Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Ciências da Computação & Engenharia Eletrônica

Sistemas Digitais

INE 5406

Aula 1-T

1. Projeto de unidade lógico-aritmética (ULA). Representação de números inteiros em binário. Convenções do nível RT. Adição de números sem e com sinal, o somador paralelo *carry-ripple* e overflow. O subtrator e o somador-subtrator, overflow. Estrutura de uma ULA simples. Outras operações lógicas e aritméticas. Unidades funcionais para mais de dois. Circuitos Digitais e Níveis de Abstração.

Prof. José Luís Güntzel
guntzel@inf.ufsc.br

www.inf.ufsc.br/~guntzel/ine5406/ine5406.html

1. Projeto de Unidade Lógico-Aritmética

Representação de Inteiros em Binário

Números Inteiros sem Sinal (Naturais)

• assumindo-se números com **4 bits**

	binário	decimal
Menor número	0000	0
Maior número	1111	15

Intervalo de representação: [0 , 15]

1. Projeto de Unidade Lógico-Aritmética

Representação de Inteiros em Binário

Números Inteiros sem Sinal (naturais)

• assumindo-se números com **8 bits**

	binário	decimal
Menor número	**00000000**	**0**
Maior número	**11111111**	**255**

Intervalo de representação: [0 , 255]

Representação de Inteiros em Binário

Números Inteiros sem Sinal (Naturais)

Observe que:

$11111111_2 =$

$= 1x2^0 + 1x2^1 + 1x2^2 + 1x2^3 + 1x2^4 + 1x2^5 + 1x2^6 + 1x2^7 =$

$= 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 =$ **255**

Observe também que:

$100000000_2 =$

$= 0x2^0 + 0x2^1 + 0x2^2 + 0x2^3 + 0x2^4 + 0x2^5 + 0x2^6 + 0x2^7 + 1x2^8 =$

$= 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 256 =$ **256 (= 255 + 1)**

Então, podemos nos referir ao 255 como "2^8 - 1", para efeitos de generalização.

Representação de Inteiros em Binário

Números Inteiros sem Sinal (naturais)

- Generalizando-se para **n bits**

	binário	decimal
Menor número	**0000...0**	**0**
Maior número	**1111...1**	$\mathbf{2^n-1}$

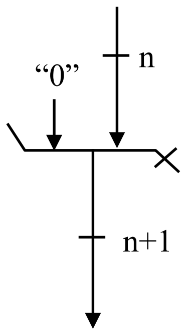
Intervalo de representação: [0 , 2^n-1]

1. Projeto de Unidade Lógico-Aritmética

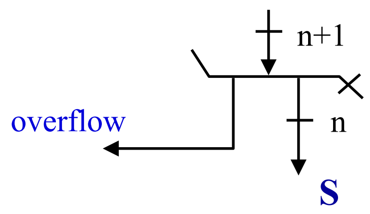
Representando Dados em Circuitos Digitais

Algumas Convenções

Indicando como um
número de *n+1* bits é
composto (outro exemplo)



"Decompondo" um número
de *n+1* bits em um número
de *n* bits e mais um sinal



1. Projeto de Unidade Lógico-Aritmética

Circuitos Aritméticos

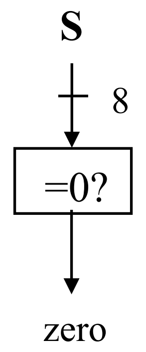
São Circuitos Combinacionais que operam sobre dados numéricos.

Exercício 1: Projetar um circuito combinacional capaz de testar se um número de 8 bits vale zero ou não. Este circuito deve operar conforme descrito na tabela abaixo.

Tabela de funcionamento:

zero	significado
0	$S \neq 0$
1	$S = 0$

Interfaces:



1. Projeto de Unidade Lógico-Aritmética

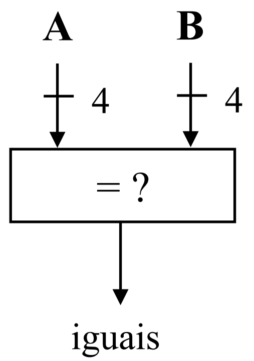
Circuitos Aritméticos

Exercício 2: Projetar um circuito combinacional que compara dois números de 4 bits cada. Este circuito deve operar conforme descrito na tabela abaixo.

Tabela de funcionamento:

iguais	significado
0	$A \neq B$
1	$A = B$

Interfaces:



Circuitos Aritméticos

Exercício 2: solução...

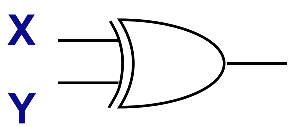
Função XOR (Exclusive OR, “OU” Exclusivo)

- Resulta “1” se as duas entradas forem iguais...

X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0

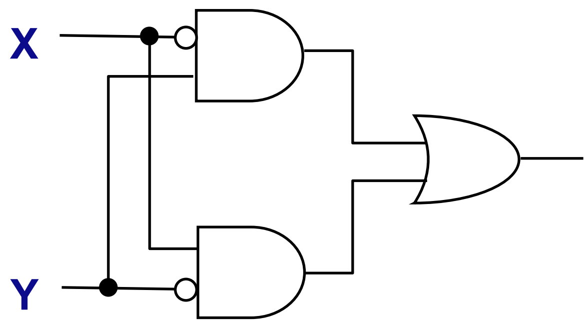
$$X \oplus Y = \overline{X} \cdot Y + X \cdot \overline{Y}$$

Porta lógica



$\equiv$

Circuito em soma de produtos



1. Projeto de Unidade Lógico-Aritmética

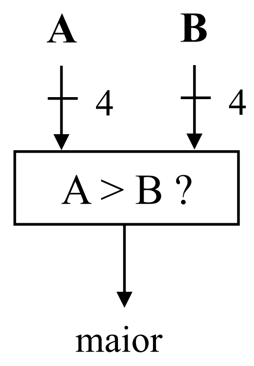
Circuitos Aritméticos

Exercício 3: **Projetar um circuito combinacional que compara dois números de 4 bits cada. Este circuito deve operar conforme descrito na tabela abaixo.**

Tabela de funcionamento:

maior	significado
0	A <= B
1	A > B

Interfaces:



1. Projeto de Unidade Lógico-Aritmética

Circuitos Aritméticos

Qual é a operação aritmética mais importante?

Resposta: a adição!

Por quê?

Porque ela serve de base para outras operações aritméticas mais frequentes nos sistemas digitais.

Revisão de Adição Binária

Adição de Números Sem Sinal

Exemplo 1:

Notação genérica

	A
+	B
	S

Análise supondo A e B com 4 bits

	0 1 0 0		**transportes (*carries*)**
	0 1 1 0	(6)	
+	0 1 0 0	(4)	
	1 0 1 0	(10)	**resultado**

Supondo um somador que trabalhe com operandos de 4 bits, então o resultado S deverá pertencer ao intervalo [0, 15]

Revisão de Adição Binária

Adição de Números Sem Sinal

Exemplo 2:

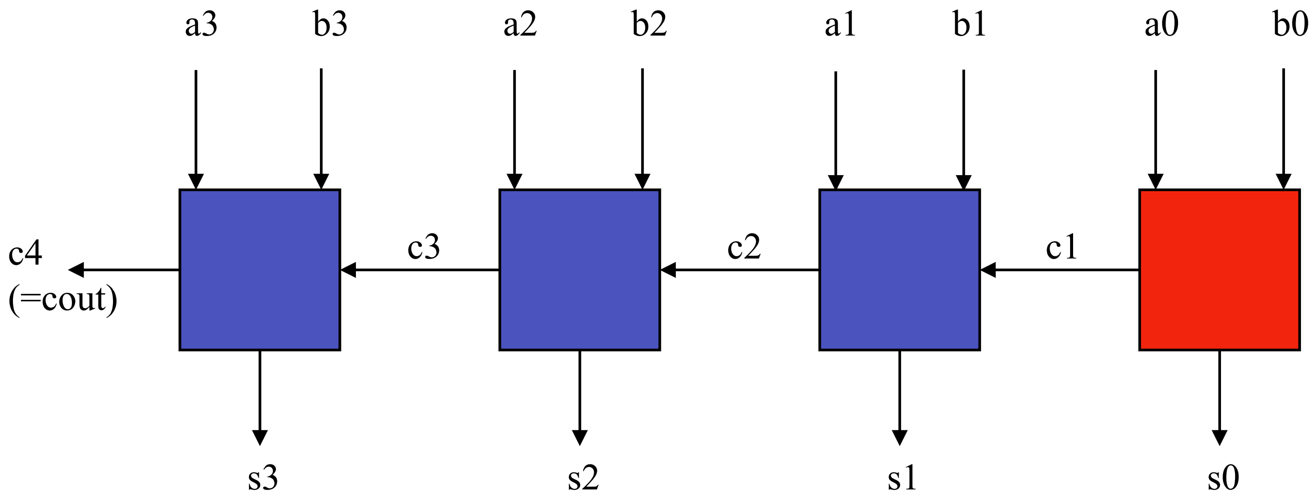
	Cout = *overflow*	1 1 1 0 0	**transportes (*carries*)**
A		1 1 0 0 (12)	
+ B	+	0 1 1 0 (6)	
S		0 0 1 0 (18)	**resultado**

Caso o resultado não puder ser representado com 4 bits, o sinal "Cout" indicará que houve um "estouro de representação" (em inglês, *overflow*).

1. Projeto de Unidade Lógico-Aritmética

Esquema da Adição Paralela

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

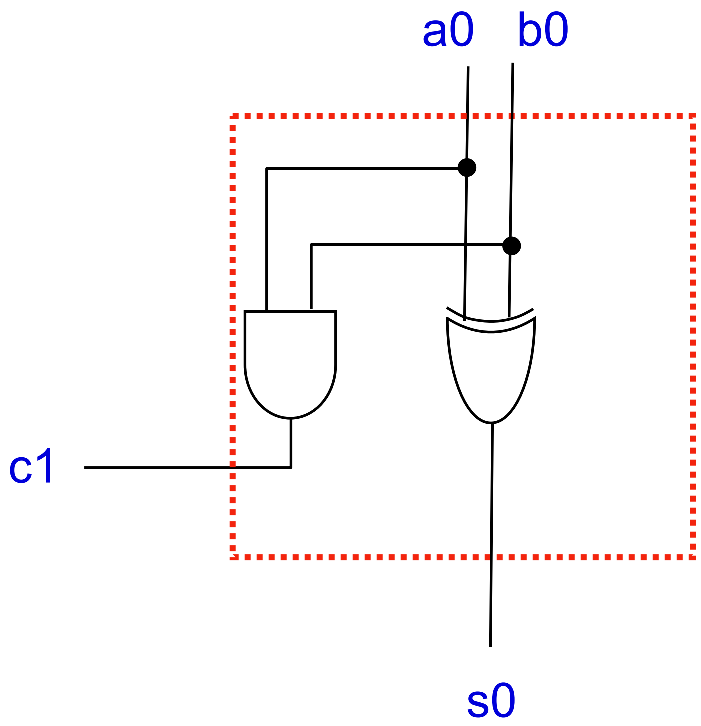


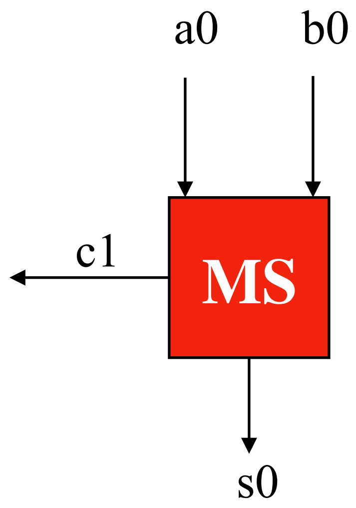
- **Há um elemento para cada coluna da soma.**
- **O sinal de transporte mais à esquerda (c4, neste caso), também recebe o nome de cout.**
- **Se este somador operar sobre inteiros sem sinal, então cout também servirá para indicar a ocorrência de *overflow***

