PCS 3115 – Sistemas Digitais I

Circuitos Combinatórios:
Somadores e Subtratores

Suporte para EAD

Parte III:
Somadores – *Ripple Carry Adder* – *Análise de tempos de propagação*.

Aula: 14 – Data: 27/04 (S)

Prof. Dr. Marco Túlio Carvalho de Andrade

versão: 1.0 (Abril/2020)

Somador binário de n bits – SLIDE 11

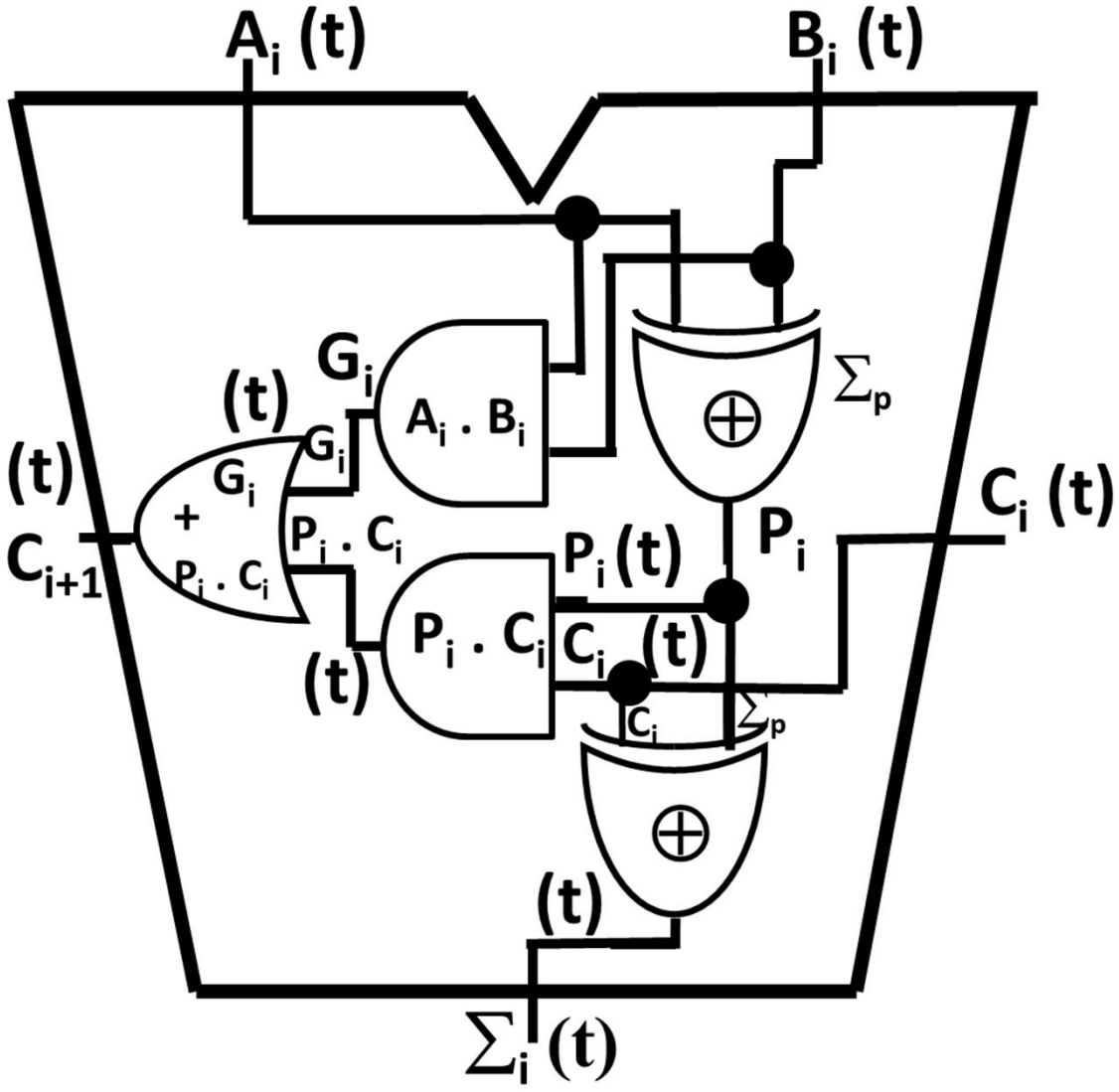
- **Pergunta**: dado um somador completo de 1 bit, como obter um somador de n bits?
- **Resposta simples**: associação em série (cascateamento) de *n* somadores completos
- **Resultado**: “somadores com propagação de vai-um”, ou *ripple adders*:
  - Cada somador completo trata um bit da soma
  - O “Vai-um” de um estágio se conecta ao “Vem-um” do estágio seguinte.
  - O “Vem-um” do estágio menos significativo costuma ficar em 0 (ou usa-se um meio-somador)

Somador binário de n bits

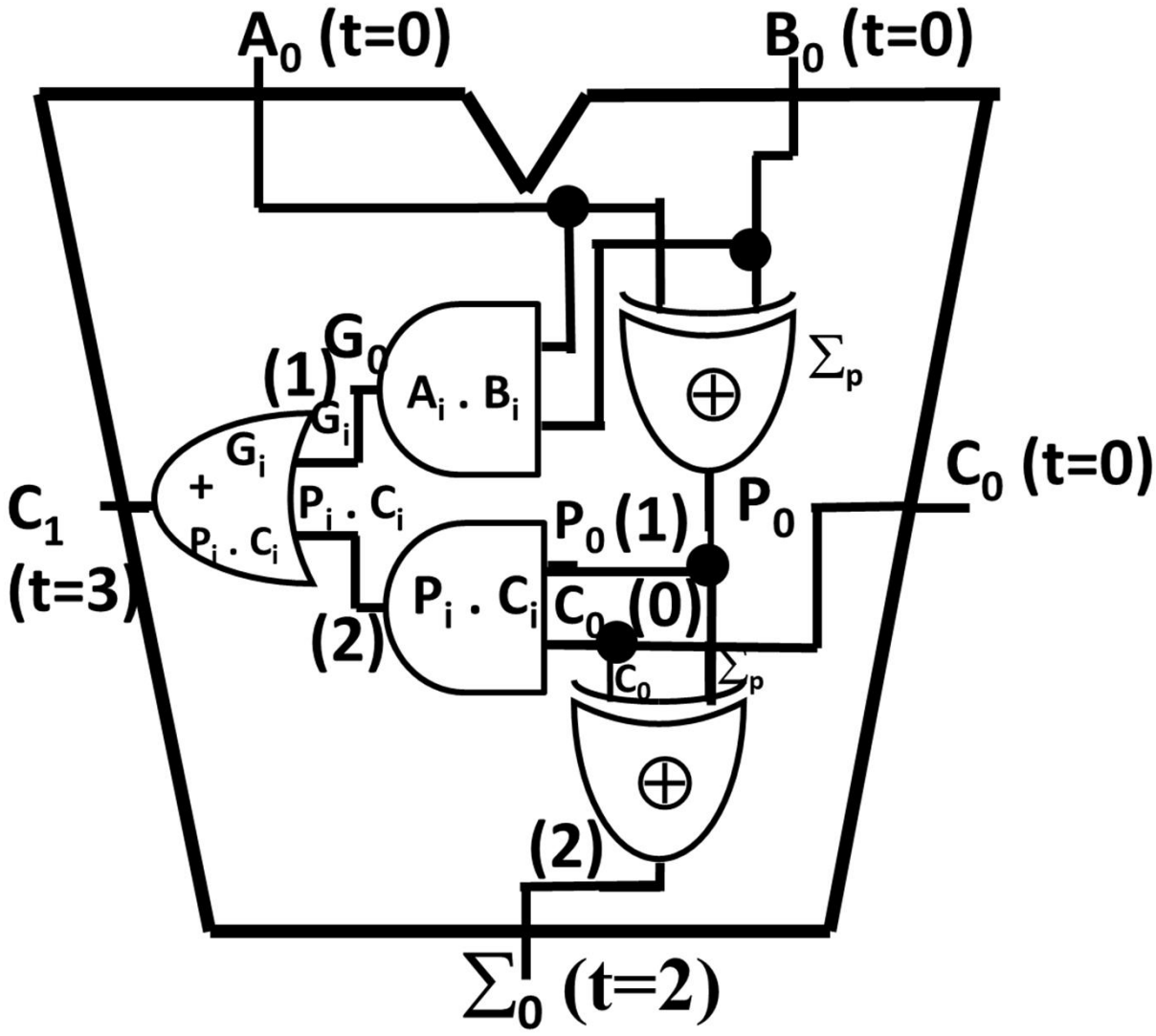
- **Vamos entender o problema – Análise de tempos de propagação dos sinais e obtenção de $\sum_i$ e C_{i+1}.**
- Somadores com propagação de "vai-um" - *Ripple Adders*:

  - Cada somador completo trata um bit da soma;
  - **Para i>0 -** O "Vai-um" de saída de um estágio menos significativo se conecta ao "Vem-um" de entrada do estágio mais significativo seguinte;
  - **Para i=0 -** O "Vem-um" do estágio menos significativo (Fatia zero "0") costuma ficar em "0" (ou usa-se um meio-somador, que não tem "Vem-um" na entrada.

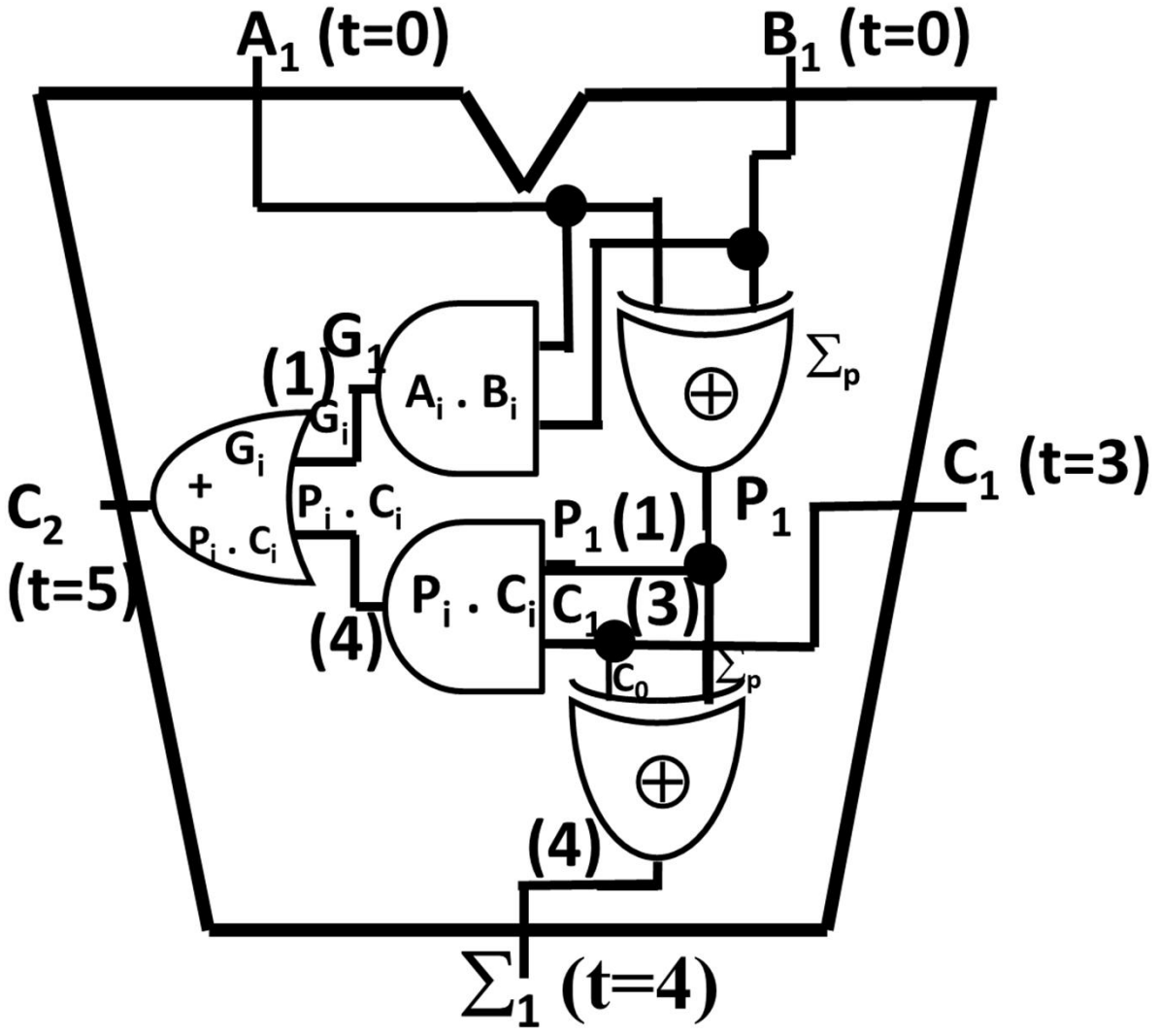
Ripple Adder – n bits – Fatia **i** – Genérica



