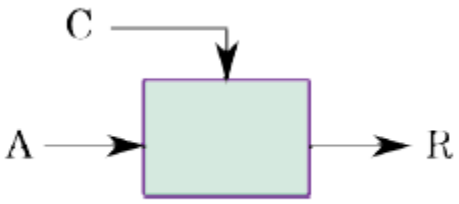
MINIMIZAREA FUNCȚIILOR LOGICE. STRUCTURI LOGICE COMBINAȚIONALE.

1. Obiective

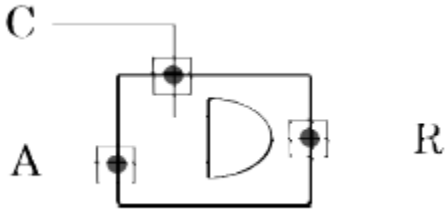
Prin parcurgerea acestei ședințe de laborator studenții vor fi capabili:

- Să implementeze o funcție logică descrisă în limbaj natural;
- Să observe efectul intrării de comandă a unei porți logice;
- Să minimizeze funcții logice folosind metoda Veitch-Karnaugh;
- Să definească principalele structuri logice combinaționale;
- Să explice modul de propagare al unui semnal printr-o schemă cu porți logice.

Dacă până acum am tratat porțile logice ca fiind pura implementare a unor funcții booleene dependente de anumite variabile de intrare și nu am considerat posibilitatea ca variabilele de intrare să poată fi de mai multe tipuri, ne vom concentra în continuare pe această idee. Variabilele booleene pot fi de două tipuri: date sau variabile de control. Variabilele de control vor restricționa fluxul de date din circuitul logic. Mai jos este ilustrat un exemplu simplu, pentru un circuit cu o variabilă de control, o intrare de date și o ieșire:



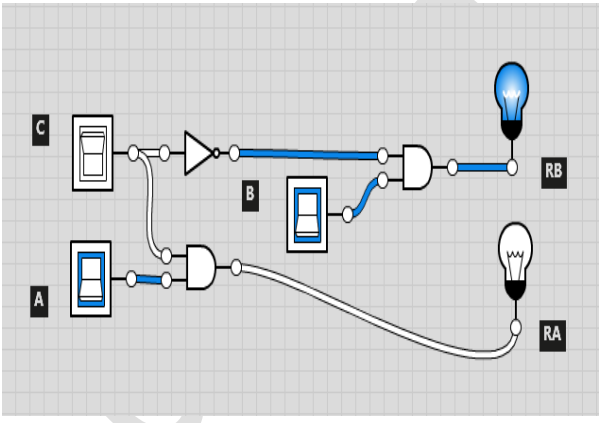
Specificațiile funcționale informale ale circuitului logic sunt: Dacă variabila de control este 1, atunci ieșirea circuitului copie intrarea, altfel ieșirea va fi 0. Cu alte cuvinte, circuitul fie permite propagarea semnalului nemodificat de la intrare până la ieșire, fie îl blochează și totul depinde de variabila de control. Implementarea circuitului este extrem de simplă deoarece acesta constă într-o simplă poartă AND:



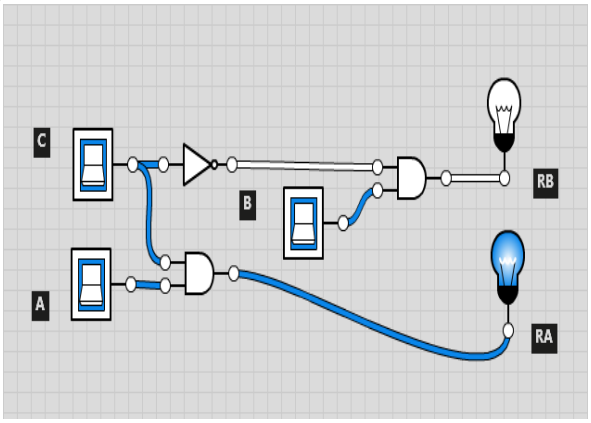
Conceptul poate fi extins pentru proiectarea unui circuit extrem de folositor. Acesta are două intrări de date: A și B, o intrare de control C și o ieșire R. Specificațiile de proiectare solicită următoarele: ”Dacă intrarea de control este 0, atunci ieșirea copie intrarea A, altfel ieșirea copie intrarea B.”. În altă ordine de idei funcționarea este similară unui comutator între cele două linii de date. Pornind de la specificații, construim tabelul de adevăr:

C	A	B	R
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Am observat în cadrul primului exemplu că o poartă logică AND poate fi folosită pentru a bloca propagarea unui semnal și putem merge pe această idee în selecția intrării de date prin inversarea intrării de control astfel încât atunci când C este 0 se va propaga semnalul de la intrarea A, iar semnalul de la intrarea B va fi blocat și invers. Putem implementa circuitul folosind doua porți AND și un invertor:

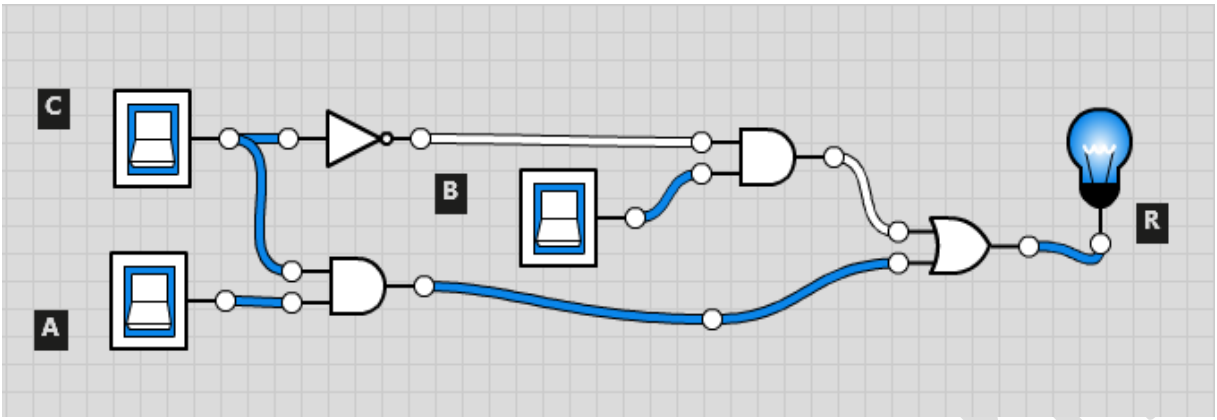


Cazul 1. Dacă A=1, B=1, C=0, atunci RA=0 și RB=1

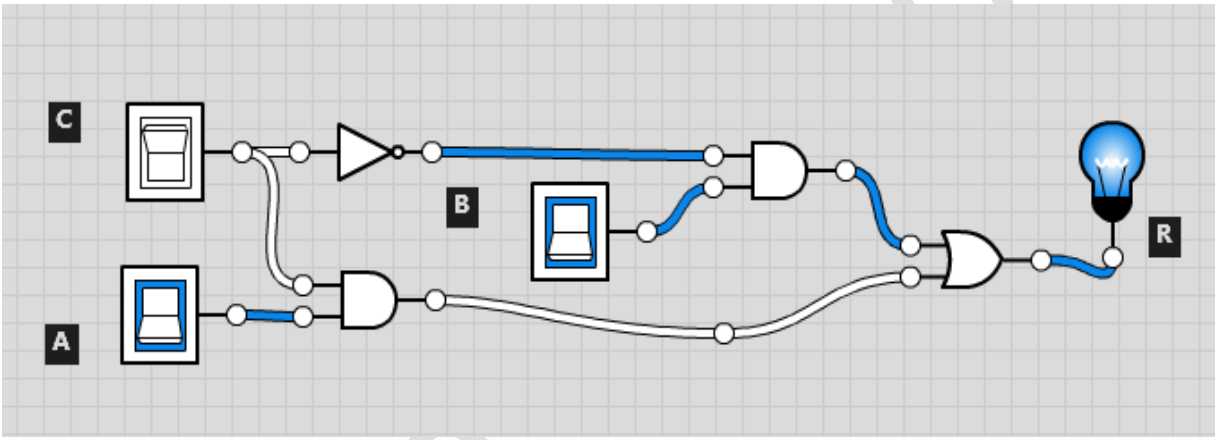


Cazul 2. Dacă A=1, B=1, C=1, atunci RA=1 și RB=0

În acest caz, circuitul obținut va avea două ieșiri R_A și R_B. Pentru a obține un circuit cu o singură ieșire R ne vom folosi de elementul neutru la adunare din algebra booleană $A + 0 = A$ și vom introduce cele două ieșiri într-o poartă OR, astfel:

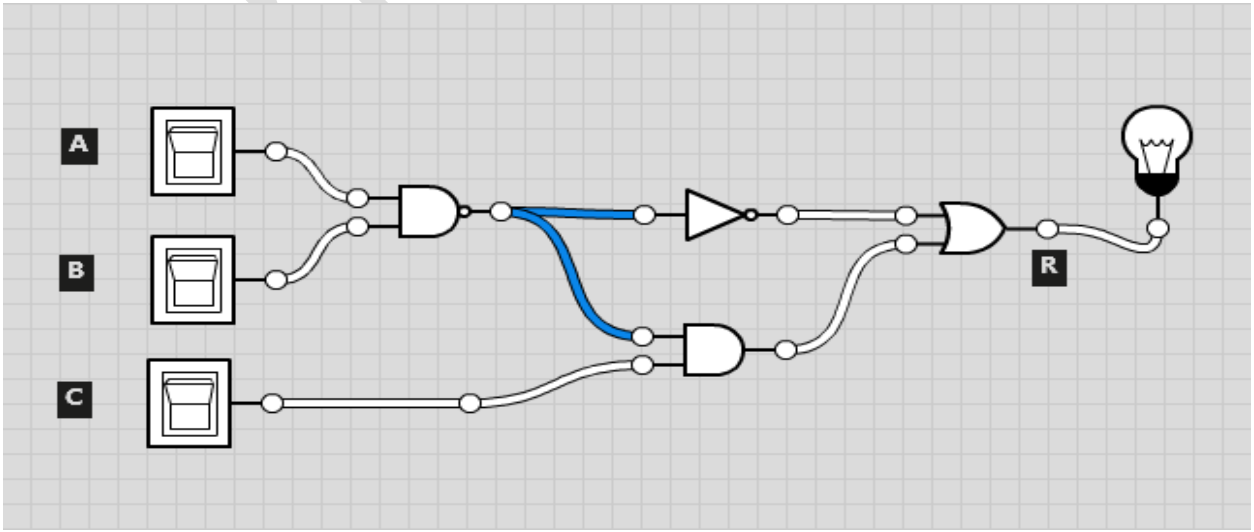


Dacă A=1, B=1, C=1, atunci RA=1 și RB=0.



Dacă A=1, B=1, C=0, atunci RA=0 și RB=1.

Circuitul obținut poartă denumirea de multiplexor și vom vedea mai târziu că este unul dintre cele mai folositoare circuite logice.

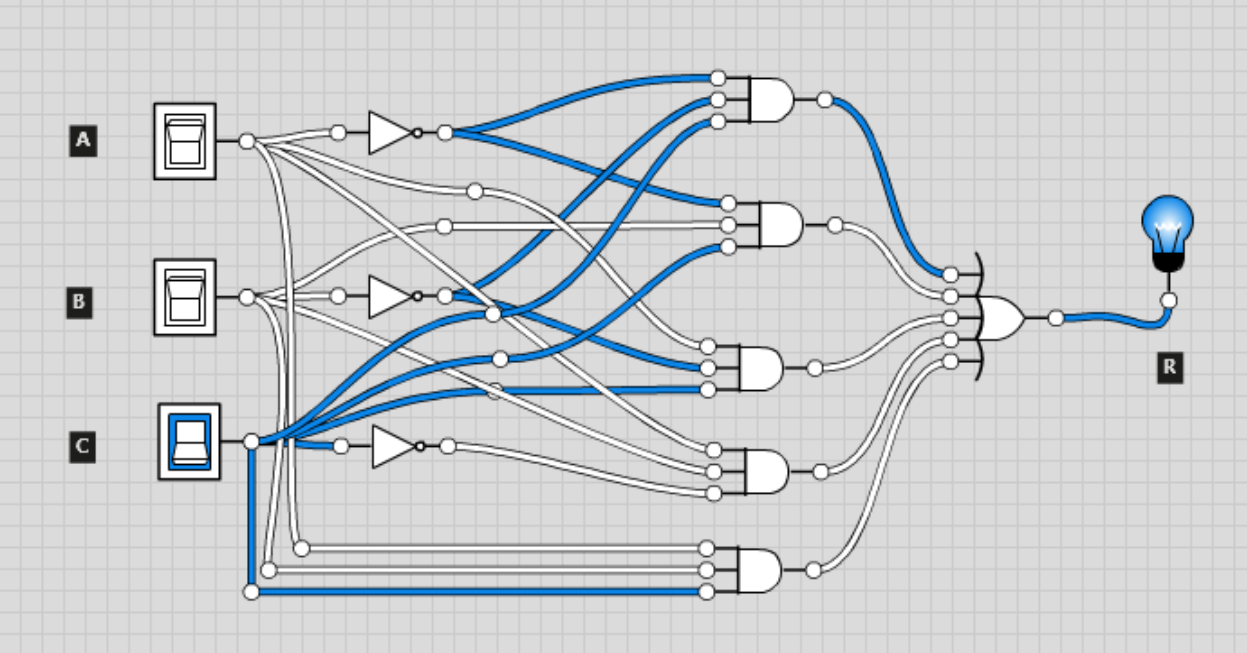


Să considerăm schema logică din figura de mai sus. Funcția reprezentată prin această schemă are tabelul de adevăr:

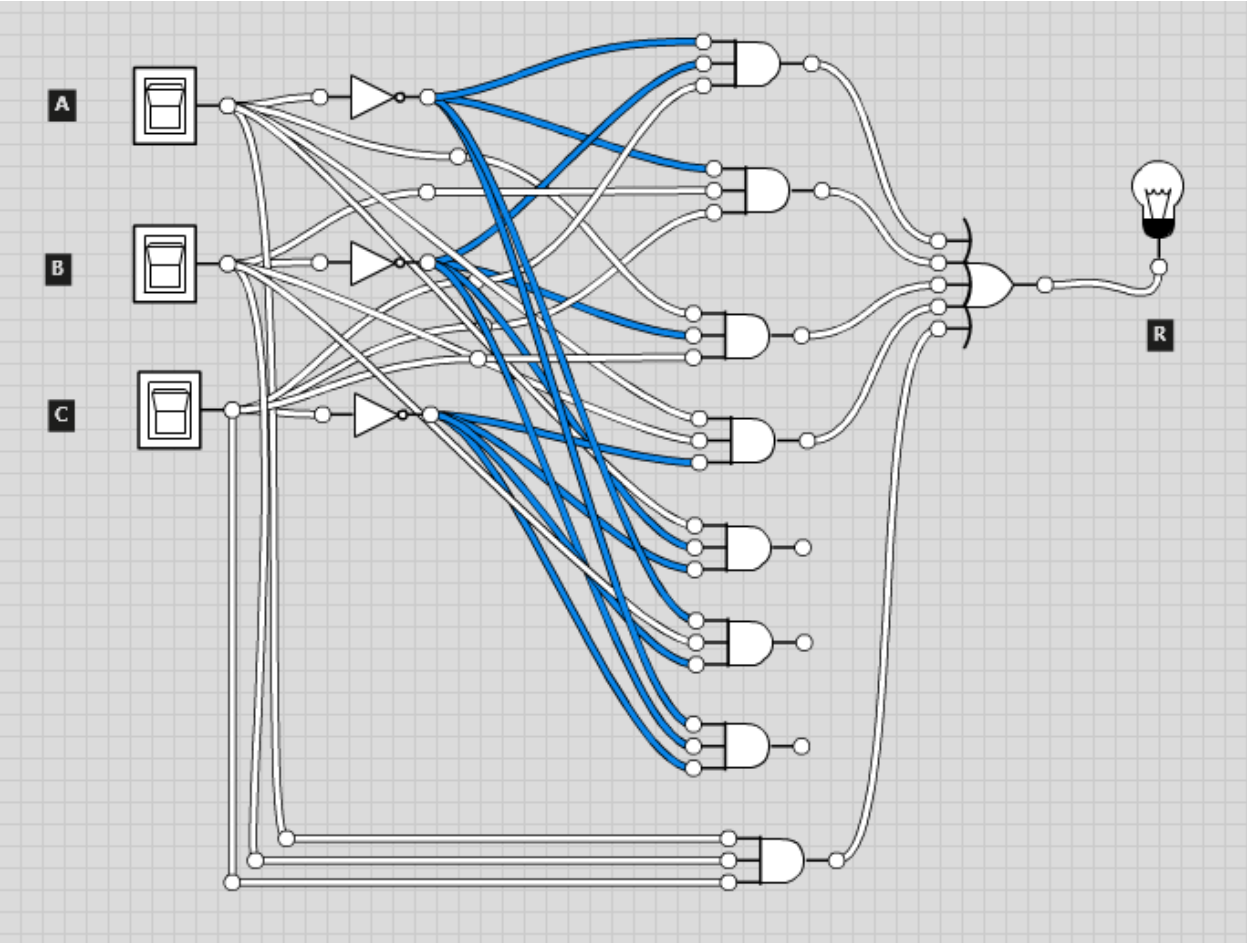
A	B	C	R
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Inspectând tabelul de adevăr și schema logică putem obține ușor formula funcției reprezentate:

$$R = \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$



Astfel, putem construi schema logică echivalentă folosind 5 porți AND cu 3 intrări și o poartă OR cu 5 intrări. Oricum, putem reprezenta schema logică într-o formă generalizată care să conțină toate cele 8 porți AND cu 3 intrări:

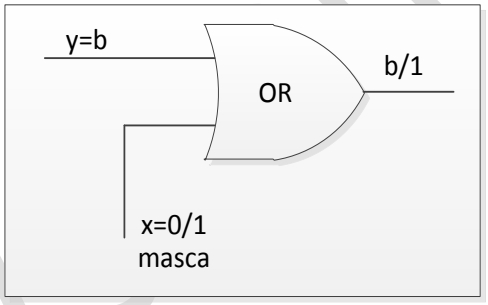


1. Intrări de comandă pentru circuitele logice elementare

1.1. Poarta SAU

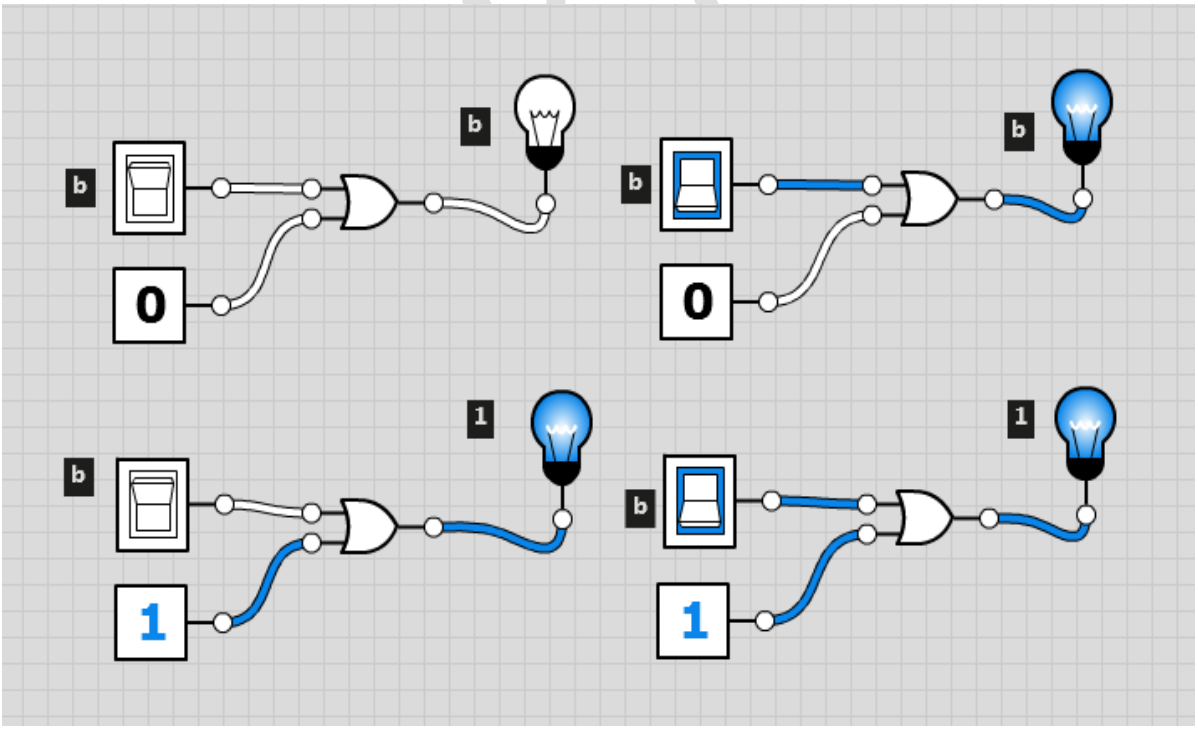
Dacă semnalul de comandă este 1,
Atunci ieșirea este forțată pe valoarea 1
Altfel ieșirea copie intrarea de date.

Notații:
- Intrarea de comandă (masca) x=0/1;
- Intrarea de date b;



Tabelul de adevăr:

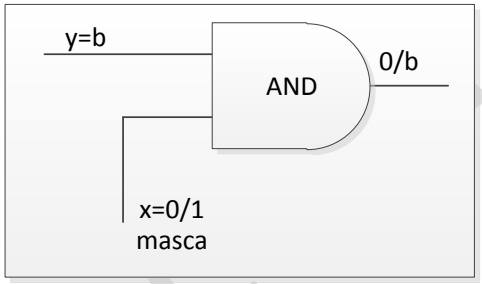
x	*y*	*b*
0		***b***
1		**1**



1.2. Poarta ȘI

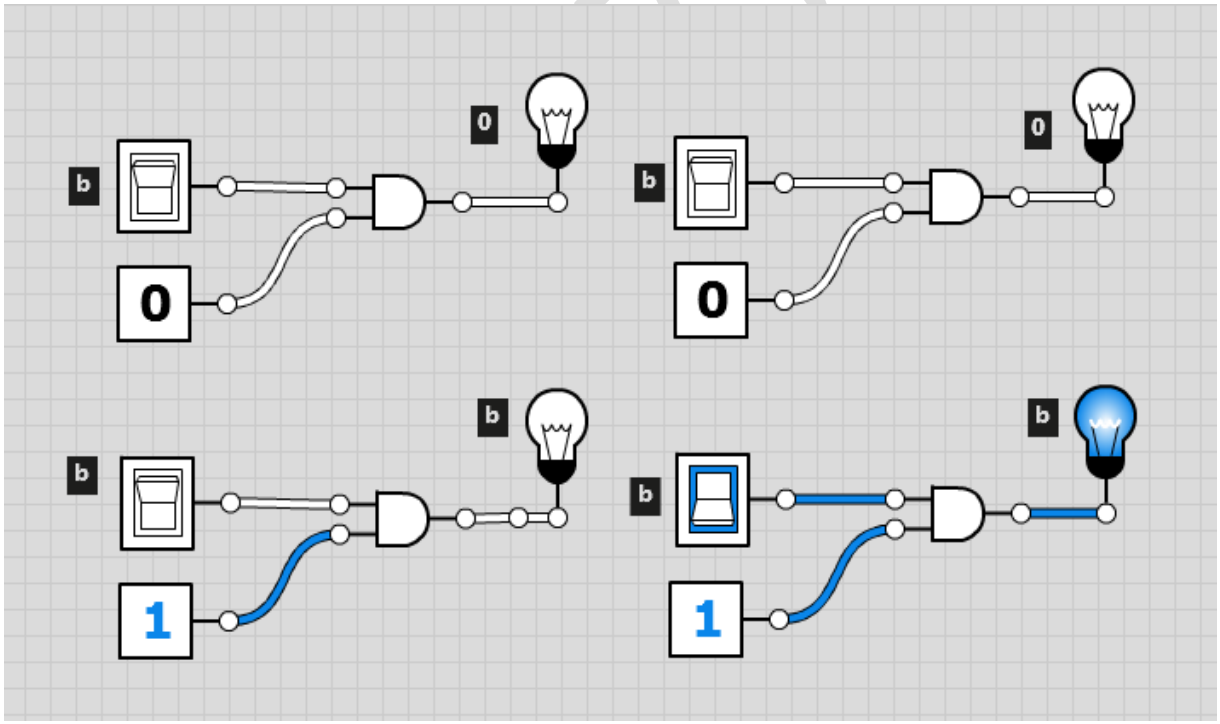
Dacă semnalul de comandă este 1,
Atunci ieșirea copie intrarea de date
Altfel ieșirea este forțată pe valoarea 0.

Notații:
- Intrarea de comandă (masca) x=0/1;
- Intrarea de date b;



Tabelul de adevăr:

x	*y*	*b*
0		**0**
1		***b***

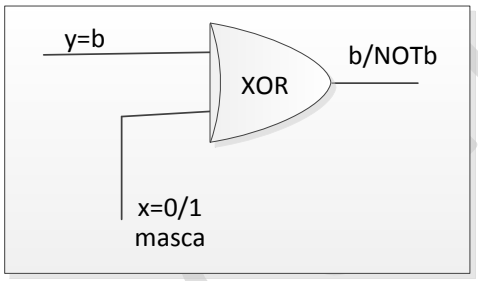


1.3. Poarta XOR

Dacă semnalul de comandă este 0,
Atunci ieșirea copie intrarea de date
Altfel ieșirea copie intrarea de date inversată.

