Laboratorul 0 AC - Recapitulare

1. Sisteme de numerație

Un număr **N** de forma $\mathbf{c_nc_{n-1}...c_2c_1c_0}$ reprezentat în baza 10 are valoarea:

$$N = c_n \cdot 10^n + c_{n-1} \cdot 10^{n-1} + ... + c_2 \cdot 10^2 + c_1 \cdot 10^1 + c_0 \cdot 10^0$$

ex: $12345 = 1 \cdot 10^4 + 2 \cdot 10^3 + 3 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0$

Baza unui sistem de numerație poate fi însă orice întreg **b ≥ 2**, ponderea cifrei din poziția **i** fiind în acest caz $\mathbf{b^i}$. În acest caz valoarea unui număr $\mathbf{c_nc_{n-1}...c_2c_1c_{0(b)}}$ reprezentat în baza **b** este:

$$N = c_n \cdot b^n + c_{n-1} \cdot b^{n-1} + ... + c_2 \cdot b^2 + c_1 \cdot b^1 + c_0 \cdot b^0 \quad (1)$$

În circuitele digitale, semnalele pot avea, în mod normal, una din cele doua stări posibile. Aceste stări sunt asociate cifrelor binare 0 și 1, ceea ce face ca baza 2 să fie o alegere naturală pentru reprezentarea numerelor în sistemele digitale.

ex: $100101_{(2)} = 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 37_{(10)}$

O altă bază folosită în sistemele digitale este baza 16, deoarece permite o reprezentare mai facilă a numerelor decât baza 2, dealtfel și conversia între baza 2 și baza 16 fiind foarte rapidă. Cifrele folosite în baza 16 sunt, în ordinea crescătoare a valorii, 0..9 plus literele A, B, C, D, E și F.

ex: $1EE7_{(16)} = 1 \cdot 16^3 + 14 \cdot 16^2 + 14 \cdot 16^1 + 7 \cdot 16^0 = 7911_{(10)}$

Pentru a găsi reprezentarea unui număr într-o bază **b** este necesar să determinăm cifrele $\mathbf{c_nc_{n-1}...c_2c_1c_0}$ care satisfac ecuația (1), ținând cont de faptul că $\mathbf{c_i} < \mathbf{b}$. Rearanjând ecuația (1) obținem:

$$N = \left(c_n \cdot b^{n-1} + c_{n-1} \cdot b^{n-2} + ... + c_2 \cdot b^1 + c_1 \cdot b^0\right) \cdot b + c_0$$

Se observă că tot ce trebuie să facem să aflăm pe c_0 este să împărțim numărul de convertit la b, c_0 fiind chiar restul împărțirii. Câtul rezultat va fi apoi împarțit din nou la b pentru a afla pe c_1 s.a.m.d. pentru celelate cifre $c_2,...,c_n$.

ex: $37_{(10)} = 100101_{(2)}$

	Cât	Rest	Cifra
37 / 2	18	1	c_0
18 / 2	9	0	c_1
9 / 2	4	1	c_2
4 / 2	2	0	c_3
2 / 2	1	0	c_4
1 / 2	0	1	c_5

$7911_{(10)} = 1EE7_{(16)}$

	Cât	Rest	Cifra
7911 / 16	494	7	c_0
494 / 16	30	14	c_1
30 / 16	1	14	c_2
1 / 16	0	1	c_3

În cazul în care între baza sursă $\mathbf{b_s}$ și baza destinație $\mathbf{b_d}$ există o relație de forma $b_s = b_d^r$ sau $b_d = b_s^r$ conversia între cele doua baze se poate face mult mai ușor având în vedere că fiecare cifră a bazei mai mari poate fi scrisă ca un grup de **r** cifre ale bazei mai mici.

ex: $16 = 2^4$

$1EE7_{(16)} = 0001\ 1110\ 1110\ 0111_{(2)}$

$100101_{(2)} = 25_{(16)}$

2. Algebra booleană

Algebra booleană este o algebră formată din:

- elementele 0 și 1
- 1 operație unară: NU (NOT), notată simbolic cu (!, sau $\bar{\ }$)
- 2 operații binare: ȘI (AND) și SAU (OR), notate simbolic cu (&, ∧ sau ·), respectiv (|, ∨ sau +)

Operațiile sunt definite astfel:

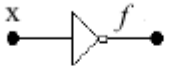
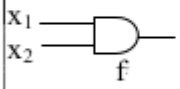
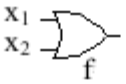
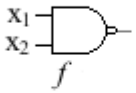
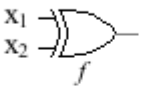
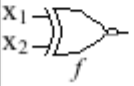
NU	ȘI	SAU
!0 = 1 !1 = 0	0 & 0 = 0 0 & 1 = 0 1 & 0 = 0 1 & 1 = 1	0 | 0 = 0 0 | 1 = 1 1 | 0 = 1 1 | 1 = 1

Din aceste definiții rezultă o serie de teoreme ale algebrei booleene dintre care amintim câteva mai uzuale:

- Principiul dublei negații:
  - !(!x) = x
- Legile lui De Morgan:
  - !(x & y) = !x | !y
  - !(x | y) = !x & !y
- Elemente neutre:
  - x & 1 = 1 & x = x
  - x | 0 = 0 | x = x

3. Porți logice

Porțile logice se folosesc în circuitele digitale pentru a implementa operațiile din algebra booleană.
În tabelul următor sunt prezentate cele mai înlâlnite porți logice împreună cu funcția booleană pe care o implementează:

Denumire	Funcţie	Simbol	Tabel de adevăr
Inversor (NOT)	$f = \overline{x}$		x | f 0 | 1 1 | 0
Poartă ŞI (AND)	$f = x_1 \cdot x_2$		x_1 | x_2 | f 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1
Poartă SAU (OR)	$f = x_1 + x_2$		x_1 | x_2 | f 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1
Poartă ŞI-NU (NAND)	$f = \overline{x_1 \cdot x_2}$		x_1 | x_2 | f 0 | 0 | 1 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0
Poartă SAU-NU (NOR)	$f = \overline{x_1 + x_2}$		x_1 | x_2 | f 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 0
SAU EXCLUSIV (XOR)	$f = x_1 \oplus x_2$		x_1 | x_2 | f 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0
SAU EXCLUSIV NU (NXOR)	$f = \overline{x_1 \oplus x_2}$		x_1 | x_2 | f 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1

Cu $\oplus$ s-a notat operația de SAU EXCLUSIV, definită ca x $\oplus$ y = (!x & y) | (x & !y). O altă notație pentru operația de SAU EXCLUSIV care mai poate fi întâlnită este simbolul ^.

4. Reprezentarea funcțiilor logice

Pe lângă expresia booleană (coloana 2) sau schema porților logice (coloana 3) o funcție booleană mai poate fi descrisă și prin tabelul ei de adevăr. Această reprezentare poate fi observată în ultima coloană a tabelului anterior. **Tabelul de adevăr** al unei funcții cu **n** parametri conține **n+1** coloane și, în general, $\mathbf{2^n}$ rânduri. Pentru fiecare rând, primele **n** coloane conțin câte o combinație de **0** și **1** pentru parametrii funcției, iar ultima coloană conține valoarea funcției pentru acea combinație a parametrilor. În anumite cazuri, tabelul de adevăr al unei funcții poate conține și **x**. Dacă simbolul **x** se găsește în coloanele parametrilor, înseamnă că, pentru acea combinație a parametrilor funcției, parametrul respectiv nu afectează ieșirea funcției. În alte cuvinte parametrul poate avea atât valoare **0** cât și valoarea **1**, iar notația **x** este o prescurtare pentru această situație. Dacă simbolul **x** se găsește în coloana rezultatului, înseamnă că respectiva combinație de paramterii a funcției este invalidă, iar în acest caz rezultatul funcției nu este definit, el putând avea oricare dintre valorile **0** sau **1**, depinzând de implementarea funcției.

Pornind de la expresia booleană sau schema logică a unei funcții tabelul de adevăr se poate determina calculând valoarea funcției pentru fiecare combinație a parametrilor ei. În sens invers, pornind de la tabelul de adevăr, o expresie booleană a funcției poate fi determinată luând fiecare rând al tabelului pentru care funcția are valoarea 1 și formând termeni compuși din parametrii negați și direcți ai funcției legați prin ȘI. Parametrul se ia în forma directă dacă pe rândul considerat el are valoare 1 și în forma negată daca are valoare 0. Toți termenii corespunzători rândurilor unde funcția are valoarea 1 sunt apoi legați prin SAU. Această formă a unei funcții mai este numită și forma normal disjunctivă. A se observa că, în general, expresia astfel obținută nu este minimă, în sensul că ea conține mai multe operații decât sunt necesare pentru a descrie funcția. Pornind de la această formulă se pot aplica diferite metode de minimizare a ei: grafică (folosind digrame Karnaugh) sau algebrică (folosind teoremele algebrei booleene).

ex:

x	**y**	**z**	**f**
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

f(x,y,z) = (!x & !y & !z) | (!x & y & z) | (x & !y & !z) | (x & y & !z)

5. Circuite Logice Combinaționale

Circuitele logice combinaționale sunt circuitele reprezentate prin porți logice ce aplică o funcție pe intrări. Valorile de ieșire depind doar de valorile de intrare, nu și de stări de reactie (feedback), iar cand starea unei intrări se schimbă, se reflectă imediat pe ieșiri.

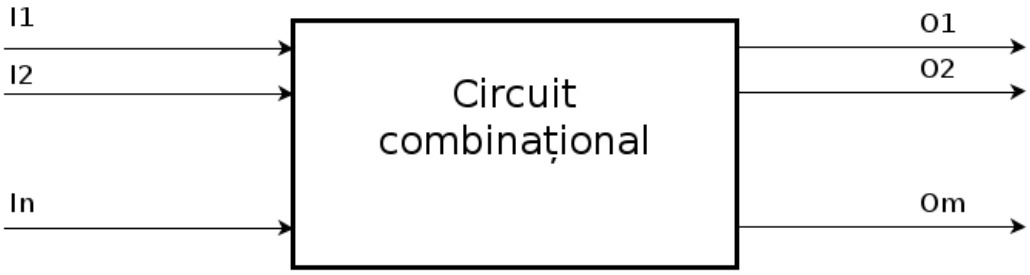


Figura 1: Circuitele combinaționale au 2 sau mai multe intrări si cel puțin o ieșire.

Un circuit combinațional poate fi descris prin schema cu interconectarea porților, prin funcția (expresia) în logica booleană ce este aplicată pe intrări și prin tabele de adevăr (dacă nu au un număr prea mare de intrări).

Circuitele combinaționale sunt folosite în procesoare în cadrul componente de calcul, iar cele mai des întalnite sunt:

- multiplexoare și demultiplexoare
- codificatoare și decodificatoare
- sumatoare
- comparatoare
- memorii ROM (read-only, nu păstrează stare)

De exemplu folosim sumatoare în cadrul Unităților Aritmetice-Logice (UAL) din procesoare.

5.1 Multiplexoare și Demultiplexoare

Un multiplexor digital este un circuit combinațional care implementează o funcție de selecție a uneia dintre intrările sale. Alegerea semnalului de ieșire se face pe baza unor intrări suplimentare care reprezintă în baza 2 numărul intrării ce trebuie selectate, astfel un multiplexor cu 2^n *intrări* va avea nevoie de *n semnale de selecție*. În exemplul din Figura 2 avem schema bloc a unui multiplexor cu 4 intrări, iar acesta are nevoie de doar două intrări de selecție. Figura 3 descrie circuitul combinațional pentru multiplexorul 4:1.

