

Arhitectura Calculatoarelor

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

25 Octombrie 2023
1 Noiembrie 2023
8 Noiembrie 2023
15 Noiembrie 2023

*Cap. 2 Analiza funcțională și sinteza
dispozitivelor de adunare și scădere, binară și
zecimală*

2.1 - Sumatoare seriale

Sumator serial:

- ▶ Adună o pereche de biți ai celor 2 operanzi, în fiecare ciclu de ceas.
- ▶ Avantaje:
 - ▶ Suprafața \Downarrow ; consum de energie \Downarrow ; frecvența de operare \Uparrow ;
- ▶ Dezavantaj:
 - ▶ Latența rezultatului final \Uparrow ;

Tipuri de sumatoare seriale :

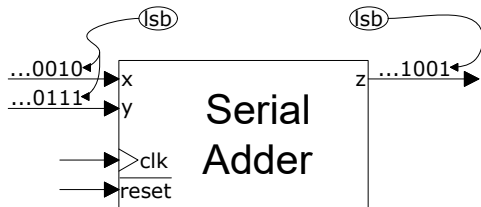
- LSDF (Least significant digit first)
- MSDF (Most significant digit first)

2.1 - Sumatoare seriale (contin.)

LSDF:

$$\begin{array}{rcccccc} X : & x_{n-1} & x_{n-2} & \cdots & x_1 & x_0 \\ Y : & y_{n-1} & y_{n-2} & \cdots & y_1 & y_0 \\ \hline Z : & z_{n-1} & z_{n-2} & \cdots & z_1 & z_0 \end{array} +$$

Simbolul sumatorului serial:



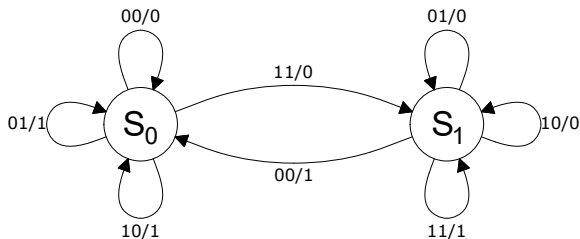
Propagarea transportului: utilizează starea internă a adunătorului

⇒ 2 stări interne $\begin{cases} S_0 : \text{fără propagare de carry din rangul anterior} \\ S_1 : \text{cu propagare de carry din rangul anterior} \end{cases}$

2.1 - Sumatoare seriale (contin.)

Etapele sintezei unui sumator serial

(A) Diagrama de tranziție:



(B) Tabelul de stări:

Stare \ Cfg. intrare	(x,y)			
	00	01	11	10
S_0	S_0 0	S_0 1	S_1 0	S_0 1
S_1	S_0 1	S_1 0	S_1 1	S_1 0

2.1 - Sumatoare seriale (contin.)

Ⓒ Codificarea stărilor:

- ▶ numărul minim de variabile de stare care pot codifica stările
 - ▶ pentru s stări, numărul minim de variabile de stare este $\lceil \log_2 s \rceil$
 - ▶ \Rightarrow pentru sumatorul serial (având $s = 2$), este necesară doar o variabilă de stare ($\lceil \log_2 s \rceil = \lceil \log_2 2 \rceil = 1$)

$w \begin{cases} \rightarrow 0 : \text{codifică } S_0 \\ \rightarrow 1 : \text{codifică } S_1 \end{cases}$

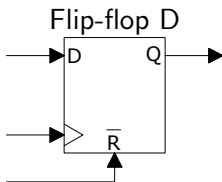
Ⓓ Tabel de tranziție:

Cfg. intrare Var. stare, w	(x,y)			
	00	01	11	10
0	0 / 0	0 / 1	1 / 0	0 / 1
1	0 / 1	1 / 0	1 / 1	1 / 0

2.1 - Sumatoare seriale (contin.)

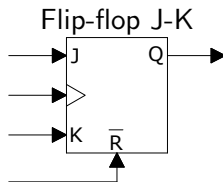
(E) Tabele de excitație:

- dependent de tipul elementelor de stocare utilizate



$$Q(t+1)=D$$

Intrări			Ieșiri	
w	x	y	D	z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$$Q(t+1)=J \cdot \overline{Q(t)} + \overline{K} \cdot Q(t)$$

Intrări			Ieșiri		
w	x	y	J	K	z
0	0	0	0	*	0
0	0	1	0	*	1
0	1	0	0	*	1
0	1	1	1	*	0
1	0	0	*	1	1
1	0	1	*	0	0
1	1	0	*	0	0
1	1	1	*	0	1

2.1 - Sumatoare seriale (contin.)

(F) Ecuațiile de ieșire și feedback:

Ecuațiile de ieșire:

$$\begin{aligned} Z &= \overline{w} \cdot \overline{x} \cdot y + \overline{w} \cdot x \cdot \overline{y} + w \cdot \overline{x} \cdot \overline{y} + w \cdot x \cdot y \\ &= \overline{w} \cdot (\overline{x} \cdot y + x \cdot \overline{y}) + w \cdot (\overline{x} \cdot \overline{y} + x \cdot y) \\ &= \overline{w} \cdot (x \oplus y) + w \cdot (\overline{x \oplus y}) \end{aligned}$$

$$Z = w \oplus x \oplus y$$

Ecuațiile feedback:

D

	<i>x, y</i>			
	00	01	11	10
<i>w</i>				
0	0	0	1	0
1	0	1	1	1

$D = w \cdot x + w \cdot y + x \cdot y$

J

	<i>x, y</i>			
	00	01	11	10
<i>w</i>				
0	0	0	1	0
1	d	d	d	d

$J = x \cdot y$

K

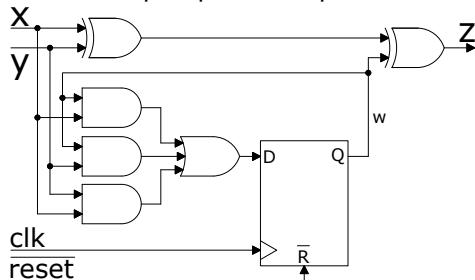
	<i>x, y</i>			
	00	01	11	10
<i>w</i>				
0	d	d	d	d
1	1	0	0	0

$$K = \overline{x} \cdot \overline{y} = \overline{x + y}$$

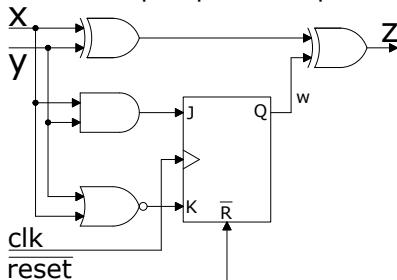
2.1 - Sumatoare seriale (contin.)

Ⓔ Sinteza sumatorului serial:

Folosind flip-flop-uri de tip D



Folosind flip-flop-uri de tip J-K



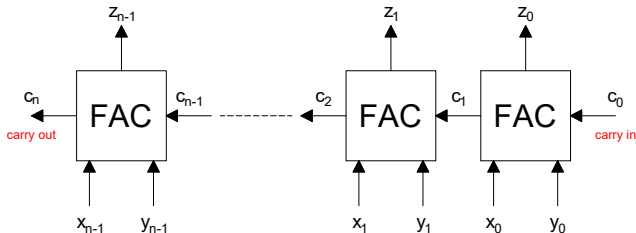
2.2 - Sumatoare și scăzătoare paralele

2.2.1 - Sumatoare paralele bazate pe propagarea serială a transportului

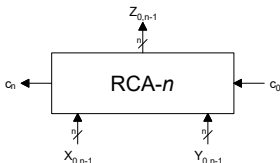
Ripple Carry Adder (RCA): utilizează celule dedicate de însumare pentru fiecare rang binar

► propagarea carry-ului: către poziția mai semnificativ (la stânga)

Arhitectură RCA pe n biți:



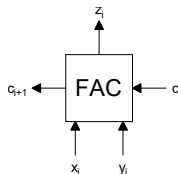
Simbolul unui sumator RCA pe n biți:



2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Full Adder Cell (FAC):

► simbol:



► tabel de adevăr:

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	0	1	0	1
1	1	0	0	1
1	1	1	1	1

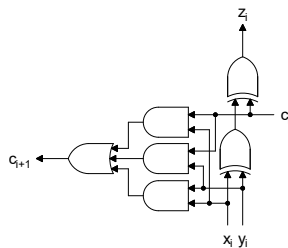
► ecuațiile ieșirilor:

$$\begin{cases} z_i = x_i \oplus y_i \oplus c_i \\ c_{i+1} = x_i \cdot y_i + x_i \cdot c_i + y_i \cdot c_i \end{cases}$$

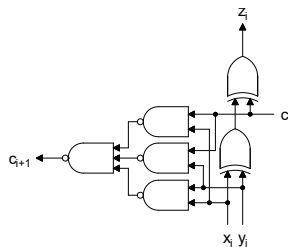
2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Sinteza FAC:

(A) porți de tip
EXOR, AND, OR:



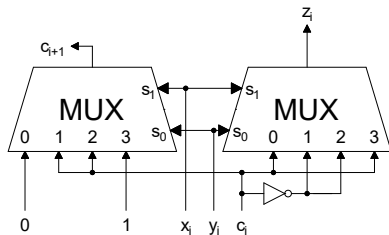
(B) porți de tip
EXOR, NAND:



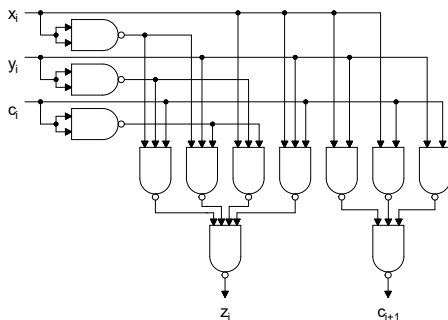
2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Sinteza FAC:

(C) multiplexoare:



(D) porți de tip
NAND:



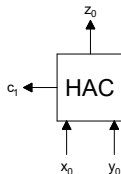
2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Dacă $c_0 = 0 \Rightarrow$ cea mai din dreapta FAC poate fi simplificată:

► ecuațiile ieșirilor:

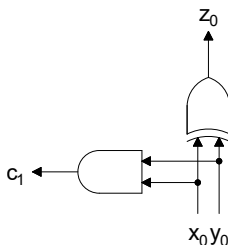
$$\begin{cases} z_0 = x_0 \oplus y_0 \oplus c_0 = x_0 \oplus y_0 \\ c_1 = x_0 \cdot y_0 + x_0 \cdot c_0 + y_0 \cdot c_0 = x_0 \cdot y_0 \end{cases}$$

► simbol:



Sinteza Half Adder Cell (HAC):

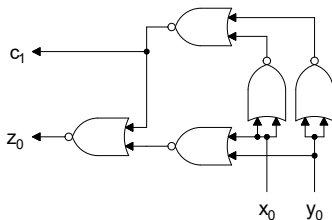
(A') porți de tip
EXOR, AND:



2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Sinteza HAC:

(B') porți de tip NOR



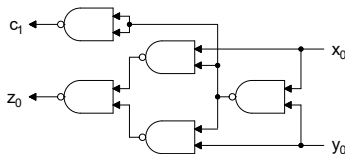
► justificare implementare:

$$\begin{aligned} z_0 &= x_0 \oplus y_0 = x_0 \cdot \overline{y_0} + \overline{x_0} \cdot y_0 = x_0 \cdot (\overline{x_0} + \overline{y_0}) + y_0 \cdot (\overline{x_0} + \overline{y_0}) \\ &= (x_0 + y_0) \cdot (\overline{x_0} + \overline{y_0}) = \overline{\overline{(x_0 + y_0) \cdot (\overline{x_0} + \overline{y_0})}} \\ &= \overline{\overline{x_0 + y_0} + \overline{\overline{x_0} + \overline{y_0}}} \\ c_1 &= x_0 \cdot y_0 = \overline{\overline{x_0 \cdot y_0}} = \overline{\overline{x_0} + \overline{y_0}} \end{aligned}$$

2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Sinteza HAC:

(C') porți de tip
NAND



► justificare implementare:

$$\begin{aligned} z_0 &= x_0 \oplus y_0 = x_0 \cdot \overline{y_0} + \overline{x_0} \cdot y_0 = x_0 \cdot (\overline{x_0} + \overline{y_0}) + y_0 \cdot (\overline{x_0} + \overline{y_0}) \\ &= \overline{\overline{x_0 \cdot (\overline{x_0} + \overline{y_0}) + y_0 \cdot (\overline{x_0} + \overline{y_0})}} = \overline{\overline{x_0 \cdot \overline{x_0 \cdot y_0} \cdot y_0 \cdot \overline{x_0 \cdot y_0}}} \\ c_1 &= x_0 \cdot y_0 = \overline{\overline{x_0 \cdot y_0}} \end{aligned}$$

2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Calea critică:

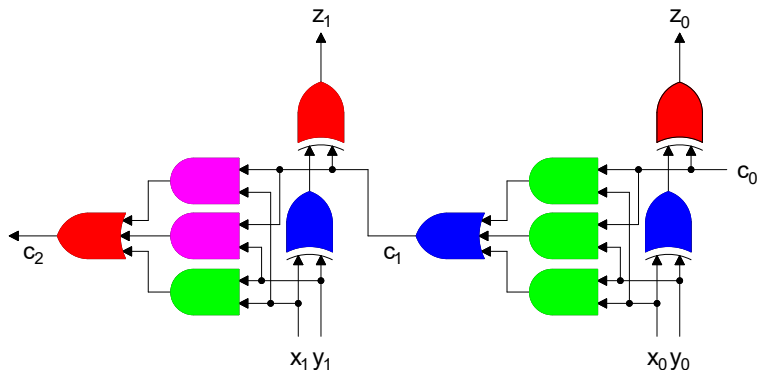
- ▶ calea de propagare din întreg circuitul corespunzătoare întârrierii maxime de propagare a semnalelor
 - ▶ orice element de circuit furnizează semnalele de ieșire cu o întârziere în raport cu semnalele de la intrare

