# 2. Digitális hálózatok

A fejezetben a ***műszaki feladatok*** megoldására tervezett, és megvalósított ***digitális hálózatok***

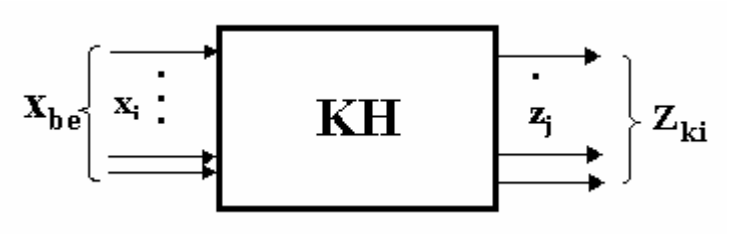
- alapvető tulajdonságait,
- a tervezés módszereit

tárgyaljuk.

Első lépésben megvizsgáljuk, hogy a logikai feladatok milyen ***rendszertechnikai*** megoldásokkal valósíthatók meg. Másodsorban megismerkedünk a ***logikai tervezés*** bevált módszereivel. Befejezésként áttekintjük azokat a ***funkcionális*** egységeket, amelyek felhasználásával egyszerűen építhetők össze a nagyobb digitális hálózatok.

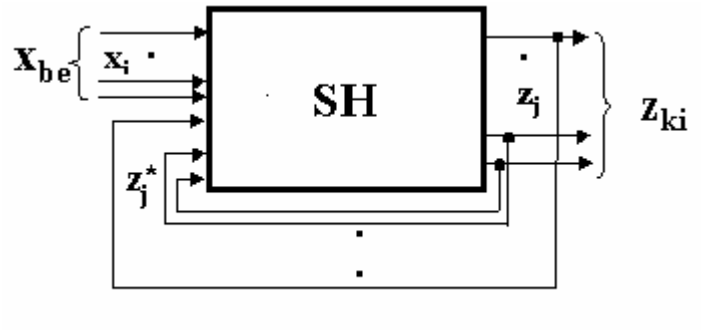
A műszaki feladatok megoldására alkalmazott digitális hálózatok ***alapvetően két nagy csoportba*** sorolható. A besorolást a be-, és a kimeneti digitális jelek közötti ***időbeli kapcsolat*** alapján tehetjük meg.

A feladatok egyik nagy csoportjánál a ***kimenet***(ek) – ***függő változó***(k) - logikai ***érték-kombinációját csak*** a ***bemeneti*** jel(ek) – ***független változó***(k) – vizsgált ***időpontbeli értékkombinációjától*** függ. Az ilyen feladatokat megvalósító digitális áramköröket ***kombinációs hálózatnak*** nevezzük. Blokkvázlata a 23.ábrán látható.




23. ábra

A másik csoportba az olyan az olyan logikai feladatok tartoznak, amelyeknél a ***kimenet***(ek) – függő változó(k) - logikai ***értékét*** a ***bemeneti*** jel(ek) – független változók – , és ***kimeneti*** jel(ek) – függő változó(k) - vizsgált ***időpontbeli értékkombinációja, együtt határozzák*** meg. Blokkvázlata a 24.ábra szerinti. A leírt tulajdonságú logikai feladatokat megvalósító digitális áramköröket nevezzük ***sorrendi hálózatoknak***. Még használják a ***szekvenciális***-, illetve ***emlékező***-hálózat elnevezéseket is.




24. ábra

## 2.1. Kombinációs hálózatok

A műszaki feladatok egy jelentős csoportjában a ***kimenetek jeleit csak*** a ***bemenetekre*** jutó ***jelek*** aktuális értéke ***szabja meg***. Az ilyen feladatok – a már definiált – kombinációs logikai hálózatokkal megvalósíthatók. Röviden áttekintjük a feladat ***logikai leírásának*** változatait. Utána megismerkedünk a feladatot megvalósító áramköri ***tervezés*** alapvető ***módszereivel***. Összefoglaljuk a leggyakrabban alkalmazott, un. ***funkcionális*** kombinációs egységek feladatait, működésüket.

⇒ **Kombinációs feladatok logikai leírása**

A kombinációs hálózatok meghatározásából következik, hogy a ***bemenetek*** (független változók), valamint a ***kimenetek*** jelei (függő változók) között ***egyértelmű logikai*** függvénykapcsolat van. Ez megadható az

$$\mathbf{Z_i = f_l\,(\,X_i\,)}$$

általános függvényleírással, amelyben $\mathbf{Z}_i$ a hálózat kimeneti kombinációja az **i**-ik időpillanatban, az $\mathbf{X_i}$ ugyanekkor érvényes bemeneti kombináció, és az $\mathbf{f_l}$ írja le a ***logikai*** függvénykapcsolatot. Logikai függvények ténylegesen nem az értékkombinációkra, hanem egy-egy valós kimenetre írhatók fel.

A tananyag első fejezetében – az elméleti alapokban – már részletesen tárgyaltuk a logikai függvények megadásának (leírásának) használt változatait. Itt most ismétlésként ismételjük meg azok összefoglalását.

A függvények megadása – leírása – történhet

- ***algebrai*** alakban**,**
- ***táblázat*** segítségével**,**
- ***matematikai*** jelölésekkel**,**
- ***grafikus*** módon**,**
- ***időfüggvény*** formájában**.**

A felsorolt leírási módok teljesen egyenértékűek, és egymásba átírhatók!

A kombinációs logikai feladatokat megvalósító hálózatok tervezésénél a megismert függvénymegadási formákat használjuk.

### ⇒ Kombinációs hálózatok logikai tervezése

Egy ***hálózat tervezése*** több részből áll. Először a függő változók – kimenetek – ***logikai függvényeit*** kell meghatározni. Ezt nevezzük ***logikai*** tervezésnek. Az eddigi munka eredményéből lehet az ***áramkört*** megtervezni. Mindezek után következhet a ***huzalozási***, illetve ***nyomtatási*** terv elkészítése. A következő szakasz már a ***gyártás***, és az ***ellenőrzés***.

A ***logikai tervezés*** bármelyik formája a feladat ***egyértelmű leírására*** épül. A ***gyakorlatban*** a megadás első változata a feladat ***szöveges leírása***. Miután – bármely nyelvben – a szöveges definíció félreértésekre is adhat okot, valamint a ***szisztematikus*** tervezést sem támogatja, ezért azt egy közbenső – egyértelműen értelmezhető – formára kell átfordítani.

A logikai tervezés a következő lépésekre bontható:

- a feladat ***változóinak*** egyértelmű meghatározása,
- a függvény ***igazságtáblázatának*** felírása,
- a függvények ***kanonikus*** alakjainak – valamelyik módszerrel (algebrai, indexelt, grafikus) történő felírása,
- az ***egyszerűsítések*** – minimalizálás – végrehajtása,
- ***logikai vázlat*** megrajzolása.

A legegyszerűbb, egyszerűsített logikai függvény meghatározása történhet:

- ***Algebrai*** úton
- A ***kanonikus*** (diszjunkt, vagy konjunkt) ***alakokból.***
- ***Grafikus*** módszerekkel
- ***Karnaugh táblázatok*** (minterm Kp, maxterm Ks) felhasználásával**.**
- ***Numerikus*** módszerrel (Quin,- Mc Closkey eljárás)

A példa segítségével tekintsük át a kombinációs hálózat tervezésének menetét.

***10.Példa***

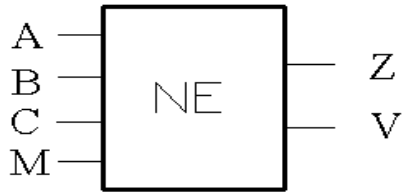
Feladat olyan hálózat tervezése, amelynek ***négy bemenete***, és ***két kimenete*** van.

A bemenetek közül három ***adat-***, egy pedig ***parancs*** jelet fogad. Az ***egyik kimenetén*** akkor kapunk logikai 1 értéket, ha a ***három*** adatbemenete közül – a parancs értékétől függetlenül - ***kettőn*** van logikai ***1*** érték. A másik kimeneten 1 szintű legyen, ha a ***parancs 0*** és az adatbemeneteken ***több*** az ***1*** érték, mint a ***0***, amikor, pedig a ***parancs 1*** értékű a ***több 0*** értéknél legyen ***1*** a kimenet.

Jelöljük az ***adatbemeneteket A***, ***B***, ***C***, míg a ***parancsot M*** betűkkel.

A ***kimeneteket,*** pedig jelöljék a ***Z*** (a parancstól független), és ***V*** betűk.

A 25. ábrán rajzoltuk meg megvalósítandó hálózat elvi ***blokkvázlatát***




25. ábra

**a.** A hálózat ***igazságtáblázatát*** láthatjuk a 26.ábrán

| ***M*** | ***C*** | ***B*** | ***A*** | ***Z*** | ***V*** |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

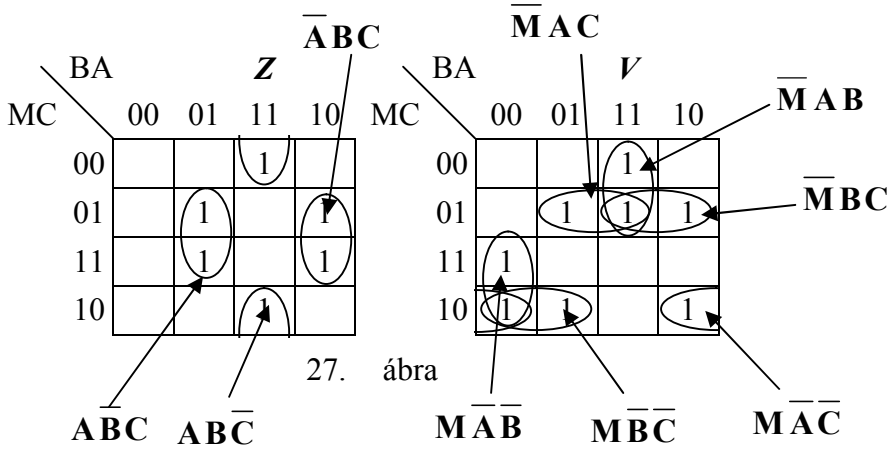
26. ábra

**b.** a kimeneti ***diszjunktív kanonikus*** függvények algebrai alakjai, és az összevonások.

$$
\begin{aligned}
Z &= A\cdot B\cdot \overline{C}\cdot \overline{M} + A\cdot \overline{B}\cdot C\cdot \overline{M} + \overline{A}\cdot B\cdot C\cdot \overline{M} + \\
&+ A\cdot B\cdot \overline{C}\cdot M + A\cdot \overline{B}\cdot C\cdot M + \overline{A}\cdot B\cdot C\cdot M = \\
&= (A\cdot B\cdot \overline{C} + A\cdot \overline{B}\cdot C + \overline{A}\cdot B\cdot C)\cdot \overline{M} + \\
&+ (A\cdot B\cdot \overline{C} + A\cdot \overline{B}\cdot C + \overline{A}\cdot B\cdot C)\cdot M = \\
&= A\cdot B\cdot \overline{C} + A\cdot \overline{B}\cdot C + \overline{A}\cdot B\cdot C = A(B\cdot \overline{C} + \overline{B}\cdot C) + \overline{A}\cdot B\cdot C
\end{aligned}
$$

$$V = A\cdot B\cdot\overline{C}\cdot\overline{M} + A\cdot\overline{B}\cdot C\cdot\overline{M} + \overline{A}\cdot B\cdot C\cdot\overline{M} + A\cdot B\cdot C\cdot\overline{M} +$$
$$+\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot M + A\cdot\overline{B}\cdot\overline{C}\cdot M + \overline{A}\cdot B\cdot\overline{C}\cdot M + \overline{A}\cdot\overline{B}\cdot C\cdot M =$$
$$= A\cdot B\cdot\overline{M} + A\cdot C\cdot\overline{M} + B\cdot C\cdot\overline{M} + \overline{A}\cdot\overline{B}\cdot M + \overline{A}\cdot\overline{C}\cdot M + \overline{B}\cdot\overline{C}\cdot M =$$
$$= \overline{M}\cdot(A\cdot B + A\cdot C + B\cdot C) + M\cdot(\overline{A}\cdot\overline{B} + \overline{A}\cdot\overline{C} + \overline{B}\cdot\overline{C})$$

**c.** ***Egyszerűsítés*** Karnaugh ( Kp) diagram segítségével (27.ábra):




27. ábra

Az összevonások után kapott függvények

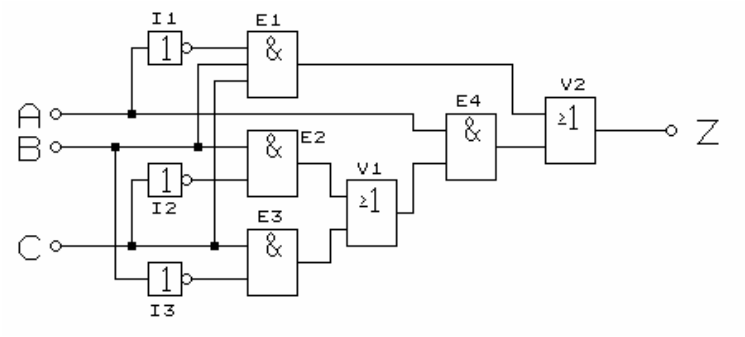
$$Z = A\cdot B\cdot\overline{C} + A\cdot\overline{B}\cdot C + \overline{A}\cdot B\cdot C = A(B\cdot\overline{C} + \overline{B}\cdot C) + \overline{A}\cdot B\cdot C$$

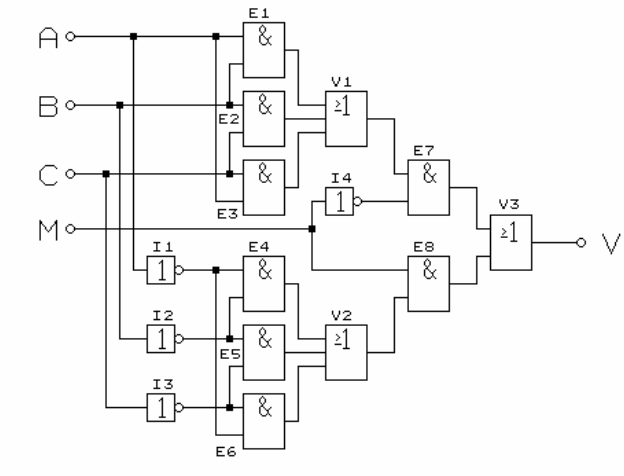
$$V = A\cdot B\cdot\overline{M} + A\cdot C\cdot\overline{M} + B\cdot C\cdot\overline{M} + \overline{A}\cdot\overline{B}\cdot M + \overline{A}\cdot\overline{C}\cdot M + \overline{B}\cdot\overline{C}\cdot M =$$
$$= \overline{M}\cdot(A\cdot B + A\cdot C + B\cdot C) + M\cdot(\overline{A}\cdot\overline{B} + \overline{A}\cdot\overline{C} + \overline{B}\cdot\overline{C})$$

A függvények megegyeznek az algebrai egyszerűsítés eredményeivel.

**d.** A logikai vázlat megrajzolása

Az ***egyszerűsítések*** alapján kapott függvények alapján, a logikai kapuk ***szimbólumaival*** rajzolható meg - a megvalósítandó - kombinációs hálózat, 28. ábra szerinti ***logikai vázlata***.




28. ábra

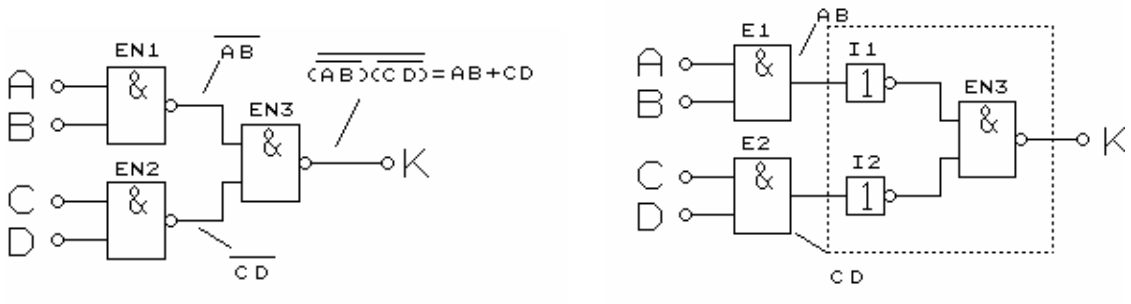
## 2.2. Összetett műveletek használata

Az eddigiekben az ***ÉS***, ***VAGY***, valamint a ***TAGADÁS*** műveletét alkalmaztuk. Továbbiakban foglalkozunk a ***NAND***, a ***NOR***, az ***XOR*** (kizáró vagy), valamint az ***XORN*** (az XOR tagadottja, equivalencia) műveletek alkalmazásával a kombinációs hálózatok megvalósításánál. Az előző kettőt nevezik ***univerzális műveletnek*** is, és elsősorban az általános kombinációs hálózatok megvalósításához használjuk. Az utóbbi két műveletet ***moduló - műveleteknek*** is nevezik, és elsődlegesen az aritmetikai feladatok (összeadás – kivonás, összehasonlítás) végrehajtásához alkalmazzák.

### ⇒ Az univerzális műveletek alkalmazása

- ***NAND kapuk alkalmazása***

A 29.a.ábrán látható hálózat csak NAND kapukból áll. Írjuk fel az egyes kapuk kimenetein érvényes logikai függvényeket, és végül a teljes hálózat K kimenetének függvényét.




a. b.

29. ábra

A kapuk kimenetein felírható függvények alapján tehát kimeneti jel értéke a $K = A\,B + C\,D$ logikai függvény szerint függ a bemenetek logikai értékeitől.

A 29.b.ábrán az EN1, és az EN2 jelű NAND kapukat szétválasztottuk ÉS kapukra, (E1, E2)valamint inverterekre (I1, I2). A ***szaggatott*** vonallal körülhatároltuk részlet ***VAGY*** műveletet valósít meg, mivel

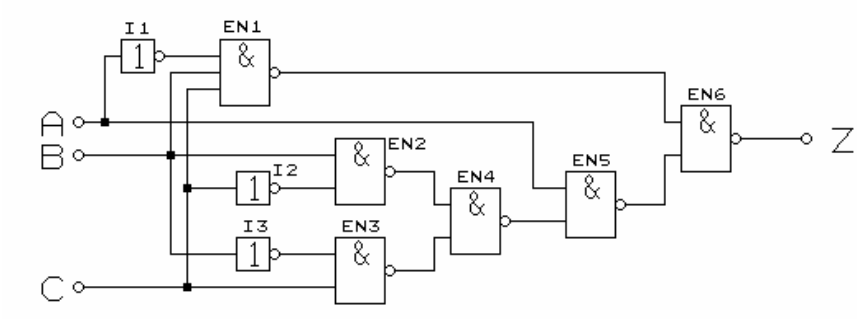
$$\overline{\overline{(A\,B)}\,\overline{(C\,D)}} = A\,B + C\,D$$

A példa azt szemlélteti, hogy NAND kapukkal megépített ***két-szintű*** kombinációs hálózat ***kimenet felőli*** szintje ***VAGY*** műveletet, azt ***megelőző szint***, pedig ***ÉS*** műveletet hajt végre. Az előző megállapítás több szintű hálózatra is kiterjeszthető, ha azt vesszük, hogy a szintek párosával mindig csoportosíthatók. A többszintű hálózatot a kimenet felől osztjuk ***páratlan – páros*** szintekre. Általánosan tehát igaz a következő:

- ***a NAND kapu páratlan szinten VAGY, míg páros szinten ÉS műveletet valósít meg,***
- ***a páros szinten bevezetett jel változatlan értékkel, míg a páratlan szinten bevezetett jel az eredeti tagadottjaként szerepel a kimenet függvényében.***

A megismert törvényszerűségek alapján, a 30. ábrán az előző példa Z kimenetét létrehozó hálózat NAND kapuk alkalmazásával megrajzolt logikai vázlata látható.

$$Z= A\cdot B\cdot\overline{C}+A\cdot\overline{B}\cdot C+\overline{A}\cdot B\cdot C=[A(B\cdot\overline{C}+\overline{B}\cdot C)]+[\overline{A}\cdot B\cdot C]$$

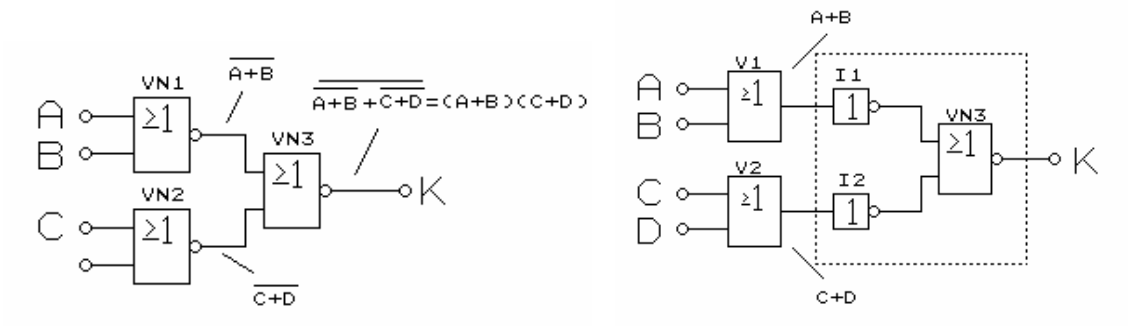



30. ábra

A függvény algebrai alakját – a szögletes zárójelek segítségével – csoportosítottuk. A Z kimenet értékét tehát két mennyiség VAGY kapcsolata adja, amit az EN6 jelű NAND kapu valósít meg. Az EN1 jelű kapu a jobboldali, míg az EN5 jelű kapu a baloldali szögletes zárójeles ÉS műveletet állítja elő. Az EN4 kapu ismét VAGY (páratlan szint), míg az EN2, EN3 jelű kapuk, pedig ÉS műveletet hoznak létre. A B, és C változók. És tagadottjaik is páros szinten jutnak a rendszerbe, ezért a függvényben is így szerepelnek. Az A változó két különböző műveletben is szerepel, és páros szinten vezetjük be. Mivel az egyik műveletben negált alakban kell, szerepeljen, ezért kellett az I1 jelű inverter.

- ***NOR kapuk alkalmazása***

A 31.a.ábrán látható hálózat csak NOR kapukból áll. Láthatók az egyes kapuk kimenetein érvényes logikai függvények, és végül a teljes hálózat K kimenetének függvénye.




a. b.

31. ábra

A kapuk kimenetein felírható függvények alapján tehát kimeneti jel értéke a $K=(A+B)(C+D)$ logikai függvény szerint függ a bemenetek logikai értékeitől.

