Sistemas Digitais

Somadores

Aula 04

Prof. Leandro Nogueira Couto
UFU – Monte Carmelo
05/2013

E **AND**		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	**Função E:** Assume 1 quando todas as variáveis forem 1 e 0 nos outros casos.	**S=A.B**
OU **OR**		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	**Função OU:** Assume 0 quando todas as variáveis forem 0 e 1 nos outros casos.	**S=A+B**
NÃO **NOT**		<table><tr><th>A</th><th>S</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	**Função NÃO:** Inverte a variável aplicada à sua entrada.	**S=$\overline{A}$**
NE **NAND**		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	**Função NE:** Inverso da função E.	**S=$\overline{(A.B)}$**
NOU **NOR**		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	**Função NOU:** Inverso da função OU.	**S=$\overline{(A+B)}$**
OU EXCLUSIVO		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	**Função OU Exclusivo:** Assume 1 quando as variáveis assumirem valorem diferentes entre si.	**S=A⊕B** **S= $\overline{A}$.B + A.$\overline{B}$**
COINCIDÊNCIA		<table><tr><th>A</th><th>B</th><th>S</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	**Função Coincidência:** Assume 1 quando houver coincidência entre os valores das	**S= A⊙B** **S= $\overline{A}$.$\overline{B}$ + A.B**

Você já pensou que **toda função** de um computador se encaixa em uma das categorias:

- Processamento de dados
- Armazenamento de dados
- Movimentação de dados
- Controle

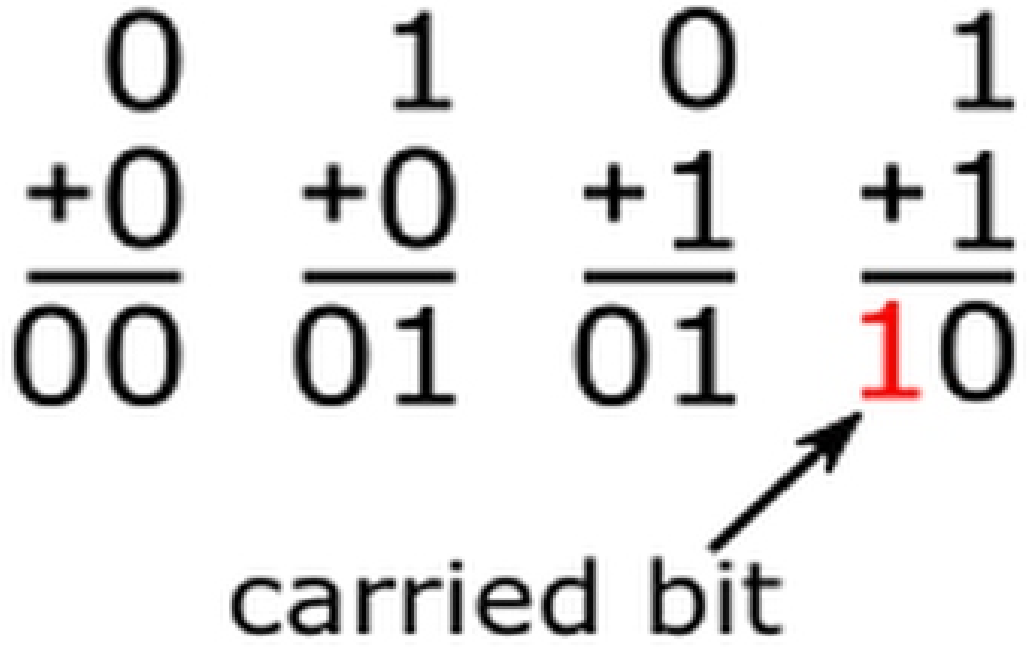
As 5 partes básicas de um computador: Entrada, Saída, Memória, Unidade Lógica Aritmética e Unidade de Controle

Somador

Vamos relembrar os resultados de uma soma de 1 bit com outro

Somador

Vamos relembrar os resultados de uma soma de 1 bit com outro



Somador

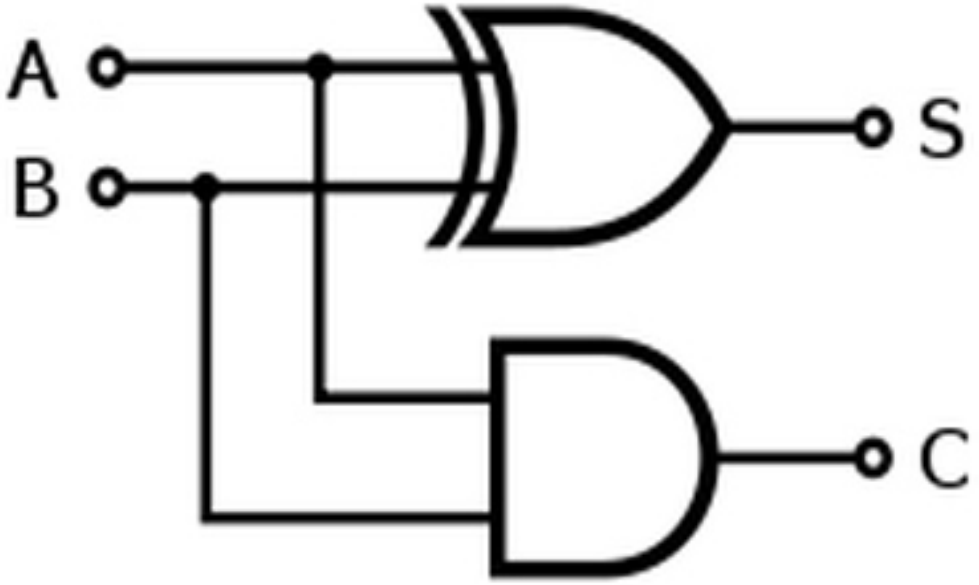
Como podemos fazer isso usando portas lógicas?

- Vamos tentar fazer a tabela verdade dos bits de saída
- Como podemos escrever um circuito para cada bit de saída?

Somador

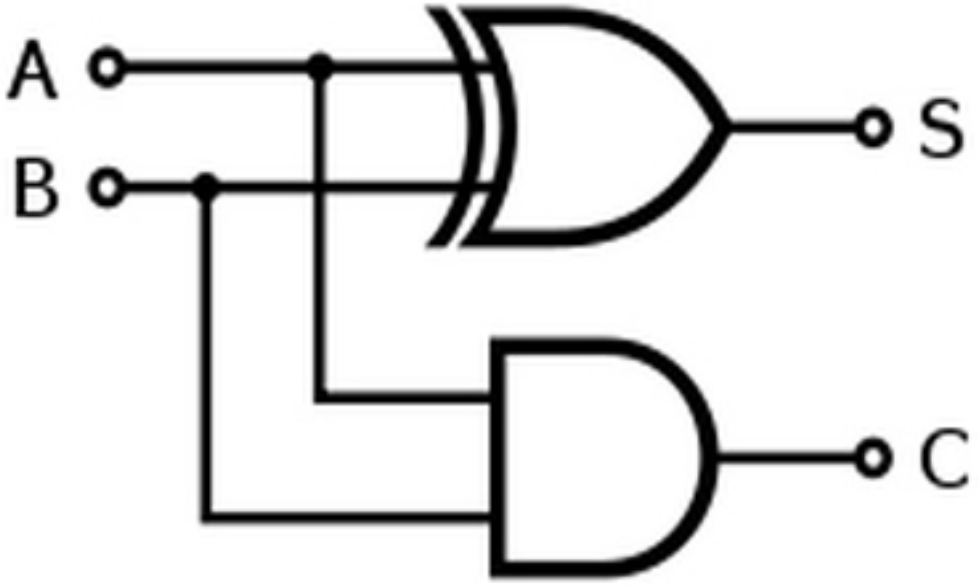
Como podemos fazer isso usando portas lógicas?

- Vamos tentar fazer a tabela verdade dos bits de saída
- Como podemos escrever um circuito para cada bit de saída?



Somador

- Chamamos o valor resultante de S (de saída) e o “vai um” de carry
- Esse circuito é chamado **meio-somador ou half-adder**. Porque?

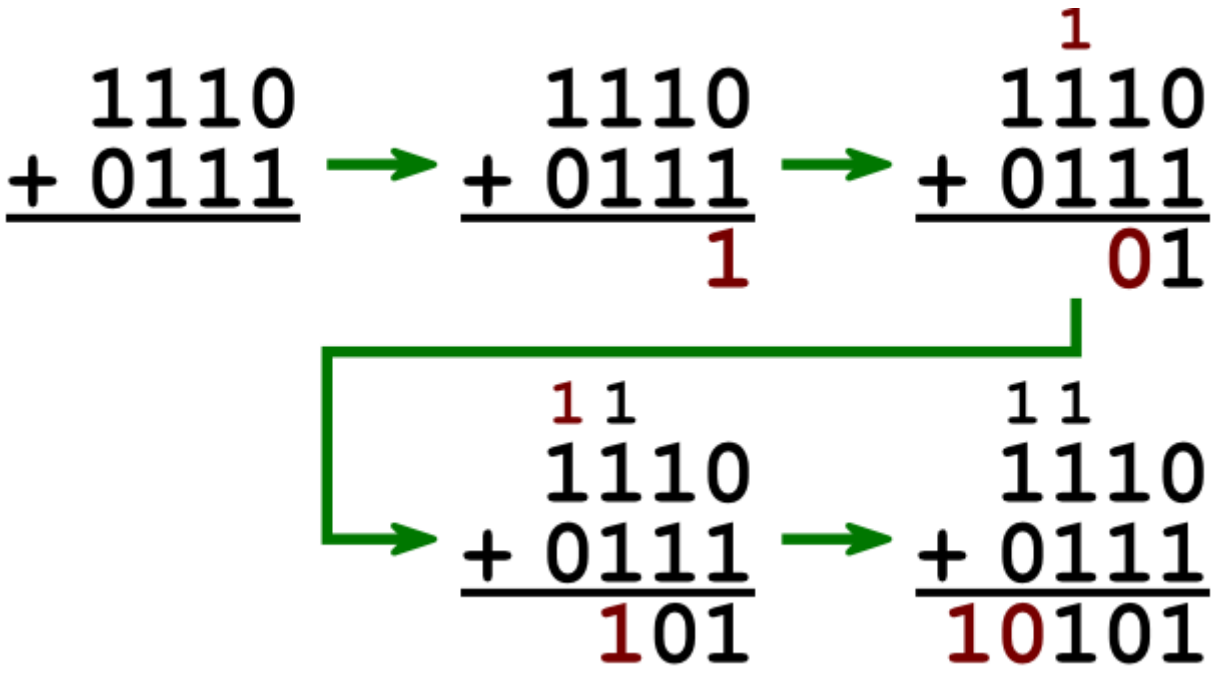


Somador

- Para podermos fazer um somador para vários bits, o que devemos fazer?
- Encadear vários somadores!
- Para isso, precisamos de algumas modificações no nosso **meio-somador**

Somador

- Para podermos fazer um somador para vários bits, o que devemos fazer?
- Encadear vários somadores!
- Para isso, precisamos de algumas modificações no nosso **meio-somador**



Somador

- Vamos escrever a tabela-verdade para um somador, agora considerando como entradas os bits A e B, e além disso se existe ou não carry (vai-um ou transporte) vindo de uma operação anterior

Somador

- Vamos escrever a tabela-verdade para um somador, agora considerando como entradas os bits A e B, e além disso se existe ou não carry (vai-um) vindo de uma operação anterior
- Note que:

  Te = Cin (transporte entrada)

  Ts = Cout (transporte saída)

A	B	T_E	S	T_S
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

Somador

- Aplicando mapa de Karnaugh, veremos que (simplificando)

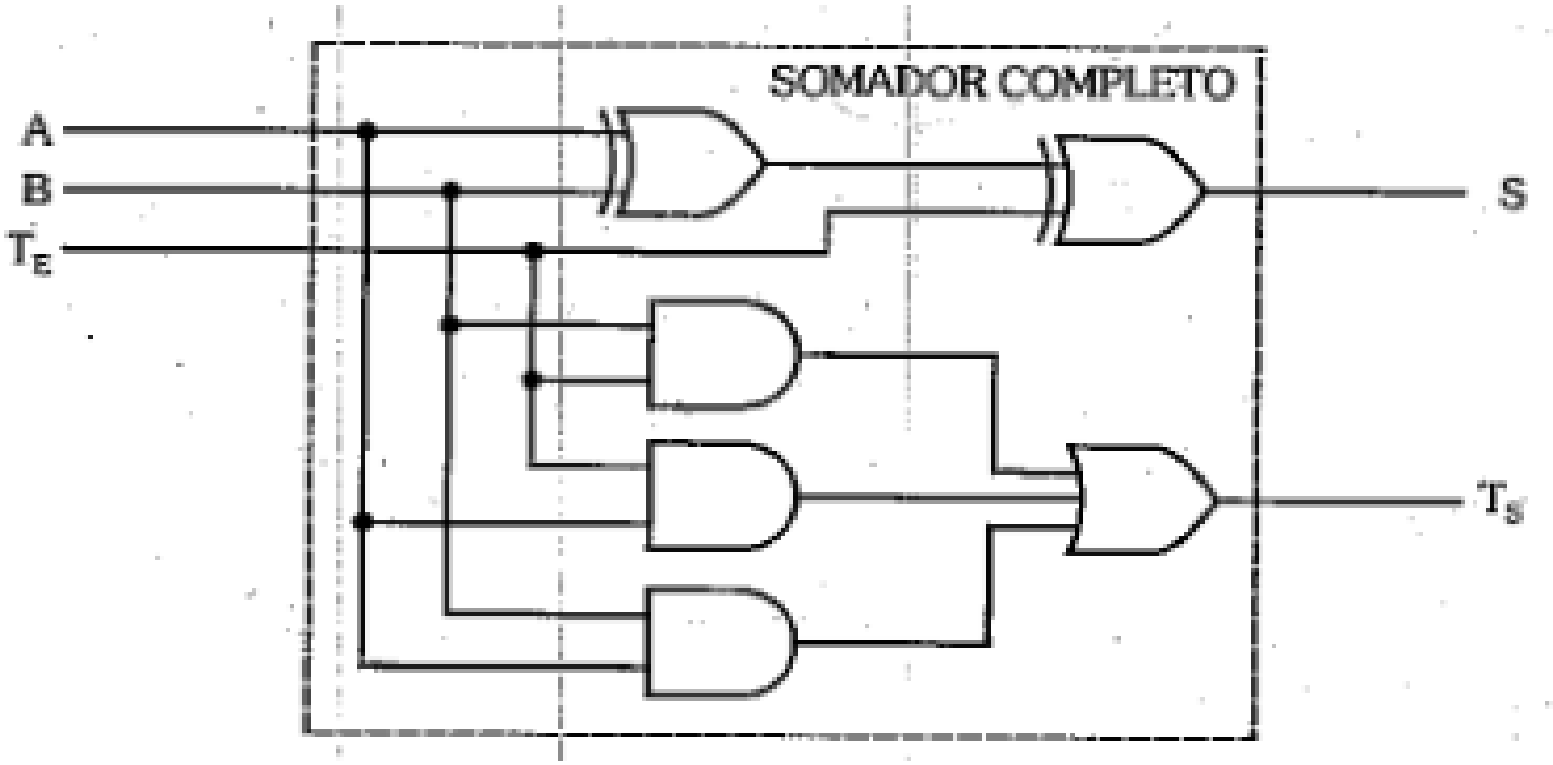
  $S = A \oplus B \oplus Cin$

- E veremos que

  $Cout = BCin + ACin + AB$

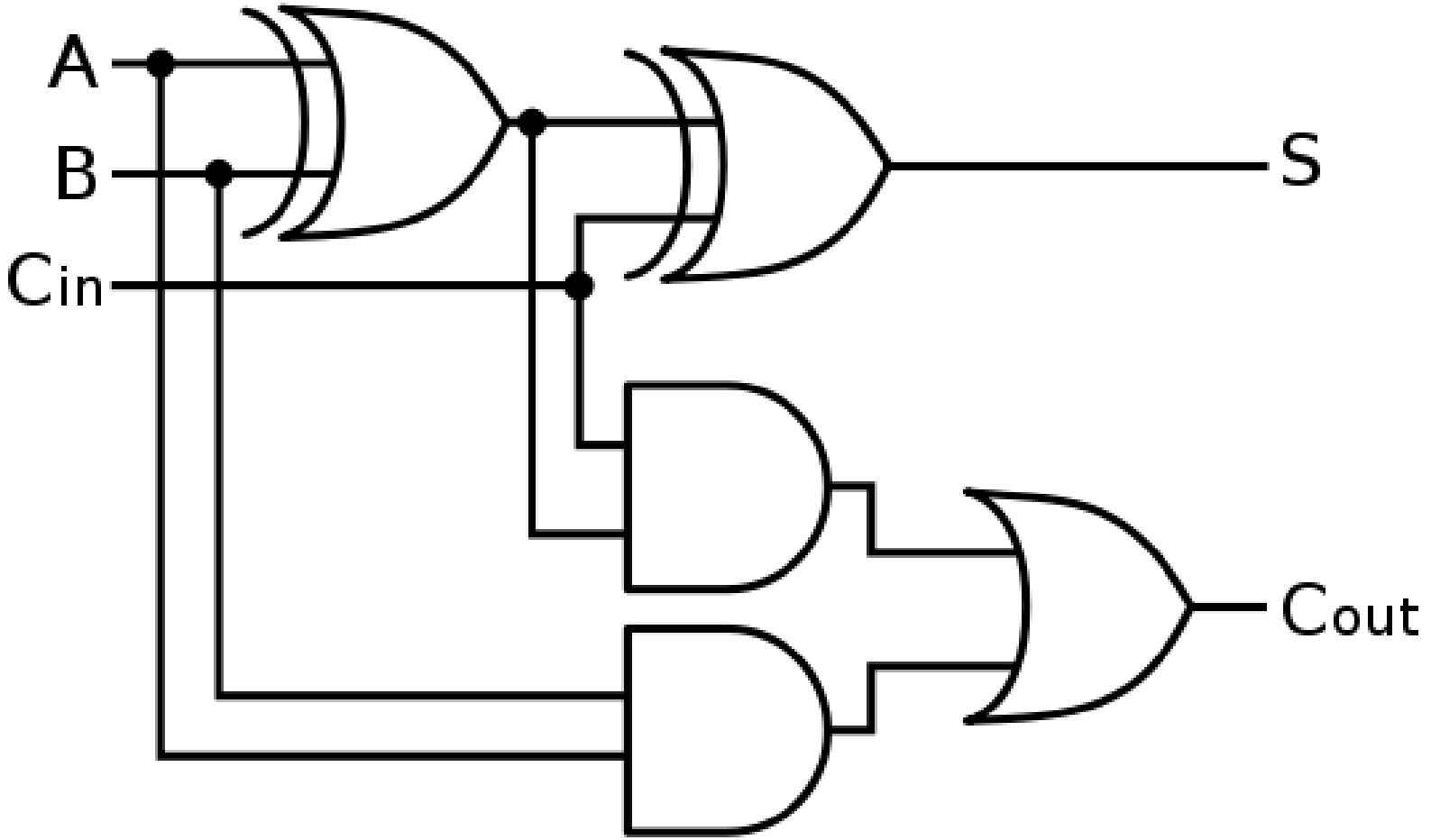
Somador

- Construindo o circuito para as fórmulas, temos o **somador-completo** ou **full-adder** (pois considera o carry)



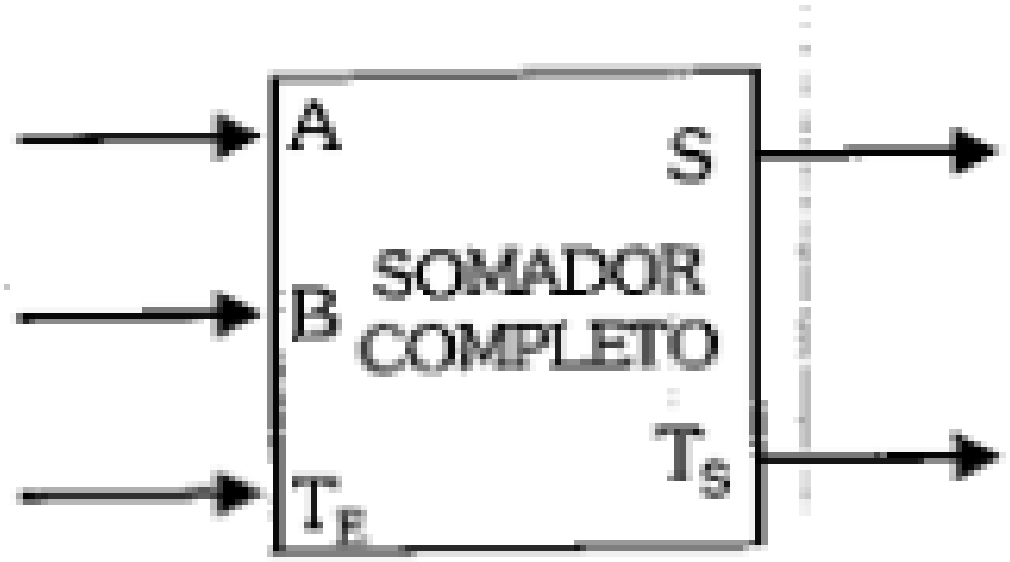
Somador

- Outra maneira de escrever o **full-adder**:



Somador

- Para evitar desenhar toda vez o circuito, usamos uma “caixinha” para representá-lo

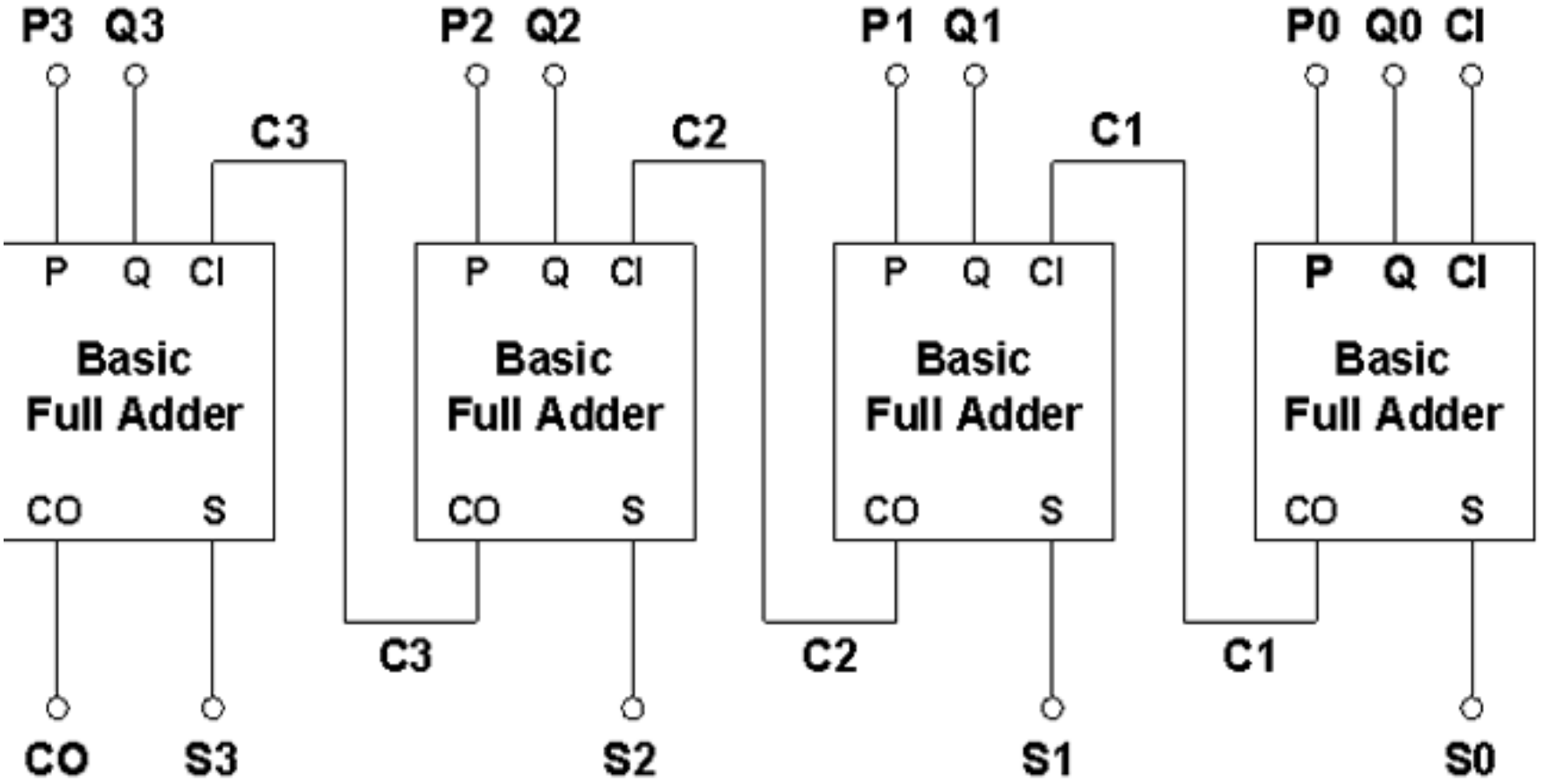


Somador

- Como usar o somador-completo para somar 2 números de 4 bits?

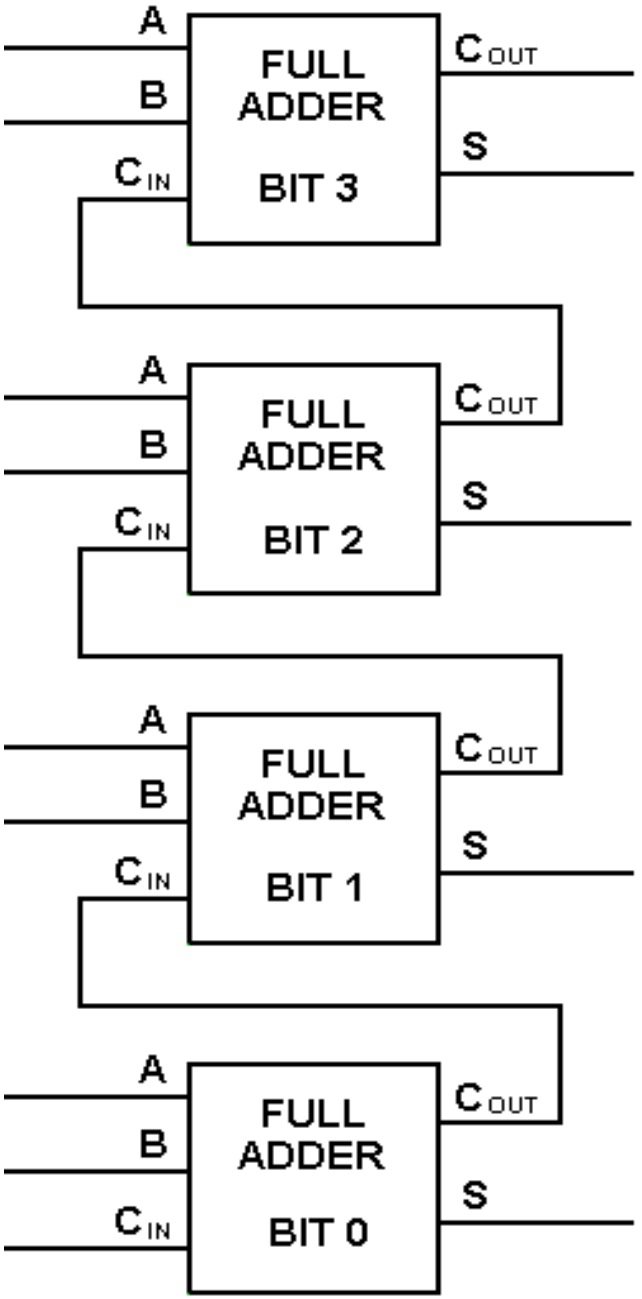
Somador

- Como usar o somador-completo para somar 2 números de 4 bits?



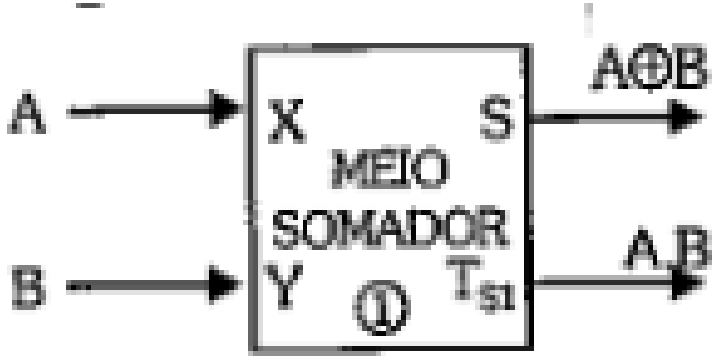
Somador

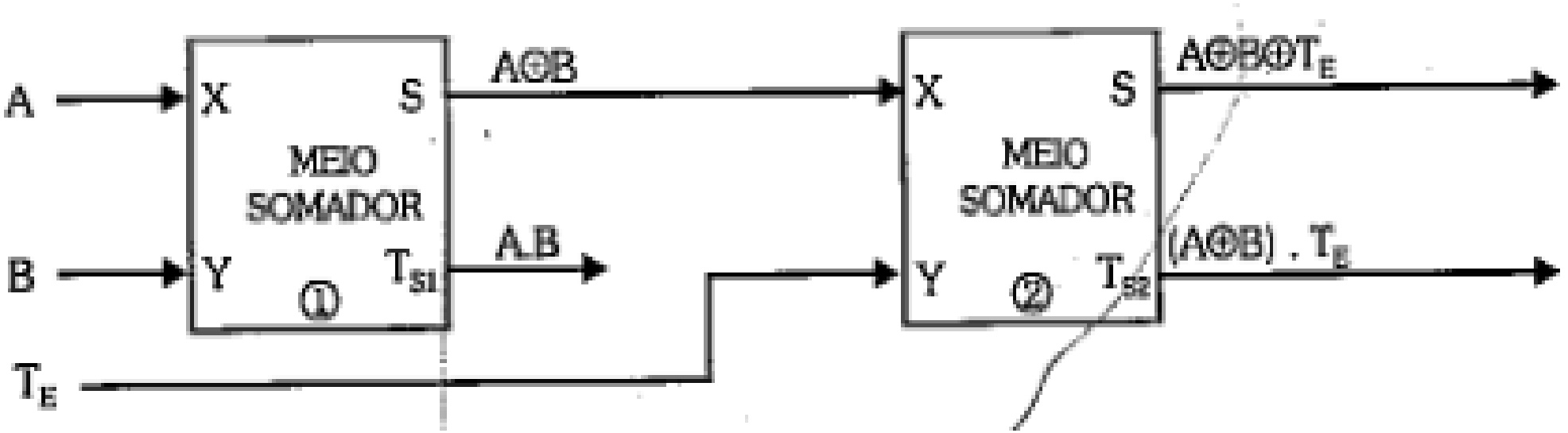
- Outra forma de desenhar



Somador

- Note que podemos fazer um **somador-completo** usando 2 **meio-somadores**
- Essa equivalência é obtida com um pouco de fatoração





Subtrator

- Podemos usar o mesmo raciocínio para fazer um circuito subtrator
- Como fica a tabela-verdade da subtração?
- Lembrando das regras:

	Minuend		Subtrahend		Difference	Borrow out
Rule 1	0	–	0	=	0	
Rule 2	0	–	1	=	1	and borrow 1
Rule 3	1	–	0	=	1	
Rule 4	1	–	1	=	0	

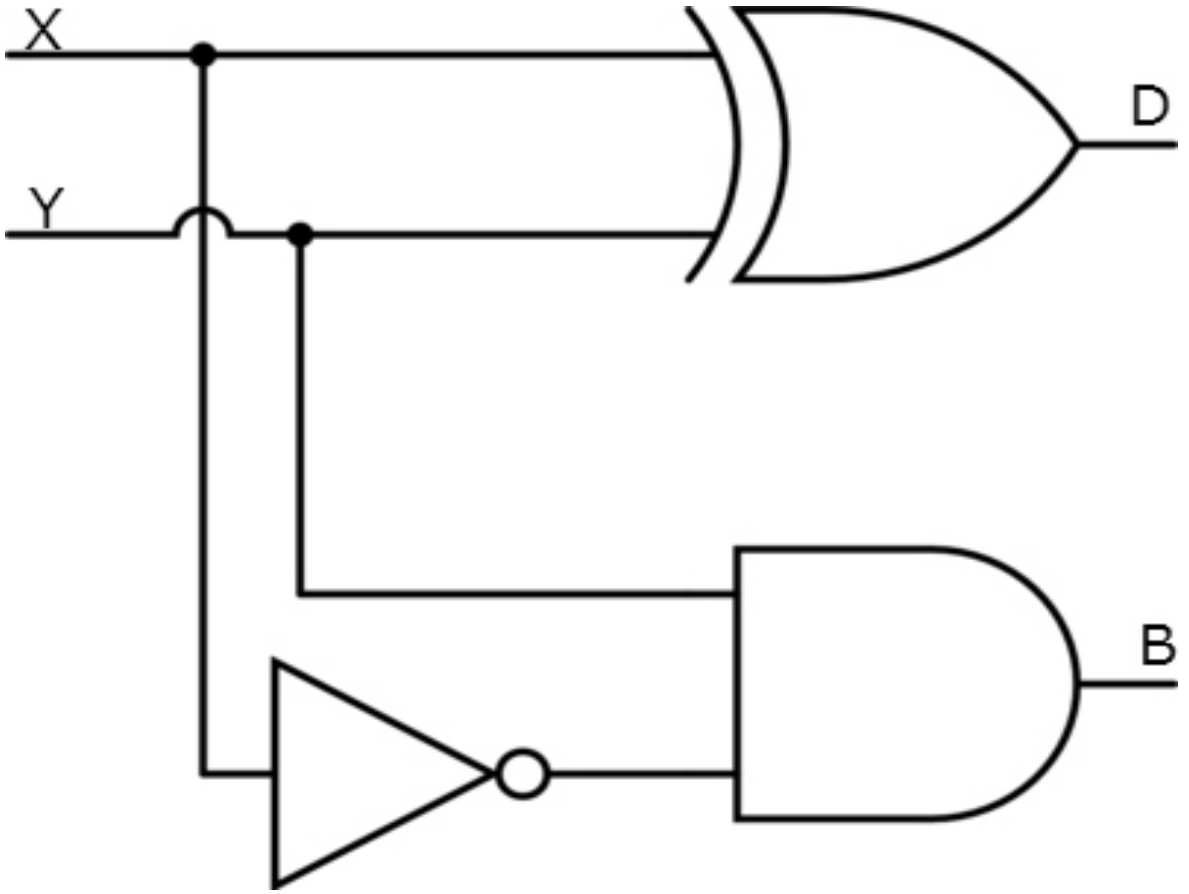
Subtrator

- Tabela-Verdade:

Half Subtractor-Truth Table			
Input		Output	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0
www.flintgroups.com			

Subtrator

- Circuito Lógico do **meio-subtrator**:



Subtrator

- E o **subtrator-completo**?

Subtrator

- E o **subtrator-completo**?
- Tabela-Verdade:

A	B	T_E	S	T_S
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Subtrator

- Aplicando mapa de Karnaugh, veremos que (simplificando)

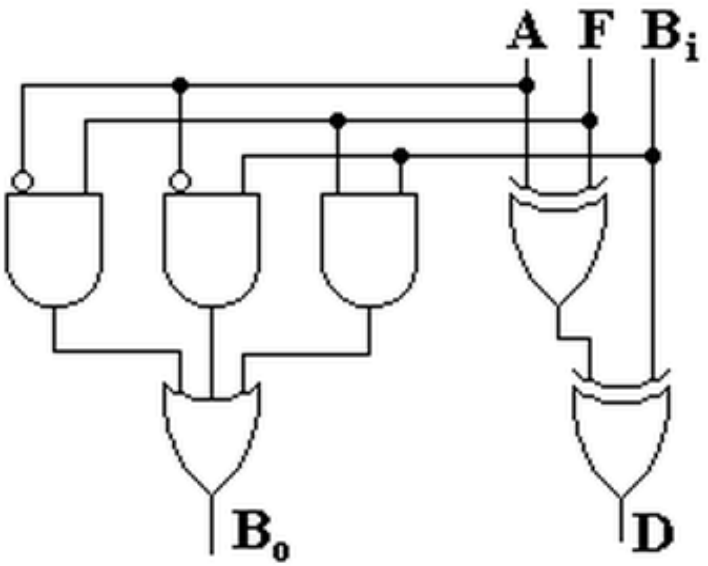
  $S = A \oplus B \oplus Cin$

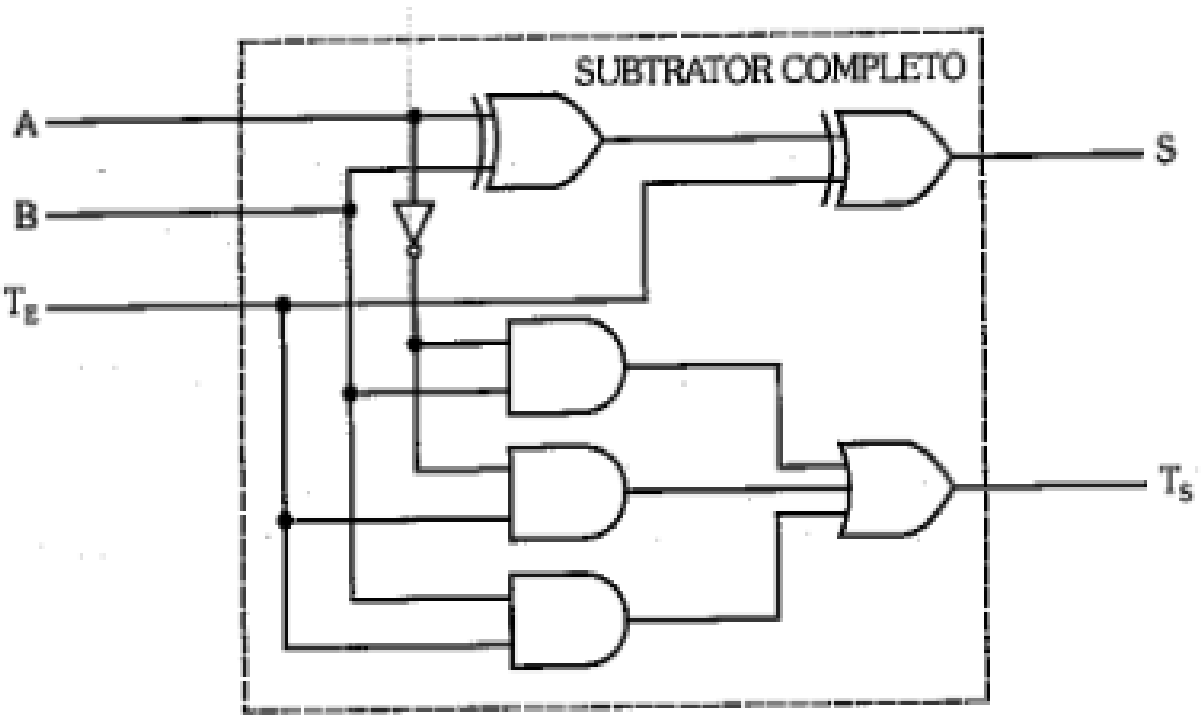
- E veremos que

  $Cout = BCin + (\sim A)Cin + (\sim A)B$

Subtrator

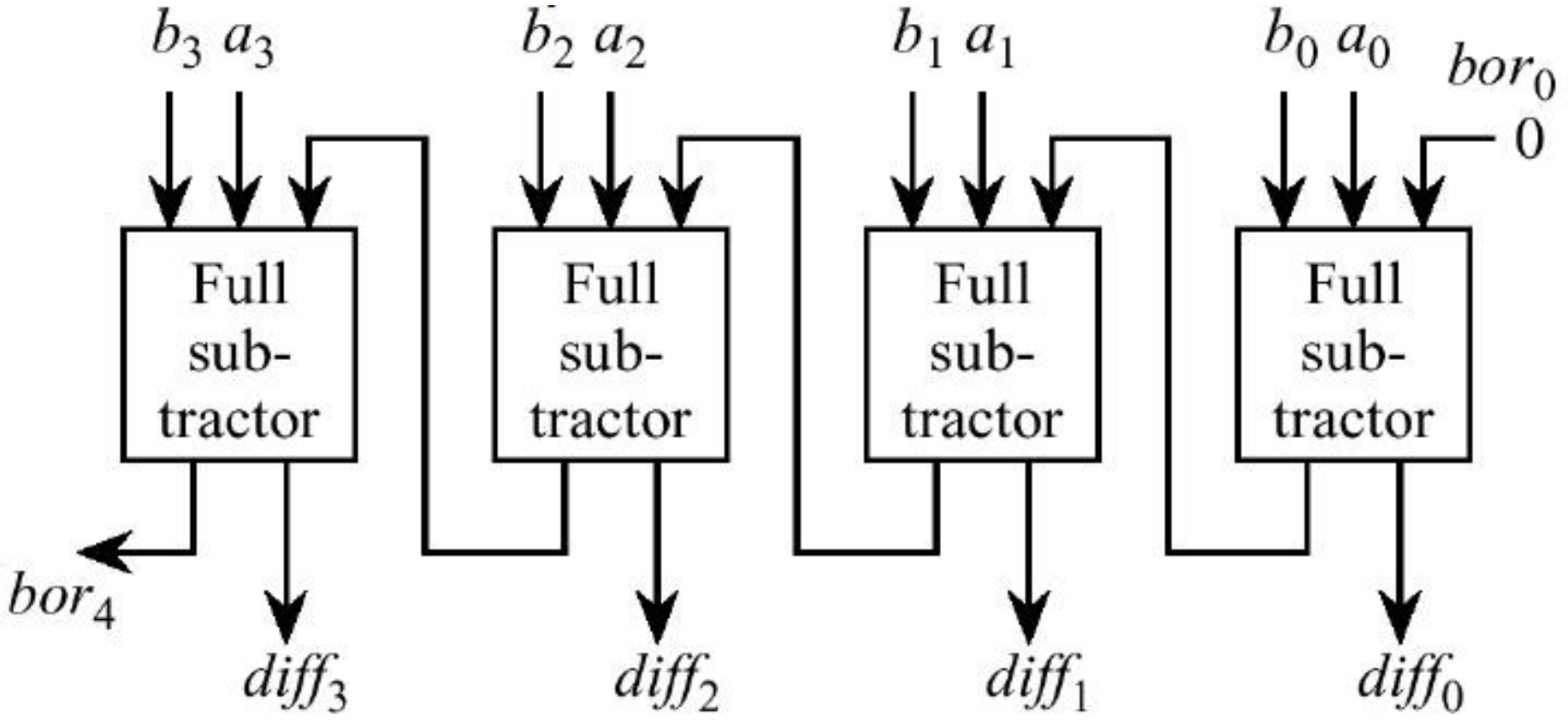
- Da mesma forma que no somador, achamos o circuito do **subtrator-completo**:





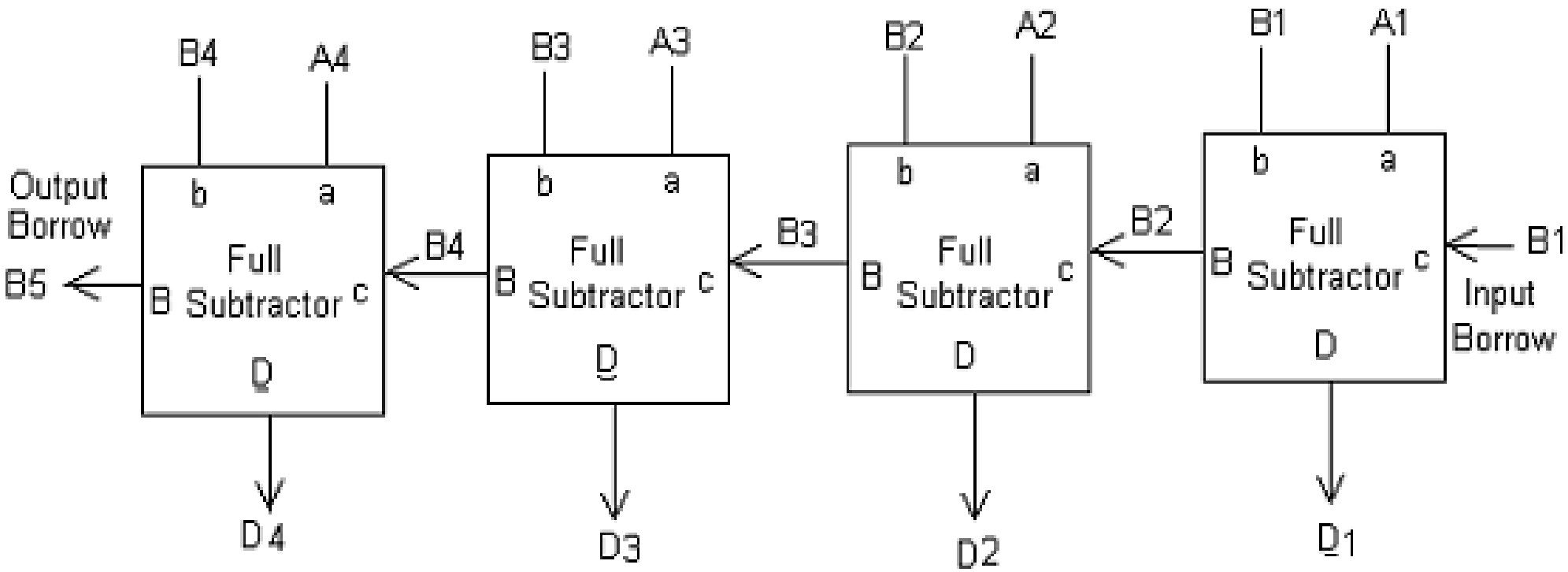
Subtrator

- Para subtrair 2 números de 4 bits:



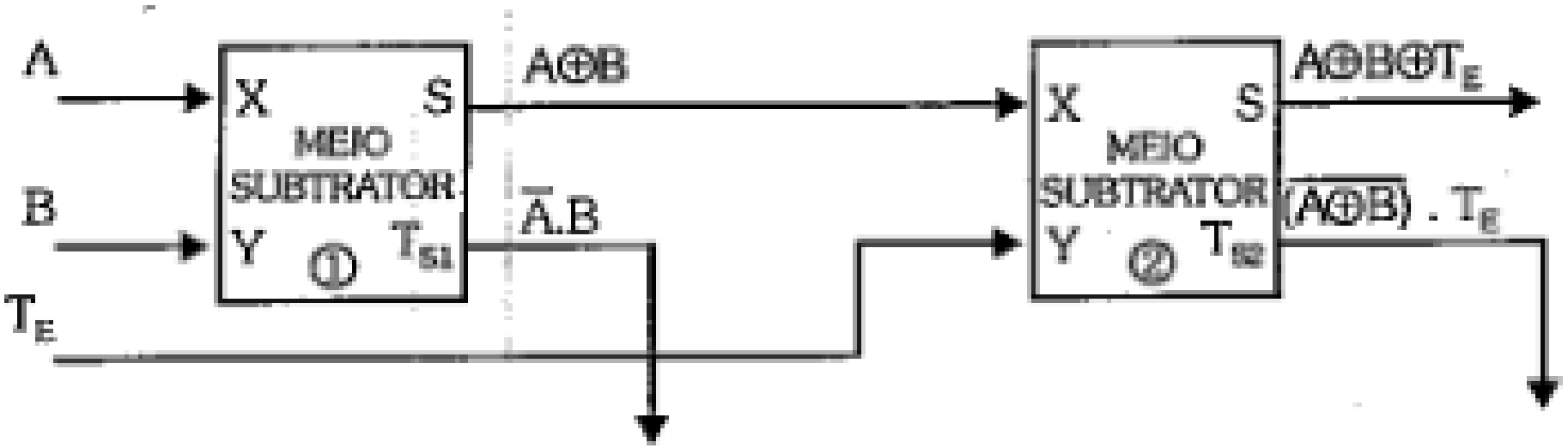
Subtrator

- Para subtrair 2 números de 4 bits:



Subtrator

- Da mesma forma podemos tirar um subtrator-completo a partir de 2 meio-subtratores



Somador/Subtrator

- Podemos ver que somador e subtrator são parecidos. Podemos fazer um circuito que faz as duas coisas?
- Quero ter uma nova entrada, um **bit de controle** que quando for 0 faz o circuito efetuar soma e quando for 1 efetua subtração

Somador/Subtrator

- Nova tabela-verdade, combinando somador e subtrator

M	A	B	T_E	S	T_S	
0	0	0	0	0	0	Soma Completa (M = 0)
0	0	0	1	1	0	
0	0	1	0	1	0	
0	0	1	1	0	1	
0	1	0	0	1	0	
0	1	0	1	0	1	
0	1	1	0	0	1	
0	1	1	1	1	1	
1	0	0	0	0	0	Subtração Completa (M = 1)
1	0	0	1	1	1	
1	0	1	0	1	1	
1	0	1	1	0	1	
1	1	0	0	1	0	
1	1	0	1	0	0	
1	1	1	0	0	0	
1	1	1	1	1	1	

Somador/Subtrator

- Vamos fazer o diagrama de Karnaugh para S e T?

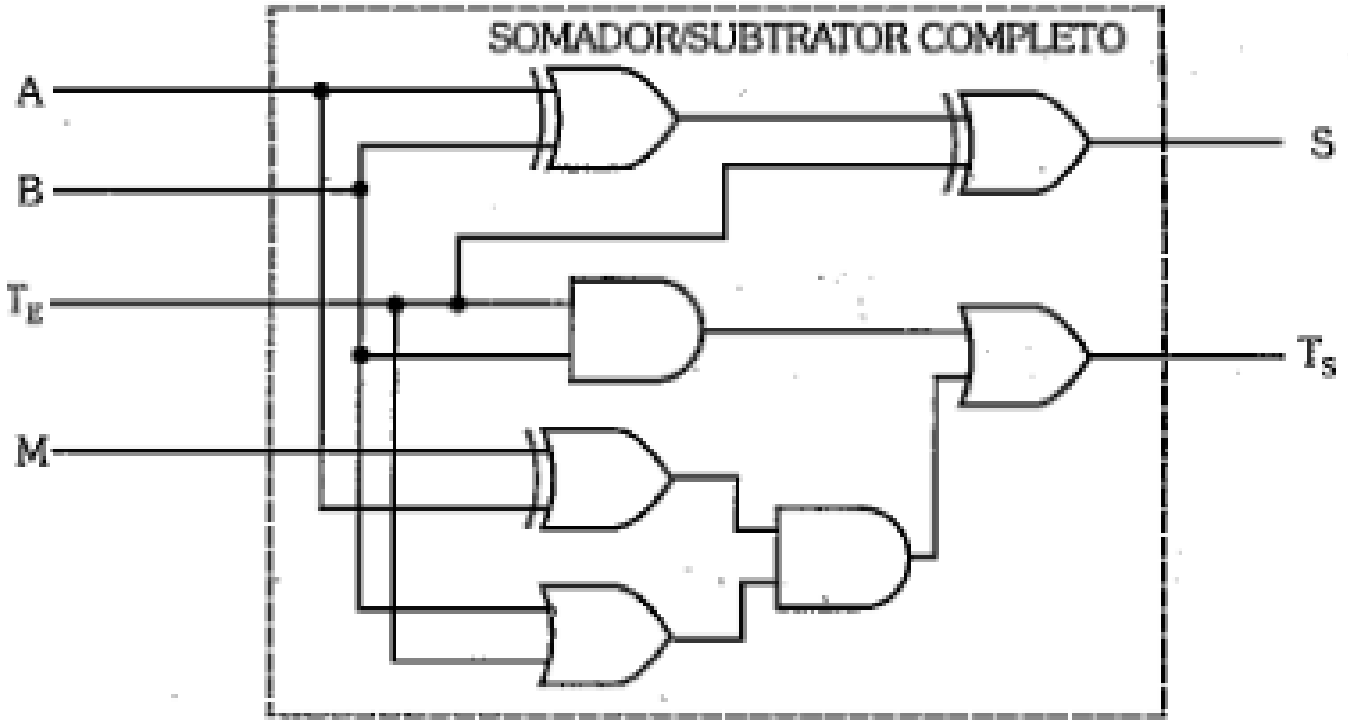
Somador/Subtrator

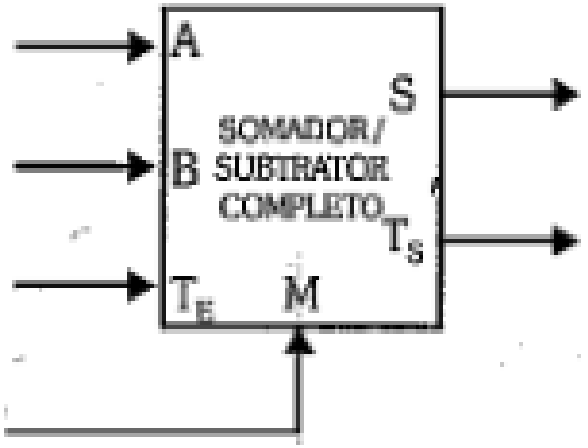
- Vamos fazer o diagrama de Karnaugh para S e T?
- $S = A \oplus B \oplus Cin$
- Cout = BCin + (~M)AB + (~M)ACin + M(~A)B + M(~A)Cin

- Fatorando Cout um pouco... como fica? (dica, coloque B e Cin em evidência)

Somador/Subtrator

- Esquema final





Somador/Subtrator

- Há outra forma de fazer um somador/subtrator, aproveitando a semelhança entre os circuitos (nota, FA = full-adder, ou somador-completo)