Ipoteze simplificatoare:

- ▶ orice poartă primitivă are latență $1d$ (o unitate de timp)
 - ▶ indiferent de numărul de intrări și timpul porții primitive
- ▶ inversoarele nu introduc întârzieri (au întârziere $0d$)
- ▶ porțile EXOR au latență de $2d$ (Q: de ce ?)
- ▶ toți operanzii sunt disponibili la momentul $0d$

2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Calea critică pentru un RCA pe 2 biți:



Întârizarea unui segment RCA pe n biți:

$$D_{RCA}^{c_{out}} = 2nd$$

$$D_{RCA}^z = 2nd$$

2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Condiții speciale ale adunării:

- ▶ rezultat nul
- ▶ carry out (transport) generat din rangul mai semnificativ
- ▶ rezultat negativ
- ▶ overflow

Overflow aritmetic:

- ▶ rezultatul operației aritmetice depășește capacitatea de stocare

2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

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: ν

Tabel de adevar pentru determinarea condiției de overflow:

Inputs			Outputs	
x_{n-1}	y_{n-1}	c_{n-1}	z_{n-1}	ν
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

2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Forma minimă a condiției de overflow este obținută ca:

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

Condiția de overflow pentru adunarea numerelor cu semn poate fi exprimată într-o forma mai simplă

Identități booleene utile:

- ▶ $I_1: (A \oplus B) \cdot C = A \cdot C \oplus B \cdot C$
- ▶ $I_2: (A + B) = A \oplus B \oplus A \cdot B$
- ▶ $I'_2: A \oplus B = (A + B) \oplus A \cdot B$

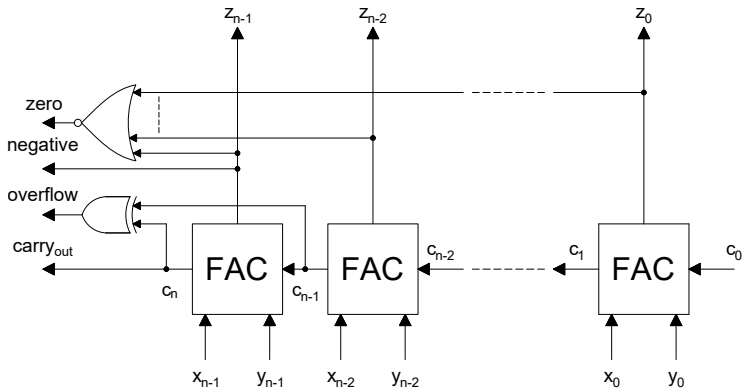
2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Forma simplificată a condiției de overflow este obținută ca:

$$\begin{aligned}\nu &= \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot c_{n-1} + x_{n-1} \cdot y_{n-1} \cdot \overline{c_{n-1}} \\&\stackrel{l_2}{=} \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \cdot \overline{c_{n-1}} \\&= \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \cdot (1 \oplus c_{n-1}) \\&\stackrel{l_1}{=} \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \\&\stackrel{l_1}{=} (\overline{x_{n-1}} \cdot \overline{y_{n-1}} \oplus x_{n-1} \cdot y_{n-1}) \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \\&\stackrel{l'_2}{=} (\overline{x_{n-1}} \cdot \overline{y_{n-1}} + x_{n-1} \cdot y_{n-1}) \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \\&= (x_{n-1} \oplus y_{n-1} \oplus 1) \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \\&\stackrel{l_1}{=} x_{n-1} \cdot c_{n-1} \oplus y_{n-1} \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \oplus c_{n-1} \\&\stackrel{l'_2}{=} (x_{n-1} \cdot c_{n-1} + y_{n-1} \cdot c_{n-1} + x_{n-1} \cdot y_{n-1}) \oplus c_{n-1} \\&\nu = c_n \oplus c_{n-1}\end{aligned}$$

2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

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



2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Adunarea cu o constantă:

► se consideră doar constante impare

► **Întrebare:** de ce?

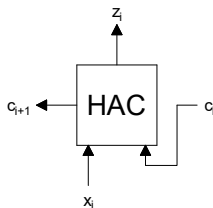
► operandii X și Y - pe n biți

► Y - constant

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

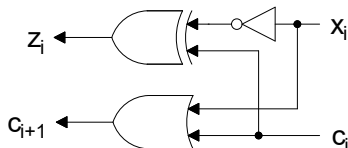
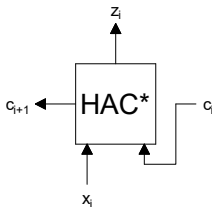
dacă $y_i = 0$:

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



2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

$$\text{dacă } y_i = 1: \quad \left\{ \begin{array}{l} z_i = x_i \oplus 1 \oplus c_i = \bar{x}_i \oplus c_i \\ c_{i+1} = x_i \cdot 1 + x_i \cdot c_i + 1 \cdot c_i = x_i + c_i \end{array} \right\} \text{HAC}^*$$



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 $Y = 110100_2$
- ▶ $Z = X + Y$, cu $c_0 = 0$

2.2.1- Sumatoare paralele bazate pe propagarea serială a transportului (contin.)

Cei mai puțin semnificativi 3 biți ai lui Z sunt determinați astfel:

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

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

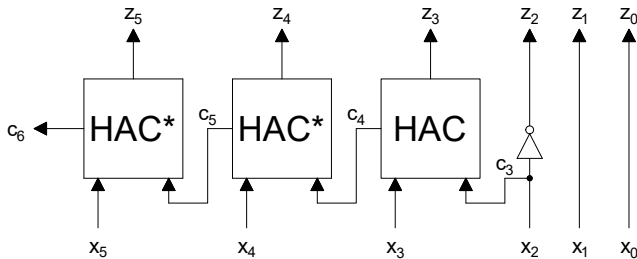
$$z_1 = x_1 \oplus 0 \oplus 0 = x_1$$

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

$$z_2 = x_2 \oplus 1 \oplus 0 = \overline{x_2}$$

$$c_3 = x_2 \cdot 1 + x_2 \cdot 0 + 1 \cdot 0 = x_2$$

Pentru celelalte ranguri ale lui Z se folosesc celule HAC si HAC*:



2.2.2- Sumatoare zecimale bazate pe propagarea serială a transportului

Obiectiv: utilizarea sumatoarelor binare

(A) Adunare BCD:

Fie X_i, Y_i, Z_i cifre BCD, Z_i reprezentând cifra sumă a lui $X_i + Y_i$

► $X_i = x_3x_2x_1x_0, Y_i = y_3y_2y_1y_0, Z_i = z_3z_2z_1z_0$

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

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

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

Pentru cazul $X_i + Y_i \geq 10$, scăderea lui 10 din $X_i + Y_i$ este interpretată ca un pas de corecție.

2.2.2 - Sumatoare zecimale bazate pe propagarea serială a transportului (contin.)

Adunând X_i și Y_i (2 numere pe 4 biți) se obține un rezultat pe 5 biți: $X_i + Y_i = c^*z_3^*z_2^*z_1^*z_0^*$.

Pentru că doar cazul $X_i + Y_i \geq 10$ necesită corecție, se impune analiza acestuia. În acest sens, inegalitatea $X_i + Y_i \geq 10$ devine $c^*z_3^*z_2^*z_1^*z_0^* \geq 10$, inegalitate din urmă care poate fi rescrisă astfel:

$$\begin{cases} 10 \leq c^*z_3^*z_2^*z_1^*z_0^* < 16 & \text{(condiția C1), SAU} \\ c^*z_3^*z_2^*z_1^*z_0^* \geq 16 & \text{(condiția C2)} \end{cases}$$

Condiția C1 implică:

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

Pentru rezolvarea inegalității $z_3^*z_2^*z_1^*z_0^* \geq 10$, în urma minimizării se obține următoarea expresie booleană: $z_3^* \cdot z_2^* + z_3^* \cdot z_1^* = 1$

2.2.2 - Sumatoare zecimale bazate pe propagarea serială a transportului (contin.)

Condiția C1 poate, deci, fi rescrisă în forma următoare:

$$\overline{c^*} \cdot (z_3^* \cdot z_2^* + z_3^* \cdot z_1^*)$$

Inegalitatea asociată condiției C2, $c^* z_3^* z_2^* z_1^* z_0^* \geq 16$, este adevărată dacă:

$$c^* = 1$$

Expresia booleană de identificare a cazului $X_i + Y_i \geq 10$ se obține ca disjuncție logică a condițiilor C1 și C2:

$$\begin{aligned} X_i + Y_i \geq 10 &\equiv c^* + \overline{c^*} \cdot (z_3^* \cdot z_2^* + z_3^* \cdot z_1^*) \\ &= c^* + z_3^* \cdot z_2^* + z_3^* \cdot z_1^* \end{aligned}$$

2.2.2 - Sumatoare zecimale bazate pe propagarea serială a transportului (contin.)

Scăderii valorii 10 din expresia lui $X_i + Y_i$ pentru obținerea cifrei sumă curentă, are ca rezultat un număr binar pe 4 biți. Astfel

$$\begin{aligned}(X_i + Y_i - 10) \bmod 2^4 &= (X_i + Y_i + 16 - 10) \bmod 2^4 \\ &= (X_i + Y_i + 6) \bmod 2^4\end{aligned}$$

Scăderea lui 10, pe 4 biți, poate fi implementată prin adunarea lui 6 ignorând transportul de ieșire din rangul cel mai semnificativ.

$$\begin{array}{rclclcl} \begin{array}{r|l} 15 \\ 10 \\ \hline 5 \end{array} & - & = & \begin{array}{r|l} 1111 \\ 1010 \\ \hline 0101 \end{array} & - & | & \begin{array}{r|l} 15 \\ 6 \\ \hline 5 \end{array} & + & = & \begin{array}{r|l} 1111 \\ 0110 \\ \hline \textcolor{red}{1} 0101 \end{array} & + \end{array}$$

$$\begin{array}{rclclcl} \begin{array}{r|l} 18 \\ 10 \\ \hline 8 \end{array} & - & = & \begin{array}{r|l} 1 & 0010 \\ & 1010 \\ \hline & 1000 \end{array} & - & | & \begin{array}{r|l} 18 \\ 6 \\ \hline 8 \end{array} & + & = & \begin{array}{r|l} 1 & 0010 \\ & 0110 \\ \hline & \textcolor{red}{1} 1000 \end{array} & + \end{array}$$

2.2.2 - Sumatoare zecimale bazate pe propagarea serială a transportului (contin.)

Corecția lui Z_i depinde de următoarea condiție booleană:

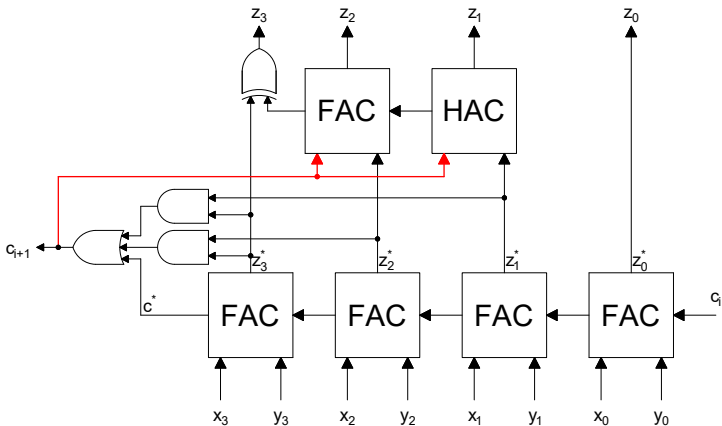
$$c^* + z_3^* \cdot z_2^* + z_3^* \cdot z_1^* \begin{cases} \nearrow \\ \searrow \end{cases} \begin{matrix} 1 \\ 0 \end{matrix} \begin{matrix} (X_i + Y_i \geq 10) \\ (X_i + Y_i < 10) \end{matrix} \Rightarrow \begin{cases} \begin{matrix} Z_i = & z_3^* & z_2^* & z_1^* & z_0^* & + \\ & 0 & 1 & 1 & 0 & (6) \end{matrix} \\ \begin{matrix} c_{i+1} = & 1 \end{matrix} \end{cases} \begin{cases} \begin{matrix} Z_i = & z_3^* & z_2^* & z_1^* & z_0^* & + \\ & 0 & 0 & 0 & 0 & (0) \end{matrix} \\ \begin{matrix} c_{i+1} = & 0 \end{matrix} \end{cases}$$

Transportului de ieșire, c_{i+1} se obține ca: $c_{i+1} = c^* + z_3^* \cdot z_2^* + z_3^* \cdot z_1^*$

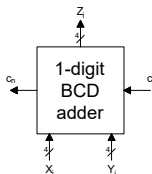
Stagiul de corecție pt. Z_i devine:
$$Z_i = \begin{matrix} z_3^* & z_2^* & z_1^* & z_0^* & + \\ 0 & c_{i+1} & c_{i+1} & 0 \end{matrix}$$

2.2.2 - Sumatoare zecimale bazate pe propagarea serială a transportului (contin.)