A 30.b.ábrán az VN1, és az VN2 jelű NOR kapukat szétválasztottuk VAGY kapukra, (V1, V2)valamint inverterekre (I1, I2) . A szaggatott vonallal körülhatárolt részlet ÉS műveletet valósít meg, mivel

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

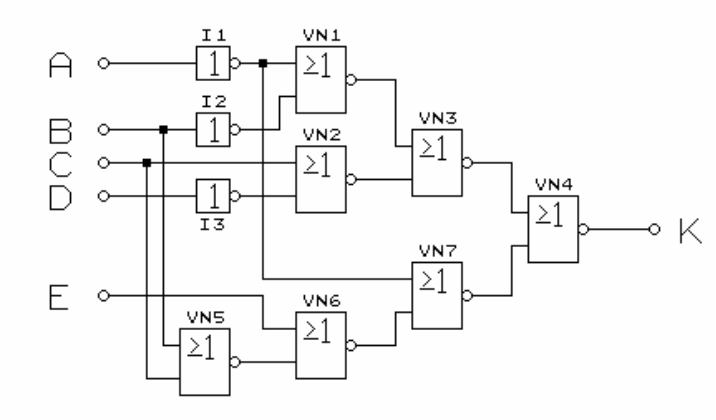
A példa azt szemlélteti, hogy a NOR kapukkal megépített ***két-szintű*** kombinációs hálózat ***kimenet felőli*** szintje ***ÉS*** műveletet, azt ***megelőző szint***, pedig ***VAGY*** műveletet hajt végre. A NAND kapus hálózatokhoz hasonlóan, törvényszerűség több szintre is megállapítható. Általánosan tehát igaz a következő:

- ***a NOR kapu páratlan szinten ÉS, míg páros szinten VAGY műveletet valósít meg,***
- ***a páros szinten bevezetett jel változatlan értékkel, míg a páratlan szinten bevezetett jel az eredeti tagadottjaként szerepel a kimenet függvényében.***

A leírtak alapján rajzoljuk meg a

$$K=(A\,B+\overline{C}\,D)(\overline{A}+\overline{E}(B+C))$$

függvény NOR kapukkal történő megvalósításának logikai vázlatát. A kimenet (K) a ***két*** – szögletes zárójelbe tett - ***mennyiség ÉS*** kapcsolata, amit a ***páratlan*** szinten álló két-bemenetű - VN4 jelű - ***NOR*** valósít meg. A kapu bemeneteihez csatlakozó – VN2, és VN7 jelű kapuk – ***páros szinten*** vannak, és ezért ***VAGY*** műveletet realizálnak sit.. A további kapuk szintje, s az általuk megvalósított logikai műveletek 32. ábrán követhetők végig.




32. ábra

***Összefoglalva***: megállapíthatjuk, hogy bármely kombinációs feladat megvalósítható csak NAND, vagy csak NOR kapuk alkalmazásával. Vegyesen nagyon ritkán használják a különböző ***univerzális*** kapukat. (Az integrált áramköri digitális áramkörök tárgyalásakor, a 3. fejezetben megismerjük azt is, hogy az univerzális kapukkal inverterek is létre hozhatók.)

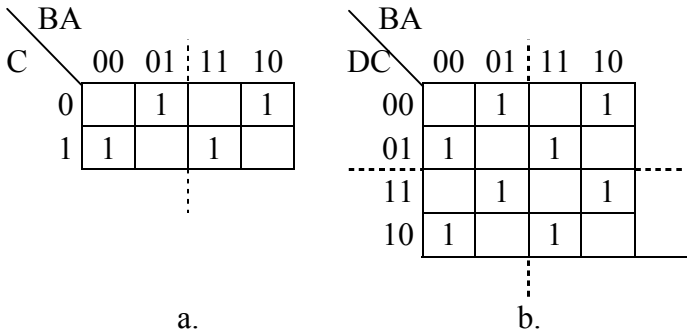
### ⇒ A kizáró-vagy kapuk alkalmazása

Az 1. fejezetben megismertük a ***KIZÁRÓ-VAGY*** (XOR) logikai műveletet, amelyet moduló-összegzésnek is neveznek. Minden olyan alkalmazásban, amelyben a függő változó csak akkor IGAZ, ha a független változók érték-variációban csak páratlan, vagy csak páros számú az IGAZ, akkor használhatók az XOR, vagy az NXOR műveletek. (Mivel a technikai, műszaki feladatokban gyakori a hasonló feltétel, ezért az integrált áramköri kapuk között ezek megtalálhatók.)

A függvények algebrai, illetve Karnaugh diagramon történő egyszerűsítésekor felismerhető az XOR kapu alkalmazhatósága. A ***kétváltozós*** moduló-összeg algebrai alakja: $\mathbf{A}\,\overline{\mathbf{B}} + \overline{\mathbf{A}}\,\mathbf{B}$, Kp diagramja pedig a 33. ábrán látható. A 34.a. és b. ábrákon a ***három***-, illetve ***négyváltozós*** XOR művelet Kp diagramja látható.

| B \ A | 0 | 1 |
|---|---|---|
| 0 | | 1 |
| 1 | 1 | |

33. ábra

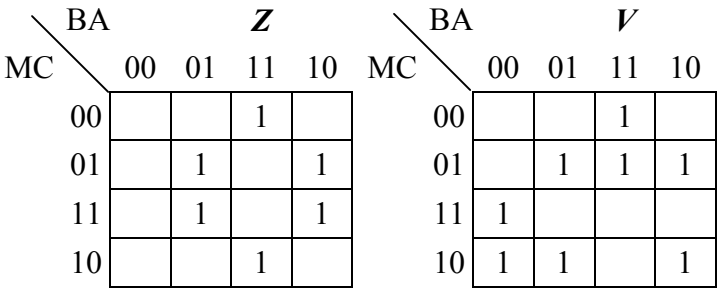



a. b.

34. ábra

A 34. ábra mindkét táblázatában megfigyelhetjük, hogy a ***szaggatott*** vonalak mentén „***összehajtva***” a táblát, akkor az 1-ek a másik rész 0 értékére esnek. A megfigyelés lehetővé teszi az XOR művelet lehetséges használatát.

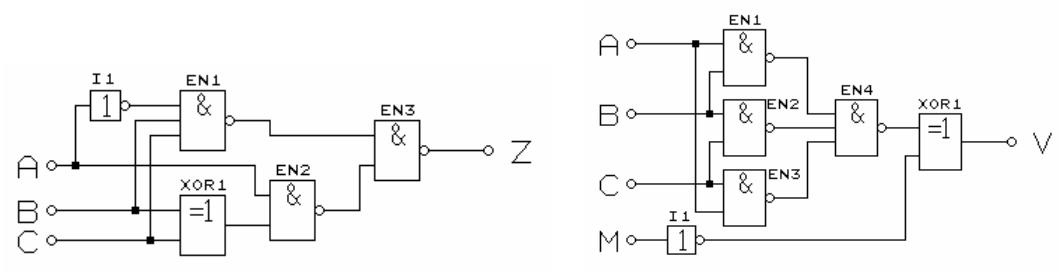
A fejezetben megoldott példa Z, és V kimeneti változók Karnaugh diagramjai láthatók a 35.ábrán.




a. b.

35. ábra

A 36.ábrán az XOR kapuk alkalmazásával kialakított logikai vázlatok láthatók.




a. b.

36. ábra

A kétbemenetű XOR kapu vezérelt inverter -nek is használható

## 2.3. Funkcionális kombinációs feladatok

A logikai kapukka1 elvileg minden logikai feladat megvalósítható. Ugyanakkor a legkülönbözőbb rendeltetésű hálózatokban megtalálhatók olyan nagyobb ***funkciókat*** ellátó egységek is, amelyek egyedi tervezéssel, kapukból is megépíthetők. Mivel gyakran használják digitális hálózatok elemeiként, ezért ***önálló áramkörként*** gyártják. Ezeket általában **r e n d s z e r t e c h n i k a i** (funkcionális) áramköröknek nevezzük.

Ilyen funkcionális egységek a következők:

- ***kódolók, dekódolók;***
- ***adatelosztók;***
- ***adatkiválasztók;***
- ***aritmetikai műveletvégzők.***

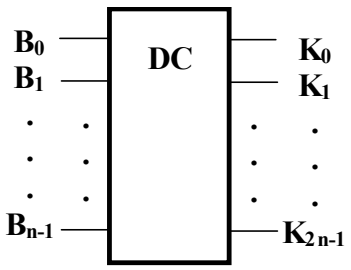
E fejezetben csak ezen funkcionális egységek logikai felépítésével, és működésével foglalkozunk.

### ⇒ D e k ó d o l ó k

A dekódoló egy ***kódátalakító*** kombinációs feladatot valósít meg. Az egység ***bemenetire*** adott ***bináris*** vagy ***BCD*** kódból az ***n*** db ***kimeneten*** un. **1 az n - ből** kódot állit elő. Ez a kód **n** bitből áll, és ezek közül mindig csak ***egy lehet aktív*** logikai értékű. Azt pedig, hogy melyik kimeneten lesz ***aktív*** érték, azt a bemeneteken lévő ***aktuális*** kód határozza meg. Ha az aktív logikai érték 1, akkor a többi bit 0, illetve fordítva.

Az ismertetett logikai hálózat működése lényegében ***kiválaszt egy kimenetet***, ugyanis a bemenetire adott kód alapján ***egyetlen*** kimenetet tesz ***aktívvá***. Joggal vetődik fel a kérdés, hogy miért is nevezzük dekoder -nek? A számítástechnika „hőskorában” a decimális számjegyek kijelzésére használt ***Nixie*** csövek (gáztöltésű kijelző csövek) ***megfelelő*** katódját kellet kiválasztani – 0 feszültséggel – ahhoz, hogy a kijelezni kívánt ***számjegy*** látsszon. Innen ered, hogy a bináris, vagy BCD kódolású számot „***dekódolta***” decimális formájú karakterre.

A dekódoló elvi blokkvázlatát a 37.ábra szemlélteti. A $\mathbf{B_0 \ldots B_{p-1}}$ jelű bemenetekhez csatlakozik az **p** bites átalakítandó kód (BCD kódnál p = 4). A $\mathbf{K_0 \ldots K_{2p-1}}$ jelű kimeneteken kapjuk az **1 az n-ből** kódot. (**p** bites ***bináris*** kódnál, a kimeneti ***bit*** -ek száma (n) maximálisan $\mathbf{2^p}$ lehet.).



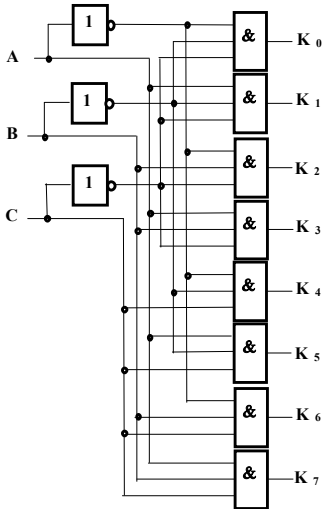

37. ábra

**⇒ Bináris dekódoló**

A 3 bites bináris kód (3-ról 8-ra) dekódolását végző kombinációs hálózat igazságtáblázata - logikai 1 szintű aktív kimenetet választva - a.38.a.ábrán a logikai vázlata, pedig a b ábrán látható.

| C | B | A | $K_0$ | $K_1$ | $K_2$ | $K_3$ | $K_4$ | $K_5$ | $K_6$ | $K_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

a.




b.

38. a

A bináris kódban általánosan az A,B,C,D betűkkel jelöljük az egyes helyi értékeket, az $\mathbf{A \div 2^0, B \div 2^1, C \div 2^2, D \div 2^3, ...}$ stb. súlyozás választásával. Az igazságtáblázatból felírhatjuk a következő logikai függvényeket:

$$\mathbf{K_0 = \overline{A}\overline{B}\overline{C}} \qquad \mathbf{K_1 = A\overline{B}\overline{C}}$$

$$\mathbf{K_2 = \overline{A}B\overline{C}} \qquad \mathbf{K_3 = AB\overline{C}}$$

$$\mathbf{K_4 = \overline{A}\overline{B}C} \qquad \mathbf{K_5 = A\overline{B}C}$$

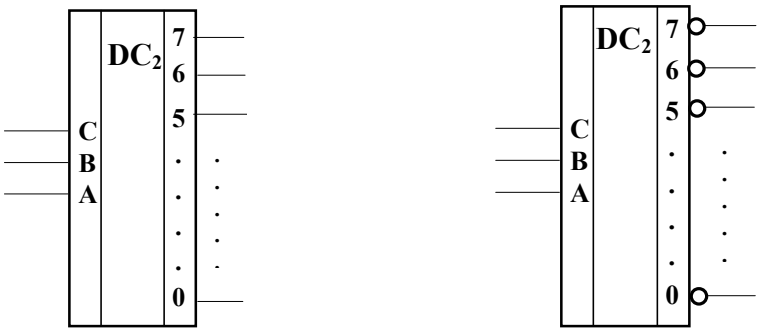
$$\mathbf{K_6 = \overline{A}BC} \qquad \mathbf{K_7 = ABC}$$

Az áramkör a mintermeket megvalósító **ÉS** kapukból és a változók tagadott értékeit előállító **inverter** -ekből áll.

A 0 értékű aktív kimeneti logikai szintnél az előző összefüggések tagadásával kapjuk a logikai egyenleteket. Az áramkör, pedig **NAND** kapukkal épül fel.

A dekódolandó bináris kód bitjeinek növelésével - az előbbiekben elemzett mindkét változatnál - a felhasznált kapuk és azok bemeneteinek száma növekszik.

A dekódolók szimbolikus jelölése látható a 39. ábrán. A 0 -val aktív kimenetet a karika ( tagadás) jelzi a b. ábrán.




a. b.

39. ábra

**⇒ BCD dekódoló**

A digitális áramköri készletek többségében van BCD decimális dekódoló is. A négy bites **BCD** kód (8 4 2 1 súlyozású), és a tíz decimális számértéket a bináris kód első tíz (K0 . . . K9) kombinációjához rendeli.

A BCD dekódoló áramköri kialakításánál egyszerűsítésre felhasználhatók a kódban elő nem forduló (K10 . . . K15) kombinációk is. A legegyszerűbb felépítésű BCD dekóder logikai függvényei a következők:

$$\mathbf{K_0 = \overline{A}\overline{B}\overline{C}\overline{D}} \quad \mathbf{K_1 = A\overline{B}\overline{C}\overline{D}}$$

$$\mathbf{K_2 = \overline{A}B\overline{C}} \quad \mathbf{K_3 = AB\overline{C}}$$

$$\mathbf{K_4 = \overline{A}\overline{B}C} \quad \mathbf{K_5 = A\overline{B}C}$$

$$\mathbf{K_6 = \overline{A}BC} \quad \mathbf{K_7 = ABC}$$

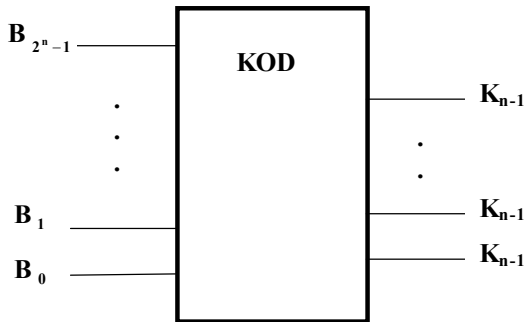
$$\mathbf{K_6 = \overline{A}D} \quad \mathbf{K_9 = AD}$$

Az ilyen megoldás ***nem teljesen dekódolt***, mivel a bemenetre adott tiltott kombinációk is aktiválhatnak kimenetet, (esetleg kimeneteket). Pl.: a DCBA kombináció hatására a K7 kimenet lesz akti - 1 szintű.

## ⇒ **Kódoló áramkörök**

A kódolás során **1 az N** - ből kódot (pl. 1 a 10-ből a decimális kód) kívánunk átalakítani **bináris, BCD** vagy **egyéb** kóddá. Ezt a feladatot megvalósító kombinációs hálózat a **kódoló**.

A kódolás tulajdonképpen a dekódolás duálja. A kódoló blokksémája a 40. ábrán látható. A maximálisan $\mathbf{2^n}$ számú bemenet ($\mathbf{B_0 \cdots B_{2^n-1}}$) egyikére jut csak aktív logikai szint (1 vagy 0), s ennek alapján állítja elő az **n** db kimeneten ($\mathbf{K_0 \cdots K_{n-1}}$) a megfelelő **n** bites bináris kódot.




40. ábra

Vizsgáljuk meg az egyik leggyakrabban használt kódoló áramkör, a decimális - BCD átalakító logikai függvényét és megvalósításának lehetőségeit. A 41.ábrán látható a kódolási feladat igazságtáblázata. A kimenetek jelölésére a szabványos A,B,C,D betűket használtuk.

A táblázat alapján felírhatók az egyes kimeneteket megvalósító logikai függvények.

$$\mathbf{A = B_1 + B_3 + B_5 + B_7 + B_9}$$
$$\mathbf{B = B_2 + B_3 + B_6 + B_7}$$
$$\mathbf{C = B_4 + B_5 + B_6 + B_7}$$
$$\mathbf{D = B_8 + B_9}$$

| $B_9$ | $B_8$ | $B_7$ | $B_6$ | $B_5$ | $B_4$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ | D | C | B | A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

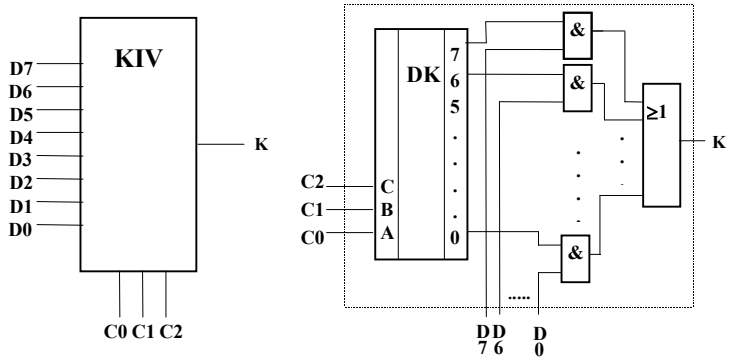
41. ábra

A logikai függvények ismeretében megtervezhető a kódolást megvalósító kombinációs hálózat.

A kódolók kitüntetett alkalmazási területe az adatbevitelre szolgáló billentyűzet (klaviatúra) és a digitális berendezés illesztése. Viszonylag egyedi felhasználásúk miatt integrált áramköri kialakításban nem készítenek ilyen kódolót. Diszkrét elemekből. IC kapukból könnyen megépíthetőek.

## ⇒ Kiválasztó áramkörök (multiplexerek)

A **kiválasztó** áramkör (adatszelektor) az **adat**-bemenetek ($\mathbf{D_1}$ . . $\mathbf{D_p}$) egyikének információját kapcsolja a **Q** kimenetre. A kiválasztást az **n** darab**címző** (kiválasztó) bemeneten ($\mathbf{C_0}$ ... $\mathbf{C_{n-1}}$) érvényes bináris kód határozza meg. (Az **n** bittel címezhető adatbemenet maximális száma $\mathbf{2^n}$ .) Elvi blokkvázlata a 42.a.ábra szerinti.




a. b.

42. ábra

Egy n = 3 címző bemenetű multiplexer 8 adatból ($2^3$) választ ki egyet. Ennek logikai függvénye a következő:

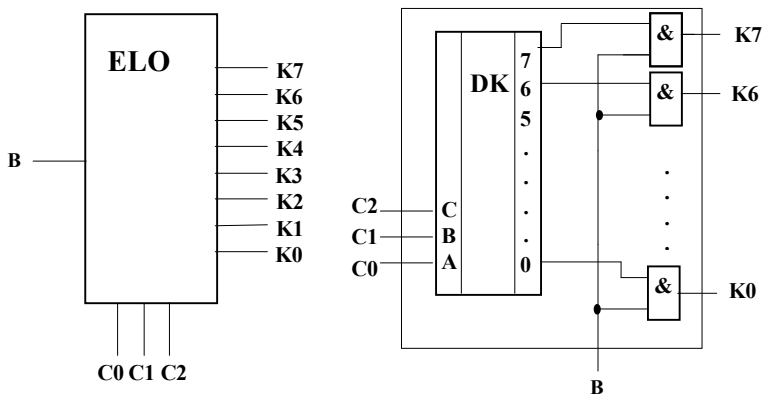
A függvény minden egyes logikai szorzatában szerepel valamelyik adat ($D_0$ ... $D_7$) és a címző bitek ($C_0$,$C_1$,$C_2$) egyik kombinációja, mintermek (a zárójelbe tett mennyiségek), amelyek kimenetei közül egyidejűleg csak egyik lehet logikai 1 értékű. Ezért a **Q** kimenetre az **adat-bit** ( $D_i$) jut, amelyhez tartozó címző variáció értéke1.

A függvénykapcsolat megvalósítható a címző $C_0$,$C_1$,$C_2$ kódot **dekódoló** áramkörből és egy **ÉS-VAGY** hálózatból. Ennek logikai vázlat t mutatja a 42.b. ábra.

### ⇒ Elosztó áramkör (demultiplexer)

Az adatelosztásra alkalmazható demultiplexer egyetlen adatbemenetről osztja szét az információt $2^n$ számú kimenetre, ahol **n** a címző (elosztó) bemenetek száma.

Az áramkör elvi blokkvázlata a 43.a.ábrán látható.




a. b.

43. ábra

A függvényeket megvalósító hálózat felépíthető a címző bemeneteket dekódoló áramkörből, és ennek kimeneteit a D adattal kell kapuzni. A megvalósítás logikai vázlata látható a 43.b.ábrán.

Az elosztási feladat logikai függvényei n=3 esetén a következők:

$$K_0 = D(\overline{C}_2\overline{C}_1\overline{C}_0) \qquad K_4 = D(C_2\overline{C}_1\overline{C}_0)$$
$$K_1 = D(\overline{C}_2\overline{C}_1C_0) \qquad K_5 = D(C_2\overline{C}_1C_0)$$
$$K_2 = D(\overline{C}_2C_1\overline{C}_0) \qquad K_6 = D(C_2C_1\overline{C}_0)$$
$$K_3 = D(\overline{C}_2C_1C_0) \qquad K_7 = D(C_2C_1C_0)$$

(A zárójelekbe tett kifejezések a dekódoló kimeneteinek a függvényei).

A demultiplexer kapuzott dekódolóként is alkalmazható, mivel a D bemenet 0 értékénél – a címző bemenetek vezérlésétől függetlenül – mindegyik kimenet 0 szintű lesz.

## ⇒ Aritmetikai műveletek megvalósítása

A digitális számítógépekben, műszerekbe, vezérlő egységekben stb. végzendő ***számítási*** műveletek ***bináris számrendszerben*** történik. A kettes számrendszer ***alapműveletei***, az ***összeadás***, ***kivonás***, ***összehasonlítás*** logikai műveletekkel elvégezhetőek. Az ***aritmetikai*** műveletvégző egységek ***kombinációs*** logikai ***hálózatokkal*** megvalósíthatóak. Itt ismertetjük a ***teljes összeadó-kivonó*** (TAK), és a két-bites ***nagyság-komparátor*** logikai felépítését, működését.

### ▪ Egy helyértékű összeadó-kivonó egység

Az összeadásnál, és a kivonásnál - bármelyik számrendszerben – helyértékenként kell a műveletet elvégezni. A ***teljes összeadást***, vagy a ***kivonást*** a tényezők ***adott helyértékű*** két számjegye és az ***előző helyértéken*** keletkezett átvitel, illetve áthozat értékével, kell elvégezni. (teljes jelző utal arra, hogy az előző helyértéken keletkező túlcsordulással – átvitellel, áthozat -al - is végzünk műveletet).Az ***összeadás*** eredményei az ***összeg*** (**S** - summa) és az ***átvitel*** (**C** - carry), míg a ***kivonásnál*** a ***különbség*** (**D** - different) és az ***áthozat*** (**B** - borrow).

A ***bináris számrendszerben*** mind a tényezők helyértéke, mind pedig a műveletek eredménye ***0***, vagy ***1*** lehet. Írjuk fel mindkét művelet értéktáblázatát (44. ábra). Az

egyes tényezőket – mindkét műveletnél – jelöljük **X**, illetve **Y**, míg az előző helyérték átvitelét $\mathbf{C_{-1}}$, áthozatás $\mathbf{B_{-1}}$ betűvel.

