Arhitectura Calculatoarelor

Curs 1

- performance improvements

BIBLIOGRAFIE:

① J. L. Hennesy, D. A. Patterson: „Computer Architecture: A Quantitative Approach", 6th Ed., Morgan Kaufma, 2017

② M. Vlăduțiu: "Computer Arithmetics: Algorit HW Design and Implementation", Springer, 2012

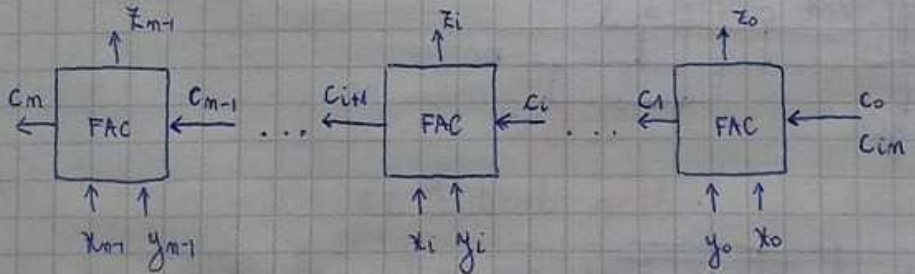
③ D.A. Patterson, J.L. Hennesy: "Computer Organization and Design, MIPS Ed. The HW/SW Interface", Morgan Kaufma, 5th Ed, 2013

④ R.E. Bryant, D.R. O'Hallaron: "Computer Systems: A Programmer's Perspective", 3rd Ed., Pearson, 2015.

CAP 1: ANALIZA FUNCȚIONALĂ ȘI SINTEZA DISPOZITIVELOR BINARE ȘI ZECIMALE DE ADUNARE ȘI SCĂDERE

1.1. Sumatoare paralele cu propagare serială a transportului Ripple Carry Adder

- Full Adder Cell (FAC)



Inputs			Outputs	
x_i	y_i	c_i	z_i	c_{i+1}
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	1	0	0	1
1	1	1	1	1
1	0	1	0	1

$$z_i = \overline{x_i}\,\overline{y_i}\,c_i + \overline{x_i}\,y_i\,\overline{c_i} + x_i\,\overline{y_i}\,\overline{c_i} + x_i\,y_i\,c_i =$$
$$= \overline{x_i}(\overline{y_i}c_i + y_i\overline{c_i}) + x_i(\overline{y_i}\overline{c_i} + y_i c_i)$$
$$= \overline{x_i}(y_i \oplus c_i) + x_i(\overline{y_i \oplus c_i})$$
$$= x_i \oplus y_i \oplus c_i$$

$C_{i+1} = \bar{x}_i y_i c_i + x_i \bar{y}_i c_i + x_i y_i \bar{c}_i + (x_i y_i c_i) \times 3$

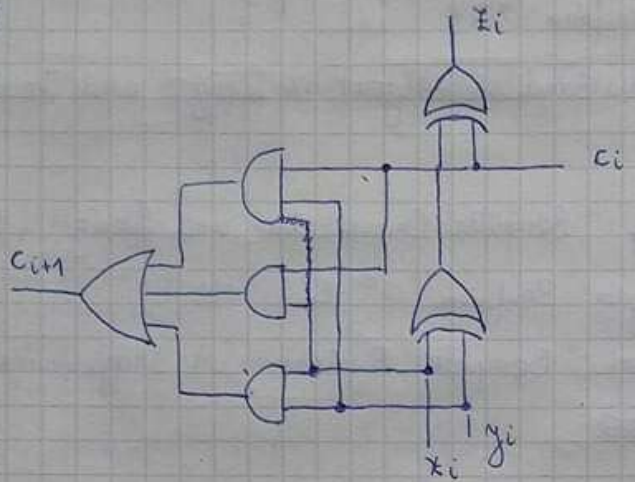
$= \bar{x}_i y_i c_i + x_i y_i c_i +$

$x_i \bar{y}_i c_i + x_i y_i c_i +$

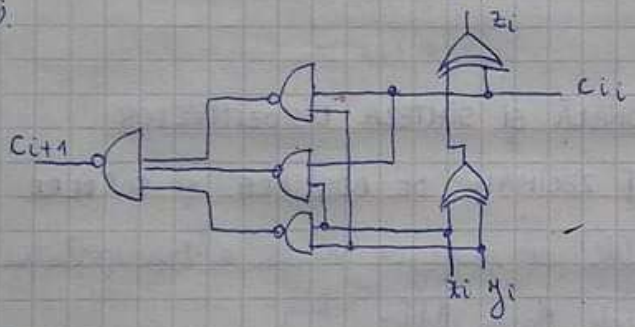
$x_i y_i \bar{c}_i + x_i y_i c_i$

$C_{i+1} = x_i y_i + x_i c_i + y_i c_i$

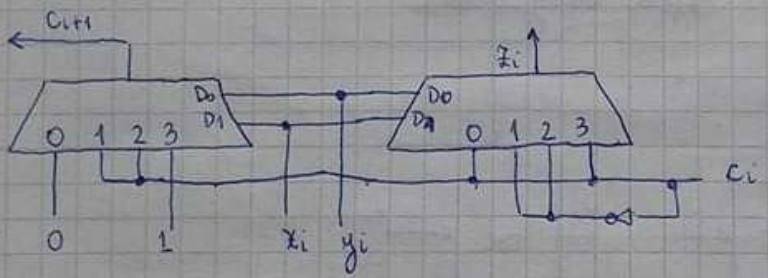
Ⓐ



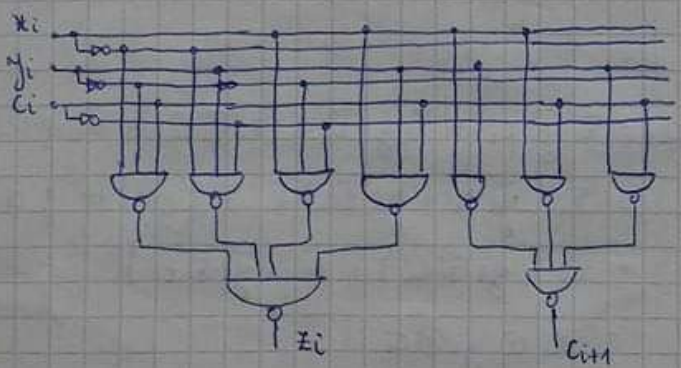
Ⓑ



Ⓒ



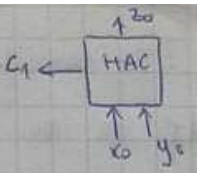
Ⓓ



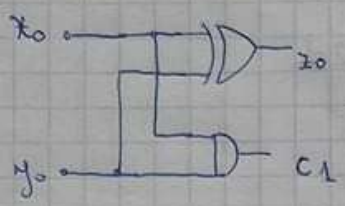
* $c_0 = 0$ $\quad z_0 = x_0 \oplus y_0 \oplus 0$

$\quad c_1 = x_0 \cdot 0 + y_0 \cdot 0 + x_0 y_0$

$\Big\}$ HAC: $\begin{cases} z_0 = x_0 \oplus y_0 \\ c_1 = x_0 y_0 \end{cases}$

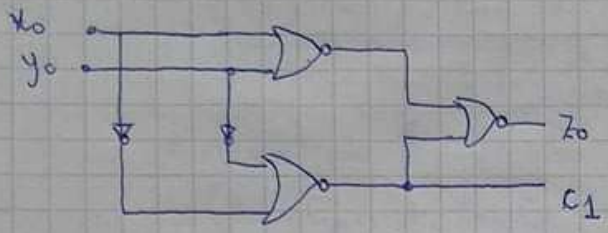


Half adder cell

(A') 

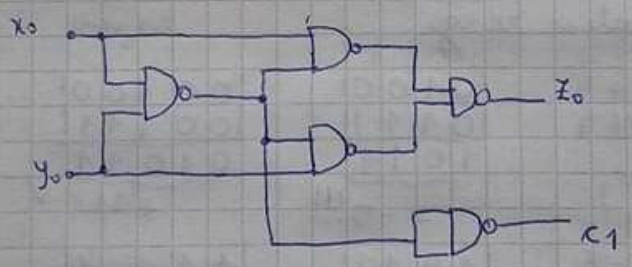
(B') $z_0 = x_0 \oplus y_0 = \bar{x}_0 y_0 + x_0 \bar{y}_0 = \overline{\overline{(x_0 + y_0)(\bar{x}_0 + \bar{y}_0)}} = \overline{\overline{x_0 + y_0}} + \overline{\overline{\bar{x}_0 + \bar{y}_0}}$

$c_1 = \overline{\overline{x_0 y_0}} = \overline{\overline{x_0} + \overline{y_0}}$



(C') $z_0 = (x_0 + y_0)(\bar{x}_0 + \bar{y}_0) = \overline{\overline{x_0(\bar{x}_0 + \bar{y}_0) + y_0(\bar{x}_0 + \bar{y}_0)}} = \overline{\overline{x_0\,\overline{x_0 y_0}} \cdot \overline{y_0\,\overline{x_0 y_0}}}$

$c_1 = x_0 y_0$

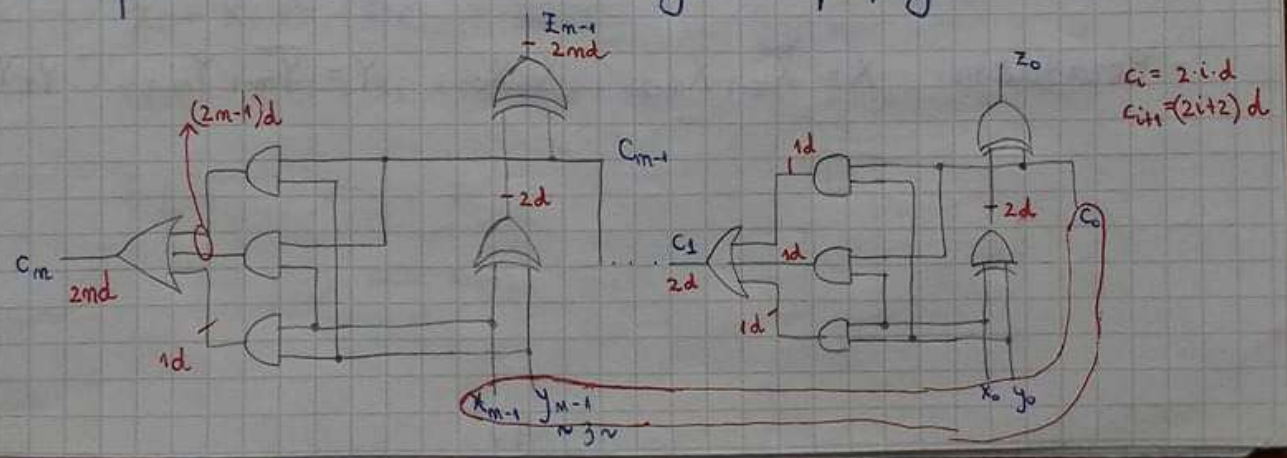


S2: 3.10.2018

CURS 2

1.1. Critical path

↳ se referă la calea cea mai lungă de propagare a semnalelor



Obs: ! Se consideră că orice poartă primitivă are o întârziere de 1d (indif. de nr. de intrări).

! Invertoarele au întârzierea 0d.

! O poartă sau-exclusiv are întârzierea 2d.

! Toți biții de intrare sunt plasați la 0d.

↳ Întârzierea: $D_{RCA}^{Z/cout.} = 2nd$

Condițiile speciale ale operației de adunare

- zero
- negativ
- Carry out from msb
- overflow

↳ exemplu: $X = +9$ | 1001 | 01001
$Y = +12$ | 1100 | 01100
• unsigned: | 0101 | 10101

Overflow-ul la nr. fără semn apare at. când se generează carry din cel mai semnificativ rang.

• signed: * $X = +4$ | 0100 | 00100
$Y = +7$ | 0111 | 00111
1011 | 01011
$= -5$!!! | $+11$ ✓

* $X = -5$ | 1011 | 1 1011
$Y = -6$ | 1010 | 11010
✗ 0101 | ✗ 10101
$= +5$!! | -11

Overflow apare când adunăm 2 nr. de același semn și rezultatul are semn contrar.

• Considerăm: $X = \overline{X}_{n-1} X_{n-2} \dots X_1 X_0$ $Y = \overline{Y}_{n-1} Y_{n-2} \dots Y_1 Y_0$

Input			Output	
x_{n-1}	y_{n-1}	c_{n-1}	z_{n-1}	v
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	0
1	0	1	0	0
1	1	0	0	1
1	1	1	1	0

$$v = \overline{x_{n-1}}\,\overline{y_{n-1}}\,c_{n-1} + x_{n-1}\,y_{n-1}\,\overline{c_{n-1}} \overset{I_2}{=}$$

I_1: $(A \oplus B)C = AC \oplus BC$

$A + B = A \oplus B \oplus AB$

I_2': $A \oplus B = (A+B) \oplus AB$

$\overline{c_{n-1}} = c_{n-1} \oplus 1$

$$= \overline{x_{n-1}}\,\overline{y_{n-1}}\,c_{n-1} \oplus x_{n-1}\,y_{n-1}\,\overline{c_{n-1}} \overset{I_1}{=}$$

$$= \overline{x_{n-1}}\,\overline{y_{n-1}}\,c_{n-1} \oplus x_{n-1}\,y_{n-1}\,c_{n-1} \oplus x_{n-1}\,y_{n-1} \overset{I_1}{=}$$

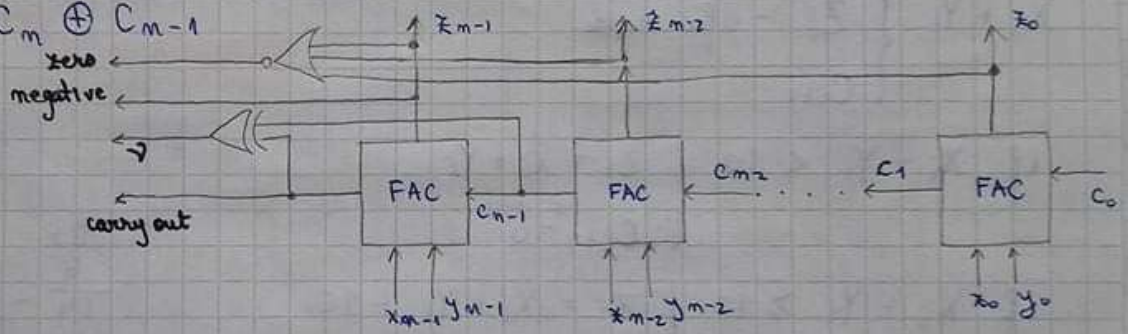
$$= (\overline{x_{n-1}}\,\overline{y_{n-1}} \oplus x_{n-1}\,y_{n-1})\,c_{n-1} \oplus x_{n-1}\,y_{n-1} \overset{I_2'}{=}$$

$$= (\underbrace{\overline{x_{n-1}}\,\overline{y_{n-1}} + x_{n-1}\,y_{n-1}}_{x_{n-1} \oplus y_{n-1} \oplus 1})\,c_{n-1} \oplus x_{n-1}\,y_{n-1} \overset{I_1}{=}$$

$$= x_{n-1}\,c_{n-1} \oplus y_{n-1}\,c_{n-1} \oplus c_{n-1} \oplus x_{n-1}\,y_{n-1} \overset{I_2'}{=}$$

$$= (x_{n-1}\,c_{n-1} + y_{n-1}\,c_{n-1} + x_{n-1}\,y_{n-1}) \oplus c_{n-1} =$$

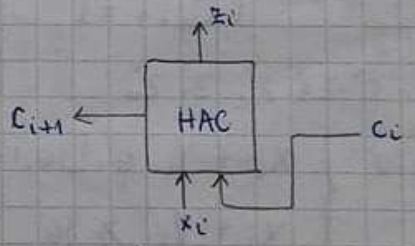
$$v = c_n \oplus c_{n-1}$$



$X = x_{n-1}\,x_{n-2} \ldots x_1\,x_0$

$Y = y_{n-1}\,y_{n-2} \ldots y_1\,y_0$

Y - constantă impară

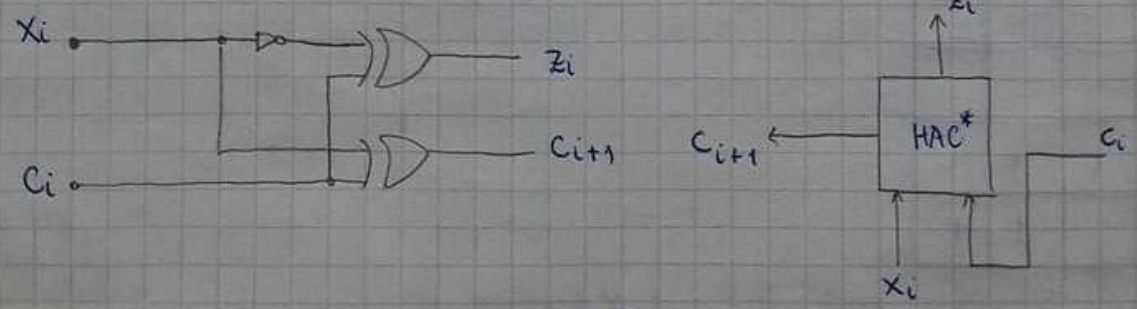


$y_i = 0 \Rightarrow z_i = x_i \oplus \overset{=0}{y_i} \oplus c_i = x_i \oplus c_i$

$c_{i+1} = x_i c_i + 0(x_i + c_i) = x_i c_i$ } HAC

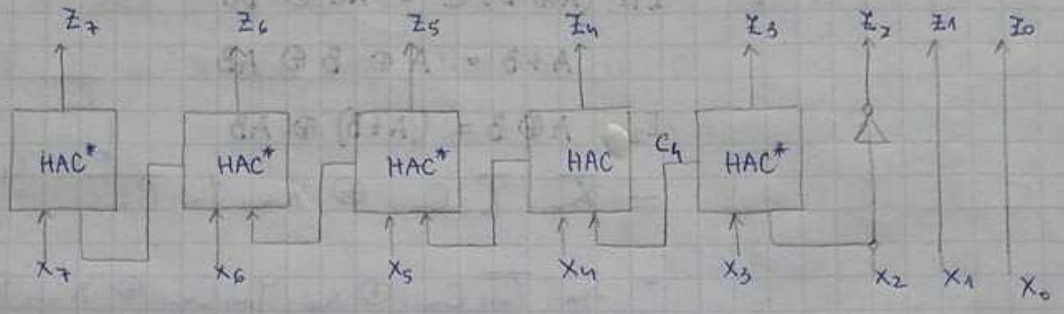
$y_i = 1 \Rightarrow z_i = x_i \oplus 1 \oplus c_i = \overline{x_i} \oplus c_i$

$c_{i+1} = x_i c_i + x_i + c_i = x_i + c_i$ } HAC*



Exemplu: $X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0$ +

1 1 1 0 1 1 0 0



1.2. Sumatoare zecimale cu propagarea serială a transportului

1.2.1. B.C.D.

$X_i = X_3^{2^3} X_2^{2^2} X_1^{2^1} X_0^{2^0}$ $\quad Y_i = Y_3 Y_2 Y_1 Y_0$ $\quad Z = Z_3 Z_2 Z_1 Z_0$

$X_i + Y_i \begin{cases} Z_i & \text{- afră sumă} \\ C_{i+1} \end{cases}$

if $X_i + Y_i < 10 \begin{cases} z_i = X_i + Y_i \\ C_{i+1} = 0 \end{cases}$

if $X_i + Y_i \geq 10 \begin{cases} z_i = X_i + Y_i - 10 \\ C_{i+1} = 1 \end{cases}$

$\begin{array}{r} 8+ \\ 6 \\ \hline 14 - \\ 10 \\ \hline 4 \end{array}$

$X_i + Y_i = C' z_3' z_2' z_1' z_0'$

$x_i + Y_i \geq 10 \begin{cases} X_i + Y_i \in [10,15] & C1 \\ x_i + Y_i \geq 16 & C_2 \end{cases}$

C_2 : $C' z_3' z_2' z_1' z_0' \geq 16 \equiv C' = 1$

$C1$: $10 \leq C' z_3' z_2' z_1' z_0' \leq 15$ $\quad \begin{cases} C' = 0 \\ 10 \leq z_3' z_2' z_1' z_0' \leq 15 \end{cases}$