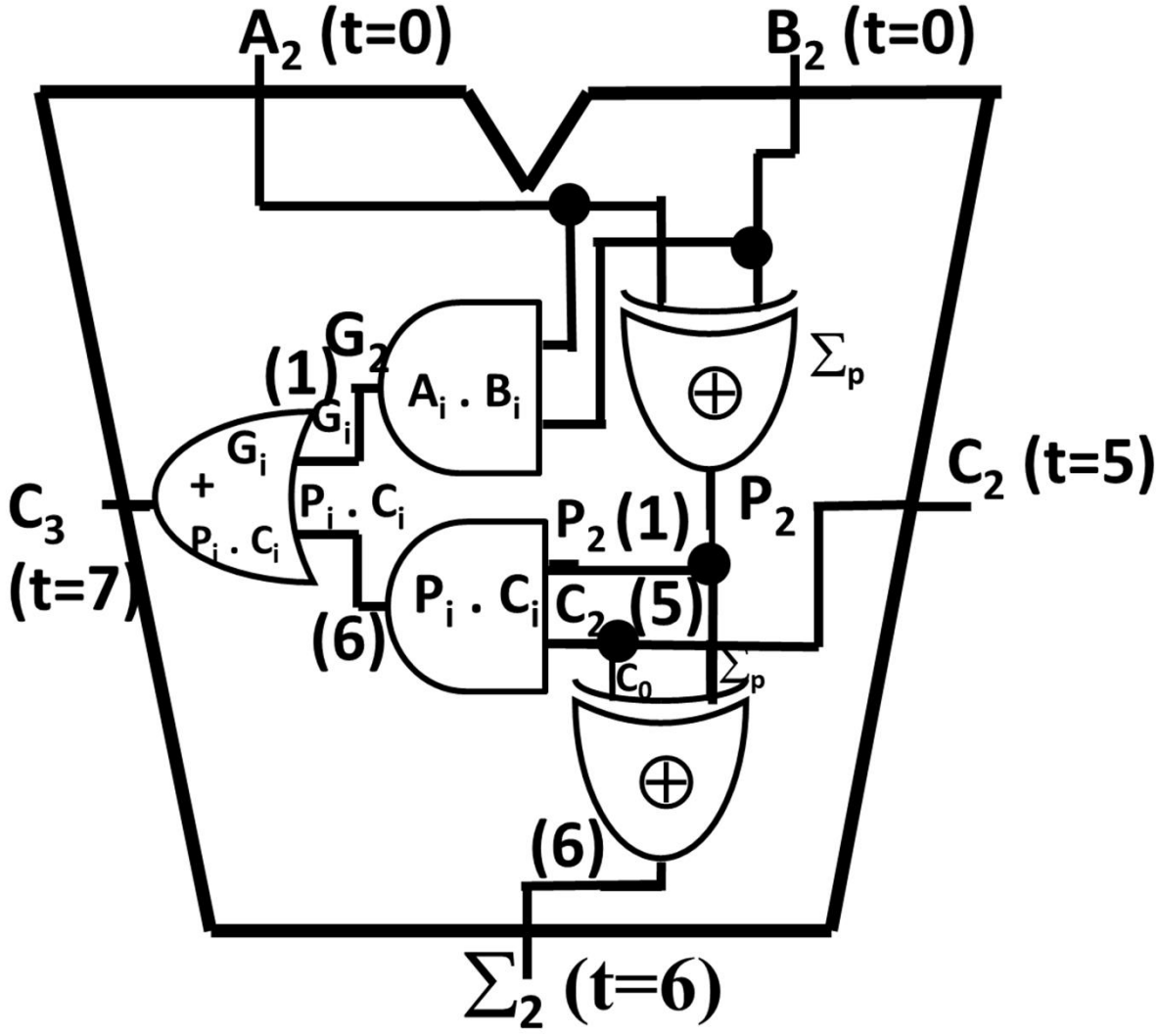
Ripple Adder – n bits – Fatia **i = 0**



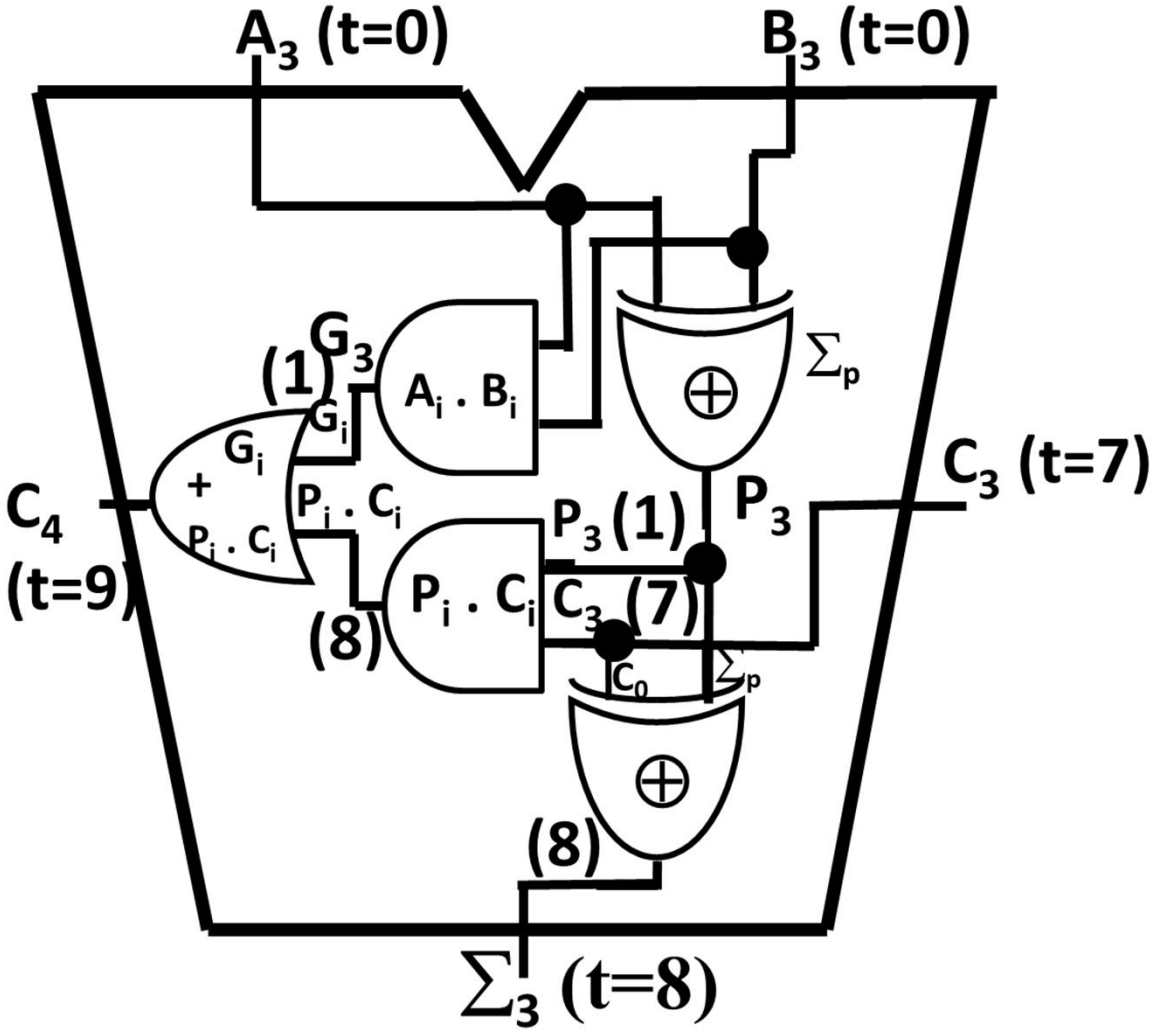
Ripple Adder – n bits – Fatia **i = 1**



