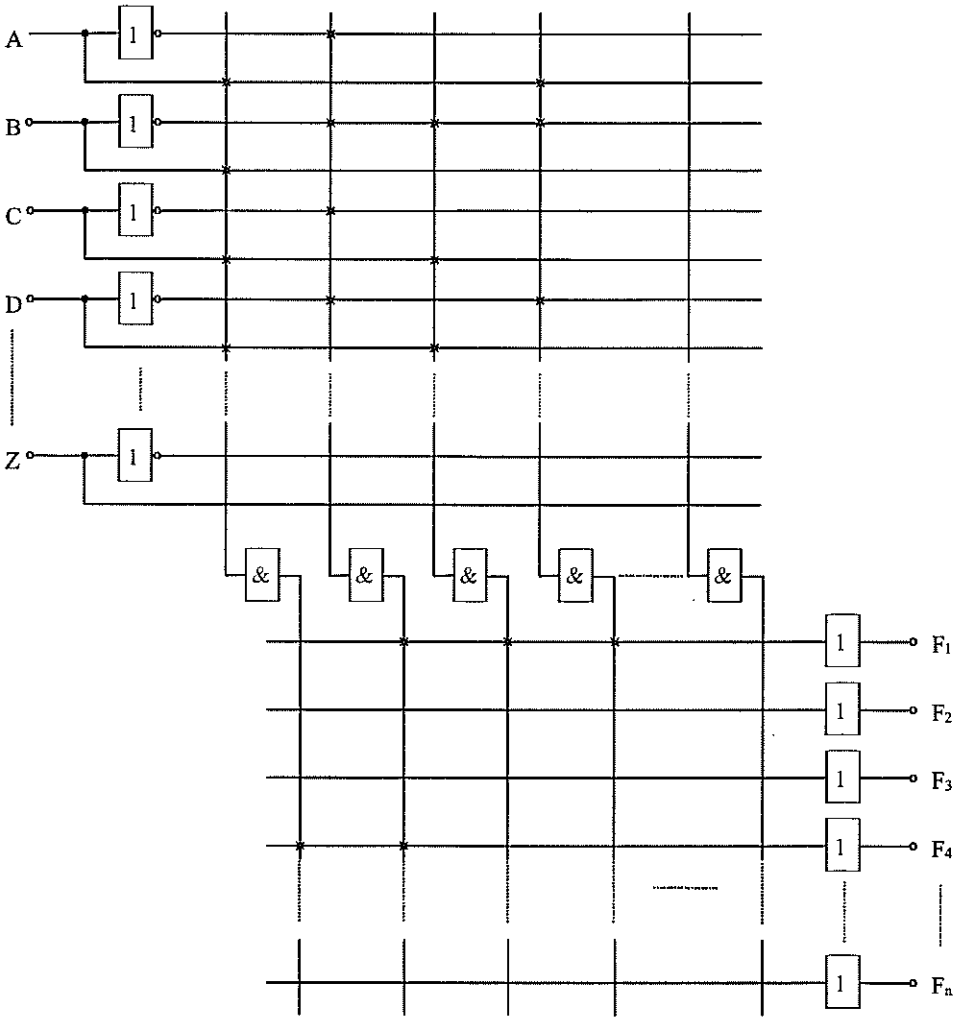
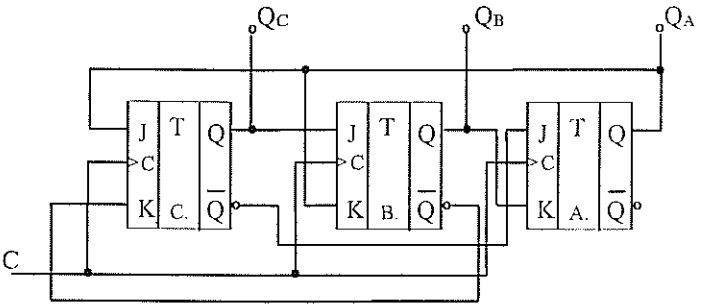


5.19. ábra. A PLA belső felépítése

5.2. Sorrendi hálózatok

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

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



5.20. ábra. Szinkron sorrendi hálózat

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

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

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

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

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

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

Ezek a vezérlések a J-K tároló vezérlési táblázata (**4.3.** ábra) alapján a következő kimeneti állapotokat hozzák létre:

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

Így a hálózat új kimeneti állapota:

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

Ez az állapot új vezérléseket jelent a tárolóknak:

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

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

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

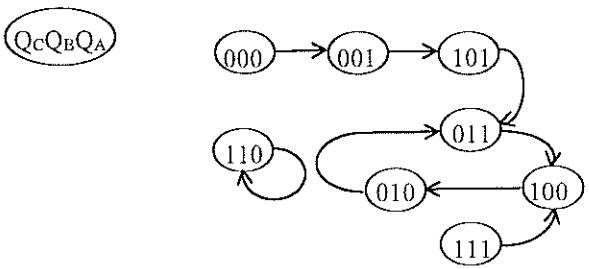
- a C tároló megváltoztatja az előző állapotát, $Q_C = 1$,
- a B tároló törlődik, $Q_B = 0$,
- az A tároló beíródik, $Q_A = 1$.

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

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

C	J_C	K_C	J_B	K_B	J_A	K_A	Q_C	Q_B	Q_A	
							0	0	0	
1.	0	1	0	0	1	0	0	0	1	
2.	1	1	0	1	1	0	1	0	1	
3.	1	1	1	1	0	0	0	1	1	
4.	1	0	0	1	1	1	1	0	0	
5.	0	1	1	0	0	0	0	1	0	
6.	0	0	0	0	1	1	0	1	1	⇐ már előfordult,
							1	1	1	⇐ feltételezett új kimeneti állapot,
1.	1	0	1	1	0	1	1	0	0	⇐ már előfordult,
							1	1	0	⇐ feltételezett új kimeneti állapot,
1.	0	0	1	0	0	1	1	1	0	⇐ már előfordult.

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



5.21. ábra. Állapotdiagram

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

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

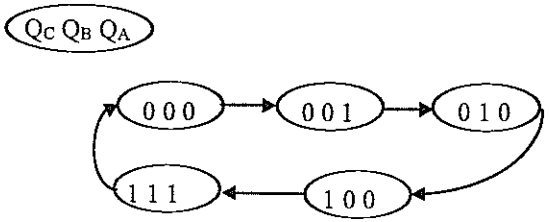
Az analízis során megismert hálózat szinkron sorrendi hálózat volt, mert a tárolók egyszerre kapták az órajelet, tehát a kimenetükön az állapotváltozás is egyidőben következett be. A szinkron működésű sorrendi hálózatokkal tanulmányaink során csak a számlálóknál találkozunk, ezért működésüket is ott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

A szinkron sorrendi hálózatok realizálása az állapotdiagram meghatározásával kezdődik.

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

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

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



5.22. ábra. A közlekedési lámpa állapotdiagramja

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

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

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

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

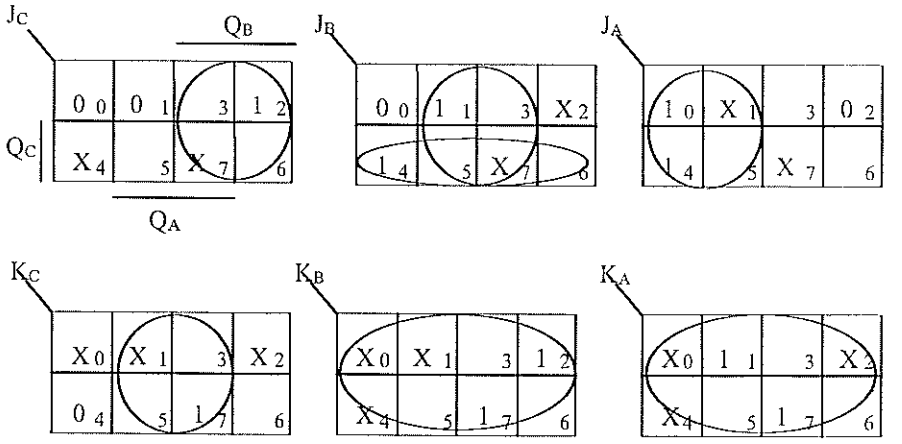
Az állapotátmeneti tábla első két oszlopa tartalmazza az összes lehetséges állapotátmenetet, amely egy feladat megoldása során előfordulhat. A J és K oszlopa pedig azokat a szükséges vezérléseket adja meg, amelyek az első két oszlopban levő állapotátmenetekhez szükségesek. A táblázat egyes sorainak magyarázata:

- első sor: ahhoz, hogy a J-K tároló kimenetén 0→0 átmenet jöjjön létre az órajel hatására, nem szabad a tárolót beírni, tehát J = 0. A K bemenetre adott vezérlés közömbös, mert ha K = 0, akkor a tároló megtartja előző állapotát, ha K = 1, akkor törlődik. Vagyis a flip-flop kimeneti állapota mindenképpen 0 lesz (l. a J-K tároló vezérlési táblázatát a **4.3.** ábrán),
- második sor: ahhoz, hogy a tároló kimenetén 1 legyen, J = 1 vezérléssel beírást kell végezni. A K bemenet vezérlése közömbös, mert K = 0-ra beírás történik, K = 1 vezérlésre pedig a flip-flop megváltoztatja előző állapotát ($J = 1, K = 1 \rightarrow Q_{n+1} = \overline{Q}_n$),
- harmadik sor: a tárolónak törlődni kell, ehhez K = 1 vezérlés kell. A J bemenet állapota közömbös, aminek magyarázata az előzőekből már következik,
- negyedik sor: nem szabad a tárolót törölni, tehát K = 0.

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

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

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



5.23. ábra. A vezérlési függvények táblázatai

A táblázatra vonatkozó hurkolási szabályok:

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

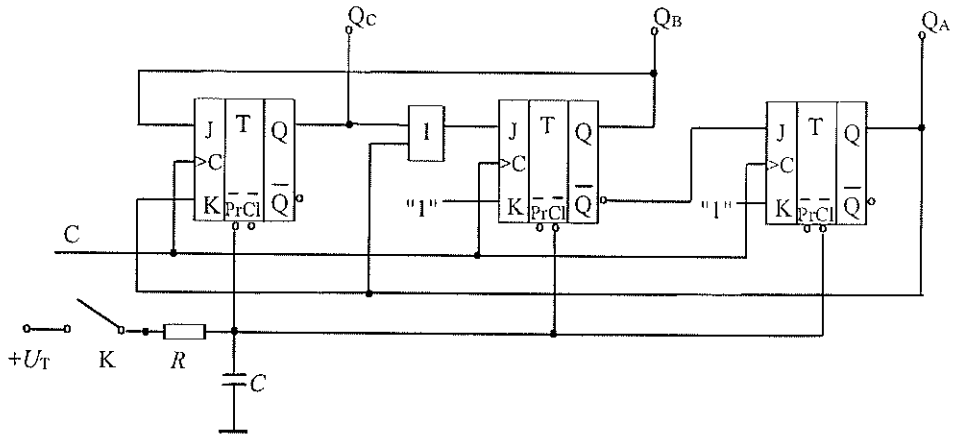
elő a működés során, kivéve a bekapcsolás pillanatát. Ha megfelelő kapcsolástechnikai megoldással gondoskodunk arról, hogy bekapcsoláskor valamelyik felhasznált állapotba kerüljön a rendszer, akkor valóban semmilyen körülmények között nem fordulhatnak elő az üres cellák termjei. Ezt feltételezve az üres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

Ezeket a szabályokat figyelembe véve a feladat vezérlési függvényei:

$J_C = Q_B$; $J_B = Q_C + Q_A$; $J_A = Q_B$,

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

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



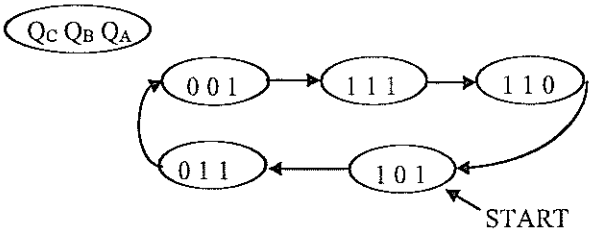
5.24. ábra. A közlekedési lámpa vezérlő áramköre

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

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

5. feladat

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



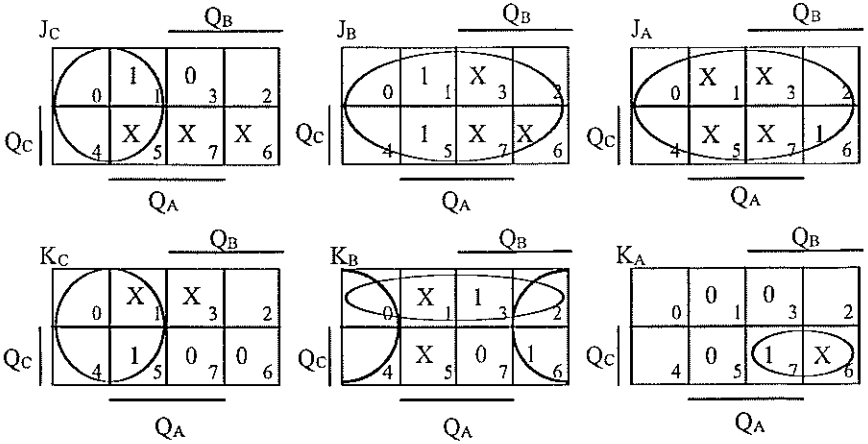
5.25. ábra. Az 5. feladat állapotdiagramja

Az 5. feladat megoldása

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

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

A bemenetek kitöltött vezérlési táblázatait az **5.26.** ábra mutatja.



