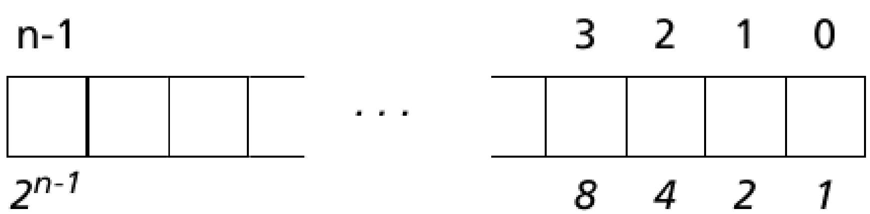
Reprezentarea binară a datelor

Bit: cea mai mică unitate de informație (da/nu, on/off, adevărat/fals, 5 V/0 V)

Octet (byte): grup de 8 biți
→ $2^8 = 256$ stări diferite

Cuvânt (word): numărul de biți care poate fi procesat de un computer într-un singur pas (lungimea regiştrilor procesorului)
→ dependent de computer/procesor
→ valori uzuale: 32 sau 64. Reprezentare:



- Un cuvânt de lungime *n* poate stoca 2^n tipare de biți (stări) distincte;
- **Semnificația acestor tipare depinde de contextul în care ele sunt utilizate!**
- Exemplu - “cuvintele” reprezentabile pe 3 biți:

000	100
001	101
010	110
011	111

→ $2^3 = 8$ tipare distincte

Sisteme de numerație importante pentru aplicații numerice

1. Introducere

- Un sistem de numerație este precizat de:
- Un **set de bază** *Z* de cifre (sau litere); de ex. *Z*={0,1,2,3,4,5,6,7,8,9}
- O **bază** *B = card(Z)*; de ex. *B = card(Z) = 10.*
- Un **număr** este o secvență liniară de cifre.
- **Valoarea unei cifre** pe o anumită poziție depinde de semnificația atribuită ei şi de poziția pe care se află.
- Ex.- un număr reprezentat pe un cuvânt de lungime *n*

$$N_B = d_{n-1}d_{n-2}\dots d_1 d_0$$

Valoarea numărului:

$$N_B = \sum_{i=0}^{n-1} d_i \cdot B^i = d_{n-1}B^{n-1} + d_{n-2}B^{n-2} + \dots + d_1 B^1 + d_0 B^0$$

2. Sisteme de numerație importante în lucrul cu computerele

Nume	**Bază**	**Set cifre**
binar	2	0,1
octal	8	0,1,2,3,4,5,6,7
zecimal	10	0,1,2,3,4,5,6,7,8,9
hexazecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

2.1 Sistemul zecimal

Z = {0,1,2,3,4,5,6,7,8,9}; B=10

- Valoarea fiecărei poziții la **stânga** unei cifre **creşte** cu o putere a lui 10
- Valoarea fiecarei poziții la **dreapta** unei cifre **scade** cu o putere a lui 10

Exemplu:

$$3795_{10} = 5 \cdot 10^0 + 9 \cdot 10^1 + 7 \cdot 10^2 + 3 \cdot 10^3$$

$$= 5 + 90 + 700 + 3000$$

2. 2 **Sistemul binar**

Z = {0,1}; B=2

- Valoarea fiecărei poziții la **stânga** unei cifre **creşte** cu o putere a lui 2
- Valoarea fiecarei poziții la **dreapta** unei cifre **scade** cu o putere a lui 2

Exemplu:

$$
\begin{aligned}
1011001_2 = \quad & 1\cdot 2^0 + \\
& 0\cdot 2^1 + \\
& 0\cdot 2^2 + \\
& 1\cdot 2^3 + \\
& 1\cdot 2^4 + \\
& 0\cdot 2^5 + \\
& 1\cdot 2^6 + \\
= {} & 1 + 8 + 16 + 64 = 89_{10}.
\end{aligned}
$$

2. 3 **Sistemul octal**

Z = {0,1,2,3,4,5,6,7}; B=8

- Valoarea fiecărei poziții la **stânga** unei cifre **creşte** cu o putere a lui 8
- Valoarea fiecarei poziții la **dreapta** unei cifre **scade** cu o putere a lui 8

Exemplu:

$$12345_8 = 5 \cdot 8^0 + 4 \cdot 8^1 + 3 \cdot 8^2 + 2 \cdot 8^3 + 1 \cdot 8^4$$

$$= 5 + 32 + 192 + 1024 + 4096 = 5349_{10}.$$

2. 4 **Sistemul hexazecimal**

Z = {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}; B=16

- Valoarea fiecărei poziții la **stânga** unei cifre **creşte** cu o putere a lui 16
- Valoarea fiecărei poziții la **dreapta** unei cifre **scade** cu o putere a lui 16

Exemplu:

$$1234B_{16} = 11\cdot16^0 + 4\cdot16^1 + 3\cdot16^2 + 2\cdot16^3 + 1\cdot16^4$$

$$= 11 + 64 + 768 + 8192 + 65536 = 74571_{10}.$$

2.5 Puteri ale lui 2

N	2^N	N	2^N
0	1	17	131,072
1	2	18	262,144
2	4	19	524,288
3	8	20	1,048,576
4	16	21	2,097,152
5	32	22	4,194,304
6	64	23	8,388,608
7	128	24	16,777,216
8	256	25	33,554,432
9	512	26	67,108,864
10	1,024	27	134,217,728
11	2,048	28	268,435,456
12	4,096	29	536,870,912
13	8,192	30	1,073,741,824
14	16,384	31	2,147,483,648
15	32,768	32	4,294,967,296
16	65,536	33	8,589,934,592

