



FEDERAL UNIVERSITY
OF SANTA CATARINA

EEL5105 – Circuitos e Técnicas Digitais

Aula 4

Prof. Héctor Pettenghi

hector@eel.ufsc.br

<http://hectorpettenghi.paginas.ufsc.br>

Aritmética com Números Binários

4.1. Soma de Números Binários

4.2. Representando Números com Sinal

4.3. Subtração usando Complemento de 2

4.4. Somador/Subtrator

4.5. *Overflow*

2

Nesta aula iremos estudar a parte de aritmética com Números Binários. Veremos como funciona a soma de números binários, a representação de números com sinal, a subtração usando complemento de 2, o que é e como é o circuito de um somador/subtrator e como detectar a condição de overflow.

4.1. Soma de Números Binários

- 4.2. Representando Números com Sinal
- 4.3. Subtração usando Complemento de 2
- 4.4. Somador/Subtrator
- 4.5. Overflow

3

Agora, veremos como funciona a soma de números binários.

4.1. Soma de Números Binários

- **Pergunta:** como você faria o projeto de um circuito digital que faz a soma de dois números binários de 8 bits (ou seja, de dois números com valores entre 0 e 255)?

4

Como seria uma possível solução para esse problema?

4.1. Soma de Números Binários

- **Pergunta:** como você faria o projeto de um circuito digital que faz a soma de dois números binários de 8 bits (ou seja, de dois números com valores entre 0 e 255)?
 - **Possível solução:** o circuito tem 16 entradas, então uma tabela verdade com 2^{16} linhas poderia ser montada e o projeto seria feito, por exemplo, por soma de minitermos.
 - $2^{16} = \textcolor{red}{65535 \text{ linhas!!!}}$
 - Tabela gigantesca para ser utilizada na prática...
 - Imagine se o projeto fosse de um somador de dois números de 16 bits ($2^{32} = \textcolor{red}{4.294.967.296}$ linhas na tabela verdade).

5

Uma possível solução seria fazer uma tabela verdade com 2^{16} linhas, e então faríamos a soma dos minitermos. O problema é que uma tabela deste tamanho teria 65535 linhas, e seria o oposto de praticidade! Imagine então para um somador com dois número de 16 bits, teríamos mais de 4 bilhões de linhas na tabela verdade.

4.1. Soma de Números Binários

- Soma **binária** pode ser feita de forma similar à soma **decimal**:

Decimal	Binário
$\begin{array}{r} & 1 \\ 1 & 6 & 3 \\ + & 8 & 9 & 2 \\ \hline 1 & 0 & 5 & 5 \end{array}$	$\begin{array}{r} & 1 & 1 \\ 0 & 1 & 0 & 1 \\ + & 1 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 0 \end{array}$

6

Para tornar esse procedimento prático, trabalhamos na soma binária de forma similar à soma decimal, como apresenta o slide.

4.1. Soma de Números Binários

- Soma **binária** pode ser feita de forma similar à soma **decimal**:

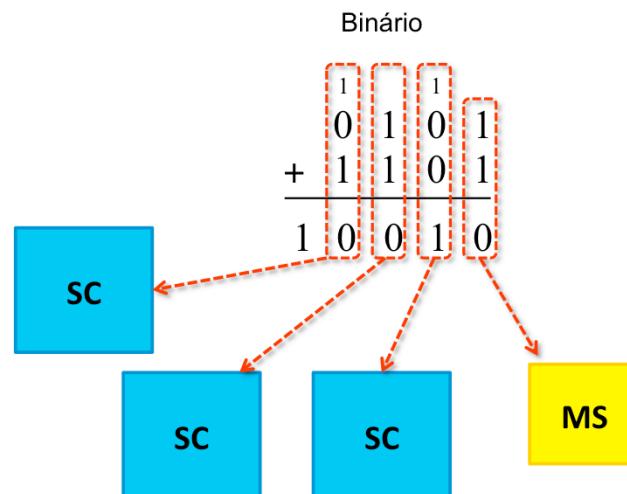
Decimal	Binário
$\begin{array}{r} \textcircled{1} \\ 1 \ 6 \ 3 \\ + 8 \ 9 \ 2 \\ \hline 1 \ 0 \ 5 \ 5 \end{array}$	$\begin{array}{r} \textcircled{1} \\ 0 \ 1 \ 0 \ 1 \\ + 1 \ 1 \ 0 \ 1 \\ \hline 1 \ 0 \ 0 \ 1 \ 0 \end{array}$
<div style="border: 1px solid blue; padding: 2px; width: fit-content;">carga, "vai um" ou carry</div>	

7

Sempre que somamos 1 e 1 em binário, temos 10. O 1 sobe como uma carga, como aconteceria caso somássemos dois números cuja soma resulte em um número ≥ 10 . O número da casa decimal "sobe" para o próximo número.

4.1. Soma de Números Binários

- Soma binária pode ser feita de forma similar à soma decimal:



8

As partes da soma binária são separadas em MS: Meio Somador e SC: Somador Completo. Nos próximos slides veremos como funcionam ambos.

4.1. Soma de Números Binários

- **Meio Somador (MS)**

- Realiza a soma de dois bits
- Não considera *carry de entrada*

The diagram illustrates a binary addition problem and a truth table for a Half Adder. The addition problem shows two 4-bit binary numbers being added: 0100 + 1101 = 1001. A blue circle highlights the carry-in from the most significant bit (MSB) position, and a red box highlights the least significant bit (LSB) result. Dotted arrows point from the truth table to the addition problem, indicating the correspondence between the inputs and the resulting sum and carry.

A	B	C_{n+1}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

9

O MS, Meio Somador, sempre realiza a soma de dois bits e não considera carry de entrada, ou seja, ele se comporta de forma similar à soma na casa da unidade quando comparada à soma de números decimais. Os valores de entrada estarão sempre entre 00 e 11, e a saída S resultará em 1 apenas quando uma das entradas for 1, e o Carry apenas acontecerá quando as entradas forem 11. No próximo slide, veremos como funciona o circuito do MS.

