Proiectarea şi Verificarea cu HDL a Circuitelor Digitale

Danuț Burdia

Facultatea de Electronică, Telecomunicații şi Tehnologia Informației

Universitatea Tehnică “Gh. Asachi” din Iaşi

Cuprins

Referințe

- 1. Robert Dueck, 2000, *Digital Design with CPLD Applications and VHDL*, Ed. Thomson Delmar Learning.
- 2. Clive Maxfield, 2004, *The Design Warrior's Guide to FPGAs*, Ed. Elsevier-Newnes.
- 3. Peter J. Ashenden, 2002, *The Designer's Guide to VHDL – Second Edition*, Ed. Morgan Kaufmann Publishers.
- 4. Zainalabedin Navabi, 2005, *Digital Design and Implementation with Field Programmable Devices*, Ed. Kluwer Academic Publishers, Boston
- 5. Richard Munden, 2005, *ASIC and FPGA Verification: A Guide to Component Modeling*, Ed. Elsevier- Morgan Kaufmann Publishers.
- 6. William K. Lam, 2005, *Hardware Design Verification: Simulation and Formal Method-Based Approaches*, Ed. Prentice Hall,

Referințe (cont)

- 7. J.P. Deschampes, G.J.A. Bioul, G. D. Sutter, *Synthesis of Arithmetic Circuits – FPGA, ASIC and Embedded Systems*, Ed. John Willey & Sons.
- 8. Pong Chu, 2006, RTL.Hardware.*Design.Using.VHDL*, Ed. John Willey & Sons.
- 9. Uwe Meyer-Baese, 2007, *Digital Signal Processing with Field Programmable Gate Arrays*, Third Edition, Ed. Springer.
- 10. Chris Spear, *SystemVerilog for Verification - A Guide to Learning the Testbench Language Features*, Ed. Springer, 2008.
- www.eda.org
- www.xilinx.com
- www.digilentinc.com

Cap. 1 Concepte ale proiectării digitale

- 1.1 Sisteme de numarare
- 1.2 Aritmetică binară
- 1.3 Circuite combinaționale
- 1.4 Circuite de stocare
- 1.5 Circuite secvențiale
- 1.6 Memorii

1.1 Sisteme de numărare

- Tranzistorul este elementul de bază al tuturor circuitelor electronice digitale.
- Într-un circuit digital tranzistorul se comportă ca un comutator.
- Toate valorile pot fi de ex: (ON-OFF), (TRUE-FALSE), (3V,0V) sau (1,0).
- De aceea, toate numerele într-un sistem digital sunt reprezentate în baza 2 (binar).

1.1 Sisteme de numărare
1.1.2 Numere binare

- Un număr zecimal are n digiți iar tăria fiecărui digit este 10^i, unde i este poziția digitului (0 dreapta, *n-1* stânga)
  - Ex $(3256)_D = 6*10^0 + 5*10^1 + 2*10^2 + 3*10^3 = 3256$
- Un număr binar este evaluat similar:
  - Ex: $(10110)_B = 0*2^0 + 1*2^1 + 1*2^2 + 0*2^3 + 1*2^4 = 22$
- Conversie zecimal – binar
  - Numărul zecimal este împărțit în termeni 2^i necesari. Corespunzător valorii lui *i* există un 1 în numărul binar echiv.
  - Ex: $325 = 256+64+4+1=2^8+2^6+2^2+2^0=(101000101)_2$.
- Similar pentru numere fracționare. În binar, digiții fracționari au ponderile 2^{-1}, 2^{-2}, 2^{-3}, şamd.
  - Ex: $1101.011 = (2^3+2^2+2^0).(2^{-2}+ 2^{-3}) = 13.375$
  - $19.7 = (2^4+2^1+2^0).(2^{-1}+ 2^{-3}+ 2^{-4} +\ldots)\approx 10011.1011\ldots$

1.1 Sisteme de numărare
1.1.2 Numere hexazecimale

- Reprezintă o formă mai compactă de a reprezenta numerele
- Un digit în baza 16 este reprezentat prin exact 4 biți în binar.
  - Ex: $(11100101)_2 = (E5)_H$.
  - $(10011.101)_2 = (13.A)_H$.

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

1.2 Aritmetică binară

- 1.2.1 **Numere cu semn (signed numbers)**
  - Într-un sistem digital numerele binare sunt reprezentate pe un număr fix de biți
  - bitul cel mai semnificativ este folosit pentru semn
    - 0 = nr. pozitiv, 1 = nr. negativ
  - Ex: pe un bus de 8 biți pot fi reprezentate numere cuprinse între (-127,+127).
    - +25 = **0**0011001 -25 = **1**0011001
- 1.2.2 **Adunarea binară**
  - Este similară cu adunarea în zecimal, chiar mai uşoară!
    - **Ex:**

Carry Bits:	0	0	0	1	1	1	1	
A:	0	1	0	0	1	0	1	1
B:	0	0	1	0	1	1	0	1
Result:	0	1	1	1	1	0	0	0

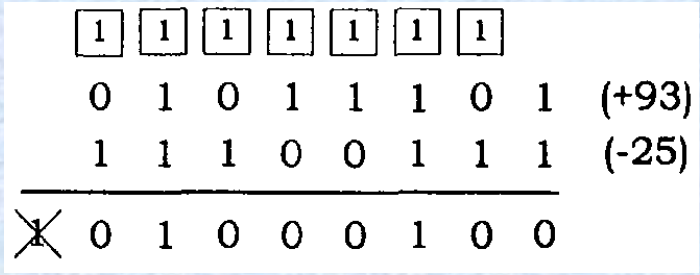
1.2 Aritmetică binară

- 1.2.3 **Scăderea binară**
  - Se poate efectua ca în zecimal folosind împrumut de la bitul mai mare
  - Necesită alt proces față de adunare, deci implementare diferită
- 1.2.4 **Reprezentarea în complement față de 2**
  - Este o alternativă pentru a efectua scăderile la fel ca adunarea.
  - Pentru a schimba un nr. pozitiv în unul negativ, se complementează toți biții, apoi se adună 1.
  - **Ex:** -25 este obținut astfel:
    - **00011001 (+25)**
    - **11100110 (complementăm toți biții)**
    - **00000001 (adunăm 1)**
    - **11100111 (-25) (în complement față de 2)**
  - **MSB** este pentru semn: **0 – pozitiv, 1 – negativ.**

1.2 Aritmetică binară

Sumarea binară folosind complement față de 2

- În loc de a efectua A-B, scăderea este realizată adunând A+(-B), unde (-B) este complementul față de 2 a lui B.
- Exemplu: 93-25=68 93=01011101 -25=11100111



- Ultimul bit este neglijat când:
  - rezultatul adunării binare a două numere negative este negativ, sau
  - dacă rezultatul este un număr pozitiv când adunăm un număr pozitiv cu unul negativ.

1.2 Aritmetică binară

Sumarea binară folosind complement față de 2

- 1.2.5 **Depăşirea numerică în complement față de 2**
  - **Este diferită față de depăşirea numerică în cazul adunării a două numere în reprezentarea binară fără semn (unsigned binary)**
- **Exemple (bitul cu roşu reprezintă transportul):**

-39 + 92 = 53:

	1			1	1			
		1	1	0	1	1	0	0	1
+	0	1	0	1	1	1	0	0
	0	0	1	1	0	1	0	1

Transport fără depăşire. Sumă corectă

-19 + -7 = -26:

	1	1	1	1			1	
		1	1	1	0	1	1	0	1
+	1	1	1	1	1	0	0	1
	1	1	1	0	0	1	1	0

Transport fără depăşire. Sumă corectă

44 + 45 = 89:

			1		1	1		
		0	0	1	0	1	1	0	0
+	0	0	1	0	1	1	0	1
	0	1	0	1	1	0	0	1

Sumă fără depăşire şi fără transport. Sumă corectă

104 + 45 = 149:

		1	1		1			
		0	1	1	0	1	0	0	0
+	0	0	1	0	1	1	0	1
	1	0	0	1	0	1	0	1

Depăşire, fără transport. Sumă incorectă

-75 + 59 = -16:

		1	1	1	1	1	1	
	1	0	1	1	0	1	0	1
+	0	0	1	1	1	0	1	1
	1	1	1	1	0	0	0	0

Sumă fără depăşire şi fără transport. Sumă corectă

-103 + -69 = -172:

		1	1	1		1	1	
	1	0	0	1	1	0	0	1
+	1	0	1	1	1	0	1	1
	0	1	0	1	0	1	0	0

Depăşire, cu transport. Sumă incorectă

1.2 Aritmetică binară

Sumarea binară folosind complement față de 2

- 1.2.5 **Depăşirea numerică în complement față de 2 (alte exemple)**

10 + -3 = 7:

	1	1	1	1				
	0	0	0	0	1	0	1	0
+	1	1	1	1	1	1	0	1
	0	0	0	0	0	1	1	1

Transport fără depăşire. Sumă corectă

127 + 1 = 128:

	1	1	1	1	1	1	1	
	0	1	1	1	1	1	1	1
+	0	0	0	0	0	0	0	1
	1	0	0	0	0	0	0	0

Depăşire, fără transport. Sumă incorectă

-1 + 1 = 0:

	1	1	1	1	1	1	1	
	1	1	1	1	1	1	1	1
+	0	0	0	0	0	0	0	1
	0	0	0	0	0	0	0	0

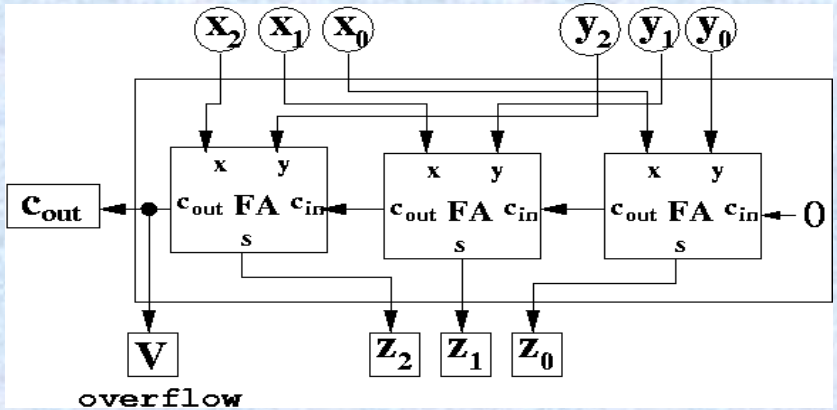
Transport, fără depăşire. Sumă corectă

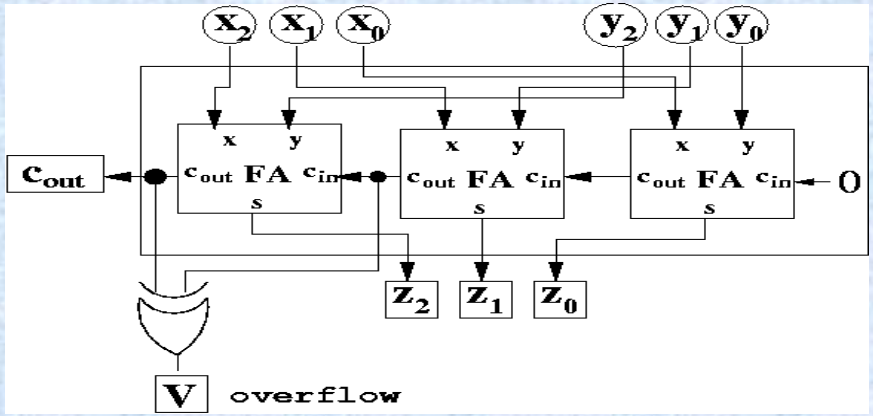
- **Reguli pentru depistarea depăşirii:**
  - **Depăşirea poate apare numai când ambii operanzi au acelaşi semn**
  - **Există depăşire atunci când bitul de semn al rezultatului este diferit de bitul de semn al celor doi operanzi.**
  - **Sau: Există depăşire atunci când pentru bitul de semn valoarea împrumutului este diferită de cea a transportului (carry_in ≠ carry_out)**

