ZÁRÓVIZSGA KÉRDÉSEK

DIGITÁLIS TECHNIKA

című tantárgyhoz

Létesítmények villamosítása és automatizálása szakirány

1. Logikai hálózatok fogalma, logikai érték, logikai változó, Boole-algebra.
2. Boole-algebra, logikai alapműveletek, logikai azonosságok, logikai függvények.
3. Logikai függvények kanonikus alakjai, minterm, maxterm fogalma.
4. Logikai függvények egyszerűsítése.
5. Szimmetrikus logikai függvények. Logikai függvények realizálása.
6. Hazárdok.
7. Aszinkron sorrendi hálózatok és állapottáblája.
8. Szinkron sorrendi hálózatok és állapottáblája.
9. Elemi sorrendi hálózatok.
10. Élvezérelt, és master-slave, flip-flop-ok.
11. Frekvenciaosztók, számlálók.
12. Aszinkron, szinkron számlálók.
13. Regiszterek, kódolók, dekódolók, multiplexerek, demultiplexerek.
14. Memóriák.
15. Számábrázolási formák.
16. Mikroprocesszorok általános felépítése.
17. Busz fogalma, jellemző jelei és működésük.
18. Interrupt és fajtái.
19. Mikroprocesszor, és mikrokontroller hasonlósága, különbsége. Mikrokontroller jellemző felépítése.
20. Stack fogalma és használata.
21. Integrált áramkörök csoportosítása végső funkciójuk kialakítása szerint.
22. A diszkrét, ASIC, PLÁ áramkörök összehasonlítása.
23. Egyszerű programozható logikai áramkörök fajtái, belső felépítésük.
24. Összetett PLÁ-k, programozási lehetőségek.

1. tétel: Logikai hálózatok fogalma, logikai érték, logikai változó, Boole-algebra.

Hálózatnak olyan berendezést, kapcsolást, áramkört nevezünk, melynek bemenetei és kimenetei vannak, és olyan feladatot old meg, ahol a kimenetek a bemenetek és egyéb mennyiségek, események (pl. idő, előzmények, stb.) függvényében vesznek fel különböző állapotokat.

Mi kizárólag elektromos áramköri elemekből felépített hálózatokat tanulnunk, de a gyakorlatban természetesen más hálózatok is vannak (pneumatikus, stb.) Ezért szinonimaként használjuk az áramkör, hálózat, kapcsolás fogalmakat e fejezetben, noha ez máshol nem egészen pontos.

Logikai hálózatnak (áramkörnek) az olyan hálózatot (áramkört) nevezzük, melynek bemenetei is és kimenetei is logikai állapotokkal jellemezhetők. A különböző logikai döntéseket logikai hálózatokkal valósítjuk meg. Olyan hálózatok, amelyek logikai műveleteket végeznek, kimeneti jeleket a bemeneti jelek függvényében logikai műveletsor eredményeként állítják elő. Nem csak logikai, hanem analóg és vegyes hálózatok is vannak. Az analóg és vegyesen analóg és digitális áramköröket tartalmazó hálózatokat később tanuljuk.

- Kombinációs logikai hálózat: olyan logikai hálózat, mely kimenetei csak a bemenetek állapotaitól, kombinációitól függnek, semmi mástól. A bemeneten fellépő ugyanazon jelkombinációkhoz a kimeneten ugyanazon kimeneti jelkombináció tartozik. Azonos kimeneti jelkombináció több bemeneti jelkombinációhoz is tartozhat
- Sorrendi, nemzetközi szóval szekvenciális: az olyan logikai hálózat melynek kimenetei nem csak a bemenetek kombinációitól, hanem az előzményektől, a különböző kombinációk sorrendjétől is függenek. Ugyanazon bemeneti kombinációhoz más-más kimeneti esemény tartozhat attól függően, hogy ezen bemeneti kombinációk milyen sorrendben kerülnek a hálózat be-menetére.

A formális logika törvényszerűségeinek leírása George Boole és Augustus De Morgan nevéhez fűződik. A matematikai logika kapcsolóelemek leírására használható részét Boole-algebrának nevezik:

- Kapcsolás algebra
- Logikai algebra

A Boole-algebra változói kétértékű események:

- Értéke van:
    - Igaz ⇔ bekövetkezik ⇔ bekapcsol ⇔ „1”
    - Hamis ⇔ nem következik be ⇔ kikapcsol ⇔ „0”
- Jelölése: az ABC betüivel
- Neve: logikai változó (Boole-változó)

Logikai változók ábrázolása

- Venn-diagram:
    - Logikai változó ⇔ síkba leképezett ponthalmaz
    - Ponthalmaz ⇔ tetszőleges síkidommal határolt terület
    - A síkidom kerületen belüli terület ⇔ „1”
    - A síkidomon kívül eső terület ⇔ „0”
    - Legfeljebb 3 változóig használatosak.
- Veitch-diagram:
    - Módosított Venn-diagram
    - Az igenleges és a nemleges értékeket egyenlő területrészekkel ábrázolják.
    - A diagram peremén vonallal és betűvel jelöljük, hogy a változó igenleges (ponált) illetve nemleges (negált) állapota mely területrészen található
    - Síkbeli Veitch-diagramon maximum 4, térbelin maximum 6 változó ábrázolható szemléletesen
- Idődiagram:
    - A kétértékű eseményeket az idő függvényében ábrázoljuk
    - Az események időbeli lefolyása követhető
    - Tetszőleges számú változó ábrázolható egyidejűleg

Boole algebra: halmazok logikája

- Halmazon a közös tulajdonságú dolgok összességét értjük.
- A halmaz elemei a halmazhoz tartozó dolgok
- Egy A halmaz kiegészítő (komplemens) halmaza alatt azt az A
- halmazt értjük, mely elemeinek nincs meg az A elemeinek tulajdonsága.
- Az üres halmaz olyan halmaz, melynek egyáltalán nincs eleme. Az üres halmazt nulla halmaznak, 0 halmaznak is nevezik. Az üres halmaz jele a 0.

- Részhalmaz egy halmaz elemeinek olyan csoportja, melyeket további tulajdonságokkal határozunk meg.
- Az üres halmaz komplemense az univerzális halmaz a Boole algebrában.
- Az univerzális halmaz jele az 1. Az univerzális halmaz komplemense a 0 halmaz.
- Két halmaz (A és B) közös része, logikai szorzata, metszete, konjunkciója alatt azokat az elemeket értjük, melyek egyszerre elemei A-nak és B-nek is. A logikai szorzatot a Boole algebrában így jelöljük: C = AB. Természetesen AB = BA. A halmazelméletben szokás AB-t A∩B-vel jelölni.
- Azok az elemek, melyek vagy az A halmazhoz, vagy a B halmazhoz, vagy mindkettőhöz tartoznak, az A és B halmaz logikai összegét, más néven egyesítését, unióját alkotják.

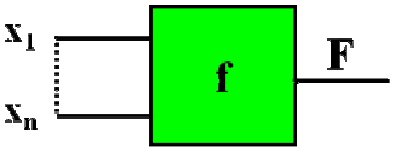
2. tétel: Boole-algebra, logikai alapműveletek, logikai azonosságok, logikai függvények.

A logikai algebra törvényszerűségének kidolgozása George Boole és Augustus De Morgan nevéhez fűződik. Vezérlések tervezéséhez Nakasima és Hanzawa japán és Shannon amerikai mérnökök használták, és fejlesztették tovább. A Boole algebra egy olyan halmazelméleti tárgyalási mód, amelyben az elemek száma kettő (hamis és igaz). A kapcsolási algebrában az elemek jelölésére használt 0,1 nem számjegyek, hanem szimbólumok, amihez a hamis és igaz értéket rendeljük. A Boole algebrában 3 alapműveletet definiálnak:

- Logikai összeadás
- Logikai szorzás
- Negáció, inverz képzés

Tanulmányaink során egyszerű, kétértékű ítéleteket hozunk, egyszerű, egyenként két lehetséges állapotú feltételekkel. Ezt táblázatban is ábrázolhatjuk. Ebben a táblázatban egyszerűen számba vesszük, az ítélet mikor, milyen feltételek esetén igaz, és mikor hamis. Az ilyen táblázatot nevezzük igazságtáblázatnak.

A logikai függvény a kimeneti és bemeneti események közötti logikai kapcsolatot adja meg. Mind a független (bemeneti), mind a függő (kimeneti) változók csak két értéket (0,1) vehetnek fel. Jelölése:



$$F = f(A, B, C, \text{K}, N).$$
$$F = f(X_1, X_2, X_3, \text{K}, X_n).$$

Mivel mind a bemeneti, mind a kimeneti változók két-értékűek, a képezhető kapcsolási függvények száma:

$$K = 2^{2^n}$$

A bemeneti változók összes lehetséges kombinációit és a függvény által meghatározott kimeneti eseményt feltüntető táblázat a függvény igazságtáblázatának (kombinációs- vagy értéktáblázatnak is hívják) nevezzük. Logikai alapműveletek közül van 2, mely egyváltozós:

- Invertálás: a bemeneti jel logikai ellentettjét jeleníti meg a kimeneten, bájtokra is kiterjeszthető, az invertálást a bájt valamennyi bitjére el kell végezni, az eredmény az eredeti tartalom egyes komplemense.

  Inverterrel realizálható

$$\bar{1} = 0 \quad \bar{\bar{A}} = A$$
$$\bar{0} = 1 \quad \bar{\bar{\bar{A}}} = \bar{A}$$

A	$F_1=\bar{A}$
0	1
1	0

Nem művelet szabályai:

- jelkövetés: a bemeneti jel logikai értékét jeleníti meg a kimeneten A bemeneten fellépő jelváltást a kimenet időkéséssel követi. Relés hálózatokban záró érintkezővel realizálható.

A	F_1^1=A
0	0
1	1

<u>Kétváltozós logikai függvények</u>

- ÉS kapcsolat (konjunkció)

$$F_8^2 = (A,B) = A \wedge B = A \& \mathrm{B} = \mathrm{A} \cdot \mathrm{B} = \mathrm{AB}$$

  - Szokásos további elnevezései:
    - AND művelet
    - Konjunkció
    - Logikai szorzás
  - Megvalósítása:
    - Záró érintkezők soros kapcsolása
    - Integrált ÉS (AND) kapu áramkörökkel
  - Az ÉS művelet szabályai:

$A \cdot 1 = A \quad \overline{A} \cdot 1 = \overline{A} \quad 0 \cdot 0 = 0$
$A \cdot 0 = 0 \quad \overline{A} \cdot 0 = 0 \quad 0 \cdot 1 = 0$
$A \cdot A = A \quad \overline{A} \cdot \overline{A} = \overline{A} \quad 1 \cdot 0 = 0$
$A \cdot \overline{A} = 0 \quad \overline{A} \cdot A = 0 \quad 1 \cdot 1 = 1$
$A \cdot B = B \cdot A \rightarrow$ *kommutatív*
$A \cdot (B \cdot C) = (A \cdot B) \cdot C = ABC \rightarrow$ *asszociatív*

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

  - Az ÉS kapu bementén a 0 jel a meghatározó.

- A művelet bájtokra is kiterjeszthető. Ez esetben a műveletet a bájt valamennyi azonos pozíciójú bitjére el kell végezni.

- VAGY kapcsolat $F_{14}^2 = (A, B) = A \vee B = A + \mathrm{B}$
- Szokásos további elnevezései:
  - OR művelet
  - Diszjunkció
  - Logikai összeadás
- Megvalósítása:
  - Záró érintkezők párhuzamos kapcsolása
  - Integrált VAGY (OR) kapu áramkörökkel
- A VAGY művelet szabályai:

$$A \vee 1 = 1 \quad \overline{A} \vee 1 = 1 \quad 0 \vee 0 = 0$$
$$A \vee 0 = A \quad \overline{A} \vee 0 = \overline{A} \quad 0 \vee 1 = 1$$
$$A \vee A = A \quad \overline{A} \vee \overline{A} = \overline{A} \quad 1 \vee 0 = 1$$
$$A \vee \overline{A} = 1 \quad \overline{A} \vee A = 1 \quad 1 \vee 1 = 1$$
$$A \vee B = B \vee A \rightarrow \textit{kommutatív}$$
$$A \vee (B \vee C) = (A \vee B) \vee C = A \vee B \vee C \rightarrow \textit{asszociatív}$$

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

A VAGY kapu bementén az 1 jel a meghatározó. A művelet bájtokra is kiterjeszthető. Ez esetben a műveletet a bájt valamennyi azonos pozíciójú bitjére el kell végezni.

<u>De Morgan szabályok</u>

- Érintkezős realizálási módnál a következő megfeleltetések igazak:
  - Záró érintkező ⇨ változó igaz értéke
  - Változó negált értéke ⇨ bontó érintkező
  - Konjunkció (ÉS művelet) ⇨ soros kapcsolás
  - Diszjunkció (VAGY művelet) ⇨ párhuzamos kapcsolás
- A NEM, az ÉS és a VAGY kapcsolatot logikai alapműveletnek tekintjük.

- Az ÉS és a VAGY műveletek között a De Morgan szabályok teremtenek kapcsolatot a NEM függvény révén.

$$\overline{X_1 \vee X_2 \vee \Lambda \vee X_n} = \overline{X_1} \wedge \overline{X_2} \wedge \Lambda \wedge \overline{X_n}$$
$$\overline{X_1 \wedge X_2 \wedge \Lambda \wedge X_n} = \overline{X_1} \vee \overline{X_2} \vee \Lambda \vee \overline{X_n}$$

- Két változó esetén a De Morgan szabály a következőképpen alakul:

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

  - Szavakban kifejezve: Szorzat negáltja egyenlő a tényezők negáltjának a szorzatával.
  - Összeg negáltja egyenlő a tagok negáltjának szorzatával.

<u>További műveletek</u>

<u>Antivalencia</u>

- Szokásos további elnevezései:
  - Kizáró VAGY művelet
  - Exclusív OR (EXOR, XOR, XRA)
- Megvalósítása:
  - Alternáló kapcsolás (váltóérintkező)
  - Integrált ANTIVALENCIA kapu áramkörökkel

Antivalencia: ($\overline{A}$xB+$\overline{B}$xA)

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

<u>Ekvivalencia</u>

- Szokásos további elnevezései:
  - Koincidencia
  - Exclusív NOR
  - XNOR
  - EXNOR

Ekvivalencia: ($\overline{A}$x$\overline{B}$)+(BxA)

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

NAND

- Az ÉS illetve NÉS kapcsolat egymás negáltja.
- A NAND kapu bemenetén a 0 jel a meghatározó
- Megvalósítása:
    - Integrált NÉS (NAND) kapu áramkörökkel

NAND: $\overline{A \times B} = \overline{A} + \overline{B}$

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

NOR

- A VAGY illetve NVAGY kapcsolat egymás negáltja.
- A NOR kapu bemenetén az 1 jel a meghatározó
- Megvalósítása:
    - Integrált NVAGY (NOR) kapu áramkörökkel

NOR: $\overline{A + B} = \overline{A} \times \overline{B}$

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

Implikáció:

$$F_{11}^2 = (A, B) = \overline{A} \vee B$$

Inhibíció:

$$F_4^2 = (A, B) = A \wedge \overline{B}$$

Duális logikai függvények

Abban térnek el egymástól, hogy benne az ÉS, illetve VAGY műveletek fel vannak cserélve

$A \wedge B$	$A \vee B$
$\overline{A \wedge B}$	$\overline{A \vee B}$
$\overline{A} \wedge B$	$\overline{A} \vee B$
$A \wedge \overline{B}$	$A \vee \overline{B}$
$A \barwedge B$	$A \forall B$

Inverz függvények

Az azonos bemeneti változó értékekhez tartozó függvény értékek egymás ellentettjei:

$A \wedge B$	$\overline{A \wedge B}$
$A \vee B$	$\overline{A \vee B}$
$A \wedge \overline{B}$	$\overline{A} \vee B$
$\overline{A} \wedge B$	$A \vee \overline{B}$
$A \forall B$	$A \barwedge B$

3. tétel: Logikai függvények kanonikus alakjai, minterm, maxterm fogalma.

A logikai változók azon kifejezése, amelyben a függő változó igazságértékét (ami 0 és 1 lehet) a független változók igazság értékei,- ami szintén 0 és 1 lehet - határoznak meg, logikai (Boole, kapcsolási) függvényeknek nevezzük. Egy függő változóhoz több független változó rendelhető.

$$Y=f(X_1, X_2,\ldots X_n)$$

A független változók a függvény bemenetei, míg a függő változó a függvény kimenete. Egy n változót tartalmazó logikai kifejezés minden változója két állapotot vehet fel, így a kifejezés összes lehetséges érték kombinációinak száma $k=2^n$. A függvénykapcsolat a k számú kombináció mindegyikéhez a függő változó 0 vagy 1 értékét rendelheti, a definiálható függvény kapcsolatot száma tehát összesen.

$$N= 2^k$$

Többváltozós logikai függvények megadási módjai

- Bármely n > 2 változós függvény felépíthető két-változós függvényekből
- Lehetséges megadási módjai:
    - Kombinációs (érték) táblázat
    - Index számos alak
    - Grafikus módszer
    - Kanonikus (normál) alakok

Kombinációs (érték) táblázat

- Teljesen határozottnak (specifikáltnak) nevezünk egy függvényt, ha a bemeneti változók minden lehetséges kombinációjához 1, illetve 0 értékek tartoznak. Ebben az esetben elegendő vagy csak az 1, vagy csak a 0 helyeket felsorolni.
- Amennyiben a független változók 2^n kombinációja közül csak „m” kombinációhoz írják elő, melyekben legyen a függvény értéke 1, illetve 0, a függvény nem teljesen specifikált. A maradék (2^n-m) kombinációban a függvény értéke tetszőlegesen 0 –nak, vagy 1 –nek vehető. Az ilyen kombinációkat közömbös (redundáns) kombinációnak nevezzük.

  Nem teljesen meghatározott függvények:

  A logikai függvény bemeneti kombinációi három csoportba sorolhatók:

- A függvény „1” értékeihez tartozó kombinációk,
- A függvény „0” értékeihez tartozó kombinációk,
- Közömbös kombinációk

A közömbös kombinációk a logikai függvények egyszerűsítésénél használhatók fel előnyösen.

Index számos alak

- A függvények megadhatók decimális index felhasználásával is:
  - A felső indexbe a változók számát,
  - Az alsó indexbe a függvény értékekből képzett 2^n helyiértékű bináris szám decimális megfelelőjét írjuk
  - A függvény csak akkor egyértelmű, ha a bemeneti változók sorrendje rögzített.

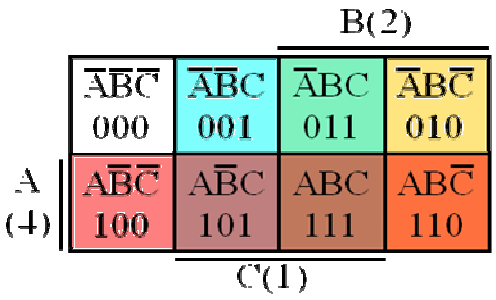
pl:

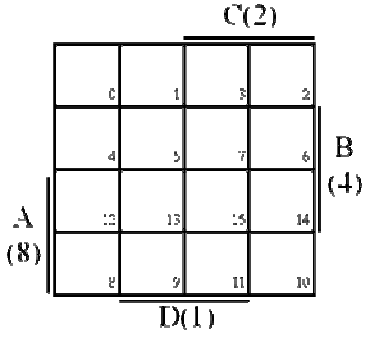
i	2^2 A	2^1 B	2^0 C	$F^3_{203}(A,B,C)$ (αi)	$F^3(A,B,C)$ (αi)
0	0	0	0	1	0
1	0	0	1	1	1
2	0	1	0	0	X
3	0	1	1	1	1
4	1	0	0	0	X
5	1	0	1	0	0
6	1	1	0	1	1
7	1	1	1	1	X

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

Grafikus módszer

- Minél több a bemeneti változók száma a kombinációs táblázat kitöltése annál több munkát igényel.
- Ilyen esetekben célszerűbb a Karnaugh-Veitch (KV) táblák használata.
- A táblázat egy-egy cellája egy-egy bemeneti kombinációnak felel meg.
- Ennek megfelelően egy „n” változós KV táblának 2^n cellája van.
- A gyakorlatban azok az elrendezések használatosak, a-melyeknél egymás mellett szomszédos bemeneti kombinációk vannak.
  - Két bemeneti kombináció akkor szomszédos, ha közöttük csak egy változó értékében van eltérés





Kanonikus (normál) alakok

- Egy függvény megadásánál a függvény 1, illetve 0 helyeiből egyaránt kiindulhatunk.
- A logikai szorzat (ÉS kapcsolat) egyetlen 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.
- Normál alakúnak mondjuk a függvényt, ha:
  - a változók szorzatainak összegéből (diszjunktív normál alak),
  - vagy összegeinek szorzatából (konjunktív normál alak)

  áll.
- A függvény teljes normál alakjának nevezzük azt a normál alakot, amelynek minden tagjában (diszjunktív alak), illetve tényezőjében (konjunktív alak) szerepel:
  - a függvény összes változója ponált(igenleges),
  - vagy negált(nemleges) alakban.
- A teljes diszjunktív alakban szereplő logikai szorzatokat mintermeknek (egy-egy cellát foglalnak el) nevezzük. Minterm alatt olyan logikai ÉS-t értünk, melyben minden változó egyszer, és csakis egyszer fordul elő tagadott, vagy nem tagadott formában.
- A KV táblát minterm táblának is szokás nevezni.

  A teljes diszjunktív normál alak képzési szabálya:

$$F\left(X_1 X_2 X_3 \Lambda\ X_n\right) = \sum_{i=0}^{2^n-1} \alpha_i^n m_i^n$$

Teljes konjunktív normál alak

A teljes konjunktív normál alak felírásához a függvény 0 helyeiből kell kiindulni.

$$\overline{F_{203}^3}(A,B,C) = \overline{A}B\overline{C} + A\overline{B}\overline{C} + A\overline{B}C$$

A De Morgan szabály alkalmazása után:

$$F_{203}^3 = \overline{\overline{F_{203}^3}} = \overline{\overline{A}B\overline{C} + A\overline{B}\overline{C} + A\overline{B}C} = \overline{\overline{A}B\overline{C}} \cdot \overline{A\overline{B}\overline{C}} \cdot \overline{A\overline{B}C}$$

$$F_{203}^3(A,B,C) = \left(A + \overline{B} + C\right)\left(\overline{A} + B + C\right)\left(\overline{A} + B + \overline{C}\right)$$

- Az olyan alakban adott függvény, amely összegek szor-zatából áll és az összeg minden tagja valamennyi füg-getlen változót tartalmazza ponált, vagy negált értéké-vel teljes konjunktív normál alakúnak mondjuk.
- E függvény elemi összegeit maxtermeknek (M_i^n) ne-vezzük.
- 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.
- Végül a függvény valamennyi 0 helyére ily módon képzett összegét összevonjuk.

A teljes konjunktív normál alak képzési szabálya:

$$F(X_1, X_2, X_3 \Lambda\ X_n) = \prod_{i=0}^{2^n-1} \left(\alpha_i^n + M_{2^n-1-i}^n\right)$$

$$F_{203}^3(A,B,C) = (1+M_7^3)(1+M_6^3)(0+M_5^3)(1+M_4^3)(0+M_3^3)(0+M_2^3)(1+M_1^3)(1+M_0^3)$$

$$F_{203}^3(A,B,C) = M_5^3 + M_3^3 + M_2^3$$

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

Mintermek és maxtermek közötti kapcsolat

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

$$\overline{m_i^n} = M_{2^n-1-i}^n$$

$$\overline{M_i^n} = m_{2^n-1-i}^n$$

4. tétel: Logikai függvények egyszerűsítése.

Minimalizálás célja:

- alkatrészek:
  - érintkezők,
  - diódák, tranzisztorok,
  - integrált áramkörök.
- utasítások:
  - memóriakapacitás,
  - végrehajtási idő

  megtakarítása.

Minimalizálási szempontok:

- Legegyszerűbb algebrai alak (minimálalak)
- Legkevesebb elemi áramkör
- Legkevesebb IC tok
- Azonos áramköri elemek
- Legkedvezőbb tranziens viselkedés
- Minimális áramköri fokozatok száma
- Minimalizálási eljárások alapja:

$$AB + A\overline{B} = A\left(B + \overline{B}\right) = A + 1 = A$$
$$\left(A + B\right)\left(A + \overline{B}\right) = AA + AB + A\overline{B} + 0 = A$$

Minimalizálási eljárások:

- Egy logikai függvény algebrai alakját minimál alaknak nevezzük, ha azonos átalakításokkal nem hozható kevesebb változót tartalmazó alakra.
- A kapcsolási függvények minimalizálására a következő módszerek használhatók:
  - algebrai,
  - grafikus,
  - numerikus.

Minimalizálási módszerek:

- Algebrai módszer:
  - A Boole algebra szabályainak ismételt alkalmazásával hajtható végre.
  - Az egyszerűsítés során felhasználható összefüggések, azonosságok:

$$A+1=1 \qquad A+0=A$$
$$A\cdot 0=0 \qquad A\cdot 1=A$$
$$A+A=A \qquad A+\overline{A}=1$$
$$A\cdot A=A \qquad A\cdot\overline{A}=0$$
$$A+B=B+A$$
$$A\cdot B=B\cdot A$$
$$A(B+C)=AB+AC$$
$$A+BC=(A+B)(A+C)$$
$$\overline{A\cdot B}=\overline{A}+\overline{B}$$
$$\overline{A+B}=\overline{A}\cdot\overline{B} \qquad (\overline{\overline{A}}=A,\ \overline{\overline{\overline{A}}}=\overline{A})$$

- Példa:

$$K=\overline{A}\,\overline{B}C+A\overline{B}C+A\overline{B}\,\overline{C}+ABC$$
$$K=\overline{B}C(\overline{A}+A)+A\overline{B}(C+\overline{C})+AC(\overline{B}+B)$$
$$K=\overline{B}C+A\overline{B}+AC$$

- Az algebrai egyszerűsítés problémái:
  - Az esetek többségében intuitív megoldást igényel
  - Nehezen algoritmizálható
  - Nehéz megtalálni a legegyszerűbb megoldást
  - Sok változó esetén bonyolult
  - Gondot okozhat, hogy egy „term” többször is felhasználható
  - Nehézséget okozhat az összevonandó részek meg-találása

- Grafikus módszer:
  - Ha egy függvény diszjunktív kanonikus alakjában két olyan minterm fordul elő, melyek egymással szomszédosak, akkor ezek egyetlen szorzattá vonhatók össze.
  - Célszerűen megválasztott peremezésű KV táblán az algebrailag szomszédos kombinációk helyileg is szomszédosak. Ez teszi lehetővé a grafikus minimalizálást.
  - Két 1-gyel jelölt szomszédos minterm egyetlen tömbbé vonható össze.

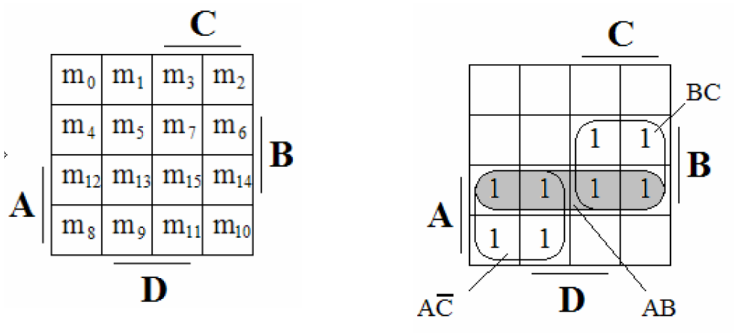
A minterm alakok egyszerűsítése

A Veitch diagram vagy Karnaugh tábla azokkal a tulajdonságokkal bír, hogy az egymás melletti sejtek egy változóban térnek el, így ránézéssel azonnal láthatóak az egyszerűsítésben egyeztethető sejtek, mintermek. Két egyeztethető sejt összevonásával az eltérő igazságértékkel rendelkező változó kiesik. A két sejt tömböt alkot, amelyben a változók száma 1-el csökken. Az egymás melletti tömbök szintén egy változóban különböznek, ezért ez a változó kiesik. Az azonos

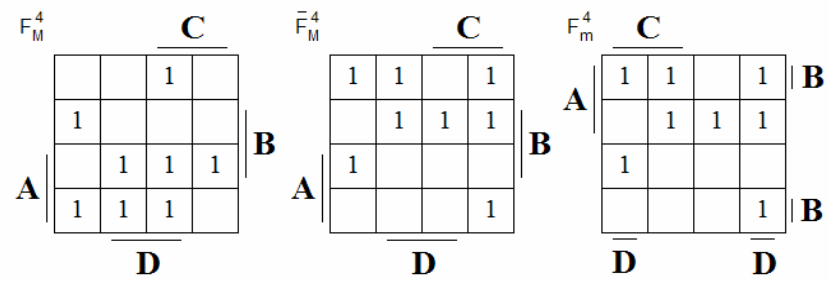
nagyságú egymás melletti tömbök összevonhatók, és főtömböt képeznek. Az egyszerűsítéssel arra kell törekedni, hogy minél nagyobb sejtszámú tömböt lehessen létrehozni, és lehetőleg minden sejt szerepeljen tömbben. A tömbök kifejezése a minterm tábla élére való kivetítéssel megállapítható. Az egyszerűsített alakban minden sejtnek szerepelnie kell lehetőleg a legnagyobb rendszámú tömbben. Csak a lényeges tömböket kell kiválasztani.

- Primimplikánsok megkeresése:
  - A függvény ábrázolása KV táblán.
  - 2^i számú szimmetrikusan elhelyezkedő szomszédos 1-gyel jelölt cella összevonása egy tömbbé. (i db változó esik ki. i=0,1,2,…n).
  - A lehető legnagyobb tömböket célszerű kialakítani.
  - Valamennyi 1 –gyel jelölt cellának legalább egy tömbben szerepelnie kell.
  - Ugyanazon cella több tömbnek is eleme lehet.
  - A KV táblák négy változóig a széleiken összefüggőek.
- A szükséges primimplikánsok kiválasztása:
  - Valamely primimplikáns lényeges, ha tartalmaz olyan m_i^n mintermet, amelyet egyetlen más primimplikáns sem tartalmaz. Ezek a primimplikánsok a függvény megvalósításához nélkülözhetetlenek.
  - Azok a tömbök lesznek a minimalizált függvény szükséges primimplikánsai, amelyek a függvény valamennyi 1 –gyel jelölt cellájának egyszeri lefedéséhez elengedhetetlenül szükségesek.

1.Példa:

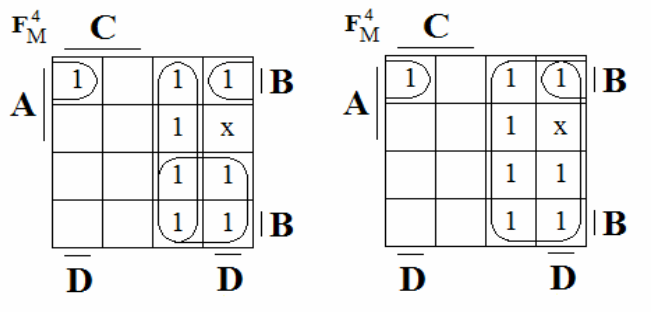


<u>Maxterm alak átírása minterm alakká igazságtáblával:</u>



Redundancia:

Redundancia alatt értjük az olyan változó érték kombinációkat, amelyek fizikailag nem következhetnek be. Ezeket a meg nem történhető, figyelmen kívül hagyható kombinációkat az egyszerűsítésnél tetszőleges értékkel lehet figyelembe venni. A kombinációnak megfelelő sejtbe x-t írunk. Egyszerűsítésnél, a tömbök létrehozásánál a redundáns sejteket 1-el vesszük figyelembe, ha az nagyobb rendszámú tömböt eredményez.



- Numerikus egyszerűsítés:
  - A grafikus eljárás max. 6 változóig használható.
  - A numerikus egyszerüsítés a változók számától függetlenül használható. (Quine-Mc Cluskey-féle numerikus egyszerűsítés)
  - A diszjunktív normál alakban adott függvény egyszerűsítésének lépései:
    - A függvényben szereplő mintermek bináris kódjának felírása és a mintermek csoportokba sorolása a kódjaikban előforduló 1-esek száma szerint, az e-gyesek száma szerint növekvő sorrendben. A csoporton belül a mintermek nagyság szerint növekvő sorrendben következnek. Két csoport szomszédos, ha a két csoporthoz tartozó 1-esek számának különbsége 1. Szomszédos mintermek ezért csak szomszédos csoportokban lehetnek. Ha két minterm szomszédos, akkor decimális megfelelőiknek különbsége 2-nek nem negatív egész kitevőjű hatványa. 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 nagyobb, 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

- Az összehasonlítást a legelső elemmel kell kezdeni. Ezt csak a szomszédos csoport elemeivel kell összehasonlítani. Ha találunk olyan számpárt, a-mely „B”-nek megfelelő értelemben összevonható, akkor mindkét oldalt „+” jellel megjelöljük és a számpár elemeit növekvő sorrendben egy új oszlopban egymás mellé írjuk, majd zárójelben megjelöljük a különbségüket is.
- Az egyszerűsítést az első és második csoport elemeinek összehasonlításával kezdjük.
- Ezt követően elvégezzük az összehasonlítást az első oszlop második és harmadik csoportjának elemei között.

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 a	10,14	(4)
15–	12,14	(2)
	7,15	(8)
	14,15	(1)

- Az összehasonlítást az első oszlop összes szomszédos csoportja között elvégezzük.
- A második oszlopból a harmadik oszlopot az előbbiekben leírt módon képezzük, de az összevonás feltétele, 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 hátrébb álló csoportból való decimális szám legyen a nagyobb.
- Ha nem végezhető el az összevonás, akkor az e-gyes oszlopokban „+” jellel nem megjelölt szorzatok 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 eddigiekben végzett művelet a grafikus eljárás tömbösítésének felel meg.
- A szükséges primimplikánsok kiválasztása a primimplikáns táblázattal történik:
  - A táblázat sorainak a száma megegyezik a primimplikánsok számával.
  - A táblázat oszlopainak száma megegyezik a függvényt alkotó mintermek számával
- Minden primimplikáns sorában megjelöljük azon min-termek oszlopait, melyeket az illető primimplikáns tartalmaz.
- Bekarikázandók azon jelek, amelyek oszlopukban egyedül állnak. Azon primimplikánsok, melyek sorában bekarikázott jel van, nélkülözhetetlen primimplikánsok.

	0	2	6	7	8	10	12	14	15	41
a										X
b	X	X			X	X				
c		X	X			X		X		
d					X	X	X	X		
e			X	X				X	X	

$A = 32,\ B = 16,\ C = 8,\ D = 4,\ E = 2,\ F = 1$

$a = 41 = 101001 = A\overline{B}C\overline{D}\overline{E}F$

$b = 0,2,8,10 \qquad (2,8) = 000000 = \overline{A}\overline{B}\overline{D}\overline{F}$

$c = 2,6,10,14 \quad (4,8) = 000010 = \overline{A}\overline{B}E\overline{F}$

$d = 8,10,12,14 \ (2,4) = 001000 = \overline{A}\overline{B}C\overline{F}$

$e = 6,7,14,15 \quad (1,8) = 000110 = \overline{A}\overline{B}DE$

$F = \overline{A}\overline{B}\overline{D}\overline{F} + \overline{A}\overline{B}C\overline{F} + \overline{A}\overline{B}DE + A\overline{B}C\overline{D}\overline{E}F$

5. tétel: Szimmetrikus logikai függvények. Logikai függvények realizálása.

Az olyan logikai függvényeket melyek kimenete csak a bemenetükön lévő 1-esek számától függ, szimmetrikus függvényeknek nevezik.

$$S_0 = S^3_{1,2}$$

- alsó index: 1-esek száma, amire a kimenet is 1-et ad
- felső index: változók száma

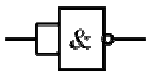
Szimmetria szám: megmutatja, hogy hány változónak kell ponáltnak lennie, hogy a függvény
értéke 1 legyen.

Függvények realizálása

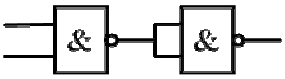
- Kapuáramkörökkel
  - ÉS-VAGY-NEM típusú hálózattal
  - NAND (Sheffer-féle univerzális művelet) kapus hálózattal
  - NOR (Pierce-féle univerzális művelet) kapus hálózattal
- Logikai alapműveletek realizálása NAND kapukkal

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

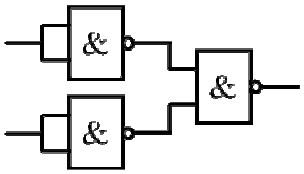
NEM művelet



ÉS művelet



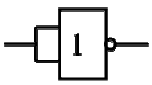
VAGY művelet



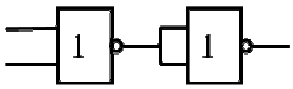
- Logikai alapműveletek realizálása NOR kapukkal

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

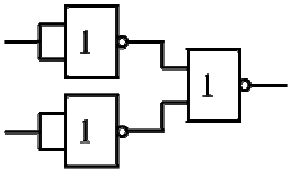
NEM művelet



VAGY művelet



ÉS művelet



Kétfokozatú (kétszintű) hálózatok

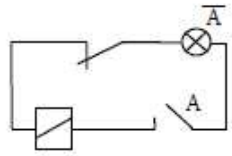
- Bármely logikai függvény 8 féle kétfokozatú alakban realizálható.
- DTL rendszerben az AND-OR illetve az OR-AND hálózatok a kedvezőek.
- TTL rendszerekben a NAND-NAND, a NOR-NOR illetve az AND-NOR típusú hálózatok a kedvezőek.
- A diszjunktív alakról közvetlenül át lehet térni a NAND elemes megvalósításra:
  - Áttéréskor minden ÉS és VAGY kaput NAND kapuval kell helyettesíteni.
  - A kapuk közötti összeköttetések nem változnak.
- A konjunktív alakról közvetlenül át lehet térni a NOR elemes realizációra:
  - Áttéréskor minden VAGY és ÉS kaput NOR kapuval kell helyettesíteni. A kapuk közötti összeköttetések nem változnak.

Többfokozatú (többszintes) hálózatok:

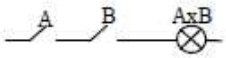
- Univerzális elemekkel (NAND, NOR) többszintű hálózatok is készíthetők, mert a feszültség eltolódások elhanyagolhatók.
  - Diódás kapuáramköröknél a feszültség eltolódások miatt maxi-mum kétfokozatú hálózatokat szokás építeni.
- Analízis szabályai homogén NAND elemes hálózatoknál:
  - A többszintű NAND hálózat minden páratlan szinten bevezetett változót komplementál.
  - A páratlan szinten bevezetett kapuk VAGY műveletet, a páros szinten bevezetett kapuk ÉS műveletet realizálnak a logikai függvényben
- Analízis szabályai homogén NOR elemes hálózatoknál:
  - A többszintű NOR hálózat minden páratlan szinten bevezetett változót komplementál.
  - A páratlan szinten lévő kapuk ÉS műveletet, a páros szinten lévő kapuk VAGY műveletet realizálnak a logikai függvényben
- Többfokozatú NAND hálózattal történő realizálásnál cél, hogy a függvényt olyan alakra hozzuk, hogy megvalósítható legyen ÉS illetve VAGY kapukkal, amelyek sorrendje:
  - … VAGY-ÉS-VAGY-ÉS …
- A negált változókat a páratlan, a ponált változókat páros szinten célszerű bevezetni

- Realizálás kontaktusokkal is lehetséges:

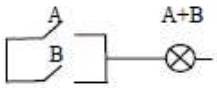
- Negáció ($\overline{A}$)

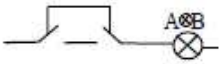


- ÉS (AxB)



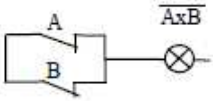
- VAGY (A+B)





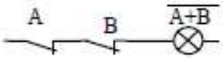
- Ekvivalencia (A xor B= $\overline{A}x\overline{B}$ +AxB)

- NA$\overline{ND}$



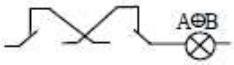
(AxB)

- $\overline{NOR}$



(A+B)

- Antivalencia:A+B



6. Hazárdok

Hazárdjelenségek

- A gyakorlatban a kapuk és vezetékek jelterjedési késleltetése nem elhanyagolható.
- A késleltető hatások átmenetileg hibás kimeneti kombinációkat hozhatnak létre.
- A hibák előfordulása a környezeti változóktól: hőmérséklet, öregedés, stb. függhet, így előzetesen nem vehetők számításba.
- Az ilyen véletlenszerű, rendszertelen hibajelenségeket hazárdjelenségeknek nevezzük.
- Tervezéskor arra kell figyelnünk, hogy a kombinációs hálózat működése a lehető legnagyobb mértékben független legyen a késleltetési viszonyok alakulásától.

Hazárdok kialakulása

- Egy adott bemeneti jelkombinációról egy attól csak egy változó ponált vagy negált értékében különböző másik jelkombinációra való áttérés úgy tekinthető, mint a Karnaugh-táblában egy szomszédos cellára való átmenet.
- Háromféle átmenet lehetséges:
  - Egy 1-gyel jelölt celláról, egy szomszédos 1-gyel megjelölt cellára való átmenet.
  - Egy 0-val jelölt celláról, egy szomszédos 0-val megjelölt cellára való átmenet.
  - 0-val jelölt celláról 1-gyel jelölt (és fordítva) cellára törté-nő átmenet.

Hazárdok típusai

- Minden esetben egy bemeneti változó megváltozásának a hatását vizsgáljuk.
- Statikus hazárd:
  - Statikus „1” hazárd:
    - Állandónak feltételezett „1” kimenet rövid időre „0” értékűre változik.
  - Statikus „0” hazárd:
    - Átmenetkor a „0” értékű állandósult kimenő jel mellett „1” értékű hamis tranziens lép fel.
- Dinamikus hazárd:
  - Átmenet esetén a kimenő jel az előírt egyszeri változás helyett többször is megváltozik, mielőtt az állandósult értéket felvenné.