3. Conversii între sisteme de numerație

3.1 Conversia numerelor naturale

Conversia din baza 10 în baza B

- Metoda împărțirii iterative, pentru a converti N_{10} în N_B:

1. Se împarte N_{10} la B => câtul Q şi restul R
   - Q este noul număr de împărțit la B.
   - R este cifra cea mai puțin semnificativă a numărului N_B.
2. Dacă $Q = 0$, atunci STOP. Altfel $N_{10} = Q$ şi sari la pasul 1.

Exemplu:

$1020_{10} = ?_8$; $N_{10} = 1020$, $B = 8$

Pasul 1	1020 : 8	= 127 rest 4
Pasul 2	127 : 8	= 15 rest 7
Pasul 3	15 : 8	= 1 rest 7
Pasul 4	1 : 8	= 0 rest 1 => STOP

=> $1020_{10} = 1774_8$

Conversia din baza *B* în baza 10

- Metoda *înmulțirii iterative*, pentru a converti N_B în N_{10}:
  Se adună "contribuția" fiecărei cifre

Exemplu:

$2F1_{16} = ?_{10}$; $N_{16} = 2F1,\ B = 10$

$$\begin{aligned} 2F1_{16} = \;& 1\cdot 16^0 + \\ & 15\cdot 16^1 + \\ & 2\cdot 16^2 \\ = \;& 1 + 240 + 512 = 753_{10}. \end{aligned}$$

=> $2F1_{16} = 753_{10}$

Conversia numerelor din baza 2^n în numere din baza 2^m

- conversia 2^n => 2^1

n = 3 octal => binar: se înlocuieşte fiecare cifră octală cu cele trei cifre binare corespunzătoare

Exemplu:

$471_8 = 100\ 111\ 001_2$

n = 4 hexazecimal => binar: se înlocuieşte fiecare cifră hexazecimală cu cele patru cifre binare corespunzătoare

Exemplu:

$4AE_{16} = 0100\ 1010\ 1110_2$

Conversia numerelor din baza 2^n în numere din baza 2^m

- conversia 2^1 => 2^n

n = 3 binar => octal: se înlocuieşte fiecare grupă de trei cifre binare cu cifra octală corespunzătoare

Exemplu:

$$100111001_2 = \underset{4}{\underset{\downarrow}{100}}\ \underset{7}{\underset{\downarrow}{111}}\ \underset{1_8}{\underset{\downarrow}{001_2}}$$

n = 4 binar => hexazecimal: se înlocuieşte fiecare grupă de patru cifre binare cu cifra hexazecimală corespunzătoare

Exemplu:

$$10010101110_2 = \underset{4}{\underset{\downarrow}{0100}}\ \underset{A}{\underset{\downarrow}{1010}}\ \underset{E_{16}}{\underset{\downarrow}{1110_2}}$$

Conversia numerelor din baza 2^n în numere din baza 2^m

- se va urma şirul de conversii $2^n \Rightarrow 2^1 \Rightarrow 2^m$

Exemplu: octal => hexazecimal

$$25671_8 = 2 \quad 5 \quad 6 \quad 7 \quad 1_8$$

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

octal => binar $010\ 101\ 110\ 111\ 001_2$

rearanjare nr. binar $0010\ 1011\ 1011\ 1001_2$

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

binar => hexazecimal $2 \quad B \quad B \quad 9_{16}$

3.2 Conversia numerelor raționale

- spre deosebire de cazul conversiei numerelor întregi, conversia numerelor raționale între sisteme de numerație distincte nu este întotdeauna posibilă **în mod exact.**
- în general vom căuta reprezentări de tipul $R_{10}=R_B+\epsilon$, cu ϵ “suficient de mic”.

Conversia din baza B în baza 10

Exemplu:

$$0.1011_2 = 1\cdot2^{-1} + 0\cdot2^{-2} + 1\cdot2^{-3} + 1\cdot2^{-4} = 0.6875_{10}$$

3.2 Conversia numerelor raționale

Conversia din baza 10 în baza B

Exemplu:

$0.18_{10} = ?_2$, cu $k = 9$ biți precizie

Pas	**N**	**Operația**	**R**	z_{-i}
1	0.18	0.18·2=0.36	0.36	0
2	0.36	0.36·2=0.72	0.72	0
3	0.72	0.72·2=1.44	0.44	1
4	0.44	0.44·2=0.88	0.88	0
5	0.88	0.88·2=1.76	0.76	1
6	0.76	0.76·2=1.52	0.52	1
7	0.52	0.52·2=1.04	0.04	1
8	0.04	0.04·2=0.08	0.08	0
9	0.08	0.08·2=0.16	0.16	0

