# Arhitectura Calculatoarelor

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

16 Octombrie 2024
23 Octombrie 2024
30 Octombrie 2024
6 Noiembrie 2024
13 Noiembrie 2024
20 Noiembrie 2024

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

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 $\Uparrow$;

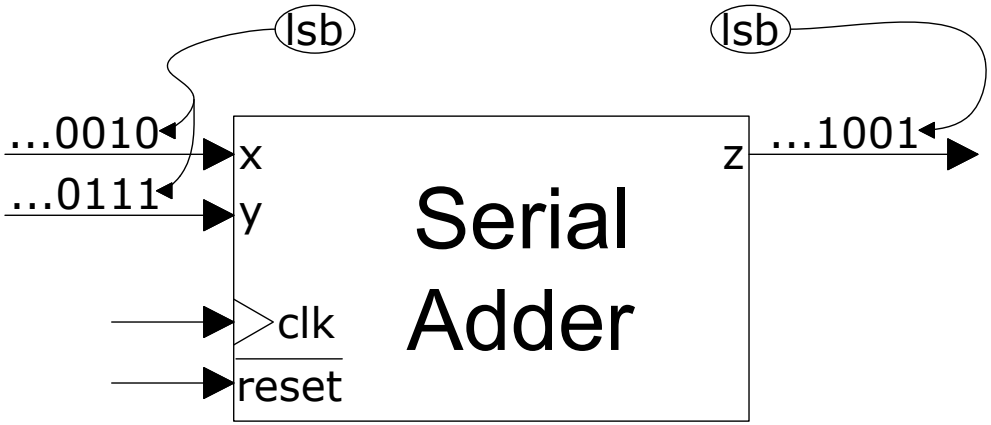
Tipuri de sumatoare seriale :
- LSDF (Least significant digit first)
- MSDF (Most significant digit first)

## 2.1 - Sumatoare seriale (contin.)

LSDF:

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

Simbolul sumatorului serial:

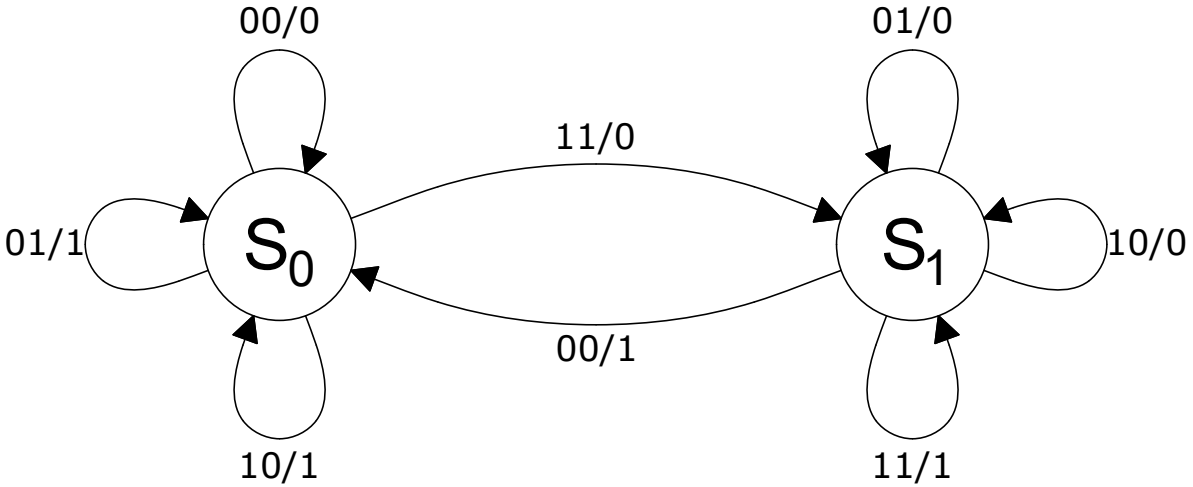



Propagarea transportului: utilizează starea internă a adunătorului

$\Rightarrow$ 2 stări interne
- $S_0$ : fără propagare de carry din rangul anterior
- $S_1$ : cu propagare de carry din rangul anterior

## 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 |

(C) 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 ($\lceil log_2 s \rceil = \lceil log_2 2 \rceil = 1$)

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

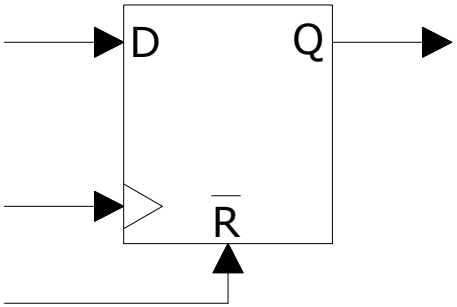
(D) Tabel de tranziție:

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

(E) Tabele de excitație:

- dependent de tipul elementelor de stocare utilizate

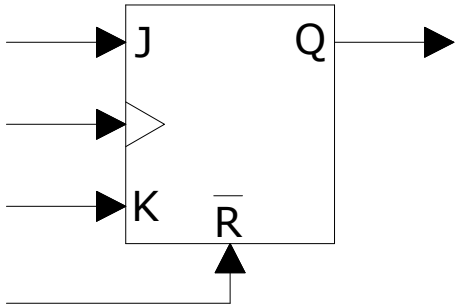
Flip-flop D




$Q(t+1)=D$

| 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 |

Flip-flop J-K




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

| 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 |

(F) Ecuațiile de ieșire și feedback:

Ecuațiile de ieșire:

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

Ecuațiile feedback:

| $D$ | $x, y$ = 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| $w$ = 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 |

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

| $J$ | $x, y$ = 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| $w$ = 0 | 0 | 0 | 1 | 0 |
| 1 | d | d | d | d |

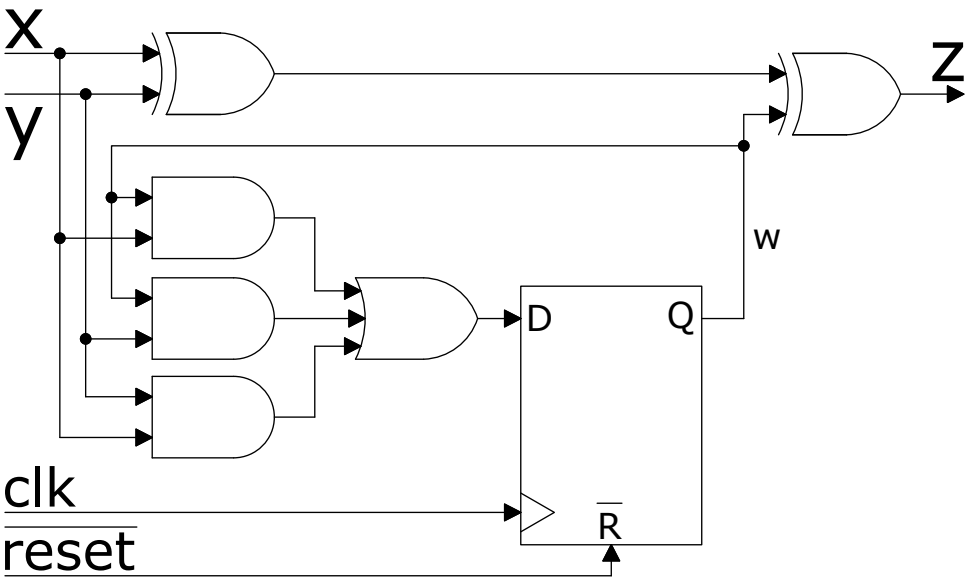
$J = x\cdot y$

| $K$ | $x, y$ = 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| $w$ = 0 | d | d | d | d |
| 1 | 1 | 0 | 0 | 0 |

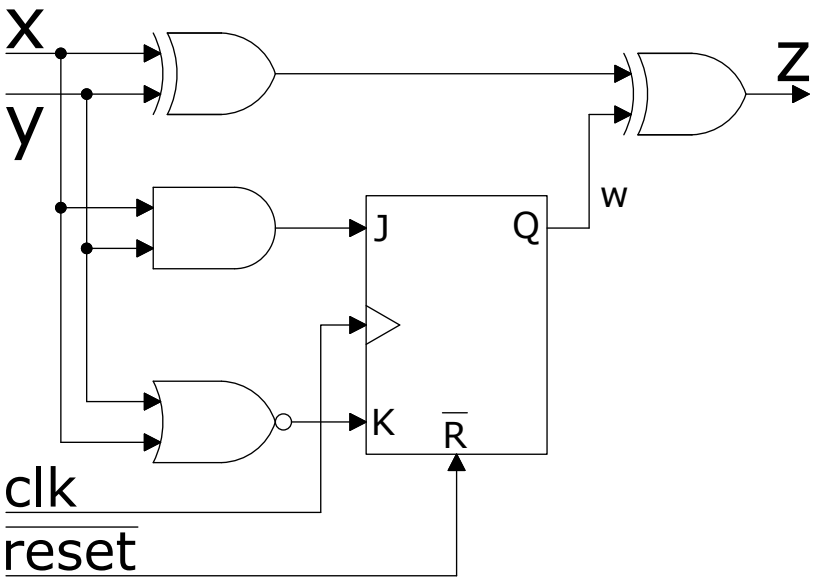
$K = \overline{x}\cdot\overline{y} = \overline{x+y}$

(G) Sinteza sumatorului serial:

Folosind flip-flop-uri de tip D




Folosind flip-flop-uri de tip J-K

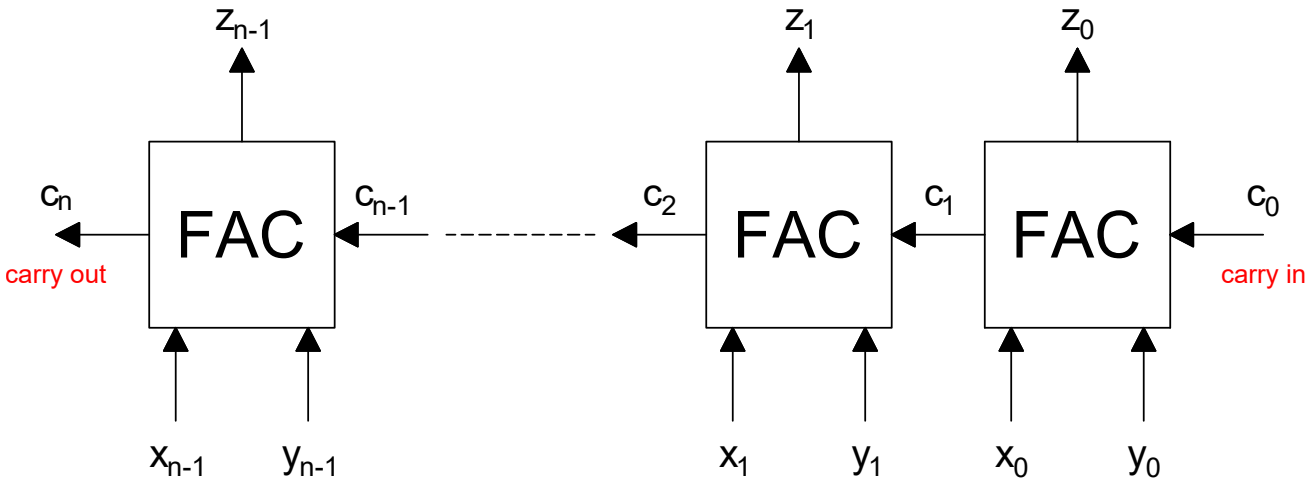


## 2.2.1 - Sumatoare paralele bazate pe propagarea serială a transportului

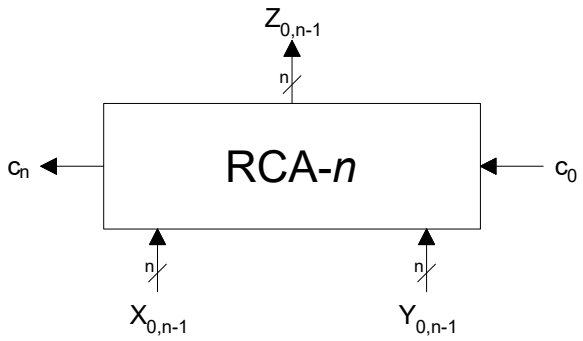
Ripple Carry Adder (RCA): utilizează celule dedicate de însumare pentru fiecare rang binar

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

Arhitectură RCA pe *n* biți:

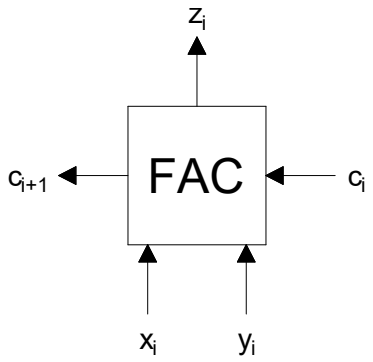



Simbolul unui sumator RCA pe *n* biți:



Full Adder Cell (FAC):

- simbol:




- tabel de adevăr:

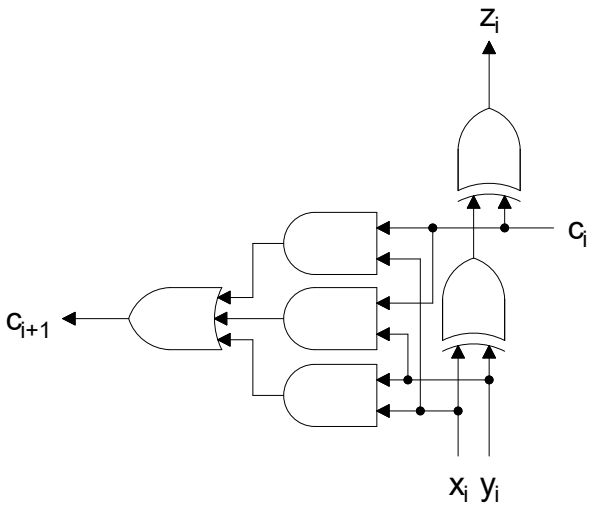
| Inputs | | | Outputs | |
|---|---|---|---|---|
| $x_i$ | $y_i$ | $c_i$ | $z_i$ | $c_{i+1}$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

- ecuatiile iesirilor:

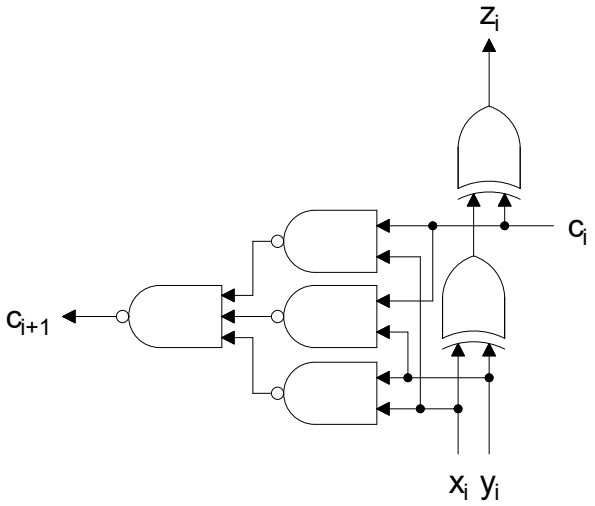
$$\begin{cases} z_i = x_i \oplus y_i \oplus c_i \\ c_{i+1} = x_i \cdot y_i + x_i \cdot c_i + y_i \cdot c_i \end{cases}$$

Sinteza FAC:

(A) porți de tip EXOR, AND, OR:

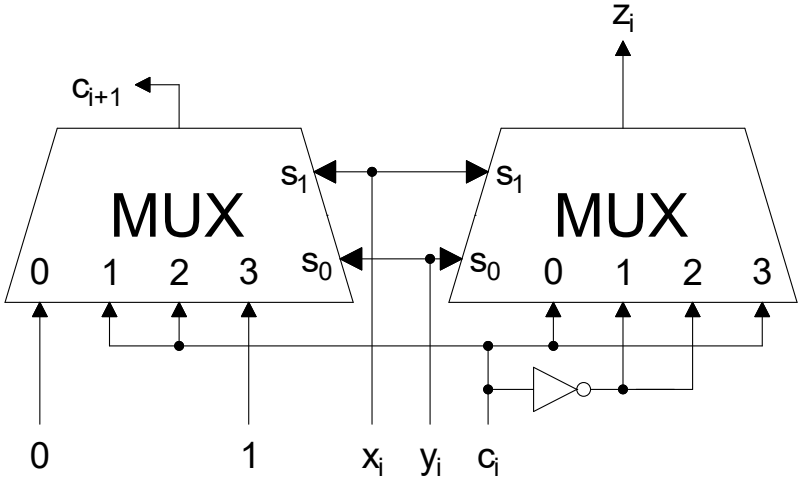



(B) porți de tip EXOR, NAND:

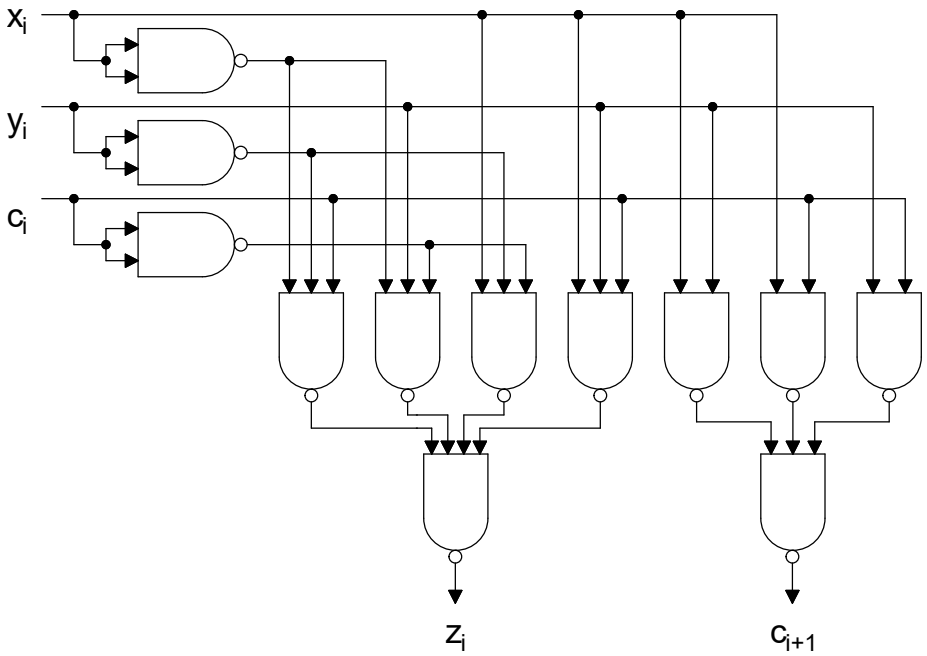


Sinteza FAC:

(C) multiplexoare:




(D) porți de tip NAND:

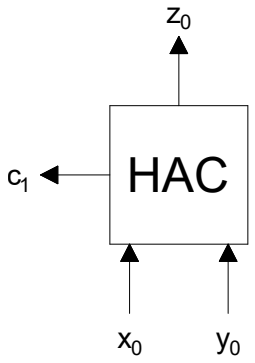


Dacă $c_0 = 0 \Rightarrow$ cea mai din dreapta FAC poate fi simplificată:

- ecuațiile ieșirilor:

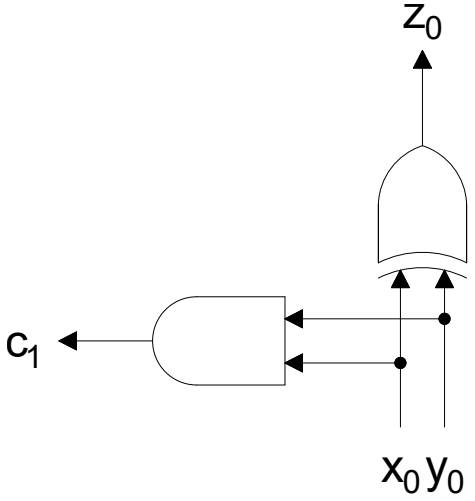
$$\begin{cases} z_0 = x_0 \oplus y_0 \cancel{\oplus c_0} & = x_0 \oplus y_0 \\ c_1 = x_0 \cdot y_0 + \cancel{x_0 \cdot c_0} + \cancel{y_0 \cdot c_0} & = x_0 \cdot y_0 \end{cases}$$

- simbol:



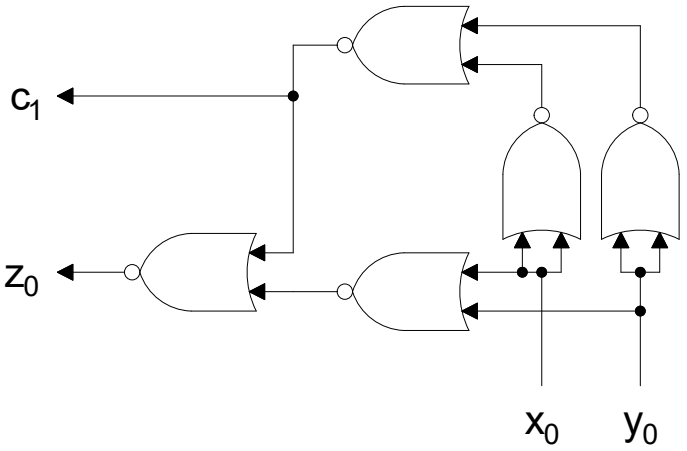

Sinteza Half Adder Cell (HAC):

A' porți de tip EXOR, AND:



Sinteza HAC: curs 4

(B') porți de tip NOR



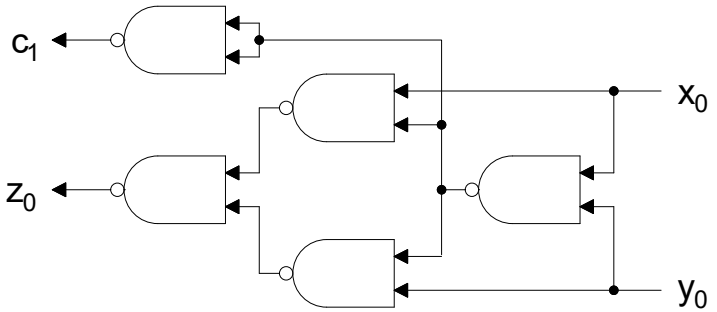

- justificare implementare:

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

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

Sinteza HAC:

C' porți de tip NAND




- justificare implementare:

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

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

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

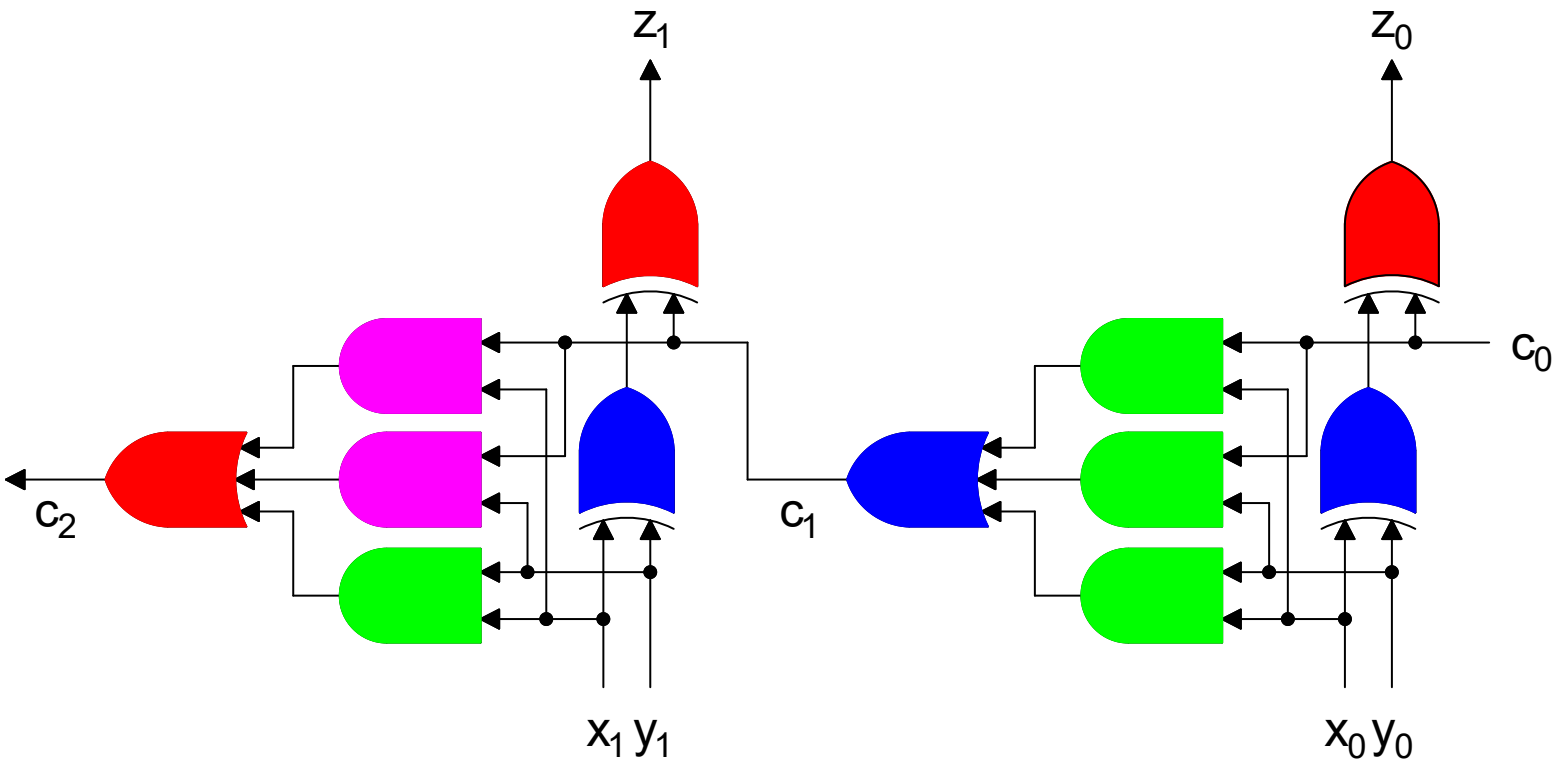
Calea critică:

- calea de propagare din intreg circuitul corespunzatoare intarierii 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*

Calea critică pentru un RCA pe 2 biți:




Întârizerea unui segment RCA pe $n$ biți:

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

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

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: $\nu$

Tabel de adevar pentru determinarea condiției de overflow:

| Inputs | | | Outputs | |
|---|---|---|---|---|
| $x_{n-1}$ | $y_{n-1}$ | $c_{n-1}$ | $z_{n-1}$ | $\nu$ |
| 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 |

Forma minimă a condiției de overflow este obținuta 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:

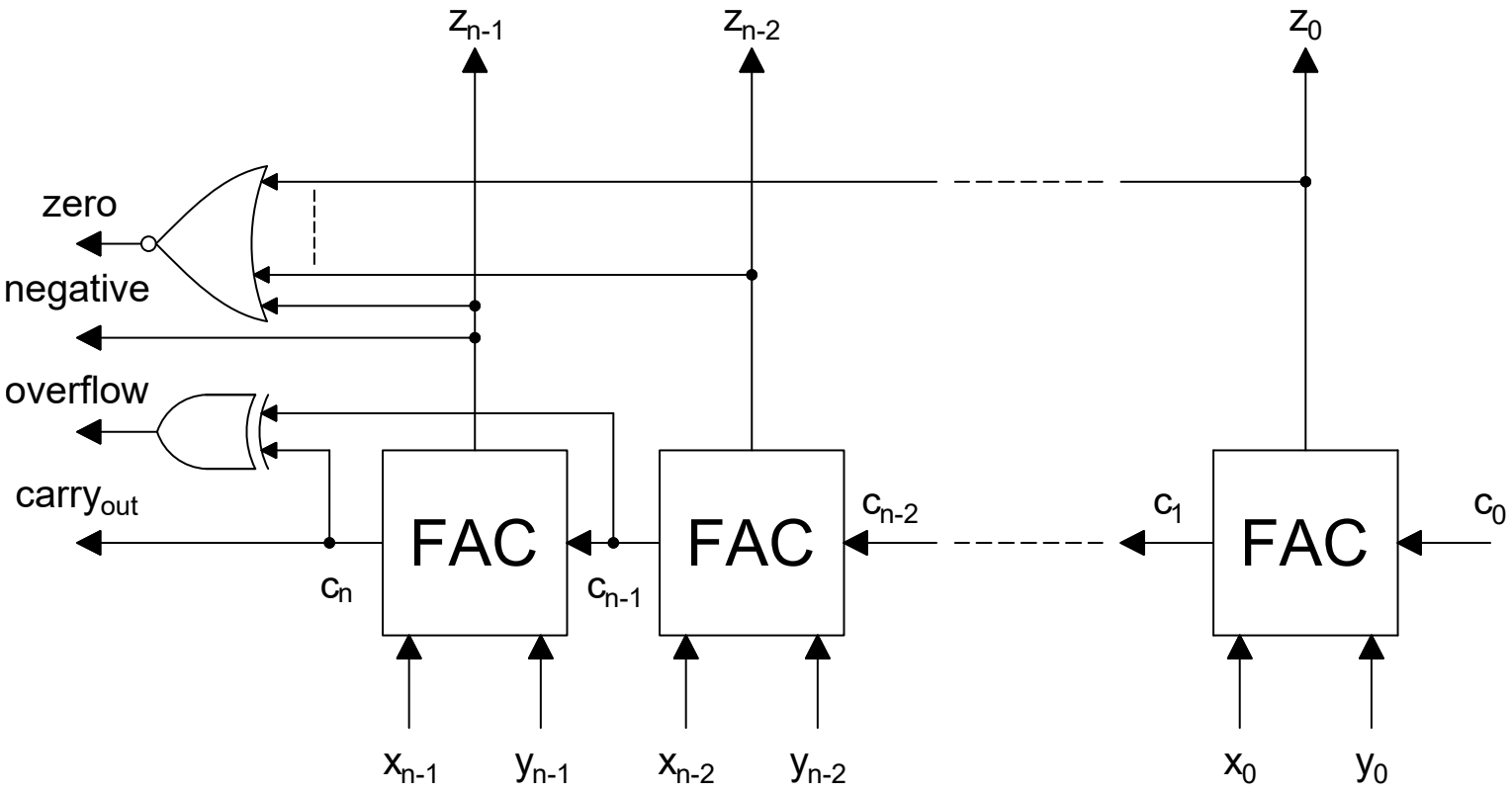
- $I_1$: $(A \oplus B) \cdot C = A \cdot C \oplus B \cdot C$
- $I_2$: $(A + B) = A \oplus B \oplus A \cdot B$
- $I_2'$: $A \oplus B = (A + B) \oplus A \cdot B$

Forma simplificată a condiției de overflow este obținuta ca:

$$
\begin{aligned}
\nu &= \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot c_{n-1} + x_{n-1} \cdot y_{n-1} \cdot \overline{c_{n-1}} \\
&\overset{I_2}{=} \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \cdot \overline{c_{n-1}} \\
&= \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \cdot (1 \oplus c_{n-1}) \\
&\overset{I_1}{=} \overline{x_{n-1}} \cdot \overline{y_{n-1}} \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \\
&\overset{I_1}{=} (\overline{x_{n-1}} \cdot \overline{y_{n-1}} \oplus x_{n-1} \cdot y_{n-1}) \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \\
&\overset{I_2'}{=} (\overline{x_{n-1}} \cdot \overline{y_{n-1}} + x_{n-1} \cdot y_{n-1}) \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \\
&= (x_{n-1} \oplus y_{n-1} \oplus 1) \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \\
&\overset{I_1}{=} x_{n-1} \cdot c_{n-1} \oplus y_{n-1} \cdot c_{n-1} \oplus x_{n-1} \cdot y_{n-1} \oplus c_{n-1} \\
&\overset{I_2'}{=} (x_{n-1} \cdot c_{n-1} + y_{n-1} \cdot c_{n-1} + x_{n-1} \cdot y_{n-1}) \oplus c_{n-1}
\end{aligned}
$$

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

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



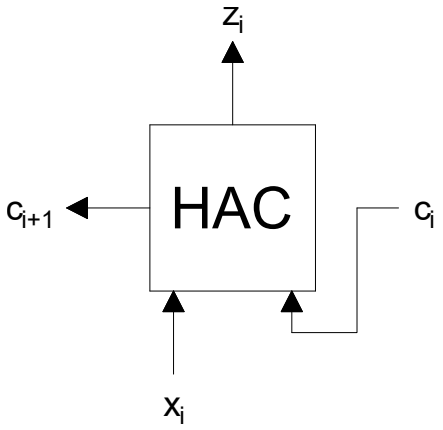
Adunarea cu o constantă:

- se consideră doar constante impare
  - **Întrebare**: de ce?
- operanzii $X$ și $Y$ - pe $n$ biți
- $Y$ - constant

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

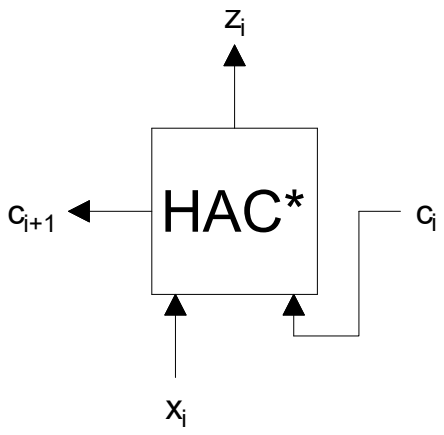
dacă $y_i = 0$:

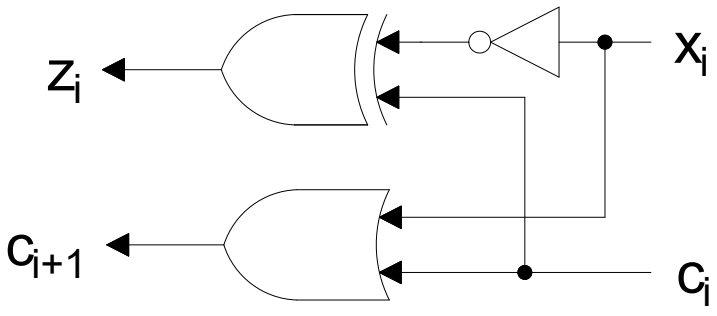
$$\left.\begin{cases} z_i = & x_i \oplus \cancel{0} \oplus c_i = & x_i \oplus c_i \\ c_{i+1} = & \cancel{x_i \cdot 0} + x_i \cdot c_i + \cancel{0 \cdot c_i} = & x_i \cdot c_i \end{cases}\right\}\text{HAC}$$



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

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



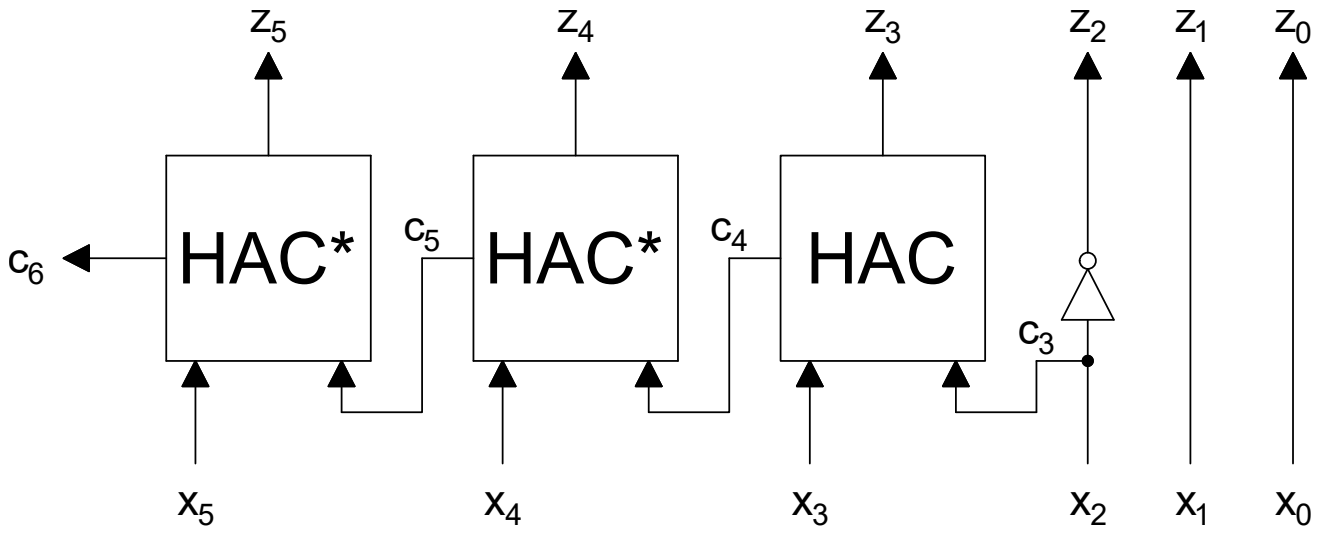

Exemplu de adunare cu o constantă având operanzi pe 6 biți:

- $X = x_5x_4x_3x_2x_1x_0$
- $Y = y_5y_4y_3y_2y_1y_0$ - operand constant
  - fie $Y = 110100_2$
- $Z = X + Y$, cu $c_0 = 0$

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

$$z_0 = x_0 \oplus 0 \oplus 0 = x_0 \qquad c_1 = x_0 \cdot 0 + x_0 \cdot 0 + 0 \cdot 0 = 0$$
$$z_1 = x_1 \oplus 0 \oplus 0 = x_1 \qquad c_2 = x_1 \cdot 0 + x_1 \cdot 0 + 0 \cdot 0 = 0$$
$$z_2 = x_2 \oplus 1 \oplus 0 = \overline{x_2} \qquad c_3 = x_2 \cdot 1 + x_2 \cdot 0 + 1 \cdot 0 = x_2$$

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 $X_i + Y_i$

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

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

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

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

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

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

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 $c^\star z_3^\star z_2^\star z_1^\star z_0^\star \geq 10$, inegalitate din urmă care poate fi rescrisă astfel:

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

Condiția C1 implică:

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

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

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

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

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

$$c^\star = 1$$

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

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

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

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

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

$$\begin{array}{r|cccc|}
15 & - & = & 1111 & \\
10 & & = & 1010 & \\
\hline
5 & & = & 0101 &
\end{array}
\qquad
\begin{array}{r|cccc|}
15 & + & = & 1111 & \\
6 & & = & 0110 & \\
\hline
5 & & = & \cancel{1}\ 0101 &
\end{array}$$

$$\begin{array}{r|cccc|}
18 & - & = & 1\ 0010 & \\
10 & & = & 1010 & \\
\hline
8 & & = & 1000 &
\end{array}
\qquad
\begin{array}{r|cccc|}
18 & + & = & 1\ 0010 & \\
6 & & = & 0110 & \\
\hline
8 & & = & \cancel{1}\ 1000 &
\end{array}$$

Corecția lui $Z_i$ depinde de următoarea condiție booleană:

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

$$\begin{cases} 0 \\ (X_i + Y_i < 10) \end{cases} \Rightarrow \begin{cases} Z_i = & z_3^\star & z_2^\star & z_1^\star & z_0^\star & + \\ & 0 & 0 & 0 & 0 & (0) \\ c_{i+1} = & 0 \end{cases}$$

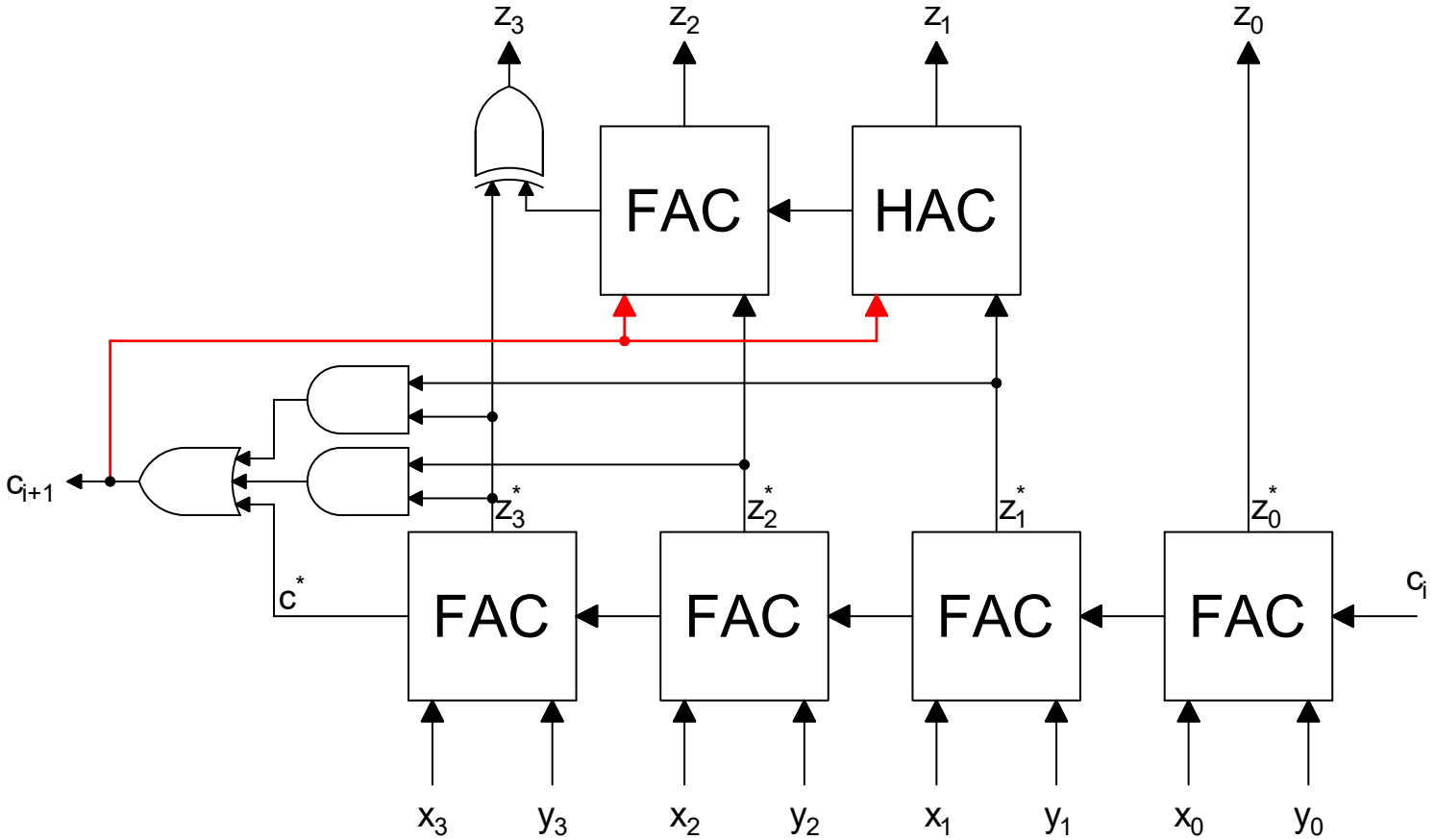
Transportului de ieșire, $c_{i+1}$ se obține ca: $c_{i+1} = c^\star + z_3^\star \cdot z_2^\star + z_3^\star \cdot z_1^\star$

Stagiul de corecție pt. $Z_i$ devine:

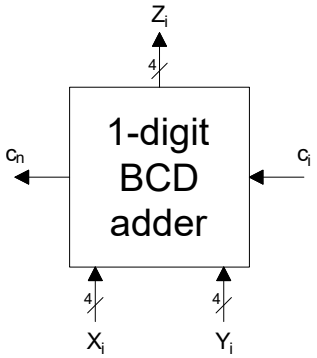
$$\begin{array}{lccccc} Z_i = & z_3^\star & z_2^\star & z_1^\star & z_0^\star & + \\ & 0 & c_{i+1} & c_{i+1} & 0 & \end{array}$$

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

Sumatorul pentru tetrade BCD:

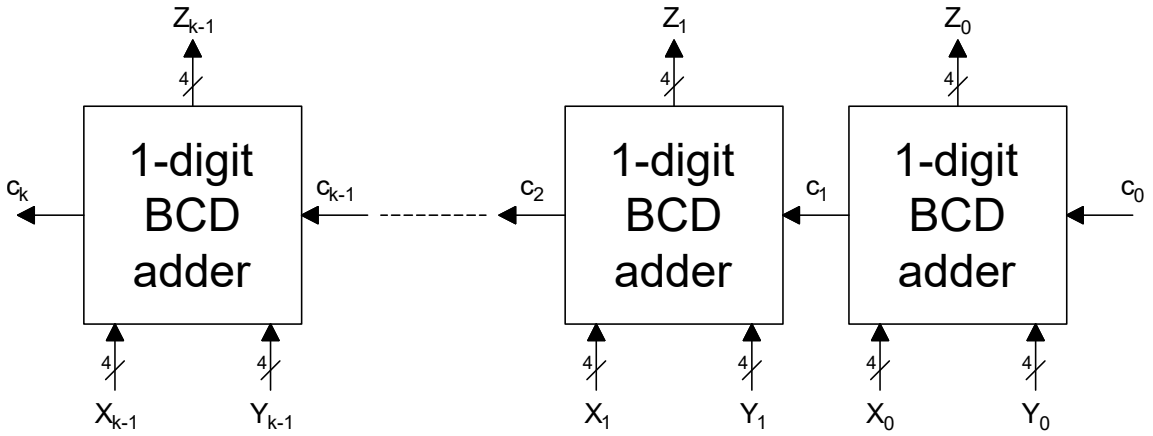



având simbolul:

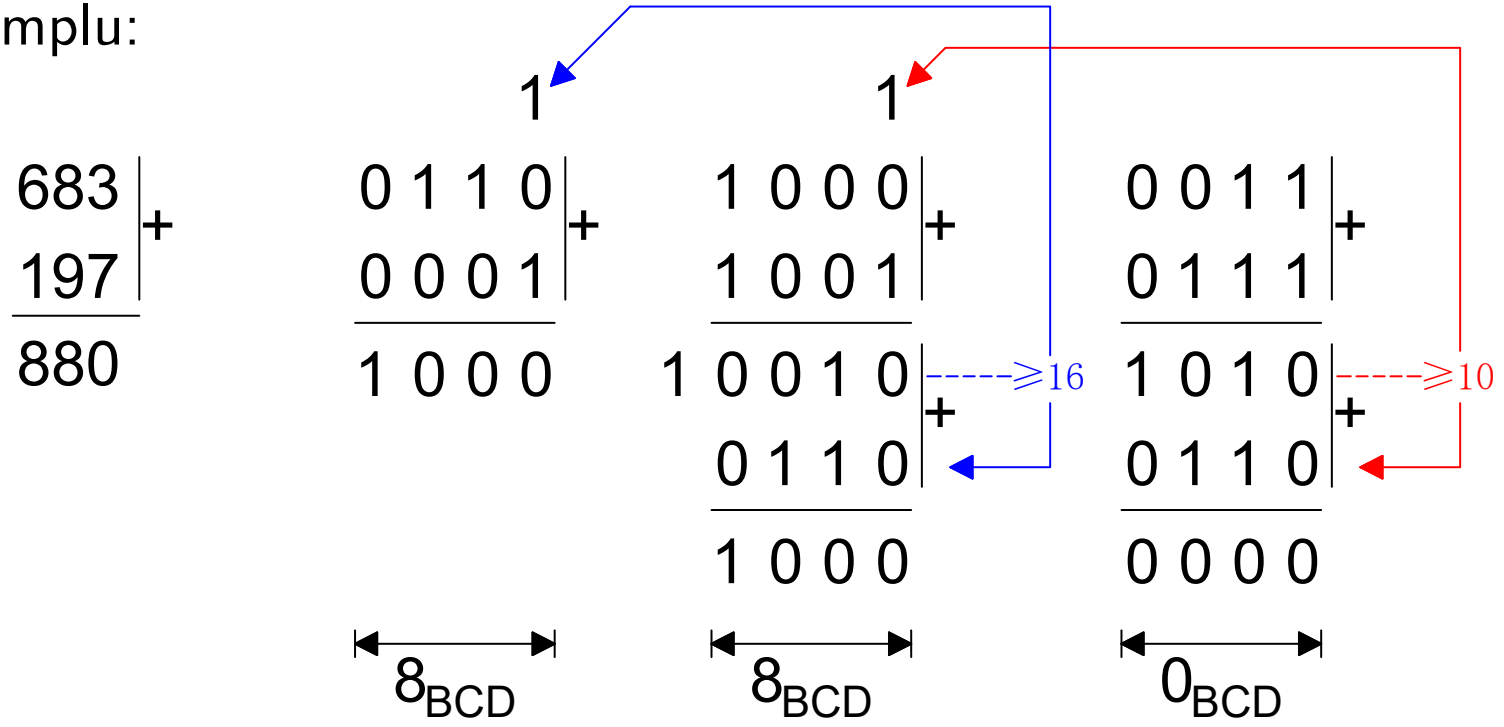


## 2.2.2 - Sumatoare zecimale bazate pe propagarea serială a transportului (contin.)

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




Exemplu:



(B) Adunare E3:

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

- $X_{i_{E3}} = x_3x_2x_1x_0$, $Y_{i_{E3}} = y_3y_2y_1y_0$, $Z_{i_{E3}} = z_3z_2z_1z_0$

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

- $X_{i_{E3}} = X_i + 3$, $Y_{i_{E3}} = Y_i + 3$, $Z_{i_{E3}} = Z_i + 3$
  - $X_i, Y_i, Z_i$ sunt cifre BCD

$$X_{i_{E3}} + Y_{i_{E3}} \begin{cases} Z_{i_{E3}} & : \text{cifra sumă} \\ c_{i+1} & : \text{transportul către cifra mai semnificativă} \end{cases}$$

$$\text{dacă } X_i + Y_i < 10 \begin{cases} Z_i = X_i + Y_i\big|_{+6} \Rightarrow Z_{i_{E3}} = X_{i_{E3}} + Y_{i_{E3}} - 3 \\ c_{i+1} = 0 \end{cases}$$

$$\text{dacă } X_i + Y_i \geq 10 \begin{cases} Z_i = X_i + Y_i - 10\big|_{+6} \Rightarrow Z_{i_{E3}} = X_{i_{E3}} + Y_{i_{E3}} - 13 \\ c_{i+1} = 1 \end{cases}$$

Pentru ambele cazuri $Z_{i_{E3}}$ necesită câte un pas de corecție.

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

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

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.

Corecția lui $Z_{i_{E3}}$ depinde de următoarea condiție booleană:

$$c'' \begin{cases} \begin{matrix} 1 \\ (X_i + Y_i \geq 10) \end{matrix} \Rightarrow \begin{cases} \begin{array}{lcccccc} Z_{i_{E3}} = & z''_3 & z''_2 & z''_1 & z''_0 & + \\ & 0 & 0 & 1 & 1 & (3) \\ \\ c_{i+1} = & 1 & & & & \end{array} \end{cases} \\ \begin{matrix} 0 \\ (X_i + Y_i < 10) \end{matrix} \Rightarrow \begin{cases} \begin{array}{lcccccc} Z_{i_{E3}} = & z''_3 & z''_2 & z''_1 & z''_0 & + \\ & 1 & 1 & 0 & 1 & (13) \\ \\ c_{i+1} = & 0 & & & & \end{array} \end{cases} \end{cases}$$

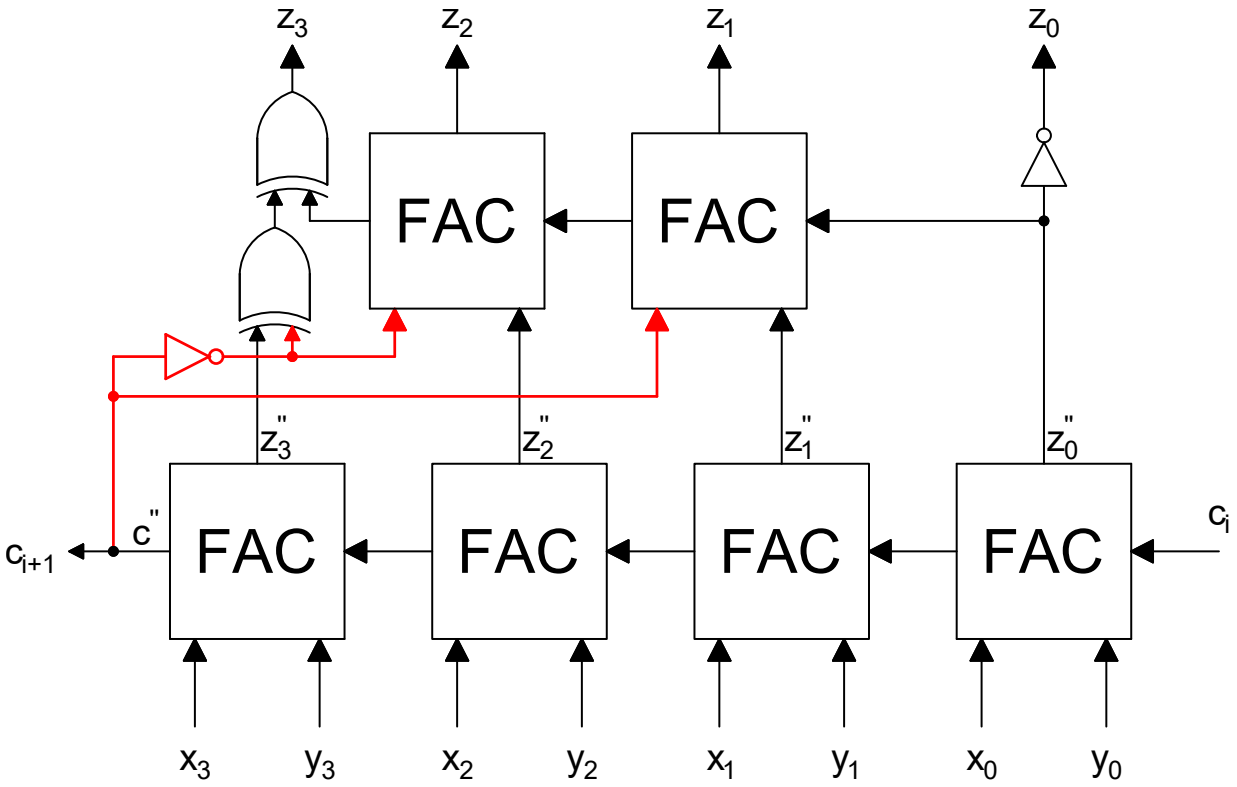
Transportului de ieșire, $c_{i+1}$ se obține ca: $c_{i+1} = c''$

Stagiul de corecție pentru $Z_{i_{E3}}$ devine:

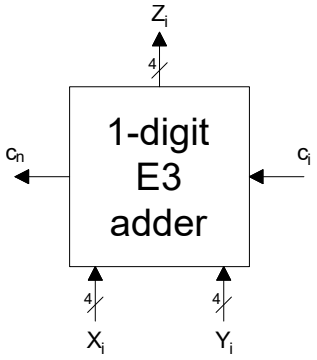
$$\begin{array}{lccccc} Z_{i_{E3}} = & z''_3 & z''_2 & z''_1 & z''_0 & + \\ & \overline{c_{i+1}} & \overline{c_{i+1}} & c_{i+1} & 1 & \end{array}$$

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

Sumatorul pentru tetrade E3:



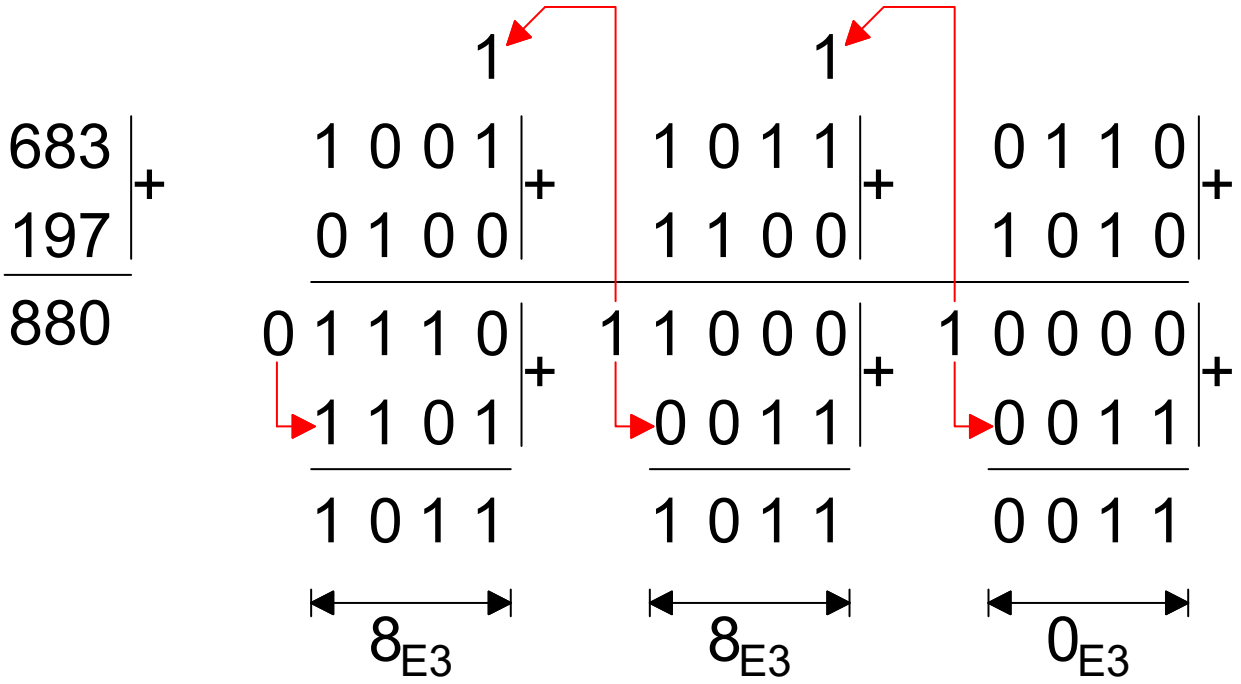

având simbolul:



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

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

Exemplu:




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

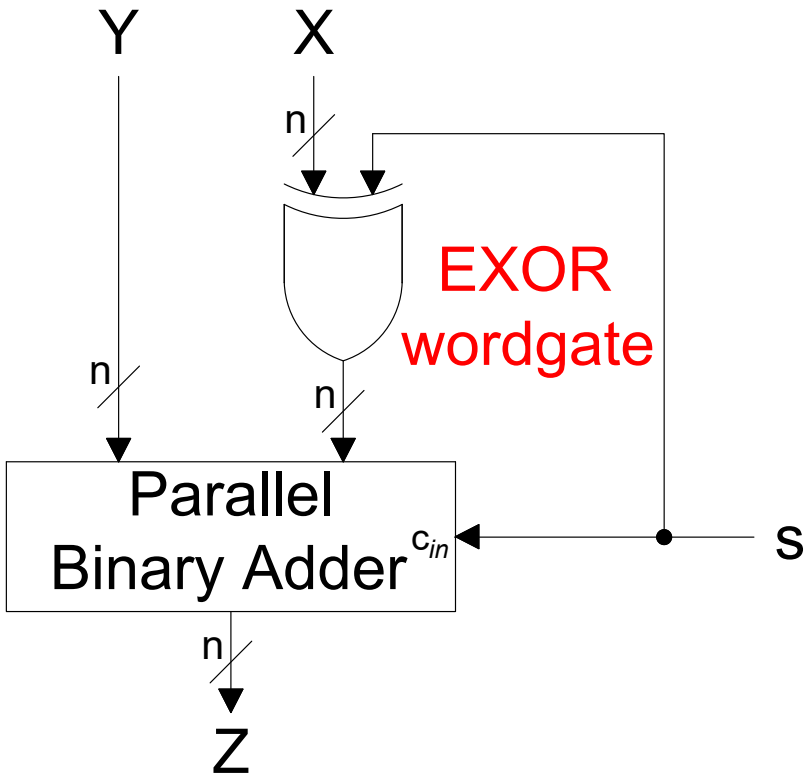
Operația de scădere:

- $X$: 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.)

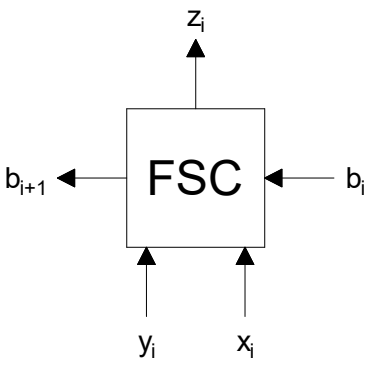
Modalități de realizare a operației de scădere

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




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

## (B) Scăzătoare dedicate

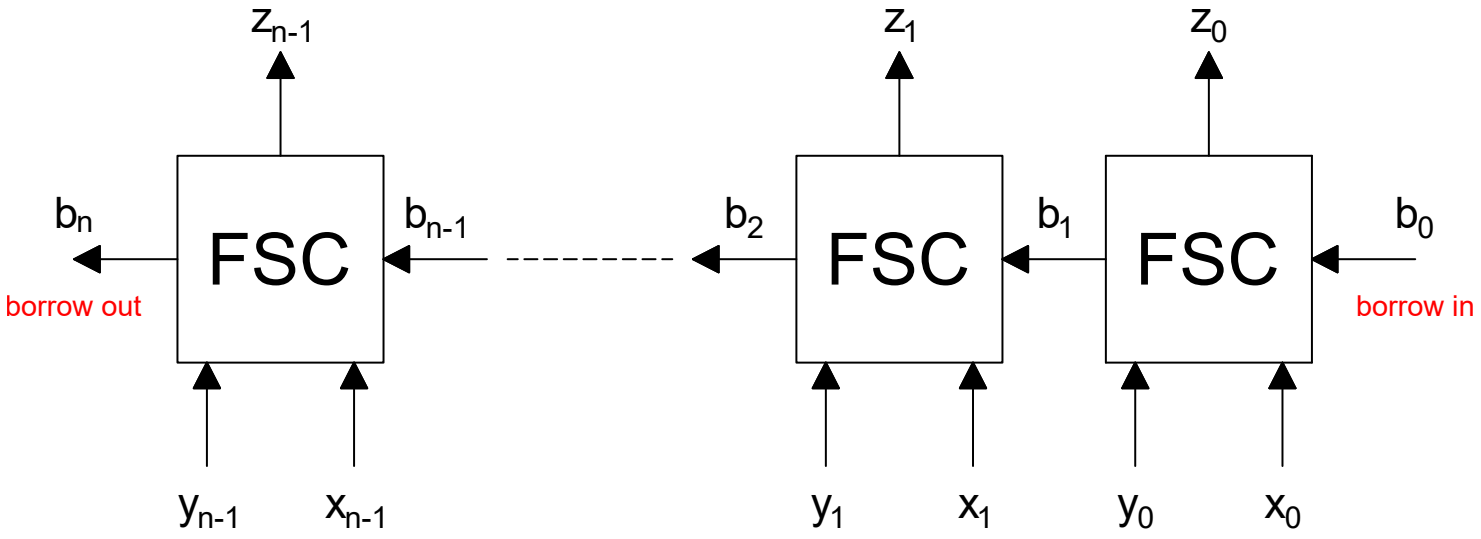



Utilizarea celulelor Full Subtracter Cells (FSCs):

- transportul este înlocuit de împrumut
- operație implementată:

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

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 |

- ecuatiile iesirilor:

$z_i$

| $y_i$ \ $x_i, b_i$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

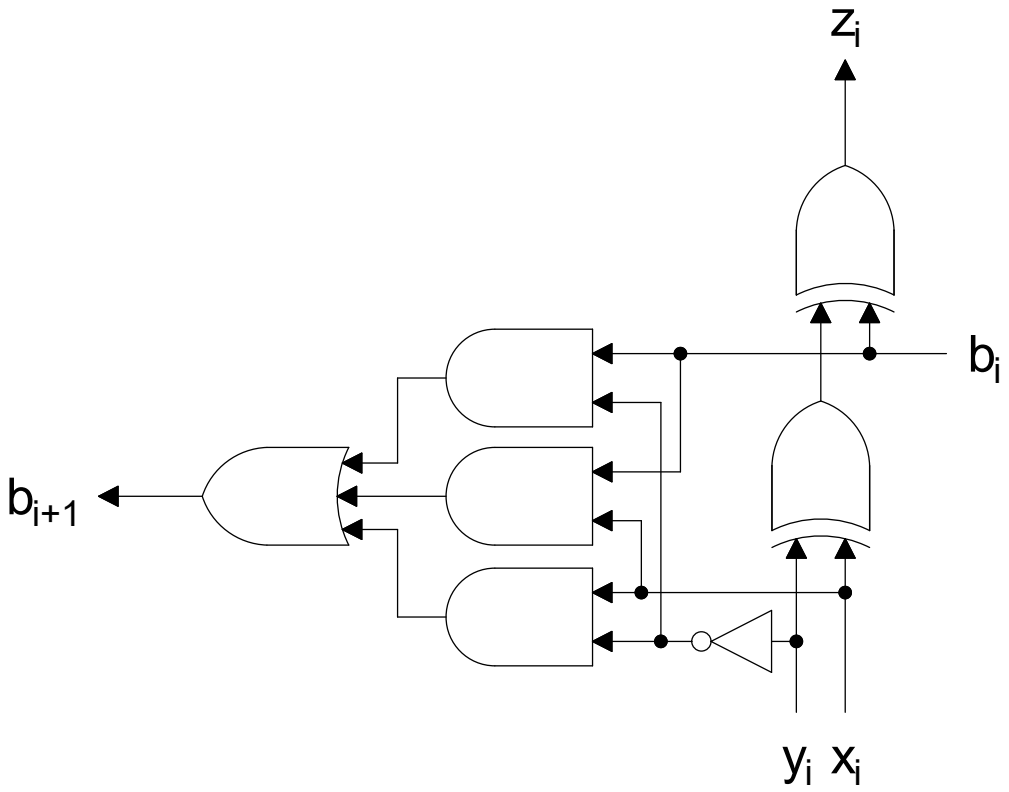
$z_i = y_i \oplus x_i \oplus b_i$

$b_{i+1}$

| $y_i$ \ $x_i, b_i$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |

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

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



## (C) 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:

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

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

$$\begin{array}{rcl}
\overline{X^\star}^{(k)} & = & \overline{X_{k-1}^\star} \quad \overline{X_{k-2}^\star} \quad \cdots \quad \overline{X_0^\star} \\
 & & \longleftarrow \text{<k> digits} \longrightarrow \\
 & = & 9 \quad 9 \quad \cdots \quad 9 \quad - \\
 & & X_{k-1} \quad X_{k-2} \quad \cdots \quad X_0 \\
\overline{X^\star}^{(k)} & = & 10^k - 1 - X^{(k)}
\end{array}$$

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

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

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

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 $X_i$: $\overline{X_i^{\star}} = x_3^{\star}x_2^{\star}x_1^{\star}x_0^{\star}$
  - cu $\overline{X_i^{\star}} = 9 - X_i$

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

Tabel de adevăr al unității pentru calcularea complementului de 9:

| Inputs | | | | Outputs | | | |
|---|---|---|---|---|---|---|---|
| $x_3$ | $x_2$ | $x_1$ | $x_0$ | $x_3^\star$ | $x_2^\star$ | $x_1^\star$ | $x_0^\star$ |
| 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:

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

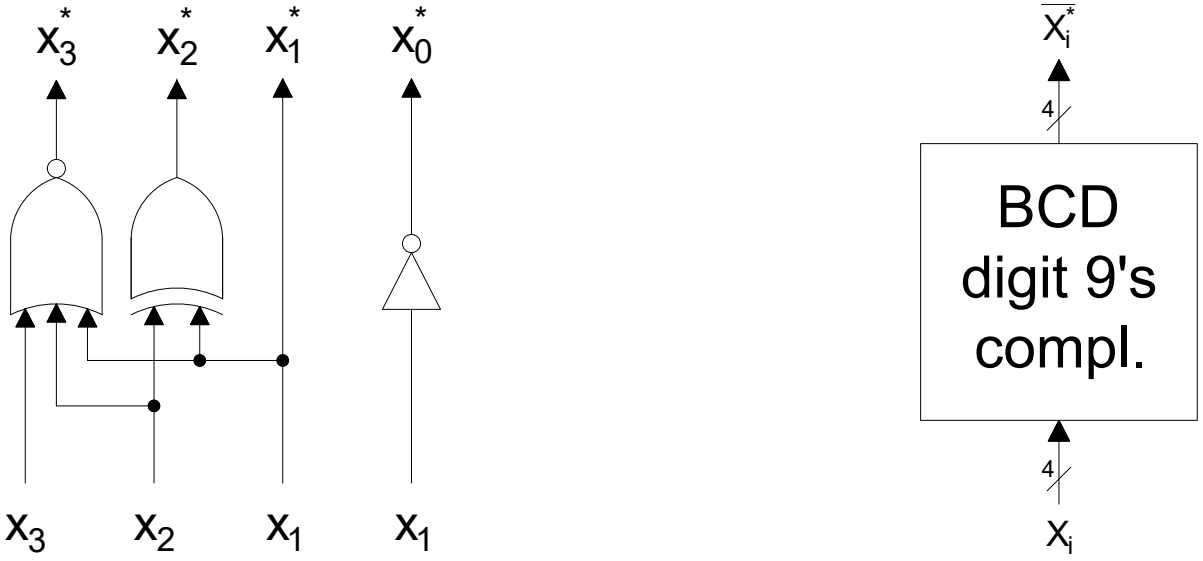
$$x_2^\star = x_2 \oplus x_1$$

$$x_1^\star = x_1$$

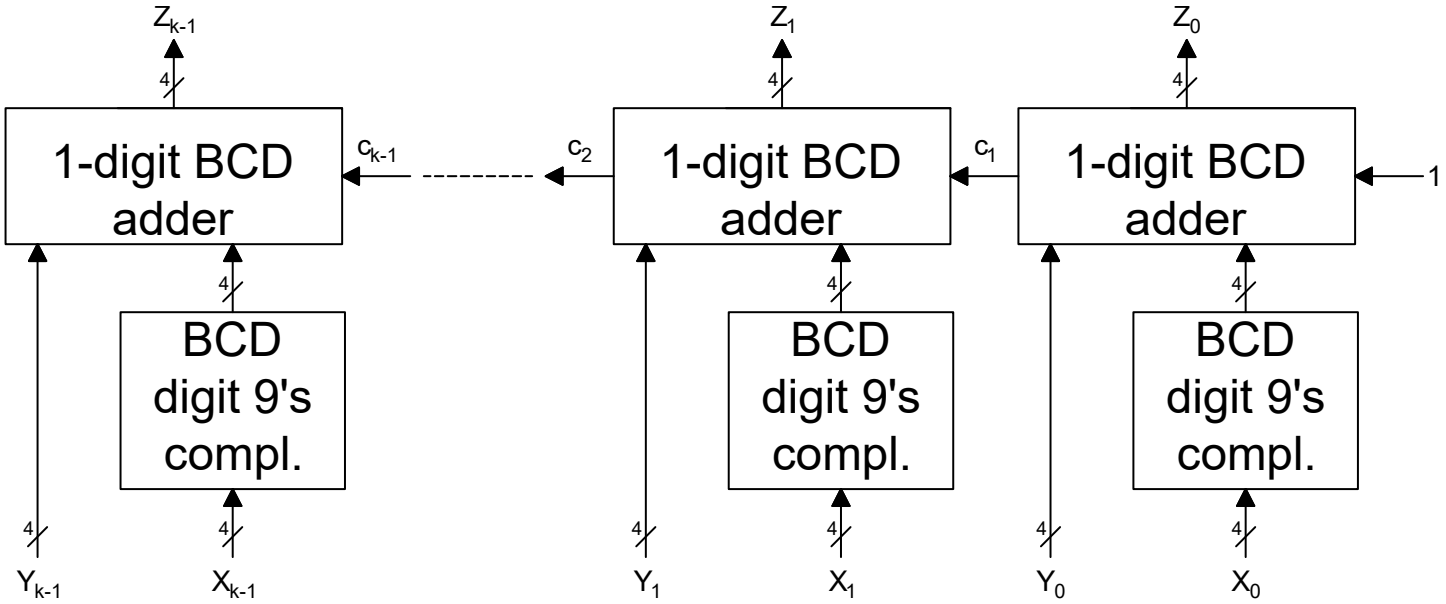
$$x_0^\star = \overline{x_0}$$

## 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:



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

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

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:

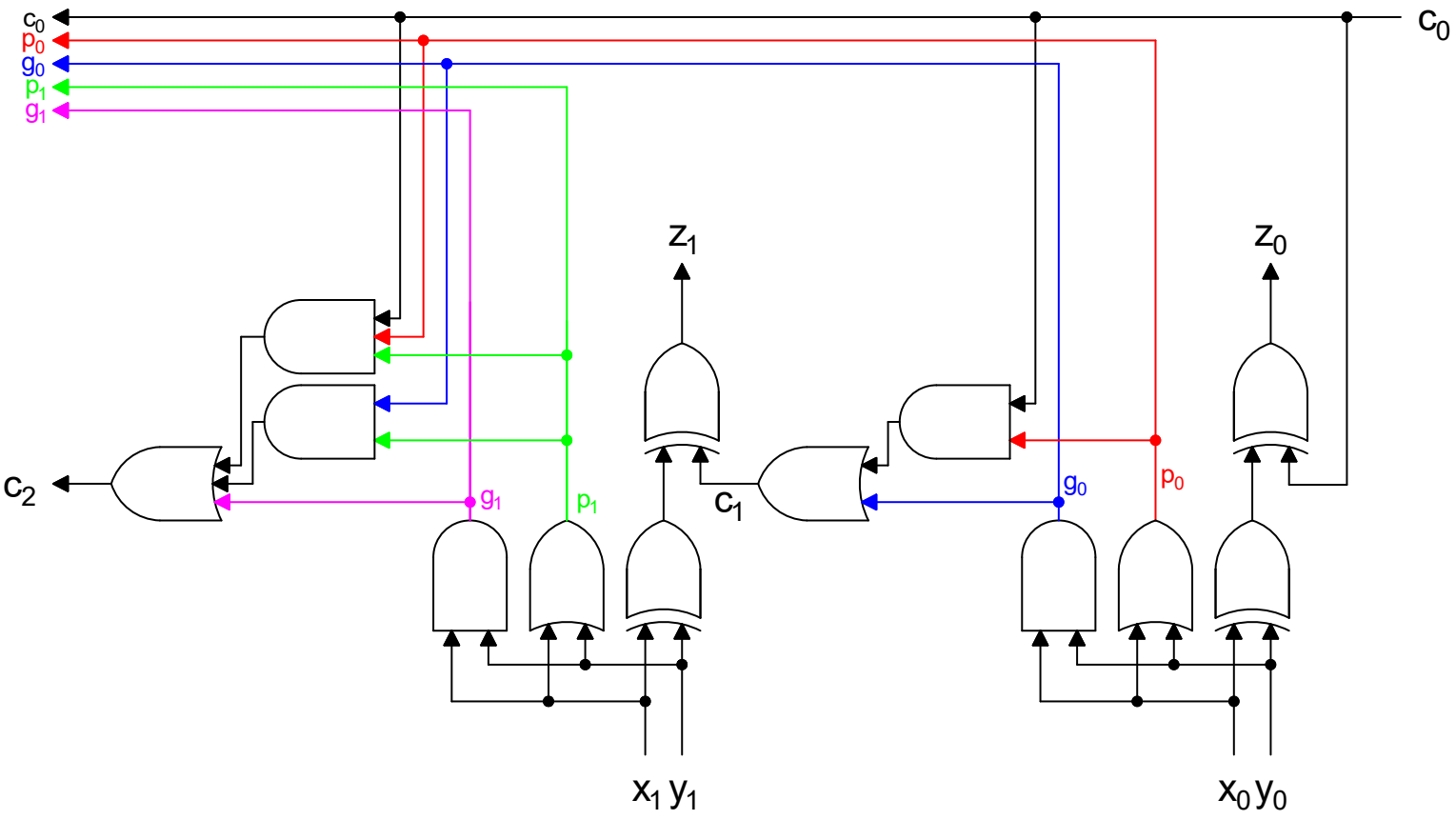
$$\begin{aligned} c_{i+1} &= g_i + p_i \cdot c_i \\ &= g_i + p_i \cdot g_{i-1} + p_i \cdot p_{i-1} \cdot c_{i-1} \\ &= \cdots \\ &= g_i + p_i \cdot g_{i-1} + \cdots + p_i \cdot p_{i-1} \cdots p_1 \cdot g_0 + p_i \cdot p_{i-1} \cdots p_0 \cdot c_0 \end{aligned}$$

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

$\Rightarrow$ Sumatoarele F-CLA operează numere de lățime redusă

## 2.3.1 Sumator Carry Lookahead (contin.)

Sumator F-CLA pe 2 biți:




Întârizerea F-CLA pe $n$ biți:

$D^{c_{out}}_{F-CLA} = 3d$

$D^{z}_{F-CLA} = 5d$

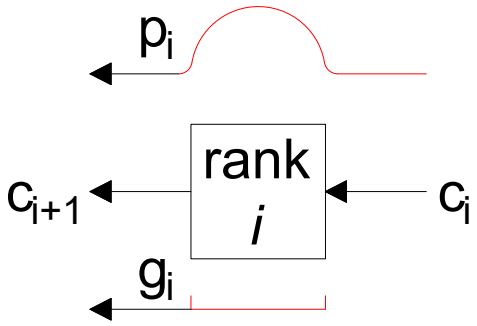
## 2.3.1 Sumator Carry Lookahead (contin.)

Ecuația $c_{i+1} = g_i + p_i \cdot c_i$, 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:



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

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

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.)

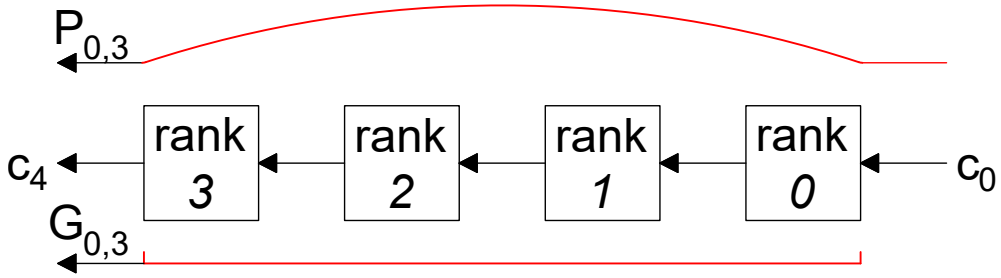
În ecuația:

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

$G_{0,3}$ 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:



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

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

Variabile de generare/propagare la nivelul unui bloc de ranguri pot fi exprimate in 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.)

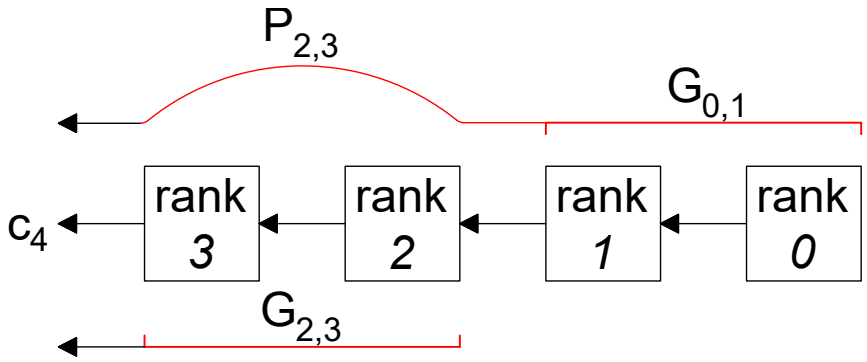
Ecuația:

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

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.)

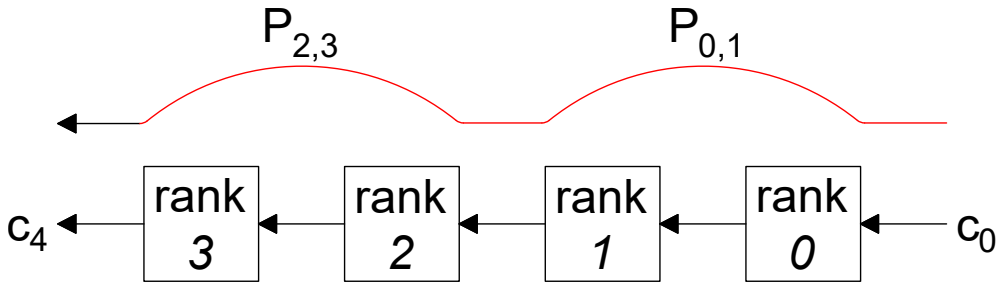
Ecuația:

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

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:

$$\begin{aligned} G_{i,i} &= g_i = x_i \cdot y_i \\ P_{i,i} &= p_i = x_i + y_i \end{aligned}$$

pentru orice rang $i$

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

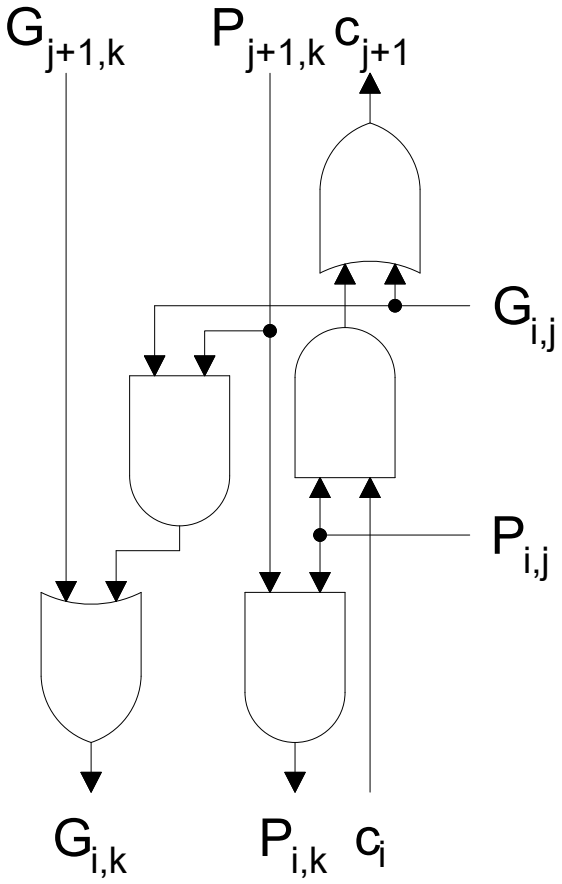
$$\begin{aligned} c_{j+1} &= G_{i,j} + P_{i,j} \cdot c_i && \forall i \leq j \\ G_{i,k} &= G_{j+1,k} + P_{j+1,k} \cdot G_{i,j} && \forall i \leq j < k \\ P_{i,k} &= P_{j+1,k} \cdot P_{i,j} && \forall i \leq j < k \end{aligned}$$

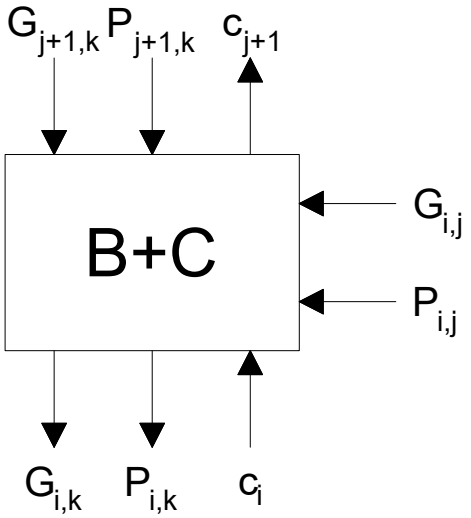
## 2.3.1 Sumator Carry Lookahead (contin.)

Toate cele 3 ecuații

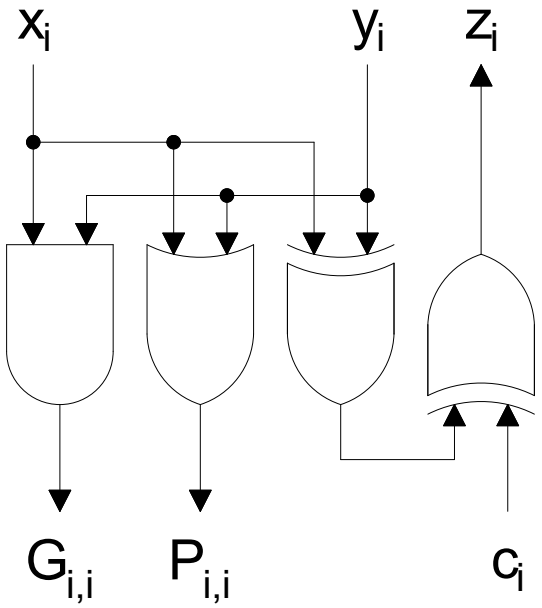
$$
\begin{aligned}
c_{j+1} &= G_{i,j} + P_{i,j} \cdot c_i \\
G_{i,k} &= G_{j+1,k} + P_{j+1,k} \cdot G_{i,j} \\
P_{i,k} &= P_{j+1,k} \cdot P_{i,j}
\end{aligned}
$$

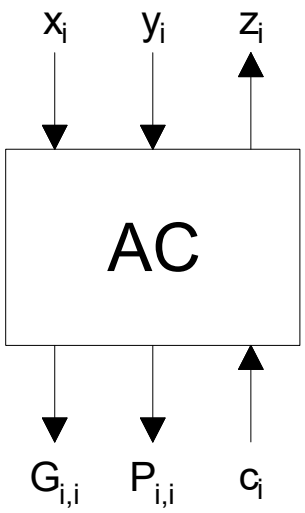
sunt implementate de celulele de tip $B + C$ descrise mai jos:



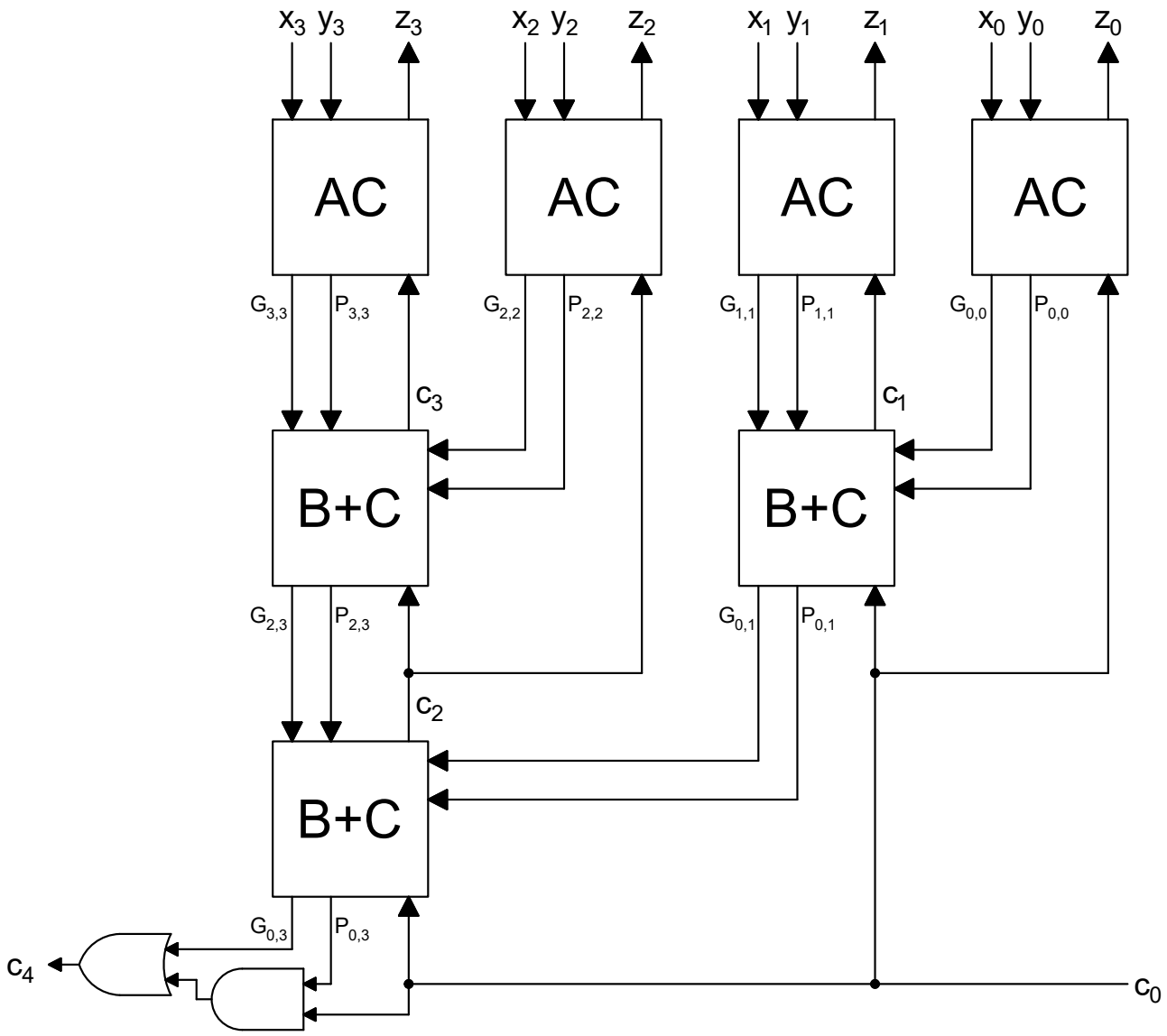
Semnalelor $G_{i,i}$ și $P_{i,i}$ (pe baza cărora se vor construi semnalele de generare/propagare la nivel de bloc) împreună cu biții sumei, $z_i$, vor fi generați de celulele de tip $AC$ de mai jos:



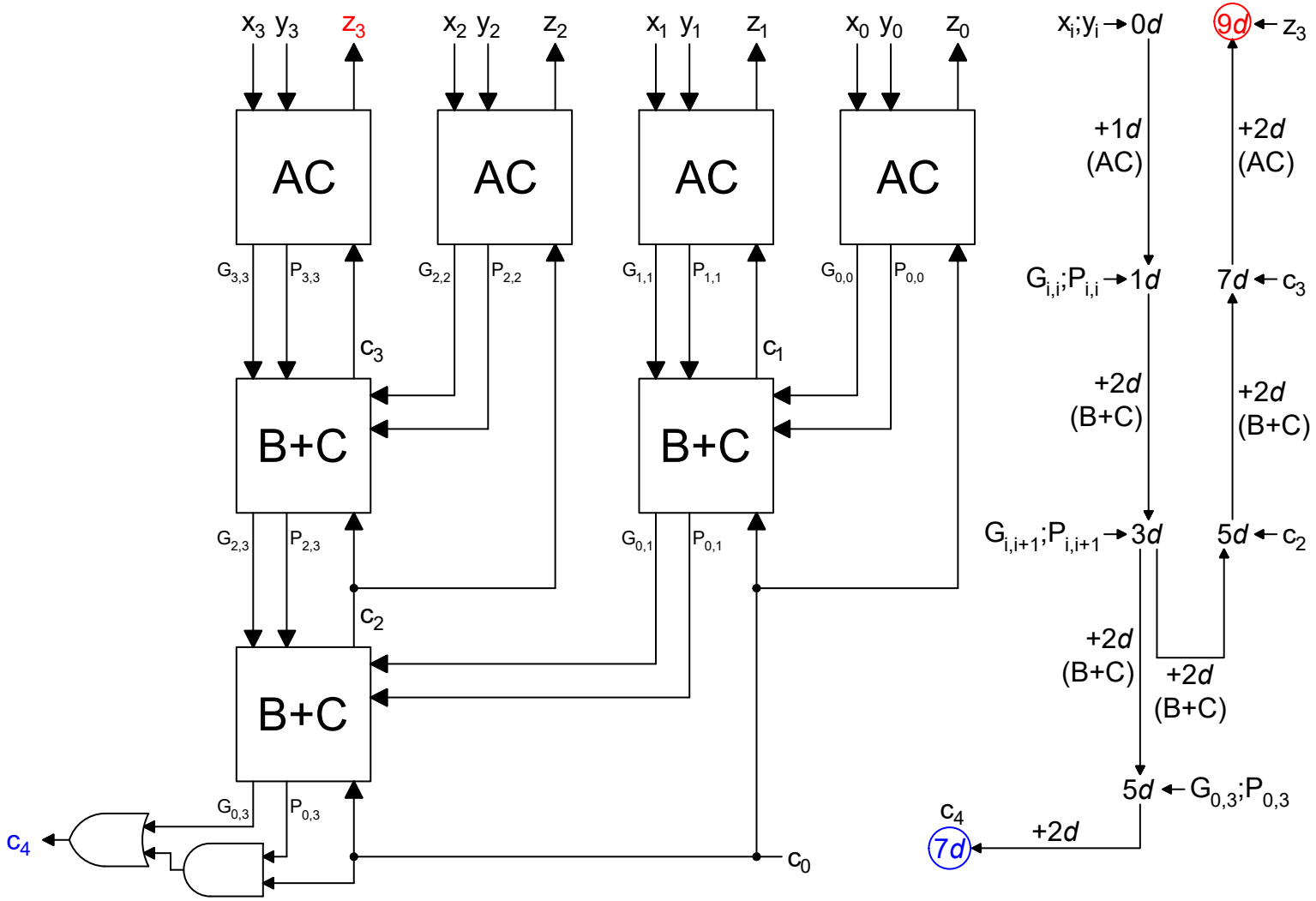
## 2.3.1 Sumator Carry Lookahead (contin.)

