

Arhitectura calculatorelor

(curs 8 - S8)

2.4. Calcul fiabil

Atributele sistemelor de calcul

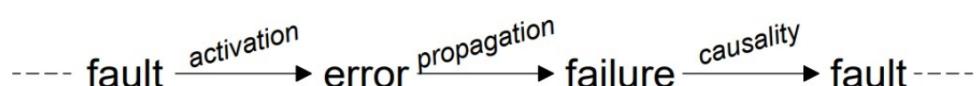
- performanță (întâzietă, nr operații/s)
- consum de putere
- fiabilitate

Atributele calculului fiabil:

- disponibilitate: sistemul este pregătit să își ofere serviciile
- fiabilitate: sistemul își oferă serviciile într-un mod continuu
- mențenabilitate: abilitatea de a efectua reparări și modificări

2.4.1. Sumătoare binare cu control prin paritate

Lanțul amenințărilor la adresa dependabilității și siguranței:



Creșterea fiabilității prin utilizarea controlului prin paritate:

→ atașarea unui bit de paritate par la operanții adunării $X, Y \oplus Z$

Se consideră operațiuni pe n biti

$$X \rightarrow X_p \quad Y \rightarrow Y_p \quad Z \rightarrow Z_p$$

$$Z = X + Y$$

! nr de biti de 1 → par

Dacă avem $101 \xrightarrow{X_p} 1010$

$$1101 \Rightarrow 11011$$

$$\Rightarrow X \rightarrow X_p = X_{m-1} \oplus X_{m-2} \oplus \dots \oplus X_0$$

$$Y \rightarrow Y_p = Y_{m-1} \oplus Y_{m-2} \oplus \dots \oplus Y_0$$

| înălătri
→ vor fi primite

$$(1) Z \rightarrow Z_p = Z_{m-1} \oplus Z_{m-2} \oplus \dots \oplus Z_0 \rightarrow \text{ieșire}$$

→ trebuie determinat

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 , definim o nouă valoare asociată cu toti bitii transport **excluzându-l pe cel mai semnificativ**.

$$C_p = C_{m-1} \oplus C_{m-2} \oplus \dots \oplus C_0 \rightarrow \text{nu include } C_m$$

$\hat{\wedge}$ Înlocuim în (1)

$$Z_p = X_{m-1} \oplus Y_{m-1} \oplus C_{m-1} \oplus X_{m-2} \oplus Y_{m-2} \oplus C_{m-2} \oplus \dots \oplus X_0 \oplus Y_0 \oplus C_0$$

$$\Rightarrow Z_p = X_p \oplus Y_p \oplus C_p \quad (2)$$

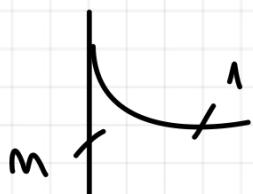
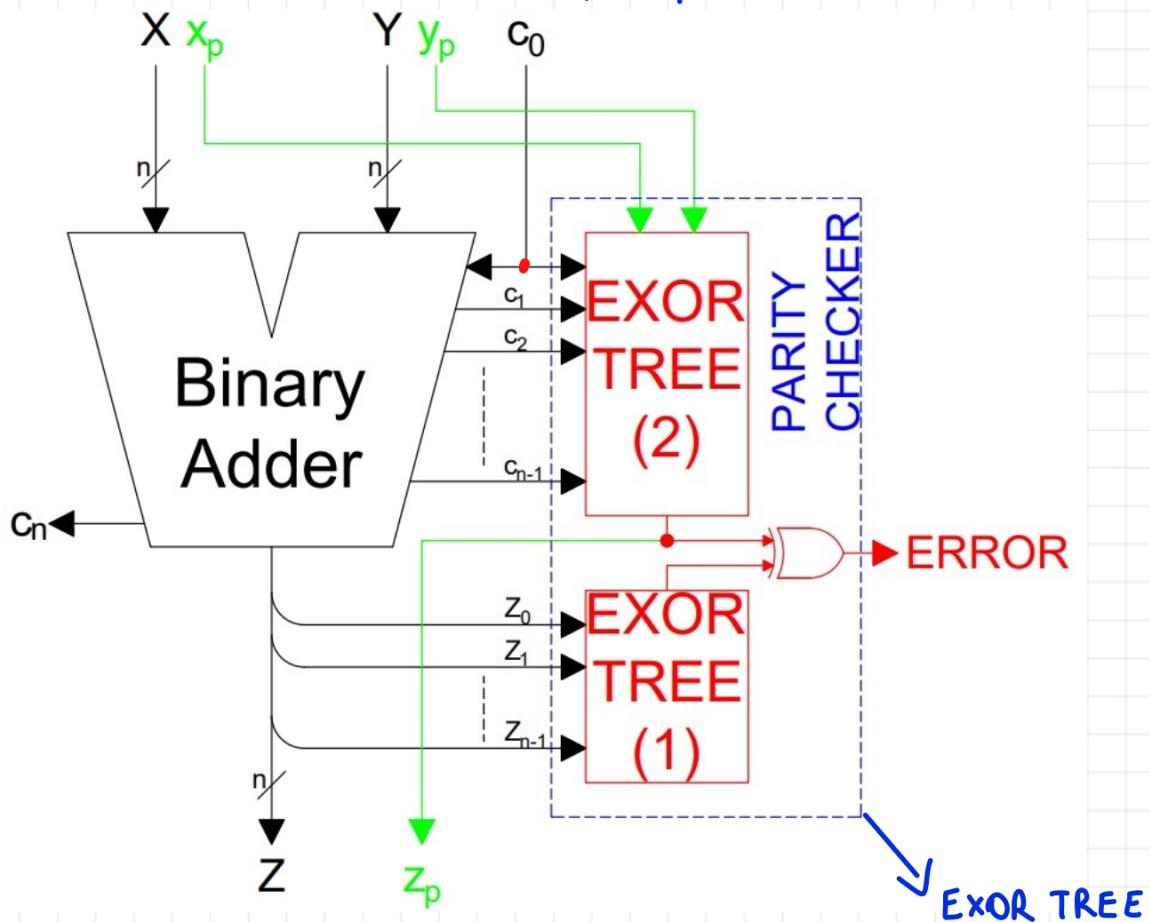
→ determinăm mai repede Z_p decât (1)

→ anticiparea valoarei lui Z_p din (1)

Pentru a verifica dacă a întrevenit o eroare:

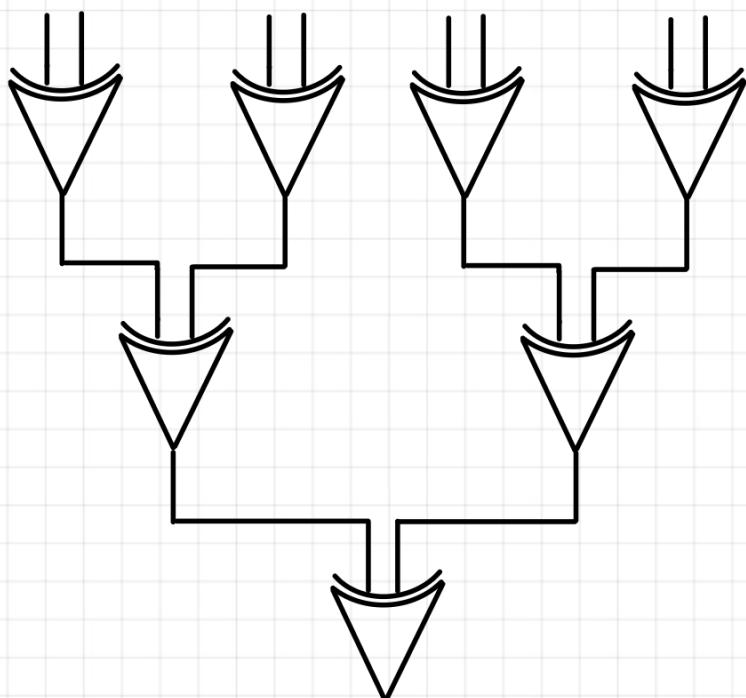
$$Z_p(1) = Z_p(2)$$

Arhitectură sumator cu control prin paritate:



\Rightarrow așa se ia un subvector din vectorul de intrare

EXOR TREE pentru 8 biti



Dacă m nu este o putere a lui 2, arborele trebuie să fie complet.

Modul de răspuns la întâlnirea unei erori depinde de cum a fost proiectat sistemul.

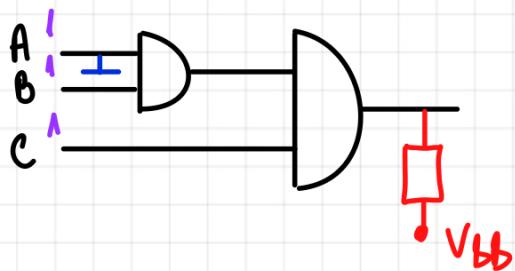
Erori care pot afecta un sumator binar:

1. defecte singulare

- ─ mai dificil de detectat | apar mai des
- ─ probabilitate mai mare | sunt mai greu de detectat

2. defecte multiple

De exemplu memorile ECC (Error Correction Memory) permit corectarea unui nr finit de biți greșiti.



A - permanent legată la 0

- pt. detectie punem 1 la toate intrările ieșire la 1

pt. detectie trice vector ce conține un 0

→ defectele multiple sunt mai ușor de detectat

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

→ manifestare logică: defectul afectează valoarea logică a cel

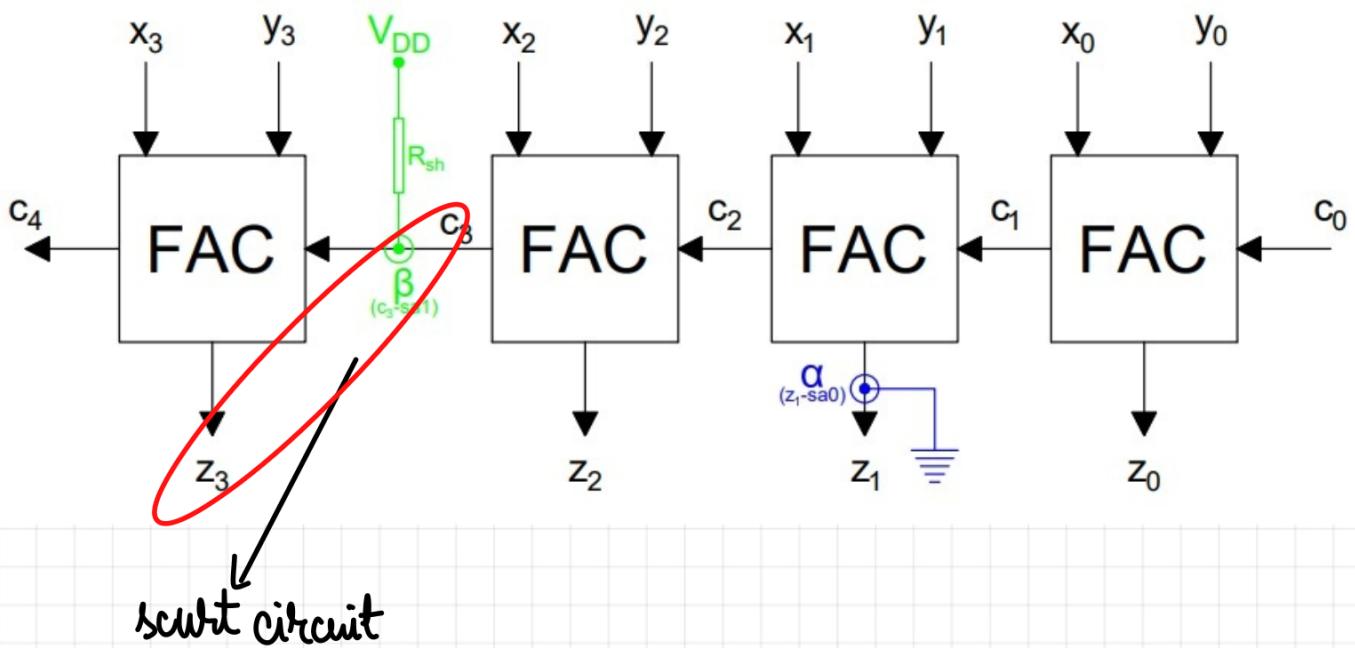
→ 0 linie rămâne la 0 putin un semnal

Ex: RCA - 4

→ defect z_1 Stuck-at-0 (SA0): afectează z_1 ; fără efect asupra c_2 ⇒ nr. impar de biți eronati

→ 2 defecte
→ defect z_3 Stuck-at-1 (SA1): afectează c_3 ; întrucât z_3 depinde de c_3 , z_3 va fi, de asemenea, afectat; c_4 poate fi sau nu poate fi afectat

→ dacă este afectat, în cazul unui sumator cu latime mai mare $\Rightarrow z_4$ va fi afectat
⇒ număr par de biți eronati



$$z_3 = x_3 \oplus y_3 \oplus c_3$$

Considerăm doar α sau doar β prezente

$$\beta \Rightarrow c_3 \rightarrow \bar{c}_3 \quad \left. \begin{array}{l} \text{de astăzi negäm} \\ \text{corect } c_3 = 1 \text{ (SA)} \end{array} \right\}$$

$\oplus \rightarrow$ funcție liniară intrare modificată \Rightarrow ieșire modificată

\rightarrow ar putea să-l modifice și pe c_4 , dar nu e neapărat nicio

$$\text{ex: } x_3 = y_3 = 0 \Rightarrow c_4 = 0$$

merci

$$x_3 = y_3 = 1 \Rightarrow c_4 = 1$$

$$x_3 \oplus y_3 = 1 \Rightarrow c_4 = c_3$$

dacă c_3 este afectat $\Rightarrow z_4$ va fi afectat

$\beta \rightarrow$ provoacă nr par de eroare
(eroare lant de carry)

$\alpha \rightarrow$ provoacă o singură eroare
(eroare bit de sumă)

DEF: $X = 0011$ $x_p = 0 \oplus 0 \oplus 1 \oplus 1 = 0$
 $Y = 0011$ $y_p = 0 \oplus 0 \oplus 1 \oplus 1 = 0$
 $C = 00110$ $c_p = 0 \oplus 1 \oplus 1 \oplus 0 = 0$ (fără MSB)
 fault free $Z = 0110$ $\Sigma_p(2) = x_p \oplus y_p \oplus c_p = 0$
 $\Sigma_p(1) = 0 \oplus 1 \oplus 1 \oplus 0 = 0$ sunt egale

În cazul lipsit de defecte, pentru că bitul de paritate Σ_p calculat prin ecuația (1) este egal cu cel calculat prin ecuația (2) \Rightarrow nu a apărut nicio eroare fault α

$\Rightarrow Z_1$ SA0 (stuck at 0) $\rightarrow Z_1 = 0$ corect

$$\begin{array}{r} X = 0011 \\ Y = 0011 \\ \hline C = 00110 \end{array} \quad \begin{array}{l} x_p = 0 \\ y_p = 0 \\ c_p = 0 \end{array}$$

$$\begin{array}{r} Z = 0100 \\ \Sigma_p(2) = 0 \end{array}$$

$$\Sigma_p(1) = 1$$

nu mai sunt egale
ERROR

Pentru cazul adunării în care Z_1 este SA0 (defectul α este activ), bitul de paritate $\Sigma_p(1)$ diferă de $\Sigma_p(2) \Rightarrow$ avem eroare

fault β

$\Rightarrow C_3$ SA1

$$\begin{array}{r} X = 0011 \\ Y = 0011 \\ \hline C = 01110 \end{array} \quad \begin{array}{l} x_p = 0 \\ y_p = 0 \\ \Sigma_p = 1 \end{array}$$

$$\begin{array}{r} Z = 110 \\ \text{ar trebui să fie } 0 \end{array}$$

$$\Sigma_p(2) = 1$$

$$\Sigma_p(1) = 1$$

NO ERROR??
nu

Pentru cazul adunării în care C_3 este SA1 (defectul β activ) bitul de paritate $\Sigma_p(1)$ nu diferă de $\Sigma_p(2) \Rightarrow$ concluzia INCORECTĂ că nu avem eroare

Un singur bit de paritate nu poate detecta erori care afectează lantul de carry, deoarece aceste erori duc la un nr. par de biți modificăți.

→ toate posibilele SSAT

Metode de detectare a SSAT care afectează lantul de transport

→ duplicarea lantului de transport

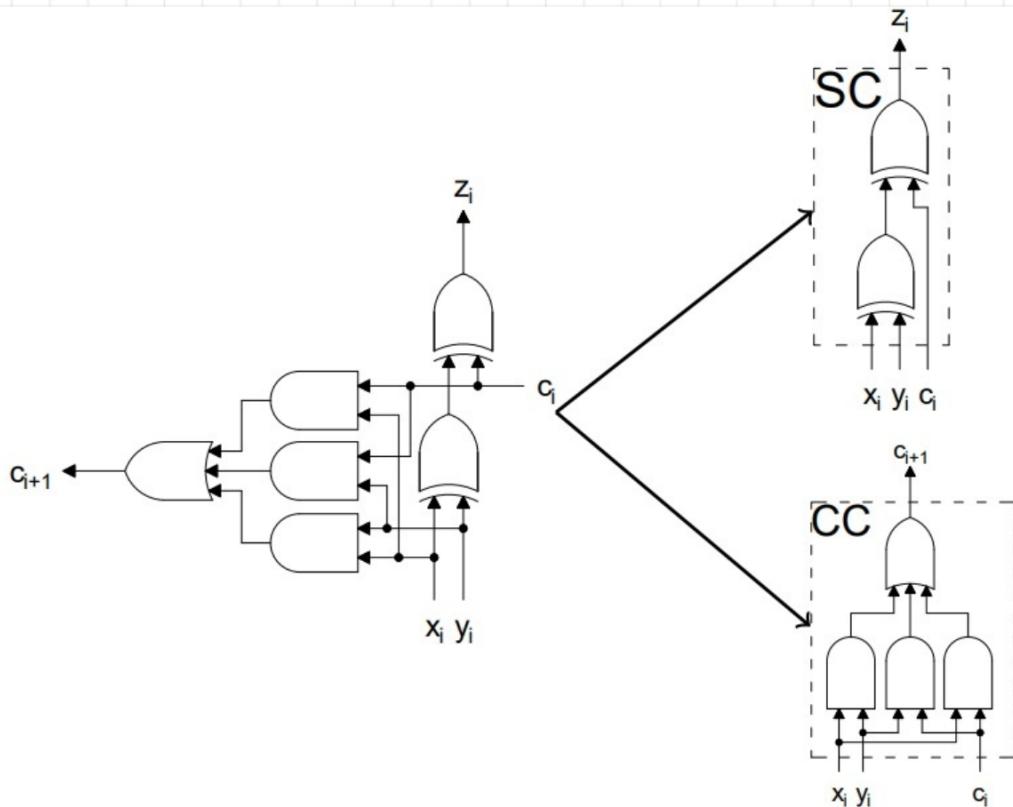
Sumator Barry Dependent Adder

2.4.2. Duplicarea lantului de transport

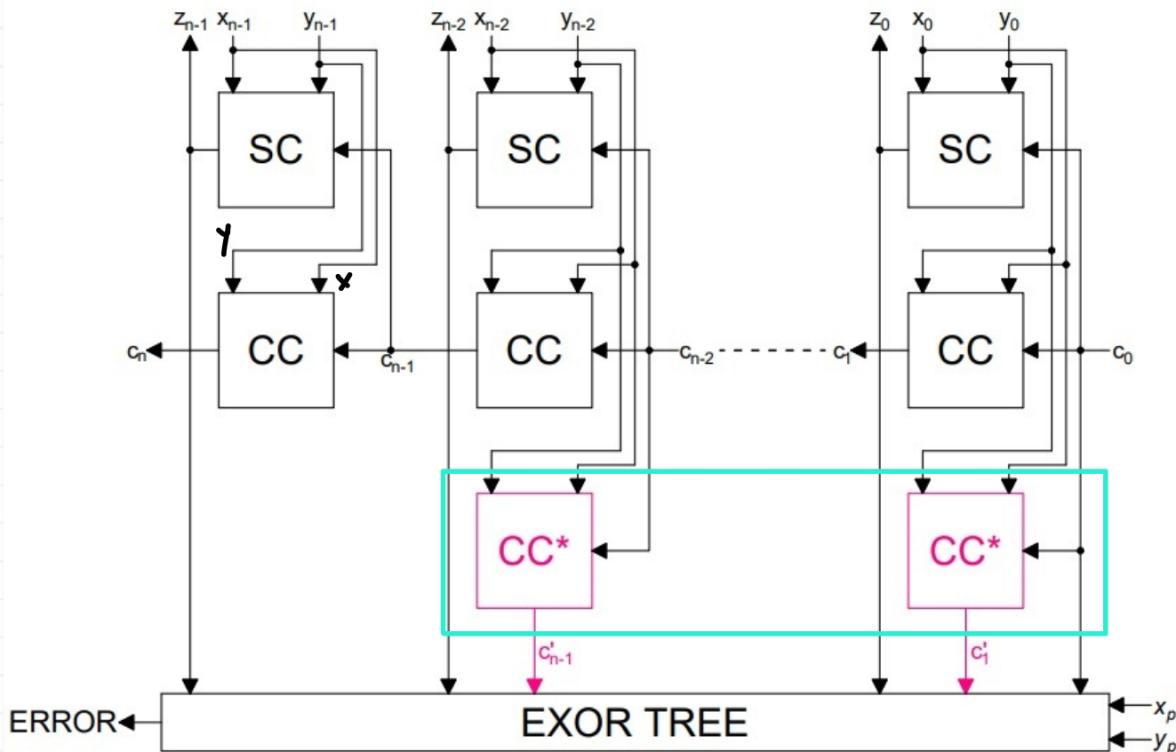
A. Duplicarea lantului de transport aplicată pentru RCA

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

! Al doilea lant de carry il voi folosi pentru a calcula tocmai c_p (bit de paritate)

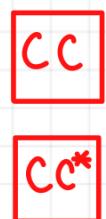


RCA cu duplicarea lantului de transport: ~ m biti



Ordinea X, Y sau Y, X nu afectează funcționarea

CC^* are circuitele sale individuale
↳ copie a CC



2 instanțe ale aceleiași componente

→ dacă c_{m-2} are eroare \Rightarrow ea va fi propagată la c_{m-1} și c'_{m-1} , $+ c_{m-2}, c'_{m-2}$

Dacă ele sunt greșite $\Rightarrow z_{m-2}, z_{m-1}$ - greșite

\Rightarrow nr. par de eroare

BOAR CĂ: în EXOR TREE nim caddy-urile din celelalte copii \Rightarrow mereu nr. impar

c_{m-2}
afectată $\rightarrow CC \Rightarrow CC'$ nu este afectată
 $\downarrow c'_{m-2}$ (luăm un singur defect) $\Rightarrow c'_{m-2} \rightarrow$ corect

\Rightarrow doar z_{m-2} greșit și c'_{m-1} (dar nu neapărat)

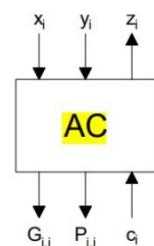
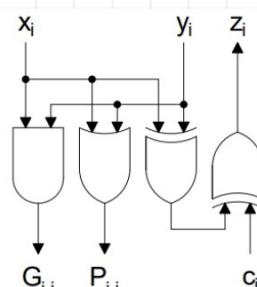
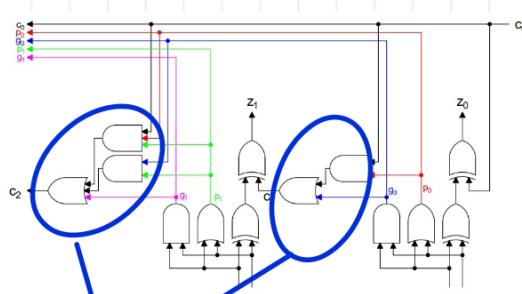
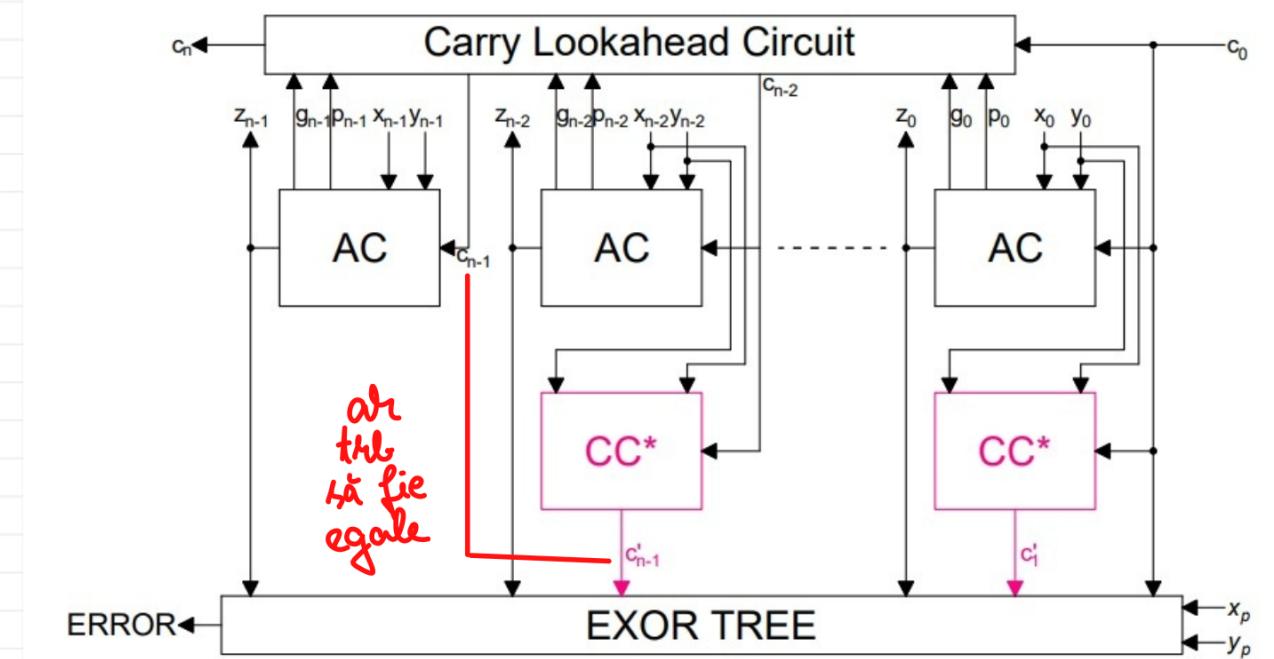
$\square \rightarrow$ lant pt detectia erorii
cel de mai sus e pt. calculul sumei

c'_m nu se generează decat ce nu e utilizat în EXOR TREE pt. paritate

și posibil să-l afecteze și pe c_{m-1} care nu afectă și z_{m-1}

! Diferența dintre $\Sigma_p(1)$ și $\Sigma_p(2)$ poate fi pusă în evidență când avem nr. impar de biți de paritate

B. Duplicarea lantului de transport aplicată pentru Carry Lookahead Adder



! intenția este de a avea 2 surse de generație pentru fiecare carry $i \in [1, m-1]$

2.4.3. Sumator Carry Dependent Sum (CDSA)

→ Abordează problema fiabilității încă din fază de proiectare.

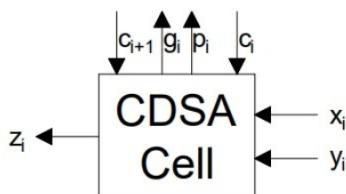
▷ când c_{i+1} este eronat CDSA forțează bitul sumă z_i la valoarea greșită

▷ intenția este de a crea un dezechilibru artificial în nr. de biți eronati

▷ pe lângă x_i, y_i și c_i , bitul transport c_{i+1} este intrata a celulei CDSA
 \Rightarrow bitii transport trebuie obținuți în modulă lookahead.

! dacă c_{i+1} - incorrect $\Rightarrow z_i$ devine incorrect

Simbol CDSA:



Tabel de adevăr

c_{i+1} - corect $\Rightarrow z_i$ - corect

	Inputs				Output
	x_i	y_i	c_i	c_{i+1}	z_i
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	1

$$0+0+0 \Rightarrow S: 0 \quad C: 0, \text{dar } c_{i+1}=1 \Rightarrow z_i=1$$

$$0+0+1 \Rightarrow S=1, C: 0 \quad \checkmark$$

$$-11-, c_{i+1}=1 \Rightarrow z_i=0$$

$$0+1+1 \Rightarrow S=0, C=1; c_{i+1}=0 \Rightarrow S=1$$

Diagramma Tafelmaugh

x_i, y_i	c_i, c_{i+1}	00	01	11	10
z_i		0	1	0	1
		0	1	3	2
00		0	1	3	2
01		1	0	5	6
11		1	0	15	14
10		1	0	4	10

$$z_i = \frac{y_i \cdot \bar{c}_i \cdot \bar{c}_{i+1}}{(4, 12)} + \frac{x_i \cdot \bar{c}_i \cdot \bar{c}_{i+1}}{(8, 12)} + \frac{\bar{x}_i \cdot \bar{y}_i \cdot \bar{c}_i \cdot c_{i+1}}{(1)}$$

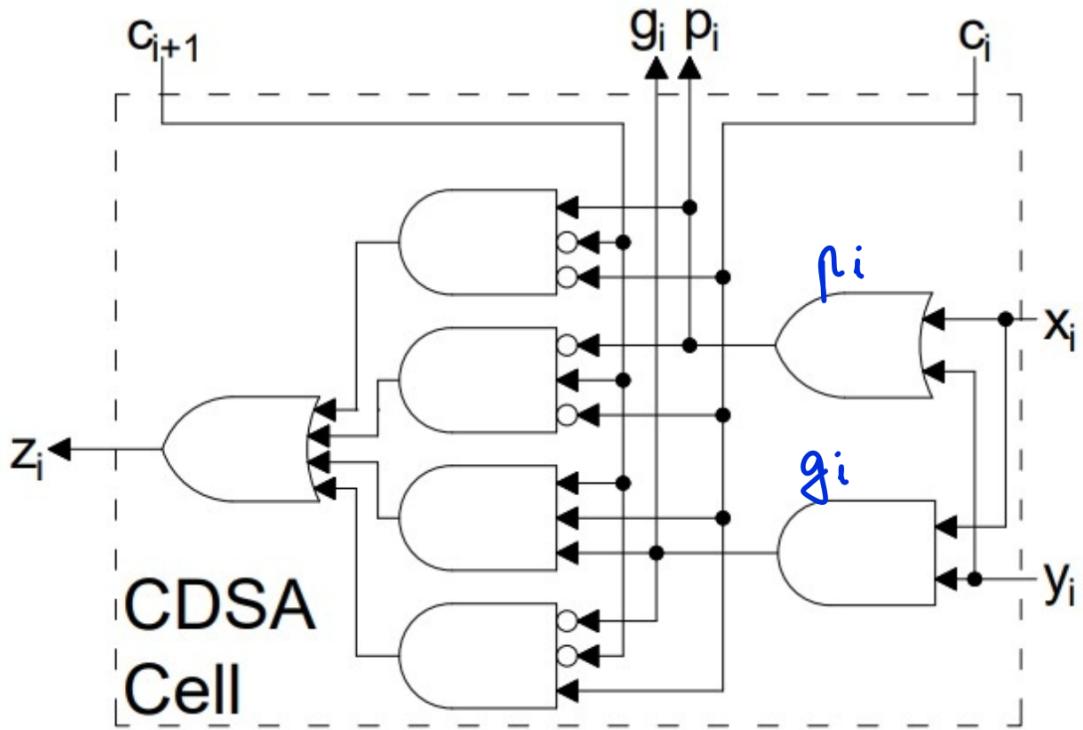
$$+ \frac{x_i \cdot y_i \cdot c_i \cdot c_{i+1}}{(15)} + \frac{\bar{x}_i \cdot c_i \cdot \bar{c}_{i+1}}{(2, 6)} + \frac{\bar{y}_i \cdot c_i \cdot \bar{c}_{i+1}}{(2, 10)}$$

Formulă: $\frac{\bar{x}_i \cdot \bar{y}_i}{x_i + y_i} = \frac{\bar{x}_i + y_i}{x_i + y_i} = \bar{p}_i$
 $\frac{x_i + \bar{y}_i}{x_i + y_i} = \frac{x_i \cdot \bar{y}_i}{x_i + y_i} = \bar{g}_i$

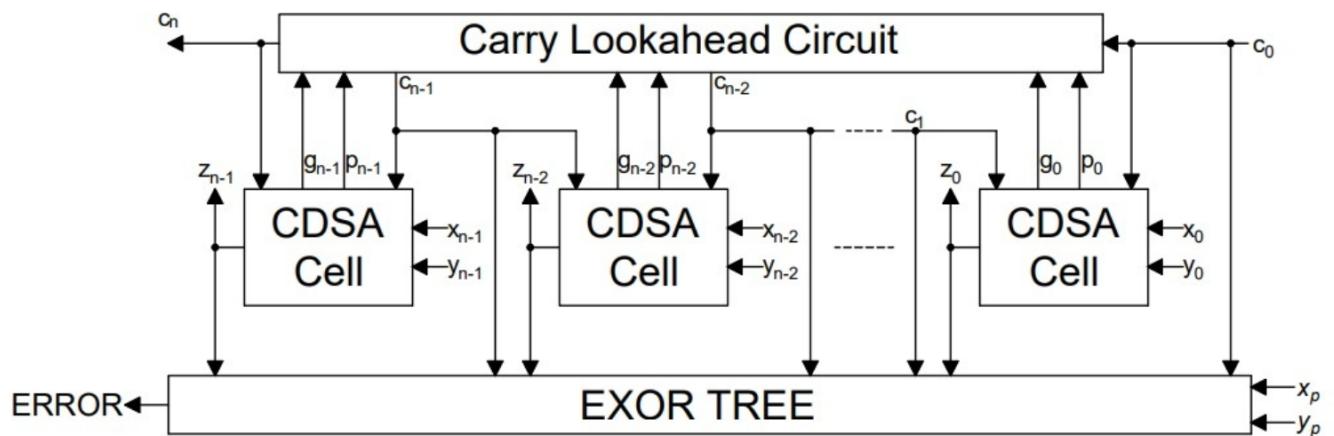
$$\begin{aligned} z_i &= \bar{c}_i \cdot \bar{c}_{i+1} (x_i + y_i) + \frac{\bar{x}_i + y_i}{\bar{c}_i \cdot c_{i+1}} + x_i \cdot y_i \cdot c_i \cdot c_{i+1} + \\ &+ c_i \cdot \bar{c}_{i+1} (\bar{x}_i + \bar{y}_i) = \\ &= \bar{c}_i \cdot \bar{c}_{i+1} \cdot p_i + p_i \bar{c}_i c_{i+1} + g_i c_i c_{i+1} + c_i \cdot \bar{c}_{i+1} \bar{x}_i \bar{y}_i \end{aligned}$$

$$z_i = p_i \bar{c}_i \bar{c}_{i+1} + p_i \bar{c}_i c_{i+1} + g_i c_i c_{i+1} + \bar{g}_i c_i \bar{c}_{i+1}$$

Sintetiza celulei



CBSA utilizând celula sintetizată



III. Analiza funcțională și sinteza unităților aritmetice de mărgulă mobilă

3.1. Operări și arhitecturi de mărgulă mobilă

În general, se consideră operații IEEE 754 de mărgulă mobilă (floating point - FP), cu excepția cazului în care se specifică altfel.

Numerele IEEE 754 pot fi:

- împachetate (sau normalize) : pentru stocare/transmisie de date (nu are hidden bit)
- despachetate : utilizate în timpul calculelor (au bit ascuns)
- Significant unnormalized

$$\text{ex: } X = X_M \cdot 2^{X_E}$$

$$Y = Y_M \cdot 2^{Y_E}$$

Cele 4 operații fundamentale:

$$\textcircled{1} \quad X + Y = \left[X_M + Y_M \cdot 2^{Y_E - X_E} \right] \cdot 2^{X_E}$$

exponent

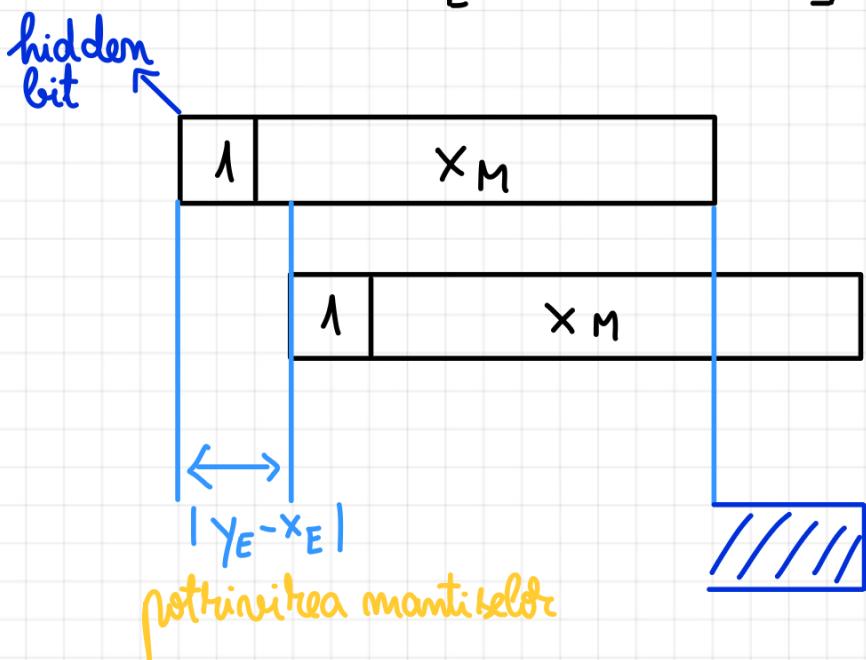
mantisa

allignment

if $X_E \geq Y_E$
învers dacă nu e adev.

$$\textcircled{2} \quad X - Y = \left[X_M - Y_M \cdot 2^{Y_E - X_E} \right] \cdot 2^{X_E}$$

if $X_E \geq Y_E$
învers dacă nu e adev.



$$\textcircled{3} \quad x \cdot y = x_M \cdot y_M \cdot 2^{x_E + y_E}$$

$$\textcircled{4} \quad \frac{x}{y} = \frac{x_M}{y_M} \cdot 2^{x_E - y_E}$$

! Toate aceste 4 operații aritmetice folosesc de regulă 2 subunități:

1. calculul exponentului: efectuarea adunării sau scăderii exponentilor
2. calculul significandului: efectuarea $+ ; - ; \cdot ; /$ a operanziilor

$\cdot 2^{Y_E - X_E}$ pentru aliniere
 \rightarrow right shift

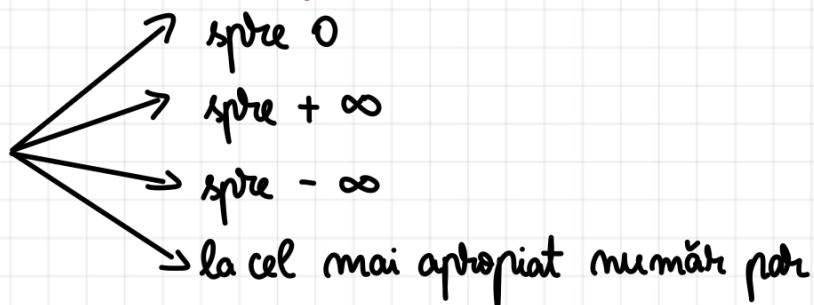
Cele 2 subunități operează doar cu operanzi de virgulă fixă:

- ▷ subunitatea exponentului folosește numere întregi de virgulă fixă (représentate în cod exces)
- ▷ subunitatea semnificandului operează cu numere fractionare de virgulă fixă.

3.2. Rotunjirea

→ se referă la conversia unei reprezentări de precizie ridicată la o reprezentare de precizie scăzută (pentru stocare/transmisie)

Moduri de rotunjire IEEE 754



În IEEE 754, rotunjirea face referire la biții fractionari cu ponderi mai mici decât ponderea celui mai puțin semnificativ bit al significandului. Pentru concizie, doar în acest paragraf, rotunjirea vizează conversia unui număr cu parte întreagă și fractionară într-un număr întreg.

Fie X

$$X = x_{m-1}x_{m-2}\dots x_1x_0 \cdot x_{-1}x_{-2}\dots x_{-m}$$

→ significandul în IEEE 754 are un singur bit în partea întreagă (bit **ascuns**).

Pentru simplitate îl vom considera nr. de virgulă fixă.

Fie x^* valoarea rotunjită a lui X , cu x^* fiind un întreg

$$x^* = x_{m-1}^* x_{m-2}^* \dots x_1^* x_0^*$$

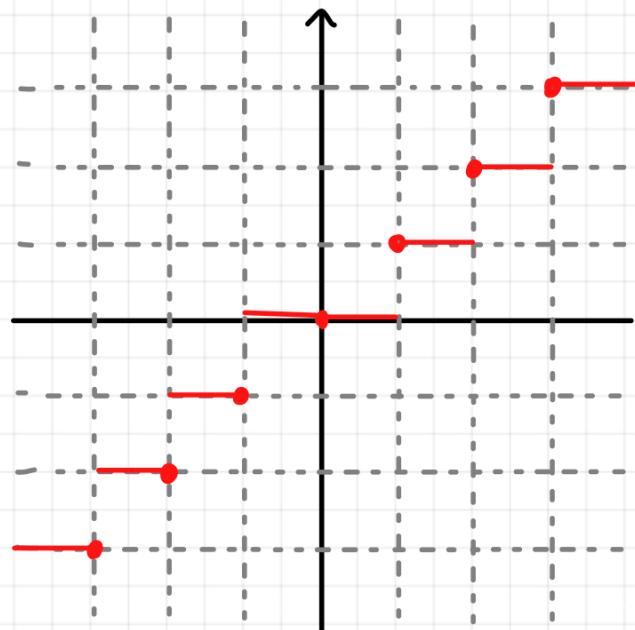
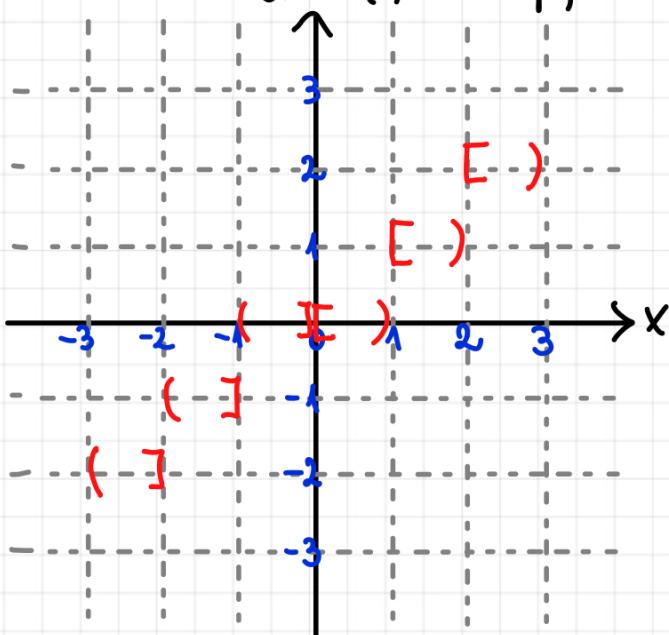
→ în precizie simplă în IEEE 754, rezultatul intermediar de $+$ / $-$ nu are mai mult de 23 de biți

A. Rotunjirea spre 0 (rotunjire spre interior / inwards)

! x^* este ales încât să fie cel mai mare întreg pentru care

$$|x^*| \leq |x|$$

down(x) → chop, truncare

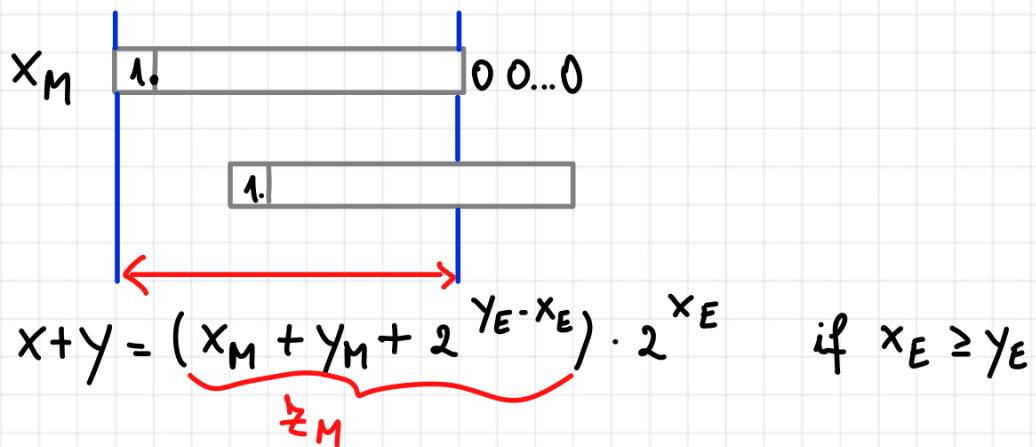


$$\begin{array}{l} 2,91 \rightarrow 2 \\ -3,25 \rightarrow -3 \\ \hline 1011.01 \end{array}$$

mă apropiu de 0

[) ≡ — în dreptul intervalului închis și lipsa punctului la)

Dacă x este reprezentat în S.M. rotunjirea către 0 înseamnă înzinsarea părții fractionare



Exercițiu:

$$x = + 3.625$$

$$x = 0 \ 011.101 \text{ SM}$$

$$\Rightarrow x^* = 0 \ 011 = 3$$

$$\text{dacă } x = -3,625$$

$$x = 1 \ 011.101 \text{ SM}$$

$$\Rightarrow x^* = 1 \ 011 = -3$$

$$|x^*| \leq |x| \quad \checkmark$$