Un demultiplexor digital realizează funcția inversă multiplexorului, preia un semnal de intrare și îl rutează pe una din cele 2^n ieșiri folosind *n* intrări de selecție.

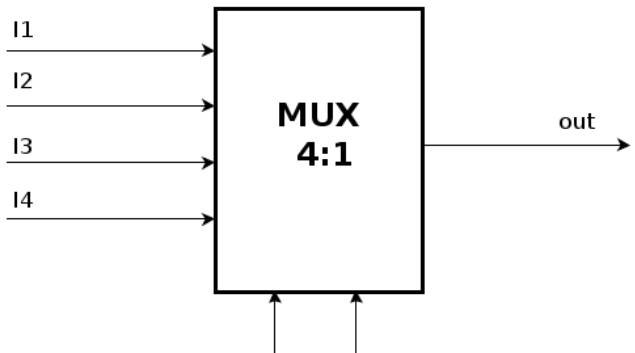


Figura 2: Schema bloc multiplexor 4:1

Funcția logică corespunzătoare multiplexorului 4:1 din exemplul de mai sus este următoarea:
$f = \overline{s_0} * \overline{s_1} * x_0 + s_0 * \overline{s_1} * x_1 + \overline{s_0} * s_1 * x_2 + s_0 * s_1 * x3$,
unde
$x_{0..3}\, intr[?][?]ri, s_{0,1}\, semnale\ selec[?][?]ie, y\ ie[?][?]irea\ , *\ reprezinta\ AND, +\ reprezint[?][?]\, OR$

Folosind această expresie putem descrie circuitul combinațional pentru multiplexor, având nevoie conform formulei de 4 porți logice AND cu trei intrări, o poartă OR și porți NOT pentru negarea semnalelor de selecție.

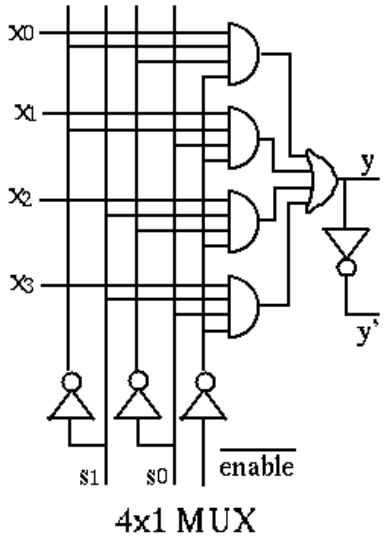


Figura 3: Implementarea folosind porți logice pentru un multiplexor 4:1 (sursa imagine)

Tabela de adevar în varianta nesimplificată pentru acest circuit este următoarea:

S_1	S_0	x_3	x_2	x_1	x_0	y
0	0	x	x	x	0	0
0	0	x	x	x	1	1
0	1	x	x	0	x	0

0	1	x	x	1	x	1
1	0	x	0	x	x	0
1	0	x	1	x	x	1
1	1	0	x	x	x	0
1	1	1	x	x	x	1

5.2 Codificatoare și decodificatoare

Codificatoarele (*encoders*) sunt circuite combinaționale care comprimă mai multe intrări binare într-un număr mai mic de ieșiri. De cele mai multe ori sunt folosite codificatoarele pe bază de prioritate (*priority encoders*) pentru a controla cererile de întrerupere către un procesor, datorită proprietății de a reacționa la intrări în funcție de prioritatea lor, punând întotdeauna pe ieșire codul intrării cu prioritatea cea mai mare.

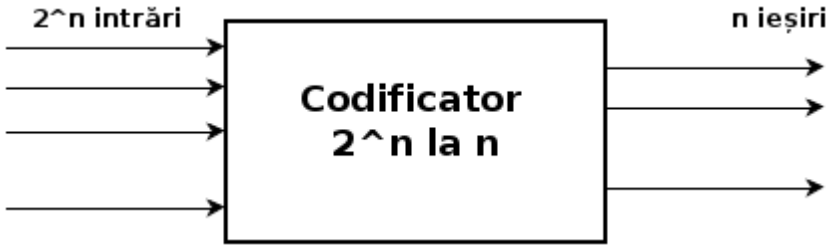


Figura 4: Schema bloc codificator

Decodificatoarele (*decoders*) realizează funcția inversă primind n intrări și furnizând 2^n ieșiri. Aceste circuite sunt utile în aplicații care folosesc multiplexarea datelor, display-uri 7-segment sau decodificarea adreselor de memorie. De exemplu, memoria principala (RAM) este organizată pe cuvinte, fiecare adresă practic indicând un cuvânt din memorie, astfel dimensiunea ei fiind limitată la 2^n locații, unde n este dimensiunea cuvântului. Deoarece memoriile sunt organizate intern din mai multe chipuri puse pe linii, coloane, pentru acesarea acestor locații de memorie se folosesc decodificatoare de adrese care iau parți din adresa și împreună vor da locația corespunzatoare.

5.3 Sumatoare

Sumatoarele (*adders*), folosite cel mai mult în unitățile aritmetice logice ale procesoarelor realizează adunări pe un număr dat de biți, furnizând la ieșirea circuitului suma și transportul (carry) rezultat în urma operației.
Există mai multe tipuri de sumatoare pentru adunarea numerelor pe *n* biți, iar acestea se bazează pe sumatoare simple de 1 bit ce pot fi de două tipuri:

- *Half Adder* - însumează doi operanzi pe 1 bit și oferă la ieșire suma acestora și un transport.
- *Full Adder* - însumează doi operanzi pe 1 bit și un transport și oferă la ieșire suma acestora trei și un transport. Sumatorul pe n biți Carry-lookahead folosește mai multe sumatoare de acest tip pentru a putea propaga transportul.

Tabela de adevăr pentru un sumator Half Adder este următoarea:

A	**B**	**S**	**C_{out}**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Formula ce poate fi dedusă din această tabelă este următoarea:

$$S = A \oplus B$$
$$C_{out} = A * B$$

Conform expresiilor logice pentru acest sumator, schema folosind porți este cea prezentată în figura 5.

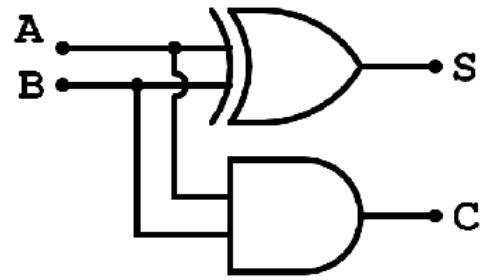


Figura 5. Half-Adder

Expresiile logice pentru un sumator Full Adder sunt prezentate mai jos, iar schema care le implementează este dată în figura 6:

$$S = (A \oplus B) \oplus C_{in}$$
$$C_{out} = (A * B) + (C_{in} * (A \oplus B))$$

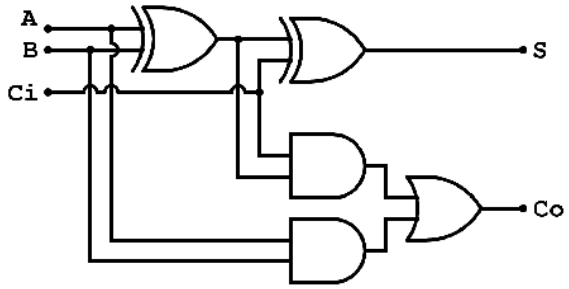


Figura 6. Full-Adder

6. Circuite logice secvențiale

Spre deosebire de circuitele logice combinaționale, cele secvențiale nu mai depind exclusiv de starea curentă a intrărilor, ci și de stările anterioare ale circuitului. După cum este arătat și în figura 7, circuitului combinațional format din porți logice i se adaugă un element de memorare a stărilor.