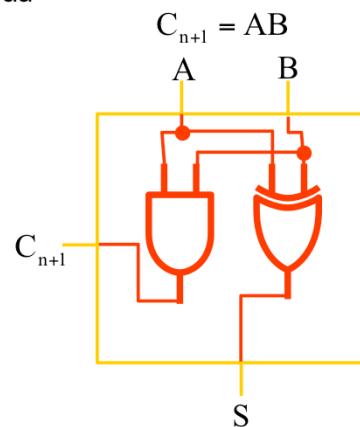
4.1. Soma de Números Binários

- **Meio Somador**

- Realiza a soma de dois bits
- Não considera *carry de entrada*

A	B	C _{n+1}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = \bar{A}B + A\bar{B} = A \oplus B$$



10

Como comentado no slide anterior, S será 1 apenas quando uma das entradas for 1, o que nos leva à utilização de uma porta XOR. Já a entrada C_{n+1} (carry), apenas será 1 quando ambas entradas forem 1, o que nos leva a uma porta AND. Este é o circuito do Meio Somador.

4.1. Soma de Números Binários

- **Somador Completo**

- Realiza a soma de dois bits, considerando o *carry* de entrada C_n

A	B	C_n	C_{n+1}	S
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

$$\begin{array}{r} & 1 & & 1 \\ & 0 & 1 & 0 & 1 \\ + & 1 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 0 \end{array}$$

11

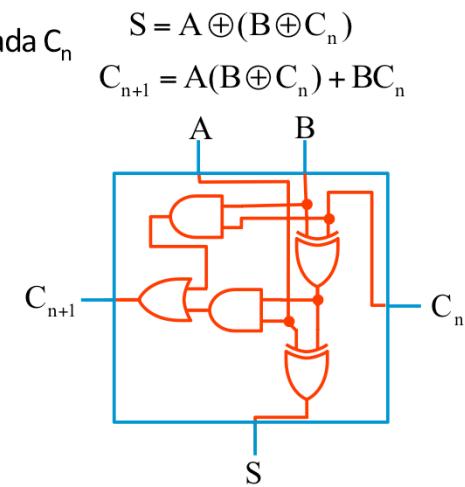
Já no Somador Completo, teremos também a soma de dois bits, porém teremos que considerar o carry de entrada C_n (que veio como resultado da soma anterior). Claro que não será em todos os casos que teremos carry=1, como pode ser visto na tabela.

4.1. Soma de Números Binários

- **Somador Completo**

- Realiza a soma de dois bits, considerando o *carry* de entrada C_n

A	B	C_n	C_{n+1}	S
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



12

Aqui está representado o circuito do SC. Da mesma forma que o MS, a saída S também é feita por portas XOR. O circuito do carry de saída é um pouco mais "complexo", mas o funcionamento dele é simples: será igual a 1 quando, e somente quando, dois ou mais valores de entrada forem iguais a 1.

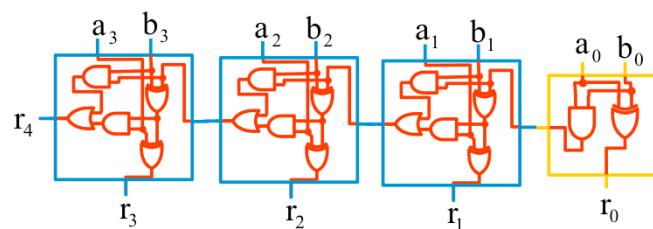
4.1. Soma de Números Binários

- **Somador Binário Paralelo**

- Soma de números de vários bits

- Exemplo: números de 4 bits

$$\begin{array}{r} a_3 \quad a_2 \quad a_1 \quad a_0 \\ + b_3 \quad b_2 \quad b_1 \quad b_0 \\ \hline r_4 \quad r_3 \quad r_2 \quad r_1 \quad r_0 \end{array}$$



13

O somador binário paralelo faz uso dos somadores explicados anteriormente: o Meio Somador e o Somador Completo.

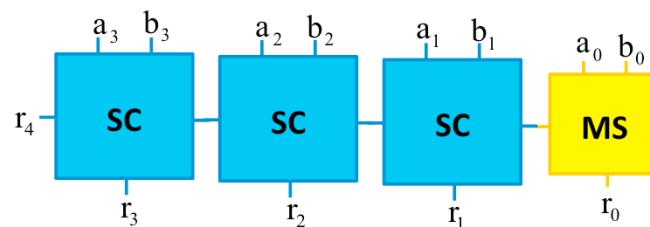
4.1. Soma de Números Binários

- **Somador Binário Paralelo**

- Soma de números de vários bits

- Exemplo: números de 4 bits

$$\begin{array}{r} a_3 \quad a_2 \quad a_1 \quad a_0 \\ + b_3 \quad b_2 \quad b_1 \quad b_0 \\ \hline r_4 \quad r_3 \quad r_2 \quad r_1 \quad r_0 \end{array}$$



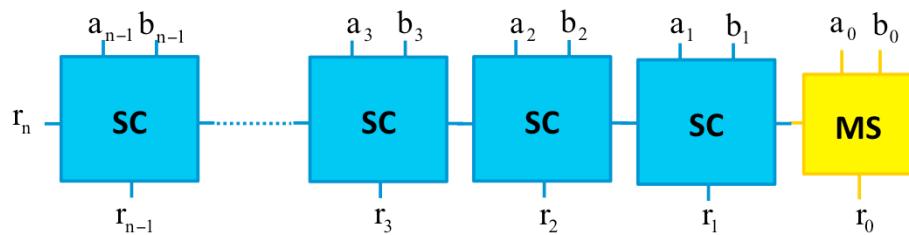
14

O carry in e out é conectado entre cada somador, ou seja, o carry out ($n+1$) do MS é conectado ao carry in (n) do SC e assim sucessivamente.

4.1. Soma de Números Binários

- Somador Binário Paralelo

- De maneira geral, somador de 2 números de n bits:



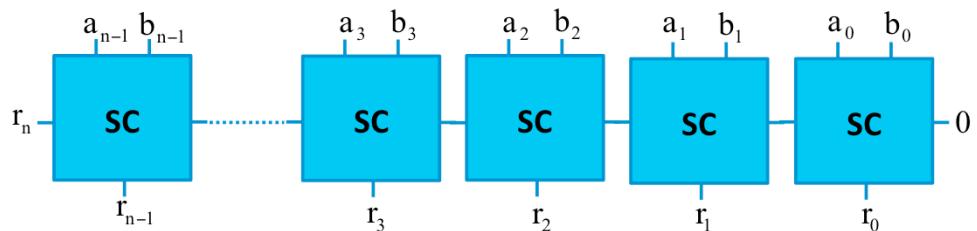
15

Uma possível representação de um somador de 2 números de n bits:

4.1. Soma de Números Binários

- Somador Binário Paralelo

- Outra forma:



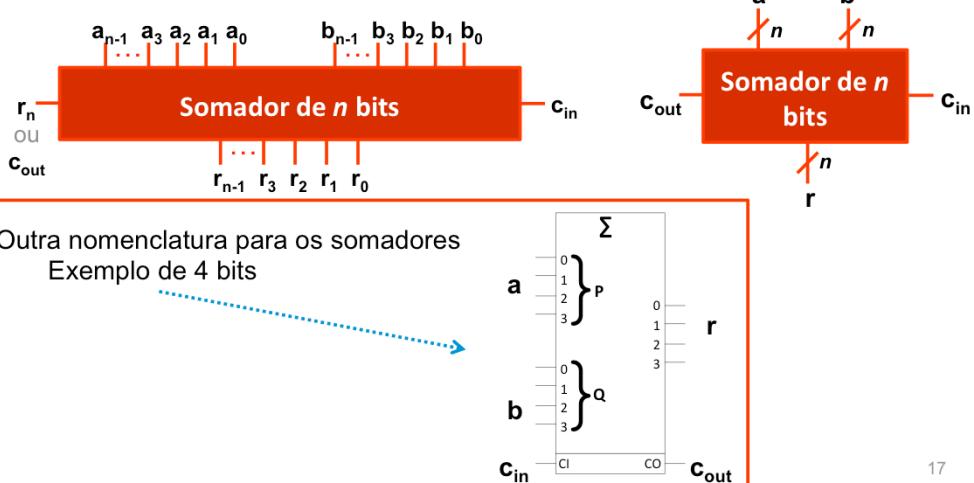
16

E aqui, temos outra forma. Observe que o MS foi substituído por um SC e agora há a possibilidade de adicionar-mos um carry in no primeiro "bloco" do somador.

4.1. Soma de Números Binários

- Somador Binário Paralelo

- Em um maior nível de abstração:



17

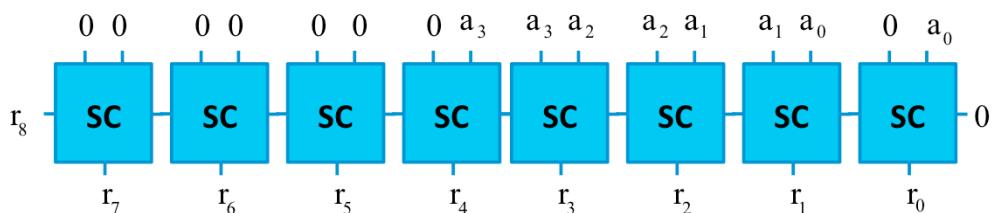
Em um maior nível de abstração temos estas representações.

Mais à direita temos o somador em RTL (Register Transfer Level), tema que será tratado com mais detalhes no futuro. Por ora, é importante saber que n é quantidade de bits do número binário que entra ou sai, que, nesta situação, será chamado de vetor. Note que, como Cin e Cout possuem apenas um bit, isso fica representado apenas com um "fio".

4.1. Soma de Números Binários

- Caso particular de Multiplicação por constante

- Se deslocamos para a esquerda o vector de entrada podemos multiplicar por potências de dois de forma simples usando unicamente somadores. Exemplo: $0110_2 \rightarrow 1100_2$ ($3_{10} \rightarrow 6_{10}$)
- **EXEMPLO 1:** Operação $R = 3 \times A = A + 2 \times A$ com $A = \{a_3 a_2 a_1 a_0\}$ de 4 bits.



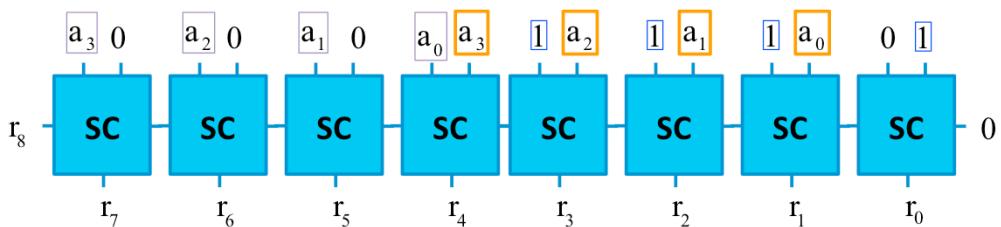
18

Aqui veremos uma forma diferente de utilização do somador binário. Ele pode ser utilizado para a multiplicação do número por uma constante. Caso desloquemos o vetor de entrada para a esquerda, podemos multiplicar este número por potências de dois de forma simples, utilizando apenas somadores. Por exemplo, peguemos a operação $3 \times A$. Esta operação pode ser separada em $A + 2 \times A$. Ou seja, teremos um vetor de A na mesma posição de entrada e o segundo vetor ($2 \times A$) deslocado uma posição para a esquerda.

4.1. Soma de Números Binários

- Caso particular de Multiplicação por constante

- EXEMPLO 2: Operação $R = 18 \times A + 15 = 16 \times A + 2 \times A + 15$ com A de 4 bits.



19

Aqui, temos um outro exemplo porém um pouco mais complexo. Queremos a operação $18 \times A + 15$. Essa operação pode ser dividida em $16 \times A + 2 \times A + 15$. Para $16 \times A$, basta deslocarmos o vetor A quatro "blocos" para a esquerda. Para $2 \times A$, deslocamos o vetor A um "bloco" para a esquerda. E para o número 15, que equivale a 1111 em binário, basta posicionar os valores nos blocos mais à direita.

PROBLEMAS

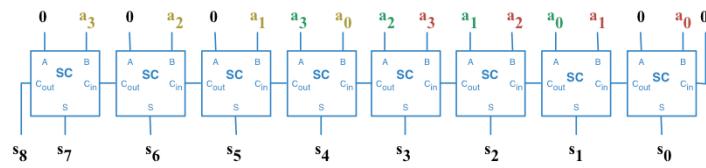
Problema 4.1. Pretende-se implementar unidades aritméticas com uma única entrada de 4 bits $A(3:0)$ sem sinal, que realize o cálculo das operações aritméticas:

- a) $f_1(7:0) = 19 \times A(3:0);$
- b) $f_2(7:0) = 19 \times A(3:0) + 33;$
- c) $f_3(7:0) = 9 \times A(3:0) + 8;$

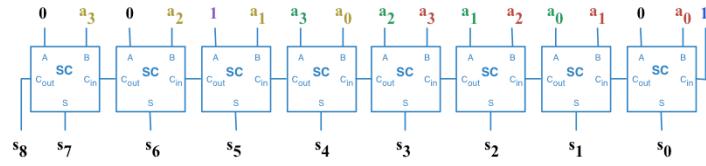
Desenhe o diagrama lógico dos circuitos utilizando **um** circuito somador de 8 bits com entrada e saída de carga (*carry in* e *carry out*) e o mínimo de lógica discreta possível.

