

“Se não der certo da primeira vez, chame de versão 1.0.”

Adders e Comparadores

Paulo Ricardo Lisboa de Almeida



Circuitos Aritméticos

Circuitos Aritméticos.

Circuitos combinacionais comumente encontrados dentro da **ALU**.

ALU - Arithmetic Logic Unit (Unidade Lógica e Aritmética).

Circuitos somadores, deslocadores de bits, operadores lógicos, ...

Como exemplo, veremos a implementação de um somador.

Assumindo que os valores somados são sempre positivos.

O somador ainda é válido para representações em complemento de dois, com alguns ajustes.

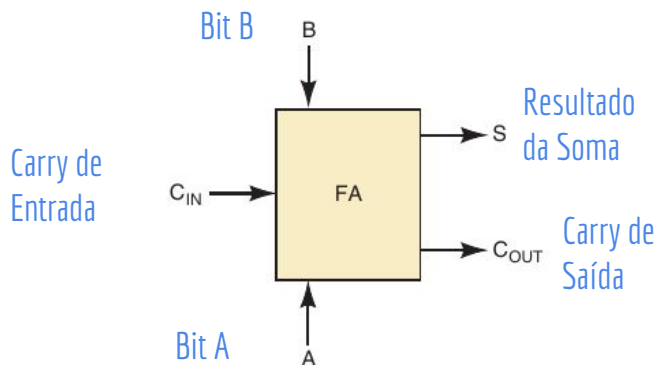
Full Adder

Full Adder (Somador Completo).

Circuito somador para um bit A com um bit B, considerando ainda um possível carry de entrada.

A soma de dois bits pode gerar um carry de saída.

Esse carry deve ser considerado no próximo adder, caso ele exista.



Faça você mesmo

Entradas			Saídas	
A	B	C_{in}	S	C_{out}
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Faça a tabela verdade para o Full Adder.

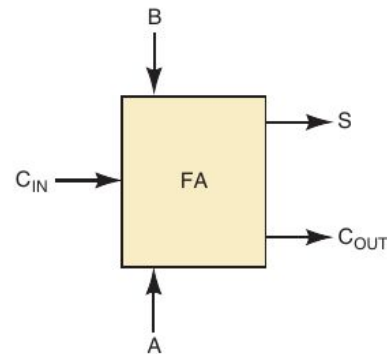
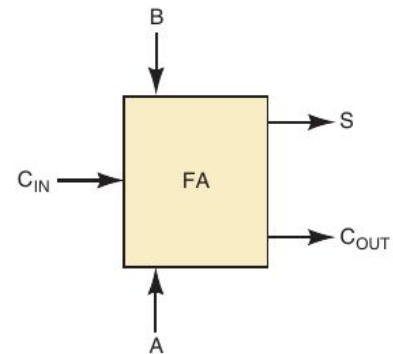


Tabela Verdade

Entradas			Saídas	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Full Adder - Soma

Existem duas saídas.

Vamos definir cada uma separadamente.

Começando por S (Resultado da Soma), são quatro linhas que geram 1.

Pela soma dos produtos (FND).

$$S = \bar{A}.\bar{B}.C_{in} + \bar{A}.B.\bar{C}_{in} + A.\bar{B}.\bar{C}_{in} + A.B.C_{in}$$

Entradas			Saída
A	B	C _{in}	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Full Adder - Soma

Existem duas saídas.

Vamos definir cada uma separadamente.

Começando por S (Resultado da Soma), temos quatro linhas que geram 1.

Pela soma dos produtos:

$$S = \bar{A}.\bar{B}.C_{in} + \bar{A}.B.\bar{C}_{in} + A.\bar{B}.\bar{C}_{in} + A.B.C_{in}$$

$$S = \bar{A}(\bar{B}.C_{in} + B.\bar{C}_{in}) + A(\bar{B}.\bar{C}_{in} + B.C_{in}) \leftarrow \text{Distributividade}$$

Utilizando uma tabela de equivalências lógicas.

$$\bar{B}.C_{in} + B.\bar{C}_{in} = B \oplus C_{in} \leftarrow \text{XOR.}$$

$$\bar{B}.\bar{C}_{in} + B.C_{in} = \overline{(B \oplus C_{in})} \leftarrow \text{XNOR.}$$

Entradas			Saída
A	B	C _{in}	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Full Adder - Soma

Dessa forma temos:

$$S = \bar{A}(B \oplus C_{in}) + A\overline{(B \oplus C_{in})}$$

Considere agora a mudança de variável $X = B \oplus C_{in}$

$$S = \bar{A}X + A\bar{X}$$

$S = A \oplus X$ <- Substituindo por um xor.

