CAPITOLUL 6

CIRCUITE LOGICE COMBINAȚIONALE (17.05.2004)

Sunt circuite cu *n* intrări, *m* ieşiri la care vectorul variabilelor logice de ieşire depinde numai de valoarea momentană a vectorului variabilelor logice de intrare. Se fabrică ca şi circuite integrate distincte sau sunt incluse în sisteme numerice integrate pe scară largă.

6.1. DECODIFICATORUL (DCD)

Funcție Serveşte la identificarea unui cod de intrare cu *n* biți prin activarea unei singure ieşiri (din cele *m*) corespunde codului de intrare. Fiecare ieşire corespunde unei anumite combinații a valorilor de intrare. În general între *n* şi *m* există relația $m = 2^n$, dar există şi DCD la care $m < 2^n$.
În schema bloc din figura 6.1, vectorul intrărilor este format din cele n linii notate x_0, x_1, ... x_{n-1}, iar vectorul ieşirilor (active SUS în varianta a, respectiv active JOS în varianta b) din liniile y_0, y_1, ..., y_{m-1}. En este o intrare de validare care poate inhiba simultan toate ieşirile DCD. În tehnologie CMOS, în seria 4000 ieşirile DCD disponibile sunt fie *active SUS*, fie *active JOS*, iar în tehnologie TTL (implicit şi în seriile CMOS rapide 74HC, 74LV, etc) ieşirile DCD sunt *active JOS*.

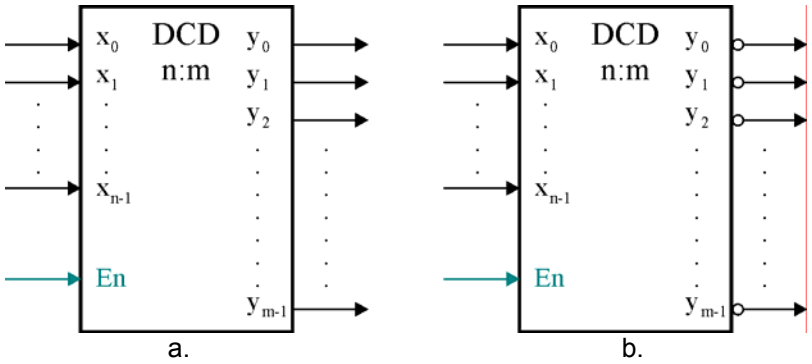


Figura 6.1. Schema bloc pentru un decodificator *n*:*m* cu validare,
a – ieşirile active SUS, b – ieşirile active JOS.

Cel mai simplu DCD are o intrare şi o ieşire, fiind realizat cu un inversor (figura 6.2). Un DCD 2:4 necesită 4 porți ŞI-NU şi două inversoare, ieşirile fiind active JOS.

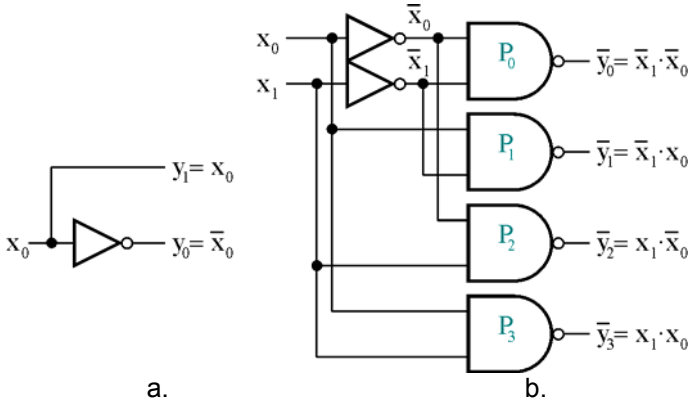


Figura 6.2. DCD simple – schema electrică,
a – DCD 1:2, ieşiri active SUS, b – DCD 2:4, ieşiri active JOS.

Schema electrică pentru un DCD 3:8 necesită 8 porți ŞI-NU cu câte 3 intrări (figura 6.3). Intrările se aplică prin perechi de inversoare pentru a asigura ca fiecare intrare să reprezinte o singură sarcină (TTL).
Schema se poate completa cu un circuit de validare (figura 6.4). Dacă circuitul nu este validat, toate ieşirile DCD sunt în starea 1. Pentru validare este necesar ca $E_2 = 1$, $E_{1A} = E_{2B} = 0$.

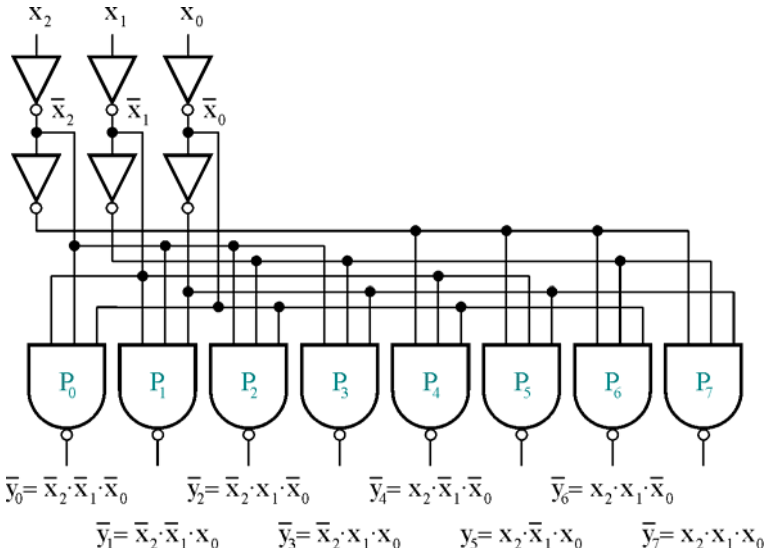


Figura 6.3. Structura unui DCD 3:8.

Fiecărei ieşiri îi corespunde un circuit ŞI-NU, ceea ce face ca ieşirile circuitului să fie active pe 0. Acest lucru înseamnă că ieşirea activată este pe 0 iar toate celelalte ieşiri sunt pe 1. De exemplu: pentru $x_0 = 1, x_1 = 0, x_2 = 1$, ieşirea $\overline{y_5} = \overline{E_2 \cdot \overline{E}_{1a} \cdot \overline{E}_{1b} \cdot (x_2 \cdot \overline{x}_1 \cdot x_0)}$ este pe 0 şi toate celelalte sunt 1.
Decodificatorul din figura 6.4 realizat în tehnologie TTL (74LS138) este foarte răspândit în aplicații datorită versatilității oferite de validarea multiplă.

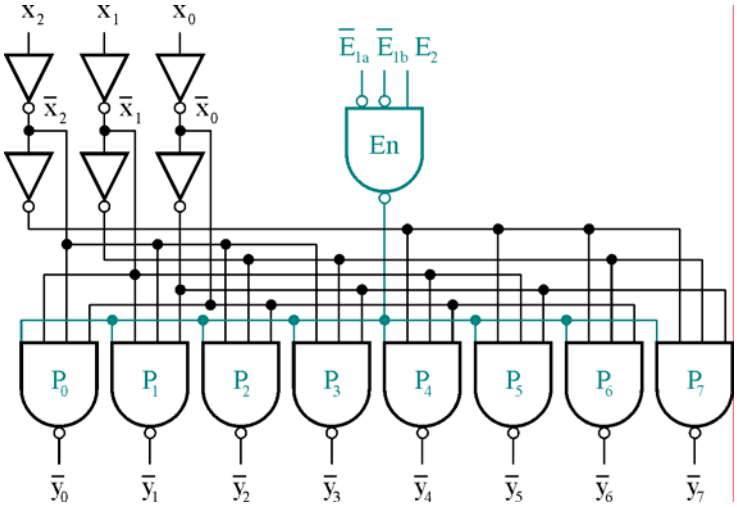


Figura 6.4. Un DCD 3:8 foarte răspândit, 74LS138.

6.1.1. Alte tipuri de decodificatoare

Principalele DCD realizate în tehnologie TTL sunt (figura 6.5):

- **74LS42** – DCD care serveşte pentru decodificarea cifrelor zecimale de la 0,..,9, codificate binar. Aici $m < 2^n$ (DCD 4:10), fără intrare de validare.

- **74LS139** – conține două decodificatoare binare 2:4 complet independente, fiecare decodificator având $n = 2$ şi $m = 4$ şi are o intrare de validare proprie activă pe 0.
- **74LS138** – DCD 3:8 conține un DCD 3:8 având $n = 3$ şi $m = 8$ şi are 3 intrări de validare active pe 0 (2), respectiv pe 1 (o intrare).
- **74LS154** – decodificator binar având $n = 4$ şi $m = 16$ (DCD 4:16).

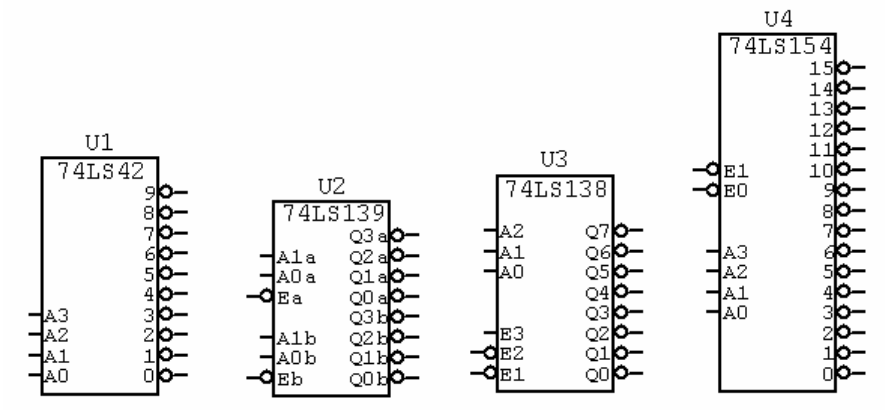


Figura 6.5. DCD în tehnologie TTL.

Principalele DCD realizate în tehnologie CMOS sunt (figura 6.6):

- **4555** – conține două DCD 2:4 independente cu ieşirile active SUS, fiecare având $n = 2$ şi $m = 4$, o intrare de validare proprie activă JOS.
- **4556** – conține două DCD 2:4 independente cu ieşirile active JOS, fiecare având $n = 2$ şi $m = 4$, o intrare de validare proprie activă JOS.
- **4028** – DCD 4:10 având $n = 4$ şi $m = 10$, cu ieşiri active SUS fără nici o intrare de validare.
- **4514** – DCD 4:16 cu ieşiri active SUS, intrare de validare activă JOS, având $n = 4$ şi $m = 16$.
- **4515** – DCD 4:16 cu ieşiri şi intrare de validare active JOS, având $n = 4$ şi $m = 16$.

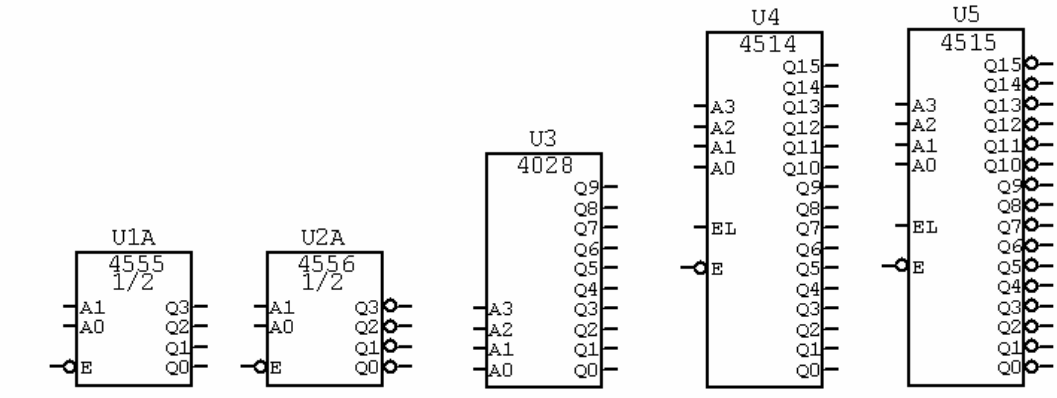


Figura 6.6. DCD în tehnologie CMOS.

O categorie aparte de decodificatoare sunt utilizate pentru comanda afişajelor cu 7 segmente (figura 6.7). În tehnologie CMOS se fabrică circuitele 4511 şi 4513, cu 4 intrări şi 7 ieşiri, iar în tehnologie TTL se produc circuite pereche (74LS47 şi 74LS247 pentru afişaje cu anod comun, respectiv 74LS48 şi 74LS248 pentru circuite cu catod comun).

- **4511** este un latch, decodificator şi etaj de ieşire capabil să furnizeze la ieşire un curent de 25 mA, potrivit pentru comanda afişajelor cu catod comun (LED). Poate afişa doar cifrele 0...9, pe care le poate şi memora de altfel.

- **4543** este un latch, decodificator şi etaj de ieşire capabil să furnizeze la ieşire un curent de 25 mA, potrivit pentru comanda afişajelor cu catod comun (LED), dacă PH = 1 logic, a afişajelor cu anod comun (LED), dacă PH = 0 logic, respectiv a afişajelor cu cristale lichide (LCD). Poate afişa doar cifrele 0...9, pe care le poate şi memora de altfel.
- **74LS47** şi **74LS247** sunt decodificatoare realizate pentru comanda afişajelor cu anod comun, care pot afişa 16 combinaţii (cifrele 0...9 şi alte 5 semne, plus afişaj stins).
- **74LS48** şi **74LS248** sunt decodificatoare realizate pentru comanda afişajelor cu catod comun, care pot afişa 16 combinaţii (cifrele 0...9 şi alte 5 semne, plus afişaj stins).

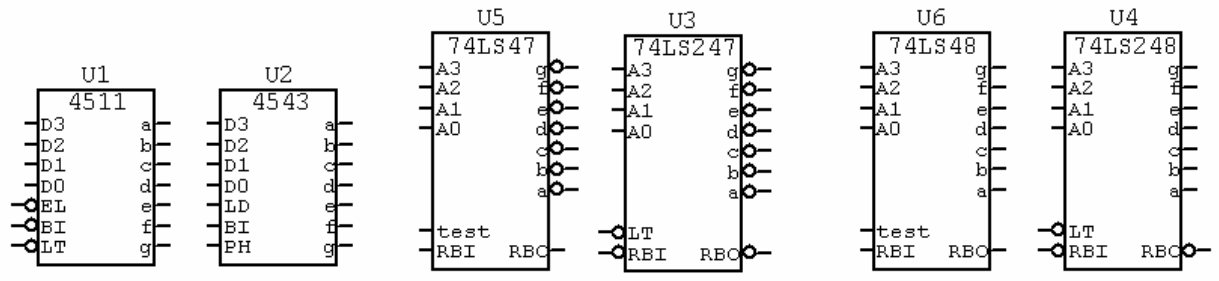


Figura 6.7. DCD binar – 7 segmente.