***Összeadás*** (S=X+Y)

| X | Y | $C_{-1}$ | S | C |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

***Kivonás*** (D=X-Y)

| X | Y | $B_{-1}$ | D | B |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

44. ábra

Az értéktáblázatok megegyeznek egy kombinációs hálózat igazságtáblázatával, amiből következik, hogy ezeket az ***aritmetikai*** műveleteket ***kombinációs hálózattal*** meg is lehet valósítani. A két táblázatot összehasonlítva azt látjuk, hogy a két műveletnél az összeg, illetve a különbség azonos értékeket ad, és csak az átvitel, illetve áthozat különbözik. A táblázatok megvalósítása adja a ***teljes összeadó*** (**TA**), illetve ***teljes kivonó*** (**TK**) áramköröket. Ezek logikai vázlatát most nem rajzoljuk meg.

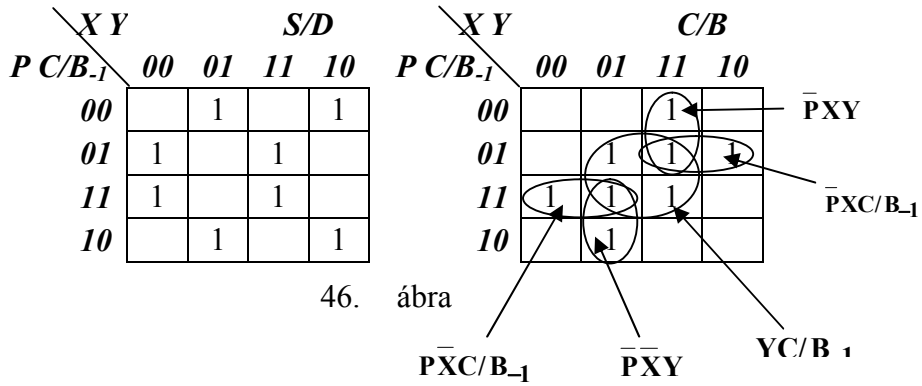
A két művelet egyetlen hálózattal is megvalósítható, amelynek – a műveleti tényezőkön kívül – van egy parancs bemenete (**P**) is. Az ilyen hálózatot nevezzük ***teljes összeadó-kivonó*** (**TAK**) áramkörnek. Az áramkör igazságtáblázata látható 45.ábrán.

| | P | X | Y | $C/B_{-1}$ | S/D | C/B |
|---|---|---|---|---|---|---|
| **összeadás** | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 1 | 1 | 0 |
| | 0 | 0 | 1 | 0 | 1 | 0 |
| | 0 | 0 | 1 | 1 | 0 | 1 |
| | 0 | 1 | 0 | 0 | 1 | 0 |
| | 0 | 1 | 0 | 1 | 0 | 1 |
| | 0 | 1 | 1 | 0 | 0 | 1 |
| | 0 | 1 | 1 | 1 | 1 | 1 |
| **kivonás** | 1 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 1 | 1 | 1 |
| | 1 | 0 | 1 | 0 | 1 | 1 |

| vonás | | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 0 | 1 | 1 | 0 | 1 |
| | 1 | 1 | 0 | 0 | 1 | 0 |
| | 1 | 1 | 0 | 1 | 0 | 1 |
| | 1 | 1 | 1 | 0 | 0 | 0 |
| | 1 | 1 | 1 | 1 | 1 | 1 |

45. ábra

Végezzük el a két kimenetre – S/D, illetve C/B – a lehetséges egyszerűsítést Kp diagram használatával (46. ábra).



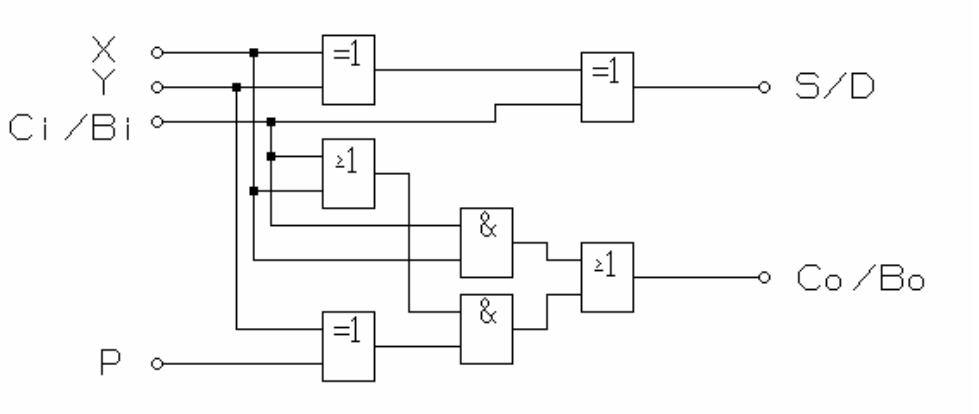

46. ábra

Az ***S/D*** (összeg - különbség) kimenetre felírt K-táblázatból is eldönthető, hogy az a P ***parancstól független*** és a három aritmetikai változó ***moduló összege*** adja az eredményt. A kimenet logikai függvénye:

$$S/D = X \oplus Y \oplus C/B_{-1}$$

A ***C/B*** (átvitel – áthozat) kimenet logikai függvénye a ***Kp diagramból*** felírva az alábbi:

A teljes összeadó-kivonó áramkör logikai vázlatát mutatja 47. ábra.



47. ábra

- ***Két bites nagyság-komparátor***

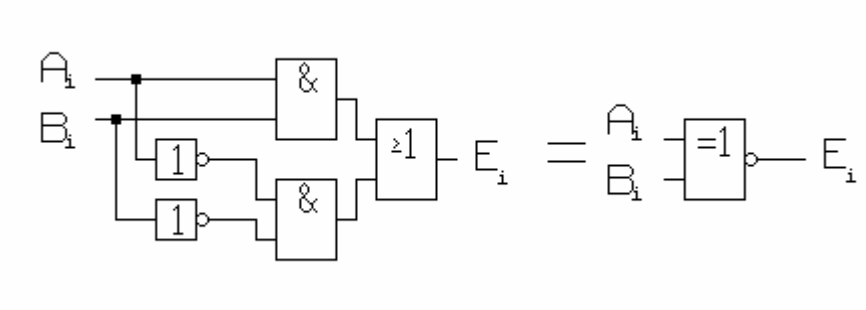
***Nagyság-komparátornak*** nevezzük azt az áramkört, amely két bináris számot hasonlít össze, és kimenetein jelzi a számok közötti ***relációkat*** (***egyenlő***, ***kisebb***, ***nagyobb***).

Az összehasonlítás az összetartozó - azonos nagyságrend – bit-párok relációjának megállapításán alapul.

Két bit (A és B) ***egyenlőségét*** az

$$\mathbf{E_i = A_i B_i + \overline{A}_i \overline{B}_i}$$

(equivalencia) írja le. Az áramkör logikai vázlata a 48. ábrán látható, amely a kizáró-vagy tagadása logikai függvény




48. ábra

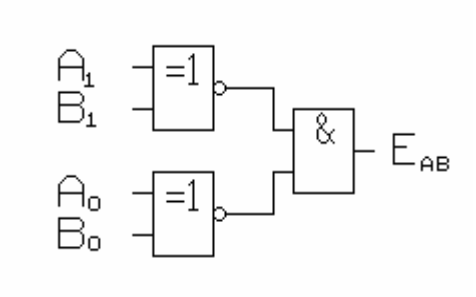
Több ***bites szám*** akkor ***egyenlő***, ha az ***azonos helyértékű*** bitek egyenlők. Legyen a a két szám:

$$\mathbf{Z_A = A_1 2^1 + A_0 2^0}$$
$$\mathbf{Z_B = B_1 2^1 + B_0 2^0}$$

Az egyenlőséget leíró logikai függvény:

$$\mathbf{E_{AB} = (A_1 B_1 + \overline{A}_1 \overline{B}_1)(A_0 B_0 + \overline{A}_0 \overline{B}_0)}$$

A függvényt megvalósító áramkör logikai vázlata a 49.ábra szerinti.




49. ábra

További bővítés az előzőek ismétlésével történik.

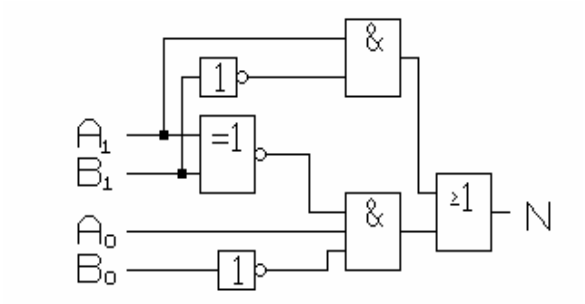
Két szám összehasonlításánál gyakran feladat - az egyenlőség jelzése mellett - a ***kisebb***, ill. ***nagyobb*** viszony kijelzése is.

A következőkben vizsgáljuk meg - két-bites számok összehasonlításánál - az egyenlőtlenségi relációkat jelző áramkörök működési feltételeit és határozzuk meg a logikai függvényeket.

A $\mathbf{Z_A > Z_B}$ akkor igaz, ha $\mathbf{A_1 > B_1}$, ill. ha $\mathbf{A_1 = B_1}$ és $\mathbf{A_0 > B_0}$. A leírt feltétel teljesülését az **N** logikai változó jelölje. Logikai függvényben ez a következőképpen fogalmazható meg:

$$\mathbf{N = A_1\overline{B}_1 + (A_1B_1 + \overline{A}_1\overline{B}_1)A_0\overline{B}_0}$$

A függvény első logikai ÉS kapcsolata fejezi ki az $\mathbf{A_1>B_1}$ feltételt, ugyanis csak az $\mathbf{A_1=1}$ és $\mathbf{B_1=0}$ esetén ad 1 értéket. A zárójeles rész az $\mathbf{A_1=B_1}$ feltételt teljesíti, míg az $\mathbf{A_0\overline{B}_0}$ tag az $\mathbf{A_0 > B_0}$ relációt adja. A függvényt megvalósító áramkör logikai vázlata látható az 50. ábrán.



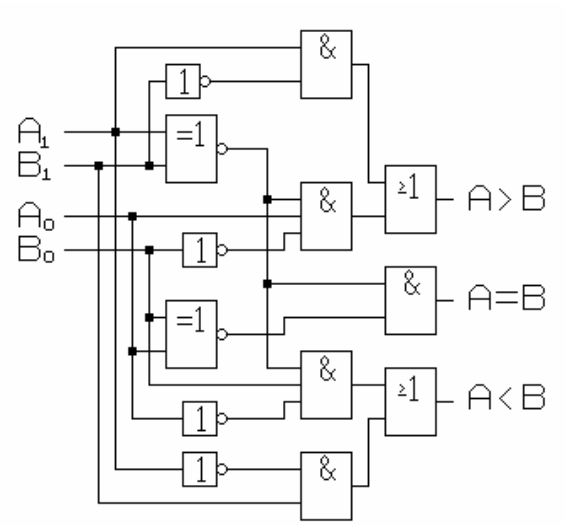

50. ábra

A $\mathbf{Z_A < Z_B}$ reláció logikai függvénye következik az előzőből, ha értelemszerűen felcseréljük a megfelelő biteknél a tagadást. Ezt a

$$\mathbf{K = \overline{A}_1 B_1 + (A_1 B_1 + \overline{A}_1 \overline{B}_1)\overline{A}_0 B_0}$$

logikai függvény fejezi ki.

A kétbites számok teljes összehasonlítását végző komparátor logikai vázlata a 51.ábrán látható.




51. ábra

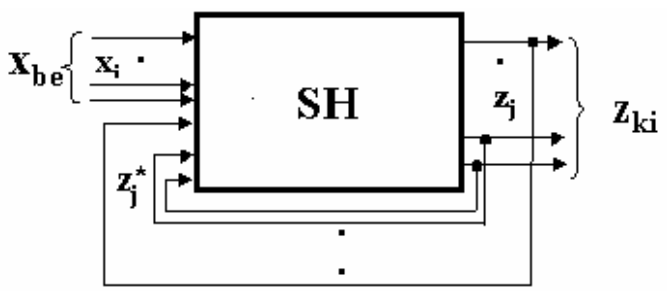
## 2.4. A sorrendi hálózatok

A logikai feladatok jelentős hányadánál – mint ahogyan ezt már az előzőekben leírtuk - a ***következtetések*** értéke - az éppen teljesülő ***állítások*** (feltételek) mellett – a

következtetések ***megelőző*** értékétől is függ. Miután egy feladatban ***állítások sorozata*** követheti egymást, ezért a ***következtetések*** is jól meghatározható ***sorozatot*** alkotnak. Ezek a **sorrendi** vagy **szekvenciális** logikai feladatok. Természetesen csak az olyan feladatok valósíthatók meg egyértelműen, amelyeknél egy új következtetést mindig egy megváltozott állítás „indít”.

Sorrendi feladatokat megvalósító logikai hálózat alapvetően két különböző módon építhető fel.

Az 1.ábra szerinti blokkvázlat szerinti felépítés az ***alapdefiníciónak*** felel meg, mely szerint a sorrendi logikai hálózat (**SH**) az új ***következtetéseket*** megadó ***kimeneti jelek*** ($\mathbf{Z_{ki}}$) értékkombinációját az éppen ***érvényes*** állításokból adódó ***bemeneti jelek*** ($\mathbf{X_{be}}$), valamint ***kimeneti jelek*** ( $\mathbf{Z_{ki}}^*$ ) értékkombinációjából határozzák meg. ( A kimeneti jeleknél a *-al azt jelezzük, hogy állapotváltozáskor az előző állapothoz tartozó kimeneti jel. A $\mathbf{Z_{ki}}$ és a $\mathbf{Z_{ki}}^*$ jelkombinációk a változáskor különböző értékek, viszont stabil - állapotban azonos értékek.).




52. ábra

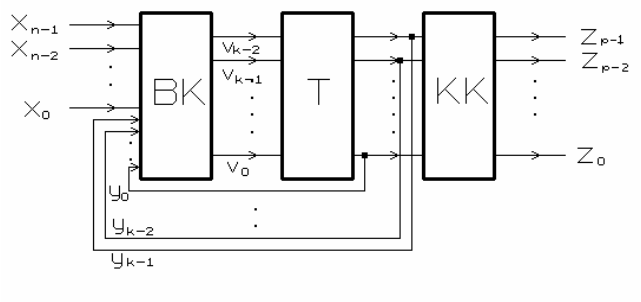
A kimeneti jelek ***visszavezetése*** következtében az új állapotkor létrejövő kimeneti változók kombinációi függnek a bemeneti-, és az előző állapot kimeneti kombinációitól. A meghatározást az alábbi összefüggés írja le:

$$\mathbf{Z_{ki} = f_z\,(X_{be},\, Z_{ki}{}^*)}$$

ahol $\mathbf{f_z}$ a kimenetek, és a bemeneti-, valamint előző állapot közötti logikai kapcsolatot (logikai függést) adja meg. Gondolati kísérlettel belátható, hogy a 1.ábra szerinti hálózat csak akkor stabil, ha $\mathbf{Z_{ki}}^* = \mathbf{Z_{ki}}$ és a bemeneti változók is **állandósultak**. Ez

csak - egy bemeneti kombinációváltást követően - késleltetve, legkevesebb a hálózat ($\mathbf{t_H}$) késleltetési ideje múlva következhet be.

A feladatok megvalósíthatóak úgy is, hogy egy ***tároló*** egység „emlékszik” az érvényes állapotra. A ***kimenetek*** jeleit - egy kombinációs hálózaton keresztül (***KK***) - a tároló kimeneti jelei – $y_i$ ***állapotjelek*** határozzák meg. A bemeneti értékek változása és a tárolt állapotjelek csak együtt – ugyancsak egy kombinációs hálózaton keresztül (***BK***) – állítják elő a vezérlő $v_j$ vezérlőjeleket, amelyek meg változtathatják a tárolók állapotát, és ezáltal hoznak létre a kimeneteken új jeleket. A leírt megoldás blokkvázlata látható a 2. ábrán.




53. ábra

Az állapotjeleket **y** -al, míg a tárolók vezérlőjeleit **v** -vel jelöltük.

Az alapdefiníciónak megfelelő függvénykapcsolat ekkor is érvényesül, mivel

$$\mathbf{Y_i = f_b\,(X_i,\, Y_i^*)}$$

$$\mathbf{Z_{ki} = f_z\,(Y_i)}$$

ahol $\mathbf{Y_i}$ a tárolók kimeneti jeleinek ($y_i$) aktuális kombinációja. Kiolvasható, hogy a kimeneti jelek ***aktuális kombinációja*** ($\mathbf{Z_{ki}}$) az $\mathbf{Y_i}$ –től függ. Ennek értéke viszont ez az $\mathbf{X_i}$ ***bemeneti jelkombináció***, és az ***előző állapotjelek*** ( $\mathbf{Y_i^*}$ ), végeredményben, pedig az ***előző kimeneti kombinációk*** ($\mathbf{Z_{ki}^*}$ ) függvénye.

***Az állapottárolókkal való megoldást olyan feladatok esetében célszerű alkalmazni, amelyeknél több kimenet van, mint ahány állapottároló-elemet (flip-flop -ot) kell felhasználni***.

### ⇒ Aszinkron, és szinkronműködés

Két **stabil** (állandósult) állapot között a kimeneti jelek – ***átmenetileg***, a belső késleltetésektől függően – ***több állapotkombinációt*** is felvehetnek, mielőtt állandósulna a ***kívánt új jelkombináció***. Az ***állapotváltozást*** vagy azok sorozatát a bemeneti jelek változása ***elindítja***, de a tranziensváltozásokat a visszacsatolás jeleinek (állapotjel) változása eredményezi. Az ilyen működésű hálózatot ***aszinkron sorrendi*** hálózatnak nevezzük. A következőkben az aszinkron magoldásnál csak számlálókat tárgyaljuk röviden.

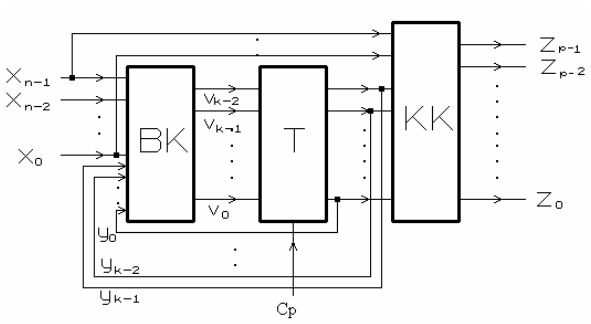
Sorrendi hálózat kialakítható olyan működéssel is, amelynél az egymást követő ***állapotváltozásokat*** az **x** bemeneti jelek megváltozása csak előkészíti, és egy ütemező jel, az un. ***szinkronozó*** („órajel" ) jel hajtja végre.

A két szinkronjel közötti időben, ***tárolni*** kell az állapotra jellemző információt. Ezek az ***állapotjelek*** (szekunder változók). Az aktuális ***bemeneti*** jelek, és a előző jel hatására tárolt ***állapotjelek*** előkészítik a kívánt új állapot vezérlőjeleit, és a következő szinkronjel fogja ezt az tároló egységbe beírni. ***Lényeges, hogy a szinkronozó jel aktív ideje alatt a bemeneti-, és az állapotjelek értéke ne változzon!***

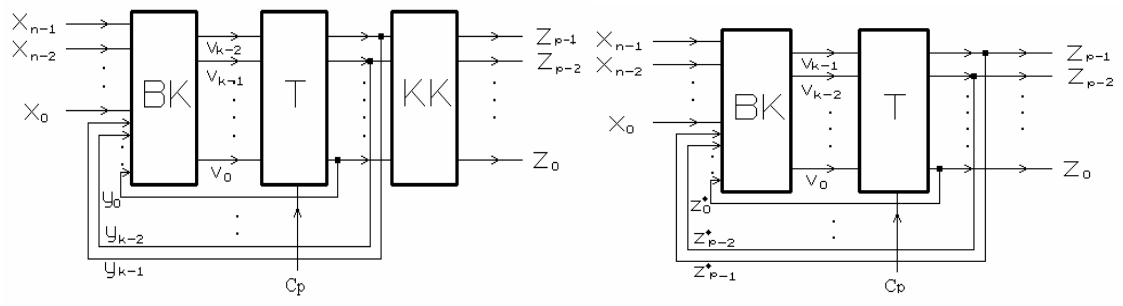
### ⇒ Szinkron sorrendi hálózat rendszertechnikai felépítése.

A leírtak szerint működő ***szinkron sorrendi hálózat***, különböző felépítés szerint célszerű megvalósítani (2. ábra). Mindhárom változatban a **T** ***állapottárolók*** új vezérlőjeleit ($v_j$) a ***bemeneti kombinációs*** hálózat (**BK**) állítja elő az $x_n$ ***bemeneti***-, és az $y_k$ ***állapotjelekből***. Az állapottárolókat az ütemező jel (**Cp**) billenti a $v_j$ által meghatározott ***új állapotba***.

Az a. ábra szerinti megvalósításban a kimenetek jeleit ($z_p$) egy kombinációs hálózat (**KK**) az ***állapotjelekből*** és közvetlenül a ***bemeneti*** jelekből, ***vagy*** azok egy ***részéből*** állítja elő. Ez a megoldás az un. ***Mealy- modell***.




a.




b. c.

54. ábra

A b. ábra szerinti változatban a ***kimenetek jeleire*** csak a tárolt ***állapotjeleken*** keresztül hatnak a ***bemeneti jelek***. A kimeneti kombinációs hálózat (**KK**) csak az állapotjelekből ($y_k$) állítja elő a hálózat kimeneti jeleit, $z_p$ -ket. E változat az un. **Moore- modell**. Az utóbbi megoldásnál esetleg több tárolóra van szükség, de egyszerűbb a felépítés. A korszerű integrált áramkörök alkalmazásával már elhanyagolandó szempont lett a tárolók száma ( különösen a programozott rendszerekben ), s ezért a Moore modell szerinti felépítés mind hardverben, mind pedig a szoftveres megoldásban nagyobb teret kap.

A Moore - modell egy változatát mutatja a c. ábra. Itt nem használunk külön kimeneti kombinációs hálózatot, hanem az $y_k$ állapotváltozókat állítjuk elő oly módon, hogy azok, vagy egy részük egyúttal a hálózat kívánt kimeneti változói is. Elsősorban a számlálóknál találkozunk ezzel a változattal.

Mindhárom felépítésű megoldásban a hálózat állapota, s így a kimenő jelek is az ***szinkronozó-jel*** (Cp) ütemezésében váltanak értéket.

Tételezzük fel, hogy a vizsgált $\mathbf{t_i}$ időpillanatban - amely a két ***szinkronozó-jel*** közötti időpont - a hálózati tranziensek lejátszódtak, a hálózat állapotát és a kimeneti értékeket a $\mathbf{Z_i}$ kimeneti jelkombináció írja le. Ugyanebben az ***előkészítési fázisban*** az új bemeneti jelkombináció állandósult értéke $\mathbf{X_i}$. Ekkor a **BK** állítja elő a **T** tárolók $\mathbf{v_{ji}}$ vezérlőjeleit bemeneti kombinációs hálózat bemenetén lévő $\mathbf{X_i}$ és $\mathbf{Y_i = Z_i}$ bemeneti értékekből. A $\mathbf{t_{i+1}}$-edik időpontban érkező szinkronozó-jel fogja - a $\mathbf{v_{ji}}$ által meghatározott állapotba - billenteni a tárolókat. Ennek eredményeként alakul ki az új ($Z_{i+1}$) kimeneti jelkombináció, ami egyúttal a következő mintavételezéshez tartozó állapotjel is. Az előzőek alapján felírhatjuk a

$$\mathbf{Z_{i+1} = fz(v_{ji})}$$
$$\mathbf{v_{ji} = fv(X_i, Z_i)}$$

függvénykapcsolatokat. Az **fz** kimeneti függvény az előállítani kívánt kimeneti ($Z_{i+1}$) és a tárolókat vezérlő $\mathbf{v_{ji}}$ jelek - billentés előtti - értékei közötti logikai kapcsolatot adja meg.