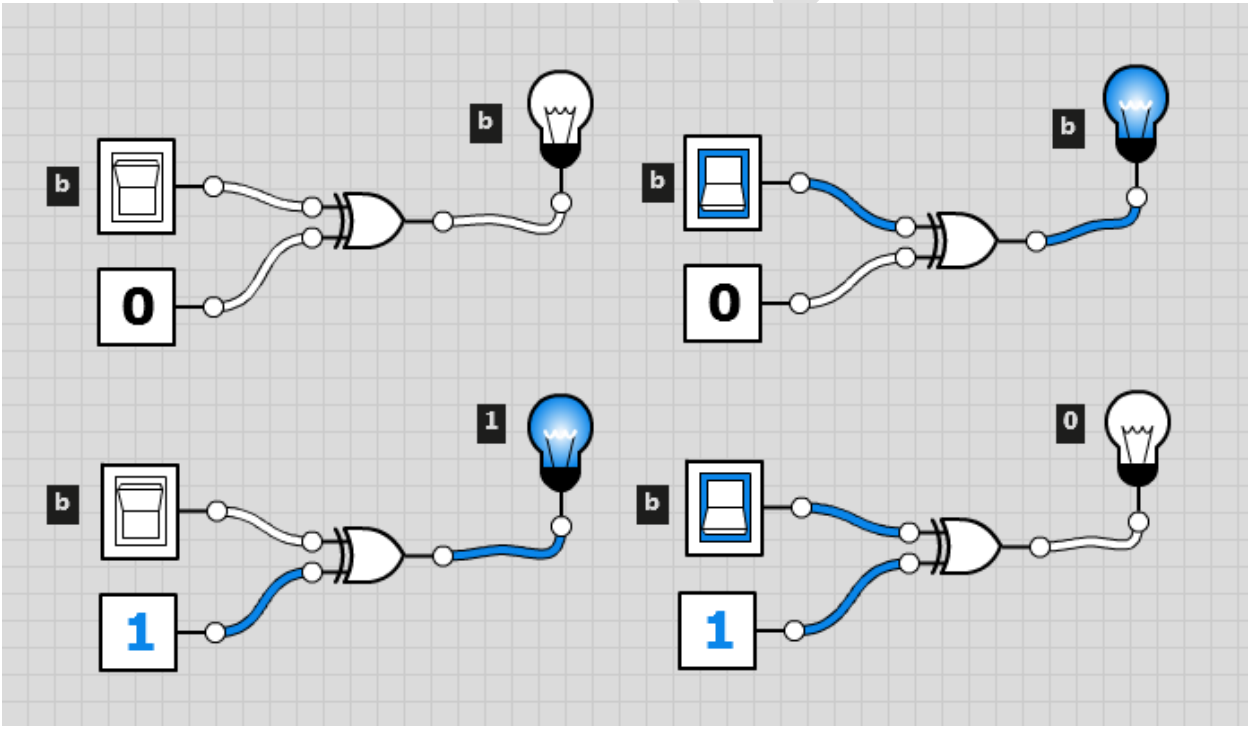
Notații:

- Intrarea de comandă (masca) x=0/1;
- Intrarea de date b;



Tabelul de adevăr:

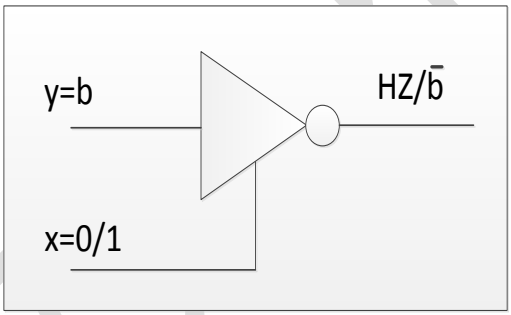
x	y	b
0		b
1		$\overline{b}$



1.4. Invertorul cu trei stări

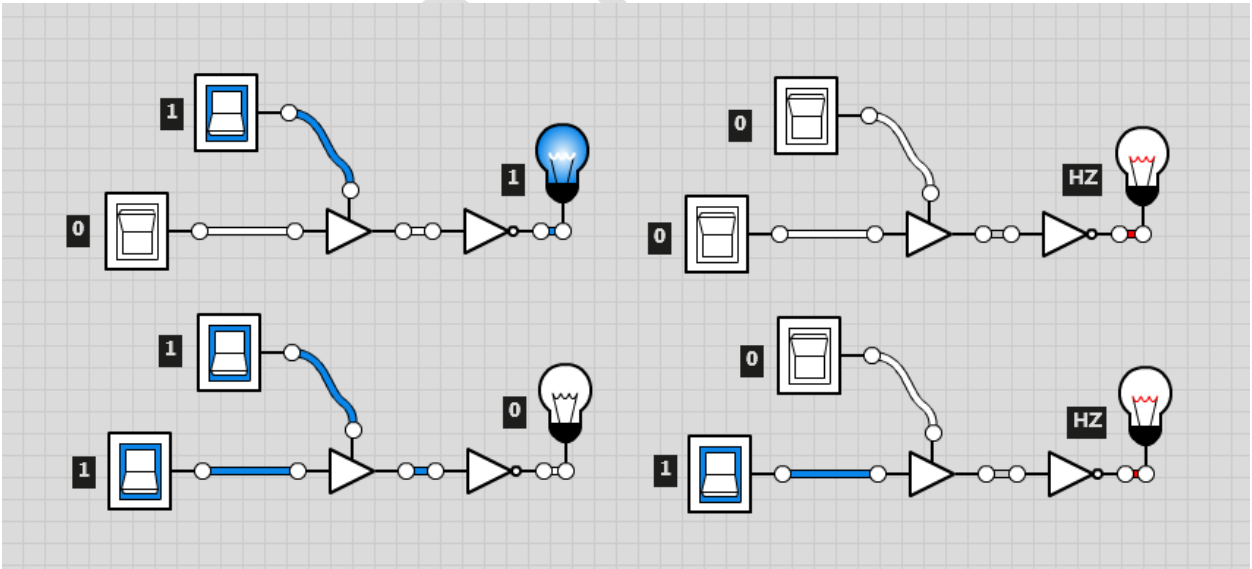
Dacă semnalul de comandă este 0,
Atunci ieșirea copie intrarea de date
Altfel ieșirea copie intrarea de date inversată.
Este folosit pentru a conecta ieșirile mai multor circuite electrice la o magistrală evitând apariția unor scurt-circuite.

Notații:
- Intrarea de comandă (masca) x=0/1;
- Intrarea de date b;
- HZ = deconectat;



Tabelul de adevăr:

x	y	b
0		HZ
1		$\bar{b}$



2. Metoda Veitch-Karnaugh

Minimizarea unei funcții logice presupune obținerea unei expresii cu cost minim pentru un număr de nivele dat.

O funcție logică poate fi minimizată prin:

1) Aplicarea proprietăților și teoremele algebrei logice;
2) Metode sistematice (metoda Veitch-Karnaugh sau metoda Quine McCluskey).

Un sir de n variabile poate fi partiționat în două grupe astfel:

$$x_{n-1}x_{n-2}\dots x_1x_0$$

$$\begin{cases} x_{n-1}\dots x_p \\ x_p\dots x_0 \end{cases}$$

cu $p = \begin{cases} \frac{n}{2} & n\ este\ par \\ \frac{n+1}{2} & n\ este\ impar \end{cases}$.

Se vor parcurge următoarele etape:

1) Se construiește o matrice ce conține un număr de linii și coloane determinat de partiția de variabilă fixată: $\lfloor 2^k, 2^p \rfloor$, unde k reprezintă numărul de elemente din primul termen al partiției, iar p numărul de elemente din al doilea termen al partiției;
2) Pentru ordonarea liniilor și a coloanelor se folosește codul Gray, care are proprietatea de adiacență. Adunând două conjuncții sau înmulțind două disjuncții adiacente dispare termenul care diferă;
3) Se completează matricea cu valori de 1 (conjuncții) sau de 0 (disjuncții) sau $\emptyset$ (indiferent) ale funcției în celulele matricei, corespunzător combinațiilor variabilelor;
4) Se grupează celulele vecine cu același conținut în grupe cât mai mari, formate dintr-un număr de elemente egal cu o putere a lui 2 (1,2,4 sau 8). Celulele laterale care sunt simetrice față de o axă mediană imaginară sunt vecine;
5) Fiecare grupă formează un termen minimizat, termen ce nu conține variabilele ce-și schimbă semnul și astfel dispar.

Observații!

- Un element poate face parte din mai multe grupe;
- Fiecare grupă trebuie să conțină cel puțin un element nou față de restul grupelor;
- Orice element aparține măcar unei grupe, fie aceasta formată chiar dintr-un singur element;
- Variabilele independente se scriu conform regulilor de la formele canonice.

2.1. Codificarea Gray

Am utilizat în lucrările precedente reprezentarea funcțiilor logice în limbaj natural, prin tabel de adevăr, prin realizant, prin schemă logică. Diagrama Veitch-Karnaugh reprezintă o altă formă de reprezentare a unei funcții logice. În cazul unei funcții logice de n variabile, diagrama V-K va fi alcătuită din 2^n căsuțe. Dispunerea căsuțelor în tabela V-K se va face astfel încât grupările vecine să difere printr-un singur bit. Codificarea căsuțelor în tabel atât pe orizontală cât și pe verticală se va face în cod Gray, deoarece acesta are proprietatea că două coduri succesive diferă printr-un singur bit.

Cod binar	Cod Gray
00	00
01	01
10	11
11	10

Codul Gray pe 2 biți

Cod binar	Cod Gray
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

Codul Gray pe 3 biți

Cod binar	Cod Gray
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

Codul Gray pe 4 biți

Numărul de vecini ai unei căsuțe este egal cu numărul variabilelor de intrare. Căsuțele sunt considerate vecine doar pe orizontală și verticală. Nu sunt vecine două căsuțe pe diagonală. Pentru a stabili toate căsuțele vecine, diagrama V-K trebuie privită rotit în jurul axelor verticală și orizontală dar și pliată.

