INF01058

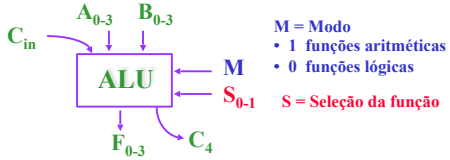
Circuitos Digitais

ULA Multifunção
Multiplicadores

Aula 13

Unidade Aritmética e Lógica - ULA

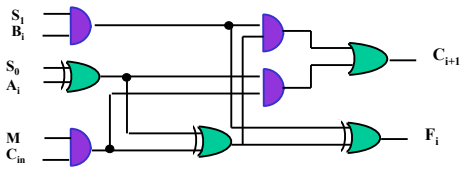
Exemplo de uma ULA Simples de 4 Bits - 12 Entradas, 5 Saídas



M = Modo
- 1 funções aritméticas
- 0 funções lógicas

S = Seleção da função

cada estágio



Funções Lógicas com M = 0

- O valor de C_{in} não importa
- C_{i+1} pode ser 0 ou 1, dependendo de A_i e B_i, mas isto não afeta o estágio seguinte

Portanto: F_i depende apenas de A_i, B_i e da seleção S_0 e S_1

S_1	S_0	F_i
0	0	A_i
0	1	$\overline{A_i}$
1	0	$A_i \oplus B_i$
1	1	$\overline{A_i \oplus B_i} = \overline{A_i} \oplus B_i$ (função equivalência)

Funções Aritméticas com M = 1 e C_{in} = 0

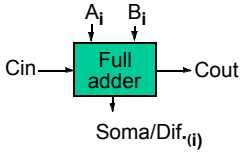
$M = 1$ faz com que C_i se propague através do estágio

S_1	S_0	F	
0	0	$F_i = A_i$	F = A
0	1	$F_i = \overline{A}_i$	F = $\overline{A}$ F = 1's de A
1	0	$F_i = B_i \oplus A_i \oplus C_i$	F = A mais B
1	1	$F_i = B_i \oplus \overline{A}_i \oplus C_i$	F = $\overline{A}$ mais B = B + 1's de A ou B – A em 1's

Funções Aritméticas com M = 1 e C_{in} = 1

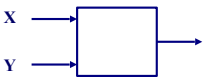
S_1	S_0	F		
0	0	$F_0 = A_0 \oplus 1$	$F_0 = A_0$ mais 1	F = A mais 1
0	1	$F_0 = \overline{A}_0 \oplus 1$	$F_0 = \overline{A_0}$ mais 1	F = $\overline{A}$ mais 1 = 2's de A
1	0	$F_0 = A_0 \oplus B_0 \oplus 1$	$F_0 = A_0$ mais B_0 mais 1	F = A mais B mais 1
1	1	$F_0 = \overline{A}_0 \oplus B_0 \oplus 1$	$F_0 = \overline{A_0}$ mais B_0 mais 1	F = $\overline{A}$ mais B mais 1 = B – A em 2's

Para M=1, S1=1 e $Cin_{(0)}$ =(0,1) a ULA converte-se em um bloco Somador/Subtrator



A	*B*	C_i	*S*	C_o	*Carry status*
0	0	0	0	0	delete
0	0	1	1	0	delete
0	1	0	1	0	propagate
0	1	1	0	1	propagate
1	0	0	1	0	propagate
1	0	1	0	1	propagate
1	1	0	0	1	generate
1	1	1	1	1	generate

Comparador

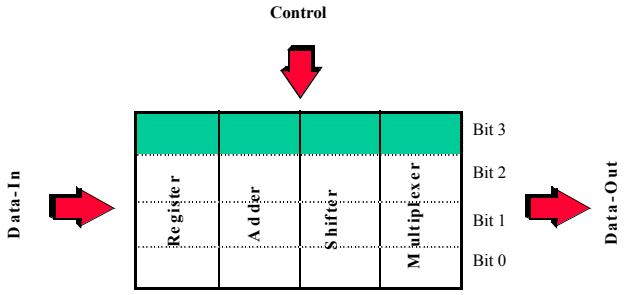


COMP = 1 se X = Y
= 0 se X ≠ Y

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

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

Planta Baixa tipo Bit-Slice



Bit-Slice **de mesma altura**
"Tiling" de elementos aritméticos e registradores

Multiplicador de 2 bits. Síntese de circuito combinacional

X_1	X_0	Y_1	Y_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

$P_3 = m_{15}$

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

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

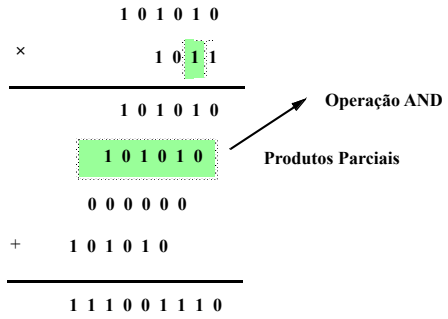
$P_0 = \sum m\ (5, 7, 13, 15)$

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

$P_1 = \overline{X}_1X_0Y_1 + \overline{Y}_1Y_0X_1 + X_1\overline{X}_0Y_0 + Y_1\overline{Y}_0X_0$

