PCS 3115 – Sistemas Digitais I

Circuitos Combinatórios:
Somadores e Subtratores

Suporte para EAD

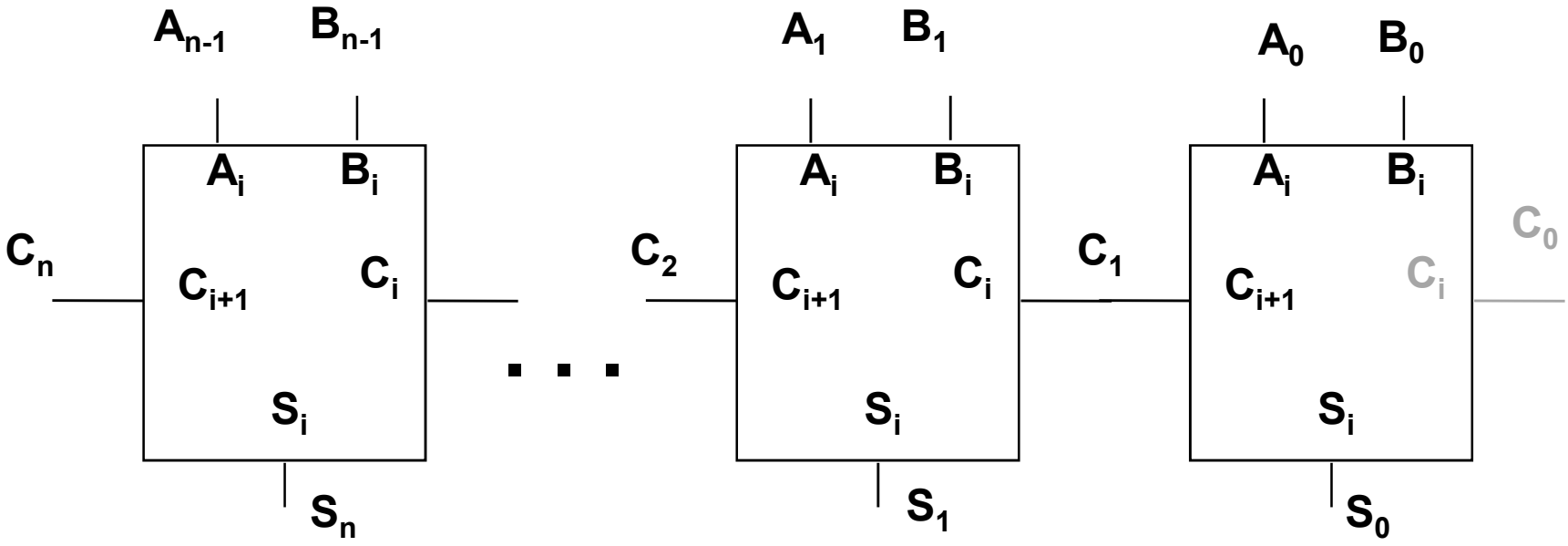
Parte IV:
Somadores – Cálculo Antecipado do *Carry* - *Carry Look-ahead*.

Aula: 14 – Data: 27/04 (S)

Prof. Dr. Marco Túlio Carvalho de Andrade

versão: 1.0 (Abril/2020)

SLIDE 12-Somador binário de n bits

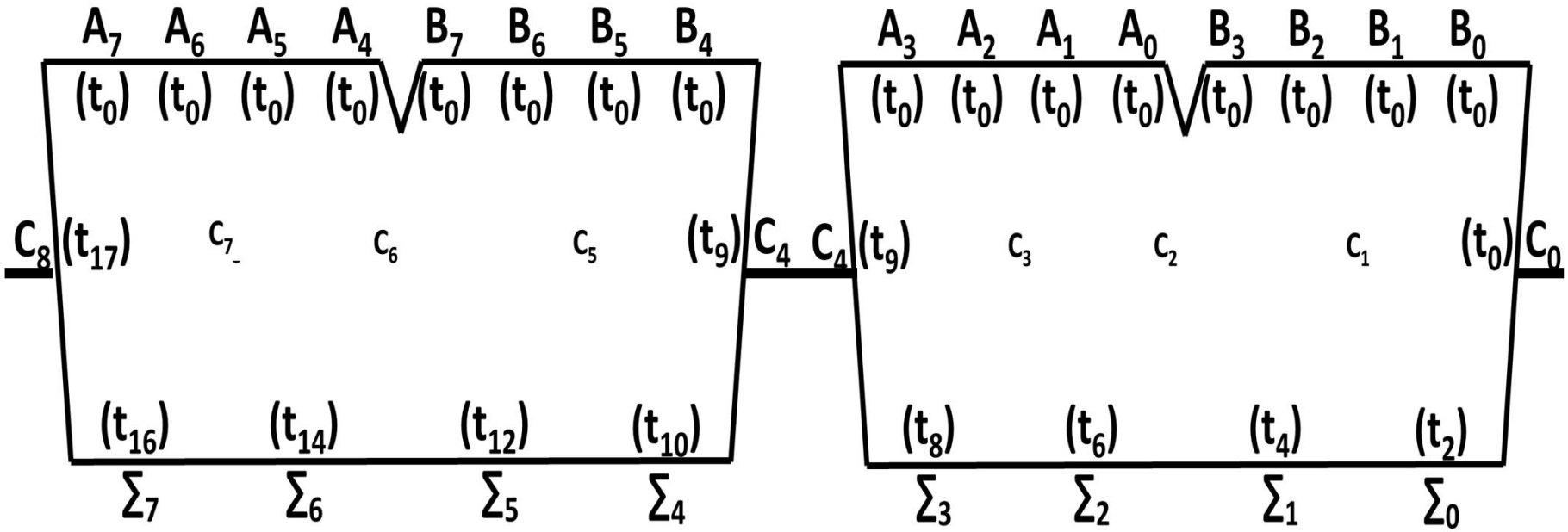


Somador binário de *n* bits com propagação de "vai-um"

➔ **Problema?**

Latência de propagação de "vai-um" do estágio menos até o mais significativo é proporcional a n…

ParteIII-SLIDE19-*Ripple Adder* – Módulos [1]+[2]– 8 bits –Cálculo de t_p



$$t_{p:\Sigma 7} = (2.8).t_p \qquad t_{p:\Sigma 3} = (2.4).t_p$$

$$t_{p:C8} = (2.8+1).t_p \qquad t_{p:C4} = (2.4+1).t_p$$

$$t_{p:Cn} = (2.n+1).t_p \qquad t_{p:\Sigma n-1} = (2.n).t_p$$

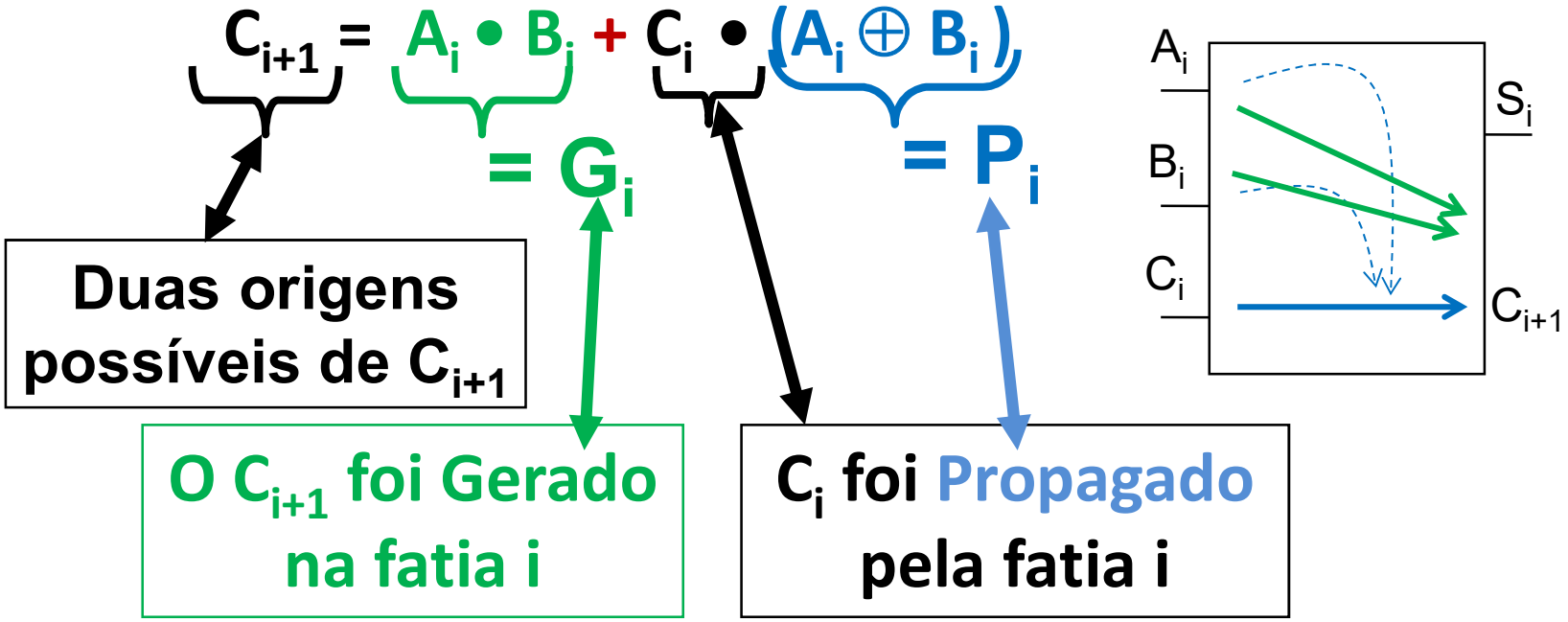
SLIDE 13-Somador binário de n bits

- **Pergunta**: Como solucionar esse problema de latência?
- **Resposta**: montando circuito que calcula saída a partir das entradas em paralelo ou diretamente.
  - Ex.: Soma de produtos levaria a um atraso de apenas dois níveis (AND e OR)…
- **Resultado**: "somadores com antecipação de vai-um", ou *carry look-ahead*

- Vamos tentar construir esse circuito...

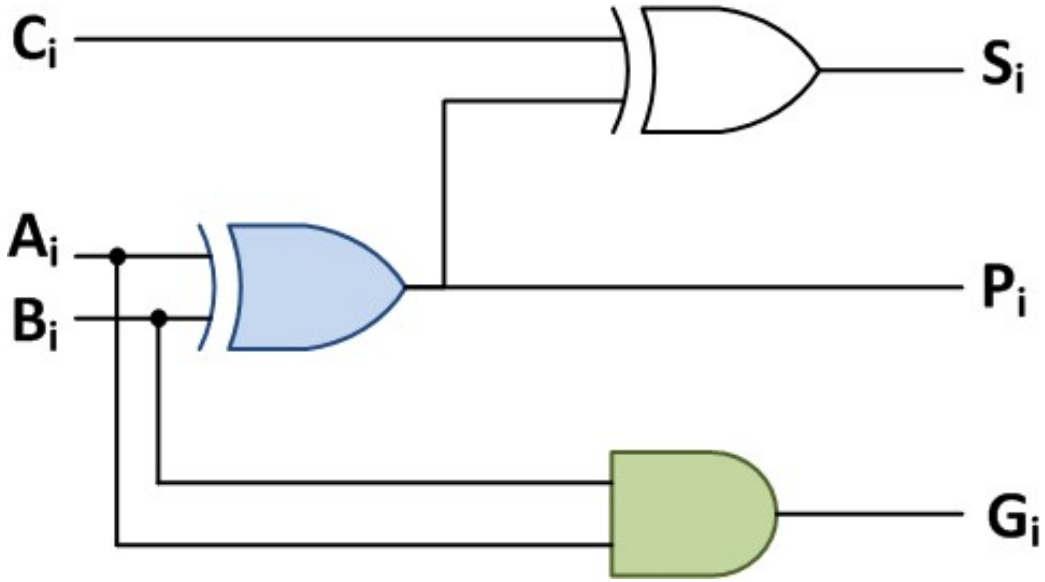
SLIDE 14-Somador binário de n bits

- *Carry* antecipado (*carry look-ahead*) – Notar que **todos** os P_i e G_i são calculados, obtidos, com tempo de latência igual a 1 (t_p=1).