Sumatorul pentru tetradе BCD:

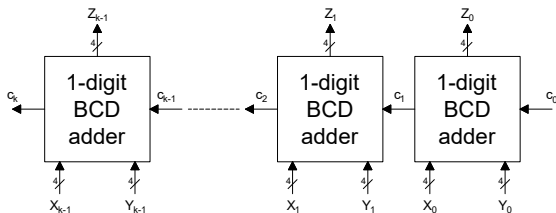


având simbolul:

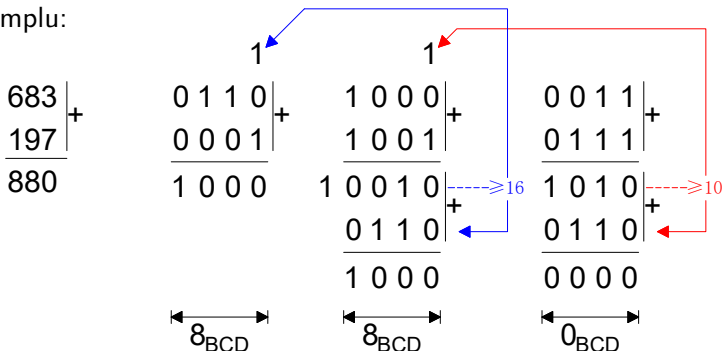


2.2.2 - Sumatoare zecimale bazate pe propagarea serială a transportului (contin.)

Sumator pentru numere BCD a câte k -cifre:



Exemplu:



2.2.2 - Sumatoare zecimale bazate pe propagarea serială a transportului (contin.)

(B) Adunare E3:

Fie X_{iE3} , Y_{iE3} , Z_{iE3} cifre E3, Z_{iE3} fiind cifra sumă a $X_{iE3} + Y_{iE3}$

► $X_{iE3} = x_3x_2x_1x_0$, $Y_{iE3} = y_3y_2y_1y_0$, $Z_{iE3} = z_3z_2z_1z_0$

Fiecărei din cele 3 cifre E3 îi corespunde câte o cifră BCD:

► $X_{iE3} = X_i + 3$, $Y_{iE3} = Y_i + 3$, $Z_{iE3} = Z_i + 3$

► X_i, Y_i, Z_i sunt cifre BCD

$X_{iE3} + Y_{iE3} \begin{cases} \nearrow Z_{iE3} & : \text{cifra sumă} \\ \searrow c_{i+1} & : \text{transportul către cifra mai semnificativă} \end{cases}$

dacă $X_i + Y_i < 10 \begin{cases} \nearrow Z_i = X_i + Y_i|_{+6} \Rightarrow Z_{iE3} = X_{iE3} + Y_{iE3} - 3 \\ \searrow c_{i+1} = 0 \end{cases}$

dacă $X_i + Y_i \geq 10 \begin{cases} \nearrow Z_i = X_i + Y_i - 10|_{+6} \Rightarrow Z_{iE3} = X_{iE3} + Y_{iE3} - 13 \\ \searrow c_{i+1} = 1 \end{cases}$

Pentru ambele cazuri Z_{iE3} necesită câte un pas de corecție.

2.2.2 - Sumatoare zecimale bazate pe propagarea serială a transportului (contin.)

Condiția care diferențiază cele 2 cazuri, poate fi rescrisă astfel:

$$X_i + Y_i \geq 10|_{+6} \Rightarrow X_{i_{E3}} + Y_{i_{E3}} \geq 16$$

Adunând $X_{i_{E3}}$ și $Y_{i_{E3}}$ (2 numere pe 4 biți) se obține un rezultat pe 5 biți: $X_{i_{E3}} + Y_{i_{E3}} = c''z_3''z_2''z_1''z_1'$.

Ținând cont de formatul binar pe 5 biți al sumei $X_{i_{E3}} + Y_{i_{E3}}$ de mai sus, condiția care diferențiază cele 2 cazuri de corecție devine:

$$X_{i_{E3}} + Y_{i_{E3}} \geq 16 \equiv c'' = 1$$

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.

2.2.2 - Sumatoare zecimale bazate pe propagarea serială a transportului (contin.)

Corecția lui Z_{iE3} depinde de următoarea condiție booleană:

$$c'' \begin{cases} \nearrow \\ \searrow \end{cases} \begin{matrix} 1 \\ (X_i + Y_i \geq 10) \\ 0 \\ (X_i + Y_i < 10) \end{matrix} \Rightarrow \begin{cases} \begin{matrix} Z_{iE3} = & z_3'' & z_2'' & z_1'' & z_0'' & + \\ & 0 & 0 & 1 & 1 & (3) \end{matrix} \\ c_{i+1} = 1 \end{cases}$$

$$\Rightarrow \begin{cases} \begin{matrix} Z_{iE3} = & z_3'' & z_2'' & z_1'' & z_0'' & + \\ & 1 & 1 & 0 & 1 & (13) \end{matrix} \\ c_{i+1} = 0 \end{cases}$$

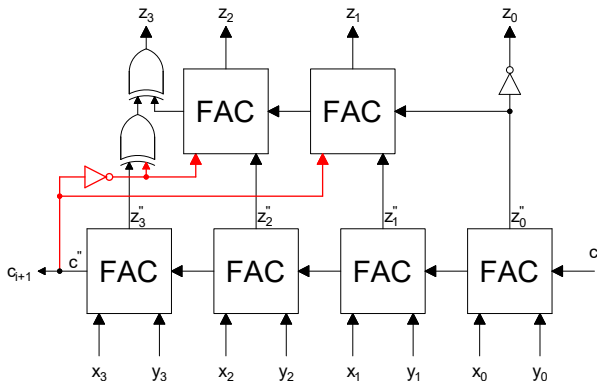
Transportului de ieșire, c_{i+1} se obține ca: $c_{i+1} = c''$

Stagiul de corecție pentru Z_{iE3} devine:

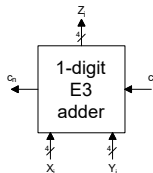
$$Z_{iE3} = \frac{z_3''}{\overline{c_{i+1}}} \frac{z_2''}{\overline{c_{i+1}}} \frac{z_1''}{c_{i+1}} \frac{z_0''}{1} +$$

2.2.2 - Sumatoare zecimale bazate pe propagarea serială a transportului (contin.)

Sumatorul pentru tetradă E3:



având simbolul:



2.2.2 - Sumatoare zecimale bazate pe propagarea serială a transportului (contin.)

Pentru adunarea operandilor în E3 pe k -cifre zecimale se conectează k sumatoare de tetradă E3, înlănțuite prin lanțul de transport (vezi sumatorul pentru operanzi pe k -cifre zecimale, reprezentați în BCD).

Exemplu:

683	+	1001	+	1011	+	0110
197		0100		1100		1010
<hr/>						
880		01110	+	11000	+	10000
		1101	+	0011	+	0011
		<hr/>		<hr/>		<hr/>
		1011		1011		0011
		8 _{E3}		8 _{E3}		0 _{E3}

Avantajele adunării în E3:

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

2.2.3 - Scazatoare bazate pe propagarea seriala a transportului/imprumutului

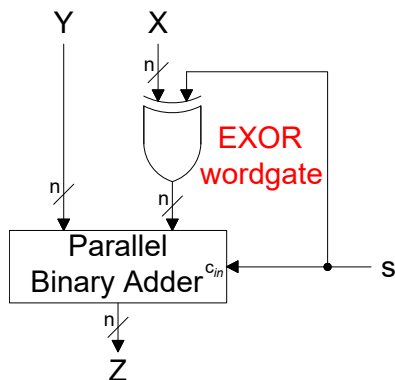
Operația de scădere:

- ▶ X : scăzător, $X = x_{n-1}x_{n-2} \cdots x_1x_0$
- ▶ Y : descăzut, $Y = y_{n-1}y_{n-2} \cdots y_1y_0$
- ▶ diferența celor 2 operanzi: $Z = Y - X$

2.2.3 - Scazatoare bazate pe propagarea seriala ... (contin.)

Modalități de realizare a operației de scădere

(A) Utilizând sumatoare binare: $Y - X = Y + (-X)$



$$s \begin{cases} 1 : Z = Y + \bar{X} + 1 = Y - X \\ 0 : Z = Y + X + 0 = Y + X \end{cases}$$

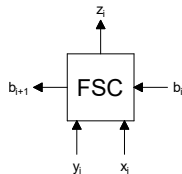
2.2.3 - Scăzătoare bazate pe propagarea seriala ... (contin.)

(B) Scăzătoare dedicate

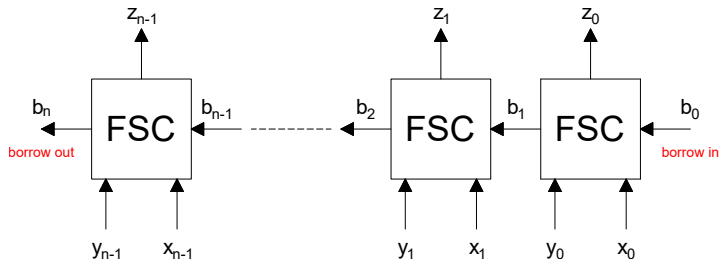
Utilizarea celulelor Full Subtractor Cells (FSCs):

- ▶ transportul este înlocuit de împrumut
- ▶ operație implementată:

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



Arhitectură de scăzător pe n biți:



2.2.3 - Scazatoare bazate pe propagarea seriala ... (contin.)

FSC:

► tabel de adevăr:

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

► ecuatiiile iesirilor:

		x_i, b_i			
		00	01	11	10
z_i	0	0	1	0	1
	1	1	0	1	0

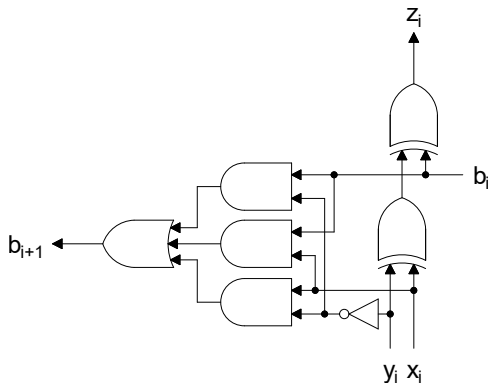
$$z_i = y_i \oplus x_i \oplus b_i$$

b_{i+1}	x_i, b_i			
	y_i	00	01	11
0	0	1	1	1
1	0	0	1	0

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

2.2.3 - Scazatoare bazate pe propagarea seriala ... (contin.)

Sinteza FSC, utilizând porți de tip EXOR, AND, OR, INV:



2.2.3 - Scazatoare bazate pe propagarea seriala ... (contin.)

Ⓒ Scăzătoare BCD

Se consideră:

- ▶ $Y^{(k)}, X^{(k)}$ 2 numere BCD pe k -cifre
 - ▶ $Y^{(k)} = Y_{k-1} Y_{k-2} \cdots Y_0$
 - ▶ $X^{(k)} = X_{k-1} X_{k-2} \cdots X_0$
 - ▶ cu Y_j și X_j - cifre BCD, $\forall j \in [0, k-1]$
- ▶ și $Z^{(k)} = Y^{(k)} - X^{(k)}$, diferența celor 2 numere

Se definește complementul de 9 al unei cifre BCD, X_i , ca fiind:

$$\overline{X_i^*} = 9 - X_i$$

Se definește complementul de 9 al numărului BCD pe k -cifre, $X^{(k)}$:

$$\begin{aligned}\overline{X^{*(k)}} &= \overline{X_{k-1}^*} \quad \overline{X_{k-2}^*} \quad \cdots \quad \overline{X_0^*} \\ &\quad \leftarrow \text{<k> digits} \rightarrow \\ &= 9 \quad 9 \quad \cdots \quad 9 \quad - \\ &\quad X_{k-1} \quad X_{k-2} \quad \cdots \quad X_0 \\ \overline{X^{*(k)}} &= 10^k - 1 - X^{(k)}\end{aligned}$$

2.2.3 - Scazatoare bazate pe propagarea seriala ... (contin.)

Diferența $Z^{(k)}$ poate fi scrisă astfel:

$$\begin{aligned} Z^{(k)} &= (Y^{(k)} - X^{(k)}) \mod 10^k \\ &= (Y^{(k)} + 10^k - 1 - X^{(k)} + 1) \mod 10^k \\ &= (Y^{(k)} + \overline{X}^{*(k)} + 1) \mod 10^k \end{aligned}$$

$$Z^{(k)} = (Y^{(k)} + \overline{X}^{*(k)} + 1)$$

Proiectarea unui modul pentru determinarea complementului de 9 a unei cifre zecimale:

- ▶ cifra BCD de convertit, $X_i = x_3x_2x_1x_0$
- ▶ complementul de 9 a lui X_i : $\overline{X}_i^* = x_3^*x_2^*x_1^*x_0^*$
 - ▶ cu $\overline{X}_i^* = 9 - X_i$

2.2.3 - Scazatoare bazate pe propagarea seriala ... (contin.)

Tabel de adevăr al unității pentru calcularea complementului de 9:

Inputs				Outputs			
x_3	x_2	x_1	x_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

În urma minimizării cei 4 biți ai ieșirii au expresiile următoare:

$$x_3^* = \overline{x_3 + x_2 + x_1}$$

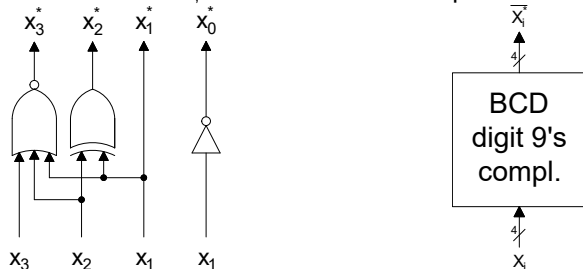
$$x_2^* = x_2 \oplus x_1$$

$$x_1^* = x_1$$

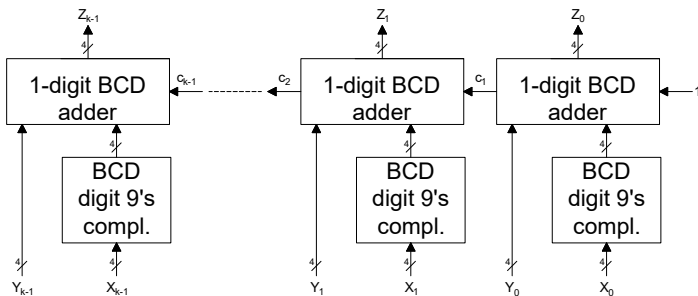
$$x_0^* = \overline{x_0}$$

2.2.3 - Scazatoare bazate pe propagarea seriala ... (contin.)