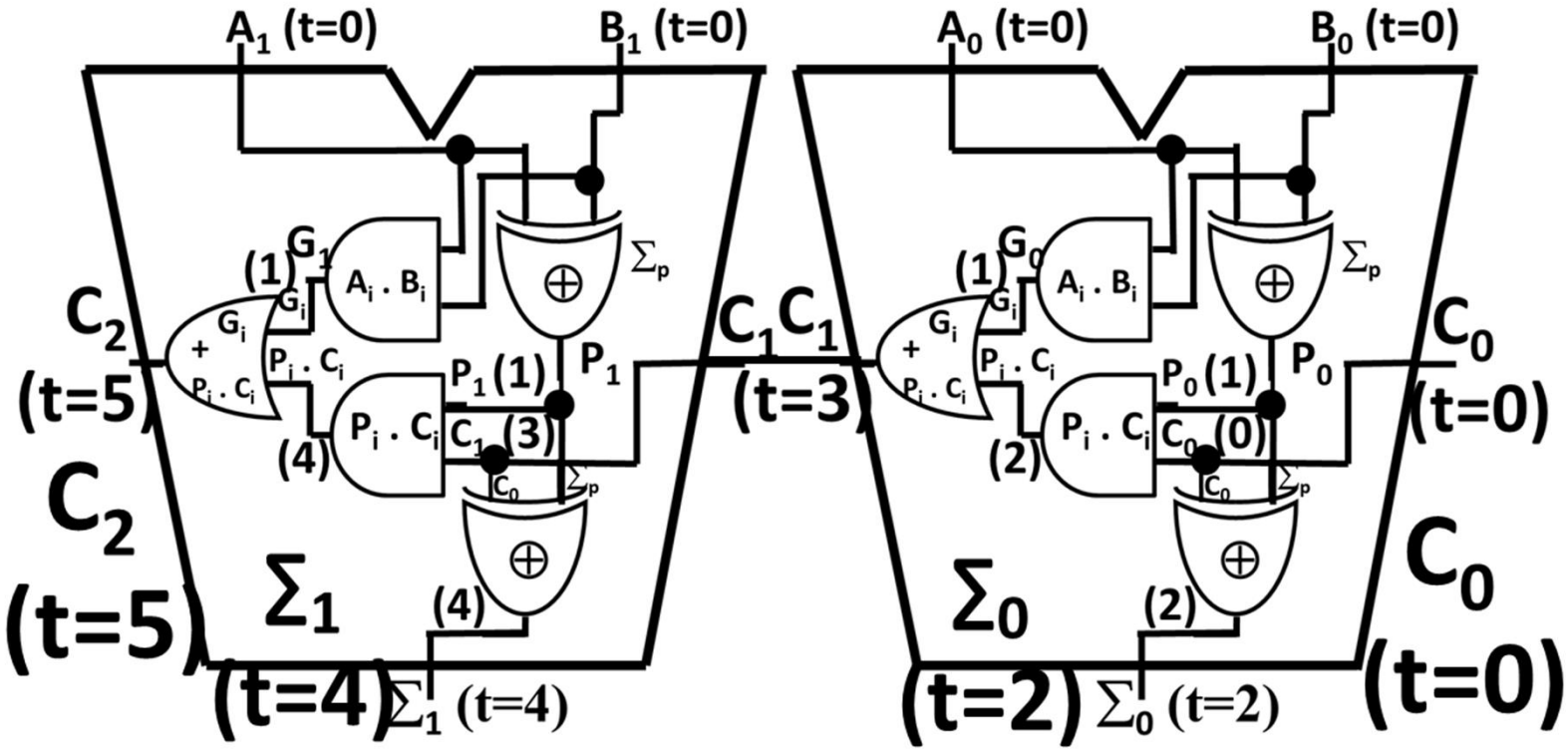
Ripple Adder – n bits – Fatia **i = 2**



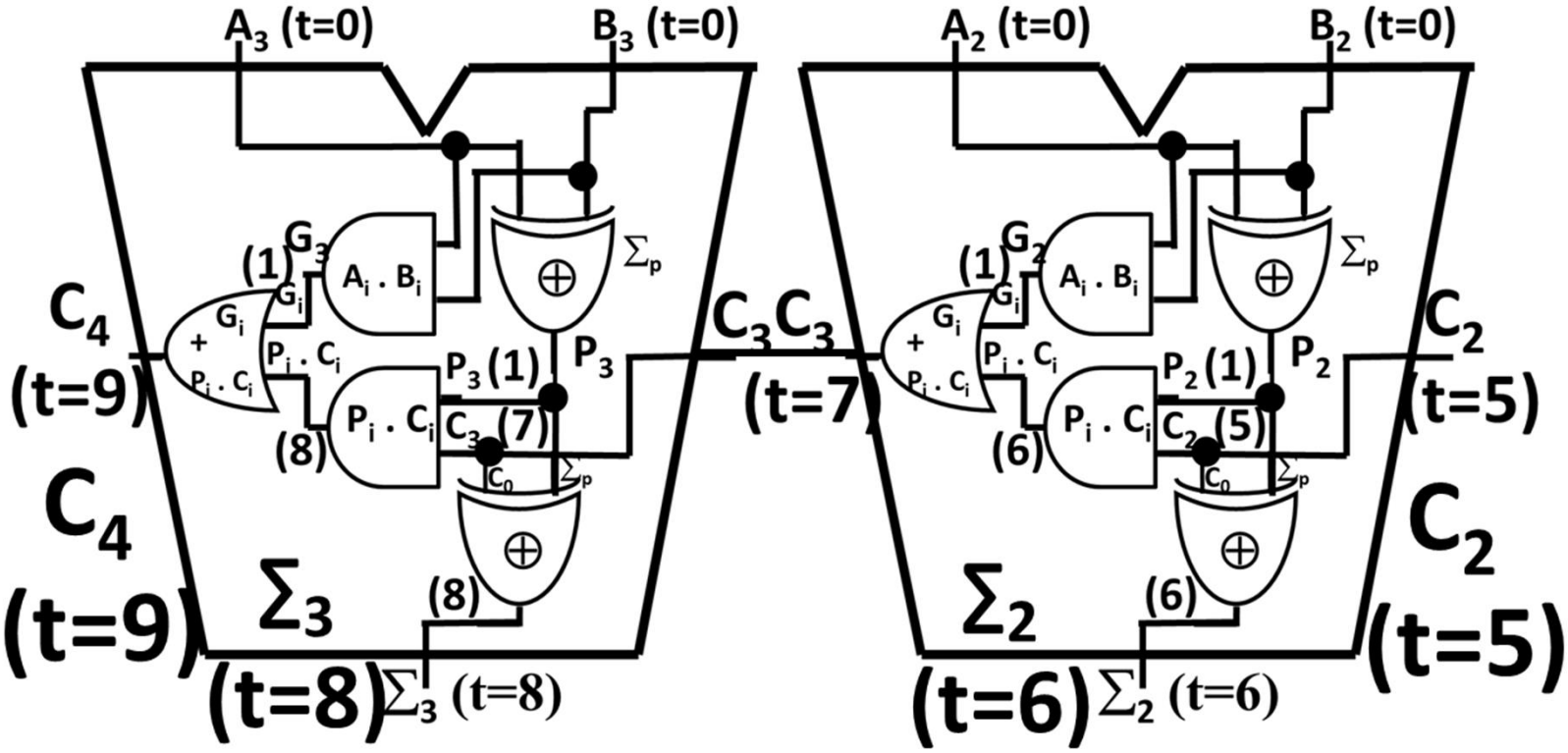
Ripple Adder – n bits – Fatia **i = 3**