Retornando X para a expressão original.

$$S = A \oplus (B \oplus C_{in})$$

Entradas			Saída
A	B	C_{in}	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Full Adder - Carry Out

Calculando agora o Carry de Saída C_{out} .

$$C_{out} = \bar{A}.B.C_{in} + A.\bar{B}.C_{in} + A.B.\bar{C}_{in} + A.B.C_{in}$$

Entradas			Saída
A	B	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Full Adder - Carry Out

Calculando agora o Carry de Saída C_{out} .

$$C_{out} = \bar{A}.B.C_{in} + A.\bar{B}.C_{in} + A.B.\bar{C}_{in} + A.B.C_{in}$$

Podemos repetir qualquer mintermo sem alterar o resultado.

Lembre-se que $A+A=A$.

$$C_{out} = \bar{A}.B.C_{in} + A.\bar{B}.C_{in} + A.B.\bar{C}_{in} + A.B.C_{in} + A.B.C_{in} + A.B.C_{in}$$

Entradas			Saída
A	B	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Full Adder - Carry Out

Reordenando temos termos em comum em pares de mintermos.

$$C_{out} = \bar{A}.B.C_{in} + A.B.C_{in} + \bar{A}.\bar{B}.C_{in} + A.B.C_{in} + A.B.\bar{C}_{in} + A.B.C_{in}$$

$$C_{out} = B.C_{in}(\bar{A} + A) + A.C_{in}(\bar{B} + B) + A.B(\bar{C}_{in} + C_{in}) \leftarrow \text{Distributividade}$$

$$C_{out} = B.C_{in} + A.C_{in} + A.B \leftarrow \text{Complemento e elemento neutro}$$

Esse é o máximo que conseguimos simplificar.

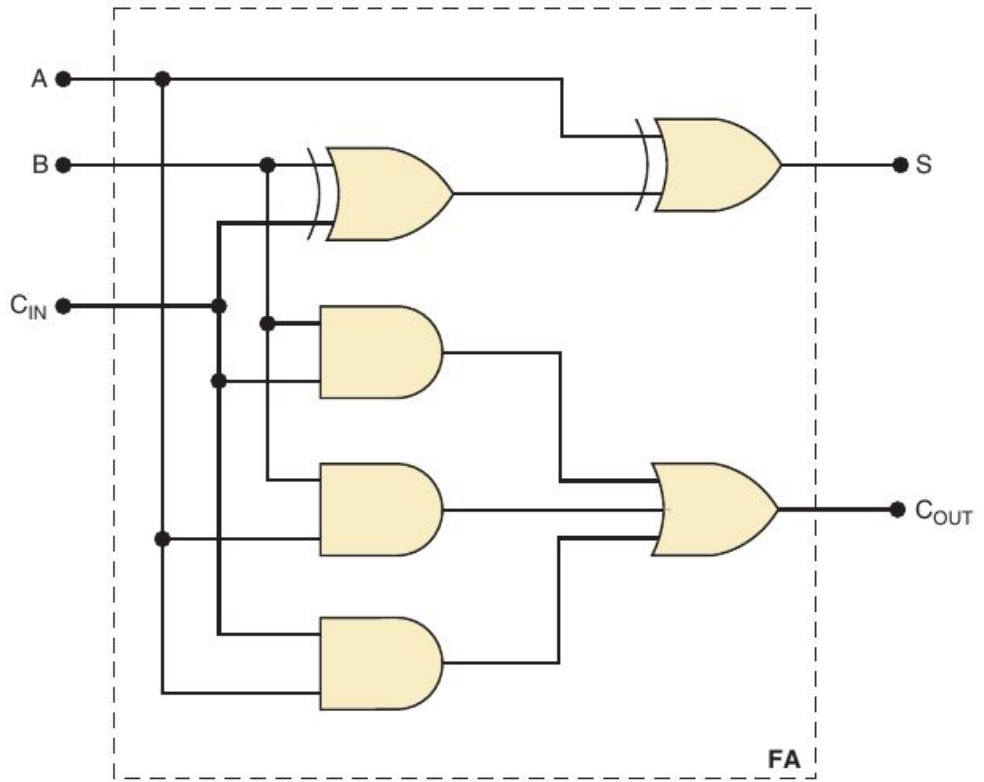
Entradas			Saída
A	B	C _{in}	C _{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Full Adder