Arhitectura și simbolul unității de calculare a complementului de 9:



Arhitectura unui scăzător pentru numere BCD pe k -cifre:



2.3 - Calculul paralel al sumei

2.3.1 Sumator Carry Lookahead

Un sumator Carry Lookahead complet (F-CLA), este caracterizat de ecuația:

$$c_{i+1} = x_i \cdot y_i + c_i \cdot (x_i + y_i) \quad \begin{cases} g_i = x_i \cdot y_i & \text{- variabilă generate} \\ p_i = x_i + y_i & \text{- variabilă propagate} \end{cases}$$

Astfel, c_{i+1} poate fi scris ca: $c_{i+1} = g_i + p_i \cdot c_i$. Utilizand definiția recursivă a lui c_{i+1} , acesta devine:

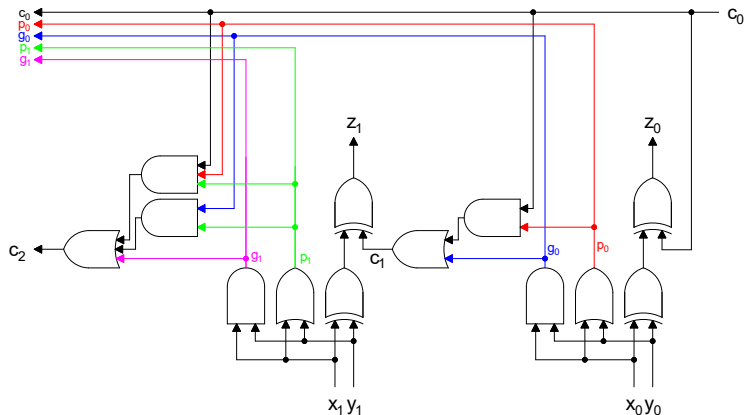
$$\begin{aligned} c_{i+1} &= g_i + p_i \cdot c_i \\ &= g_i + p_i \cdot g_{i-1} + p_i \cdot p_{i-1} \cdot c_{i-1} \\ &= \dots \\ &= g_i + p_i \cdot g_{i-1} + \dots + p_i \cdot p_{i-1} \cdot \dots \cdot p_1 \cdot g_0 + p_i \cdot p_{i-1} \cdot \dots \cdot p_0 \cdot c_0 \end{aligned}$$

Dezavantaje: $\begin{cases} \text{fan-out ridicat: } p_i \text{ este utilizat de } i + 1 \text{ termeni} \\ \text{fan-in ridicat: } c_{i+1} \text{ are } i + 2 \text{ termeni} \end{cases}$

\Rightarrow Sumatoarele F-CLA operează numere de lățime redusă

2.3.1 Sumator Carry Lookahead (contin.)

Sumator F-CLA pe 2 biți:



Întârizarea F-CLA pe n biți:

$$D_{F-CLA}^{C_{out}} = 3d$$

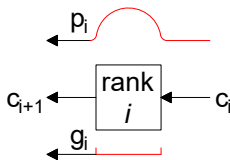
$$D_{F-CLA}^Z = 5d$$

2.3.1 Sumator Carry Lookahead (contin.)

Ecuția $c_{i+1} = g_i + p_i \cdot c_i$, poate fi interpretată prin prisma celor 2 variabile, p_i și g_i astfel:

- ↙ g_i : transportul este generat "în" rangul i
- ↘ p_i : transportul este propagat "peste" rangul i

Cele două condiții de generare a transportului c_{i+1} pot fi simbolizate grafic astfel:



2.3.1 Sumator Carry Lookahead (contin.)

Se consideră transportul, c_4 și modalitatea de exprimare recursivă a acestuia:

$$\begin{aligned}c_4 &= g_3 + p_3 \cdot c_3 \\&= g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot c_2 \\&= \dots \\&= \underbrace{g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0}_{G_{0,3}} + \underbrace{p_3 \cdot p_2 \cdot p_1 \cdot p_0}_{P_{0,3}} \cdot c_0\end{aligned}$$

Transportul c_4 poate fi re-scris astfel:

$$c_4 = G_{0,3} + P_{0,3} \cdot c_0$$

2.3.1 Sumator Carry Lookahead (contin.)

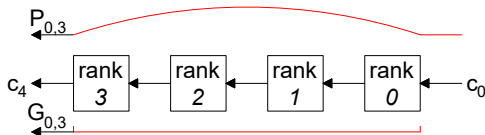
În ecuația:

$$c_4 = G_{0,3} + P_{0,3} \cdot c_0$$

$G_{0,3}$ și $P_{0,3}$ se numesc variabile de generare, respectiv propagare, la nivel de bloc, având semnificațiile următoare:

- ▶ $G_{0,3}$ indică faptul că transportul este generat în blocul de ranguri de la 0 până la 3, inclusiv
- ▶ $P_{0,3}$ indică faptul că transportul este propagat peste blocul de ranguri de la 0 până la 3, inclusiv

Ecuația de mai sus poate fi simbolizată grafic astfel:



2.3.1 Sumator Carry Lookahead (contin.)

Pe de altă parte, expresia extinsă a lui c_4 poate fi grupată ca în ecuațiile de mai jos:

$$\begin{aligned} c_4 &= g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0 \\ &= \underbrace{g_3 + p_3 \cdot g_2}_{G_{2,3}} + \underbrace{p_3 \cdot p_2}_{P_{2,3}} \cdot \underbrace{(g_1 + p_1 \cdot g_0)}_{G_{0,1}} + \underbrace{p_3 \cdot p_2}_{P_{2,3}} \cdot \underbrace{p_1 \cdot p_0}_{P_{0,1}} \cdot c_0 \\ &\quad \underbrace{\hspace{10em}}_{G_{0,3}} \hspace{10em} \underbrace{\hspace{10em}}_{P_{0,3}} \end{aligned}$$

Variabile de generare/propagare la nivelul unui bloc de ranguri pot fi exprimate în termenii variabilelor de propagare/generare la nivel de sub-bloc, ca în expresiile de mai jos:

$$\begin{aligned} G_{0,3} &= G_{2,3} + P_{2,3} \cdot G_{0,1} \\ P_{0,3} &= P_{2,3} \cdot P_{0,1} \end{aligned}$$

2.3.1 Sumator Carry Lookahead (contin.)

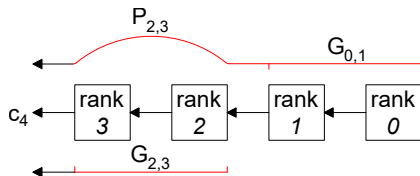
Ecuția:

$$G_{0,3} = G_{2,3} + P_{2,3} \cdot G_{0,1}$$

indică faptul că pentru a fi generat în blocul de ranguri de la 0 până la 3, transportul fie:

- ▶ este generat în blocul de ranguri de la 2 până la 3,
- ▶ fie este generat în blocul de ranguri de la 0 până la 1 și este propagat peste rangurile de la 2 până la 3

Generarea transportului în blocul de ranguri de la 0 până la 3 este simbolizată grafic astfel:



2.3.1 Sumator Carry Lookahead (contin.)

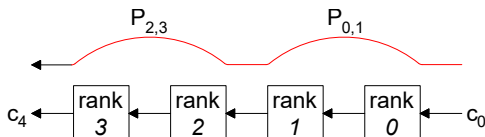
Ecuția:

$$P_{0,3} = P_{2,3} \cdot P_{0,1}$$

indică faptul că pentru a fi propagat peste blocul de ranguri de la 0 până la 3, transportul :

- ▶ trebuie să fie propagat peste rangurile de la 2 până la 3, ȘI
- ▶ trebuie să fie propagat peste ranguri de la 0 până la 1

Propagarea transportului peste blocul de ranguri de la 0 până la 3 este simbolizată grafic astfel:



2.3.1 Sumator Carry Lookahead (contin.)

Se notează:

$$G_{i,i} = g_i = x_i \cdot y_i$$

$$P_{i,i} = p_i = x_i + y_i$$

pentru orice rang i

În general, transportul împreună cu variabilele de generare/propagare la nivel de bloc pot fi exprimate astfel:

$$\begin{aligned} c_{j+1} &= G_{i,j} + P_{i,j} \cdot c_i & \forall i \leq j \\ G_{i,k} &= G_{j+1,k} + P_{j+1,k} \cdot G_{i,j} & \forall i \leq j < k \\ P_{i,k} &= P_{j+1,k} \cdot P_{i,j} & \forall i \leq j < k \end{aligned}$$

2.3.1 Sumator Carry Lookahead (contin.)

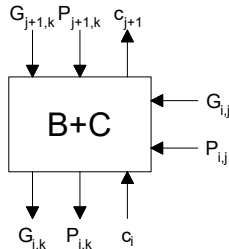
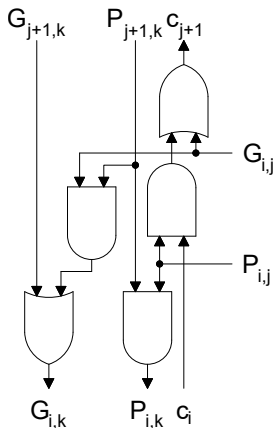
Toate cele 3 ecuații

$$c_{j+1} = G_{i,j} + P_{i,j} \cdot c_i$$

$$G_{i,k} = G_{j+1,k} + P_{j+1,k} \cdot G_{i,j}$$

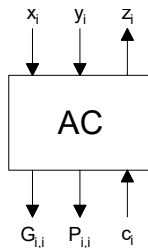
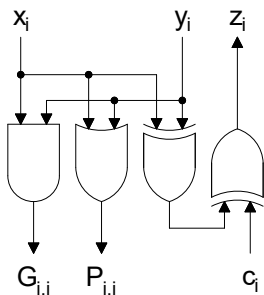
$$P_{i,k} = P_{j+1,k} \cdot P_{i,j}$$

sunt implementate de celulele de tip $B + C$ descrise mai jos:



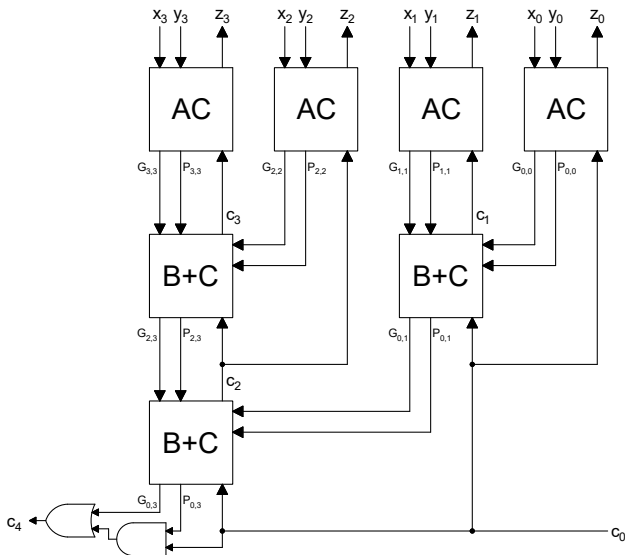
2.3.1 Sumator Carry Lookahead (contin.)

Semnalelor $G_{i,i}$ și $P_{i,i}$ (pe baza cărora se vor construi semnalele de generare/propagare la nivel de bloc) împreună cu biții sumei, z_i , vor fi generați de celulele de tip AC de mai jos:



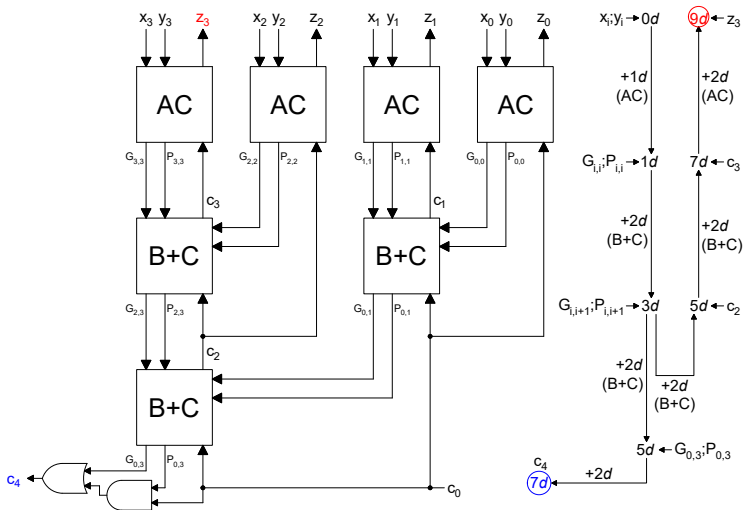
2.3.1 Sumator Carry Lookahead (contin.)

Arhitectura CLA multinivel pe 4 biți:



2.3.1 Sumator Carry Lookahead (contin.)

Determinarea latenței unui CLA multinivel pe 4 biți:



$$D_{ML-CLA-4}^{Cout} = 7d$$

$$D_{ML-CLA-4}^z = 9d$$

2.3.1 Sumator Carry Lookahead (contin.)

Se consideră un sumator CLA multinivel pe n biți. Pentru o astfel de arhitectură, există $\lceil \log_2 n \rceil$ nivele de celule $B + C$.

