

Microeletrônica

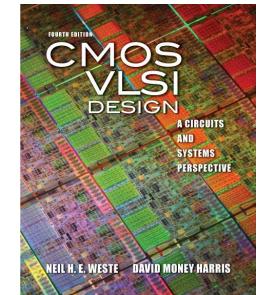
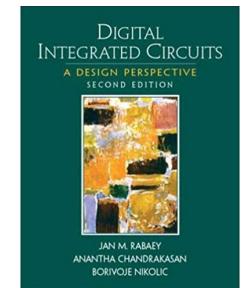
Aula #10 → Multiplicação e Divisão: conceitos básicos

□ Professor: Fernando Gehm Moraes

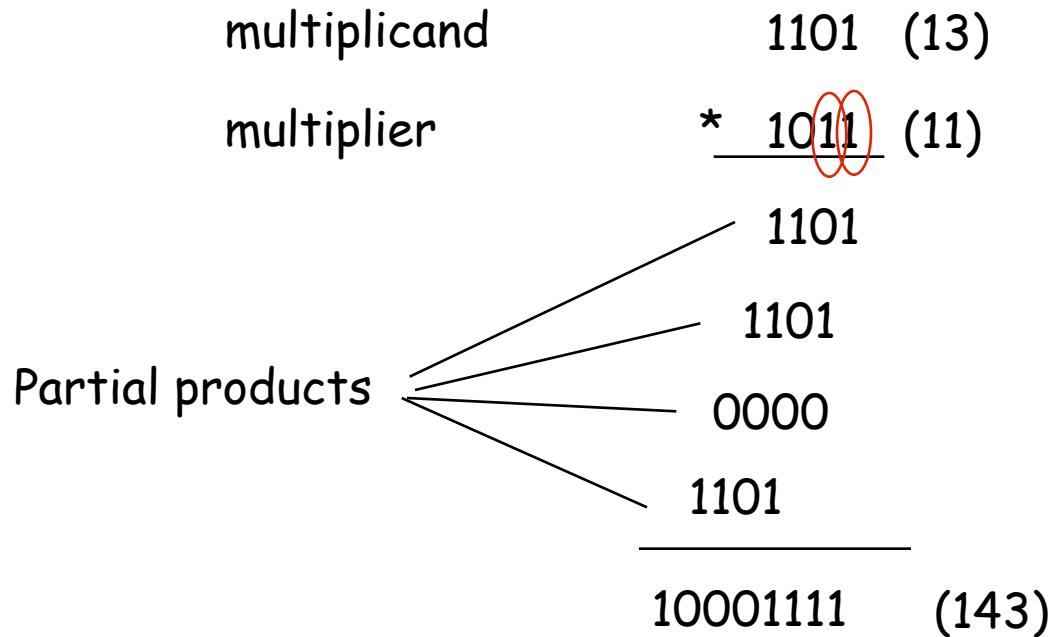
□ Livro texto:

Digital Integrated Circuits a Design Perspective - Rabaey

C MOS VLSI Design - Weste

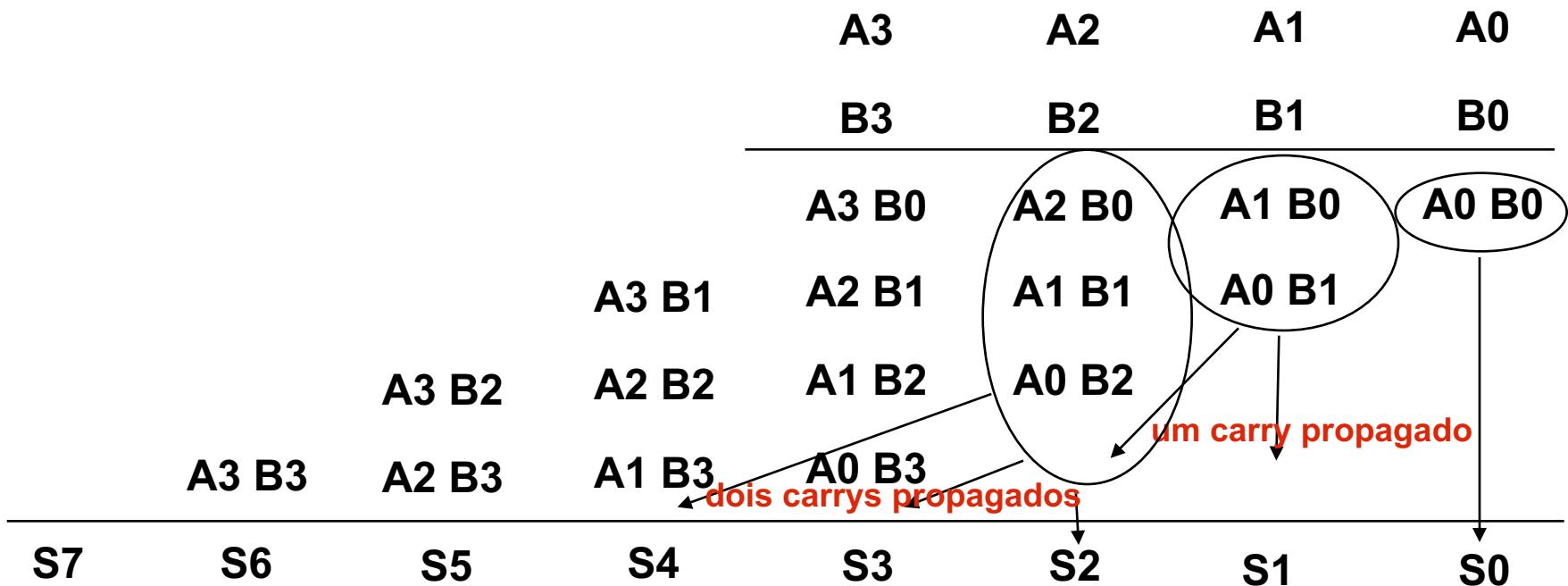


Conceitos básicos para multiplicação

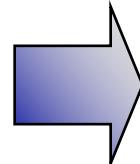


- product of 2 n-bit numbers is an 2n-bit number
 - sum of n n-bit partial products
- unsigned

Multiplicação

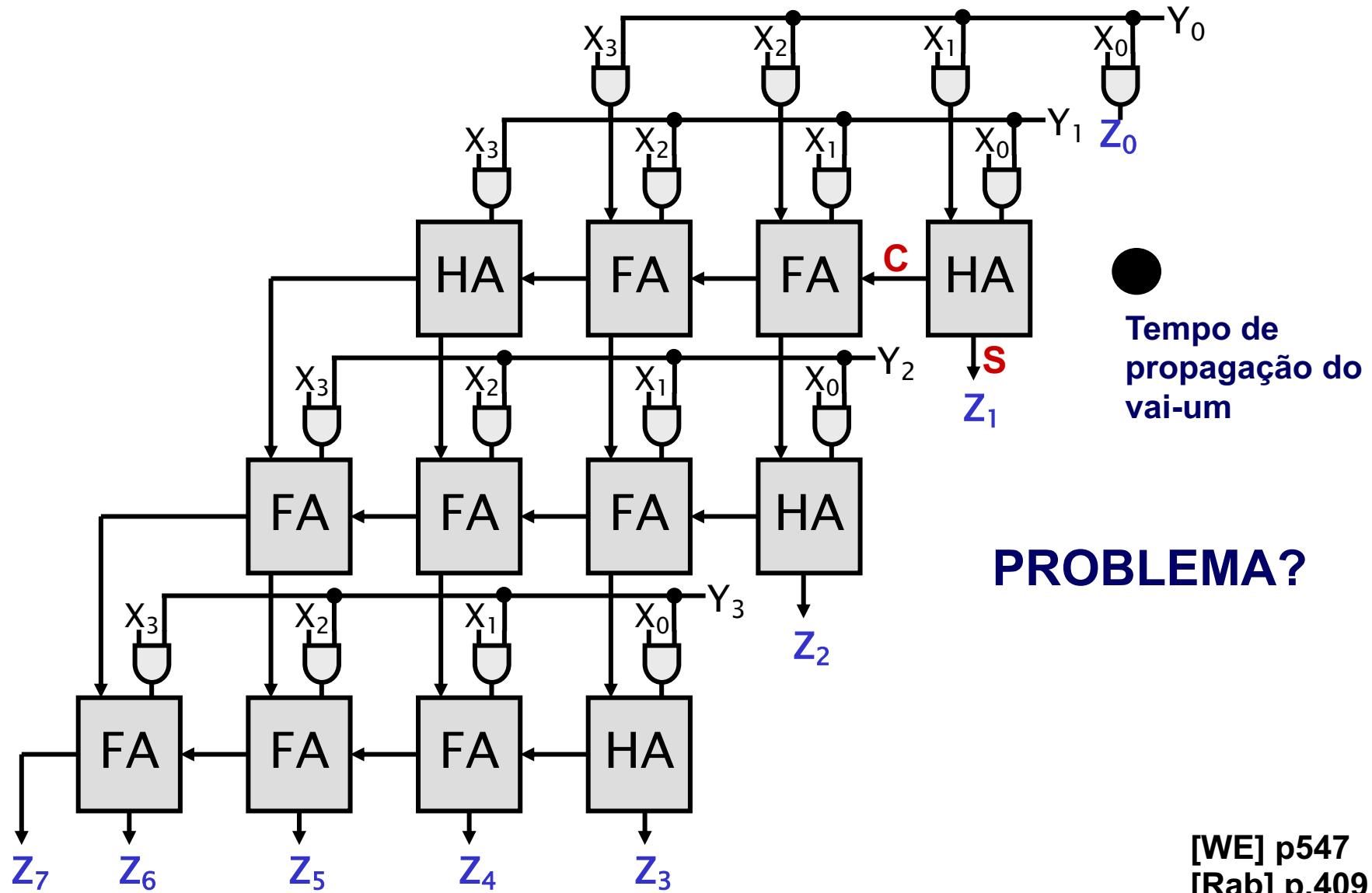


Quanto maior o número de produtos parciais a somar maior o número de bits de vai-um gerados



PROBLEMA

Multiplicador “array”



Carry-save Addition

- Speeding up multiplication is a matter of speeding up the summing of the partial products.
- “Carry-save” addition can help.
- Carry-save addition passes (saves) the carries to the output, rather than propagating them.

- Example: sum three numbers,
 $3_{10} = 0011$, $2_{10} = 0010$, $3_{10} = 0011$

$$\begin{array}{r} 3_{10} \quad 0011 \\ + 2_{10} \quad 0010 \\ \hline c \quad 01\boxed{0}0 \\ s \quad 000\boxed{1} \end{array} \quad \left. \begin{array}{l} = 4_{10} \\ = 1_{10} \end{array} \right\} \text{carry-save add}$$

$$\begin{array}{r} 3_{10} \quad 0011 \\ + 2_{10} \quad 0010 \\ \hline c \quad 0010 \\ s \quad 0110 \\ \hline 1000 \end{array} \quad \left. \begin{array}{l} = 2_{10} \\ = 6_{10} \\ = 8_{10} \end{array} \right\} \text{carry-propagate add}$$

- In general, carry-save addition takes in 3 numbers and produces 2.
- Whereas, carry-propagate takes 2 and produces 1.
- With this technique, we can avoid carry propagation until final addition

Carry-save Addition

□ Fazer a soma $7 + 3 + 4$

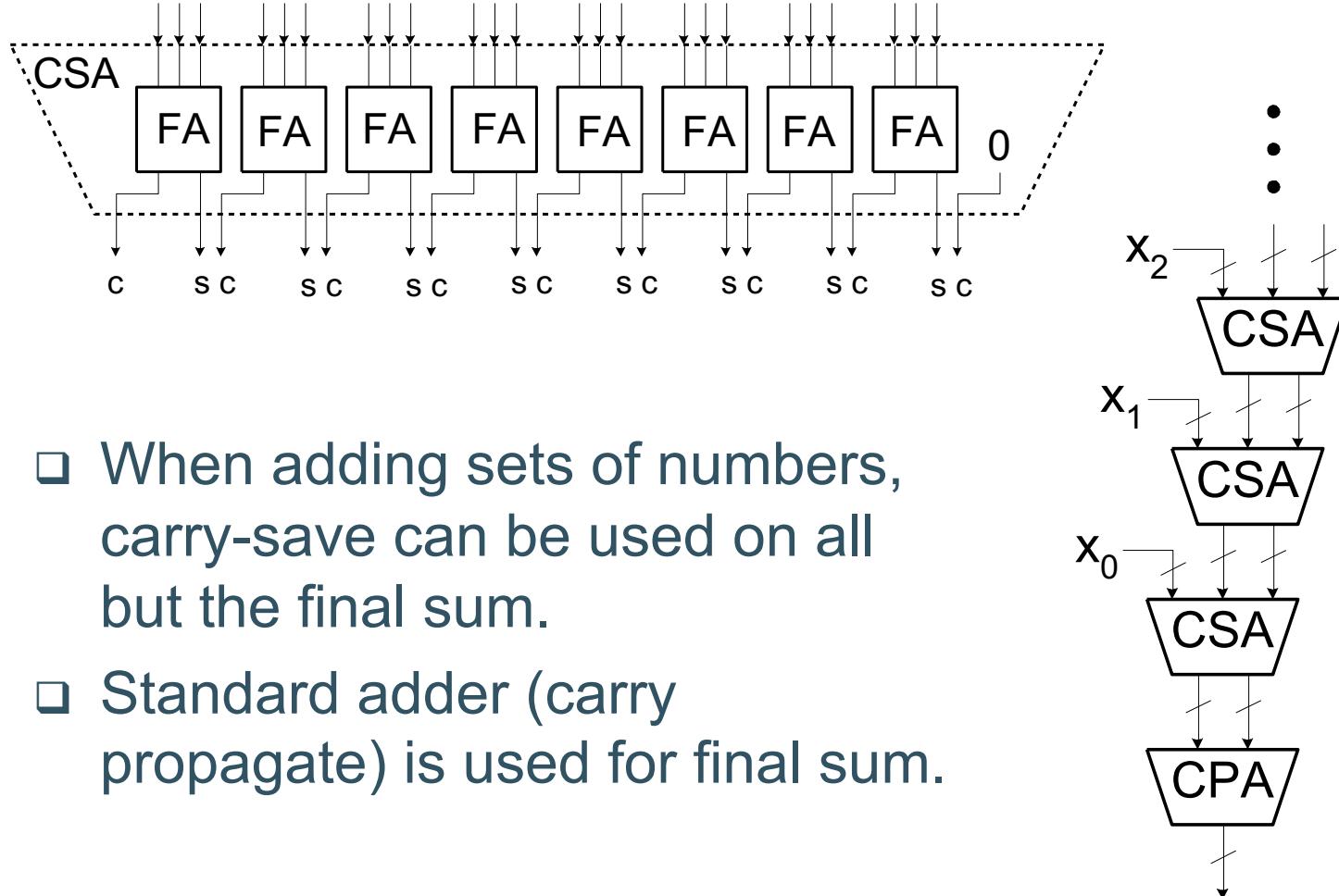
$$\begin{array}{r} 0111 \\ 0011 \\ \hline 0 \\ \hline 0100 \\ 0 \\ \hline \end{array}$$

