

# Arhitectura calculatoarelor

## (curs 6 - 56)

### 2.3.2. Sumator Carry-Skip

→ Din ecuația de generare a transportului :

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

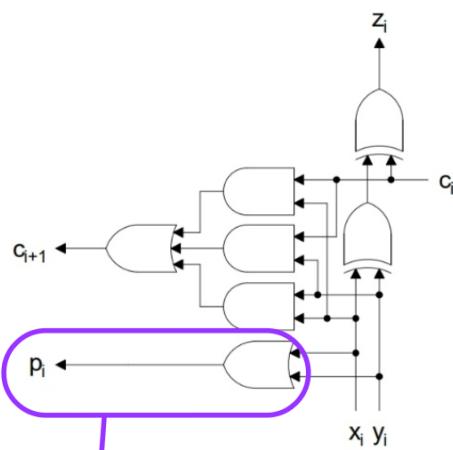
variabila de propagare la nivel de bloc se obține mai simplu.

→  $P_{i,j}$  este mai ușor de calculat

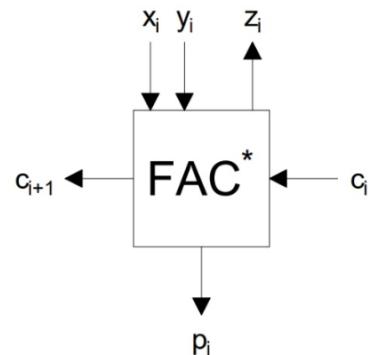
→ Se ignoră  $G_{i,j}$  pentru că este mai ușor de calculat carry-ul pe partea cealaltă.

Potem aplica principiile la celula Full Adder Cell, dar vom pierde din vedere situația în care  $C_i = 0$ .

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:



**Simbol :**

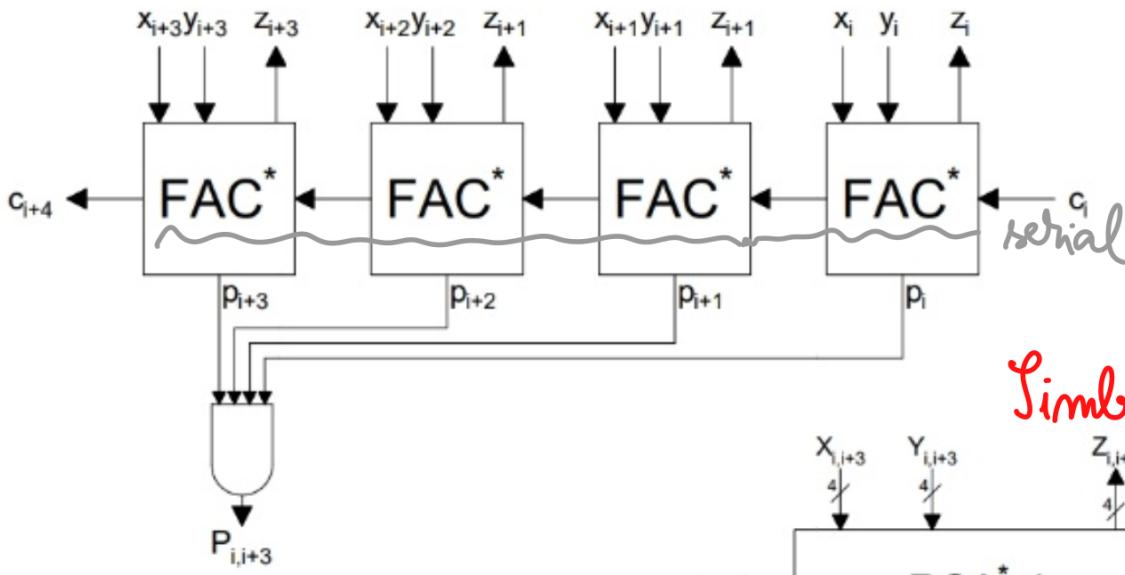


este singura modificare față de celula FAC

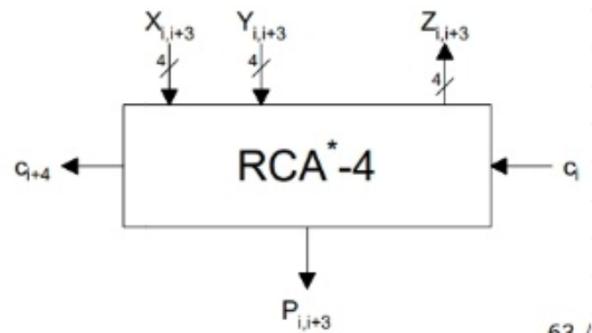
→ bitul de propagare  $p_i$

Utilizând celule FAC\*, se poate construi un segment RCA care generează variabila de propagare la nivelul întregului bloc de ranguri:

## Segment RCA\* pe 4 biți



Simbol:



→ Toate ieșirile de propagare sunt aggregate (de la  $p_i$  la  $p_{i+3}$ )  
Se obține astfel variabila de propagare la nivel de bloc  $P_{i,i+3}$

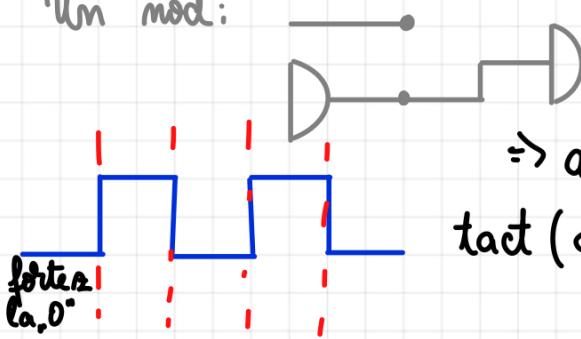
→ Dacă  $c_i = 1$       }       $P_{i,i+3} = 1 \Rightarrow c_{i+4} = 1$

Această metodă de a activa  $c_{i+4}$  este mai rapidă decât dacă am lăsa transportul să se propage serial (el săricum o va face, deoarece există conexiuni de la un tranzistor altul)

**CMOS pre-discharging**

→ celălalt element care poate aduce accelerarea  
→ modalitate de proiectare a unei arhitecturi prin care un set de moduri sunt aduse la valoarea 0 înainte de începerea calculelor

Un mod:

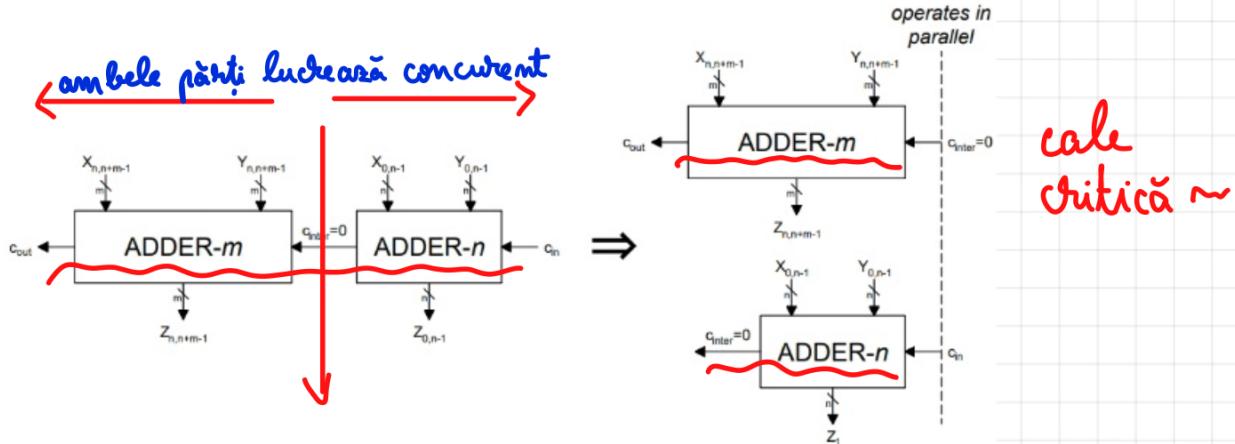


⇒ această operatie se bazează pe semnalul de tact (devin "0" de la frontul crescător)

## Pentru sumatorul Carry Skip (CSKA)

→ toate semnalele transport inter-rang sunt prevenite cu CMOS pre-discharge

- ⇒ dacă valoarea finală, corectă, a unui transport este 0, ea va fi calculată corect de la momentul inițial.



Sumator pe  $n + m$  biti

$c_{\text{inter}} = \text{transport intermediar } (c_m) \rightarrow 0 \Rightarrow c_{\text{inter}} \text{ a fost corect de la } 0!$

Pentru ca acest transport intermediar să aibă valoarea 1, în condiția în care toate transporturile anterioare sunt încă în curs de calcul, trebuie să am:

$$x_{m-1} = y_{m-1} = 1 \Rightarrow c_{\text{inter}} = 1$$

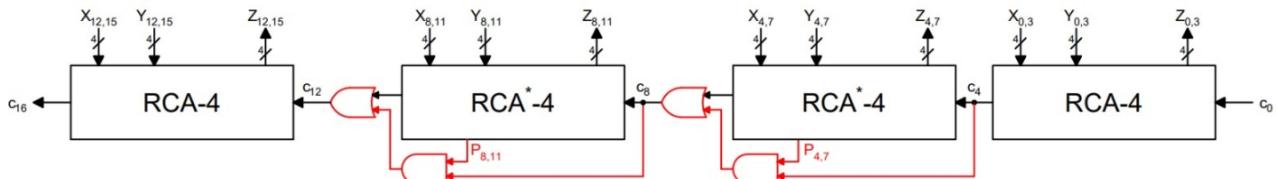
! În orice altă situație trebuie să aștept să ajung la calcularea lui

Cele 2 sumatoare lucrătoare concurrent devorează ADDER-m nu mai trebuie să aștepte 2nd pîmă când îl obțin pe  $c_{\text{inter}}$  (dacă valoarea lui corectă este 0)  
 $\Rightarrow$  Calea critică nu mai este  $2(m+n)d$  ci

$$2 \cdot \max(m, n)d \Rightarrow \text{performanță scăzută}$$

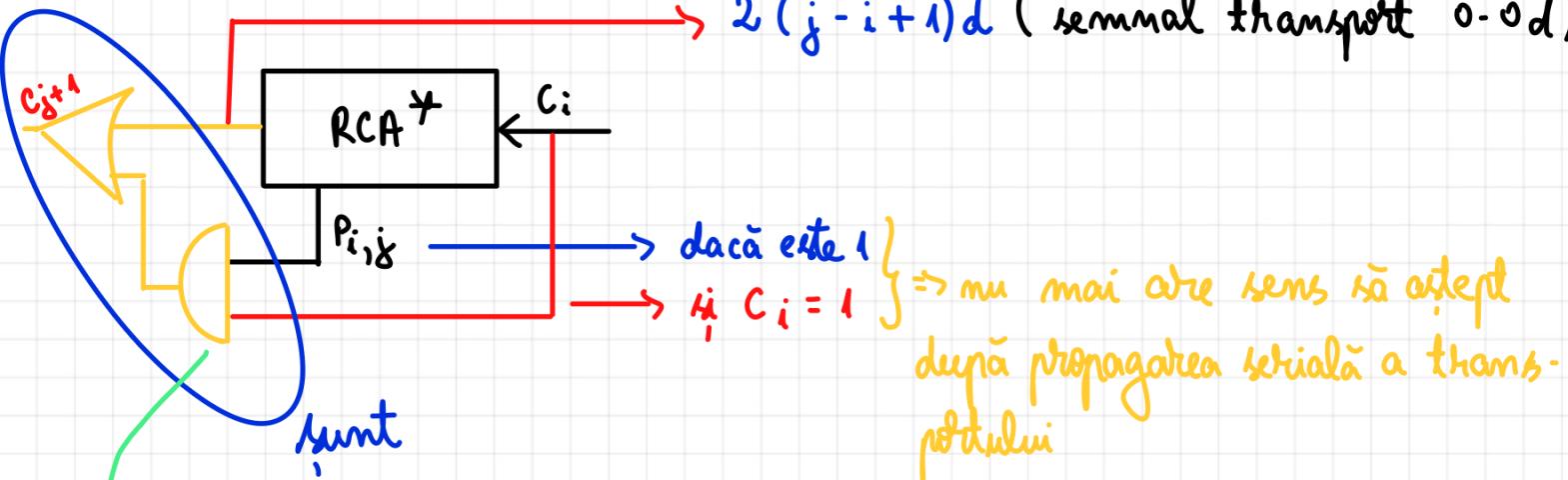
Arhitectura unui sumator Carry Skip (CSKA) pe 16 biti

→ Pentru blocurile RCA\* este utilizată logica de şunt („ship logic”, trasată cu roşu) → facilitează propagarea transportului peste întregul bloc



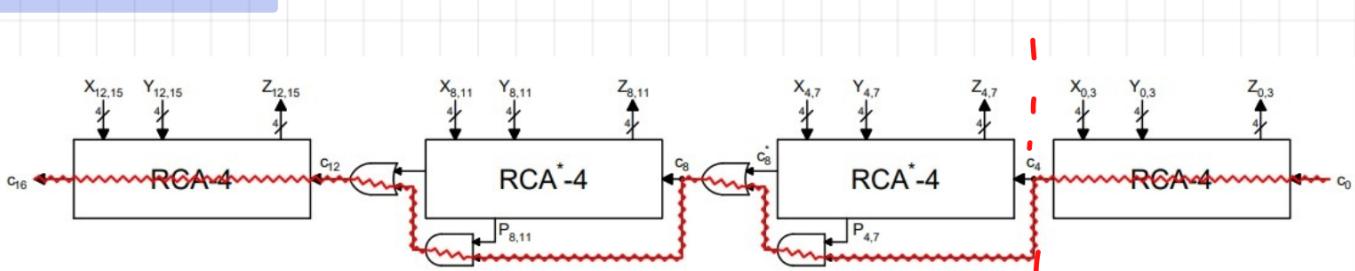
Logica de şunt

$2(j-i+1)d$  (semnal transport 0-0d)



! Dacă  $C_i = 0 \Rightarrow$  transportul se va propaga serial (există, fiind caz defavorabil)  
 $\rightarrow +1d$  dacă  $P_{i,j} = C_i = 1 \Rightarrow$  Pot obține  $C_{j+1}$  după  $+2d$

Calea critică



Teorema: de ce folosește calea critică logica de şunt?

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 concurrent ⇒ cale critică mai scurtă)

→ în mod cert, lungimea ar fi mai mare dacă propagă în mod serial

Semnalul de transport  $c_8$  are expresia:

$$c_8 = \frac{c_8^*}{+1d} + \frac{p_{4,7} \cdot c_4}{+1d} \rightarrow \text{preferat pt. viteză}$$

$\ominus$  considerăm 1

$\diamond$   $c_4 = 1$  pentru a nu desparti sumatorul în două părți care vor opera concurrent

Dar și  $p_{4,7}$  ar trebui să aibă valoarea 1. Demonstrăm că învechii nu se poate.

$\diamond$  Pentru a evita logica de sunt, se impune  $p_{4,7} = 0$ . Însă:

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

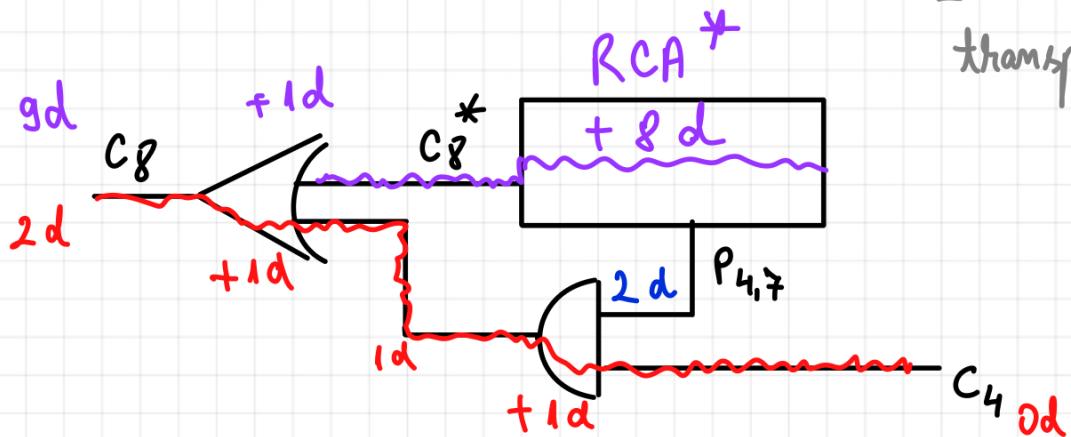
$\Rightarrow \exists$  minim un  $p_i$ , cu  $i \in [4, 7]$ , pentru care  $p_i = 0$

—————> Pentru acel indice se poate scrie:

$$p_i = x_i + p_i = 0 \Rightarrow x_i = y_i = 0$$

$\Rightarrow c_{i+1} = 0$  (indiferent de  $c_i$ )

[Orice  $c_i$  ar avea, nu pot genera transport,  $s_{max} = 0 + 1 = 1 \downarrow c_i$ ]

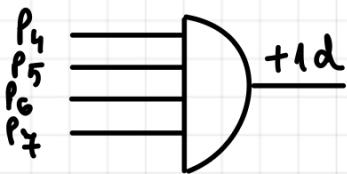


Logica de sunt oferă propagarea mai rapidă a transportului:

- ▶ prin sunt, 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

Întârzierea  $p_{4,7}$  nu a fost luată în considerare deoarece propagate-urile se calculează după  $1d$  (dim poarti SAU)  $\xrightarrow{x_i} \xrightarrow{y_i} P_i$

$P_i$ -urile sunt apoi agregate printr-o poartă și



$\Rightarrow P_{4,7}$  se obține la  $2d$

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

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

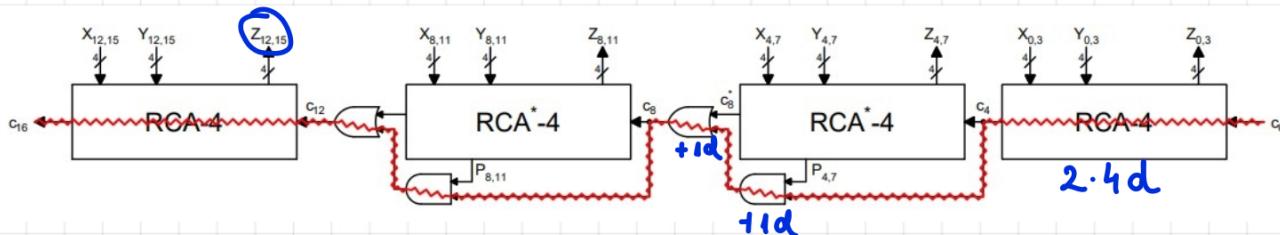
$$c_8 = c_8^* + P_{4,7} \cdot c_4 \xrightarrow{=1} \text{transport de ieșire}$$

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

Calculare întârziere → poate fi problema / aplicatie la examen  
 → Prinul și ultimul segment nu au logica de skip pentru a putea avea o comparație echivalabilă cu RCA.

Dacă am avea sunt, transportul s-ar calcula extrem de rapid, iar înlocuitorul sumă s-ar calcula mult mai târziu.

⇒ Pt. a nu avea cazuri extreme se elimină sunt dim MSR, LSR



$$b_{CSRA-16}^{\Sigma} = \frac{8d}{c_4} + \frac{2d}{c_8} + \frac{2d}{c_{12}} \Big| + \frac{8d}{\cancel{z}_{15}} = 20d$$

RCA 2nd pt  $\Sigma$

$$b_{CSRA-16}^{Cout} = \frac{8d}{c_4} + \frac{2d}{c_8} + \frac{2d}{c_{12}} \Big| + \frac{8d}{c_{16}} = 20d$$

$\cancel{z}_{15} \rightarrow 2 \cdot 4d = 8d$

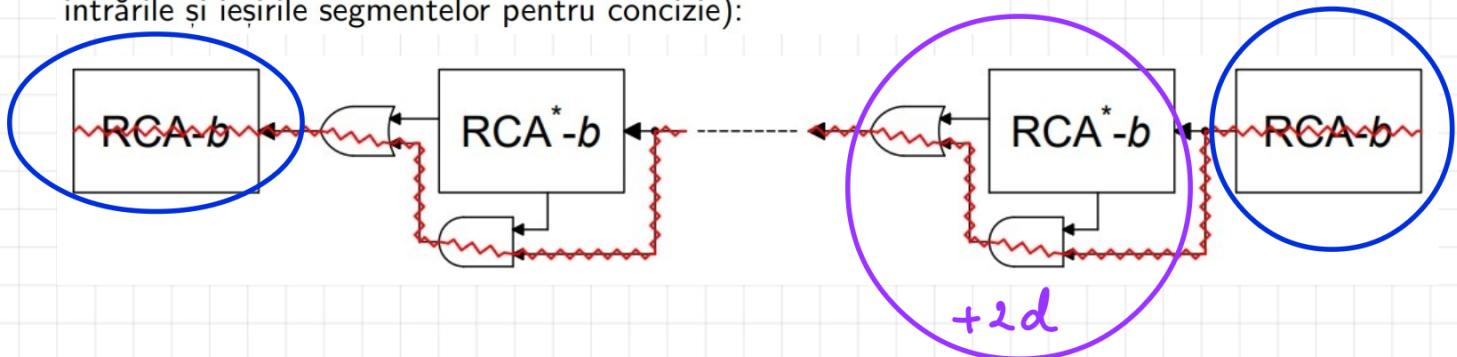
## Determinarea lungimii optime a segmentelor RCA

→ Se consideră o structură CSkA având:

- segmente RCA de lungime  $b$  biți;
- segmentul MS și cel LS nu au logica de sunt significant - $11$
- operații cu  $m$  biți, cu  $m = k \cdot b$ ,  $k \in \mathbb{N}$

## Lungimea segmentelor RCA afectează întârziile.

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):



→ Desigur nu este vorba de segmentele cu sunt (aici, în cazul de-favorabil se va trece prin sunt, indiferent de lungime)

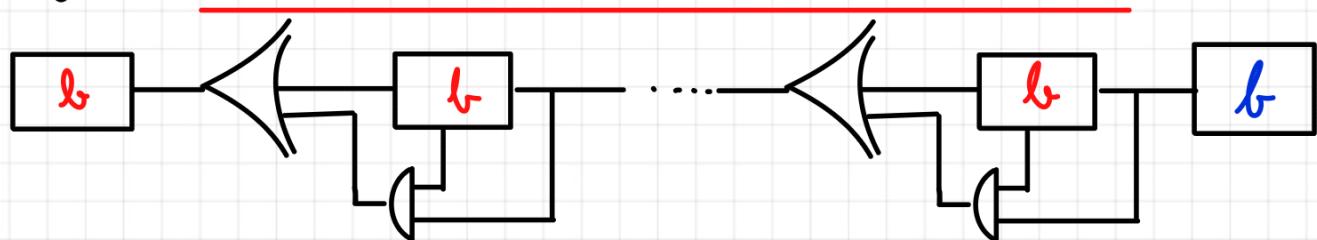
→ Contrează MS și LS segment + câte blocuri  $RCA^*-b$  avem

$$m - \text{bits} \quad m = k \cdot b$$

$$\text{segment de } b \text{ biți} \quad \Rightarrow \frac{m}{b} \text{ segmente}$$

$\frac{m}{b} - 2$  au sunt

$\underline{\underline{2}}$  → fără sunt



$$t_{CSkA-m}^{z/cout} = \frac{2 \cdot b \cdot d}{RCA} + \underbrace{2 \cdot \left( \frac{m}{b} - 2 \right) d}_{\substack{\text{sunt} \\ \text{seg. cu} \\ \text{sunt}}} + \frac{2bd}{RCA} = \left( 4b + \frac{2m}{b} - 4 \right) d$$

→ Lungimea optimă,  $b_{opt}$ , reprezintă un punct de extrem local al funcției  $b_{CSkA-m}^{2/cout}$ , deci:

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

$$\Rightarrow b_{opt} = \frac{\sqrt{2m}}{2} \quad \text{cu latență optimă } b_{CSkA-m}^{2/cout} = 4(\sqrt{2m}-1)d_{min}$$

Ex:

$$M=32 \Rightarrow 4b + \frac{64}{b} - 4$$

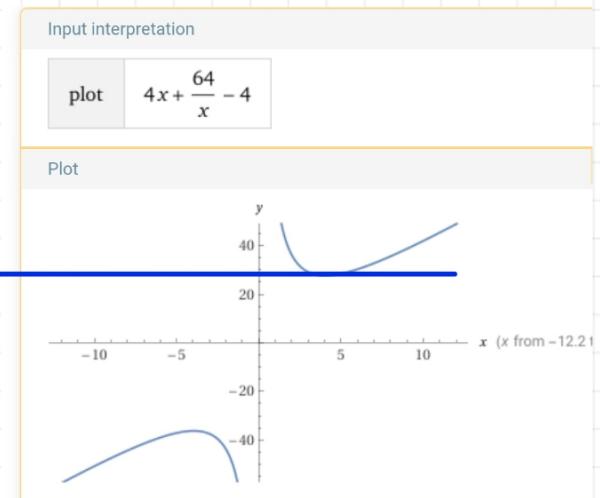
Demonstratie,

$$\Rightarrow \frac{\partial b}{\partial b} = 0$$

$$4 - \frac{2m}{b^2} = 0 \Rightarrow \frac{2m}{b^2} = 4$$

$$\Rightarrow b^2 = \frac{2m}{4} \Rightarrow b = \frac{\sqrt{2m}}{2} \quad (\text{luăm doar valoarea pozitivă})$$

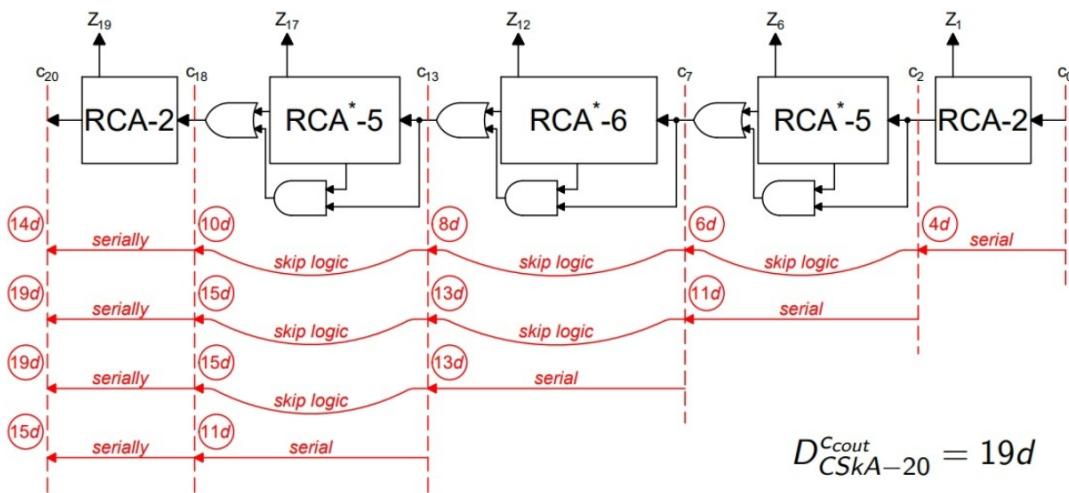
are minimum local



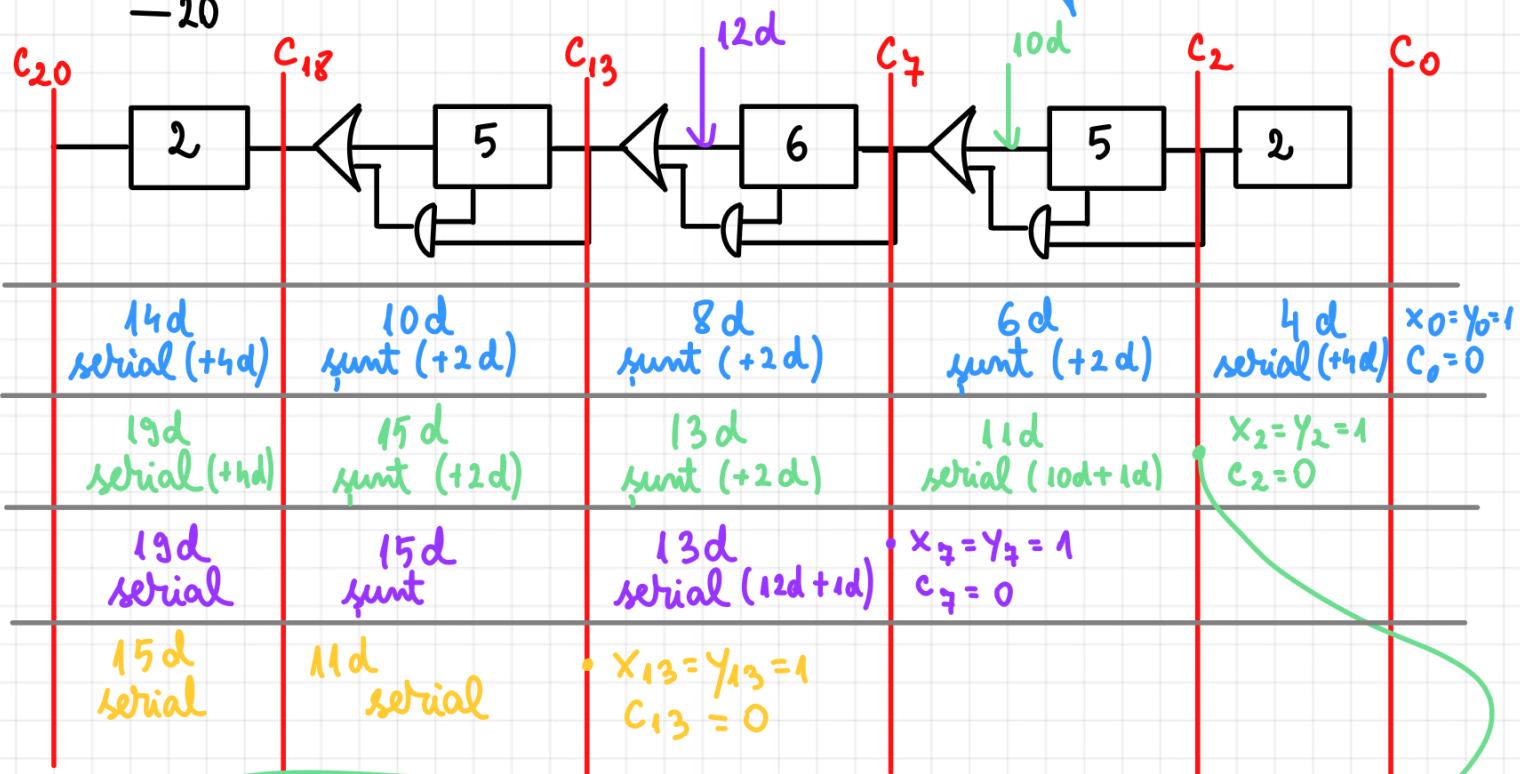
$$\left\{ \begin{array}{l} b_{CSkA-32 \text{ min}}^{2/cout} = 28d \\ b_{RCA-32}^{2/cout} = 64d \end{array} \right.$$

$$= 64d \quad (\text{ca termen de comparație})$$

Determinarea RCA de lungimi variabile → poate apărea la Ex  
→ se consideră operanții pe 20 de biti și structura CSkA următoare:



→ construiesc cazul cel mai nefavorabil



⇒ am împărțit sumatorul în 2 (seg 2, și cele 4 seg din stânga)  
→ sumt dispare

$$\Rightarrow b_{CSRA-20}^{C_{out}} = 19d$$

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).

Pentru sumă, cazul defavorabil se determină evaluând latența maximă a celui mai semnificativ bit sumă pentru fiecare segment  $RCA / RCA^* (\Sigma_{19}, \Sigma_{17}, \Sigma_{12}, \Sigma_6, \Sigma_1)$

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

$$b_{\Sigma_1} = b_{C_0} + \frac{4d}{\text{seg 2 biti}} = (0+4)d = 4d$$

$$b_{\Sigma_6} = b_{MAX}^{C_2} + \frac{2 \cdot 5d}{RCA \ 5} = 4d + 10d = 14d$$

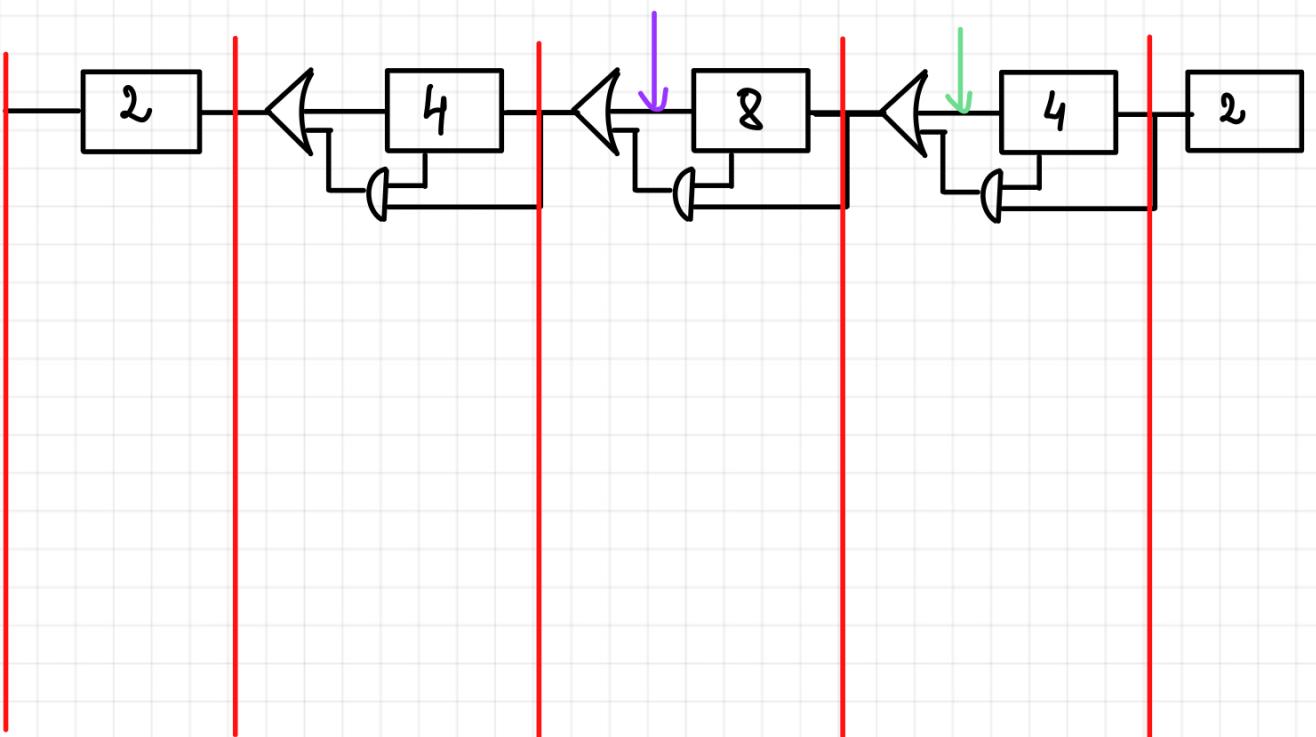
$$b_{\Sigma_{12}} = b_{max}^{C_7} + 2 \cdot 6d = 11d + 12d = 23d$$

$$b_{\Sigma_{17}} = b_{MAX}^{C_{13}} + 2 \cdot 5d = 13d + 10d = 23d$$

$$b_{\Sigma_{19}} = b_{max}^{C_{18}} + 4d = 15d + 4d = 19d$$

$$\Rightarrow b_{CSRA-20}^{\Sigma} = 23d$$

ex curs (de terminat)

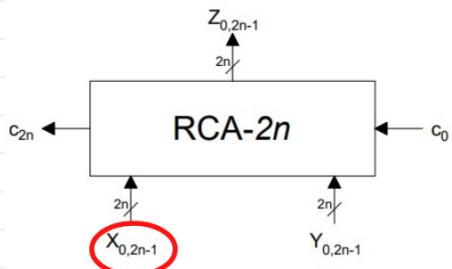


### 2.3.3. Garry Select Adder (CSeA)

→ Bazat pe principiul sumei conditionate prin transport.

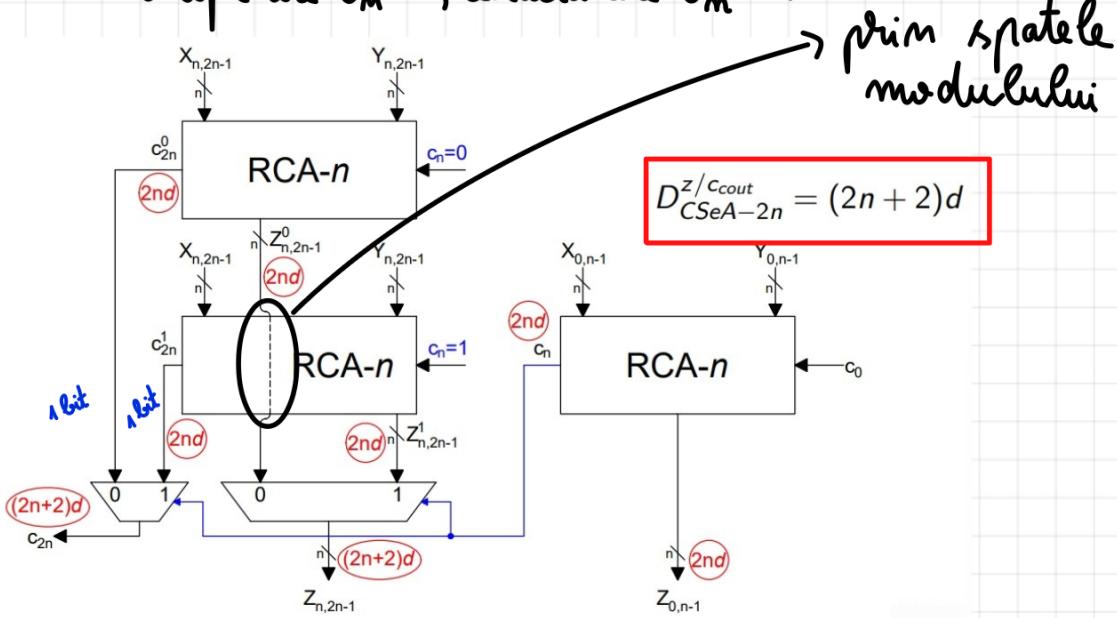
- !  $c_i$  are doar 2 valori posibile  $\begin{cases} 0 \\ 1 \end{cases}$
- Se calculează  $z_i$  și  $c_{i+1}$  în 2 cazuri
  - $\rightarrow (z_i^0, c_{i+1}^0)$ , dacă  $c_i = 0$
  - $\rightarrow (z_i^1, c_{i+1}^1)$ , dacă  $c_i = 1$
- Se va selecta una din cele 2 perechi  $(z_i, c_{i+1})$  de mai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

Construcția unui CSeA permite de la arhitectura unui RCA pe 2m biti.



Construcția CSeA:

- se împarte sumatorul RCA-2m în două jumătăți;
- jumătatea mai semnificativă se duplică
- o copie are  $c_m = 0$ , cealaltă are  $c_m = 1$ .