Statikus hazárdok algebrai feltételei

- Szükséges feltétel:

- A függvény algebrai alakjában forduljon elő a következő formába rendezhető primimplikáns:

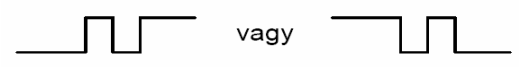
$$VX \wedge \overline{V}Y$$

- Elégséges feltétel:
  - A fenti primimplikánsok X és Y részének ne legyen közös része:

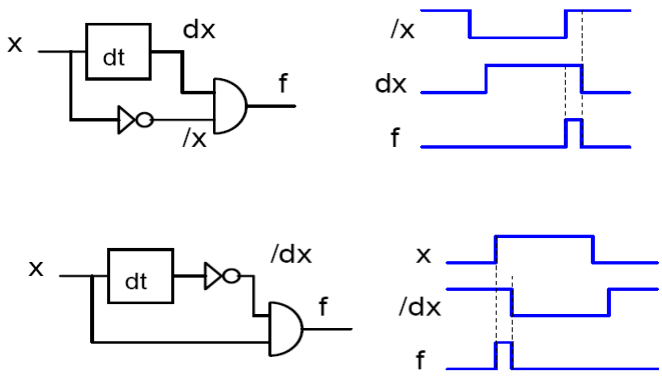
$$XY \neq 0$$

Statikus hazárd típusok

Dinamikus hazárd típusok



Példa



Statikus hazárd

- Oka:
  - Akkor beszélünk statikus hazárdról, ha valamely bemeneti kombinációról egy szomszédos bemeneti kombinációra ugrunk (vagyis egyetlen bemenet értékét megváltoztatjuk), és a kívánt stabil kimenet helyett egy impulzust kapunk.
  - A Karnaugh-táblán ez akkor következik be, ha az egyik hurokból épp egy másikba ugrunk át.

- Kiküszöbölésének módja:
  - A szomszédos hurkok közötti kritikus átmenetet is le kell fedni egy újabb hurokkal.
  - A statikus hazárdmentesítés tehát abból áll, hogy minden szomszédos hurkot összekötünk egy további hurokkal.
- Egy hálózat akkor és csak akkor mentes a statikus hazárdtól, ha bármely két szomszédos mintermet lefed egy-egy primimplikáns (azaz a Karnaugh-táblában bármely két szomszédos 1-est lefed egy-egy hurok.)

Dinamikus hazárd

- Abban az esetben, ha egyetlen bemenet átállításakor egy egyszerű kimeneti jelváltozást várnánk, de ehelyett egy impulzussal tarkított jelváltozást tapasztalunk dinamikus hazárdról beszélünk.
- Dinamikus hazárd csak abban a hálózatban alakul-hat ki, amelynek valamelyik részéből nem küszöböltük ki a statikus hazárdot – a jelenséget ugyanis a statikus hazárd okozta impulzus idézi elő.

Funkcionális hazárd

- Oka:
  - Két bemenetet változtatunk meg egy időben.
  - A valóságban azonban két jel soha nem változik egyszerre, egy nagyon kicsi eltérés biztos van a változásuk időpontja között.
- Megszüntetésének módja:
  - Szinkronizációval szüntethető meg.
  - A kombinációs hálózat elé és mögé olyan szinkronizáló elemeket teszünk (ún. léptető regisztereket), amelyek csak bizonyos időközönként engedik a kombinációs hálózat bemeneteire a jeleket, megvárják, míg a hazárdok eltűnnek a hálózatból, és csak ezután jelenítik meg a kimeneteken a változásokat.
- Ez a fajta megoldás azonban sajnos csökkenti a hálózat gyorsaságát.

7. Aszinkron sorrendi hálózatok és állapottáblája

Sorrendi hálózatok

- Visszacsatolásos hálózatok
    - A visszacsatolás eleve tárolási tulajdonságot jelent.
- A kimenőjelet nem kizárólag a bemeneti jelkombi-náció határozza meg, hanem függ a rendszer pillanatnyi állapotától (q) is.
    - Azaz a kimeneti esemény attól is függ, hogy az egyes bemeneti kombinációk milyen sorrendben érkeznek az áramkör bemenetére.
- Sorrendi hálózatok leírási módszerei:
    - Ütemdiagram,
    - Állapot tábla,
    - Állapot diagram,
        - Folyamatábra,
        - Létradiagram

Ütemdiagram

- A gyakorlatban az okozza a problémát, hogy csak a bemenő- és kimenő jelek ismeretében nem lehet egymástól minden ütemet megkülönböztetni.
- A különböző ütemek megkülönböztetéséhez szükség van egy ún. szekunder vagy memória (q) elemre.
- A szekunder elemek részt vesznek saját állapotuk fenntartásában.
    - Azaz visszahatnak a saját bemenetükre.
    - A sorrendi hálózat tehát visszacsatolást tartalmaz.

Állapottáblák

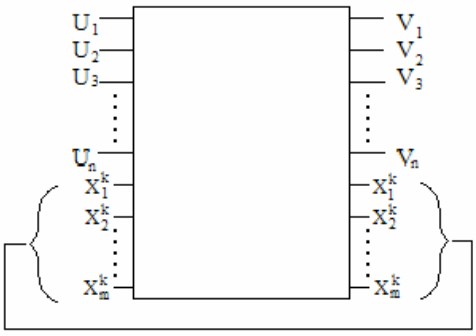
- Gerjesztési tábla:
    - A következő állapotot határozza meg.
    - Táblázat vízszintes peremezése:
        - Bemeneti változók
    - Függőleges peremezés:
        - Szekunder változók (ami szerint visszacsatolást hoztunk létre)
    - Stabil állapot:
        - $Q = q$

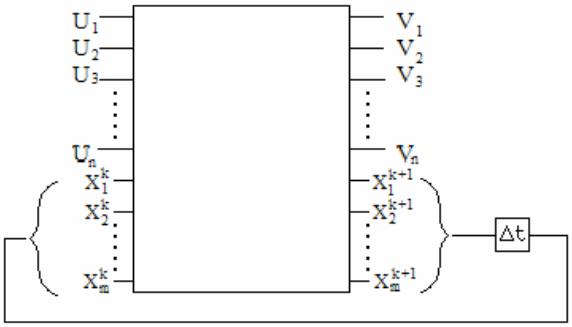
- Instabil (átmeneti) állapot:
  - Nem maradhat fenn sokáig

- Kimeneti tábla:
  - A kimenet értékét határozza meg.

<u>Aszinkron sorrendi hálózat</u>

- Az állapotváltozásokat a bemenő változók értékének megváltozása közvetlenül előidézi.
  - Az időviszonyokat csak az egyes elemek késleltetései befolyásolják.
- Csak akkor léphet fel új bemeneti kombináció, ha a meglévő bemeneti kombináció mellett már stabil állapot alakult ki.
- A stabil állapotokat ezek a rendszerek instabil állapotokon keresztül érik el.
  - Ezek kizárólag azt a célt szolgálják, hogy a megfelelő stabil állapotba juttassák a rendszert
- Versenyhelyzet lép fel az elemek különböző késleltetése miatt.
- A hazárdok komoly gondot okozhatnak:
  - A rendszer a hazárdok következtében nem kívánt végső állapotba juthat, ami működési hibához vezet.
    - A gerjesztési függvények nem tartalmazhatnak statikus hazárdot





- Sorrendi Hálózatoknál állapottáblát használunk igazságtábla helyett.

8. tétel: Szinkron sorrendi hálózatok és állapottáblája.

-Visszacsatolásos hálózatok. A visszacsatolás eleve tárolási tulajdonságot jelent. A kimenőjelet nem kizárólag a bemeneti jelkombináció határozza meg, hanem függ a rendszer pillanatnyi állapotától (q) is. Azaz a kimeneti esemény attól is függ, hogy az egyes bemeneti kombinációk milyen sorrendben érkeznek az áramkör bemenetére. Sorrendi hálózatok leírási módszerei:

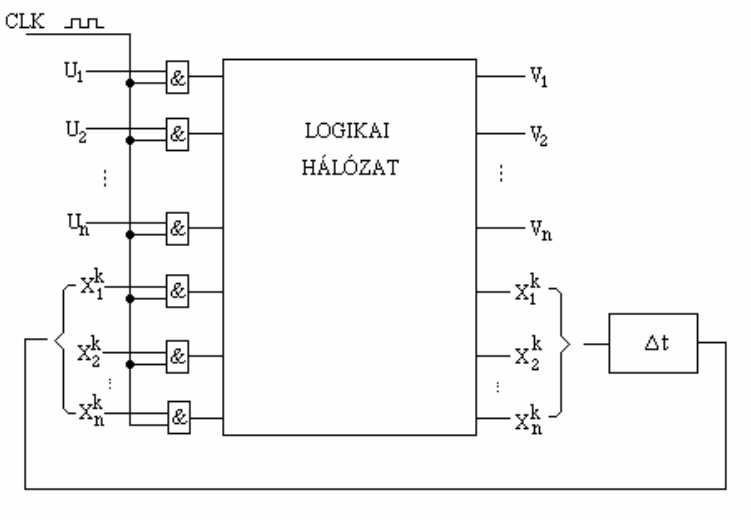
- Ütemdiagram,
- Állapot tábla,
- Állapot diagram,
    - Folyamatábra,
    - Létradiagram,

Állapottáblák

- Gerjesztési tábla:
    - A következő állapotot határozza meg.
    - Táblázat vízszintes peremezése:
        - Bemeneti változók
    - Függőleges peremezés:
        - Szekunder változók (ami szerint visszacsatolást hoztunk létre)
    - Stabil állapot:
        - Q = q
    - Instabil (átmeneti) állapot:
        - Nem maradhat fenn sokáig
- Kimeneti tábla:
    - A kimenet értékét határozza meg.

Szinkron sorrendi hálózatok

- A belső állapotváltozásokat ütemező jelek (órajelek) determinálják.
- Az ilyen hálózatokban minden állapot stabil.
- Bennük lényeges hazárd és/vagy versenyhelyzet nem léphet fel.
- Ezért a digitális rendszerek nagy része szinkron működésű.



Szinkron sorrendi hálózatokban fellépő káros jelenségek

- Órajel elcsúszás:
    - Oka: a szinkronizált flip-flopok nem azonos időben billennek.
    - Főként huzalozott logikában fordul elő.
    - Hibás működést okozhat.
    - Minél rövidebb órajel vezeték vagy késleltető elemek beiktatásával küszöbölhető ki.
- Metastabilitás:
    - Oka: az adat és az órajel egyszerre változik meg.
    - A rendszer nem a két stabil állapot valamelyikébe, ha-nem egy közbülső állapotba kerül.
    - Az órajel aktív élének közelében az adatbemenetek ne változzanak.

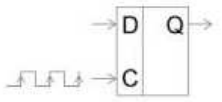
9. tétel: Elemi sorrendi hálózatok.

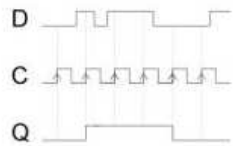
Elemi sorrendi hálózatok

Flip-flopoknak vagy magyarul tárolóknak nevezzük azokat a gyakran használt egyszerű sorrendi hálózatokat, amelyeknek mindössze két állapotuk van, egyetlen kimenettel rendelkeznek, és jellemzőjük még, hogy magát a kimenetet csatoljuk vissza a bemenetre, így az aktuális állapot és kimeneti érték mindig megegyezik. Léteznek aszinkron és szinkron egységek is, ez utóbbiak órajel bemenetükről ismerhetők fel. A tároló elnevezés arra utal, hogy többnyire egy bit tárolására használják őket. A készen vásárolható áramköröknél rendszerint a kimenet negáltja is kivezetésre kerül.

D tároló

Szinkron elemi sorrendi hálózat. A C bemenetre kapcsolt órajel felfutó élekor a kimenet felveszi D értékét, és egészen a következő felfutó élig megőrzi azt.



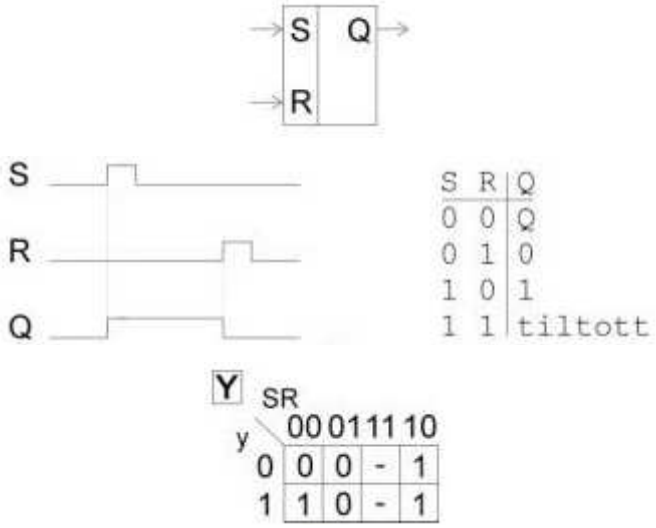


Más megközelítésben tehát a D tároló állapota az órajel ciklusában követi a bemenet változásait.

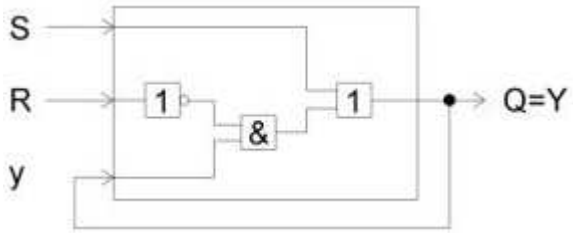
Y	D = 0	D = 1
y = 0	0	1
y = 1	0	1

S-R tároló

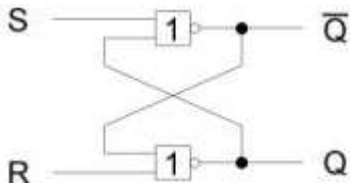
Az S-R flip-flop Set és Reset bemenetinek funkciója a következő: ha az S bemenetre 1-et helyezünk, akkor a kimenet 1-be íródik, ha az R bemenetre helyezünk egyet, akkor a kimenet 0 lesz. Ha mindkét bemenet 0, akkor a kimenet változatlan marad. Egyszerre mindkét bemenetre egyet írni nem szabad, ez tiltott.

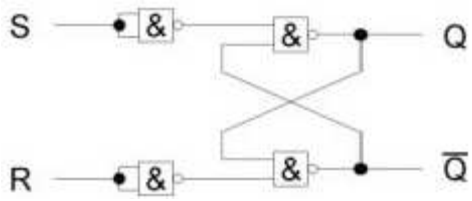


A tároló kapcsolási rajza:



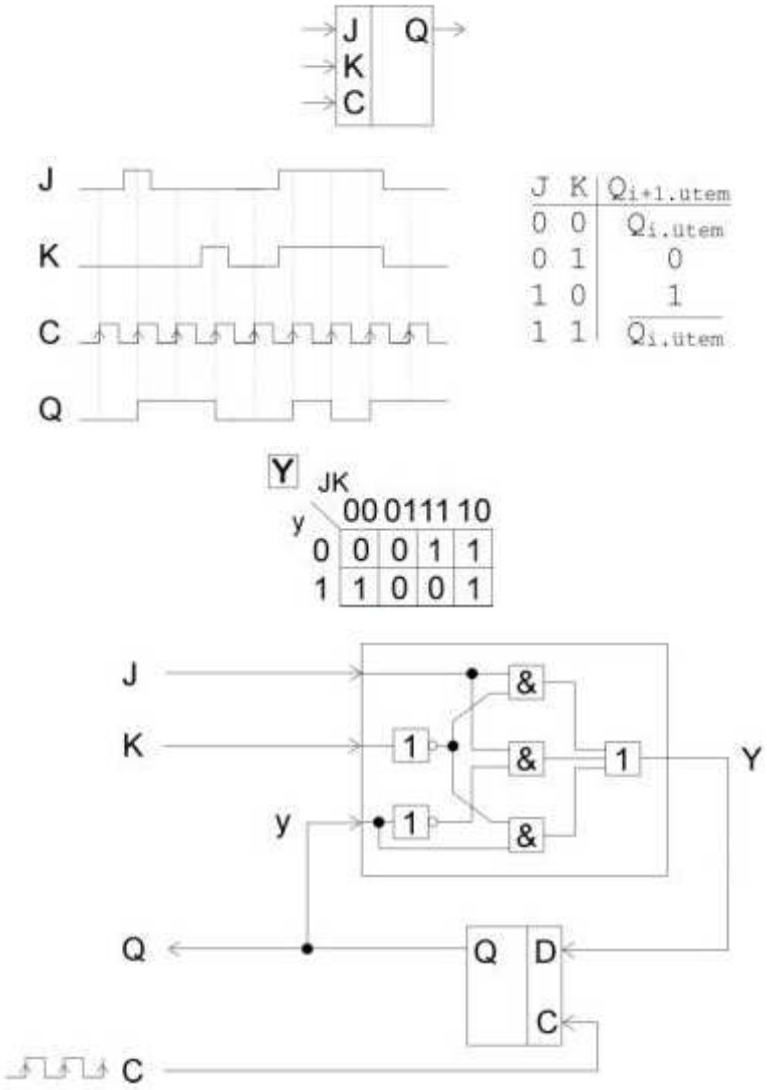
Tudjuk, hogy minden kombinációs hálózat megvalósítható NOR vagy NAND kapukkal, először a NOR, majd a NAND kapus megoldás kerül ismertetésre.





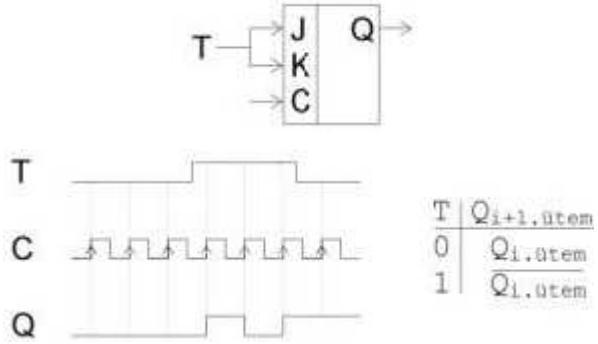
J-K tároló

Működése megegyezik az S-R tárolóval, csak abban különbözik, hogy itt az 11 bemeneti kombináció is engedélyezett, ekkor a kimenet állapota a negáltjára változik. A J-K flip-flop szinkron működésű, tartós 11 bemeneti kombinációra tehát ciklusról ciklusra változtatja a kimenetét.



T tároló

T tárolót úgy kapunk, hogy ha a J-K tároló J és K bemenetit összekötjük, és elnevezzük T-nek. A beíró és törlő funkció elveszik, ha T=0, a kimenet megtartja értékét, ha pedig T=1, Q a negáltjára változik az órajel ütemében.

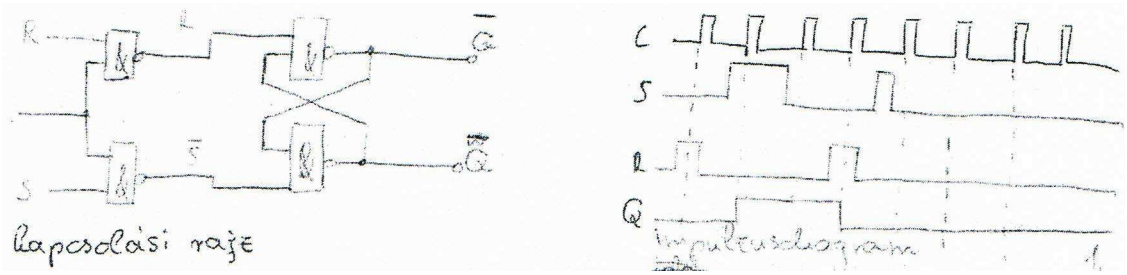


10. tétel: Élvezérelt és master-slave flip-flopok

A digitális áramkörök legfontosabb adattároló eleme a bistabil multivibrátor, vagy más néven flip-flop. Két fontos tulajdonsága van, az egyik, hogy két ellentétes állapottal rendelkezik, külső beavatkozás nélkül akármelyiket megtartja, és egy vagy több bemenettel van ellátva, amelyek lehetővé teszik az áramkör egyik vagy másik állapotba jutását. A sorrendi hálózatok csoportjába tartoznak, két állapotuk révén 1 bit információ tárolására szolgálnak.

Élvezérelt R-S tároló

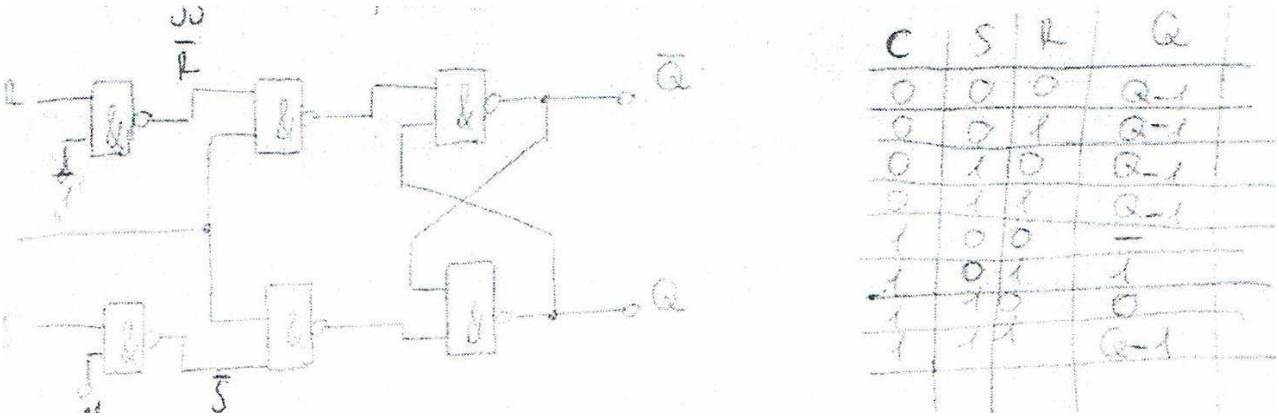
Az áramköri igények gyakran olyan R-S tárolót követelnek, amelyek csak egy meghatározott időtartam alatt veszik figyelembe a bemeneti tárolók állapotát. Ezeket az időtartamokat a járulékos C órajelváltozó határozza meg. Ha az órajelváltozó 0, akkor az eredeti állapotot tartja meg.



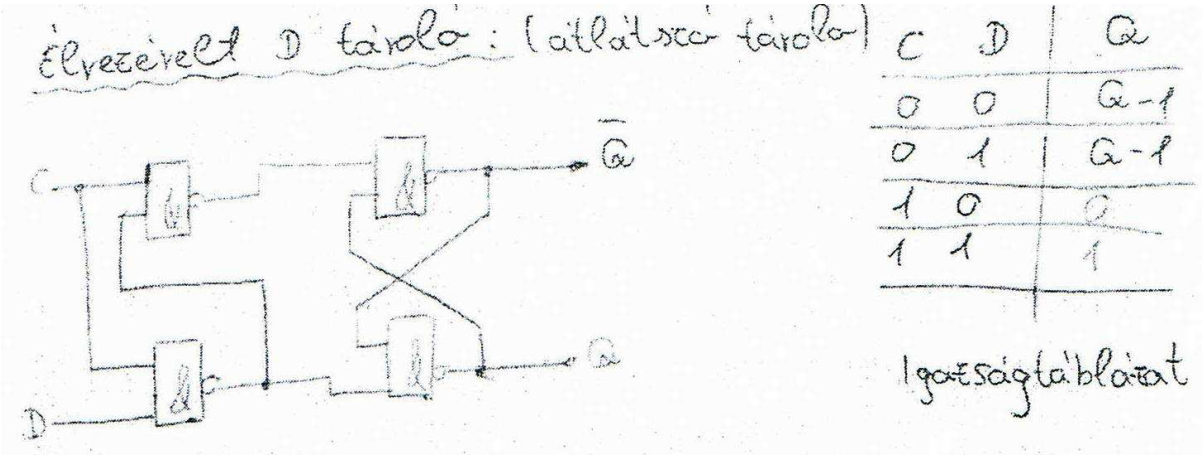
Ha C=1, akkor közönséges R-S tárolóként működik. A kapuzó jel a bemenetre kerülő véletlen jel hatását küszöböli ki. Az S=R=1 bemenő jelkombináció az órajel impulzus alatt kerülendő, mivel ebben az esetben a kimenet értéke bizonytalan.

Élvezérelt inverz R-S tároló

A berendezés elve ugyanaz, csak vezérlése fordított.

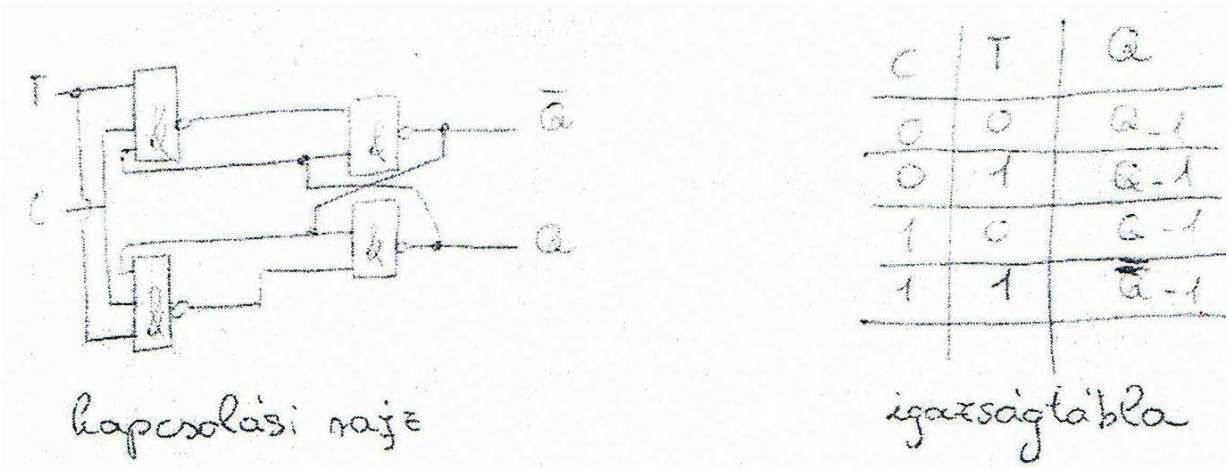


Élvezérelt D tároló

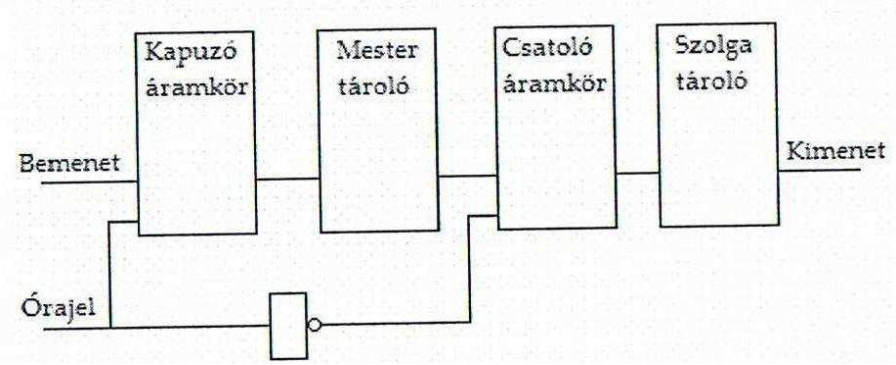


Az így kialakított tárolócella addig tartja a Q=D állapotot, amíg C=1. E tulajdonsága miatt az órajel által vezérelt tárolócellákat D tárolónak nevezzük. Ha C=0, akkor az éppen beállt kimeneti állapot tárolódik.

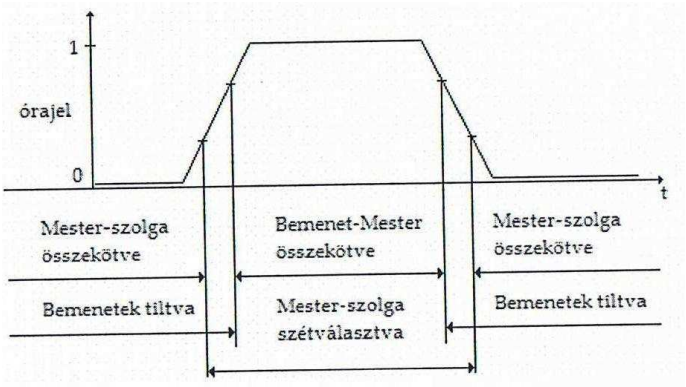
Élvezérelt T tároló



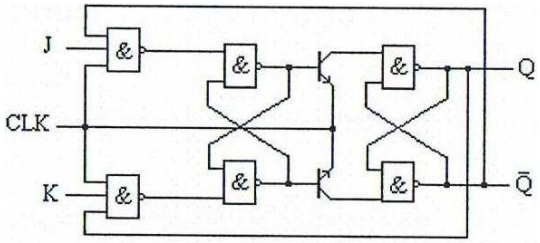
Master-Slave J-K tároló



Működését az órajel vezérli. Az órajel logikai 0 állapotában a kapuáramkört zárja, a Mester tárolót összeköti a Szolga tárolóval, ekkor a bemenetek a kimenetekre nincsenek hatással. Az órajel logikai 1 állapotában a kapuáramkört zárja, és szétválasztja a Mester és a Szolga tárolót. Ekkor a bemenetek a Mester tárolót beállítják, amelynek állapota csak az órajel lefutása után kerül a Szolga tárolón keresztül a kimenetre, de ekkor a bemenetek le vannak zárva. A kapu és a csatoló áramköröknek zsilip szerepe van, amit az órajel vezérel.



Látható, hogy a kimeneten az a jel jelenik meg az órajel lefutása után, amit a bemenetek az órajel lefutása előtt a Mesterbe írtak.



LOGIKAI 1 H= +V
0 L= 0V

J	K	Q^{n+1}
L	L	Q^n
L	H	L
H	L	H
H	H	Q^n

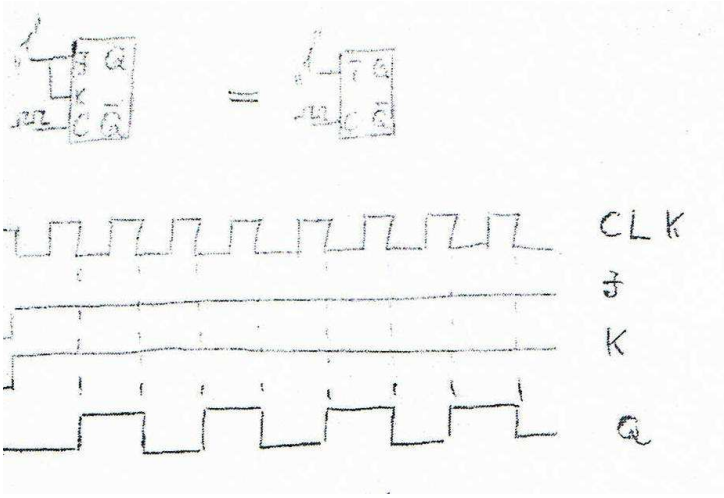
A kimenetek visszacsatolása biztosítja, hogy a J és K bemenetekre adott egyidejű beíró és törlő jelek határozott állapotba viszik a hálózatot. A J-re adott magas jellel beírni, a K-ra adott magas jellel törölni lehet. Mindkét bemenetre magas szint adása a tárolót az ellentétes állapotba billenti. A hálózatot az órajeltől független statikus beíró és törlő bemenetekkel is elláthatják.

11. tétel: Frekvenciaosztók, számlálók.

Frekvenciaosztók

Kialakíthatók J-K (T) és D tárolóból egyaránt.

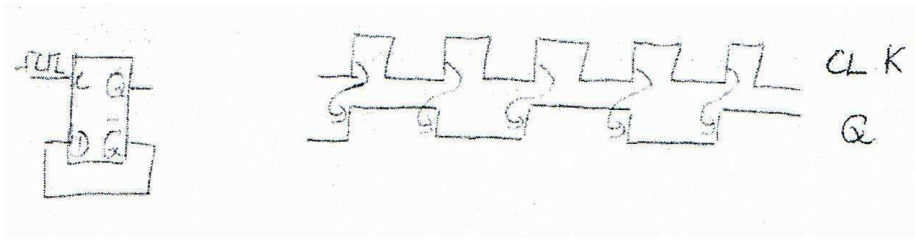
J-K tároló, mint frekvenciaosztó



J=K=1 esetén a tároló Q kimenete minden szinkronjelre átvált. Látható, hogy ebben az esetben a kapcsolás a bemeneti frekvenciát felezi.

D tároló, mint frekvenciaosztó

Master-Slave D tárolóból úgy alakíthatunk ki a legkönnyebben frekvenciaosztót, ha a negált kimenetet visszakötjük a D bemenetre. Ekkor a Q kimenet minden pozitív órajelre átvált.



A tárolóknak ezen tulajdonságát felhasználva könnyen tudunk kialakítani számláló áramköröket.

Számlálók

Olyan sorrendi hálózatok, amelyek impulzusok számlálására és a számlált értékek tárolására alkalmasak. A számlálás alapját két jól elkülöníthető művelet képezi:

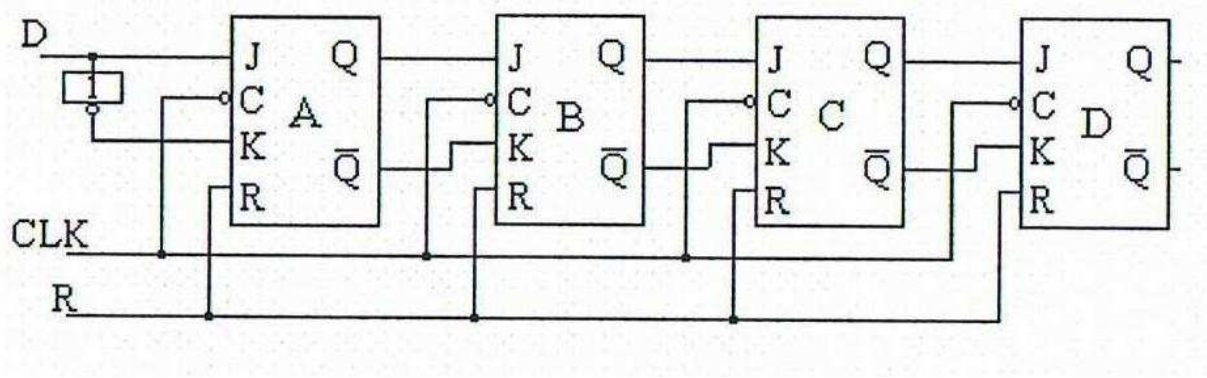
- Tárolás
- Összegzés

A számlálók csoportosíthatók a számlálót alkotó flip-flopok működése alapján, valamint a számlálás iránya alapján.

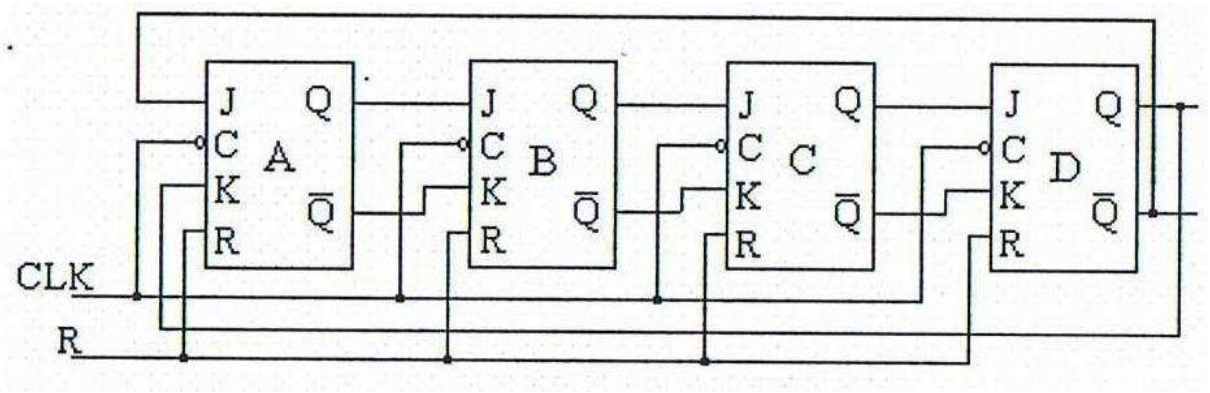
A számlálók csoportosítása

- A flip-flopok működése alapján:
  - Aszinkron számlálók: a flip-flopok egymást billentik.
  - Szinkron számlálók: a flip-flopok billentése egyidejűleg történik, az állapotváltozásokat a flip-flopok előző értéke által vezérelt kombinációs hálózat határozza meg.
- A számlálás iránya alapján:
  - Előre számlálók: minden beérkező impulzus hatására a tárolt értéket eggyel növelik.
  - Visszaszámlálók: minden beérkező impulzus hatására a tárolt értéket eggyel csökkentik.
  - Kétirányú (reverzibilis): a vezérléstől függően előre vagy visszaszámlálnak.

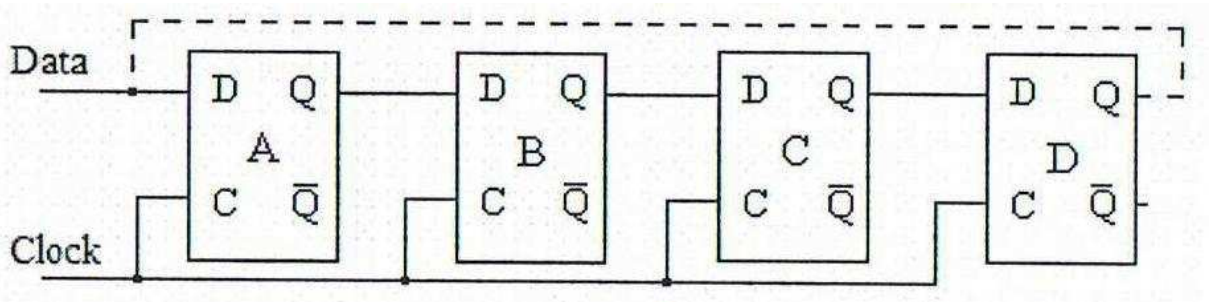
Léptető (shift) regiszter



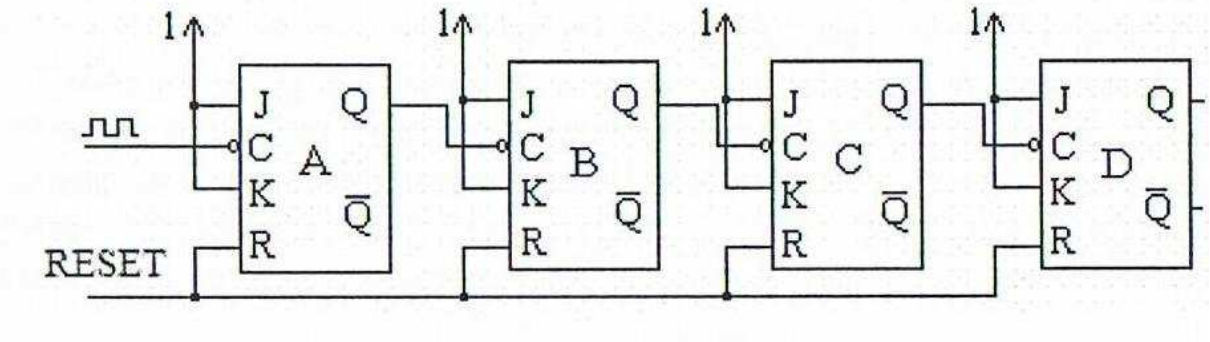
Az órajel minden lefutásakor a D bemeneti érték az A regiszterbe kerül, és minden regiszter tartalma egyel jobbra tolódik. Ha a D jelű tároló inverz kimenetére kötjük a D bemenetet, akkor szinkron gyűrűs számlálót kapunk.



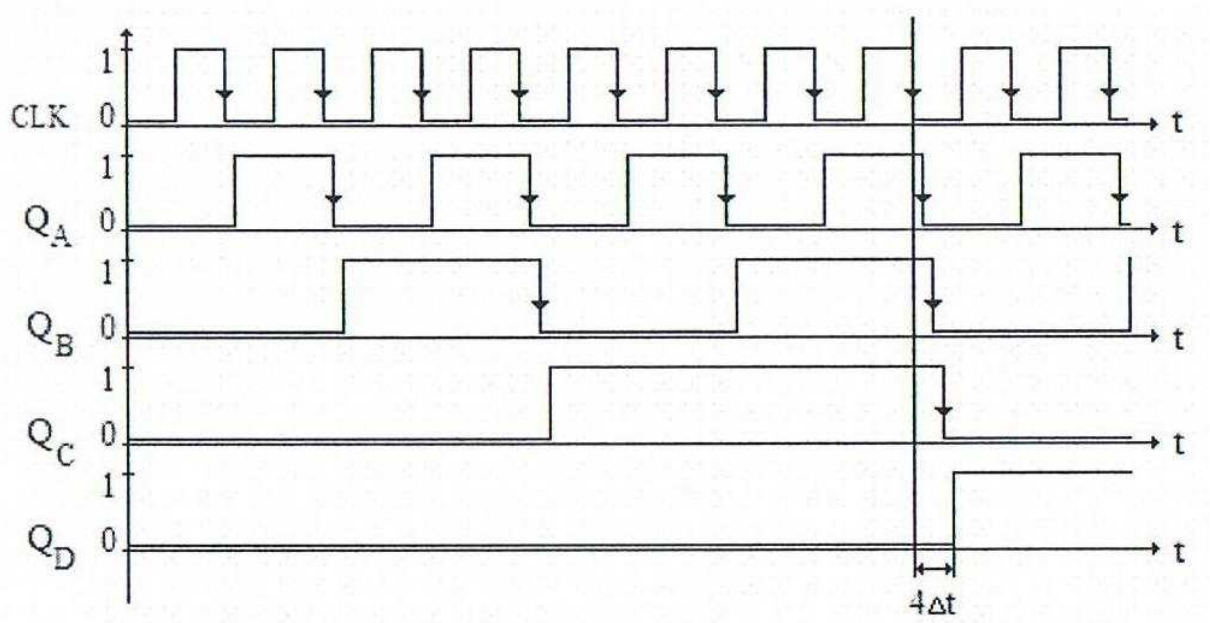
Ugyanezt a hálózatot D tárolóval még egyszerűbben meg lehet valósítani. A léptetés az órajel felfutó élénél következik be, a J-K tárolóval ellentétben.