Arhitectura CLA multinivel pe 4 biți:



## 2.3.1 Sumator Carry Lookahead (contin.)

Determinarea latenței unui CLA multinivel pe 4 biți:




$$D^{c_{out}}_{ML-CLA-4} = 7d$$
$$D^{z}_{ML-CLA-4} = 9d$$

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:
$D^{c_{out}}_{ML-CLA-n} = (2\lceil log_2 n \rceil + 3)d$
$D^{z}_{ML-CLA-n} = (4\lceil log_2 n \rceil + 1)d$

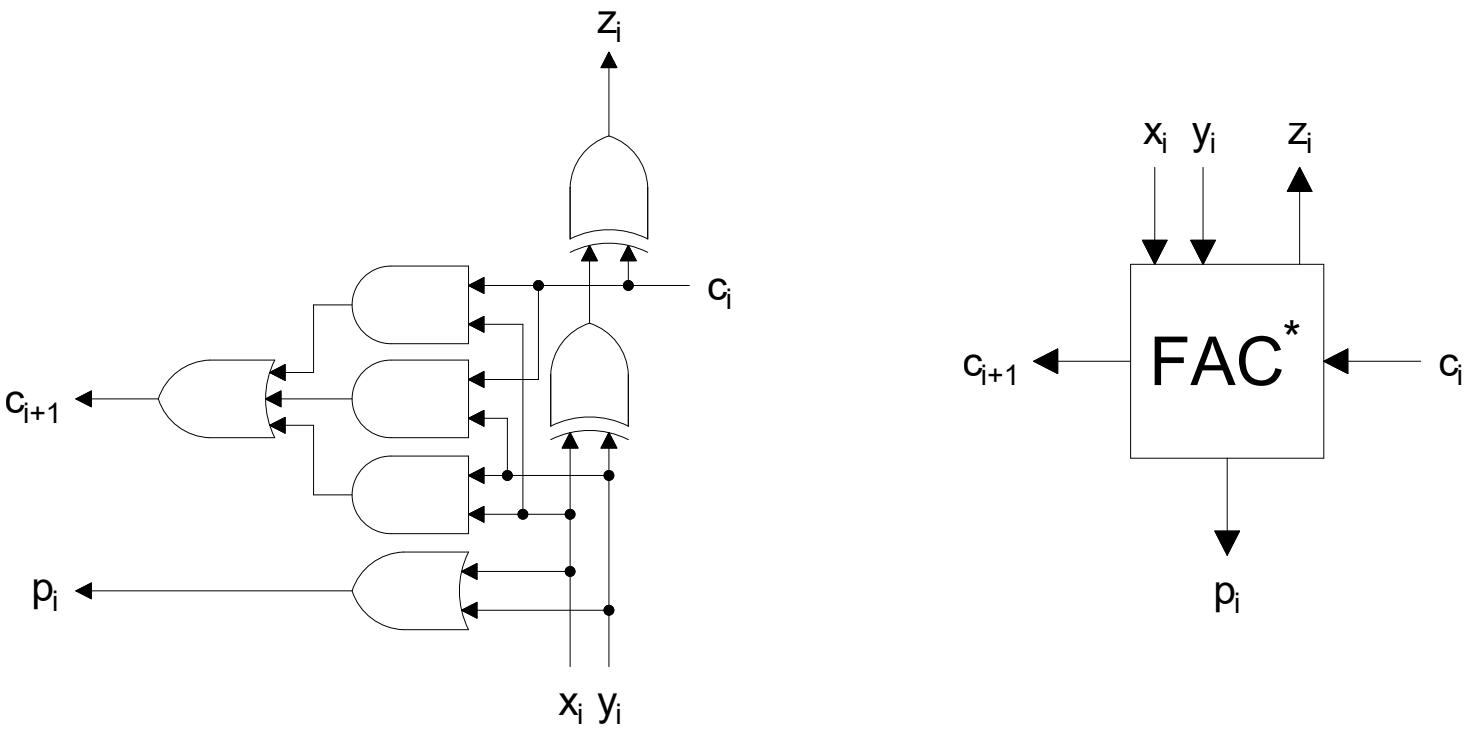
## 2.3.2 Sumator Carry Skip

Din ecuația de generare a transportului:

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

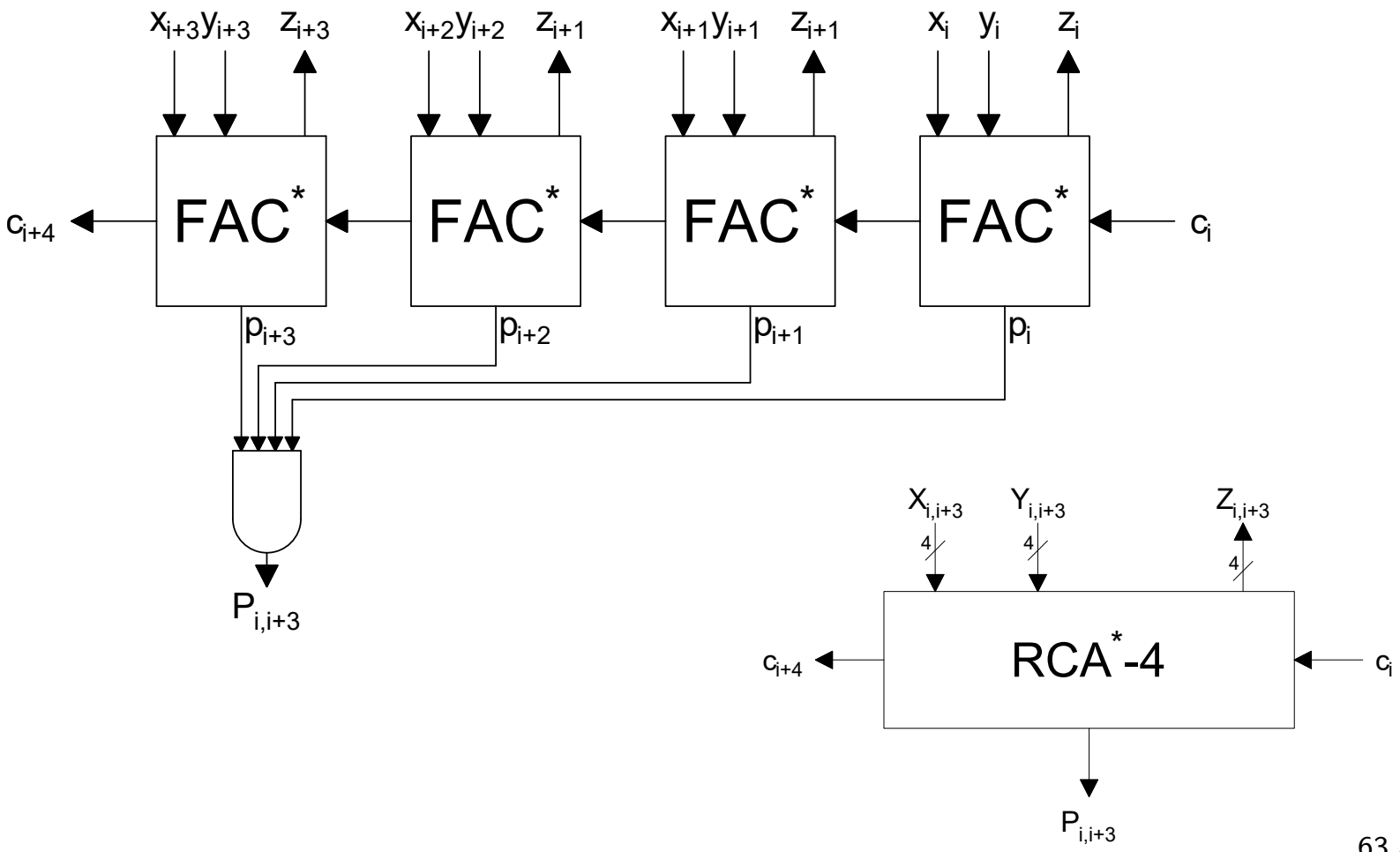
variabila de propagare la nivel de bloc, $P_{i,j}$, se obține mai simplu.

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*$^\star$, 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:



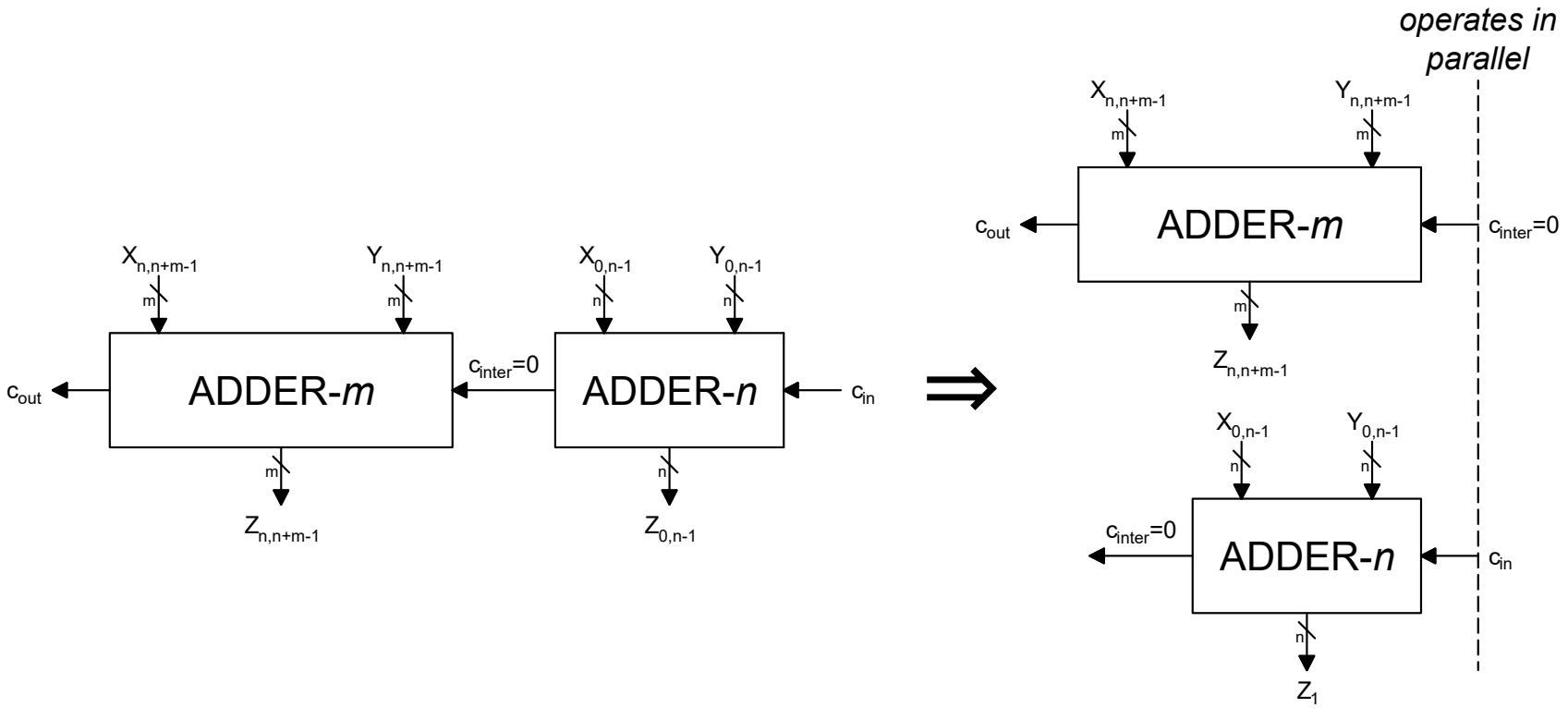
## 2.3.2 Sumator Carry Skip (contin.)

CMOS pre-discharging:

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

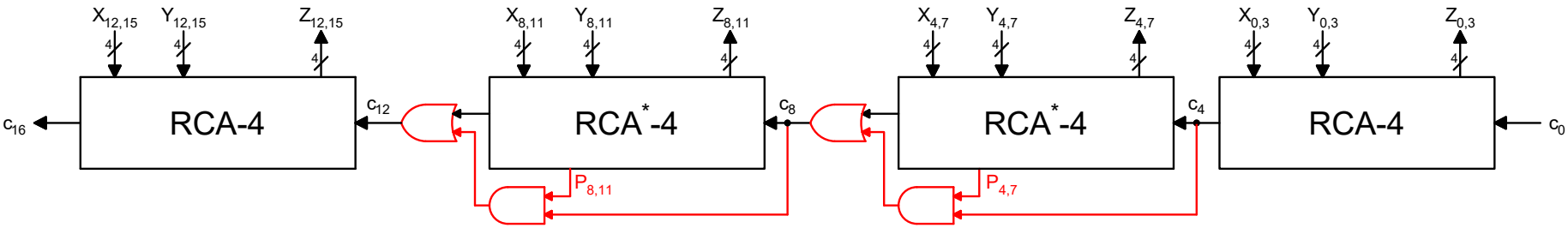
Pentru sumatorul Carry Skip (CSkA):

- 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.



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

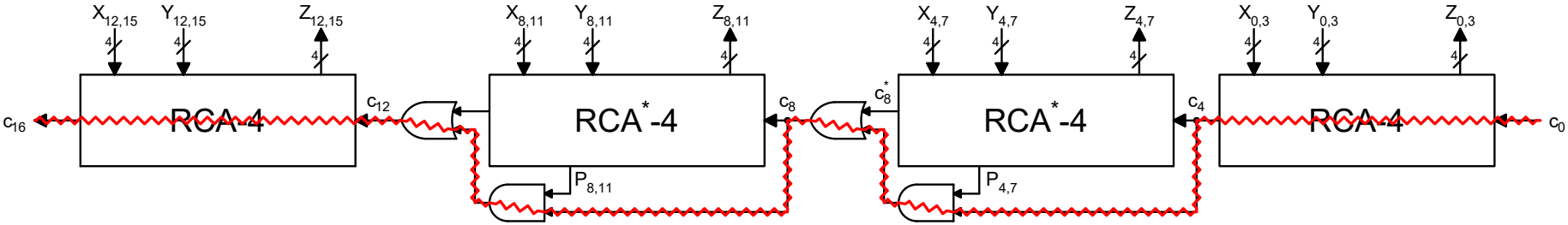
- pentru blocuri de tip *RCA*$^\star$ este utilizată logica de șunt ("skip logic", evidențiată prin culoare) pentru a facilita propagarea transportului peste întregul bloc




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 biți:




**Teorema**: de ce folosește calea critică logica de șunt?
**Demostrație**: 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:

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

## 2.3.2 Sumator Carry Skip (contin.)

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

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:

$$
\begin{aligned}
&P_{4,7} = 0 \Rightarrow p_4 \cdot p_5 \cdot p_6 \cdot p_7 = 0 \\
&\quad \Rightarrow \exists \text{ cel puțin un } p_i, \text{ cu } i \in [4,7], \text{ pentru care } p_i = 0 \\
&\qquad \Rightarrow \text{pentru acel indice } i \text{ se poate scrie : } x_i + y_i = 0 \Rightarrow x_i = y_i = 0 \\
&\qquad\qquad \Rightarrow c_{i+1} = 0
\end{aligned}
$$

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 $P_{4,7} = 1$.

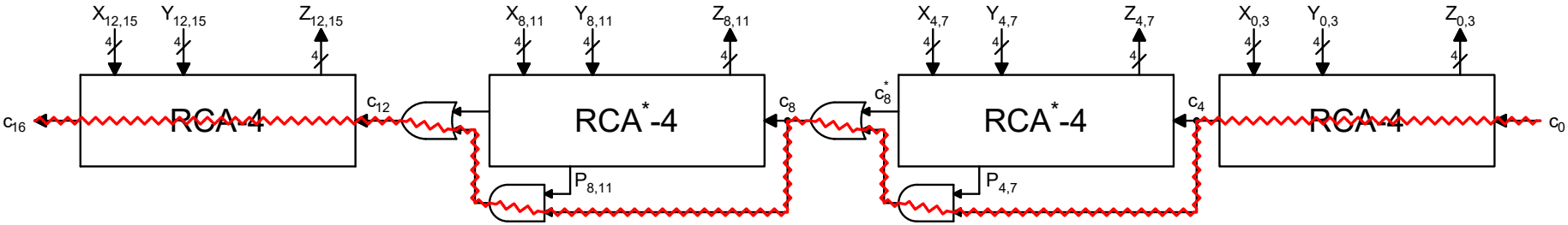
Urmarea acestei observații, evidentă din expresia lui $c_8$:

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

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:




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

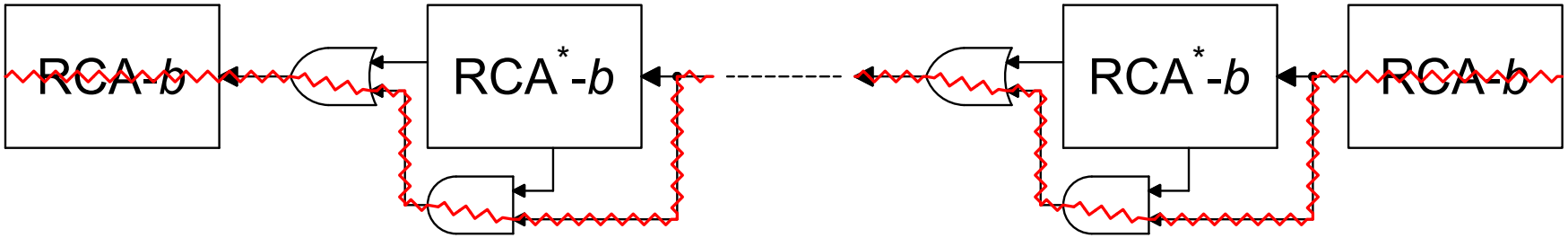
$$D^{c_{out}}_{CSkA-16} = \underbrace{8d}_{c_4} + \underbrace{2d}_{c_8} + \underbrace{2d}_{c_{12}} + \underbrace{8d}_{c_{16}} = 20d$$

## 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:

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

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

$$\frac{\partial D_{CSkA-n}^{z/c_{cout}}}{\partial b_{opt}} = 0 \Rightarrow -\frac{2n}{b^2} + 4 = 0$$

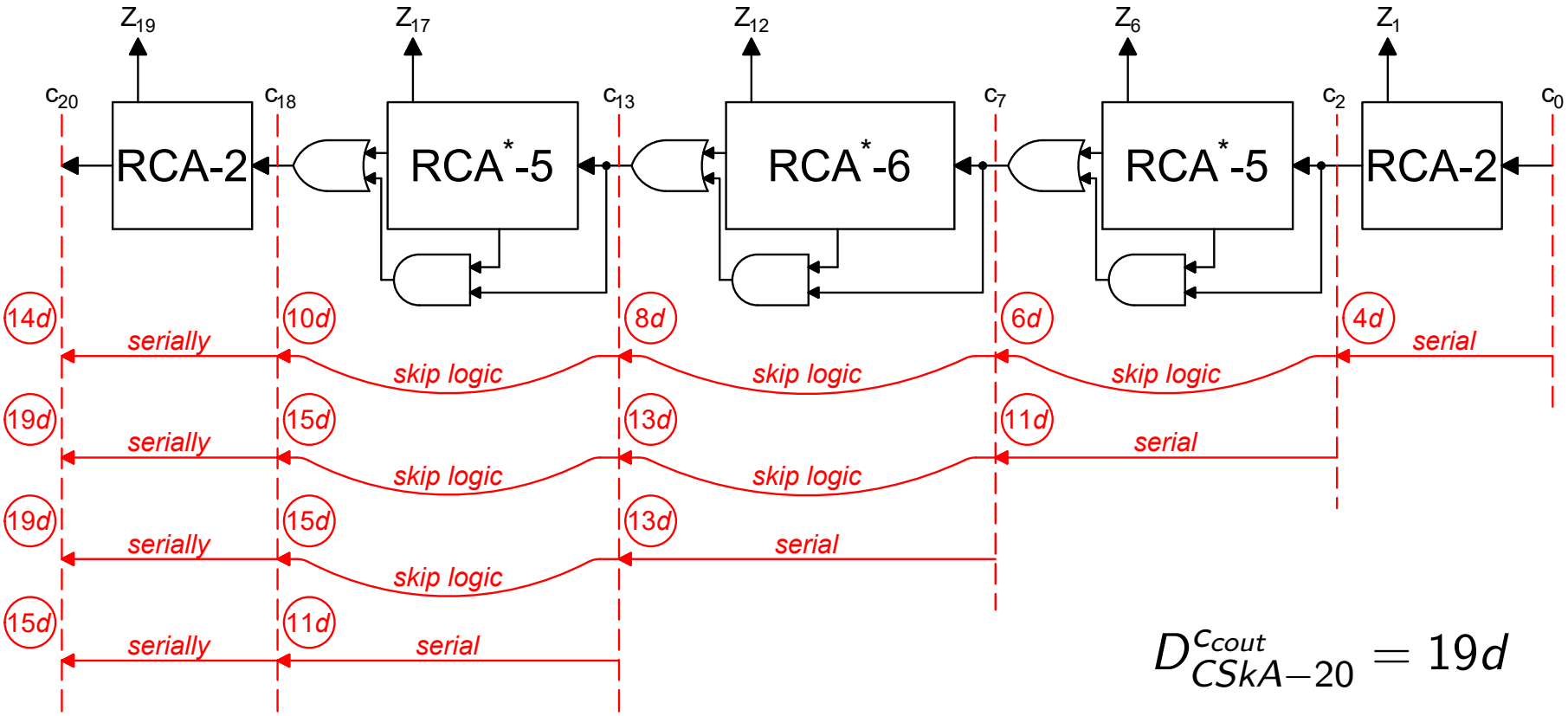
$$\Rightarrow b_{opt} = \frac{\sqrt{2n}}{2}, \text{ cu latența optimă : } D_{CSkA-n_{opt}}^{z/c_{cout}} = 4(\sqrt{2n} - 1)d$$

Exemplu: se consideră $n = 32$

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

## Segmente RCA de lungimi variabile

Se consideră operanzi pe 20 biți și structura CSkA următoare:




$$D^{c_{cout}}_{CSkA-20} = 19d$$

Pentru transportul de ieșire, cazul defavorabil începe propagarea transportului din segmentul $RCA^{\star} - 5$ 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$^\star$ ($z_{19}$, $z_{17}$, $z_{12}$, $z_6$, $z_1$).