6.1.2. Extinderea capacităţii de decodificare

Extinderea capacităţii este una din cele mai comune probleme din aria de utilizare a circuitelor integrate digitale, aplicabilă practic la toate tipurile de circuite logice: decodificatoare, codificatoare, multiplexoare, numărătoare, memorii, etc. Pentru decodificatoare extinderea tipică se realizează după schema din figura 6.8, în care U_{4A} este utilizat pentru validarea unui singur DCD dintre U_0 - U_3, în funcţie de combinaţia variabilelor x_4, x_3.

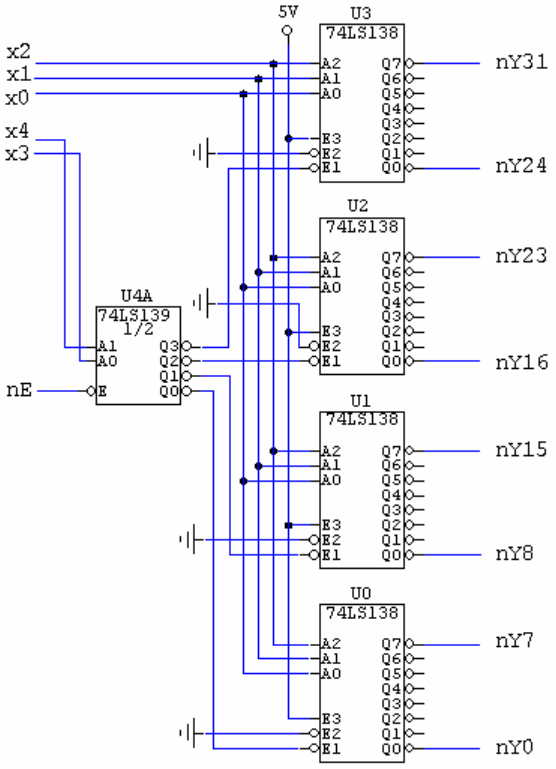


Figura 6.8. Obţinerea unui DCD 5:32 cu validare.

Combinațiile variabilelor x_4 - x_0 sunt prezentate în tabelul 6.1. nEN este o intrare globală de validare; pentru nEN = 1, toate ieşirile sunt inactive (1 logic). De obicei x_4 - x_0 sunt biți de adresă, rangurile mai semnificative fiind utilizate pentru selecția decodificatorului activ ($x_4 - x_3$ selectează $U_0 - U_3$), iar rangurile mai puțin semnificative o anumită ieşire dintr-un DCD ($x_2 - x_0$ selectează una din cele 8 ieşiri ale unui DCD).
Versatilitatea intrărilor de validare de la 74LS138 permite o implementare mai simplă a extinderii (figura 6.9), prin utilizarea unui circuit inversor şi renunțarea la posibilitatea validării globale.

Tabelul 6.1

Funcționarea DCD 5:32 cu validare

nE	x_4	x_3	x_2	x_1	x_0	DCD	Ieşire activă
1	x	X	x	x	x	-	-
0	0	0	0	0	0	U0	nY_0
0	0	0	0	0	1	U0	nY_1
.	.	.	.	.	.	.	.
0	0	0	1	1	1	U0	nY_7
0	0	1	0	0	0	U1	$nY8$
.	.	.	.	.	.	.	.
0	0	1	1	1	1	U1	nY_{15}
0	1	0	0	0	0	U2	nY_{16}
.	.	.	.	.	.	.	.
0	1	0	1	1	1	U2	nY_{23}
0	1	1	0	0	0	U3	nY_{24}
.	.	.	.	.	.	.	.
0	1	1	1	1	1	U3	nY_{31}

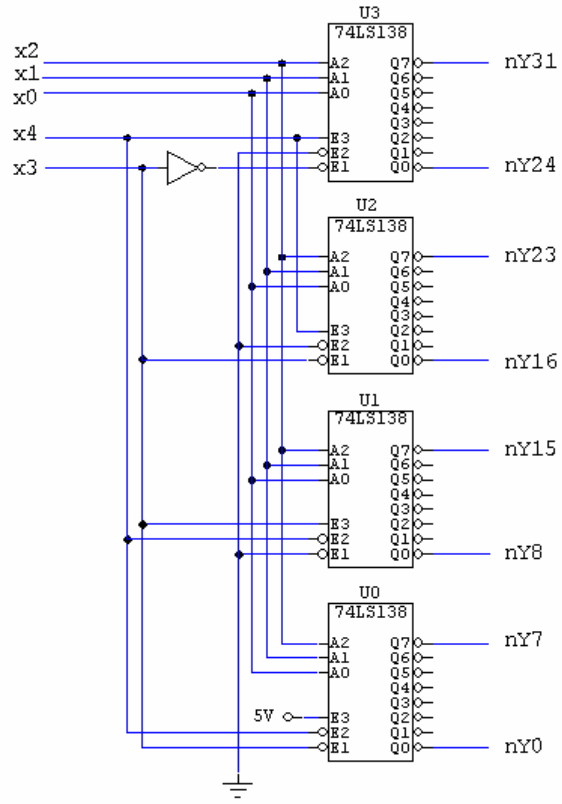


Figura 6.9. Obținerea unui DCD 5:32, varianta a II-a.

Într-un microsistem DCD se utilizează la selecția diferitelor circuite integrate sau *porturi*. Un circuit complex (port) poate răspunde la mai multe adrese adiacente (de exemplu circuitul Intel 8255 are o intrare de selecție circuit nCS şi două linii de adresă A0 şi A1, conținând astfel 4 porturi – 3 de date şi unul de comandă). Decodificarea adreselor se poate face *complet*, caz în care *toate* liniile de adresă ajung la DCD sau incomplet – doar o parte din liniile de adresă şi sau adrese sunt decodificate. Exemplele din figurile 6.8 şi 6.9 sunt decodificări complete, pe când cea din figura 6.10 este incompletă.

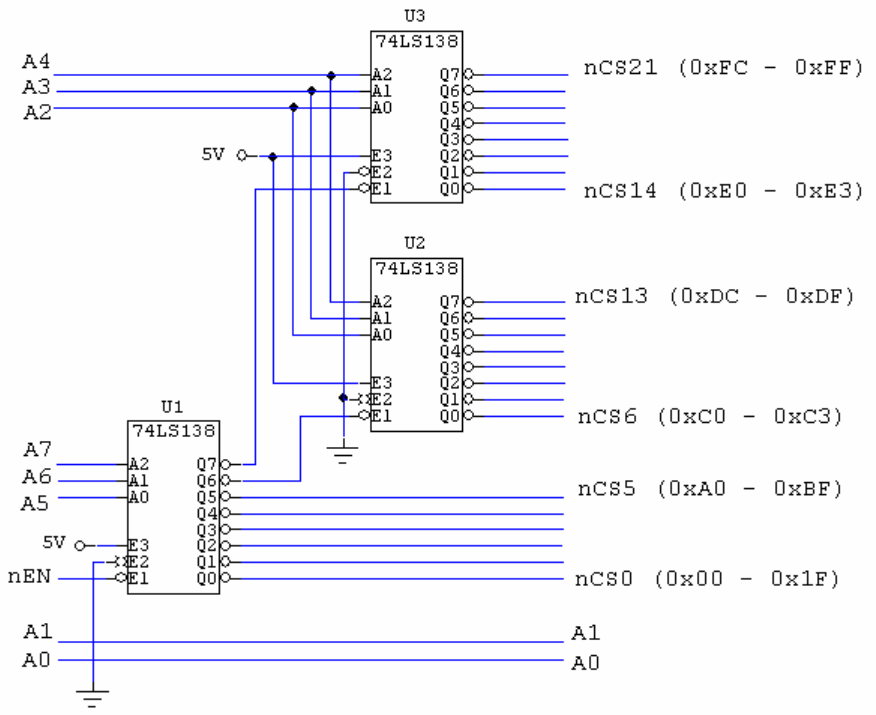


Figura 6.10. Decodificare incompletă a magistralei de adrese într-un microsistem.

Tabelul 6.2

Decodificare completă şi incompletă cu explicarea spațiului de adresare

nEN	A7	A6	A5	A4	A3	A2	A1	A0	nCS	Domeniu adresare	Exemple de trunchiere	DCD
1	X	X	X	X	X	X	X	X	-	-	-	-
0	0	0	0	X	X	X	X	X	nCS0	0x00 – 0x1F	0x00, 0x04,..., 0x1C	1
0	0	0	1	X	X	X	X	X	nCS1	0x20 – 0x3F	0x21, 0x25,..., 0x3D	1
0	0	1	0	X	X	X	X	X	nCS2	0x40 – 0x4F	0x42, 0x46,..., 0x5E	1
0	0	1	1	X	X	X	X	X	nCS3	0x60 – 0x7F	0x63, 0x67,..., 0x9F	1
0	1	0	0	X	X	X	X	X	nCS4	0x80 – 0x9F	0x81, 0x85,..., 0x9D	1
0	1	0	1	X	X	X	X	X	nCS5	0xA0 – 0xBF	0xA3, 0xA7,..., 0xBF	1
0	1	1	0	0	0	0	X	X	nCS6	0xC0 – 0xC3	-	1, 2
0	1	1	0	0	0	1	X	X	nCS7	0xC4 – 0xC7	-	1, 2
.	.	.	.	.	.	.	.	.	.	.	.	.
0	1	1	0	1	1	1	X	X	nCS13	0xDC – 0xDF	-	1, 2
0	1	1	1	0	0	0	X	X	nCS14	0xE0 – 0xE3	-	1, 3
0	1	1	1	0	0	1	X	X	nCS15	0xE4 – 0xE7	-	1, 3
.	.	.	.	.	.	.	.	.	.	.	.	.
0	1	1	1	1	1	1	X	X	nCS21	0xFC – 0xFF	-	1, 3

6.1.3. Aplicații ale decodoficatoarelor

1. **Identificarea unui cod** – este chiar funcția fundamentală a unui DCD.

2. **Implementarea funcţiilor logice** cu *n* variabile, unde *n* corespunde cu numărul de intrări de cod ale DCD. Implementarea funcţiilor logice folosind DCD este foarte avantajoasă pentru că ieşirile unui DCD binar reprezintă termenii $\overline{P}$ din exprimarea canonică disjunctivă a funcţiilor logice. Numărul funcţiilor (de acelaşi număr de variabile binare) ce pot fi implementate nu este limitat decât de factorul de branşament la ieşire, ce corespunde ieşirilor DCD. Există două variante de implementare: DCD şi o poartă ŞI-NU, respectiv un DCD şi o poartă ŞI.

În primul caz la intrările circuitului ŞI-NU se conectează ieşirile DCD ce corespund termenilor $\overline{P}$ cuprinşi în funcţie. Pentru a doua variantă, la intrările circuitului ŞI se conectează ieşirile DCD ce corespund termenilor $\overline{P}$ necuprinşi în funcţia *F*.

Fie funcţia $\boldsymbol{F = P_0 + P_3 + P_5}$. În acest caz $n = 3$ şi se poate utiliza un DCD 3:8.

Varianta DCD + ŞI-NU

$$F = P_0 + P_3 + P_5 = \overline{\overline{P_0} \cdot \overline{P_3} \cdot \overline{P_5}} = \overline{\overline{\overline{A} \cdot \overline{B} \cdot \overline{C}} \cdot \overline{A \cdot B \cdot \overline{C}} \cdot \overline{A \cdot \overline{B} \cdot C}}$$

Varianta DCD + ŞI

$\overline{F} = P_1 + P_2 + P_4 + P_6 + P_7$, adică $F = \overline{P_1 + P_2 + P_4 + P_6 + P_7} = \overline{P_1} \cdot \overline{P_2} \cdot \overline{P_4} \cdot \overline{P_6} \cdot \overline{P_7}$

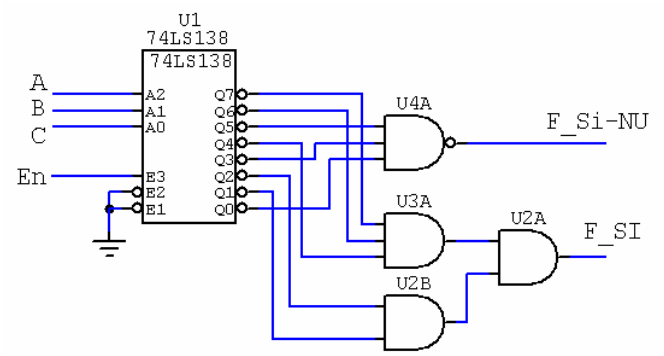


Figura 6.11. Implementarea de funcţii cu DCD şi porţi.

Pentru reducerea numărului de circuite, în cazul funcţiilor de 3 variabile se utilizează varianta cu porţi ŞI-NU pentru funcţii cu maxim 4 termeni *P*, iar varianta cu ŞI atunci când numărul termenilor care nu apar în funcţie este mai mic de 4. În tehnologie TTL circuitele ŞI-NU se fabrică cu 2, 3, 4, 8, 13 intrări, pe când circuitele ŞI se fabrică cu 2, 3, 4 intrări.

3. **Comanda afişajelor cu 7 segmente**. Schemele electrice din figurile 6.9 şi 6.10 sunt evident incomplete, lipsind rezistoarele de limitare a curentului prin segmente. Acestea se conectează în serie cu ieşirile DCD şi au o valoare tipică de 330Ω. În ambele figuri este redată situaţia afişării semnului corespunzător la 1100_2 = 0x0C.

Intrarea nLT (*Lamp Test*) activă JOS determină atunci când este trecută în 0 logic aprinderea tuturor segmentelor, permiţând astfel verificarea afişajului. nRBI (*Ripple Blank Input*) este intrarea de mascare a zerourilor nesemnificative, iar nRBO este ieşirea corespunzătoare. Pinul nRBO oferă şi funcţia suplimentară de ştergere (BI *Blanking Input*) prin care se poate comanda stingerea tuturor segmentelor afişajului.

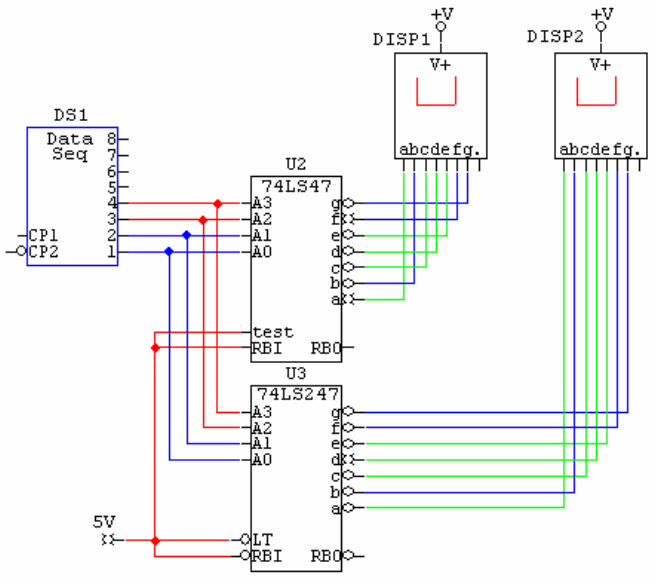


Figura 6.12. Afişaje cu anod comun comandate de circuitele TTL 74LS47, respectiv 74LS247.

