7. hét Sorrendi hálózatok építőelemei II.

7.1. Bevezetés

Tulajdonképpen a szinkron sorrendi hálózatok építése és felhasználása nagyon elterjedt a gyakorlatban. Több minden más mellett ilyen egységekből épül fel a CPU is, melyről a későbbiekben lesz szó részletesen. Elöljáróban számunkra most csak annyi fontos, hogy a soron következő elemeket már fel tudjuk használni a mérnökinformatikusok munkaeszköze: számítógép működésének jobb megértésében. Ezek az ismeretek később abban is segíteni fognak, hogy nem csak megértsük a számítógép működését, hanem tervezni is tudjunk.

A most következő fejezetben 3 különböző egységről lesz szó:

1) regiszterek
2) számlálók
3) memóriák

7.2. Regiszter

A regiszterek funkciója az **<u>átmeneti tárolás.</u>** Ezért azonos órajelű D tárolókból épül fel. Azért van szükség több tárolóra, mert több bites adatok átmeneti tárolására alkalmas. Ezek az adatok egyrészt lehetnek vezérlő információk, másrészt a regiszterek alkalmasak műveletek operandusainak és eredményének tárolása is.

Mivel a regiszter azonos órajelű D tárolókból épül fel, ezeknek a tárolóknak a működését össze kell hangolni. Ezt egyrészt megteszi a minden tárolón azonos órajel, másrészt a regiszter ún léptetése.

Léptetés alatt azt értjük, hogy a tárolók kimenete egy másik tároló bemenetére csatlakozik, ezért az órajel hatására az információ az egyik tárolóból a másikba képes íródni.

Az ilyen típusú regisztereket ún. léptető, vagy shiftregisztereknek nevezzük.

A léptetés történhet

- jobbra,
- balra,
- mindkét irányba.

A bemeneti jel tehát áthaladva a láncon késleltetve, de változatlanul jelenik meg a kimeneten. A léptetőregiszter bináris jelek tárolására szolgál és minden egyes flip flop, amely a regisztert alkotja, egy bit információ tárolására alkalmas. A léptetőregiszterekben a megfelelő működés érdekében, hogy minden léptetési parancsra egy és csakis egy léptetés történjen, feltétlenül órajelvezérelt flip-flopokat

kell alkalmazni. Tudjuk azt is, hogy közös vezérlőjelekkel vannak ellátva, így valamennyi flip-flop egyszerre azonos műveletet végez.

A léptetőregiszterek esetén a soros és párhuzamos beírás és kiolvasás, valamint a kétféle léptetési irány lehetőségének variációival sokféle típus állítható elő. Az ábra az egy irányban (jobbra) léptethető, soros beírású és kiolvasású 4 bites áramkör kapcsolását szemlélteti. Ha az áramkör minden flip-flopját kimeneti kivezetéssel látjuk el, a léptetőregiszter párhuzamos kiolvasásra is alkalmassá válik.

Az áramkör működésekor a soros adatbemenetre érkező 1 bites jel csak egy órajel hatására kerül az első D tároló kimenetére. Minden egyes órajel hatására a következő tároló kimenetén jelenik meg, tehát négy órajel hatására kerül az áramkör kimenetére.

Használjuk fel a flip-flopok statikus beíró és törlő bemeneteit is! Így alkalmassá válik a léptetőregiszter párhuzamos beírásra is. Az ábra egy 4 bites soros/párhuzamos kiolvasású és beírású léptetőregiszter kapcsolási rajzát mutatja be.

Az áramkör működésekor a párhuzamos beírás és a léptetés funkcióját az ÉS-VAGY kapuk választják szét. A beírás előtt szükséges törlés a CLR (clear: törlés) bemeneten keresztül történik logikai 0 szinttel. Beírni az S (set: beírás) bemenetről logikai 0 szinttel lehet.

Ha az üzemmód vezérlő (angolul: MC - Mode Control) bemenetén logikai 0 szint van, a kettős ÉS kapucsoportok kapui közül azok vannak engedélyezve, amelyek a párhuzamos bemenetről veszik az információt. Ha a vezérlőbemenet logikai 1 es szinten van, akkor az ÉS kapucsoportok kapui közül azok vannak engedélyezve, amelyek a szomszédos flip-flop kimenetére csatlakoznak. A bemenő jel a soros bemenetről érkezik.

Összegezve a vezérléseket:

- Párhuzamos beírás MC = 0 esetén lehetséges
- Soros beírás MC = 1 esetén lehetséges

Ne felejtsük el, hogy bármelyik működés csak órajel hatására jön létre.

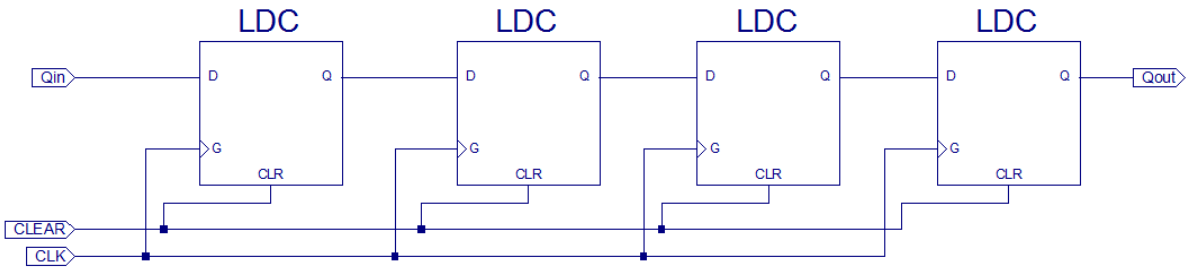
A léptetőregiszter (shift- vagy helyiérték toló regiszter) flip-flopok olyan lánca, amely lehetővé teszi, hogy a bemenetére adott információ minden egyes órajel hatására egy flip-floppal tovább lépjen. A léptetés irányának a megadására a jobbra, illetve balra megjelöléseket alkalmazzák.

Soros beírásnál és kiolvasásnál a regiszter első és utolsó flip-flopjához lehet hozzáférni. Ebben az esetben szükséges, hogy az információt a regiszterben léptetni lehessen. Ezeket a regisztereket a léptetőregisztereknek nevezzük.

Párhuzamos beírásnál és kiolvasásnál az információt a regiszter minden flip-flopjába egyszerre írják be, illetve onnan egyszerre olvassák ki. Mivel ezeknél a regisztereknél léptetés nem szükséges, a

regiszter csak tárolási feladatra alkalmas. Ezeket a típusokat átmeneti tároló vagy közbenső tároló (puffer) regisztereknek nevezik.

7.2.1. Jobbra léptető shiftregiszter megvalósítása

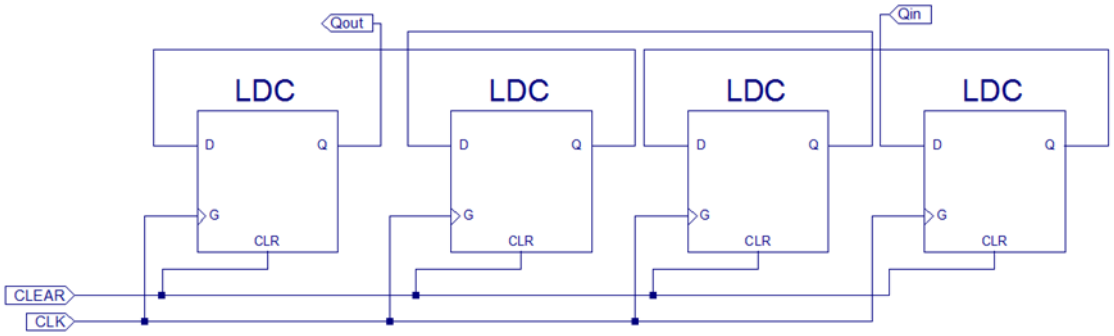


1. ábra A jobbra léptető regiszter

```
Process
Begin
   if (clear = '1') then
      q <= "0000";
   elsif (clk`event and clk = '1') then
      q(2 downto 0) <= q(3 downto 1);
      q(3) <= qin;
   end if;
end;
```

2. ábra A jobbra léptető regiszter kódja

7.2.2. Balra léptető shift regiszter megvalósítása



3. ábra Léptetőregiszter balra

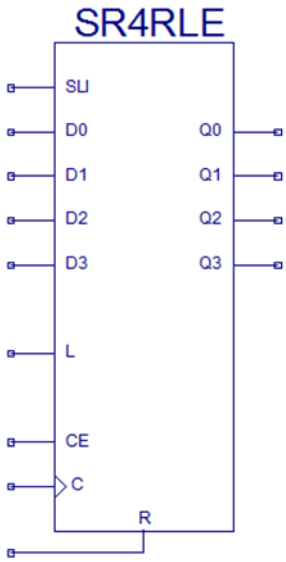
```
Process
Begin
  if (clear = ’1’) then
            q <= ”0000”;
  elsif (clk`event and clk = ’1’) then
            q(3 downto 1) <= q(2 downto 0);
            q(0) <= qin;
end if;
end;
```

4. ábra A balra léptető regiszter kódja

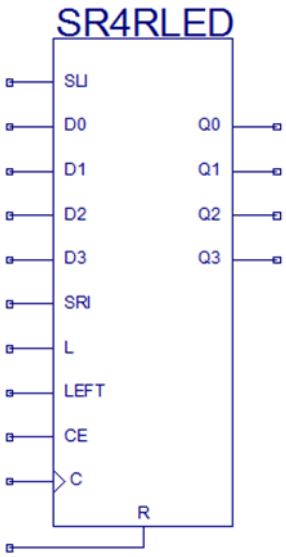
7.2.3. Mindkét irányba léptető shift regiszter megvalósítása

A jobbra balra léptetés esetén párhuzamos be- és kimenet, és/vagy soros be- kimenet működik. Ezért a regiszter alkalmas soros/párhuzamos átalakításra.

Az alábbi két rajzon két léptetőregiszter látható,



5. ábra Léptetőregiszter 1



6. ábra Léptetőregiszter 2.

ahol

SLI: baloldali soros bemenet

SRI: jobboldali soros bemenet

D: párhuzamos bemenetek

Q: párhuzamos kimenetek

CE: órajel engedélyező bemenet

C: órajel

L: beírás engedélyezés

LEFT: balra/jobbra léptetés

R: szinkron törlés

A regiszter portjait természetesen állapottáblával is felírhatjuk.

Inputs								**Outputs**		
R	**L**	**CE**	**LEFT**	**SLI**	**SRI**	**D3 : D0**	**C**	**Q0**	**Q3**	**Q2 : Q1**
1	X	X	X	X	X	X	↑	0	0	0
0	1	X	X	X	X	D3 : D0	↑	D0	D3	Dn
0	0	0	X	X	X	X	X	No Change	No Change	No Change
0	0	1	1	SLI	X	X	↑	SLI	q2	qn-1
0	0	1	0	X	SRI	X	↑	q1	SRI	qn+1
qn-1 or qn+1 = state of referenced output one setup time prior to active clock transition										

7. ábraA Shift regiszter állapottáblája

7.2.4. A regiszter felhasználása

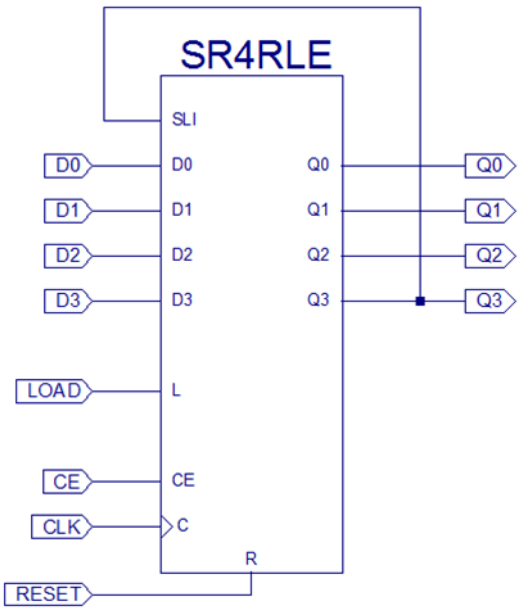
A regisztereket több dologra fel lehet használni. Készíthetünk belőle átmeneti adattárolót, és speciális számlálót, valamint felhasználhatjuk a véletlen számok előállítása során is.

7.2.4.1. Adat átmeneti tárolása

A regiszter alap felhasználásának tekinthető a d tárolókon keresztül mozgatott adatok átmeneti tárolás és továbbítása, melyről szó volt az előzőekben.

7.2.4.2. A gyűrűs számláló

A gyűrűs számláló speciális számlálónak tekinthető, hiszen egy regiszterrel valósítjuk meg, amely nem számol valójában, hanem működése során adatot mozgat. A gyűrűs számláló megvalósítása során az előző regisztert alakítjuk át egy hurok segítségével, melyben a Q3kimenetetet és az SLI bemenetet kapcsoljuk össze.

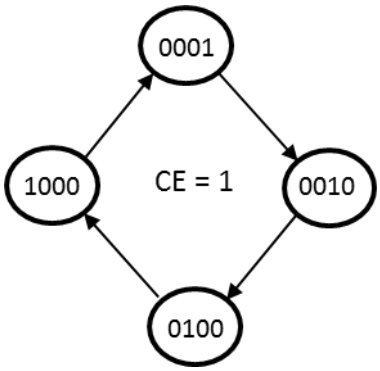


8. ábra A gyűrűs számláló megvalósítása

A működtetés során a LOAD bemenettel a D3-D0 bemeneteket 0001 alaphelyzetbe állítjuk. Az órajel engedélyezése után minden órajel ciklusban az 1-es továbblép a következő helyi értékre.A visszacsatolás miatt 4 ciklus után újra kezdődik a folyamat. A 2n (16) lehetséges állapotból csak n (4) valósul meg (12 tiltott állapot).

Q_3	Q_2	Q_1	Q_0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0
0	0	0	1
0	0	1	0
...			