Somador binário de n bits

- *Carry look-ahead*: **Unidade somadora** de 1 bit

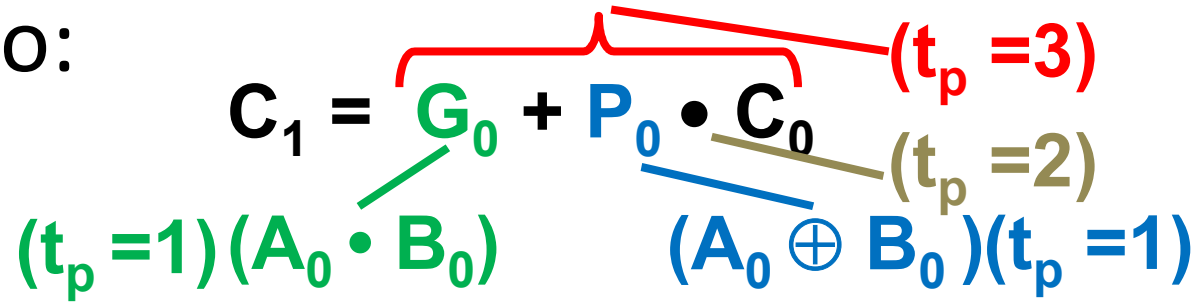


$$G_i = A_i \bullet B_i \quad ; \quad P_i = (A_i \oplus B_i)$$

- **Observação importante – Qualquer que seja G_i ou P_i, é obtido com tempo $t_p = 1$!**

SLIDE 15-Somador binário de n bits

- Antecipação de carry (*carry look-ahead*) – Generalizando:



- Obtém-se C_1 com $t_p = 3$ e $\sum_0$ com $t_p = 2$.
- Prosseguindo com a generalização:

$$C_2 = G_1 + P_1 \bullet C_1 = G_1 + P_1 \bullet (G_0 + P_0 \bullet C_0)$$
$$= G_1 + P_1 \bullet G_0 + P_1 \bullet P_0 \bullet C_0$$

$(A_1 \bullet B_1)$ $(t_p = 1)$ — geração — $(t_p = 1)(A_1 \oplus B_1)$ — propagação — $(t_p = 1)$ $(A_0 \oplus B_0)$

Antecipação de carry – C*arry look-ahead*

- Nesta soma de produtos (expressão algébrica de C_2) constata-se que:
  - Cada termo com uma literal é obtido em t_p=1;
  - Cada termo produto é obtido em t_p=2;
  - A soma de produtos C_2 é obtida em t_p=3;
  - Tem-se que $\sum_1$ é obtido com t_p = 4.

$$C_2 = G_1 + P_1 \bullet C_1 = G_1 + P_1 \bullet (G_0 + P_0 \bullet C_0)$$

$$= G_1 + P_1 \bullet G_0 + P_1 \bullet P_0 \bullet C_0$$

$(A_1 \bullet B_1)$ $(t_p = 1)$

geração

$(t_p = 1)(A_1 \oplus B_1)$

propagação $(t_p = 1)$ $(A_0 \oplus B_0)$

Antecipação de carry – C*arry look-ahead*

- Prosseguindo com a generalização – Na expressão algébrica de C_3) constata-se que:
  - Cada termo com uma literal é obtido em t_p=1;
  - Cada termo produto é obtido em t_p=2;
  - A soma de produtos C_3 é obtida em t_p=3;
  - Conclui-se que $\sum_2$ é obtido com t_p = 4.

$$C_3 = G_2 + P_2 \bullet C_2 = G_2 + P_2 \bullet (G_1 + P_1 \bullet G_0 + P_1 \bullet P_0 \bullet C_0)$$

$$C_3 = \underbrace{G_2 + P_2 \bullet G_1 + P_2 \bullet P_1 \bullet P_0 \bullet G_0}_{\text{Geração nas fatias 0, 1 ou 2}} + \underbrace{P_2 \bullet P_1 \bullet P_0}_{\text{Propagação nas fatias 0, 1 e 2}} \bullet C_0$$

Antecipação de carry – C*arry look-ahead*

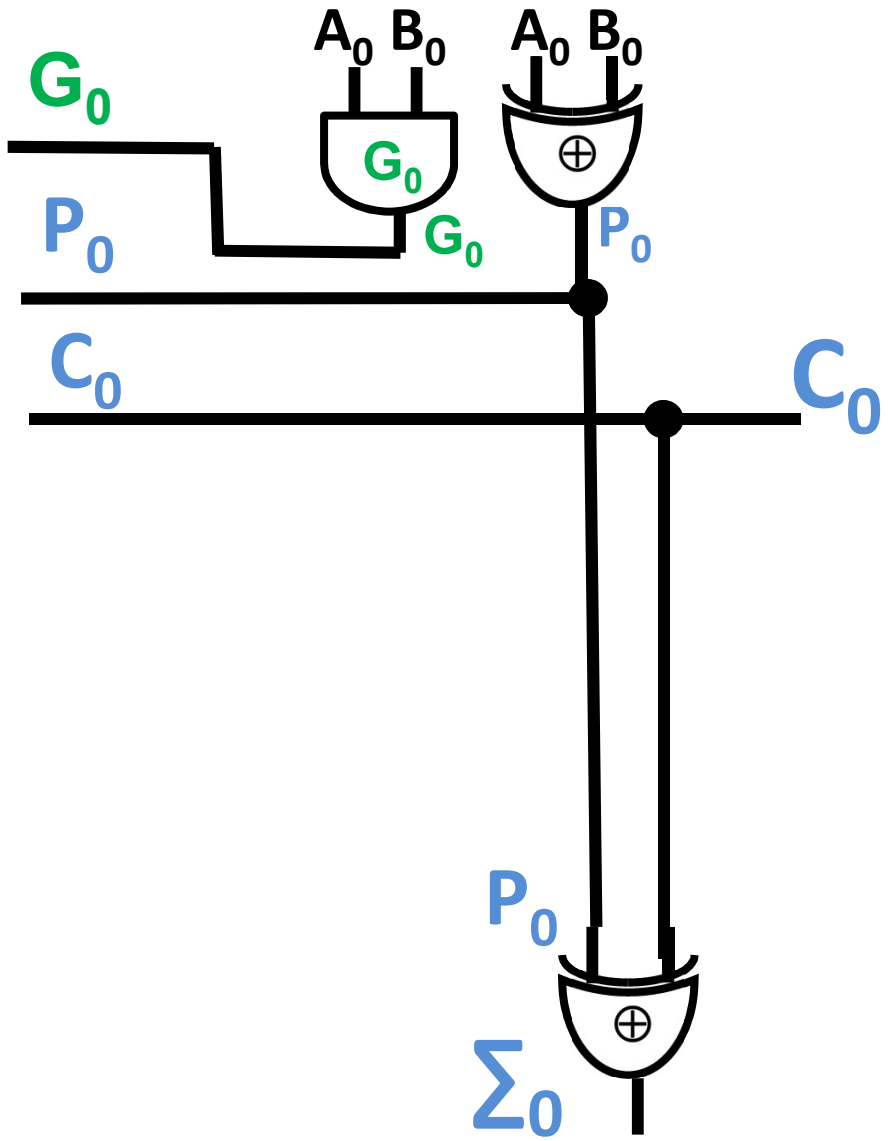
- Seguindo – C_4 é obtido em $t_p = 3$ e $\sum_3$ é obtido em $t_p = 4$ (o *Carry* é obtido "um" t_p antes da Soma).
- C_4 – Pode ser útil se este bloco for conectado a outro bloco usando *ripple carry*.

$$C_4 = G_3 + P_3 \bullet C_3 = G_3 + P_3 \bullet (G_2 + P_2 \bullet G_1 + P_2 \bullet P_1 \bullet G_0 + P_2 \bullet P_1 \bullet P_0 \bullet C_0)$$

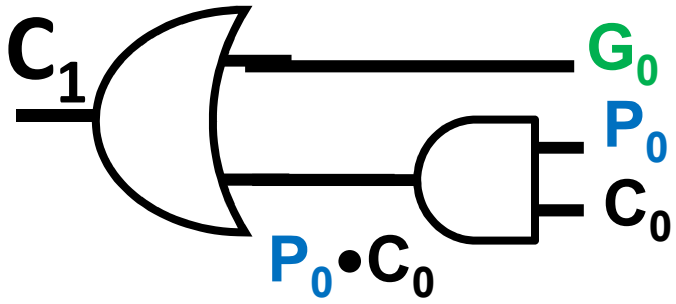
$$C_4 = \underbrace{G_3 + P_3 \bullet G_2 + P_3 \bullet P_2 \bullet G_1 + P_3 \bullet P_2 \bullet P_1 \bullet G_0}_{\text{Geração nas fatias 3, 2, 1 ou 0}} +$$

$$+ \underbrace{P_3 \bullet P_2 \bullet P_1 \bullet P_0}_{\text{Propagação nas fatias 3, 2, 1 e 0}} \bullet C_0$$

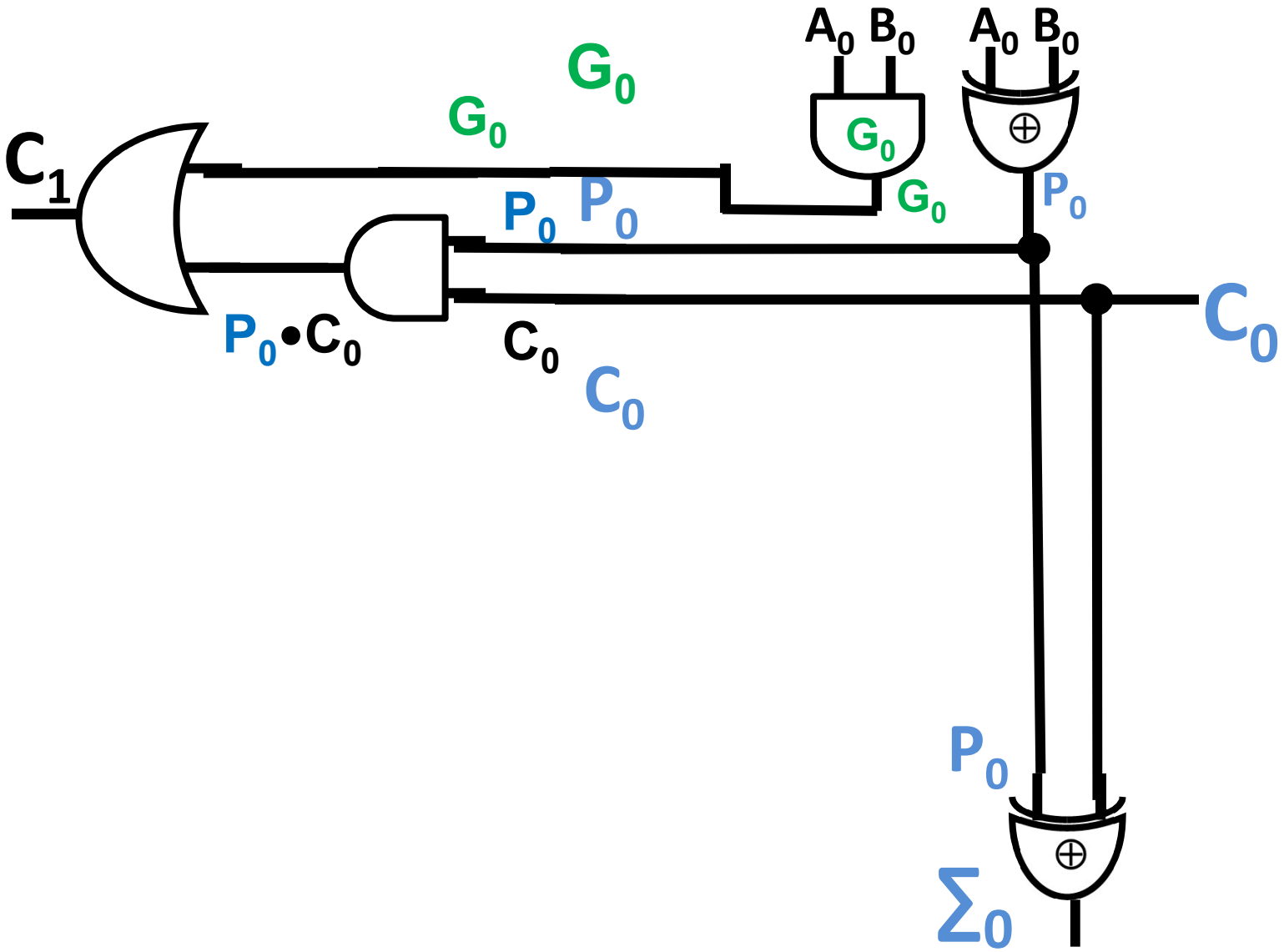
C*arry look-ahead* – Projeto das Fatias – Fatia 0



