Universidade Federal da Fronteira Sul
Curso de Ciência da Computação
Campus Chapecó

Circuitos Aritméticos

Prof. Luciano L. Caimi
lcaimi@uffs.edu.br

Circuitos Aritméticos

Adição Binária

Adição de números sem sinal

	A		0	1	0	0			**Transporte (*carry*)**
+	B			0	1	0	0	(4)	
	S	+		0	1	1	0	(6)	
				1	0	1	0	(10)	**resultado**

O maior número binário que podemos representar com 4 bits é 15

Circuitos Aritméticos

Adição Binária

Adição de números sem sinal

		1	1	0	0			**Transporte (*carry*)**
overflow			1	1	0	0	(12)	
	+		0	1	1	0	(6)	
			0	0	1	0	(18)	**resultado**

Circuitos Aritméticos

Adição Binária

Outro exemplo (8 bits)

	0	1	1	0	1	1	0	0			
		0	1	1	0	1	1	0	1	(109) →	A
+		0	1	1	0	0	1	1	0	(102) →	+ B
		1	1	0	1	0	0	1	1	(211) →	S

Circuitos Aritméticos

Adição Binária

Adição de números sem sinal

	0	1	1	0	1	1	0	0		
		0	1	1	0	1	1	0	1	(109)
+		0	1	1	0	0	1	1	0	(102)
		1	1	0	1	0	0	1	1	(211)

		a_0	**adição de 2 bits**
+		b_0	
	c_1	s_0	**resultado em 2 bits**

Adição Binária

A partir do 2º bit

	0	1	1	0	1	1	0	0		
		0	1	1	0	1	1	0	1	(109)
+		0	1	1	0	0	1	1	0	(102)
		1	1	0	1	0	0	1	1	(211)

		c_1	
		a_1	**adição de 3 bits**
+		b_1	
	c_2	s_1	**resultado em 2 bits**

Circuitos Aritméticos

▶ Adição Binária

Generalizando, a partir do 2º bit

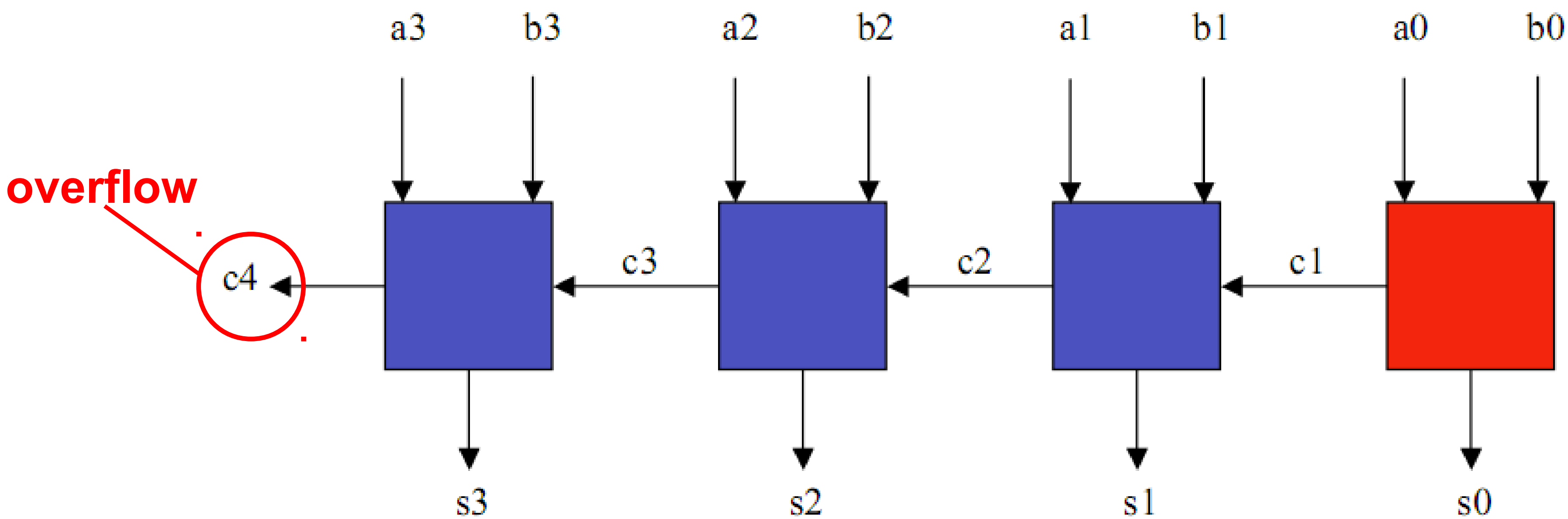
	0	1	1	0	1	1	0	0		
		0	1	1	0	1	1	0	1	(109)
+		0	1	1	0	0	1	1	0	(102)
		1	1	0	1	0	0	1	1	(211)

		c_i	
		a_i	**adição de 3 bits**
+		b_i	
	C_{i+1}	s_i	**resultado em 2 bits**

Circuitos Aritméticos

Adição Binária

Esquema para soma paralela



Observe:

- Existe um elemento para cada coluna da soma
- O sinal de *overflow* será o *carry* mais significativo

Adição Binária

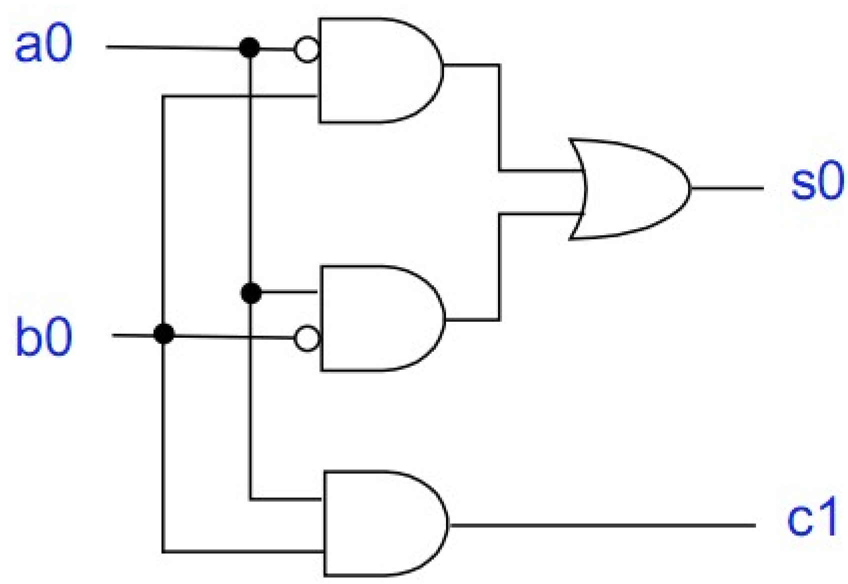
Projetando o circuito para primeira coluna:

Meio-Somador (Half-Adder)

$$s_0 = \overline{a_0} . b_0 + a_0 . \overline{b_0}$$

$$c_1 = a_0 . b_0$$

entradas		saídas	
a_0	b_0	c_1	s_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

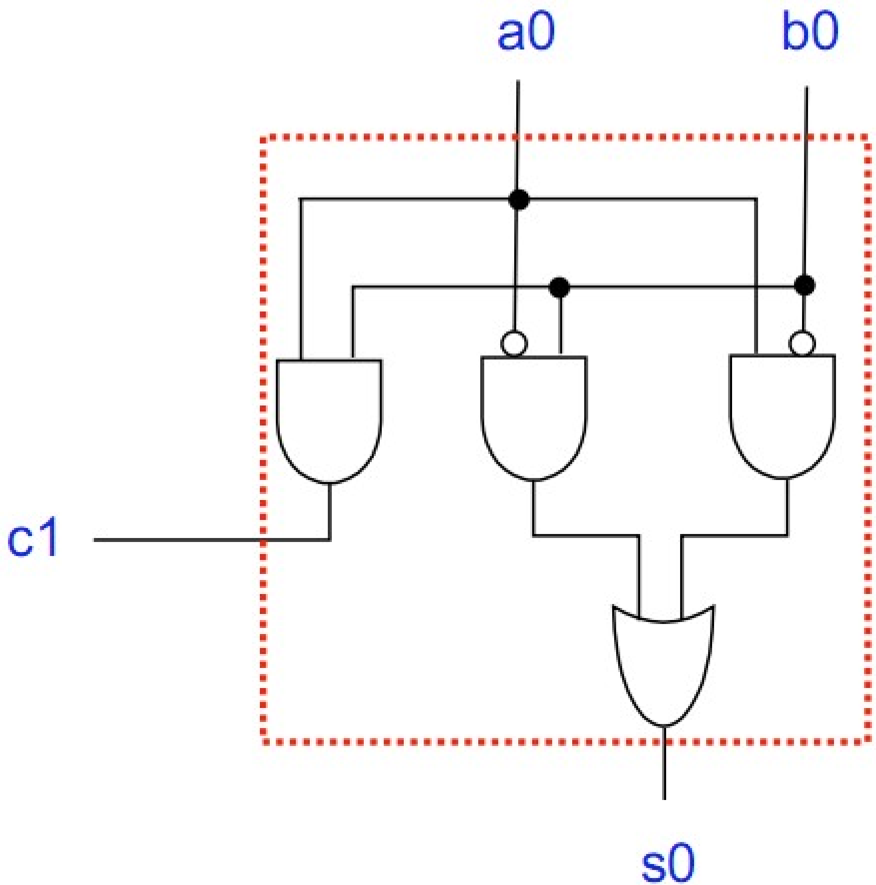


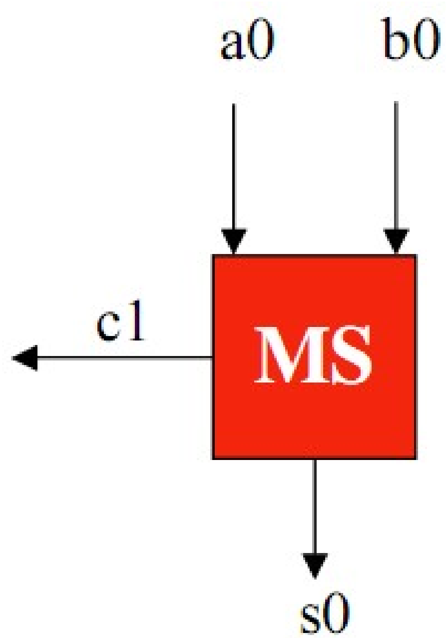
Circuitos Aritméticos

Adição Binária

Meio-Somador (Half-Adder)

...se preferir





$$s_0 = \overline{a_0} . b_0 + a_0 . \overline{b_0}$$

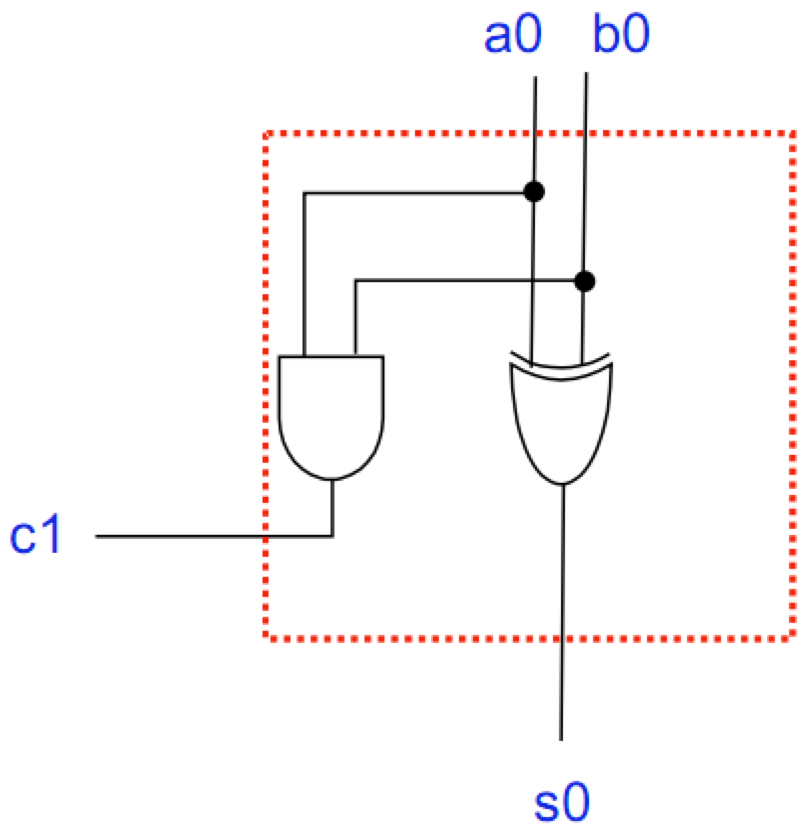
$$c_0 = a_0 . b_0$$

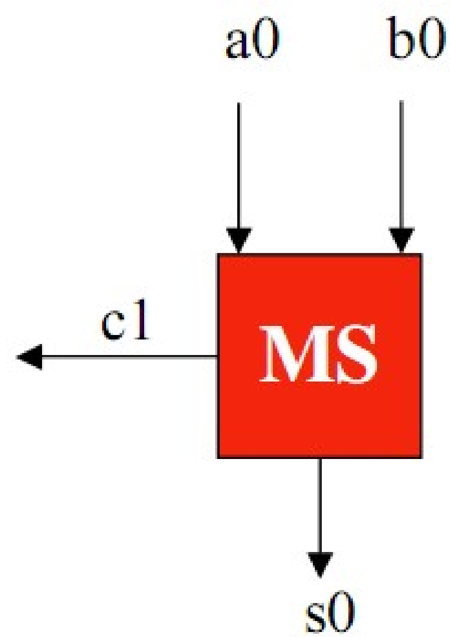
Circuitos Aritméticos

Adição Binária

Meio-Somador (Half-Adder)

...ou ainda





$$s_0 = \overline{a_0} . b_0 + a_0 . \overline{b_0}$$

$$a_0 \oplus b_0$$

$$c_0 = a_0 . b_0$$

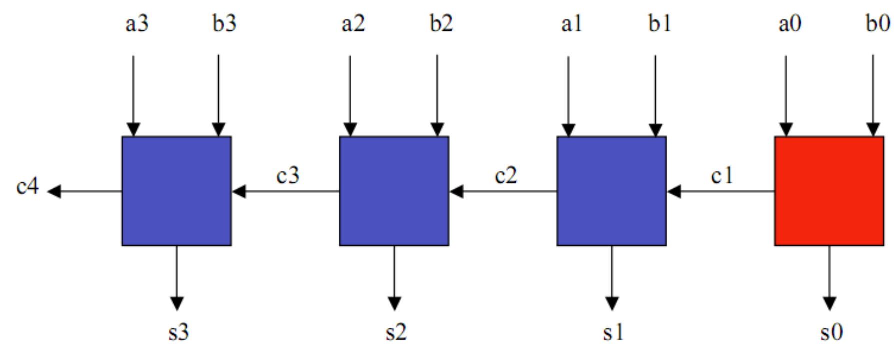
Circuitos Aritméticos

Adição Binária

Projetando as demais colunas:

Somador-Completo (Full-Adder)

	0	1	1	0	1	1	0	0
		0	1	1	0	1	1	0	1
+		0	1	1	0	0	1	1	0
		1	1	0	1	0	0	1	1



		c_i
		a_i
+		b_i
	C_{i+1}	s_i

adição de 3 bits

resultado em 2 bits

Circuitos Aritméticos

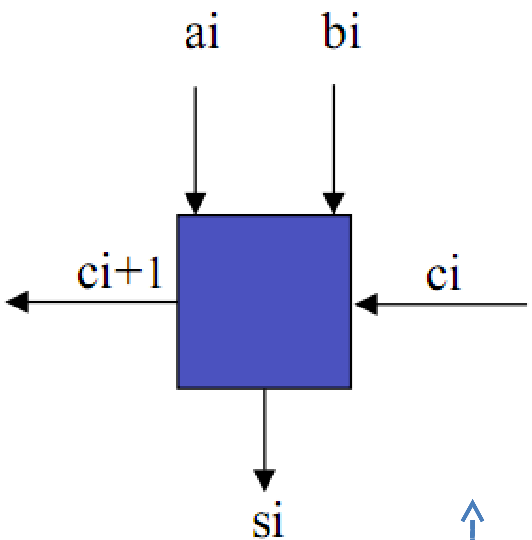
Adição Binária

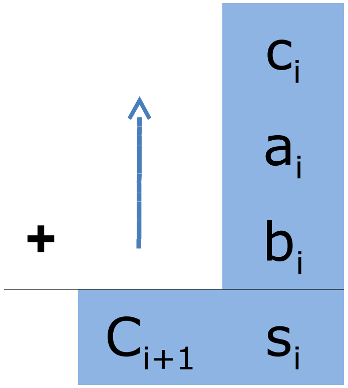
Projetando as demais colunas:

Somador-Completo (Full-Adder)

entradas saídas

a_i	b_i	c_i	C_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1





adição de 3 bits

resultado em 2 bits

Circuitos Aritméticos

Adição Binária

Projetando as demais colunas:

Somador-Completo (Full-Adder)

a_i	b_i	c_i	C_{i+1}	s_i
	entradas		saídas	
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Mapa de Karnaugh para c_{i+1}

c_{i+1}	$\overline{b_i}$	$\overline{b_i}$	b_i	b_i
$\overline{a_i}$	0	0	1	0
a_i	0	1	1	1
	$\overline{c_i}$	c_i	c_i	$\overline{c_i}$

$$c_{i+1} = a_i . b_i + a_i . c_i + b_i . c_i$$

Circuitos Aritméticos

Adição Binária

Projetando as demais colunas:

Somador-Completo (Full-Adder)

entradas saídas

a_i	b_i	c_i	C_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Mapa de Karnaugh para s_i

s_i	$\overline{b_i}\,\overline{c_i}$	$\overline{b_i}\,c_i$	$b_i\,c_i$	$b_i\,\overline{c_i}$
$\overline{a_i}$	0	1	0	1
a_i	1	0	1	0

Não é possível simplificar, logo:

$$s_i = \overline{a_i}.\overline{b_i}.c_i + \overline{a_i}.b_i.\overline{c_i} + a_i.\overline{b_i}.\overline{c_i} + a_i.b_i.c_i$$

Adição Binária

manipulando através da álgebra de boole:

$$s_i = \overline{a}_i . \overline{b}_i . c_i + \overline{a}_i . b_i . \overline{c}_i + a_i . \overline{b}_i . \overline{c}_i + a_i . b_i . c_i$$

$$s_i = \overline{a}_i . \left(\overline{b}_i . c_i + b_i . \overline{c}_i \right) + a_i . \left(\overline{b}_i . \overline{c}_i + b_i . c_i \right)$$

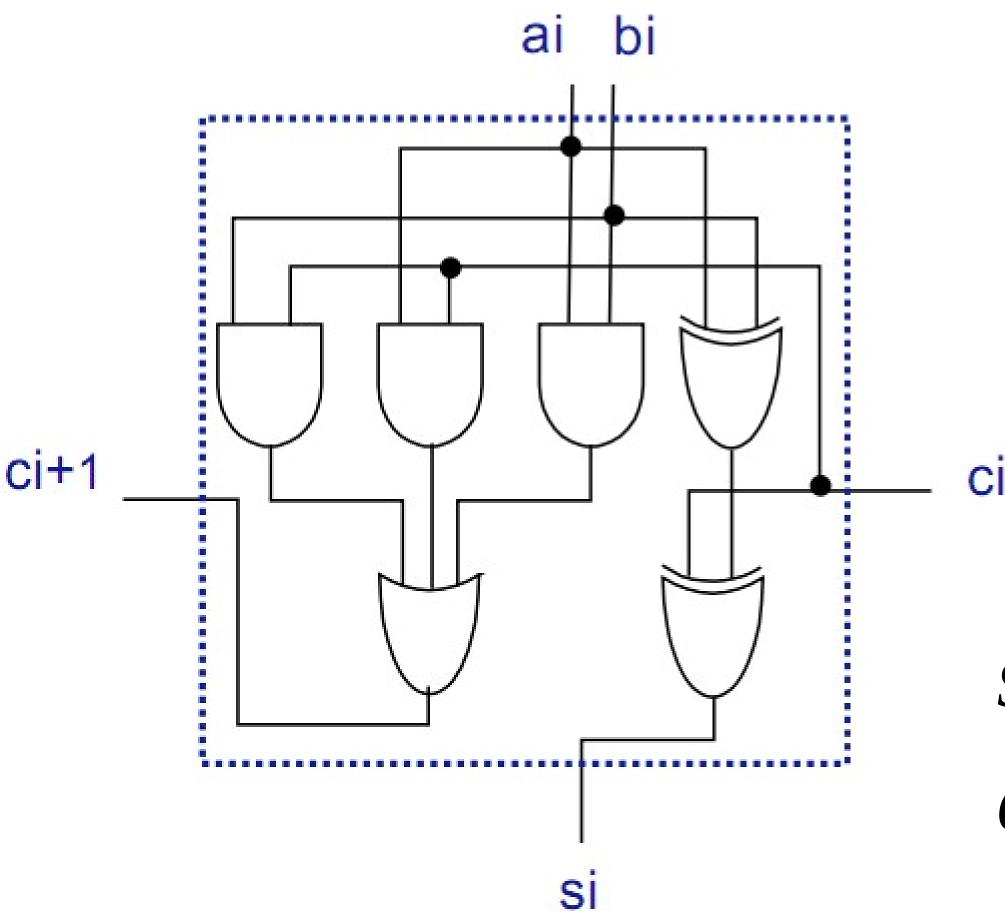
$$s_i = \overline{a}_i . \left(b_i \oplus c_i \right) + a_i . \left(\overline{b_i \oplus c_i} \right)$$

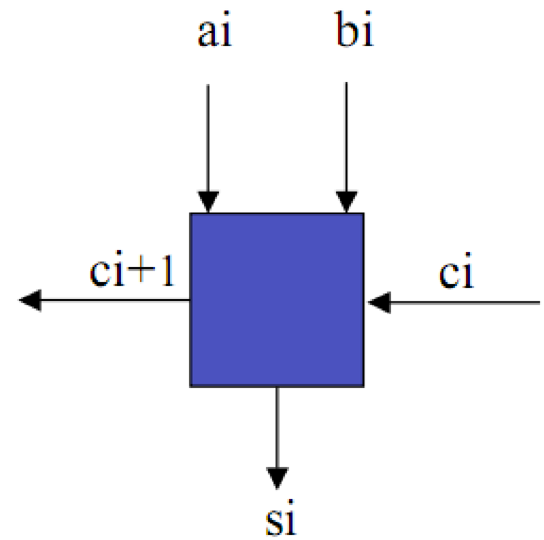
$$s_i = a_i \oplus \left(b_i \oplus c_i \right)$$

a	b	xor	$\overline{\text{xor}}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Adição Binária

Somador-Completo (Full-Adder)





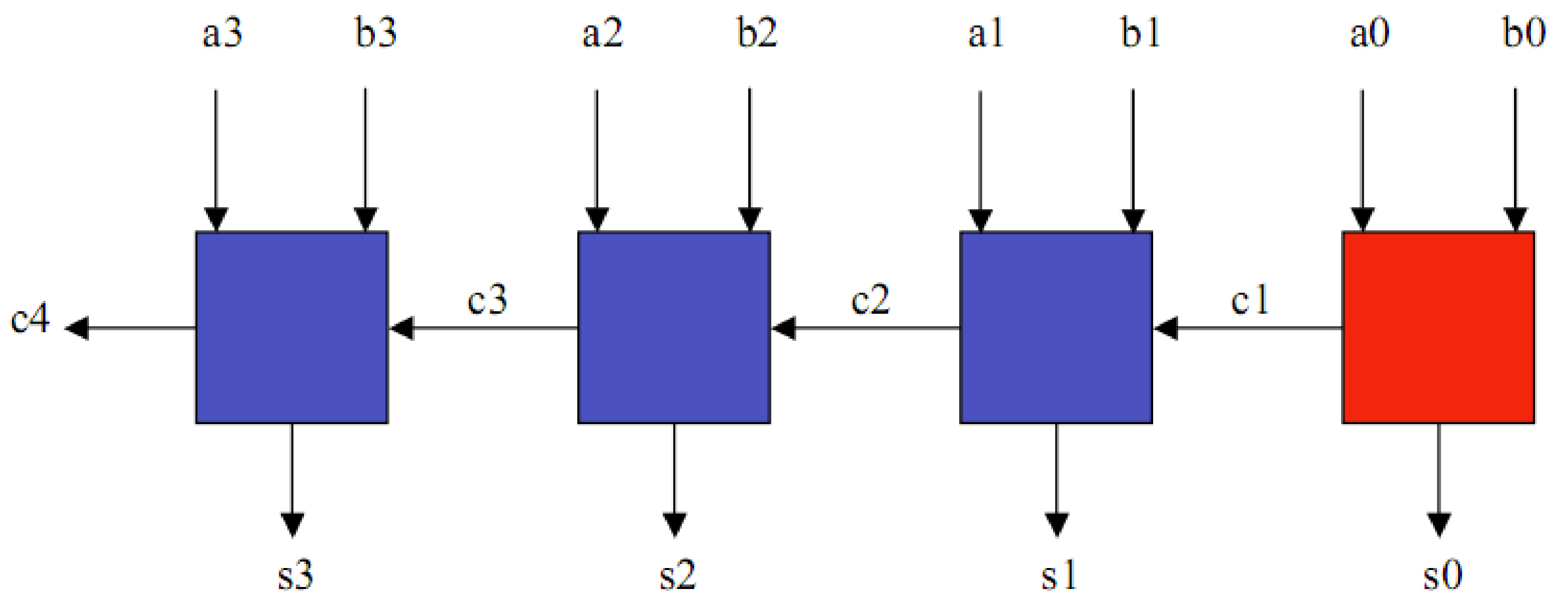
$$s_i = a_i \oplus \left(b_i \oplus c_i \right)$$

$$c_{i+1} = a_i . b_i + a_i . c_i + b_i . c_i$$

Circuitos Aritméticos

Adição Binária

Considerando dois números (A e B) de 4 bits cada

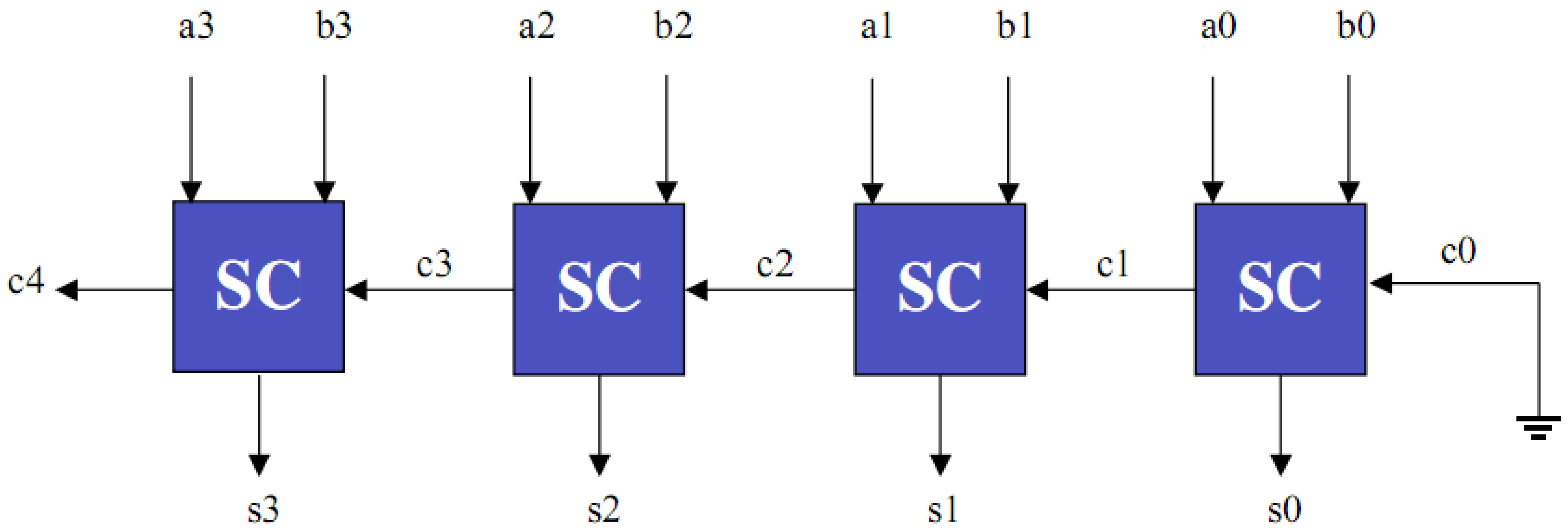


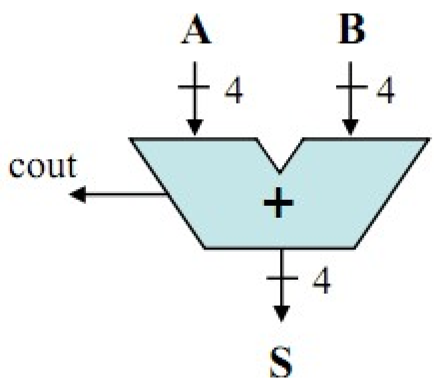
Chamado somador *ripple-carry*

Circuitos Aritméticos

Adição Binária

Versão utilizando somente somador-completo





usto é ligeiramente maior.

Circuitos Aritméticos

Subtração Binária

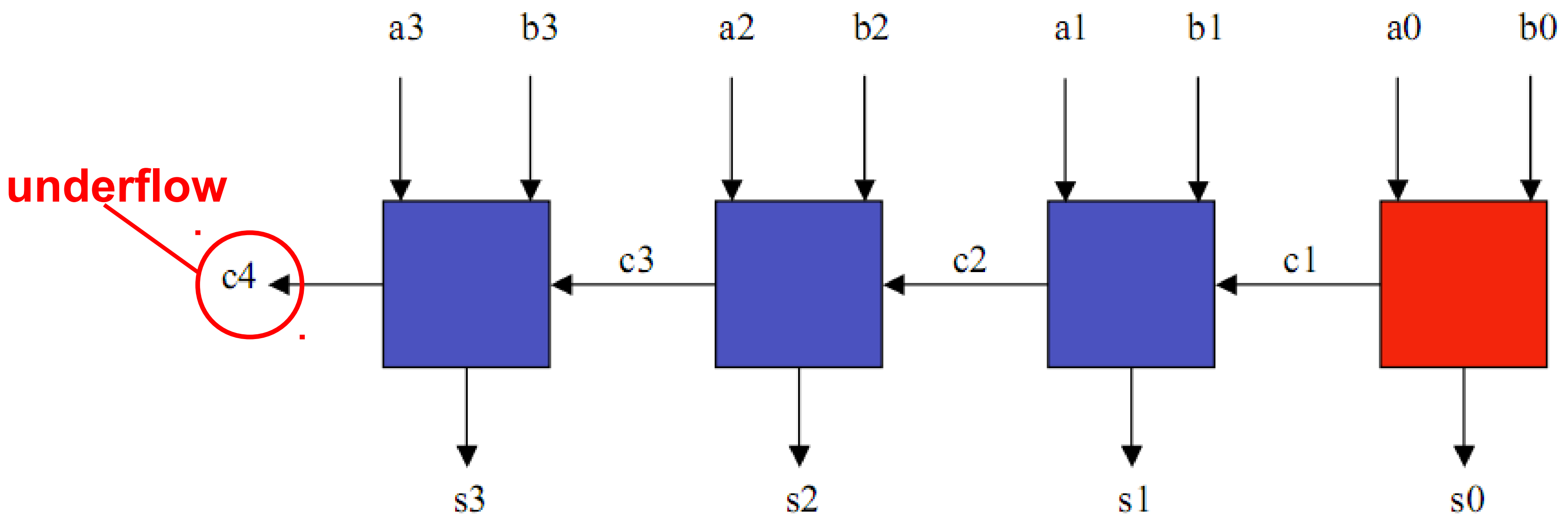
Subtração de números sem sinal

	A		0	0	2	~~2~~ 1	2		***(borrow)***
-	**B**			1	~~1~~	0	0	(12)	
		-		0	1	0	1	(5)	
	S			0	1	1	1	(7)	**resultado**

Circuitos Aritméticos

Subtração Binária

Esquema para subtração paralela



Observe:

- Existe um elemento para cada coluna da soma
- O sinal de *underflow* será o *borrow* mais significativo

Subtração Binária

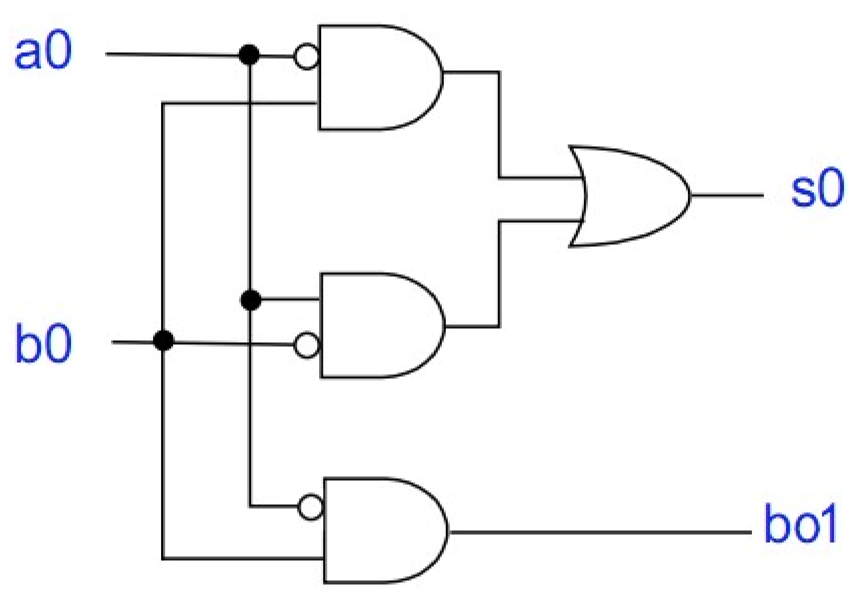
Projetando o circuito para primeira coluna:

Meio-Substrator (half-sub)

$$s_0 = \overline{a_0}.b_0 + a_0.\overline{b_0}$$
$$c_1 = \overline{a_0}.b_0$$

a_0	b_0	bo_1	s_0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

(entradas: a_0, b_0; saídas: bo_1, s_0)



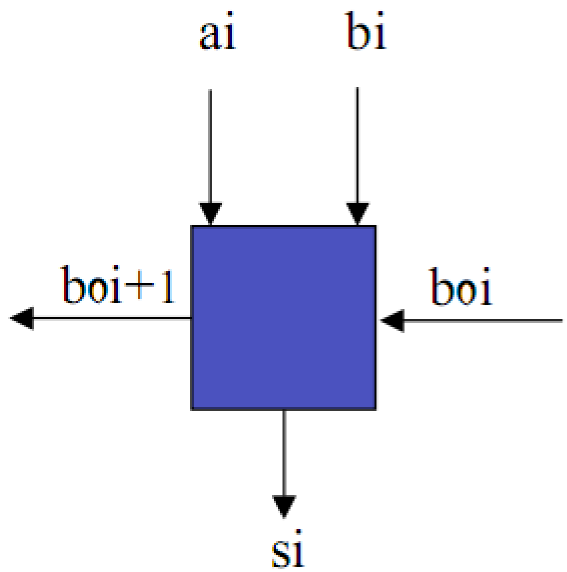
Circuitos Aritméticos

Subtração Binária

Projetando as demais colunas:

Subtrator-Completo (full-sub)

entradas			saídas	
a_i	b_i	c_i	bo_{i+1}	s_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



$$
\begin{array}{rr}
 & bo_i \\
 & a_i \\
- & b_i \\
\hline
bo_{i+1} & s_i
\end{array}
$$

subtração de 3 bits

resultado em 2 bits

Circuitos Aritméticos

Subtração Binária

Subtrator-Completo (full-sub)

entradas			saídas	
a_i	b_i	c_i	bo_{i+1}	s_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Mapa de Karnaugh para bo_{i+1}

bo_{i+1}	$\overline{b_i}$		b_i	
$\overline{a_i}$	0	1	1	1
a_i	0	0	1	0
	$\overline{c_i}$	c_i		$\overline{c_i}$

$$bo_{i+1} = \overline{a}_i . b_i + \overline{a}_i . c_i + b_i . c_i$$

Circuitos Aritméticos

Subtração Binária

Projetando as demais colunas:

Subtrator-Completo (full-sub)

entradas			saídas	
a_i	b_i	c_i	C_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Mapa de Karnaugh para s_i

s_i	$\overline{b_i}$	$\overline{b_i}$	b_i	b_i
$\overline{a_i}$	0	1	0	1
a_i	1	0	1	0
	$\overline{c_i}$	c_i	c_i	$\overline{c_i}$

Não é possível simplificar, logo:

$$s_i = \overline{a_i}.\overline{b_i}.c_i + \overline{a_i}.b_i.\overline{c_i} + a_i.\overline{b_i}.\overline{c_i} + a_i.b_i.c_i$$

Circuitos Aritméticos

Subtração Binária

manipulando através da álgebra de boole:

$$s_i = \overline{a}_i.\overline{b}_i.c_i + \overline{a}_i.b_i.\overline{c}_i + a_i.\overline{b}_i.\overline{c}_i + a_i.b_i.c_i$$

$$s_i = \overline{a}_i.\left(\overline{b}_i.c_i + b_i.\overline{c}_i\right) + a_i.\left(\overline{b}_i.\overline{c}_i + b_i.c_i\right)$$

$$s_i = \overline{a}_i.\left(b_i \oplus c_i\right) + a_i.\left(\overline{b_i \oplus c_i}\right)$$

$$s_i = a_i \oplus \left(b_i \oplus c_i\right)$$

a	b	xor	$\overline{\text{xor}}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Circuitos Aritméticos

Adição de números inteiros

E se quisermos realizar operações sobre números com sinal???

- Precisaremos considerar uma representação que sirva tanto para binários positivos quanto binários negativos
- A representação mais usada, neste caso, é complemento de 2
- Porque mesmo?!?!?!?

Adição de números inteiros

Representação em complemento de 2

Sinal	Regra de formação	exemplo
positivo	igual a sinal-maginitude	+13 = 00001101
negativo	1. toma-se a representação em sinal e magnitude 2. faz-se o complemento de 1 (inverte o número bit a bit) 3. soma-se 1	+13 = 00001101 11110010 + 1 -13 = 11110011

-128	64	32	16	8	4	2	1
1	1	1	1	0	0	1	1

▶ Adição de números inteiros

Adição de números em complemento de 2

Considere valor binários de 4 bits, portanto, com faixa de representação de [-8:+7]

Exemplo 1: números positivos com soma ≤ 7

	0	0	0	0			**Transporte (*carry*)**
		0	1	0	0	(+4)	
+		0	0	1	0	(+2)	
		0	1	1	0	(+6)	**Resultado correto**

Circuitos Aritméticos

Adição de números inteiros

Adição de números em complemento de 2

Exemplo 2: nros negativos com soma ≥ -8

	1	1	0	0		Transporte (*carry*)
		1	1	0	0 (-4)	
+		1	1	1	0 (-2)	
		1	0	1	0 (-6)	Resultado correto

Apesar deste último carry valer 1, não houve "overflow"

Circuitos Aritméticos

▶ Adição de números inteiros

Adição de números em complemento de 2

Exemplo 3: um número positivo e outro negativo com resultado positivo

	1	1	0	0			Transporte (*carry*)
		0	1	0	0	(+4)	
+		1	1	1	0	(-2)	
		0	0	1	0	(+2)	**Resultado correto**

Novamente...

Circuitos Aritméticos

Adição de números inteiros

Adição de números em complemento de 2

Exemplo 4: um número positivo e outro negativo com resultado negativo

	0	0	1	0			**Transporte (*carry*)**
		0	0	1	1	(+3)	
+		1	0	1	0	(-6)	
		1	1	0	1	(-3)	**Resultado correto**

Circuitos Aritméticos

Adição de números inteiros

Adição de números em complemento de 2

Exemplo 5: um número positivo e outro negativo iguais em módulo

	1	1	1	1			Transporte (*carry*)
		0	1	0	1	(+5)	
+		1	0	1	1	(-5)	
		0	0	0	0	(-0)	Resultado correto

Circuitos Aritméticos

Adição de números inteiros

Adição de números em complemento de 2

Exemplo 6: dois números positivos

	0	1	0	0		**Transporte (*carry*)**
		0	1	0	1	(+5)
+		0	1	0	0	(+4)
		1	0	0	1	(+9) **Resultado Errado**

Em complemento de 2 o resultado foi -7

Adição de números inteiros

Adição de números em complemento de 2

Exemplo 7: dois números negativos

	1	0	0	0		**Transporte (*carry*)**
		1	1	0	0 (-4)	
+		1	0	1	1 (-5)	
		0	1	1	1 (-9)	**Resultado Errado**

Em complemento de 2 o resultado foi 7

Circuitos Aritméticos

Adição de números inteiros

Resultados Corretos

	0	0	0	0	
		0	1	0	0 (+4)
+		0	0	1	0 (+2)
		0	1	1	0 (+6)

	1	1	0	0	
		1	1	0	0 (-4)
+		1	1	1	0 (-2)
		1	0	1	0 (-6)

	0	0	1	0	
		0	0	1	1 (+3)
+		1	0	1	0 (-6)
		1	1	0	1 (-3)

	1	1	1	1	
		0	1	0	1 (+5)
+		1	0	1	1 (-5)
		0	0	0	0 (-0)

Resultados Errados

	0	1	0	0	
		0	1	0	1 (+5)
+		0	1	0	0 (+4)
		1	0	0	1 (+9)

	1	0	0	0	
		1	1	0	0 (-4)
+		1	0	1	1 (-5)
		0	1	1	1 (-9)

Circuitos Aritméticos

Adição de números inteiros

Resultados Corretos

0	0	0	0		
	0	1	0	0	(+4)
+	0	0	1	0	(+2)
	0	1	1	0	(+6)

1	1	0	0		
	1	1	0	0	(-4)
+	1	1	1	0	(-2)
	1	0	1	0	(-6)

0	0	1	0		
	0	0	1	1	(+3)
+	1	0	1	0	(-6)
	1	1	0	1	(-3)

1	1	1	1		
	0	1	0	1	(+5)
+	1	0	1	1	(-5)
	0	0	0	0	(-0)

Resultados Errados

0	1	0	0		
	0	1	0	1	(+5)
+	0	1	0	0	(+4)
	1	0	0	1	(+9)

1	0	0	0		
	1	1	0	0	(-4)
+	1	0	1	1	(-5)
	0	1	1	1	(-9)

Circuitos Aritméticos

Adição de números inteiros

Adição de números em complemento de 2

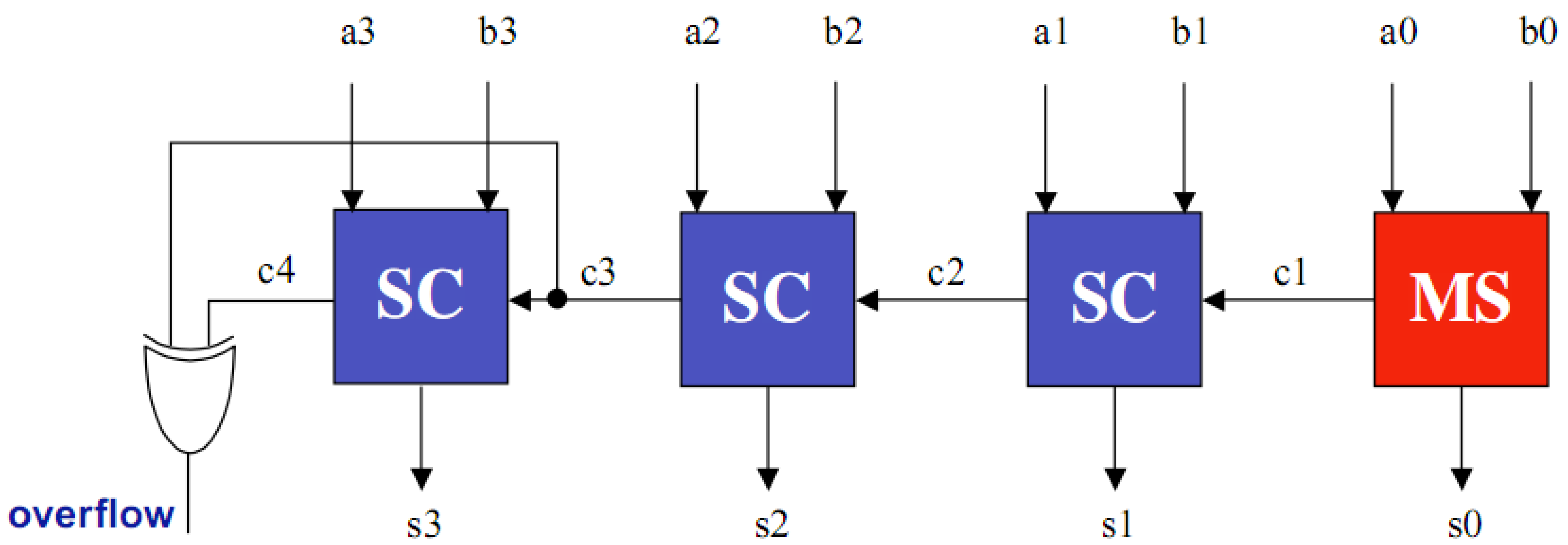
Conclusões:

- Números binários em complemento de 2 podem ser adicionados como se fossem número binários sem sinal
- Neste caso, a detecção de overflow se dá comparando-se os dois últimos sinais de carry

Circuitos Aritméticos

Adição de números inteiros

Adição de números em complemento de 2



esquemático de blocos

Subtração Binária

Princípio Básico

$$A - B = A + (-B)$$

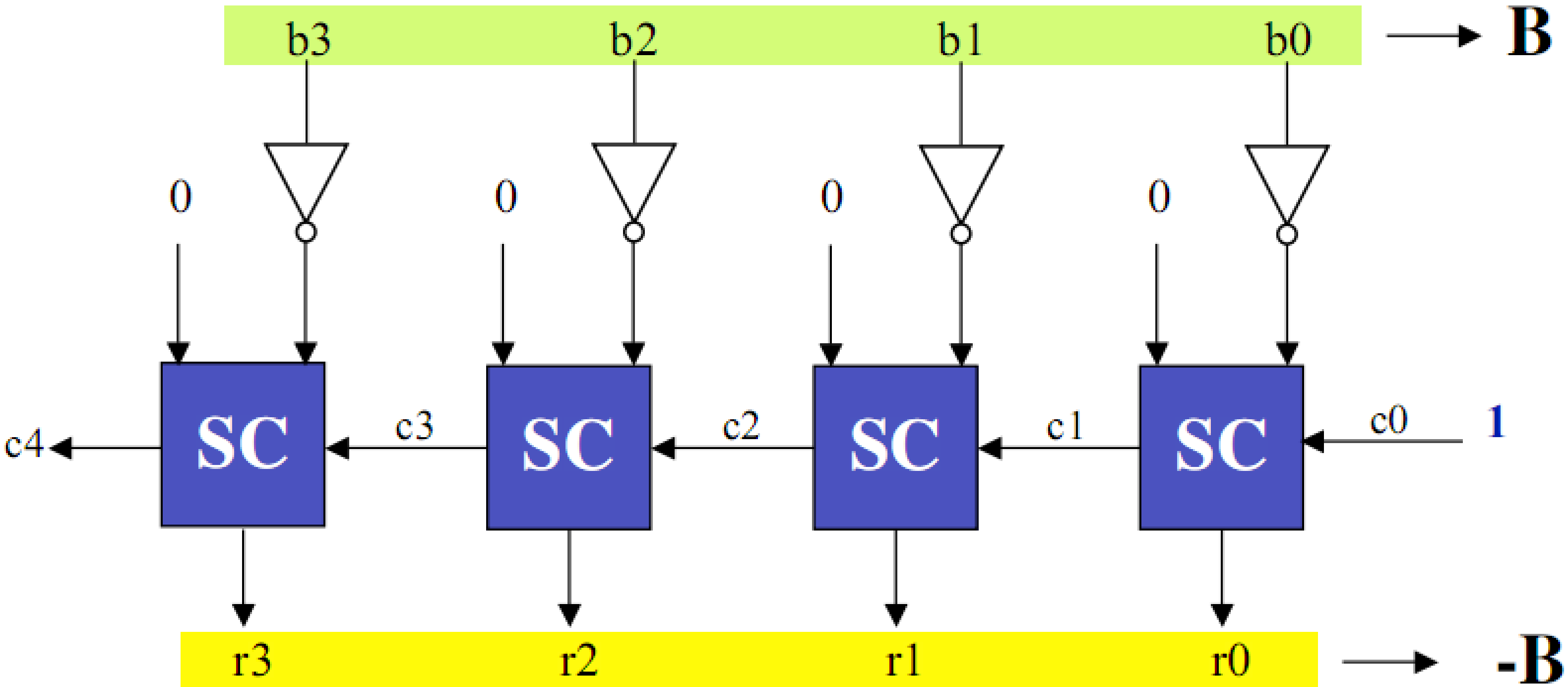
Onde –B é o número B com o sinal trocado

Em complemento de 2:

a) faz o complemento do número

b) soma 1 unidade

Subtração Binária

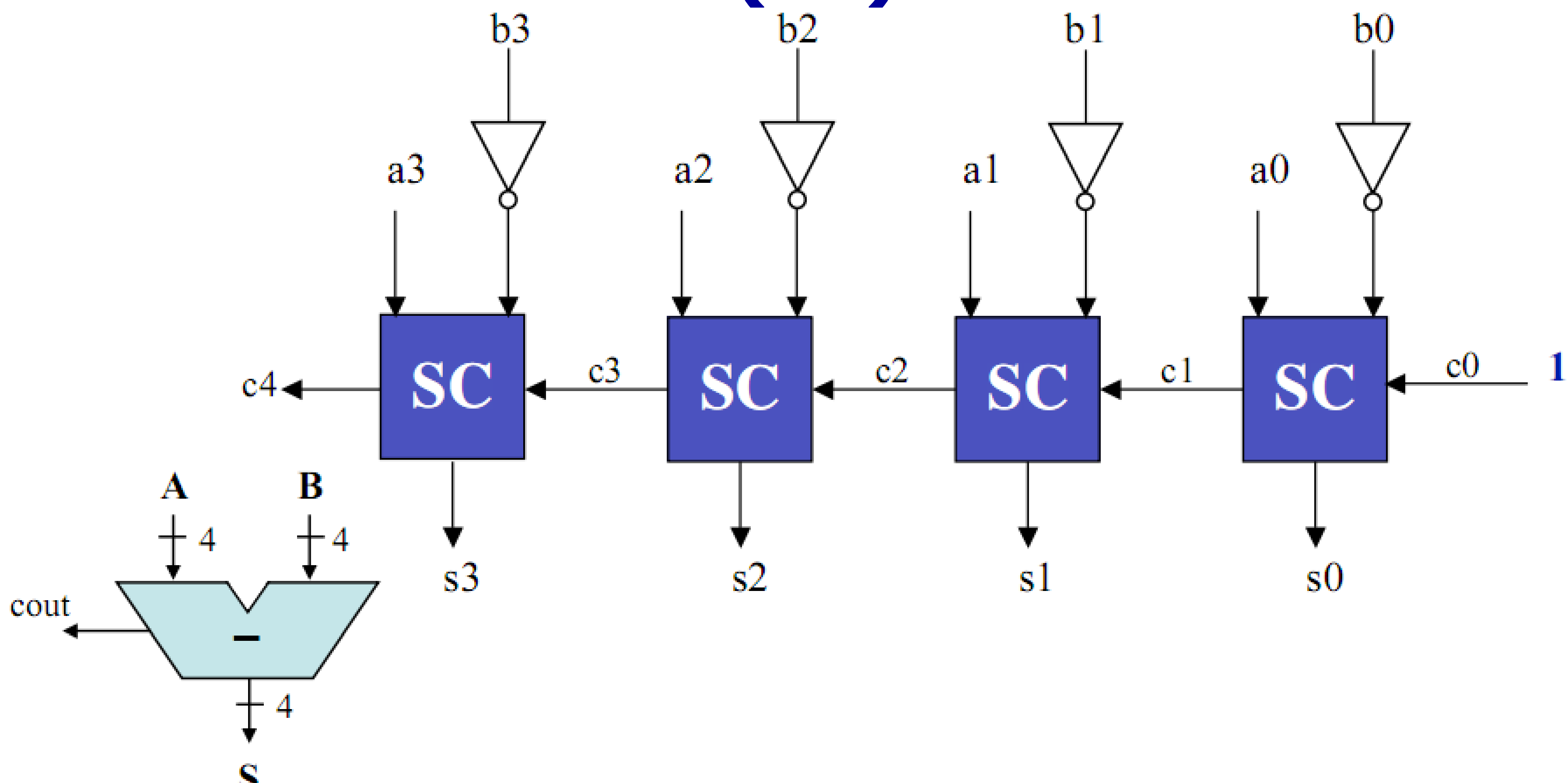


Circuitos Aritméticos

Subtração Binária

Considerando a operação completa

A – B = A + (-B)



Circuitos Aritméticos

Subtração Binária

Mas com tanta semelhança entre os circuitos de adição e subtração não será possível "programar" as 2 operações com o "mesmo" hardware???

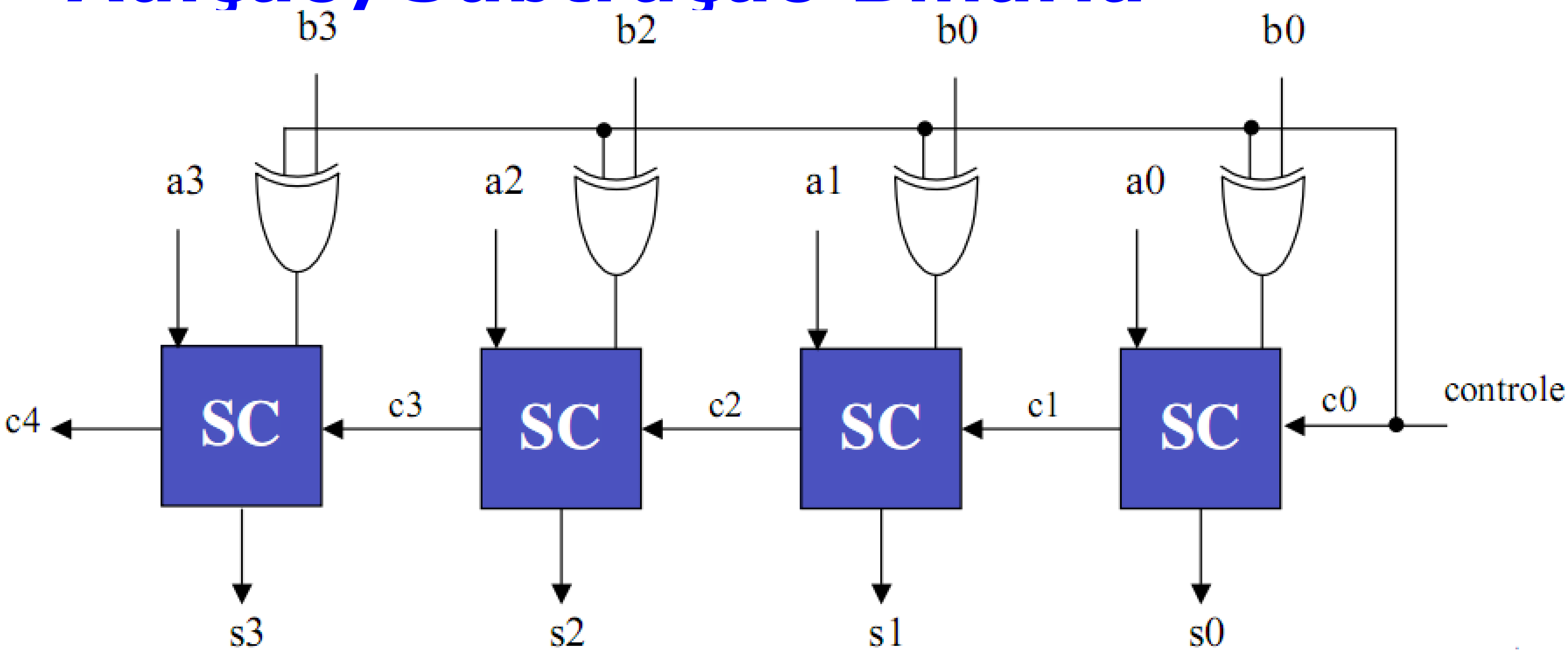
Sim, modificações necessárias:

a) Substituir os inversores por "negadores controlados" (xors);

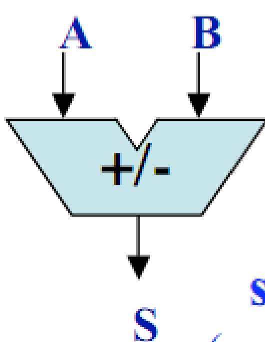
b) Controlar o valor de c0 (0 para adição/1 para subtração)

Circuitos Aritméticos

Adição/Subtração Binária



controle	operação
0	**S= A + B**
1	**S= A - B**

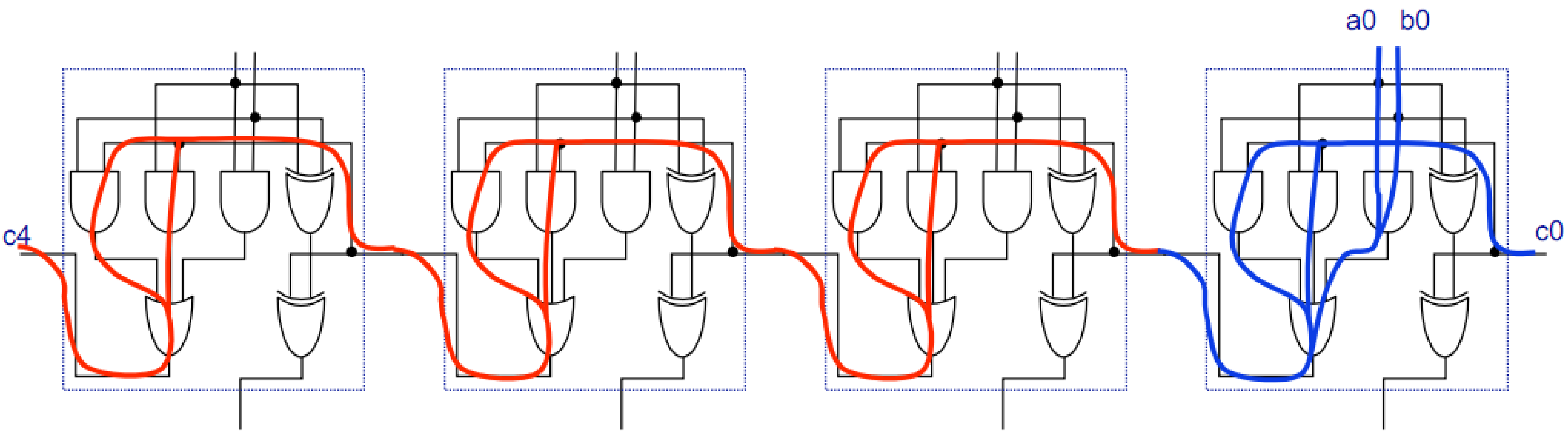


símbolo
(no nível RT)

Circuitos Aritméticos

▶ Analisando o *carry*

Problema da propagação do *carry*



Sinal c_4 estabiliza somente depois de c_1, c_2 e c_3 estabilizarem:

Caminho de maior atraso de a_0 e b_0 ou c_i até c_{i+1}

Caminho de maior atraso de c_i até c_{i+1}