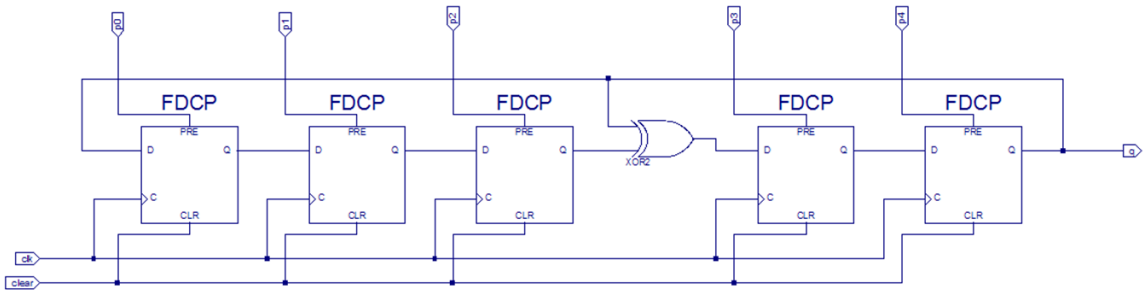
9. ábra a gyűrűs számláló állapottáblája



10. ábra A gyűrűs számláló működése

7.2.4.3. Véletlen számgenerátor

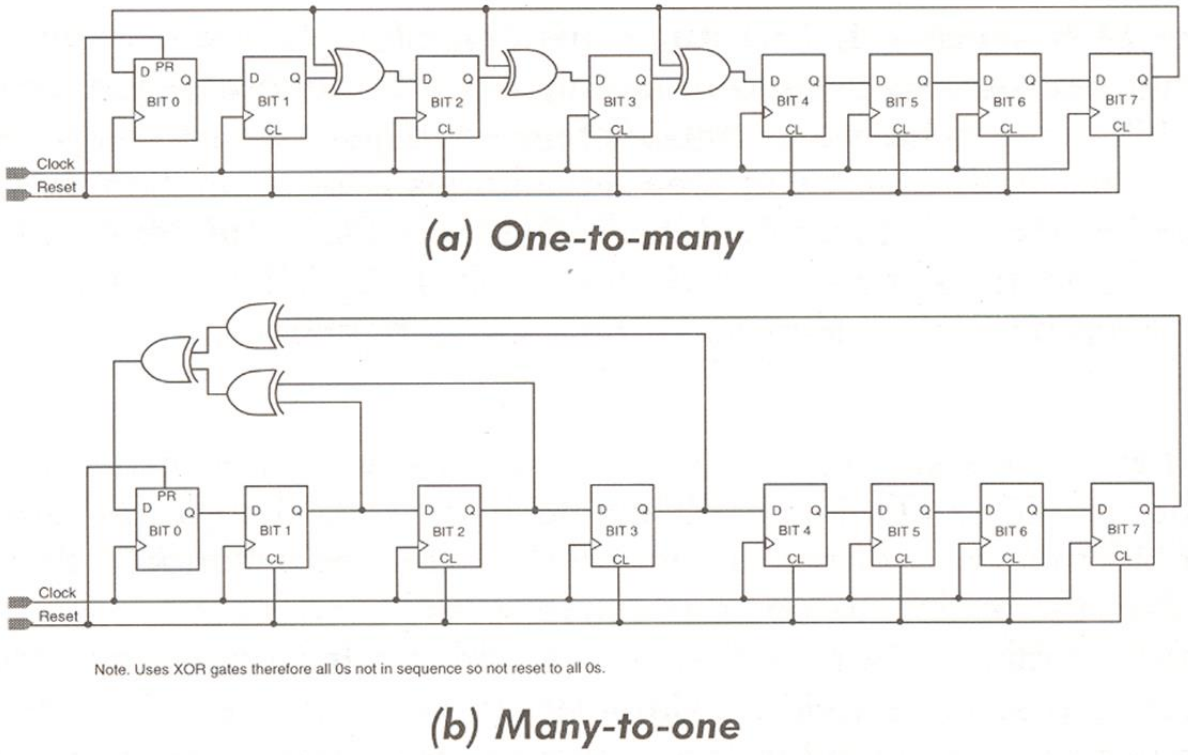
Az álvéletlen szám generátor léptetőregiszterrel (LFSR (Linear Feedback Shift Register) úgy történik, hogy, ha a regiszterek tartalma 0, ez az állapot marad. Nem nulla kezdőállapot után azonban véges hosszúságú periodikus jelet állít elő a kimeneten. A periódus hossz maximum 2n-1 (n a regiszterek száma).



11. ábra Az álvéletlenszám generátora

A generátornak két fajtáját tudjuk megkülönböztetni:

1) Egy irányból készítünk sokat
2) Sokból készítünk keveset



12. ábra A véletlenszámgenerátor

n	XNOR from	n	XNOR from	n	XNOR from	n	XNOR from
3	3,2	45	45,44,42,41	87	87,74	129	129,124
4	4,3	46	46,45,26,25	88	88,87,17,16	130	130,127
5	5,3	47	47,42	89	89,51	131	131,130,84,83
6	6,5	48	48,47,21,20	90	90,89,72,71	132	132,103
7	7,6	49	49,40	91	91,90,8,7	133	133,132,82,81
8	8,6,5,4	50	50,49,24,23	92	92,91,80,79	134	134,77
9	9,5	51	51,50,36,35	93	93,91	135	135,124
10	10,7	52	52,49	94	94,73	136	136,135,11,10
11	11,9	53	53,52,38,37	95	95,84	137	137,116
12	12,6,4,1	54	54,53,18,17	96	96,94,49,47	138	138,137,131,130
13	13,4,3,1	55	55,31	97	97,91	139	139,136,134,131
14	14,5,3,1	56	56,55,35,34	98	98,87	140	140,111
15	15,14	57	57,50	99	99,97,54,52	141	141,140,110,109
16	16,15,13,4	58	58,39	100	100,63	142	142,121
17	17,14	59	59,58,38,37	101	101,100,95,94	143	143,142,123,122
18	18,11	60	60,59	102	102,101,36,35	144	144,143,75,74
19	19,6,2,1	61	61,60,46,45	103	103,94	145	145,93
20	20,17	62	62,61,6,5	104	104,103,94,93	146	146,145,87,86
21	21,19	63	63,62	105	105,89	147	147,146,110,109
22	22,21	64	64,63,61,60	106	106,91	148	148,121

13. ábraA z álvéletlenszám generátor működése

Az álvéletlenszám generátor felhasználható:

- Bitminta generálásra,
- Titkosításra,
- Hibavédelemre.

7.3. Számláló

7.3.1. Összeadó

Az összeadás definíciója

Olyan áramkörök, amelyek a bemeneteikre kerülő két operandust bitenként összeadják, figyelembe véve az egyes helyi értékeken keletkező átvitelt is, és az eredményt a keletkezett átvitellel együtt megjelenítik a kimenetükön. Ennek leggyakrabban alkalmazott formája a teljes összeadó.

A teljes összeadó megértéséhez fontos a számláló elvének megértése is.

A számlálás során, ha decimálisan írjuk a számokat, 10 darab számjeggyel írjuk le az összes lehetséges mennyiséget. A használt számjegyek 0,1,2,3,4, 5, 6, 7, 8, 9, .

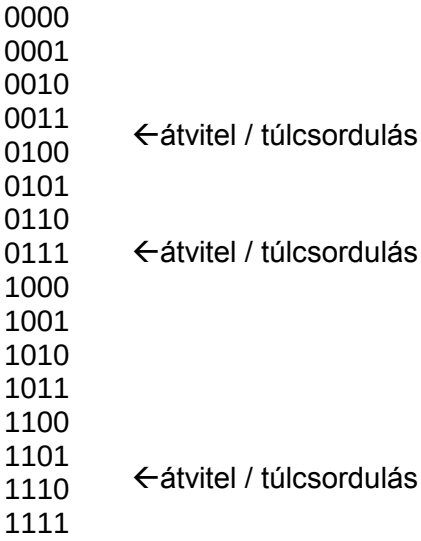
9 után azonban elfogynak a számjegyek. Mi ilyenkor a teendő?

A rendszerben ilyenkor túlcsordulás keletkezik: egyet lépünk a helyiértéken, és újra kezdjük a számlálást.

01
02
03
04
05
06
07
08
09
10 ←átvitel / túlcsordulás
11
12
…

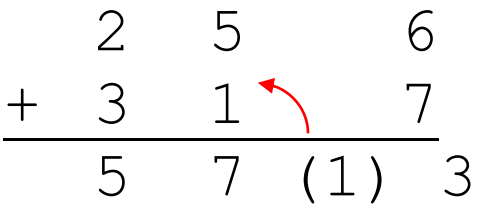
14. ábra A decimális számláló túlcsordulása

Ugyanez történik bináris rendszerben is, azzal a különbséggel, hogy mivel itt csak két számjegyünk van, ezért hamarabb fog jelentkezni a túlcsordulás.



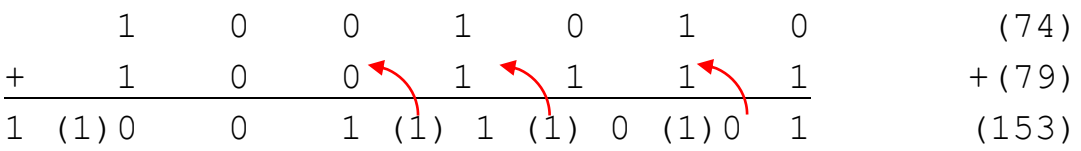
15. ábra a bináris számláló túlcsordulása

Az összeadás során is ezzel a túlcsordulással kell számolni. Ezért a decimális esetben a papíron végzett összeadásnál, jobbról balra haladva sorra összeadjuk az egyes számjegyeket. Ahol kilencnél nagyobb eredményt kapunk, ott hozzáadjuk a maradék egyest az eggyel nagyobb helyiértékű számjegyekhez.



16. ábra Összeadás decimális szám esetén

A bináris esetben pedig ugyanígy járunk el, csak a felhasználható számjegyek 0 és 1. Ahol 1-nél nagyobb eredményt kapunk, ott hozzáadjuk a maradék egyest az eggyel nagyobb helyiértékű számjegyekhez.



17. ábra Összeadás kettes számrendszerben

A **teljes összeadónak** 3 bemenete van, amelyekre a két összeadandót és az előző helyiértéken keletkezett átvitelt vezetjük. Ilyen 1 bites modulok felhasználásával készíthetünk több bites összeadó áramköröket.

Mivel akettes számrendszerben történő összeadéás esetén az átviteleket is figyelembe véve:

Összeadás	Példa
0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = 10	1011_2 $+\quad 11_2$ 1110_2

Kivonás	Példa
0 − 0 = 0 0 − 1 = −1 1 − 0 = 1 1 − 1 = 0	1011_2 $-\quad 111_2$ 100_2

Szorzás	Példa
0 . 0 = 0 0 . 1 = 0 1 . 0 = 0 1 . 1 = 1	1010_2 $\cdot\quad 11_2$ 11110_2

Osztás	Példa
0 / 0 = n.def. 0 / 1 = 0 1 / 0 = n.def. 1 / 1 = 1	1010_2 $/\quad 10_2$ 101_2

18. ábra A műveletek kettes számrendszerben

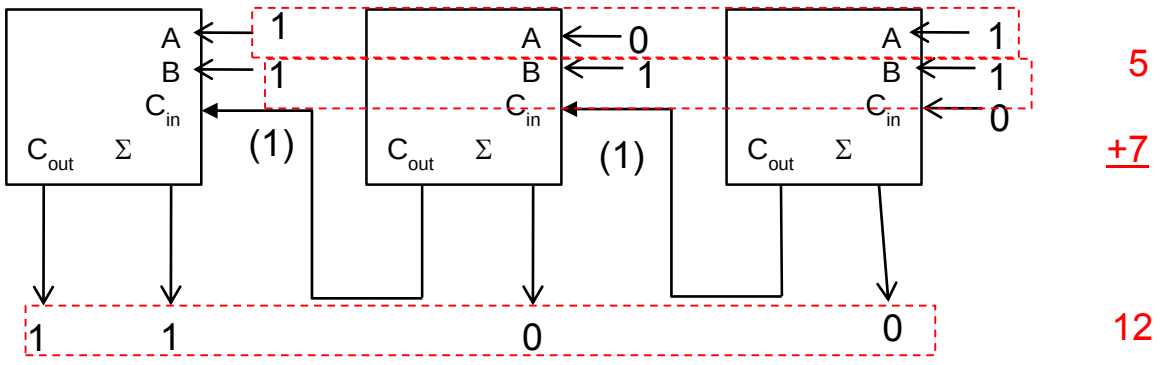
Ezért a rendszer igazságtáblázata a következőképpen írható fel:

C_{in}	A	B	Σ	C_{out}
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

19. ábra Az összeadó igazságtáblázata

Az összeadó (Áramköri felépítése az alábbi ábrán látható.) tehát

- Két bináris szám egy-egy bitjét adja össze
- Figyelembe veszi az eggyel kisebb helyiértékről érkező átvitelt.
- Ha nála is keletkezik átvitel, akkor továbbítja azt.
- Több egybites összeadót összekapcsolva két bármilyen hosszú bináris számot összeadhatunk
- Kettes komplemens kóddal kivonóként is használható, az utolsó átvitel használata nélkül



20. ábra Az teljes összeadó

A teljes összeadó működését tekintve kétféle lehet: soros vagy párhuzamos.

Soros összeadó

Az egyes helyiértékeken lévő bitek összeadása időben egymás után történik, kezdve a legkisebb helyiértékkel. Így csak egy 1 bites teljes összeadó áramkörre, regiszterekre és egy 1 bites késleltető tárolóra van szükség. Ezzel a módszerrel viszonylag hosszú idő szükséges egy nagyobb bitszámú művelet elvégzéséhez.

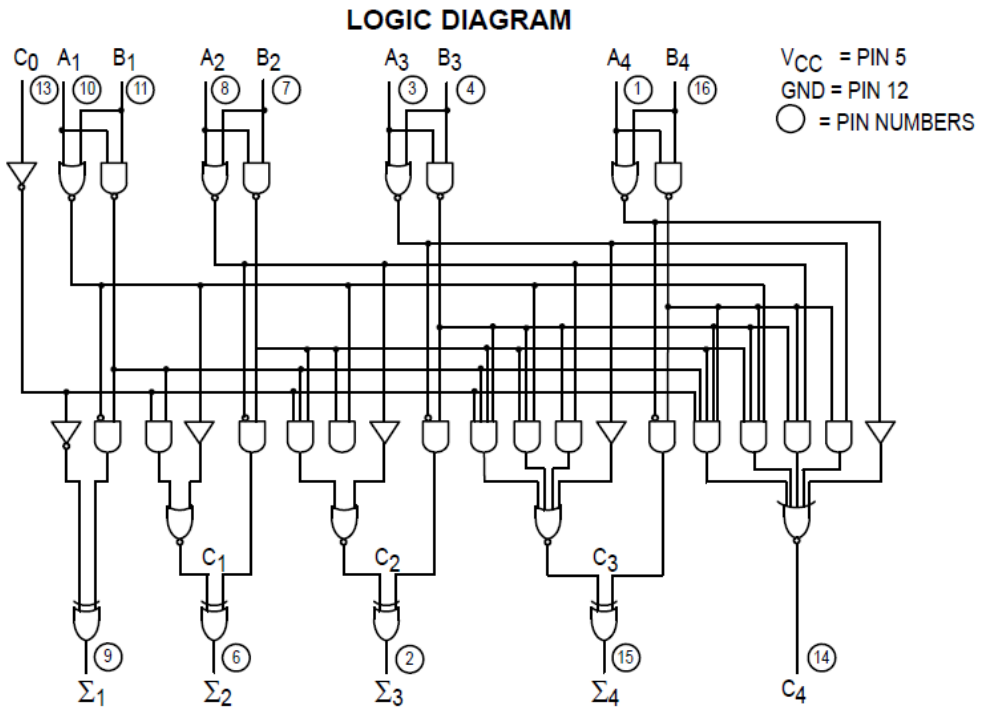
Párhuzamos összeadó

Az egyes helyiértékeken lévő bitek összeadása azonos időben történik, de természetesen az átvitelbitek megjelenéséig szükség van valamekkora továbbterjedési időre, vagy valamilyen áramkörre, amely az átvitel bitek létrehozását gyorsítja.