Bináris aszinkron számláló



Mindegyik tároló J és K bemenete logikai 1-re van kötve, ezért a kimenet minden lefutó élnél ellentétesre változik. Aszinkron a hálózat, mivel a tárolók billenése nem egyszerre, hanem egymás után következik be. Működését az idődiagramban nyomon követhetjük.

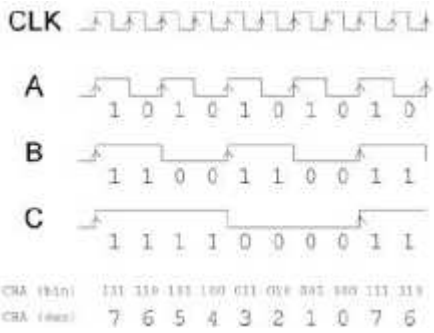


Látható, hogy a számláláson felül frekvenciaosztásra is lehet használni a hálózatot. Minden tároló a bementére jutó jel frekvenciáját felezi.

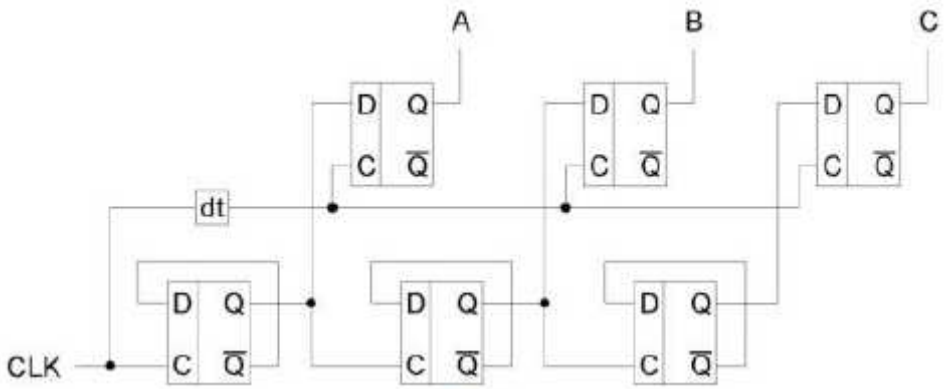
12. tétel: Szinkron, aszinkron számlálók.

Aszinkron számlálók

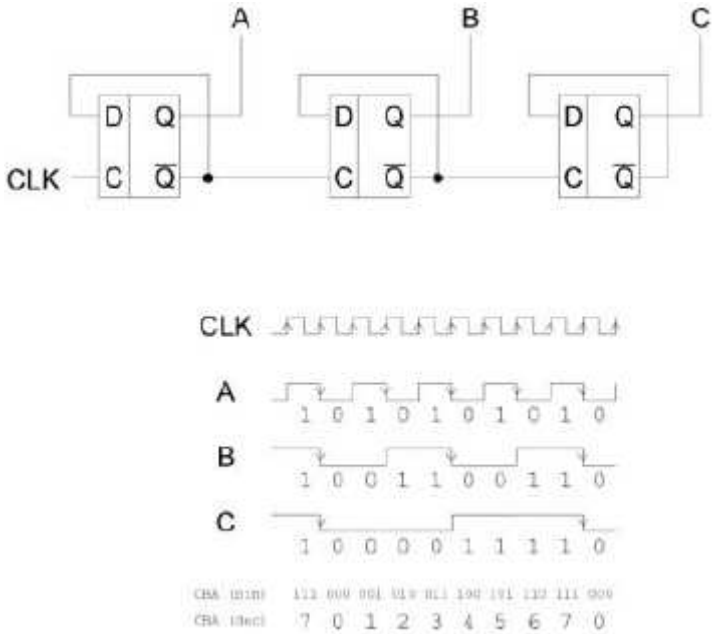
Írjuk fel egy frekvenciaosztó áramkör egymást követő kimeneti kombinációit, majd olvassuk le az eredmény.



Láthatjuk, hogy a frekvenciaosztó egyben számlál is, méghozzá kettes számrendszerben 7-től 0-ig lefelé, majd újrakezdi. Az áramkör teljes neve modulo 8-as aszinkron lefelé számláló., mivel 8 értéket tud megkülönböztetni, illetve aszinkron, mert a tárolók nem szinkronban működnek, ahogy az órajel fut végig az elsőtől az utolsóig, úgy működteti a számlálókat. Ennek az az eredménye, hogy minden számláláskor rövid ideig hibás kombinációs jelennek meg a kimeneten. Ezt a hibát kiküszöbölhetjük, ha egyszerre működésbe lépő tárolókkal szinkronizáljuk a kimeneteket. Ekkor a számlálók csak akkor váltanak, ha a számlálón már végighaladt az órajel.



Amennyiben a negált kimenetről visszük tovább az órajelet, akkor aszinkron felfelé számlálót kapunk. A változás csupán csak annyi, hogy az órajel lefutó élénél lép működésbe a második, majd ugyanígy a harmadik tároló. Az előbbiekben ismertetett probléma itt is fennáll, a javítása hasonló módon történhet.

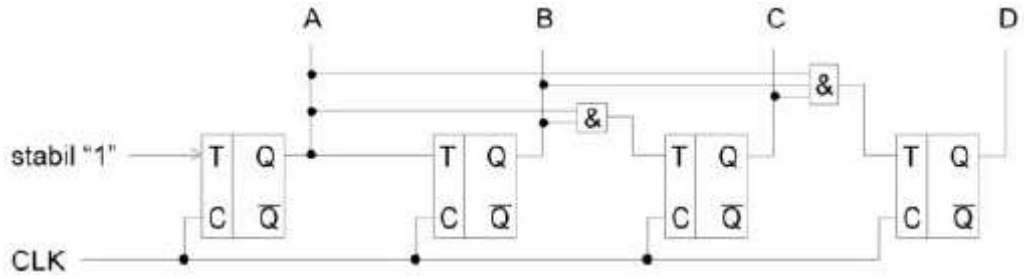


Szinkron számlálók

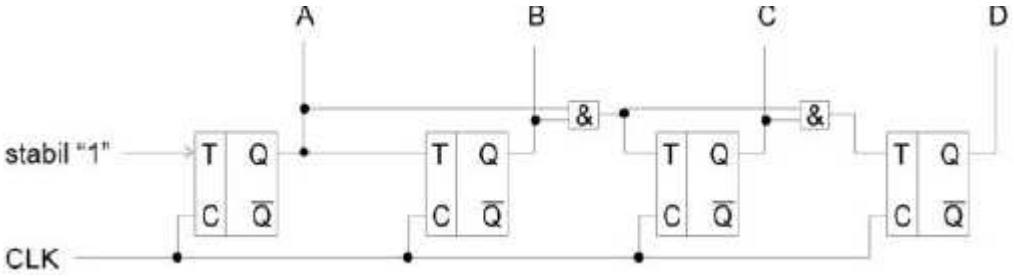
Az aszinkronitásból eredő hibát szinkron számlálók alkalmazásával is kiküszöbölhetjük. Ezeknél az eszközöknél a külső órajel egyszerre vezérli az összes tárolót. A szinkron felfelé számláló tervezésénél a következő a tervezés alapja: egy tetszőleges bináris számot úgy tudunk eggyel növelni, hogy megváltoztatjuk a legkisebb számjegyét, továbbá azokat a számjegyeket, amelyekre igaz, hogy a náluk kisebb helyiértéken mindenütt 1 áll.

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

Tudjuk, hogy a T tároló akkor változtatja meg az állapotát, ha az órajel felfutó élekor 1 van az adatbemenetén. Az első T tároló adatbementére fix 1-et kötünk, így minden órajel ciklusban váltani fog. A többi tároló adatbementére a náluk előrébb álló egységek kimenetének és kapcsolatát vezetjük, így, ha mindegyik 1, akkor a flip-flop váltani fog.



A kapcsolás hátránya, hogy minden tároló elé annyi bemenetű és kapu kell, amennyi tároló szerepel előtte a sorban, és minden és kapuhoz külön kell elvinnünk a vezetékeket, ez a párhuzamos átvitel. A problémát a soros átvitellel küszöbölhetjük ki, itt felhasználjuk az alacsonyabb szinteken álló ÉS kapuk eredményét is. A soros átvitel hátránya, hogy a jelterjedést végig kell várnunk, és csak utána folytathatjuk a számlálást. Ez a jelterjedés viszont nem okoz hibás kimeneti kombinációt, mivel a tárolók egyszerre nűködnek.



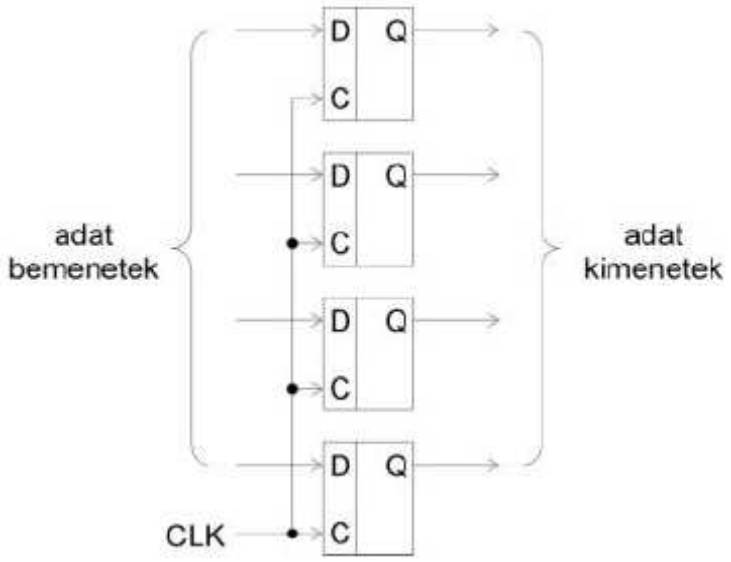
A megismert eszközökön kívül még számos számlálót lehet akár tokozott formában is beszerezni, vannak még decimális és BCD kóddal működők is. Léteznek 8, 16 bitesek és kaszkádolhatók is, tehát a kapacitást szinte korlátlanra növelhetjük.

13. tétel: Regiszterek, kódolók, dekódolók, multiplexerek, demultiplexerek.

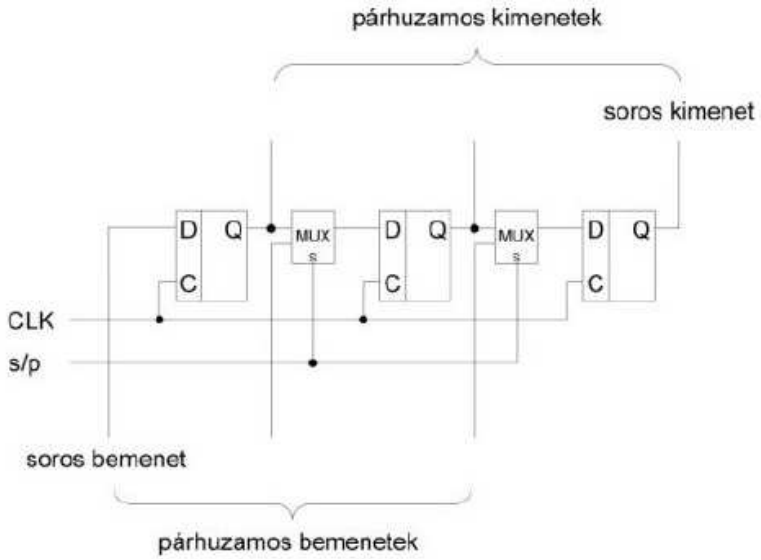
A következőkben olyan kombinációs hálózatból felépülő áramkörök kerülnek bemutatásra, amelyekkel sokszor találkozhatunk digitális eszközökben. Gyakori alkalmazásuk miatt IC-be integrálva, készen is megvásárolhatjuk őket.

Regiszterek

A tároló regiszterek több bites számok tárolására alkalmasak. Az alábbi ábrán egy 4 bites, 4 darab párhuzamosan kötött D tárolóból kialakított regiszter látható. Az órajel felfutó élénél elraktározza az adatbemenetre adott számot, amelyet a kimeneten bármikor leolvashatunk. A számot a következő órajel felfutásig tárolja, ekkor az újat írja be.



Vannak olyan regiszterek, amelyek a tárolt szám jegyeit képesek egy vagy több helyiértékkel eltolni akármelyik irányba, ezeket léptető agy shift regisztereknek nevezzük. Ezekkel megtehetjük, hogy a számjegyeket egyetlen kimeneten, egymás után olvassuk ki, ezt soros kiolvasásnak nevezzük. Létezik hasonló módon soros beírás is, ezeknek egy bemenete van. Az alábbi képen egy univerzális léptető regiszter látható, amellyel sorosan és párhuzamosan is be tudunk írni, illetve ki tudunk olvasni. Ha az s/p bemenetet egyre állítjuk, akkor az órajel felfutó élére a párhuzamos bemenetekről töltődnek be az adatok. Ezeket a párhuzamos kimeneten rögtön ki is olvashatjuk. Az s/p 0-ra állításával az egymást követő tárolók ki és bemeneteit köti össze, a felfutó élkor az első tároló értéke átkerül a másodikba, és így rendre tovább. Az órajel ütemekkel a számjegyek sorra kiesnek a soros kimeneten, de közben ugyanígy beolvasás folyik a soros bemeneten. A sorosan beolvasott számot természetesen bármikor párhuzamosan is kiolvashatjuk.

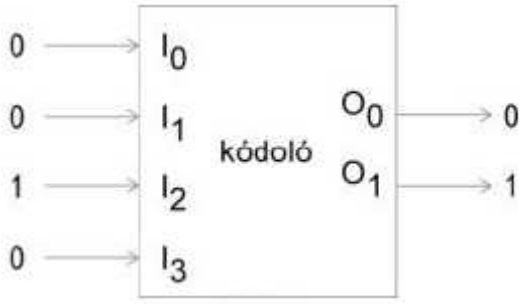


Kódátalakító egységek

A digitális technikában különböző kódokat használnak. Azt tudjuk, hogy a számjegyeket csak bináris kódban tudjuk kezelni. Ha mégis tízes számrendszerbeli számokkal akarunk dolgozni, akkor a BCD kódot választjuk, ahol minden egyes számjegyet külön-külön, kettes számrendszerben írunk fel. A kódátalakító egységek egyik kódból a másikba lépnek át.

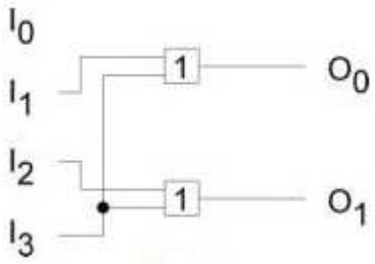
Kódoló

A köznapi nyelvben kódolónak hívják azt a kódátalakító eszközt, melynek bemenetei közül csak az egyiken szerepelhet 1, a többi 0, a kimenetein pedig annak a bemenetnek a száma jelenik meg, ahol binárisan 1 van.



Az n számú bemenettel rendelkező kódolónak 2^n bemenete van.

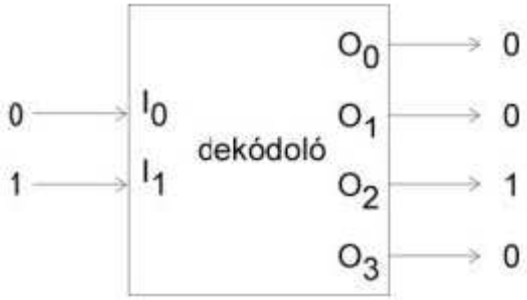
I_3	I_2	I_1	I_0	O_1	O_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
az összes többi kombinációra:				-	-



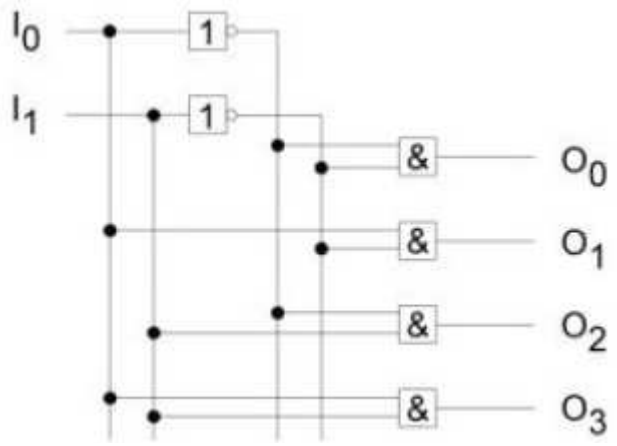
A kapcsolás érdekessége, hogy I_0 bemenetre nincs szükség, mivel erre kizárásos alapon következtetünk.

Dekódoló

A kódolónak éppen a fordítottja, bemenetére egy bináris számot vár, és a számmal megegyező sorszámú kimenetre rak 1-et, amíg a többire 0-t.



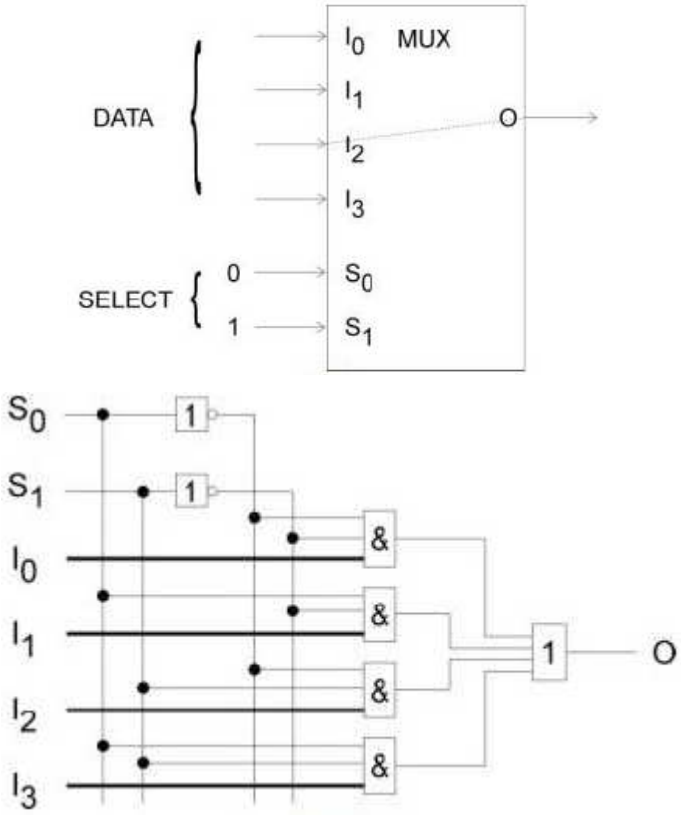
I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



Adatút választó eszközök

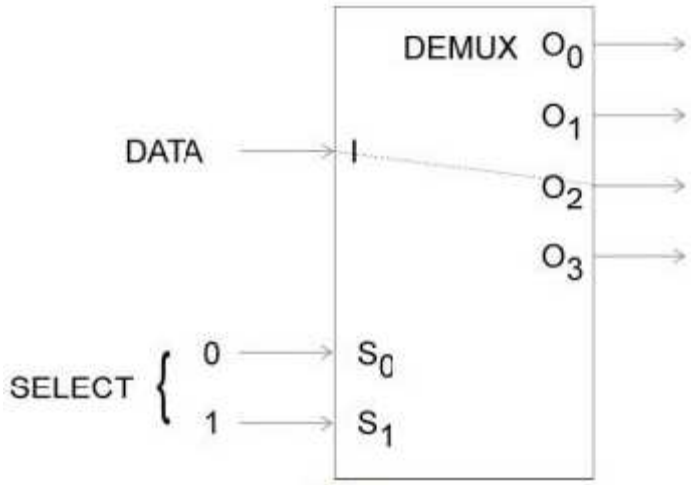
Multiplexer

A multiplexer a SELECT bementre egy bináris számot vár, és az ezen számmal megegyező sorszámú adatbemenetet összeköti a kimenettel.



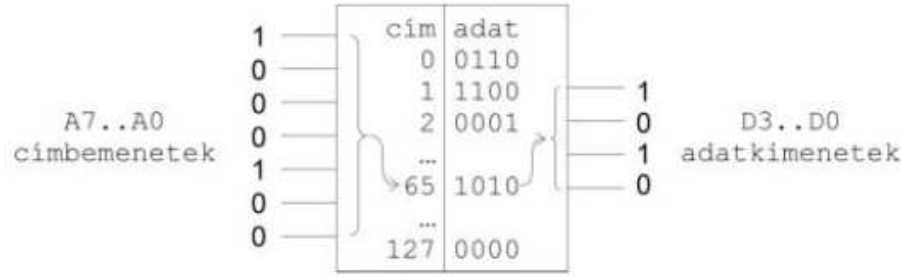
Demultiplexer

Egyetlen adatbemenete van, és ezt a SELECT bemenetek által választott kimenettel köti össze.



14. tétel: Memóriák

Míg a D tároló egyetlen bit, a regiszter pedig egy bináris szám tárolására alkalmas, addig a memóriákban több ilyen számot is tárolhatunk. Az adatokat számozott rekeszekben tároljuk. Egy adott rekesz tartalmát úgy tudjuk kiolvasni, hogy a címbemenetekre adjuk annak sorszámát, mire az adatkimeneten a hozzáférési idő elteltével megjelenik a rekeszben tárolt szám.



Az egy rekeszben tárolható bitek száma adja meg a memória szóhosszúságát. A memória kapacitása pedig a beírható szavak számát jelöli. N darab címvezeték esetén a memória 2^n rekeszt tartalmaz.

A memóriákat több féleképpen sorolhatjuk csoportokba. A tárolás jellege alapján két típust különböztetünk meg:

- ROM, csak olvasható memória, ezeket az eszközöket a memóriát tartalmazó rendszer fejlesztése során töltjük fel adatokkal, a rendszer működtetésénél már csak olvasunk belőlük. A beírt adatok az áramellátás megszűnése után, akár évekig is megmaradnak. A beíráshoz sokszor ún. égetőáramköröket alkalmazunk. Vannak ROM-ok, amelyek csak egyszer programozhatók, újraírásukra nincsen lehetőség. Ezek az ún. OTP áramkörök, amiket általában sorozatban gyártott eszközökben alkalmaznak, miután teljesen lezárult a fejlesztés folyamata. Az újraírhatóság elengedhetetlen a prototípusok fejlesztésénél, ezért az ún. EPROM-ok a beírás után a chip tetejére irányított UV fénnyel törölhető, és újra felhasználható. Az EEPROM-ok elektromosan törölhetők és írhatók is. Ezek egyik fajtája a ma is divatos FLASH memória, amely technológiai jellemzői szerint ROM, felhasználása miatt azonban a következő csoportba sorolható.
- RWM, írható és olvasható memória, melynek tetszőleges rekeszét tetszés szerint írhatjuk és olvashatjuk is. Kikapcsolás után az adatok elvesznek, ezért illanó memóriának is hívjuk őket. Az adatvezetékeik kétirányúak. A beírás folyamatát általában egy WE, (Erite Enable) bemenettel vezéreljük, ezzel döntjük el, hogy írni vagy olvasni kívánjuk az adott rekesz tartalmát. Kétféle technológiával készítenek

ilyen eszközöket, a statikus memóriákban sok flip-flop tárolja az adatokat, a dinamikus memóriák pedig mátrix alakban elhelyezett kondenzátorok segítségével raktározza el a biteket. Ha egy kondenzátor fel van töltve, akkor az adott bit 1-es, ha nincs, akkor 0 értékű. Mivel ezek a kondenzátorok nagyon kicsik, gyakran ki kell olvasni, és újra kell írni őket, azaz frissítésre van szükség, különben a töltések elszivárognak, és elveszik a tárolt információ. A korszerű dinamikus memóriák egy speciális áramkört is tartalmaznak, amely időről időre automatikusan elvégzi a frissítést.

A ROM-ok ellenpárjaként a köztudatban nem az RWM, hanem a RAM szerepel. A valóságban egy RAM is lehet ROM, a rövidítés ugyanis azt takarja, hogy minden rekesz kiolvasási ideje megegyezik egymással. Ezzel szemben a soros hozzáférésű memória (SAM) legutolsó rekeszéhez csak az összes előtte álló adat kiolvasása után férhetünk hozzá.

Az fent említett memóriák mellett vannak tartalom szerint címzett (asszociatív) memóriák is, amelyek a tárolt információ egy része alapján választják ki a megfelelő adatot, az információ egy részéről asszociálnak az egészre.

<u>Írási képesség szerint:</u>

- High end
  - A memóriát processzor segítségével írhatjuk egyszerűen és gyorsan
  - Pl.: RAM
- Middle range
  - A memóriát processzor segítségével írjuk, de az írási folyamat lassabb
  - Pl.: FLASH, EEPROM
- Lower range
  - Speciális berendezésre van szükség az íráshoz
  - Pl.: EPROM, OTP-ROM
- Low end
  - A bitek csak a gyártás során írhatók le
  - Pl.: Mask-programmed ROM

<u>Tárolási teljesítmény szerint:</u>

- High end
  - Lényegében soha nem vesztik el tartalmukat

    - Pl.: Mask-programmed ROM
- Middle range
    - A tartalmukat a tápfeszültség kikapcsolása után napokig, hónapokig vagy akár évekig is mergőrzik.
    - Pl.: NVRAM
- Lower range
    - Tartalmukat mindaddig megőrzik, amíg a tápfeszültség ellátás meg nem szűnik.
    - Pl.: SAM
- Low end
    - Tartalmukat már az írási folyamat befejezése után kezdik elveszíteni
    - Pl.: DRAM

Cache memória

- Általában SRAM-ból tervezik
    - Gyorsabb, de sokkal drágább, mint a DRAM
- Általában a processzorral egy chipen található
    - Kicsi a rendelkezésre álló hely
    - Gyorsabb hozzáférés
- Működése:
    - Kérés a rendszermemóriához történő hozzáférésre
    - Először ellenőrzik, hogy a cache tartalmazza-e a kért információt
    - Megtaláljuk a cache-ben, gyors hozzáférés

15. tétel: Számábrázolási formák.

Létezik:

- előjeles abszolút értékes ábrázolás,
- komplemens ábrázolás,
- ofszet, vagy többletes ábrázolás,
- BCD ábrázolás,
- fixpontos ábrázolás,
- lebegőpontos ábrázolás.

Előjeles abszolút értékes ábrázolás: egy bináris szám lehet pozitív, vagy negatív éppen úgy, mint egy decimális szám. Az előjel ábrázolása 0 és 1 számokkal valósulhat meg. 0 a pozitív számot, 1 a negatív számot jelöli. Csak akkor egyértelmű a szám megadása, ha a szóhossz rögzített. Példa:

$$+11810 = 0\ 111\ 01102$$
$$-11810 = 1\ 111\ 01102$$

Jellemzők:

- a nulla kétféleképpen ábrázolható,
- a legkisebb ábrázolható szám −127, a legnagyobb 127 (8 bites ábrázolás esetén).

Komplemens ábrázolás: a szám előjelét nem kell elkülönítve kezelni. Két módszer létezik:

- alapszám: alap (abszolút) érték,
- csökkentett alap: alap értéke mínusz egy:
    - 9-es komplemens: a 10-es alap csökkentett alapja,
    - 1-es komplemens: a 2-es alap csökkentet alapja.

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	$N_{k1} = 2^n - 1 - N$

Előjeles számok komplemens ábrázolása

1-es komplemens (1-es kiegészítős számábrázolás)

- az első bit az előjel bit, a számot most 7 biten ábrázoljuk,
- a pozitív számok ábrázolása megegyezik az előjeles abszolút érték ábrázolással, pl.:

$$+25_{10} = 0001\ 1001_2$$

- egy negatív szám az azonos abszolút értékű pozitív szám komplemense (1-es kiegészítője),
- Példa:

$$-25_{10} = 0001\ 1001_2 \text{ (előjeles abszolút)}$$
$$-25_{10} = 1110\ 0110_2 \text{ (egyes komplemens)}$$

Jellemzők:

- a nulla kétféleképpen ábrázolható,
- a legkisebb ábrázolható szám −127, a legnagyobb 127 (8 biten),

2-es komplemens (2-es kiegészítős ábrázolás)

- a pozitív és a negatív bináris számok legelterjedtebb ábrázolás,
- Neumann János „találmánya”,
- a pozitív számok ábrázolása azonos az előjeles abszolút érték és az 1-es komplemens számábrázolásával,
- negatív számok ábrázolása: 1-es komplemenshez 1-et hozzáadva kapjuk meg
- Példa:

$$-25_{10} = 0001\ 1001_2 \text{ (előjeles abszolút)}$$
$$-25_{10} = 1110\ 0110_2 \text{ (1-es komplemens)}$$
$$-25_{10} = 1110\ 0111_2 \text{ (2-es komplemens)}$$

Jellemzők:

- a nulla egyértelműen ábrázolható,
- a legkisebb ábrázolható szám −128(!), a legnagyobb 127.

Ofszet bináris (többletes) ábrázolás

Vannak olyan áramkörök, amelyek csak pozitív számok feldolgozására alkalmasak. A legnagyobb helyi értéket a pozitív szám bitjeként értelmezi. Ilyen esetben a számtartomány közepét jelöljük ki 0-nak (eltolt ofszet ábrázolás). 8 bites számmal 0…255 tartomány fogható át, 2-es komplemens ábrázolásban −127…128 tartomány írható fel 8 bites ábrázolással. Az ofszet bináris számábrázolásra való áttéréshez összeadással eltoljuk a számtartományt 128 hozzáadásával a 0-tól 255-ig terjedő tartományba. A 128 feletti számok eszerint pozitívak, a 128 alattiak pedig negatívak. A 128 képezi a tartomány közepét, tehát a 0-t. 128 hozzáadását egészen egyszerűen a 2-es komplemens formában értelmezett szám előjelbitjének negálásával végezzük el.

BCD (Binary Coded Decimal – binárisan kódolt decimális)

Számkódok használatakor a decimális szám jegyeit adjuk meg 4-4 bittel. Ez például akkor hasznos, amikor eredményeket, adatokat kijelzésre alkalmassá, az ember számára könnyedén értelmezhetővé kell alakítani. Négy bittel 2n = 24 = 16 különböző kódszót lehet előállítani, ebből 10-et tekintünk megengedettnek (vagyis a BCD-ben működő áramkörök nincsenek teljesen kihasználva, ez az ára a könnyebb kezelhetőségnek). A szokásos BCD kódnak nevezett 8-4-2-1 súlyozású kód készletét 0000-tól 1001-ig terjedő bináris számok alkotják (1001-nél nagyobb számok tiltottak, megjelenésük a rendszerben hibát okozhatnak). Pl.:

+30110 = 0000 0011 0000 0001

Negatív számok BCD ábrázolása: 9-es, vagy 10-es komplemens kóddal történik, pl.:

−30110 = 1001 0110 1001 1000 (9-es komplemens)

−30110 = 1001 0110 1001 1001 (10-es komplemens)

Fixpontos ábrázolás

Ebben az ábrázolási módban a bináris pont helye – ami a bal oldalon található egészeket választja el a jobb oldalon található törtektől – rögzített, és a számokat többnyire kettes komplemens kódban ábrázolják.

előjel (2n – 1) | egész rész (2n – 2) | bináris pont (nincs ábrázolva) | törtrész (20)

A számok ábrázolásának két fontos jellemzője van a felhasználás szempontjából:

- ábrázolandó számok nagysága,
- az ábrázolás pontossága.

A két jellemző az alkalmazott regiszter mérettől, szóhossztól és a bináris pont helyétől függ. Ha a bináris pontot balra toljuk el, akkor:

- a számábrázolási tartomány csökken,
- az ábrázolás pontossága nő,
- ha a bináris pont a regiszter bal szélén van, akkor a szám fixpontos tört.

Ha pedig a bináris pontot jobbra toljuk el:

- a számábrázolási tartomány nő,
- az ábrázolás pontossága csökken,
- ha a bináris pont a regiszter jobb szélén van, akkor a szám fixpontos egész.

Lebegőpontos ábrázolása

A számokat ebben az esetben normalizált alakban használjuk:

$$N2 = \pm M \cdot 2^{\pm E}, \text{ ahol}$$

N2 az ábrázolandó bináris szám, M az ún. normalizált mantissza, E pedig a karakterisztika. A mantisszát leggyakrabban előjeles abszolút értékes formátumban tárolják. A normalizálásra kétféle gyakorlat terjedt el:

Törtre normalizálás

A bináris pontot addig toljuk el, amíg a mantissza értéke ˝ és 1 közötti értékű nem lesz. Pl.:

$$N2 = 0{,}00010110012 = 0{,}1011001 \cdot 2^{-3}$$

Mivel a 2-1 helyi értéken lévő bit mindig 1 értékű, ezért a szám eltárolása előtt kiveszik. Ezt implicit bitnek hívják. Így a tárolt mantissza (m) értéke: 01100100.

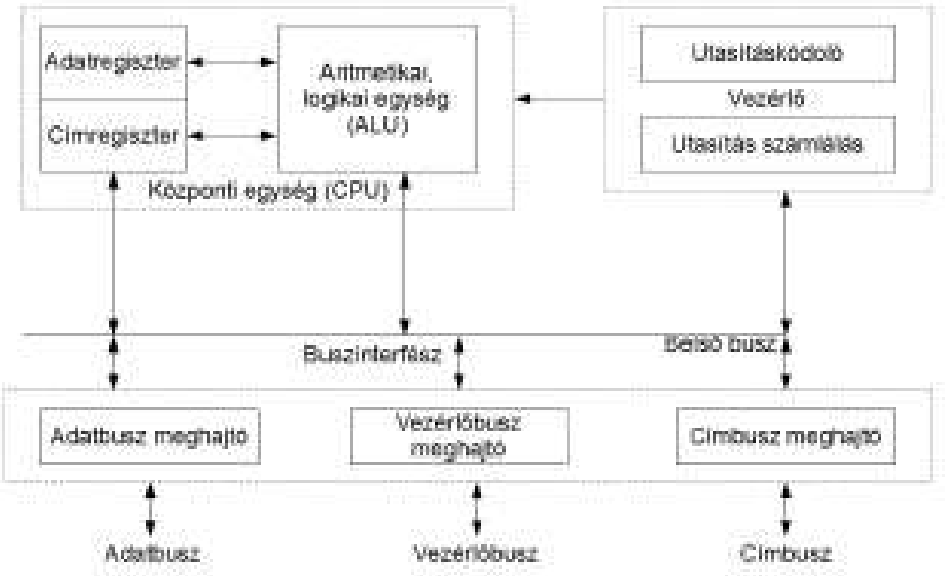
Egészre normalizálás

Ez esetben a normalizált mantissza értéke 1 és 2 közé esik, pl.:

$$N2 = 0{,}00010110012 = 1{,}011001 \cdot 2^{-4}$$

Itt az egészek helyén áll mindig egy, ezért tárolása szükségtelen. A tárolt mantissza megegyezik az előzővel (m = 01100100). Természetesen műveletvégzés előtt mindkét esetben a nem tárolt biteket vissza kell helyezni, hiszen ellenkező esetben hibás eredményt kapnánk. A karakterisztikát ofszet bináris számábrázolási formában tárolják, hogy ne kelljen az előjelet ábrázolni.

16. tétel: Mikroprocesszorok általános felépítése.



Három fő működési részegysége van:

- Központi (végrehajtó) egység
- Vezérlőegység (kontroller)
- Buszillesztő (interfész) egység

A központi egység hajtja végre az aritmetikai és a logikai utasításokat. A műveletvégzések operandusai vagy az adat-, illetve címregiszterekben találhatók, vagy a belső buszon át jutnak a központi egységbe.

A vezérlőegység az utasításkódolóból és az utasításszámlálóból áll. Az utasításszámláló egymás után behívja a program utasításait. Az utasításdekódoló elindítja az utasítás elvégzéséhez szükséges műveleteket. A vezérlőegység feladatát egy szekvenciális (sorrendi) hálózat látja el, amelynek igazságtáblázatát az új mikroprocesszorokban ROM tárolja. Az ilyen ROM-ok tartalmát mikroprogramoknak is nevezik. A külső utasítások ebben az esetben meghatározzák a mikroprogramba való belépés címét.

A program indításakor az utasításszámlálót a kezdő címre kell állítani. Ez a cím a címbuszon át a tárba jut. Ha a vezérlőbuszon olvasási utasítás érkezik, a címnek megfelelő tároló tartalma megjelenik az adatbuszon, és az utasításdekódolóban tárolódik. Az utasításdekódoló elindítja az utasítás elvégzéséhez szükséges műveleteket. Az utasításdekódoló az utasítás végrehajtása után az utasításszámlálót a következő utasítás címére állítja.

17. tétel: Busz fogalma, jellemző jelei és működésük.

a számítógépekben az egyes ki és beviteli egységek közötti adatáramlás elektromos vezetőkön keresztül valósul meg. Ezeket a vezetékeket csoportokba foglalják, ezt nevezik síneknek.

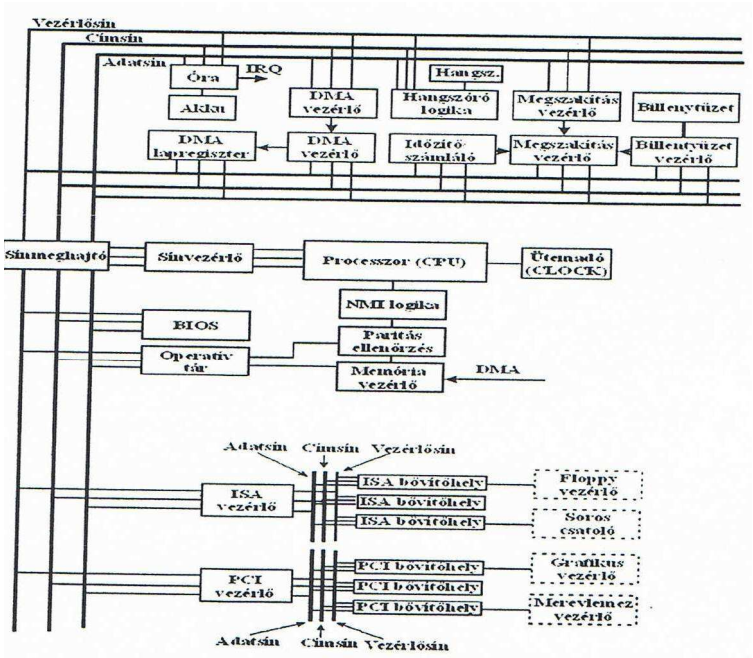
A mikroszámítógépekben három sínt különböztetünk meg:

- adatsín: az egységek számára szükséges adatok továbbítására szolgál, általában a processzor által meghatározott számú vezetéket tartalmaz, rendszerint 8, 16, 32 vagy 64 bit szélességű.
- címsín: az egységek kijelölése címük alapján történik, ez a sín ezen adatok továbbítására szolgál. A címsín szélességét minden esetben a processzor határozza meg, a mai processzorok esetén szélessége általában 32 bit.
- vezérlősín: a részegységek működési módját ennek a sínnek a használatával oldják meg. Itt nem igaz, hogy egymással párhuzamosan futó vezetékek alkotják. Minden olyan jel ide tartozik, amely állapotának vagy valamely egység működésének megváltozását vonja maga után. Csoportosításuk:
    - adatátvitelt vezérlő jelek:
        - memória/IO
        - írás/olvasás
        - szó/bájt
        - címtárolás
        - adattárolás
        - kész
    - megszakítást vezérlő jelek
        - megszakítást kérő jelek
        - megszakítást elfogadó jelek
    - sínvezérlő jelek
    - szinkronizációs jelek
    - egyéb jelek

A számítógép minden egysége ugyanazt a sínrendszert használja a kapcsolat megvalósítására. Nagyon fontos a helyes működés kialakítása érdekében, hogy az alábbi három feladatot elvégezzük:

- meg kell oldani, hogy azok az eszközök kapcsolódjanak össze, amelyeknek szükséges, tehát ki kell jelölni a résztvevőket.

- fontos eldönteni az adatmozgás irányát, tehát ki kell jelölni a küldőt és az adót. Erre a vezérlősín jeleit használják fel.
- nagyon fontos, hogy az adatokat és a címeket minden, a kommunikációban részt vevő eszköz ugyanakkor értelmezze, tehát össze kell hangolni a működésüket. Ezt a folyamatot nevezik szinkronizációnak.



A mikroszámítógép vázlatos felépítése

A sínrendszer felépítése

A processzor belsején bináris adatokkal dolgozó modulok találhatók. A mikroszámítógépekben megkülönböztetünk:

- belső sínrendszert, amely a processzor belső egységeinek összekapcsolásáért felelős
  - 1 sínes rendszerek: esetében a processzor közös vezetéket használ az adat és a címjelek továbbítására. Természetesen ez egy időben nem történhet meg, ezért időben szét kell választani. Ezt a struktúrát multiplexelt sínrendszernek nevezik, egyszerűbb processzoroknál figyelhető meg. Azt, hogy éppen adat vagy cím van-e a sínen, a vezérlőjel határozza meg. Ezzel a rendszerrel a kivezetések számát jelentősen csökkenteni lehet, ezzel pedig a méret is csökken. A gyártók kezdenek szakítani ezzel a felépítéssel, mivel a teljesítménynövekedés iránti igény egyre jelentősebb.

- 2 sínes rendszerek: a legelterjedtebbek, ilyenek a mai PC-kben használt processzorok is. Külön sín szolgál a cím és adatjelek továbbítására. Előnye, hogy cím és adattovábbításra egy időben van lehetőség.
- 3 sínes rendszerek: a leggyorsabbak. A 2 sínes rendszerek egyik nagy hátránya, hogy az írás és olvasás csak más ciklusban hajtható végre. A 3 sínes rendszer két adatsínt használ, az egyiken az írás, a másikon az olvasás zajlik, ami egy időben is lehetséges. Sebességnövekedés érhető el vele, a RISC processzorok alkalmazzák.

- külső sínrendszert, amely a processzor és a memória, valamint az I/O eszközök közötti kapcsolatot valósítják
  - helyi busz: kapcsolja össze a processzort és a hozzá közvetlenül tartozó eszközöket. Közvetlenül a processzor hajtja meg, azon eszközök kapcsolódnak rá, amelyeknél fontos a maximális sebesség. Az eszközök száma és a maximális sínhossz is korlátozott.
  - rendszersín a processzort és általában az I/O eszközöket kapcsolja össze. A processzor után egy sínmeghajtót helyeznek el, aminek köszönhetően a sínre kapcsolt eszközök és a sínhossz is jelentősen növekedhet. Hátránya, hogy a sebességet lassíthatja, ez azonban nem jelentős. A rendszer maximális sebességét a leglassabb eszköz határozza meg.
  - memóriasín: amely kifejezetten a processzor és az operatív tár közötti kapcsolat kialakításáért felelős. Nem minden rendszerben különül el kifejezetten a rendszersíntől, azonban a teljesítmény és az adatbiztonság növeléséhez célszerű elválasztani.

A sínrendszer működése

A digitális rendszerekben az adatátvitel és ezzel együtt a sínhez kapcsolódás csak megadott rendszerben történhet. Az adatátvitel vezérlését két különböző ódon lehet megvalósítani:

- szinkron vezérlésről beszélünk akkor, ha az események rögzített időpontban vannak és a sínre kapcsolódó eszközök ugyanazt az órajelet használják a működés vezérléséhez. Az órajelnek az éleit használják hasznosítják szinkronizációra, mivel észlelésük könnyű és lefolyásuk gyors. Az adatok adása és vétele azonos sebességgel történik, mivel a kapcsolat mindig fennáll, nem kell kapcsolatfelvétel és visszaigazolás sem. A

megoldás gyors adatátvitelt tesz lehetővé, azonban az órajel továbbítása plusz vezetékeket igényel. A megoldás folyamatos adatátvitelt igényel. Fontos, hogy csak akkor használható hatékonyan, ha a kommunikációban résztvevő egységek azonos sebességűek.

- aszinkron sínvezérlés alkalmazásakor az események időpontja tetszőleges lehet, közös órajel helyett sajáttal rendelkeznek az eszközök. A kapcsolat csak az átvitel ideje alatt áll fenn, így szükség van kapcsolatfelvételre, valamint a vételt vissza kell igazolni. Nincs szükség az órajel átvitelére, megoldható a kommunikáció nagyon lassú eszközökkel is, nem jelent problémát az sem, ha az adatátvitel nem folyamatos. Minden átvitel megelőz egy kapcsolatfelvétel.

A síneken az átviteli kapacitás függ:

- az átviteli sebességtől
- a sín bitszélességétől
- az átviteli protokolltól
- a vezérlők számától.

Különböző vezérlési módszerek vannak, melyeket korszerű sínek menet közben is válthatnak:

- dinamikus adatsín szélesség változás lehetséges
- protokoll váltás lehetséges
- blokkos átvitel lehetséges

18. tétel: Interrupt és fajtái

A program futása során előfordulhatnak olyan váratlan események, melyek hatására a processzornak meg kell szakítania a feladat végrehajtását és az újonnan kialakult helyzettel kell foglalkoznia. A váratlan esemény kiváltója lehet:

- maga a program: például 0-val való osztás esetén, vagy ha egy memóriarekeszbe vagy regiszterbe túl nagy számot akarunk tölteni (túlcsordulás). A software-es váratlan események elnevezése: *exception (kivétel)*.
- Váratlan eseményt okozhat valamelyik hardware elem is: a számítógéphez kapcsolt perifériák egyike egy előre definiált módon (pl. egy külön erre a célra fenntartott bemeneten) jelzi, hogy meg akarja szakítani a program futását, mert fontosabb kötelezője akadt (pl. adatot helyezett a bemeneti kapura, amelynek nem szabad elvesznie). A hardware-es váratlan eseményeket *interruptnak* (*megszakításnak*) nevezzük.

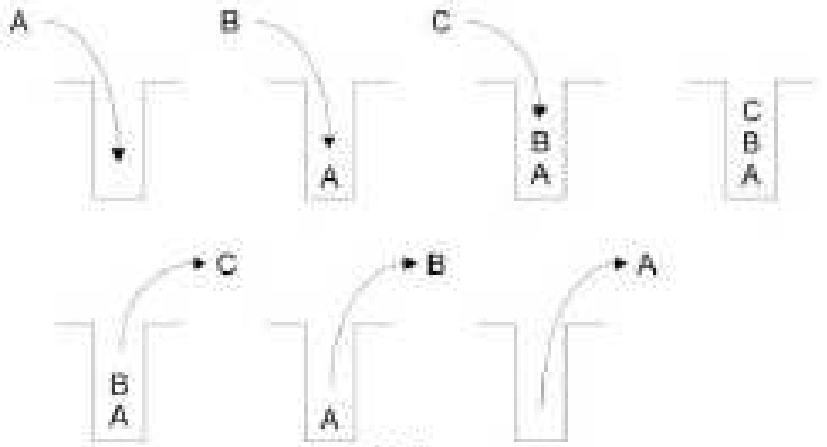
Az esemény bekövetkezési helyét tekintve két típust különböztethetünk meg:

- *Szinkron váratlan események*: azok, amelyek a program futása szempontjából jól meghatározható helyen következhetnek be. (Nem biztos, hogy bekövetkeznek, ezért váratlanok.) Ilyen pl. a nullával való osztás, és a többi software-es esemény.
- *Aszinkron váratlan események*: a program futása szempontjából előre kiszámíthatatlan helyen következnek be. A hardware-es események tartoznak ide.

A váratlan eseményekre a programozónak fel kell készítenie: minden lehetséges megszakításra egy-egy programocskát, *szubrutint* kell írnia, amely annak bekövetkezésére reagál. (A kivételek kezelését csak a bonyolultabb mikroprocesszorok támogatják. Legegyszerűbben úgy védekezhetünk ellenük, ha hibátlan programot írunk.) Vegyünk például egy épületriasztó-rendszert. A vezérlő számítógép egy óramgába visszatérő célhot futtat: várakozik. Egy mozgásérzékelő megszakíthatja ezt a folyamatot: ekkor elindít egy programrutint, amely először is megnézi, hogy élesítve van-e a rendszer, és ha igen, riasztást küld a központnak. Az épületbe belépők személyi kódjukat egy számbillentyűzetbe írják: ez is megszakítást idéz elő, de itt már egy másik program kezdi futni, amely például kinyit egy elektromos ajtót.

A váratlan eseményeket úgy kell feldolgozni, hogy a megszakított program ne vegye észre, hogy futása közben bármi is történt. Ez egy speciális memóriastruktúra: a *verem*, más néven *stack* használatával lehet elérni. A verem elnevezése jól tükrözi a működését: ez ugyanis egy olyan memóriatartomány, amelybe úgy lehet adatokat elhelyezni, hogy mindig csak azt az adatot tudjuk kiolvasni belőle, amelyik legelőbb van (97. ábra). Egy program végrehajtása közben a következőképp lehet egy másik programot „feltűnésmentesen" lefuttatni:

1. lépés: A processzor befejezi az aktuálisan végrehajtott utasítást.
2. lépés: A programszámláló és a regiszterek tartalmát a veremben tárolja.
3. lépés: Lefuttatja a megszakítást vagy kivétek kiszolgáló szubrutint (amely szabadon dolgozhat a regiszterekkel).
4. lépés: Visszatölti a veremből a programszámláló és a regiszterek tartalmát, így az eredetileg futó program nem vehet észre semmit.
5. lépés: *Megszakítás esetén* a program végrehajtását a következő utasítástól folytatja. *Ha kivétel történt*, azt maga a program, vagyis a legutóbb végrehajtott utasítás okozta. Éppen ezért, miután a szubrutinnal megpróbálnak kijavítani a hibát, a processzor újra megkísérli végrehajtani a hibát okozó utasítást. Ha ez most sem sikerül, végzetes kivétellel állunk szemben, a program terminálódik.



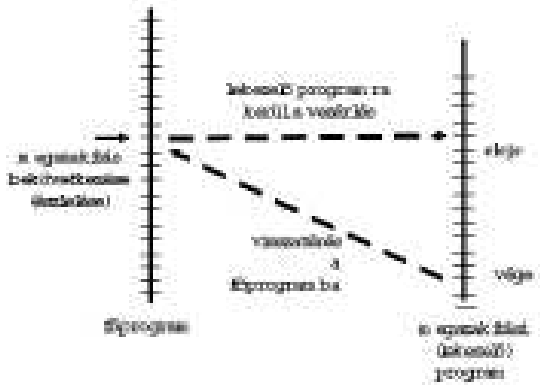
Az interrupt fajtái

- lekérdező megszakítás: a processzor a perifériás áramkörök megszakításkérő kimeneteit sorra lekérdezi. Ha valamelyiken megjelenik a megszakításkérő jel, akkor végrehajtja a megfelelő megszakítási programot és továbblép. A megszakítás elsőbbségét a perifériás áramköröknek a lekérdezési szekvenciában lévő relatív helyzete határozza meg.
- vektorizált megszakítás: a vektorizált megszakítás hatékonyabb, mint a lekérdezéses. Ebben az esetben a processzor válaszként a megszakításkérésre az adatbuszon egy perifériaazonosító címet küld ki. Ez az ún. vektor. A megszakításkérő periféria azonosítása után elvégzi a megfelelő megszakítási programot.

Programozás szempontjából ez a következőképpen valósul meg: amint egy megszakítás érvényre jut, a CPU befejezi az éppen végrehajtás alatt lévő utasítást. Ezután automatikusan generál egy LCALL utasítást, tehát a megszakítás forrásától függően meghív egy adott című szubrutint. Az LCALL utasítás elmenti a PC-t a stackbe. Ezután a PC regiszter a megfelelő címet veszi fel. Miután a megszakítás végrehajtódott a CPU visszaállítja a PC regiszter értékét, a program végrehajtása a megszakítás helyétől folytatódik. Ha ekkor is van megszakítás, a CPU a főprogramból akkor is végrehajt legalább egy utasítást, mielőtt az újabb megszakítást elindítaná. Megszakítások elsőbbségénél az IP regiszter tartalma dönt.

Megszakítások

A mikroprocesszor működése, ahogy fentebb is láttuk, utasítás-végrehajtások egymásutániságából áll. Ennek a szekvenciának kívülről történő befolyásolására nyújt lehetőséget a megszakítás (interrupt). Minden mikroprocesszor egy vagy több megszakítás-bemenettel rendelkezik. E bemenetek fizikai jelbemenetei a processzornak. Ezen bemeneteken keresztül jelezhetjük a processzornak, hogy meg szeretnénk változtatni az utasítás-végrehajtás szekvenciáját. Általában ezt azért tesszük, mert a főprogramnál aktuálisabb eseménnyel foglalkozó feladattal szeretnénk a processzort megbízni. Ez is természetesen egy program, de nem a főprogram, hanem egy attól eltérő, az illető esemény specifikus programja, ún. megszakítás-lekezelő program.



Az ábrán látható két programmemória-részlet bárhol lehet a memóriában. A főprogram mellett nyilall bejelölt időpontban (amely időpont bárhol lehet a vonal mellett) aszinkron módon megszakítást kezdeményeztünk a mikroprocesszornál.

Amennyiben a körülmények ezt lehetővé teszik (arról, hogy ez mit jelent, később részleteiben még beszélünk), úgy a szekvenciális programvégrehajtás megszakad és a program a megszakítás okául szolgáló eseményhez tartozó program végrehajtására tér át. A megszakításokat leggyakrabban külső periférikus eszköz okozza.

Az ábrákon láthatóan hasonlítanak az események a szubrutinnál látottakhoz. Két lényeges eltérést azonban észre kell vennünk:

- míg ott adott helyen és mi váltottuk ki az eseményt, úgy itt ez aszinkron módon történhet. A megszakítás bekövetkeztének konkrét időpontját ugyanis általában nem tudjuk megmondani.
- a szubrutin hívásával ellentétben a lekezelő program azonosítására vonatkozó információ a megszakítást okozó eszköztől függ, esetenként ezt az információt a megszakítást okozó eszköz küldi meg a mikroprocesszornak. A megszakításbemenetek egy része olyan, hogy annak aktívvá válását és a megszakítás elfogadását követően automatikusan egy adott címre térül el a főprogram (kvázi az adott bemenethez hozzá van rendelve a lekezelő program kezdőcíme). Ezeket a megszakításokat, tekintettel arra, hogy nem kell a processzorral külön (kívülről) közölni a lekezelő program kezdőcímét (az ún megszakítási vektort) nem vektoros megszakításoknak nevezzük.

A megszakítások másik részénél a megszakítás-bemenet olyan módon aktivizálja a processzort, hogy amennyiben a processzor fogadni tudja a megszakítást, úgy a külső eszköztől várja a lekezelő program kezdőcímére vonatkozó információt:

- a lekezelő program kezdőcímét vagy annak elégséges részét olyan utasítást, amelyben a kezdőcím is benne foglaltatik. Az ilyen megszakításokat nevezzük vektoros megszakításoknak, mivel a megszakítási programhoz a kezdőcímet (megszakítási vektort) a külső eszköz információja alapján elő kell állítani. Ha a processzor, működése során olyan fázisba jut, amikor semmiképpen nem kívánja, hogy adott utasítássorozat közben megszakítsák működését, úgy a programozó adott utasítás-sorozat alatt az utasításkészlet egy utasításával letilthatja, majd később újra engedélyezheti azt. Ez jelenti azt, amire korábban utaltunk, hogy elfogadja-e a mikroprocesszor az IT-t vagy sem.

Több megszakítás-bemenet esetén gyakran globális és szelektív tiltásra és engedélyezésre is mód van. Csaknem minden mikroprocesszor rendelkezik olyan megszakítás-bemenettel is, amelynek aktívvá válásakor a mikroprocesszor nem döntheti el, hogy elfogadja-e vagy sem a megszakítást, tehát mindenképpen érvényre jut a megszakítás. Ezt nevezzük nem- maszkolható megszakításnak. Esetenként történhetnek olyan események a mikroprocesszoros rendszer életében (pl. egy mikroszámítógépes folyamatirányítási rendszerben az irányított folyamat állapota valamilyen okból kritikussá válik), amelyek nem engedhetik meg a lekezelés késlekedését, mindenképpen azonnali beavatkozást igényelnek. Ilyen célra ideális a nem-maszkolható megszakítás használata. Azon megszakítások, amelyeknél a programozó befolyásolhatja azt, hogy a megszakítás érvényre jusson-e, vagy sem, maszkolható megszakításnak nevezzük.

Az eddigiekben ú.n. hardver megszakításról beszéltünk. A korszerűbb mikroprocesszoroknál két új megszakítás-típust vezettek be:

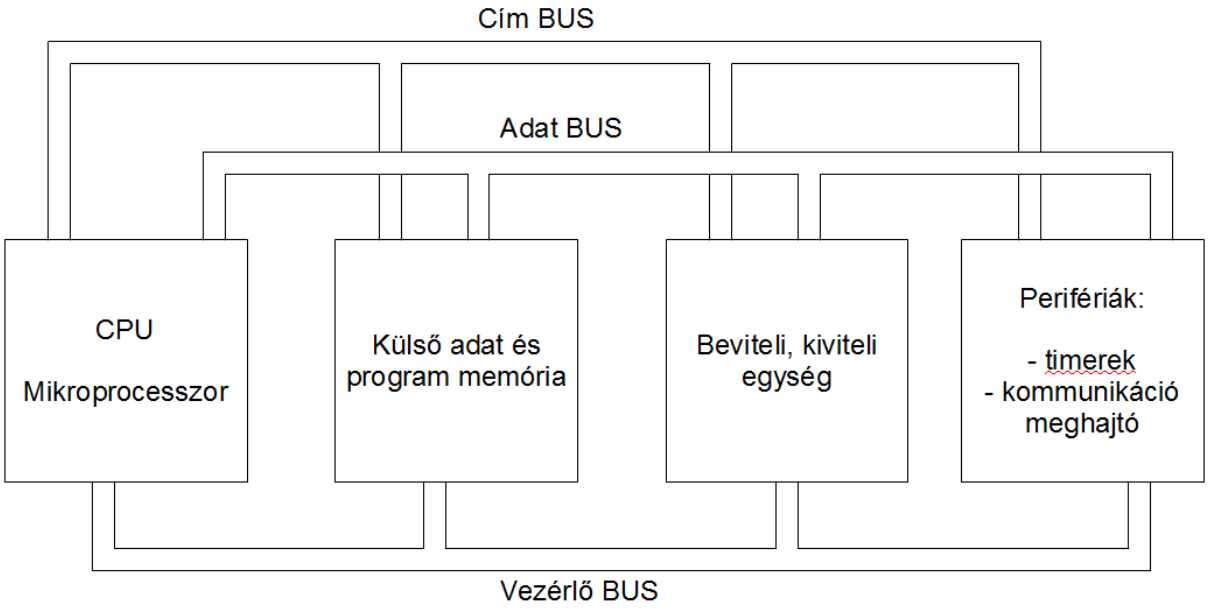
- szoftver megszakítást (utasítás útján lehet megszakítást kiváltani) elsősorban tesztelési célokra,
- belső megszakításokat, amelyek a mikroprocesszor állapotának nem kívánt változása vagy nem kívánt esemény esetén (pl. 0-val való osztás, stb.) aktivizálódnak.

A megszakítások általában különböző prioritásúak, ez azt jelenti, hogy amennyiben egy időben több megszakítás érkezik, úgy a magasabb prioritású juthat érvényre. Emellett általában kisebb prioritású megszakítás magasabb prioritásút nem szakíthat meg. A megszakítások közül a nem maszkolható megszakítás a legmagasabb prioritású.

19. tétel: Mikroprocesszor, és mikrokontroller hasonlósága, különbsége. Mikrokontroller jellemző felépítése.

A mikroprocesszorok a Neumann struktúrát míg a mikrokontrollerek a Harvard struktúrát követelik meg.

Neumann struktúra



Jellemzői:

- beviteli, kiviteli egység,
- adat- és programtároló (közös memória),
- CPU (Central Processing Unit),
- vezérlőegység,
- aritmetika, logikai egység,
- memória és periféria interface,
- mikroprocesszor (belül Neumann struktúra),
    - vezérlőegység (CPU),
    - ALU,
    - memória és periféria interface,
    - közös külső adat- és programtár.

Neumann elvek:

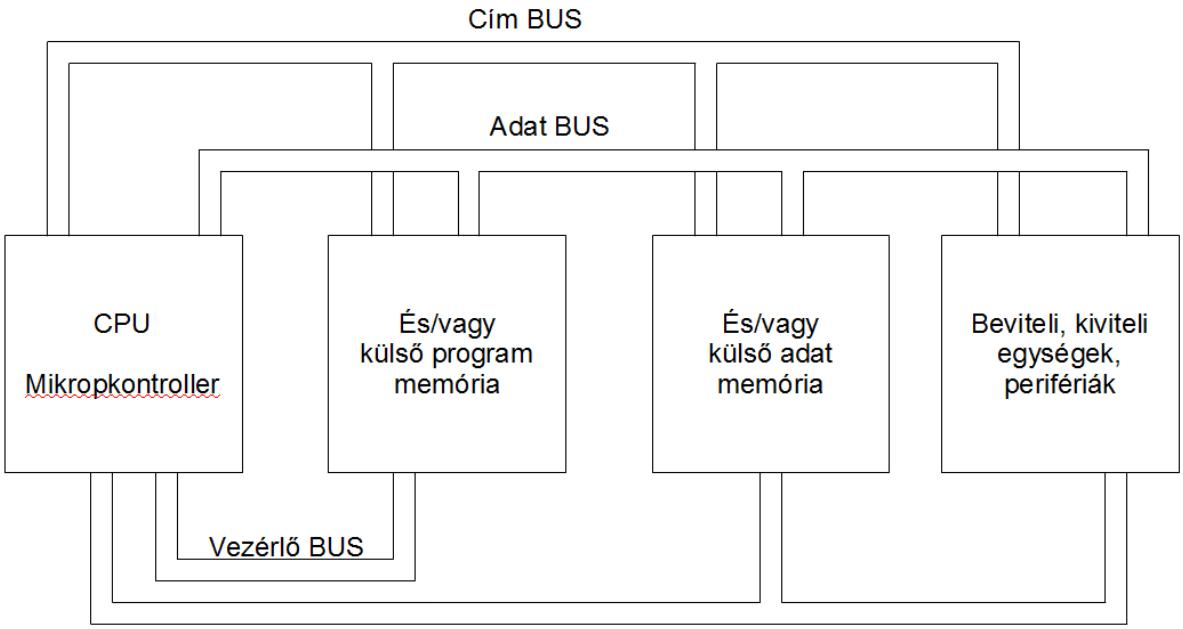
- számítógép működését tárolt program vezérli (Turing);

- a vezérlést vezérlés-folyam (control-flow) segítségével lehet leírni
- a gép belső tárolójában a program utasításai és a végrehajtásukhoz szükséges adatok egyaránt megtalálhatók (közös utasítás és adattárolás, a program felülírhatja magát – Neumann architektúra definíciója);
- az aritmetikai és logikai műveletek (programutasítások) végrehajtását önálló részegység (ALU) végzi;
- az adatok és programok beolvasására és az eredmények megjelenítésére önálló egységek (perifériák) szolgálnak;
- 2-es (bináris) számrendszer alkalmazása.