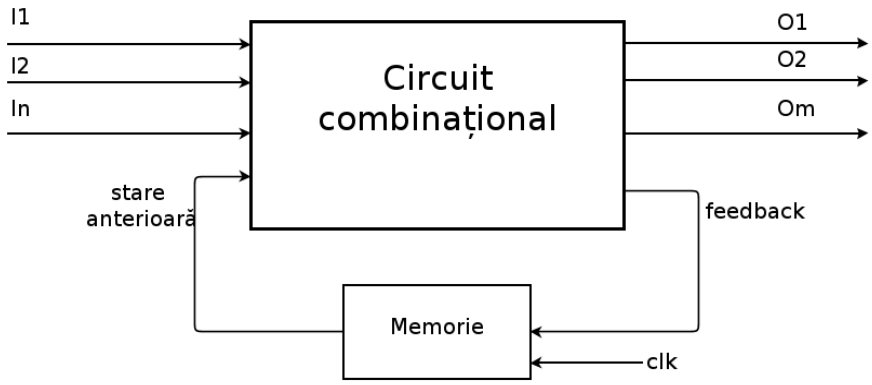


Figura 7. Circuit logic secvențial

6.1 Bistabili

Circuitele secvențiale sunt de două tipuri, asincrone si sincrone. Cele asincrone sunt rar folosite si țin cont de ordinea în care se schimbă intrările, elementul de memorare fiind de fapt căi de feedback. Circuitele sincrone se folosesc pentru memorarea stării (acel “blackbox” din figura 7) bistabili (*Flip-Flop*) ce își schimbă valoarea stocată doar la schimbarea semnalului de ceas primit la intrare. Una din utilizările bistabililor este în registrele procesoarelor, ce reprezintă cea mai mică si mai rapidă unitate de stocare din ierarhia de memorie. În continuare vom prezenta mai multe detalii despre aceste circuite și tipurile lor.

Bistabilii au următorii pini:

- semnal de ceas
- una sau două intrări
- un semnal de ieșire
- complementul ieșirii (opțional)
- semnal de clear
- Vcc si Ground

Tipuri de bistabili:

- SR
- T
- JK
- D

6.1.1 Bistabilul SR (RS Latch)

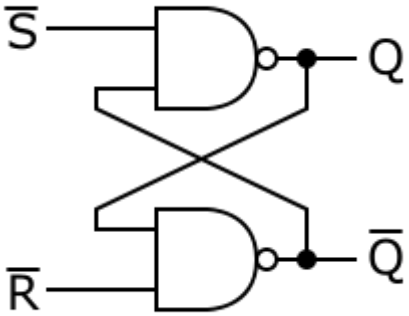


Figura 8. Bistabilul SR folosind porți NAND cu S și R active pe low

Bistabilul SR are două intrări, S - Set folosită pentru a pune Q pe 1, deci a stoca valoarea 1, iar R - Reset folosită pentru a pune Q pe 0, deci a stoca valoarea 0. După cum se observă din figura 8, ieșirile depind de valorile precedente, iar starea se schimbă imediat ce intrările se modifică. Dacă ambele intrări sunt pe low atunci ieșirile rămân neschimbate iar dacă ambele sunt pe high ieșirea este într-o stare nedeterminată.

- S, R active atunci stare nedeterminată
- S, R pe low atunci Q nu se schimbă
- S activ atunci ieșirea este 1
- R activ atunci ieșirea este 0

6.1.2 Bistabilul T

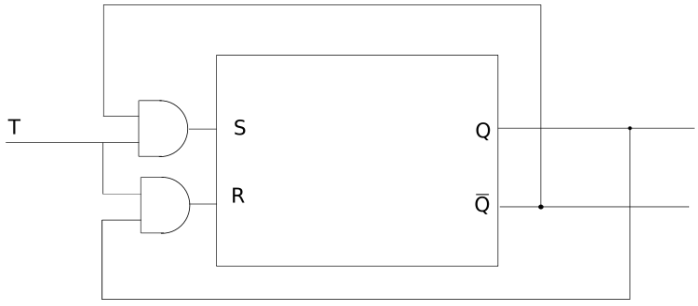


Figura 9. Bistabilul T

Bistabilul T are un singur pin de intrare:

- Când T este activ ieșirea se schimbă la fiecare puls al ceasului.
- Când T nu este activ ieșirea rămâne neschimbată

Tabela de adevăr pentru acest bistabil este următoarea:

T	Q	Q'

0	0	0
0	1	1
1	0	1
1	1	0

6.1.3 Bistabilul JK

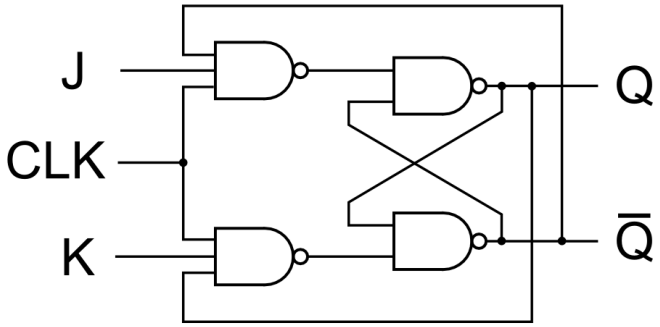


Figura 10. Bistabilul JK

Acest bistabil este similar celui SR, având două intrări J și K, iar ca o diferență față de acesta elimină starea ambiguă pentru ambele intrări pe ‘1’:

- J = K=1 atunci Q se schimbă
- J != K atunci Q = J
- J = K = 0

J	K	Q	Q'
0	0	0	0
0	0	1	1
0	1	X	0
1	0	X	1
1	1	0	1
1	1	1	0

6.1.4 Bistabilul D

Figura 11. Bistabilul D văzut ca un black-box, semnalul de ceas CLK se mai noteaza si E - enable

Ideea acestui bistabil pornește tot de la eliminarea stării ambigue de la bistabilul SR, și reușește acest lucru prin a nu mai permite ca S și R să fie egale. Bistabilul D are o singură intrare (D), conectând cele două intrări SR printr-un invertor. Acest tip de bistabil este cel mai des utilizat, datorită proprității sale de a se comporta ca o celula de memorare fiind folosit pentru realizarea registrelor procesoarelor.
Pentru a înțelege mai ușor comportamentul bistabilelor, pe langă tabele de adevăr mai sunt utile si diagramele de semnale (‘timing diagrams’) cum este cea din figura 12, unde se poate observa cum ieșirea Q se schimbă doar pe frontul crescător de ceas.

CLK	**D**	**Q**
front crescător	0	0
front crescător	1	1
	x	$Q_{anterior}$

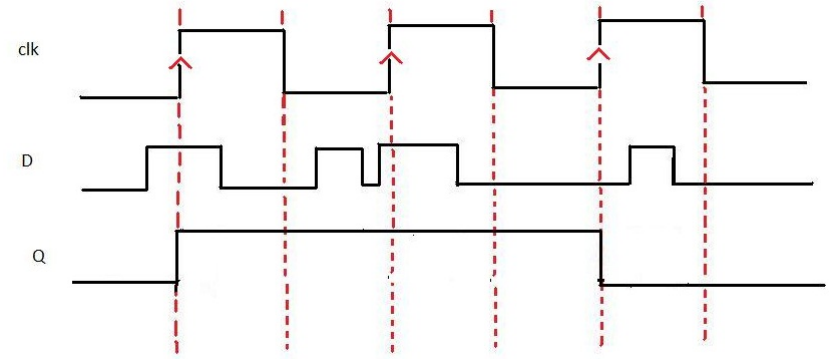


Figura 12. Diagrama de semnale pentru bistabilul D (sursa imagine)

Linkuri utile

- Simulare online circuite http://www.falstad.com/circuit/
- Porți logice, logică secvențială si combinațională http://www.asic-world.com/digital/tutorial.html