Példa

Jó példa az összeadóra a **74LS8 amely egy 4 bites teljes összeadó.**

21. ábra A négy bites öszeadó blokkja



22. ábra A négy bites számláló kapcsolási rajza

	C_0	A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4	Σ_1	Σ_2	Σ_3	Σ_4	C_4	
Logic Levels	L	L	H	L	H	H	L	L	H	H	H	L	L	H	
Active HIGH	0	0	1	0	1	1	0	0	1	1	1	0	0	1	(10+9 = 19)
Active LOW	1	1	0	1	0	0	1	1	0	0	0	1	1	0	(carry+5+6 = 12)

23. ábra A négy bites számláló igazságtáblázata

7.3.2. Kivonó

A kivonás definíciója

A kivonás művelete matematikailag **a kivonandó szám mínusz egyszeresének hozzáadásával végezhető el.** Erre a 2-es komplemens kódot használják. Az áramköri megoldás lényege az, hogy a kivonandó minden bitjét negáljuk és ehhez még 1-et hozzáadunk. Majd az így keletkezett operandust és a másik operandust vezetjük be egy összeadó áramkörbe. Az így kapott eredmény 2-es komplemens kódban áll rendelkezésre. A kivonandó komplemensét előállíthatjuk párhuzamosan vagy sorosan.

A kivonó áramkörök szintén lehetnek soros vagy párhuzamos működésűek.

Példa:

A számítógépek a kivonást a kettes komplemens segítségével végzik. A kisebbítendőhöz hozzá kell adni a kivonandó kettes komplementerét. Ez két lépést jelent. Át kell váltani a kivonandót kettes számrendszerbe, majd a bitjeit át kell billenteni, majd 1-et hozzá kell adni. Az így kapott számot és a kisebbítendő bináris számát össze kell adni. például vonjuk ki a 8-ból az 5-öt.

8-5=3

8: 1000 5: 0101

5 I: 1010 (negálás)

5 II: 1011 (negálthoz adunk 1-et)

Ezeket most össze kell adni:

1000

+1011

_____

10011

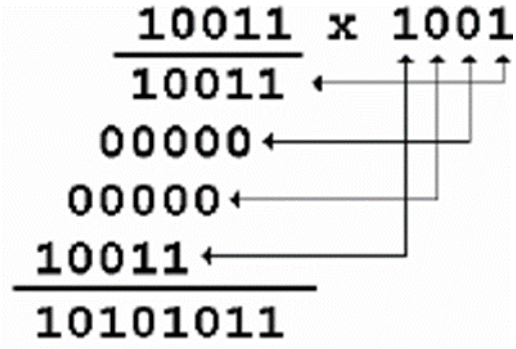
A kapott eredmény az 10011. Itt viszont keletkezik egy felesleges bit, túlcsordulás, amit le kell választani, arra nincs szükség, ezt levágjuk.

1 | 0011

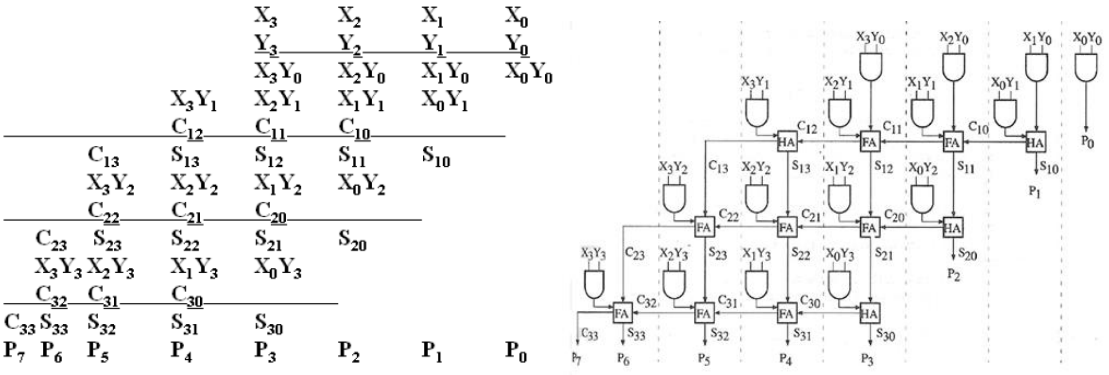
Az így kapott számot, ha vissza váltjuk 10-es számrendszerbe, akkor megkapjuk az eredményt.
0011 = 1+2 = 3

7.3.3. Szorzás

Két bináris szám összeszorzásánál is ugyanúgy járunk el, mint a decimális esetben: A szorzandót a szorzóval úgy szorozzuk meg, hogy a szorzó mindenegyes helyiértékével külön elvégezzük a szorzást, és a részeredményeket összeadjuk. A helyiértékek helyességére, ugyanúgy figyelünk, mint decimális esetben. fontos megjegyezni, hogy két N bites szám szorzata egy 2N bites számot eredményez!

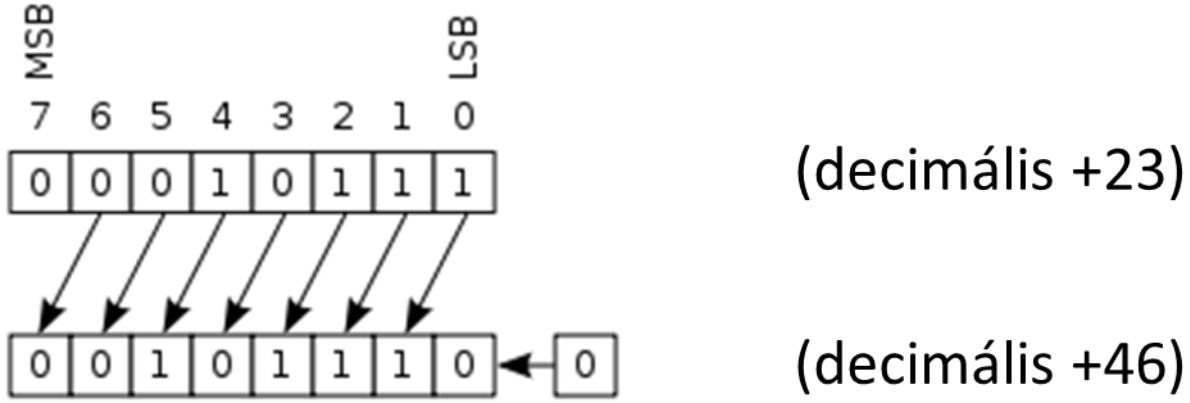


24. ábra A bináris szorzás

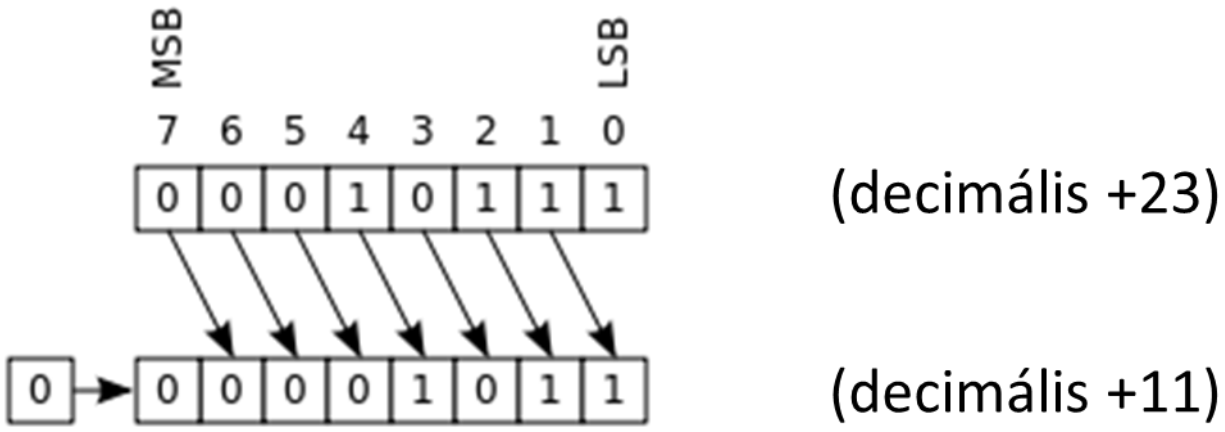


25. ábra A bináris szorzás megvalósítása logikai kapukkal

A szorzás megvalósítása kapukkal mindenképp bonyolultabb művelet, ezért egy egyszerűbb megoldásra kell törekedni. Ezt a lehetőséget a regiszterek kínálják a számunkra. A regiszterek ugyanis jobb, bal és mindkét irányba léptethetők. A léptetés során egy bit (ahogy „toljuk") kiesik, hogy a megfelelő bitszámunk meglegyen, ezért egy nulla logikai értékkel helyettesítjük, abból az irányból, melyből eltoltuk. Ugyanakkor fontos, hogy a balra tolás egy 2 – vel való szorzásmnak a jobbra tolás pedig egy kerttővel való osztásnak felel meg.



26. ábra Balra shiftelés - kettővel való szorzás

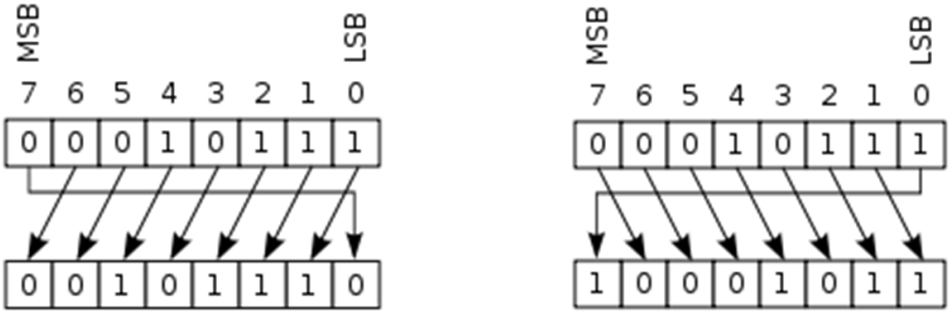


27. ábra Jobbra shiftelés - kettővel való osztás

Mivel ez egy jóval egyszerűbb megvalósítás, ezért a bonyolultabb számításokat is igyekszünk erre visszavezetetni.

7.3.4. Számlálók

A számlálok egy speciális esete az ún körkörös számláló. Ennek egy speciális esete a z Ún Jhonson s- számláló. Ebben az esetben a kieső bitet a shiftelés során a regiszter másik végére vezetjük vissza. Ebben az esetben az egymást követő bináris számok csak 1 bitben térnek el, ezlért ún Jhonson – számlálót kapunk.



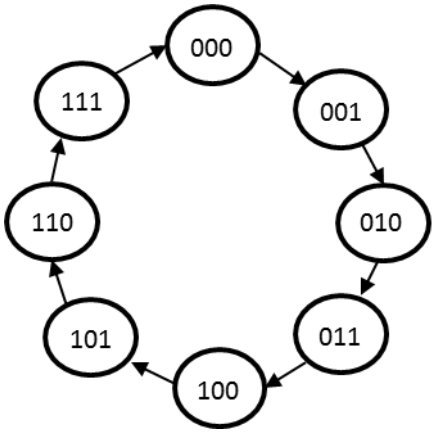
28. ábra Johnson számláló

7.3.4.1. Bináris felfele számlálók

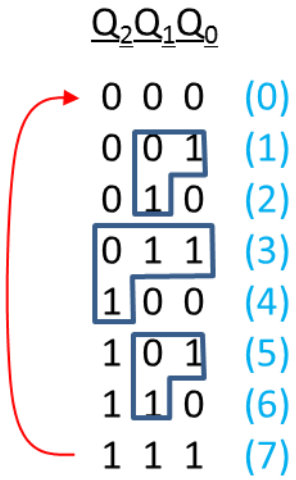
A bináris felfele számlálók működése nagyon egyszerűen leírható.

Pl. 0-tól 7-ig (000b – 111b) egyesével számlál minden órajel ciklusban (3-bites esetben természetesen). A korábban tanultak alapján a hálózat 2^3 = 8 állapotot vehet fel, és nincs tiltott állapota. A Q_0 bit minden órajelre billen. A T-tároló pedig T = 1 állandó bemenettel rendelkezik. A Q_1 bit akkor billen a következő órajelre, ha Q_0 = 1.A T-tároló T = Q_0 állandó bemenettel rendelkezik ekkor.

A Q_2 bit pedig akkor billen a következő órajelre, ha $Q_1Q_0 = 11$, és így a T-tároló T = (Q_0 ÉS Q_1) állandó bemenettel fog rendelkezni.

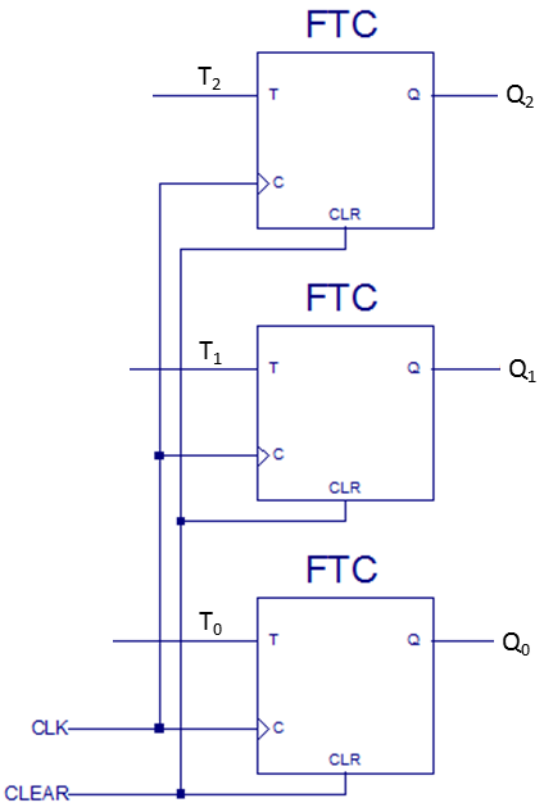


29. ábra A bináris felfele számláló működése



30. ábra A felfele számláló bináris számláló működése

A fenti ábrán jól látható, hogy a rendszer hogyan mozgatja az 1-es számot előre, azaz a regiszterhez hasonló működésű.



31. ábra a sa gszámláló megvalósítása

7.3.4.2. Bináris lefelé számláló

Hasonló az előzőhöz, csak lefelé működik.

Pl. 7-től 0-ig (111b – 000b) egyesével számlál minden órajel ciklusban (3-bites), a hálózat $2^3 = 8$ állapotot vehet fel, és nincs tiltott állapot.

VHDL:

```
use IEE.STD_LOGIC_ARITH.ALL;
…
signal q : STD_LOGIC_VECTOR(2 downto 0);
signal clk: STD_LOGIC;
signal clear: STD_LOGIC;
…
process
Begin
  if (clear = ’1’) then
            q <= ”000”;
```

```
        elsif (clk`event and clk = ’1’) then
  q <= q-1;
end if;
      end process;
```

7.3.4.3. Bináris fel-le számláló

A számlálási irányt megadó X bemenet függvényében fel, vagy lefele számol.

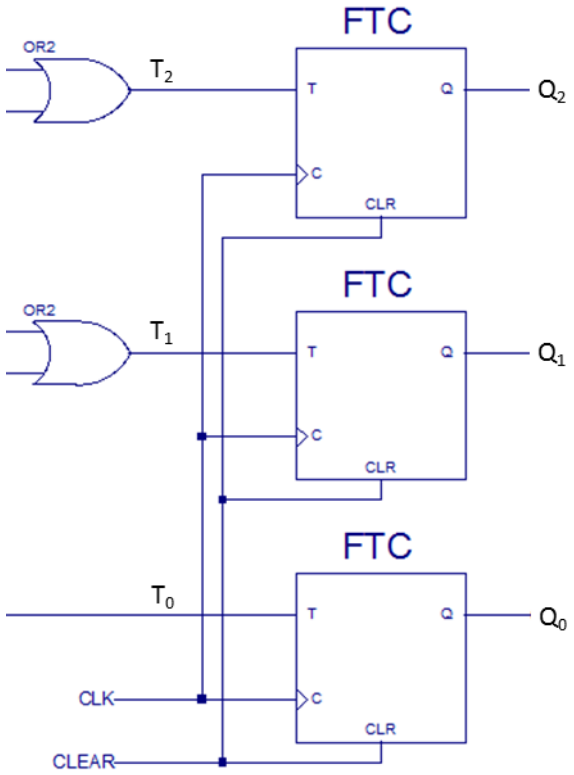
Pl. 7-től 0-ig (111b – 000b) egyesével számlál minden órajel ciklusban (3-bites)

Pl. 0-tól 7-ig (000b – 111b) egyesével számlál minden órajel ciklusban (3-bites)

$$T_2 = Q_1 \cdot Q_0 \cdot \bar{X} + \bar{Q}_1 \cdot \bar{Q}_0 \cdot X$$

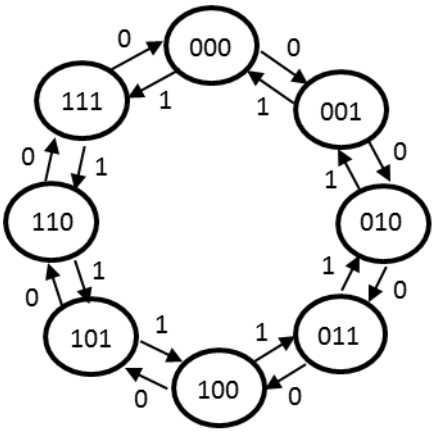
$$T_1 = Q_0 \cdot \bar{X} + \bar{Q}_0 \cdot X$$

$$T_0 = 1$$



32. ábra A fel leszámláló megvalósítása

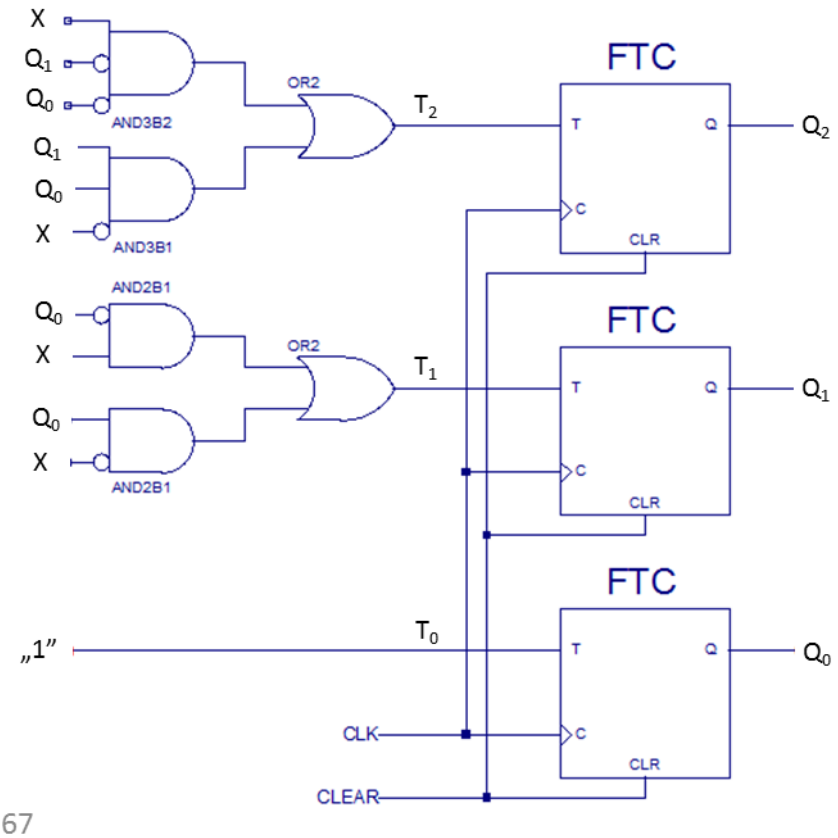
A fel le számláló működési elve az előbb elmondottakon alapul. Az a különbség, hogy egy érték logikai állapota megmondja, hogy a számláló melyik irányba léptet, számol.



33. ábra A fel le számláló működési elve

```
use IEE.STD_LOGIC_ARITH.ALL;
…
signal q : STD_LOGIC_VECTOR(2 downto 0);
signal x: STD_LOGIC;
signal clk: STD_LOGIC;
signal clear: STD_LOGIC;
…
process
Begin
   if (clear = ’1’) then
      q <= ”000”;
   elsif (clk`event and clk = ’1’) then
       if (x = ’0’) then q <= q+1;
      else q <= q-1;
      end if;
   end if;
end process;
```

34. ábra A fel- le számláló megvalósítása.



35. ábra A fel- le számláló kpcsolási rajza.

7.3.4.4. Decimális felfelé számláló

Decimális számláló megvalósításához már 4 bitre lesz szükségünk.

Pl. 0-tól 9-ig (0000b – 1001b) egyesével számlál minden órajel ciklusban (4-bites BCD kód)

A hálózat 2^4 = 16 állapotot vehet fel, ebből 6 állapot lesz tiltott állapot.

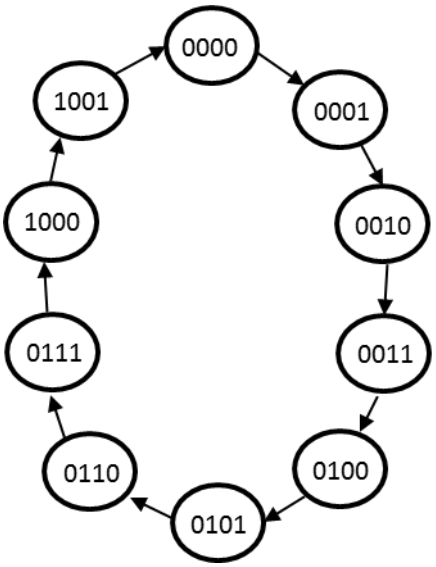
1001b-ig úgy működik mint egy bináris számláló, de 1001b után 0000b kell következzen.

Ezért a megvalósítás során a következő állapottáblát tudjuk felírni.

Q_3	Q_2	Q_1	Q_0	
0	0	0	0	(0)
0	0	0	1	(1)
0	0	1	0	(2)
0	0	1	1	(3)
0	1	0	0	(4)
0	1	0	1	(5)
0	1	1	0	(6)
0	1	1	1	(7)
1	0	0	0	(8)
1	0	0	1	(9)

36. ábra A decimális számláló állapottáblázata

A bemeneti kombinációs hálózat a bináris számlálótól különböző lesz



37. ábra A decimális számláló működése

A rendszert ez alapján az alábbi egyenletek segítségével lehet felírni:

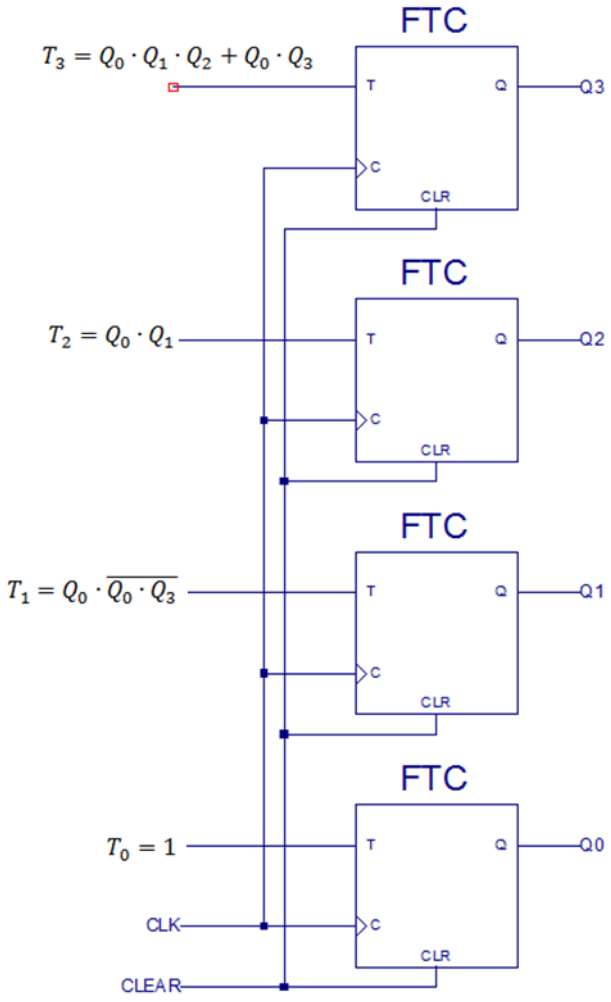
$$T_0 = 1$$

$$T_1 = Q_0 \cdot \overline{Q_0 \cdot Q_3}$$

$$T_2 = Q_0 \cdot Q_1$$

$$T_3 = Q_0 \cdot Q_1 \cdot Q_2 + Q_0 \cdot Q_3$$

Ezért a számláló az alábbi módon valósítható meg.



38. ábra A decimális számláló megvalósítása

7.3.4.5. Decimális lefelé számláló

Pl. 0-tól 9-ig (0000b – 1001b) egyesével számlál minden órajel ciklusban (4-bites BCD kód)

A hálózat 2^4 = 16 állapotot vehet fel, ebből 6 tiltott állapot ugyan úgy, mint az előző esetben.

Q_3 kimenet billen ha

$Q_0 = Q_1 = Q_2 = 1$ vagy

$Q_3Q_2Q_1Q_0 = 1001$

Q_2 kimenet billen ha

$Q_0 = Q_1 = 1$

Q_1 kimenet billen ha

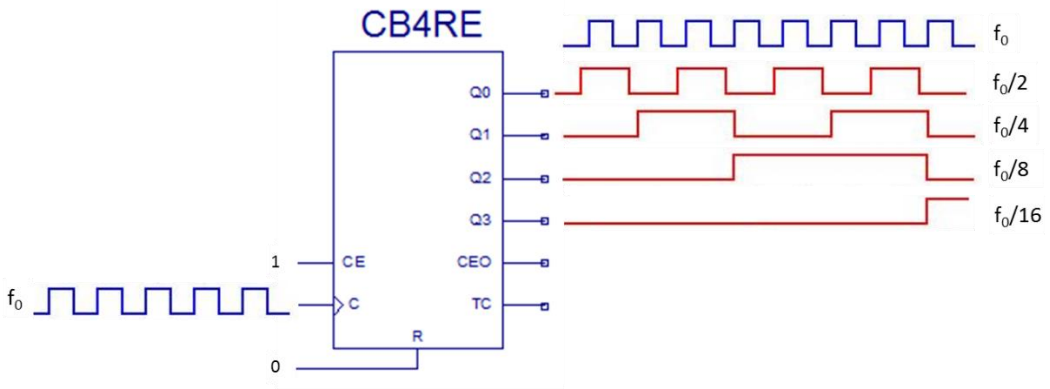
$Q_0 = 1$ és

$Q_3Q_2Q_1Q_0 \neq 1001$

Az ”1001” állapot jelzéséhez BCD kódban elég ha $Q_3 = Q_0 = 1$. Így sokkal gyorsabban ellenőrizhető is az állapot megléte: gyakorlatilag nem a teljes kódot, csak két bitet ellenőrzünk.

7.3.4.6. Frekvenciaosztás számlálókkal

Számláló segítségével egy négyszögjel frekvenciaosztása is lehetséges az alábbi módon. Egy n bites számláló frekvenciaosztása: 2^n.



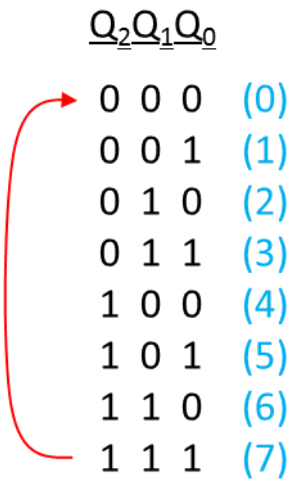
39. ábra A frekvenciaosztás

7.3.4.7. Aszinkron számlálók: Bináris felfelé számláló

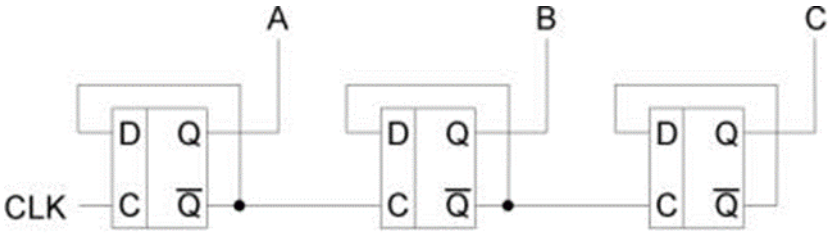
Eddig csak a szinkron megvalósítású számlálókról beszéltünk.

Pl. 0-tól 7-ig (000b – 111b) egyesével számlál minden órajel ciklusban (3-bites). A hálózat $2^3 = 8$ állapotot vehet fel, nincs tiltott állapot, a T tárolók kimenete pedig a következő tároló órajel bemenetére csatlakozik, így a tárolók kimeneti jelének frekvenciája fele a bemeneti órajel frekvenciájának. A rendszer előnye, hogy nem kellenek kiegészítő kapuáramkörök, hátrány viszont, hogy a késleltetések miatt a tárolók nem egyszerre billennek, valamint az órajel változásakor rövid időre határozatlan kimeneti jelet kapunk.

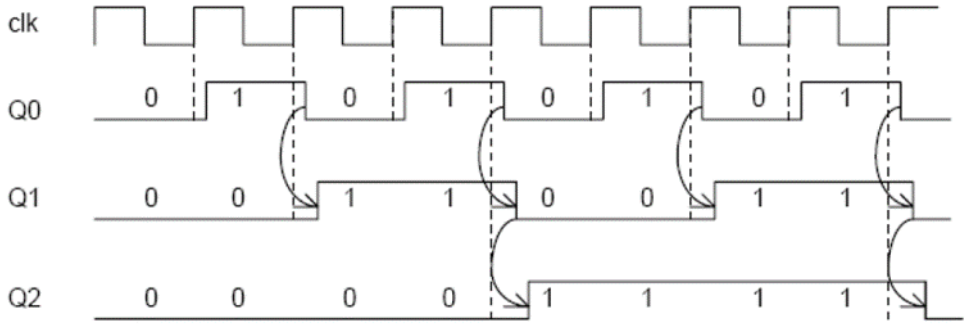
A rendszer állapottáblája ugyanaz, mint a szinkron változatnál.