carry-save add {

7
3
carry
soma } carry-save add

4
carry
soma
resultado

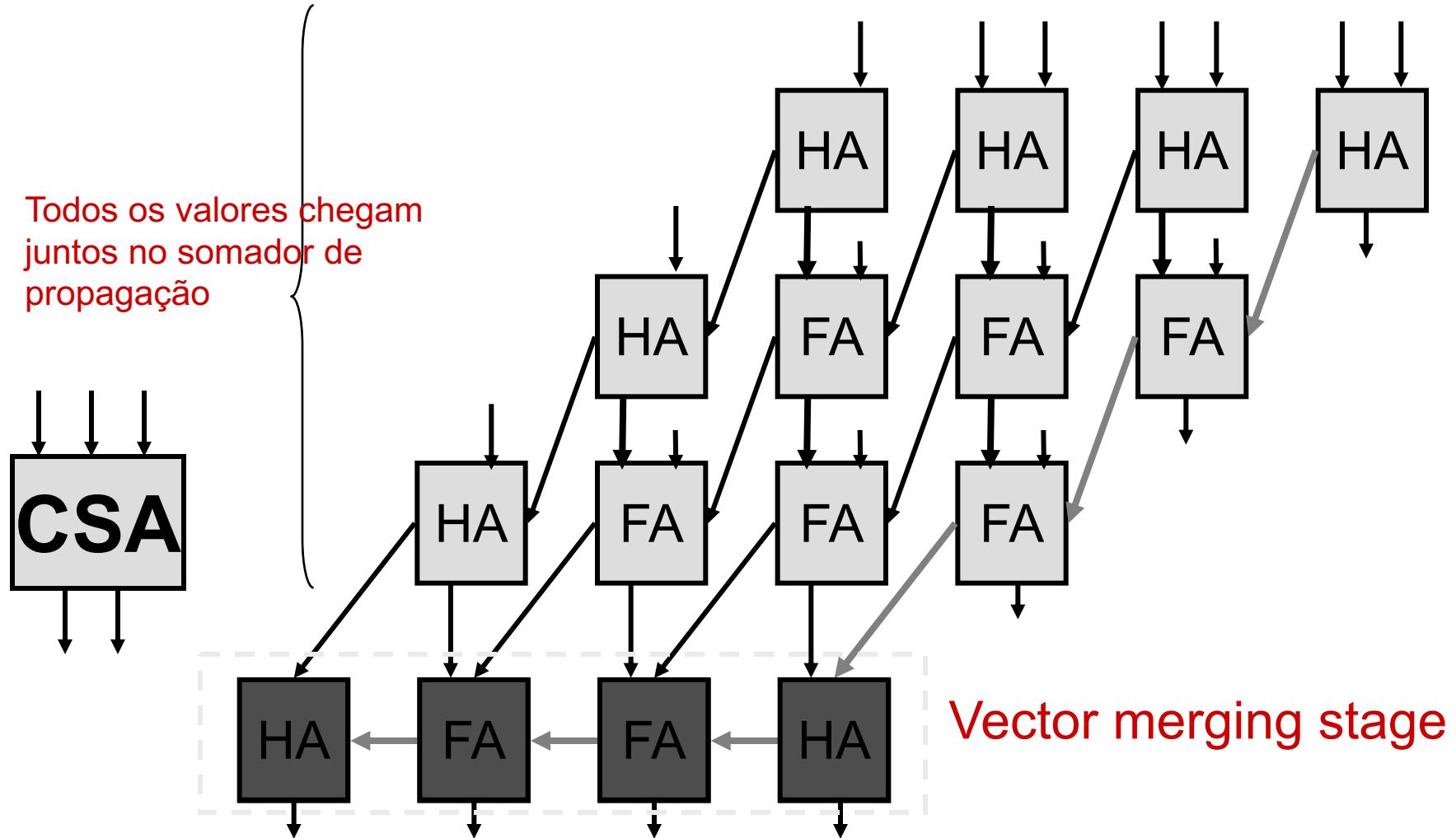
Carry-save Circuits



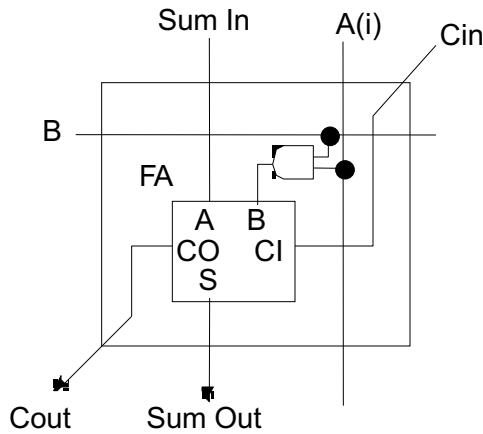
- When adding sets of numbers, carry-save can be used on all but the final sum.
- Standard adder (carry propagate) is used for final sum.

Carry-Save Adder: structure

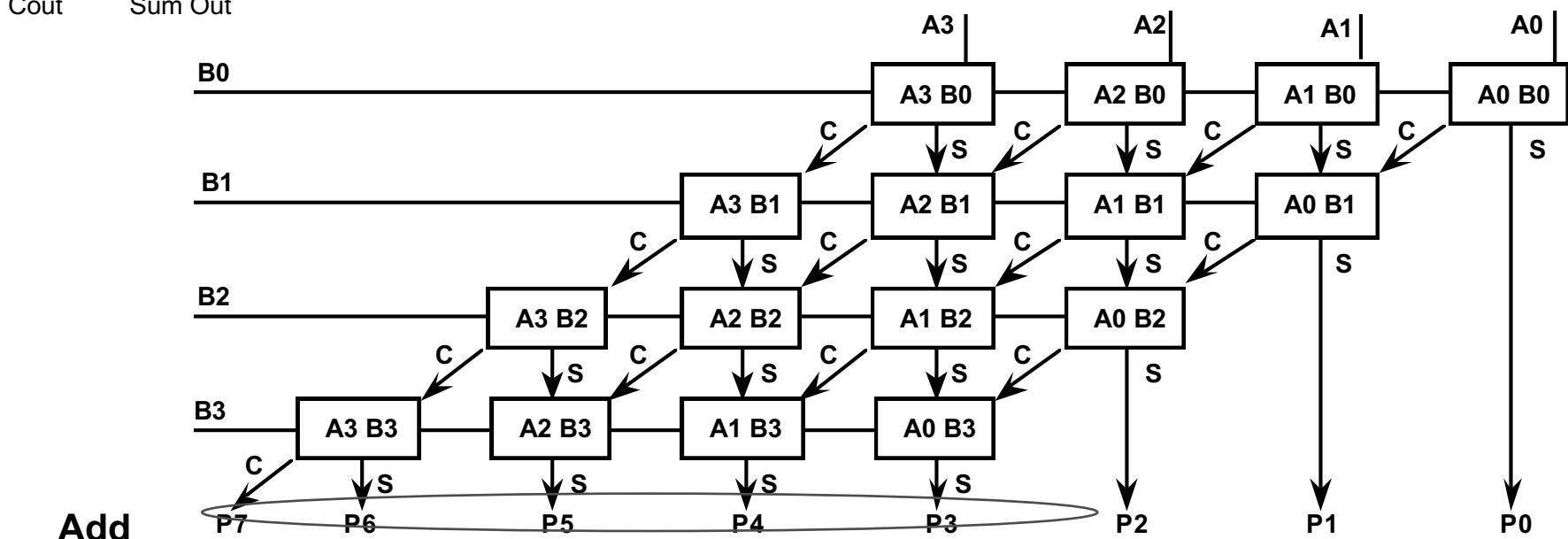
- Postpone the “carry propagation” operation to the last stage



Implementação do multiplicador



Building block: full adder + and



CPA

4 x 4 array of building blocks

architecture Mult_CSAdder of Mult_CSAdder is

```
type mat4x4 is array(0 to 3) of std_logic_vector(3 downto 0);
```

```
signal soma : mat4x4;
```

```
signal vaium : mat4x4;
```

```
signal andab : mat4x4;
```

```
signal co: std_logic_vector(3 downto 0);
```

```
begin
```

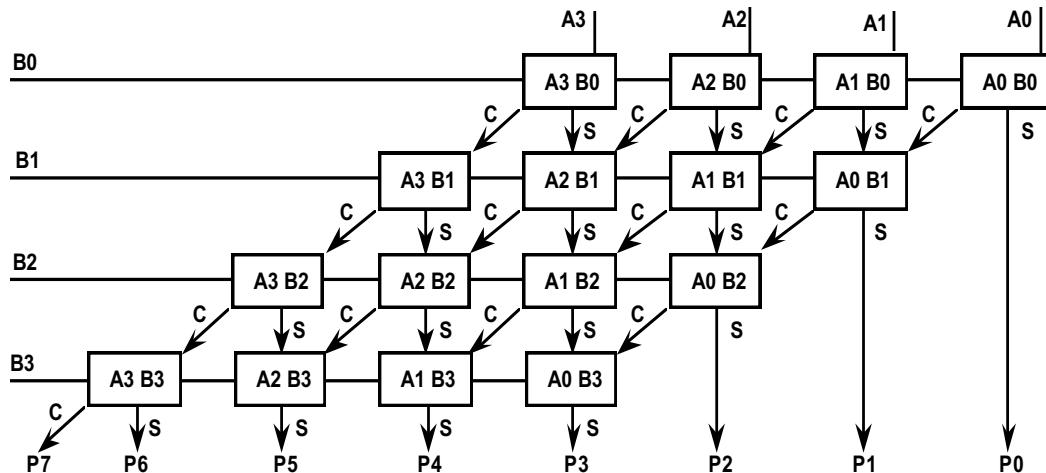
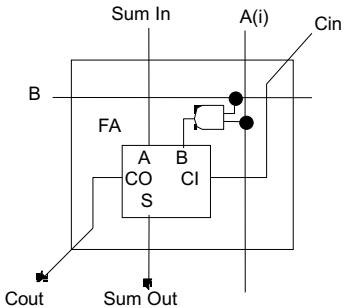
--- primeira camada de geração de produtos parciais (substitui os FA por uma porta AND) ---

```
bits0: for j in 0 to 3 generate
```

```
    soma(0)(j) <= a(j) and b(0);
```

```
    vaium(0)(j)<= '0';
```

```
end generate bits0;
```



linhas: for i in 1 to 3 generate
 bits: for j in 0 to 3 generate

andab(i)(j) <= a(j) and b(i);

g1: if j<3 generate

Adder1: entity **FullAdder** port map(a=>andab(i)(j), b=>vaium(i-1)(j),
 c=>soma(i-1)(j+1), S=>soma(i)(j), Cout=>vaium(i)(j));

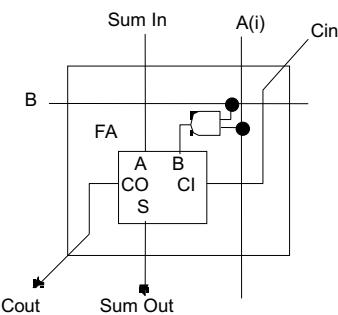
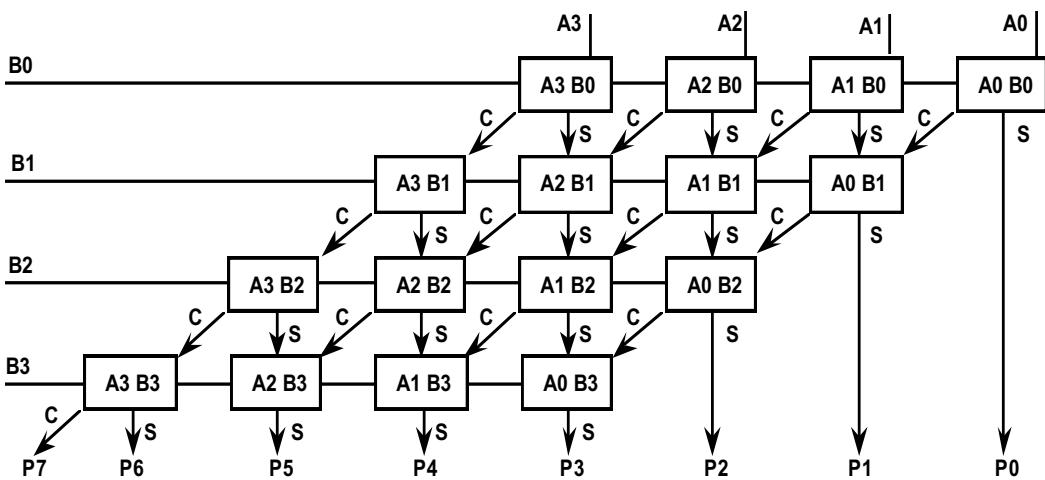
end generate;

g2: if j=3 generate

Adder2: entity **FullAdder** port map(a=>andab(i)(j), b=>vaium(i-1)(j), **c=>'0'**,
 S=>soma(i)(j), Cout=>vaium(i)(j));

end generate;

end generate bits;
 end generate linhas;



-- primeiros bits da multiplicação são obtidos diretamente dos carry-save adders---

P(3 downto 0) <= soma(3)(0) & soma(2)(0) & soma(1)(0) & soma(0)(0);

--- geração dos bits restantes do produto, a partir dos resultados dos carry-save adders

ad1: entity FullAdder PORT MAP

(a=>soma(3)(1), b=>vaium(3)(0), c=>'0', S=>P(4), Cout=>co(0));

fa_last: for j in 1 to 3 generate

g3: if j<3 generate

entity FullAdder PORT MAP(soma(3)(j+1), vaium(3)(j), co(j-1), S=>P(4+j), Cout=>co(j));

end generate;

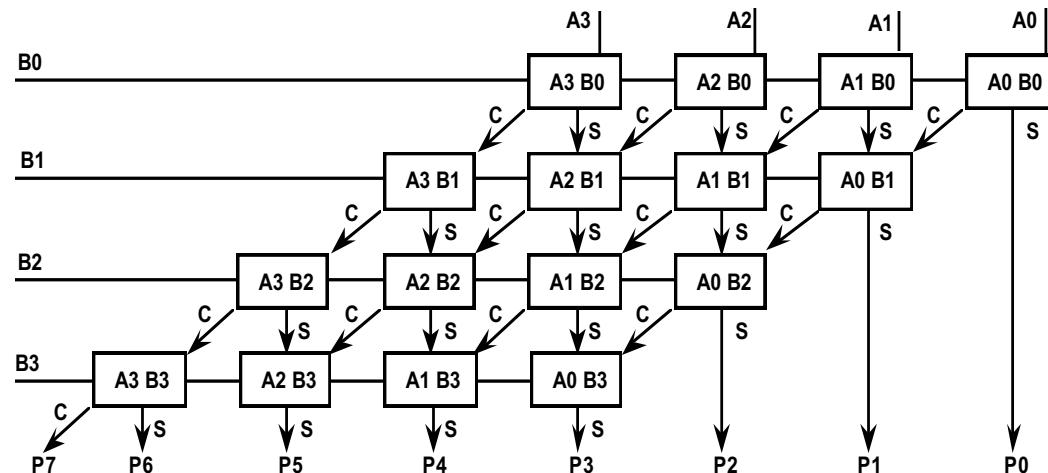
g4: if j=3 generate

entity FullAdder PORT MAP('0', vaium(3)(j), co(j-1), S=>P(4+j), Cout=>co(j));

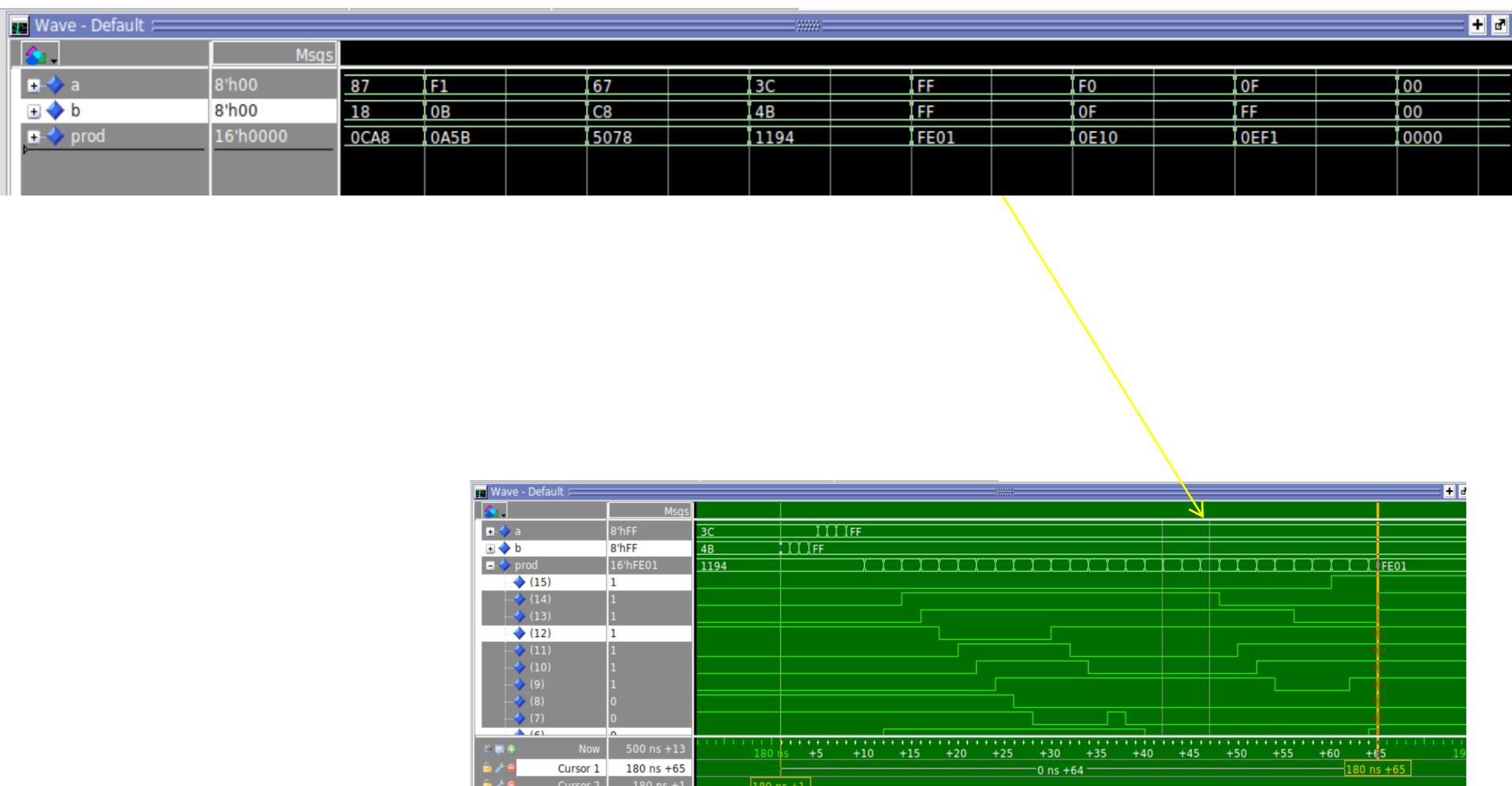
end generate;

end generate fa_last;

end Mult_CSAdder;



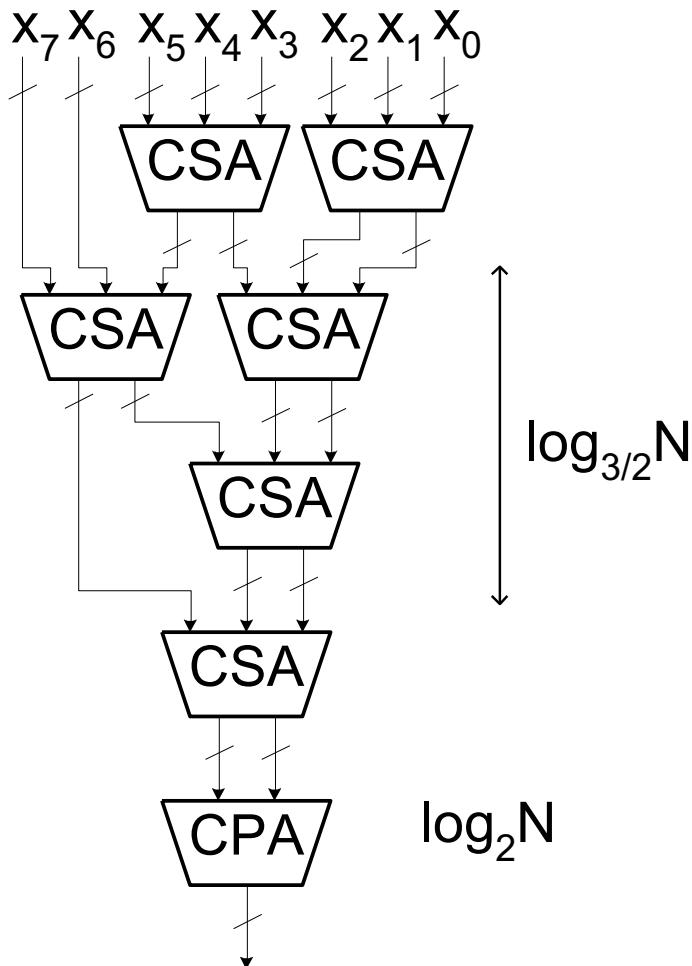
Simulação e atrasos



Carry-save Addition

CSA is associative and commutative. For example:

$$((X_0 + X_1) + X_2) + X_3 = (X_0 + X_1) + (X_2 + X_3)$$



- A balanced tree can be used to reduce the logic delay.
- This structure is the basis of the **Wallace Tree Multiplier**.
- Partial products are summed with the CSA tree. Fast CPA (ex: CLA) is used for final sum.
- Multiplier delay $\propto \log_{3/2} N + \log_2 N$

O somador carry save, utilizado nos multiplicadores, serve para realizar a soma quando há várias parcelas a serem adicionadas.

Dados os seguintes valores:

$$A = 001101 \text{ (} 13_{10} \text{)}$$

$$B = 000101 \text{ (} 5_{10} \text{)}$$

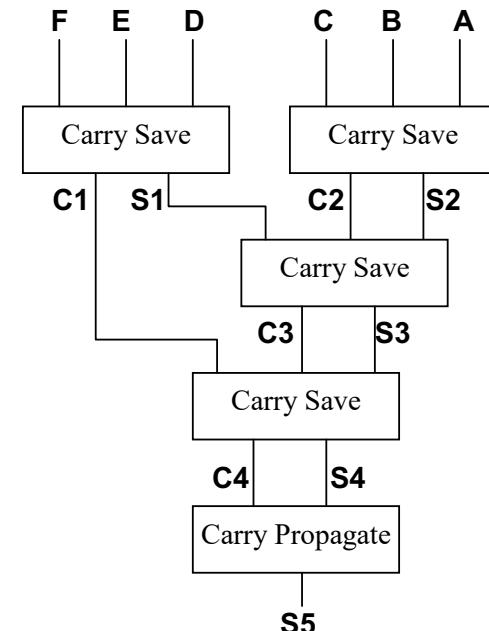
$$C = 001011 \text{ (} 11_{10} \text{)}$$

$$D = 001010 \text{ (} 10_{10} \text{)}$$

$$E = 000111 \text{ (} 7_{10} \text{)}$$

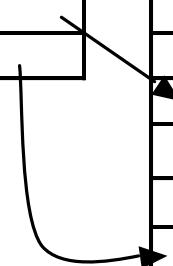
$$F = 001100 \text{ (} 12_{10} \text{)}$$

Mostre a obtenção e os valores de $(C1/S1)$, $(C2/S2)$, $(C3/S3)$, $(C4/S4)$ e a soma final ($S5$).



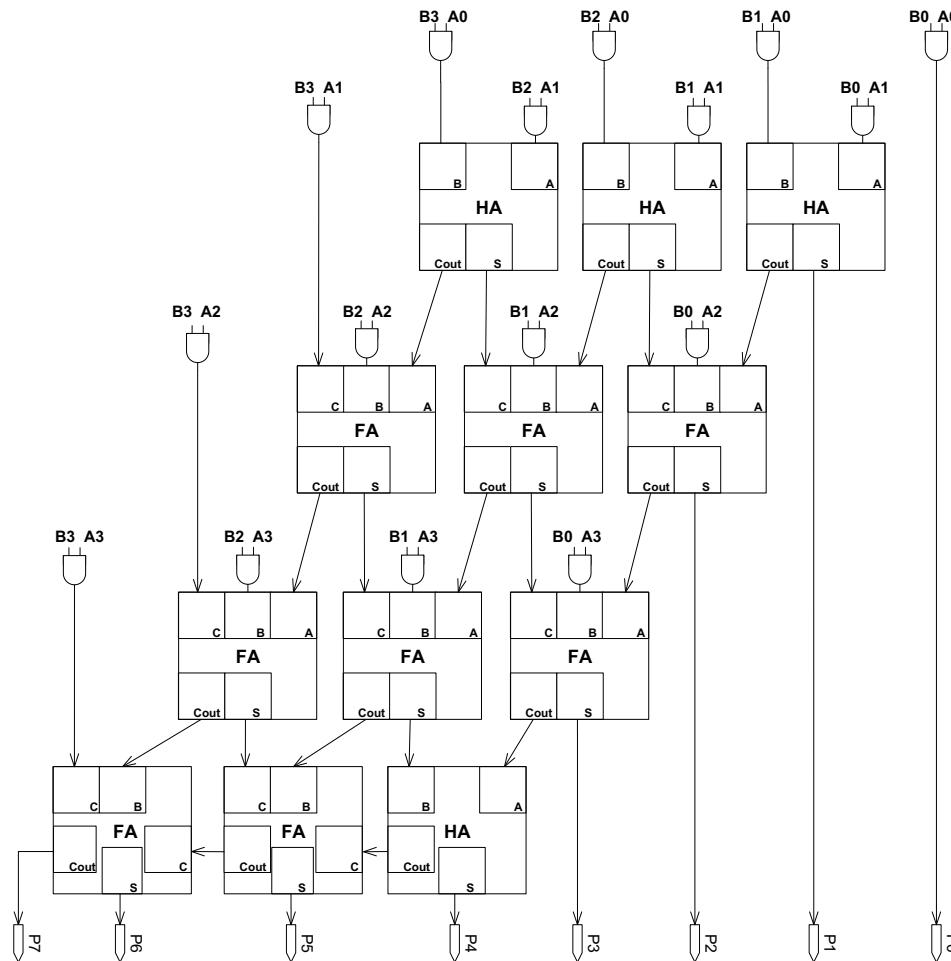
D	0	0	1	0	1	0
E	0	0	0	1	1	1
F	0	0	1	1	0	0
S1						
C1						

A	0	0	1	1	0	1
B	0	0	0	1	0	1
C	0	0	1	0	1	1
S2						
C2						
S1						
S3						
C3						
C1						
S4						
C4						
Soma						



Multiplicação. Apresenta-se abaixo o diagrama lógico de um multiplicador de 4 bits.

- Explique como o carry é propagado no interior do circuito. Qual a vantagem deste esquema de propagação de carry?
- Considere: $A=1011$ e $B=1101$. Desenhe sobre o circuito os valores booleanos correspondentes, verificando se o valor obtido pela multiplicação é o correto.

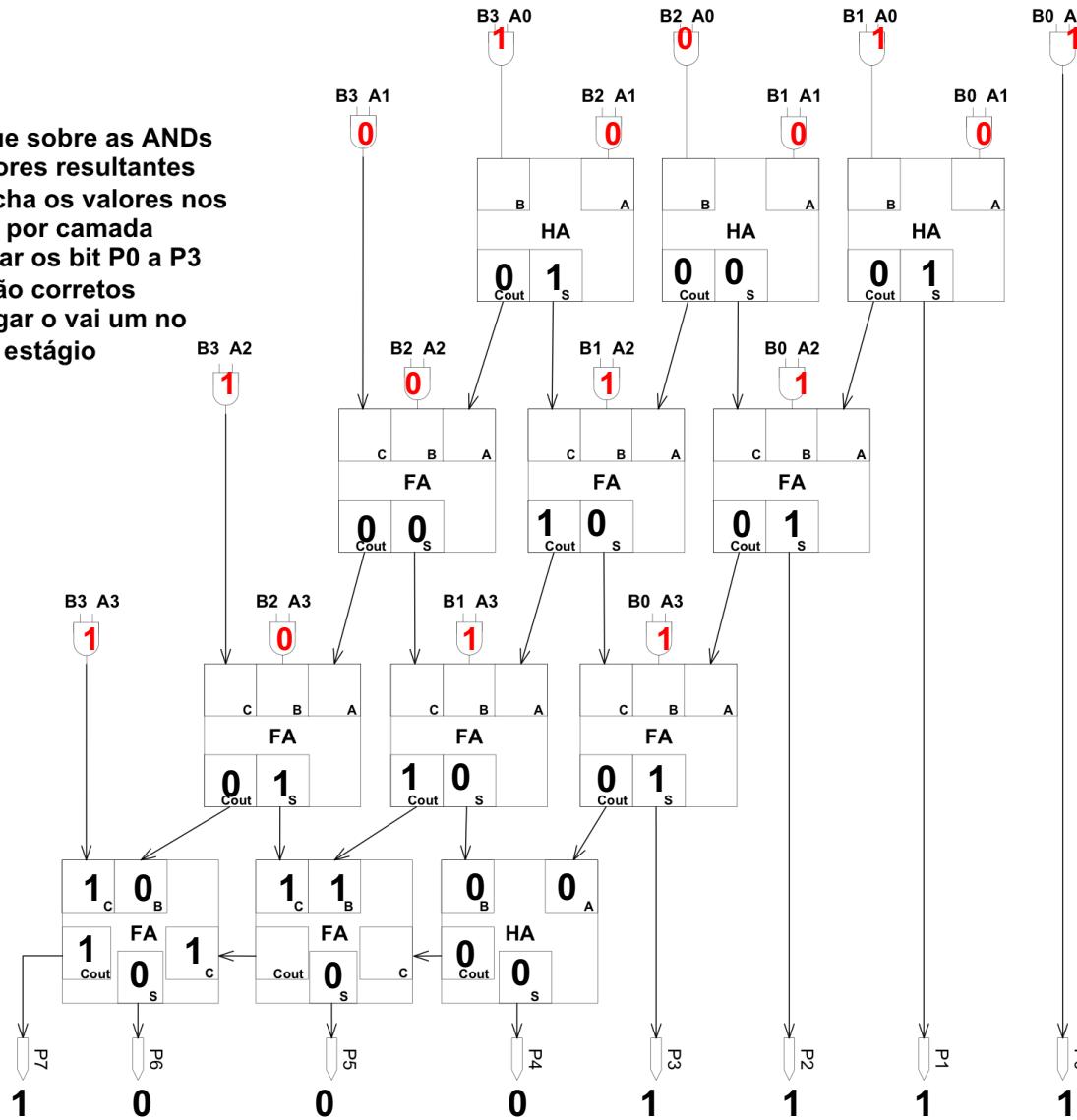


A=1101
B=1011

Multiplicador com carry-save

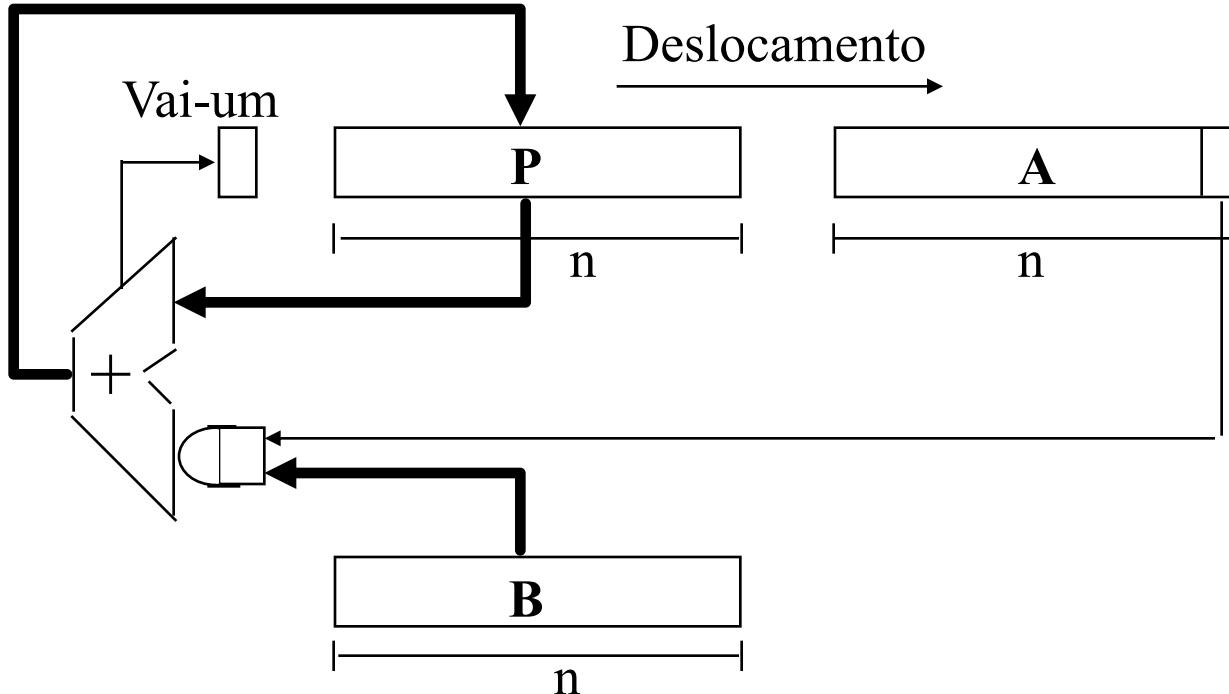
DICA:

- 1- coloque sobre as ANDs os valores resultantes
- 2- Preencha os valores nos HA/FA por camada
- 3- Verificar os bit P0 a P3 se estão corretos
- 4- Propagar o vai um no último estágio



Multiplicação serial

- Solução natural para a^*b : somas sucessivas n passos



- Inicialmente, $P=0$, $A=a$, $B=b$. Cada passo, duas partes:
- soma carregada em P ;
 - P & A deslocado um bit para a direita.

Multiplicação A*B

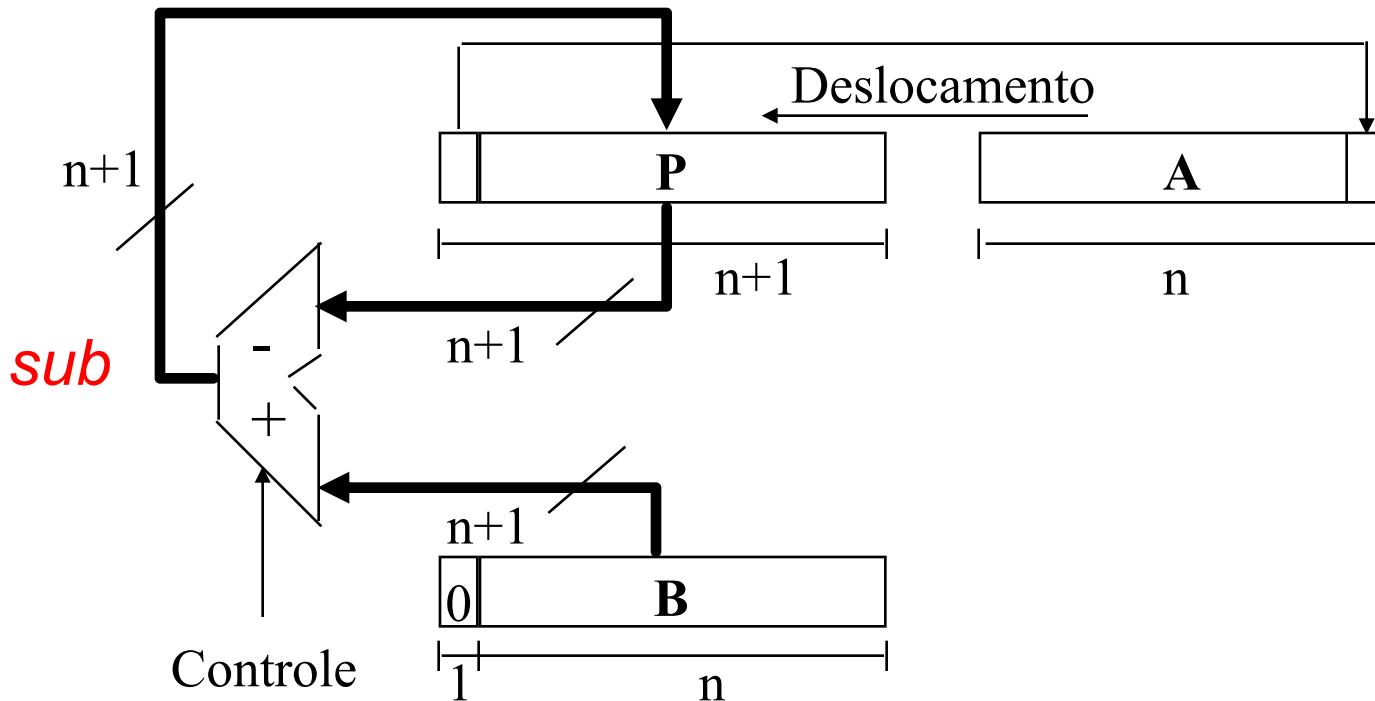
$$\begin{array}{l} A = 11011 \quad (27) \\ B = 00101 \quad (5) \end{array}$$

$$135 \rightarrow 100\ 00111$$

passo	P						A				
	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	1	0	1	1	1	0	1	1
	0	0	0	0	1	0	1	1	1	0	1
2	0	0	0	1	1	1	1	1	0	1	
	0	0	0	0	1	1	1	1	1	1	0
3	0	0	0	0	1	1	1	1	1	1	0
	0	0	0	0	0	1	1	1	1	1	1
4	0	0	0	1	1	0	1	1	1	1	1
	0	0	0	0	1	1	0	1	1	1	1
5	0	0	1	0	0	0	0	1	1	1	1
	0	0	0	1	0	0	0	0	1	1	1

Divisão serial

- Solução para a/b : subtrações sucessivas, n passos



- Algoritmo:
 - 1) desloca P&A p/ esq 1 bit; $\text{sub} \leftarrow P - B$;
 - 2) if ($\text{sub} < 0$), $A_0 = 0$ else $\{ A_0 = 1; P \leftarrow \text{sub} \}$

Divisão A/B

$$\begin{array}{ll} A = 11011 & (27) \\ B = 00101 & (5) \end{array}$$

- 1) desloca P&A p/ esq 1 bit; sub \leftarrow P-B;
- 2) if (sub<0), A0=0 else { A0 =1; P \leftarrow sub}

passo	P (conterá o resto)						A (conterá a divisão)				
	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2											
3											
4											
5											

Divisão A/B

$$\begin{array}{l} A = 11011 \text{ (27)} \\ B = 00101 \text{ (5)} \end{array}$$

- 1) desloca P&A p/ esq 1 bit; sub \leftarrow P-B;
- 2) if (sub<0), A0=0 else { A0 =1; P \leftarrow sub}

	P (conterá o resto)						A (conterá a divisão)				
passo	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	0
	0	0	0	0	1	1	0	1	1	0	0
3											
4											
5											

Divisão A/B

$$A = 11011 \text{ (27)}$$

$$B = 00101 \text{ (5)}$$

- 1) desloca P&A p/ esq 1 bit; sub \leftarrow P-B;
- 2) if (sub<0), A0=0 else { A0 =1; P \leftarrow sub}

	P (conterá o resto)						A (conterá a divisão)				
passo	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	0
	0	0	0	0	1	1	0	1	1	0	0
3	0	0	0	1	1	0	1	1	0	0	0
	0	0	0	0	0	1	1	1	0	0	1
4											
5											

$$00110 - 00101 = 001$$

Divisão A/B

$$\begin{aligned} A &= 11011 \text{ (27)} \\ B &= 00101 \text{ (5)} \end{aligned}$$

- 1) desloca P&A p/ esq 1 bit; sub \leftarrow P-B;
 2) if (sub<0), A0=0 else { A0 =1; P \leftarrow sub}

	P (conterá o resto)						A (conterá a divisão)				
passo	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	0
	0	0	0	0	1	1	0	1	1	0	0
3	0	0	0	1	1	0	1	1	0	0	0
	0	0	0	0	0	1	1	1	0	0	1
4	0	0	0	0	1	1	1	0	0	1	0
	0	0	0	0	1	1	1	0	0	1	0
5											

Divisão A/B

$$\begin{aligned} A &= 11011 \text{ (27)} \\ B &= 00101 \text{ (5)} \end{aligned}$$

- 1) desloca P&A p/ esq 1 bit; sub \leftarrow P-B;
- 2) if (sub<0), A0=0 else { A0 =1; P \leftarrow sub}

	P (conterá o resto)						A (conterá a divisão)				
passo	0	0	0	0	0	0	1	1	0	1	1
1	0	0	0	0	0	1	1	0	1	1	0
	0	0	0	0	0	1	1	0	1	1	0
2	0	0	0	0	1	1	0	1	1	0	0
	0	0	0	0	1	1	0	1	1	0	0
3	0	0	0	1	1	0	1	1	0	0	0
	0	0	0	0	0	1	1	1	0	0	1
4	0	0	0	0	1	1	1	0	0	1	0
	0	0	0	0	1	1	1	0	0	1	0
5	0	0	0	1	1	1	0	0	1	0	0
	0	0	0	0	1	0	0	0	1	0	1

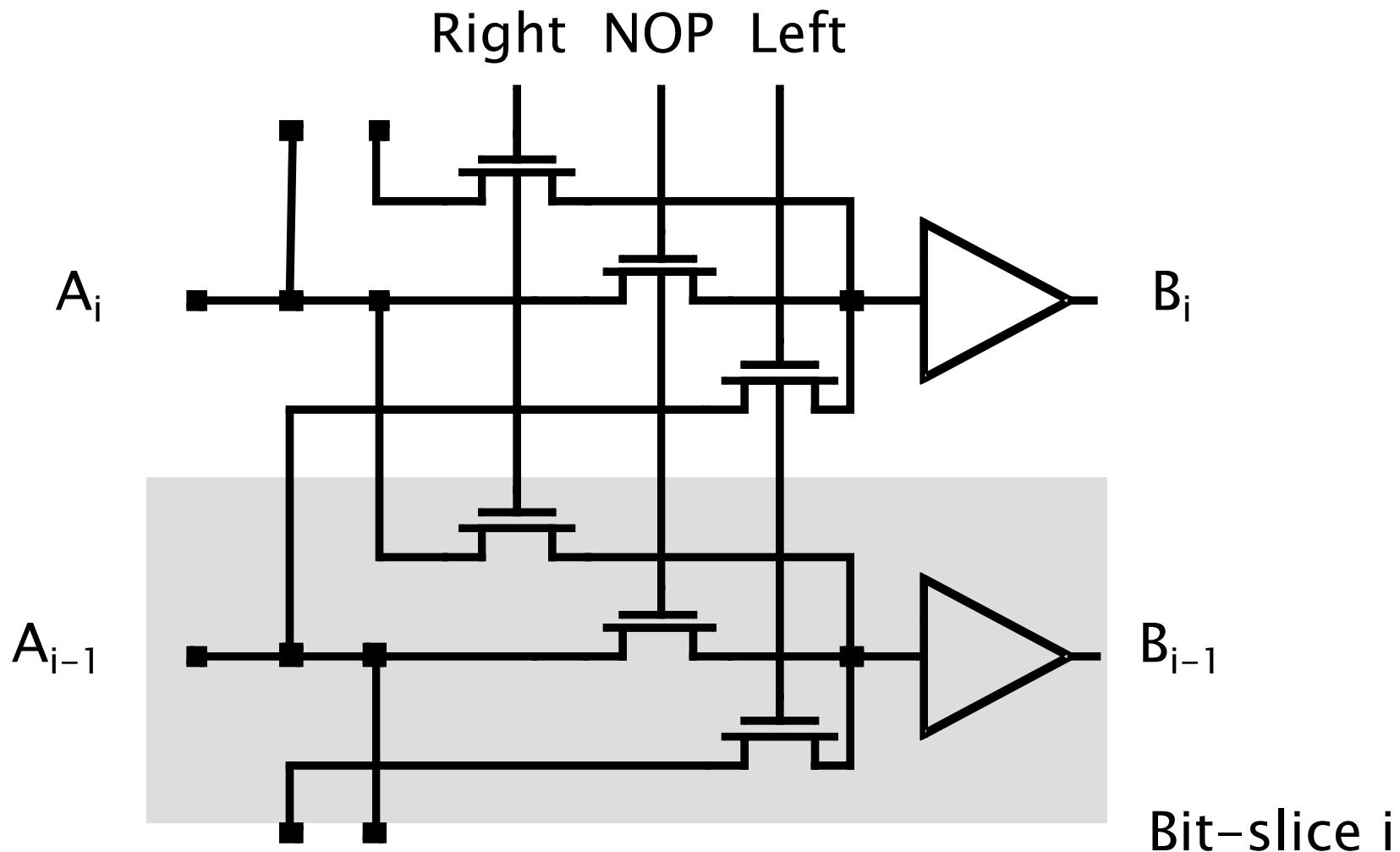
Resto = 2

resultado=5

Shift and Rotate Operations

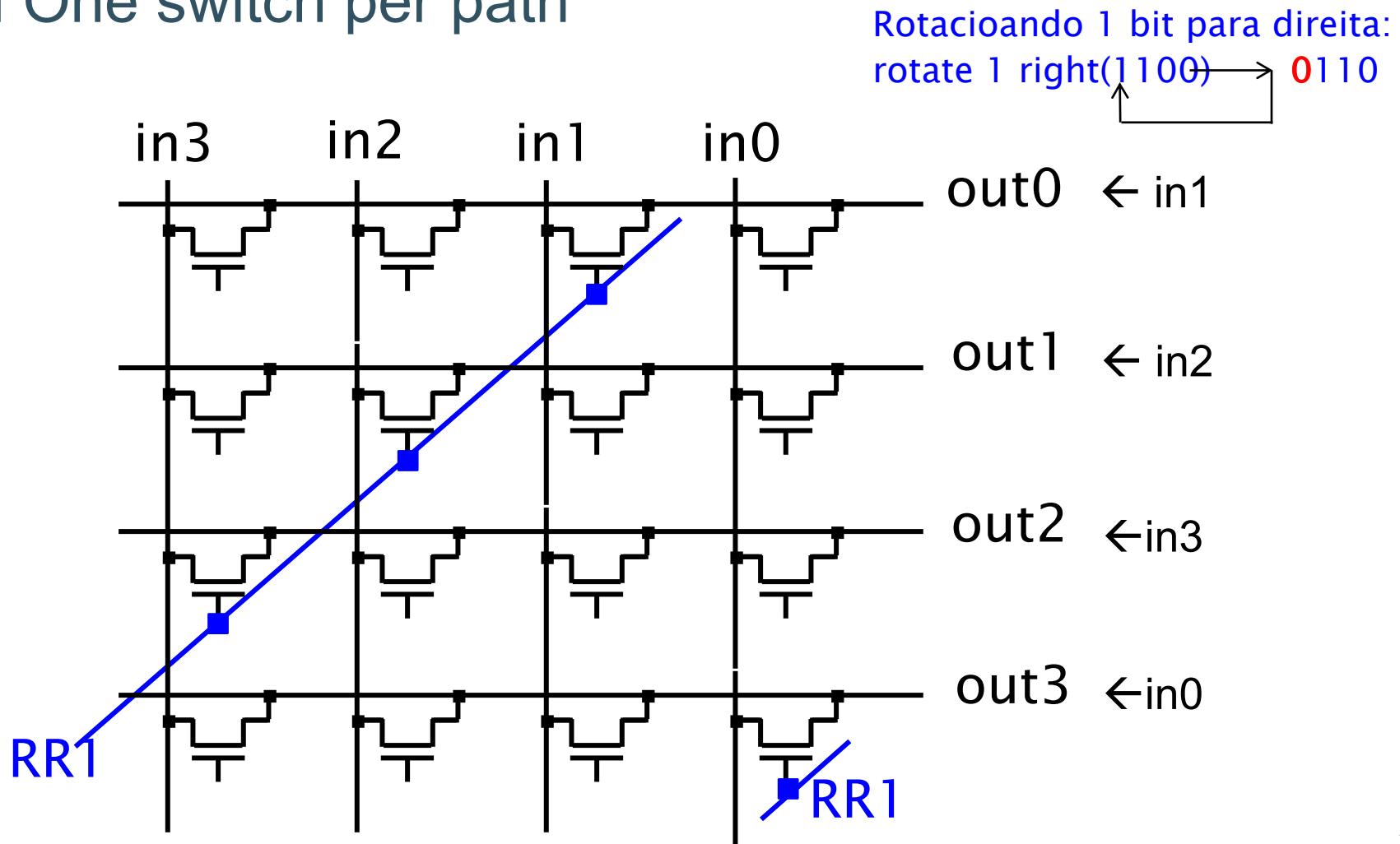
- Used in:
 - Microprocessors
 - Encryption algorithms
- If fixed shift, simply wire the inputs to the correct output positions
- Variable shift
 - One-bit shifter
 - Barrel shifter
 - Logarithmic shifter

One-bit Shifter



n-bit Shifter (barrel shifter)

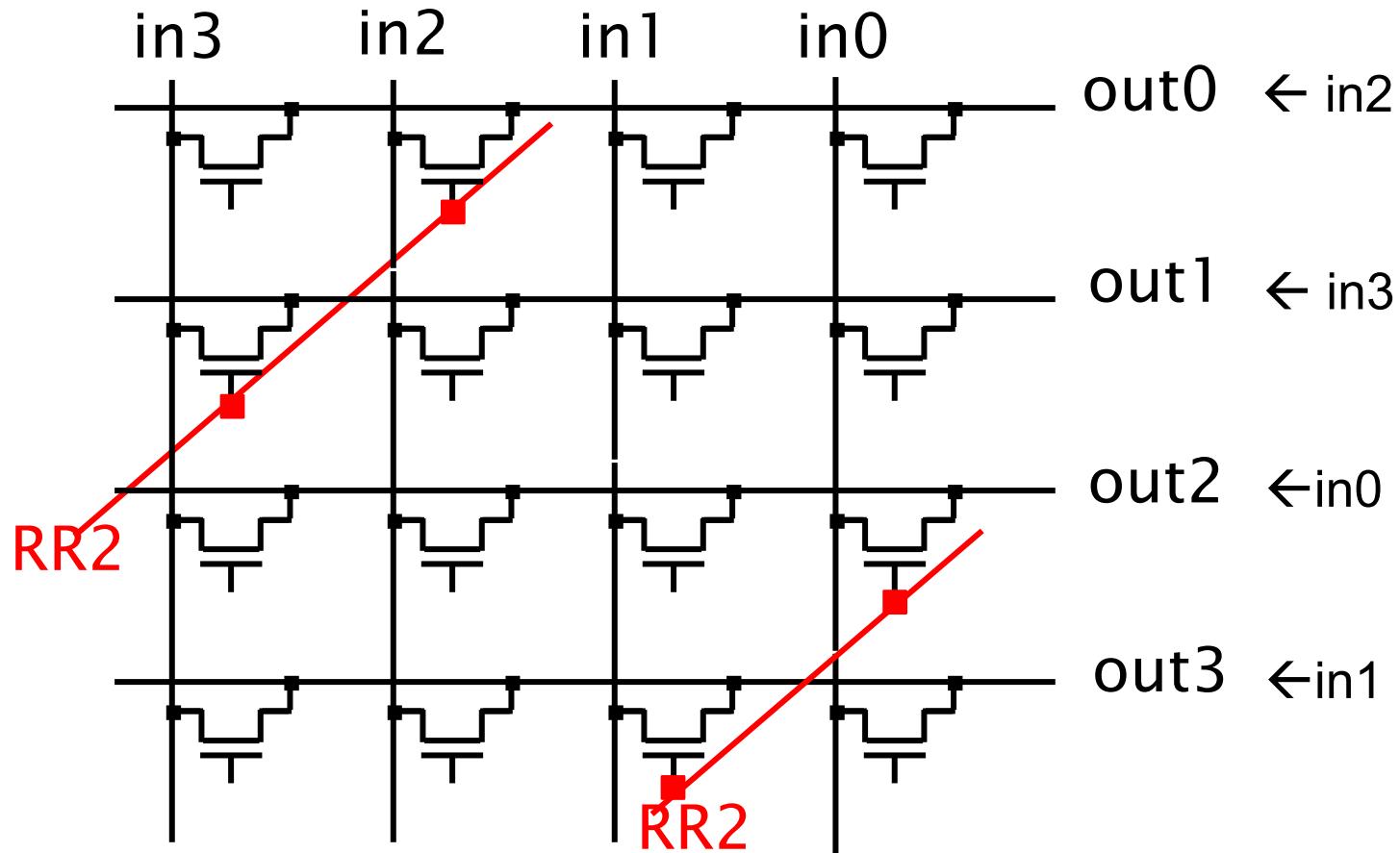
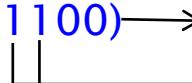
- Quadratic number of transistors
- One switch per path



n-bit Shifter (barrel shifter)

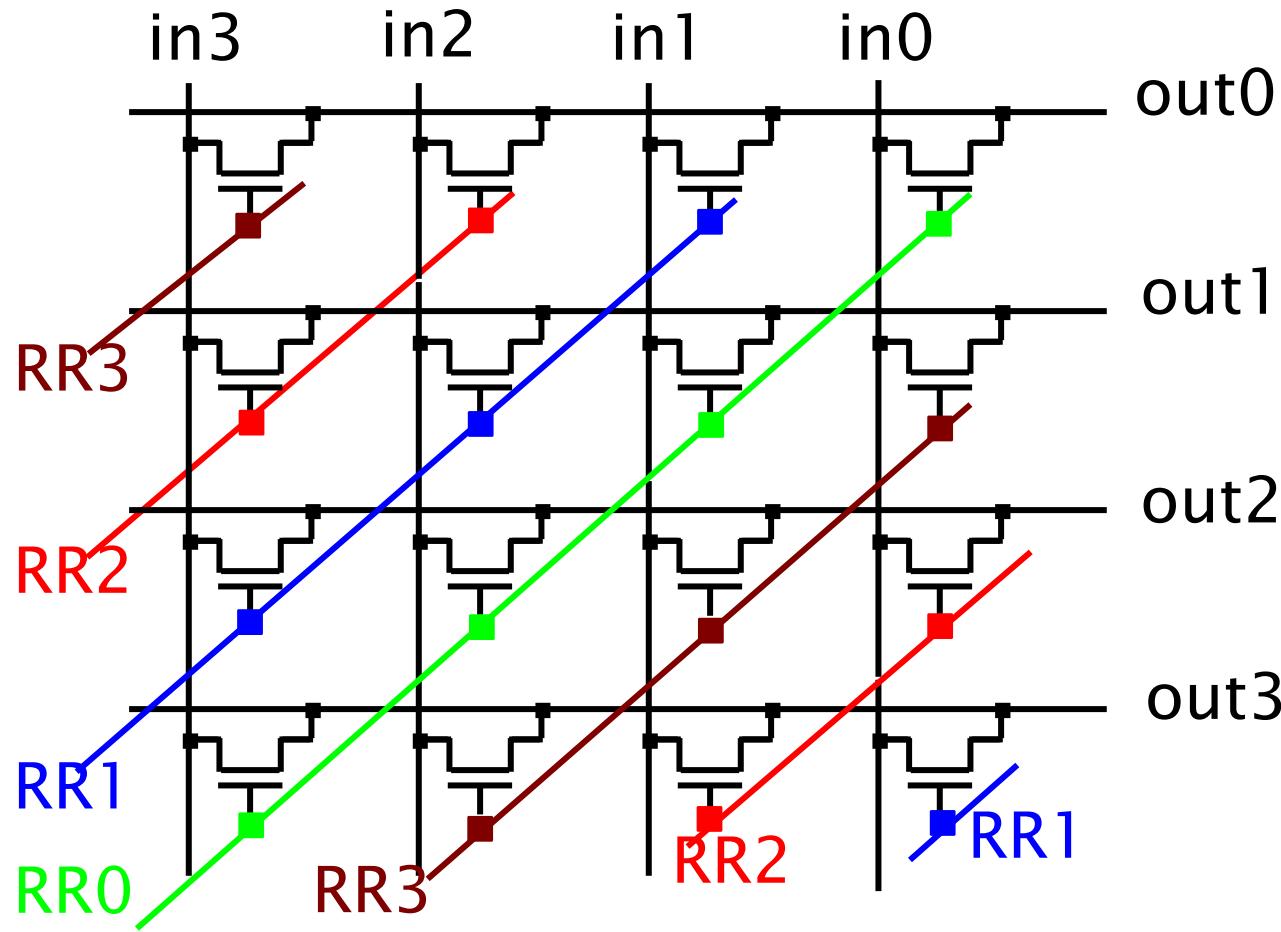
Rotacionando 2 bits para direita:

rotate 1 right(1100) → 0011



n-bit Shifter (barrel shifter)

Circuito completo:



Deslocamento Aritmético

□ Deslocamento lógico

10001 → 1 bit para a direita 01000

1 bit para a esquerda 00010

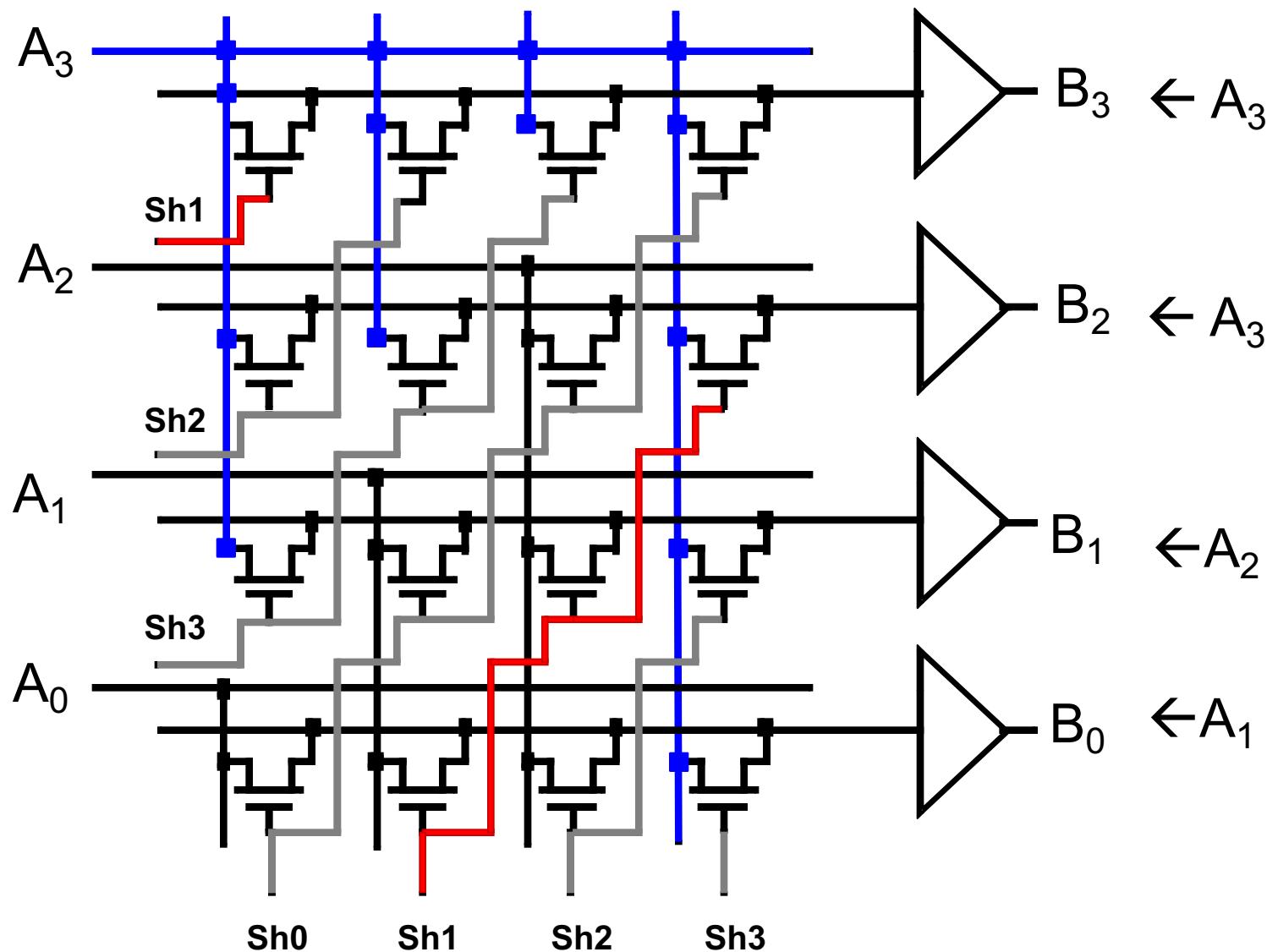
□ Deslocamento aritmético

00100 → 1 bit para a direita 00010 (de 4 para 2)

10100 → 1 bit para a direita 11010 (de -12 para -6)

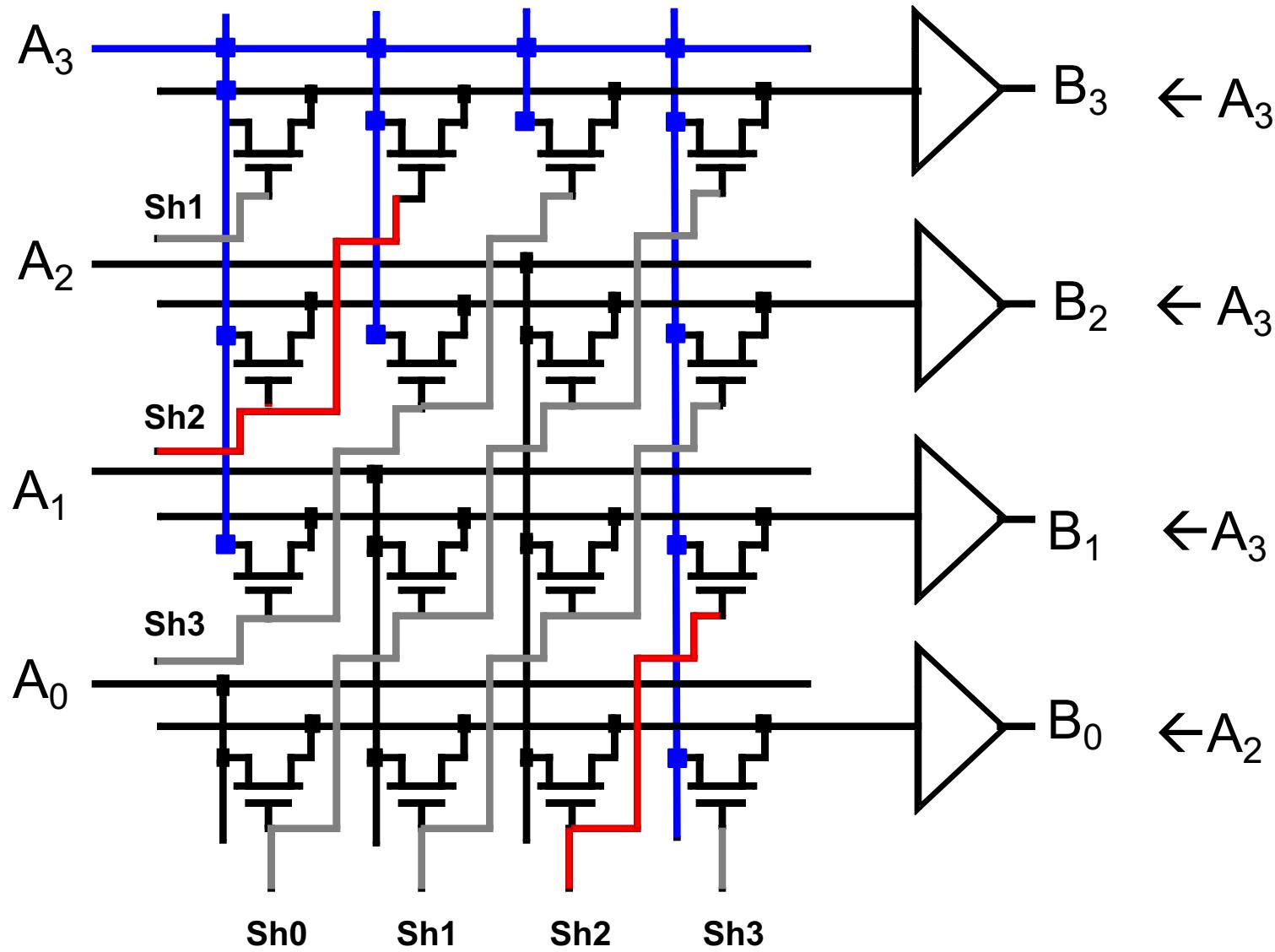
SH1 ← 1

Bit 3 wrapped around



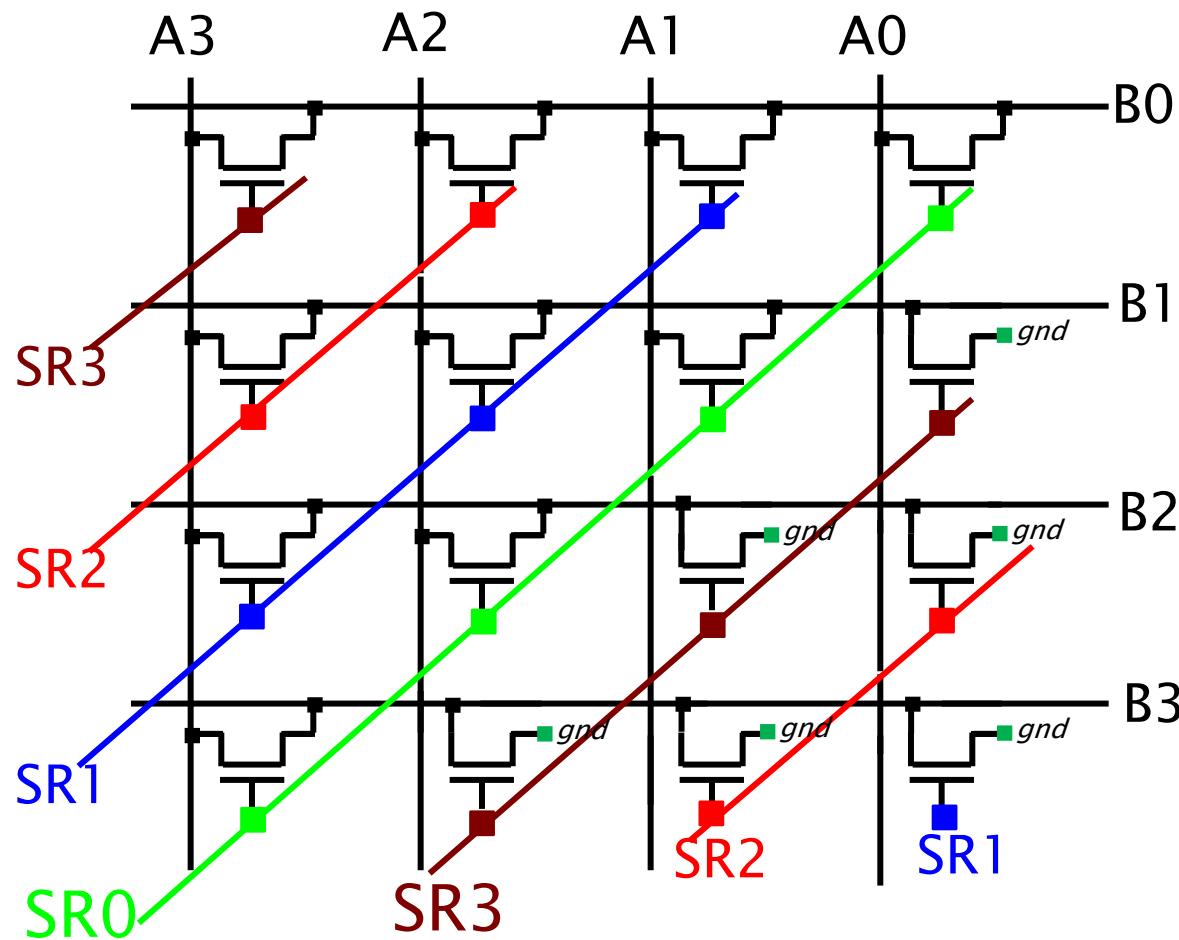
Area dominated by wiring

SH2 \leftarrow 1



4. (1,5) Circuitos de rotação e deslocamento. Apresentar uma matriz de transistores capaz de realizar deslocamentos para a direita para palavras de 4 bits, onde o bit mais significativo recebe '0' (gnd). A tabela abaixo ilustra os valores que os bits de saída (B3 a B0) devem receber em função dos bits de entrada (A3 a A0).

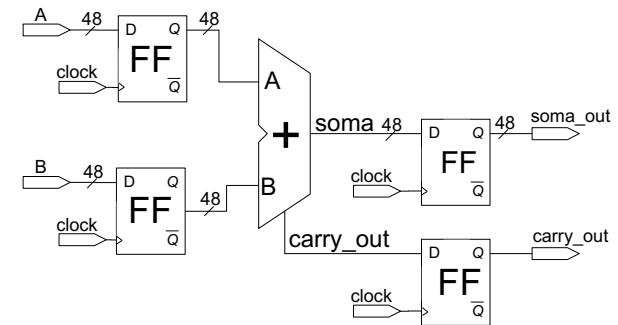
Ação	Comando	B3	B2	B1	B0
Não desloca	SRO	A3	A2	A1	A0
Deslocamento de 1 bit a direita	SR1	0	A3	A2	A1
Deslocamento de 2 bits a direita	SR2	0	0	A3	A2
Deslocamento de 3 bits a direita	SR3	0	0	0	A3



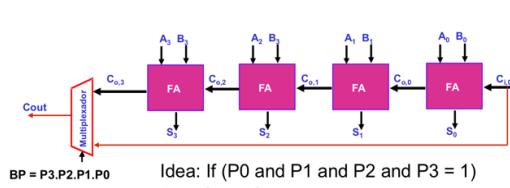
Exercício

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:

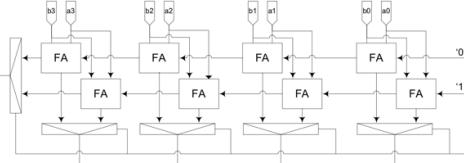
- Flip-flop D, com tempo de propagação $D \rightarrow Q$ igual a 25ps e o tempo de setup igual a 5ps.
- Somador completo (FA), com tempo de propagação igual a 90ps.
- Multiplexador, com tempo de propagação igual a 20ps.



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:

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

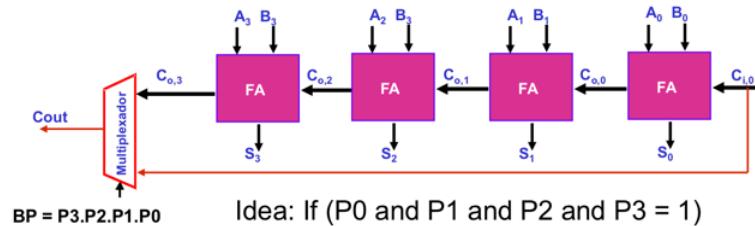
$$t_{BP} = (48/4) * 20 + 2*(4 * 90) = 960$$

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

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

Considere o somador *carry bypass* com duas configurações distintas: estágios (M) de 4 bits e de 8 bits, com $t_{FA} = 90\text{ps}$ e $t_{MUX} = 20\text{ps}$.

- Determine o valor do número de bits em que o atraso dos somadores com $M=4$ e $M=8$ se igualam $\rightarrow N_{igual}$
- De 8 bits até N_{igual} qual configuração do somador é mais rápida?



Idea: If (P_0 and P_1 and P_2 and $P_3 = 1$)
then $C_{0,3} = C_0$
Estágio de 4 bits do *carry bypass* (CBP4)

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

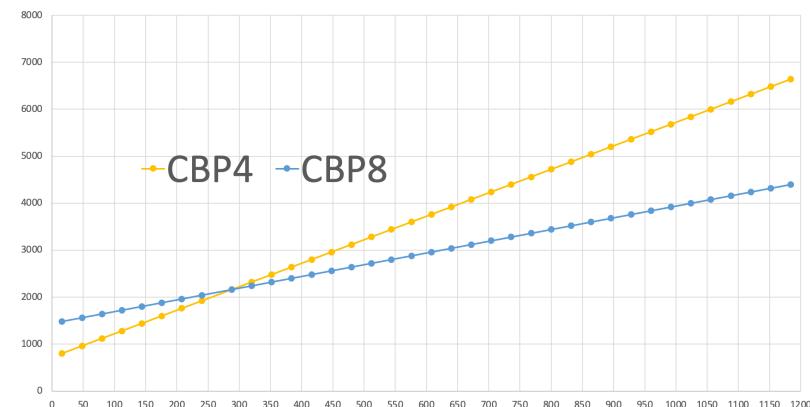
$$2 * 8 * 90 + \frac{N}{8} * 20 < 2 * 4 * 90 + \frac{N}{4} * 20$$

$$1440 + 2,5N < 720 + 5N$$

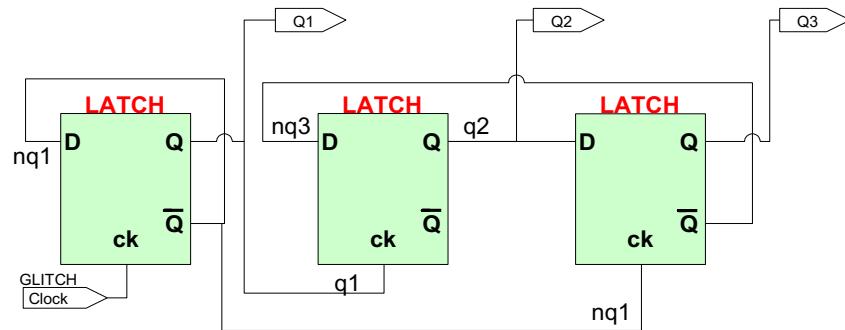
$$2,5N > 720$$

$$N > 288$$

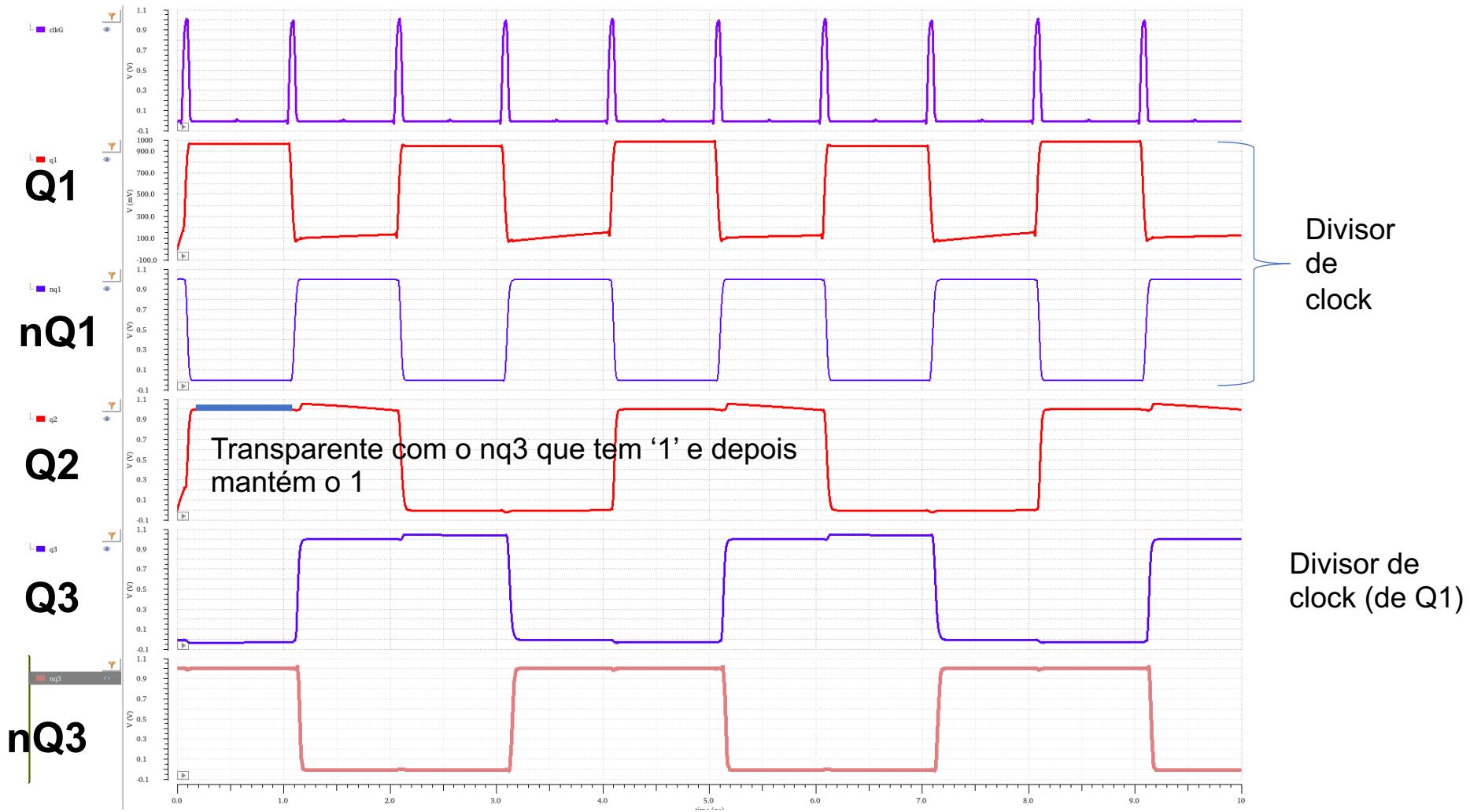
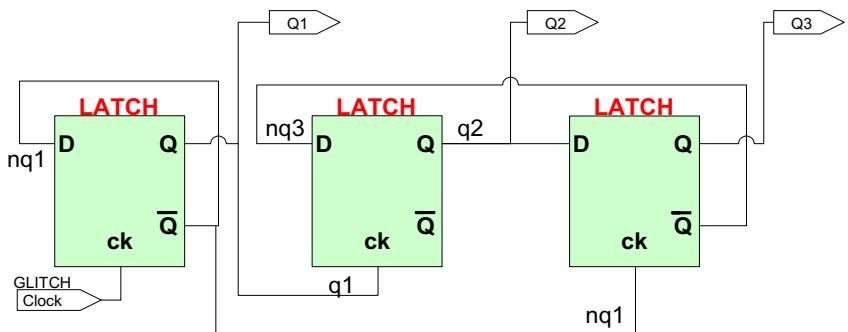
Até 288 bits o CP4 é mais rápido que o CP8. Após 288 bits o CP8 é mais rápido, pois há menos muxes no caminho



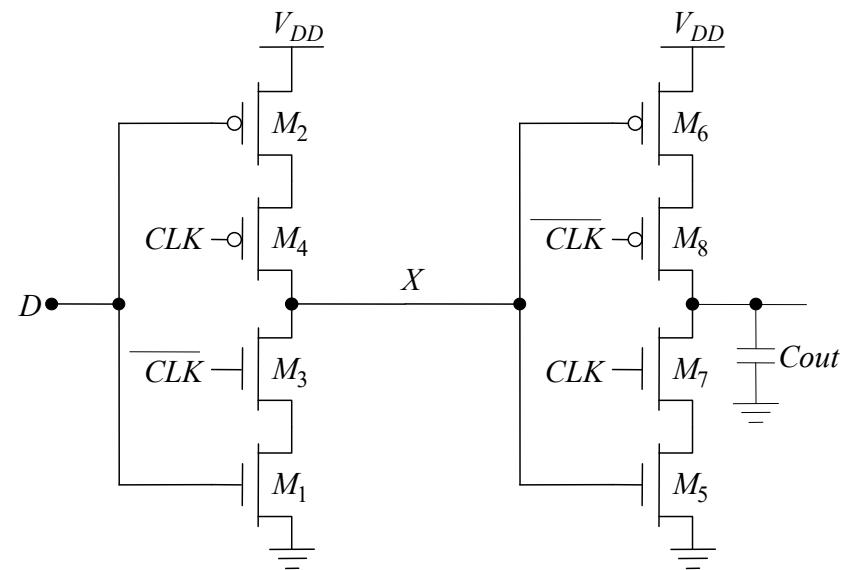
Um projetista desenvolveu o circuito abaixo para implementar um contador, usando apenas flip-flops do tipo *latch*, transparentes ao nível '1' do sinal 'ck', e um *clock* na forma de pulso (*glitch clock* - GCK) conectado ao pino 'ck' da primeira *latch*.



- Apresente, para os primeiros 8 pulsos de GCK (assumindo q1, q2, e q3 inicialmente em zero) as formas de onda para os sinais: GCK, q1, nq1, q2, q3.
- O sinal 'q1' opera como um divisor de *clock*? Explique.
- Qual o comportamento do par 'q2-q3' em relação ao sinal 'q1'?



Considere o flip-flop representado abaixo.



- Desenhar o circuito equivalente para os níveis de $CLK='0'$ e $CLK='1'$, explicando o comportamento do circuito.
- O circuito, para armazenamento da entrada D, é sensível ao nível ou à borda do sinal clock (CLK)? Se for sensível ao nível do sinal clock, para qual nível o circuito é transparente, ou se for sensível à borda, para qual borda o circuito armazena a informação.
- Explique o comportamento do circuito quando acontecer clock overlap em '1' ou seja, $CLK='1'$ e $CLK='1'$.

Exemplo de Circuito Complexo

IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 43, NO. 1, JANUARY 2008

29

An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS

Sriram R. Vangal, *Member, IEEE*, Jason Howard, Gregory Ruhl, *Member, IEEE*, Saurabh Dighe, *Member, IEEE*, Howard Wilson, James Tschanz, *Member, IEEE*, David Finan, Arvind Singh, *Member, IEEE*, Tiju Jacob, Shailendra Jain, Vasantha Erraguntla, *Member, IEEE*, Clark Roberts, Yatin Hoskote, *Member, IEEE*, Nitin Borkar, and Shekhar Borkar, *Member, IEEE*

Abstract—This paper describes an integrated network-on-chip architecture containing 80 tiles arranged as an 8×10 2-D array of floating-point cores and packet-switched routers, both designed to operate at 4 GHz. Each tile has two pipelined single-precision floating-point multiply accumulators (FPMAC) which feature a single-cycle accumulation loop for high throughput. The on-chip 2-D mesh network provides a bisection bandwidth of 2 Terabits/s. The 15-FO4 design employs mesochronous clocking, fine-grained clock gating, dynamic sleep transistors, and body-bias techniques. In a 65-nm eight-metal CMOS process, the 275 mm² custom design contains 100 M transistors. The fully functional first silicon achieves over 1.0 TFLOPS of performance on a range of benchmarks while dissipating 97 W at 4.27 GHz and 1.07 V supply.