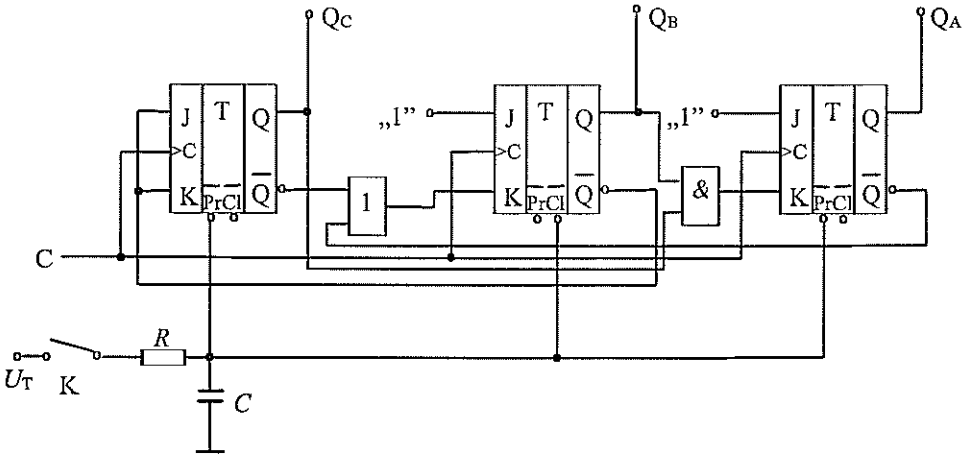
5.26. ábra. Az 5. feladat vezérlési táblázatai

A vezérlési táblázatokból kiolvasható vezérlési függvények:

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

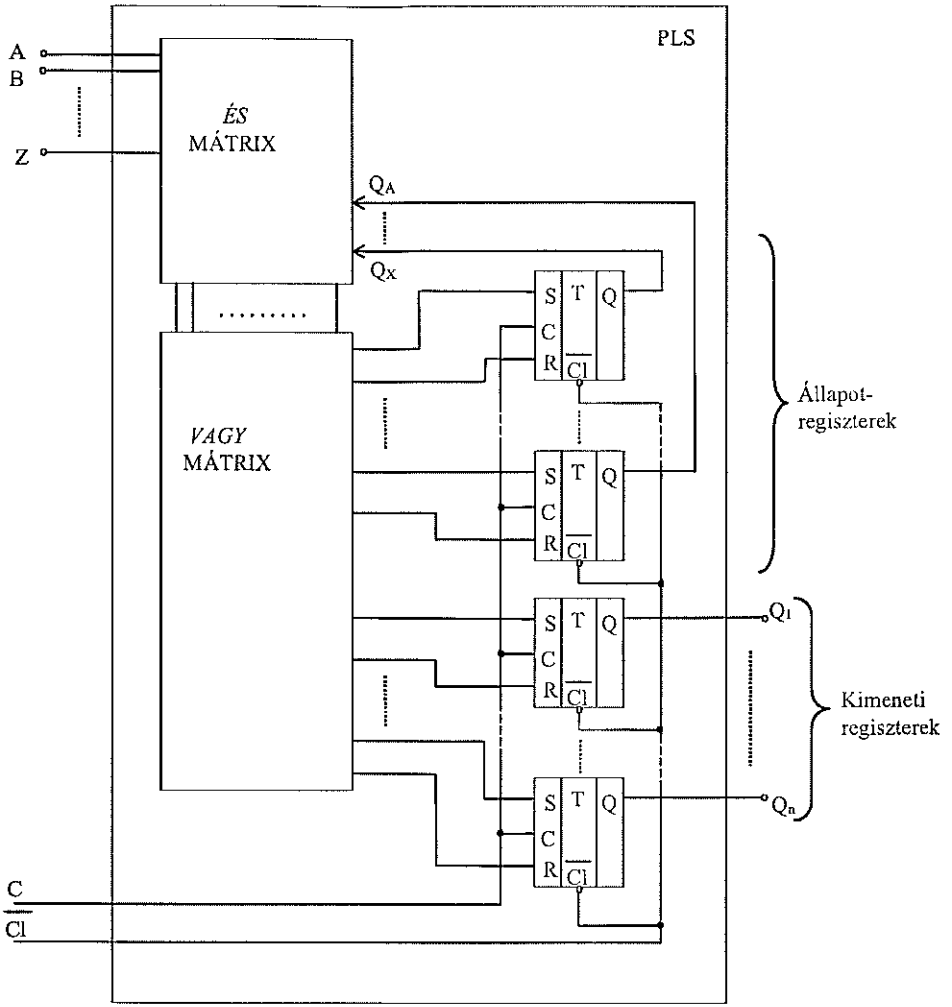
$K_C = \overline{Q_B}$; $K_B = \overline{Q_C} + \overline{Q_A}$; $K_A = Q_C \cdot Q_B$.

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



5.27. ábra. Az 5. feladat áramköre

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



5.28. ábra. A PLS áramkör belső felépítése

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

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

Ellenőrző kérdések

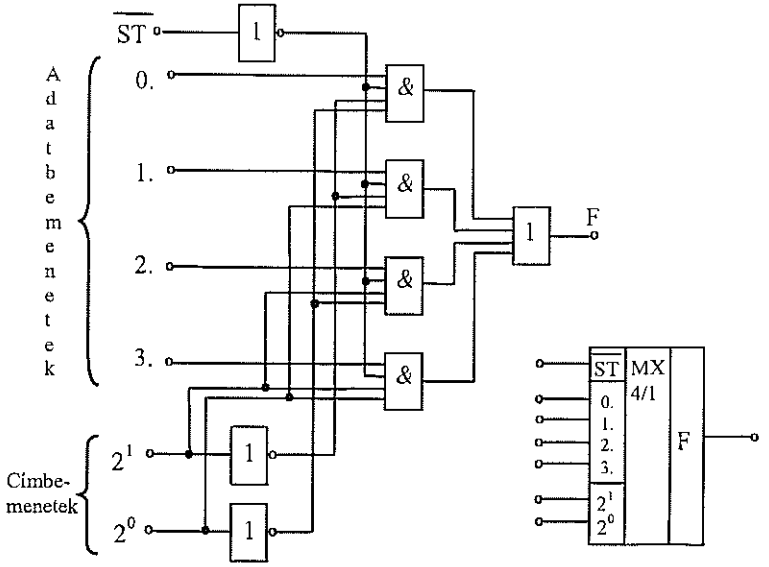
1. Mi a funkcionálisan teljes rendszer fogalma?
2. Ismertessük a NAND és NOR kapuval történő realizálás szabályait!
3. Mit jelent a hazárd, milyen formái vannak, hogyan szüntethető meg?
4. Rajzoljuk fel a PLA belső felépítését!
5. Milyen elemekből épülnek fel a sorrendi hálózatok?
6. Ismertessük a szinkron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

6. Funkcionális áramkörök

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

6.1. Multiplexerek

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



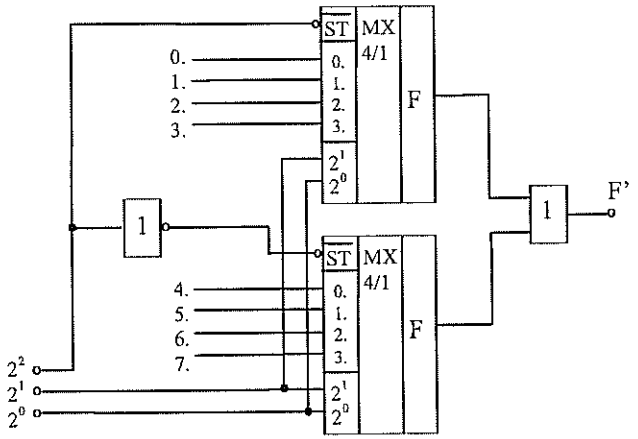
6.1. ábra. A 4/1 multiplexer belső felépítése és jelképi jelölése

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

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

Az SN sorozatban pl. a 74151 típus egy 8/1, a 74150 egy típus 16/1 multiplexer. A CD sorozatból a 4512B egy 8/1, a 4539B típus pedig két 4/1 multiplexer egy tokban.

A multiplexerek bemeneteinek száma **kaszkádosítással** (bővítéssel) növelhető. A **6.2.** ábra a módszer bemutatására két 4/1 multiplexer felhasználásával egy 8/1 multiplexer kialakítását mutatja.

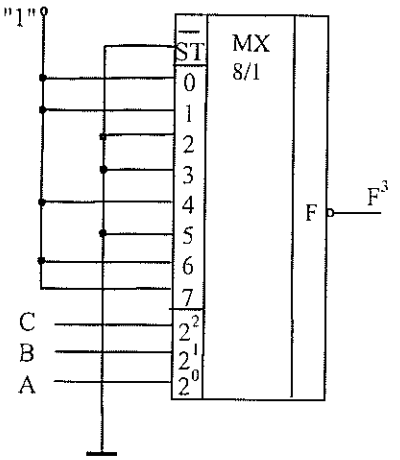


6.2. ábra. Multiplexerek kaszkádosítása

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

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

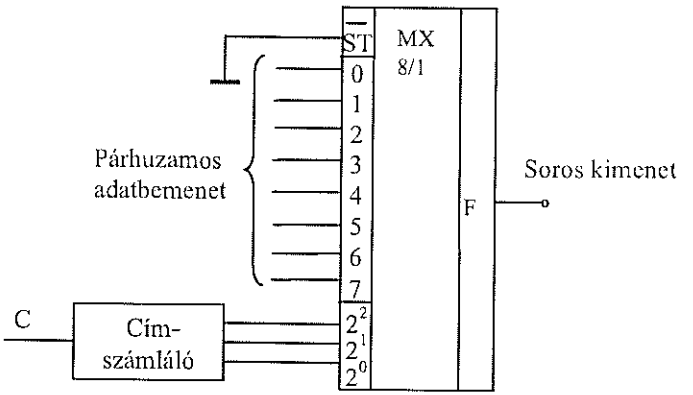
A multiplexerrel diszjunktív szabályos alakú függvényt lehet realizálni. Erre mutat példát a **6.3.** ábra. Az áramkör az $F^3 = \Sigma^3(0,1,4,6,7)$ függvényt realizálja.