Pentru o funcție definită cu patru variabile de intrare x_3, x_2, x_1, x_0, tabela V-K va avea următoarea **formă** și **conținutul** conform cu valorile funcției plasate pe pozițiile indicate în colțul fiecărei căsuțe.

x_1x_0 / x_3x_2	**00**	**01**	**11**	**10**
00	**0** $f(0000) = f(0)$	**1** $f(0001) = f(1)$	**3** $f(0011) = f(3)$	**2** $f(0010) = f(2)$
01	**4** $f(0100) = f(4)$	**5** $f(0101) = f(5)$	**7** $f(0111) = f(7)$	**6** $f(0110) = f(6)$
11	**12** $f(1100) = f(12)$	**13** $f(1101) = f(13)$	**15** $f(1111) = f(15)$	**14** $f(1110) = f(14)$
10	**8** $f(1000) = f(8)$	**9** $f(1001) = f(9)$	**11** $f(1011) = f(11)$	**10** $f(1010) = f(10)$

Pentru o funcție definită cu trei variabile de intrare x_2, x_1, x_0, tabela V-K va avea următoarea **formă** și **conținutul** conform cu valorile funcției plasate pe pozițiile indicate în colțul fiecărei căsuțe.

x_0 / x_2x_1	**0**	**1**
00	**0** $f(000) = f(0)$	**1** $f(001) = f(1)$
01	**2** $f(100) = f(2)$	**3** $f(011) = f(3)$
11	**6** $f(110) = f(6)$	**7** $f(111) = f(7)$
10	**4** $f(100) = f(4)$	**5** $f(101) = f(5)$

2.2. Transcrierea din tabelul de adevăr în tabela V-K

	x_3x_2	x_3	x_2	x_1	x_0	y
I	00	0	0	0	0	0
		0	0	0	1	0
		0	0	1	0	1
		0	0	1	1	0
II	01	0	1	0	0	1
		0	1	0	1	1
		0	1	1	0	0
		0	1	1	1	1
III	10	1	0	0	0	0
		1	0	0	1	0
		1	0	1	0	1
		1	0	1	1	0
IV	11	1	1	0	0	0
		1	1	0	1	1
		1	1	1	0	1
		1	1	1	1	0

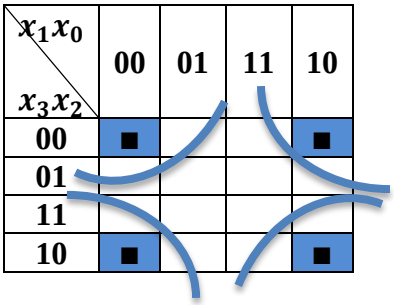
x_3x_2 \ x_1x_0		00	01	11	10
I	00	0	0	0	1
II	01	1	1	1	0
IV	11	0	1	0	1
III	10	0	0	0	1

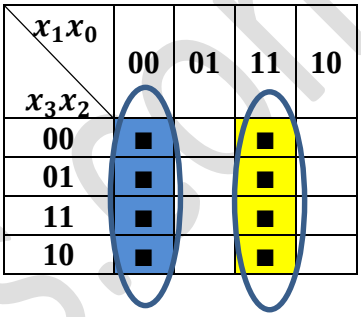
2.3. Modul de formare al grupărilor în tabela V-K și regula de reducere a variabilelor de intrare din tabela V-K

2.3.1. Grupare pe elemente de 1

Tabela conține o singură grupare care conține toate cele patru colțuri în care se regăsește câte un pătrățel negru.

Tabela conține două grupări, fiecare conținând o coloană cu 4 pătrățele negre.





Variabilele de intrare care variază în cadrul grupării sunt haşurate cu roşu şi se reduc. Variabilele de intrare care sunt evaluate cu 1 logic în cadrul grupării nu vor fi negate, iar cele care sunt evaluate cu 0 vor fi negate. Variabilele de intrare vor fi legate prin ŞI logic (.), iar grupările vor fi legate prin SAU logic (+).

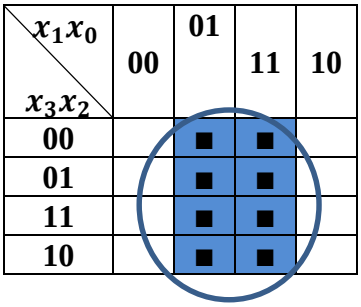
~~x_3~~	x_2	~~x_1~~	x_0
~~0~~	0	~~0~~	0
~~1~~	0	~~1~~	0

~~x_3~~	~~x_2~~	x_1	x_0
~~0~~	~~0~~	0	0
~~0~~	~~1~~		
~~1~~	~~1~~		
~~1~~	~~0~~		

~~x_3~~	~~x_2~~	x_1	x_0
~~0~~	~~0~~	1	1
~~0~~	~~1~~		
~~1~~	~~1~~		
~~1~~	~~0~~		

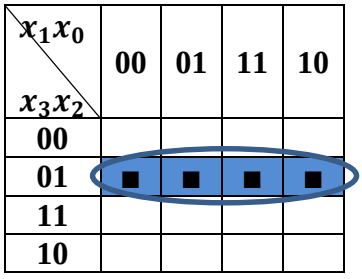
$\overline{x_2} \cdot \overline{x_0}$

$f = \overline{x_2} \cdot \overline{x_0}$



$\overline{x_1} \cdot \overline{x_0}$ $x_1 \cdot x_0$

$f = \overline{x_1} \cdot \overline{x_0} + x_1 \cdot x_0$



x_3	x_2	x_1	x_0
0	0	0	1
0	1	1	1
1	1		
1	0		

$$f = x_0$$

Tabela conține două grupări de câte un element, localizate în cele două colțuri diagonal opuse.

x_3	x_2	x_1	x_0
0	1	0	0
		0	1
		1	1
		1	0

$$f = \overline{x_3} \cdot x_2$$

Tabela conține o grupare formată din cele 4 pătrățele negre.

x_1x_0 / x_3x_2	00	01	11	10
00				■
01				
11				
10	■			

x_1x_0 / x_3x_2	00	01	11	10
00				
01		■	■	
11		■	■	
10				

x_3	x_2	x_1	x_0
1	0	0	0

x_3	x_2	x_1	x_0
0	0	1	0

$$f = x_3 \cdot \overline{x_2} \cdot \overline{x_1} \cdot \overline{x_0} + \overline{x_3} \cdot \overline{x_2} \cdot x_1 \cdot \overline{x_0}$$

Tabela conține o grupare formată din cele 4 pătrățele negre localizate pe verticală pe marginile tabelei.

x_3	x_2	x_1	x_0
0	1	0	1
1	1	1	1

$$f = x_2x_0$$

Tabela conține o grupare formată din cele 4 pătrățele negre localizate pe orizontală pe marginile tabelei.

x_1x_0 / x_3x_2	00	01	11	10
00				
01	■			■
11	■			■
10				

x_1x_0 / x_3x_2	00	01	11	10
00		■	■	
01				
11				
10		■	■	

~~x_3~~	x_2	~~x_1~~	x_0
~~0~~	1	~~0~~	0
~~1~~	1	~~1~~	0

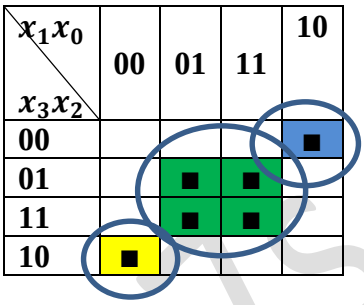
$f = x_2 \cdot \overline{x_0}$

~~x_3~~	x_2	~~x_1~~	x_0
~~0~~	0	~~0~~	1
~~1~~	0	~~1~~	1

$f = \overline{x_2} \cdot x_0$

2.3.2. Grupare pe elemente de 0

Tabela conține 3 grupări: două dintre ele au în componență un singur element marcat printr-un pătrățel negru, iar cea de-a treia are patru elemente.



Variabilele de intrare care variază în cadrul grupării sunt marcate cu roșu și se reduc. Variabilele de intrare care sunt evaluate cu 1 logic în cadrul grupării vor fi negate, iar cele care sunt evaluate cu 0 nu vor fi negate. Variabilele de intrare vor fi legate prin SAU logic (+), iar grupările vor fi legate prin ȘI logic (.).

x_3	x_2	x_1	x_0
1	0	0	0

~~x_3~~	x_2	~~x_1~~	x_0
~~0~~	1	~~0~~	1
~~1~~	1	~~1~~	1

x_3	x_2	x_1	x_0
0	0	1	0

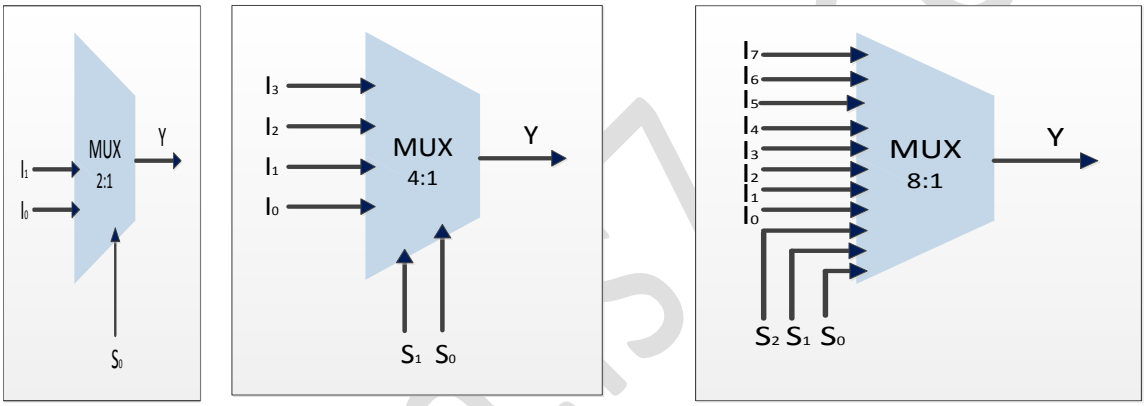
$$f = (\overline{x_3} + x_2 + x_1 + x_0) \cdot (\overline{x_2} + x_0) \cdot (x_3 + x_2 + \overline{x_1} + x_0)$$

3. Structuri logice combinaționale

Interconectarea circuitelor digitale este realizată prin intermediul funcțiilor complementare de demultiplexare și de multiplexare.