1.2 Aritmetică binară

Circuite pentru sumarea binară şi detectarea depăşirii

- Depăşire la adunarea binară a numerelor fără semn (unsigned)
  - Depăşirea apare cînd suma este mai mare decât numărul maxim ce poate fi reprezentat pe k biți (2^k-1)
- Depăşire la adunarea binară a numerelor cu semn (signed)
  - Depăşirea este detectată folosind o poartă XOR pentru cin şi cout aferent bitului cel mai semnificativ (bitul de semn)





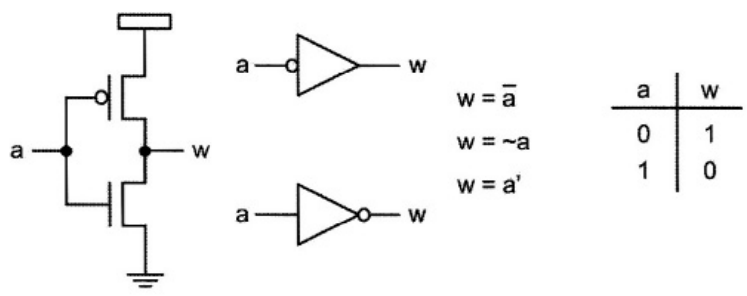
Concluzie: depăşirea este echivalentă cu cout=1 doar la adunarea numerelor fără semn. Depăşirea depinde de reprezentarea numerelor (unsigned sau 2C)

1.3 Porți logice elementare

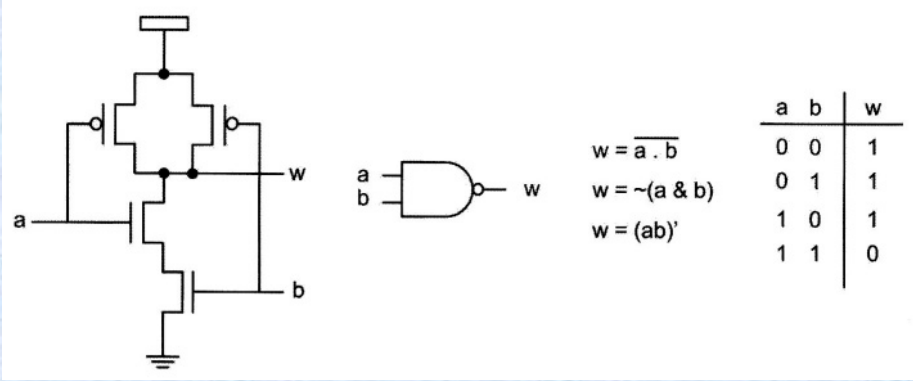
- Sistemul valorilor logice
  - Exemplu: sistemul cu 4 valori logice

Value	Description
0	Forcing 0 or Pulled 0
1	Forcing 1 or Pulled 1
Z	Float or Hign Impedence
X	Uninitialized or Unknown

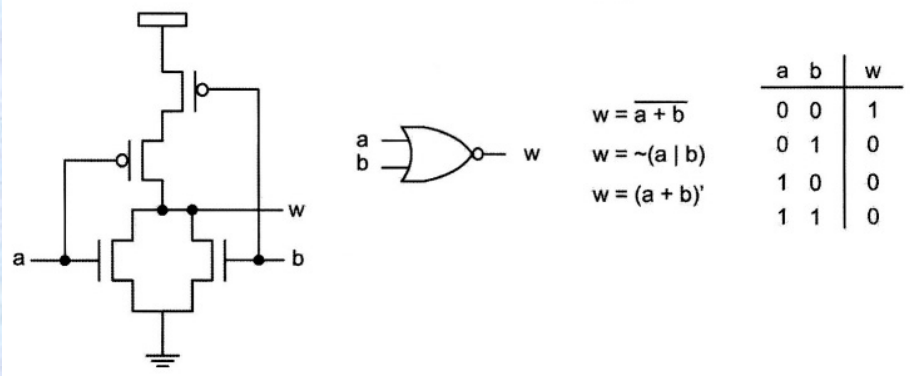
- Inversorul CMOS



- Poarta CMOS NAND (SI-NU)

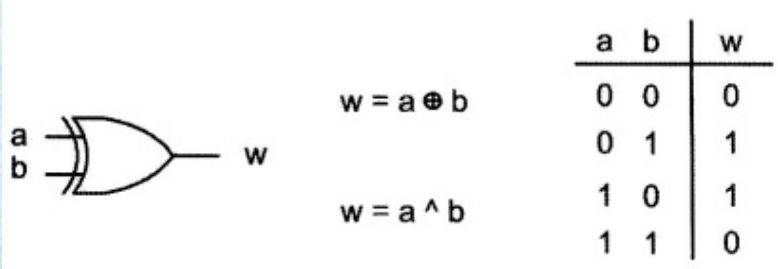


- Poarta CMOS NOR (SAU-NU)



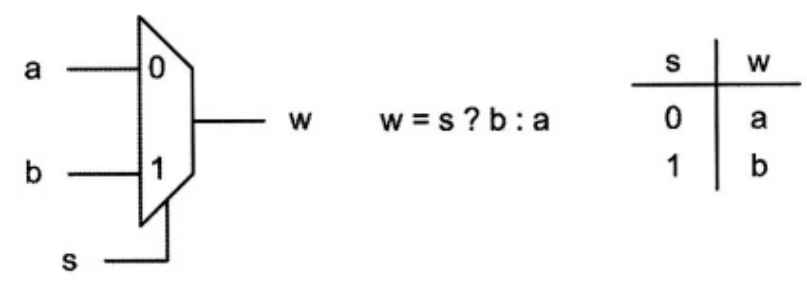
1.3 Porți logice elementare

- Poarta XOR (SAU-exclusiv)



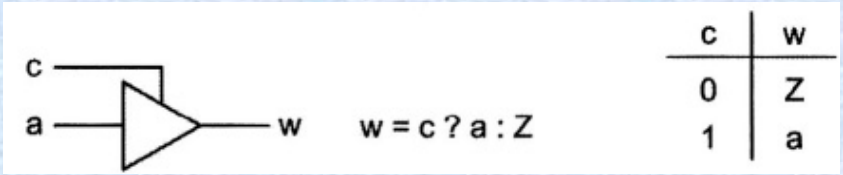
a	b	w
0	0	0
0	1	1
1	0	1
1	1	0

- Multiplexor



s	w
0	a
1	b

- Porți cu trei stări (tri-state)



c	w
0	Z
1	a

- Porțile elementare formează un set de structuri cu care poate fi proiectat orice circuit digital.
- În continuare sunt prezentate metode pentru implementarea funcțiilor logice folosind sistemul de porți elementare

1.4 Proiectarea circuitelor combinaționale

- 1.4.1 Alegebra booleană
  - Este utilizată pentru a facilita corespondența dintre porțile logice şi funcționalitatea unui proiect digital.
  - $a + 0 = a$
  - $a \cdot 1 = a$
  - $a + 1 = 1$
  - $a \cdot 0 = 0$
  - $a + a = a$
  - $a \cdot a = a$
  - $a + b = b + a$
  - $a \cdot b = b \cdot a$
  - $a + (b + c) = (a + b) + c$
  - $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
  - $a + b \cdot c = (a + b) \cdot (a + c)$
  - $a \cdot (b + c) = a \cdot b + a \cdot c$

$$a = \overline{\overline{a}}$$

$$a + \overline{a} = 1$$

$$a \cdot \overline{a} = 0$$

$$a + \overline{a} \cdot b = a + b$$

$$a \cdot (\overline{a} + b) = a \cdot b$$

- Legile lui DeMorgan

$$\overline{a \cdot b} = \overline{a} + \overline{b}$$

$$\overline{a + b} = \overline{a} \cdot \overline{b}$$

- Odată ce se cunoaşte funcționalitatea, aceasta se înlocuieşte cu expresii booleene
- Pe baza regulilor de mai sus, funcționalitatea poate fi aranjată, minimizată şi pusă într-o formă care poate fi realizată cu porți logice.