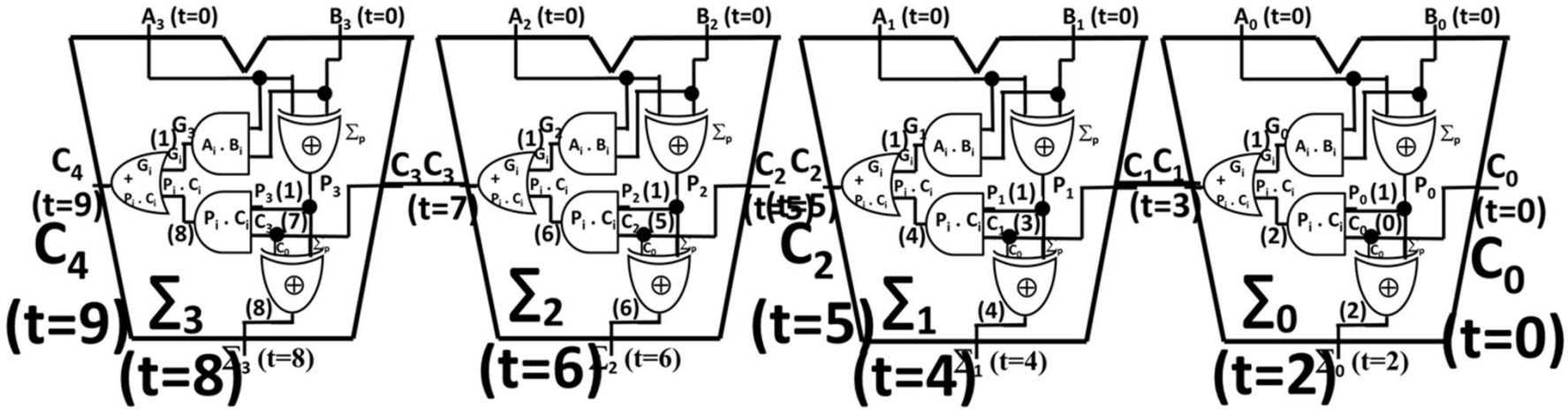
Ripple Adder – n = 4 bits – Fatias 1 e 0 em cascata



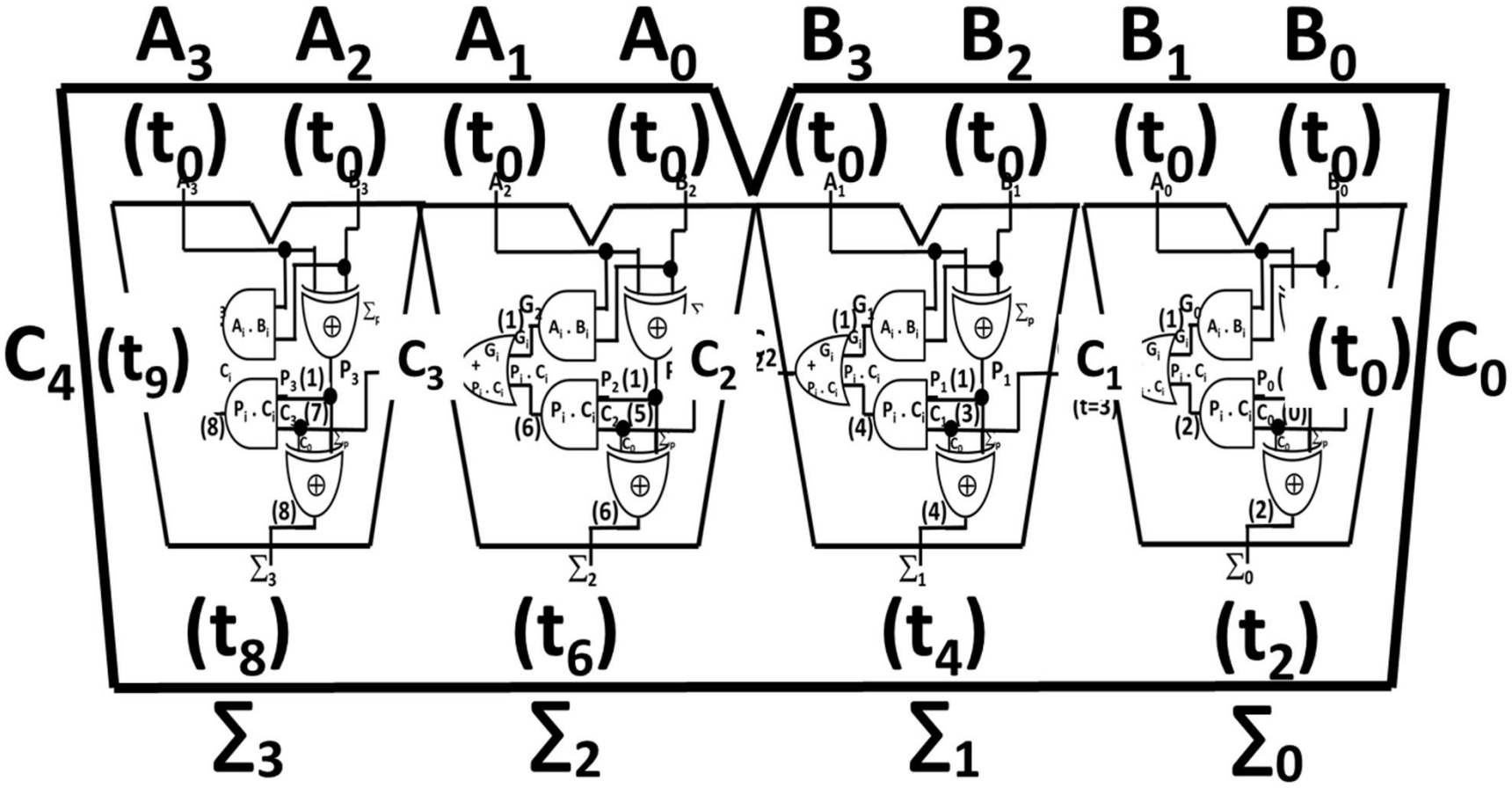
Ripple Adder – n = 4 bits – Fatias 3 e 2 em cascata