4. Logica binară

- **ŞI logic (x ∧ y)**

AND	0	1
0	0	0
1	0	1

- **SAU logic (x ∨ y)**

OR	0	1
0	0	1
1	1	1

- **negarea logică (⌉x)**

x	**⌉x**
0	1
1	0

- **SAU exclusiv (XOR)**

XOR	0	1
0	0	1
1	1	0

- **ŞI negat (NAND)**

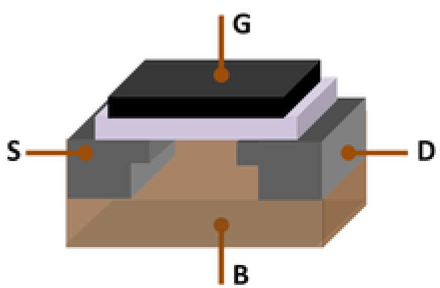
NAND	0	1
0	1	1
1	1	0

Exemple

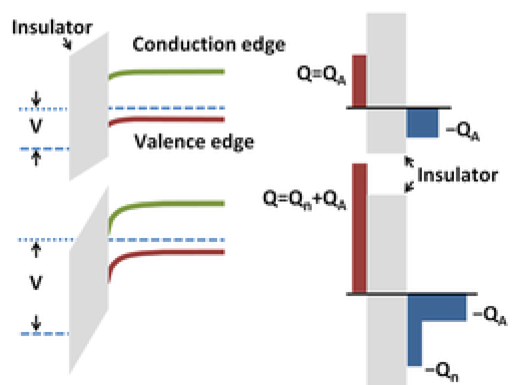
	11001010 01000111 11110000
AND	00110101 01110010 10101010
	00000000 01000010 10100000
	11001010 01000111 11110000
NAND	00110101 01110010 10101010
	11111111 10111101 01011111
	11001010 01000111 11110000
OR	00110101 01110010 10101010
	11111111 01110111 11111010
	11001010 01000111 11110000
XOR	00110101 01110010 10101010
	11111111 00110101 01011010

Logica binară – implementare în tehnologia CMOS

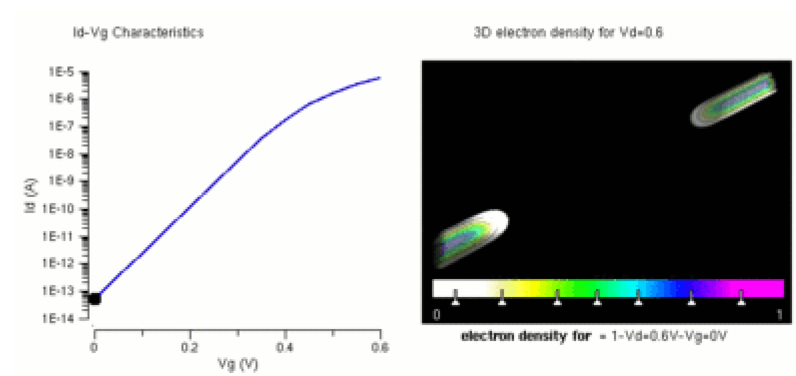
- **MOSFET – Structură şi mod de operare**
  - Structura metal–oxid–semiconductor (MOS) clasică se obţine prin creşterea unui strat de SiO_2 (dielectricul de poartă) la suprafaţa unui substrat de siliciu, peste care se depune un electrod metalic (electrodul de poartă).
  - Tranzistoarele cu canal de tip n se numesc n-FET, cele cu canal p se numesc p-FET.



Structura unui MOSFET [1]



Formarea canalului într-un n-FET: Aplicarea unei tensiuni pozitive pe poartă conduce la apariţia unei regiuni sărăcite în goluri la interfaţa SiO_2/p-Si. Creşterea V_G induce crearea unui strat de inversie (canalul de conducţie). [1]

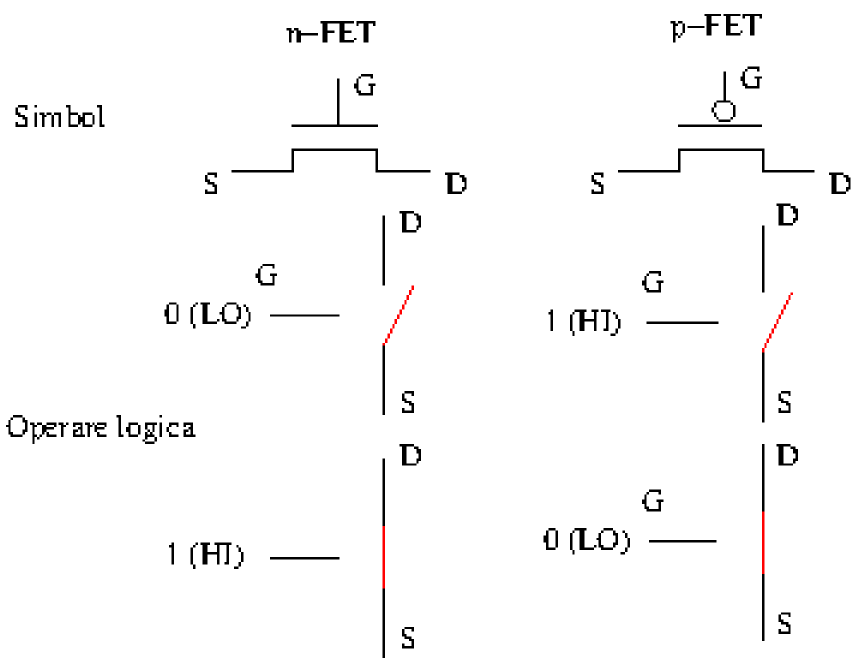


Simulare numerică a formării Canalului de inversie într-un MOSFET 1D (bazat pe un nanofir). [1]