1.4 Proiectarea circuitelor combinaționale

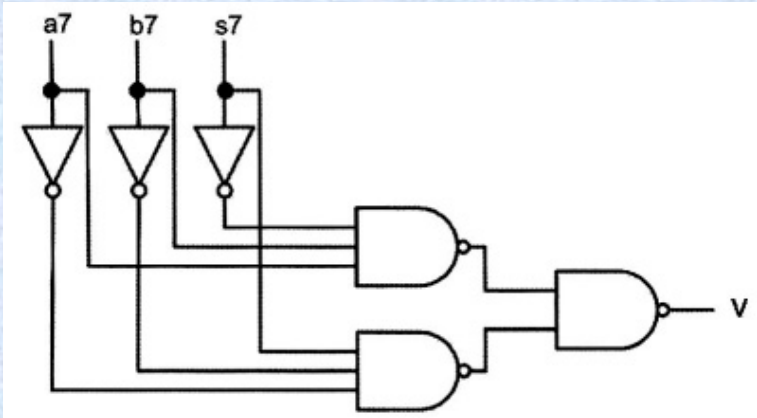
- Exemplu: considerăm problema depăşirii în cazul adunării în reprezentarea în complement față de 2. Presupunând reprezentarea pe 8 biți, biții de semn ai operanzilor şi rezultatului sunt *a*7, *b*7 şi *s*7. Depăşirea (*v*=1) are loc dacă *a*7=1, *b*7=1 şi *s*7=0 **sau** *a*7=0, *b*7=0 şi *s*7=1.
- Această funcționalitate este exprimată de următoarea expresie booleană:

$$v = a7 \cdot b7 \cdot \overline{s7} + \overline{a7} \cdot \overline{b7} \cdot s7$$

- Aplicând regulile lui DeMorgan rezută:

$$v = \overline{\overline{a7 \cdot b7 \cdot \overline{s7}} \cdot \overline{\overline{a7} \cdot \overline{b7} \cdot s7}}$$

- Rezultă următoarea implementare:
-

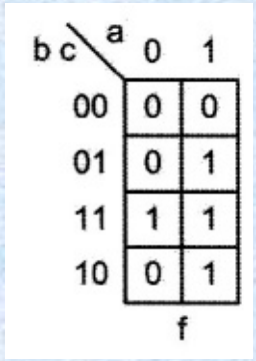


Detector depăşire în cazul adunării în complement față de 2

1.4 Proiectarea circuitelor combinaționale

- **1.4.2 Diagrama Veitch-Karnaugh**

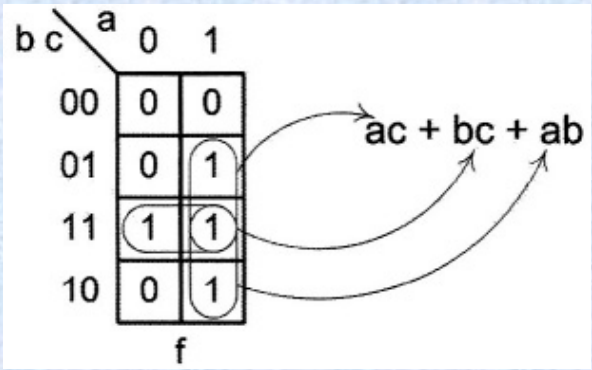
Row	a	b	c	f
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



Suma de produse (mintermeni)

$$f = \bar{a} \cdot b \cdot c + a \cdot \bar{b} \cdot c + a \cdot b \cdot c + a \cdot b \cdot \bar{c}$$

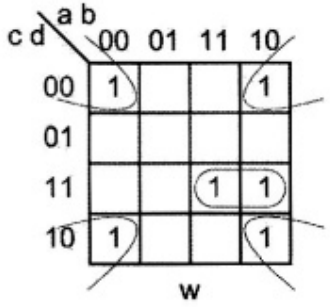
- **Minimizare**



1.4 Proiectarea circuitelor combinaționale

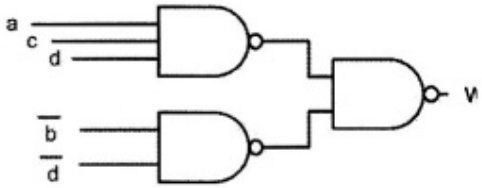
Diagrama Veitch-Karnaugh: minimizare şi implementare

a	b	c	d	w
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

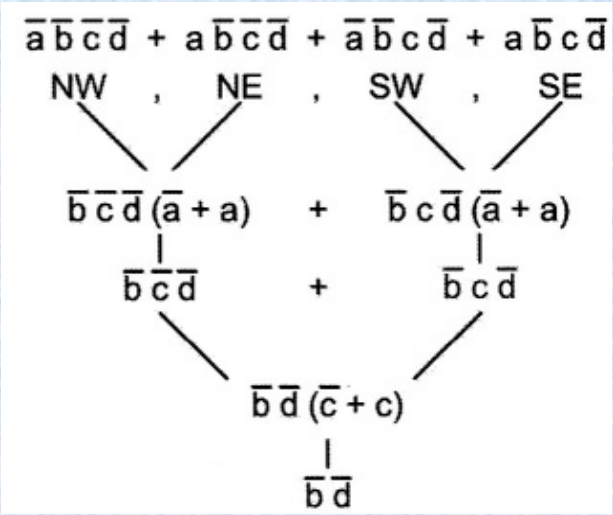


$w(a, b, c, d) = \overline{b}\,\overline{d} + acd$

$= \overline{\overline{b}\,\overline{d}} \cdot \overline{acd}$



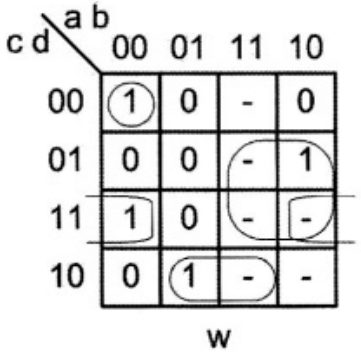
- Combinarea celor 4 colțuri din diagramă



1.4 Proiectarea circuitelor combinaționale

Diagrama Veitch-Karnaugh: valori care nu contează (don't care)

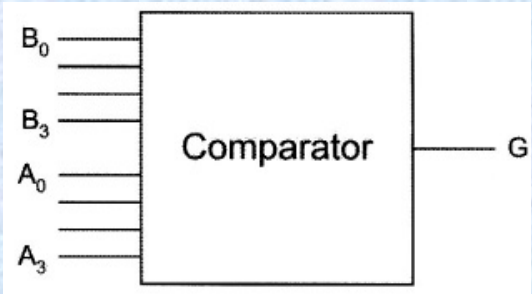
a	b	c	d	w
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	-
1	0	1	1	-
1	1	0	0	-
1	1	0	1	-
1	1	1	0	-
1	1	1	1	-



$w(a, b, c, d) = \bar{a}.\bar{b}.\bar{c}.\bar{d} + \bar{b}.c.d + b.c.\bar{d} + a.d$

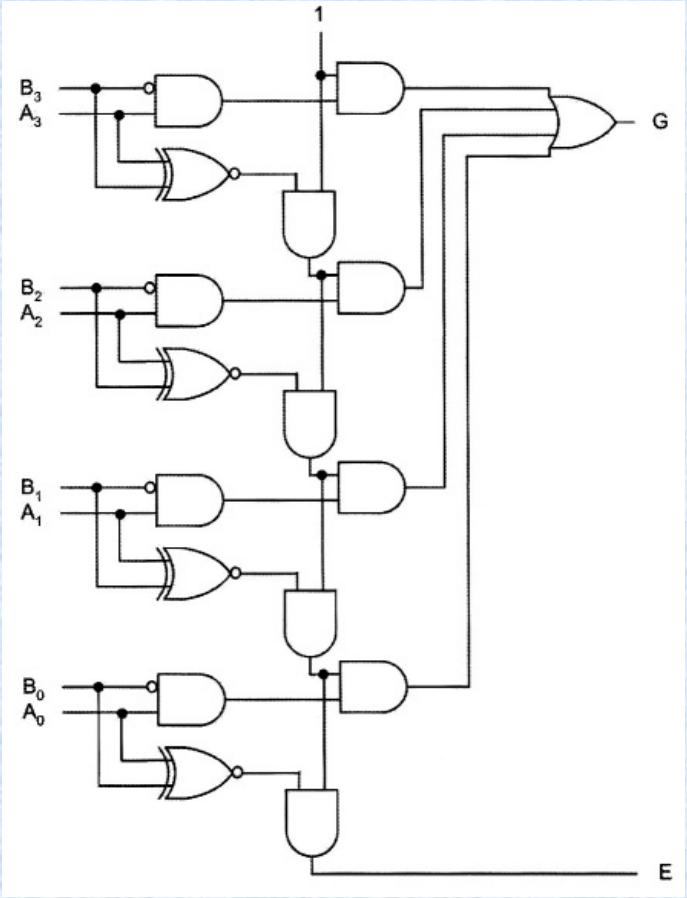
1.4 Proiectarea circuitelor combinaționale

- **1.4.3 Structuri iterative**
  - Minimizarea funcțiilor folosind regulile booleene sau diagrama V-K este aplicabilă doar în cazul funcțiilor mici.
- **Exemplu: Comparator pe 4 biți**



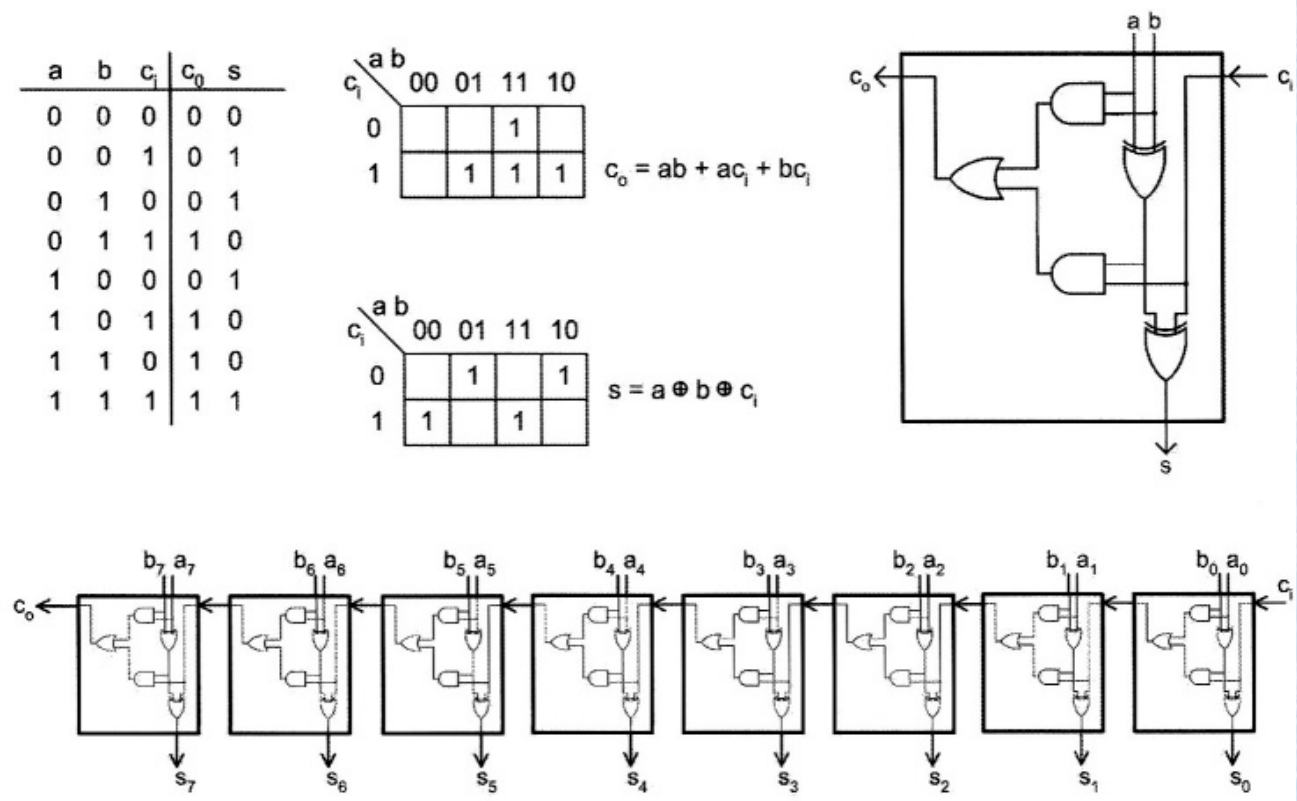
- **Putem compara bit cu bit începând de la MSB**
- **Ex:** Dacă A_3>B_3 atunci G=1.
- **D.pd.v. logic:** Termenul $A_3 \cdot \overline{B_3}$ formează o poartă AND care este intrare pentru o poartă OR ce generează G
- Decizia de comparare pe baza A_2 şi B_2 are loc numai dacă A_3=B_3, adică dacă $\overline{A_3 \oplus B_3} = 1$

- **Implementare comparator**



1.4 Proiectarea circuitelor combinaționale

Structuri iterative: sumator pe 8 biți cu transport



- **Structurile iterative pot fi cascadabile, extensibile şi, uneori, configurabile**
- **În proiectare este utilă existența unei biblioteci cu asemenea componente.**

1.4 Proiectarea circuitelor combinaționale

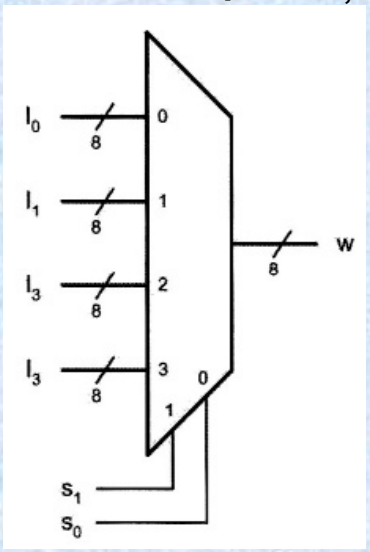
1.4.4 Multiplexoare şi decodoare

- Multiplexoare

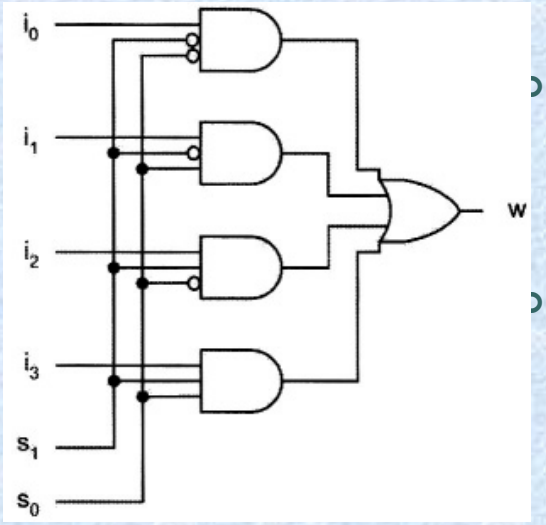
- Nr. de biți ai intrărilor determină dimensiunea multiplexorului

- Un MUX cu n intrări necesită $s = \log_2(n)$ linii de selecție

4 to 1 MUX pe 8 biți



4 to 1 MUX pe 1 bit



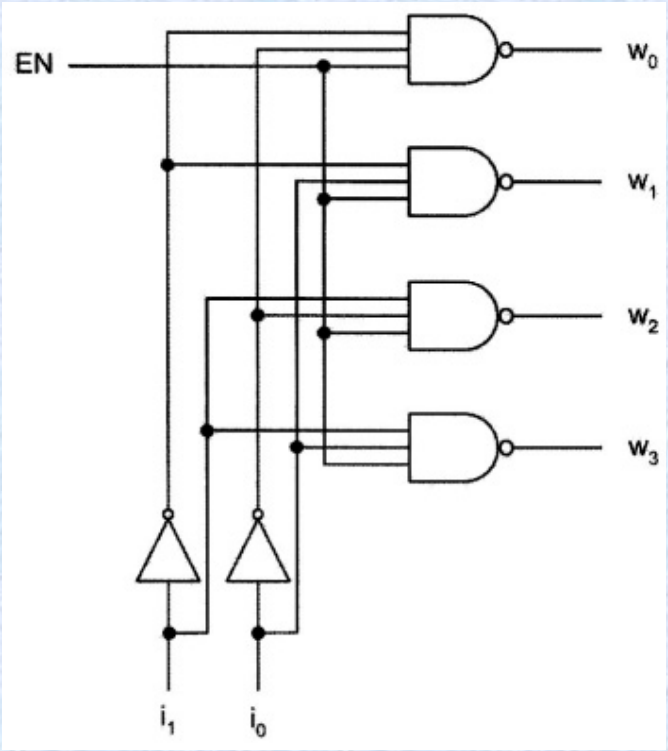
- Multiplexoarele sunt folosite pentru selectare date, “bussing”, conversie paralel-serial şi pentru implementarea funcțiilor logice arbitrare.
- Un 2-to-1 Mux pe 1-bit poate fi folosit pentru implementarea porților NOT, AND, şi OR.
- Împreună cu un inversor (NOT) un 2-to-1 Mux poate fi utiliyat pentru implementarea majorităților primitivelor logice.
- Datorită acestei proprietăți, multe celule FPGA conțin multiplexoare pentru implementarea funcțiilor logice.

1.4 Proiectarea circuitelor combinaționale

1.4.4 Multiplexoare şi decodoare

- Decodor
- În general este un circuit combinațional care în funcție de anumite valori (cod) de la intrări generează diferite coduri la ieşiri.
- **Exemplu:** decodor BCD afişaj 7 segmente: are intrare pe 4 biți (cod BCD) şi ieşire pe 7 biți corespunzătoare celor 7 segmente
- Altă definiție: un decodor are un număr de ieşiri egal cu numărul de combinații ale intrărilor. Pentru fiecare combinație ale intrărilor numai o anumită ieşire a decodorului este activă.

- Decodor binar 2x4



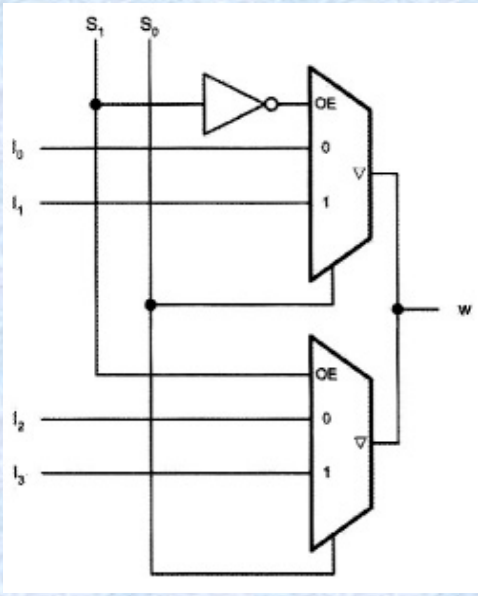
1.4 Proiectarea circuitelor combinaționale

1.4.5 Circuite cu intrări de activare/dezactivare

Dacă un circuit are intrare EN (Enable) atunci toate ieşirile circuitului sunt inactive când intrarea EN este inactivă.

Un circuit cu intrare OE (output-enable) este pentru ieşiri cu 3 stări. Dacă OE este inactiv, atunci ieşirile sunt în starea Z (înaltă impedanță)

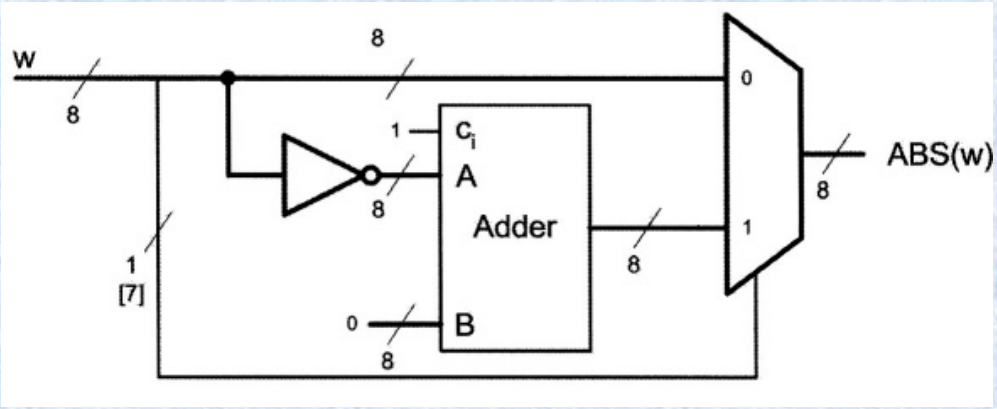
Circuit cu intrare OE



1.4 Proiectarea circuitelor combinaționale

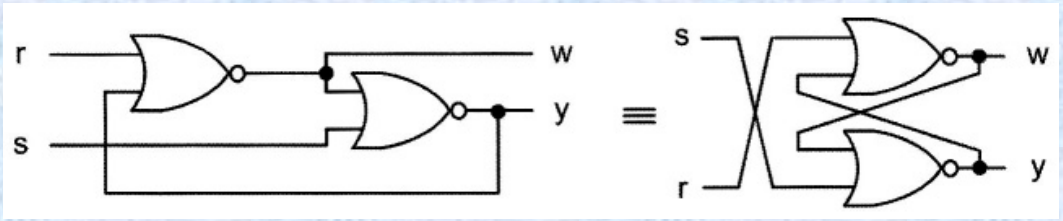
Descrierea la nivel mai înalt

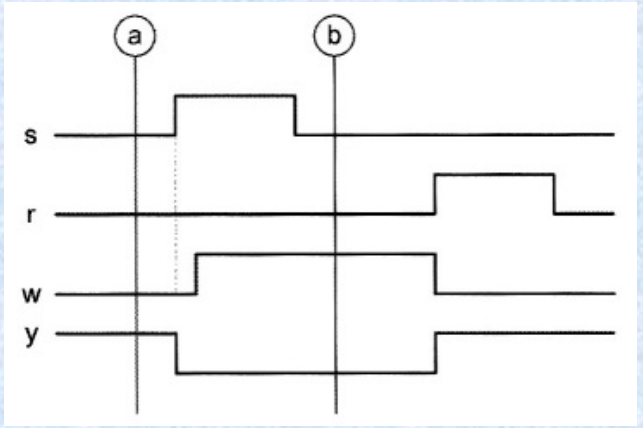
- Tranzistoare ⟹ Porți logice. Sunt nivele inferioare de descriere.
- Porți logice în alte structuri: **sumatoare, comparatoare, decodoare şi multiplexoare**
- La nivelul acestor structuri proiectantul este capabil să gândeasca proiectul la un nivel funcțional mai înalt
- Acest nivel: RTL (Register Transfer Level). Majoritatea proiectelor digitale actuale sunt gândite la acest nivel.
- **Ex**: circuit de calcul a valorii absolute conținând un sumator şi multiplexor



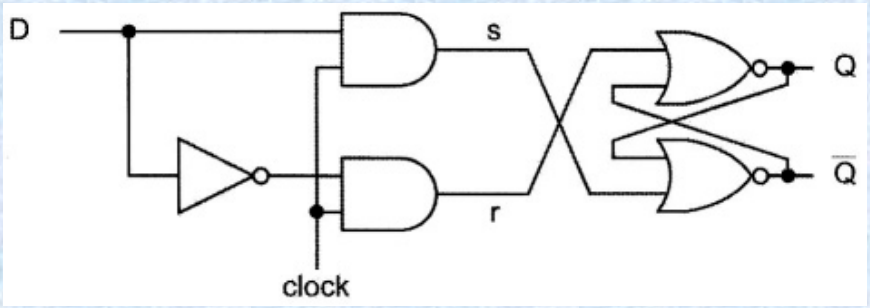
1.5 Elemente de stocare

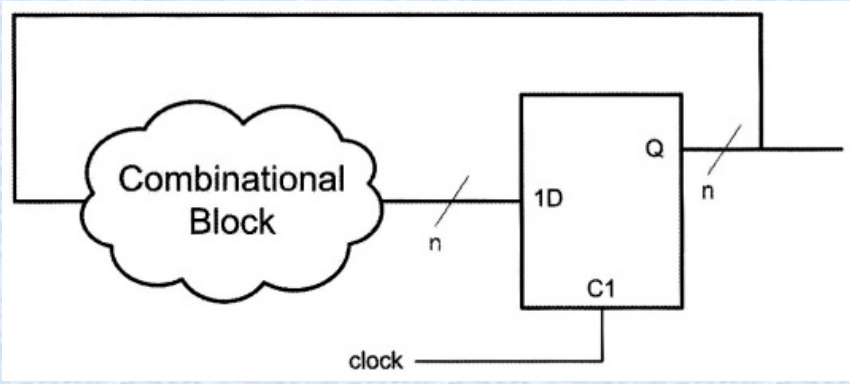
- Latch simplu





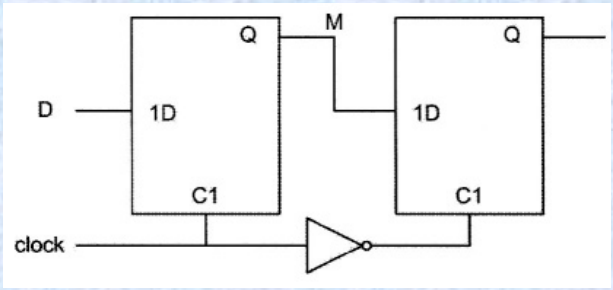
- Latch D
  - Când clock=1, Q=D şi o memorează până când din nou clock=1
  - Datorită transparenței nu poate fi folosit în circuite cu reacție (rezultat imprevizibil, oscilații)

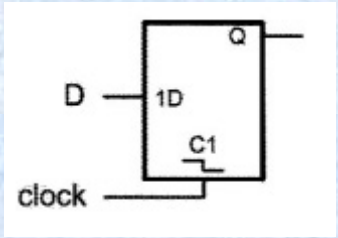


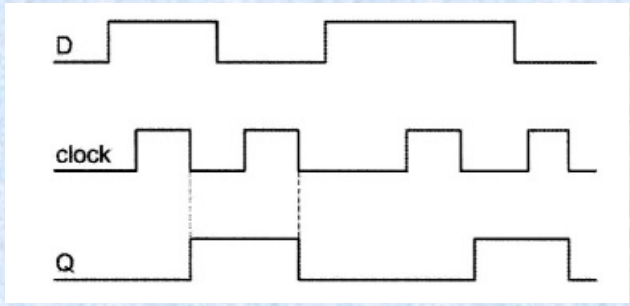


1.5 Elemente de stocare

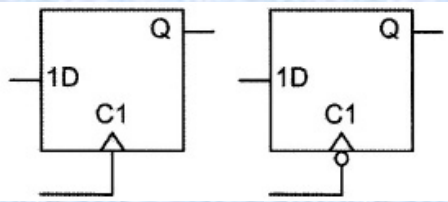
- Bistabile (flip-flops)
  - Două latch-uri cu clock inversat



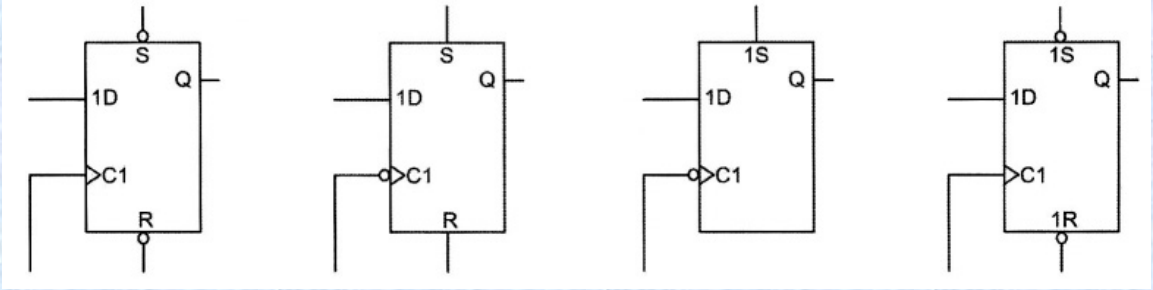




  - Bistabile active pe frontul clock-ului (edge-trigger flip-flops)

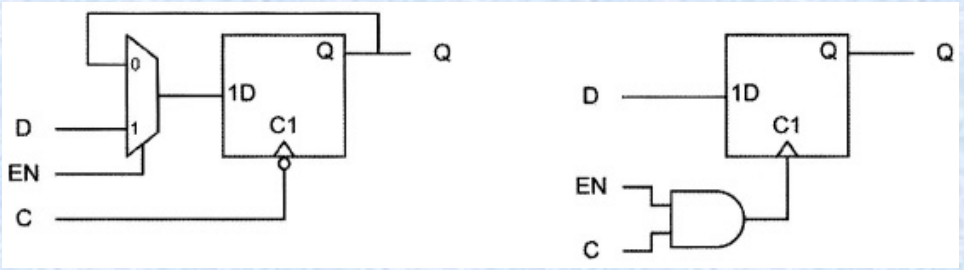


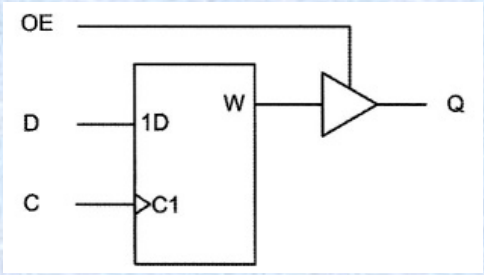
  - Bistabile cu intrări de control (set, reset, enable)
    - Sincrone sau asincrone



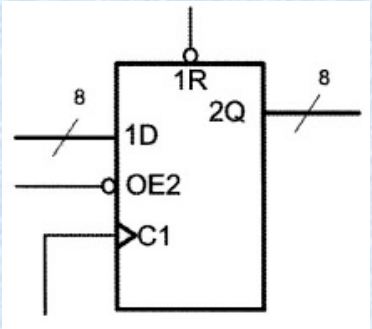
1.5 Elemente de stocare

- Bistabile cu intrare de activare (enable)



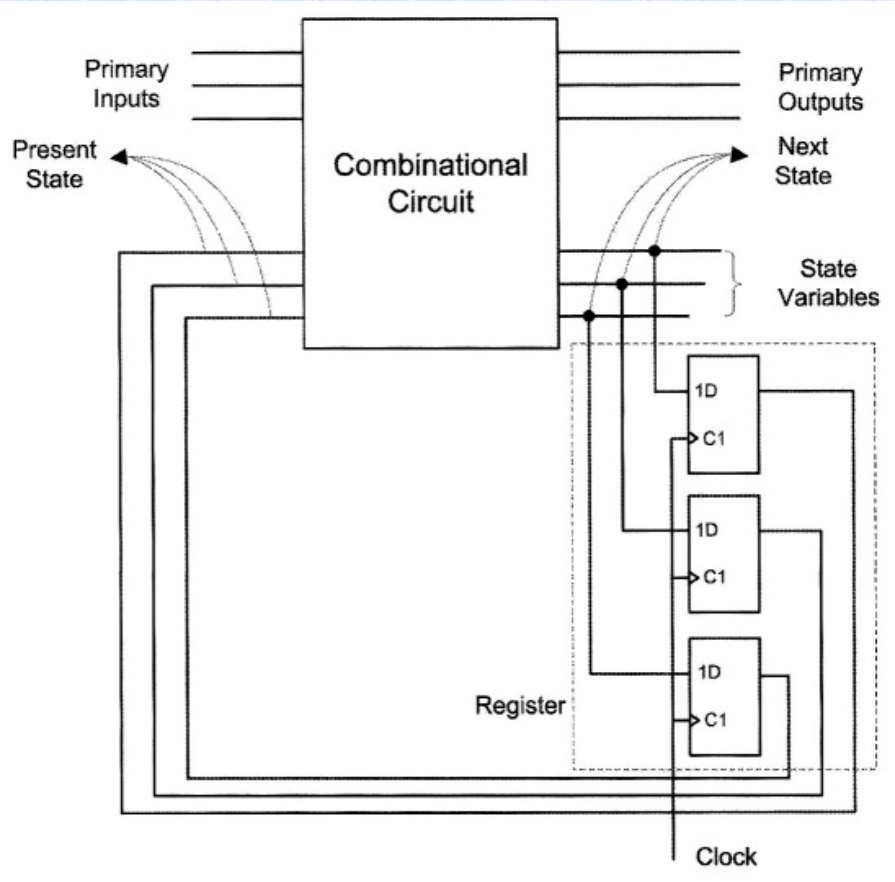


- Registre
  - Structuri formate din mai multe bistabile având intrări comune pentru clock şi pentru semnalele de control.



