózatok tervezése. TK. 1973.
Arató-Risztics: Logikai hálózatok tervezése I. TK. 1973.
Janovics-Tóth: A logikai tervezés módszerei Műszaki Könyvkiadó, 1971.
Szittya: Logikai kapcsolástan BME-Tankönyvkiadó, 1971.

4. DIGITÁLIS ÁRAMKÖRÖK

Az előző fejezetben megismertük a különféle funkciójú logikai alapelemeket. Ebben a fejezetben bemutatjuk ezen "sötét dobozok" belső felépítését, működését.

4.1. A digitális áramkörök jellemzői

A következőkben a logikai egységek áramköri leírásánál használt fontosabb fogalmakat ismertetjük. Ezek: logikai szintek, fan-out (F. O.), fan-in (F. I.), jelterjedési idő (t_{pd}), zavarvédettség, disszipáció $\left(P_D\right)$, karakterisztikák.

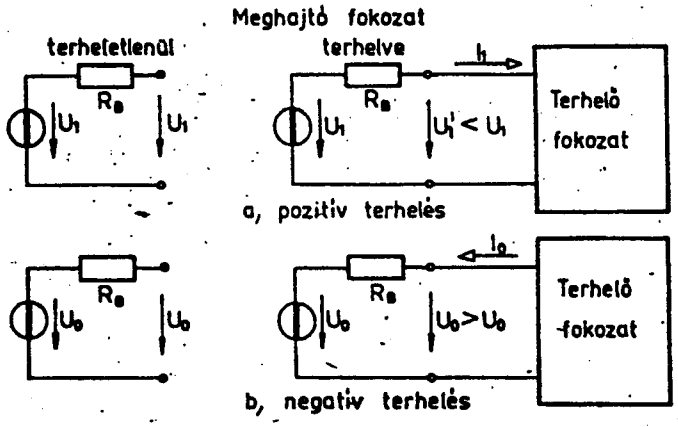
a) Logikai szintek

Az elektronikus digitális áramkörökben a 0 és 1 logikai értékeket hordozó fizikai jellemző leggyakrabban a **feszültség.** A logikai szintekhez rendelt feszültség értékeket **logikai szinteknek** nevezzük. Az áramkörök kimenetén az alkatelemek szórása és a változó környezeti feltételek. (hőmérséklet, terhelés, tápfeszültség) miatt a logikai értékekhez tartozó feszültség kisebb-nagyobb mértékben különbözik. Ezért a logikai értékhez egy **feszültségtartományt** szokás hozzárendelni. A logikai IGEN szint névleges értékét U_1-gyel, a felső határát $\overline{U}_1$-mal ($U_{1\,max}$) alsó határát $\underline{U}_1$-mal ($U_{1\,min}$) jelöljük. A NEM szint átlagos értékét U_o-val; a felső határát $\overline{U}_o$-mal ($U_{o\,max}$), alsó határát pedig $\underline{U}_o$-mal ($U_{o\,min}$) jelöljük. Megjegyezzük, hogy az aktív áramköröknél a bemeneten szélesebb tartományt engedünk meg, mint a kimeneten. Ez az áramkörök **zavarvédettségét** biztosítja.
Az áramköri leírásokban a pozitívabb logikai feszültségszintet magas (high, H), a negatívabb feszültségszintet alacsony (low, L) szintnek is szokás nevezni.

Pozitív logikáról beszélünk, ha az IGEN szintet a **pozitívabb** feszültséghez rendeljük, míg **negatív logika** esetén az IGEN szintet **negatívabb** feszültség reprezentálja. A gyakorlatban rendszerint a nagyobb abszolút értékű feszültségszinthez rendelik az 1 logikai értéket. A logikai 0-t többnyire a 0V-hoz rendelik. A logikai értéket meghatározott értékű **feszültségtartomány**, illetve meghatározott amplitúdójú **impulzus** jelenléte vagy hiánya reprezentálhatja. Előbbit statikus, utóbbit dinamikus megjelenítési formának nevezzük.

b) **Terhelhetőség (fan-out: F. O.)**

Mivel az egyes logikai egységek bemeneti impedenciája véges értékű, a bemenetek a vezérlő fokozat számára **terhelést** jelentenek. A terhelés lehet **negatív** vagy **pozitív** értelmű, attól függően, hogy a meghajtó generátor kapocsfeszültsége az eredeti üresjárási feszültségnél nagyobb lesz vagy kisebb lesz (4.1. ábra). Figyeljük meg a **meghajtó** ill. **terhelő fokozat** között folyó áramok irányát pozitív ill. negatív terhelés esetén! A digitális áramkörökben az egyes egységek bemenetei hasonló felépítésűek, így azonos nagyságú terhelést jelentenek. Ezt a leggyakrabban előforduló **áramértéket egységterhelésnek** nevezzük. Az egységek kimenetei az előírt specifikációt csak megengedett nagyságú terhelő áramok esetén teljesítik. A megengedett terhelő áram és az egységterhelés áramának hányadosát **dc. fan-out**-nak (egyenáramú terhelhetőség) nevezzük. A **dc. fan-out** tehát azt adja meg, hogy egy kimenet hány bemenetet vezérelhet. A bemenetek terhelő **kapacitása** a meghajtó áramkörök **jelterjedési idejét** növeli. Ezért néhány áramköri rendszerben külön definiálják az ún. **ac. fan-out** értékét, mely megadja azon vezérlő bemenetek számát, amelynél a specifikált **kapcsolási idők** még biztosan teljesülnek.



4.1. ábra

c) **Terhelés (fan-in: F. I.)**

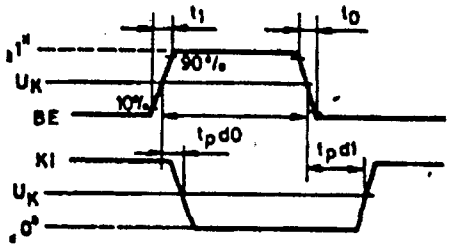
A katalógusokban Fan-in alatt egy bemenet által képviselt egységterhelést adják meg. Ha például egy rendszeren belül az egységterhelés értéke 1,5 mA, akkor az F.I. = 3 azt jelenti, hogy a kérdéses bemenet 4,5 mA-rel terheli a meghajtó fokozatot. A bonyolultabb áramköröknél az egyes bemenetek különböző F.I. értékűek lehetnek.

d) Terjedési idő

A digitális áramkörök a logikai funkciókat egy bizonyos időkéséssel realizálják. Az áramkörök működési sebességéről a terjedési idő ad felvilágosítást. A **terjedési idő** a kimenet megváltozásának késése a változást előidéző bemeneti jelhez képest. A kimenet 1→ 0 változásának késési idejét **lefutási késésnek** mondjuk és t_{pdo} -val jelöljük. Hasonlóképpen a kimenet 0→ 1 változásának késését **felfutási késésnek** nevezzük és t_{pd1} -gyel jelöljük. **Terjedési idő alatt a két késési idő átlagát értjük.**

$$t_{pd} = \frac{t_{pd1} + t_{pdo}}{2}. \qquad (4\text{-}1)$$

A logikai áramköröknél a terjedési idő a komparálási szint (U_K) elérésénél mérik (4.2. ábra). **Komparálási szint** alatt azt a feszültséget értik, amelynek elérésekor az áramkör **átvált** a másik állapotba.



4.2. ábra

e) Zavarvédettség

A digitális berendezésekben alkalmazott áramkörök bemenetein nem kívánatos, ún. **zavaró jelek** léphetnek fel. Ezek **külső** és **belső** eredetűek lehetnek. Ha a zavaró jelek amplitúdója olyan nagy, hogy az áramkör kimenetének állapota megváltozik anélkül, hogy a vezérlő áramkörök állapota megváltoznék, akkor hibás működés lép fel. Az áramköröknek a zavarójelekkel szembeni érzéketlenségét **zavarvédettségnek** (zajvédettségnek) nevezzük. **A zavarvédettséget a hibás működést még éppen nem okozó zavaró jel amplitúdójával jellemezzük.** Megkülönböztetünk **statikus** és **dinamikus** zavarvédettséget. **Statikus** zavaró jelek alatt az olyan zavaró jeleket értjük, melyek időtartama hosszabb, mint a logikai áramkörökben a jelek átlagos terjedési ideje $\left(t_{pd}\right)$. Tehát, ha a **zavaró jel egyenáramú** vagy **igen lassan változik,**

akkor **statikusnak** tekintjük. A statikus zavarvédettség (zajimmunitás) tehát azt a **járulékos feszültségszintet** adja meg, amely a vizsgált áramkör bemenetét vezérlő jelszintre adható anélkül, hogy az áramkör kimenetén lévő állapot emiatt megváltoznék. A **dinamikus zavarvédettség** esetén a t_{pd} időtartamnál rövidebb zavarjelek hatását vizsgáljuk. Jellemzésére a zavaró jel **amplitúdójának** és **időtartamának szorzata** használatos $(U_Z \cdot t_Z)$.

f) Disszipáció (P_D)

Disszipációnak azt a teljesítményt tekintjük, amely az áramkörben 50 %-os kitöltési tényezőjű vezérlés mellett hővé alakul.

g) Karakterisztikák

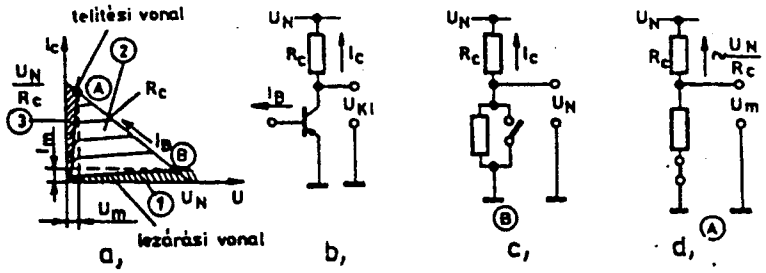
A logikai áramkörök viselkedése jól követhető a különböző **karakterisztikák** alapján. Ilyenek: bemeneti, kimeneti és transzfer karakterisztika. A megadás legtöbbször grafikusan történik.

4.2. Második generációs digitális áramkörök

Az érintkezős ill. elektroncsöves áramköröket kronológiai sorrendben a diszkrét félvezetős (második generációs) áramkörök követték. Az ilyen áramköri egységek diszkrét alkatrészekből (ellenállás, dióda, tranzisztor stb.) többnyire forrasztás útján vannak szerelve. A második generációs áramkörök tervezésénél és kivitelezésénél a minimális alkatrész és szerelési igény volt a legfőbb szempont. A konkrét áramkörök analíziséhez tekintsük át a félvezetők kapcsoló üzemét.

a) Bipoláris tranzisztoros logikai áramkörök

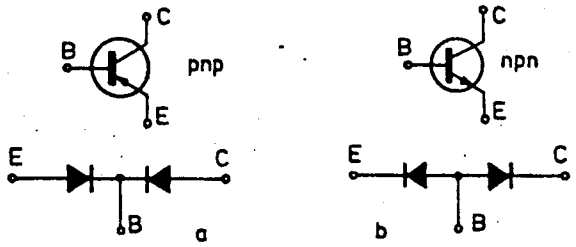
A tranzisztor **földelt emitteres** kapcsolásának előnyei kapcsoló üzemben is fennállnak, így a logikai áramkörökben leggyakrabban ez a kapcsolási mód használatos. Üzem közben a tranzisztor pillanatnyi állapotát a munkapont helyzetével jellemezhetjük. a 4.3.a) ábrán a földelt emitterű tranzisztor és kimeneti karakterisztikája látható. Az R_C munkaellenállással terhelt tranzisztor lehetséges munkapontja az R_C egyenesen helyezkedik el. A tranzisztor kimeneti karakterisztikájában **három** tartományt különböztetünk meg. E tartományok jellemzéséhez célszerű megrajzolni a tranzisztorok diódás helyettesítő képét (4.4. ábra).



4.3. ábra

A három tartomány:

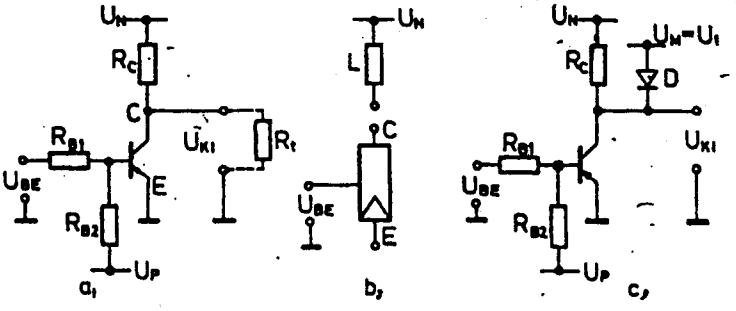
1. **Lezárási tartomány**: ebben a tartományban mind a B-E, mind pedig a C-B dióda záróirányú feszültséget kap, tehát mindkét dióda zárt.
2. **Az aktív (lineáris)** tartományban a B-E nyitó, a C-B pedig záró irányba előfeszítve üzemel. **Ez a tartomány a tranzisztor mint kapcsoló szempontjából érdektelen és kerülendő.**
3. A **telítési tartományban** mind a B-E, mind pedig a C-B diódán nyitóirányú feszültség van, azaz mindkét dióda vezet.



4.4. ábra

A lezárási tartomány és az aktív tartomány határa a zérus B-E feszültségű $\left(U_{BE}=0\right)$ illetve zérus bázisáramú karakterisztika mentén van. Az aktív tartomány és a telítési tartomány határa annak a vonalnak a mentén van, amelyiknél a C-B feszültség zérus. A 4.3. ábra karakterisztikája alapján látható, hogy az 1 illetve 3 tartományban üzemeltetett tranzisztor **veszteséges kapcsolónak** tekinthető. A lezárási (1) tartományban üzemel a pnp tranzisztor, ha bázisára 0V vagy annál **pozitívabb** feszültség kerül. Ilyenkor tehát **mindkét dióda zárt** állapotban van, így a kollektor áram a maradékáramra (Im) korlátozódik (B munkapont). Eszerint a kellően nagy jellel **lezárt tranzisztor nyitott veszteséges** kapcsolóként viselkedik. A pnp tranzisztor **telített** állapotba kerülésének feltétele, hogy bázisára az emitteréhez képest **kellően nagy negatív feszültség jusson**. Ekkor a **B-E dióda kinyit**. A bázisáram növekedésével az ismert összefüggés szerint **nő a kollektoráram**, ugyanakkor csökken a kimeneti feszültség. A munkapont az **aktív tartományon át** az A pont felé vándorol.

Amikor a rohamosan csökkenő U_{CE} eléri U_{BE} értékét, a **C-B dióda is kinyit** (A munkapont). A telítési tartományban üzemelő tranzisztor **kollektorárama nem függ a bázisáramtól,** értéke jó közelítéssel: $\frac{U_N}{R_C}$. Ilyenkor a kimeneten U_m **maradékfeszültség** mérhető. Eszerint a telítésbe vitt tranzisztor a **veszteséges kapcsoló zárt** állapotának felel meg. A tranzisztort logikai áramkörökben az **invertálás** műveletének realizálására, valamint **teljesítmény kapcsolóként** alkalmazzák. A tagadás műveletét megvalósító inverter kapcsolása látható a 4.5. ábrán.



4.5. ábra

Elemezzük a 4.5.a) ábra szerinti ún. **szabadszintű** inverter működését. Az inverter bemenetére jutó U_0 jelnek a tranzisztort biztonságosan le kell tudni zárni. Ehhez a bázisára enyhén **pozitív** feszültséget kell juttatni. Ezt biztosítja a **bemeneti feszültség osztó.** Ugyanis, ha a vezérlő feszültség OV, akkor az U_p pozitív feszültség az $R_2 - R_1$ ellenállásokból álló osztón a bázisra jut, tehát a tranzisztor lezár. Kimenetén terhelés nélkül közel U_N; R_T terhelés esetén pedig a 4-2 szerinti feszültség jelenik meg.

$$U_{ki} = U_N \frac{R_T}{R_C + R_T}. \qquad (4\text{-}2)$$

A szabadszintű inverter logikai IGEN szintje tehát nem állandó érték, hanem a **terhelés függvénye.** A tranzisztort a bemenetére jutó U_1 jelnek **telítésbe** kell juttatni. A tranzisztor biztonságos nyitása végett a telítés határesetéhez tartozó értéknél nagyobb nyitó irányú feszültség kerül. Az inverter kimenetén ilyenkor a maradék feszültség jelenik meg. Az IGEN jelet szolgáltató szabadszintű inverter kimeneti ellenállása nagy értékű (1...2 kΩ), NEM szint esetén pedig kicsi, a vezető tranzisztor ellenállásával egyezik meg (10...20 Ω).

A **szabadszintű rendszer előnyei:** 2 tápfeszültséget igényel, olcsóbb és gyorsabb működésű.

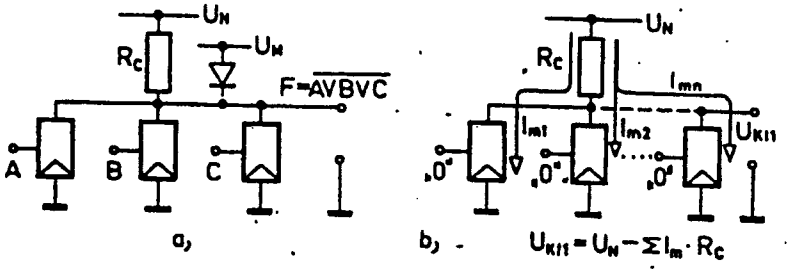
Hátrányai: kimenő ellenállása változó, az U_1 változó, az egységek illesztése nagyobb áramköri gyakorlatot igényel.

A **megfogott (kötött) szintű inverter** kapcsolása technikailag csak a D **megfogó diódában** különbözik 4.5.c) ábra, a szabadszintűtől. A kötött jelszintű kapcsoláshoz az U_N és U_P feszültségen kívül szükség van egy U_M **megfogó** feszültségre is.

A kötött szintű rendszer előnyei: mindkét jelszint közel állandó értékű, mindkét állapotban kis kimenő impedencia miatt nagyobb zajvédettség, az egységek illesztése egyszerű.

Hátrányai: nagyobb teljesítmény igény, 3 tápfeszültséget és többletdiódát igényel.

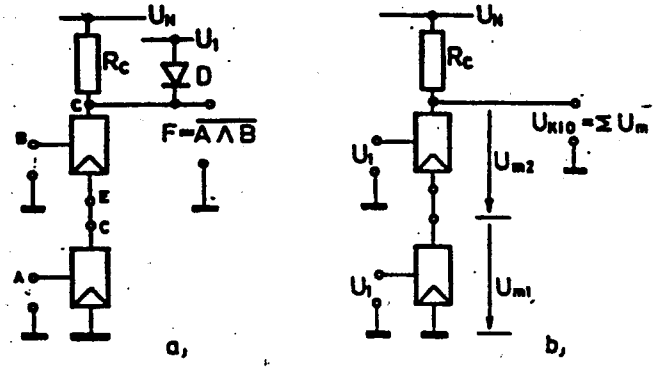
A tagadás műveletén kívül a NAND illetve NOR kapcsolatok is realizálhatók inverterekkel. E célból a logikai kártyákon a kollektor és emitter kivezetéseket **szabadon** hagyják. Ennek szimbolikus jelölése látható a 4.5.b) ábrán. Inverter **párhuzamos kapcsolásával a NOR művelet** valósítható meg. Ha a 4.6.a) ábrán párhuzamosan kapcsolt inverterek valamelyik bemenetére U_1 jelet adunk a szóban forgó tranzisztor telítésbe kerül, s a **közös kollektor** pontot a maradék feszültség értékére csökkenti. A kimeneten csak **valamennyi** tranzisztor **lezárt állapotában** van U_1 jel. A párhuzamosan kapcsolható tranzisztorok számát a zárt tranzisztorok **maradék áramai korlátozzák,** melyek a közös R_C-n a terhelő árammal összegeződve a kimenő feszültséget U_1 alá csökkenti. Emiatt legföljebb 10...15 tranzisztor köthető párhuzamosan.



4.6. ábra

A 4.7. ábrán a **NAND kapcsolatot** inverterek **soros kapcsolása** révén realizáltuk. Amennyiben legalább egy bemeneten U_0 jel van, ez a tranzisztor

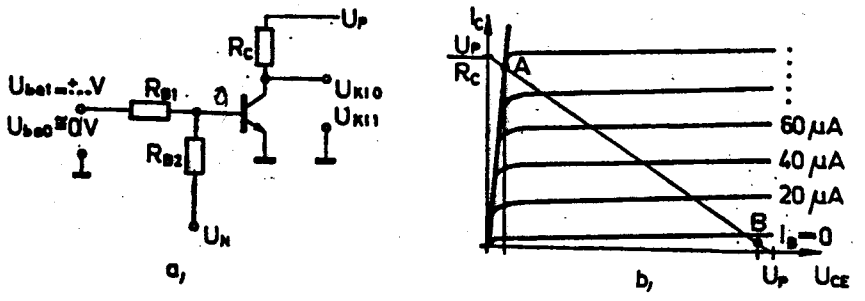
lezár, s a kimeneten U_1 mérhető. Ha valamennyi bemenetet U_1-gyel vezéreljük, akkor valamennyi tranzisztor vezet, s a kimeneten a **maradékfeszültségek összege** jelenik meg. A sorba köthető tranzisztorok számát a vezető tranzisztorokon eső **maradék feszültségek** korlátozzák, ugyanis ezek összegződve túllépheti a logikai NEM jelre specifikált érték felső határát $\overline{U}_O$. Emiatt 3 tranzisztornál többet nem szokás sorba kapcsolni.



4.7. ábra

Napjainkban a tranzisztorok soros ill. párhuzamos kapcsolásával a CMOS áramköröknél találkozunk. Mivel a NAND és NOR függvények **univerzális** műveletek, látható, hogy inverterek soros, illetve párhuzamos kapcsolásával tetszőleges logikai hálózatok építhetők.

Az eddigiekben a pnp tranzisztoros inverter működését vizsgáltuk. A következőkben a **pozitív logikájú, npn szilícium tranzisztoros inverter** felépítését, működését ismertetjük részletesen. Kapcsolása a 4.5. ábra feszültségviszonyainak értelemszerű cseréjével adódik 4.8.a) ábra.



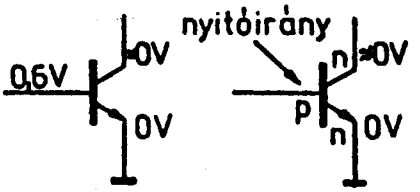
4.8. ábra

Logikai 0 (≈ 0V) feszültség vezérlés esetén a tranzisztor bázis-emitter diódája lezár, a kollektoráram gyakorlatilag 0-ra csökken, a munkapont, (B) a munkaegyenesen az $I_B = 0$ karakterisztika vonal **alatt** van 4.8.b) ábra. Ilyenkor az I_{CER} kollektor **maradékáram** folyik, amely kisebb, mint az $I_B = 0$-hoz tartozó I_{CEO}, és amely szilícium tranzisztoroknál általában elhanyagolható (kisteljesítményű tranzisztorokra szobahőmérsékleten nA alatti értékű). A germánium tranzisztoros logikában az inverterek bemenetére az R_1 áramkorlátozó ellenálláson kívül egy **ellenkező feszültségre kapcsolt** R_2 ellenállást is tesznek a **biztos lezárás** érdekében (ld. előbb)! A kollektoráram ilyenkor I_{CER}-nél is kisebb. A lezáró feszültséget szolgáltató R_2-őt esetenként szilícium tranzisztoros logikáknál is alkalmazzák azért, hogy a logikai 1 vezérlő feszültségről logikai 0 feszültségre történő átváltáskor a **lezárás gyors legyen**. Integrált áramkörökben ezt a megoldást kerülik a kétféle tápfeszültség igénye miatt. Az elhanyagolhatóan kis kollektoráram az R_C munkaellenálláson átfolyva elhanyagolhatóan kis feszültséget hoz létre rajta, így a kimeneti feszültség $\left(U_{kil}\right)$ terhelés nélkül gyakorlatilag megegyezik a tápfeszültséggel. Végeredményben a logikai 0 bemenetre logikai 1 a válasz a kimeneten. Ha a tranzisztort a digitális berendezésben **meghajtó áramkörként** (teljesítmény kapcsolóként) használjuk, akkor fokozottan kell ügyelni a **fogyasztók nemlinearitására pl. izzó, LED dióda, induktív fogyasztó, relé tekercs** működtetése esetén. **A telítéses üzemmód egyik ismérve** ezek szerint:

$$I_B \gg \frac{I_C}{B},$$

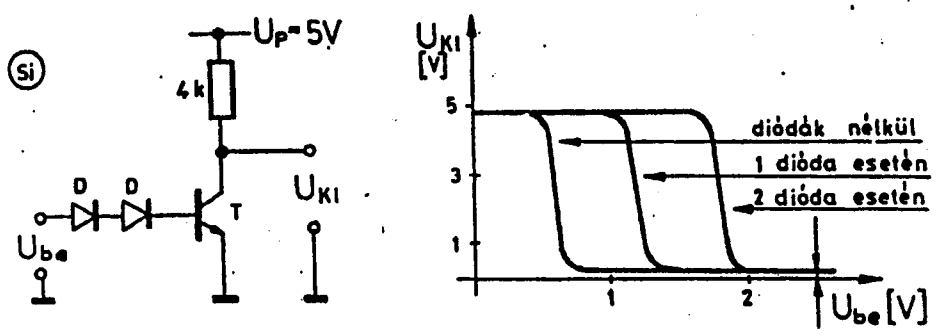
vagyis a bázisáram sokkal nagyobb, mint az adott kollektoráramhoz a lineáris üzemmódban szükséges érték.

A **telítéses üzemmód** másik ismérve az ilyenkor fellépő sajátságos potenciálviszonyokból adódik; NPN tranzisztor esetén a bázis feszültsége a 0 V-os emitterhez képest kb. + 0,6...0,7 V, kollektor feszültsége 10...100 mV nagyságrendű, gyakorlatilag 0 V (4.9. ábra). NPN rétegsorrendnél ez azt jelenti, hogy mivel a P bázisréteg + 0,6 V-on van, valamint az N emitter és a N kollektor egyaránt 0 V-on van, **nemcsak a bázis-emitter dióda, hanem a kollektor-bázis dióda is nyitó irányba van előfeszítve.** Ez lényegesen eltér a lineáris üzemmódban szokásos záróirányú kollektor-bázis feszültségű munkaponti beállítástól.



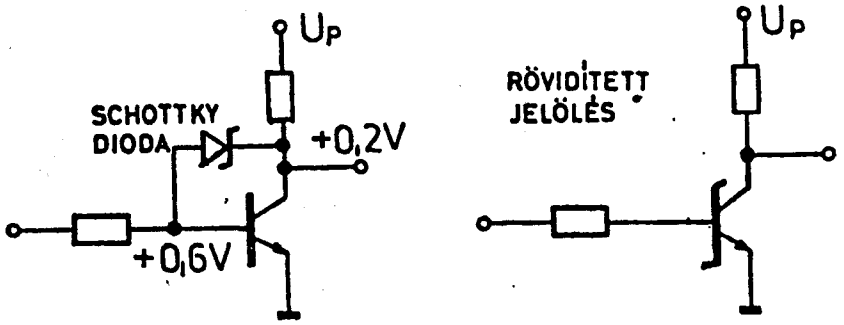
4.9. ábra

Láttuk, hogy a szilícium tranzisztor telítésbe viteléhez legalább 0,7 V nagyságú nyitófeszültség szükséges. Ezért az ilyen invertereknél a lezáró feszültség el is hagyható. Növelhető a kapcsolás **zajvédettsége nem lineáris elemek (dióda, zener dióda)** beiktatásával. A 4.10. ábrán a kapcsolást és a transzfer karakterisztikát is megrajzoltuk. Természetesen az ilyen inverterek is készíthetők kötött ill. szabadszintű kivitelben.



4.10. ábra

A másik megoldás szerint az inverter kapcsolást egy diódával úgy egészítik ki, hogy a **kollektor-bázis dióda ne tudjon kinyitni** (4.11. ábra).



4.11. ábra

Ehhez olyan fajta dióda kell, melynek nyitófeszültsége kisebb a kollektor-bázis (Si) dióda nyitófeszültségénél. Ilyen a **Schottky dióda**, mely kb. 0,3...0,4 V-

nál nyit ki. Amikor az inverter bemenetére pozitív log 1 feszültség érkezik, a bázis feszültsége 0,6 V körüli lesz. A kollektor feszültsége nem csökkenhet le közel 0 V-ra, mert a "megfogó dióda" kinyit, és mivel anódja a bázison, + 0,6 V-on van, a kollektor potenciálját nem engedi + 0,2 V-nál lejjebb csökkenteni. A Schottky dióda a kinyitás után a soros bázis korlátozó ellenállás áramának nagy részét is elvezeti a kollektor felé, így a bázisáram csak akkora lesz, amekkorának a kb. + 0,2 V-os kollektorfeszültség eléréséhez kell lennie (és a dióda árama is a kollektoráramhoz adódik). **A tranzisztor tehát nem lesz telítésben**, így lezárása sokkal rövidebb idő alatt végbemehet. A lezárást a dióda nem akadályozza, mert a pozitív kollektorfeszültség záróirányba feszíti elő. A Schottky diódás integrált áramkör változatokat elterjedten alkalmazzák (pl. Schottky-TTL, Schottky védett I^2L, stb.). Működésüket a későbbiekben tárgyaljuk.

b) Diódás logikai áramkörök: (DDL-DIODE-DIODE LOGIC)

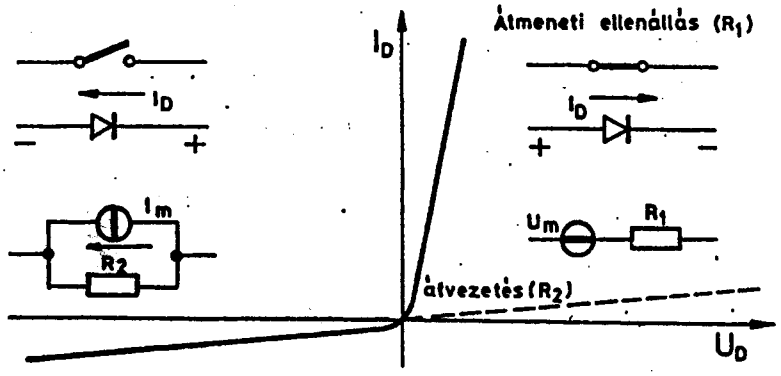
Az ideális dióda - mint tudjuk - egyik irányban szakadást, a másikban viszont rövidzárat jelent. A valóságos diódák tulajdonságai csak durván közelítik meg az ideális diódáét. A félvezető dióda olyan kétállapotú veszteséges kapcsolónak felel meg, amely **vezet**, ha kellően **nagy nyitó irányú** feszültség jut elektródáira, **nem vezet**, ha **kellő mértékű záró irányú** feszültség jut elektródáira.

A vezető dióda R_1 átmeneti ellenállással és U_m maradékfeszültséggel, a zárt dióda R_2 átvezetési ellenállással és I_m maradékárammal jellemezhető (4.12. ábra). A gyakorlatban azokat a diódákat alkalmazzák, melyek záróirányú ellenállása legalább $10^4 \ldots 10^6$-szorosa nyitóirányú ellenállásuknak. Tájékoztatásul a germánium és szilícium diódák hozzávetőleges adatait a 4.1. táblázatban adtuk meg.

4.1. táblázat

Típus	$R_1[\Omega]$	$R_2[M\Omega]$	$I_m[\mu A]$	$U_m[V]$
Ge	20...100	0,1...1	10...100	0,3
Si	60	$10^2 \ldots 10^3$	0,01...0,1	0,7

A diódás logikai áramköröket **passzív** áramköröknek is szokás nevezni, mert aktív elemet nem tartalmaznak. **Diódás áramkörökkel a konjunkció és a diszjunkció műveletét lehet megvalósítani.**

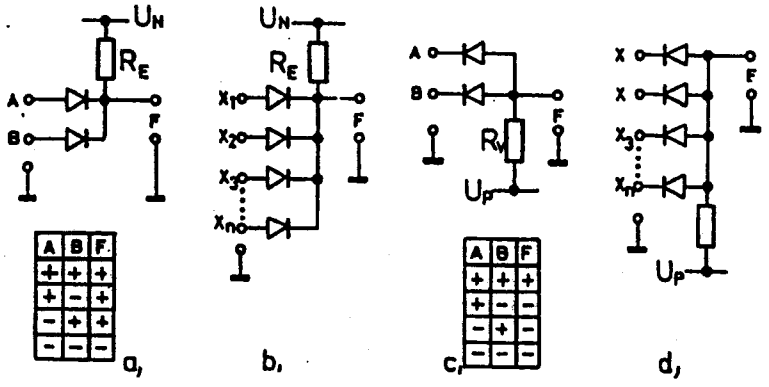


4.12. ábra

A 4.13.a)b) illetve c)d) ábrákon két különböző felépítésű diódás kaput láthatunk. A be- és kimeneti jeleket összevetve az ÉS és VAGY műveleteket definiáló értéktáblázatokkal, látható, hogy csak a **pozitív és negatív logika választásán** múlik, hogy melyik kapu végzi az ÉS, melyik a VAGY műveletet. Alkalmazzuk a következő hozzárendelést:

negatív logika: "-" = 1, "+" = 0
pozitív logika: "-" = 0, "+" = 1.

Eszerint az a)b) ábra **pozitív logikában VAGY, negatív logikában ÉS**, a c)d) ábra **negatív logikában VAGY, pozitív logikában ÉS** műveletet realizál. A diódás áramkörök tervezését a karakterisztika alapján meghatározható lineáris helyettesítő kép alapján végezhetjük el.



4.13. ábra

4.3 Digitális integrált áramkörök

A diszkrét félvezetős logikai áramköröket az integrált technikával készített ún. harmadik generációs áramkörök váltották fel. Ezek közül a legjobban elterjedt a TTL ill. CMOS bázisú rendszer.

4.3.1. A TTL rendszer

A tranzisztor-tranzisztor logikai rendszer (TTL, T^2L) megvalósítását az **integrálási technológia** tette lehetővé, melynek kiinduló **alapanyaga a szilícium egykristály** lemezke, amelyre egymás után következő diffúziós lépésekkel alakítják ki a tranzisztorokat, diódákat, ellenállásokat. Több lapka esetén chip-ről beszélünk. Az integrált áramköri technológiával készült áramkörök (integrated circuits, IC) lényeges ismérve, hogy megbonthatatlan fizikai egységet képeznek, alkatelemeikre roncsolás nélkül nem szedhetők szét. Az IC-k előnyei: kis méret és súly, nagy megbízhatóság, hosszú élettartam, nagy alkatrész sűrűség, nagy működési sebesség, kis fogyasztás. Az IC-k felosztása az integráltság mértéke szerint:

- **SSI** (Small-Scale Integration) kis mértékű integráltságú (1...10 kapu)
- **MSI** (Medium-SI) közepes integráltságú (10...99 kapu), pl. számlálók
- **LSI** (Large-SI) nagy mértékű integráltságú (99...999 kapu), pl. 8 bites μP
- **VLSI** (Very LSI) igen nagy mértékű integráltsága, pl. 16/32 bites μP
- **ULSI** (Ultra LSI) ultra nagy mértékű integráltságú, pl. 64 bites mikroprocesszorok
- **GSI** (Gigantic Scale Integration) gigantikus méretű integráltságú (pl. 64 Mbit RAM, integrált multiprocesszorok).

Az IC-k tokozott formában kerülnek forgalomba. A különböző tokozási formákat a 4.2. táblázatban foglaltuk össze.

4.2. táblázat

Tokozási típusok (package)	Tipikus lábszám	Szerelés
Dual-in-line (DIP)	8-64	TH
Single-in-line (SIP)	5-40	TH
Zig-Zag-in-line (ZIP)	14-28	TH
Quad-in-line (QUIP)	14-64	TH
Small outline (SO)	8-32	SM
Chip carrier (CC)	16-200	SM
Flat pack (FP)	10-300	SM
Pin grid array (PGA)	68-500	SM

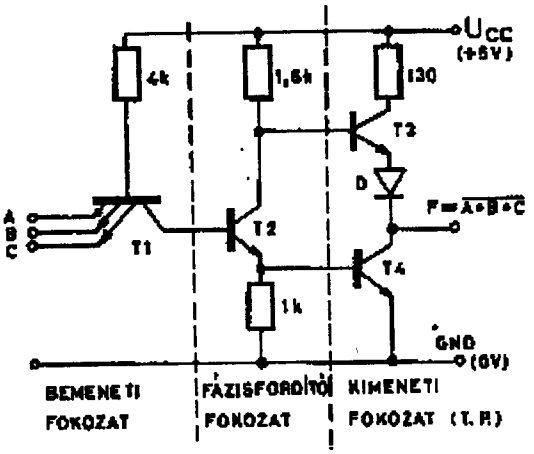
A táblázatban a TH (through-hole) a hagyományos NYÁK szerelést, az SM (surface mounting) felületszereléses technológiát jelöl. A rajzokon az **áramkörök kivezetéseit felülnézetben** ábrázolják. A továbbiakban a Texas Instruments cég által bevezetett TTL áramköröket (SN74 rendszer) ismertetjük.

Az SN74 rendszeren belül a következő sorozatokat fejlesztették ki:

- standard (SN74)
- kis teljesítményű (Low Power: 74L)
- nagy sebességű (High Speed: 74H)
- Schottky (74S)
- kis teljesítményű Schottky (74LS)
- módosított LS (TTL-ALS)
- módosított S (TTL-AS).

4.3.1.1. A TTL NAND kapu

A TTL rendszer alapeleme a pozitív logikájú NAND kapu, amely 3 részből áll: bemeneti fokozat, fázishasító fokozat, kimeneti fokozat (4.14. ábra).



4.14. ábra

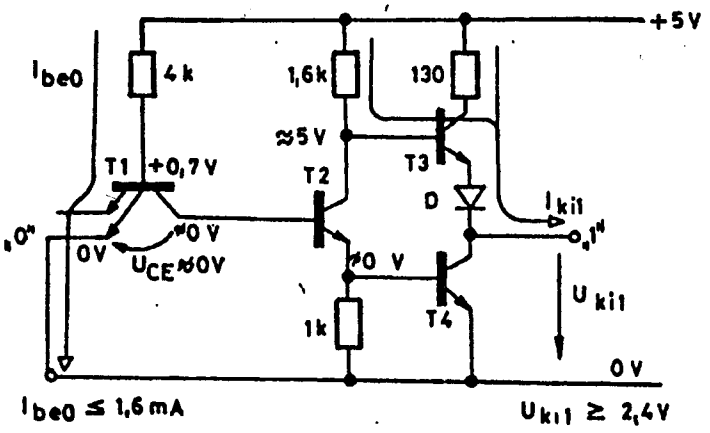
A TTL NAND kapu bemeneti fokozatát az ún. **multiemitteres tranzisztor** képezi, amely az ÉS kapcsolatot valósítja meg (B-E diódák) és szinteltolást végez (C-B dióda). A bemeneti fokozatot a **fázishasító** fokozat követi (T_2), amely T_3, T_4 kimeneti tranzisztorokat működteti **ellenütemben**. A TTL NAND kapu kimenete háromféle kivitelű lehet:

a) **ellenütemű** (totem-pole: T.P.)
b) **nyitott kollektoros** (open collector: O.C.)
c) **három állapotú** (three state: T.S.).

a) Az ellenütemű kimenetű TTL NAND kapu

A kapu működésének vizsgálatát két esetre kell megvizsgálni: ha **legalább egy bemeneten** U_0 szint van, ill. **minden bemeneten** U_1 szint van. Ha valamelyik bemeneten van $U_0 = 0\,V$ (4.15. ábra), akkor a földelt bemenethez tartozó dióda

kinyit, hiszen bázisa a 4 kΩ-on keresztül nyitóirányú előfeszítést kap. A T_1 vezető B-E diódáján kb. 0,7 V feszültségesés jön létre, T_1 kinyit, a C-B dióda vezet, így a C-E feszültsége gyakorlatilag 0 V körüli feszültség a T_2, T_4 tranzisztort lezárja. Mivel a T_2 lezár, a T_4 tranzisztor bázisába az 1,6 kΩ-os ellenálláson keresztül áram folyik és a kimenet feszültségét a nyitott D diódán keresztül **pozitív potenciálra** kapcsolja.



4.15. ábra

A kimeneti feszültség üresjárásban:

$$U_{Ki1} = U_{CC} - U_{BE3} - U_D = 5V - 2 \times 0{,}7\ V = 3{,}6\ V$$

$$I_{Ki1max} = 0{,}4\ mA \text{ esetén (F.O.} = 10) \text{ az } U_{Kimin} = 2{,}4\ V.$$

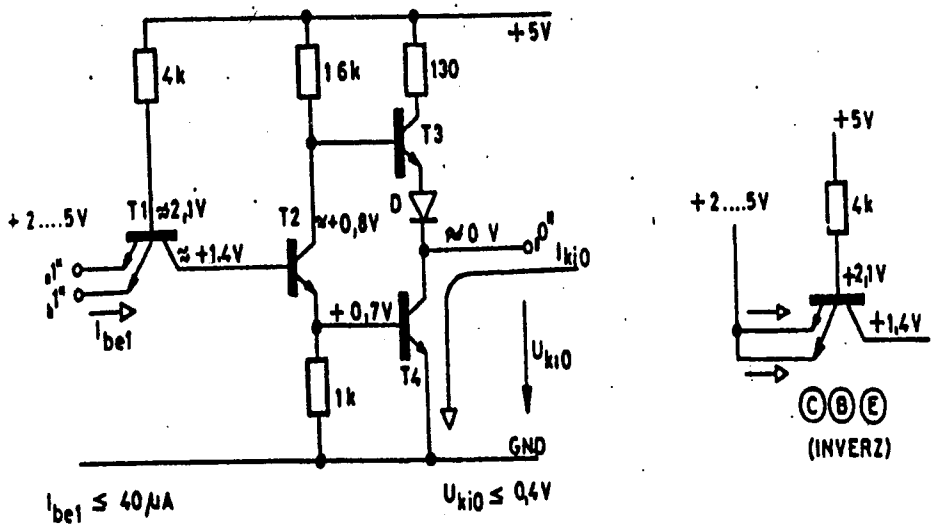
A bemeneten folyó áram:

$$I_{be0} = \frac{U_{CC} - U_{BE} - U_{be0}}{4\,k\Omega} \sim \frac{5 - 0{,}7}{4} \sim 1{,}1\ mA.$$

A legkritikusabb érték $I_{be0max} = 1{,}6\ mA$.

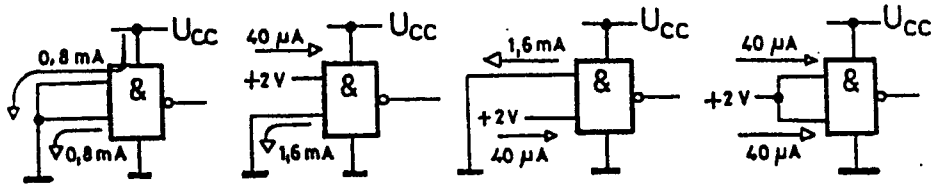
Az I_{be0} a bemenetet meghajtó áramkör szempontjából **a bemenet feszültségét pozitív irányba növelő „húzóáram"** (Sink current). Az I_{be0} áram értéke függ a „0"-ra kapcsolt bemenetek számától. Az összekötött bemenetek U_0 szinten nem növelik a meghajtó fokozat terhelését, mert **az áramok megoszlanak.**

Ha mindegyik bemenet feszültsége min. 2 V, akkor a T_1 B-E diódái lezárnak, mivel az emitterek magasabb potenciálra kerülnek a bázishoz képest. A T_1 C-B diódája mindig vezet, s a kollektor feszültség ~ 1,4 V, ami nyitja a T_2 és T_4 tranzisztorokat. T_4 vezet, így a kimeneten U_{ki0} feszültség jelenik meg. A T_3 tranzisztor zárását egyrészt a T_2 tranzisztor kollektor feszültsége, másrészt a D szinteltoló dióda biztosítja. A fellépő potenciálviszonyokat szemlélteti a 4.16. ábra. A bemeneten fellépő „inverz" áram átlagos értéke $I_{be1} = 5-10\ \mu A$, maximális értéke $I_{be1max} = 40\ \mu A$. **Valamennyi U_1-re kapcsolt bemeneten I_{be1} áram folyik.**



4.16. ábra

Egy kétmenetű NAND kapu bemenetein folyó áramokat rajzoltuk meg a 4.17. ábrán a legkritikusabb értékek feltüntetésével.



4.17. ábra

A TTL kapu bemenetein fellépő feszültség és áram értékek szabványban rögzítettek. Ez a **„TTL specifikáció"** napjainkban szinte minden digitális

berendezésre vonatkoztatható, s a leírásokban egyszerűen „be és kimeneteik TTL kompatibilisek" megjegyzéssel látják el. A TTL specifikáció kiemelt jelentőségű, ezért a 4.3. táblázatban foglaltuk össze.

4.3. táblázat

$I_{be0max} = -1{,}6$ mA	$U_{be0max} = 0{,}8$ V
$I_{be1max} = 40$ µA	$U_{be1min} = 2$ V
$I_{ki0max} = 16$ mA	$U_{ki0max} = 0{,}4$ V
$I_{ki1max} = 400$ µA	$U_{ki1min} = 2{,}4$ V

Fel kell hívni a figyelmet arra, hogy a fenti értékek a **normál sorozatra** vonatkoznak. A különböző sorozatú TTL áramkörök (S, LS, stb.) a fenti értékekben eltérnek. A TTL specifikált értékei a transzfer karakterisztikán is megtalálhatók. A transzfer karakterisztikának négy jellegzetes szakasza van (4.18. ábra).

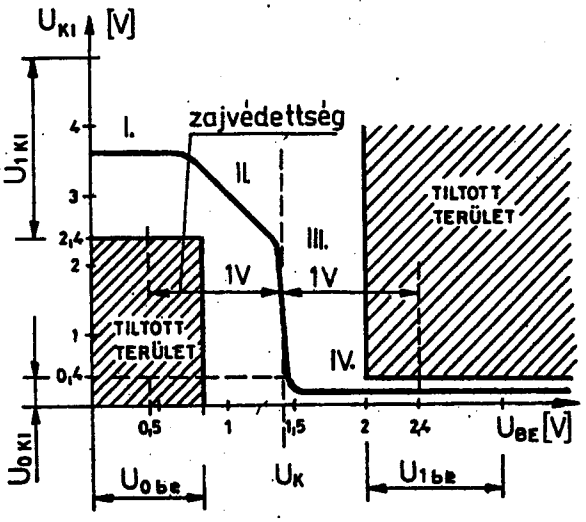
Az I. szakasz jellemzői:

$U_{be} \leq 0{,}8$ V, T_2 zár, T_4 zár, T_3 vezet, $U_{Ki} \geq 2{,}4$ V.

Az II. szakasz jellemzői: $U_{be} = 0{,}8$ V...1,4 V statikusan tiltott állapot.

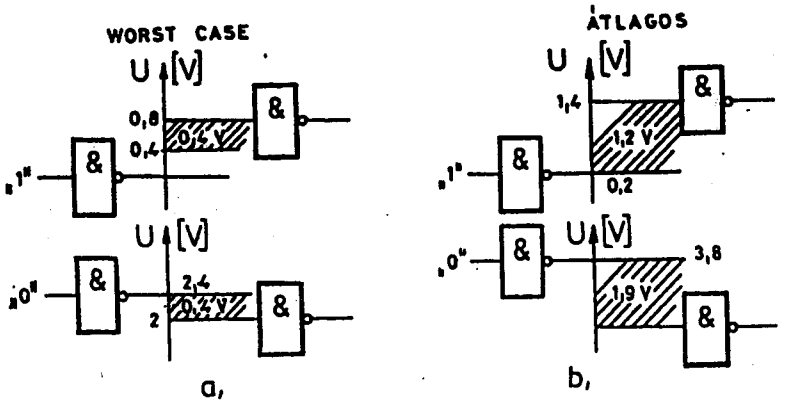
Az III. szakasz jellemzői: $U_{be} = 1{,}4$ V, a komparálási feszültség, ilyenkor mind a 4 tranzisztor vezet, s a kapu áramfelvétele megnő, statikusan tiltott állapot.

Az IV. szakasz jellemzői: $U_{be} \geq 2$ V, $U_{Ki} \leq 0{,}4$ V.



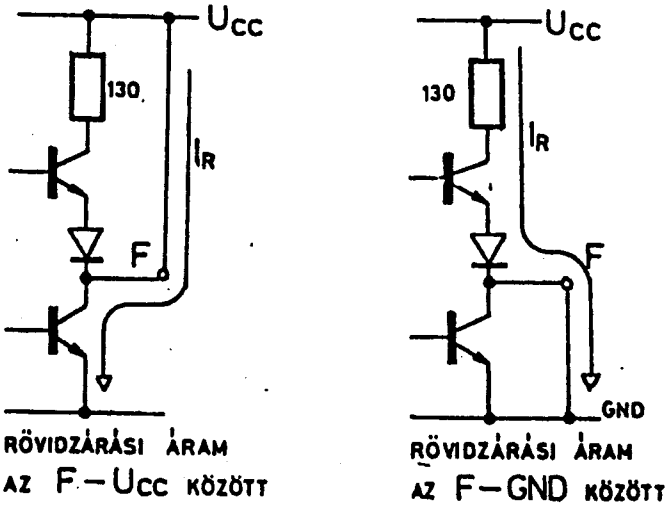
4.18. ábra

A transzfer karakterisztika természetesen függ a tápfeszültség, a hőmérséklet és a F.O. értékétől. A transzfer karakterisztika és a TTL specifikáció alapján meghatározható az áramkörök zajvédettsége. A legkritikusabb esetben a zajvédettség a (4.19.a) ábra) szerinti, azaz a 0 jel legkritikusabb értékéhez (0,4 V) még 0,4 V zavarójel szuperponálódhat, hiszen a következő áramkör a 0,8 V-ot még garantáltan 0 jelként kezeli.



4.19. ábra

U_1 szinten ugyancsak 0,4 V a zajvédettség a legkritikusabb esetben. Ez azt jelenti, hogy az áramkör kimenetén az U_{kilmin} = 2,4 V, még 0,4 V-ot csökkenhet, mert a következő kapu a 2 V-ot még garantáltan 1 jelként kezeli. Átlagos üzemi körülmények között (szobahőmérséklet, F.O. < 10, U_{CC} = 5 V) a zajvédettség magasabb, 1 V körüli (4.19.b) ábra). Az A.C. (váltakozó áramú) zajvédettség a gyorsan változó, nagyfrekvenciális zavarjelek maximális megengedett amplitúdó x Δt értékét adja meg, amellyel még nem érzékeli tévesen a bemeneti logikai szinteket. Minél rövidebb idejűek a zavaró impulzusok, annál nagyobb lehet az amplitúdójuk anélkül, hogy zavart okoznának a működésben. Külön kell foglalkozni a T.P. kimenetű kapuk közvetlen U_{CC}-re ill. GND-re kapcsolása esetén fellépő viszonyokra. A T.P. kimenet közvetlen GND-re kapcsolása esetén a kimenet és a GND között rövidzárási áram folyik, amelyet csak a 130 Ω-os ellenállás korlátoz. Ez a rövidzárási áram $\left(Iz = \frac{5\,V}{130\ \Omega}\right)$ nem, vagy nem azonnal okozza a kapu tönkremenetelét. A **T.P. kimenet azonnali tönkremenetelét** a „0"-ba vezérelt kimenet közvetlen U_{CC}-re kapcsolása eredményezi. Ilyenkor ugyanis nincs korlátozó ellenállás az áramkörben. A kimeneten fellépő zárlati áramokat szemlélteti a 4.20. ábra.

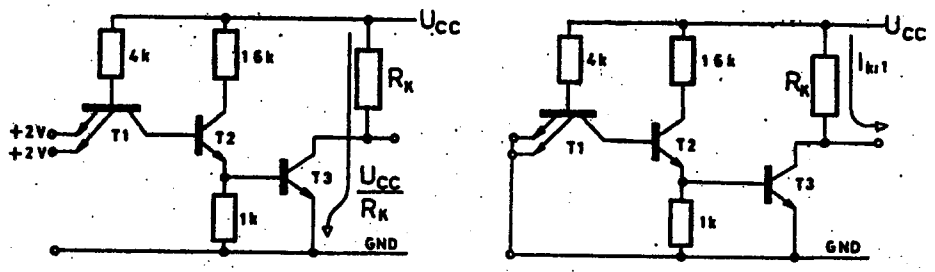


4.20. ábra

Zárlati áramok léphetnek fel az ellentétesen vezérelt T.P. kimenetű kapuk kimeneteinek összekötése esetén is. Ezért a T.P. kimenetű kapuk kimeneteinek összekötése csak abban az esetben megengedett, ha a bemeneteik azonos vezérlést kapnak (össze vannak kötve). Ezt a kapcsolást leginkább a terhelhetőség (F.O.) növelésére használják.

b) Nyitott kollektoros (O.C.) kapu

A T.P. kimenetű kapu működését az előzőekben részletesen bemutattuk. Ennek **hátránya, hogy a különbözőképpen vezérelt kapuk kimeneteit nem szabad egymással összekötni** a kimenetek közötti zárlati áramok veszélye miatt. Az O.C. kimenetű kapuba a T_3 tranzisztort (és annak járulékos elemeit) nem építik be (4.21. ábra), a munkaellenállást (R_K) külsőleg kell az áramkörhöz csatlakoztatni. R_K nélkül a kapu csak az U_{ki0} jelet tudja létrehozni. Az U_{ki1} az R_K ellenálláson jut a kimenetre. Az R_K értékét a T_3 tranzisztorra maximált áram korlátozza.

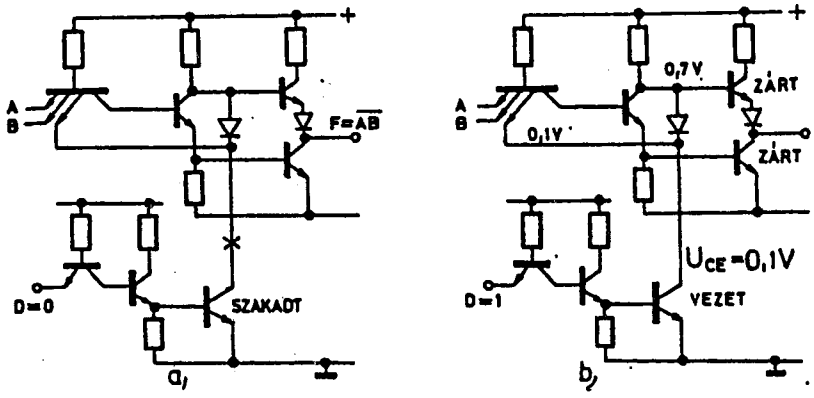


4.21. ábra

Az O.C. kimenetű kapuk kimenetei **egymással összeköthetők egy közös R_K** ellenállás beiktatásával. Ilyenkor az egyes kapuk **T_3 tranzisztorai párhuzamosan** kapcsolódnak. Az ilyen kapcsolást wired-AND (huzalozott ÉS) kapcsolásnak hívják. A kimeneti függvény: $F = \overline{AB \vee CD}$, ahol A és B az egyik, C és D a másik kapu bemenetére kapcsolt logikai változó. A R_K ellenállás értékét számítással vagy táblázatból határozhatjuk meg az összekötött kimenetű kapuk számának és a kimenetre csatlakozó bemenetek számának függvényében. Megjegyezzük, hogy a nyitott kollektoros kapukat a huzalozott ÉS funkción túlmenően igen gyakran LED-ek, lámpák, relék meghajtására, illetve szintillevőként alkalmazzák. A 7405 típusú IC 6 db O.C. kimenetű invertert tartalmaz.

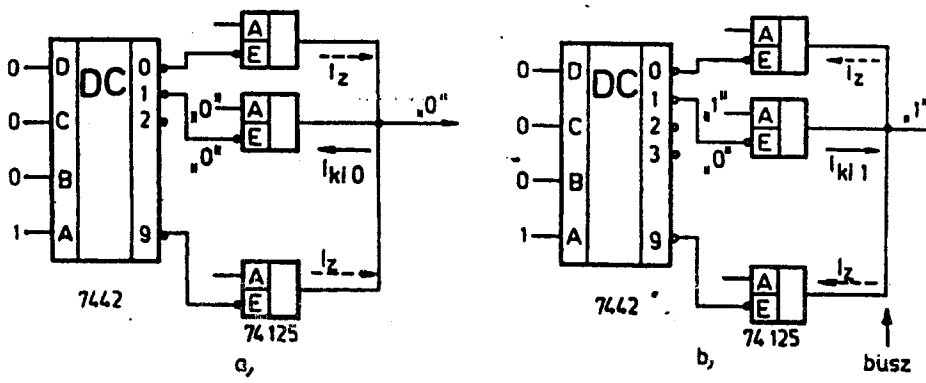
c) Háromállapotú kimenet (T.S.)

Láttuk az O.C. kimenetű kapuk összeköthetőségéből származó előnyöket. Ugyanakkor az O.C. kimenetű kapuk működési sebessége elmarad a T.P. kimenetűhöz képest a KΩ nagyságrendű R_K és a szórt kapacitás miatt. Az O.C. és a T.P. kimenetű kapuk előnyeit egyesíti a **háromállapotú kimenet** (T.S.). A három állapot: logikai 0, logikai 1 és nagyimpedanciás állapotot. Utóbbit egy külön „bénító" (Disable) bemenet vezérlésével lehet előidézni a 4.22. ábra szerint.



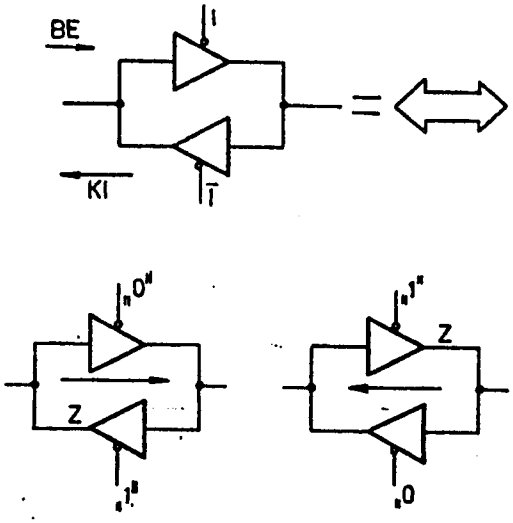
4.22. ábra

A kapcsolás működése az ábra alapján követhető. Ha D=0, a kapu **egy T.P. kimenetű NAND** elemként működik. D=1 esetén mindkét kimeneti tranzisztor zárt (nagyimpedanciás állapot), s a kimenetre egy másik, aktív T.S. kimenetű köthető, és ez hozza létre a kimeneti értéket (0 ill. 1). **Összekötött kimenetű T.S. kapuk közül mindig csak egy lehet aktív,** a többit bénítani kell a zárlati áramok elkerülése céljából. A 4.23. ábrán a T.S. kapuk engedélyezését egy dekódoló végzi. Az ábrán a közös sínen fellépő áramokat is feltüntettük. Az áramkör egy 8/1-es MUX-ként működik.



4.23. ábra

A háromállapotú elemek az ún. **sínrendszerek** (bus system) alapját képezik. A sínrendszerek a programozható logikák (μP, PLC, stb.) elengedhetetlen elemei. A 4.23. ábrán kialakított közös adatvezeték tulajdonképpen a sínrendszer 1 bitje. Megjegyezzük, hogy sínrendszert MUX-ok ill. O.C. kapuk felhasználásával is lehet készíteni, de a T.S. kimenetű a legelőnyösebb. A **sínrendszer** lehet **egy** ill. **kétirányú** az információ áramlás irányától függően. Az adatok mozgatásánál általában kétirányú sínre van szükség. **Kétirányú buszvonalat** a T.S. elemek 4.24. ábra szerinti kapcsolásával készítenek. Mint az ábrából kitűnik, az egyik, elem mindig passzív állapotban van. Az így kialakított kapcsok a vezérléstől függően be- ill. kimenetenként viselkednek. Az ábra szerinti kapcsolás az **adatsín** egy bitjeként funkcionál. Eszerint a 8 bites adatsín 8 darab ilyen kapcsolást (16 db T.S. elemet) tartalmaz.

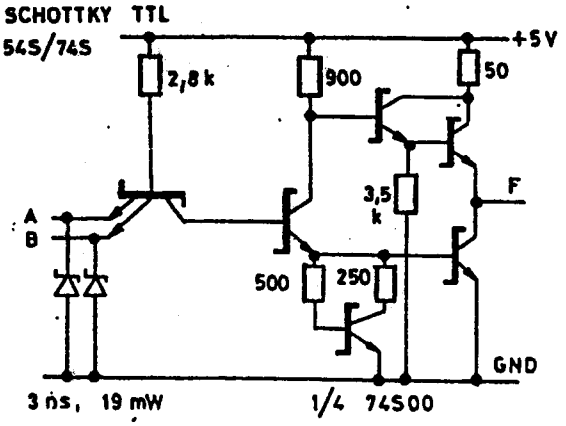


4.24. ábra

4.3.1.2. TTL családok

a) Schottky TTL család

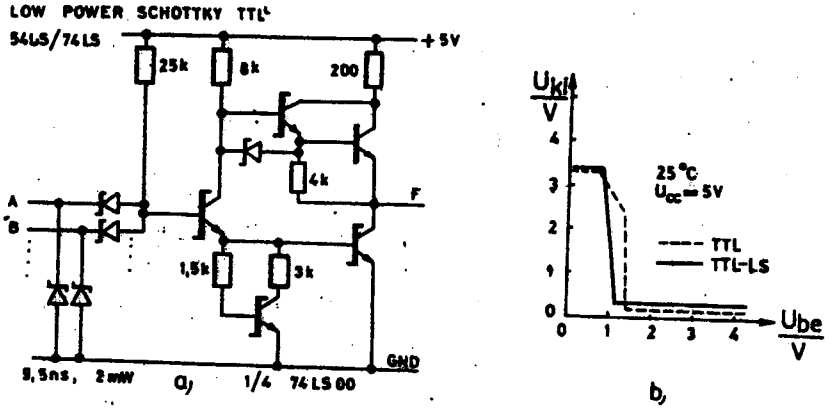
A Schottky TTL család alapeleme a 4.25. ábra szerinti kapcsolású. Az S sorozat terjedési ideje 3 ns körüli, amit a Shottky diódával ellátott tranzisztorokkal érik el. Mivel a Schottky dióda nyitó irányú feszültségesése (0,4 V) kisebb, mint a C-B diódáé (0,6 V), a tranzisztorok bekapcsolásakor a túlvezérlő bázisáram a Schottky diódán folyik el, így a **tranzisztorok nem kerülnek telítésbe.**



4.25. ábra

b) Kis fogyasztású Schottky TTL (TTL-LS)

Ez az univerzális célra legjobban felhasználható TTL változat. Fogyasztása kapunként mindössze 2 mW, késleltetési ideje ~ 9,5 ns. Az LS NAND kapu bemeneti fokozata nem multiemitteres, hanem Schottky diódás ÉS kapu. Az LS sorozatú NAND kapu felépítése és transzfer karakterisztikája a 4.26. ábrán látható.



4.26. ábra

Mind az S, mind LS sorozat el van látva bemeneti, ún. reflexió elleni diódákkal. **Reflexió esetén ugyanis negatív feszültségcsúcsok** jelenhetnek meg a kapu bemenetein.

4.3.1.3. Terhelés és terhelhetőség

A TTL rendszerben az egységterhelések:

$$I_{e"0"} = -1{,}6 \text{ mA ill. } I_{e"1"} = 40\ \mu A.$$

A terhelhetőség (F.O.) tehát

$$F.O._{"0"} = \frac{I_{ki0max}}{I_{e"0"}} \text{ill.} F.O._{"1"} = \frac{I_{ki1max}}{I_{e"1"}}.$$

Normál TTL-nél a terhelhetőség:

$$F.O._{"0"} = \frac{-16\text{mA}}{-1{,}6\text{mA}} = 10 \text{ ill. } F.O._{"1"} = \frac{400\ \mu A}{40\ \mu A} = 10.$$

Gyakran előfordul, hogy egyetlen berendezésen belül többféle sorozat elemeit építik be (pl. normál, LS, S, AS, stb.). **Az egyes TTL sorozatok egymással kompatibilisek a feszültségek vonatkozásában,** de a **be- és kimeneti áramok tekintetében eltérőek.** Ezért a fenti F.I. és F.O. értékkel csak azonos sorozatú elemekből épített hálózatok esetén számolhatunk. Eltérő esetben a gyártó cég katalógus adatait kell használni. Példaként a 4.4. táblázatban a normál, a Schottky ill. az LS sorozatú elemek be és kimeneti áramait adtuk meg.

4.4. táblázat

Sorozat	Max. bemeneti áron		Max. kimeneti áron	
	I_{be0max}	I_{be1max}	I_{ki0max}	I_{ki1max}
Normál	- 1,6 mA	40 μA	-16 mA	400 μA
LS	- 0,36 mA	20 μA	- 8 mA	400 μA
S	- 2 mA	50 μA	- 20 mA	1 mA

4.3.1.4. Alkalmazástechnikai ajánlások

A következőkben a TTL rendszerekre vonatkozóan néhány alkalmazási ajánlást ismertetünk.

a) Nem használt bemenetek

Ilyen bemenetekről beszélünk, amikor nem használjuk fel az elem valamennyi bemenetét. Készülékekben **nem szabad a bemeneteket szabadon hagyni,** mert ez az áramkör **terjedési idejét,** zavarérzékenységét növeli. Az ilyen bemeneteket stabil feszültségre kell kapcsolni. Az AND, NAND kapuk bemeneteit U_1, az OR, NOR kapuk bemeneteit pedig U_0-ra.

b) Áramellátás

A TTL rendszer működtetéséhez **stabilizált tápegység** szükséges. Legalább ilyen fontos az alacsonyfrekvenciás szűrés 5-10 µF-os tantál kondenzátorokkal.

c) Áramcsúcsok csökkentése

A TTL kapu átváltásakor 10 mA nagyságrendű és 6 ns időtartamú áramcsúcsok jelennek meg a tápvezetéken. Ezt **kis impedenciájú tápvezetékkel** és hidegítő kondenzátorokkal lehet csökkenteni.

d) Reflexió

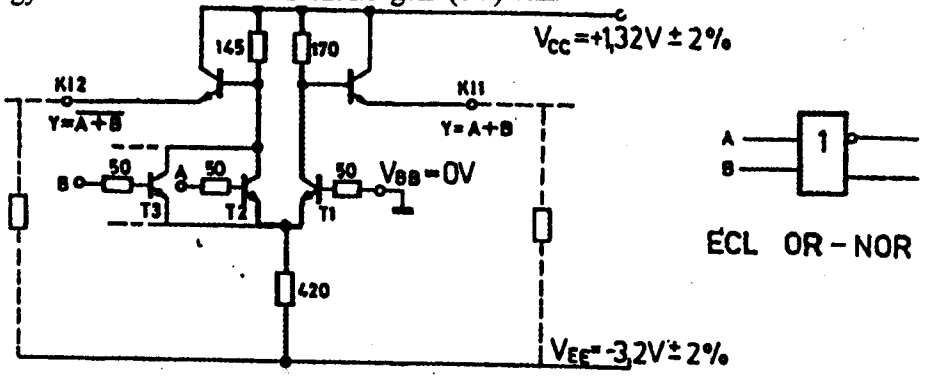
A TTL kimenetek aszimmetriája miatt hosszú vezetékeken **reflexiók** léphetnek fel, amelyek **negatív feszültséget** is eredményezhetnek a bemeneten. Ez ellen a jelvezeték rövidítésével, védődiódákkal ill. kábel adó-vevő áramkörökkel védekezhetünk.

e) Külső zavarok

A külső zavarok **elektromos** (áthallás), **mágneses** eredetűek lehetnek. Emellett ügyelni kell a digitális rendszer és a teljesítmény fokozat szakszerű **földelésére gyűjtősín** kialakítása révén.

4.3.2. Emitter csatolású logika (ECL)

Az emittercsatclású áramkörök jelenleg a legnagyobb sebességű digitális áramkörök. Az ECL áramkörök felépítése olyan, hogy a bennük lévő nagyfrekvenciás tranzisztorok nem vezérlődnek telítésbe a működés során, ezért **nem lép fel töltéstárolási effektus**. Az ECL család alapeleme az OR/NOR elem (4.27. ábra) Az áramkör egy bemeneti **differenciál erősítőt** tartalmaz, amelynek egyik bemenete állandó feszültségen (0V) van.



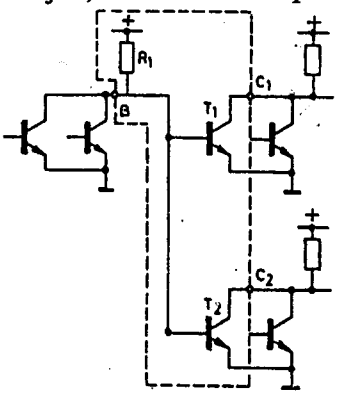
4.27. ábra

Ha az A bemenetre pozitív (néhány tized V-os) logikai 1 feszültséget kapcsolunk, akkor a T_2 kinyit, a közös emitter potenciál megemelkedik, minek

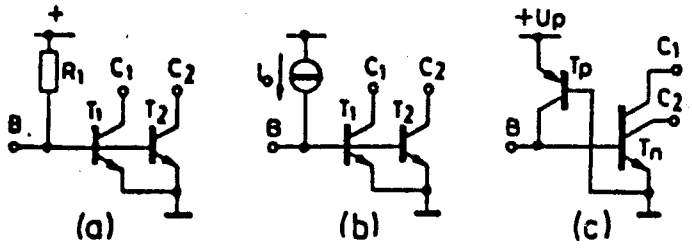
következtében T_1 lezár (bázisa 0 V-on van). Az emitter ellenállás árama T_2-n folyik keresztül, így munkaellenállásán a feszültség lecsökken és a KI1 emitterkövető kimeneten pozitív logikai 0 feszültség jelenik meg. Ugyanakkor T_1 zárt lévén, munkaellenállásán nem esik feszültség, így a KI_1 emitterkövető kimeneten pozitív logikai 1 szint jelenik meg. Amikor az A bemenetre néhány tized V-os negatív feszültség (U_0) kerül, akkor T_2 lezár, mert bázisa negatívabb, mint a T_1 tranzisztoré, T_1 nyit és a közös emitter ellenállás áramát T_1 vezeti el. A KI_1-en logikai 0, a KI_2-n logikai 1 szint jelenik meg. A T_1 és T_2 tranzisztor úgy működik, **mint egy áramutas kapcsoló, amely az emitter ellenállás áramát egyik vagy másik** irányba tereli. Az ECL logika legnagyobb előnye a nagy sebesség, hátránya az alacsony zajtartalék, az alacsony logikai szintek, a gerjedékenység. Ezért ECL áramköröket csak ott célszerű alakalmazni, ahol erre a sebességre feltétlenül szükség van: pl. nagy frekvenciás osztók első dekádja ECL, a második már TTL is lehet. Ehhez ECL-TTL ill. TTL-ECL szinttevők állnak rendelkezésre.

4.3.3. I^2L logika

Az **integrált injekciós logika** (I^2L) a hagyományos DCTL áramkörből származtatható. Ha a 4.28. ábrán szaggatott vonallal jelölt hálózatot a 4.29. ábrán vázoltak szerint átalakítjuk, akkor az I^2L alapelemet kapjuk.

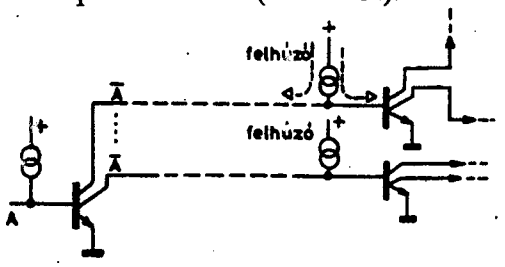


4.28. ábra



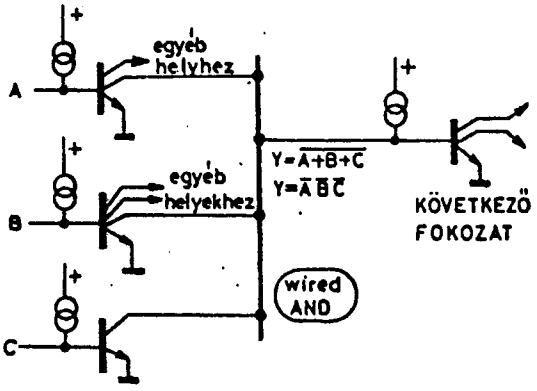
4.29. ábra

Az R1 ellenállás ugyanis egy **áramgenerátorral** helyettesíthető (4.29.b) ábra). Áramgenerátorként pedig egy pnp tranzisztort előnyösebb alkalmazni (c) ábra). A pnp tranzisztor, melynek emittere a néhány tized V-os pozitív tápfeszültségre kapcsolódik áramgenerátorként bázisáramot injektál (innen az elnevezés) az npn **„többkollektoros"** tranzisztor bázisába. Az injektált bázisáram miatt az npn tranzisztor kinyit, és kollektorai gyakorlatilag földpotenciálra kerülnek úgy, mint a hagyományos inverter esetében. A kollektorok munkaellenállásai az ezekhez csatlakozó ugyanilyen szerkezetű I^2L kapuk bemeneti pnp áramgenerátorai, amelyek igyekeznek „felhúzni" a kollektor potenciált. Azonban, ha a meghajtó kollektor földpotenciában van, akkor a rákapcsolódó következő fokozat áramgenerátorok árama ebbe a kollektorba folyik és nem nyitja ki a hozzátartozó npn tranzisztort (4.30. ábra).



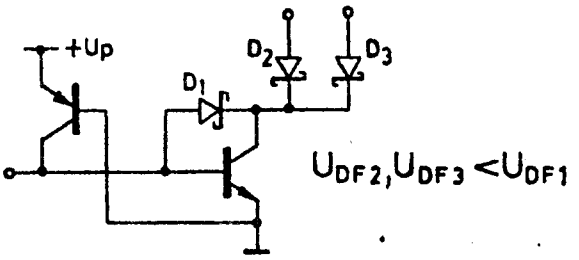
4.30. ábra

A logikai kapcsolatok a megfelelő nyitott kollektorok összekötése révén wired-**AND** logikával képződnek. Az összekötött kollektorok munkaellenállása az összekötési pontokra csatlakozó következő I^2L kapu áramgenerátora (4.31. ábra).



4.31. ábra

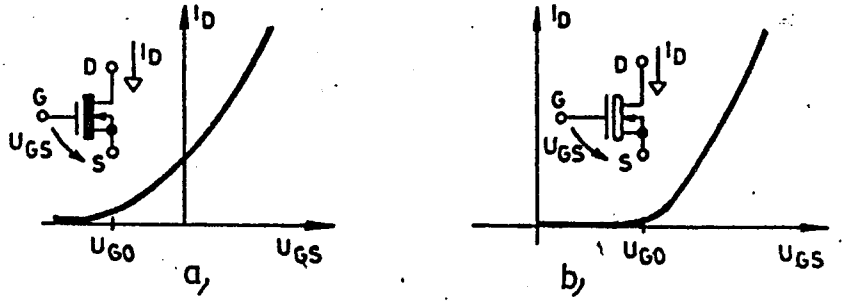
Az I^2L logikai előnyei: igen kis helyigény, 10^3-10^4 kapu egy chipen, igen alacsony tápfeszültség (~ 1,5 V), igen kis disszipáció-késleltetési idő szorzat. A jelterjedési idő Schottky diódák ill. tranzisztorok alkalmazásával tovább csökkenthető. Az I^2L-Schottky alapelem a 4.32. ábrán látható.



