6 Circuite logice combinaționale elementare din categoria MSI

6.1 Obiective

Se analizează și se verifică funcționarea celor mai utilizate componente integrate din clasa MSI (Medium Scale Integration) precum demultiplexoarele, multiplexoarele și decodificatoarele. Se studiază codificatorul prioritar și convertorul din cod binar în cod Gray.

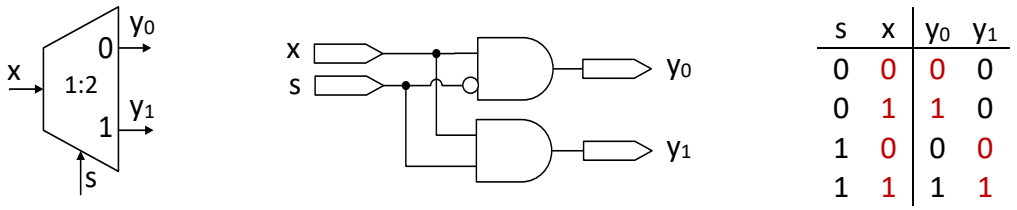
6.2 Considerații teoretice

Circuitele TTL din categoria MSI au în componența lor un număr de 50÷500 tranzistoare. Datorită funcționalității mai complexe față de porțile logice fundamentale, acestea sunt preferate în procesul de implementare a circuitelor, fiindcă pot reduce dimensiunea acestora, înlocuind un număr semnificativ de porți logice și conexiunile dintre ele. În general, se folosește o abordare mixtă, care combină circuitele MSI cu porțile logice fundamentale. Se vor studia în continuare câteva circuite MSI cu structură simplă, dar având o largă răspândire în implementarea hardware.

6.2.1 Circuite de demultiplexare

Demultiplexorul 1:*n* (DMUX 1:*n*) are un semnal de intrare de date **x**, *n* semnale de ieșire $\mathbf{y}_i$ și *m* semnale de selecție $\mathbf{s}_k$, unde $n=2^m$. Semnalul de ieșire cu indicele indicat de valoarea zecimală înscrisă pe semnalele de selecție va prelua valoarea semnalului de intrare, iar celelalte semnale de ieșire vor avea valoarea 0. Efectul demultiplexorului este descris de următoarea relație: $y_i = x$, dacă $i = \sum_{k=0}^{m-1} s_k \times 2^k$, altfel $y_i = 0$.

Circuitul care implementează demultiplexorul 1:2 este prezentat în Figura 6. 1, împreună cu simbolul asociat și tabelul de adevăr. Expresiile $y_0 = x \cdot \overline{s_0}$, $y_1 = x \cdot s_1$ se pot determina din tabelul de adevăr, prin minimizare la Forma Disjunctivă Minimă.



s	x	y_0	y_1
0	0	0	0
0	1	1	0
1	0	0	0
1	1	1	1

Figura 6. 1 Simbolul demultiplexorului DMUX 1:2 (stânga), schema logică (mijloc) și tabelul de adevăr (dreapta)

Demultiplexoarele cu număr mai mare de ieșiri se pot implementa folosind atât porți logice fundamentale, cât și tehnica de cascadare a unor demultiplexoare mai mici. În ce privește tehnica de cascadare, pentru a implementa un demultiplexor DMUX 1:4 se pot folosi 3 demultiplexoare DMUX 1:2, cascadate pe două niveluri logice, ca în figura următoare:

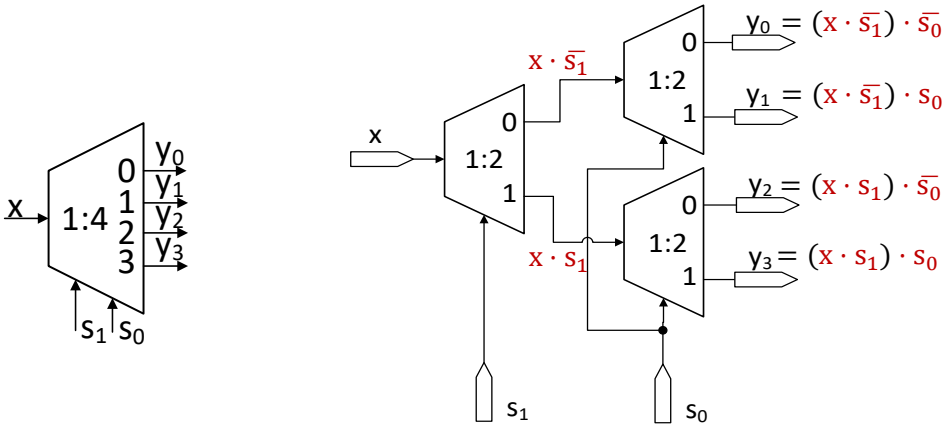


Figura 6. 2 Simbolul DMUX 1:4 (stânga) și implementarea prin cascadare (dreapta)

Notă: Dacă x = 1, atunci demultiplexorul activează numai ieșirea indicată de semnalele de selecție, având astfel funcționalitatea unui decodificator.

Există demultiplexoare cu un număr mai mare de ieșiri precum DMUX 1:8, 1:16, 1:32, etc.

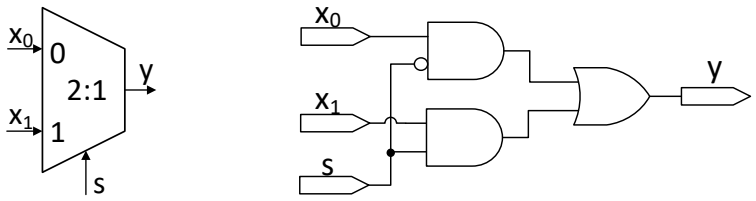
În Logisim, demultiplexoarele se regăsesc în librăria *Plexers*. Dimensiunea lor se poate stabili cu atributul `Select Bits`, care definește numărul de biți de selecție. Pentru ca intrarea și ieșirile de date să fie pe 1 bit se va seta `Data Bits = 1`. Opțional, demultiplexoarele pot prezenta o intrare de activare, care se poate elimina cu opțiunea `Include Enable = No`.

În Project Navigator, demultiplexoarele 1:4, 1:8, 1:16 se pot realiza cu ajutorul decodificatoarelor din categoria *Decoder*, denumite D2_4E, D3_8E, respectiv D4_16E. Intrarea de activare E (Enable) a decodificatoarelor poate fi folosită cu rol de intrare de date a demultiplexorului, iar intrările de date A_i, cu rol de selecție. Demultiplexorul 1:2 se poate regăsi în librăria locală a proiectului *ttl_env*, cu denumirea D1_2.

6.2.2 Circuite de multiplexare

Multiplexorul *n*:1 (MUX *n*:1) are *n* semnale de intrare de date $\mathbf{x}_i$, un semnal de ieșire **y** și *m* semnale de selecție $\mathbf{s}_k$, unde $n=2^m$. Semnalul de ieșire va prelua valoarea semnalului de intrare cu indicele indicat de valoarea zecimală înscrisă pe semnalele de selecție. În gereral, multiplexoarele sunt utilizate atunci când se dorește selectarea pe ieșire a unui singur semnal, din mai multe semnale de intrare. Efectul multiplexorului poate fi descris astfel: $y = x_i$, unde $i = \sum_{k=0}^{m-1} s_k \times 2^k$.