1.6 Circuite secvențiale

- Circuit secvențial = sistem digital care are memorie şi deciziile pe care le ia pentru o anumită intrare depind de ce este memorat.
- **Maşini cu stări finite**
  - Numărul de stări este determinat de numărul elementelor de memorie.
  - Un circuit cu n biți de memorie are 2^n stări
  - Toate circuitele secvențiale pot fi considerate ca maşini cu stări finite (FSM)
  - Dacă în calea de reacție există bistabile cu clock atunci circuitul secvențial este sincron (ex: modelul Huffman)

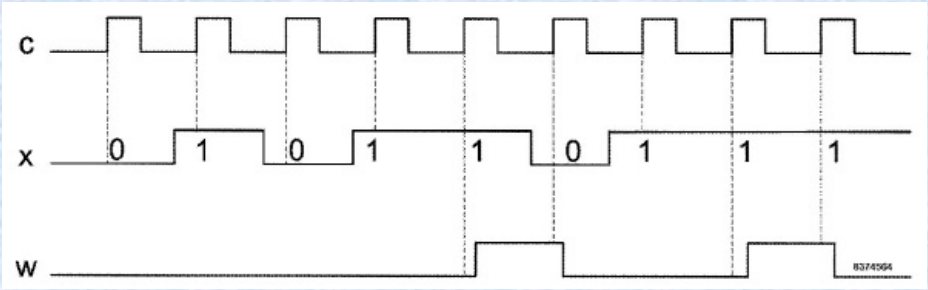


Modelul Huffman a unui circuit secvențial

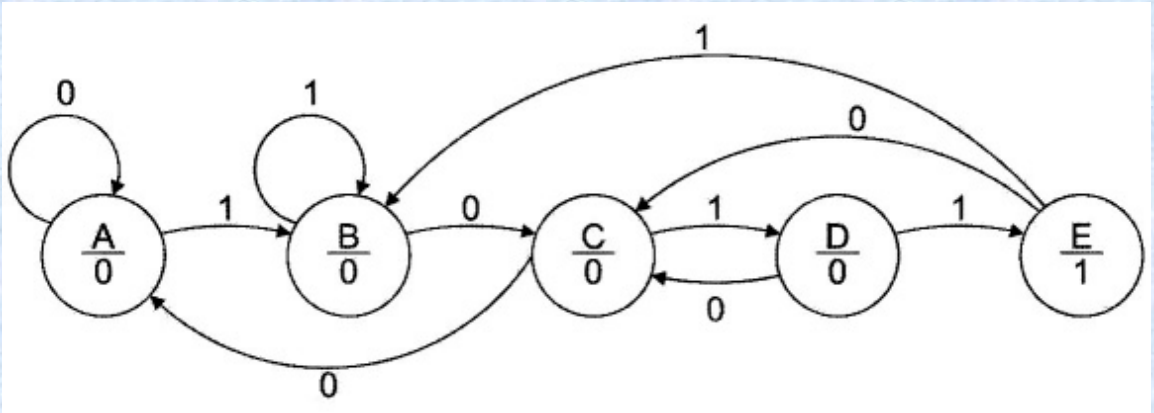
1.6 Circuite secvențiale

Proiectarea maşinilor cu stări finite

- Exemplificare: detector de secvență 1011



- Diagrama de stări



Tabelul stărilor

State	x = 0	x = 1	w
A	A	B	0
B	C	B	0
C	A	D	0
D	C	E	0
E	C	B	1
	State⁺		w

1.6 Circuite secvențiale

Proiectarea maşinilor cu stări finite

- Codarea stărilor (states asignements)
  - Se alocă un număr binar pentru fiecare stare. Ex:
  - y_2, y_1, y_0 = variabile de stare a FSM
- Tabelul de tranziție a stărilor

				x = 0	x = 1	
	y_2	y_1	y_0	$y_2^+ y_1^+ y_0^+$		w
A:	0	0	0	0 0 0	0 0 1	0
B:	0	0	1	0 1 0	0 0 1	0
C:	0	1	0	0 0 0	0 1 1	0
D:	0	1	1	0 1 0	1 0 0	0
E:	1	0	0	0 1 0	0 0 1	1

State	y_2	y_1	y_0
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0

- Partea combinațională + partea de regiştri
  - Regiştri = bistabile cu clock comun
  - Partea combinațională:

    Stabileşte valorile de la intrările bistabilelor pe baza ieşirilor acestora şi a intrării FSM (tabelul de intrări a bistabilelor)

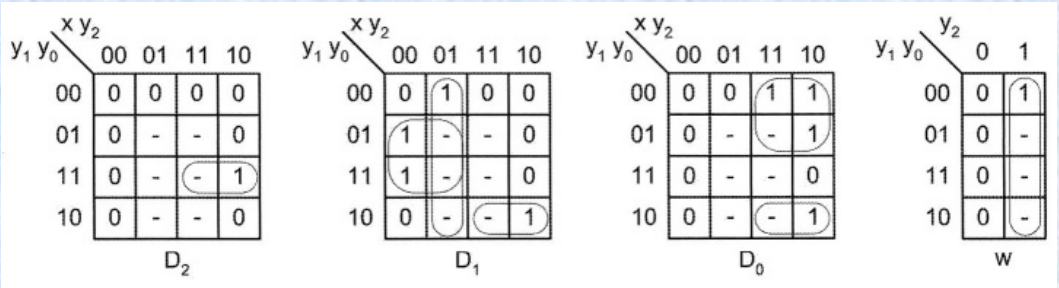
Tabelul de intrări a bistabilelor

			x = 0	x = 1	
y_2	y_1	y_0	$D_2 D_1 D_0$		w
0	0	0	0 0 0	0 0 1	0
0	0	1	0 1 0	0 0 1	0
0	1	0	0 0 0	0 1 1	0
0	1	1	0 1 0	1 0 0	0
1	0	0	0 1 0	0 0 1	1
1	0	0	- - -	- - -	-
1	0	0	- - -	- - -	-
1	0	0	- - -	- - -	-

1.6 Circuite secvențiale

Proiectarea maşinilor cu stări finite

- Implementarea părții combinaționale



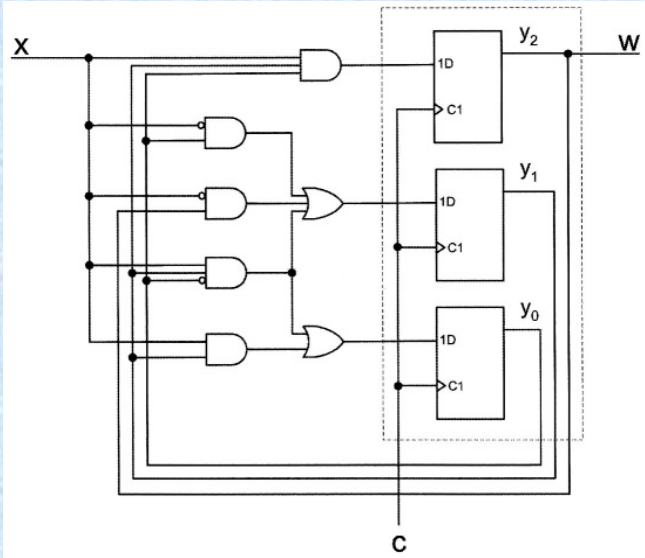
$$D_2 = x\, y_1\, y_0$$

$$D_1 = \overline{x}\, y_0 + \overline{x}\, y_2 + x\, y_1\, \overline{y_0}$$

$$D_0 = x\, \overline{y_1} + x\, \overline{y_0}$$

$$w = y_2$$

- Implementarea decodorului de secvență 1011

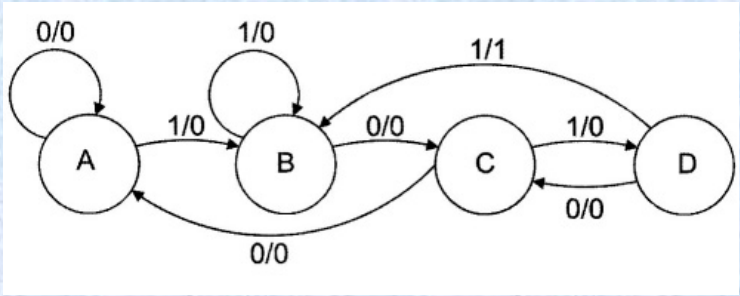


1.6 Circuite secvențiale

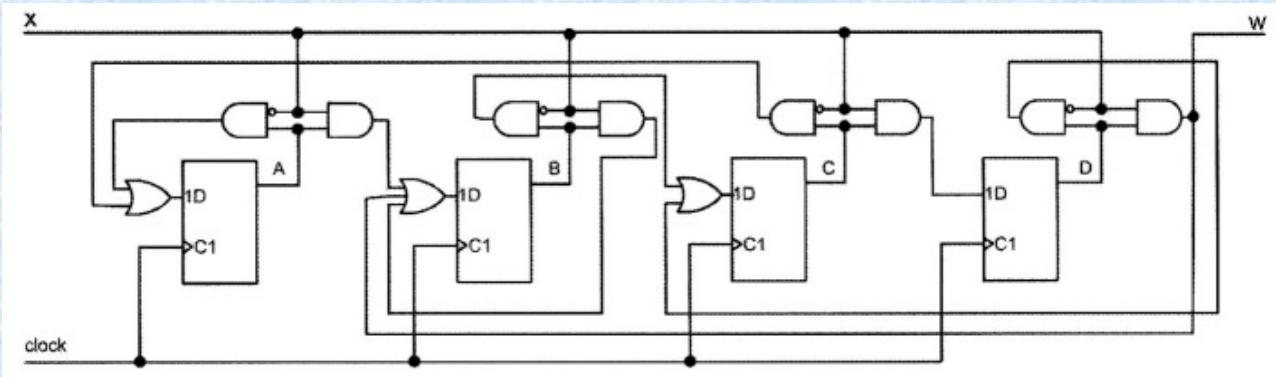
Maşini Moore şi Mealy

- FSM Moore – ieşirea depinde doar de stare
- FSM Mealy – ieşirea depinde atât de stare cât şi de intrare

Diagrama de stări pentru FSM Mealy a detectorului 1011



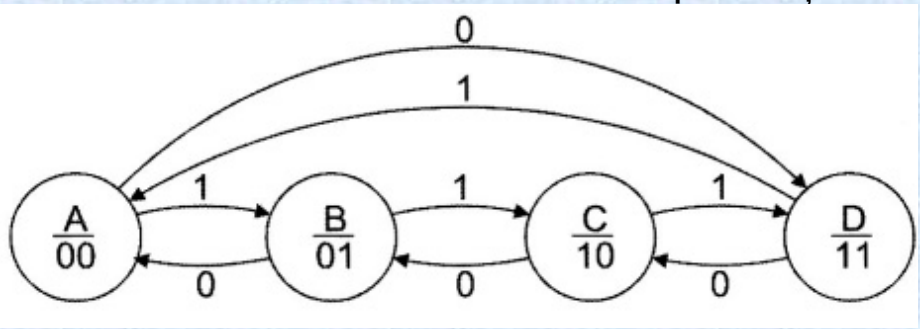
- Implementare “one-hot”: se alocă câte un bistabil pentru fiecare stare

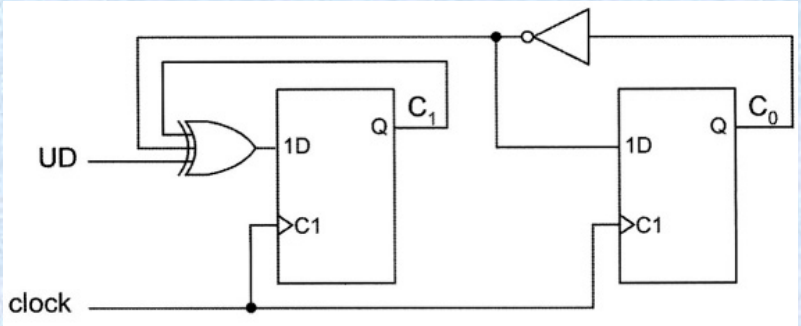


1.6 Circuite secvențiale

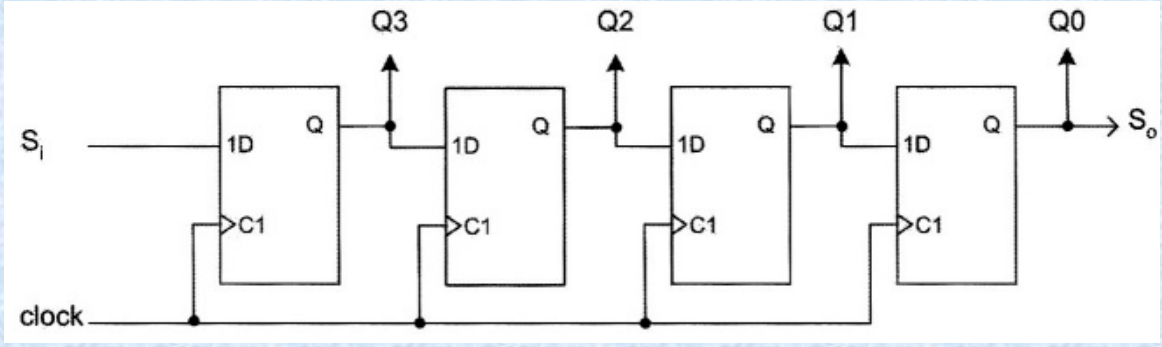
Blocuri secvențiale utilizate în proiectarea la nivel RTL

- Numărătoare
  - Ex: Numărător reversibil pe 2 biți





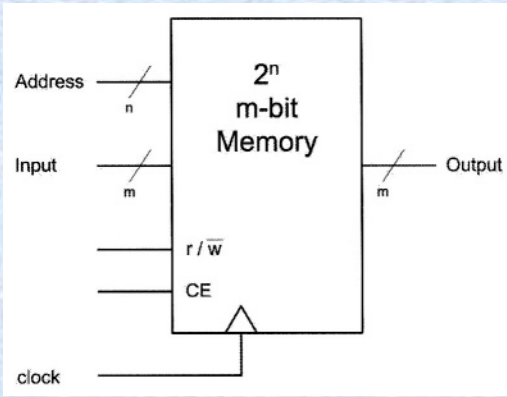
- Registre de deplasare (shifters)



Registru de deplasare pe 4 biți

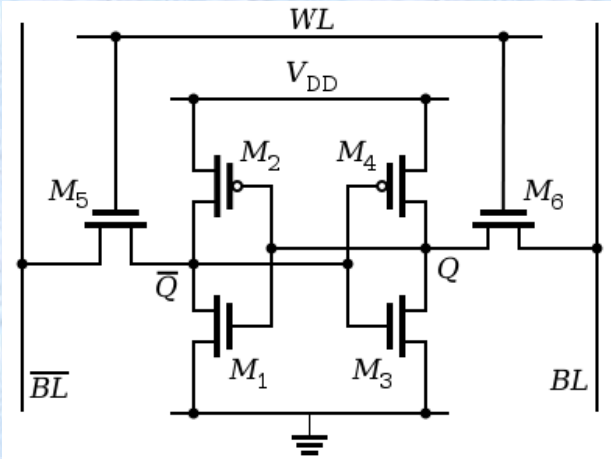
1.7 Memorii

- În cea mai simplă formă, sunt tablouri (matrice) 2-D de bistabile sau tablouri 1-D de registre.
- Numărul de bistabile (celule) dintr-o linie = lungimea cuvintelor (datelor), m.
- Cuvintele de memorie sunt aranjate a.î. fiecare pot fi citite-scrise individual.
- O memorie cu n linii de adresă are cel mult 2^n cuvinte de m biți.
- Deoarece accesarea cuvintelor din memorie se poate face independent de locație, memoriile sunt numite RAM (Random Access Memory).
- Diverse tipuri de RAM: Static RAM (SRAM), Dynamic RAM (DRAM), Synchronous Dynamic RAM (SDRAM), etc.
- Memorii volatile (SRAM, DRAM, SDRAM, etc.)
- Memorii nevolatile (ROM, Flash memory, etc)

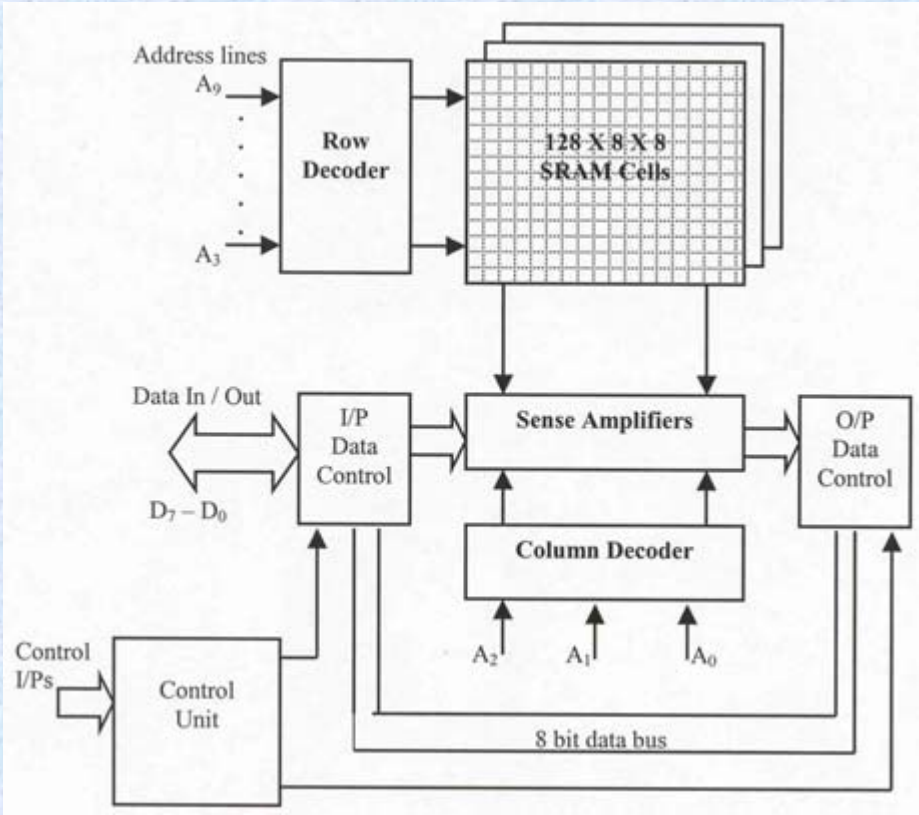


1.7 Memorii
Structura SRAM

- **Celula de memorie SRAM**

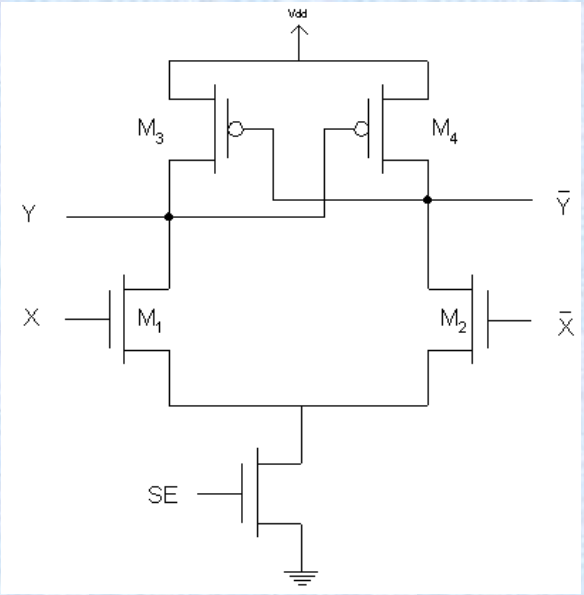


- **Memorie SRAM 1KBytes**

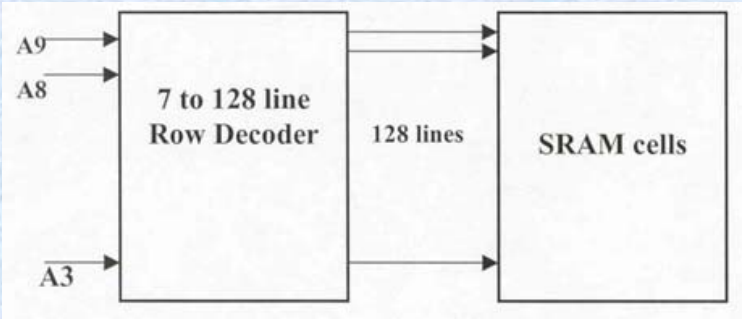


Structura SRAM

- **Amplificator de sens** (Sense Amplifier)
- În timpul citirii x şi $\overline{X}$ sunt conectate la BL şi $\overline{BL}$.



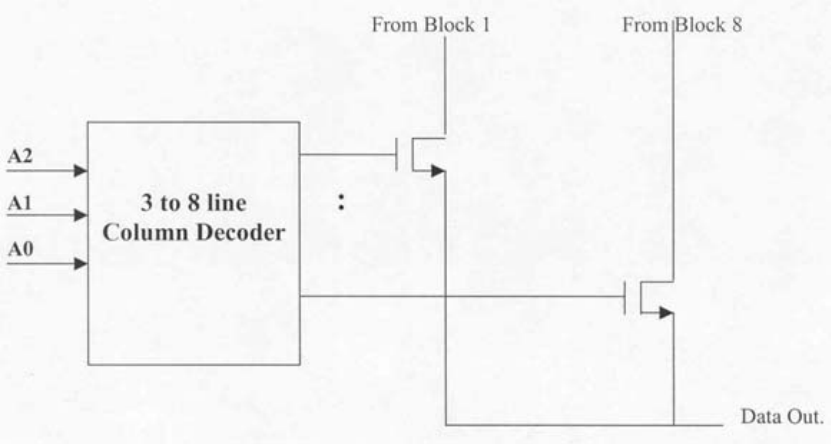
- **Decodor de linii** (7 la 128)



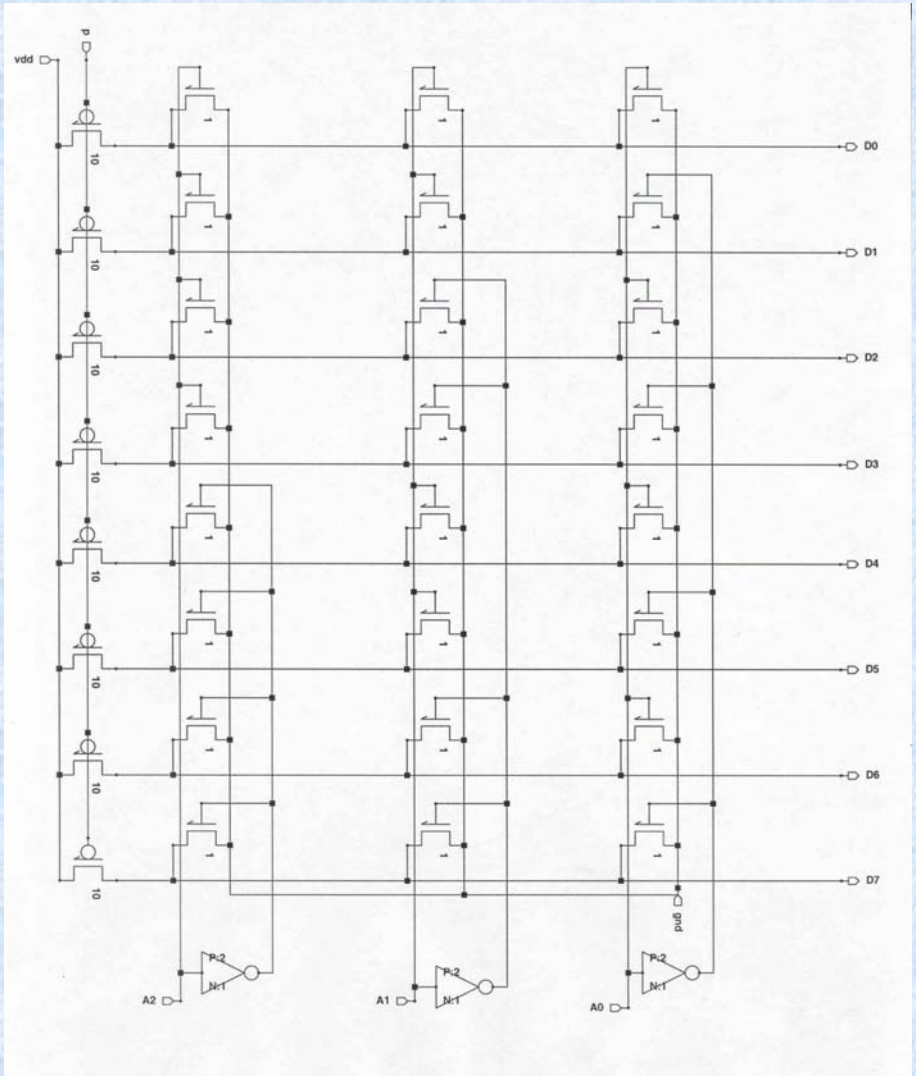
- Memoria SRAM este organizată ca 128 linii x 8 coloane, fiecare coloană conținând 8 biți.
- Cele 128 linii de la decodor sunt conectate la liniile de adresă (WL) ale celulelor SRAM.
- Fiecare din cele 128 linii comandă câte 64 celule SRAM (8 coloane x 8 biți)

Structura SRAM

Decodor de coloane 3x8



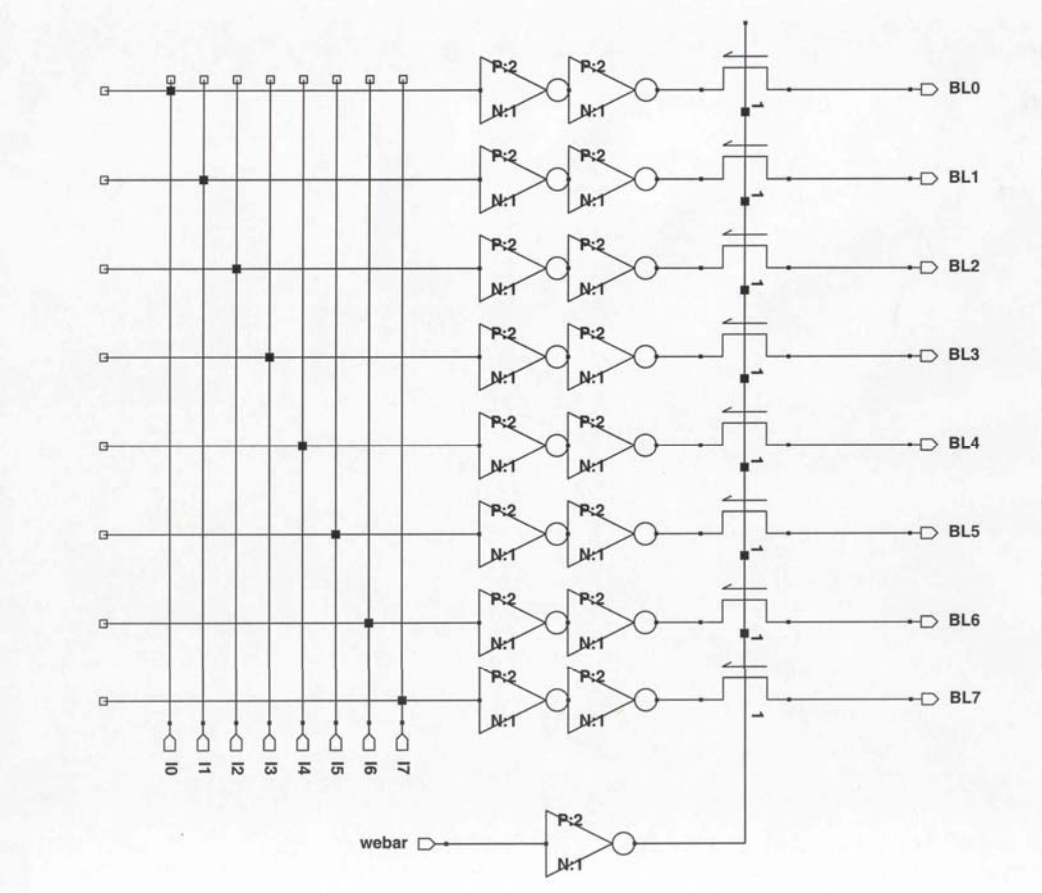
- Structura din figură este de tip decodor SAU-NU dinamic şi necesită 2 faze:
  - Preîncărcare (pre-charge)
  - Evaluare
- **Pre-charge:** Intrarea P este activată, tranz. PMOS conduc, ieşirile deocodorului sunt la VDD.
- **Evaluare**: PMOS blocate. Intrările de adresă sunt activate. Toate liniile se descarcă cu excepția uneia singure (linia decodată).
- Linia decodată (1 logic) va conduce toate tranzistoarele NMOS din acea linie, astfel că datele vor fi disponibile la ieşire.



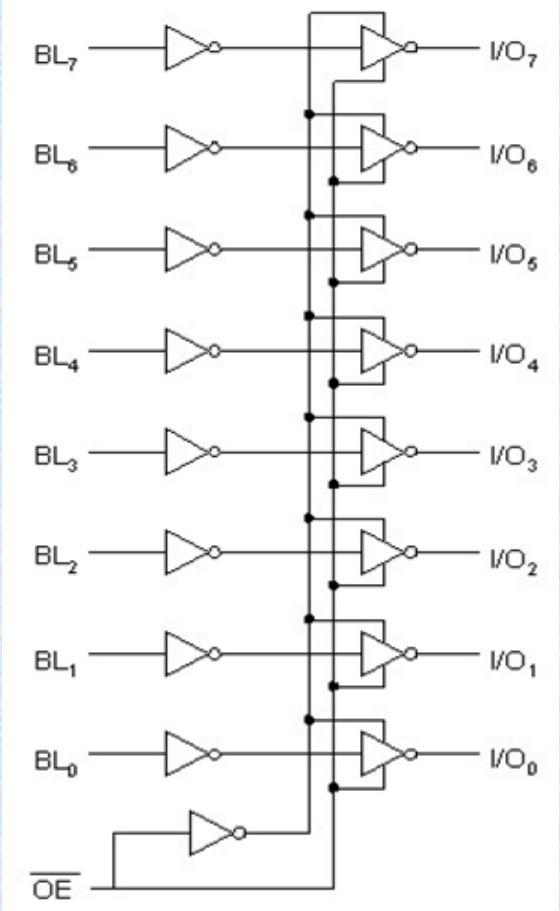
Structura SRAM

Blocuri de control intrare şi ieşire

Bloc control date intrare



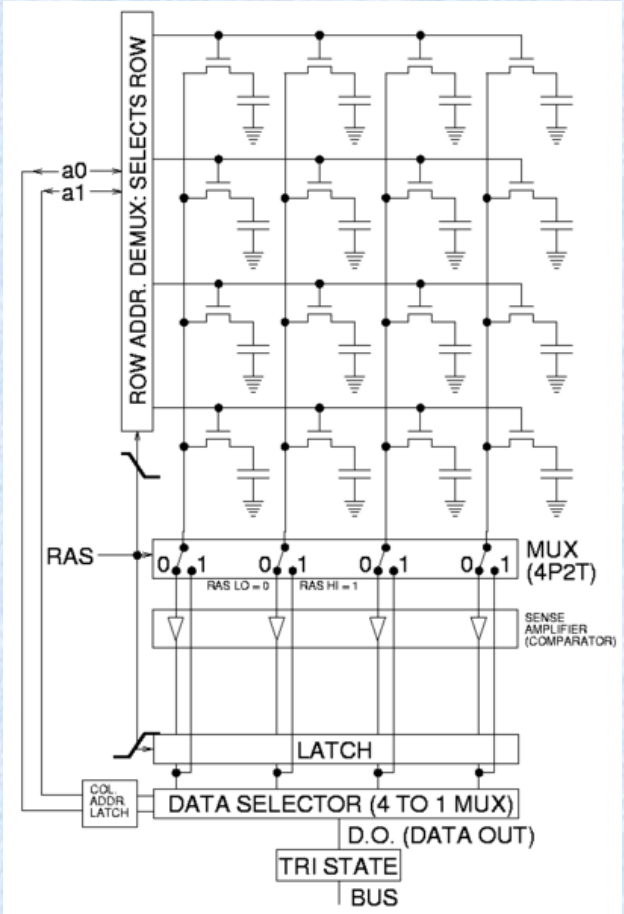
Bloc control date ieşire



DRAM (Dynamic RAM)

- **Structură simplă:** numai un tranzistor şi o capacitate (tot tranzistor) pentru fiecare bit de memorie.
- Necesită refresh periodic, de aceea denumirea de “dynamic” RAM.

Principiu citire 4x4 DRAM



Principiu scriere 4x4 DRAM