$\bar{C}' (z_3' z_2' + z_3' z_1') = 1$

$z_3' z_2'$ \ $z_1' z_0'$	00	01	11	10
00				
01				
11	1	1	1	1
10			1	1

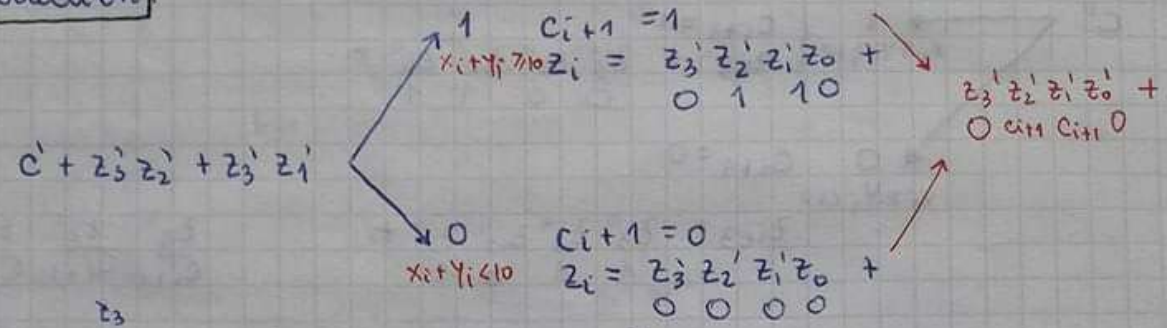
$C_1 \cup C_2$: $c' + \underbrace{\bar{c}'}_{\text{absorbed}}(z_3' z_2' + z_3' z_1') = 1 \Rightarrow c' + z_3' z_2' + z_3' z_1' = 1$

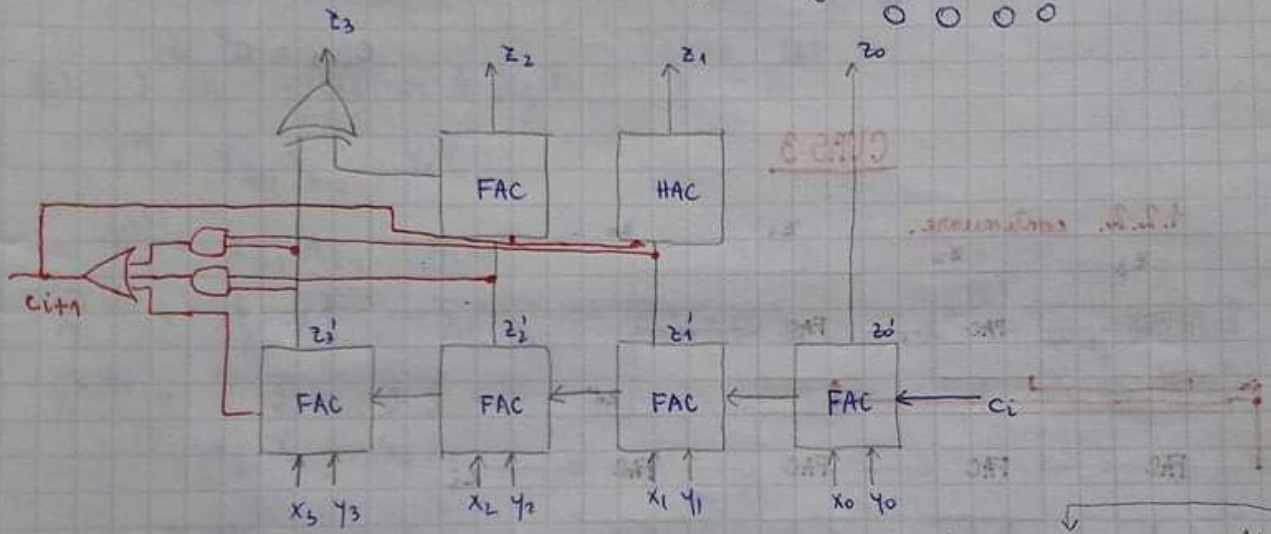
$(X_i + Y_i - 10) \bmod 2^4 = (X_i + Y_i - 10 + 16) \bmod 2^4 = (X_i + Y_i + 6) \bmod 2^4$

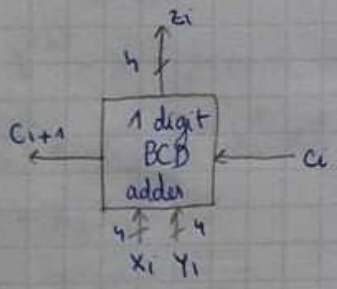
Scăderea lui 10 pe 4 cifre $\Leftrightarrow$ Adunarea cu 6

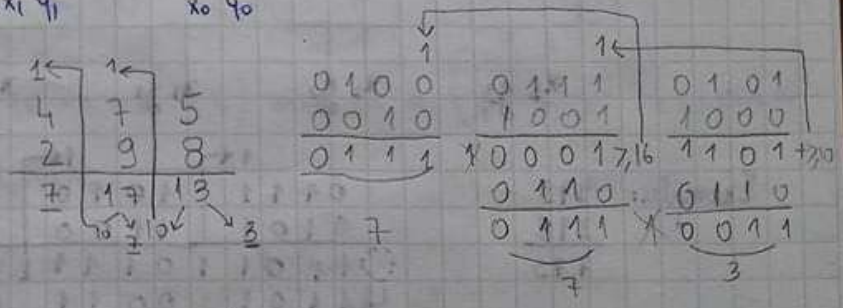
$(8 + 7 - 10) = 5 \qquad (8 + 7 + 6) \bmod 2^4 = 21 \bmod 2^4 = 5$

Z_i's correction:









1.2.2. Exces de 3

$X_{iE3} = X_3 X_2 X_1 X_0 \quad Y_{iE3} = Y_3 Y_2 Y_1 Y_0 \quad Z_{iE3} = Z_3 Z_2 Z_1 Z_0$

$X_{iE3} = X_i + 3 \quad Y_{iE3} = Y_i + 3 \quad Z_{iE3} = Z_i + 3$

$X_{iE3} + Y_{iE3} \begin{cases} Z_{iE3} \\ C_{i+1} \end{cases}$

if $X_i + Y_i < 10 \begin{cases} C_{i+1} = 0 \\ Z_i = X_i + Y_i \big|_{+6} \end{cases} \Rightarrow Z_{iE3} = X_{iE3} + Y_{iE3} - 3$

if $X_i + Y_i \geq 10$ $\begin{cases} C_{i+1} = 1 \\ Z_i = X_i + Y_i - 10 \end{cases}\Big|_{+6} \Rightarrow Z_{iE3} = X_{iE3} + Y_{iE3} \underbrace{-13}_{\equiv +3}$

$X_{iE3} + Y_{iE3} = C'' z_3'' z_2'' z_1'' z_0''$
$X_i + Y_i \geq 10 \Big|_{+6} \Rightarrow X_{iE3} + Y_{iE3} \geq 16$ $\Big\} C'' = 1$

C'' → 1 ($x_i + y_i \geq 10$): $\begin{cases} C_{i+1} = 1 \\ Z_{iE3} = z_3'' z_2'' z_1'' z_0'' + \\ \quad\quad 0\ 0\ 1\ 1 \end{cases}$

→ 0 ($x_i + y_i < 10$): $C_{i+1} = 0$

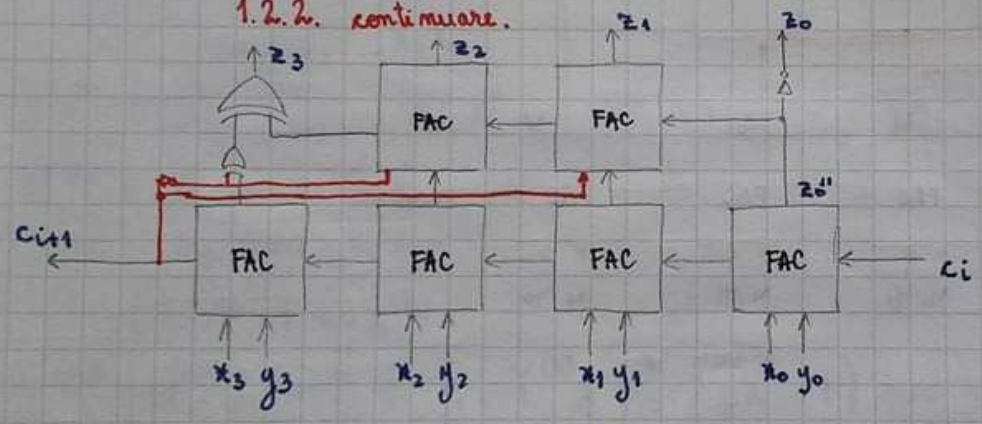
$Z_{iE3} = z_3'' z_2'' z_1'' z_0'' +$
$\quad 1\ 1\ 0\ 1$

$\underline{z_3''}\ \underline{z_2''}\ \underline{z_1''}\ \underline{z_0''} +$
$C_{i+1}\ C_{i+1}\ C_{i+1}\ 1$

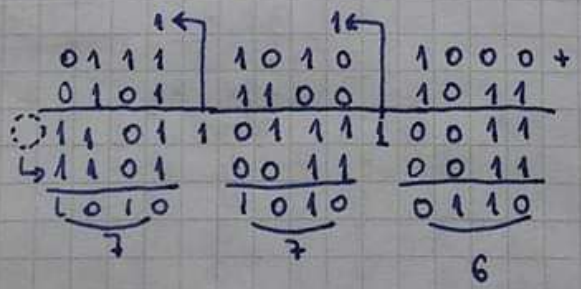
$C_{i+1} = C''$

CURS 3

1.2.2. continuare.



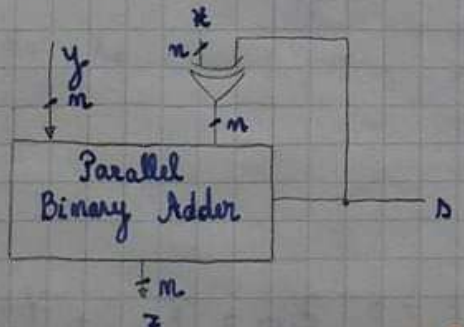
Exemplu: 475 + 298 = 773



1.3. Scăzătoare binare și zecimale

Ⓐ. Sumatorul scăzător

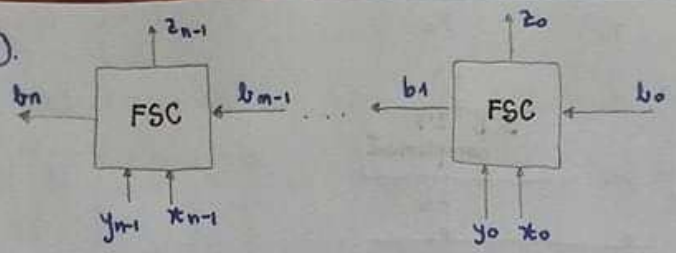
ⓐ.



D → 1: $Z = Y + \bar{X} + 1 = Y - X$

D → 0: $Z = Y + X$

b).

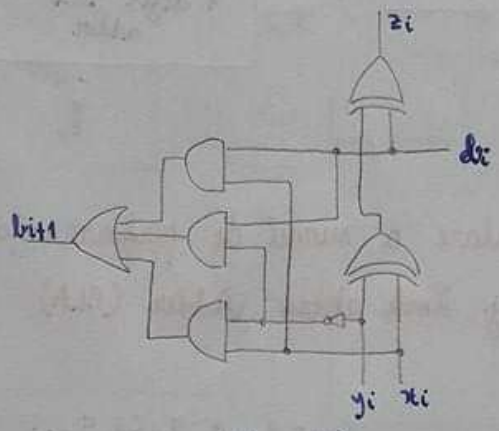


$y_i - x_i - b_i \begin{cases} z_i \\ b_{i+1} \end{cases}$

$z_i = y_i \oplus x_i \oplus b_i$

$b_{i+1} = \overline{y_i}\, x_i + \overline{y_i}\, b_i + x_i b_i$

Inputs			Outputs	
y_i	x_i	b_i	z_i	b_{i+1}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



c). $Y^{(k)}, X^{(k)}$ - BCD on k digits ; $Y^{(k)} - X^{(k)}$

$Y^{(k)} = Y_{k-1} Y_{k-2} \dots Y_1 Y_0$

$X^{(k)} = X_{k-1} X_{k-2} \dots X_1 X_0$

$\overline{X^*}^{(k)} = \overline{X^*_{k-1}}\, \overline{X^*_{k-2}} \dots \overline{X_1^*}\, \overline{X_0^*}$; $\overline{X_i^*} = 9 - X_i$; $\overline{X^*}^{(k)}$ - complementul de 9 al nr. X pe k cifre

$\overline{X^*}^{(k)} = \underbrace{9\ 9 \dots 9\ 9}_{k} - X_{k-1} X_{k-2} \dots X_1 X_0$

$= 10^k - 1 - X^{(k)}$

$(Y^{(k)} - X^{(k)}) \bmod 10^k = (Y^{(k)} + \overline{X^*}^{(k)} - 10^k + 1) \bmod 10^k = (Y^{(k)} + \overline{X^*}^{(k)} + 1) \bmod 10^k$

$\overline{X_i^*} = X_3^* X_2^* X_1^* X_0^* = 1001 - X_3 X_2 X_1 X_0$

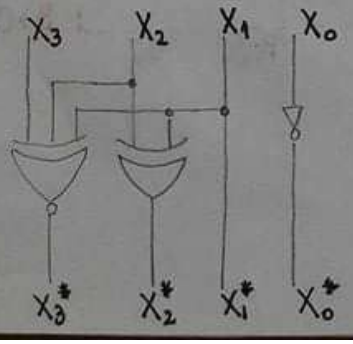
$X_i = X_3 X_2 X_1 X_0$

$X_3^* = \overline{X_3 + X_2 + X_1}$

$X_2^* = X_2 \oplus X_1$

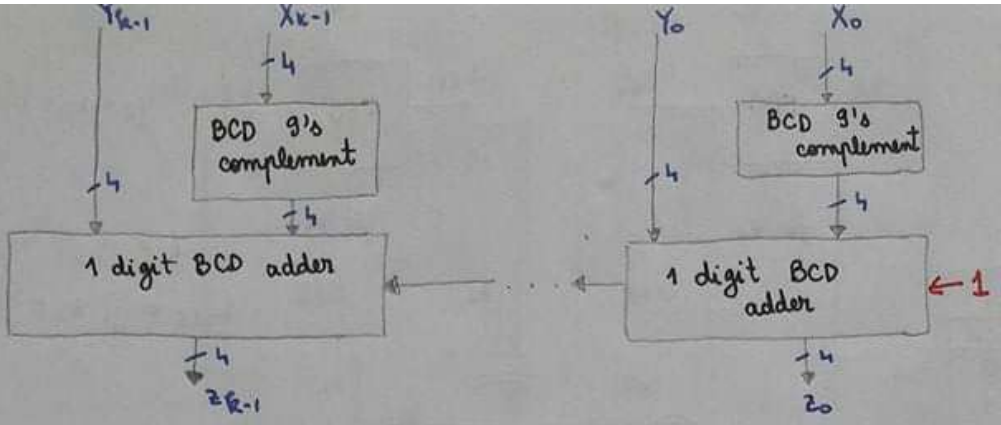
$X_1^* = X_1$

$X_0^* = \overline{X_0}$



Inputs				Outputs			
Y_3	Y_2	Y_1	Y_0	X_3^*	X_2^*	X_1^*	X_0^*
0	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	0	0	0	1	1
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	d	d	d	d
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	1	1	1	d	d	d	d

~g~

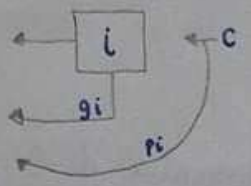


1.4. Metode de calculare a sumei de performanță ridicată

1.4.1. Multilevel Carry Look ahead Adder (CLA)

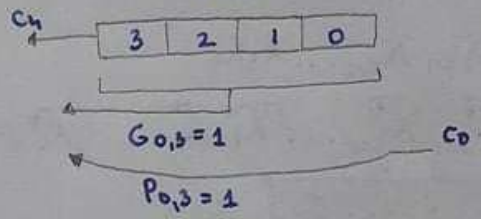
Full CLA ↗

$c_{i+1} = \underbrace{g_i}_{x_i \cdot y_i} + \underbrace{p_i}_{x_i + y_i} c_i$



$c_4 = g_3 + p_3 c_3 = \dots$

$= \underbrace{g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0}_{G_{0,3}} + \underbrace{p_3 p_2 p_1 p_0}_{P_{0,3}} \cdot c_0$



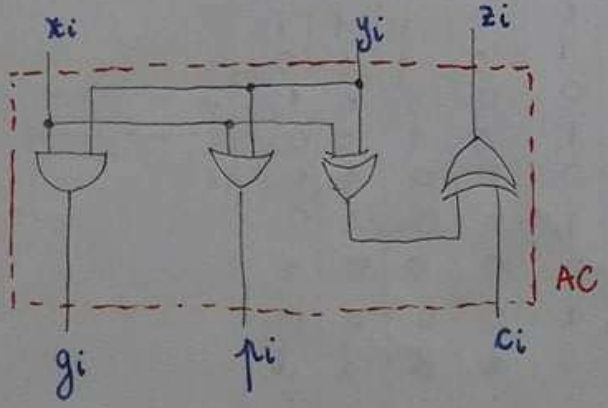
$= \underbrace{\underbrace{g_3 + p_3 g_2}_{G_{2,3}} + \underbrace{p_3 p_2}_{P_{2,3}} \underbrace{(g_1 + p_1 g_0)}_{G_{0,1}}}_{G_{0,3}} + \underbrace{\underbrace{p_3 p_2}_{P_{3,2}} \cdot \underbrace{p_1 p_0}_{P_{0,1}}}_{P_{0,3}} \cdot c_0$

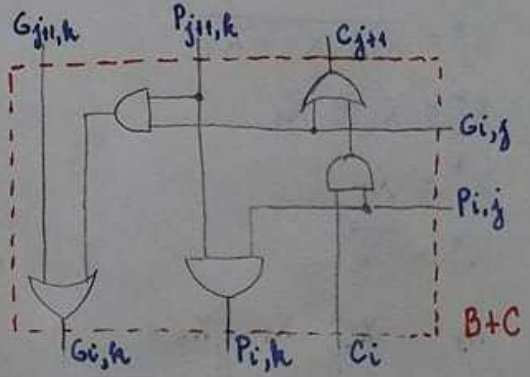
În general:

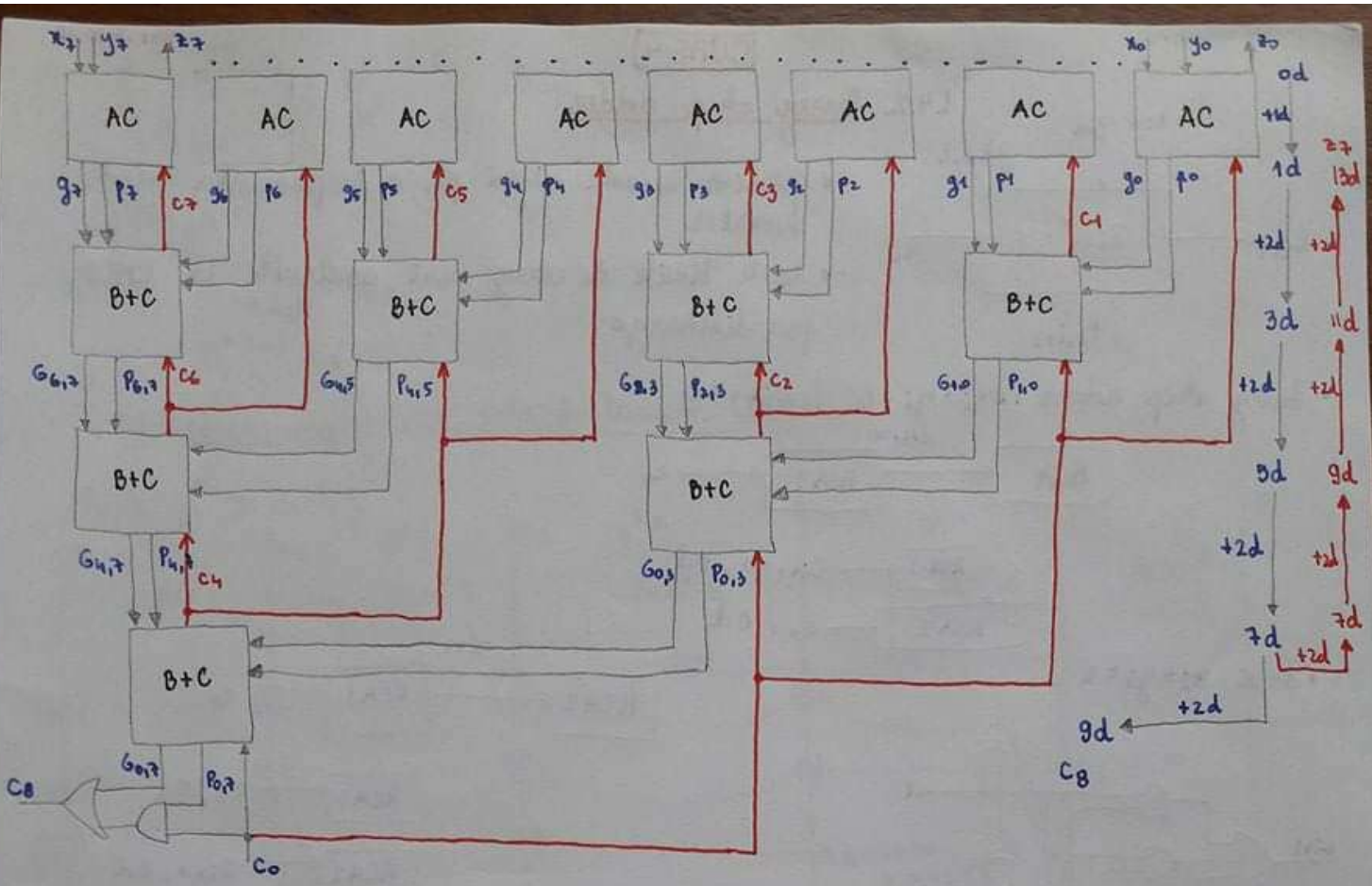
$c_{k+1} = G_{i,k} + P_{i,k} \cdot c_i \quad ; \quad i \le k$

$G_{i,k} = G_{j+1,k} + P_{j+1,k} \cdot G_{i,j} \quad ; \quad i \le j < k$

$P_{i,k} = P_{j+1,k} \cdot P_{i,j} \quad , \; i \le j < k$







$D^{z}_{MLCLA} = 13d$; $D^{cout}_{MLCLA} = 9d$

$D^{z}_{RCA-8} = 16d$; $D^{cout}_{RCA-8} = 16d$

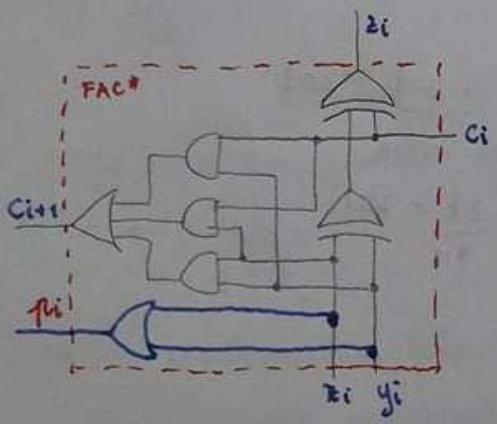
În general: $D^{z}_{MLCLA} = (4\lceil \log_2 m \rceil + 1)d$

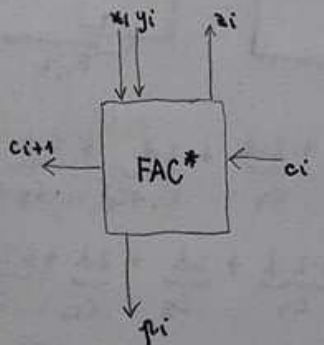
$D^{cout}_{MLCLA} = (2\lceil \log_2 n \rceil + 3)d$

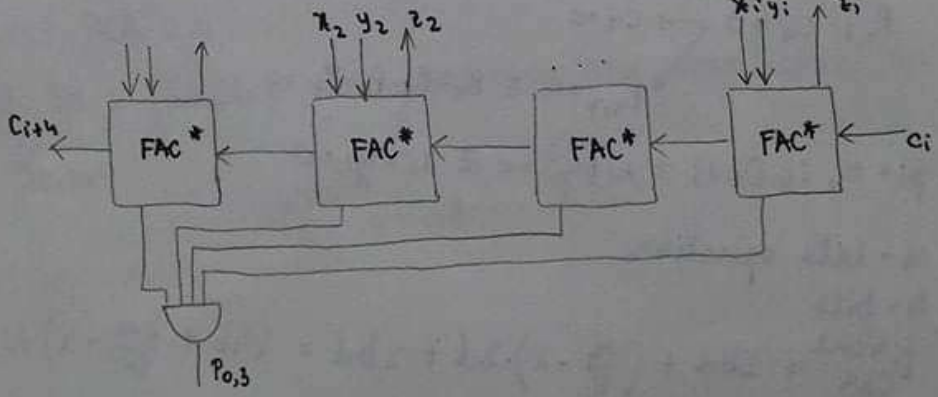
1.4.2. Sumator carry skip adder

$G_{0,3}$ $\underline{P_{0,3}}$

FAC ⟶ FAC* (+ p_i)

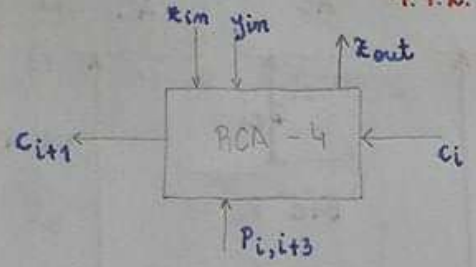






CURS 4

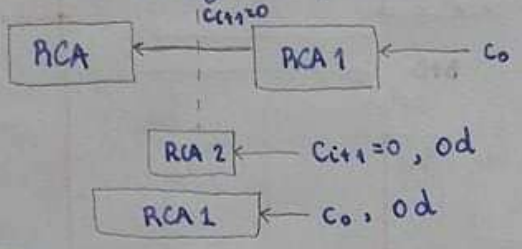
1.4.2. Carry skip adder



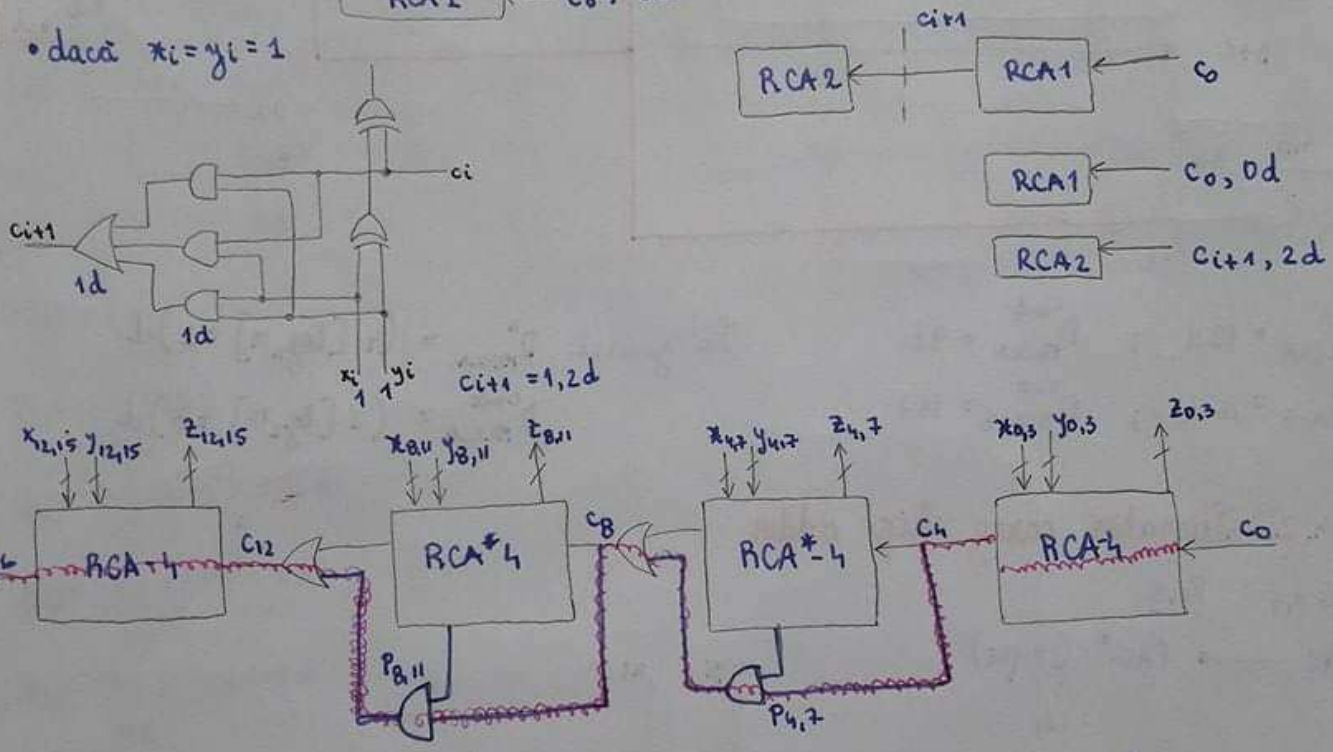
→ folosim această celulă pt. a implementa acest sumator

→ toate liniile de carry sunt construite în CMOS pro-discharge

• Carry skip adder : • $x_i = y_i = 0$, $c_{i+1} = 0$



• dacă $x_i = y_i = 1$



Calea cea mai lungă : $D^{cout}_{CSAA} = \underbrace{4 \cdot 2d}_{C_4} + \underbrace{2d}_{C_4 \to C_8} + \underbrace{2d}_{C_8 \to C_{12}} + \underbrace{4 \cdot 2 \cdot d}_{C_{12} \to C_{16}} = 20d$

$D_{CSA} = \underbrace{4 \cdot 2d}_{C_4} + \underbrace{2d}_{C_8} + \underbrace{2d}_{C_{12}} + \underbrace{2d}_{C_{15}} + \underbrace{2d}_{z_{15}} = 20d$

$P_{4,7} \cdot C_4 = 0 \longrightarrow C_4 = 0$

$\searrow P_{4,7} = 0 \equiv P_4 \cdot P_5 \cdot P_6 \cdot P_7 = 0$

$p_i = 0, i \in [4:7] \equiv x_i + y_i = 0 \equiv x_i = y_i = 0$

• n - bits operation

b - bits

$D^{z/cout}_{CSA} = 2bd + \left(\frac{n}{b} - 2\right) \cdot 2d + 2bd = \left(4b + \frac{2n}{b} - 4\right) d$

derivata, $\frac{d D^{z/cout}_{CSA}}{d b} = 4 - \frac{2m}{b^2}$ $\quad b_{opt} = \frac{\sqrt{2m}}{2}$ (opt. = optim)

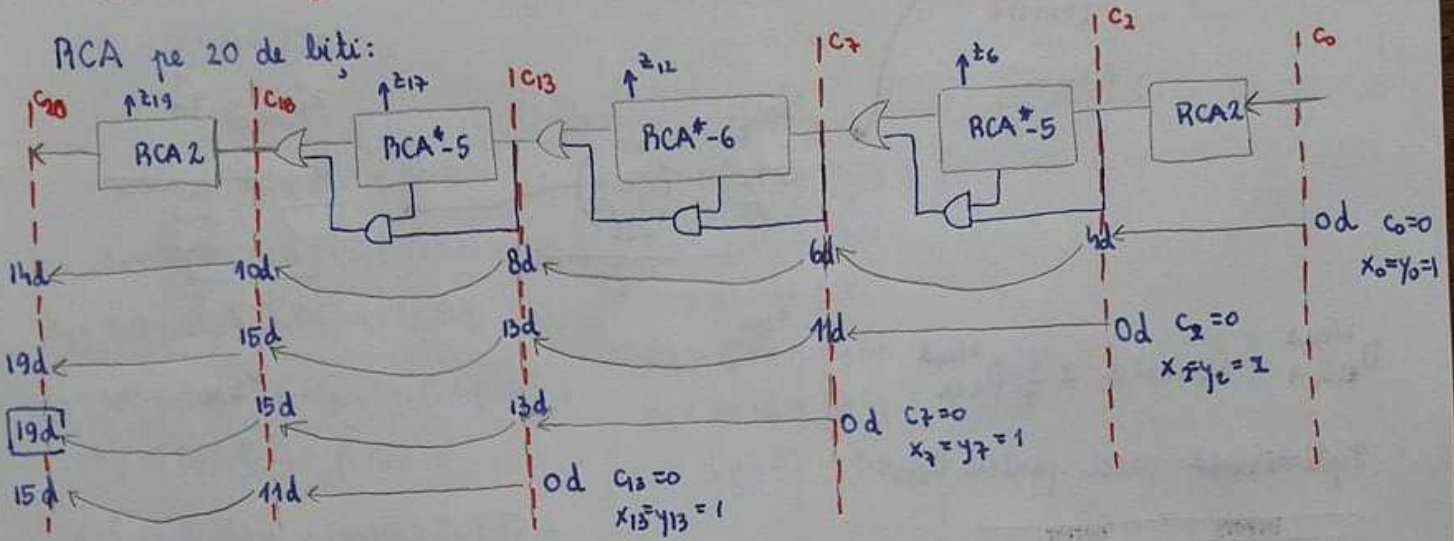
$\Rightarrow D^{z/cout}_{CSA\ opt} = 4(\sqrt{2m}-1)d$

- $m=32$ $\quad D^{z/cout}_{CSA\ opt} = 28d$

$D^{z/cout}_{RCA} = 64d$

Utilizarea segmentelor RCA de lungimi variabile

RCA pe 20 de biți:



$z_6 = \underbrace{4d}_{c_2} + 10d = 14d$

$z_{12} = \underbrace{11d}_{c_7} + 12d = 23d$

$z_{13} = \underbrace{13d}_{c_{13}} + 10d = 23d$

$z_{19} = \underbrace{15d}_{c_{16}} + 4d = 19d$

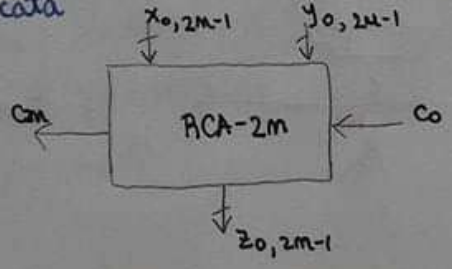
1.4.3. Carry Select Adder

↳ sumă condiționată de transport

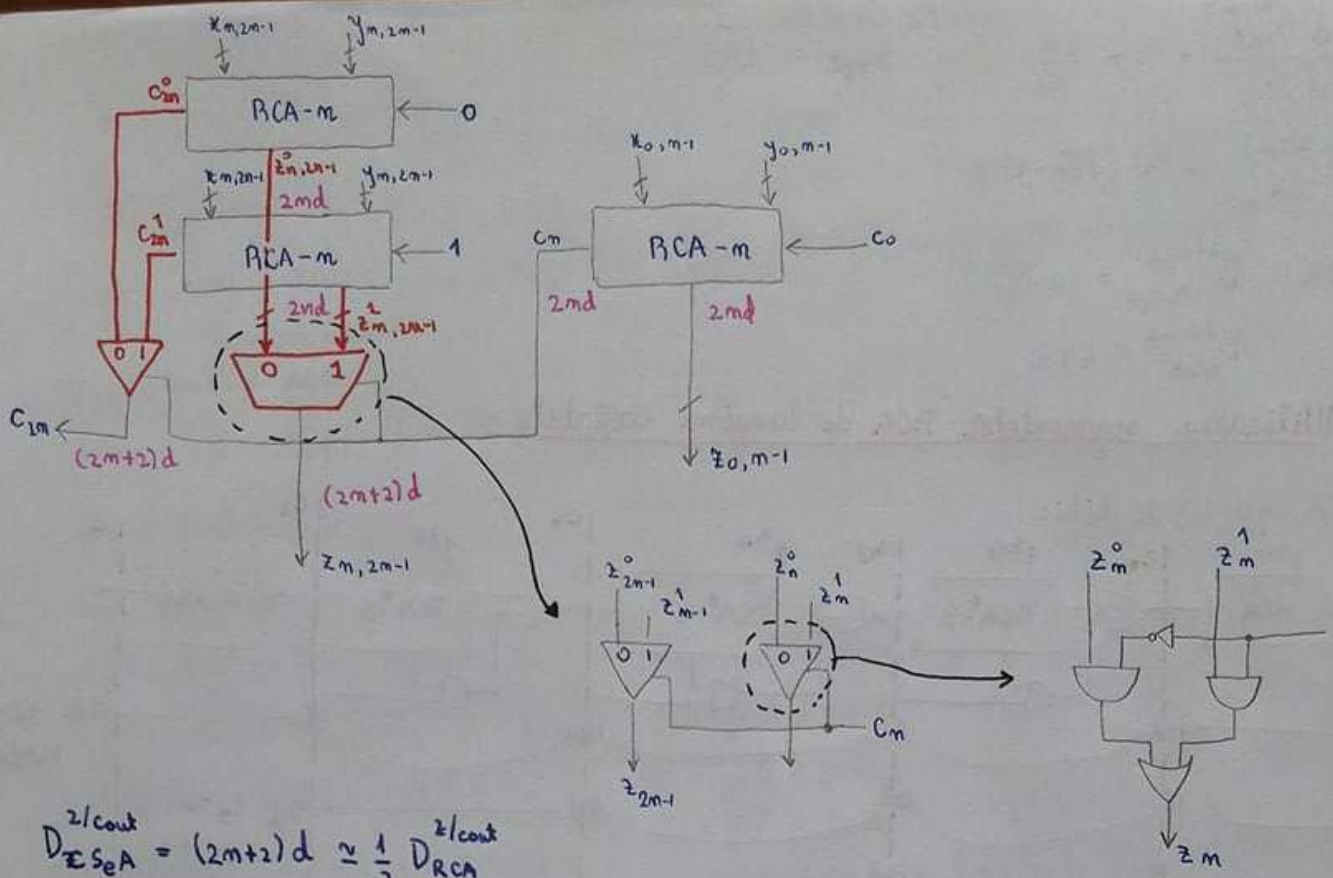
$c_i \begin{cases} \to 0 & (z_i^0, c_{i+1}^0) \\ \to 1 & (z_i^1, c_{i+1}^1) \end{cases}$

↳ Pt. implementare folosim: RCA 2m

↳ va împărți sumatorul în 2 jumătăți, iar jumătatea mai semnificativă va fi duplicată



$D^{z/cout}_{RCA} = 4md$



$$D^{2/cout}_{CSeA} = (2m+2)d \simeq \frac{1}{2} D^{2/cout}_{RCA}$$

Optimizarea ariei pentru C_{2m}:

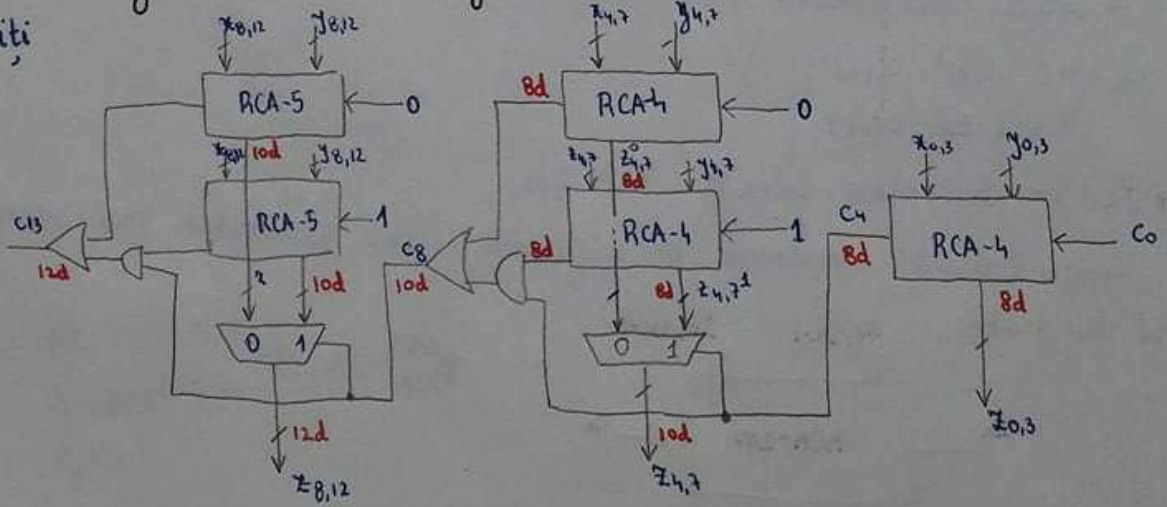
INPUTS			OUTPUT
C^0_{2m}	C^1_{2m}	C_m	C_{2m}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	d
1	0	1	d
1	1	0	1
1	1	1	1

$C^0_{2m} > C^1_{2m}$ (nu se poate obține)

$C_{2m} = C^0_{2m} + C^1_{2m} \cdot C_m$

- CSeA cu lungimi variabile ale segmentelor RCA:

↳ pe 13 biți



↳ se poate implementa și pe mai multe nivele

1.4.4. Conditional sum adder (CSA)

↳ primul nivel calculează: z_i^0, c_{i+1}^0
z_i^1, c_{i+1}^1

$X = x_3x_2x_1x_0$ $Y = y_3y_2y_1y_0$ $c_0 = 0$

$$c_{i+1} = \underbrace{x_iy_i}_{g_i} + x_ic_i + y_ic_i = x_iy_i\bar{c_i} + x_ic_i(1+y_i) + y_ic_i = \underbrace{x_iy_i}_{g_i}\bar{c_i} + \underbrace{(x_i+y_i)}_{p_i}c_i$$