PROBLEMAS

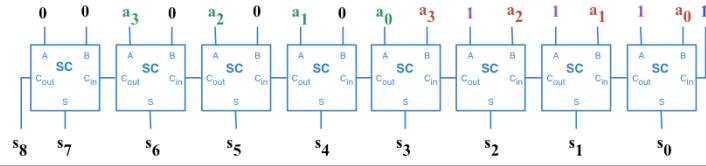
Solução Problema 4.1: a) $19A = 16A + 2A + A$



b) $19A + 33 = 16A + 2A + A + 32 + 1$



c) $9A + 8 = 8A + A + 7 + 1$



4.1. Soma de Números Binários

4.2. Representando Números com Sinal

4.3. Subtração usando Complemento de 2

4.4. Somador/Subtrator

4.5. Overflow

22

A seguir, veremos como representar números com sinal.

4.2. Representando Números com Sinal

- Com **3 bits**, podemos representar números sem sinal de **0** a **7**.
- Como você representaria então os números de **-7** a **7**? → Números complementares

23

Com 3 bits, podemos representar números sem sinal de 0 a 7. Mas como representaríamos os números de –7 a 7? Com números complementares.

4.2. Representando Números com Sinal

- Números Complementares
 - Representações simples
 - Facilitam operações de **soma** e **subtração**
 - A representação complementar mais usada é Complemento de 2.
 - Quando usados para representar números com sinal:
 - **Número positivo é o binário puro com MSB = 0**
 - **Em Complemento de 2 Número negativo é o complemento do binário puro mais um.**

24

Os números complementares possuem representações simples e facilitam as operações de soma e subtração. A representação complementar mais usada é o complemento de 2. Quando usado para representar números com sinal: o número positivo é o binário puro com bit mais significativo igual a 0. Em complemento de 2 o número negativo é o complemento do binário mais 1.

4.2. Representando Números com Sinal

- **Complemento de 2**

- Exemplo: Representação em Complemento de 2 com 4 bits

<u>0000</u>	0
<u>0001</u>	+1
<u>0010</u>	+2
<u>0011</u>	+3
<u>0100</u>	+4
<u>0101</u>	+5
<u>0110</u>	+6
<u>0111</u>	+7

<u>1111</u>	-1
<u>1110</u>	-2
<u>1101</u>	-3
<u>1100</u>	-4
<u>1011</u>	-5
<u>1010</u>	-6
<u>1001</u>	-7
<u>1000</u>	-8

$$A_{C2} = \overline{A} + 1$$

25

Aqui vemos uma tabela com exemplos da representação em complemento de 2 com 4 bits. O complemento de 2 funciona da seguinte forma: suponhamos que temos o número 0001 (+1 em binário). O seu complemento de 2, ou seja, -1, é o complemento de 0001 -> 1110, mais 1 bit, resultando em 1111. 1111 é o complemento de 2 de 0001.

4.2. Representando Números com Sinal

- **Complemento de 2**

- Com **8 bits**, qual faixa de valores de números inteiros com sinal pode ser representada usando a **Representação em Complemento de 2?**

4.2. Representando Números com Sinal

- **Complemento de 2**

- Com **8 bits**, qual faixa de valores de números inteiros com sinal pode ser representada usando a **Representação em Complemento de 2?**

De -128 a +127.

- Em geral, com **n bits** a faixa de valores de números inteiros com sinal pode ser representada usando a representação em complemento de 2.

De $-2^{(n-1)} \text{ a } +2^{(n-1)}-1$.

27

Podemos representar 256 números com 8 bits (2^8). Como queremos representar números inteiros utilizando a representação em complemento de 2, podemos representar números de -128 a +127. Isso porque sempre haverá um dos números com o MSB = 0 representando o 0 decimal.

4.2. Representando Números com Sinal

- **Complemento de 2**

- Vantagem comparado com outros métodos: operações de **soma** tornam-se mais simples
- Complemento de 2: complemento com relação a 2^N
- Portanto, todo número, quando somado ao seu complemento de 2, resulta em 2^N
 - Exemplo (com números de $N = 5$ bits):

$$\begin{array}{r} 0 \ 1 \ 0 \ 0 \ 1 \\ + 1 \ 0 \ 1 \ 1 \ 1 \\ \hline 1 \ 0 \ 0 \ 0 \ 0 \end{array} \quad \begin{array}{r} 1 \ 0 \ 0 \ 0 \ 1 \\ + 0 \ 1 \ 1 \ 1 \ 1 \\ \hline 1 \ 0 \ 0 \ 0 \ 0 \end{array}$$

28

O complemento de 2 possui uma vantagem importante em relação aos outros complementos: as operações de soma se tornam mais simples. O complemento de 2 é feito com relação a 2^N , em que N é o número de bits dos números binários em complemento de 2. Caso somemos dois números de N bits em que um é o complemento do outro, o resultado será 2^N .

Veja os exemplos, somando dois números de 5 bits, sendo que um é o complemento do outro, obtemos como resultado 100000, ou $32 = 2^5$.

4.2. Representando Números com Sinal

- **Complemento de 2**

- Se o bit de índice **N** for ignorado, a soma de um número com seu complemento de 2 resulta em **zero**

- Exemplo (com números de **N= 5 bits**):

$$\begin{array}{r} 0 \ 1 \ 0 \ 0 \ 1 \\ + 1 \ 0 \ 1 \ 1 \ 1 \\ \hline \cancel{1} \ 0 \ 0 \ 0 \ 0 \end{array} \quad \begin{array}{r} 1 \ 0 \ 0 \ 0 \ 1 \\ + 0 \ 1 \ 1 \ 1 \ 1 \\ \hline \cancel{1} \ 0 \ 0 \ 0 \ 0 \end{array}$$

29

Após a soma, caso o bit de índice N seja ignorado, a soma de um número com seu complemento de 2 sempre resultará em 0.

4.2. Representando Números com Sinal

- **Complemento de 2**

- Se o bit de índice **N** for ignorado, a soma de um número com seu complemento de 2 resulta em **zero**

- Exemplo (com números de **N= 5 bits**):

$$\begin{array}{r} 0 \ 1 \ 0 \ 0 \ 1 \\ + 1 \ 0 \ 1 \ 1 \ 1 \\ \hline \cancel{1} \ 0 \ 0 \ 0 \ 0 \end{array} \quad \begin{array}{r} +9 \\ -9 \\ \hline 0 \end{array}$$
$$\begin{array}{r} 1 \ 0 \ 0 \ 0 \ 1 \\ + 0 \ 1 \ 1 \ 1 \ 1 \\ \hline \cancel{1} \ 0 \ 0 \ 0 \ 0 \end{array} \quad \begin{array}{r} -15 \\ +15 \\ \hline 0 \end{array}$$

- Similaridade com números decimais com sinal:

- Somar **um número decimal** com **o seu negativo** resulta em **zero**
- Somar **um número binário** (desprezando o bit de índice **N**) com **o seu complemento de 2** também resulta em **zero**

30

E podemos destacar a similaridade com a soma de números decimais com sinal. Caso somemos um número decimal com o seu negativo, obtemos também o valor 0.

4.2. Representando Números com Sinal

- **Complemento de 2**
 - Assim, a **soma** de números binários com sinal representados em Complemento de 2 pode ser feita **sem se preocupar com o sinal dos números envolvidos**
 - **Consequência prática:** Representação em Complemento de 2 é a mais utilizada em sistemas digitais/computacionais

31

Assim, concluímos o exposto neste slide.

PROBLEMAS

Problema 4.2. faça a conversão dos seguintes números decimais com sinal para as representações em sinal em grandeza, complemento de 1 e complemento de 2 com 10 bits.

- a) -23_{10}
- b) 23_{10}
- c) 64_{10}
- d) -64_{10}
- e) -500_{10}
- f) 128_{10}

PROBLEMAS

Solução Problema 4.2:

- a) Para obter o valor negativo em Complemento 2 (C2) de $-23_{(10)}$ obtenho o valor positivo,

$$23_{(10)} = 0000010111_{(2)} \text{ complemento e sumo 1:}$$
$$\begin{array}{r} 1111101000 \\ + 1 \\ \hline 1111101001_{(C2)} = -23_{(10)} \end{array}$$

- b) A representação em binário corresponde ao C2 já que é um valor positivo:

$$23_{(10)} = 0000010111_{(2)} = 0000010111_{(C2)}$$

- c) A representação em binário corresponde ao C2 já que é um valor positivo:

$$64_{(10)} = 0001000000_{(2)} = 0001000000_{(C2)}$$

- d) Para obter o valor negativo em Complemento 2 (C2) de $-64_{(10)}$ obtenho o valor positivo,

$$64_{(10)} = 0001000000_{(2)} \text{ complemento e sumo 1:}$$
$$\begin{array}{r} 1110111111 \\ + 1 \\ \hline 1111100000_{(C2)} = -64_{(10)} \end{array}$$

- e) Para obter o valor negativo em Complemento 2 (C2) de $-500_{(10)}$ obtenho o valor positivo,

$$500_{(10)} = 0111110100_{(2)} \text{ complemento e sumo 1:}$$
$$\begin{array}{r} 1000001011 \\ + 1 \\ \hline 1000001100_{(C2)} = -500_{(10)} \end{array}$$

- f) A representação em binário corresponde ao C2 já que é um valor positivo:

$$128_{(10)} = 0010000000_{(2)} = 0010000000_{(C2)}$$

33

PROBLEMAS

Problema 4.3. Faça a conversão dos seguintes números decimais com sinal para representações em complemento de 2 com 8 bits e 16 bits. A partir dos resultados obtidos, observe que a representação de um número com um maior número de bits pode ser obtida fazendo a extensão do sinal do mesmo número representado com um menor número de bits.

a) -53_{10}

b) 53_{10}

PROBLEMAS

Solução Problema 4.3:

- a) Para obter o valor negativo em Complemento 2 (C2) de $-53_{(10)}$ obtenho o valor positivo, com 8 bits $53_{(10)} = 00110101_{(2)}$ complemento e sumo 1:

$$\begin{array}{r} 11001010 \\ + \quad 1 \\ \hline 11001011_{(C2)} = -53_{(10)} \end{array}$$

Para obter a expressão com 16 bits faço uma extensão do bit de sinal

The diagram shows the conversion of a 8-bit binary number $1001011_{(C2)} = -53_{(10)}$ to a 16-bit representation. A red arrow labeled "Bit de sinal" points to the leftmost bit of the 8-bit number. An arrow points from this 8-bit number to a 16-bit number where the leftmost bit is also circled in red. Ellipses between the two numbers indicate the continuation of the 8-bit pattern.

$$1001011_{(C2)} = -53_{(10)} \rightarrow 11111110|001011_{(C2)} = -53_{(10)}$$

- b) A representação em binário corresponde ao C2 já que é um valor positivo:

com 8 bits $53_{(10)} = 00110101_{(2)} = 00110101_{(C2)}$

Para obter a expressão com 16 bits faço uma extensão do bit de sinal

The diagram shows the conversion of a 8-bit binary number $00110101_{(C2)} = 53_{(10)}$ to a 16-bit representation. A red arrow labeled "Bit de sinal" points to the leftmost bit of the 8-bit number. An arrow points from this 8-bit number to a 16-bit number where the leftmost bit is also circled in red. Ellipses between the two numbers indicate the continuation of the 8-bit pattern.

$$00110101_{(C2)} = 53_{(10)} \rightarrow 00000000|00110101_{(C2)} = 53_{(10)}$$

35

- 4.1. Soma de Números Binários
- 4.2. Representando Números com Sinal

4.3. Subtração usando Complemento de 2

- 4.4. Somador/Subtrator
- 4.5. Overflow

36

A seguir, veremos a subtração usando o complemento de 2.

4.3. Subtração usando Complemento de 2

- Princípio para fazer **subtração**: usar representação em **Complemento de 2** e fazer

The diagram illustrates the decomposition of subtraction into addition. A yellow box at the top contains the equation $X - Y = X + (-Y)$. Two red dashed arrows point from the terms $-Y$ and $(-Y)$ to the words "adição" (addition) and "número negativo" (negative number) respectively. A large blue arrow points downwards from the yellow box to the resulting addition expression at the bottom.

$$X - Y = X + (-Y)$$

adição número negativo

$$X - Y = X + (\text{complemento de 2 de } Y)$$

37

O procedimento para a subtração utilizando complemento de 2 é simples: tendo dois números binários X e Y, basta fazermos a operação $X + (-Y)$. Traduzindo, basta pegarmos o complemento de 2 de Y e somarmos a X.

4.3. Subtração usando Complemento de 2

- Para determinar o **Complemento de 2**:
 - 1) Obter o **Complemento de 1** do número positivo
(inversão de todos os bits)
 - 2) Somar 1

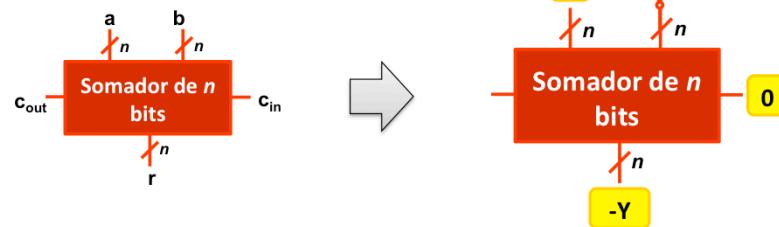
38

Como comentamos antes, para determinar o complemento de 2, basta obter o complemento de 1 do número (inverter todos os bits) e então somar 1. Exemplo: complemento de 2 do número 0001 => 1110 + 1 => 1111.

4.3. Subtração usando Complemento de 2

- Para determinar o **Complemento de 2**:
 - 1) Obter o **Complemento de 1** do número positivo (inversão de todos os bits)
 - 2) Somar 1

- Usando somador de n bits:

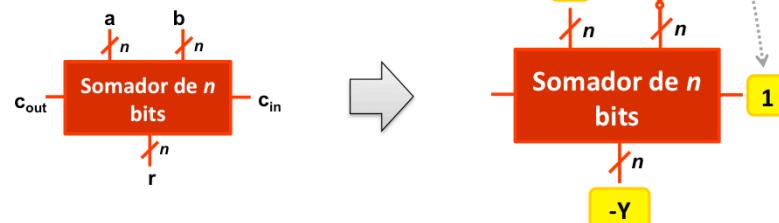


39

Utilizando o somador de n bits:

4.3. Subtração usando Complemento de 2

- Para determinar o **Complemento de 2**:
 - 1) Obter o **Complemento de 1** do número positivo (inversão de todos os bits)
 - 2) Somar 1
- Usando somador de n bits:

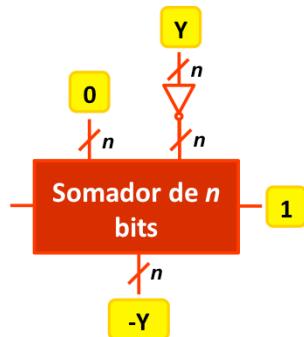


40

Utilizando o somador, basta invertermos todos os bits de Y e adicionar-mos 1 no carry in.

4.3. Subtração usando Complemento de 2

- Mas, como fazer $X - Y = X + (-Y)$ a partir da estrutura abaixo?

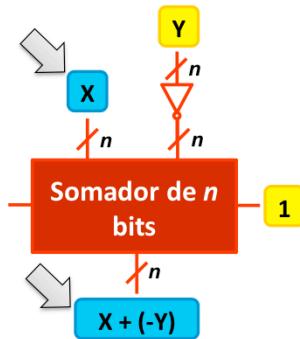


41

Como podemos fazer essa operação a partir da estrutura abaixo?

4.3. Subtração usando Complemento de 2

- Da seguinte forma:

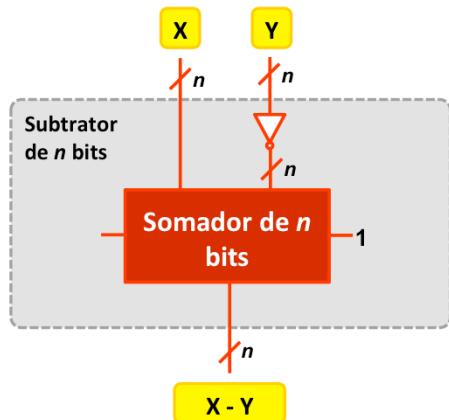


42

Basta fazermos isto: Soma-se X e Y' com 1 no bit de Carry in.

4.3. Subtração usando Complemento de 2

- Circuito para subtração:



43

Este é o circuito para subtração.

- 4.1. Soma de Números Binários
- 4.2. Representando Números com Sinal
- 4.3. Subtração usando Complemento de 2

4.4. Somador/Subtrator

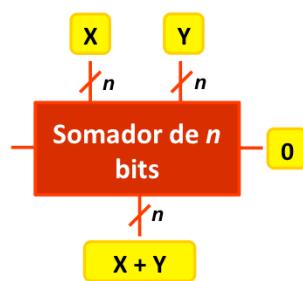
- 4.5. Overflow

44

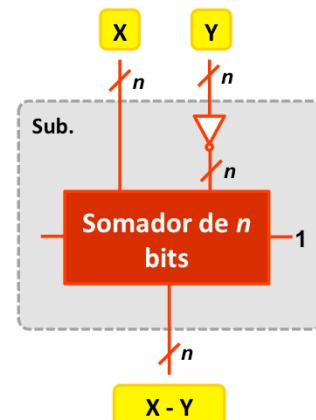
A seguir, veremos como obter o circuito de um somador/subtrator

4.4. Somador/Subtrator

- Somador:



- Subtrator:

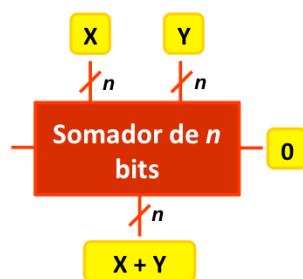


45

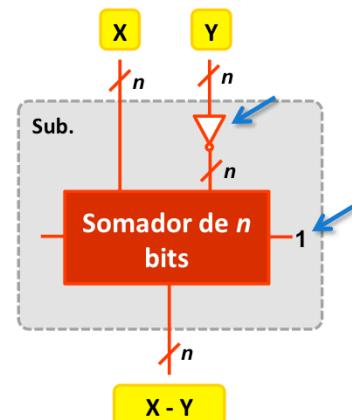
Aqui temos dois exemplos, um de um somador de n bits e um subtrator de n bits.

4.4. Somador/Subtrator

- Somador:



- Subtrator:



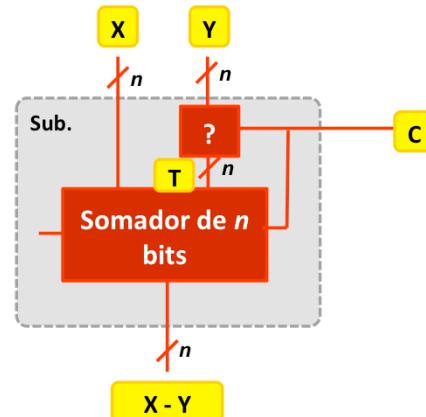
46

Note que a única diferença entre os dois é que, na subtração, a entrada Y é negada e há 1 no Carry in, pois utilizamos o complemento de 2 do vetor Y.

4.4. Somador/Subtrator

- Para obter um somador/subtrator, precisamos de um inversor que possa ser ativado e desativado

	C	Y	T
Soma	0	0	0
	0	1	1
Subtração	1	0	1
	1	1	0



47

Caso desejemos obter um somador/subtrator, precisamos de um inversor que possa ser ativado e desativado, possibilitando a modificação do sinal da operação, ou seja, negando ou não o vetor Y .

4.4. Somador/Subtrator

- Para obter um somador/subtrator, precisamos de um inversor que possa ser ativado e desativado

	C	Y	T
Soma	0	0	0
	0	1	1
Subtração	1	0	1
	1	1	0

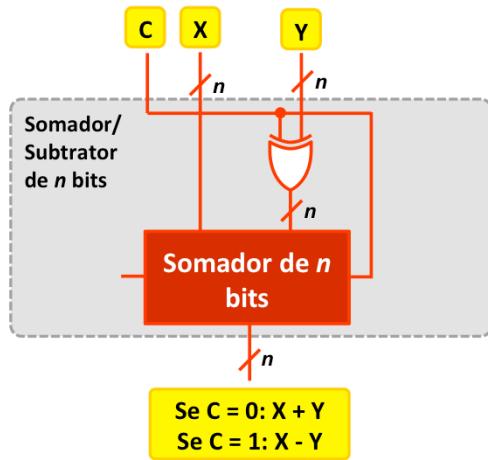
$$T = \bar{C}Y + C\bar{Y} = C \oplus Y$$

XOR funciona como um inversor controlado!

48

Precisamos de uma porta lógica que seja este inversor controlado, para isso adicionamos um sinal de controle C. Observando a tabela verdade, podemos ver que a única porta que se comporta da forma que desejamos é a porta XOR.

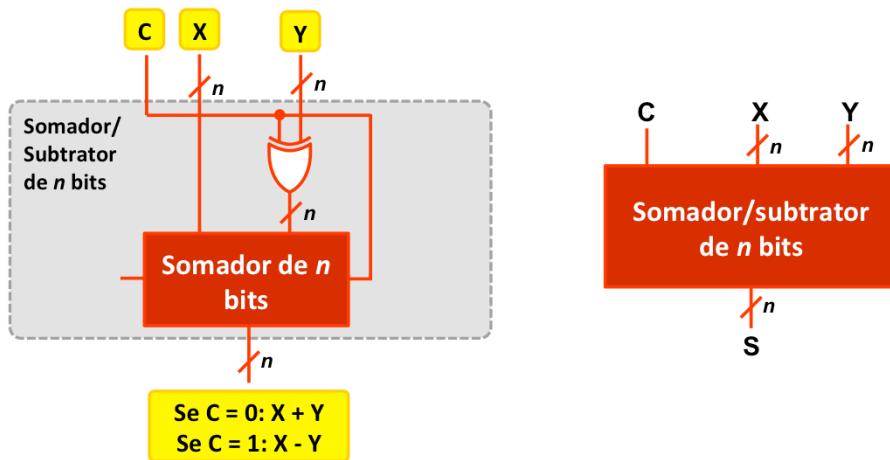
4.4. Somador/Subtrator



49

Logo, temos que nosso inversor controlado é adicionado ao circuito desta maneira, possibilitando que nosso somador faça ambas operações: soma e subtração. Observe que C é conectado APENAS ao vetor Y e ao Carry in, invertendo Y e somando 1 quando C estiver ativo, ou seja, quando a subtração estiver selecionada.

4.4. Somador/Subtrator



50

A representação de tudo isso em RTL é a seguinte.

- 4.1. Soma de Números Binários
- 4.2. Representando Números com Sinal
- 4.3. Subtração usando Complemento de 2
- 4.4. Somador/Substrator

4.5. Overflow

51

A seguir, veremos o que é o overflow e como ele funciona.

4.5. Overflow

- Ocorre quando o **resultado** de uma operação é **maior** (ou **menor**) do que o valor **máximo** (ou **mínimo**) que pode ser representado com um **determinado número de bits**

52

O overflow ocorre quando o resultado de uma operação é maior (ou menor) do que o valor máximo (ou mínimo) que pode ser representado com um determinado número de bits.

4.5. Overflow

- Ocorre quando o **resultado** de uma operação é **maior** (ou **menor**) do que o valor **máximo** (ou **mínimo**) que pode ser representado com um **determinado número de bits**
 - Exemplo: **5 bits** → **2^5** valores diferentes
 - Em binário puro: **0 até 31**
 - Em complemento de 2: **-16 até 15**

53

Por exemplo, com um número binário de 5 bits podemos representar 32 valores. De 0 a 31 em binário puro e de -16 a 15 em complemento de 2.

4.5. Overflow

- Ocorre quando o **resultado** de uma operação é **maior** (ou **menor**) do que o valor **máximo** (ou **mínimo**) que pode ser representado com um **determinado número de bits**
 - Exemplo: **5 bits** → **2⁵** valores diferentes
 - Em binário puro: **0 até 31**
 - 5 + 9 = 14** (ok)
 - 30 + 5 = 35 (overflow!)**
 - Em complemento de 2: **-16 até 15**
 - 5 + 9 = 14** (ok)
 - 15 + 5 = 20 (overflow!)**
 - 5 - 15 = -10** (ok)
 - 9 - 9 = -18 (overflow!)**
 - Em geral, o **overflow** precisa ser detectado e/ou tratado

54

Peguemos os valores em binário puro, de 0 a 31. Caso façamos uma soma de dois números dentro desta faixa, há a chance de ocorrer o overflow, como mostra o exemplo. Caso somemos 30+5, temos overflow, ou seja, não conseguimos representar o resultado dentro da faixa de bits em que estamos trabalhando. A mesma coisa acontece para os valores em complemento de 2. Caso façamos uma operação que é maior ou menor do que o possível de representar dentro da faixa de bits, temos o overflow. E este deve ser detectado e/ou tratado.

4.5. Overflow

- **Overflow** só ocorre se os dois números de uma soma são ambos positivos ou ambos negativos
 - Exemplos (**Complemento de 2 com 5 bits**) valores de saída -16 a +15:

$$\begin{array}{r} 10 \\ + 8 \\ \hline 18 \end{array} \quad \begin{array}{r} 0\overset{1}{1}010 \\ + 01000 \\ \hline 10010 \end{array} \text{ Errado!}$$

$$\begin{array}{r} -9 \\ + -11 \\ \hline -20 \end{array} \quad \begin{array}{r} 10111 \\ + 10101 \\ \hline 101100 \end{array} \text{ Errado!}$$

55

Podemos observar que o overflow ocorre apenas quando ambos os números de uma soma são positivos ou negativos.

4.5. Overflow

- **Overflow** só ocorre se os dois números de uma soma são ambos positivos ou ambos negativos
 - Exemplos (**Complemento de 2 com 5 bits**) valores de saída -16 a +15:

$$\begin{array}{r} 10 \\ + 8 \\ \hline 18 \end{array} \quad \begin{array}{r} 0\ 1\ 0\ 1\ 0 \\ + 0\ 1\ 0\ 0\ 0 \\ \hline 1\ 0\ 0\ 1\ 0 \end{array}$$

-14? Errado!

$$\begin{array}{r} -9 \\ + -11 \\ \hline -20 \end{array} \quad \begin{array}{r} 1\ 0\ 1\ 1\ 1 \\ + 1\ 0\ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 1\ 0\ 0 \end{array}$$

+12? Errado!

Atenção: em soma de números em C2,
esse bit deve ser desprezado.

56

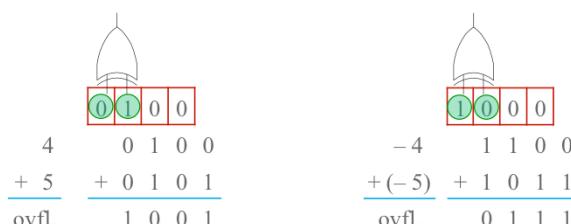
Lembre-se de que em caso de soma de números em Complemento de 2, o bit de índice N deve ser ignorado. Neste caso, o bit de índice 5.

4.5. Overflow

- A condição de *overflow* pode ser detectada por inspecção dos dois bits de *carry* mais significativos.

Exemplo:

$$\text{Overflow} = \text{CarryOut}_{N-1} \oplus \text{CarryOut}_{N-2}$$



57

Podemos detectar a condição de overflow com números em complemento de 2 através da análise dos bits de carry mais significativos. Ao utilizar uma porta XOR nos valores, caso tenhamos uma saída igual a 1, o overflow acontece.

PROBLEMAS

Problema 4.4. Para duas entradas (X, Y) em complemento de 2 com 5 bits. Indique se existe *overflow* quando são somadas os valores X e Y . Indique qual é o valor em decimal correspondente das entradas X e Y . Caso não exista *overflow*, indique qual é o valor em decimal correspondente da soma $S=X+Y$.

- a) $X=10001_{C2}, Y=01111_{C2}$.
- b) $X=11110_{C2}, Y=11111_{C2}$.
- c) $X=00101_{C2}, Y=01010_{C2}$.
- d) $X=00011_{C2}, Y=10000_{C2}$.

Problema 4.5. Projetar um somador de n bits com detector de *overflow*.

PROBLEMAS

Solução Problema 4.4:

a) Fazemos a soma colocando a informação dos *carry*s (em verde) para detectar o *overflow*:

$$\begin{array}{r}
 1 \oplus 1=0 \quad \leftarrow \\
 + \quad \boxed{1}111 \\
 + \quad 10001_{(C2)}=-15_{(10)} \\
 \hline
 01111_{(C2)}=15_{(10)} \\
 \hline
 00000_{(C2)}=0_{(10)}
 \end{array}$$

O valor 0 indica que não existe overflow e que o resultado da soma está correto.

b) Fazemos a soma colocando a informação dos *carry*s (em verde) para detectar o *overflow*:

$$\begin{array}{r}
 1 \oplus 1=0 \quad \leftarrow \\
 + \quad \boxed{1}110 \\
 + \quad 11110_{(C2)}=-2_{(10)} \\
 \hline
 11111_{(C2)}=-1_{(10)} \\
 \hline
 11101_{(C2)}=-3_{(10)}
 \end{array}$$

O valor 0 indica que não existe overflow e que o resultado da soma está correto.

c) Fazemos a soma colocando a informação dos *carry*s (em verde) para detectar o *overflow*:

$$\begin{array}{r}
 0 \oplus 0=0 \quad \leftarrow \\
 + \quad \boxed{0}0000 \\
 + \quad 00101_{(C2)}=-5_{(10)} \\
 \hline
 01010_{(C2)}=-10_{(10)} \\
 \hline
 01111_{(C2)}=15_{(10)}
 \end{array}$$

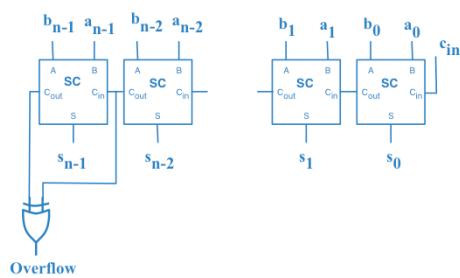
O valor 0 indica que não existe overflow e que o resultado da soma está correto.

d) Fazemos a soma colocando a informação dos *carry*s (em verde) para detectar o *overflow*:

$$\begin{array}{r}
 0 \oplus 0=0 \quad \leftarrow \\
 + \quad \boxed{0}0000 \\
 + \quad 00011_{(C2)}=-3_{(10)} \\
 \hline
 10000_{(C2)}=-16_{(10)} \\
 \hline
 10011_{(C2)}=-13_{(10)}
 \end{array}$$

PROBLEMAS

Solução Problema 4.5:



60



FEDERAL UNIVERSITY
OF SANTA CATARINA

EEL5105 – Circuitos e Técnicas Digitais

Aula 5

Prof. Héctor Pettenghi

hector@eel.ufsc.br

<http://hectorpettenghi.paginas.ufsc.br>

Exercícios

- Os exercícios da 11^a edição do livro do Tocci indicados abaixo são os recomendados:
 - A partir do **6.2** até o **6.5**;
 - A partir do **6.7** até o **6.10**;
 - A partir de **6.18** até o **6.20**;
 - Dê preferência aos exercícios marcados com * pois estes tem resposta no final do capítulo;
 - No caso dos exercícios sem resposta, você pode testar o resultado em algum simulador.
- **A versão digital da 11^a edição do livro do Tocci está disponível no site da BU**
 - Mais especificamente em:
http://150.162.4.10/pergamen/biblioteca_s/php/login_pearson.php

62