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

$$
\begin{aligned}
D^{z_6}_{max} &= D^{c_2}_{max} + \underbrace{2 * 5d}_{D^z_{RCA^\star - 5}} &= 4d + 10d &= 14d \\
D^{z_{12}}_{max} &= D^{c_7}_{max} + 2 * 6d &= 11d + 12d &= 23d \\
D^{z_{17}}_{max} &= D^{c_{13}}_{max} + 2 * 5d &= 13d + 10d &= 23d \\
D^{z_{19}}_{max} &= D^{c_{18}}_{max} + 2 * 2d &= 15d + 4d &= 19d
\end{aligned}
$$

În concluzie:

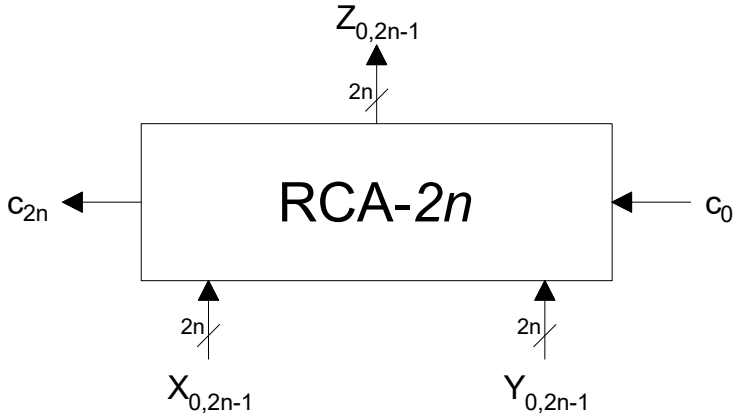
$$D^z_{CSkA-20} = 23d$$

## 2.3.3 Carry Select Adder

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

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

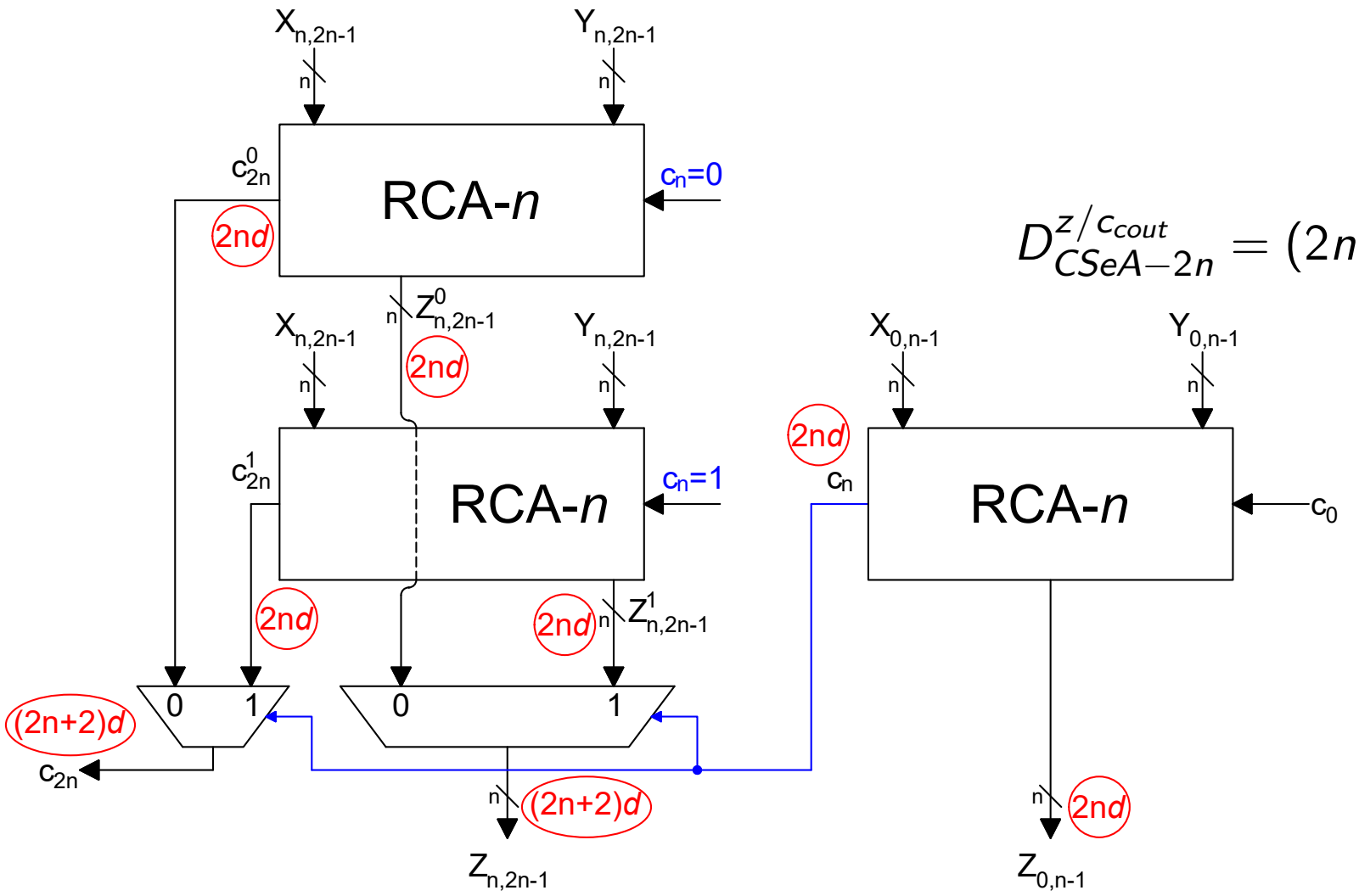
Construcția unui sumator Carry Select (CSeA), pornește de la arhitectura unui RCA pe $2n$ biți:



## 2.3.3 Carry Select Adder (contin.)

Construcția CSeA:

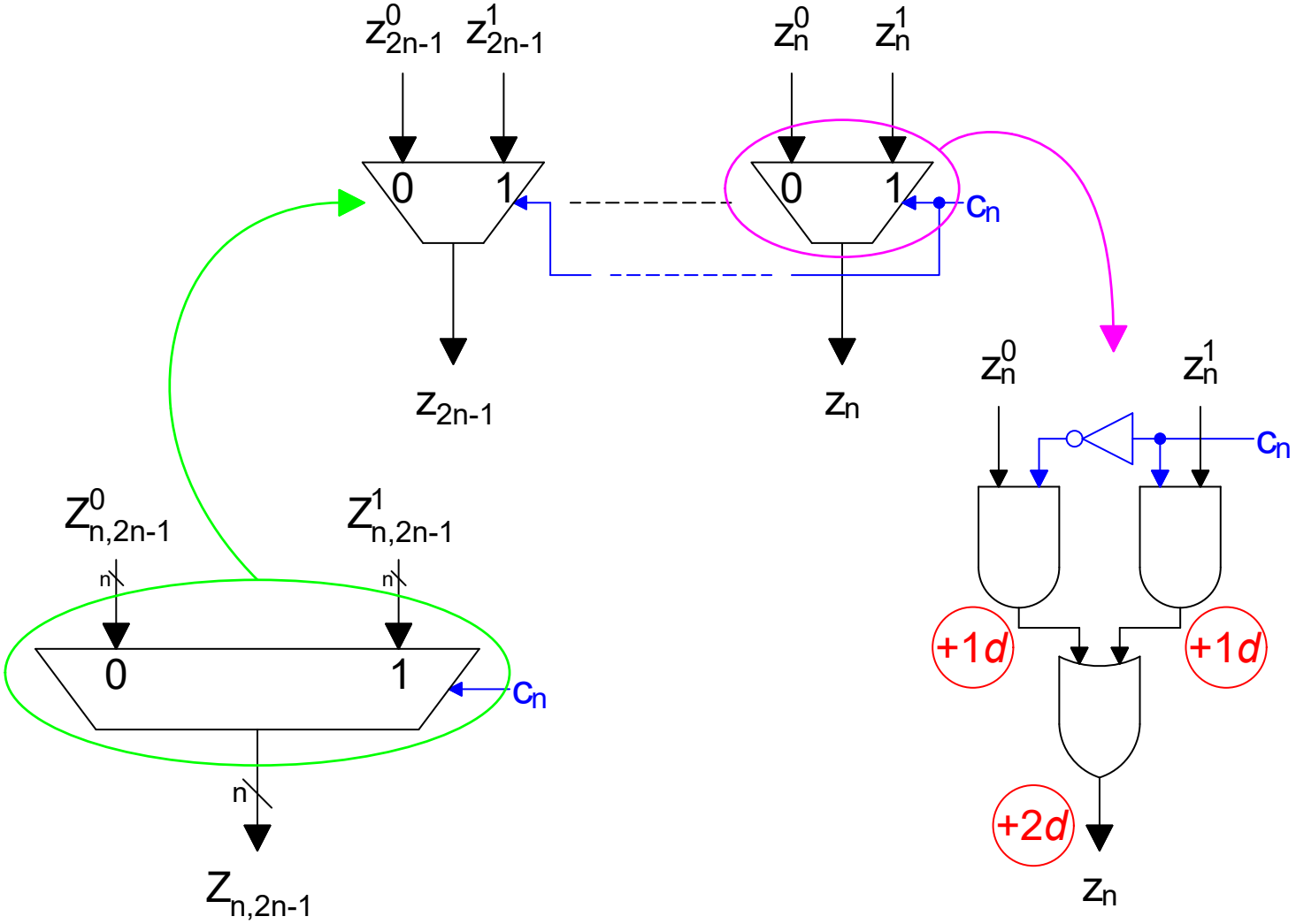
- se împarte sumatorul RCA$-2n$ în două jumătăți egale
- se duplică jumătatea mai semnificativă
  - o copie are $c_n = 0$, celalată are $c_n = 1$




$$D^{z/c_{cout}}_{CSeA-2n} = (2n+2)d$$

## 2.3.3 Carry Select Adder (contin.)

Detalii întârziere multiplexor:



## 2.3.3 Carry Select Adder (contin.)

Optimizarea ariei CSeA:

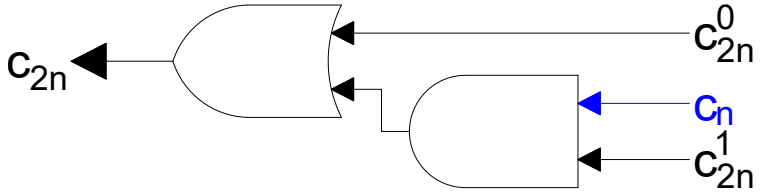
- se pornește de la tabelul de adevăr al semnalului $c_{2n}$
- **Niciodată** $c_{2n}^0$ nu poate fi mai mare decât $c_{2n}^1$!
  - **Întrebare**: de ce?
  - $\Rightarrow$ tabelul de adevăr folosește *don't care* pentru cazurile imposibile

| Inputs | | | Output |
|---|---|---|---|
| $c_n$ | $c_{2n}^0$ | $c_{2n}^1$ | $c_{2n}$ |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | $d$ |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | $d$ |
| 1 | 1 | 1 | 1 |

| $c_{2n}$ \ $c_n$ | 00 ($c_{2n}^0$, $c_{2n}^1$) | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | d |
| 1 | 0 | 1 | 1 | d |

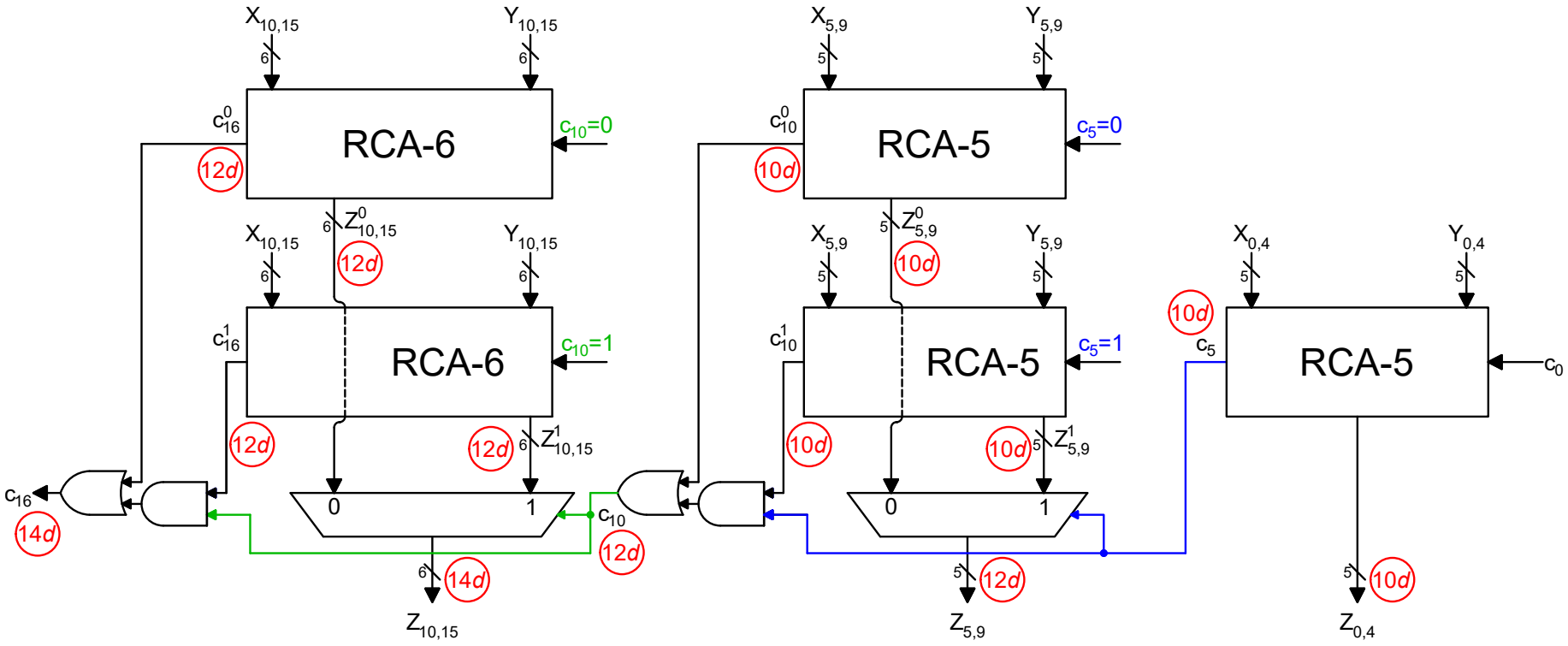
$$c_{2n} = c_{2n}^0 + c_{2n}^1 \cdot c_n$$

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



## 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 $c_0 = 0$:

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

## 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:

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

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

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

## 2.3.4 Conditional Sum Adder (contin.)

Sum bits

$$z_0 = x_0 \oplus y_0 \oplus c_0 = x_0 \oplus y_0$$
$$z_1 = x_1 \oplus y_1 \oplus c_1 = \overline{x_1 \oplus y_1} \cdot c_1 + (x_1 \oplus y_1) \cdot \overline{c_1}$$
$$z_2 = x_2 \oplus y_2 \oplus c_2 = \overline{x_2 \oplus y_2} \cdot c_2 + (x_2 \oplus y_2) \cdot \overline{c_2}$$
$$\begin{aligned} z_3 &= x_3 \oplus y_3 \oplus c_3 = \overline{x_3 \oplus y_3} \cdot c_3 + (x_3 \oplus y_3) \cdot \overline{c_3} \\ &= \overline{x_3 \oplus y_3} \cdot (g_2 \cdot \overline{c_2} + p_2 \cdot c_2) + \\ &\quad (x_3 \oplus y_3) \cdot (\overline{g_2} \cdot \overline{c_2} + \overline{p_2} \cdot c_2) \\ &= (\overline{x_3 \oplus y_3} \cdot g_2 + (x_3 \oplus y_3) \cdot \overline{g_2}) \cdot \overline{c_2} + \\ &\quad (\overline{x_3 \oplus y_3} \cdot p_2 + (x_3 \oplus y_3) \cdot \overline{p_2}) \cdot c_2 \end{aligned}$$

Carry bits

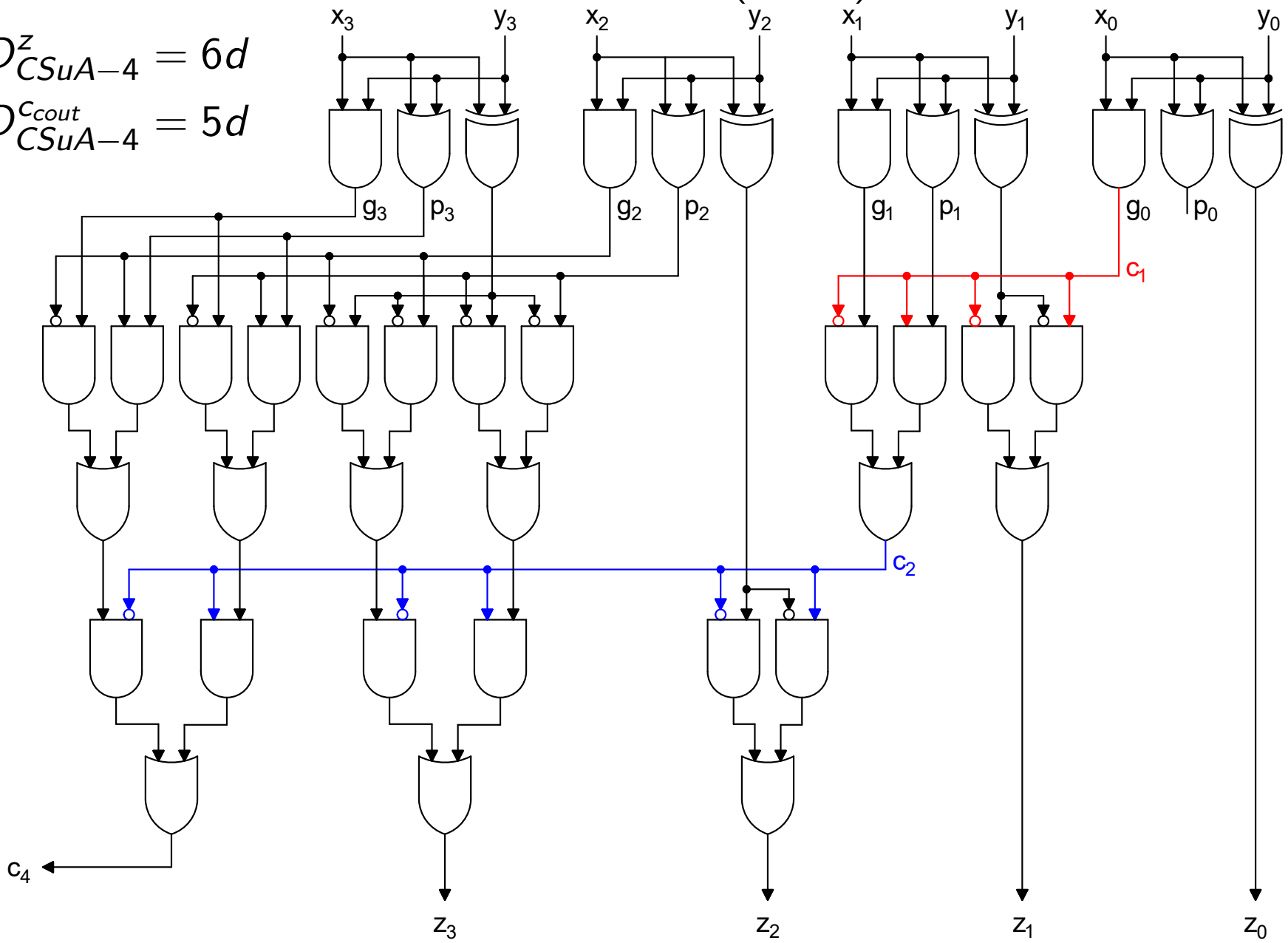
$$c_1 = g_0 \cdot \overline{c_0} + p_0 \cdot c_0 = g_0$$
$$c_2 = g_1 \cdot \overline{c_1} + p_1 \cdot c_1$$
$$c_3 = g_2 \cdot \overline{c_2} + p_2 \cdot c_2$$
$$\overline{c_3} = \overline{g_2} \cdot \overline{c_2} + \overline{p_2} \cdot c_2$$
$$\begin{aligned} c_4 &= g_3 \cdot \overline{c_3} + p_3 \cdot c_3 \\ &= g_3 \cdot (\overline{g_2} \cdot \overline{c_2} + \overline{p_2} \cdot c_2) + \\ &\quad p_3 \cdot (g_2 \cdot \overline{c_2} + p_2 \cdot c_2) \\ &= (g_3 \cdot \overline{g_2} + p_3 \cdot g_2) \cdot \overline{c_2} + \\ &\quad (g_3 \cdot \overline{p_2} + p_3 \cdot p_2) \cdot c_2) \end{aligned}$$

## 2.3.4 Conditional Sum Adder (contin.)

Arhitectura Conditional Sum Adder (CSuA):