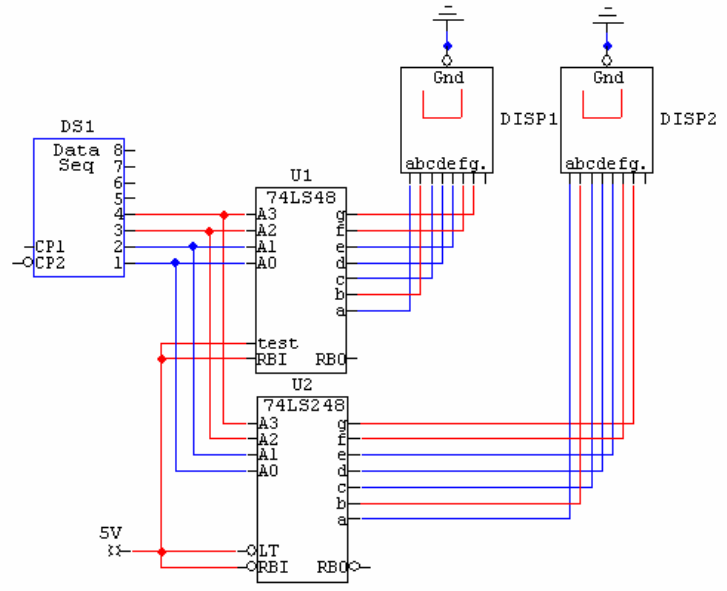


Figura 6.13. Afişaje cu catod comun comandate de circuitele TTL 74LS47, respectiv 74LS247.

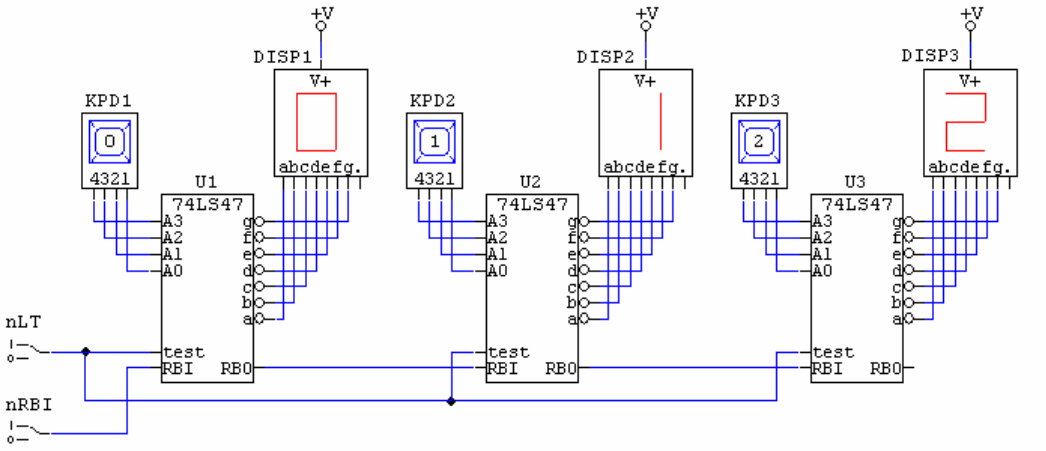


Figura 6.14. Afişaj cu trei cifre – configurație standard.

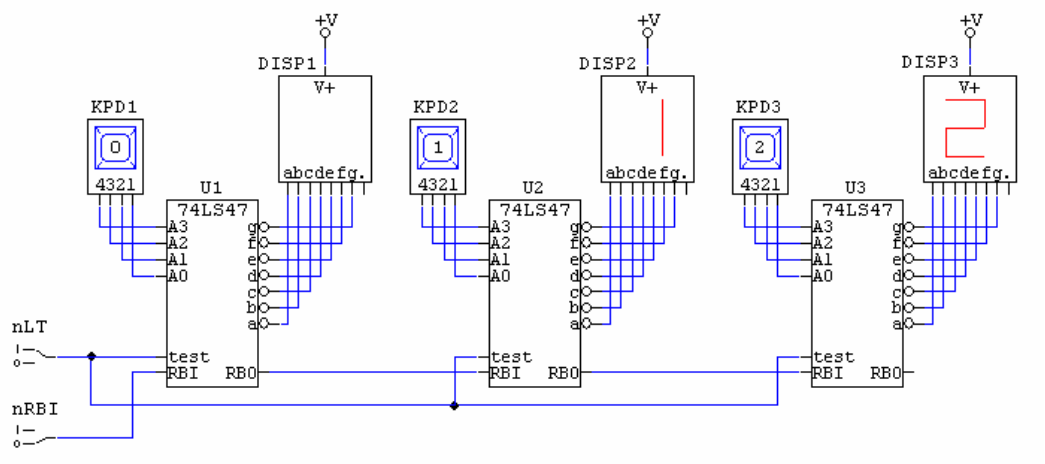


Figura 6.15. Afişaj cu trei cifre – mascarea zeroului semnificativ.

Intrarea RBI permite stingerea zerourilor nesemnificative atunci când este conectată la ieşirea RBO a decodificatorului de rang imediat adiacent.

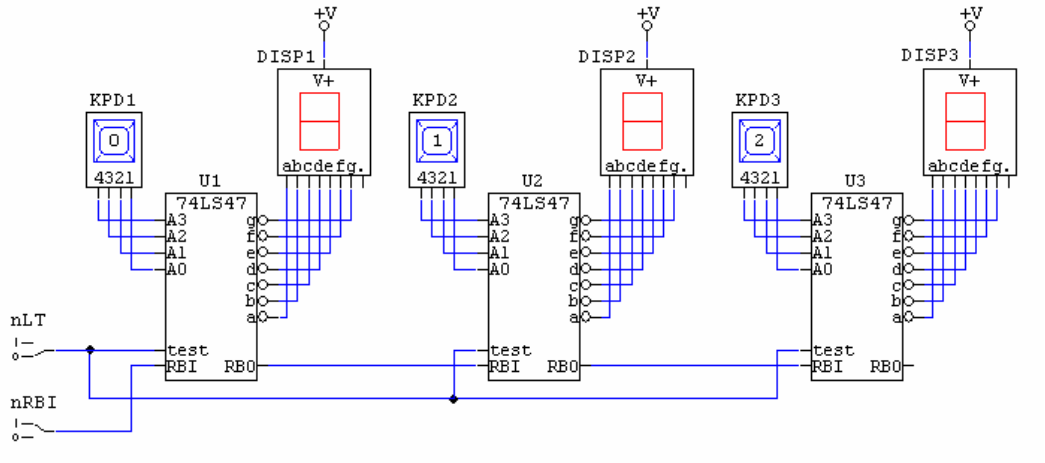


Figura 6.16. Afişaj cu trei cifre – testarea segmentelor.

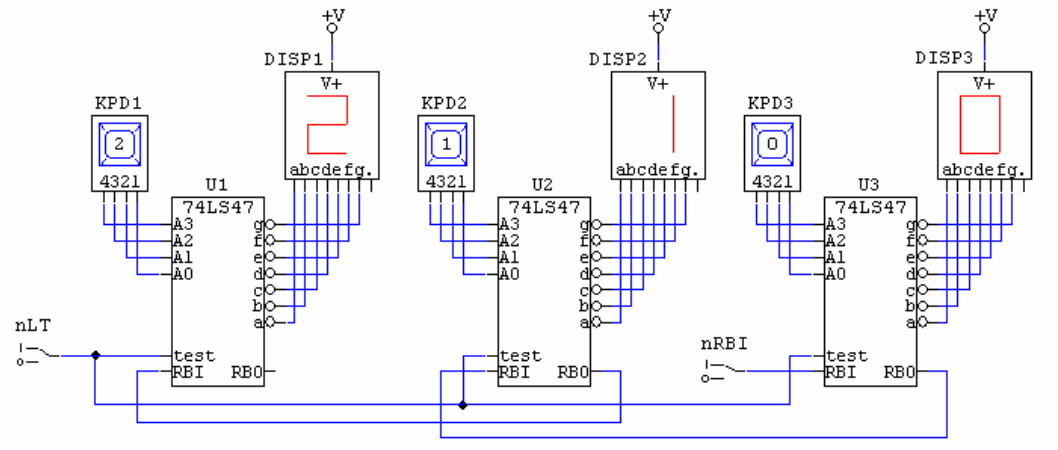


Figura 6.17. Afişaj cu trei cifre cu zero în poziția cea mai puțin semnificativă.

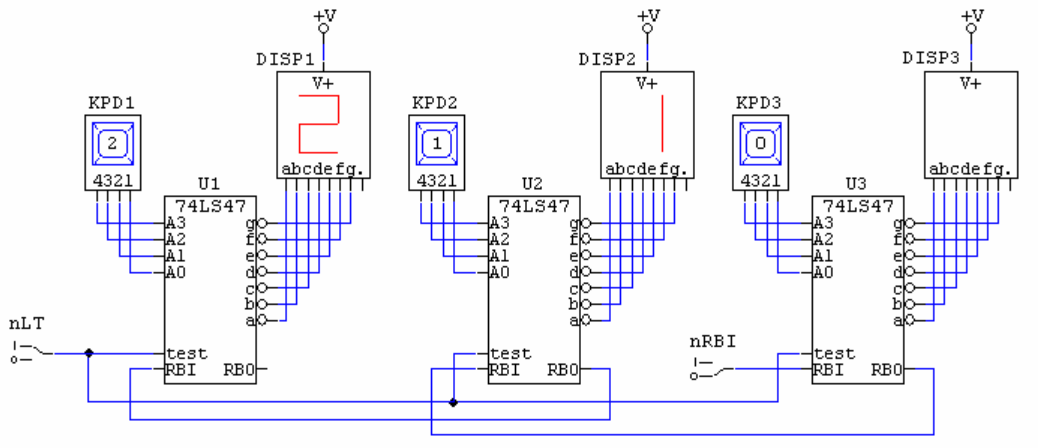


Figura 6.18. Afişaj cu trei cifre – mascarea zeroului mai puțin semnificativ.

Pentru un număr mai mare de cifre comandate se folosesc tehnici de multiplexare a afişării, care vor fi prezentate la §x.x.

4. **Utilizarea unui DCD 4:10 fără intrare de validare ca DCD 3/8 cu intrare de validare**.

Se realizează utilizând intrarea de rang semnificativ, notată x3, D sau A3 ca intrare de validare activă jos (nEN). Se pot folosi doar ieşirile nY0 – nY7 ale DCD.

5. **Temă.** Să se proiecteze un circuit cu 4 intrări care să semnalizeze momentele în care exact una dintre intrări este 1 logic.
Rezolvare: sunt necesare un DCD 4/16 cu ieşiri active sus şi o poartă SAU cu 4 intrări.

6.2. DEMULTIPLEXORUL (DMUX)

Funcție. Asigură transmiterea datelor de la o singură sursă de date la *m* receptoare succesive. Selecția receptorului se realizează printr-un cod de selecție de $n = \log_2 m$ biți.

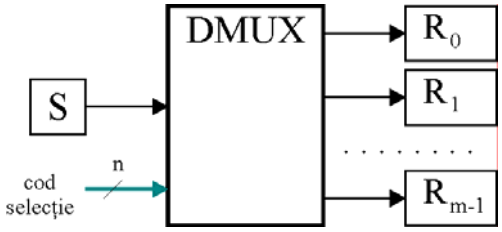


Figura 6.20. Demultiplexorul – schema bloc de utilizare.

Funcția definită anterior sugerează posibilitatea de a folosi orice DCD care are cel puțin o intrare de validare pentru realizarea unui DMUX. Modul în care un DCD 74LS138 devine DMUX şi noua semnificație a intrărilor este ilustrată în figura 6.21. Considerând codul de selecție $A = 1$, $B = 1$, $C = 0$, datele prezente la intrarea de date D_i se vor regăsi la ieşirea L_3 dacă şi numai dacă circuitul este validat corect, deoarece $L_3 = \overline{(1 \cdot \overline{D_i} \cdot 1) \cdot (A \cdot B \cdot \overline{C})} = D_i$. Datele transmise serial suferă două inversări, deci ajung la receptorul selectat neinversate. În cazul în care D_i se conectează la G_1, datele ajung la receptoare complementate (dacă $G_2 = Di$ şi $G_{1A} = G_{1B} = 0$, atunci $L_3 = \overline{D_i}$).

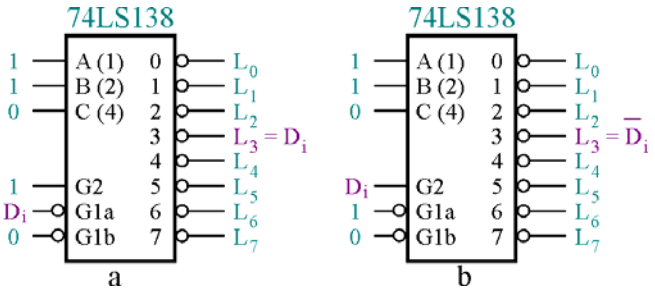


Figura 6.21. Utilizarea unui DCD ca DMUX.
a. fără inversare; b. cu inversare.

6.2.1. Extinderea capacității de demultiplexare

Extinderea capacității de demultiplexare se realizează similar cu extinderea decodificării. De exemplu două DCD 4:16 se pot folosi ca un DCD 5:32 sau DMUX cu 32 linii de ieşire şi cod de selecție de 5 biți.

6.2.2. Aplicații

Comanda în impulsuri a unui motor de curent continuu cu punte H şi DMUX.

În acționările electrice un motor de curent continuu se comandă cel mai adesea modulând în durată un semnal dreptunghiular de frecvență fixă. Tensiunea medie la bornele motorului este direct proporțională cu factorul de umplere al semnalului PWM (*pulse-width modulation*) de comandă. Această metodă simplă permite variația în limite largi a turației unui motor de curent continuu, dar nu şi modificarea sensului de rotație – caz în care este necesară utilizarea unei punți H.
O punte H se poate realiza şi cu tranzistoare (bipolare sau MOS), dar în cazul cel mai simplu o punte integrată satisface cerințele pentru puteri mici-medii.

