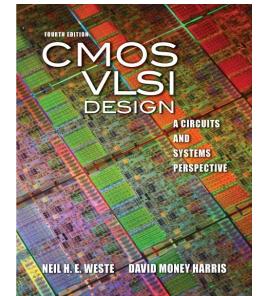
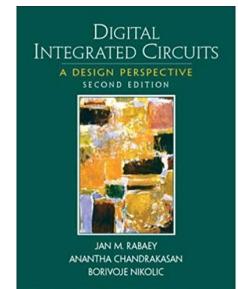


Microeletrônica

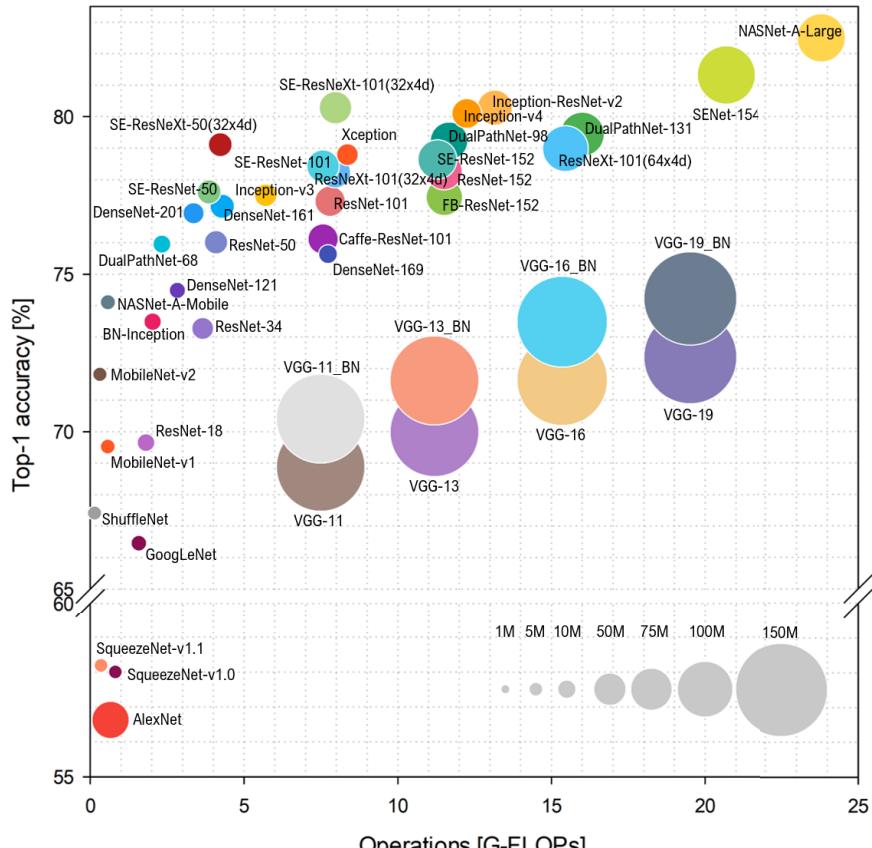
Aula #9 → Circuitos Aritméticos

- Professor: Fernando Gehm Moraes
- Livro texto:
Digital Integrated Circuits a Design Perspective - Rabaey
C MOS VLSI Design - Weste

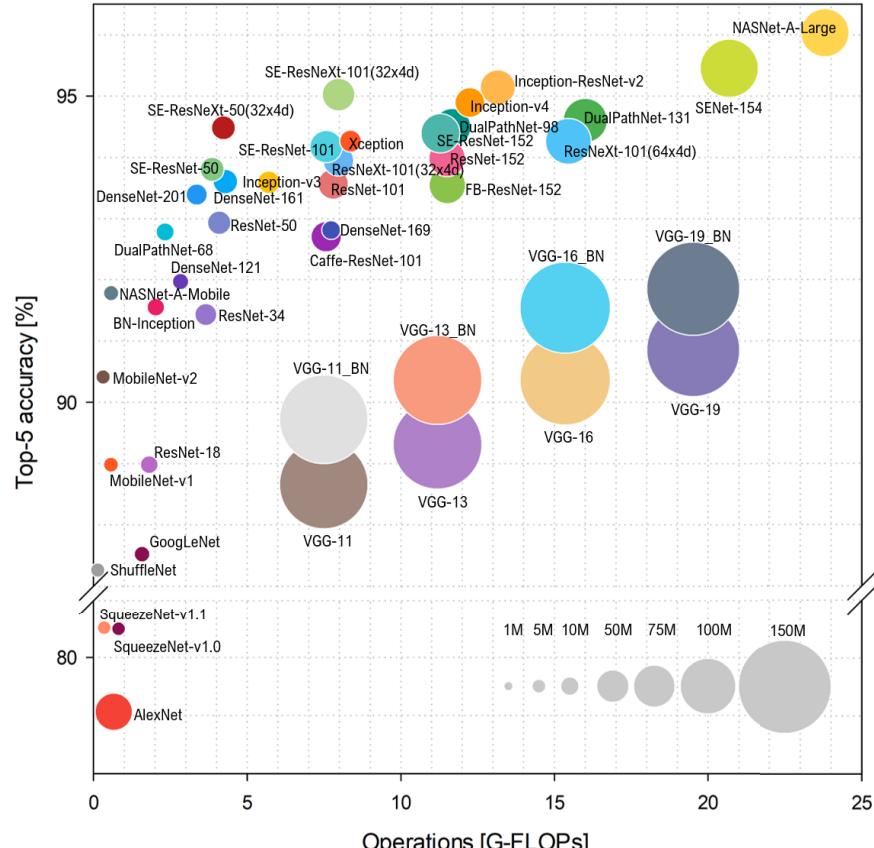


Motivação para importância de circuitos aritméticos

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



(a)



(b)

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.

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)

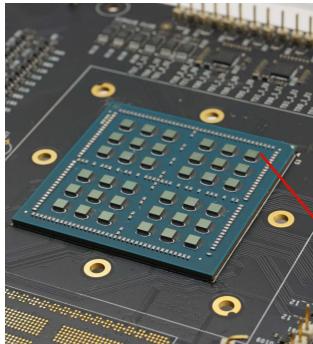


Fig. 16. Prototype system connects 36 chips on the package.

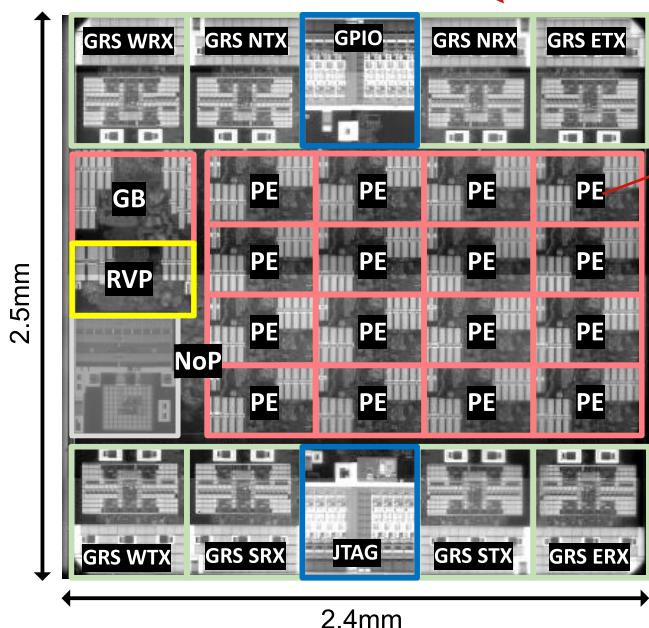


Fig. 14. Chip micrograph annotated with the floorplan of the chip.

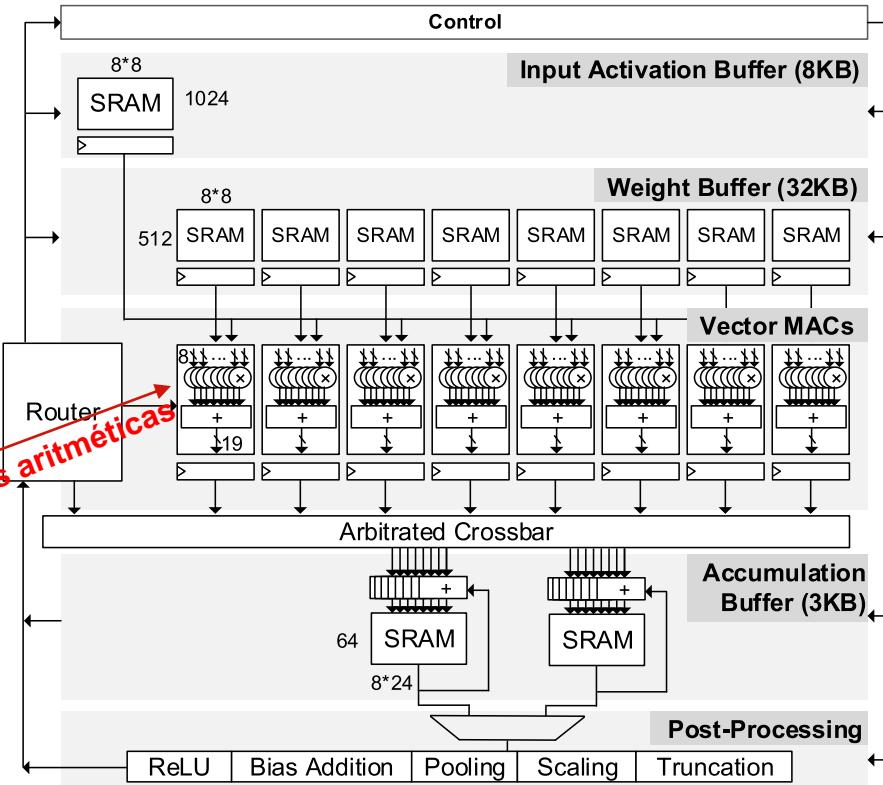
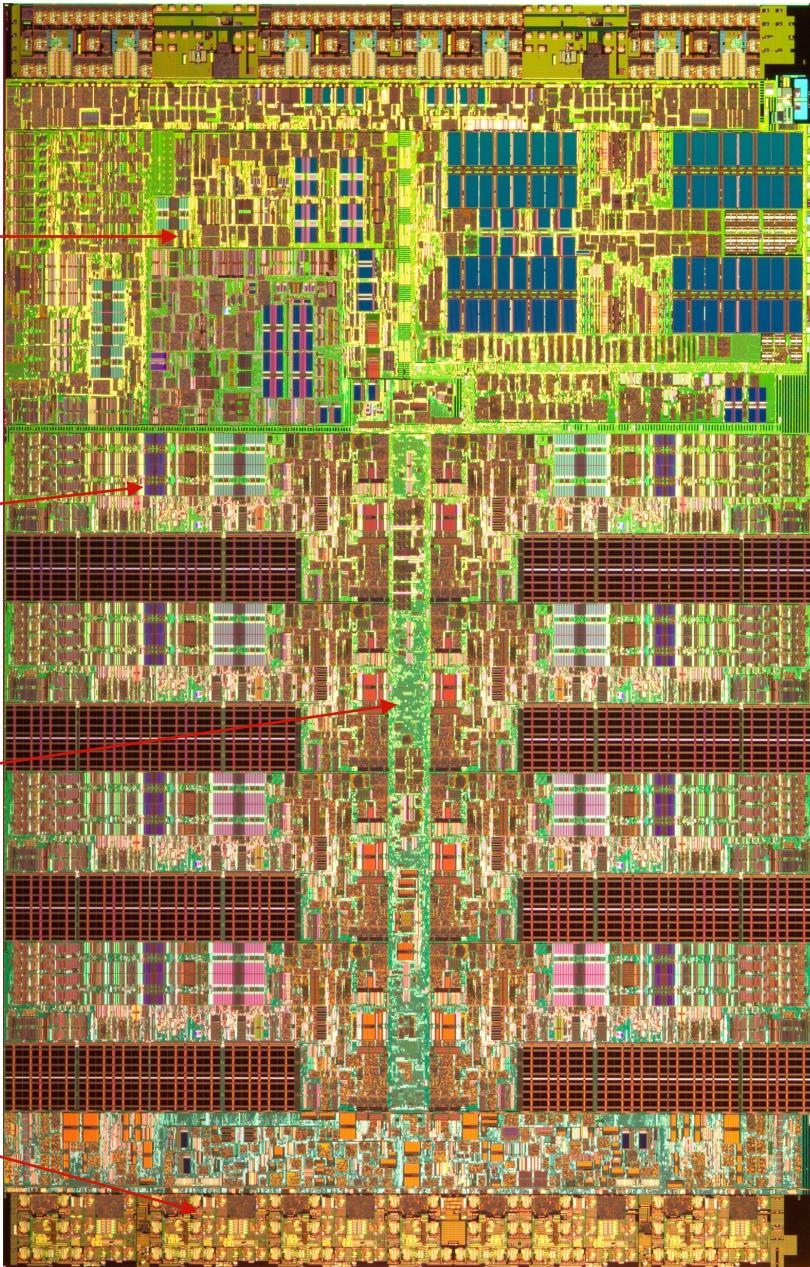


Fig. 8. Architecture of the PE.

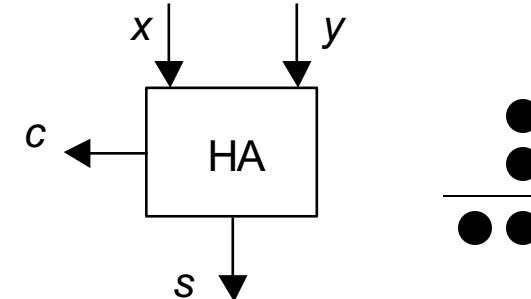
Cell Processor (IBM)

- ❑ 1 PowerPC Processor Element
 - Funções de controle
 - Sistema Operacional
- ❑ 8 Synergistic Processor Elements
 - Arquitetura SIMD 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



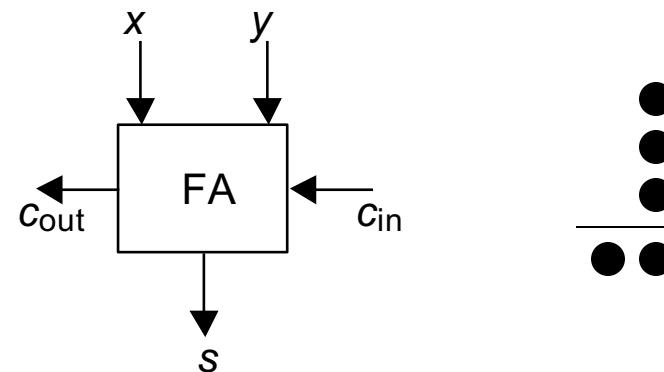
Somador: HA e FA

Inputs		Outputs	
x	y	c	s
<hr/>			
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



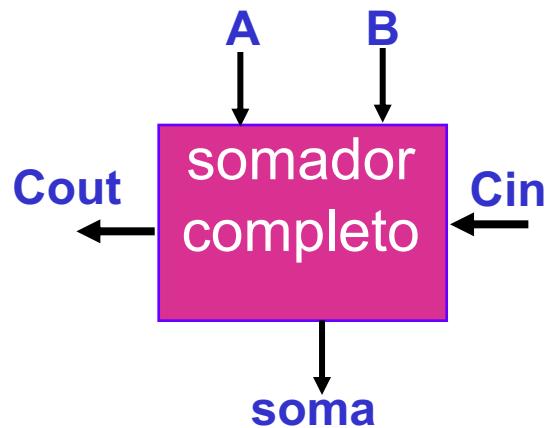
Half-adder (HA): Truth table and block diagram

Inputs			Outputs	
x	y	c_{in}	c_{out}	s
<hr/>				
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)



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

Soma:

a\b\c	00	01	11	10
0		1		1
1	1		1	

Co:

a\b\c	00	01	11	10
0			1	
1		1	1	1

$$S = A \cdot \bar{B} \cdot \bar{C}_i + \bar{A} \cdot B \cdot \bar{C}_i + \bar{A} \cdot \bar{B} \cdot C_i + A \cdot B \cdot C_i$$

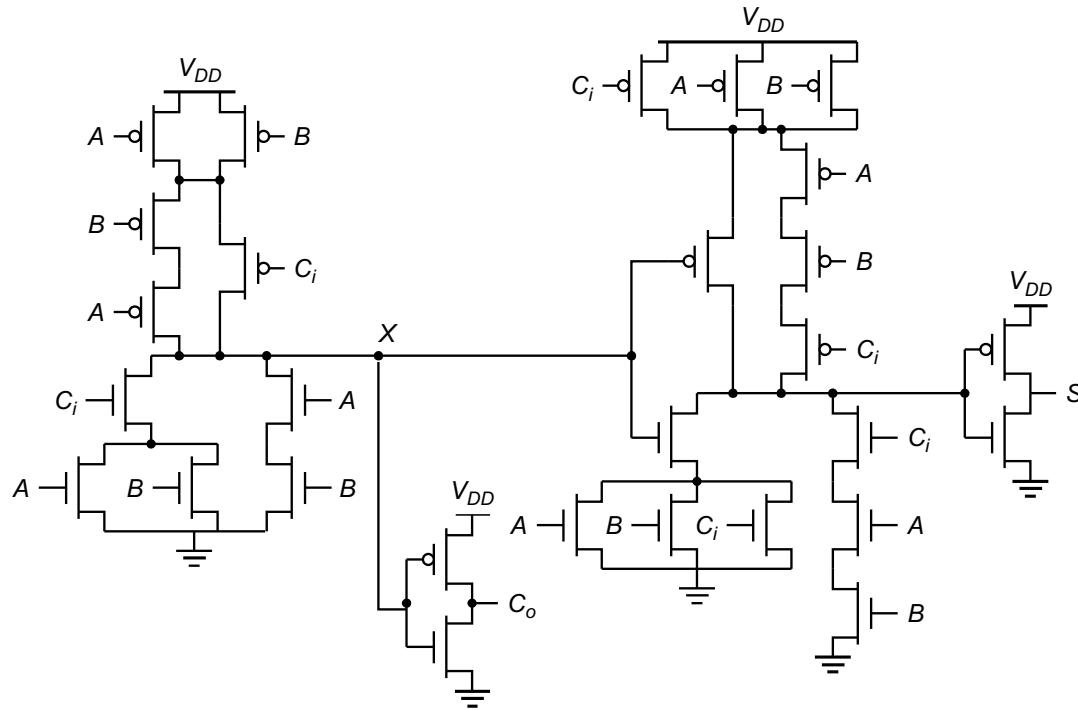
$$S = A \oplus B \oplus C_i$$

$$C_o = A \cdot B + B \cdot C_i + A \cdot C_i$$

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

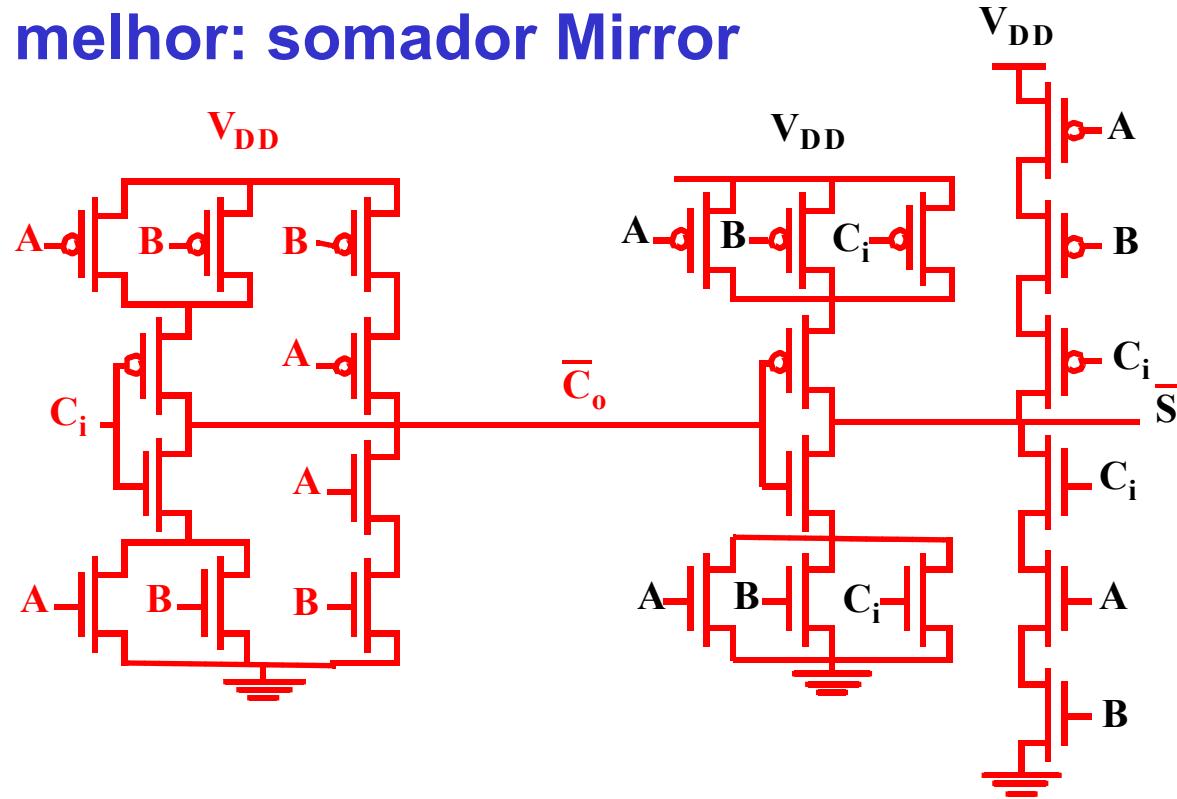
$$C_o = A \cdot B + B \cdot C_i + A \cdot C_i$$

$$\overline{C_o} = \overline{A} \cdot \overline{B} + \overline{A} \cdot \overline{C}_i + \overline{B} \cdot \overline{C}_i$$

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

Diagrama de transistores

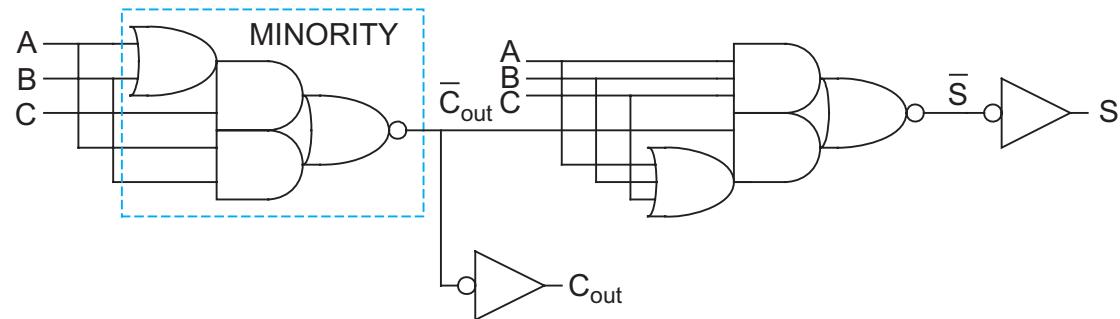
Uma estrutura melhor: somador Mirror



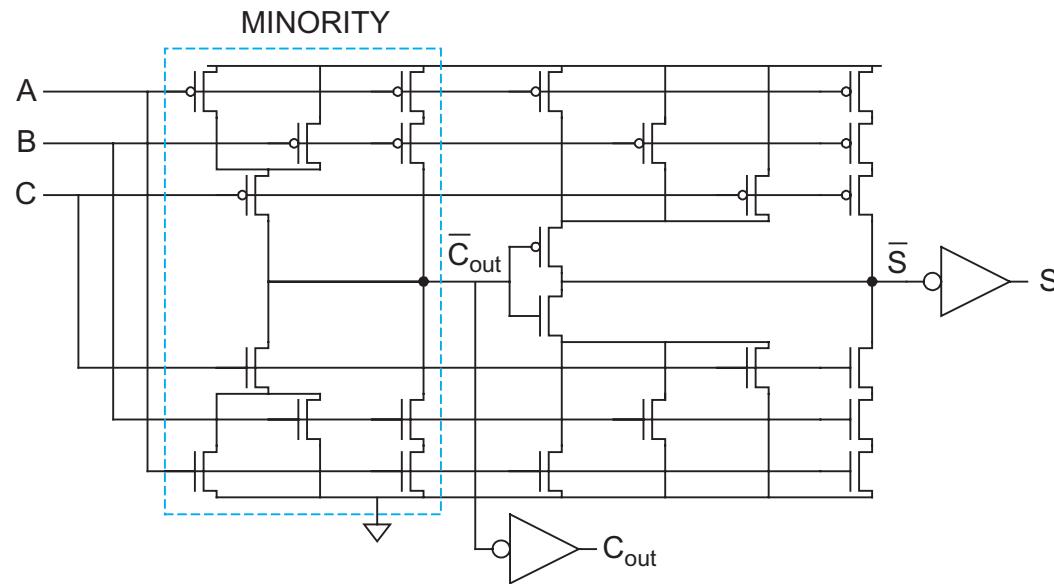
- Planos NMOS e PMOS simétricos
- Tempos de propagação de subida e descida idênticas se os dispositivos NMOS e PMOS estiverem dimensionados corretamente.
- Máximo de dois transistores em série no circuito de geração de carry

Diagrama de transistores

Somador Mirror

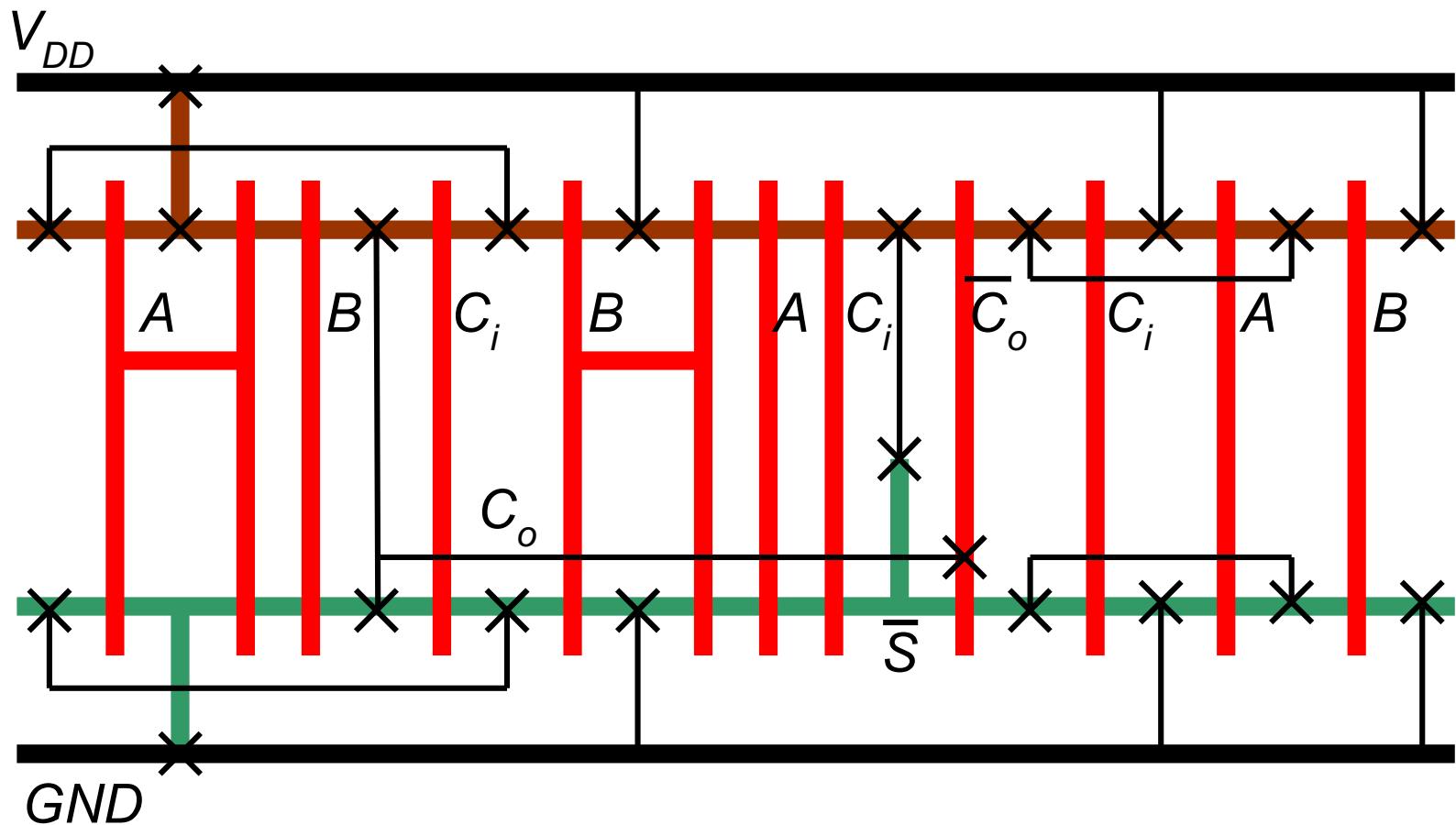


(a)

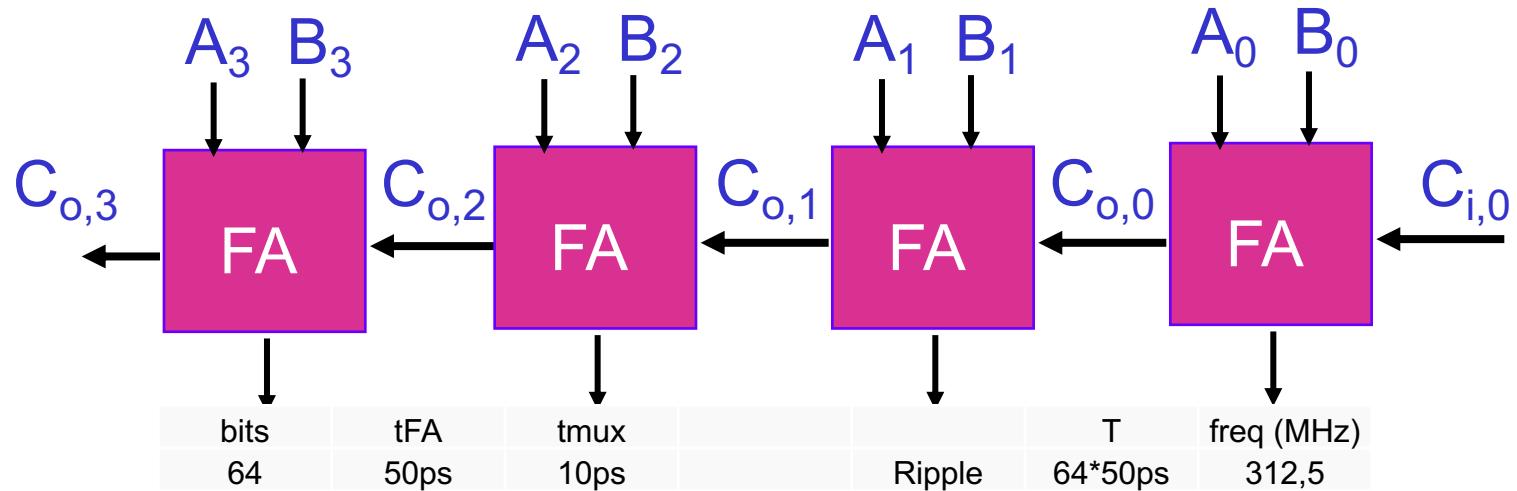


(b)

Somador Mirror – Diagrama Stick



(1) RIPPLE CARRY



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

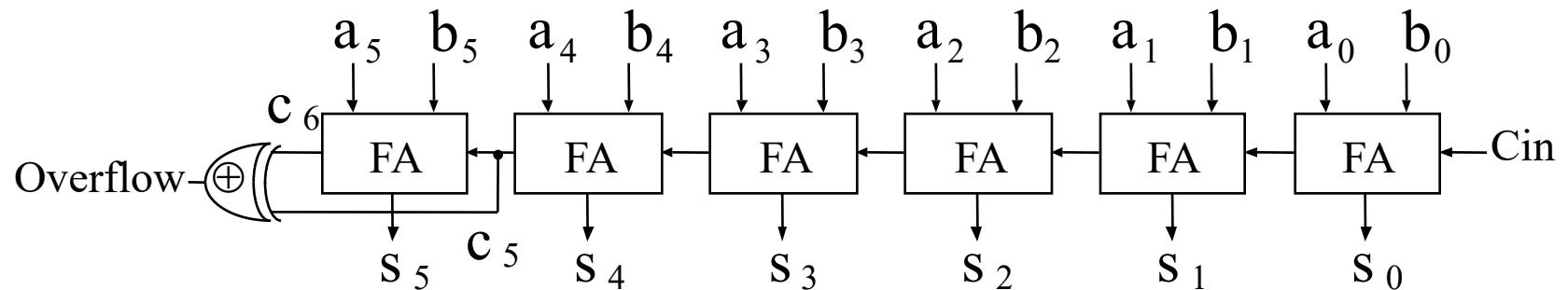
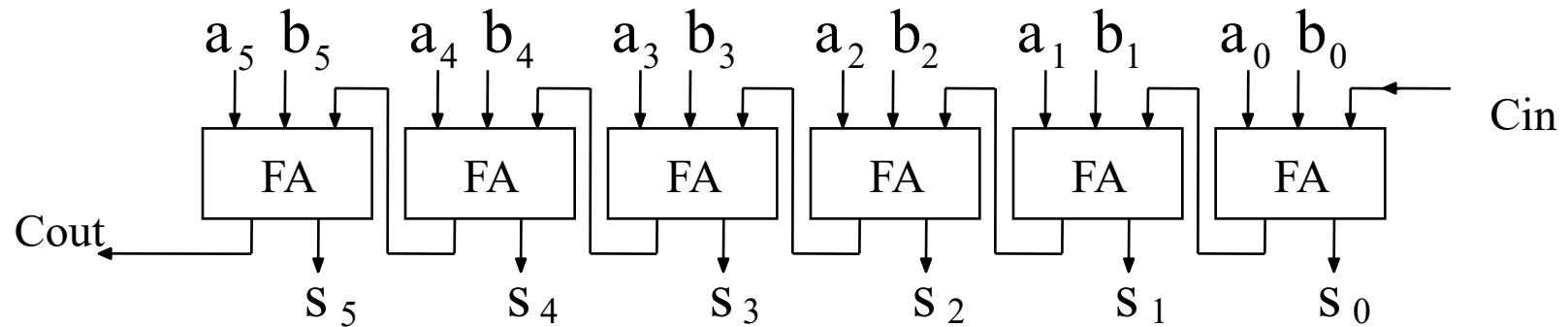
$$T_{\text{adder}} = N \cdot t_{\text{FA}}$$

Exemplo:

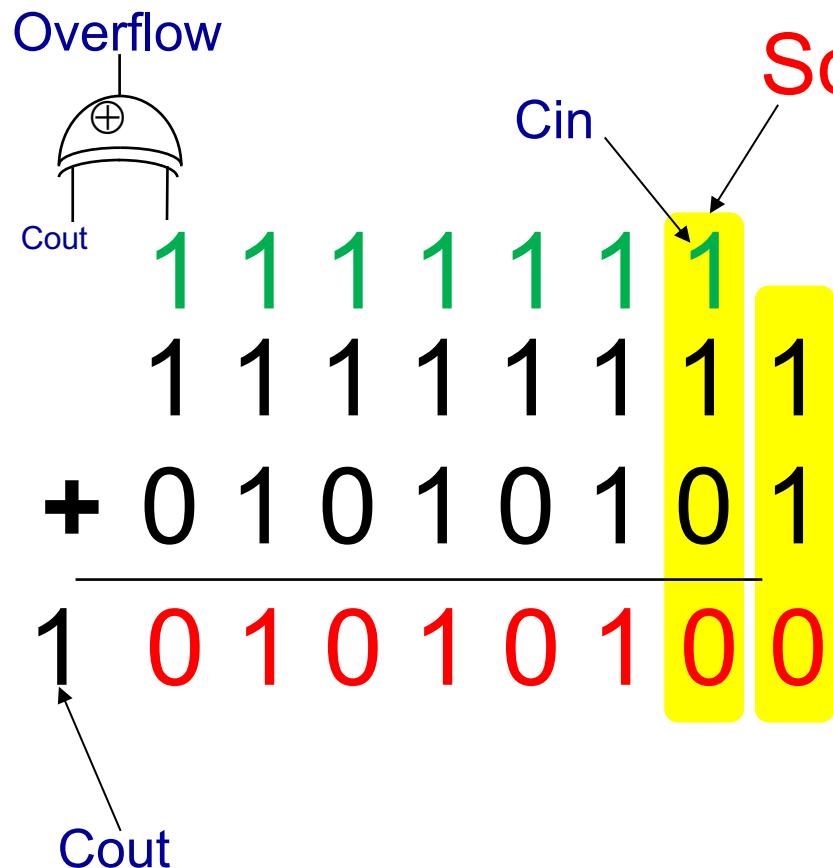
bits	tFA	tmux
64	50ps	10ps

	T	freq (MHz)
Ripple	$64 \times 50\text{ps}$	312,5

Soma - carry out e overflow



Soma - carry out e overflow



Somador completo (FA)

-Meio-somador (HA)

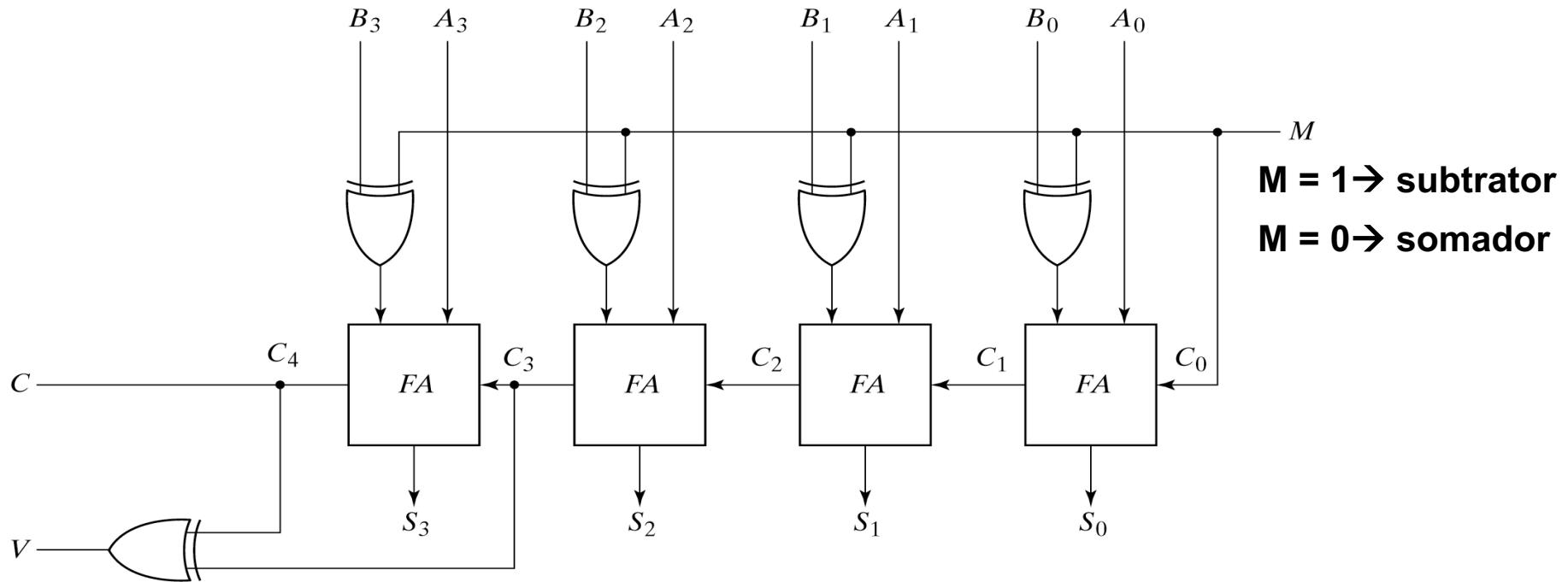
Hexadecimal: FF + 55 = (1) 54

Unsigned: $255 + 55 = 54$ (**errado**, Cout=1)

2's comp: $-1 + 55 = 54$ (**certo** – Overflow=0)

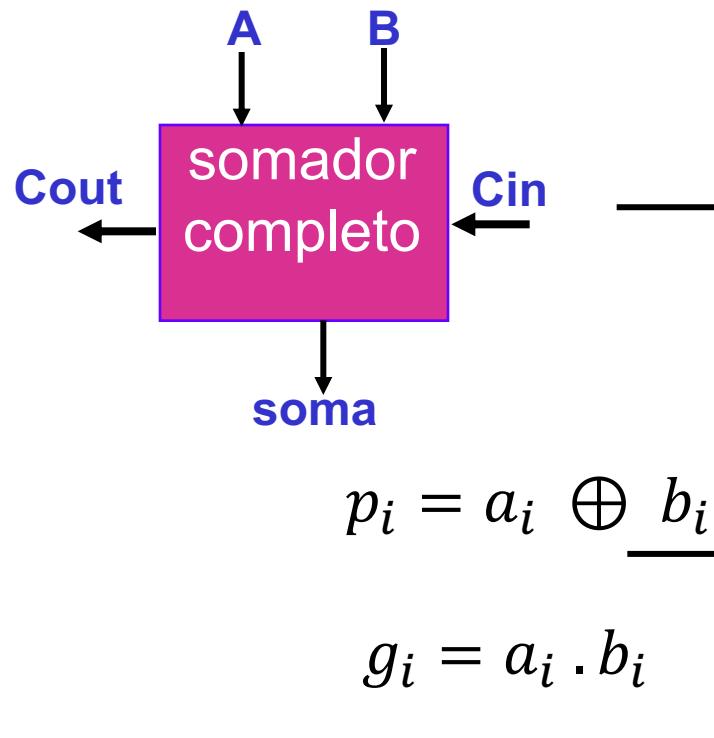
Somador / subtrator

Subtração: $a - b \rightarrow a + \overline{b} + 1$



Propagação e Geração

Propriedades do FA



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

$$p_i = a_i \oplus b_i \quad g_i = a_i \cdot b_i$$

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

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

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

$$C_{out} = c \cdot p + g$$

A	B	C_t	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

$$p_i = a_i \oplus b_i$$

$$g_i = a_i \cdot b_i$$

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

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

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

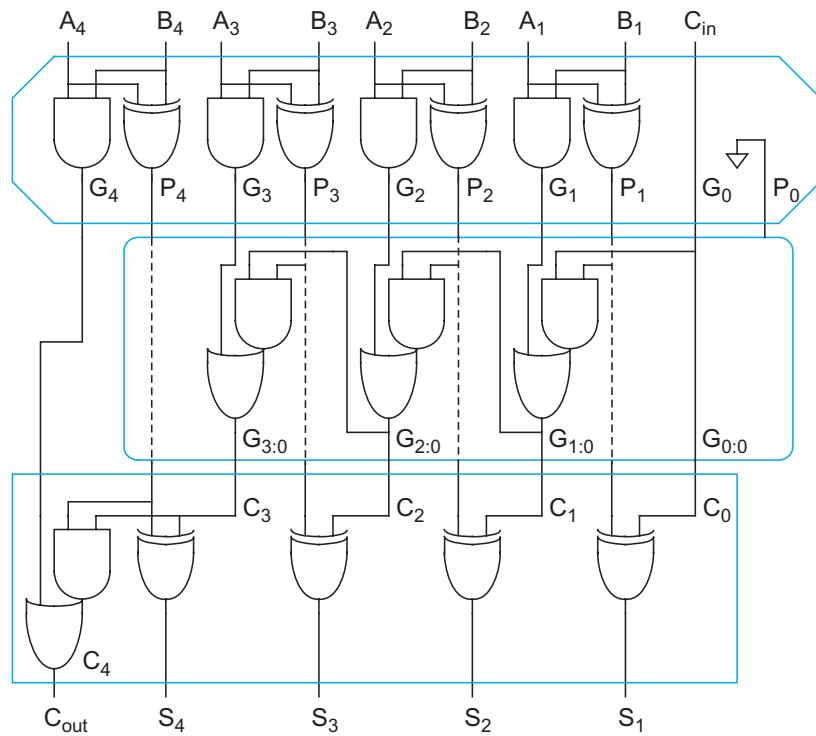


FIGURE 11.14 4-bit carry-ripple adder using PG logic

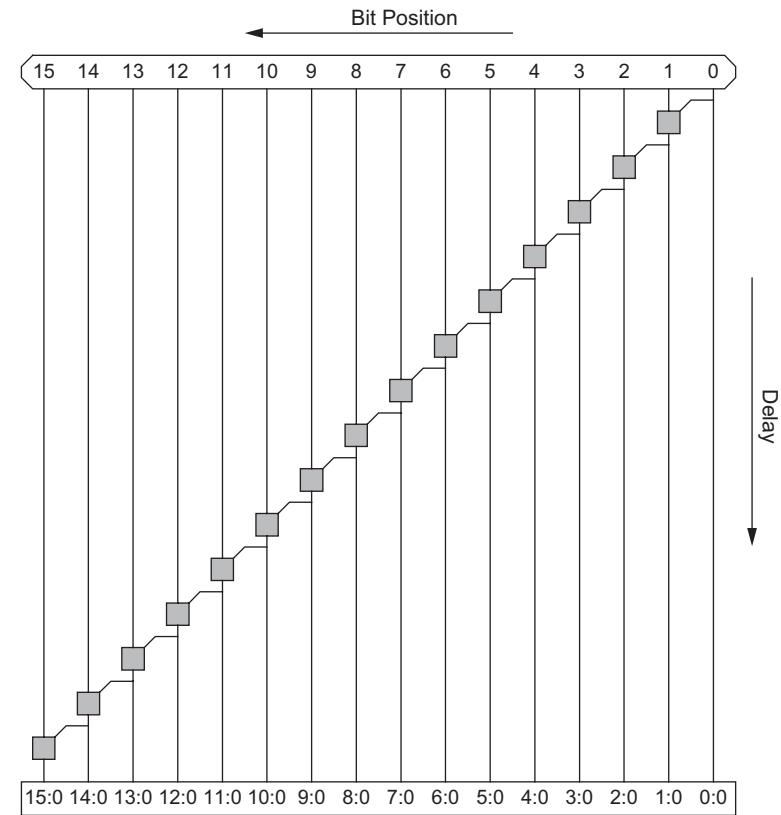
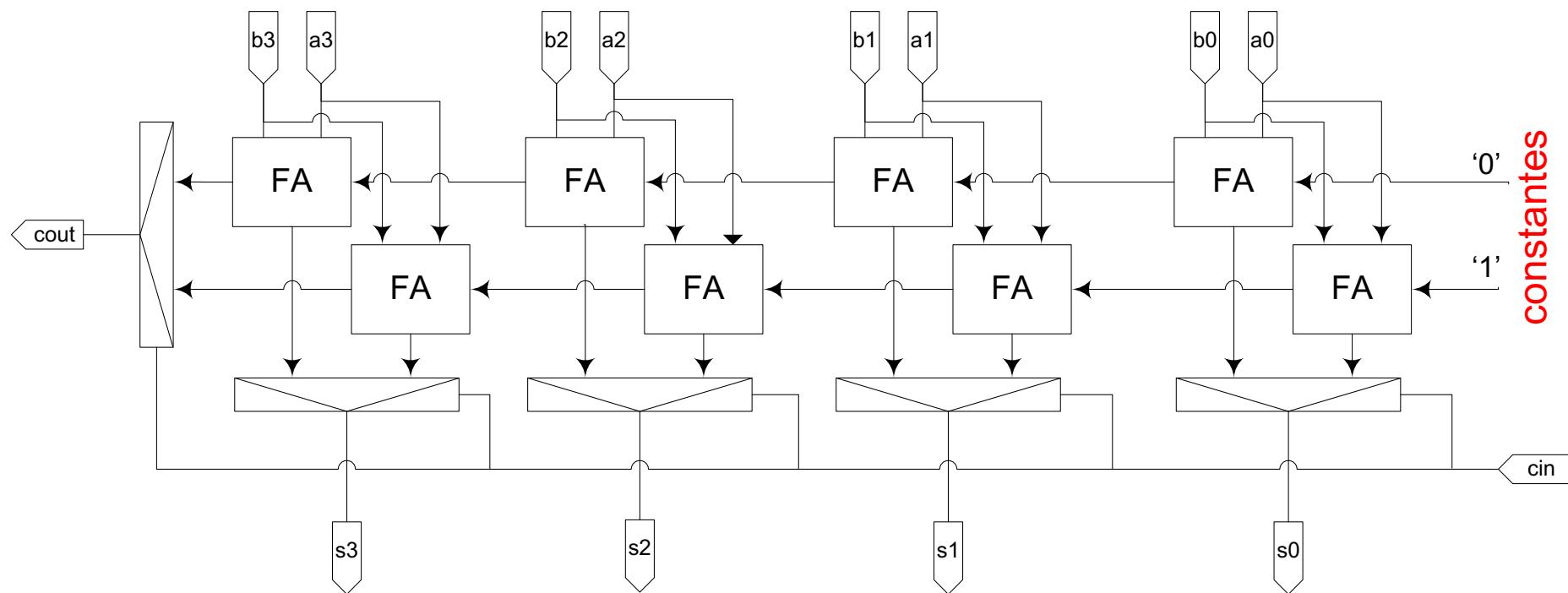


FIGURE 11.15 Carry-ripple adder group PG network

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

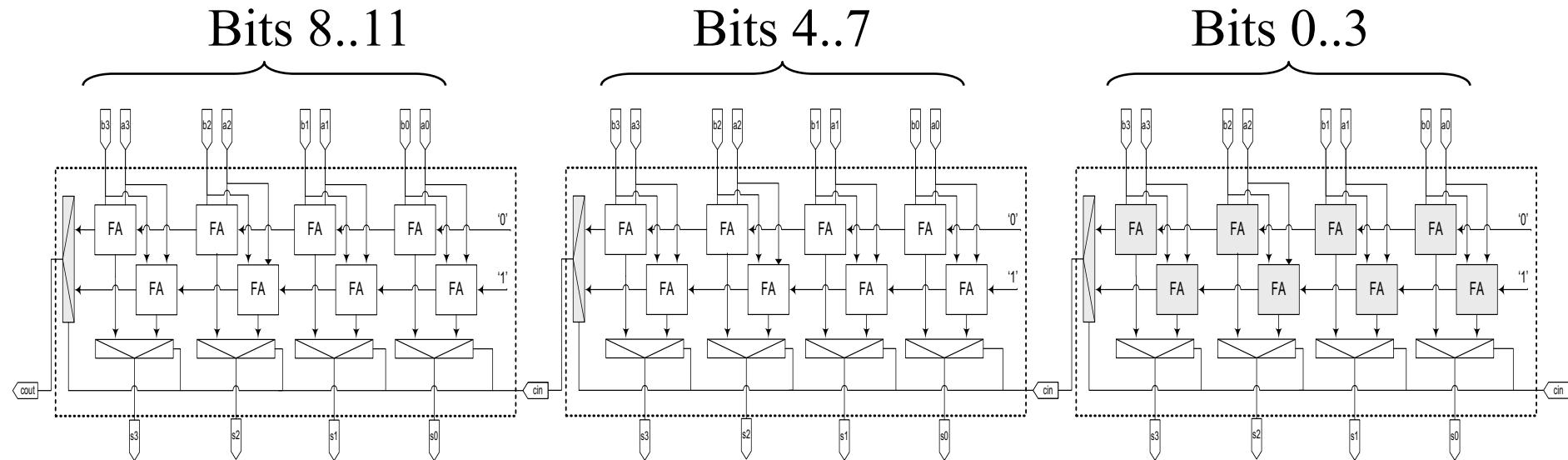
Duplica a geração da soma e do carry para cada bit



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

Carry Select Adder

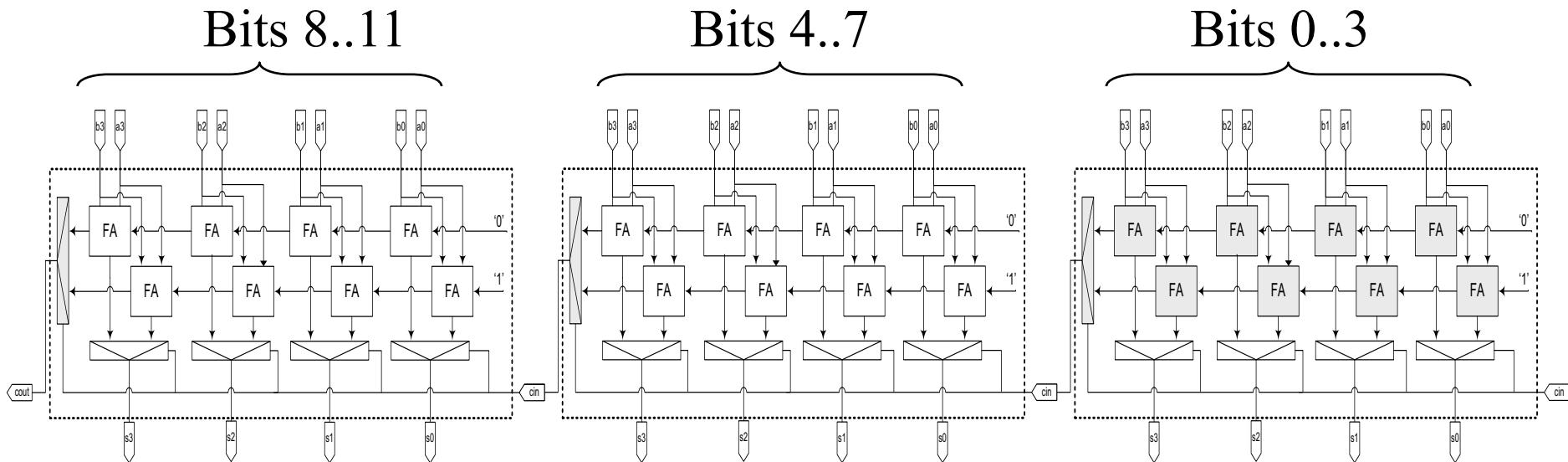
Carry select adder – 3 estágios de 4 bits



- **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



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

Exemplo:

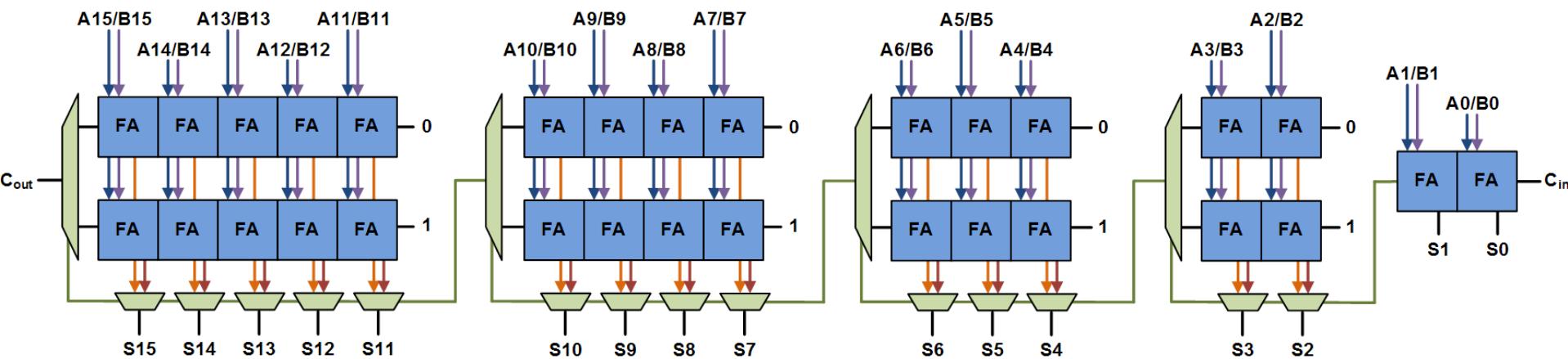
bits	tFA	tmux
64	50ps	10ps

Simplificando:

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

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

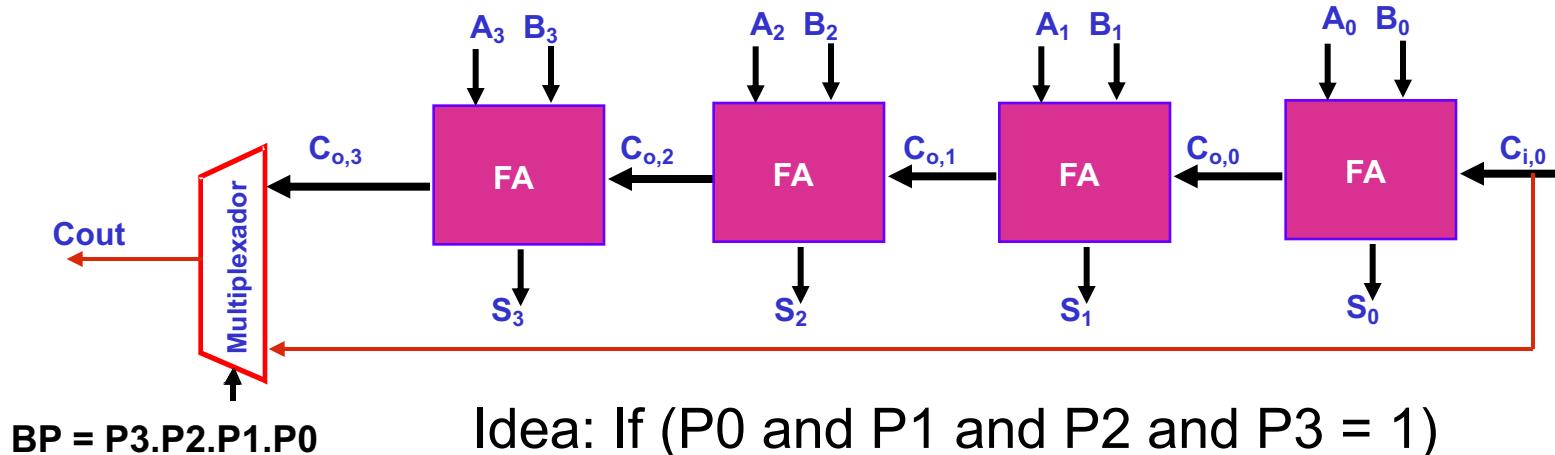
Square Root Carry Select



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

(3) Somador Carry-Bypass

Bloco básico



Se não há propagação o carry é gerado!

$$p_i = a_i \oplus b_i$$

$$g_i = a_i \cdot b_i$$

$$C3 = G3 + P3.C2$$

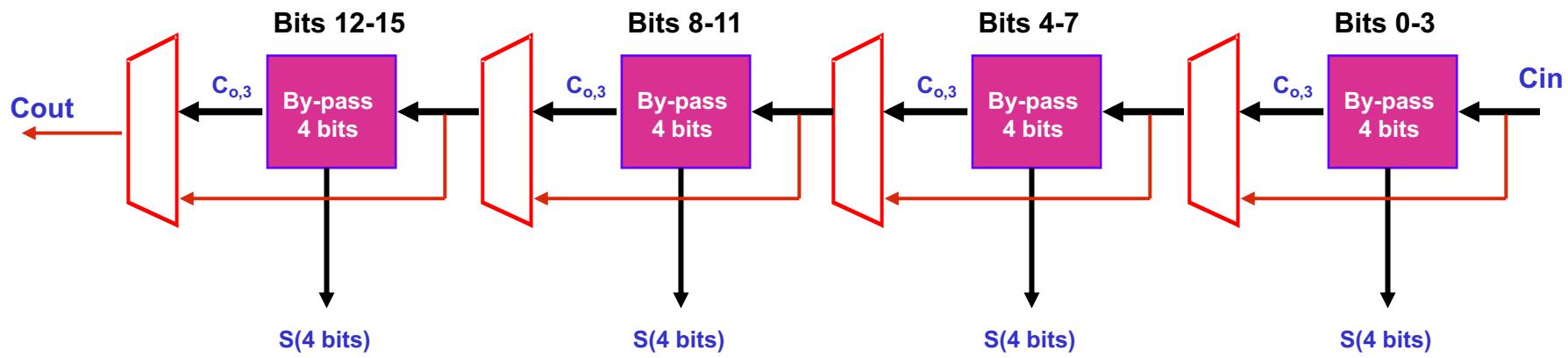
$$C3 = G3 + P3.(G2 + P2C1)$$

....

$$C_{out} = c \cdot p + g \quad C3 = G3 + P3G2 + P3P2G1 + P3P2P1G0 + \mathbf{P3P2P1P0Cin}$$

Somador Carry-Bypass

Construção do bypass de n bits

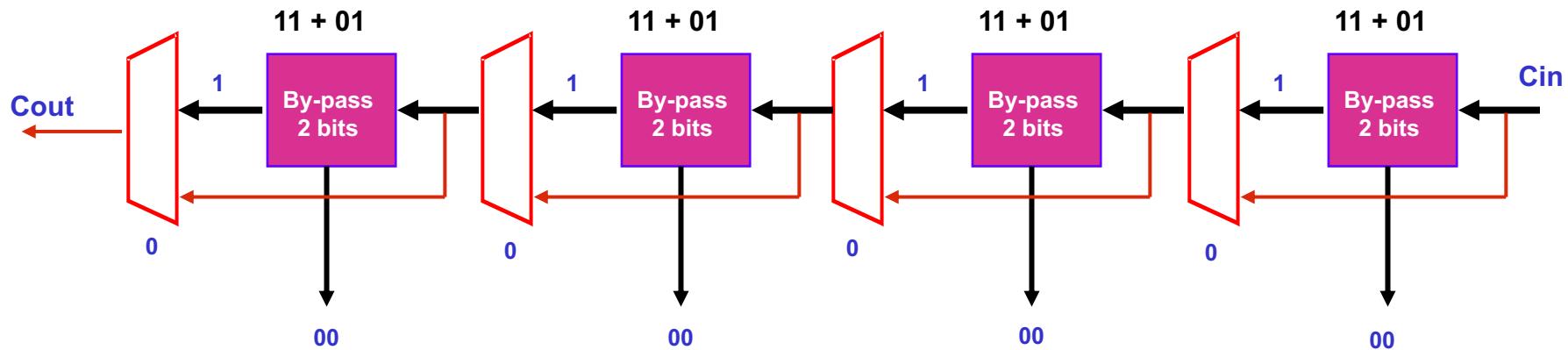


Propagação do vai-um só ocorre no primeiro e último estágio

Somador Carry-Bypass

$$\begin{array}{r} 11 \ 11 \ 11 \ 11 \\ + \ 01 \ 01 \ 01 \ 01 \end{array}$$

Tempo 0

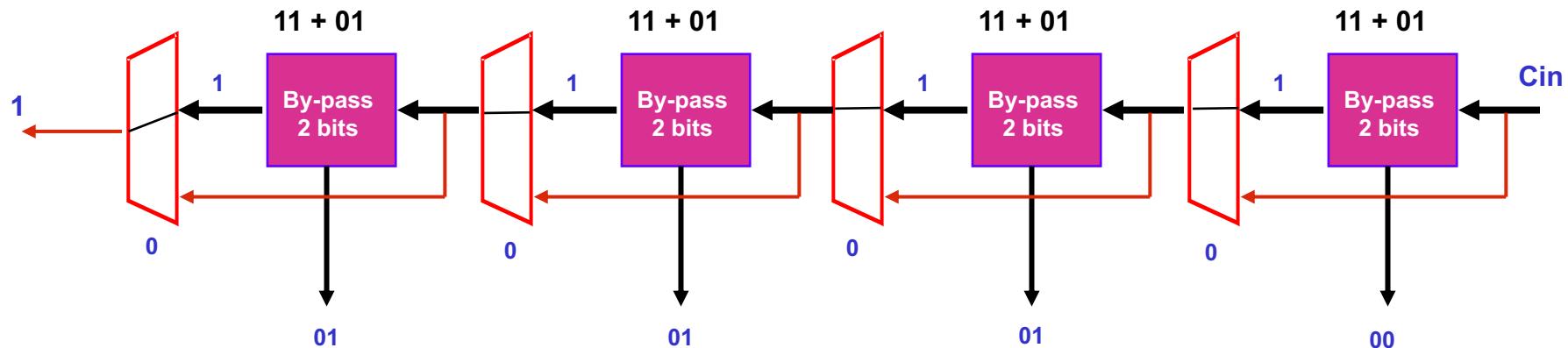


As soma ocorrem sem o carry

Somador Carry-Bypass

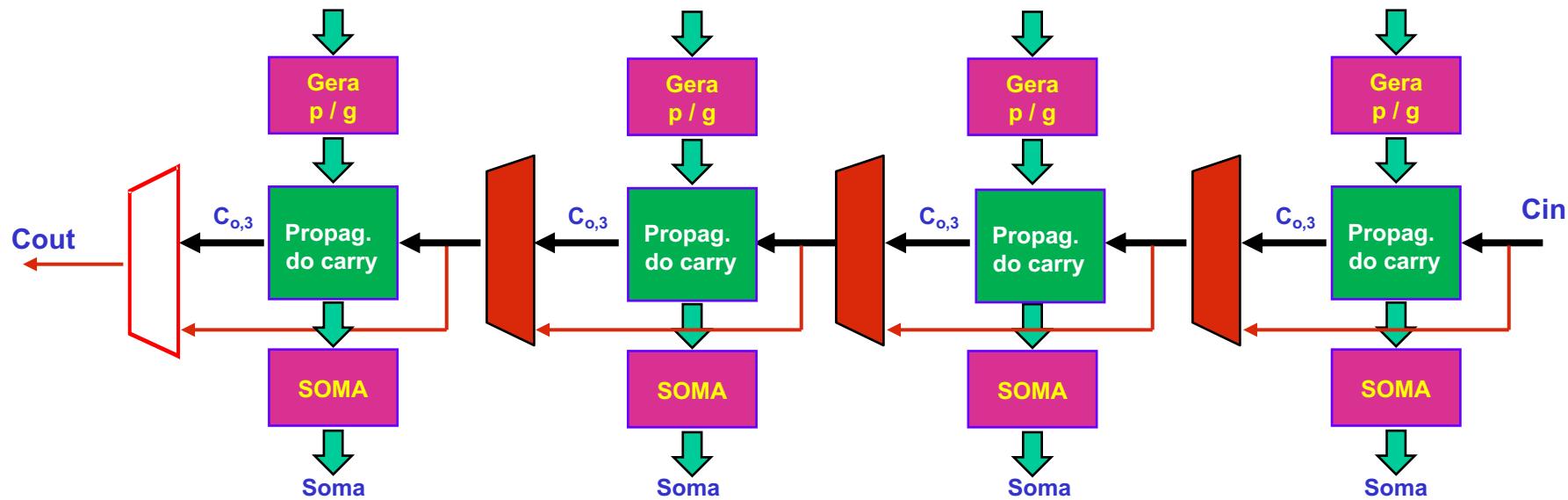
$$\begin{array}{r} 11 \ 11 \ 11 \ 11 \\ + \ 01 \ 01 \ 01 \ 01 \end{array}$$

Tempo 1



O carry é considerado

Somador Carry-Bypass (página 572 livro texto)



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

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

bits	tFA	tmux
64	50ps	10ps

Simplificando:

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

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

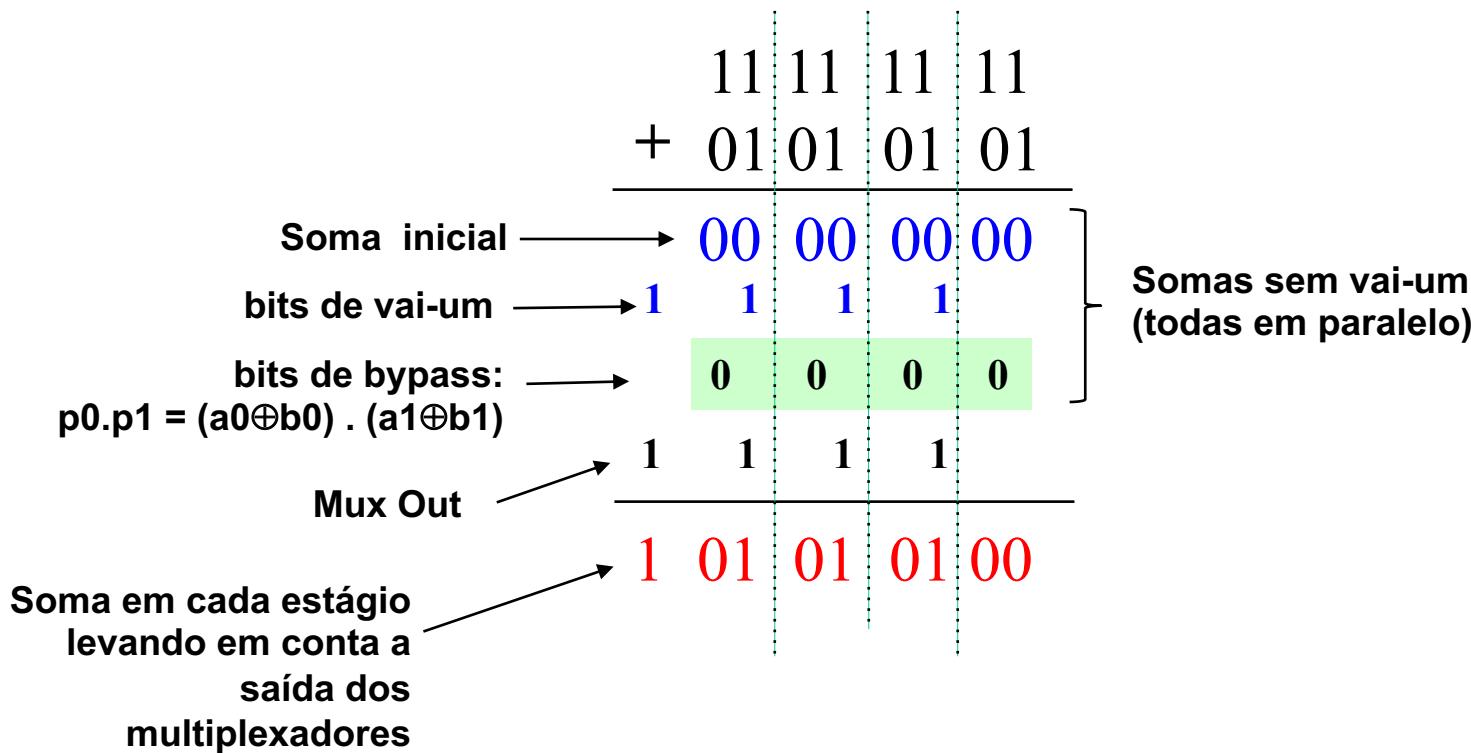
Somador Carry-Bypass

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

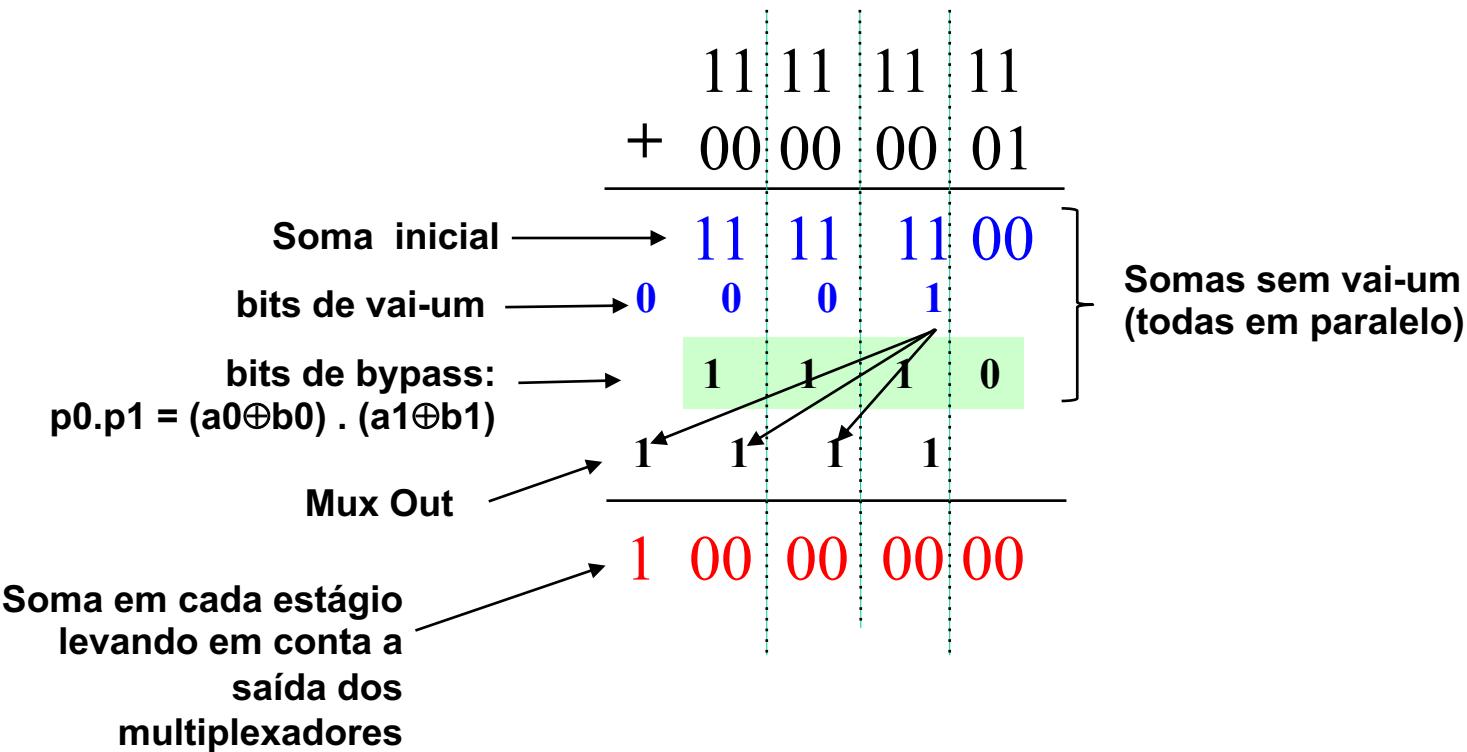
- Não há nenhum bit de bypass
- O ganho está no fato que o *vai-um* de **cada estágio** é calculado em paralelo



Somador Carry-Bypass

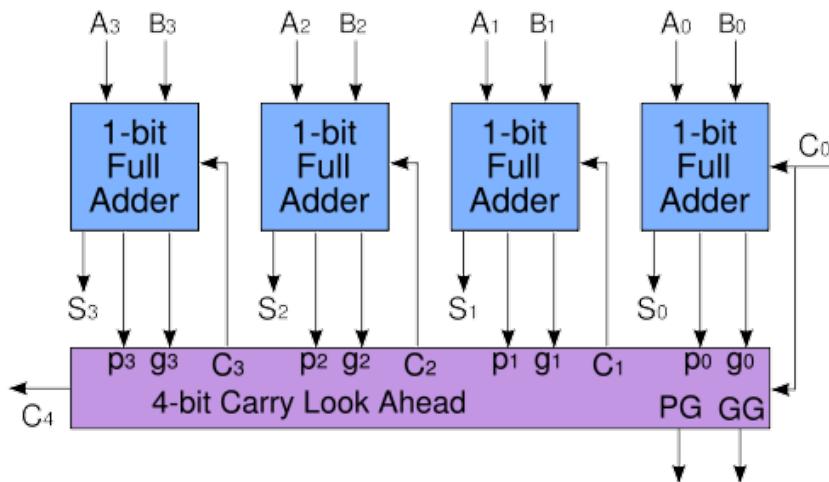
Exemplo 2 (supondo estágio de dois bits):

- 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



$$C_0 = \text{cin}$$

$$C_1 = c_0 p_0 + g_0$$

$$C_2 = c_0 p_0 p_1 + g_0 p_1 + g_1$$

$$\begin{aligned} C_3 &= c_2 p_2 + g_2 \\ &= c_0 p_0 p_1 p_2 + g_0 p_1 p_2 + g_1 p_2 + g_2 \end{aligned}$$

$$\begin{aligned} C_4 &= c_3 p_3 + g_3 \\ &= c_0 p_0 p_1 p_2 p_3 + g_0 p_1 p_2 p_3 + g_1 p_2 p_3 + g_2 p_3 + g_3 \end{aligned}$$

- Na prática: ripple carry de 4 bits e cálculo de C_4

Carry Look-Ahead

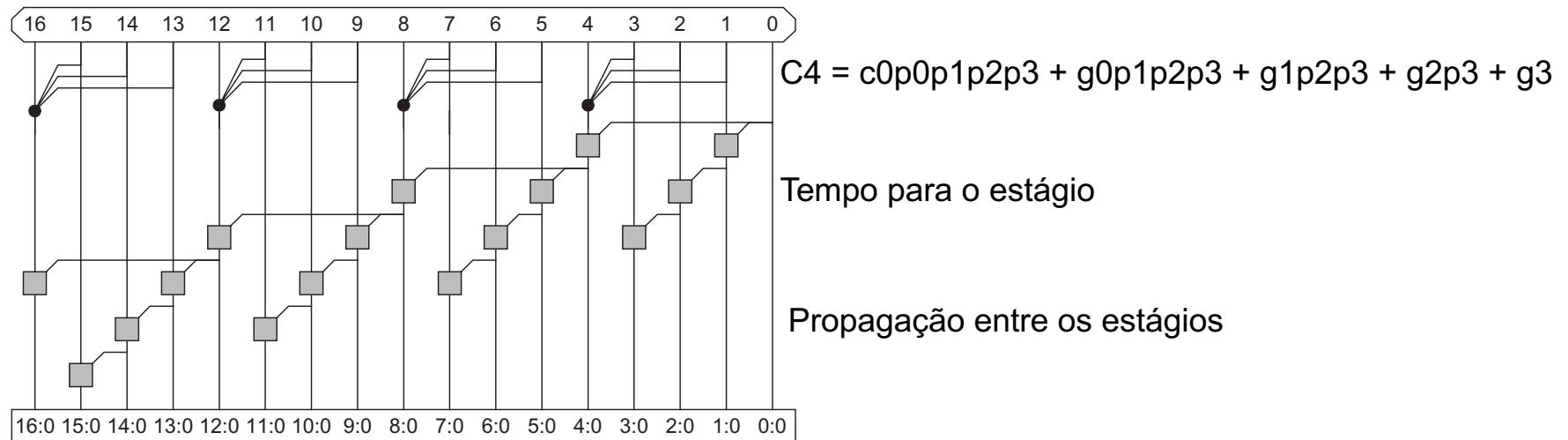


FIGURE 11.22 Carry-lookahead adder group PG network

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: $T = N \cdot t_{FA}$

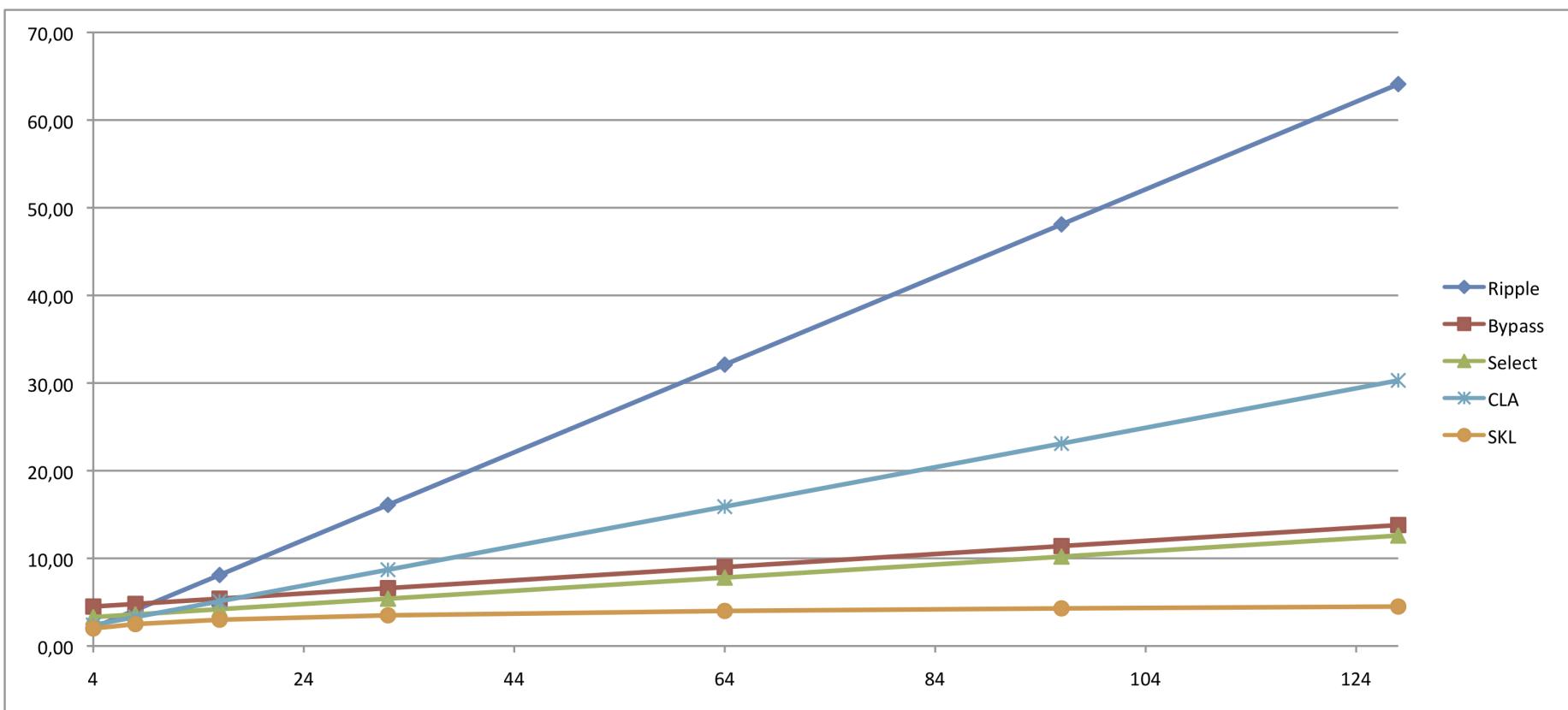
Carry select: $T = M \cdot t_{FA} + \left(\frac{N}{M} \right) \cdot t_{mux}$

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

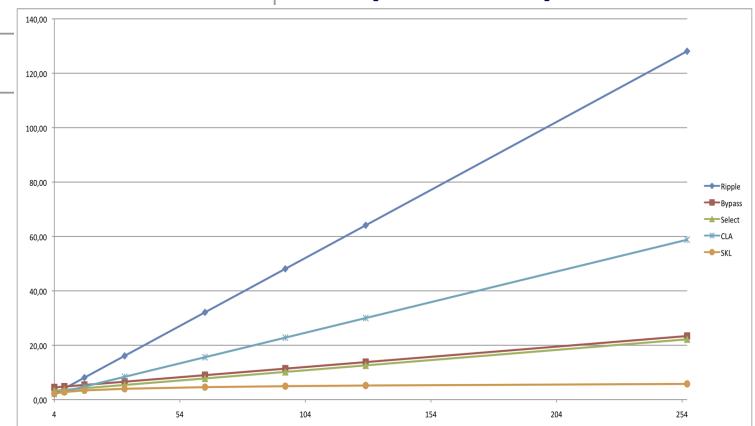
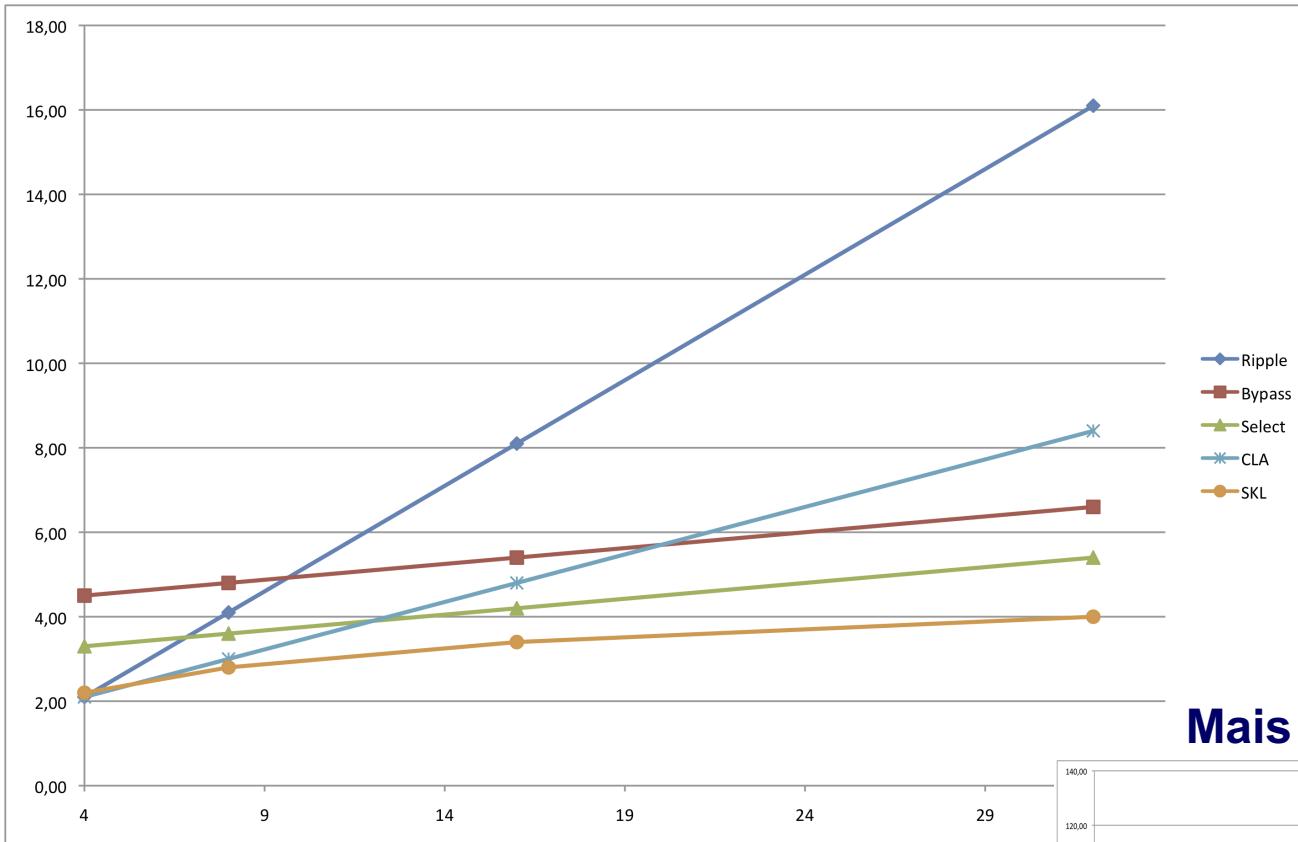
CLA: $T = t_{setup} + M \cdot t_{FA} + \left(\frac{N}{M} - 1 \right) \cdot t_{CLA}$

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

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



Fazendo um zoom



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

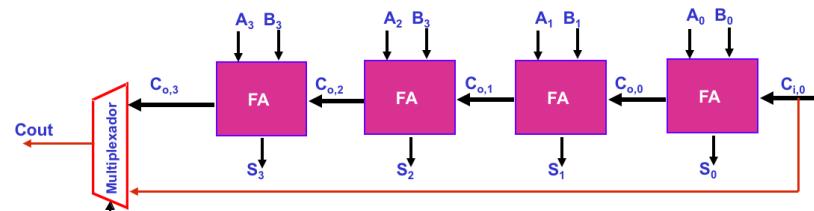
Ripple: 28 transistores * número de bits do somador

Bypass: bloco básico de 4 bits: $32 \times 4 + 6(\text{mux}) + 8(\text{nand}) = 142$

Select: $[(28(\text{FA})+28(\text{FA})+6(\text{MUX})) \times 4 + 6(\text{mux})] = 254$

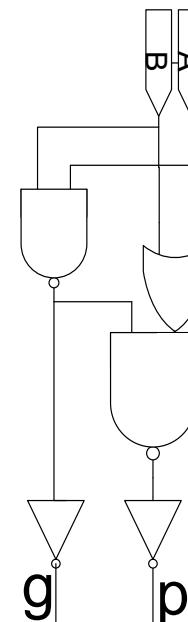
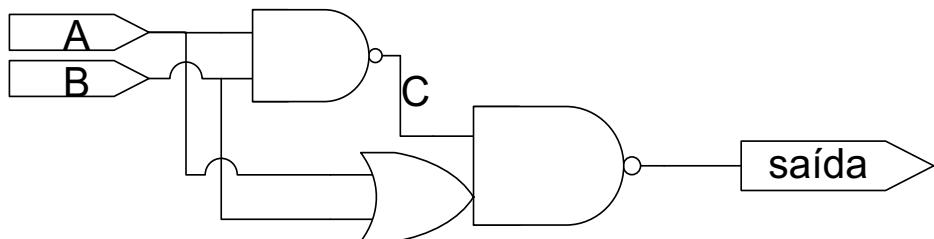
CLA: bloco básico: $28 \times 4 + 20 = 132$ (supondo que só há célula de vai-um para geração de vai-um na saída do estágio)

Número de transistores Carry-Bypass

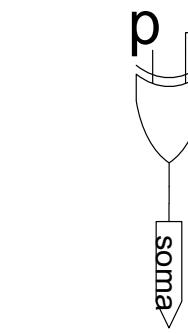


$$p = A \oplus B_i \quad g = A \cdot B$$

XNOR



$$Cout = Cin.p + g$$



Bypass: bloco básico de 4 bits:

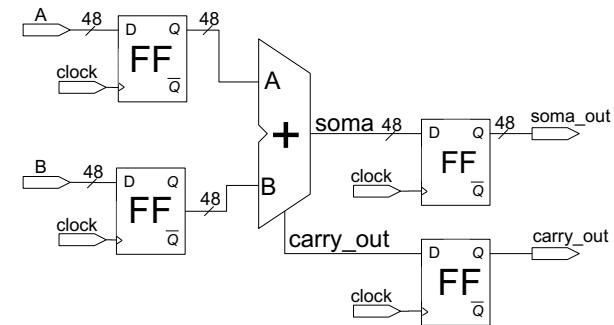
$$32*4 + 6(\text{mux}) + 8 (\text{nand 4 entradas}) = 142$$

10 transistores

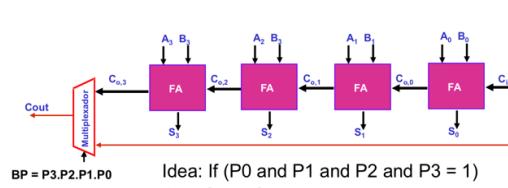
Exercício

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:

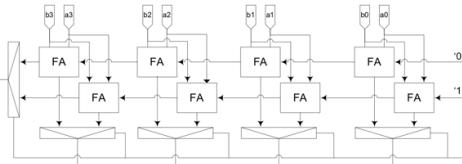
- Flip-flop D, com tempo de propagação D \rightarrow Q igual a 25ps e o tempo de setup igual a 5ps.
- Somador completo (FA), com tempo de propagação igual a 90ps.
- Multiplexador, com tempo de propagação igual a 20ps.



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 \rightarrow 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? Justifique a resposta em função da frequência de operação e do custo em área do somador de 48 bits.

Circuito com somador ripple-carry, carry-bypass e carry-select.

	FF $D \rightarrow Q$ (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	0,230 0,2
Circuito com somador carry-bypass	25	5	$= 12 * (4 * 32 + 6 + 8)$ 0,4 $= 1704$	0,4	0,1 990	1,010 0,2
Circuito com somador carry-select	25	5	$= 12 * (8 * 28 + 30)$ 0,4 $= 3048$	0,4	0,1 630	1,587 0,2

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

Tempo dos somadores:

$$t_{FA} = 48 \text{ bits} * 90s = 4.320$$

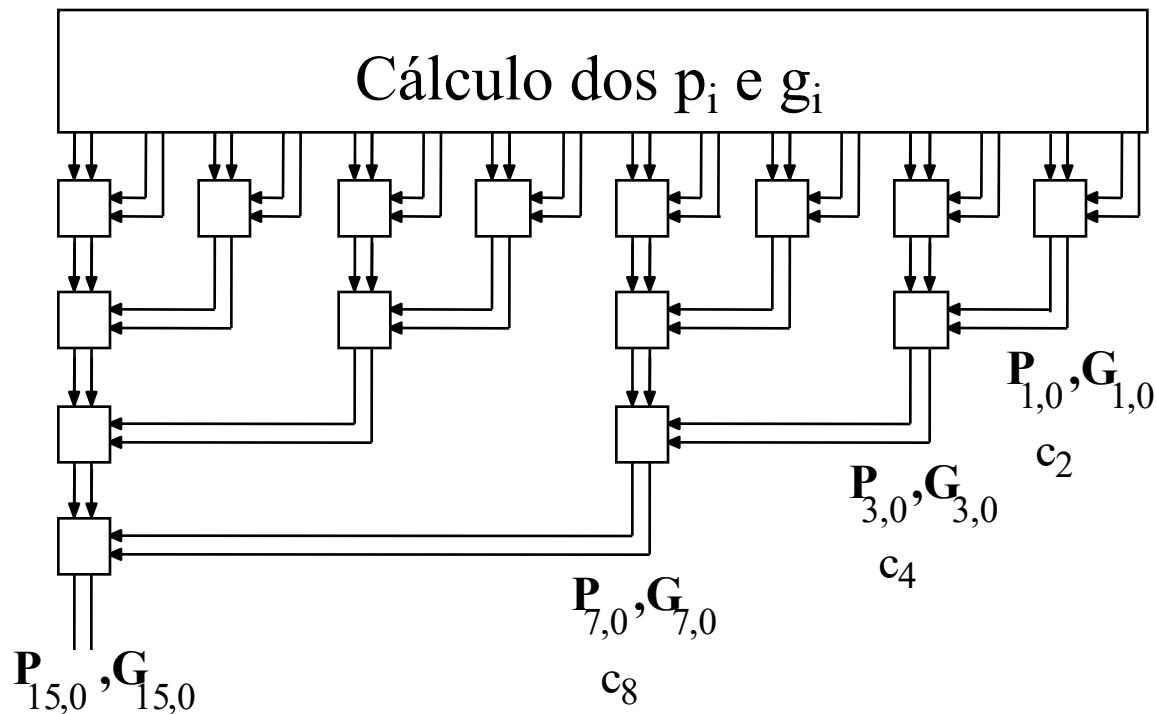
$$t_{BP} = (48/4) * 20 + 2 * (4 * 90) = 960$$

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

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

Somador Logarítmico

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



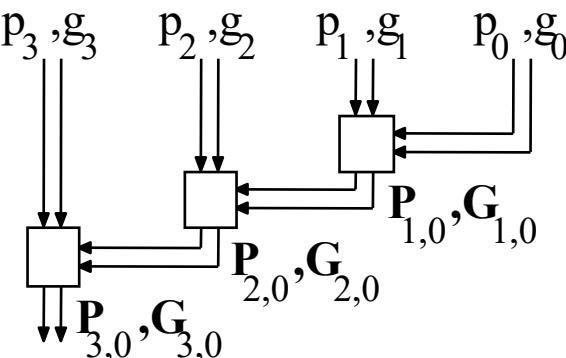
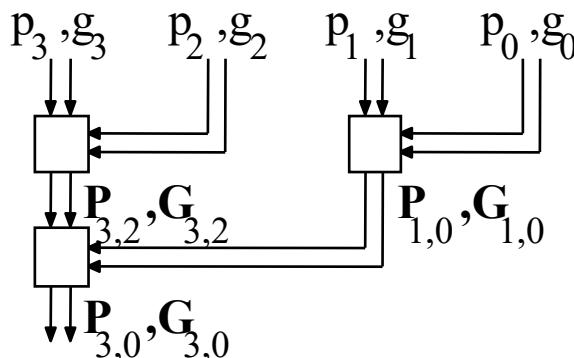
Célula de Brent et Kung

Nome dado ao conjunto de portas lógicas que calcula:

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

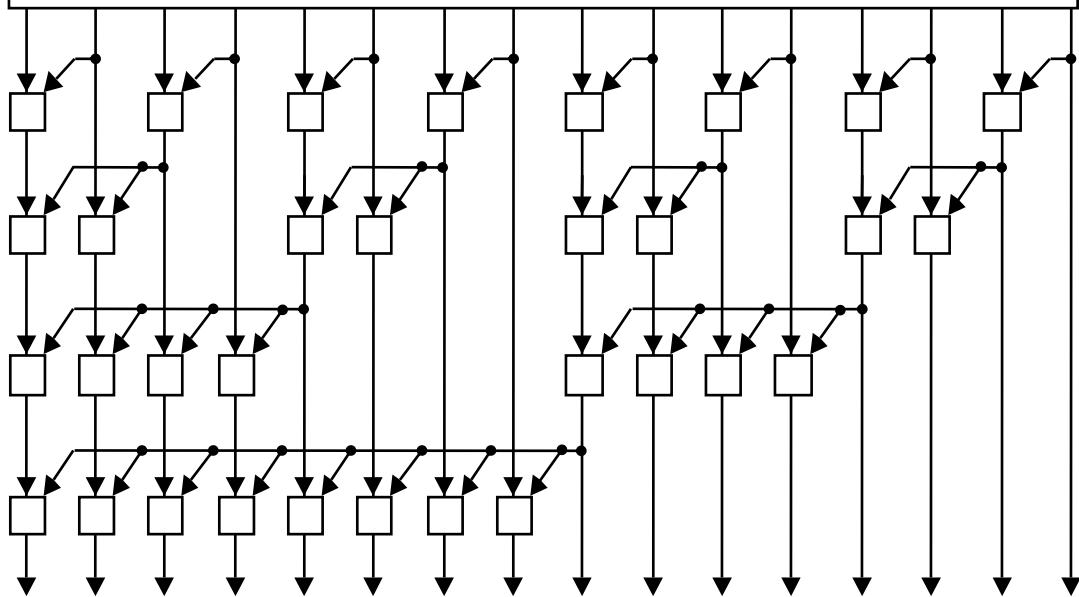
$$P_{i,k} = P_{i,j} \cdot P_{j-1,k} \quad (\text{propagação do vai-um entre bits } i \text{ e } k)$$

Exemplos de combinações:



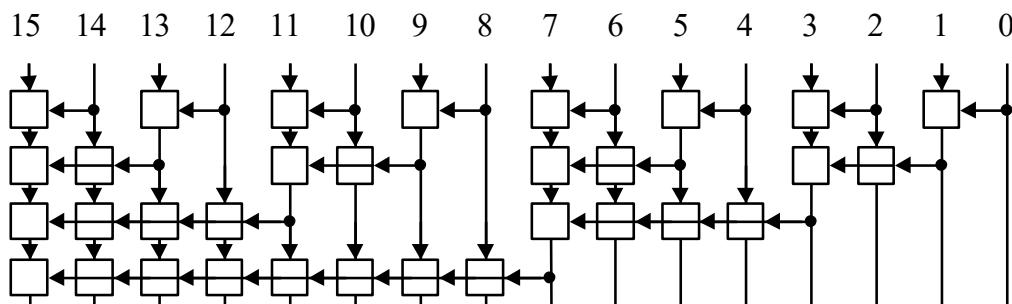
Somador de Sklansky em tempo log2(n)

Cálculo dos p_i e g_i



Geração das somas

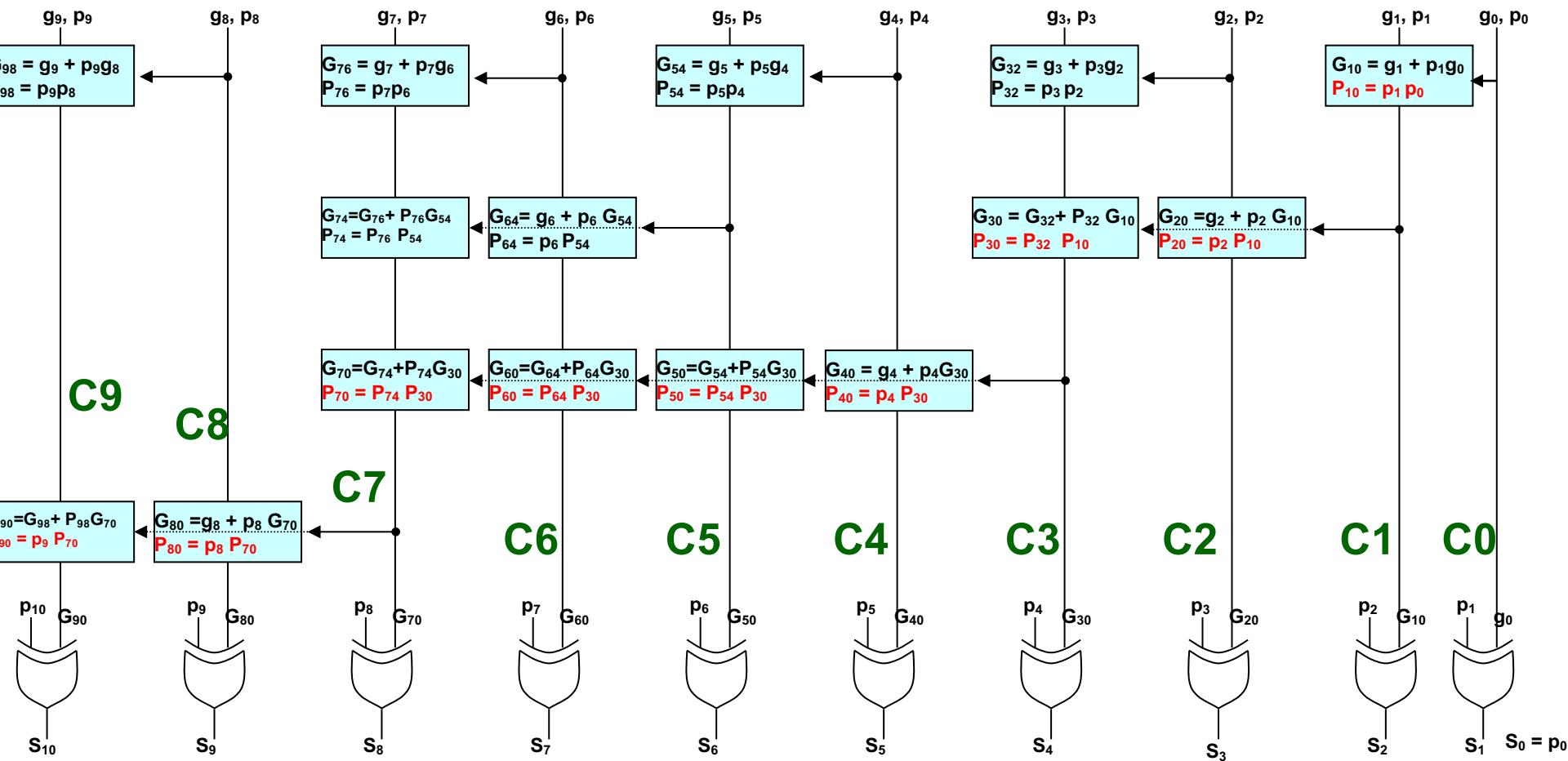
$$s_i = p_i \oplus G_{i-1,0}$$



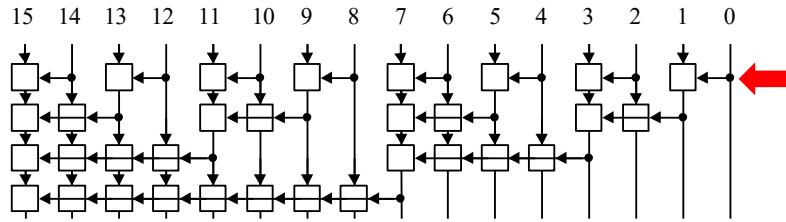
Sklansky – bits 0 a 10

$$G_i = a_i \cdot b_i$$

$$P_i = a_i \oplus b_i$$



Detalhando o somador – carry do bit 0



□ $C_{in} = 0$

Equação do primeiro carry:

$$C_0 = a_0 b_0 + a_0 C_{in} + b_0 C_{in}$$

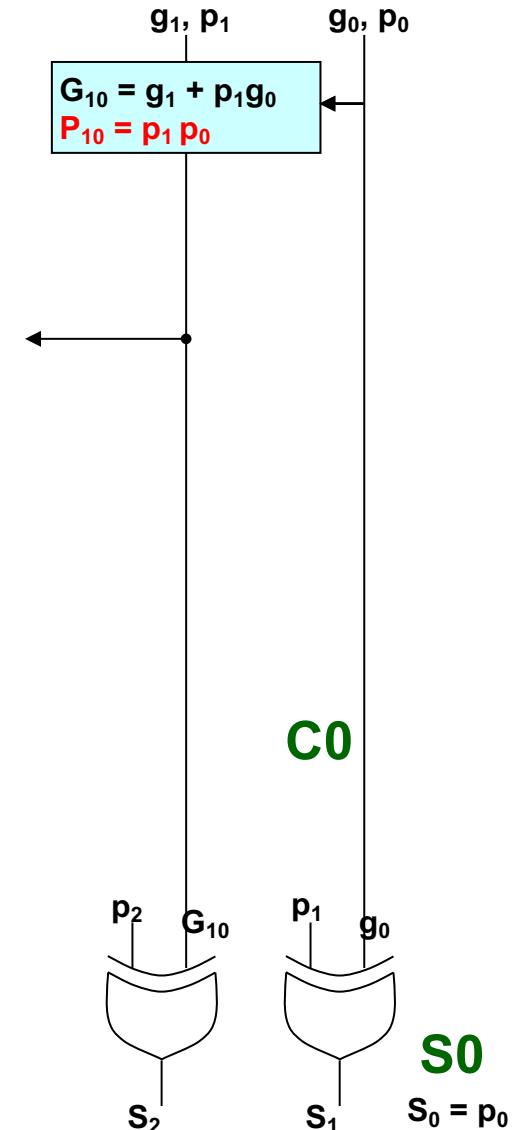
$$C_0 = a_0 b_0 = g_0$$

Soma do bit 0:

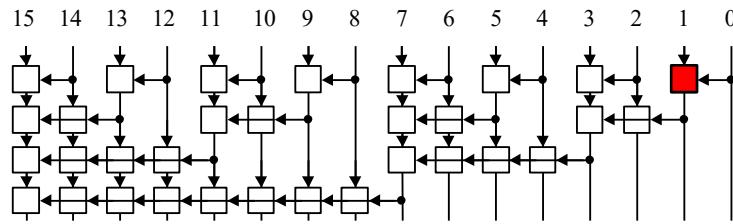
$$S_0 = a_0 \oplus b_0 \oplus C_{in}$$

Considerando $C_{in} = 0$

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



Detalhando o somador – carry do bit 1



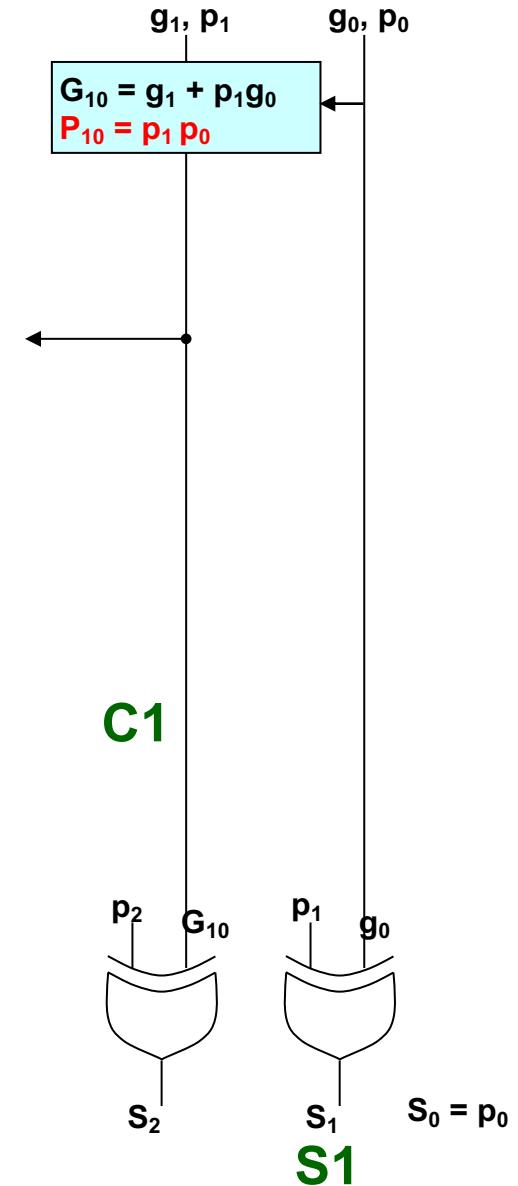
$$C_1 = g_1 + p_1 \cdot C_0$$

$$C_1 = g_1 + p_1 \cdot g_0$$

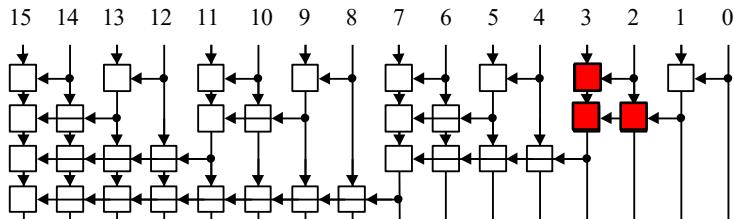
$$C_1 = G_{10} \quad \leftarrow \text{Carry do bit 1 ao 0}$$

$$S_1 = a_1 \oplus b_1 \oplus c_0 \quad \text{Todas as demais somas são calculadas desta forma}$$

$$S_1 = p_1 \oplus g_0$$



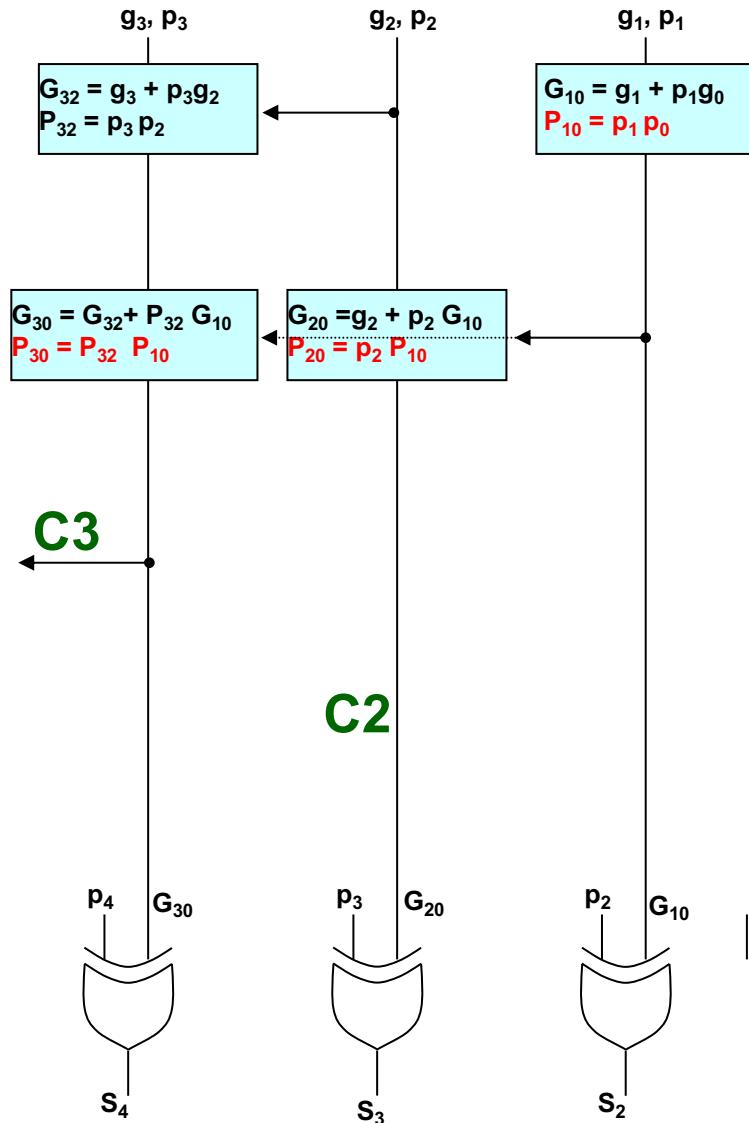
Detalhando o somador – carry dos bits 3/2



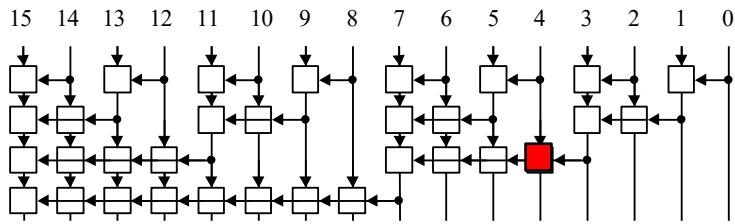
$$C_2 = g_2 + p_2 \cdot c_1 = g_2 + p_2 \cdot G10 = G20$$

$$\begin{aligned} C_3 &= g_3 + p_3 \cdot c_2 = g_3 + p_3 \cdot (g_2 + p_2 \cdot c_1) \\ &= g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot G10 \\ &= G32 + P32 \cdot G10 = G30 \end{aligned}$$

- 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

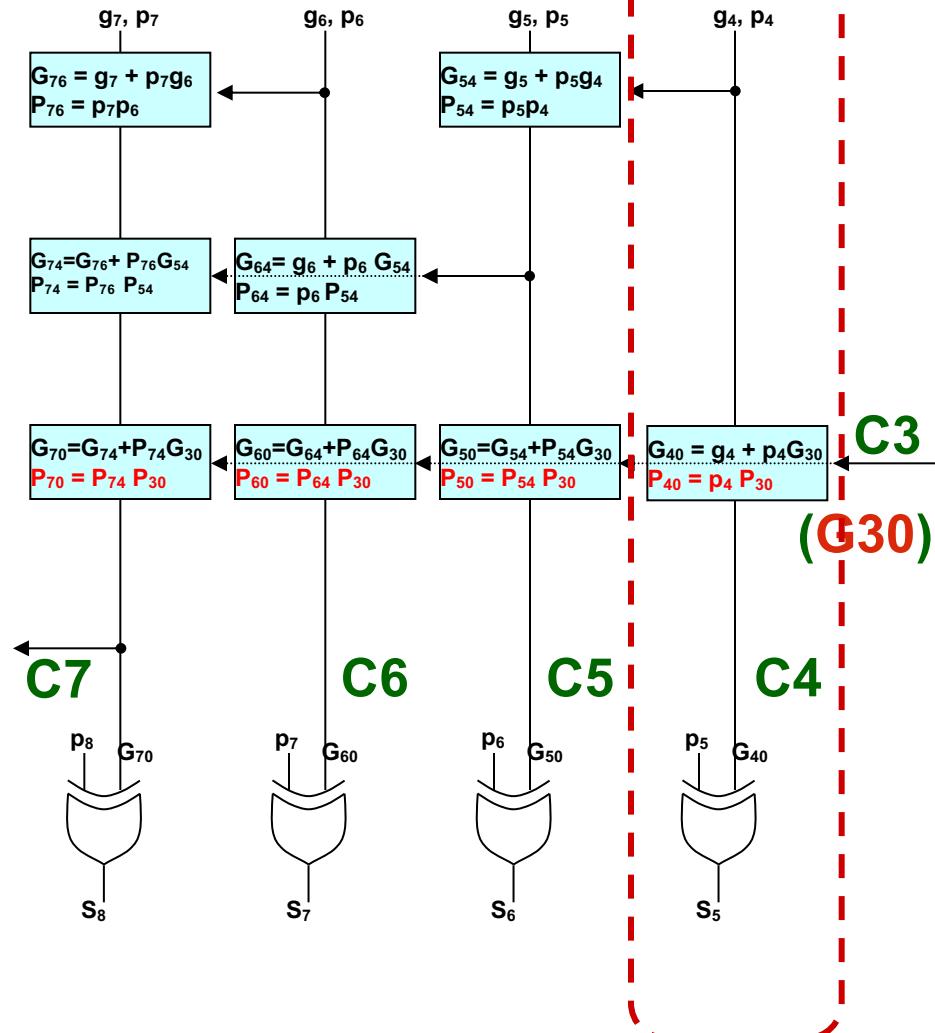


Detalhando o somador – carry dos bit 4



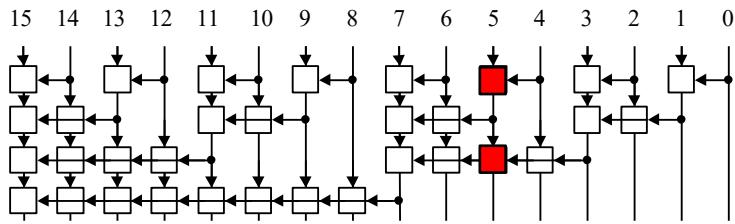
- Aumento dos níveis de propagação em até 3 camadas

$$C_4 = g_4 + p_4 \cdot C_3 = g_4 + p_4 \cdot G_{30} = G_{40}$$

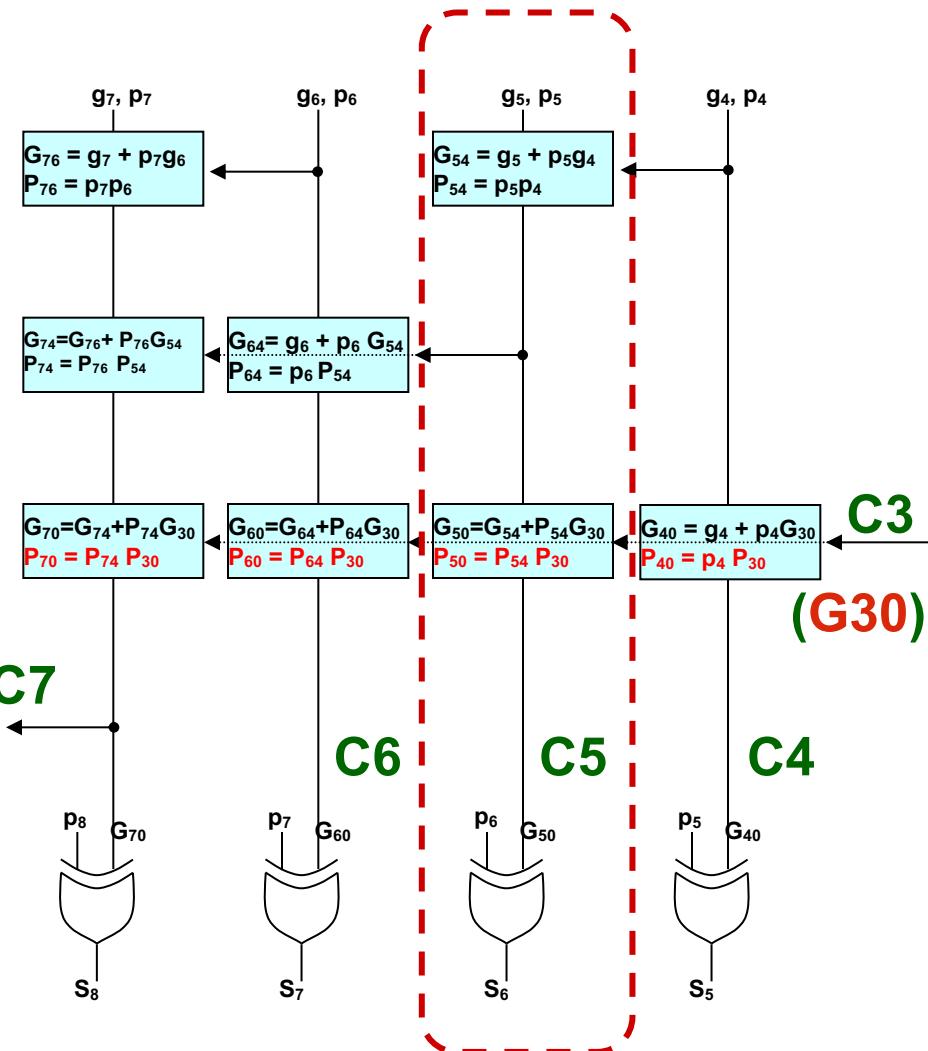


Expande a equação do carry até o carry "de entrada" do nível

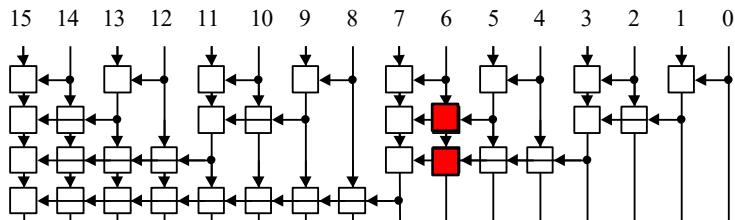
Detalhando o somador – carry dos bit 5



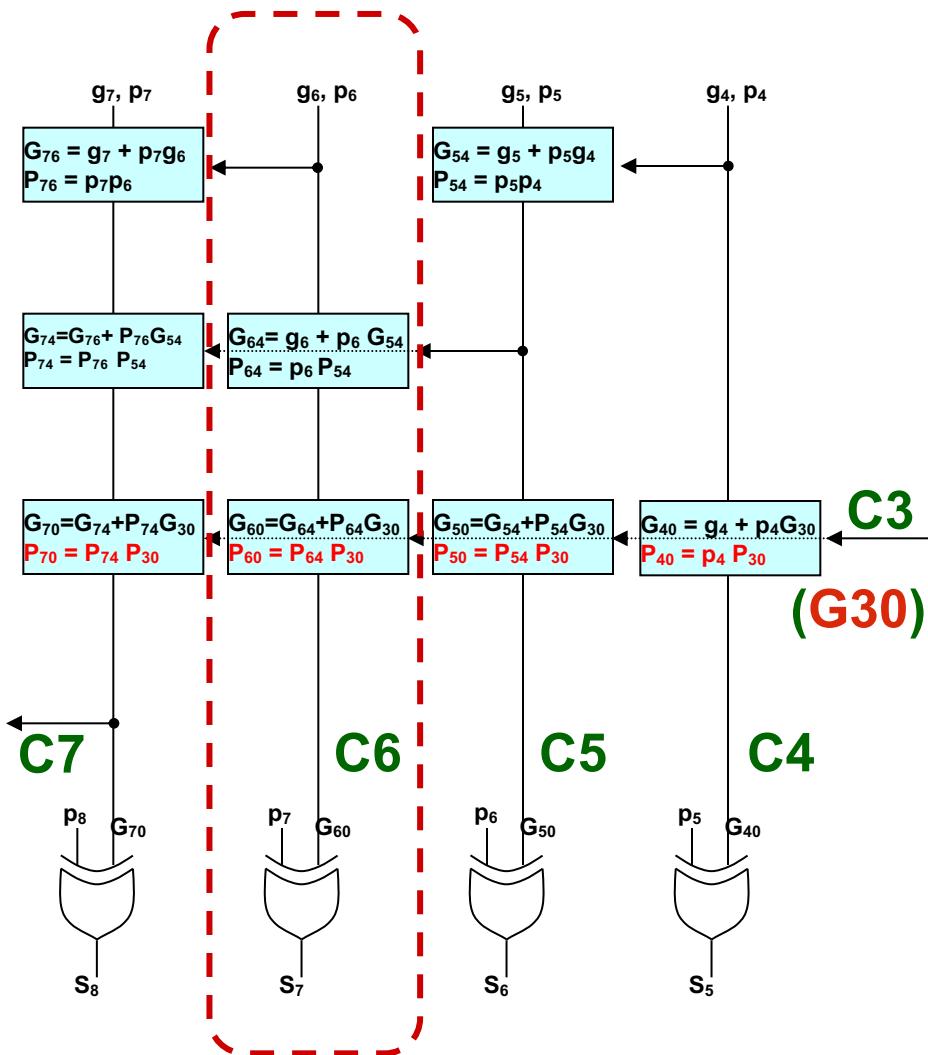
$$\begin{aligned}
 C_5 &= g_5 + p_5 \cdot c_4 = g_5 + p_5 \cdot g_4 + p_5 \cdot p_4 \cdot G_{30} \\
 &= G_{54} + P_{54} \cdot G_{30} = G_{50}
 \end{aligned}$$



Detalhando o somador – carry dos bit 6



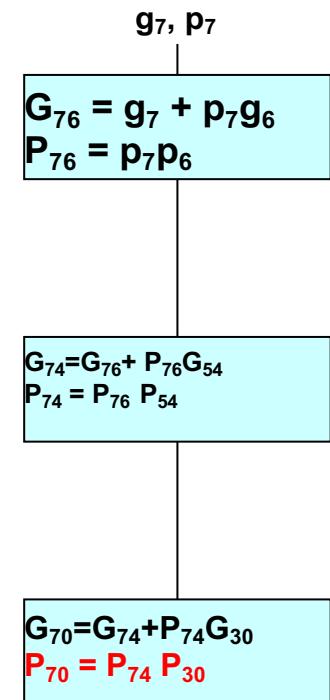
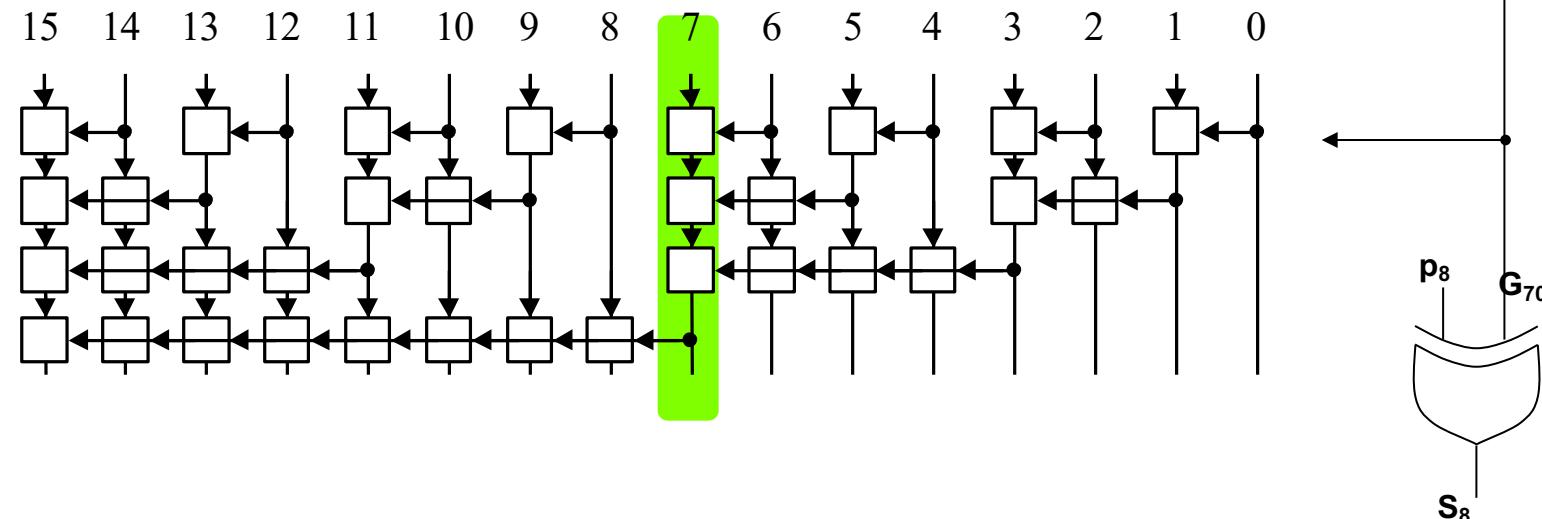
$$\begin{aligned}
 C_6 &= g_6 + p_6 \cdot c_5 \\
 &= g_6 + p_6 \cdot g_5 + p_6 \cdot p_5 \cdot g_4 + p_6 \cdot p_5 \cdot p_4 \cdot G_{30} \\
 &= g_6 + p_6 \cdot (g_5 + p_5 \cdot g_4) + p_6 \cdot p_5 \cdot p_4 \cdot G_{30} \\
 &= g_6 + p_6 \cdot (G_{54}) + p_6 \cdot P_{54} \cdot G_{30} \\
 &= G_{64} + P_{64} \cdot G_{30} = G_{60}
 \end{aligned}$$



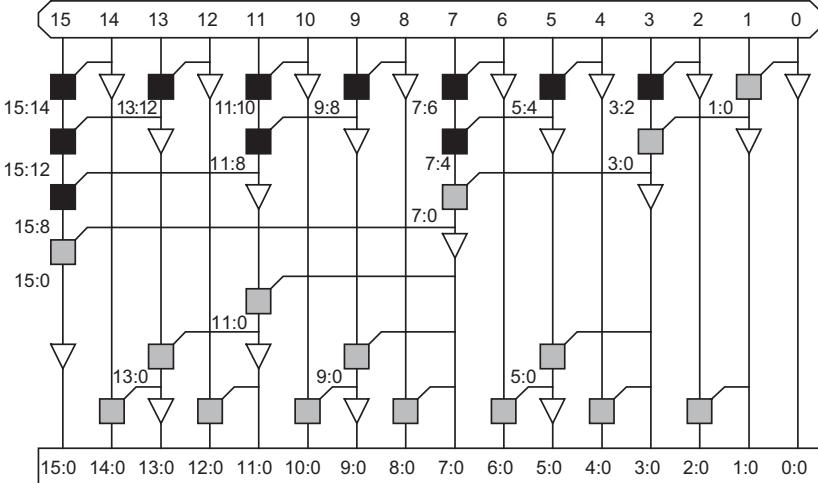
Detalhando o somador – carry dos bit 7

$$C_7 = g_7 + p_7 \cdot c_6$$

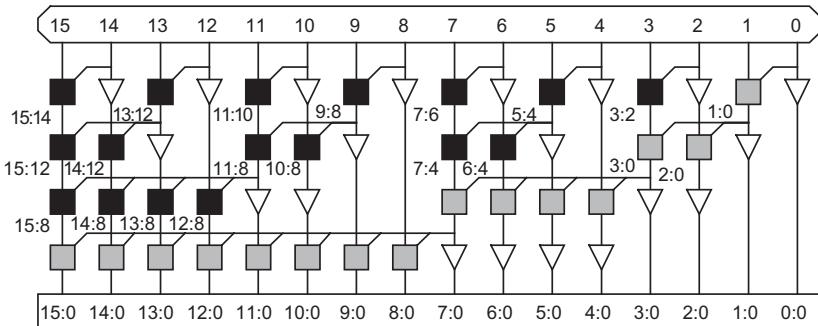
$$\begin{aligned}
 &= \underbrace{g_7 + p_7 \cdot g_6}_{G76} + \underbrace{p_7 \cdot p_6 \cdot g_5}_{P76 \cdot G54} + \underbrace{p_7 \cdot p_6 \cdot p_5 \cdot g_4}_{P76 \cdot P54 \cdot G30} + \underbrace{p_7 \cdot p_6 \cdot p_5 \cdot p_4}_{P76 \cdot P54 \cdot P30} \cdot G30 \\
 &= \underbrace{G76 + P76 \cdot (g_5 + p_5 \cdot g_4)}_{G74} + \underbrace{P76 \cdot P54 \cdot G30}_{P74 \cdot G30} \\
 &= G74 + P74 \cdot G30 = G70
 \end{aligned}$$



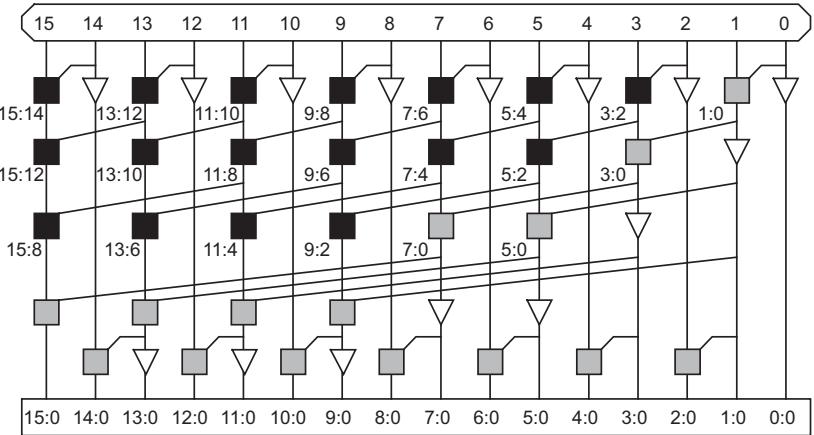
Exemplos de somadores log



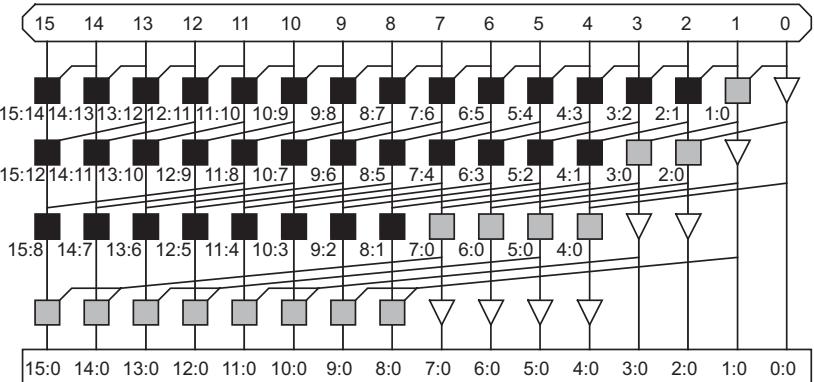
(a) Brent-Kung



(b) Sklansky

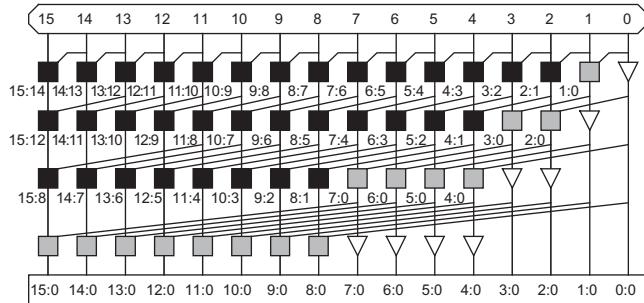


(d) Han-Carlson

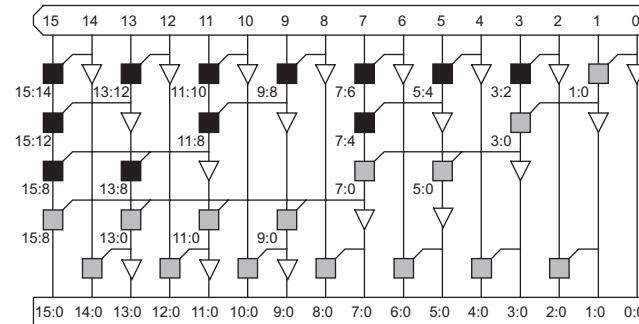


(e) Knowles [2,1,1,1]

Exemplos de somadores log



(c) Kogge-Stone **fan-out é sempre 2**



(f) Ladner-Fischer

FIGURE 11.29 Tree adder PG networks