↳ $x_iy_ic_i + x_iy_i\bar{c_i}$

$$c_{i+1} = g_i\bar{c_i} + p_ic_i$$

$$\overline{c_{i+1}} = (\bar{g_i}+c_i)(\bar{p_i}+\bar{c_i}) = \bar{g_i}\bar{p_i} + \bar{g_i}\bar{c_i} + \bar{p_i}c_i + c_i\bar{c_i}$$

$$\overline{c_{i+1}} = \bar{g_i}\bar{c_i} + \bar{p_i}c_i$$

Sum

$z_0 = x_0 \oplus y_0 \oplus c_0 = x_0 \oplus y_0$

$z_1 = x_1 \oplus y_1 \oplus c_1 = (\overline{x_1 \oplus y_1}) \cdot c_1 + (x_1 \oplus y_1)\bar{c_1}$

$z_2 = x_2 \oplus y_2 \oplus c_2 = (\overline{x_2 \oplus y_2})\, c_2 + (x_2 \oplus y_2)\,\bar{c_2}$

$z_3 = x_3 \oplus y_3 \oplus c_3 = (\overline{x_3 \oplus y_3})(g_2\bar{c_2} + p_2c_2) +$

$+ (x_3 \oplus y_3)(\bar{g_2}\bar{c_2} + \bar{p_2}c_2) =$

$= ((\overline{x_3 \oplus y_3})\, g_2 + (x_3 \oplus y_3)\,\bar{g_2})\,\bar{c_2} +$

$((\overline{x_3 \oplus y_3})\, p_2 + (x_3 \oplus y_3)\, p_2)\, c_2$

Carry

$c_0 = g_0\bar{c_0} + p_0c_0 = g_0$

$c_2 = g_1\bar{c_1} + p_1c_1$

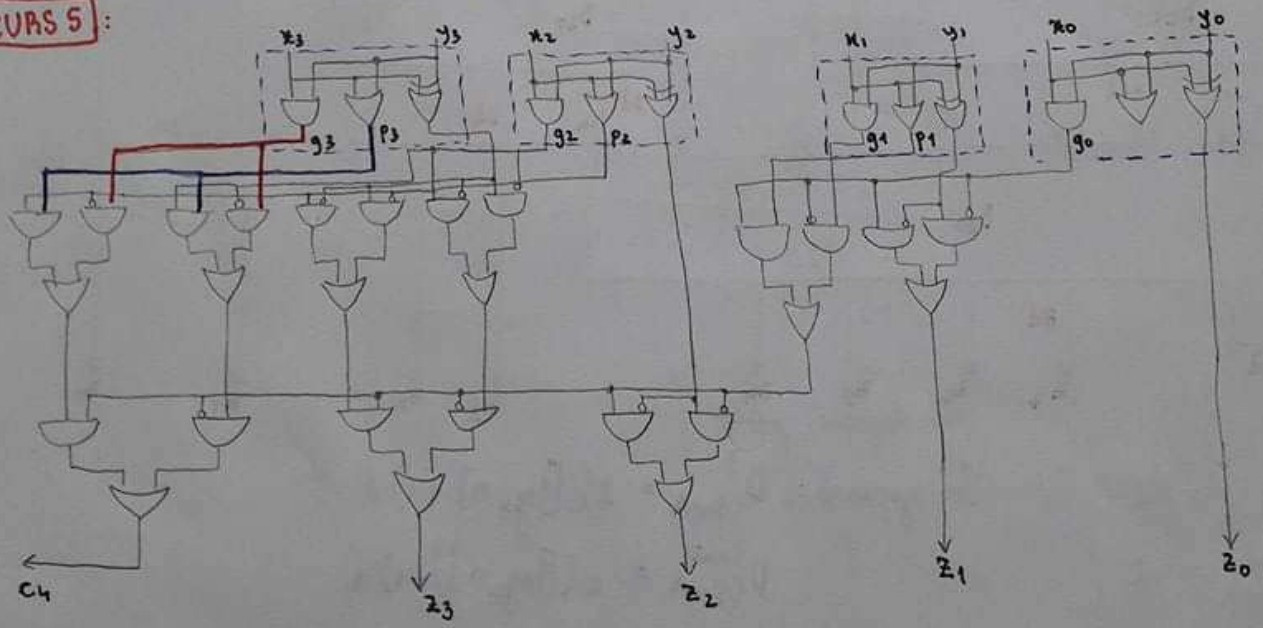
$c_3 = g_2\bar{c_2} + p_2c_2$

$\bar{c_3} = \bar{g_2}\bar{c_2} + \bar{p_2}c_2$

$c_4 = g_3\bar{c_3} + p_3c_3 = (g_3\bar{g_2} + p_3g_2)\,\bar{c_2} +$

$(g_3\bar{p_2} + p_3p_2)\, c_2$

CURS 5:



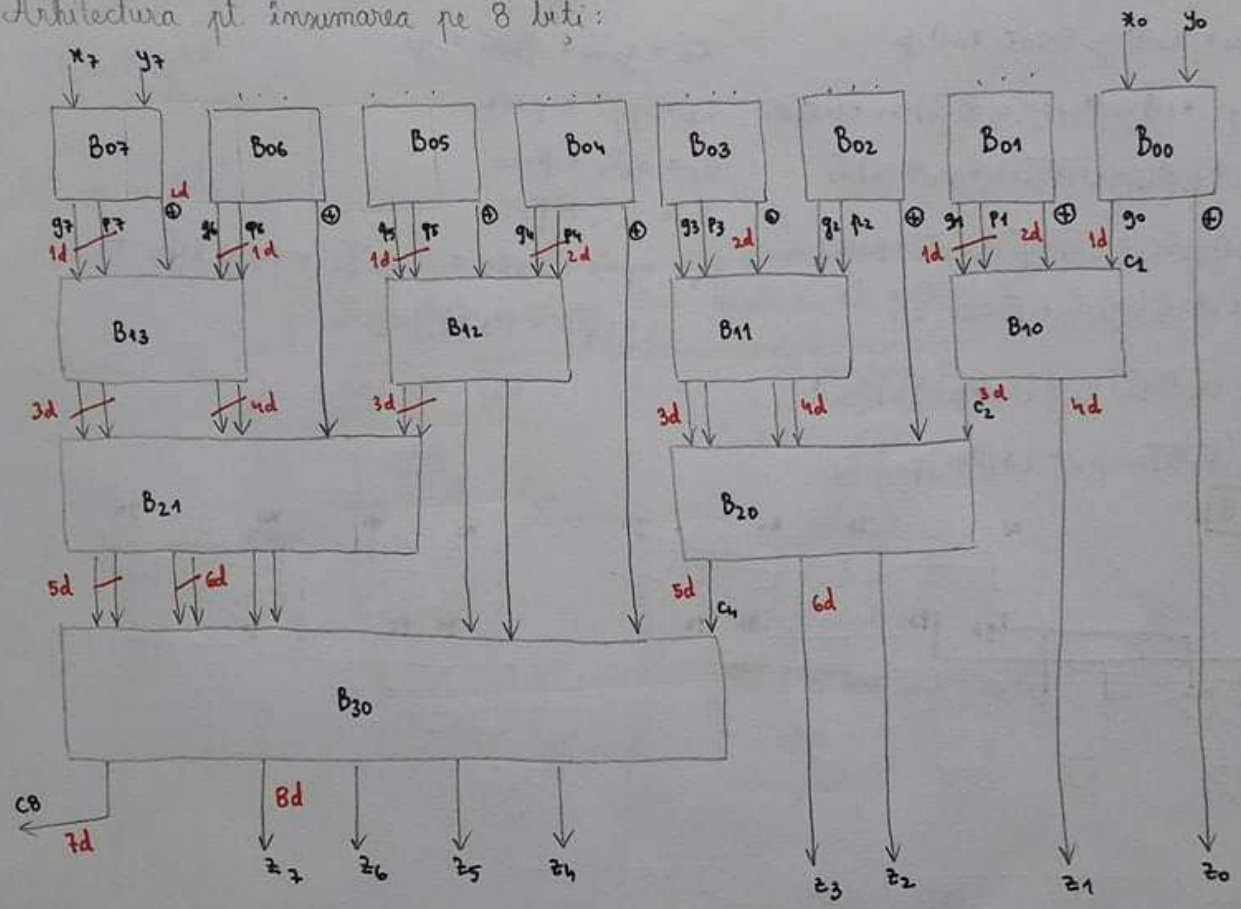
Exemplu: $X = 10001101$

$Y = 01110101$

$c_0 = 0$

Inputs \ Routes		7	6	5	4	3	2	1	0
X		1	0	0	0	1	1	0	1
Y		0	1	1	1	0	1	0	1
Block level	Carry in	C S	C S	C S	C S	C S	C S	C S	C S
i=0	Cin=0	0 1	0 1	0 1	0 1	1 0	1 0	0 1	0 1
	Cin=1	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0
i=1	Cin=0	0 1	1 0	1 0	1 0	1 0	0 1	1 0	0 1
	Cin=1	1 0	0 1	0 1	0 1	0 1	1 0	1 0	1 0
i=2	Cin=0	0 1	1 0	1 0	1 0	0 1	0 1	1 0	0 1
	Cin=1	1 0	0 1	0 1	0 1	0 1	1 0	1 0	1 0
i=3	Cin=0	1 0	0 1	0 1	0 1	0 1	0 1	1 0	0 1
	Cin=1								

Arhitectura pt insumarea pe 8 biti:



$D^{z}_{CSmA} = 8d$

$D^{Cout}_{CSmA} = 7d$

Im general: $D^{z}_{CSmA} = 2([\log_2 m] + 1)d$

$D^{Cout}_{CSmA} = 2([\log_2 m]) + 1)d$

Compararea sumatoarelor:

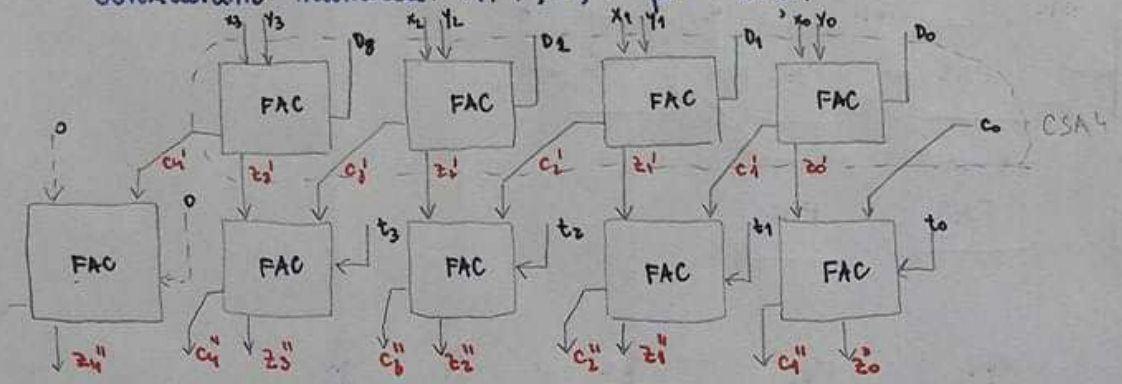
Exemplu: $m = 128$ biti

$D^{z}_{CSuA} = 16d$	$D^{z}_{CSeA} = 130d$	$D^{z}_{CSkA} = 4 \cdot 17 = 68d$	$D^{z}_{ML\text{-}CLA} = 29d$	$D^{z}_{RCA} = 256d$
$D^{Cout}_{CSuA} = 15d$	$D^{Cout}_{CSeA} = 130d$	$D^{Cout}_{CSkA} = 68d$	$D^{Cout}_{ML\text{-}CLA} = 17d$	$D^{Cout}_{RCA} = 256d$

1.4.5. Carry Save Adder

↳ result is redundant : → carry vector
→ sum vector

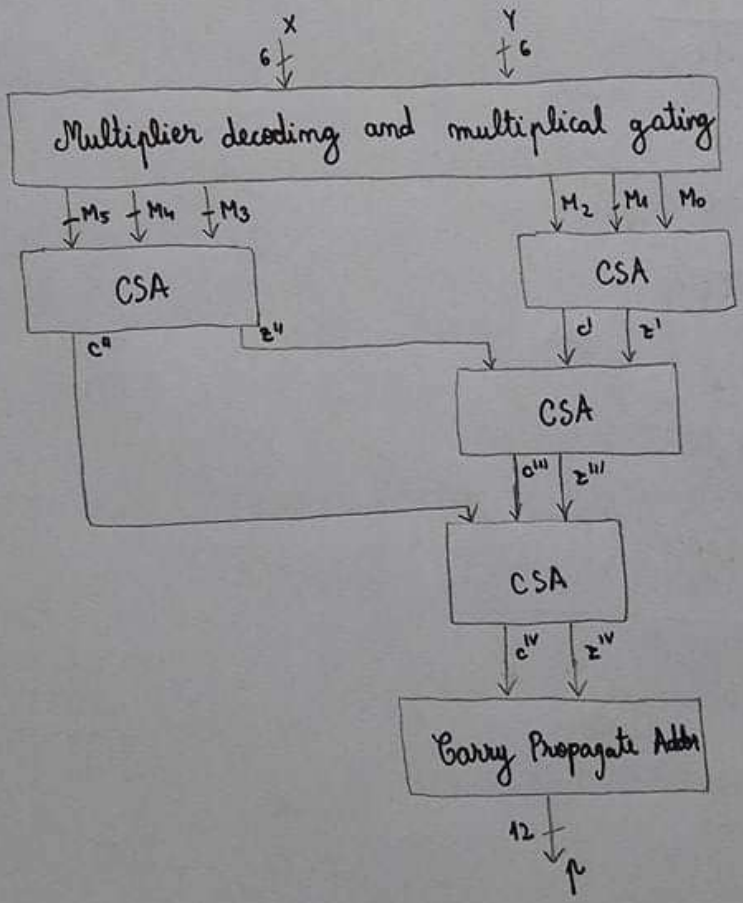
Considerăm numerele: X, Y, S, T pe 4 biți.



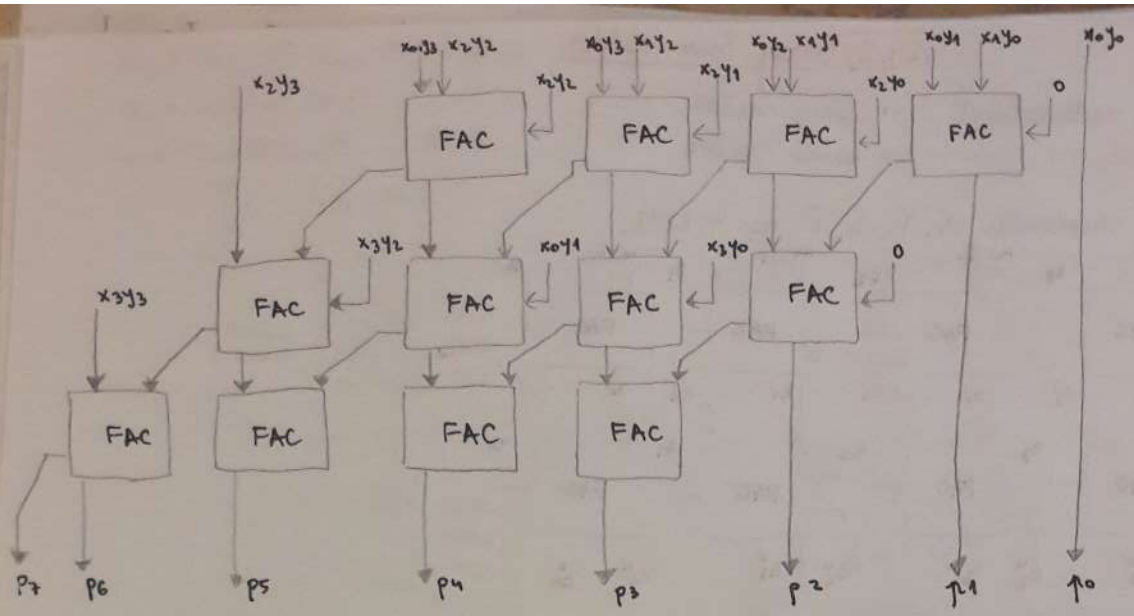
X, Y, S, T → z'' , c''

X, Y pe 6 biți

$M_i = X_i \cdot Y \cdot 2^i$



• Înmulțim X și Y pe 4 biți folosind arhitectura:



X = 15 1111

Y = 13 1101

$M_0 = X_0 Y \cdot 2^0$ . . . .1101 | + CSA

$M_1 = X_1 Y \cdot 2^1$ . . . .1101 | + CSA

$M_2 = X_2 Y 2^2$ . . . .1101 | + CSA

z^1 100011

c^1 011100

$2c^1$ 011100

z^1 . . . . . 100011 | + CSA

M_3 = ... 1101 | + CSA

z^{11} 1110011

c^{11} 0101000

$2c^{11}$ 0101000

z^{11} . . . . . 1110011 | +

P 11000011

S6: 30.10.18

CURS 6

1.5. Reliable Computing

→ disponibilitatea (availability)

→ fiabilitatea (reliability)

→ maintainability (mentenanța)

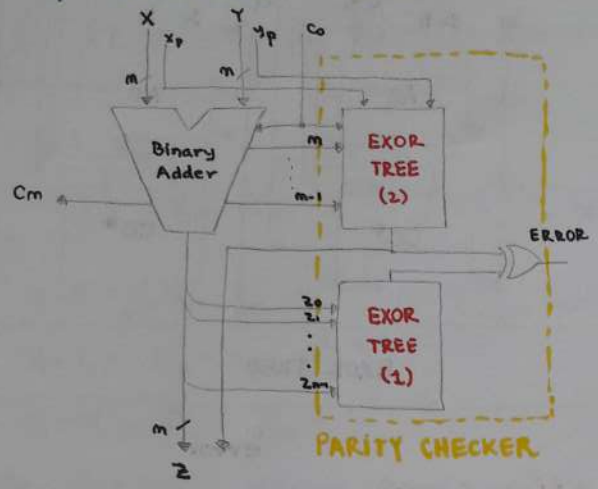
1.5.1. Sumatoare cu control de paritate

X, Y Z = X + Y

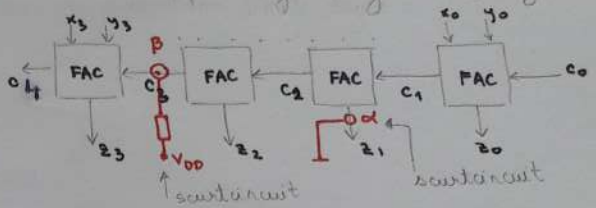
$x_p = x_{m-1} \oplus x_{m-2} \oplus \ldots \oplus x_0$

$y_p = y_{n-1} \oplus y_{n-2} \oplus \ldots \oplus y_0$

$$z_p = z_{m-1} \oplus z_{n-2} \oplus \dots \oplus z_0 \quad (1)$$
$$z_i = x_i \oplus y_i \oplus c_i$$
$$c_p = c_{n-1} \oplus c_{n-2} \oplus \dots \oplus c_0$$
$$\Rightarrow z_p = x_p \oplus y_p \oplus c_p \quad (2)$$



Discutam defecte logice singulare (Single Stuck-at):



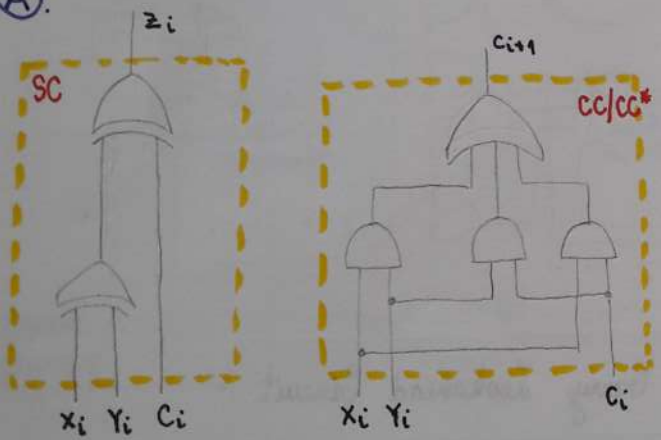
fault free: $X = 0011$, $Y = 0011$, $C = 00110$ (c_0), $Z = 0110$
$x_p = 0$, $y_p = 0$, $c_p = 0$ $\Rightarrow z_p = 0$ (2)
$z_p = 0$ (1) ✓

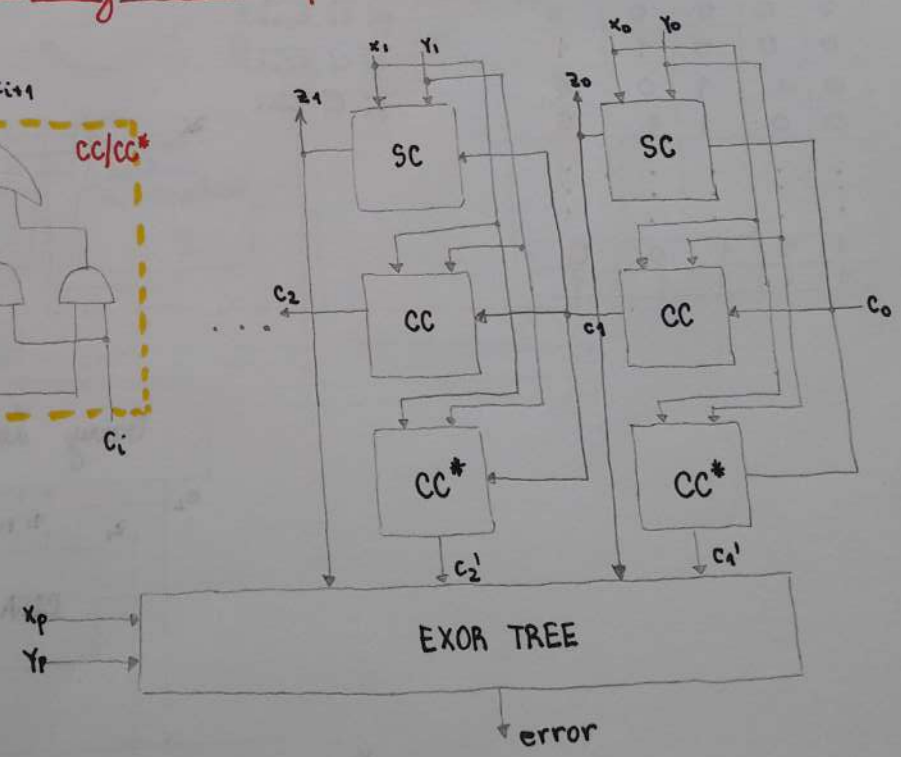
α: $X = 0011$, $Y = 0011$, $C = 00110$, $Z = 0100$
$x_p = 0$, $y_p = 0$, $c_p = 0$, $z_p = 1$ (1) $\Rightarrow z_p = 0$ (2) !!

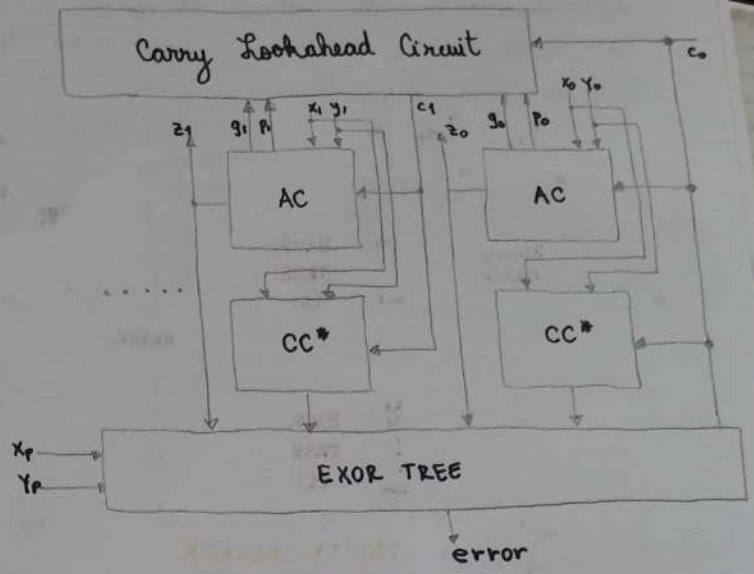
β: $X = 0011$, $Y = 0011$, $C = 01110$, $Z = 1110$
$x_p = 0$, $y_p = 0$, $c_p = 1$, $z_p = 1$ (1) $\Rightarrow z_p = 1$ (2)

1.5.2. Carry Chain Duplication

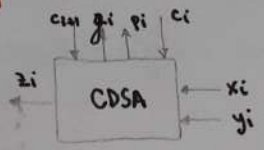
(A).





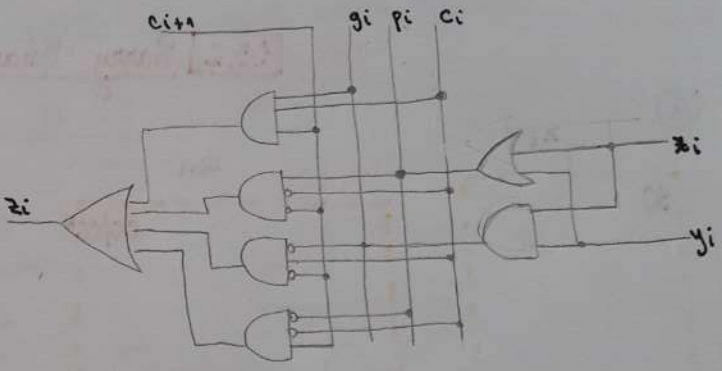


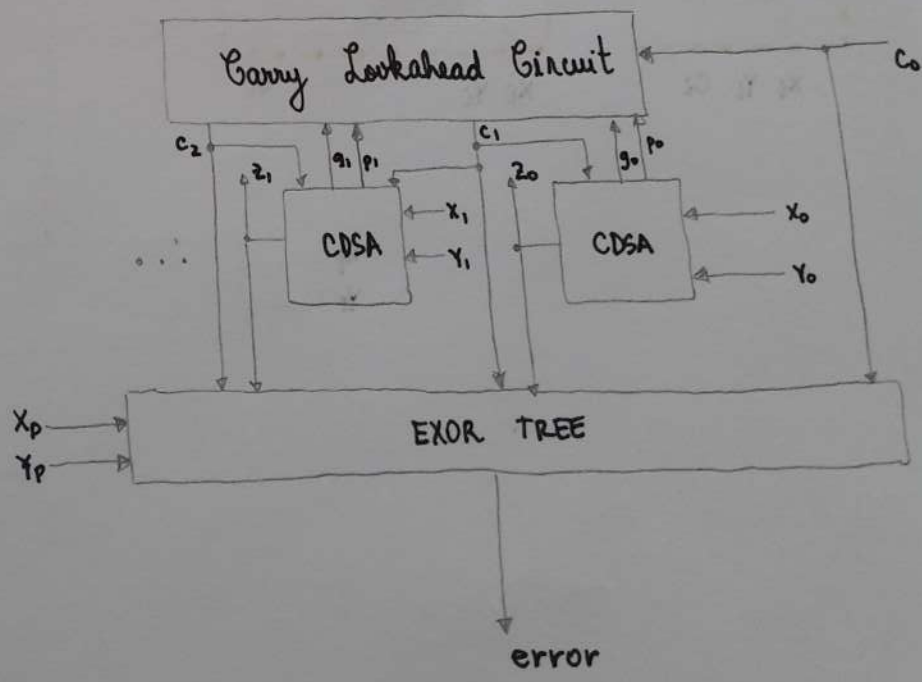
1.5.3. Carry Dependent Sum Adder (CDSA)



Input				Output
x_i	y_i	c_i	c_{i+1}	z_i
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
⋮	⋮	⋮	⋮	⋮
1	1	1	0	0
1	1	1	1	1

$$z_i = g_i c_i c_{i+1} + p_i \overline{c_i}\ \overline{c_{i+1}} + \overline{g_i} c_i \overline{c_{i+1}} + \overline{p_i}\ \overline{c_i} c_{i+1}$$





CAPITOLUL ②: OPERAȚII ARITMETICE CU NR. ÎN VIRGULĂ FLOTANTĂ

2.1. IEEE 754

→ packed : pt. stocare și transmitere de date

→ unpacked : pt. operații

$X = X_M \cdot 2^{X_E}$; $Y = Y_M \cdot 2^{Y_E}$

$X + Y = (X_M + Y_M \cdot \underbrace{2^{Y_E - X_E}}_{\text{aliniere}}) \cdot 2^{X_E}$, $X_E \geq Y_E$

$X - Y = (X_M - Y_M \cdot 2^{Y_E - X_E}) \cdot 2^{X_E}$, $X_E \geq Y_E$

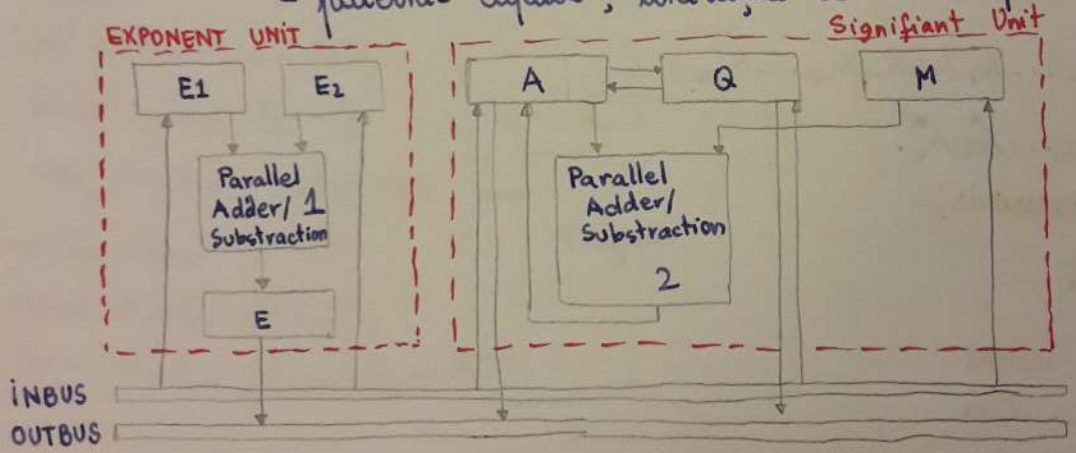
$X * Y = X_M * Y_M * 2^{X_E + Y_E}$

$X \div Y = X_M \div Y_M * 2^{X_E - Y_E}$

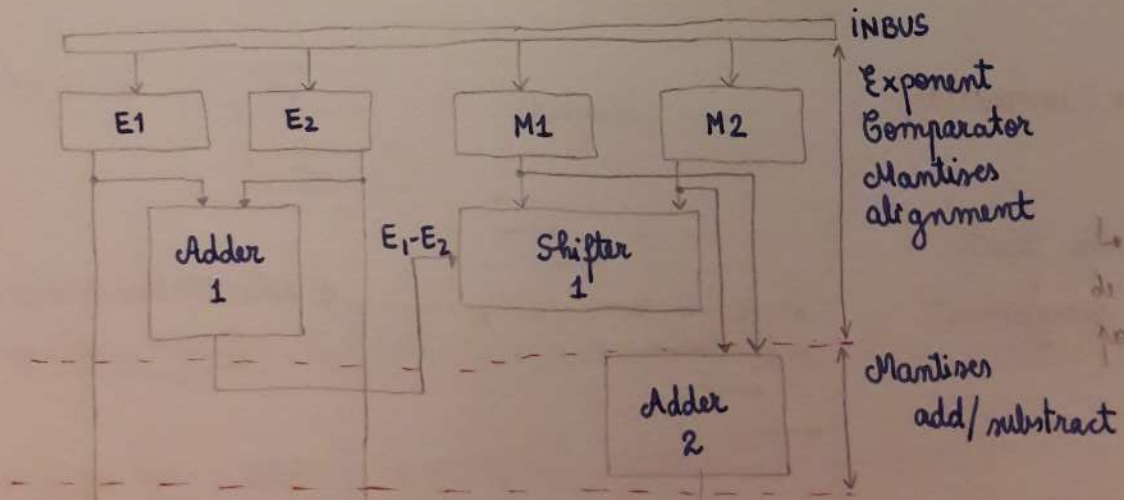
Obs: Operațiile folosesc două unități : exponent , mantisă (nr. de virgulă fixă).
(+/-) (+/÷/*/÷/aliniere)

Există două tipuri de arhitecturi : - loosly coupled (slab cuplate, unitățile de exp. și mantisă comunică prin magistrale partajate

- puternic cuplate, unitățile comunică prin legături directe.

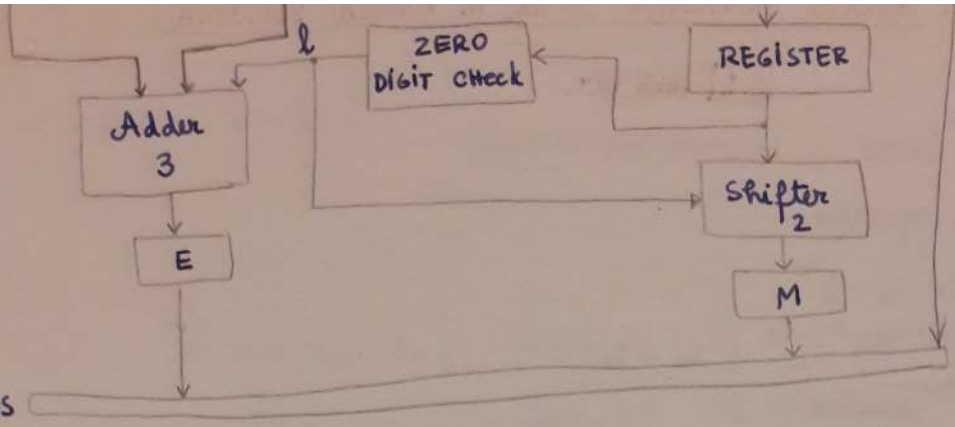


CURS 7



Shifter 1 - realizează accelerarea alinierii

Zero Digit Blocker - det. nr de grupuri de 4 biți de zero în poz. mantisei



$$X+Y = (X_M + Y_M \cdot \underbrace{2^{Y_E - X_E}}_{\text{align}}) \cdot 2^{X_E}, \quad X_E \geq Y_E$$

$E_1 - E_2 = K \ (K>0)$, $4K \rightarrow M_2$

$E_1 - E_2 = -K \ (K \geq 0)$, $4K \rightarrow M_1$

2.2 Rotunjirea nr. de virgulă flotantă

DEF: Rotunjirea rep. conversia unei rep. de precizie înaltă într-o reprezentare de precizie scăzută.

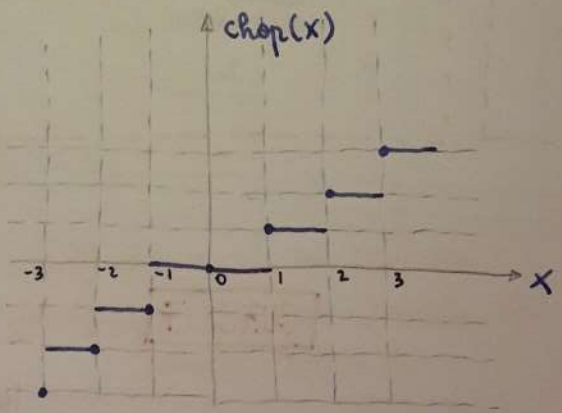
$$X = X_{m-1} X_{m-2} \ldots X_1 X_0 . X_{-1} X_{-2} \ldots X_{-n}$$

$$X^* = X^*_{m-1} X^*_{m-2} \ldots X^*_1 X^*_0 .$$

(A). towards 0 (inwards)

$X^*: |X^*| \leq |X|$

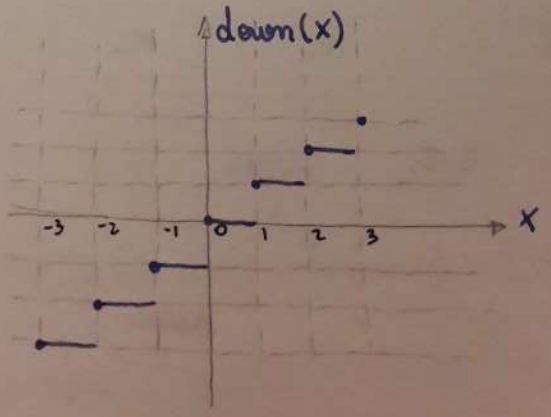
Dacă X e rep. în semn mărime, rotunjirea către 0 înseamnă trunchierea părții fracționare



(B). towards $-\infty$ (downwards)

$X^*: X^* < X$

Trunchierea părții fract. pt. nr. rep. în complement de 2.

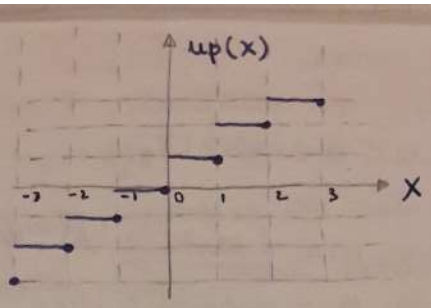


↳ erorile se acumulează spre o limită inferioară

© towards $+\infty$ (upwards)

$X^*: X^* > X$

↳ erorile se acumulează în aceeași direcție (în sus)



(D') to nearest

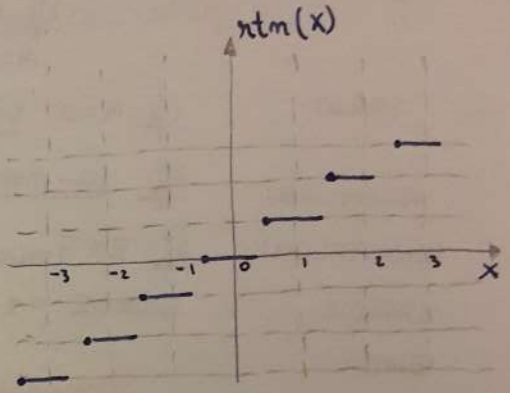
$X \geq 0$, $X^* \rightarrow X_{m-1}X_{m-2}\dots X_1X_0$, if $X_{-1}X_{-2}\dots X_{-m} < \frac{1}{2}$

$\rightarrow X_{m-1}X_{m-2}\dots X_1X_0 + 1$ if $X_{-1}X_{-2}\dots X_{-m} > \frac{1}{2}$

Error accumulation: $X_{-3} = X_{-4} = \dots = X_{-m} = 0$

	Inputs		Outputs	
	X_{-1}	X_{-2}	X^*	$\varepsilon = X^* - X$
.00 = .0	0	0	$X_{m-1}X_{m-2}\dots X_1X_0$	0
.01 = 0.25	0	1	$X_{m-1}X_{m-2}\dots X_1X_0$	$-\frac{1}{4}$
.10 = 0.5	1	0	$X_{m-1}X_{m-2}\dots X_1X_0 + 1$	$\frac{1}{2}$ ($+\frac{1}{2}$ / $-\frac{1}{2}$)
.11 = 0.75	1	1	$X_{m-1}X_{m-2}\dots X_1X_0 + 1$	$\frac{1}{4}$

$\varepsilon_{mediu} = \frac{1}{8}$



(D) to nearest even

$X \geq 0$

$X^* \rightarrow X_{n-1}X_{m-2}\dots X_1X_0$,

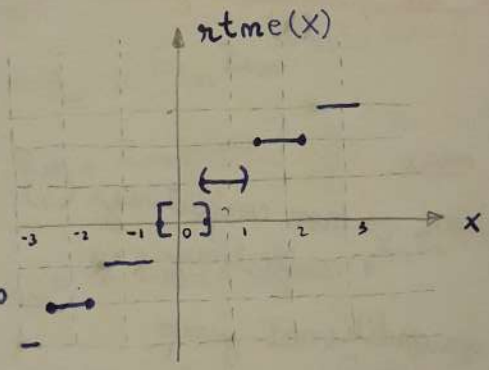
dacă $.X_{-1}X_{-2}\dots X_{-n} < \frac{1}{2}$

sau $.X_{-1}.X_{-2}\dots X_{-m} = \frac{1}{2}$ și $X_0 = 0$

$\rightarrow X_{m-1}X_{m-2}\dots X_1X_0 + 1$

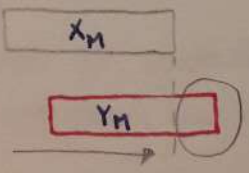
dacă $.X_{-1}X_{-2}\dots X_{-m} > \frac{1}{2}$

sau $.X_{-1}X_{-2}\dots X_{-m} = \frac{1}{2}$ și $X_0 = 1$



2.3 Adunarea și scăderea numerelor de virgulă flotantă cu precizie infinită

$X_E \geq Y_E$



adder/substracter 1

declare register $E_1[(e-1):0]$, $E_2[(e-1):0]$, $E[(e-1):0]$, $A[(m+1):0]$, $M[(m+1):0]$, A_COUT, ERROR;

declare bus INBUS $[(m+e+1):0]$, OUTBUS $[(m+e+1):0]$;

```
BEGIN:      ERROR=0, A_COUT=0,
INPUT:      E1:=INBUS(XE), A:=INBUS(XM);
            E2:=INBUS(YE), M:=INBUS(YM);
COMPARE:    E:=E1-E2
ALIGN:      if E<0 then A:=rightshift(A), E:=E+1, go to ALIGN;
            if E>0 then M:=rightshift(M), E:=E-1, go to ALIGN;
ADD/
SUBSTRACT:  A:=A±M, E:=max(E1,E2);
OVERFLOW:   if A_COUT=1 then begin
                if E=Emax, then go to ERROR;
                A:=rightshift(A), E:=E+1, go to END; end
ZERO:       if A=0 then E=0, go to END;
NORMALIZE:  if is-normalised(A) then go to END;
UNDERFLOW:  if E>Emin then A:=leftshift(A), E:=E-1, go to NORMALIZE;
ERROR:      ERROR:=1;
END:
```

Elemente ale pseudolimbajului

① A_COUT.A (rezultat final)
↳ concatenare

② IOBUS

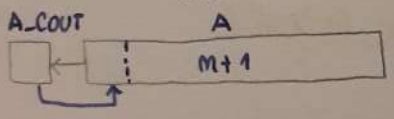
③ :=, rightshift, max (fct. cablate)

④ Secuential operation → non_conflicting (,)
→ conflicting (;)

⑤ Control flow → unconditional jump
→ conditional jump

⑥ A:=A+M, M:=0;

- operanzii: m+e+2 → 1 sign
e exponent
m+1 mantisa
- registrii: A, M;



E_1, E_2

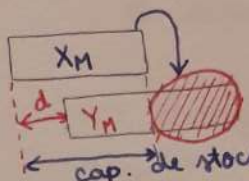
ERROR

CURS 8

2.4. Rotunjirea rezultatului adunării / scăderii

$X_M + Y_M$, Y_M aligned $|Y_E - X_E|$ $\quad X = 1.X_{m-2}X_{m-3}\dots X_i\dots X_1X_0$ (exponenți: 2^{-1}, 2^{-2}, 2^{-m+1})

- Y_M aligned → precizie infinită: păstrează toți biții cu $\omega < -2^{m+1}$
  → precizie finită: 3 biți numiți sticky:



① g: bitul de gardă - folosit pt păstrează preciziei, pondere 2^{-m} ultimul bit care părăsește Y_M prin shift. la dreapta
② r: bitul de rotunjire; are rol în rotunjire, pondere 2^{-m-1}
③ s: sticky bit = un sau logic între toți ceilalți biți, pondere 2^{-m-2}

$Z_m = X_M + Y_M$ aliniat
$Z_M = \underbrace{Z_m}_{\text{cout}} Z_{m-1}.Z_{m-2}Z_{m-3}\dots Z_1Z_0 \,!\, grs$
↓ din poz cea mai semnificativă suprainitar: 2^0

$Z_M = Z_m Z_{m-1}.Z_{m-2}Z_{m-3}\dots Z_1Z_0 \,!\, grs$

$Z_{M_m} = 1.Z_{m-2_m}Z_{m-3_m}\dots Z_{1_m}Z_{0_m} \,!\, RS$

① $Z_m = 1$ (1 bit shiftat la dreapta) $\quad Z_{M_m} = 1.Z_{m-1}Z_{m-2}\dots Z_2Z_1 \,!\, Z_0$ (g or r or s)

② $Z_m = 0$, $Z_{m-1} = 1$ (Z_M normalizat) $\quad Z_{M_m} = 1.Z_{m-2}Z_{m-3}\dots Z_1Z_0 \,!\, g$ (r or s)

③ $Z_m = 0$, $Z_{m-1} = 0$, $Z_{m-2} = 1$ (1 bit shiftat la stânga) $\quad Z_{M_m} = 1.Z_{m-3}Z_{m-4}\dots Z_0 g \,!\, r\ s$

④ $Z_m = 0$, $Z_{m-1} = 0$, $Z_{m-2} = 0$, $Z_{m-3} = 1$ (2 biți shiftați) $\quad Z_{M_m} = 1.Z_{m-4}Z_{m-5}\dots g\,0 \,!\, 0\,0$

Atunci când pt. normalizare este necesară o shiftare de mai mulți biți la stânga, Z_M este completat cu bitul de gardă urmat de 0, iar R și S sunt și ele egalate cu 0.

Rotunjire: $Z_M = Z_m Z_{m-1}.Z_{m-2}Z_{m-3}\dots Z_1Z_0 \,!\, grs$

$Z_{M_m} = 1.Z_{m-1_m}Z_{m-2_m}\dots Z_{1_m}Z_{0_m} \,!\, RS$

Z_M^*

Mod de rotunjire	$Z_{Mm} \geq 0$	$Z_{Mm} \leq 0$
towards 0	$Z_M^* = Z_{Mm}$ discard R,S	$Z_M^* = Z_{Mm}$ discard R,S
towards $-\infty$	$Z_M^* = Z_{Mm}$	if R or S then $Z_M^* = Z_{Mm}^{-1}$
towards $+\infty$	if R or S then $Z_M^* = Z_{Mm} + 1$	-
to nearest even	if R and (S or Z_{0m}) then $Z_M^* = Z_{Mm}+1$	if R and (S or Z_{0m}) then $Z_M^* = Z_{Mm}^{-1}$

Dacă $Z_{m-1} \dots Z_0$ sunt 1 se generează carry out și se realizează postnormalizare.

2.5 Algoritmul de adunare/scădere a nr. de virgulă flotantă cu rotunjire

↳ Op. de adunare/scădere cu precizie finită.

↳ Eroarea de rotunjire nu este corelată cu diferența exponenților

- semn: 1 bit
- exponent: 3 biți bias $= 2^{3-1} - 1 = 3$
- significand: 4 biți

$X = 0.5625 = 0.1001 + 2^0 = 1.\widetilde{001} \cdot 2^{-1}$

$Y = -3.75 = -11.11 = -1.\underline{111} + 2^1$

X: 0 0 1 0 0 0 1

Y: 1 1 0 0 1 1 1 (= exp. + bias)

Step 1: despachetare + verificare cazuri speciale

X: 0 0 1 0 1.0 0 1

Y: 1 1 0 0 1. 1 1 1

Step 2: se calculează diferența exponenților

$d = X_E - Y_E = -2$

if $d < 0$ => interschimbăm X cu Y (doar Y poate fi shiftat) pt economisire de memorie

$z_E = \max(X_E, Y_E)$ => $z_E = 4$

X: 1 1 0 0 1 1 1 1

Y: 0 0 1 0 1 0 0 1 1.001

Step 3: if sign(X) ≠ sign(Y) . pt evitarea scăderii complementam un nr

$Y_{MC_2} = 0.111$ ← cel mai din dreapta rămâne necomplementat

Step 4: aliniere : Y_M shiftat la dreapta cu $|d=2|$ biți

Dacă Y_M a fost complementat în step3, la aliniere se introduc biți de 1. Altfel, se introduc biți de 0 în pozițiile cele mai semnificative poziții

Se păstrează biții de sticky.

$Y_{M_{C_2}al} = 1.101 \,!\, \overset{g}{1}\ \overset{r}{1}\ \overset{s}{0}$

Step 5: adunarea celor 2 significanți

$$\begin{array}{l} X_M = 1.111 \quad g\ r\ s \\ Y_{M_{C_2}al} = 1.101 \,!\, 1\ 1\ 0 \\ \hline Z_M = \underset{\text{cout}}{\cancel{1}}1.100 \,!\, 1\ 1\ 0 \end{array} \Big| +$$

if sign(X) = sign(Y)
- if cout from msb ⇒ cout se păstrează

if sign(X) ≠ sign(Y) — rezultatul e negativ
- if no cout from msb ⇒ $Z_{M_{C_2}}$ ← complementat
- if cout from msb ⇒ se neglijează cout

Step 6: prenormalizarea (sect. 2.4.)

Verificăm excepțiile:
- if $z_E = z_{E\,max}$ (6, for 3 bit exponents) & Z_M 1 bit shiftat ⇒ OVERFLOW
- if $z_E = z_{E\,min}$ (1) & Z_M are nevoie de 1 bit /mai mulți pt shiftare la stânga ⇒ UNDERFLOW

$Z_{M_m} = 1.100$ (z_{om}) $\qquad z_E = 4$

Step 7: R, S - conform sect. 2.4

pt. cazul nostru : $R = g = 1$

$S = r \text{ or } s = 1 \text{ or } 0 = 1$

Step 8: Rotunjirea (sect. 2.4.)

Dacă se generează în urma rotunjirii un cout mai semnificativ se va impune o postnormalizare (= 1 bit shiftat la dreapta)

Verificăm cazul de excepție.

Considerăm : rotunjirea to nearest even

$R \text{ and } (S \text{ or } z_{0_m}) = 1 \text{ and } (1 \text{ or } 0) = 1 \Rightarrow Z_{M_m} = 1.100 +$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad 0.001$

$Z_M^* = 1.101$

Step 9: Semnul rezultatului Z

if sign(X) == sign(Y)
semnul lui X|Y

else :

Swap #2	Complement step5	sign X	sign Y	sign Z
YES		+	−	−
YES		−	+	+
NO	YES	+	−	−
NO	YES	−	+	+
NO	NO	+	−	+
NO	NO	−	+	−

inițiale

swap = YES
sign(X) = +
⇓
sign(Z) = −

Step 10: impachetam rezultatul

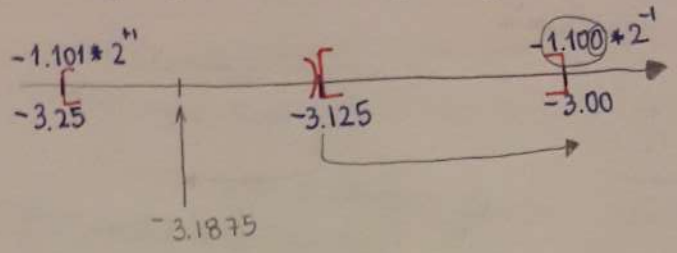
z

1	1	0	0	1	0	1

Verificăm rezultatul: $X = 0.5625$; $Y = -3.75$

$X + Y = -3.1875$

$z = (-1)^1 * 2^{4-3} * (1.101) = -2 * 1.101 = -11.01 = -3.25$



PARTIAL ↑

Prenormalization shifter

↳ implementează step 6 și step 7

↳ combinational

step5: $Z_M = \overbrace{Z_4 Z_3}^{\text{1 if out}} . Z_2 Z_1 Z_0 \mid grs$

$Z_{Mm} = 1. \; Z_{2m} \; Z_{1m} \; Z_{m0} \mid RS$

① $Z_4 = 0$, $Z_3 = 1$ (Z_M normalizat) $Z_{MN} = 1. \; Z_2 \; Z_1 \; Z_0 \mid g \; (r \text{ or } s)$ — l/r_0 ← var. booleene

② Z_M 1 bit LShifted, $Z_4 = 0$ $Z_3 = 0$ $Z_2 = 1$: $Z_{Mm} = 1. \; Z_1 \; Z_0 \; g \mid r \; s$ — l_1

③ Z_M 2 bit LShift: $Z_{Mm} = 1. \; Z_0 \; g \; 0 \mid 0 \; 0$ — l_2

④ Z_M 3 bit LShift: $Z_{Mm} = 1. \; g \; 0 \; 0 \mid 0 \; 0$ — l_3

⑤ Z_M 1 bit RShift: $Z_{Mm} = 1. \; Z_3 \; Z_2 \; Z_1 \mid Z_0 \; (g \text{ or } s \text{ or } r)$ — r_1

$Z_{2m} = Z_2 \cdot l/r_0 + Z_1 \cdot l_1 + Z_0 l_2 + g l_3 + Z_3 r_1$

$Z_{1m} = Z_1 \cdot l/r_0 + Z_0 \cdot l_1 + g \cdot l_2 + Z_2 r_1$

$Z_{0m} = Z_0 \cdot l/r_0 + g \cdot l_1 + Z_1 \cdot r_1$

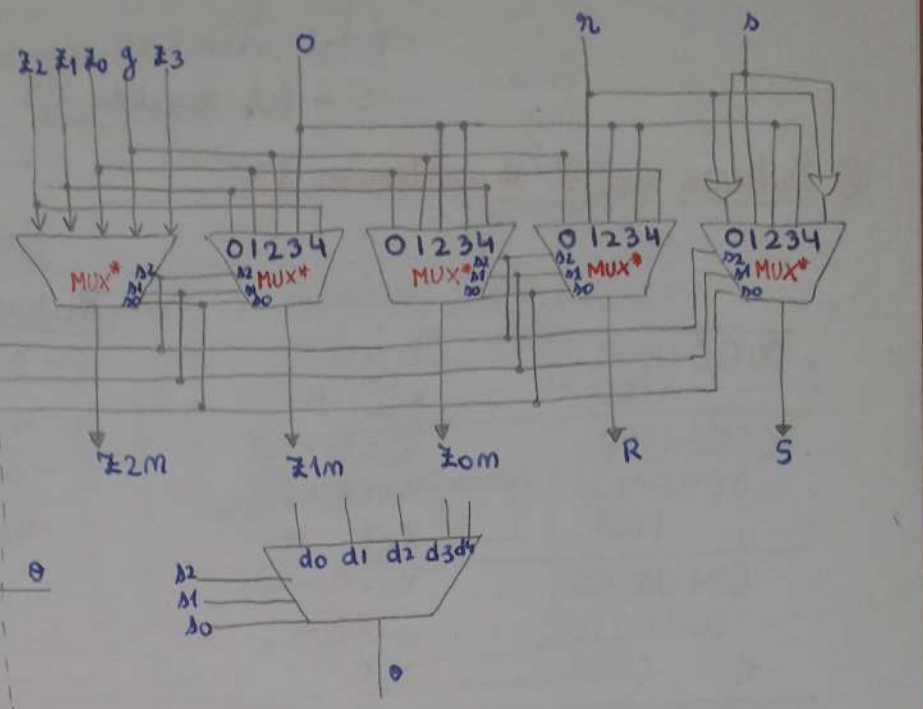
$R = g \cdot l/r_0 + r \cdot l_1 + Z_0 r_1$

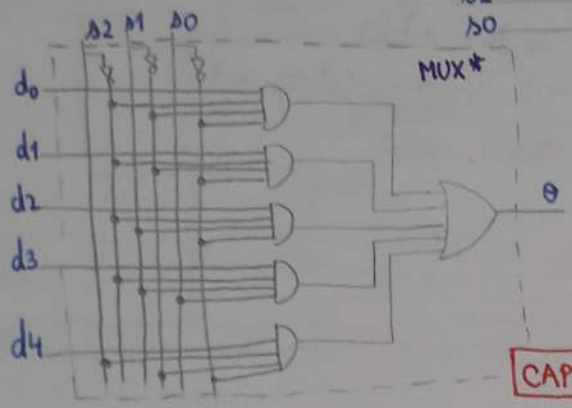
$S = (r \text{ or } s) \cdot l/r_0 + s \cdot l_1 + (g \text{ or } r \text{ or } s) \cdot r_1$

CURS 9

2.5. $\Delta 2\,\Delta 1\,\Delta 0$

Input					Output		
x_1	l_3	l_2	l_1	$l_{/r0}$	$\Delta 2$	$\Delta 1$	$\Delta 0$
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	1
1	0	0	0	0	1	0	0





CAPITOLUL 3

Analiza funcțională și sinteza dispozitivelor de înmulțire binară

3.1. Metode de înmulțire

(A) Paper & pencil

$$\begin{array}{l} 1100\ldots Y \\ 1011 = X_3X_2\ldots X \\ \hline 1100\ldots X_0 Y\cdot 2^0 \\ 1100\ldots X_1 Y\cdot 2^1 \\ 0000\ldots X_2 Y\cdot 2^2 \\ 1100\ldots X_3 Y\cdot 2^3 \\ \hline 10000100\ldots P=\sum_{i=0}^{3} x_i y_i \end{array}$$

(1 bit product)

$P = 132$

HW implementation
- ↳ multi-operand adder : 8-bit
- ↳ 2 4-bit X, Y
- ↳ multiplicand gating

(B) Păstrează produse parțiale fixe

$$\begin{array}{l} 1100\ldots Y \\ 1011 = X_3X_2X_1X_0X \\ \hline 00000000\ldots P_0 := 0 \\ 1100\ldots X_0 Y\cdot 2^0 \ \ 1+ \\ \hline 00001100\ldots P_1 := P_0 + X_0 Y\cdot 2^0 \\ 1100\ldots X_1 Y\cdot 2^1 \\ \hline \end{array}$$

$$\begin{array}{l} 00100100\ldots P_2 := P_1 + X_1 Y\cdot 2^1 \\ 0000\ldots X_2 Y\cdot 2^2 \ \ 1+ \\ \hline 00100100\ldots P_3 := P_2 + X_2 Y 2^2 \\ 1100\ldots X_3 Y\cdot 2^3 \ \ 1+ \\ \hline 10000100\ldots P_4 := P_3 + X_3 Y\cdot 2^3 \\ = P \end{array}$$

Iteration step: $P_{i+1} = P_i + X \cdot Y \cdot 2^i$; $i \geq 0$, $P_0 = 0$

Hardware implementation: - 8 bit
- left shifter (MUX)
- 8 bit adder
- 2 4 bit registers X, Y

© Păstrează un bit al produsului fix:

Iteration step: $\begin{cases} P_i = P_i + X_i Y_i \\ P_{i+1} = P_i * 2^{-1} \end{cases}$; $i \geq 0$, $P_0 = 0$

Hardware implementation:
- 8 bit registers produse parțiale & Shifter
- 4 bit adder
- 2 4-bit registers X, Y

```
   1100....Y
   1011 = X3X2X1X0...X
-------------------------
00000000.......P0 := 0
   1100.......X0·Y  |+
-------------------------
00001100.......P0 := P0 + X0·Y
 00001100......P1 := P0 * 2^-1
   1100.......X1·Y  |+
-------------------------
 00100100......P1 := P1 + X1Y
  00100100.....P2 := P1 * 2^-1
+|   0000.....X2Y
-------------------------
  00100100.....P2 := P2 + X2Y
   00100100....P3 = P2 * 2^-1
+|   1100.....X3Y
-------------------------
  10000100.....P3 := P2 + X3Y
   10000100....P4 := P3 * 2^-1
        = P
```

3.2. Dispozitiv secvențial pt. înmulțirea nr. binare în semn mărime

X, Y - fracționare, semn mărime

$X = X_7 | X_6 X_5 X_4 X_3 X_2 X_1 X_0$

$Y = Y_7 | Y_6 Y_5 Y_4 Y_3 Y_2 Y_1 Y_0$

semn ← magnitudine →

$P = P_{15} | P_{14} P_{13} \ldots P_2 P_1 P_0$

semn ↳ $P_0 = 0$

$P_{15} = X_7 \oplus Y_7$

ALGORITM: Multiplier 2

declare register A[7:0], Q[7:0], M[7:0], COUNT[2:0];

declare bus INBUS[7:0], OUTBUS[7:0];

BEGIN: A := 0, COUNT := 0, } ..... {c0}

INPUT M := INBUS; }

Q := OUTBUS; ..... {c1}

TEST1: if Q[0] = 0 then go to RSHIFT, ... {c2}

ADD: A[7:0] = A[6:0] + M[6:0] } ..... {c3}

RSHIFT: A[7:0] = 0, A[6:0].Q := A.Q[7:1], }

INCREMENT: COUNT := COUNT + 1;

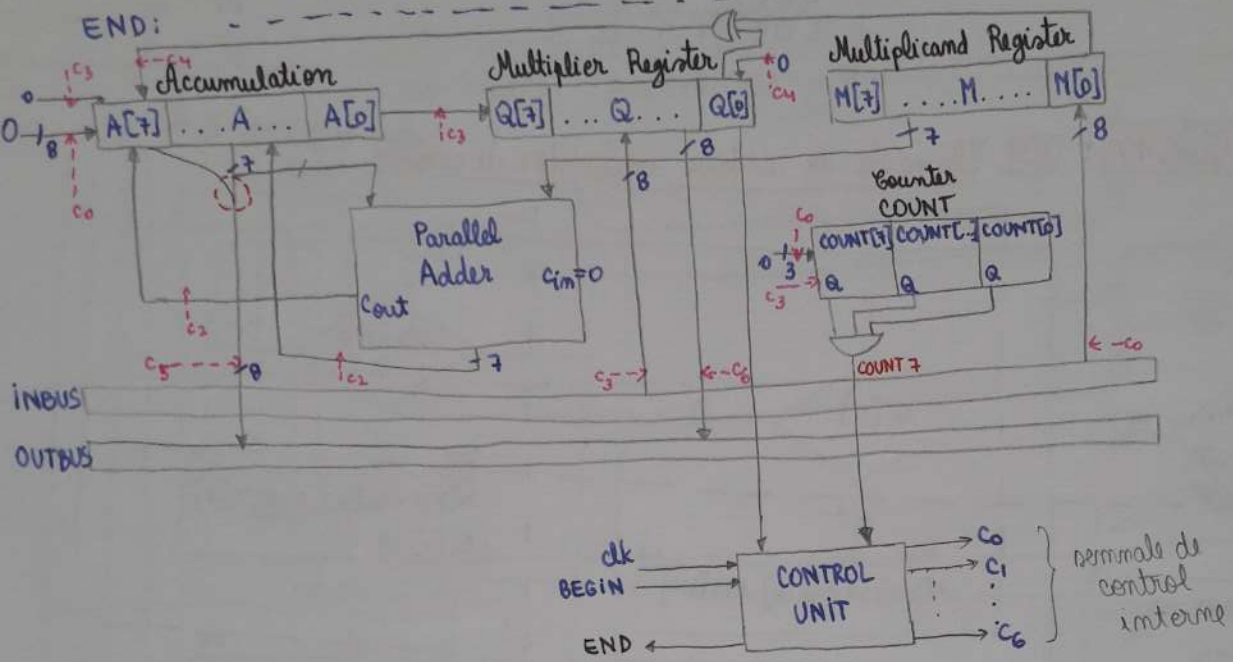
TEST2: if COUNT7 ≠ 1 then go to TEST1, ... {c4}

SIGN: A[7] := M[7] exor Q[0], Q[0] := 0; ..... {c5}

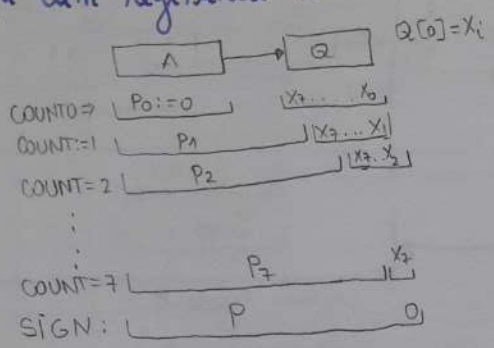
OUTPUT: OUTBUS := A; ..... {c6}

OUTBUS := Q; ..... {END}

END:



Obs: Produsele partiale sunt stocate în reg. A concatenate cu cele mai semnificative poziții din registrul Q.



A	Q(X)	M(X)	COUNT	I.C.S.A.
0000		1111	00	c0
	1101			c1
+1111				
0111				c2
0011	1110		01	c3
0001	1111		10	c3
+111				
1000				
0100	0111		11	c3
0100	0110			c4
0100				c5
	0110			c6

Exemplu 1 $X = -0.625 = -5 * 2^{-3} = 1.101_{SM}$

$Y = -0.875 = -7 * 2^{-3} = 1.111_{SM}$

$P = 0.1000110$

$= 1000011 * 2^{-6} = 35 * 2^{-6}$ ✓

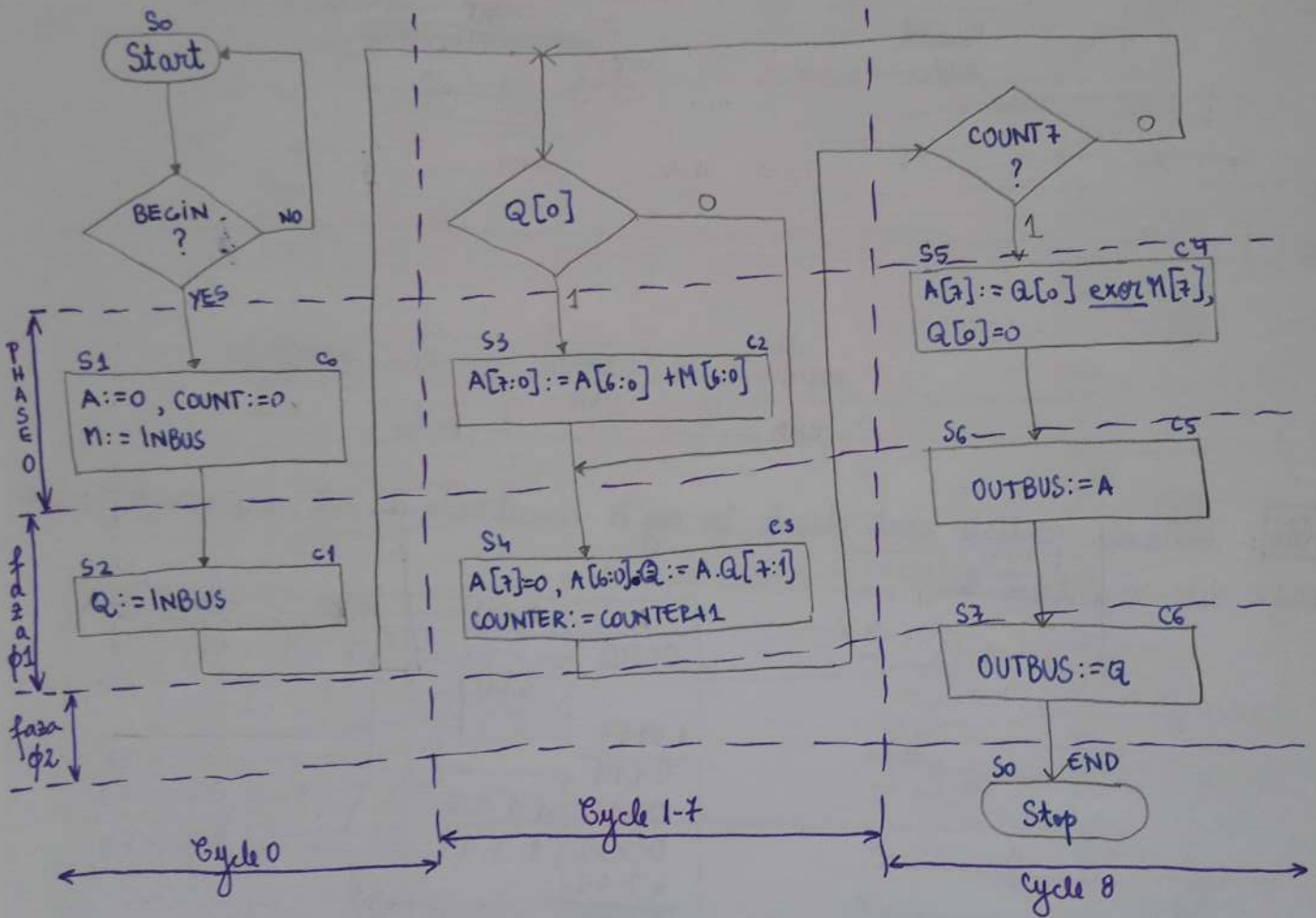
Exemplu 2: $X = -5 = 1101_{SM}$

$Y = +6 = 0110_{SM}$

A	Q	M	COUNT
0000	1101	0110	00
+110			
0110			
0011	0110		01
0001	1011		10
+110			
0111			
0011	1101		11
1001	1110		

$P = 10011110 = -11110 = -30$

CURS 10 : 3.3. Elemente de sinteza unităților de control

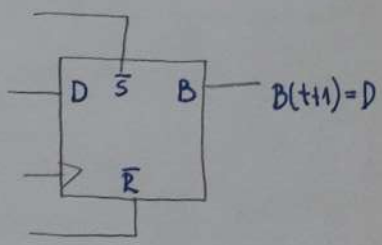


(A) **One Hot**: va asocia fiecărei stări câte un element de stocare propriu

Variabile de stare : $B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0$

Codificarea stărilor:

State	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
S0	0	0	0	0	0	0	0	1
S1	0	0	0	0	0	0	1	0
⋮								
S7	1	0	0	0	0	0	0	0



Ecuațiile de feedback:

$D_0 = B_0 \cdot \overline{BEGIN}$ or B_7

$D_1 = B_0 \cdot BEGIN$

$D_2 = B_1$

$D_3 = B_2 \cdot Q[0]$ or $B_4 \cdot \overline{COUNT7} \cdot Q[0]$

$D_4 = B_2 Q[0]$ or B_3 or $B_4 \cdot \overline{COUNT7} \cdot \overline{Q[0]}$

$D_5 = B_4 \cdot COUNT7$

$D_6 = B_5$

$D_7 = B_6$

Ecuațiile de ieșire:

$END = B_0$

$C_0 = B_1$

$C_1 = B_2$

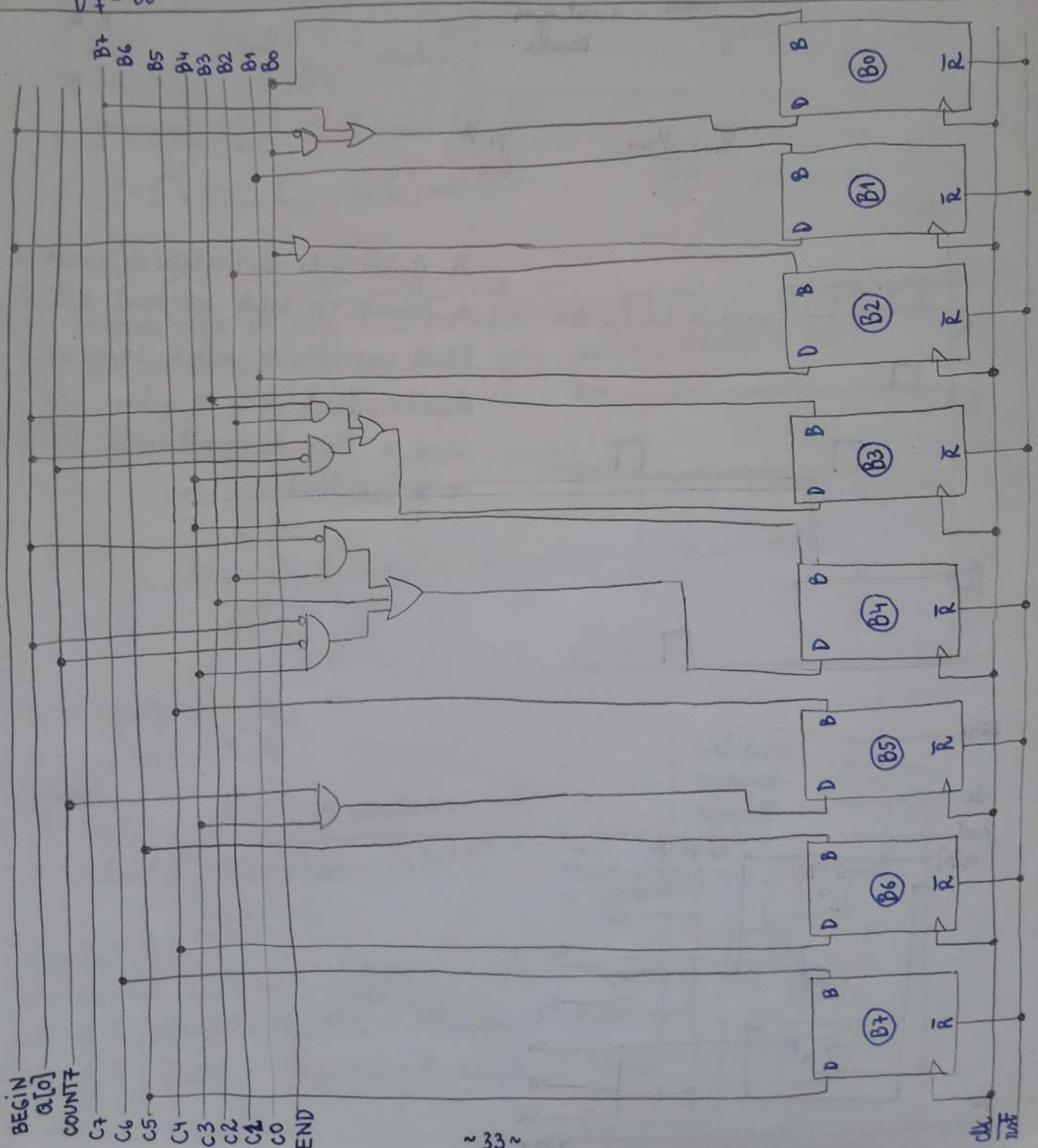
$C_2 = B_3$

$C_3 = B_4$

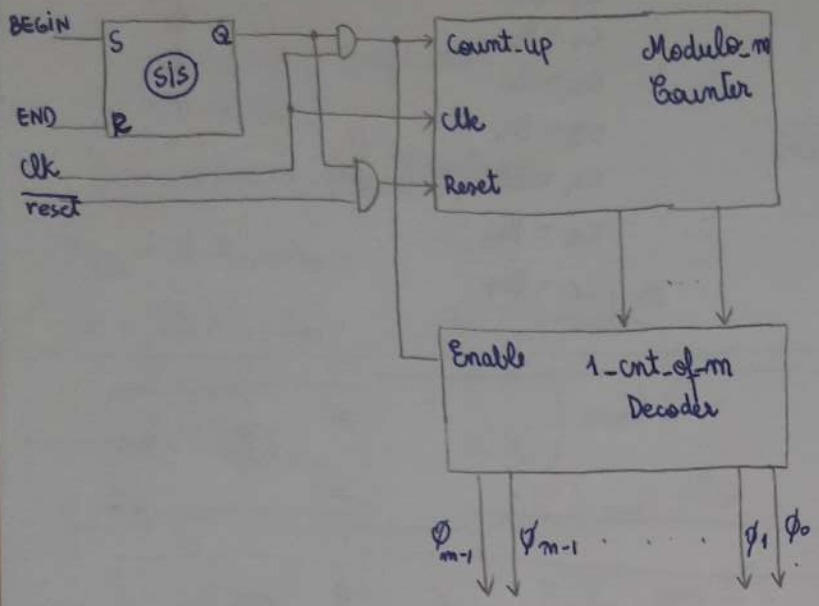
$C_4 = B_5$

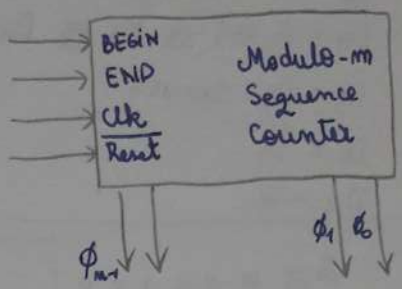
$C_5 = B_6$

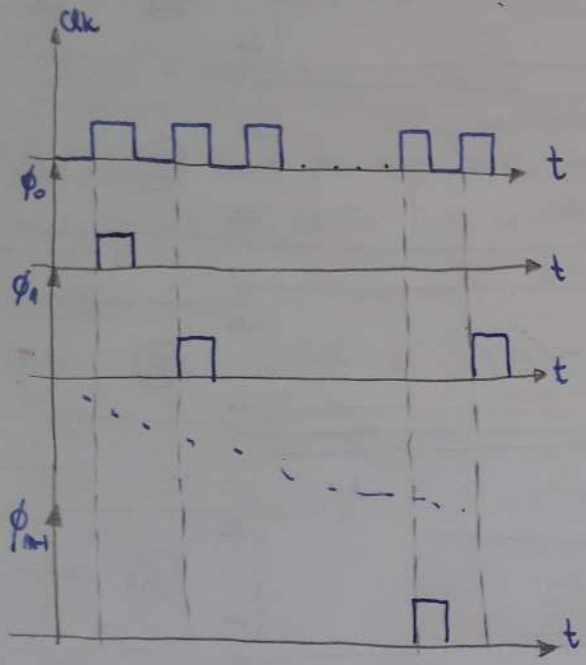
$C_6 = B_7$



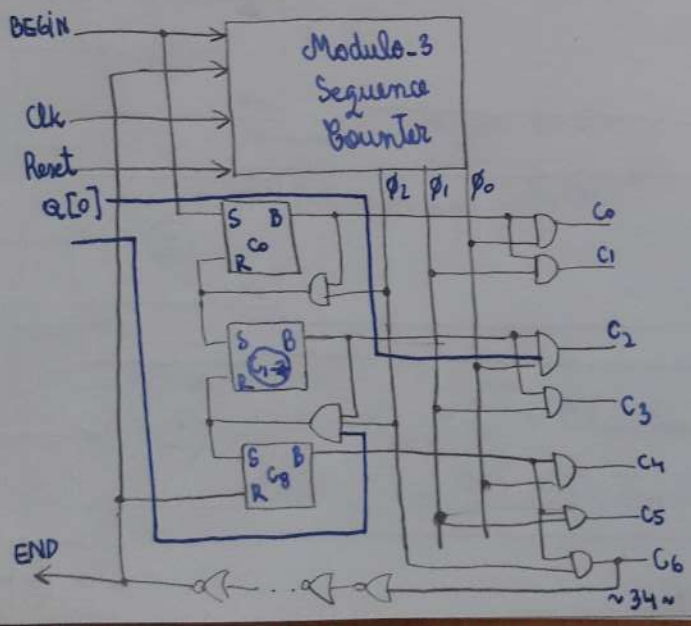
B. Metoda Sequence Counter : va genera impulsuri sincronizate cu clock pe mai multe faze







Pt. fiecare ciclu al unității de control se prevede un latch set-reset distinct Când unitatea de control începe ciclul respectiv, latch-ul se va activa, iar când se trece la următ. ciclu latch-ul va fi dezactivat



3.4 Dispozitiv secvențial de înmulțire binară în C2 după procedura lui Robertson

C2 → SM
* în SM } timpi mari
SM → C2

Interpretarea lui Robertson:

$X_{C2} = 1\,X'_{n-2}X'_{n-3}\ldots X'_1X'_0$

$X = (1\,X'_{n-2}X'_{n-3}\ldots X'_1X'_0) \bmod 2^n$

$= \begin{pmatrix} 100\ldots\ldots00 \\ +0X'_{n-2}X'_{n-3}\ldots\ldots X'_1X'_0 \end{pmatrix} \bmod 2^n =$

$= (2^{n-1} + 0X'_{n-2}X'_{n-3}\ldots X'_1X'_0) \bmod 2^n =$

$= (-2^n + 2^{n-1} + 0X'_{n-2}X'_{n-3}\ldots X'_1X'_0) \bmod 2^n =$

$= -2^{n-1} + \underbrace{0X'_{n-2}X'_{n-3}\ldots X'_1X'_0}_{C_2,\ \text{particular case}}$

Obs: Valoarea unui nr. neg în C_2 se obține scăzând ponderea asociată bitului de semn din val. nr. pozitiv care se obține prin ștergerea bitului de semn.

Exemplu: 1101 $\xrightarrow{*(-1)}$ 0011 $= +3$

1101 $\xrightarrow{ZR}$ $-2^3 + 0101 = -8 + 5 = -3$

10011001 $\xrightarrow{-103}$

10011001 $\xrightarrow{ZR}$ $-2^7 + 2^4 + 2^3 + 2^0 = -128 + 16 + 8 + 1 = -128 + 25 = -103$

CURS 11

X, Y - float

$X = X_{n-1}X_{n-2}\ldots X_1X_0$

- **X < 0**

$P = X * Y$

$= (-X_{n-1} * 2^0 + \overbrace{0X_{n-2}X_{n-3}\ldots X_1X_0}^{X^*,\ \text{poz. în SM}}) * Y = \underbrace{-X_{n-1} * Y * 2^0}_{\text{correction}} + X^* * Y$

X în C2

$X = X_{n-1}X_{n-2}\ldots X_1X_0$

val. of X, integer $= -X_{n-1} * 2^{n-1} + 0X_{n-2}X_{n-3}\ldots X_1X_0$

val. of X, fractal $= -X_{n-1} * 2^0 + 0.X_{n-2}X_{n-3}\ldots X_1X_0$

• [Y<0] p. prod. (produs partial)

Multiplier 3

declare register A[7:0], Q[7:0], M[7:0], COUNT[2:0], F;

declare bus INBUS[7:0], OUTBUS[7:0];

BEGIN: A:=0, COUNT:=0, F:=0, } ---------- {c0}

INPUT: M:=INBUS; ---------- {c1}

Q:=INBUS; ----------

TEST 1: if Q[0]=0 then go to RSHIFT,

ADD: A:=A+M, F:= F or (Q[0] and M[7]); ---------- {c2}

RSHIFT: A[7]:=F, A[6:0].Q = A.Q[7:1], } ---------- {c3}

INCREMENT: COUNT = COUNT+1;

TEST 2: if COUNT7 ≠1 then go to TEST 1,

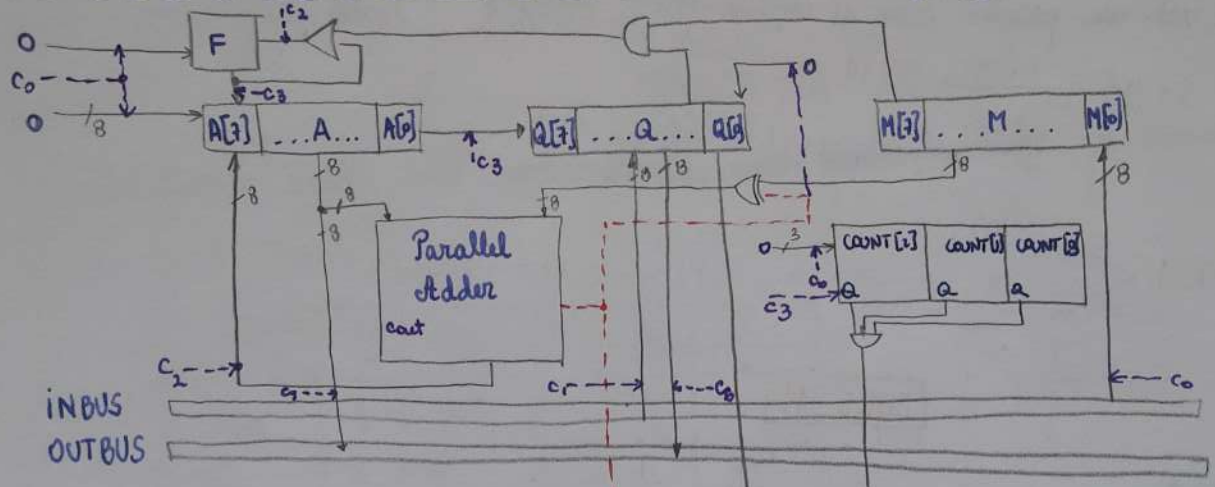
TEST 3: if Q[0]=0 then go to OUTPUT,

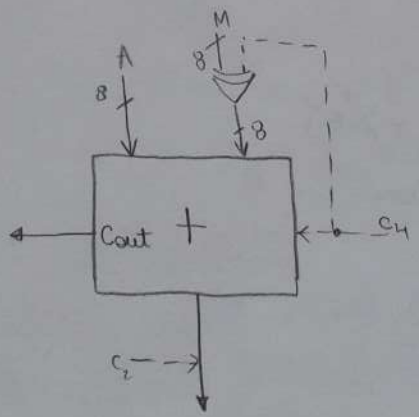
CORRECTION: A:=A-M, Q[0]:=0; ---------- {c2,c4}

OUTPUT: OUTBUS:=A; ---------- {c5}

OUTBUS:=Q; ---------- {c6}

END: ---------- [END]





3.5. Booth

$$\begin{cases} P_i := P_i + X_i Y \\ P_{i+1} = P_i * 2^{-i} \end{cases} \qquad P_0 := 0$$

$X = 0.11111\,10$ → −

Y $2^0 . 2^{-1}$

$$P = X * Y = (2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6}) * Y = Y * \frac{2^0 - 2^{-6}}{2-1} = +2^0 * Y - 2^{-6} * Y$$

$P = X * Y$ → +

$X_i X_{i-1}$ → 01: +Y; 10: −Y; 00, 11: f

$X_0 X_{-1}$ → val 0, weight $\frac{1}{2}$ of X_0

Procedura lui Booth folosește recodificarea Booth: - adăugarea bitului X_{-1}
- scanarea din dreapta spre stânga și înlocuirea fiecărei perechi $X_i X_{i-1}$ cu următorul digit: $X_i X_{i-1}$ → 00: 0; 01: 1; 10: $\bar{1}$ (−1); 11: 0

Recodificarea Booth este redundantă care folosește biți cu semn $(0, 1, \bar{1})$

Exemplu: $X = -1 * 2^{-3}$

Rank / Operand	X_3	X_2	X_1	X_0	X_{-1}
	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
X_{SM}	1	0	0	1	
X_{C_2}	1	1	1	1	0
X_B	0	0	0	1	

$X_B = \bar{1} * 2^{-3} = -1 * 2^{-3}$

$X_{C_2} = = X_B = Y_{C_2} * X_{C_2} = = Y_{C_2} * X_B$

multiplier 4

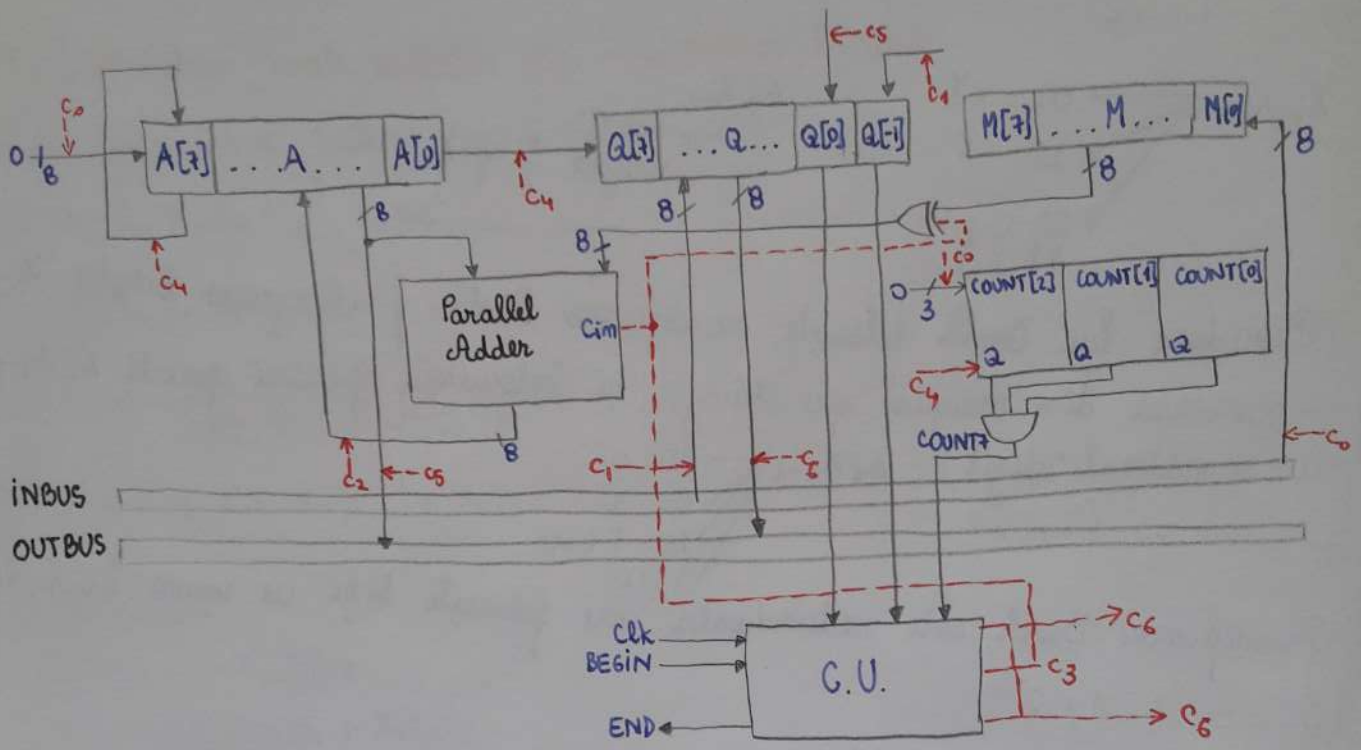
declare register A[7:0], Q[7:-1], M[7:0], COUNT[2:0];

declare bus INBUS[7:0], OUTBUS[7:0];

BEGIN: A:= 0, COUNT:=0; } ← ---- {c_0}

INPUT: M:= INBUS;

```
        Q[7:0] := INBUS[7:0], Q[-1] := 0;                          {c1}
TEST 1: if Q[0]Q[-1] = 01, then A := A+M, go to TEST 2;            {c2}
        if Q[0]Q[-1] = 10, then A := A-M;                          {c2, c3}
TEST 2: if COUNT7 = 1 then go to OUTPUT,                           {c4}
RSHIFT: A[7] := A[7], A[6:0].Q := A.Q[7:0],
INCREMENT: COUNT := COUNT + 1 go to TEST 1;                        {c5}
OUTPUT: OUTBUS := A, Q[0] = 0;                                     {c6}
        OUTBUS[7:0] := Q[7:0];
END:                                                               {END}
```



$X = -5 * 2^{-3} = 1.011_{C_2}$

$Y = -7 * 2^{-3} = 1.001_{C_2}$

A	Q	M	COUNT	i.C.S.A.
0000		1001	00	c_0
	10110			c_1
−1001				
0111				c_2, c_3
0011	11011		01	c_4
0001	11101		10	c_4
+1001				
1010			COUNT7 = 1	c_2
1101	01110		11	c_4
−1001				
0100				c_2, c_3
0100				c_5
	01110			c_6

~38~

$X = +7 = 0111_{c_2}$

$Y = -3 = 1101_{c_2}$

$P = 11101011 =$

$= -128+64+32+11$

$= -21$

A	Q	M	COUNT
0000	0111<u>0</u>	1101	00
-1101			
0011			
0001	1011<u>1</u>		01
0000	1101<u>1</u>		10
0000	01101		11
+1101			
1110	10110		
1110	1011		

3.6 Structuri comb. sistolice pt. înmulțirea binară

X, Y - fără semn, întregi, cu k biți

$X = X_3 X_2 X_1 X_0 = \sum_{i=0}^{3} x_i * 2^i$

$Y = Y_3 Y_2 Y_1 Y_0 = \sum_{j=0}^{3} y_j \cdot 2^j$

$P = X * Y = \left(\sum_{i=0}^{3} x_i \cdot 2^i\right)\left(\sum_{j=0}^{3} y_j \cdot 2^j\right) = \sum_{i=0}^{3} 2^i \left(\sum_{j=0}^{3} x_i \cdot y_j \cdot 2^j\right)$

$P = 2^0 \cdot (X_0 Y_0 2^0 + X_0 Y_1 2^1 + X_0 Y_2 2^2 + X_0 Y_3 2^3) + \ldots + 2^3 (X_3 Y_0 2^0 + X_0 Y_1 2^1 + X_3 Y_2 2^2 + X_3 Y_3 2^3)$

$P = 2^6 X_3 Y_3 +$

$2^5 (X_3 Y_2 + X_2 Y_3) +$

$2^4 (X_3 Y_1 + X_2 Y_2 + X_1 Y_3) +$

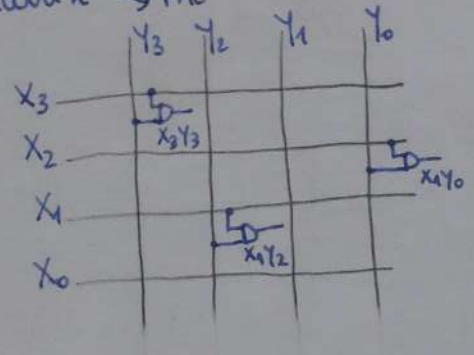
$2^3 (X_3 Y_0 + X_2 Y_1 + X_0 Y_3) +$

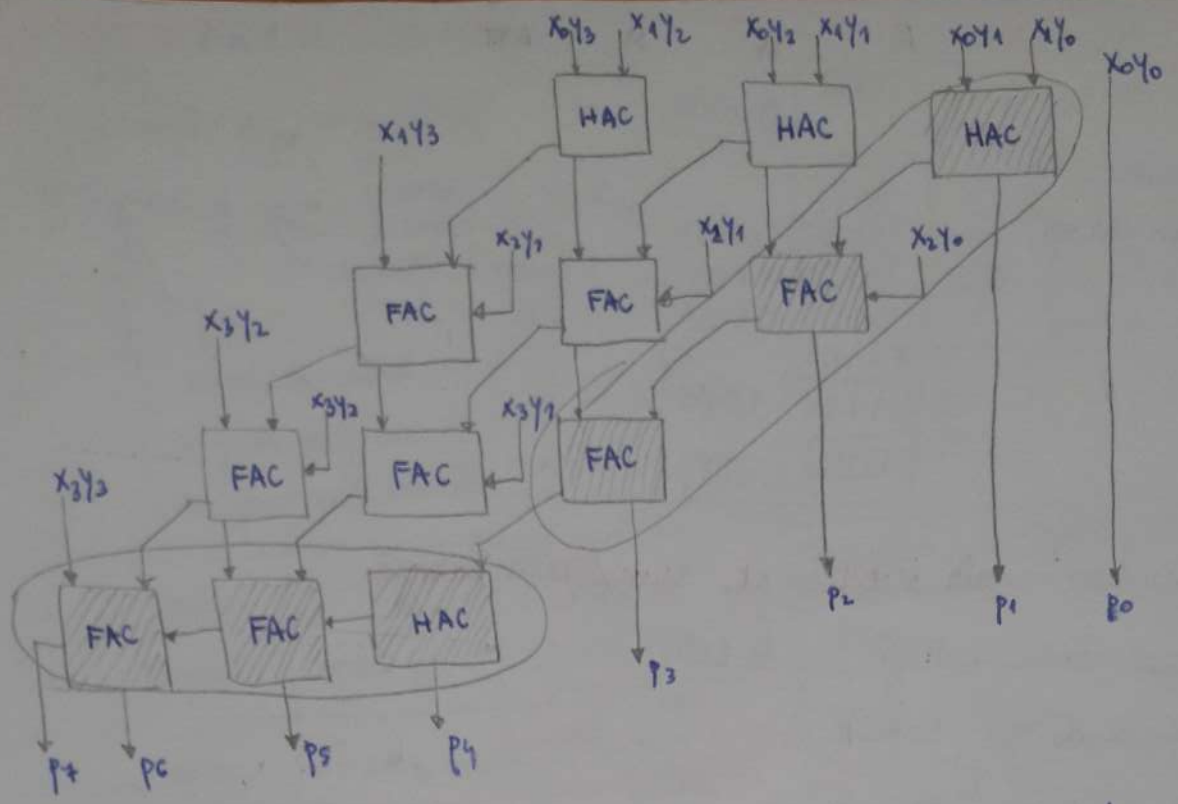
$2^2 \quad (X_2 Y_0 + X_1 Y_1 + X_0 Y_2) +$

$2^1 \quad (X_1 Y_0 + X_0 Y_1) +$

$2^0 \quad X_0 Y_0$

$X_i Y_i$ Matrix ↔ AND FAC



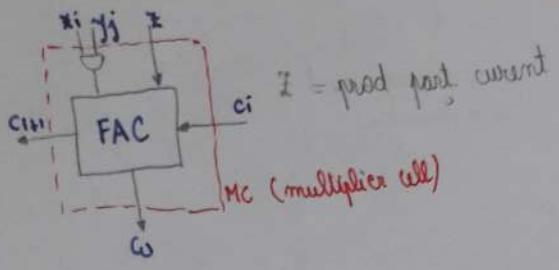


$$D = D_{\substack{AND\\matrix}} + D_{\substack{FAC\\matrix}} = 1d + 2 * \left(\underbrace{1d}_{HAC} + \underbrace{2(m-2)d}_{FAC} \right) = (4m-5)d$$

$$A = A_{\substack{AND\\matrix}} + A_{\substack{FAC\\matrix}} \simeq m^2 A_{\substack{AND\\gate}} + m(m-1) A_{\substack{FAC\\matrix}}$$

$$A = \Theta(m^2)$$

CURS 13



$$\begin{cases} P_{i+1} := P_i + x_i \cdot Y \cdot 2^i \\ P_0 := 0 \end{cases}$$

$D = \underbrace{1d}_{\text{AND}} + 2 \cdot m \cdot 2 \cdot d = (4m+1)d$

$A = m(m+1) \cdot A_{MC} \Rightarrow A = \Theta(m^2)$

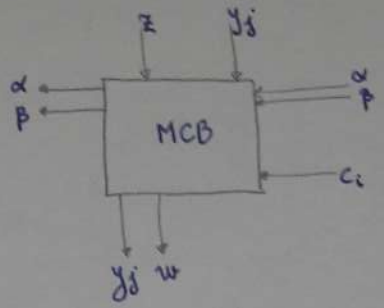
MCB → 3 op. → + / − } carry/borrow chain $x_i x_{i-1}$; no arith. (nicio op.)

→ y_j

→ 2 var. de control: α, β

→ z: prod. part. curent

→ w: prod. part. următ.



Operație	Outputs	
	w	c_{i+1}
Adunare	$w = z \oplus y_j \oplus c_i$	$c_{i+1} = z y_j + z c_i + y_j c_i$
Scădere	$w = z \oplus y_j \oplus c_i$	$c_{i+1} = \bar{z}(y_j + c_i) + y_j c_i$
Nicio operație	$w = z$	irelevant

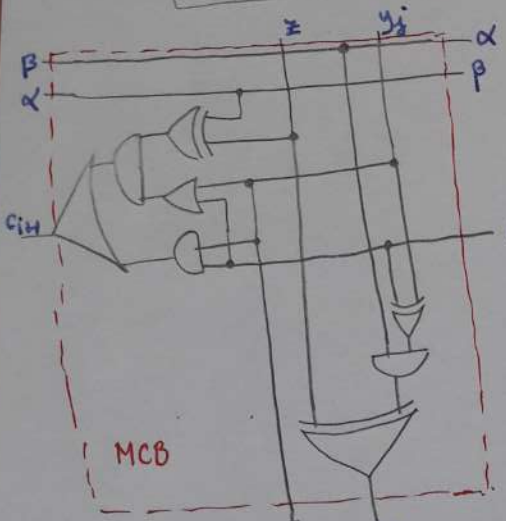
Operation	Control var.	
	α	β
Adunare	1	0
Scădere	1	1
Nicio op.	0	d

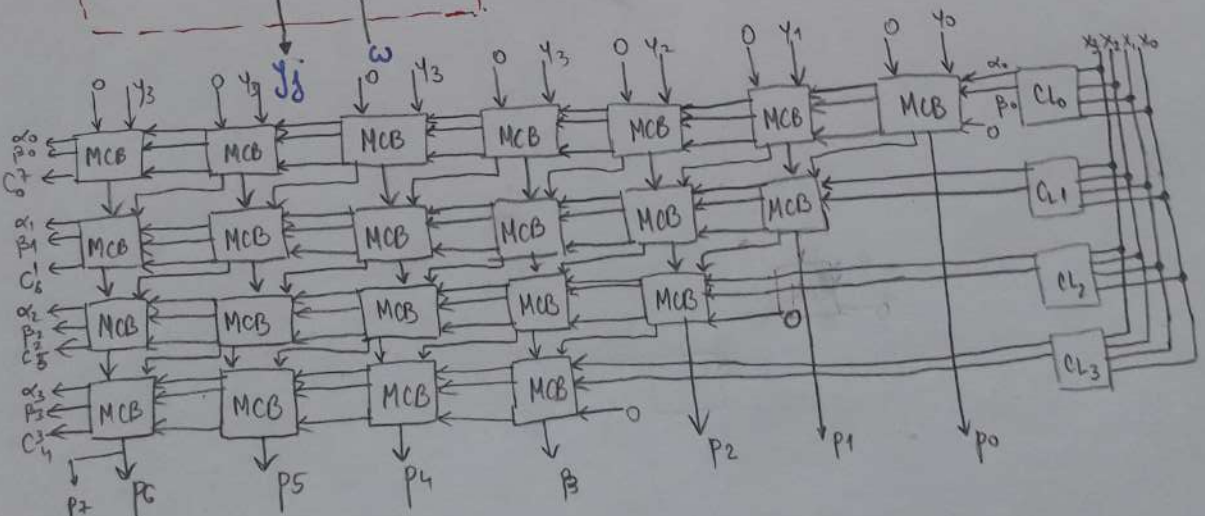
$w = z \oplus \alpha (y_j \oplus c_i)$

$c_{i+1} = (z \oplus \beta)(y_j + c_i) + y_j c_i$

$X = 10\,10\,10$

$\bar{1}\,1\,\bar{1}\,1\,\bar{1}$





X = +5 = 0101

Y = -6 = 1010

```
0000000 ---------- P0
1111010 ---------- Y·2^0 | -
0000110 ---------- P1 := P0 - Y·2^0
 11 1010 --------- Y = 2^1
1111010 ---------- P2 := P1 + Y·2^1
 1 1010 ---------- Y 2^2
0010010 ---------- P3 = P2 - Y·2^2
 1010 ------------ Y·2^3 | +
1100010 ---------- P4 := P3 + Y·2^3
```

P = 11100010 = -128 + 64 + 32 + 2 = -30

α_0, β_0

⋮

α_3, β_3

INPUT				Booth				outputs							
x_3	x_2	x_1	x_0	x_{B3}	x_{B2}	x_{B1}	x_{B0}	α_3	β_3	α_2	β_2	α_1	β_1	α_0	β_0
0	0	0	0	0	0	0	0	0	d	0	d	0	d	0	d
0	0	0	1	0	0	1	$\bar{1}$	0	d	0	d	1	0	1	1
0	0	1	0	0	1	$\bar{1}$	0	0	d	1	0	1	1	0	d
0	0	1	1	0	1	0	$\bar{1}$	0	d	1	0	0	d	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	1	0	0	0	$\bar{1}$	0	0	0	d	1	1	0	d	0	d
1	1	0	1	0	$\bar{1}$	1	$\bar{1}$	0	d	1	1	1	0	1	1
1	1	1	0	0	0	$\bar{1}$	0	0	d	0	d	1	1	0	d
1	1	1	1	0	0	0	$\bar{1}$	0	d	0	d	0	d	1	1

$\beta_0 = 1$

$\alpha_0 = x_0$

$\beta_1 = x_1$

$\alpha_1 = x_1 \oplus y_0$

$\beta_2 = x_2$

$\alpha_2 = x_2 \oplus x_1$

$\beta_3 = x_3$

$\alpha_3 = x_3 \oplus x_1$