Multiplexorul 2:1 prezentat în Figura 6. 3 selectează o singură intrare, pe baza selecției s. Pornind de la tabelul de adevăr și realizând minimizarea la Forma Disjunctivă Minimă, se poate deduce următoarea expresie a ieșirii: $y = x_0 \cdot \overline{s_0} + x_1 \cdot s_1$.



s	x_0	x_1	y
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

Figura 6. 3 Simbolul multiplexorului MUX 2:1 (stânga), schema logică (mijloc) și tabelul de adevăr (dreapta)

Similar cu tehnica de la demultiplexoare, un multiplexor cu mai multe intrări se poate implementa prin cascadare de multiplexoare de dimensiuni reduse sau cu porți logice fundamentale. De exemplu, un multiplexor MUX 4:1 se poate realiza prin cascadarea a 3 multiplexoare MUX 2:1 amplasate pe 2 niveluri logice:

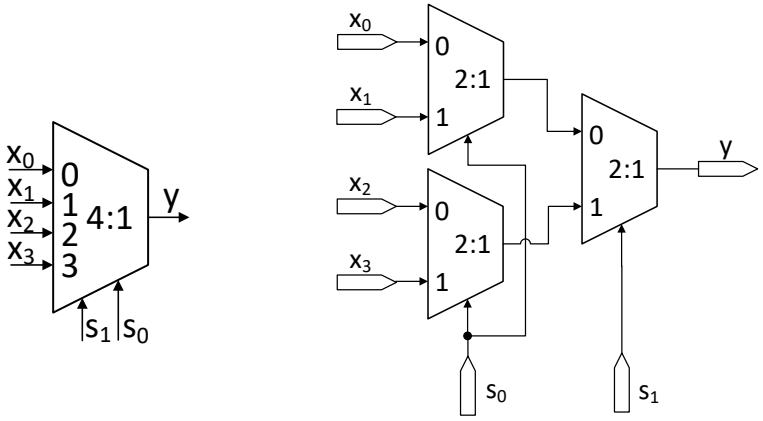


Figura 6. 4 Simbolul MUX 4:1 (stânga) și implementarea prin cascadare (dreapta)

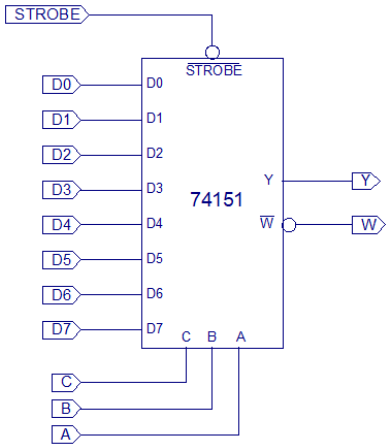
Se pot implementa astfel multiplexoare extinse precum: MUX 8:1, 16:1, 32:1, etc.

În Logisim, multiplexoarele se află în librăria *Plexers*. Atributul `Select Bits` definește numărul de biți de selecție; `Data Bits` setează numărul de biți pe intrările și ieșirea de date. Intrarea de activare se poate elimina setând `Include Enable = No`.

În Project Navigator, multiplexoarele sunt amplasate în categoria *Mux*, cu denumirile: M2_1, M2_1E, M4_1E, M8_1E. Sufixul E indică prezența intrării de activare E (Enable), care trebuie conectată la VCC, altfel ieșirea va fi 0.

Circuitul integrat 74151 (Figura 6. 5) implementează un MUX 8:1 cu intrările de date D_0, …, D_7, intrările de selecție A, B, C și 2 ieșiri complementare: Y, respectiv W. Ieșirea Y funcționează în logica pozitivă (activă pe 1), iar ieșirea W în logica negativă (activă pe 0): $W = \overline{Y}$. Selecțiile codifică numărul binar CBA_2. Suplimentar, există și o intrare de activare STROBE (S), în logica negativă. Dacă STROBE = 0 circuitul prezintă un regim de funcționare normal, altfel Y = 0 și $W = \overline{Y} = 1$.

În Logisim, toate circuitele integrate se regăsesc în librăria TTL. În proiectul *ttl_env*, integratele se găsesc în librăria proiectului, cu prefixul **TTL_** urmat de codul circuitului.