1. Projeto de Unidade Lógico-Aritmética

O Meio Somador

Redesenhando, usando porta XOR





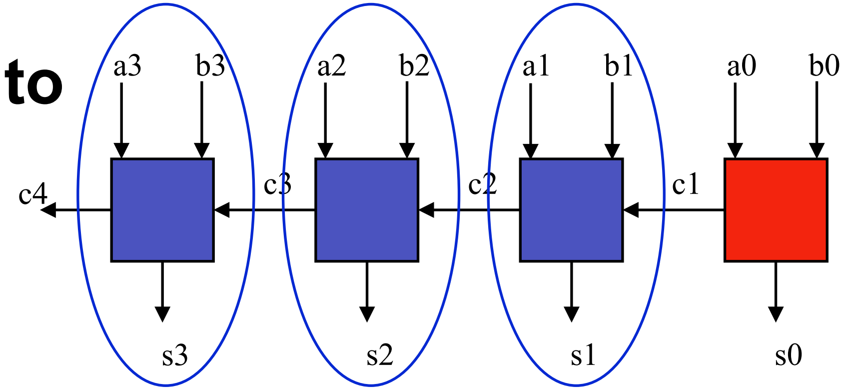
$$s0 = \overline{a0}\cdot b0 + a0\cdot\overline{b0} = a0 \oplus b0$$

$$c1 = a0\cdot b0$$

1. Projeto de Unidade Lógico-Aritmética

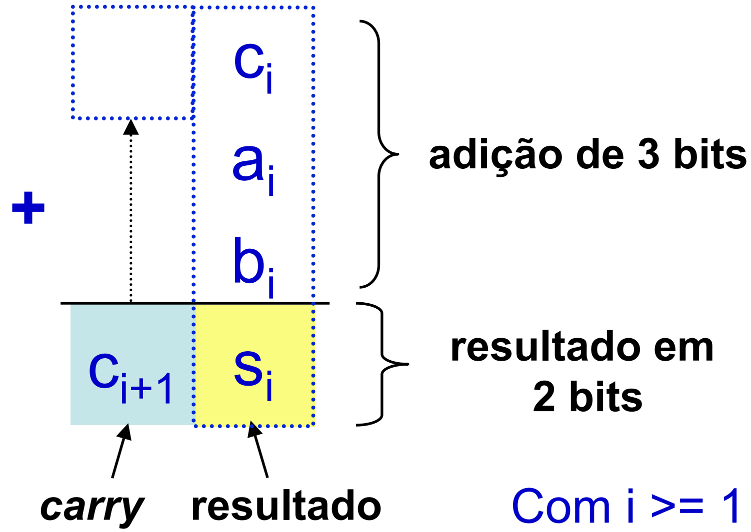
Somador Para as Demais Colunas

Projetando um tipo de circuito para as demais colunas:



$$
\begin{array}{rcccc}
 & \ldots\ c_4\ c_3 & c_2 & c_1 & \\
 & \ldots\ a_3 & a_2 & a_1 & a_0 \\
+ & \ldots\ b_3 & b_2 & b_1 & b_0 \\
\hline
 & \ldots\ s_3 & s_2 & s_1 & s_0
\end{array}
$$

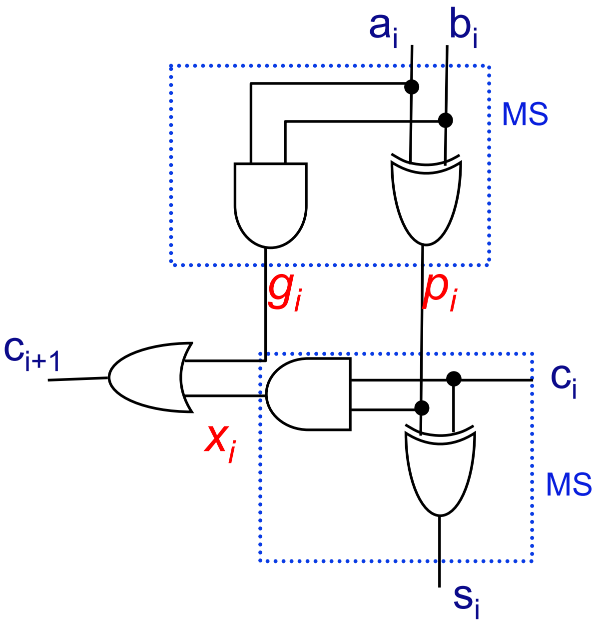
Generalizando...

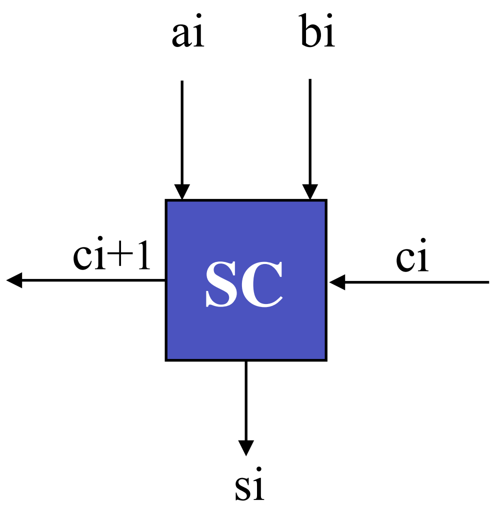


Com i >= 1

1. Projeto de Unidade Lógico-Aritmética

O Somador Completo





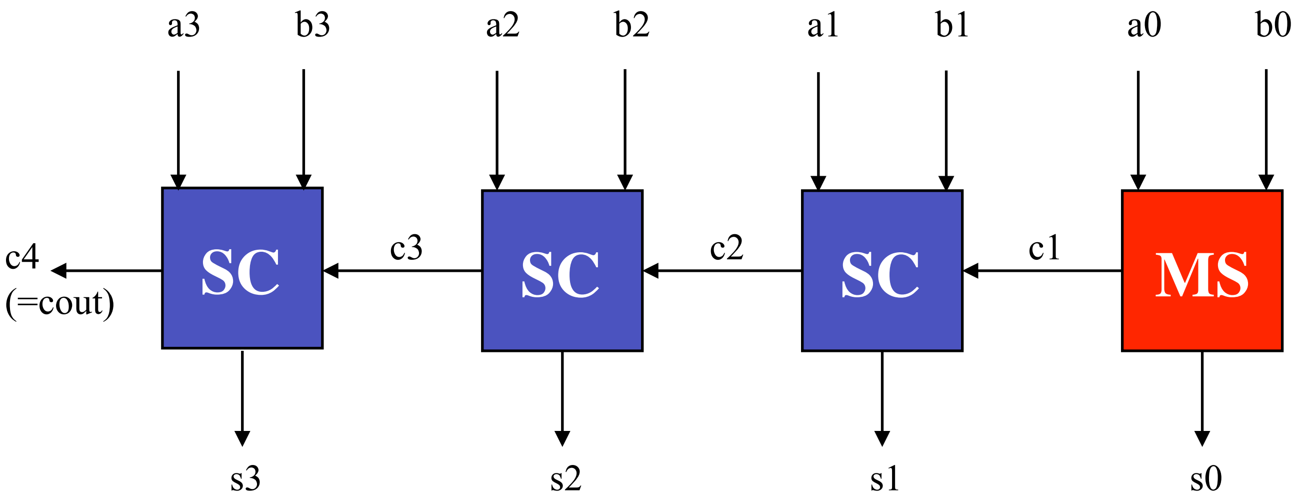
$si = ci \oplus ai \oplus bi$

$ci+1 = ai \cdot bi + ai \cdot ci + bi \cdot ci$

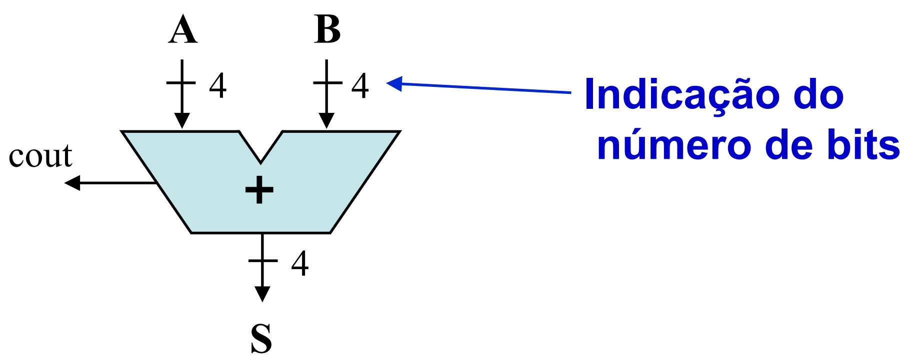
1. Projeto de Unidade Lógico-Aritmética

O Somador Paralelo *Carry-Ripple* (de 4 Bits)

Diagrama de Blocos (Nível Lógico)



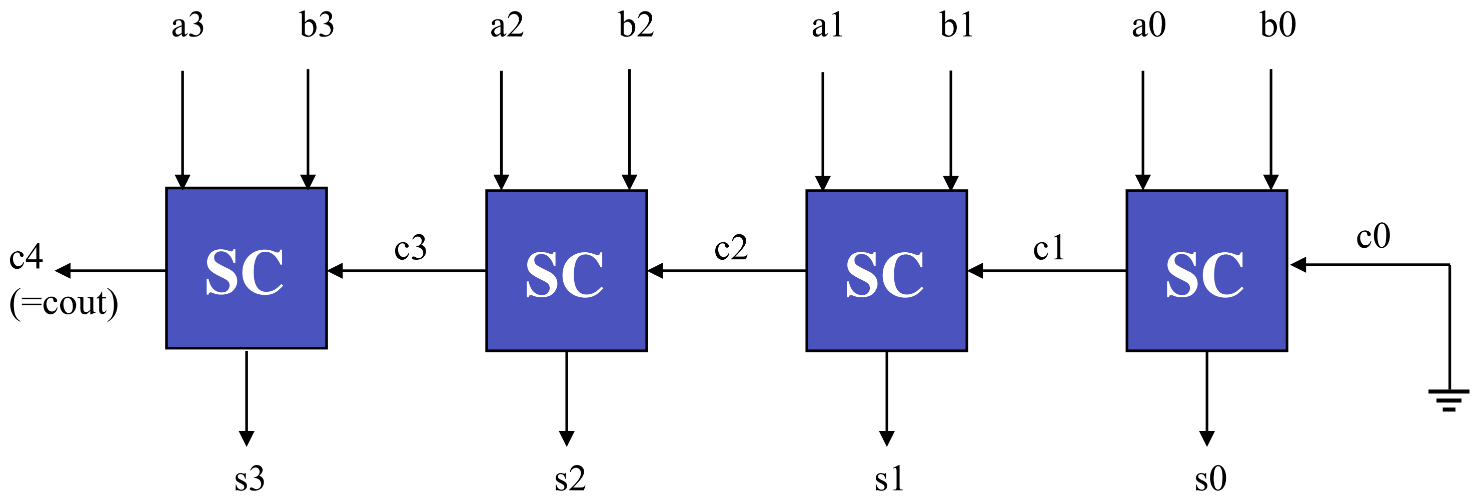
Símbolo no Nível RT



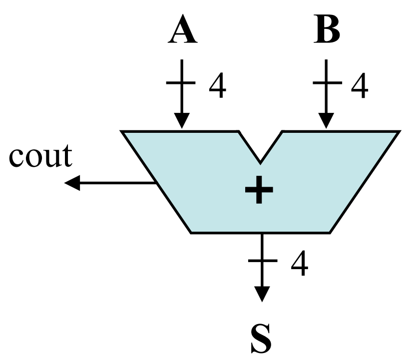
1. Projeto de Unidade Lógico-Aritmética

O Somador Paralelo *Carry-Ripple* (de 4 Bits)

Diagrama de Blocos (Nível Lógico): versão 2



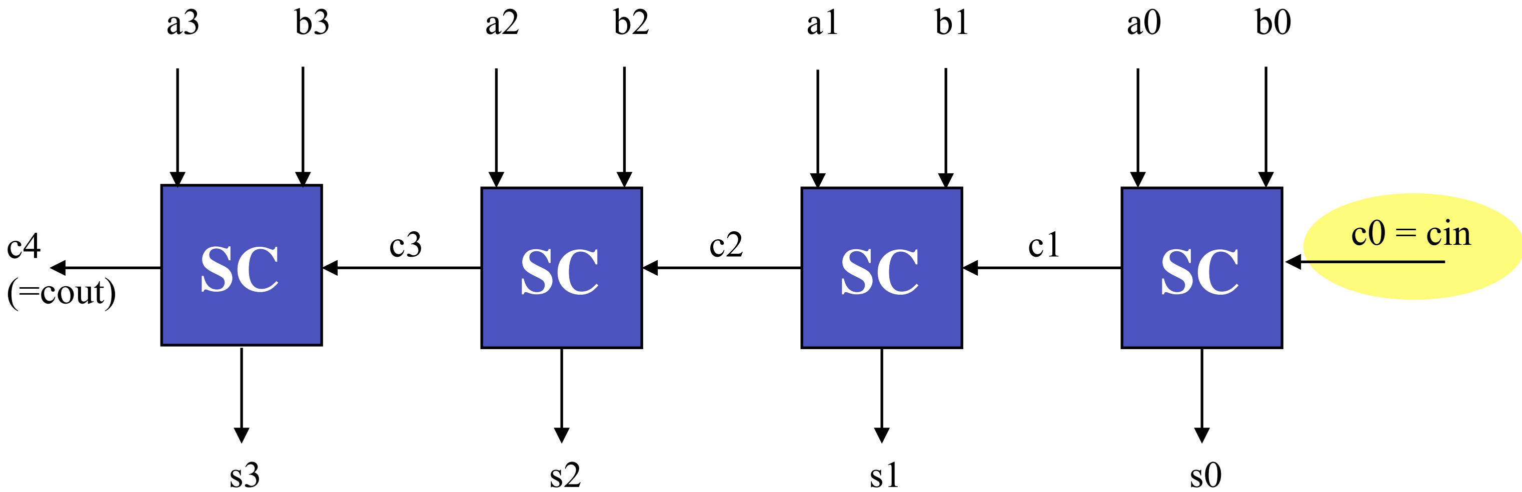
Símbolo no Nível RT



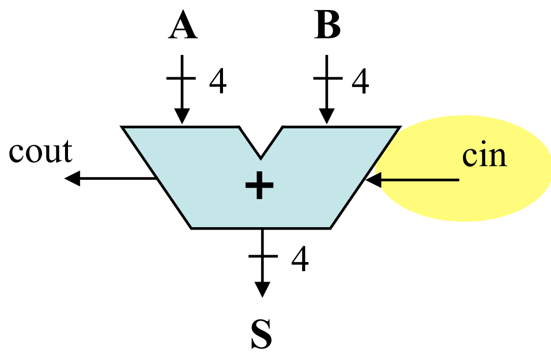
1. Projeto de Unidade Lógico-Aritmética

O Somador Paralelo *Carry-Ripple* (de 4 Bits)

Diagrama de Blocos (Nível Lógico): versão 3



Símbolo no Nível RT

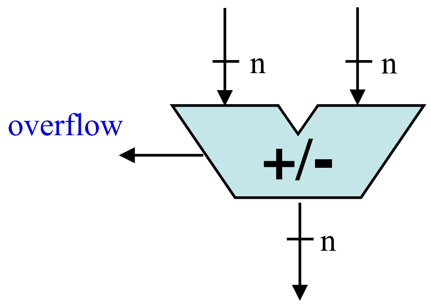


1. Projeto de Unidade Lógico-Aritmética

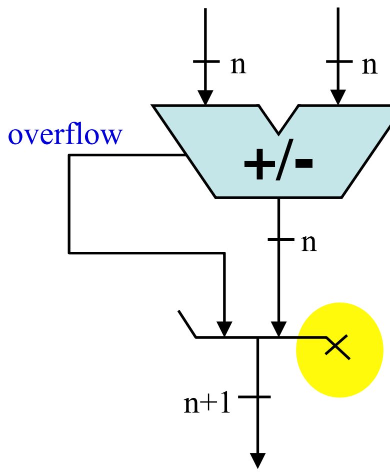
Representando Dados em Circuitos Digitais

Mais Convenções

Somador-subtrator para operandos com *n* bits cada



Indicando como um número de *n+1* bits é composto

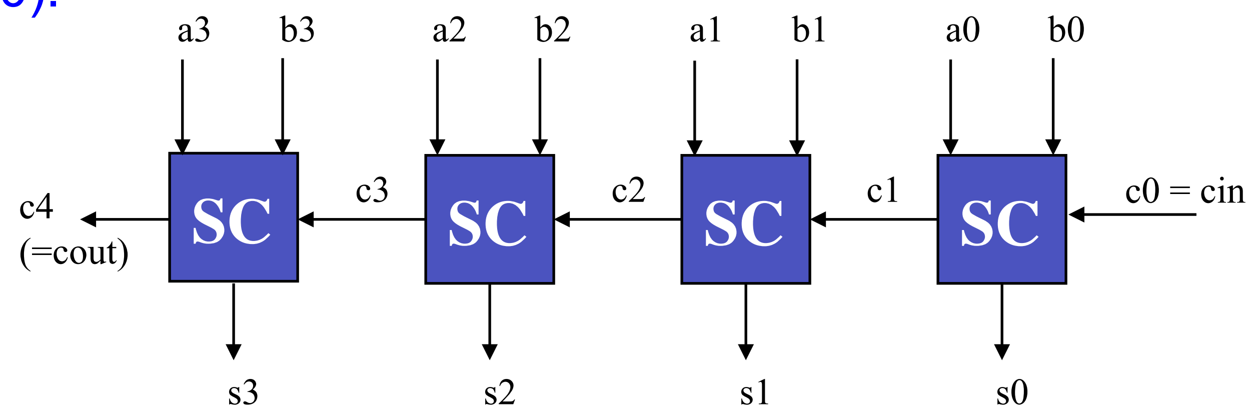


Convenção para indicar que o bit menos significativo está mais à direita

1. Projeto de Unidade Lógico-Aritmética

Somador *Carry-Ripple*: Custo x Desempenho

- **A estrutura do somador *carry-ripple* baseia-se na fatoração da expressão do *carry*. A consequência disto é:**
  - Redução do custo (i.e., menor número de portas lógicas)
  - Aumento do atraso (o cálculo de c4 depende de c3, que depende de c2, que depende de c1, que depende apenas das entradas c0, a0 e b0).



Os slides complementares "somadores rápidos" mostra outros tipos de somadores.

Representação de Inteiros em Binário

Como Representar Inteiros Negativos?

Sinal-magnitude

sinal

+3 = 0 0 1 1
-3 = 1 0 1 1

Complemento de 1

sinal

+3 = 0 0 1 1
-3 = 1 1 0 0

Obtido a partir do +3, aplicando-se a inversão bit a bit

Complemento de 2

sinal

+3 = 0 0 1 1
-3 = 1 1 0 1

Obtido a partir do +3, aplicando-se a inversão bit a bit e após, somando-se 1 (uma unidade)

Representação de Inteiros em Binário

Números Inteiros com Sinal

- Para facilitar a construção de circuitos aritméticos, os negativos são representados em complemento de dois
- Assumindo-se números com **4 bits**

	binário	**decimal**
Menor número	**1000**	**-8**
Zero	**0000**	**0**
Maior número	**0111**	**+7**

Intervalo de representação: [-8 , +7]

Representação de Inteiros em Binário

Números Inteiros com Sinal

Observe que:

$0111_2 = 1x2^0 + 1x2^1 + 1x2^2 + 0x2^3 =$

$= 1 + 2 + 4 + 0 =$

$= \mathbf{+7}$

Observe também que:

$1111_2 = 1x2^0 + 1x2^1 + 1x2^2 - 1x2^3 =$

$= 1 + 2 + 4 \mathbf{- 8} =$

$= \mathbf{-1}$

O bit mais à esquerda representa o sinal: se ele valer 1, o número é negativo. Caso contrário, o número é positivo ou zero.

Representação de Inteiros em Binário

Números Inteiros com Sinal

- assumindo-se números com **8 bits**

	binário	decimal
Menor número	10000000	-128
Zero	00000000	0
Maior número	01111111	+127

Intervalo de representação: [-128 , +127]

Representação de Inteiros em Binário

Números Inteiros com Sinal

- Generalizando-se para **n bits**

	binário	decimal
Menor número	**1000...0**	-2^{n-1}
Zero	**0000...0**	**0**
Maior número	**0111...1**	$+(2^{n-1}-1)$

Intervalo de representação: $[\ -2^{n-1}\ ,\ +(2^{n-1}-1)\ ]$

Adição de Inteiros com Sinal

Assumindo:

- Negativos são representados em complemento de dois
- Números com **4 bits**

	binário	decimal
Menor número	1000	-8
Zero	0000	0
Maior número	0111	+7

Intervalo de representação: [-8 , +7]

1. Projeto de Unidade Lógico-Aritmética

Adição de Inteiros com Sinal

(Assumindo Negativos em Complemento de 2)

Exemplo 3: dois números positivos, cuja soma $\in$ [-8,+7]

	0 0 0 0		transporte (*carry*)
	0 0 1 0	(+2)	
+	0 1 0 0	(+4)	
	0 1 1 0	(+6)	resultado correto

1. Projeto de Unidade Lógico-Aritmética

Adição de Inteiros com Sinal

(Assumindo Negativos em Complemento de 2)

Exemplo 4: dois números negativos, cuja soma seja ≥ -8

	1 1 0 0		transporte (*carry*)
	1 1 1 0	(-2)	
+	1 1 0 0	(-4)	
	1 0 1 0	(-6)	resultado correto

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

Adição de Inteiros com Sinal

(Assumindo Negativos em Complemento de 2)

Exemplo 5: um número positivo e um número negativo, tais que o resultado é positivo

1 1 1 1		transporte (*carry*)
0 1 1 1	(+7)	
+ 1 1 1 1	(-1)	
0 1 1 0	(+6)	resultado correto

Novamente…

Adição de Inteiros com Sinal

(Assumindo Negativos em Complemento de 2)

Exemplo 6: um número positivo e um número negativo, tais que o resultado é negativo

	0 0 0 1		transporte (*carry*)
	1 0 0 1	(-7)	
+	0 0 0 1	(+1)	
	1 0 1 0	(-6)	resultado correto

Adição de Inteiros com Sinal

(Assumindo Negativos em Complemento de 2)

Exemplo 7: um positivo e um 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

E novamente…

1. Projeto de Unidade Lógico-Aritmética

Adição de Inteiros com Sinal

(Assumindo Negativos em Complemento de 2)

Exemplo 8: 2 números positivos

	0 1 0 0	transporte (*carry*)
	0 1 0 0	(+4)
+	0 1 0 1	(+5)
	1 0 0 1	(-7)

Resultado errado !

ocorre *overflow* quando esses 2 bits são diferentes

o resultado excede o intervalo de representação = *overflow*

1. Projeto de Unidade Lógico-Aritmética

Adição de Inteiros com Sinal

(Assumindo Negativos em Complemento de 2)

Exemplo 9: 2 números negativos

	1 0 0 0	transporte (*carry*)
	1 1 0 0	(-4)
+	1 0 1 1	(-5)
	0 1 1 1	(+7)

Resultado errado !

ocorre *overflow* quando esses 2 bits são diferentes

o resultado excede o intervalo de representação = *overflow*

1. Projeto de Unidade Lógico-Aritmética

Adição de Inteiros com Sinal

(Assumindo Negativos em Complemento de 2)

Conclusões:

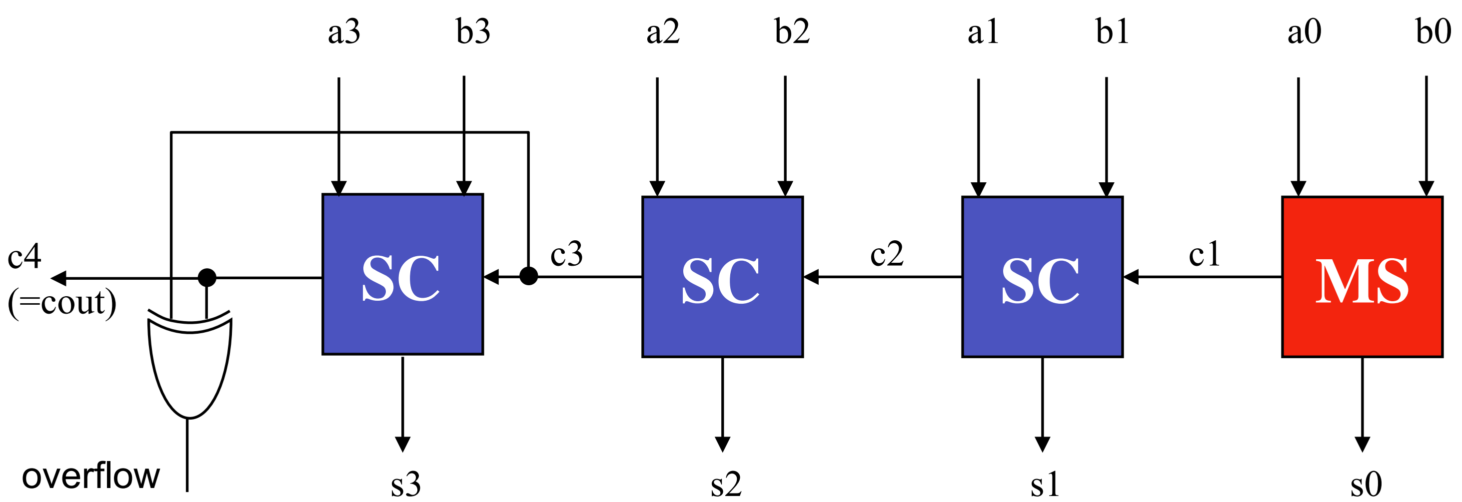
- **Números binários em complemento de 2 podem ser adicionados como se fossem números binários sem sinal!**
- **Neste caso, a detecção de *overflow* se dá comparando-se os dois últimos sinais de *carry***

O Somador Paralelo *Carry-Ripple* (de 4 Bits)

Modificado para Operar Sobre Números com Sinal

(Assumindo negativos em complemento de 2)

Diagrama de Blocos (Nível Lógico)



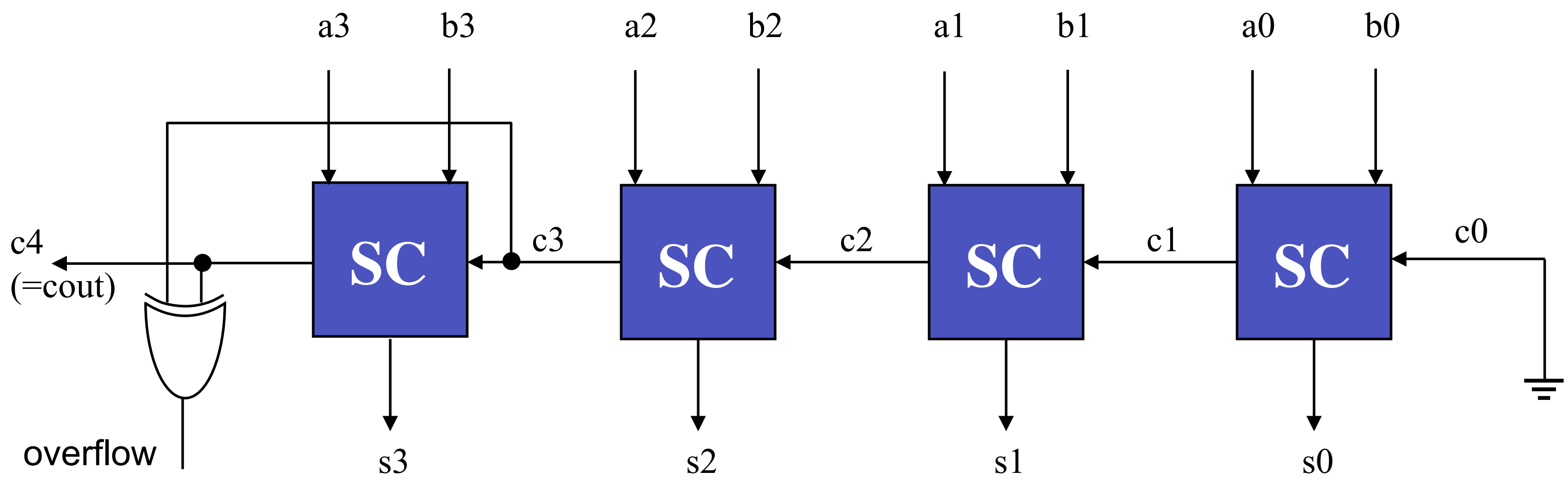
1. Projeto de Unidade Lógico-Aritmética

O Somador Paralelo *Carry-Ripple* (de 4 Bits)

Modificado para Operar Sobre Números com Sinal

(Assumindo negativos em complemento de 2)

Diagrama de Blocos (Nível Lógico): versão 2

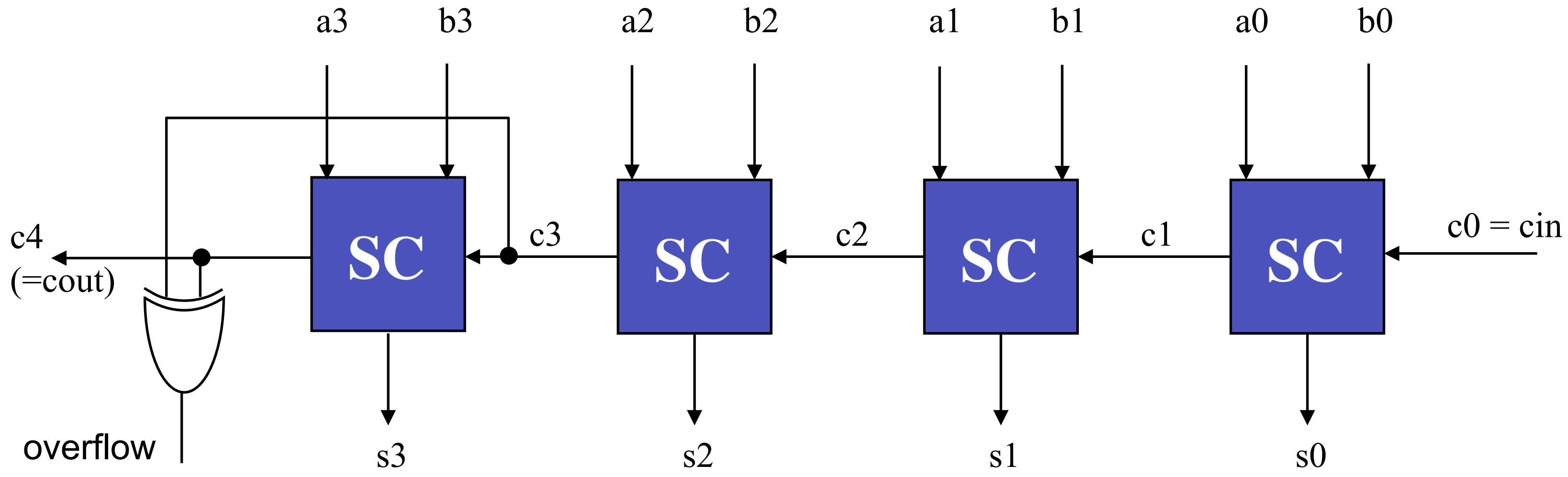


O Somador Paralelo *Carry-Ripple* (de 4 Bits)

Modificado para Operar Sobre Números com Sinal

(Assumindo negativos em complemento de 2)

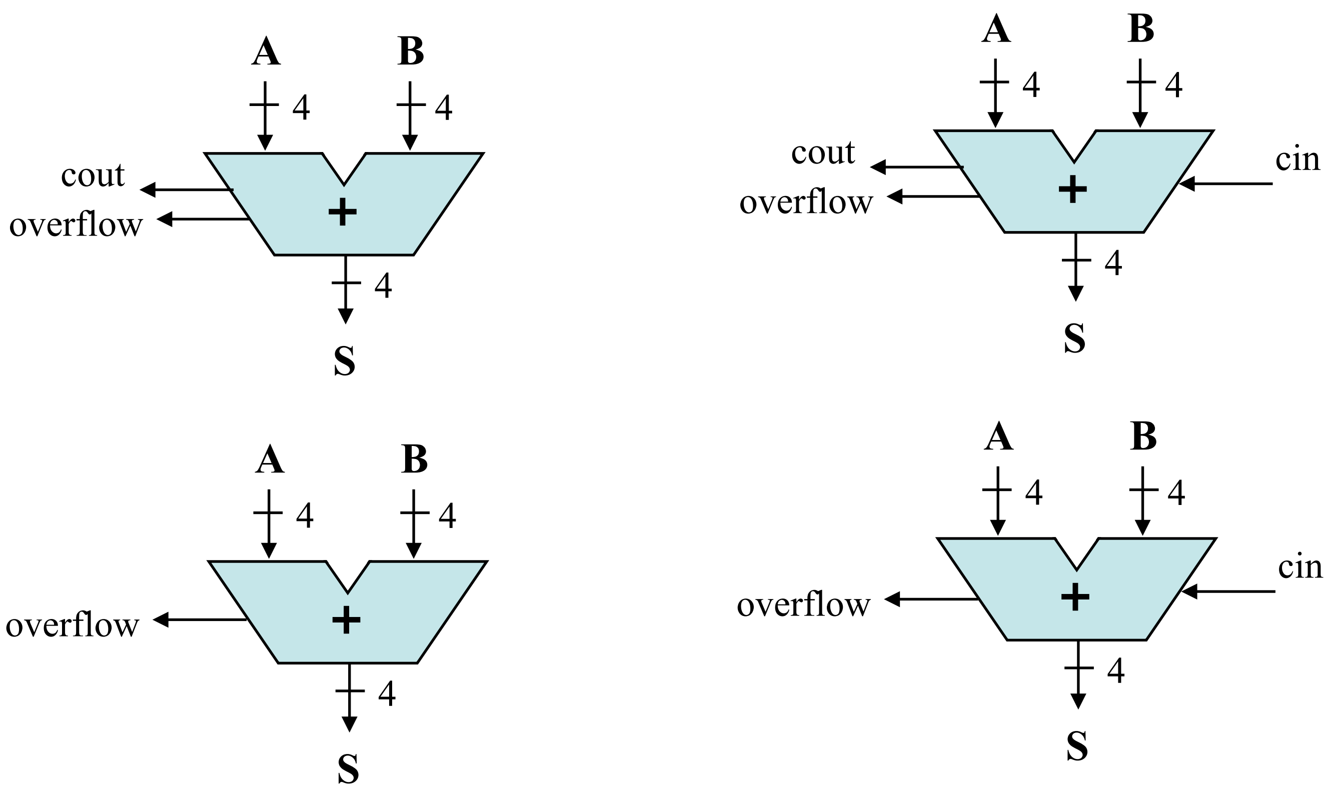
Diagrama de Blocos (Nível Lógico): versão 3



1. Projeto de Unidade Lógico-Aritmética

O Somador Paralelo *Carry-Ripple* (de 4 Bits)

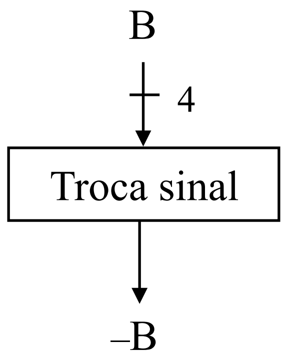
Símbolos no Nível RT



Circuitos Aritméticos

Exercício 4: **Usando o somador *carry-ripple*, projetar um circuito combinacional que troca o sinal de um número inteiro de 4 bit.**

Interfaces:



Circuitos Aritméticos

Exercício 4: **Solução**

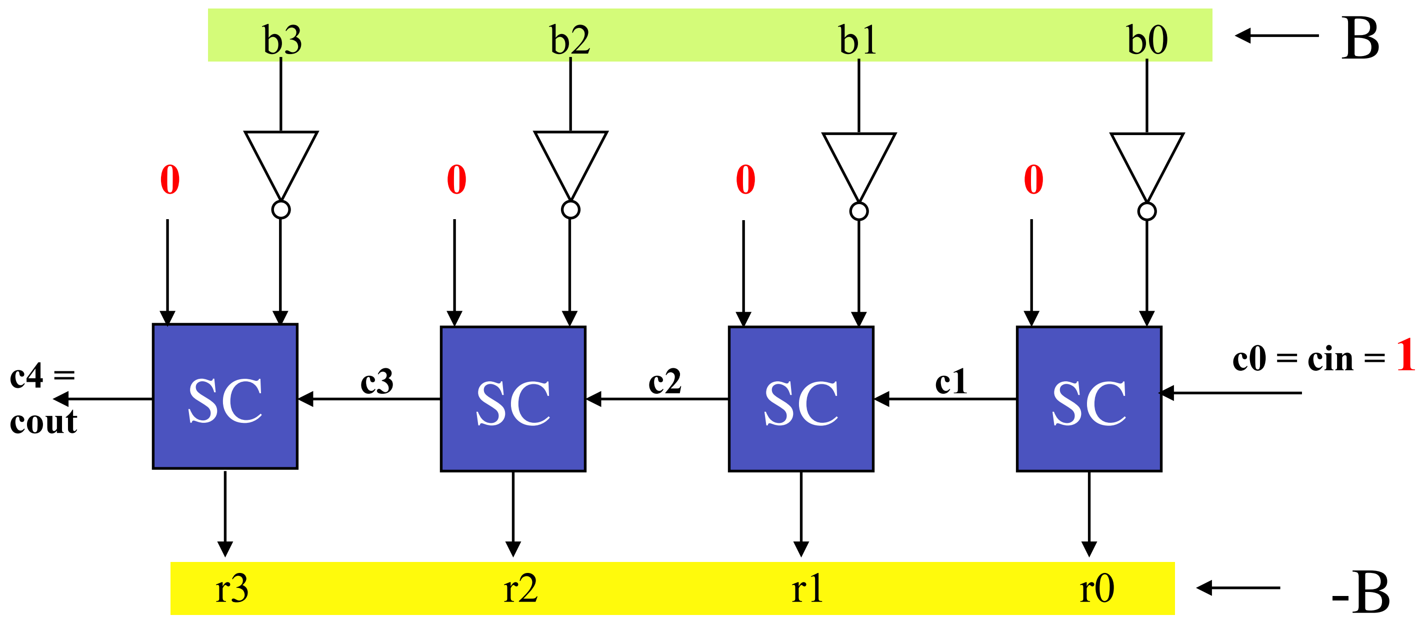
Trocar o sinal significa aplicar as regras do complemento de dois ao número, ou seja:

1. Negar (“NOT”) bit a bit o número

2. Somar uma unidade ao resultado do passo anterior

Circuitos Aritméticos

Exercício 4: Solução



1. Projeto de Unidade Lógico-Aritmética

Subtração de Números Inteiros em Binário

Princípio

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

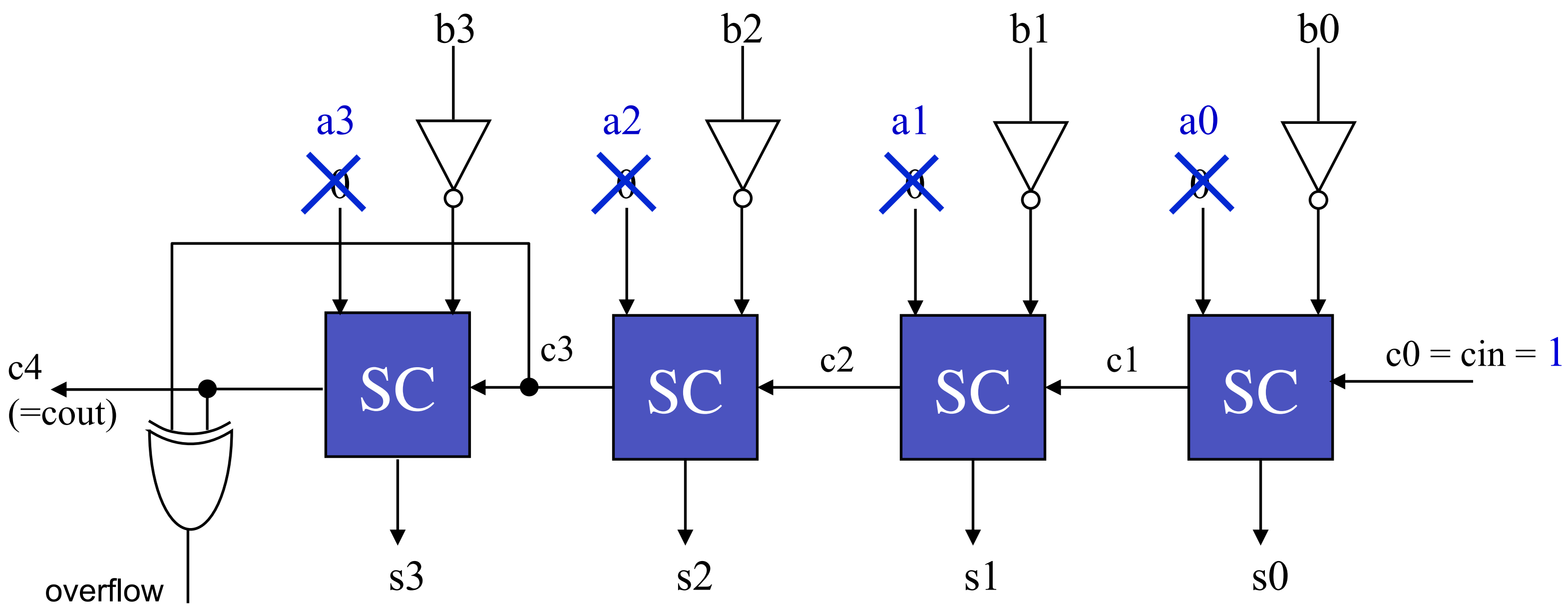
Onde −B é o número B de sinal trocado!

Ora, que coincidência!! (Ou não?)

1. Projeto de Unidade Lógico-Aritmética

Subtrator Paralelo (de 4 bits)

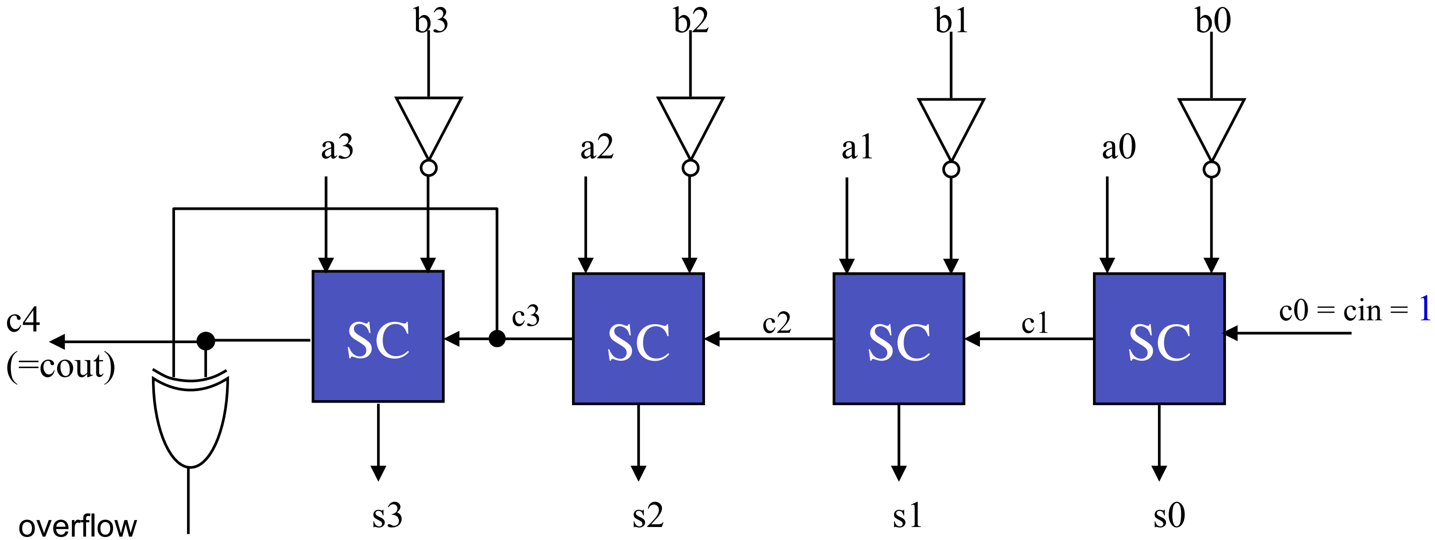
A − B = A + (− B)



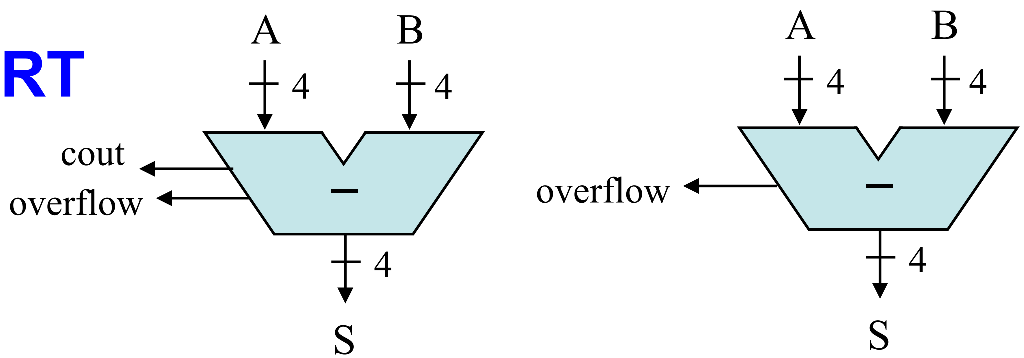
1. Projeto de Unidade Lógico-Aritmética

Subtrator Paralelo (de 4 bits)

Diagrama de Blocos (Nível Lógico)



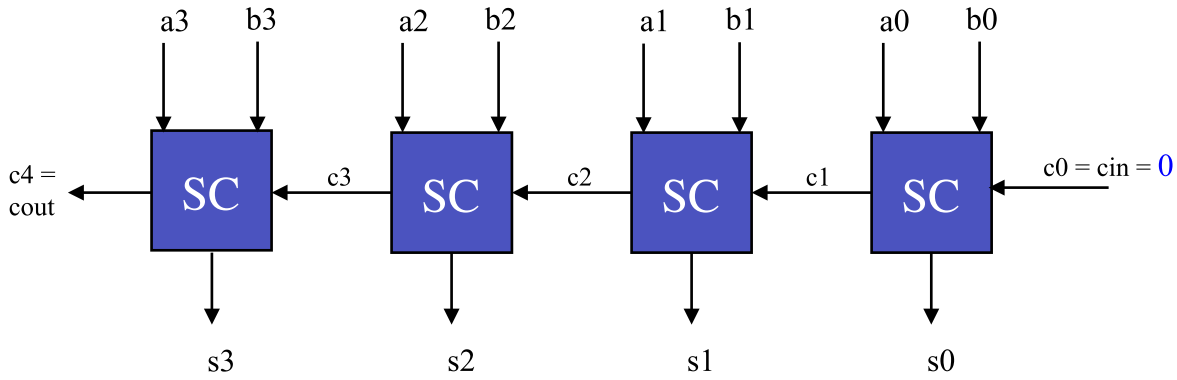
Símbolos no Nível RT

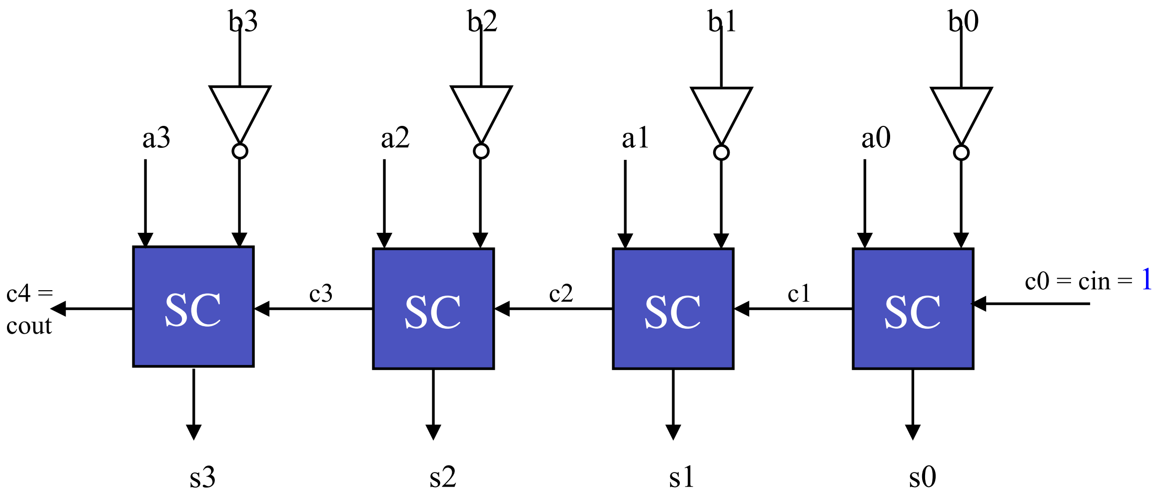


1. Projeto de Unidade Lógico-Aritmética

Subtrator/Subtrator Paralelo (de 4 bits)

Somador



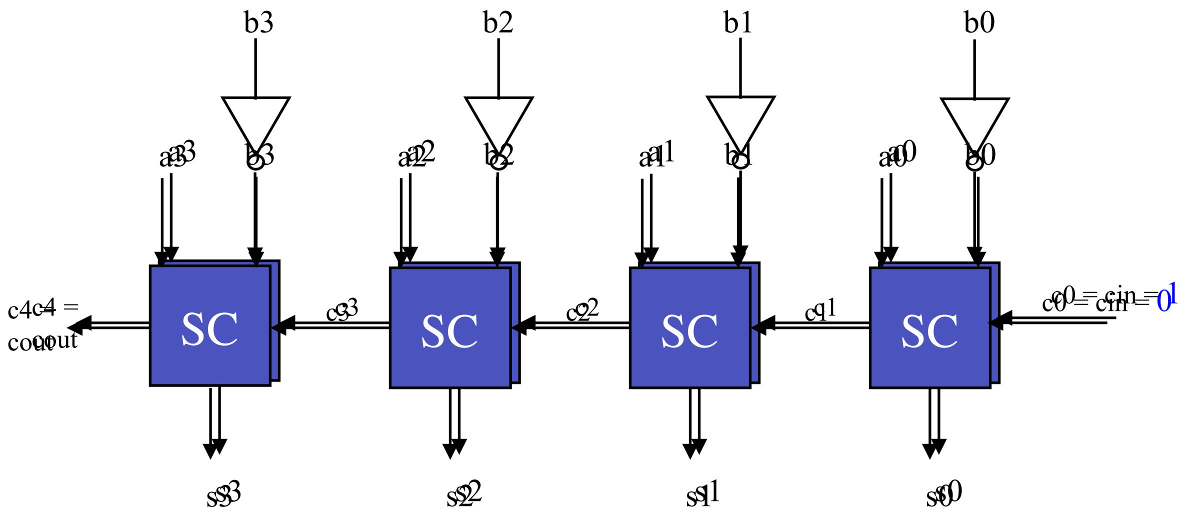


Subtrator

1. Projeto de Unidade Lógico-Aritmética

Subtrator/Subtrator Paralelo (de 4 bits)

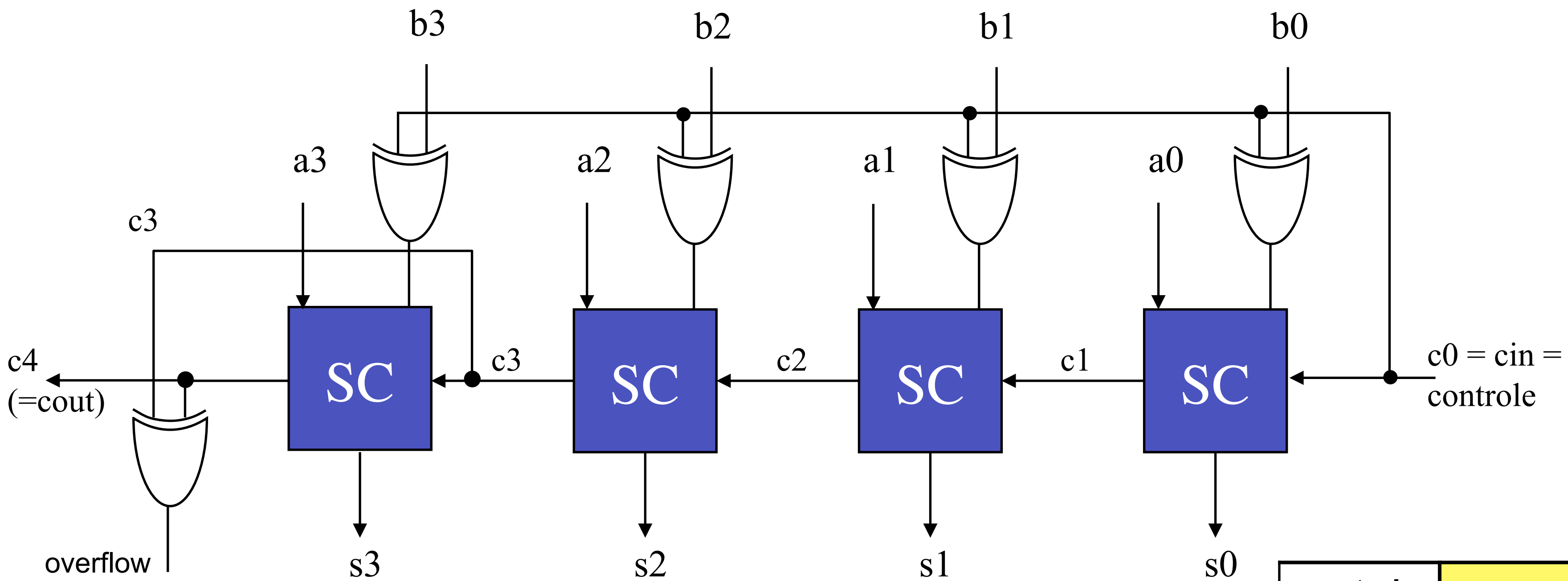
Como uni-los em um único circuito, configurável?



1. Projeto de Unidade Lógico-Aritmética

Subtrator/Subtrator Paralelo (de 4 bits)

Resposta!!!



controle	operação
0	S=A+B
1	S=A-B

1. Projeto de Unidade Lógico-Aritmética

Subtrator/Subtrator Paralelo (de 4 bits)

Símbolo no Nível RT

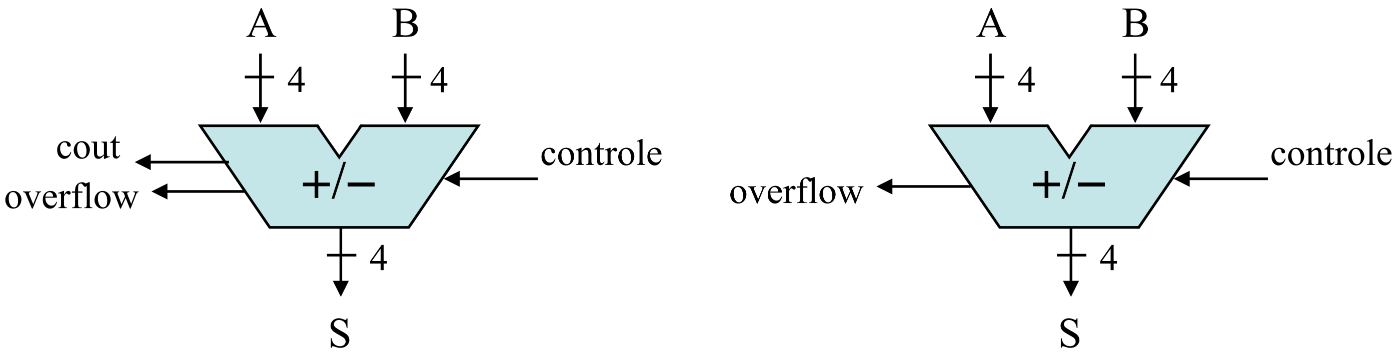


Tabela de Operação

controle	operação
0	S=A+B
1	S=A−B

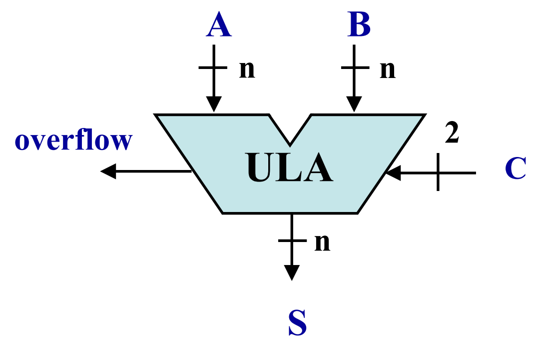
1. Projeto de Unidade Lógico-Aritmética

ULA Simples

Suponha que se necessite de uma Unidade Lógico-Aritmética (ULA) capaz de realizar as seguintes operações

C1	C0	operação	comentário
0	0	S = A + B	adição
0	1	S = A - B	subtração
1	0	S = A AND B	"E" bit a bit
1	1	S = A OR B	"OU" bit a bit

Símbolo no nível RT



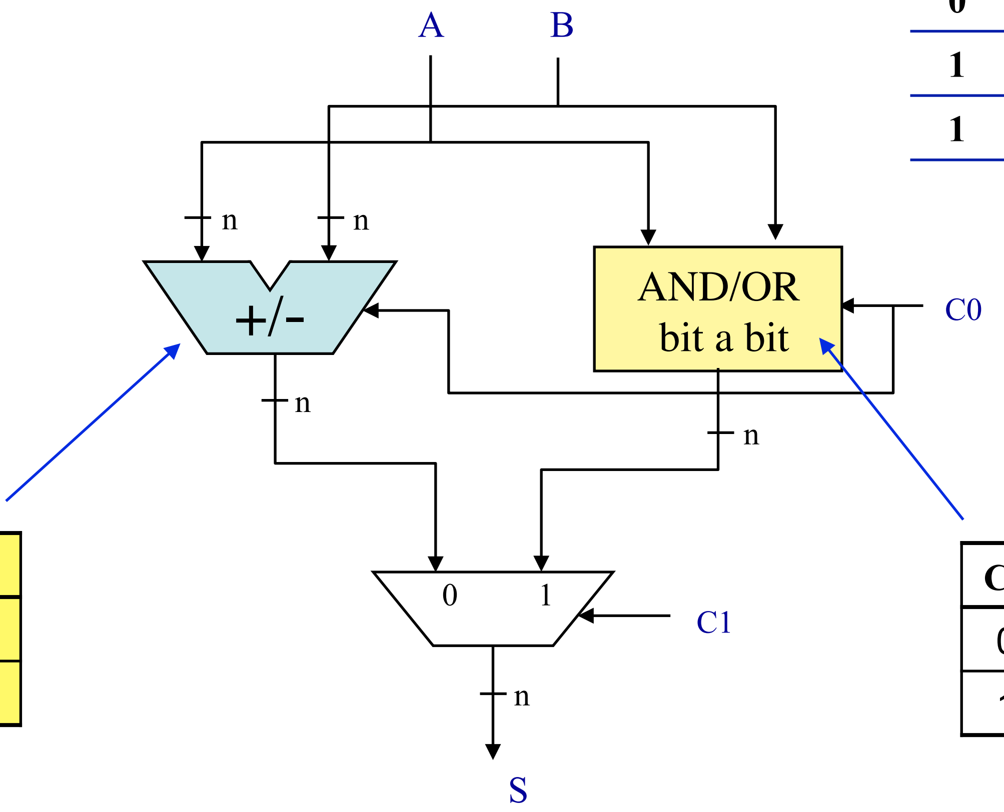
Obs: o sinal de overflow pode ou não ser necessário…

1. Projeto de Unidade Lógico-Aritmética

ULA Simples

Visão Geral desta ULA

C1	C0	operação
0	0	S = A + B
0	1	S = A - B
1	0	S = A AND B
1	1	S = A OR B



C0	operação
0	S=A+B
1	S=A-B

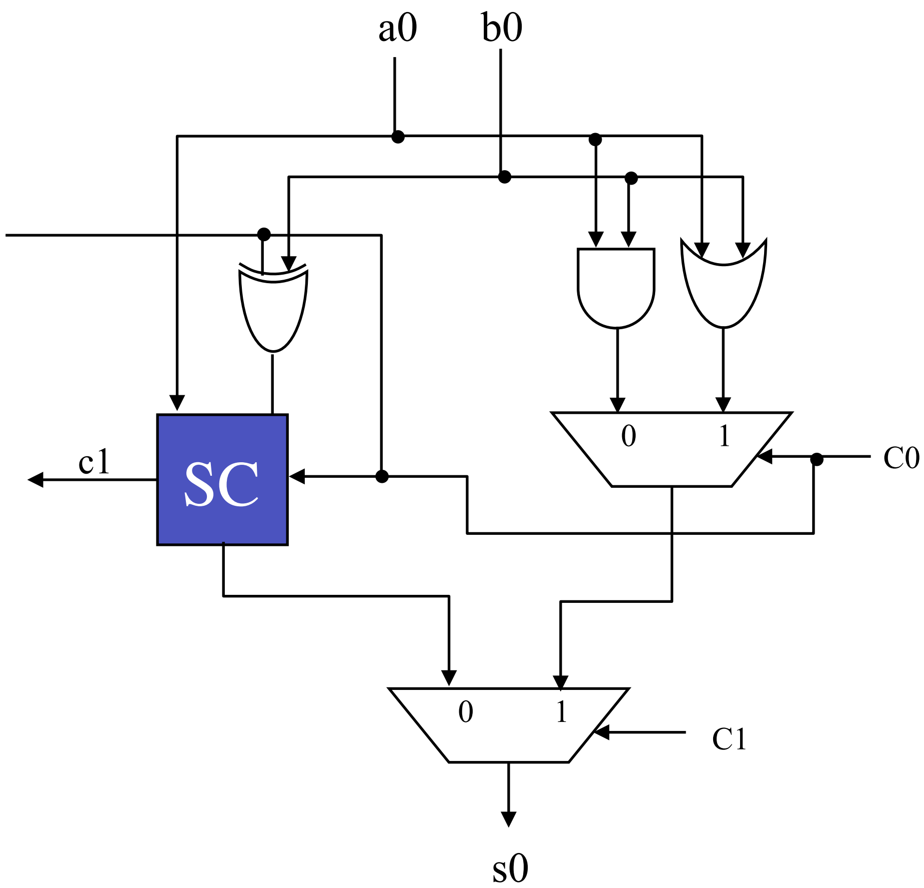
C0	operação
0	S=A **AND** B
1	S=A **OR** B

1. Projeto de Unidade Lógico-Aritmética

ULA Simples

Visão de um bit desta ULA (os demais bits serão similares)

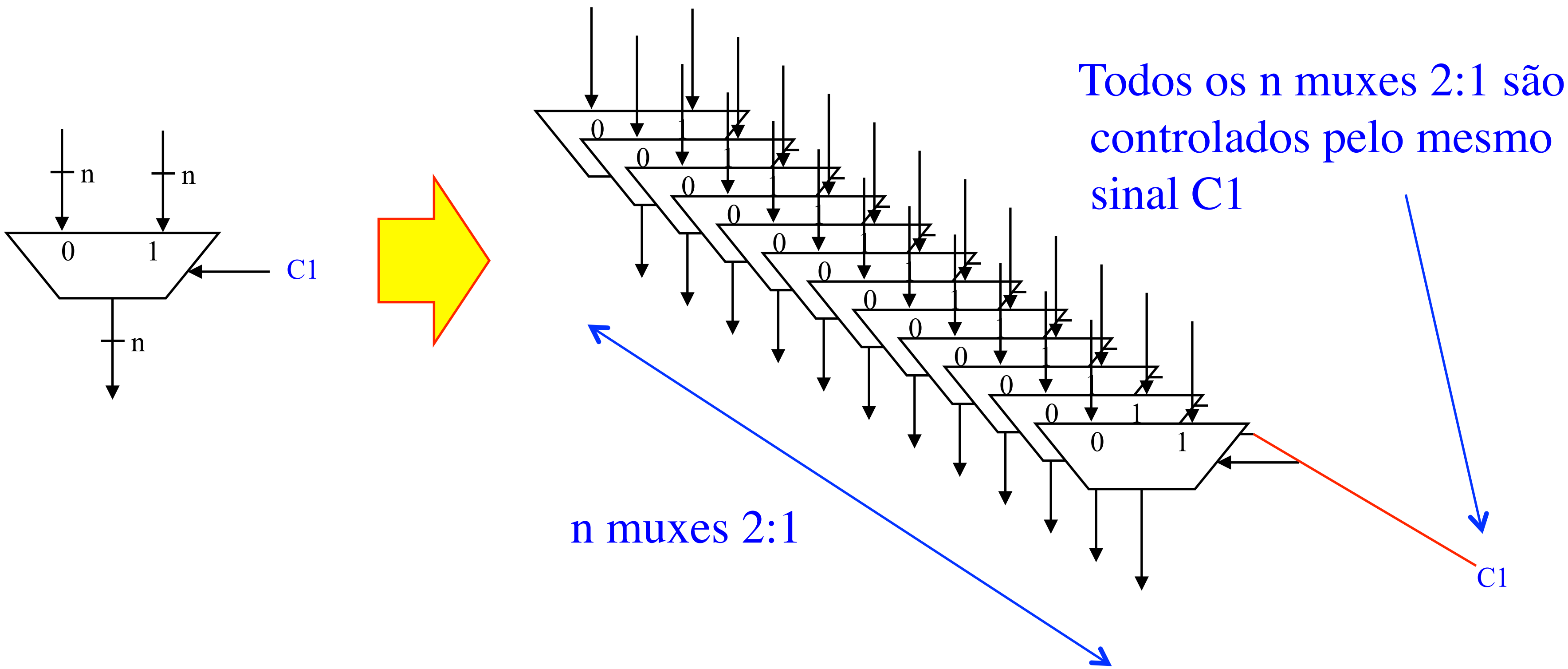
C1	C0	operação
0	0	S = A + B
0	1	S = A - B
1	0	S = A AND B
1	1	S = A OR B



1. Projeto de Unidade Lógico-Aritmética

ULA Simples

Multiplexador no Nível RT...

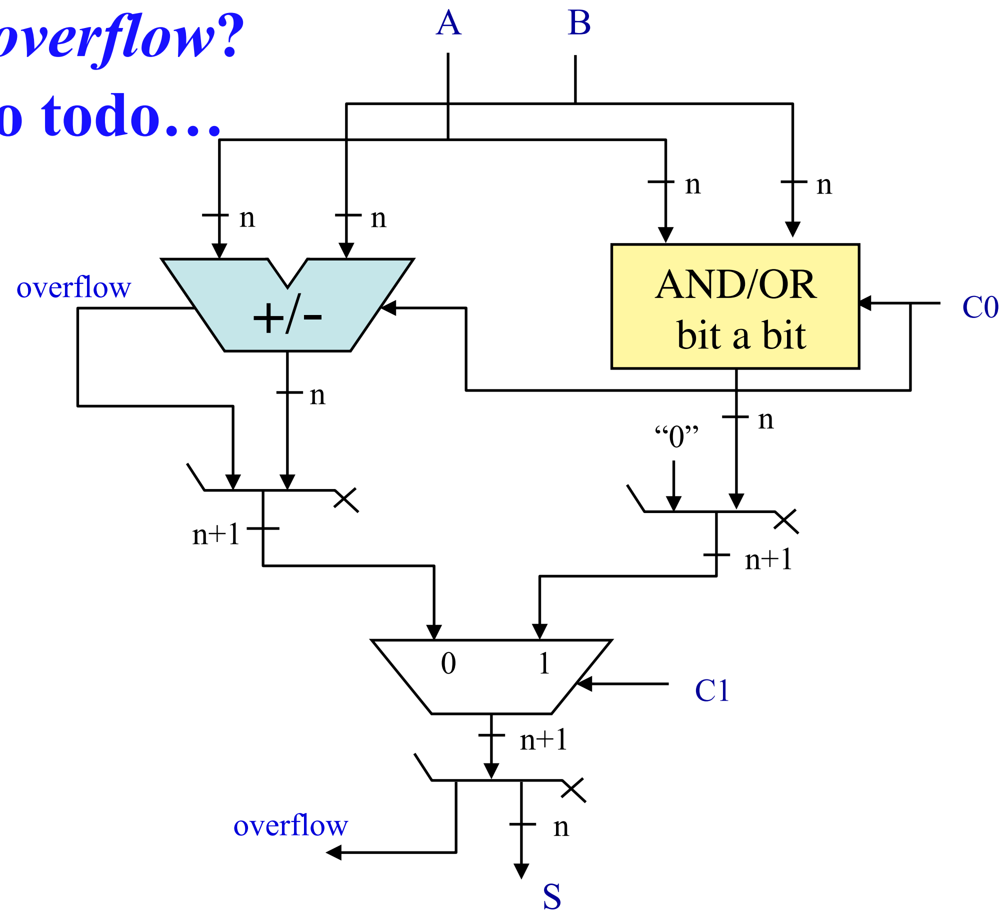


1. Projeto de Unidade Lógico-Aritmética

ULA Simples

Mas onde foi parar o *overflow*?

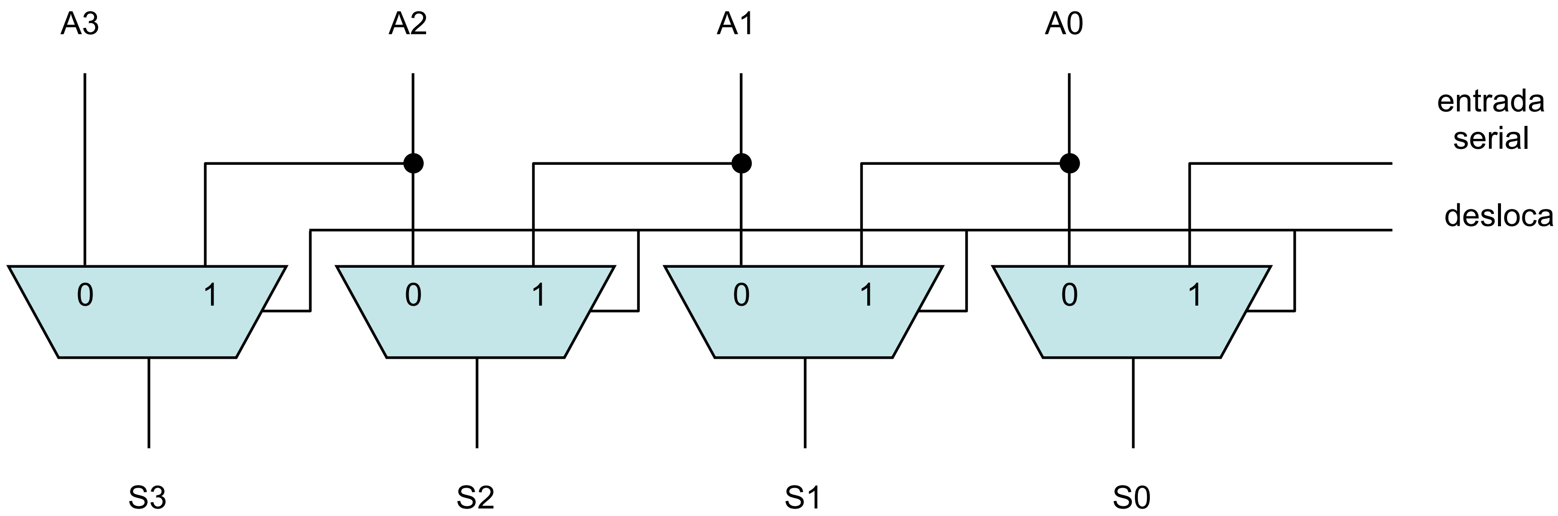
Voltando ao projeto do todo…



1. Projeto de Unidade Lógico-Aritmética

Deslocador Combinacional

Um deslocador (*shifter*) com uso de multiplexadores 2:1

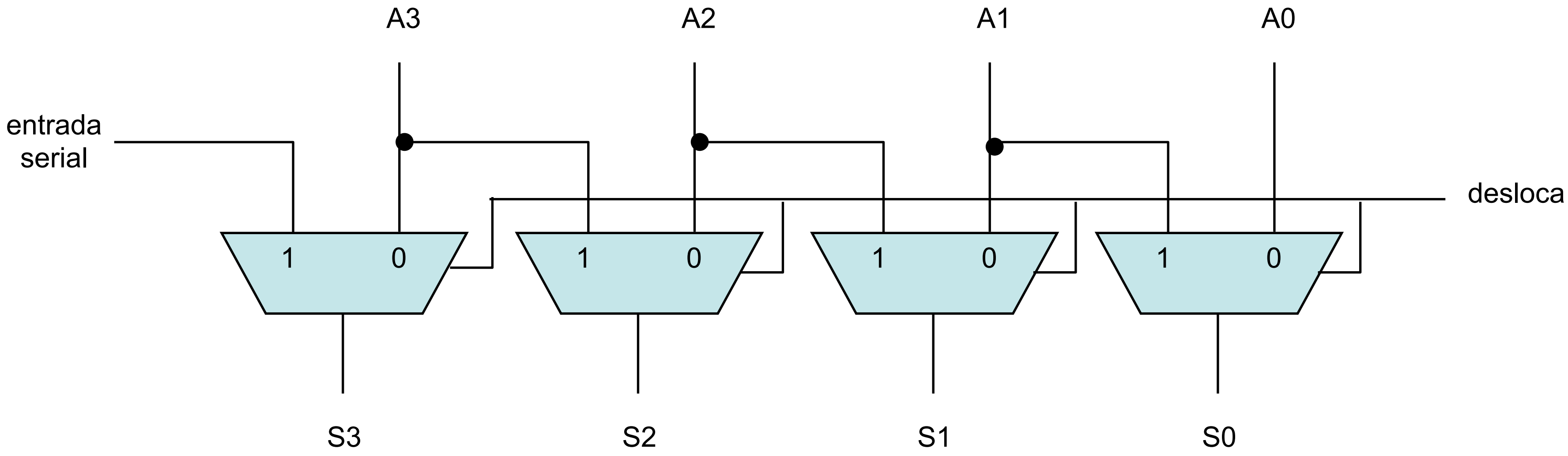


- Se desloca=1, este circuito desloca cada bit uma posição para a esquerda
- Qual é o significado desta operação?

1. Projeto de Unidade Lógico-Aritmética

Deslocador Combinacional

Outro deslocador (*shifter*) com uso de multiplexadores 2:1



- Se desloca=1, este circuito desloca cada bit uma posição para a direita
- Qual é o significado desta operação?

1. Projeto de Unidade Lógico-Aritmética

Multiplicação com Circuito Combinacional

O Multiplicador Matricial

- É uma implementação direta do esquema ao lado
- Cada bit dos produtos parciais é gerado por meio de um "E" lógico

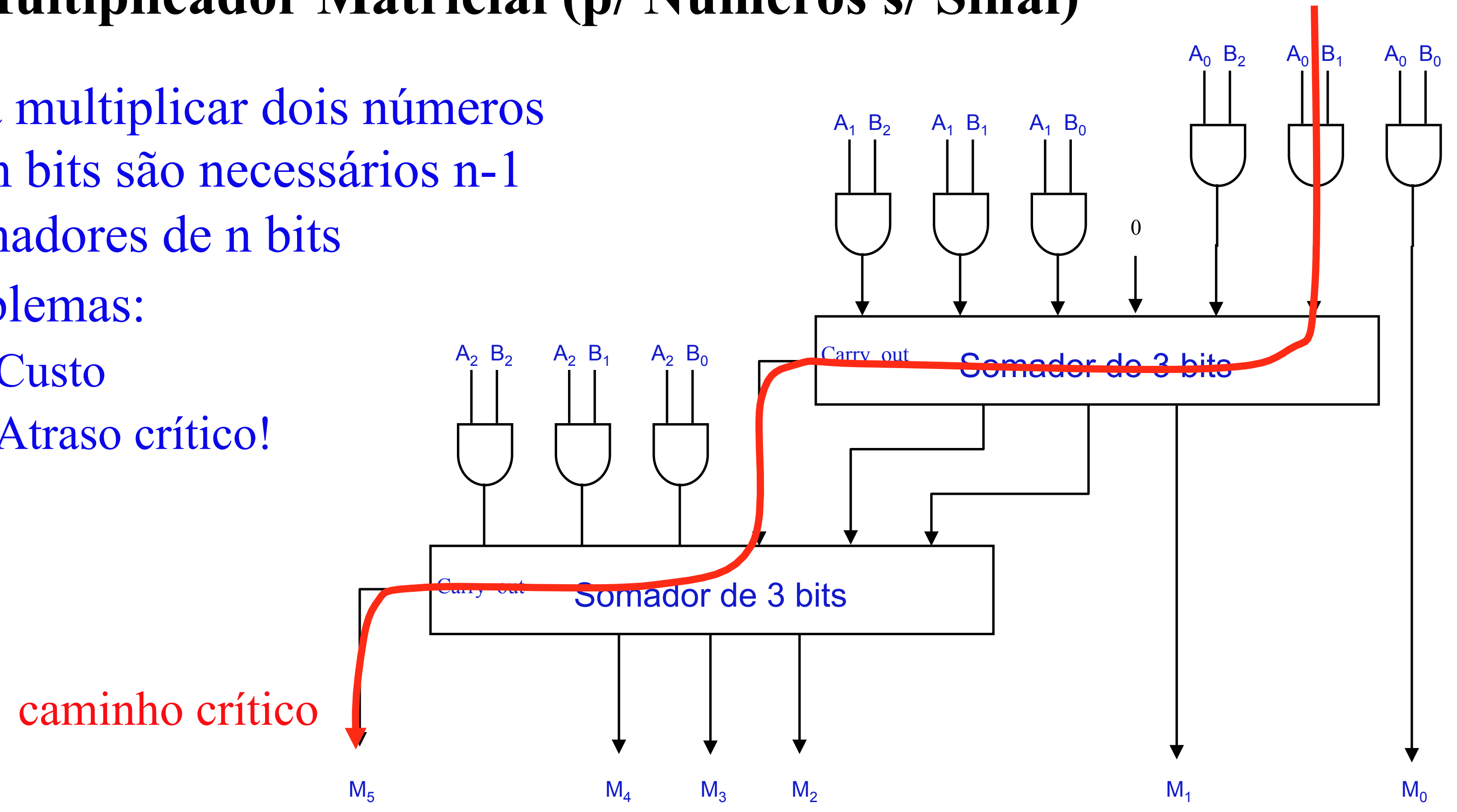
```
          1 0 0 1      multiplicando
x         1 0 1 1      multiplicador
-----------------
          1 0 0 1   }
        1 0 0 1 -   }  produtos
+     0 0 0 0 - -   }  parciais
    1 0 0 1 - - -   }
-----------------
  1 1 0 0 0 1 1        resultado
```

1. Projeto de Unidade Lógico-Aritmética

Multiplicação com Circuito Combinacional

O Multiplicador Matricial (p/ Números s/ Sinal)

- Para multiplicar dois números de n bits são necessários n-1 somadores de n bits
- Problemas:
  - Custo
  - Atraso crítico!



1. Projeto de Unidade Lógico-Aritmética

Multiplicação com Circuito Combinacional

O Multiplicador Matricial (p/ Números s/ Sinal)

O Símbolo no Nível RT