40. ábra A tállapottáblája



41. ábraA számláló megvalósítása



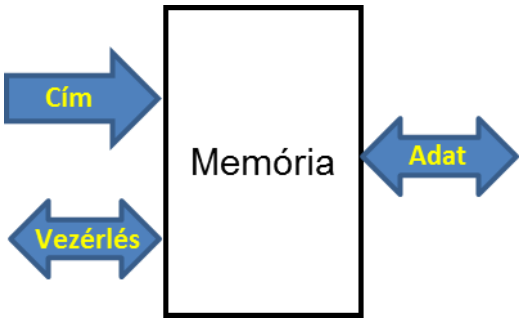
42. ábra A belső állapotok váltzása az aszinkron hálózat esetén.

7.4. Memória

7.4.1. A memória definíciója:

A memóriák tulajdonképpen nagyobb mennyiségű információ átmeneti vagy tartós tárolására szolgáló egységek. Valójában speciális regisztereknek tekinthetők.

Néhány bit, vagy bitcsoport (4, 8, 16, 32 …stb.) tárolására alkalmas. Több regiszterből nagyobb tárolókapacitású tárolók építhetők, ehhez azonban kiegészítő és vezérlő egységek szükségesek a tárolt információ célszerű kezeléséhez. Ezért fontos a más áramkörökkel való együttműködés, illeszthetőség foglaltság, készenlét stb. jelzése.



43. ábra A memória

7.4.2. Memóriák működése

A memória működése során a „Cím” bemenetre érkező információval jelöljük ki azt, amit szeretnénk az információ memórián belüli helyét (memória rekeszt)

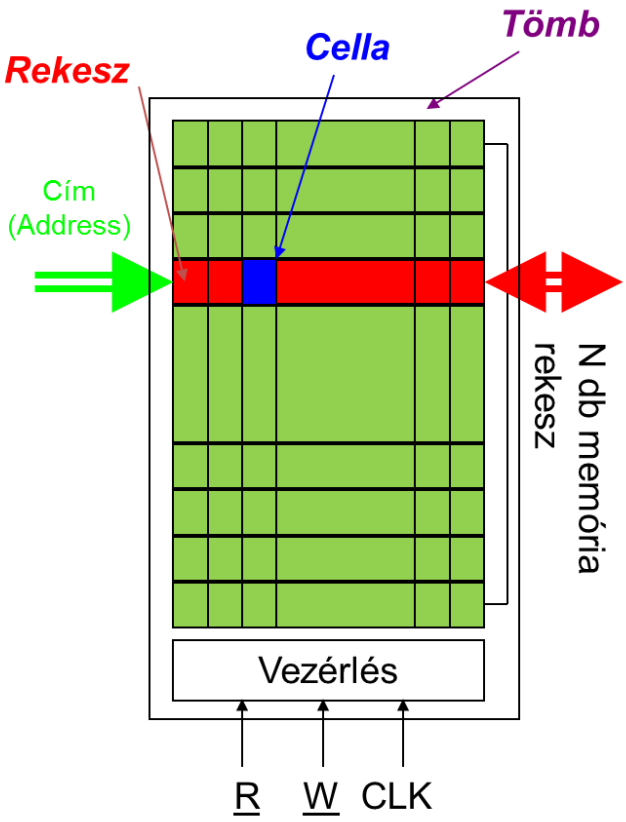
Az Adat csatlakozó szerepe kettős:

1) az „Adat” csatlakozóra érkező, tárolandó
2) az „Adat” csatlakozón távozó, kiolvasandó

adatok egyaránt itt fognak mozogni.

A „Vezérlés” csatlakozásokon keresztül egyrészt utasításokat adhatunk, (beírás, kiolvasás stb…) másrészt információt nyerhetünk a memória működésére vonatkozólag (foglaltság, készenlét stb…).

7.4.2.1. A memória általános felépítése



44. ábra A memória felépítése

A memória felépítése több tároló egységében képzelhető el, melyhez egy három állapotot kódoló vezérlő egység csatlakozik. A memória legnagyobb egysége a tömb, mely rekeszekre (sorokra) osztható. Egy rekesz pedig több cellából épül fel. A legkisebb egység tehát a cella. 1 db cella 1 bit tárolására képes. 1 rekesz (sor) K db cellából épül fel, ezért 1 sor mérete K bit. Ettől függően léteznek:

- Byte szervezésű (K = 8)
- Szó szervezésű (K = 16)
- Duplaszó szervezésű (K = 32)

A párhuzamos hozzáférésű memóriáknál az adatbusz hozzávezetéseinek számát K adja meg.

A memória tömb N = 2L db rekeszből épül fel, ezért a párhuzamos címzésű memóriáknál a címbusz hozzávezetéseinek száma L.

Memória tárolókapacitása ebből következően megadható:

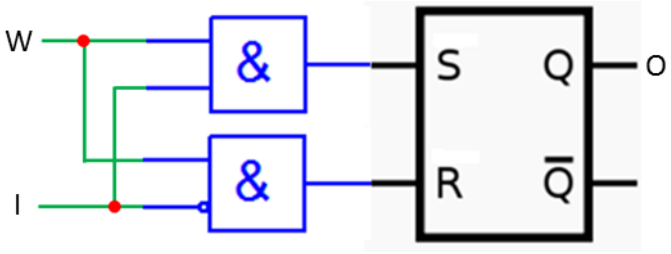
tárolt bitek száma = 1 rekesz mérete * rekeszek darabszáma = K * N bit

Vezérléstől függően 3 féle állapotot vehet fel a memória adott cellája. :

1) Írás: R = 1, W = 0
2) Olvasás : R = 0, W = 1
3) Üresjárat / tárolás: R = 1, W = 1

7.4.2.1.1. Egy cella felépítése

A cella, mely 1 bit tárolására képes tulajdonképpen RS tárolóból és a hozzá kapcsolódó vezérlő egységekből épül fel.

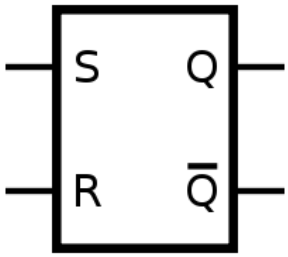


45. ábra Egy cella felépítése

R	S	Q^{n+1}
0	0	Q^n
0	1	1
1	0	0
1	1	X

46. ábra Az RS tároló igazságtáblázata

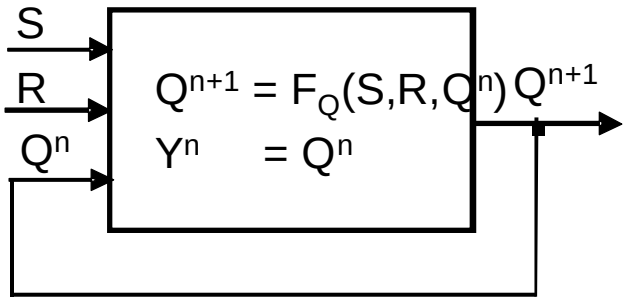
❖ **Az S-R flip-flop**



47. ábra Az R -S flip- flop

Az S-R flip-flopnak egy beállító (Set), és egy törlő (Reset) bemenete van. Az egyik legegyszerűbb flip-flopnak tekinthető, bár alapvetően tároló. A két bemenet egyidejű felemelését tiltani szokták, mivel ez instabil állapotot idézne elő (ld. versenyhelyzet).

❖ **Az RS tároló modellje**



48. ábra Az RS tároló felírása modellel

❖ **Az RS tárolót leíró függvény**

A modell alapján a rendszer függvényét is fel tudjuk írni.

$$Q^{n+1} = S + \bar{R}Q^n$$

$$\bar{Q}^{n+1} = R + \bar{S}\bar{Q}^n$$

❖ **Az RS tároló állapottáblája**

A rendszer felírását elvégezhetjük állapottábla segítségével. Így jól nyomon követhető a működés is.

❖ Az S(Set) bemenetre adott „1”-es a kimenetet „1”-be állítja

- Az R(Reset) bemenetre adott „1”-es a kimenetet „0”-ba állítja

R	S	Q^{n+1}
0	0	Q^n
0	1	1
1	0	0
1	1	X

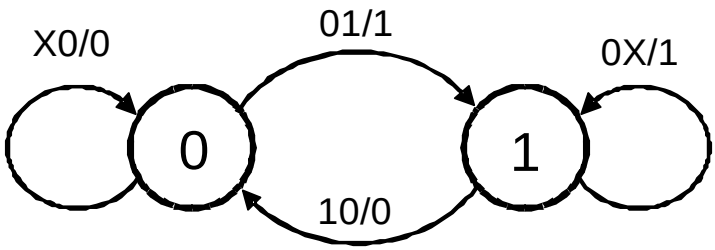
49. ábra A rendszer állapottáblája, hogy jól látszon a FLIP _ FLOP

Jól látszik a fenti állapottáblán a rendszer flip-flopozása is, mert így válik megfigyelhetővé, hogy vagy egyik, vagy másik bemenet lehet 1 . A tároló azon kívül, hogy memóriája van, kapcsolóként is működik.

R	S	Q^n	Q^{n+1}	
0	0	0	0	Változatlan
0	0	1	1	
0	1	0	1	Beírás
0	1	1	1	
1	0	0	0	Törlés
1	0	1	0	
1	1	0	X	Tiltott
1	1	1	X	

50. ábra Az RS tároló állapotáblája

Állapot gráffal történő felírás is elvégezhető természetesen, a már ismert szabályok felhasználásával.



51. ábra Az állapotgráf

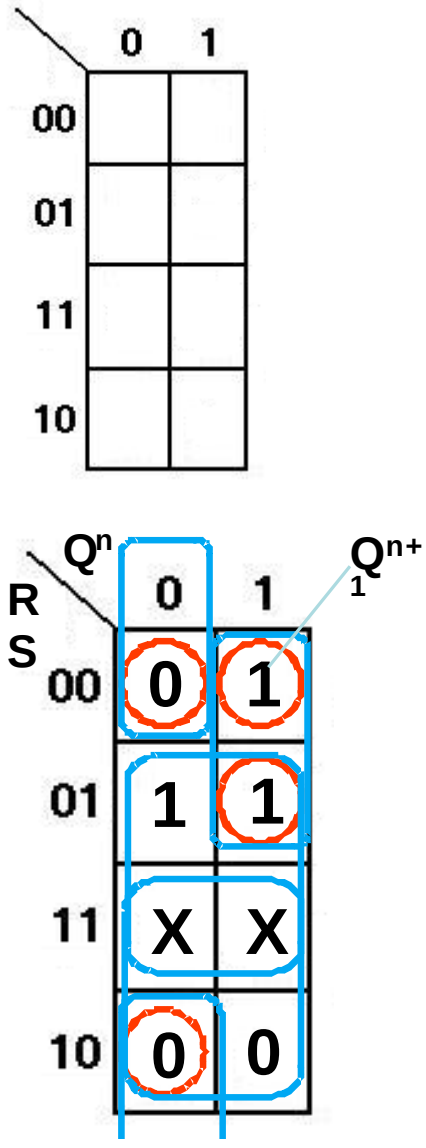
- **Az RS tároló vezérlési táblája**

Ebben a rendszerben nincs versenyfutás vagy oszcilláció, tehát az aszinkron működés is stabilnak tekinthető. Vannak viszont érdektelen (Don't care állapotok).

Az állapottáblát Karnaugh-táblának tekintve, Qn+1-re elvégezve az összevonásokat az egyszerűsített logikai függvény:

$$Q^{n+1} = S + \bar{R}Q^n$$

$$\bar{Q}^{n+1} = R + \bar{S}\bar{Q}^n$$



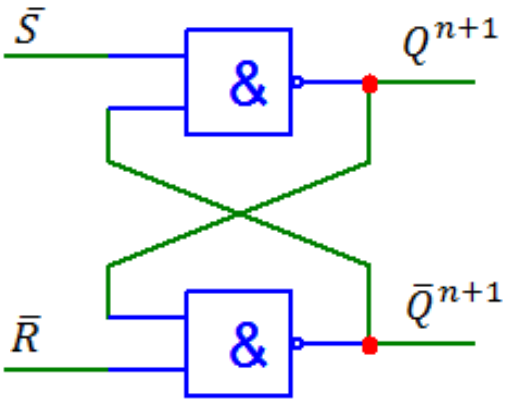
52. ábra Az RS tároló vezérlő táblája

- **Szinkron RS tároló (Filp-flop) megvalósítása**

A Qn+1-et és Qn+1 -et megvalósító kombinációs hálózat logikai függvénye. A hálózat NAND kapus megvalósítása a következőképpen hozható létre. Itt fontos szempont, hogy így csak NAND kapukat kell használni.

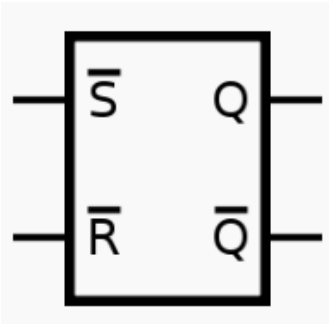
$$Q^{n+1} = S + \bar{R}Q^n = \overline{\bar{S} \cdot \overline{\bar{R}Q^n}}$$

$$\bar{Q}^{n+1} = R + \bar{S}\bar{Q}^n = \overline{\bar{R} \cdot \overline{\bar{S}\bar{Q}^n}}$$

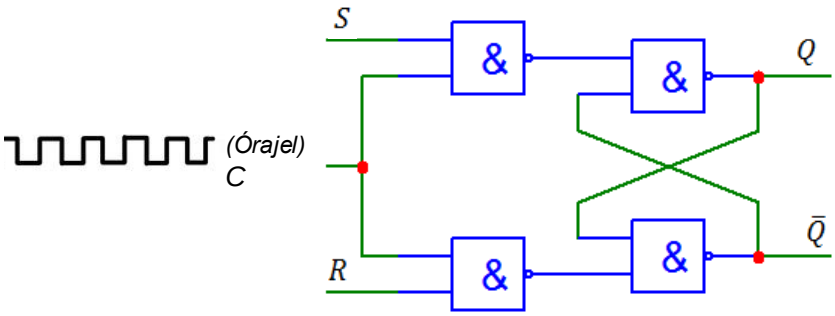


53. ábra Az RS tároló megvalósítása

A megvalósítás során a fentebb leírt két egyenletet kell figyelembe vennünk. Ugyanakkor az is fontos, hogy az R és S bemenetek hatása a szinkronjel (órajel) megérkezésekor érvényesüljön, ezért még két AND kapu felhasználásával a rendszerre rákötjük az órajelet is.