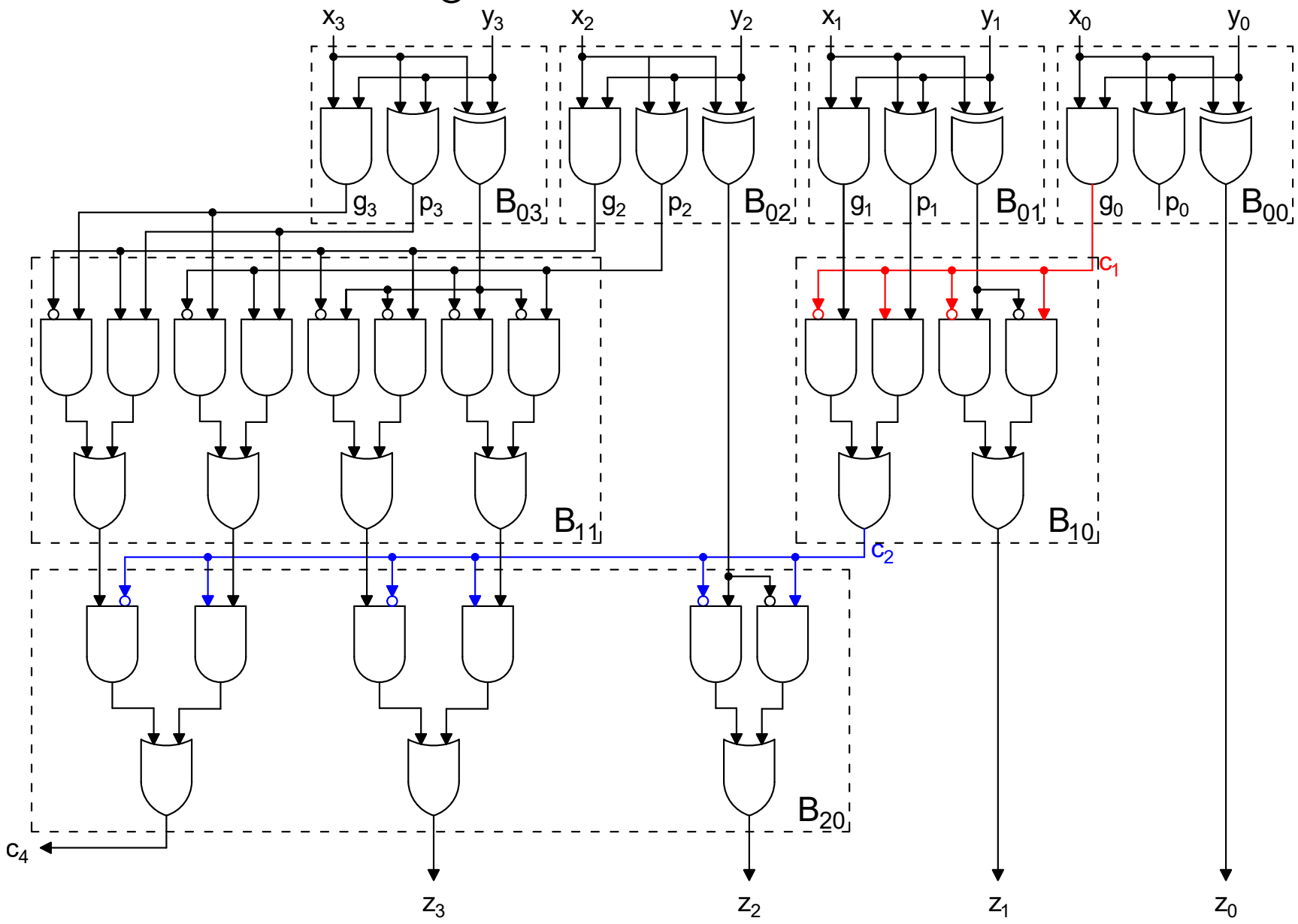
$$D^{z}_{CSuA-4} = 6d$$

$$D^{c_{cout}}_{CSuA-4} = 5d$$



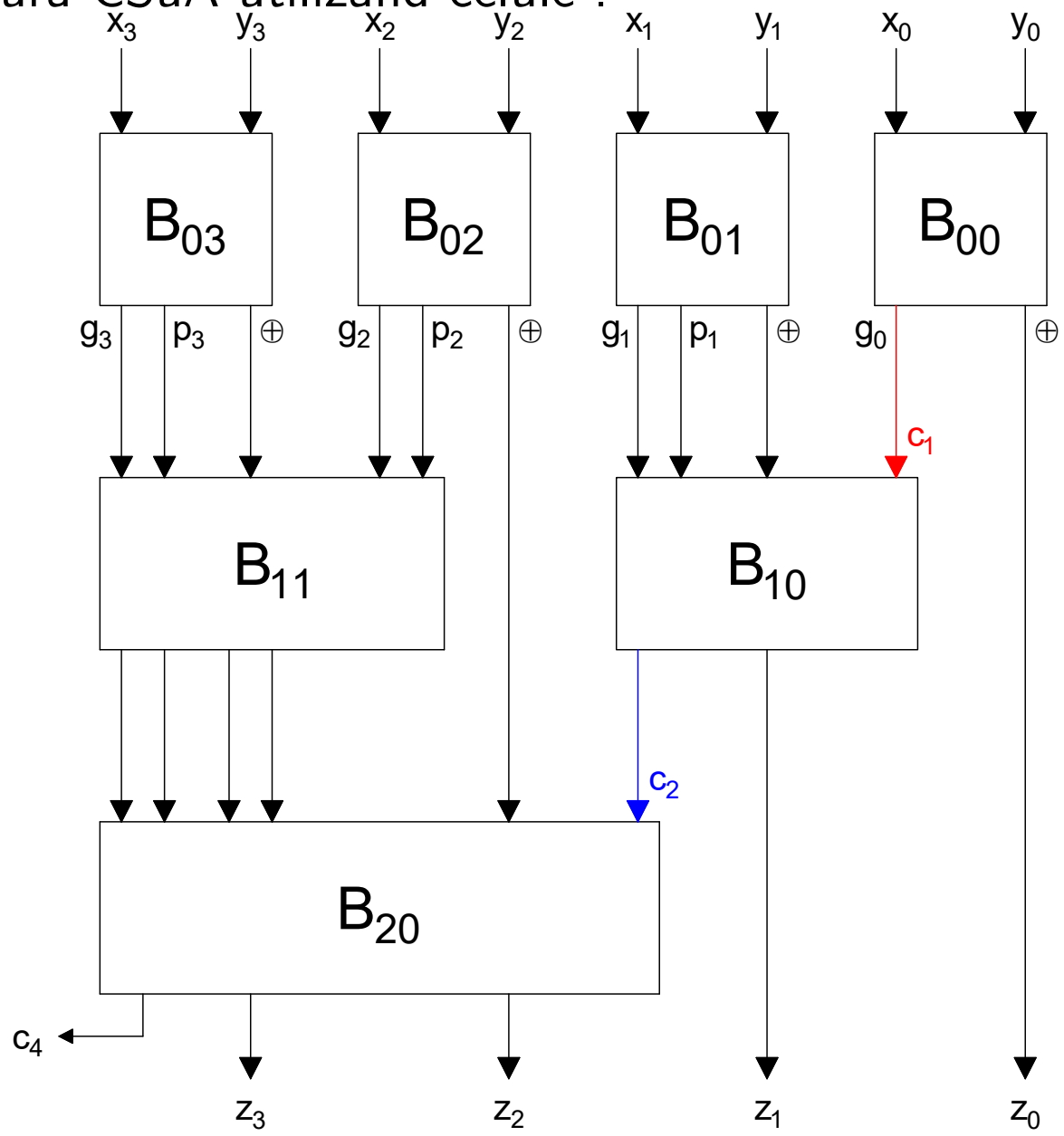
## 2.3.4 Conditional Sum Adder (contin.)

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, exista $\lceil log_2 n \rceil$ alte nivele în arhitectura CSuA. Aceste nivele coerspund celulelor $B_{ij}$, cu $j >= 1$ si sunt similare nivelelor $B + C$ din structura CLA multi-nivel.

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

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

## 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: Însumați operanzii pe 8 biți $X$ și $Y$ după metoda CSuA:

$$\begin{aligned} X = &\quad 10001101 \quad (= 128 + 13 = 141) \\ Y = &\quad 01110101 \quad (= 64 + 32 + 16 + 5 = 117) \end{aligned}$$

## 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

| Operand \ Rank | | 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 | $c_{in}$=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 | | | | | | | | | | | | | | | | |
| | $c_{in}$=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.

| Operand \ Rank | | 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 | $c_{in}$=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 | | | | | | | | |
| | $c_{in}$=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.

| Operand \ Rank | | 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 | $c_{in}$=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 | | | | | | | | |
| | $c_{in}$=1 | | | | | | | | |

## 2.3.4 Conditional Sum Adder (contin.)

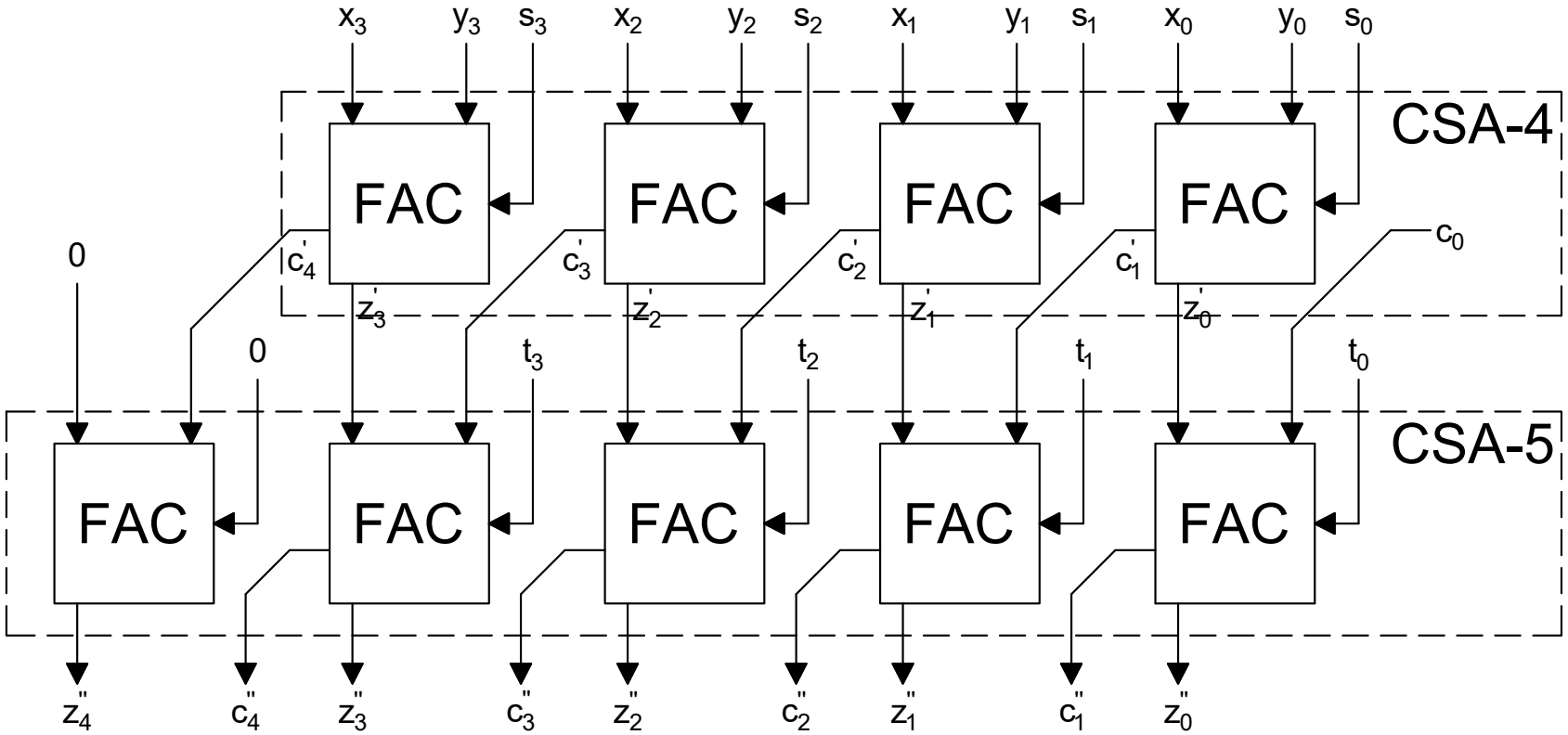
Nivelul al patrulea: determinarea sumei corecte pentru întreg setul de ranguri al celor 2 operanzi.

| Operand \ Rank | | 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 | $c_{in}$=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 → sumă, transport
  - vectorul transport este cu o poziție mai semnificativ decât cel sumă
- permite realizarea adunării multi-operand

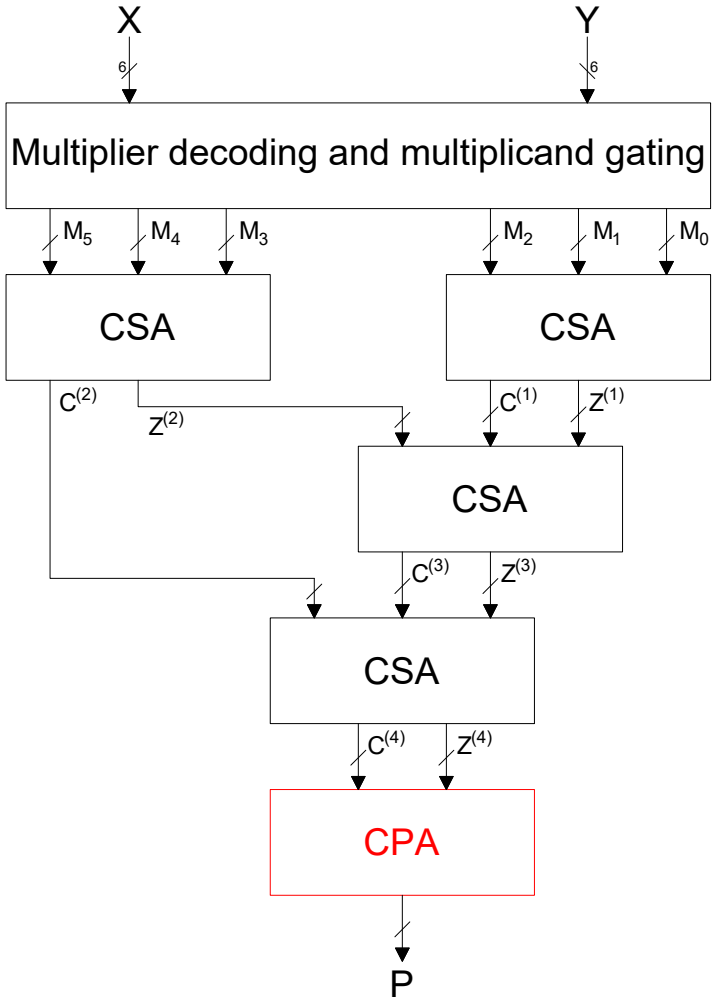
Se consideră operanzii $X$, $Y$, $S$ și $T$, pe 4 biți. Suma $Z = X + Y + S + T$ poate fi calculata astfel:



## 2.3.5 Carry Save Adder (contin.)

- facilitează realizarea operației de înmulțire (combinațional)

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.)

Exemplu:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | $x_3$ | $x_2$ | $x_1$ | $x_0$ |
| $X=15$ | | | | | | 1 | 1 | 1 | 1 |
| $Y=13$ | | | | | | 1 | 1 | 0 | 1 |
| $M_0=x_0Y2^0$ | | | | | | 1 | 1 | 0 | 1 |
| $M_1=x_1Y2^1$ | | | | | 1 | 1 | 0 | 1 | 0 |
| $M_2=x_2Y2^2$ | | | | 1 | 1 | 0 | 1 | 0 | 0 |
| $Z^{(1)}$ | | | | 1 | 0 | 0 | 0 | 1 | 1 |
| $C^{(1)}$ | | | | 0 | 1 | 1 | 1 | 0 | 0 |
| $2C^{(1)}$ | | | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| $Z^{(1)}$ | | | | 1 | 0 | 0 | 0 | 1 | 1 |
| $M_3=x_3Y2^3$ | | 1 | 1 | 0 | 1 | 0 | 0 | 0 | |
| $Z^{(2)}$ | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | |
| $C^{(2)}$ | | 0 | 1 | 0 | 1 | 0 | 0 | 0 | |
| $2C^{(2)}$ | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | |
| $Z^{(2)}$ | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | |
| $P$ | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | $=195$ |

$+_{CSA}$ $+_{CSA}$ $+_{CPA}$

Atributele sistemelor de calcul:

- performanță
- consum de energie
- fiabilitate

Atributele calculului fiabil:

| | |
|---|---|
| *disponibilitate* | sistemul este pregătit să își ofere serviciile |
| *fiabilitate* | sistemul își oferă serviciile într-un mod continuu |
| *mentenabilitate* | abilitatea de a efectua reparații și modificări |

## 2.4.1 Sumatoare binare cu control prin paritate

Lanțul amenințărilor la adresa dependabilității și siguranței:

$$\text{---- fault} \xrightarrow{\text{activation}} \text{error} \xrightarrow{\text{propagation}} \text{failure} \xrightarrow{\text{causality}} \text{fault ----}$$

Creșterea fiabilității prin utilizarea controlului prin paritate:

- atașarea unui bit de paritate par la operanzii adunării $X$, $Y$ și $Z$

Se consideră operanzii pe $n$ biți:

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

$Y \longrightarrow x_p = y_{n-1} \oplus y_{n-2} \oplus \cdots \oplus y_1 \oplus y_0$

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

## 2.4.1 Sumatoare binare cu control prin paritate (contin.)

Totuși, bitul de paritate $z_p$ poate fi anticipat, pornind de la definiția lui $z_i$:

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

Pentru a anticipa valoarea lui $z_p$ este definită o nouă valoare asociată cu toți biții transport excluzându-l pe cel mai semnificativ:

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

În ecuația (1), se poate înlocui $z_i$ ca mai sus iar după regrupare, expresia lui $z_p$ se obține astfel

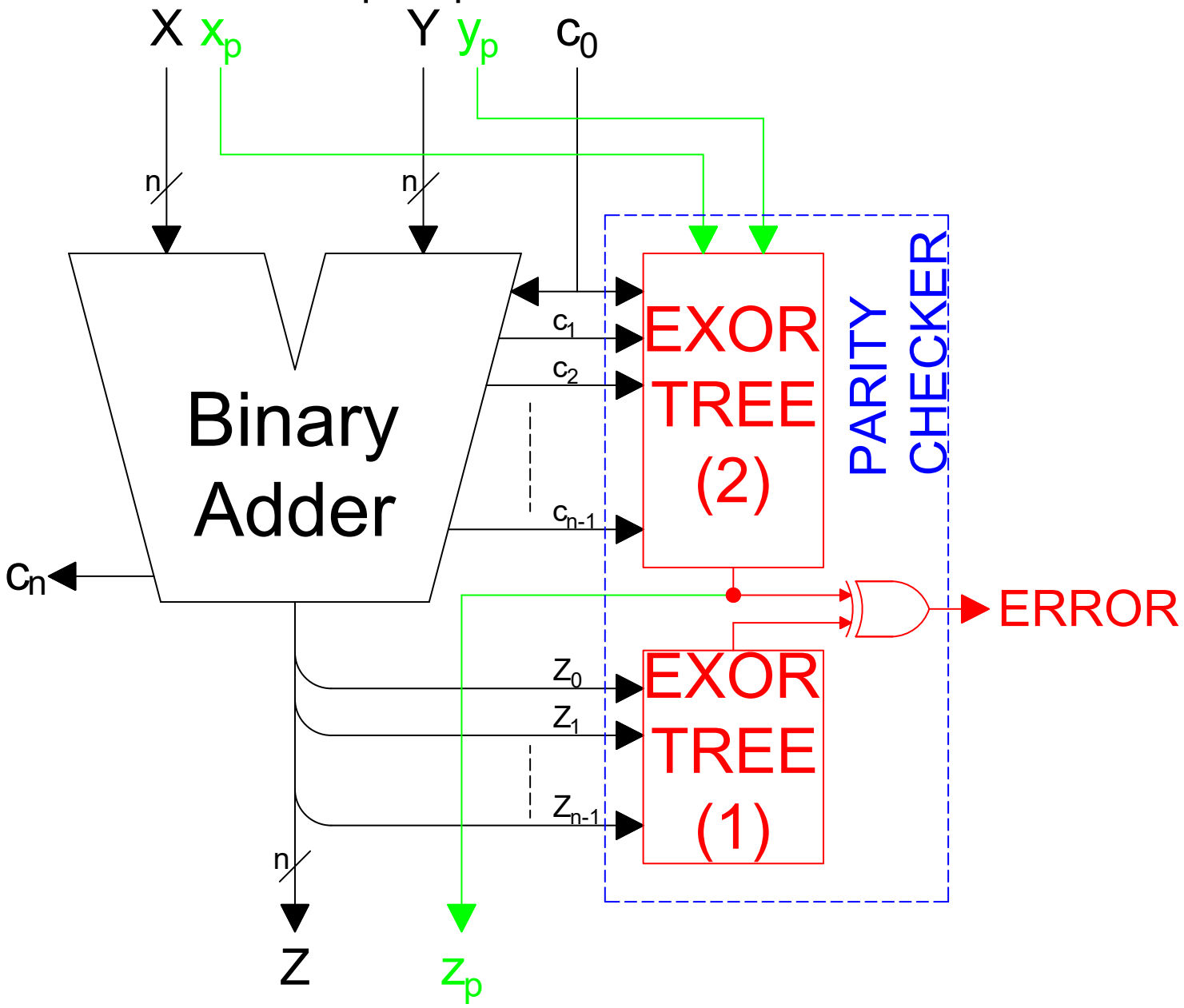
$$z_p = x_{n-1} \oplus y_{n-1} \oplus c_{n-1} \oplus x_{n-2} \oplus y_{n-2} \oplus c_{n-2} \oplus \cdots \oplus x_0 \oplus y_0 \oplus c_0$$
$$z_p = x_p \oplus y_p \oplus c_p \text{ ecuația (2)}$$

Ecuația (2):

- determină $z_p$ mai repede decât (1)
- anticipează valoarea lui $z_p$ din (1)

## 2.4.1 Sumatoare binare cu control prin paritate (contin.)

Sumator cu control prin paritate:



Erori care pot afecta un sumator binar:

- defecte singulare
  - mai dificil de detectat
  - probabilitate mai mare
- defecte multiple

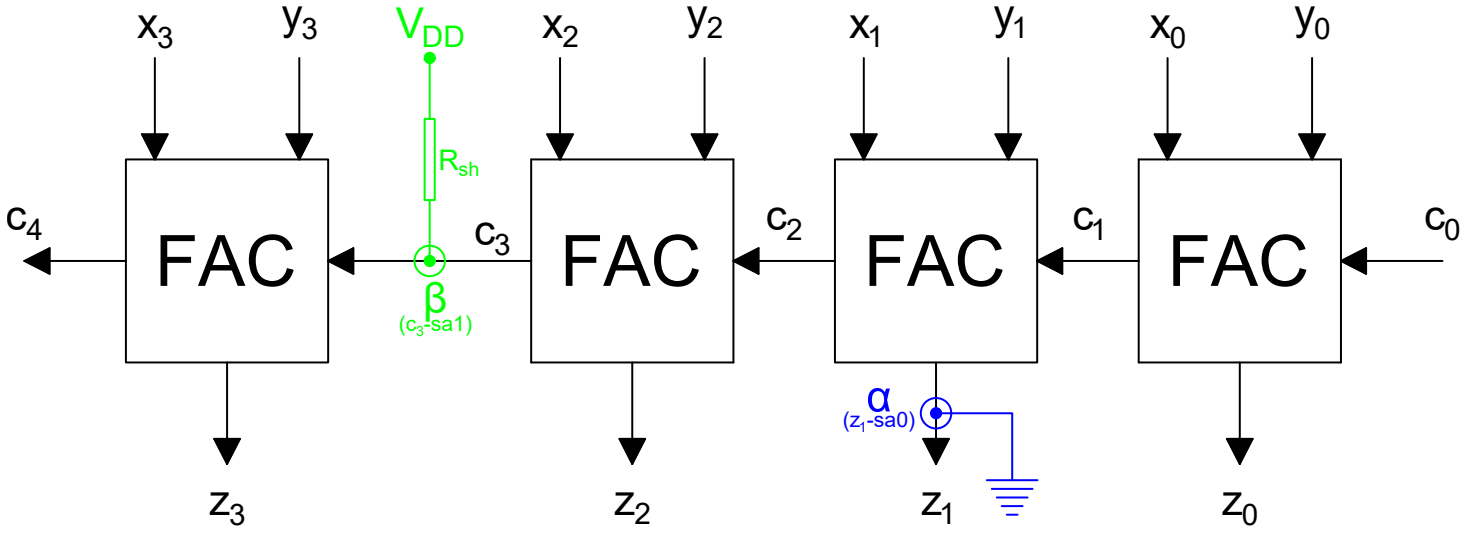
Defecte singulare cu manifestare logică: Single Stuck-at Fault (SSaF)

- *manifestare logică*: defectul afectează valoarea logică a cel putin unui semnal

## 2.4.1 Sumatoare binare cu control prin paritate (contin.)

Se consideră un RCA pe 4 biți, afectat de 2 defecte

- defect $z_1$ Stuck-at-0 (SA0): afectează $z_1$; fără efect asupra $c_2$ $\Rightarrow$ număr impar de biți eronați
- defect $c_3$ Stuck-at-1 (SA1): afectează $c_3$; întrucât $z_3$ depinde de $c_3$, $z_3$ este afectat, de asemenea; $c_4$ poate să fie sau nu afectat: dacă $c_4$ este afectat, în cazul unui sumator cu lățime mai mare, implicitl $z_4$ va fi afectat $\Rightarrow$ număr par de biți eronați



Se consideră operanzii $X = 0011$, $Y = 0011$ și $c_{in} = 0$.

