Universidade Federal de Pelotas
Instituto de Física e Matemática
Departamento de Informática
Bacharelado em Ciência da Computação

Técnicas Digitais

Aula 15

4. Circuitos Combinacionais: Arquiteturas de somadores rápidos

Profs. José Luís Güntzel & Luciano Agostini

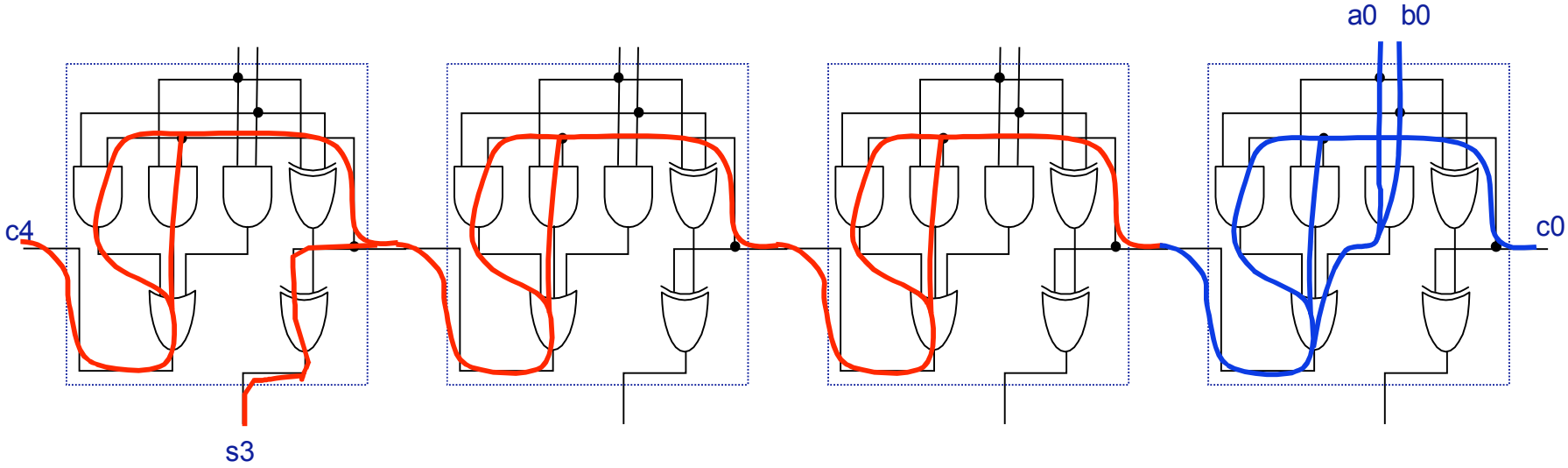
{guntzel,agostini}@ufpel.edu.br

www.ufpel.edu.br/~guntzel/TD/TD.html

4. Circuitos Combinacionais

Analisando a Propagação do *Carry*

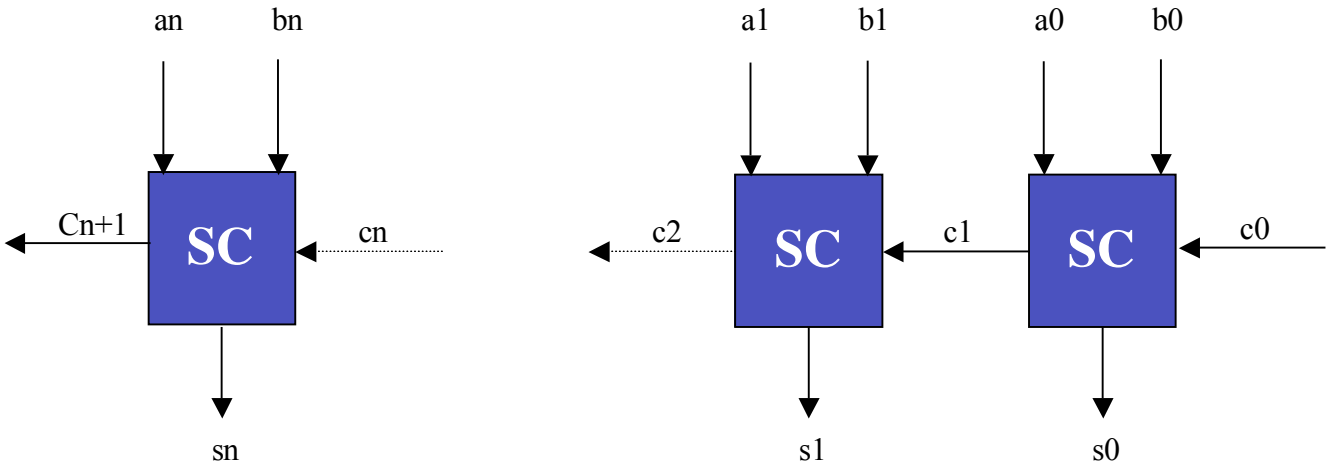
Os caminhos críticos em somadores *Ripple Carry* passam pela cadeia de propagação do *carry*



Portanto, os sinais S_3 e C_4 só estabilizam após todos os sinais antecessores estabilizarem

4. Circuitos Combinacionais

Analisando a Propagação do *Carry*



Pergunta: será que não é possível alterar a arquitetura do somador, de modo a "quebrar" ou reduzir tal interdependência?

A resposta é … **SIM!!!**

4. Circuitos Combinacionais

Analisando a Propagação do *Carry*

Há três diferentes casos na propagação do *carry* que são mutuamente exclusivos:

ai	bi	ci	ki	ci+1	si
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	0	1	1

Caso 1 (sinal ***Kill***)

Quando as entradas A e B do somador local são iguais a zero, **independentemente do *carry* de entrada**, o carry de saída será igual a **zero** e, portanto, o estágio "**matará**" a propagação do *carry* do estágio anterior.

Analisando a Propagação do *Carry*

ai	bi	ci	gi	ci+1	si
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	1	1

Caso 2 (sinal ***Generate***)

Quando as entradas A e B do somador local **são iguais a um**, independentemente do *carry* de entrada, **o *carry* de saída será igual a um** e, portanto, **haverá geração de *carry*** neste estágio.

Analisando a Propagação do *Carry*

ai	bi	ci	pi	ci+1	si
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	1	0
1	1	1	0	1	1

Caso 3 (sinal ***Propagate***)

Quando uma das entradas, A ou B, do somador local for igual a um e a outra for igual a zero, **o *carry* de saída será igual ao *carry* de entrada** e, portanto, o *carry* será **propagado**.

4. Circuitos Combinacionais

Analisando a Propagação do *Carry*

Resumindo os três casos do *carry*, temos:

Caso	a_i	b_i	c_{i+1}
Kill (k_i)	0	0	0
Propagate (p_i)	0	1	c_i
	1	0	c_i
Generate (g_i)	1	1	1

Agora, a idéia é:

- **Encontrar uma equação para cada um dos três sinais (k_i, p_i e g_i)**
- **Achar as equações de s_i e c_{i+1} em função de k_i, p_i e g_i**

Mas qual é a vantagem desta "reestruturação lógica"?

4. Circuitos Combinacionais

Analisando a Propagação do *Carry*

Caso	a_i	b_i	c_{i+1}
Kill (k_i)	**0**	**0**	**0**
Propagate (p_i)	**0**	**1**	c_i
	1	**0**	c_i
Generate (g_i)	**1**	**1**	**1**

Equações das funções k_i , p_i e g_i são:

$$\mathbf{k_i} = \overline{a_i} \cdot \overline{b_i} = \overline{a_i + b_i}$$

$$\mathbf{p_i} = a_i \oplus b_i$$

$$\mathbf{g_i} = a_i \cdot b_i$$

Analisando a Propagação do *Carry*

Caso	a_i	b_i	c_{i+1}
Kill (k_i)	0	0	0
Propagate (p_i)	0	1	c_i
	1	0	c_i
Generate (g_i)	1	1	1

Então, usando soma de produtos, o *carry out* do estágio i pode ser definido como:

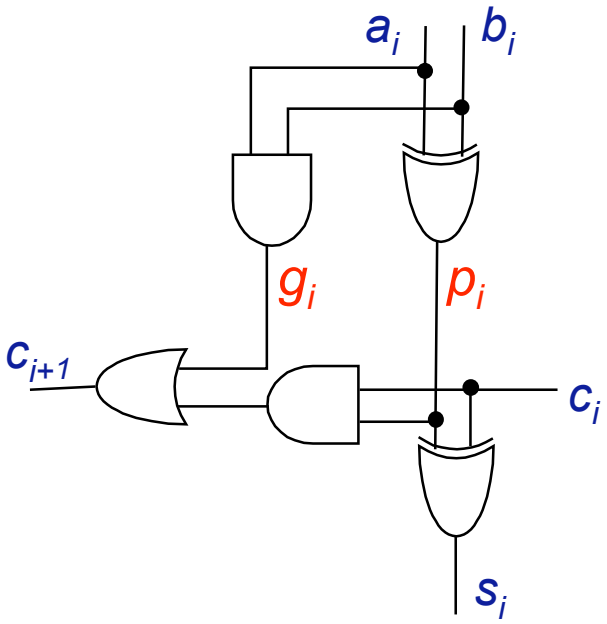
$$c_{i+1} = g_i + p_i \cdot c_i$$

E a saída "soma" do estágio i pode ser expressa (em função de p_i e c_i) por:

$$s_i = a_i \oplus b_i \oplus c_i = p_i \oplus c_i$$

Analisando a Propagação do *Carry*

As fórmulas para s_i e c_{i+1} geram o mesmo circuito já estudado, com base em dois meio-somadores.



$$\mathbf{p_i} = a_i \oplus b_i$$

$$\mathbf{g_i} = a_i \cdot b_i$$

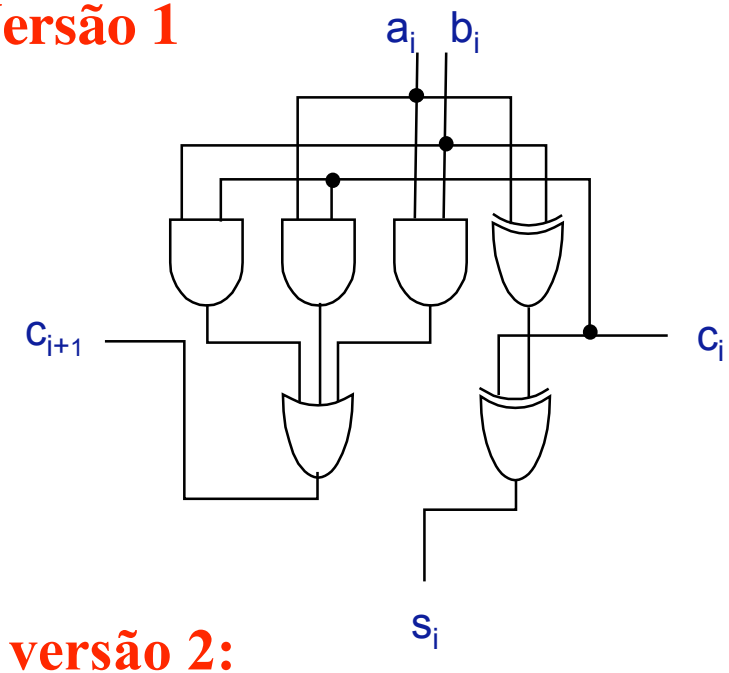
$$\mathbf{c_{i+1}} = g_i + p_i \cdot c_i$$

$$\mathbf{s_i} = p_i \oplus c_i$$

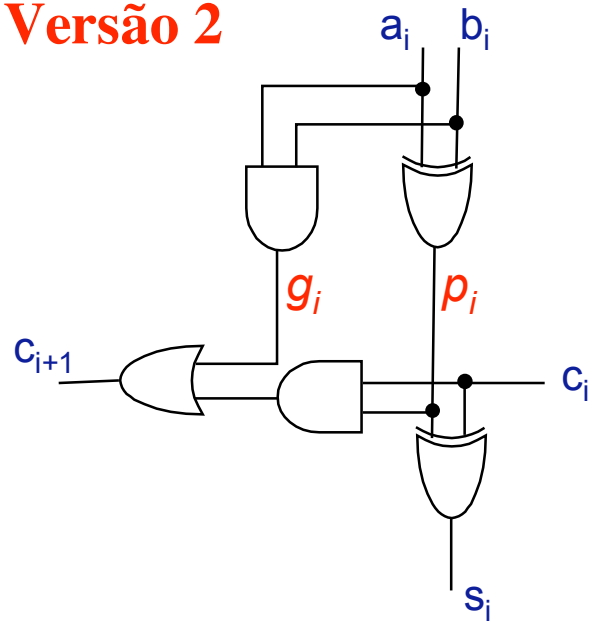
4. Circuitos Combinacionais

Comparando o Desempenho das Duas Versões

Versão 1



Versão 2



Na versão 2:

- Se $g_i=1$ não é preciso esperar pelo o valor de c_i para determinar o valor de c_{i+1}
- Se $p_i=1$, então já se sabe *a priori* que $c_{i+1} = c_i$

4. Circuitos Combinacionais

Somadores Rápidos

Para reduzir-se o atraso na propagação do *carry* as seguintes abordagens podem ser usadas:

1. **Reduzir o atraso na geração do *carry* (aplicada nos somadores Manchester);**
2. **Diminuir o atraso da cadeia de propagação do *carry* (aplicada nos somadores *Carry-Lookahead*, *Carry-Select*, *Carry-Skip*, etc.);**
3. **Mudar o sistema de representação numérica (não será tratado na disciplina).**

Estas soluções investem em desempenho, mas resultam em acréscimo de recursos (número de transistores utilizados).

4. Circuitos Combinacionais

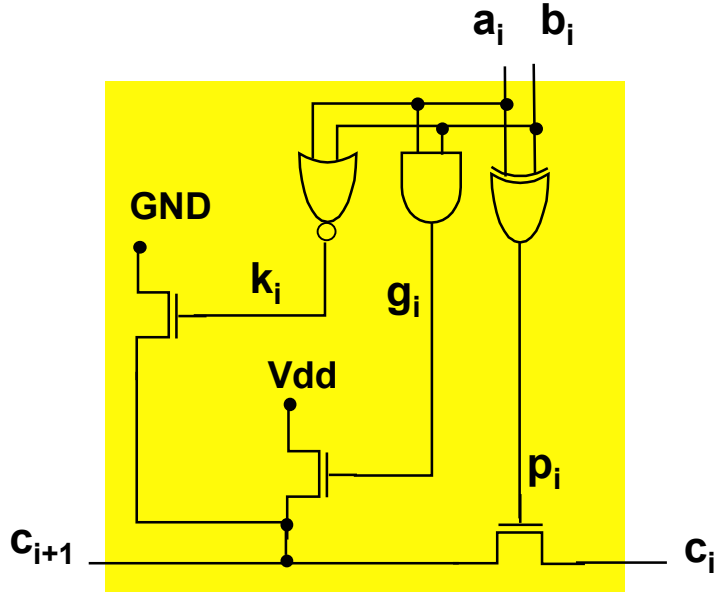
Somadores Manchester *Chain*

Princípio: usar um circuito mais rápido para propagar a cadeia de *carry*;

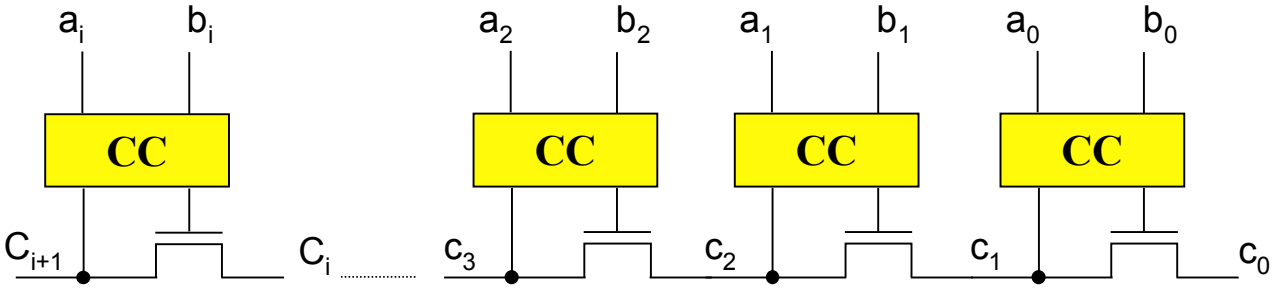
- **Este circuito mais rápido pode ser constituído de *Transmission Gates* ou transistores de passagem;**
- **Como as situações de *kill*, *propagate* e *generate* são mutuamente exclusivas, pode-se construir um conjunto de chaves com a seguinte lógica:**
  - **Se $k_i = 1$, então $c_{i+1} = 0$**
  - **Se $g_i = 1$, então $c_{i+1} = 1$**
  - **Se $p_i = 1$, então $c_{i+1} = c_i$**

Somadores Manchester *Chain*

Uma Implementação com transistores de passagem



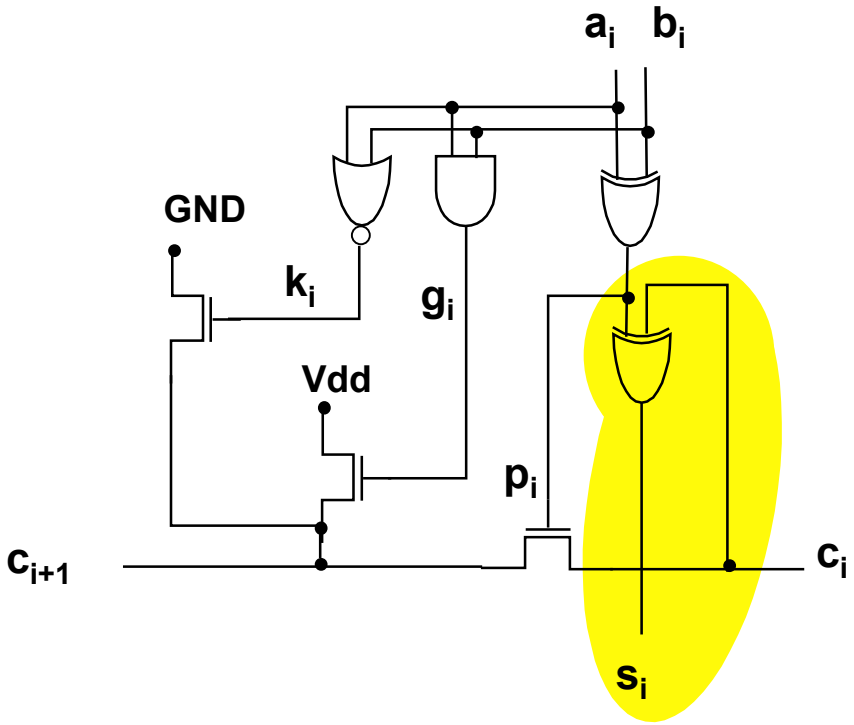
Controle da Cadeia de Propagação do *Carry* (CC)



Cadeia de propagação do *Carry*

Somadores Manchester *Chain*

E como seria o somador *Manchester Chain* completo?



Basta modificar o controle da cadeia de propagação do *carry* para que ele calcule também a saída s_i

Somadores *Carry Lookahead*

Princípio: paralelizar o cálculo dos *carries*

- **No extremo, todos os *carries* podem ser computados ao mesmo tempo.**
- **Para o somador *Carry-Lookahead*, pode-se considerar que não existe mais a exclusividade mútua entre os sinais *generate* e *propagate*.**
- **Então, a função *propagate* pode ser simplificada para um simples "OU" entre as duas entradas, pois se o nível de soma gera *carry out* ($g_i = 1$), não importa o valor de *propagate*.**

Somadores *Carry Lookahead*

Assim, para o *carry-lookahead* temos que:

$$\mathbf{p_i} = a_i + b_i$$

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

Então:

$$\mathbf{c_1} = g_0 + p_0 \cdot c_0$$

$$\mathbf{c_2} = g_1 + p_1 \cdot (g_0 + p_0 \cdot c_0) = g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_0$$

$$\mathbf{c_3} = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_0$$

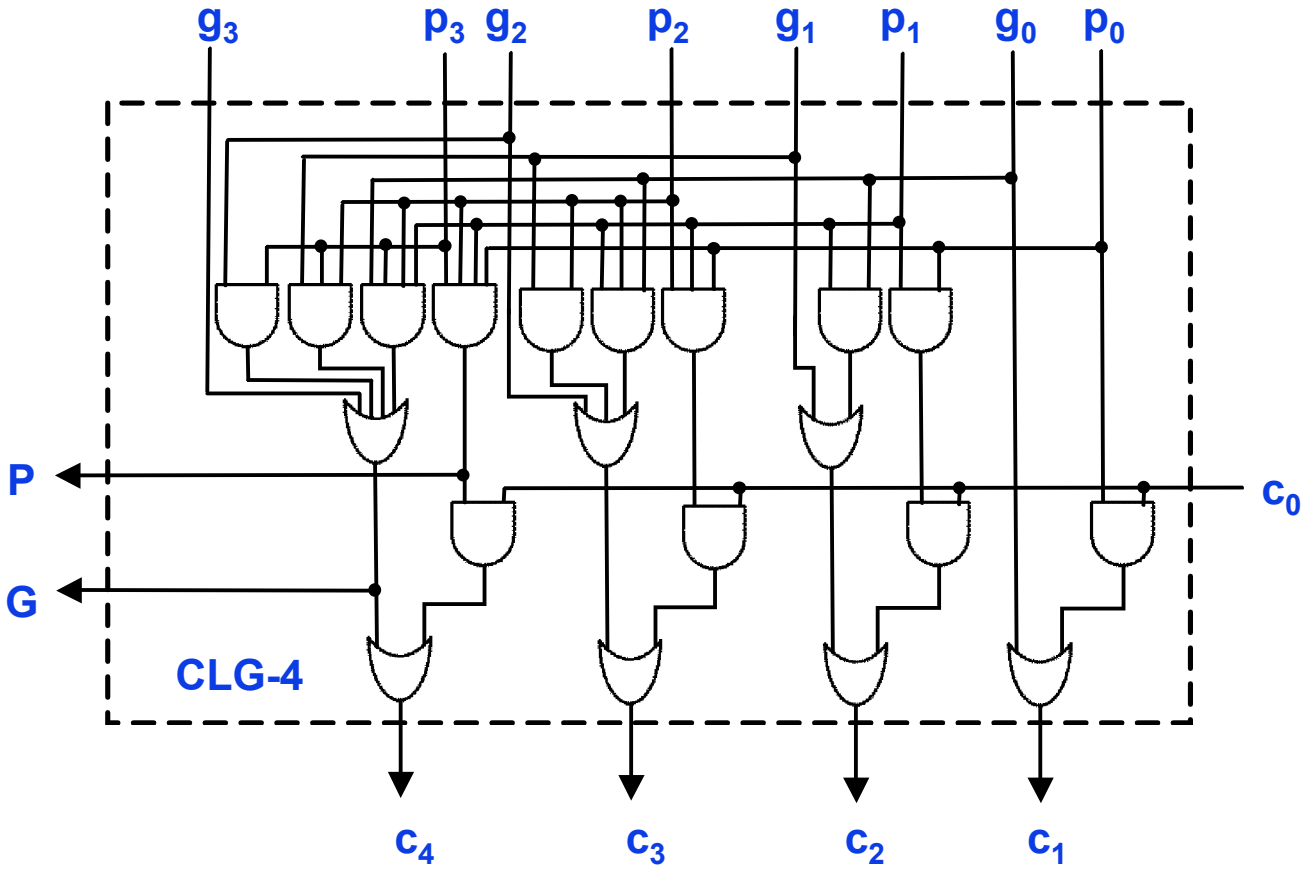
$$\mathbf{c_4} = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0$$

Generalizando...

$$\mathbf{c_{i+1}} = g_i + p_i \cdot g_{i-1} + p_i \cdot p_{i-1} \cdot g_{i-2} + \ldots + (p_i \cdot p_{i-1} \cdot p_{i-2} \cdot \ldots \cdot p_0) \cdot c_0$$

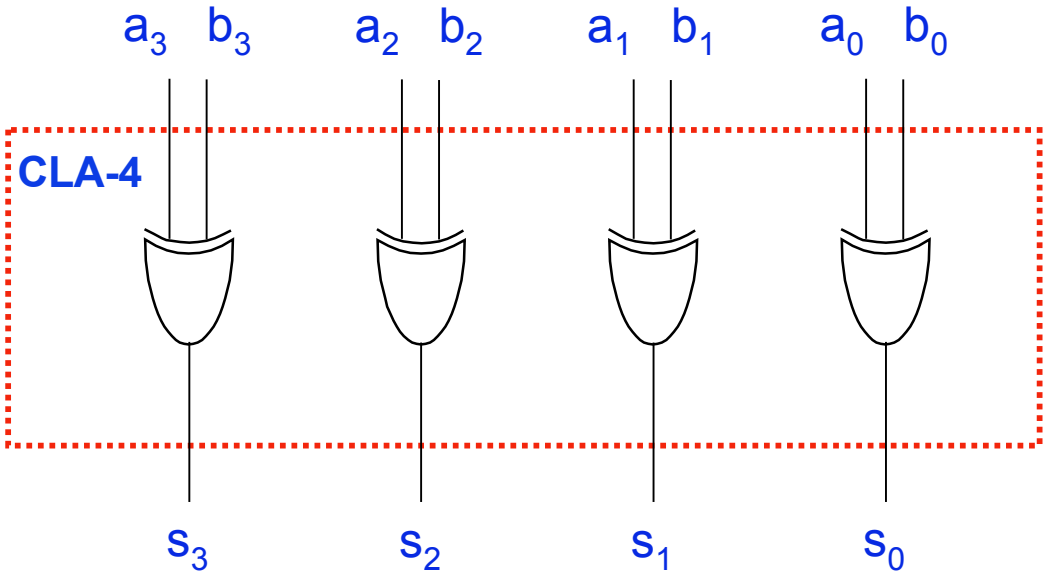
Somadores *Carry Lookahead*

Bloco *Carry Lookahead Generator* (CLG) de 4 Bits



Somadores *Carry Lookahead*

Bloco *Carry Lookahead Adder* (CLA) de 4 Bits



4. Circuitos Combinacionais

Somadores *Carry Lookahead*

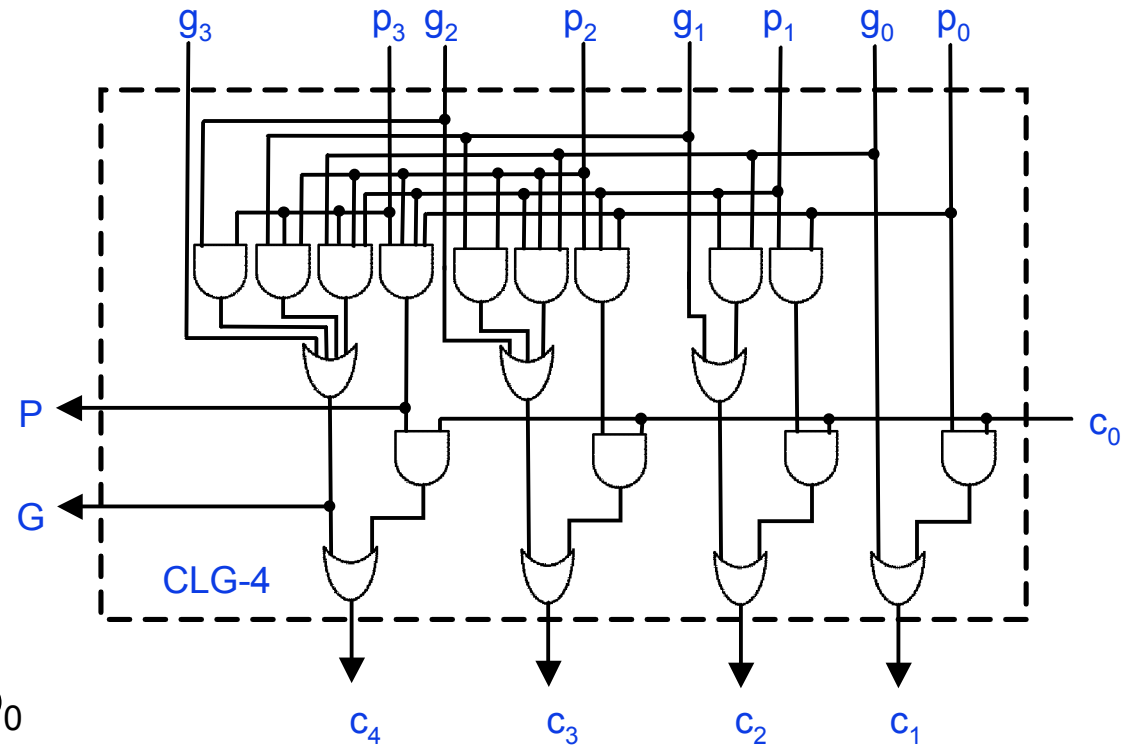
- **Problema com os CLGs: a complexidade da equação do *carry* cresce muito rapidamente!!!**
- **Para somadores com entradas usando muitos bits, a propagação do *carry* com cadeia *carry-lookahead* é mais lenta que um *ripple carry*.**
- **Tipicamente, o *carry-lookahead* não é usado para somadores com entradas maiores que 4 bits.**

Somadores *Carry Lookahead*

- **Solução: aplicar o *lookahead* em grupos de somadores com, no máximo, 4 bits.**
- **Funções auxiliares *P* e *G* são necessárias e indicam, respectivamente:**
  - **Se P = 1, então o *carry* é propagado pelo grupo.**
  - **Se G = 1, então o grupo gera *carry out*.**

Somadores *Carry Lookahead*

Carry Lookahead Generator (CLG)

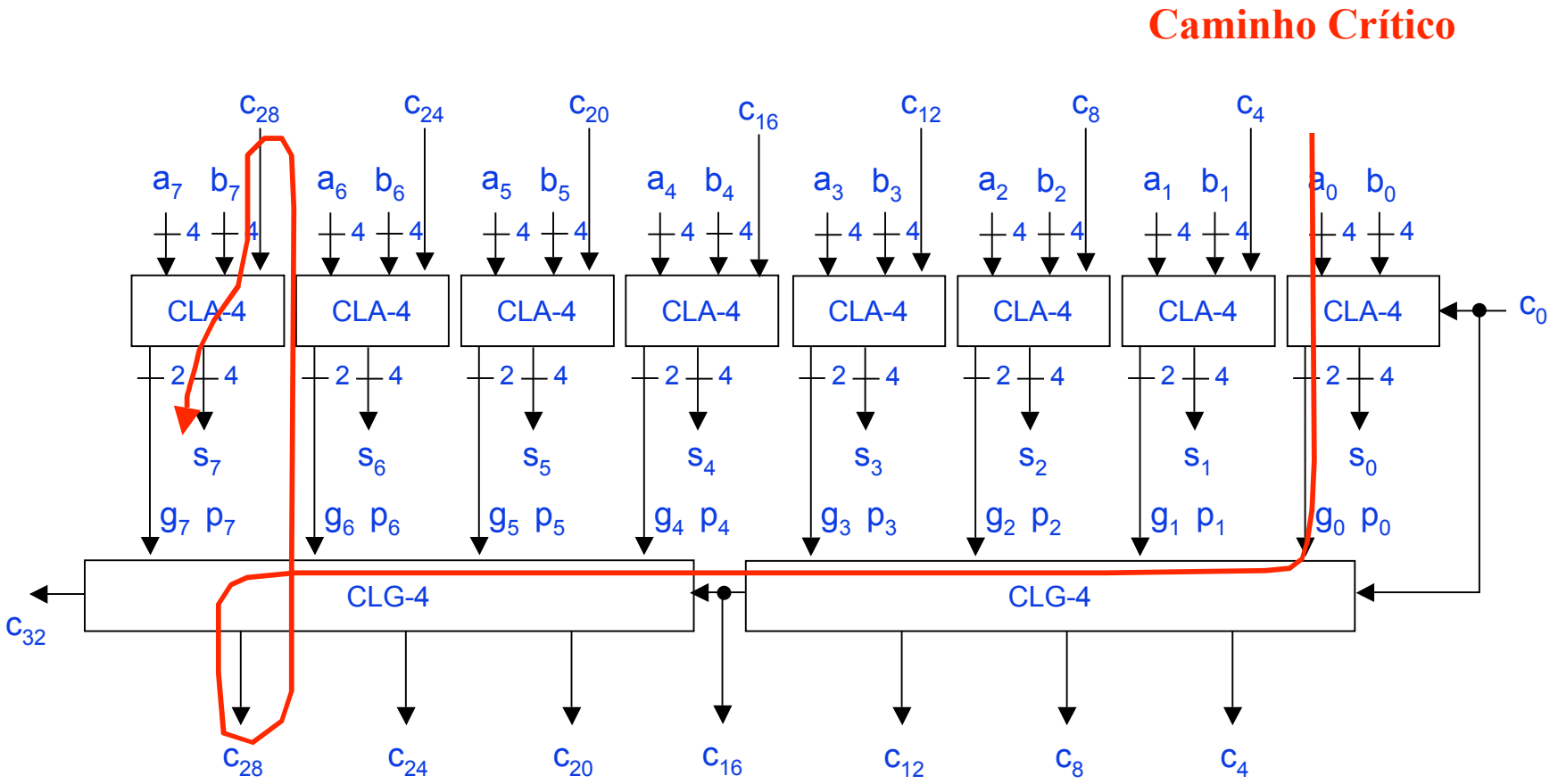


$$\mathbf{P} = p_3 \cdot p_2 \cdot p_1 \cdot p_0$$

$$\mathbf{G} = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0$$

4. Circuitos Combinacionais

Somadores *Carry Lookahead*



4. Circuitos Combinacionais

Somadores *Carry Select*

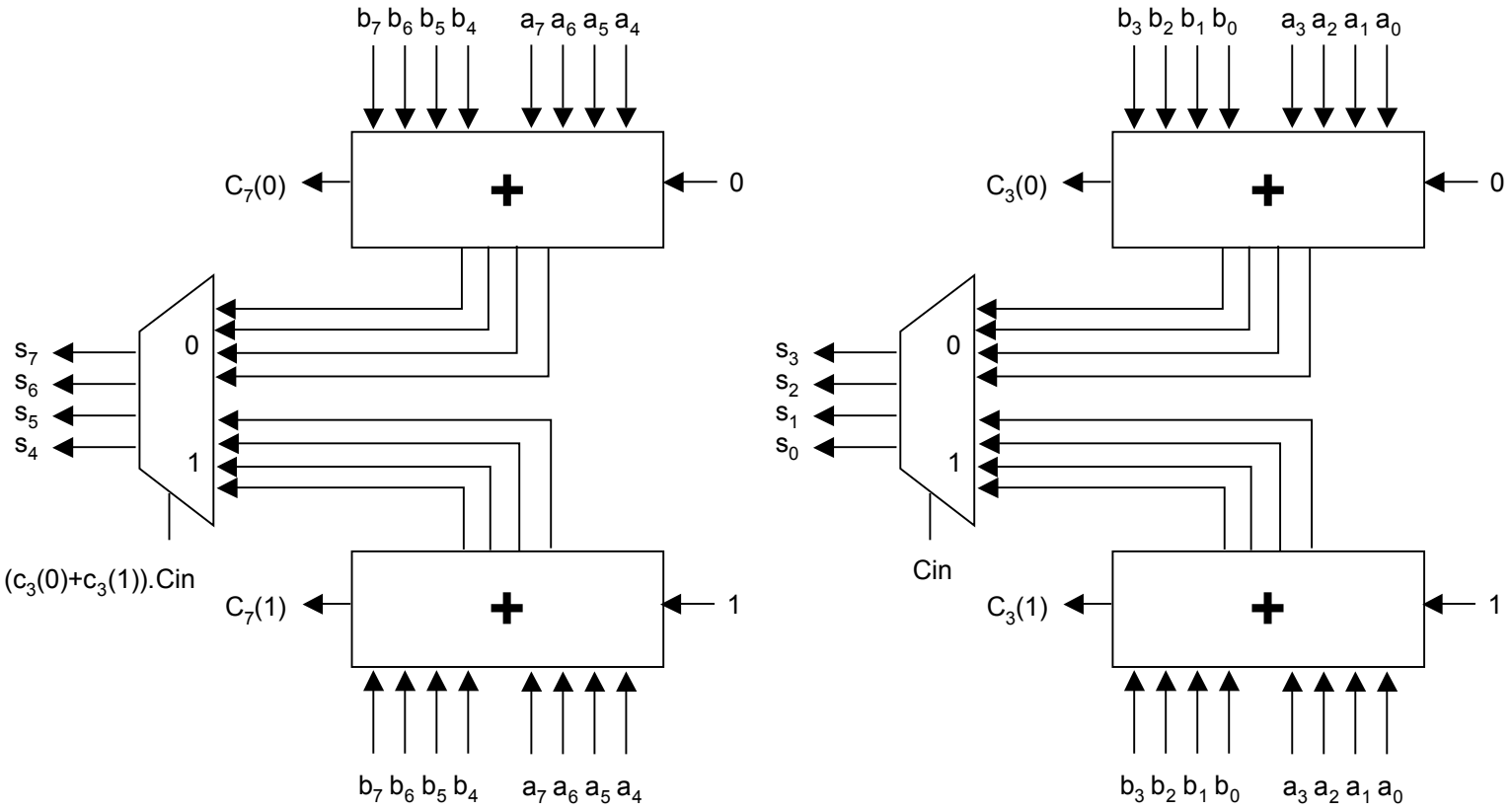
Princípio: dividir a adição em seções de 4 ou 8 bits e em cada seção, realizar a adição simultaneamente para os dois casos possíveis (*carry in*=0 e *carry in*=1)

- **Em cada seção de adição são usados dois somadores (geralmente, *ripple carry*) idênticos e um multiplexador**
- **O multiplexador seleciona um dos dois resultados, utilizando como controle o *carry out* da seção anterior**

4. Circuitos Combinacionais

Somadores *Carry Select*

Somador *Carry Select* de 8 Bits (com seções de 4 Bits)



4. Circuitos Combinacionais

Resultados Práticos

Comparação dos resultados de síntese obtidos com o uso de diversos tipos de somadores em um circuito complexo (DCT 2-D)

DCT-2D	RCA	CLA	CLAH	CSA	Ajuste fino
Número de LCs	3664	3777	3828	3979	3856
Freqüência (MHz)	45,57	49,24	54,47	57,81	57,19
Potência (mW)	490	800	726	1312	561
Acréscimo de desempenho	-	8,05%	19,53%	26,86%	25,5%
Acréscimo no uso de recursos	-	3,08%	4,48%	8,6%	5,24%
Acréscimo de potência	-	63,27%	48,16%	167,8%	14,49%