Lab. 2.
Proiectarea circuitelor logice combinationale.

D1. Circuite logice combinationale (CLC) sunt circuite la care în fiecare moment de timp starea logică a ieşirii depinde de modul în care se combină nivelurile logice ale intrărilor în acel moment de timp. Ele nu au capacitatea de memorare a informaţiei. Ieşirile circuitelor logice combinaţionale se determină direct de funcţia logică aplicată la starea intrărilor (”0” sau ”1” logic la orice moment în timp).

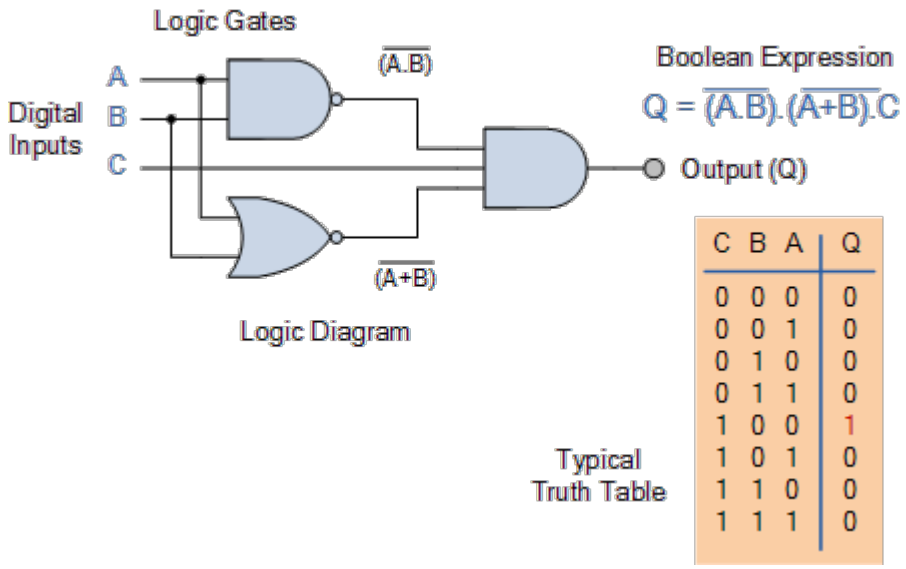
Rezultatul este că circuitele logice combinaționale nu au feedback, iar orice schimbare a semnalelor care se aplică la intrarile lor va avea imediat un efect la ieșire. Cu alte cuvinte, într-un circuit logic combinaţional, ieșirea este dependentă în orice moment de timp de combinația intrărilor sale. Deci, dacă starea intrarilor se schimbă de la 0 la 1 sau 1 la 0, immediat se schimba ieşirile, din cauza că circuitele logice combinaționale implicite, nu are "memorie", "sincronizare" sau "bucle de feedback" în cadrul designului lor.

Circuitele logice combinate sunt alcătuite din porți logice de bază NAND, NOR sau NOT care sunt "combinate" sau conectate împreună pentru a produce circuite de comutare mai complicate. Aceste porți logice sunt elementele de bază ale circuitelor logice combinaționale. Un exemplu de circuit combinaţional este un convertor binar-decimal, care convertează datele codului binar aplicat la intrarea sa într-un număr zecimat echivalent aplicat la ieşirea.

Circuitele logice combinate pot fi foarte simple sau foarte complicate și orice circuit combinațional poate fi implementat numai cu porți NAND și NOR deoarece acestea sunt clasificate ca porți "universale".

Cele trei moduri principale de specificare a funcției unui circuit logic combinațional sunt:

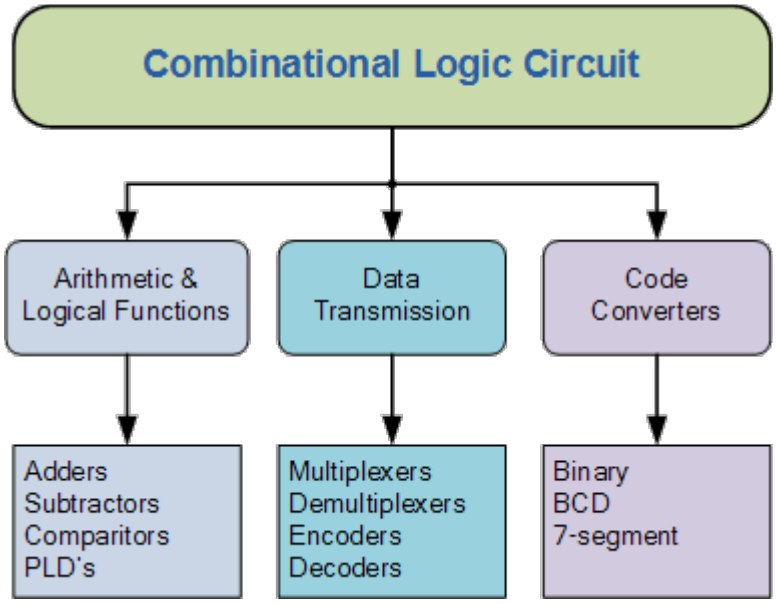
1. **Algebra booleană** - Aceasta formează expresia algebrică care arată funcția circuitului logic pentru fiecare variabilă de intrare Adevărată sau Falsă, rezultând o ieșire logică "1" sau ”0”.
2. **Tabel de Adevăr** - Un tabel de adevăr definește funcția unei porți logice prin prezentarea unei liste concise care arată toate stările de ieșire în formă unui table pentru fiecare combinație posibilă a variabilei de intrare pe care poarta ar putea să o primească.
3. **Diagrama logică** - Aceasta este o reprezentare grafică a unui circuit logic care arată cablurile și conexiunile fiecărei porți logice individuale, reprezentată printr-un simbol grafic specific, care implementează circuitul logic.



Din cauza că circuitele logice combinaționale sunt alcătuite numai din porți logice individuale, ele pot fi considerate ca "circuite de luare a deciziilor" și logica combinațională este de a combina porțile logice împreună pentru a procesa două sau mai multe semnale pentru a produce cel puțin un semnal de ieșire, dependent de funcția logică a fiecărei porți logice.

Circuitele combinaționale tipice alcătuite din porți logice individuale care efectuează o aplicație dorită sunt: Multiplexori, Demultiplexori, Encoderi, Decodori, Sumatoare complete și semisumatoare etc...

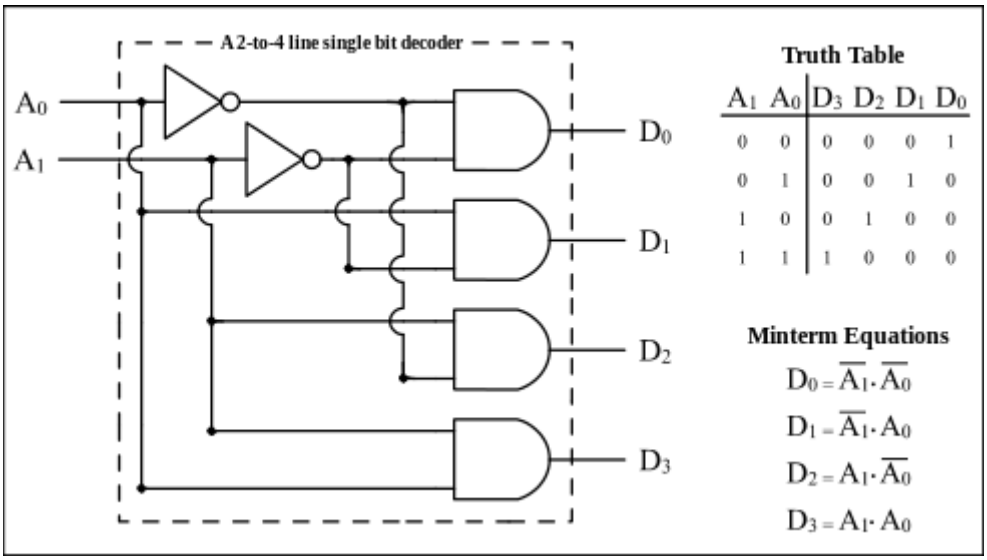
Clasificarea logicii combinate



Una dintre cele mai comune utilizări ale logicii combinaționale este în circuitele de tip Multiplexor și Demultiplexor. Aici sunt conectate mai multe intrări sau ieșiri la o linie comună de semnal, iar porțile logice sunt utilizate pentru a decoda o adresă pentru a selecta o singură intrare sau ieșire de date.

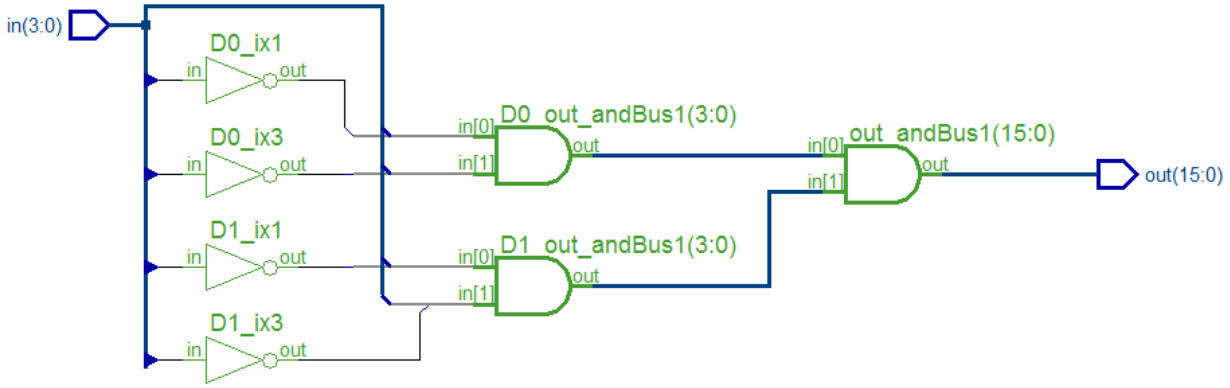
Proiectarea deodificatorului:

În electronica digitală, un decodificator binar este un circuit logic combinațional care convertează cod binar de la intrările, la un cod de poziţie. Se utilizează într-o mare varietate de aplicații, inclusiv demultiplexoarea datelor, afișarea pe șapte segmente și decodarea adreselor de memorie.



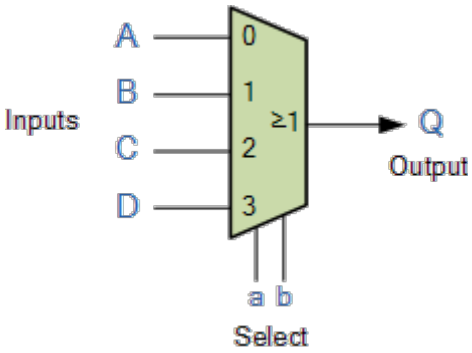
Descrierea în Verilog DCD

```
module DCD2x4(in,out);
	input[1:0] in;
	output[3:0] out;
	not N0(notn0,in[0]);
	not N1(notn1,in[1]);
	and a0(out[3],notn0,notn1);
	and a1(out[2],in[0],notn1);
	and a2(out[1],notn0,in[1]);
	and a3(out[0],in[0],in[1]);
endmodule

module DCD_TB();
	parameter input_nr = 4;
	parameter output_nr = input_nr*input_nr;
	reg[input_nr-1:0] generator;
	wire[output_nr-1:0] result;

	DCD4x16 DVT(generator,result);
	initial
		begin
			generator=0;
		end
	always
		begin
			#5
			$display("in=%b, out=%b", generator, result);
			generator=generator+1;
		end
endmodule
```



Schema sintetizata moduluilui DCD

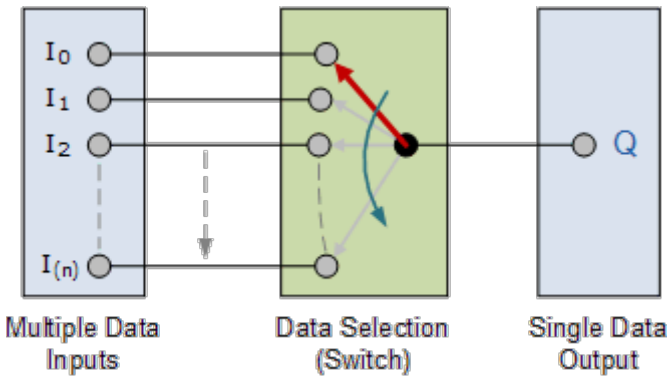
Proiectarea Multiplexorului



Multiplexarea este termenul generic folosit pentru a descrie modul de trimitere a unuia sau mai multor semnale analogice sau digitale pe o singură linie de transmisie comună la diferite momente de timp sau viteze. Dispozitivul pe care îl folosim pentru a executa aceasta funcţia o numim Multiplexor.