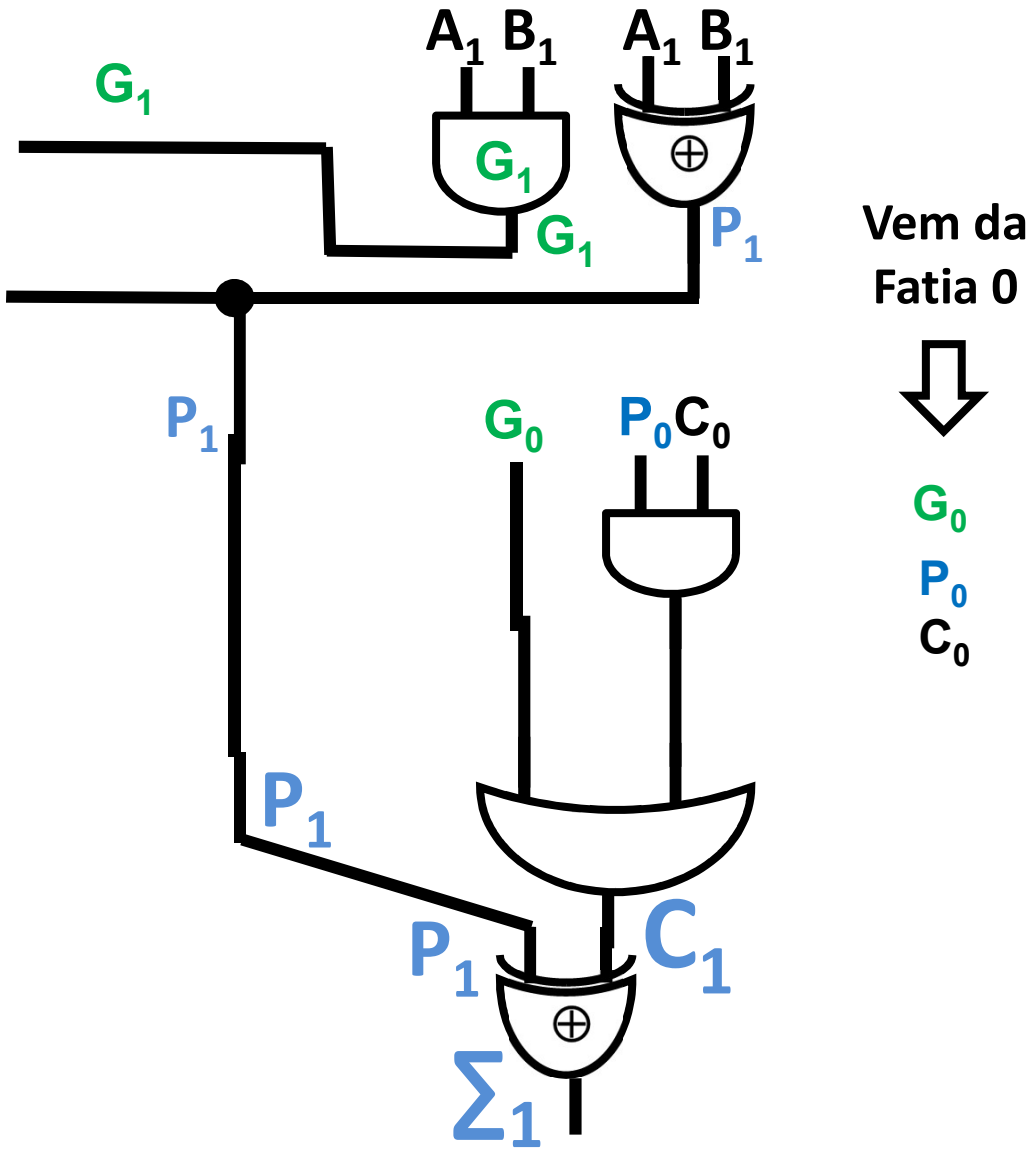
C*arry look-ahead* – Projeto das Fatias – *Carry*1



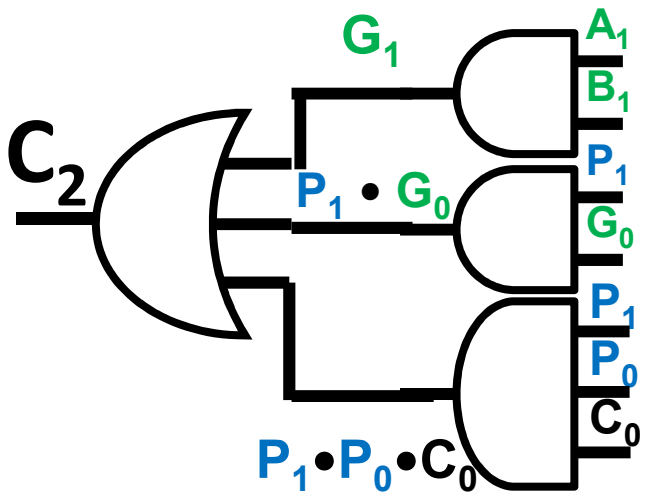
C*arry look-ahead* – Projeto das Fatias – Fatia 0 + C_1



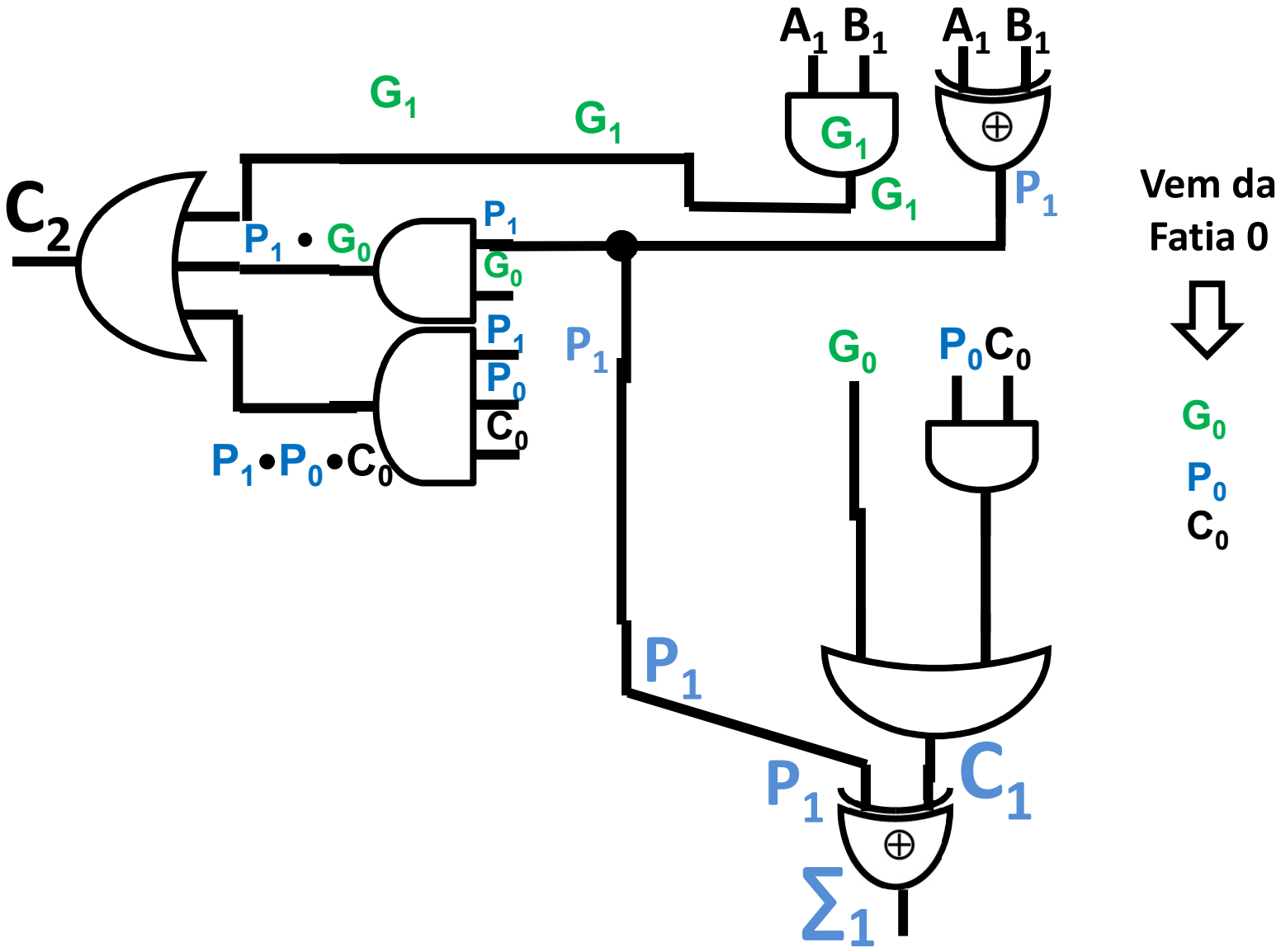
Carry look-ahead – Projeto das Fatias – Fatia 1



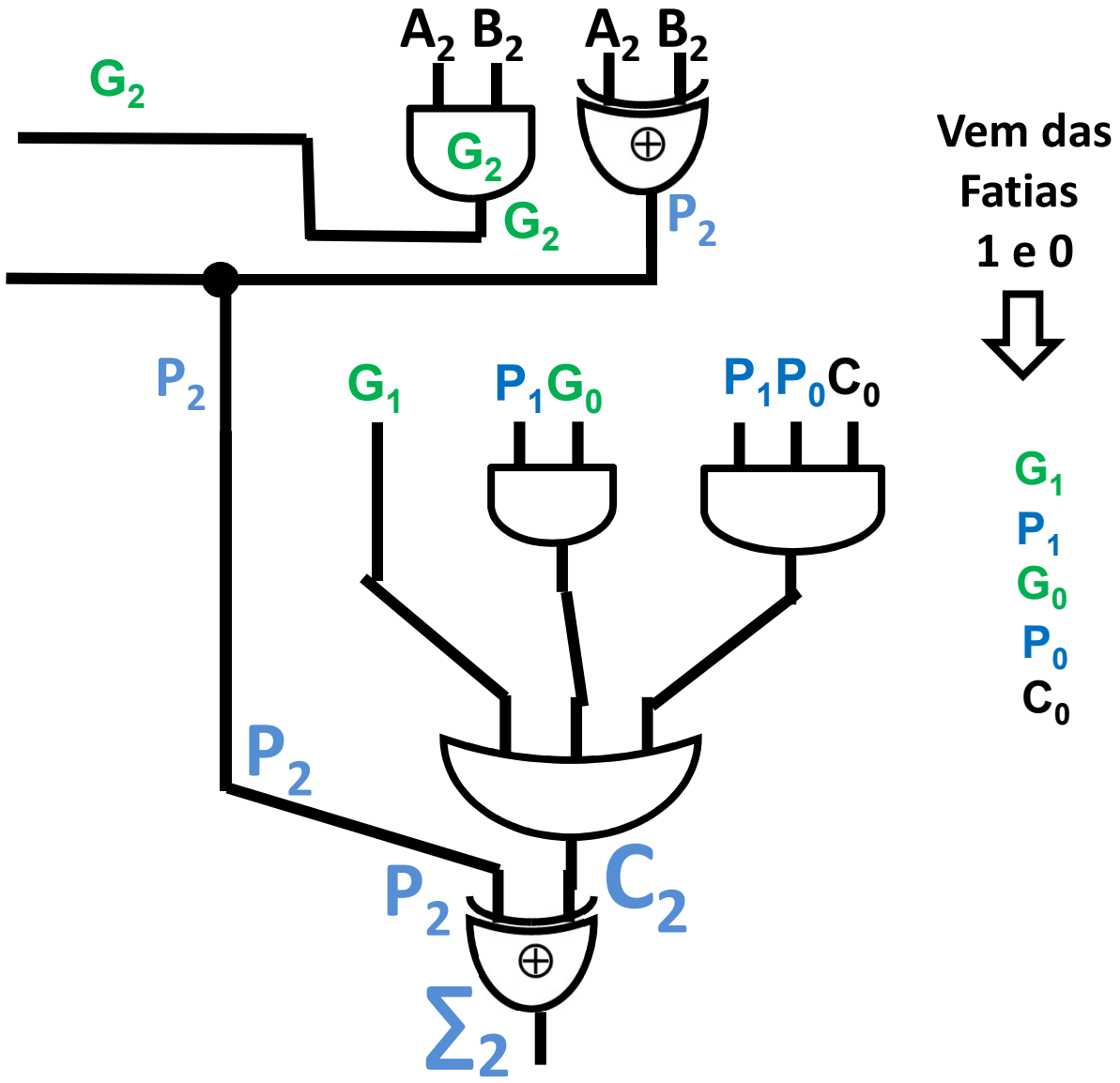
C*arry look-ahead* – Projeto das Fatias – *Carry*2