## ⇒ Sorrendi feladatok logikai leírása

A sorrendi logikai feladatokat megvalósító sorrendi hálózatok tervezéséhez szükségünk van a kívánt működést egyértelműen megadó, az ismert hálózattervezési módszerek alkalmazását elősegítő leírásra. A leírás egyértelműen kell meghatározza (jelölje)

***az állandósult állapotokat,***

***az állapotátmeneteket indító bemeneti jelkombinációkat,***

***az állapotátmenetek irányát,***

***a kimeneti jelek állandósult értékeit.***

A továbbiakban röviden ismertetünk egy-egy, ***a logikai kapcsolatokat***

***állapotgráf*** -al jelölt grafikus,

az ***állapottáblázat*** -ba foglalt táblázatos,

a kimenetek ***állapotfüggvényét*** megadó algebrai, és

a ki-, valamint bemeneti jelek időbeli változását mutató ***ütemdiagram*** ( állapot-diagram )grafikus

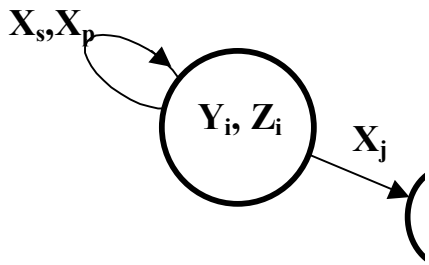
feladat-leírási módszert.

## ⇒ Állapotgráf

A sorrendi logikai feladatokhoz gyakran használják az **állapot-gráfnak** nevezett szemléltető leírást, amelynek elemeit mutatja az 55. ábra.

A hálózat minden ***állandósult állapotát*** egy **körrel -** gráf-csomópont – jelöljük. A körökbe az adott állapothoz tartozó ***állapot-*** (***Yi***), és a ***kimeneti*** (***Zi***) változók kombinációját írjuk.

A körökből **nyilak** indulnak ki, amelyek vagy egy ***másik***-, vagy az ***induló*** körnél (állapotnál) végződnek. Ezek jelzik a lehetséges állapotátmeneteket, és irányukat. A nyilakra ráírjuk az ***állapotváltozást*** elindító bemeneti kombinációt ($X_j$).

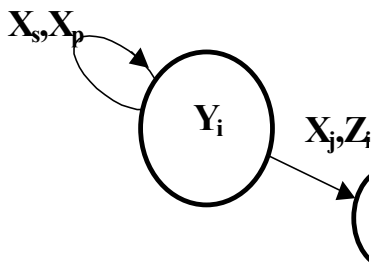



55. ábra

Minden körtől annyi nyíl indul, amennyi az ***állapotváltozást okozó bemeneti kombinációk*** száma. Ha több kombináció eredményez azonos állapotátmenetet, akkor azokat egyazon nyílra írjuk. Az ábrán pl. az $X_s$, $X_p$ jelű bemeneti kombinációk nem

indítanak állapotváltozást, míg az $\mathbf{X_j}$ egy másik állapotot jelző körig tart . A nyíl irányítása adja meg az állapotváltozás irányát.

A bemutatott állapotgráf részlet olyan megoldásra utal, amelynél a ***kimenetek*** kombinációját csak az ***állapotjelek*** határozzák meg (Moore modell), és ezért a körbe írjuk a kimeneti kombináció (***$Z_i$***) jelét. Amikor a kimeneti kombinációt az állapotváltozást jelző nyílra írjuk, azt jelezzük, hogy a kimeneti ***értékváltozást*** már a ***bemeneti jelváltozás*** indítja (Mealy modell) .(56.ábra).




56. ábra

Egy sorrendi feladatot leíró állapot-gráf felrajzolásához ismernünk kell:

az állandósult ***állapotok számát***, vagyis a ***modulust*** (m),

a szükséges ***kimeneti*** jelek ***kombinációját***,

az állapotváltozásokat ***eredményező bemeneti*** jelkombinációkat.

A leírt kiinduló adatok ismeretében az alábbi lépésekben kell megrajzolni az állapotgráfot:

megrajzoljuk az ***m*** darab ***állapotcsomópontot*** (kört),

beírjuk a körökbe az ***állapotjellemző*** (Y) jelét, és indexét, valamint a ***kimeneti*** jelkombináció (Z) jelét és indexét

körökre rárajzoljuk a ***visszatérő nyilat***, és arra felírjuk azokat a ***bemeneti*** jelkombinációk (X) jeleit, és indexeit, amelyek az adott állapotot ***nem változtatják*** meg,

csompontonként felrajzoljuk az ***állapotváltozást*** jelentő nyilakat, és azokra felírjuk a ***kiváltó*** bemeneti jelkombináció jelét, és indexét.

**⇒ Állapottáblázat**

Egy sorrendi hálózat állapotai, valamint a bemeneti-, és kimeneti változói közötti kapcsolatrendszert táblázattal, az un ***állapottáblázattal*** is megadhatjuk. A táblázat minden egyes sora egy állandósult állapotot jelent, és ezt az $\mathbf{Y_i^*}$ ***állandósult állapotváltozóval*** jelöljük. Az oszlopok a lehetséges állapotváltozásokat okozó bemeneti jelkombinációkat $\mathbf{X_k}$ jelentik. A táblázat celláiba kell beírni, hogy az adott állapotból (sor) milyen ***új állapotba*** $\mathbf{Y_j}$ viszi a hálózatot az oszlop által jelölt bemeneti jelkombináció. Ugyancsak a cellába kell jelölni, hogy milyen kimeneti kombináció $\mathbf{Z_s}$ érvényes az adott állapotban.

| Áll. jell. | Bemeneti jelkombinációk | | |
|---|---|---|---|
| | $X_0$ | | $X_p$ |
| $Y_0^*$ | $Y_1, Z_0$ | | $Y_0, Z_0$ |
| | | | |
| | | | |
| $Y_j^*$ | $Y_0, Z_s$ | | $Y_j, Z_s$ |

57. ábra

Az 57. ábra szerinti táblázat bal felső cellájában az $\mathbf{Y_1}$ bejegyzés azt jelenti, hogy ha a hálózat $\mathbf{Y_0}$ ***állapotban van*** és a ***bemeneti*** kombináció $\mathbf{X_0}$ ra vált, akkor a ***hálózat új állapota*** $\mathbf{Y_1}$ lesz. Amikor az állapotsor indexe megegyezik a cellába írt index-el, akkor – az oszlop szerinti bemeneti kombináció – ***nem okoz*** állapotváltozást ( pl. az első sor utolsó cellája).A példa táblázatban a kimeneti kombinációk $\mathbf{Z_s}$ egy sorban azonosak, és azt jelzi ez, hogy értékét csak az aktuális állapot határozza meg (Moore modell). A Mealy modell szerinti megoldás állapottáblázatának egy sorában különböző kimeneti kombinációk is lehetnek.

A leírtak szerint az állapotgráf, és az állapottáblázat ugyanazt írja le. A táblázat ***sora*** felel meg a ***gráf-csomópontnak***, és az ***oszlopok*** jelentik a ***nyilakat***.

### ⇒ Állapotfüggvény

A sorrendi hálózatok minden kimenetére felírható egy algebrai alakú logikai függvény. Ezek a függvények abban különböznek a kombinációs feladatoknál megismert logikai függvényektől, hogy független változói között szerepelnek az állandósult kimeneti értékek is. Általánosan tehát a $\mathbf{Z_j}$ kimeneti kombináció függvénye az állandósult $\mathbf{Z_k}$ kimeneti-, és az $\mathbf{X_i}$ bemeneti kombinációknak.

$$\mathbf{Z_j = f(X_i, Z_k^*)}$$

Az adott összefüggést úgy is értelmezhetjük, hogy a $\mathbf{Z_j}$ kimeneti kombináció akkor következik be, ha a hálózat kimenetén $\mathbf{Z_k}$ kombináció van, és a bemenetekre az $\mathbf{X_i}$ jelkombinációra vált.

### ⇒ Ütem- (állapot-) diagram

Az első fejezetben már megismertük a logikai függvények idő-diagramban történő ábrázolását. Tulajdonképpen a sorrendi hálózatok be-, és kimeneti jelei is ugyanúgy ábrázolhatók az idő függvényében. Ránézésre nem állapítható meg azonnal, hogy kombinációs-, illetve sorrendi-hálózat jeleit látjuk-e. A lényeges eltérés, hogy egy kimeneti jel változását a bemeneti-, és a kimeneti jelek előző értékei együtt határozzák meg.

### ⇒ A sorrendi hálózat áramköri megvalósítása

Az előzőekben megismert feladat-leírási módszerek közül

- az ***állapotgráf***

  szemléletes, de csupán a feladat értelmezését segíti,

- az ***állapottáblázat***

alapján az áramköri tervezést – a táblázat kódolása, és felbontása után - elvégezhetjük,

- az ***állapotfüggvény***

segítségével, az esetleges algebrai egyszerűsítés után tervezhetjük meg az áramkört,

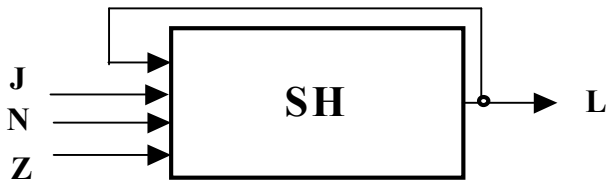
- az ***ütemdiagram*** -ot

a PLC- megjelenése előtt főleg a relés vezérlések tervezésénél használták

### *11.Példa*

Tervezzük meg egy autóbusz ajtajának nyitását kérő jelzőlámpa vezérlését. Az ***L*** jelű ***lámpa*** kezdjen világítani, ha az ajtó zárva van, és megnyomjuk a ***J*** jelű ***nyomógombot***. A világítás szűnjön meg, ha kinyílt az ajtó. Az ajtó zárt állapotát a ***Z*** jelű, míg nyitott állapotát az ***N*** jelű ***érintkezők*** zárása jelzi.

A feladat egy három bemenetű, és egy kimenetű sorrendi (emlékező) hálózattal valósítható meg, amelynek blokkvázlatát szemlélteti az 58.ábra.




58. ábra

Állapotok száma: ***m = 2***

Kimenetek szám **1** (L), tehát két kimeneti kombináció van $\mathbf{Z_0}$ (L=0), **Z1** (L=1).

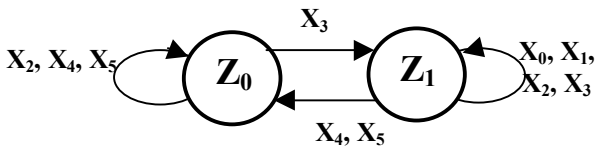
Bemenetek száma **3** (J,N,Z), tehát az alábbi nyolc bemeneti kombináció lehetséges.

| | **N** | **Z** | **J** | **Jelentés** |
|---|---|---|---|---|
| $\mathbf{X_0}$ | 0 | 0 | 0 | Ajtó közbenső helyzetben, jelzés nincs |
| $\mathbf{X_1}$ | 0 | 0 | 1 | Ajtó közbenső helyzetben, jelzés van |

| | | | | |
|---|---|---|---|---|
| $\mathbf{X_2}$ | 0 | 1 | 0 | Ajtó zárt helyzetben, jelzés nincs |
| $\mathbf{X_3}$ | 0 | 1 | 1 | Ajtó zárt helyzetben, jelzés van |
| $\mathbf{X_4}$ | 1 | 0 | 0 | Ajtó nyitott helyzetben, jelzés nincs |
| $\mathbf{X_5}$ | 1 | 0 | 1 | Ajtó nyitott helyzetben, jelzés va |
| $\mathbf{X_6}$ | 1 | 1 | 0 | Nem fordulhat elő |
| $\mathbf{X_7}$ | 1 | 1 | 1 | Nem fordulhat elő |

Az $\mathbf{X_6}$ és $\mathbf{X_7}$ kombinációk azt jelentik, hogy mindkét érintkező zárt, ami viszont sohasem fordulhat elő.

***Állapotgráf***:




***Állapottáblázat***:

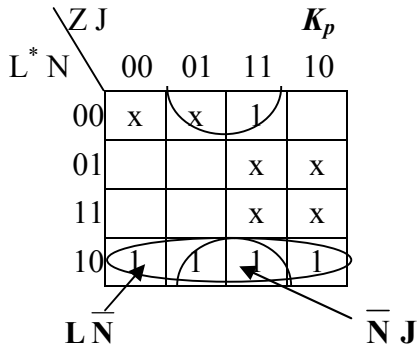
| $Z^*$ | Bemeneti kombinációk | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $\mathbf{X_0}$ | $\mathbf{X_1}$ | $\mathbf{X_2}$ | $\mathbf{X_3}$ | $\mathbf{X_4}$ | $\mathbf{X_5}$ | $\mathbf{X_6}$ | $\mathbf{X_7}$ |
| $\mathbf{Z^*_0}$ | x | x | $\mathbf{Z_0}$ | $\mathbf{Z_1}$ | $\mathbf{Z_0}$ | $\mathbf{Z_0}$ | x | x |
| $\mathbf{Z^*_1}$ | $\mathbf{Z_1}$ | $\mathbf{Z_1}$ | $\mathbf{Z_1}$ | $\mathbf{Z_1}$ | $\mathbf{Z_0}$ | $\mathbf{Z_0}$ | x | x |

x – el jelöltük azokat a bemeneti kombinációkat, amelyek az adott állapotban nem fordulhatnak elő, csak hiba esetén. Pl. az $\mathbf{Z^*_0}$ állapotban $\mathbf{X_0}$ azért nem fordulhat elő, mert ebben a kombinációban az N, és Z érzékelők közül egyik sem zárt, amely csak az ajtó nyitása közben fordulhat elő. Ha nem világít a lámpa, akkor nincs ajtónyitás. Az $\mathbf{X_6}$ és $\mathbf{X_7}$ kombinációkról már írtunk.

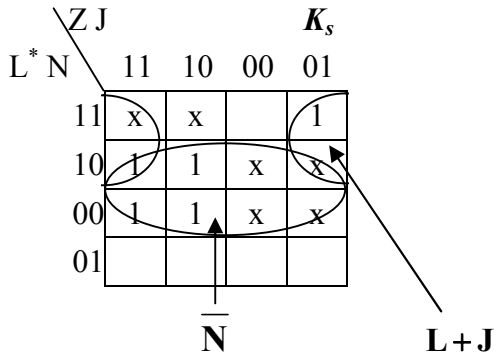
***Kódolt állapottáblázat:***

| $L^*$ | Bemeneti kombinációk | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $X_0$ | | | $X_1$ | | | $X_2$ | | | $X_3$ | | | $X_4$ | | | $X_5$ | | | $X_6$ | | | $X_7$ | | |
| | N | Z | J | N | Z | J | N | Z | J | N | Z | J | N | Z | J | N | Z | J | N | Z | J | N | Z | J |
| | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | x | | | x | | | 0 | | | 1 | | | 0 | | | 0 | | | x | | | x | | |
| 1 | 1 | | | 1 | | | 1 | | | 1 | | | 0 | | | 0 | | | x | | | x | | |

***Az L kimenetre érvényes Kp diagram:***




***Az L kimenetre érvényes Ks diagram:***



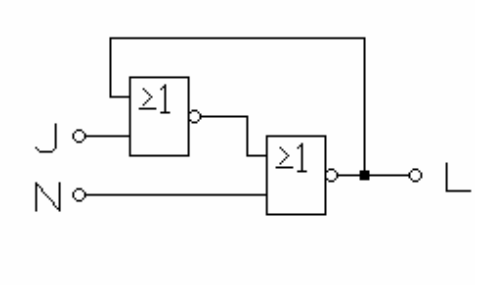

***A kimenet állapotfüggvényei:***

$$L = L\overline{N} + \overline{N}J = \overline{N}(L + J) \qquad \text{(a Kp diagram alapján)}$$

$$L = \overline{N}(L + J) \qquad \text{(a Ks diagram alapján)}$$

A két megoldás ugyanazt az eredményt adta.

***Az áramkör logikai vázlata:***




59. ábra

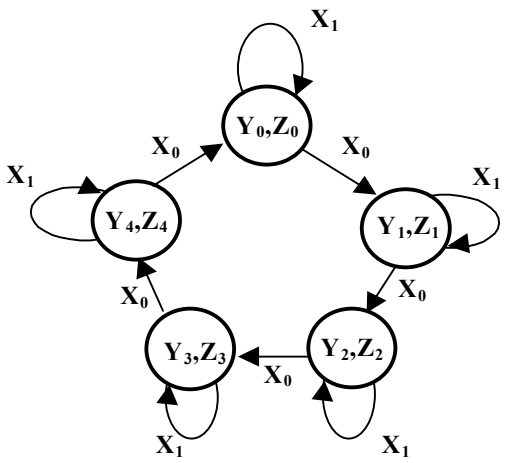
A példa megoldása a tároló alapáramkör az un. RS flip-flop. (A tárolók tárgyalásánál térünk vissza e megoldásra).

## ⇒ Sorrendi hálózatok főbb típusai

A sorrendi hálózatok egyik csoportosítása az ***állapot-sorozatok*** száma alapján is történhet. Beszélhetünk ***egy***-, és ***több-szekvenciájú*** sorrendi hálózatokról.

- Az **egy-szekvenciájú** hálózatban

  az állapotok ***mindig ugyanabban a sorrendben*** követik egymást. A 59.ábrán egy öt állapotú – egy-szekvenciájú – sorrendi hálózat állapotráfja látható.



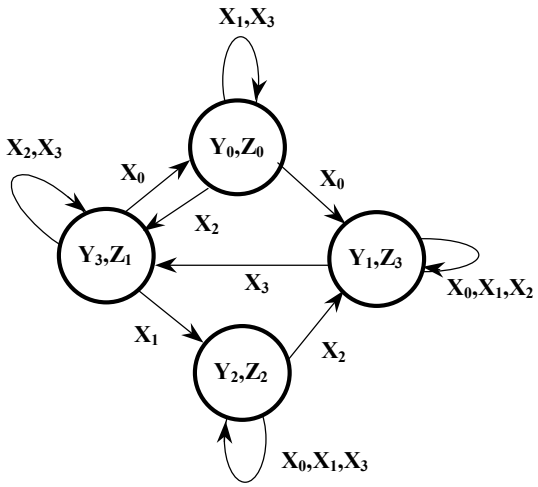

60. ábra

Itt az **….$Y_0$ – $Y_1$ – $Y_2$ – $Y_3$ – $Y_4$ – $Y_0$ - ……**állapotsor ismétlődik. Az **$X_0$** bementi jelkombináció indít minden állapotváltást.

A sorrendi hálózatok ilyen változatát ***lefutó típusú*** -nak is szokták nevezni.

- A **több-szekvenciájú** hálózat

Olyan változat, amelyben ***több***, egymástól ***eltérő állapotsorozat*** is felléphet a különböző bemeneti jelkombináció-sorozat hatására. Egy négy állapotú ***általános*** – több szekvenciájú - sorrendi hálózat állapot-gráfja látható a 60.ábrán.




61. ábra

A példa szerint működő hálózatban lehetséges szekvenciák közül néhányat írtunk fel a következő sorokban.

**….$Y_0$ – $Y_1$ –$Y_3$ –$Y_0$ - ……**

**….$Y_0$ – $Y_1$ – $Y_3$ – $Y_2$ – $Y_1$ – $Y_0$ - ……**

**….$Y_0$ – $Y_3$ – $Y_2$ – $Y_1$ – $Y_3$ – $Y_0$ - ……**

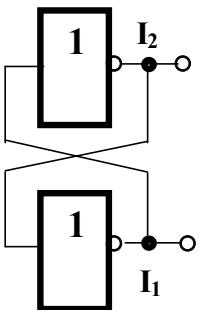
## 2.5. Sorrendi hálózatok alapelemei

A **sorrendi**, vagy más néven **szekvenciális** feladatok megvalósításához – az eddig megismert kapukon kívül - olyan elemekre is szükség van, amelyek az ***elemi***

***információt – bit -*** et **– *tárolják.*** A következőkben ismertetjük a leggyakrabban alkalmazott tároló-elemek (**flip-flop**) felépítését, és működését.

- **Tároló alapáramkörök**

A tároló alapáramkörök - flip-flop -ok - **két stabil** állapotú áramköri kapcsolások. A két stabil állapot **1 bit** információ **tárolására** teszi alkalmassá a flip-flop -ot.
A kétállapotú elem két keresztbecsatolt inverter -ből alakítható ki. (62. ábra)




62. ábra

A két inverter **keresztbe csatolása** biztosítja a felvett állapot tartását. Az ábra szerint elrendezésben a tápfeszültség bekapcsolása után – a két inverter kapcsolási sebességének különbözősége miatt - véletlenszerűen alakul ki a stabil helyzet. Ha az $\mathbf{I_1}$ jelű inverter kimenetén **1** szint lesz, az $\mathbf{I_2}$ bemenetére jutva biztosítja ennek a kimenetén a **0** szintet. A keresztbecsatolás fent tartja az $\mathbf{I_1}$ kimenetén az **1** szintet.

A fentiekben röviden elemzett áramkörnek nincs állapotváltozást **vezérlő** bemenete. A kívánt állapotváltozást a **vezérlő**-bemenetek és **billentési módok** különböző változataival lehet megoldani. A megoldási módozatok alapján csoportosítjuk a flip-flop -kat.

⇒ **Flip-flop típusok**

A flip-flop -ok két nagy csoportba sorolhatók annak alapján, hogy az **információ**-közlést és a **billentés**-t (az állapot beállítását) ugyanaz, vagy különböző jelek látják-e el. Ennek megfelelően:

- **közvetlen,** és

- **kapuzott** vezérlésű

tárolókat különböztünk meg.

A tárolandó értéket (információt) közlő bemenetek alapján leggyakrabban alkalmazott típusok az

- **RS,**
- **JK,**
- **T és**
- **D típusú** flip-flop -ok**.**

Az egyes flip-flop -ok **billentési módja** szerint megkülönböztetünk:

- **statikus és**
- **dinamikus billentési megoldásokat.**

A kapuzott vezérlésű tároló elemek között - elsődlegesen az integrált áramköri kialakításban - további két nagy csoport létezik, a

- **együtemű, és**
- **kétütemű vezérlésű**

áramköri változat. A kétütemű vezérlést ***közbenső tároló*** alkalmazásával valósítják meg.

Az ***aszinkron***, illetve a ***szinkron-***működési módot, már a flip-flop -ok esetében is értelmezhetjük. ***Aszinkron*** működésűnek nevezhetjük azokat a flip-flop - kat, amelyeknél a beírandó értéket (információt), valamint ennek beírását a tároló elembe ugyanazon jel végzi. Ilyenek a ***közvetlen*** vezérlésű tárolók. Értelemszerűen a ***szinkron-***működésű tárolóknál a két vezérlési funkciót – információ, és tárolást (billentést) – vezérlő jelek különbözőek. A ***kapuzott*** vezérlésű tárolok, alkotják ezt a csoportot.

A további tárgyalásoknál többször is beszélünk valamelyik bemenet ***aktív vezérlési*** szintjéről. A fogalom azt jelenti, hogy melyik az a logikai érték, amely a jelölt funkciót (beírás, törlés, billentés stb.) vezérli.

Az általános csoportosítás után először az információs (tárolandó értéket közlő) bemenetek alapján megkülönböztetett típusok elvi működését elemezzük.

- Az **RS flip-flop** olyan tároló, amelynek két információt közlő bemenete van, amelyek közül az **S** jelű (set) a beíró és az **R** jelű (reset) a törlő bemenet. Ezek szerint a tárolt információ **1** lesz, ha a ***beíró*** bemenetre (S) érkezik ***aktív*** logikai szintű vezérlő jel, és **0** érték lesz, ha a ***törlő*** (R) bemenet kap ilyen vezérlést. A helyes működés feltétele, hogy a két vezérlőbemenet ***együttesen*** nem kaphat aktív vezérlést.