4.32. ábra

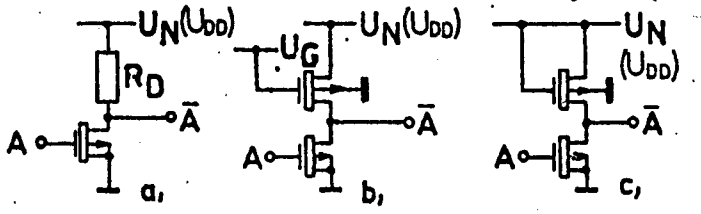
4.3.4. Digitális MOS áramkörök

A MOS áramkörök alapeleme a MOS (Metall-Oxid-Semiconductor) tranzisztor. A MOS tranzisztor lehet **növekményes** vagy **elzáródásos** (kiürítéses) típusú. Az elzáródásos MOS tranzisztor zérus vezérlő feszültség esetén is vezet 4.33.a) ábra, így nyitásához és zárásához **kétféle polaritású** feszültséget igényel. A **növekményes** MOS tranzisztor zérus vezérlőfeszültség esetén nem vezet 4.33.b) ábra, így nyitásához és zárásához **egyféle feszültség** elegendő. Emiatt a digitális MOS áramkörökben növekményes típusú MOS tranzisztorokat alkalmaznak.



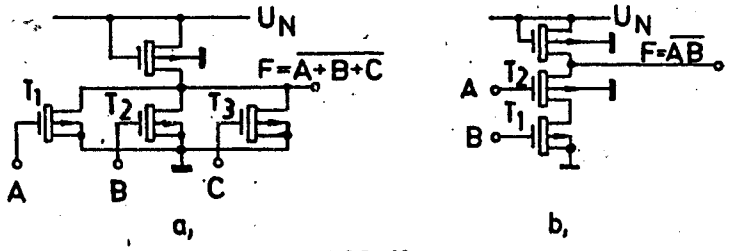
4.33. ábra

Gyártástechnológiai okok miatt a p csatornás tranzisztorok alkalmazása terjedt el. A p csatornás növekményes MOS tranzisztoros inverter munkaellenállását 100 KΩ fölé választják a kis fogyasztás érdekében. A MOS inverterek munkaellenállását rendszerint MOS tranzisztorokból alakítják ki (4.34. ábra).



4.34. ábra

NOR kaput a MOS tranzisztorok párhuzamos, NAND kaput pedig soros kapcsolásával készítenek a 4.35. ábra szerint.



4.35. ábra

Megjegyezzük, hogy a MOS kapuáramkörök többnyire csak az LSI áramkörök részeként fordulnak elő, önállóan nem.

A MOS digitális áramkörök előnyei:

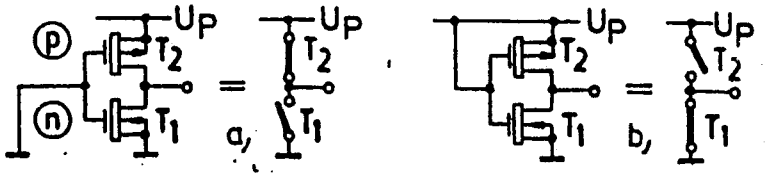
- kis teljesítmény fogyasztás
- igen jól integrálhatók
- igen nagy d.c.F.O.

Hátrányai:

- a MOS elemek lassúbb működésűek a TTL-nél
- más rendszerekkel csak szintáttevők révén csatlakoztathatók
- bemeneteik a sztatikus feltöltődásre érzékenyek, ezért tárolásuk és forrasztásuk esetén erre ügyelni kell!

4.3.4.1. CMOS logikai áramkörök

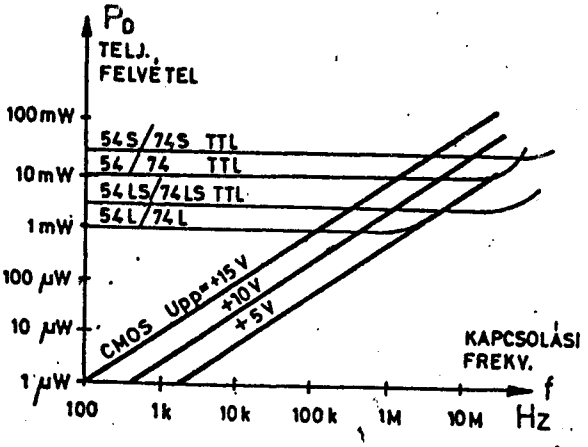
A p és n csatornás MOS tranzisztorokat alkalmazva kiváló tulajdonságú digitális áramkörök készíthetők. Az ilyen áramkörök szokásos elnevezése CMOS (Complementary MOS), ami a komplementer jellegű felépítésre utal. A CMOS inverter kapcsolása látható a 4.36. ábrán mindkét vezérlési állapotában. A GATE elektródák azonos vezérlést kapnak, de komplementer tranzisztorok lévén ellenütemben működnek.



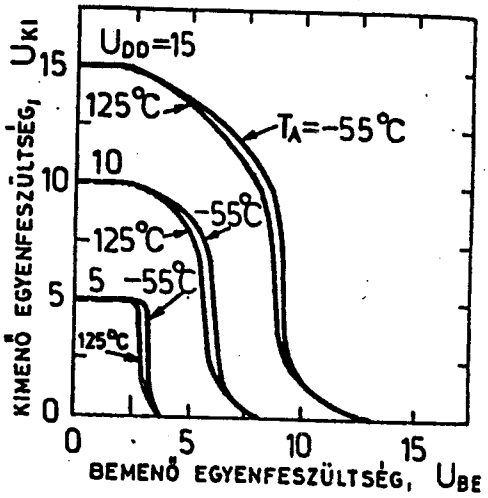
4.36. ábra

Ha az inverter bemenetére $U_0 = 0$ V kerül, az n csatornás T_1 lezár, a p csatornás T_2 pedig nyit, mert GATE elektródája jóval negatívabb az S elektródánál.

A kimenetre $U_1 = U_p$ kerül a) ábra. Ha a CMOS inverter bemenetére $U_1 = U_p$ van kapcsolva, akkor T_2 lezár, a vezető T_1 pedig a kimenetet föld potenciálra kapcsolja b) ábra. A CMOS inverter működéséből következik, hogy az **egyik tranzisztor mindig zárt állapotban** van, ezért az áramfogyasztás statikus üzemben közel zérus (nA). A felvett teljesítmény nW nagyságrendű. Dinamikus üzemben 0→1 ill. 1 →0 átváltáskor egy rövid időre mindkét tranzisztor vezet ill. a terhelő kapacitásokat fel kell tölteni. A CMOS áramkörök fogyasztása tehát frekvencia függő és a **MHz-es tartományban eléri a TTL áramkörökét** (4.37. ábra). A CMOS áramkörök szimmetrikus felépítése következtében a transzfer karakterisztika nagyon közel van az ideálishoz, a komparálási szint közelítőleg a **tápfeszültség felével** egyenlő. Ezt szemlélteti a 4.38. ábra.

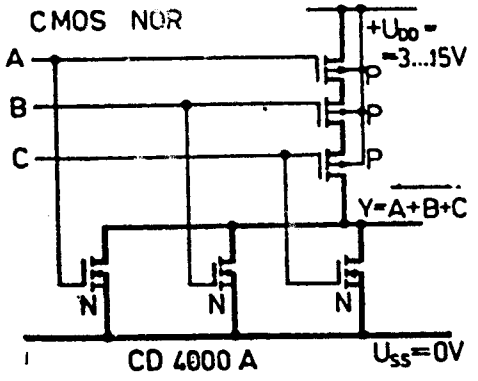


4.37. ábra

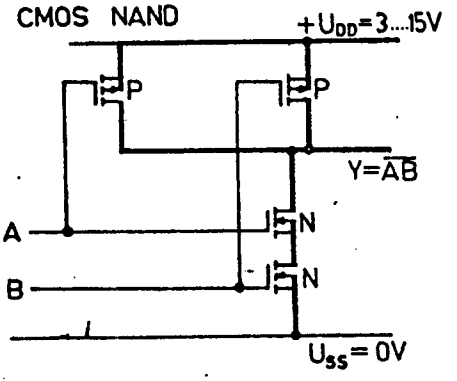


4.38. ábra

A CMOS áramkörök **zajvédettsége** gyakorlatilag a **tápfeszültség felével** azonos. Emiatt az ipari vezérlő berendezésekben is alkalmazzák. A CMOS **NOR** kapu felépítése a 4.39. ábra szerinti. A bemenetekkel azonos számú pMOS tranzisztor sorba, míg a bemenetekkel azonos számú nMOS tranzisztor párhuzamosan van kapcsolva. A CMOS **NAND** kapu esetén az NMOS tranzisztorok sorba, a pMOS tranzisztorok párhuzamosan vannak kapcsolva a 4.40. ábra szerint.



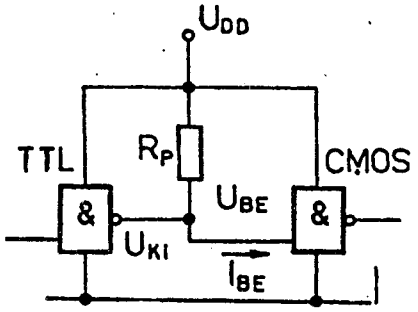
4.39. ábra



4.40. ábra

A gyárilag forgalmazott NAND ill. NOR elemeket bufferelt kimenetekkel látják el. A CMOS logikai áramkörök sokkal előnyösebb tulajdonságúak mint a TTL. A CMOS elemek bemeneteit mindig fix potenciálra kell kapcsolni: a NAND elemét U_1-re, a NOR elemét U_0-ra. A 4.41. ábrán TTL kapu hajt meg CMOS kaput. Az R ellenállás az U_{ki1}-et növeli. Erre azért van szükség mert a TTL kapu U_{ki1}min értéke (2,4 V) a CMOS kapu komparálási értékének közelében van (Uk=5 V/2), de nem éri el az U_1min értékét. A legjobban elterjedt CMOS

áramkörök a CD 4000-es (RCA) ill. a 74C sorozat. Utóbbi a TTL sorozattal azonos lábelrendezésű.



4.41. ábra

Foglaljuk össze a CMOS áramkörök előnyeit:

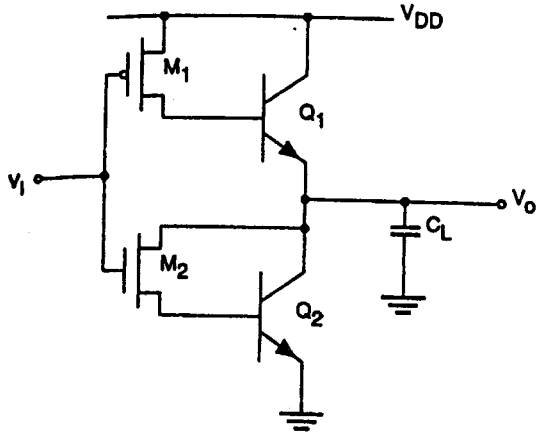
- kis teljesítmény felvétel (10 nW/kapu)
- nagy zajvédettség (kb. a tápfeszültség fele)
- üzembiztos működés +3...+15 V tápfeszültség értékek között
- kis kimeneti impedancia ($R_{ki} < 1K\Omega$)
- igen nagy d.c.F.O.

Hátrányai:

- fogyasztása frekvencia függő
- TTL-hez kapcsoláshoz illesztőre van szükség
- statikus feltöltődéstől óvni kell.

4.3.4.2. BiCMOS logikai áramkörök

A bipoláris tranzisztorok gyorsabb működésének és a CMOS technika előnyeinek integrálását a BiCMOS áramkörök valósítják meg. A BiCMOS alapinverter kapcsolása a 4.42. ábrán látható.



4.42. ábra

Amikor $V_i = V_L$, akkor az M_1 tranzisztor vezet, M_2 zár, s a kimeneten „1" jelenik meg. Amikor a $V_i = V_H$, M_1 és Q_1 zárt, M2 és Q_2 vezet, így a kimeneten „0" jel jelenik meg.

A BiCMOS logikai áramkörök előnyei:

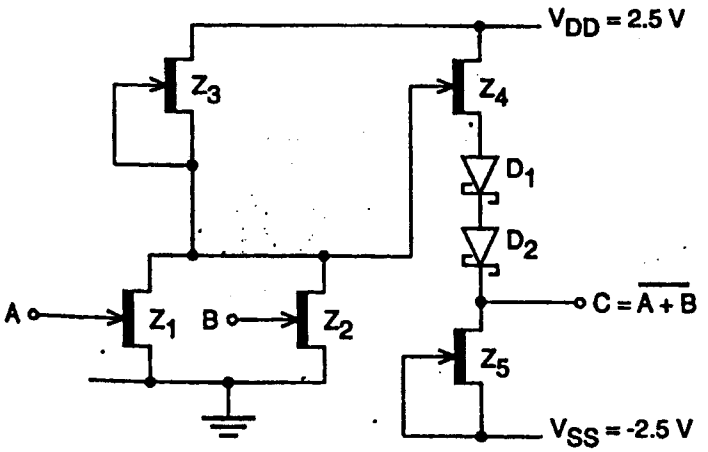
- a max. frekvencia magasabb, mint a CMOS esetén
- a bemeneti impedancia nagy
- a tápfeszültség alacsony értékű
- a kimeneti terhelhetőség nagy.

4.3.4.3. GALLEUM-ARZENID logikai áramkörök (GaAs)

A GaAs integrált áramkörök olyan n csatornás MOSFET (MESFET) áramkörök, amelyeknél az alapréteg (substrate) galleum-arzenid felépítésű. A GaAs szubsztrát réteg ellenállása $10^6 \sim 10^8$ Ωcm. Ez nagy előnye a Si alapréteghez képest, mert lecsökkenti az elektróda-alapréteg kapacitást. Három féle GaAs logikai áramkör terjedt el:

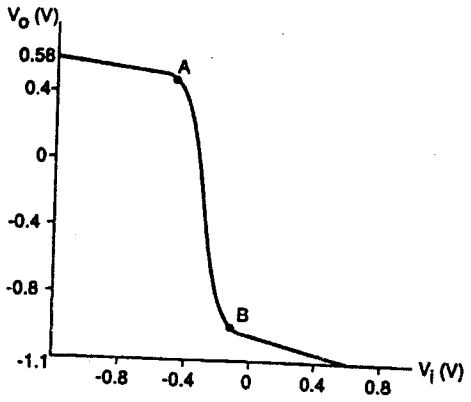
- bufferelt FET logika (BFL)
- Schottky diódás FET logika (SDFL)
- direkt csatolt FET logika (DCFL).

Egy **BFL NOR kapu** látható a 4.43. ábrán. A $V_H = 0{,}6$ V, a $V_L = -1{,}2$ V. A kapcsolás két tápfeszültséget igényel: ± 2,5 V.

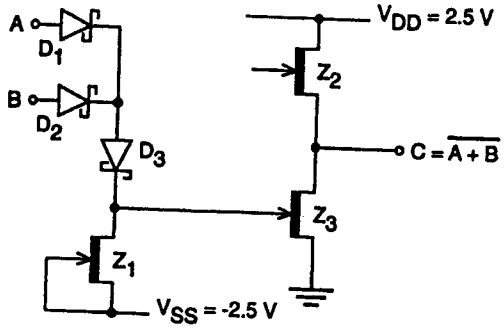


4.43. ábra

Az áramkör transzfer karakterisztikája a 4.44. ábra szerinti.

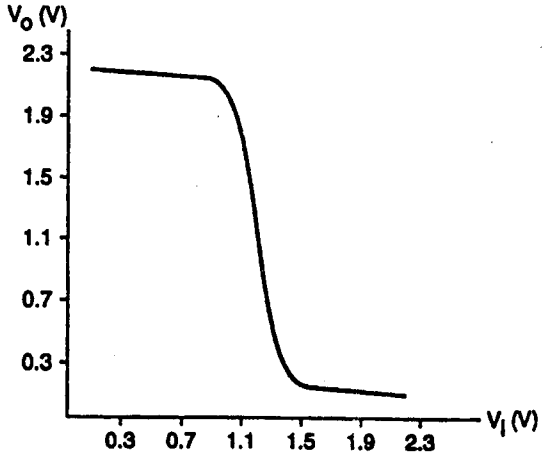


4.44. ábra



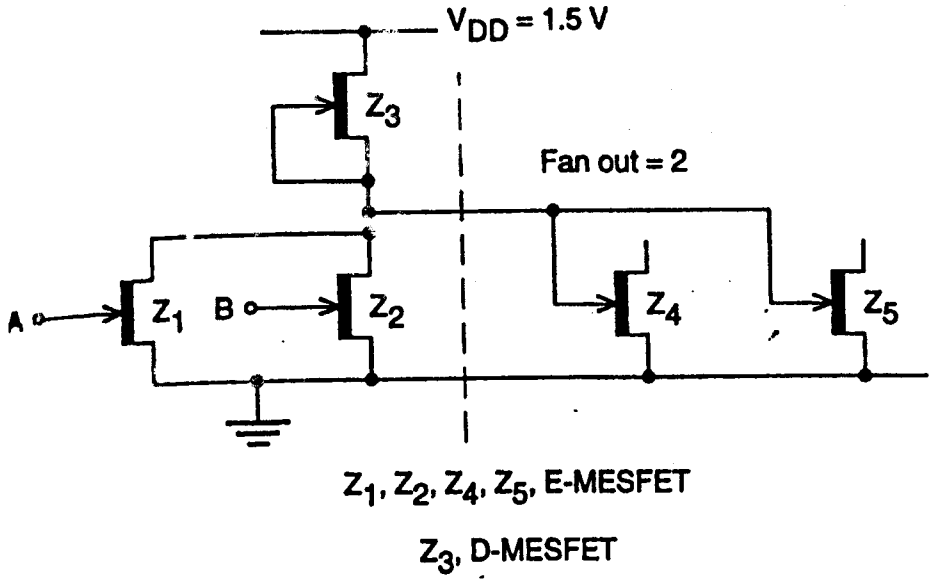
4.45. ábra

Az **SDFL kapu** (4.45. ábra) szintén tartalmaz Schottky szinteltoló diódákat, de a bemeneti oldalon. Ez a bemeneti fokozat hasonlít a TTL-LS kapuéhoz. A transzfer karakterisztika a 4.46. ábrán található.



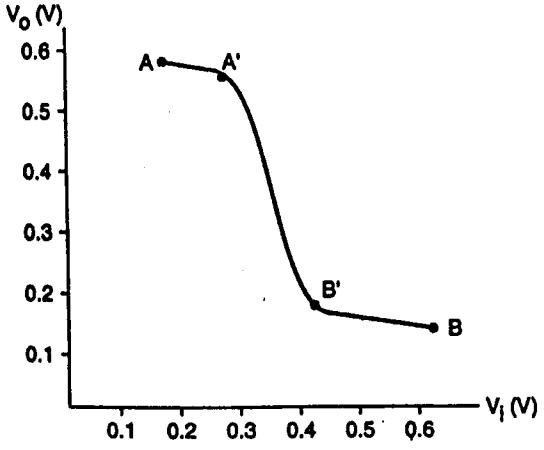
4.46. ábra

A **DCFL NOR kapu** az első és legegyszerűbb felépítésű GaAs logika. Kapcsolása a 4.47. ábrán látható. A bemeneti fokozat láthatóan a két párhuzamosan kapcsolt Z_1, Z_2 tranzisztorokkal valósítja meg a NOR kapcsolatot és két tranzisztor szolgál a kimenetek előállításához (Z_4, Z_5).



4.47. ábra

Az áramkör transzfer karakterisztikája a 4.48. ábrán látható.



4.48. ábra

A három GaAs áramkör típus összehasonlítását a 4.5. táblázatban adtuk meg.

4.5. táblázat

BFL	SDFL	DCFL
A legnagyobb sebesség és fan out.	Közepes sebesség és alacsony fan-out.	Közepes sebesség és alacsony fan-out.
A legnagyobb kapunkénti bemeneti terhelés.	Közepes kapunkénti bemeneti terhelés.	A legalacsonyabb bemeneti terhelés kapunként.
Két tápfeszültség.	Két tápfeszültség.	Egy tápfeszültség.
Jó zajvédettség (~ 0,7 V).	Jó zajvédettség (~ 0,7 V).	Alacsony zajvédettség (~ 0,2 V).
Kapu terjedési idő: (~ 80 ps).	Kapu terjedési idő: (~ 200 ps).	Kapu terjedési idő: (~ 100 ps).

4.3.5. Összehasonlító táblázatok

A következőkben az egyes áramkörtípusok összehasonlító adatait adjuk meg. Hőmérséklet és tápfeszültség adatok az SSi/MSi áramkörökhöz (4.6. táblázat).

4.6. táblázat

Család	Hőhérséklet tartomány (^{o}C)	Névleges tápfeszültség (V)	Tápfeszültség tartomány (V)
TTL (valamennyi sorozat)			
54	M	5,0	4,5-től 5,5-ig
74	I	5,0	4,5-től 5,5-ig
CMOS (valamennyi sorozat)			
54	M	5,0	2,0-tól 6,0-ig (AC, HC)
74	C	5,0	4,5-től 5,5-ig (ACT, ACT)
BICMOS			
54	M	5,0	4,5-től 5,5-ig
74	C	5,0	4,5-től 5,5-ig
ECL 10KH	0-tól 75	- 5,2	4,94-től - 5,46-ig
ECL 10K	- 30-tól +85	- 5,2	4,68-től - 5,72-ig
MECL III	- 30-tól +85	- 5,2	4,68-től - 5,72-ig
LV-CMOS	C	3,3	1,0-tól - 3,6
CMOS-LVC	C	3,3	1,2-től - 3,6
CMOS-ALVC	C	3,3	1,2-től - 3,6
BICMOS LVT			
54	C	3,3	1,2-től - 3,6
74	M	3,3	1,2-től - 3,6

Jelölések:

- C: 0 °C - ... 70 °C (általános igény)
- I: - 40 °C + ... 85 °C (ipari igény)
- M: - 55 °C + ... 125 °C (katonai igény).

A TTL ill. ECL kapu terjedési idő, bemeneti teljesítmény és számlálási frekvencia összehasonlító táblázata (4.7. táblázat).

4.7. táblázat

Sorozat	Tp (ns)	P_D (mW)	Sebesség-teljesítmény szorzat (pJ)	Frekvencia MHz
TTL LS	9,5	2	19	45
TTL S	3,0	19	57	125
TTL ALS	4,0	1	4	50
TTL AS	1,5	20	30	200
ECL 10K	2,0	25	50	125
ECL 10KH	1,0	25	25	250
MECL III	1,0	60	60	500
ECL in PS	0,35	39	14	1400

CMOS áramkörök terjedési idő, bemeneti kapacitás és számlálási frekvencia szerinti összehasonlító táblázata (4.8. táblázat).

4.8. táblázat

Család	tp (ns)	Ci (pF)	Frekvencia (MHz)
HC/HCT	7,5	10	50
AC/ACT	4,8	4,5	160
LV-CMOS	9,0	3,5	
LVC	4,0	3,5	125
HLL, ALVC	2,1	3,0	350

A CMOS, BICMOS és TTL áramkörök maximális kimeneti áramait a 4.9. táblázat foglalja össze (4.9. táblázat).

4.9. táblázat

Család	IoH (mA)	IoL (mA)
CMOS HC/HCT	- 8	4
CMOS AC/ACT	- 24	24
BICMOS ABT	- 32	64
BICMOS LVT	- 32	64
LV-HCMOS	- 6	6
LVC, HLL, ALVC	- 24	24

TTL LS	- 0,4	24
TTL ALS	- 0,4	8
TTL AS	- 2	20
TTL S	-1	20

Maximális TTL bemeneti áramok

4.10. táblázat

TTL család	I_{iH} (μA)	I_{iL} (mA)
LS	20	- 0,4
S	50	- 2
ALS	20	- 0,1
AS	20	- 0,5

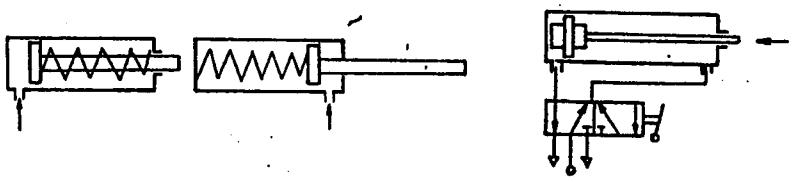
A logikai áramkörök egyik legfontosabb jellemzője az ún. **jósági tényező**, amely a terjedési idő és teljesítményfelvétel szorzata:

$$Sp = Tpd \cdot P_D .$$

A jósági tényező annál kedvezőbb, minél alacsonyabb (ld. 4.7. táblázat). Megjegyezzzük, hogy az ún. programozható logikai áramköröket a 8. fejezetben ismertetjük.

4.4. Pneumatikus logikai elemek

A logikai kapcsolatok nemcsak villamos, hanem más (pneumatikus, hidraulikus) elemekkel is realizálhatók. A pneumatikus rendszerekben az **információt hordozó közeg** a sűrített levegő, melynek **nyomása** vagy **áramlása** képezi az **információt.** Pneumatikus elemeket főként a gépiparban használnak, ahol a kimenő jelet egyenes vonalú mozgással kifejtett erőátvitelre használják. Másik alkalmazási terület a vegyi és rokon iparágak, ahol a **tűz és robbanásveszélyes terek** automatizálásánál alkalmazzák. Napjainkban a gyújtószikramentes villamos eszközök elterjedésével az utóbbi területen a jelentőségük csökken, ezért csak a gépiparban elterjedt útszelepekkel foglalkozunk. A nagy nyomású pneumatikus rendszerek két jellegzetes elemtípusból épülnek fel: az útszelepekből és a munkahengerekből. Egyenesvonalú mozgások létrehozására a legegyszerűbb végrehajtó szerv a **munkahenger.** A munkahengerek lehetnek egy- ill. kétoldali működtetésűek. Az egyoldali működtetésű munkahenger visszatérítését rendszerint rugóerő végzi (4.49. ábra).



4.49. ábra

A munkahengerek működtetését **útszelepek** végzik. Az útszelepek a sűrített levegő áramlásának irányát határozzák meg. A fő áramlási irányok: a táplevegő hálózatból a felhasználási hely felé, valamint a felhasználási helytől az atmoszférába. Az útszelepeket három fő jellemzőjük alapján különböztetjük meg:

- kapcsolási helyzetek száma
- csatlakozóik száma
- működtetési módjuk.

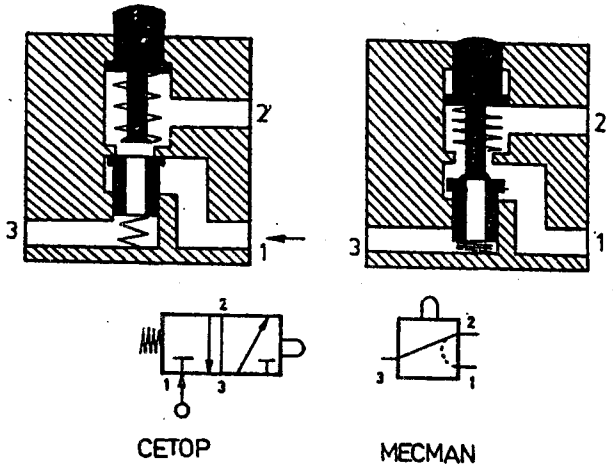
Leggyakrabban a **kétállású** útszelepet használják. Az útszelepek jelölésére a CETOP (Európai Olajhidraulika és Pneumatika Bizottság jelölése terjedt el, de még mindig találkozunk pl. MECMAN jelöléssel is. A CETOP szimbólum a szelepek működését és funkcióját szemlélteti. A **szelep minden működési állapotát egy-egy négyzetnek** (téglalapnak) rajzoljuk és bejelöljük az áramlás irányát. E szerint egy kétállású szelep kettő, a háromállású szelep szimbóluma három négyzetből áll. Ha működtető szervet működtetünk, mindig azt a kapcsolási helyzetet kapjuk, melyet a **vezérlő szerv ábrája melletti négyzetben** tüntettünk fel. A csatlakozó nyílások elhelyezésének nem kell megegyeznie azok valóságos helyzetével.

Csatlakozások számozása:

1:	beömlés (csatlakozás a sűrített levegő hálózatba)
2:	kiömlés, kivezetés a fogyasztóhoz (pl. munkahenger)
3:	a 2-es kivezetéshez tartozó kipufogás
4:	kiömlés, kivezetés a fogyasztóhoz (pl. munkahengerhez)
5:	a 4-es kivezetéshez tartozó kipufogás
10:	vezérlő levegő csatlakozás, amely zárja a normál helyzetben nyitott szelepet 2 utú szelepeknél
12:	vezérlő levegő csatlakozás, amely az 1-2 utat nyitja
14:	vezérlő levegő csatlakozás, amely az 1-4 utat nyitja.

A kétféle jelölést és a háromutú szelepek működését szemlélteti a 4.50. ábra.

A szelepek egy csoportja kitüntetett alaphelyzettel rendelkezik (ezt rugó jelkép mutatja). A működtető jel megszűnte után a szelepek automatikusan az alaphelyzetbe állnak. Aszerint, hogy a táplevegőt a szelep ebben a helyzetben tovább engedi vagy elzárja, megkülönböztetünk **alaphelyzetben nyitott**, ill. **alaphelyzetben zárt** kapcsolási módokat. Az impulzus vezérlésű szelepek nem rendelkeznek alaphelyzettel, a szelep helyzetét a legutóbbi működtetés határozza meg. A 4.51. ábrán a leggyakrabban használt szelepek CETOP jelképét mutatjuk be. Az ábrán a reteszelő szelepeket is feltüntettük. Ebbe a csoportba a visszacsapó, váltó, kétjelműködtetésű és a gyors légtelenítő szelepek tartoznak.



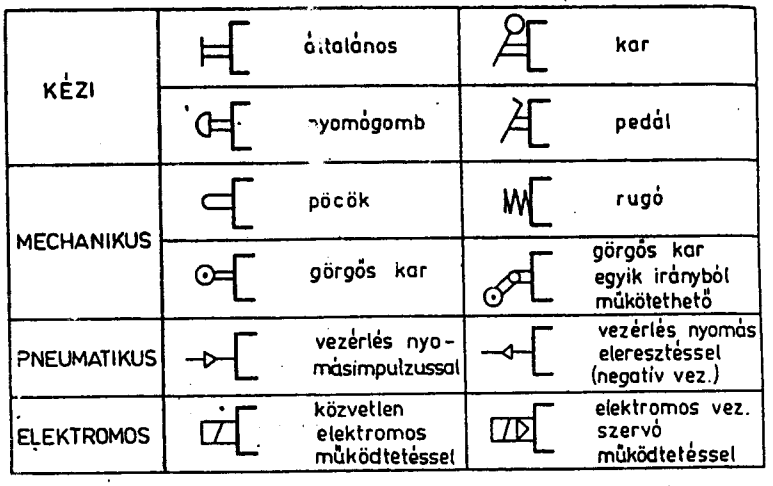
4.50. ábra

ÚTSZELEPEK	2/2-es útszelep alaphelyzetben zárt	2/2-es útszelep alaphelyzetben nyitott
	3/2-es útszelep alaphelyzetben zárt	3/2-es útszelep alaphelyzetben nyitott
	4/2-es útszelep	5/2-es útszelep
	3/3-as útszelep középhelyzetben zárt	4/3-as útszelep középhelyzetben zárt
	5/3-as útszelep középhelyzetben zárt	5/3-as útszelep középhelyzetben nyitott
RETESZELŐ SZELEPEK	váltó szelep (VAGY-elem)	kétjel szelep (ÉS-elem)
	visszacsapó szelep	fojtó visszacsapó szelep
	gyorslégtelenítő szelep	

4.51. ábra

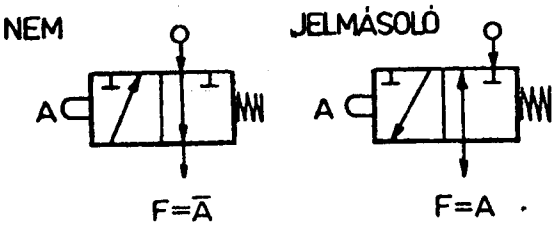
A **visszacsatoló szelepek** az egyik irányban szabad áramlást biztosítanak, míg a másik irányú áramlást megakadályozzák. A **váltószelepek** (kettős visszacsapó szelepek) a logikai VAGY, a kétjelműködtetésű szelepek az ÉS kapcsolatot valósítják meg. A **gyorslégtelenítő szelep** a levegő áramlásának irányától

függően az 1-2 áramlási irányt biztosítja, illetve ellenkező levegőáramlás esetén megnyitja a 2-3 áramlási irányt, így a 2-es csatlakozóról a levegő a 3-as csatlakozón át közvetlenül a szabadba távozik. Ezzel a szeleppel a munkahengerek dugattyúsebességét növelhetjük azáltal, hogy a kipufogó hengerkamra gyors leürítését biztosítja a szelep. Az útszelepek átkapcsolása, azaz a sűrített levegő áramlási irányának kiválasztása a szelep működtetésével történik. A különböző működtetési módokat és azok jelképeit a 4.52. ábrán találjuk.

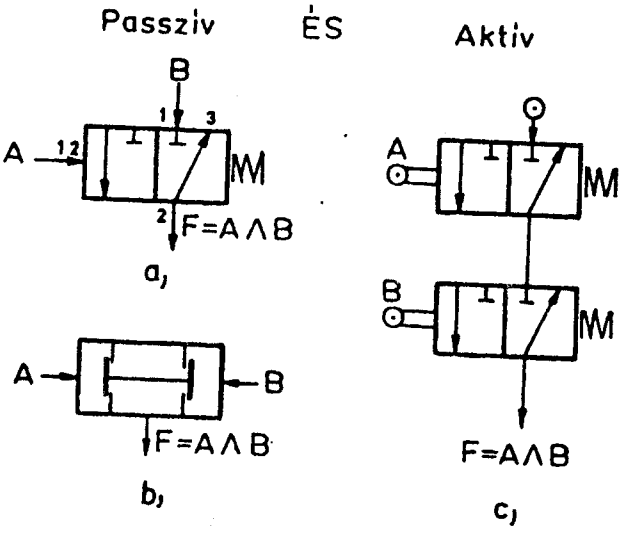


4.52. ábra

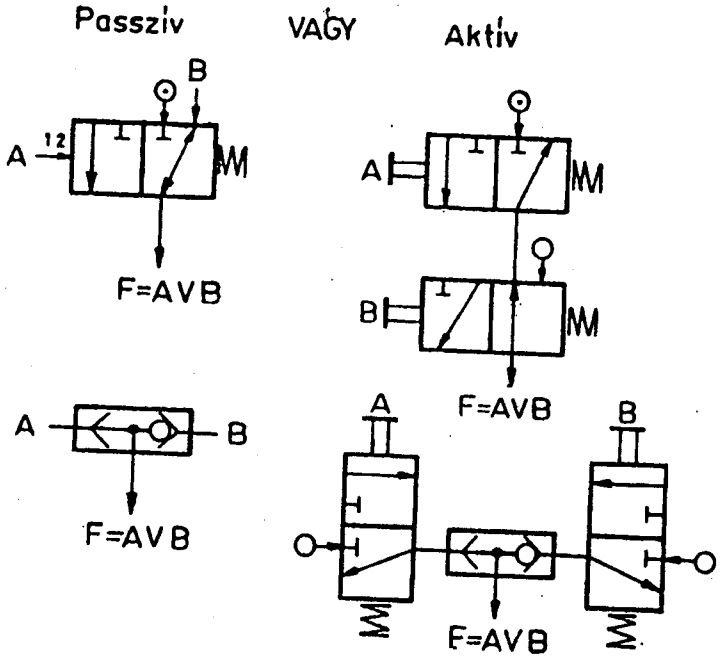
Az útszelepek az érintkezőkhöz hasonlóan sorba ill. párhuzamosan kapcsolhatók. **Párhuzamos kapcsolás esetén a kimeneten váltószelepet** kell beiktatni a káros kipufogás elkerülésére. Az egyes kapcsolások lehetnek **aktívak** és **passzívak** attól függően, hogy a kimenő jelet bevezetett változó (passzív) vagy a tápnyomás (aktív) adja. **A NEM kapcsolatot alaphelyzetben nyitott, a jelmásolást alaphelyzetben zárt szeleppel valósíthatjuk meg** (4.53. ábra). A 4.54. ábra **kétféle passzív** és egy **aktív ÉS** kapcsolást mutat. Utóbbit két jelmásoló soros kapcsolásával realizáltuk c) ábra. **Kétféle passzív** és **két aktív** VAGY kapcsolás látható a 4.55. ábrán. A d) ábra szerinti kapcsolásban váltószelep nélkül A = 0, B = 1 esetén (és fordítva) a táplevegő az alaphelyzetben nyitott szelepen a szabadba távozna.



4.53. ábra

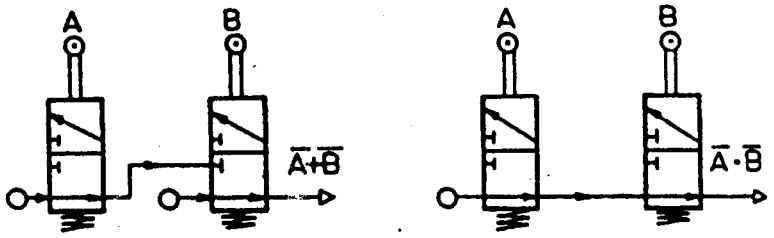


4.54. ábra



4.55. ábra

Hasonlóképpen realizáltuk a NAND ill. NOR kapcsolatot is De Morgan szabályok felhasználásával (4.56. ábra).



4.56. ábra

Felhasznált irodalom

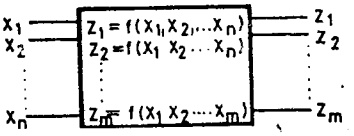
Ajtonyi I: Vezérléstechnika I. J 14-1417 Tankönyvkiadó, Budapest, 1987.
Ajtonyi I: Vezérléstechnika II. J 14-1418 Tankönyvkiadó, Budapest, 1987.
Hainzman: Digitális elektronika. TK. 1973.
L.J. Herbst: Integrated Circuit Engineering. Oxford University Press, 1996.
Simonfalvi-Pótczy-Mayer: Digitális integrált áramkörök alkalmazástechnikája. BME Továbbképző Intézet, 1972.
TEXAS: TTL receptek. MK. 1978.
TTL Integrated Circuits from Texas Instruments.
CMOS Catalog Texas Instruments.

5. KOMBINÁCIÓS HÁLÓZATOK

A digitális rendszerek egy részét az jellemzi, hogy a rendszer állandósult állapotában annak **kimenő jelét** kizárólag a **bemenetre kapcsolt** jelkombináció határozza meg. Tehát **ugyanazon bemeneti kombinációhoz ugyanazon kimeneti esemény tartozik**. Az ilyen logikai rendszert **kombinációs hálózatnak** nevezzük.

A kombinációs hálózatok alapvető jellegzetessége, hogy **a bemeneti feltételek egyértelműen meghatározzák a kimeneti eseményeket,** de ez fordítva nem áll fenn. Eszerint az eddigiekben megismert kapuáramkörök elemi kombinációs hálózatok.

Egy n bemenetű $(X_1, X_2, \ldots, X_n)$ és m kimenetű $(Z_1, Z_2, \ldots, Z_m)$ kombinációs hálózat általános sémája az 5.1. ábra szerinti.



5.1. ábra

Valamely **n** bemenetű és **m** kimenetű kombinációs hálózat tervezését az eddigiek alapján az alábbi lépésekben végezhetjük el.

1) A feladat egyértelmű megfogalmazása és táblázatos megadása.
2) Kapcsolási függvények (m db) megadása.
3) A realizálás módjától függő optimális struktúra minimalizált függvényeinek meghatározása.
4) Rendszertechnikai, áramköri problémák megoldása (terhelés, illesztés stb.).
5) A hálózat megrajzolása és ellenőrzése.
6) Kivitelezési problémák megoldása (tápegység, konstrukció, NYÁK, stb.).

5.1. Kombinációs hálózatok strukturális kérdései

A kapcsolási függvények egyszerűsítésénél ismertetett eljárások (ld. 3. fejezet) áramkapcsoló tulajdonságú rendszerekben (pl. érintkezős hálózat) **soros-párhuzamos** vagy **párhuzamos-soros** hálózati struktúrát eredményeznek. Ugyanezen módszerek a **feszültséglogikájú** rendszerekben **kétfokozatú** (ÉS-VAGY ill. VAGY-ÉS) struktúrát adnak. Mivel a kombinációs hálózatok nagyobb része **többkimenetű**, ez újabb lehetőséget nyújt a hálózat további egyszerűsítésére. Tekintettel arra, hogy napjainkban a huzalozott logika jelentősége csökken, ezért ezen strukturális egyszerűsítési lehetőségeket csak érintőlegesen ismertetjük.

5.1.1. Többkimenetű áramlogikájú hálózati struktúrák

a) **Csillag (Y) struktúra**

b) **Delta struktúra**

c) **H típusú struktúra**

d) **Inverz kimenetű hálózat realizálása morze érintkezőkkel**

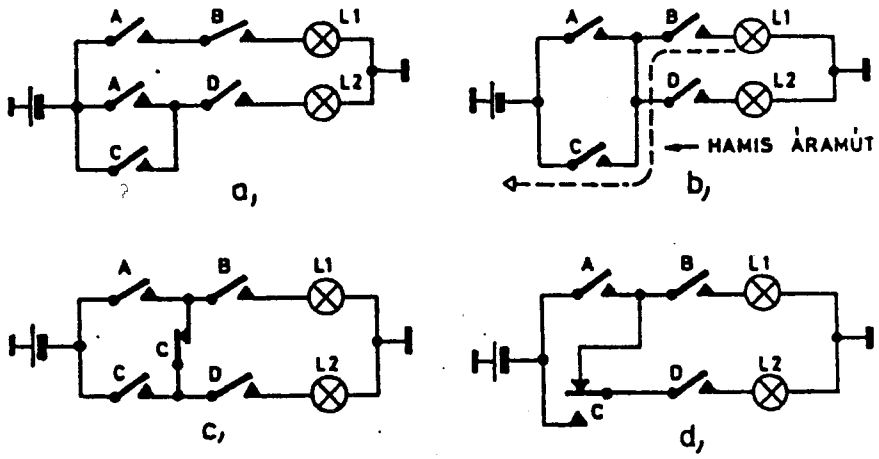
e) **Szimmetrikus hálózati struktúra**

Ilyen hálózatokkal leggyakrabban az ipari jelző-reteszelő rendszerekben találkozhatunk (pl. n darab készülékből k darab működik, vagy páros számú készülék működik, stb).

f) **Faáramkörök**

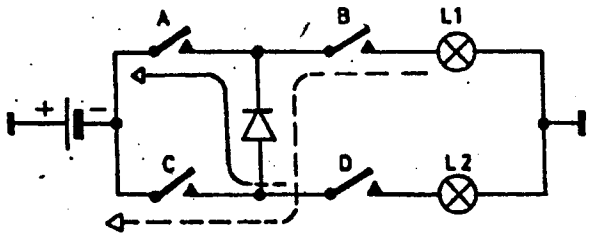
A faáramkörök egy bemenetű és n kimenetű hálózatok. A bemenet bármely kimenettel összeköthető.

g) Hálózati elemek **közös részének kiemelése** révén **többpólus** létrehozása. Ez rendszerint spekulatív módszerrel történik. Ilyenkor az esetlegesen létrejövő **hamis áramutat morze érintkezőkkel** vagy **diódákkal** (egyenáramú hálózat) küszöbölik ki. Példaként az 5.2. ábrán követhetjük végig a módszer lépéseit.



5.2. ábra

A diódás realizáció az 5.3. ábra szerinti.



5.3. ábra

5.1.2. Feszültséglogikájú hálózatok struktúrái

a) Közös implikánsok keresése

A többkimenetű logikai hálózatok közös részeinek kiemelése a kapu típusú hálózatoknál is járulékos egyszerűsítéseket eredményezhet. Ilyenkor a hálózat **közös implikánsainak** megkeresése két lépésből áll:

- a lehetséges implikánsok megkeresése,
- a szükséges implikánsok kiválasztása.

b) MSI áramkörök alkalmazása

Az MSI áramkörök célorientált hálózatok, amelyek előnyösen felhasználhatók a kombinációs hálózatok realizálásánál. Leggyakrabban a multiplexerek és a ROM memóriák használatosak e célra. Tekintettel arra, hogy ezek további ismereteket igényelnek, ezért a későbbiekben erre visszatérünk.

5.2. Kombinációs típusú funkcionális egységek

A **kombinációs típusú funkcionális egységek** napjainkban **MSI áramkörökben** vannak integrálva, vagy programozható berendezésekben **szubrutinok, makrók, táblázatok** formájában vannak tárolva. Ezek ismerete főként a nagybonyolultságú digitális berendezések megértése, alkalmazása céljából fontos. Ilyen funkciók: aritmetikai, kódolás, dekódolás, adatszelektálás.

5.2.1. Aritmetikai áramkörök

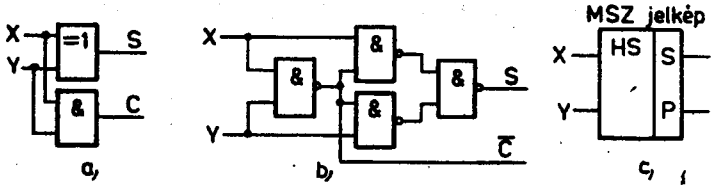
A számítási jellegű műveleteket végző áramköröket **aritmetikai** áramköröknek nevezzük. Az aritmetikai áramkörök alapeleme a **félösszeadó** ill. **teljes összeadó.**

5.2.1.1. Félösszeadó, teljes összeadó

A **félösszeadó** olyan kombinációs hálózat, amelynek **két bemenete a két összeadandó bit (X,Y) és két kimenete az összeg (S) és az átvitel (C).**
A félösszeadó logikai függvényei a 2.4. táblázat alapján:

$$S = X \forall Y \quad \text{illetve} \quad C = X \& Y . \qquad (5\text{-}1)$$

Az 5.4. ábrán a félösszeadót kétféle realizálásban láthatjuk. A b) esetben az átvitel negált értékét $(\overline{C})$ állítottuk elő, ami a teljes összeadó kialakításánál előnyös.

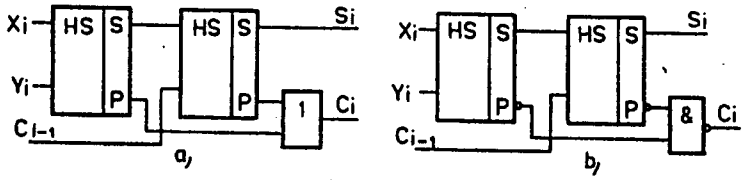


5.4. ábra

A **félösszeadó** - mint ismeretes - az **előző helyiértéken keletkező átvitelt nem tudja figyelembe venni,** ezért legfeljebb a **legkisebb helyiértékek** összegzésére alkalmas. Két n bites szám tetszőleges i-edik helyiértékének összeadására a **teljes összeadót** alkalmazzák, melynek logikai függvényei a 2.5. táblázat alapján:

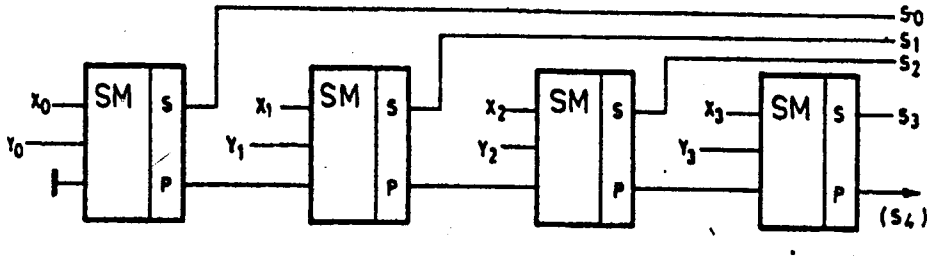
$$S_i = X_i \vee Y_i \vee C_{i-1}$$
$$C_i = X_i\, Y_i \vee X_i\, C_{i-1} \vee Y_i\, C_{i-1} \qquad (5\text{-}2)$$

A függvények alapján a teljes összeadó realizálható, logikai sémája megrajzolható. A **teljes összeadó két félösszeadóból is felépíthető** az 5.5. ábra szerint. Az 5.4. és 5.5. ábrák összevetéséből megállapíthatjuk, hogy a teljes összeadó 9 db kétbemenetű NAND elemmel realizálható.



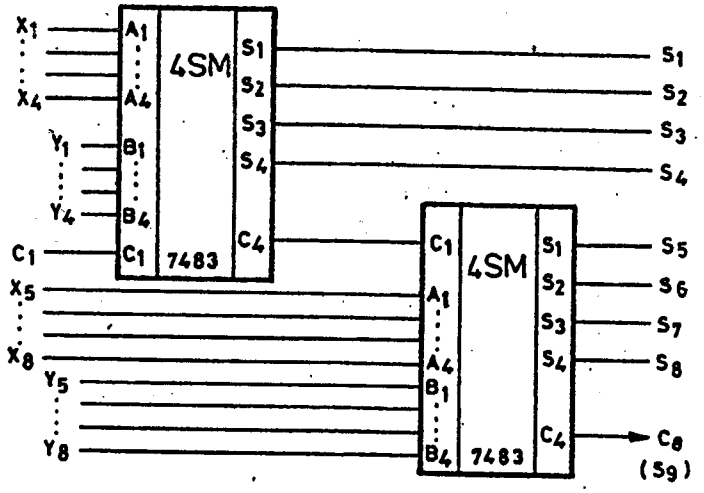
5.5. ábra

A teljes összeadók 5.6. ábra szerinti összekapcsolásával ún. **soros átvitelű, párhuzamos üzemű n** bites összeadó készíthető. Az elnevezés arra utal, hogy a **helyes eredmény közel egyidőben (párhuzamos), de valamennyi összeadó terjedési idejének elteltével** (soros átvitel) jelenik meg a kimeneten.



5.6. ábra

Az MSI áramkörök legtipikusabb aritmetikai áramköre a **7483 típusszámú 4 bites teljes összeadó,** amelynek 9 db bemenete $\left(A_1 \ldots A_4,\ B_1 \ldots B_4,\ C_o\right)$ és 5 db kimenete $\left(S_1 \ldots S_4,\ C_4\right)$ van. A **7483-as összeadók** bővítésével tetszőleges bináris számok összegezhetők **soros átvitelű párhuzamos üzemmódban** az 5.7. ábra szerint.



5.7. ábra

Megjegyezzük, hogy a félösszeadó és a teljes összeadó közötti funkcionális különbség a mikroprocesszortechnikában **bájtokra értelmezve** bukkan fel. Például az ADD művelet nem veszi figyelembe az előző bájtról hozott átvitelt, így a félösszeadóhoz hasonlít. Az ADC utasítás viszont tetszőleges bájtok összeadására alkalmas. Láttuk a **soros átvitelű párhuzamos összeadó** hátrányos tulajdonságát, ami abban nyilvánul meg, hogy a **műveletvégzési idő az összeadandó bitek számával lineárisan nő** (n.tpd). Tekintettel arra, hogy az aritmetikai funkciók a digitális berendezések stratégiailag talán legfontosabb elemei ezért a kutatások a műveletvégzési idő csökkentésére állandó jelleggel folynak.

Az összeadó áramköröket illetően **két strukturális** megoldást dolgoztak ki a sebesség növelése céljából:

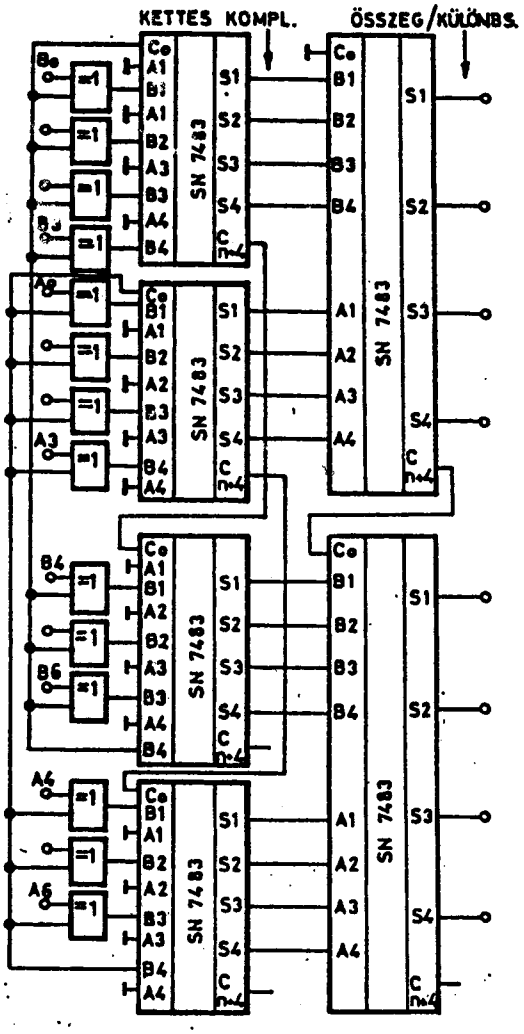
a) **tisztán párhuzamos összeadó** (LOOK AHEAD CARRY)
b) **átvitel mentő összeadó** (CARRY SAVE).

Az **a)** megoldás lényege, hogy az összeg és átvitel képzésénél **nem használja fel az előző carry értéket,** hanem azt **minden bitre külön** előállítja. Ilymódon a hálózat a bitek számával nemlineáris módon bonyolódik, de ez az ára a gyorsabb működésnek. Emiatt vegyes megoldást is alkalmaznak.

A **b)** szerinti **átvitel mentő módszer** több adat összeadásának strukturális módosításával éri el a gyorsítást.

5.2.1.2. Komplemens aritmetikai áramkörök

A **kettes komplemens** hozzáadásával üzemelő összeadó/kivonó kapcsolását az 5.8. ábra mutatja. A 2.1.5.2. fejezetben leírtak alapján a kapcsolás működése követhető. Figyeljük meg, hogy a **hálózat nem igényli a végső átvitelt.** Megemlítjük, hogy a **mikroprocesszorok aritmetikai-logikai egysége** az aritmetikai műveleteket **előjeles kettes komplemensű aritmetikában** végzi.



5.8. ábra

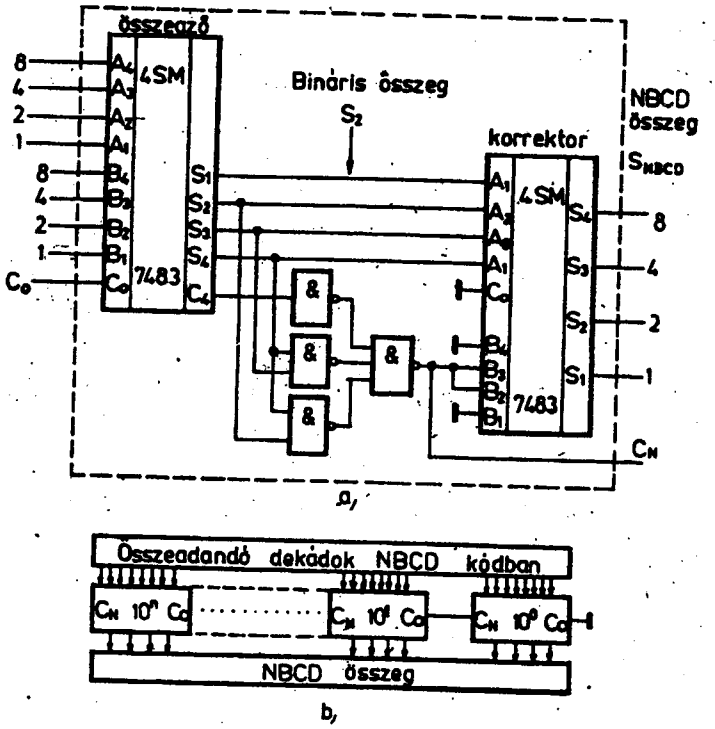
5.2.3. NBCD kódú összeadó

Két **NBCD kódú szám összeadásakor** a bináris 0-tól 19-ig ($9 + 9 + C_0$) minden érték előfordulhat. Ha a decimális összeg ≥ 10, az összeget 10_{10} levonásával korrigálni kell. Végezzük a kivonást 10_{10} **kettes komplemensének** (0110) hozzáadásával.

A **korrekció és a dekádátvitel** feltétele:

$$C_N = C_n \vee S_4S_3 \vee S_4S_2. \qquad (5\text{-}3)$$

Az 5.9. ábra az NBCD kódban üzemelő párhuzamos összeadó kapcsolását és a bővítés sémáját mutatja.



5.9. ábra

Az **NBCD kódú kivonás** a 9-es komplemens hozzáadásával végezhető el az 1-es komplemensű aritmetika analógiájára.

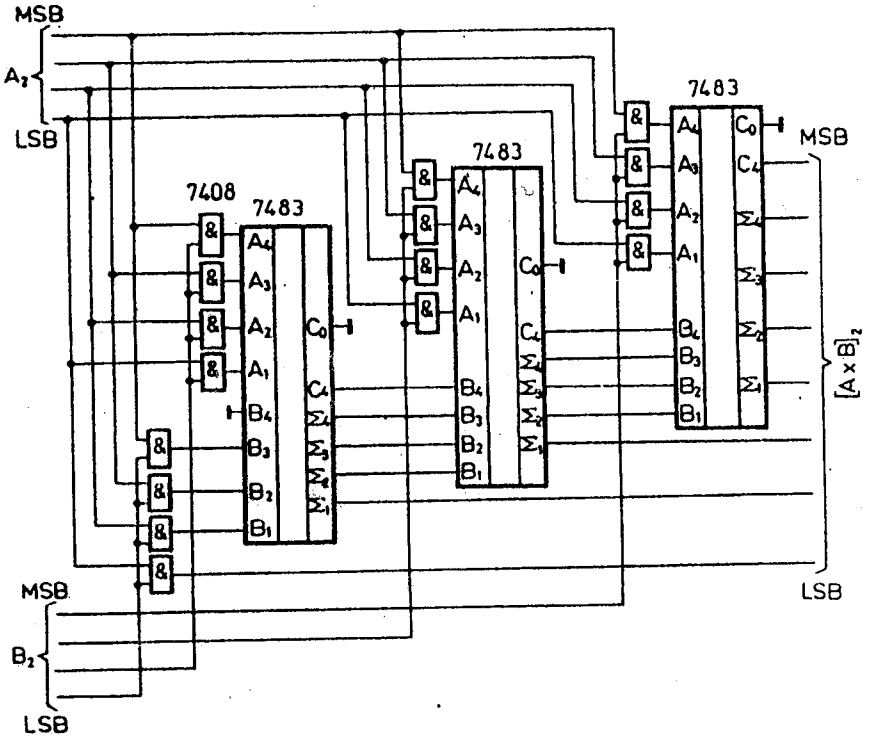
Megjegyezzük, hogy a mikroprocesszortechnikában a BCD kódú összeadás korrekciója a leírt hardver analógiájára a **DAA utasítással** történik.

5.2.1.4. Párhuzamos szorzó

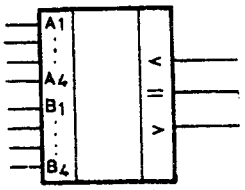
A **bináris szorzás** szabályai alapján **összeadókból** és **ÉS kapukból** könnyen felépítető a **párhuzamos üzemű bináris szorzó**. Az 5.10. ábrán két négybites szám szorzására alkalmas kapcsolást találunk. Egy tokba integrálva a 74284/285 áramköröket forgalmazzák 4 x 4 bites számok szorzásához.

5.2.1.5. Digitális komparátorok

Az aritmetikai áramkörök közé sorolják az **egyenlőségvizsgáló** áramköröket, **digitális komparátorokat** is. Az egyenlőségvizsgáló áramkörök a bináris helyiértékek egyenlősége esetén adnak jelzést a kimeneten, ugyanis **két bináris kódszó akkor egyenlő, ha valamennyi helyiértéken azonos szimbólum** áll. A digitális komparátorok két kódszó relatív nagyságát is képesek eldönteni. Ilyen célra fejlesztették ki a 7485 típusú **bővíthető 4 bites digitális komparátort**. Az áramkör a következő algoritmust realizálja: **két bináris kódszó közül** az a **nagyobb**, amelynél a **legmagasabb helyiérték felől a legalacsonyabb felé haladva az első eltérő helyiértéken 1-es áll**. A komparátor MSz jelképe az 5.11. ábra szerinti. A **7485** típusú komparátorok bővíthetők soros átvitellel és - az összeadók analógiájára - átvitelgyorsítással.



5.10. ábra



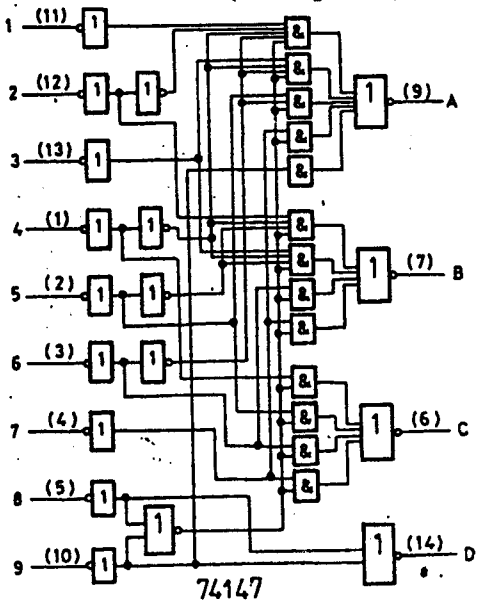
5.11. ábra

5.2.2. Kódolási művelettel kapcsolatos áramkörök

A **párhuzamos üzemű kódátalakítók tulajdonképpen több bemenetű és több kimenetű kombinációs hálózatok.** Tervezésük általános esetben az 5.1-ben leírt módon történik. Mivel az ilyen jellegű funkciók a vezérlő berendezésekben nagy gyakorisággal fordulnak elő, ezért az alkalmazástechnikai szempontból fontos realizációkat tárgyaljuk.

5.2.2.1. Kódoló áramkörök

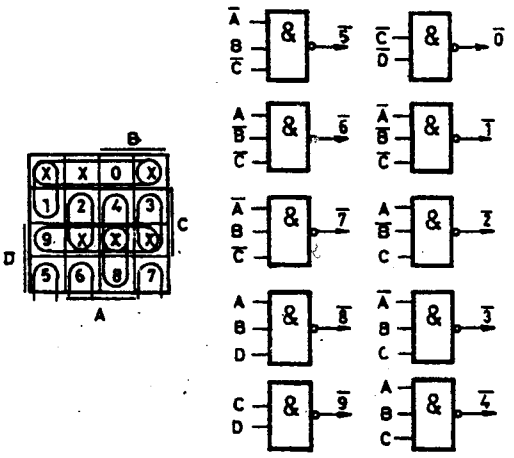
A **kódoló (encoder)** áramkörök az "1 az n"-ből kódot alakítják át nem 1-es súlyozású kóddá. Leggyakoribb funkciók: 1 a 10-ből (NBCD) "1 a 8"-ból (oktális), "1 a 16"-ból (hexadecimális) kódoló. Ilyen áramköröket klaviatúra, nyomógombsor jelének kódolt bevitelénél - ember → gép kapcsolat - használunk. A **kódoló áramkörök mátrix-struktúrában felépített VAGY kapuk**. Az SN74 rendszerben a 74147 típusú MSI áramkörrel is megoldhatjuk a decimális számok konvertálását NBCD kóddá (5.12. ábra). Megjegyezzük, hogy a szoftver technológiában (mikroszámítógépek) a billentyűkódok felismerése **mátrix jelleggel** és **táblázatkezeléssel** (look-up table) történik.



5.12. ábra

5.2.2.2. Dekódoló áramkörök

A **dekódoló áramkörök** - mint a gép → ember kapcsolat fontos láncszemei - a nem 1-es súlyozású kódot alakítják át 1-es súlyozású **"1 az n"-ből** kóddá. A dekódoló áramköröket leggyakrabban a **digitális kijelzők** működtetésére használjuk. A relés dekódolók **faáramkör**, a második generációs dekódolók ÉS **mátrix** felépítésűek. Egy kétszintű hálózattal megoldott Stibitz/decimális dekódoló tervezését mutatjuk be az 5.13. ábrán. **Ügyeljünk arra, hogy a KV tábla cellái bináris súlyozásúak, tehát nem egyeznek meg a 3 fölösleg kódbeli információ kódolt értékei a minterm számmal!** A **digitális kijelzők vezérlésére egy tokba integrált dekódoló áramköröket forgalmaznak.** Megjegyezzük, hogy a szoftvertechnológiában a **dekódolási műveletet** ROM memóriában történt letárolás és **táblázatkezelési módszerrel** oldják meg.

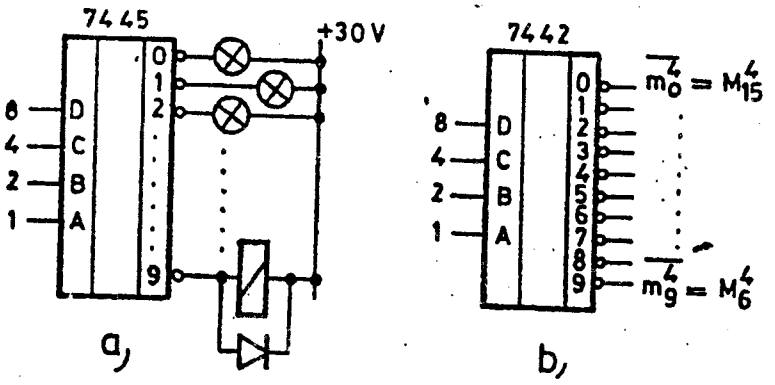


5.13. ábra

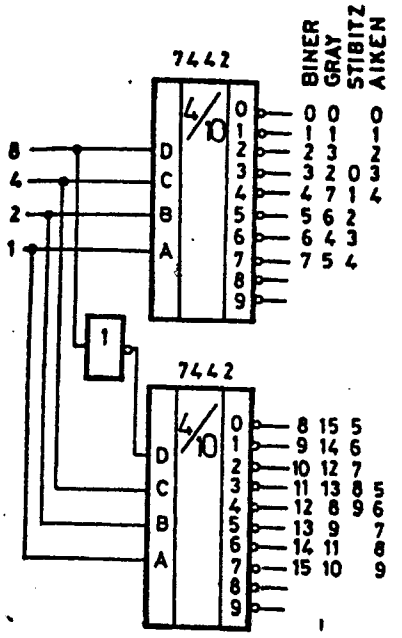
A 74145 típusú áramkör **nyitott kollektoros kimenetei** 80 mA áram kapcsolására alkalmasak, így lámpák, relék, LED-ek működtetésére közvetlenül használható az 5.14.a) ábra szerinti kapcsolásban. Az üzemeltetésnél ügyelni kell arra, hogy a kimeneti magas feszültség U_{CC} hiányában az áramkört tönkre teheti.

A 7442 típusú dekódoló TTL szintű **totem-pole kimenetekkel** van ellátva, így **minterm generátorként** alkalmazható az (5.14.b) ábra) egy- ill. többkimenetű hálózatok realizálásánál. Számlálóval vezérelve **állapotdekódolásra** használják.

Az SN74 rendszerben a másik logikai szintű dekódoló a 74154 típusú áramkör, amellyel a 4 bit bináris → "1 a 16"-ból dekódolás elvégezhető. Ezzel az áramkörrel **bármely 4 bites kód dekódolása** elvégezhető a kimenetek értelemszerű átírásával az 5.15. ábra szerint.



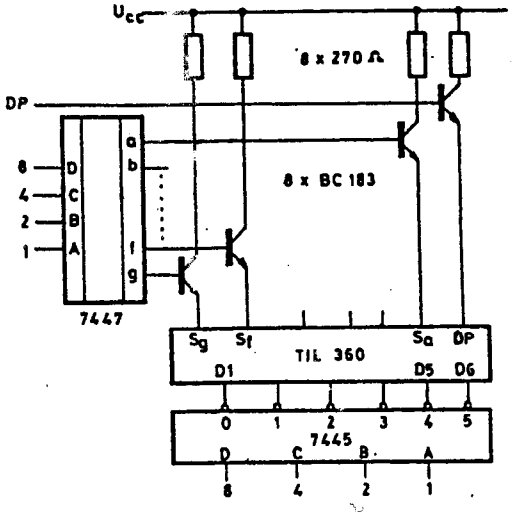
5.14. ábra



5.15. ábra

A **hétszegmensű** kijelzők vezérlése nem 1 az n-ből kód szerint történik, de a hasonló funkció miatt vezérlő áramkörüket dekódolónak nevezik. Az SN74 rendszerben a hat különböző 7 szegmensű dekóder azonos logikai felépítésű, de más-más áramköri specifikációjú. Jellegzetes vezérlési pontjuk az LT, mely révén a LED-ek (szegmensek) **tesztvizsgálatát** végezhetjük el (LAMP TEST). Az RB0 kivezetés megfelelő kapcsolásával a fölösleges 0-ák kiolthatók.

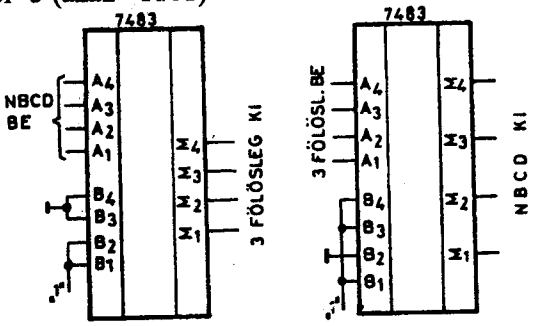
Multidigites kijelző multiplex vezérlését mutatja az 5.16. ábra. A 74145-ös áramkörrel a **digitet választjuk ki**, míg a megfelelő **szegmensek gyújtását** a 7448-as áramkör és a kapcsoló tranzisztorok végzik. A dekódolók további alkalmazási példái: **memória chipek** szelektálása, **kimeneti kapcsok (portok)** szelektálása programozható vezérlőkben, mikroprocesszoros rendszerekben.



5.16. ábra

5.2.2.3. Kódátalakító áramkörök

A párhuzamos üzemű kódátalakító áramkörök a több bemenetű és több kimenetű hálózatoknál leírtak szerint építhetők fel. A tervezésnél ügyelni kell arra, hogy a **KV tábla decimális számozása a bináris kód súlyozásának felel meg, emiatt, ha a bemeneti kód nem 8421 súlyozású, akkor a kódszó által képviselt információ nem egyezik meg a minterm számmal.** Az additív pozícionális kódok közötti átkódolási feladatok **aritmetikai** műveletekre visszavezethetők. Az 5.17. ábrán az NBCD → STIBITZ ill. az STIBITZ → NBCD átalakítást egy-egy 7483-as 4 bites összeadóval oldottuk meg. Előző esetben +3, utóbbi -3 (azaz +1101) hozzáadásával.

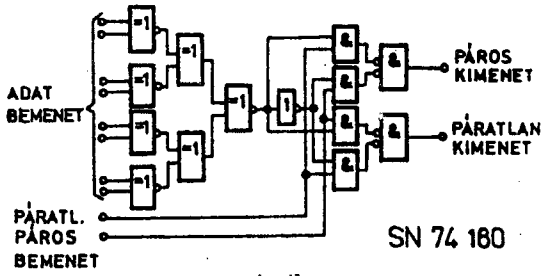


5.17. ábra

A nagyobb bonyolultságú kódok konvertálásánál előnyösen használhatók a **ROM memóriák.** A kódátalakítók jellegzetes csoportját alkotják az NBCD → bináris (74184) ill. bináris → NBCD (74185) kódátalakítók.

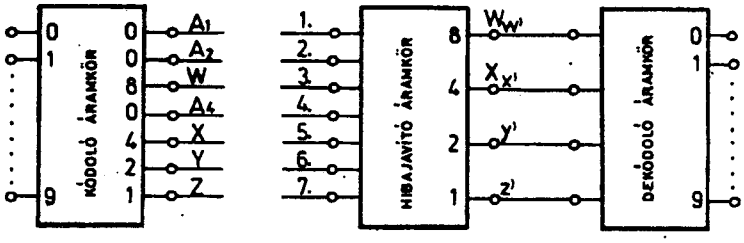
5.2.2.4. Hibafelfedő és javító áramkörök

A 2. fejezetben leírt **párosságellenőrzést** igen gyakran alkalmazzák, mert így tetszőleges kódtípus felruházható **hibafelfedő** képességgel. A párosság ellenőrző áramkör egy **szimmetrikus áramkör**. Például az NBCD kódot páratlan egyesekre kiegészített kód hibajelző áramköre egy $S^5_{0,2,4}$ szimmetrikus hálózat. A paritás bit előállítása és ellenőrzése ugyanazzal az áramkörrel elvégezhető. TTL rendszerben a 74180 típusú áramkör 8 bit páratlan ill. párosság ellenőrzésre és generálásra használható az 5.18. ábra szerint.



5.18. ábra

A hibajavító áramkörök közül a **Hamming-féle hibajavító** kóddal ellátott adatátviteli rendszer felépítését és működését ismertetjük az 5.19. ábra felhasználásával.



5.19. ábra

Az áramkör 1 hiba javítására alkalmas. A Hamming-kód táblázata:

Információs vezetékek: W, X, Y, Z.

Ellenőrző vezetékek: A_1, A_2, A_4.

	1	2	3	4	5	6	7
	0	0	8	0	4	2	1
	A_1	A_2	W	A_4	X	Y	Z
0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1
2	0	1	0	1	0	1	0
3	1	0	0	0	0	1	1
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	1
6	1	1	0	0	1	1	0
7	0	0	0	1	1	1	1
8	1	1	1	0	0	0	0
9	0	0	1	1	0	0	1

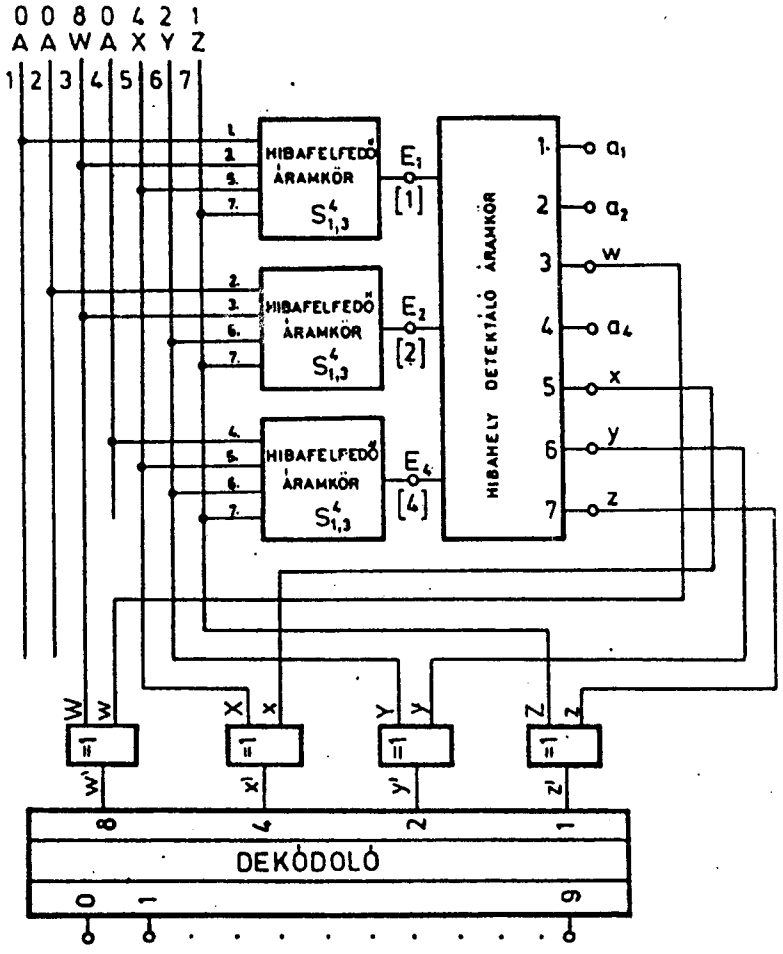
Az ellenőrző bitek 3 négyes csoportban vannak az információs bitekhez rendelve az alábbiak szerint. Az A_1 paritás bit a 2^0 t tartalmazó, az A_2 a 2^1 -t, az A_4 pedig a 2^2 -t tartalmazó vezetékek párosságellenőrzését végzi.