În general:

$$D_{ML-CLA-n}^{Cout} = (2\lceil \log_2 n \rceil + 3)d$$

$$D_{ML-CLA-n}^z = (4\lceil \log_2 n \rceil + 1)d$$

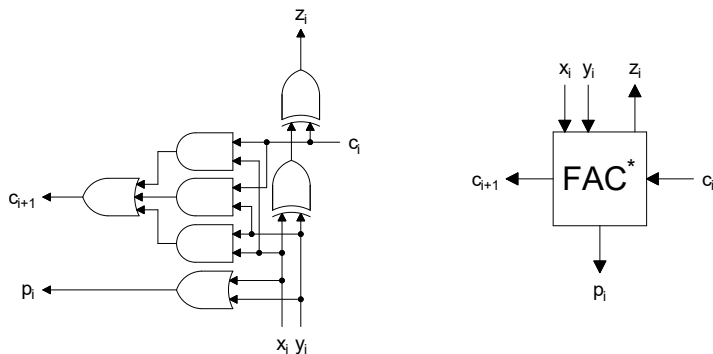
2.3.2 Sumator Carry Skip

Din ecuația de generare a transportului:

$$c_{j+1} = G_{i,j} + P_{i,j} \cdot c_i$$

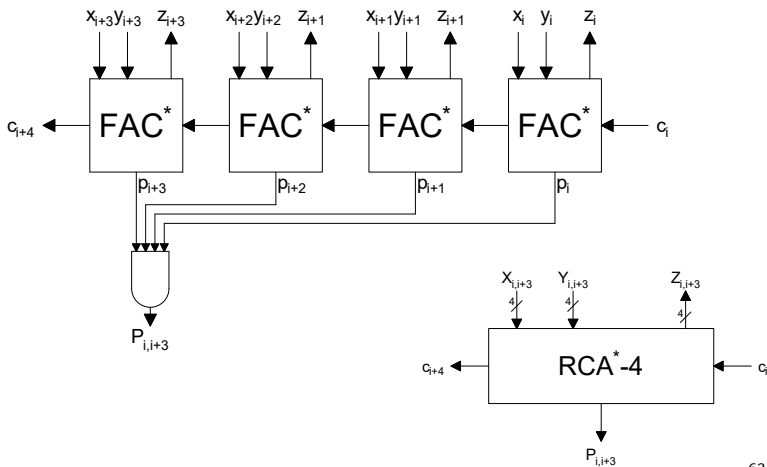
variabila de propagare la nivel de bloc, $P_{i,j}$, se obține mai simplu.

În consecință, celula FAC va fi extinsă cu logică pentru generarea variabilei p_i la nivel de rang binar, ca în figura de mai jos:



2.3.2 Sumator Carry Skip (contin.)

Utilizând celule FAC^* , se poate construi un segment RCA care generează variabila de propagare la nivelul întregului bloc de ranguri. În figura de mai jos este descris un astfel de segment, pe 4 biți, împreună cu simbolul asociat:



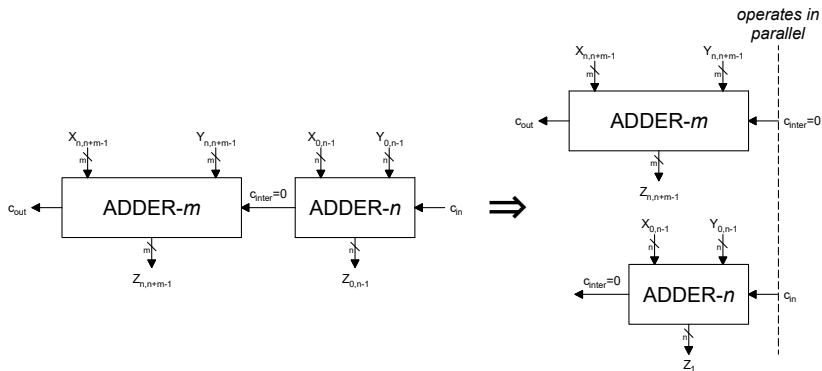
2.3.2 Sumator Carry Skip (contin.)

CMOS pre-discharging:

- modalitate de proiectare a unei arhitecturi prin care un set de noduri sunt aduse la valoarea 0 înainte de începerea calculului

Pentru sumatorul Carry Skip (CSkA):

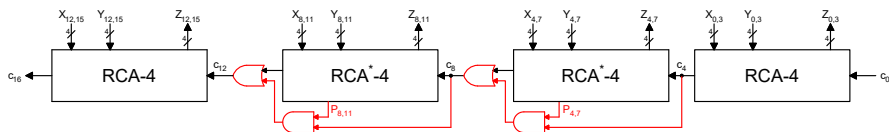
- toate semnalele transport inter-rang sunt prevăzute cu CMOS pre-discharge
 - \Rightarrow dacă valoarea finală, corectă, a unui transport este 0, ea va fi calculată corect de la momentul inițial.



2.3.2 Sumator Carry Skip (contin.)

Arhitectura unui sumator Carry Skip (CSkA) pe 16 biți:

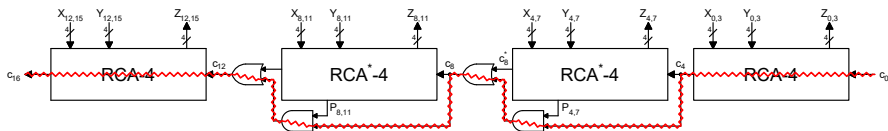
- pentru blocuri de tip RCA^* este utilizată logica de șunt ("skip logic", evidențiată prin culoare) pentru a facilita propagarea transportului peste întregul bloc



În arhitectura de mai sus, cel mai puțin semnificativ și cel mai semnificativ segment nu prezintă logică de șunt.

2.3.2 Sumator Carry Skip (contin.)

Calea critică a arhitecturii CSkA pe 16 biti:



Teorema: de ce folosește calea critică logică de șunt?

Demonstrație: cazul cel mai defavorabil de propagare a semnalelor necesită ca toate semnalele transport să aibă valoarea 1 (un transport de valoare 0 fiind corect de la momentul $0d$, desparte sumatorul în 2 sumatoare mai scurte care vor opera concurent \Rightarrow cale critică mai scurtă)

Semnalul de transport c_8 are următoare expresie:

$$c_8 = c_8^* + P_{4,7} \cdot c_4$$

2.3.2 Sumator Carry Skip (contin.)

În expresia lui c_8 ($c_8^* + P_{4,7} \cdot c_4$) semnalul c_4 va avea valoarea 1 (pentru a nu despărți sumatorul în 2 părți care vor opera concurrent).

Logica de șunt oferă propagarea mai rapidă a transportului:

- ▶ prin șunt, transportul are întârzierea $2d$: poarta ȘI urmată de poarta SAU finală
- ▶ prin RCA, transportul are întârzierea $9d$: $8d$ pentru RCA urmat de poarta SAU finală

Pentru a evita logica de șunt, se impune ca $P_{4,7} = 0$. Însă:

$$P_{4,7} = 0 \Rightarrow p_4 \cdot p_5 \cdot p_6 \cdot p_7 = 0$$

$$\Rightarrow \exists \text{ cel puțin un } p_i, \text{ cu } i \in [4, 7], \text{ pentru care } p_i = 0$$

$$\Rightarrow \text{pentru acel indice } i \text{ se poate scrie : } x_i + y_i = 0 \Rightarrow x_i = y_i = 0$$

$$\Rightarrow c_{i+1} = 0$$

2.3.2 Sumator Carry Skip (contin.)

Un transport intern al segmentului RCA de valoare 0 reduce calea critică totală (2 sumatoare mai scurte operând concurent). Ca urmare, cazul cel mai defavorabil de propagare a transportului revendică $P_{4,7} = 1$.

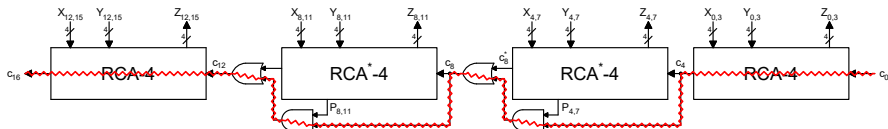
Urmarea acestei observații, evidentă din expresia lui c_8 :

$$c_8 = c_8^* + P_{4,7} \cdot c_4$$

este că în cazul cel mai defavorabil de propagare, transportul va urma logica de șunt. ■

2.3.2 Sumator Carry Skip (contin.)

Calcularea întârzierii maxime pentru sumatorul CSkA pe 16 biți:



$$D_{CSkA-16}^Z = \underbrace{8d}_{c_4} + \underbrace{2d}_{c_8} + \underbrace{2d}_{c_{12}} + \underbrace{8d}_{z_{15}} = 20d$$

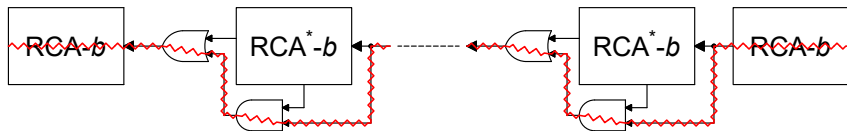
$$D_{CSkA-16}^{c_{out}} = \underbrace{8d}_{c_4} + \underbrace{2d}_{c_8} + \underbrace{2d}_{c_{12}} + \underbrace{8d}_{c_{16}} = 20d$$

Determinarea lungimii optime a segmentelor RCA

Se consideră o structura CSkA având:

- ▶ segmente RCA de lungime b biți
- ▶ segmentul cel mai semnificativ și cel mai puțin semnificativ nu au logică de șunt
- ▶ operanzii au n biți, cu $n = k * b, k \in \mathbb{N}$

Figura următoare descrie sumatorul CSkA pe n biți și modul de propagare a semnalelor de-a lungul căii sale critice (sunt omise intrările și ieșirile segmentelor pentru concizie):



2.3.2 Sumator Carry Skip (contin.)

Întârzierea de generare a sumei/transportului de ieșire pentru arhitectura CSkA pe n biți:

$$D_{CSkA-n}^{z/c_{cout}} = 2bd + 2\left(\frac{n}{b} - 2\right)d + 2bd = \left(\frac{2n}{b} + 4b - 4\right)d$$

Lungimea optimă, b_{opt} , reprezintă un punct de extrem local al funcției $D_{CSkA-n}^{z/c_{cout}}$, deci:

$$\begin{aligned}\frac{\partial D_{CSkA-n}^{z/c_{cout}}}{\partial b_{opt}} = 0 &\Rightarrow -\frac{2n}{b^2} + 4 = 0 \\ \Rightarrow b_{opt} = \frac{\sqrt{2n}}{2}, &\text{ cu latența optimă : } D_{CSkA-n_{opt}}^{z/c_{cout}} = 4(\sqrt{2n} - 1)d\end{aligned}$$

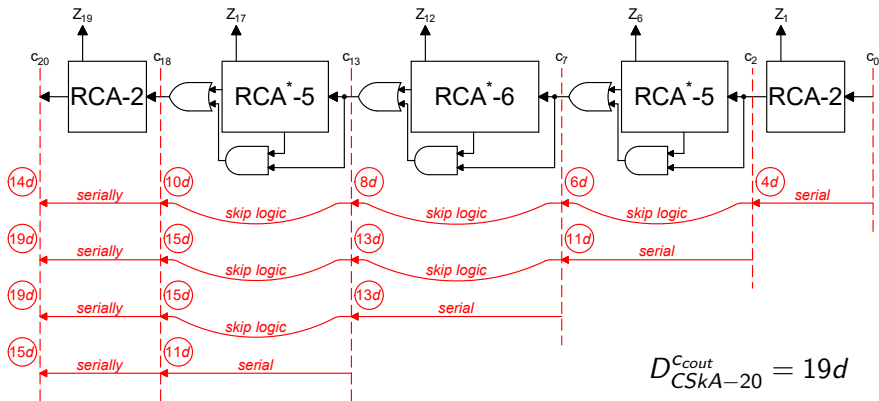
Exemplu: se consideră $n = 32$

- ▶ $b_{opt} = 4$
 - ▶ $D_{CSkA-32_{opt}}^{z/c_{cout}} = 28d$
 - ▶ $D_{RCA-32}^{z/c_{cout}} = 64d$ (ca termen de comparație)

2.3.2 Sumator Carry Skip (contin.)

Segmente RCA de lungimi variabile

Se consideră operanzi pe 20 biti și structura CSkA următoare:



Pentru transportul de ieșire, cazul defavorabil începe propagarea transportului din segmentul $RCA^* - 5$ mai puțin semnificativ (la fel de defavorabil este și cazul începerii din segmentul RCA de 6 biți).

2.3.2 Sumator Carry Skip (contin.)

Pentru sumă, cazul defavorabil se determină evaluând latența maximă a celui mai semnificativ bit sumă pentru fiecare segment RCA/RCA^* ($z_{19}, z_{17}, z_{12}, z_6, z_1$).