- A **JK flip-flop** ugyancsak két információt közlő bemenete van. A **J** jelű bemenet ***beíró***, míg a **K** jelű a ***törlő*** feladatokra szolgál. A fentiekben ismertetett RS flip-flop -tól az különbözteti meg, hogy ***engedélyezett*** a J és K ***együttes*** aktív vezérlése is. Ebben az esetben a flip-flop a tárolt állapot ***ellenkezőjére*** (komplemens -ére) vált át.

- A **T flip-flop** egyetlen vezérlőbemenettel rendelkező tároló elem. A **T** bemenetre jutó aktív vezérlés a tároló állapotát ***ellenkezőjére*** változtatja.

- A **D flip-flop** –nak ugyancsak egyetlen vezérlőbemenete van. A tároló mindenkor a D bemenet logikai értékét tárolja, vagyis **D=0** esetén a tároló ***törlődik***, míg **D=1** értéknél ***beíródik***. A leírt működés alapján a tárolót ***adat*** flip-flop -nak is nevezik.

⇒ **Statikus billentésű flip-flop -ok**

- Statikusnak nevezzük azt a billentési módot, melynél a vezérlőjel logikai szintje a hatásos. A flip-flop mindaddig vezérelt állapotban van, míg a bemeneten az aktív logikai szint érvényes. Aktív lehet a logikai 1 és a logikai 0 szint is. A kívánt aktív vezérlés kiválasztása után a megvalósítandó flip-flop ***működési***, vagy ***állapot táblázatából*** felírhatók működést leíró ***állapot-egyenletek***. Ezek alapján a szükséges vezérlési megoldás áramköri változata kialakítható.

A ***statikus billentésű*** - logikai 1 szinttel vezérelt - **RS** flip-flop állapottáblázata a 63. ábrán látható táblázat szerinti.

| $S_n$ | $R_n$ | $Q_n$ | $Q_{n+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | * |
| 1 | 1 | 1 | * |

63. ábra

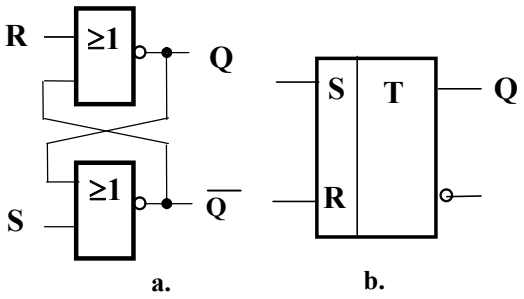
A táblázat oszlopai között a $\mathbf{Q_n}$ mint bemenő változó szerepel (a változás előtti állapot). A vezérlés utáni ***új állapotot*** ($\mathbf{Q_{n+1}}$) a ***vezérlés*** ($R_n$, $S_n$) mellett az ***előző állapot*** ($Q_n$) is befolyásolja. A *-al jelölt vezérlési kombinációk ***tiltottak***. A táblázatból felírható ***állapotfüggvények*** az alábbiak:

$$\mathbf{Q_{n+1} = S_n + \overline{R}_n Q_n}$$
$$\mathbf{S_n R_n = 0}$$

(Az összefüggésben és a továbbiakban is az **n** index a vezérlés időpontjára utal.)

Ezek a logikai függvények az **RS** flip-flop ***működését*** írják le. Az összefüggés szerint az új állapot ($Q_{n+1}$) 1 szintű lesz - az előző állapottól függetlenül - ha a beíró (S) bemenet 1 szintű. Ugyancsak 1 szintű lesz a kimenet, ha már a vezérlés előtti állapotban is $Q_n = 1$ és az R bemeneten 0 szint van. A második összefüggés a ***tiltott vezérlést*** adja meg. Eszerint a két vezérlő bemeneten, együttesen nem lehet 1 szint.

A fentiek szerint működő RS flip-flop két NOR kapuból alakítható ki a 64.a. ábra szerinti kapcsolásban. Az 1 szinttel vezérelhető RS flip-flop ***szimbolikus*** jele a b. ábra szerinti.




64. ábra

A kapcsolás működésének elemzése alapján könnyen belátható, hogy azért kell tiltani az együttes aktív vezérlést, mert ekkor mindkét kapu kimenete 0 szintű lesz. Az új állapot, pedig a vezérlőjelek megszűnésének sorrendjétől függ, ezért – legtöbbször - előre nem határozható meg.

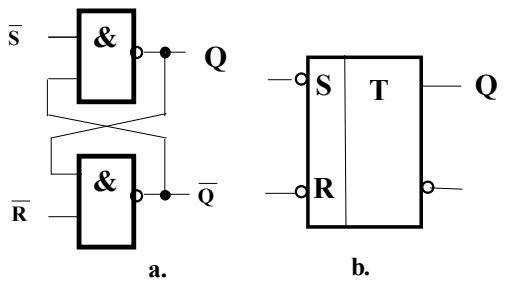
A **0** ***aktív vezérlési*** szintre billenő flip-flop működését az

$$\mathbf{Q_{n+1} = (\overline{S}_n + Q_n)R_n}$$
$$\mathbf{S_n + R_n = 1}$$

állapot egyenletek írják le .

Az összefüggés szerint az új állapot **1** lesz, ha a törlő bemenet (R) **1** (nem aktív), és a beíró bemenet (S) **0** (aktív) szintű, vagy vezérlés előtt is $Q_n = 1$ állapot volt.

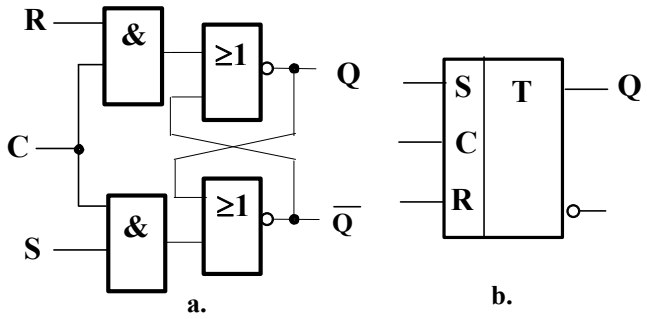
A második összefüggés írja le a tiltást, miszerint a két bemenet legalább egyikén 1 szintnek kell lenni. Áramkörileg NAND kapukkal valósítható meg statikus billentésű - 0 szinttel vezérelhető - RS flip-flop (63. ábra)




65. ábra

Az ismertetett két flip-flop ***közvetlen*** vezérlésű. A tárolandó információt hordozó ***beíró*** vagy **törlő** jel egyúttal a ***billentést*** is vezérli. A beírandó adatot hordozó-, és a billentő jelet kapuzással lehet fizikailag szétválasztani.

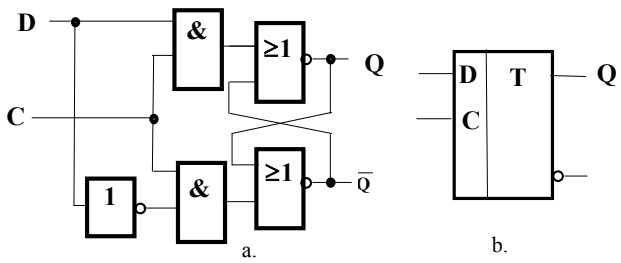
Statikus 1 szinttel vezérelt RS flip-flop vezérlőbemeneteit **C *billentő jellel kapuzva*** (66. ábra) kapjuk a kapuzott (szinkronozott) vezérlést. Ennél a flip-flop típusnál az R és S együttes aktív vezérlése csak a billentő (C) jel 1 szintjénél tiltott. Az **S** és **R** információs bemeneteken az ***előkészítés*** és a **C** jel hatására a tényleges beírás vagy törlés, vagyis az ***adat*** (információ) ***bevitel*** következik be.




66. ábra

A statikus billentésű 0 szinttel vezérelt RS flip-flop kapuzása VAGY kapukkal oldható meg, miután az aktív szint 0 mind a billentő, mind pedig az információs bemeneteknél.

A kapuzott RS flip-flop -ból alakítható ki a **D** flip-flop. Az inverter biztosítja a beíró és törlő bemenetek ellentétes szintű vezérlését (67. ábra). A **D = 1** szintnél az **S** előkészítő bemeneten **1**, míg az **R** bemeneten **0** szint lesz. A **C** (szinkronozó) bemenetre érkező 1 szint a flip-flop –ot **1-be *billenti***, vagyis ettől kezdődően **1-et *fog tárolni***.




67. ábra

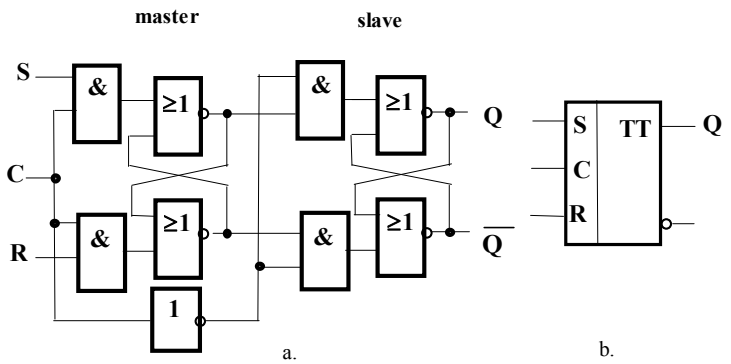
**D = 0** esetében a törlés előkészítése, és a C jel hatására a **0** beírása következik. A D flip-flop két szinkronozó jel közötti időtartamra tárolja az információt. A D tároló egyik legfontosabb felhasználási területe az ***információ-bevitel szinkronozása*** a C jel által meghatározott ütemezésben.

Az eddigiekben elemzett két flip-flop típus (RS és D) fő jellemzője, hogy billentő jel hatására az új információ - a billenési idő elteltével - azonnal megjelenik a kimeneten. Az ilyen működésű flip-flop -ot nevezzük egy-ütemű billentésű tárolónak. A ***szimbolikus jelbe*** beírt **egy T** betű jelenti az ***együtemű*** működést.

A digitális módon megvalósított jelfeldolgozásokban jelentős helyet foglalnak el azok a feladatok, melyekben az alkalmazott flip-flop -ok vezérlőbemenetére kimenetük értékét is ***vissza*** kell ***vezetni***. Ilyen esetekben csak olyan flip-flop -ok alkalmazhatók, melyek kimenetén csak akkor jelenik meg az új állapot értéke, amikor a bemeneti vezérlés már hatástalan. Ez az igény ***közbenső-tárolással*** vagy ***élvezérelt*** billentéssel oldható meg.

### ⇒ Közbenső tárolós (ms) flip-flop

A közbenső tárolós **ms** (master-slave) flip-flop legegyszerűbb elvi változata a 66. ábra szerinti két kapuzott RS flip-flop -ból áll. Amíg a **C** billentő jel szintje **0**, addig a külső (RS) bemenetek szintjétől függetlenül az első flip-flop (master) $\mathbf{R_1}$ és $\mathbf{S_1}$ bemenetein is **0** szint van. A két flip-flop -ot elválasztó kapukra jutó C = 1 szintű jel hatására a master állapota átíródik a második (slave) flip-flop –ba.




68. ábra

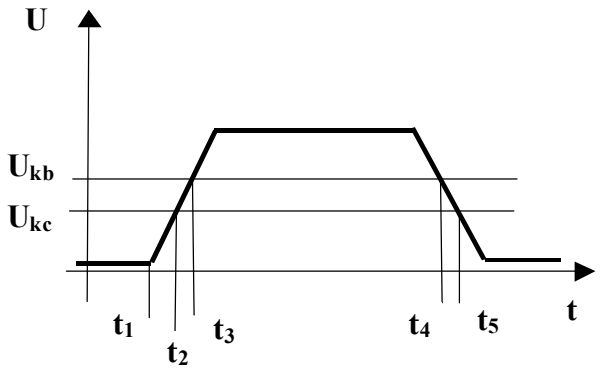
Amikor a **C** jel logikai **1** szintű, akkor a bemeneti vezérlés (S és R értéke) határozzák meg az első flip-flop állapotát, és ***tiltott*** a két tároló közötti csatolás. A ***második flip-flop*** változatlanul tárolja az ***előző információt***, és ezért a kimenet logikai értéke is változatlan. Az ***új információ*** a kimeneten csak **C=0** szintnél jelenik meg, amikor már a bemeneti információközlő vezérlés az első tárolóra hatástalan.

Az ***ms flip-flop szimbólumában*** a kétütemű billentést a **TT** (kettős T) jelöli.

Az elemzett megoldás csak elvileg ad helyes működést. Ha az ellenütemű vezérlést biztosító ***inverter késleltetése nagyobb***, mint a ***flip-flop billenési ideje***, akkor a master még a slave vezérlésének tiltása előtt felveheti az új állapotot, és az át is íródhat a slave -be. Ekkor a billentés közvetlen lesz, vagyis egy-ütemű. Ez hibás működést eredményezhet. A tényleges áramköri megoldásoknál ezért a ***két flip-flop közötti csatolás*** letiltása ***hamarabb*** kell bekövetkezzen, mint a ***bemeneti*** kapuzás ***engedélyezése***. Egyik megoldást a két komparálási szintű kapuzás biztosítja. A megoldás lényege, hogy a billentő (kapuzó) más értékénél – $\mathbf{U_{kb}}$ - nyitnak a bemeneti kapuk, és más értéknél – $\mathbf{U_{kc}}$ - a két flip-flop közötti csatoló kapuk

A kettős komparálás fogalmát a billentő-jel időbeli változása alapján elemezzük. A 69.ábra a Cp bemenetre jutó billentő impulzus időbeli változását mutatja.

A $\mathbf{t_1}$ időpontban kezdődik a billentő-jel felfutó éle, és amikor a $\mathbf{t_2}$ időpontban eléri az $\mathbf{U_{kc}}$ értéket, akkor lezár a master és a slave flip-flop -ok közötti csatolás, de még zárt a bemeneti csatolás is. Egyik tároló tartalma sem változik. A billentő jel további növekedésekor – a $\mathbf{t_3}$ időpontban - eléri a bemeneti kapuk komparálási szintjét – $\mathbf{U_{kb}}$ –t - , és ezután az R és S bemenetekre jutó jel értékétől függő információ íródik a master tárolóba. A slave tároló még tartja az előző értéket, tehát – a bemenetekre jutó információtól függetlenül – a ***„régi”*** érték van a kimeneten is. A billentő jel csökkenésekor a $\mathbf{t_4}$ időpontban lezárnak a bemeneti kapuk, és a $\mathbf{t_5}$ időpontban a flip-flop -ok közötti csatolás nyit ki. Ekkor kerül a kimenetre az ***„új”*** érték. A leírt működés biztosítja azt, hogy a tároló kimeneti értékét vissza lehessen vezetni a bemenetre is.



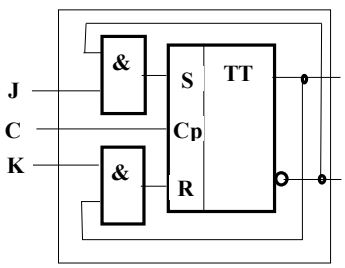

69. ábra

- **Közbenső tárolós JK flip-flop**

A **JK** típusú flip-flop - amelynek elvi működési feltételét a korábbiakban már elemeztük - csak ***kétütemű***, vagy ***élvezérelt*** billentéssel alakítható ki. A flip-flop állapot egyenlete az alábbi:

$$Q_{n+1} = J_n \overline{Q}_n + \overline{K}_n Q_n$$

A JK flip-flop - közbenső tárolós RS flip-flop -ból a 8. ábra szerint épül fel. A ***bemeneti*** vezérlő ***jelek kapuzása*** a ***kimeneti jelekkel*** biztosítja a kívánt működést. Amikor a flip-flop 1-t tárol (Q = 1) akkor csak a K bemenetre jutó vezérlés eredményez állapotváltozást, ill. 0 tárolását követően (Q = 0) a J bemenetre jutó vezérlés a hatásos. Ez a kapuzás (70. ábra) egyúttal engedélyezi a **J és K *bemenetek együttes vezérlését*** is. Ekkor ugyanis a flip-flop előző állapota határozza meg a billentő-jel hatására bekövetkező állapotváltozást.

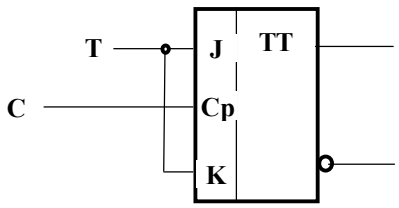



70. ábra

Az előzőekben elemzett JK flip-flop –ból **T *típusú tároló*** olyan módon alakítható ki, hogy a két vezérlő bemenetet (J és K) összekötjük s ez lesz a T vezérlő bemenet. Az állapotegyenlet - 1 szintű aktív vezérlésnél - a következő:

$$Q_{n+1} = T_n \overline{Q}_n + \overline{T}_n Q_n$$

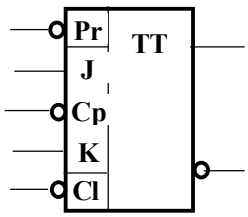
A tárolóba információt csak a T ***vezérlő bemenet 1 szintjénél*** lehet beírni. Az állapot-változást a T = 0 vezérlés letiltja. A T flip-flop -ot elsősorban számláló áramkörök kialakítására használják. Integrált áramköri kialakításban ezt a flip-flop változatot önállóan nem gyártják, miután a JK típusból külső kötéssel kialakítható. A 69. ábra a T flip-flop logikai felépítését és szimbolikus jelét ábrázolja.




71. ábra

- **Közbenső tárolós flip-flop -ok aszinkron billentése**

Az integrált áramköri közbenső tárolós - master-slave - flip-flop -oknak ***aszinkron törlő*** és ***beíró*** bemenetei is vannak. Az aszinkron statikus vezérlés együtemű. Ez azt jelenti, hogy a vezérlőjel - a flip-flop mindkét tárolóját - egyidejűleg billenti a kívánt állapotba. A 72. ábra közbenső tárolós **JK *preset flip-flop*** szimbolikus jelét mutatja. Az aszinkron vezérlő bemenetek, a **Cl** (Clear) **törlő** és a **Pr** (Preset) ***beíró*** bemenet.



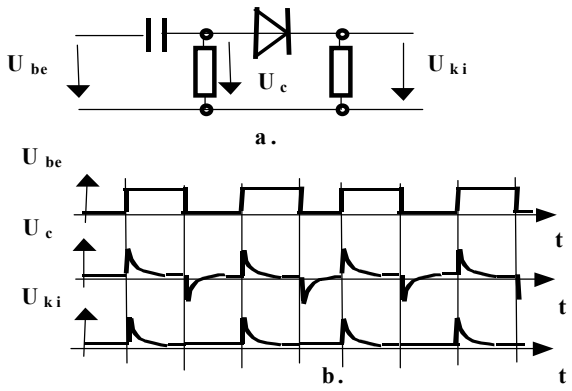

72. ábra

A jelölésben a bemeneti mező középső részéhez csatlakoznak a kétütemű vezérlésű ms flip-flop bemeneti jelei. A **Cp** billentő bemeneten lévő invertáló jel (karika) azt jelzi, hogy az új érték a kimeneteken a ***billentő jel*** **0** szintjénél jelenik meg. A **Pr** aszinkron beíró, illetve Cl törlő bemenetek ***aktív*** szintje **0**. A két utóbbi bemenet szerint a tároló **RS *típusú*** együtemű flip-flop.

- **Dinamikus billentésű flip-flop -ok**

Az eddigiekben elemzett flip-flop -ok közös jellemzője a statikus billentés. A tárolók másik nagy csoportját alkotják a ***dinamikus*** billentésű (***élvezérelt***) áramköri megoldások. A továbbiakban külön elemezzük a dinamikus billentés diszkrét ill. integrált áramköri megoldásait.

Az ***élvezérlés*** diszkrét elemekkel un ***trigger - áramkörrel*** alakítható ki. A trigger áramkör kimenetén csak akkor jelenik meg jel, ha bemenetén logikai szintváltás van. A trigger áramkör vagy más néven dinamikus csatolókapu egyik legegyszerűbb változata a 73. ábra szerinti.

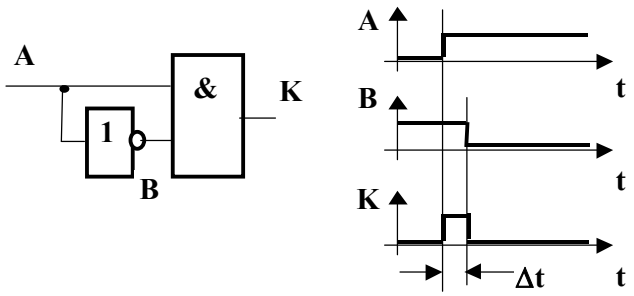



73. ábra

Az a. ábrán a kapcsolási vázlat, míg a b. ábrán a jellegzetes feszültségalakokat szemlélteti. Ha a kapu bemenetére kapcsolt $\mathbf{U_{be}}$ feszültség négyszöghullám, akkor a belső ponton csak a bemeneti ***szintváltáskor*** mérhető feszültségugrás, mégpedig a szintváltás irányának megfelelő polaritású. A diódán csak a ***pozitív*** feszültség-változás

hajt át áramot, ezért a kimeneti ponton a felfutó élekkor lesz jel. A diódát fordítva kötve, a negatív éleknél lesz a kimeneten jelváltás.

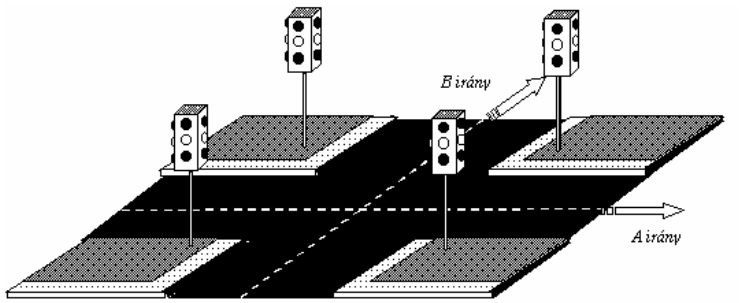
Az élvezérlés egyik megoldásánál a billentő impulzus felfutó élénél mesterségesen létrehozott ***hazárd*** vezérli az állapotváltozást. Ennek elve, hogy ha a logikai ÉS kapu két bemenetén a jelek ellenkező értelemben változóak és az 1 - 0 átmenet késleltetett, akkor a kimeneten a késleltetéssel megegyező idejű – tű-impulzus (hazárd) jön létre. A kapcsolást és az időviszonyokat a 74. ábra szemlélteti. A **Δt** jelű elem késleltetési ideje határozza meg a tű-impulzus szélességét.




74. ábra

***12.Példa***

Kétirányú útkereszteződés forgalomirányító lámpáinak – 75. ábra -vezérlése




75. ábra

***A feladat:*** olyan vezérlőáramkőr kialakítása, amelynek a bemenetére jutó jel ***váltja*** az ***állapotokat.*** A hat logikai – irányonként 3-3 – ***kimenet vezérli*** a megfelelő ***lámpák***

teljesítményillesztő egységét. A lámpák vzérlésének sorrendje a KRESZ szabályainak feleljen meg.