	1	3	5	7
	A_1	W	X	Z
	0	8	4	1
0	0	0	0	0
1	1	0	0	1
2	0	0	0	0
3	1	0	0	1
4	1	0	1	0
5	0	0	1	1
6	1	0	1	0
7	0	0	1	1
8	1	1	0	0
9	0	1	0	1

	2	3	6	7
	A_2	W	Y	Z
	0	8	2	1
0	0	0	0	0
1	1	0	0	1
2	1	0	1	0
3	0	0	1	1
4	0	0	0	0
5	1	0	0	1
6	0	0	1	0
7	0	0	1	1
8	1	1	0	0
9	0	1	0	1

	4	5	6	7
	A_4	X	Y	Z
	0	4	2	1
0	0	0	0	0
1	1	0	0	1
2	1	0	1	0
3	0	0	1	1
4	1	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	0	0	0	0
9	1	0	0	1

A hibajavító áramkör sematikus rajza az 5.20. ábra szerinti felépítésű. A hiba felfedését a 3 szimmetrikus áramkör végzi. Az E_1, E_2, E_4 pontokon megjelenő hiba kombináció a hibás vezeték számának bináris megfelelőjét mutatja, így dekódolással a hibahely meghatározható. Természetesen csak az információs vezetéken fellépett hibát kell javítani. A korrigálást ANTIVALENCIA elemmel, mint **vezérelt inverterrel** végezhetjük el (5.20. ábra).



5.20. ábra

Példaként tételezzük fel, hogy a vevő áramkör bemenetére az

1	2	3	4	5	6	7
A_1	A_2	W	A_4	X	Y	Z
1	1	0	0	0	1	0

kombináció érkezik.

Az E_1 kimeneten megjelenő érték 1, mert

A_1	W	X	Z	
1	0	0	0	→ páratlan, emiatt $E_1 = 1$.

Az E_2 kimeneten megjelenő érték

2	3	6	7	
A_2	W	Y	Z	
1	0	1	0	→ páros → $E_2 = 0$.

Az E_4 vezetéken megjelenő érték

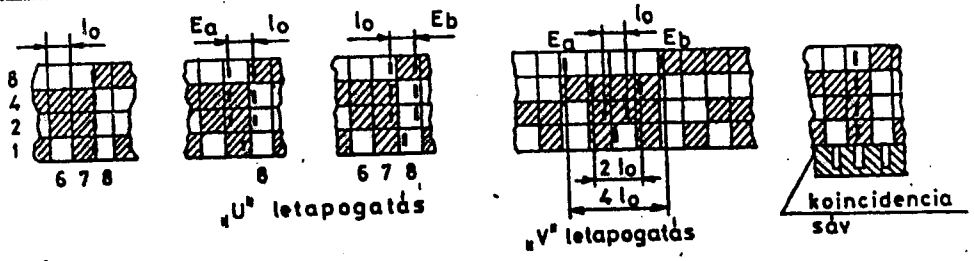
4	5	6	7	
A_4	X	Y	Z	
0	0	1	0	→ páratlan → $E_4 = 1$.

Tehát a hiba helye a 101_2 bináris kódú, azaz az 5-ös (X) vezeték. A korrekció:

$X \forall \overline{X} = 0 \forall 1 = 1.$

5.2.3. Helyzet/digitál átalakítók

A kódolás egyik ipari alkalmazási területe a **digitális elmozdulás** és **elfordulásmérés,** melyet **kódléccel** illetve **kódtárcsával** oldanak meg. A kódolt eszköz felületén **bináris állapotokat** képeznek ki (pl. vezető - szigetelő, fényt áteresztő - nem áteresztő stb.). A használt mintázattól függő érzékelőt alkalmaznak. Ilyenek: leszedő kefe, foto elektromos átalakító. Az érzékelőket a sávok függőleges határvonalaival párhuzamosan helyezik el. A **kódlécet,** illetve **tárcsát** a **mozgó alkatrészekhez** rögzítik. A kódolt eszközök ábrázolásánál megállapodás szerint 1-et kapunk az érzékelő kimenetén, ha feketített és 0-át, ha üresen hagyott részen áll. Az érzékelők pontos beállítása a sok sávban nehézkes. **NBCD** és **bináris kódolású lécek** esetén az érzékelők helyzetének kismérvű megváltozása **átmenetileg hamis, leolvasást** eredményezhet. Ennek megvilágítására tekintsük meg az 5.21.a) ábrát, amelyen egy NBCD kódolású léc van. Könnyen látható, ha az érzékelők nem merőlegesen egy egyenesben vannak, akkor az átmenet pillanatában hibás leolvasás történik. Az említett letapogatási hibák kiküszöbölésére elvileg két alapvetően különböző módszert használnak: **a) több érzékelő alkalmazása, b)egyátmenetű kódok alkalmazása.**



a, *b,* *c,*

5.21. ábra

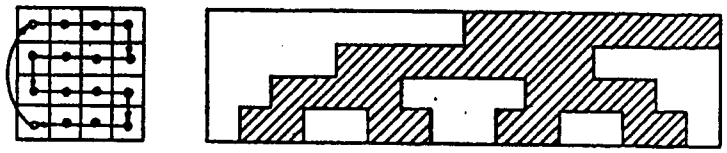
Mindkét módszernek megvan az előnye és a hátránya.

a) **"U" ill. "V" letapogatás**

Az 5.21.b) ábra az ún. **"U" letapogatást** szemlélteti. "U" letapogatás az érzékelők párhuzamos kettőzésével érhető el. Jellegzetessége, hogy az 1-es sávon egy érzékelő ($É_1$), míg az összes többi sávon 2-2 érzékelőt helyeznek el egymástól egy osztásnyi (1_0) távolságra. Ha $É_1 = 1_0$, akkor az $É_a$ érzékelőkről, ha $É_1 = 0$, akkor az $É_b$-ről vesszük a jelet, ugyanis a választott érzékelők az átmenettől (még) legalább 1/2 1_0 távolságra vannak. A módszer hátránya, hogy kis 1_0 osztású léceknél a két érzékelő egymás melletti elhelyezése nehézkes. A **"V" letapogatás** az érzékelők V alakú kettőzésével oldható meg. Azzal a megfontolással alakítható ki, hogy bináris kódlécnél a magasabb helyi értékű sávon az egyértelmű leolvasás akkor is biztosított, ha az érzékelőket ott egymástól távolabb ($2^i \cdot 1_0$, ahol i a helyiérték) helyezzük el (5.21.c) ábra). Ezáltal természetesen nagyobb elhelyezési pontatlanság engedhető meg. A "V" letapogatásnál a helyzet meghatározása azaz az érzékelők kiválasztása bonyolultabb, mint az "U" letapogatásnál. Mindkét módszer előnye, hogy a leolvasott kód továbbra is bináris, vagy NBCD és ehhez szabványos hardverek ill. szoftverek állnak rendelkezésre. Hátrányuk az érzékelők duplázásából és a korrekciós áramkörökből származó **járulékos költség**.

b) **Egyátmenetű kódok**

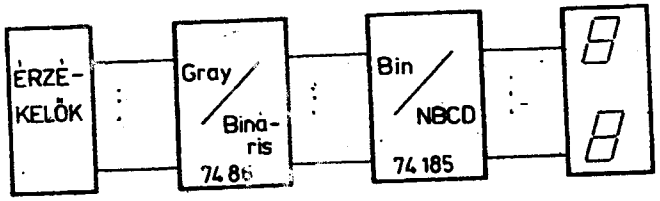
Az említett leolvasási hibák **egyátmenetű (egylépésű, progresszív, ciklikus)** kódok alkalmazásával is kiküszöbölhetők. Egylépéses kódot úgy készíthetünk, hogy a KV tábla szomszédos mintermjeit valósítjuk meg. A leggyakrabban használt egyátmenetű kód a GRAY-kód. A 4 bites **GRAY kód** származtatása és a kódléc mintázata látható az 5.22. ábrán. Ezt a kódot **tükrözött bináris** kódnak is szokás nevezni. Hátránya, hogy 2^n-re egyátmenetű, így a többdekádos dekódoló áramköre bonyolult.



5.22. ábra

Egy Gray-kódolású helyzet-digitál átalakító rendszertechnikai sémája látható az 5.23. ábrán az érzékeléstől a kijelzésig. Ismeretes, hogy a decimális adatfeldolgozáshoz a 10^n-re egyátmenetű, azaz dekádonként egyátmenetű

kódok előnyösebbek. Ilyen kódokat úgy készíthetünk, ha az egyátmenetet a 9 → 0 átmenetnél is biztosítjuk. Ez azt jelenti, hogy a KV táblából úgy választunk ki 10 egymással szomszédos mintermet, hogy az egyátmenetet a 9 → 0 esetére is biztosítjuk.



5.23. ábra

Számos ilyen kódot dolgoztak ki (pl. Watts kód, O'Brien kód, Topkins kód, stb.). Megjegyezzük, hogy az iparban alkalmazott kódolt eszközöknél a 12 bitnél nagyobb felbontásút ($\approx 5'16''$) gazdasági okok miatt nem alkalmaznak.

5.2.4. Multiplexerek, demultiplexerek

A **multiplexerek** (adatszelektorok, vonalkiválasztók) **több bemenetű és egy kimenetű kombinációs hálózatok.** A bemenetek funkciójuk szerint két nagy csoportba sorolhatók:

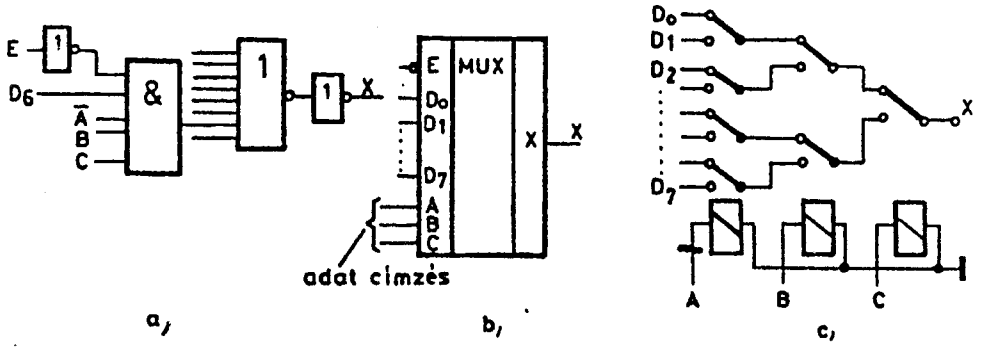
- adat bemenetek $(D_0 ... D_i)$
- választó (címző) bemenetek (A, B, C...).

A multiplexer címző bemeneteire adott bináris jelkombinációnak megfelelő decimális indexű D_i bemenet jele (0 vagy 1) kerül a kimenetre. A multiplexerek tulajdonképpen **ÉS/VAGY hálózatok**. Egy 8-ról 1 vonalra szelektáló áramkör 8 darab 4 bemenetű (3 bit címzés + D_i) ÉS kapuból és egy 8 bemenetű VAGY kapuból áll. Ennek egy közbenső kapujának bekötését (D_6) szemlélteti az 5.24.a) ábra, a b) ábrán pedig a szimbólumát találjuk.

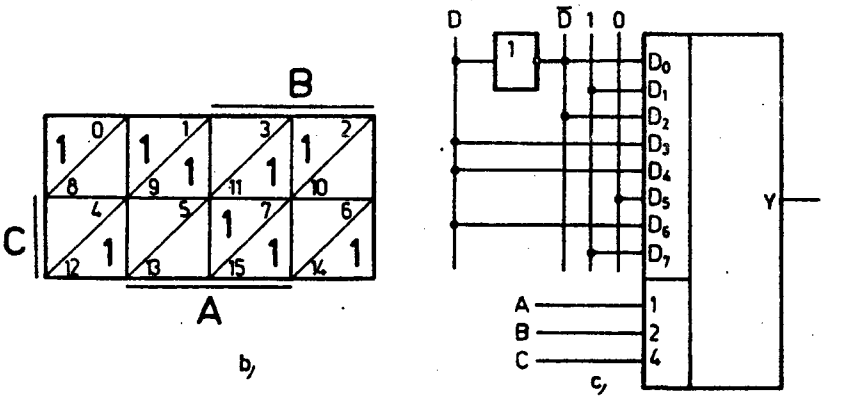
A 74150 típusszámú IC 8/1-es, a 74151 típusú pedig 16/1 vonalszelektálást lát el. Az **adatszelektorok az E bit felhasználásával bővíthetők**. Láttuk, hogy az adatszelektorok célszerűen kialakított ÉS/VAGY hálózatok. Amennyiben a **MUX címző bemeneteire a logikai változókat kapcsoljuk, úgy univerzális logikai modulokhoz (ULM)** jutunk, amelyekkel a logikai függvények könnyen realizálhatók. Erre láthatunk példát az 5.25. ábrán, ahol az

$$F(D,C,B,A) = \Sigma(0,1,2,7,9,11,14,15).$$

függvényt egy 8/1-es adatszelektorral és egy inverter felhasználásával realizáltuk.



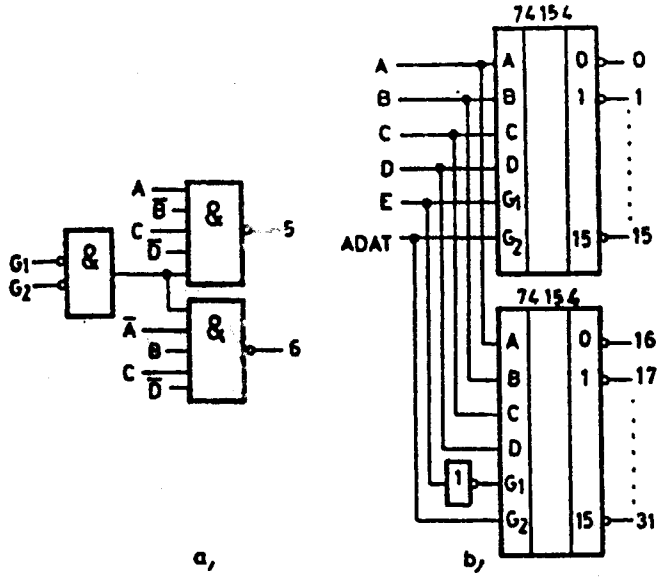
5.24. ábra



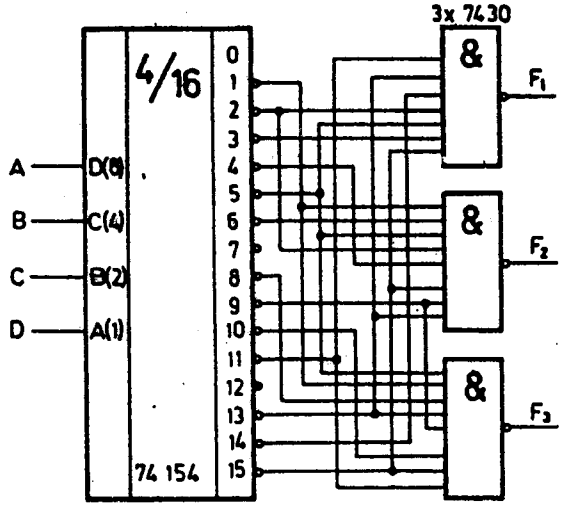
5.25. ábra

A **demultiplexerek a bemeneti-adatvonal (D) jelét (0 vagy 1) a címző vezetékekre adott kombináció által meghatározott kimenetre kapcsolják**, azaz a multiplexerekkel ellentétes műveletet végeznek. A demultiplexerek felépítése igen hasonlít a **dekódolók** felépítéséhez, a különbség az engedélyező (adat) bemenetekben van. A demultiplexerek tehát engedélyező bemenettel ellátott minterm generátorok. Az 5.26. ábrán egy 4/16 vonalú DEMUX két kimenetének belső kapcsolását, valamint 5/32 vonalra bővített változatát rajzoltuk meg.

A MUX-ok és a DEMUX-ok együttes alkalmazásával azonos címzés esetén **párhuzamos/soros,** ill. **soros/párhuzamos** üzemmód átalakítás valósítható meg. Láttuk, hogy egy DEMUX áramkör a beépített NAND kapuk révén **mintermgenerátorként** is felfogható. Ennélfogva előnyösen felhasználható **logikai függvények realizálására a diszjunktív alak közvetlen megvalósítása** révén. Szemléltetésül egy három kimenetű (F_1, F_2, F_3) hálózatot az 5.27. ábrán mutatunk be. Írja fel a hálózat által realizált függvényeket teljes diszjunktív normál alakban!



5.26. ábra



5.27. ábra

5.3. Hazárdok

Az eddigiekben a kombinációs hálózat tervezésénél és vizsgálatánál feltételeztük, hogy az összetartozó bemeneti jelek **egyidőben** jelennek meg az áramkör bemenetén. A valóságban a jelek **különböző útvonalakon** haladva **különböző késleltetéseket** szenvedve jutnak el a kimenetig. Emiatt a valóságos kombinációs áramkör kimenő jele általában csak **állandósult állapotban**

határozható meg egyértelműen a bemenő változók függvényeként. Valahányszor a **bemeneten jelváltozás lép fel**, fennáll a veszély, hogy a kimeneten **átmenetileg hamis jel** (tranziens) lép fel.

A tranzienseket keletkezésük oka szerint két csoportba sorolhatjuk:

1) Tranziens **ideális hálózat** kimenetén is felléphet, ha két állandósult állapothoz tartozó bemeneti kombináció között a **Hamming-féle távolság egynél nagyobb**. Ezen jelenség egyik speciális esetét és kiküszöbölését láttuk a bináris kódléc esetében.

2) A hamis kimenő jelek másik csoportja magából a **hálózatból ered** és akkor is felléphet, ha a bemeneti jelkombinációk csak **egy változó értékében** különböznek. A továbbiakban a hamis jelek ezen fajtáival foglakozunk.

Ismeretes, hogy egy adott bemeneti jelkombinációról egy attól **csak egy változó** ponált vagy negált értékében különböző másik jelkombinációra való áttérés úgy tekinthető, mint a KV tábla adott celláról egy szomszédos cellára való átmenet. Háromféle átmenetet különböztetünk meg:

a) **egy 1-gyel jelölt celláról egy szomszédos 1-gyel megjelölt cellára** való átmenet,
b) **egy 0-val megjelölt celláról egy szomszédos 0-val megjelölt cellára** való átmenet,
c) **0-val jelölt celláról 1-gyel jelölt (és fordítva) történő átmenet.**

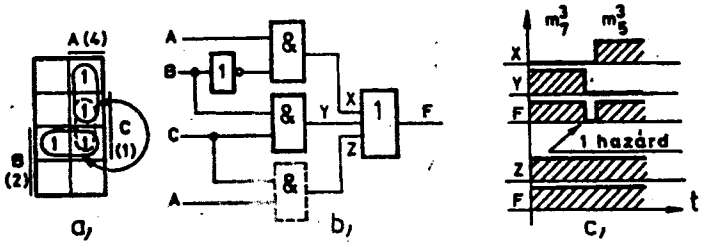
Belátható, hogy az **a)** típusú átmenetnél az áramkör kimenetén megjelenő jelnek változatlanul 1-nek kell maradnia. Amennyiben az átmenet pillanatában ennek ellenére 0 jel jelenik meg, akkor azt mondjuk, hogy az átmenet **statikus 1 hazárd-ot tartalmaz. Statikus 1 hazárd esetén tehát az állandónak feltételezett 1 értékű kimenet rövid időre 0 értékűre változik.**

A **b)** típusú átmenet **statikus 0 hazárdot** eredményez, amennyiben átmenetkor a 0 értékű állandósult kimenő jel mellett 1 értékű hamis tranziens lép fel.

A **c)** típusú átmenet **dinamikus hazárdnak** nevezzük, ha átmenetkor a kimenő jel az előírt **egyszeri változás helyett többször is megváltozik**, mielőtt az állandósult értékét felvenné.

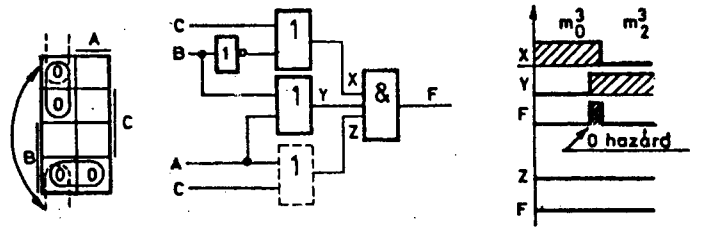
A **statikus hazárdok** vizsgálatára tekintsük az 5.28.a) ábrát, melyen egy 3 változós függvényt KV táblán adtuk meg. Az egyszerűsítést az ismert eljárással végezve kapjuk:

$$F = A\bar{B} \vee BC .$$



5.28. ábra

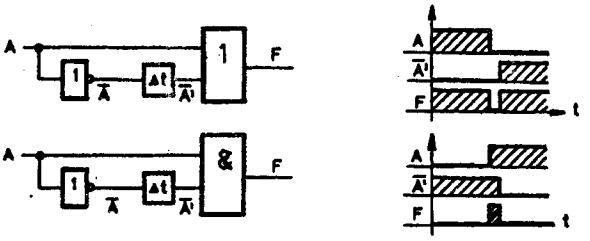
Ez a kapcsolás látható (folytonos vonal) a b, ábrán, a c, ábrán pedig azt az esetet rajzoltuk meg, amikor a hálózat bemenetén az m_7^3-et az m_5^3 követi. Az idődiagramon jól látható, hogy átmenetkor hamis 0 lép fel. Figyeljük meg, hogy azt az esetet vizsgáltuk meg, amikor két **különböző tömb** között történt az átmenet a KV táblán. **Statikus 1 hazárd mindig előfordul,** ha az **átmenet nem egyetlen hurok** tartományán belül megy végbe. A hálózat tehát akkor tartalmaz statikus 1 hazárdot, ha a hálózatot leíró függvény minterm diagramján található **két 1-gyel jelölt szomszédos cella, amely nincs közös tömbbel lefedve.** Ha tehát el akarjuk kerülni a statikus hazárdot, akkor mindig a KV tábla egy területrészét **lefedő tömb belsejében kell** az átmenetnek történnie. Tehát a statikus hazárdot úgy tudjuk megszüntetni, hogy egy célszerűen megválasztott **redundáns tömbbel az átmenetet lefedjük** (szaggatott vonal). A redundáns tömb megvalósítása újabb ÉS kaput igényel. Az 5.29. ábrán ugyanezen függvény konjunktív alakját realizáltuk és az $m_0^3 \rightarrow m_2^3$ átmenetet vizsgáltuk idődiagramon. A hazárdmentesítést most is **redundáns tömbbel** végeztük



5.29. ábra

Ismeretes, hogy a korábban bemutatott minimalizálási eljárások alapja az $A \vee \overline{A} = 1$ illetve $A \cdot \overline{A} = 0$ összefüggések. Az eddigiek során feltételeztük, hogy a kapcsolóelemek realizálják ezen összefüggéseket. Ezek azonban csak **ideális kapcsolók** esetén teljesülnének. Tudjuk, hogy a **valóságos kapcsolók** késleltetéssel (Δt) rendelkeznek. A statikus 1 és 0 hazárdok ezen logikai tételeket megvalósító hálózatokon is tanulmányozhatók, s a **hazárdot létrehozó alapkapcsolásoknak** is tekinthetők (5.30. ábra). Megállapíthatjuk, hogy ezen

összefüggések valóságos kapcsoló elemekkel történő realizáláskor **csak állandósult állapotban teljesülnek.** A **dinamikus hazárdok** a **többfokozatú hálózatokban** fordulnak elő. **Kétszintű ÉS - VAGY realizáció nem tartalmaz dinamikus hazárdot.** Ha egy logikai függvény valamennyi primimplikánsa realizálva van ÉS kapuval (diszjunktív alak), illetve VAGY kapuval (konjunktív alak), akkor a **kétszintű hálózat nem tartalmaz statikus hazárdot.**



5.30. ábra

Ha a **hazárdmentes ÉS - VAGY realizáció kapuit NAND kapukkal** helyettesítjük, akkor **hazárdmentes megoldást** kapunk. Ez azt jelenti, hogy a kétszintű hálózatok minimalizálása esetében az összes primimplikánssal lefedve a függvényt, szükségképpen hazárdmentes megoldást kapunk. Előfordulhat azonban, hogy a hazárdmentes hálózat megvalósításához **nincs szükség az összes primimplikánsra.** A logikai függvény KV táblázatos hazárdvizsgálatának elvén algebrai feltételek is adhatók a hazárdvizsgálatra. A statikus hazárd algebrai feltételei:

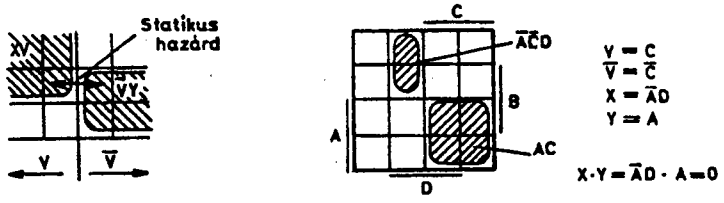
a) **Szükséges feltétel,** hogy **a függvény algebrai alakjában forduljon elő a következő formába rendezhető primimplikáns:**

$$VX \text{ é } s\overline{V}Y.$$

b) **Elégséges feltétel,** hogy a fenti primimplikánsok **X és Y részének legyen közös része:**

$$XY \neq 0.$$

Ennek megvilágítására tekintsük az 5.31. ábrát, amelyen egy tetszőleges változó számú KV tábla egy részletét ábrázoltuk, továbbá egy függvény két egymást nem fedő primimplikánsához tartozó hurkok egy-egy részletét. Ismeretes, hogy a függvény minden primimplikánsának a függvény KV táblázatán egy-egy hurok felel meg, és fordítva. A KV táblák sajátossága, hogy az ábrázolt függvény bármelyik V független változójához tartozik a táblázatnak egy olyan szimmetria vonala, amely a táblázatot két egyenlő területre osztja, s e területek közül az egyiket V, a másikat $\overline{V}$ határozza meg. Az ábrán egy ilyen szimmetriavonalat vastag vonallal jelöltünk. Természetesen ez fordítva is igaz, hiszen a táblázat valamennyi vonala a függvény valamelyik független változójához tartozó szimmetriavonal.



5.31. ábra

Ha a függvénynek van két olyan egymást nem fedő primimplikánsa, amelyek az 5.31.a) ábrán látható módon szomszédosak, akkor azokat feltétlenül egy ilyen szimmetriavonal választja el egymástól. Ez azt jelenti, hogy az egyik algebrai kifejezésben a szimmetriavonalra jellemző V változó, a másikában pedig ugyanaz negált ($\overline{V}$) értékkel szerepel. A diszjunktív kanonikus alakból számított primimplikánsokat feltételezve a szomszédos primimplikánsok algebrai alakjai tehát

$$VX \text{ illetve } \overline{V}Y$$

ahol X illetve Y a függvény változóiból alkotott olyan logikai szorzatok, melyeknek tényezői között sem V, sem $\overline{V}$ nem fordul elő. Az 5.31.b) ábra alapján belátható, hogy ez a feltétel nem elégséges. Végezzük el az algebrai hazárdvizsgálatot az 5.28. ábra esetére

$$F = \underbrace{A\&\overline{B}}_{a} \vee \underbrace{B\&C}_{b}.$$

Az a és b primimplikánsok között a **szükséges feltétel teljesül**, hiszen a B változó mindkét primimplikánsban szerepel. Az a-ban negált, a b-ben pedig ponált értékével.

$$A\&C \neq 0.$$

Vagyis az **elégséges feltétel** is teljesül. A hazárdmentesítő tömböt az algebrai alakból úgy kapjuk, hogy a hazárdos primimplikáns párból kiemeljük a differenciálváltozót (tehát a B illetve $\overline{B}$-at) és a megmaradt kifejezéseket egyesítjük. Így példánkban a **hazárdmentesítő hurok** AC. A statikus hazárd **működési zavarokat** okozhat, amennyiben a kombinációs hálózat gyors működésű sorrendi áramkört vezérel. Az **aszinkron sorrendi áramkörökben** fellépő statikus hazárd okozta problémára visszatérünk. Bármelyik hazárd típus kiküszöbölhető a logikai hálózat megfelelő pontjain elhelyezett **energiatárolós** késleltető elemekkel, de az mindenképpen csökkenti a rendszer sebességét. A statikus hazárdot azonban nem minden esetben kell kiküszöbölni. Nyilvánvalóan megengedhető olyan esetekben, amikor a kombinációs hálózat olyan rendszerre dolgozik, amelyek működését nem zavarja a hazárd. Ilyenek pl. a lassú működésű kijelző rendszerek. Nem okoz zavart a meghajtó hálózat hazárdja akkor sem, ha olyan vezérlési kombináció során fordul elő, amely a meghajtott rendszer szempontjából közömbös. Az ilyen hazárdot **közömbös statikus**

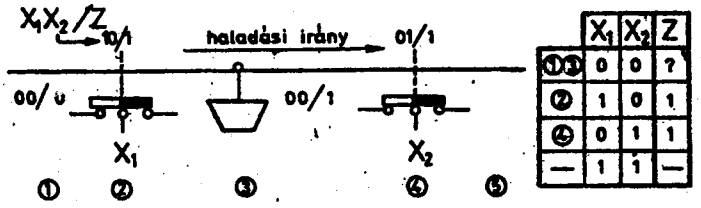
hazárdnak nevezzük. A hazárdmentes hálózat kialakítása a Quine és a numerikus minimalizálási módszer alábbi korrigálását igényli. A primimplikánsok kiválasztása természetesen nem igényel változtatást. A **szükséges primimplikánsok kiválasztása** viszont a primimplikáns táblázat módosítását teszi szükségessé. A módosított primimplikáns táblázatban a függvény minden szomszédos mintermje és nem párosítható különálló mintermje egy-egy, oszloppal szerepel. A párosításnál tulajdonképpen a második oszlopban egymás mellé írt mintermeket kell figyelembe venni, ugyanis ezek csak egy változóban térnek el. A szükséges primimplikánsokat az ismert módon választjuk ki: a primimplikánsoknak valamennyi oszlopot le kell fedniük. Természetesen a hazárdvizsgálatot és hazárdmentesítést úgy is elvégezhetjük, hogy a numerikus módszerrel egyszerűsített függvény **algebrai alakjára** alkalmazzuk a statikus hazárd **szükséges és elégséges feltételét.** Érdemes megjegyezni, hogy a **dinamikus hazárd** a mechanikus és **elektromechanikus kapcsolók működésekor fellépő tipikus jelenség** (pl. prell), de a **többszintű hálózatokban** is igen gyakran előfordul.

Felhasznált irodalom

Ajtonyi I: Vezérléstechnika I. J 14-1417 Tankönyvkiadó, Budapest, 1987.
Ajtonyi I: Vezérléstechnika II. J 14-1418 Tankönyvkiadó, Budapest, 1987.
Ajtonyi I: Vezérléstechnika példatár J 14-1419 Tankönyvkiadó, Budapest, 1987.
Arató-Kondorosi-Risztics: Logikai hálózatok tervezése II. TK. 1973.
Arató-Risztics: Logikai hálózatok tervezése I. TK. 1973.
Janovics - Tóth: A logikai tervezés módszerei. Műszaki Könyvkiadó, 1971.
Morris-Miller: Designing with TTL Integrated Circuits Mc GRAW-HILL, 1971.
Wickes: Logic Design with Integrated Circuits John Wiley, 1968.
The Integrated Circuits Catalog for Design Engineers Texas Instruments.

6. SORRENDI HÁLÓZATOK

A digitális rendszerek funkcionálisan kombinációs ill. sorrendi (szekvenciális) típusúak lehetnek. Az eddigiekben megismert kombinációs hálózatokkal a feladatoknak csak egy részét lehet megoldani. Ennek szemléltetésére legyen feladatunk a 6.1. ábra szerinti jelző áramkör tervezése. Az ábra szerinti pályaszakaszon a csillék a megjelölt irányban haladhatnak. Az X_1, X_2 morze típusú végálláskapcsolók akkor adnak rövidzárat, amikor a csille előttük tartózkodik.



	X_1	X_2	Z
①⑤	0	0	?
②	1	0	1
④	0	1	1
—	1	1	—

6.1. ábra

Amennyiben a feladat megoldását a kombinációs hálózatoknál ismertetett módszerrel kezdjük, akkor már a táblázat első sorában problémát okoz, hogy az X_1 X_2= 00 kombinációhoz milyen kimeneti érték tartozik. A bemeneti és kimeneti értékeket az ábrán is bejelöltük (X_1, X_2, Z). Az ábrából kitűnik, hogy **ua. bemeneti kombinációhoz más-más kimeneti esemény tartozik attól függően, hogy ezen bemeneti kombinációk milyen sorrendben érkeznek az áramkör bemenetére.** Az ilyen hálózatokat **sorrendi (időrendi, szekvenciális)** hálózatoknak nevezzük. Ez azt jelenti, hogy kombinációs táblázattal a sorrendi áramkörök nem írhatók le, ezért új leírási módra van szükség.

6.1. Sorrendi hálózatok leírási módszerei

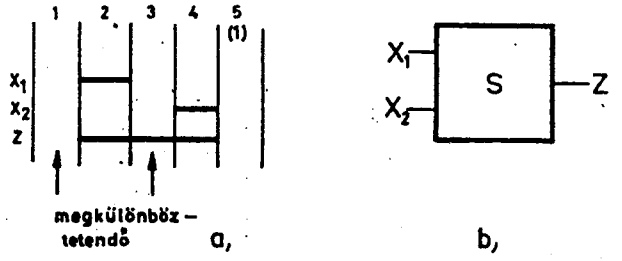
A sorrendi áramkörök leírására az alábbi módszerek terjedtek el:

a) ütemdiagram
b) állapot tábla
c) állapot diagram.

Meg kell azonban jegyezni, hogy a **programozható eszközök** elterjedésével előtérbe kerültek a számítástechnikából átvett, ill. adaptált módszerek, mint pl. a **folyamatábra, létradiagram,** stb.

a) Ütemdiagram

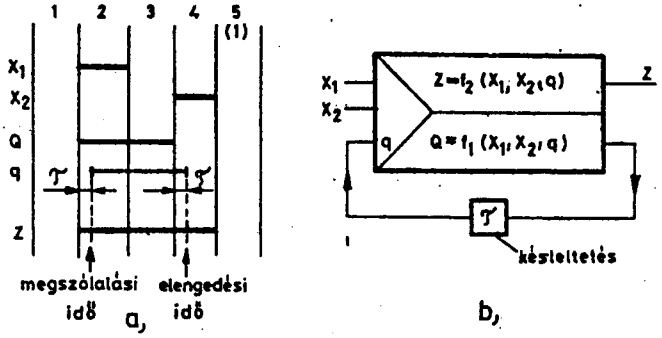
A 6.1. ábrán bevezetett példa **ütemdiagramja** a 6.2.a) ábra szerinti, míg a b) ábra a hálózat be és kimeneteit szemlélteti.



6.2. ábra

Az ábrán a **megkülönböztetendő ütemeket** is berajzoltuk. Ezek az 1 ill. 3 jelű ütemek, amelyeket a bemeneti változókkal nem lehet megkülönböztetni. Ezért szükség van ún. **memória,** vagy **szekunder elemre** (q). Példánkban az 1 és 3 ütemek (állapotok) megkülönböztetéséhez egyetlen szekunder elemre van szükség. A sorrendi hálózatokban a szükséges emlékező funkciók miatt a **szekunder elemek részt vesznek saját állapotuk fenntartásában,** vagyis **visszahatnak** saját bemenetükre. A szekunder változót is tartalmazó ütemdiagram és blokkséma a 6.3. ábrán látható. Az ábra alapján spekulatív úton felírható a gerjesztési (Q) ill. kimeneti függvény (Z):

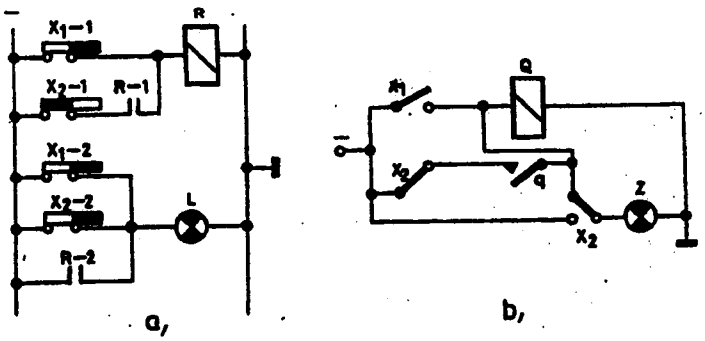
$$Q = X_1 \vee \overline{X_2}\, q \qquad Z = X_1 \vee \overline{X_2} \vee q\,.$$



6.3. ábra

A megvalósított relés hálózat a 6.4. ábra szerinti.

Az ábrából kiderül, hogy relés realizációban a **q a szekunder elemként működtetett relé záró érintkezőjét, a Q pedig a relé gerjesztését jelenti.** A 6.4. ábra kapcsolásában a **relé tartóáramkörös (öntartó)** kapcsolására ismerünk. Fontos leszögezni, hogy **sorrendi hálózatokban a relék mindig öntartó kapcsolásban szerepelnek.**



6.4. ábra

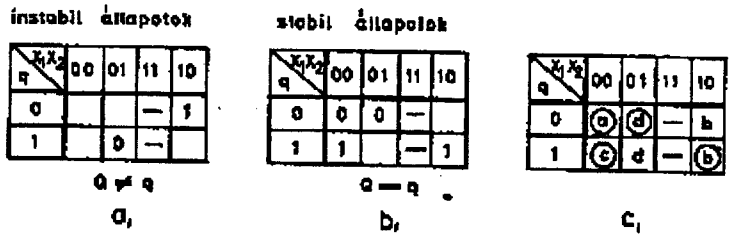
b) Állapot táblák

Láttuk a K-V táblák előnyeit a kombinációs hálózatok leírásánál. A sorrendi hálózatok táblázatos leírásához olyan KV táblát kell szerkeszteni, amely a bemeneti, ill. szekunder változók függvényében adja meg a gerjesztési (Q_i)ill. kimeneti (Z_i)értékeket. Szemléltetésül a 6.5. ábrán a "csillés" példa gerjesztési (Q) ill. kimeneti tábláját (Z) adtuk meg. Mivel két bemeneti (X_1, X_2)és egy szekunder változónk (q) van, így egy 3 változós KV táblát kapunk. A gerjesztési (Q) táblát tanulmányozva megállapíthatjuk, hogy vannak olyan cellák, amelyekben a $Q \neq q$. Az ilyen állapotot **instabil (átmeneti) állapotnak** nevezzük, hiszen ez nem maradhat fenn sokáig. A másik fajta állapotot a $Q = q$ egyenlet jellemez. Ez az állapot **stabil állapot**, amely a **bemeneti változók megváltozásáig fennmarad.**

	Q		X_1		Z		X_1	
q \ X_1X_2	00	01	11	10	00	01	11	10
0	0	0	—	1	0	1	—	1
1	1	0	—	1	1	1	—	1
			X_2				X_2	

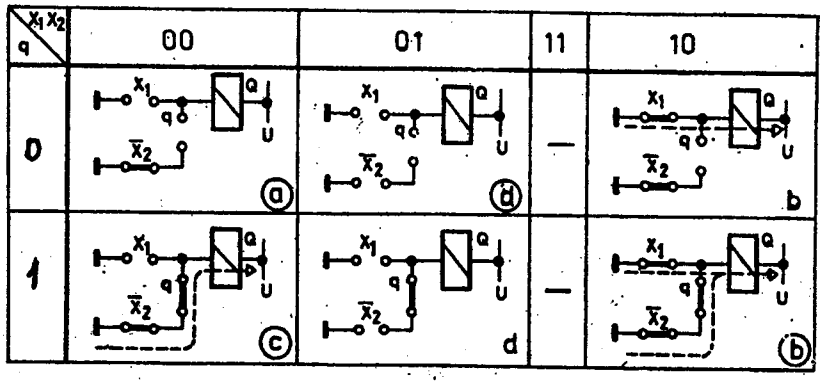
6.5. ábra

A 6.6. ábrán a stabil ill. instabil állapotokat külön is megrajzoltuk a) b). Szokás az egyes állapotokat betűkkel jelölni. Így az ún. **állapot táblát** kapjuk, amely az állapotok sorrendjéről is tájékoztat. Figyeljük meg, hogy az állapot táblán a **bemeneti változók állapotának megváltozása a táblán vízszintes irányú, míg a szekunder változók állapotának megváltozása a táblán függőleges irányú** mozgást jelent 6.6.c) ábra.



6.6. ábra

Az állapot táblás leírási mód megértésének segítését célozza a 6.7. ábra, amelyen rendhagyó módon az egyes állapothoz tartozó kapcsoló állásokat is berajzoltuk. A sorrendi hálózatok leírásánál még két további táblát szoktak megadni: az átmeneti táblát és a változás táblát. Az **átmeneti táblán** nyilakkal jelölik az egymás után következő stabil és instabil állapotokat. (6.7. ábra).

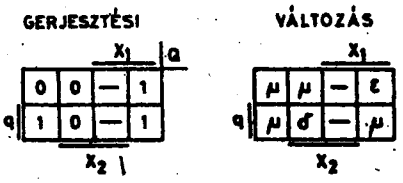


6.7. ábra

Gyakran annak ismerete szükséges, hogy a szekunder elemek milyen változáson mennek át. Ennek leírására a **változás tábla** szolgál. Egy szekunder elem az n-edik ütemben felvett értéke ismeretében az n+1-edik ütemben négyféle változáson mehet keresztül. Ezek:

qn	qn+1	A változás típusa
0	0	μ_0
0	1	ε
1	1	μ_1
1	0	δ

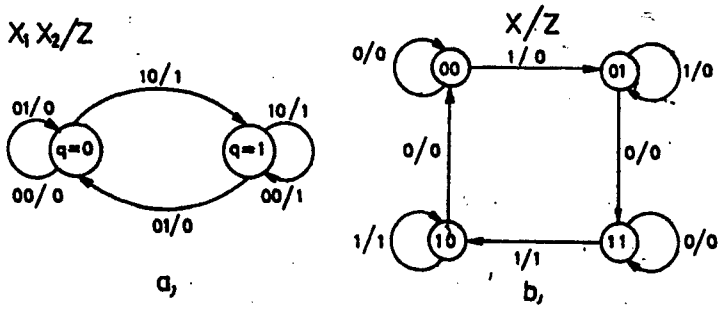
A **változás táblát** a gerjesztési tábla alapján tölthetjük ki. Példaként a „csillés" példa változás tábláját a 6.8. ábrán rajzoltuk meg. Minden szekunder elemhez külön változás tábla tartozik.



6.8. ábra

c. Állapotdiagram

Az **állapotdiagramon** a rendszer belső állapotait ($q_1...q_n$ kombinációit) rendszerint körökkel jelölt csomópontok ábrázolják. A csomópontok között elhelyezkedő irányított vonalak az állapotok közötti átmeneteket jelölik. Az átmeneteket jelző vonalak mellett gyakran feltüntetik egy tört számlálójában a bemeneti változók azon értékkombinációját, amely az átmenetet kiváltotta, a nevezőben pedig a kimeneti változók kívánt állapotát: $X_1...X_n/Z_1...Z_p$. Fentiek illusztrálására a „csillés" példa állapotgráfját a 6.9.a) ábrán, valamint a bináris osztó állapotgráfját a b) ábrán rajzoltuk meg.



6.9. ábra

6.2. Sorrendi hálózatok csoportosítása

Az állapotváltozások előidézésének módja szerint megkülönböztetünk:

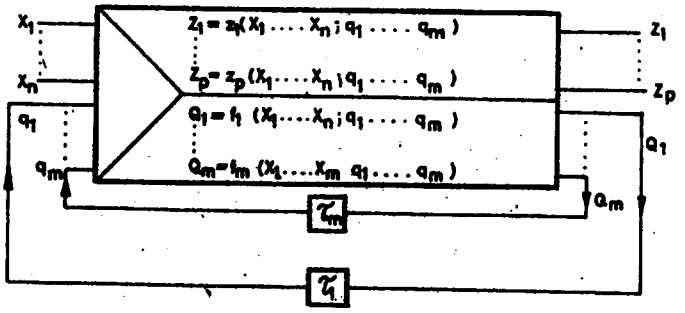
a) aszinkron

b) szinkron sorrendi hálózatot.

a) Aszinkron sorrendi hálózat

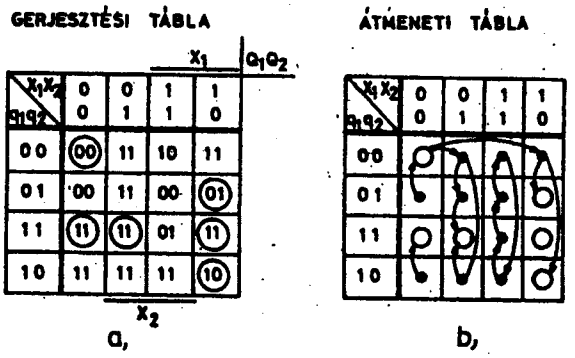
Aszinkron jellegű a rendszer, ha az állapotváltozásokat a bemenő változók értékkombinációjának megváltozása közvetlenül idézi elő, és az időviszonyokat csak a szekunder elemek késleltetései determinálják. Aszinkron sorrendi hálózatok esetén csak akkor léphet fel új bemeneti kombináció, ha a meglévő bemeneti kombináció mellett már stabil állapot alakult ki. Az aszinkron sorrendi hálózatoknál feltételezzük, hogy az egymást követő bemeneti kombinációk szomszédosak, azaz egyszerre csak egy bemenő jel változik meg. Az aszinkron sorrendi

hálózatok stabil állapotaikat instabil állapotokon keresztül érik el. Ez általában több egymás utáni állapotváltozást jelent. Így az **instabil állapotok kizárólag azt a célt szolgálják, hogy a megfelelő stabil állapotba vezessék a rendszert.** A visszacsatolással realizált aszinkron sorrendi hálózat általános sémája a 6.10. ábra szerinti.



6.10. ábra

Az aszinkron sorrendi hálózatokban káros jelenségek léphetnek fel. Ezek: versenyhelyzet ill. lényeges hazárd. **Versenyhelyzet** az egynél több szekunder elemet tartalmazó aszinkron sorrendi hálózatokban alakulhat ki a **szekunder elemek különböző késleltetési ideje miatt.** Ennek vizsgálatára tekintsük a 6.11. ábrát, amelyen egy aszinkron sorrendi hálózat gerjesztési és átmeneti tábláját adtuk meg.

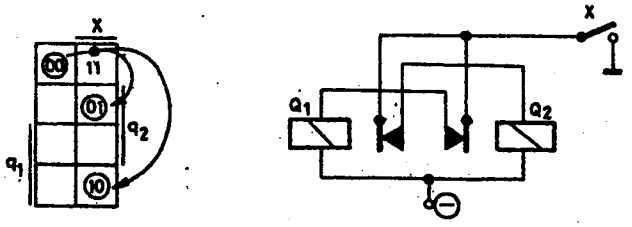


6.11. ábra

Induljunk ki az $m_0^4\left(q_1, q_2, X_1, X_2\right)$ értékkel jellemzett stabil állapotból. Tételezzük fel, hogy a bemeneten X_1 0-ról 1-re változik. Ekkor mindkét szekunder változó egyidejűleg vezérlést kap $\left(m_2^4\right)$. Előírásszerűen a szekunder elemnek az m_{14}^4 értékkel jellemzett stabil állapotba kellene kerülni. Ez azonban

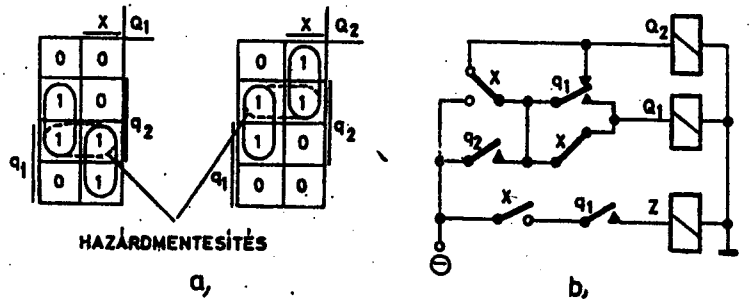
a legtöbb realizációnál nem fog bekövetkezni. Ennek oka a szekunder elemek különböző működési sebessége. Ha pl. Q_1 relé gyorsabb ($t_{m1} < t_{m2}$, ahol t_m a megszólalási idő), akkor a hálózat az m_{10}^4 értékkel jellemzett stabil állapotba kerül és ott is marad. Ha pedig $t_{m2} < t_{m1}$, akkor az m_6^4-nak megfelelő stabil állapot alakul ki és ez fenn is marad. Látható, hogy akár Q_1, akár Q_2 működik gyorsabban, a rendszer helytelen végállapotba kerül. **Egy kapcsolórendszer állapotában azt az esetet, amikor egynél több szekunder elemnek kell állapotát egyidejűleg megváltoztatni, versenyhelyzetnek mondjuk. Más szavakkal, ha az aszinkron sorrendi hálózatok belső állapotainak megváltoztatásakor két egymás után következő állapot kódja közti Hamming távolság $D \geq 2$, akkor versenyhelyzet keletkezik.** Ez a versenyhelyzet létrejöttének feltétele. A versenyhelyzet lehet kritikus és nem kritikus. **Kritikusnak** mondjuk a versenyhelyzetet, ha a végső stabil állapot függ az állapotváltozások sorrendjétől. A kritikus versenyhelyzet kialakulásának **szükséges feltétele**, hogy az állapot tábla kérdéses instabil állapotot tartalmazó oszlopában legalább **két stabil** állapot legyen. Példánkban a versenyhelyzet feltétele az m_2^4 minterm instabil állapotára teljesül. Ebben az oszlopban $\left(X_1 X_2 = 10\right)$ három db stabil állapot van, tehát a kritikus versenyhelyzet szükséges feltétele is adott. Láttuk, hogy a kialakuló végleges stabil állapot függ az állapotváltozások sorrendjétől. Ez a versenyhelyzet tehát **kritikus**. Végezzük el a fenti analízist az $X_1 X_2 = 01$ oszlopra is. Ha az alap állapotban $\left(m_0^4\right)$ lévő hálózat bemenetén $X_2 = 1$ vált, akkor mindkét szekunder elem gerjesztést kap (ld. az m_1^4-et). A versenyhelyzet feltétele tehát most is teljesül, mert a két stabil állapot kódja közti Hamming távolság 2. Azonban a kritikus versenyhelyzet szükséges feltétele nem teljesül, mivel az $X_1 X_2 = 01$ oszlopban csak egy stabil állapot van. Az ábrából is látható, hogy a rendszer végül is - az állapotváltozások sorrendjétől függetlenül - **többszörös átmeneteken** keresztül a kívánt végállapotba kerül $\left(m_{13}^4\right)$. **A versenyhelyzet nem kritikus, ha a hálózat a belső állapotváltozások sorrendjétől függetlenül egy és csakis egy végállapotba kerül.** Sorrendi hálózatok működésénél felléphető káros jelenség az **oszcilláció** is. Erre az $X_1 X_2 = 11$ oszlopban találhatunk példát. Ez gyakorlatilag azt jelenti, hogy a hálózat $X_1 X_2 = 11$ bemeneti kombináció

esetén oszcillál. Az egymást követő instabil állapotok sorrendje az átmeneti tábláról leolvasható. **Hangsúlyozni kívánjuk, hogy a kritikus versenyhelyzet és az oszcilláció helytelen működést eredményeznek, ezért megengedhetetlenek. A nem kritikus versenyhelyzetet eredményező többszörös átmenetek a hálózatok tervezésénél előnyösen felhasználhatók.** Érdekességként említjük, hogy két relé közül a gyorsabb meghúzású relé kiválasztható a versenyhelyzeten alapuló kapcsolással (6.12. ábra). X kapcsoló zárása után mindkét relé a másik nyugalmi érintkezőjén kap gerjesztést. Így az egyik relé meghúzása esetén a másik áramköre megszakad, és az már nem tud meghúzni. Az ábrán a gerjesztési táblát is feltüntettük.



6.12. ábra

Lényeges hazárd az aszinkron sorrendi hálózatok gerjesztési függvényében léphet fel. Ismeretes, hogy a kombinációs hálózatokban fellépő statikus hazárd (1 vagy 0) a bemeneti kombináció megváltozásának idején a kimeneti változó értékét rövid időre meghamisítja. Ezt az időt a hálózat belső késleltetései determinálják. Ezen idő eltelte után **visszaáll az előző kimeneti érték.** Vizsgáljuk meg az aszinkron sorrendi hálózatok **gerjesztési függvényében** fellépő statikus hazárd tulajdonságát. E célból realizáljuk a bináris osztó gerjesztési függvényét (6.13. ábra).



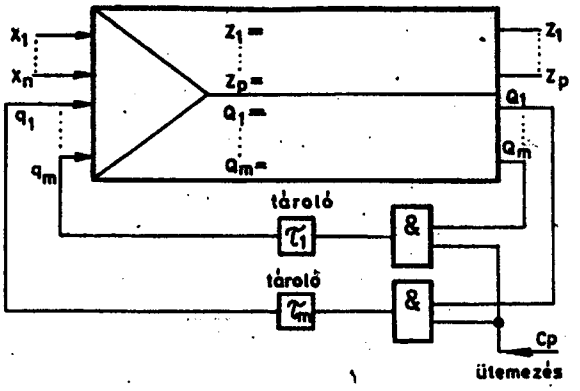
6.13. ábra

A hazárd vizsgálatot grafikus vagy algebrai módszerrel könnyen elvégezhetjük. Mind a Q_1, mind a Q_2 statikus hazárdot tartalmaz. Példaként a Q_2 relé áramkörét vizsgálva megállapíthatjuk, hogy a tartó áramkörben $\left(\bar{x}q_2\right)$ szerepel

a meghúzató $\left(x\bar{q}_1\right)$ egyik elemének (X) tagadottja. Amennyiben a Q_2 relé elengedési ideje összemérhető ezen változó (X) átváltási idejével, úgy **a tartó áramkör nem tud kialakulni, azaz a rendszer nem a kívánt belső állapotba kerül. Az aszinkron sorrendi hálózatok gerjesztési (vezérlési) függvényében fellépő statikus hazárdokat lényeges hazárdoknak nevezzük.** Míg a kombinációs hálózatokban a statikus hazárdok csak az **átmenet pillanatában** vezettek hibás eredményre, az aszinkron sorrendi áramköröknél a hálózatot **nem kívánt végső állapotba** juttatják. Ezért a **gerjesztési függvények nem tartalmazhatnak statikus hazárdot.** A lényeges hazárdot célszerűen választott átfedő tömbökkel küszöbölhetjük ki.

b) Szinkron sorrendi hálózatok

A szinkron jellegű sorrendi hálózatok belső állapotváltozásait ütemező jelek (órajelek) determinálják. A szinkron sorrendi hálózatok általános sémája a 6.14. ábra szerinti.



6.14. ábra

Az ilyen hálózatokban minden állapot stabil. A szinkron sorrendi hálózatokban tehát lényeges hazárd és/vagy versenyhelyzet nem léphet fel. Ezért a digitális rendszerek nagy része (mikroprocesszorok stb.) szinkron működésű. Megjegyezzük, hogy a szinkron sorrendi hálózatokban fellépő káros jelenséget a flip-flopok kapcsán ismertetjük.

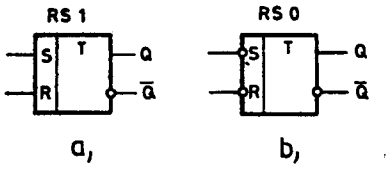
6.3. Flip-flopok

A sorrendi hálózatokban az emlékezés funkcióját **tároló elemek** valósítják meg. Ezen **elemi memóriák** két jól megkülönböztethető állapotuk révén **egy bit** információ tárolására alkalmasak. Külső jellel egyik állapotból a másikba billenthetők. Amikor a tároló elemeket relékből építették fel, az átbillenést hallani is lehetett. Erre utal az angol hangutánzó **flip-flop** elnevezés. A tároló elem **két kimenetén** mindig **ellentétes logikai érték** van. A flip-flop

megállapodás szerint akkor van igenleges azaz 1-es állapotban, ha **Q kimenetén IGEN** van. **Beírás alatt azt a műveletet értjük, amikor külső vezérlés hatására a flip-flopot Q = 1 állapotba billentjük.** Amennyiben a Q = 0, a tároló **törölt állapotban van. A törlés a beírással ellentétes művelet. Törléskor a flip-flopot külső jellel Q = 0 állapotba billentjük. Az egyidőbeni beírás és törlés értelmetlen. Az elektronikus tároló elemek a tápfeszültség bekapcsolásakor bármely állapotukat felvehetik.** A különböző típusú flip-flopok vezérlési módjukban (0 ill. 1, statikus ill. dinamikus), bemeneteik számában, áramköri felépítésükben térnek el egymástól. A következőkben csak a napjainkban leggyakrabban használt típusokat ismertetjük.

6.3.1. Az RS ill. inverz RS flip-flop

Az RS flip-flop jelképi jelölése látható a 6.15. ábrán. A flip-flop két kívülről vezérelhető bemenettel rendelkezik: S ill. R. Az **S bemenet beírásra** (set-beírás), az **R bemenet törlésre** (reset - törlés) szolgál. Amennyiben az S bemenetet hatásos jellel vezéreljük, - a billenési idő eltelte után - Q = 1 lesz. Az R bemenetre adott hatásos vezérlés Q = 0-át eredményez. Az R és S bemenet **egyidejűleg hatásos** jellel nem vezérelhető. Amennyiben az RS flip-flop nincs hatásos jellel vezérelve, úgy **megtartja előző állapotát.** A hatásos vezérlő jel egyaránt lehet az 1 ill. a 0. Az olyan RS flip-flop, melynek bemenetein a 0 a hatásos jel **inverz RS** flip-flopnak (RS0) mondjuk.



6.15. ábra

Az RS flip-flopok igazság táblázata a 6.1. táblázat szerinti.

6.1. táblázat

RS flip-flop (RS1)				Inverz RS flip-flop (RS0)	
R	S	Q_{n+1}	Funkció	Q_{n+1}	Funkció
0	0	Q_n	tárol (marad)	-	tiltott
0	1	1	beír	0	töröl
1	0	0	töröl	1	beír
1	1	-	tiltott	Q_n	tárol (marad)

A táblázatban a Q_n az FF vezérlés előtti, a Q_{n+1} pedig a vezérlés utáni állapotát jelöli. A fenti táblázat egyértelműen leírja a flip-flop működését, de nem adja meg a Q_{n+1} konkrét értékét (1 ill. 0).

Ehhez olyan táblázatot kell készíteni, amelyben a Q_n a független változók között szerepel. Így a 6.16. ábra szerinti KV táblákat kapjuk, ahonnan felírhatók a Q_{n+1} függvények mindkét FF esetére.

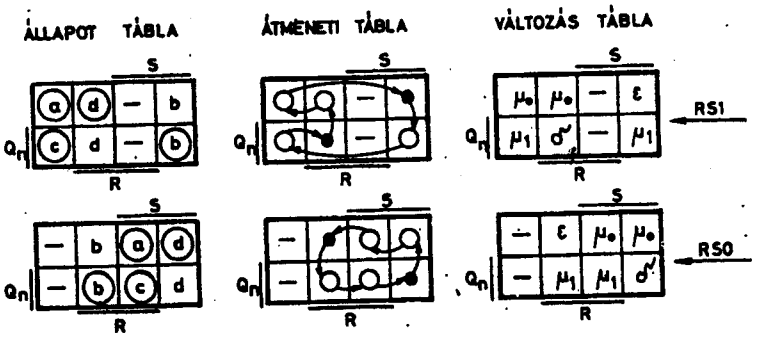
6.16. ábra

$$Q^{RS1}_{n+1} = S \vee \overline{R}Q_n \text{ ill. } Q^{RS0}_{n+1} = \overline{S} \vee RQ_n . \qquad (6\text{-}1)$$

Eszerint az RS1 FF Q kimenetén akkor van 1-es jel, ha vagy éppen beírás történik (S =1), vagy a Q értéke előzőleg már 1-es értékű és nincs törlés (R=0).

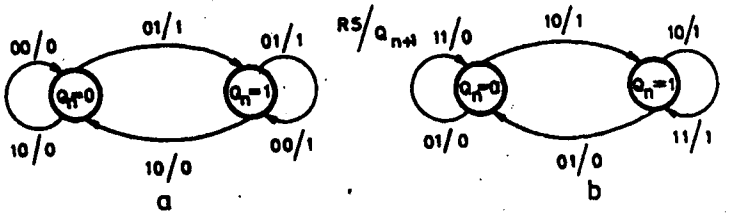
Vezérlési feltételek: RS1FF: R&S=0, RS0FF: R∨S=1.

A kétféle RS FF jellemző tábláit a 6.17. ábrán láthatjuk.



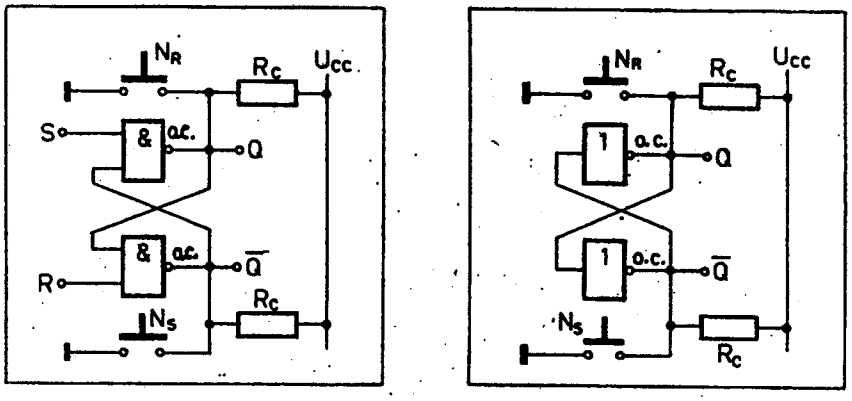
6.17. ábra

A 6.18. ábra az RS FF-ok állapotgráfját szemlélteti.



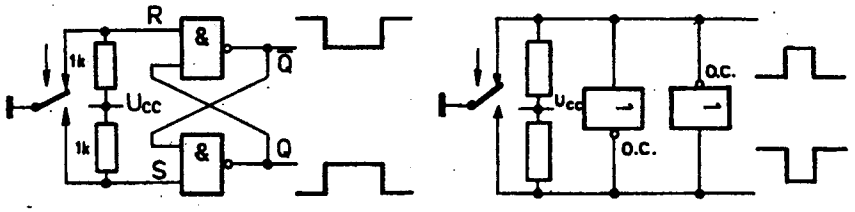
6.18. ábra

Az RS FF-ok realizációit szemlélteti a 6.19. ábra. Eszerint az RS1 FF két NOR, az RS0 FF két darab NAND kapuból visszacsatolás révén realizálható. Tanulmányozza az áramkör működését az idődiagram alapján! Természetesen a kapcsolás az NR és NS gomb nélkül is működik.



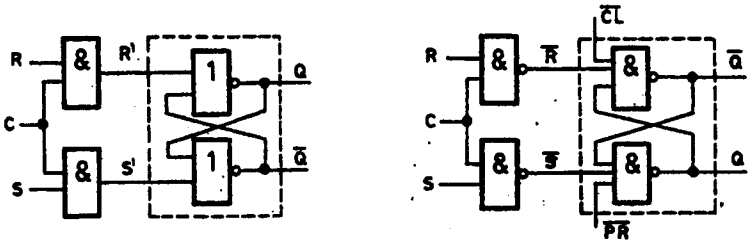
6.19. ábra

A **mechanikus kapcsolók pergését** igen gyakran RS FF-pal oldják meg. Két megoldást szemléltet a 6.20. ábra.



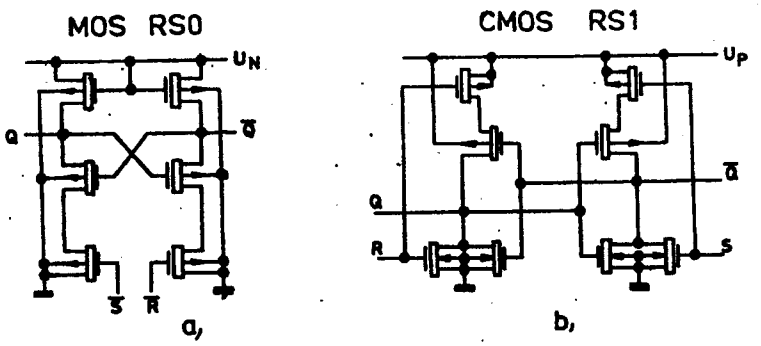
6.20. ábra

Órajellel szinkronizálható RS FF-ot kapunk, ha az RS1 FF bemeneteit ÉS kapuval látjuk el a 6.21.a) ábra szerint. Az RS0 FF-ot NAND elemmel kapuzva ugyancsak szinkronizálható RS FF-hoz jutunk b) ábra, melynek bemenetein az R & S = 1 vezérlés kerülendő és a U_1 jel a hatásos. A kapcsolást aszinkron jellegű (PR-preset, CL-clear) bemenetekkel is elláttuk. A tagadás jel arra utal, hogy mind a beírás, mind a törlés 0 jellel történik.



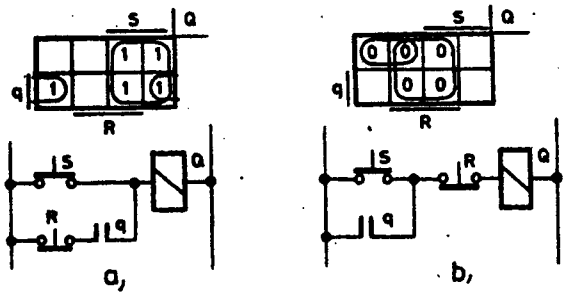
6.21. ábra

A 6.19. ábra analógiájára MOS NAND elemekkel felépített RS0 ill. CMOS NOR kapukból felépített RS1 FF látható a 6.22. ábrán. Fel kell hívni a figyelmet arra, hogy a statikus RAM memóriák alapcelláit RS0 ill. RS1 FF-ok alkotják.



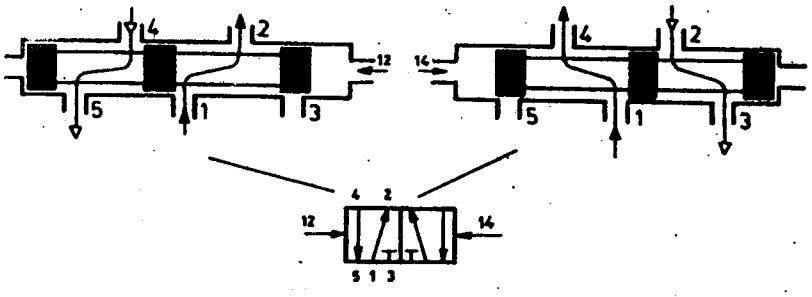
6.22. ábra

Láttuk, hogy a feszültség logikájú RS1 FF-oknál az R & S = 1 esetén mindkét kimeneten azonos logikai érték jelenik meg, ezért ez a vezérlés **tiltott**. A relés realizációnál a megegyező kimeneti érték a konstrukció miatt egyidejűleg nem jöhet létre. Így attól függően, hogy a gerjesztési tábla m_3, m_7 mintermjeibe határozottan 1-et vagy 0-át írunk, más-más tulajdonságú tároló elemet kapunk. A 6.23.a) ábra **beírásra**, a b, pedig **törlésre biztosít elsőbbséget**. A **törlésre (elengedésre) elsőbbséget biztosító relés tartó áramkört az iparban villamos hajtómotorok, működtető berendezések nyomógomb vezérlésű** be- és kikapcsolásánál széles körben alkalmazzák. **Az elengedésre elsőbbséget biztosító kapcsolást balesetvédelmi okból szabvány írja elő!** Figyeljük meg, hogy a két kapcsolás az R=S=1 esetet kivéve teljesen azonosan viselkedik. Az eltérés **mindkét nyomógomb egyidejű működése** esetén szembetűnő.



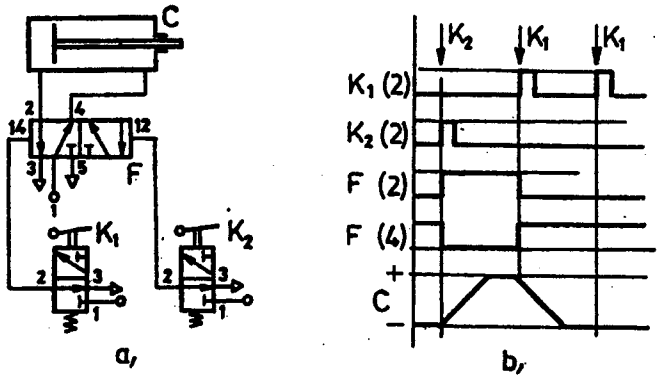
6.23. ábra

Az RS FF **pneumatikus realizálása** legtermészetesebben **ötútú szeleppel realizálható.** Az **ötútú szelep konstrukciójánál fogva memória** tulajdonsággal rendelkezik. Egy **síktolattyús 5/2-es szelep** áramlási útjait szemlélteti a 6.24. ábra. Az ábrából kitűnik, hogy az 1-2 ill. 1-4 utakon **átáramló levegő nyomása fenntartja** a szelep pozícióit. Ebben rejlik a memória jelleg. A beírás ill. törlés impulzus szelepekkel történik (12 ill. 14 vezérlés).



6.24. ábra

Pozitív vezérlésű kapcsolás látható a 6.25. ábrán. A K_1, K_2 alaphelyzetben zárt 3/2-es impulzus szelepek adják a nyomás impulzusokat az F 5/2-es bistabil útszelep 12 ill. 14 jelű vezérlő csatlakozóira. A működés a b, ábra szerinti diagram alapján követhető. A kapcsolásnak létezik **negatív vezérlésű** változata is. Ennél az átváltást a **nyomás megszűnte** váltja ki.

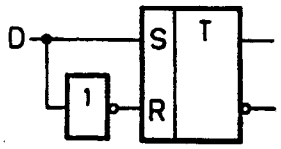


6.25. ábra

6.3.2. A D flip-flop

Az RS FF-ok kedvezőtlen tulajdonsága, hogy bemenetükön a vezérlési feltételt biztosítani kell. Ezt a hátrányt kiküszöbölhetjük, ha az RS FF bemenetére egy invertert kapcsolunk. (6.26. ábra). Így a flip-flopnak egyetlen bemenete marad: D.

D	Q_{n+1}
0	0
1	1

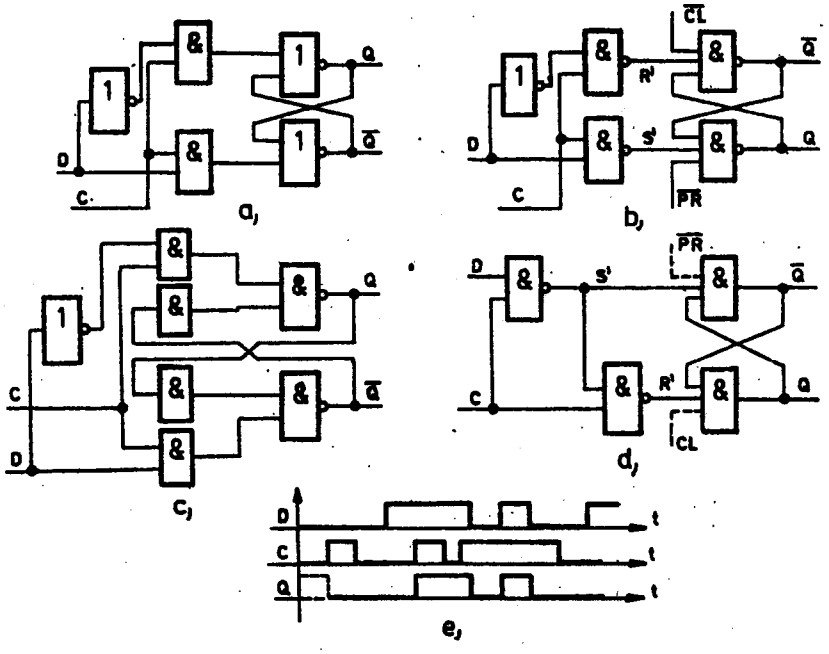


6.26. ábra

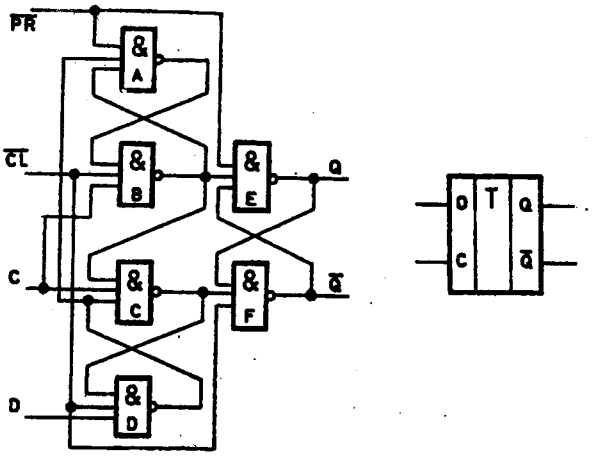
Karakterisztikus egyenlete: $Q^{D}_{n+1} = D$. Fenti egyenletben a FF előző állapota (Q_n) nem szerepel. Ez abból származik, hogy az inverter beiktatásával nemcsak a nem kívánt R=S=1 bemeneti vezérlést, hanem az R=S=0 kombinációt is megszüntettük. Ezért a 6.26. ábra szerinti kapcsolás nem flip-flop, mert nem képes információ tárolásra. Amennyiben a D FF-ot **szinkronizált RS FF-ból** alakítjuk ki a 6.27. ábra szerint, úgy a két órajel közötti időben megőrzi a bemenet állapotát. Az ilyen kapcsolást **D latch-nek nevezik.**

A 6.27. ábra a D latch-et 4 féle realizálásban mutatja be. A c) ábra szerinti felépítésű a 7475 típusú 4 db D latch-et tartalmazó IC, amelyet leggyakrabban a kijelző vezérlésénél ill. programozható be/ki eszközökben információ tárolásra használnak. A D FF-ok másik jellegzetes típusa az ún. **élvezérlésű D FF,** amelyet regiszterekben, számlálókban használnak. A 6.28. ábra szerinti kapcsolásban a D bemeneten lévő információ mintavételezése és a kimenet állapotának megváltozása egyaránt az **órajel pozitív élénél** következik be.

Ilyen felépítésű pl. az SN7474 típusú IC, amely élvezérlésű D FF-okat tartalmaz. A gyakorlatban igen fontos a D latch és a D FF közötti különbség ismerete, mert ennek hiánya hibás áramkör tervezést eredményezhet. Ezt segíti elő az is, hogy a katalógusokban az egyiket latch-nek, a másikat FF-nak nevezik.



6.27. ábra



6.28. ábra

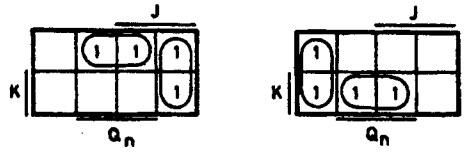
6.3.3. A JK flip-flop

A JK FF logikailag annyiban különbözik az RS FF-tól, hogy mindkét bemenet egyidejű hatásos vezérlése megengedett, ilyenkor a FF kimenete állapotot vált. A J bemenet beírásra, a K bemenet törlésre szolgál. A JK FF egyszerűsített működési táblázata a 6.2. táblázat szerinti. Figyeljük meg, hogy a táblázat első 3 sora megegyezik az RS FF-éval.

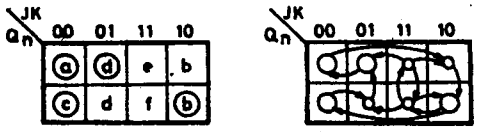
6.2. táblázat

JK flip-flop					Inverz JK flip-flop	
m_i^3	K	J	Q_{n+1}	Funkció	Q_{n+1}	Funkció
0	0	0	Q_n	tárol	$\overline{Q}_n$	komplementál
1	0	1	1	beír	0	töröl
2	1	0	0	töröl	1	beír
3	1	1	$\overline{Q}_n$	komplementál	Q_n	tárol

A karakterisztikus egyenlet felírásához írjuk be a Q_n konkrét értékét (0 ill. 1), így egy 8 soros táblázatot kapunk. Ennek KV táblái láthatók a 6.29. ábrán.