[1] http://en.wikipedia.org/wiki/MOSFET

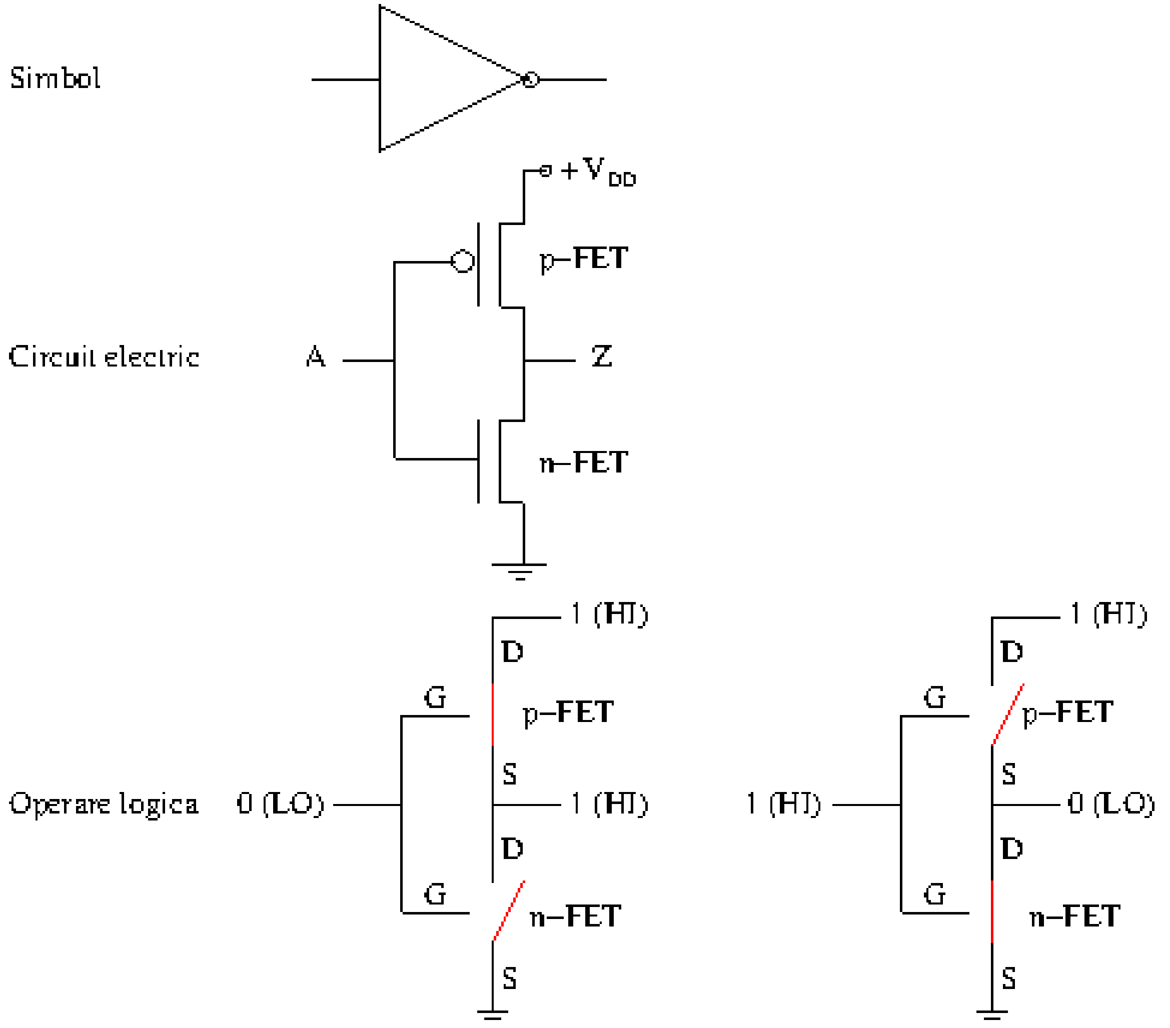
Logica binară – implementare în tehnologia CMOS

- **MOSFET - Model de operare simplificat**
  - Modul de operare al circuitelor logice CMOS nu depinde de detaliile comportamentului electrical tranzistorilor
  - Model simplificat: tranzistoarele n-FET sau p-FETacționează ca nişte comutatori electronici