$$S = A \oplus (B \oplus C_{in})$$

$$C_{out} = B.C_{in} + A.C_{in} + A.B$$

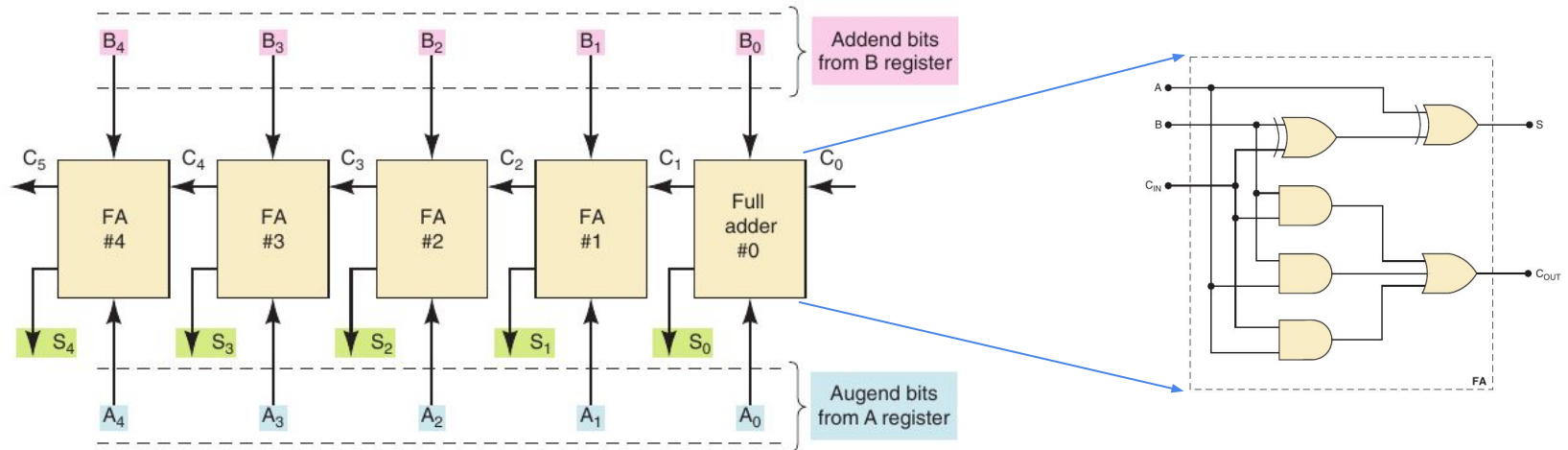
Obs.: Existem outras implementações que nos levam ao mesmo resultado.



Full Adder - Limitação

Veja a implementação (em blocos) de um somador para 5 bits.

Onde está a limitação (em tempo) deste circuito?



Full Adder - Limitação

Precisamos esperar os carries.

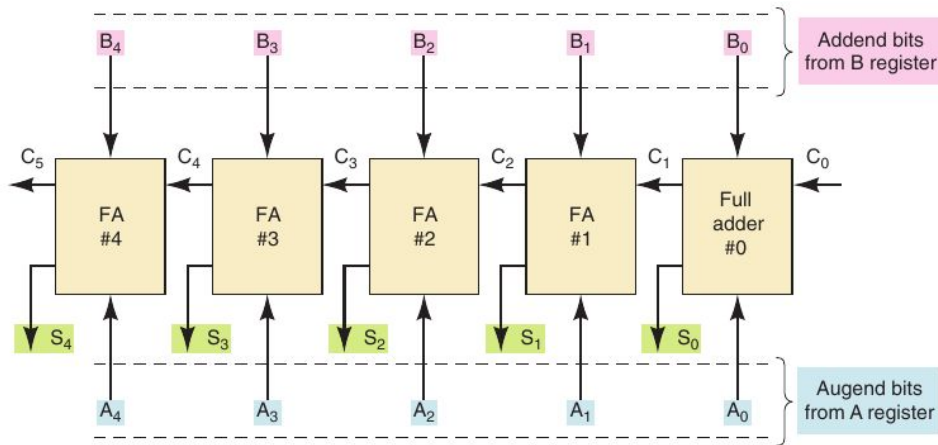
O carry C_1 precisa passar para FA#1, que vai gerar C_2 , que será necessário em FA#2, ...

Especialmente custoso em somadores grandes.

Exemplo: somadores de 64 bits.

Esse problema é conhecido como **carry propagation**.

Propagação do carry.



Carry Propagation

Circuitos de alta velocidade devem tratar esse problema.

Esquemas chamados *carry-lookahead* são adicionados.

“Predizem” se determinada operação vai gerar um carry ou não, sempre precisar passar por cada um dos Full Adders.

Requerem uma grande quantidade extra de portas lógicas para serem implementados.

Podem se tornar bastante complexos dependendo da implementação.

Por enquanto nos contentaremos com o Full Adder sem *carry lookahead*.

Implementações de circuitos de carry-lookahead podem ser encontradas na literatura da disciplina.

Circuitos Integrados

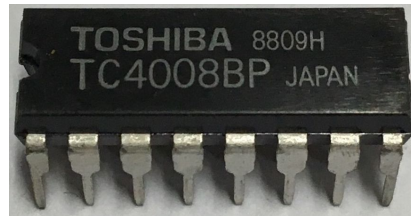
Existem diversos circuitos integrados que implementam o circuito discutido em aula.

Exemplos:

7483 → Somador completo de 4 bits.



CD4008 → Somador completo de 4 bits com carry look ahead.



Dica

Boa parte do conteúdo desta aula foi baseado em Tocci et al (2016).

Em Bignell e Donovan (2010) o mesmo circuito é criado utilizando uma abordagem diferente.

No livro existe também a implementação do half-adder.

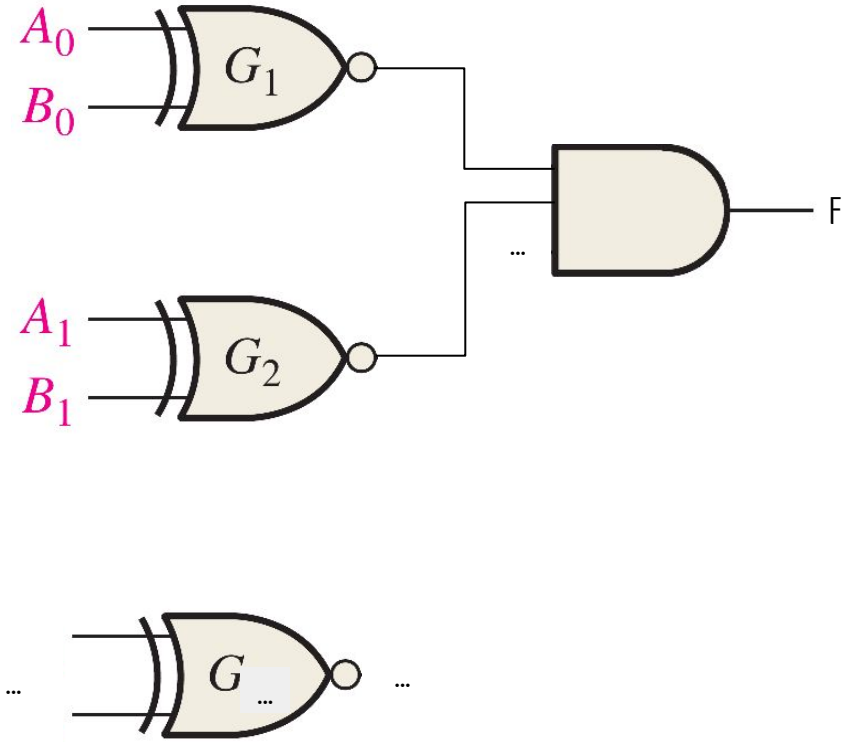
Comparando

Considere dois valores A e B, compostos dos bits:

$A_3 A_2 A_1 A_0$ e $B_3 B_2 B_1 B_0$.

Como criar um circuito que compara se $A=B$?

Comparando



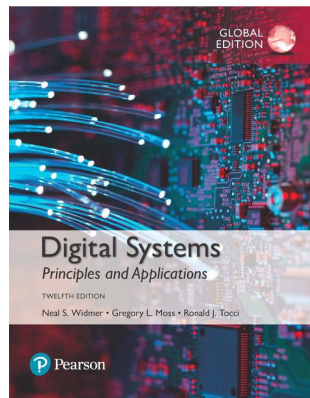
X	Y	$\overline{X \oplus Y}$
0	0	1
0	1	0
1	0	0
1	1	1

Exercícios

1. Considere o somador de 5 bits dos slides anteriores. Para o bit menos significativo, que não recebe carry, crie um circuito de soma que **não recebe** C_{in} para o cálculo. O nome desse circuito é *half-adder*.
 - a. Tente definir você mesmo esse circuito.
 - i. A implementação está disponível na literatura.
 - b. Efetue testes no circuito, aplicando algumas entradas e verificando seus resultados.
2. Considere dois valores inteiros maiores ou iguais a zero de 4 bits, A e B. Como criar um comparador que possui uma saída $F=1$ se $A>B$, e $F=0$ caso contrário? Faça uma tabela verdade e crie o circuito.

Referências

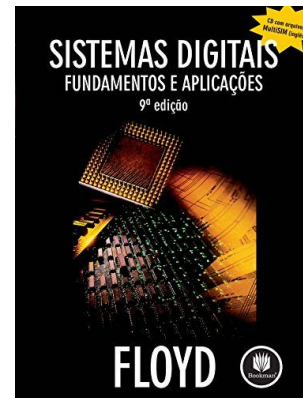
Ronald J. Tocci, Gregory L. Moss, Neal S. Widmer. Sistemas digitais. 10a ed. 2017.



James Bignell, Robert Donovan. Eletrônica digital. Cengage Do Brasil, 2010.



Thomas Floyd. Widmer. Sistemas Digitais: Fundamentos e Aplicações. 2009.



Licença

Esta obra está licenciada com uma Licença [Creative Commons Atribuição 4.0 Internacional](https://creativecommons.org/licenses/by/4.0/).