6.29. ábra

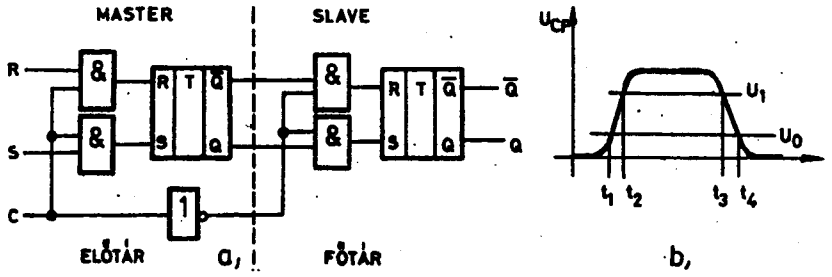


6.30. ábra

Karakterisztikus egyenletek:

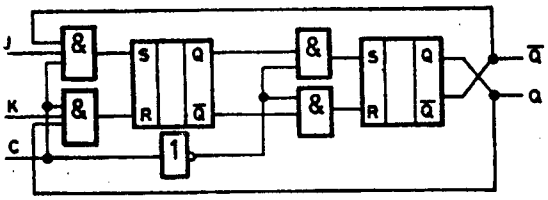
$$Q_{n+1}^{JK1} = J\overline{Q}_n \vee \overline{K}Q_n \qquad \text{ill.}$$
$$Q_{n+1}^{JK0} = \overline{J}\,\overline{Q}_n \vee KQ_n. \qquad (6\text{-}2)$$

A 6.30. ábrán vázolt táblázatból kitűnik, hogy a JK=11 oszlopban nincs stabil állapot, ezért mindkét bemeneti változó tartós vezérlése esetén **oszcilláció** lép fel. E káros jelenség kiküszöbölésére a JK flip-flopot **mester-szolga** vagy **élvezérelt** változatban készítik. **A mester-szolga (master-slave: ms)** FF-ok közbenső tárolós szinkron működésű FF-ok. Sematikus felépítésük a 6.31. ábra szerinti. A FF működése a CP függvényében (b, ábra) követhető. CP=0 esetén az R és S bemenetek le vannak választva a mester tárolóról. A CP=0→1 váltás hatására az R és S bemenetekre adott információ a mester fokozatba íródik, ugyanakkor a szolga FF bemenetei zárva vannak az inverter révén. CP=1 idején a mester FF tartalma a bemenet megváltozása révén változhat. A CP 1→0 átmenetekor a mester FF bemenetei zárnak, a szolga FF bemenetei pedig nyitnak, és a mester FF tartalma kijut a kimenetre.



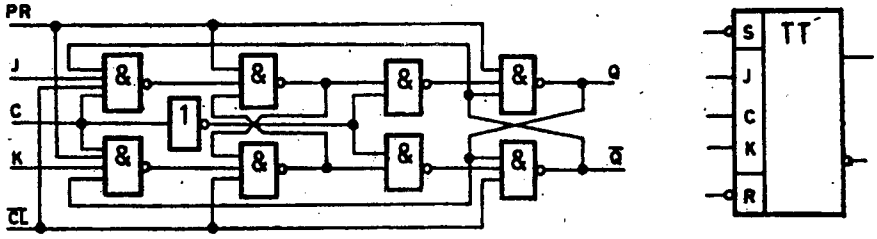
6.31. ábra

A kimeneten a váltás, tehát a CP 1→0 átmenetekor jelenik meg. Amennyiben a 6.31. ábrába a két FF-helyére berajzoljuk a NAND kapukat, úgy egy ms típusú RS FF-ot kapunk. Közbenső tárolós (ms) RS FF-ból JK FF-ot az $R = KQ$, $S = J\overline{Q}$ visszacsatolás révén készíthetünk a 6.32. ábrán vázolt séma szerint.



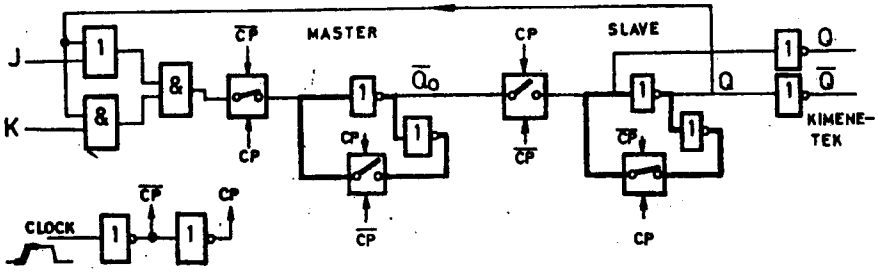
6.32. ábra

Ezen séma átrajzolásával egy JK ms FF kapcsolása a 6.33. ábra szerinti.



6.33. ábra

A **CMOS JK ms FF-ok** felépítése eltér a TTL változatokétól. A CMOS ms JK FF 2-2 inverterből és 2-2 átteresztő kapuból épül fel a 6.34. ábra szerint.



6.34. ábra

Az áramkör működése: mindkét FF akkor tárol, ha a visszacsatoló ágban lévő kapcsoló zár, a hurokban a fázisfordítás 360^{o} (pozitív visszacsatolás), ezért a kimenet állapota fennmarad. Eközben a soros kapcsoló szakadás. Beíráskor a soros kapcsoló bekapcsol, miközben a visszacsatoló ág kapcsolója nyit. A mester és a szolga FF ellentétesen működik és az egészet a CLOCK jel vezérli.

Megjegyzések:

- A JK FF-ból a J és K bemenetek összekötése révén ún. **T FF-ot** nyerünk, amely T=0 esetén tárol, T=1 esetén komplementál.
- A J=K=1 esetén ún. **komplementáló FF-ot** kapunk, amelyet leggyakrabban a számlálókban használnak.

6.3.4. Szinkron sorrendi hálózatokban fellépő káros jelenségek

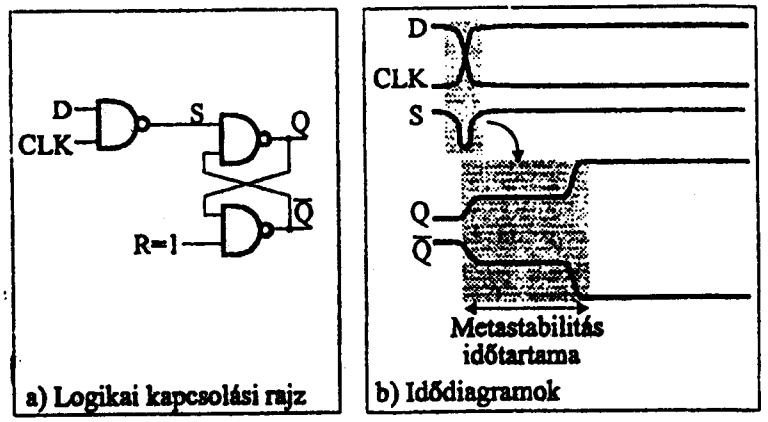
A szinkron sorrendi hálózatokban fellépő káros jelenségek közül az **órajel elcsúszás** (skew) ill. a **metastabilitás** jelenségét említjük.

a) Órajel elcsúszás

A jelenség akkor lép fel, ha a **közös órajel vezeték hosszú**, ezáltal a szinkronizált FF-ok nem azonos időben billennek. Ez az időeltolódáson túlmenően végzetes hibához is vezethet, mert időközben a FF-okat vezérlő állapotkombinációk megváltozhatnak, ami hibás működéshez vezet. Az órajel elcsúszás főként a huzalozott logikában fordul elő. Kiküszöbölése a minél rövidebb CLOCK jelvezeték ajánlott (IC-k elhelyezésének áttervezése) ill. az azonos késleltetéshez késleltető elemek beiktatása (jelmásoló).

b) Metastabilitás

A metastabilitás a szinkronizált FF-ok működésénél léphet fel. Ha egy FF adatbemenete (D) és CLK órajele közel egyidejűleg változik meg, akkor ún. **metastabilitás** léphet fel. A jelenség lényege az, hogy a tároló az egyidejűleg megváltozó bemenőjelek hatására nem a két stabilis állapotának valamelyikébe kerül, hanem egy közbülső (a logikai 0 ill. 1 állapotnak megfeleltetett feszültség tartományokon kívüli) állapotba kerül. Ezen metastabil állapot az áramkör fizikai paramétereitől függő (R, C) ideig maradhat fenn. Metastabilitás az aszinkron FF-oknál is felléphet. Gondoljunk arra az esetre, ha az RS FF (6.35. ábra) beírójelét abban a pillanatban szüntetjük meg, amikor a visszacsatoló ágon a hatásos jel még éppen nem alakul ki stabilan. Ilyenkor nehéz megmondani, melyik állapotba kerül a FF. A metastabil állapot a szinkronizált FF-oknál az adatbemenet és a CLK jel egyszerre történő megváltozásakor lép fel. A metastabilitás szemléltetésére tekintsük meg a 6.35. ábrán vázolt kapcsolást.



6.35. ábra

F statikus RS FF, amelynek beíró bemenőjelét egy D adatjel és egy Cl órajel NAND kapcsolatával állítjuk elő. Ha a két jel közel egyidejűleg megváltozik, akkor a FF S bemenetére egy rövid impulzus kerül, amely megindítja annak átbillenését, de ez az energia csak arra elég, hogy metastabil állapotba vigye a FF-ot. Ebben az állapotban aztán meghatározatlan ideig tartózkodhat.

A metastabil állapot kialakulását akkor kerülhetjük el, ha az órajel aktív élének közelében biztosítjuk, hogy az adatbemenetek ne változzanak.

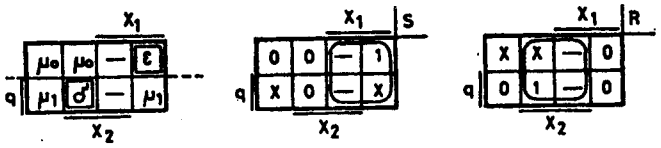
6.3.5. Flip-flopok vezérlési függvényeinek meghatározása

A FF-ok vezérlési függvényeinek meghatározása a gerjesztési tábla alapján lehetséges. Bonyolítja a problémát, hogy némely FF több bemenettel bír (pl. RS, JK), a gerjesztési táblában pedig egy változót egy bit reprezentál. Ilyenkor jó szolgálatot tesz a **változás tábla.** A FF változásaihoz egyértelműen hozzárendelhetjük a kívánt bemeneti jelkombinációt. Ezt szemlélteti a 6.36. ábra.

Q_n	Q_{n+1}	Vált. (Ip.)	R	S	D	J	K	T
0	0	μ_0	X	0	0	0	X	0
0	1	ε	0	1	1	1	X	1
1	1	μ_0	0	X	1	X	0	0
1	0	σ	1	0	0	X	1	1

6.36. ábra

Ez alapján készítettük el a bevezetőben ismertetett „csillés" példa vezérlési tábláit a 6.37. ábrán.



6.37. ábra

Vezérlési függvények: $S = X_1$ ill. $R = X_2$. Természetesen **több szekunder változó esetén valamennyi változóra el kell készíteni a változás táblákat.**

6.4. Regiszterek

A **regiszterek** viszonylag kis mennyiségű információ (pl. dekád, bájt, szó, dupla szó) tárolására **és gyors hozzáférésére** szervezett flip-flop csoportok. A regisztereket funkciójuk alapján két csoportba soroljuk:

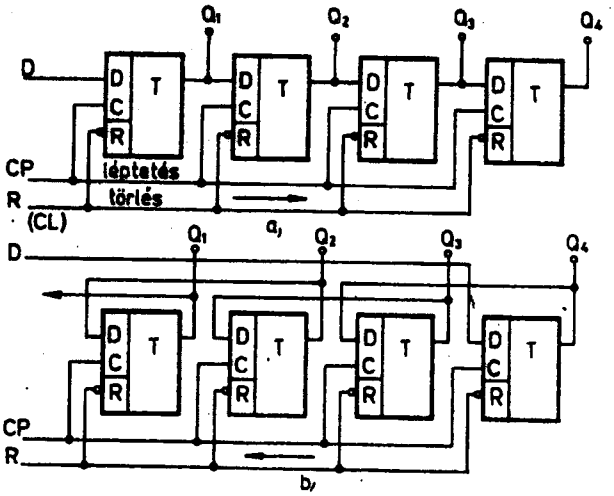
a) statikus tároló regiszterek
b) léptető (shift) regiszterek.

6.4.1. Statikus regiszterek

A statikus regiszterek az információ átmeneti tárolására alkalmasak, többnyire statikus RS ill. DFF-okból épülnek fel. Rendszerint közös funkcióval (pl. közös törlés) vannak ellátva. A 74100 típusú D latch 8 bites szó tárolására alkalmas.

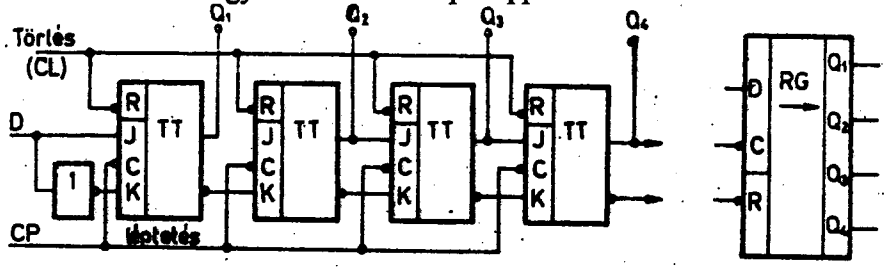
6.4.2. Léptető regiszterek

A **léptető regiszterek** az **információ tárolására és helyiértékenkénti léptetésére** alkalmasak. A léptetés iránya szerint megkülönböztetünk jobbra, balra ill. kétirányú (revezibilis) regisztereket. A **jobbra léptető regiszter** állapotegyenlete: $A_i^n = A_{i-1}^{n-1}$, ahol az alsó index a flip-flopnak a regiszterben elfoglalt helyét, a felső index az n-edik órajel előtti (n-1) ill. utáni (n) állapotot jelöli. A **balra léptető regiszter** állapotegyenlete: $A_i^n = A_{i+1}^{n-1}$. TTL rendszerben élvezérlésű D vagy ms JK flip-flopokból készíthető. D FF-okból felépített 4 bites jobbra (a) ill. balra (b) léptető regisztert szemléltet a 6.38. ábra.



6.38. ábra

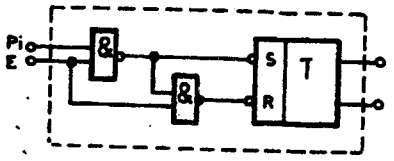
JK FF-okból épített 4 bites léptető regiszter és MSz jelképe látható a 6.39. ábrán. Az első FF-ot egy inverterrel D flip-floppá alakítottuk.



6.39. ábra

Kövesse végig egy 4 bites adat bevitelét a CP jel függvényében!

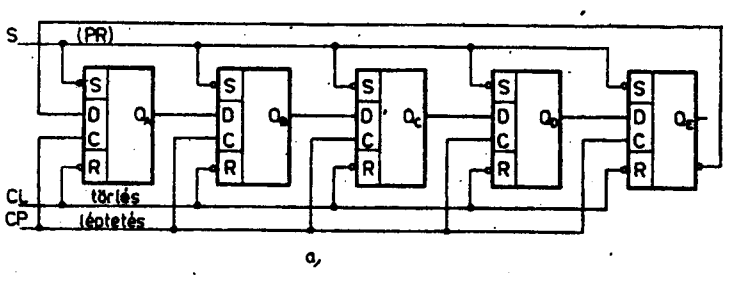
A **reverzibilis regiszter** i-edik elemének bemenetére a vezérléstől függően az (i+1)-edik (balra), illetve az (i-1)-edik (jobbra) flip-flop kimenetét kell kapuzni. Készítsünk olyan reverzibilis regisztert, amely M=0 esetén balra, M=1 esetén jobbra léptet! A regisztereket elláthatjuk párhuzamos beírási lehetőséggel. Ehhez a FF-ok PR és CL bemeneteit használhatjuk fel. Egy E=0 esetén léptetést, E=1 esetén párhuzamos beírást végző logika egy bitjét szemlélteti a 6.40. ábra.



6.40. ábra

6.4.3. Visszacsatolt regiszterek

Az eddig bemutatott regiszterek nem tartalmaztak visszacsatolást. **Visszacsatolt regisztert** úgy kapunk, hogy a regiszter flip-flopjainak kimenetéről egy kombinációs hálózaton keresztül visszacsatolunk a regiszter információ bemenetére. A visszacsatolt regisztereket gyakran számlálóként használják. Ilyen visszacsatolt regiszter pl. az ún. Johnson számláló (6.41. ábra).

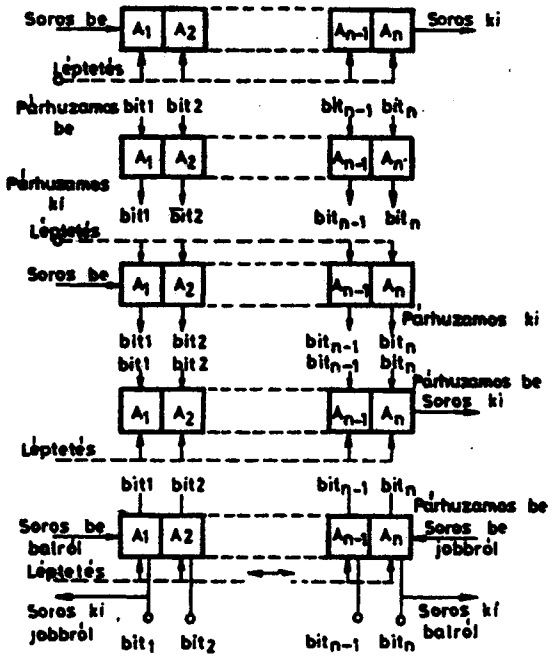


	Q_E	Q_D	Q_C	Q_B	Q_A
CL	0	0	0	0	0
CP1	0	0	0	0	1
CP2	0	0	0	1	1
CP3	0	0	1	1	1
CP4	0	1	1	1	1
CP5	1	1	1	1	1
CP6	1	1	1	1	0
CP7	1	1	1	0	0
CP8	1	1	0	0	0
CP9	1	0	0	0	0

b)

6.41. ábra

A visszacsatolt regisztereket gyakran **álvéletlen generátorként** alkalmazzák. A regiszterekkel felépített **üzemmód átalakítók** típusait a 6.42. ábrán foglaltuk össze.



6.42. ábra

6.5. Számlálók

A **számlálók** olyan sorrendi hálózatok, amelyek **impulzusok számlálására** és a **számlált érték tárolására** alkalmasak. A **számlálás alapját két jól elkülöníthető művelet képezi: a tárolás és az összegzés.** A számláláshoz tehát egyrészt **tároló elemekre,** másrészt olyan áramkörökre van szükség, amelyek elvégzik a **sorozatos összegzéseket**. A számlálókat többféle szempont szerint csoportosíthatjuk.

a) A számlálót alkotó FF-ok működése alapján megkülönböztetünk **aszinkron** ill. **szinkron** számlálókat. Az **aszinkron** számlálóknál az **FF-ok egymást billentik.** A **szinkron** számlálók esetén a FF-ok billenése egyidejűleg történik és az **állapotváltozásokat a FF-ok előző értéke által vezérelt kombinációs hálózat határozza meg.**

b) A számlálás iránya alapján megkülönböztetünk:

- **előre számlálókat,** melyek minden beérkező impulzus hatására eggyel növelik a tárolt értéket (inkrementálnak)
- **visszaszámlálókat,** amelyek minden beérkező impulzus hatására csökkentik a tárolt értéket (dekrementálnak)
- **kétirányú** (reverzibilis) számlálókat, amelyek a vezérléstől függően előre vagy visszaszámlálnak.

c) A számlálás **rendszere** szerint különféle kódrendszerek felhasználásával működő számlálókat különböztetünk meg: bináris, NBCD, Gray stb.

d) A számlálás **végértékének meghatározottsága** szempontjából beszélünk **fix hosszúságú** ill. **beállítható hosszúságú**, valamint **ismétlő** vagy **leálló** számlálóról.

6.5.1. Aszinkron számlálók

A legegyszerűbb felépítésű számlálók az **aszinkron bináris számlálók**, melyek a számlált értéket bináris kódban tárolják. Az aszinkron bináris számlálók T FF-okból építhetők fel. A **számlálás iránya** két dologtól függ:

- a FF-ok milyen átmenetre (0→1 ill. 1→0) billennek
- a billenést az előző FF Q vagy $\overline{Q}$ kimenete végzi.

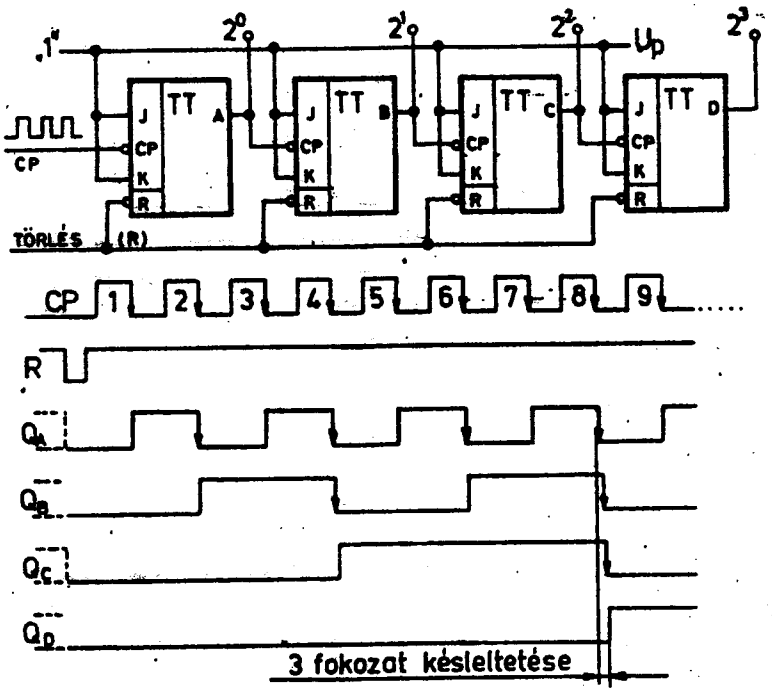
Négy bites (0...15) előre számlálót ábrázol a 6.43. ábra 1→0 átmenetre billenő JK ms FF-okkal. A működés az időidiagram alapján követhető. Az ábrán a **hatásos átmeneteket** nyíllal jelöltük.

Amennyiben a 6.43. ábrán $\overline{Q}_i$-CP_{i+1} összeköttetést készítünk, úgy **bináris visszaszámlálót** kapunk. Az aszinkron számlálók hátrányai:

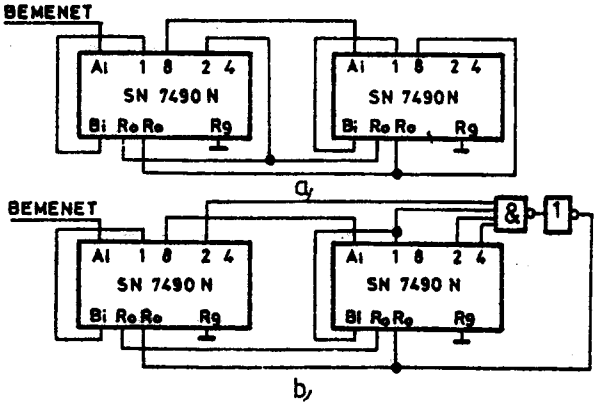
- **a számlálható frekvencia felső határa (f_{max}) a FF-ok számával** (n) **csökken,** mivel a FF-ok beállási ideje t = n · tpd.
- az aszinkron számlálók dekódolásánál a dekódolt kimeneteken az **átváltások idején hazárd** impulzusok jelenhetnek meg.

A TTL rendszerben a 7493 típusú 4 bites bináris számlálót említjük. A 6.43. ábra szerint kialakított számlálók 2^n-re periodikusak. Amennyiben nem 2^n-re periodikus számlálót akarunk készíteni, úgy **visszacsatolás** vagy a **FF-ok billenésének letiltása** révén kell a nem kívánt állapotokat kizárni. Erre példa a **7490 típusú dekádikus számláló**, amely 0...9-ig ismétlő számláló. Egy 82-ig (a) és egy 72-ig ismétlő számlálót szemléltet a 6.44. ábra.

Önmagát leállító számlálót úgy készíthetünk, ha a végérték elérésekor megakadályozzuk a 2^o súlyozású FF billenését a bejövő impulzus kapuzásával, vagy a J = K= 0-ra kapcsolásával. Az ilyen számláló a RESET jel hatására indul újra. **Beállítható hosszúságú** (programozható) számlálót úgy készíthetünk, hogy a számlált értéket a beállított értékkel összehasonlítjuk és ezen komparátor kimenetét csatoljuk vissza.



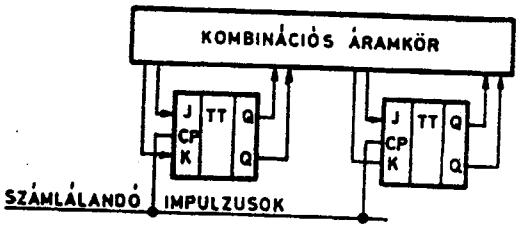
6.43. ábra



6.44. ábra

6.5.2. Szinkron számlálók

Az aszinkron számlálóknál fellépő késleltetési problémák **szinkron számlálók** alkalmazásával küszöbölhetők ki. A **szinkron számláló** valamennyi FF-jának CLOCK bemenete közösítve van, emiatt a flip-flopok egyszerre billennek. Sémája a 6.45. ábra szerinti.



6.45. ábra

A szinkron számlálók tervezése a fenti séma alapján a **FF-ok számának meghatározására** és a **kombinációs hálózat** tervezésére egyszerűsödik.

A módszer lépései:

1. lépés: a FF-ok számának meghatározása $n \geq N$, ahol N a modulus.
2. lépés: a FF-ok **állapot táblázatának** felvétele az előírt **kód, számlálási irány** stb. értelmében.
3. lépés: a FF-ok **vezérlési függvényének** meghatározása.
4. lépés: realizálás.

Példaként tervezzünk egy 4 bites szinkron bináris előre ismétlő számlálót.

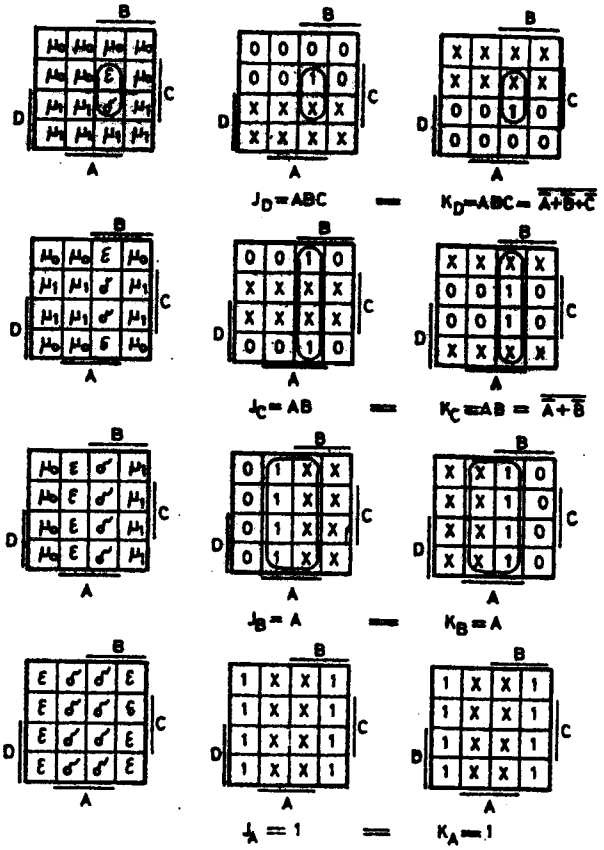
Állapot táblázat

6.3. táblázat

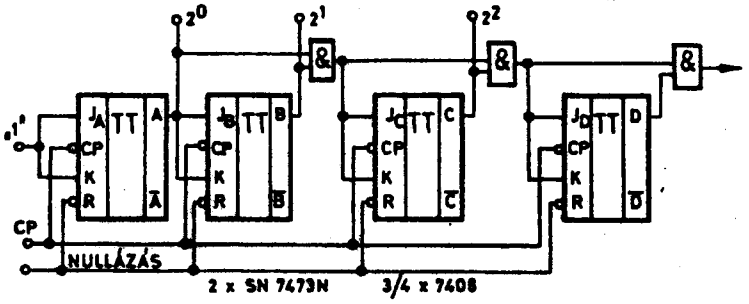
CLOCK	Előző állapot				Következő állapot				Változás típusa			
	D	C	B	A	D	C	B	A	D	C	B	A
0	0	0	0	0	0	0	0	1	μ_0	μ_0	μ_0	ε
1	0	0	0	1	0	0	1	0	μ_0	μ_0	ε	δ
2	0	0	1	0	0	0	1	1	μ_0	μ_0	μ_1	ε
3	0	0	1	1	0	1	0	0	μ_0	ε	δ	δ
4	0	1	0	0	0	1	0	1	μ_0	μ_1	μ_0	ε
5	0	1	0	1	0	1	1	0	μ_0	μ_1	ε	δ
6	0	1	1	0	0	1	1	1	μ_0	μ_1	μ_1	ε
7	0	1	1	1	1	0	0	0	ε	δ	δ	δ
8	1	0	0	0	1	0	0	1	μ_1	μ_0	μ_0	ε
9	1	0	0	1	1	0	1	0	μ_1	μ_0	ε	δ
10	1	0	1	0	1	0	1	1	μ_1	μ_0	μ_1	ε
11	1	0	1	1	1	1	0	0	μ_1	ε	δ	δ
12	1	1	0	0	1	1	0	1	μ_1	μ_1	μ_0	ε
13	1	1	0	1	1	1	1	0	μ_1	μ_1	ε	δ
14	1	1	1	0	1	1	1	1	μ_1	μ_1	μ_1	ε
15	1	1	1	1	0	0	0	0	δ	δ	δ	δ

Vezérlési táblák (6.46. ábra).

Megjegyezzük, hogy a FF-ok vezérlési függvényeit a 6.3.5-ben leírtak alapján határoztuk meg. A szinkron bináris előre számláló realizálása a 6.47. ábrán található.

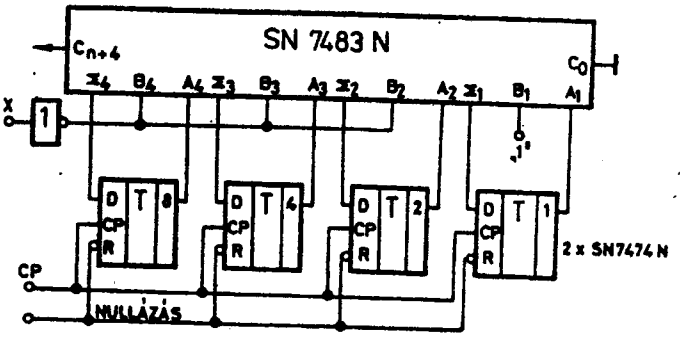


6.46. ábra



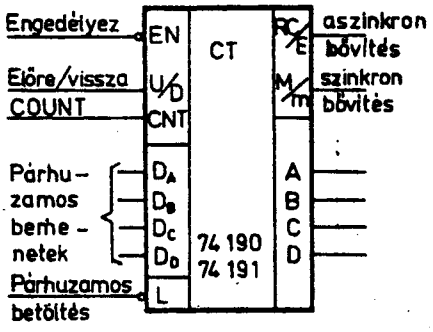
6.47. ábra

Szellemes megoldású reverzibilis szinkron bináris számlálót mutat a 6.48. ábra.



6.48. ábra

A FF-ok billentését egy 4 bites összeadó (7483) vezérli az alábbiak szerint: X=1 esetén 0001, X=0 esetén 1111 értéket ad a tárolt értékhez. A TTL ill. CMOS katalógusban több egy tokba integrált szinkron számlálót találunk. Közülük a 74190/91 típusú áramkört mutatjuk be. Az áramkör sémája a 6.49. ábra szerinti.



6.49. ábra

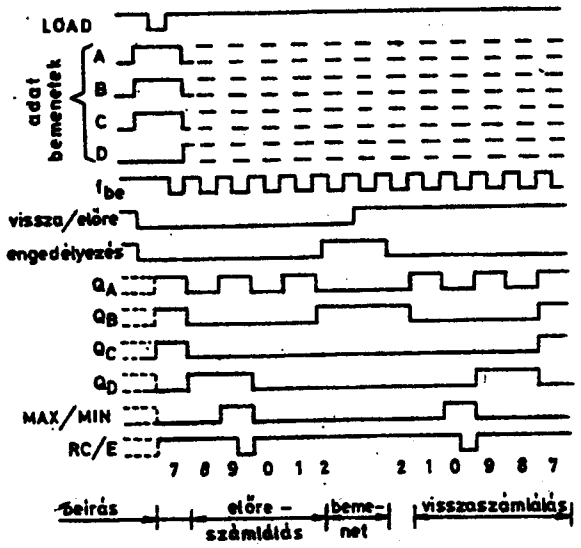
Működési táblázata:

U/D	E	L	Üzemmód
X	X	0	párhuzamos beírás D_A...D_D
X	1	1	nincs változás
0	0	1	előre számlálás
1	0	1	visszaszámlálás

Az áramkör működése a 6.50. ábra szerinti idődiagramon követhető.

A 74190 típusú NBCD kódú ill. 74191 típusú bináris kódú programozható számláló igen sokrétű alkalmazást tesz lehetővé. A számlálókat **frekvencia osztókban, digitális műszerekben** (frekvenciamérő, fordulatszám mérő, stb.), **A/D konverterekben, mikroprocesszorokban** (program számláló) stb. igen széles körűen alkalmazzák. A mikroprocesszortechnikában az előre

számlálásnak az **inkrementálás** (INR), a visszaszámlálásnak a **dekrementálás** (DCR) utasítások felelnek meg.



6.50. ábra

6.6. Félvezető alapú memóriák

A félvezető alapú memóriák a mai digitális berendezések nélkülözhetetlen elemei. Funkció szerint két nagy csoportba sorolhatók:

- RAM memóriák (Random Access Memories)
- ROM memóriák (Read Only Memories).

A memória **tárkapacitását** a tároló cellák számával adjuk meg pl.: 1024 x 8 bit = 1 Kbájt. A memóriában az információt sorszámozott rekeszekben helyezik el a könnyű visszakeresés céljából. Ezt a sorszámot **címnek** nevezik. A szervezés alapján bit, bájt, szó, dupla szó szervezésű memóriákat különböztetünk meg. **Hozzáférési időn** a címzés pillanata és a tárolt információ megjelenése között eltelt időt értjük. A félvezető alapú memóriák a címzés módja szerint hely szerint címzett (RAM, ROM) ill. **tartalom szerint címzett** (CAM) memóriákat különböztetünk meg. A hozzáférés módja szerint tetszőleges ill. soros hozzáférésű memóriákat különböztetünk meg. A tetszőleges hozzáférésű memóriáknál a hozzáférési idő nem függ az információ helyétől.

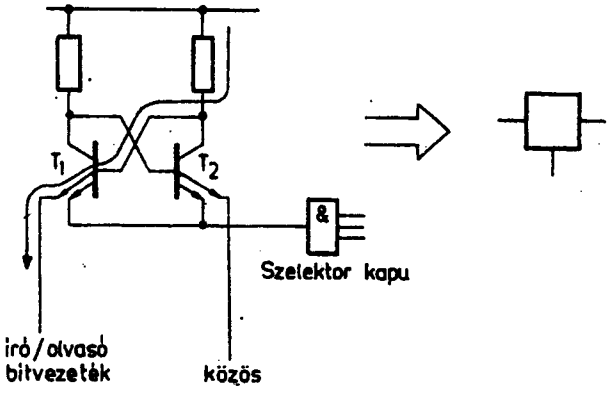
6.6.1. RAM memóriák

A közvetlen tetszőleges hozzáférésű memóriák a változtatható tartalmú, **írható, olvasható (R/W)** memóriák azon csoportját alkotják, amelyeknél az információ hozzáférési ideje annak fizikai helyétől (címétől) független. A RAM memóriák készülnek bipoláris, ill. MOS technológiával. **Bipoláris technikával csak**

statikus (SRAM), **MOS technikával statikus ill. dinamikus (DRAM)** egyaránt készíthető. Az **SRAM-ban az információt** egy címzési, írási, olvasási logikával kiegészített **RSFF**-ot tárolja. Ez azt jelenti, hogy egy 2 K x 8 bites SRAM 16384 darab FF-ot tartalmaz. A **dinamikus RAM** esetében a vezérlő elektróda és az **alapréteg közötti kapacitás** tárolja az információt **töltés** formájában. Ez a kapacitás a szivárgási áramok miatt kisül, ezért az információ megőrzése érdekében a kapacitás töltésének pótlására, azaz **információ frissítésére** van szükség.

a) SRAM memóriák

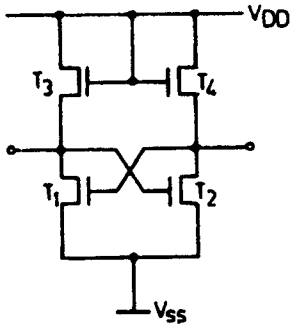
A **bipoláris SRAM** alapcellát szemlélteti a 6.51. ábra.



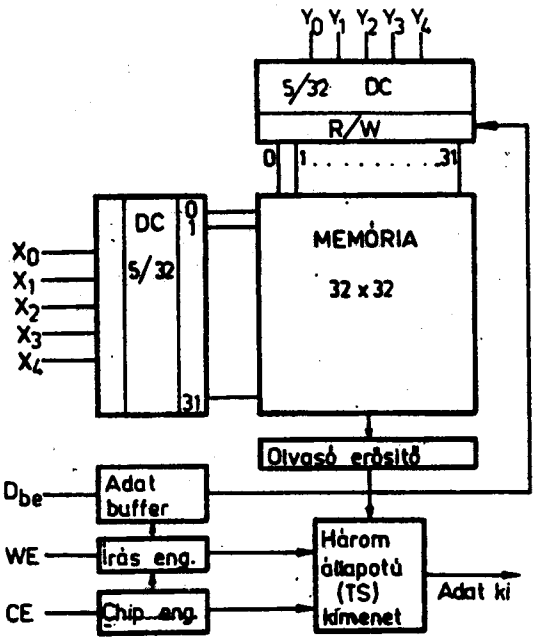
6.51. ábra

A **bipoláris SRAM** előnye az **igen rövid hozzáférési idő,** hátránya viszont a kis mértékű integrálhatóság és a **nagy teljesítmény igény** (1 mW/bit).
MOS SRAM alapcella látható a 6.52. ábrán.

Az ábrán jól felismerhető az RSFF. A **címzés, írás, olvasás megvalósításához** további MOS tranzisztorok szükségesek. Így léteznek 8 ill. 6 tranzisztoros alapcellák. Egy MOS SRAM cella blokkvázlata látható a 6.53. ábrán.



6.52. ábra



6.53. ábra

A **CMOS SRAM-okat** ellátják egy **tartalék üzemmóddal** (standby). Ilyenkor **igen alacsony a teljesítményfelvétel, az információt megtartja, de írás, olvasás nem lehetséges.** Összehasonlításul néhány jellegzetes SRAM típust és adatait a 6.4. táblázatban adtuk meg.

Gyors CMOS SRAM-ok

6.4. táblázat

Típus	Tárkapacitás	Szervezés	Címzési idő (ns)	Tápfeszültség (V)	Teljesítmény igény Aktív/passzív (mW)
MCM 6206C15	256 K	32 K x 8	15	5,0	825/100
MCM 62V06D	256 K	32 K x 8	25	3,3	200/3
CYC107	1 M	1 M x 1	12	5,0	825/275
MCM6249	1 M	1 M x 4	25	5,0	750/250

Alacsony teljesítmény igényű CMOS RAM-ok

6.5. táblázat

Típus	Tárkapacitás	Szervezés	Címzési idő (ns)	Tápfeszültség (V)	Teljesítmény igény Aktív/passzív (mW)
µPD43256B-B12	256 K	32 K x 8	120	3,3	150/0,2
µPD43256B	256 K	32 K x 8	55	5,0	250/10
µPD431000 A-B15	1M	128 K x 8	150	3,3	230/0,2
µPD43100A	1M	128 K x 8	70	5,0	350/10
µPD43100A	4M	512 x 8	70	5,0	375/10

BICMOS SRAM-ok

6.6. táblázat

Típus	Tárkapacitás	Szervezés	Címzési idő (ns)	Tápfeszültség (V)	Teljesítmény igény Aktív/passzív (mW)
µPD46458	256 K	32 K x 8	9	3,3	430/66
CY7B1094	256 K	64 K x 4	6	5,0	900/10
µPD461016	1M	64 K x 16	10	3,3	860/230
MCM6728A	1M	256 K x 4	8	5,0	875/100

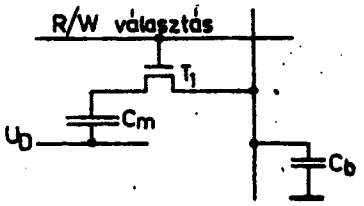
Bipoláris ECL SRAM-ok

6.7. táblázat

Típus	Tárkapacitás	Szervezés	Címzési idő (ns)	Teljesítmény igény Aktív/passzív (mW)
µPB1076LL	4 K	1 K x 4	3	1,82
µPB1084A	16 K	4 K x 4	6	1,66
µPD461016	1M	64 K x 16	10	860/230
MCM6728A	1M	256 K x 4	8	875/100

b) Dinamikus RAM-ok (DRAM)

Az **egy-tranzisztoros dinamikus MOS RAM** cella felépítése látható a 6.54. ábrán.



6.54. ábra

Az információt a C_m kapacitás tárolja, amelynek értéke néhány század pF, C_b pedig néhány tized pF. tehát $C_b >> C_m$. A kiolvasás a C_m töltésének elvesztésével jár, ezért minden kiolvasást egy **újraírásnak** kell követnie. A beírás vagy visszaírás a B vezetéken a nyitott tranzisztoron keresztül történik. A **DRAM-ok frissítés idején „hozzáférhetetlenek".** Frissítéskor a címeket számláló határozza meg. A frissítést időben kétféleképpen végezhetjük:

- összevontan (burst refresh)
- elosztottan (distributed refresh).

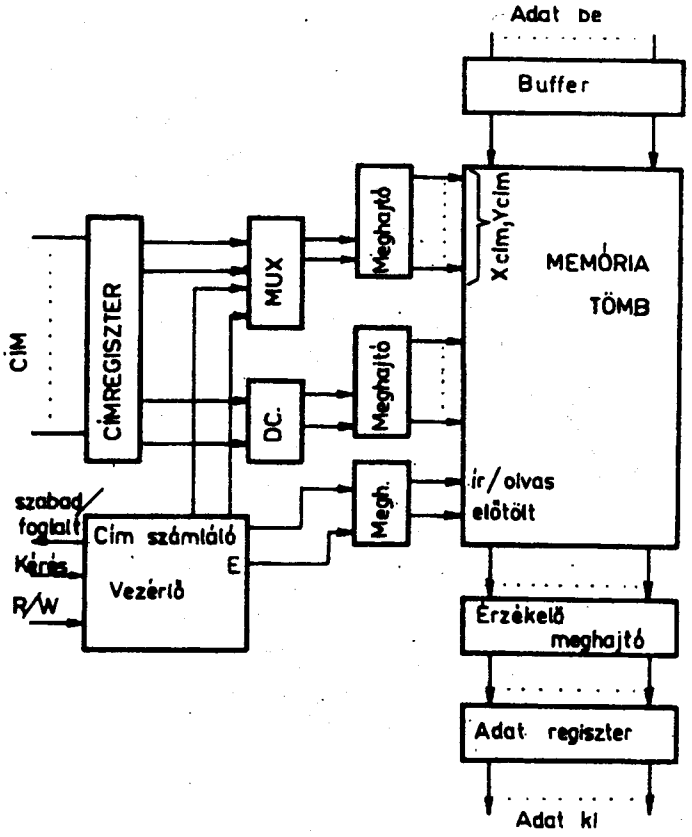
Az **összevont (folyamatos)** frissítésnél ~ 2 ms-onként végigléptetjük a számlálót, azaz elvégezzük a frissítést.

Az **elosztott üzemű frissítésnél** a 2 ms-ot feloszthatjuk annyi időszeletre, ahány frissítési ciklus szükséges és minden időszeletben egy frissítést végzünk. Nagy tárkapacitás esetén az **elosztott üzemi frissítés az előnyösebb,** mert nem tartja foglaltként a DRAM-ot olyan hosszú ideig. Példaként egy 4Mx1 CMOS DRAM egy sorának frissítési ideje 13 μs. A frissítés 1024 ciklusban van szervezve 4096 bitenként. A teljes frissítési idő 1024x130 μs. Egy DRAM rendszertechnikai felépítése látható a 6.54. ábrán. A memória cella a címet kaphatja a címregiszterből ill. a frissítést vezérlő számlálóról. Ezért van szükség a cím multiplexerre. Napjainkban a frissítést rendszerint a mikroprocesszor vagy számítógép végzi.

Tipikus DRAM szervezési módok (6.8. táblázat).

6.8. táblázat

Tárkapacitás	Szervezési alternatívák		
1M	1Mx1	256Kx4	64Kx16
4M	4Mx1	1Mx4	512Kx8
16M	16Mx1	4Mx4	2Mx8



6.55. ábra

CMOS DRAM-ok jellemző adatai (6.9. táblázat).

6.9. táblázat

Típus	Tárkapacitás	Címzési idő (ns)	Frissítési intervallum (ns)	Tápfeszültség (V)	Teljesítmény felvétel aktív/passzív (mW)
TMS46100	4M	70	16	3,3	200/1,7
TMS46100P	4M	70	128	3,3	200/1,0
µPD4216400	16M	60	64	3,3	265/2
µPD42516400	16M	60	128	3,3	265/0,5

6.6.2. ROM memóriák

Ebbe a csoportba a **csak kiolvasásra szánt** memóriák tartoznak. A ROM memóriák csoportosítása (6.10. táblázat).

6.10. táblázat

Típus	Technológia	Működés
Maszkprogramozott ROM	Bipoláris, MOS, CMOS	Gyárilag programozott ROM
PROM	Bipoláris	Elektromosan programozható (egyszer). Nem törölhető.
EPROM	MOS, CMOS	Elektromosan programozható, UV fénnyel törölhető.
EEPROM (E^2PROM)	CMOS	Elektromosan programozható és törölhető. A törlés rendszerint bájtonként történik.
Flash EPROM	CMOS	EEPROM, amely nem törölhető bájtonként, hanem chipenként.
EAROM	CMOS	Elektromosan törölhető és programozható ROM. Működése az EEPROM-hoz hasonló.

a) **Maszkprogramozott ROM memóriák** rendszerint tranzisztoros (bipoláris vagy MOS) mátrixból épülnek fel. Az 1 ill. 0 információt az **emitter és a bázis közötti összekötés vagy szakadás** képviseli, amit lézertechnológiával alakítanak ki. Példák maszkprogramozott ROM alkalmazásokra: mikroszámítógép monitor program, PLC monitor, stb.

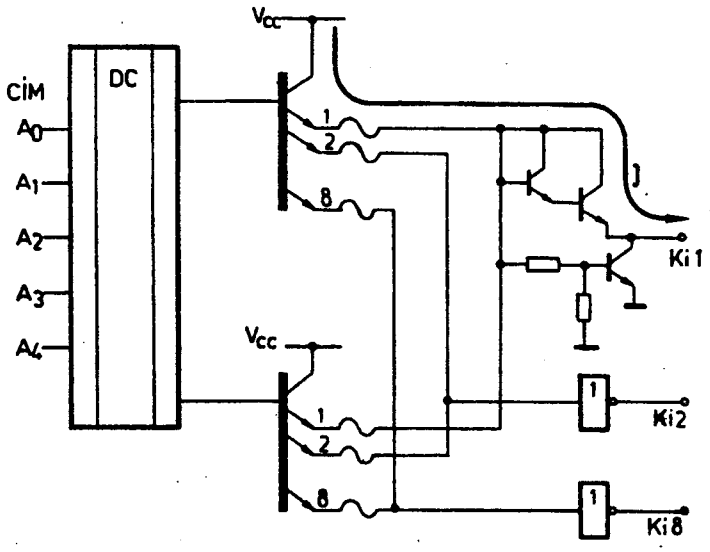
b) **PROM memóriák**

A PROM-ok csak **bipoláris technikával** készíthető, a felhasználó által egyszer programozható, nem törölhető ROM-ok. Programozása történhet:

- erre a célra kiképzett **ellenállás maszkok** megszakításával (kiégetésével)
- erre a célra kialakított **záróirányú p-n átmenet átütésével** létrehozott rövidzárral.

Az elégethető ellenállás anyaga többnyire Ni-Cr vagy polikristályos Si. A bipoláris PROM programozásánál fellépő áramot mutatja a 6.56. ábra.

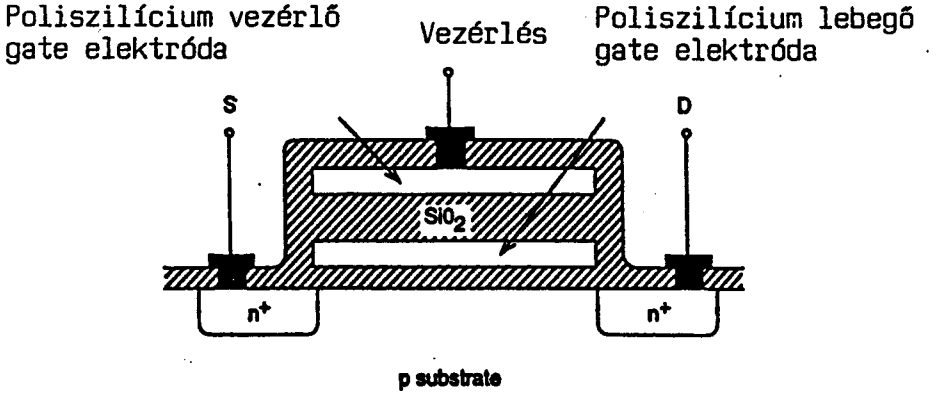
A ~ jel a **kiégethető biztosítékra** utal. A PROM programozásakor szükséges nagyobb teljesítményhez a **programozás idején** nagyobb tápfeszültségre kell kapcsolni a programozó készülék segítségével.



6.56. ábra

c) EPROM memóriák

Az EPROM memóriák a **felhasználó által törölhető és újraprogramozható** ROM-ok. Az EPROM memóriák alapeleme a **lebegő vezérlő elektródájú MOS** (FAMOS) tranzisztor. A lebegő vezérlő elektróda (nincs kivezetve!) polikristályos Si anyagú és minden oldalról igen jól szigetelő oxidréteg veszi körül. A kiváló szigetelés miatt a lebegő vezérlő elektródára juttatott töltés akár évtizedekig is megmarad. A **beírás**, a lebegő vezérlő elektróda feltöltése a drain elektródán keresztül történik. Az EPROM memóriák törlése - azaz a lebegő vezérlő elektródán tárolt töltések eltávolítása, vagy **ultraviola** (UV) besugárzással (EPROM) vagy **elektromos jellel** (EEROM, EAROM) történhet. Utóbbi esetben két vezérlő elektródával látják el: az egyik nincs kivezetve, a másik egy control kivezetésre csatlakozik (6.57. ábra).



6.57. ábra

Néhány EPROM típus jellemzői (6.11. táblázat).

6.11. táblázat

Típus	Tárkapacitás	Szervezés	Címzési idő (ns)	Tápfeszültség (V)	Programozási feszültség (V)
TMS27C240	4M	256Kx16	100	5	13
µPD27C800	8M	1Mx8	150	5	12,5

Flash EEPROM-ok (6.12. táblázat).

6.12. táblázat

Típus	Tárkapacitás	Szervezés	Címzési idő (ns)	Programozási feszültség (V)
TMS28F512	256K	64Kx8	100	12
TMS28F210	1M	64Kx16	100	12
TMS28F040	4M	512x8	100	12
NM29N16	16 M	2Mx8	80	-

CMOS EEPROM-ok (6.13. táblázat).

6.13. táblázat

Típus	Tárkapacitás	Szervezés	Címzési idő (ns)
µPD28C804	4K	512Kx8	200
µPD28C64	64K	8Kx8	150
µPD256K	256K	32x8	200

6.6.3. Memóriák bővítése

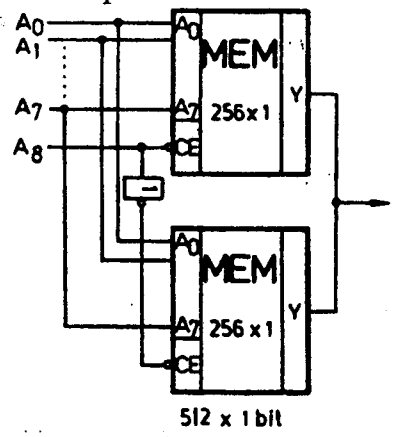
A memóriák bővítésére akkor van szükség, ha az egy tokban lévő tárkapacitás nem elegendő az adott feladat megoldásához.
A bővítés lehet:

a) címnövelő
b) szóhosszúság növelő
c) vegyes (cím és szóhosszúság növelő).

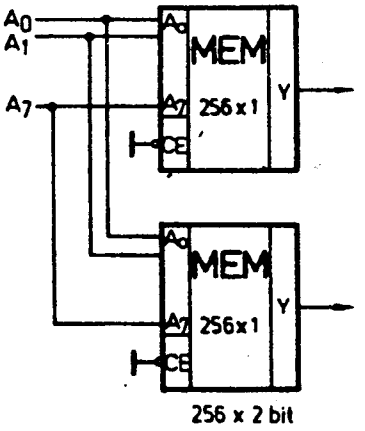
A bővítést megkönnyíti, hogy a memóriákat ellátják ún. chip engedélyező (CHIP ENABLE) bemenettel és három állapotú kimenettel.

a) **Címnövelő bővítést** a CE bemenetek tiltásával és a háromállapotú kimenetek összekötésével érhetünk el. Erre láthatunk példát a 6.58. sz. ábrán, ahol 2 darab 256x1 bites memóriával és egy inverterrel egy 512x1 bites memóriát készítettünk. Nagyobb címnövelő bővítéshez címdekódolót kell alkalmazni.

b) **Szóhosszúságnövelő bővítésnél** a címvezetékeket párhuzamosan kötjük. Erre mutat példát a 6.59. ábra.

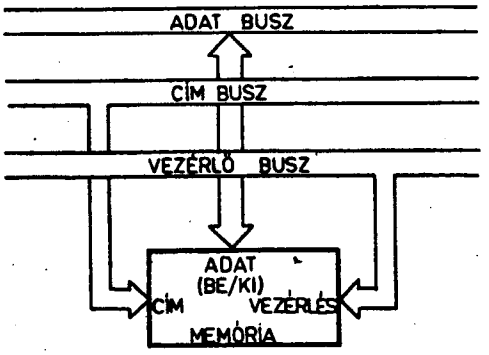


6.58. sz. ábra



6.59. sz. ábra

Példaként készítsünk 1 Kbájtos memória modult 256 x 4 bites RAM chipekből. Megállapíthatjuk, hogy a címzést négyszeresre, a szóhosszúságot kétszeresre kell növelni. Ez összesen 8 chipet jelent. A memóriák rendszertechnikailag a 6.60. ábra szerint csatlakoztathatók a digitális rendszerhez (mikroprocesszor, számítógép).



6.60. sz. ábra

Felhasznált irodalom

Ajtonyi I: Vezérléstechnika I. J 14-1417 Tankönyvkiadó, Budapest, 1987.
Ajtonyi I: Vezérléstechnika II. J 14-1417 Tankönyvkiadó, Budapest, 1987.
Ajtonyi I: Vezérléstechnika példatár J-1419 Tankönyvkiadó, Budapest, 1987.
Janovics-Tóth: A logikai tervezés módszerei Műszaki Könyvkiadó, 1971.
LJ. Herbst: Integrated Circuit Engineering. Oxford University Press, 1996.
Morris-Miller: Designing with TTL Integrated Circuits John Wiley, 1968.
Szittya: Logikai kapcsolástan BME-Tankönyvkiadó, 1971.
TEXAS: TTL receptek MK. 1978.

7. MIKROPROCESSZOROK, MIKROSZÁMÍTÓGÉPEK

Az 1970-es évek közepétől a digitális rendszerek fejlődése igen jelentős irányt váltott. A mai számítógépes kultúra kialakulását ennek a technikai forradalommal felérő változásnak köszönheti. Ebben a fejezetben a mikroprocesszorokkal, mikroszámítógépekkel kapcsolatos legfontosabb információkat foglaljuk össze.

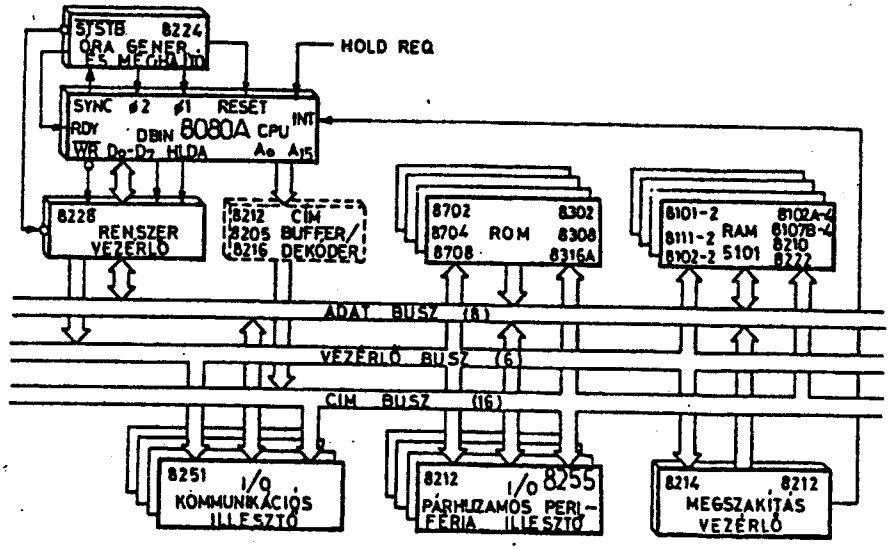
7.1. Bevezetés a mikroprocesszor technikába

Az MSI áramkörök kifejlesztése után a fejlesztők az LSI technika felé fordultak. Kifejlesztésre kerültek a RAM, ROM, EPROM memóriák, a három állapotú sínrendszer és az LSI technika. Ezek együttesen megteremtették a lehetőséget a digitális számítógép funkcióinak néhány chipen való integrálásához.

7.1.1. Mikroprocesszor történelem

A mikroprocesszorok története az 1970-es években kezdődött. 1971 júniusában jelentette meg az Intel cég a 4 bites 40004-es, majd ez év novemberében a 4040-es típust, majd ugyanezen év decemberében a 8 bites 8008-as típust. Ezen típusokhoz a járulékos áramköröket a TTL eszköztárból kellett a felhasználónak biztosítani. A mikroprocesszortechnika történetében a mérföldkövet 1973 decembere jelentette, amikor az Intel cég piacra dobta az MCS80 (Microcomputer System) mikroszámítógép rendszert, melynek központi egysége a 8080 típusú mikroprocesszor volt. Azért tekinthető ez mérföldkőnek, mert minőségi változást jelentett, ugyanis a rendszer kínálta áramkörökből mikroszámítógépet lehetett építeni. Kezdetben nagyon sok cég volt a piacon, mára már csak néhány cég maradt. A 70-es években fejlesztett mikroprocesszorok (Motorola 6800, Zilog Z80, Intel 8085 stb.) még **általános célú processzorok** voltak, mert úgy ipari, mint számítástechnikai célra alkalmazást nyertek. Az 1970-es évek második félévében a fejlesztés több irányt vett: egyrészt a 16 bites processzorok irányába (8086-1977), másrészt az egychipes mikroszámítógépek irányába (8048-1976), illetve a speciális mikroprocesszorok irányába (aritmetikai, be/ki processzor, stb.). A mikroprocesszorok fejlődését nagymértékben befolyásolták a személyi számítógépek igen rohamos elterjedése. Az első ilyen gép a Z-80 bázisú ZX81 még megjelenítőként TV vevőt, háttértárolóként kereskedelmi magnetofont használt. Forradalmi újdonságként hatott a Rockwell R6500 típusú processzorra épült COMMODORE gépcsalád, amely először használt háttértárolóként floppy diszket. A PC piacon a minőségi változást az IBM PC-k megjelenése jelentette, amely az Intel processzorok fejlődését is meghatározta a 80-as évek után. Ugyancsak a 80-as években az ipari alkalmazások terén a nagy zavarvédettségű **CMOS processzorok** terjedtek el, melyek napjainkra **mikrovezérlőkké** nőtték ki magukat, mivel már az A/D ill. D/A konvertereket is tartalmazzák. A processzorok hardvere mellett a szoftver fejlődése is döntően befolyásolta a fejlesztést. Az első 8 bites gépek **operációs rendszere** az 1-2 Kbájtos **monitor**

programok voltak, amelyeket a CP/M operációs rendszer követett. Ebből fejlődött ki a DOS operációs rendszer, amelynek feltétele a diszkes háttértárolás. Napjainkban a számítógépekben a különböző architektúrák (CISC, RISC) versenye tapasztalható. A mikroszámítógépek történelmi bemutatására a már említett MCS 80 rendszer áramköri szintű alapkonfigurációját láthatjuk a 7.1. ábrán. Megjegyezzük, hogy a perifériális áramkörök egy része ma is használatos. A fejlődés leginkább a CPU-t és a memóriákat érintette.



7.1. ábra

7.1.2. A digitális számítógépek általános felépítése

Említettük, hogy a mikroprocesszorok tulajdonképpen a számítógépek kicsinyített másai, ezért bevezetés szinten foglalkozunk a számítógépek felépítésével és tipikus műveleteivel. Egy Neumann-féle számítógép funkcionálisan 3 fő részből és a hozzákapcsolódó sínrendszerből áll.
Ezek:

- központi feldolgozó egység (Central Processor Unit = CPU)
- memóriák (RAM, ROM)
- beviteli/kiviteli egység (Input-Output Unit).

A Neumann gép tehát tartalmaz egy memória modult, amely egyrészt tárolja a végrehajtandó program utasításait, (ROM), valamint az utasítások által feldolgozandó adatokat (RAM). A **programtárolóban tárolt, logikailag összefüggő kódolt utasításokat programnak** nevezzük. A memória egység tehát két jól elkülönített részre osztható: **program memória** (PM) és **adatmemória.** Adatmemóriaként csak írható/olvasható memória (RAM)

szerepelhet, míg programmemóriaként ROM és RAM is szerepelhet. Utóbbi főként program tesztelés esetén. A központi feldolgozó egység a tárolt utasításokat egyenként sorra véve hajtja végre a vezérlő egység és az aritmetikai-logikai egység (ALU) révén. A CPU a vezérlő egység és az ALU működéséhez kisebb-nagyobb számban használ **speciális, vagy általános célú regisztereket** adatok tárolása céljából. A CPU két legfontosabb speciális célú regisztere az **utasításszámláló** regiszter (PC = Program Counter vagy IP = Instruction Pointer), amely a tárolt program soronkövetkező utasításának memóriabeli címét tárolja és az **utasítás regiszter** (IR = Instruction Register), amely a tárolóból előkeresett, végrehajtandó utasítást fogadja be a feldolgozás időtartamára. Megjegyezzük, hogy a korszerű processzorok esetén az IR funkciója erősen módosult. Az utasítás végrehajtás folyamata:

- **utasítás előkészítés, lehívás (fetching)** során a CPU a PC tartalma (amely a soronkövetkező utasítás memóriabeli címe) alapján lehívja a tárból a következő utasítás kódját és azt az IR-be helyezi
- a **programszámláló tartalmának növelése** az utasítás hosszának megfelelő értékkel való növelése abból a célból, hogy a PC a soronkövetkező utasítás címére mutasson
- a **műveleti kód (operation code)** értelmezése, az operandus helyének meghatározása: ez az IR-ben tárolt utasítás kódjának értelmezését jelenti, azaz arra a kérdésre válaszol, hogy mit kell a gépnek csinálnia, ill. az így megfejtett tevékenységhez szükséges operandusok helyének meghatározása az utasítás címre vonatkozó része alapján, vagy a végrehajtási fázis meghatározása (pl. vezérlés átadó utasítás esetén)
- a **művelethez szükséges adatok előkészítése**: a művelet elvégzéséhez az előző lépésben meghatározott cím alapján az operandus(ok) kikeresése a regiszterből, vagy memóriából és áthelyezése az utasítás által előírt helyre, amely leggyakrabban az ALU akkumulátor regiszter (AC), vagy más speciális regiszter
- **utasítás végrehajtás (executing):** ebben a fázisban történik az előző lépésekben előkészített operandusokkal az utasítás által előírt művelet végrehajtása
- az **eredmény elhelyezése** az előírt helyre (AC, regiszter, memória), majd az utasítás-feldolgozási ciklus újra kezdése.

A fentiek szerint részletezett utasítás-végrehajtás a gépi kódú utasításokra vonatkozik, amelyek a számítógép számára közvetlenül értelmezhetőek. A vezérlő egység a CPU-n belül az utasítás értelmezés (ld. előbb) után, az abban előírtaknak megfelelően vezérli a gép egészét, az áramköri egységeket. Ez a vezérlés történhet közvetlenül, **huzalozott logikával** ill. közvetett módon **mikroprogramozott logikával.** A mikroprogramozott vezérlés azt jelenti, hogy az utasítás műveleti kódja elindít egy elemi vezérlési lépéseket tartalmazó mikroprogramot és ennek révén vezérli a CPU vonatkozó egységeit. A számítógép **utasításainak hossza lehet rögzített fix érték** (mint pl. a RISC

processzorok esetén), ill. **változó hosszúságú** (pl. 1-6 bájt az Intel processzoroknál). Az utasítás szerkezete határozza meg, hogy a processzornak az utasítás mely részét hogyan kell értelmeznie. Értelmezés szempontjából az utasítások három fő részre oszthatók:

- **műveleti kód** (operation code, opcode), amely az elvégzendő művelet fajtáját definiálja
- **címrész** (address field), amely a művelet végrehajtásához szükséges operandusok memóriabeli helyére vonatkozó információt tartalmazza
- **módosító rész,** amely a műveleti kód értelmezésére adhat módosító előírást.

Az utasítás végrehajtásakor a processzornak általában négy tároló címre van szüksége: az első operandus címe, második operandus címe, az eredmény címe, a soronkövetkező utasítás címe.

Alapvető címzési módok:

- **közvetlen** (direkt, direct) címzés esetén az utasításban maga a tárolóhelycím (regiszter vagy memória) szerepel. A közvetlen címzési módon belül megkülönböztetünk **abszolút** ill. **relatív címzési** módot: Abszolút címzés esetén a tényleges címet tartalmazza az utasítás, míg relatív címzés esetén valamilyen **alapcímhez viszonyított címet** tartalmaz az utasítás és a konkrét címet az **alapcím és relatív cím összegeként** határozza meg a processzor
- **közvetett** (indirekt, indirect) címzés esetén az operandus az utasításban megcímzett tárolóhelyen megadott memória címen (tehát közvetetten megadott címen) van
- **közvetlen adatcímzés** (immediate) tulajdonképpen álcímzés, mivel az operandus magában az utasításban van megadva (tehát címzésről csak formailag van szó).

A legtipikusabb címmódosítási eljárás az **indexelés** (indexing), amely segítségével a megcímzett tárolóhely címértéke folyamatosan növelhető ill. csökkenthető, így pl. memória blokk kezelésére használható.

Tárhierarchia

Az adatok kellő időben és idő alatt való elérésének és elhelyezésének strukturális eszköze a **tárhierarchia.** A nagykapacitású tárolók elérési ideje általában nagyságrendekkel nagyobb, mint a processzor közvetlen közelében lévő tárolóhelyeké. A processzorhoz legközelebb a regiszterek vannak, amelyek viszonylag kevés (max. 100 bájt) adat tárolására alkalmasak, ugyanakkor elérési idejük a legkisebb (10-30 nsec). A program végrehajtásához közvetlenül szükséges programok és adatok tárolására a főtár (ROM, RAM) szolgál, melynek kapacitása néhány Mbájt és elérési ideje 70-100 nsec. Az éppen nem szükséges adatok tárolását a nagykapacitású háttértárolók (floppy, Winchester, stb) biztosítják több Gbájtos tárkapacitással és 10-30 msec-os elérési idővel. Az

adatok csak a tárolóhierarchián végighaladva kerülhetnek feldolgozásra a processzorban. Az adatáramlás folyamatosságának biztosítására, pufferelési céllal, az egyes tárolási szintek közé kisebb kapacitású, a felhasználó számára „láthatatlan" gyors működésű tárolók kerülnek, amelyeket **cache-táraknak** nevezik.

A tárolókezelés funkciói:

- a tárolóhierarchia leghatékonyabb működtetése
- a rendelkezésre álló memória szétosztása a feldolgozandó feladatok között
- a kellő védelem biztosítása az adatok biztonságos feldolgozásához.

A tárolókezelés feladatainak megoldására szolgál a processzorok tárolókezelő egysége (MMU = Memory Management Unit).
A számítógép a **beviteli, kiviteli** egység révén létesít kapcsolatot a külvilággal az adatok be/kivitele révén. A **sínrendszer** az adatok kétirányú mozgatására, valamint a címek és vezérlő jelek kiadására szolgál. A sínrendszert a CPU működteti.

7.1.3. Mikroszámítógép funkciói

Egy mikroszámítógép alapvetően három fő részből áll: CPU, memóriák, be/ki eszközök. Ezen részegységeket a sínrendszer köti össze.

a) A mikroprocesszor

A mikroprocesszor a számítógép funkcióit ellátó digitális VLSI áramkör, melynek 3 fő része van: időzítő-vezérlő egység, aritmetikai-logikai egység (ALU) és regiszterek. Az egyes részegységek funkciói az alábbiak szerint foglalhatók össze. Az **időzítő-vezérlő egység** feladata a program utasításai vagy külső kérések (megszakítás, tartás, várakozás) alapján a gép részeinek irányítása. Ez részben az ALU műveleteinek vezérlését, valamint az egyes adatútvonalak nyitását, zárását, a sínek működtetését, másrészt a külső egységek: a memória és az I/O egységek vezérlését jelenti. Az utasítások végrehajtása többnyire **mikroprogram** alapján történik. Minden utasítás műveleti kódja egy kis kapacitású ROM tárban, azaz a **mikroprogramtárban** elhelyezett programot indít el. A mikroprocesszor időbeni működését biztosító órajelet az **időzítő egység** fogadja. A vezérlő egység fontos része az **utasítás regiszter** (Instruction Register = IR), amely program memóriából a fetch ciklusban lehívott **utasítás kódját** tárolja, amíg az **utasítás dekódoló** és értelmező logika meghatározza az elvégzendő műveletet és elindítja a végrehajtást vezérlő mikroprogramot. A korszerű processzorokban az IR szerepét a **pipeline regiszter** váltotta fel. A vezérlő egység fontos funkciója a különböző aszinkron jellegű kérések (program megszakítás, tartás kérés, várakozás kérés) fogadása és az ezekhez tartozó vezérlési procedúra lefolytatása. A mikroprocesszor másik fontos egysége az **aritmetikai-logikai egység** (Arithmetic Logic Unit = ALU), amely az

utasításokban meghatározott aritmetikai és logikai műveleteket hajtja végre. Az ALU-hoz szorosan hozzátartozik az **akkumulátor regiszter** ill. a **flag regiszter** (ld. később). Fontos megjegyezni, hogy a processzorok általában csak néhány aritmetikai műveletet (összeadás, kivonás, szorzás) képesek elvégezni, ezért a korszerű mikroprocesszorokhoz ma már nélkülözhetetlenül hozzárendelnek egy **aritmetikai társprocesszort** (coprocessor) pl. I486, MC68040.

b) Regiszterek

A mikroprocesszorok **speciális** és **általános célú** regisztereket tartalmaznak. **Speciális célú regiszterek:** utasításszámláló regiszter, utasításregiszter, állapotregiszter, veremmutató. Ezek szinte valamennyi mikroprocesszorban megtalálhatók, de az egyes típusok további speciális célú regisztereket tartalmazhatnak, pl. idexregiszter, báziscímregiszter, stb. **Utasításszámláló regiszter** (PC = Program Counter vagy IP = Instruction Pointer), amely mindig a soron következő utasítás memóriabeli címét tartalmazza. A PC kezdő értékét, azaz a program első utasításának helyét az operációs rendszer jelöli ki. A mikroprocesszor **törlés** (reset) bemenetét hatásosan vezérelve a PC-be a 0000hex cím töltődik. A PC tartalma vagy minden memória hozzáférés után eggyel nő, vagy vezérlésátadó utasítás esetén (JUMP, CALL, RETURN stb.) a vezérlő egység a PC-be az új címet tölti be. Az **utasításregiszter** funkcióját a vezérlőegység leírásakor bemutattuk. **Állapotregiszter(ek)** (flag regiszter) ún. jelzőbiteket (feltétel biteket) ill. más vezérlő, ellenőrző biteket tartalmaznak. Korábban a flag bitek az ALU műveletekhez voltak hozzárendelve pl. **átvitel bit (carry),** az eredmény **nulla voltát jelző bit (zero), túlcsordulás bit** stb. Az újabb mikroprocesszorok esetén számos **vezérlési információt jelzőbitek** tárolnak pl. megszakítás kiszolgálásának letiltása, memória lapozás engedélyezése, stb. A **veremmutató** (stack pointer = SP) egy speciális regiszter, amely a veremtár legfelső elemének címét tartalmazza. A **veremtároló az adatmemória (RAM) egy lefoglalt területe.** Adatokat csak a verem tetejére lehet tenni és csak onnét lehet levenni. Ezt a memória kezelési módot **„utoljára be, elsőre ki"** (Last-In-First-Out = LIFO) kezelésnek nevezik. A stack pointer mindig arra a helyre mutat, ahová a következő adatot elhelyezi. Az SP minden stack betöltésénél a betöltött bájtok számával csökken (dekrementálódik) ill. kiolvasáskor inkrementálódik.

Adatbetöltéskor az SP először dekrementálódik és aztán következik be az adat beírása, kiolvasáskor először a processzor olvas, aztán az SP inkrementálódik. Ezért gyakran **predekrementáló** ill. **posztinkrementáló** jellegűnek tekintik a stack műveletet.

A CPU fontos részét képezi a **sínredszer.** Ezen egyrészt a belső egységek közötti adatforgalom, valamint a külső egységek (memória, I/O) közötti

adatforgalom bonyolódik. A sínrendszer funkcionálisan 3 féle sínt foglal magába: adatsín, címsín, vezérlő sín. A sínrendszer funkciói:

- meg kell oldania az adatforgalomban résztvevő eszközök kijelölését (**címsín** = address bus)
- meg kell határozni az adatátvitel irányát (adatsín vezérlés)
- biztosítani kell az adatok útját (**adatsín** = data bus)
- meg kell oldani a kapcsolatban résztvevő eszközök működésének összehangolását (**vezérlő sín** = control bus).

A háromféle sín jellemzése:

- **adatsín kétirányú, háromállapotú,** 8-32 bit szélességű (processzortól függően)
- **címsín: egyirányú, háromállapotú,** 16-32 bit szélességű (azaz ennyi vezeték)
- **vezérlősín: egyirányú, háromállapotú,** 5-15 bit szélességű (azaz ennyi vezeték). A legegyszerűbb vezérlősín 5 bites: MR, MW, I/OR, I/OW és interrupt. A vezérlősín révén lehet az azonos címen lévő memória illetve I/O műveleteket megkülönböztetni.

A külső sínrendszer lehet **helyi sín (local bus)** jellegű, amely a processzorhoz közvetlenül kapcsolódó részt jelenti, ill. lehet **rendszer sín (system bus),** amely esetén a processzor sínmeghajtásán keresztül kapcsolódik a rendszer elemeihez. A sínrendszer használatának előnye, hogy a szabványosított jel és vezetékkiosztás miatt az egyes részegységek könnyen cserélhetők. Ugyanakkor fel kell hívni a figyelmet arra, hogy a rendszerbemenetei hardver jelleggel terhelik a sínrendszert, ezért kell bizonyos számú modul esetén **sínmeghajtást** használni. Megjegyezzük, hogy a mikroprocesszoros rendszerekben használatos sín párhuzamos sínnek tekinthető, míg a számítógéphálózatok soros sínnek tekinthetők.