Întârzierea de generare a unui bit sumă este maximă dacă transportul de intrare al segmentul RCA respectiv are întârzierea maximă. În consecință:

$$D_{max}^{z_6} = D_{max}^{c_2} + \underbrace{2 * 5d}_{D_{RCA^*-5}^z} = 4d + 10d = 14d$$

$$D_{max}^{z_{12}} = D_{max}^{c_7} + 2 * 6d = 11d + 12d = 23d$$

$$D_{max}^{z_{17}} = D_{max}^{c_{13}} + 2 * 5d = 13d + 10d = 23d$$

$$D_{max}^{z_{19}} = D_{max}^{c_{18}} + 2 * 2d = 15d + 4d = 19d$$

În concluzie:

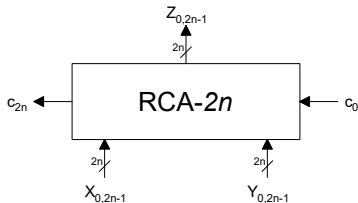
$$D_{CSkA-20}^z = 23d$$

2.3.3 Carry Select Adder

Bazat pe **principiul sumei condiționate prin transport**:

- ▶ c_i are doar 2 posibile valori: $c_i \begin{matrix} \nearrow 0 \\ \searrow 1 \end{matrix}$
- ▶ se calculează z_i și c_{i+1} , în 2 cazuri: $\begin{matrix} \nearrow (z_i^0, c_{i+1}^0), \text{ dacă } c_i = 0 \\ \searrow (z_i^1, c_{i+1}^1), \text{ dacă } c_i = 1 \end{matrix}$
- ▶ se va selecta una din cele 2 perechi (z_i, c_{i+1}) de mai sus ca fiind varianta corectă, **după** obținerea valorii corecte a lui c_i

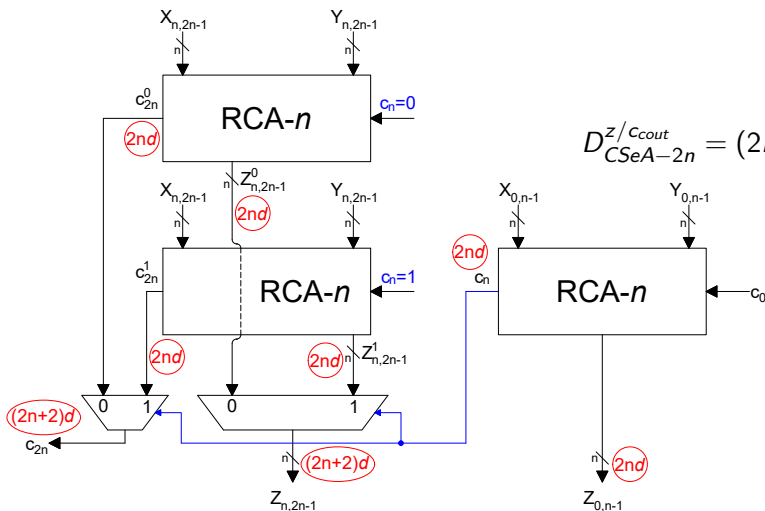
Construcția unui sumator Carry Select (CSeA), pornește de la arhitectura unui RCA pe $2n$ biți:



2.3.3 Carry Select Adder (contin.)

Construcția CSeA:

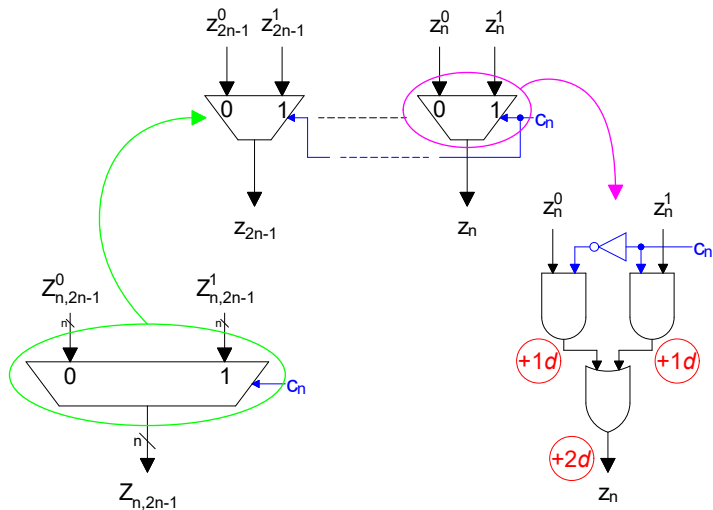
- ▶ se împarte sumatorul RCA- $2n$ în două jumătăți egale
- ▶ se duplică jumătatea mai semnificativă
 - ▶ o copie are $c_n = 0$, celalată are $c_n = 1$



$$D_{CSeA-2n}^{Z/c_{cout}} = (2n + 2)d$$

2.3.3 Carry Select Adder (contin.)

Detalii întârziere multiplexor:



2.3.3 Carry Select Adder (contin.)

Optimizarea ariei CSeA:

- ▶ se pornește de la tabelul de adevăr al semnalului c_{2n}
- ▶ **Niciodată** c_{2n}^0 nu poate fi mai mare decât c_{2n}^1 !
- ▶ **Întrebare:** de ce?
- ▶ \Rightarrow tabelul de adevăr folosește *don't care* pentru cazurile imposibile

Inputs			Output
c_n	c_{2n}^0	c_{2n}^1	c_{2n}
0	0	0	0
0	0	1	0
0	1	0	<i>d</i>
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	<i>d</i>
1	1	1	1

c_{2n}

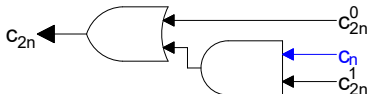
c_{2n}^0, c_{2n}^1

c_n

	00	01	11	10
0	0	0	1	<i>d</i>
1	0	1	1	<i>d</i>

$c_{2n} = c_{2n}^0 + c_{2n}^1 \cdot c_n$

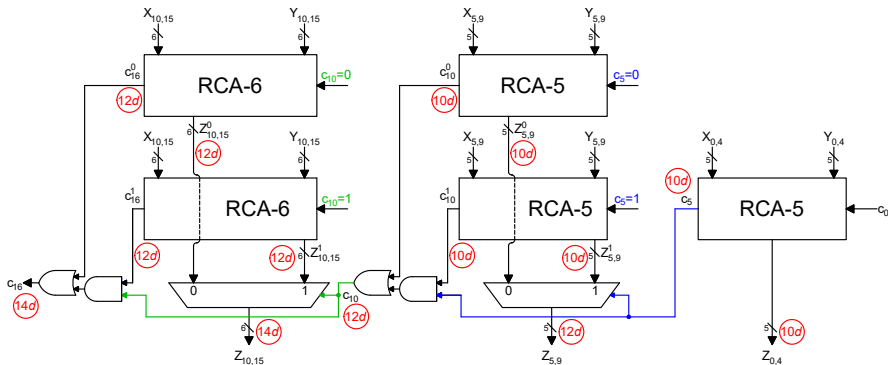
În consecință, transportul de ieșire c_{2n} va fi generat astfel:



2.3.3 Carry Select Adder (contin.)

Segmente RCA de lungimi variabile

Se consideră operanzi pe 16 biți și structura CSeA următoare:



$$D_{CSeA-16}^{Z/c_{out}} = 14d$$

Notă: Se pot construi structuri CSeA multi-nivel.

2.3.4 Conditional Sum Adder

Bazat pe același **principiu al sumei condiționate prin transport**:

- ▶ este o generalizare a CSeA
 - ▶ arhitectură multi-nivel
- ▶ structurat în mai multe etaje
- ▶ primul etaj calculează $\begin{cases} (z_i^0, c_{i+1}^0), & \text{dacă } c_i = 0 \\ (z_i^1, c_{i+1}^1), & \text{dacă } c_i = 1 \end{cases}$
- ▶ etajele următoare utilizează biții transport intermediari pentru a selecta ranguri ale sumei

Se consideră operanzii pe 4 biți: X și Y , și $c_0 = 0$:

$$\begin{aligned} c_{i+1} &= x_i \cdot y_i + x_i \cdot c_i + y_i \cdot c_i = x_i \cdot y_i \cdot (c_i + \overline{c_i}) + x_i \cdot c_i + y_i \cdot c_i \\ &= x_i \cdot y_i \cdot \overline{c_i} + x_i \cdot y_i \cdot c_i + x_i \cdot c_i + y_i \cdot c_i \\ &= x_i \cdot y_i \cdot \overline{c_i} + x_i \cdot c_i \cdot (y_i + 1) + y_i \cdot c_i \\ &= x_i \cdot y_i \cdot \overline{c_i} + x_i \cdot c_i + x_i \cdot c_i + y_i \cdot c_i \\ &= \underbrace{x_i \cdot y_i \cdot \overline{c_i}}_{g_i} + \underbrace{(x_i + y_i) \cdot c_i}_{p_i} \end{aligned}$$

2.3.4 Conditional Sum Adder (contin.)

Transportul c_{i+1} poate fi scris:

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

Forma complementată a lui c_{i+1} se obține astfel:

$$\begin{aligned}\overline{c_{i+1}} &= \overline{g_i \cdot \overline{c_i} + p_i \cdot c_i} = \overline{g_i \cdot \overline{c_i}} \cdot \overline{p_i \cdot c_i} \\ &= (\overline{g_i} + c_i) \cdot (\overline{p_i} + \overline{c_i}) = \underbrace{\overline{g_i} \cdot \overline{p_i}}_{\text{absorbit}} + \overline{g_i} \cdot \overline{c_i} + p_i \cdot c_i + \cancel{c_i \cdot \overline{c_i}}\end{aligned}$$

asa încât pentru $\overline{c_{i+1}}$ se obține:

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

2.3.4 Conditional Sum Adder (contin.)

Sum bits

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

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

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

$$z_3 = x_3 \oplus y_3 \oplus c_3 = \overline{x_3 \oplus y_3} \cdot c_3 + (x_3 \oplus y_3) \cdot \overline{c_3}$$

$$\begin{aligned} &= \overline{x_3 \oplus y_3} \cdot (g_2 \cdot \overline{c_2} + p_2 \cdot c_2) + \\ &\quad (x_3 \oplus y_3) \cdot (\overline{g_2} \cdot \overline{c_2} + \overline{p_2} \cdot c_2) \\ &= (\overline{x_3 \oplus y_3} \cdot g_2 + (x_3 \oplus y_3) \cdot \overline{g_2}) \cdot \overline{c_2} + \\ &\quad (\overline{x_3 \oplus y_3} \cdot p_2 + (x_3 \oplus y_3) \cdot \overline{p_2}) \cdot c_2 \end{aligned}$$

Carry bits

$$c_1 = g_0 \cdot \overline{c_0} + p_0 \cdot c_0 = g_0$$

$$c_2 = g_1 \cdot \overline{c_1} + p_1 \cdot c_1$$

$$c_3 = g_2 \cdot \overline{c_2} + p_2 \cdot c_2$$

$$\overline{c_3} = \overline{g_2} \cdot \overline{c_2} + \overline{p_2} \cdot c_2$$

$$c_4 = g_3 \cdot \overline{c_3} + p_3 \cdot c_3$$

$$= g_3 \cdot (\overline{g_2} \cdot \overline{c_2} + \overline{p_2} \cdot c_2) +$$

$$p_3 \cdot (g_2 \cdot \overline{c_2} + p_2 \cdot c_2)$$

$$= (g_3 \cdot \overline{g_2} + p_3 \cdot g_2) \cdot \overline{c_2} +$$

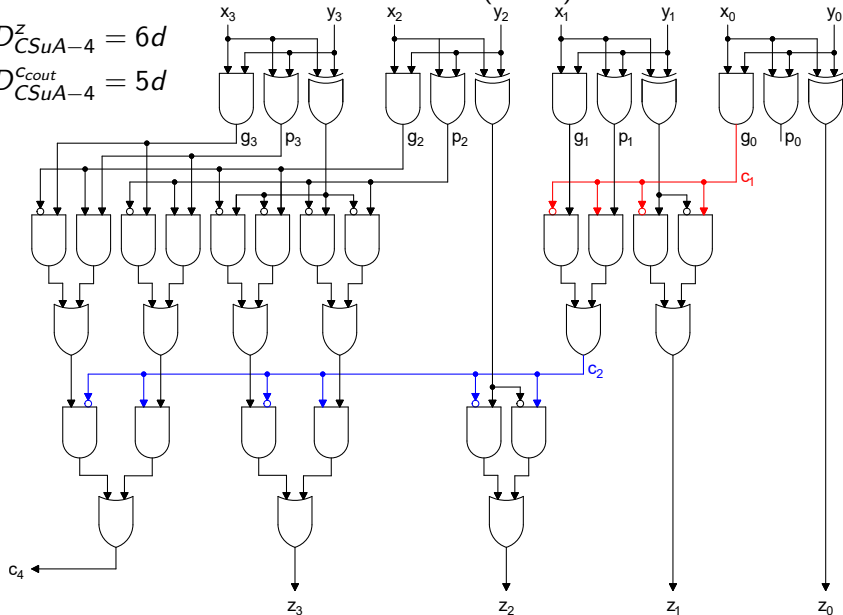
$$(g_3 \cdot \overline{p_2} + p_3 \cdot p_2) \cdot c_2$$

2.3.4 Conditional Sum Adder (contin.)

Arhitectura Conditional Sum Adder (CSuA):

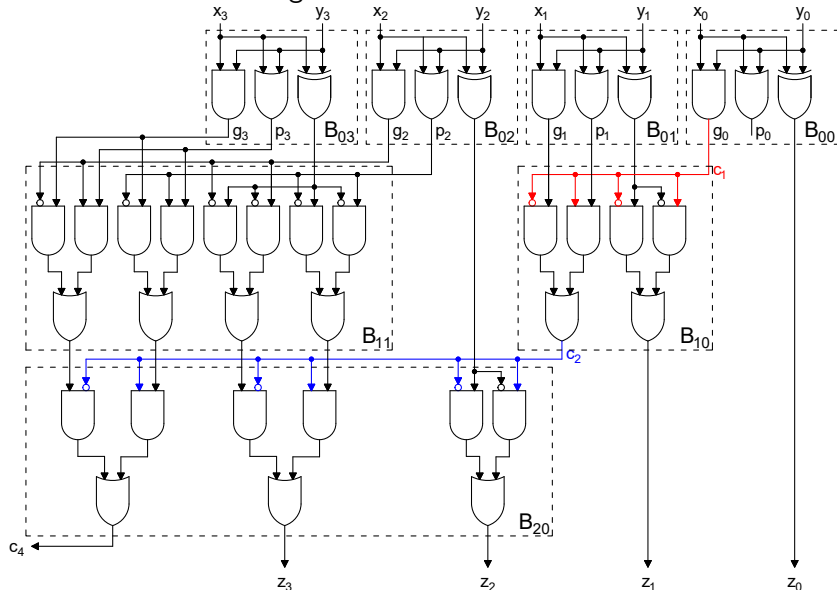
$$D_{CSuA-4}^z = 6d$$

$$D_{CSuA-4}^{c_{out}} = 5d$$



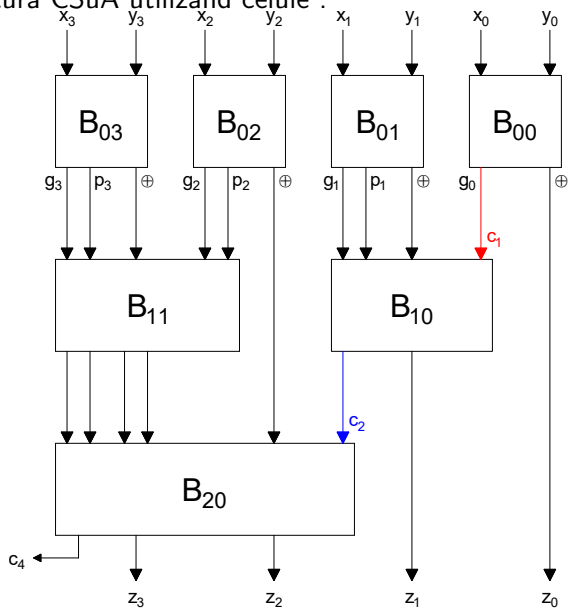
2.3.4 Conditional Sum Adder (contin.)

Arhitectura CSuA organizată în celule:



2.3.4 Conditional Sum Adder (contin.)

Arhitectura CSuA utilizând celule :



2.3.4 Conditional Sum Adder (contin.)

Pentru evaluarea latenței CSuA în cazul general, se consideră operanzi pe n biți.

Exceptând primul nivel, există $\lceil \log_2 n \rceil$ alte nivele în arhitectura CSuA. Aceste nivele corespund celulelor B_{ij} , cu $j \geq 1$ și sunt similare nivelelor $B + C$ din structura CLA multi-nivel.

Fiecare nivel B_{ij} , cu $j \geq 1$ întârzie semnalele cu $2d$ (2 porți AND urmate de o poartă OR).

$$\begin{aligned}
 D_{CSuA-n}^z &= \underbrace{2d}_{\substack{\text{poartă EXOR} \\ \text{din } B_{0j}}} + \underbrace{2}_{\substack{\text{întârzierea} \\ B_{ij} \text{ cu } i \geq 1}} * \underbrace{\lceil \log_2 n \rceil}_{\substack{\text{număr nivele} \\ \text{exceptând primul}}} * d \\
 &= 2(\lceil \log_2 n \rceil + 1)d \\
 D_{CSuA-n}^{Cout} &= \underbrace{1d}_{g_i, p_i} + \underbrace{2}_{\substack{\text{întârzierea} \\ B_{ij} \text{ cu } i \geq 1}} * \underbrace{\lceil \log_2 n \rceil}_{\substack{\text{număr nivele} \\ \text{exceptând primul}}} * d \\
 &= (2\lceil \log_2 n \rceil + 1)d
 \end{aligned}$$

2.3.4 Conditional Sum Adder (contin.)

Scurtă comparație a latențelor sumatoarelor pe $n = 64$ biți:

CSuA	CSeA	ML-CLA	RCA
$D^Z = 14d$	$D^Z = 66d$	$D^Z = 25d$	$D^Z = 128d$
$D^{C_{out}} = 13d$	$D^{C_{out}} = 66d$	$D^{C_{out}} = 15d$	$D^{C_{out}} = 128d$


Exemplu: Însuțați operanzii pe 8 biți X și Y după metoda CSuA:

$$X = 10001101 \quad (= 128 + 13 = 141)$$

$$Y = 01110101 \quad (= 64 + 32 + 16 + 5 = 117)$$

2.3.4 Conditional Sum Adder (contin.)

Primul nivel: calcularea perechilor (z_i^0, c_{i+1}^0) și (z_i^1, c_{i+1}^1) pentru fiecare rang

Rank Operand		7		6		5		4		3		2		1		0		
		X		1		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	
		C _{in} =0		0	1	0	1	0	1	0	1	0	1	1	0	0	0	1
i=0		C _{in} =1		1	0	1	0	1	0	1	0	1	1	1	0	1		
i=1	C _{in} =0																	
	C _{in} =1																	
i=2	C _{in} =0																	
	C _{in} =1																	
i=3	C _{in} =0																	
	C _{in} =1																	

2.3.4 Conditional Sum Adder (contin.)

Nivelul al doilea: determinarea variantelor de sume corecte pentru blocuri de câte 2 ranguri binare.

Rank		7		6		5		4		3		2		1		0			
Operand		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	$c_{in}=0$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1		
	$c_{in}=1$	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0		
i=1	$c_{in}=0$	0	1		1	0	1		1	1	0		0	0	1		0		
	$c_{in}=1$	1	0		0	1	0		0	1	0		1						
i=2	$c_{in}=0$																		
	$c_{in}=1$																		
i=3	$c_{in}=0$																		
	$c_{in}=1$																		

2.3.4 Conditional Sum Adder (contin.)

Nivelul al treilea: determinarea variantelor de sume corecte pentru blocuri de câte 4 ranguri binare.

Rank Operand		7		6		5		4		3		2		1		0	
		C	S	C	S	C	S	C	S	C	S	C	S	C	S		
X		1	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1
Y		0	1	1	1	1	1	0	0	1	1	0	0	1	0	0	1
Block level	Carry in	C	S	C	S	C	S	C	S	C	S	C	S	C	S	C	S
i=0	$c_{in}=0$	0	1	0	1	0	1	0	1	0	1	1	0	0	0	1	0
	$c_{in}=1$	1	0	1	0	1	0	1	0	1	0	1	1	0	1	<div></div>	
i=1	$c_{in}=0$	0	1	1	0	1	0	1	1	1	0	0	1	0	1		
	$c_{in}=1$	1	0	0	1	0	1	0	0	1	0	1	1	<div></div>			
i=2	$c_{in}=0$	0	1	1	0	1	0	1	1	1	0	0	1				
	$c_{in}=1$	1	0	0	1	0	1	0	0	1	0	1	1	<div></div>			
i=3	$c_{in}=0$																
	$c_{in}=1$																

2.3.4 Conditional Sum Adder (contin.)

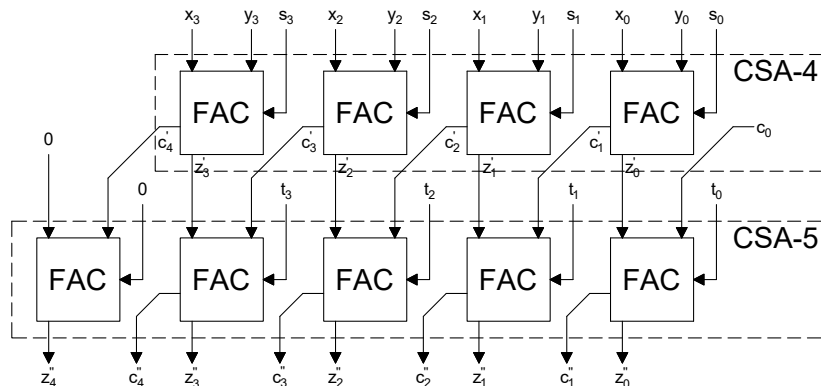
Nivelul al patrulea: determinarea sumei corecte pentru întreg setul de ranguri al celor 2 operanzi.

Rank		7		6		5		4		3		2		1		0					
Operand		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	
i=0	$c_{in}=0$	0	1	0	1	0	1	0	1	0	1	1	0	0	0	1	0				
	$c_{in}=1$	1	0	1	0	1	0	1	0	1	0	1	1	0	1						
i=1	$c_{in}=0$	0	1		1	0	1		1	1	0		0	0	1		0				
	$c_{in}=1$	1	0		0	1	0		0	1	0		1								
i=2	$c_{in}=0$	0	1		1		1		1	1	0		0		1		0				
	$c_{in}=1$	1	0		0		0		0												
i=3	$c_{in}=0$	1	0		0		0		0		0		0		1		0				
	$c_{in}=1$																				

2.3.5 Carry Save Adder

- ▶ sumă în format redundant: 2 vectori $\begin{cases} \text{sumă} \\ \text{transport} \end{cases}$
- ▶ vectorul transport este cu o poziție mai semnificativ decât cel sumă
- ▶ permite realizarea adunării multi-operand

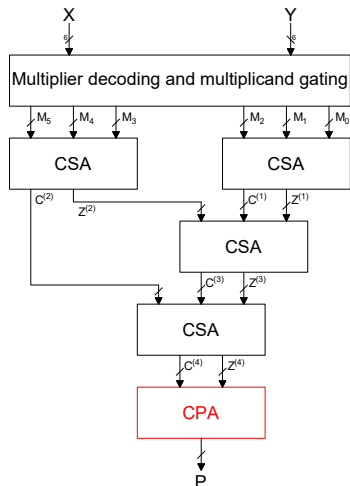
Se consideră operanzii X , Y , S și T , pe 4 biți. Suma $Z = X + Y + S + T$ poate fi calculată astfel:



2.3.5 Carry Save Adder (contin.)

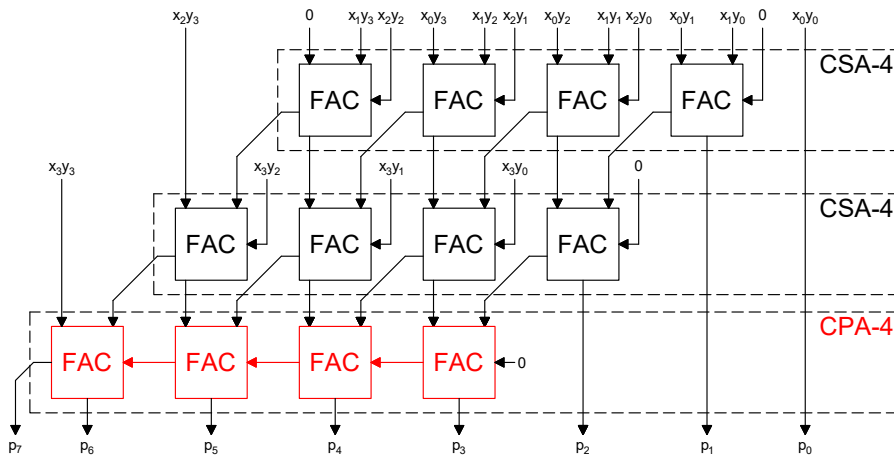
- facilitează realizarea operației de înmulțire (combi-național)

Fie X și Y fără semn pe 6 biți. Produsul $P = X * Y$ este obținut prin adunarea produselor de 1-bit $M_i = x_i * Y * 2^i$



2.3.5 Carry Save Adder (contin.)

Structura de mai jos permite calcularea produsului a două numere fără semn pe 4 biți:



2.3.5 Carry Save Adder (contin.)

Exemplu:

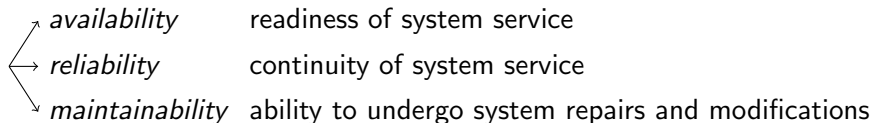
	x_3	x_2	x_1	x_0		
$X=15$	1	1	1	1		
$Y=13$	1	1	0	1		
<hr/>						
$M_0=x_0Y2^0$	1	1	0	1	} + _{CSA}	
$M_1=x_1Y2^1$	1	1	0	1		0
$M_2=x_2Y2^2$	1	1	0	1		0
<hr/>						
$Z^{(1)}$	1	0	0	0	1	1
$C^{(1)}$	0	1	1	1	0	0
<hr/>						
$2C^{(1)}$	0	1	1	1	0	0
$Z^{(1)}$	1	0	0	0	1	1
<hr/>						
$M_3=x_3Y2^3$	1	1	0	1	0	0
<hr/>						
$Z^{(2)}$	1	1	1	0	0	1
$C^{(2)}$	0	1	0	1	0	0
<hr/>						
$2C^{(2)}$	0	1	0	1	0	0
$Z^{(2)}$	1	1	1	0	0	1
<hr/>						
P	1	1	0	0	0	1
						1
						=195

2.4 Reliable computing

Attributes of computing systems:

- ▶ performance
- ▶ power consumption
- ▶ **reliability**

Attributes of reliable computing:



→ <i>availability</i>	readiness of system service
→ <i>reliability</i>	continuity of system service
→ <i>maintainability</i>	ability to undergo system repairs and modifications

2.4.1 Binary adders with parity control

Chain of Dependability and Security Threats:



Increase reliability by using parity control:

- ▶ attach an even parity bit to addition's operands X , Y and Z

Consider operands on n bits:

$$X \rightarrow x_p = x_{n-1} \oplus x_{n-2} \oplus \cdots \oplus x_1 \oplus x_0$$

$$Y \rightarrow y_p = y_{n-1} \oplus y_{n-2} \oplus \cdots \oplus y_1 \oplus y_0$$

$$Z \rightarrow z_p = z_{n-1} \oplus z_{n-2} \oplus \cdots \oplus z_1 \oplus z_0 \text{ equation (1)}$$