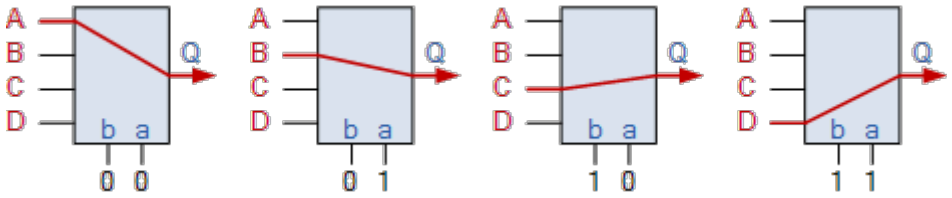
Multiplexorul, scurtat la "MUX" sau "MPX", este un circuit logic combinat conceput pentru a comuta una sau mai multe linii de intrare într-o singură linie comună de ieșire prin aplicarea unui semnal de selecţie(control). Multiplexoarele funcționează ca niște întrerupătoare rotative multi-poziționate cu poziționare foarte rapidă care conectează sau controlează mai multe linii de intrare numite "canale" şi le comutează la o ieșire.

Multiplexoarele sau MUX-urile pot fi ca circuite digitale realizate din porți logice de mare viteză folosite pentru a comuta date digitale sau binare, atît şi analogice care utilizează tranzistori, MOSFET sau relee pentru comutarea uneia dintre intrările de tensiune sau curent la o singură ieșire.

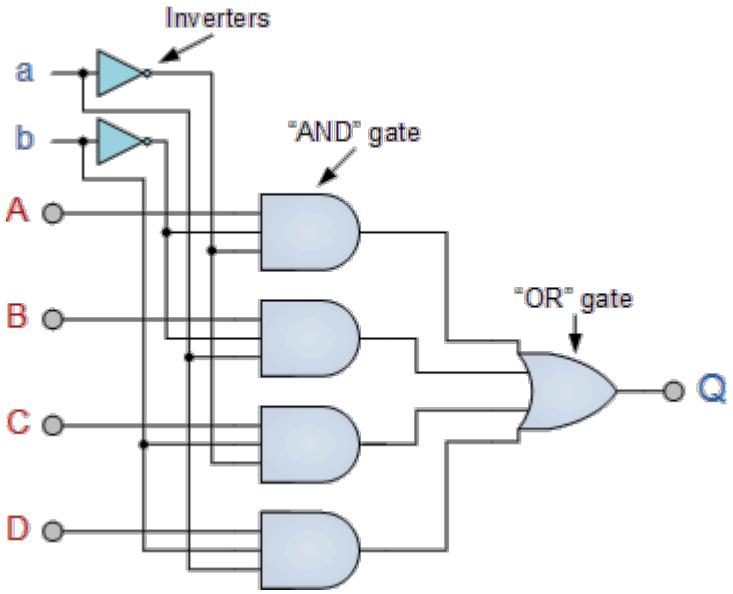


Schema generală de funţionare a multiplexorului

Selectarea liniei de intrare a multiplexorului



Adăugarea mai multor linii de adresă de control, (n) va permite multiplexorului să controleze mai multe intrări, deoarece poate comuta intrări 2n, însă fiecare configurație a liniei de control va conecta numai o intrare la ieșire.

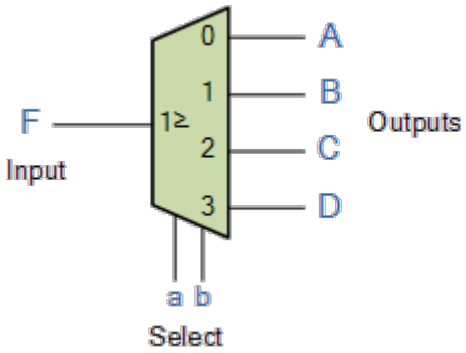


Schema electrica-logică multiplexorului.

```
module MUX(in,sel,out);
        input[3:0]in;
        input[1:0]sel;
        output out;
        wire[3:0]fire;
        DCD2x4 D0(sel,fire);
        and(out,fire[0],in[0]);
        and(out,fire[1],in[1]);
        and(out,fire[2],in[2]);
        and(out,fire[3],in[3]);
endmodule
```

```
module MUX_TB();
	reg[5:0]generator;
	wire result;
	MUX DUT(generator[3:0],generator[5:4],result);
	initial
		begin
			generator=0;
		end
	always
		begin
			#5
			generator=generator+1;
			$display("in=%b,out=%b",generator,result);
		end
endmodule
```

