INF01 118

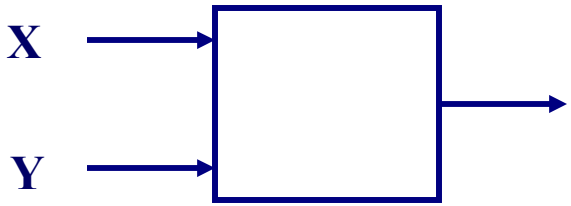
Técnicas Digitais para Computação

UFRGS

ULA Multifunção
Multiplicadores

Aula 15

Comparador

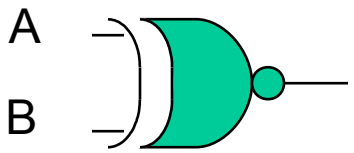


COMP = 1 se X = Y

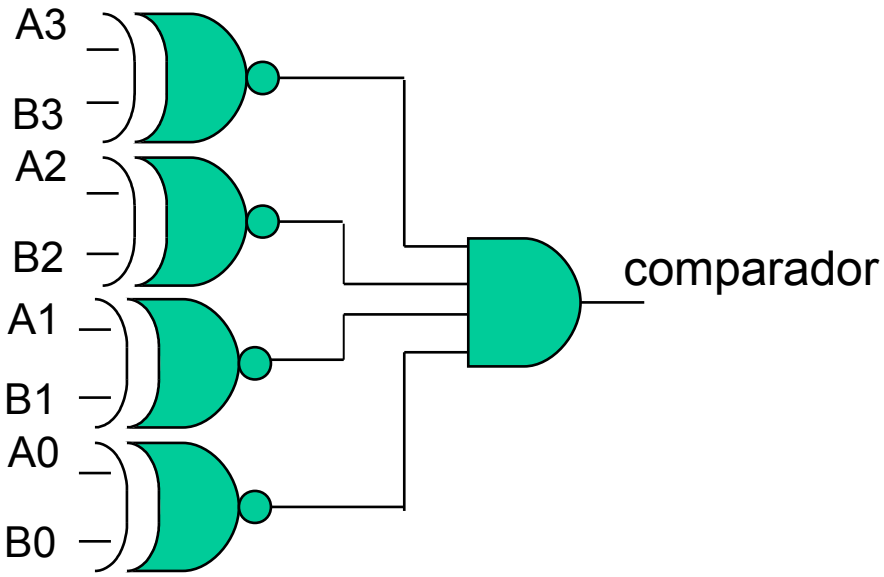
COMP = 0 se X ≠ Y

X	Y	COMP
0	0	1
0	1	0
1	0	0
1	1	1

$$COMP = X\,Y + \overline{X}\,\overline{Y} = \overline{X \oplus Y}$$

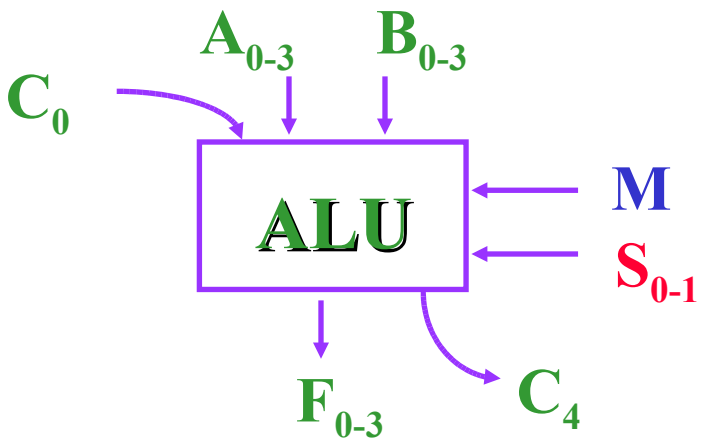


Comparador (igualdade) de 4 bits (A3A2A1A0 e B3B2B1B0)



Projeto de Unidade Aritmética e Lógica

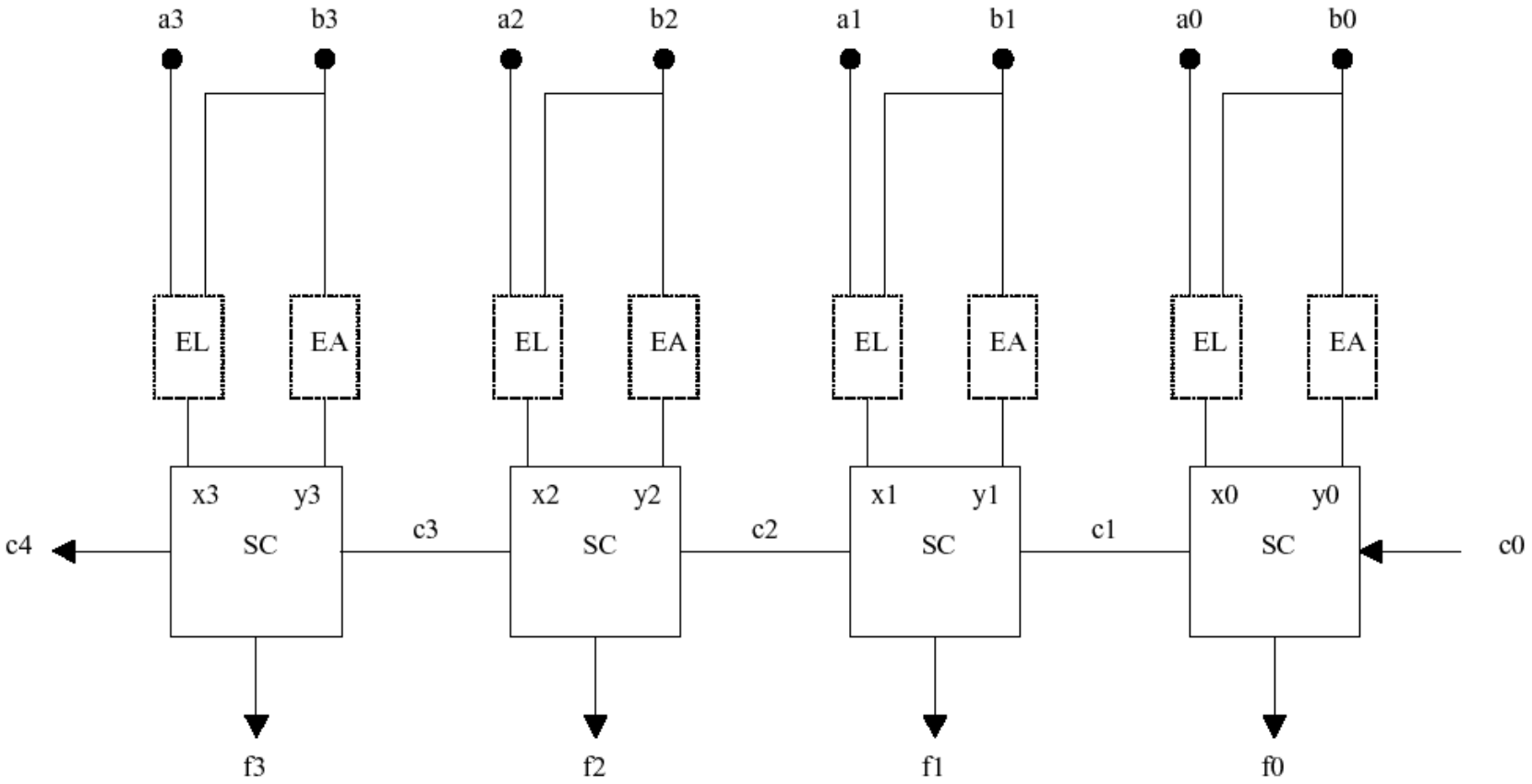
Exemplo de uma ULA Simples



M = Modo
- 1 funções aritméticas
- 0 funções lógicas

S = Seleção da função

Projeto de Unidade Aritmética e Lógica



Projeto de Unidade Aritmética e Lógica

M	S1	S0	nome da função	F	X	Y	C0
0	0	0	Complementa	A’	A’	0	0
0	0	1	E	A E B	A E B	0	0
0	1	0	Identidade	A	A	0	0
0	1	1	OU	A OU B	A OU B	0	0
1	0	0	Decrementa	A-1	A	Todos 1s	0
1	0	1	Soma	A+B	A	B	0
1	1	0	Subtrai	A+B’+1	A	B’	1
1	1	1	Incrementa	A+1	A	Todos 0s	1