6.3. ábra. Függvényrealizálás multiplexerrel

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

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

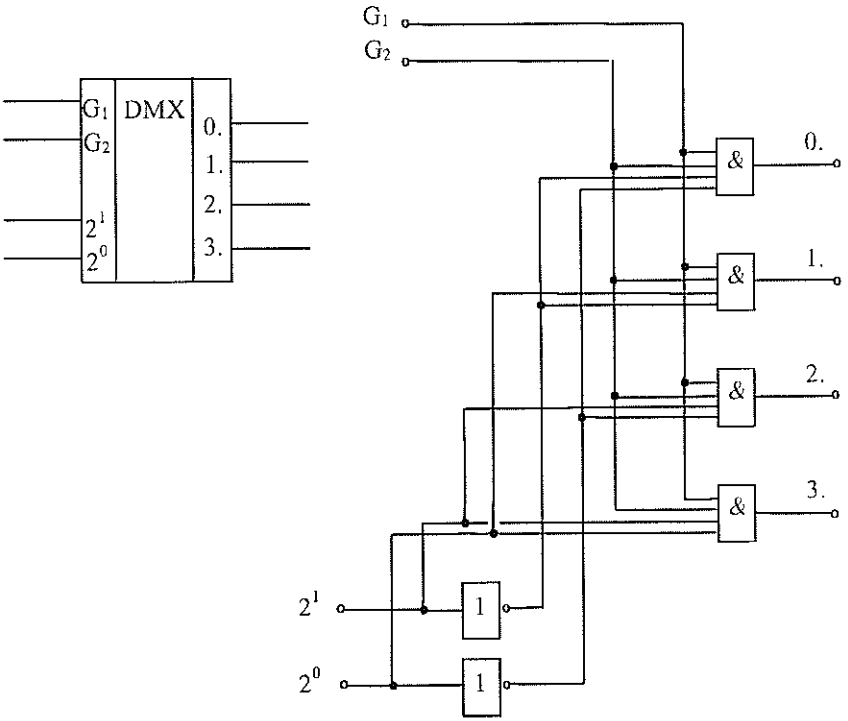


6.4. ábra. Párhuzamos-soros átalakító multiplexerrel

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

6.2. Demultiplexerek, dekódolók

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



6.5. ábra. A demultiplexerek belső felépítése

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

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

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

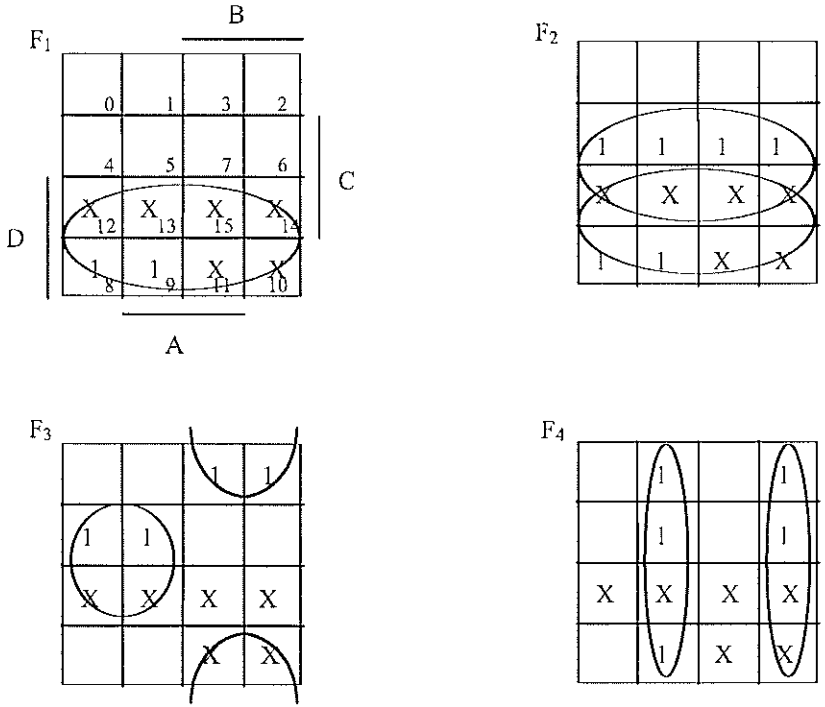
Az adatszétosztáson kívül a demultiplexerek dekódolóként is alkalmazhatók. A **6.5.** ábra áramköre pl. egy kétbites bináris számot dekódol decimális számmá. Ha az engedélyező bemenetekre 1 szintet kapcsolunk, akkor a címzésre használt kétbites bináris szám decimális megfelelőjével sorszámozott kimeneten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

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

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

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

Második lépésként – a kódtáblázatot egy többkimenetű kombinációs hálózat igazságtáblázatának tekintve – az F_1, F_2, F_3, F_4 függvényeket egyszerűsítjük, célszerűen grafikus módszerrel. A függvények V–K mintermtábláit a **6.6.** ábra szemlélteti.

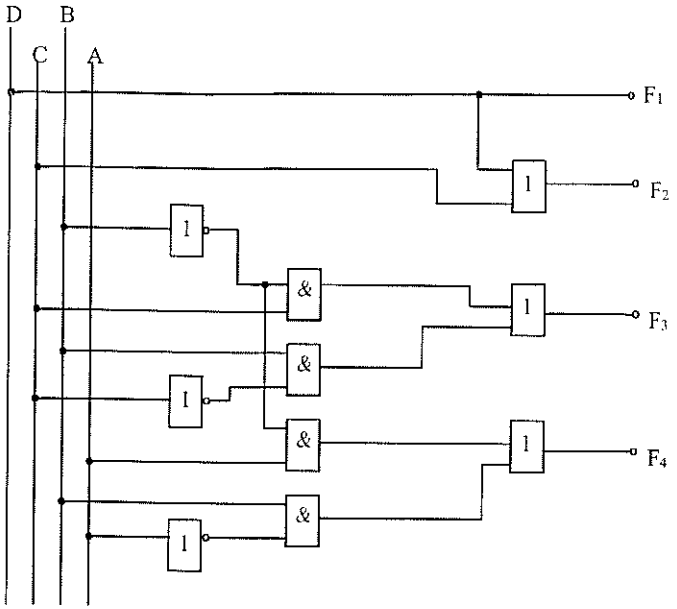


6.6. ábra. BCD-Gray-dekódoló V–K táblái

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

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

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



6.7. ábra. BCD-Gray-dekódoló

6.3. Aritmetikai áramkörök

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

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

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

B_0	A_0	S_0	C_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

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

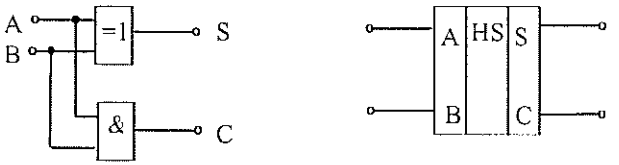
- az egyes helyi értékeken az összeadásnál figyelembe kell venni az előző helyi értékről jövő átvitelt. A táblázatban az átvitel indexe is erre utal: a 0. helyi értéken keletkező átvitelt az 1. helyi értéken kell figyelembe venni,
- a legnagyobb helyi értéken keletkező átvitel az összeg legnagyobb helyi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

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

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

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

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



6.8. ábra. Félösszeadó áramkör

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

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

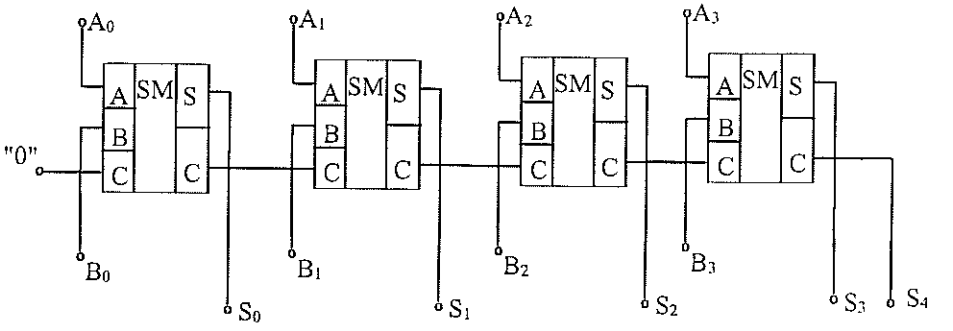
Az igazságtáblázat alapján felírható a teljes összeadó összeg- és átvitelfüggvénye. Az összegfüggvény nem egyszerűsíthető, az átvitelfüggvény igen. Egyszerűsítés után a teljes összeadó függvényei:

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

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

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

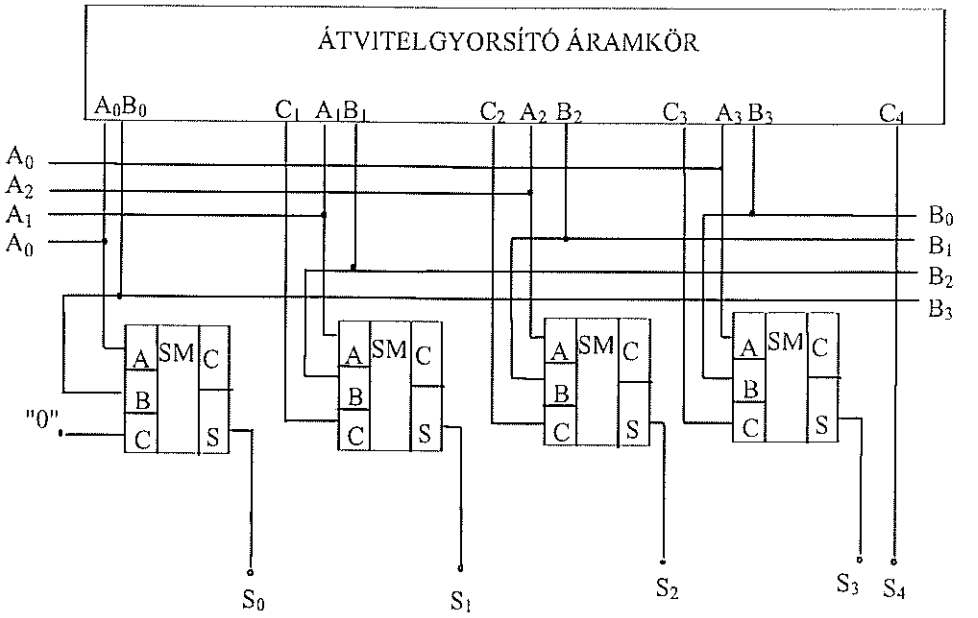
Az egy helyi értékes teljes összeadó felhasználásával készíthetünk több bites összeadót is úgy, hogy egymással párhuzamosan annyi egy helyi értékes összeadót alkalmazunk, ahány bitesek az operandusok. A **6.9.** ábra négybites teljes összeadót mutat.



6.9. ábra. Négybites teljes összeadó

A legkisebb helyi érték C_0 átvitel bemenetét a használat során 0 szintre kötjük, hiszen itt nincs előző helyi értékről jövő átvitel. A legnagyobb helyi értéken keletkező C_3 átvitel az összeg legnagyobb helyi értékű S_4 bitje. Az egy helyi értékes összeadók között az átvitel sorosan terjed, mert egy következő helyi értéknek mindig meg kell várnia az előző helyi értéken való összeadás befejezését. A gyors összeadást te-

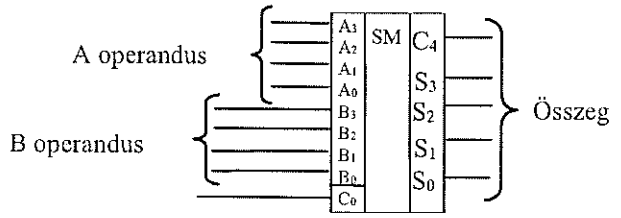
hát az átvitel soros terjedése akadályozza. Készülnek ezért olyan összeadók, amelyek **átvitelgyorsító áramkört tartalmaznak.** A bonyolult felépítésű áramkörök részletes ismertetésére nincs mód, működési elvük azonban egyszerű: az operandusok egyes bitjeiből egy kombinációs hálózat meghatározza az egyes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



6.10. ábra. Összeadó gyorsított átvitelképzéssel

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

A belső felépítéstől eltekintve az összeadókat kapcsolási rajzokon jelképi jelölésükkel ábrázoljuk. Négybites összeadó jelképi jelölése látható a **6.11.** ábrán.



6.11. ábra. Négybites összeadó jelképi jelölése

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

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

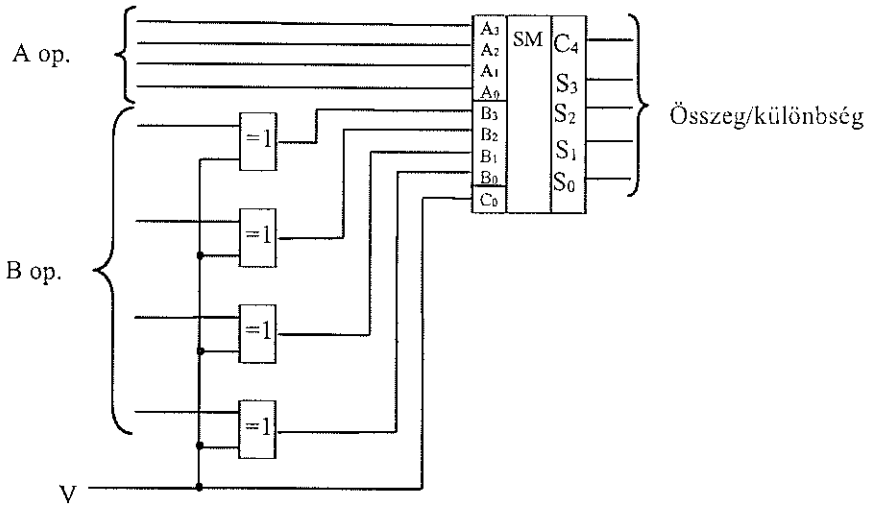
A negatív számokat kettes komplemens kódjukkal ábrázoljuk a számításokban. Ezért az előző azonosság így fogalmazható meg: két operandus különbségét úgy képezzük, hogy a kisebbítendőhöz hozzáadjuk a kivonandó kettes komplemensét. A kettes komplemens képzését már ismerjük: a szám bitenkénti negálásával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

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

$+12_{10} = 0\ 1100_2$; $-12_{10} = 1\ 0100_{2K}$.

A kivonás tehát így irható le: $S = A + B_{2K}$.

Ezen az elven működő négybites összeadó/kivonó áramkört szemléltet a **6.12.** ábra.



6.12. ábra. Bináris összeadó kivonó áramkör

Az összeadó B bemenetein lévő antivalenciakapuk vezérelhető inverterek. Az igazságtáblázatban az egyik bemeneti változót vezérlőjelnek tekintve a táblázat sorait a következőképpen értelmezhetjük:

B V F

0 0 0 a vezérlőjel 0, a kimeneten a B változó jelenik meg,

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

1 0 1 a vezérlőjel 0, a kimeneten a B változó jelenik meg,

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

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

A **6.12.** ábra antivalenciakapui V = 1 vezérlésre negálják a B operendus egyes bitjeit, tehát a kivonandó egyes komplemensét képezik. A vezérlőjel az C_0 bemenetre is be van kötve, ezért az összeadó a legkisebb helyi értéken az egyes komplemenshez hozzáad az összegzés során egyet. Így egyszerre történik meg az egyes komplemensből a kettes komplemens képzése és a kivonásnak számító összeadás. Ha V = 0, akkor nincs komplemensképzés, az áramkör összeadóként működik.

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

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

Bináris és BCD összegek. **6.1.** táblázat

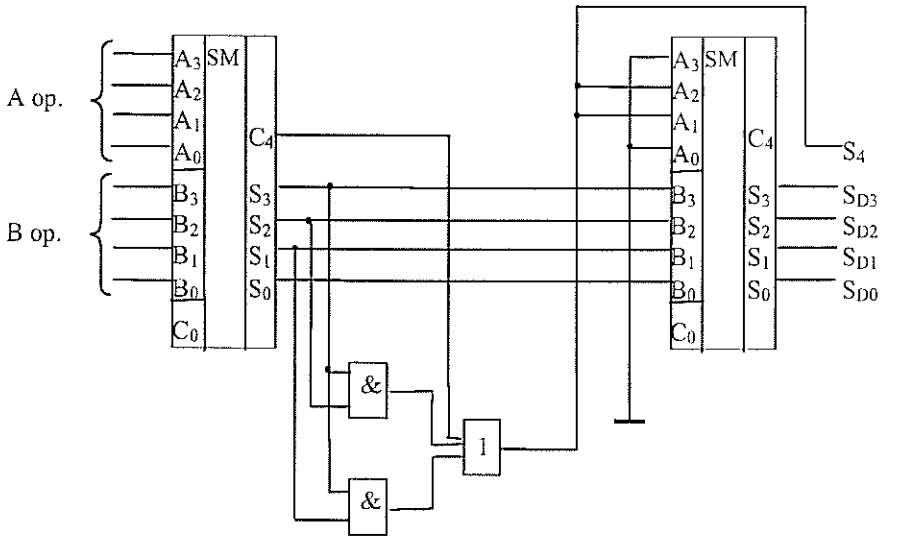
Decimális összeg	Bináris összeg					BCD összeg				
	C_4	S_3	S_2	S_1	S_0	C_4	S_3	S_2	S_1	S_0
0 (0+0)	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5 (pl. 2+3)	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0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

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

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

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

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



6.13. ábra. A BCD összeadó kapcsolási rajza

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

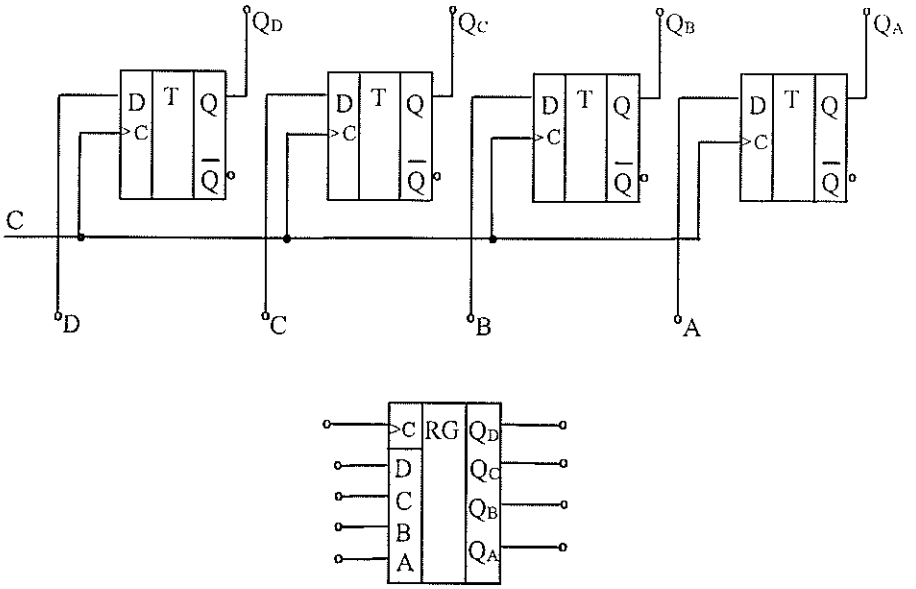
A BCD kivonó áramkör hasonló elven készíthető.

6.4. Regiszterek

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

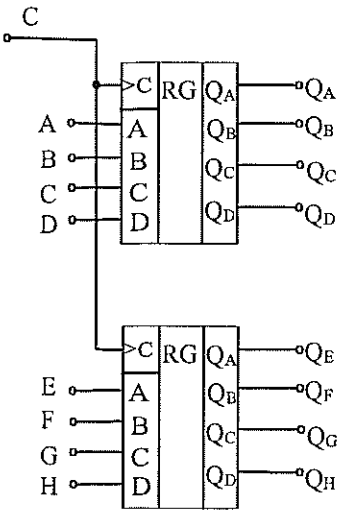
- átmeneti tárolók,
- léptetőregiszterek.

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



6.14. ábra. Pufferregiszter

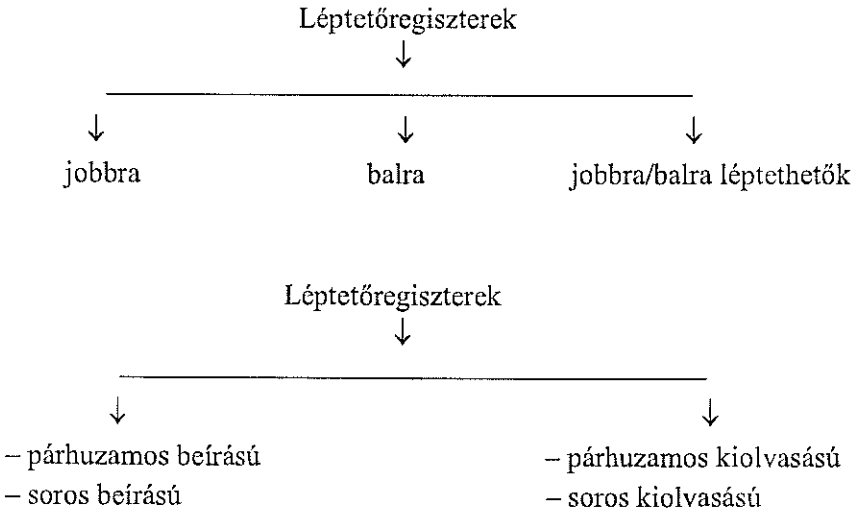
Ha a tárolandó információ több bitet tartalmaz, mint amennyit egy regiszter képes tárolni, akkor több regiszterből készítünk nagyobb kapacitásút (tárolóképességűt) kaszkádosítással. Két négybitesből összeállított 8 bites regisztert mutat a **6.15.** ábra.



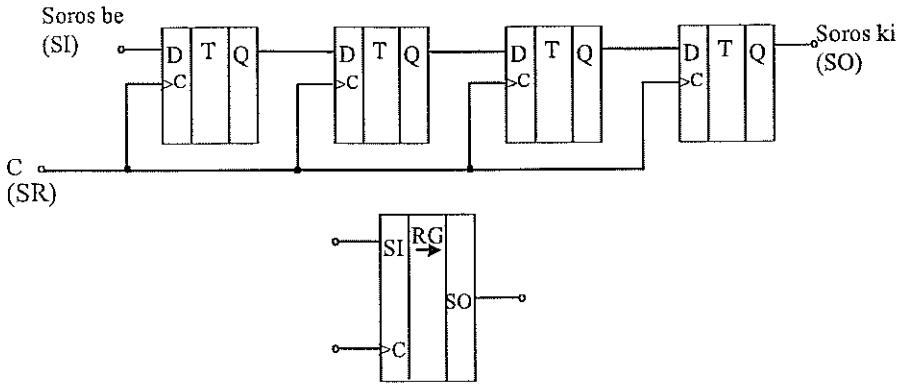
6.15. ábra. A regiszterek kapacitásának bővítése

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

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



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

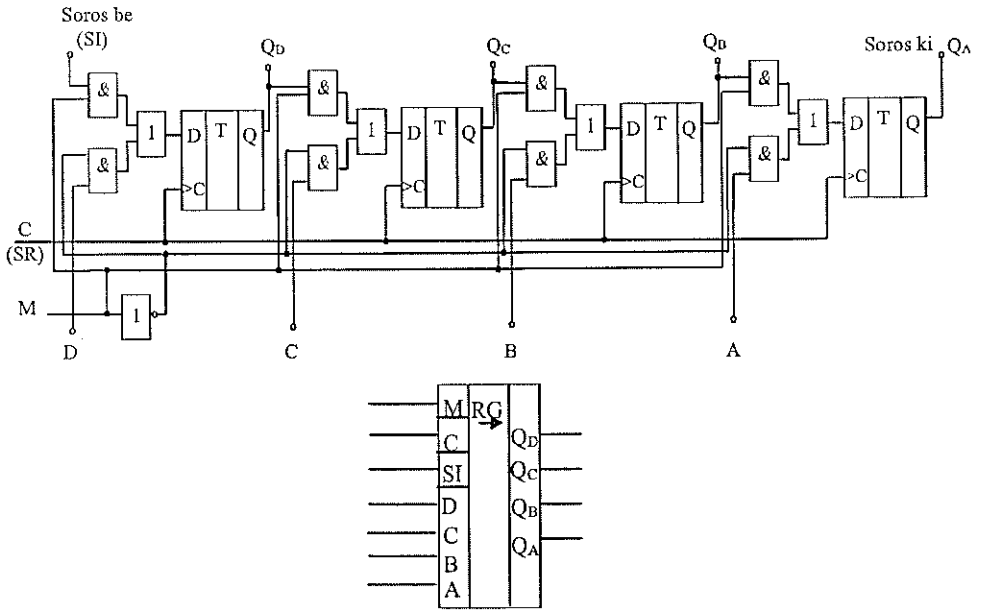


6.16. ábra. Jobbra léptethető, soros be/soros ki regiszter

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

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

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

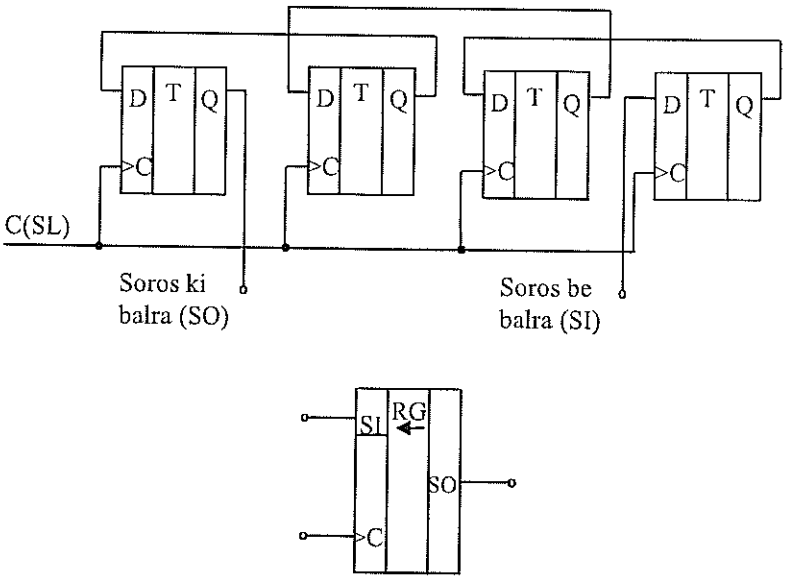


6.17. ábra. Jobbra léptethető, párhuzamos/soros be, párhuzamos/soros ki regiszter

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

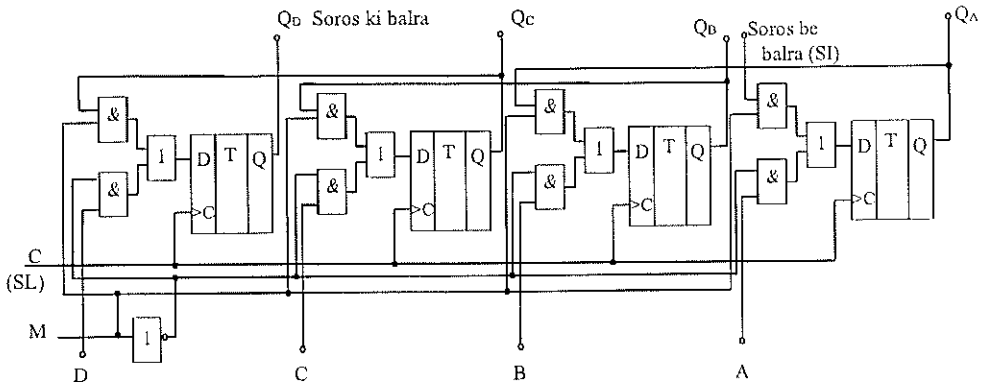
Ha a 6.17. ábrán látható kapcsolásból, a QA kimenetet kivéve, elhagyjuk a párhuzamos kimeneteket, akkor egy **soros/párhuzamos beírású, soros kiolvasású jobbra léptethető** regiszterhez jutunk. Ilyen regisztert tartalmaz pl. az SN 7494, a CD 4014.

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



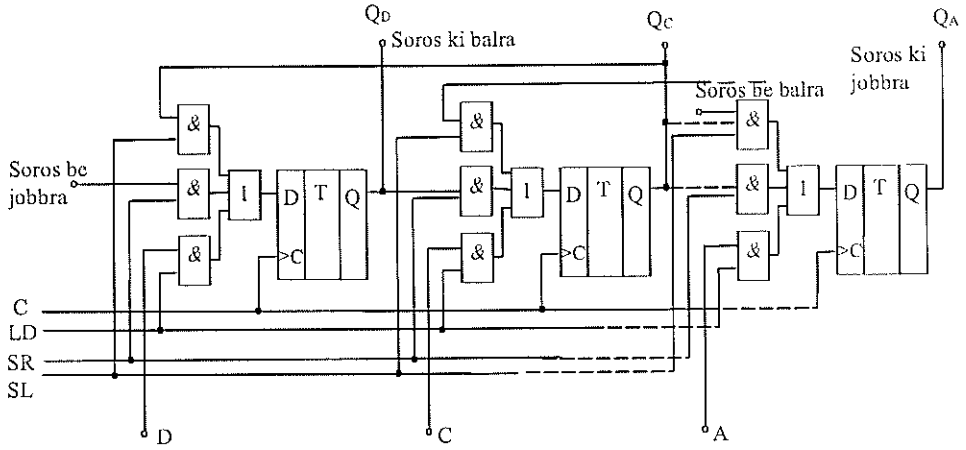
6.18. ábra. Balra léptethető soros be/soros ki regiszter

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



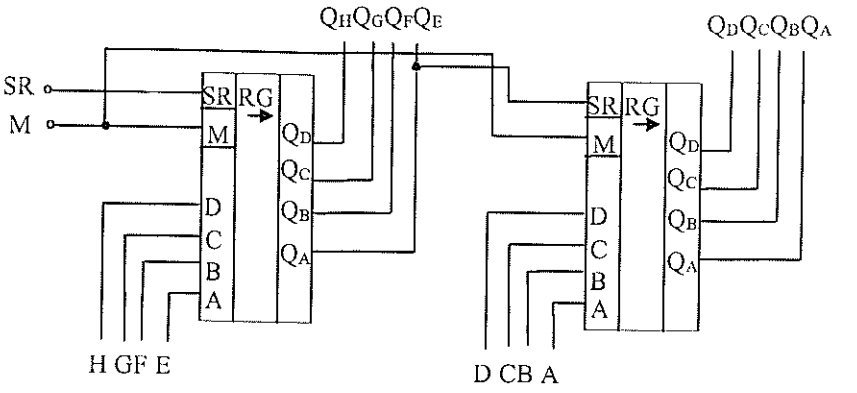
6.19. ábra. Balra léptethető, párhuzamos/soros be, párhuzamos/soros ki regiszter

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



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

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



6.21. ábra. A léptetőregiszterek kaszkádosítása

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

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

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

6.5. Számlálók

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

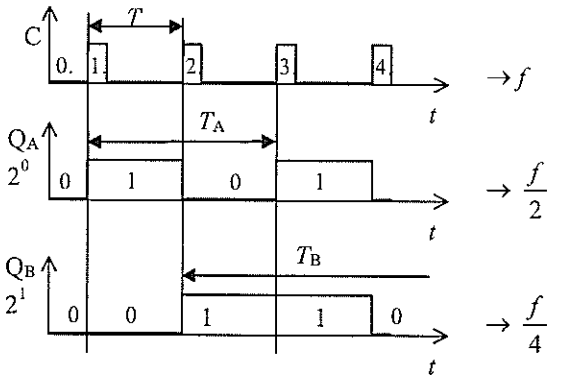
Impulzusok száma	Kimeneti állapot	
C	Q_B (2^1)	Q_A (2^0)
0	0	0 alapállapot,
1	0	1
2	1	0
3	1	1 végállapot.

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

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

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

A **6.22.** ábrán a számlálás elve idődiagramon követhető.



6.22. ábra. A számlálás elve

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

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

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

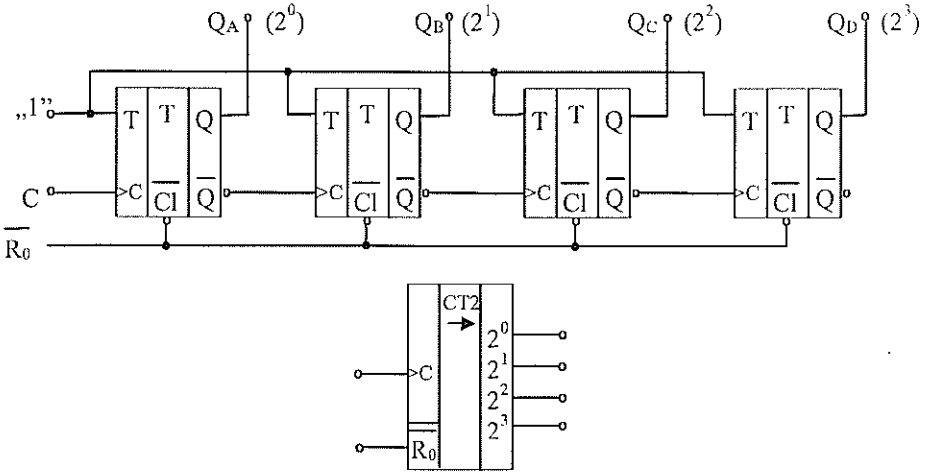
A tárolók vezérlési táblázatain végigtekintve megállapíthatjuk, hogy az élvezérelt T típusú tároló a T bemenetére adott 1 vezérlés mellett minden órajel hatására megváltoztatja kimeneti állapotát. A számláló minden helyi értékén élvezérelt T flip-flopot alkalmazva a számlálás alapelve szerint működő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

A számlálók sokféle típusát a következők szerint csoportosíthatjuk:

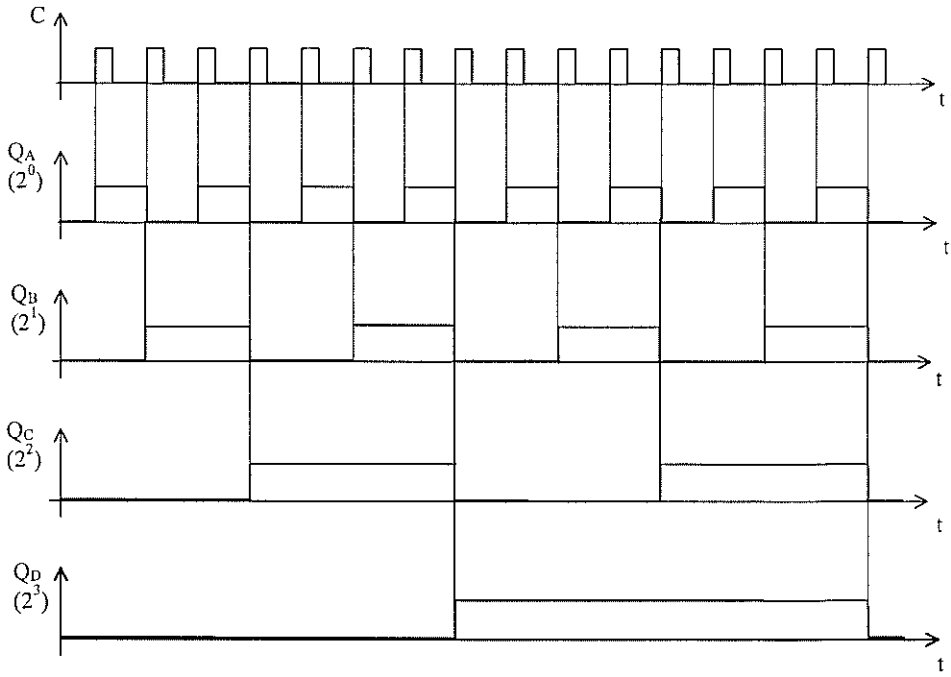
a belső felépítés szerint	a kijelzés kódja szerint	a számlálás iránya szerint	a kezdeti érték beállítása szerint
– aszinkron	– bináris	– előre (felfelé)	– párhuzamos beírású – sztatikus törlésű
– szinkron	– BCD	– hátra (lefelé) – reverzíbilis (fel/le)	

A legegyszerűbb belső felépítésű számláló az **aszinkron, bináris előreszámláló, sztatikus törlőbemenettel.** A **6.23.** ábra négybites számlálót mutat jelképi jelölésével együtt. A jelképi jelölésen szereplő CT (*Counter* – számláló) a számláló funkciójele. A mellette szereplő 2 bináris számlálót jelöl.

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



6.23. ábra. Négybites aszinkron-előreszámláló



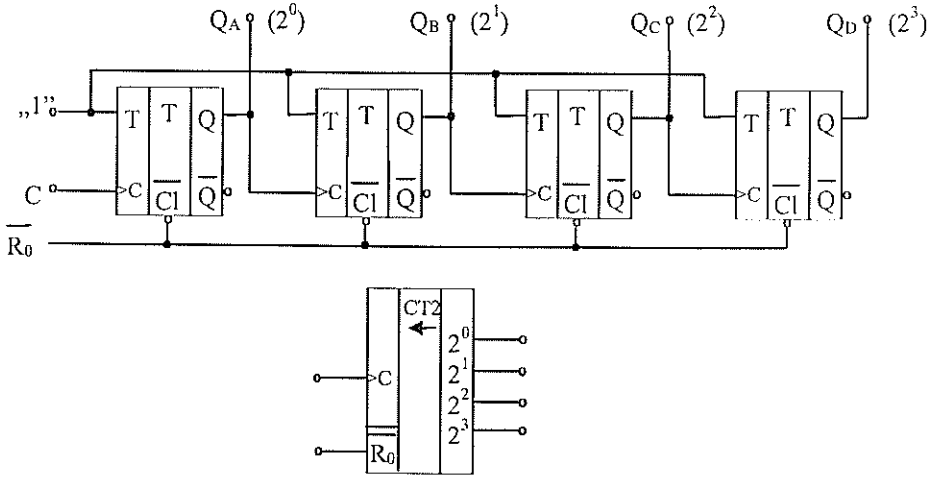
6.24. ábra. A négybites előreszámláló idődiagramja

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

számlálás közben bármikor törölhető, alapállapotba állítható a számláló. Az órajeltől független sztatikus törlést **aszinkron sztatikus törlésnek** nevezzük.

Az ismertetetthez hasonló típus az SN 7493.

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



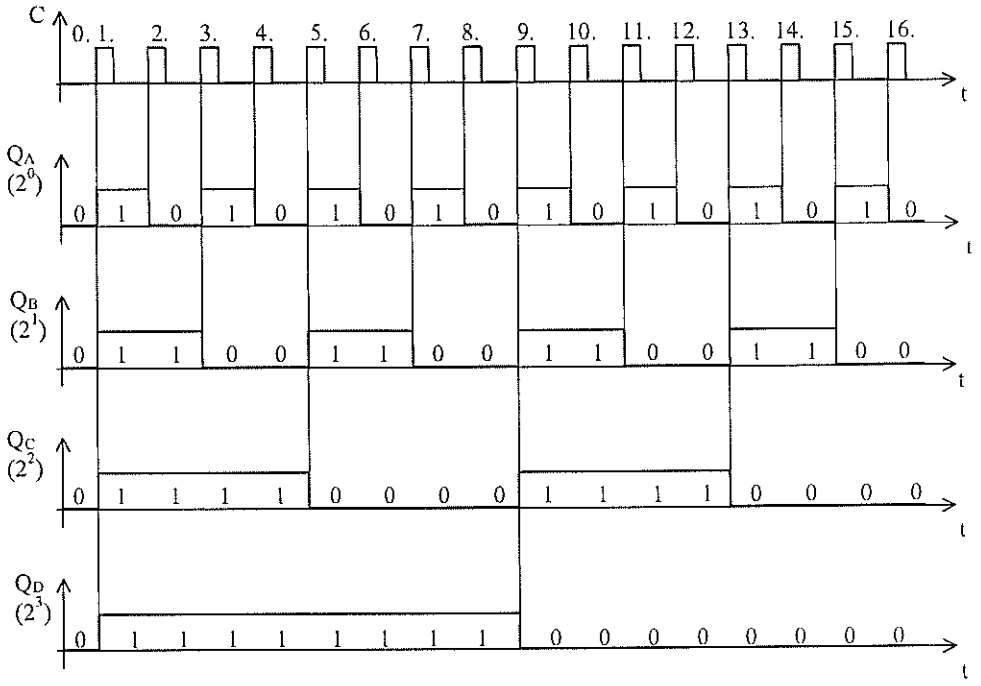
6.25. ábra. Négybites bináris visszaszámláló

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

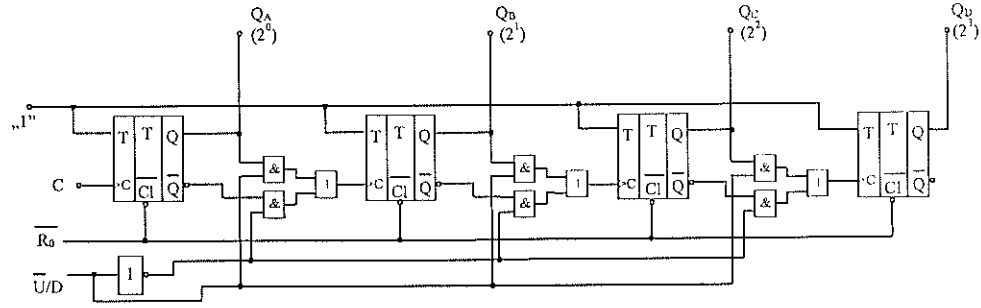
Az aszinkronszámlálók felépíthetők negatív élvezérelt tárolókból is. Ilyen felépítés mellett az első tároló invertálva kapja az órajelet, és a további tárolók vezérlése a Q kimenetekről történik.

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

A tárolók között lévő kombinációs hálózat a számlálási irányt vezérlő U/$\overline{D}$ bemenet állapotától függően engedélyezi az órajelbemenetek vezérlését az előző tároló $\overline{Q}$, vagy Q kimenetéről.



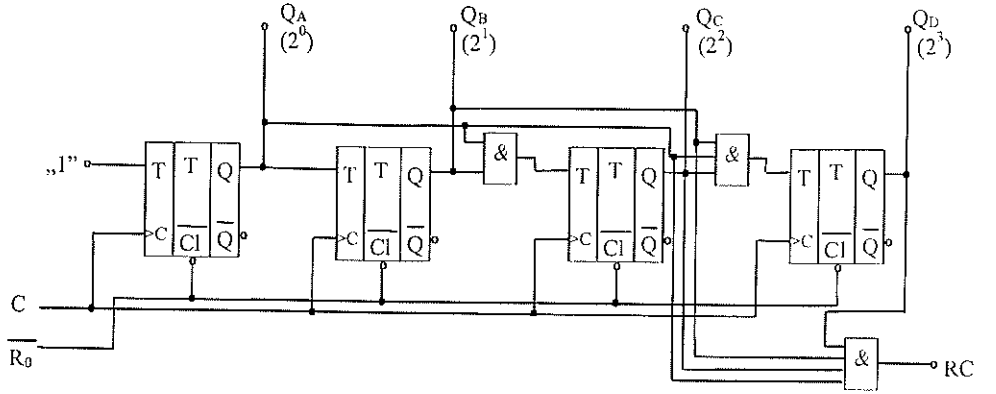
6.26. ábra. Bináris visszaszámláló idődiagramja



6.27. ábra. Négybites reverzíbilis számláló

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

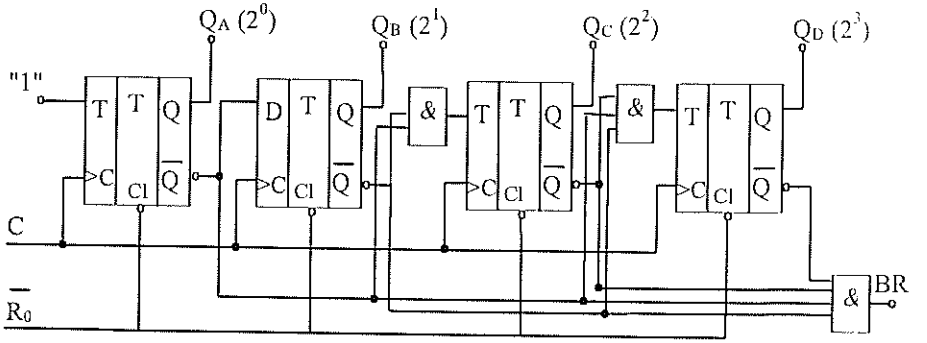
A **6.28.** ábrán egy **négybites szinkron-előreszámláló** látható **sztatikus aszinkron törlőbemenetekkel.**



6.28. ábra. Négybites szinkron-előreszámláló

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

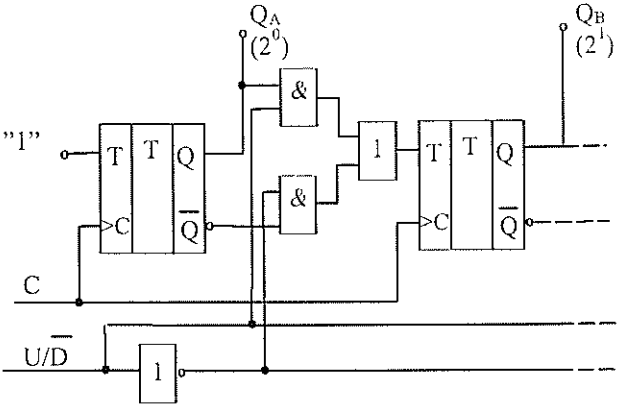
Szinkron bináris visszaszámláló hasonló elven készíthető, de a T bemeneteket vezérlő ÉS kapukat az előző tárolók negált kimenetéről kell vezérelni, mert visszaszámláláskor akkor kell egy tárolónak billeni, amikor már az összes előző tároló kimenete 0 szintű. A **6.29.** ábra egy szinkron bináris visszaszámlálót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



6.29. ábra. Szinkron bináris visszaszámláló

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

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

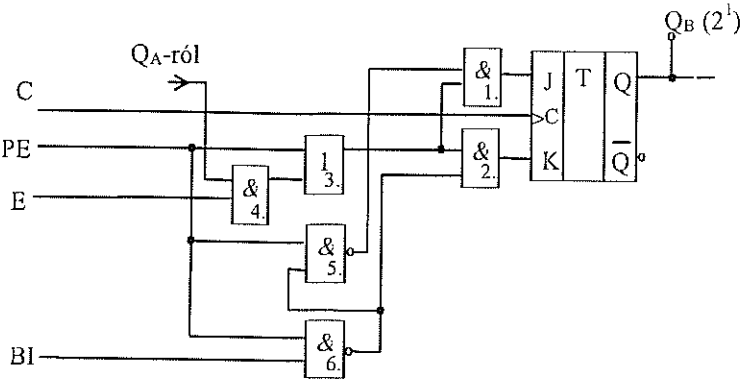


6.30. Szinkron reverzíbilis számláló tárolójának vezérlése

A szinkron reverzíbilis számlálók is rendelkeznek olyan kimenettel, amely a kaszkádosítást segíti. Ez a tokon belül a RC és BR kimenetek VAGY kapuval való közösítésével jön létre. Ez a kimenet tehát 1 szintet ad akkor is, amikor előreszámlálásnál a szinkronszámláló elérte a végállapotot, és akkor is, amikor hátraszámlálásnál alapállapotba került. A közösített kimenetet általában M/m (Maximum/minimum) jelöléssel látják el.

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

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



6.31. ábra. A párhuzamos beírás elve

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

A PE (*paralel enable*) bemenet a párhuzamos engedélyező bemenet, az E (*enable*) bemenet a számlálót engedélyezi, a BI (B-input) a B tároló párhuzamos beíró bemenete.

Számláláskor az E bemenet engedélyezett: E = 1, a párhuzamos beírás nem: PE = 0. Az 5. és 6. kapu tiltva van, kimenetük 1 szintű, ezért az 1. és 2. kapu engedélyezett. Az E = 1 engedélyezi a 4. kaput, ezért a Q_A kimenetről jövő felfutó él a 4. és 3. kapun keresztül az engedélyezett 1. és 2. kapuk kimenetére jut. A J és a K bemenetet tehát együtt vezérli a Q_A kimenet. Ilyenkor tehát a J-K tároló egy T flip-flopnak számít.

Párhuzamos beírás úgy lehetséges, ha PE = 1. Ez a vezérlés a 3. VAGY kapun keresztül engedélyezi az 1. és 2. kapukat, lehetővé téve a J és K bemenetek vezérlését és engedélyezi az 5. és 6. kapukat is.