Neumann architektúra hátrányai

- „Önmagát változtató” programok (self-modifying code): Már eleve hibásan megírt program „kárt” okozhat önmagában, ill. más programokban is: „malware”=„malfunction”+„sw”. OS szinten: rendszer leállás. Pl., Buffer túlcsordulás: kezelése hozzáféréssel, memória védelemmel
- Neumann „bottleneck”: sávszélesség korlát a CPU és memória között (ezért kellett bevezetni a cache memóriát), amely probléma a nagy mennyiségű adatok továbbítása során lépett fel.
- A nem-cache alapú Neumann rendszerekben, egyszerre vagy csak adat írás/olvasást, vagy csak az utasítás beolvasását lehet elvégezni

Harvard struktúra



Jellemzők:

- vezérlőegység,
- aritmetikai, logikai egység (ALU),
- memória és periféria interface,
- külön külső és/vagy belső programtár,
- külön külső és/vagy belső adattár,
- timerek,
- periféria meghajtók, portok,

Olyan számítógéprendszer, amelynél a program*utasításokat* és az *adatokat* fizikailag különálló memóriában tárolják, és külön buszon érhetők el.

Harvard architektúra tulajdonságai

- Szóhosszúság, időzítés, tervezési technológia memória címzés kialakítása is különböző lehet.
- Az utasítás (program) memória gyakran szélesebb mint az adat memória (mivel több utasítás memóriára lehet szükség)
- Utasításokat a legtöbb rendszer esetében ROM-ban tárolják, míg az adatot írható/olvasható memóriában (pl. RAM-ban).
- A számítógép különálló buszrendszere segítségével egyidőben akár egy utasítás beolvasását, lletve adat írását/olvasását is el lehet végezni (cache nélkül is).
- „Módosított” Harvard architektúra
- Konstans adat (pl: string, inicializáló érték) utasítás memóriába töltésével a változók számára további helyet spórolunk meg az adatmemóriában
- Mai modern rendszereknél a Harvard architektúra megnevezés alatt, ezt a módosított változatot értjük.
- Gépi (alacsony) szintű assembly utasítások

Harvard architektúra hátrányai

- Az olyan egychipes rendszereknél (pl. SoC), ahol egyetlen chipen van implementálva minden funkció nehézkes lehet a különbözőmemória technológiák használata az utasítások és adatok kezelésénél. Ezekben az esetekben a Neumann architektúra alkalmazása megfelelőbb.

- A magas szintű nyelveket sem közvetlenül támogatja (nyelvi konstrukció hiánya az utasítás adatként való elérésére) – assembler szükséges

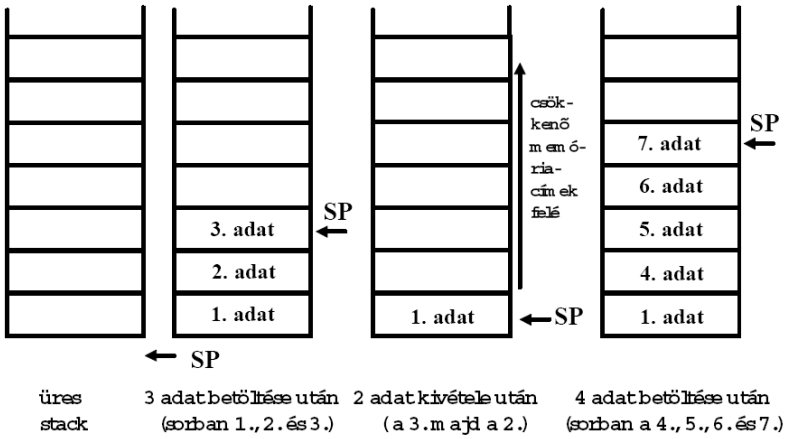
20. tétel: Stack fogalma és használata.

A stack (magyar fordítása: verem) egy logikai tárolási eszköz, amely ún LIFO (Last In First Out, az utolsó betett adat férhető először hozzá) szervezésű és átmeneti adattárolást tesz lehetővé. A stack fizikailag a memóriában helyezkedik el. Dinamikus elem, mert mérete (hossza) és helye (vagyis hogy hol helyezkedik el a memóriában) változik/változhat. A stack implementálásához egy – speciális regiszterben elhelyezkedő - mutató szükséges, mely mutatót (a regiszter tartalmát) speciálisan a stacket kezelő dedikált utasítások mozgatják, változtatják a szükségesek szerint.

A stackben többnyire regiszterek tartalmát tároljuk (mentjük el) átmenetileg. Ennek oka az, hogy a mikroprocesszor leggyorsabban a belső regiszterekkel tud műveletet végezni. A regiszterek száma viszont korlátozott. Példaként a következőt hozzuk fel: gyakran fordul elő, hogy az összes regiszter már olyan információt tartalmaz, amely még nem felülírható, de az adott részfeladat elvégzéséhez szükség van további regiszter(ek) használatához. Ekkor valamely regiszter(ek) tartalmát ideiglenesen a stackbe tudjuk kivinni (majd később a stackből a regiszter tartalmát vissza tudjuk állítani), és a regiszterbe más, aktuálisabb tartalmat tudunk betölteni. Ez a művelet általában gyorsabb és kényelmesebb, mint a memóriába írni a regiszter-tartalmat. Hiszen ilyenkor meg kellene választani a címzést, meg kellene jegyezni (tárolni kellene) a tárolási címet és a tárolt adat hosszát.

A stack működésére mutat példát a 3.ábra, ahol a kezdetben üres stackbe először 3 adategységet beteszünk, majd 2 adatot kiveszünk, végül 4 újabb adatot beteszünk. Az ábrán látható a zsák- (verem)szerű elrendezés és működés. Az említett mutatót Stack Pointernek (verem-mutató, SP) nevezzük. Helye a stack pointer regiszterben van a mikroprocesszor regisztertömbjének speciális célú regiszterei között.

A stack létrehozása az SP értékének beállításával kezdődik. Ettől kezdődően az üres stack definiálva van, és lehet bele írni, majd olvasni is onnan. A mikroprocesszoroknál a stack – az implementálás ilyen megoldás miatt - többnyire a kisebb memóriaterületek felé töltődik, így általában a RAM-memória tetején (a legnagyobb memóriacímű területen) helyezkedik el. Ez azt jelenti, hogy az SP kezdeti értékét a stack memóriában kiválasztott területének legnagyobb címére kell beállítani.



A korábbi mikroprocesszoroknál ún. belső stack létezett, vagyis a processzoron belül volt a verem, amely igencsak korlátozta annak méretét, különösen az integrálhatóság akkori fokán. Ma minden processzornál RAM-ban elhelyezhető külső stack-megoldás került implementálásra.

Az ún. hardver stack azt jelenti, hogy az SP és az egész stack-megoldás a mikroprocesszor inherens része, a stack kezelésére speciális utasítások állnak rendelkezésre, vagyis az adatok be- és kivétele igen kényelmes és egyszerű. Ezen adatmozgató utasítások egyben automatikusan „utána állítják” az SP-t is.

Ezzel ellentétben egy szoftveresen megvalósított stacknél nekünk kell mindezt "kézi vezérléssel" megoldani (egy SP'-t tétrehozni és azt amit a hardveres stack-nél a speciális utasítások elvégeznek, standard utasítások sorozatával kellene megoldani - SP-tartalom' „utána-állítás”, adatmozgatás, stb.).

21. tétel: Integrált áramkörök csoportosítása végső funkciójuk kialakítása szerint.

Elterjednek a félvezető alkatrészekés különböző komponensek:

- Mikroprocesszor
- Nagy kapacitású memóriák
- Általános célú logikai áramkörök (GPL: General Purpose Logic)
- Programozható logikai eszközök (PLD: Programmable Logic Device)
- Alkalmazás specifikus áramkörök (ASIC)
- Tervező rendszerek

Milyen eszközöket készítsünk, hogy jól használható legyen?

- Univerzális:
  - a logika egy részét nem használom (redundancia),
  - de be kell építeni, (méret növekedéssel jár)
  - tápfeszültség kell az ellátásához, (nagyobb fogyasztás)
  - sok kivezetéssel kell rendelkeznie.
- Felhasználó (alkalmazás) igényeihez kell illeszteni:
  - csak nagy darabszám esetén kifizetődő,
  - a tervezése meglehetősen bonyolult és többlépcsős eljárás,
  - a megvalósítás átfutási ideje jelentős lehet.

Következtetés:

- Nem a katalógusból akarok áramkört választani:
  - legyen beleszólásunk, hogy mi van benne
  - ne legyenek nagyok a méretek
- a túl egyedi megoldás nem jó (de a túl univerzális sem!)
- Befolyásoló tényezők:
  - ár
  - átfutási idő
  - eszközigény

CUSTOM áramkörök:

- A gyártó kész áramköröket szállít,

- amit a felhasználó specifikál:
    - tipikus logikai egységeket használva,
    - logikai specifikációt megadva,
- és a gyártó készre gyárt.

Full custom áramkörök:

- Egyedi tervezésű áramkör:
    - a tervezés alkatrész szinten (tranzisztor) történik
    - a tervező kötöttsége ebben az esetben a legkisebb
- teljesen a felhasználó specifikálja az áramkört
- átfutási idő kritikus lehet:
    - a specifikáció és a gyártás között több iterációs lépésre is szükség lehet,
    - nem mindegy a megjelenés ideje (piaci verseny!)
- csak nagy darabszám esetén kifizetődő
- A gyártónak csak annyi az egyszerűsítés, hogy nem ő tervezi meg a logikát.

Standard cellás áramkörök:

- A gyártó tipikus áramköröket (standard cella) bocsát a felhasználó rendelkezésére.
    - cellakönyvtár
- Ezekből a logikai modulokból „építkezhetünk”:
    - a tervezési- és az átfutási idő lecsökkenthető
- Hátrányok:
    - több iterációs lépésre van szükség a tervezés során
    - a működés közben derülhetnek ki bizonyos működési sajátosságok

Semi Custom áramkörök:

- A gyártó félkész áramköröket gyárt le
- Ezek önmagukban semmire sem használhatók
    - a gyártás befejezése a felhasználó dolga
- A felhasználó befolyásolhatja a megvalósítandó logikát
- A logika befolyásolásához segédeszközökre van szükség:
    - fejlesztőrendszer
    - programozó készülékek,

- égető eszközök van szükség.

Gate Array áramkörök:

- A gyártó egy teljesen univerzális hálózatot (cellahálózat) alakít ki:
  - sok esetben mátrix elrendezésű
- A hálózat csomópontjaiban univerzális vezérlő logika található, amellyel kiválasztható, hogy az univerzális cellák milyen kapcsolatban legyenek egymással
  - A felhasználónak az építőelemek (cellák) közötti összeköttetéseket kell megterveznie.
- A gyártás gyors
  - Csak néhány maszkra (fémezési maszk) van szükség (átfutási idő lerövidül)

PLD áramkörök:

- Kis integráltságú áramkörök
- N db logikai függvény realizálható velük
- Félkész áramkörök
- Alacsony fejlesztési költség:
  - Az „üres” PLD –k meglehetősen drágák
- Viszonylag magas egy példányra jutó költség
- Egyetlen berendezésnél is kifizetődő lehet
- Sok fajtája másolás ellen védetté tehető (titkosítás)

22. tétel: A diszkrét, ASIC, PLÁ áramkörök összehasonlítása

ASIC:

- általában széleskörű használhatóság
- általában a felhasználó tervezi
- a sorozatszám széles határok között változhat
- drágábbak a katalógus IC-knél
- CPLD (Complex Programmable Logic Device)

Tipikus ASIC aramkorok (FPGA, CPLD):

- adott különleges célokra készülnek
- teljes egészében előre gyártottak
- a kívánt logikai feladatkört az összeköttetések programozásával valósíthatjuk meg, hasonlókeppen, mint a programozható ROM-ok esetében.
- logikai cellákból es programozható összeköttetések mátrixából állnak.
- konfigurálásuk vagy biztosítékok kiégetésével, vagy programozható összeköttetések programozásával történik → a programozható összeköttetések lehetnek: MOS áteresztő tranzisztorok vagy EPROM/EEPROM típusu tranzisztorok

Előny:

- gyorsan elkészíthető
- olcsó
- sokszor újraprogramozható

Hátrány:

- korlátozott bonyolultság
- korlátozott paraméter értékek (sebesség, stb.)

	CPLD	FPGA
Huzalozás	Limitált	Flexibilisebb
Logikai elemek	AND –OR hálózat	LUT (look up table)
I/O blokkok	Makrocellához rendelt	Konfigurálható
Komplexitás	Alacsony	Változó
Sebesség	150 MHz	250 MHz
Megvalósítható Kapuk száma	1 millió	10 ezer
alkalmazás	Összetett hálózatok: Sorrendi és kombinációs hálózatokat is használnak	Kombinációs hálózathoz, vezérlő logikákhoz

FPGA:

- minden blokk általános célú nincsenek be es kimeneti blokkok

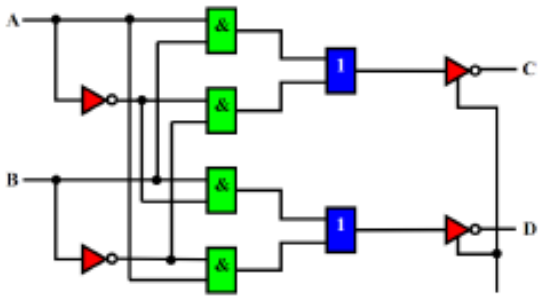
CPLD:

- Programozható IC-k fontos részei:
  - a kiegyenlített késleltetésű órajel meghajtók
  - és az órajel elosztó hálózat

23. tétel: Egyszerű programozható logikai áramkörök fajtái, belső felépítésük.

Logikai kapu alapú PLD -k felépítése

- Bemeneti buffer.
  - Bemeneti adatok fogadása
  - Negált változók előállítása (annyi invertert tartalmaz amennyi a bemenetek száma)
- ÉS kapcsolatok mátrixa:
  - A bemenetek ÉS kapukkal való összekötése
- ÉS kapuk
- VAGY kapcsolatok mátrixa:
  - Az ÉS és a VAGY kapuk közötti összeköttetések
- VAGY kapuk
- Kimeneti buffer



Logikai kapu alapú PLD -k jellemzői

- A kombinációs hálózat tulajdonságait az „ÉS kapcsolatok mátrixa” és a „VAGY kapcsolatok mátrixa határozza meg.
  - Ezek bármelyikének megváltoztatásával új kombinációs hálózatok hozhatók létre
- Alapesetei:
  - Az ÉS kapcsolatok mátrixa fix huzalozású, a VAGY kapcsolatok mátrixa programozható
  - Az ÉS kapcsolatok mátrixa programozható, a VAGY kapcsolatok mátrixa fix huzalozású
  - Mindkét mátrix programozható

Fajtái

ROM típusú áramkörök:

- Az ÉS mátrix fix, a VAGY mátrix programozható
- A hálózat annyi ÉS kaput tartalmaz, ahány bemeneti kombináció létezik
    - Minden mintermet egyedileg valósít meg
- A VAGY kapuknak annyi bementük van, amennyi az ÉS kapuk száma
    - A logikai függvény teljes diszjunktív alakja valósítható meg vele
- Adatok tárolására alkalmas
    - Bemenetek: címvezetékek
    - Kimenetek: adatkimenetek
- Az adatok csak egyszer írhatók bele (beégetés)
- Normál üzemben csak olvashatók

P AL típusú áramkörök

- Az ÉS mátrix programozható a VAGY mátrix fix
- Az elrendezés kötött:
    - A VAGY kapuk bemeneteinek a száma eleve meghatá-rozza, hogy hány mintermből
    - állhat egy-egy függvény
    - Adott hálózaton belül realizálható függvények számát VAGY kapuk száma határozza meg
- A beégetés alapjául az egyszerűsített függvényalakok szolgálnak
    - Meg lehet küzdeni a hazárdjelenségekkel

PLA áramkörök

- Mindkét mátrix (ÉS és a VAGY is!) programozható
    - Nagy flexibilítást biztosít
- Az ÉS mátrix programozhatósága lehetővé teszi az egyszerűsített függvényalakok használatát
- A VAGY mátrix programozhatósága megengedi, hogy egyes mintermek több függvény elemeként is alkalmazhatóak legyenek

24. tétel: Összetett PLÁ-k, programozási lehetőségek.

CPLD (**C**omplex **P**rogrammable **L**ogic **D**evice)

- Nagyobb, bonyolultabb feladatok megoldására való
- 4 típusa van:
    - Szimmetrikus felépítésű FPGA:
        - A gyártó mátrix felépítést alakít ki
        - Három részből áll:
            - Be- és kimeneti bufferek tömbje (be- és kimeneti blokkok)
            - Logikai blokkok:
                - Néhány egyszerűbb alapáramkört tartalmaz
                - Konfigurálható (CLB)
            - Programozható kapcsolatrendszer (vezetékezés)
- Kihasználtsági korlát: 60-80%
- „Elfelejti” a benne lévő információt:
        - A működés 2 fázisú:
            - Konfigurálás (beolvasás)
            - Normál logikaként működik

    - Csatornázott (channeled) tömbök – TEXAS:
        - Nincsenek külön be- kimeneti és logikai blokkok:
            - Minden blokk általános célú
            - Feladata programozástól függ
        - A szomszédos logikai blokkok között közvetlen kapcsolat van
        - A nem szomszédos blokkok között vezetékezés alakítható ki

    - Sea of Gates:
        - nincs vezetékezés kialakítva az egyes blokkok között

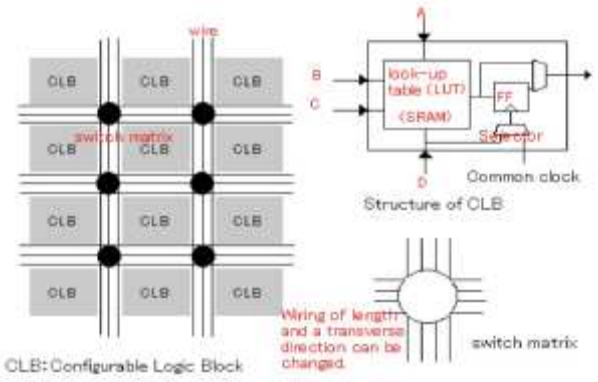
    - Hierarchikus PLD:
        - kevesebb, de nagyobb bonyolultságú blokkot tartalmaz (tárolókat is tartalmazhat)
        - a blokkokat buszrendszer kapcsolja össze

Szimmetrikus felépítésű FPGA

- A sorozatszám kapu ekvivalenciára utal:
  - Pl.: 4000-es sorozat esetén a realizálandó logika kb. ennyi kapu segítségével lenne
  - realizálható.
- Kihasználtsági korlát: a struktúra 60-80% -a:
  - Ez akkor érhető el, ha a logika és a struktúra egybeesik
  - Korlátozó tényezők:
    - Blokkok közötti összeköttetés: (vezetékezés), (10 mm van rá)
    - Tokozás: I/O funkció. (különféle tokozással készítik, ami meghatározza a lábszámot
- A megvalósított logika statikus RAM tartalomtól függ.
- IOB (I/O Bokkok):
  - Feladata: belső logika és a tényleges áramköri láb közötti kapcsolat megvalósítása
  - 5 bitnyi konfigurációs adattal állítható be
- CLB (Konfigurálható Logikai Blokkok):
  - Működési módok:
    - 1*5 bemenetű (32 soros) bemeneti függvény (F és G azonos)
    - 2*4 bemenetű (16 soros) függvény (F és G különböző)
  - A logikai függvényeket egy kombinációs függvény segítségével állíthatjuk be.
- A konfigurációs bitek mondják meg, hogy mely mintermek szerepelnek a kimeneti függvényben és melyek nem.
- Kapcsolatok (CLB –k között hozhatók létre):
  - Közvetlen szomszédok közötti:
    - Ez a legpraktikusabb (lehetőleg erre kell törekedni!)
    - X (vízszintes)-, Y(függőleges) irányú kapcsolatok kialakítására szolgál
  - Általános célú:
    - A kapcsolómátrixon belüli összeköttetések száma korlátozott:
      - Minden pont összeköthető a mellette levő kettővel,
      - A szemben levő kettővel,
      - És a mellette levő legtávolabbi szélsővel
  - Long lines:
    - Végigmennek a sorok és az oszlopok között

- Sorok között 3 db,
- Oszlopok között 2 db lehet

- Célszerű általános jelekre (Reset, Clock) használni



XILINKS XC series FPGA