TA 8050P este o punte H integrată cu tranzistoare bipolare produse de firma Toshiba, care are următoarele caracteristici:

- Comanda bidirecțională a motorului de curent continuu;
- Patru moduri de operare: Direct, Invers, Stop şi Frânare;
- Comanda se face cu nivelurid e tensiune TTL;
- curent comandat: 1,5 A;
- Tensiuni de alimentare recomandate cuprinse între 6 V şi 16 V;
- Diode de protecție împotriva tensiunii de autoinduse;
- Protecție integrată la:
  - Scurtcircuit;
  - Supraîncălzire;
  - Supratensiune;
- Capsulă HSIP cu 7 pini.

Schema tipică de utilizare este cea din figura 6.22, iar tabelul de funcționare 6.x.

Tabelul 6.x

Comanda punții TA 8050P

Intrare		Ieşire		Mod de lucru
Di1	**Di2**	**M(+)**	**M(-)**	
0	0	OFF (HiZ)	OFF (HiZ)	Stop
0	1	L	H	Invers
1	0	H	L	Direct
1	1	L	L	Frânare

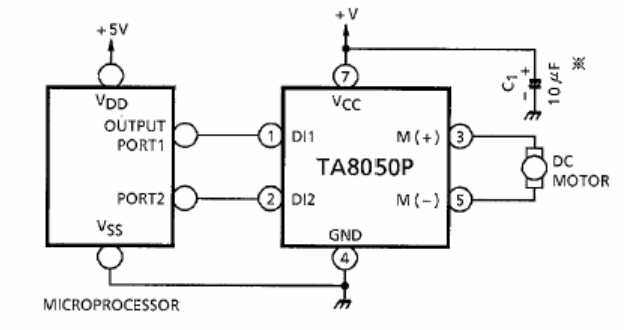


Figura 6.22. Schema de aplicație a punții H.

6.3. MULTIPLEXORUL (MUX)

Funcție. Permite transmiterea succesivă a datelor de la *m* surse de date la un receptor unic. În cazul general, un MUX este prevăzut cu:

- *m* canale de date de intrare de câte *b* biți;
- un canal ieşire pe *b* biți;
- un cod de selecție a canalului de intrare cu *n* biți unde $n = \log_2 m$;
- o intrare de validare a funcționării.

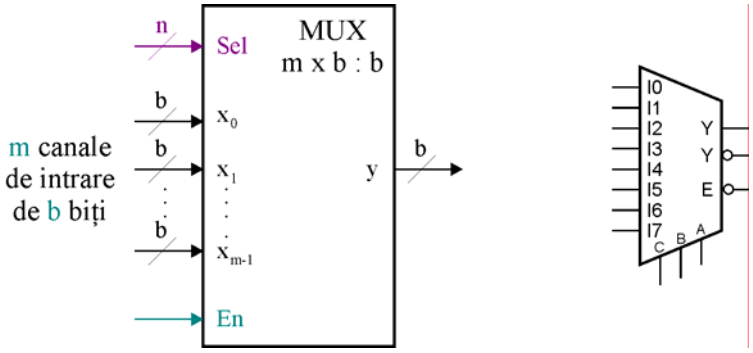


Figura 6.24. Schema funcțională a unui multiplexor şi simbolul unui MUX 8:1.

Cel mai simplu MUX are 2 intrări şi o ieşire (figura 6.25.a). Un MUX 4:1 necesită 4 porți ŞI-NU, o poartă SAU şi minim 3 inversoare (figura 6.25.b).

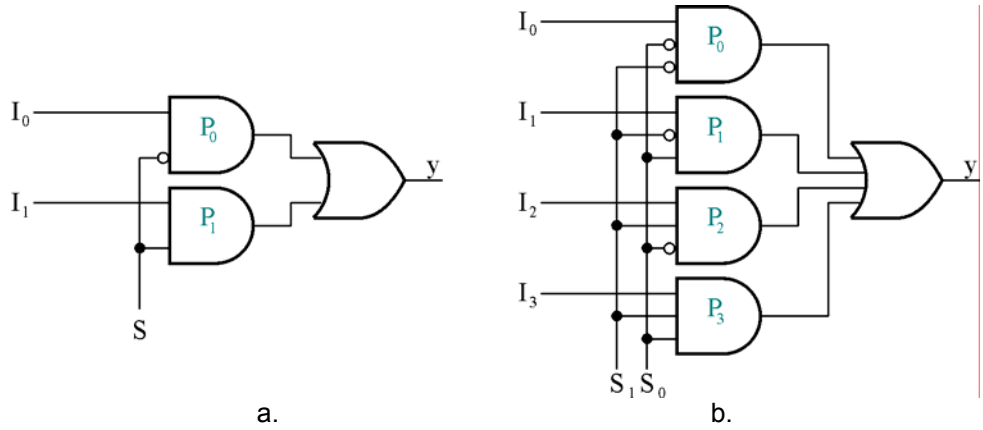


Figura 6.25. MUX simple – schema electrică,
a – MUX 2:1, b – MUX 4:1.

6.3.1. Exemple de multiplexoare TTL

- **74LS151** (figura 6.26):

Este un multiplexor cu $m = 8$ canale de intrare, $n = \log_2 m = 3$ linii de selecție şi un canal de ieşire de $b = 1$ bit.

Ecuația care descrie funcționarea MUX 74LS151 este:

$$Y = En\sum_{i=0}^{7} I_i \cdot P_i = EN \cdot [I_0 \cdot (\overline{S_0} \cdot \overline{S_1} \cdot \overline{S_2}) + I_1 \cdot (S_0 \cdot \overline{S_1} \cdot \overline{S_2}) + I_2 \cdot (\overline{S_0} \cdot S_1 \cdot \overline{S_2}) + I_3 \cdot (S_0 \cdot S_1 \cdot \overline{S_2}) +$$

$$+ I_4 \cdot (\overline{S_0} \cdot \overline{S_1} \cdot S_2) + I_5 \cdot (S_0 \cdot \overline{S_1} \cdot S_2) + I_6 \cdot (\overline{S_0} \cdot S_1 \cdot S_2) + I_7 \cdot (S_2 \cdot S_1 \cdot S_0)]$$

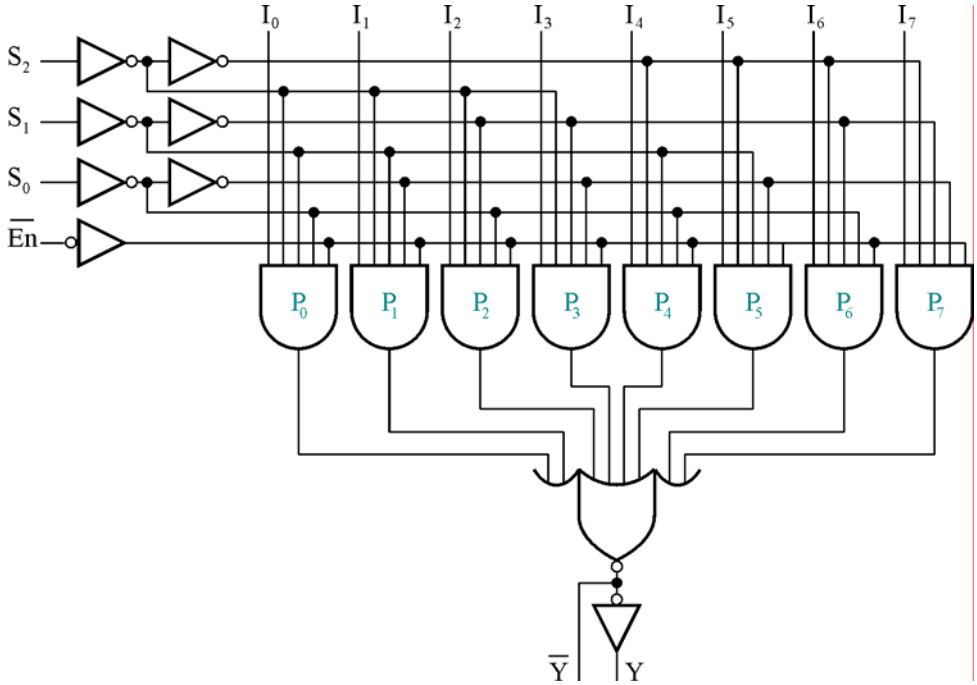


Figura 6.26. MUX 74LS151.

- **74LS251**:

Are o structură asemănătoare cu 74LS151 (m = 8, n = 3, b = 1), față de care prezintă însă următoarele deosebiri:

- ieşirile Y şi $\overline{Y}$ sunt de tip trei stări, validate de semnalul $\overline{OE}$ activ pe 0 (*Output Enable*), aşa cum se poate observa în figura 6.27. Nu mai există (şi nici nu mai este necesară) intrarea $\overline{En}$ (înlocuită cu $\overline{OE}$). Dacă $\overline{OE} = 1$, ambele ieşiri sunt în stare de impedanță ridicată HiZ.
- sunt circuite utilizate pentru conectare la magistrale

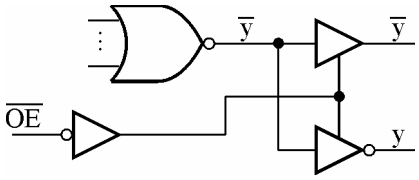


Figura 6.27. Ieşirea multiplexorului 74LS251.

- **74LS157** (figura 6.28).

Este un multiplexor cvadruplu 2:1 cu intrare de validare, având m = 4 canale, b = 2 biți, n = 1 bit. Funcționare: Dacă $\overline{En} = 1$, liniile L_1, L_2 vor fi 0 logic, iar ieşirile $Y_0,\ldots,Y_3$ vor fi de asemenea 0 logic. Dacă $\overline{En} = 0$, porțile P_1, P_2 funcționează pentru semnalul de selecție ca inversoare; pentru S = 0 sunt selectate intrările I_{0a}, I_{0b}, I_{0c}, I_{0d}, iar pentru S = 1 sunt selectate intrările I_{1a}, I_{1b}, I_{1c}, I_{1d}.

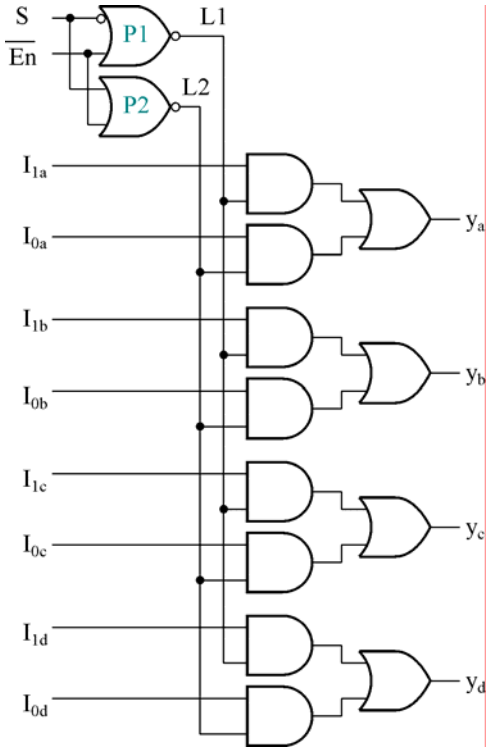


Figura 6.28. Multiplexor cvadruplu 2:1 cu intrare de validare, 74LS157.

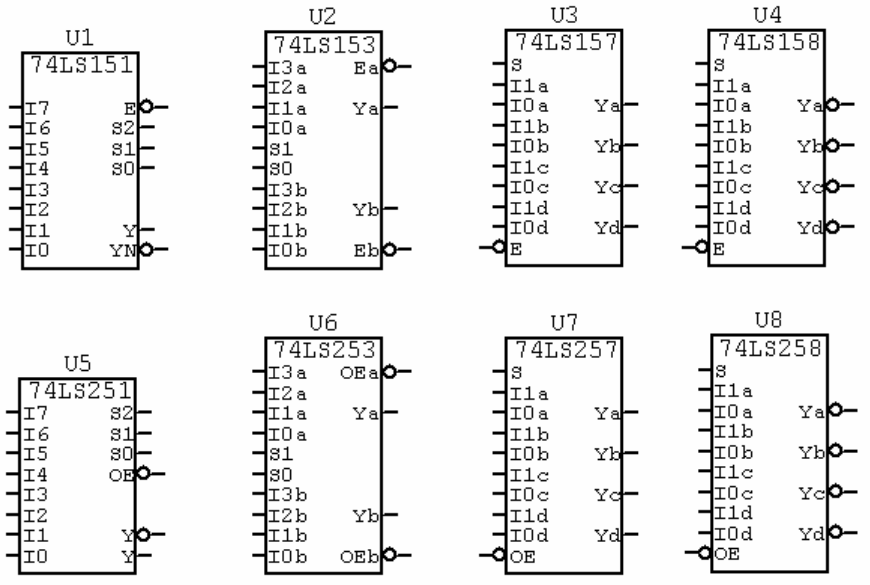


Figura 6.29. Multiplexoare în tehnologie TTL.

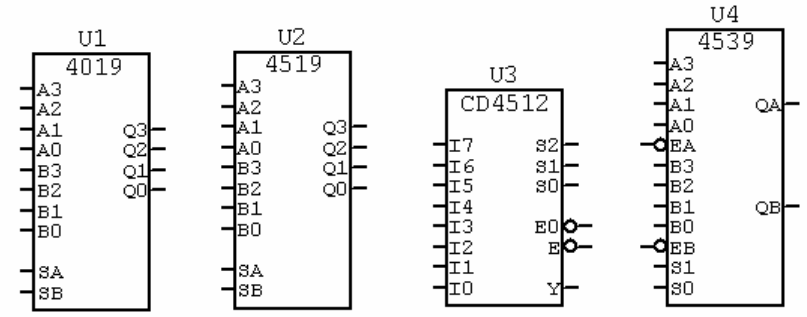


Figura 6.30. Multiplexoare în tehnologie CMOS.

6.3.1. Extinderea capacității de multiplexare

1. Extinderea numărului de canale *m*, fără modificarea numărului de biți *b* pe canal.
2. Extinderea numărului de biți *b*, fără modificarea numărului de canale *m*.
3. Extinderea numărului de canale *m* şi a numărului de biți *b* pe canal.

6.3.2. Aplicații ale multiplexoarelor

1. **Transmiterea succesivă a datelor de la *m* surse de date la un singur receptor** (aplicația fundamentală):

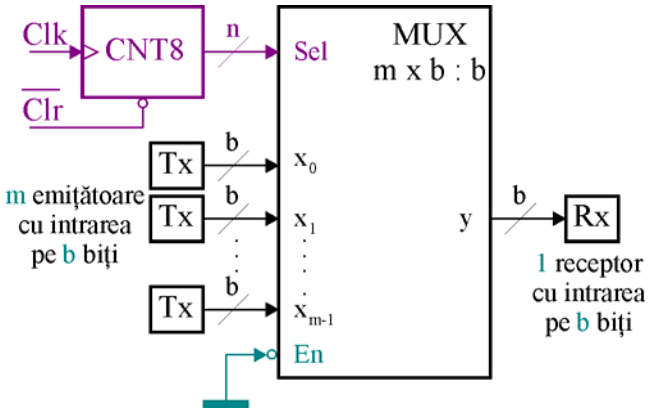


Figura 6.31. Multiplexor utilizat pentru transmisia succesivă a informației.

CNT este un numărător binar modulo *m*. Când este activată, intrarea nCLR determină ştergerea numărătorului. Aplicarea unui impuls de tact Clk determină incrementarea codului de la ieşirea CNT. Se selectează astfel succesiv cele $m = 2^n$ canale de date, iar informația prezentă la intrare este transferată succesiv la receptorul Rx.

2. **Conversia paralel-serie a unui cuvânt binar cu *m* biți**

Se foloseşte un MUX cu *m* canale de câte 1 bit. De exemplu pentru conversia paralel-serie a unui cuvânt binar se poate folosi MUX 74LS151. Cei 8 biți aplicați paralel la intrările de date, apar succesiv la ieşire, bit după bit. După 8 impulsuri de tact (CK) la ieşire se obține întregul cuvânt, în formă serială.

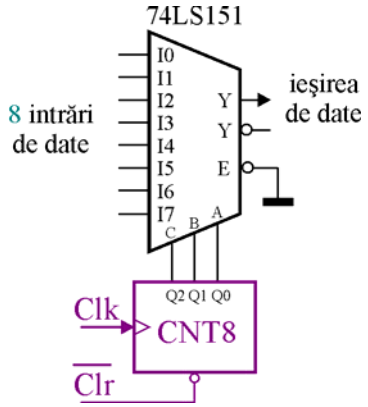


Figura 6.32. Conversia paralel – serie cu MUX 74LS151.

3. Implementarea funcţiilor logice

Spre deosebire de decodificator care permite teoretic implementarea unui număr ori cât de mare de funcţii în acelaşi timp, multiplexorul are o singură ieşire. Acest lucru permite implementarea doar a unei singure funcţii logice (respectiv a valorii negate a acesteia). Se utilizează în acest scop multiplexoare care au $b = 1$. Pot fi implementate funcţii cu un număr de variabile egale cu numărul de biţi ai codului de selecţie n. Implementarea se bazează pe relaţia care exprima variabila de ieşire Y în funcţie de codul de selecţie şi datele de intrare.

Exemplul 1. Fie $F = P_1 + P_3 + P_5 + P_6$. Se notează cu A, B, C intrările aferente variabilelor binare.

$$Y = En\sum_{i=0}^{7} I_i \cdot P_i = En \cdot (I_0 \cdot P_0 + I_1 \cdot P_1 + I_2 \cdot P_2 + I_3 \cdot P_3 + I_4 \cdot P_4 + I_5 \cdot P_5 + I_6 \cdot P_6 + I_7 \cdot P_7)$$

Pentru ca la ieşirea Y să se găsească funcţia F, se dau următoarele valori intrărilor: $I_1 = I_3 = I_5 = I_6 = 1$, $I_0 = I_2 = I_4 = I_7 = 0$

Dacă se doreşte utilizarea ieşirii $\overline{Y}$, se pun pe 0 intrările I_i care corespund termenilor P existenţi în funcţie şi pe 1 intrările I_i ce corespund termenilor P care lipsesc din funcţie. În exemplul de mai sus, pentru ca $\overline{Y} = F$ se dau următoarele valori intrărilor: $I_1 = I_3 = I_5 = I_6 = 0$, $I_0 = I_2 = I_4 = I_7 = 1$.

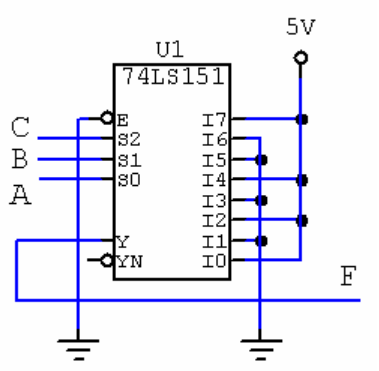


Figura 6.33. Implementarea unei funcţii cu MUX.

Exemplul 2. În afară de situaţia descrisă anterior, este de menţionat că este posibilă implementarea unei funcţii de $n + k$ variabile binare cu ajutorul unui multiplexor cu n biţi ai codului de selecţie, dacă numărul termenilor P din funcţia F nu depăşeşte numărul canalelor de intrări m.
Pentru 74LS151: codul de selecţie fiind pe 3 biţi, $k = 1$ - ceea ce corespunde la 4 variabile de intrare, iar numărul termenilor P trebuie să fie cel mult egal cu 8.

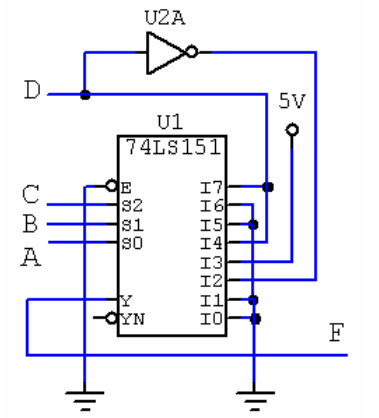


Figura 6.34. Implementarea unei funcţii cu multiplexor – varianta a II-a.

Fie $F = P_2 + P_3 + P_{11} + P_{12} + P_{15}$. Este o funcție de 4 variabile, dar implementarea se poate face cu un multiplexor având $n = 3$ deoarece numărul termenilor P este mai mic decât $m = 2^3 = 8$. Se rescrie funcția:

$$F = \overline{A}B\overline{C}\overline{D} + AB\overline{C}\overline{D} + AB\overline{C}D + \overline{A}B\overline{C}D + ABCD = \overline{D}.(P_2' + P_3') + D.(P_3' + P_4' + P_7') =$$
$$= P_2'.\overline{D} + P_3'(\overline{D} + D) + P_4'D + P_7'D = P_2' \cdot \overline{D} + P_3' + P_4' \cdot D + P_7' \cdot D$$

Dacă se foloseşte 74LS151 şi ieşirea Y, la intrările de date se aplică:
$I_0 = I_1 = I_5 = I_6 = 0$, $I_2 = \overline{D}$, $I_3 = 1$, $I_4 = D$, 0, $D_7 = D$, iar $\overline{En} = 0$ (figura 6.34).

6.4. CODIFICATORUL (CD)

CD furnizează la ieşire un cod de n biți corespunzător aceleia dintre cele m intrări ale sale care este activată (numărul de linii intrări este m, iar numărul biților codului de ieşire este n).
În situația în care fiecărei linii de intrări îi corespunde un cod distinct este valabilă relația: $n \geq \log_2 m$.
Exemplificarea structurii interne a unui codificator se face considerând codificarea binară a cifrelor zecimale 0,…, 9. În acest caz sunt necesare $m = 10$ intrări iar numărul de biți ai codului de ieşire este $n \geq \log_2 10 = 3{,}33$. Numărul de biți trebuie să fie un număr întreg şi deci $n \geq 4$. Reprezentarea simbolică a unui astfel de CD este:

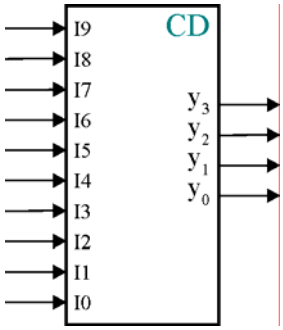


Figura 6.35. Schema bloc a unui codificator BCD.

4 biți sunt suficienți pentru codificarea a 16 intrări. 6 dintre codurile valorilor logice posibile nu se vor utiliza. Presupunem ca din cele 16 se aleg primele 10 coduri în ordine naturală crescătoare. Rezultă tabelul de funcționare 6.3.

Tabelul 6.3
Funcționarea codificatorului BCD

Linia activă	Y_3	Y_2	Y_1	Y_0
I_0	0	0	0	0
I_1	0	0	0	1
I_2	0	0	1	0
I_3	0	0	1	1
I_4	0	1	0	0
I_5	0	1	0	1
I_6	0	1	1	0
I_7	0	1	1	1
I_8	1	0	0	0
I_9	1	0	0	1

Funcțiile binare ce corespund celor 4 ieşiri sunt:

$$Y_3 = I_8 + I_9$$
$$Y_2 = I_4 + I_5 + I_6 + I_7$$
$$Y_1 = I_2 + I_3 + I_6 + I_7$$
$$Y_0 = I_1 + I_3 + I_5 + I_7 + I_9$$

În aceste funcții nu intervine I_0 - dacă intrările $I_1,....I_9$ sunt inactive (0), codul de ieşire trebuie să fie 0.

Dezavantajul principal al codificatoarelor (denumite *neprioritare*) este că nu funcţionează corect în situaţii în care se activează simultan două sau mai multe intrări. Dacă se activează de exemplu simultan intrările I_6 şi I_9, atunci codul de ieşire este 1 1 1 1.
CD se pot utiliza în aplicaţii în care nu sunt activate simultan două sau mai multe intrări. Codificatoarele *nu* se fabrică ca şi circuite integrate distincte, ele fac parte din circuite mai complexe.

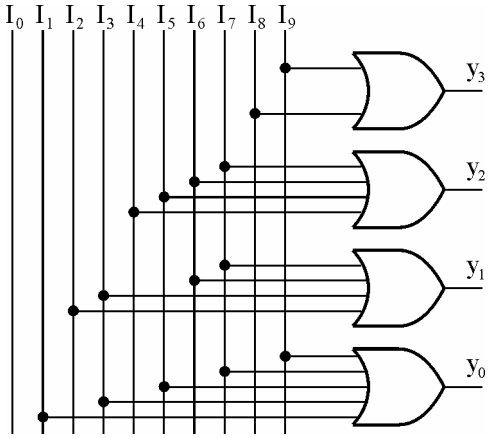


Figura 6.36. Schema electrică a codificatorului BCD.

6.4.1. Codificatoare prioritare

Codificatoarele prioritare (CDP):

- Înlătură dezavantajele CD (neprioritare)
- CDP se fabrică şi sub forma unor CI distincte, dar pot fi integrate ca subcircuite.
- În cazul activării simultane a două sau mai multe intrări furnizează la ieşire codul corespunzător intrării cu cea mai mare prioritate dintre cele activate.

Codificatoarele prioritare asigură atribuirea unor priorităţi intrărilor. Uzual intrarea cu indice mai mare este prioritară faţă de intrările cu indicele mai mic. În cazul activării simultane a două sau mai multe intrări, codul de ieşire va corespunde intrării cu prioritate maximă.

În cazul CDP prioritatea scade cu scăderea indicelui intrării. Gradul de prioritate al intrării se stabileşte prin structura circuitului integrat. Reprezentarea simbolică pentru CDP cu $m = 8$, $n = 3$ biţi: EI validează funcţionarea circuitului. EO este utilizat pentru validarea intrării EI a unui circuit similar cu intrări având prioritate imediat inferioară (atunci când se doreşte extinderea numărului de intrări, de exemplu de la 8 la 16).

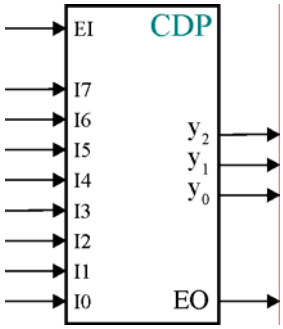


Figura 6.37. Schema bloc a codificatorului prioritar.

EI – *Enable Input* - validează circuitul.
EO – *Enable Output* (ieşire de validare), care este activă când CDP este validat (EI = 1) şi când nici una dintre intrările I_0, I_1, ...,I_7 nu este activată

EO este utilizat pentru validarea unui alt circuit similar cu acesta, cu grad de prioritate mai mic, în cazul în care nu este activată nici o intrare I_0, I_1, ...,I_7. Circuitul validat corespunde unor intrări cu prioritate inferioara lui I_0. $EO = EI.(\overline{I_7}.\overline{I_6}...\overline{I_0})$

Structura unui CDP: considerăm un CDP cu 8 intrări şi 3 ieşiri. Prima etapă o constituie reprezentarea tabelului de funcționare pentru un codificator neprioritar cu 8 intrări şi un cod de ieşire pe 3 biți.

Tabelul 6.x

Funcționarea codificatorului neprioritar

Intrare activă	**Ieşiri**		
	Y'_2	**Y'_1**	**Y'_0**
I_7	1	1	1
I_6	1	1	0
I_5	1	0	1
I_4	1	0	0
I_3	0	1	1
I_2	0	1	0
I_1	0	0	1
I_0	0	0	0

$$Y_2^{'} = I_7 + I_6 + I_5 + I_4$$
$$Y_1^{'} = I_7 + I_6 + I_3 + I_2$$
$$Y_0^{'} = I_7 + I_5 + I_3 + I_1$$

Pentru a obține un CDP fiecărei intrări i se atribuie o anumită prioritate prin intermediul unei variabile intermediare Z. Folosind această substituire, funcțiile de ieşire pentru CDP sunt:

$$\begin{aligned} Y_2^{'} &= Z_7 + Z_6 + Z_5 + Z_4 \\ Y_1^{'} &= Z_7 + Z_6 + Z_3 + Z_2 \\ Y_0^{'} &= Z_7 + Z_5 + Z_3 + Z_1 \end{aligned} \qquad (6.1)$$

$Z_7 = I_7$ – corespunde celei mai prioritare intrări;

$Z_6 = \overline{I_7}.I_6$ – dacă I_7 nu este activată, I_6 rămâne cea mai prioritară intrare;

$Z_5 = \overline{I_7}.\overline{I_6}.I_5$ – dacă I_7 şi I_6 nu sunt activate, I_5 rămâne cea mai prioritară intrare;

…

Înlocuind în (6.1) pe Z se obțin funcțiile $Y = f(Z)$, care apoi se minimizează.

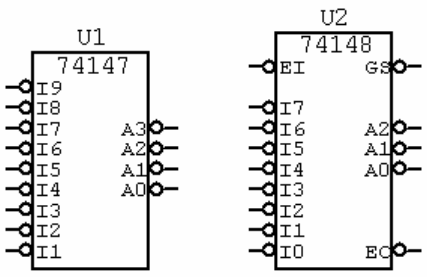


Figura 6.38. Codificatoare prioritare TTL (74LS147 - BCD, 74LS148 - octal).

Exemplu de CDP frecvent utilizat: **74LS148**

- toate intrările şi ieşirile sunt active pe 0
- există o intrare de validare nEI, activă pe 0

- codul de ieşire este pe 3 biţi
- nEO va fi activă (pe 0) dacă circuitul este validat şi nici una dintre intrările $I_0,\ldots,I_7$ nu este activată

$$EO = EI.(\overline{I_7}.\overline{I_6}...\overline{I_0})$$

- GS (group select) – selecţie de grup. Aceasta este activă dacă circuitul este validat şi cel puţin una dintre intrările circuitului este activă

$$GS = EI \cdot (I_7 + I_6 + ... + I_0)$$

6.4.2. Extinderea numărului de intrări la CDP

Dacă se doreşte un CDP cu 16 intrări, folosim două CDP74LS148.

a.) Circuitele 1 şi 2 au intrări active pe 0. Dacă circuitul 1 are cel puţin o intrare activă $(\overline{EO_1} = 1)$, atunci circuitul 2 nu este validat. Codul de ieşire va corespunde intrării activate celei mai prioritare a circuitului 1. GS va fi 1. (de exemplu când cea mai prioritară intrare este I_{10} se obţine la ieşire codul $Y_3Y_2Y_1Y_0$ – 1010, Y_3 = 1 deoarece EO_1= 0).

b.) Dacă circuitul 1 nu are nici o intrare activă $(\overline{EO_1} = 0)$, atunci circuitul 2 este validat. Dacă una din intrările circuitului 2 este activă, atunci GS=1 (dacă de ex. nici una din intrările $I_{15},...,I_8$ nu este activă Y_3=0, Y_2, Y_1, Y_0 corespund intrării celei mai prioritare a circuitului 2, de exemplu I_5 : $Y_3Y_2Y_1Y_0$ – 0101

c.) Nici o intrare nu este activă. În această situaţie ambele circuite sunt validate, dar neavând nici o intrare activă, codul de ieşire este $Y_3Y_2Y_1Y_0$ – 0 0 0 0, iar GS=0.

Principala aplicaţie a unui astfel de circuit îl constituie arbitrarea întreruperilor într-un microsistem. În funcţionarea unui microsistem are loc prelucrarea informaţiei într-o succesiune stabilită într-un program principal. Microsistemul este interconectat cu periferice. Programul principal poate fi întrerupt printr-o solicitare din partea unui periferic. Solicitarea de întrerupere pentru satisfacerea unei solicitări a perifericului are loc astfel: perifericul pune pe 0 linia de intrare care-i corespunde; se activează *GS* trecând pe 1, atenţionând microsistemul că a fost cerută o întrerupere. Microsistemul termină secţiunea în lucru din programul principal şi trece la deservirea întreruperii. El citeşte codul de ieşire al CDP, cod care determină pentru fiecare periferic adresa subrutinei de deservire a perifericului. După terminarea acestei subrutine, microsistemul revine la programul principal. Dacă mai sunt şi alte cereri, microsistemul le deserveşte în ordinea priorităţii, până când *GS* = 0.

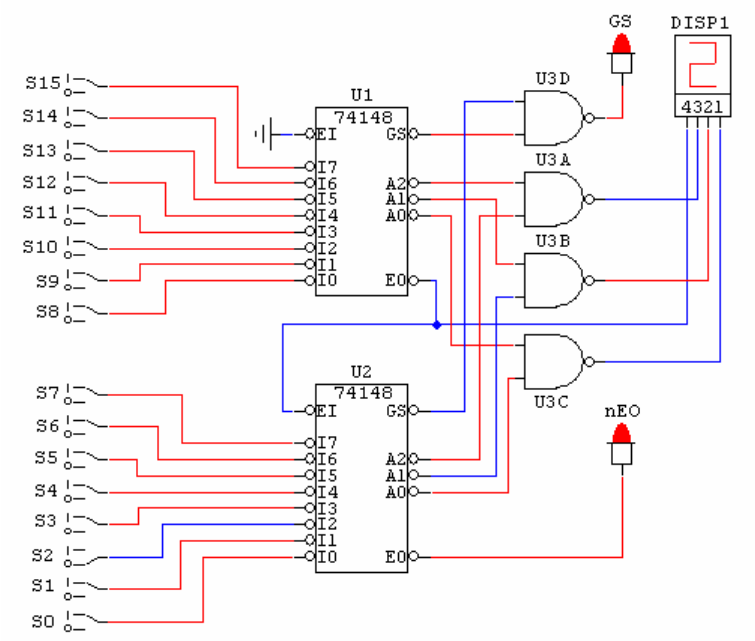


Figura 6.39. Extinderea capacității de codificare – activarea intrării 2.

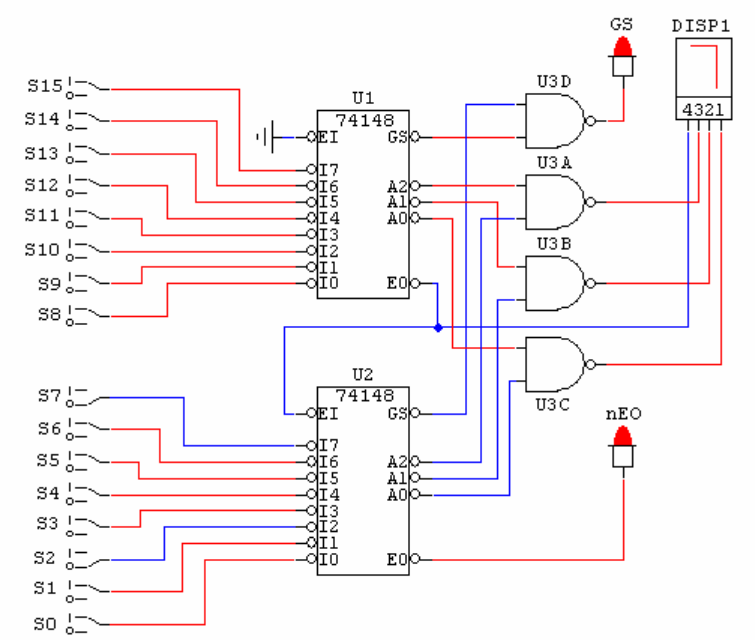


Figura 6.40. Intrările 2 şi 7 activate simultan.

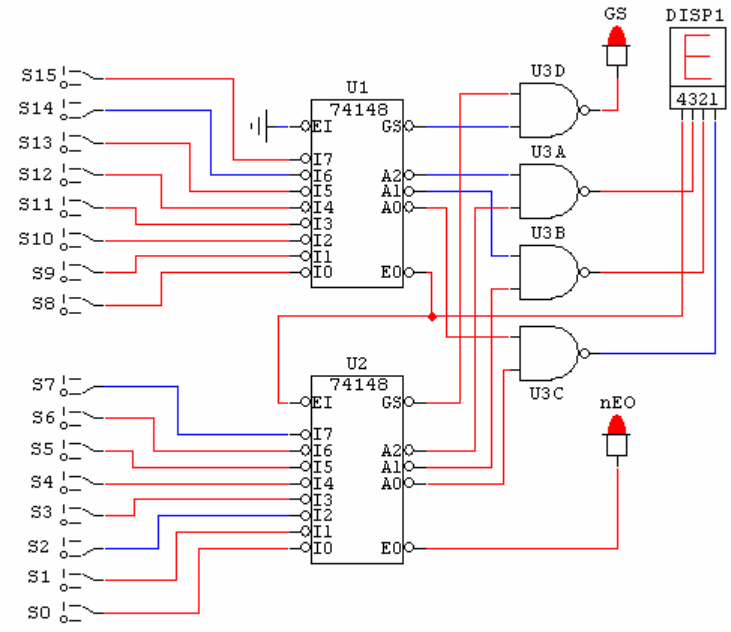


Figura 6.41. Intrările 2, 7 şi 14 activate simultan.

6.5. COMPARATORUL NUMERIC (CN)

Are funcția de a stabili valoarea relativă a două numere binare, A şi B, care au acelaşi număr de biți. Dacă numărul de biți este *b,* CN are *2b* intrări şi în general, trei ieşiri:

- $F_e = 1 \Leftrightarrow A = B$ (*egal*);
- $F_s = 1 \Leftrightarrow A > B$ (*superior*);
- $F_i = 1 \Leftrightarrow A < B$ (*inferior*).

Observație: În unele circuite există doar F_e şi F_s, iar F_i se deduce.
Pentru a analiza structura unui comparator se are în vedere comparatorul elementar pentru doi biți a_k, b_k, (rangul k al numerelor A şi B). Un comparator pentru un număr de b biți se compune din b comparatoare elementare pentru numere de câte un bit (acelaşi bit pentru A şi B) şi din alte circuite combinaționale auxiliare.

6.5.1. Comparatoare elementare

Pentru a determina egalitatea dintre a_k şi b_k se scrie relația: $f_{ek} = \overline{a_k \cdot \overline{b_k} + \overline{a_k} \cdot b_k} = a_k \cdot b_k + \overline{a_k} \cdot \overline{b_k}$.

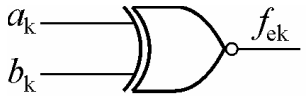


Figura 6.42. Comparator elementar pe 1 bit.

Pentru a obține f_{sk} si f_{ik} se foloseşte câte un circuit ŞI cu două intrări, una din ele fiind complementată.

Tabelul 6.1

Definirea funcțiilor f_{ek}, f_{sk} şi f_{ik}

a_k	b_k	f_{ek}	f_{sk}	f_{ik}
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

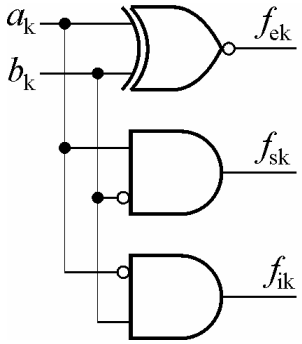


Figura 6.43. Obținerea funcțiilor f_{ek}, f_{sk} şi f_{ik}.

6.5.2. Comparatoare pe 4 biți

Fie numerele A şi B, reprezentate pe 4 biți: $A \rightarrow [A_0, A_1, A_2, A_3]$ şi $B \rightarrow [B_0, B_1, B_2, B_3]$.

a) Condiția de egalitate între A şi B este îndeplinită atunci când toți biții de acelaşi rang din A şi B au valori egale. În cazul discutat pentru 4 biți, această condiție se scrie:

$$F_e = f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{e0}$$

b) Condiția de superioritate între 2 numere binare A şi B ($A > B$), notată F_s se scrie astfel:

- $A > B$ dacă $a_3 > b_3$ SAU ($a_3 = b_3$ ŞI $a_2 > b_2$) SAU ($a_3 = b_3$ ŞI $a_2 = b_2$ ŞI $a_1 > b_1$) SAU ($a_3 = b_3$ ŞI $a_2 = b_2$ ŞI $a_1 = b_1$ ŞI $a_0 > b_0$), adică:

$$F_s = f_{s3} + f_{e3} \cdot f_{s2} + f_{e3} \cdot f_{e2} \cdot f_{s1} + f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{s0}$$

c) Similar $F_i = f_{i3} + f_{e3} \cdot f_{i2} + f_{e3} \cdot f_{e2} \cdot f_{i1} + f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{i0}$

Dintre valorile F_e, F_s, F_i numai una este adevărată la un moment dat, iar $F_e = \overline{F_s} \cdot \overline{F_i}$, $F_s = \overline{F_e} \cdot \overline{F_i}$, $F_i = \overline{F_s} \cdot \overline{F_e}$. Comparatorul se poate realiza în consecinţă şi cu două ieşiri F_e, F_s, iar $F_i = \overline{F_e + F_s}$. Evident F_i necesită un circuit combinaţional suplimentar, ceea ce implică o diferenţă temporală între apariţia F_e, F_s pe de o parte şi F_i pe de altă parte. Dacă acest defazaj este deranjant, o soluţie simplă este întârzierea cu un circuit neinversor (de exemplu o poartă ŞI) a ieşirilor F_e şi F_s.

Implementarea lui F_e: Dacă se realizează compararea pentru biţii 0...3, $F_e^{'} = 1$.

$$F_e = f_{e0} \cdot f_{e1} \cdot f_{e2} \cdot f_{e3} \cdot F_e^{'}$$

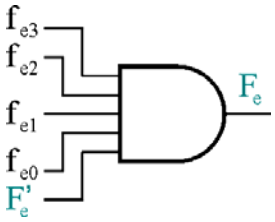


Figura 6.44. Obţinerea funcţiei F_e.

Implementarea lui F_s: Dacă se compară biţii 4...7, se face conectarea: $F_s^{'}$ la F_s a circuitului anterior. Dacă se compară biţii 0...3, $F_s^{'}$ se leagă la 0 (similar $F_i^{'}$ se leagă la masă).

$$F_s = f_{s3} + f_{e3} \cdot f_{s2} + f_{e3} \cdot f_{e2} \cdot f_{s1} + f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{s0} + f_{e3} \cdot f_{e2} \cdot f_{e1} \cdot f_{e0} \cdot F_s^{'}$$

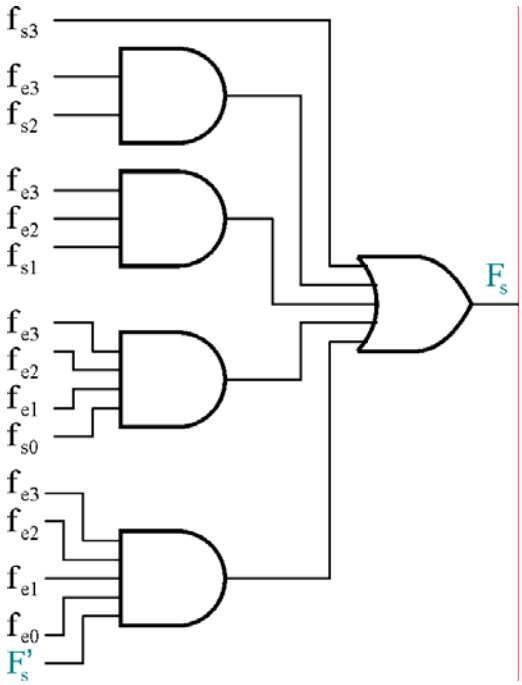


Figura 6.45. Obţinerea funcţiei F_s.

6.5.3. Exemplu de comparator TTL

74LS85 este un comparator pentru două numere binare de câte 4 biți. Are 3 intrări de interconectare $F_e^{'}, F_s^{'}, F_i^{'}$, destinate unui alt comparator cu semnificație imediat inferioară.

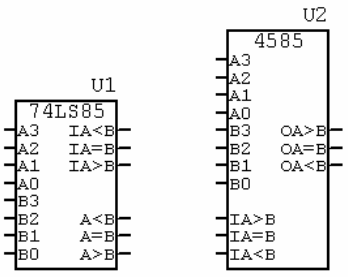


Figura 6.46. Comparator integrat pe 4 biți, 74LS85.

6.5.4. Extinderea capacității de comparare

Comparatorul 1 are influență asupra deciziei comparatorului 2, doar dacă simultan $A_4 = B_4$, $A_5 = B_5$, $A_6 = B_6$, $A_7 = B_7$.

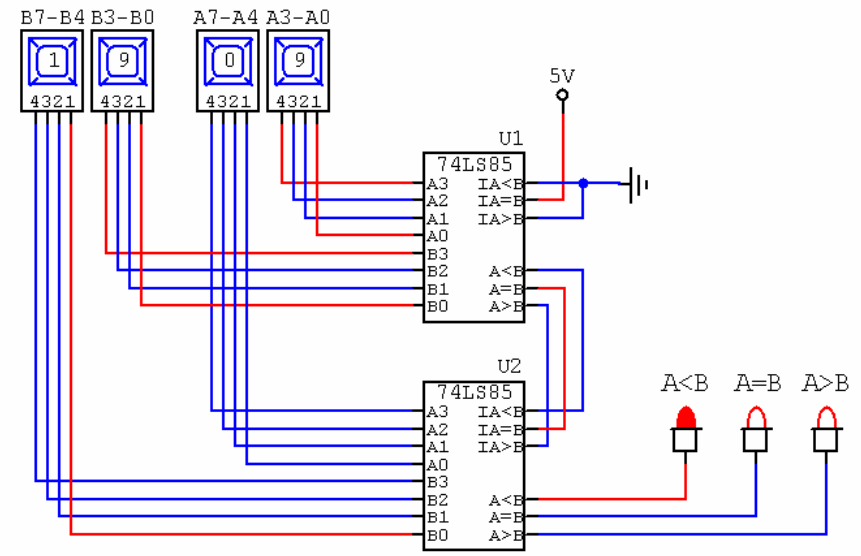


Figura 6.47. Comparator TTL pe 8 biți.

6.5.5. Aplicație 74LS85 – diagrame de semnal pentru comparatorul pe 8 biți

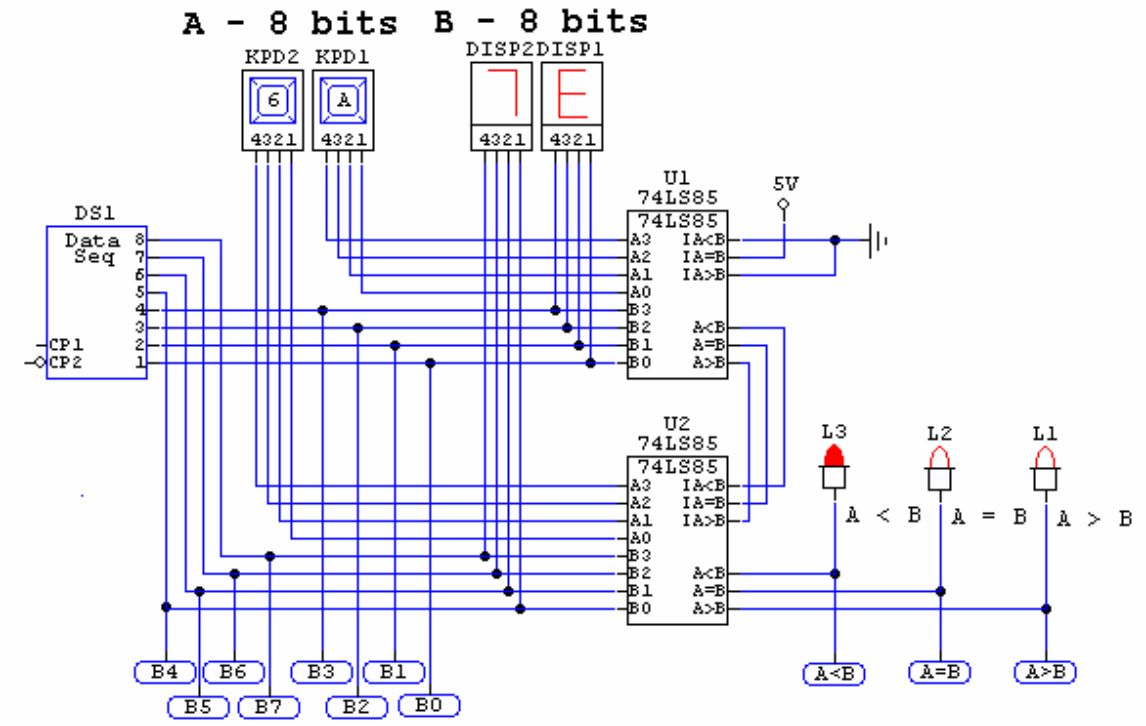


Figura 6.48. Funcționarea comparatorului pe 8 biți.

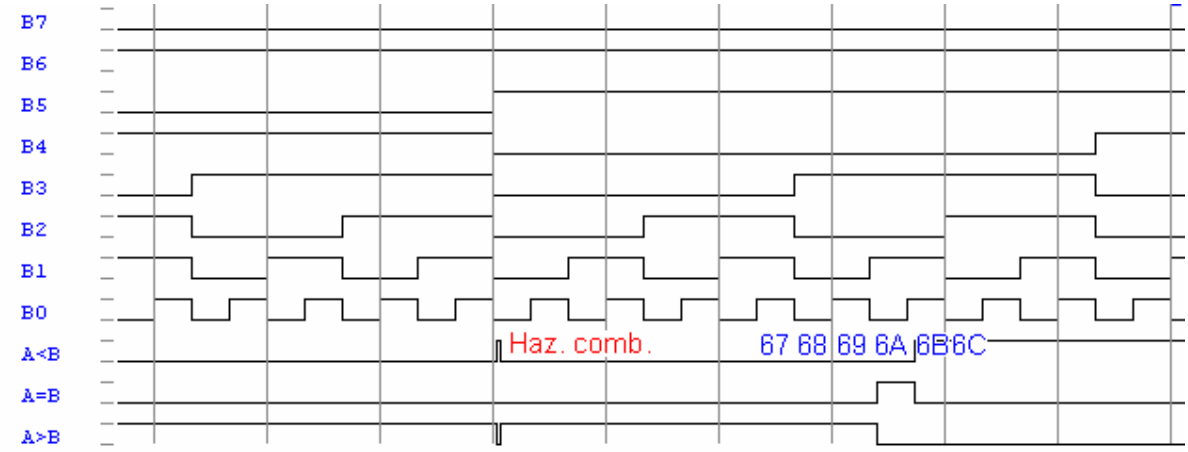


Figura 6.49. Diagrame de semnal pentru comparatorul din figura 6.48.

6.5.6. Comparator MSI pe 8 biți

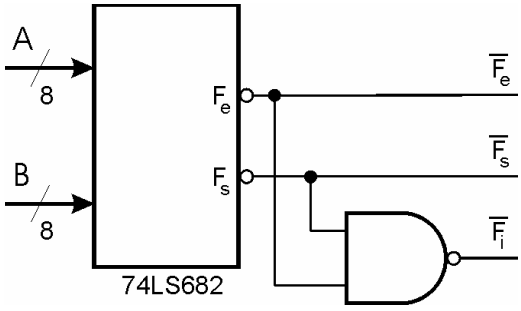


Figura 6.50. Comparator complet pe 8 biți.

74LS682 are 2 x 8 intrări active pe 1 şi două ieşiri $\overline{F_e}, \overline{F_s}$ active pe 0.

6.5.7. Temă

Pentru un comparator de tip 74LS85 la care $F'_e = 1$, să se completeze în diagrama de semnal de mai jos variația F_e, F_i, F_s.

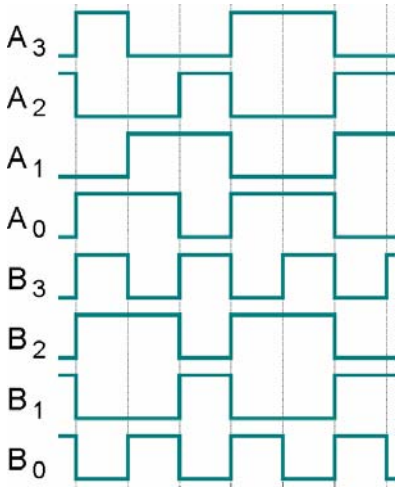


Figura 6.51. Diagrame de semnal.

6.6. DETECTORUL ŞI GENERATORUL DE PARITATE SAU IMPARITATE

Este utilizat pentru detectarea erorilor de transmisie a informației binare. Funcția este un circuit logic combinațional care determina paritatea sau imparitatea numărului de variabile de intrare egal cu 1, generând un bit de paritate sau imparitate. Un astfel de detector se bazează pe detectoare elementare de imparitate cu două intrări (circuit SAU-EXCLUSIV).

Tabelul 6.x

Funcționarea porții SAU-EXCLUSIV ca generator de imparitate

I_1	I_2	IMP
0	0	0
0	1	1
1	0	1
1	1	0

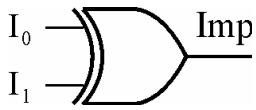


Figura 6.52. Generator de imparitate din poarta SAU-EXCLUSIV.

Structura detectorului poate fi în lanț sau arborescentă.

a. **Structura în lanț**. La aceasta structura trebuie să ținem seama de:
- pentru n intrări sunt necesare n-1 circuite XOR
- timpul de propagare pe traseul critic: $t_p = (n - 1)\, t_{pXOR}$
- numărul de intrări n poate fi un număr par, cât şi impar

b. **Structura arborescentă**. La această structură trebuie să ținem seama de:
- pentru n intrări sunt necesare n - 1 circuite XOR;
- timpul de propagare $t_p = (\log_2 n)\, t_{pXOR}$ este mai mic decât la structura în lanț;
- numărul de intrări n trebuie sa fie un număr par.

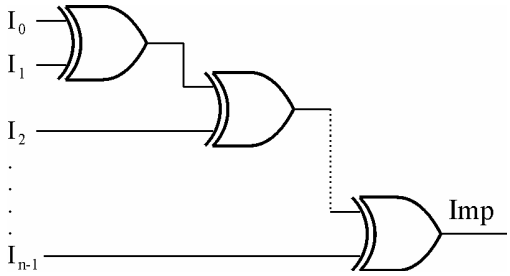


Figura 6.53. Generator de imparitate cu structură în lanț.

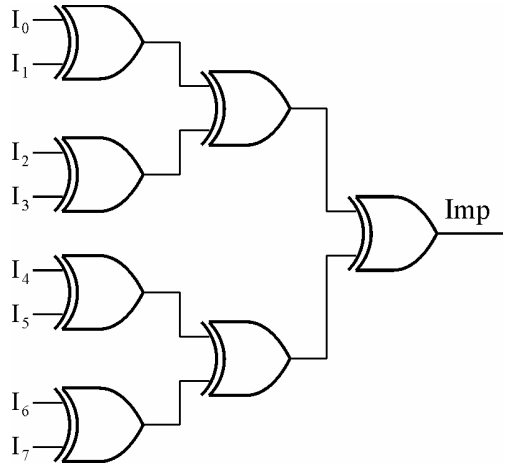


Figura 6.54. Generator de imparitate cu structură arborescentă.

Orice detector de imparitate se poate transforma intr-unul de paritate prin folosirea unui inversor suplimentar. Astfel de circuite permit utilizatorului, în funcție de aplicație, să aleagă funcția îndeplinită, stabilind printr-un bit dacă circuitul funcționează ca un detector de paritate sau imparitate.

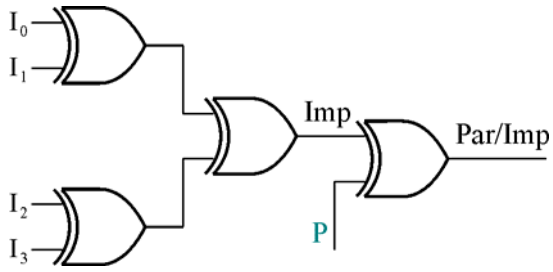


Figura 6.55. Generator de paritate / imparitate selectabil.
(P = 1⇒ inversor P = 0⇒ neinversor)

În prezent se folosesc:

74HC180

- are 8 intrări; 2 ieşiri PAR şi IMPAR; 2 intrări de interconectare;
- structura arborescentă;
- se foloseşte pentru detectarea erorilor de transmisie.

74LS280

- are 9 intrări; 2 ieşiri PAR şi IMPAR
- structura in lanț
- este folosit pentru detectarea erorilor de memorare ale unui cuvânt binar cu 8 biți. Verifică dacă informația citită din memorie are aceeaşi paritate ca şi cea înscrisă. În afară de cei 8 biți memoria trebuie să asigure şi memoria de paritate.

0	1	2	3	4	5	6	7	P

În cazul unei linii de transmisie exista câmpuri electromagnetice care pot să modifice informația trimisă de la sursă.

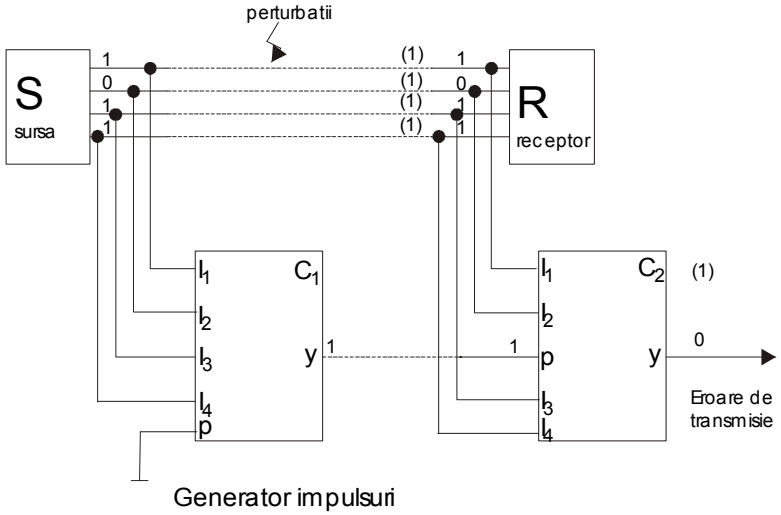


Figura 6.56. Sistem de transmisie cu semnalizarea parității.

6.7. SUMATORUL

Funcția: Efectuarea de operații aritmetice (adunare sau scădere) cu două numere binare având un număr egal de biți. Orice sumator pe mai mulți biți este construit din sumatoare elementare pe un bit. Sumatoarele elementare pe un bit pot fi:

- *semisumatoare* (sumator pentru bitul zero), acest sumator elementar se caracterizează prin faptul că nu ține seama de transportul de la bitul cu semnificație imediat inferioară.
- *sumatoare complete pe un bit* care țin seama de transportul de la bitul cu semnificație imediat inferioară.

6.7.1. Semisumatorul (sumatorul pentru bitul zero)

- intrările celor două numere pe un bit sunt reprezentate prin X_0 si Y_0;

- ieşirile sunt: S_0 - (suma celor două numere) şi C_1 - (Carry - transportul către bitul 1).

$$S_0 = \overline{X}_0 \cdot Y_0 + X_0 \cdot \overline{Y}_0 = X_0 \oplus Y_0, \quad \text{iar } C_1 = X_0 \cdot Y_0.$$

Tabelul 6.x

Funcționarea semisumatorului

X_0	Y_0	C_1	S_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

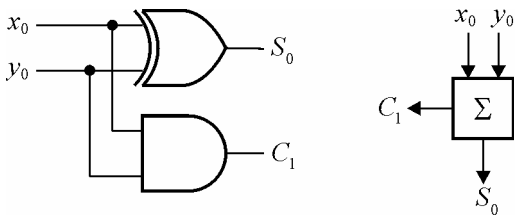


Figura 6.57. Semisumator pe un bit.

6.7.2. Sumatorul complet pe un bit

Sumatorul complet pe un bit ține cont de transportul de la bitul de semnificație imediat inferioară. Are intrările: X_n, Y_n, C_n şi ieşirile: S_n, C_{n+1}. Funcționarea sa se bazează pe tabelul de mai jos.

Din tabel se deduc relațiile care descriu dependența ieşirilor de intrări:

$$S_n = \overline{X}_n \overline{Y}_n C_n + \overline{X}_n Y_n \overline{C}_n + X_n \overline{Y}_n \overline{C}_n + X_n Y_n C_n = C_n \left(\overline{X}_n \overline{Y}_n + X_n Y_n\right) + \overline{C}_n \left(\overline{X}_n Y_n + X_n \overline{Y}_n\right) =$$

$$= C_n \cdot \overline{X_n \oplus Y_n} + \overline{C}_n \cdot X_n \oplus Y_n = C_n \oplus X_n \oplus Y_n$$

$$C_{n+1} = \overline{X}_n Y_n C_n + X_n \overline{Y}_n C_n + X_n Y_n \overline{C}_n + X_n Y_n C_n = X_n Y_n \left(\overline{C}_n + C_n\right) + C_n \left(\overline{X}_n Y_n + X_n \overline{Y}_n\right) =$$

$$= X_n Y_n + C_n \left(X_n \oplus Y_n\right)$$

Prin implementarea relațiilor obținute anterior, se obține următoarea schemă pentru un sumator complet de 1 bit. Dacă se determină timpul de propagare de la intrări la ieşiri se constată că:

$$t_{P_S} = 2 \cdot t_{p_{XCR}} = 2 \cdot 3 \cdot t_{p_{Si-NU}} \qquad \text{deoarece} \qquad t_{p_{XCR}} = 3 \cdot t_{p_{Si-NU}}$$

$$t_{p_C} = t_{p_{XCR}} + t_{p_{Si}} + t_{p_{SAU}} = 6 \cdot t_{p_{Si-NU}} \qquad t_{p_{Si}} = t_{p_{SAU}} = 1{,}5 \cdot t_{p_{Si-NU}}$$

Dacă, pentru obținerea ieşirii de transport, se foloseşte schema din dreapta, timpul de propagare se reduce la: $t_{p_C} = 5 \cdot t_{p_{Si-NU}}$

Tabelul 6.x

Funcționarea sumatorului complet

X_n	Y_n	C_n	C_{n+1}	S_n
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

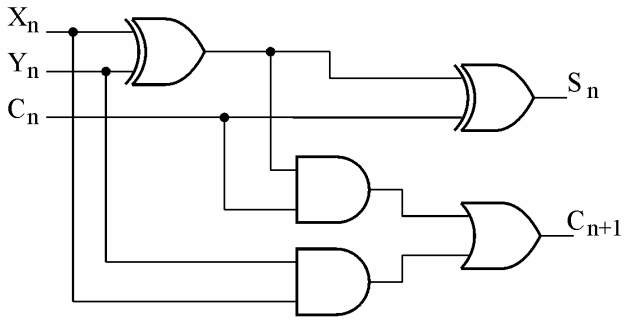


Figura 6.58. Sumator complet pe 1 bit, varianta 1.

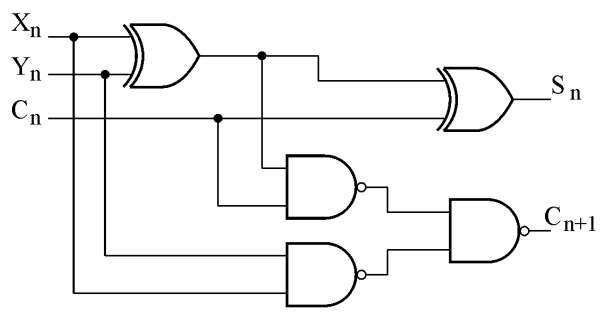
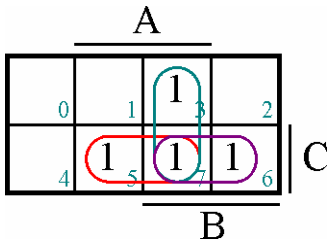


Figura 6.59. Sumator complet pe 1 bit, varianta 2.

Se poate reduce în continuare, t_{p_C} prin minimizarea relației lui C_{n+1} cu ajutorul diagramei VK. Relația de definirea a lui C_{n+1} este:

$$C_{n+1} = \overline{X}_n Y_n C_n + X_n \overline{Y}_n C_n + X_n Y_n \overline{C}_n + X_n Y_n C_n$$

Dacă se consideră $X_n = A = 2^0$, $Y_n = B = 2^1$, $C_n = C = 2^2$, atunci diagrama VK este:



Se obține: $C_{n+1} = X_n Y_n + Y_n C_n + X_n C_n$, care conduce la următoarea schemă pentru un sumator complet pe un bit:

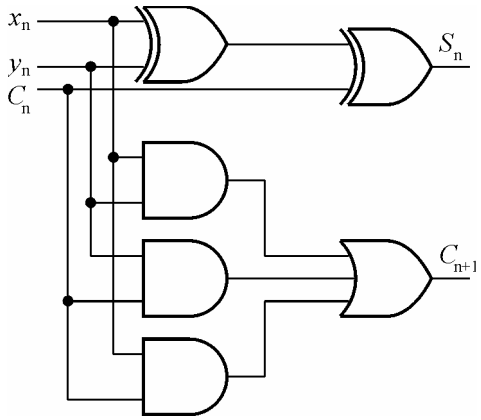


Figura 6.60. Sumator complet pe un bit, varianta 3.

În acest caz timpul de propagare de la orice intrare la ieşirea de transport este: $t_{p_C} = 3 \cdot t_{p_{Si-NU}}$

6.7.3. Sumator cu transport succesiv 74LS83 (4 biți)

Schema acestui sumator pe patru biți cuprinde patru sumatoare complete pe un bit interconectate ca în figură:

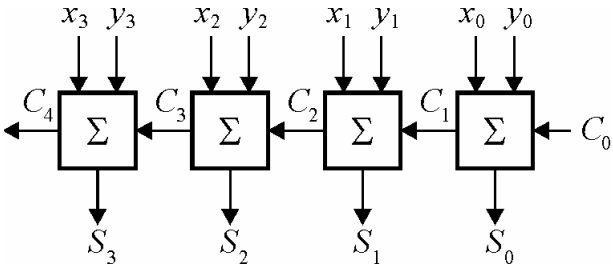


Figura 6.61. 74LS83 – schema funcțională.

C_0 - Se pune la masă dacă circuitul este folosit pentru însumarea a două numere cu 4 biți, deoarece nu există transport de la un bit cu semnificație mai mică. Când se extinde numărul de biți folosind două sau mai multe circuite conectate se face concordanta cu următoarea schemă:

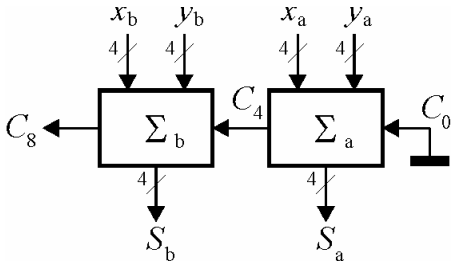


Figura 6.62. Extinderea capacității de adunare.

Un astfel de sumator furnizează rezultatul final după un timp ce corespunde generării transportului C_n. Dacă se consideră la $t = 0 \mid_y^x$ valorile care vor apare pentru sume şi Carry nu sunt cele finale, este necesar ca să se compună timpii de întârziere cu care sunt generate transporturile C_1, C_2, C_3, C_n, numai după această întârziere suma S şi transportul C_n sunt corecte (transportul C_4 apare cu o întârziere de $4t_{p_C}$).

Un astfel de sumator se numeşte *sumator succesiv* (dacă suntem în cazul cel mai defavorabil fiecare sumator de un bit generează un transport "1"

x 1 1 1 1
y 0 0 0 1
1 0 0 0 0

cu cât folosim mai multe sumatoare cu atât t_p e mai mare). Pentru a obține viteze mari e necesar ca întârzierile să fie cât mai mici.

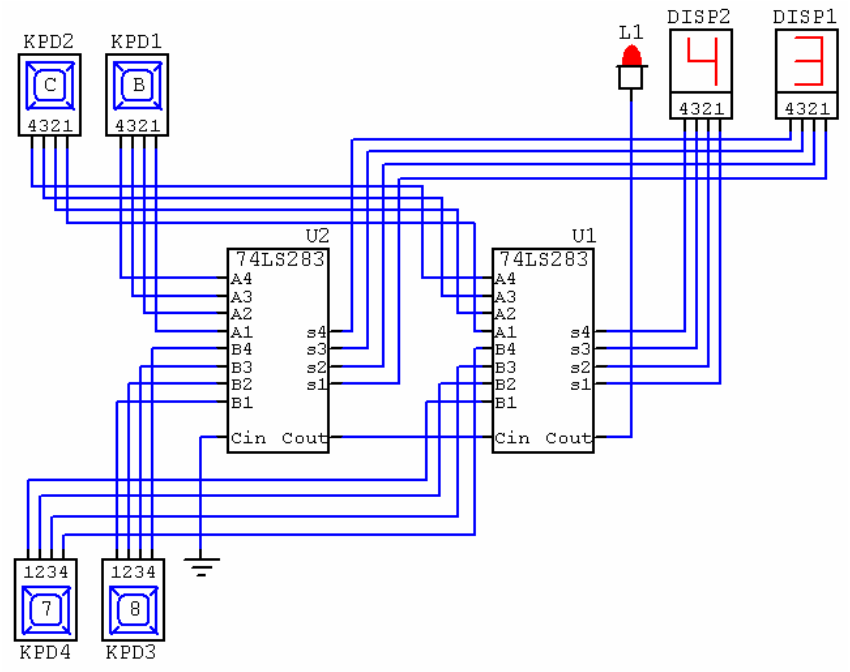


Figura 6.62. Extinderea sumatoarelor.

6.7.4. Aplicație. Sumator folosit pentru scădere

Este necesară complementarea biților scăzătorului: Intrarea de transport si ieşirea de transport sunt interpretate ca intrare de împrumut. $\overline{B}_n, \overline{B}_{n+1}$. In cazul sumatorului 83 avem nevoie de patru astfel de inversoare; C_0 se leagă la "1" iar C_n se considera B_n.

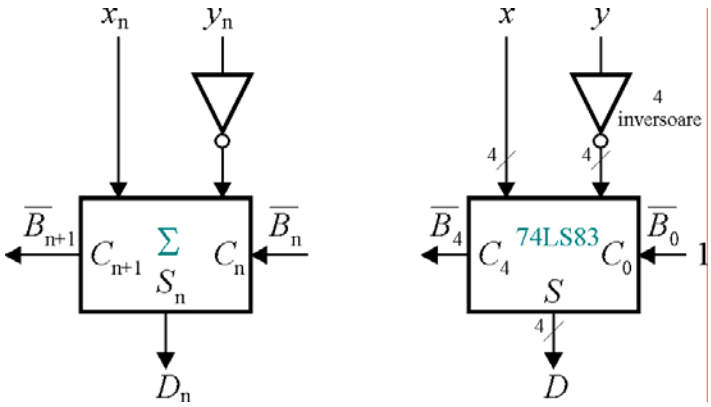


Figura 6.63. Sumator folosit pentru scădere.

6.7.5. Aplicație. Sistem simplu de votare.

Un sumator de tipul 74LS83 adună numere binare în care fiecare bit are o anumită *pondere*. Pentru a aduna biți de *aceeaşi pondere*, de exemplu într-un sistem de votare este necesară utilizarea mai multor sumatoare pe un bit, cascadate, ca în figură. U1 şi U2 vor aduna fiecare câte trei biți de pondere egală, conectați la intrările A1, B1 şi Cin. Cele două sume parțiale astfel obținute sunt folosite pentru calcularea sumei finale, cu ajutorul lui U3. Rezultatul final este afişat pe DISP1.

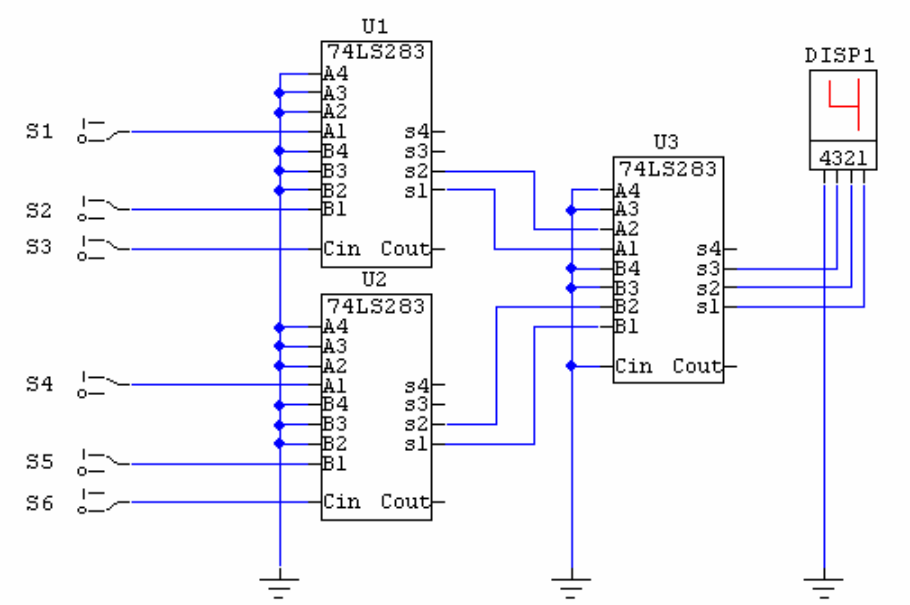


Figura 6.64. Sistem simplu de votare.

6.8. CONVERTOARE DE COD

6.8.1. Convertor de cod binar - cod Gray

6.8.2. Convertor de cod Gray - binar

6.8.3. Convertor de cod 7 segmente - binar