3.1. Multiplexorul

Multiplexorul este o structură logică combinațională care selectează în funcție de codul binar primit pe cele n intrări de selecție, una dintre cele 2^n intrări de date și permite propagarea semnalului de la intrarea respectivă până la unica ieșire. Multiplexorul implementează în hardware funcția de decizie, similară cu ”daca…atunci…altfel” pentru $n = 2$ sau ”swicth…case…” pentru $n > 2$.



S_0	I_1	I_0	Y
0	-	0 1	0 1
1	0 1	-	0 1

S_1	S_0	I_3	I_2	I_1	I_0	Y
0	0	-	-	-	0 1	0 1
0	1	-	-	0 1	-	0 1
1	0	-	0 1	-	-	0 1
1	1	0 1	-	-	-	0 1

S_2	S_1	S_0	I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y
0	0	0	-	-	-	-	-	-	-	0 1	0 1
0	0	1	-	-	-	-	-	-	0 1	-	0 1
0	1	0	-	-	-	-	-	0 1	-	-	0 1
0	1	1	-	-	-	-	0 1	-	-	-	0 1
1	0	0	-	-	-	0 1	-	-	-	-	0 1
1	0	1	-	-	0 1	-	-	-	-	-	0 1
1	1	0	-	0 1	-	-	-	-	-	-	0 1
1	1	1	0 1	-	-	-	-	-	-	-	0 1

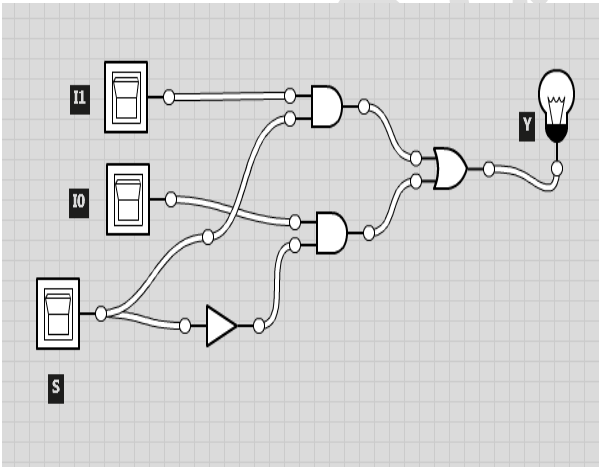
Ecuațiile funcțiilor logice implementate de MUX sunt:

MUX 2:1 $Y = S_0 I_1 + \overline{S_0} I_0;$
MUX 4:1 $Y = S_1 S_0 I_3 + S_1 \overline{S_0} I_2 + \overline{S_1} S_0 I_1 + \overline{S_1}\,\overline{S_0} I_0;$
MUX 8:1 $Y = S_2 S_1 S_0 I_7 + S_2 S_1 \overline{S_0} I_6 + S_2 \overline{S_1} S_0 I_5 + S_2 \overline{S_1}\,\overline{S_0} I_4 + \overline{S_2} S_1 S_0 I_3 + \overline{S_2} S_1 \overline{S_0} I_2 + \overline{S_2}\,\overline{S_1} S_0 I_1 + \overline{S_2}\,\overline{S_1}\,\overline{S_0} I_0;$

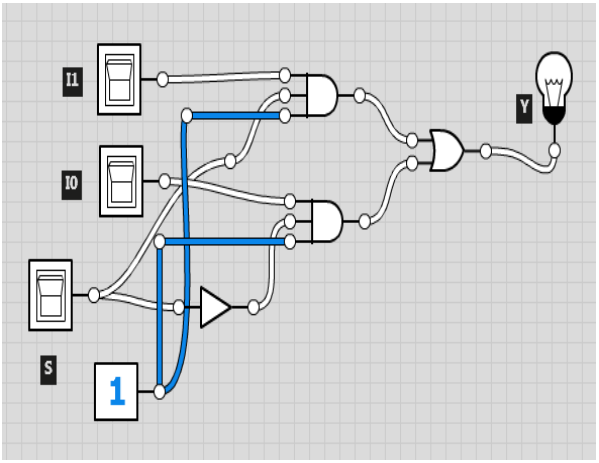
Circuitele MUX pot avea o intrare de validare. Funcționarea este neschimbată dacă intrarea de validare este activată. Dacă intrarea de validare este inactivă ieșirea multiplexorului este egală cu 0 indiferent de valorile intrărilor de date (multiplexorul nu transmite la ieșire nicio valoare primită la intrare). În acest caz, ecuațiile se modifică astfel:

MUX 2:1 $Y = E(S_0 I_1 + \overline{S_0} I_0);$
MUX 4:1 $Y = E(S_1 S_0 I_3 + S_1 \overline{S_0} I_2 + \overline{S_1} S_0 I_1 + \overline{S_1}\,\overline{S_0} I_0);$
MUX 8:1 $Y = E(S_2 S_1 S_0 I_7 + S_2 S_1 \overline{S_0} I_6 + S_2 \overline{S_1} S_0 I_5 + S_2 \overline{S_1}\,\overline{S_0} I_4 + \overline{S_2} S_1 S_0 I_3 + \overline{S_2} S_1 \overline{S_0} I_2 + \overline{S_2}\,\overline{S_1} S_0 I_1 + \overline{S_2}\,\overline{S_1}\,\overline{S_0} I_0);$

Structuri logice care implementează multiplexoare:



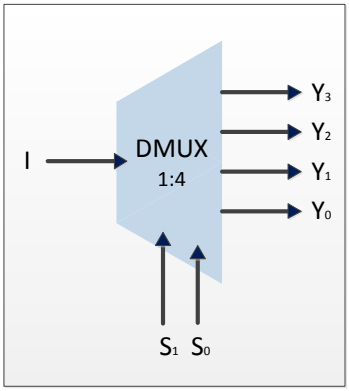
MUX 2:1



MUX 2:1 cu validare

3.2. Demultiplexorul

Demultiplexorul este o structură logică combinațională care selectează în funcție de codul binar primit pe cele n intrări de selecție una dintre cele 2^n ieșiri prin care va permite propagarea semnalului de la unica intrare de date. Circuitele DMUX $1:2^n$ se comportă similar cu decodificatoarele pe n biți cu intrare de validare. Demultiplexorul implementează o structură de tipul ”dacă selecția este 0, atunci intrarea este conectată la ieșirea Y_0; dacă selecția este 1, atunci intrarea este conectată la ieșirea Y_1” în cazul $n = 2$.

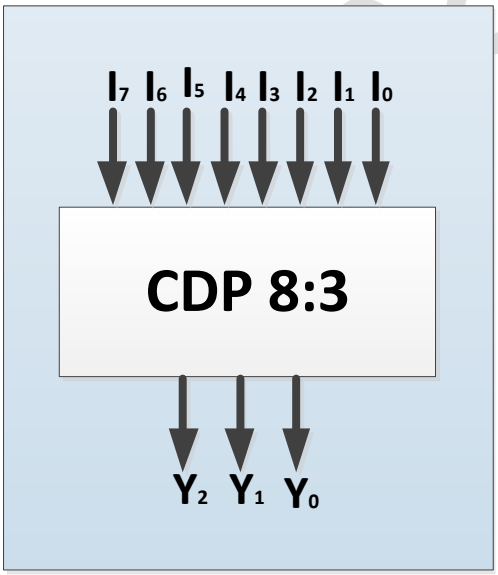


S_1	S_0	I	Y_3	Y_2	Y_1	Y_0
0	0	0 1	-	-	-	0 1
0	1	0 1	-	-	0 1	-
1	0	0 1	-	0 1	-	-
1	1	0 1	0 1	-	-	-

$$Y_3 = S_1 \cdot S_0 \cdot I$$
$$Y_2 = S_1 \cdot \overline{S_0} \cdot I$$
$$Y_1 = \overline{S_1} \cdot S_0 \cdot I$$
$$Y_0 = \overline{S_1} \cdot \overline{S_0} \cdot I$$

3.3. Codificatorul prioritar

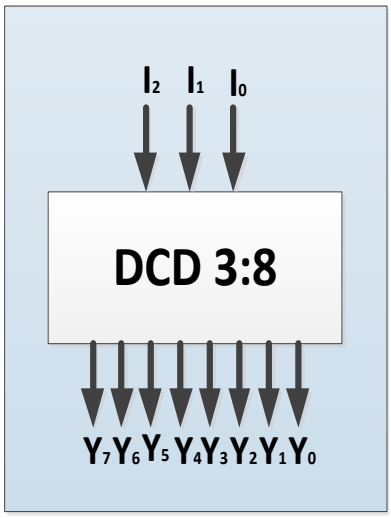
Codificatorul prioritar de 2^n biți este un circuit logic combinațional cu 2^n intrări și n ieșiri. Ieșirea copie codul binar al celei mai prioritare intrări active.