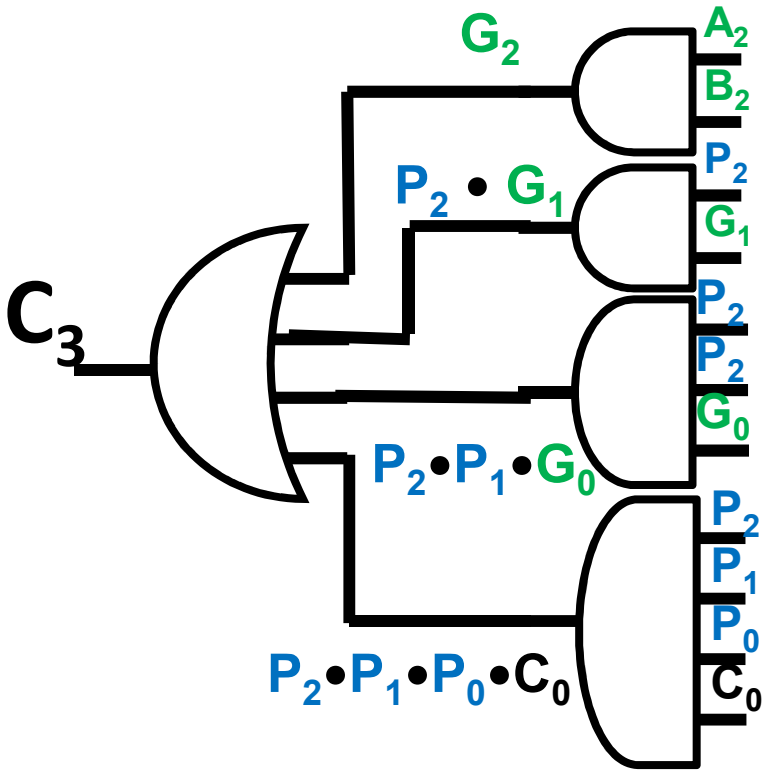
Carry look-ahead – Projeto das Fatias – Fatia 1 + C_2



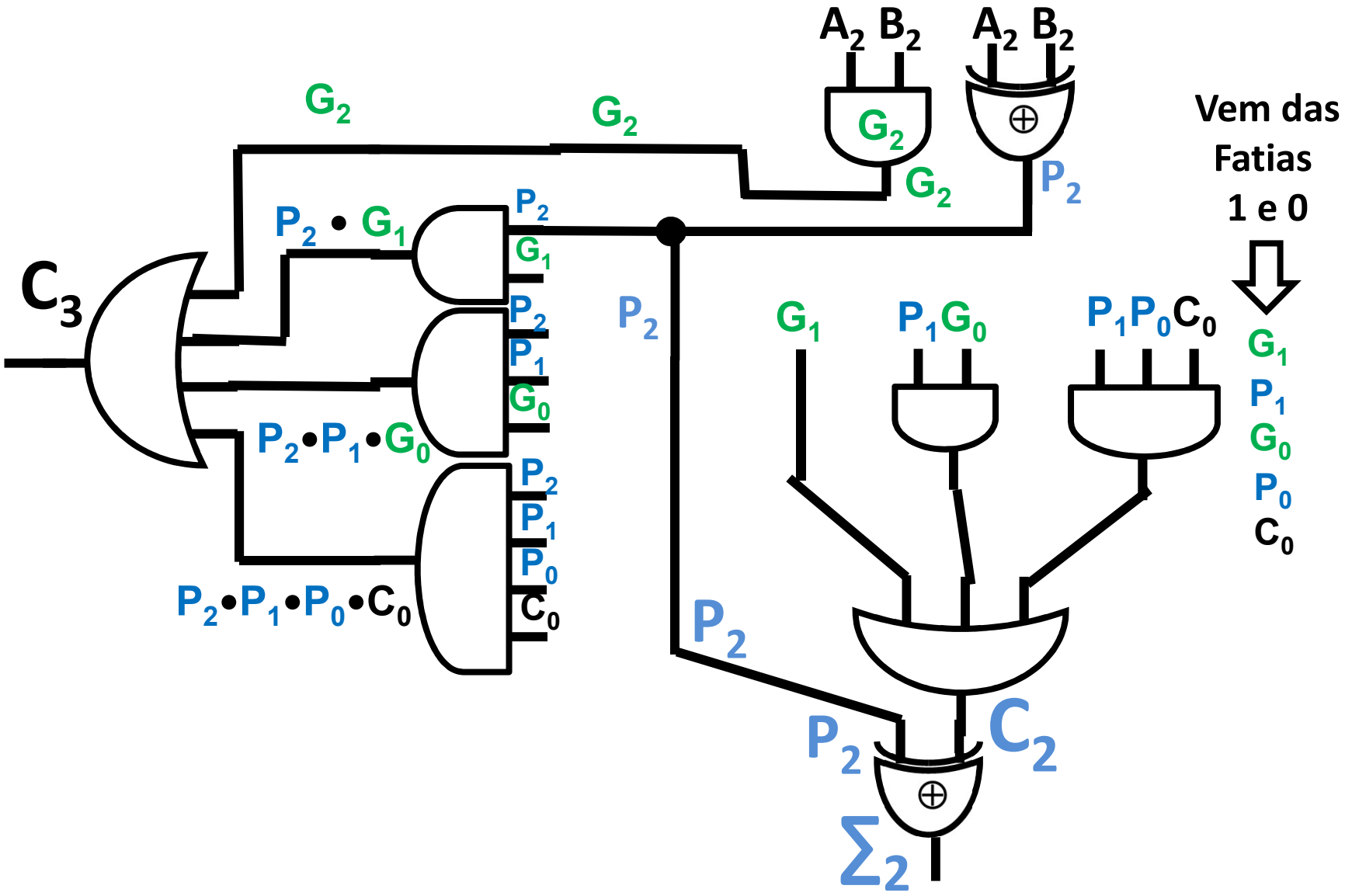
Carry look-ahead – Projeto das Fatias – Fatia 2



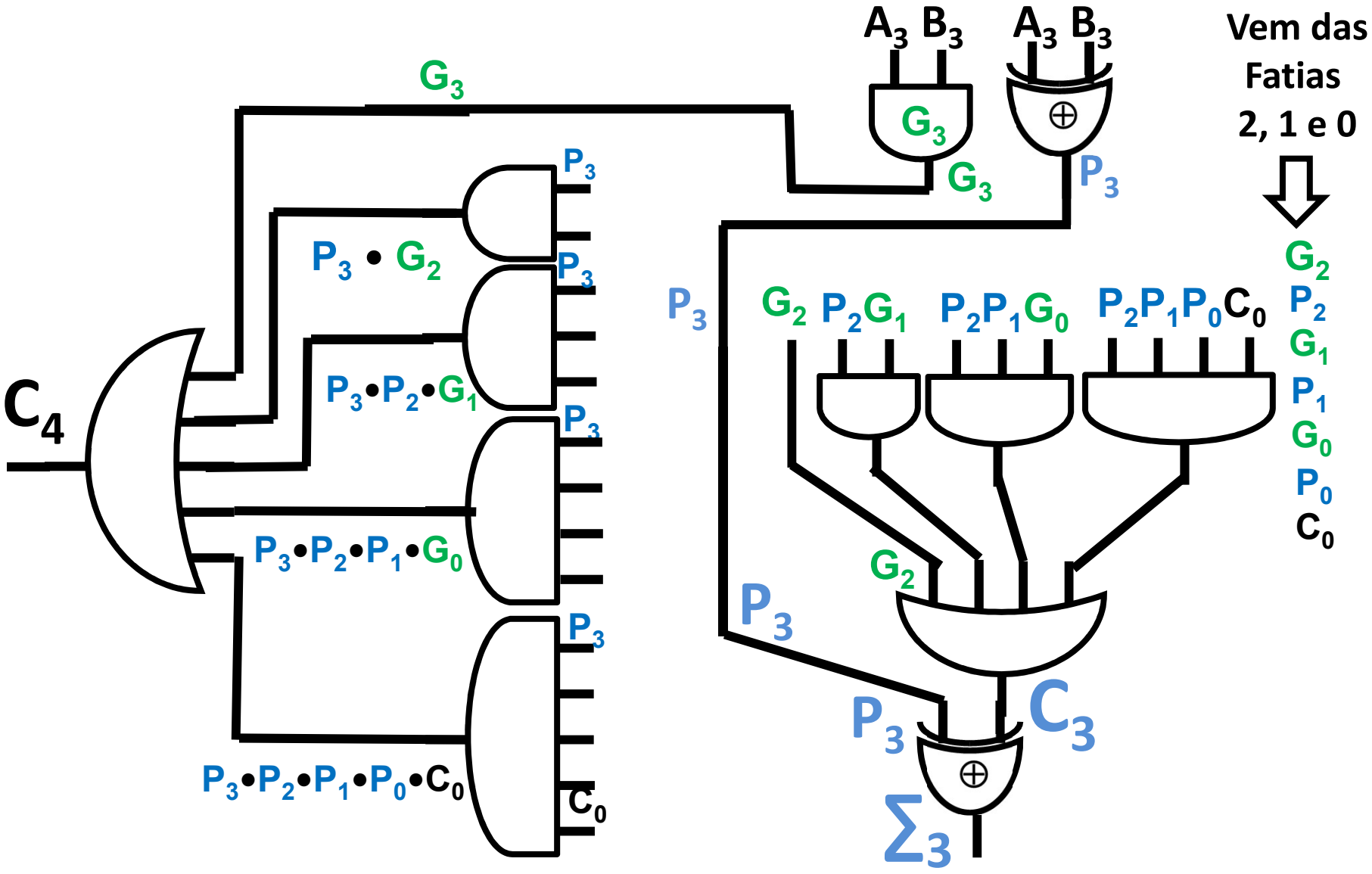
C*arry look-ahead* – Projeto das Fatias – *Carry*3



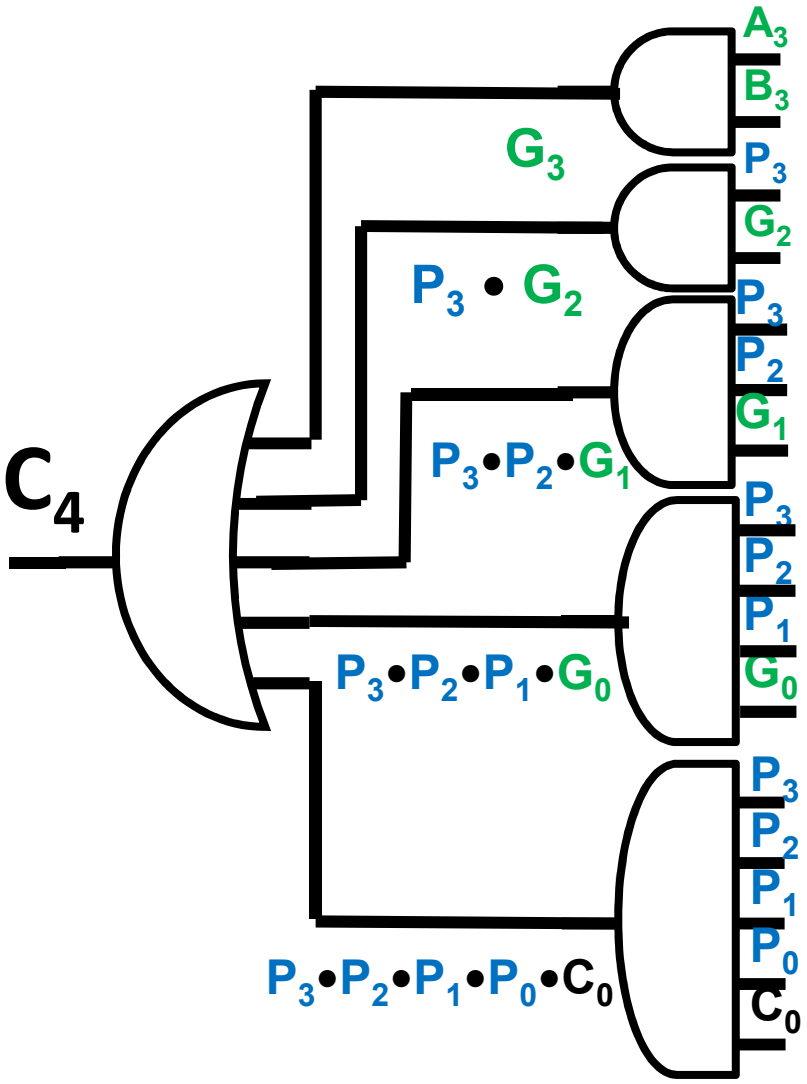
Carry look-ahead – Projeto das Fatias – Fatia 2 + C_3



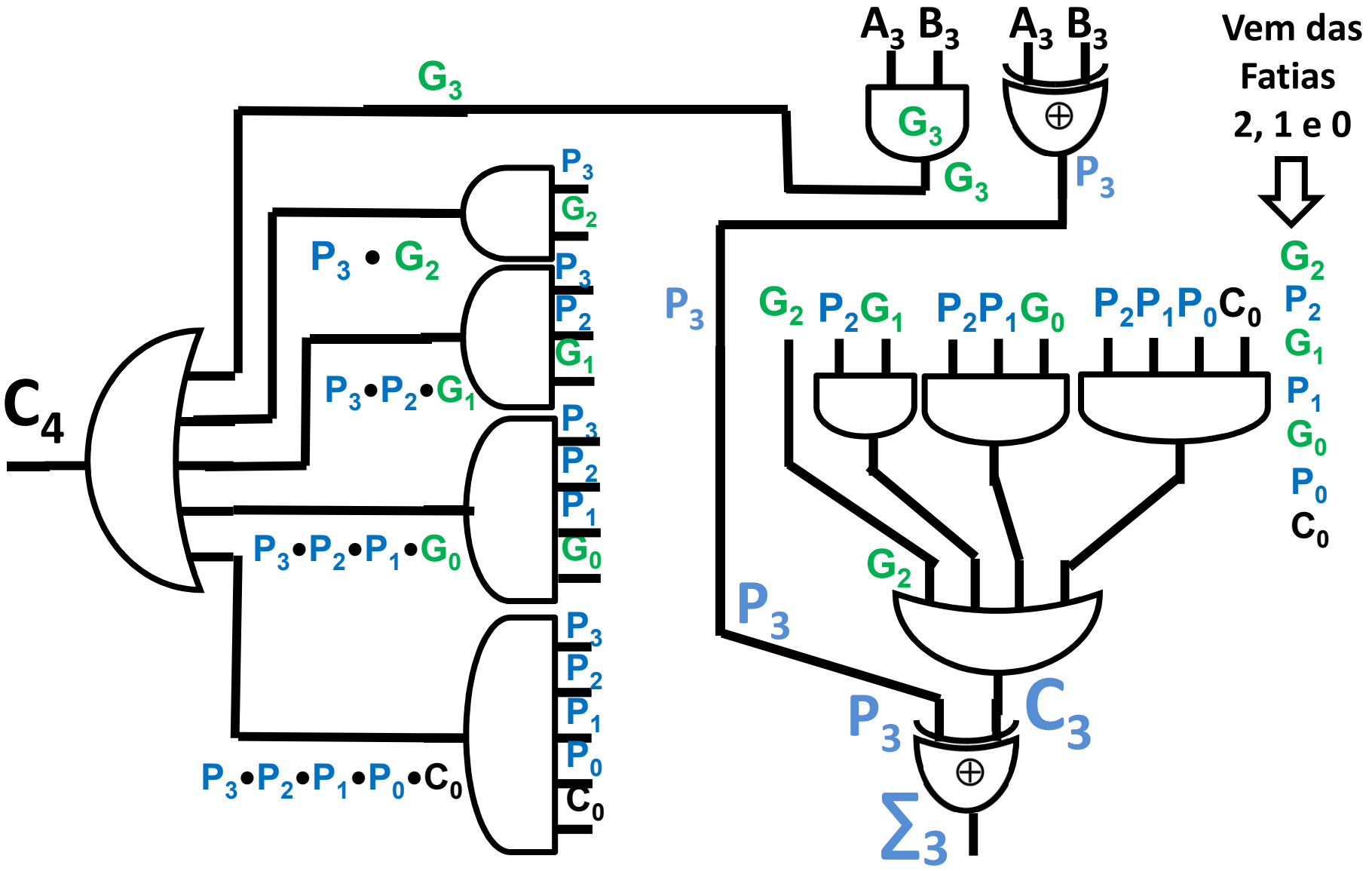
Carry look-ahead – Projeto das Fatias – Fatia 3



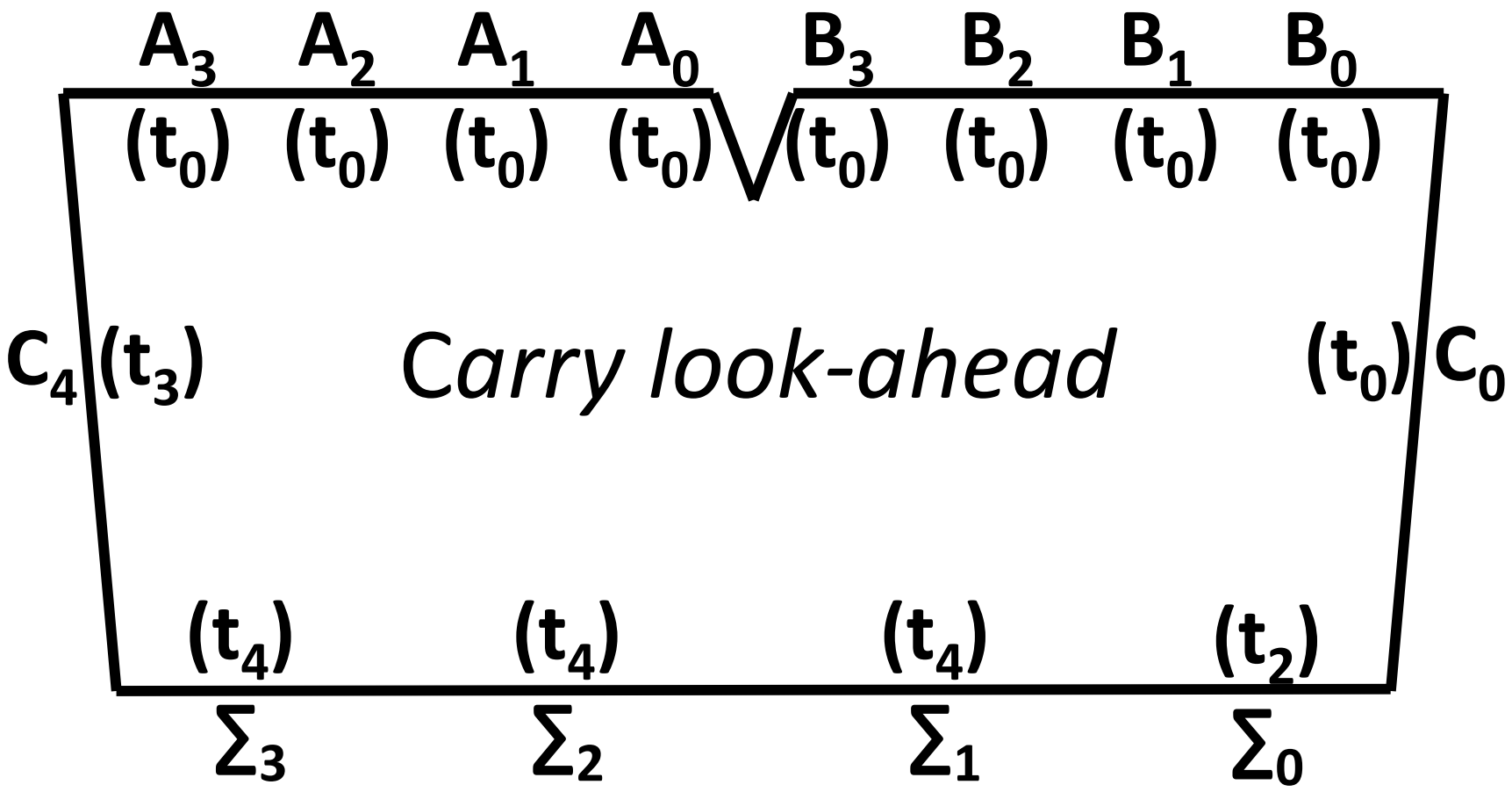
Carry look-ahead – Projeto das Fatias – *Carry*4



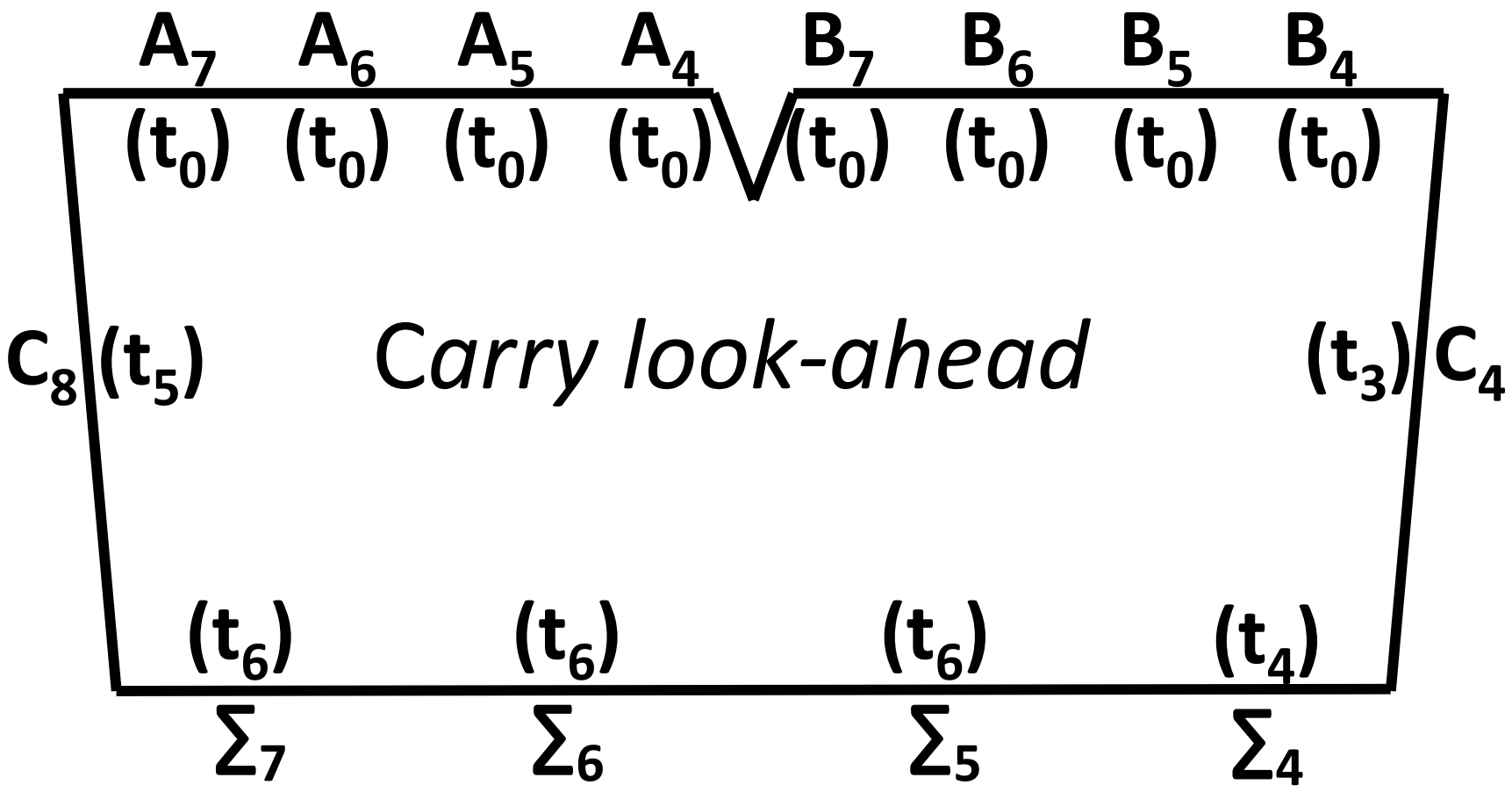
C*arry look-ahead* – Projeto das Fatias – Fatia 3 + *Carry*4



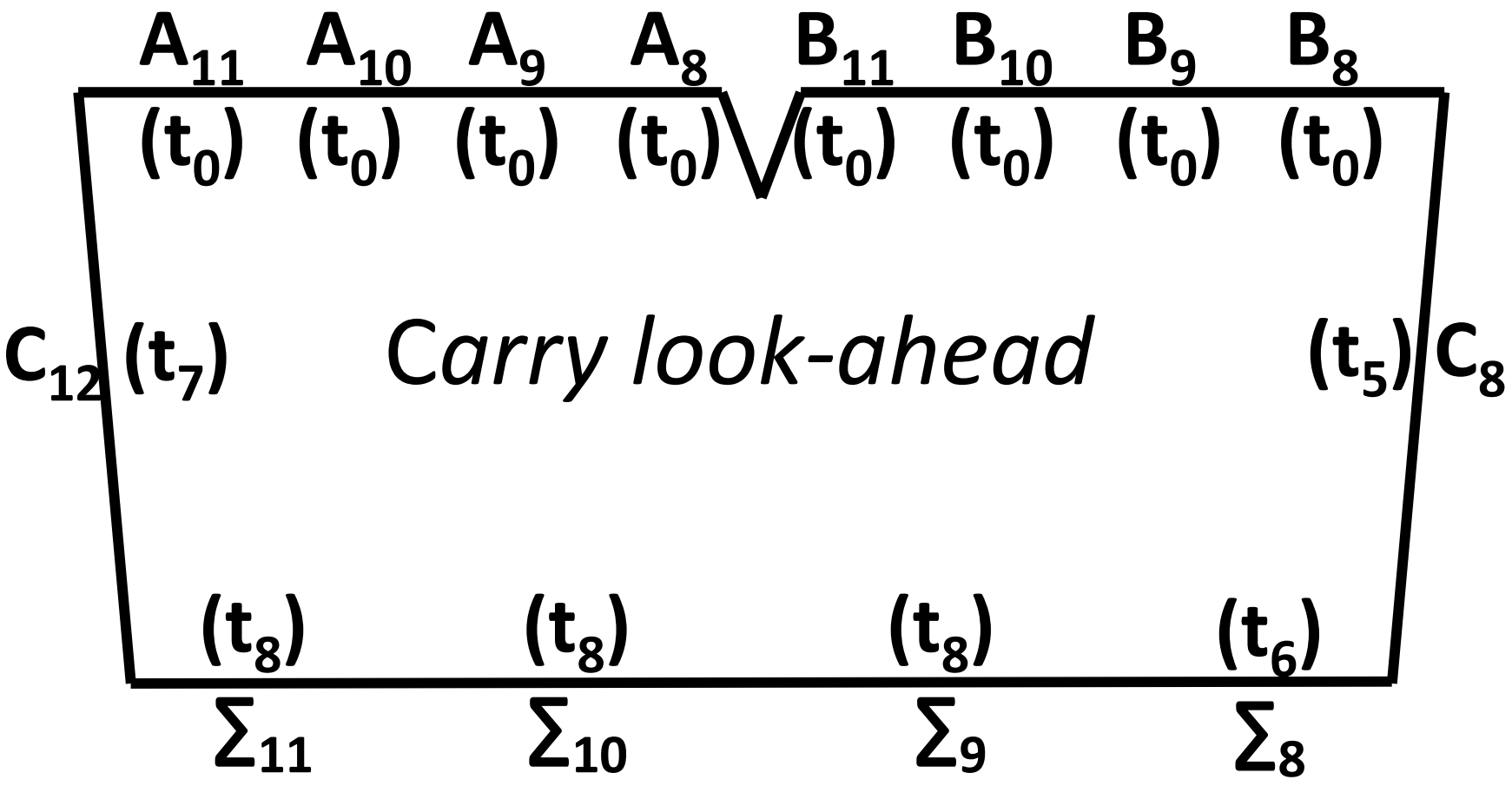
C*arry look-ahead* – **Módulos de 4 bits [1]– Cálculo de t_p**



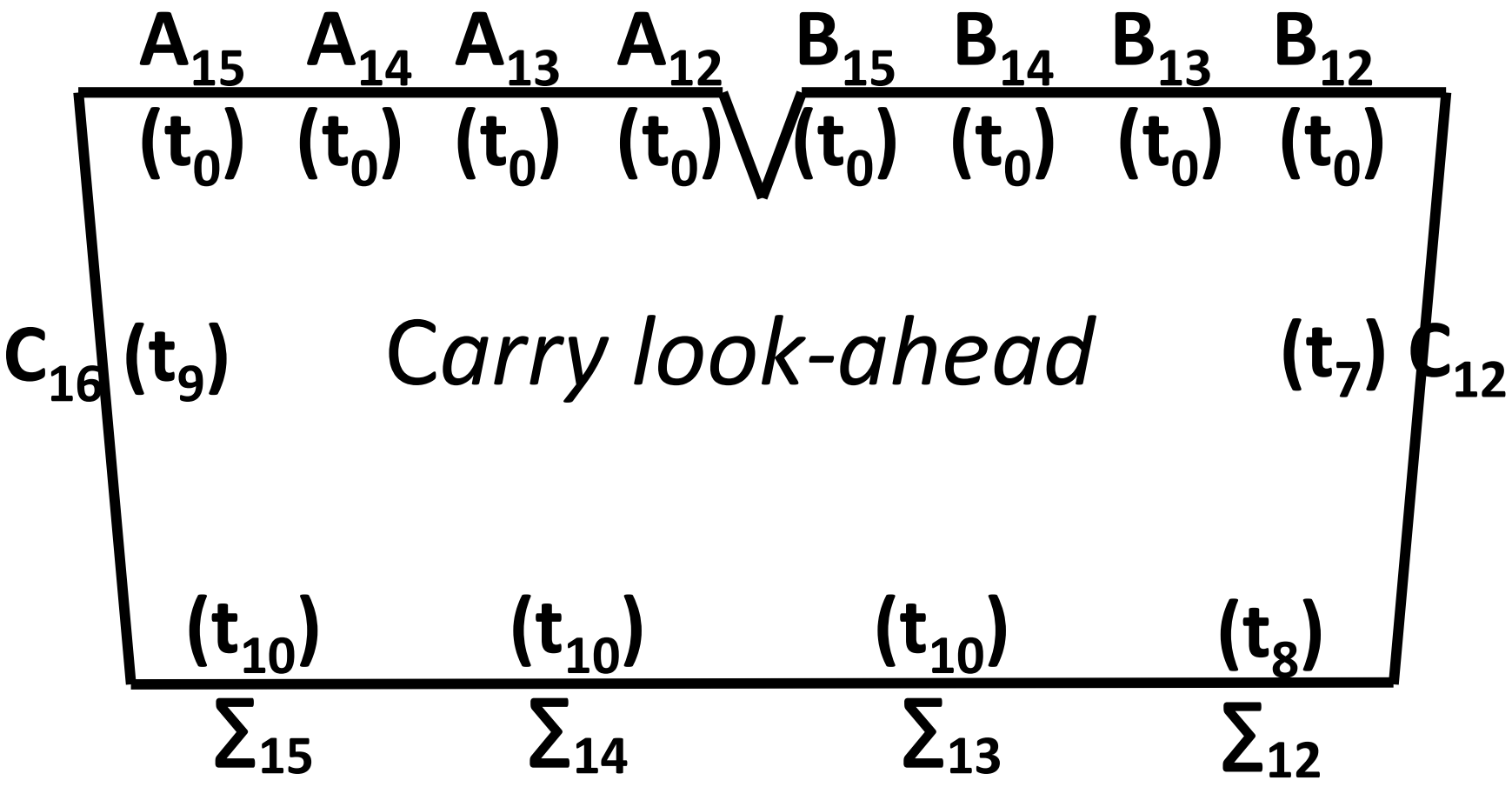
C*arry look-ahead* – **Módulos de 4 bits [2]– Cálculo de t_p**



C*arry look-ahead* – **Módulos de 4 bits [3]– Cálculo de t_p**



Carry look-ahead – **Módulos de 4 bits [4]– Cálculo de t_p**



SLIDE 18-Somador binário de n bits

- Se conexão for feita com outro bloco usando carry look-ahead, podemos escrever:

$$C_{BLOCO4} = G_{BLOCO4} + P_{BLOCO4} \bullet C_0$$

onde:

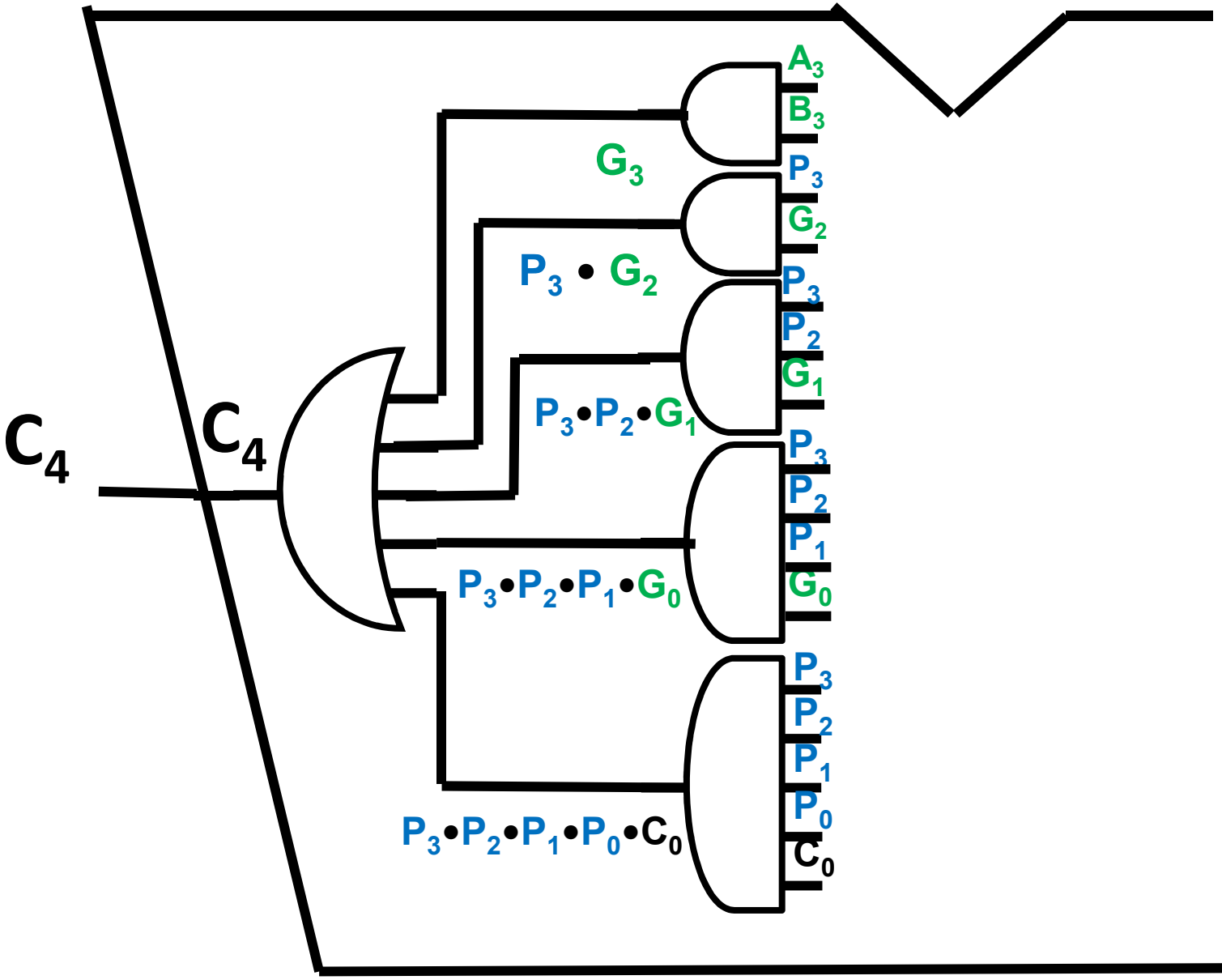
$$G_{BLOCO4} = G_3 + P_3 \bullet G_2 + P_3 \bullet P_2 \bullet G_1 + P_3 \bullet P_2 \bullet P_1 \bullet G_0$$

com $G_i = A_i \bullet B_i$

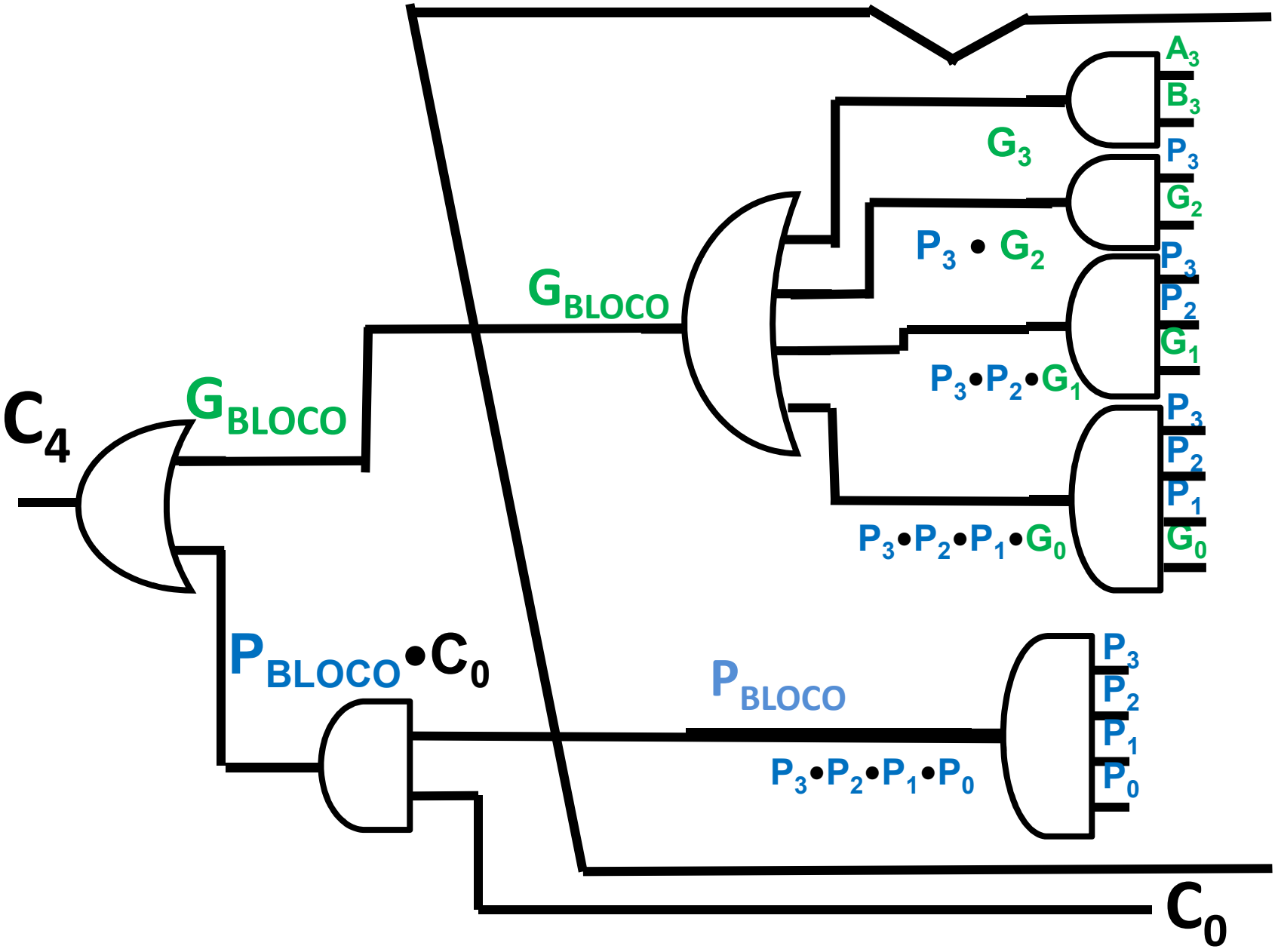
$$P_{BLOCO4} = P_3 \bullet P_2 \bullet P_1 \bullet P_0$$

com $P_i = (A_i \oplus B_i)$

Carry look-ahead – Carry 4 – Interno

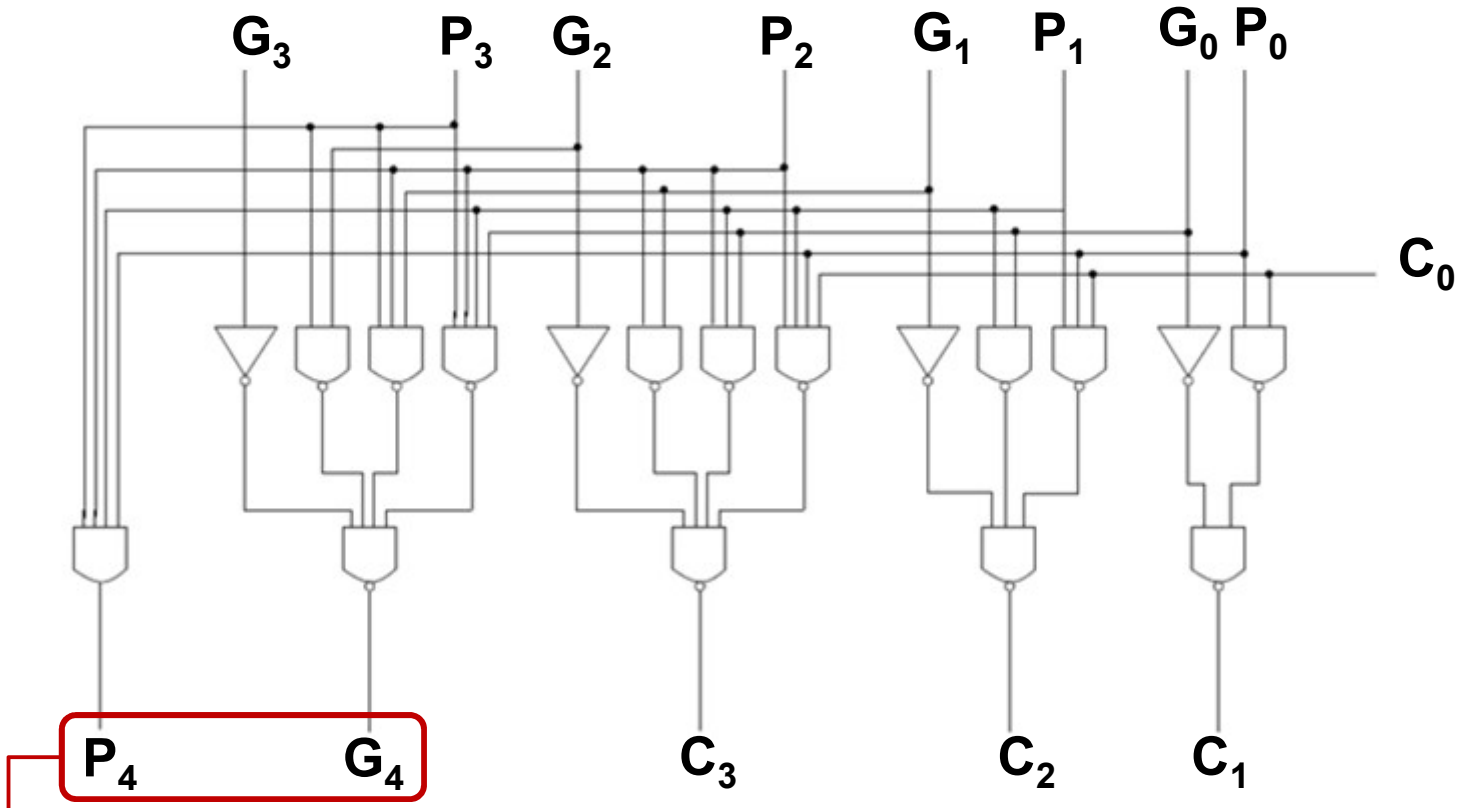


Carry look-ahead – P_{BLOCO} e G_{BLOCO} – Carry 4 – Externo



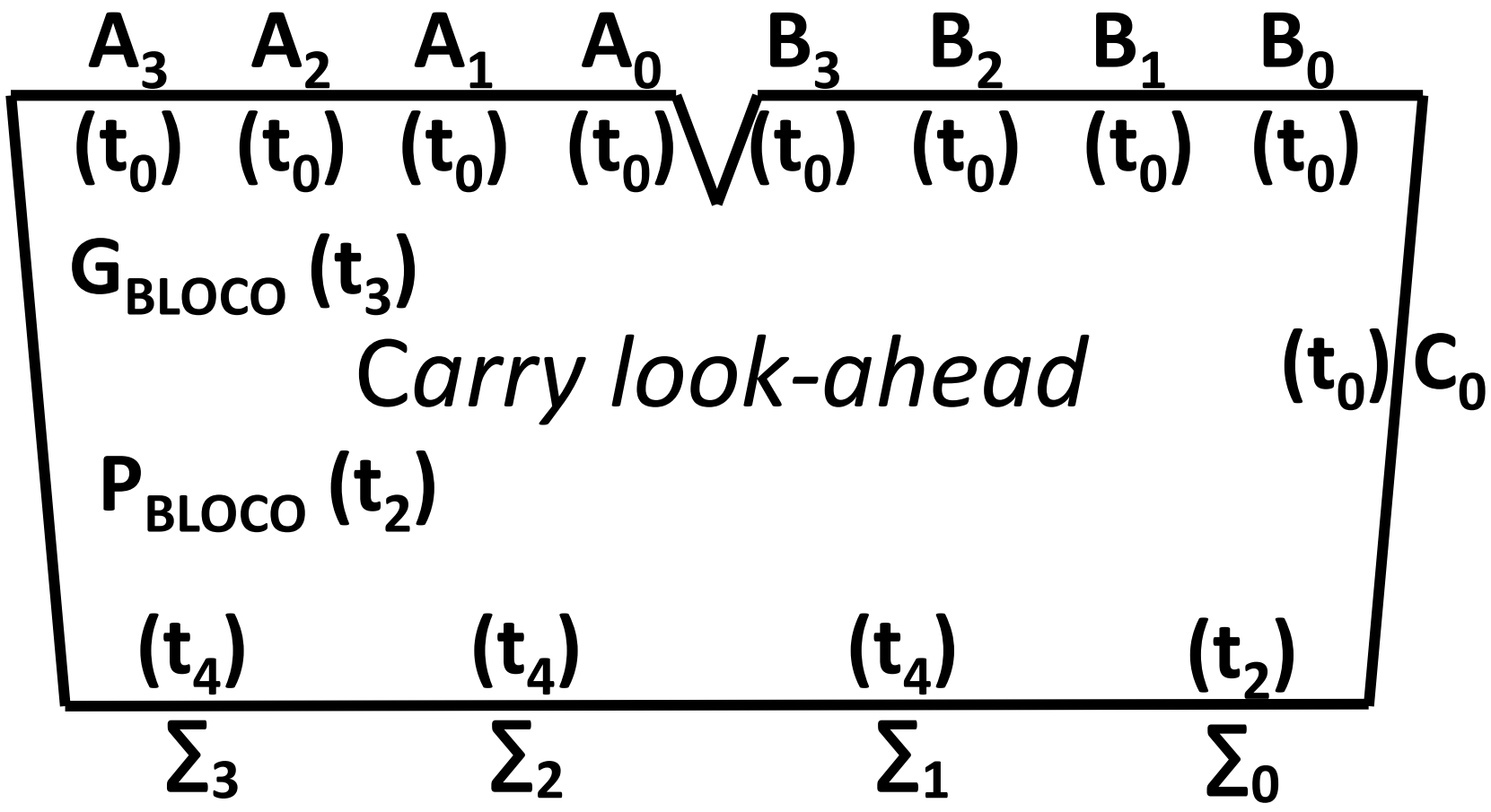
SLIDE 20-Somador binário de n bits

- Carry look-ahead: bloco para **cálculo antecipado** do carry

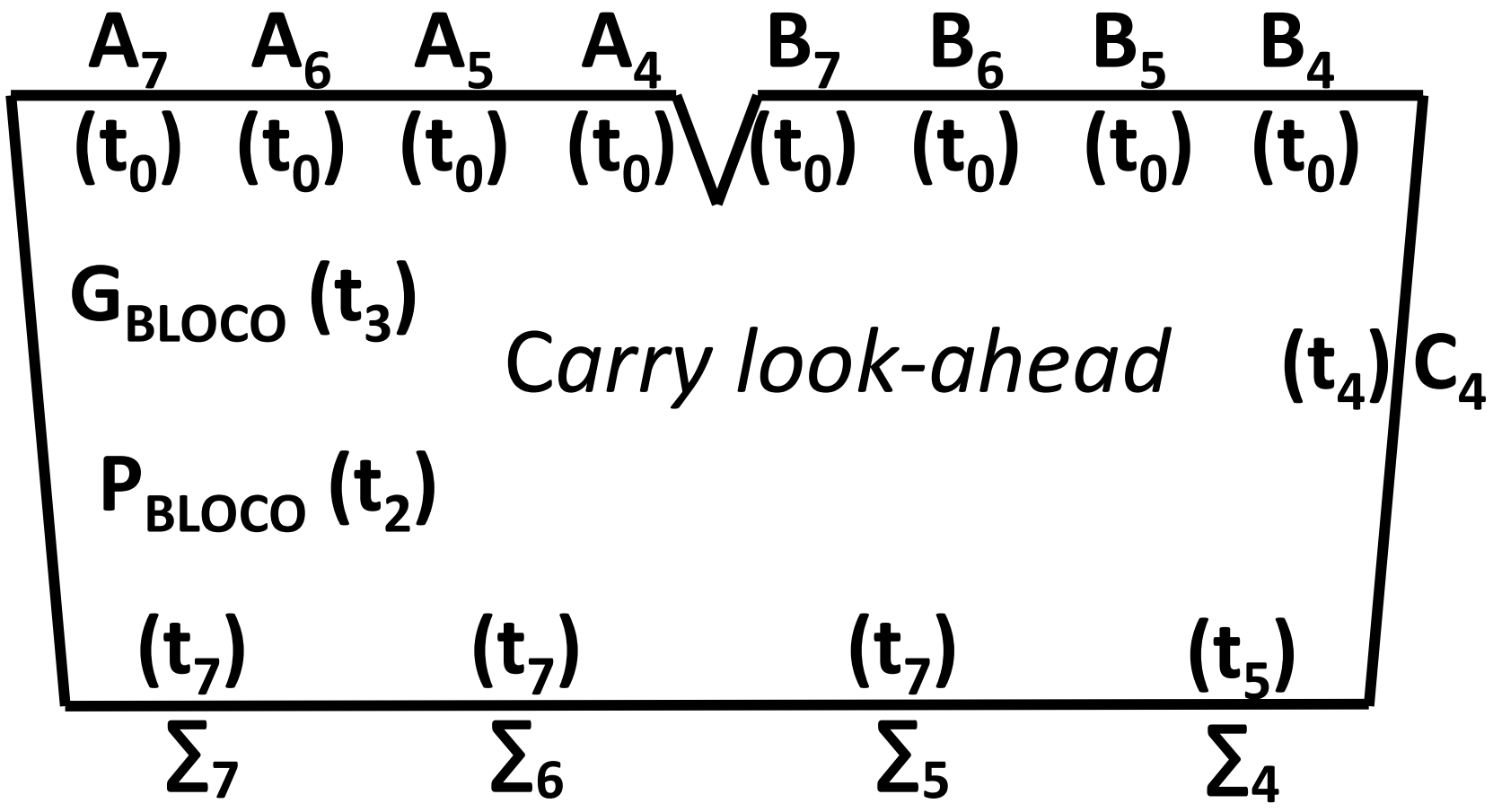


Podem ser usados para calcular C4 (útil para conectar blocos do tipo carry ripple) ou diretamente em um outro bloco de carry look-ahead

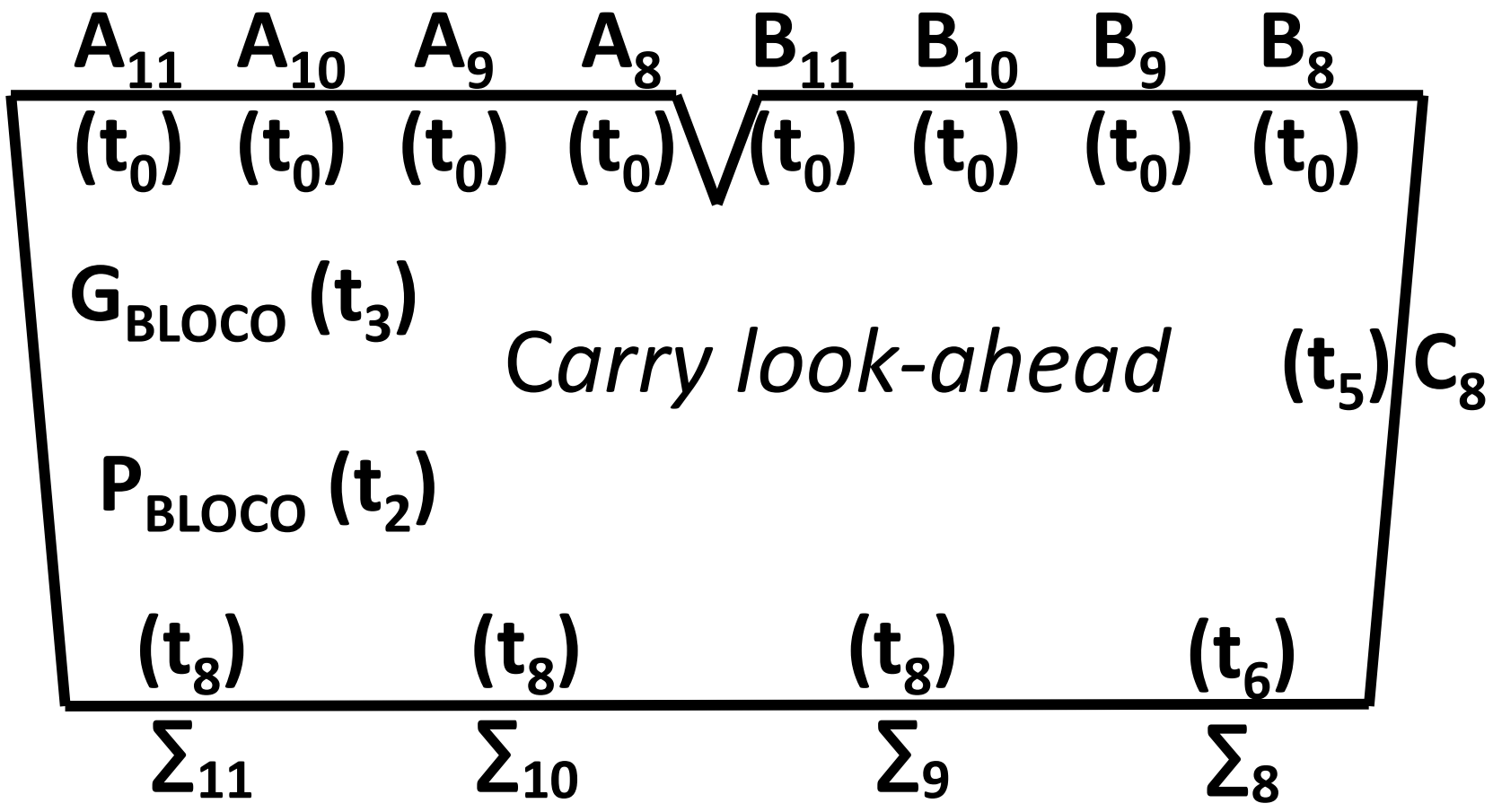
C*arry look-ahead* – Módulos 4 bits [1] – P_{BLOCO} e G_{BLOCO} – Carry 4 – Externo



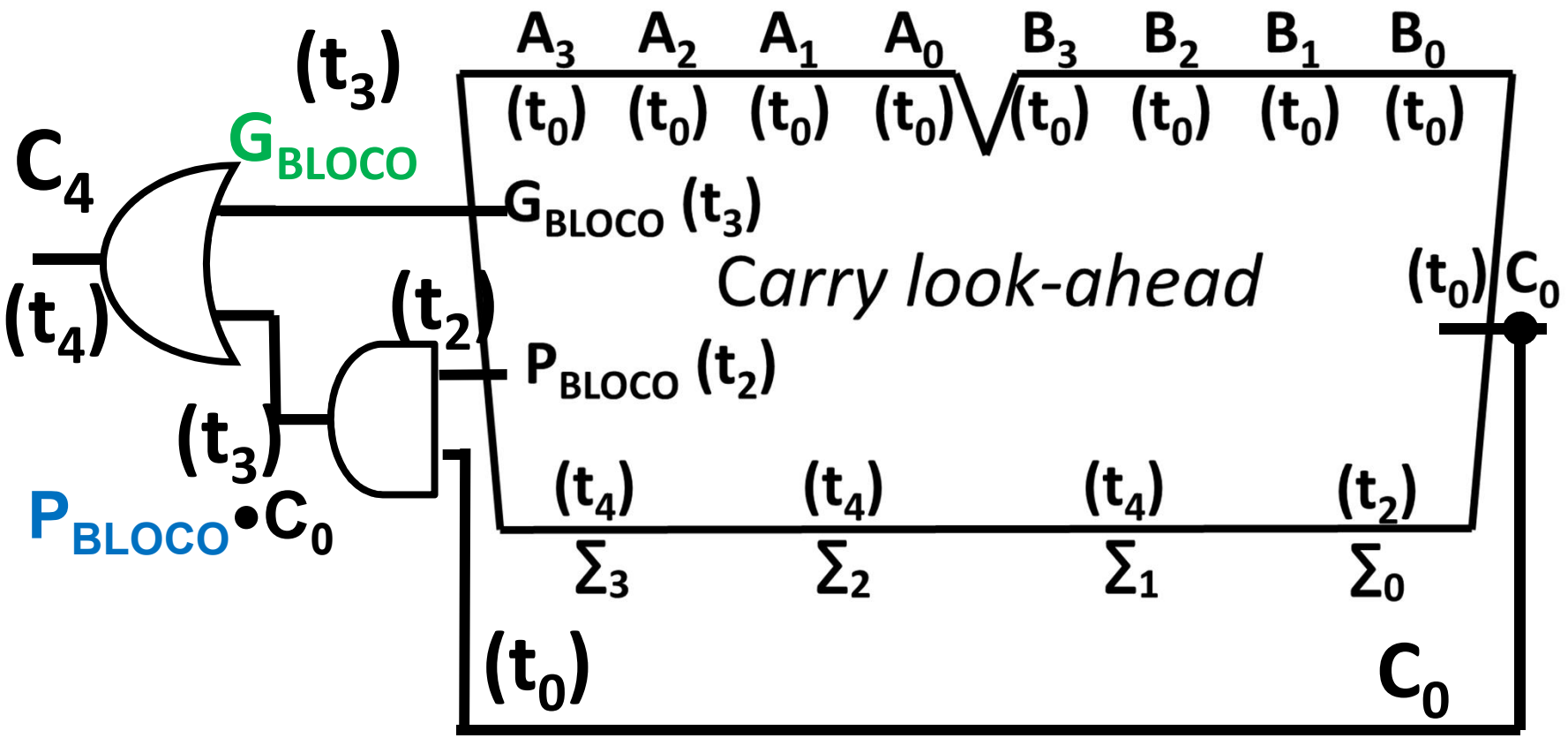
C*arry look-ahead* – Módulos 4 bits [2] – P_{BLOCO} e G_{BLOCO} – C_4 Externo- obj



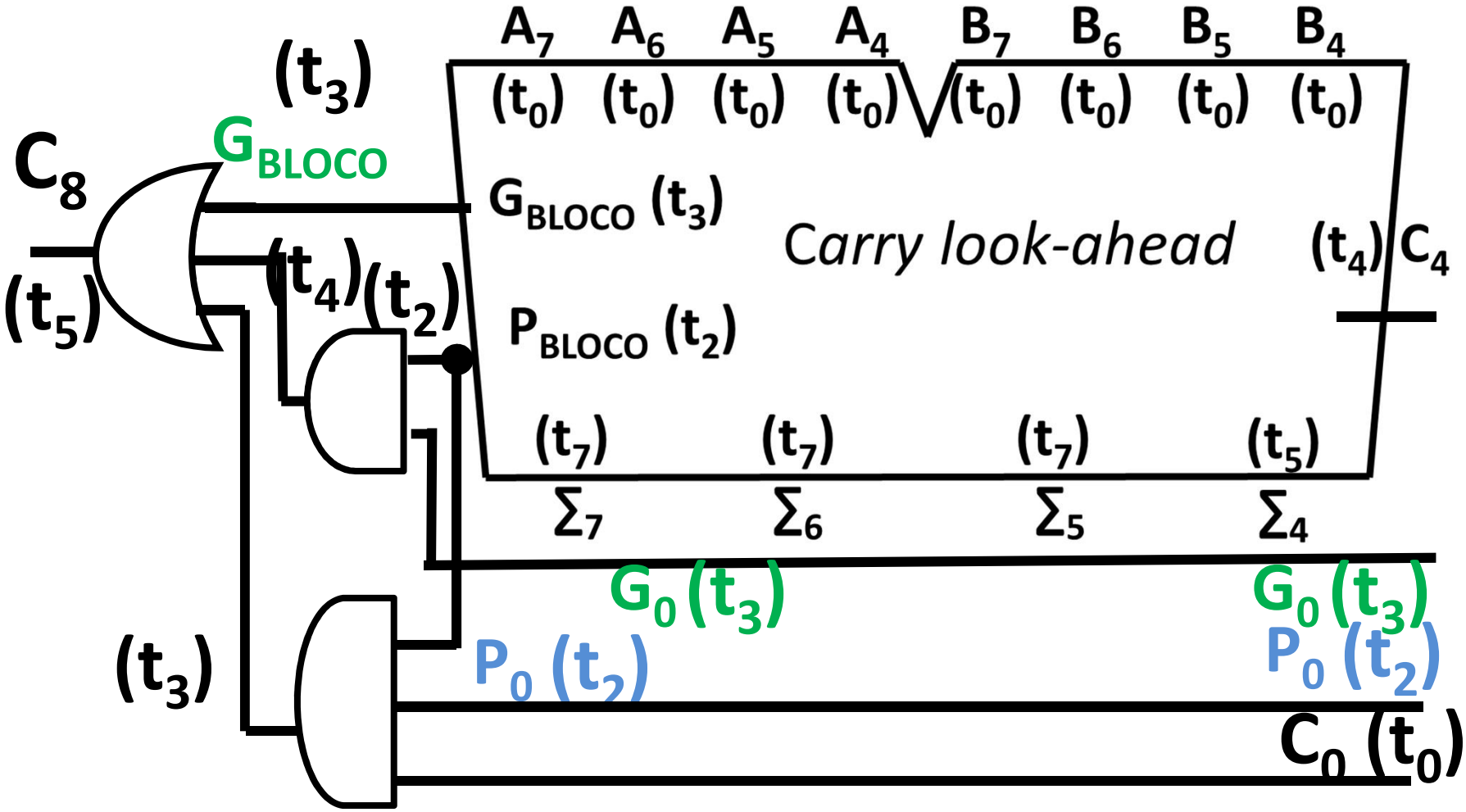
C*arry look-ahead* – Módulos 4 bits [3] – P_{BLOCO} e G_{BLOCO} – C_4 Externo- obj



Carry look-ahead – Módulos 4 bits [1] – P_{BLOCO} e G_{BLOCO} – C_4 Externo obj+PNG



Carry look-ahead – Módulos 4 bits [2] – P_{BLOCO} e G_{BLOCO} – C_4 Externo- obj PNG



C*arry look-ahead* – Módulos 4 bits [1] e [2] – P_{BLOCO} e G_{BLOCO} – C_4 Externo PNG