În cazul lipsit de defecte, pentru că bitul de paritate $z_p$ calculat prin ecuația (1) este egal cu cel calculat prin ecuația (2), concluzia este că nu a apărut nicio eroare.

$$
\begin{array}{lll}
X= & 0\ 0\ 1\ 1 & x_p=0\oplus0\oplus1\oplus1=0 \\
Y= & 0\ 0\ 1\ 1 & y_p=0\oplus0\oplus1\oplus1=0 \\
C= & \overset{c_4}{0}\ \underbrace{0\ 1\ 1\ 0}_{z_p(1)=0\oplus1\oplus1\oplus0=0}\ \overset{c_0}{} & c_p=0\oplus1\oplus1\oplus0=0 \\
Z= & 0\ 1\ 1\ 0 & z_p(2)=x_p\oplus y_p\oplus c_p=0
\end{array}
$$

Pentru cazul adunării în care $z_1$ este SA0 (defectul $\alpha$ este activ), bitul de paritate $z_p$ calculat cu ecuația (1) diferă de cel calculat cu ecuația (2), concluzia fiind că a apărut o eroare.

$$
\begin{array}{lll}
X= & 0\ 0\ 1\ 1 & x_p=0\oplus0\oplus1\oplus1=0 \\
Y= & 0\ 0\ 1\ 1 & y_p=0\oplus0\oplus1\oplus1=0 \\
C= & \overset{c_4}{0}\ 0\ 1\ 1\ \overset{c_0}{0} & c_p=0\oplus1\oplus1\oplus0=0 \\
Z= & \underbrace{0\ 1\ 0\ 0} & z_p(2)=x_p\oplus y_p\oplus c_p=0 \\
 & & z_p(1)=0\oplus1\oplus0\oplus0=1
\end{array}
$$

Here is the translation of the given text into Romanian:
Pentru cazul adunării în care $c_3$ este SA1 (defectul $\beta$ este activ), bitul de paritate $z_p$ calculat cu ecuația (1) este același cu cel calculat cu ecuația (2), concluzia **incorectă** fiind că nu a apărut nicio eroare.

$$
\begin{array}{lll}
X= & 0\ 0\ 1\ 1 & x_p=0\oplus0\oplus1\oplus1=0 \\
Y= & 0\ 0\ 1\ 1 & y_p=0\oplus0\oplus1\oplus1=0 \\
C= & \overset{c_4}{0}\ 1\ 1\ 1\ \overset{c_0}{0} & c_p=1\oplus1\oplus1\oplus0=1 \\
Z= & \underbrace{1\ 1\ 1\ 0} & z_p(2)=x_p\oplus y_p\oplus c_p=1 \\
 & & z_p(1)=1\oplus1\oplus1\oplus0=1
\end{array}
$$

Un singur bit de paritate nu poate detecta toate posibilele SSaF:

- nu poate detecta erorile care afectează lanțul de transport, deoarece aceste erori duc la un număr par de biți modificați

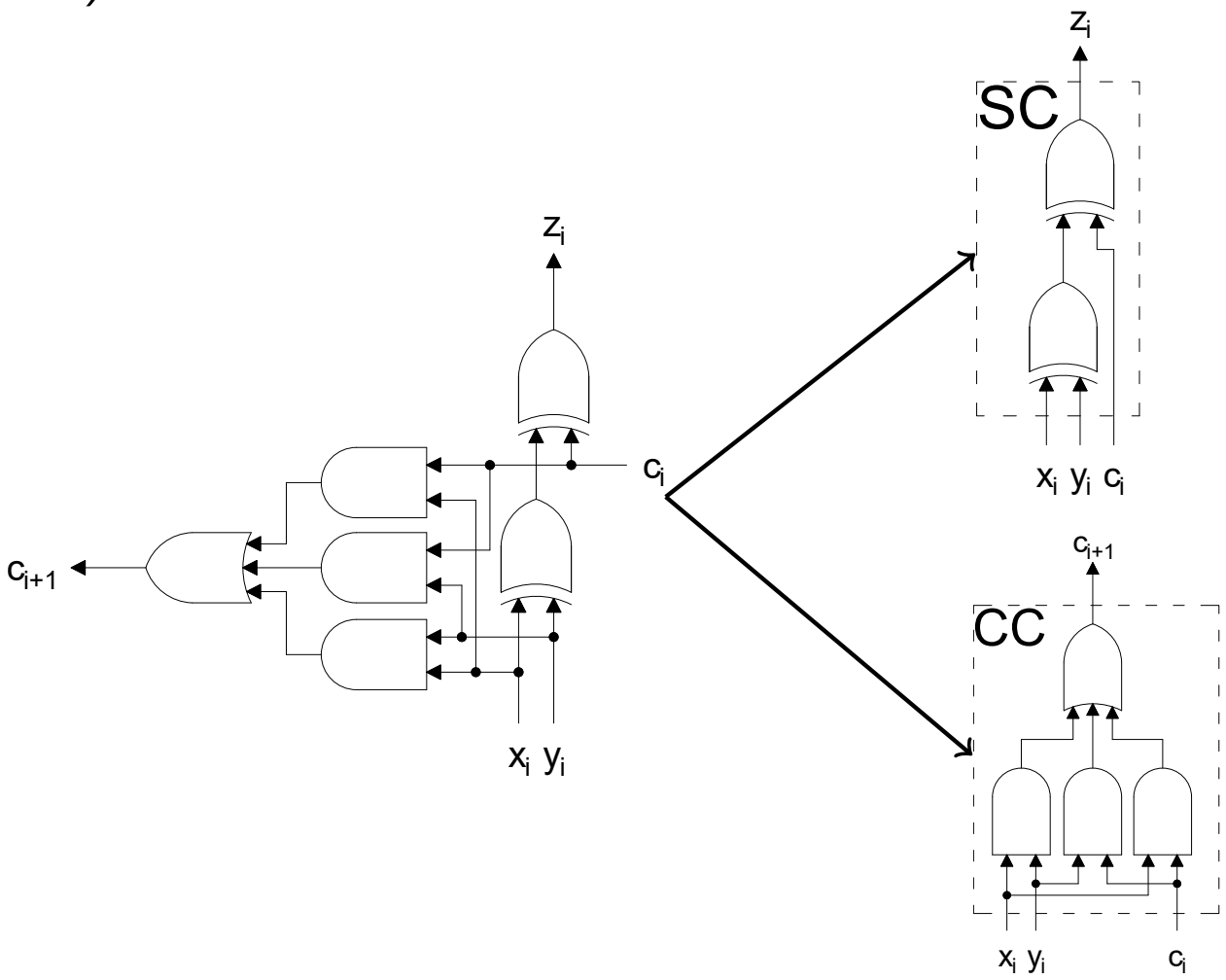
Metode de detectare a SSaF care afectează lanțul de transport:

- Duplicarea lanțului de transport
- Sumator Carry Dependent Sum

## 2.4.2 Duplicarea lanțului de transport

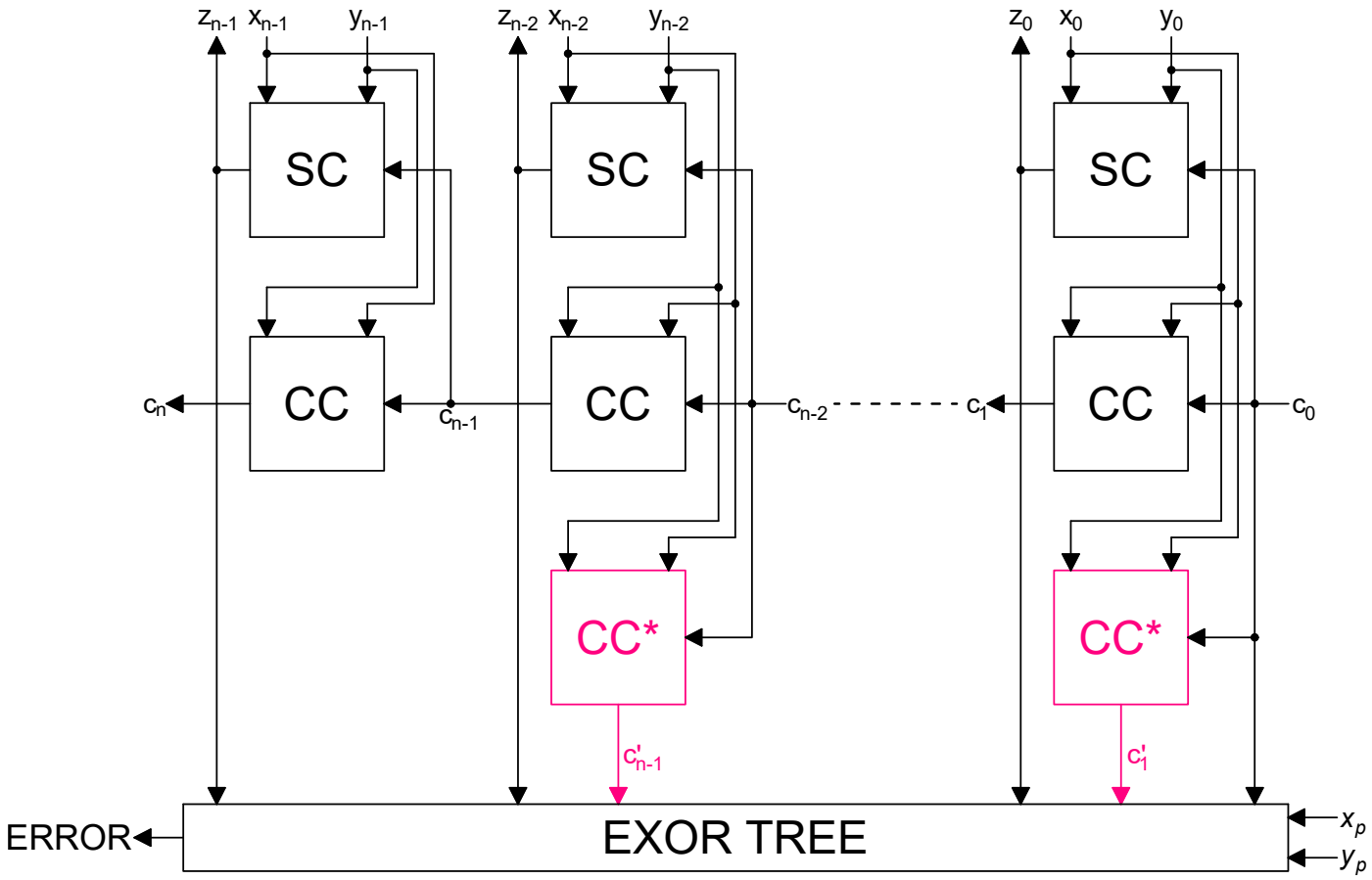
(A) Duplicarea lanțului de transport aplicată pentru RCA

FAC este împărțită în două părți: una calculează bitul de sumă (Sum Cell - SC) și alta generează transportul de ieșire (Carry Cell - CC).

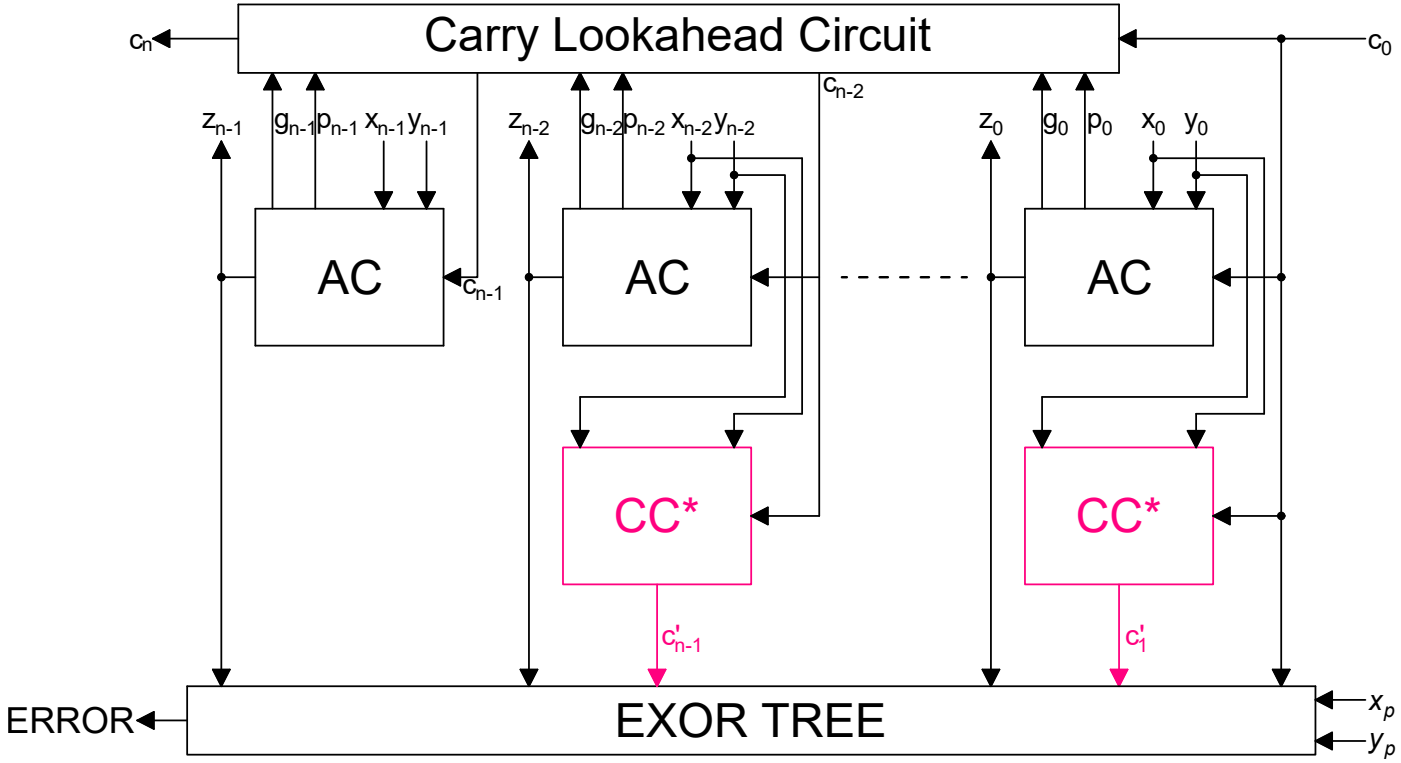


## 2.4.2 Duplicarea lanțului de transport (contin.)

RCA cu duplicarea Lnțului de transport este ilustrat în figura de mai jos:



(B) Duplicarea lanțului de transport aplicată pentru Carry Lookahead Adder

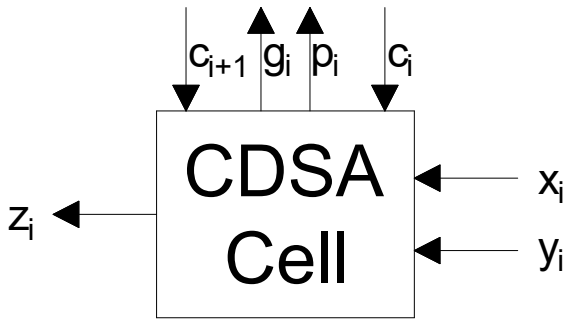


## 2.4.3 Sumator Carry Dependent Sum

Abordează problema fiabilității încă din faza de proiectare:**

- când $c_{i+1}$ este eronat, sumatorul Carry Dependent Sum (CDSA) forțează bitul sumă $z_i$ la valoare greșită
  - crează un dezechilibru artificial în numărul de biți eronați
  - pe lângă $x_i$, $y_i$ și $c_i$, bitul transport $c_{i+1}$ este intrare a celulei CDSA $\Rightarrow$ biții transport trebuie obținuți în maniera lookahead

Simbolul celulei CDSA:



## 2.4.3 Sumator Carry Dependent Sum (contin.)

Tabelul de adevăr al celulei CDSA:

| 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 |

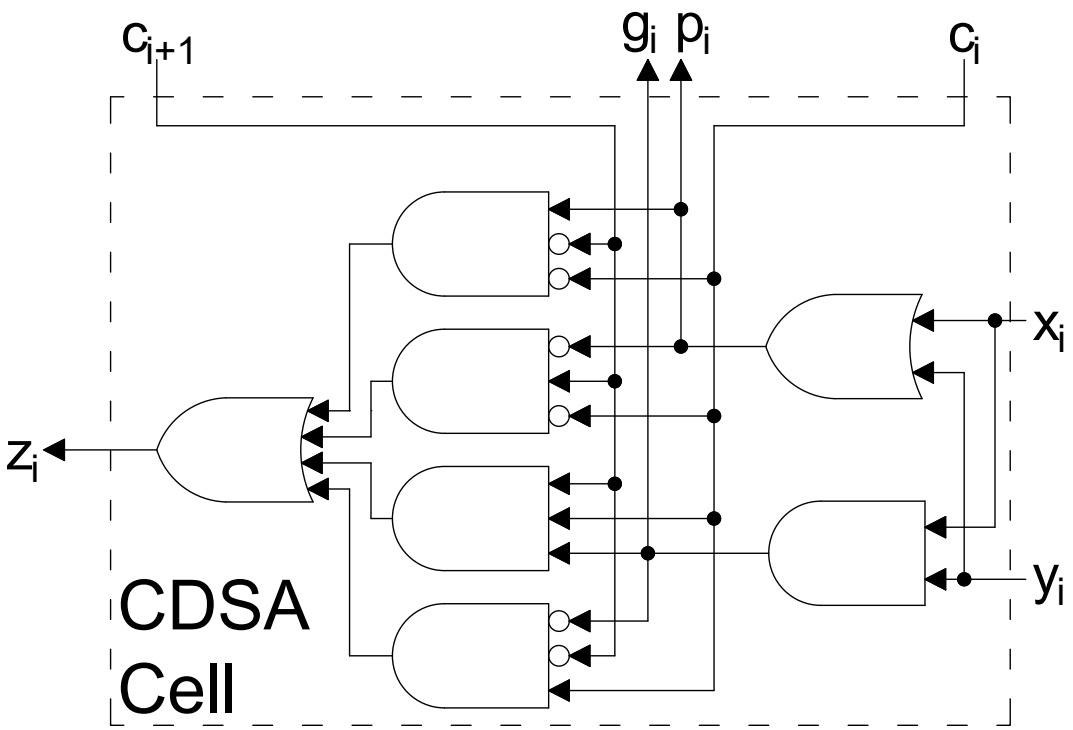
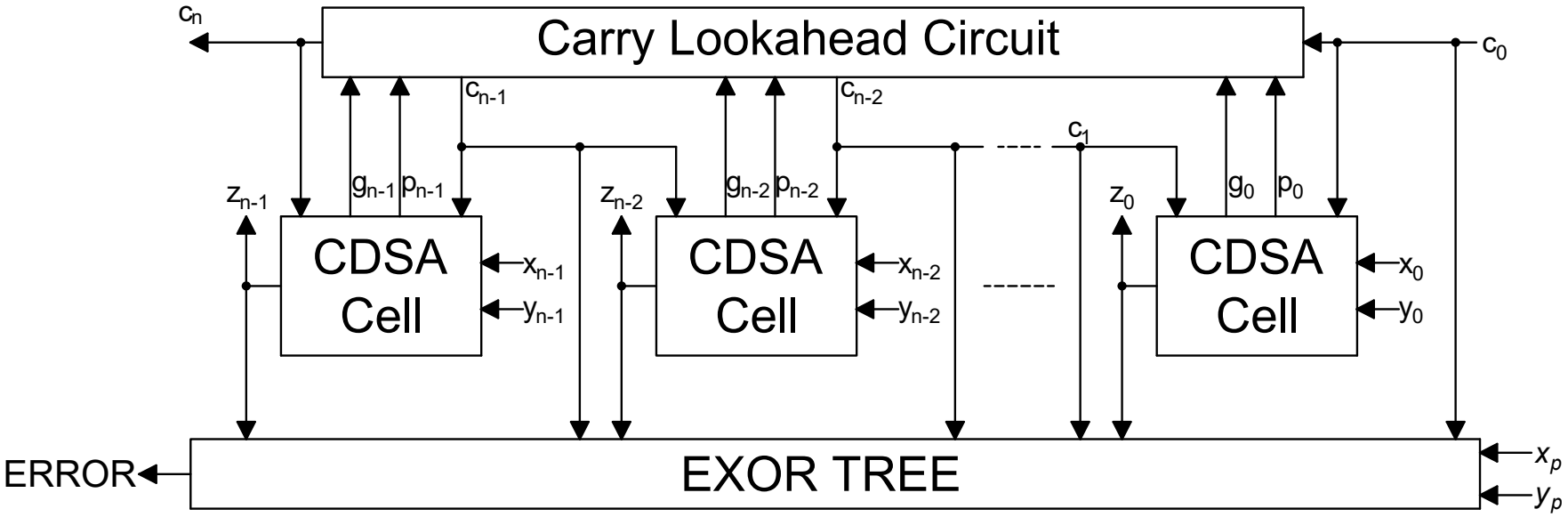
## 2.4.3 Sumator Carry Dependent Sum (contin.)

Diagrama Karnaugh a celului CDSA și ecuația ieșirii:

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

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

Sinteza celulei CDSA:



Utilizând celula sintetizată, arhitectura CDSA este prezentată în figura de mai jos: