Arhitectura calculatoarelor 23.10.2024

(curs 4 - S4)

Half Adder Cell (HAC)

Dacă $c_0 = 0$ => cea mai din dreapta FAC de la RCA poate fi simplificată

Ecuațiile ieșirilor

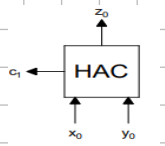
$$z_0 = x_0 \oplus y_0 \oplus \cancel{c_0} = x_0 \oplus y_0$$

$$c_1 = x_0 y_0 + \cancel{x_0 c_0} + \cancel{y_0 c_0} = x_0 y_0$$

- necesara din punct de vedere al modularitatii
- de ce sa nu simplific daca stiu ca mereu c0 va fi 0?

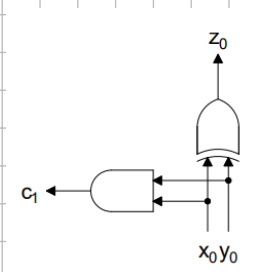
$0 \oplus \alpha = \alpha$

Simbol:



Sinteza HAC

(A) porți de tip EXOR, AND:



(B) porți de tip NOR

-> se pornește de la z_0

$a \oplus b = \bar{a} b + a \bar{b}$

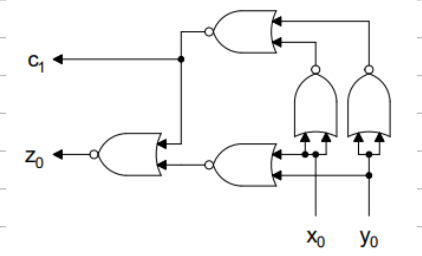
$$z_0 = x_0 \oplus y_0 = x_0 \bar{y}_0 + \bar{x}_0 y_0 = x_0(\bar{x}_0 + \bar{y}_0) + y_0(\bar{x}_0 + \bar{y}_0) =$$

$x_0 \bar{x}_0 = 0$

$$= (\overline{x_0} + \overline{y_0})(x_0 + y_0) = \frac{}{(\bar{x}_0 + \bar{y}_0)(x_0 + y_0)} =$$

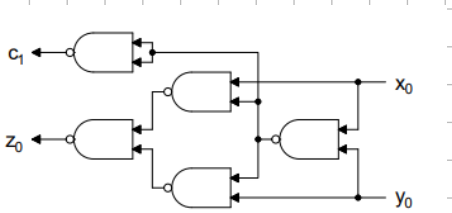
$\overline{a \cdot b} = \bar{a} + \bar{b}$

$$= \frac{}{\overline{\bar{x}_0 + \bar{y}_0} + \overline{x_0 + y_0}}$$



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

c) porți de tip NAND



dublu negat

$$z_0 = x_0 \oplus y_0 = x_0 \cdot \overline{y_0} + \overline{x_0} \cdot y_0 = x_0 \cdot (\overline{x_0} + \overline{y_0}) + y_0 \cdot (\overline{x_0} + \overline{y_0})$$
$$= \overline{\overline{x_0 \cdot (\overline{x_0} + \overline{y_0}) + y_0 \cdot (\overline{x_0} + \overline{y_0})}} = \overline{\overline{x_0 \cdot \overline{x_0} \cdot \overline{y_0}} \cdot \overline{y_0 \cdot \overline{x_0} \cdot \overline{y_0}}}$$

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

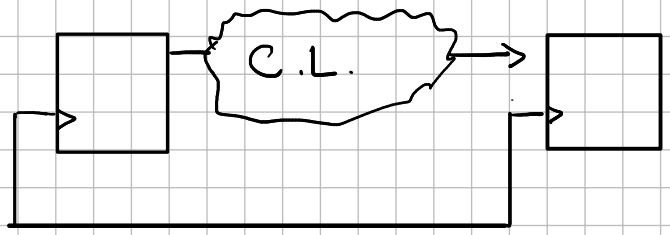
apoi folosim $\overline{a} + \overline{b} = \overline{a \cdot b}$

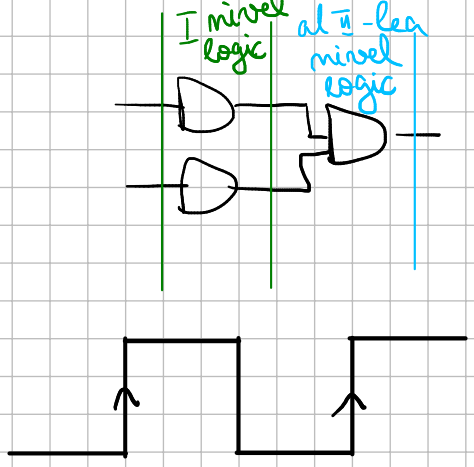
⟶ NAND este mai rapid decât NOR

Calea critică

⟶ calea de propagare din întreg circuitul corespunzătoare întârzierii maxime de propagare a semnalelor

- orice element de circuit furnizeaza semnalele de iesire cu o intarziere in raport cu semnalele de la intrare





Ipoteze simplificatoare

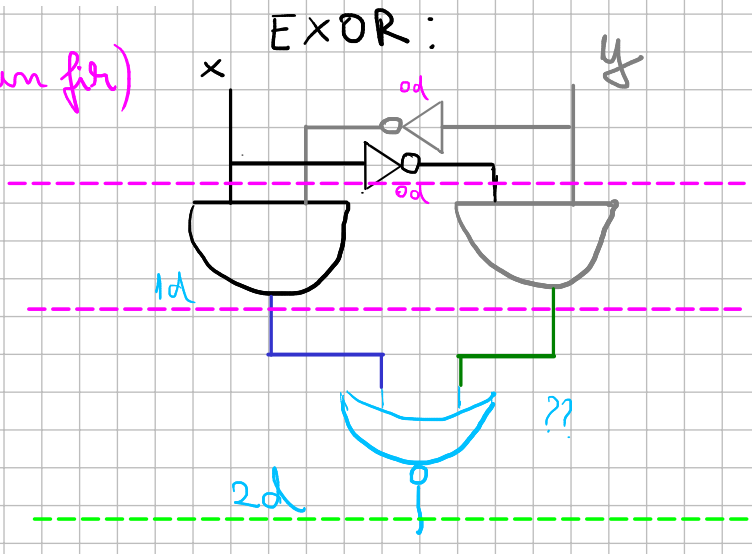
- orice poarta primitiva are latența *1d* (o unitate de timp)
  - indiferent de numărul de intrări si timpul portii primitive
- inversoarele nu introduc întârzieri (au intarziere *0d*)
- porțile EXOR au latență de *2d* (Q: de ce ?)
- toți operanzii sunt disponibili la momentul *0d*

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

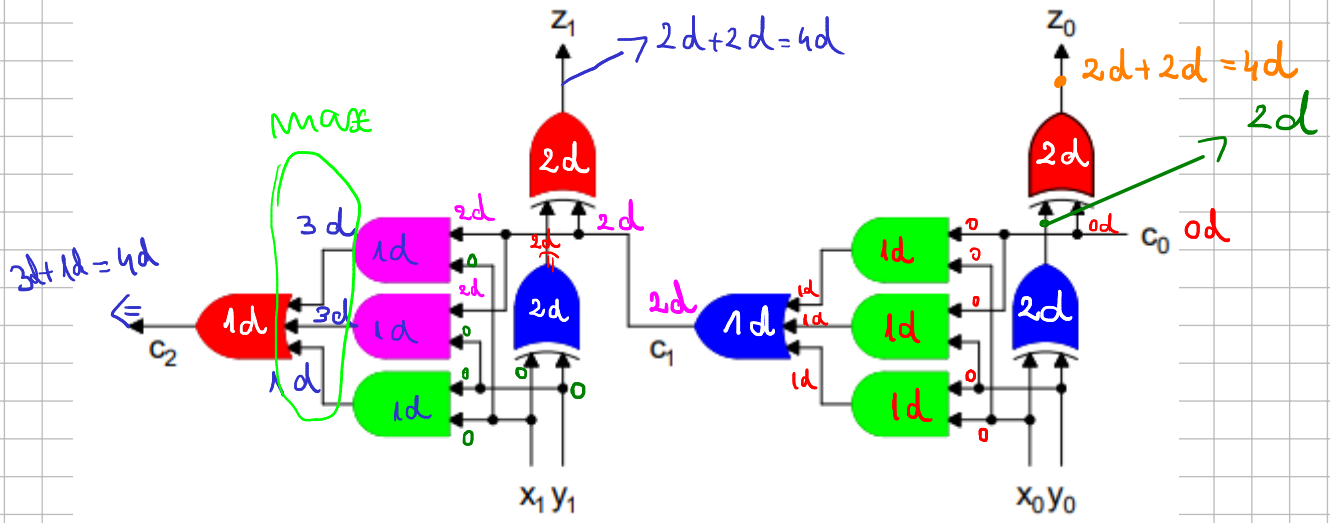
INV ⟶ 0d (ca un fir)

AND, NAND, OR, NOR } 1d

EXOR ⟶ 2d



Calea critică pentru RCA pe 2 Biți



EXOR = 2d

altele = 1d

intrarile sunt conectate la momentul 0d

Întârizerea unui segment RCA pe n biți:
$D^{c_{out}}_{RCA} = 2nd$
$D^{z}_{RCA} = 2nd$

Excepție $n=1$

Condiții speciale ale adunării

1. rezultat nul
2. carry out generat din MSB → 1000 + 1010 = 1 0010
3. rezultat negativ
4. overflow

Overflow aritmetic:
- rezultatul operației aritmetice depășește capacitatea de stocare

Overflow aritmetic la operarea numerelor fără semn:
- se consideră $X = 35$, $Y = 33$ fără semn, pe 6 biți

(storing capacity)

```
X=35:  1 0 0 0 1 1
X=33:  1 0 0 0 0 1   +
       1 0 0 0 1 0 0   =4  (overflow)
```

- dacă X și Y erau reprezentați pe 7 biți:

(storing capacity)

```
X=35:  0 1 0 0 0 1 1
X=33:  0 1 0 0 0 0 1   +
       1 0 0 0 1 0 0   =68
```

Notă: Overflow-ul la operarea numerelor fără semn apare când este generat un transport din Most Significant Bit (MSB).

7 biți

x = 90 1 0 1 1 0 1 0

y = 100 1 1 0 0 1 0 0

1 0 1 1 1 1 0

fără c_0 = 62

dacă am avea 8 biți ✓

Overflow aritmetic la operarea numerelor cu semn (C2):

- se consideră $X = +19$, $Y = +14$ fără semn, pe 6 biți

storing capacity

X=+19: $0\,1\,0\,0\,1\,1_{C2}$ |+
X=+14: $0\,0\,1\,1\,1\,0_{C2}$ |
$1\,0\,0\,0\,0\,1_{C2}$ =~~31~~ (overflow)

- dacă X și Y erau reprezentați pe 7 biți:

storing capacity

X=+19: $0\,0\,1\,0\,0\,1\,1_{C2}$ |+
X=+14: $0\,0\,0\,1\,1\,1\,0_{C2}$ |
$0\,1\,0\,0\,0\,0\,1_{C2}$ =+33

Note: Overflow-ul la operarea numerelor cu semn apare când adunărea a doua numere de acelasi semn produce un rezultat de semn contrar.

Întrebare: Poate genera overflow adunarea a două numere de semne diferite? → Nu: $|a| - |b| = c$, $|c| \leq \max(|a|, |b|)$

Determinarea condiției de overflow la adunarea numerelor cu semn

- operanzii X și Y, pe n biți, în C2
- Z: rezultatul adunării lui X și Y
- semnele celor 3 numere: x_{n-1}, y_{n-1} și z_{n-1}
- simbol overflow: v

S: 0 0 1
S: 1 1 0
⇒ overflow

Tabel de adevar pentru determinarea condiției de overflow:

Inputs			Outputs	
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

Overflow

(I) $(A \oplus B)C = AC \oplus BC$

(III) $A \oplus B = (A + B) \ominus AB$

(II) $A + B = A \oplus B \oplus AB$

$x \oplus 1 = \overline{x}$

Din tabel:

$$v = \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot c_{n-1} + x_{n-1} \cdot y_{n-1} \cdot \overline{c_{n-1}}$$

$$\nu = \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot c_{n-1} + x_{n-1} \cdot y_{n-1} \cdot \overline{c_{n-1}}$$

$$\overset{I_2}{=} \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \cdot \overline{c_{n-1}}$$

$$= \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \cdot (1 \oplus c_{n-1})$$

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

$$\overset{I_1}{=} (\overline{x_{n-1}} \cdot \overline{y_{n-1}} \oplus x_{n-1} \cdot y_{n-1}) \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1}$$

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

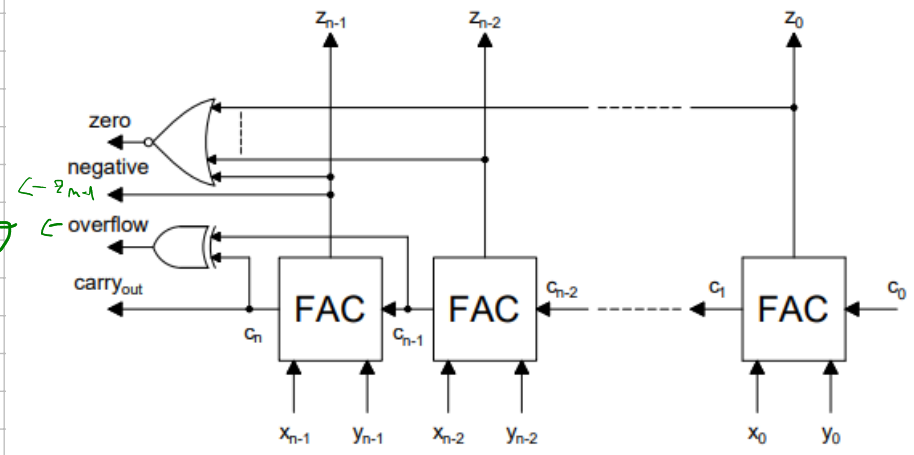
$$= (x_{n-1} \oplus y_{n-1} \oplus 1) \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1}$$

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

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

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

Sumator RCA pentru numere pe n biți cu generarea condițiilor speciale ale adunării:



Adunarea cu o constantă

→ se consideră doar constante impare

ex: counter

→ X, Y - pe n biți

→ Y - constant

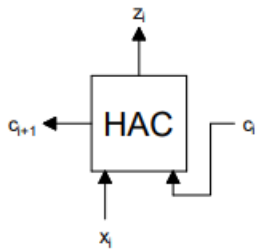
$$\begin{cases} X = x_{n-1}x_{n-2}\cdots x_0 \\ Y = y_{n-1}y_{n-2}\cdots y_0 \\ Z = X + Y \end{cases}$$

doar constante impare -> LSB va fi mereu 1

constantele pare -> am face doar adunari triviale, y0 = 0

dacă $y_i = 0$:

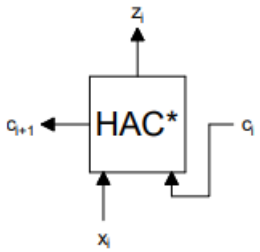
$$\left\{ \begin{array}{rcl} z_i = & x_i \oplus 0 \oplus c_i = & x_i \oplus c_i \\ c_{i+1} = & x_i \cdot 0 + x_i \cdot c_i + 0 \cdot c_i = & x_i \cdot c_i \end{array} \right\} \text{HAC}$$



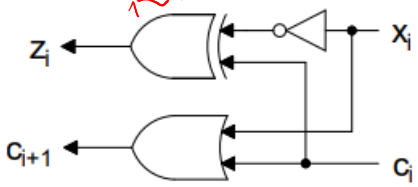
dacă $y_i = 1$:

$$\left\{ \begin{array}{rcl} z_i = & x_i \oplus 1 \oplus c_i = & \overline{x_i} \oplus c_i \\ c_{i+1} = & x_i \cdot 1 + x_i \cdot c_i + 1 \cdot c_i = & x_i + c_i \end{array} \right\} \text{HAC*}$$

$1 \oplus \alpha = \overline{\alpha}$



$c_i(1+x_i)$



Exemplu de adunare cu o constantă având operanzi pe 6 biți:

- $X = x_5x_4x_3x_2x_1x_0$
- $Y = y_5y_4y_3y_2y_1y_0$ - operand constant
  - fie $Y = 110100_2$ ← 52 → $-32+20 = -12$ cu semn
- $Z = X + Y$, cu $c_0 = 0$

$z_0 = x_0 \oplus 0 \oplus c_0 = x_0 \oplus 0 = x_0$

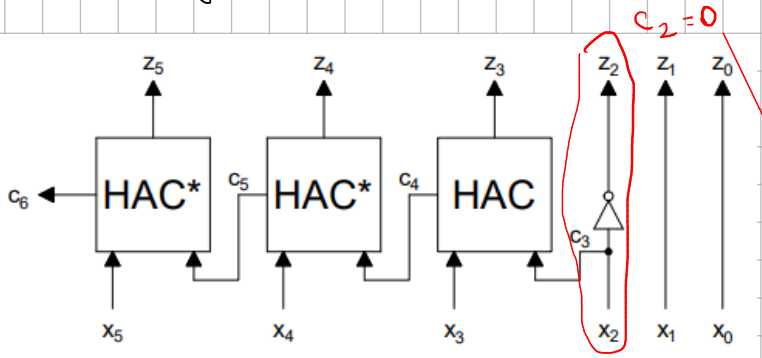
$z_1 = x_1 \oplus 0 \oplus c_1 = x_1 \oplus 0 = x_1$

$z_2 = x_2 \oplus 1 \oplus c_2 = x_2 \oplus 1 = \overline{x_2}$ (y_2)

$c_1 = x_0 \cdot 0 + x_0 \cdot c_0 + 0 \cdot c_0 = 0$ ($c_0 = 0$)

$c_2 = x_1 \cdot 0 + x_1 \cdot c_1 + 0 \cdot c_1 = 0$

$c_3 = x_2 \cdot 1 + x_2 \cdot c_2 + 0 \cdot c_2 = x_2$ (y_2)



$c_2 = 0$

$y_3 = 0 \rightarrow$ HAC

$y_4 = 1 \rightarrow$ HAC*

$y_5 = 1 \Rightarrow$ HAC*

De asta ne folosim HAC*

ex : cours pt 1 0 1 1 1 0 = −32+14 = −18

2.2.2. Sumatoare zecimale bazate pe propagarea serială a transportului

(A) Adunare BCB:

⟶ Fie x_i, y_i și z_i — cifre BCB, unde $z_i = x_i + y_i$

$X_i + Y_i \begin{cases} Z_i & : \text{cifra sumă} \\ c_{i+1} & : \text{transportul către cifra mai semnificativă} \end{cases}$

ex:

$$\begin{array}{r} 9+ \\ 5 \\ \hline (1)\,4 \end{array} \quad \rightarrow c_{i+1}$$

$$x_i = x_3 x_2 x_1 x_0$$

$$y_i = y_3 y_2 y_1 y_0$$

$$z_i = z_3 z_2 z_1 z_0$$

$$\text{dacă } X_i + Y_i < 10 \begin{cases} Z_i = X_i + Y_i \\ c_{i+1} = 0 \end{cases}$$

$$\text{dacă } X_i + Y_i \geq 10 \begin{cases} Z_i = X_i + Y_i - 10 \\ c_{i+1} = 1 \end{cases}$$

=> unified architecture for both cases

$\boxed{-10}$ => pas de corecție

Daca adunam 2 numere pe 4 biti si se obtine un rezultat pe 5 biti: (pt. a evita overflow)

$$x_i + y_i = c^* z_3^* z_2^* z_1^* z_0^*$$

$$\Rightarrow c^* z_3^* z_2^* z_1^* z_0^* \geq 10$$

* -> adaugata pt ca avem rezultat intermediar
=> nu ele vor fi cifra suma si cifra transport, pe baza lor le vom calcula

Putem rescrie această inegalitate

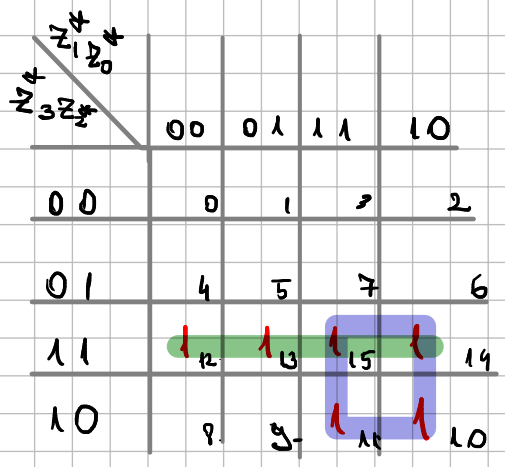
$$\begin{cases} 10 \leq c^* z_3^* z_2^* z_1^* z_0^* < 16 & C1 \\ c^* z_3^* z_2^* z_1^* z_0^* \geq 16 & C2 \end{cases}$$

SAU

C_1 implică

$$\begin{cases} c^* = 0 \\ z_3^* z_2^* z_1^* z_0^* \geq 10 \end{cases}$$

Pentru rezolvarea inegalității rezolvăm următoarea minimizare:



provenită din tabelul:

z_3^*	z_2^*	z_1^*	z_0^*	f
0	0	0	0	0
-	-	-	-	-
1	1	1	1	1

>10

12, 13, 15, 14 : $z_3^* z_2^*$

15, 14, 11, 10 : $z_3^* \cdot z_1^*$

$$\Rightarrow f = z_3^* z_2^* + z_3^* z_1^* = 1$$

⟶ Putem rescrie condiția C_1:

$$\overline{C^*}\,(z_3^* z_2^* + z_3^* z_1^*)$$

pentru a forta val 0 -> negatie si & cu f

⟶ Inegalitatea asociată condiției C_2 este adevărată dacă

$$\boxed{C^* = 1} \Leftarrow$$

Conditia simpla a fost impartita in C1 si C2 => pentru nr ≥16 avem nevoie de 5 biti

Prin disjuncție C_1 sau C_2

$$\Rightarrow x_i + y_i \geq 10 \equiv C^* + \overline{C^*} \cdot (z_3^* z_2^* + z_3^* z_1^*)$$

$$= C^* + z_3^* z_2^* + z_3^* z_1^*$$

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

Scăderea lui 10 pe 4 biți

$z_i = (x_i + y_i - 10) \longrightarrow$ rezultat pe 4 biți

$z_i = (x_i + y_i - 10) \bmod 2^4$

$a \bmod b = (a+b) \bmod b = (a-b) \bmod b$

$\Rightarrow z_i = (x_i + y_i + 16 - 10) \bmod 2^4 =$

$= (x_i + y_i + 6) \bmod 2^4$

=> Scaderea poate fi implementata prin adunarea lui 6, ignorand transportul de iesire din rangul cel mai semnificativ

Ex:

15	−	=	1111	−		15	+	=	1111	+	
10		=	1010			6		=	0110		
5		=	0101			5		=	1 0101		
18	−	=	1 0010	−		18	+	=	1 0010	+	
10		=	1010			6		=	0110		
8		=	1000			8		=	1 1000		

13 − 10 = 3 : 1101 − 1010 = 0011

19 − 10 = 9 : 1 0011 − 1010 = 1001

13 + 6 : 1101 + 0110 = 0011 (*) = 3

19 + 6 : 0011 + 0110 = 1001 (*)

O alta explicatie pentru echivalenta -10 <=> +6

Pe 10 îl scad dintr-un nr ≥ 10 $\quad x_i + y_i \geq 10,\ z_i = x_i + y_i - 10$

=> lucrăm cu nr. fără semn

$c^* z_3^* z_2^* z_1^* z_0^* -$

1	0	1	0

Dacă interpretăm ca nr C_2

1010_{C_2} sau => 0110_{C_2} unsigned

$\Rightarrow 1110_{SM} = -6$ (semn mărime)

$c^* z_3^* z_2^* z_1^* z_0^* - (-6_{C_2}) = c^* z_3^* z_2^* z_1^* z_0^* + 6_{C_2}$

→ Corectia lui Z_i depinde de următoarea condiție booleană

$C^* + z_3^* z_2^* + z_3^* z_1^*$

1 $(x_i + y_i \geq 10)$

$Z_i = z_3^* \; z_2^* \; z_1^* \; z_0^* + (6)$

$0 \quad 1 \quad 1 \quad 0$

$C_{i+1} = 1$

0 $(x_i + y_i < 10)$

O singura cifra

$Z_i = z_3^* \; z_2^* \; z_1^* \; z_0^* + (0)$

$0 \quad 0 \quad 0 \quad 0$

$C_{i+1} = 0$

Transportul se obține ca:

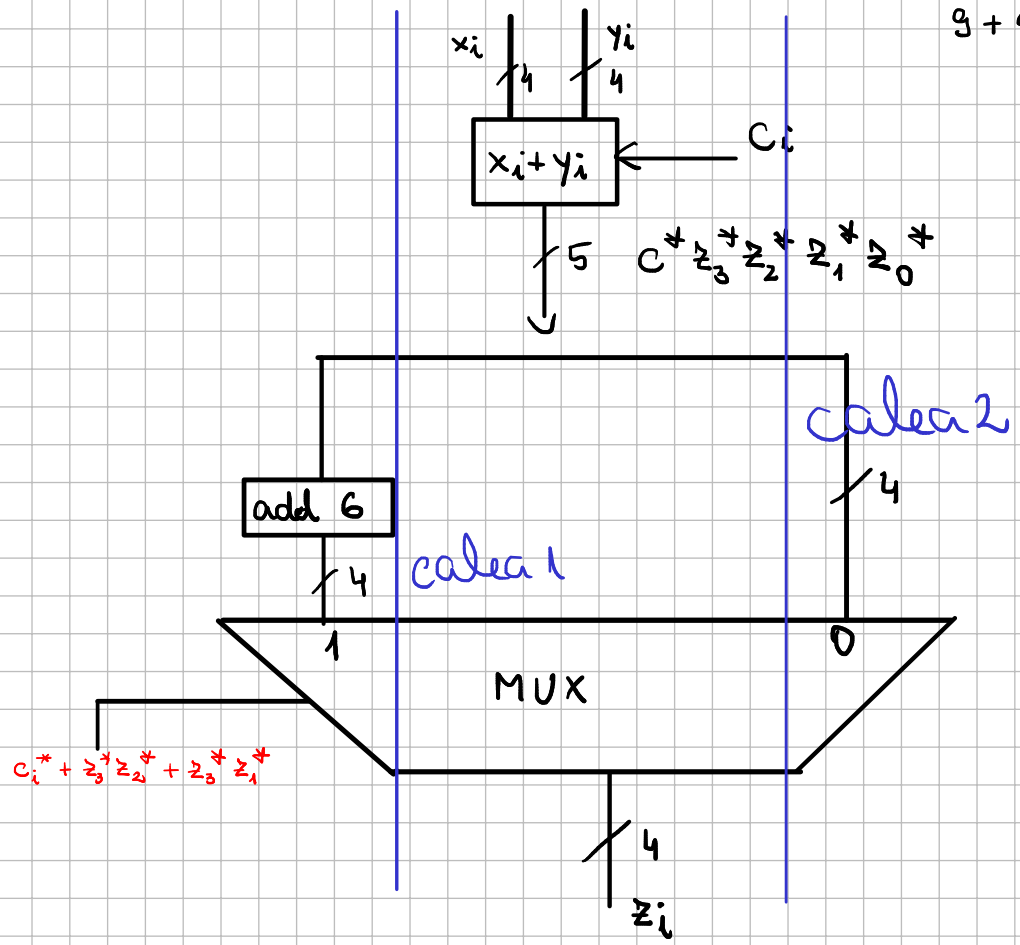
$$C_{i+1} = C_i^* + z_3^* z_2^* + z_3^* z_1^*$$

Trebuie sa am un sumator care ne va aduna 6 in cazul de sus.

El nu poate fi exclus din situatia de jos decat prin:

$Z_i = x_i + y_i + C_i \leq 19$

$9 + 9 + 1$ (maxim)

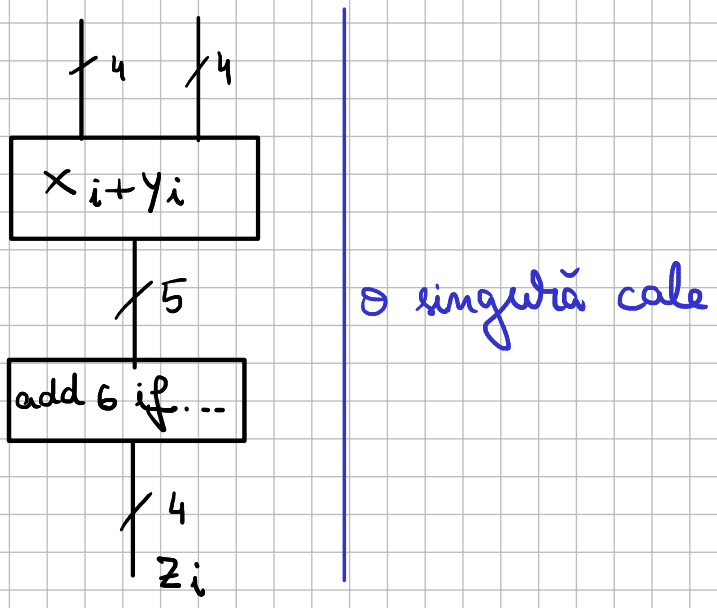


Are 2 probleme : 1. Am o unitate suplimentara pe calea lui Zi (MUX) => delay la obtinere rezultat
2. latenta acestui dispozitiv depinde de calea pe care ma duc

latență = max(calea 1 , calea 2)

worst case

Cea mai buna solutie ar fi urmatoarea :



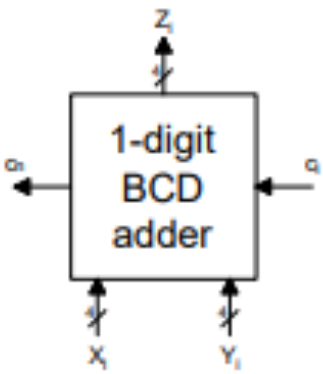
Cea mai simpla solutie e sa adun 0 daca nu am de facut corectii, transportul de iesire va fi 0

Stagiul de corecție pentru z_i devine

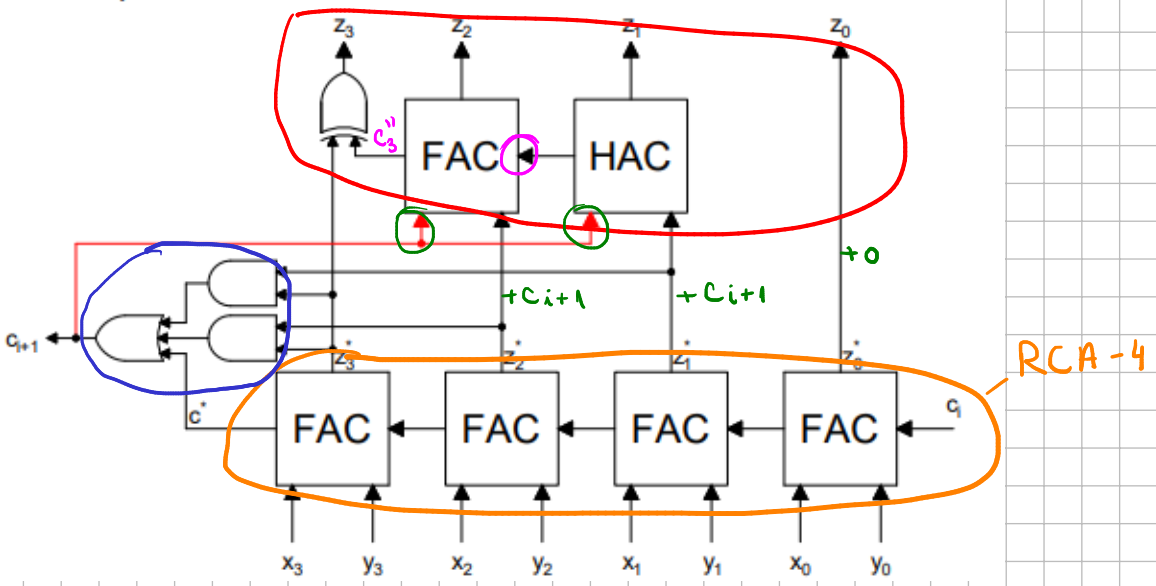
z_i =	z_3^*	z_2^*	z_1^*	z_0^*	+
	0	c_{i+1}	c_{i+1}	0	

Implementare:

Simbol :



Sumatorul pentru tetrade BCD:



$C_{i+1} = C^* (z_3^* z_2^* + z_3^* z_1^*)$ ⟶ expresia încercată

Nivel inferior ⟶ calculează valorile intermediare (suma intermediară)

Nivel de corecție / Stagiu de corecție

Folosesc HAC pentru ca am de adunat doar 2 biti, unde pot obtine si un bit de iesire

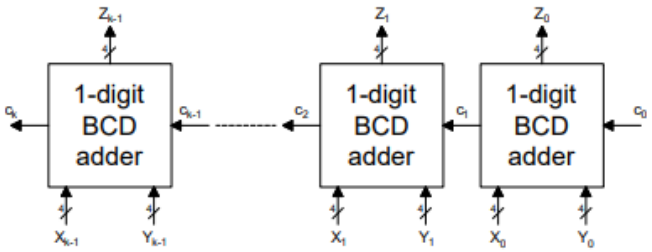
In modulul FAC adunam la Z2* atat Ci+1 cat si transportul provenit de la HAC

De aici rezulta si transportul pentru Z3, obtinut prin adunarea Z3* cu c3"

$z_3^* + c_3''$

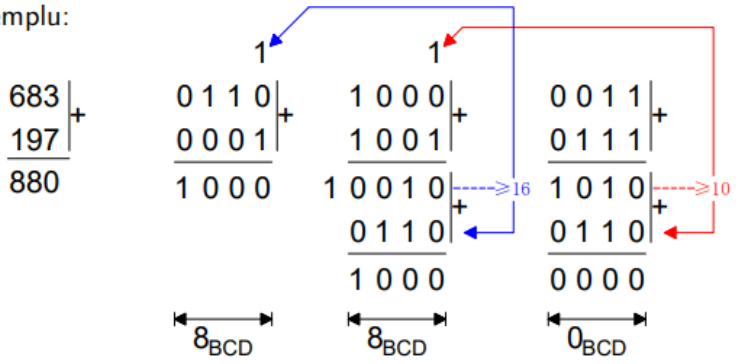
Ignoram noul transport, nu ne trebuie HAC => doar poarta SAU-EXCLUSIV

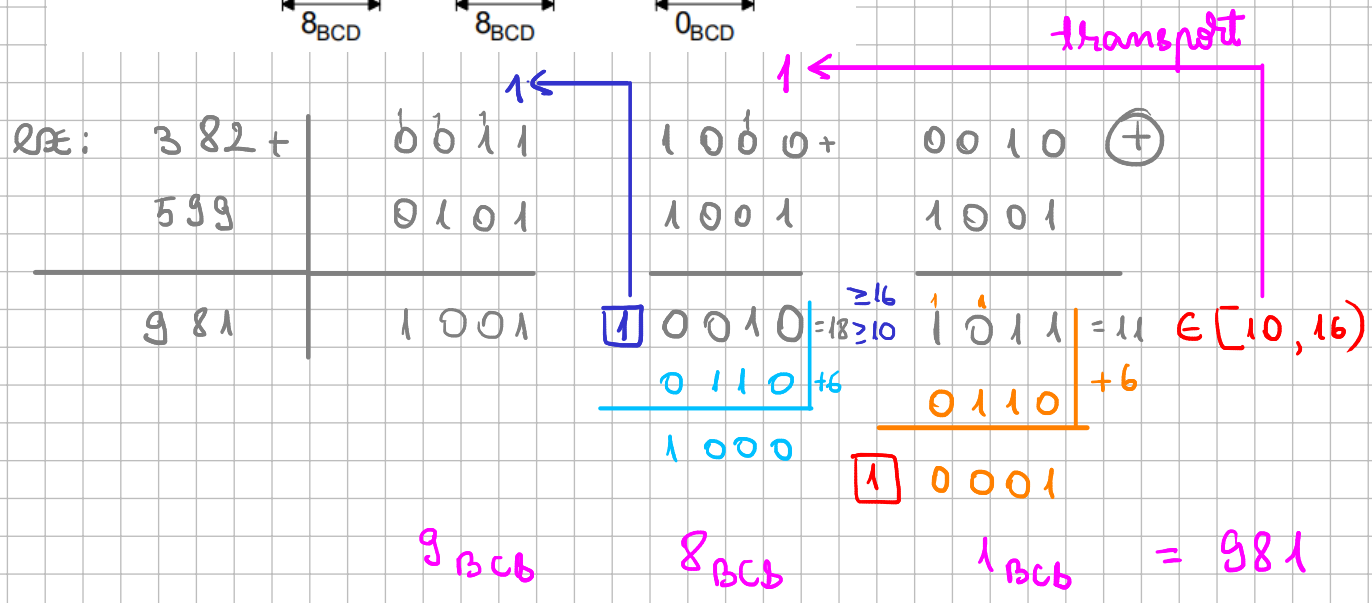
Sumator pentru numere BCD a câte *k*-cifre:



un nr BCD pe k cifre are 4·k biți

Exemplu:





! Poate fi aplicație de examen

ex curs (de rezolvat)

4 9 7 +

4 8 5

9 8 2

(B) Adunare E3

Performanta este un atuu in cadrul sumatoarelor E3

→ Fie x_{iE_3}, y_{iE_3}, z_{iE_3} cifre E_3, unde

$$z_{iE_3} = x_{iE_3} + y_{iE_3}$$

x_{iE3}	x_3	x_2	x_1	x_0
y_{iE3}	y_3	y_2	y_1	y_0
z_{iE3}	z_3	z_2	z_1	z_0

$= x_i + 3$

$= y_i + 3$

$= z_i + 3$

cifre BCD

$X_{iE_3} + Y_{iE_3}$ → Z_{iE_3} : cifra sumă

c_{i+1} : transportul către cifra mai semnificativă

≥ 10:

$z_i = x_i + y_i - 10 \Big|_{+6} \Rightarrow z_i + 3 + 3 = x_i + 3 + y_i + 3 - 10$

$\Leftrightarrow z_{iE_3} = x_{iE_3} + y_{iE_3} - 13$

$c_{i+1} = 1$

< 10:

$z_i = x_i + y_i \Big|_{+6} \Rightarrow z_i + 3 + 3 = x_i + 3 + y_i + 3$

$\Leftrightarrow z_{iE_3} = x_{iE_3} + y_{iE_3} - 3$

$(a + b + 13) \bmod 16$

$c_{i+1} = 0$

Pentru ambele cazuri, Z_{iE_3} necesita corectie

→ Corectia care diferentiază cele 2 cazuri poate fi restricția:

$x_i + y_i \geq 10 \Big|_{+6}$

Se poate lua oricare din forme.
O luam pe a doua pentru ca atunci va fi activa carry-in.

$\Rightarrow x_{iE_3} + y_{iE_3} \geq 16 \Rightarrow c_{i+1} = c^+$

Insumarea in E3 este mai avantajoasa, deoarece este simplificata conditia a ceea ce trebuie sa adunam ca si corectie => imbunatateste performanta, rezultatul propriu zis va fi obtinut mai repede, fiind mai putine nivele logice care il genereaza pe c_{i+1}.

Adunand 2 nr pe 4 biti => se obtine un rezultat pe 5 biti

$$x_{i_{E3}} + y_{i_{E3}} = C^{u}\ z_3^{u}\ z_2^{u}\ z_1^{u}\ z_0^{u}$$

Tinand cont de formatul binar pe 5 biti al sumei de mai sus, conditia care diferentiaza cele 2 cazuri de corectie devine:

$$x_{i_{E3}} + y_{i_{E3}} \geq 16 \equiv C^{u} = 1$$

Se poate demonstra faptul că scăderea lui 3 pe 4 biți poate fi realizată prin adunarea lui 13 cu ignorarea transportului de ieșire din rangul cel mai semnificativ (a se vedea discuția privind scăderea valorii 10 pe 4 biți la adunarea BCD). În mod simetric, scăderea lui 13 pe 4 biți poate fi realizată prin adunarea lui 3 cu ignorarea transportului de ieșire din rangul cel mai semnificativ.

$x_i + y_i - 13 = x_i + y_i + 3$

-13 : 1101 c_2

0011 $+3$

$x_i + y_i - 3 = x_i + y_i + 13$

Corecția lui $z_{i_{E3}}$ depinde de următoarea cond. booleană

x_i, y_i - cifre BCD

C^{u}

$(x_i + y_i \geq 10)$ (1)

$z_{i_{E3}} = z_3^{u}\ z_2^{u}\ z_1^{u}\ z_0^{u} +$

0 0 1 1 (3)

sau -13

$C_{i+1} = 1$

$(x_i + y_i < 10)$ (0)

$z_{i_{E3}} = z_3^{u}\ z_2^{u}\ z_1^{u}\ z_0^{u} +$

1 1 0 1 (13)

sau -3

$C_{i+1} = 0$

⟶ Transportul de ieșire : $C_{i+1} = C^{u}$ (prima condiție)

Unificarea corectiei:

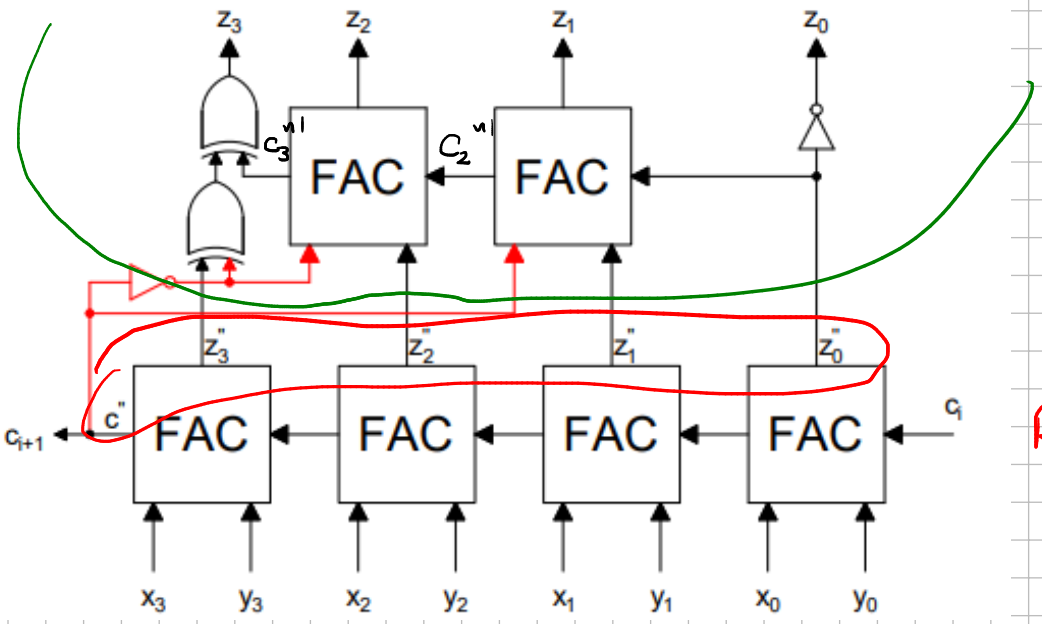
⟶ Stagiul de corecție devine:

$$z_{i_{E3}} = \underline{z_3^{u}}_{C_{i+1}}\ \underline{z_2^{u}}_{C_{i+1}}\ z_1^{u}\ z_0^{u} +$$

C_{i+1} C_{i+1} 1

Implementare arhitectură:

Sumatorul pentru tetrade E3:



RCA-4

Rezultat intermediar RCA-4

$c_{i+1} = c''$

Nivel de corectie

→ La z_0'' mereu adun 1 => z_0'' ▷○— z_0

→ negarea bitului

→ z_0'' este carry in pt z_1

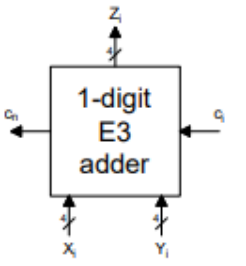
Pt. ultimul rang

$z_3'' + \overline{c_{i+1}}$ (primul x OR)

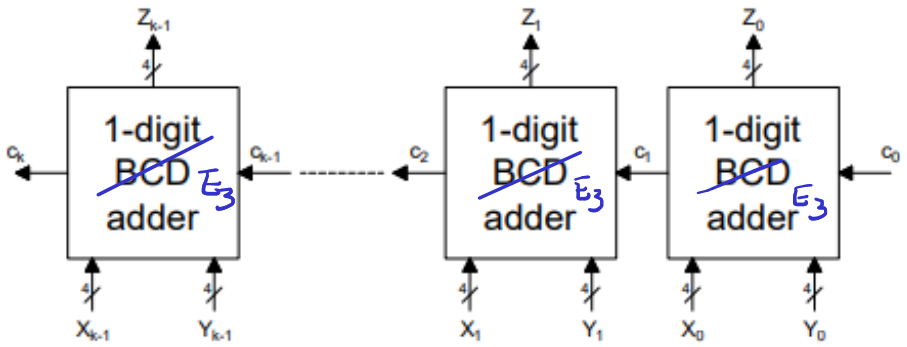
$+ \quad c_3''$ (al doilea x OR)

Nu folosim FAC pentru adunarea celor 3 intrari deoarece nu ne mai intereseaza cat va fi carry-out la prima poarta si nici la a doua. Se ignora transportul rangului de corectie cel mai semnificativ.

Simbol



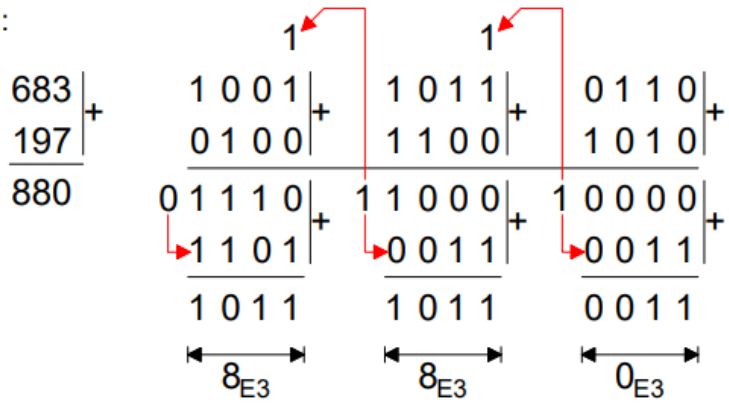
Sumator pentru numere E3 a câte k-cifre

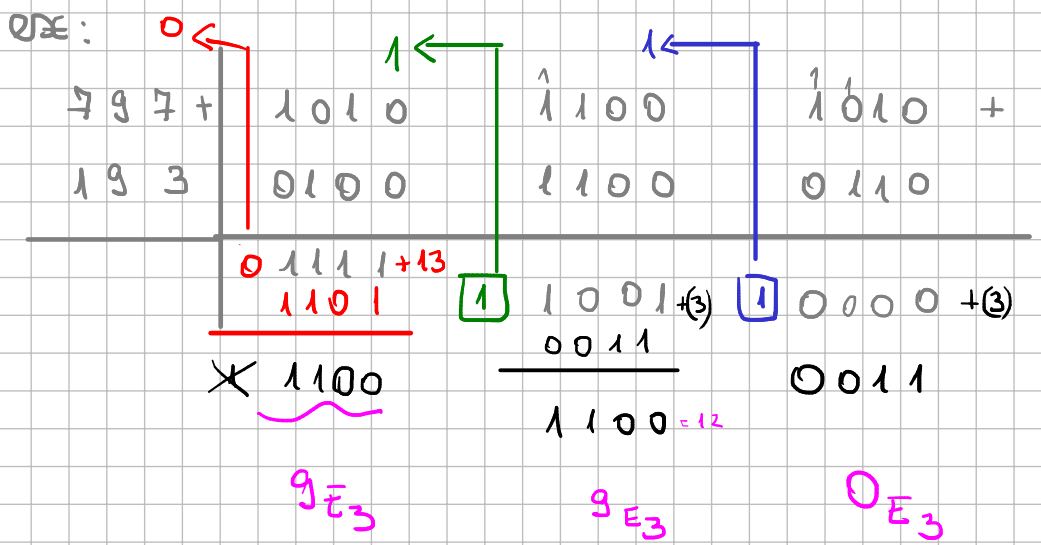


Avantajele adunării în E3:

- transportul de ieșire generat mai rapid
  - ⇒ adunarea va fi efectuată mai rapid
- poate utiliza sumatoare binare
  - este necesar accesul la transporturile generate între tetrade

Exemplu:





ex curs (de făcut)

4 9 7 +

4 8 4