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

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

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

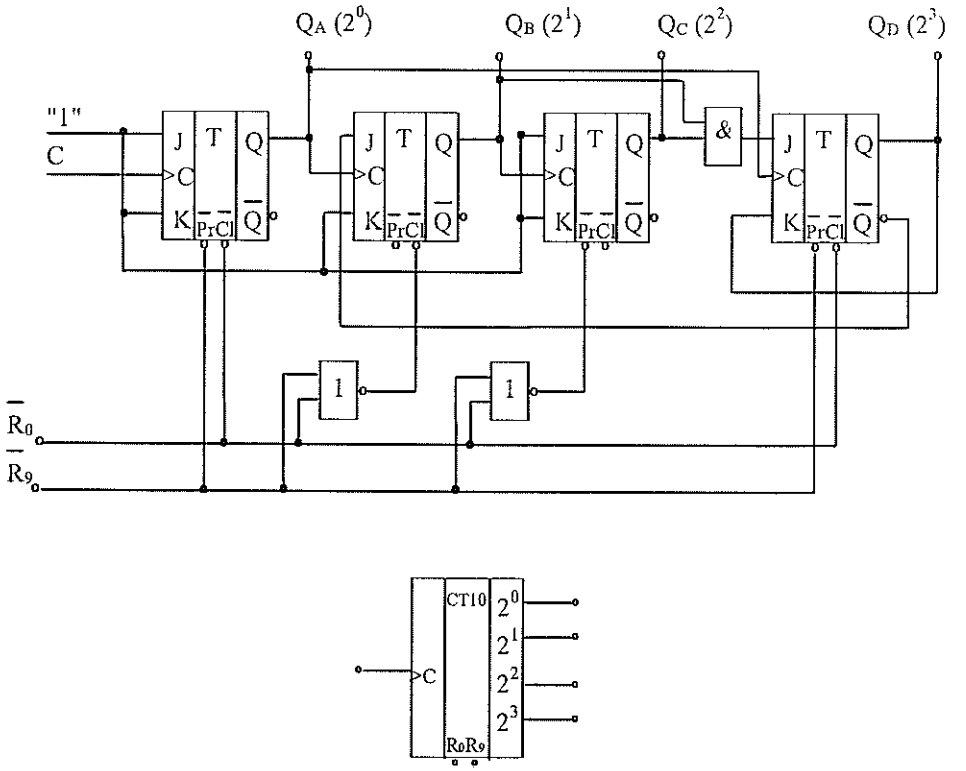
A **sztatikus törlőbemenetek** feladatát a korábbi kapcsolásokban már láttuk: a számlálás közben alapállapotba állítható a számláló. Általában az integrált áramkörökben az előző ábrákon – pl. **6.23., 6.25.** ábrák – szereplő megoldást alkalmazzák,

tehát az R0 bemenetre adott vezérlés a sztatikus clear bemenetekre hat. Ezek órajeltől függetlenek, ezért a sztatikus törlés is aszinkron. Létezik néhány olyan számláló – pl. SN 74163, CD 40163 –, amelynél a sztatikus törlés is csak az órajellel együtt hatásos. Az ilyen számlálók a **szinkron sztatikus törlésű számlálók.**

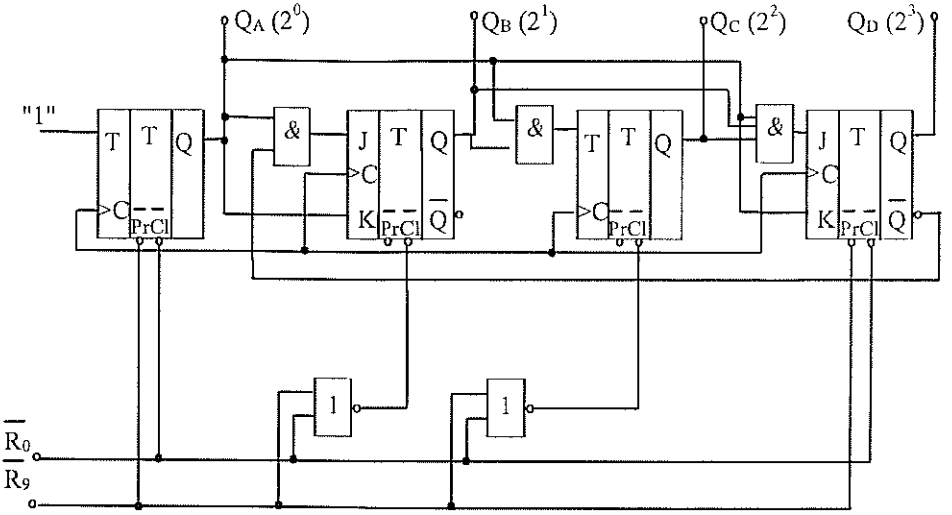
A **BCD kódú számlálók** modulusa 10, a megszámolt impulzusokat 0-tól 9-ig képesek megszámolni és négy biten, tehát BCD kódban kijelezni. Gyakran dekádszámlálónak nevezzük, mert több BCD számlálót egymás után sorolva (kaszkádosítva) minden számláló a teljes érték egy dekádját (BCD helyi értékét) számolja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

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

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



6.32. ábra. Aszinkron BCD számláló



6.33. ábra. Szinkron BCD számláló

A **6.32.** ábra kétféle sztatikus beállító-bemenetet mutat. Az R_0 a már ismert törlőbemenet, amely az összes tárolót törli. Az R_9 bemenet csak BCD számlálóknál létezik. A tárolókat a Pr sztatikus beíróbemeneteiket is felhasználva az 1001 állapotba állítja.

A szinkron- és az aszinkronszámlálók működését összehasonlítva megállapíthatjuk, hogy a szinkronszámlálóknál valamennyi kimenet egyidőben veszi fel az új értéket, az aszinkronszámlálók kimenetei viszont egymás után, időkéséssel. Az órajel hatásos éléhez képest a szinkronszámláló összes kimenetén annyi idő múlva jelenik meg az új érték, amennyi egy tároló és a bemeneteit vezérlő kombinációs hálózat együttes jelkésleltetési ideje. Pl. a **6.28.** ábra szinkronszámlálójánál a max. jelkésleltetés egy kapu és egy tároló jelkésleltetési idejének összege. Ahhoz, hogy a számláló helyesen működjön, ez idő alatt nem érkezhet újabb órajel a bemenetre, hiszen még az előzőnek megfelelő értéket sem vette fel a számláló kimenete. A kapu késleltetését pl. 10 ns-nak, a tároló késleltetését 15 ns-nak véve, az órajel periódusidejét 25 ns-nál nagyobbra kell választani. Ez frekvenciára átszámítva:

$$f_{max} = \frac{1}{2{,}5 \cdot 10^{-8}} = 4 \cdot 10^7, \quad f_{max} = 40\ \text{MHz}.$$

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

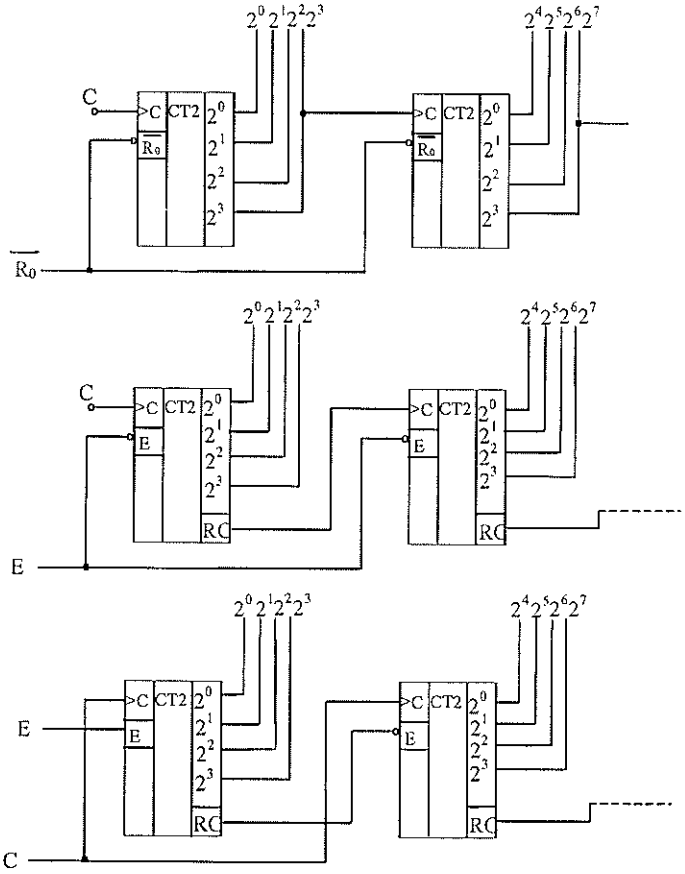
$$f_{max} = \frac{1}{6 \cdot 10^{-8}} = 1{,}66 \cdot 10^{7}, \quad f_{max} = 16{,}6 \text{ MHz}.$$

A feladatból levonható az a következtetés, hogy **az aszinkronszámlálók kisebb frekvenciáig használhatók, mint a szinkronszámlálók.**

A számlálók eredeti számlálási ciklusának módosítása két szempontból merülhet fel:

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

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



6.34. ábra. A számlálók kaszkádosítása

a) Aszinkronszámlálók kaszkádosítása, *b)* Szinkronszámlálók kaszkádosítása, *c)* Szinkronszámlálók szinkron kaszkádosítása

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

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

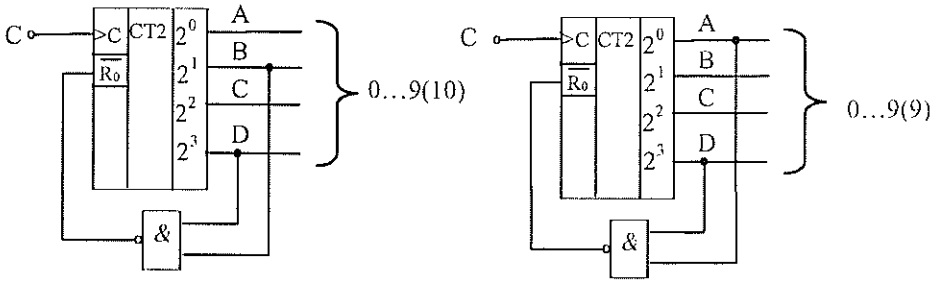
A számlálók **ciklusrövidítése** úgy valósítható meg, hogy egy áramkörrel figyeljük az előírt végállapotot, és amikor a kimenet ezt elérte, töröljük a számlálót a sztatikus törlőbemeneten keresztül. A törlés módja a számláló típusától függően lehet aszinkron és szinkron sztatikus törlés.

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

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

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

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



6.35. ábra. A számlálók ciklusrövidítése

a) Aszinkrontörlés *b)* Szinkrontörlés

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

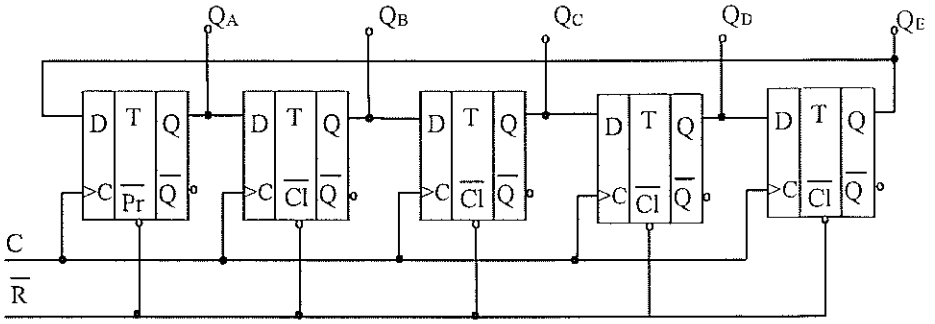
A számlálókat sok esetben **frekvenciaosztóként** használjuk. Erre az ad lehetőséget, hogy a számlálók egy-egy kimenetén az órajel leosztott értéke jelenik meg, ahogyan azt a **6.22.** ábrán már láttuk. A frekvenciaosztók bemeneti jele tehát az órajel, kimenete pedig az a számlálókimenet, amelyen a szükséges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

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

6.6. Gyűrűs számlálók

A gyűrűs számlálók léptetőregiszterek visszacsatolásával kialakított számlálóáramkörök.