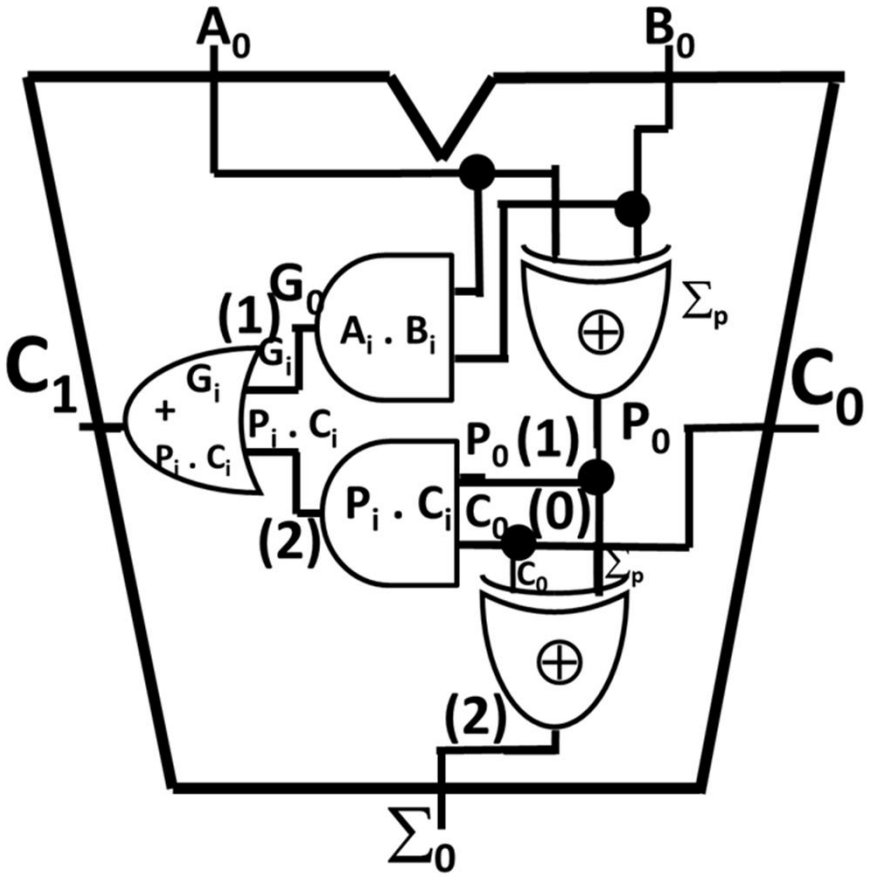
Ripple Adder – 4 bits – Fatias **3, 2, 1, 0** – Cálculo de t_p



Ripple Adder – 4 bits – Cálculo de t_p



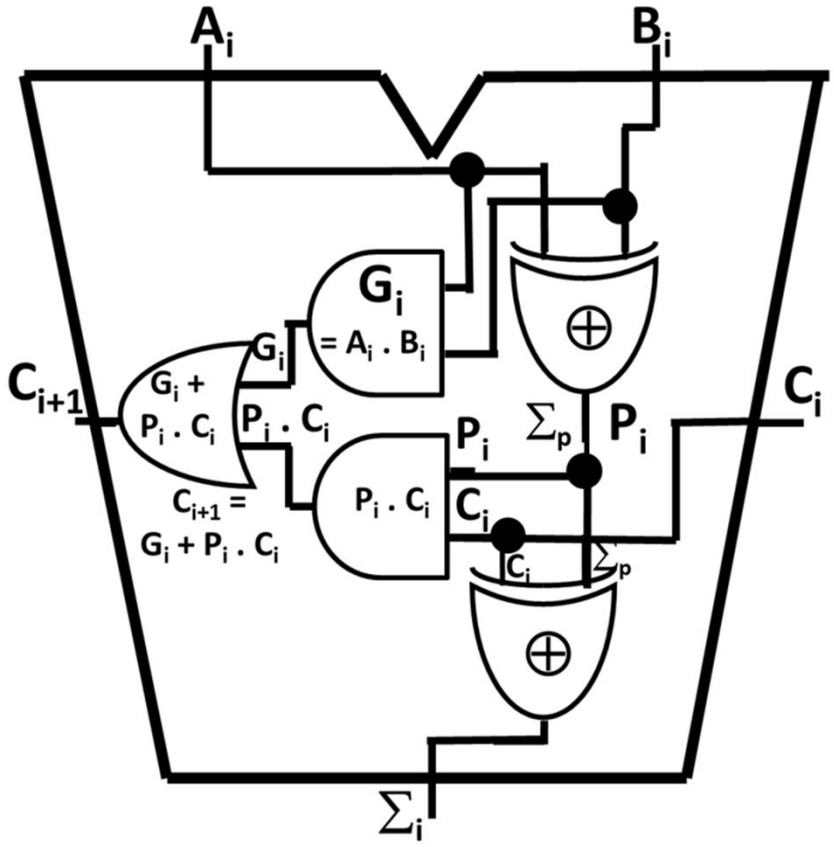
Ripple Adder – Generalização para n bits – Cálculo t_p – **Fatia 0**



$$t_{p:\sum 0} = t_{p:\sum p0} + 1.t_p = 2.t_p$$

$$t_{p:C1} = t_{p:\sum p0} + 2.t_p = 3.t_p$$

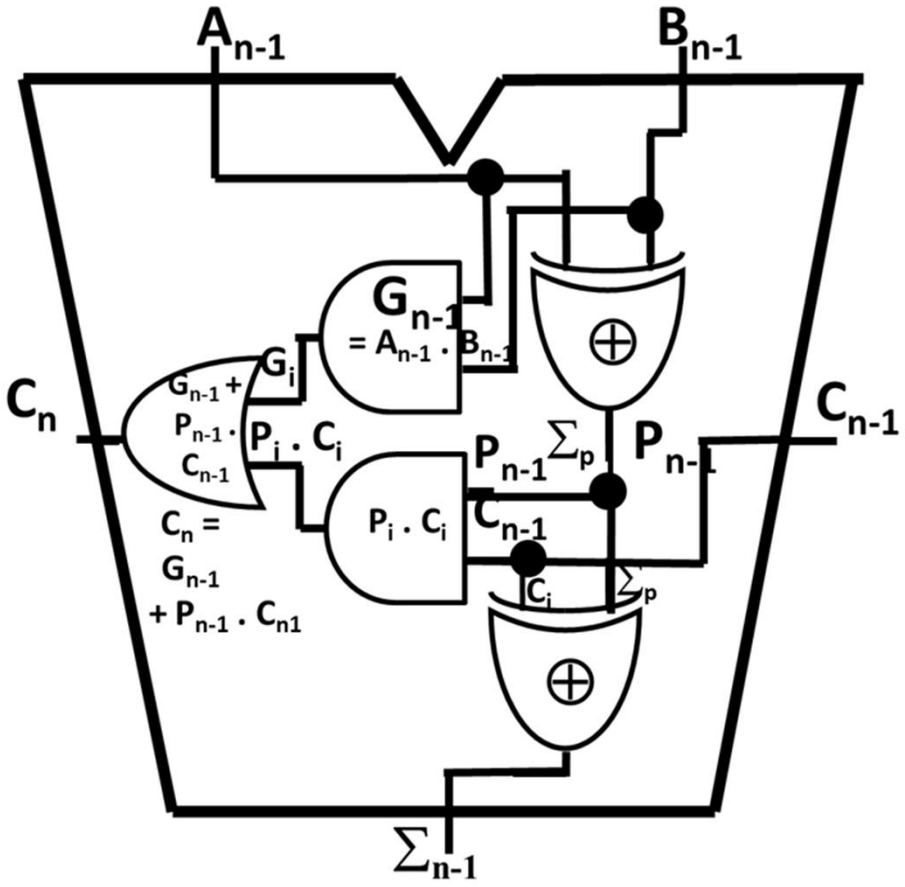
Ripple Adder – Generalização para n bits – Cálculo t_p – **Fatia i**