MOSFET – Circuite logice

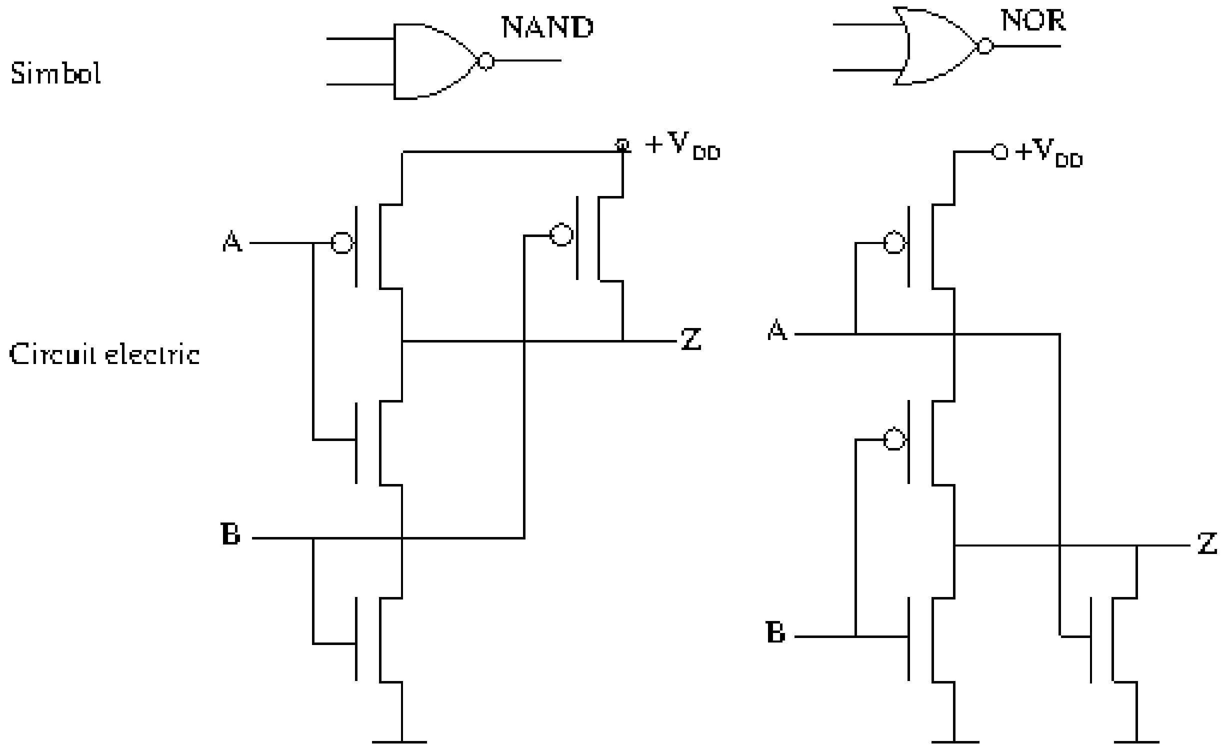
- **Exemplul 1: circuit de negare** – intrarea A controlează simultan elementele n-FET şi p-FET, deci unul va fi deschis (ON), iar celălalt închis (OFF), în funcţie de valoarea A.



A	Z
0	1
1	0

MOSFET – Circuite logice

- **Exemplul 2 : porţile logice NAND şi NOR**
- Toate funcţiile logice booleene pot fi construite doar cu porţi NAND şi NOR. În principiu este posibil ca un întreg procesor să fie construit doar cu aceste două tipuri de porţi (deşi în practică poate să nu fie soluţia optimă).

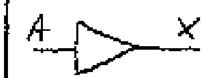
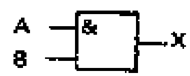
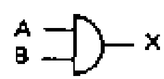
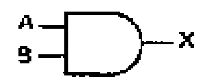
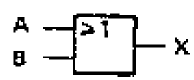
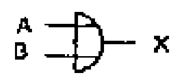
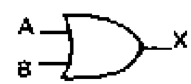
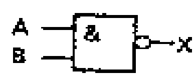
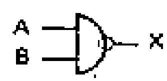
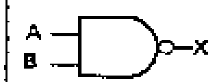
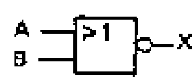
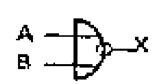
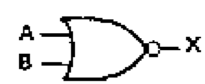
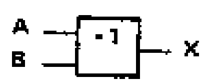
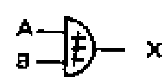
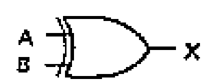
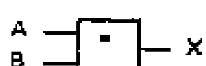
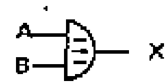
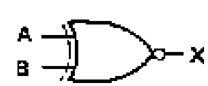


Operare logica

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

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

Porți logice

	Circuit	IEC norm	DIN norm 40700	American standard	Boolean function
7407	Buffer				$X = A$
7404	NOT (Inverter)				$X = \overline{A}$
7408	AND				$X = A \cdot B$
7432	OR				$X = A + B$
7400	NAND				$X = \overline{A \cdot B}$
7402	NOR				$X = \overline{A + B}$
7486	Exclusive OR XOR				$X = A\overline{B} + \overline{A} \cdot B$ $A \oplus B$
	Comparator				$X = A\,B + \overline{A}\,\overline{B}$ $A \equiv B$

5. Aritmetica binară – numere întregi

Reguli de bază pentru adunare, scădere şi înmulţire

	Operaţie	**Rezultat**	**Bitul de transport ("carry bit")**
Adunare	0+0 0+1 1+0 1+1	0 1 1 0	0 0 0 1
Scădere	0-0 1-0 0-1 1-1	0 1 1 0	0 0 1 0
Înmulţire	0·0 1·0 0·1 1·1	0 0 0 1	0 0 0 0

Exemple

Adunare

```
          1010
+         1011
------------------
carry:    1 1
+       1 1
------------------
         10101
```

$$10_{10} + 11_{10} = 21_{10}$$

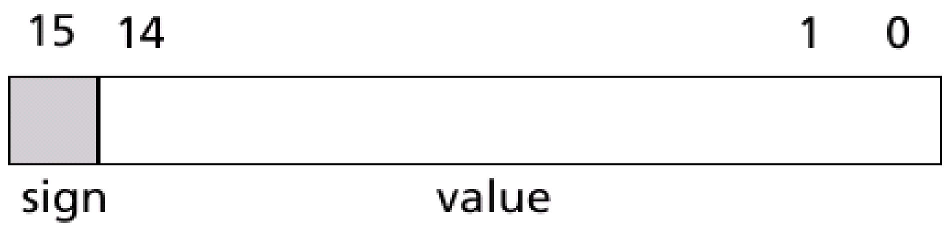
Scădere

```
          1101
-         1010
------------------
carry:      1
-          1
------------------
          0011
```

$$13_{10} - 10_{10} = 3_{10}$$

6. Numere negative

Numere întregi cu semn: o reprezentare posibilă:



- Sign = 0 => număr pozitiv
- sign = 1 => număr negativ

Domeniul de variație al numereleor reprezentate pe n biți:

$$-2^{n-1}+1 \,..\, 2^{n-1}-1$$

Exemplu (1): n=3

Bit de semn	**Valoare**	**Număr**
0	00	0
0	01	1
0	10	2
0	11	3
1	00	**??**
1	01	-1
1	10	-2
1	11	-3

Zero “negativ”??

Domeniu de variație: $-2^{3-1}+1 \;..\; 2^{3-1}-1 = -3 \;..\; +3$

Probleme legate de reprezentarea numerelor negative...

Operații simple cu două numere, unul pozitiv, celălalt negativ:

```
    0  0 1 1 0 0  (base 2)            12      (base 10)
+   1  0 1 1 1 1                     -15
 ------------------               ---------
    1  1 1 0 1 1  (= -27₁₀)           -3
```

"1-1": 1 - 1 = 1 + (-1) = 0 ?

```
    0   0 0 1                          0 0 1
+   1   0 0 1                   -      0 0 1
 ----------------                -----------
    1   0 1 0  (= -2₁₀)                0 0 0
```

Problemă!!

Soluția: reprezentare în complement la 2

Complement la 1 (complement la B-1)

Reprezentăm numerele negative prin **negarea logică** a reprezentării numerelor **pozitive** (şi reciproc):

- fiecare 0 devine 1;
- fiecare 1 devine 0.

Valoare binară	Valoare zecimală	Complement la 1	Valoare zecimală
0 1001	+9	1 0110	-9
1 1001	-6	0 0110	+6
0 0000	0	1 1111	-15 (!)
0 1111	+15	1 0000	??

Problemă!!

Complement la 2 (complement la B)

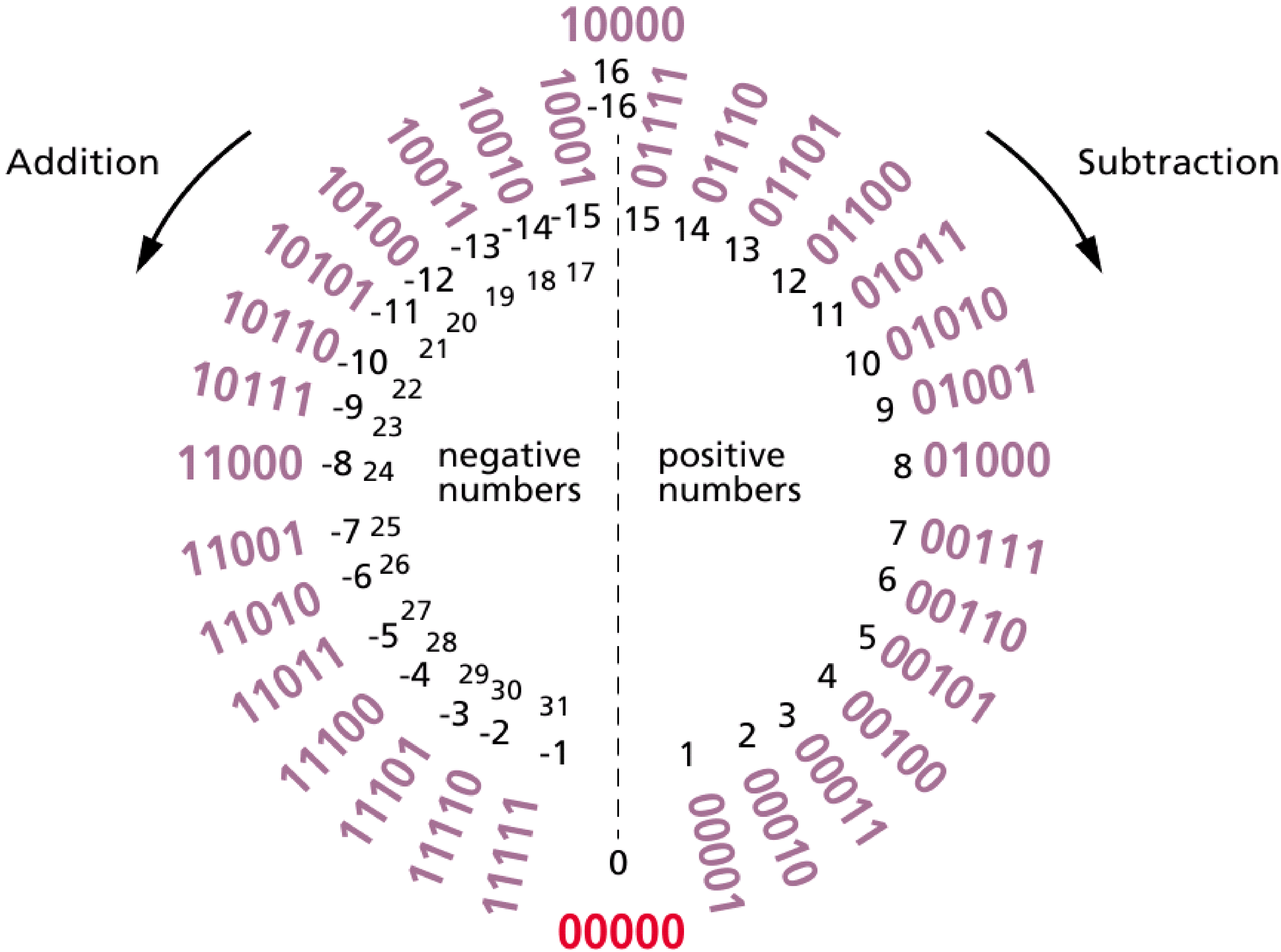
Numerele pozitive sunt reprezentate în modul obişnuit. Numerele negative sunt reprezentate prin valoarea obţinuta astfel:prin **negarea logică** a reprezentării numerelor **pozitive** (şi reciproc):

- se **neagă logică** valorea **pozitivă**;
- se **adaugă** 1.

Valoare binară	Valoare zecimală	Complement la 1	Valoare zecimală
0 1001	+9	1 0110 + 1 = 1 0111	-9
1 1001	-7	⌉(1 1001 – 1) = ⌉(1 1000) = 0 0111	+7
0 0000	0	1 1111 + 1 = 0 000	0
0 1111	+15	1 0000 + 1 = 1 0001	-15
1 1111	-1	⌉(1 1111 – 1) = ⌉(1 1110) = 0 0001	+1

În această reprezentare, scăderile pot fi efectuate ca simple adunări cu numărul negativ corespunzător!!

Inelul numerelor reprezentate pe 5 biți, în complement la 2



6. Numere în virgulă mobilă

Mantisa şi exponentul

Numere în reprezentarea ştiinţifică obişnuită:

- 3.14159265358979323846264338327950288 4197
- 7.5×10^{-3}

De obicei un computer afişează aceste numere sub forma:

- 0.31415e1
- 0.75e-2

Acest tip de reprezentare (**mantisă + exponent**) economiseşte spaţiul de stocare şi reduce redundanţa.

Forma de reprezentare obişnuită pentru un număr în virgulă mobilă, pe 32 de biţi (float) este:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
s	exponent								fraction																						
1 bit	8 bits								23 bits																						

Aritmetica numerelor în virgulă mobilă

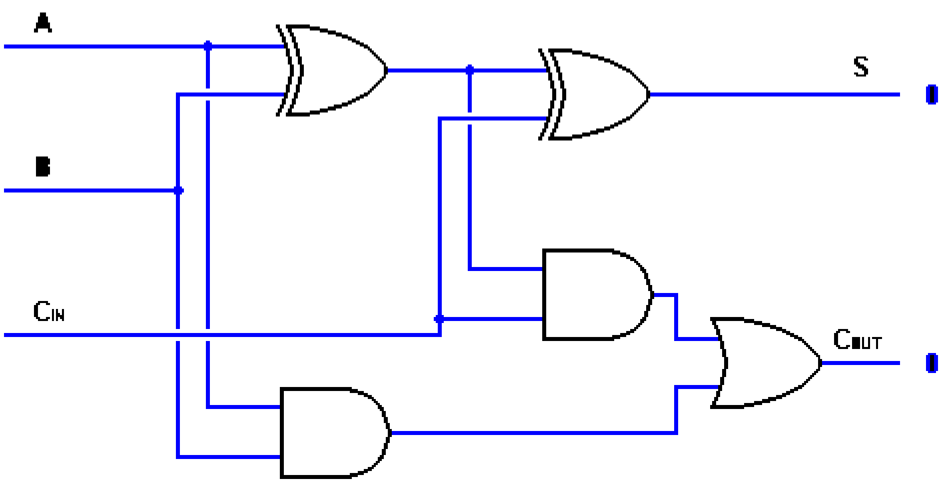
Exemplu: $0.270 \cdot 10^4 + 0.127 \cdot 10^2$

Operanzi	Compară exponenții	Ajustează exponenți + mantise
$0.270 \cdot 10^4$ $0.127 \cdot 10^2$	$e1 - e2 = 2$	$0.270 \cdot 10^4$ $+ 0.001 \cdot 10^4$
		Rezultat: $0.271 \cdot 10^4$

Algoritm adunare:

1. identifică mantise şi exponenți;
2. compară exponenții;
3. ajustează exponenții, dacă este necesar;

Aritmetica binară: circuit logic de adunare pe 2 biți



Input bit for number A	Input bit for number B	Carry bit input C_{IN}	Sum bit output S	Carry bit output C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Algoritm înmulțire

1. multiplică mantisele
2. normalizează mantisa rezultatului
3. ajustează exponenții
4. adună exponenții