STROBE	C	B	A	Y	W
1	X	X	X	0	1
0	0	0	0	D_0	$\overline{D_0}$
0	0	0	1	D_1	$\overline{D_1}$
0	0	1	0	D_2	$\overline{D_2}$
0	0	1	1	D_3	$\overline{D_3}$
0	1	0	0	D_4	$\overline{D_4}$
0	1	0	1	D_5	$\overline{D_5}$
0	1	1	0	D_6	$\overline{D_6}$
0	1	1	1	D_7	$\overline{D_7}$

Figura 6. 5 Circuitul 74151 implementează un MUX 8:1

6.2.3 Circuite de decodificare

Decodificatorul are *n* semnale de intrare și maxim 2^n ieșiri. Decodificatorul activează ieșirea cu indicele indicat de codul aplicat pe intrări. Toate celelalte ieșiri sunt menținute inactive. Decodificatoarele pot avea ieșirile în logica pozitivă sau negativă.

Integratul 7442 (Figura 6. 6 – stânga) este un decodificator BCD-zecimal cu 4 intrări, care alcătuiesc codul binar $DCBA_2$, și 10 ieșiri în logica negativă, notate de la 0 la 9. Funcționarea acestuia este în conformitate cu Tabelul 6. 1. Pentru codurile binare în intervalul $1010_2 \div 1111_2$ toate ieșirile sunt inactive (au valoarea 1).

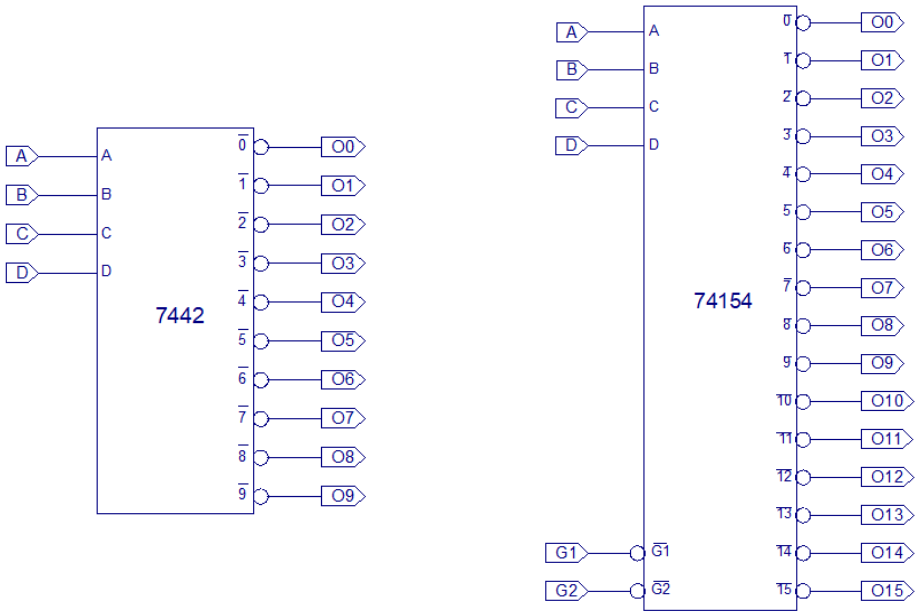


Figura 6. 6 Decodificatorul zecimal 7442 și decodificatorul hexazecimal 74154

Există și varianta cu 16 ieșiri, sub forma integratului 74154, denumit decodificator hexazecimal 4:16 (DCD 4:16). Orice combinație pe intrări, cu valori între 0 și 15, va activa ieșirea corespunzătoare (valoarea 0). Acest circuit are 2 intrări suplimentare de activare numite G_1 și G_2, care funcționează în logica negativă. Pentru activarea circuitului este necesar ca $G_1 = G_2 = 0$. Orice altă combinație inactivează circuitul plasând ieșirile pe 1.

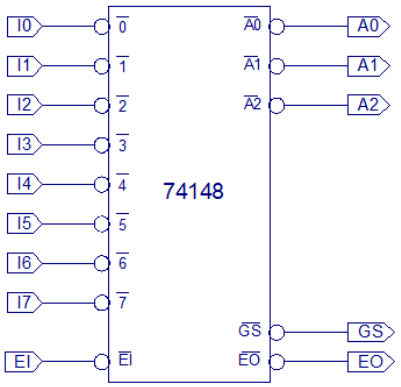
Tabelul 6. 1 Tabelul de adevăr al decodificatorului 7442

BCD				Zecimal									
D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

6.2.4 Circuite de codificare – codificatorul prioritar

Codificatorul prioritar are 2^n intrări și n ieșiri. Ieșirile codifică indicele intrării active, cu prioritatea cea mai mare. Prioritatea crește progresiv cu indicele.

Circuitul 74178 este un codificator prioritar cu 8 intrări și 3 ieșiri, în logica negativă (Figura 6. 7). Suplimentar, intrarea EI are rol de activare a circuitului (0 – activ, 1 – inactiv). Ieșirea GS se activează (GS = 0), când codul binar $A_{2:0}$ expus la ieșire este valid. Codul poate fi invalid când circuitul este dezactivat (EI = 1) sau dacă nicio intrare nu este activă. La rândul ei, ieșirea EO se activează (EO = 0), dacă circuitul este activ și toate intrările sunt inactive. Comportamentul circuitului este prezentat în tabelul din figura următoare:



	Intrări								Ieșiri				
EI	0	1	2	3	4	5	6	7	A_2	A_1	A_0	GS	EO
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	1	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	0	1	1	0	1
0	X	X	X	0	1	1	1	1	1	0	0	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

Figura 6. 7 Codificatorul prioritar 74148 (stânga) și tabelul de adevăr (dreapta)

6.2.5 Circuite de conversie – convertorul binar-Gray pe 4 biți

Convertorul realizează conversia între coduri de reprezentare. În general, sunt folosite pentru comunicația între sisteme care prelucrează informația în codificări diferite.

Convertorul binar-Gray pe 4 biți prezintă 4 intrări și 4 ieșiri, pe care se realizează codificările respective. La intrare se aplică un cuvânt în cod binar, iar la ieșire se obține codul Gray corespunzător (similar, se poate implementa și conversia inversă Gray-binar). Funcționalitatea circuitului este prezentată în tabelul următor:

B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

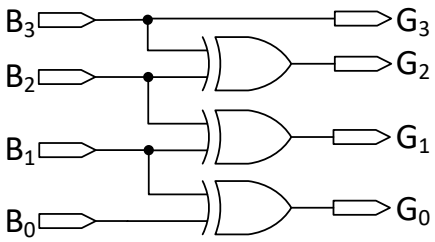


Figura 6. 8 Conversia binar-Gray (stânga) și implementarea convertorului (dreapta)

Fiecare ieșire G_i poate fi tratată ca o funcție logică aparte. În urma simplificării cu tehnici de minimizare și folosind proprietățile algebrei booleene, se obțin următoarele expresii pentru ieșiri: $G_3 = B_3$, $G_2 = B_2 \oplus B_3$, $G_1 = B_1 \oplus B_2$, $G_0 = B_0 \oplus B_1$.

6.3 Activități practice

1. Implementați și testați pe placă demultiplexorul 74151.
2. Implementați și testați pe placă decodificatorul zecimal 7442.
3. Implementați și testați pe placă codificatorul prioritar 74148.
4. Implementați cu porți logice și testați în Logisim un demultiplexor DMUX 1:2.
5. Implementați cu demultiplexoare DMUX 1:2 și testați în Logisim un demultiplexor DMUX 1:4.
6. Implementați cu multiplexoare MUX 2:1 și testați în Logisim un multiplexor MUX 4:1.
7. Implementați și testați în Logisim decodificatorul hexazecimal 74154.
8. Implementați cu porți XOR și testați în Logisim convertorul binar-Gray pe 4 biți.
9. Implementați funcția logică $f(A, B, C, D, E) = \overline{B} \cdot \overline{C} \cdot D + C \cdot \overline{D} \cdot E + B \cdot \overline{C} \cdot \overline{E} + A \cdot \overline{B} \cdot \overline{D} + A \cdot B \cdot C \cdot D$, folosind numai un multiplexor 16:1, semnalele 0 și 1 și variabilele conectate direct, fără a fi inversate.