**A tervezés lépései:**

- ***Be-, kimenetek deklarálása:***

| | | |
|---|---|---|
| ***C*** | bemenet | a jel ***1 – 0*** átmenete váltja az állapotokat, ezt jelöljük az $\mathbf{X_0}$ kombinációval, míg a nem hatásosat $\mathbf{X_1}$ – el. |
| $P_B$ | kimenet | ***B*** irány ***piros*** |
| $S_B$ | kimenet | ***B*** irány ***sárga*** |
| $Z_B$ | kimenet | ***B*** irány ***zöld*** |
| $P_A$ | kimenet | ***A*** irány ***piros*** |
| $S_A$ | kimenet | ***A*** irány ***sárga*** |
| $Z_A$ | kimenet | ***A*** irány ***zöld*** |

- ***A szükséges kimeneti variációk meghatározása***

A ***hat*** kimeneti jelek lehetséges ***64 variációja*** (Zi) lehetséges, viszont a vezérléshez ***csak*** a következő ***négy*** szükséges:

| Zi | $P_B$ | $S_B$ | $Z_B$ | $P_A$ | $S_A$ | $Z_A$ | Funkció |
|---|---|---|---|---|---|---|---|
| $\mathbf{Z_{12}}$ | 0 | 0 | 1 | 1 | 0 | 0 | **B** irány ***zöld***, **A** irány ***piros*** |
| $\mathbf{Z_{24}}$ | 0 | 1 | 0 | 1 | 1 | 0 | **B** irány ***sárga***, **A** irány ***piros-sárga*** |
| $\mathbf{Z_{33}}$ | 1 | 0 | 0 | 0 | 0 | 1 | **B** irány ***piros***, **A** irány ***zöld*** |
| $\mathbf{Z_{50}}$ | 1 | 1 | 0 | 0 | 1 | 0 | **B** irány ***piros-sárga***, **A** irány ***sárga*** |

***Megjegyzés****:* A kimeneti variációk indexét a változók balról –jobbra történő bináris súlyozása alapján számítottuk ki.

- ***A kimeneti variáció szekvenciái:***

Az állapotváltások csak egy kötött sorrendben követhetik egymást, tehát az előállítandó szekvencia:

**…… $Z_{12}$ - $Z_{24}$ - $Z_{33}$ - $Z_{50}$ - $Z_{12}$ …..**

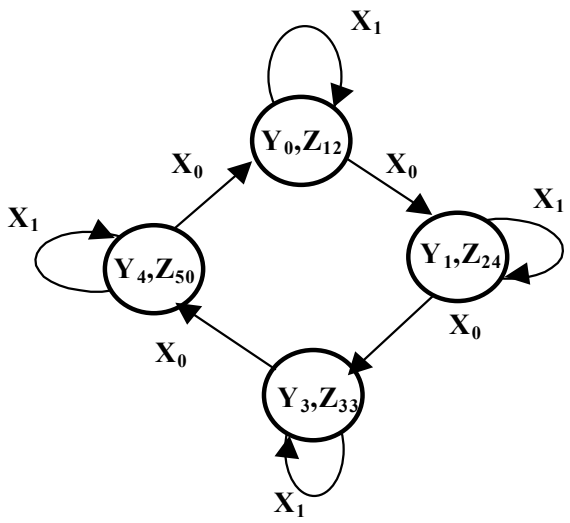
- ***A szükséges állapotok száma***

Az állapotok számát – ***m*** – a szükséges kimeneti kombinációk száma határozza meg, amely a jelen feladatban ***négy***, melyeket jelöljük ***$Y_0$, $Y_1$, $Y_2$, $Y_3$*** –al.

- ***Az alkalmazandó rendszertechnikai felépítés***

A feladatban ***hat*** kimeneten kell jeleket kiadni, viszont a belső állapotok száma csak ***négy***. Az állapotok tárolásához elégséges ***két*** flip-flop, amelyek maximálisan csak négy kimenetet adhatnak. Szükséges tehát ***kimeneti kombinációs*** hálózat alkalmazása. Célszerű a ***Moore-modell*** szerint ***szinkron sorrendi*** hálózattal megvalósítani a feladatot.

- ***Állapotgráf felrajzolása***




- ***Állapottáblázat felrajzolása***

| | Bemeneti komb. | |
|---|---|---|
| Állapot | $X_0$ | $X_1$ |
| $Y_0^*$ | $Y_1$, $Z_{12}$ | $Y_0$, $Z_{12}$ |
| $Y_1^*$ | $Y_2$, $Z_{24}$ | $Y_1$, $Z_{24}$ |
| $Y_2^*$ | $Y_3$, $Z_{33}$ | $Y_2$, $Z_{33}$ |
| $Y_3^*$ | $Y_0$, $Z_{50}$ | $Y_3$, $Z_{50}$ |

***Megjegyzés:*** A * indexet az előző állapot jelzésére használtuk.

- ***A feladat megoldásához alkalmazott flip-flop kiválasztása***

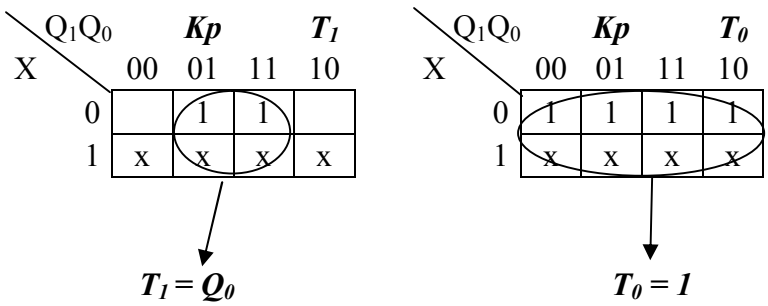
A sorrendi hálózatokban csak elvezérelt, vagy kétütemű vezérlésű tárolók használhatók. Alkalmazzunk 1 szinttel engedélyezhető ***T*** típusú ***master-slave*** (ms) flip-flop -ot.

- ***A kódolt vezérlési táblázat megrajzolása, és a vezérlési függvények meghatározása***

Az állapotok tárolásához szükséges két flip-flop Qi kimenetein megjelenő jelkombinációkhoz rendeljük az állapotokat, és a Ti bemeneteken tiltjuk, vagy engedélyezzük a tároló billenését, amelyet a C jel 0-1 átmenete vezérel.

| | $Y_i^*$ | | $X_0$ | | $X_1$ | |
|---|---|---|---|---|---|---|
| | $Q_1$ | $Q_0$ | $T_1$ | $T_0$ | $T_1$ | $T_0$ |
| $Y_0^*$ | 0 | 0 | 0 | 1 | x | x |
| $Y_1^*$ | 0 | 1 | 1 | 1 | x | x |
| $Y_2^*$ | 1 | 0 | 0 | 1 | x | x |
| $Y_3^*$ | 1 | 1 | 1 | 1 | x | x |

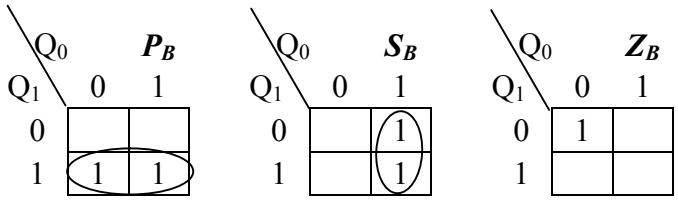
Karnaugh diagramok



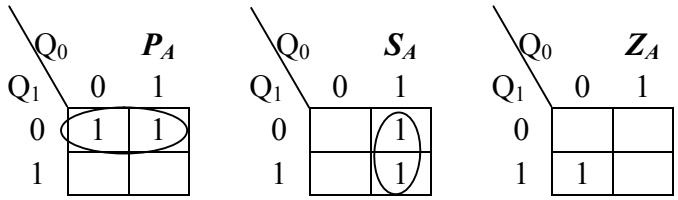

- ***A kódolt kimeneti táblázat megrajzolása, és a kimenetek függvényeinek meghatározása***

| | $Q_1$ | $Q_0$ | $P_B$ | $S_B$ | $Z_B$ | $P_A$ | $S_A$ | $Z_A$ |
|---|---|---|---|---|---|---|---|---|
| $Y_0^*$ | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| $Y_1^*$ | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| $Y_2^*$ | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| $Y_3^*$ | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

A kimenetek Kp diagramjai:

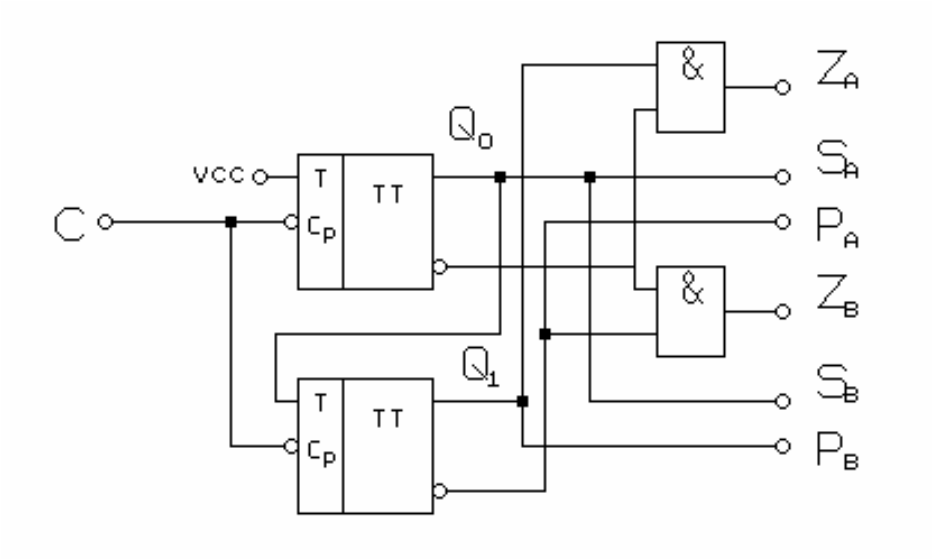



$P_B = Q_1$ $S_B = Q_0$ $Z_B = \overline{Q_1}\,\overline{Q_0}$




$P_A = \overline{Q_1}$ $S_A = Q_0$ $Z_A = Q_1\,\overline{Q_0}$

- ***Logikai vázlat***




76. ábra

## 2.6. Funkcionális sorrendi hálózatok.

A sorrendi logikai feladatok nagytöbbségében szükséges valamilyen változó (esemény) bekövetkeztének ***számosságát*** meghatározni. Ugyancsak gyakori az a feladat, hogy egy ***adatvonalon*** – szabályos időközönként – érkező ***n*** darab logikai érték ***tárolása***.

A vázolt feladatok gyakorisága indokolja, hogy funkcionális egységenként állítsák elő – gyártsák le – ezeket a hálózatokat. Két leggyakoribb hálózat a:

- ***számláló***, és a
- ***léptetőregiszter***.

### ⇒ Számlálók

A különböző irányítási, adatfeldolgozási és mérési feladatokban gyakran szereplő részfeladat a legkülönfélébb ***jelek***, tágabb értelemben ***események számlálása***. A funkciót ellátó áramköröket nevezzük ***számlálóknak***.

A számláláskor alapvetően két műveletet kell végezni, úgymint **tárolni** az eddig már bekövetkezett események ***számát***, majd az újabb esemény hatására - a kiválasztott számlálási iránynak megfelelően - az eddigi értéket ***növelni*** (inkrementálás), ill. ***csökkenteni*** 1-gyel (dekrementálás).

A hozzáadás, ill. a levonás - mint ahogy ezt már megismertük - kombinációs logikai feladatként is kezelhető. Az előbbiek alapján tehát a számlálás sorrendi hálózattal megvalósítható.

Az egyirányú számlálók olyan speciális sorrendi hálózatok, amelyeknek állapotai csak egy meghatározott sorrendben követik egymást, s ez az állapotsorozat ciklikusan ismétlődik. A számlálandó jel fogadására a számlálónak egyetlen bemenete van. A kimenetek és a szükséges ***állapottárolók*** számát a ***kapacitás*** szabja meg. A ***kapacitás*** azt a legnagyobb számot jelenti, amellyel egy ciklus befejeződik. Ezt a továbbiakban **k-**val jelöljük. A számsorozat így **0, 1, 2. . . k** értékekből áll. A számlálónak tehát

$$\mathbf{m = k + 1}$$

különböző értéket kell megkülönböztetnie. Az **m** - et nevezzük a számláló ***modulusának***.

### ⇒ A számlálók csoportosítása

A számlálókat többféle szempont alapján csoportosíthatjuk, úgymint:

- működési ***mód***,
- számlálási ***irány***,
- az információ-tárolás ***kódja***,
- ***áramköri*** megvalósítás

szerint végezhetjük el a felosztást.

A ***működési mód*** szerint

- ***aszinkron*** és
- ***szinkron***

számlálókat különböztetünk meg. Az ***aszinkron*** működés lényege, hogy a számlálandó jel csak elindítja a szükséges állapotváltozási sorozatot. A továbbiakban az egyes ***flip - flop*** - ok ***billentik*** egymást.

A ***szinkron*** működés alapja, hogy két számlálandó jel között történik a következő állapotba billentés előkészítése, s a flip-flop - okat **a *számlálandó jel billenti***.

A számlálás **iránya** szerint

- ***előre*** (UP - fel),
- ***hátra*** (DOWN - le),
- ***előre-hátra*** (reverzibilis - UP/DOWN)

működések lehetnek. A ***reverzibilis*** számlálóknál a számlálási irányt külső vezérlőjel változtatja meg.

A számtartalmat (információt) ***tárolhat****juk*

- ***bináris***
- ***BCD és***
- ***egyéb***

kódokban. Ezt a számláló megnevezésében jelöljük, pl. szinkron bináris előre számláló.

Az áramköri megvalósításnál

- ***diszkrét*** elemes és az
- ***integrált*** áramköri

számláló megkülönböztetés elsődlegesen formai és nem a működés lényegére utal.

A számlálókat - elsődlegesen az integrált áramköri kivitelben - ki szokták még egészíteni járulékos funkciókkal. Ilyen kiegészítés, hogy az egyes flip-flop - ok - a számlálási funkciótól függetlenül is - külső jellel beállíthatók 0 vagy 1 állapotba. Ezek az elő-beírású vagy PRESET számlálók. A másik gyakori megoldás, hogy a számlálás végszámát jelző áramkör is a számláló tartozéka (egyazon tokban van).

## ⇒ Bináris számlálók

Leggyakrabban használjuk a 2-es számrendszerben számláló bináris számlálók különböző változatait. Ebben a pontban részletesen foglalkozunk a számlálók e csoportjának működésével, és logikai tervezésével.

### ▪ *Bináris számlálók logikai tervezése*

Egy **k** kapacitású ***bináris számláló*** logikai tervezésének lépései:

- a tárolók ***számának*** meghatározása,
- az ***állapottáblázat*** felvétele,
- az alkalmazott ***flip-flop típus*** kiválasztása,
- a ***kódolt állapottáblázat*** felírása,

- a ***vezérlő függvények*** meghatározása,
- a ***logikai vázlat*** megrajzolása.

A logikai tervezést egy **k = 7** kapacitású ***szinkron*** **bináris** számláló példáján ismertetjük.

A számlálónak **m = k + 1 = 8** állapotot kell megkülönböztetnie. A szükséges állapottárolók száma tehát ***három*** ($2^3$=8). A számláló kimenetein - az egyes ütemekben - a 0 - 7 értékek bináris kódját kell kapjuk. Ezek az értékek három bináris helyértékkel kifejezhetők, tehát az állapottárolók és a hálózat kimenetei ugyanazok is lehetnek.

A számláló állapottáblázata (77. ábra) három oszlopot, és nyolc sort tartalmaz. Miután egyetlen bemeneti jel van (C), két bemeneti kombináció lehetséges**,** $\mathbf{X_0}$**,** $\mathbf{X_1}$. Válasszuk $\mathbf{X_0}$ -hoz, amikor nincs számlálandó jel (***C = 0***), az $\mathbf{X_1}$ –hez, pedig, amikor van (***C = 0***). A nyolc kimeneti kombináció ($\mathbf{Z_0}$ **–** $\mathbf{Z_7}$) pedig a bináris számértékeknek megfelelő állapotok. A megállapodás szerint * felső index a ***jelenlegi***, míg anélkül a ***következő*** állapotot jelenti.

| ***Kimenetek*** | ***Bemenetek*** | |
|---|---|---|
| $Z^*$ | $X_0$ | $X_1$ |
| $Z_0^*$ | $Z_0$ | $Z_1$ |
| $Z_1^*$ | $Z_1$ | $Z_2$ |
| $Z_2^*$ | $Z_2$ | $Z_3$ |
| $Z_3^*$ | $Z_3$ | $Z_4$ |
| $Z_4^*$ | $Z_4$ | $Z_5$ |
| $Z_5^*$ | $Z_5$ | $Z_6$ |
| $Z_6^*$ | $Z_6$ | $Z_7$ |
| $Z_7^*$ | $Z_7$ | $Z_0$ |

77. ábra

Következő lépésként a kódolt állapottáblázatot kell felírni (78.ábra).

- A ***bemeneti*** kódolást már meghatároztuk, amely szerint

$\mathbf{X_0}$**-nál C = 0,** $\mathbf{X_1}$**-nél C= 1.**

- A ***kimeneti*** kombináció-sorozat, pedig a $z_0 = 2^0$, $z_1 = 2^1$, $z_2 = 2^2$ kimeneteken megjelenő ***bináris számsor*** (a helyértékek a jelölés szerintiek).

- Az ***állapotvezérlő*** jelek kombinációi ($V_0 – V_7$) billentik a flip-flop -okat a soron következő állapotkombinációba. Változás csak a C=1 értéknél van, és ekkor értékük egyértelműen meghatározza a következő állapotot. Az egyes vezérlőjelek ($v_i$) flip-flop –okat billentik. A táblázatban a következő jelölést használtuk: jel = 1, ha kell változtatni a tároló értékét, és 0 ha nem.

*Megjegyzés:* a különböző tárolóknál más, és más lehet a vezérlőjel értéke.

| ***Kimeneti kombinációk*** | | | ***Bemeneti-*(C)*, és vezérlőjel*** ($V_i$) ***kombinációk*** | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | ***0*** | | | ***1*** | | |
| $z_2$ | $z_1$ | $z_0$ | $v_2$ | $v_1$ | $v_0$ | $v_2$ | $v_1$ | $v_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |

78. ábra

A táblázatból megállapítható, hogy a számlálók vezérlőtáblázatából elhagyhatók a C=0 – állapotváltozás nincs – oszlopok, és ezzel egyszerűbb táblázatot kapunk. A következő példáknál ezt követjük.

### ⇒ Szinkron bináris számlálók

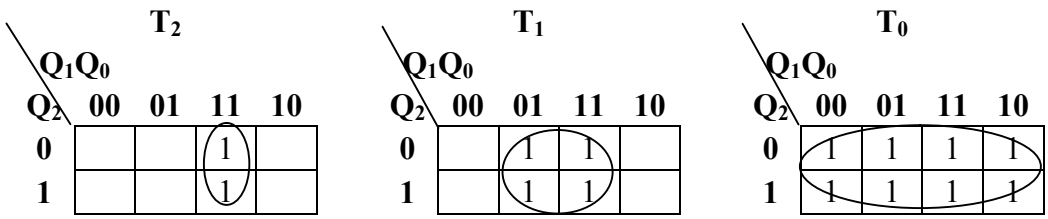
Számlálók ***él-vezérelt*** vagy ***közbenső tárolós*** (master-slave) flip-flop –ból építenek, mivel ezeknél lehet a billentés feltételébe a kimenetek jeleit visszacsatolni. Ugyanakkor a számlálandó jel mindegyik tároló billentő bemenetére vezethető, vagyis kettéválasztottuk az ***előkészítő***, és a ***billentő*** jeleket. Ez a számlálás ***szinkron*** üzemű megoldása.

A számlálót alakítsuk ki **T**-típusú **ms** flip-flop –al. Ekkor az egyes tárolók T bemeneteire kell csatlakoztatni az állapotvezérlő jeleket. Ekkor az állapotváltozók kódolását abból a feltételből írjuk fel, hogy **1** szint ***engedélyezi*** a flip-flop billentését, **0** szint, pedig ***nem***. Az előbbi megállapodások szerinti kódolt állapottáblázat látható az 78.ábrán. ( A táblázatban az állapotváltozókat a ***T0, T1, T2*** –al, a kimeneteket, pedig ***Q0, Q1, Q2*** –al - a szakirodalomban legtöbbször használt betűkkel - jelöltük.)

| $Q_2$ | $Q_1$ | $Q_0$ | $T_2$ | $T_1$ | $T_0$ |
|---|---|---|---|---|---|
| **0** | **0** | **0** | **0** | **0** | **1** |
| **0** | **0** | **1** | **0** | **1** | **1** |
| **0** | **1** | **0** | **0** | **0** | **1** |
| **0** | **1** | **1** | **1** | **1** | **1** |
| **1** | **0** | **0** | **0** | **0** | **1** |
| **1** | **0** | **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** | **0** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** |

79. ábra

A ***kódolt állapottáblázatból*** az egyes engedélyező bemenetekre külön-külön felírhatunk egy-egy ***Karnaugh - diagramot***. Ezek segítségével aztán a legegyszerűbb logikai függvények meghatározhatók. Ezzel tulajdonképpen a bemeneti kombinációs hálózat logikai felépítését is meghatározzuk.




80. ábra

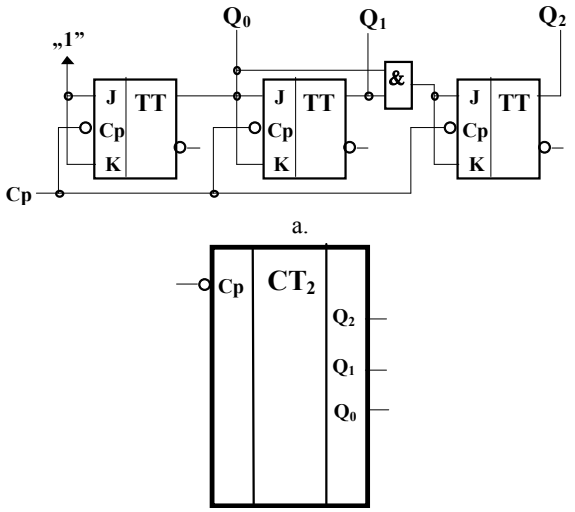
A 80.ábrán láthatók a $T_0$, a $T_1$ és a $T_2$ változókra felirt K diagramok, amelyek alapján az egyes vezérlőfüggvények:

$$T_0 = 1,$$

$$T_1 = Q_0,$$

$$T_2 = Q_0Q_1$$

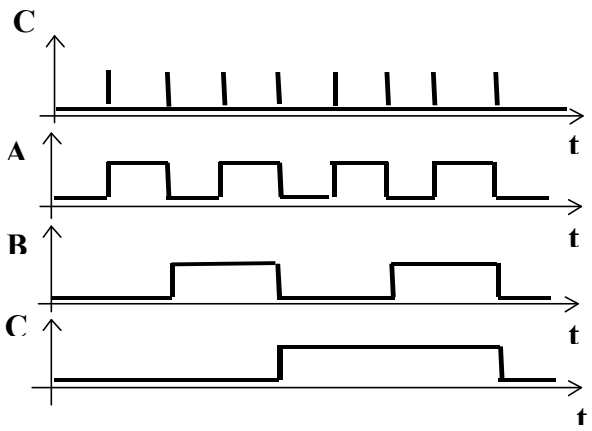
A számláló logikai vázlata az 81.a.ábrán, a számláló szimbolikus jele, pedig a b. ábrán láthatók. A CT (COUNTER) jelölés melletti index a számrendszerre utal, pl.$CT_2$ bináris számláló.




b.

81. ábra

A kimenetek, és a számlálandó jel időfüggvényeit mutatja a 82. ábra. Az időfüggvények felett feltüntettük az egyes ütemek kimeneti állapot - kombinációit. Ezek sorozata - a kitűzött célnak megfelelően - a növekvő bináris számsort adják.




82. ábra

A számláló kapacitását további flip-flop -okkal növelni lehet. Ezek vezérlőfüggvényeit az előzőekhez hasonlóan határozhatjuk meg. Ezt most mellőzve, az időfüggvényekből is

következtethetünk a törvényszerűségre. A soron következő flip-flop mindig olyankor vált állapotot, amikor minden előző flip-flop nál 1 - 0 állapotátmenet van. Ennek alapján a az **i.** flip-flop vezérlőfüggvényének általános alakja:

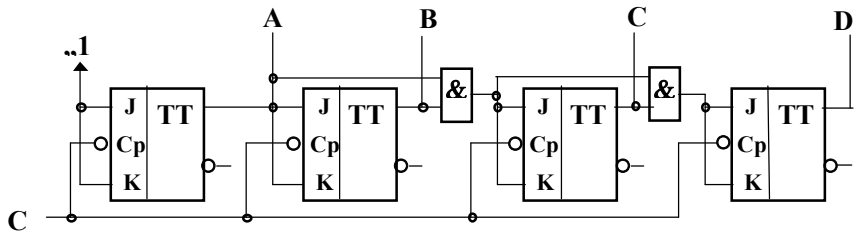
$$\mathbf{T_i = Q_0 Q_1\, Q_2\, .\, .\, .\, \; Q_{i-1}}$$

A függvény alapján megállapíthatjuk, hogy a kapacitásbővítéshez - az újabb flip-flop mellett - mindig 1-gyel több bemenetű ÉS kapu kell. Ezt a megoldást nevezzük **párhuzamos átvitelűnek**. A megnevezés arra utal, hogy minden egyes előkészítő bemenetre egyidejűleg (párhuzamos csatornákon) jutnak a megfelelő kimenetek értékei. Ezáltal egy kapunyi jelkésleltetés múlva az újabb billentés előkészítése befejeződik.

Az összefüggések átalakíthatók a következők szerint:

$$\mathbf{T_0 = 1,}$$
$$T_1 = Q_0 = \mathbf{T_0\, Q_0}$$
$$T_2 = Q_0 Q_1 = \mathbf{T_1\, Q_1}$$
$$\vdots$$
$$T_i = Q_0 Q_1\, Q_2\, .\, .\, .\, \; Q_{i-1} = \mathbf{T_{i-1}\, Q_{i-1}}$$

Az átalakított vezérlőfüggvények szerint kialakított m = 16 modulusú bináris számláló logikai vázlatát mutatja a 83. ábra. A kimenetek elnevezésénél – a számlálóknál használt – **A,B,C,D** jelölést rajzoltuk.



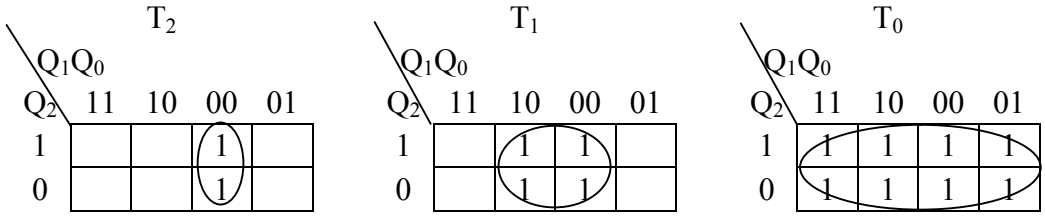

83. ábra

Ebben a megoldásban egységesen két bemenetű ÉS kapuk állítják elő a vezérlőjeleket. Az áramköri egyszerűsítés ára, hogy a számláló határfrekvenciája csökken, mert a legutolsó flip-flop előkészítő bemenetére a jel két sorba kötött kapun keresztül jut. Ezt az áramköri megoldást nevezzük ***soros átvitelűnek***. További kapacitásbővítés újabb kapukat, s így késleltetéseket iktat be.

Hárombites bináris ***hátraszámláló*** kódolt állapottáblázatát láthatjuk az 84.ábrán. Ezt is T típusú flip-flop -okból alakítjuk ki.

| $Q_2$ | $Q_1$ | $Q_0$ | $T_2$ | $T_1$ | $T_0$ |
|---|---|---|---|---|---|
| **1** | **1** | **1** | **0** | **0** | **1** |
| **1** | **1** | **0** | **0** | **1** | **1** |
| **1** | **0** | **1** | **0** | **0** | **1** |
| **1** | **0** | **0** | **1** | **1** | **1** |
| **0** | **1** | **1** | **0** | **0** | **1** |
| **0** | **1** | **0** | **0** | **1** | **1** |
| **0** | **0** | **1** | **0** | **0** | **1** |
| **0** | **0** | **0** | **1** | **1** | **1** |

84. ábra




85. ábra

Az előkészítő bemenetek vezérlőfüggvényei a Kp diagramok alapján (85.ábra) a következők:

$$\mathbf{T_0 = 1}$$
$$\mathbf{T_1 = \overline{Q}_0}$$
$$\mathbf{T_2 = \overline{Q}_0 \overline{Q}_1}$$

A vezérlőfüggvény általános alakja:

$$\mathbf{T_i = \overline{Q}_0 \overline{Q}_1 \cdots \overline{Q}_{i-1}}$$

lesz. E függvények közvetlen megvalósításával párhuzamos átvitelű ***bináris hátraszámlálót*** kapunk. Logikai átalakítások után – az előreszámlálóhoz hasonlóan - a soros átvitelű bináris hátraszámláló is kialakítható.

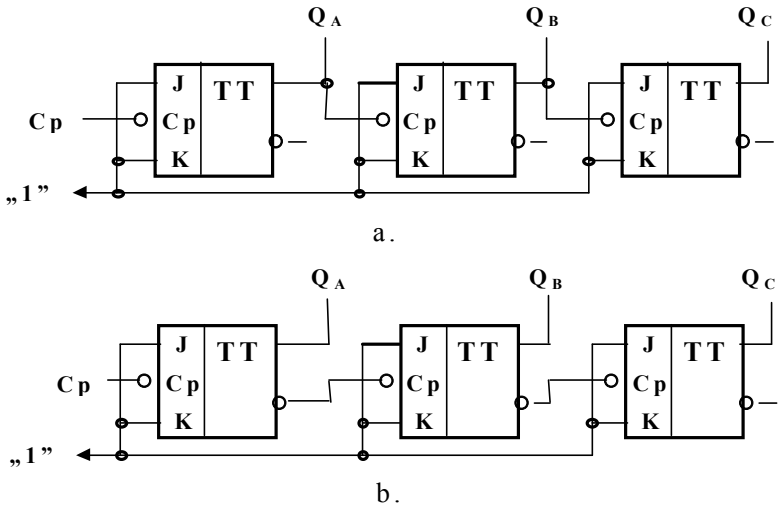
Az előre-, ill. hátraszámláló vezérlőfüggvényeinek ismeretében felírhatjuk a ***reverzibilis*** bináris számláló függvényeit is. A számlálási irányt a **P** külső parancs vezérli (P = 0 előreszámlálás, P = 1 hátraszámlálás

- **Aszinkron bináris számlálók**

Az aszinkron működésű számlálóknál a számlálandó jel csak ***elindítja*** a soron következő ***állapotváltozást***, de az egyes flip-flop -ok ***egymást*** billentik. A bináris előreszámláló kimeneteinek időfüggvényénél (82. ábra) láttuk, hogy mindegyik flip-flop a megelőző tároló 1 - 0 átmeneténél kell, billenjen. Így **1 - 0** átmenetre billenő **T** flip-flop -ok 85.a.ábra szerinti kapcsolásával bináris előreszámlálót kapunk.

A bináris hátraszámlálónál az előző flip-flop -ok a megelőző tároló 0 - 1 állapotváltozásánál kell billenjenek. Ugyancsak 1 - 0 átmenetre billenő T flip-flop -okból kialakított hátraszámláló logikai vázlata a 86 b. ábrán látható.

Az aszinkron számlálók nagyon egyszerű felépítése mellett hátránya a ***kisebb határfrekvencia***. Ez abból adódik, hogy az új stabil állapot csak az egymást követő billenések, befejezése után áll be. Ugyancsak hátrány, hogy az átmeneti időszakban nem ***kívánt kombinációk*** is előfordulnak a kimeneteken. Ez a csatlakozó hálózatnál zavart okozhat.



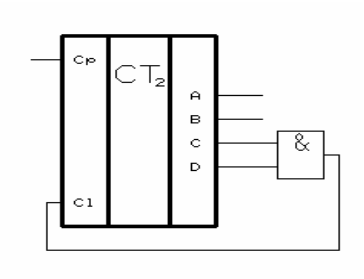

86. ábra

## ⇒ BCD kódolású számlálók

A különböző digitális mérő- és egyéb adatfel-dolgozó berendezésekben az adatok kijelzése, ill. bevitele rendszerint 10-es számrendszerben történik. Ezért a belső adat-forgalomnál, így a számlálásnál is esetenként célszerű a használata. Ezt teszik lehetővé a különböző ***BCD*** kódolású számlálók.

A tananyagban csupán a ***BCD 8421*** súlyozású számlálókkal foglalkozunk. A megismert tervezési módszer alapján azonban a további BCD kódolású számlálók is megter-vezhetők.

A ***8 4 2 1*** súlyozású BCD számláló működése a 4 bites bináris számlálótól abban tér el, hogy a tizedik impulzus hatására a kezdő ***0 0 0 0*** állapotba tér vissza a számláló.




87. ábra

Ennek a ***modulus csökkentésnek*** egy lehetséges megoldása, hogy egy 4 bites bináris számlálót - amelynek aszinkron törlő bemenete is van - olyan logikai hálózattal egészítünk ki, ami az **1 0 1 0** (decimális 10) állapot megjelenésekor minden flip-flop -ot töröl és ezzel **0 0 0 0** állapot áll be. Ezt a megoldást szemlélteti a 87. ábra.

E megoldás hátránya, hogy a 11. állapot egy rövid ideig - a kapu késleltetés és a billenési idő összegéig - bekövetkezik. Ez járulékos hibát okoz, különösen frek-venciaosztóként való alkalmazáskor.

A logikai tervezés során, a bináris számlálónál megismert induló fázisokat - az általános állapot táblázat és vezérlőfüggvényeinek felírását - elhagyjuk. A tervezést a kiválasztott flip-flop típusra érvényes kódolt állapottáblázat felírásával kezdjük.

- **Szinkron BCD számlálók**

A szinkron BCD számlálók felépítését és működését JK típusú ms flip-flop -ok alkalmazásával ismertetjük.

Először röviden összefoglaljuk a JK flip-flop vezérlésének feltételeit. Ezt mutatja a 88. ábrán levő táblázat. A $\mathbf{Q_i}$ a billentés előtti, $\mathbf{Q_{i+1}}$ pedig a billentő impulzus hatására bekövetkező új állapotot jelzi. Az **x** közömbös értéket jelent, vagyis 0 vagy 1 is lehet.

| $Q_i$ | $Q_{i+1}$ | J | K |
|---|---|---|---|
| 0 | 0 | 0 | x |
| 0 | 1 | 1 | x |
| 1 | 0 | x | 1 |
| 1 | 1 | x | 0 |

88. ábra

A BCD 8 4 2 1 súlyozású előreszámláló kódolt állapottáblázata látható a 89. ábrán. A kimeneteket - a nemzetközileg egységesen használt - A, B, C, D betűkkel jelöltük, ahol az $\mathbf{A=2^0}$ helyérték.

| D | C | B | A | $J_D$ | $K_D$ | $J_C$ | $K_C$ | $J_B$ | $K_B$ | $J_A$ | $K_A$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | x | 0 | x | 0 | x | 1 | x |
| 0 | 0 | 0 | 1 | 0 | x | 0 | x | 1 | x | x | 1 |
| 0 | 0 | 1 | 0 | 0 | x | 0 | x | x | 0 | 1 | x |
| 0 | 0 | 1 | 1 | 0 | x | 1 | x | x | 1 | x | 1 |
| 0 | 1 | 0 | 0 | 0 | x | x | 0 | 0 | x | 1 | x |
| 0 | 1 | 0 | 1 | 0 | x | x | 0 | 1 | x | x | 1 |
| 0 | 1 | 1 | 0 | 0 | x | x | 0 | x | 0 | 1 | x |
| 0 | 1 | 1 | 1 | 1 | x | X | 1 | x | 1 | x | 1 |
| 1 | 0 | 0 | 0 | x | 0 | 0 | x | 0 | x | 1 | x |
| 1 | 0 | 0 | 1 | x | 1 | 0 | x | 0 | x | x | 1 |

89. ábra

Az egyes flip-flop -ok J és K bemeneteinek vezérlési feltételeit 90. ábrán levő Kp diagramok segítségével határozzuk meg.

**$J_A$**

| BA \ DC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | x | 1 |
| 01 | x | x | x | x |
| 11 | x | x | x | x |
| 10 | 1 | 1 | x | x |

**$K_A$**

| BA \ DC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | x | x | x | x |
| 01 | 1 | 1 | x | 1 |
| 11 | 1 | 1 | x | x |
| 10 | x | x | x | x |

**$J_B$**

| BA \ DC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | x | 0 |
| 01 | 1 | 1 | x | 0 |
| 11 | x | x | x | x |
| 10 | x | x | x | x |

**$K_B$**

| BA \ DC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | x | x | x | x |
| 01 | x | x | x | x |
| 11 | 1 | 1 | x | x |
| 10 | 0 | 0 | x | x |

**$J_C$**

| BA \ DC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | x | x | 0 |
| 01 | 0 | x | x | 0 |
| 11 | 1 | x | x | x |
| 10 | 0 | x | x | x |

**$K_C$**

| BA \ DC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | x | 0 | x | x |
| 01 | x | 0 | x | x |
| 11 | x | 1 | x | x |
| 10 | x | 0 | x | x |

**$J_D$**

| BA \ DC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | x | x |
| 01 | 0 | 0 | x | x |
| 11 | 0 | 1 | x | x |
| 10 | 0 | 0 | x | x |

**$K_D$**

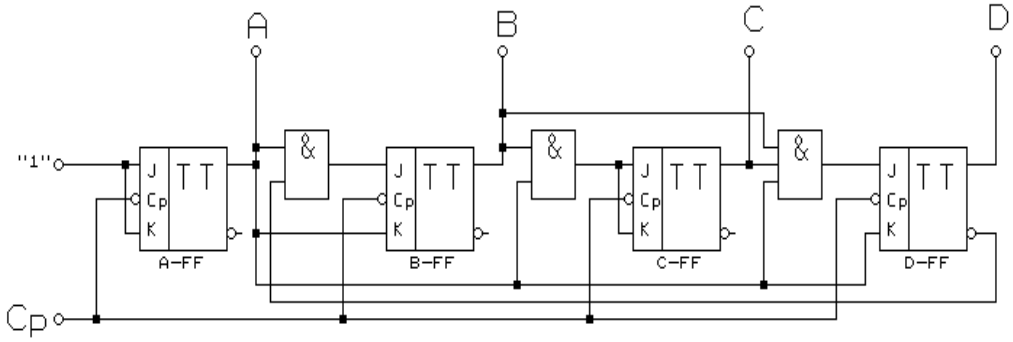
| BA \ DC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | x | x | x | 0 |
| 01 | x | x | x | 1 |
| 11 | x | x | x | x |
| 10 | x | x | x | x |

90. ábra

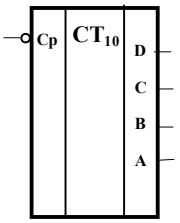
A BCD kódban nem szereplő kombinációkat is **x** -el jelölhetjük, s így a függvény egyszerűsítéseknél felhasználhatjuk. A logikai függvények:

$$J_A = 1 \qquad K_A = 1$$
$$J_B = A\overline{D} \qquad K_B = A$$
$$J_C = AB \qquad K_C = AB$$
$$J_D = ABC \qquad K_D = A$$

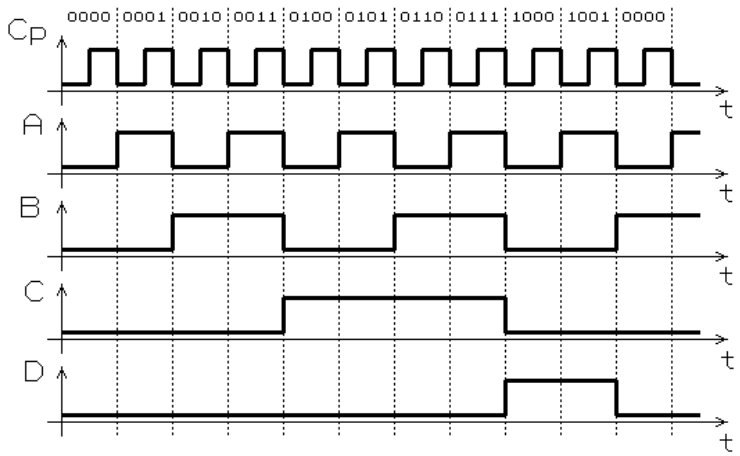
A szinkron BCD előreszámláló logikai vázlatát, szimbolikus jelölését, és a kimenetek időbeli változását a 91. a, b, c. ábrák mutatják.




a.




b.




c.

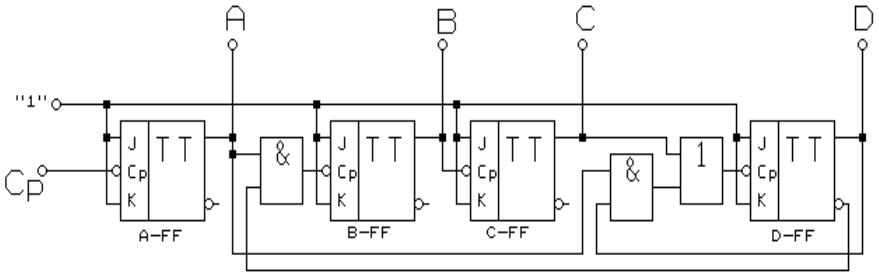
91. ábra

- **Aszinkron BCD számlálók**

A legegyszerűbb **aszinkron** üzemű **BCD** előreszámlálót egyetlen billentő bemenettel rendelkező flip-flop -okból építhetünk. Az egyes tárolók billentési feltételeit a 90. c. ábra jelalakjai alapján is felírhatjuk. 1 - 0 átmenetre billenő flip-flop -nál az egyes billentési feltételek:

$$\mathbf{C_{pA} = C_s}$$
$$\mathbf{C_{pB} = A\overline{D}}$$
$$\mathbf{C_{pC} = B}$$
$$\mathbf{C_{pD} = C + AD}$$

Az élvezérelt flip-flop -okból kialakított aszinkron BCD előreszámláló logikai vázlata a 92. ábrán látható.



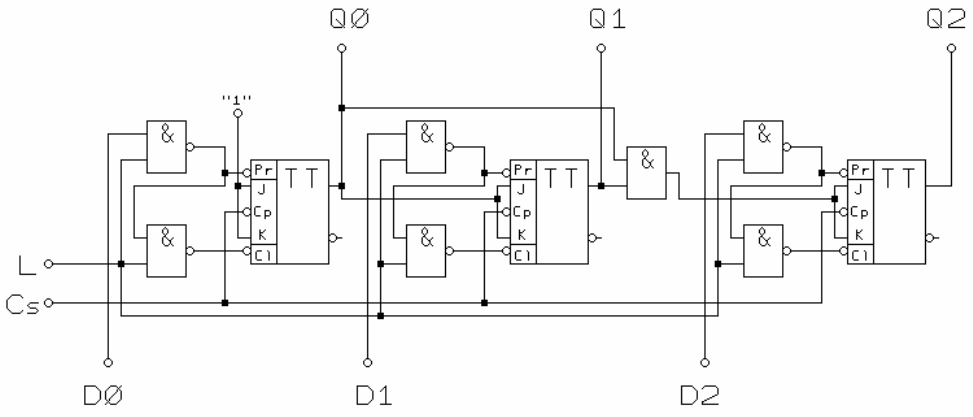

92. ábra

Az aszinkron hátra- és reverzibilis BCD számlálók kialakítása a megismert logikai tervezési eljárás segítségével lehetséges.

## ⇒ Preset számlálók

Számlálók alkalmazásakor szükség lehet arra, hogy a számlálást esetenként ne a 0-tól, vagy a kapacitás végértékétől kezdjük, hanem egy közbenső számtól. Ehhez szükséges, hogy külön külső parancs hatására, a számláló flip-flop -jait tetszőleges állapot-kombinációba lehessen billenteni. Ezeket nevezzük preset (elő beírású) számlálóknak.

A számláló tartalmának - egyidejű párhuzamos - változtatása, programozása is történhet aszinkron, ill. szinkron módon.

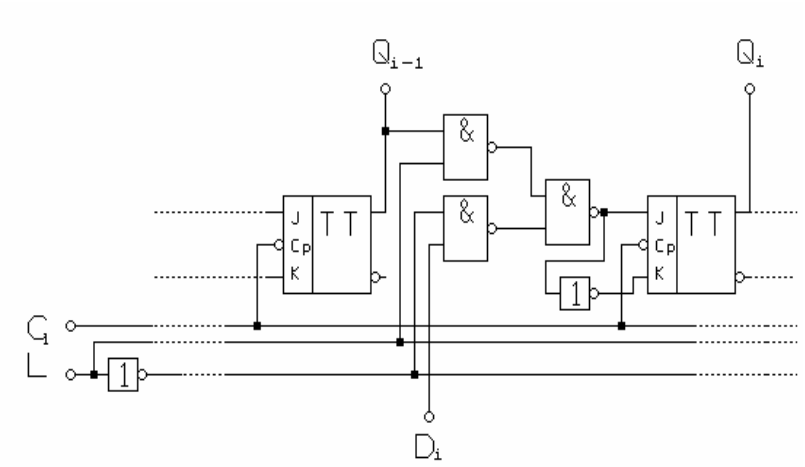



93. ábra

Az ***aszinkron*** programozás - az adatbeírás - a számlálandó jeltől ***függetlenül*** történik. A 93. ábrán látható 3 bites szinkron bináris előre számláló flip-flop -jai aszinkron üzemű beíró (**Pr**) és törlő (**Cl**) bemeneteit használjuk fel a párhuzamos adatbeírásra. Az adatbemenetek $\mathbf{D_0}$, $\mathbf{D_1}$, $\mathbf{D_2}$ és a beírást vezérlő jel az **L** (Load). Ha az L-re 0 szintet adunk, akkor a flip-flop -okba az adatbemeneteken érvényes információ íródik.

Amíg az L-en aktív jel van, addig az áramkör számlálóként nem működik. Az aszinkron bemenetek (Pr, Cl) hatása erősebb a Cp billentő jelnél.

A szinkronprogramozású megoldásnál a beírandó adatot mindig a soron következő Cp jel írja be a flip-flop -okba. Ennek egy áramköri megvalósítására mutat példát a 94. ábra.




94. ábra

Mindegyik flip-flop előtt azonos felépítésű kiválasztó áramkör van. Az L beíró jel 1 szintjénél a $\mathbf{D_i}$ adatút tiltott és a számlálási feltételek kerülnek a flip-flop -ok előkészítő bemeneteire.

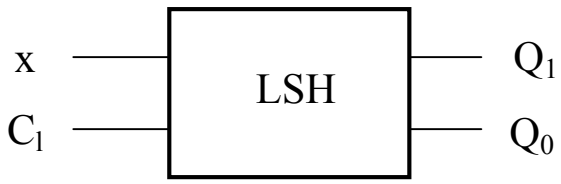
L = 0 vezérlésnél az adatok értéke jut a JK be-menetekre. A tényleges beírás ekkor is a Cp jel 1 - 0 átmenetekor következik be. Ameddig az L aktív, addig a párhuzamos beírás érvényesül.

## 2.7. Léptetőregiszterek

A léptető regiszterek (shift - regiszter) kettős feladatot ellátó ***funkcionális*** áramkörök. Egyrészt egy n bites **digitális szó tárolására**, másrészt egy-egy léptető jel hatására, a bemenetre érkező jel, valamint a már tárolt információ **léptetésére** használhatók. A hálózat annyi tárolóból (flip-flop -ból) áll, ahány bites információt kell tárolni, illetve léptetni.

A következőben határozzuk meg a hálózat felépítését. Az általánosságon nem esik csorba, ha két bitre végezzük el a feladatot.

A megvalósítandó hálózatnak két kimenete (**$Q_0$, $Q_1$**), egy információs (**x**), és egy léptető (**$C_l$**) bemenete kell legyen. A blokkvázlat látható a 95. ábrán.




95. ábra

Írjuk fel a feladat állapottáblázatát (96. ábra). Négy lehetséges bemeneti (Xi), kimeneti kombináció ($Z_i$) lehet. A lehetséges állapotok száma is négy.

- A bemeneti kombinációk az x, és a $C_l$ értékvariációiból adódnak.

| | x | $C_l$ |
|---|---|---|
| $X_0$ | 0 | 0 |
| $X_1$ | 0 | 1 |
| $X_2$ | 1 | 0 |
| $X_3$ | 1 | 1 |

- A kimeneti kombinációkat a $Q_0$, és a $Q_1$ variációi adják,

| | $Q_1$ | $Q_0$ |
|---|---|---|
| $Z_0$ | 0 | 0 |
| $Z_1$ | 0 | 1 |
| $Z_2$ | 1 | 0 |
| $Z_3$ | 1 | 1 |

- Az állapotok megegyeznek a kimeneti kombinációkkal.

| ***Ki-mene-tek*** | ***Bemenetek*** | | | |
|---|---|---|---|---|
| $\mathbf{Z^*}$ | $\mathbf{X_0}$ | $\mathbf{X_1}$ | $\mathbf{X_2}$ | $\mathbf{X_3}$ |
| $\mathbf{Z_0^*}$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_1$ |
| $\mathbf{Z_1^*}$ | $Z_1$ | $Z_2$ | $Z_1$ | $Z_3$ |
| $\mathbf{Z_2^*}$ | $Z_2$ | $Z_0$ | $Z_2$ | $Z_1$ |
| $\mathbf{Z_3^*}$ | $Z_3$ | $Z_2$ | $Z_3$ | $Z_3$ |

96. ábra

A 97. ábrán látható kódolt vezérlési táblázatot – a számlálóéhoz hasonlóan (lásd. 78. ábra) – írtuk fel.

| ***Kimeneti kombiná-ciók*** | | ***Bemeneti-*** ($X_i$)***, és vezérlőjel*** ($V_i$) ***kombinációk*** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ***X0*** | | ***X1*** | | ***X2*** | | ***X3*** | |
| | | ***x Cl*** | | ***x Cl*** | | ***x Cl*** | | ***x Cl*** | |
| | | ***0 0*** | | ***0 1*** | | ***1 0*** | | ***1 1*** | |
| $\mathbf{Q_1}$ | $\mathbf{Q_0}$ | $\mathbf{v_1}$ | $\mathbf{v_0}$ | $\mathbf{v_1}$ | $\mathbf{v_0}$ | $\mathbf{v_1}$ | $\mathbf{v_0}$ | $\mathbf{v_1}$ | $\mathbf{v_0}$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

97. ábra

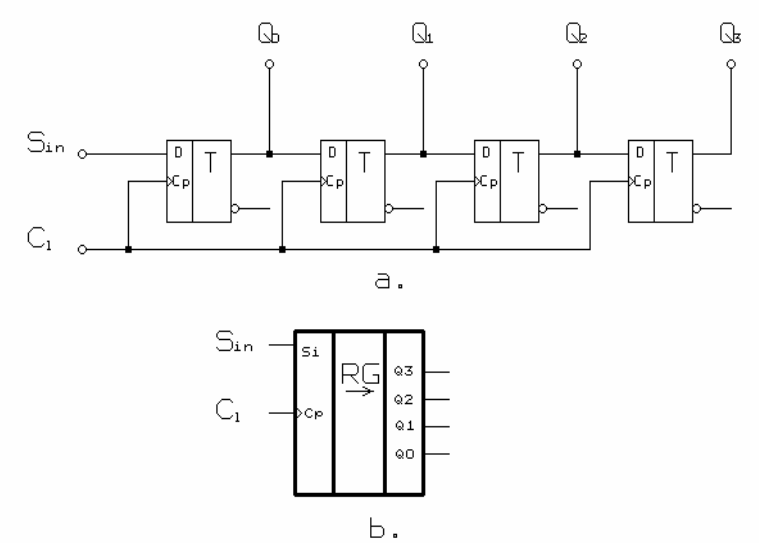
A táblázatból elhagyhatók azok a bemeneti kombinációk oszlopai, amelyeknél ($X_0$, $X_2$) léptetőjel nincs ($C_l$=0). A táblázat tovább egyszerűsíthető azáltal, hogy a léptetőjel minden flip-flop billentő bemenetére kapcsolódik. Az így kapott táblázatban (98.ábra) már csak az előkészítő bemenetek ($v_i$) maradnak.

| ***Kimeneti kombiná-ciók*** | | ***Bemenet*** (x)***, és vezérlőjelek*** ($v_i$) | | | |
|---|---|---|---|---|---|
| | | ***0*** | | ***1*** | |
| $Q_1$ | $Q_0$ | $v_1$ | $v_0$ | $v_1$ | $v_0$ |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 |

98. ábra

A megfelelő flip-flop kiválasztása után írható fel a tényleges vezérlési táblázat

A 95. ábrán látható egy 4 bites - élvezérelt D típusú flip-flop -okból kialakított – léptető regiszter logikai vázlata. Léptető regiszternél - az információ átmeneti tárolása mellett - a szomszédos flip-flop -ok között olyan csatolást kell megvalósítani, amely biztosítja, hogy közöttük egy külső léptető jel hatására információ átadás történjen.




| t | $Q_0$ | $Q_1$ | $Q_2$ | $Q_3$ |
|---|---|---|---|---|
| $t_1$ | 1 | x | x | x |
| $t_2$ | 0 | 1 | x | x |
| $t_3$ | 1 | 0 | 1 | x |
| $t_4$ | 0 | 1 | 0 | 1 |

c.

99. ábra

A $\mathbf{C_l}$ léptető jel 0 - 1 átmeneténél mindegyik tároló elembe a D bemenetén érvényes logikai érték íródik. Azáltal, hogy az egyes $\mathbf{D_i}$ bemeneteket az előző flip-flop $\mathbf{Q_{i-1}}$ kimenetével kötöttük össze, a tárolt digitális szó léptetése történik. A legelső flip-flop - ba pedig az $\mathbf{S_i}$ jelű bemenet aktuális értéke íródik. A 37.b. ábrán a léptető regiszter **szimbolikus jele** látható. A 94.c. ábrán táblázatban szemléltetjük a működési ütemeket, ha a soros bemenetre ($\mathbf{S_i}$) **1 0 1 0** jelsorozat érkezik a léptető-jel ütemezésében ( az x a léptetés előtti ismeretlen tartalmat jelzi ). Az egyes sorok az egymás után érkező léptető impulzusok hatására bekövetkező állapotokat tartalmazzák.

## ⇒ A léptető regiszterek fajtái

A léptető regisztereket csoportosíthatjuk

- az információ beírása és
- a kiolvasás

***módja*** szerint, valamint a ***léptetés*** iránya alapján.

A ***beírás*** és a ***kiolvasás*** szerint megkülönböztetünk

- párhuzamos és
- soros

beírású, ill. kiolvasású léptető regisztereket.

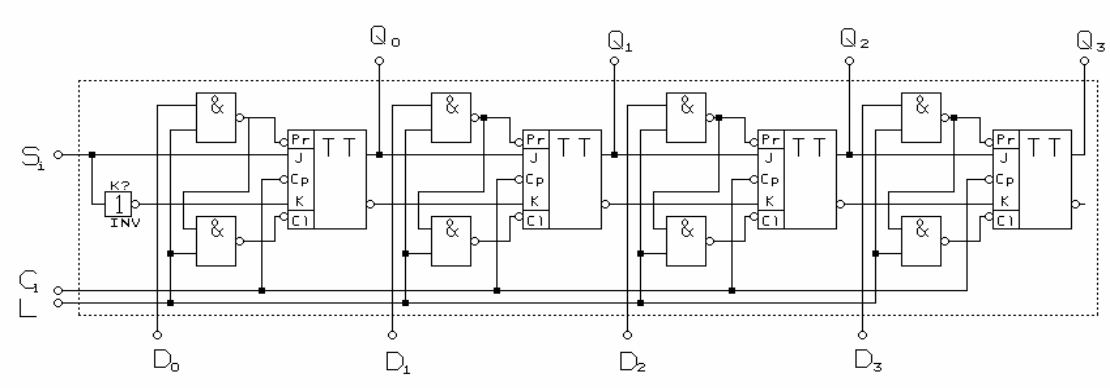
A léptetés ***iránya*** szerint

- ***jobbra*** (az alacsonyabb nagyságrend irányába),
- ***balra*** (a magasabb nagyságrend irányába), és
- ***kétirányú*** léptetésű

regiszterek vannak.

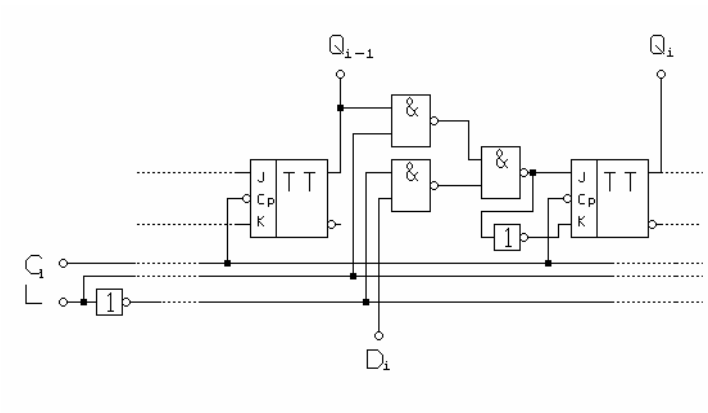
A 99. ábra szerinti léptető regiszter **soros beírású** jobbra léptető regiszter. Amennyiben a kimenetek ($\mathbf{Q_0}$, $\mathbf{Q_1}$, $\mathbf{Q_2}$, $\mathbf{Q_3}$) mindegyike kivezetett, akkor a kiolvasás **párhuzamos**. Amennyiben csak a $\mathbf{Q_3}$ kimenethez csatlakozhatunk, akkor a **kiolvasás** módja **soros**. Ez az utóbbi megoldás elsődlegesen az integrált áramköri kialakításoknál használt a szükséges lábszám csökkentéséhez.

A párhuzamos információ-beírás - a számlálóknál már megismertekhez hasonlóan - **aszinkron** és **szinkron** módon történhet. Az 100. ábrán egy aszinkron párhuzamos beírású, jobbra léptető regiszter logikai vázlata látható. A párhuzamos szó beírása az **L**=1 értéknél történik az egyes flip-flop -ok **Pr** és **Cl** bemenetein keresztül, tehát aszinkron módon. Ekkor a $\mathbf{C_l}$ léptető jel hatása nem érvényesül. Az **L**=0 értéknél soros beírású ($\mathbf{S_i}$ = serial input), jobbra léptető regiszterként működtethető az áramkör.




100. ábra

A **szinkron** üzemű **párhuzamos beírás** egy áramköri megoldását mutatja a 101. ábra, amely egy léptető regiszter egy részletét mutatja. Az **L** beíró jel **1** értéke a léptetési üzemmódot választja. Ekkor az **i** - ik flip-flop -ba - a $\mathbf{C_l}$ 1 - 0 jelváltásakor - az **i-1** - ik flip-flop értéke íródik.




101. ábra

Az **L=0** vezérlésnél a $\mathbf{C_l}$ jel a $\mathbf{D_i}$ információ érvényes értékét írja az **i** - ik flip-flop -ba, vagyis párhuzamos beírás történik.

## ⇒ A léptetőregiszterek alkalmazása

A léptetőregisztereket, mint átmeneti tárolókat (tartó áramkörök) szinte minden digitális berendezésben alkalmazzák. Ezzel itt részletesebben nem foglalkozunk. Viszont tárgyaljuk a léptetőregiszterek

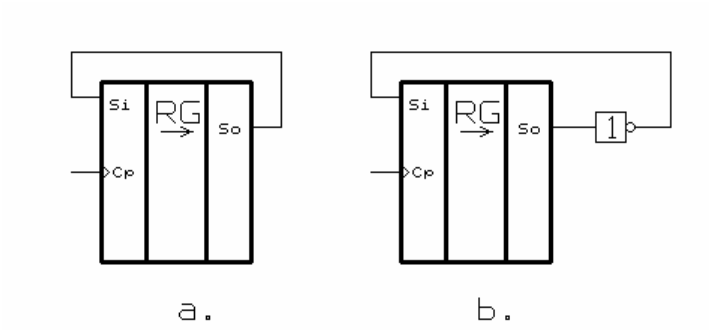
- gyűrűs számlálóként, ill.
- soros-párhuzamos és
- párhuzamos-soros

***kódátalakítókban*** való felhasználását.

### ▪ Gyűrűs számlálók

Amennyiben egy léptető regiszter soros kimenetét (**So**) a soros bemenettel (**Si**) összekötjük, akkor olyan áramkört kapunk, amelyben az információ kering ( a kilépő bit beíródik az első tárolóba). Ezt a megoldást nevezzük ***gyűrűs számlálónak***.

Az adat visszavezetése történhet egyenes és tagadott alakban is (102. ábra). Az a. ábra szerinti visszavezetési megoldással ***n - modulusú***, míg a b. ábra szerint ***2n - modulusú*** gyűrűs számlálót kapunk, ahol **n** a regiszter tárolóinak száma.




102. ábra

Az ***n - modulusú*** gyűrűs számlálónál az eredeti információ az n. lépés után kerül vissza a regiszter megfelelő helyértékeire. Erre mutat példát a 103.ábra szerinti működési táblázat, amelyen egy 4 bites n modulusú gyűrűs számláló egyes ütemeinek állapota látható az **1 0 0 0** kezdő feltételből indulva.

| t | $Q_0$ | $Q_1$ | $Q_2$ | $Q_3$ |
|---|---|---|---|---|
| $t_1$ | 1 | 0 | 0 | 0 |
| $t_2$ | 0 | 1 | 0 | 0 |
| $t_3$ | 0 | 0 | 1 | 0 |
| $t_4$ | 0 | 0 | 0 | 1 |
| $t_5$ | 1 | 0 | 0 | 0 |

103. ábra

Az áramkört felhasználhatjuk, pl. soros működésű aritmetikai egység átmeneti tárolójaként, ha az egyik tényezőt - műveletvégzés után - változatlanul kívánjuk megtartani.

Számlálóként is használhatjuk a gyűrűs számlálót. Ha a regiszterben egy darab 1-et léptetünk, akkor minden állapotban egyetlen kimenet értéke lehet 1 szintű. Ha az n kimenet mindegyikéhez egy N alapszámú számrendszer egy számjegyét rendeljük, akkor ***1 az N-ből*** kódolású számlálót kapunk.

A ***2n modulusú*** gyűrűs számlálóban **2n** számú léptetés után kapjuk vissza az eredeti állapotot. A 104.a. ábrán levő táblázat mutatja egy 4 bites 2n modulusú gyűrűs számláló állapotsorozatát, ha a **0000** állapotból indulunk ki. Ugyanezen gyűrűs számlálóban a b. ábra táblázata szerinti állapotsorozat is kialakulhat. Mindkét sorozat 8-8 állapotból (2n) - két teljes ciklusból - áll. A kettő együtt tartalmazza a lehetséges 16 kombinációt.

| **Ütem** | **$Q_3$** | **$Q_2$** | **$Q_1$** | **$Q_0$** |
|---|---|---|---|---|
| **1** | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 0 | 1 |
| **3** | 0 | 0 | 1 | 1 |
| **4** | 0 | 1 | 1 | 1 |
| **5** | 1 | 1 | 1 | 1 |
| **6** | 1 | 1 | 1 | 0 |
| **7** | 1 | 1 | 0 | 0 |
| **8** | 1 | 0 | 0 | 0 |
| **9** | 0 | 0 | 0 | 0 |

a.

| **Ütem** | **$Q_3$** | **$Q_2$** | **$Q_1$** | **$Q_0$** |
|---|---|---|---|---|
| **1** | 1 | 0 | 0 | 1 |
| **2** | 0 | 0 | 1 | 0 |
| **3** | 0 | 1 | 0 | 1 |
| **4** | 1 | 0 | 1 | 1 |
| **5** | 0 | 1 | 1 | 0 |
| **6** | 1 | 1 | 0 | 1 |
| **7** | 1 | 0 | 1 | 0 |
| **8** | 0 | 1 | 0 | 0 |
| **9** | 1 | 0 | 0 | 1 |

b.

104. ábra

Általánosan a következő törvényszerűség fogalmazható meg: egy **n** bites léptető regiszterből kialakított ***2n*** modulusú gyűrűs számláló **k** féle ***teljes*** ciklusban működtethető, ahol

$$\mathbf{k = \frac{2^n}{2n}}$$

hányados egész része. Amennyiben az osztás eredménye nem egész szám, akkor csonka ciklus is van. ***Csonka ciklusnak*** nevezzük az olyan sorozatot, amely 2n lépésnél hamarabb veszi fel a kezdő kombinációt. A csonka ciklus állapotainak száma az osztásnál kapott maradékkal egyezik meg.

***Példa:***

**n=3** esetén **egy** 6 állapotú (2n = 6) ***teljes*** ciklus és **egy** kétállapotú ***csonka*** ciklus lehetséges. **n=5** bites gyűrűs számlálónál **három** 10 állapotú ***teljes*** ciklus és **egy** kétállapotú ***csonka*** ciklus létezik. A kezdőszám fogja meghatározni, hogy melyik ciklusban üzemel a számláló. Amennyiben több teljes ciklus is lehetséges, ezek közül azt tekintjük ***alap-ciklusnak,*** amely tartalmazza az összes bit 0 kombinációt.

Az **öt**bites **2n** modulusú gyűrűs számlálót decimális számlálóként is használjuk. A lehetséges három teljes ciklusból a 00000 állapotot is tartalmazó sorozatot (alap-ciklus) nevezzük ***Johnson - kódnak***. Ahhoz, hogy a gyűrűs számláló mindig az alap- ciklusban üzemeljen, biztosítani kell, hogy az esetleges ciklustévesztés után (pl. külső zavar) automatikusan kerüljön vissza az alap-ciklusba. Egyik megoldás lehet, ha egy élvezérelt D flip-flop a soros kimenet 1 - 0 átmenetekor bebillen és törli a számláló flip-flop -jait. Ez a törlés a helyes működést nem zavarja, mivel az alap-ciklusban egyébként is ez az állapot kell következzen. A következő órajel 1 szintje aszinkron módon törli a D flip-flop –ot. Ha valamilyen okból hibás állapot áll be, ezt - néhány ütem után - automatikusan törölni fogja a D tároló.

***Példa:*** Vegyük azt, hogy valamilyen zavar eredményeként az **10010** hibás állapotot lép fel. A következő ütem az **11001**, majd **<u>01100</u>** lenne, de az utóbbi beálltakor a D flip-flop is bebillen s ez a számláló **00000** állapotát állítja be. Ennek eredményeként csak

egyetlen hibás ciklus lesz.

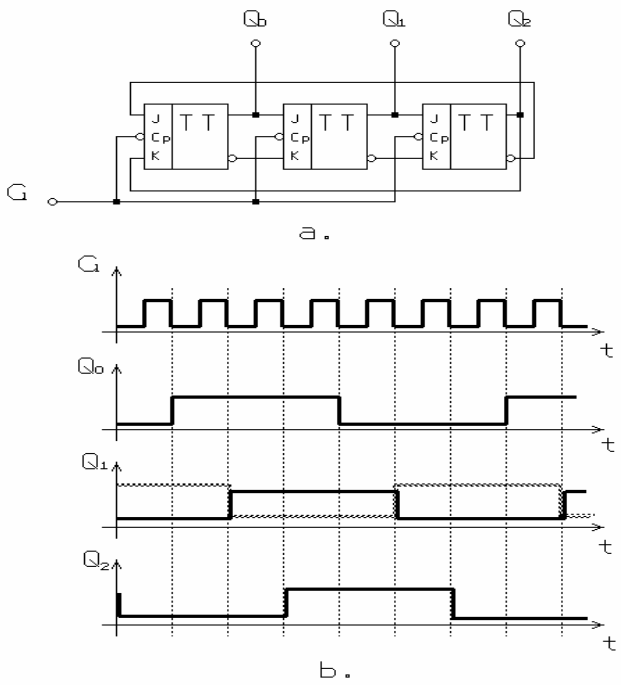
Röviden említést teszünk a **2n** modulusú gyűrűs számlálók egy speciális vezérlés-technikai felhasználásáról. Amennyiben **n=k*3**, vagyis a három egész számú több-szöröse, akkor a számláló kimenő jeleiből mindig előállítható ***3 fázisú szimmetrikus jelrendszer***.

A 105. a. ábrán **3** bites 2n modulusú gyűrűs számláló logikai vázlata látható. A b. ábra szemlélteti az órajel és a kimeneti jelek idő-függvényeit.

Mindhárom kimenet jele szimmetrikus négyszögjel, és frekvenciája

$$\mathbf{f_{ki} = \frac{f_q}{2n}}$$

ahol $\mathbf{f_q}$ a léptető jel frekvenciája, és n a regiszter bitjeinek száma. A b. ábrán látható, hogy az egyes kimenetek jelei egy léptető-jel periódus idejével késnek egymáshoz képest.




105. ábra

Mindegyik jel periódus-ideje 6 ütem, amit tekinthetek 360 villamos foknak. Ebből következik, hogy az egyes jelek közötti fázistolás:

$$\Phi = \frac{2\Pi}{n} = \frac{360^\circ}{6} = 60^\circ$$

Amennyiben a $Q_0, \overline{Q}_1, Q_2$ jelsorozatot tekintjük, ezek - bármilyen órajel frekvenciánál - pozitív sorrendű szimmetrikus háromfázisú rendszert alkotnak. Ezért háromfázisú rendszerek - pl. aszinkron motorok fordulatszám változtatásánál stb.- vezérlő jeleként felhasználhatók.

- **Párhuzamos-soros kódátalakítás**

A fejezetben röviden ismertetjük a léptetőregiszterek alkalmazásával megvalósítható **párhuzamos-soros** kódátalakítást.

Az átalakítás elve, hogy az átalakítandó, párhuzamos kódolású információt a léptetőregiszterbe - a **párhuzamo**s adatbemeneteken keresztül - **írjuk be**. Ezt követően - az órajel ütemében - léptetve a regiszter tartalmát, annak **soros kimenetén** (So) időben egymás után - egyetlen csatornán - kapjuk az információ egyes bitjeit. Ezzel soros kódolásban áll rendelkezésünkre az eredeti információ. A kódátalakító áramkörnek biztosítania kell, hogy minden párhuzamos beírást - a szóhossznak megfelelő - **n** számú léptetés kövessen. Ezután ismét a párhuzamos beírás, vagyis az új információ fogadása következik.

A párhuzamos-soros kódátalakítókat leggyakrabban a nagyobb távolságú adatátviteli rendszereknél használják. Soros kódban való információátvitelhez egyetlen adatcsatorna szükséges.

- **Soros-párhuzamos kódátalakítás**

A soros-párhuzamos kódátalakítás elve, hogy az átalakítandó **n** bites ***információt*** CL órajel lépteti be a ***regiszterbe***. Majd az **n+1**-edik ütemben (”szó szünet”) kerül a párhuzamosan kódolt információ a kimeneti csatornákra.