Exemplu: $(0.270\cdot10^{4}) \cdot (0.127\cdot10^{2})$

$0.270 \cdot 0.127 =$

$0.270\cdot10^{-1} +$
$0.540\cdot10^{-2} +$
$1.890\cdot10^{-3}$

$0.270\cdot10^{-1} \longrightarrow 0.270\cdot10^{-1}$

$0.540\cdot10^{-2} \longrightarrow 0.054\cdot10^{-1}$

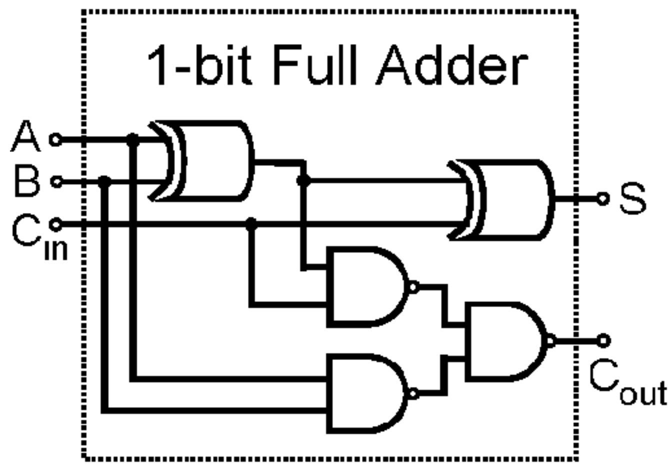
$0.270\cdot10^{-1} + 0.054\cdot10^{-1} \longrightarrow 0.324\cdot10^{-1}$

$+\ 0.189\cdot10^{-2} \longrightarrow 0.018\cdot10^{-1}$

$0.324\cdot10^{-1} + 0.018\cdot10^{-1} \longrightarrow 0.342\cdot10^{-1}$

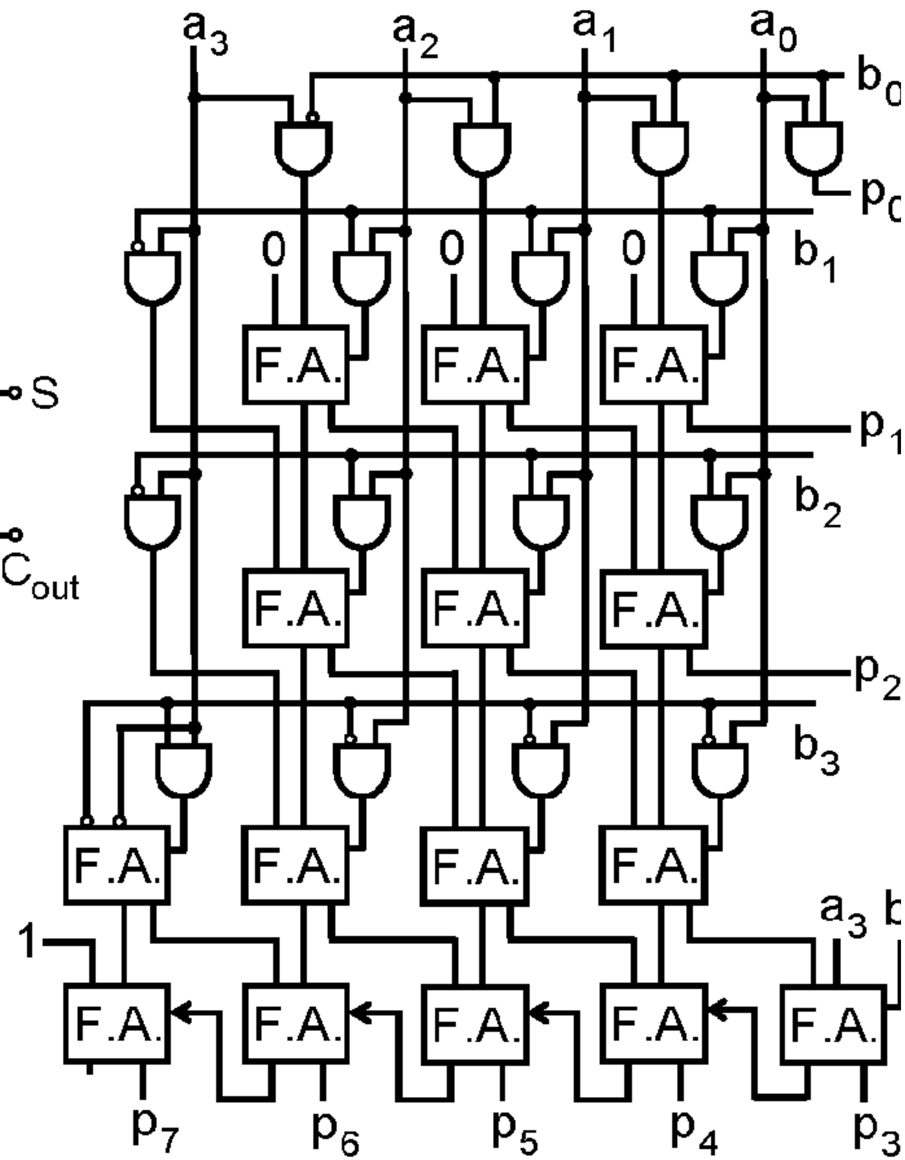
- Adună exponenții: $4 + 2 + (-1) = 5$
- Rezultat: $0.342\cdot10^{5}$

Un exemplu de circuit multiplicator pe 4 biți



(a)

4-bit digital multiplier



(b)