2.4.1 Binary adders with parity control (contd.)

However, the even parity bit of Z , z_p , can be predicted in a different manner, starting from the definition of z_i :

$$z_i = x_i \oplus y_i \oplus c_i$$

In order to predict the value of z_p a new value needs to be defined, associated with all but the most significant carry:

$$c_p = c_{n-1} \oplus c_{n-2} \oplus \cdots \oplus c_1 \oplus c_0$$

In equation (1), one can replace each z_i as expressed above and after regrouping, the new expression for z_p is obtained as

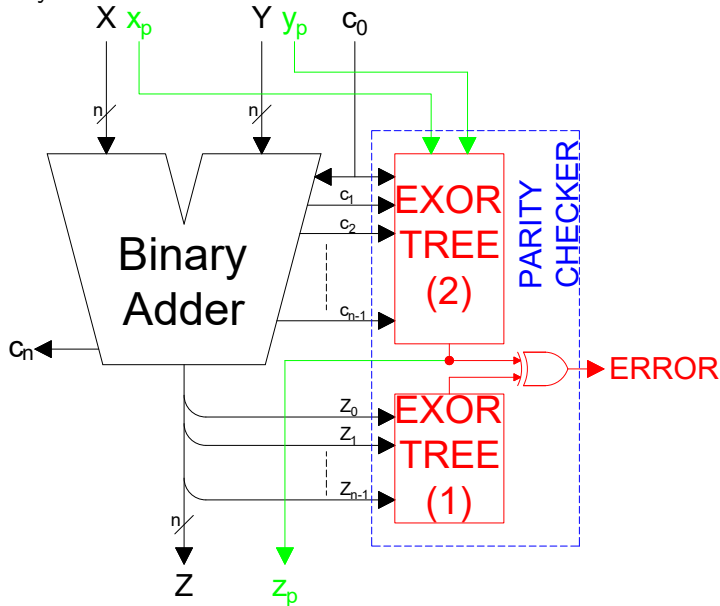
$$\begin{aligned} z_p &= x_{n-1} \oplus y_{n-1} \oplus c_{n-1} \oplus x_{n-2} \oplus y_{n-2} \oplus c_{n-2} \oplus \cdots \oplus x_0 \oplus y_0 \oplus c_0 \\ z_p &= x_p \oplus y_p \oplus c_p \text{ equation (2)} \end{aligned}$$

Equation (2):

- ▶ determines z_p faster than (1)
- ▶ anticipates the value of z_p from (1)


2.4.1 Binary adders with parity control (contd.)

Parity checked adder:



2.4.1 Binary adders with parity control (contd.)

Errors that can affect the binary adder:

- ▶ single faults 
 - more difficult to detect
 - higher probability
- ▶ multiple faults

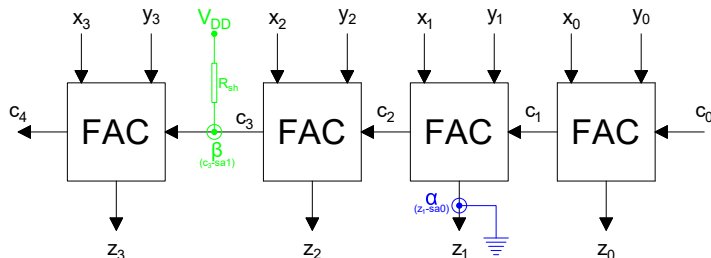
Single fault with logic manifestation: Single Stuck-at Fault (SSaF)

- ▶ *logic manifestation*: the fault affects the logic value of at least one signal

2.4.1 Binary adders with parity control (contd.)

Consider a RCA on 4 bits, being affected by two defects

- ▶ fault z_1 Stuck-at-0 (SA0): affects z_1 ; no effect over $c_2 \Rightarrow$ an odd number of incorrect bits
- ▶ fault c_3 Stuck-at-1 (SA1): affects c_3 ; because z_3 depends linearly on c_3 , z_3 is also affected; c_4 can or cannot be affected: if c_4 is affected, for a longer adder, implicitly z_4 will be affected \Rightarrow an even number of incorrect bits



2.4.1 Binary adders with parity control (contd.)

Consider operands $X = 0011$, $Y = 0011$ and $c_{in} = 0$.

In the fault-free addition case, because the parity bit z_p calculated with equation (1) is the same as the one calculated with equation (2), the conclusion is that no error occurred.

$$\begin{array}{rcll} X = & 0 & 0 & 1 & 1 & x_p = 0 \oplus 0 \oplus 1 \oplus 1 = 0 \\ Y = & 0 & 0 & 1 & 1 & y_p = 0 \oplus 0 \oplus 1 \oplus 1 = 0 \\ C = & 0 & 0 & 1 & 1 & 0 & c_p = 0 \oplus 1 \oplus 1 \oplus 0 = 0 \\ Z = & & 0 & 1 & 1 & 0 & z_p(2) = x_p \oplus y_p \oplus c_p = 0 \end{array}$$

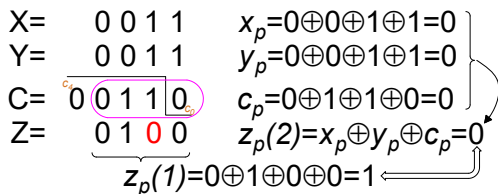
$\underbrace{\hspace{1.5cm}}_{z_p(1)=0 \oplus 1 \oplus 1 \oplus 0 = 0}$

2.4.1 Binary adders with parity control (contd.)

For the addition case where z_1 is SA0 (fault α is active) the parity bit z_p calculated with equation (1) differs to the one calculated with equation (2), the conclusion being that an error occurred.

$$\begin{array}{rcll} X= & 0 & 0 & 1 & 1 & x_p=0\oplus 0\oplus 1\oplus 1=0 \\ Y= & 0 & 0 & 1 & 1 & y_p=0\oplus 0\oplus 1\oplus 1=0 \\ C= & \overset{c_p}{0} & \overset{c_p}{0} & \overset{c_p}{1} & \overset{c_p}{1} & \overset{c_p}{0} & c_p=0\oplus 1\oplus 1\oplus 0=0 \\ Z= & 0 & 1 & \color{red}{0} & 0 & z_p(2)=x_p\oplus y_p\oplus c_p=0 \end{array}$$

$\underbrace{\hspace{1.5cm}}_{z_p(1)=0\oplus 1\oplus 0\oplus 0=1}$



2.4.1 Binary adders with parity control (contd.)

For the addition case in which c_3 is SA1 (fault β is active) the parity bit z_p calculated with equation (1) is the same as the one calculated with equation (2), the **incorrect** conclusion being that no error occurred.

$$\begin{array}{rcll} X= & 0 & 0 & 1 & 1 & x_p=0\oplus 0\oplus 1\oplus 1=0 \\ Y= & 0 & 0 & 1 & 1 & y_p=0\oplus 0\oplus 1\oplus 1=0 \\ C= & 0 & \textcolor{red}{1} & 1 & 1 & 0 & c_p=1\oplus 1\oplus 1\oplus 0=1 \\ Z= & \textcolor{red}{1} & 1 & 1 & 0 & z_p(2)=x_p\oplus y_p\oplus c_p=1 \end{array}$$

$\underbrace{\hspace{1.5cm}}_{z_p(1)=1\oplus 1\oplus 1\oplus 0=1}$

2.4.1 Binary adders with parity control (contd.)

Single parity bit cannot detect all possible SSaF:

- ▶ it cannot detect errors affecting the carry chain because these error results in an even number of modified bits

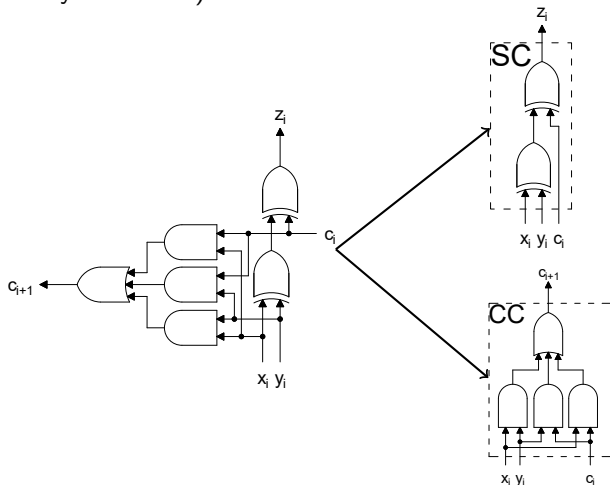
Methods for detection of SSaF affecting the carry-chain:

- Carry chain duplication
- Carry Dependent Sum Adder

2.4.2 Carry chain duplication

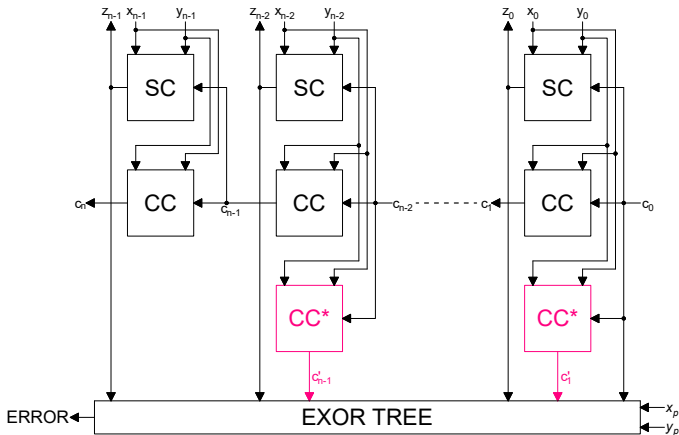
(A) Carry chain duplication applied to RCA

A FAC is split in two parts: one computing the sum bit (the Sum Cell - SC) and the other for generation of the carry output (the Carry Cell - CC)



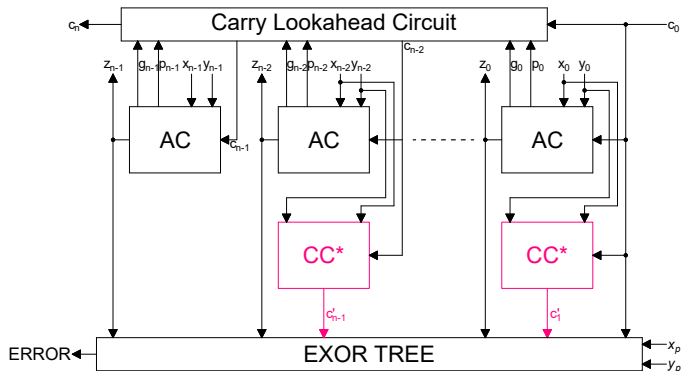
2.4.2 Carry chain duplication (contd.)

The RCA having the carry chain duplicated is depicted in the diagram below:



2.4.2 Carry chain duplication (contd.)

Ⓑ Carry chain duplication applied to the Carry Lookahead Adder

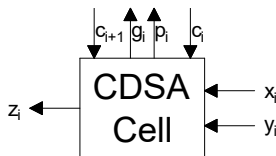


2.4.3 Carry Dependent Sum Adder

Addresses the reliability issue from the design phase:

- ▶ when c_{i+1} is incorrect, the Carry Dependent Sum Adder (CDSA) will force sum bit z_i to be incorrect, as well
 - ▶ create an artificial imbalance in the number of incorrect bits
 - ▶ beside x_i , y_i and c_i , the c_{i+1} carry bit is an input to the CDSA cell
⇒ carries are to be obtained in a lookahead manner

The CDSA cell symbol:



2.4.3 Carry Dependent Sum Adder (contd.)

CDSA cell truth table:

Inputs				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
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

2.4.3 Carry Dependent Sum Adder (contd.)

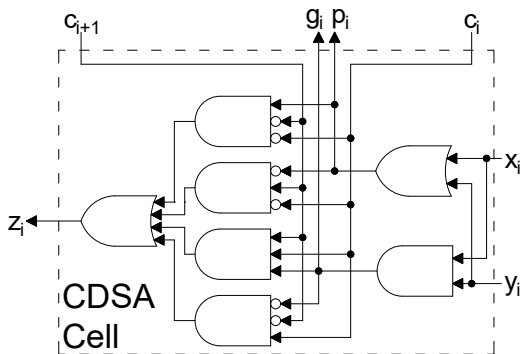
CDSA cell truth table and output equation:

z_i x_i, y_i		c_i, c_{i+1}			
		00	01	11	10
00	0	1	0	1	
01	1	0	0	1	
11	1	0	1	0	
10	1	0	0	1	

$$\begin{aligned}
 z_{i+1} &= y_i \cdot \overline{c_i} \cdot \overline{c_{i+1}} + x_i \cdot \overline{c_i} \cdot \overline{c_{i+1}} + \overline{x_i} \cdot \overline{y_i} \cdot \overline{c_i} \cdot c_{i+1} + \\
 &\quad x_i \cdot y_i \cdot c_i \cdot c_{i+1} + \overline{x_i} \cdot c_i \cdot \overline{c_{i+1}} + \overline{y_i} \cdot c_i \cdot \overline{c_{i+1}} \\
 &= (x_i + y_i) \cdot \overline{c_i} \cdot \overline{c_{i+1}} + \overline{x_i + y_i} \cdot \overline{c_i} \cdot c_{i+1} + \\
 &\quad x_i \cdot y_i \cdot c_i \cdot c_{i+1} + \overline{x_i \cdot y_i} \cdot c_i \cdot \overline{c_{i+1}} \\
 &= p_i \cdot \overline{c_i} \cdot \overline{c_{i+1}} + \overline{p_i} \cdot \overline{c_i} \cdot c_{i+1} + \\
 &\quad g_i \cdot c_i \cdot c_{i+1} + \overline{g_i} \cdot c_i \cdot \overline{c_{i+1}}
 \end{aligned}$$

2.4.3 Carry Dependent Sum Adder (contd.)

CDSA cell synthesis:



2.4.3 Carry Dependent Sum Adder (contd.)

Using the synthesized cell, the diagram bellow presents the CDSA architecture:

