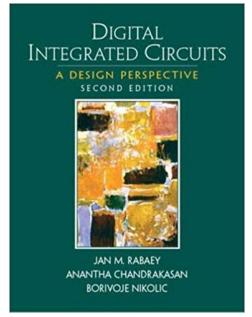
Microeletrônica

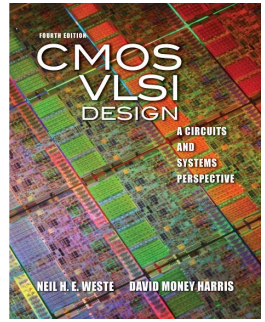
Aula #9 → Circuitos Aritméticos

- Professor: Fernando Gehm Moraes
- Livro texto:

  Digital Integrated Circuits a Design Perspective - Rabaey

  C MOS VLSI Design - Weste





Revisão das lâminas: 27/outubro/2024

Motivação para importância de circuitos aritméticos

Fonte: Benchmark Analysis of Representative Deep Neural Network Architectures (2018)

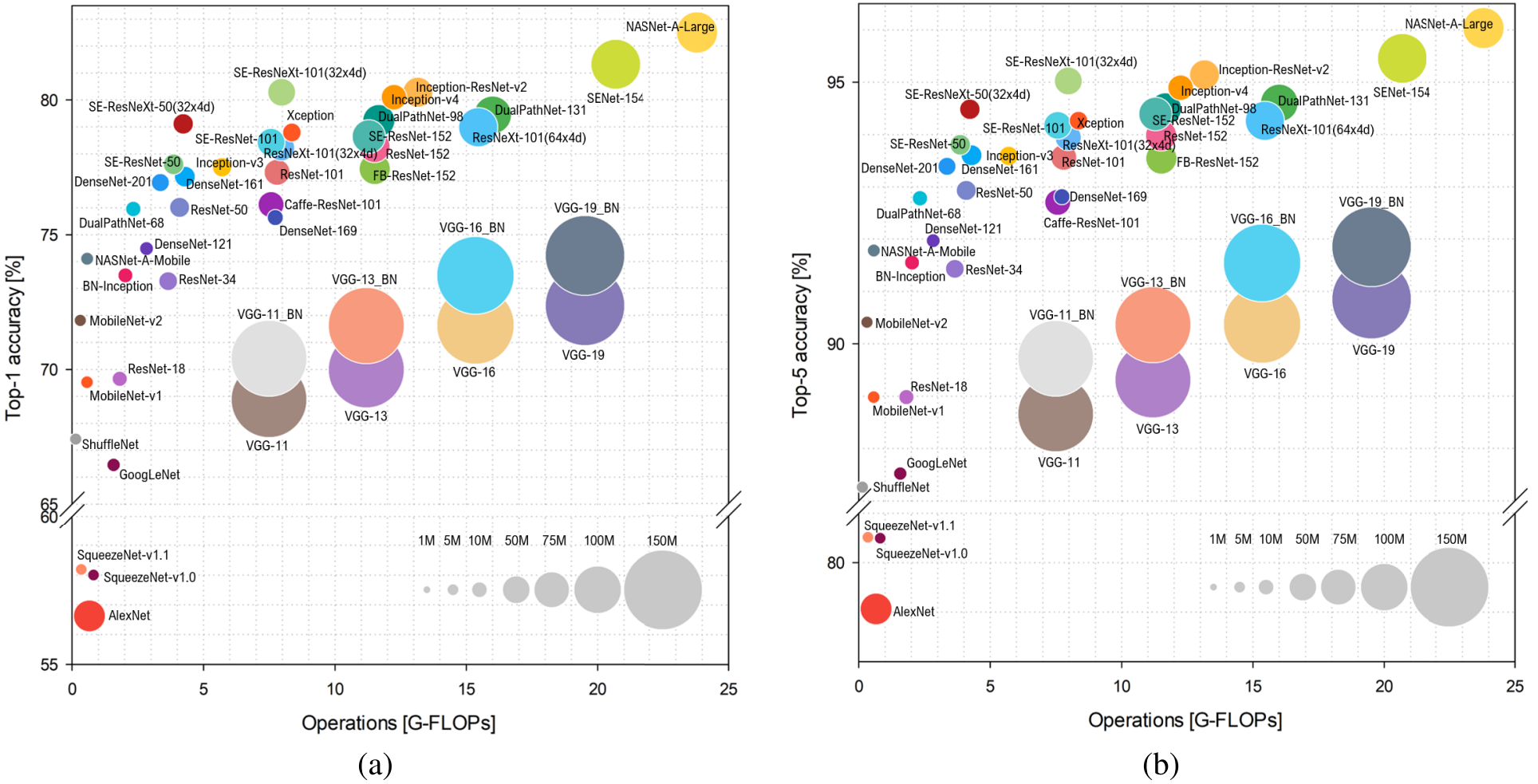


FIGURE 1. **Ball chart reporting the Top-1 and Top-5 accuracy *vs.* computational complexity. Top-1 and Top-5 accuracy using only the center crop versus floating-point operations (FLOPs) required for a single forward pass are reported. The size of each ball corresponds to the model complexity. (a) Top-1; (b) Top-5.**

Motivação para importância de circuitos aritméticos

Fonte: A 0.32–128 TOPS, Scalable Multi-Chip-Module-Based Deep Neural Network Inference Accelerator With Ground-Referenced Signaling in 16 nm (2020)

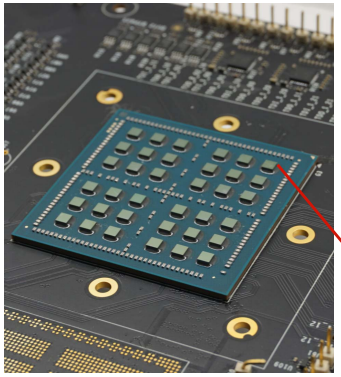


Fig. 16. Prototype system connects 36 chips on the package.

Interposer com 36 chiplets

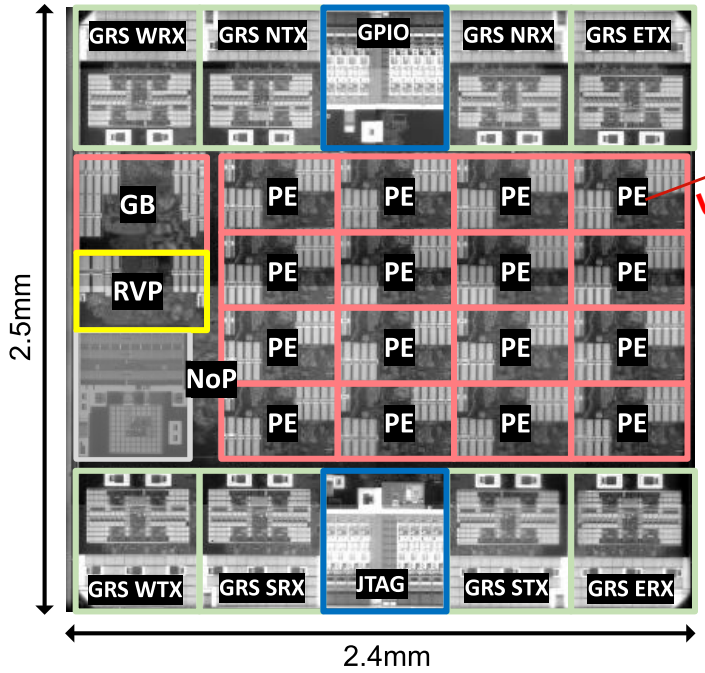


Fig. 14. Chip micrograph annotated with the floorplan of the chip.

Unidades aritméticas

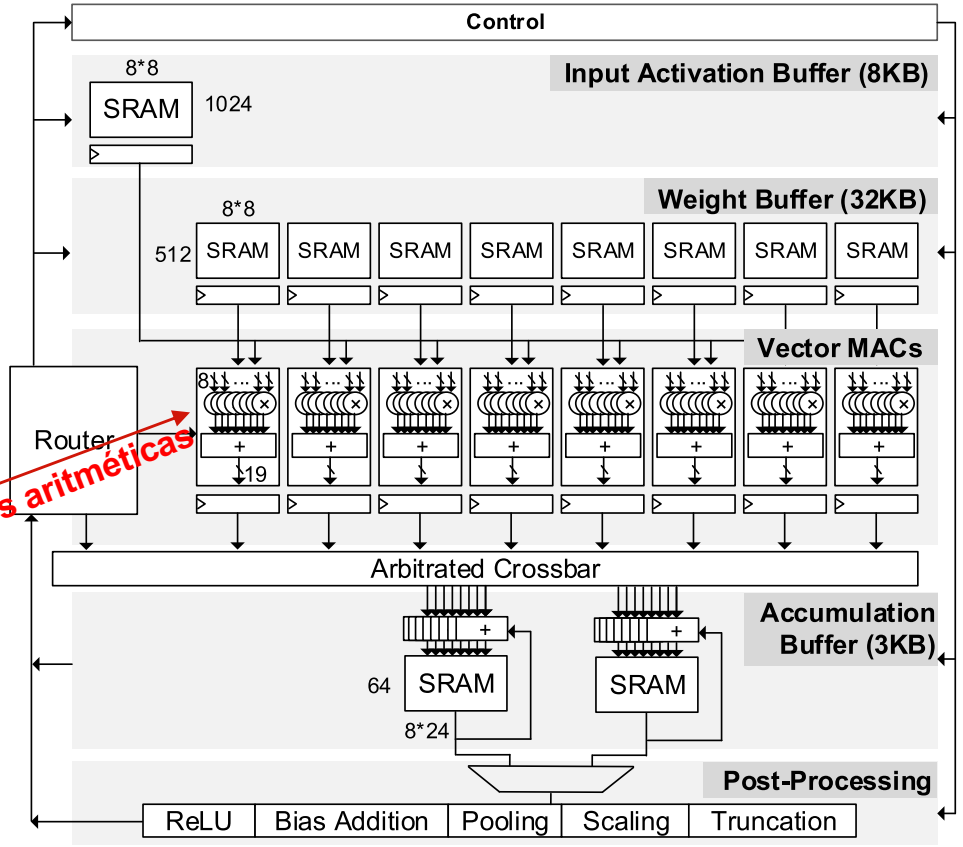
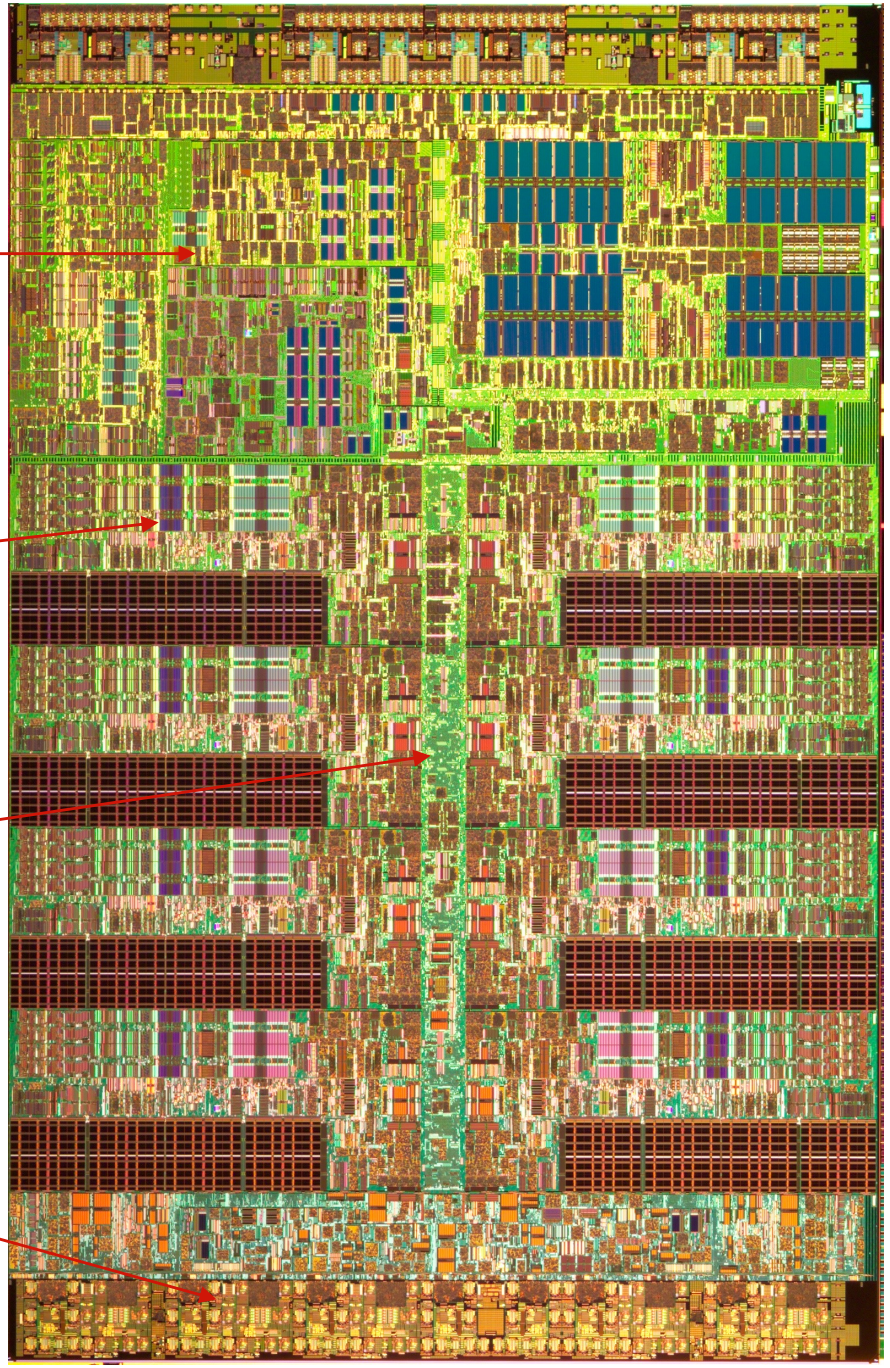


Fig. 8. Architecture of the PE.

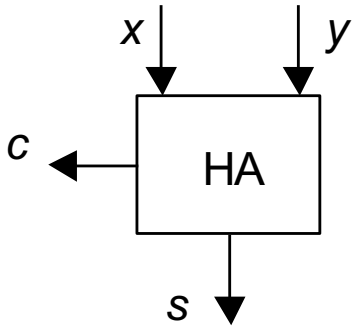
Cell Processor (IBM)

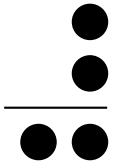
- 1 PowerPC Processor Element
  - Funções de controle
  - Sistema Operacional
- 8 Synergistic Processor Elements
  - ArquiteturaSIMD RISC
  - Processamento de alto desempenho
- Element Interconnect Bus
  - Rede intra-chip de alto desempenho
- Dispositivos de entrada e saída
  - Memory Interface Controller
  - I/O Controller



Somador: HA e FA

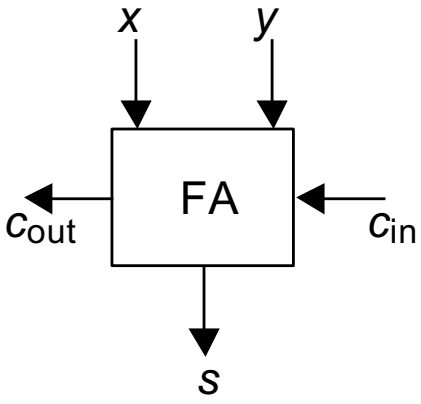
Inputs		Outputs	
x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

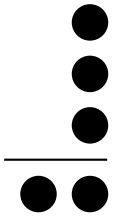




*Half-adder (**HA**): Truth table and block diagram*

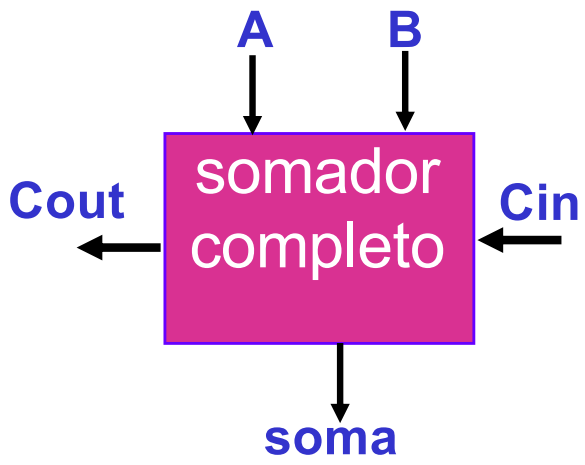
Inputs			Outputs	
x	y	c_{in}	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1





*Full-adder (**FA**): Truth table and block diagram*

Somador Completo – Full Adder (FA)



Soma:

a\bc	00	01	11	10
0		1		1
1	1		1	

Co:

a\bc	00	01	11	10
0			1	
1		1	1	1

A	B	C_i	S	C_o
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

$$S = A.\overline{B}.\overline{C_i} + \overline{A}.B.\overline{C_i} + \overline{A}.\overline{B}.C_i + A.B.C_i$$

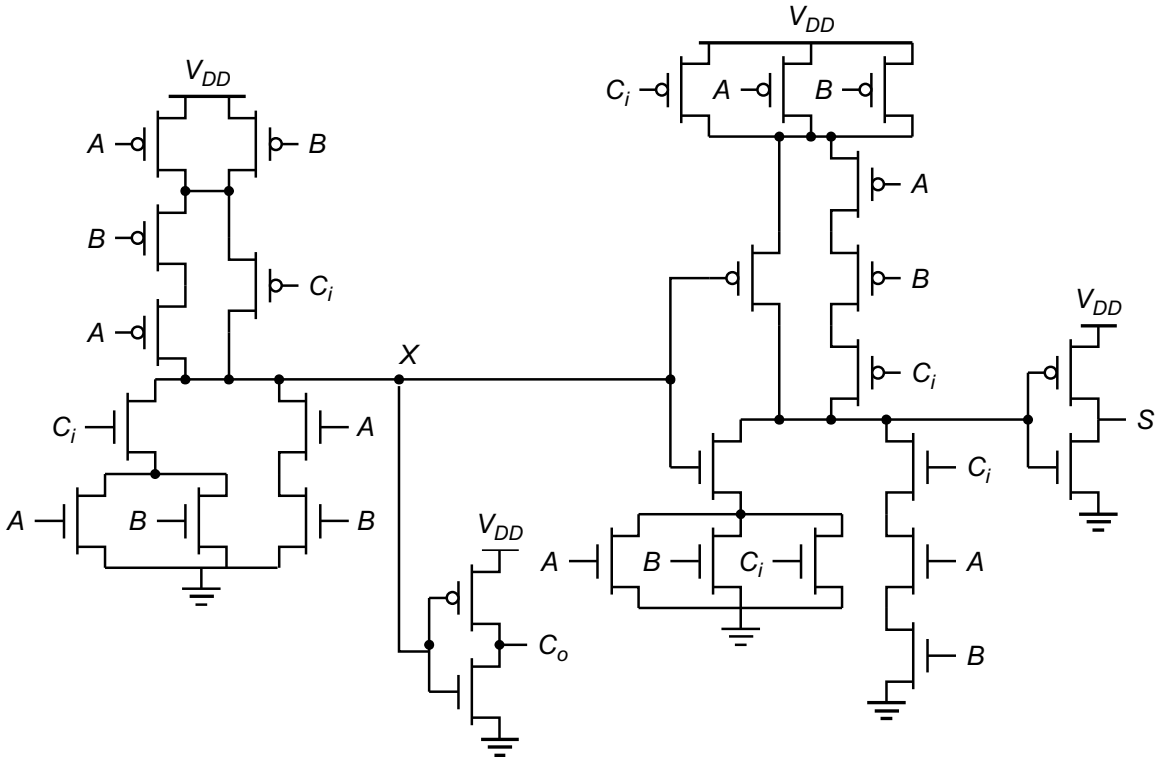
$$S = A \oplus B \oplus C_i$$

$$C_o = A.B + B.C_i + A.C_i$$

FA - Diagrama de transistores

Somador Completo - CMOS Estático Complementar

28 transistores



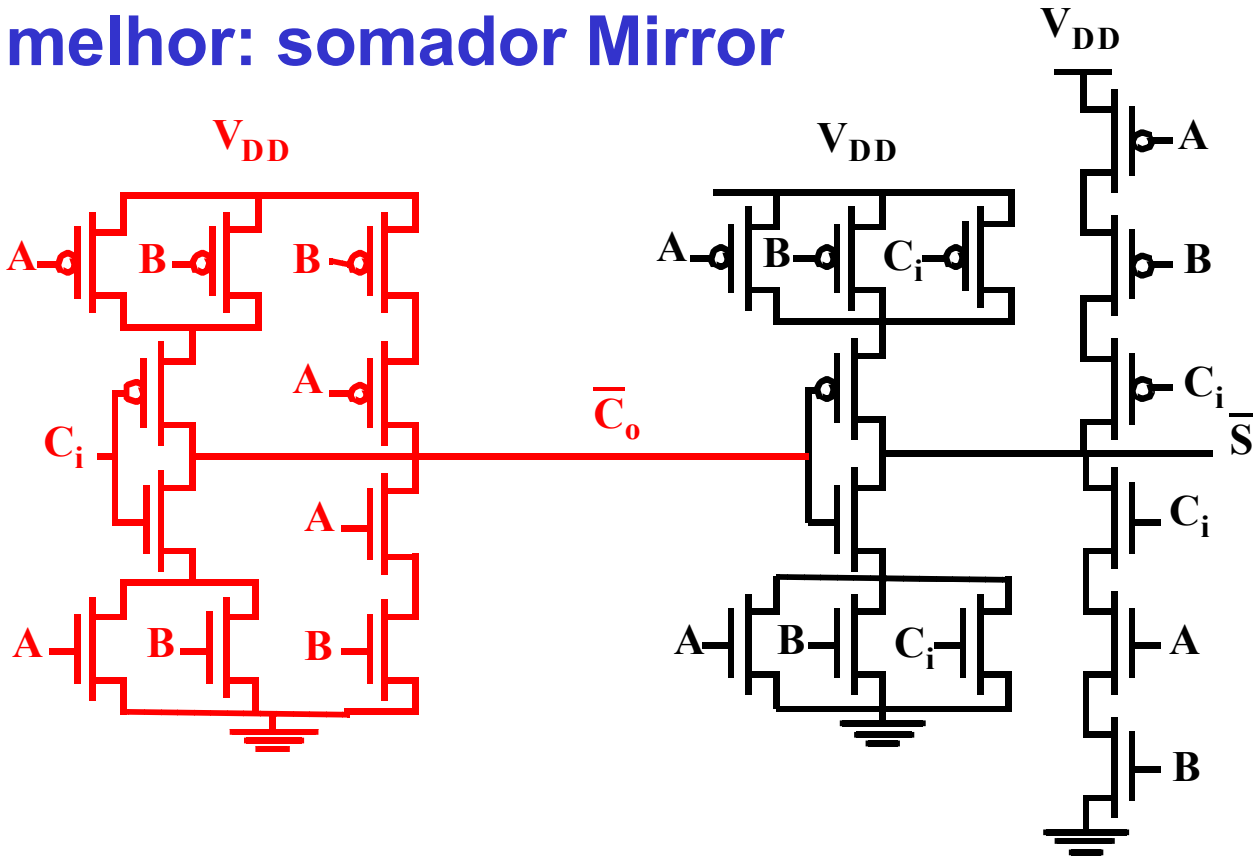
A	B	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$C_o = A.B + B.C_i + A.C_i \qquad \overline{C_o} = \overline{A}.\overline{B} + \overline{A}.\overline{C_i} + \overline{B}.\overline{C_i}$$

$$S = A.\overline{B}.\overline{C_i} + \overline{A}.B.\overline{C_i} + \overline{A}.\overline{B}.C_i + A.B.C_i = (A+B+C).\overline{C_o} + A.B.C_i$$

Diagrama de transistores

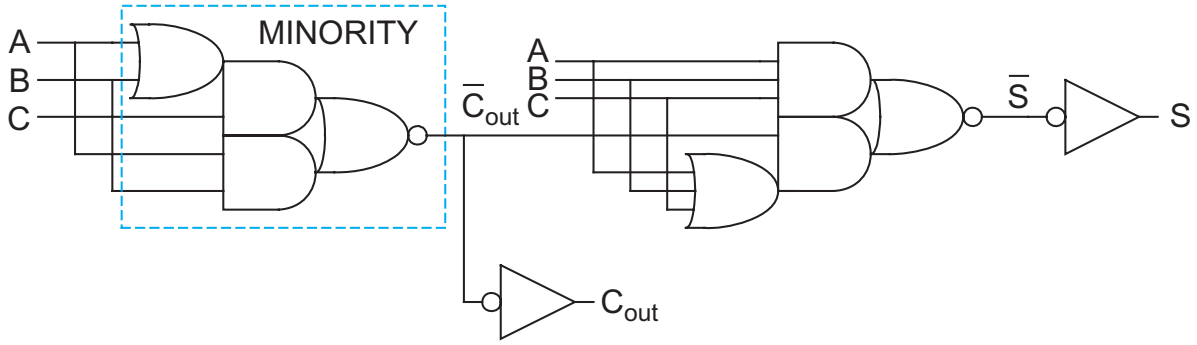
Uma estrutura melhor: somador Mirror



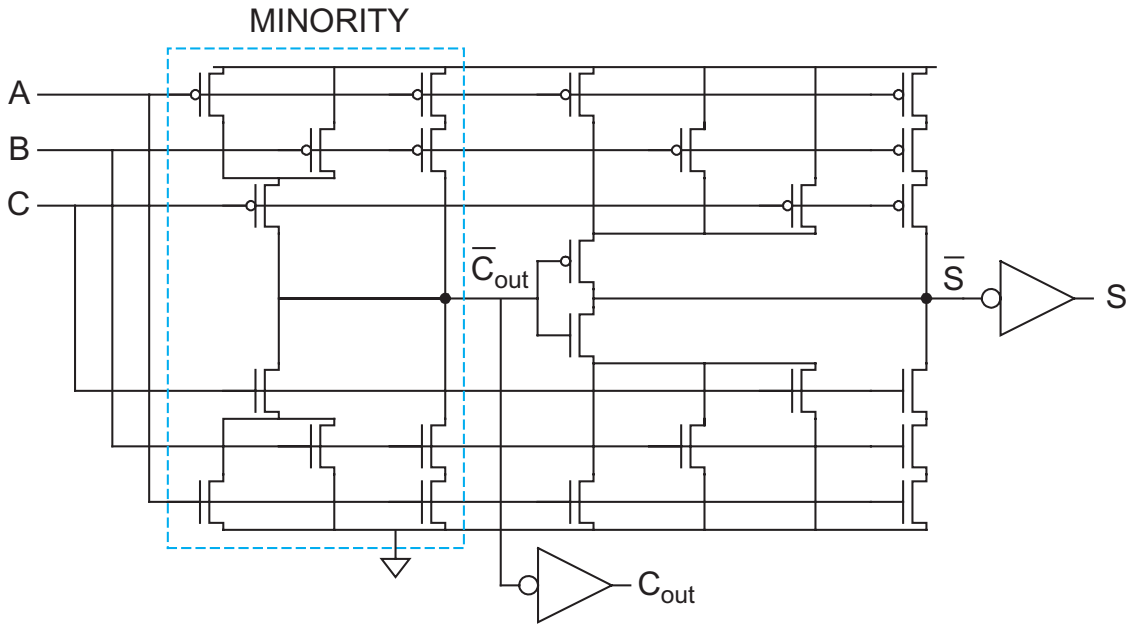
- Planos NMOS e PMOS simétricos
- Tempos de propagação de subida e descida idênticas se os dispositivos NMOS e PMOS estiverem dimensionados corretamente.
- Máximo de dois transistores em série no circuito de geração de *carry*

Diagrama de transistores

Somador Mirror



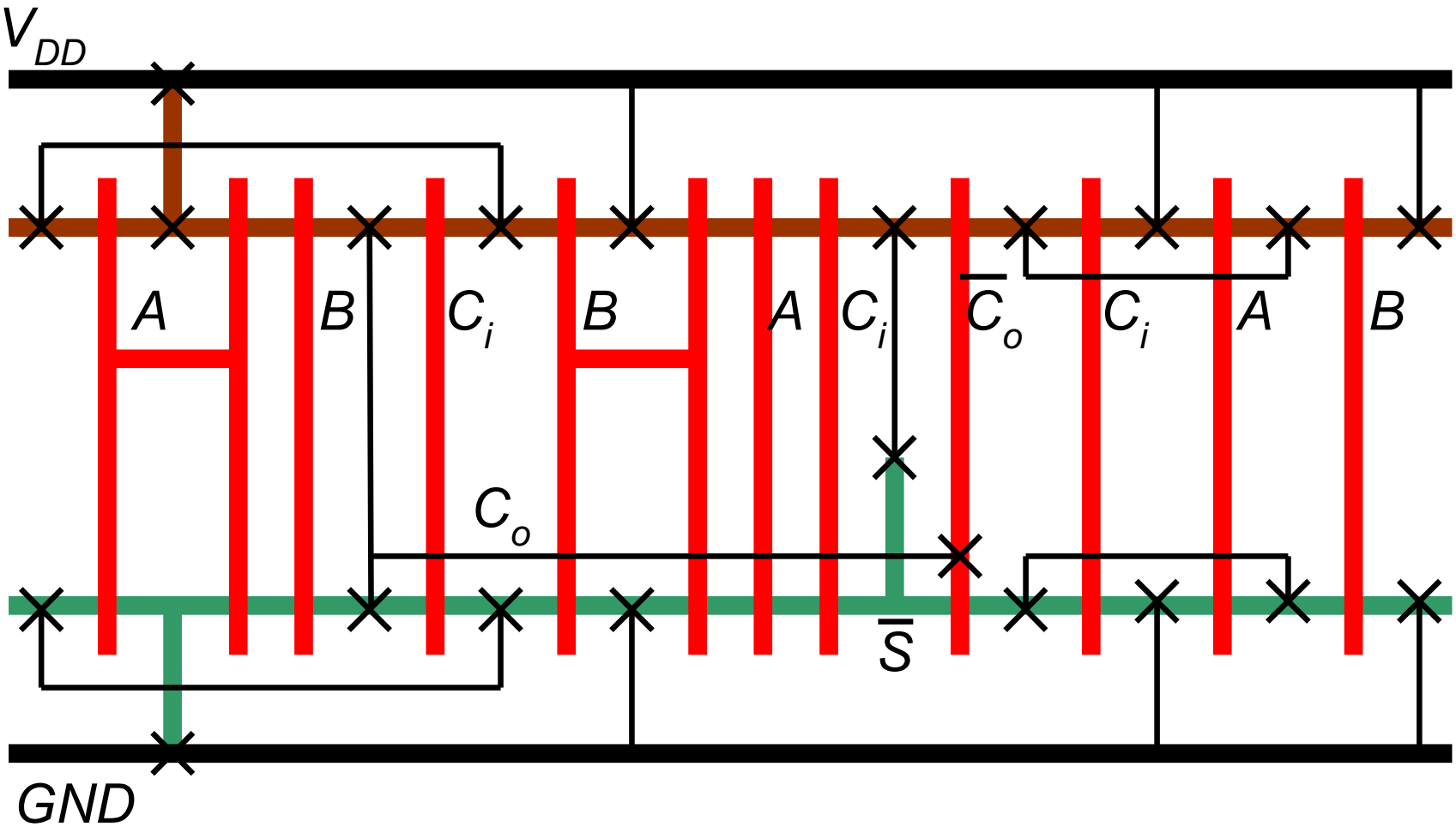
(a)



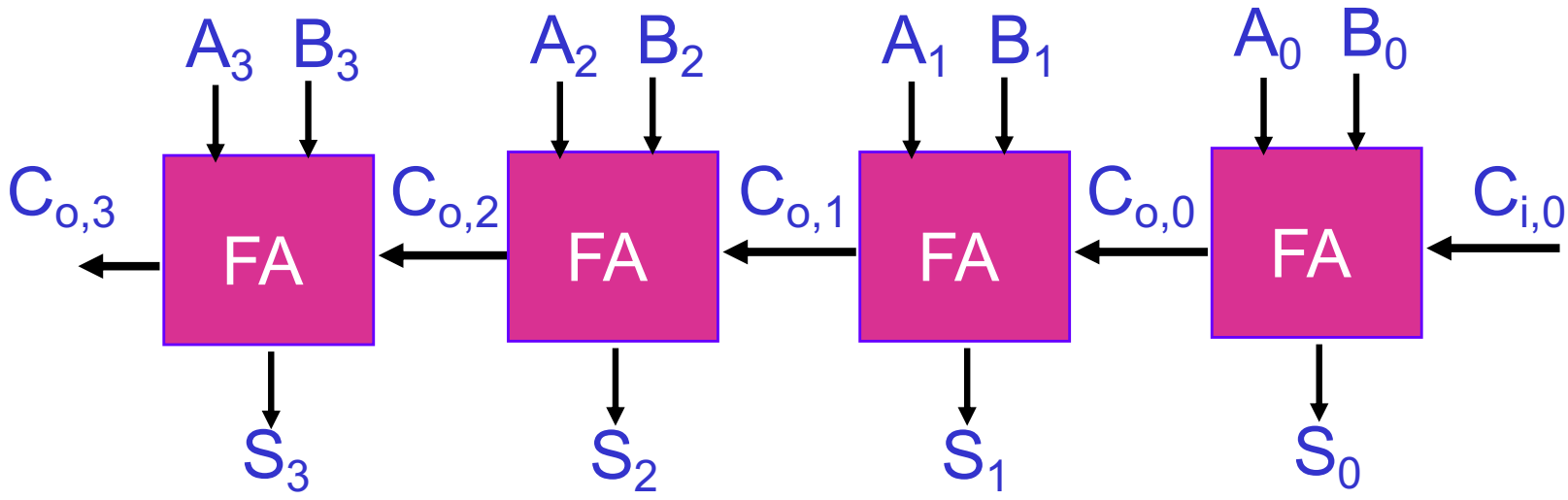
(b)

Weste. P 432

Somador Mirror – Diagrama Stick



(1) RIPPLE CARRY



Pior caso de atraso é linear com o número de bits: $T_d = O(N)$

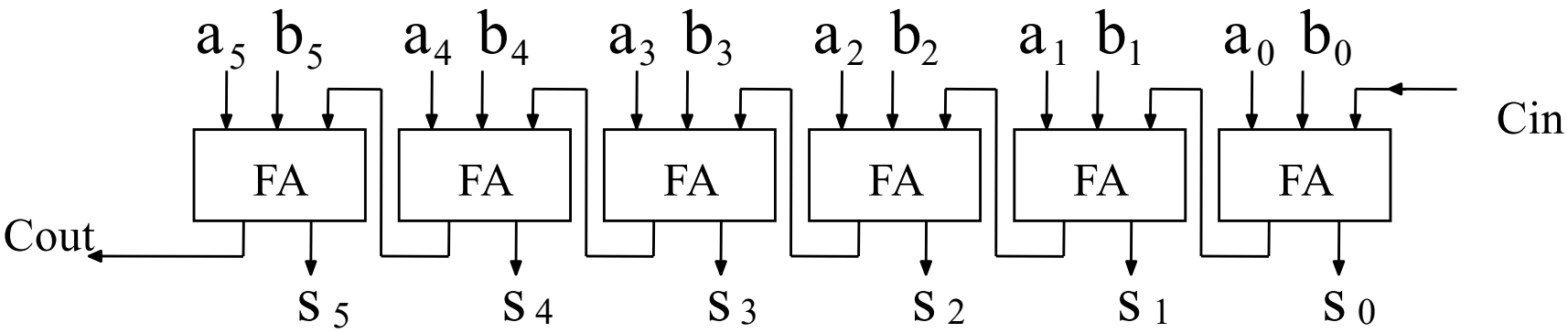
$$T_{adder} = N \cdot t_{FA}$$

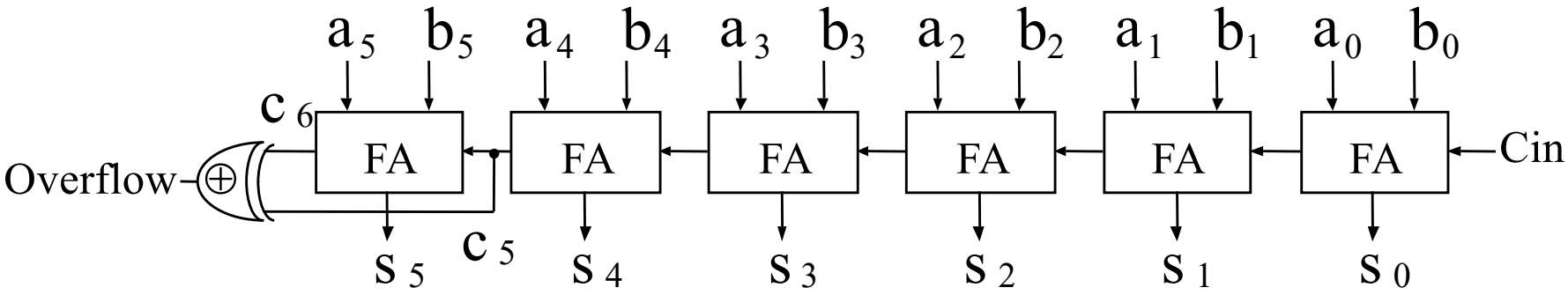
Exemplo:

bits	tFA	tmux
64	50ps	10ps

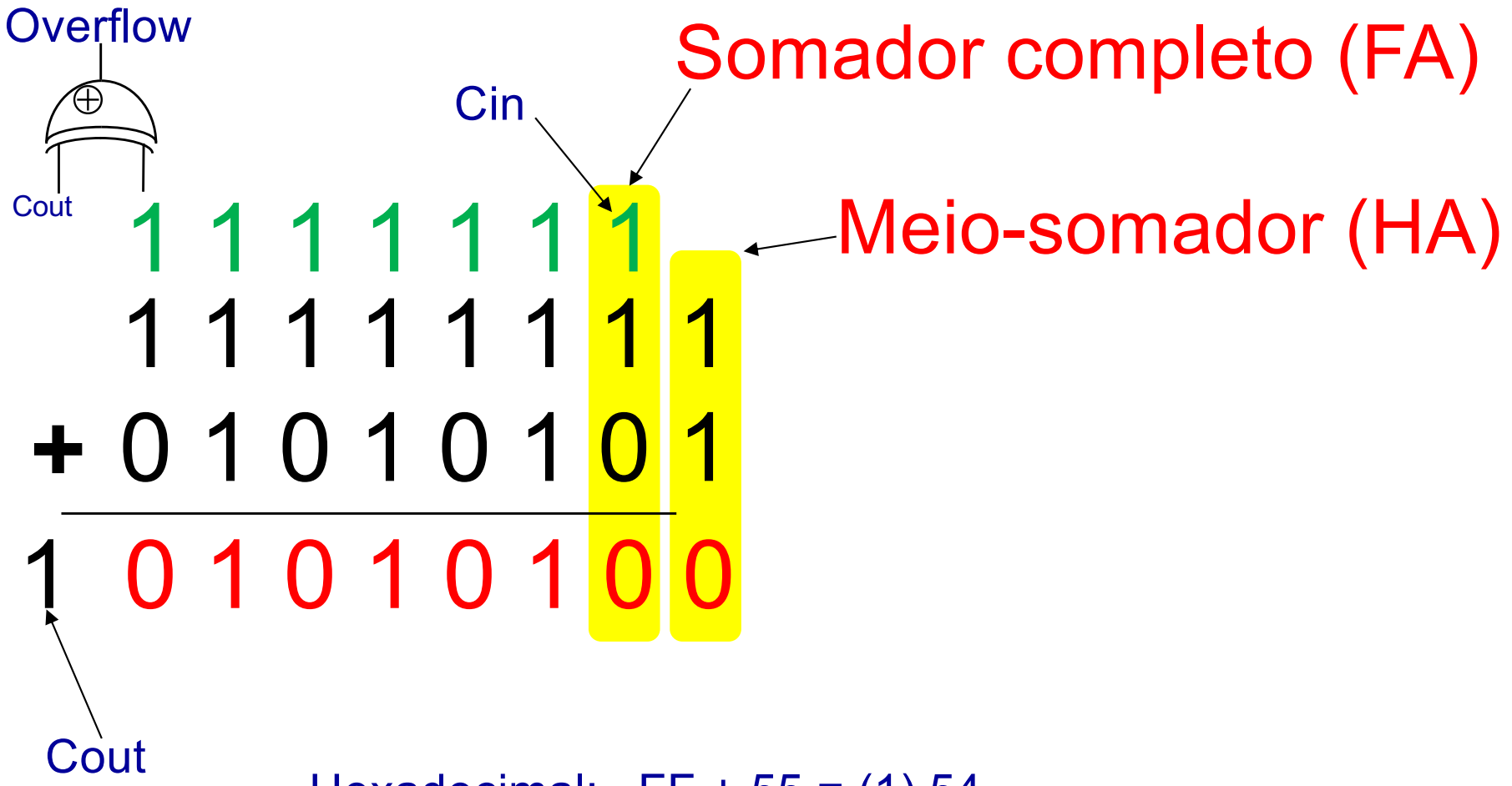
	T	freq (MHz)
Ripple	64*50ps	312,5

Soma - carry out e overflow





Soma - carry out e overflow



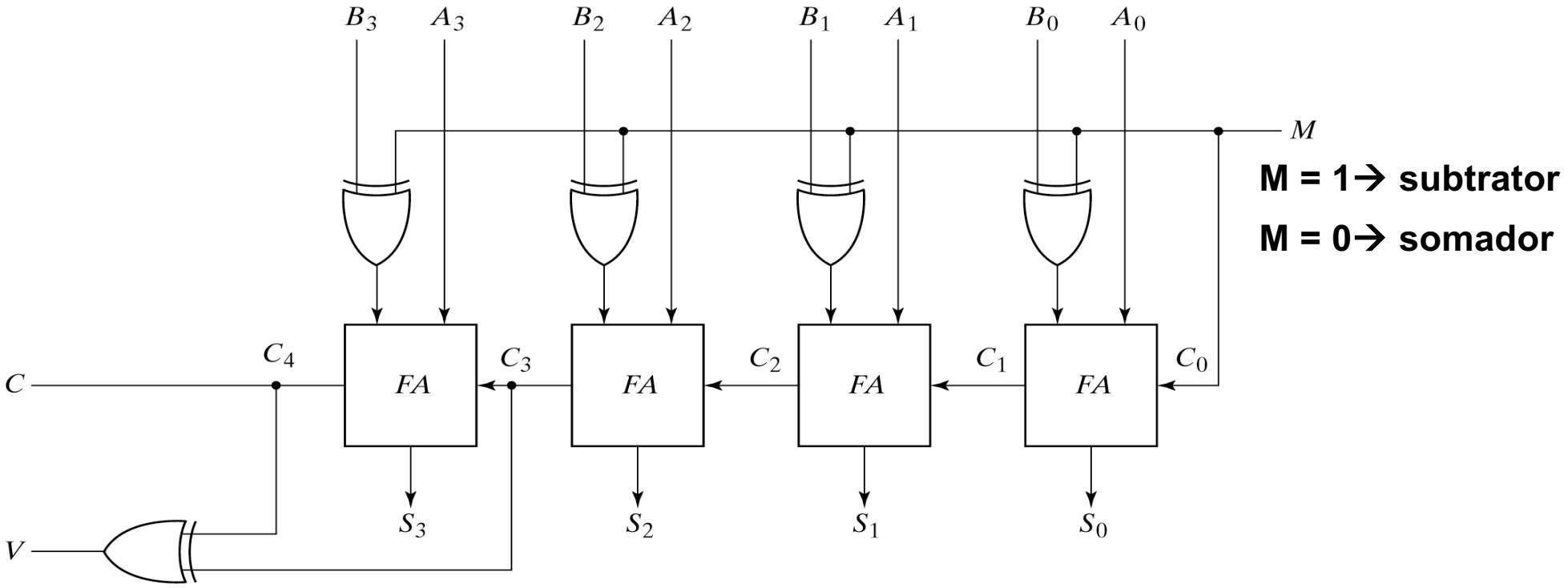
Hexadecimal: FF + 55 = (1) 54

Unsigned: 255 + 85 = 84 (**errado**, Cout=1)

2's comp: -1 + 85 = 84 (**certo** – Overflow=0)

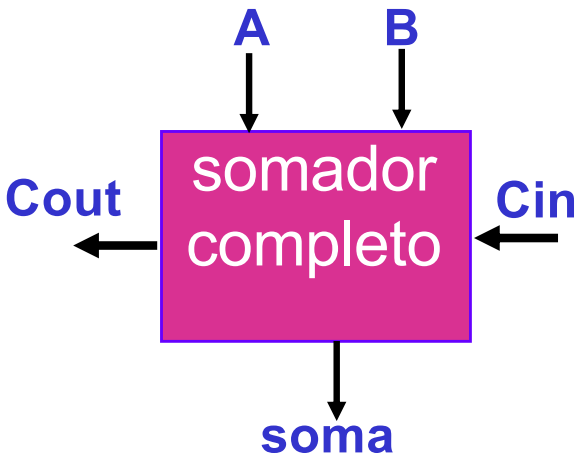
Somador / subtrator

Subtração: a – b ➔ a + $\overline{b}$ +1



Propagação e Geração

Propriedades do FA



$$p_i = a_i \oplus b_i$$

$$g_i = a_i \,.\, b_i$$

A	B	C_i	S	C_o	*Carry status*
0	0	0	0	0	delete
0	0	1	1	0	delete
0	1	0	1	0	propagate
0	1	1	0	1	propagate
1	0	0	1	0	propagate
1	0	1	0	1	propagate
1	1	0	0	1	generate
1	1	1	1	1	generate

Propagação e Geração - *Cout*

$p_i = a_i \oplus b_i$ $\quad$ $g_i = a_i \,.\, b_i$

$$C_{out} = \bar{a}.b.c + a.\bar{b}.c + a.b.\bar{c} + a.b.c$$

$$C_{out} = c.(\bar{a}.b + a.\bar{b}) + a.b$$

$$C_{out} = c.(a \oplus b) + a.b$$

$$C_{out} = c.\boldsymbol{p} + \boldsymbol{g}$$

A	B	C_i	S	C_o	*Carry status*
0	0	0	0	0	delete
0	0	1	1	0	delete
0	1	0	1	0	propagate
0	1	1	0	1	propagate
1	0	0	1	0	propagate
1	0	1	0	1	propagate
1	1	0	0	1	generate
1	1	1	1	1	generate

Ou seja: o carry é gerado quando temos geração (g=1) ou propagação e carry de entrada igual a ‘1’

Propagação e Geração

$$p_i = a_i \oplus b_i$$

$$g_i = a_i \,.\, b_i$$

Soma: $s_i = a_i \oplus b_i \oplus C_{in}$

$$\boldsymbol{s_i = p_i \oplus C_{in}}$$

Cout: $\boldsymbol{C_{out} = c.p + g}$

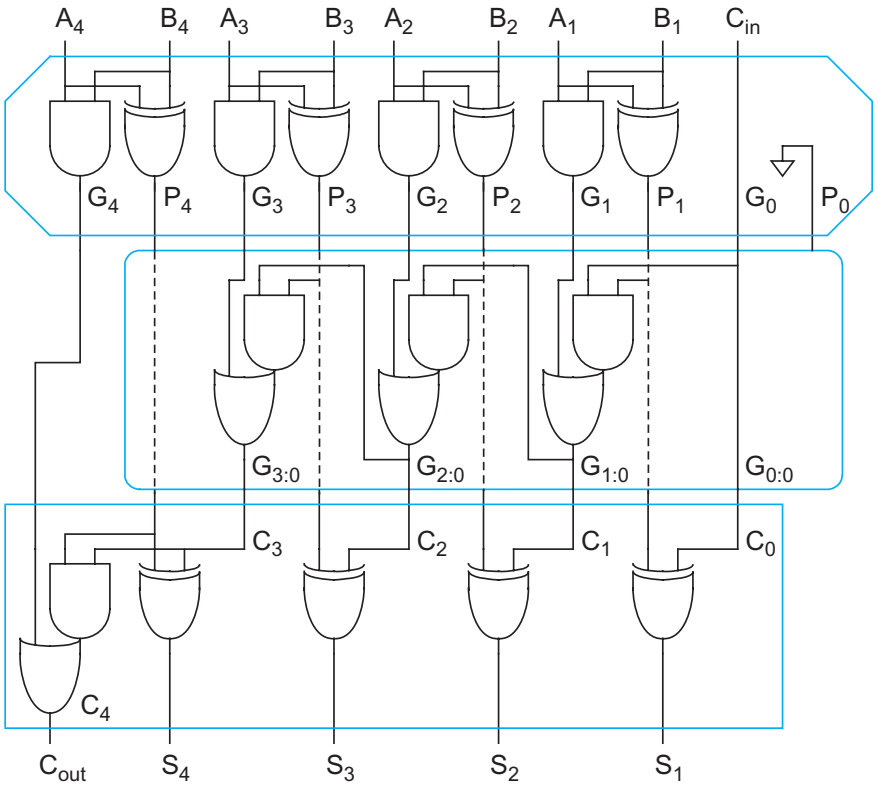


FIGURE 11.14 4-bit carry-ripple adder using PG logic

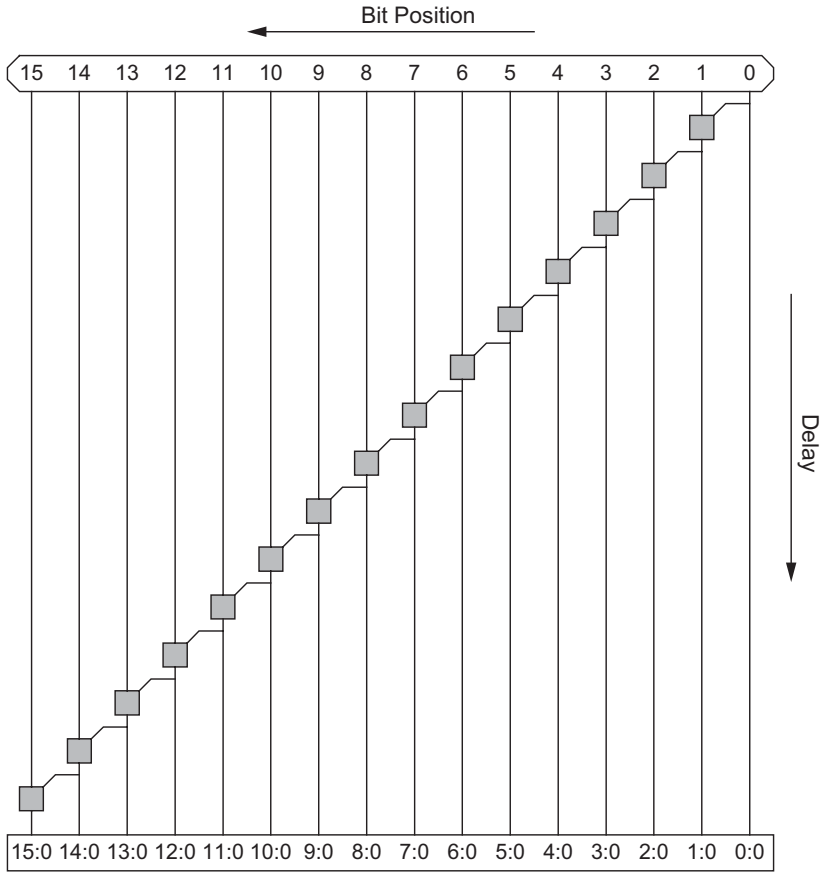
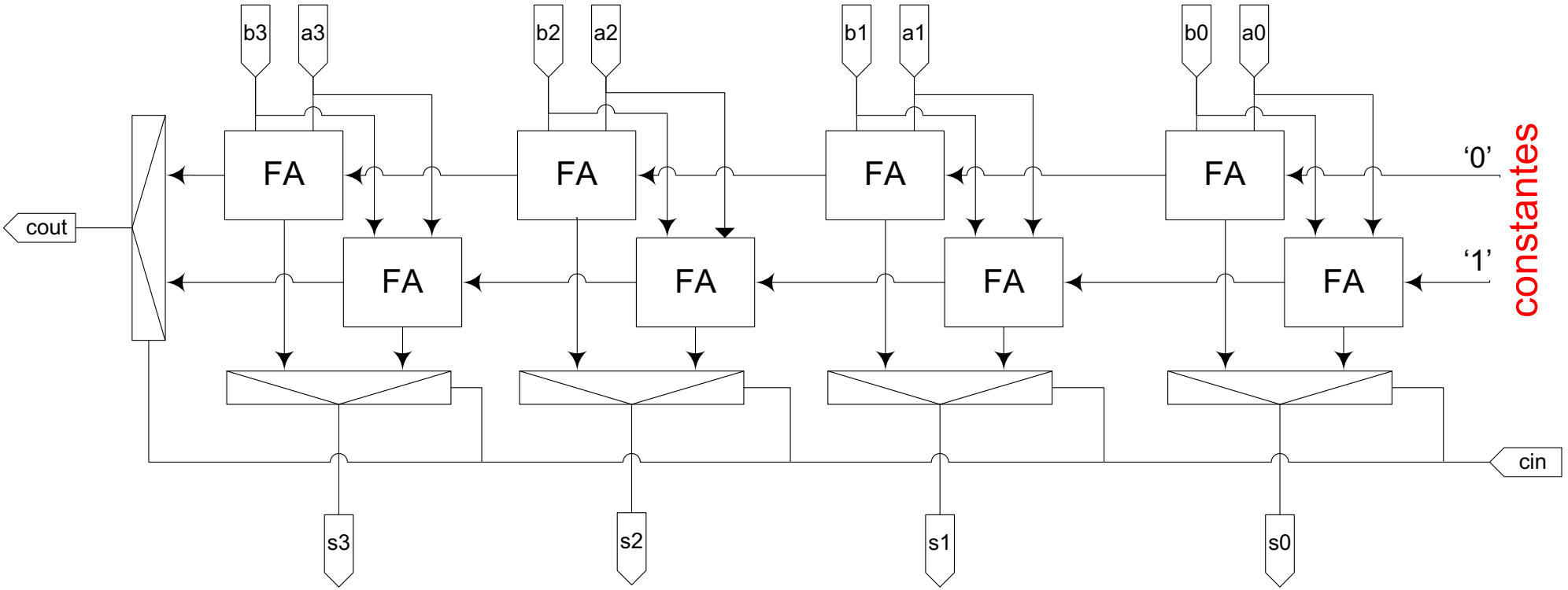


FIGURE 11.15 Carry-ripple adder group PG network

(2) Carry select adder – 1 estágio com 4 bits

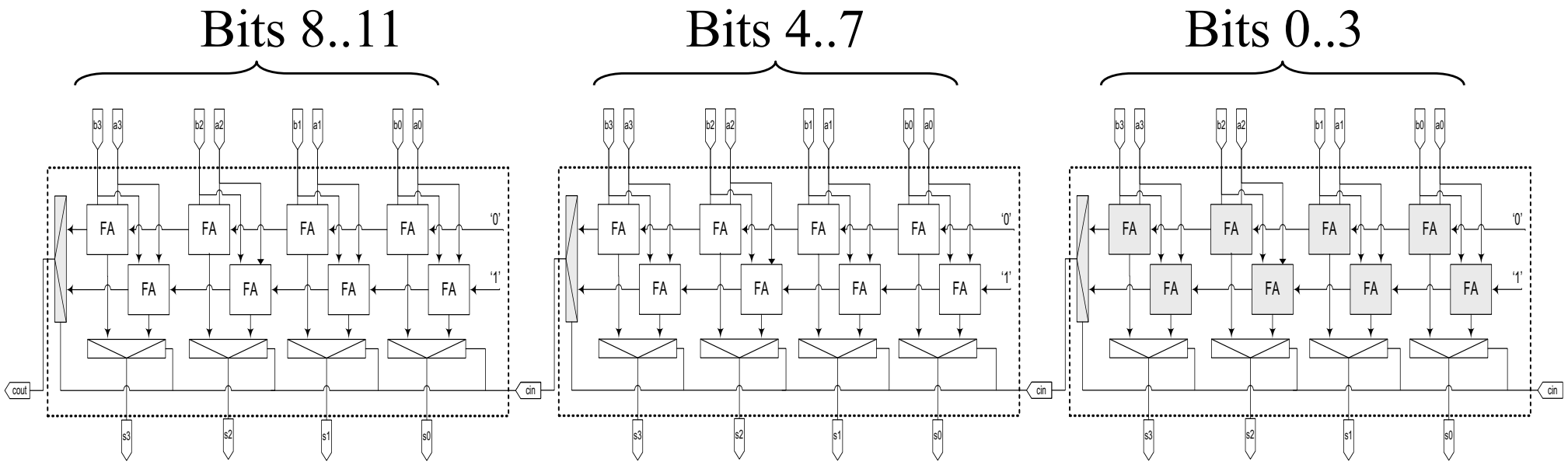
Duplica a geração da soma e do carry para cada bit



O C_{out} do estágio é escolhido entre os c_{outs} calculados no estágio anterior

Carry Select Adder

Carry select adder – 3 estágios de 4 bits

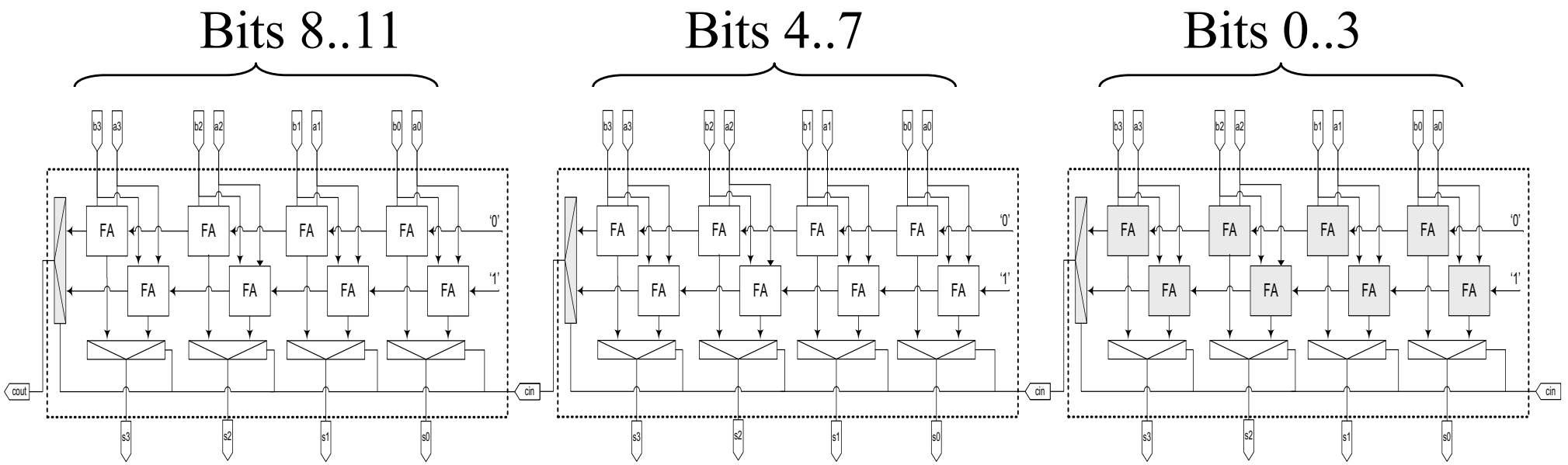


- **Vantagem:** todos os *n* grupos calculam em **paralelo** os valores da soma para os dois possíveis valores de carry de entrada

- A escolha de qual soma é a correta é feita pela propagação do carry através dos multiplexadores (marcados na cor cinza)

Carry Select Adder

Carry select adder – 3 estágios de 4 bits



$$T = t_{setup} + M.t_{carry} + \left(\frac{N}{M} - 1\right).t_{mux} + t_{soma}$$

Simplificando:

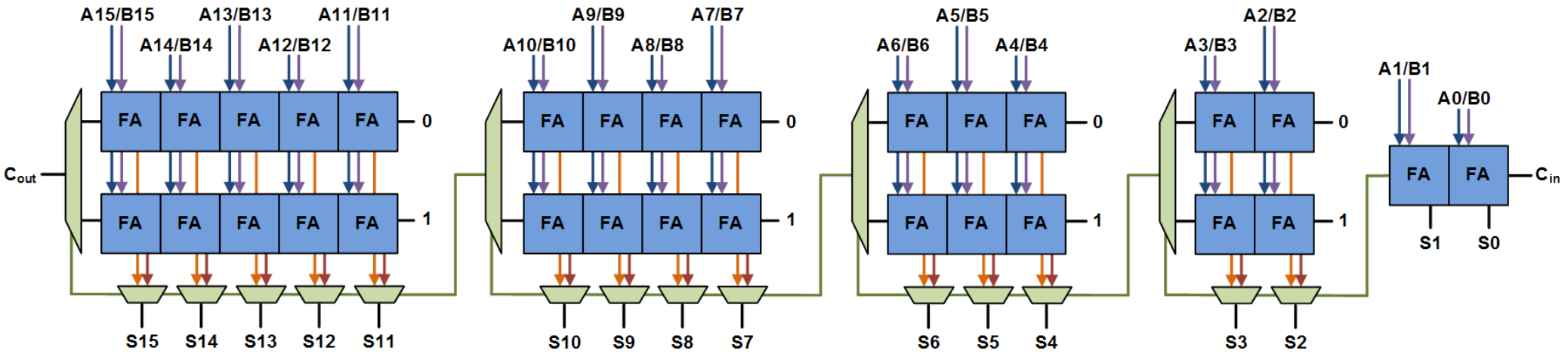
$$T = M.t_{FA} + \frac{N}{M}.t_{mux}$$

Exemplo:

bits	tFA	tmux
64	50ps	10ps

	T	freq (MHz)
Ripple	64*50ps	312,5
Select	4*50ps+16*10ps	2.777,8

Square Root Carry Select

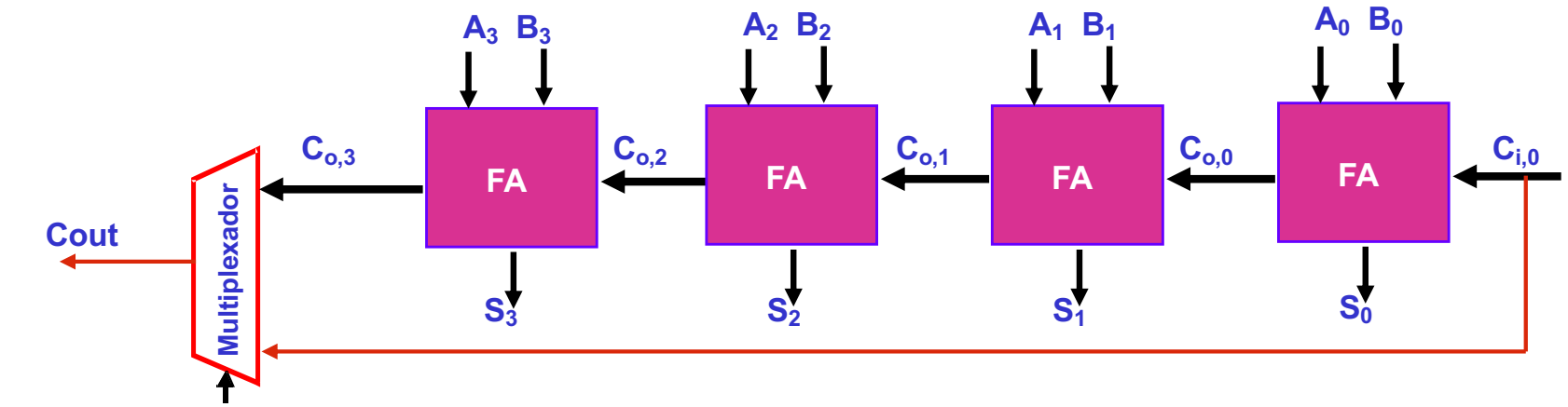


$$t_{add} = t_{setup} + P \cdot t_{carry} + (\sqrt{2N})t_{mux} + t_{sum}$$

https://en.wikipedia.org/wiki/Carry-select_adder

(3) Somador Carry-Bypass

Bloco básico



Idea: If (P0 and P1 and P2 and P3 = 1)
then $C_{o3} = C_0$

Se não há propagação o carry é gerado!

$p_i = a_i \oplus b_i$

$g_i = a_i \,.\, b_i$

$C_{out} = c.p + g$

C3 = G3 + P3.C2

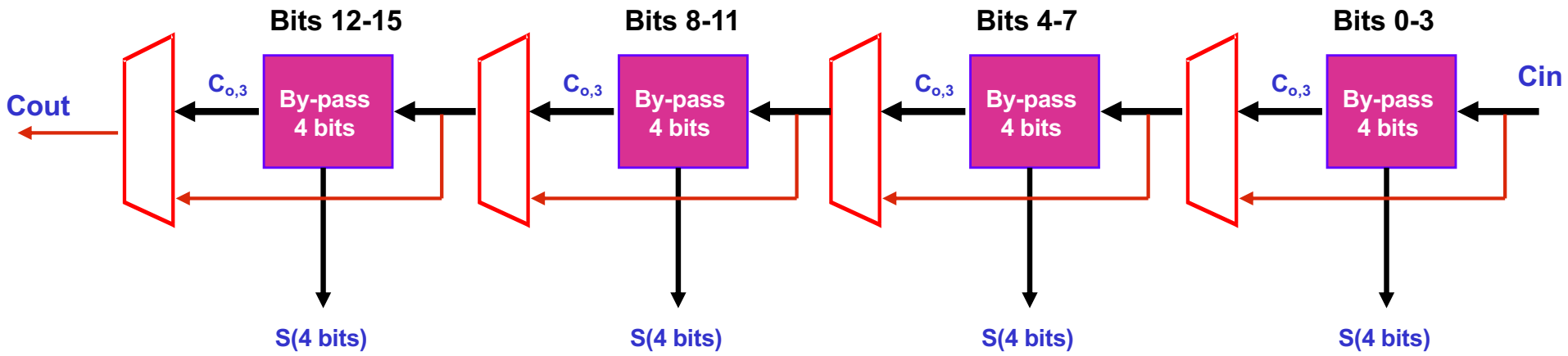
C3 = G3 + P3.(G2 + P2C1)

....

C3 = G3 + P3G2 + P3P2G1 + P3P2P1G0 + **P3P2P1P0**Cin

Somador Carry-Bypass

Construção do bypass de *n* bits

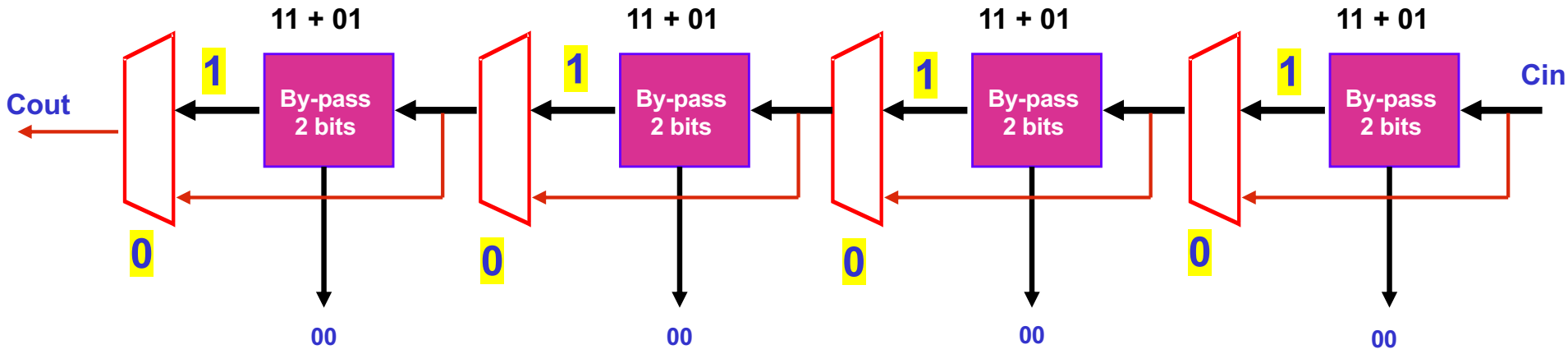


Propagação do vai-um só ocorre no primeiro e último estágio

Somador Carry-Bypass

```
   11 11  11  11
+  01 01  01  01
```

Tempo 0

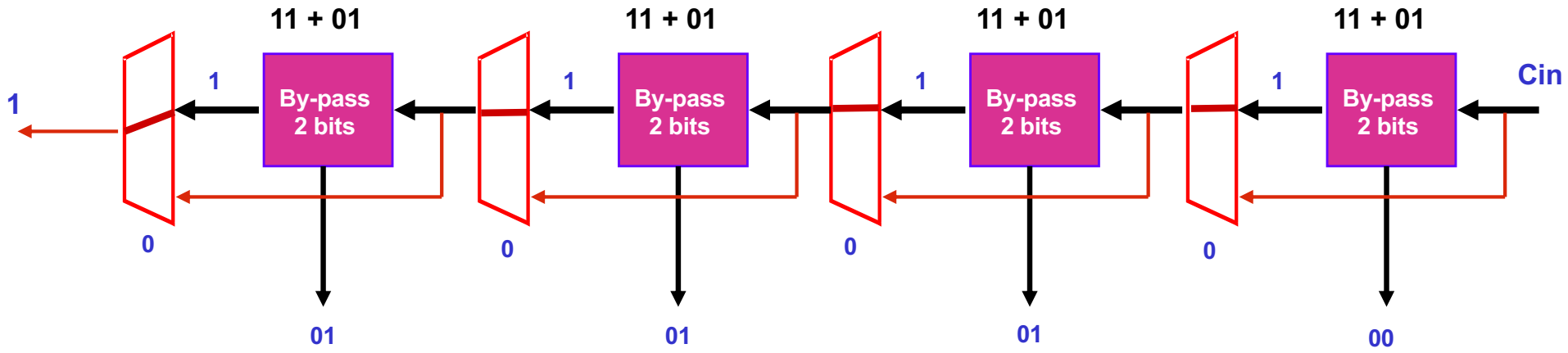


As soma ocorrem sem o carry

Somador Carry-Bypass

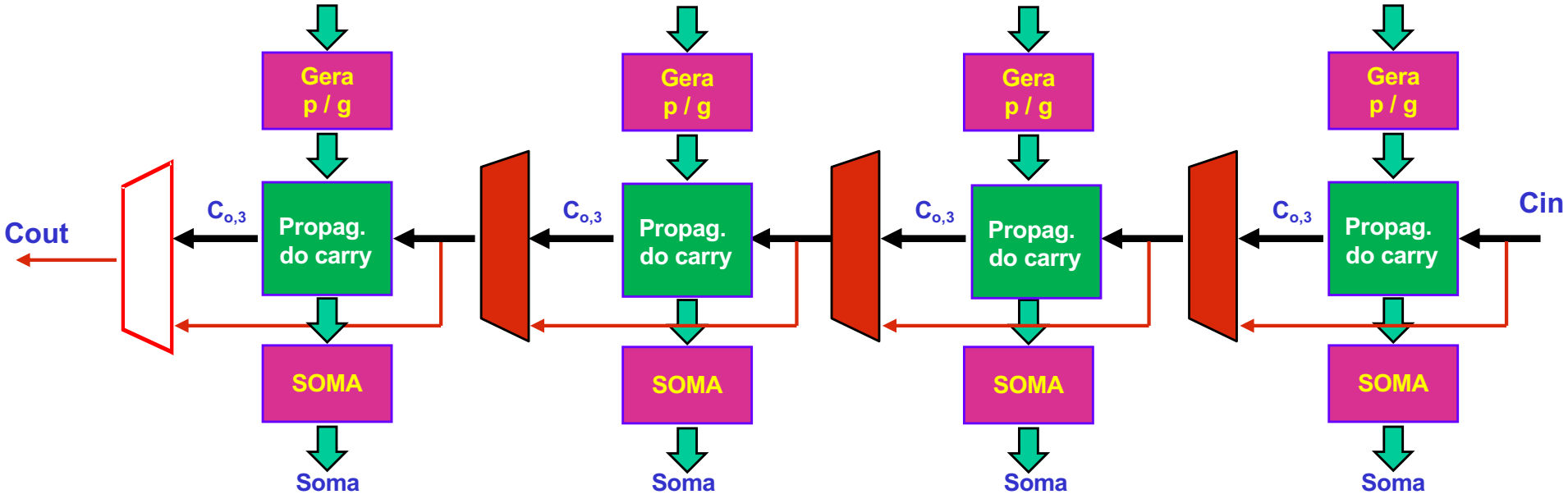
$$\begin{array}{rcccc} & 11 & 11 & 11 & 11 \\ + & 01 & 01 & 01 & 01 \end{array}$$

Tempo 1



O carry é considerado

Somador Carry-Bypass (página 572 livro texto)



$$T = t_{setup} + M.t_{carry} + \left(\frac{N}{M} - 1\right).t_{mux} + (M-1).t_{carry} + t_{soma}$$

M → número de bits em cada estágio
N → número total de bits
T_{setup} – tempo para calcular em paralelo os p,g
T_{carry} – tempo de propagação do vai-um em M bits

Simplificando:

$$T = 2.M.t_{FA} + \frac{N}{M}.t_{mux}$$

bits	tFA	tmux
64	50ps	10ps

	T	freq (MHz)
Ripple	64*50ps	312,5
Select	4*50ps+16*10ps	2777,8
Bypass	8*50ps+16*10ps	1785,7

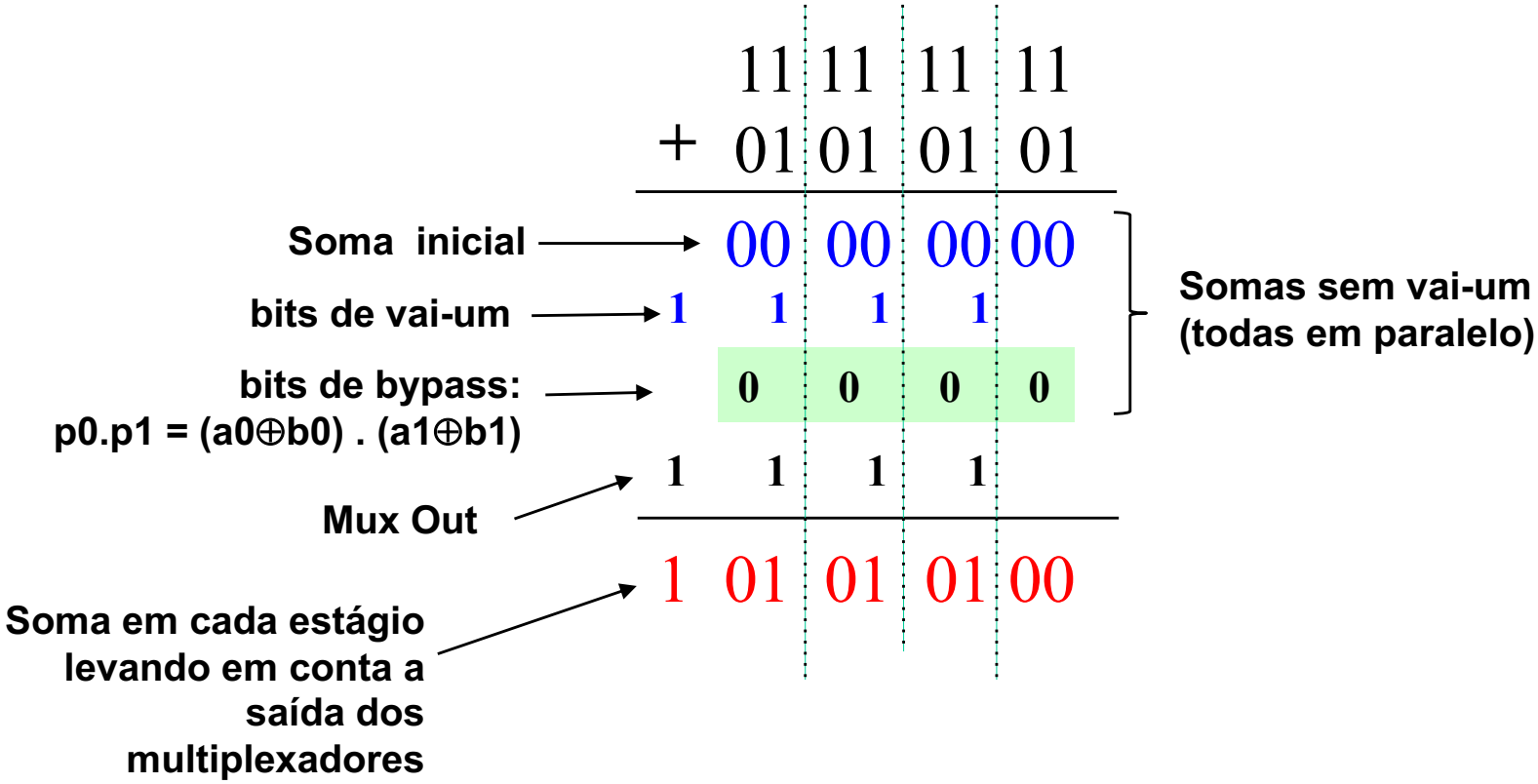
Somador Carry-Bypass

Exemplo 1 (supondo estágio de dois bits):

Etapa 1: calcula os carrys e os bits de by-pass, sem propagação de vai-um

Etapa 2: realiza a soma em função dos bits de carry e bypass

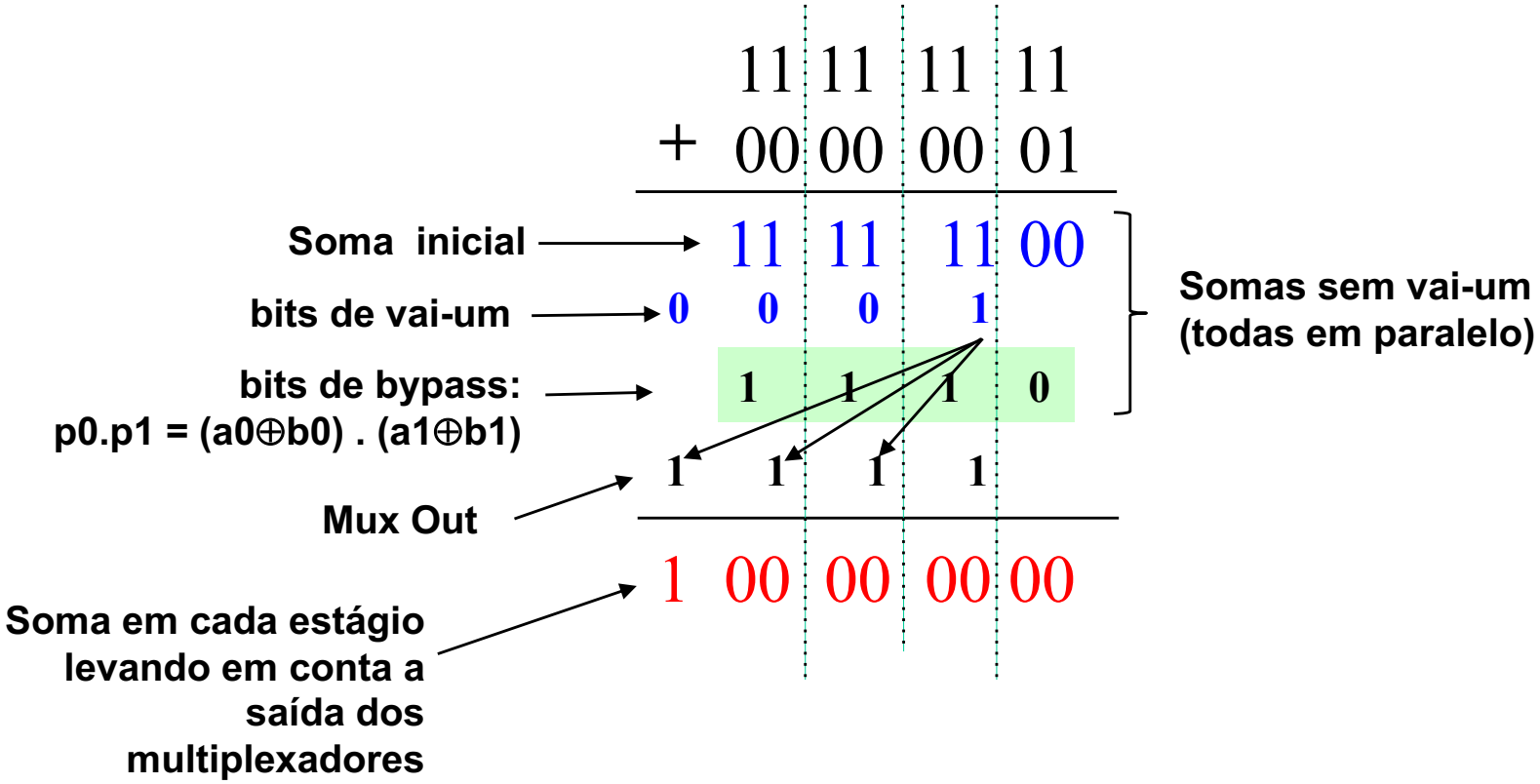
- Não há nenhum bit de bypass
- O ganho está no fato que o *vai-um* de **cada estágio é calculado em paralelo**



Somador Carry-Bypass

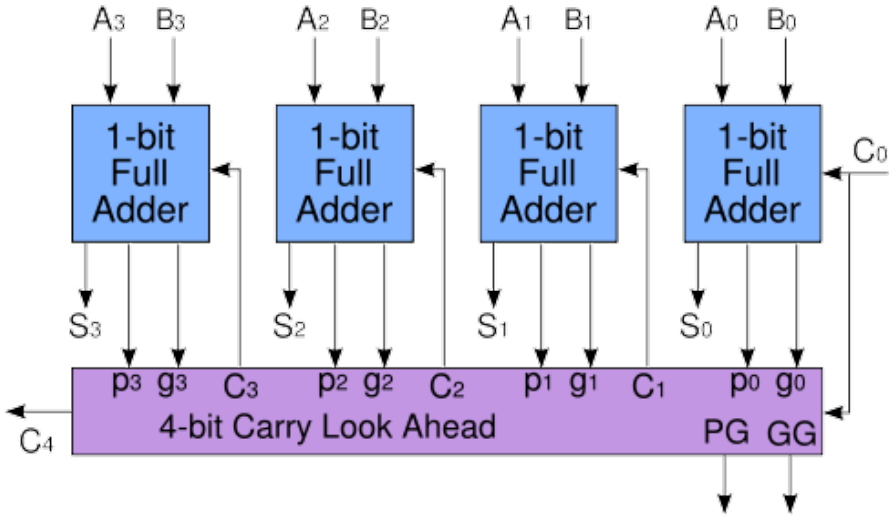
Exemplo 2 (supondo estágio de dois bits):

- **Dado que todos os bits de bypass são calculados em paralelos, todas as parcelas irão receber o *vai-um* do primeiro estágio**



(4) Carry Look-Ahead

- **Mesmo princípio: um bloco básico seguido de *n* instâncias do bloco básico**
- **PRINCIPAL DIFERENÇA: o cálculo do vai-um é feito por uma porta lógica**
- **LIMITAÇÃO: atraso da porta lógica limita o uso do bloco básico do CLA em 4 bits**



C0 = cin

C1 = c0p0 + g0

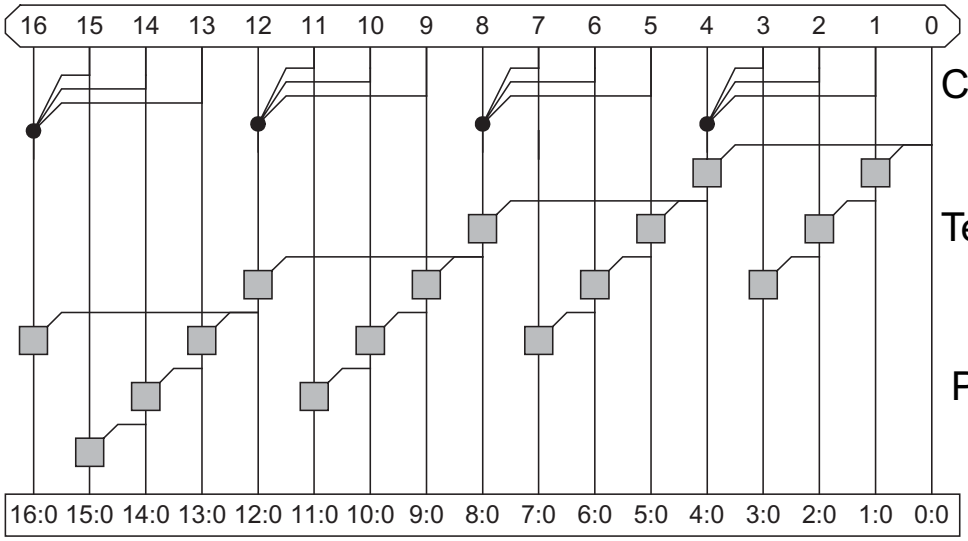
C2 = c0p0p1 + g0p1+ g1

C3 = c2p2 + g2
= c0p0p1p2+ g0p1p2+g1p2 +g2

C4 = c3p3 + g3
= c0p0p1p2p3 + g0p1p2p3 + g1p2p3 + g2p3 + g3

- **Na prática: ripple cary de 4 bits e cálculo de C4**

Carry Look-Ahead



C4 = c0p0p1p2p3 + g0p1p2p3 + g1p2p3 + g2p3 + g3

Tempo para o estágio

Propagação entre os estágios

FIGURE 11.22 Carry-lookahead adder group PG network

Equações – livro texto

Ripple carry: $T = (N-1).t_{carry} + t_{soma}$

Carry select: $T = t_{setup} + M.t_{carry} + \left(\frac{N}{M} - 1\right).t_{mux} + t_{soma}$

Carry Bypass: $T = t_{setup} + M.t_{carry} + \left(\frac{N}{M} - 1\right).t_{mux} + (M-1).t_{carry} + t_{soma}$

CLA: $T = \left(\frac{N}{M} - 1\right).t_{CLA} + (M-1).t_{carry} + t_{soma}$

LOG (Sklansky): $T = t_{setup} + M.t_{carry} + \log_2 N.t_{BK} + t_{soma}$

Equações – simplificando

Ripple carry: $T = N \,.\, t_{FA}$

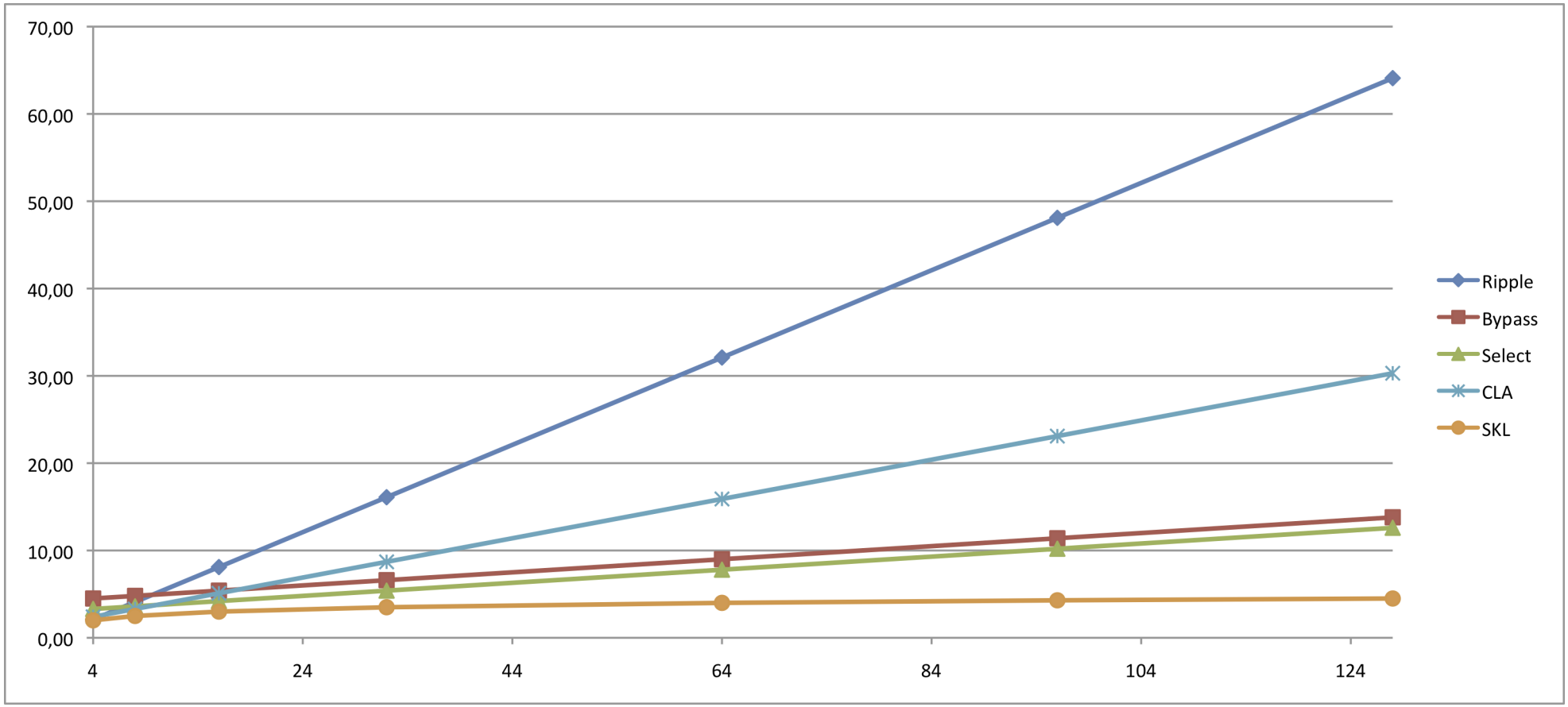
Carry select: $T = M.t_{FA} + \left(\frac{N}{M}\right).t_{mux}$

Carry Bypass: $T = 2\,.\,M.t_{FA} + \left(\frac{N}{M}\right).\,t_{mux}$

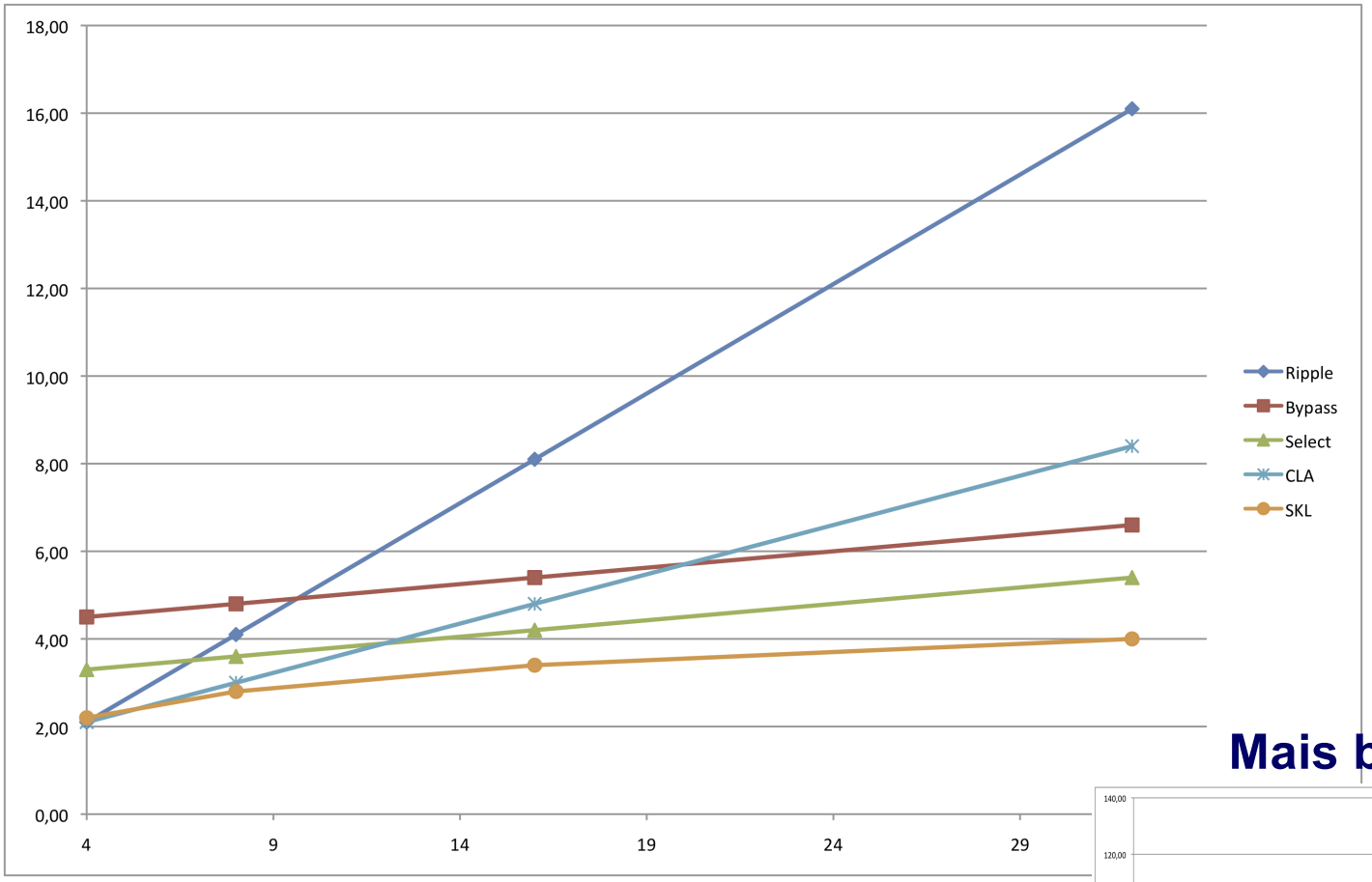
CLA: $T = t_{setup} + M.t_{FA} + \left(\frac{N}{M} - 1\right).t_{CLA}$

LOG (Sklansky): $T = t_{setup} + M.t_{carry} + \log_2 N.t_{BK} + t_{soma}$

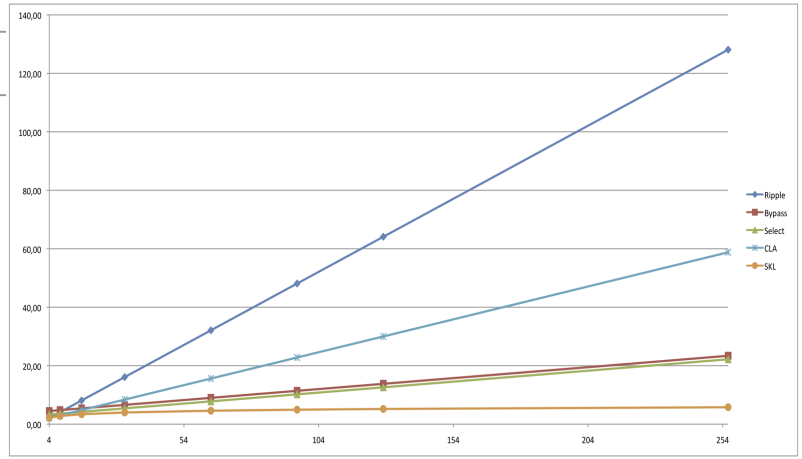
Comparação do tempo de propagação do vai-um dos diversos somadores (equações do livro texto)



Fazendo um zoom



Mais bits (até 256)

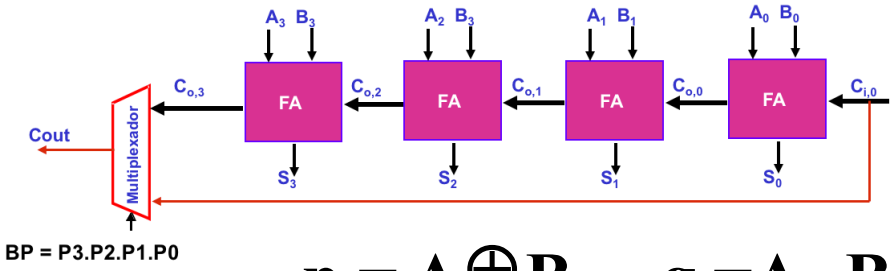


Estimativa do número de transistores

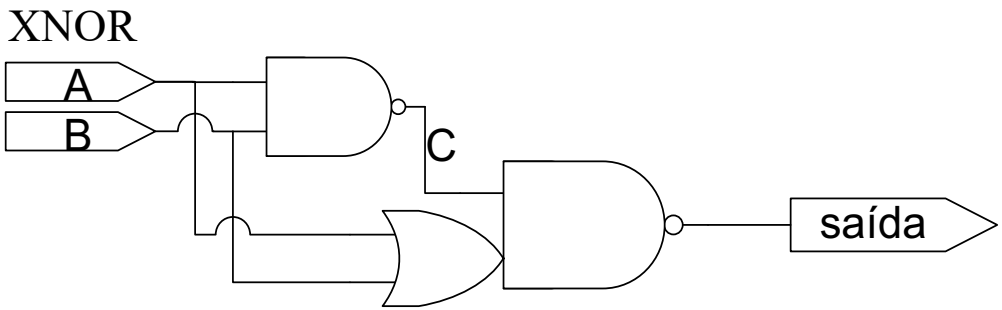
bits	RIPPLE	Bypass	Select	CLA	SKL
4	112	142	254	132	142
8	224	284	508	264	350
16	448	568	1016	528	822
32	896	1136	2032	1056	1878
64	1792	2272	4064	2112	4214
128	3584	4544	8128	4224	9334
256	7168	9088	16256	8448	20470

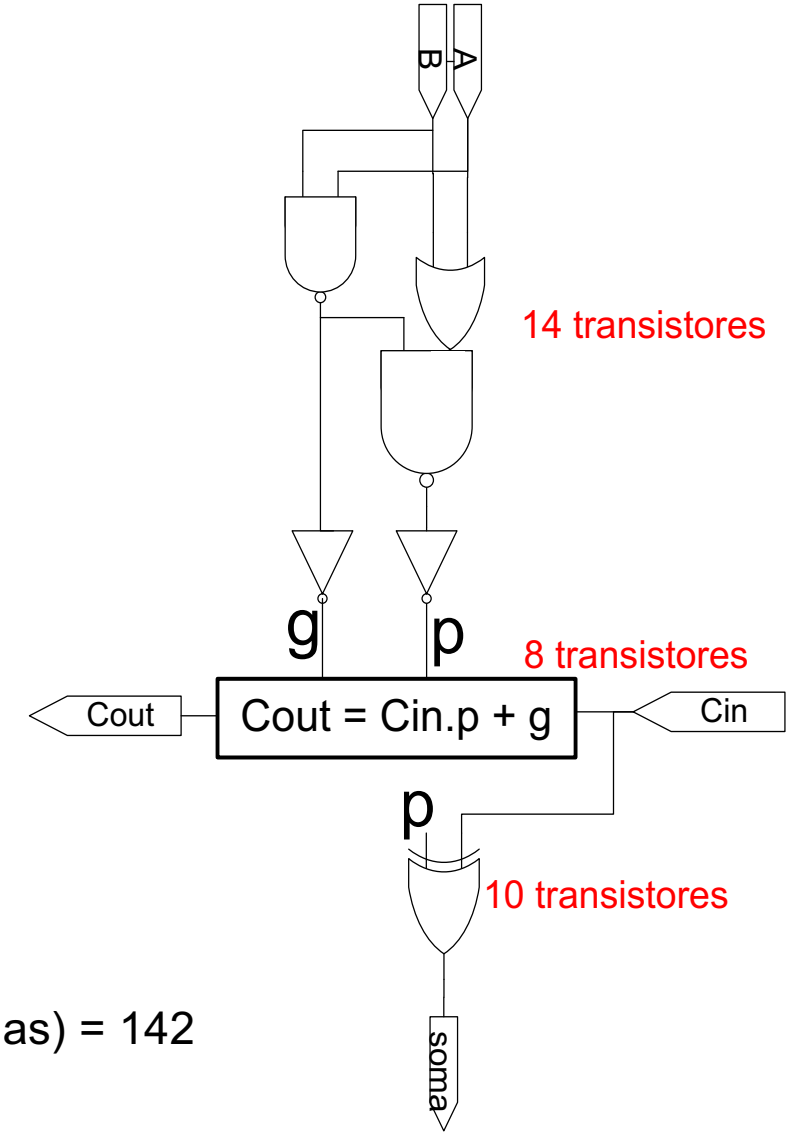
Ripple: 28 transistores * número de bits do somador
Bypass: bloco básico de 4 bits: 32*4 + 6(mux) + 8(nand) = 142
Select: [(28(FA)+28(FA)+6(MUX)]*4+6(mux)= 254
CLA: bloco básico: 28*4 + 20= 132 (supondo que só há célula de vai-um para geração de vai-um na saída do estágio)

Número de transistores Carry-Bypass



$$p = A \oplus B_i \qquad g = A \cdot B$$



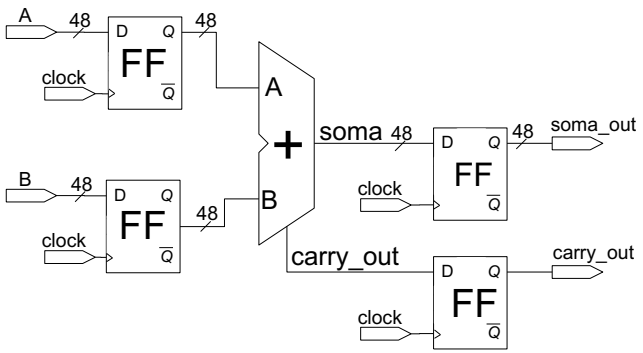


Bypass: bloco básico de 4 bits:

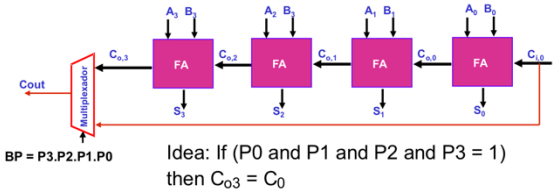
32*4 + 6(mux) + 8 (nand 4 entradas) = 142

Exercício

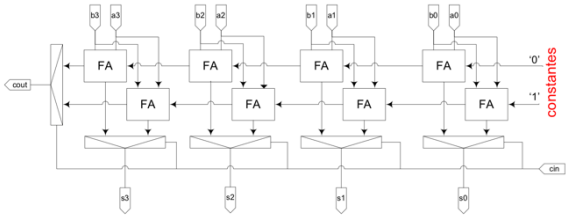
1. (3,5) Considere o circuito abaixo, composto por 2 entradas de 48 bits (A e B), e duas saídas (soma_out de 48 bits, e carry_out). O tempo de um estágio em um circuito pipeline compreende o atraso devido aos registradores e à lógica combinacional. Os operandos A e B são armazenados no FFs de entrada e somados. Os FFs de saída (segundo estágio) armazenam a soma das entradas do ciclo de clock anterior. **O projetista tem por restrição de projeto a frequência de operação deste circuito maior ou igual a 1 GHz, com o menor consumo de área de silício possível (ou seja, número de transistores). Os componentes que o projetista possui são:**
   - **Flip-flop D**, com tempo de propagação D→Q igual a **25ps** e o tempo de setup igual a **5ps**.
   - **Somador completo (FA)**, com tempo de propagação igual a **90ps**.
   - **Multiplexador**, com tempo de propagação igual a **20ps**.



Dentre as opções para soma, o projetista deve considerar: *ripple carry*, *carry-bypass* com estágio de 4 bits, e *carry-select* com estágio de 4 bits.



Estágio de 4 bits do carry-bypass



Estágio de 4 bits do carry-select

a) (3,1) Complete a tabela abaixo, considerando os **três somadores**.

	FF D→Q (ps)	FF t_{Setup} (ps)	Somador de 48 bits (ps)		Atraso do 1º estágio (ps)	Freq. de operação do circuito (GHz)
			Número de transistores	Atraso (ps)		
Circuito com somador ripple-carry	25	5	0,3	0,3	0,1	0,2
Circuito com somador carry-bypass	25	5	0,4	0,4	0,1	0,2
Circuito com somador carry-select	25	5	0,4	0,4	0,1	0,2

b) (0,4) Qual a arquitetura de somador escolhida pelo projetista? <u>Justifique</u> a resposta em função da frequência de operação e do custo em área do somador de 48 bits.

	FF D→Q (ps)	FF t_{Setup} (ps)	Somador de 48 bits (ps)		Atraso do 1º estágio (ps)	Freq. de operação do circuito (GHz)
			Número de transistores	Atraso (ps)		
Circuito com somador ripple-carry	25	5	48*28=1344 0,3	0,3	0,1 4350	0,2 0,230
Circuito com somador carry-bypass	25	5	=12*(4*32+6+8) 0,4 = 1704	0,4	0,1 990	0,2 1,010
Circuito com somador carry-select	25	5	=12*(8*28+30) 0,4 =3048	0,4	0,1 630	0,2 1,587

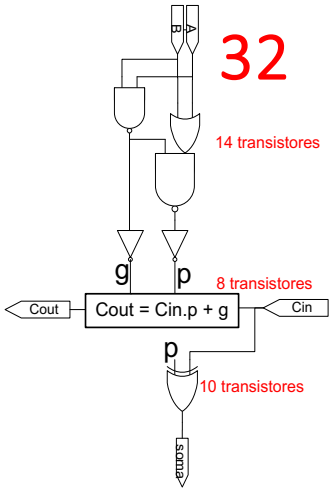
rquitetura de somador escolhida pelo projetista? **Justifique** a resposta em função da frequência de operação e do custo

Tempo dos somadores:

t_{FA} = 48 bits * 90ps = 4.320

t_{BP} = (48/4) * 20ps + 2*(4 * 90) = 960

t_{CS} = (48/4) * 20ps + 4 * 90 = 600



O projetista deve escolher o CARRY-BYPASS, pois ele tem um número de transistores inferior ao carry-select e atende à frequência especificada.

Somadores. Considere 5 somadores, com as seguintes configurações:

a. ripple carry (**RC**)
b. carry-select com estágio de 4 bits (**CS4**)
c. carry-select com estágio de 8 bits (**CS8**)
d. carry-bypass com estágio de 4 bits (**CBP4**)
e. carry-bypass com estágio de 8 bits (**CBP8**)

Estágio de 4 bits (128/4) → 32 estágios
Estágio de 8 bits (128/8) → 16 estágios

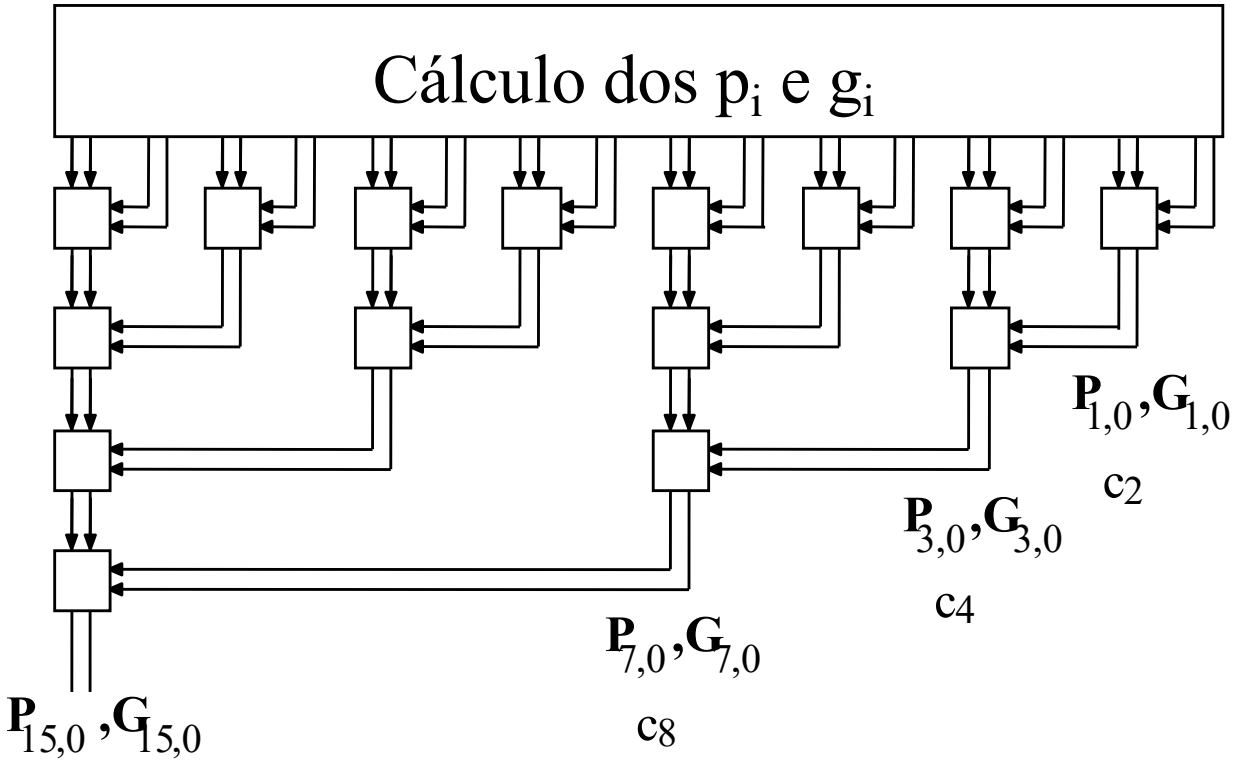
Somador	Atraso do somador (128 bits)
RC	= 128 * 120 = 15.360 ps
CS4	= (4*120) + **32***32 = 480 + 1024 = **1.504 ps**
CS8	= (8*120) + **16***32 = 960 + 512 = **1.472 ps**
CBP4	= 2*(4*120) + **32***32 = 960 + 1024 = **1.984 ps**
CBP8	= 2*(8*120) + **16***32 = 1920 + 512 = **2432 ps**

b/ 625 MHz ? **Escolhe-se o CS8**

c/ T = 100 + 20 + 1472 = **1.592 ps** → **f= 628 MHz**

Somador Logarítmico

- Todos os somadores $\log_2(n)$ iniciam por um cálculo em árvore



Célula de Brent et Kung

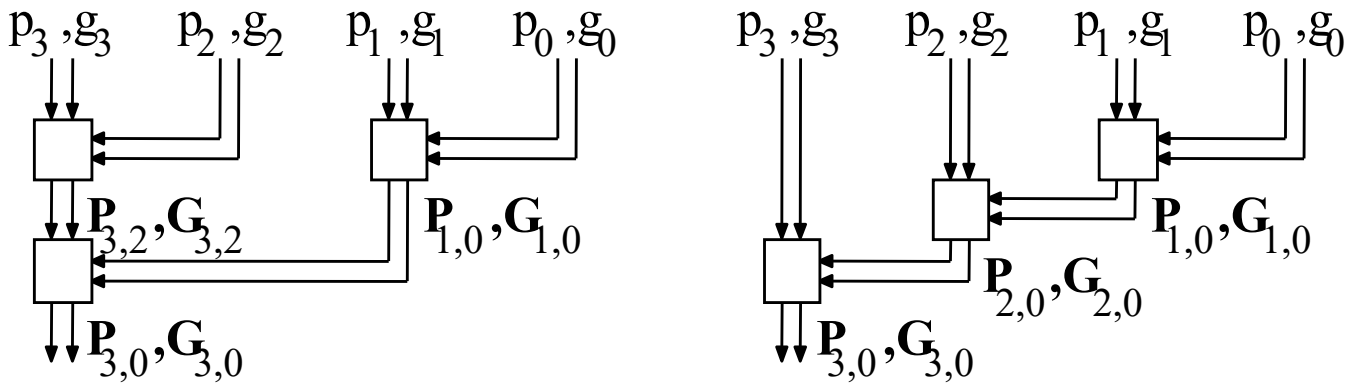
Nome dado ao conjunto de portas lógicas que calcula:

$$\mathbf{G}_{i,k} = \mathbf{G}_{i,j} + \mathbf{P}_{i,j} \cdot \mathbf{G}_{j-1,k}$$

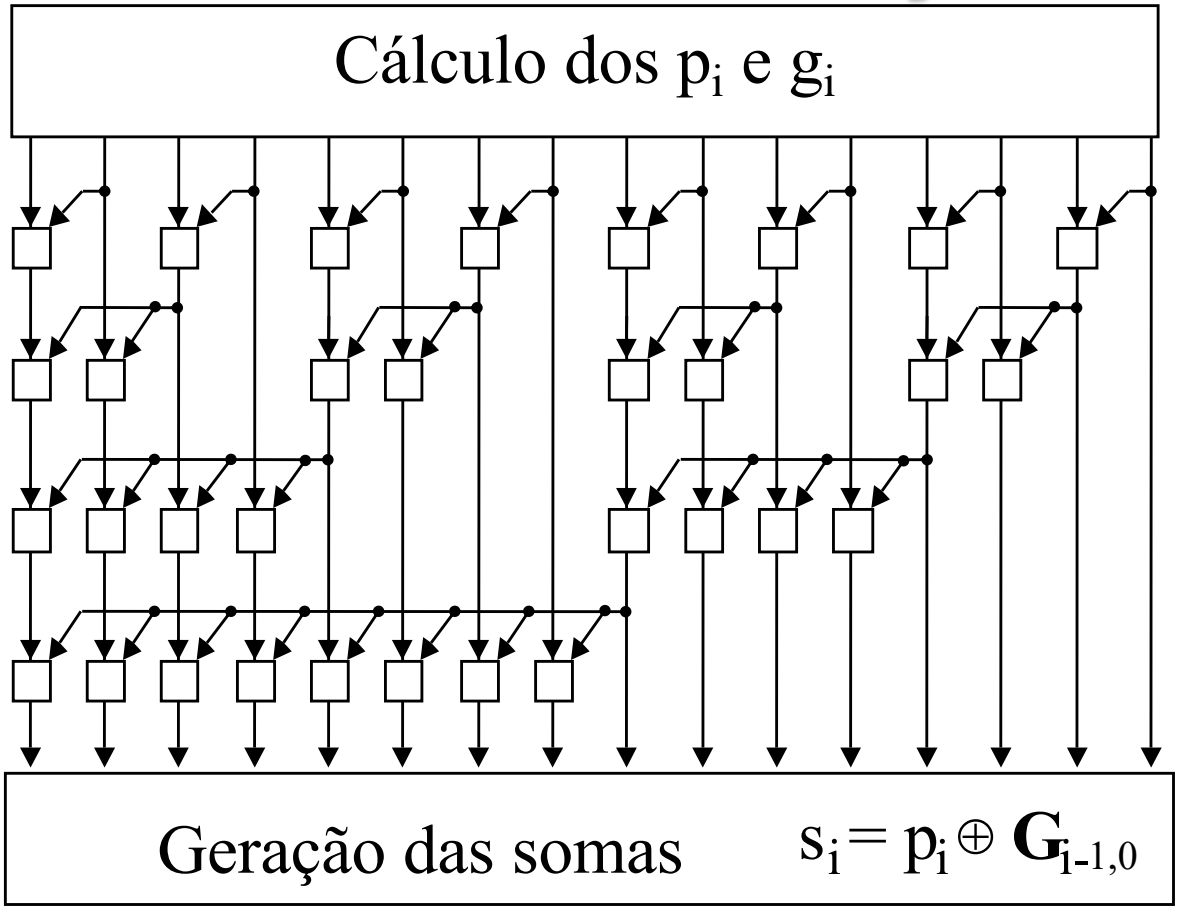
$$\mathbf{P}_{i,k} = \mathbf{P}_{i,j} \cdot \mathbf{P}_{j-1,k}$$

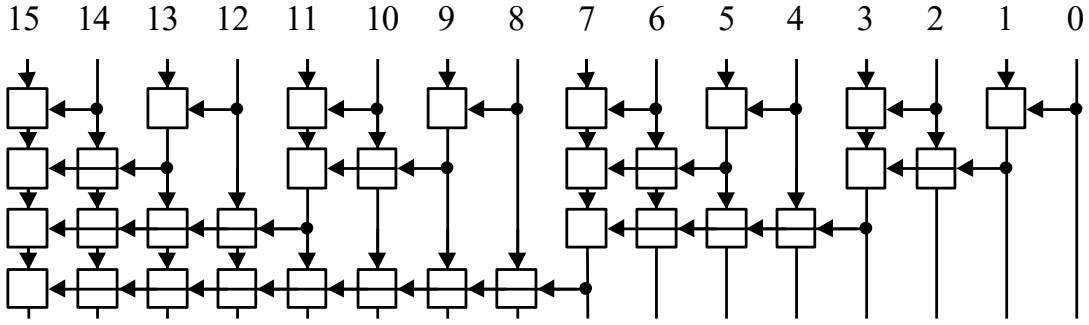
(propagação do vai-um entre bits *i* e *k*)

Exemplos de combinações:



Somador de Sklansky em tempo log2(n)

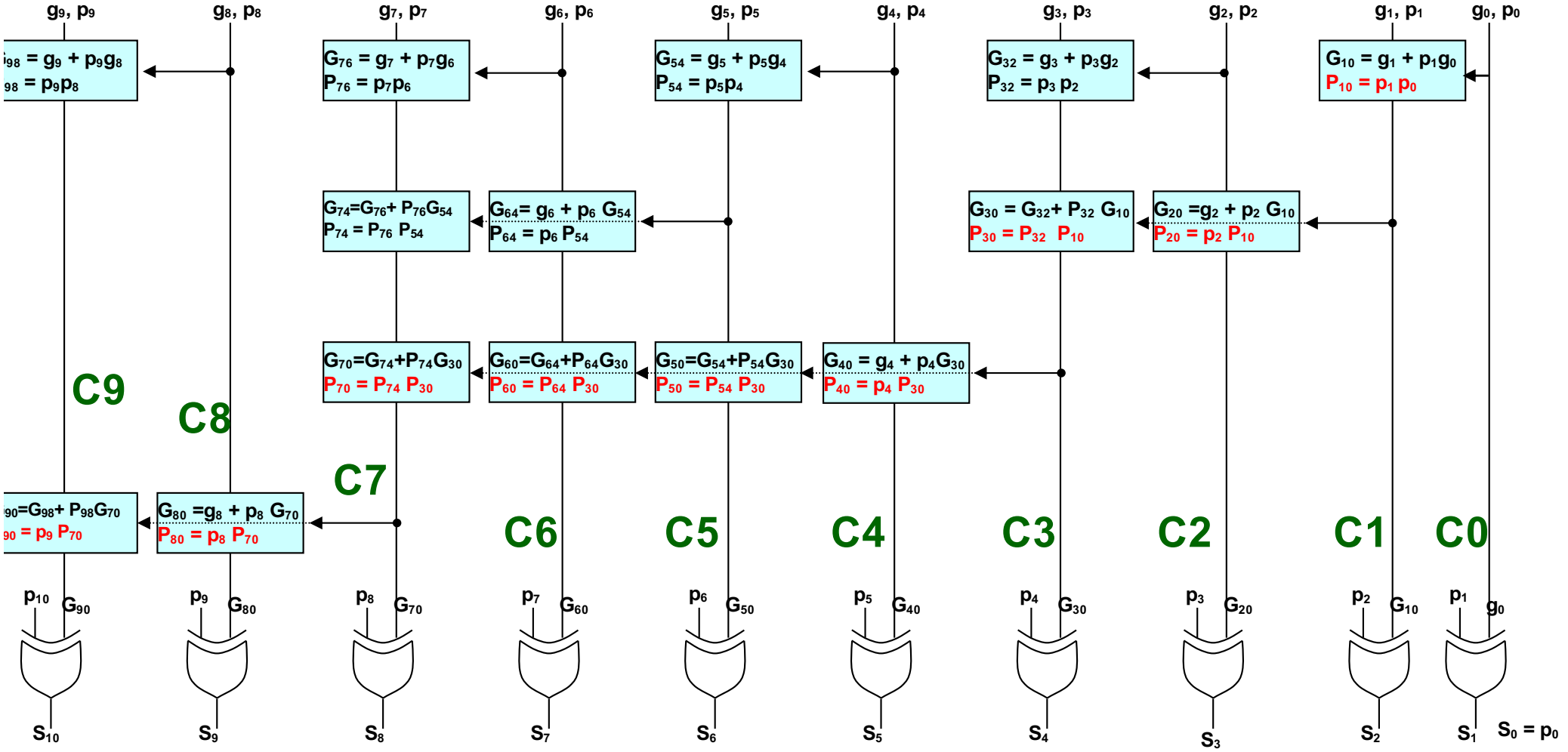




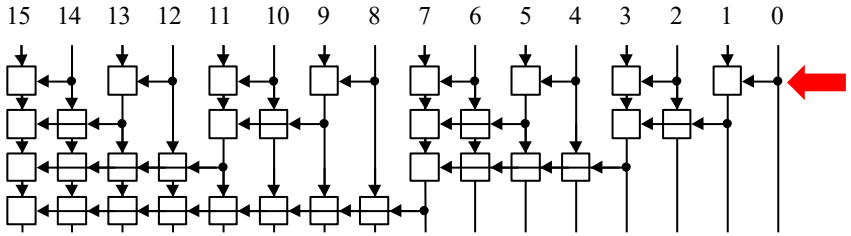
Sklansky – bits 0 a 10

Gi = ai . bi

Pi = ai ⊕bi



Detalhando o somador – carry do bit 0

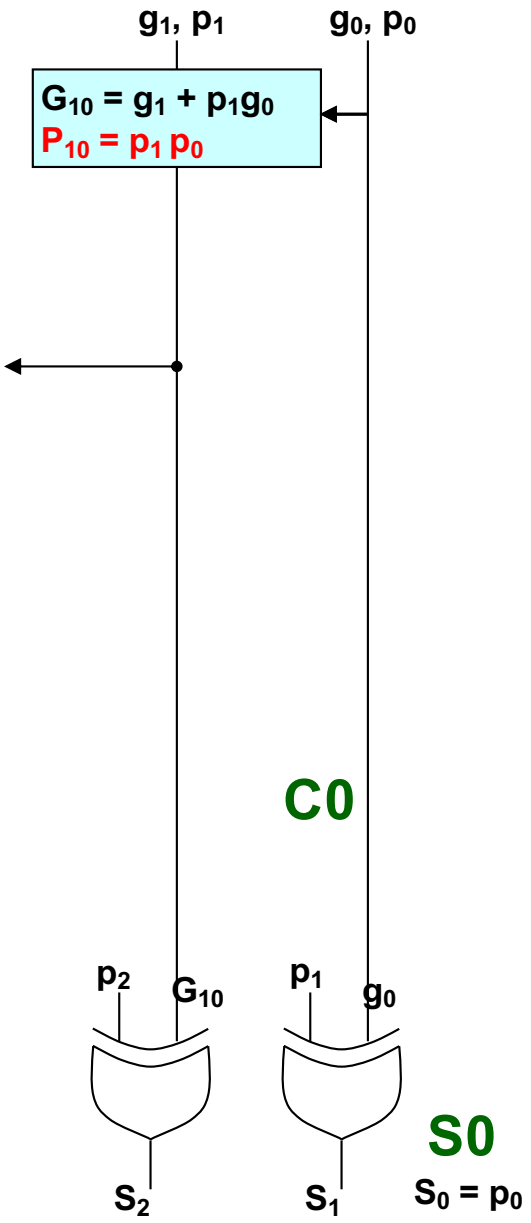


- $C_{in} = 0$

  Equação do primeiro carry:

  $C_0 = a_0b_0 + a_0C_{in} + b_0C_{in}$

  $C_0 = a_0b_0 = \mathbf{g_0}$



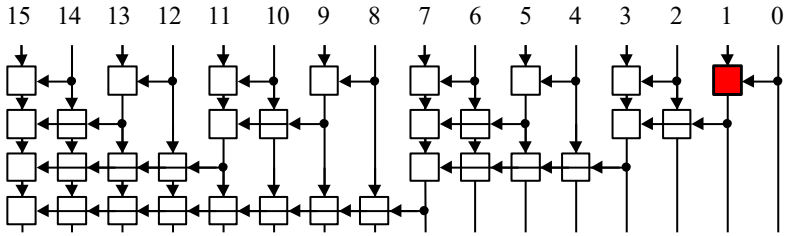
Soma do bit 0:

$$S_0 = a_0 \oplus b_0 \oplus C_{in}$$

$$\text{Considerando } C_{in} = 0$$

$$S_0 = a_0 \oplus b_0 = p_0$$

Detalhando o somador – carry do bit 1



$$C_1 = g_1 + p_1 . c_0$$

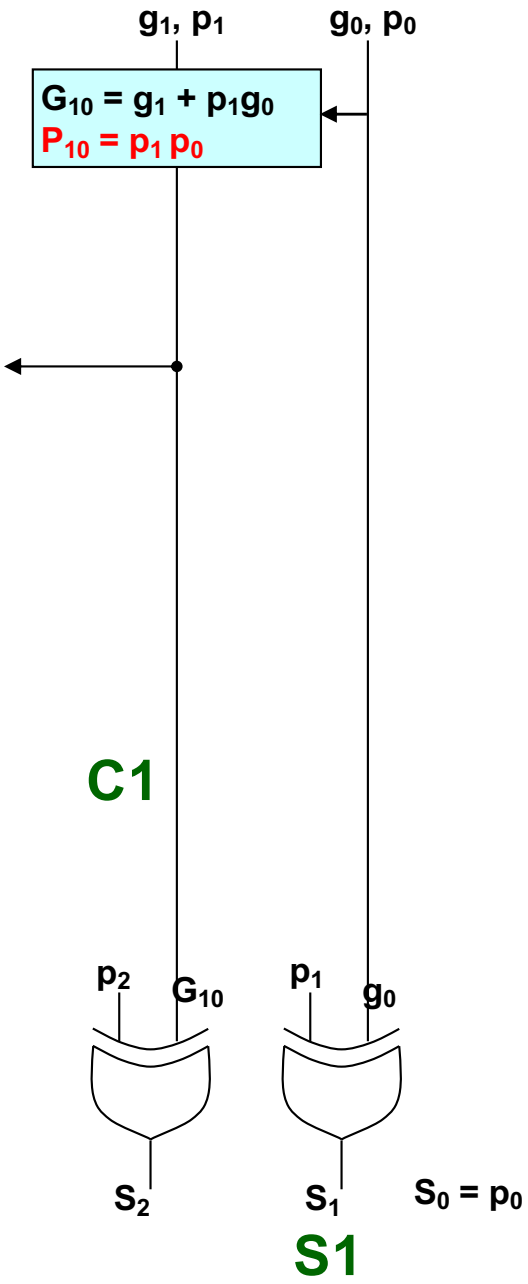
$$C_1 = g_1 + p_1 . g_0$$

$$C_1 = G10$$ ⟸ Carry do bit 1 ao 0

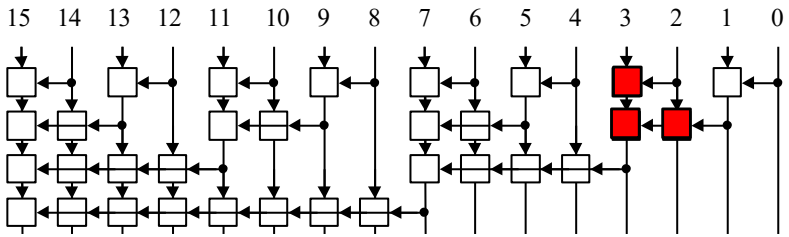
$$S_1 = a_1 \oplus b_1 \oplus c_0$$

$$S_1 = p_1 \oplus g_0$$

Todas as demais somas são calculadas desta forma

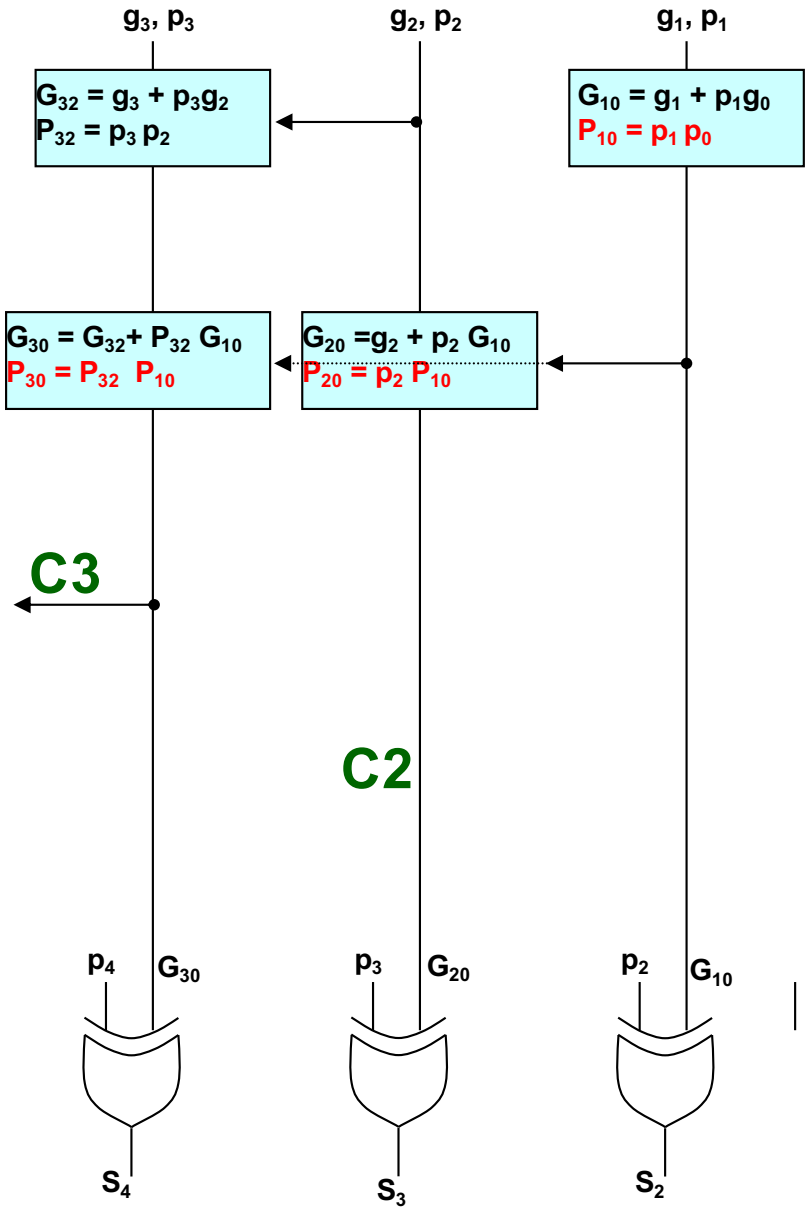


Detalhando o somador – carry dos bits 3/2



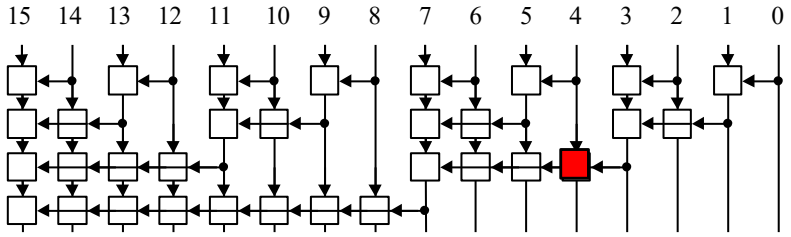
$$C_2 = g_2 + p_2 . c_1 = g_2 + p_2 . G10 = G20$$

$$C_3 = g_3 + p_3 . c_2 = g_3 + p_3 . (g_2 + p_2 . c_1)$$
$$= g_3 + p_3 . g_2 + p_3 . p_2 . G10$$
$$= G32 + P32 . G10 = G30$$



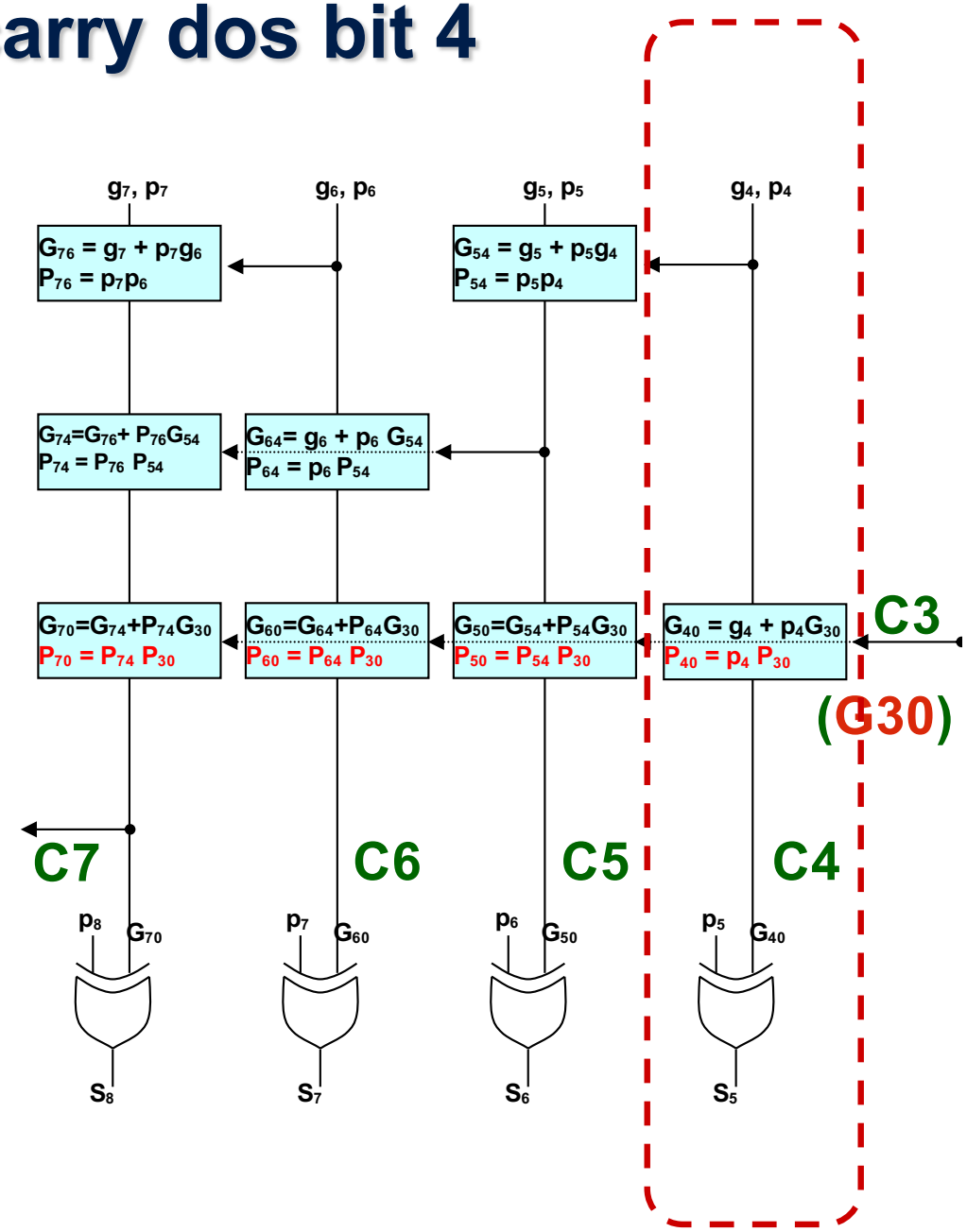
- C2 é propagado em um nível e depende do C1
- C3 é propagado em dois níveis e depende da propagação 3-2 e 1-0

Detalhando o somador – carry dos bit 4



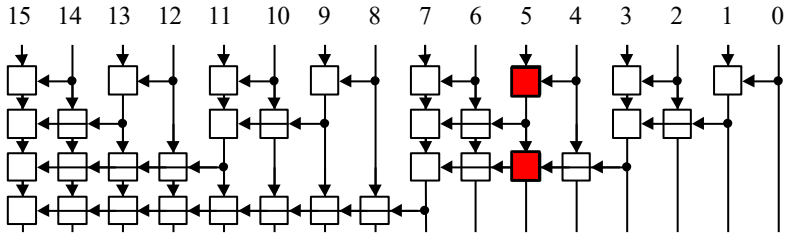
- Aumento dos níveis de propagação em até 3 camadas

$$C_4 = g_4 + p_4 . c_3 = g_4 + p_4 . G30 = G40$$

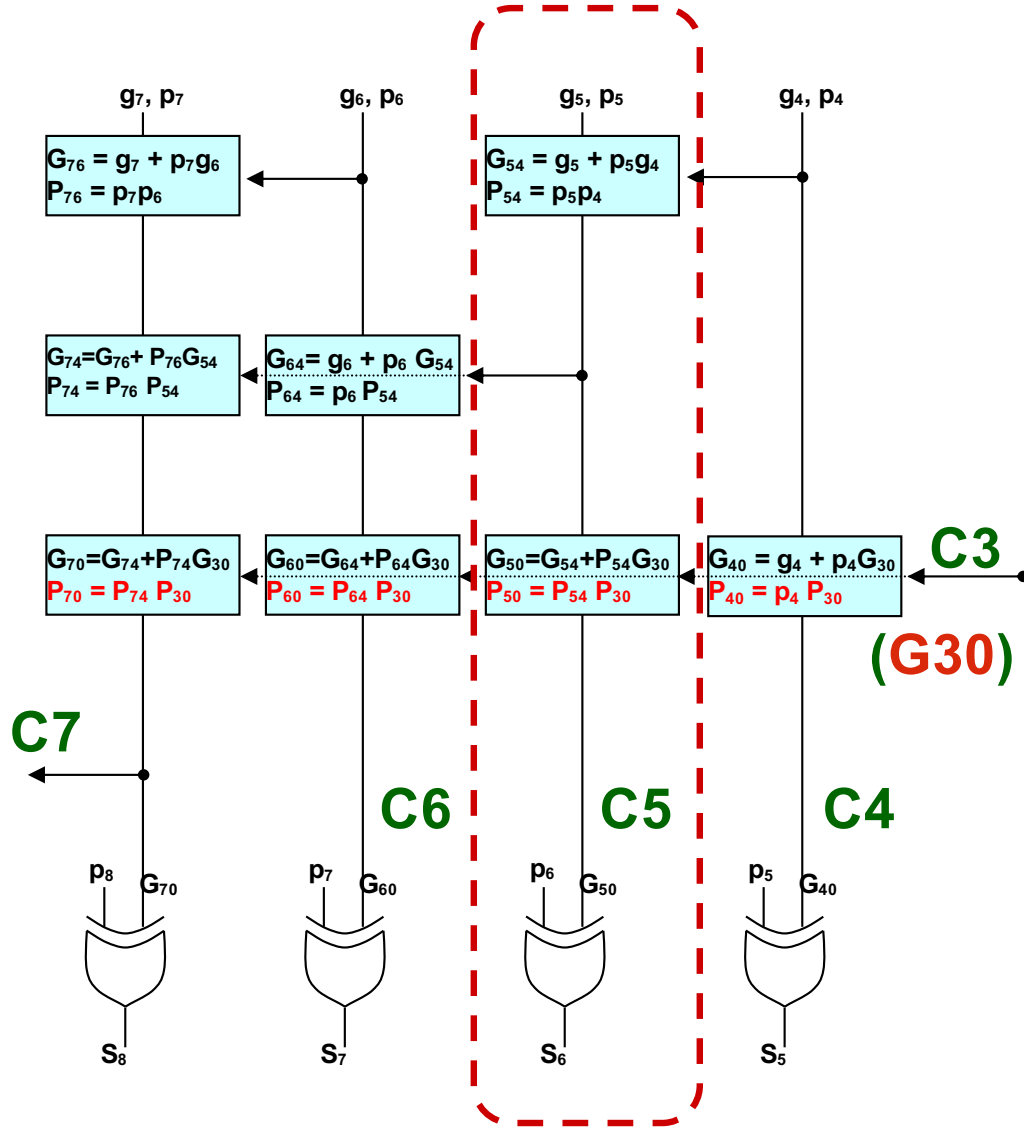


Expande a equação do carry até o carry ”de entrada” do nível

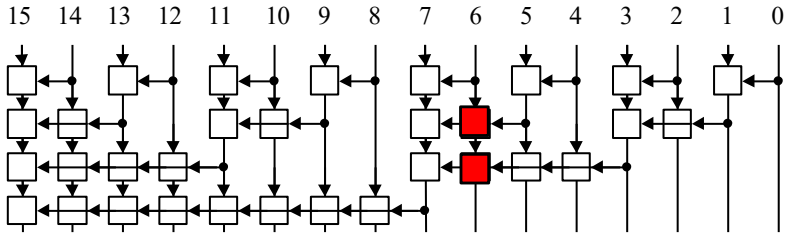
Detalhando o somador – carry dos bit 5



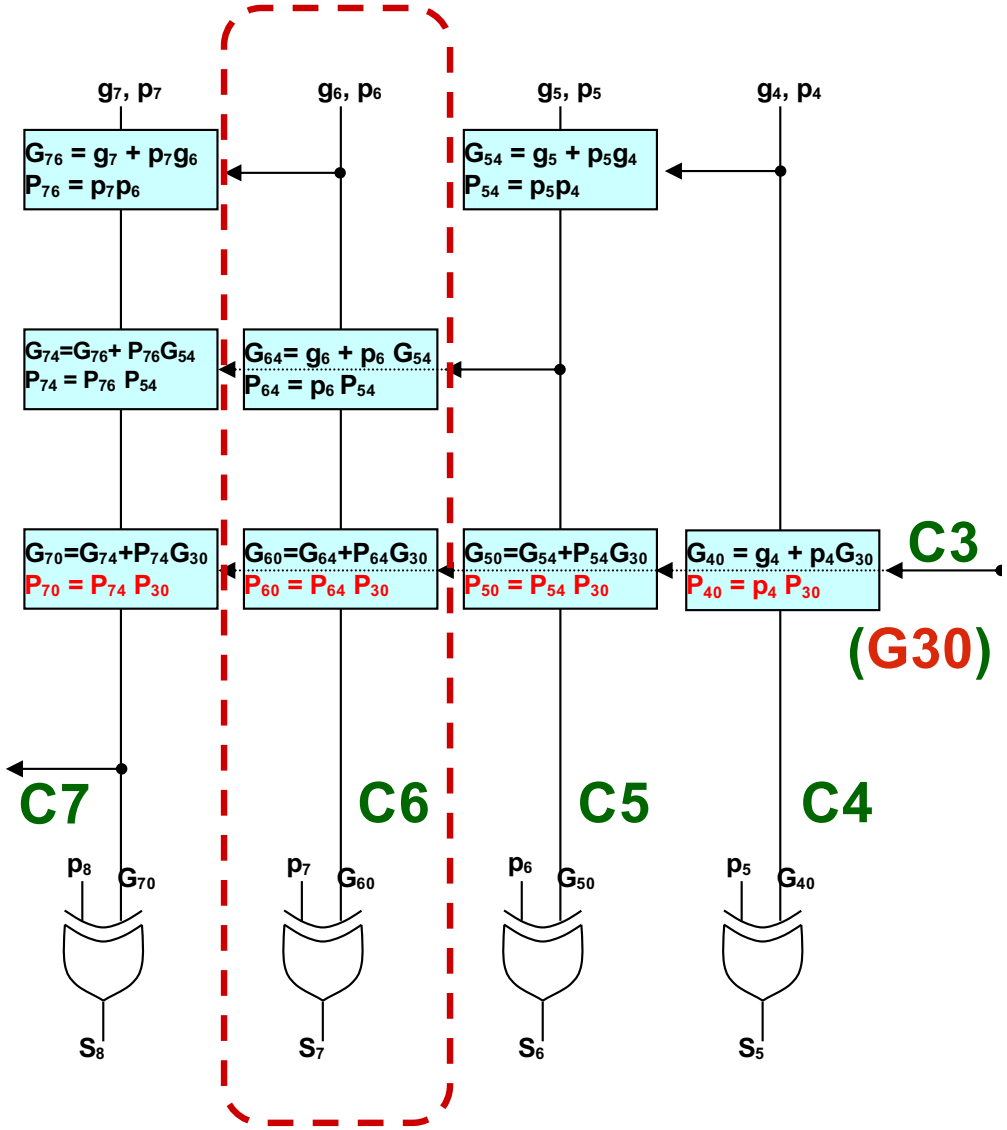
$$C_5 = g_5 + p_5 . c_4 = \quad g_5 + p_5 . g_4 + p_5 . p_4 . G30$$
$$= G54 + P54 . G30 = G50$$



Detalhando o somador – carry dos bit 6

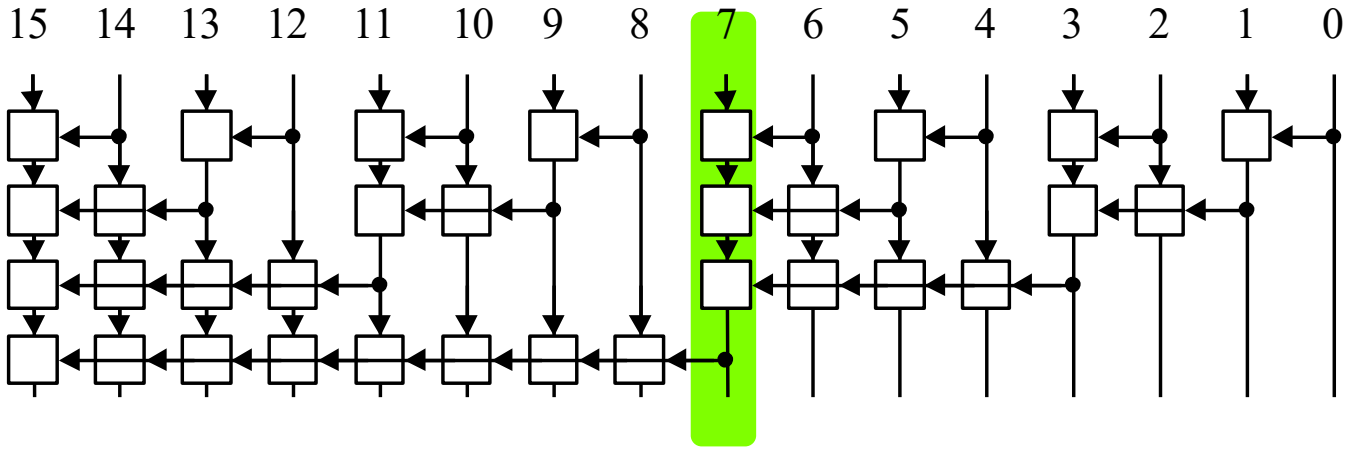


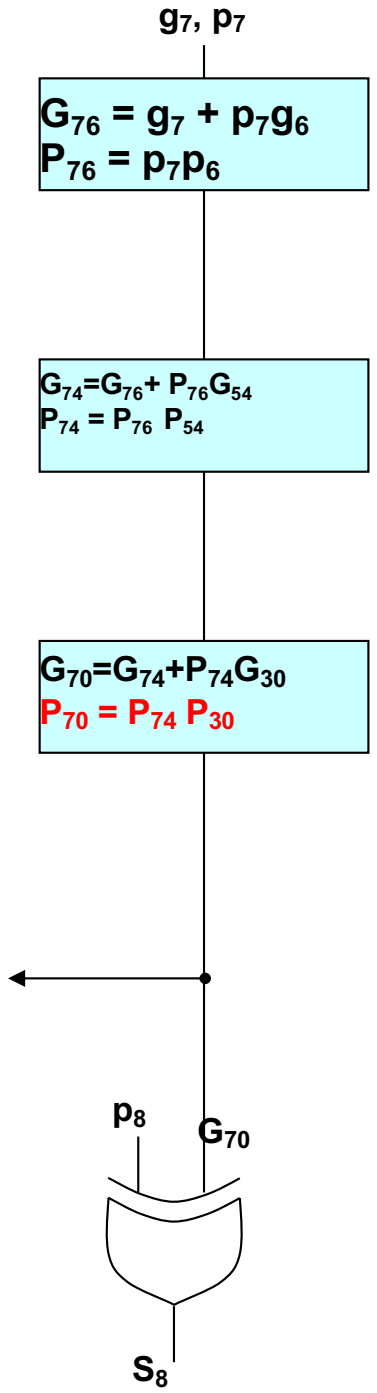
$$C_6 = g_6 + p_6 . c_5$$
$$= g_6 + p_6 . g_5 + p_6 . p_5 . g_4 + p_6 . p_5 . p_4 . G30$$
$$= g_6 + p_6 . (g_5 + p_5 . g_4) + p_6 . p_5 . p_4 . G30$$
$$= g_6 + p_6 . (G54) + p_6 . P54 . G30$$
$$= G64 + P64 . G30 = G60$$



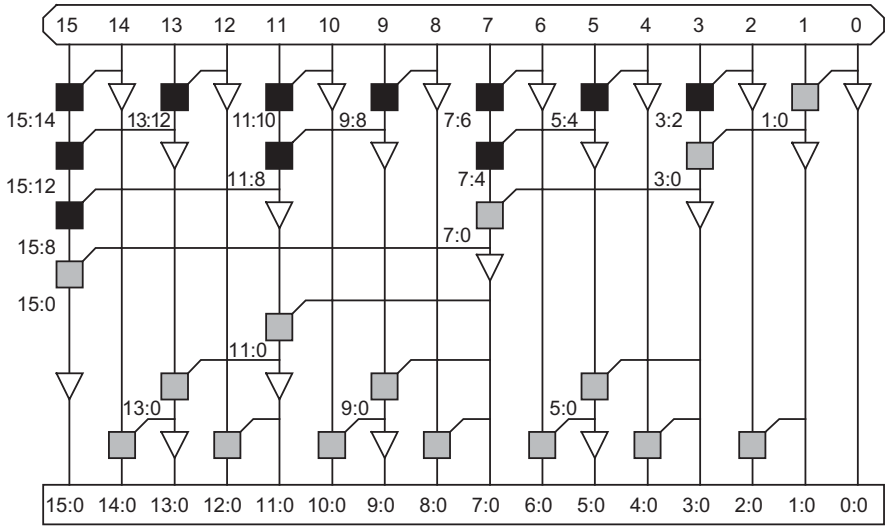
Detalhando o somador – carry dos bit 7

$$C_7 = g_7 + p_7 . c_6$$
$$= \underline{g_7 + p_7 . g_6} + \underline{p_7 . p_6 . g_5 + p_7 . p_6 . p_5 . g_4} + \underline{p_7 . p_6 . p_5 . p_4} . G30$$
$$= \underline{G76 + P76 . (g_5 + p_5 . g_4)} + \underline{P76 . P54} . G30$$
$$= G74 + P74 . G30 = G70$$

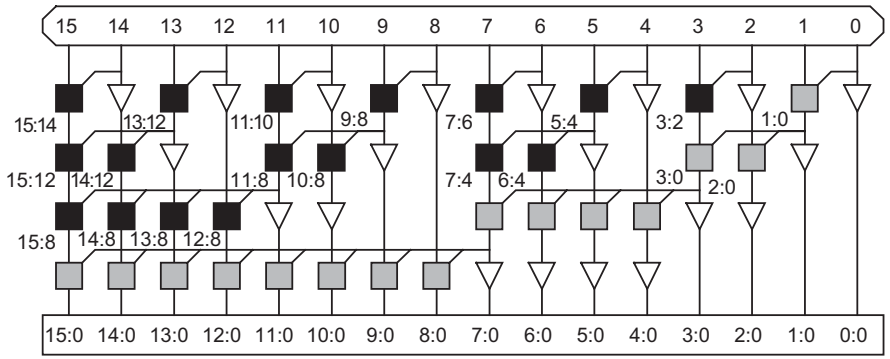




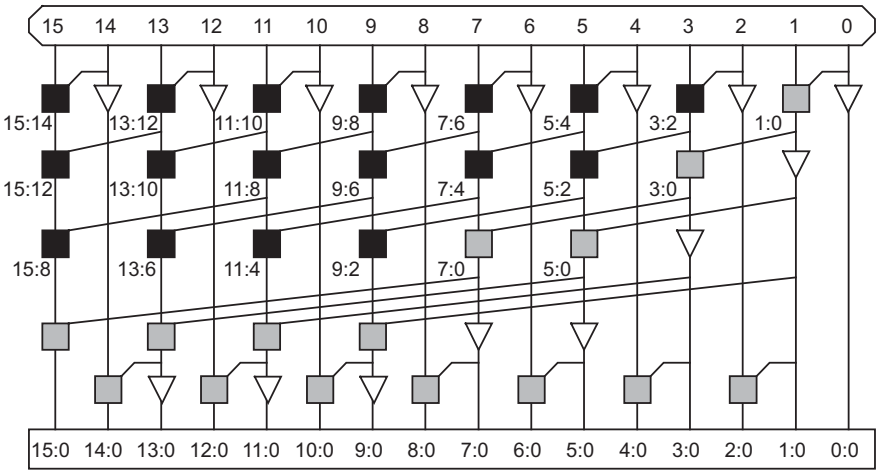
Exemplos de somadores log



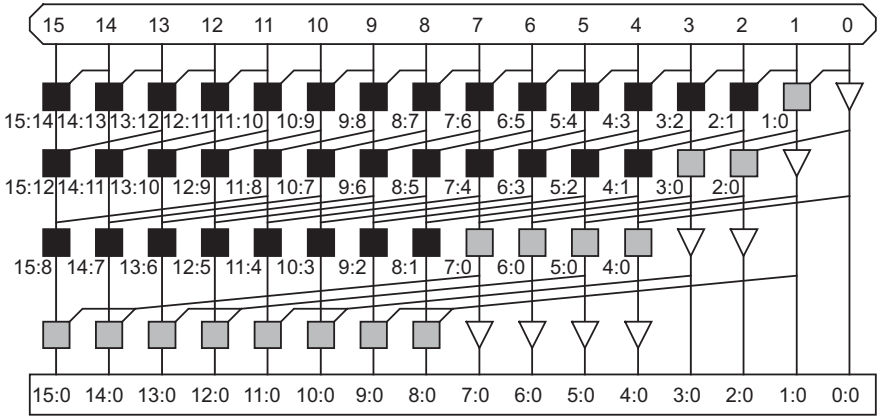
(a) Brent-Kung



(b) Sklansky

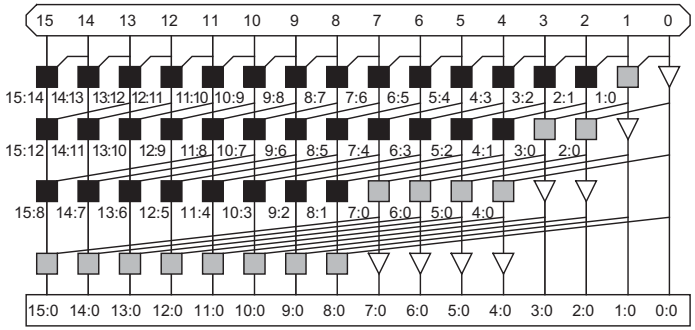


(d) Han-Carlson

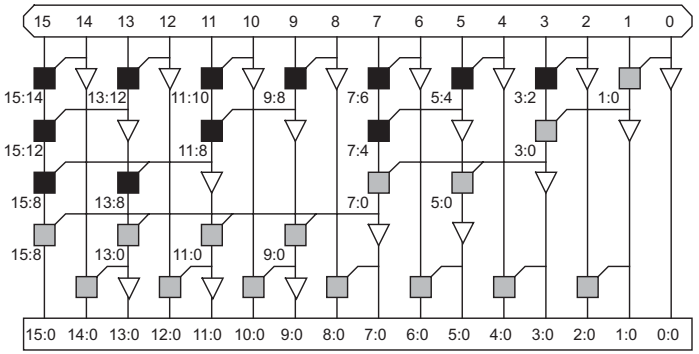


(e) Knowles [2,1,1,1]

Exemplos de somadores log



(c) Kogge-Stone fan-out é sempre 2



(f) Ladner-Fischer

FIGURE 11.29 Tree adder PG networks