

Laboratorul 03 - Simularea circuitelor digitale

Simularea circuitelor digitale

Pentru simularea circuitelor vom folosi tot simulatorul din laboratorul 1 și 2, <http://www.falstad.com/circuit> [<http://www.falstad.com/circuit>].

Stări logice

Prin circuite digitale înțelegem circuite în care sunt numai două stări posibile (de exemplu: un tranzistor poate fi în saturație, sau nu conduce). Numim aceste stări nivele logice și le notăm prin HIGH sau LOW. Ele pot însemna mai multe lucruri:

- Dacă un switch este deschis sau închis
- Dacă un semnal este prezent sau absent
- Dacă un nivel analog este sub sau peste o anumită limită prestabilită

HIGH și LOW

Stările HIGH și LOW reprezintă stările TRUE și FALSE din logica booleană. Dacă într-un anumit punct HIGH este definit ca TRUE, atunci avem logică pozitivă. Dacă HIGH este definit ca FALSE, atunci avem logică negativă. De exemplu faptul că un switch este închis este adevărat când outputul sau este LOW.








În circuitele digitale, nivelele logice HIGH și LOW reprezintă anumite intervale (ex: HIGH:3.5-5V, LOW:0-2.5V; între 2.5V și 3.5V nivelul fiind nedefinit).

Imunitatea la zgomot

Imunitatea la zgomot reprezintă nivelul maxim de zgomot ce poate fi adăugat la nivelele logice fără ca poarta logică să funcționeze greșit.

Porți Logice

Scopul unui circuit digital este de a prelucra intrări digitale pentru a produce ieșiri digitale. Pentru a realiza acest lucru, se folosesc porți logice. Folosind aceste porți, se pot crea circuite mai complexe.

Tip	Simbol	Funcție
AND		$A \cdot B$
OR		$A + B$
NOT		\overline{A}
NAND		$\overline{A \cdot B}$
NOR		$\overline{A + B}$
XOR		$A \oplus B$
XNOR		$\overline{A \oplus B}$

RTL

- folosesc pe post de rețea de intrare rezistențe și pentru logica de comutare folosesc tranzistori.
- este prima familie de circuite logice digitale cu tranzistori.
- erau foarte utile în perioada în care s-au inventat pentru că foloseau un număr minim de tranzistoare, iar în acea perioadă erau cele mai scumpe componente de produs
- necesită mai mult curent la intrare
- disipă multă căldură

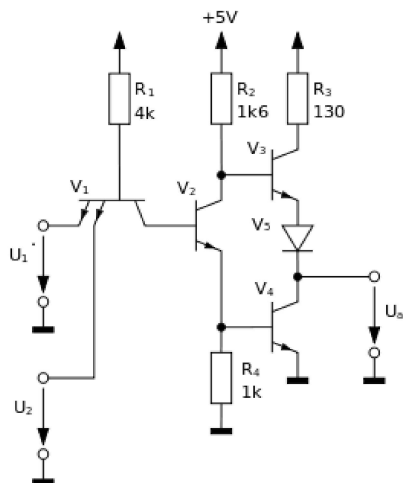
DTL

- folosește diodele pentru a simula funcția logică și folosește tranzistoare pentru amplificare.
- este tehnologia ce precedează TTL.

- are ca dezavantaj principal timpul de propagare.

TTL

- folosește tranzistoare atât pentru implementare funcției logice, cât și pentru amplificare.
- etajul de ieșire poate avea și un "totem-pole" în loc de un simplu tranzistor.



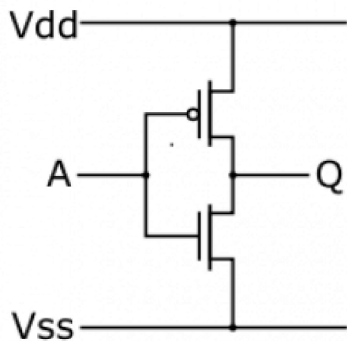
- subtipuri:
 - LPTTL - Low-power TTL
 - HSTTL - High-speed TTL
 - STTL - Schottky TTL
 - LPSTTL - Low-power Schottky TTL
 - LVTTTL - Low Voltage TTL

CMOS

CMOS(complementary MOS) este o familie de porți logice ce folosește atât tranzistoare nMOS, cât și tranzistoare pMOS pentru a realiza circuite logice ce necesită foarte puțin curent.