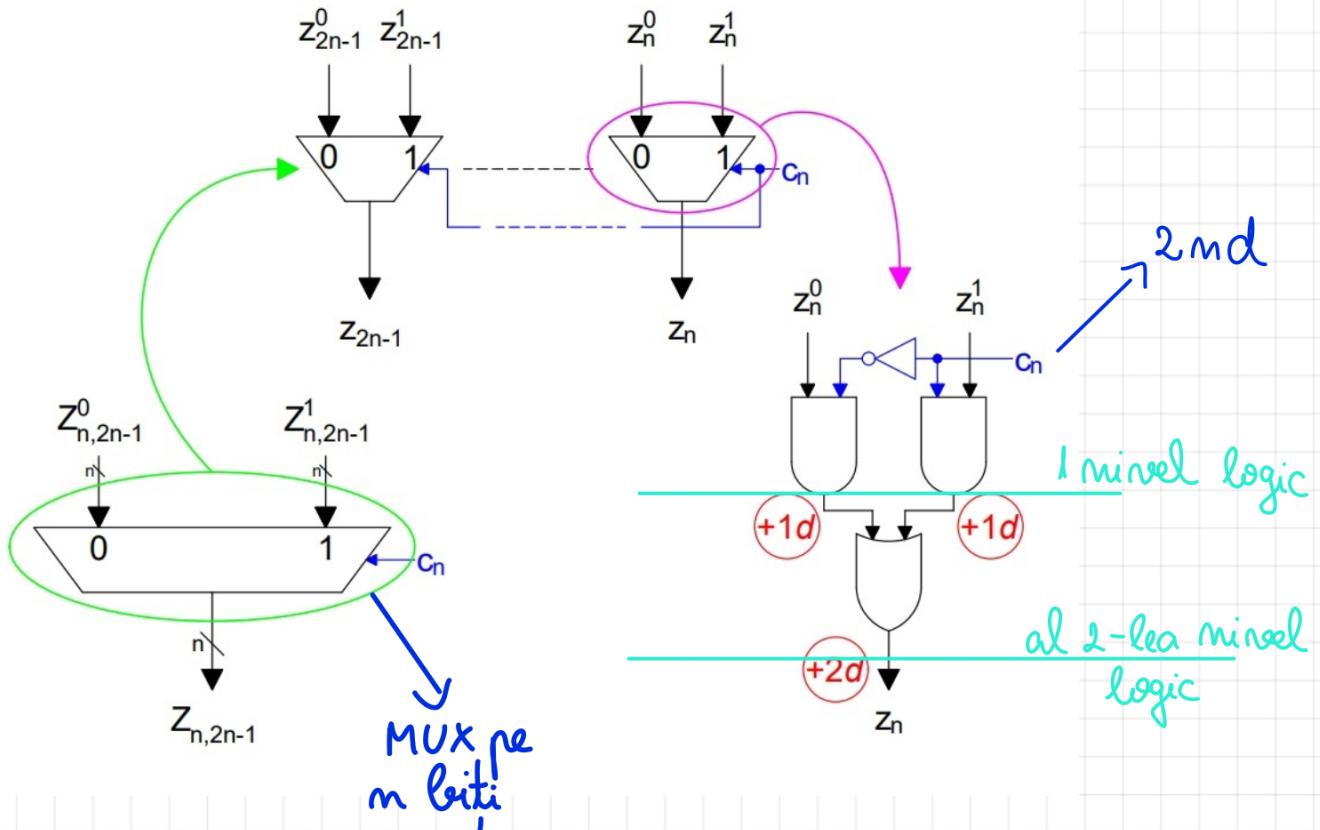


→ toate cele 3 segmente lucrează în mod paralel

! Dacă  $m$  este mare  
 $b_{CSeA-2m}^{z/\text{out}} = (2 \cdot m + 2) d$

| îngrijătăște timpul pentru un RCA pe  
 m biti

Detalii întăriere  $m$  MUX-uri → lucrează în paralel



Optimizarea ariei CSeA

→ Se permite de la tabelul de adevăr al semnalului  $c_{2m}$

? Niciodată  $c_{2m}^0$  nu poate fi mai mare decât  $c_{2m}^1$

↳ deoarece tabelul de adevăr folosește don't care pentru cazuri imposibile

→ adună mîte operanți cu 1 mai mult

$$\begin{array}{c} \xrightarrow{\text{la } c_{2m}^0 : x_{m,2m-1} + y_{m,2m-1} + 0} \\ \xrightarrow{\text{la } c_{2m}^1 : x_{m,2m-1} + y_{m,2m-1} + 1} \end{array} >$$

Tabelul se deduce din dim desen, dim logica MUX 2:1

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

2:1 Multiplexer

86



Truth Table

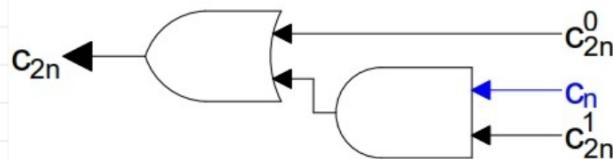
$S_0$	$I_0$	$I_1$	$Y$
0	0	X	0
0	1	X	1
1	X	0	0
1	X	1	1

## Mimimizare

$c_{2n}$	$c_n^0$	$c_n^1$	$c_{2n}$
0	0	0	1
1	0	1	1

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

$\Rightarrow$  transportul de ieșire  $C_{2n}$  va fi generat astfel:

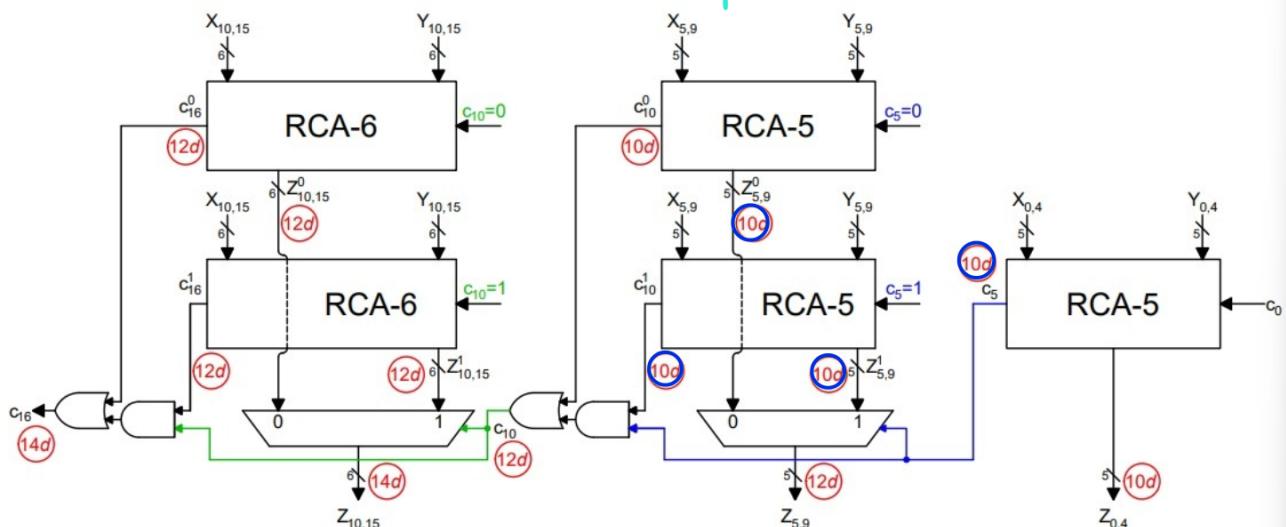


Segmente RCA de lungimi variabile

$\rightarrow$  operații pe 16 biti  $\longrightarrow 6+5+5$

$\rightarrow$  structură CS<sub>e</sub>A:

duplicat



Dim cauza duplicării RCA-5  $c_{10}$  și  $Z_{5,9}$  se obțin cu 2d mai târziu.  $\Leftrightarrow$  I rang RCA  $\Rightarrow$  următorul rang va avea cu un bit în plus

La MUX-ul de la RCA-5, toate intrările ajung în același timp

$\Rightarrow$  Dacă ar vrea să continuă arhitectura ilustrată, ar introduce o grupare RCA-7

(are întârziere 14d)

Latenta curentă:  $b_{CSE-A-16} = 14d$

Dacă-l spărgeam în 2:  $(2 \cdot 8 + 2)d = 18d$

? Se pot construi structuri CS<sub>e</sub>A multi-level

erf curs (de terminat)

