ismerd meg!

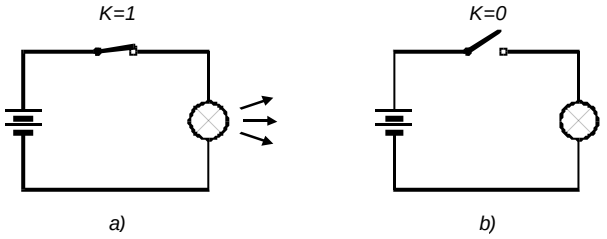
A PC vagyis a személyi számítógép

III. rész

A számítógép belsõ felépítését, a fontosabb alapegységek mûködését a továbbiakban csak úgy érthetjük meg, ha a digitális logikai áramkörök elméletében is kissé járatosak vagyunk. A számítógép bármely alapegységét az integrált áramköri technológiának köszönhetõen egy néhány nagy integráltsági fokú digitális logikai áramkörrel valósítják meg.

Logikai alapkapcsolások és a Boole-algebra

Egy bonyolult integrált logikai áramkör mûködését úgy elemezhetjük részletesebben, ha kisebb funkcionális alegységekre bontjuk fel. Ezeket a funkcionális alegységeket, gyûjtõ fogalommal, *logikai hálózatok*nak nevezzük. Egy logikai hálózat megvalósítható, bármilyen bonyolult is legyen, néhány logikai alapkapcsolás megfelelõ kombinációjával. A logikai kapcsolások elméletét *George Boole* (1815-1864) angol matematikus által kifejlesztett *logikai algebra* írja le, amelyet szerzõjérõl *Boole-algebrá*nak is szokás nevezni. A logikai algebra tulajdonképpen a kétértékû változós, kétértékû függvények algebrája. A logikai *változó*kat, vagyis eseményeket *logikai jelek*nek is nevezzük. Logikailag két eset lehetséges: vagy bekövetkezik az esemény, vagy nem. Ha bekövetkezik, akkor logikai értéke *igaz* és ezt az egyszerûség kedvéért *„1”*-gyel jelöljük. Ha nem következik be az esemény, akkor logikai értéke *hamis* és ezt *„0”*-val jelöljük. Megjegyezzük, hogy a logikai algebrában a 0 és az 1 nem szám, hanem csak célszerû szimbólumok, amelyekkel a kijelentések közötti összefüggéseket képletbe lehet foglalni.



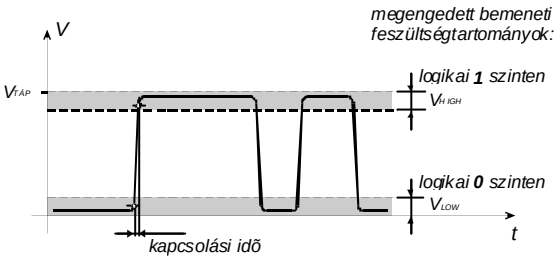
1. ábra A logikai szimbólumok egyszerû áramköri átültetése

	a)	*b)*
kapcsoló:	be	ki
égõ:	világít	nem világít
	K = 1	K = 0
logikai:	igaz	hamis
		(nem igaz)

A logikai szimbólumok egyfajta áramköri átültetését egy egyszerû kétállapotú áramkörbe az 1. ábra szemlélteti. A digitális logikai áramkörökben a kapcsoló szerepét

tranzisztor tölti be, többnyire szigetelt kapus térvezérlésû tranzisztor (MOSFET). Általában a digitális logikai áramkörök alapvetõ építõeleme a kapcsoló üzemmódban mûködõ félvezetõ, amely két állapottal rendelkezik, az egyik a kikapcsolt, míg a másik a bekapcsolt állapot. A logikai áramkör kimenetén e két állapotnak két jól eltérõ feszültségszint-tartomány felel meg: az egyik kis, a nullához közeli, a másik pedig nagy, a tápfeszültséghez közeli feszültségszint-tartomány. A logikai áramkörök jellegzetes hullámalakját a 2. ábrán láthatjuk. Jól megfigyelhetjük a logikai értékeknek megfelelõ két jól elkülöníthetõ feszültségszint-tartományt. A nullához közeli feszültségszint-tartomány (V_{LOW}) a logikai „0”-át vagyis a változó hamis értékét képviseli. A tápfeszültséghez közeli feszültségszinttartomány (V_{HIGH}) a logikai „1”-et, vagyis a változó igaz értékét képviseli.

A logikai hálózat a bemeneten kapott jelekbõl elõállítja azokat a kimenõ jeleket, amelyek a hálózat rendszerbeni funkcionális követelményeinek megfelelnek. Általános esetet véve, *n* bemenõ változóból, legyenek ezek $x_1, x_2, \ldots x_n$, elõállít *m* kimenõ változót, legyenek ezek $y_1, y_2, \ldots y_m$.



2. ábra
Logikai áramkörök jellegzetes hullámalakja

A logikai hálózatokat két nagy csoportba sorolhatjuk: *kombinációs* és *szekvenciális logikai hálózatok*.

Kombinációs logikai hálózatok

A kombinációs logikai hálózatok alapvetõ jellemvonása, hogy a kimenõ változók adott idõpontban levõ értékei, a bemenõ változók ugyanabban az idõpontban levõ értékeibõl származnak (3. ábra). A kimenõ és bemenõ változók közötti összefüggést a hálózatra jellemzõ logikai függvények határozzák meg. A Boole-algebrában sokszor alkalmazzák a matematikában ismert, értéktáblázattal történõ függvényleírást is, ezt *igazságtáblázat*nak nevezik.

3. ábra
Kombinációs logikai hálózat tömbvázlata

A Boole-algebra rámutat arra, hogy bármilyen bonyolult is legyen egy kombinációs logikai hálózat függvénye, az kifejezhetõ néhány logikai alapmûvelet segítségével. Tapasztalat szerint egy logikai függvény akkor legát-tekinthetõbb, ha a következõ három *logikai alapmûvelet*tel fejezzük ki: *ÉS* (*AND*), *VAGY* (*OR*) valamint *NEM* (*NOT*). Ezeket alapfüggvényeknek is nevezik és az 1. táblázat foglaltuk össze. Megjegyezzük, hogy az ÉS valamint a VAGY mûvelet több változóra is alkalmazható nemcsak kettõre.

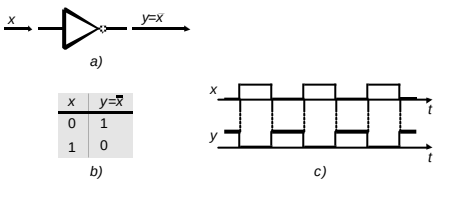
Logikai alapmûvelet	Jelölés	Egyéb elnevezés	Egyéb jelölés
NEM	$y = \overline{x}$	negáció, invertálás, komplementálás	
ÉS	$y = x_1 \cdot x_2$	konjunkció	$\wedge$, $\cap$
VAGY	$y = x_1 + x_2$	diszjunkció	$\vee$, $\cup$

1. táblázat A logikai alapmûveletek

Az alapmûveletet megvalósító áramköri építõelemet *logikai kapu*nak nevezik (4., 5. és 6. ábra). A kapu elnevezésnek megvan a magyarázata, ugyanis az egyik bemenõjelet a másikkal a kimenetre lehet „kapuzni". Az egyik jelet, a „kapuzott" bemenõjelet, csak akkor kapjuk meg a kimeneten, ha a másiknak, a „kapuzó" jelnek, egy adott logikai értéke van. Például az:

ÉS kapu esetében: $x_2 = 0 \Rightarrow y = x_1 \cdot 0 = 0$ – a kimenet független x_1-tõl,
$x_2 = 1 \Rightarrow y = x_1 \cdot 1 = x_1$ – a kimeneten az x_1-et kapjuk;

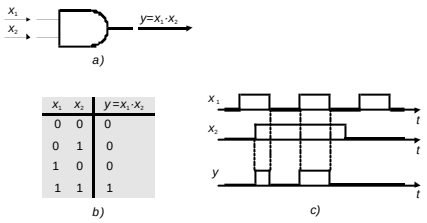
VAGY kapu esetében: $x_2 = 1 \Rightarrow y = x_1 + 1 = 1$ – a kimenet független x_1-tõl,
$x_2 = 0 \Rightarrow y = x_1 + 0 = x_1$ – a kimeneten az x_1-et kapjuk.



x	y=x̄
0	1
1	0

b)

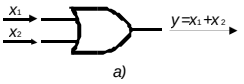
4. ábra
A NEM (NOT) kapu (inverter) rajzjele (*a*), igazságtáblázata (*b*) és jelalakja (*c*)



x1	x2	y=x1·x2
0	0	0
0	1	0
1	0	0
1	1	1

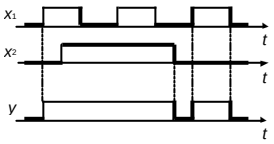
b)

5. ábra
Az ÉS (AND) kapu rajzjele (*a*), igazságtáblázata (*b*) és jelalakja (*c*)



x1	x2	y=x1+x2
0	0	0
0	1	1
1	0	1
1	1	1

b)



6. ábra
A VAGY (OR) kapu rajzjele (a), igazságtáblázata (b) és jelalakja (c)

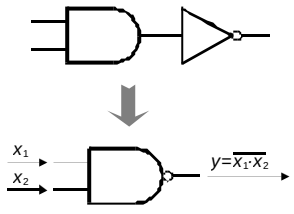
A logikai függvény áramköri átültetése logikai áramkörtervezéssel történik, ezáltal az adott logikai függvény kombinációs hálózatát határozzuk meg. Elsõ sorban a függvényt olyan alakúra kell hozni, hogy kizárólag csak a logikai alapmûveletek szerepeljenek benne. Ezután a hálózatot úgy kapjuk meg, hogy a logikai mûveleteknek megfelelõ

kapukat a függvény szerint kapcsoljuk össze. Áramköri szempontból az ÉS kapunál valamivel egyszerûbb a *NEM-ÉS* (*NAND*) kapu, és a VAGY kapunál pedig a *NEM-VAGY* (*NOR*) kapu. Ezek az alábbi logikai mûveleteket végzik:

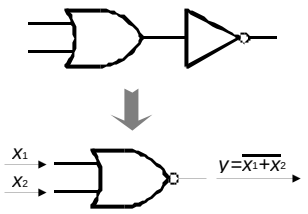
NEM-ÉS: $y = \overline{x_1 \cdot x_2}$

NEM-VAGY: $y = \overline{x_1 + x_2}$

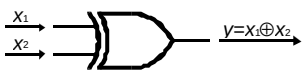
A fenti kifejezésekbõl láthatjuk hogy a NEM-ÉS és a NEM-VAGY kapu elvileg egy ÉS ill. egy VAGY kapuból áll, melyet egy inverter követ (7. és 8. ábra). Be van bizonyítva, hogy bármely logikai függvény kifejezhetõ a NEM-ÉS, a NEM-VAGY (NOR) és a NEM (NOT) függvényekkel is. Tehát a logikai függvények áramköri átültetésénél e három függvényt szintén olyan jól lehet alkalmazni, mint a három alapfüggvényt. Léteznek elterjedtebb logikai mûveleteket végzõ kapuk is. Ezekkel a kombinációs hálózatok tervezése és megvalósítása egyszerûbbé válik. Ilyen például a *KIZÁRÓ-VAGY* (*XOR — EXCLUSIVE-OR*) kapu (9. ábra). A kizáró-vagy mûvelet jelölése: $y = x_1 \oplus x_2$ antivalencia mûveletnek is nevezik, mivel a kimenete csak akkor igaz, ha a bemenõ változók ellentétesek.



7. ábra
A NEM-ÉS (NAND) kapu =
= ÉS kapu + inverter



8. ábra
A NEM-VAGY (NOR) kapu =
= VAGY kapu + inverter

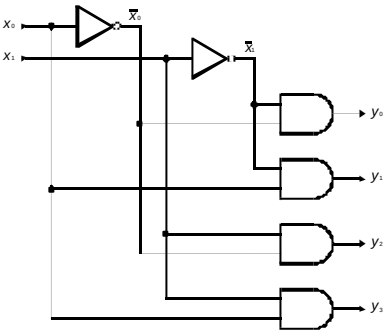


a)

x_1	x_2	$y=x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

b)

9. ábra A KIZÁRÓ-VAGY (XOR) kapu
rajzjele (*a*) és igazságtáblázata (*b*)



10. ábra
4-bõl 1 dekódoló

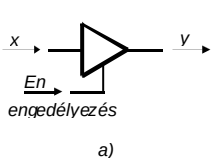
Kombinációs logikai hálózatokról a 10. ábrán bemutatott példával alkothatunk fogalmat. Ez egy 4-bõl 1 dekódoló, amelynek 4 kimenete és 2 bemenete van. A bemenõ változók az $X = x_1 x_0$ szám bináris kódját ábrázolják. A kapcsolásnak annyi kimenete van, amennyit meghatároz az X-szel ábrázolható bináris szám. Ebben az esetben négy kimenet van, amelyeket 0-tól 4-1=3-ig számozunk. Az áramkör mûködését igazságtáblázata alapján követhetjük (2. táblázat). Egy meghatározott kimenet akkor lesz logikai 1 értékû, ha a bemenetre adott X bináris szám értéke egyenlõ az illetõ kimenet

sorszámával, egyébként logikai 0. Tehát az y_x kimeneten logikai 1-et kapunk amíg a többin 0-át.

X	$x_1\ x_0$	$\overline{x}_1\ \overline{x}_0$	$y_3 = x_1 \cdot x_0$	$y_2 = x_1 \cdot \overline{x}_0$	$y_1 = \overline{x}_1 \cdot x_0$	$y_0 = \overline{x}_1 \cdot \overline{x}_0$
0	0 0	1 1	0	0	0	1
1	0 1	1 0	0	0	1	0
2	1 0	0 1	0	1	0	0
3	1 1	0 0	1	0	0	0

2. táblázat A 4-bõl 1 dekódoló igazságtáblázata

A sorozatunk elsõ részében láthattuk, hogy a számítógépen belül, az építõegységek közötti információáramlás, az azokat összekötõ busz- vagy sínrendszeren keresztül valósul meg. A sínrendszer adatátviteli vonalain az információt nemcsak egy irányban vihetjük át, hanem mindkét irányban. Ebben nagyon fontos szerepet töltenek be az ún. *háromállapotú kapu*k. Egy háromállapotú kapu kimenetén – ahogy az elnevezése is mutatja – nemcsak a két logikai szintnek megfelelõ két állapot található, hanem egy harmadik is (11. ábra). A harmadik, ún. nagyimpedanciás állapotban a háromállapotú kapu kimenete elveszíti a szokásos logikai kapukra jellemzõ tulajdonságokat. Így a logikai szintek biztosítására nem szolgáltat sem feszültséget, sem áramot. Ez funkcionálisan azt eredményezi, mintha a kapu kimenetét lekapcsolnánk a meghajtott kimeneti vonalról. Ilyenkor a kapu kimenete le van tiltva és ezt az állapotot az engedélyezõ bemenetre adott logikai 0-ával érjük el. Ha az engedélyezõ bemenetre logikai 1-et adunk, akkor a kimenetet engedélyezzük és rajta a bemenõ jeltõl függõ kimenõjelet kapunk.



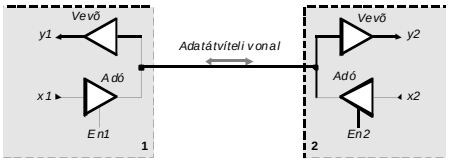
a)

En	x	y
0	0	TS*
0	1	TS*
1	0	0
1	1	1

b)

* TS – harmadik állapot

11. ábra
A háromállapotú (tri-state) kapu rajzjele (*a*) és igazságtáblázata (*b*)



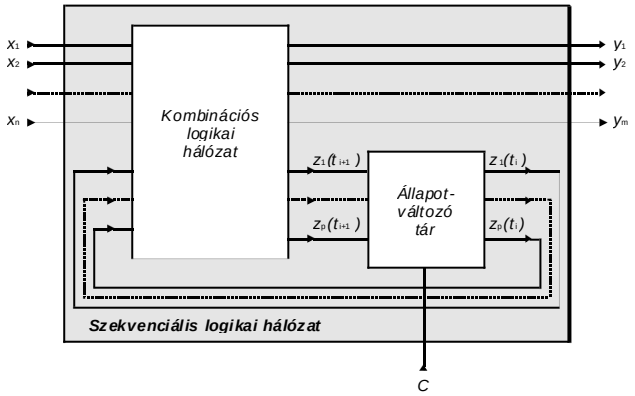
12. ábra
Kétirányú adatátvitel háromállapotú kapukkal

A háromállapotú kapuk szerepét a 12. ábrán látható, két egység közötti adatátviteli példával szemléltetjük. Abban az esetben ha *En1*=1 és *En2*=0, akkor az **1.** egységben levõ háromállapotú kapu az adatátviteli vonalat meghajtja, míg a **2.** egységben levõ háromállapotú kapu azt felszabadítja vezérlése alól. Ezáltal a **2.** egység vevõje veszi az **1.** egységben levõ háromállapotú kapu által kiküldött adatot, vagyis: *y2=x1.* Ha *En2*=1 és *En1*=0, akkor a **2.** egységben levõ háromállapotú kapu hajtja meg az adatátviteli vonalat, míg az **1.** egységben levõ háromállapotú kapu felszabadítja azt vezérlése alól. Ekkor az **1.** egység vevõje veszi a **2.** egységben levõ háromállapotú kapu által küldött adatot, vagyis: *y1=x2.* Végülis, ha *En2*=0 és *En1*=0 az adatátviteli vonal felszabadul mindkét egység vezérlése alól, és ezáltal lehetõvé válik egy hasonló, harmadik egységnek a vonalra való kapcsolása.

Szekvenciális logikai hálózatok

A szekvenciális, vagyis a sorrendi logikai hálózatok kimenõ változói, a kombinációs logikai hálózatoktól eltérõen, nemcsak a bemenõ változók adott idõpontban levõ értékeitõl, hanem a hálózat belsõ állapotától is függenek. A hálózat belsõ állapotát a bemenõ változók korábbi értéksorozata határozza meg. Szinkron és aszinkron típusú sorrendi hálózatokat különböztetünk meg. A szinkron sorrendi hálózatok állapota egy *C* órajel ütemére változik meg, az órajellel szinkronban (13. ábra). Az aszinkron sorrendi hálózat abban különbözik a szinkron sorrendi hálózattól, hogy állapotváltozása nincs ütemjelhez kötve.

Az órajel által elõre meghatározott szabályos idõpontokat t_i -vel szokták jelölni, amelyben az *i* index az órajelütem sorszámát fejezi ki. A t_i idõpontnak megfelelõ állapotot S_i -vel jelöljük, amelyet a hálózat belsõ tárában tárolt $z_1(t_i)$, $z_2(t_i)$, … $z_p(t_i)$ jelek, ún. állapotváltozók képviselnek. Ez az S_i állapot a soronkövetkezõ t_{i+1} idõpontig megmarad, amikor a hálózat az órajelütem hatására az S_{i+1} állapotba kerül. Ezt a $z_1(t_{i+1})$, $z_2(t_{i+1})$, … $z_p(t_{i+1})$ állapotváltozók határozzák meg, melyeket a szekvenciális hálózat belsõ kombinációs hálózata állított elõ az ezelõtti $z_1(t_i)$, $z_2(t_i)$, … $z_p(t_i)$ állapotváltozókból, valamint az x_1, x_2, … x_n bemenõ változókból. Az y_1, y_2, … y_m kimenõ változókat ugyancsak a belsõ kombinációs logikai hálózat állítja elõ az x_1, x_2, … x_n bemenõ változókból és a $z_1(t_i)$, $z_2(t_i)$, … $z_p(t_i)$ állapotváltozókból.



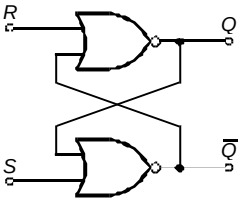
13. ábra Szekvenciális logikai hálózat tömbvázlata

A szekvenciális hálózatok csoportjába tartoznak a *bistabil billenõáramkör*ök. Bistabil multivibrátoroknak vagy az angol megfelelõjével *flip-flop*-oknak is szokták õket nevezni. A flip-flop a szekvenciális hálózat tárának alapvetõ építõeleme. A flip-flop olyan szekvenciális áramkör, amely két ellentétes állapottal rendelkezik és külsõ beavatkozás nélkül bármelyiket megtartja. Egy vagy több bemenettel rendelkezik, amelyek lehetõvé teszik az áramkör egyik vagy másik állapotba való billentését. A legegyszerûbb felépítésû flip-flop két keresztbecsatolt NEM-ÉS vagy NEM-VAGY kapuból alakítható ki (14. ábra). Az áramkör két vezérlõ bemenettel rendelkezik: az egyik az *S* (*S*et: beíró) bemenet és a másik az *R* (*R*eset: törlõ) bemenet, ezért *RS flip-flopnak* nevezik és a legegyszerûbb aszinkron sorrendi hálózat. A *Q* és a $\overline{Q}$ komplemens kimenetek két ellentétes állapotot határoznak meg:

$Q = 1, \quad \overline{Q} = 0 \quad \Rightarrow$ a flip-flopba logikai 1 van beírva – beállított állapot,

$Q = 0,\quad \overline{Q} = 1 \quad \Rightarrow$ a flip-flopba logikai 0 van beírva – törölt állapot.

A felsõ és az alsó kapu kimenete a NEM-VAGY mûveletek szerint:

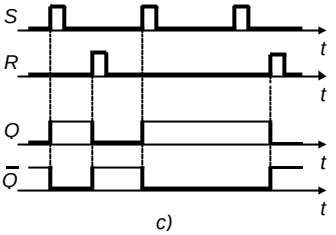


$Q_{i+1} = \overline{R + \overline{Q_i}}$ ill. $\overline{Q_{i+1}} = \overline{S + Q_i}$.

a)

R	S	Q_{i+1}	$\overline{Q}_{i+1}$
0	0	Q_i	$\overline{Q}_i$
0	1	1	0
1	0	0	1
1	1	*	*

b)



c)

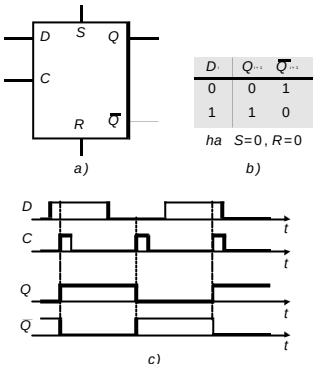
14. ábra RS flip-flop (*a*) igazságtáblázata (*b*) és jellegzetes hullámalakja (*c*)

Logikai 1 beírásánál (röviden beírás), amikor $S = 1$ és $R = 0$, az alsó kapu kimenete $\overline{Q}_{i+1} = \overline{1 + Q_i} = \overline{1} = 0$ lesz, függetlenül a felsõ kapu Q_i kimenetétõl, míg a felsõ kapu kimenete $Q_{i+1} = \overline{0 + \overline{1}} = \overline{0} = 1$. Törlésnél, amikor $R = 1$ és $S = 0$, a felsõ kapu kimenete $Q_{i+1} = \overline{1 + \overline{Q}_i} = \overline{1} = 0$ lesz, függetlenül a felsõ kapu $\overline{Q}_i$ kimenetétõl, míg az alsó kapu kimenete $\overline{Q_{i+1}} = \overline{0 + 0} = \overline{0} = 1$. Az áramkör részletesebb tanulmányozása után észrevehetjük, hogy a keresztbecsatolás egy pozitív visszacsatolást eredményez, amely az átbillenést gyorsítja. Megjegyezzük, hogy a vezérlõ bemenetek nem végezhetnek egyidejûleg beírást és törlést is, vagyis $S = 1$ és $R = 1$ tiltott vezérlési állapot. Ha nincs sem beírás ($S = 0$), sem törlés ($R = 0$), akkor a két kimenet a fenti logikai mûveletek szerint:

$Q_{i+1} = \overline{0 + \overline{Q_i}} \quad \Rightarrow \quad \overline{0 + \overline{Q_i}} = \overline{\overline{Q_i}} = Q_i$ és $\overline{Q_{i+1}} = \overline{0 + Q_i} \quad \Rightarrow \quad \overline{0 + Q_i} = \overline{Q}_i$.

vagyis változatlan marad.

Az egyszerû RS flip-flop bonyolult, órajelvezérlésû flip-flopok felépítésére szolgál. Ilyen az *órajelvezérlésû RS flip-flop*, a *JK mester-szolga (master-slave) flip-flop* és a *D flip-flop*, hogy csak a legismertebb típusokat soroljuk fel. Egyik leggyakrabban alkalmazott flip-flop a D flip-flop. Rajzjelét, igazságtáblázatát és tipikus hullámalakját a 15. ábrán láthatjuk. A D flip-flop állapota a *C* órajel felfutó élével szinkronban változik meg. Ekkor a *Q* és $\overline{Q}$ kimenet a *D* adatbemenet logikai értékét ill. invertált értékét veszi át. Órajelimpulzusok közötti adatbemenetváltozás nincs hatással a kimenetekre. Egyes áramköri alkalmazásban felmerül, hogy a flip-flopot nemcsak az órajellel szinkronban, hanem akármikor lehessen állítani. Erre a célra közvetlen beíró (*S*) és törlõ (*R*) bemenetek állnak rendelkezésünkre. Ezek éppen úgy vezérlik az áramkört, mint a fentiekben tárgyalt egyszerû RS flip-flopot. A flip-flop csak akkor mûködhet az órajelimpulzus vezérlése alatt, ha a közvetlen vezérlõ bemeneteken logikai 0 szintet biztosítunk.

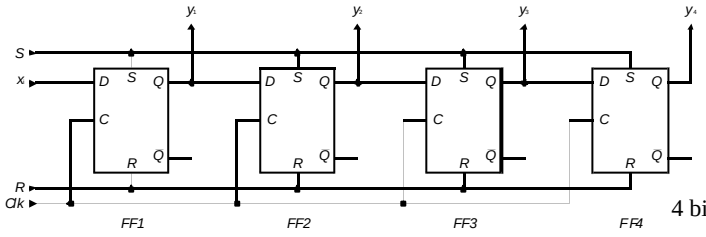


15. ábra
D flip-flop (*a*) igazságtáblázata (*b*) és jellegzetes hullámalakja (*c*)

i	Y_1	Y_2	Y_3	Y_4
1	X_1	0	0	0
2	X_2	X_1	0	0
3	X_3	X_2	X_1	0
4	$\boldsymbol{X_4}$	$\boldsymbol{X_3}$	$\boldsymbol{X_2}$	$\boldsymbol{X_1}$
5	X_5	X_4	X_3	$\boldsymbol{X_2}$
6	X_6	X_5	X_4	$\boldsymbol{X_3}$
7	X_7	X_6	X_5	$\boldsymbol{X_4}$

3. táblázat
A 4 bites léptetõregiszter mûködési táblázata

A D flip-flopok alkalmazásáról a 16. ábrán látható szekvenciális hálózati példával alkothatunk fogalmat. Ez egy 4 bites léptetõregiszter, amelynek mûködését a 3. táblázaton követhetjük nyomon.



16. ábra
4 bites léptetõregiszter

Általában a léptetõregiszter flip-flopok olyan lánca, amely lehetõvé teszi, hogy a bemenetére adott adat, minden egyes órajelimpulzus hatására egy flip-floppal tovább lépjen. A bemeneti jel áthaladva a flip-flopok láncán, késleltetve, de változatlanul jelenik meg a kimeneten. Kiinduláskor az összes flip-flopot az *R* bemenetre kapcsolt 1 logikai jellel töröljük. Ezután az órajel minden egyes ütemére az elsõ *FF1* flip-flop a bementi x_1 adatot olvassa be, a többi pedig az elõtte levõ flip-flop tartalmát veszi át. Tehát az órajel elsõ ütemére az x_1 -et az elsõ *FF1* flip-flop olvassa be. A második ütemben a beolvasott adatot átadja a második *FF2* flip-flopnak és egyidejûleg beolvassa az újabb x_2 bementi adatot. Tehát a regiszterben az adatok minden egyes órajel hatására balról jobbra egyet lépnek. A negyedik órajel után a sorosan beírt adatokkal a regiszter megtelik. A soros információ párhuzamos alakban leolvasható a négy flip-flop kimenetén. A további órajel impulzusok hatására az információ az utolsó *FF4* flip-flop kimenetén sorosan jelenik meg.

A fenti regiszterbe az adatokat sorosan írjuk be és párhuzamosan vagy sorosan olvashatjuk ki. A soros információ esetében az adatbitek sorban, idõben egymásután következnek míg a párhuzamos információnál az összes adatbitet egyidejûleg, egyszerre kapjuk meg. Megjegyezzük, hogy léteznek párhuzamos beírású léptetõregiszterek is. A regiszterekkel még fogunk találkozni, ugyanis a mikroprocesszorok belsõ alapvetõ tároló elemei.

Kaucsár Márton