- Poarta NOT. Este realizată din doua tranzistoare, unul pMOS și unul nMOS. Cel pMOS are sursa legată la Vcc, iar cel nMOS are sursa legată la ground. Gateurile sunt legate amandouă la intrare, iar drenele sunt legate la ieșire.
- Când A = "0":
 - tranzistorul nMOS este închis, deci ieșirea este deconectată de la ground
 - tranzistorul pMOS este deschis, deci se face legatura între sursă și ieseire
- Când A = "1"
 - tranzistorul pMOS este închis.
 - tranzistorul nMOS este deschis, deci leagă ieșirea la ground

Celalalte porți logice se pot realiza pornind de la inversorul CMOS. Tranzițiile între High și Low se fac rapid.



Logica Combinatională

<http://www.falstad.com/circuit> [<http://www.falstad.com/circuit>].

În teoria circuitelor digitale, logica combinatională este domeniul logicii numerice care este implementat de circuite booleene, în care ieșirea circuitului depinde numai de stările intrărilor circuitului respectiv. Acest comportament este diferit de cel al logicii secvențiale, unde starea ieșirii depinde nu numai de starea prezentă a intrărilor ci și de stările anterioare. În alte cuvinte, logica secvențială are "memorie" iar logica combinatională nu.

Logica combinatională este folosită în circuite de calcul pentru implementarea algebrei booleene pe semnalele de intrare sau pe datele stocate în memorie. Circuitele digitale conțin de obicei o mixtură de circuite secvențiale și combinationalale. De exemplu, partea UAL-ului care se ocupă numai cu calculele matematice este construită numai cu circuite combinationalale, pe când registrele UAL-ului sunt implementate în logică secvențială. Alte circuite care sunt implementate în logică combinatională: half adder, full adder, sumator carry look-ahead, scăzătoare, multiplexoare, demultiplexoare, encodere și decodere.

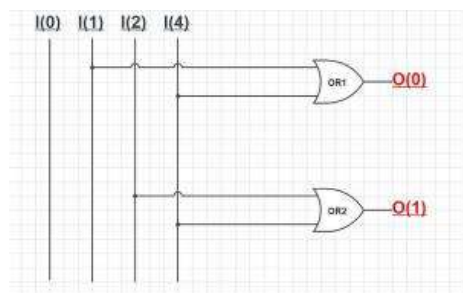
Encodere/Decodere

Un encoder este un circuit combinational care comprimă mai multe intrări binare într-un număr mai mic de ieșiri. De cele mai multe ori este folosit pentru a controla cererile de întrerupere către un procesor, pentru că are proprietatea să reacționeze la intrări în funcție de prioritatea lor și să pună întotdeauna pe ieșire codul intrării cu prioritatea cea mai mare. Drept exemplu aveți în tabelul mai jos modul de funcționare al unui encoder 4 la 2:

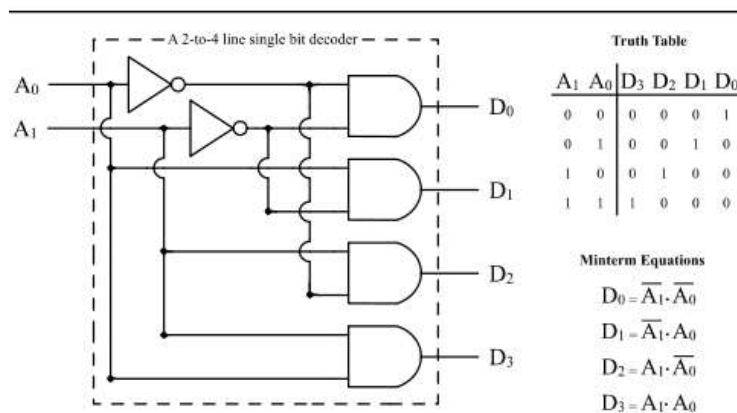
I1	I2	I3	I4	O1	O2
0	0	0	x	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

Dacă două sau mai multe intrări sunt active la un moment dat, cea cu prioritate mai mare va avea precedentă. În exemplul de mai sus, "x" reprezintă fie un 0 sau un 1 (nu contează ce valoare are intrarea de prioritate mai mică în momentul în care este concomitentă cu una de prioritate mai mare).

Un exemplu de implementare a unui encoder este dat în figura de mai jos:

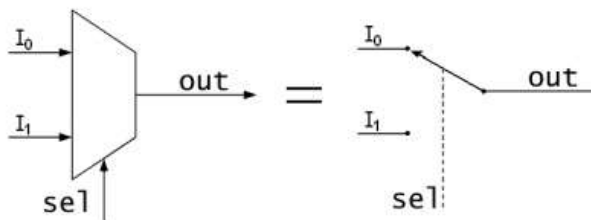


Funcția inversă unui encoder este implementată de un decoder. Acesta primește "n" intrări și furnizează "2^n" ieșiri. Decodificarea este utilă în aplicații care folosesc multiplexarea a datelor, display-uri 7-segment sau decodificarea adreselor de memorie. De exemplu, mai jos aveți un decoder 2 la 4:



Multiplexoare/Demultiplexoare

Un multiplexor sau mux este un dispozitiv care implementează multiplexarea unor semnale; acesta selectează o intrare analogică sau digitală din "2^n" intrări și o rutează la ieșirea circuitului. Selecția intrărilor se face cu ajutorul a "n" semnale de comandă. Multiplexoarele fac posibilă partajarea unei singure resurse sau mediu de transmisie de către mai multe semnale, de exemplu un convertor Analog-Digital sau o linie de comunicație pot transmite sau măsura mai multe semnale, fără a fi nevoie să se replice structura de măsurare sau transmisie. Un multiplexor poate fi asemănat cu un întrerupător cu un singur contact și mai multe poziții care este comandat de către liniile de selecție:



Un multiplexor 2 la 1 este cea mai simplă structură de acest gen. El are următoarea ecuație de funcționare:

$$Z = (A \cdot \bar{S}) + (B \cdot S)$$

unde A și B sunt cele două intrări, S este intrarea de selecție iar Z este ieșirea.

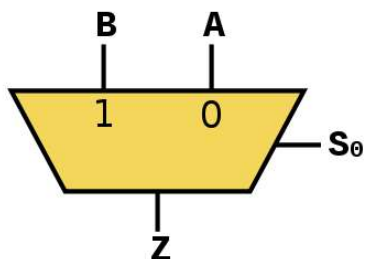
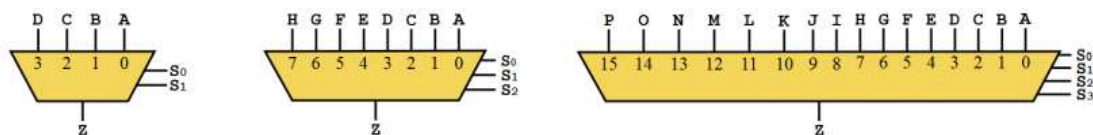


Tabela de adevăr pentru intrările și ieșirile multiplexorului 2 la 1 este următoarea:

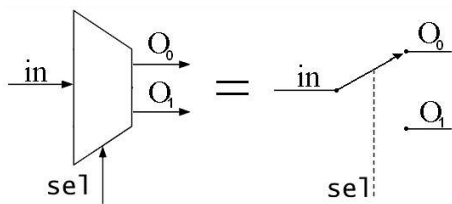
S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Adică ieșirea are valoarea lui A atunci când $S = 0$ și valoarea lui B atunci când $S = 1$.

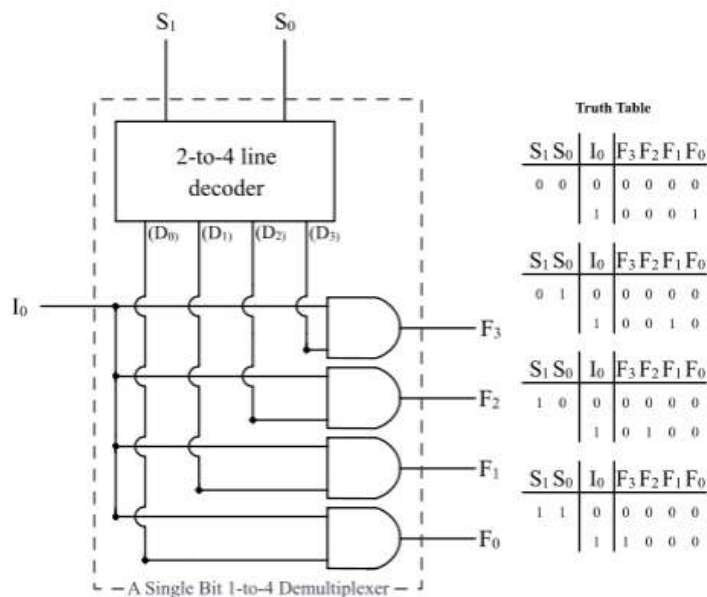
Folosind aceeași logică se pot construi multiplexoare cu mai mult de două intrări. Cele mai comune sunt 4 la 1, 8 la 1 și 16 la 1:



Un demultiplexor realizează funcția inversă multiplexorului, preia datele de pe o intrare și le rutează pe una din cele 2^n ieșiri în funcție de starea logică a celor n linii de selecție.



Un exemplu de circuit demultiplexor 1 la 4 este dat în figura de mai jos:



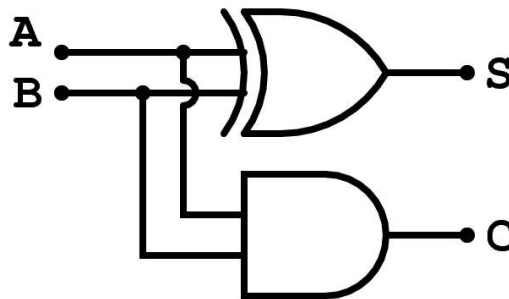
Sumatoare

Un sumator este un circuit digital care realizează operația de adunare a numerelor. În calculatoarele moderne, sumatoarele fac parte din Unitatea Aritmetică Logică (UAL) din cadrul procesorului.

Half Adder

Un 'half adder' este un circuit digital care realizează suma a doi operanzi de un singur bit notați **A** și **B**. Un circuit half adder furnizează la ieșire două semnale: C_{out} și S , unde $sum = 2 \times C_{out} + S$. Mai jos este inclusă tabela de adevăr pentru un half adder:

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



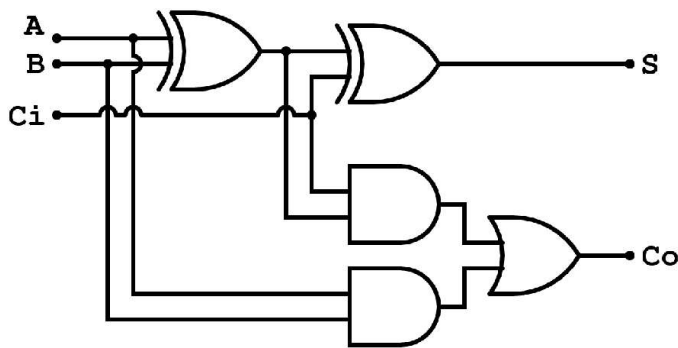
Un Half Adder este implementat cu două porți logice: un XOR și un AND:

Full Adder

Un sumator complet este un circuit digital care realizează suma a trei operanzi de un bit, notați ca **A**, **B**, și C_{in} . Sumatorul complet are două ieșiri reprezentate de obicei de către semnalele C_{out} și S unde $sum = 2 \times C_{out} + S$. Tabela de adevăr a circuitului este dată mai jos:

A	B	C_i	C_o	S
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

Un exemplu de implementare a sumatorului complet folosește formulele $S = (A \oplus B) \oplus C_{in}$ și $C_{out} = (A \cdot B) + (C_{in} \cdot (A \oplus B))$.



În această implementare poartă SAU finală înainte de ieșirea de carry poate fi înlocuită de o poartă XOR fără să se modifice funcționarea corectă. Acest lucru poate fi avantajos atunci când circuitul este implementat folosindu-se circuite integrate care conțin în un singur tip de poartă logică în capsulă lor.

Sumatorul Carry Look-ahead

Pentru a reduce timpul necesar unui calcul, determinat în special de propagarea semnalului de carry pe rangurile superioare, proiectanții folosesc sumatorul carry look-ahead. Ele funcționează prin generarea a două noi semnale (P și G) pentru fiecare rang binar în funcție de starea intrărilor : dacă un carry este propagat la rangurile superioare (cel puțin o intrare este '1'), dacă un carry este generat la nivelul celui rang (ambele intrări sunt '1') sau dacă un carry este oprit la acel rang (ambele intrări sunt '0'). În cele mai multe cazuri, P este ieșirea de suma a unui half adder și G este ieșirea de carry a aceluiași sumator. După ce termenii P și G sunt generați sunt creați biții de carry pentru fiecare rang în parte . Alte arhitecturi pot folosi carry look ahead la un nivel mai avansat cum ar fi Manchester carry chain, sumatorul Brent-Kung sau sumatorul Kogge-Stone.

În cazul adunării binare, $A + B$ se generează carry dacă și numai dacă ambele intrări A și B sunt 1. Dacă scriem $G(A,B)$ că predicatul logic care este adevărat atunci când $A + B$ generează, avem:

$$G(A,B) = A \cdot B$$

Adunarea a două intrări de un bit A și B va "propaga" dacă în urmă ei va rezultă un carry către rangul superior datorită faptului că avem un carry de la un rang inferior. De exemplu, dacă adunăm 37 + 62 suma cifrelor zecilor (3 și 6) va "propagă" orice carry venit de la rangul inferior (în cazul de față nu se va propaga nimic, dar acest lucru se schimbă dacă adunăm 38 + 62, de exemplu). În cazul adunării binare, $A + B$ propagă dacă și numai dacă cel puțin A sau B au valoarea 1. Dacă definim $P(A,B)$ că predicatul binar care este adevărat dacă și numai dacă $A + B$ propagă, avem:

$$P(A,B) = A + B$$

Uneori se folosește și o altă definiție a propagării. Conform acesteia, $A + B$ va propagă dacă suma generează carry dacă avem carry de la rang inferior și nu propagă dacă nu avem carry inferior. Orice definiție am folosi, modul în care sunt generați și propagați biții într-un sumator carry look-ahead este același. În cazul adunării binare, definiția de mai sus este exprimată prin:

$$P'(A,B) = A \oplus B$$

Pentru aritmetica binară, or este mult mai rapid față de xor și necesită mai puțini tranzistori pentru a fi și implementat. Cu toate acestea, pentru un sumator carry look-ahead pe mai multe nivele este mai avantajos să folosim $P'(A,B)$.

Date fiind conceptele de generare și propagare, când va exista carry în urma unei adunări binare? Acesta va avea loc în momentul în care adunarea generează "sau" rangul mai puțin semnificativ de carry și suma propagată. Dacă scriem în algebra booleană, cu C_i drept bitul de carry pentru rangul i și P_i și G_i biții de generare și propagare pentru rangul i, vom avea următoarea relație:

$$C_{i+1} = G_i + (P_i \cdot C_i)$$

Detalii de implementare

Pentru fiecare rang dintr-o secvență binară, sumatorul va determina dacă perechea de biți care trebuie adunată poate genera sau propaga carry. Acest lucru permite circuitului să "pre-proceseze" cei doi termeni ai adunării pentru a determina transportul înainte de a efectua adunarea propriu-zisă. Apoi, când această are loc, nu va exista nici o întârziere de propagare a carry-ului, ca în cazul unui Ripple Adder. Mai jos aveți un exemplu de calcul al termenilor de propagare și generare pentru un sumator de 4 biți :

$$C_1 = G_0 + P_0 \cdot C_0$$

$$C_2 = G_1 + P_1 \cdot C_1$$

$$C_3 = G_2 + P_2 \cdot C_2$$

$$C_4 = G_3 + P_3 \cdot C_3$$

Înlocuind C_1 în expresia lui C_2 , apoi C_2 pentru C_3 , apoi C_3 pentru C_4 obținem următoarele ecuații:

$$C_1 = G_0 + P_0 \cdot C_0$$

$$C_2 = G_1 + G_0 \cdot P_1 + C_0 \cdot P_0 \cdot P_1$$

$$C_3 = G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + C_0 \cdot P_0 \cdot P_1 \cdot P_2$$

$$C_4 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3$$

Circuite Integrate

In seria 7400 [http://en.wikipedia.org/wiki/7400_series] de circuite integrate exista mai multe tipuri de sumatoare, multiplexoare, demultiplexoare si encodeere:

S.No.	IC No.	Function	Output State
1	74157 [https://www.calstatela.edu/sites/default/files/groups/Department%20of%20Electrical%20and%20Computer%20Engineering/labs/74157.pdf]	Quad 2:1 mux.	Ieșirea are aceeași stare ca și intrarea
2	74158 [http://www.datasheetcatalog.org/datasheet2/e/0z44590ru86zdc1iyjd77hrp3yy.pdf]	Quad 2:1 mux.	Ieșirea este intrarea inversată
3	74153 [http://www.datasheetcatalog.org/datasheets/134/231447_DS.pdf]	Dual 4:1 mux.	Ieșirea are aceeași stare precum intrarea
4	74352 [http://www.datasheetcatalog.org/datasheet/HitachiSemiconductor/mXyzszvu.pdf]	Dual 4:1 mux.	Ieșirea este intrarea inversată
5	74151A [http://www.datasheetcatalog.org/datasheet/toshiba/4305.pdf]	8:1 mux.	Ambele tipuri de Ieșiri sunt disponibile (Ieșiri complementare)
6	74151 [http://www.datasheetcatalog.org/datasheet/philips/74HC_HCT151_CNV_2.pdf]	8:1 mux.	Ieșirea este intrarea inversată
7	74150 [http://webpages.ull.es/users/fexposit/74150.pdf]	16:1 mux.	Ieșirea este intrarea inversată
8	74139 [http://www.datasheetcatalog.org/datasheet/philips/74HC_HCT139_CNV_2.pdf]	Dual 1:4 demux.	Ieșirea este intrarea inversată
9	74156 [http://pdf1.alldatasheet.com/datasheet-pdf/view/117903/NSC/74156.html]	Dual 1:4 demux.	Ieșirea este open collector
10	74138 [http://www.datasheetcatalog.org/datasheet/philips/74HC_HCT138_CNV_2.pdf]	1:8 demux.	Ieșirea este intrarea inversată
11	74154 [http://www.datasheetcatalog.org/datasheet/philips/74HC_HCT154_CNV_2.pdf]	1:16 demux.	Ieșirea este intrarea inversată
12	74159 [http://pdf1.alldatasheet.com/datasheet-pdf/view/27375/TI/74159.html]	1:16 demux.	Ieșirea este open collector
13	74147 [http://www.datasheetcatalog.org/datasheet/philips/74HC_HCT147_CNV_2.pdf]	10:4 Priority Encoder	
14	74148 [http://www.datasheetcatalog.org/datasheets/90/455802_DS.pdf]	8:3 Priority Encoder	
15	74348 [http://www.datasheetarchive.com/74348-datasheet.html]	8:3 Priority Encoder	Ieșiri three-state
16	74155 [http://pdf1.alldatasheet.com/datasheet-pdf/view/23044/STMICROELECTRONICS/74HC155.html]	Dual 2:4 Decoder/Demultiplexer	
17	74237 [http://www.datasheetcatalog.org/datasheet/philips/74HC_HCT237_CNV_2.pdf]	1:8 Decoder/Demultiplexer	Address latch, Ieșiri active pe 1
18	74238 [http://www.datasheetcatalog.org/datasheet/philips/74HC_HCT238_CNV_2.pdf]	1:8 Decoder/Demultiplexer	Ieșiri active pe 1
19	74239 [http://maven.smith.edu/~thiebaut/270/datasheets/sn74ls48rev5.pdf]	Dual 2:4 Decoder/Demultiplexer	Ieșiri active pe 1
20	74248 [http://maven.smith.edu/~thiebaut/270/datasheets/sn74ls48rev5.pdf]	BCD to 7-segment Decoder/Driver	Ieșiri cu pull-up intern
21	74249 [http://maven.smith.edu/~thiebaut/270/datasheets/sn74ls48rev5.pdf]	BCD to 7-segment Decoder/Driver	Ieșiri open-collector
22	74537 [http://www.datasheetarchive.com/74537-datasheet.html]	BCD to Decimal Decoder	Ieșiri three-state
23	74538 [http://www.datasheetarchive.com/74538-datasheet.html]	1:8 Decoder	Ieșiri three-state
24	74539 [http://www.datasheetarchive.com/74539-datasheet.html]	Dual 1:4 Decoder	Ieșiri three-state

Lista completă o aveți aici: http://en.wikipedia.org/wiki/List_of_7400_series_integrated_circuits
[http://en.wikipedia.org/wiki/List_of_7400_series_integrated_circuits]

Exerciții

- (4p) Implementați funcția logică $f = [(a \text{ OR } b \text{ OR } c) \text{ AND } d]$ folosind oricare două din:
 - logică RTL
 - logică DTL
 - logică TTL
 - logică CMOS
- (3p) Implementați funcția logică $f = [(a \text{ OR } \text{not}(b \text{ AND } c)) \text{ XOR } d]$ folosind logică CMOS.

3. (2p) Realizați un sumator carry look-ahead pe 2 biți.
4. (1p) Realizați un comparator de 1 bit ce activează următorul circuit când $A = B$.
 - Hint: folosiți un comparator secvențial pe 2 biți și eliminați un bit de intrare
 - Hint: celelalte două ieșiri $A > B$ și $A < B$ vor rămâne în aer.

icalc/laboratoare/lab3.txt • Last modified: 2021/03/22 23:25 by giorgiana.vlasceanu