I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	Y_2	Y_1	Y_0
1	-	-	-	-	-	-	-	1	1	1
0	1	-	-	-	-	-	-	1	1	0
0	0	1	-	-	-	-	-	1	0	1
0	0	0	1	-	-	-	-	1	0	0
0	0	0	0	1	-	-	-	0	1	1
0	0	0	0	0	1	-	-	0	1	0
0	0	0	0	0	0	1	-	0	0	1
0	0	0	0	0	0	0	1	0	0	0

3.4. Decodificatorul

Decodificatorul este o structură logică combinațională cu n intrări de selecție și 2^n ieșiri. Va fi setată pe 1 logic numai ieșirea al cărei identificator binar este determinat de către cele n intrări. Restul ieșirilor vor avea valoarea 0 logic.

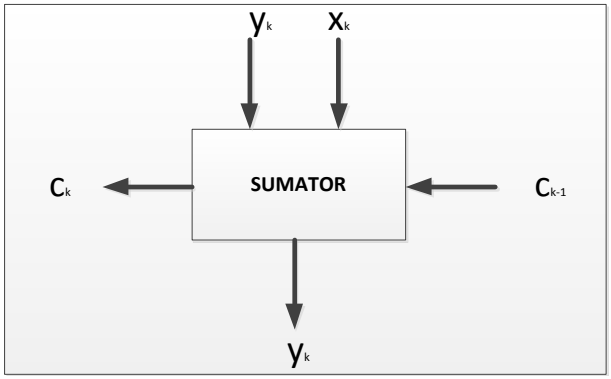


I_2	I_1	I_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

3.5. Sumatorul

Sumatorul este un circuit logic combinațional cu mai multe ieșiri care adună două intrări se date cu una de transport și generează o ieșire de tip sumă și una de tip transport.

Se dau două numere binare $X = x_{n-1} \dots x_0$ și $Y = y_{n-1} \dots y_0$. Sumarea celor două numere se face simultan în toate rangurile. Pentru rangul $k, k = \overline{0, n-1}$ va fi necesar un circuit combinațional similar cu cel din figura de mai jos:



Sumatorul pentru rangul k

unde s-a notat cu C_{k-1} transportul din rangul k-1, iar C_k transportul din rangul k. Tabelul de adevăr al sumatorului pe un rang k este prezentat mai jos

x_k	y_k	C_{k-1}	S_k	C_k
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**

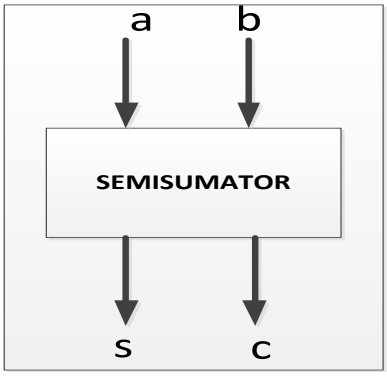
C_{k-1} \ $x_k y_k$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$x_k y_k$ \ $C_{k-1} C_k$	00	01	11	10
00	0	∅	∅	1
01	1	∅	0	∅
11	∅	0	1	∅
10	1	0	0	∅

$$C_k = x_k \cdot y_k + c_{k-1} \cdot (x_k + y_k) \qquad S_k = x_k \cdot y_k \cdot C_{k-1} + \overline{C_k} \cdot (x_k + y_k + C_{k-1})$$

3.6. Semisumatorul

Fie circuitul cu următoarea schemă bloc:

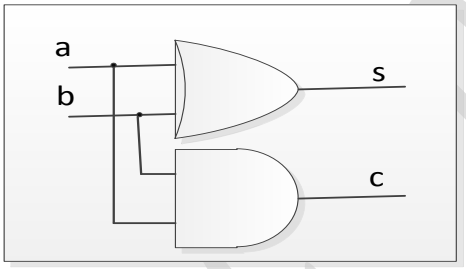


și următorul tabel de adevăr:

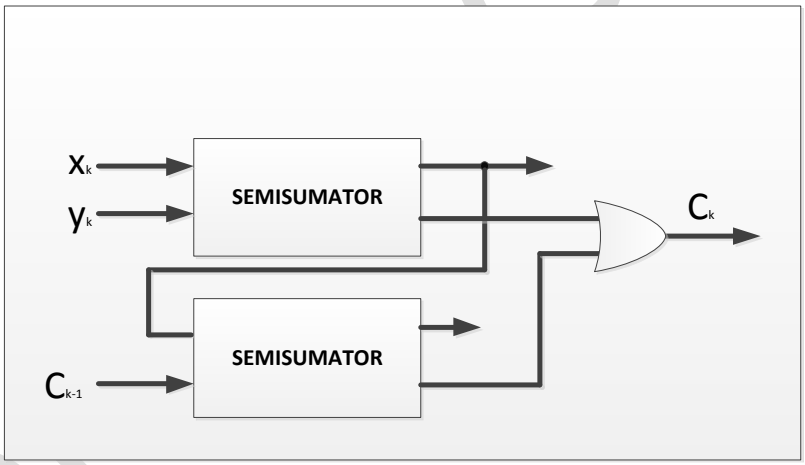
a	b	s	c
0	0	**0**	**0**
0	1	**1**	**0**
1	0	**1**	**0**
1	1	**0**	**1**

Sintetizând semisumatorul rezultă

$s = a \oplus b$
$c = a \cdot b$



Implementarea unui sumator complet folosind semisumatoare este redată în figura:



$S_k = x_k \oplus y_k \oplus C_{k-1}$

$C_k = x_k \cdot y_k + C_{k-1} \cdot (x_k + y_k)$

4. Aplicații propuse:

4.1.Se dau numerele:

$A = a_7a_6a_5a_4a_3a_2a_1a_0$ $B = b_7b_6b_5b_4b_3b_2b_1b_0$

Să se calculeze C cu ajutorul operatorilor logici și a măștilor.

a) $C = \overline{a_7}\,\overline{a_6}\,\overline{a_5}010b_3b_2$
b) $C = \overline{b_5}\,\overline{b_4}\,a_3a_2b_1b_201$
c) $C = 10b_1b_0a_7a_6a_5a_4$
d) $C = 1111b_7b_6\overline{b_5}\,\overline{b_4}$
e) $C = 00b_7\overline{b_6}\,\overline{a_5}\,a_411$

a)	
b)	
c)	

d)	
e)	

4.2.Se dau tabelele Veitch-Karnaugh. Să se evidențieze grupările pe 0 și pe 1 în tabele:

a)

x_1x_0 / x_3x_2	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

b)

x_1x_0 / x_3x_2	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

c)

x_1x_0 / x_3x_2	00	01	11	10
00	0	1	1	0
01	0	0	0	0
11	0	1	1	0
10	0	1	1	0

d)

x_1x_0 / x_3x_2	00	01	11	10
00	0	0	0	0
01	1	0	0	1
11	0	0	0	0
10	1	0	0	1

e)

x_0 / x_2x_1	0	1
00	1	1
01	0	1
11	0	1
10	1	1

f)

x_1x_0 / x_3x_2	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

g)

x_1x_0 / x_3x_2	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	0	0	0
10	1	0	0	1

h)

x_1x_0 / x_3x_2	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

i)

x_0 / x_2x_1	0	1
00	0	0
01	0	1
11	0	1
10	0	0

j)

x_0 / x_2x_1	0	1
00	1	0
01	0	1
11	1	0
10	0	1

4.3.Să se minimizeze funcțiile definite prin tabelele de la problema 4.2.

a)

b)
c)
d)
e)

f)
g)
h)

i)
j)

4.4. Să se minimizeze funcțiile logice folosind metoda Veitch-Karnaugh:

a) $f(x_3, x_2, x_1, x_0) = R_1(1,2,6,8,10,11,15)$;
b) $f(x_2, x_1, x_0) = R_1(1,3,4)$;
c) $f(x_3, x_2, x_1, x_0) = R_1(4,5,8,9,10,12,13)$;
d) $f(x_3, x_2, x_1, x_0) = R_1(0,1,2,4) + R_\emptyset(3,5,10)$;
e) $f(x_3, x_2, x_1, x_0) = R_0(1,3,4,6,9,11,12,14)$;
f) $f(x_3, x_2, x_1, x_0) = R_0(2,3,4,5,8,9,14,15)$;
g) $f(x_3, x_2, x_1, x_0) = R_1(2,3,6,7,8,9,12,13)$;
h) $f(x_3, x_2, x_1, x_0) = R_0(0,1,4,5,10,11,14,15)$;
i) $f(x_3, x_2, x_1, x_0) = R_1(0,4,8,12)$;
j) $f(x_3, x_2, x_1, x_0) = R_1(3,7,11,15)$;
k) $f(x_3, x_2, x_1, x_0) = R_0(4,5,6,7,8,9,10,11)$;
l) $f(x_3, x_2, x_1, x_0) = R_1(0,2,7,9) + R_\emptyset(3,8,10)$;
m) $f(x_3, x_2, x_1, x_0) = R_1(1,4,11,14) + R_\emptyset(3,6,9,12)$;
n) $f(x_3, x_2, x_1, x_0) = R_1(2,4,7,8,13,14) + R_\emptyset(0,5,6,10,12,15)$.
o) $f(x_2, x_1, x_0) = R_1(1,3,4)$;
p) $f(x_2, x_1, x_0) = R_0(0,1,2,3)$.

a) $f(x_3, x_2, x_1, x_0) = R_1(1,2,6,8,10,11,15)$;

x_3x_2 \ x_1x_0	00	01	11	10
00	0	1	0	1
01	0	0	0	1
11	0	0	1	0
10	1	0	1	1

Se obțin grupările
$G_1 = \overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1}x_0$;
$G_2 = \overline{x_3}x_1\overline{x_0}$;
$G_3 = x_3x_1x_0$;
$G_4 = x_3\overline{x_2} \cdot \overline{x_0}$.

rezultând că $f = G_1 + G_2 + G_3 + G_4 = \overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1}x_0 + \overline{x_3}x_1\overline{x_0} + x_3x_1x_0 + x_3\overline{x_2} \cdot \overline{x_0}$.

b) $f(x_2, x_1, x_0) = R_1(1,3,4)$;

c) $(x_3, x_2, x_1, x_0) = R_1(4,5,8,9,10,12,13);$

d) $f(x_3, x_2, x_1, x_0) = R_1(0,1,2,4) + R_{\emptyset}(3,5,10);$

e) $f(x_3, x_2, x_1, x_0) = R_0(1,3,4,6,9,11,12,14)$;

f) $f(x_3, x_2, x_1, x_0) = R_0(2,3,4,5,8,9,14,15)$;

g) $f(x_3, x_2, x_1, x_0) = R_1(2,3,6,7,8,9,12,13)$;

h) $f(x_3, x_2, x_1, x_0) = R_0(0,1,4,5,10,11,14,15)$;

i) $f(x_3, x_2, x_1, x_0) = R_1(0,4,8,12)$;

j) $f(x_3, x_2, x_1, x_0) = R_1(3,7,11,15)$;

k) $f(x_3, x_2, x_1, x_0) = R_0(4,5,6,7,8,9,10,11)$;

l) $f(x_3, x_2, x_1, x_0) = R_1(0,2,7,9) + R_\emptyset(3,8,10)$;

m) $f(x_3, x_2, x_1, x_0) = R_1(1,4,11,14) + R_\emptyset(3,6,9,12)$;

n) $f(x_3, x_2, x_1, x_0) = R_1(2,4,7,8,13,14) + R_\emptyset(0,5,6,10,12,15)$;

o) $f(x_2, x_1, x_0) = R_1(1,3,4)$;

p) $f(x_2, x_1, x_0) = R_0(0,1,2,3)$.

4.5.Să se implementeze cu porți logice formulele obținute după minimizarea funcțiilor de la problema 4.4.

$a) f(x_3, x_2, x_1, x_0) = R_1(1,2,6,8,10,11,15)$;

b) $f(x_2, x_1, x_0) = R_1(1,3,4)$;

c) $(x_3, x_2, x_1, x_0) = R_1(4,5,8,9,10,12,13)$;

d) $f(x_3, x_2, x_1, x_0) = R_1(0,1,2,4) + R_\emptyset(3,5,10)$;

e) $f(x_3, x_2, x_1, x_0) = R_0(1,3,4,6,9,11,12,14)$;

f) $f(x_3, x_2, x_1, x_0) = R_0(2,3,4,5,8,9,14,15)$;

g) $f(x_3, x_2, x_1, x_0) = R_1(2,3,6,7,8,9,12,13)$;

h) $f(x_3, x_2, x_1, x_0) = R_0(0,1,4,5,10,11,14,15)$;

i) $f(x_3, x_2, x_1, x_0) = R_1(0,4,8,12)$;

j) $f(x_3, x_2, x_1, x_0) = R_1(3,7,11,15)$;

k) $f(x_3, x_2, x_1, x_0) = R_0(4,5,6,7,8,9,10,11)$;

l) $f(x_3, x_2, x_1, x_0) = R_1(0,2,7,9) + R_\emptyset(3,8,10)$;

m) $f(x_3, x_2, x_1, x_0) = R_1(1,4,11,14) + R_\emptyset(3,6,9,12)$;

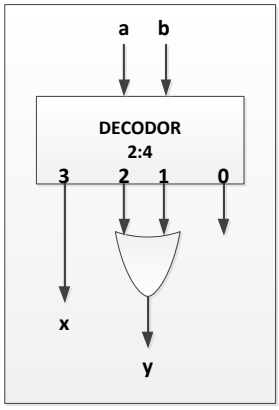
n) $f(x_3, x_2, x_1, x_0) = R_1(2,4,7,8,13,14) + R_\emptyset(0,5,6,10,12,15)$;

o) $f(x_2, x_1, x_0) = R_1(1,3,4)$;

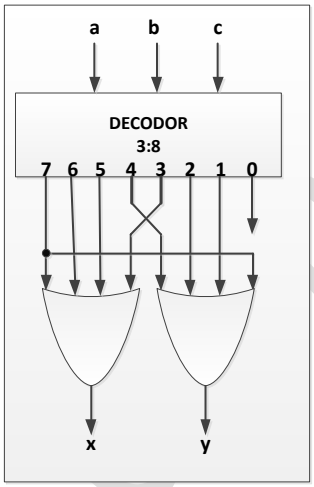
$p)$ $f(x_2, x_1, x_0) = R_0(0,1,2,3)$.

4.6. Să se analizeze funcționarea schemei următoare și să se găsească blocul logic combinațional standard cu același rol:

a)



b)

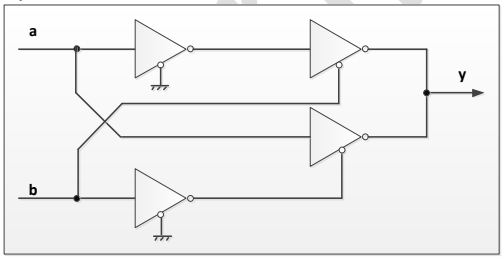


a)

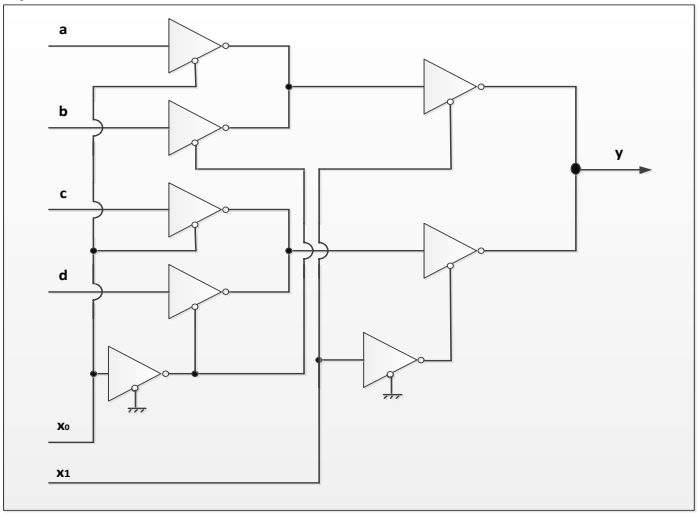
b)

4.7.Să se explice de ce schema următoare nu produce scurt-circuit pe ieșiri, apoi să se determine funcția logică implementată și să se găsească blocul logic combinațional echivalent ca funcționare:

a)



b)

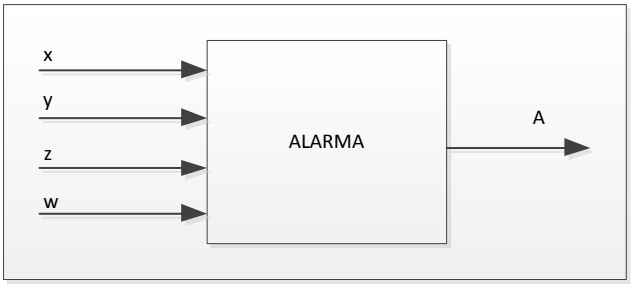


a)

b)

4.8. Să se proiecteze folosind porți logice un sistem de alarmă. Se consideră că alarma va fi activată în unul din cazurile:

1) Este acționat un buton de validare și ușa este deschisă; sau
2) Este acționat butonul de validare, ușa este închisă, fereastra este deschisă și este trecut de ora 22:00.



Indicații

Sunt necesare patru variabile de intrare. Se notează variabilele de intrare în felul următor: x – butonul de validare; y – ușa; z – fereastra și w – dacă este trecut de ora 22:00.

5. Referințe bibliografice

[1] Manta V., Ungureanu F., *Introducere în știința sistemelor și a calculatoarelor,* Volumul I, Editura ”Gh.Asachi”, Iași, 2002.