Uso de Produtos Parciais



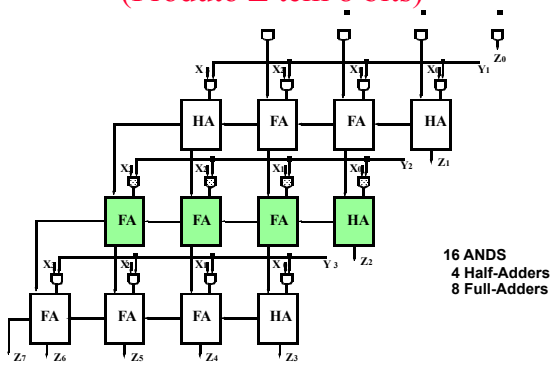
Multiplicador através de 'Rede de Somadores'

(A x B = P)

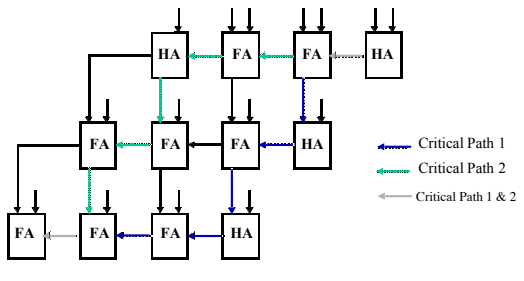
A ⇒	a3	a2	a1	a0			
B ⇒	b3	b2	b1	b0			
	a3.b0	a2.b0	a1.b0	a0.b0			
	a3.b1	a2.b1	a1.b1	a0.b1			
a3.b2	a2.b2	a1.b2	a0.b2				
a3.b3	a2.b3	a1.b3	a0.b3		+		
P7	P6	P5	P4	P3	P2	P1	P0

- Os produtos an.bn são obtidos através da função AND entre an e bn.
- As somas são realizadas com somadores completos (FA) e meio-somadores (HA)
- Há diversas formas de agrupar as somas, cuidando sempre para que os *carry-outs* ('vai um') gerados sejam considerados na coluna seguinte.

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

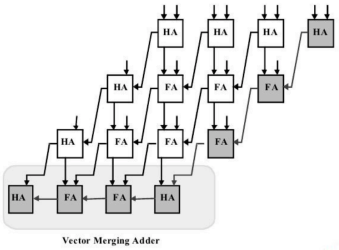


Multiplicador de M bits x N bits - Caminhos críticos (ex: 4x4)



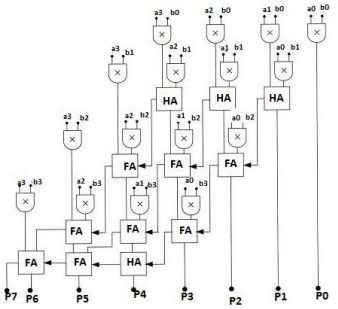
$$t_{mult} = [(M-1)+(N-2)]t_{carry} + (N-1)t_{sum} + (N-1)t_{and}$$

Multiplicador tipo carry-save (ex: 4x4)



Wrong!!!

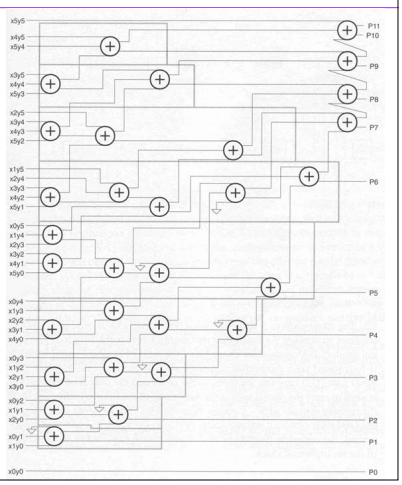
Correct:



Multiplicador *Wallace Tree*

- Faz as somas de cada coluna de produtos com uma árvore de somadores;
- Criar "camadas", níveis dessa árvore, até que haja somente dois operandos em cada coluna;
- Somar os dois "números" finais com um somador normal;

https://en.wikipedia.org/wiki/Wallace_tree



Multiplicadores - Sumário

- Otimização específica das Células de FA, HA, AND
- Identificar o caminhos críticos na lógica
- Otimizações nas somas:
  Atraso logarítmico com Árvores (Wallace)
- Otimizações no carry:
  Carry look-ahead, carry-select, …

Outras técnicas:

Recodificação do Multiplicando (Codificação de Booth)
Pipeline: conserva atraso, aumenta *throughput*
Multiplicadores por constantes