$$t_{p:\sum i} = máx[t_{p:\sum pi}; t_{p:Ci}] + 1.t_p$$

$$t_{p:Ci+1} = máx[t_{p:\sum pi}; t_{p:Ci}] + 2.t_p$$

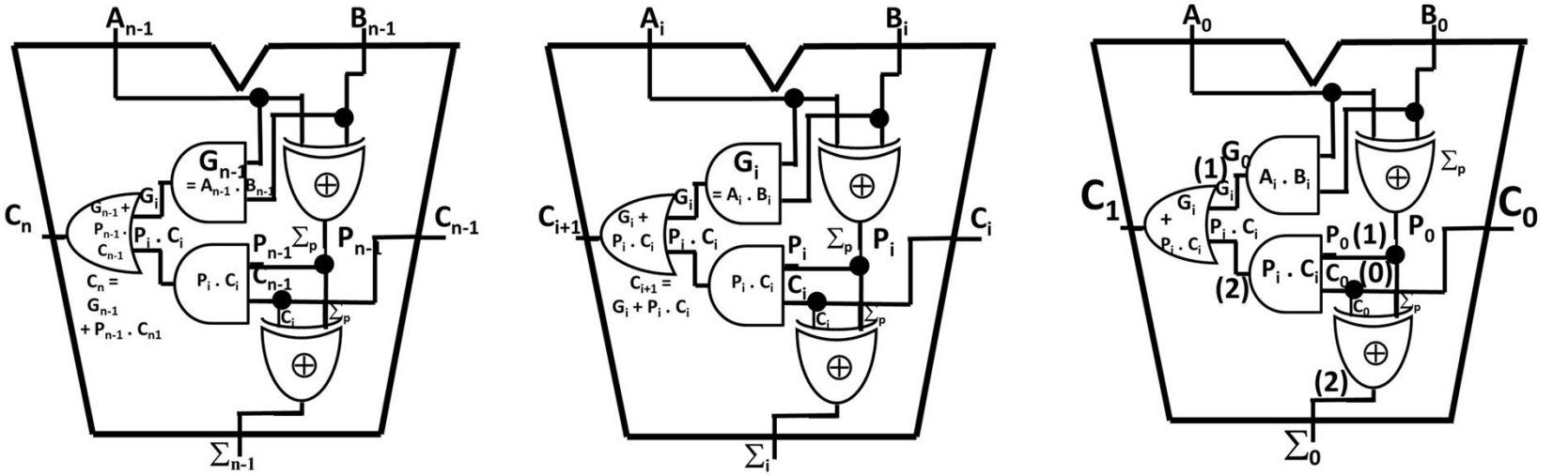
Ripple Adder – Generalização para n bits – Cálculo t_p – **Fatia n-1**



$$t_{p:\sum n-1} = (2.n).t_p$$

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

Ripple Adder – Módulo de n bits – Cálculo t_p



$t_{p:\sum n-1} = (2.n).t_p$

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

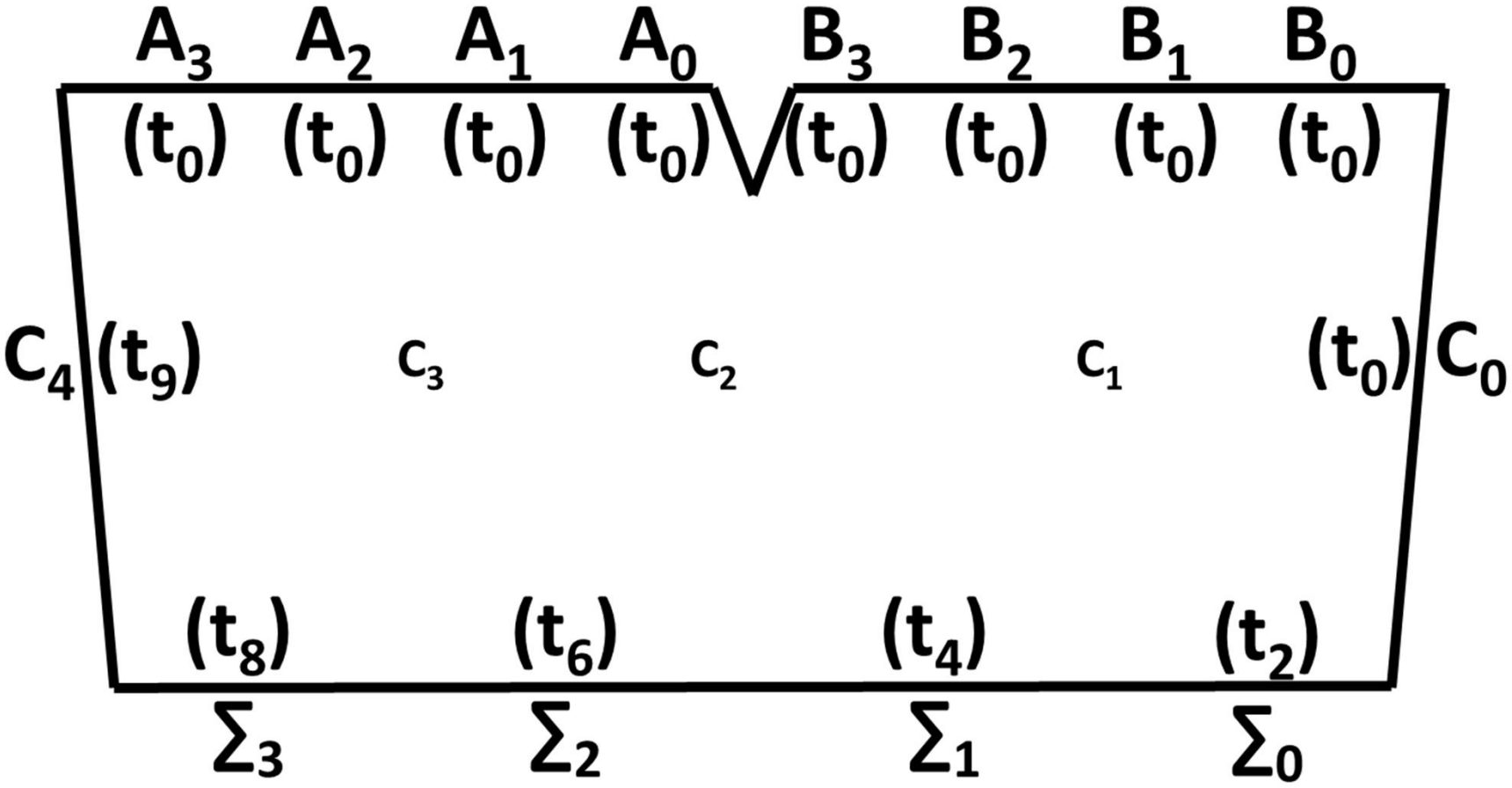
$t_{p:\sum i} = máx[t_{p:\sum pi}; t_{p:Ci}] + 1.t_p$