7.1.4. A mikroprocesszor tipikus műveletei

A következőkben azon részműveleteket írjuk le, amelyekből a számítógép működése „összerakható".

a) Időzítés

A **CPU működése ciklikus: utasítás lehívás, végrehajtás, lehívás, végrehajtás,** stb. Ezt a pontos sorrendiséget a **rendszeróra** vezérli. A CPU működésében a legelemibb időegység a **gépi állapot**, amely rendszerint egy órajel periódusa alatt játszódik le. Egy gépi állapothoz egy jól definiált művelet tartozik: pl. a címinformáció kijuttatása az címsínre stb. Általában **több gépi állapot alkot egy gépi ciklust.** A **gépi ciklus** összetettebb műveletet jelent, mint pl. a memória egy rekeszének kiolvasása (MR), ill. írása (MW), vagy I/O készülék írása, ill. olvasása (I/OW, I/OR), utasítás lehívás (fetching), stb. Egy utasítás lehívásának és végrehajtásának együttes

műveletét **utasításciklusnak** mondjuk. Az utasításciklus 1...8 gépi ciklusból állhat az utasítás bonyolultságától függően. Általában azt mondhatjuk, hogy egy utasításciklus annyi gépi ciklusból áll, ahányszor a CPU-nak a memóriához vagy I/O-hoz kell fordulnia. **Minden utasításciklus utasítás lehívási gépi ciklussal kezdődik.**

b) Utasítás lehívás (fetching)

Minden utasításciklus első gépi ciklusa egy utasítás lehívás a programmemóriából. Folyamata:

- a CPU kijuttatja az **utasításszámláló tartalmát a címsínre**
- a CPU az **adatsínt bemeneti állapotba** állítja
- a CPU kiad egy **MR vezérlőjelet a vezérlő sínre**
- a CPU az adatsínen megjelenő információt az **utasítás regiszterbe** (IR) tölti
- a PC tartalmát 1-gyel növeli.

Ha több bájtos utasításról van szó, akkor a többi bájt beolvasása hasonló művelettel történik, de az adatok nem az IR-be kerülnek. Megjegyezzük, hogy a mai gépeknél az utasítás lehívás ettől eltérő módon történik.

c) Memória olvasás (Memory Read:MR). Folyamata:

- a CPU kiadja a **címsínre a kiolvasandó memória címét**
- a CPU az **adatsínt bemeneti állapotba állítja**
- a CPU kiad egy **MR jelet a vezérlősínre**
- a CPU az adatsínen lévő információt valamelyik regiszterébe tölti.

d) Memória írás (Memory Write: MW)

A memória írás művelete hasonló a MR-hez a viszonyok értelemszerű cseréjével. Folyamata:

- a CPU kiadja a memória címet a címsínre
- a CPU az **adatsínt kimeneti állapotba állítja**
- a CPU kiadja a **beírandó adatokat az adatsínre**
- a CPU kiad egy **memória írás (MW) jelet a vezérlő sínre.**

f) Beviteli eszköz olvasása (Input: I/OR)

g) Kiviteli eszköz írása (Output: I/OW)

A beviteli eszköz írása és olvasása igen hasonlít a memória olvasás és írás műveletéhez, a folyamat a memória szavak I/O-ra cserélésével leírható. Látható, hogy az azonos című memória ill. I/O egység írása ill. olvasása a vezérlő sínen lévő eltérő vezérlő jel révén van megkülönböztetve és így kerülhető el a **sínkonfliktus**, pl. olvasás esetén.

h) Verem memória olvasása (Stack Read)

i) Verem memória írása (Stack Write)

E két művelet tulajdonképpen megegyezik a MR, MW műveletekkel azzal a különbséggel, hogy a **memória címét a veremmutató** (SP) adja és a vezérlő egységnek gondoskodni kell az **SP automatikus növeléséről ill. csökkentéséről**. Figyeljük meg, hogy a FETCH, MR, Stack Read ciklusok csak a mikroprocesszor belsejében lévő információ forrásokban és célhelyekben térnek el egymástól, kívülről a sínrendszeren csak a pontos memória cím ismeretében lehet megállapítani, hogy melyik művelet zajlik a sínen.

j) Várakozás (Wait)

Kiderült az előzőekből, hogy a CPU tevékenységét egy kvarc oszcillátor vezérli. Egy mikroszámítógép rendszert úgy terveznek, hogy a **CPU és a memóriák azonos sebességgel** (frekvencián) működjenek. Előfordul azonban, hogy ez nem áll fenn, pl. ha egy rendszerben a CPU-t gyorsabbra, vagy kényszerűségből a memória egy részét lassúbbra választják. Hangsúlyozni kell, hogy arról az esetről van szó, ha a memóriának csak egy **kisebb részének nagyobb a hozzáférési ideje**, mint a CPU gépi állapot ideje. Ugyanis, ha a teljes memória tartomány lassabban működik, akkor le kell csökkenteni a CPU oszcillátorának frekvenciáját. Térjünk vissza az előző esetekhez, amikoris előnytelen lenne a rendszer sebességét lecsökkenteni. E helyett az lenne az optimális, ha csak akkor működne lassabban a rendszer, ha lassúbb memóriával (vagy I/O-val) kommunikál. Erre dolgozták ki a **READY-WAIT** funkciót. Ennek lényege, hogy egy a címsínre csatlakozó logika (dekódoló) felismeri a címkombinációból, annak a memória blokknak a címét, amelyik egy gépi állapotnál több időt igényel és kimenete **várakozást kér** a processzor megfelelő (pl. Ready) bemenetén. Ezen kérést elfogadva a CPU ún. **„wait"** állapotba kerül, amit a megfelelő kimeneten (wait) jelzi. A wait állapot azt jelenti, hogy a **CPU a címsínen és az adatsínen hagyja az előző információt egy (újabb kérés esetén több) gépi állapot idejére.** Ha pl. a CPU gépi állapot ideje 500 ns és a memória hozzáférése 650 ns, akkor egy ready kérés 2x500 ns hozzáférési időt biztosít a memóriának, ami már bőségesen elegendő. A címdekódoló logikára pedig azért van szükség, hogy ez a **várakozás csak a lassú memória esetén lépjen fel.** Megjegyezzük, hogy a wait állapotot az ún. **egylépésű üzemmód** kialakítására is felhasználhatjuk. Ez a hibakeresésnél előnyös. A címdekódoló logika egy monostabil multivibrátoron keresztül működteti a ready vonalat a végleges leállás megakadályozására. A mikroprocesszor-technikában a várakozás miatt kiesett gépi állapotokat **elengedett ütemeknek** is szokás nevezni.

k) Programmegszakítás (Interrupt)

A számítógépes feldolgozás közben igen gyakran következnek be olyan események, amelyek a feldolgozás szempontjából **váratlannak** tekintendők,

pl. egy irányítórendszer esetén valamely érzékelő hibát jelez (tűz érzékelő, füst érzékelő, feszültség kimaradás stb.). Ezek közel **azonnali feldolgozást** igényelnek. Az ilyen események feldolgozására szolgál a mikroszámítógép **megszakítás rendszere (interrupt system).** Ez lehetőséget biztosít arra, hogy egy hardver eszköz (pl. érzékelő) **programmegszakítást** kérjen. A megszakítás kérelem egy jelzés a processzor számára valamely fontos esemény bekövetkeztéről, amely valamilyen **kiszolgálóprogram** révén reagál ezen aszinkron jellegű eseményre. A megszakítás kérést a processzor egy belső procedúra szerint fogadja el, amit a megfelelő kimeneten (INTA) jelez vissza. Hangsúlyozni kell, hogy a megszakítási funkció ellátására egy **hardver-szoftver** együttes szolgál, amely együttesen végzi el a kérelem kiértékelését és annak végeredményeként a szükséges tevékenységet. A megszakításokat **mikroprocesszoron belüli események** is kiválthatják (pl. 0-val való osztás esetén). A **megszakítás kéréseket a processzor szubrutin jelleggel** hatja végre. A konkrét lebonyolítás részletezésére a későbbiekben visszatérünk. A **megszakítás kérést szoftverből** is indíthatjuk. Több megszakítás kérés esetén a **megszakításokat prioritási sorrendjük** alapján dolgozza fel a processzor. A hardver megszakítások rendszerint **maszkolhatók,** ami azt jelenti, hogy pl. 8 megszakítási szint közül az 5-ös szintűt engedélyezzük vagy tiltjuk, nem pedig mind a 8-at egyszerre. Ugyanakkor van **nem maszkolható megszakításkérés** (NMI = Non Maskable Interrupt) is. Ez azt jelenti, hogy **az NMI szoftver úton nem tiltható le.** Az NMI funkciót rendszerint valamilyen súlyos hardver hibához (pl. feszültségkimaradás) rendelik. Hangsúlyozni kívánjuk, hogy a **mikroszámítógép hatékony megszakításrendszere feltétele a mérés ill. irányítástechnikában, távközléstechnikában elengedhetetlenül fontos valós idejű (real-time) adatfeldolgozásnak.**

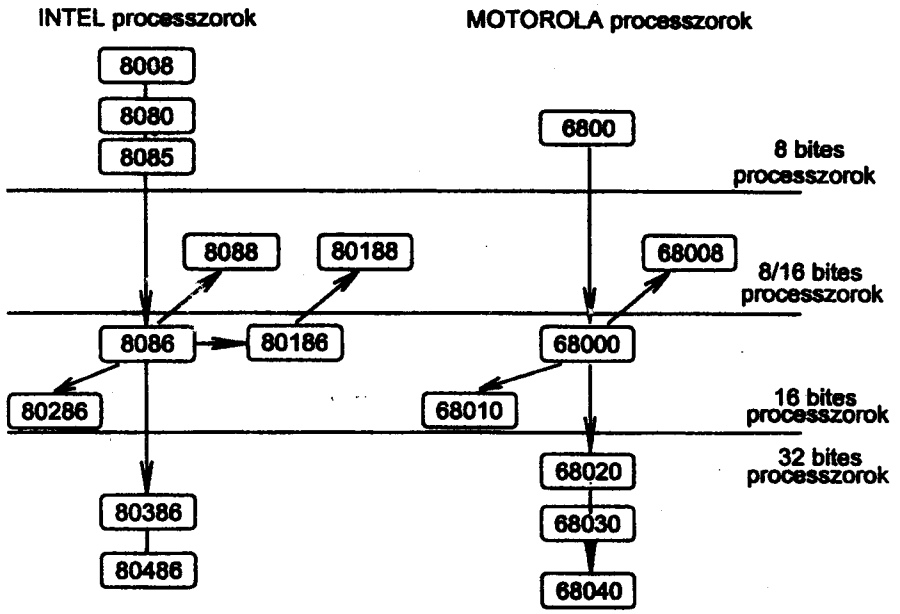
l) Tartás (HOLD)

A mikroszámítógépen belül a memória és be/ki eszköz közötti adatforgalom csak két lépésben történhet:.memória - CPU - CPU - be/ki eszköz és viszont. Ez nagy mennyiségű adat esetén hátrányosan növeli (duplázza) az adattranszfer idejét. Ezen a problémán segít a processzor **HOLD** állapota. A CPU rendelkezik egy **tartáskérés bemenettel** (HOLD vagy BUSRQ). Amennyiben ezen a bemeneten tartáskérés jelenik meg, azt a processzor egy belső procedúra után a megfelelő **kimeneten nyugtázza** (HLDA vagy BUSACK), ami egyben azt jelenti, hogy a **sínrendszerét nagyimpedenciás (Z) állapotba helyezi.** Ezután egy külső eszköz rákapcsolódhat a sínrendszerre és közvetlen adatforgalmat bonyolíthat a memória és a be/ki egységek között. Az ilyen memória hozzáférést **direkt memória kezelésnek** (DMA = Direct Memory Access) nevezik. Azt az eszközt, amely az adatforgalom vezérlését végzi, **DMA vezérlőnek** nevezik (DMA controller). DMA adatátvitelt alkalmaznak pl. floppy diszk kezelésnél, vagy többprocesszoros rendszerekben a közös memória kezelésnél stb. Felhívjuk a

figyelmet, hogy a HOLD állapot szinte kizárólagosan hardver procedúra, lekezelése nem jár a CPU belső állapotának megváltozásával, mint az a programmegszakításnál történik. HOLD állapotban a processzor lehetőséget biztosít a programmegszakítás lebonyolítására.

7.2. Nyolc bites mikroprocesszorok

Az Intel és a Motorola processzorok fejlődését szemlélteti a 7.2. ábra.

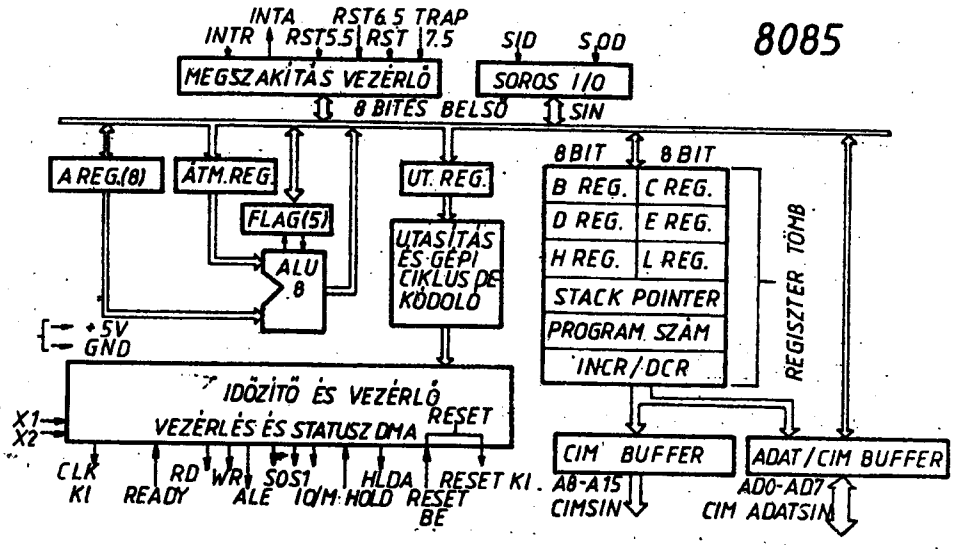


7.2. ábra

A 8 bites mikroprocesszorok működésének bemutatását **didaktikai** okból tartjuk fontosnak. Egy 8 bites mikroprocesszor még nem tartalmaz annyi összevont funkciót, így megértése a kezdők számára elősegítheti a komplexebb processzorok működésének megértését. Ezért választottuk a 8085-ös processzor bemutatását.

7.2.1. Az i8085 CPU hardver felépítése

A 8085 típusú CPU blokkvázlata a 7.3. ábrán látható. A 8085 sínrendszere ún. **multiplex sín,** ami azt jelenti, hogy a 16 bites címsín alsó felét az AD0...AD7, felső felét az A8...A15 vezetékeken adja ki. Az AD0...AD7 sínbitek címjellegét a μP az ALE kimenetén jelzi. Ezután a CPU a cím alsó bájtját visszaveszi az AD0...AD7 vezetékekről, s a továbbiakban a cím alsó bájtját egy latch-ban tárolni kell. Ezután az AD0...AD7 adatsínként funkcionál.



7.3. ábra

A CPU belső áramköreit 3 csoportba oszthatjuk:

- regiszter blokk
- aritmetikai blokk
- utasítás végrehajtással és vezérléssel kapcsolatos egységek.

7.2.1.1. Regiszterblokk

A regiszterblokkot 6 db. **általános célú regiszter, stack pointer** (SP) **utasításszámláló** alkotja. A 6 db. regiszter (B, C, D, E, H, L) egyenként 8 bit tárolására alkalmas. A regisztereknek kétféle üzemmódja van:

- 6 db 8 bites regiszter
- 3 db 16 bites regiszterpár (BC, DE, HL).

A regiszterek ill. regiszterpárok tartalmát a program határozza meg. A regiszterek a tápfeszültség bekapcsolásakor **tetszőleges értéket** felvehetnek, ezért azokat a **program elején be kell állítani (inicializálni).** Az utasításszámláló 16 bites, amely mindig az éppen végrehajtás alatt álló utasítás címét tartalmazza. Az utasításszámlálót RESET-tel lehet 0000H címre állítani. A veremmutató (SP) 16 bites. A stack kezelés két bájtos, mivel a 16 bites memória címet 2 memória bájtban lehet tárolni.

7.2.1.2. ALU-blokk

Az ALU blokk részei:

- 8 bites akkumulátor
- 8 bites ALU
- flag regiszter
- decimális korrekciós hálózat.

Az **aritmetikai blokk** elemeinek feladata a programban előírt aritmetikai, logikai műveletek végrehajtása. Az **akkumulátor** tulajdonképpen egy speciális funkciójú 8 bites regiszter, amely **forrás** és **eredményregiszterként** egyaránt tekinthető. Az aritmetikai és logikai műveletek egyik operandusa szinte mindig az akkumulátorban van és az eredmény ugyancsak az akkumulátorban képződik (legtöbbször). Az ALU egy **speciális kombinációs hálózat**, amely az aritmetikai ill. logikai műveletek realizálására alkalmas. A **flag regiszter** a flag (jelző) biteket tárolja, a 8085 esetén tulajdonképpen 5 db RS FF. A **flag bitektől függően lehet feltételes** utasításokat (pl. JUMP IF...) kezdeményezni, ezért gyakran feltétel biteknek is nevezik. A 8085 CPU flag bitjei: Z, S, P, C, AC.

Z flag (zérus): ha az ALU által elvégzett művelet eredménye nulla, akkor Z = 1, egyébként Z = 0.

S flag (Sign = előjel): értéke az **akkumulátor legmagasabb bitjének** (A7) értékével egyezik meg. Negatív eredmény esetén S = 1, pozitív és zérus eredmény esetén S = 0.

P flag (parity = paritás): az adatátviteli műveletek ellenőrzésére szolgáló bit, melynek értéke P = 1, ha az eredményben páros számú 1-es értékű bit van, egyébként P = 0.

C flag (carry = átvitel): bit az aritmetikai műveletek esetén a **legmagasabb helyiértéken keletkező átvitelt** jelzi. Amennyiben keletkezik átvitel a legmagasabb biten, akkor C = 1, egyébként 0. Megjegyezzük, hogy a C regisztertől való megkülönböztetés céljából a carry flag-et gyakran CY-nal jelölik.

AC flag (auxiliary carry = segéd átvitel): tulajdonképpen a **decimális átvitelt** mutatja. Ennek akkor van jelentősége, ha nem bináris, hanem decimális számot ábrázolunk BCD kódban. Az AC flag-et a **decimális korrigáló** utasítás (DAA) használja.

7.2.1.3. A 8085 időzítő-vezérlő egység

Az **időzítő-vezérlő egység** feladata az **utasítások kódjának tárolása** (IR), **dekódolása (ID)** és az utasítások végrehajtásához szükséges jelek előállítása. **Fontosabb kivezetések funkciói:**

RESET IN - **törlés bemenet**, melynek aktiválásakor a PC tartalma 0000H lesz
RESET OUT - más egységek alapállapotba hozásához használatos kimenet
READY - **várakozás** kérés bemenet
HOLD - **tartás kérés bemenet**
HLDA - a **HOLD kérés elfogadását** nyugtázó kimenet.

A CPU HOLD bemenetére adott 1-es jel hatására a CPU a belső időzítési viszonyaitól függő idő elteltével **sínrendszerét nagyimpedenciás** állapotba helyezi (HLDA = 1). A HOLD állapot mindaddig fennmarad, amíg a HOLD bemeneten a kérő jel („1") fennáll. A HOLD kérés elfogadását a CPU-ban

szoftver úton letiltani nem lehet. A HOLD kérés megszűnte után a HLDA jel 0-ra vált.

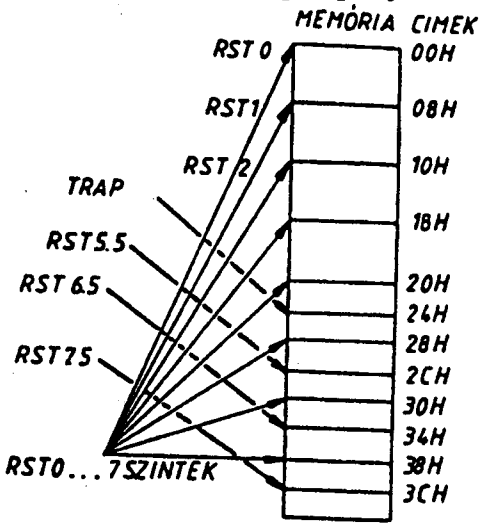
7.2.1.4. A 8085 megszakítás rendszere

INTR bemenet, amelyre rendszerint egy **megszakításkérő egység kimenete** csatlakozik (pl. 8279).

RST 5.5
RST 6.5 | **Maszkolható megszakításkérő bemenetek,** amelyeken a kérés elfogadásáig fenn kell maradnia a kérő jelnek.

RST 7.5 bemeneten a jel 0 → 1 átmenete (éle) okoz megszakításkérést, amely a jel megszűnte után tárolásra kerül.

TRAP bemeneten a jel 0 → 1 átmenete és az utána folyamatosan fennálló „1" jel eredményez megszakításkérést.
A TRAP megszakítást a CPU-n belül **nem lehet letiltani**. Az RST jelű megszakításkérések szoftver úton **egyenként engedélyezhetők ill. letilthatók.** Az INTR bemeneten keletkező megszakítás az EI utasítással engedélyezhető és a DI utasítással letiltható, de egyenként nem maszkolható. A 8085 megszakítási térképe, az egyes megszakítás kérések belépési pontjai a 7.4. ábrán láthatók.



7.4. ábra

A 8085 μP megszakításkéréseit a 7.1. táblázatban foglaltuk össze.

7.1. táblázat

Bemenet	Prio-ritás	Megszakítási szubrutin	Megszakítási feltétel	Maszkolás
TRAP	1	24H	0-1 átmenet és folyamatos 1 szint a mintavételezésig	nem maszkolható

RST7.5	2	3CH	Latchelt 0-1 átmenet	EI/DI, egyedileg: SIM
RST6.5	3	34H	Mintavételezésig fennálló 1 szint	EI/DI, egyedileg: SIM
RST5.5	4	2CH	Mintavételezésig fennálló 1 szint	EI/DI, egyedileg: SIM
INTR	5	CPU-n kívülről	Mintavételezésig fennálló 1 szint	EI/DI

A 8085 CPU tartalmaz egy **soros kommunikációs egységet**, melynek bemenete SID, kimenete a SOD. A **SID** (Serial Input Data) bemenet beolvasását a **RIM**, a **SOD** (Serial Output Data) kimenet beállítását a **SIM** utasításokkal végezhetjük.

7.2.1.5. A 8085 folyamatábrája

A 8085 folyamatábrája a 7.5. ábra szerinti.

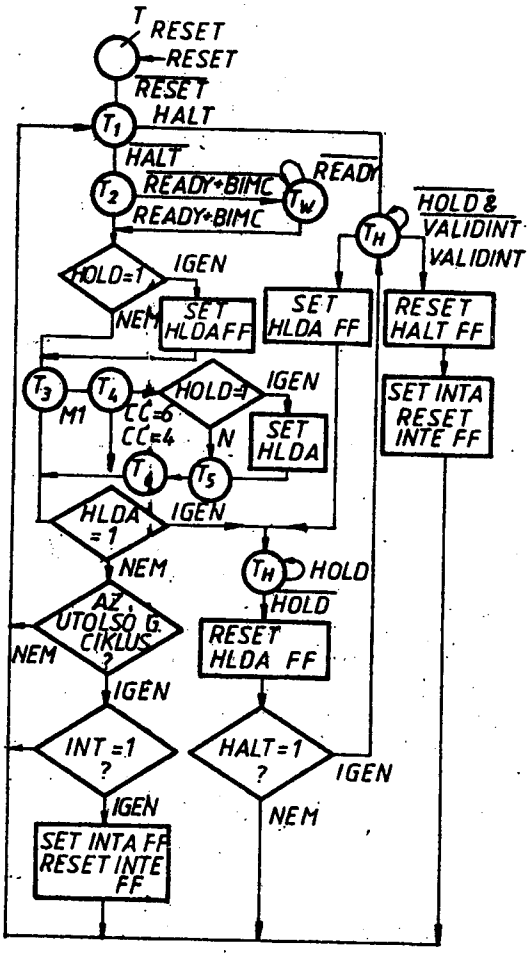
A folyamatábra egy **gépi ciklust** szemléltet. A T_i állapotok egy-egy **gépi állapotot** reprezentálnak. Egy gépi ciklus maximum 6 gépi állapotból állhat. A 8085 gépi ciklusai: utasítás lehívás, memória olvasás, I/O olvasás, I/O írás, megszakítás.

A 8085 CPU-nak 4 lehetséges állapota van: RUN, WAIT, HALT, HOLD.

RUN állapotban a CPU az órajelek által előírt sebességgel utasításokat hajt végre.
WAIT várakozási állapot, akkor lép fel, ha a gépi ciklus meghatározott időpontjáig nem érkezik meg a READY jel. Megszűnik, ha a READY = 1. Bekövetkezését szoftver úton letiltani nem lehet.
HALT állapotban a CPU nem hajt végre műveleteket. Ebbe az állapotba a CPU a HLT utasítás után kerül. A HALT állapot elfogadását letiltani nem lehet.

A HALT állapot megszüntetésére három lehetőség van:

- **külső HOLD igény** hatására a HALT állapot felfüggesztődik. A HOLD kérés megszűnte után a processzor ismét HALT állapotba megy vissza.
- A **CPU törlésekor** $\left(\overline{\text{RESET IN}} = 0\right)$ a CPU a 0000hex címről újraindul.
- **Engedélyezett megszakításkérés** elfogadásakor a CPU kilép a HALT állapotból. **HOLD** kérést a 8085 CPU minden **gépi ciklus** végén **elfogadhatja, míg INT kérést csak befejezett utasításciklus után** fogad el.



7.5. ábra

7.2.2. A 8085 μP címzési módok

7.2.2.1. Címzési módok

Az I8085-ös mikroprocesszor felépítése és utasításkészlete az adatok címzésének (vezérlésátadás) 2 módját teszi lehetővé.

a) Adatcímzések

- **Közvetlen adatçímzés**
  Közvetlen adatcímzés 2 és 3 bájtos utasításoknál fordul elő. Három bájtos utasítások esetén a 2. és 3. bájt memória címet, két bájtos utasításoknál a 2. bájt I/O címet jelent.

- **Regiszter címzés**
  Regiszter címzés esetén az utasításban, kódolt formában, közvetlenül ki van jelölve az a regiszter vagy regiszterpár, amelynek tartalma a műveletekben operandusként vesz részt.
- **Indirekt regisztercímzés**
  Indirekt regisztercímzés esetén az utasításban kijelölt regiszter pár tartalma a művelet operandusának címe.
- **Közvetlen adat**
  Közvetlen adat a két ill. három bájtos utasításoknál fordul elő. Ekkor a kétbájtos utasítás második bájtja a 8 bites, a három bájtos utasítás 2. és 3. bájtja a 16 bites operandust tartalmazza.

b) **Utasításcímzés**

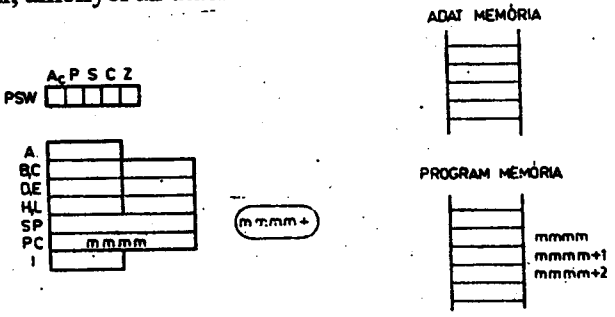
- **Közvetlen utasításcímzés**
  A programvégrehajtás menetét befolyásoló utasítások közvetlen utasításcímzések esetén kódolt (RST), vagy kódolatlan formában tartalmazzák annak az utasításnak a címét, amelyen a programvégrehajtást, esetleg valamely feltétel fennállása esetén folytatni kell. Az RST utasítást kivéve 3 bájtos utasítások tartoznak ebbe a csoportba, amelyek 2. és 3. bájtja az **ugrási címet** tartalmazza.
- **Indirekt, regiszterpáron keresztül utasításcímzés**
  Az utasításban kijelölt regiszterpár tartalma azonos az ugrási címmel.

7.2.2.2. Utasításkészlet

Az utasítások a műveletek típusától függően az alábbi csoportokba sorolhatók:

- adatmozgató utasítások
- aritmetikai utasítások
- logikai utasítások
- vezérlésátadó utasítások
- stack, I/O és a gépi vezérlés utasításai.

Az utasítások által végzett műveletek legkönnyebben az ún. **végrehajtási modell** alapján értelmezhetők. A 8085 processzor végrehajtási modelljét a 7.6. ábra szemlélteti, amellyel az utasítások 90 %-a értelmezhető.



7.6. ábra

Az utasítások leírásánál használt jelölések:

A	akkumulátor
data8	8 bites adat (bájt)
data16	16 bites adat (2 bájt)
1. bájt	utasítás első bájtja (műveleti kód)
2. bájt	utasítás második bájtja
3. bájt	utasítás harmadik bájtja
port	I/O port 8 bites címe
r, r1, r2	az A, B, C, D, E, H, L regiszterek egyike
DDD	célhely regiszter kijelölése a bitkombinációban
SSS	forráshely regiszter kijelölése a bitkombinációban
rp, RP	regiszterpár
rh	regiszterpár felső bájtja
rl	regiszterpár alsó bájtja
PC	a 16 bites utasításszámláló
PCH	a PC felső bájtja
PCL	a PC alsó bájtja
SP	16 bites veremmutató
gc, gá	gépi ciklus, gépi állapot
Z, S, P, AC, CY	flag bitek
( )	valamely regiszter vagy tároló tartalma

Regiszterek kijelölése az utasításkódban

DDD vagy SSS	regiszter
000	B
001	C
010	D
011	E
100	M
101	L
110	Memória
111	A

Regiszterpár kijelölése

RP	regiszterpár
00	BC
01	DE
10	HL
11	SP

7.2.2.2.1. Adatmozgató utasítások

Az adatmozgató utasítások közös jellemzői:

- lehetséges mozgatások: **regiszter - regiszter, regiszter - memória, memória - regiszter**
- az XCHG utasítást kivéve a **célhely utasítás előtti tartalma elvész, a** forráshely tartalma megmarad
- egyetlen adatmozgató utasítás sem befolyásolja a feltétel (jelző) biteket.

MOV r_1, r_2 (move regiszter)

Művelet: (r_1) ← (r_2), azaz az r_2 regiszter tartalmát átviszi az r_1 regiszterbe.
Utasításkód: 01DDDSSS. Egy gépi ciklus, 4 gépi állapot, regiszter címzés. Pl.: MOV A,C.

MOV r, M (move from memory)

Művelet: (r) ← ((H) (L)), azaz annak a memória rekesznek a tartalmát amelyiknek a címét a HL regiszterpár tartalmazza, betölti az r regiszterbe.
Utasításkód: 01DDD110. Két gc, 7 gá, regiszterpáron keresztül indirekt címzés. Pl.: MOV D, M.

MOV M, r (move to memory)

Művelet: ((H) (L)) ← (r), azaz az r regiszter tartalmát betölti a HL regiszterpár által megcímzett memória rekeszbe.
Utasításkód: 01110SSS. Két gc, 7 gá, regiszterpáron keresztüli indirekt címzés. Pl.: MOV M,H.

MVI r, data (move to regiszter immediate)

Művelet: (r) ← data 8, azaz az utasítás 2. bájtjában megadott adatot (d8) betölti az r regiszterbe. Az utasítás a regiszterek inicializálására szolgál.
Utasításkód: 00DDD110, d8. Két gc, 7 gá, közvetlen adatcímzés. Pl. MVI A, 3BH

MVI M data 8 (move to memory immediate)

Művelet: ((H) (L)) ← d8, azaz az utasítás 2. bájtjában megadott adatot betölti a HL által megcímzett memória rekeszbe. RAM betöltése a ROM-ból.
Utasításkód: 00110110, d8. Három gc, 10 gá, közvetlen adat és regiszterpáron keresztüli indirekt címzés. Pl. MVI M, 9EH.

LXI rp, data 16 (load regiszter pair immediate)

Művelet: (rh) ← 3. bájt

(rh) ← 2. bájt, azaz az utasítás 3. bájtjában megadott adatot a kijelölt regiszterpár magasabb, a 2. bájtjában megadott adatot a kijelölt regiszterpár alacsonyabb helyiértékű részébe tölti be.
Utasításkód: 00RP0001, d8l, d8h. Három gc, 10 gá, közvetlen regisztercímzés. Pl: LXI SP, 3FFFH, LXI B, 468AH, LXI D, 8A8BH, LXI H, 2222H.

LDA addr 16 (load accumulator direct)

Művelet: A ← ((3 bájt) (2 bájt)), azaz az utasítás 2. és 3. bájtjában egy memória rekesz van megadva, melynek tartalmát az utasítás betölti az akkumulátorba.
Utasításkód: 3AH, a8l, a8h. Négy gc, 13 gá, közvetlen adatcímzés. Pl.: LDA 4B3CH.

STA addr 16 (Store accumulator direct)
Művelet: ((3. bájt) (2. bájt)) ← (A), azaz utasítás 2. és 3. bájtjában megcímzett memóriarekeszbe betölti az akkumulátor tartalmát.
Utasításkód: 32H, a8l, a8h. Négy gc, 13 gá, közvetlen adatcímzés. Pl.: STA 2009H.

LDAX rp (load accumulator indirect)
Művelet: (A) ← ((rp)), azaz az utasításban kijelölt regiszterpár (BC vagy DE) 16 bites tartalma egy memória bájt címe. Ennek a memória bájtnak a tartalmát az utasítás betölti az akkumulátorba.
Utasításkód: 000rp1010. Két gc, 7 gá, regiszterpáron keresztüli indirekt címzés. Pl.: LDAX B, LDAX C.

STAX rp
Művelet: ((rp)) ← (A) (lásd LDAX rp-t, ellentétes irányú adatmozgatással).
Utasításkód: 000rp0010. Pl.: STAX C.

LHLD addr 16 (load H and L direct)
Művelet: (L) ← ((3. bájt) (2. bájt))
(H) ← ((3. bájt) (2. bájt + 1))
Az utasítás 2. és 3. bájtjában egy memória rekesz címe van megadva. Ennek a memória bájtnak tartalma az L, a következő eggyel magasabb címen lévő memória bájt tartalma a H regiszterbe kerül.
Utasítás kód: 1AH, a8l, a8h. Öt gc, 16 gá, közvetlen adatcímzés. Pl.: LHLD 88B7H.

SHLD addr 16 (Store H and L direct)
Művelet: ((3. bájt) (2. bájt)) ← (L)
((3. bájt) (2. bájt)) ← (H).
Lásd LHLD utasítást ellentétes adatátvitellel.
Utasításkód: 22H, a8l, a8h. Pl.: SHLD 93BBH.

XCHG (exchange H and L with D and E)
Művelet: (H) ⟷ (D)
(L) ⟷ (E).
Az utasítás kicseréli a H és L regiszterek tartalmát a D és E regiszterek tartalmával.

7.2.2.2.2. Aritmetikai utasítások
Az utasítások jellemzői:

- **valamennyi utasítás módosítja a jelző biteket**
- az összeadási és a **kivonási műveleteket kettes komplemensű aritmetika** szabályai szerint végzik
- a CY flag „1" értékű lesz, ha van maradék (borrow) a legmagasabb bitről és „0" lesz, ha nincs.

a) Egybájtos összeadási műveletek

ADD r (add register)
Művelet: (A új) ← (A régi) + (r), azaz az akkumulátor és az r regiszter tartalmának összegét az utasítás elhelyezi az akkumulátorban.
Utasításkód: 10000SSS. Egy gc, 4 gá, regiszter címzés. Módosított flag-ek: S, Z, P, CY, AC. Pl.: ADD C.

ADD M (add memory)

Művelet: (Aúj) ← (A régi) + ((H)(L)).
Utasításkód: 86H. Két gc, 7 gá. Módosított flag-ek: Z, S, P, CY, AC.

ADI data8 (add immediate)
Művelet: (A új) ← (A régi) + (d8).
Utasításkód: C6H, d8. Két gc, 7 gá. Módosított flag-ek: Z, S, P, CY, AC. Pl.: AD 7AH.

b) Több-bájtos összeadási műveletek

ADC r (add register with carry)
Művelet: (Aúj) ← (Arégi) + (r) + (CY).
Utasításkód: 10001SSS. Egy gc, 4 gá, regiszter címzés. Módosított flag-ek: Z, S, P, CY, AC. Pl.: ADC B.

ADC M (add memory with carry)
Művelet: (A új) (A régi) + ((H) (L)) + (CY).
Utasításkód: 8EH. Két gc, 7 gá, regiszteren keresztüli indirekt címzés. Módosított flag-ek: Z, S, P, CY, AC.

ACI data8 (add immediate with carry)
Művelet: (Aúj) ← (Arégi) + data8 + (CY).
Utasításkód: CEH, d8H. Két gc, 7 gá, immediate címzés. Módosított flag-ek: Z, S, P, CY, AC. Pl.: ACI 39H.

c) Egybájtos kivonási műveletek

SUB r (substract register)
Művelet: (Aúj) ← (A régi) - (r).
Utasításkód: 10010SSS. Egy gc, 4 gá, regiszter címzés. Módosított flag-ek: Z, S, P, CY, AC. Pl.: SUB H.

SUB M (substract memory)
Művelet: (Aúj) ← (A régi) - ((H) (L)).
Utasításkód: 96 H. Két gc, 7 gá. Módosított flag-ek: Z, S, P, CY, AC.

SUI data8 (subtract immediate)
Művelet: (Aúj) ← (A régi) - d8.
Utasításkód: D6H, d8. Két gc, 7 gá. Módosított flag-ek: Z, S, P, CY, AC.
Pl.: SUI 7CH.

d) Több-bájtos kivonási műveletek

SBB r (Substract regiszter with borrow)
Művelet: (Aúj) ← (A régi) - (r) - (CY).
Utasításkód: 10011SSS. Egy gc, 4 gá, regiszter címzés. Módosított flag-ek: Z, S, P, CY, AC. Pl.: SBB C.

SBB M (subtract memory with borrow)
Művelet: (Aúj) ← (A régi) - ((H) (L)) - (CY).
Utasításkód: 9EH. Két gc, 7 gá. Módosított flag-ek: Z, S, P, CY, AC.

SBI data8 (subtract immediate with borrow)
Művelet: (Aúj) ← (A régi) - data8 - (CY).
Utasításkód: DEH, data8. Két gc, 7 gá. Módosított flag-ek: Z, S, P, CY, AC.

e) Inkrementáló, dekrementáló utasítások

Az inkrementálás egy regiszter vagy memória tartalom **1-gyel való növelését** jelenti (előre számlálás).

INR r (increment regiszter)
Művelet: (r) ← (r) + 1.
Utasításkód: 00DDD100. Egy gc, 4 gá, regiszter címzés. Módosított flag-ek: Z, S, P, AC.

INR M (increment memory)
Művelet: ((H) (L)) ← ((H) (L) + 1).
Utasításkód: 34H. Három gc, 10 gá. Módosított flag-ek: Z, S, P, AC.

A dekrementálás egy regiszter vagy memória tartalom **1-gyel való csökkentését** jelenti.

DCR r (decrement register)
Művelet: (r) ← (r) - 1.
Utasításkód: 00DDD101. Egy gc, 4 gá. Módosított flag-ek: Z, S, P, AC.

DCR M (decrement memory)
Művelet: ((H) (L)) ← ((H) (L)) - 1.
Utasításkód: 35H. Három gc, 10 gá. Módosított flag-ek: Z, S, P, AC.

INX rp (increment register pair)
Művelet: (rp) ← (rp) + 1.
Utasításkód: 00RP0011. Egy gc, 6 gá. Az utasítás a flag-eket nem módosítja!
Pl.: INX H, INX D, INX B, INX SP.

DCX rp (decrement register pair)
Művelet: (rp) ← (rp) - 1.
Utasításkód: 00RP1011. Egy gc, 6 gá. Az utasítás a flag-eket nem módosítja. Pl.: DCX H, DCX B, DCX D, DCX SP.

DAP rp (add register pair to H and L)
Művelet: (H) (L) ← (H) (L) + (rp).
Utasításkód: 00RP1001. Három gc, 10 gá. Módosított flag: CY ← 1. Ez az egyetlen aritmetikai utasítás, amelyben az akkumulátor szerepét a HL regiszter pár tölti be. A DAD utasítást a PCHL utasítással kombinálva relatív címzést eszközölhetünk. Pl.: DAD B, DAD D, DAD H, DAD SP.

DAA (decimal adjust accumulator)
Művelet: az utasítás **két BCD kódú forrásadaton** végzett, az akkumulátorban képződő **eredmény BCD kódú korrekcióját** végzi a következő szabályok szerint:

- ha az akkumulátor alsó 4 bitjén tárolt szám nagyobb mint 9 vagy az AC = 1, akkor 6-ot hozzáad (10-et levon) az akkumulátor ezen 4 bitjéhez
- ha az akkumulátor 4 felső bitjén tárolt szám nagyobb, mint 9 vagy a CY = 1, akkor 6-ot hozzáad (10-et levon) az akkumulátor ezen 4 bitjéhez.

Utasításkód: 27H. Egy gc, 4 gá. Módosított flag-ek: Z, S, P, CY, AC.
Példa:
```
MVI A, 49D
MVI B, 57D
ADD B
DAA.
```

7.2.2.2.3. Logikai utasítások

Az utasítások ezen csoportja **Boole műveleteket** végez az adatokon. A Boole műveleteket az **azonos bitpozíció bitjein** végzi. A logikai utasítások a flag-biteket módosítják, a CY = 0, AC = 0 lesz.

ANA r (AND register)
Művelet: (A új) ← (A régi) & (r).
Utasításkód: 10100SSS. Egy gc, 4 gá. Módosított flag-ek: Z, S, P, CY, AC. Pl.: ANA C, ANA H.

ANA M (AND memory)
Művelet: (A új) ← (A régi) & ((H) (L)).
Utasításkód: A6H. Két gc, 7 gá. Módosított flag-ek: Z, S, P, CY, AC.

ANI data 8 (AND immediate)
Művelet: (A új) ← (A régi) & d8.
Utasításkód: E6H, d8.Két gc, 7 gá. Módosított flag-ek: Z, S, P, CY, AC. Pl.: ANI 3BH.

ORA r (OR regiszter)
Művelet: (A új) ← (A régi) V (r).
Utasításkód: 10110SSS. Egy gc, 4 gá. Módosított flag-ek: Z, S, P, CY, AC.
Pl.: ORA D.

ORA M (OR memory)
Művelet: (A új) ← (A régi) V ((H) (L)).
Utasításkód: B6H. Két gc, 7 gá. Módosított flag-ek: Z, S, P, AC, CY.

ORI data 8 (OR immediate)
Művelet: (A új) ← (A régi) V data 8.
Utasításkód: F6H, d8. Két gc, 7 gá. Módosított flag-ek: Z, S, P, CY, AC.
Pl.: ORI 7CH.

XRA r (exclusive OR register)
Művelet : (A új) ← (A régi) ∀ (r).
Utasításkód: 10101SSS. Egy gc, 4 gá. Módosított flag-ek: Z, S, P, CY, AC.
Pl.: XRA A.

XRA M (exclusive OR memory)
Művelet : (A új) ← (A régi) ∀ [(H) (L)].
Utasításkód: AEH. Két gc, 7 gá. Módosított flag-ek: S, Z, P, CY, AC. Pl.: XRA A.

XRI data 8 (exclusive OR immediate)
Művelet: (A új) ← (A régi) ∀ data 8.
Utasításkód: EEH, d8. Két gc, 7 gá. Módosított flag-ek: Z, S, P, CY, AC.
Pl.: XRI 7DH.

CMP r (compare register)
Művelet: (A) - (r).
Az utasítás elvégzi a regiszter tartalmának kivonását az akkumulátorban lévő számból, de az **akkumulátor tartalma változatlan marad**, csak a jelző bitek változnak.
Z ← 1, ha (A) = (r),
CY ← 1, ha az akkumulátor tartalma kisebb az r regiszter tartalmánál.
Utasításkód: 10111SSS. Egy gc, 4 gá. Módosított flag-ek: Z, S, P, CY, AC.
Pl.: CMP C.

CMP M (compare memory)
Művelet: (A) - [(H) (L)].
A művelet leírása megegyezik a CMP r utasításéval az operandus értelemszerű módosításával.
Utasításkód: BEH. Két gc, 7 gá. Módosított flag-ek: Z, S, P, CY, AC.

CPI data 8 (compare immediate)
Művelet: (A) - data 8. A művelet leírása megegyezik a CMP r utasításéval az operandus értelemszerű módosításával.
Utasításkód: FEH, d8. Két gc, 7 gá. Módosított flag-ek: Z, S, P, CY, AC.
Pl.: CPI 0AFH.

RLC (rotate left)
Művelet: az **akkumulátor tartalma** egy helyiértékkel **balra** lép. Az A7 bit az A0-ba és a CY-ba kerül. A CY eredeti értéke elvész. Az utasítás csak a CY flag-et módosítja. Az utasítás hatására 8 bit forog.
Utasításkód: 07H. Egy gc, 4 gá.

RRC (rotate right)
Művelet: az **akkumulátor tartalma** egy helyi értékkel **jobbra** lép. Az A0 bit értéke az A7-be és a CY-ba kerül. A CY eredeti értéke elvész. Az utasítás csak a CY flag-et módosítja. Az utasítás hatására 8 bit forog.
Utasításkód: 0FH. Egy gc, 4 gá.

RAL (rotate left through carry)
Művelet: az **akkumulátor tartalma** a CY flag-en keresztül egy helyiértékkel balra lép. CY ← (A7), A∅ ← (CY).
Utasításkód: 17H. Egy gc, 4 gá. Csak a CY flag módosul. Az utasítás végrehajtása során a CY eredeti értéke nem vész el (9 bit forog).

RAR (rotate right through carry)
Művelet: az **akkumulátor tartalma** egy bittel **jobbra** forog. A CY eredeti értéke az A7-be, az A0 eredeti értéke a CY-ba íródik (9 bit forog).
Utasításkód: 1FH. Egy gc, 4 gá. Csak a CY flag módosul.

CMA (complement accumulator)
Művelet: (Aúj) ← ($\overline{A}$ régi).
Az utasítás **bitenként invertálja az akkumulátor** tartalmát.
Utasításkód: 3FH. Egy gc, 4 gá. Az utasítás csak a CY jelzőbitet módosítja.

STC (set carry)
Művelet: (CY) ← „1".
Utasításkód: 37H. Egy gc, 4 gá. Az utasítás a CY flag-et állítja.

CMC (complement carry)
Művelet: (CY új) ← ($\overline{C}$ régi). Egy gc, 4 gá.

7.2.2.2.4. Vezérlésátadó utasítások

Az ebbe a csoportba tartozó utasítások a **programvégrehajtás menetét befolyásolják. Ezek az utasítások a flag biteket nem változtatják meg.** A feltételes vezérlésátadó utasításoknál a CCC bitek az alábbiak szerint jelölik az ugrási feltételt:

Feltétel			CCC
NZ	nem zérus,	Z = 0	000
Z	zérus,	Z = 1	001
NC	nincs átvitel,	CY = 0	010
C	van átvitel,	CY = 1	011
PO	páratlan paritás,	P = 0	100
PE	páros paritás,	P = 1	101
P	pozitív,	S = 0	110
M	negatív,	S = 1	111

A feltételes vezérlésátadó utasítások attól függően, hogy a feltétel teljesül vagy nem teljesül, a vezérlés más-más idő alatt hajtódnak végre.

JMP addr 16 (jump)
Művelet: (PC új) ← (3. bájt) (2. bájt), azaz feltétel nélküli ugrás az addr 16 címre.
Utasításkód: C3H, addr 8l, addr 8h. Három gc, 10 gá.

Jcond addr 16 (conditional jump)
Művelet: (PC új) ← (3. bájt) (2. bájt) ha a feltétel teljesül, egyébként (PC új) ← (PC régi + 3).
Utasításkód: 11CCC010, addr 8l, addr 8h. Két vagy három gc és 7 vagy 10 gá.
A CCC-től függő utasításvariációk: JNZ, JZ, JNC, JC, JPO, JPE, JP, JM.

CALL addr 16 (call subroutine)
Művelet:
((SP) - 1)) ← (PCH)
((SP) - 2)) ← (PCL)
(SP) ← (SP-2)
(PC új) ← (3. bájt) (2. bájt).
Feltétel nélküli szubrutin hívó utasítás, amelynek végrehajtása a PC tartalmának a verem memóriába történő mentésével indul, ezután pedig a szubrutin kezdő címének PC-be töltésével fejeződik be, aminek hatására a program végrehajtása a szubrutin végrehajtásával folytatódik.
Utasításkód: CDH, addr 8l, addr 8h. Öt gc, 18 gá.

C cond addr (condition call)
Művelet: ha a feltétel teljesül, akkor ugyanaz játszódik le, mint a feltétel nélküli CALL utasítás esetén, ha a feltétel nem teljesül, akkor a program a C cond utasítás utáni utasítással folytatódik.
Utasításkód: 11CCC100, addr 8l, addr 8h. A feltétel teljesítése esetén 5 gc, 18 gá, egyébként 2 gc, 9 gá.
Utasítás változatok: CNZ, CZ, CNC, CC, CO, CE, CP, CM.

RET (return from subroutine)
Művelet:
(PCL) ← ((SP-2))
(PCH) ← (SP)) + 1
(SP) ← (SP) + 2).
Feltétel nélküli, a szubrutinból visszatérést eredményező utasítás.
Utasításkód: C9H. Három gc, 10 gá.

R cond (conditional return from subroutine)
Művelet: ha a CCC feltétel teljesül, akkor ugyanaz játszódik le, mint a RET utasításnál, ha nem, akkor a következő utasítás hajtódik végre.
Utasításkód: 11CCC00. Ciklusok száma: 4/1, gépi állapotok száma: 12/6.
Utasítás változások: RNZ, RZ, RNC, RC, RO, RE, RP, RM.

RST n (restart)
Művelet:

((SP) - 1)	←	(PCH)
((SP) - 2)	←	(PCL)
(SP)	←	(SP)-2)
(PC új)	←	8. NNN.

Szoftver úton kezdeményezett megszakítás, ami egy egy-bájtos szubrutinhívásnak felel meg. A szubrutin kezdő címét az NNN bináris szám 8-szorosa adja: 0, 8, 16, 24, 32, 40, 48, 56.
Utasításkód: 11NNN111. **Három gc, 11 gá. Hardver úton** kezdeményezett szubrutinhívás esetén (interrupt) az NNN értékét a megszakításkérés prioritás szintje adja meg, amely az **adatbuszon keresztül kapuzódik a PC-be** a következők szerint: PC: 0000 0000 00NN N000.

PCHL (jump H and L indirect)
Művelet:
(PCH) ← (H)
(PCL) ← (L).
Feltétel nélküli ugrás a HL-ban tárolt címre. Utasításkód: E9 H. Egy gc, 6 gá..

7.2.2.2.5. Stack, I/O és a gépi vezérlés utasításai

PUSH rp (push register pair into stack)
Művelet: ((SP) - 1) ← rh
((SP) - 2) ← rl
(SP) ← (SP) - 2.
A regiszterpárok (BC, DE, HL, PSW) regiszterpárok tartalmának mentése a stackbe. Utasításkód: 11RP0101. Három gc, 13 gá.
Utasítás változatok: PUSH PSW, PUSH B, PUSH D, PUSH H.
A PSW (program status word) az akkumulátor tartalmából (8 bit) és a flag regiszter tartalmából (SZ0AC0P.1 képzett) 2 bájtos szó.

POP rp (pop register pair from stack)
Művelet:
(rl) ← ((SP))
(rh) ← ((SP)) + 1
(SP) ← (SP) + 2.

Regiszterpár visszatöltése a stackből.
Utasításkód: 11RP0001. Három gc, 10 gá.
Utasítás variációk: POP PSW, POP B, POP D, POP H.
A regiszter mentés és visszatöltés helyes sorrendje pl.: PUSH PSW, PUSH D...POPD, POP PSW.

XTHL (exchange stack top with H and L)
Művelet: (L) ⟷ ((SP))
(H) ⟷ ((SP +1)).
Utasításkód: E3 H. Öt gc, 16 gá. Az SP változatlan marad.

SPHL (move HL to SP)
Művelet: (SP) ← (H) (L).
A HL regiszterpár 16 bites tartalmát az utasítás betölti az SP-be.
Utasításkód: F9H. Egy gc, 6 gá. A flag-eket nem módosítja.

IN port addr8 (input from port)
Művelet: (A új) ← (port). Az utasítás hatására az utasítás 2. bájtjában megadott portról az adat az akkumulátorba kerül.
Utasításkód: DBH, addr8. Három gc, 10 gá. Pl.: IN 09H.

OUT port addr8 (output to port)
Művelet: (port) ← (A).
Az utasítás hatására az **akkumulátor tartalma kikerül** a 2. bájtban megadott **portra.** Utasításkód: D3H, addr8. Három gc, 10 gá. Pl.: OUT 9DH.

EI (enable interrupt)
Az utasítás végrehajtása után a CPU megszakítási rendszere engedélyezett állapotba kerül (INTE = 1). Utasításkód: FBH. Egy gc, 4 gá.

DI (disable interrupt)
A DI utasítás végrehajtása után közvetlenül a CPU megszakítási kéréseket nem fogad el (INTE = 0). Engedélyezett állapot EI hatására jöhet létre.
Utasításkód. F3H. Egy gc, 4 gá.

HLT (halt)
A CPU a HLT utasítás hatására leáll, HALT állapotba kerül. A regiszterek és a flag-ek tartalma nem változik meg. Utasításkód: 76H.

NOP (no operation)
A NOP utasítás hatására nincs változás a memória, a portok vagy a CPU regisztereinek állapotában. A NOP utasítás hatására a CPU nem végez műveletet. Utasításkód: 00H. Egy gc, 4 gá.

SIM (set interrupt mask)
Művelet:

(RST 5.5 maszk)	←	(A0), ha (A3) = 1
(RST 6.5 maszk)	←	(A1), ha (A3) = 1
(RST 7.5 maszk)	←	(A2), ha (A3) = 1
(A3)	←	maszk beállítás (A2, A1, A0) engedélyezése
(A4)	←	RST 7.5 megszakítás maszk töltése
(A6)	←	soros kimenet (SOD) beállítás engedélyezése
SOD	←	(A7), ha (A6) = 1.

A SIM **többfunkciós utasítás**, amely a végrehajtás előtt az akkumulátorban létrehozott adat alapján maszkolhatja az RST megszakításbemeneteket és beállíthatja a soros kimenetet. Az RST 5.5, RST 6.5, RST 7.5 megszakításbemeneteket le lehet maszkolni, ha az (A3) = 1 és a megfelelő akkumulátor bit (A0, A1, A2) tartalma 0. Ha A3 = 0, akkor a maszkbitek értéke közömbös.
Utasításkód: 30H. Egy gc, 4 gá.

RIM (read interrupt mask)
Művelet:

(A0)	←	RST 5.5. mask
(A1)	←	RST 6.5. mask
(A2)	←	RST 7.5. mask
(A3)	←	INTE
(A4)	←	RST 5.5. mask
(A5)	←	RST 6.5. mask
(A6)	←	RST 7.5. mask
(A7)	←	SID.

A RIM utasítás szintén **két funkciós** utasítás, amely létrehozza a 8085 megszakításrendszerére és a soros bemenetének állapotára vonatkozó bájtot. Az állapotszó tartalmazza az RST bemenetek állapotára és maszkjára vonatkozó adatokat, valamint a megszakításkérés elfogadásának engedélyezettségét (INTE). A legmagasabb biten a soros bemenet (SID) aktuális értéke jelenik meg.
Utasításkód: 20H. Egy gc, 4 gá.

7.2.3. Mikroszámítógépek programozási technikája

A mikroszámítógépes rendszerek programozása három szinten történhet: gépi kódban, assembly nyelven ill. magas szintű nyelven.

a) **Gépi kódú programozás** esetén a bináris ill. hexadecimális kódot használják. Ez a **legalacsonyabb** szintű programozás. Ma már csak a **beüzemelés, hibakeresés** esetén használatos. Példa a gépi kódú (hex) programra: 3A, 5B, 32, 86, 66.

b) Az **assembly nyelvű programozás** az utasítás mnemonikus kódján, és további, a programozást segítő hivatkozásokon alapszik. Az **assembly programozás** - a gépi kódúhoz hasonlóan - **mikroprocesszor függő**, így létezik Z80 assembly, 8086 assembly, M68000 assembly, stb. Az assembly nyelvű utasítás formátuma:
CIMKEMEZŐ MNEMONIKUS KÓD OPERANDUSOK MEGJEGYZÉS.
A **cimkemezőben** szereplő cimke használata opcionális, 5 alfanumerikus karakterből áll. A cimkét „:" kell, hogy kövesse. A megjegyzés szintén opcionális, első karaktere kötelezően „;". Az **operandus mezőben** az információ típusa szerint előfordulhat regiszter, regiszterpár, adat vagy memória cím. Az információ specifikálásnak módja lehet hexadecimális (H), decimális (D), oktális (O), bináris (B) vagy ASCII konstans. Az assembly nyelv számos **álutasítást (direktívát) tartalmaz**. Ilyenek: ORG, SET, EQU, MACRO, END MACRO stb. Az álutasítások csak az assembly nyelvben vannak értelmezve, ezeket az utasításokat a processzor nem érti, így **gépi kódjuk sincs.**

Az assembly nyelvű programfejlesztést a CP/M operációs rendszer támogatta leginkább. Ez minimálisan az alábbi funkcionális szoftvereket tartalmazza: EDITOR a programszerkesztéshez, ASSEMBLER a tárgykód előállításához, DISASSEMBLER a tárgykód assembly nyelvű visszaállításához, DEBUGGER a program hibakereséséhez. A makrók (ld. később) kezeléséhez a MACROASSEMBLER alkalmas. Példaként a DISASSEMBLER-re a 3E 2B hexadecimális kódot MVI A, 2BH alakúra fordítja. Az ASSEMBLER olyan fordító program, amely a **felhasználó által megírt forrásnyelvű programot gépi kóddá fordítja.** Ez a fordítás rendszerint **két menetben történik.** Az első menetben valamennyi mnemonikus kód ill. direktíva beolvasása és formai ellenőrzésre kerül sor. Ebben a fázisban a címkék címazonosítása is megtörténik. A gépi kód generálása a **második menetben** történik meg. Amennyiben a programfejlesztés nem a fejlesztőgép saját processzorára történik, akkor **kereszt (cross) assemblerre** van szükség.

c) A személyi számítógépek tömeges elterjedésével a mikroszámítógépes rendszerek programfejlesztése a **magasszintű nyelven** (rendszerint C

nyelven) történik és az assembly nyelvű ill. gépi kódú program generálása fordító programmal történik. A magasszintű nyelven megírt programot **kompilátor** (compiler) révén fordíthatjuk gépi kódra, assembly nyelvre. Program kidolgozásának a célja az, hogy meghatározzuk az **utasítások megfelelő sorozatát,** majd azt a végrehajtás céljából betöltjük a mikroszámítógép programmemóriájába. Az összetettebb feladatok kidolgozását a **feladat részekre bontásával** kezdjük. A leggyakrabban használt célravezető módszer a feladat részfeladatokra bontása, majd további bontása. Egy technológiai feladat legegyszerűbben **folyamatábrán** írható le. A **folyamatábra** (flowchart) a feladatot leíró algoritmus lépéseinek, a részfeladatoknak a sorrendjét és az információk áramlását grafikusan szemlélteti. A folyamatábrák készítésének három szintjét különböztetjük meg:

a) feladat megfogalmazási szint
b) algoritmus szint
c) utasítás szint.

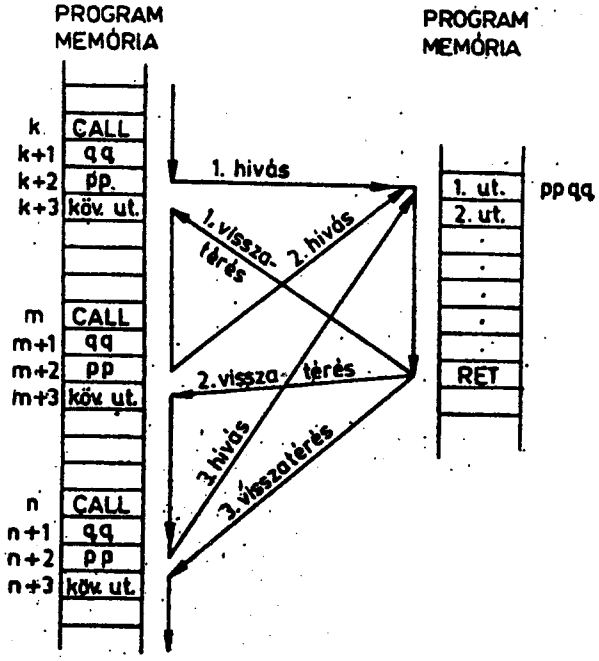
Az a) és b) szint elkészítéséhez még nem szükséges a mikroszámítógép típusának és utasításkészletének az ismerete. Az utasítás szintű folyamatábrák már figyelembe veszik a mikroszámítógép felépítését, utasításait.

7.2.3.1. Szubrutin kezelés

A **szubrutin** (alprogram) az **ismételten előforduló** feladat részek programozásának hatékony eszköze. A **szubrutin egy olyan önálló funkciójú utasítás sorozat,** amely **egyszer szerepel** a programban, de a feladat elvégzésekor többször lehet használni (hívni). A szubrutin utolsó utasítása RET, RET IF utasítás kell legyen. A **szubrutinokat rendszerint a program végén** helyezik el. Példaként néhány tipikus szubrutint említünk: BCD kivonó szubrutin, bináris szorzás, stb. A programozó maga is készíthet és **definiálhat saját szubrutint** és használhat **könyvtári szubrutint.** A szubrutin annál előnyösebb a tárkapacitás szempontjából, minél többször van felhasználva és minél több utasításból áll. Egy szubrutin többszöri hívását szemlélteti a 7.7. ábra. Megfigyelhető, hogy **egyetlen szubrutin ismételt hívása 2 bájtot igényel a verem memóriában.**

A **szubrutinok egymásba ágyazhatók** (skatulyázhatók), azaz a **szubrutin maga is tartalmazhat újabb szubrutin hívást.** Egy **többszörösen beágyazott szubrutin kezelés** lefolytatását mutatja a 7.8. ábra.

Kövesse végig a beágyazott szubrutin kezelés stack műveleteit, figyelje meg, hogy minden egyes újabb beágyazott szubrutinhívás 2 bájtot foglal le a stackben. **Szubrutin kezeléssel kapcsolatos programozási hibák**: a szubrutinba JUMP utasítással való belépés; az SP megváltoztatása a szubrutin területen belül; az SP által kezelt memória rekesz tartalmának átírása, stb.

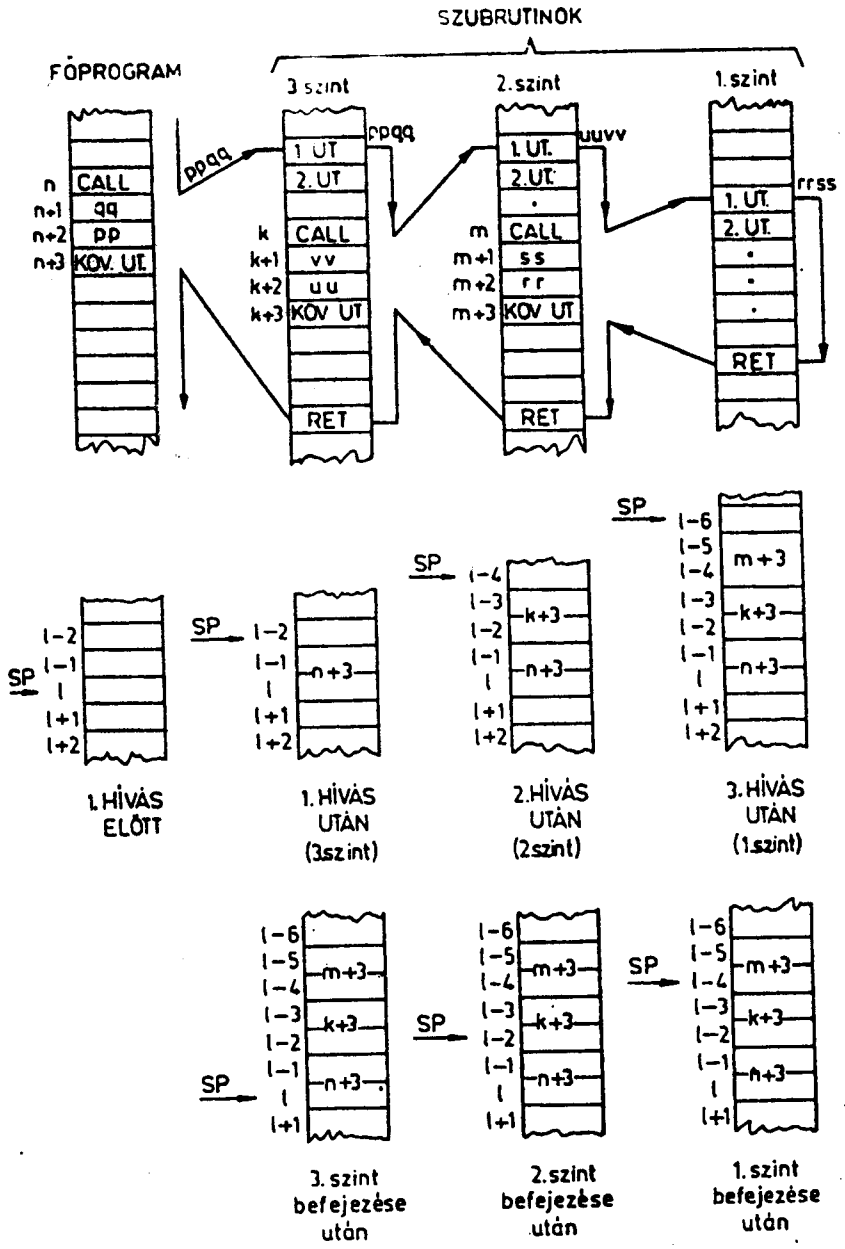


7.7. ábra

Példa: ASCII → BCD konvertáló szubrutin:

ASBCD	CPI	30H	; karakter vizsgálat (alsó)
	JM	ERROR	; ha negatív, ugrás ERROR-ra
	CPI	3AH	; karakter vizsgálat (felső)
	JP	ERROR	; ha pozitív, ugrás ERROR-ra
	ANI	0FH	; a 4...7. bitek maszkolása
	RET		
ERROR	STC		; carry bit beállítása.

Szubrutinok megadásánál megadják a szubrutin által **módosított regiszterek** nevét ill. a jelzőbiteket. Ilyen esetben a **szubrutin hívása előtt el kell menteni** az érintett regiszterek tartalmát a stack-be (PUSH) és hívás utáni részben először vissza kell tölteni az elmentett adatokat (POP). Biztonsági okból gyakran a **processzor teljes állapotát elmentik** (4 x PUSH). Fenti mentési ill. visszatöltési műveleteket gyakran a **szubrutin területén** oldják meg, ilyenkor a szubrutin nem módosítja a CPU állapotát. Ez a szubrutin által megvalósított feladattól is függhet.

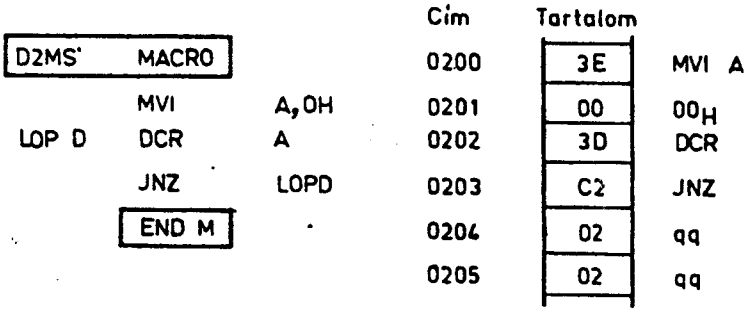


7.8. ábra

7.2.3.2. Makrók

Az **ismétlődő programrészek** realizálását kétféleképpen építhetjük be a programba: szubrutinként (ld. előbb) ill. makrók formájában. A makrók

assembly szintű utasítások egy csoportját jelentik, amelyeket MACRO....END MACRO utasítások közé írva definiálunk és a makró elé a cimke zónába írt névvel látunk el. Példaként egy 2 ms-os időkésleltetést realizáló makrót szemléltet a 7.9. ábra.



7.9. ábra

Az időzítésnél 0,5 μs-os gépi állapot időt tételeztünk fel. A DCR utasítás 5, a JNZ 10 gépi állapot alatt hajtódik végre. A két utasítás együttes végrehajtási ideje: 15 x 0,5 μs = 7,5 μs. Mivel az akkumulátorba a 00H számot töltöttük be, ezt 256-szor dekrementálva teljesül a JNZ utasítás ugrási feltétele:

$$256 \times 7{,}5\ \mu s = 1920\ \mu s \sim 2\ ms.$$

A **makrók paraméterezéséről** akkor beszélünk, amikor a funkción belül a program különböző paraméterek révén rugalmasan módosítható. Példaként a bevezetett késleltetés makró:

DXMS:	MACRO	TIME
	MVI	A, TIME
LOOP D:.	DCR	A
	JNZ	LOOP D
	END M	

A példában a TIME paraméterrel tetszőleges (2 ms-tól kisebb) késleltetést definiálhatunk. **A makrók - a szubrutinnal ellentétben - a hívás helyén kerülnek a program memóriába és annyiszor, ahányszor hívás történik,** ennél fogva a makrók használata akkor célszerű, ha **kevés számú utasításokból** állnak (max. 10 bájt). A makrók tartalmazhatnak szubrutinokat is.
Hasonlítsuk össze a makrókat ill. szubrutinokat!

7.2. táblázat

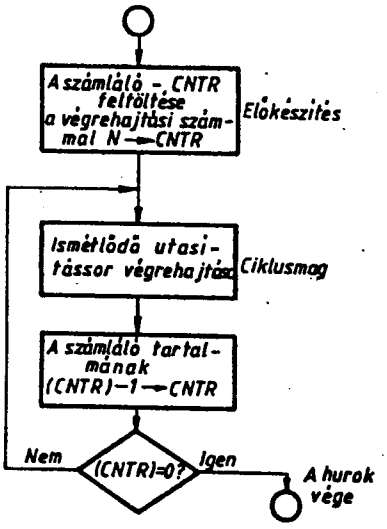
Szubrutin	Makró
A főprogramtól elkülönített területen kerül elhelyezésre.	A hívás helyén a főprogramban kerül elhelyezésre.
Van gépi kódja (CALL).	Csak az assembly nyelvű programban létezik, gépi kódja nincs.

Kezelése 2 bájt stack memóriát igényel.	Kezelése nem igényel stack műveletet.
Használata annál gazdaságosabb, minél hosszabb és minél többször kerül beírásra.	Használata csak rövid programok eseten gazdaságos.
A CALL, RET utasítások járulékos végrehajtási időnövekedést okoznak.	A makró nem jár járulékos gépi utasításokkal, ezért azonnal végrehajtódik.

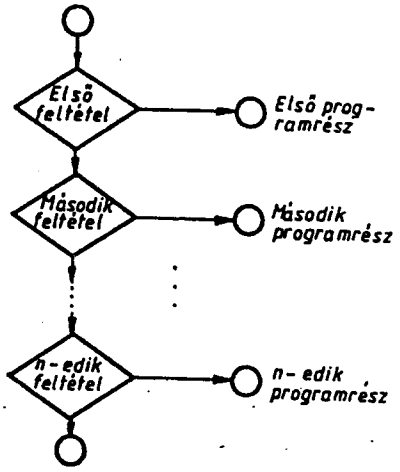
A makrók használata az **assembly nyelvű program írásánál** előnyös, de a **gépi kódú programban semmilyen előnyt** nem jelent. Az INTEL cég makró könyvtára számos makrót tartalmaz.

7.2.3.3. Programhurkok, programelágazások

A **közvetlenül egymásután végrehajtandó utasítások** szervezésének hatékony módszere a **ciklus,** amely rendszerint tartalmazza a **ciklus magot, ismétlés számlálót** és **tesztelő logikát**. A **ciklusmag az ismétlődően végrehajtandó műveleteket** tartalmazza. Az **ismétlés számláló** (rendszerint valamelyik regiszter) a ciklusmag végrehajtásának ismétlődését számlálja **felülről lefelé**. A tesztelő logika azt vizsgálja, hogy az ismétlés számláló tartalma megegyezik a kívánt értékkel. Ezt visszaszámlálás esetén a **Z bit** figyelésével oldja meg. A kívánt számú ismétlés végeztével a CPU kilép a ciklusból. A tesztelés végezhető a **ciklusmag előtt** (elől tesztelés) ill. a **ciklusmag után** (hátul tesztelés). **Egy ciklus újabb ciklust tartalmazhat** (beágyazott ciklus). Egyszeres ciklusra találhatunk példát a 2 ms-os makró (ld. előbb) leírásánál. A ciklust a címkében rendszerint LOOP-pal jelölik. Tipikus **ciklus programozási hibák**: az ismétlés számláló beállításának elhagyása, JMP utasítással belépés a ciklusmagba, végtelen ciklus, stb. **Hátul tesztelő ciklus** folyamatábrája látható a 7.10. ábrán. A mikroprocesszorok nagy része tartalmaz közvetlen **hurokképző utasítást** (DJNZ). Főleg vezérléstechnikai alkalmazásoknál gyakori a sok feltételtől függő **program elágazás.** Több feltétel esetén az elágazások egyik módja az egymás utáni feltételvizsgálatok alkalmazása (7.19. ábra). Nagy számú elágazási feltétel esetén hatékonyabb lehet az **elágazási táblázatok** használata. Ez a tárban lévő táblázat tartalmazza a különböző algoritmus lépésekhez tartozó programrészek kezdőcímeit, amelyekre közvetett címzéssel térhetünk rá.



7.10. ábra



7.11. ábra

7.2.3.4. Megszakítás kezelés

A számítógépi adatfeldolgozási folyamatok közben keletkező eseményeket a következőképpen csoportosíthatjuk:

- vannak a program futása szempontjából, jól meghatározható helyen, időpontban **várható események**, ezeket **szinkron eseményeknek** nevezik (ilyenek pl.: nullával való osztás, túlcsordulás stb.)
- vannak a program futása során várható, de időpontjuk szempontjából ismeretlen **váratlan események**, amelyeket aszinkron, várható eseményeknek nevezik (ilyenek pl. adatok beolvasása, kiíratása, DMA vezérelt adatátvitel)
- vannak olyan események, amelyek **váratlanok** és időpontjuk ebből származóan nem meghatározható, ismeretlen, ezek az események az **aszinkron,** váratlan események (pl. áramkimaradás, hardver hiba, technológia hiba).

Ezen események kiváltó oka lehet maga a **program a szinkron eseményeknél** ill. lehet a **hardver** az **aszinkron** események esetén. Az ilyen események feldolgozására szolgálnak a **megszakítások** ill. a mikroszámítógép megszakítási rendszere. A **megszakítási kérelem** egy jelzés a processzor számára valamely esemény bekövetkezésekor, amely valamilyen kiszolgáló folyamatot igényel. A program megszakítás a futó folyamat felfüggesztése a megszakítási kérelem hatására annak kiszolgálása céljából. A **megszakítási kérelem kiszolgálására egy hardver-szoftver együttes szolgál** és a kérelem hatására végrehajtott tevékenységsor a megszakítási kérelem **kiszolgálása.** A megszakítási események kapcsán különbséget kell tenni a többnyire **külső eredetű**

megszakítások (interrupt) és az utasítások szabályszerű végrehajtását megállító kivételek (exception) között. Egyes eszközök esetében a **megszakítás engedélyezhető** (enable - EI) ill. **letiltható** (disable - DI). A megszakítási kérelmek között vannak **maszkolható** és **nem maszkolható kérelmek** (pl. RST 5.5, RST 6.5, stb.). A megszakítási kérelmeknek két forrása lehet: hardver ill. szoftver. A **szoftver megszakítási kérelmek azok, amelyek programból lettek kezdeményezve**, nem maszkolhatóak. A **hardver megszakítási kérelmek többsége maszkolható**, de van **nem maszkolható** (NMI = Non Maskable Interrupt) kérelem is. Az NMI-t rendszerint olyan eseményhez rendelik, amely valamilyen **súlyos hardver hiba esetén** következik be (pl. áramkimaradás, tűz, stb.). A megszakítások kiszolgálásakor több olyan kérdés van, melyet a rendszernek meg kell oldania. ilyenek:

- a **megszakítási kérelem helyének** a megállapítása, (melyik eszköz kezdeményezte a megszakítást)
- a **megszakítások prioritásának meghatározása**, azaz a megszakítási sorrend meghatározása, több, **egyidőben jelentkező kérelem** esetén
- **egyes eszközök ideiglenes kizárása** a megszakítási procedúrából, azaz a **megszakítások maszkolása**
- **többszörös megszakításkiszolgálás** megoldása.

A megszakítási kérelem **keletkezési helyének megállapítására** két módszer alkalmazható: a szoftver és a hardver módszer. **Szoftver módszer esetén** az operációs rendszer részét képező rutin bizonyos időközönként sorra megvizsgálja a szóbajöhető eszközök állapotjelzőjét és amelyiknél a megszakítási kérelem jelentkezik, ott elindítja az aktuális eszközökhöz tartozó kiszolgáló programot. Ezt az eljárást nevezik **lekérdezéses megszakításkezelésnek (polling interrupt).** A **hardver módszerek** esetében a **megszakításvezérlő** határozza meg program segítségével (vagy anélkül) a beérkező kérelmek kiszolgálását. A mikroszámítógépek megszakításrendszerei egy vagy több megszakítási vezetékkel rendelkeznek. A megszakítási rendszer nagymértékben processzorfüggő. **Egy megszakítási vonal esetén több kérés VAGY kapuval eszközölhető.** Ilyenkor a keletkezés helyének meghatározása történhet szoftver úton, **lekérdezéses (polling)** módszerrel. Ez esetben a beérkező INT jel egy kiszolgáló rutint indít el a processzor segítségével. Ez a rutin végig vizsgálja az eszközöket (a VAGY kapu bemeneteit) és így állapítja meg a kérő eszközt. A kérelem helyének hardver úton történő meghatározására példaként a Z80 processzornál kialakított **daisy chain** módszert említjük, amelynél a I/O eszközök INT-IACK be/kimenetei **sorba vannak kapcsolva**, így a visszaigazoló (IACK) jel a kiszolgálást kérő eszköztől már nem halad tovább, hanem elindítja a kiszolgáló rutint. Több megszakítási vonal esetén, **amikor minden eszköz saját megszakítást kérő vezetékkel** rendelkezik, a kérelem helye egyértelműen megállapítható és a hozzá tartozó kiszolgáló rutin elindítható. A megszakítási rendszerek legáltalánosabban használt formája a **vektoros módszer. Vektoros módszer** esetén a megszakításkérő eszköz

valamilyen módon a **kiszolgáló rutin** (int. service routine) **kezdő címét** határozza meg. A következő formákat alkalmazzák:

- a megszakításkérő eszköz az őt kiszolgáló **rutint hívó utasítást** (CALL INT) teljes egészében **átadja a processzornak**
- a megszakítást kérő eszköz annak **a memóriának a címét adja át a processzornak,** amelyben a kiszolgáló rutint hívó utasítás (CALL INT) található, azaz ennek a címnek a tartalmát a processzor FETCH ciklussal lehívja és elindítja a kiszolgáló rutint
- a **megszakításkérő eszköz az őt kiszolgáló rutinnak a kezdő címét adja át** a processzornak, amely azt mint a soronkövetkező utasítás címét betölti a PC-be és onnét folytatja az utasítások feldolgozását
- a megszakítást kérő eszköz egy **sorszámot** ad át a processzornak, amely sorszám a megszakításokat kiszolgáló rutinok kezdőcímeit tartalmazó táblázatban, a **megszakítási vektortáblában,** kijelöli az adott eszközt kiszolgáló rutin kezdőcímét, ez a módszer a **vektoros megszakítás-kiszolgálás** (vector interrupt) leggyakrabban alkalmazott módszere.

Nagyon gyakran a hardver megszakítási vonalak (INT jelek) fixen összekapcsoltak a vektortáblázat meghatározott bejegyzéseivel. Ezt a módszert alkalmazzák az IBM-PC gépek esetén is. A 16 bites INTEL processzorok esetében a megszakítási vektorszám vagy a vektortábla egy 4 bájtos sorát jelöli ki, amely egy szegmens címet és azon belüli **eltolást (relatív címet)** tartalmaz, vagy védett üzemmódban egy **táblázatnak** (IDT = Interrupt Descriptor Table) egy sorára mutat. A táblázatban található deszkriptor adja meg a kiszolgáló rutinnak a helyét a tárban.

A megszakítási kérelem helyének és a kiszolgáló rutin kezdőcímének meghatározása után a processzor elindíthatja magát a **kiszolgáló eljárást** (kiszolgáló rutint). Ennek lépései:

hardver által:

- a **megszakításkérő eszköz** beállítja processzor INT vonalán a hatásos jelet, jelezve ezzel a megszakítási kérelmet
- a **processzor az éppen futó utasításciklus befejezése után elfogadja az INT kérést és visszaigazolja** (nyugtázza) a kérés elfogadását (INTACK/IACK)
- a processzor **tárolja a megszakítási vektor** sorszámát
- a **processzor elmenti az utasításszámláló** (PC) tartalmát a veremmemóriába
- a processzor **betölti a kiszolgáló rutin kezdő címét az utasításszámlálóba,** majd elkezdi a kiszolgáló rutin végrehajtását.

Szoftver által:

- a megszakított feldolgozás **részeredményeit elmenti** a regiszterekből a stack-be

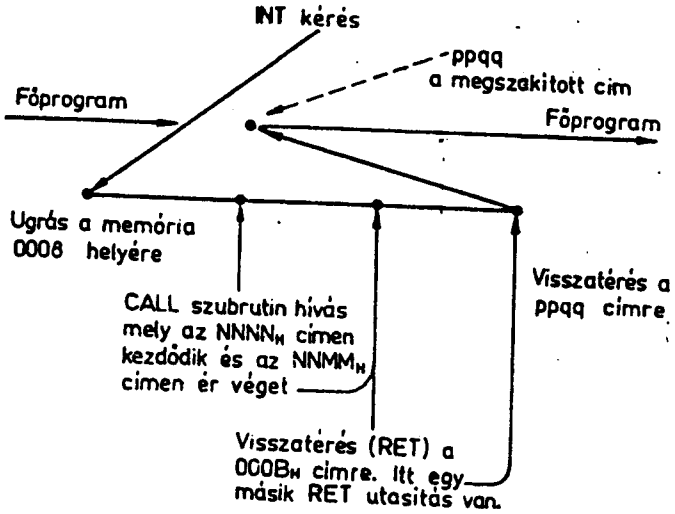
- ha a megszakítást kiszolgáló rutin több eszközhöz is tartozhat, akkor a megszakításkérő eszköz **azonosítását** (eszközazonosítót) tárolja
- a megszakításhoz tartozó **esemény kezelése**
- a **megszakítás kiszolgálása befejeztének jelzése**
- a **felfüggesztett adatok visszatöltése** a CPU regisztereibe
- a kiszolgáló rutin befejezése, **visszatérés, a feldolgozás folytatása** a hardver által
- az elmentett PSW és PC visszatöltése, a feldolgozás folytatása.

A **prioritások** kezelése azaz az egyszerre bekövetkező megszakítási kérelmek feldolgozása többféle megoldással történhet. A **prioritás megállapítása** történhet szoftver és hardver úton. **Szoftver úton a programbani vizsgálati sorrend** meghatározza a kiszolgálás sorrendjét és így az eszközök prioritását. A szoftver rendszernél előnyösebb a **megszakítás vezérlő alkalmazása**, amely hardver és szoftver együttes alkalmazását jelenti. Az ilyen eszközök esetén a prioritás meghatározása történhet **hardver úton** (pl. I8214 PICU) az Intelnél vagy **daisy chain rendszer** a Z80-nál) ill. **szoftver úton** (ld. 8259 Intel). Egyszintű megszakítási rendszerben nincs lehetőség a kiszolgáló rutin felfüggesztésére egy újabb megszakítási kérelem által. A **többszintű megszakítási rendszerekben a megszakítást kiszolgáló rutin is megszakítható, de csak bizonyos szabályok** betartásával. Ezek a módszerek az alábbiak:

- a kiszolgáló rutin a vele egyező, vagy **nála alacsonyabb prioritású** megszakítási kérelmeket letiltja, így azok semmilyen körülmények között nem szakíthatják meg a kiszolgálási folyamatot
- a kiszolgáló rutin a folyamat kezdetekor ideiglenesen **alacsonyabb prioritási szintre sorolja magát**, így a vele egyező vagy nála magasabb prioritású kérelmek megszakíthatják a kiszolgálás folyamatát
- a kiszolgáló rutin ideiglenesen **új prioritásokat rendel** az egyes eszközökhöz és így a kiszolgálás alatt más prioritási rend érvényesül, mint egyébként.

Az I8085-ös rendszerben használt ún. **nullalapos megszakításkérést** mutatjuk be. Az egyes INT belépési pontokat a 8085 CPU kapcsán bemutattuk. Egyszeres megszakításkérés esetén (pl. RST 1) a megszakítás felléptekor **megszakított program következő utasításának címe** (PC) a veremmemóriába kerül, majd a PC-be a 0008H, azaz az RST1 belépési pontja (1 x 8 = 08 D = 08 H) kerül. Itt 8 bájt áll rendelkezésre, mivel a RST2 belépési pontja (2 x 8 = 10 H). A 0008 H címen rendszerint egy CALL NNNN utasítás szerepel, amit egy RET utasítás követ a 000BH címen. Ez a RET a megszakításból való visszatérést jelenti. A CALL NNNN címen kezdődő kiszolgáló rutin végrehajtódik az NNNM címig, ahol a kiszolgáló rutin végét jelző RET utasítás van, ami a szubrutin végét jelzi és a 000BH címre adja át a vezérlést. A megszakításkezelés tehát **beágyazott**

szubrutinkezeléssel kerül lebonyolításra, így verem igénye 4 bájt. A kiszolgálás során lezajló eseményeket szemlélteti a 7.12. ábra.



7.12. ábra

7.2.3.5. I/O kezelés

A **perifériális eszközök kapcsolatát** a processzorral az eszközvezérlőkben található **regiszterek segítségével** oldják meg. Minden adatforgalom, parancs küldés ill. állapotlekérdezés ezeken keresztül valósul meg.

Az I/O portokat a következő regiszterek alkotják (többnyire):

a) parancsregiszter
b) állapotregiszter
c) adatkimenet regiszter
d) adatbemenet regiszter.

Gyakran az a), b), ill. c), d) funkciókat összevonják. Az I/O portokat közvetlen I/O utasításokkal ill. a memóriában leképzett módon érheti el processzor. Az adatátvitel lehet **párhuzamos ill. soros**. Párhuzamos átvitel esetén az adatszó **minden bitje egyszerre kerül átvitelre, míg soros átvitelnél az adatbitek időben egymás után kerülnek átvitelre.**

Az I/O eszközök kezelése lehet:

- **feltétel nélküli közvetlen** (direkt) átvitel (pl. kijelző vezérlés, kapcsolók beolvasása - mod0 a 8255-nél)
- **feltételes átvitel** (pl. hand-shaking átvitel - mod1 a 8255-nél)
- **DMA-s átvitel**
- **I/O processzor** alkalmazása
- **megszakításos** átvitel.

A **soros átvitel simplex, fél-duplex ill. duplex rendszerű. Simplex átvitelnél az adatátvitel csak egyirányban lehetséges. Fél-duplex átvitelnél az adattovábbítás mindkét irányban lehetséges, de egyidőben csak egy irányban. Duplex üzemmódban egyidőben mindkét irányban lehet adatokat továbbítani. A fél-duplex üzemmódhoz 2, a duplexhez 4 vezetékes kapcsolat szükséges. Az átvitel módját meghatározó szabályrendszert nevezik protokollnak.** A soros adatátvitel lehet **aszinkron** és **szinkron** ütemezésű. **Aszinkron átvitelnél a karakterek ütemezés nélkül követik egymást. Minden karakter egy start bitből, 7 adatbitből, 1 paritásbitből és 1-2 stop bitből, összesen 10-11 bitből tevődik össze.**

7.2.3.6. Bitmaszkolás, táblázatkezelés

Az assembly nyelvű programkészítés során gyakran előforduló feladat a **bájtműveletekről áttérni a bitműveletekre**, amit **maszkolással** érhetünk el, ezért gyakran **bitmaszkolásnak** neveznek. A bit maszkolást leggyakrabban inmediate jellegű AND művelettel érhetjük el: pl. ANI 01H. Ez esetben a legalacsonyabb bitet maszkoltuk, s az eredmény a zérus bitben jelenik meg. További példák: ANI 80H a legmagasabb bit maszkolása, ANI 0FH, az alacsonyabb dekád maszkolása. Megjegyezzük, hogy egyes mikroprocesszoroknál a bitvizsgálatra külön utasítást definiáltak, pl. Z80 μP esetén a TEST művelet.

A **táblázatkezelés** szintén igen gyakori művelet a programozás során. Jellegzetessége, hogy az akkumulátorba be kell tölteni a kiolvasandó rekesz sorcímét, miközben a táblázat kezdőcímét valamely regiszterpár tartalmazza. A kiolvasott adat rendszerint az akkumulátorba kerül. A 8086 μP-nál e célra az XLAT utasítást definiálták. E funkcióval pl. sín, cosín értékek kiolvasását, mérőérzékelők linearizálását, kódátalakítást stb. oldhatunk meg.

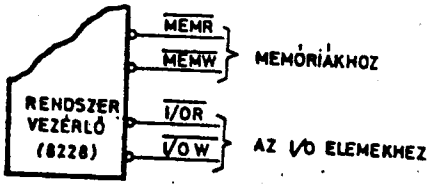
7.2.4. Beviteli/kiviteli eszközök (Input/output Devices)

A be/kiviteli eszközök (I/O) a mikroprocesszor, mikroszámítógép és a külvilág közötti kapcsolatot valósítják meg. A processzor az I/O-t a memóriához hasonlóan kezeli. Általában az I/O kezelésre jóval kevesebb utasítás áll rendelkezésre, mint a memória műveleteknél. Alapvetően kétféle I/O kezelést alkalmaznak:

a) elszigetelt I/O (isolated I/O)
b) memóriaként leképzett I/O (memory mapped I/O).

a) Elszigetelt bevitel/kivitel (Isolated I/O)

Az ilyen típusú bevitel/kiviteli módnál a **memória cím mezeje el van választva** az I/O címterétől. Ilyen esetben az I/O eszközökre csak az e célra specifikált utasítások vonatkoznak (pl. IN, OUT). További jellemzője, hogy a memória címtartományt nem befolyásolja az I/O címtartománya. Egy elszigetelt I/O-t alkalmazó rendszer vezérlő sín kialakítását szemlélteti a 7.13. ábra.



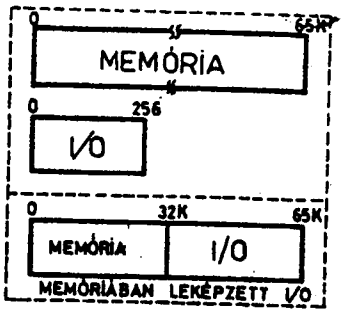
7.13. ábra.

b) Memóriaként leképzett bevitel/kivitel

Ha a memória címtartományának egy részét hozzárendeljük az I/O címtartományhoz, akkor a programozás hatásfoka javítható, mert az I/O elemeket (mint memória bájtokat ill. szavakat) ugyanazon utasításokkal tudjuk kezelni, mint a memória helyeket. Más szavakkal a **memóriára vonatkozó utasításokat kiterjeszthetjük az I/O kezelésre**. Ehhez a vezérlő sín funkcióit az alábbiak szerint kell módosítani:

- az eredeti I/OR, I/OW bitek nem kerülnek felhasználásra
- az „új" I/OR, I/OM jeleket egy járulékos logikával állítjuk elő, amelyek egyrészt az MR, MW jelekből és az A_{15} címbitből kerülnek kialakításra.

A kétféle beviteli/kiviteli mód esetén a **memória térkép** ill. I/O térkép a 7.14. ábra szerint alakul.



7.14. ábra

Természetesen a memóriaként leképzett I/O-t csak akkor lehet használni, ha a memória igény nem több mint 32 kBájt. $A_{15} = 0$ esetén a memória aktív, $A_{15} = 1$ esetén az I/O eszköz aktív, tehát az A_{15} a továbbiakban nem címbit, hanem vezérlőbit.

A **beviteli/kiviteli eszköz címzésénél** szintén kétféle módszert alkalmaznak:

a) **teljes dekódolású I/O címzés**
b) **lineáris címzés.**

A **teljes dekódolású I/O címzést** általában a nagy rendszerekben használják, mert lehetőséget nyújt a maximális I/O szám kezelésre (pl. 256). A **lineáris címzés kis rendszerek esetén használatos,** amikor az összes címinformáció

elhelyezhető egy bájtban. A lineáris címzés **előnye,** hogy a **címdekóderek elhagyhatók, hátránya a korlátozott számú I/O eszköz címezhetősége.** Könnyen belátható, hogy 8 bites I/O cím esetén max. 8 db. I/O eszköz választható ki lineáris kiválasztással. Ha az I/O eszköz belső port szelektálást is igényel (pl. 8255), akkor 6 db ilyen eszköz kiválasztása lehetséges. Lineáris címzés esetén a **címbájtnak csak egy bitje lehet aktív,** különben **sínkonfliktus** ill. hw. zárlat lép fel.

c) **Beviteli/kiviteli modulok.** A főbb beviteli/kiviteli modulok:

- programozható párhuzamos periféria illesztő egység (8255-PPI)
- programozható soros illesztő egység (8251-USART)
- programozható időzítő számláló (8253)
- programozható megszakításvezérlő (8259)
- DMA vezérlő (8257)
- Klaviatúra/kijelző vezérlő (8275).

7.3. MCS 8086 mikroszámítógép

Az MCS 86 mikroszámítógép a HMOS technológiával (High Performance MOS) készült és 3 processzorra épül (7.3. táblázat).

7.3. táblázat

uP	Technológia, Lábak	Funkció
8086	HMOS, 40	8/16 bites, általános célú központi egység (CPU), 16 bites külső adatsín
8088	HMOS 40	8/16 bites, általános célú központi egység, 8 bites külső adatsín
8089	HMOS 40	8/16 bites mikroprocesszor nagysebességű I/O műveletekhez, 8 ill. 16 bites külső adatsín

Az MCS 86 rendszer további elemei (7.4. táblázat)

7.4. táblázat

Típus		Technológia	Lábak	Funkció
8259A		NMOS Interrupt Controller	28	Programozható megszakításvezérlő
8282	8 bites Latch	Bipoláris (invertáló)	20	8 bites tároló címsínhez
8283	8 bites Latch	Bipoláris (invertáló)	20	
8284	Clock generátor	Bipoláris	18	Órajel generátor
8286 8278	8 bites sínmeghajtó	Bipoláris (invertáló)	20	Adatsín meghajtó
8288	Bus Controller	Bipoláris	20	Általános célú buszmeghajtó
8289	Bus Arbiter	Bipoláris	20	Multimaster buszvezérlő

A fenti kiegészítő elemek mellett a 8080, ill. 8085 rendszer elemei csatlakoztathatók.

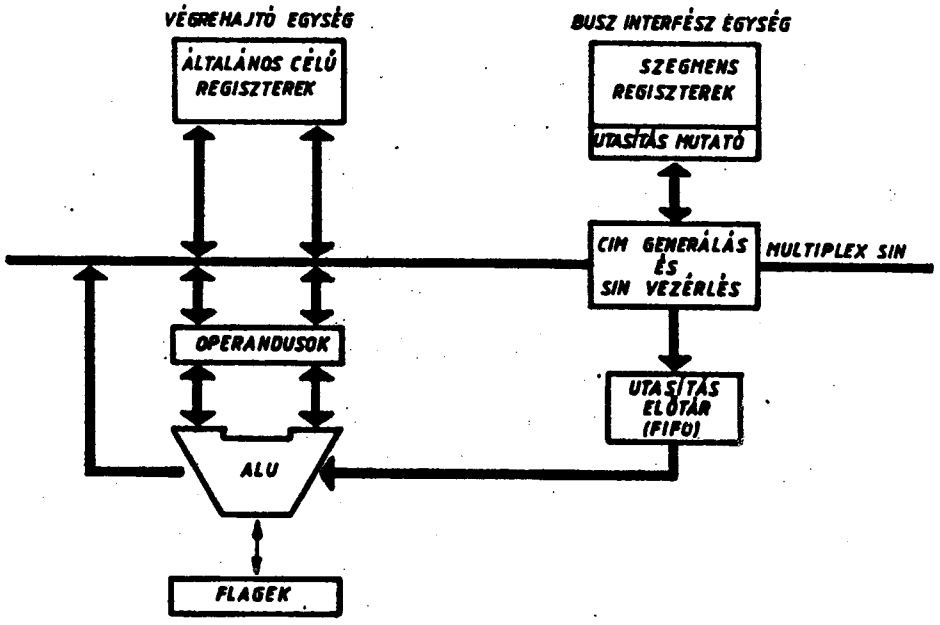
7.3.1. 8086 hardver

7.3.1.1. A 8086 ill. 8088 CPU

Ebben a részben a CPU HW-t csak olyan mélységben tárgyaljuk, amely az utasításkészlet megértéséhez szükséges. A 8088 típus 8 bites külső adatsínnel rendelkezik, míg a 8086 egyidejűleg 16 bites adatok átvitelére is alkalmas. Ennek ellenére a két processzor egymással SW kompatibilis: az egyik processzorra írt program a másikon változtatás nélkül futtatható. Általánosságban elmondható, hogy a 2 MHz-es frekvenciával működő 8080 uP-hoz viszonyítva a 8 MHz frekvenciájú 8086 mintegy **10-szer hatékonyabb.**

7.3.1.2. Processzor architektúra

A 8086 ill. 8088 két különálló egységből áll (7.15. ábra).



7.15. ábra

a) **végrehajtó egység** (Execution Unit. EU)
b) **busz illesztő egység** (BUS-Interface Unit: BIU).

A két egység adatokat cserél egymással, azonban eltérő funkciók alapján legnagyobb részben egymástól függetlenek. A BIU tartja a kapcsolatot a külvilággal.

A BIU részei:

a) regiszter készlet: ES, CS, SS, DS, IP
b) címösszeadómű
c) FIFO regiszter.

A BIU az utasításokat **előre kiolvassa** a memóriából és a FIFO-ba rakja. Az EU innen veszi ki az utasításokat és operandusokat anélkül, hogy a memória hozzáférést kivárná. A BIU az utasításlehíváson kívül elvégzi az operandus (esemény) átvitelét az EU, memória ill. I/O között, s generálja az ehhez szükséges **fizikai címeket a saját összeadóműve révén**. Ezzel a szervezéssel érték el, hogy a magas processzor sebesség ellenére relatíve lassú tárolóelemek is alkalmazhatók. A **végrehajtó egység** (EU) tartalmazza az **általános célú regisztereket, az ALU-t, operandus regisztert** és a **flag regisztert.** Az EU funkciója: a BIU által lehívott utasítások értelmezése, végrehajtása és az adatok átadása a BIU-nak. Hasonlítsuk össze a második generációs processzorok programvégrehajtási metódusát a harmadik generációs 8086-os processzoréval.

Második generációs proc.	CPU SIN	FETCH FGL.	VH -	FETCH FGL.	MR FGL.	VH -	FETCH FGL.	MW FGL.
	EU:	VH		VH			VH	
8086	BIU:	FETCH	FETCH	MW	FETCH	MR	FETCH	
8088	SIN:	FGL	FGL	FGL	FGL	FGL	FGL	

Jelölések:

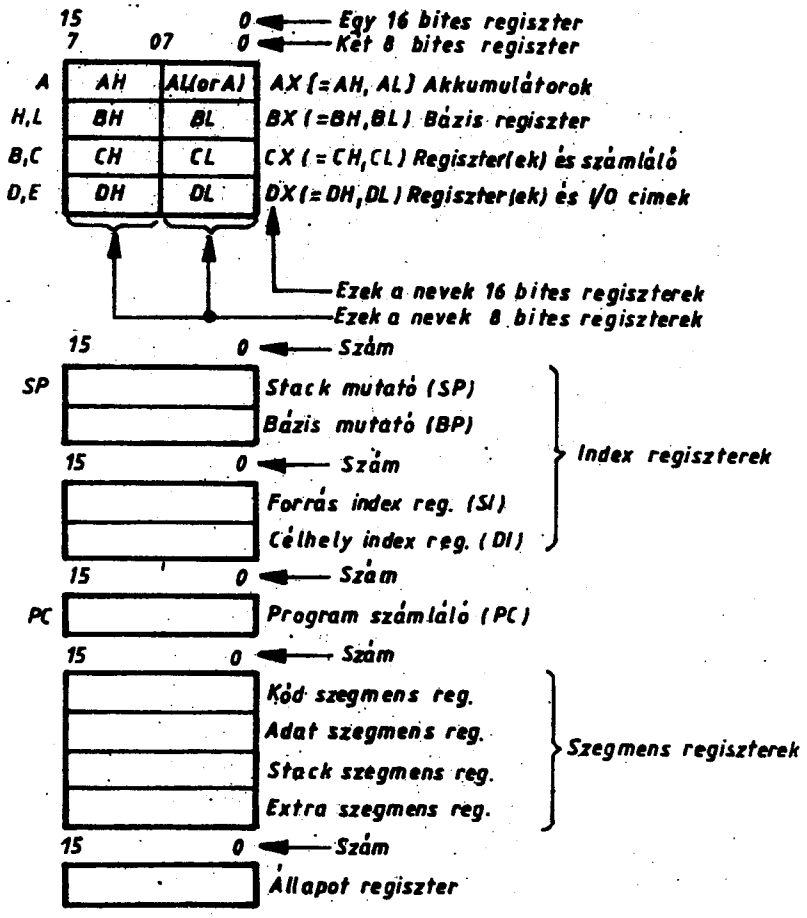
VH - végrehajtás
FGL - foglalt
MR - memória olvasás
MW - memória írás

Jól látható a 8086-os rendszer optimális sín kihasználása és az **átlapolásos** üzemmód. Az Intel cég ennél a processzornál kezdte el alkalmazni a **pipeline** feldolgozást, azaz az átlapolásos technikát, amelyet a RISC processzoroknál tökéletesítettek.

7.3.1.3. A 8086 processzor regiszterei és flag-jei

A 8086 processzor

- négy 16 bites általános célú regisztert
- két 16 bites pointer regisztert
- két 16 bites indexregisztert
- egy 16 bites programszámlálót (PC) és
- egy 16 bites flag regisztert tartalmaz (7.16. ábra).



7.16. ábra

7.3.1.3.1. Általános célú regiszterek

Az általános célú regiszterekre egymástól függetlenül, mint két 8 bites regiszterre, vagy egy 16 bites regiszterre hivatkozhatunk. Az általános célú regiszterek a 8 vagy 16 bites aritmetikai/logikai műveletek operandusait tartalmazhatják.

Az **AX regiszternek** (akkumulátor) megkülönböztetett szerepe van: minden I/O művelet ezen keresztül bonyolódik le és a közvetlen adatokon operáló utasítások egyik operandusát is az AX regiszter tartalmazza (a 8086 egycímes gép). Az AL regiszter általánosságban megfelel a 8085 processzor A regiszterének.

A **BX regiszterre, mint bázisregiszterre** (base reg.) hivatkozhatunk. Ez az általános regiszter a 8086 címképzésében vesz részt (hasonlóan a 8085 H és L regiszterpárjához). Minden memóriareferens utasítás, amely a cím

kiszámításában részt vesz, alapértelmezésben a DS (data segment) regiszter hivatkozik.
A **CX regiszter** mint **számláló** (counter) használatos. A regiszter értéke dekrementálódik a string- és a ciklusutasításokban: CX tipikus alkalmazása a ciklusok iterációs számának beállítása. Több bit-shift és rotate utasításban is részt vesz (CH a 8085 B regiszterének, CL a 8085 C regiszterének felel meg).
A **DX regiszter**, mint **adatregiszter** használatos az aritmetikai műveletekben (beleértve a szorzást és az osztást is): néhány I/O utasításban az I/O címet is tartalmazza. A DH regiszternek a 8085 D, és DL regiszternek a 8085 E regisztere felel meg.

7.3.1.3.2. Pointer regiszterek

Ezek a regiszterek a stack-szegmens adatait címezhetik meg: minden 16 bites aritmetikai/logikai műveletnek operandusai lehetnek.
Az SP regiszterre **stack-pointerként** hivatkozhatunk (8085 megfelelője a SP), amely a stack-et jelöli ki a memóriában. Minden SP-vel történő címzés **egyben implicit hivatkozás** az SS (stack segment) regiszterre is.
A **BP (base pointer)** regiszterrel adatcímzés végezhető a stack-szegmensben: tipikus felhasználási módja a **stack-ben történő paraméterátadás.**

7.3.1.3.3. Index regiszterek

Az **index regiszterekkel** memóriabeli adatok címezhetők, szerepük elsősorban a **string-műveleteknél** van. Operandusai lehetnek minden 16 bites aritmetikai/logikai műveletnek.

7.3.1.3.4. Szegmens regiszterek

A **szegmens regiszterek minden memóriacím képzésében részt vesznek**, minden szegmens regiszter egy-egy **64 kB-os blokkot határoz meg** a 8086
1 MB-os címtartományában, amelyre azután az illető szegmens regiszterrel hivatkozhatunk:
Pl. a **DS (data segment)** regiszterrel elérhető adatok alkotják az aktuális adatszegmens értéket.
A **CS (code segment) regiszter a kódszegmens** regiszter. Minden utasítás-fetch előtt a programszámláló (PC) értéke (4 bittel jobbra eltolva, ... de erről bővebben a címzési módoknál szólunk!) hozzáadódik a CS regiszter értékéhez, s az így meghatározott címről történik az utasítás lehívása.
A **DS (data segment)** az adatszegmens regiszter. Minden adatra történő hivatkozás - három kivételtől eltekintve - a DS regiszterhez képest relatív: a három kivétel a következő:

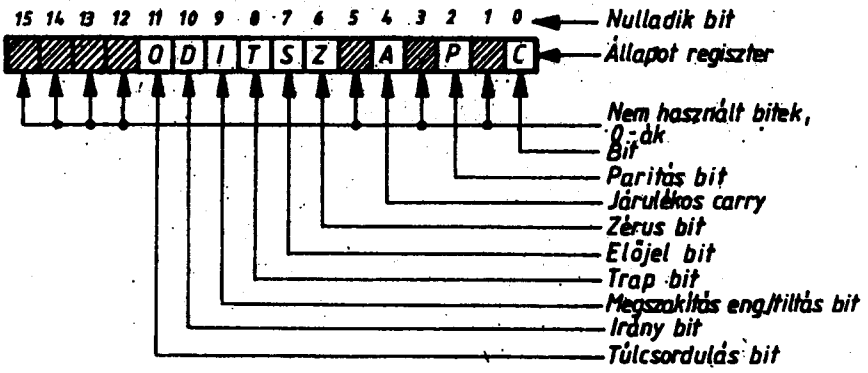
- stack címzés, amely az SP regiszterrel történik
- az adatcímzés a BP regiszter használata esetén az SS regiszterhez képest relatív
- string-műveletek alkalmával (amely a DI regisztert használja a cím képzésben) a cím relatív.

Az SS (**stack segment**) regiszter a stack szegmenst jelöli ki. Minden olyan adatra történő hivatkozás, amelyben SP, vagy a BP regiszter szerepel, az SS regiszterhez képest relatív: így minden stack-en operáló utasítás (PUSH, POP, CALL, RET és INT) az SS regisztert, mint stack-szegmens regisztert használja.
Az **ES (extra segment)** regiszter az extraszegmens regiszter, string-műveletek címképzésekor a DI regiszter adja az eltolási címet ES értékéhez képest.

A szegmens regiszterek használatára, a címképzésre a továbbiakban még visszatérünk.

7.3.1.3.5. A flag regiszter

A 8086 processzor egy 16 bites flag regisztert tartalmaz, amelyre status regiszter, vagy PSW (program status word) néven hivatkozhatunk. Az egyes bitek jelentése a 7.17. ábrán látható.
A **direction flag** értéke string műveletek alkalmával meghatározza, hogy az index-regiszterek autoincrement vagy autodecrement módban működjenek. Ha a D = 1, akkor SI és DI értékei dekrementálódnak (a string végéről kezdődik a feldolgozás), egyébként SI és DI auto-increment módban dolgoznak (a string hozzáférés a memória alacsonyabb címeiről kezdődik.
Az **interrupt flag** a master interrupt enable/disable. Ha értéke 1, akkor a megszakítás engedélyezett, egyébként minden interrupt le van tiltva.
A **trap flag** egy speciális „debug" lehetőséget ad 8086-on: ha értéke 1, akkor a processzor ún. „single step" módban működik. A carry, auxiliary carry, parity, sign és zéró flag-ek a 8085 A processzoron szintén megtalálhatók. Ami új a 8086 processzorban, az az overflow, direction, interrupt és trap flag-ek jelentése.



7.17. ábra

7.3.1.4. A 8086 processzor címzési módjai

Minden 8086 memóriacím valamely **szegmens regiszter tartalmának** és egy **effektív memóriacímnek az összege.** Az effektív memóriacím a különböző címzési módokból számítható, hasonlóan más típusú mikroprocesszorokhoz, a kiválasztott szegmens regiszter tartalma 4 bittel balra lépve adódik hozzá az

effektiv memóriacímhez (mindkettő 16 bites), így képződik a **20 bites fizikai** cím:

szegmens reg. tartalma	xxxxxxxxxxxxxxxx0000
eff. mem. cím	+ 0000yyyyyyyyyyyyyyyy
fizikai cím	zzzzzzzzzzzzzzzzzzzz

Az így kiszámított 20 bites fizikai címmel 1.048.576 bytes méretű memóriatartomány címezhető meg közvetlenül. (A szegmens regiszter tartalmára mint szegmens címre, az effektív címre pedig, mint offset (eltolási) címre fogunk hivatkozni. Mint azt már említettük, minden 8086 szegmens regiszter azonosítja a memóriának egy 65.536 byte-os tartományát. **Az aktuális fizikai cím mindig valamelyik négy memóriaszegmens tartományából kerül ki (mivel a 8086-nak négy szegmens regisztere van).**
A szegmens regiszterek tartalmára nézve nincs semmilyen megkötés, így a 8086 memóriája nincs 64 kB-os blokkokra osztva, a **szegmens regiszterek tetszés szerint átlapolhatják** egymást. A 8086 címzési módjai két fő részre oszthatók:

1) program memória címzés
2) adat memória címzés.

7.3.1.4.1. Programozási címzési módok

Amikor egy utasítás-fetch befejeződött a kiszámított fizikai címről, a programszámláló (program counter, PC) értéke az utasítás hosszával inkrementálódik, hogy a következő utasításra mutasson. Egy JUMP, vagy CALL utasítás azonban a következő három módon módosítja a PC tartalmát:

1) Program-relatív címzéssel: egy 8 vagy 16 bites eltolási címet, mint közvetlen adatot tartalmaz maga az utasítás, s ez a cím, mint előjeles bináris érték adódik a PC értékéhez. Ez a művelet a CS (code-segment) regiszter értékét természetesen nem változtatja meg. Ezt a címzést **szegmensen belülinek** nevezzük (intrasegment).

2) Direkt címzéssel: a lehívott utasítás explicit adatként tartalmaz új 16 bites címeket, amely betöltődik a PC-be és a CS regiszterbe. Erre a műveletre **interszegmens** operációként fogunk hivatkozni.

3) Indirekt címzéssel: bármely adatmemória-címzési lehetőség (amelyeket a következőkben írunk le) által szolgáltatott adatot egy JUMP vagy CALL utasítás értelmezhet memóriacímként is.

A 8086 felhasználónak kétféle indirekt címzési lehetőség áll rendelkezésére:

- a kiolvasott 16 bites adatot a PC-be töltve **változatlan CS regiszterérték** mellett értelmezi címként egy JUMP, vagy egy CALL utasítás, vagy
- két 16 bites adatot olvas ki a memóriából az utasítás, amelyek közül **az első a PC-be, a második a CS regiszterbe kerül,** s az ezután

végrehajtott JUMP vagy CALL utasítással a teljes 1 MB-os memóriatartomány elérhető.

7.3.1.4.2. Adat memóriacímzési módok

A 8086 igen változatos címzési lehetőségekkel rendelkezik, amelyeket hat alapvető osztályba sorolhatunk.

1) Közvetlen címzési mód
2) Direkt címzési mód
3) Direkt, indexelt címzési mód
4) Implicit címzési mód
5) Bázis relatív címzési mód
6) Stack címzési mód.

7.3.1.4.3. A címzési mód byte

Az előbbiekben áttekintettük a 8086 processzor különböző címzési módjait. Ebben a pontban a következő kérdésre kapunk választ: hogyan valósul meg a címzési mód deklarálása a generált object kódban? A 8086 processzor a legtöbb adatmemória címzési módot az utasítás object-kódjában egy külön byte-ban értelmezi, amelyet a **címzési mód byte**-nak fogunk nevezni. A címzési mód byte az utasítás object-kódjának mindig a második byte-ja, kivéve valamely prefix utasítást. A címzési mód bájt felosztása a következő:

mod mező	Regiszter mező	Reg/mem mező
X X	Y Y Y	Z Z Z

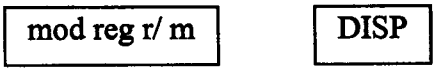
ahol

- a **mod mező** különbözteti meg a memória címzést valamelyik regiszter címzésétől. Memóriacímzés esetén a címzési mód byte-ot (vagyis ezen két X X bitet tartalmazó byte-ot követő displacement byte-ok számát adja meg.
- a **Regiszter mező** Y Y Y három bitje definiálja azt, hogy melyik regiszter vesz részt a műveletben. (A továbbiakban a regiszter mezőt reg-nek fogjuk rövidíteni).
- a **Reg/mem mező** Z Z Z három bitje a mód mezővel együtt specifikálja a címzési módot. (A továbbiakban a reg/mem mezőt R/M-ként fogjuk rövidíteni).

- **mod = 00 memóriacímzés**: a címzési módra vonatkozó további információkat a reg/mem mező tartalmazza. Az utasítás **eltolási címet** (displacement-et) **nem tartalmaz**. (Implicit utasításhossz megadás).
- **mod = 01 memóriacímzés**: a címzési módra vonatkozó további információkat a reg/mem mező tartalmazza. Az utasítás negyedik byte-ja **egy bájtos displacement-et** tartalmaz. Ez a displacement a + 127, - 128 tartományban változhat. Amikor ezt a displacement-et memóriacím

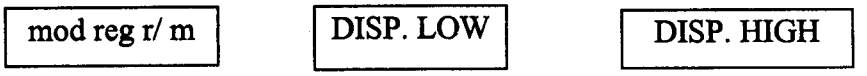
számításra használjuk az utasítás harmadik byte-ja a kiterjesztett előjelet tartalmazza: u., mint a direkt, indirekt címzési mód displacement-jei.

A címzési mód byte-jai most ilyenek:



ahol mod = 01 és DISP = 8 bites előjeles bináris szám.

- **mod = 10 memóriacímzés**, a címzési módra vonatkozó további információt a Reg/meg mező tartalmazza. Az utasítás harmadik és negyedik byte-ja **16 bites displacementet** ad. A harmadik bájt tartalmazza a nyolc alacsonyabb helyiértékű bitet, míg a negyedik a nyolc magasabb rendű bitet tárolja. Amikor ezt a displacement-et címképzésben használjuk, abban, mint egy 16 bites előjel nélküli egész szám vesz részt. A címzési mód byte-jait ebben az esetben így ábrázolhatjuk:



ahol mod = 10, DISP. LOW = a 16 bites előjel nélküli bináris szám alsó 8 helyiértékét, DISP. HIGH = a 16 bites előjel nélküli bináris szám felső 8 helyiértékét tartalmazza.

- **mod** = 11 regiszter **címzési mód**. Ilyenkor R/M specifikálja a konkrét regisztert attól függően, hogy az utasítás object-kódjának utasításrészét meghatározó byte LSB-je, azaz a W bit 1 ill. 0 értékű.

- **reg** ha **w = 1**, akkor egy 16 bites regisztert jelöl ki, ha viszont **w = 0**, akkor egy 8 bites regiszterre vonatkozik a cím:

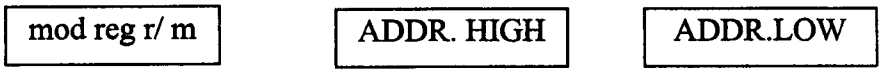
reg	w = 0	w = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Az **r/m** specifikálja a „mod" mezővel együtt a címzési módot (7.5. táblázat).

7.5. táblázat

r/m	mod=00	mod=01	mod=10	mod=11	
				w = 0	w = 1
000	BX + SI	BX + SI + DISP	BX + SI + DISP	AL	AX
001	BX + DI	BX + DI + DISP	BX + DI + DISP	CL	CX
010	BP + SI	BP + SI + DISP	BP + SI + DISP	DL	DX
011	BP + DI	BP + DI + DISP	BX + DI + DISP	BL	BX
100	SI	SI + DISP	SI + DISP	AH	SP
101	DI	DI + DISP	DI + DISP	CH	BP
110	DIRECT ADDR.	BP + DISP	BP + DISP	DH	SI
111	BX	BX + DISP	BX + DISP	BL	DI

A fenti táblázat maga helyett beszél, egyedül talán a „DIRECT ADDR." bejegyzés szorul magyarázatra. Nos: amikor **mod = 00**, és **r/m = 110**, akkor a címzési mód byte-ot követően nem displacement, hanem közvetlenül az offset cím található a következőképpen:

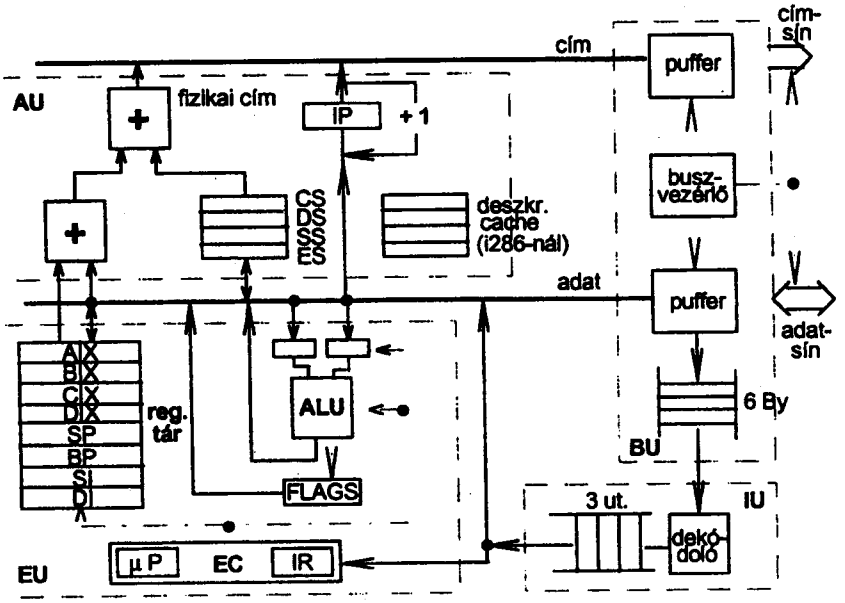


itt mod = 00, r/m = 110, ADDR.LOW az alacsonyabb 8-, ADDR.HIGH pedig a magasabb 8 helyiértékű címbitet tartalmazza.

7.3.2. Az i80286-os processzor

Az i80286-os processzor felépítését tekintve hasonló az i8086-oséhoz azzal a többlet különbséggel, hogy a kompatibilitás megőrzése végett kétféle üzemmódot alakítottak ki rajta. Az i80286-os processzor **valós,** ill. **védett üzemmódban** működhet. **Valós (real) üzemmód** esetén úgy működik, mint egy i8086-os processzor, azaz a címezhető memória tartománya csak 1 MB. **Védett (protected) üzemmód** esetén a címezhető fizikai tárolóterület a 24 vezetékes címsín miatt 16 MB, míg a virtuális memóriaterület 1 GB nagyságú. A védett üzemmód és a virtuális memóriakezelés csak az i80286-os ill. az azt követő processzorok jellemzője. Az 80286-os processzor fő egységeit a 7.18. ábra mutatja. A processzor négy fő funkcionális részből áll. A sínvezérlő egység (BU = busz unit) feladata a processzor és a memória vagy az I/O eszközök közötti kapcsolat kialakítása. Előkészíti a tárból az utasításokat, tárolja az adatokat, a szabad sínciklus alatt előkészíti és tárolja a következő utasítást egy 6 bájtos FIFO tárban. Az előkészített utasítást dekódolja a végrehajtáshoz. Az utasításfeldolgozó egység egyszerre 3 utasítást képes dekódolni és sorbaállítani a dekódoláshoz. A **végrehajtó egység** (EU = execution unit) a dekódolt utasításokban előírtakat hajtja végre a műveleti vezérlő (EC = execution control) irányítása alatt. A műveleti vezérlő az utasításregiszterben (IR) tárolt műveleti kód alapján elindítja a ROM-ban tárolt végrehajtó mikroprogramot. A

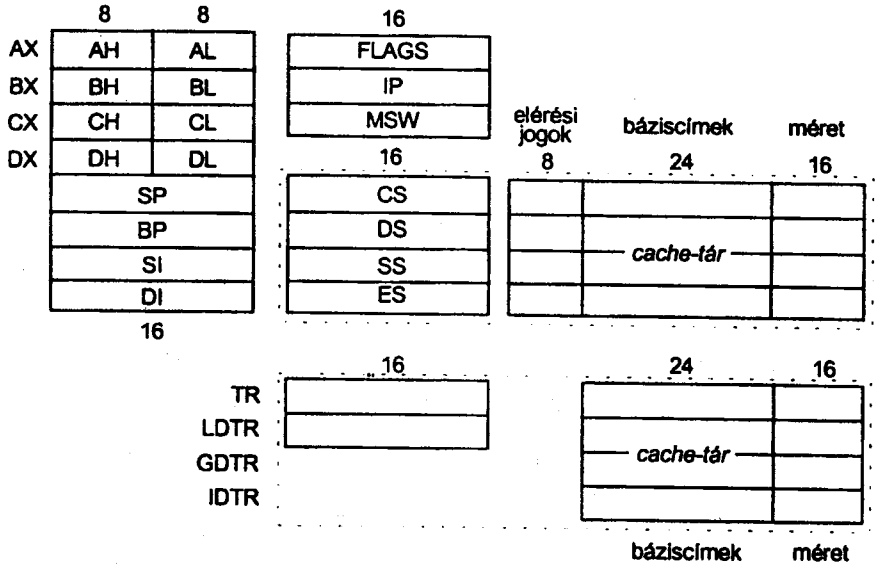
végrehajtó egység foglalja magában az ALU-t és a 8 regiszterből álló általános célú regisztertárat.



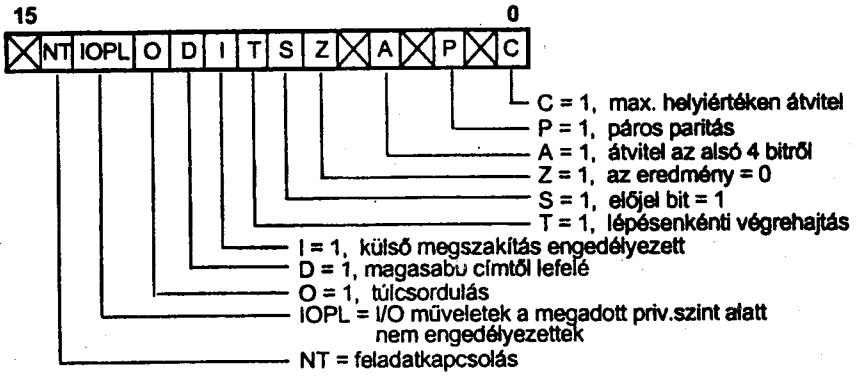
7.18. ábra

Az utasításban lévő operandus címeket a címkiszámító egység (AU = adressing unit) állítja elő és adja át a sínvezérlő egységnek íráshoz, olvasáshoz. A címkiszámító egység **valós üzemmódban 20 bites, védett üzemmódban 24 bites** címeket állít elő. A **virtuális tárkezeléshez** a szegmensszelektorokat tároló szegmensregiszterek mellett, a **szegmensleírók** (deskriptorok) tárolására egy kisméretű (4x48 bit) **cache tár** található. Az i80286-os regiszter készlete megegyezik a 8086 típus regisztereivel, a különbség annyi, hogy a védett üzemmód alkalmazásakor ezek mindegyikéhez tartozik egy - a felhasználó által nem elérhető - a szegmensleírót (deskriptort) tároló cache tár (7.19. ábra).

A 8086-os processzorhoz képest újdonság a **védett üzemmód mellett használt deszkriptortáblák szelektorait tároló TR (task segment register)** és **LDTR (local descriptor table register)** regiszterek. Ezek mellett magát a deszkriptort egy cache-tár tárolja. Az i80286-os processzor FLAGS regiszter tartalma két további bitet definiál. Az IOPL (I/O privilage level) bitek az I/O műveletek engedélyezett védettségi szintjét jelzik. Az NT (nested task) bit a beágyazott feladatot, feladatvégrehajtást jelzi. A processzor állapotát tükrözi vissza még a belső használatú MSW (machine status word) regiszter, amelynek bitjei a védett működési módot (protected mode enabled), a koprocesszorral összefüggésben a **taszkváltást** (TS = task switched), valamint a koprocesszor állapotát jelzik (7.20. ábra).

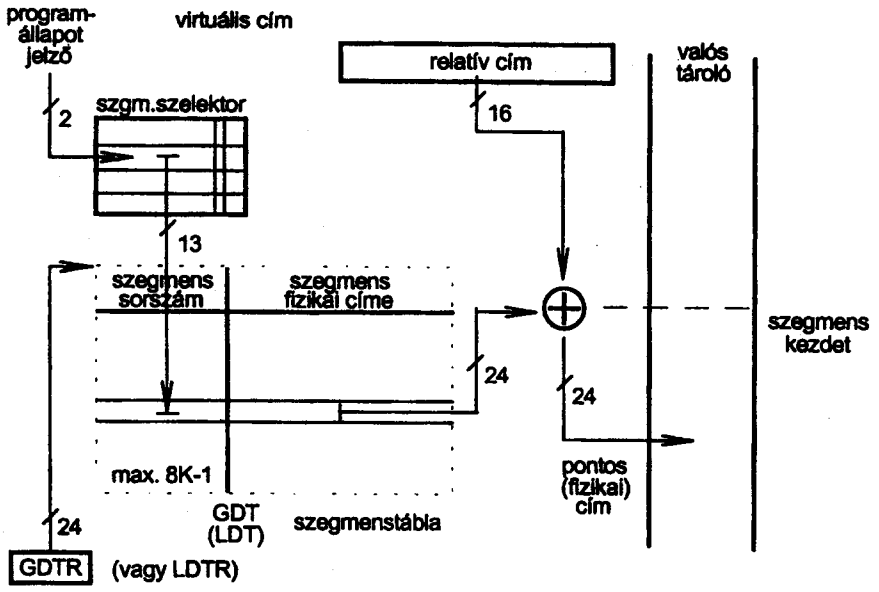


7.19. ábra



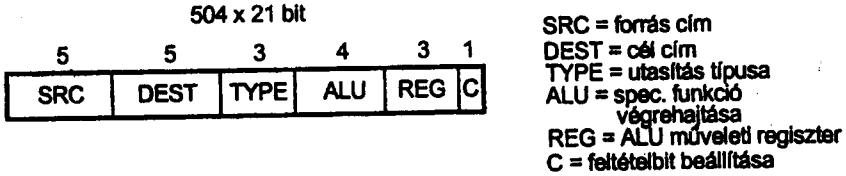
7.20. ábra

Az i286-os processzor által nyújtott új lehetőség a **virtuális címzés** használata, amelynek révén a közvetlenül címezhető memória tartomány mérete 1 GB nagyságú lehet. A védett, virtuális működési mód csak a 286-os processzortól kezdve került alkalmazásra. Az **így elérhető tárméret 1 GB**, ugyanakkor a **fizikailag kezelhető memória mérete a 24 db címvezeték révén 2^{24} = 16 MB** nagyságú. A virtuális cím méretét a szegmensszelektor felső 14 bitje és a 16 bites relatív cím alkotja. Az i286-os processzor esetében csak **szegmentált tárkezelésre van lehetőség** és egy-egy szegmens maximális mérete a relatív cím által meghatározott: 2^{16} = 64 KB méretű. A címkiszámítás módját a 7.21. ábra szemlélteti.



7.21. ábra

A **szegmens kezdőcímét egy deszkriptor, a szegmensleíró** tartalmazza, amely a **globális** (GDT) vagy **lokális** (LDT) **deszkriptortáblában** található meg. A használandó deszkriptortáblát a szegmensszelektor 2. bitje jelöli ki. Kikeresés után ez a deszkriptor ideiglenes jelleggel átkerül a szegmensszelektorok mellett található cache tárba. A deszkriptorból a GDT-ről és az LDT-ről az i386/486-os processzorok kapcsán teszünk említést. Az i286-os processzor gépi utasítások végrehajtásában egy **4 fokozatú átlapolódás (pipelining)** érvényesül. Ezek a fokozatok: **utasítások előkészítése, az utasítás dekódolása, a címkiszámítás és a művelet végrehajtása**, amelyek végrehajtására önálló egységek szolgálnak a processzoron belül. Az utasítások végrehajtásának vezérlésére mikroprogramozott vezérlő szolgál, amely az összes Intel processzornál közel azonos formájú. A mikroutasítás egy-egy elemi művelet végrehajtását eredményezi. A mikroutasítások felépítését szemlélteti a 7.22. ábra.



7.22. ábra

A TYPE mezőben megadható utasítás típusok:

- ALU művelet
- tároló művelet

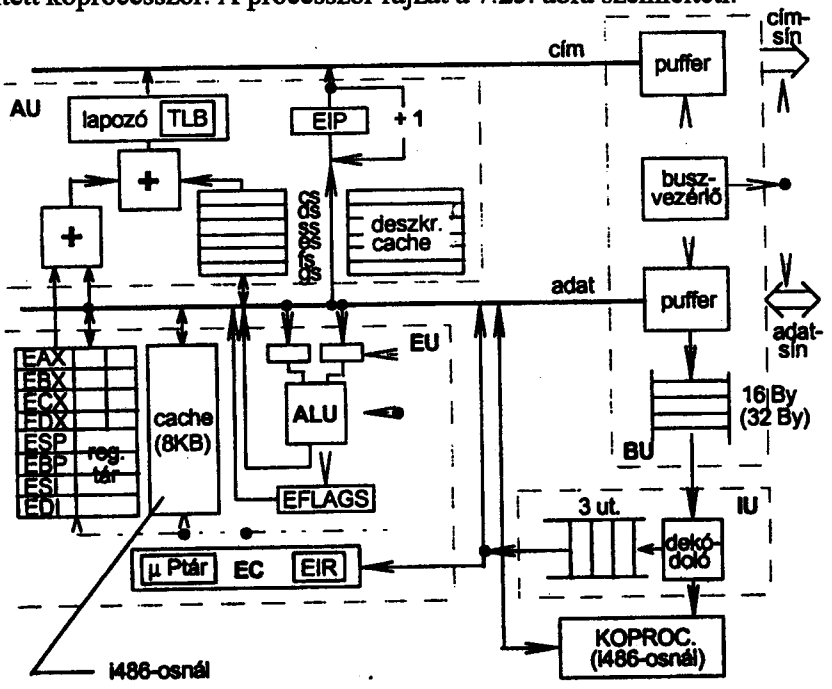
- rövid vagy hosszú ugrás
- mikroeljárás hívás, stb.

7.3.3. Az i80386/486-os processzorok

Az i80386-os és az i80486-os processzorok 32 bites eszközök. Belső felépítésük azonosnak mondható azzal a kiegészítéssel, hogy az i486-os processzor **beépített koprocesszorral** és egy **8 kB-os belső cache tárral** rendelkezik. A 32 bites címsínből adódóan a címezhető tártartomány 4 GB. Mindkét processzor a kompatibilitás megőrzése végett többféle üzemmódban képes dolgozni.

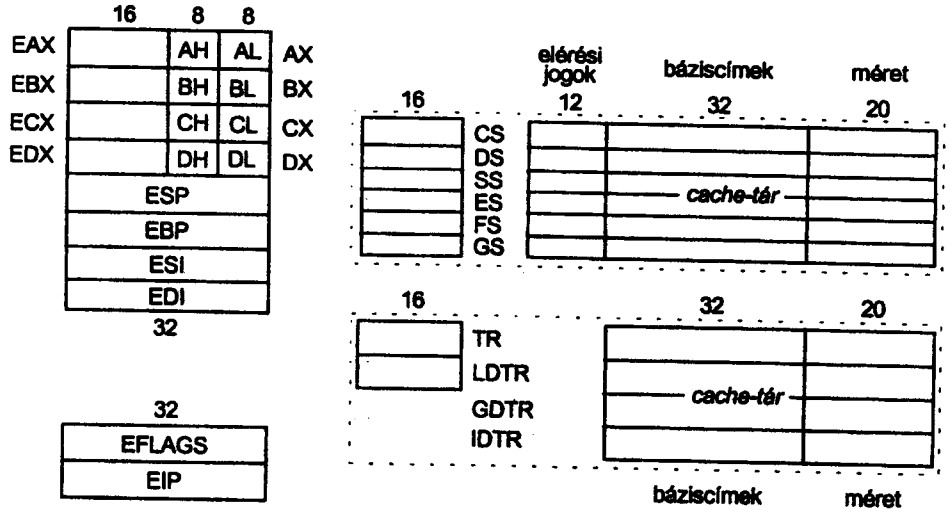
- **Valós (real) üzemmódban** a processzorok úgy működnek, mint egy 8086-os processzor.
- **Védett (protected) üzemmódban** a processzorok úgy működnek, mint az i286-os processzor.
- **16 bites, virtuális** i8086-os üzemmód esetén a védett üzemmódból eredően a 8086-oson futó program úgy kerül végrehajtásra a 386/486-os processzoron, mintha annak egy taszkja lenne.
- **32 bites üzemmód** a processzorok saját üzemmódja, azaz az összes erőforrás igénybevételével, **multitaszkos** feldolgozási lehetőséggel.

Az i386/486-os processzorok belső felépítése nagy mértékben hasonló az i286-oséhoz a következő többletekkel: memóriakezelés (lapozás), belső cache, a beépített koprocesszor. A processzor rajzát a 7.23. ábra szemlélteti.



7.23. ábra

A processzor regiszter modellje a 7.24. ábra szerinti.



7.24. ábra

Az i386/486 processzorok által használt **utasítások hossza 1-17 bájt** között változhat. A **virtuális működési módban szegmentált ill. szegmentált lapozásos tárkezelésre** van lehetőség. Az i386/486 típusú processzorok által biztosított védelem az alapja a **multitaszkos** (multiprogramozott) feldolgozásnak. **Multitaszkos feldolgozáskor** több feladat látszólag egyidejű (valójában időosztásos) végrehajtása történik. Ez azt jelenti, hogy a feladatok (taszkok) egymásnak adják át a processzor használatát (taszkváltás = task switching) az operációs rendszer által megszabott módon. A taszkváltást szegmensek közötti ugrás vagy eljáráshívás eredményezi, ha a **TSS deszkriptorra** hivatkozik. Egy-egy **taszkhíváskor az aktuális állapotot a taszk állapotszegmensbe** (TSS) menti a processzor.

A taszkváltás oka lehet:

- szegmensek közötti CALL, JMP vagy INT utasítás, amely a TSS deszkriptorra mutat
- szegmensek közötti CALL, JMP vagy INT utasítás amely taszk kapura mutat
- visszatérés a taszkból
- megszakítás.

A multitaszkos feldolgozáshoz az alábbi táblákat használja a processzor:

- GDT: globális deszkriptortábla
- LDT: lokális deszkriptortábla
- IDT: interrupt deszkriptortábla.

A 386/486-os processzorok további információit illetően az irodalomra hivatkozunk.

7.4. RISC processzorok

Az előző pontokban a 70-80-as években kifejlesztett processzorokat mutattuk be. Ezek közös jellemzői: **nagyszámú, bonyolult utasításokból álló utasításkészlet, sokféle címzési mód, mikroprogramozott vezérlő egység.** Az ilyen processzorokat **összetett utasításkészletű** (**CISC** = Complex Instruction Set Computer) processzoroknak nevezik. A teljesítőképesség növelése érdekében a korszerű technológiák adta lehetőségek kihasználásával a 80-as évektől az **utasításkészlet egyszerűsítésével** a processzorok architektúráját is egyszerűsíteni lehetett, amelynek következtében a **processzor teljesítőképessége is nőtt.** Ezeket a processzorokat nevezik **egyszerűsített utasításkészletű** (**RISC** = Reduced Instruction Set Computer) azaz **csökkentett utasításkészletű processzoroknak**. A RISC processzorok kifejlesztésének célja a **feldolgozási sebesség növelése.** Mielőtt a RISC processzorok legfontosabb jellemzőit összefoglaljuk, röviden áttekintjük a processzorok teljesítőképességének mérését ill. összetevőit. A processzorok működési sebességét két oldalról vizsgálhatjuk:

- **technikai oldalról** a sebességét egyrészt az áramkörök működési sebessége, azaz az alkalmazott technológia, másrészt a processzor struktúrája (memória elérés, utasítás végrehajtás, szervezés, stb.) határozza meg
- **felhasználói oldalról** a teljesítő képességet a feldolgozás időtartama határozza meg, azaz mennyi idő alatt lehet megoldani egy-egy feladatot.

Ezt az időtartamot három tényező határozza meg:

$$\mathbf{Tu = C.T.I\ (idő/feladat)} \qquad (7\text{-}1)$$

ahol **C** (cycle per instruction) a **gépi utasításonkénti ciklusok száma, T** (time per cycle) az **órajel ciklusok időtartama** (idő/ciklus), amely az órajel frekvenciájának reciproka (T = 1/f). Ez a mai processzoroknál 10 ns körüli érték (100 MHz-es órajel esetén). I (instruction per task) **feladatonkénti utasítások száma** (utasítás/feladat). A korábbi elemzések szerint a teljesítmény jellemzésére az utasítás-végrehajtás időigénye alapján számított **időegységenként végrehajtott utasítások számát adják meg MIPS** (millions of instruction per second) egységben:

$$\mathbf{P} = \frac{1}{\mathbf{C} \cdot \mathbf{T}} = \frac{\mathbf{f}}{\mathbf{C}}. \qquad (7\text{-}2)$$

A fenti képletből látható milyen jelentős szerepe van **teljesítőképesség (P)** szempontjából, hogy hány óraciklusonként fejeződik be egy utasítás végrehajtása.

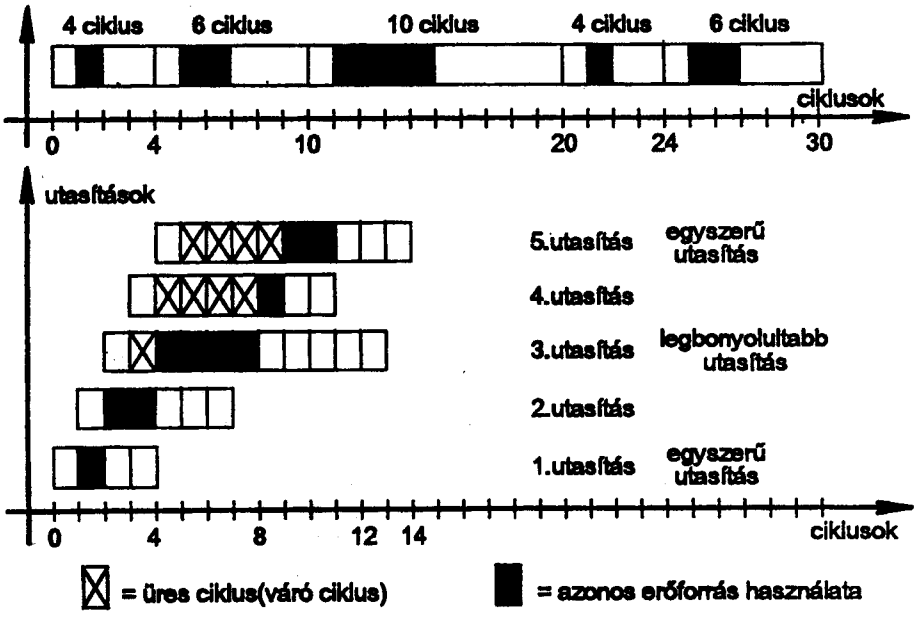
A C ciklusszám csökkentésének módszerei:

a) **utasítások átlapolt** végrehajtása (pipelining)
b) **LOAD/STORE** struktúra alkalmazása
c) **késleltetett LOAD utasítás**
d) **késleltetett elágazás-feldolgozás.**

a) Az **utasítás-végrehajtásonkénti ciklusszám** növelésének egyik hatásos módja az **utasítások átlapolásos** feldolgozása. Ha az **utasítás-feldolgozás folyamatát közel azonos időtartamú részfeladatokra** bontjuk **(utasítás-előkészítés, dekódolás, operandus előkészítés, művelet végrehajtás, az eredmény tárolása),** akkor az egyes fázisokhoz kapcsolódó erőforrások egy-egy utasítás-végrehajtás alatt csak rövidebb ideig vannak lekötve, miáltal a **következő utasítás előbb hozzáférhet ugyanahhoz az erőforráshoz,** tehát az egyes utasítások végrehajtása átlapolható. Az átlapolt feldolgozás miatt megoldható a **végrehajtás egyenletes ütemezése,** amely az azonos végrehajtási idejű utasításoknál könnyen megoldható. Ha az utasítás-végrehajtás folyamata n részfeladatra van felbontva, akkor a pipeline feldolgozásnál ideális esetben

$$C_{ppl} = \frac{C}{n} = 1 \text{ (ciklus/utasítás).}$$

A gyakorlatban az effektív ciklusszám ennél nagyobb. A soros ill. az átlapolt utasítás-végrehajtás folyamatát szemlélteti a 7.25. ábra.



7.25. ábra

Az **átlapolt utasítás-végrehajtás** tulajdonképpen az egyprocesszoros gépen futó **ún. belső párhuzamosítás.** A hatékonyság növelésének további lehetősége a **többprocesszoros architektúra** alkalmazása (transzpúteres ill. szuperskalár feldolgozás). Az utasítás pipeline módszerrel a RISC processzoroknál találkozunk.

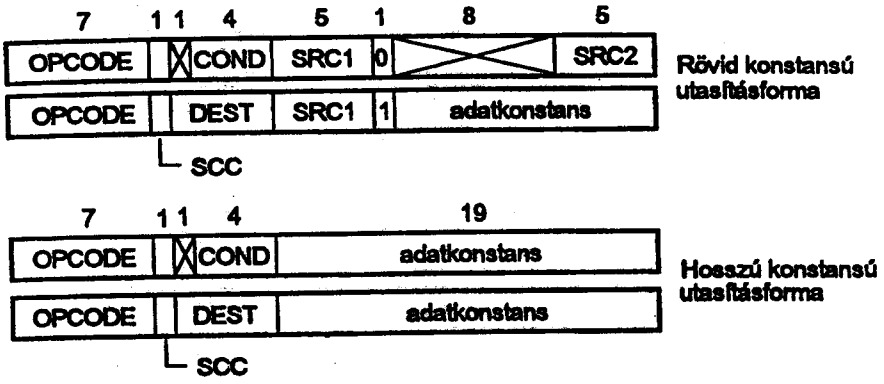
b) LOAD/STORE struktúra

A fentiekben vázolt átlapolásos végrehajtás **csak azonos idejű végrehajtási fázisok esetén lehet hatékony**. Ha valamelyik fázis időigénye nagyobb, akkor vagy fel kell függeszteni az átlapolásos működést **váró ciklussal** vagy az egyes **ütemek hosszát kell megnövelni.** Ez természetesen növeli a C értékét. Mivel a tárműveletek időtartama hosszabb, mint a belső CPU műveleteké (a memória működési sebessége alacsonyabb mint a processzoré, az adatátvitel lassúbb), ezért a **hatékonyság növelése érdekében a memória utasítások számát csökkenteni célszerű.** Ennek megvalósítására dolgozták ki a **LOAD/STORE** struktúrát, amelynek jellemzője a **nagyszámú, processzoron belüli általános célú regiszter,** valamint a tárkezelésre szolgáló két utasítás: LOAD és STORE. Azonban a LOAD és STORE utasítás időtartama hosszabb, mint a processzor műveleteké. Ilymódon a LOAD utasítás által beolvasott adat nem áll megfelelő időben rendelkezésre a soron következő utasításhoz. Ezt vagy **váró ciklus** beiktatásával, vagy **utasítás áthelyezéssel** lehet megoldani. Utóbbi esetben a LOAD utasítástól nem függő utasítást iktatnak be (áthelyeznek) közvetlenül a LOAD utasítás utáni késleltetési fázisba. Ez az ún. **késeltetett LOAD/STORE** módszer, amelyeknél a késleltetési fázis optimális kihasználása a fordítóprogram feladata. Az **utasításpipeline** ütemezésében a LOAD/STORE utasításhoz hasonló problémát vet fel a **feltételes vezérlő átadások** végrehajtása. Itt a **késleltetést az elágazás feltétele teljesülésének megállapítása jelenti.** Ezt a fordítóprogram úgy oldja meg, hogy a vezérlésátadó utasítást egy vagy két **utasítással „előre" veszi.**

7.4.1. RISC processzorok utasításkészlete

A RISCI. és RISCII. processzorok utasításszerkezetét a 7.26. ábra szemlélteti.

Az **utasítások 32 bit hosszúságúak** és az **első hét bit tartalmazza a műveleti kódot.** A lehetséges 127 utasításból csak 31 ill. 39 utasítás került kihasználásra. A SCC bit jelzi a feltételbit beállításának szükségességét. A DEST mező az **eredmény regiszter címét** tartalmazza, vagy feltételes ugró utasítások esetén a feltétel kódját (COND). Az utasítás további részén vagy **két regisztercím** (SRC1, SRC2) vagy **egy regisztercím és egy adatkonstans** helyezhető el.



7.26. ábra

A RISC processzorok utasításkészlete a műveletek alapján három csoportba sorolható:

- **CPU-memória közötti adatmozgató utasítások** (LOAD/STORE)
- **aritmetikai-logikai utasítások**
- **vezérlésátadó utasítások.**

A **CPU - memória (I/O)** közötti **adatátviteli** utasítások a **tárolóba írás** (STORE) és a **tárolóolvasási** (LOAD) műveletre korlátozódnak. Ezek az utasítások két vagy több bájt átvitelére szolgálnak. Az **aritmetikai-logikai utasítások** rendszerint egy vagy két operandussal dolgoznak. Az operandusok mindig a regiszter tárban találhatók. Az ALU utasítások három csoportba sorolhatók:

- aritmetikai műveletek
- logikai műveletek (AND, OR, EOR)
- léptető (Shift) műveletek.

A RISC processzorok többnyire **nem rendelkeznek összetett műveleti utasításokkal** (pl. fixpontos szorzás, osztás). A **lebegőpontos műveleteket** általában **külön aritmetikai processzor végzi.** Az aritmetikai műveletek végrehajtásakor az eredménytől függően az állapot (flag) regiszter egyes bitjeit a processzor beállítja. A flag-ek tartalmától függően van lehetőség a **program feltételes elágaztatására. A logikai műveletek bitenként értendők. A léptető utasítások** az operandusok bitjeit **jobbra** vagy **balra egy vagy több helyiértékkel** eltolják. A program futását befolyásoló vezérlésátadó utasítások kétfélék lehetnek: **feltétel nélküli** vezérlésátadó utasítások ill. a **feltételes** vezérlésátadó utasítás. A processzor működését befolyásoló utasítások pl. a megszakítást engedélyező és tiltó utasítások ill. a processzor működését megszakító, leállító utasítás (BREAK, HALT). A RISC processzorok vezérlő egysége PLA-val (programozható logikai tömb) megoldott **huzalozott** vezérlés. A RISC processzorok esetében a **magas szintű programozási** elven

elkészített felhasználói programot egy **egyszerű utasításokból álló gépi kódra fordítják le.** Ezek az utasítások nem igénylik a mikroutasításokkal megvalósított értelmező rendszert, mert utasításai közvetlenül végrehajthatók. A RISC processzorok gépi kódú programozása nagyon közel van a CISC processzorok mikroprogramozásához. Emiatt a RISC processzorokat nem könnyű gépi kódban programozni, azt fordítóprogramra kell bízni.

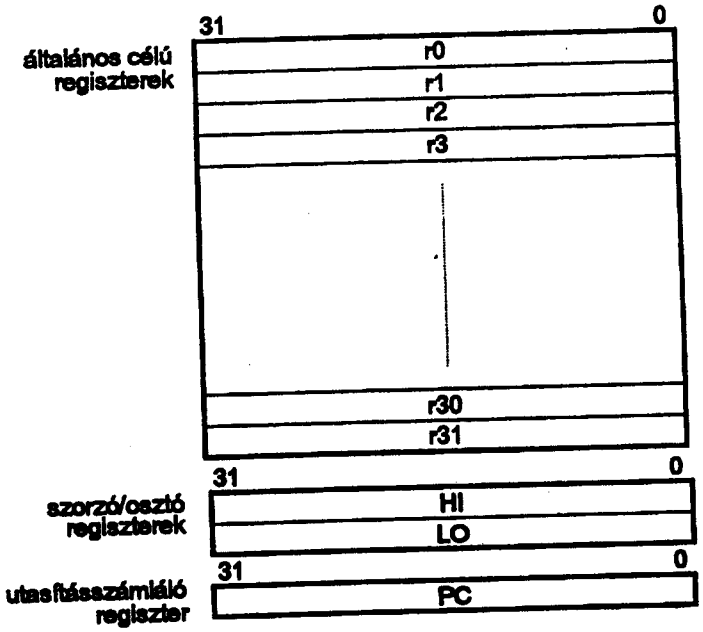
7.4.2. MIPS processzorok

A **MIPS** (Microprocessor without Interlocked Pipeline Stages) processzorokat a Stanford University-n fejlesztették ki. A tervezők elsősorban a **hardver gyorsaságát kívánták** biztosítani és ennek érdekében a feladatok ritkábban szükséges részét a szoftverre bízták.

A MIPS processzorok jellemzői:

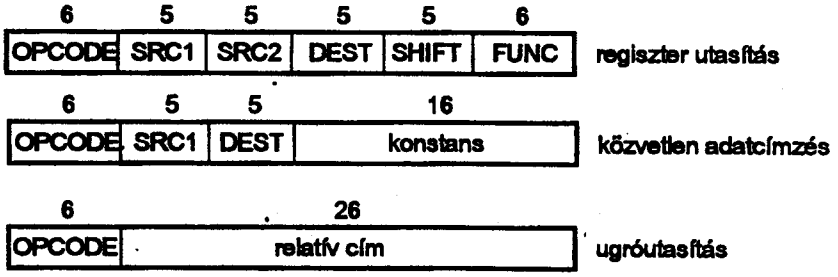
- A processzorok **32 bitesek, 32 darab 32 bites általános célú regiszterrel.**
- A **memória kezelést az MMU** végzi, **nagykapacitású cache** memóriák beiktatásával.
- A **processzorok pipeline feldolgozást** alkalmaznak.
- A processzorokhoz a beépített vezérlő processzoron kívül **három koprocesszor** csatlakozik.

Az **alapprocesszor regiszter készlete** a 7.27. ábra szerinti.



7.27. ábra

Az alapprocesszor mellett a **rendszervezérlő koprocesszor** és a **lebegőpontos koprocesszor** végzi az utasítások végrehajtását. A MIPS **processzorok mindössze három utasításformát** ismernek, amelynek szerkezete a 7.28. ábra szerinti.



7.28. ábra

Az **utasításformátum első 6 bitje a műveleti kód,** amely összesen **64 utasítás** kialakítását teszi lehetővé. A legalacsonyabb 6 biten (FUNC) **speciális funkciók** jelölhetők ki. Az **első utasítás egy háromcímes szerkezet,** míg a **második egy kétcímes konstanssal kiegészített utasítás.** A **harmadik utasítás vezérlésátadó utasítás.** A SHIFT bitek (5) az eltolási utasításhoz adják meg az eltolás mértékét.

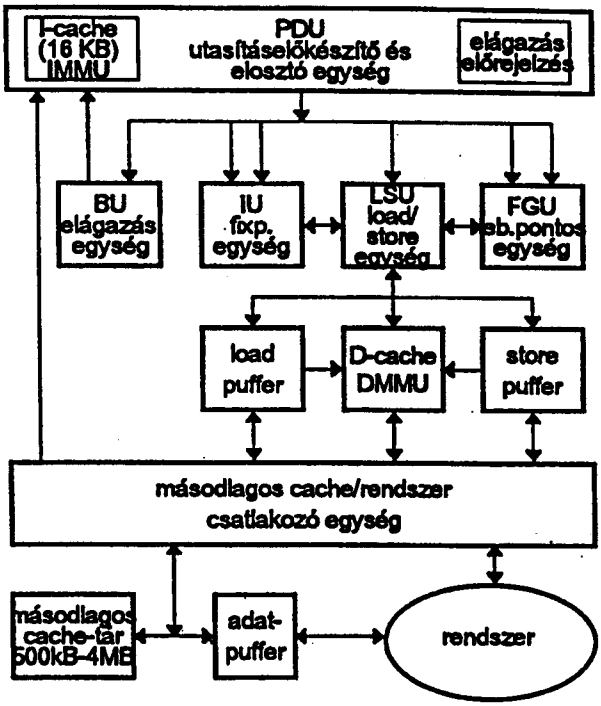
7.4.3. SPARC processzorok

A SPARC (Scalable Processor Architecture) processzorok kifejlesztője a 80-as évek közepén a SUN Microsystems volt. 1992-ben jelent meg az Ultra SPARC processzor.

A SPARC processzorok jellemzői:

- a **processzor 64 bites,** mind a **címvonalak,** mind az **adatutak**
- a **processzor által kezelhető tárméret 2^{64} bájt**
- a processzor **utasításkészlete egyszerű felépítésű, kevés címzési lehetőséggel**
- **pipeline adatfeldolgozás.**

A SPARC processzorok két lényegi része az **egész típusú mennyiségek kezelésére szolgáló „Integer Unit"** (IU) és a **lebegőpontos adatok feldolgozására** szolgáló „Floating Point Unit" (FPU). Az egységekhez saját regiszterek tartoznak. Az egészekhez tartozó regiszterek 64 bitesek, a lebegőpontos regiszterek pedig 32, 64 vagy 128 bit szélességűek. Az ULTRA SPARC processzor felépítését a 7.29. ábra mutatja. A processzor kétféle üzemmódban működhet: **privilagizált** és **nem privilagizált** üzemmódban. Előbbi üzemmódban a processzor minden utasítást végre tud hajtani, utóbbi esetén a privilagizált utasításokat nem.



7.29. ábra

7.4.4. A CISC és RISC processzorok összehasonlítása

a) a CISC processzorok utasításkészletének főbb jellemzői

- Az utasítások bonyolult műveletsor végrehajtását eredményezik és ehhez **több gépi ciklust** használnak fel.
- **Sokféle utasítást** (100 - 200) és **címzési módot** (8-20) használnak.
- **Sokféle, memóriát közvetlenül igénybevevő,** megcímző utasítás használati lehetősége.
- **Mikroprogramvezérelt** utasítás-végrehajtást használnak.
- Az **utasítások változó hosszúságúak.**
- A CISC processzorok komplexebb utasításai révén **rövidebb programokat** eredményeznek.
- A CISC processzorokhoz készített **fordítóprogramok egyszerűbbek.**

b) A RISC processzorok utasításkészletének főbb jellemzői

- Kevéssé bonyolult utasítások: LOAD/STORE, műveleti és vezérlésátadó utasítások.
- **Kevés utasítás** (~ 50) és címzési mód (2 - 4) használata.
- Az **utasítások rögzített hosszúságúak.**
- Memória kezelésre csak a **LOAD/STORE** utasítások használatosak.

- Az utasítások végrehajtásakor **csak egy gépi ciklust** használnak fel utasításonként.
- Az **utasítások végrehajtását nem mikroprogram** végzi, hanem **huzalozott vezérlés.**
- **Pipeline azaz átfedéses** utasítás feldolgozás gyorsítja a végrehajtást.

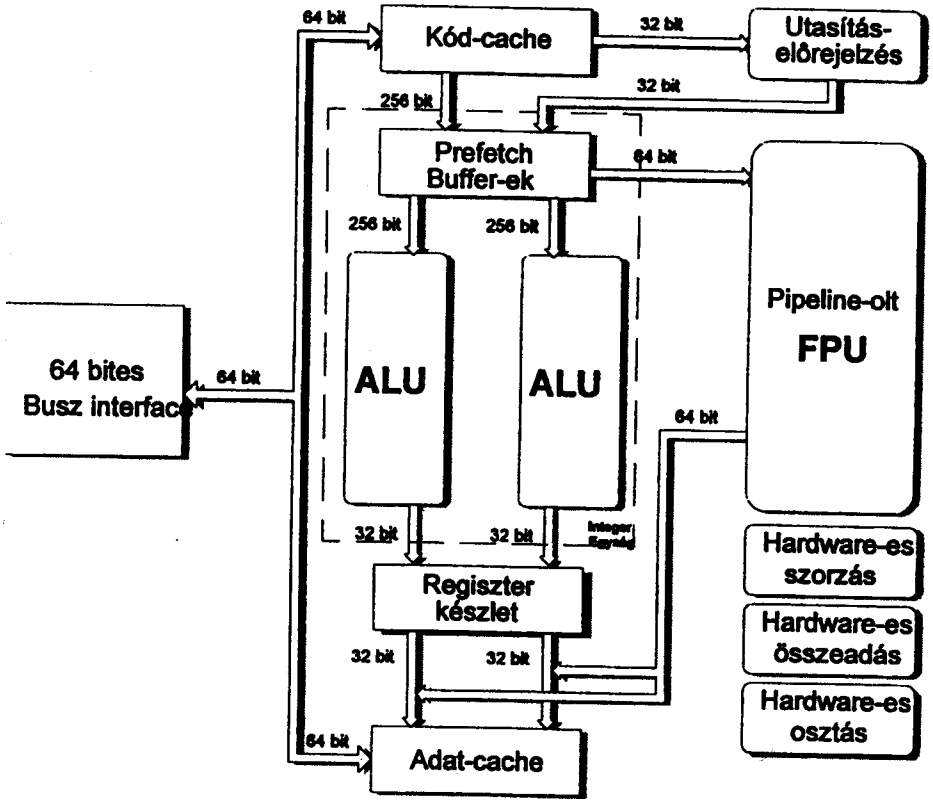
c) CISC és RISC processzorok összehasonlítása

7.6. táblázat

	CISC processzorok	RISC processzorok
1	Összetett utasítások, melyek végrehajtása több gépi ciklust igényel	Egyszerű utasítások, melyek végrehajtása 1 gépi ciklust igényel
2	Bármely, erre alkalmas utasítás igénybe veheti a tárolót	Csak a LOAD/STORE utasítások fordulhatnak a memóriához
3	Az adatcsatornás (pipelining) feldolgozás kismértékű	Erőteljes adatcsatornás (pipelining) feldolgozás
4	Utasításvégrehajtás mikroprogram által vezérelt	Huzalozott utasításvégrehajtás
5	Változó hosszúságú utasítások	Rögzített utasításhossz
6	Sokféle utasítás és címzési mód	Kevés utasítás és címzési mód
7	Bonyolult mikroprogram	Bonyolult fordítóprogram
8	Kis számú regiszter	Nagy méretű regisztertár

7.5. A Pentium processzor

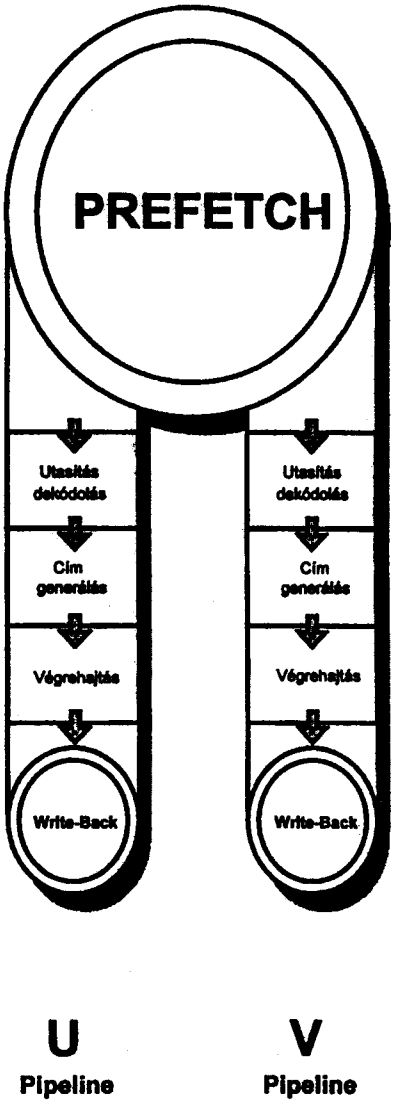
Az Intel Pentium processzora (80502, 80503) az x86-os architektúrákkal lefelé kompatibilis. A Pentium processzor blokkvázlata a 7.30. ábrán látható.
A processzor ún. **szuperskalár** architektúrával épül fel, ami azt jelenti, hogy **egyetlen óraciklus alatt egynél több utasítást** tud végrehajtani. Beépített lebegőpontos aritmetikai egységgel, a processzorlapkára integrált külön kód- és write-back adat-cache-sel, 64 bites külső adatsínnel, az újabb 90-100 MHz-es típusok Multi Processor (MP) vezérlővel rendelkeznek. A Pentium processzor **két beépített** és egymástól függetlenül működő **pipeline funkcióval** működik. Ezek a pipeline-ok teszik képessé a Pentium processzort, hogy **egy óraciklus alatt két integer utasítást** végrehajtson. Ez közel kétszeres chip sebességet jelent az azonos frekvencián működő I486 chiphez képest. A Pentium processzor pipeline-jai azonosak az Intel 486 CPU egyszerű pipeline-jával, de magasabb sebességre optimalizálták őket. Mindegyik **pipeline öt utasítás végrehajtási szintre** tagozódik. Ezek: **prefetch, utasítás dekódolás, címgenerálás, végrehajtás, write-back.** Az utasítások előbetöltése (Instruction Prefetch): az utasítások kódjának előzetes beolvasása a processzorba. Amikor egy utasítás a prefetch-ből az utasításdekóderbe kerül a pipeline kész egy másik, új utasítás végrehajtásának elkezdésére.



7.30. ábra

A Pentium processzor két utasítást is el tud végezni egyszerre - egy-egy utasítást mindegyik pipeline-ban - ezt a módszert **utasítás párosításnak** nevezik (7.31. ábra).

A párosítás feltétele, hogy mindkét utasításnak egyszerűnek kell lennie. Mindegyik **pipeline saját ALU-val, címregeneráló egységgel és adatcache interfésszel** rendelkezik. A cache egységek 8 kBájt kapacitásúak és az egyik csak **adatokat,** a másik csak **utasításokat** tárol. Az előbb említett cache-ok **kétutas csoport-asszociatív** szervezésűek. A csoport-asszociatív szervezés a cache felépítés olyan formája, amely egy adattömb helyét a főmemóriában leszűkíti ugyan, de ennek helye a cach-ben nincs teljes mértékben meghatározva. Az adatcache-nek két interfésze van: mindkét pipeline-hoz egy-egy, amely biztosítja, hogy egy óraciklusón belül két különböző művelet is fel tudja használni ugyanazt az adatot.



7.31. ábra

Ha az adat már éppen kikerül az adatcache-ből, a processzor visszaírja azt a fő memóriába. Ez a technika **write-back cache** néven ismert. Abban az esetben, ha a processzor az adatot egyidejűleg a főmemóriába és a cache-be is beírja, akkor **write through cache** kezelésről beszélünk.
A Pentium processzor támogatja a write-through kezelést. A cache kezelés az ún. **MESI** (Modified, Exclusive, Shared, Invalid) protokoll szabványként ismert. A szabvány 4 állapotot definiál, amelyek mindegyike a cache vonalához van rendelve. A MESI protokoll támogatja a **multiprocesszoros** működést.

A 4 állapot: **módosított, kizárólagos, megosztott, érvénytelen.** Sebességnövelő technikák a Pentium processzornál:

- szuperskalár architektúra
- szétválasztott cache-ek
- write/back cache kezelés
- gyors TPU
- 64 bites adatsín memóriák felé
- cache kezelés burst módban
- a MOVE és néhány ALU utasítás hardver megoldású, ami gyorsabb működést biztosít.

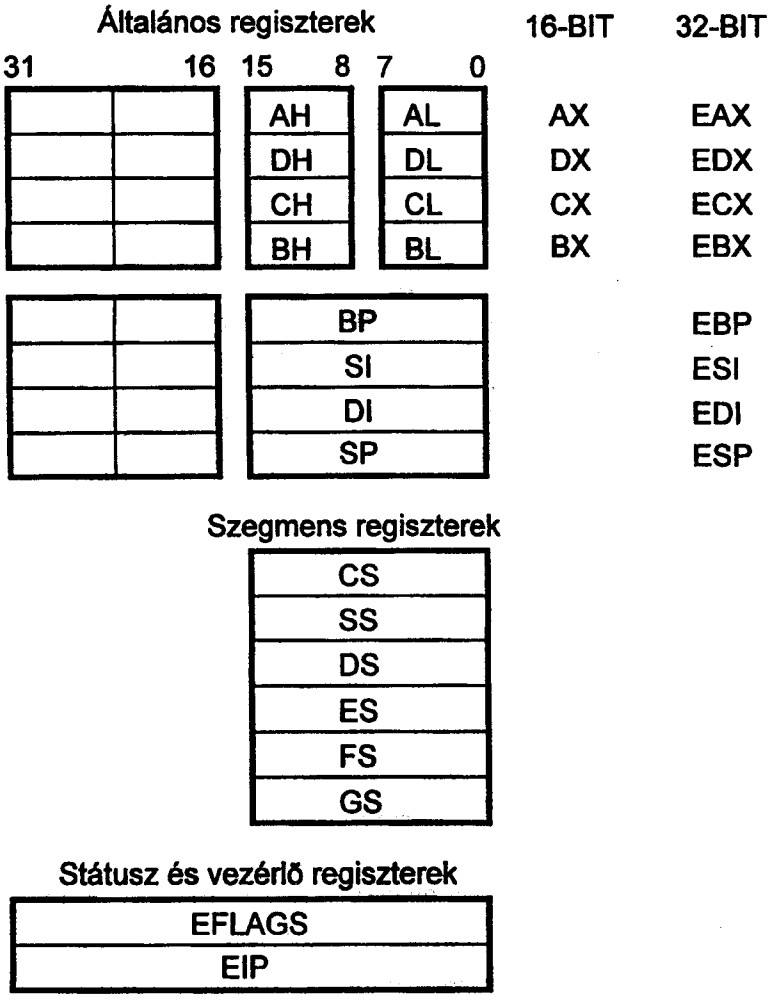
A Pentium processzor architektúrája támogatja a multiprocesszoros architektúrát azáltal, hogy a chip tartalmaz egy multiprocesszoros vezérlő áramkört. Ez a vezérlő két részre oszlik: a **programozható megszakítás-vezérlőre** (APIC = Advanced Programmable Interrupt Controller) és az **arbitrátorra.** Ezen vezérlő segítségével olyan szerver architektúrák is megvalósíthatók, amelyek 256 processzort tartalmaznak, mindegyiket saját cache-sel. A Pentium processzor különböző **adatvédelmi** technikát alkalmaz. A **hibadetektálás** kétszintű: **paritásvizsgálat** a lábakon és a **logikára integrált memória struktúrákon** (cache, bufferek, ROM). Ugyanakkor a Pentium rendelkezik egy funkcionális redundancia ellenőrzési funkcióval (Functional Redundancy Checking = FRC). Az FRC két Pentium chipet igényel: az egyik a **master**, a másik a **checker.** A chip-ek párhuzamosan futnak és a checker összehasonlítja az általa kapott eredményeket a master eredményeivel, hogy egészen biztosan hibamentesek legyenek az eredmények. Az FRC alkalmazásával a hibadetektálás hatásfoka 99 %. A processzor tartalmaz néhány teszt lehetőséget a chip megbízhatóságára vonatkozóan. Az **önteszt** minden RESET esetén lefut és a processzor 70 %-át ellenőrzi az IEEE 1149.1 szabvány alapján (Boundary Scan lásd később).

A memória felépítése

A Pentium processzor sínrendszerére kapcsolt memória a **fizikai memória,** amely bájtos szervezésű. A címezhető fizikai címtartomány 2^{32} - 1 (4 Gbájt) terjedelmű. A memória kezelést a hardver végzi. A memória kezelés használata esetében a programok nem érik el közvetlenül a fizikai memóriát, hanem egy memória modellt címeznek, melynek **virtuális memória** a neve. A memória kezelése szegmentálásból és lapozásból áll. Egy program által használt címek a **logikai címek.** A szegmentálást végző hardver a logikai címet a nemszegmentált címtartomány címzéséhez szükséges lineáris címmé, a lapozást végző hardver pedig fizikai címmé alakítja át.

Regiszterek

A Pentium processzornak 16, a programozó által használt regisztere van. (7.32. ábra).



7.32. ábra

A regiszterek csoportosítása

a) **általános célú regiszterek** (8 darab 32 bites), melyeket a programozó szabadon használhat

b) **szegmens regiszterek** (6 db) a különböző memória szegmenseket tartalmazzák

c) **állapot és vezérlő regiszterek**, amelyek a processzor állapotáról adnak tájékoztatást.

a) Általános célú regiszterek

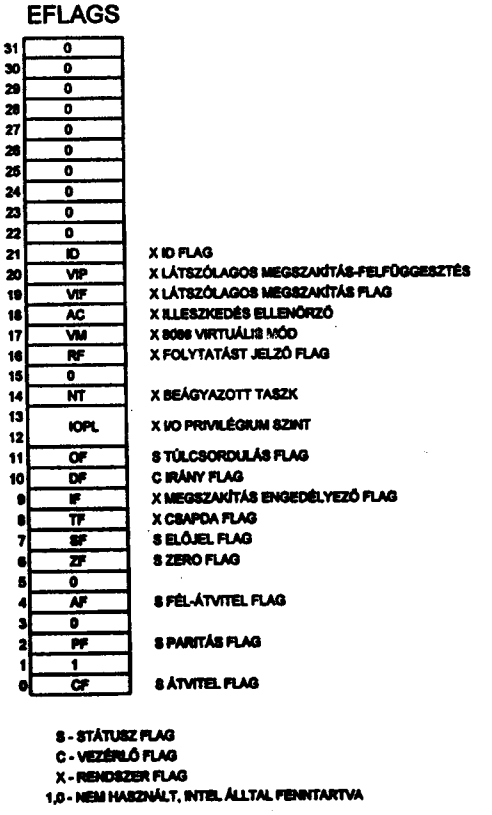
Az EAX, EBX, ECX, EDX, EBP, ESP, ESI, EDI 32 bites regiszterek. Az említett regiszterek a 8086 processzor AX, BX, CX, DX, BP, SP, SI, DI regiszterek neveiből származnak. Ezekkel a nevekkel lehet hivatkozni a 32 bites regiszterek alsó 16 bitjére. Az AX, BX, CX, DX 16 bites regiszterek mindkét bájtjának saját neve is van: AH, BH, CH, DH (felső bájtok) ill. AL, BL, CL és DL (alsó bájtok).

b) Szegmens regiszterek

A Pentium 6 szegmens regisztert tartalmaz. **Kódszegmens regiszter** (CS) a program memória szegmens címét tartalmazza. A processzor a kódszegmensben lévő utasításokat olvassa be, a szegmensen belüli ofszetként az EIP regiszter tartalmát felhasználva. A DS, ES, FS, GS regiszterek **adatszegmenseket** definiálnak. Az SS a **stack szegmens**, amely ofszetként az SP-t (ESP-t) használja. A verem nagysága max. 4 Gbájt lehet. Az, amelynek szektorát az SS regiszter tartalmazza az aktuális verem.

c) Flagregiszter

A feltétel kódokat a 32 bites flagregiszter tartalmazza (EFLAGS). A flagek elrendezését a regiszteren belül a 7.33. ábra mutatja.



7.33. ábra

Feltétel bitek

OF -túlcsordulás bit
SF - előjel bit
ZF - zérus bit
AF - félátvitel bit
PF - paritás bit
CF - átvitel bit
DF - irány bit a sztring műveletekhez.

A Pentium processzor további ismereteit illetően az irodalomra utalunk.

7.6. Mikrovezérlők

A mikroprocesszorok jellegzetes csoportját a **mikrovezérlők** (microcontrollers) alkotják. A mikrovezérlők elődjének az egychipes mikroszámítógépek tekinthetők (i8048, 8051 családok, stb.). Az egychipes mikroszámítógépek a mikroprocesszoron túlmenően a RAM, EPROM memóriákat ill. I/O felületet is egy chipen biztosítják, ezáltal egy-egy kisebb feladatot önmagukban képesek megoldani. A mikrovezérlők a méréstechnikai, irányítástechnikai, telekommunikációs feladatok egyszerű és biztonságos realizálásához fejlesztették ki. Napjainkban igen nagy a gyártók közötti verseny a mikrovezérlők fejlesztése terén.

A mai mikrovezérlők főbb funkcionális jellemzői:

- RISC jellegű utasításkészlet
- stand-by jellegű üzemeltetési lehetőség
- speciális törlési, indítási, watch-dog logika
- speciális megszakítási lehetőségek
- FLASH-EPROM memória alkalmazása
- speciális I/O egységek alkalmazása
- A/D ill. D/A konverterek
- fájl jellegű regiszterkezelés.

A nagyszámú mikrovezérlőről nagyon nehéz általános ismereteket közreadni. A következőkben a Microchip Technology Inc. (USA) által kifejlesztett PIC családot ismertetjük vázlatosan. A részletes információkat illetően a vonatkozó katalógusokra ill. laboratóriumi gyakorlatokra utalunk.

7.6.1. A PIC mikrovezérlő család

A PIC mikrovezérlők CMOS technológiával készült és RISC jellegű, könnyen megtanulható utasításkészlettel ellátott eszközök. További jellemzőjük, hogy **csak a belső RAM memóriát** képesek kezelni. A PIC mikrovezérlők 8 bites áramkörök, a CPU és a belső adatsín is 8 bites, de az utasítások 12 ill. 14 bitesek. Így érték el, hogy az utasításkészlet minden utasítása egy szavas, ami a működési sebesség szempontjából igen fontos.

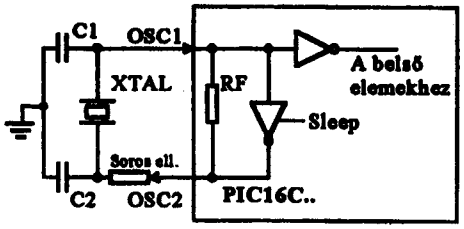
7.6.1.1. A PIC16C mikrovezérlő család

A PIC165X család **utasításkészlete 33 utasításból** áll, a végrehajtásuk általában **egy gépi ciklust** igényel. Az utasítások 12 ill. 14 bit hosszúságúak. Mint a mikroprocesszorok többsége, a PIC áramkörök és az ún. Harward architektúrát követi, azaz különálló program ill. adatmemóriát tartalmaznak. Ez teszi lehetővé, hogy az EPROM 12 ill. 14 bites, a RAM pedig 8 bites szervezésű. Az EPROM kapacitása széles tartományból választható (0,5 k...4 k szó) és a belső adattár kapacitása is változó. (25...192 bájt). Mivel külső memória áramkörök nélkül dolgozik, így a stack mérete is korlátozott. A különböző PIC típusokban 1...4 db **párhuzamos port** található és minden típusban van **időzítő/számláló** egység is, utóbbi a valós idő mérésére, események számlálására alkalmas ill. a **watch-dog** timer kialakítására. A watch-dog („őrző kutya") áramkör a program futását ellenőrző hardver, melynek működése azon alapszik, hogy a programfejlesztő a programban olyan funkciókat épít be, amelyek meghatározott időnként jelezve „ébren" tartják a watch-dog áramkört. Amennyiben a program elszáll, ezek a jelzések elmaradnak és a watch-dog áramkör beavatkozik (pl. leállítja a processzort). A család A/D konverterrel ill. soros porttal ellátott változatot is tartalmaz. A PIC mikrovezérlők névleges tápárama 2 mA (4 MHz órafrekvencia mellett), de 3 V-os tápfeszültség és 32 KHz-es órafrekvencia mellett tápáram igénye kisebb, mint 1 μA. A PIC mikrovezérlők a RESET áramköri megoldások, az oszcillátor kialakításának módja tekintetében és az utasításkészlet szempontjából egységesek. A PIC mikrovezérlők minden típusa többféle oszcillátorral is működhet. Az oszcillátor típust az EPROM-ban kialakított biztosítékkal lehet kiválasztani. A kvarcablakos tokozatú (EPROM) változatok és az EEPROM alapú PIC-ek esetén a felhasználó a beprogramozással egyidejűleg a fejlesztő egység segítségével állítja be az oszcillátor típusát. Az egyszer programozható (OTP: On-time Programmable) mikrovezérlőket zárt műanyag tokozással gyártják, ezeknél az oszcillátor típusát a gyártó már beállította a megrendelés alapján.

A négy oszcillátor lehetőség a következő:

- LP: kisteljesítményű kvarcoszcillátor
- XT: kvarcoszcillátor
- HS: nagysebességű kvarcoszcillátor
- RC: R-C hangolású oszcillátor.

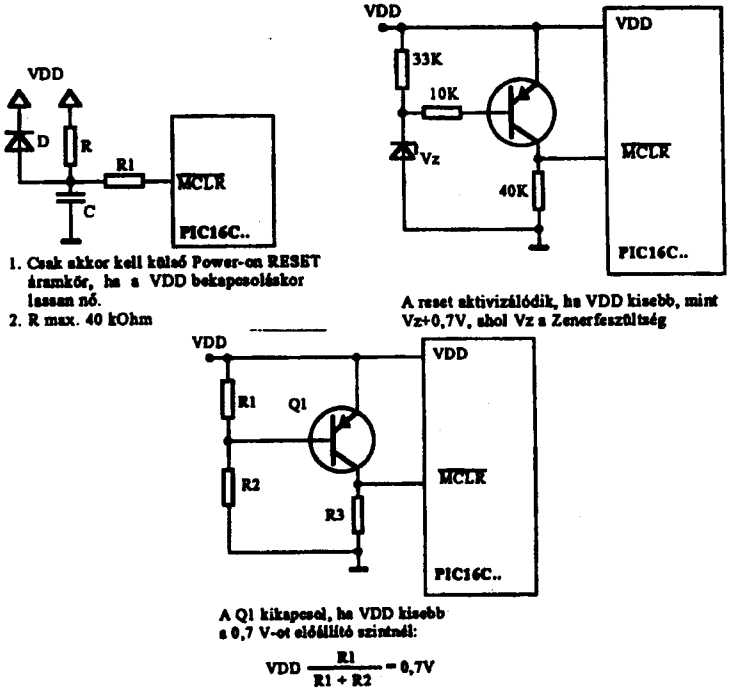
A PIC bemeneteire (OSC1, OSC2) közvetlenül rákötthető egy rezgőkvarc vagy kerámia rezonátor a 7.34. ábra szerint.



A soros ellenállás az AT metszésű kvarcok esetén szükséges

7.34. ábra

A kvarcoszcillátor a PIC belső elemeinek használata nélkül is megépíthető. Ilyenkor az oszcillátor jelét a CLKIN bemenetre célszerű vezetni. A PIC mikrovezérlők az $\overline{\text{MCLR}}$ bemenetén a **tápfeszültség bekapcsolásakor automatikusan működő RESET (Power Up Reset) áramkört alakítottak ki. Ez a megoldás hosszú RESET** időtartamot biztosít az RC értékétől függően, ami akkor előnyös, ha a tápfeszültség bekapcsolásakor csak lassan éri el a névleges értéket. Néhány ajánlott RESET áramkört szemléltet a 7.35. ábra.



7.35. ábra

A RESET áramkör megfelelő kialakítása igen fontos, mert az **egész rendszer megbízhatósága** múlhat rajta. Ha ugyanis a RESET megoldás nem megfelelő, akkor a belső vezérlési folyamatok nem mennek végbe a RESET időtartama

alatt, így a mikrovezérlő véletlenszerű állapottal kezdi meg a működést és ebből a helyzetből nem is tud önmagától kijönni. Ez vezérlőberendezés esetén balesetet is okozhat!

7.6.1.2. A PIC mikrovezérlők átfedéses utasításvégrehajtása

A PIC mikrovezérlőkben külön vezérlő egység kezeli a programmemória kiolvasását és az utasítás végrehajtását. Ez átfedéses utasítás-végrehajtást tesz lehetővé. Az alaposzcillátor (OSC1) impulzusaiból a PIC belső időzítő-vezérlő egysége négy, egymást át nem fedő impulzus sorozatot állít elő, amelynek frekvenciája az OSC1/4. A négy leosztott órajel egy-egy impulzusa alkot egy gépi ciklust. A lejátszódó események:

- Q1 alatt inkrementálódik a PC
- Q2-Q3 idején olvassa be a megcímzett rekeszből az utasításszót
- Q4 alatt a beolvasott utasításszó átmásolódik az utasításregiszterbe.

A következő Q1...Q4 ciklusokban az utasítás dekódolódik és végrehajtódik, miközben a fenti sorrendben a következő utasításszót olvassa be a CPU a programtárból. Minden utasítás értelmezése és végrehajtása **egyetlen gépi ciklus** alatt megtörténik, kivéve az ugró utasításokat, amelyekhez két gépi ciklus szükséges.

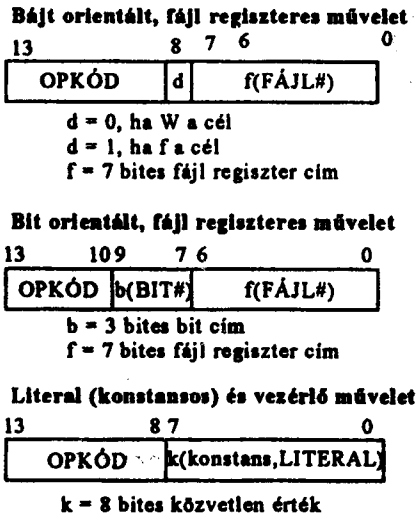
A PIC 16CXX mikrovezérlő utasításkészlete a 7.7. táblázat szerinti.

7.7. táblázat

Mnemonik, operandus		Leírás	Ciklus	14 bites opkód MSb LSb	módosuló flag	megj.
BÁJT ORIENTÁLT, FÁJL REGISZTERES UTASÍTÁSOK						
ADDWF	f,d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f,d	AND W and f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 1fff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f,d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f,d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f,d	Decrement f, Skip if 0	1 (2)	00 1011 dfff ffff		1,2,3
INCF	f,d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f,d	Increment f, Skip if 0	1 (2)	00 1111 dfff ffff		1,2,3
IORWF	f,d	Inclusive OR W and f	1	00 0100 dfff ffff	Z	1,2
MOVF	f,d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 1fff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f,d	Rotate left f through carry	1	00 1101 dfff ffff	C	1,2
RRF	f,d	Rotate right f through carry	1	00 1100 dfff ffff	C	1,22
SUBWF	f,d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f,d	Swap halves f	1	00 1110 dfff ffff		1,2

XORWF	f,d	Exclusive OR W and f	1	00 0110 dfff ffff	Z	1,2
BIT ORIENTÁLT, FÁJL REGISZTERES UTASÍTÁSOK						
BCF	f,b	Bit Clear f	1	01 00bb bff ffff		1,2
BSF	f,b	Bit Set f	1	01 01bb bff ffff		1,2
BTFSC	f,b	Bit Test f, Skip if Clear	1 (2)	01 10bb bff ffff		3
BTFSS	f,b	Bit Test f, Skip if Set	1 (2)	01 11bb bff ffff		3
LITERAL (KONSTANSOS) ÉS VEZÉRLŐ UTASÍTÁSOK						
ADDLW	k	Add literal to W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal to W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear vatchdog timer	1	00 0000 0110 0100	$\overline{TO}, \overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal to W	1	11 1000 kkkk kkkk	Z	
MOVVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from subrutine	2	00 0000 0000 1000		
SLEEP	-	Go to standby mode	1	00 0000 0110 0011	$\overline{TO}, \overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal to W	1	11 1010 kkkk kkkk	Z	

Az utasítások felépítése a 7.36. ábra szerinti.



7.36. ábra

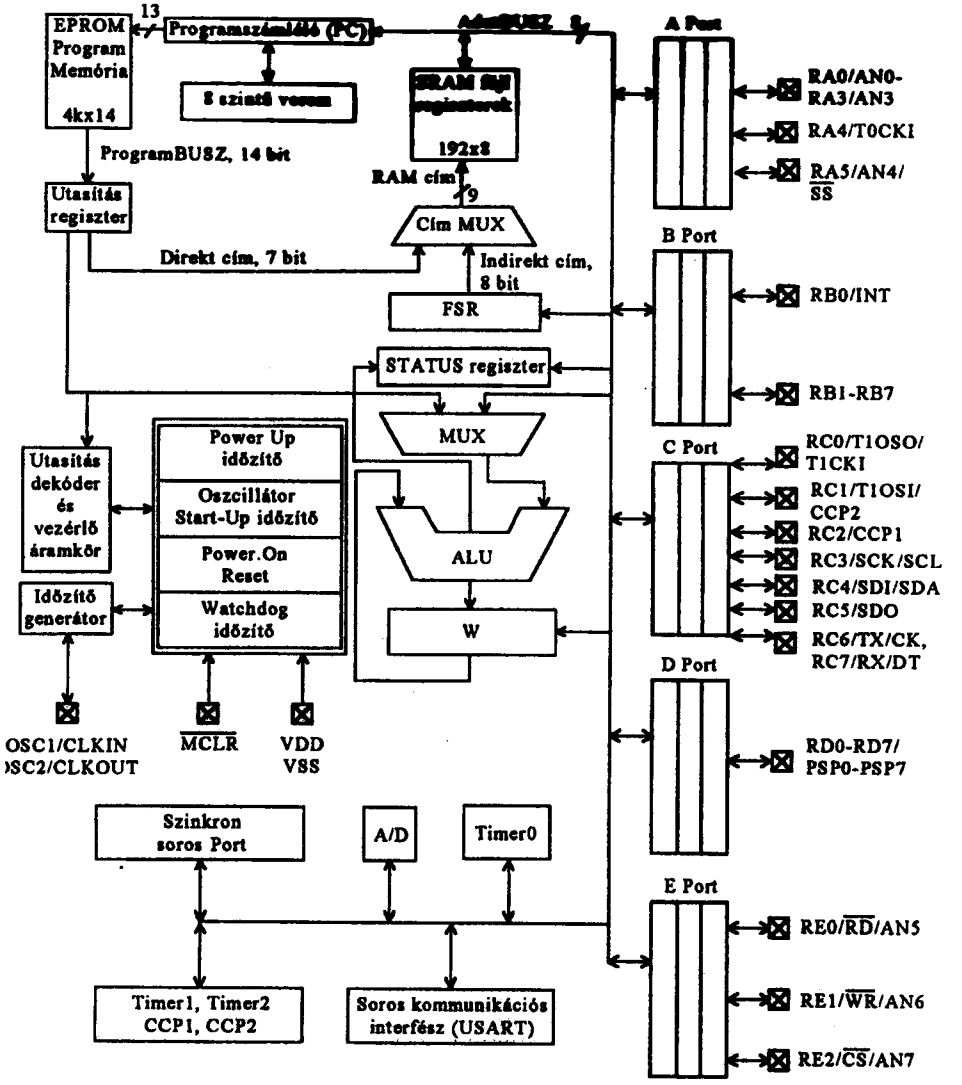
7.6.2. A PIC 16C74 típusú mikrovezérlő

A PIC 16CXX csoportba tartozó mikrovezérlők közül a legbonyolultabb felépítésű 16C74 típust mutatjuk be.

A PIC 16C74 áramkör 192 x 8 bites **adatmemóriával** és 4 k x 14 bites **EPROM** memóriával van ellátva. Minden utasítás egyszavas, az előzőleg beolvasott utasítás végrehajtása és a következő utasításszó beolvasása egyidejűleg történik (pipeline működés). A veremmemória 8 szintes és a PC teljes belső állapotának elmentésére alkalmas. A SRAM rekeszeket direkt és indirekt címzéssel el lehet érni és mind a 45 speciális funkciójú regiszter a fájl regiszter mezőben található. Az egyetlen, az SRAM-tól függetlenül működő belső regiszter a W munkaregiszter, amely **akkumulátorként** funkcionál. A W és az ALU 8 bites, és két operandus közül az egyik operandust a W regiszter adja. A másik operandus lehet egy fájl regiszterben vagy az utasításban, mint konstans. A W közvetlenül nem címezhető regiszter. Az ALU az éppen végrehajtott művelet végeredménye alapján módosítja a flag biteket (C: carry, 8 bites átvitel; DC: digit carry, átvitel az alsó 4 bitről; Z: zero, zérus bit; kivonáskor a C és a DC negált kölcsön flag, borrow). A flag bitek a státusz fájl regiszterben találhatók. A 16C74 mikrovezérlő belső felépítése a 7.37. ábrán látható.

Az oszcillátor felső határfrekvenciája 20 MHz. A 16C74 fontosabb belső egységei és jellemző tulajdonságai:

- 33 I/O csatlakozó láb, egyenkénti adatirány kijelölési lehetőséggel
- nagyteljesítményű kimeneti fokozatok (egy LED közvetlenül működtethető róluk)
- két IC láb az idő kiolvasást (capture) vezérli, vagy a komparáláshoz illetve a PWM működéshez kimenetként szolgál (a kiolvasás - capture - és a komparálás felbontása 16 bit, a PWM felbontása 1...10 bit lehet; 8 bites felbontás esetén a legnagyobb PWM frekvencia 80 kHz, 10 bites esetben csak 20 kHz
- Timer1, TMR1: 16 bites **időzítő/számláló**, 8 bites periódus regiszterrel, előosztóval és utóosztóval, a PWM működéséhez szolgáltathat időalapot
- Timer1, TMR2: 8 bites **időzítő/számláló**, 8 bites periódus regiszterrel, előosztóval és utóosztóval, a PWM működéshez szolgáltathat időalapot
- Timer0, TMR0: a PIC16C5x áramkörben lévő RTCC-vel azonos felépítésű és használatú, 8 bites időzítő/számláló egység, 8 bites programozható előosztóval
- **8 bites A/D, nyolc bemenő csatornával, analóg multiplexerrel**, a konverziós idő 16 μs
- **soros kommunikációs interfész**, SCI (USART), teljes duplex aszinkron vagy félduplex szinkron soros átvitelhez
- szinkron soros Port, két üzemmóddal:
  - három vezetékes SPI
  - I^2C(IIC)

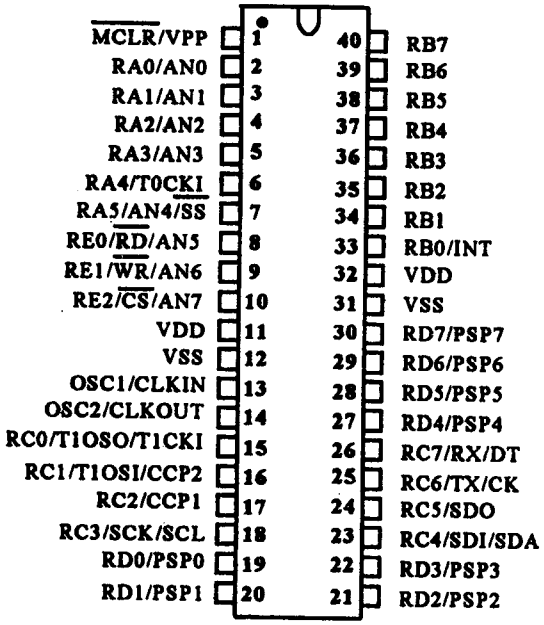


7.37. ábra

- **párhuzamos Slave Port** (PSP), 8 bites, külső $\overline{RD}$,$\overline{WR}$,$\overline{CS}$ vezérlő jelekkel működtethető Port (mikroprocesszor BUSZ interfész)
- tápfeszültség **bekapcsolási Reset** (Power On Reset, POR)
- Tápfeszültség felfutási idő mérő (Power-up Timer, PWRT) és oszcillátor indító (Oscillator Start-up Timer, OST)
- **működést felügyelő számláló** (Watchdog Timer, WDT), belső oszcillátorral
- kódvédő EPROM biztosíték

- tápteljesítmény felvételt csökkentő **Sleep üzemmód**
- EPROM biztosítékokkal választható oszcillátor típus
- **soros in-system** programozási lehetőség (2 IC lábon át).

A CMOS technológiának köszönhetően a működési sebesség nagy; a fogyasztás kis értékű és még tovább csökkenthető a Sleep állapot kiváltásával. A mikrovezérlő bekapcsolási folyamata módosult, a WDT, az OST és PWRT együttműködése révén még nagyobb biztonsággal valósul meg a mikrovezérlő bekapcsolásakor és Sleep állapotból való ébredéskor az, hogy csak teljesen kialakult, tökéletes alakú és időzítésű órajelek mellett kezdődik meg a belső egységek működése. A 16C74 nagyszámú megszakítási lehetőséggel rendelkezik (külső megszakítás, TMRO túlcsordulás, a B Port négy pontján fellépő értékváltozás miatt megszakítás, a TMR1 és TMR2 megszakításait, az SCI aszinkron vételi és adási megszakítása, a szinkron soros Port megszakítása, az SCI megszakítása, az A/D megszakítása és a párhuzamos Slave Port megszakítása). A megszakítások globálisan engedélyezhetők illetve tilthatóak, de egyenként is maszkolhatóak (ha globálisan engedélyezettek). Arra is lehetőség van, hogy egy megszakítás a Sleep állapotból felébressze a mikrovezérlőt. A 16C74 esetén programozható, hogy a teljes programtárat kívánjuk-e titkosítani, vagy a felső felét, vagy a felső harmadát.
A PIC 16C74, mint minden PIC mikrovezérlő, beszerezhető kvarcablakos tokozással, ezek a változatok programfejlesztésre használhatóak. A fejlesztő munka során, ha szükséges, a korábbi program kitörölhető, ugyanúgy, mint egy közönséges EPROM-ból.
A kvarcablakos, CERPID tokozású áramkörök ára meglehetősen magas, sokkal olcsóbbak a zárt, fröccsöntött műanyag tokozású változatok. Igaz, ezeket nem lehet törölni és átprogramozni (OTP, One Time Programmable). A már kifejlesztett, tesztelt programot ezekbe betöltve készülhet el a sorozatgyártott termék. Az OTP áramkörökben az oszcillátor típust a gyártó rögzíti.
A Microchip további kényelmi fokozatot is biztosít. A Quick-Turnaround-Production (QTP) áramkörök tulajdonképpen OTP elemek, de a beprogramozást, tesztelést a gyártó vállalja magára, a felhasználó a kész, programozott és ellenőrzött mikrovezérlőket kapja kézhez. A Serialized Quick-Turnaround-Production (SQTP) megoldás a OTP-hez hasonló, de a sorozat egyes darabjai tartalma kis mértékben eltérhet egymástól. A felhasználó igényének megfelelően néhány rekeszbe a Microchip ilyen rendeléskor az egyedi sorozatszám lehet véletlen érték, álvéletlen vagy folyamatosan növekvő.
A 16C74 mikrovezérlőt 40 kivezetéses DIL, 44 kivezetéses PLCC és 44 kivezetéses PQFP tokozással forgalmazza a Microchip. A DIP tokozás lábkiosztása látható a 7.38. ábrán.



7.38. ábra

A nagyszámú belső egység, a sokoldalú kapcsolati rendszer a külső környezettel sokkal több csatlakozási pontot kívánna meg, mint 40 illetve 44. Más gyártók az ilyen bonyolultságú mikrovezérlőket már nem is gyártják DIL tokozással. A 40 kivezetés csak úgy elegendő a 16C74 számára, hogy az IC lábak többsége többfunkciós. Van, amelyiknek három különféle feladata is lehet. A programmemóriát (EPROM) a PC tartalmával címezi a mikrovezérlő. A 16C74 programszámlálója 13 bites, de a mikrovezérlő csak 12 bitet használ ebből (0000-0FFFh), hiszen a programtára 4k x14 kapacitású. A RESET vektor (a RESET folyamat után aktivizálódó cím) 0000h, a megszakítási vektor (a megszakításkor aktivizálódó cím) 0004h.

Itt találjuk a speciális funkciójú regisztereket (fájl regiszterek) és az általános célra használható rekeszeket is. A Bank0 akkor választódik ki, ha a STATUS regiszterben RP0=0, az RP0=1 a Bank 1-et jelöli ki. Mindkét Bank 128 bájtos, mindkettő első 32 bájtja a speciális funkciójú regiszterek számára van fenntartva. A 20h...7Fh (Bank0) és az A0h...FFh (Bank1) regiszterek az általános célra felhasználható SRAM rekeszek.

Az összesen 256 SRAM rekesz mindegyike elérhető közvetlen címzéssel is és közvetett, regiszteres címzéssel is (ekkor az FSR fájl választó regiszter tartalma a cím). A speciális funkciójú regiszterek tartalmának változtatásával lehet a mikrovezérlő belső moduljait kezelni, programozni.

A további információkat illetően Dr. Madarász László: A PIC mikrovezérlők c. jegyzetre utalunk.

Felhasznált irodalom

Abonyi-Benesáczky-Hainzman: Illesztés mikroszámítógéphez Műegyetem Kiadó, 1997.

Agárdi-Hadi: Fókuszban a Pentium LSI-Számalk, 1996.

Ajtonyi I:Vezérlés és mikroprocesszortechnika gyakorlatok III. és adatgyűjtemény Tankönyvkiadó, Budapest, 1986.

Ajtonyi I: Vezérlés- és mikroprocesszortechnika II. J 14-1650 Tankönyvkiadó, Budapest 1989.

Ajtonyi I: Vezérlés- és mikroprocesszortechnika I. J 14-1629 Tankönyvkiadó, Budapest, 1989.

Cserny L: RISC processzorok LSI-Számalk, ISBN 963 577 155 X.

Cserny L: Mikroszámítógépek LSI-Számalk, ISBN 963 577 188 6.

Madarász L: A PIC16C Mikrovezérlők GAMF Kecskemét, 1996.

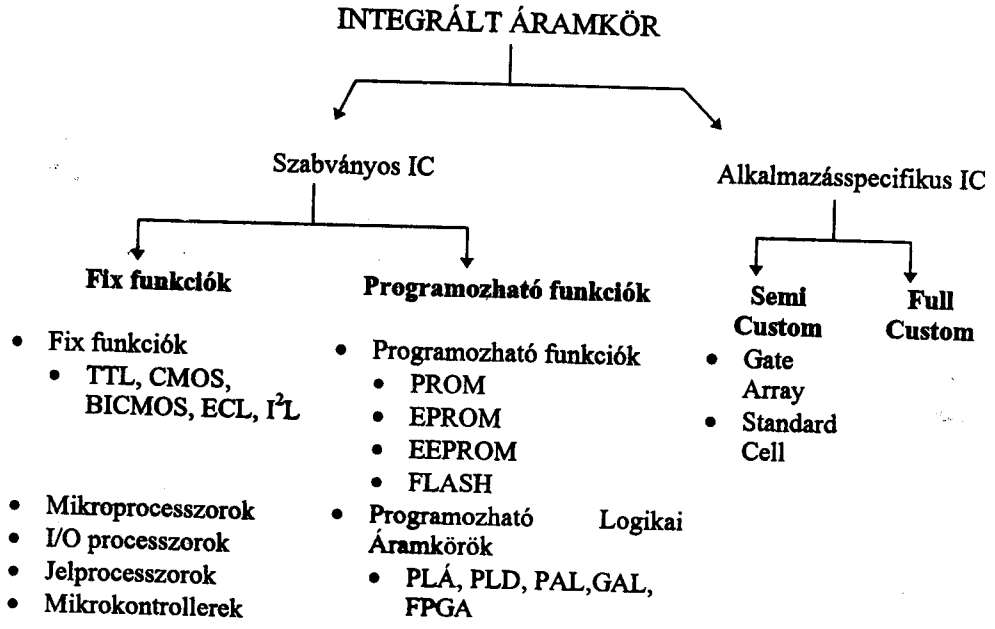
S.A. Money: Practical Microprocessor Interfacing. John Wiley & Sons, 1987.

8. PROGRAMOZHATÓ LOGIKAI ESZKÖZÖK

A huzalozott logikában használatos digitális integrált áramköröket a 4. fejezetben bemutattuk. A huzalozott logikai elemeket a programozható digitális áramkörök követték. A programozható digitális áramkörök alapvetően két nagy csoportba sorolhatók. Az egyik csoportba az **időosztásos elven működő áramkörök** tartoznak, amelyek működését **kristályoszcillátor ütemezi** és a **program ismétlődően 'lefutva" hajtódik** végre. Az ilyen áramkörök (pl. mikroprocesszor, PLC) egyik tipikus jellemzője, hogy a **kombinációs típusú függvényeket is sorrendi hálózat realizálja**. Ezen áramköröknek számos előnye mellett két nagy hátrányuk van:

- az időosztásos működés miatt a realizált hálózat működési frekvenciája alacsony
- a sorrendi jellegű hardver miatt nagy a zavarérzékenységük.

Az utóbbi 15 évben a fejlesztés a hagyományos huzalozott logika felé fordult annak programozhatóvá alakítása céljából. Így fejlődtek ki a különböző **programozható logikai elemek,** amelyek a hagyományos **huzalozott architektúrát** követik, **a program a működés előtt kerül letöltésre**, ezért gyakran **letöltő programnak** nevezik. Ezen áramkörök további előnyei, hogy **felhasználó specifikusak,** tervezésükhöz speciális szoftverek állnak rendelkezésre, tervezésük gyors, nehezen másolhatók. Ebben a fejezetben ezen utóbbi típusú programozható elemekkel foglalkozunk. Rendszertechnikailag az alábbiak szerint csoportosíthatók ezek az áramkörök.



Foglaljuk össze, mit jelentenek az eddig nem használt rövidítések.
A PLD (Programmable Logic Device), amelynek két osztálya van:

- SPLD (Simple PLD)
- CPLD (Complex PLD).

A programozható logikai eszközök egy olyan **kétszintű kapuáramköri tömböt**, illetve **flip-flop sorozatot** tartalmaznak, amelyek egymással való összekötése még nincs kialakítva. A kapuáramkörök közötti összeköttetéseket a felhasználó véglegesíti az ún. letöltő program révén, amelyet gyakran **konfigurálásnak** neveznek. **A felhasználó által programozható kapu mátrix áramkörök** (Field Programmable Gate Array = FPGA) egy olyan mátrix struktúrájú elrendezésből állnak, ahol egyrészt a **cellák közötti összekötések**, másrészt a **cellák logikai funkciói** is a felhasználó által programozhatók. Az **alkalmazás specifikus integrált áramkörök** (ASIC = Application Specific Integrated Circuits) alapvetően abban különböznek a szabványos IC-ktől, hogy maga a felhasználó részben vagy egészben meghatározza a legyártandó áramkört, azaz az áramkör által megvalósítandó funkciókat, annak műszaki paramétereit, stb. Az ASIC áramkörök két fontosabb osztálya létezik:

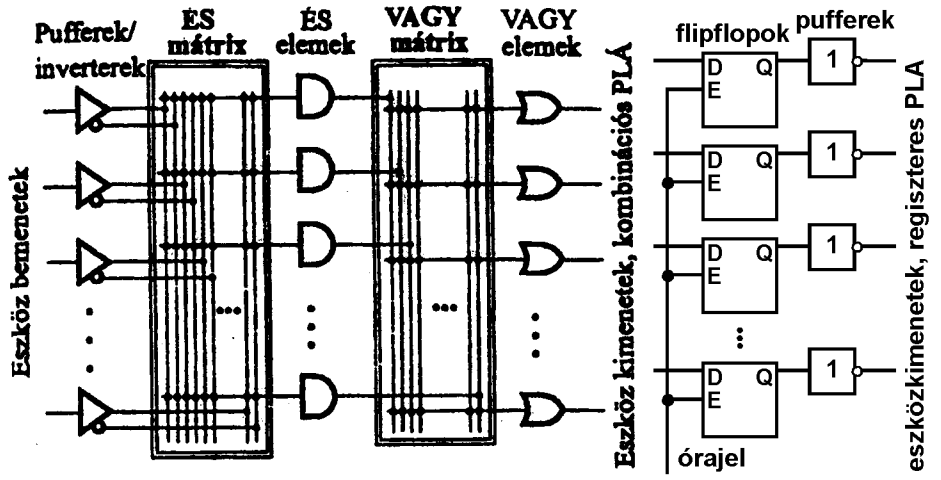
- **Semi custom** technológia esetén a gyártás részben a felhasználó igényei szerint megy végbe (a könyvtárilag definiált maszkokból a felhasználó állítja össze a végső hardvert). Ilyen áramkörök a **berendezés orientált áramkörök** (BOÁK).
- **Full Custom** technológia esetén a gyártási folyamat a kezdetétől fogva a felhasználó előírása szerint megy végbe. Költségei miatt csak nagy sorozatok esetén használják. Hangsúlyozni kell. hogy az angol irodalomban szinte valamennyi programozható logikai áramkört az ASIC kategóriába sorolják, mivel a felhasználó a program révén tulajdonképpen a hardvert módosítja.

8.1. PLA, FPLA eszközök

A **programozható logikai áramkörök** (Programmable Logic Array = PLA) olyan **kétszintű ÉS/VAGY hálózatot tartalmaznak, amelyekben mind az ÉS mátrix mind a VAGY mátrix bekötései programozhatók.** Két alapvető típusuk van a hálózat típusa szerint:

- **kombinációs típusú PLA**
- **regiszteres típusú PLA** (a kombinációs típusú PLA egy flip-flop sorral van kiegészítve).

A felhasználó által programozható PLA neve Field Programmable Logic Array (FPLA). A PLA/FPLA áramkörök általános struktúrája a 8.1. ábra szerinti.



8.1. ábra

Az ábra szerint az elem bemeneti fokozata jelmásoló/jelfordító funkciót tartalmaz, amely azt biztosítja, hogy **az ÉS kapukba a változó ponált vagy negált értéke a programozástól függően bevihető legyen.** Az ÉS mátrix az ÉS elem bemenetére jutó változó kombinációt realizálja a program szerint. A VAGY mátrix a program szerint a VAGY kapuba vezetendő ÉS kimenetek bekötését realizálja. A PLA-kat a ROM-okhoz, az FPLA-kat az EPROM-okhoz hasonló technológiával gyártják. A PLA-k tulajdonképpen **hiányos címmezejű ROM** memóriaként is felfoghatók. Eszerint a PLA tulajdonképpen két ROM-ból épül fel: egy ÉS ROM-ból és egy VAGY ROM-ból. Ezt szemlélteti a 8.2.a) ábra.

A 8.2.c) ábra a P_1, P_2 ill. Q_1, Q_2 függvények származtatását mutatja be.

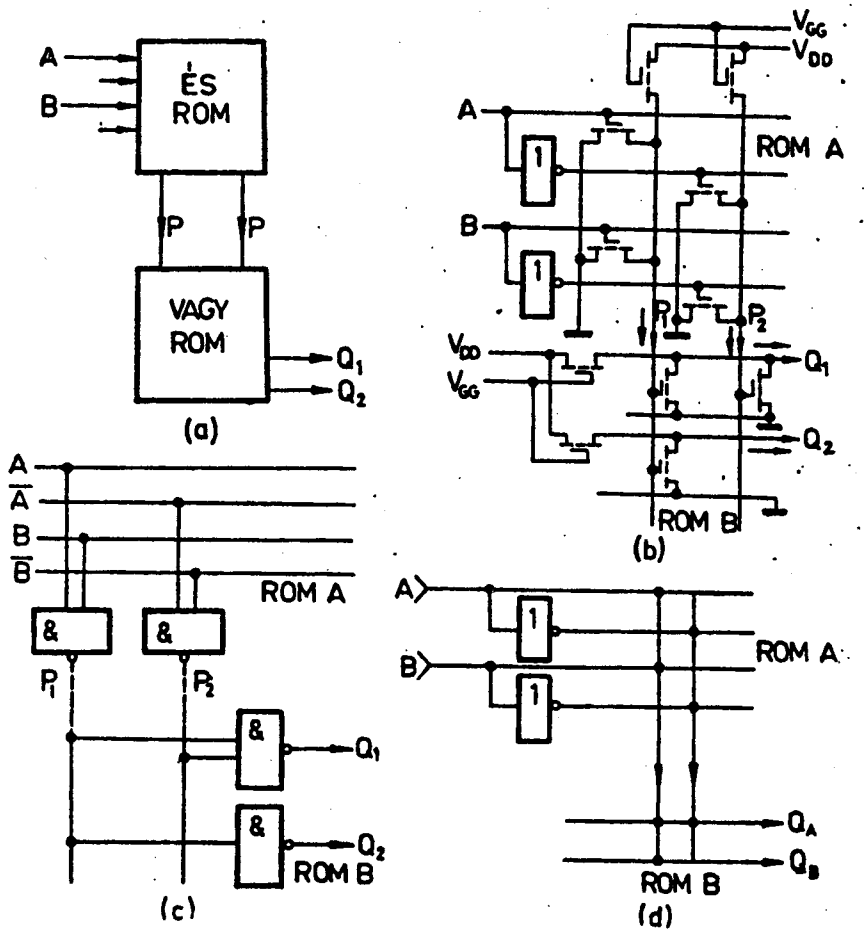
$$P_1 = \overline{A \& B}$$

$$P_2 = \overline{\overline{A} \& \overline{B}} = A \vee B.$$

A d) ábra a PLA-nál szokásos jelölés rendszert szemlélteti.

$$Q_1 = \overline{P_1 \& P_2} = \overline{\overline{A \& B} \& (A \vee B)} = \overline{\overline{A \& B}} \vee \overline{(A \vee B)} = A \& B \vee \left(\overline{A} \& \overline{B}\right)$$

$$Q_2 = \overline{P_1} = A \& B.$$

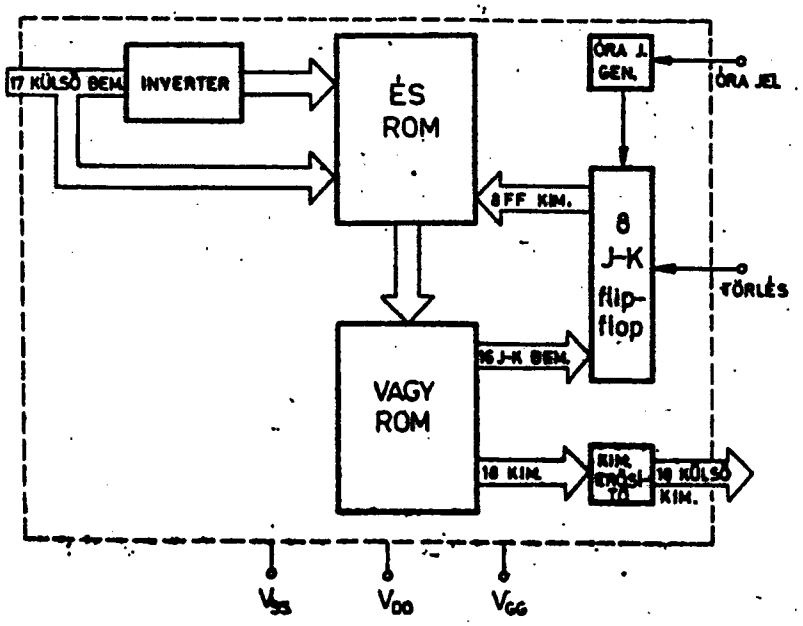


8.2. ábra

A TEXAS típusú regiszteres PLA struktúráját mutatja a 8.3. ábra, két típus adatait pedig a 8.1. táblázat tartalmazza.

8.1. táblázat

Az áramkör jellemzői:	TMS 2000	TMS 2200
Bemenetek száma:	60	72
ÉS kapuk száma:	17	13
Kimenetek (VAGY kapuk) száma:	18	10
JK FF-ok száma	8	10



8.3. ábra

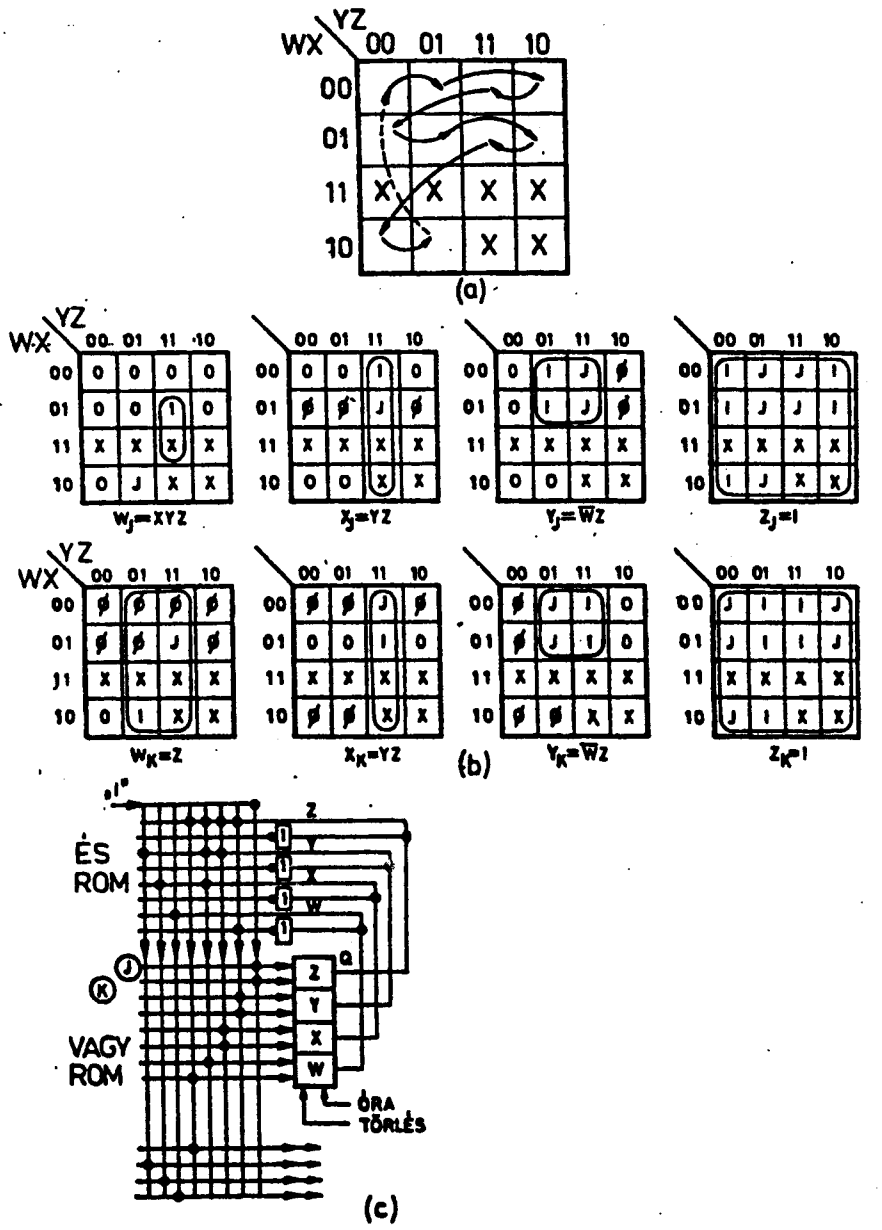
A PLA-k programozása gyárilag, maszkolással történik a felhasználó megrendelése és specifikálása alapján.
Az FPLA-k programozása rendszerint 3 lépésben történik a specifikálásban megadott műszaki feltételek mellett speciális programozó készülék segítségével a felhasználó által:

- az ÉS mátrix programozása
- a VAGY mátrix programozása
- a kimenet programozása.

Foglaljuk össze a PLA/FPLA-k legfontosabb jellemzőit:

- a PLA-k olyan eszközök, amelyekben **mind az ÉS, mind a VAGY mátrix keresztpontjai programozhatók,**
- mivel a PLA eszközökben a ÉS mátrix programozható, ezért csak azokat az ÉS kapcsolatokat kell előállítani, **amelyeket a függvény tartalmaz,**
- olyan logikai funkciók megvalósítására előnyösek, melyek **sok bemeneti változót** és **kevés szorzattagot** igényelnek,
- a kimeneti függvényben szereplő **közös szorzattagot** (közös implikánst) csak egyszer kell előállítani,
- mivel a VAGY kapu bemenetek száma korlátozott, **tetszőleges logikai függvény megvalósítása nem garantált,**
- ellentétben a memóriákkal a PLA-k a logikai függvényt nem az igazságtáblázat, hanem **logikai egyenlet alapján realizálják.**

Egy NBCD kódú szinkron előre számláló tervezése és realizálása látható a 8.4. ábrán a PLA/FPLA-knál szokásos jelölésekkel.



8.4. ábra

- A PLA-k alaptípusait a Signetics cég fejlesztette ki, és IFL (Integrated Fuse Logic = integrált biztosíték logika) névvel látta el. A Signetics négy alaptípust fejlesztett ki:
  - FPLA (Field Programmable Logic Array)
  - FPLS (Field Programmable Logic Sequencer = felhasználó által programozható logikai vezérlő)
  - FPRP (Field Programmable ROM Patch = felhasználó által programozható ROM foltozó)
  - FPGA (Field Programmable Gate Array = felhasználó által programozható kapu tömb).

Utóbbi nem egyezik meg a napjainkban használatos FPGA eszközökkel!

8.2 PAL/GAL eszközök

a) PAL eszközök

A programozható tömb logikák (Programmable Array Logic) elnevezés az Advanced Micro Devices cég regisztrált mintavédjegye. A PAL-ok struktúrális felépítése megegyezik a PLA-kéval azzal az alapvető eltéréssel, hogy **a PAL-ok ÉS mátrixa programozható,** de **a VAGY mátrixa nem programozható**, emiatt a PLA-khoz képest **gyorsabb működésűek**. A gyakorlatban használt PAL-okban az egy kimenethez tartozó szorzattagok száma korlátozást jelent, ezért főként a **kevés szorzattagot** igénylő függvényeket lehet csak kétszintű logikával realizálni PAL-okkal. Típus megjelölésük rendszerint az alábbiak szerint történik:

PAL aa b cc

ahol

aa az ÉS mátrix maximális bemeneteinek számát adja meg

b a kimenetek típusát definiálja a következők szerint:

H	aktív magas kimenet
L	aktív alacsony kimenet
P	programozható polaritású kimenet
C	komplemens kimenet

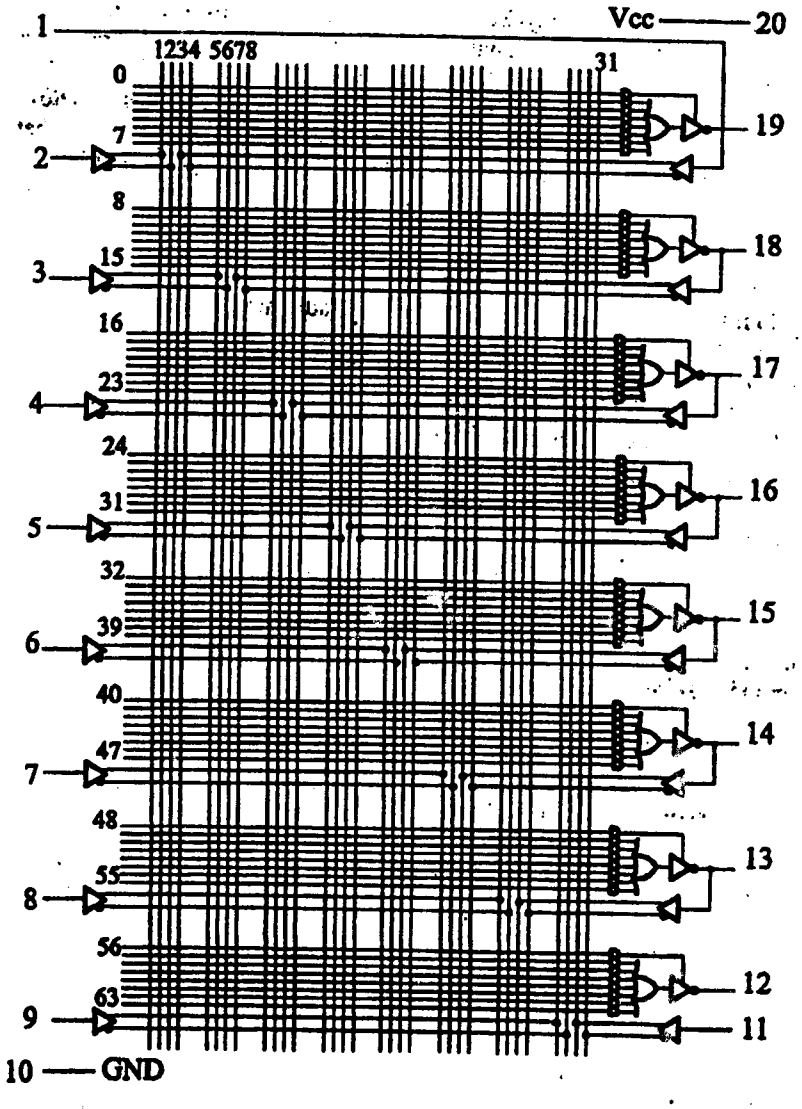
regiszteres PAL esetén

R	regiszteres kimenet
RP	regiszteres, programozható polaritású kimenet
V	flexibilis, azaz programozható makrocellás kimenet.

cc a dedikált vagy programozott kimenetek maximális száma.

Példaként a PAL 16L8 áramkör sémáját adjuk meg a 8.5. ábrán. A 16-os szám a típusjelben az ÉS mátrix bemeneteinek számát adja meg. Egy változó ponált és negált értékének a becsatolása az ÉS mátrixba egy bemenetnek számít, ill. az ÉS

mátrix bemeneteibe az esetlegesen a **kimenetről visszacsatolt jelek is beszámítanak**. Az ábrán látható eszköznél 10 csak bemenetként használható (ún. dedikált) bemenet csatlakozik az ÉS mátrixba, másrészt a 8 kimenet közül 6 vissza van csatolva az ÉS mátrixba. Ezeket a kivezetéseket be/kimenetnek nevezik, mivel a TS elemek programozásával be ill. kimenetként egyaránt használhatók. A 8 kimenet közül 2 csak kimenetként használható. Tehát a PAL 16L8-nak 10 dedikált bemenete, 2 dedikált kimenete és 6 be-vagy kimenetként egyaránt használható kivezetése van. Az ábrából az is megállapítható, hogy kimenetenként max 7 szorzattag áll rendelkezésre.



8.5. ábra

A **regiszteres PAL-ok** kimenetükön a kimenetekkel megegyező számú **DFF-ot** tartalmaznak. Ezek többsége **szinkronizált működésű**, azaz a DFF-ok közös CP jelre billennek. Ezeket **szinkronizált regiszteres PAL**-oknak nevezik. E mellett gyártanak aszinkron működésű regiszteres PAL-okat is, melyeknél az egyes DFF-ok egymástól független órajelre működnek és az egyes FF-ok egyedi aszinkron beíró és törlő bemenetekkel is rendelkeznek.
Néhány AMD gyártmányú PAL áramkör műszaki jellemzőit a 8.2. táblázatban adtuk meg:

8.2. táblázat

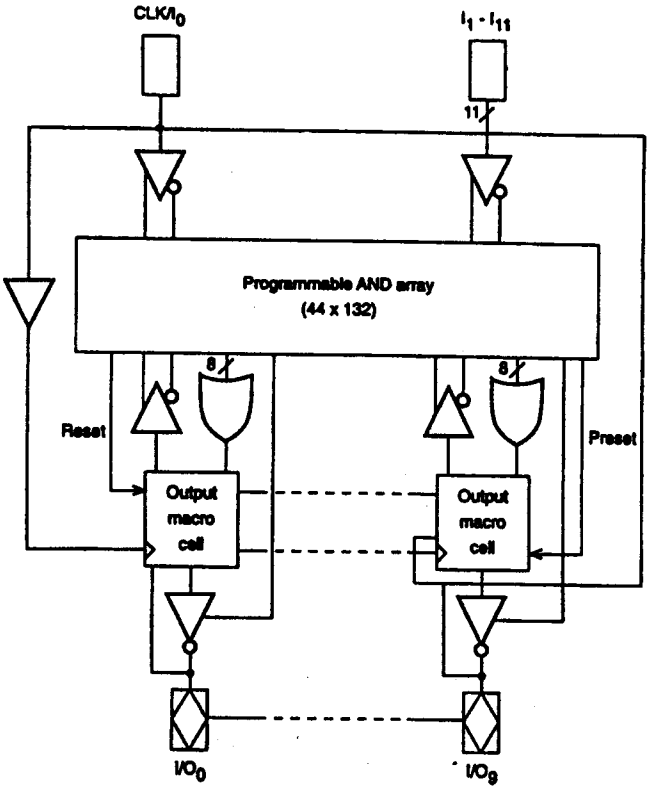
Típus	Technológia	tpd (ns)	Icc (mA)	fmax (MHz)
PAL 16L8-4	TTL	4,5	210	125
PAL 22P10B	TTL	15,0	180	-
PALCE 16V8H	EECMOS	5,0	125	166
PALCE 26V12H	EECMOS	7,5	115	125

b) GAL eszközök

A **General Array Logic (GAL)** eszközöket a Lattice cég fejlesztette ki. A GAL áramkör a PAL továbbfejlesztésének tekinthető, mivel az **ÉS mátrix programozhatósága mellett egy összetett kimeneti áramkör programozhatóvá tételével jelentősen nőtt az áramkör rugalmassága.** Ezt a programozható kimeneti modult **makrocellának** nevezték el. Példaként a leggyakrabban használt Lattice GAL CELLV10 architektúráját mutatjuk be a 8.6. ábrán.

Az eszköznek **12 bemenete** van és ezek mindegyike becsatlakozik az ÉS mátrixba. Az I_0 bemenet a makrocellákban található órajelbemenetként használható. Az eszköznek 10 kimenete, pontosabban **I/O kimeneti makrocellája** van. Az egyes makrocellákhoz csatlakozó szorzattagok száma nem azonos: 8, 10, 12, 14, 16, 16, 14, 12, 10, 8. Ezek összege 120. Mivel azonban minden kimeneti puffer engedélyezése egy-egy szorzattaggal történik (10) és az aszinkron törlőjelet egy-egy szorzattag állítja elő, így szorzattagok száma 120 + 10 + 2 = 132. Az ÉS mátrixba a 12 dedikált bemeneten kívül minden kimeneti makrocellából is becsatlakozik egy-egy visszacsatolás, ebből adódik az ÉS mátrix mérete, azaz keresztpontjainak száma: 2 x (12 + 10) x 132 = 44 x 132. A makrocella felépítése a 8.7.a) ábra alapján követhető.
A makrocella egyaránt használható bemenetként, kombinációs kimenetként és regiszteres kimenetként a programozástól függően. A programozása az S_0, S_1 üzemmódvezérlőkkel végezhető az alábbiak szerint:

00 regiszteres, aktív alacsony
01 regiszteres, aktív magas
10 kombinációs, aktív alacsony
11 kombinációs, aktív magas.



8.6. ábra

A b) ábra szemlélteti a 4 féle állapotot. Megjegyezzük, hogy a szabványosnak tekinthető Lattice GAL 22V10 elemtől némileg módosított funkciójú GAL áramköröket is gyártanak (Intel, National Semiconductor) és jelenleg is fejlesztenek.

PAL programozási példák

a) ÉS kapu

74-es sorozat: 08, 11, 21.
4000-es sorozat: 4073, 4081, 4082.

PAL programozási tábla

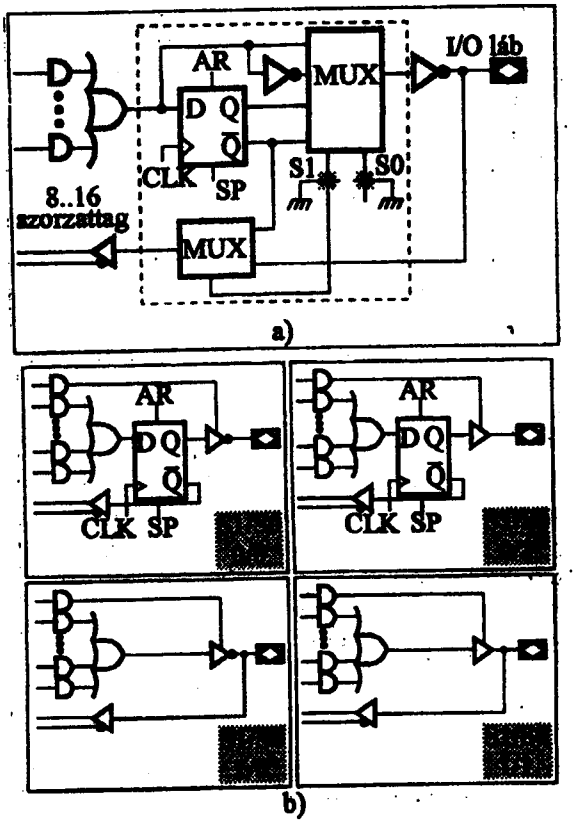
Aktív szint:	L
A B C	Y
H H H	A

b) NAND kapu

74-es sorozat: 00, 10, 20, 33, 133.
4000-es sorozat: 4011, 4012, 4023, 4068.

PAL programozási tábla

Aktív szint:	L
A B C	Y
H H H	A



8.7. ábra

c) VAGY kapu

Aktív szint:	L
A B C	Y
L L L	A

d) NOR kapu

74-es sorozat: 02, 77.
4000-es sorozat: 4000, 4001, 4002, 4025, 4078.

PAL programozási tábla

Aktív szint:	H
A B C	Y
L L L	A

Megjegyzés: figyeljük meg, hogy az utóbbi esetben a PAL AND mátrixot kell OR ill. NOR műveletre programozni (De Morgan szabály).

e) Exlusív OR kapu

74-es sorozat: 86, 135.

4000-es sorozat: 4030, 4070, 4077, 4507.

PAL programozási tábla

Aktív szint:	H
A B	Y
H L	A
L H	A

f) RS0 FF

74-es sorozat: 279.

4000-es sorozat: 4043, 4044.

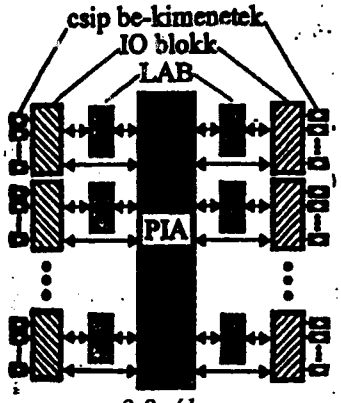
PAL programozási táblázat

Bemenetek				Kimenetek	
R	S	Q	Q	Q	Q
Aktív szint				L	L
H	-	-	H	A	-
-	H	H	-	-	A

8.3. CPLD eszközök

A **komplex programozható áramkörök** (Complex Programmable Logic Devices = PLD) a PLA és PAL áramkörök után kerültek kifejlesztésre, fejlesztésük napjainkban is folyik. A CPLD tulajdonképpen a PAL/PLA áramkörök továbbfejlesztésének tekinthető. A CPLD architektúrája általában 3 fő részből áll a 8.8. ábra szerint.

- LAB = Logic Array Block (logikai tömb blokk)
- I/O Block = Input/Output Block (be-kimeneti blokk)
- PIA = Programmable Interconnect Array (programozható összekötő tömb).

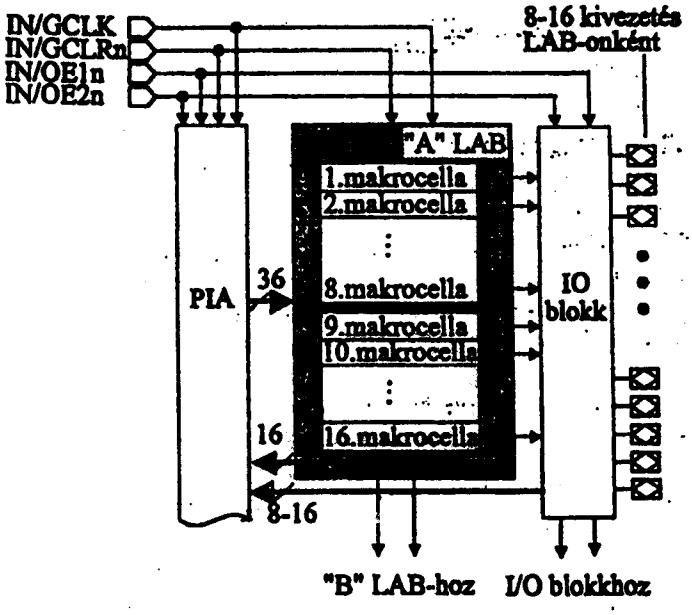


8.8. ábra

A LAB-ok **két makrocella** készletet tartalmaznak, amelyek önmagukban egy-egy SPLD-nek felelnek meg. A LAB-ok a PIA-n keresztül köthetők össze egymással. A PIA ugyanis bármely LAB bemenetét vagy kimenetét bármely

másik LAB kimenetével vagy bemenetével össze tudja kötni. A PIA tehát egy **globális busz** szerepét látja el. A PIA-n keresztüli összekötés szintén programozható, ez képezi a LAB-ok programozása (alsó szint) mellett a programozás felső szintjét. A LAB-ok rendszerint makrocellákat tartalmaznak. A LAB-ok architektúráját a 8.9. ábra szemlélteti. Minden LAB két macrocella csoportból épül fel, s egy-egy csoport nyolc makrocellát tartalmaz.

Minden makrocella egyedileg **kombinációs** vagy **regiszteres típusúra** programozható. A kombinációs logikát a **programozható logikai tömb** valósítja meg, amely makrocellánként minimum 5 ÉS kaput tartalmaz. Regiszteres üzemmódban az egyes makrocellák regiszterei egyedileg T, D, JK vagy RS FF-ként programozhatók. A CPLD chip makrocellái egyedileg **nagysebességű** (Turbo bit = 1) vagy **kisfogyasztású** (Turbo bit = 0) üzemmódba működtethetők attól függően, hogy a logikai rendszeren belül milyen sebességigényre van szükség. A pLSI és ispLSi CPLD családot a Lattice cég fejlesztette ki. Az **ispLSi** a **rendszerbe helyezetten programozható** (in system programmable) jellegre utal. Ez azt jelenti, hogy a pLSi csak egy külön programozó eszközben programozható, az ispLSi áramkörök viszont a célrendszerbe való beépítésük után is programozhatók és újraprogramozhatók.



8.9. ábra

Ez azt jelenti, hogy a pLSi csak egy külön programozó eszközben programozható, az ispLSi áramkörök viszont a célrendszerbe való beépítésük után is programozhatók és újraprogramozhatók. A pLSi és ispLSi eszközök legalapvetőbb elemei a logikai funkciókat megvalósító **általános logikai** blokkok (Generic Logic Block = GLB).

Egy GLB első közelítésben egy 18 bemenetű, programozható ÉS/VAGY/KIZÁRÓ VAGY mátrixú és 4 kimenetű SPLD-nek felel meg. Egy **GLB akár kombinációs, akár regiszteres (D, T, JK)** üzemmódra konfigurálható és mindegyik csatlakozik a kimeneti összekötő hálózathoz és a globális összekötő hálózathoz. Ez utóbbi biztosítja, hogy **bármelyik GLB kimenet bármelyik GLB bemenettel összeköthető** legyen. Az eszköz és környezete közötti kapcsolatot az **IOC (I/O Cell)** valósítja meg. 8 GLB, 16 IOC, 2 dedikált bemenet és egy kimeneti összekötő hálózat (ORP) alkot egy **megablokkot.** A **globális összekötő hálózat** (Global Routing Pool = GRP) bemenőjeleit a GLB-k kimenő jelei és a kétirányú IOC-k jelei alkotják. Az eszköz tartalmaz egy órajel elosztó hálózatot. Néhány Lattice pLSi eszköz összehasonlítás a 8.3. táblázatban található.

8.3. táblázat

Paraméter	pLSi 1000		pLSi 2000		pLSi 3000	
	1016	1048	2032	2096	3192	3320
PLD kapuk	2000	8000	1000	4000	8000	14000
fmax (MHz)	80	80	137	110	110	80
GLB-k száma	16	48	32	96	192	320
I/O (max)	36	106	34	102	96	160

8.4. FPGA eszközök

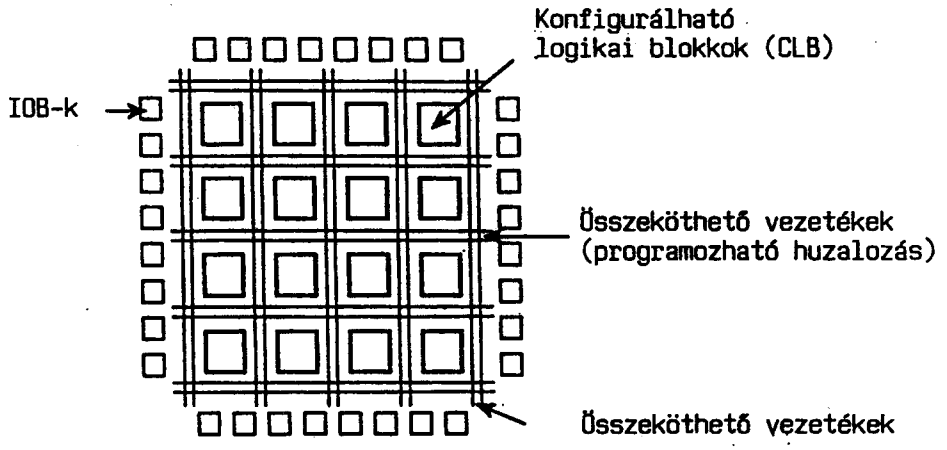
A **programozható kapu tömbök** (Field Programmable Gate Array = FPGA) abban térnek el a korábban ismertetett programozható eszközöktől, hogy ezektől eltérően nem valamilyen programozható ÉS/VAGY mátrixot, hanem az egyes logikai cellák közötti programozható összekötés (más szóval routing) révén **sokkal nagyobb szabadságfokkal** rendelkeznek. Felépítésük leginkább a nyomtatott áramköri technikákhoz hasonlít, amikoris az egyes IC-ket a NYÁK-on összehuzalozzuk. Az FPGA-k jellemzői:

- a **programozható logikai blokkok** kétdimenziós (mátrix jellegű) tömbjéből épülnek fel,
- **programozható összekötő hálózattal** rendelkeznek
- **a felhasználó által programozhatók**
- **többszintű logikát** valósítanak meg.

Az FPGA-k három alapvető eleme:

- **a programozható logikai blokk** (Configurable Logic Block = CLB)
- az FPGA és a környezet közötti kapcsolatot biztosító **be/kimeneti blokk** (Input-Output Block = IOB)
- a CLB-ket egymással és az IOB-kel összekötő **programozható huzalozás.**

Az FPGA architectúrája a 8.10. ábrán látható.



8.10. ábra

Az FPGA-kat az összekötő huzalozási elrendezés alapján az alábbi csoportokba soroljuk:

- **szimmetrikus huzalozási** elrendezés (pl. XILINX)
- **asszimetrikus huzalozási** elrendezés (pl. Actel)
- **kaputenger topológiájú** elrendezés (pl. Atmel)
- **hierarchikus architektúrájú** elrendezés (pl. Altera).

Az FPGA eszközök által megvalósított logikai funkciókat a **logikai blokkok** valósítják meg. Az egyes FPGA-k logikai blokkjai nagy mértékben eltérhetnek egymástól: pl. NAND kapu, ULM modul stb. A logikai blokkok egyszerű vagy bonyolult volta és az összehuzalozási kapacitás között szoros kapcsolat van. Belátható, egyszerű felépítésű logikai blokk esetén a bonyolultabb hálózat kiépítéséhez több elemre és huzalozásra van szükség. Ugyanakkor a komplexebb logikai blokkok kevesebb huzalozást igényelnek, de blokkonként nő a logikai blokkok kihasználtsága. A **be/kiviteli blokkok** (IOB) a logikai blokkok és a külvilág közötti interfész szerepét látják el. **Általában bemenetként, kimenetként,** ill. **kétirányú elemként** programozhatók. A be- ill. kimenetek **regiszteres** ill. **regiszter nélküli** üzemmódban programozhatók. Az összekötési mátrix **vezetékeket és programozható kapcsolókat** tartalmaz. Programozáskor ezen kapcsolók bekapcsolásával hozhatunk létre **galvanikus összeköttetéseket** két vezeték között. **Programozható kapcsolóként RAM cellával vezérelt áteresztő tranzisztor, anti biztosíték, EPROM vagy EEPROM tranzisztorok** egyaránt használatosak.

8.4.1. XILINX FPGA eszközök

Az első FPGA áramkört a XILINX cég fejlesztette ki 1985-ben. A cég három családot fejlesztett ki. Ezek kronológiai sorrendben XC 2000, XC 3000, XC 4000. Az egyes családok között főként mennyiségi különbségek vannak. Példaként:

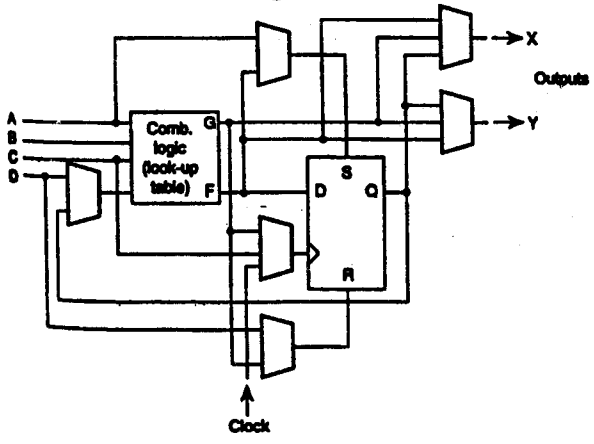
- a CLB-vel megvalósítható logikai függvény: 4, 5, 5 változós a fenti típusok esetén
- a CLB flip-flopjainak száma: 1, 2, 2 a fenti típusok esetén.

Az egyes XILINX családok főbb jellemzőit a 8.4. táblázatban foglaltuk össze.

8.4.táblázat

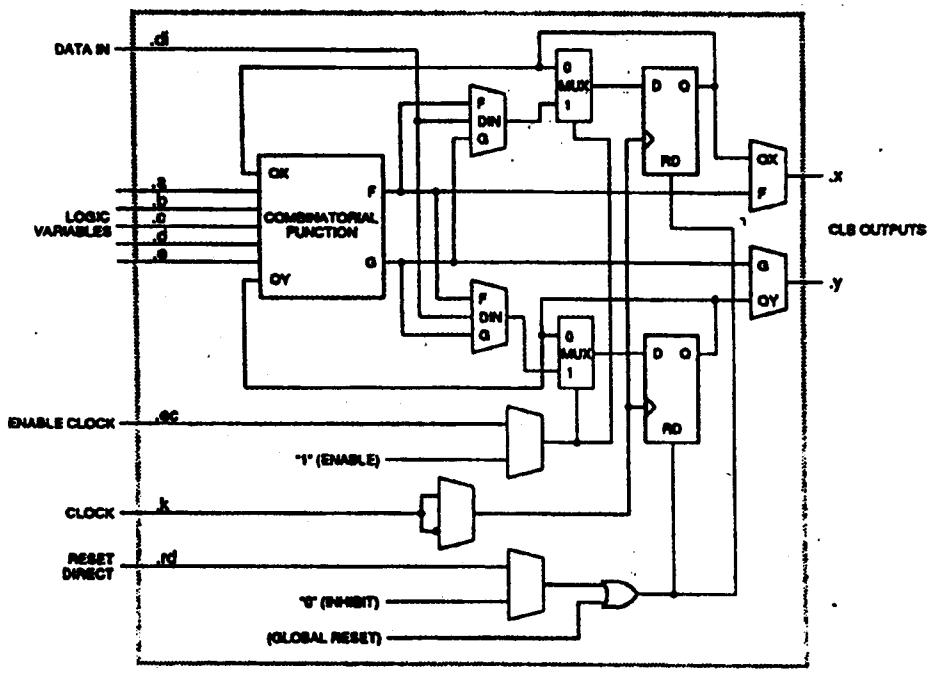
Típus	Ekvivalens kapuk száma	CLB-k száma	Tömb mérete	I/O-k száma	Konfigurációs RAM (bit)
XC 2000					
XC 2064	1200	64	8x8	58	12038
XC 2018	1800	100	10x10	74	17878
XC 3000					
XC 3020	2000	64	8x8	64	14779
XC 3030	3000	100	10x10	80	22176
XC 3090	9000	320	16x20	176	94984
XC 4000					
XC 4005	5000	196	14x14	112	81332
XC 4008	8000	324	18x18	144	147504
XC 4010	10.000	400	20x20	160	178096
XC 4025	25.000	1024	32x32	256	422128

Az XC 2000-es CLB funkcionális felépítése a 8.11. ábrán látható.



8.11. ábra

Összehasonlításul az XC 3000-es (8.12. ábra) ill. XC 4000-es CLB funkcionális felépítése a 8.13. ábrán látható.

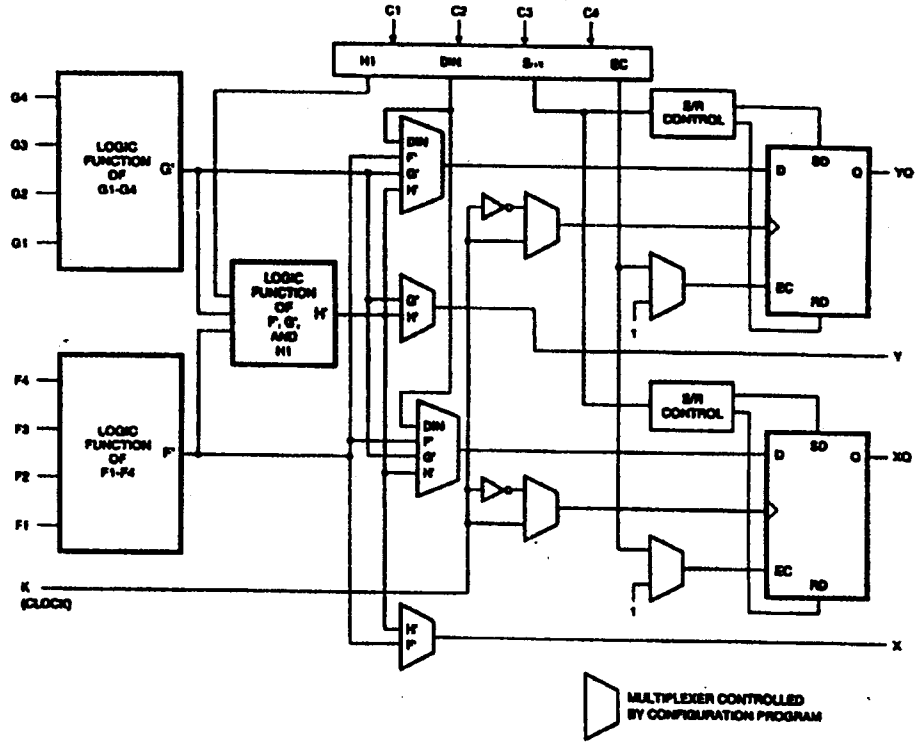


8.12. ábra

Az ábrákon a **trapéz alakú szimbólumok multiplexereket** ábrázolnak, melyek szintén programozhatók. A 4000-es sorozat CLB-je két darab 4 bemenetű és egy hárombemenetű kombinációs logikai hálózatot tartalmaz, amellyel bármely két négy-változós, bármilyen 1 ötváltozós logikai függvény realizálható. A C1....C4 kiválasztó egység a következő jeleket állítja elő:

- EC (órajel engedélyezés)
- S/R (aszinkron beírás/törlés)
- DIN (adat bemenet)
- H1 a második szintű kombinációs hálózat bemeneti jele.

A CLB két DFF-ot tartalmaz. A két kombinációs hálózat kimenete egymástól függetlenül megjelenhet a CLB kimenetén (X, Y), de ezek a jelek a DFF révén sorrendi funkciók megvalósítására is alkalmasak. Az S/R bit révén a két DFF aszinkron módon beírható vagy törölhető. A CLB-k bemeneti blokkjai (F, G) RAM memóriaként ill. aritmetikai modulként is konfigurálhatók. Összehasonlításul néhány gyakori digitális rendszertechnikai funkció megvalósításának adatait adjuk közre (8.4. táblázat).



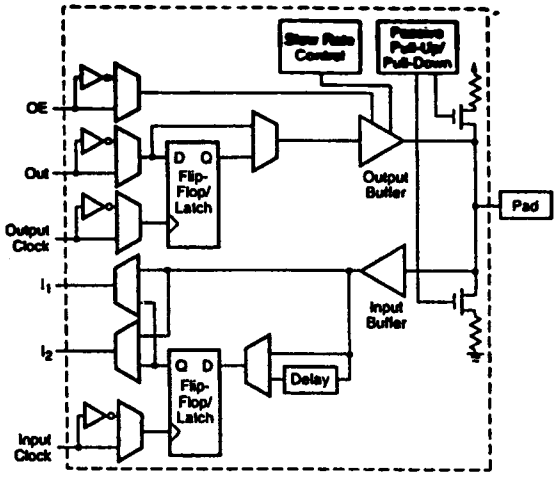
8.13. ábra

8.4. táblázat

	XC 3000		XC 4000	
24 bites accumulátor	17 MHz	46 CLB	32 MHz	13 CLB
16/1 Multiplexer	16 ns	8 CLB	16 ns	5 CLB
16 bites előre/hátra számláló	30 MHz	27 CLB	40 MHz	8 CLB
16 bites összeadó	30 ns	41 CLB	20 ns	9 CLB

Az XC 4000 sorozat I/OB felépítése látható a 8.14. ábrán.

Az I/OB a korábbi be/ki funkciók mellett azokat az eszközöket is tartalmazza, amelyek a **peremfigyeléses tesztelés** végrehajtásához szükségesek (ld. később). Az ábrából kitűnik, hogy a kimenet akár **kombinációs**, akár **regiszteres lehet**. A kivezetések felhúzó vagy lehúzó ellenállással is konfigurálhatók. Emellett a **kimeneti meghajtó áramkör gyors vagy lassú lefutásúra programozható. A lassú lefutás a zajok csökkentése szempontjából előnyös.** A bemenet szintén kombinációs vagy regiszteres lehet. A regiszteres bemenet esetén **szintvezérlésűre** vagy **élvezérlésűre** egyaránt programozható. A XC 4000-es fejlettebb elemei (a 4005-től fölfelé) külön dekódoló egységet tartalmaznak a nagyobb dekódolási feladatok ellátására.



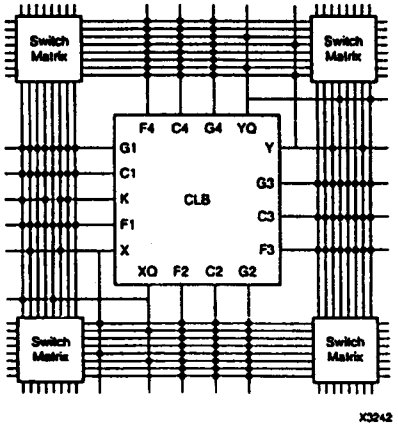
8.14. ábra

Az FPGA eszközök harmadik fő eleme a **huzalozási mátrix.** A chipen belül az összes összekötést a különböző hosszúságú huzalszakaszok és azokat a programtól függően összekapcsoló **programozható kapcsolók** alkotják. A vezetékek három csoportba sorolhatók.

a) **Általános célú egy ill. két blokknyi távolságot összekötő vezetékszakaszok.** Hosszabb útvonalak kialakítása ilyen vezeték elemekkel a késleltetések miatt nem célszerűek.

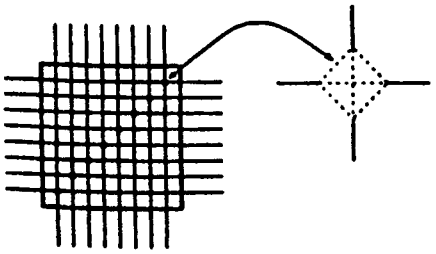
b) **Hosszú vezeték szakaszok**, amelyek a chip teljes szélességében és hosszúságában felhasználhatók.

c) **Globális vezetékek**, melyek az órajelek és az időzítések szempontjából fontos jelek szétosztására használatosak. Négy darab ún. elsődleges globális vezeték és 4 ún. másodlagos globális vezeték áll rendelkezésre. A CLB-k és az egyszeres hosszúságú vezetékek kapcsolatát a 8.15. ábra szemlélteti.



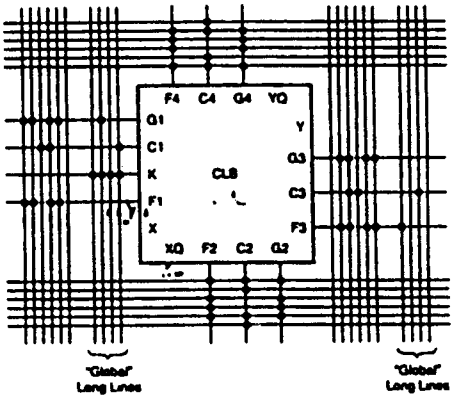
8.15. ábra

Egy **kapcsolómátrix** (Switch Mátrix) felépítése és lehetséges útvonala látható a 8.16. ábrán.



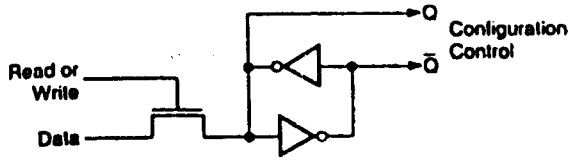
8.16. ábra

Hosszú vezetékezést szemléltet a 8.17. ábra.



8.17. ábra

A kapcsolómátrixok programozható n csatornás áteresztő tranzisztorokból épülnek fel. Két vezeték metszéspontjában 6 db. áteresztő tranzisztort helyeztek el. Az FPGA eszközök programozása (konfigurálása) azt az eljárást jelenti, amikor a megfelelő adatokat (0 vagy 1) betöltjük a CLB-k, IOB-k és a huzalozást kapcsoló statikus RAM cellákba. Az XC 4000-es család tagjainál átlagosan 300 bitet igényel egy CLB-re a hozzákapcsolódó huzalozást figyelembe véve. Minden egyes konfigurációs bit egy-egy statikus RAM cella állapotát definiálja. A CMOS RAM cella felépítése a 8.18. ábra szerinti.



8.18. ábra

Az XC 4000 hat féle módon konfigurálható:

- **mester soros** üzemmód
- **szolga soros** üzemmód
- **mester párhuzamos soros** üzemmód (bájtos fel ill. le)
- **szinkron periféria soros** üzemmód
- **aszinkron periféria soros** üzemmód.

Mester üzemmódban az FPGA maga állítja elő a cím és vezérlőjeleket, amelyek alapján egy bit vagy bájtszervezésű ROM-ból az információ betöltődik az FPGA-ba. **Bájtos üzemmódban** a betöltés kezdődhet a felső ill. az alsó címtől kezdődően. A **három** féle mester üzemmód tehát: **bájtos fel, bájtos le** és **soros.**

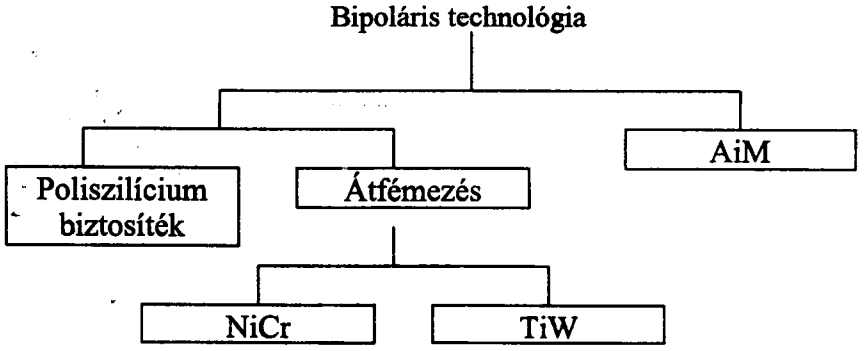
Soros szolga üzemmódban az FPGA a soros konfigurációs adatot a kívülről jövő órajellel időzítve egy **soros EPROM-ból** kapja. A XILINX ehhez illeszkedő soros EPROM-ot gyárt. Mint már említettük az FPGA elemekből a konfigurációs adatok kiolvashatók az elem működése közben, annak megzavarása nélkül. Ez a **tesztelést, ellenőrzést** nagy mértékben növeli.

8.5. A programozhatóságot biztosító eszközök

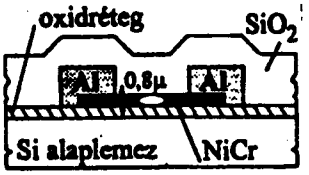
A programozható logikai eszközökben a programozhatóságot speciális áramköri megoldású kapcsolók biztosítják. Ezen kapcsolók beállítása az eszköz lábain keresztül villamos úton valósítható meg. A **kapcsolók állapotának beállítási folyamatát tekintjük programozásnak**. Attól függően, hogy ezen kapcsolók beprogramozott állapotukat a tápfeszültség után megőrzik vagy nem, beszélünk **volatile (felejtő)** ill. **nonvolatile (nem felejtő)** típusokról. Előbbiek RAM tárral, utóbbiak ROM tárral rendelkeznek. Megkülönböztetünk **egyszer programozható** ill. **többször programozható elemeket.**

8.5.1. Bipoláris eszközök

A bipoláris programozható eszközök az alábbiak szerint csoportosíthatók:

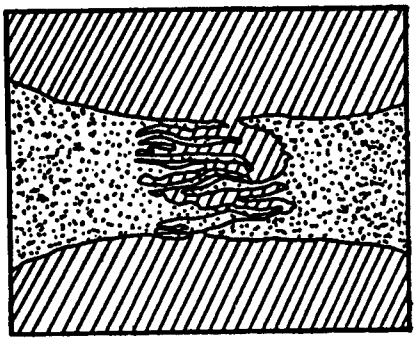


A legrégebbi megoldás a **nikkel-króm (NiCr) biztosítékok** használata. A NiCr biztosíték felépítését a 8.19.a) ábra szemlélteti.



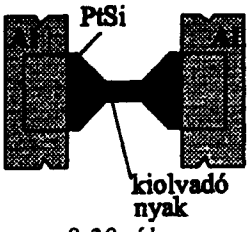
8.19.a) ábra

A szilícium alaplemezre először oxidréteget visznek fel, majd ezen alakítják a néhány nm vastagságú nikkel-króm réteget. A csatlakozások alumíniumból készülnek. Programozás során a megfelelő áramerősségű (20-50 mA) és időtartamú (10-20 ms) impulzus hatására a NiCr film egy része kiolvad. Egy kiolvadt biztosítékot szemléltet a 8.19.b) ábra, amely fotó alapján készült.



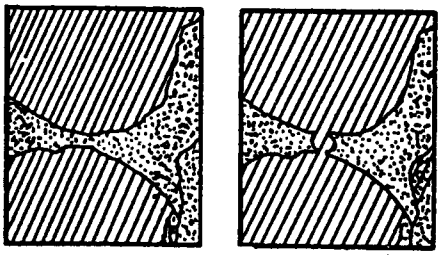
8.19.b) ábra

Az ábrából kitűnik a módszer hátránya: a **kiégetés nem tökéletes,** nagy az „újra visszanövés" veszélye, amely ismét rövidzárat okozhat. Ezért egyes gyártók a Ni-Cr fémfilm helyett **titán-wolfrám** (Ti-W) ötvözetet használnak. Ezek geometriai szerkezete gyakorlatilag megegyezik a Ni-Cr biztosítékéval. A **polikristályos szilícium elemi biztosíték** (Silicon fuse) vezetőképessége az adalék anyag mennyiségével állítható be a kívánt értékre. A programozás során átfolyó áram az ún. kioldó nyakon (8.20. ábra) 1400 ^{o}C körüli hőmérsékletet hoz létre, ami **oxidálja** és ezáltal **szigetelővé teszi** a szilíciumot. Az eszköz **egyszer programozható.**



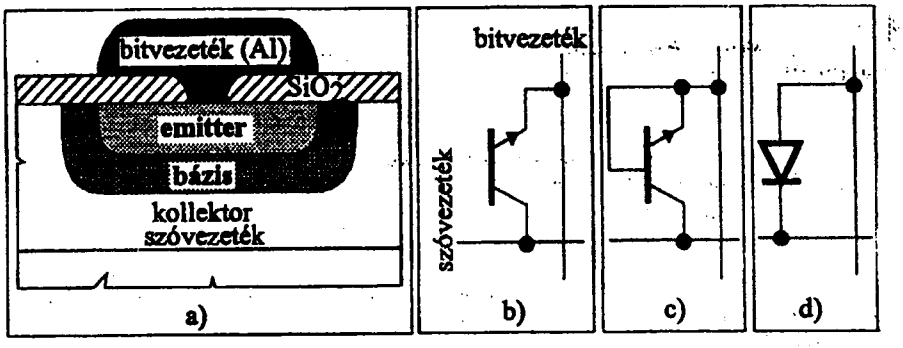
8.20. ábra

Egy szilícium biztosítékot mutat a 8.21. ábra programozás előtt és után. A szilícium biztosíték előnye, hogy a **visszaéledés veszélye szinte kizárt.** Hátránya viszont a **nagyobb programozó áram.**



8.21. ábra

Az **AIM technika** az ún. vertikális programozható technikán alapul. Működésük abban áll, hogy a **pn átmenetet lavina átütés** révén vezetővé teszik. A lavina átütéssel egyidejűleg az anyagrészecskék vándorlása következik be a pn átmenetek között. Innen az elnevezés: AIM = Avalanche Induced Migration (lavina indukált vándorlás). Programozáskor fordított polaritású (30 - 40 V) és viszonylag nagy áramerősségű (kb. 100 mA) áramot juttatunk a tranzisztorra, aminek hatására a **rövidzár** kialakul. Figyeljük meg, hogy **az előző módszerekkel szakadást, az AIM révén rövidzárt** hoznak létre. Egy tipikus AIM kapcsolóelem és helyettesítő kapcsolása látható a 8.22. ábrán.



8.22. ábra

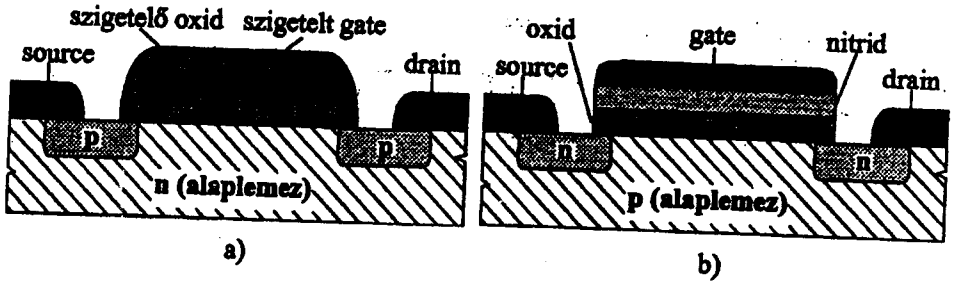
Az ábra szerint egy AIM kapcsolóelemet egy olyan npn tranzisztor valósít meg, amelynek bázisa nincs kivezetve, s a vezető vagy nem vezető állapot a tranzisztor emittere és kollektora között hozandó létre. A b) ábra a programozás előtti állapotban szemlélteti a tranzisztort a c) d) ábrák pedig a programozás után. Megjegyezzük, hogy a bipoláris technikájú eszközöknél a programozás viszonylag hosszabb ideig tart, mivel az **égetésnél fellépő hőmérséklet növekedés a chip felmelegítésével jár.** Ezért a programozás csak bitenként történik és rendszerint a **chip hűtéséről** is gondoskodni kell.

8.5.2. MOS programozható elemek

A MOS alapú eszközök programozható elemei négy csoportba sorolhatók:

- a FAMOS tranzisztorral megvalósított eszközök
- az MNOS tranzisztorral megvalósított eszközök
- az antifuse elemmel megvalósított eszközök
- statikus RAM eszközök.

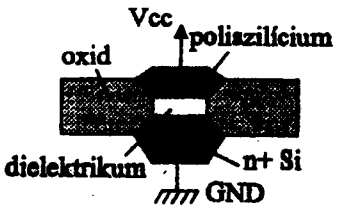
A FAMOS tranzisztort a 6. fejezetben bemutattuk, a 8.23.a) ábrán csak az MNOS tranzisztorral való összehasonlítás céljából ismertetjük. Az **MNOS tranzisztornál** a vezérlő elektróda ki van vezetve, de a vezérlőelektróda és a félvezető réteg között **kétszeres szigetelő réteg** található: egy **vastagabb szilíciumnitrid** és egy **vékonyabb szilíciumoxid** réteg. A két szigetelőréteg határfelületén villamos töltések halmozódnak fel a programozás során és ott **hosszú ideig tárolódnak.** Ezek a felhalmozódott töltések vezető MOS tranzisztort hoznak létre. A programozó feszültséget a gate és az alaplemez (szubsztrát) közé kapcsolják, aminek hatására **alagúteffektus** révén a pozitív elektróda (gate) irányában elektronok lépnek be a szilíciumnitrid rétegbe, ahol felhalmozódnak és tárolódnak.



8.23. ábra

Az MNOS tranzisztorok esetén a **törlés elektromos úton hajtható** végre úgy, hogy a beírással ellentétes feszültséget kapcsolnak az eszközre. Ennek hatására nem az elektronok, hanem a lyukak fognak vándorolni a tároló szigetelő rétegbe.

Az **„antifuse"** elem felépítése a 8.24. ábra szerinti. Ez tulajdonképpen ellentéte a poliszílicium biztosítéknak (innen a neve: antifuse biztosíték). Programozáskor ezeket nem kiégetik, hanem ellenkezőleg vezetővé teszik. Az antifuse elemek poliszílícium és n^+ szennyezettségű szilícium vezető rétegek között kialakított szigetelő réteggel valósíthatók meg. Megfelelő feszültség hatására a dielektrikum átüt és létrejön a vezetés. A dielektrikum ellenállása programozás előtt 100 MΩ, programozás után 500 Ω körüli értékű. A programozás **irreverzibilis**, tehát **nem újraprogramozható** az eszköz. Előnye, hogy ez a legkisebb helyigényű eszköz.



8.24. ábra

A statikus MOS RAM cellákat a 6. fejezetben bemutattuk. Ezeket többnyire az FPGA áramkörökben alkalmazzák.

8.6. Programozható eszközök összehasonlítása

Korábban a programozható logikai eszközök összehasonlítására az **ekvivalens kapu számát** használták. A programozható logikai áramkörök teljesítőképességének jellemzésére szolgáló mutatókat **benchmark**-oknak nevezik. Az előírások szerint 9 tipikus funkciót definiálnak és ezek alapján 9 benchmarkot határoznak meg. A 9 logikai funkció a következő:

1. Egy MUX-ot, egy közönséges és egy léptető regisztert tartalmazó adatút
2. Időzítő, számláló funkció
3. Egyszerű állapotvezérlő
4. Bonyolult állapotvezérlő
5. Szorzást és összeadást tartalmazó aritmetikai funkció
6. 16 bites akkumulátor
7. 16 bites szinkron számláló
8. 16 bites skálázott számláló
9. Speciális busz-hiba detektáló funkció.

Az adott funkcióhoz tartozó benchmark első eleme azt mutatja meg, hogy az adott funkciót hányszor lehet az illető eszközökben megvalósítani, azaz hány példány fér el az eszközben (pl. 16 bites szinkron számláló). Ez tehát az eszköz **logikai kapacitását** jellemzi. Az adott funkcióhoz tartozó benchmark második elemeként azt adják meg, hogy az eszköz logikai elemeinek hány százalékát lehetett felhasználni. Ez tehát az **útválasztással** kapcsolatos képességet jellemzi. A benchmark következő három eleme az eszköz belső működési sebességét jellemzi. A belső működési frekvenciát a következő két késleltetés hossszabbikának reciprokával állítják elő:

- a leghosszabb út az egyik példány FF-jától a másik példány FF-jáig
- a leghosszabb út egy példány két FF-ja között.

A benchmarkok utolsó eleme a külső működési frekvenciát jellemzi az adott funkció esetén.
Példaként a XILINX 4010 elemre adjuk meg a benchmark értéket az aritmetikai funkcióra.

	Példány (db)	Hányad %	Legrosszabb	Legjobb	Átlag %	Külső %
XC 4010	25	100	82	89	82	70

8.7. Digitális rendszerek tervezése programozható eszközökkel

A programozható logikai áramkörökből felépülő digitális rendszerek tervezési módszere nagy mértékben hasonlít a huzalozott logika esetén használatoshoz. A programozható logikai áramkörök szintézisének lépései:

1. logikai függvények minimalizálása
2. technológiai leképzés
3. elhelyezés és útválasztás (routing) meghatározása.

A tervezés fenti feladatai közül az elsőt minden típusú eszköz esetén el kell végezni, míg az utóbbi kettőt csak az FPGA eszközöknél és a PLD-k egyes típusainál kell eszközölni. Hangsúlyozni kell, hogy a fenti műveleteket nem a tervezőnek kell megoldani, ezt elvégzik a rendelkezésre álló **CAD eszközök.** Megjegyezzük, hogy a programozható eszközökhöz kidolgozott CAD tervezőrendszerekben az alkalmazott struktúrákhoz (PAL, GAL, PLD, FPGA) **illeszkedő minimalizálási módszereket** dolgoztak ki, különösen vonatkozik ez az FPGA elemek esetén, ahol az egyes logikai funkciókat **többszintű hálózatokból kell** felépíteni. A technológiai leképzés szerepe, hogy a CAD által meghatározott logikai függvényeknek egy olyan hálózatba történő **leképzése,** amelyek az adott rendszer (PAL, GAL, FPGA) szempontjából **optimálisnak** tekinthetők. A **technológiai leképzés,** azaz **lay-out tervezés,** amely az összekötő vezetékeket még nem tartalmazza. A technológiai leképzés egyik módszere a **könyvtár alapú technológiai leképzés.** Ennek végeredménye rendszerint többszintű NAND hálózat. Az FPGA-k esetén egyrészt a multiplexer bázisú CLB-k ill. a XILINX típusú CLB-k lay-out-jának a meghatározása a szempont. Ehhez a XILINX **macro könyvtárral** rendelkezik. A logikai minimalizálás és a technológiai leképzés után rendelkezésre áll a **logikai komponensek hálózatának listája (component net list).** Ezután két feladatot kell végrehajtani:

- a logikai komponenseket hozzá kell rendelni a felhasználandó FPGA fizikai komponenseihez
- létre kell hozni a felhasznált FPGA komponensei között a **szükséges fizikai összekötéseket** (a tulajdonképpeni huzalozást).

Ez a folyamat két lépésből áll:

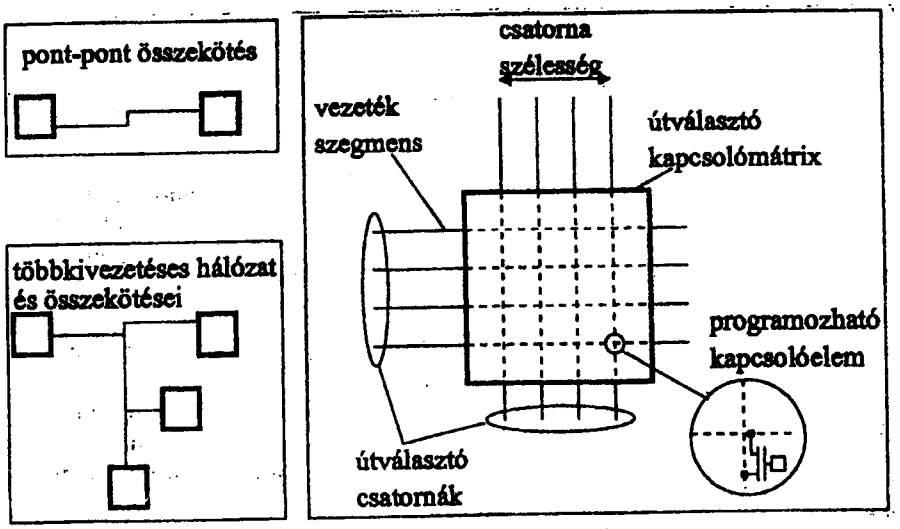
- elhelyezés (placement) és
- útválasztás (routing).

Az **elhelyezés és útválasztás egymással kölcsönösen összefüggő folyamat.** Ennek optimális megoldása rendszerint **iterációs lépések** alapján hozható létre.

Az útválasztás meghatározásánál a CAD tervezésénél az alábbi fogalmakkal találkozunk:

- **Kivezetés** (pin): a CLB-k és IOB chipen belüli be- és kimenetei.
- **Összekötés** (connection): két galvanikusan összekötött kivezetés.
- **Hálózat** (net): elektromosan összekötött kivezetések csoportja.
- **Útválasztó kapcsoló** (routing switch): programozható kapcsoló, amely két vezeték szegmens összekötésére használható.
- **Vezeték szegmens** (wire segment): a vezetéknek egy egyenes szakasza, amelyet az összekötés kialakítására használunk.
- **Útvonal** (track): az útválasztó csatorna teljes szélességét vagy hosszát befutó egyenes vezeték szakasz.
- **Útválasztó csatorna** (routing chanel): téglalap alakú terület, amely a logikai blokkok két sora vagy két oszlopa között fekszik.

Fenti fogalmakat szemlélteti a 8.25. ábra.



8.25. ábra

8.8. ASIC CAD eszközök

Az ASIC eszközök programozását **CAD tervezőrendszerek** segítik, melyek az **áramköri architektúrához** illeszkednek. Ezen tervező rendszerek (szoftverek) állandó fejlődésen mennek keresztül, miáltal egyre több szolgáltatást végeznek. A tervezés végcélja az ún. **„biztosíték térkép"** elkészítése, amely letölthető az ASIC áramkörbe. Ezen CAD rendszerek alapvetően **kétféle filozófia** szerint épülnek fel, úgy mint **eszközfüggő tervezés,** ill. **eszközfüggetlen tervezés.** Az első esetben a felhasználó **megadja** az ASIC eszköz típusát, így a CAD rendszer **erre az eszközre orientált megoldást** valósít meg. **Az eszköz független tervezés esetén, a tervező nem specifikálja a konkrét típust,** hanem csak a

feladatot és CAD rendszer funkciója azon eszközök meghatározása, amelyekkel a specifikált feladat megoldható.

8.8.1. Az FPLA, PAL, GAL, SPLD tervezőrendszerek

A fenti áramkörök CAD rendszerei eltérnek az FPGA tervező rendszerek architektúrájától. A legalapvetőbb kérdés a feladat gépi leírása. E célra különböző hardver leíró nyelvek állnak rendelkezésre. Ilyenek:

- ABEL (Advanced Boolean Expression Language)
- PALASM (PAL Assembler)
- PLD ASM (PLD Assembler)
- AHDL (Altera Hardware Design Language).

Egyes CAD rendszerek a VLSI IC-k tervezésére kidolgozott **VHDL** (Very High Speed integrated circuit Hardware Design Language) nyelvű leírást is elfogadják. A **feladat bevitele** a CAD rendszerbe történhet **szöveges ASCII fájl** formájában, ill. **logikai kapcsolási rajz** alapján. A tervező rendszerek lehetőséget nyújtanak bizonyos optimalizálási, szimulációs, verifikációs és hibadiagnosztikai funkciók ellátására. A tervezés végeredménye az ún. **JEDEC fájl** (Joint Electronic Devices Engineering Consil). Ez egyrészt a **biztosíték térképét,** másrészt a **fizikai teszteléshez szükséges információkat** tartalmazza a (JEDEC szimuláció).

8.8.2. FPGA alapú CAD rendszerek

Az FPGA eszközök tervezése sokkal inkább **eszközfüggő,** mint az SPLD eszközöké. Ez azt jelenti, hogy a XILINX fejlesztő rendszer nem használható más típusú FPGA-k programjának elkészítéséhez. A tervezés főbb lépései:

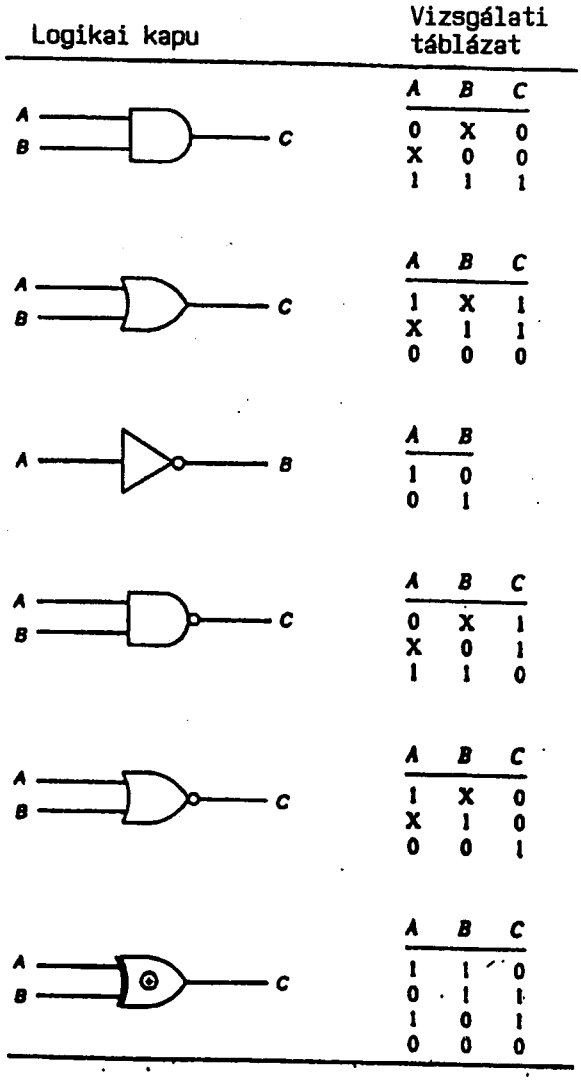
- technológiai leképzés
- elhelyezés (placement)
- útválasztás (routing)
- szimuláció.

A tervezés végeredménye a **konfigurációs fájl** előállítása, amely az FPGA RAM ill. EPROM memóriájába letölthető.

8.9. Diagnosztika és tesztelés

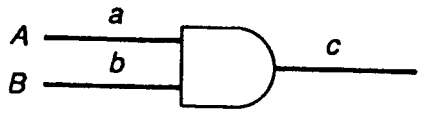
Az előzőekben láttuk, hogy az LSI, VLSI, GLSI technika egyre több és több logikai elemet tartalmaz, az áramkörök egyre bonyolultabbá válnak. Ez egyre inkább igényli az **áramkörök tesztelhetőségének** kérdését, másrészt pedig lehetőséget nyújtanak ún. **hibatoleráns redundáns rendszerek** kialakítására. A diagnosztika feltétele a tesztelhetőség szempontjából a **megfigyelhetőség** és a **vezérelhetőség.** Egy digitális rendszer megfigyelhetőnek mondható, ha valamennyi pontján fellépő logikai érték megfigyelhető. A **vezérelhetőség** pedig azt jelenti, hogy a rendszer bármely csomópontja a megkívánt állapotba vihető, azaz vezérelhető. Ha egy digitális rendszer nem vezérelhető vagy csak

részben - akkor a szóban forgó rendszer nem vagy csak részben tesztelhető. Ez azt jelenti, hogy a rendszer hibái nem detektálhatok. **Nem megfigyelhető** rendszerre példaként az egyik bemenetén állandóan „1"-re kapcsolt VAGY kaput említjük. mivel ez esetben a másik bemeneten megjelenő 0 ill. 1 jel hatása nem figyelhető meg a kimeneten. **Vezérelhetetlen** például két olyan sorba kötött inverterből álló kapcsolás, amelynek kimenete vissza van csatolva a bemenetre. Mivel így az áramkörnek nincs vezérelhető bemenete, ezért vezérelhetetlen. Az alapkapuk megfigyelhetőségét és vezérelhetőségét szemlélteti a 8.26. ábra.



8.26. ábra

Egy ÉS hálózat megfigyelését a 8.27. ábra szerint végezhetjük el. Egy hálózatban a 0-hoz ill. 1-hez történő zárlat ("stuck-at-zero" és "stuck-at-one") vizsgálatához, a kapuk analíziséhez figyelembe kell venni a hozzávezetések állapotát is. Vizsgáljuk meg egy két bemenetű És kapu (8.27. ábra) esetére vonatkozó ajánlásokat.



8.27. ábra

$$P_a = A\&a_n \vee a_1 .$$

Ez azt jelenti, hogy a kapu A bemenetén akkor lesz 1 jel, ha A = 1 és az a vezeték nem szakadt (1) vagy az a vezeték = 1, azaz zárlatba van az U1-gyel. Ugyanígy írhatjuk az

$$\overline{P_a} = \overline{A}\&a_n \vee a_o$$

ami azt jelenti, hogy a kapu A bemenetén akkor lesz 0 jel, ha A = 0 és a vezeték hibátlan vagy amikor az a vezeték földzárlatban van. Hasonlóképpen írhatjuk a B bemenetre és a C kimenetre a következő egyenleteket.

$$P_b = B\&b_n \vee b_1$$

$$\overline{P_b} = \overline{B}\&b_n \vee b_o$$

$$P_c = \left(P_a \& P_b\right) \cdot C_n \vee C_1$$

$$\overline{P_c} = \left(\overline{P_a} \vee \overline{P_b}\right) C_n \vee C_o .$$

Fenti ajánlásokat a 8.28. ábrán foglaltuk össze az alapkapukra.

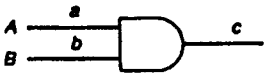
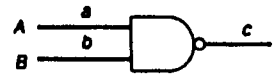
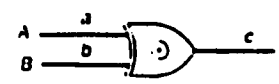
Az **áramkörök tesztelésénél** a Boole differenciák is jól használhatók. Egy Y = f(X) logikai függvény Boole differenciája.

$$\frac{dY}{dX} = Y_X(X = 0) \forall Y_X(X = 1)$$

ahol $Y_X(0)$=Y értéke, ha X = 0, és $Y_X(1)$=Y értéke X = 1 esetén. Ha dY/dX = 1, akkor a $Y_X(0)$ helyen és $Y_X(1)$ helyen különböző. Így matematikailag definiálható a tesztelés:

$\overline{X}\frac{dY}{dX}=1$ teszt az 1 – es kimenethez

$X\frac{dY}{dX}=1$ teszt a 0 – ás kimenethez.

Logikai kapu	Függvények
	$P_a = Aa_n + a_1$ $P'_a = A'a_n + a_0$ $P_b = Bb_n + b_1$ $P'_b = B'b_n + b_0$ $P_c = P_aP_bc_n + c_1$ $P'_c = (P'_a + P'_b)c_n + c_0$
	$P_a = Aa_n + a_1$ $P'_a = A'a_n + a_0$ $P_b = Bb_n + b_1$ $P'_b = B'b_n + b_0$ $P_c = (P_a + P_b)c_n + c_1$ $P'_c = (P'_aP'_b)c_n + c_0$
	$P_a = Aa_n + a_1$ $P'_a = A'a_n + a_0$ $P_b = P'_ab_n + b_1$ $P'_b = P_ab_n + b_0$
	$P_a = Aa_n + a_1$ $P'_a = A'a_n + a_0$ $P_b = Bb_n + b_1$ $P'_b = B'b_n + b_0$ $P_c = (P'_a + P'_b)c_n + c_1$ $P'_c = (P_aP_b)c_n + c_0$
	$P_a = Aa_n + a_1$ $P'_a = A'a_n + a_0$ $P_b = Bb_n + b_1$ $P'_b = B'b_n + b_0$ $P_c = (P'_aP'_b)c_n + c_1$ $P'_c = (P_a + P_b)c_n + c_0$
	$P_a = Aa_n + a_1$ $P'_a = A'a_n + a_0$ $P_b = Bb_n + b_1$ $P'_b = B'b_n + b_0$ $P_c = (P_aP_b + P'_aP'_b)c_n + c_1$ $P'_c = (P_aP_b + P'_aP'_b) + c_n + c_0$

8.28. ábra

Vizsgáljuk meg a fentieket a NOR kapu esetére:

$$Y = \overline{A \vee B}$$

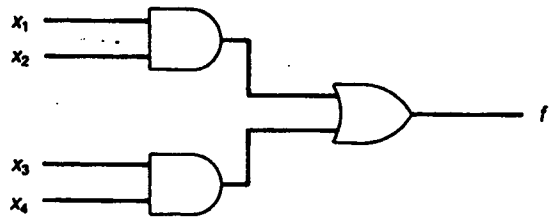
$$Y_A(A = 0) = \overline{B}$$

$$Y_A(A = 1) = 0.$$

Tehát

A dY/dA tehát akkor lesz 1, ha B = 0. E szerint a két bemenetű NOR kaput akkor tudjuk tesztelni, ha az egyik bemenete 0-ára van kötve és a másik bemenetet teszteljük.

Vizsgáljuk meg a Boole differenciák alkalmazását egy ÉS/VAGY hálózat esetére (8.29. ábra).



8.29. ábra

$F = X_1 \,\&\, X_2 \vee X_3 \,\&\, X_4.$

A vizsgálatokat az X_1 bemenetre végezve:

$$F_{X1}(0) = X_3 \& X_4$$

$$F_{X1}(1) = X_2 \vee X_3 \& X_4$$

$$\frac{dF}{dX_1} = F_{X1}(0) \forall F_{X1}(1) = X_2 \& \overline{X}_3 \vee X_2 \& \overline{X}_4$$

$$X_1 \frac{dF}{d_{X1}} = X_1\left(X_2 \& \overline{X}_3 \& X_2 \& \overline{X}_4\right) = X_1 \& X_2 \& \overline{X}_3 \vee X_1 \& X_2 \& \overline{X}_4 = 1.$$

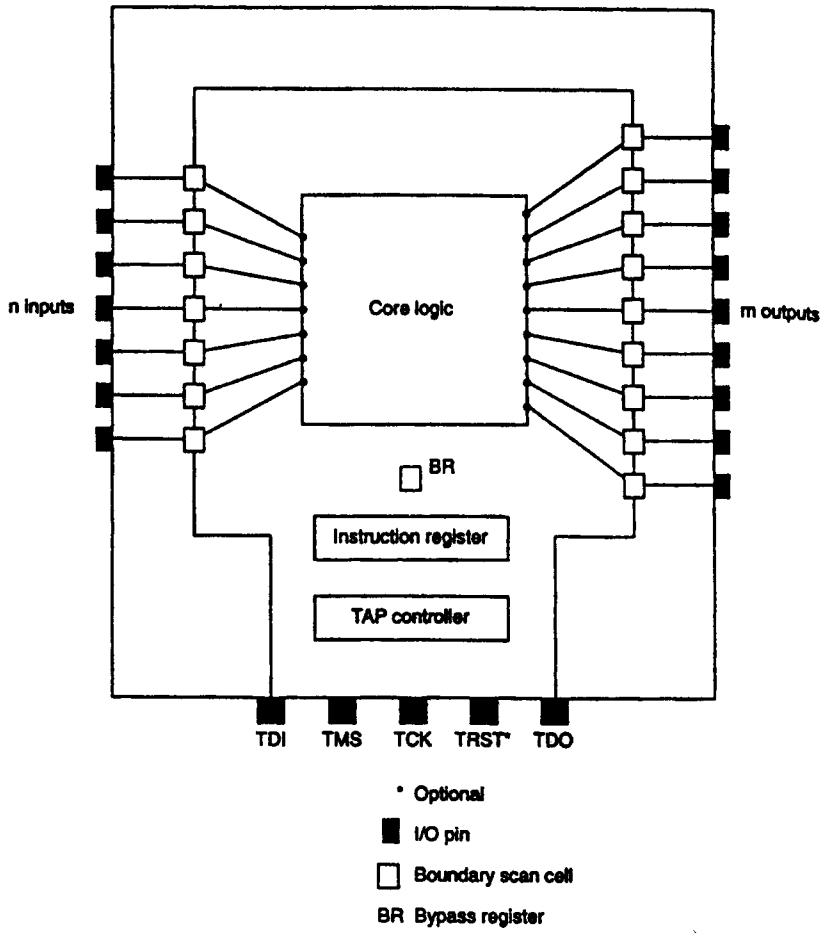
Ez a kimenet 0-ás jelének X_1-től függőség vizsgálatához az alábbi bitmintákat jelenti: 1100, 1101, 1110.

$$\overline{X}_1 \frac{dF}{d_{X1}} = \overline{X}_1\left(X_2\,\overline{X}_3 \vee X_2\,\overline{X}_4\right) = \overline{X}_1 X_2\,\overline{X}3 \vee \overline{X}_1 X_2\,\overline{X}_4 = 1.$$

Az 1-es kimenethez tartozó bitminták: 0100, 0101, 0110.

8.10. Peremfigyeléses tesztelés (Boundary Scan)

A **peremfigyeléses tesztelés** filozófiája az, hogy egy chip valamennyi be- és kimenetét egy **megfigyelési láncba** vonjuk be, amelyet **peremfigyelési láncnak** hívunk. A módszer a kártyák és alrendszerek vizsgálatára is kiterjeszthető. Egy peremfigyeléses logikával ellátott integrált áramkör felépítését a 8.30. ábrán láthatjuk. Ezen látható, hogy a chip **funkcionális logikáján** (core logic) egy **boundary scan logikát** is tartalmaz. Ez abban nyilvánul meg, hogy a chip tápfeszültség és földcsatlakozó lábian kívül **minden lábra egy peremfigyelő cella** csatlakozik, amelyekbe **egymással sorosan és párhuzamosan** is bevihetők az adatok, illetve a láncból a környezetbe hasonló módon vihetők ki az adatok.

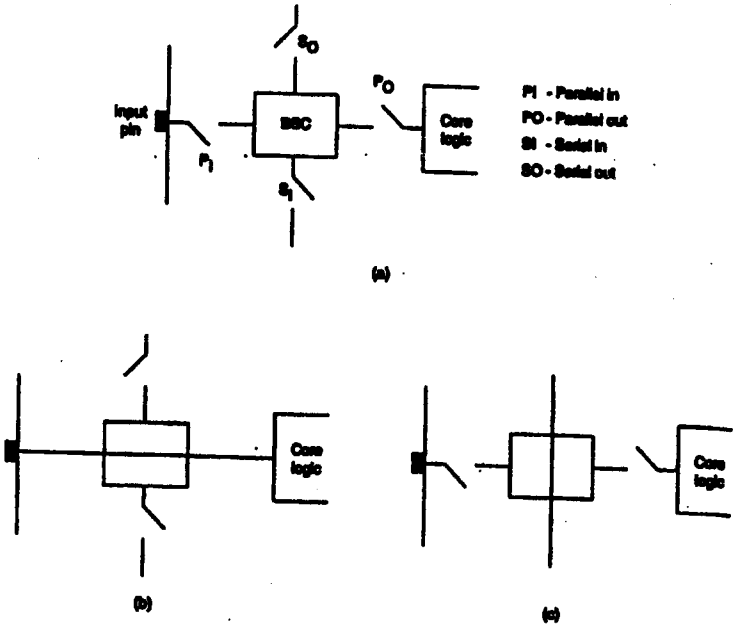


8.30. ábra

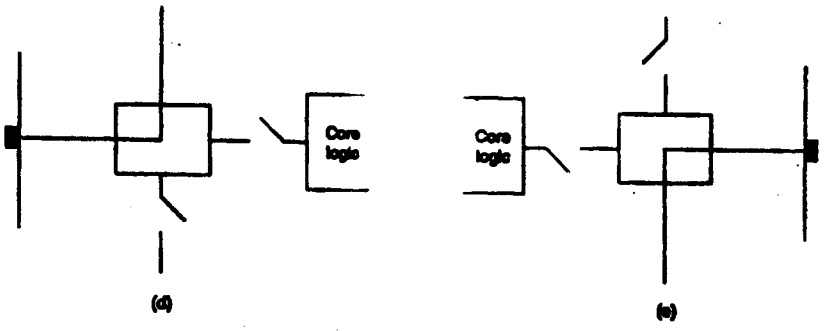
A peremfigyeléses logika tehát a **lábankénti peremfelügyelő cellákból**, egy **vezérlő áramkörből** (TAP controller: Test Access Port) egy **utasítás regiszterből** és egy **bypass regiszterből** (BP) áll. A TAP controller egy **16 belső állapottal rendelkező vezérlő**, amely funkcionálisan **négy jellel** kapcsolódik a külvilághoz. Ezek:

- TDI (Test Data In) **teszt adat bemenet**, amely a teszt adatoknak és az utasításoknak a bevitelére szolgál.
- TDO (Test Data Out) **teszt adat kimenet**, amely a teszt adatok kiléptetésére szolgál.
- TMS (Test Mode Select) **teszt üzemmód kiválasztás**, amely a chipen végrehajtandó különböző teszteket vezérli.
- TCK (Test Clock) **Tesztelési órajel**, amely lehetővé teszi, hogy a peremfigyeléses tesztet a rendszerórajeltől függetlenül lehessen végrehajtani.

A TAP **ötödik bemenete** a TRST (Test Reset), amely a TAP vezérlő alapállapotba hozatalára szolgál (törlés). A peremfigyeléses módszer belső és külső tesztelésre egyaránt alkalmas. A **belső teszt** (INTEST) a **chipen elhelyezett logika tesztelésére** szolgál. A **külső teszt** a kártyán elhelyezett alkatrészek közötti összekötéseket ellenőrzi. Így megvizsgálható pl. két chip közötti összekötés. Egy boundary scan cella **négy kapcsolóját** a 8.31.a) ábra szemlélteti. A b) ábra **párhuzamos be- párhuzamos ki**, a c) **soros be-soros ki**, a d) **párhuzamos be- soros ki** és az e) **soros be- párhuzamos ki** üzemmódot szemléltet.

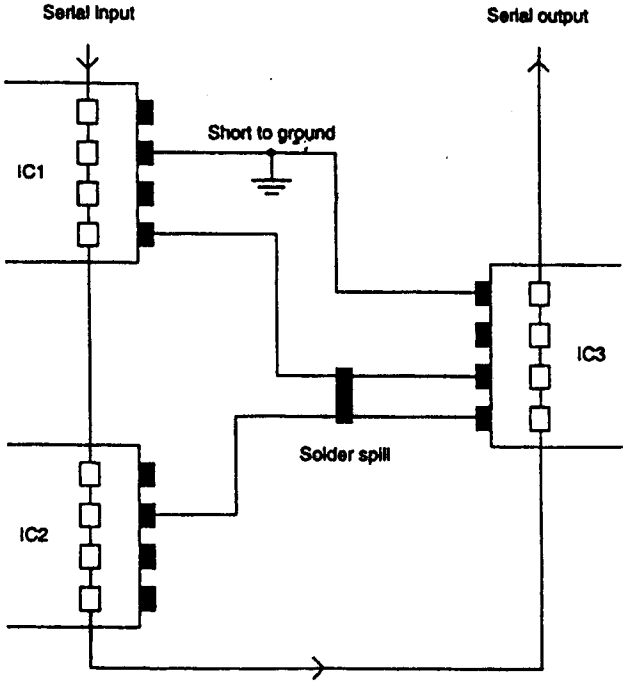


8.31.a)b)c) ábra



8.31.d)e) ábra

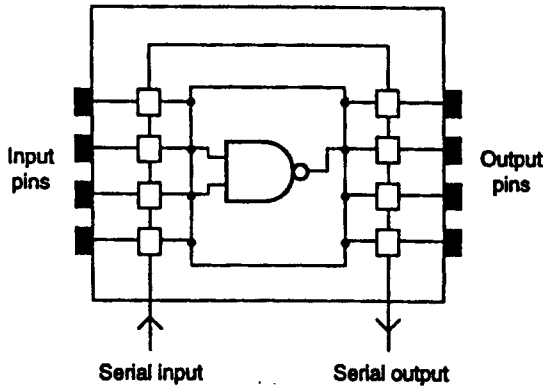
A peremfigyeléses tesztelés egyik igen előnyös adottsága a **mintavételes üzemmód,** amely lehetővé teszi, hogy a **chip jeleit üzemközben**, azok megzavarása nélkül megfigyeljük. Három chip külső peremfigyelés elvét szemlélteti a 8.32. ábra.



Input	Output	
	Expected	Actual
x1x1x0xxxxx	xxxxxxxx01x1	xxxxxxxx11x0
x0x0x1xxxxx	xxxxxxxx10x0	xxxxxxxx11x0

8.32. ábra

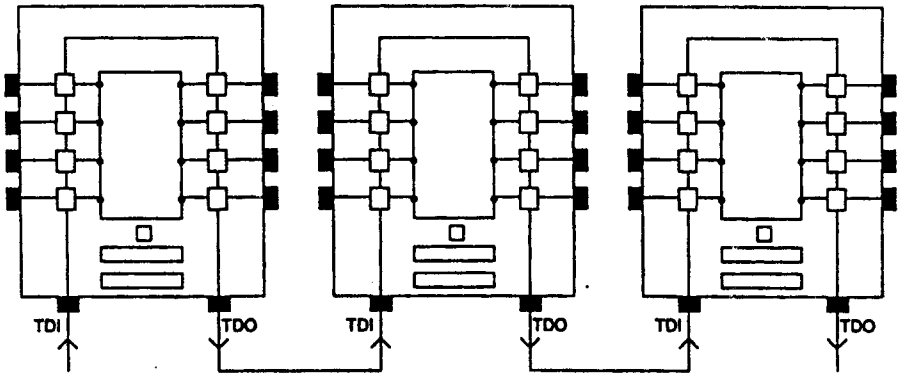
Az ábra egy **földzárlat** ill. **két láb közötti** zárlat esetén a vizsgált bitsorozatot is megadja. Egy belső NAND elem tesztelését mutatja a 8.33. ábra a beadott ill. kapott válaszjelek feltüntetésénél.



Input	Correct output
x10xxxxx	xxxxx1xx
x01xxxxx	xxxxx1xx
x11xxxxx	xxxxx0xx

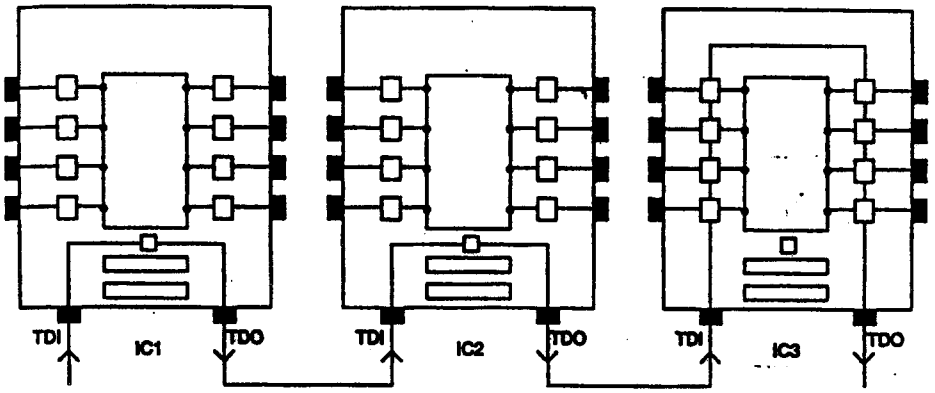
8.33. ábra

Három sorosan kötött chip peremfigyeléses elve látható a 8.34. ábrán. Figyeljük meg, hogy a regiszterláncba valamennyi BSC be van kötve.

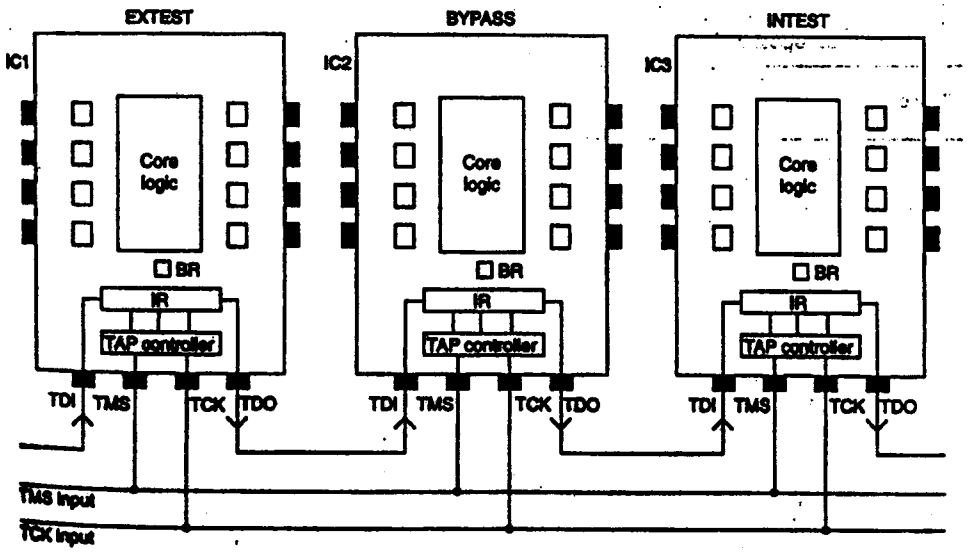


8.34. ábra

A BR regiszter lehetőséget nyújt a **sorba kötött chipek kiiktatására** A 8.35. ábrán az IC3 tesztelése az IC1 és IC2 kiiktatásával gyorsabban végezhető el. A TAP vezérlő utasítás regiszterének betöltését szemlélteti a 8.36. ábra, ahol az IC1 külső, az IC3 belső tesztelése, az IC2 pedig bypass üzemre lesz programozva.



8.35. ábra



8.36. ábra

A peremfigyeléses vizsgálat igen hatékony korszerű módszer, amely IEEE 1149 jelzéssel szabvánnyá vált Boundary Scan Test elnevezéssel. A XILINX FPGA-k ill. korszerű mikroprocesszorokban alkalmazzák (ld. PENTIUM).

Felhasznált irodalom

Ajtonyi I: Vezérléstechnika II. Tankönyvkiadó, Budapest, 1987.
Gál T: Programozható logikák BME-Tankönyvkiadó, 1994.
P. Ammon: Kapumátrix-áramkörök Műszaki Könyvkiadó, 1989.
R.C. Seals and G. F. Whapshott: Programmable logic: PLDs and FPGAs School of Engineering University of Greenwich.
The Technical Staff of Monolithic Memories, Inc.: Designing withProgrammable Array Logic ISBN 0-07-042723-2 New York, 1981.
XILINX The Programmable Gate Array Data Book San Jose, California 95124, 1991.
XILINX The Programmable Logic Data Book San Jose, California 95124, 1994.

TARTALOMJEGYZÉK

ETEM UNIVERSITY OF MISKOLC UNIVERSITÄT MISKOLC МИШКОЛЬЦСКИЙ УНИВЕРСИТЕТ L'UNIVERSITÉ DE
MISKOLCI EGYETEM UNIVERSITY OF MISKOLC UNIVERSITÄT MISKOLC МИШКОЛЬЦСКИЙ УНИВЕРСИТЕТ L'U
TÉ DE MISKOLC MISKOLCI EGYETEM UNIVERSITY OF MISKOLC UNIVERSITÄT MISKOLC МИШКОЛЬЦСКИЙ УН
IVERSITÉ DE MISKOLC MISKOLCI EGYETEM UNIVERSITY OF MISKOLC UNIVERSITÄT MISKOLC МИШКОЛЬЦСК
КИЙ УНИВЕРСИТЕТ L'UNIVERSITÉ DE MISKOLC MISKOLCI EGYETEM UNIVERSITY OF MISKOLC UNIVERSITÄT
ИШКОЛЬЦСКИЙ УНИВЕРСИТЕТ L'UNIVERSITÉ DE MISKOLC MISKOLCI EGYETEM UNIVERSITY OF MISKOLC U
MISKOLC МИШКОЛЬЦСКИЙ УНИВЕРСИТЕТ L'UNIVERSITÉ DE MISKOLC MISKOLCI EGYETEM UNIVERSITY OF