54. ábra A RS tároló NAND kapus megvalósítása

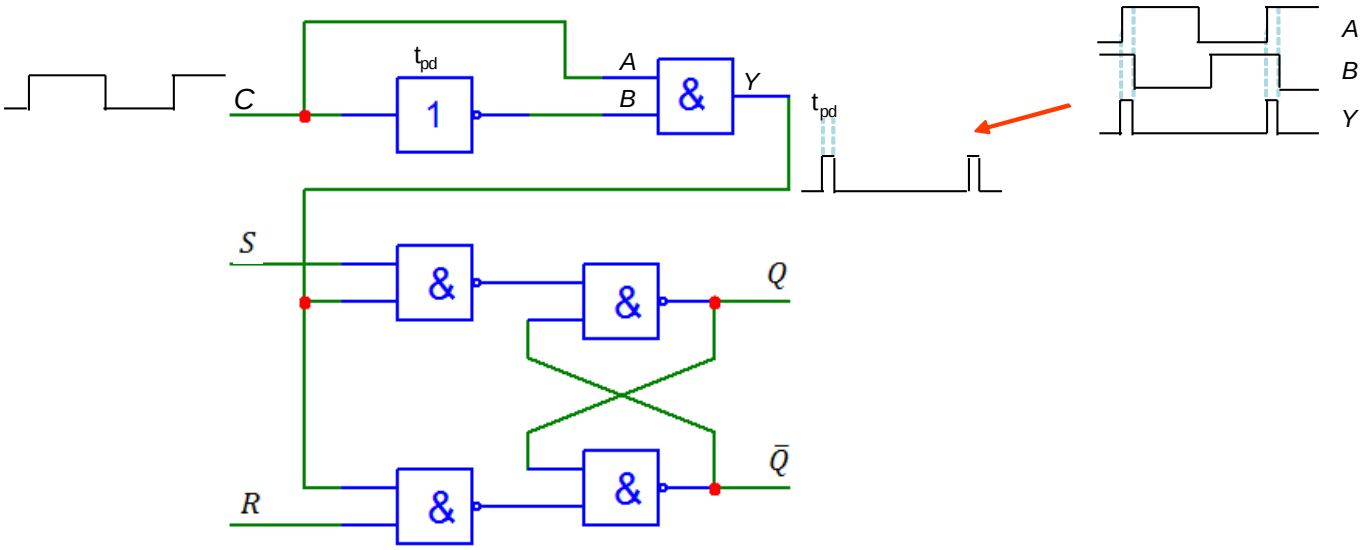


55. ábra A tárolóra rákötjük az órajelet is.

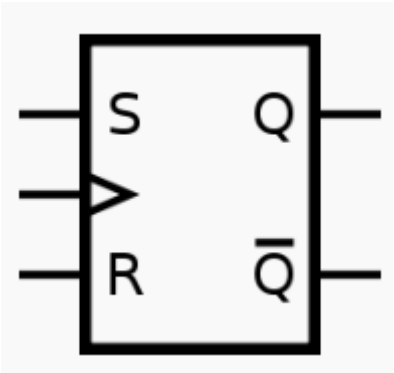
A rendszer statikus vezérlés/ szint vezérlés esetén csak akkor fog átbillenni, ha az órajel 1 értékű. Ez a megoldás nem használható szinkron hálózat építésére mert ún. „átlátszó”. Az órajel „1” értékénél az esetleges többszöri változás a bemeneten a kimenetet is többször átbillentheti, és ez tovább is terjed a flip-flopon keresztül. Ezért ún. dinamikus élvezérlést alkalmazunk, melynek során nem engedjük folyamatosan az órajel „1” értéke alatt hatni a bemeneteket csak egy rövid időre, amíg a tároló át tud billeni, ez után elvesszük a beíró (óra) jelet. Ezt úgy valósítjuk meg, hogy lerövidítjük az órajel „1” értékét, azaz szándékosan hazárdos órajel formáló hálózatot „csinálunk”.

Viszont ez idő alatt az ilyen elemekből felépített hálózat teljes egésze aszinkron módon viselkedne.

Ez szinkron hálózatban nem megengedhető, mert ott egy szinkron jel csak egy változásra adhat lehetőséget.

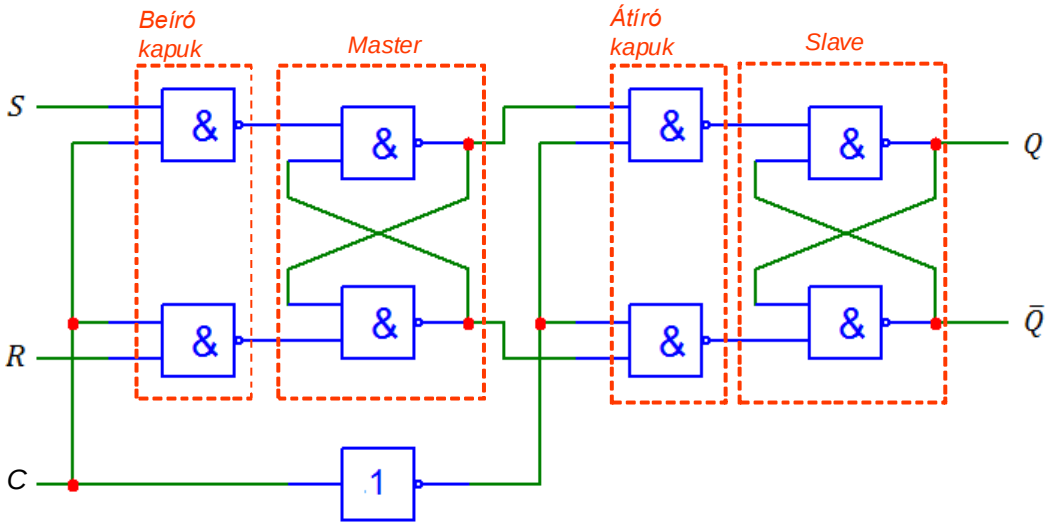


56. ábra Az órajel rövidítése



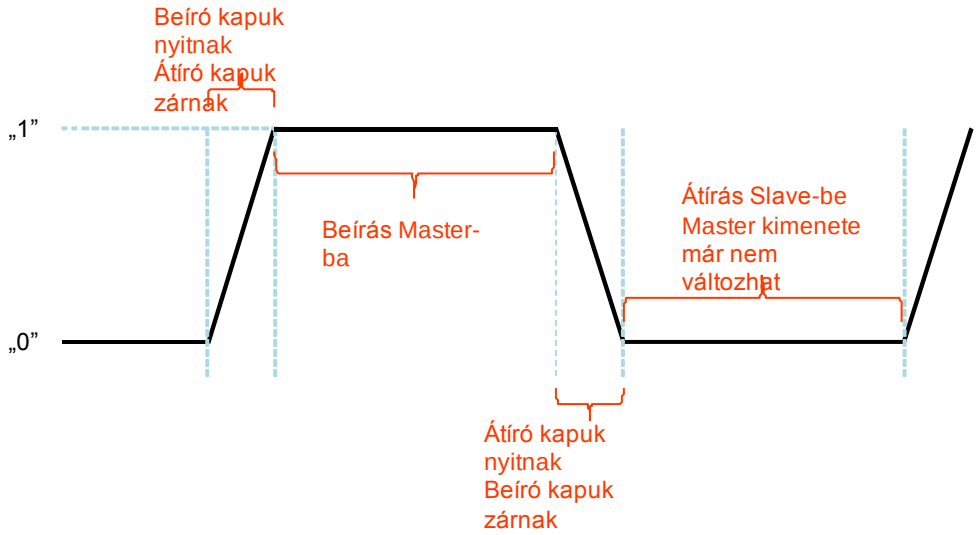
57. ábra RS tároló - órajel rövidítés funkcióval

A megvalósítás során ún. Kétfokozatú tároló (Master-Slave flip-flop) létrehozása az ideális, mert a Master-be írás alatt lehet tranzien,s de az átírás előtt már lecseng, és ezért tiltás alatt Master kimenete állandó.



58. ábra Az RS tároló Master - Slave megvalósítása

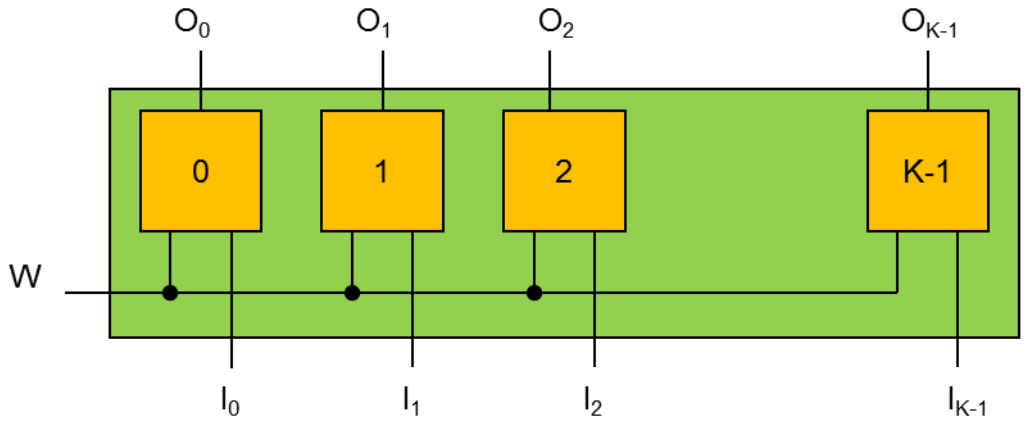
A Master – Slave kétfokozatú tároló megvalósítás lényege, hogy kétfokozatú tároló a Master-be írás alatt lehet tranziens, de az átírás előtt már lecseng, és az átírás alatt Master kimenete állandó



59. ábra Master - Slave kétfokozatú tároló működési elve

7.4.2.1.2. Egy rekesz felépítése

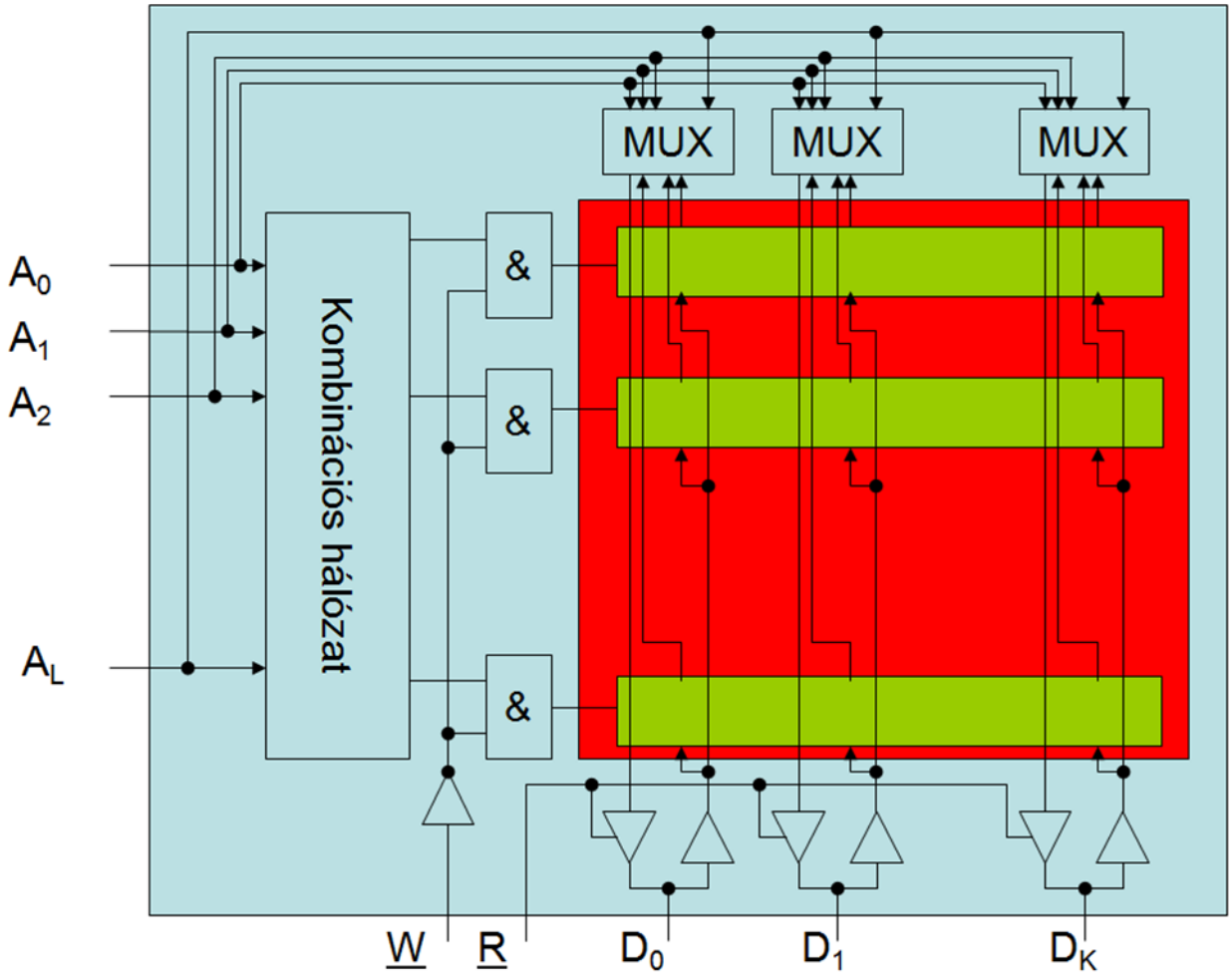
A rekesz esetében a regiszterhez hasonlóan, tároló cellák vannak párhuzamosan kapcsolva.



60. ábra A rekesz felépítése

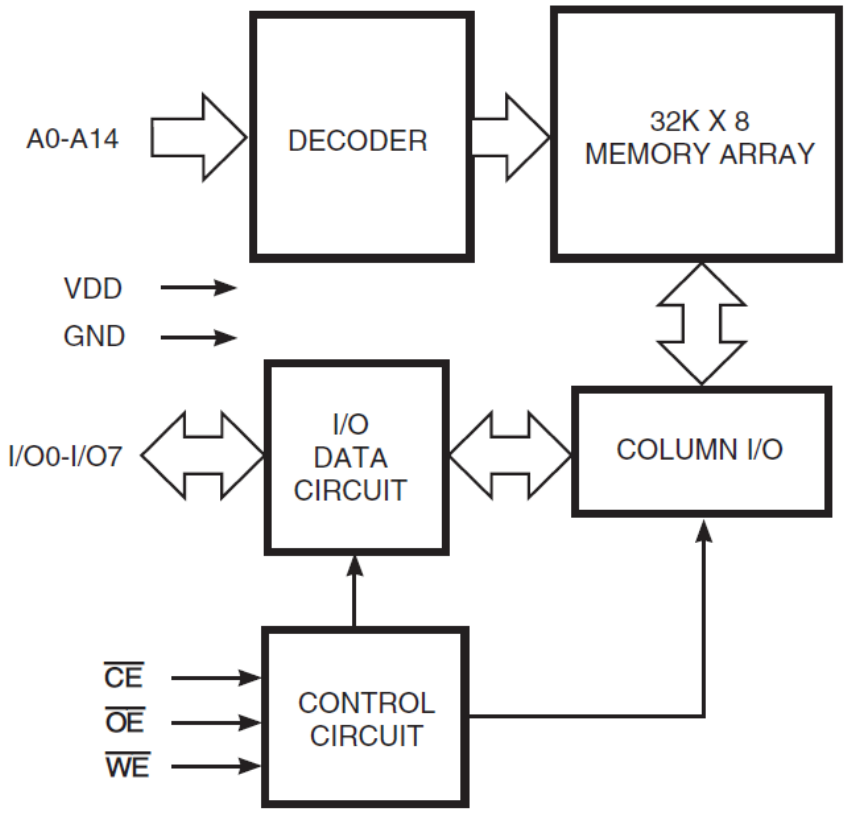
7.4.2.1.3. A tömb felépítése

A tömb szervezése során a rekeszek egységeit multiplexerekkel tudjuk összekötni, a bemenetek vezérlését és kapukkal és egy egyszerű kombinációs hálózattal tudjuk megoldani.



61. ábra A tömb felépítése

A valóságban ez a kapcsolat a következő ódon ábrázolható:



ahol:

a decoder:

Memmory array:

VDD

GND

I/O Data circuit

Colum /I/o

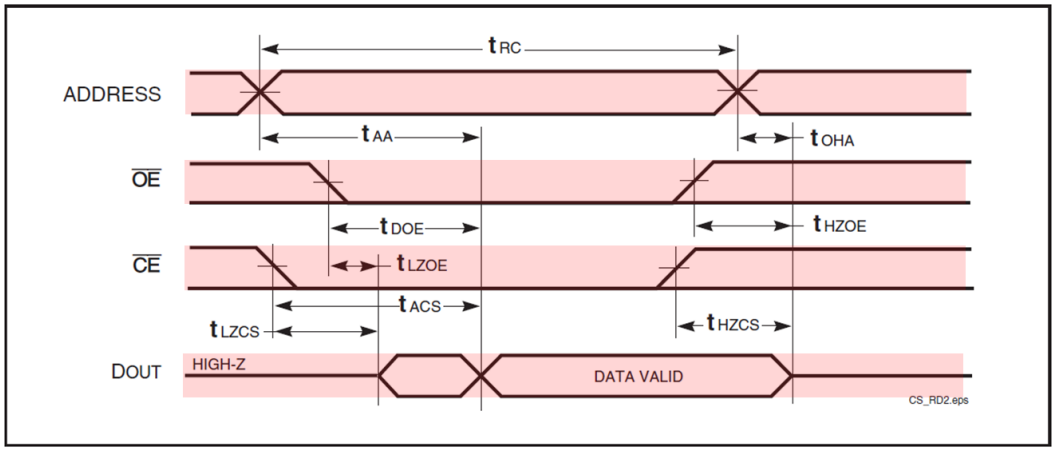
Conrol circuit:

CE: Chip Enable

OE : Output Enable

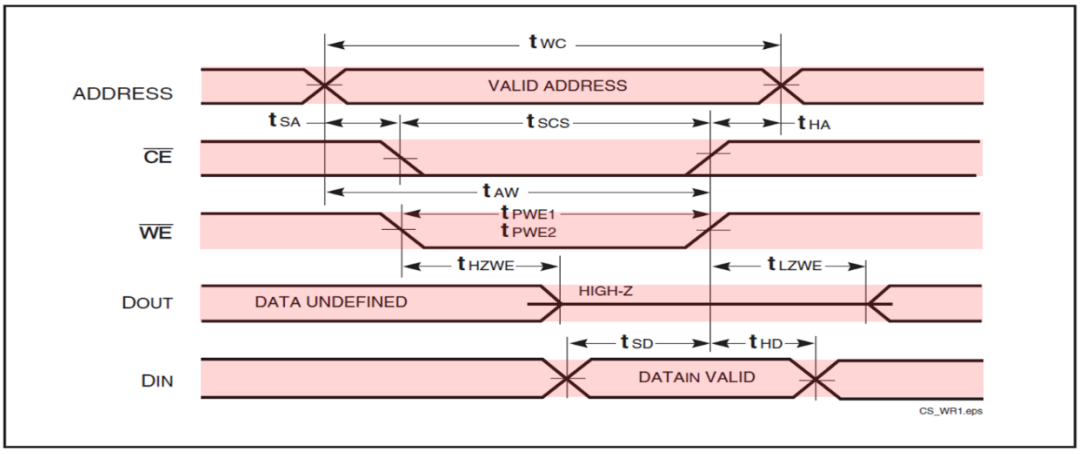
WE: Write enable

Olvasás, párhuzamos független adat és címvezetékekkel



62. ábra Olvasás, párhuzamos független adat és címvezetékekkel

Írás, párhuzamos független adat és címvezetékekkel



63. ábra Írás, párhuzamos független adat és címvezetékekkel

7.4.3. A memóriák csoportosítása

A memóriákat több szempont szerint csoportosíthatjuk.

1) A megcímzett rekesz hozzáférési módja szerint
2) Az információ beírhatósága szerint
3) A tárolás időbeli módja szerint

7.4.3.1. A megcímzett rekesz hozzáférési módja szerint

A megcímzett rekesz hozzáférési módja szerint lehet

1) Tetszőleges (véletlen) hozzáférésű memória (RAM: Random Access Memory)

   Ebben a memóriában bármely adat a címtől függetlenül azonos idő alatt elérhető.
2) Soros hozzáférésű memória (SAM: Serial Access Memory)

Ebben az esetben az adat címétől függő elérési idővel kell dolgoznunk. Ilyen Pl. a mágnesszalagos tárolás.

3) Asszociatív memória (CAN : Content Addressable Memory)

   Ez a fajta memória megadja, hogy az adott információ a memória mely címén található.

7.4.3.2. Az információ beírhatósága szerint

Az információ beírhatósága szerint két memóriatípust különítünk el.

1) Csak olvasható memória (ROM: Read Only Memory)
2) Módosítható memória (RWM: Read Write Memory)

RAM

7.4.3.3. A tárolás időbeli módja szerint

A tárolás módja szerint lehetnek a memóriák:

1) Statikus (pl.: SRAM) Tápfeszültség esetén az információt korlátlan ideig megőrzi
2) Dinamikus (pl.: DRAM) A memória tartalma időnként frissítésre szorul (ez hátrány) (De) nagy tároló kapacitás érhető el.

7.4.3.4. Hozzáférés szerint:

Az alapján is különbséget tudunk tenni a memória szervezésben, hogy a hozzáférésünk soros, vagy párhuzamos módon valósul meg. A példában vegyünk egy egyszerű memóriát, ahol *512M x 32 (K =29, L=32) tömb esetén az*

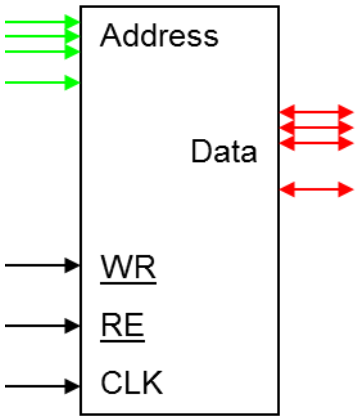
***M**: chip lábainak száma,*

***Q**: 1 művelethez szükséges órajel periódusok száma.*

Így minden típusra kiszámíthatjuk a megfelelő értékeket, és a memóriák kapacitása, előnyei és hátrányai láthatóvá vállnak.

1) Párhuzamos

Független cím- és adatbusz esetén



64. ábra Független cím- és adatbusz

M = 66
Q = 1 + 1

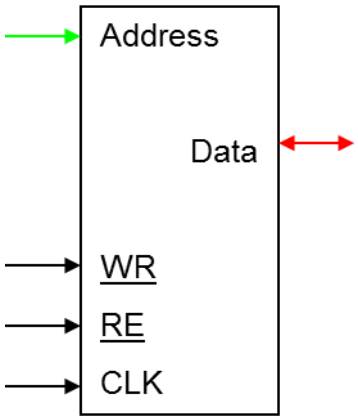
Univerzális (egyesített) Cím- és adatbusz

65. ábra Univerzális (egyesített) Cím- és adatbusz

M = 38
Q = 1 + 1

2) Soros

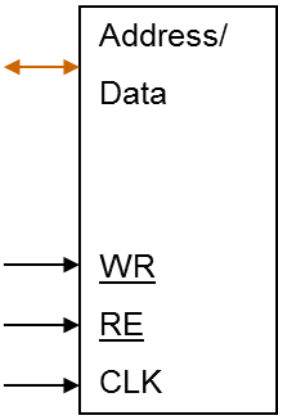
Független cím-és adatbusz



M = 7
Q = 29+32

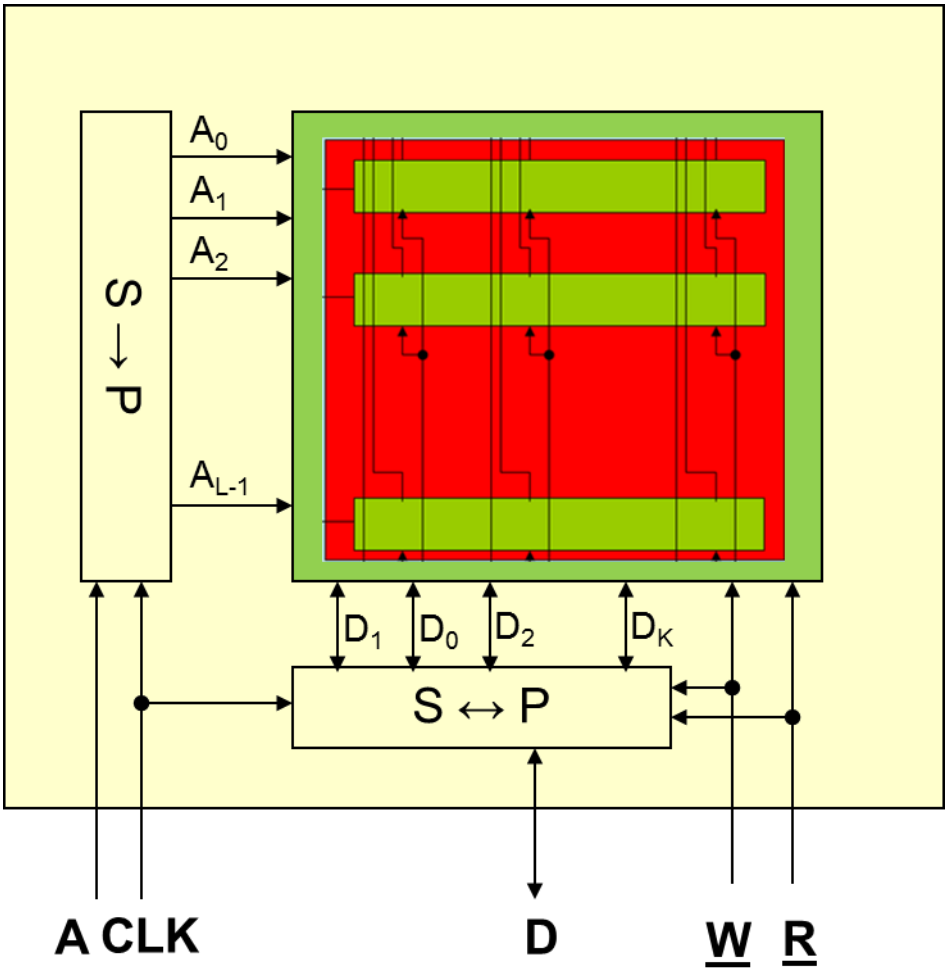
66. ábra Független cím-és adatbusz

Univerzális (egyesített) Cím- és adatbusz



M = 6
Q = 29 + 32

Soros hozzáférésű memória megvalósítása lehetséges párhuzamos memóriából, soros-párhuzamos, párhuzamos-soros átalakítókkal.



67. ábra Soros memória megvalósítása

Pl.: A következő módon lehet egy memóriát megvalósítani.

216 bit tárolása (64 Kibit), 216 cella

8 bit (byte) szervezés: 8 KiByte, 213 rekesz

Párhuzamos szervezéssel

8 adatvezeték

13 cím vezeték

Soros szervezéssel csökkenteni lehet a kivezetések számát

Egy 32 bites 2GiByte-os memóriának

32 adatvezetés

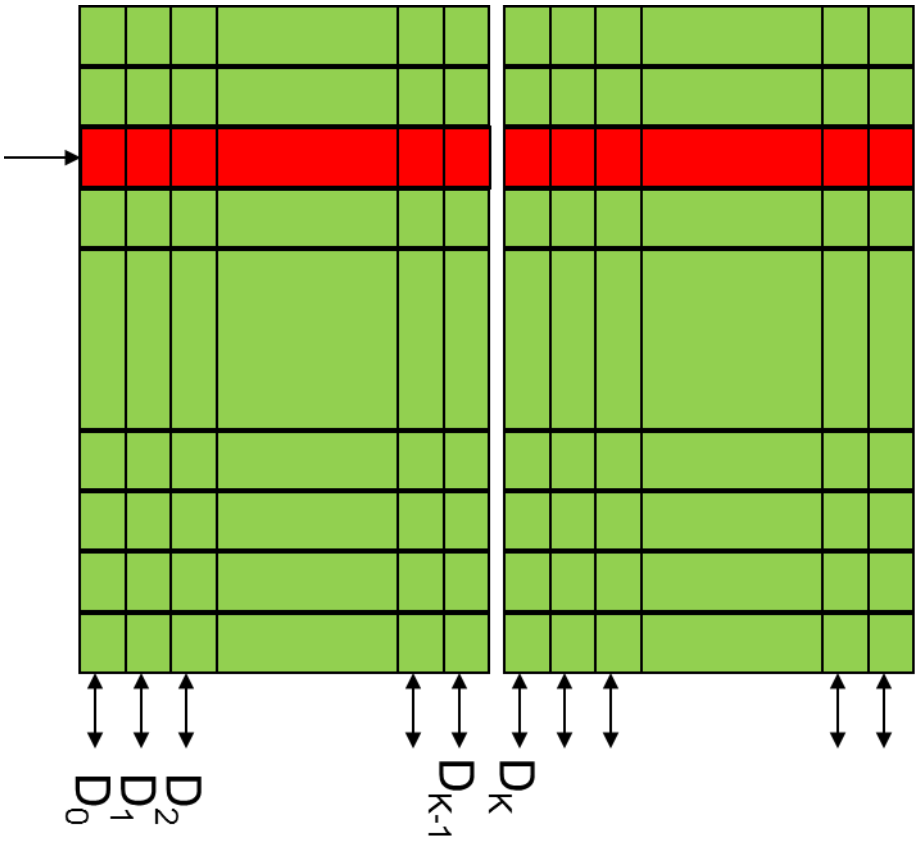
29 címvezeték

Párhuzamos memória + soros-párhuzamos átalakítás

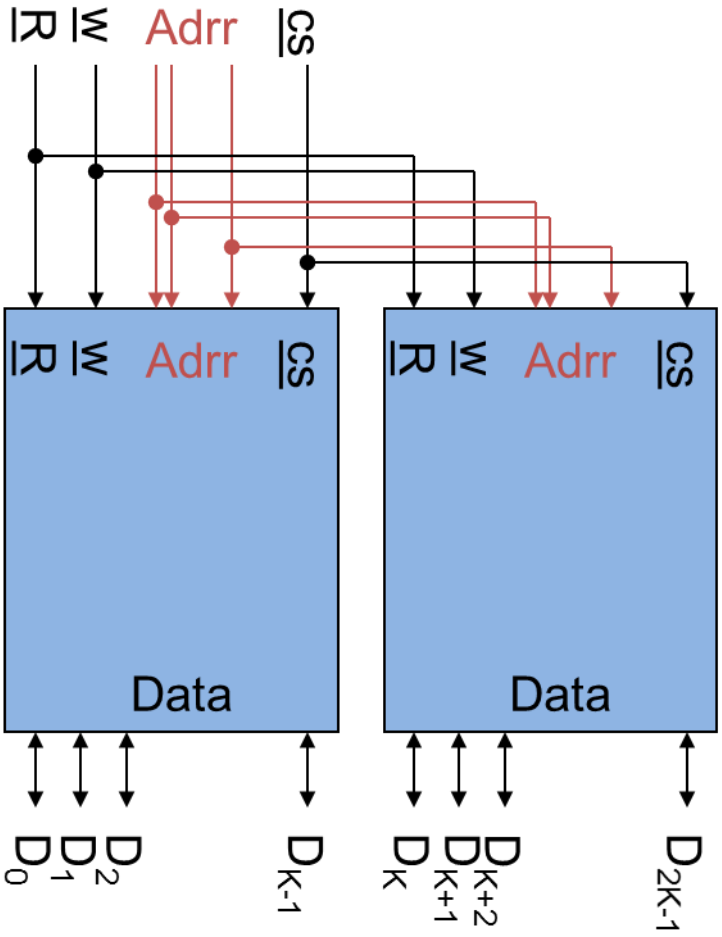
7.4.3.5. A memóriák bővítése

A memória bővítése alapvetően kétféle módon valósítható meg. Egyrészt tudunk szóhosszt bővíteni, másrészt képesek vagyunk kapacitásbővítést is létrehozni.