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



6.36. ábra. Moduló-5 számláló

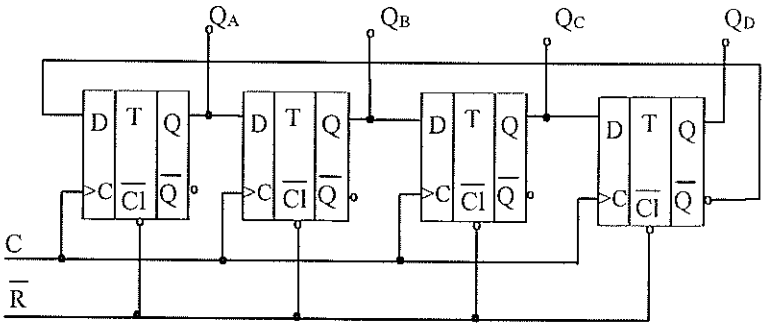
A számláló alapállapota az 10000 állapot, amit a törlőbemenettel lehet beállítani. Az órajelek hatására az első tárolóban lévő 1 végiglép a tárolón és a visszacsatolás miatt a működés ciklikusan ismétlődik. Az áramkör tehát a kimenetein az N-ből egy kódot állítja elő.

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

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

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

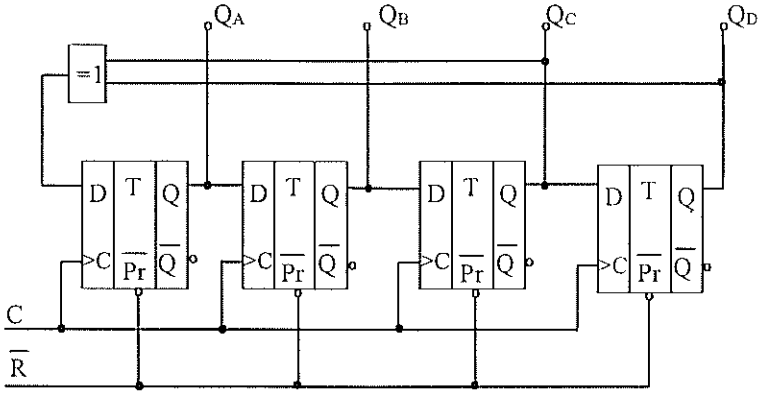
A **Johnson-számláló** egy léptetőregiszter negált soros kimenetének soros bemenetre való visszacsatolásával hozható létre, amint azt a **6.37.** ábra mutatja.



6.37. ábra. Négybites Johnson-számláló

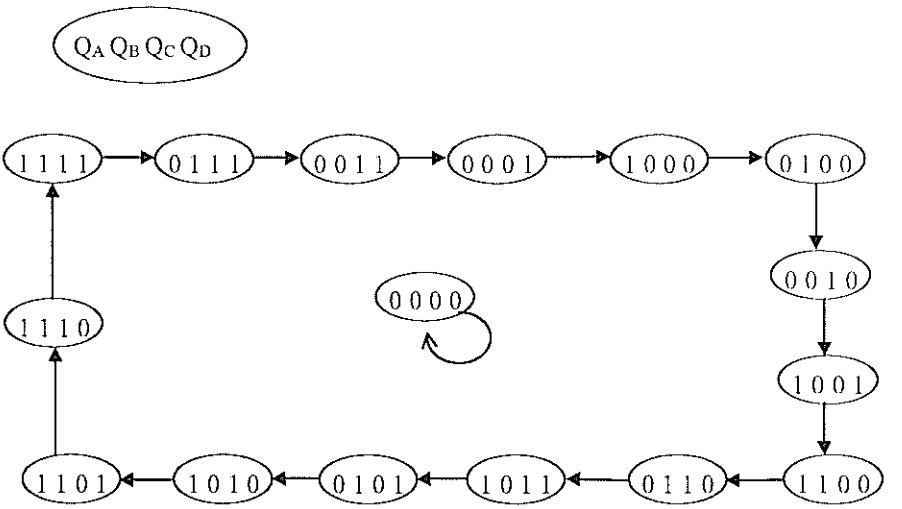
Törlés után az órajelek hatására a számláló a Johnson-kódot állítja elő a kimenetein: a ciklus első részében az A tárolótól indulva feltölti a tárolókat egyesekkel, majd a ciklus második részében nullákkal.

A **maximális ciklusú számláló** antivalenciakapuval visszacsatolt léptetőregiszter. A **6.38.** ábrán látható négybites számlálónál visszacsatolás a léptetőregiszter C és D kimeneteiről történik.



6.38. ábra. Maximális ciklusú számláló

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



6.39. ábra. A maximális ciklusú számláló állapotdiagramja

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

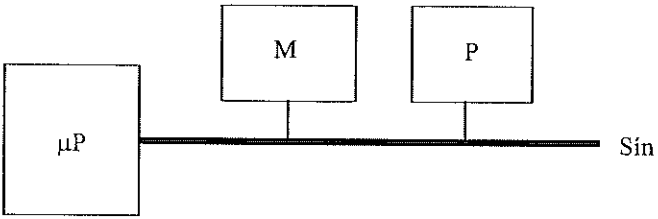
Ellenőrző kérdések

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

7. MIKROPROCESSZOROK

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

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

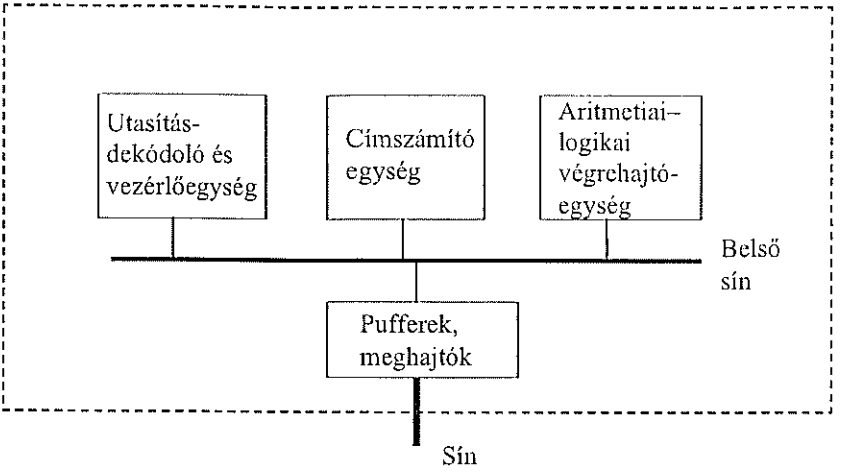


7.1. ábra. Mikroprocesszoros rendszer

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

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

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



7.2. ábra. A mikroprocesszor belső felépítése

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

A címszámító egység a memória és perifériacímek kialakítását végzi.

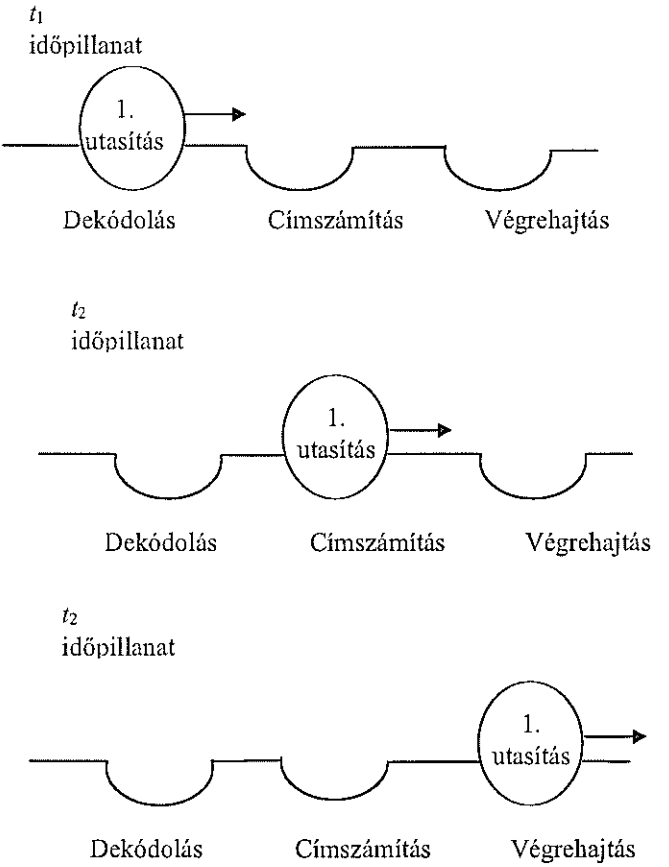
A végrehajtó egység a bináris adatokon matematikai vagy logikai műveleteket végez.

A felsorolt egységek minden mikroprocesszor-típusban megtalálhatók. Az egyes processzortípusok a belső egységek együttműködésének szervezésében, a sebesség és a teljesítménynövelését szolgáló rendszertechnikai kialakításban térnek el egymástól.

A mikroprocesszorok teljesítménye az elmúlt 10 évben 900%-kal növekedett. Ezt a hihetetlen teljesítménynövekedést egyrészt az órajel frekvenciájának emelése, másrészt a teljesítményt növelő rendszertechnikai változások eredményezték. A processzorok teljesítményét növelő és a jelenleg korszerűnek számító processzorokban alkalmazott rendszertechnikai megoldásokat a következő alfejezetekben tárgyaljuk.

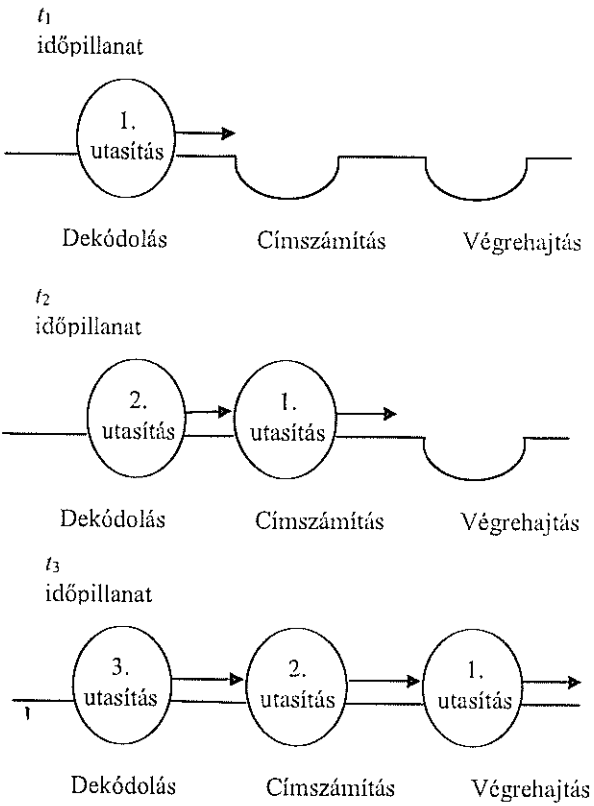
7.1. Pipe-line elv

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



7.3. ábra. Utasítás végrehajtás pipe-line nélkül

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



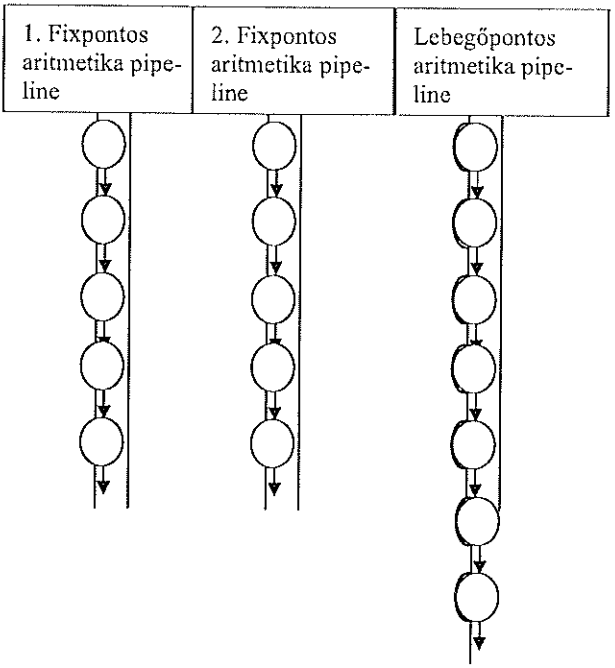
7.4. ábra. Pipe-line utasítás-végrehajtás

7.2. Lebegőpontos aritmetika, szuper-skalár felépítés

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

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

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



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

7.3. Cache alkalmazása

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

7.4. Virtuális tárkezelés

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

7.5. Multitask rendszer

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

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

Ellenőrző kérdések

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