Projeto de Unidade Aritmética e Lógica

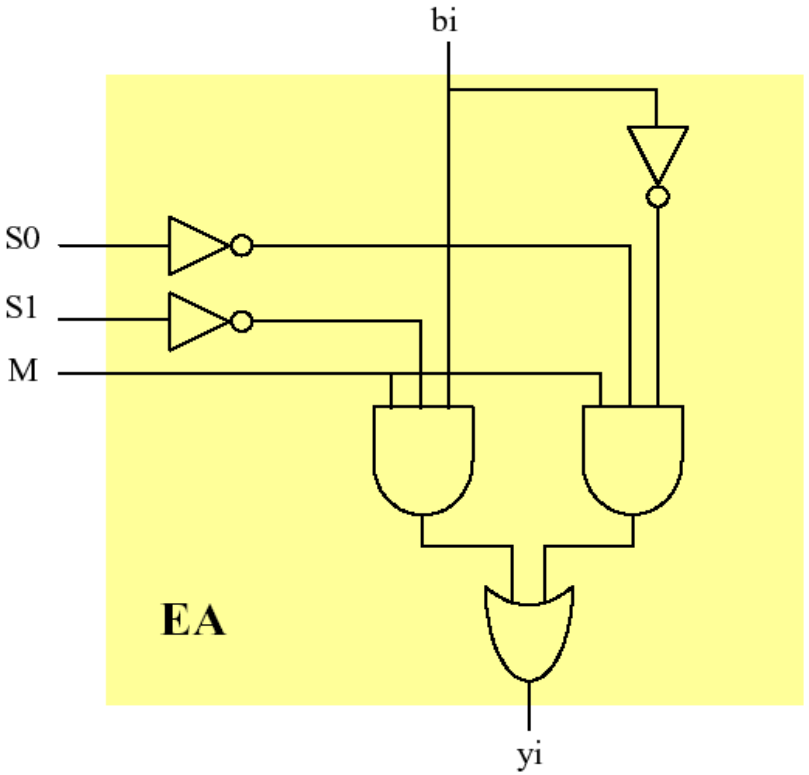
Tabela 2 – Tabela-verdade para o extensor aritmético.

S1	S0	bi	yi
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Tabela 3 – Mapa de Karnaugh para o extensor aritmético.

S1S0 bi	00	01	11	10
0	**1**	0	0	**1**
1	**1**	**1**	0	0

Extensor Aritmético



Projeto de Unidade Aritmética e Lógica

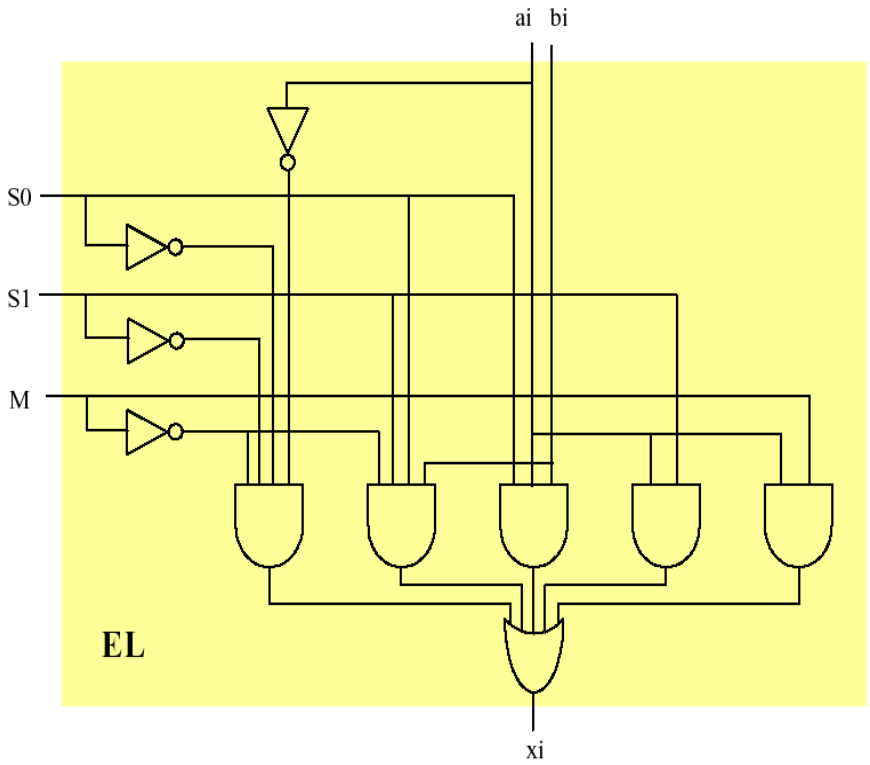
Tabela 4 – Tabela-verdade para o extensor lógico.

M	S1	S0	xi
0	0	0	ai'
0	0	1	ai.bi
0	1	0	ai
0	1	1	ai+bi
1	?	?	ai

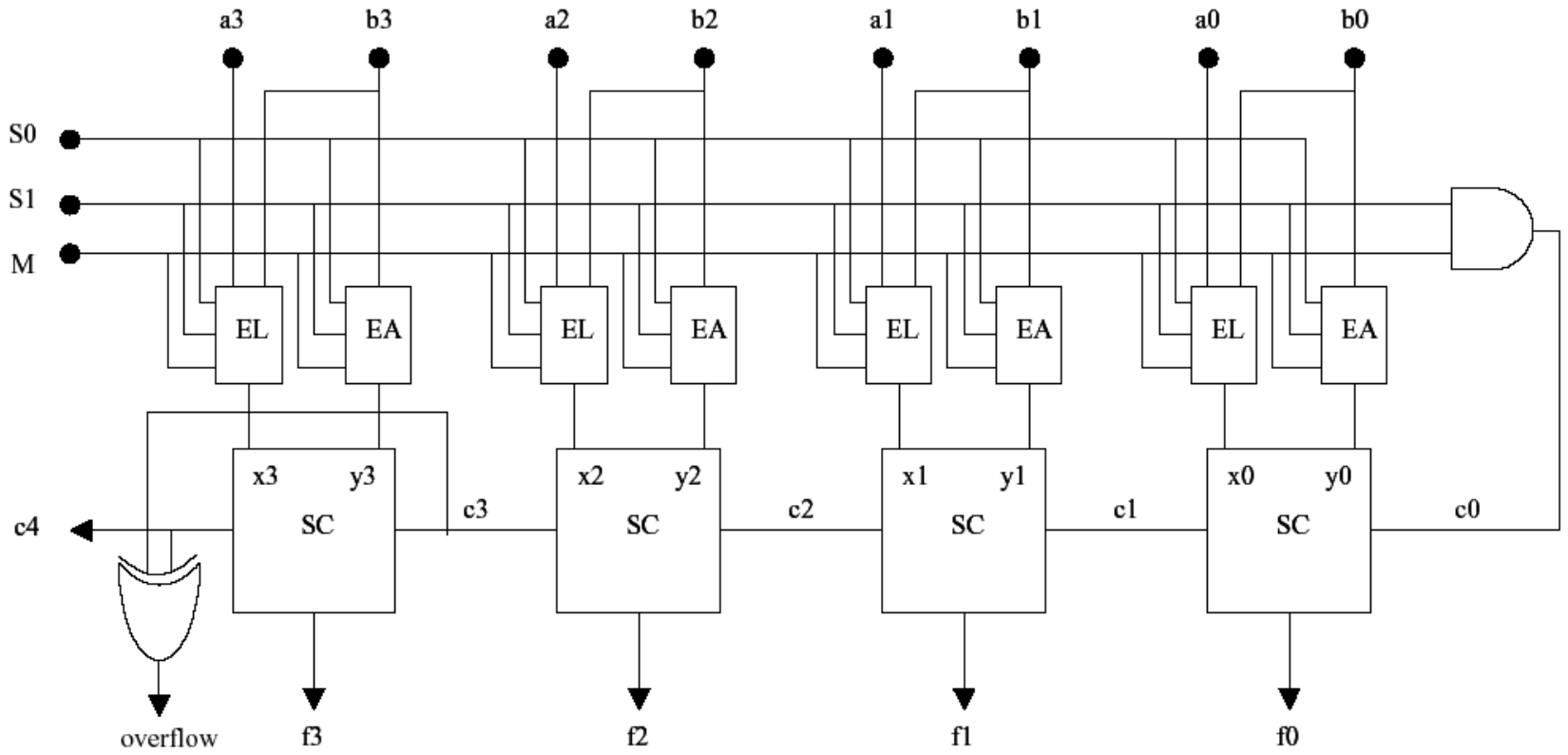
Tabela 5 – Mapa de Karnaugh para o extensor aritmético.

	M=0				M=1			
S1S0 / ai bi	00	01	11	10	00	01	11	10
00	**1**							
01	**1**		**1**					
11		**1**	**1**	**1**	**1**	**1**	**1**	**1**
10			**1**	**1**	**1**	**1**	**1**	**1**

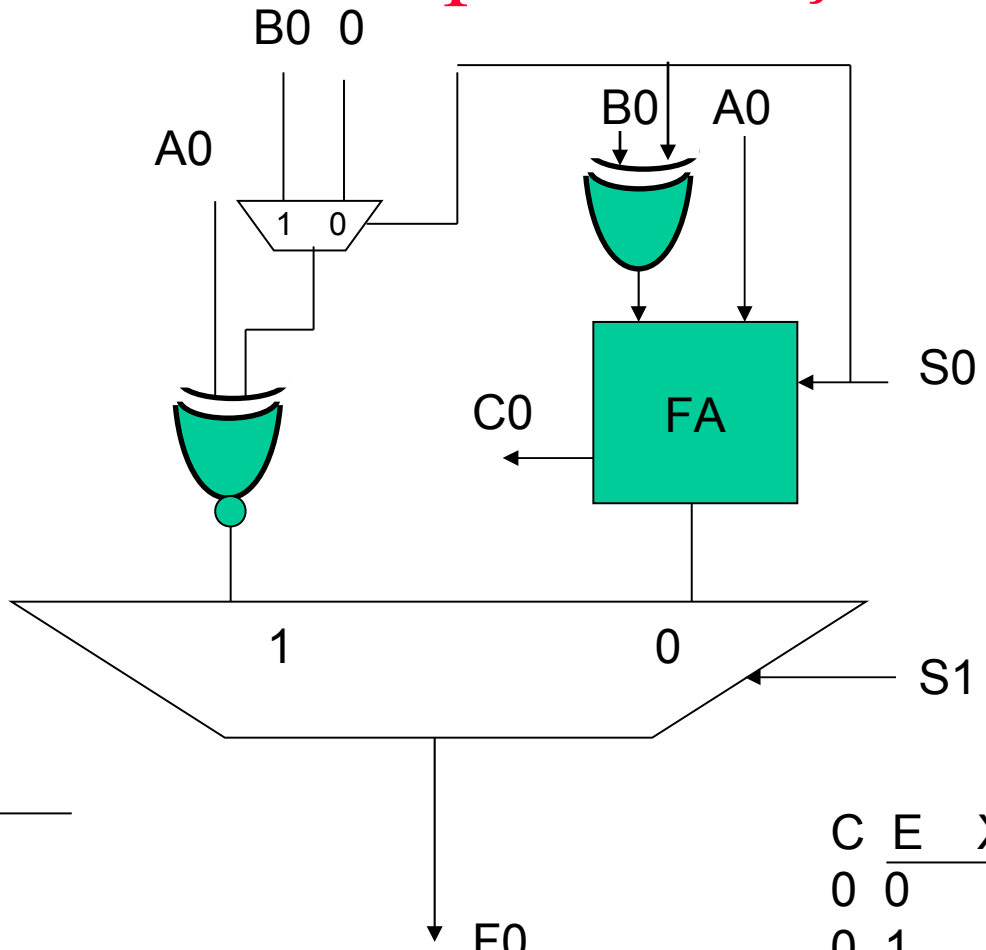
Extensor Lógico



Projeto de Unidade Aritmética e Lógica



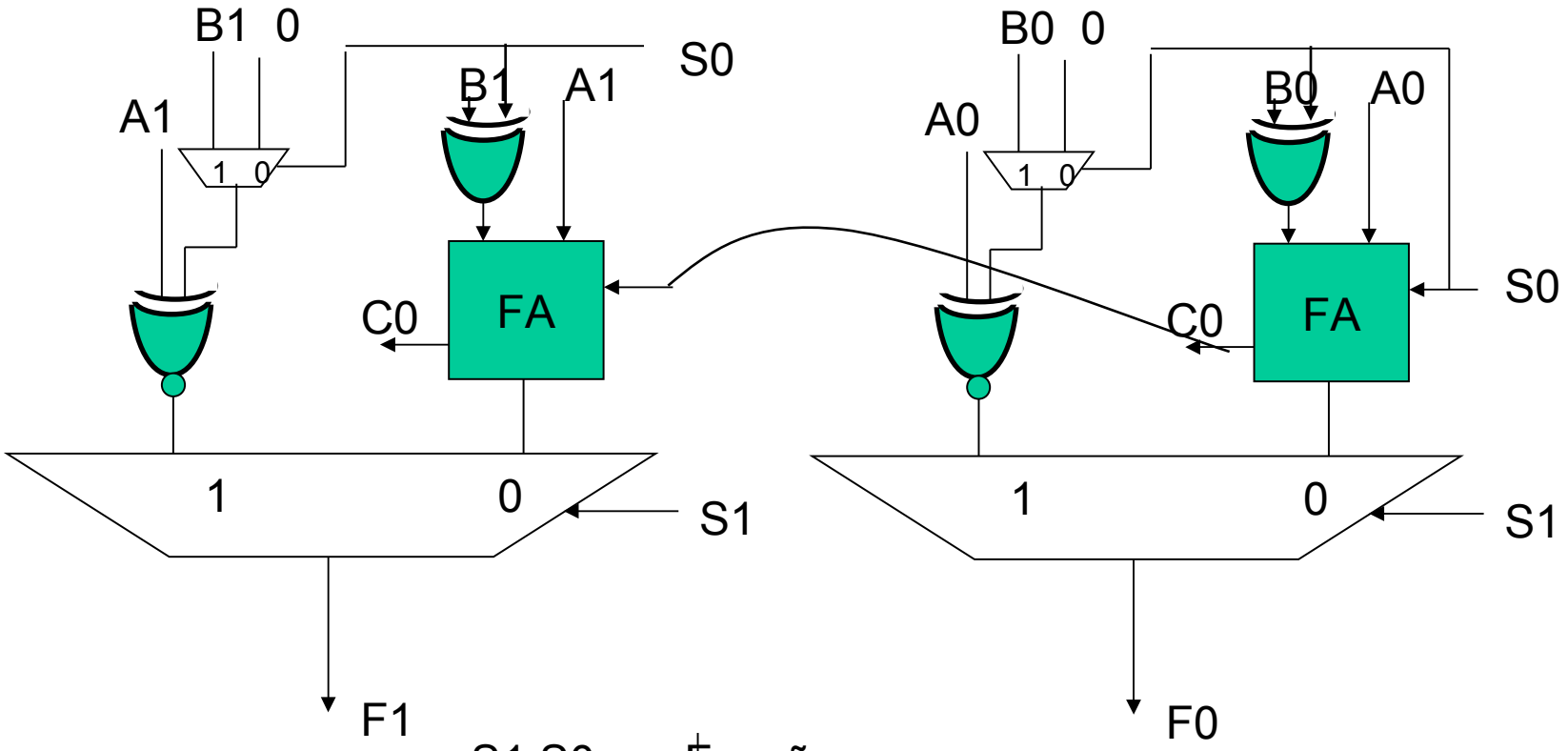
Projeto de ULA de 1 bit
Alternativa de implementação #2



S1	S0	Função
0	0	soma
0	1	subtração
1	0	inversão
1	1	comparação

C	E	XOR	XNOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

ULA de 2 bits



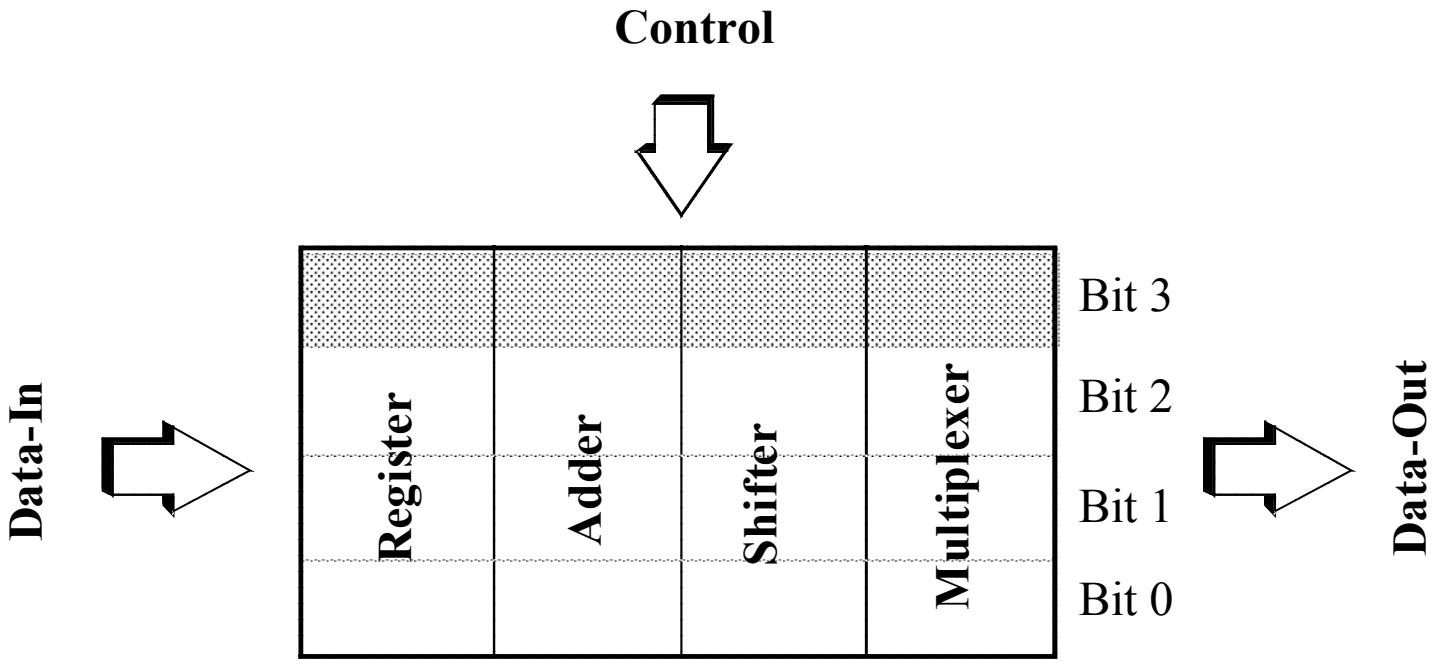
S1	S0	Função
0	0	soma
0	1	subtração
1	0	inversão
1	1	comparação

ULA em VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ULA is
   Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
          B : in  STD_LOGIC_VECTOR (3 downto 0);
          C : in  STD_LOGIC_VECTOR (1 downto 0);
          F : out  STD_LOGIC_VECTOR (3 downto 0));
end ULA;

architecture Behavioral of ULA is
Begin

process (A, B, C)
begin
CASE C is
WHEN "00" => F <= A+B;
WHEN "01" => F <= A-B;
WHEN "10" => F <= A xor B;
WHEN others => F <= not A;
END CASE;
end process;

end Behavioral;
```

ULA

Projeto Físico de ULA - Planta Baixa tipo Bit-Slice



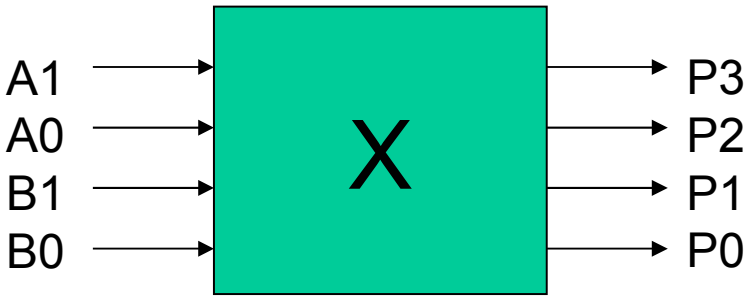
Bit-Slice **de mesma altura**

"Tiling" de elementos aritméticos e registradores

Multiplicador Combinacional

$A_1A_0 \times B_1B_0 = P_3P_2P_1P_0$ Exemplo: Mult de 2 bits

1)



A_1	A_0	B_1	B_0	P_3	P_2	P_1	P_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

2) Equações em SDP, simplificado por mapa de Karnaugh

$P_3 = m_{15}$

$P_2 = \sum m\ (10, 11, 14)$

$P_1 = \sum m\ (6, 7, 9, 11, 13, 14)$

$P_0 = \sum m\ (5, 7, 13, 15)$

→

$P_0 = A_0B_0$

$P_1 = \overline{A}_1A_0B_1 + \overline{A}_1B_0A_1 + A_1\overline{A}_0B_0 + B_1\overline{B}_0A_0$

$P_2 = A_1\overline{A}_0B_1 + A_1B_1\overline{B}_0$

$P3 = A_1A_0B_1B_0$

Multiplicador de 2 bits. Síntese de circuito combinacional

A_1	A_0	B_1	B_0	P_3	P_2	P_1	P_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

A1 A0 = X1 X0

B1 B0 = Y1 Y0

$P_3 = m_{15}$

$P_2 = \sum m\ (10, 11, 14)$

$P_1 = \sum m\ (6, 7, 9, 11, 13, 14)$

$P_0 = \sum m\ (5, 7, 13, 15)$

P_1

X_1X_0 \ Y_1Y_0	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	1	0	1
10	0	1	1	0

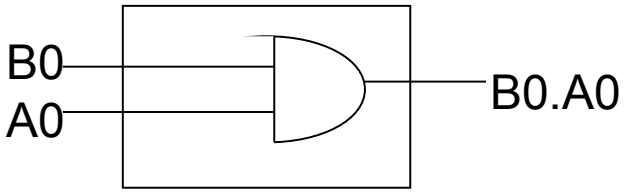
$P_1 = \overline{X_1}X_0Y_1 + \overline{Y_1}Y_0X_1 + X_1\overline{X_0}Y_0 + Y_1\overline{Y_0}X_0$

Multiplicador Combinacional

$A_1A_0 \times B_1B_0 = P_3P_2P_1P_0$

$$\begin{array}{lcccc} & & & A_1 & A_0 \\ x & & & B_1 & B_0 \\ \hline & & & B_0A_1 & B_0A_0 \\ + & B_1A_1 & B_1A_0 & & \\ \hline P_3 & P_2 & P_1 & & P_0 \end{array}$$

1) Multiplicação é equivalente a uma operação E



2) Soma binária dos termos:

$$\mathbf{Soma} = X \oplus Y \oplus C_{in}$$
$$C_{out} = XY + XC_{in} + YC_{in}$$

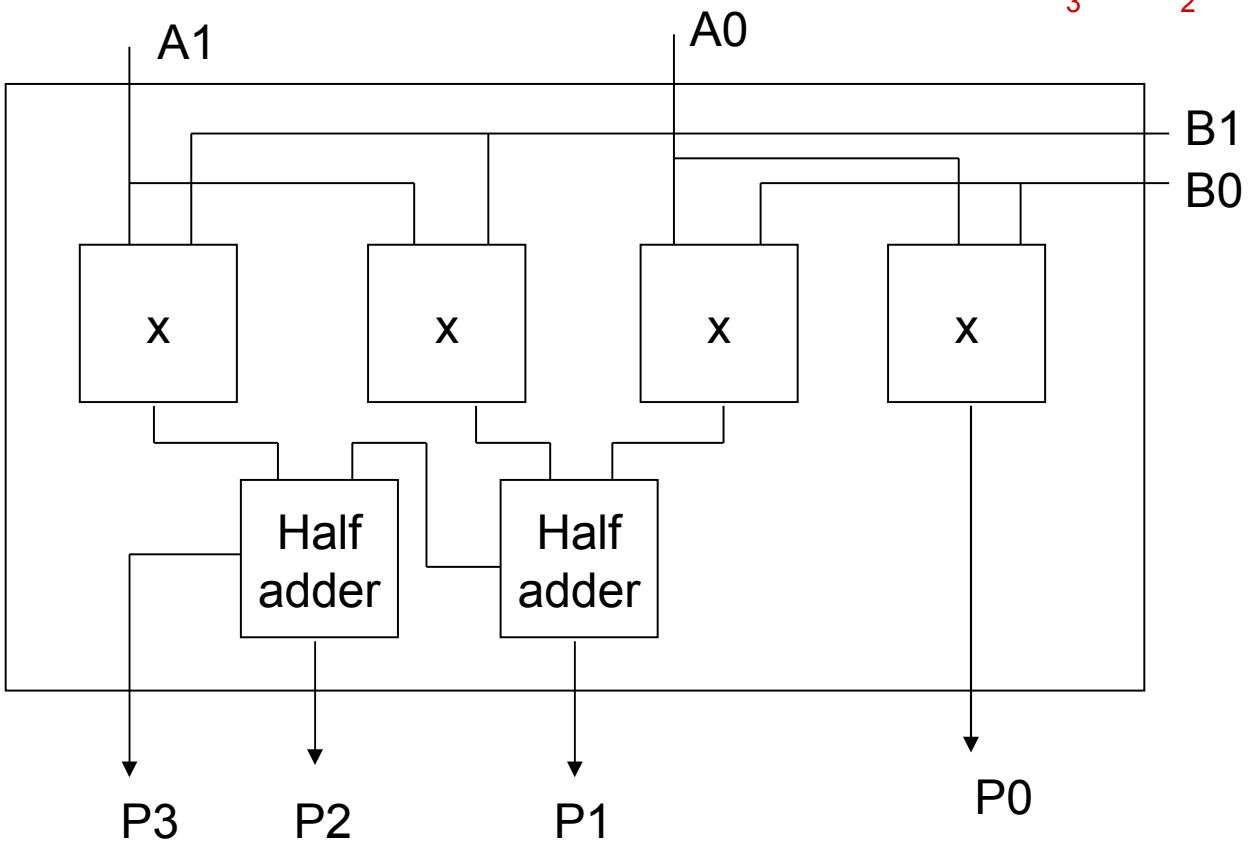
3) Substituindo os X e Y pelos termos da multiplicação:

$$P_0 = A_0B_0$$
$$P_1 = \overline{A}_1A_0B_1 + \overline{A}_1B_0A_1 + A_1\overline{A}_0B_0 + B_1\overline{B}_0A_0$$
$$P_2 = A_1\overline{A}_0B_1 + A_1B_1\overline{B}_0$$
$$P3 = A_1A_0B_1B_0$$

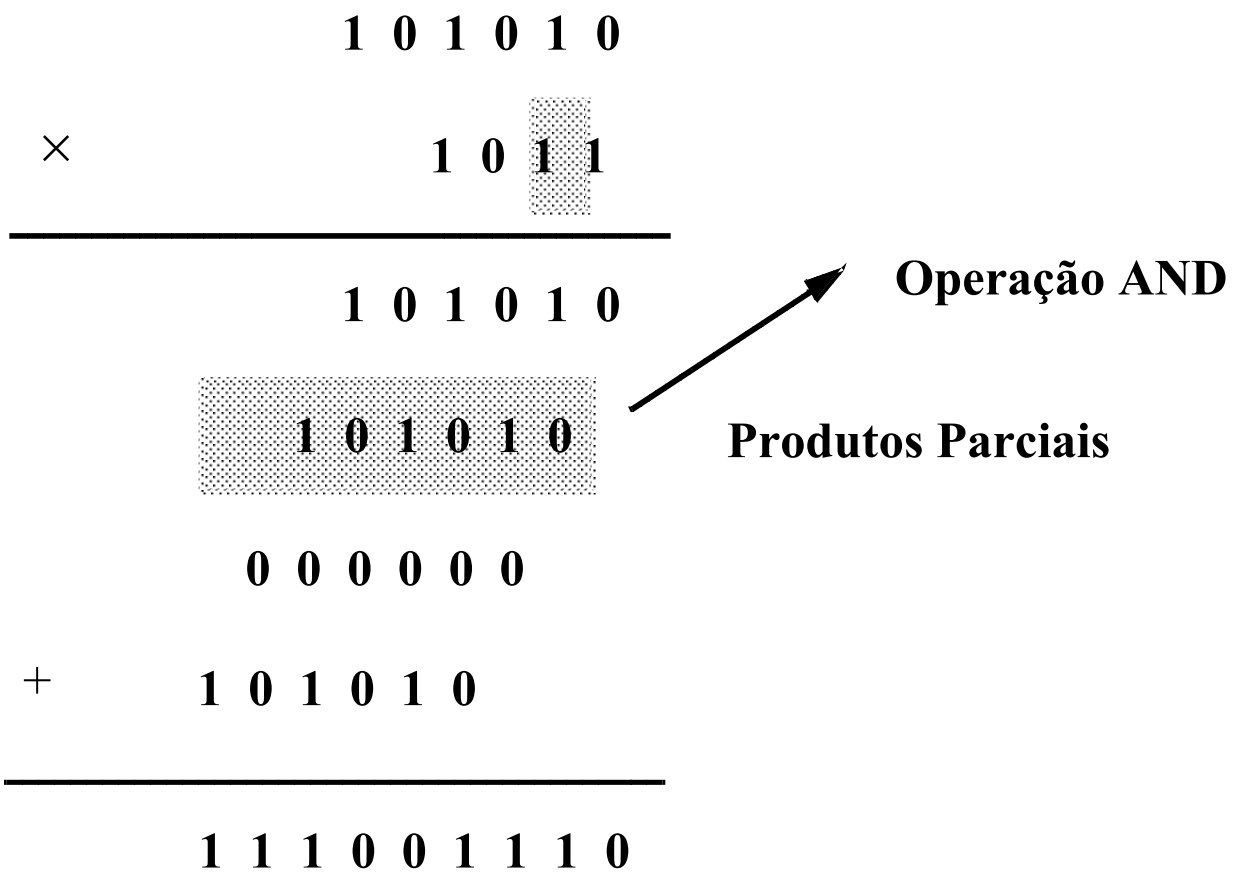
Multiplicador Combinacional

$A_1A_0 \times B_1B_0 = P_3P_2P_1P_0$

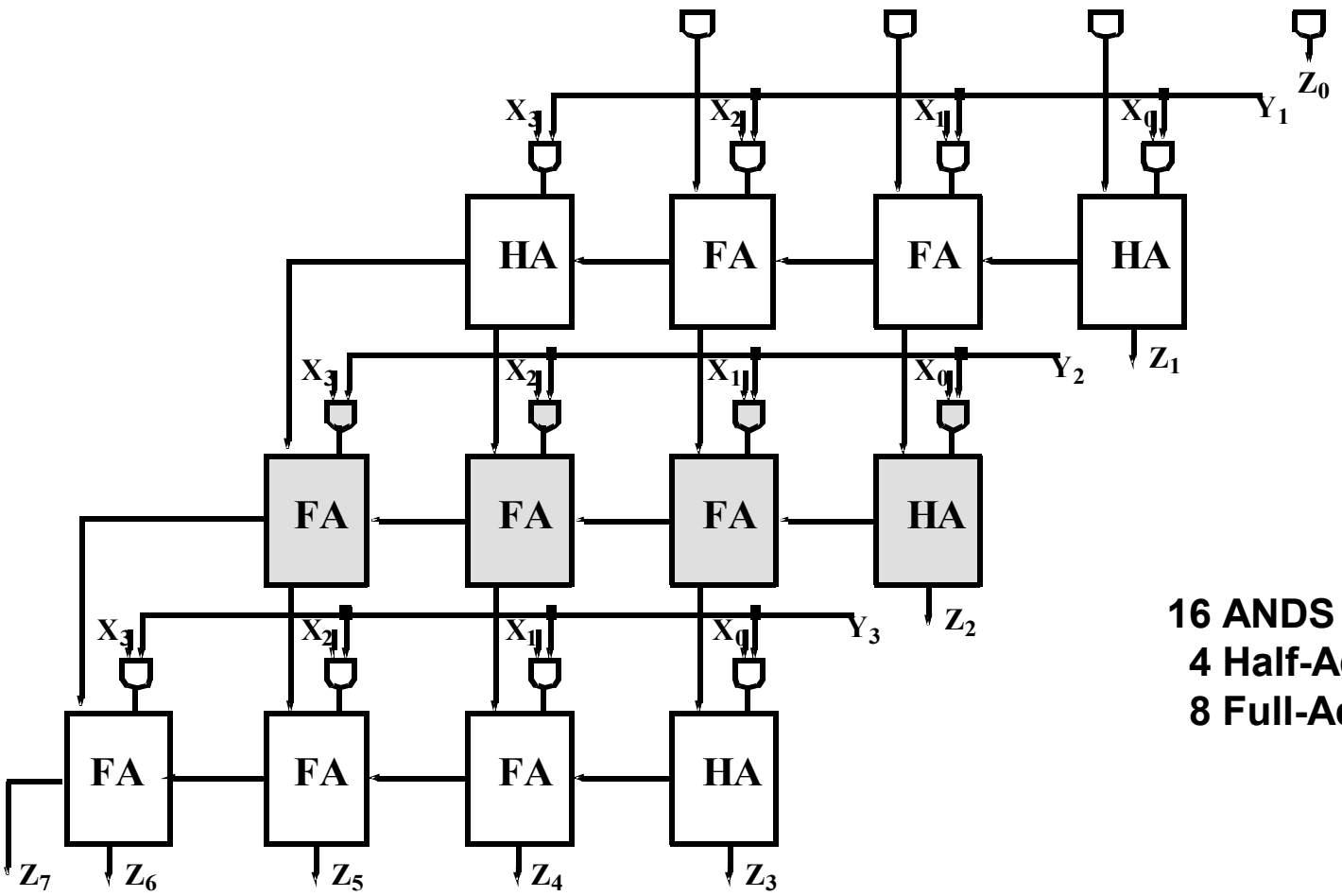
$$
\begin{array}{lcccc}
 & & & A_1 & A_0 \\
\times & & & B_1 & B_0 \\
\hline
 & & B_0A_1 & B_0A_0 & \\
+ & B_1A_1 & B_1A_0 & & \\
\hline
P_3 & P_2 & P_1 & & P_0
\end{array}
$$



Uso de Produtos Parciais



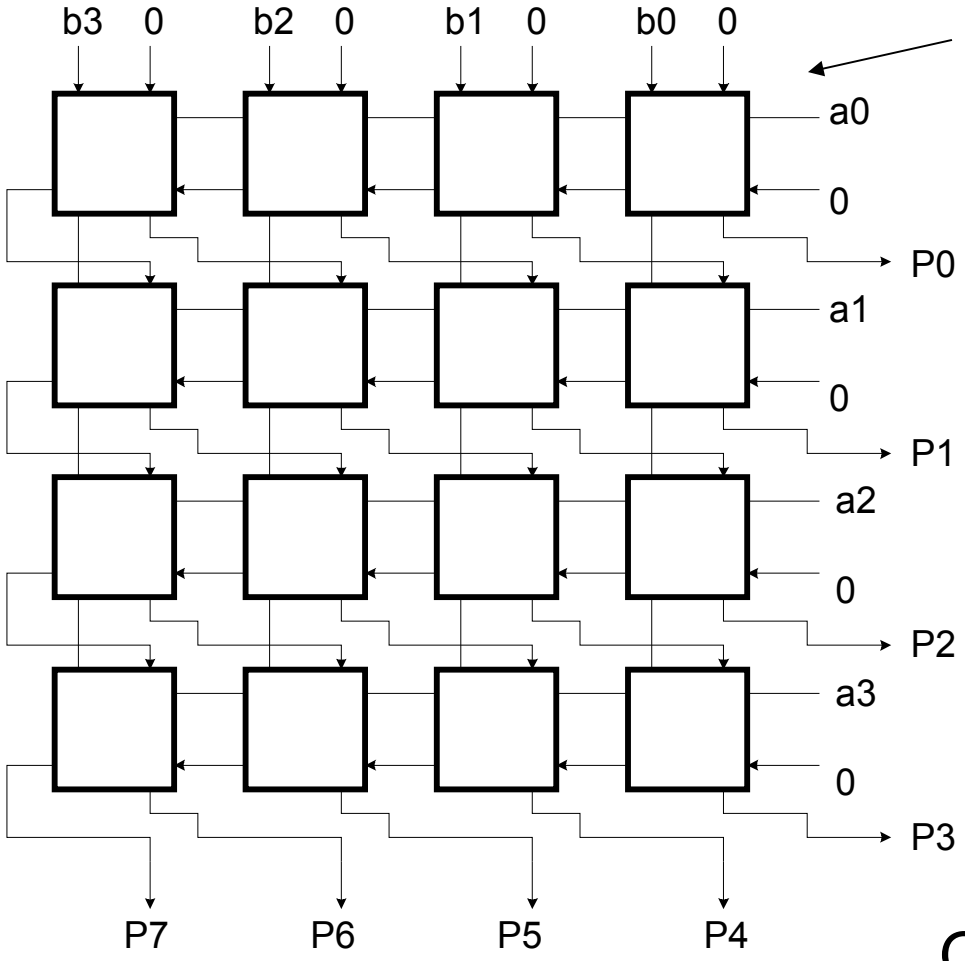
Multiplicador tipo array de 4 bits (Produto Z tem 8 bits)



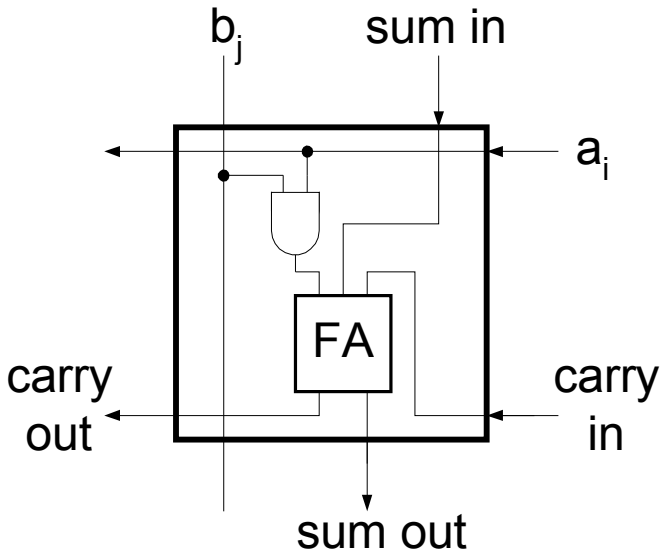
16 ANDS
4 Half-Adders
8 Full-Adders

Multiplicador tipo Array

Gera todos os “n” produtos parciais simultaneamente

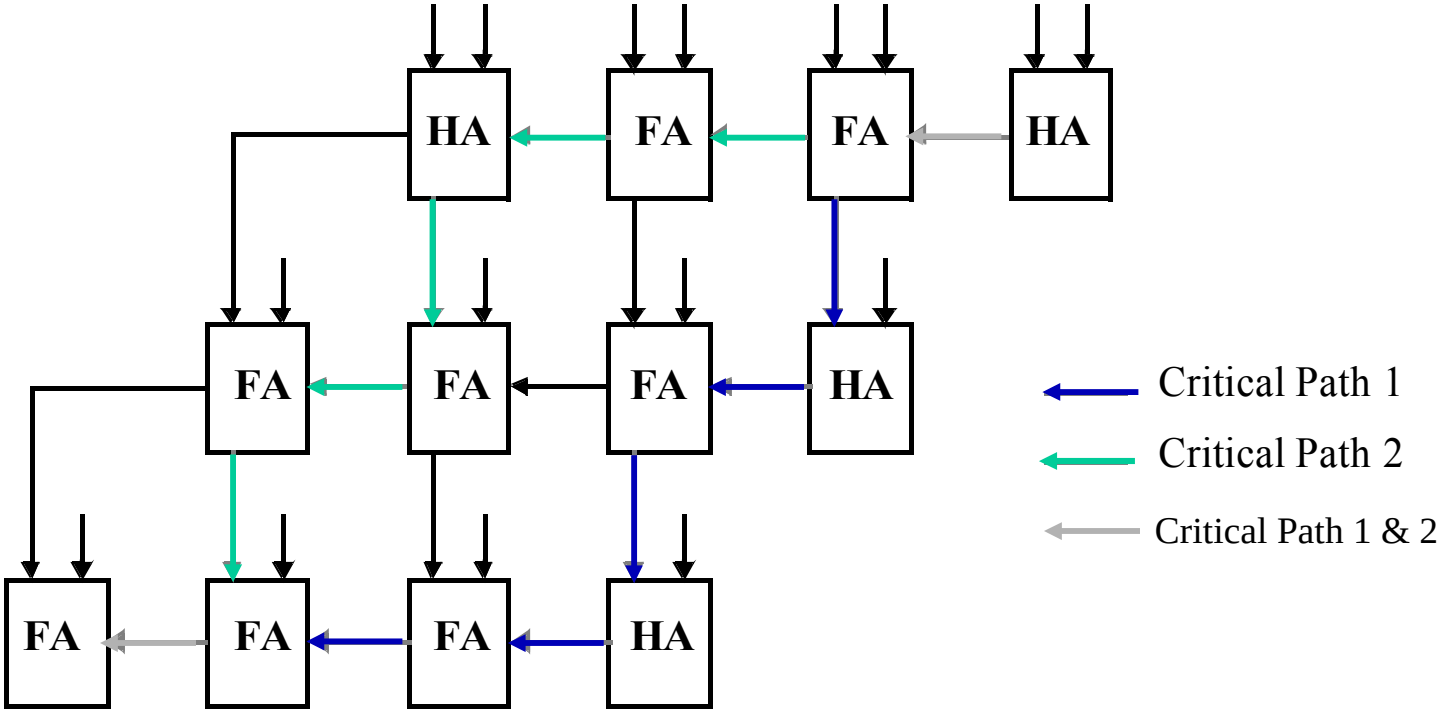


Each row: n-bit adder with AND gates



Qual é o caminho Crítico ?

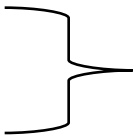
Multiplicador de M bits x N bits - Caminhos críticos



$$t_{mult} \approx [(M-1)+(N-2)]t_{carry}+(N-1)t_{sum}+(N-1)t_{and}$$

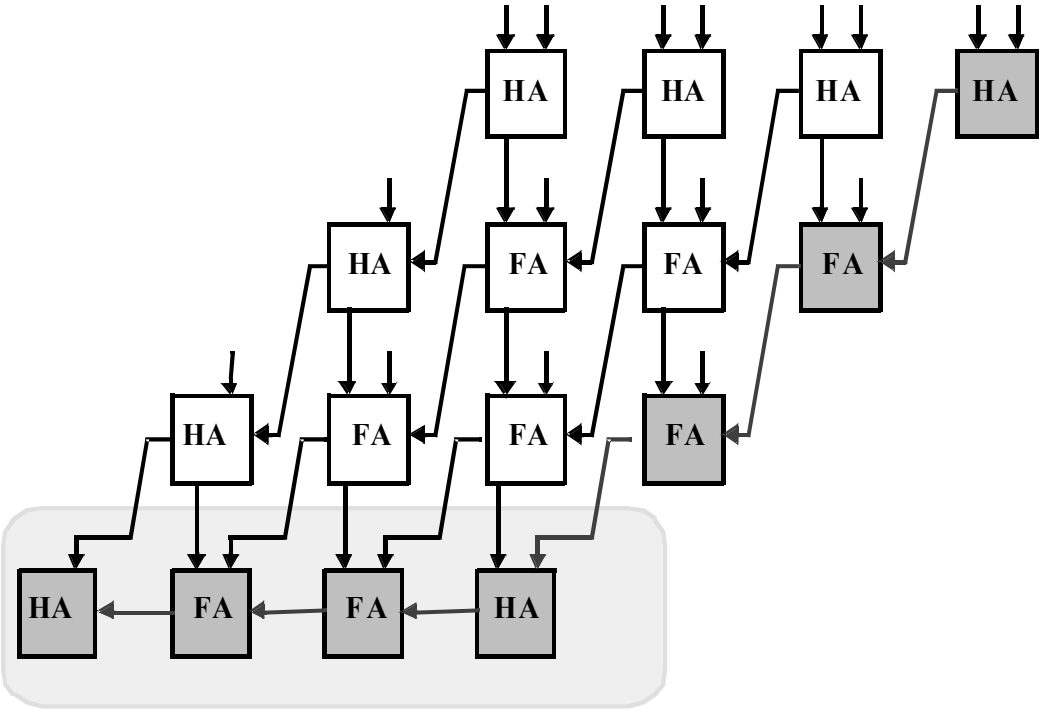
Multiplicador em VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity MULT is
   Port ( A : in  STD_LOGIC_VECTOR (3 downto 0);
          B : in  STD_LOGIC_VECTOR (3 downto 0);
          F : out  STD_LOGIC_VECTOR (7 downto 0));
end MULT;

architecture Behavioral of MULT is
Begin

process (A, B)
begin
F<= A*B;
end process;

end Behavioral;
```



multiplicador

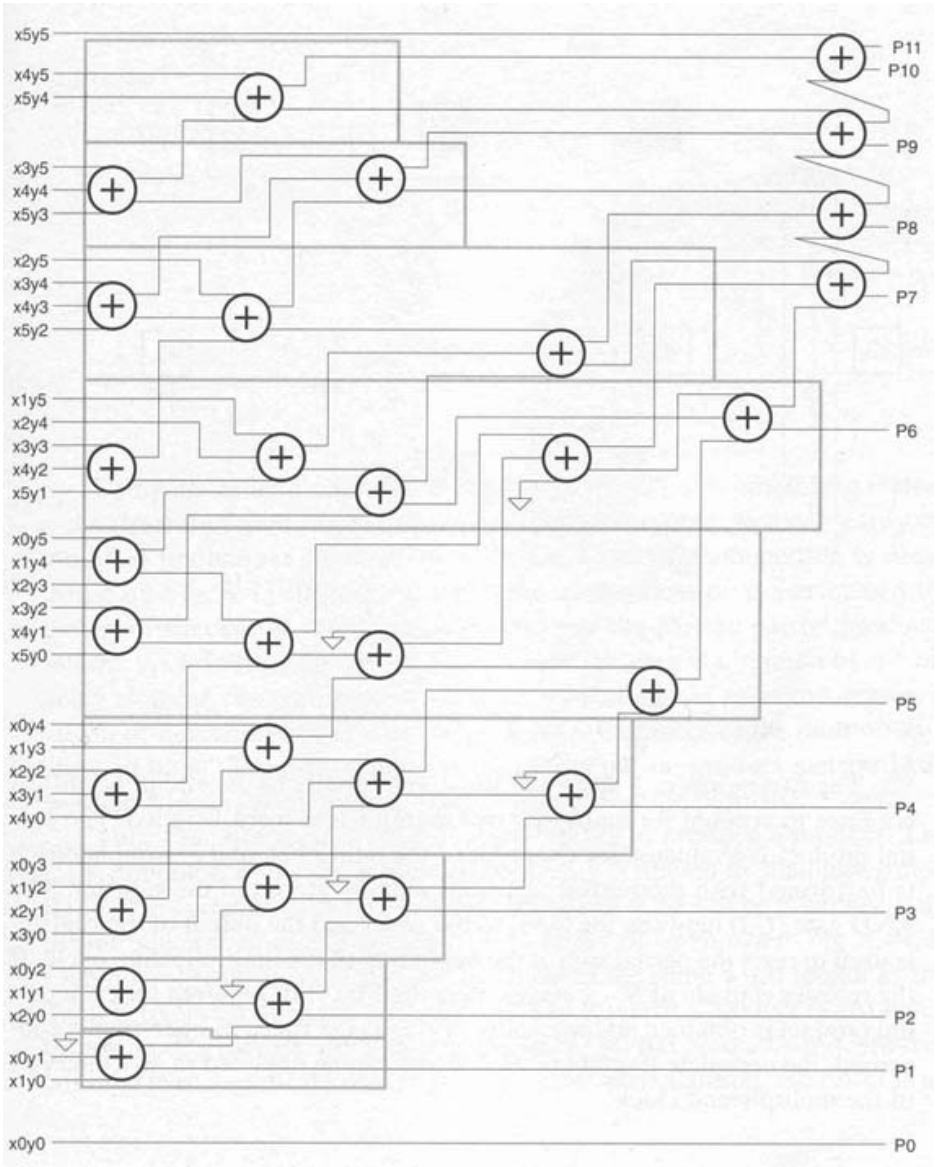
Multiplicador tipo carry-save



Vector Merging Adder

$$t_{mult} = (N-1)t_{carry} + (N-1)t_{and} + t_{merge}]$$

Multiplicador tipo *Wallace Tree*



Multiplicadores - Sumário

- **Otimização específica das Células de FA, HA, AND**
- **Identificar o caminho crítico combinacional**
- **Outras técnicas utilizadas na prática**
  - **Atraso logarítmico versus Linear (Árvore de Wallace para Multiplicação)**
  - **Recodificação do Multiplicando (Codificação de Booth)**
  - **Pipelinização do multiplicador**

Otimizações no nível sistema: Barramentos p/ Operando