A szóhossz bővítés során két memóriatömböt kapcsolunk „egymás mellé”, úgy, hogy az azonos regiszterek egymás mellé kerüljenek. Ez azt is jelenti, hogy ezt csak két olyan tömbbel tudjuk megtenni, ahol a regiszterek/sorok száma azonos.

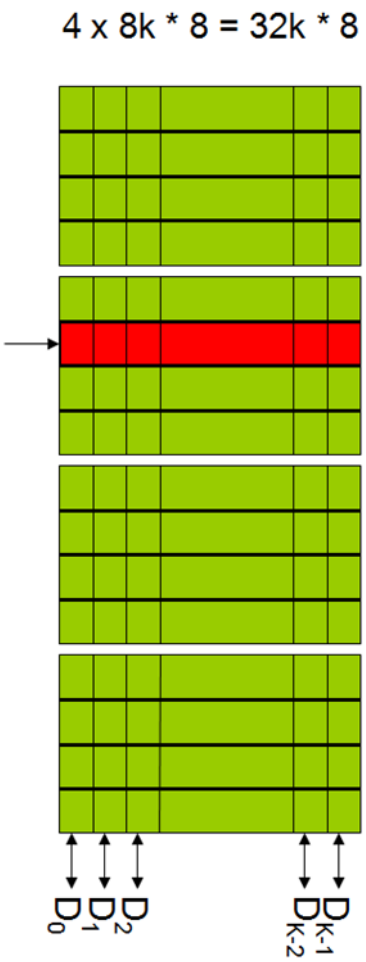


68. ábra Szóhosszbővítés 1.

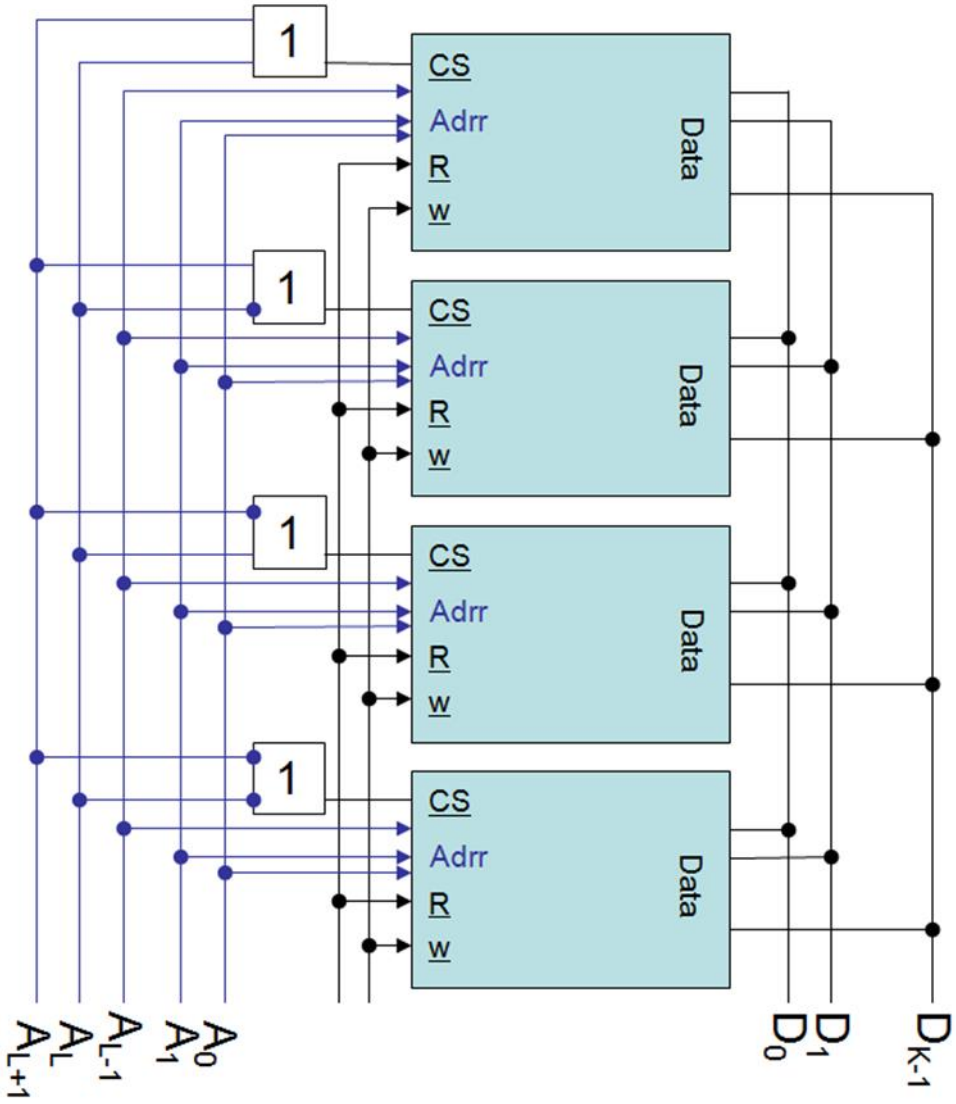


69. ábra Szóhosszbővítés 2.

A második esetben, a kapacitásbővítés során pedig úgy járunk el, hogy a két memóriatömböt egymás alá helyezzük el. Így a regiszterek/sorok számának növelésével növeljük meg a memória kapacitását.

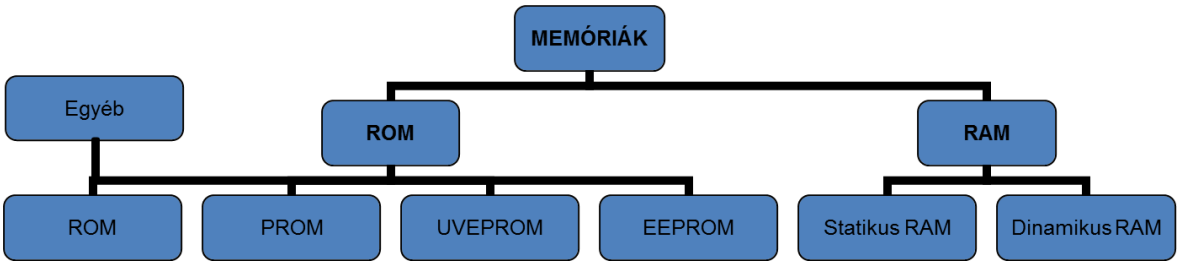


70. ábra Kapacitásbővítés 1



71. ábra kapacitásbővítés 2

7.4.3.6. A memóriák csoportosítása működés szerint



❖ **ROM**

Ezek a memóriák olvashatóak.

1) ROM

Írás a gyárban történik, törlés nem lehetséges.

2) PROM

1x felhasználó által is írható (beégethető), de törlés nem lehetséges

3) UVEPROM

Felhasználó által is írható, törlés 10”-es UV-s levilágítással történhet, ha szükséges.

4) EEPROM

Felhasználó által is írható és elektromosan törölhető

❖ **RAM**

1) Statikus RAM

Tápfeszültség nélkül elveszti a tartalmát, viszont gyors és nem kell frissíteni.

2) Dinamikus RAM

Tápfeszültség alatt is néha frissíteni kell, és lassabb.

7.5. Hogyan készítünk decimális Ellenőrző kérdések

1) Definiálja a regiszter fogalmát!
2) Sorolja fel a regiszter típusait!
3) Jellemezze a jobbra shiftelő regisztert!
4) Jellemezze a balra shiftelő regisztert!
5) Jellemezze a mindkét irányba shiftelő regisztert!
6) Mire használhatóak a regiszterek?
7) Hogyan készíthetünk véletlen szám generátort?
8) Hogyan készíthetünk gyűrűs számlálót?
9) Miképpen tárolhatunk átmenetileg adatot?
10) Hogyan adunk össze bináris számokat? Válaszát példával illusztrálja!
11) Hogyan vonunk ki bináris számokat? Válaszát példával illusztrálja!
12) Hogyan szorzunk össze bináris számokat? válaszát példával illusztrálja!
13) Hogyan készítünk bináris felfele számlálót?
14) Hogyan készítünk bináris lefele számlálót?
15) Hogyan készítünk bináris fel-lefele számlálót?
16) Hogyan készítünk decimális lefele számlálót?
17) Hogyan készítünk decimális felfele számlálót?
18) Mit tudunk az aszinkron számlálókról?
19) Hogyan oszthatunk frekvenciát számlálók segítségével.?
20) Hogyan épül fel a memória?
21) Hogyan valósítható meg a memória a gyakorlatban?
22) Milyen szempont szerint és hogyan csoportosíthatjuk a memóriákat? Jellemezze az egyes csoportokat!

7.6. Feladatok

1) T tárolók és ÉS –VAGY kombinációs hálózat segítségével tervezze meg és rajzolja fel egy 3 bites szinkron számláló MEALY - MODELL szerinti logikai kapcsolási rajzát, amely a következő sorrendben számlál: 0, 1, 2, 3, 5 . Ezután ismétlődik. A belső állapotokat Q0, Q1, Q2, a tároló bemeneteket pedig D0, D1, D2 szimbólumokkal jelölje.
2) D tárolók és ÉS –VAGY kombinációs hálózat segítségével tervezze meg és rajzolja fel egy 3 bites szinkron számláló MEALY - MODELL szerinti logikai kapcsolási rajzát, amely a következő sorrendben számlál: 0, 1, 2, 3 . Ezután ismétlődik. A belső állapotokat Q0, Q1, Q2, a tároló bemeneteket pedig D0, D1, D2 szimbólumokkal jelölje.
3) D tárolók és ÉS –VAGY kombinációs hálózat segítségével tervezze meg és rajzolja fel egy 3 bites szinkron számláló MEALY - MODELL szerinti logikai kapcsolási rajzát, amely a következő sorrendben számlál: 0, 1, 2, 3, 4, 5 . Ezután ismétlődik. A belső állapotokat Q0, Q1, Q2, a tároló bemeneteket pedig D0, D1, D2 szimbólumokkal jelölje.
4) D tárolók és ÉS –VAGY kombinációs hálózat segítségével tervezze meg és rajzolja fel egy 3 bites szinkron számláló MEALY - MODELL szerinti logikai kapcsolási rajzát, amely a következő sorrendben számlál: 0, 1, 3, 5 . Ezután ismétlődik. A belső állapotokat Q0, Q1, Q2, a tároló bemeneteket pedig D0, D1, D2 szimbólumokkal jelölje.
5) D tárolók és ÉS –VAGY kombinációs hálózat segítségével tervezze meg és rajzolja fel egy 3 bites szinkron számláló MEALY - MODELL szerinti logikai kapcsolási rajzát, amely a következő sorrendben számlál: 0, 1, 3, 5, 6. Ezután ismétlődik. A belső állapotokat Q0, Q1, Q2, a tároló bemeneteket pedig D0, D1, D2 szimbólumokkal jelölje.
6) D tárolók és ÉS –VAGY kombinációs hálózat segítségével tervezze meg és rajzolja fel egy 3 bites szinkron számláló MEALY - MODELL szerinti logikai kapcsolási rajzát, amely a következő sorrendben számlál: 0, 1, 2, 3, 5, 6, 7 . Ezután ismétlődik. A belső állapotokat Q0, Q1, Q2, a tároló bemeneteket pedig D0, D1, D2 szimbólumokkal jelölje.
7) D tárolók és ÉS –VAGY kombinációs hálózat segítségével tervezze meg és rajzolja fel egy 3 bites szinkron számláló MEALY - MODELL szerinti logikai kapcsolási rajzát, amely a következő sorrendben számlál: 1, 2, 3, 5 . Ezután ismétlődik. A belső állapotokat Q0, Q1, Q2, a tároló bemeneteket pedig D0, D1, D2 szimbólumokkal jelölje.
8) D tárolók és ÉS –VAGY kombinációs hálózat segítségével tervezze meg és rajzolja fel egy 3 bites szinkron számláló MEALY - MODELL szerinti logikai kapcsolási rajzát, amely a következő sorrendben számlál: 0, 1, 2, 5, 6 . Ezután ismétlődik. A belső állapotokat Q0, Q1, Q2, a tároló bemeneteket pedig D0, D1, D2 szimbólumokkal jelölje.

9) D tárolók és ÉS –VAGY kombinációs hálózat segítségével tervezze meg és rajzolja fel egy 3 bites szinkron számláló MEALY - MODELL szerinti logikai kapcsolási rajzát, amely a következő sorrendben számlál: 0, 1, 5, 6, 7 . Ezután ismétlődik. A belső állapotokat Q0, Q1, Q2, a tároló bemeneteket pedig D0, D1, D2 szimbólumokkal jelölje.
10) D tárolók és ÉS –VAGY kombinációs hálózat segítségével tervezze meg és rajzolja fel egy 3 bites szinkron számláló MEALY - MODELL szerinti logikai kapcsolási rajzát, amely a következő sorrendben számlál: 0, 3, 5, 6, 7 . Ezután ismétlődik. A belső állapotokat Q0, Q1, Q2, a tároló bemeneteket pedig D0, D1, D2 szimbólumokkal jelölje.

7.7. Irodalom

Kóré László: Digitális elektronika I. (BMF 1121)

Zsom Gyula: Digitális technika I. (Műszaki Könyvkiadó, Budapest, 2000, KVK 49-273/I, ISBN 963 6 1786 6)

Zsom Gyula: Digitális technika II. (Műszaki Könyvkiadó, Budapest, 2000, KVK 49-273/II, ISBN 963 16 1787 4)

Arató Péter: Logikai rendszerek tervezése (Tankönyvkiadó, Budapest, 1990, Műegyetemi Kiadó 2004, 55013)

Zalotay Péter: Digitális technika (http://www.kobakbt.hu/jegyzet/DigitHW.pdf)

Rőmer Mária: Digitális rendszerek áramkörei (Műszaki Könyvkiadó, Budapest, 1989, KVK 49-223)

Rőmer Mária: Digitális technika példatár (KKMF 1105, Budapest 1999)

Matijevics István: Digitális Technika Interaktív példatár (ISBN 978-963-279-528-7 Szegedi Tudományegyetem)

http://www.inf.u–szeged.hu/projectdirs/digipeldatar/digitalis_peldatar.html