$t_{p:Ci+1} = máx[t_{p:\sum pi}; t_{p:Ci}] + 2.t_p$

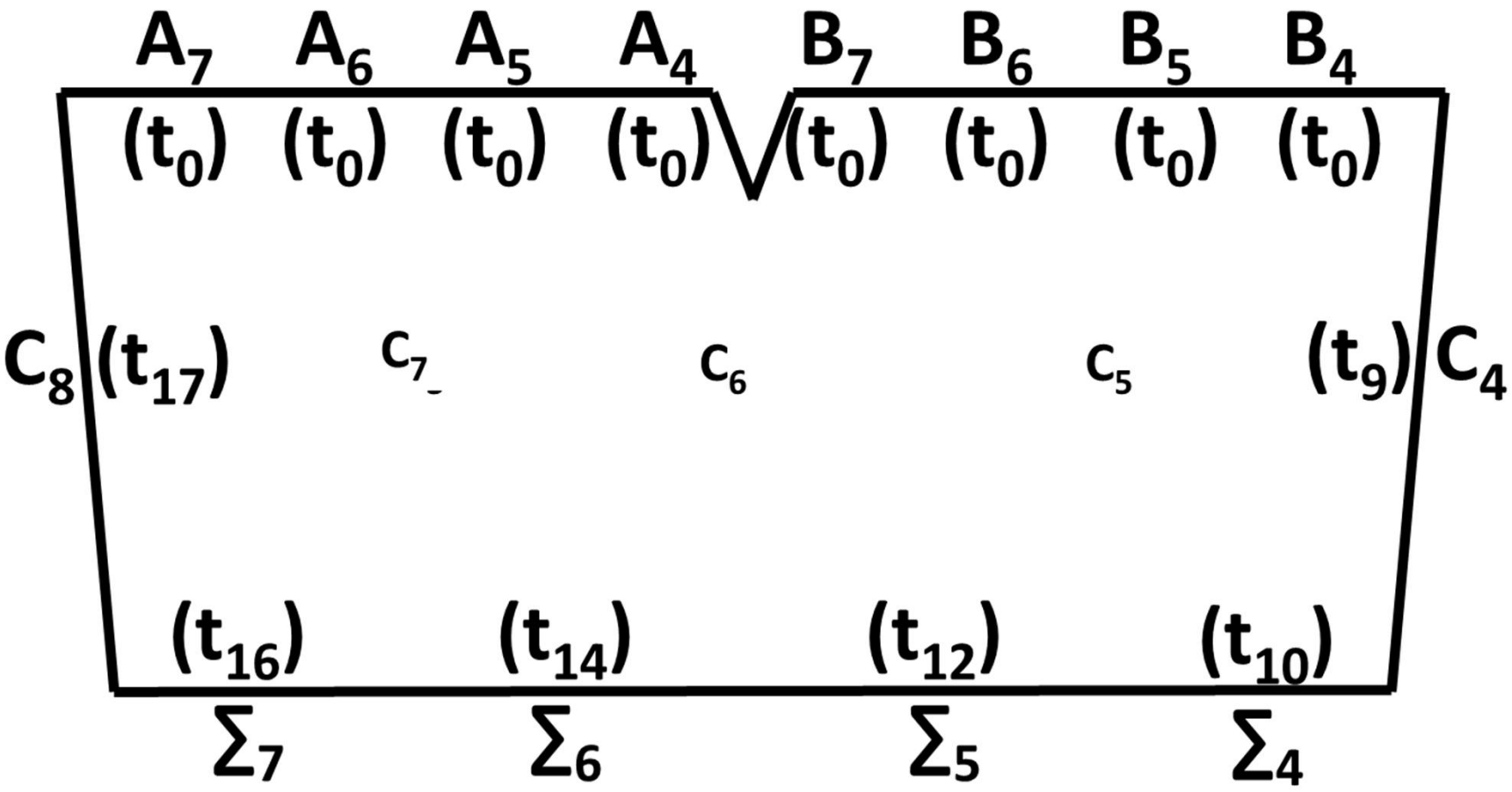
$t_{p:\sum 0} = t_{p:\sum p0} + 1.t_p = 2.t_p$

$t_{p:C1} = t_{p:\sum p0} + 2.t_p = 3.t_p$

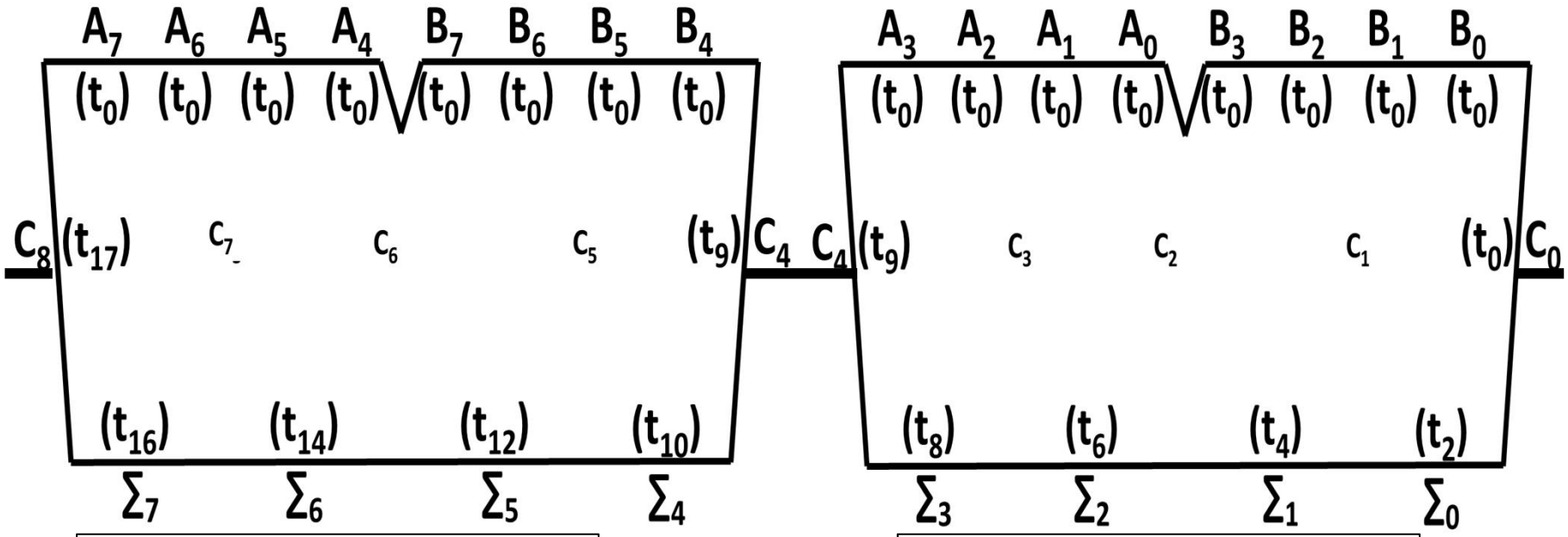
Ripple Adder – Módulo [1] de 4 bits – Cálculo de t_p



Ripple Adder – Módulo [2] de 4 bits – Cálculo de t_p



Ripple Adder – Módulos [1] + [2] – 8 bits – Cálculo de t_p



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

$$t_{p:C8} = (2.8+1).t_p$$

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

$$t_{p:C4} = (2.4+1).t_p$$

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

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

A pergunta que não quer calar

- Se o tempo para obtenção de C_4 no módulo [1] é :

$$\mathbf{t_{p:C4} = (2.n+1).t_p}$$

- Então por que o tempo para obtenção de C_8 no módulo [2] não é ???????

$$\mathbf{t_{p:C8} = (4.n+2).t_p}$$

A pergunta que não quer calar

- Para obter a resposta, faça como Jimmy Neutron ...

.... Pense!! ....

.... Pense!! ....

.... Pense!! ....