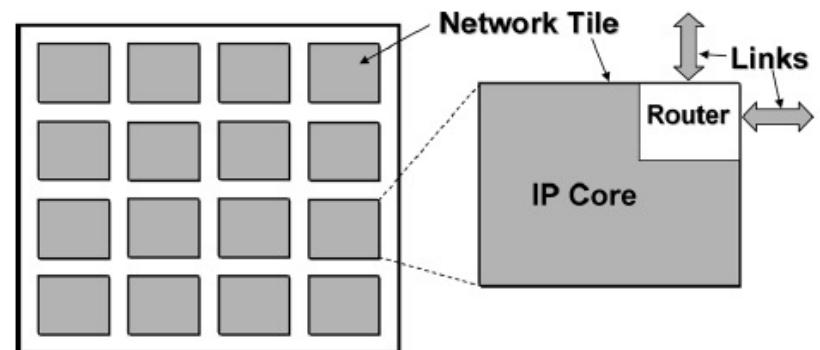
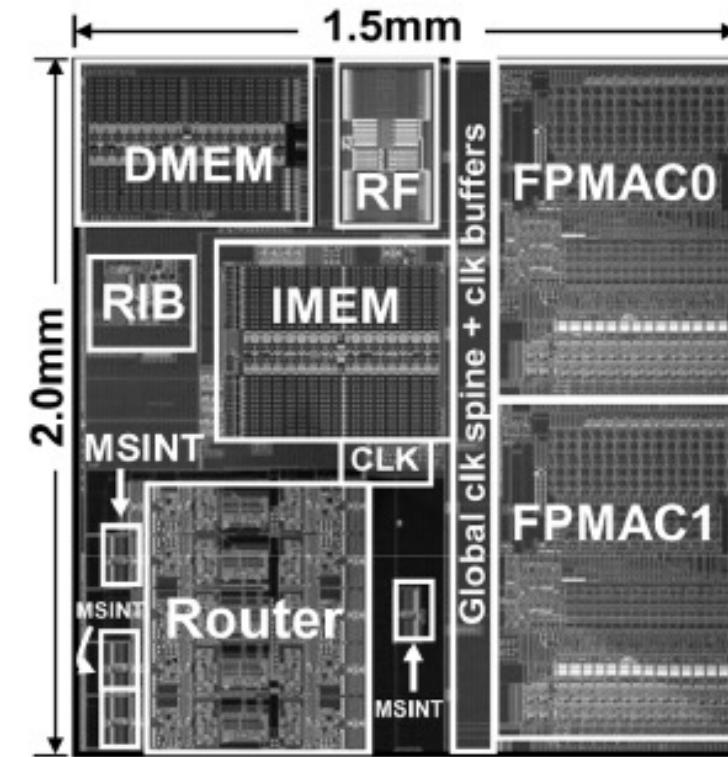
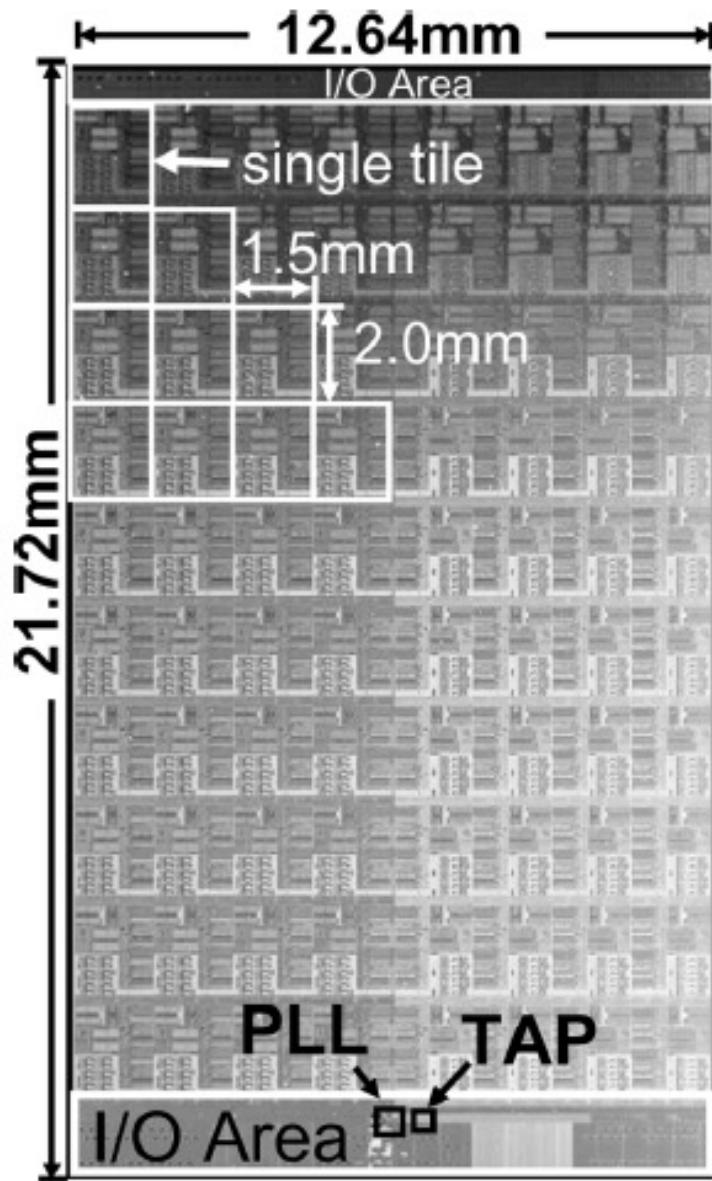
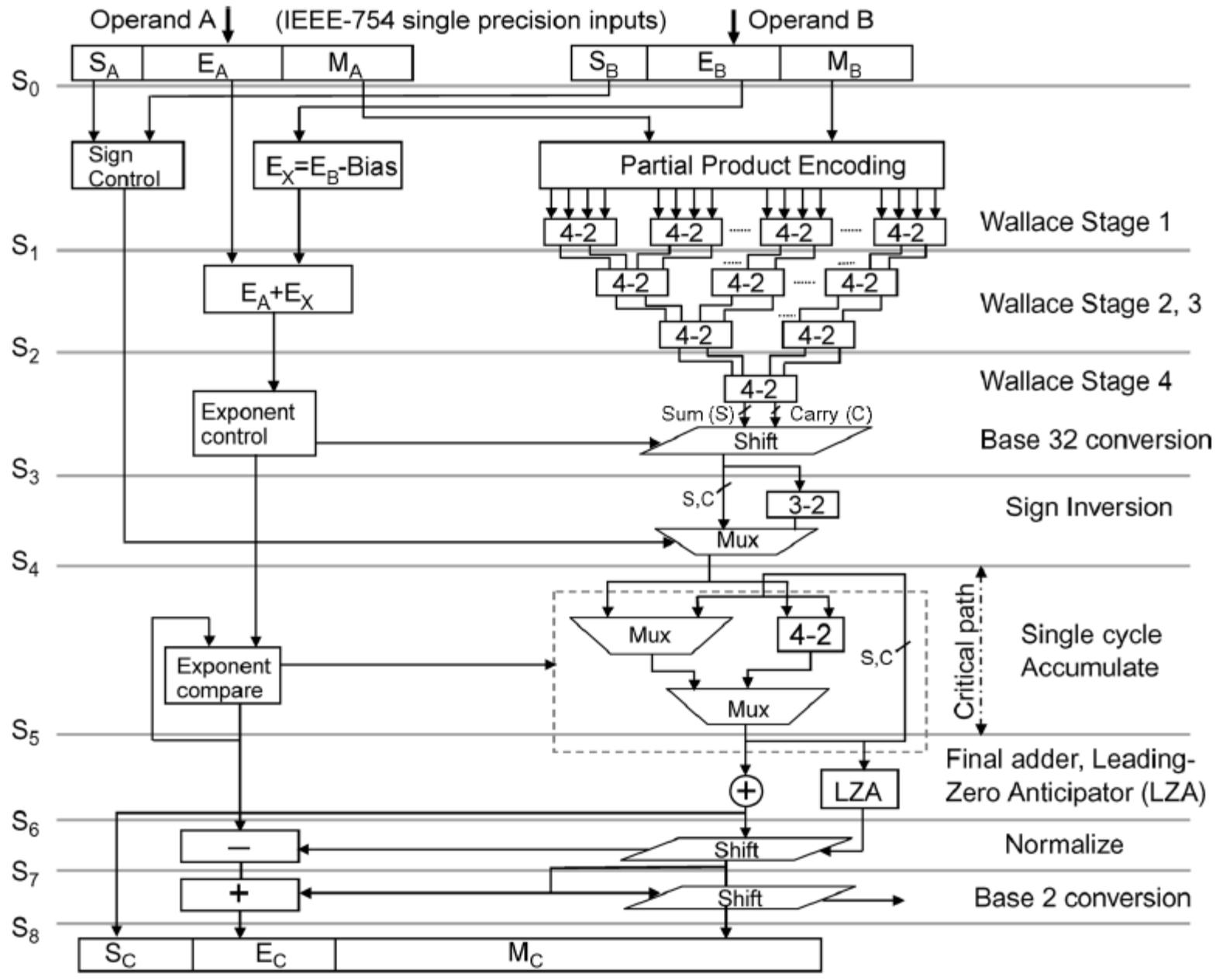


Fig. 1. NoC architecture.



Technology	65nm, 1 poly, 8 metal (Cu)
Transistors	100 Million (full-chip) 1.2 Million (tile)
Die Area	275mm ² (full-chip) 3mm ² (tile)
C4 bumps #	8390



FPMAC nine-stage pipeline with single-cycle accumulate loop.

Flip-flops utilizados no circuito

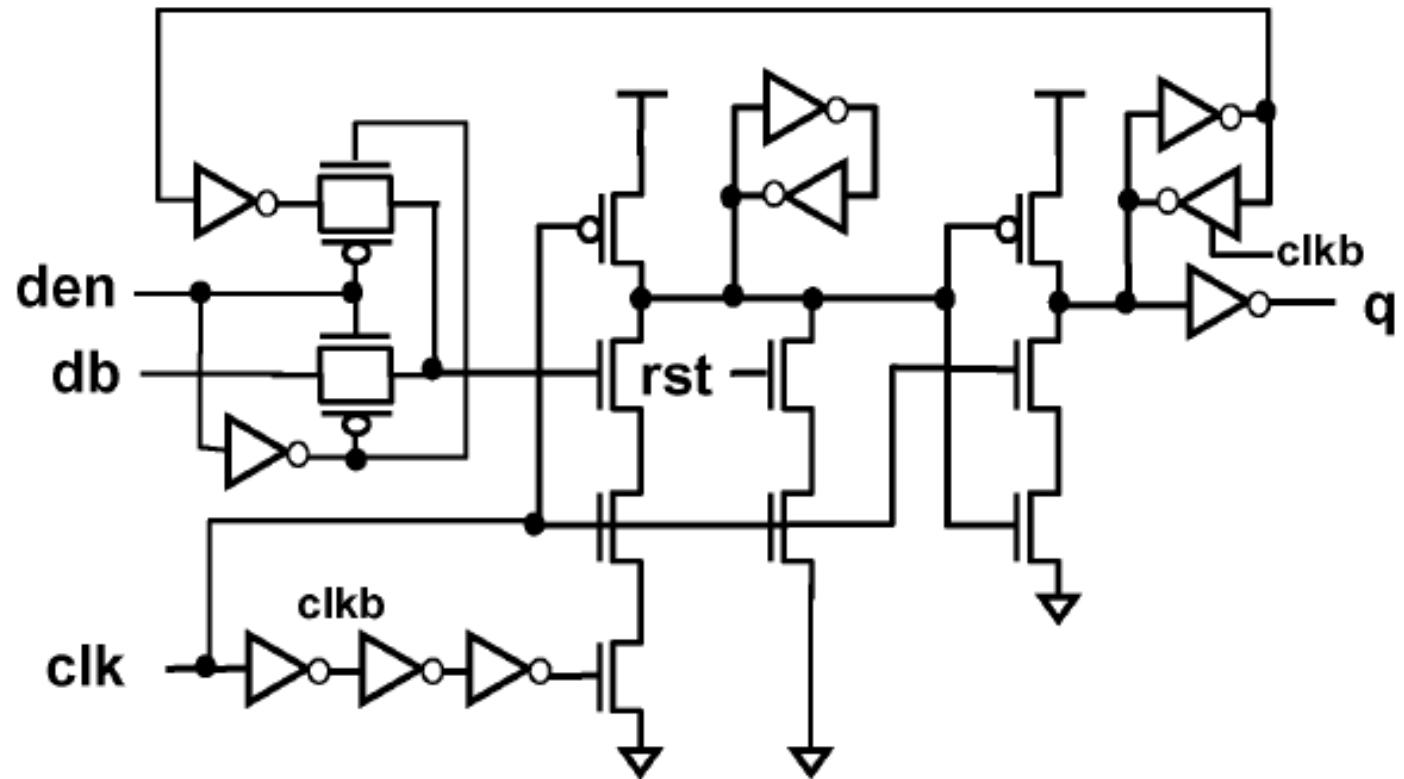
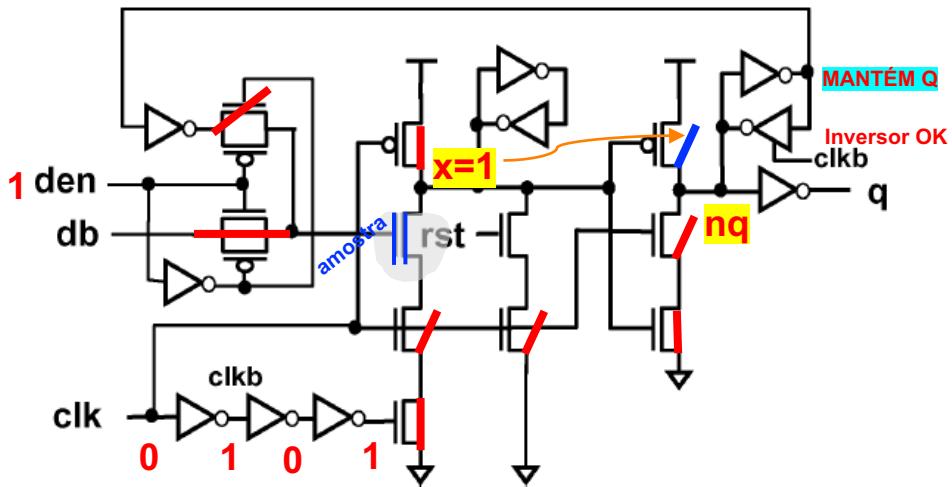
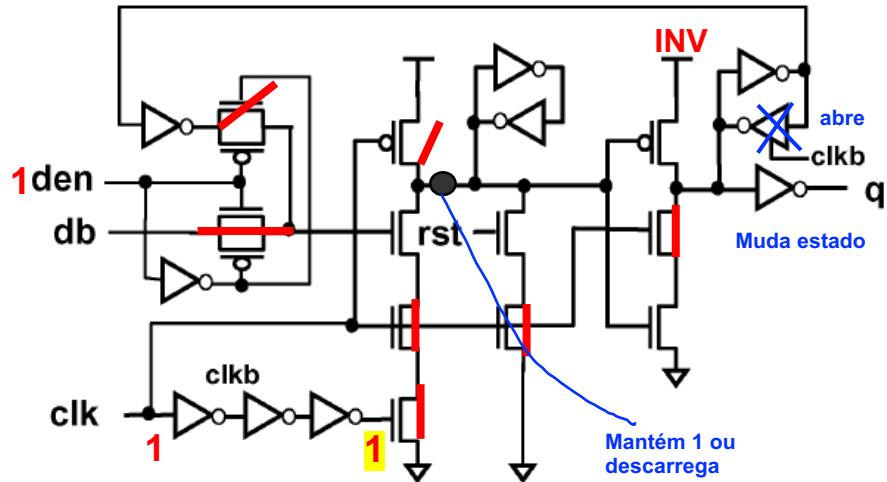


Fig. 8. Semi-dynamic flip-flop (SDFF) schematic.

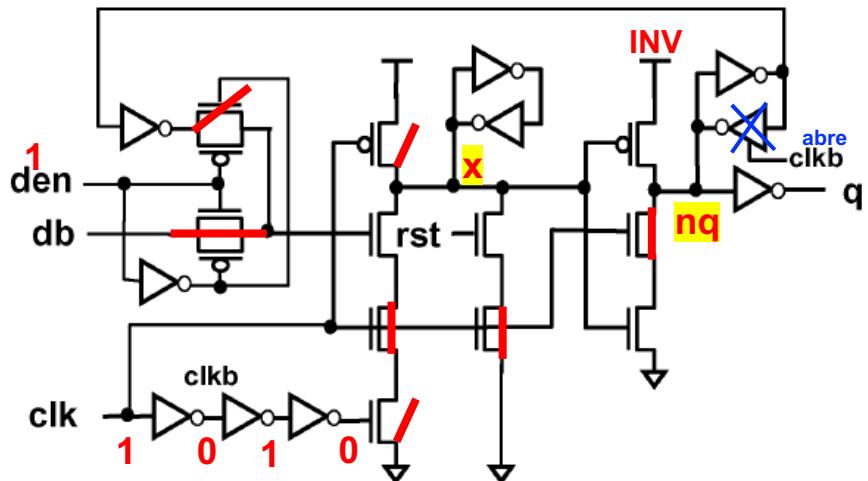


x: em pré-carga
com rst vai a zero - última informação amostrada

ck=0



ck= sobe



ck=1

x: último dado, isolado por 'Z' – não muda

ck= desce

- mesma condição de ck=0
- transfere Q do laço interno para o laço externo

den=0: mantém a informação anterior, independente do clock