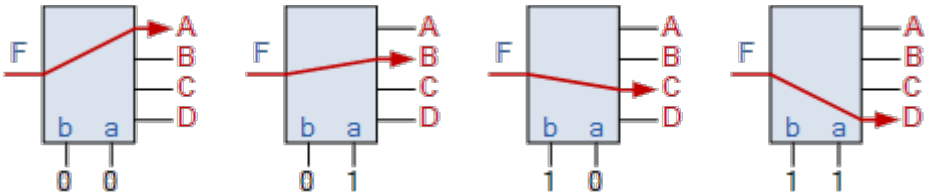
Proiectarea Demultiplexorului



Demultiplexorul

Distribuitorul de date, cunoscut mai des ca un Demultiplexer sau "Demux" pe scurt, este exact opusul Multiplexorului pe care l-am văzut la punctul precedent.

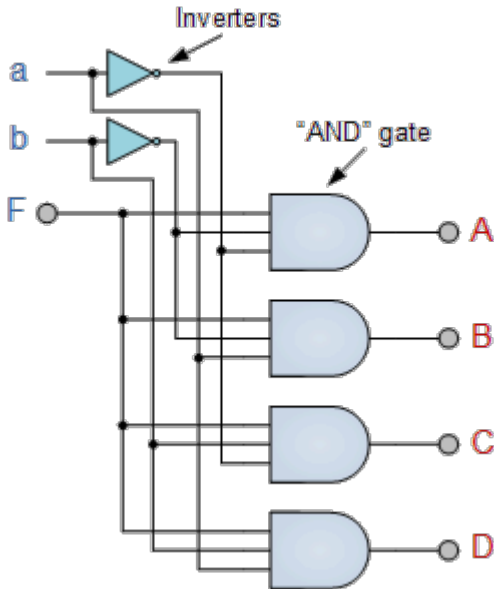
Demultiplexorul ia o singură linie de date de intrare și o comută pe oricare din mai multe linii de ieșire individuale. Demultiplexorul convertează un semnal de date seriale de la intrare în forma de date paralele la liniile sale de ieșire, după cum se arată mai jos.



Ca și în cazul circuitului de multiplexare anterior, adăugând mai multe intrări de linie de adresă, este posibilă comutarea mai multor ieșiri, oferind ieșiri dintr-o linie de date de la 1 la 2n.

Unele CI de demultiplexor standard au, de asemenea, un "pin de ieșire" suplimentar care dezactivează sau împiedică transmiterea intrării la ieșirea selectată. De asemenea, unele au blocuri integrate în ieșirile lor pentru a menține nivelul logic de ieșire după ce intrările adreselor au fost schimbate.

Cu toate acestea, în circuitele standarde, intrarea adresei va determina care ieșire de date unică va avea aceeași valoare ca şi intrarea, iar celelalte ieșiri de date vor avea valoarea logică "0".



Schema electrica-logică demultiplexorului.

Implementarea demultiplexorului in verilog

```
module DMUX(in,sel,out);
        input in;
        input[1:0]sel;
        output[3:0]out;
        wire[3:0]fire;
        DCD2x4 D0(sel,fire);
        and(out[0],fire[0],in);
        and(out[1],fire[1],in);
        and(out[2],fire[2],in);
        and(out[3],fire[3],in);
endmodule
```

Demultiplexor test bench;

```
module DMUX_TB;
        reg[2:0]generator;
        wire [3:0]result;
        DMUX DUT(generator[2],generator[1 : 0],result);
        initial
                begin
                        generator=0;
                end
        always
                begin
                #5
                generator=generator+1;
                $display("in=%b,out=%b",generator,result);
        end
endmodule
```

Mersul Lucrarii:

1. Implementaţi în cod Verilog **Decodificatorul** cu 2 intrari.
2. Implementaţi un **TestBench** pentru **Decodificator** şi testaţi modulul proiectat.
3. Sintetizaţi modulul Decodificator.
4. Implementaţi în cod Verilog **Multiplexorul** cu 2 intrari.
5. Implementaţi un **TestBench** pentru **Multiplexor** şi testaţi modulul proiectat.
6. Sintetizaţi modulul **Multiplexor**.
7. Implementaţi în cod Verilog **Demultiplexor** cu 2 intrari.
8. Implementaţi un **TestBench** pentru **Demultiplexor** şi testaţi modulul proiectat.
9. Sintetizaţi modulul **Demultiplexor**.