Sistemas Digitais

Aula 07 - Circuitos combinacionais

Apresentação

Em aulas anteriores, você estudou o funcionamento das portas lógicas básicas e usamos álgebra booleana para descrever e analisar os circuitos realizados a partir da combinação dessas portas estudadas.

Esses circuitos, obtidos da combinação das portas lógicas, são chamados circuitos combinacionais, ou seja, eles são constituídos da combinação das portas lógicas. A saída do circuito depende apenas de valores que são colocados nas entradas, ou seja, dos níveis lógicos das entradas.

Assim, um **circuito combinacional** não tem memória, suas saídas dependem apenas das entradas atuais.

Para exemplificar, podemos pensar no nosso controle remoto da televisão. Temos duas maneiras de mudar o canal, uma é utilizando o teclado numérico e colocando o valor do canal desejado. Por exemplo, quero assistir ao canal 12, aperto as teclas dos números 1 e 2 no controle.

A segunda maneira seria através da tecla canal, que passa para cima ou para baixo os canais. Dessa segunda maneira, podemos ver que para localizar o canal desejado se estamos, por exemplo, no canal 15, teremos que apertar três vezes para baixo para sintonizar o canal 12. Para isso, o circuito interno deve saber em qual canal está para mudar para um canal superior ou inferior, ou seja, é necessária uma memória nesse circuito. Isso não ocorre na primeira maneira, já que não se precisa de nenhuma informação prévia para se chegar ao canal desejado.

O circuito que tem memória é chamado circuito sequencial, mas não será alvo desta nossa aula. Estudaremos esse tipo de circuito em aulas futuras.

Vamos também estudar alguns tipos básicos de circuitos combinacionais, como o circuito somador, que, como o próprio nome sugere, faz a adição de números, ou seja, é um circuito fundamental em uma unidade lógica aritmética. Além do circuito somador, veremos outros circuitos combinacionais, como: multiplexador, demultiplexador, comparador e decodificador.

Objetivos

Ao final desta aula, você será capaz de:

- Escrever expressões lógicas através da soma de produtos ou do produto de somas.
- Compreender o funcionamento dos circuitos Exclusive-OR (XOR) e Exclusive-NOR (XNOR).
- Entender e descrever o funcionamento de alguns circuitos combinacionais mais utilizados: somador, multiplexador, demultiplexador, comparador e decodificador.

Soma-de-produtos e produto-de-somas

Começaremos nossa aula vendo, de uma maneira bem simples, as formas de se escrever uma expressão lógica através da soma de produtos ou do produto de somas.

Observe os exemplos a seguir de expressões da soma de produtos e você perceberá que é exatamente a soma (+) de multiplicações (produto):

a. $ABC + \overline{A}B\overline{C}$

b. $AB + C\overline{D} + D + AB\overline{C}$

c. $A + BC$

Observe agora os exemplos de expressões do produto de soma e você perceberá que é exatamente a multiplicação (.) de várias somas:

a. $(A + B + C).(\overline{A} + \overline{B} + \overline{C})$

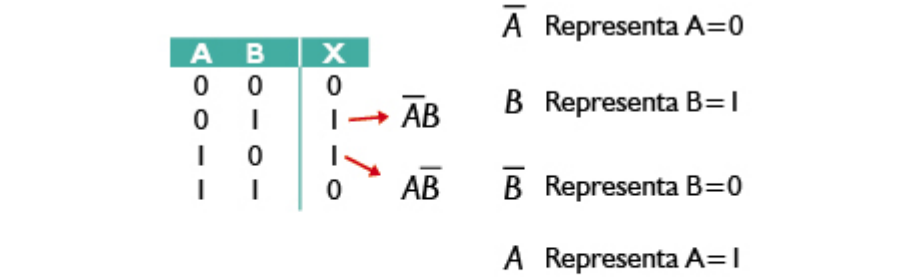
b. $(A + B).(C + \overline{D}).D$

c. $(A + C).(B + C)$

Revisando Soma-de-Produtos

Para tirarmos a expressão de uma tabela verdade, vamos trabalhar sobre a lógica verdadeira, que chamamos de “1”, ou seja, vamos analisar quando a saída for “1” (ou estiver, como podemos dizer, em nível lógico alto). Observe a tabela da **Figura 1**.

Figura 01 - Tabela verdade da função $\overline{A}.B + A.\overline{B}$



Como x=1, nestas duas situações, podemos dizer que a saída será igual a '1', quando quaquer uma das duas situações situadas ocorrer, $\overline{A}B$ ou qaundo $A\overline{B}$

Fonte: Tocci, Neal e Gregory (2007).

Como $X = 1$, em duas situações na tabela verdade da Figura 1, podemos dizer que a saída será igual a "1" quando qualquer uma dessas duas situações ocorrer: $\overline{A}.B$ ou $A.\overline{B}$.

A saída "1" é indicada por duas possibilidades, como já descrito. Sabemos que para gerar o "1", teremos que ter para a primeira situação A = 0 e B = 1, sendo representada por $(\overline{A}.B)$, pois esse "e" nada mais é que a nossa porta AND. A outra situação será representada por A = 1 e B = 0, ou seja, $A.\overline{B}$.

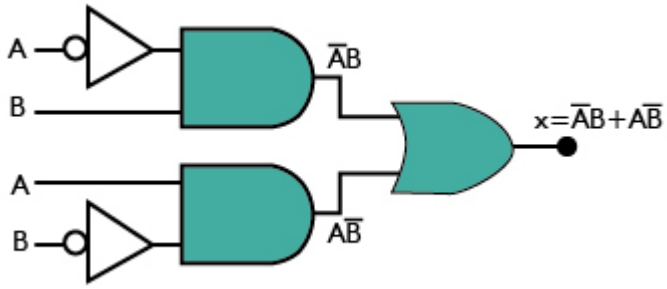
A saída de em "1" poderá acontecer em uma OU outra situação, assim podemos escrever a expressão como:

$$x = \overline{A}B + A\overline{B}$$

onde o nosso **OU** está representada pelo (+), que é a nossa porta lógica OR ("OU"), como vimos na aula de portas lógicas.

Dessa forma, olhando a tabela verdade, podemos escrever a expressão lógica e, por consequência, projetar o circuito, como mostrado na **Figura 2**.

Figura 02 - Circuito para a função $A\overline{B} + \overline{A}B$



Fonte: Tocci, Neal e Gregory (2007).

Muitas vezes, quando escrevemos a expressão de uma tabela verdade, ela é muito grande. Existem métodos de simplificação, mas que não serão alvo do nosso estudo, por enquanto.

Atividade 01

1. Quais das expressões a seguir estão na forma de soma de produtos?
    a. $\overline{A}B + CD + E$
    b. $(A + B)(C + D)$
    c. $\overline{\overline{MN}} + PQ$
    d. $(Z + Q).(A + L)$
2. Quais das expressões anteriores estão na forma de produto de somas?
3. Para resolver essa questão, observe primeiramente as duas tabelas a seguir.

A	B	X	
0	0	1	→ $\overline{A}\overline{B}$
0	1	0	
1	0	0	
1	1	1	→ AB

$\{ x = \overline{A}\overline{B} + AB \}$

A	B	C	D	X	
0	0	0	0	0	
0	0	0	1	1	→ $\bar{A}\bar{B}\bar{C}D$
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	→ $\bar{A}B\bar{C}D$
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	0	
1	1	0	1	1	→ $A\bar{B}CD$
1	1	1	0	0	
1	1	1	1	1	→ $ABCD$

$$\left\{ x = \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + A\bar{B}CD + ABCD \right\}$$

Seguindo os exemplos que foram apresentados nessas duas tabelas, agora tente definir a expressão da tabela a seguir:

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Circuitos Exclusive-OR (XOR) e Exclusive-NOR (XNOR)

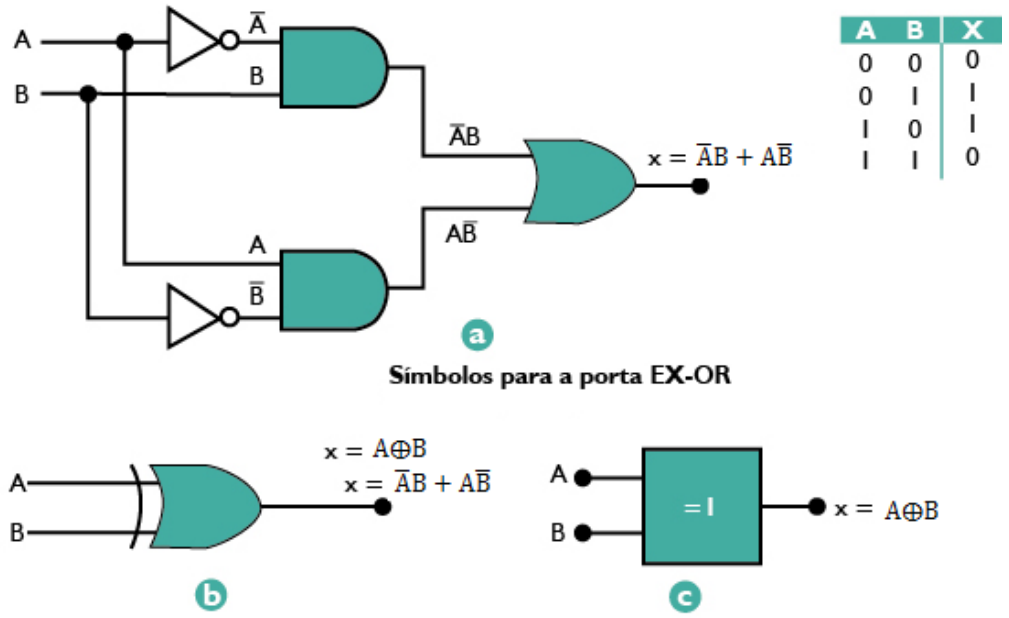
Agora que já sabemos extrair a expressão lógica de uma tabela verdade e já praticamos, vamos estudar alguns circuitos bastante utilizados.

Circuito Exclusive-OR (XOR)

O circuito Exclusive-OR, abreviado como **XOR**, produz uma saída em nível alto “1”, quando as entradas são diferentes, como podemos perceber na tabela verdade da Figura 3. Essa porta é muito utilizada, tem um símbolo próprio, apesar de poder

ser produzida pela combinação das portas AND, OR e inversores, como mostrado na Figura 3 (a). A expressão lógica está expressa na Figura 3 (b) com o símbolo da XOR, que também pode ser representada pelo símbolo IEEE/ANSI (Figura 3 (c)).

Figura 03 - (a) Circuito XOR e tabela verdade; (b) símbolo tradicional da porta XOR; (c) símbolo IEEE/ ANSI para a porta XOR



Fonte: Tocci, Neal e Gregory (2007).

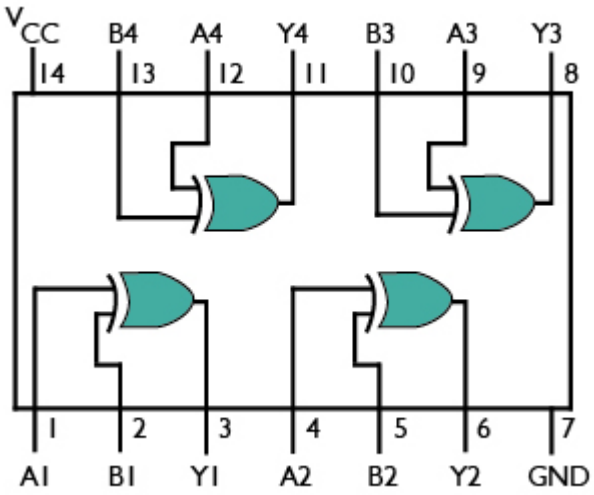
Essa porta apresentada possui apenas duas entradas, mas podemos ter versões com três ou quatro entradas. Uma forma abreviada de representar a expressão XOR é dada por:

$$x = \overline{A}B + A\overline{B} = A \oplus B$$

O símbolo $\oplus$ representa a operação da porta XOR.

O circuito integrado XOR mais conhecido é o 7486, que tem sua pinagem mostrada na **Figura 4**. Temos também outros circuitos como 74LS86, 74C86 e 74HC86.

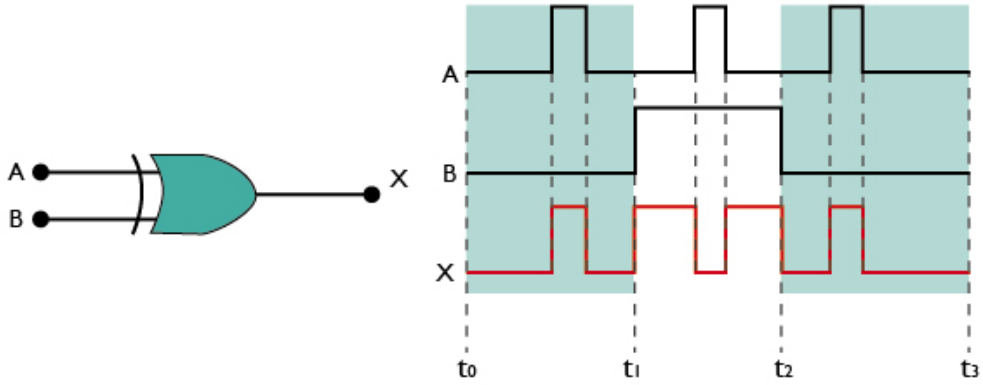
Figura 04 - Circuito integrado 7486 possui quatro portas XOR com duas entradas cada



Fonte: <http://www.clubedohardware.com.br/artigos/introducao-as-portas-logicas/1139/7> Acesso em: 14 ago. 2012.

O formato de onda característico da porta XOR está descrito na **Figura 5**. Nela, podemos perceber que sempre que as entradas se encontram em níveis lógicos diferentes, a saída se apresenta em "1".

Figura 05 - Formato de onda característico da porta XOR



Fonte: Tocci, Neal e Gregory (2007).

Atividade 02

Defina as expressões lógicas para as tabelas verdade a seguir:

a.

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

b.

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

Circuito Exclusive-NOR (XNOR)

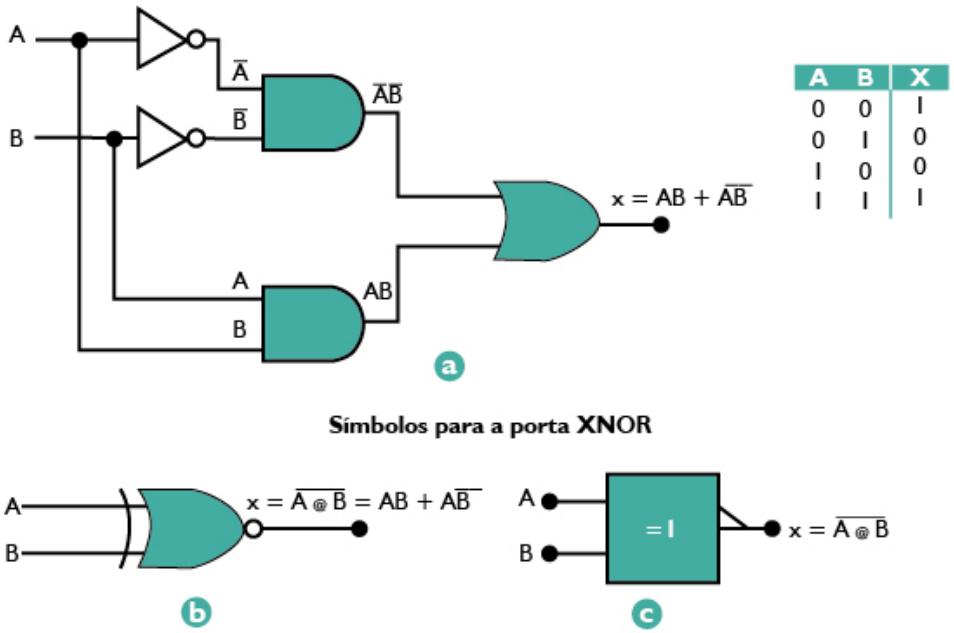
Observando as funções da **Atividade 2**, verificamos que na tabela verdade do item (a) a saída somente está em “1” quando as entradas são diferentes, o que representa um circuito XOR. Na tabela verdade do item (b), a saída somente está em “1” quando as entradas são iguais. Isso é o contrário do circuito XOR, o que representa um circuito XNOR.

O circuito Exclusive-NOR, abreviado como **XNOR**, produz uma saída em nível alto “1”, quando as entradas são iguais, de forma exatamente contrária ao XOR. A expressão é descrita da seguinte forma:

$$x = \overline{A}\,\overline{B} + AB = \overline{A \oplus B} = A \odot B$$

Você pode ver uma representação do circuito XNOR e da sua tabela verdade na Figura 6(a). O símbolo da porta XNOR (Figura 6(b)) é bem parecido com o da porta XOR, excetuando por apresentar na saída da porta o símbolo “o”, que representa a inversão do sinal. Já o símbolo IEEE/ANSI (Figura 6(c)) acrescenta um pequeno triângulo na saída da porta para representar a inversão do XOR para o XNOR

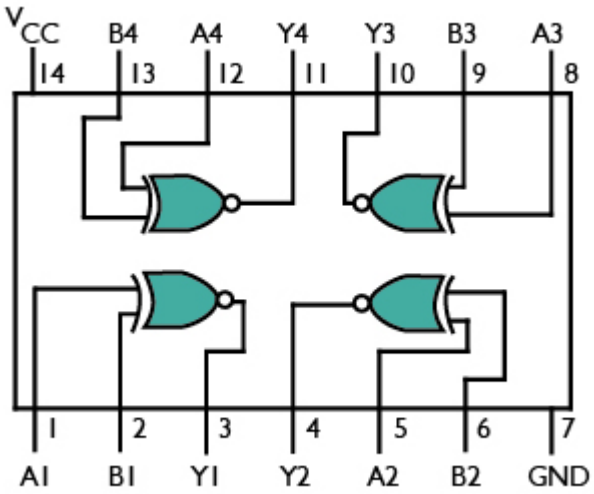
Figura 06 - (a) Circuito XNOR e tabela verdade; (b) símbolo tradicional para a porta XNOR; (c) símbolo IEEE/ ANSI para a porta XNOR



Fonte: Tocci, Neal e Gregory (2007).

Existem vários tipos de circuitos integrados (CIs) que apresentam portas lógicas XNOR (Figura 7). Podemos citar entre eles: 74LS266, 74C266 e 74HC266.

Figura 07 - Circuito integrado 7486 possui quatro portas XOR com duas entradas cada



Fonte: <http://www.clubedohardware.com.br/printpage/Introducao-as-Portas-Logicas/1139> Acesso em: 14 ago. 2012.

Agora, depois de ter conhecido (nesta aula e em anteriores) as portas NOT, AND, NAND, OR, NOR, XOR e XNOR, e aprendido como elas funcionam, ou seja, saber qual o valor da saída de cada porta para um determinado valor aplicado na entrada, você vai começar a ver que essas portas se juntam e são na verdade conectadas para formar circuitos que têm finalidades específicas.

Assim, vamos começar com o circuito somador. Como o próprio nome sugere, ele realiza soma de duas variáveis, de dois sinais ou de dois bits de entrada.

Caracterização de Circuitos Combinacionais Básicos

Até aqui, vimos o circuito XOR e o circuito XNOR. Agora, vamos ver alguns circuitos combinacionais básicos, começando pelo circuito somador.

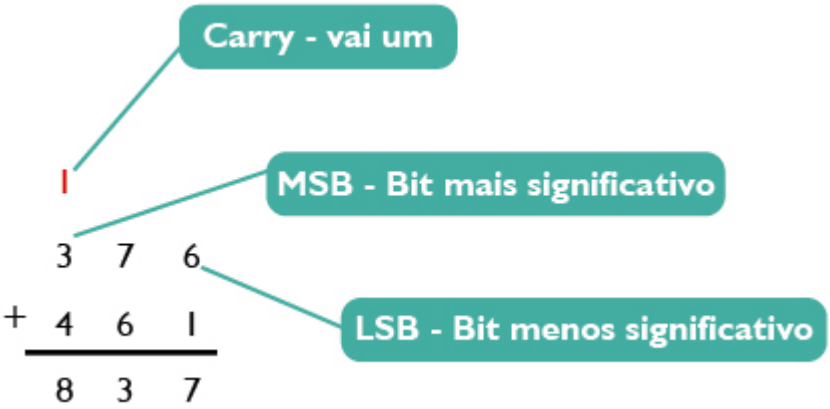
Circuito Somador

Primeiramente, antes de entendermos o funcionamento do circuito, temos que entender como a soma algébrica booleana é realizada.

Vamos nos deter a estudar a soma ou adição binária apenas. Existem outras, como a adição no sistema complemento de 2, hexadecimal e BCD, mas que não serão alvo deste curso. Em um curso de circuitos digitais mais detalhado, tais temas são abordados.

A adição de dois números binários é realizada da mesma forma que a adição de números decimais. Sendo que a adição binária é até mais simples porque se restringe a menos casas (apenas “0” e “1”). Primeiro, será interessante nós revermos o caso da adição decimal:

Figura 08 - Exemplo de uma soma decimal



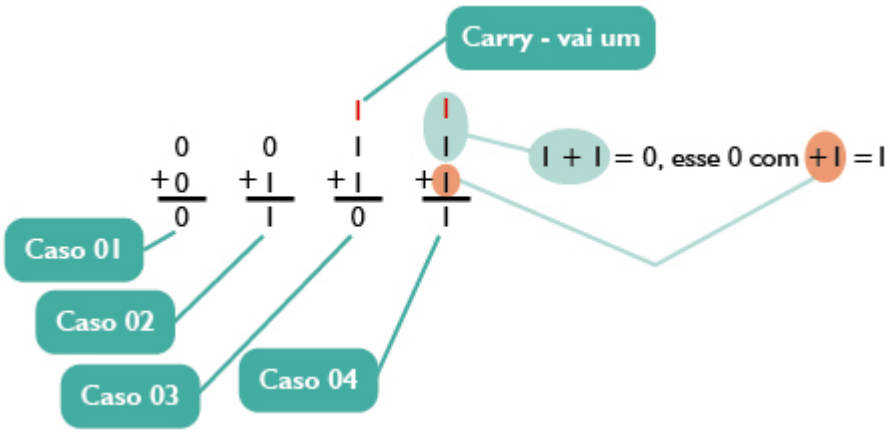
Fonte: Tocci, Neal e Gregory (2007).

Podemos notar, na **Figura 8**, que temos a soma de dois números (376 e 461), o algarismo mais significativo de 376 é o número 3 e o algarismo menos significativo é o número 6, já do número 461, o número 4 é o mais significativo e o número 1 é o menos significativo.

Você pode agora deduzir que o resultado (837) tem o número 8 como mais significativo e o número 7 como menos significativo. O número 1 é chamado carry, ou "vai um" como conhecido na soma de decimal, que é o número carregado resultante da soma anterior, pois 7 + 6 resulta em 13, sendo que fica o 3 abaixo dos números 7 e 6 e o 1 será carregado para a próxima soma (3 e 4).

A mesma ideia é seguida na adição binária, porém, nos limitamos apenas a 4 casos, listados na **Figura 9**.

Figura 09 - Soma binária

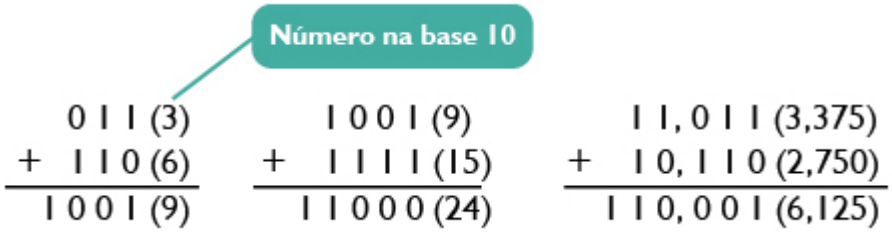


Fonte: Tocci, Neal e Gregory (2007).

O caso **01** é apenas a soma **0 + 0 = 0**, bem simples assim. No caso **02**, temos **0 + 1 = 1**. No caso **03**, precisamos um pouco mais de atenção, porque **1 + 1 = 0**, porém, temos um *carry* igual a “1”. O caso mais complicado é o caso **04**, em que temos **1 + 1 + 1 = 1**, vamos por parte, **1 + 1 = 0**, com *carry* "1" para a próxima casa. Esse resultado somando 1, do carry anterior, $(0 + 1)$ será então 1.

A seguir (Figura 10), vemos alguns exemplos de adição de números binários (entre parênteses, está colocado o número decimal correspondente). Quando mais de dois números são somados, os dois primeiros são somados e o resultado é somado a um terceiro número, e assim por diante.

Figura 10 - Exemplos de somas binárias



Fonte: Tocci, Neal e Gregory (2007).

Adição é a operação aritmética mais importante nos sistemas digitais, pois operações de subtração, multiplicação e divisão usam a operação adição nos computadores modernos de hoje.

Atividade 03

Observando o que foi mostrado nas Figuras 9 e 10, efetue a soma dos seguintes pares de números binários:

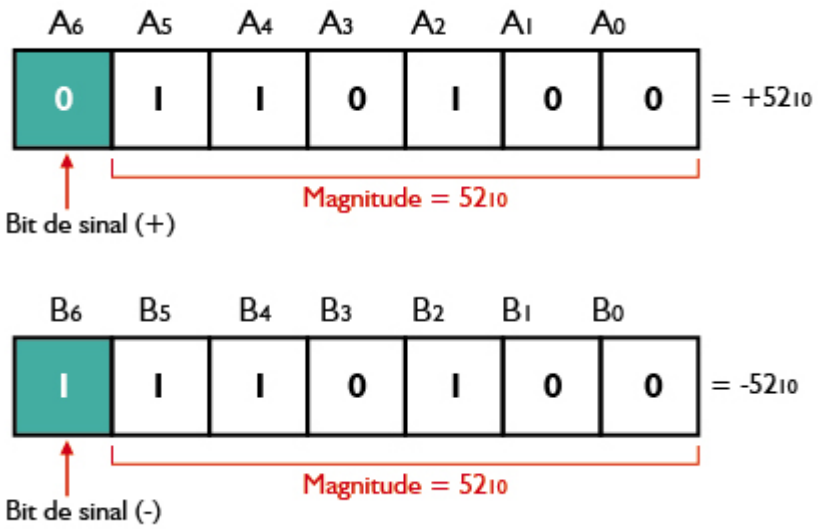
a. $10110 + 00111$

b. $011,101 + 010,010$

Você deve agora estar se perguntando como representar um número negativo. É bem fácil!

Podemos utilizar o que chamamos de bit de sinal, ou seja, é um bit que adicionamos do lado esquerdo do número. Esse número será “0”, se o número for positivo e “1” se o número for negativo. Observe, no exemplo a seguir (Figura 11), o número +5210 (número 52 na base 10) com um “0” no bit de sinal e o número -5210 com o número “1” no local do bit de sinal.

Figura 11 - Exemplo da representação de um número positivo e de um número negativo, através do bit de sinal



Fonte: Tocci, Neal e Gregory (2007).

Agora, podemos começar a pensar quais portas lógicas seriam adequadas para realizarmos a adição de dois bits. Sabemos que na lógica booleana "1 + 1" (1 **ou** 1) é diferente da soma algébrica.

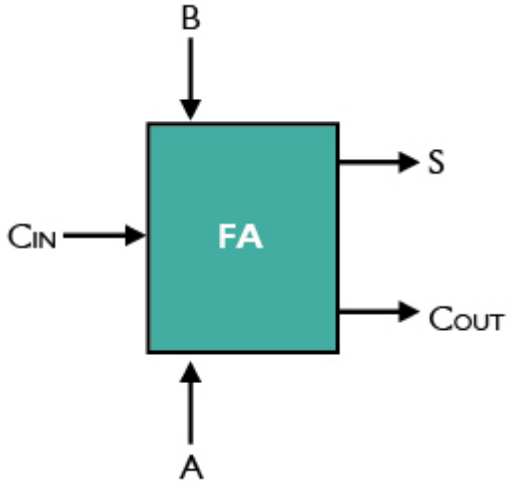
Porém, se voltarmos a pensar nos quatro casos da soma booleana, que estudamos a pouco, vemos que a função **OU EXCLUSIVO (XOR)** é igual à soma aritmética. Mas a semelhança ainda é incompleta.

Na operação de soma, precisamos considerar também um dígito de transporte ("vai um" ou *carry*) e a função mencionada não tem esse recurso. Considerando isso, podemos concluir que a operação de soma seja executada por circuitos específicos (somadores).

Observação sobre o carry: para manter a uniformidade dos nomes, mantemos aqui a notação inglesa, isto é, a letra C ("carry") para representá-lo. Utilizaremos **Cin** ("carry" e "in"), se for entrada de circuito, e **Cout** ("carry" e "out"), se for saída.

A **Figura 12** mostra um bloco chamado FA (*Full Adder*, em inglês, que quer dizer somador completo), no qual identificamos as entradas dos circuitos (A, B e C_{IN}) e a saída (S e C_{OUT}) tanto as entradas quanto às saídas possuem somente 1 bit.

Figura 12 - Bloco mostrando bits de entrada (A e B), carry de entrada (CIN) e bit de saída (S), que é o resultado da soma dos bits de entrada e o carry de saída (COUT)



Fonte: Tocci, Neal e Gregory (2007).

Para representar a tabela verdade do somador completo (**Tabela 1**), temos que considerar três entradas (A, B, C_{IN}), realizar as combinações e calcular as saídas S e C_{OUT}, respectivamente, a adição e o carry de saída.

Bit de entrada da 1ª parcela	Bit de entrada da 2ª parcela	Bit de entrada do carry	Bit de saída da soma	Bit de saída do carry
A	B	CIN	S	COUT
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

Tabela 1 - Tabela verdade de um somador de dois bits
Fonte: Tocci, Neal e Gregory (2007).

1. Se considerarmos a primeira linha da Tabela 1 (A = 0, B = 0 e C_{IN} = 0), a soma (S = A + B + C_{IN}) será igual a zero (S = 0) e o carry de saída (C_{OUT}) igual a zero (C_{OUT} = 0) .

2. Na 2ª linha da Tabela 1 (A = 0, B = 0 e C_{IN} = 1), a soma (S) será igual a um (S = 1) e o carry de saída (C_{OUT}) igual a zero (C_{OUT} = 0).

3. Na 3ª linha da Tabela 1 (A = 0, B = 1 e C_{IN} = 0), a soma (S) será igual a um (S = 1) e o carry de saída (C_{OUT}) igual a zero (C_{OUT} = 0).

4. Na 4ª linha da Tabela 1 (A = 0, B = 1 e C_{IN} = 1), a soma (S) será igual a zero (S = 0) e o carry de saída (C_{OUT}) igual a um (C_{OUT} = 1).

5. Na 5ª linha da Tabela 1 (A = 1, B = 0 e C_{IN} = 0), a soma (S) será igual a um (S = 1) e o carry de saída (C_{OUT}) igual a zero (C_{OUT} = 0).

6. Na 6ª linha da Tabela 1 (A = 1, B = 0 e C_{IN} = 1), a soma (S) será igual a zero (S = 0) e o carry de saída (C_{OUT}) igual a um (C_{OUT} = 1).

7. Na 7ª linha da Tabela 1 (A = 1, B = 1 e C_{IN} = 0), a soma (S) será igual a 0 (S = 0) e o carry de saída (C_{OUT}) igual a um (C_{OUT} = 1).

8. Por último, na 8ª linha da Tabela 1 (A = 1, B = 1 e C_{IN} = 1), a soma (S) será igual a um (S = 1) e o carry de saída (C_{OUT}) igual a um (C_{OUT} = 1).

Utilizando o que aprendemos sobre a porta lógica XOR e soma de produtos, para podermos visualizar melhor essa tabela, vamos fazer as seguintes considerações: escrever expressões a partir da tabela verdade, podemos escrever a soma S como a seguinte expressão:

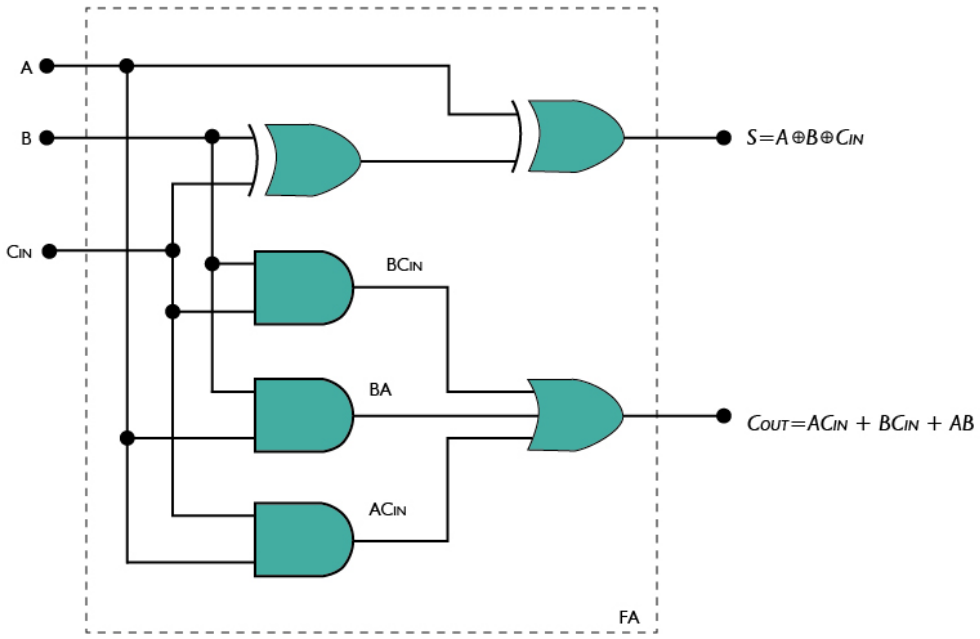
$$S = A \oplus B \oplus C_{in}$$

Da mesma forma, a expressão do carry de saída C_{OUT}, pode ser escrita como:

$$C_{out} = AC_{in} + BC_{in} + AB$$

Olhando para a expressão, podemos desenhar o circuito como mostrado na **Figura 13**. Para a expressão da adição, S, notamos que teremos duas portas XOR (OU EXCLUSIVO) e, para a expressão do C_{OUT}, teremos três portas AND e uma porta OR de três entradas.

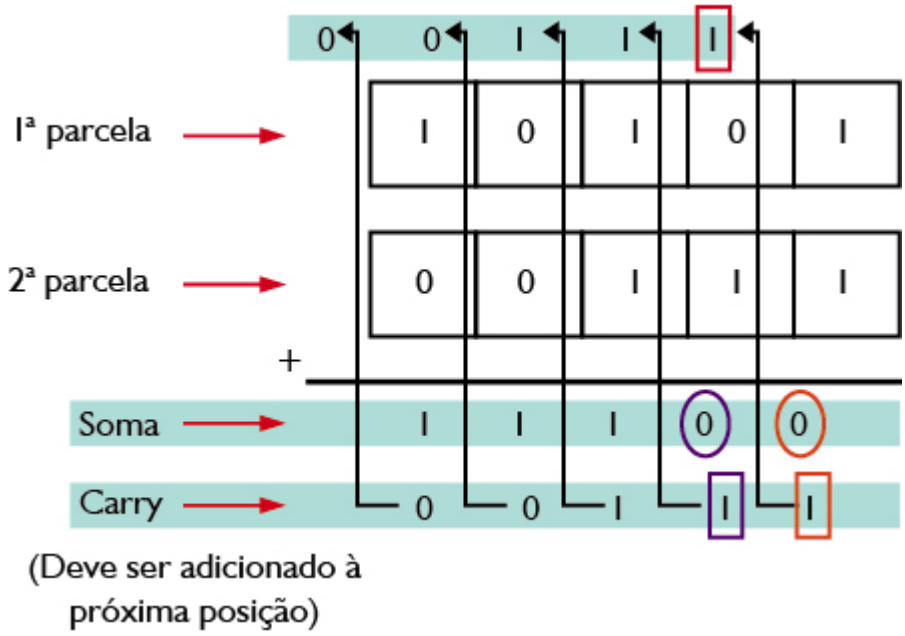
Figura 13 - Circuito somador completo (FA – fulladder)



Fonte: Tocci, Neal e Gregory (2007).

Se notarmos bem, esse circuito está realizando apenas a soma de dois bits, A e B, sem obviamente esquecer do carry. Mas e se quisermos realizar a soma de dois números binários que contenham vários bits, por exemplo, $10101 + 00111$, como mostrado na **Figura 14** a seguir.

Figura 14 - Soma de dois números binários com vários bits



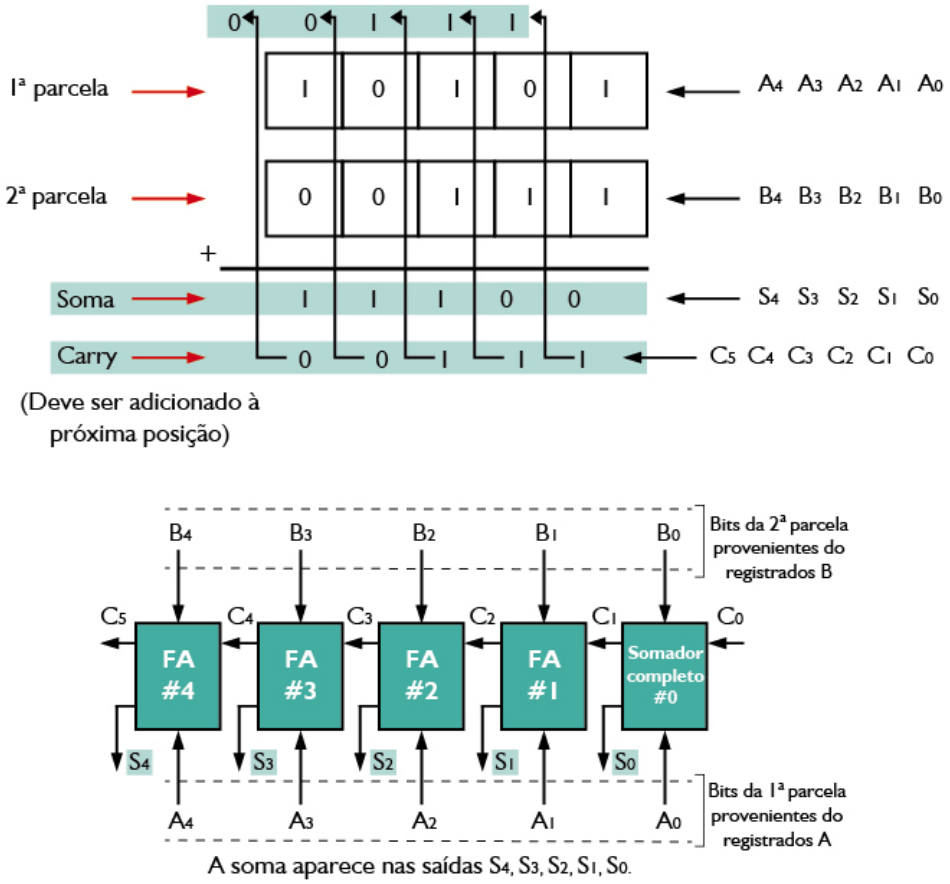
Fonte: Tocci, Neal e Gregory (2007).

A saída (Soma) terá como resultado 11100. Cada carry é transportado para a próxima parcela, assim, na primeira coluna da direita para esquerda temos 1 (1a parcela) + 1 (2a parcela), que será igual a $\boxed{0}$ com *carry* igual a $\boxed{1}$. Esse carry passa para a próxima parcela, então teremos $\boxed{1}$ (carry) + 0 (1a parcela) + 1 (2a parcela) = ... e com carry igual $\boxed{1}$. A próxima será 1 (carry) + 1 (1a parcela) + 1 (2a parcela) = 1 e com carry igual a 1. Depois 1 (carry) + 0 (1ª parcela) + 0 (2ª parcela) = 1 com carry igual a 0. Por último, 0 (carry) + 1 (1ª parcela) + 0 (2ª parcela) = 1 com o carry igual a 0 .

Finalmente, podemos afirmar que a soma de números binários com vários algarismos pode ser realizada com vários FA em paralelo. Para a soma de dois números de n bits são necessários *n* FullAdder de 1 bit.

Assim, poderíamos pensar em utilizar vários circuitos somadores completos em paralelo, um para cada soma de dois bits. Podemos representar essa soma como mostrado na Figura 15, na qual temos cinco circuitos trabalhando, cada bloco realiza a soma de dois bits (ex.: o bloco mais à direita soma A_0 e B_0, não esquecendo do carry) e assim sucessivamente para os outros bits.

Figura 15 - Circuito somador completo (FA – fulladder)



Fonte: Tocci, Neal e Gregory (2007).

Muito importante notar na Figura 15 que o carry de saída de um bloco (C_{OUT}) é o carry de entrada (C_{IN}) do bloco seguinte. Por exemplo, no somador completo #0, temos como entrada o carry C_0 e de saída o carry C_1.

Esse C_1 será entrada no bloco do somador #1. Já a saída C_2 do bloco somador #1, será entrada no bloco somador #2 e assim sucessivamente. Notamos que estamos transportando o carry como deve ser realmente realizado.

Atividade 04

---

1. Quantas entradas temos em um somador completo? E quantas saídas? Quais são as portas lógicas utilizadas?

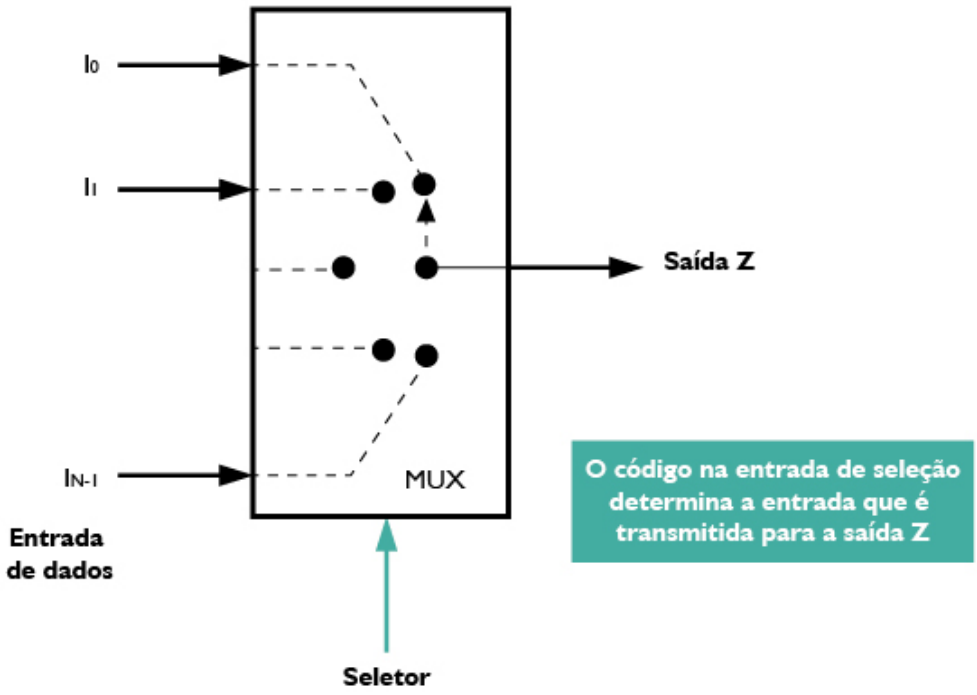
2. Quantos blocos de somadores completos (FA) serão necessários para somar dois números, cada um com 3 bits (por exemplo, $0\ 1\ 0 + 1\ 1\ 1$)?

3. Quantos somadores completos são necessários para somar um número de 10 bits a outro de 8 bits? Explique sua resposta.

Multiplexador (MUX) – Seletores de Dados

Um circuito multiplexador ou seletor de dados (**Figura 17**) tem o objetivo de selecionar, dentre várias entradas, uma entrada específica, por exemplo: em um sistema de som, podemos selecionar entre *compact disc* (CD), rádio (AM) e rádio (FM). Assim, teremos uma chave seletora, que selecionará uma das três opções e a enviará para o amplificador de saída e alto-falantes.

Figura 17 - Multiplexador (MUX) – seletor de dados

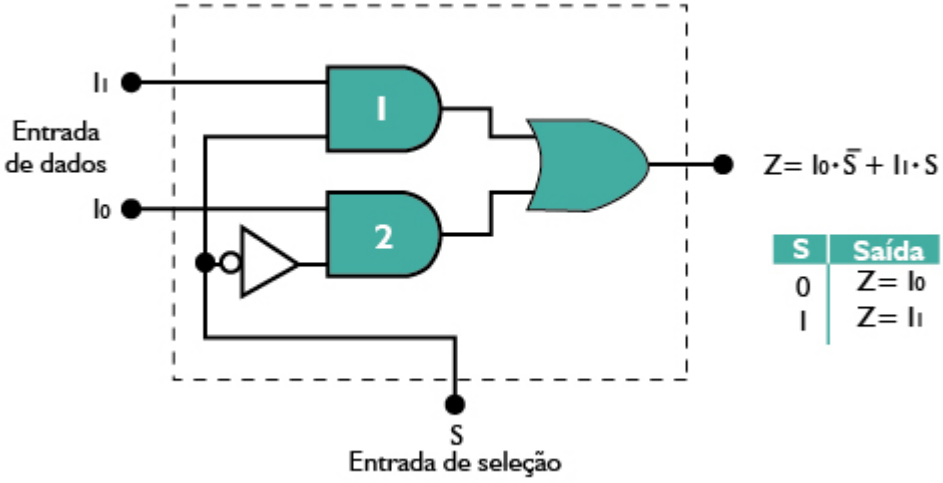


Fonte: Tocci, Neal e Gregory (2007).

Vamos pegar como exemplo um MUX de duas entradas (I_1 e I_0). Esse MUX (**Figura 18**) terá um seletor (S) que terá que decidir entre essas duas opções, assim, o seletor deverá ter apenas um *bit*. Se S for zero, então, na saída Z, teremos os

dados da entrada I_0 e, se S for um, teremos na saída Z a entrada I_1. Podemos expressar a saída Z pela seguinte expressão:

Figura 18 - Circuito multiplexador de duas entradas



Fonte: Tocci, Neal e Gregory (2007).

Podemos ter circuitos de quatro e de oito entradas, como mostrado na Figura 19. A mesma ideia do circuito de duas entradas é utilizada, porém, temos que ter atenção com o seletor, pois como temos mais entradas, teremos que colocar mais bits no seletor para representar essas novas entradas.

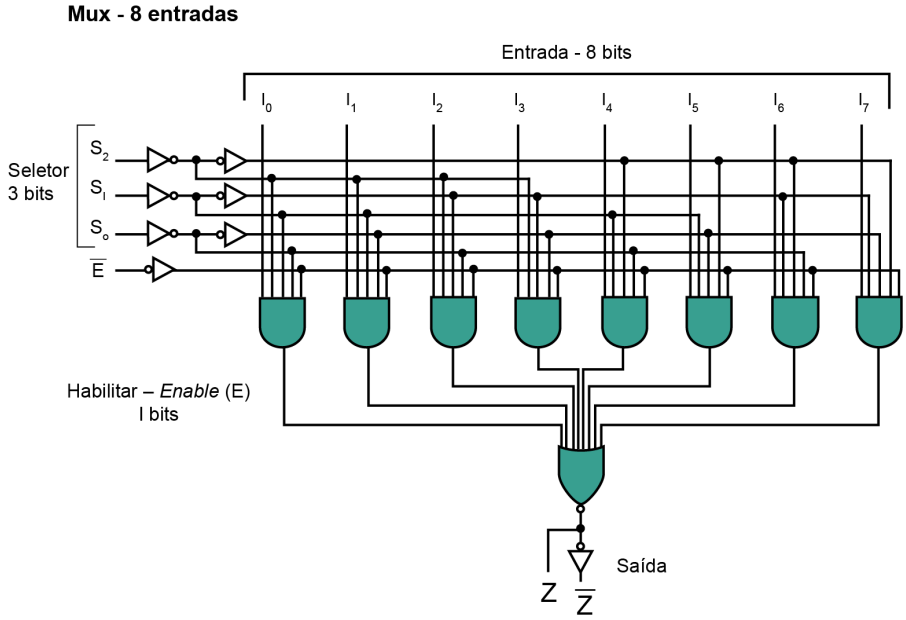
Para o MUX de quatro entradas, o seletor deverá ter no mínimo dois bits $(00, 01, 10, 11)$, pois assim teremos quatro opções para poder escolher uma das quatro entradas. Já para o MUX de oito entradas, o seletor deverá ter três bits $(000, 001, 010, \ldots, 110, 111)$, pois assim teremos oito opções para poder escolher entre as oito entradas.

No MUX de 8 bits de entradas, adicionamos uma entrada chamada de habilitador ou E (**Enable**, em inglês). Essa entrada tem por finalidade habilitar ou desabilitar o circuito. Perceba na Figura 19 que o habilitador é barrado (negado). Isso quer dizer que se E = 0 o circuito está habilitado. Por outro lado, caso E = 1, o circuito MUX não está habilitado.

ATENÇÃO: Assim como este MUX possui o habilitador barrado, muitos outros circuitos também o possuem. Se o habilitador estiver barrado, o circuito é habilitado em ‘0’ e desabilitado em ‘1’. Preste muita atenção!

Não se preocupe com a expressão booleana que descreve esses circuitos, o importante é você entender o princípio de funcionamento e qual a utilização desse circuito.

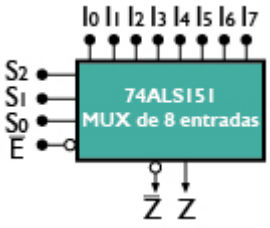
Figura 19 - Circuito multiplexador de 4 e 8 entradas



Fonte: Tocci, Neal e Gregory (2007).

A **Figura 20** mostra o chip de um circuito de um MUX de 8 entradas I (de I_0 até I_7), três seletores (S_0, S_1 e S_2), um habilitador barrado ($\overline{E}$) e duas saídas (Z e $\overline{Z}$)

Figura 20 - Chip de um circuito multiplexador de 8 entradas



Fonte: Tocci, Neal e Gregory (2007).

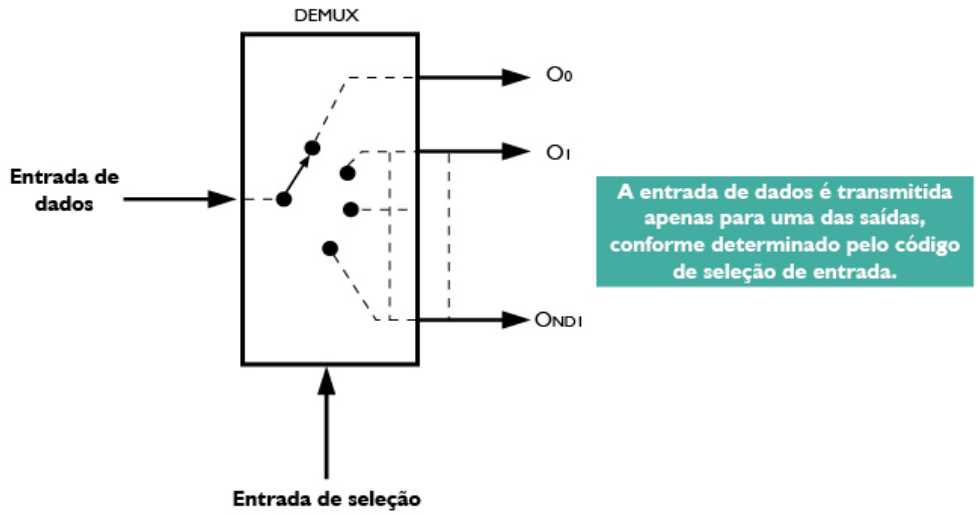
Atividade 05

1. Qual a função das entradas de seleção de um MUX?
2. Se um MUX deve escolher entre 32 entradas diferentes, quantas entradas diferentes deverá ter a chave seletora, ou seja, quantos bits deverá ter o seletor?
3. Implemente em VHDL um MUX de 32 entradas, conforme sua resposta na questão anterior. Não é necessário implementar o habilitador.

Demultiplexador (Demux)

Um circuito demultiplexador realizará a função inversa de um MUX. Teremos uma única entrada e ela será desmembrada ou distribuída para um dos vários canais (Figura 21). Da mesma maneira que no circuito de um MUX, teremos também o SELETOR, ou chave seletora.

Figura 21 - Um circuito demultiplexador genérico



Fonte: Tocci, Neal e Gregory (2007).

Observando a Figura 21, vemos que o circuito demultiplexador tem uma entrada de dados e várias saídas, não esquecendo da entrada seletora.

Atividade 06

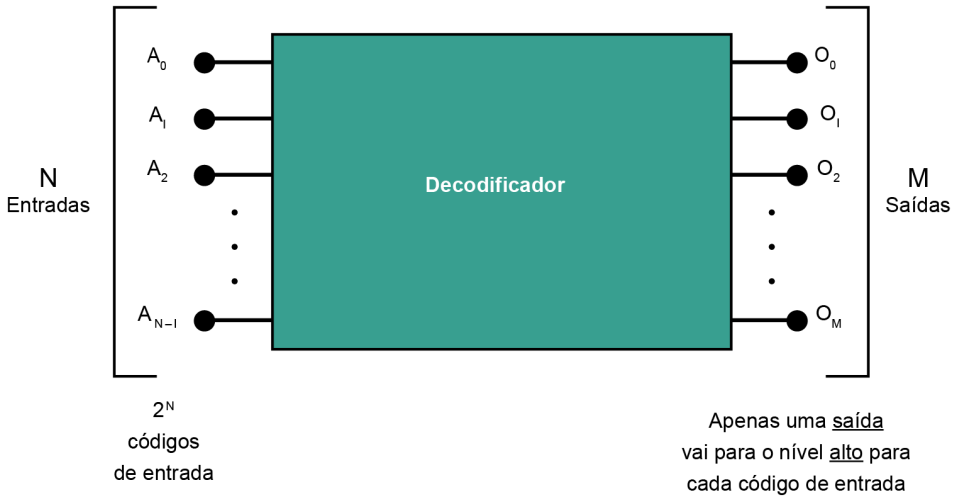
1. Qual a função de um circuito DEMUX?
2. Pesquise suas aplicações na internet e tente citar pelo menos três exemplos de aplicações.
3. Se um DEMUX possuir 1024 saídas, quantas entradas diferentes terá a chave seletora?
4. Implemente em VHDL um DEMUX de 32 entradas, conforme sua resposta na questão anterior. Não é necessário implementar o habilitador.

Decodificador

Os decodificadores são circuitos lógicos que convertem informações de um código para outro. Eles vão receber um conjunto de informações na entrada (N bits) que representa um número binário e ativam apenas uma das M saídas.

Na **Figura 22**, temos um decodificador genérico com várias entradas de dados e várias saídas, mas apenas uma saída é selecionada para cada valor da entrada.

Figura 22 - Um circuito decodificador genérico



Fonte: Tocci, Neal e Gregory (2007).

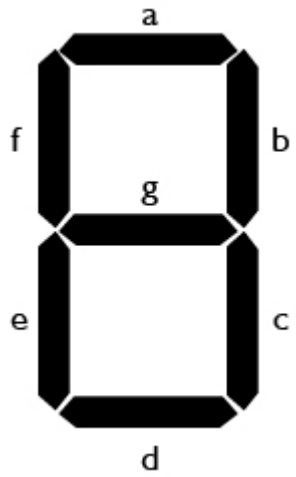
Uma das aplicações mais comuns dos decodificadores é a conversão de informações de um código para o acionamento de *displays*, de forma que algarismos ou letras codificados digitalmente sejam mais compreensíveis aos usuários. Vamos ver um exemplo com display de 7 segmentos? É só continuar estudando!

Decodificador BCD de 7 segmentos

Um decodificador bastante usado em sistemas digitais é o decodificador BCD de 7 segmentos. Você conhece esse decodificador? Sabe como ele funciona? Não? Então vamos conhecê-lo.

Esse é um dos decodificadores mais utilizados em sistemas digitais porque converte informações codificadas em BCD para um código especial que, aplicado ao *display* de 7 segmentos, fornece visualmente as informações. Os displays de 7 segmentos são dispositivos formados por 7 led (a,b,c,d,e,f,g) dispostos como mostra a **Figura 23**.

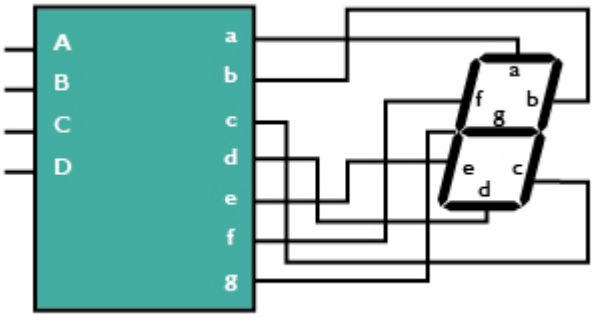
Figura 23 - Um display de 7 segmentos com led



Fonte: Tocci, Neal e Gregory (2007).

Como ele pode indicar dígitos de 0 a 9 (10 dígitos), a informação binária precisa ter 4 dígitos binários, pois com três bits somente oito valores poderiam ser representados. Podemos então imaginar um circuito como o da **Figura 24**.

Figura 24 - Circuito conectado ao display de 7 segmentos (a,b,c,d,e,f,g)

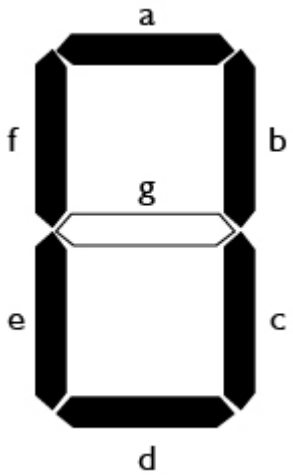


Fonte: <http://www.mspc.eng.br/eledig/eldg0610.shtml> Acesso em: 12 jul. 2012.

Cada um dos 7 segmentos do *display* é formado por um led. Esses led podem ser acesos no nível lógico “1”, se estiverem conectados pelo catodo (catodo comum), ou no nível lógico “0”, se estiverem conectados pelo ânodo (ânodo comum).

Por exemplo, para o código em BCD igual a “0000”, equivalente ao algarismo decimal zero, somente o segmento “g” do display deve permanecer apagado, como mostrado na **Figura 25**.

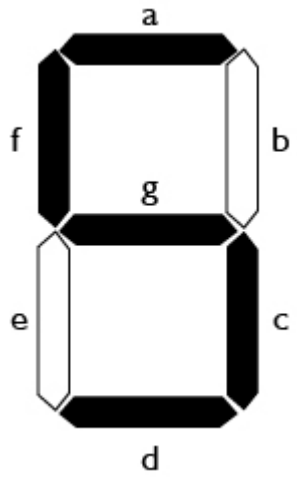
Figura 25 - A representação do número 0 (Note que o segmento g está apagado)



Fonte: Tocci, Neal e Gregory (2007).

Para o código BCD em “0101”, equivalente em decimal ao algarismo 5, teremos a situação mostrada na **Figura 26**, na qual os segmentos “b” e “e” estão apagados.

Figura 26 - Representação do número 5 (Note que os segmentos h e e estão apagados)



Fonte: Tocci, Neal e Gregory (2007).

O mesmo raciocínio é utilizado para o restante dos algarismos de 0 a 9. Os números decimais maiores que 9 podem ser representados utilizando-se um *display* de 7 segmentos para cada casa decimal. Alguns *displays* podem possuir um segmento a mais no formato de ponto para indicar casas decimais.

No circuito decodificador, ABCD são as quatro entradas binárias e abcdefg são as saídas para os sete segmentos do *display*. A tabela verdade da Tabela 2 segue esse raciocínio.

	A	B	C	D		a	b	c	d	e	f	g
0	0	0	0	0		I	I	I	I	I	I	0
I	0	0	0	I		0	I	I	0	0	0	0
2	0	0	I	0		I	I	0	I	I	0	I
3	0	0	I	I		I	I	I	I	0	0	I
4	0	I	0	0		0	I	I	0	0	I	I
5	0	I	0	I		I	0	I	I	0	I	I
6	0	I	I	0		I	0	I	I	I	I	I
7	0	I	I	I		I	I	I	0	0	0	0
8	I	0	0	0		I	I	I	I	I	I	I
9	I	0	0	I		I	I	I	I	0	I	I
	I	0	I	0		Ø	Ø	Ø	Ø	Ø	Ø	Ø
	I	0	I	I		Ø	Ø	Ø	Ø	Ø	Ø	Ø
	I	I	0	0		Ø	Ø	Ø	Ø	Ø	Ø	Ø
	I	I	0	I		Ø	Ø	Ø	Ø	Ø	Ø	Ø
	I	I	I	0		Ø	Ø	Ø	Ø	Ø	Ø	Ø
	I	I	I	I		Ø	Ø	Ø	Ø	Ø	Ø	Ø

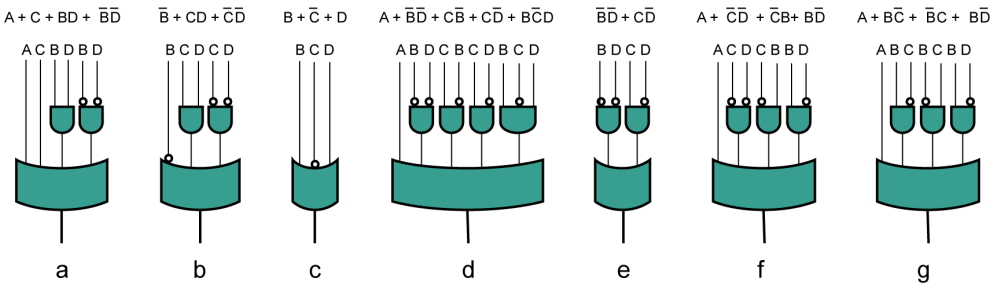
Tabela 2 - Tabela verdade do decodificador BCD de 7 segmentos.
Fonte: <http://www.mspc.eng.br/eledig/eldg0610.shtml>. Acesso em 12 jul. 2012.

A notação Ø indica valor indiferente (pode ser 0 ou 1), uma vez que não há valor decimal a indicar acima da combinação 9. O circuito que fornece as entradas deve evitar combinações nesses casos (algumas vezes, as combinações que sobram, total de seis, são usadas para sinal negativo, sinal de erro e outros).

Trabalhando com a tabela verdade, podemos realizar simplificações e chegar aos circuitos digitais para cada saída, como apresentado a seguir. Não vamos nos deter por enquanto como chegamos nesses circuitos, pois isso foge um pouco do escopo do nosso curso. Porém, apresentamos esses circuitos lógicos na Figura 27.

Você pode identificar quais são as portas lógicas utilizadas em cada função, pois cada função terá como saída uma letra (a, b, c, d, e, f, g) representando o segmento do *display* que irá acender. A primeira expressão resultará no segmento "**a**" aceso.

Figura 27 - Representação de como montar cada circuito lógico para o decodificador BCD de 7 segmentos



Fonte: <http://www.mspc.eng.br/eledig/eldg0610.shtml> Acesso em: 12 jul. 2012

É evidente que, com os circuitos integrados disponíveis, dificilmente alguém irá montar o circuito anterior. Isso serve apenas para mostrar como funciona.

A Figura 28 mostra o diagrama de pinos do decodificador para *display* CD4511BC da Fairchild Semiconductor.

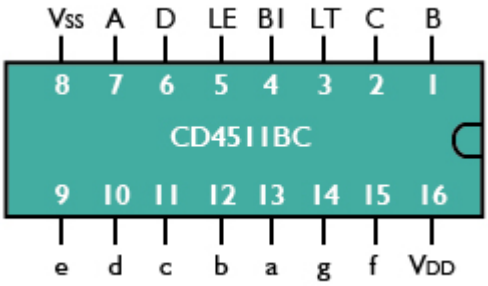
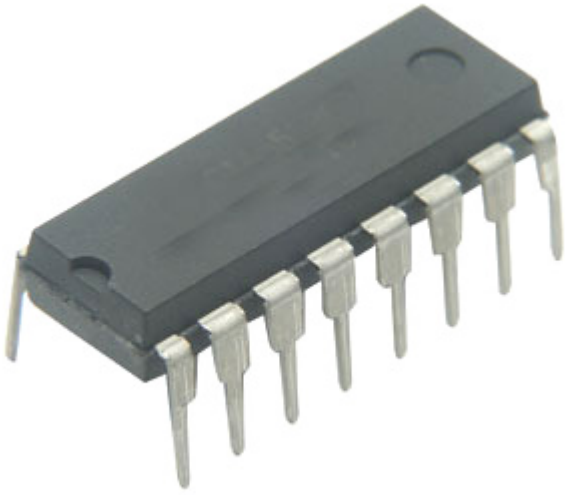


Figura 28 - (a) Diagrama do chip CD4511BC; (b) foto do circuito integrado SN74LS47N também utilizado como decodificador



Fonte: <http://www.datasheetcatalog.com/datasheets_pdf/S/N/7/4/SN74LS47N.shtml>; <http://www.mspc.eng.br/eledig/eldg0610.shtml> Acesso em: 18 ago. 2010.

Observando a Figura 28, você perceberá que temos as entradas binárias A, B, C e D nos pinos 7, 1, 2 e 6, respectivamente. Perceberá também que as saídas “a”, “b”, “c”, “d”, “e”, “f” e “g” para o *display* de 7 segmentos estão nos pinos 13, 12, 11, 10, 9, 15 e 14, respectivamente. E nos pinos 16 e 8, você terá, respectivamente: V_{DD}, que é a tensão de alimentação (3 a 15 V), e V_{SS}, que é o terra.

Atividade 07

1. Qual a função de um decodificador?
2. Procure na internet algumas aplicações para o decodificador.
3. O que é um decodificador BCD para 7 segmentos?
4. Um decodificador pode ter mais de uma saída ativada de cada vez?
5. Implemente em VHDL um Decodificador BCD de 7 segmentos.

   DICA: Utilizar vetores tanto para entrada quando para saída pode facilitar e reduzir a implementação.

Comparador

Já estudamos até aqui os seguintes circuitos combinacionais básicos: somador, multiplexador, demultiplexador e decodificador. Vamos agora estudar o circuito combinacional comparador.

Um circuito comparador combinacional compara duas entradas binárias (A e B) para definir se existe uma igualdade ou diferença entre essas entradas.

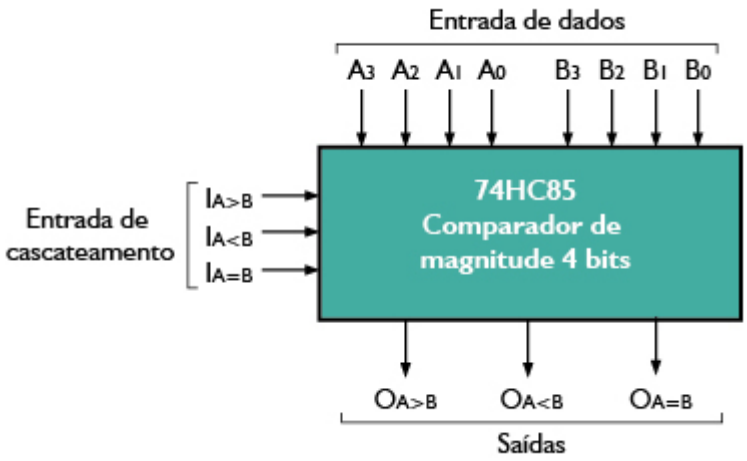
O resultado dessa comparação é expresso em uma das três saídas desse circuito: $A = B, A > B$ e $A < B$. Dependendo do resultado da comparação, apenas uma dessas saídas (a que for verdadeira) será colocada em “1”. As outras duas terão valor “0”.

Mas, e quando quisermos comparar entradas com mais de um bit? Será que é possível comparar números com vários bits?

A Figura 29 responde a essa pergunta. Ela mostra o diagrama de um comparador de 4 bits, no qual as entradas **A** (A_3, A_2, A_1, A_0) e **B** (B_3, B_2, B_1 e B_0) são compostas de 4 bits cada uma, ou seja, teremos 4 entradas de um bit para o número **A** e 4 entradas de um bit para a entrada **B**.

Mas observe que a quantidade de saídas continua a mesma: **3**.

Figura 29 - Símbolo lógico para um comparador de quatro bits 74HC85 (7485, 74LS85)



Fonte: Tocci, Neal e Gregory (2007).

Olhando para a Figura 29, vemos as saídas:

- $O_{A>B}$: será "1", quando o nº binário A for MAIOR que o nº binário B .
- $O_{A<B}$: será "1", quando o nº binário A for MENOR que o nº binário B.
- $O_{A=B}$: será "1", quando o nº binário A for IGUAL ao nº binário B.

Vemos ainda outras três entradas ($I_{A>B}$, $I_{A<B}$ e $I_{A=B}$), chamadas entradas de cascateamento. Essas entradas podem ser usadas para conectar as saídas de outro comparador e assim podermos expandir a quantidade de bits dos números A e B que queremos comparar.

Então, pela Figura 29, temos as entradas (A_3, A_2, A_1, A_0 e B_3, B_2, B_1, B_0) e três saídas de comparação. Além disso, existem também as entradas de cascateamento ($I_{A>B\$}$,l_{A < B}el_{A=B}$) que podem ser conectadas a outros circuitos para expansão da quantidade de bits na operação de comparação.

A Tabela 3 mostra a tabela verdade desse comparador de 4 bits.

Tabela-verdade

Entradas de comparação				Entradas de cascateamento			Saídas		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$O_{A>B}$	$O_{A<B}$	$O_{A=B}$
$A_3>B_3$	X	X	X	X	X	X	H	L	L
$A_3<B_3$	X	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X	H	L	L
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	L	L	H	H	H
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	H	L	L	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	X	H	L	L	H
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	L	L	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	H	L	L	H	H

H= Nível de tensão **Alto**
L= Nível de tensão **Baixo**
X= Irrelevante

Tabela 3 - Tabela verdade de um circuito comparador
Fonte: Tocci, Neal e Gregory (2007).

Observando a tabela verdade do comparador na Tabela 3, vemos na primeira linha que se $A3 > B3$, então, a saída $O_{A>B}$ terá o nível de tensão alto (H = "1") e as outras saídas terão o nível de tensão baixo (L="0"), pois o nº A será maior que o nº B.

Na 2ª linha, podemos notar que se $A_3 < B_3$, a saída $O_{A>B}$ estará em nível baixo (L), a saída $O_{A<B}$ estará em nível alto (H) e a saída $O_{A=B}$ estará em nível lógico baixo (L), sinalizando que o nº A é menor que o nº B.

Você percebe, através dessas duas linhas, que não precisamos verificar os demais bits (A_2 e B_2; A_1 e B_1; A_0 e B_0), pois se os bits mais significativos de cada um dos números (A_3 e B_3) são diferentes, isso indica que $A > B$ ou que $A < B$. Somente precisaremos verificar um bit menos significativo quando os bits mais significativos que ele forem iguais aos do outro número que se quer comparar. É o que vemos, por exemplo, na terceira e na quarta linha, quando $A_3 = B_3$.

Nesse caso, iremos comparar A_2 e B_2. Se $A_2 > B_2$ (3ª linha), a saída $O_{A>B}$ estará em nível alto (H), a saída $O_{A<B}$ estará em nível baixo (L) e a saída $O_{A=B}$ estará em nível baixo (L). Por sua vez, se $A_2 < B_2$ (4ª linha), a saída $O_{A>B}$ estará em nível baixo (L), a saída $O_{A<B}$ estará em nível alto (H) e a saída $O_{A=B}$ estará em nível baixo (L).

Quando $A_2 = B_2$, a comparação passará para o bit seguinte (A_1 e B_1), como vimos na 5ª e na 6ª linha da tabela. E quando $A_1 = B_1$, a comparação segue para o bit seguinte (A_0 e B_0), como mostram as duas linhas seguintes (7ª e 8ª linhas).

Quando todos os bits forem iguais ($A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$ e $A_0 = B_0$), o comparador avaliará as entradas de cascateamento para definir se $A > B$, $A < B$ ou se $A = B$, como mostram as demais linhas da tabela verdade.

Atividade 08

---

1. Analise as linhas 4, 5, 8 e 11 da Tabela 3.
2. Pesquise na internet uma utilização para um circuito comparador.
3. Implemente em VHDL um circuito comparador de 4 bits, conforme Figura 29. Não é necessário implementar as entradas de cascateamento.

Roteiro Prático da Aula 07

Apresentação

Neste roteiro, você irá implementar um circuito combinacional usado em todas as aplicações que envolvem aritmética, o somador completo. Serão necessários os conceitos da linguagem VHDL e do software Quartus. Você vai aprender a criar circuitos a partir de diagramas de blocos no software Quartus.

Objetivos

Ao final das atividades previstas para este roteiro, você será capaz de:

- Implementar o circuito do somador;
- Criar circuitos utilizando diagramas de blocos.

Relembrando: Circuito Somador

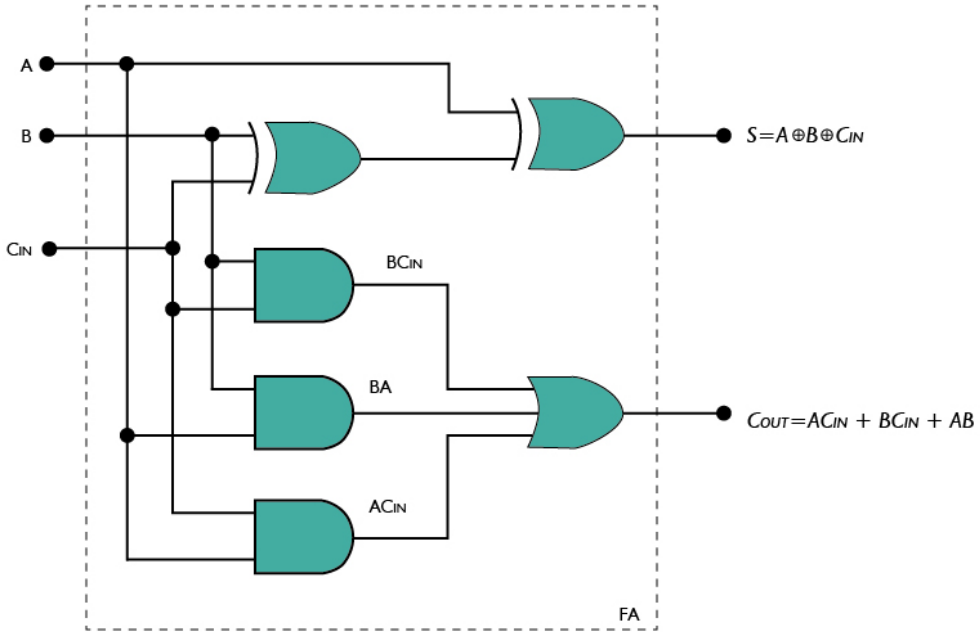
Como visto nas aulas anteriores, o circuito somador completo é definido por duas expressões booleanas, mostradas abaixo.

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

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

Com essas expressões acima fica fácil de construir o circuito em VHDL! Basta saber que a operação do circuito “Exclusive-OR” é definido como “xor” na linguagem. Podemos facilmente descrever o circuito do somador completo em diagramas de blocos através dessas expressões, como mostrado na imagem abaixo.

Figura 30 - Circuito Somador



Construindo o Circuito em VHDL

Como já vimos em outras aulas, um circuito é facilmente construído na linguagem VHDL quando tem as expressões booleanas. Basta descrever as expressões dentro da arquitetura do código, como o exemplo abaixo.

- Expressão

$$W = (Y \oplus X) + K$$

- Código

```
1        architecture exemplo of entidadeEx
2        begin
3        w <= (y xor x) or k;
4      end exemplo;
5
```

Atividade 09

a. Crie um projeto no Quartus II de nome “somador”;

b. Crie um arquivo VHDL para o somador;

1. Na entidade defina 3 entradas (A, B, C_{in}) e 2 saídas (S, C_{out});
2. Na arquitetura defina as expressões booleanas do circuito somador completo;

OBS: Lembre-se que o nome do arquivo VHDL deve ser igual ao do projeto para compilar normalmente.

c. Compile.

A tabela abaixo vai lembrar a você como é realizada a soma binária!

A	**B**	**S**	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabela 4 - Soma binária

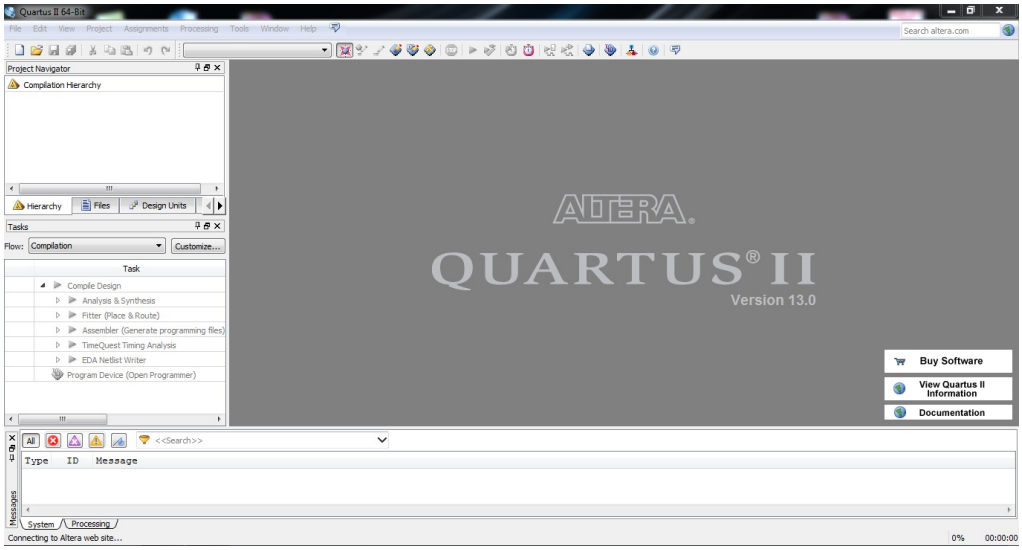
Atividade 10

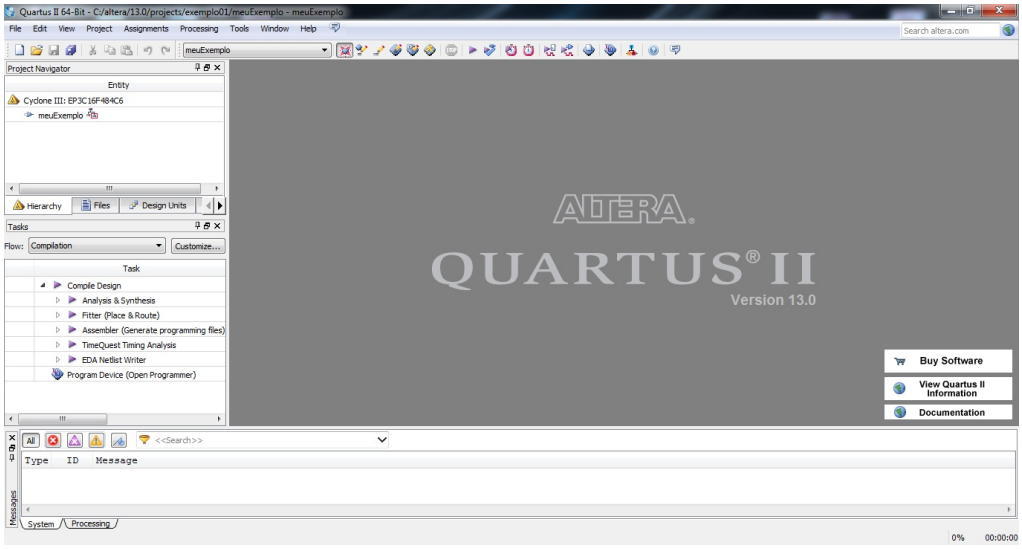
a. Crie um arquivo “University Program VWF”;

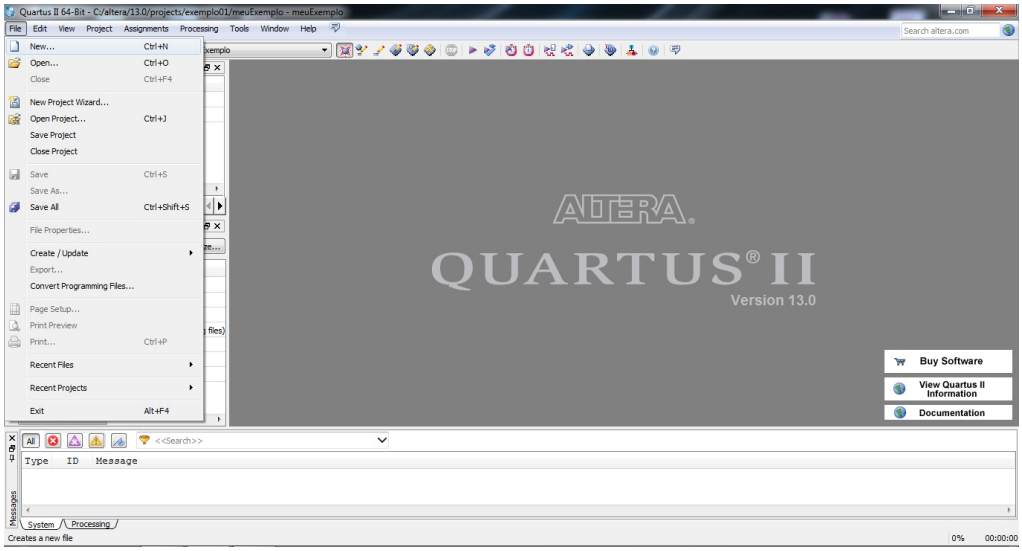
b. Simule o circuito com os valores da Tabela 01.

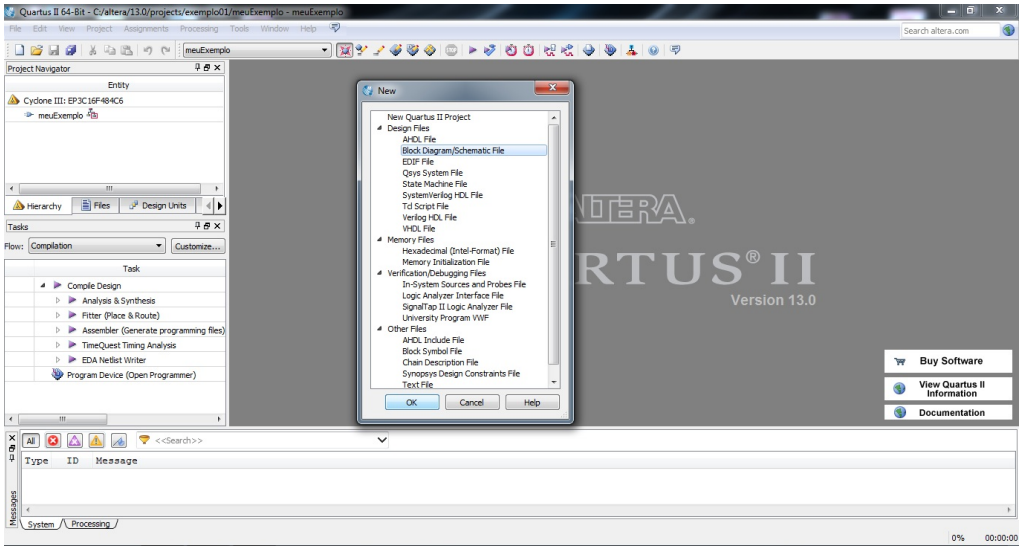
Construindo o Circuito em Diagrama de Blocos

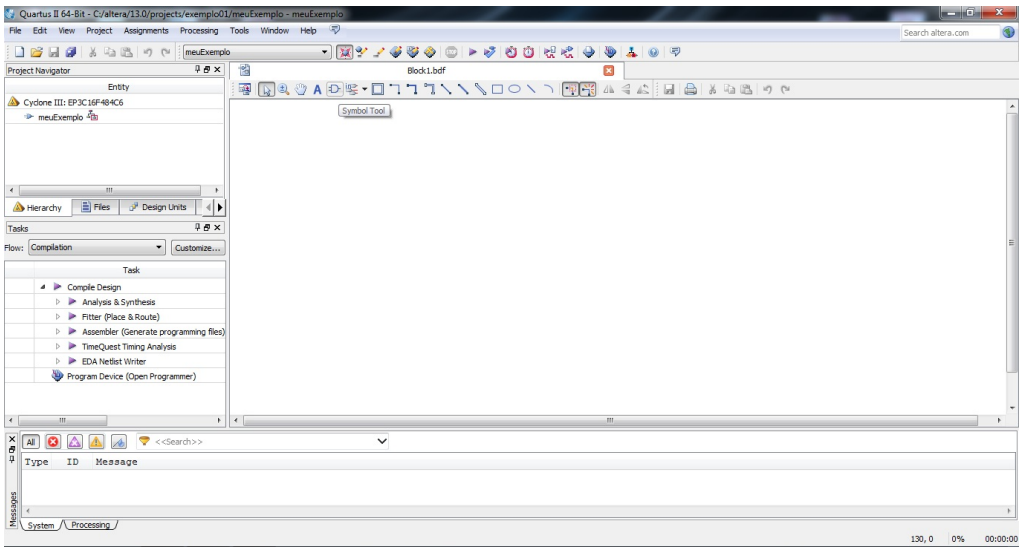
O Quartus permite que você construa o circuito sendo descrito em forma de diagrama de blocos. Essa forma permite uma visualização melhor do circuito e muitas vezes maior facilidade de descrevê-lo. O slideshow abaixo exemplifica a criação de um circuito a partir de diagrama de blocos ou esquemático.

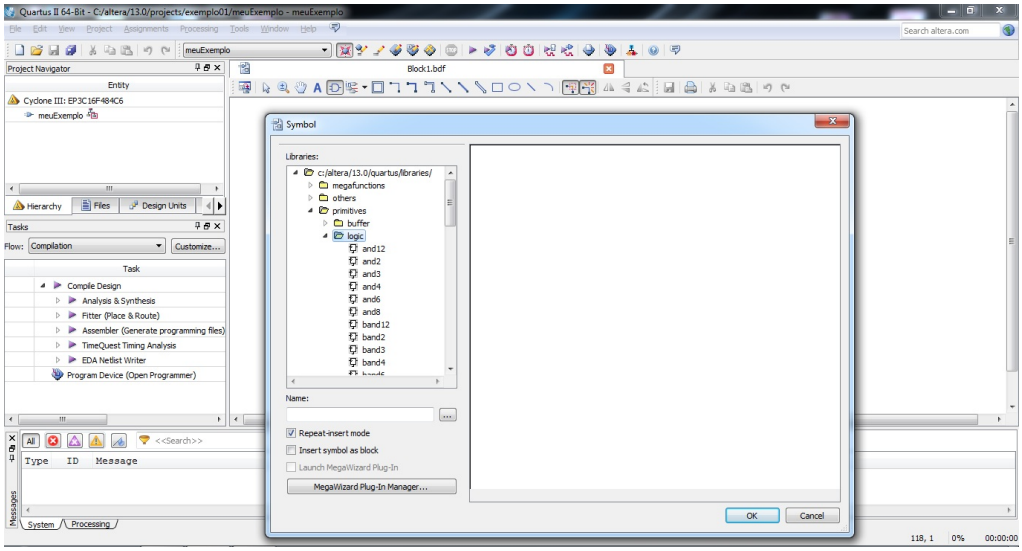


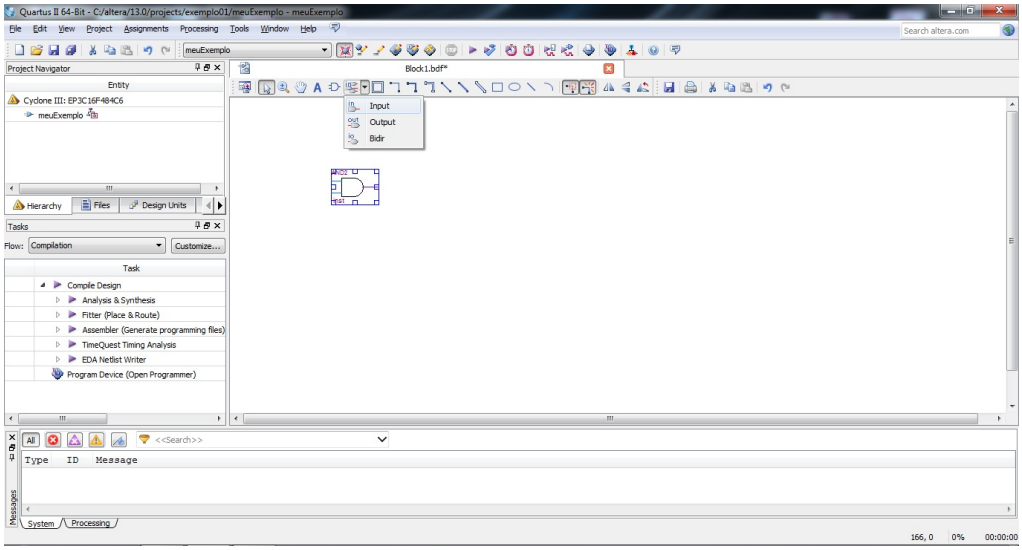


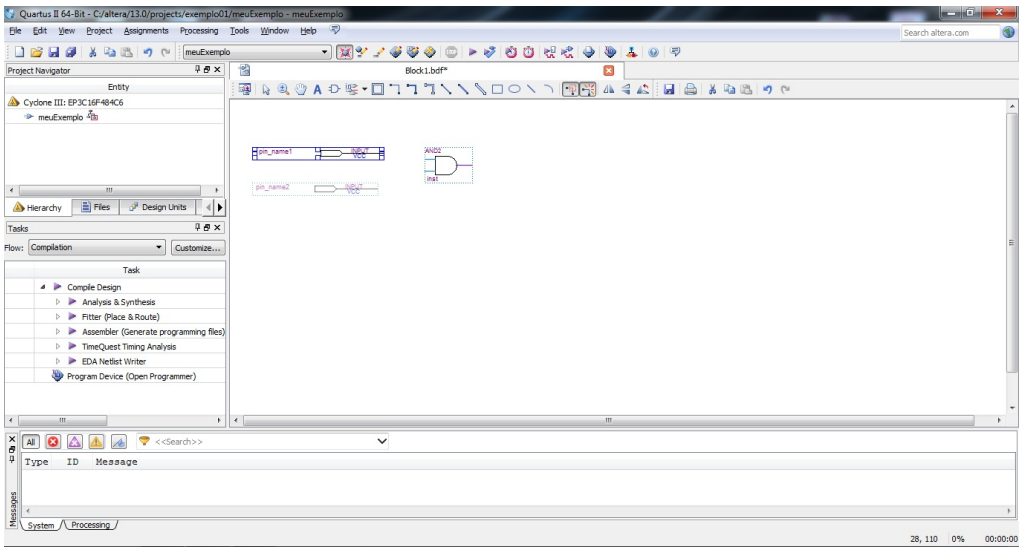

Quartus II 64-Bit - C:/altera/13.0/projects/exemplo01/meuExemplo - meuExemplo
File Edit View Project Assignments Processing Tools Window Help
New... Ctrl+N
Open... Ctrl+O
Close Ctrl+F4
New Project Wizard...
Open Project... Ctrl+J
Save Project
Close Project
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
File Properties...
Create / Update
Export...
Convert Programming Files...
Page Setup...
Print Preview
Print... Ctrl+P
Recent Files
Recent Projects
Exit Alt+F4
ALTERA
QUARTUS® II
Version 13.0
Buy Software
View Quartus II Information
Documentation
Type ID Message
System Processing
Creates a new file
0% 00:00:00

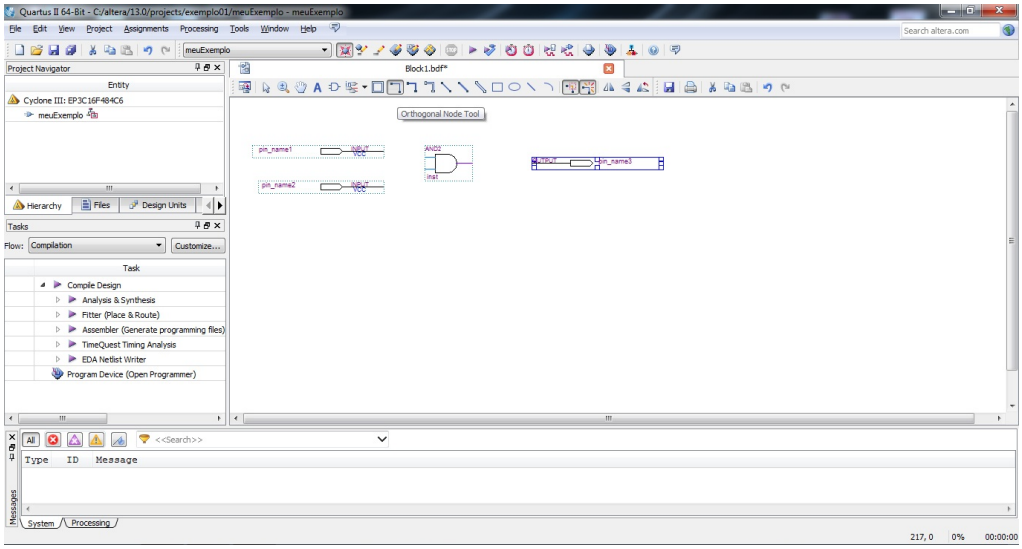

Quartus II 64-Bit - C:/altera/13.0/projects/exemplo01/meuExemplo - meuExemplo
Project Navigator
Entity
Cyclone III: EP3C16F484C6
meuExemplo
Hierarchy Files Design Units
Tasks
Flow: Compilation Customize...
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
TimeQuest Timing Analysis
EDA Netlist Writer
Program Device (Open Programmer)
New
New Quartus II Project
Design Files
AHDL File
Block Diagram/Schematic File
EDIF File
Qsys System File
State Machine File
SystemVerilog HDL File
Tcl Script File
Verilog HDL File
VHDL File
Memory Files
Hexadecimal (Intel-Format) File
Memory Initialization File
Verification/Debugging Files
In-System Sources and Probes File
Logic Analyzer Interface File
SignalTap II Logic Analyzer File
University Program VWF
Other Files
AHDL Include File
Block Symbol File
Chain Description File
Synopsys Design Constraints File
Text File
OK Cancel Help
Version 13.0
Buy Software
View Quartus II Information
Documentation
Type ID Message
System Processing
0% 00:00:00

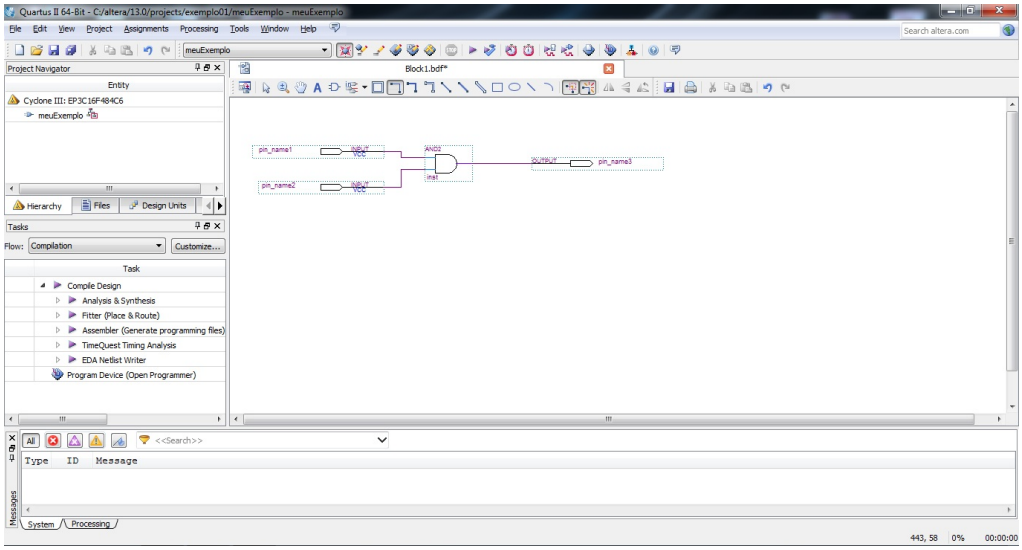

Quartus II 64-Bit - C:/altera/13.0/projects/exemplo01/meuExemplo - meuExemplo
Project Navigator
Entity
Cyclone III: EP3C16F484C6
meuExemplo
Block1.bdf
Symbol Tool
Hierarchy Files Design Units
Tasks
Flow: Compilation Customize...
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
TimeQuest Timing Analysis
EDA Netlist Writer
Program Device (Open Programmer)
Type ID Message
System Processing
130, 0 0% 00:00:00


Quartus II 64-Bit - C:/altera/13.0/projects/exemplo01/meuExemplo - meuExemplo
Symbol
Libraries:
c:/altera/13.0/quartus/libraries/
megafunctions
others
primitives
buffer
logic
and12
and2
and3
and4
and6
and8
band12
band2
band3
band4
Name:
Repeat-insert mode
Insert symbol as block
Launch MegaWizard Plug-In
MegaWizard Plug-In Manager...
OK
Cancel


Quartus II 64-Bit - C:/altera/13.0/projects/exemplo01/meuExemplo - meuExemplo
Input
Output
Bidir


Quartus II 64-Bit - C:/altera/13.0/projects/exemplo01/meuExemplo - meuExemplo





A criação através do esquemático é apenas uma forma mais fácil de descrever alguns circuitos, porém é importante saber que o mesmo encapsula um arquivo HDL. Assim o circuito descrito em forma de esquemático é transformado em um arquivo de linguagem de descrição de hardware (HDL), porém essa transformação é invisível para o usuário.

Atividade 11

---

a. Crie um novo projeto no Quartus II de nome “somador2”;

b. Crie um arquivo “Block Diagram/Schematic File” e descreva o circuito mostrado na figura 01;

   OBS: Verifique as conexões das portas e pinos.

c. Compile;

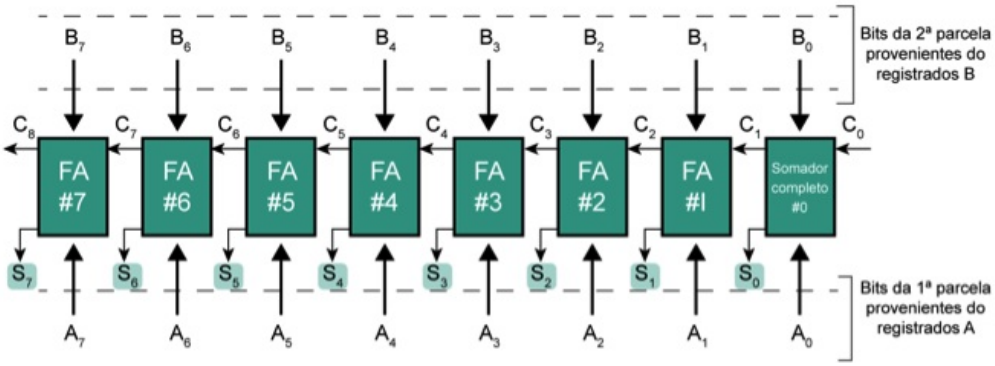
d. Simule com os mesmos valores da tarefa 02.

OBS: Os resultados das simulações devem ser iguais.

Construindo somadores completos

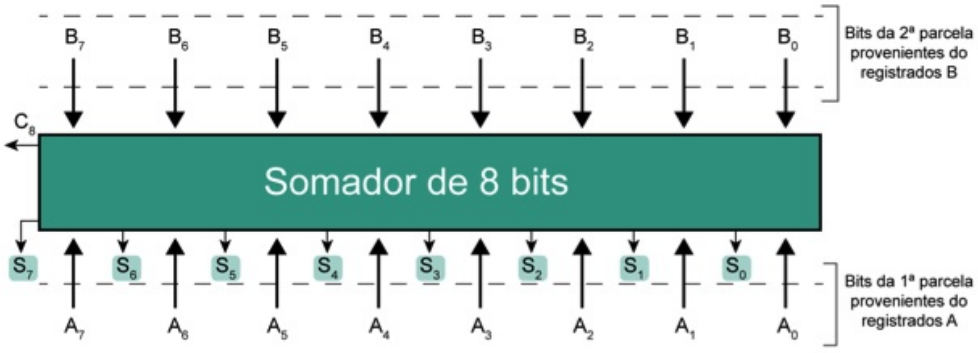
Agora que você entendeu como funciona um somador, que tal fazermos um somador de 8 bits? Para tal, precisamos de 8 Somadores completos, cada um deles 3 entradas (A, B, C_{in}) e duas saídas (S e C_{out}), conforme **Figura 31** abaixo.

Figura 31 - Circuito somador de 8 bits



Entretanto, você sabe muito bem que não estamos interessados em Carry, exceto o último (C_8). Ainda mais, o C_0 é sempre '0' quando somamos dois números. Afinal, o que importa é entrar com dois valores e obter a saída e o último carry. Desta forma, podemos interpretar um Somador de 8 bits como mostra a **Figura 32**.

Figura 32 - Circuito somador de 8 bits para implementação



Vamos então implementar e simular?

Atividade 12

a. Crie um projeto chamado somador_8bits

b. Adicione o código VHDL abaixo

```
1
2       library ieee;
3       use ieee.std_logic_1164.all;
4
5       ENTITY somador_8bits IS
6       PORT (
7          A,B: in std_logic_vector  ( 7 downto 0);
8          S: out std_logic_vector ( 7 downto 0);
9          carry_final: out std_logic
10      );
11      END somador_8bits;
12
13      ARCHITECTURE arquitetura_somador8bits OF somador_8bits IS
14      BEGIN
15            process (A,B)
16            variable Cout: std_logic_vector  ( 7 downto 0); --Carry Out é variável
17            interna ao processo
18            begin
19               for I in 0 to 7 loop
20                  if (I=0) then
21                     S(I) <= (A(I) xor B(I)) xor '0';
22                     Cout(I) := (A(I) and '0') or (B(I) and '0') or (A(I) and B(I));
23                  else
24                     S(I) <= (A(I) xor B(I)) xor Cout(I-1);
25                     Cout(I) := (A(I) and Cout(I-1)) or (B(I) and Cout(I-1)) or (A(I) and B(I));
26                  end if;
27               end loop;
28               carry_final <= Cout(7);
29
30         end process;
31      END arquitetura_somador8bits;
32
```

c. Em diversos momento do código são utilizados comandos como S(I) ou A(I), onde S e A são vetores e I é um inteiro. O que isto faz? Procure na internet.

d. Entenda o que o código está fazendo. Se não conseguir entender tudo, pergunte o que não entendeu ao Tutor

e. Simule o código abaixo utilizando o Simulation Waveform Editor

f. A simulação está correta?

Leitura Complementar

Se você quiser estudar mais um pouco e saber mais sobre circuitos combinacionais, você poderá encontrar mais informações nestes links.

- <http://pt.wikipedia.org/wiki/Multiplexador>
- <http://pt.wikipedia.org/wiki/Contadores_digitais>
- <http://pt.wikipedia.org/wiki/Decodificador>

Resumo

Nesta aula, você aprendeu a avaliar uma expressão em soma de produtos e produto de somas.

Você viu como escrever a expressão a partir de uma tabela verdade e, por consequência, representar o circuito digital.

Você estudou os circuitos das portas lógicas XOR e XNOR e compreendeu seu funcionamento.

Você viu também que para representar um número binário com sinal, um bit de sinal é adicionado como MSB.

Coloca-se “0” para o número positivo e “1” para o número negativo. Estudou o circuito somador completo e viu que ele realiza a operação de adição de dois bits mais o carry. Além disso, você aprendeu que um somador binário em paralelo é realizado adicionando-se somadores completos em cascata.

Você viu que o circuito multiplexador (MUX) tem como função escolher, dentre uma das entradas, qual estará na saída. A seleção é realizada através de uma chave seletora. Além disso, você estudou que um circuito demultiplexador realizará a função inversa de um MUX, teremos uma única entrada e essa entrada será desmembrada ou distribuída por vários canais.

Você viu também que os decodificadores são circuitos lógicos que convertem informações de um código para outro. Ele vai receber um conjunto de informações na entrada que representa um número binário e ativa apenas a saída que corresponde ao número recebido. Um bom exemplo é o decodificador BCD de 7 segmentos.

Por fim vimos o circuito comparador combinatório que compara duas entradas binárias (A e B de n bits) para indicar a relação de igualdade ou desigualdade entre elas por intermédio de “três bandeiras lógicas” (*flags*) que correspondem às relações A igual a B (A = B), A maior que B (A > B) e A menor que B (A < B).

No roteiro prático você reforçou seus conhecimentos sobre o software Quartus II. Você construiu o circuito do somador tanto escrevendo o código em VHDL como descrevendo o diagrama de blocos do circuito.

Autoavaliação

1. Realize as seguintes adições:
    a. $0011 + 0001$
    b. $01011 + 11100$
    c. $00 + 10$
    d. $001 + 101$
2. Represente os seguintes números decimais em binário:
    a. $+9$
    b. -9
    c. $+128$
    d. -128
    e. $+84$
    f. $+3$
    g. -1

3. Para cada item, indique se ele se refere a um MUX ou a um DEMUX.
    a. Usa entradas de seleção.
    b. Apenas uma de suas saídas pode ser ativada de uma vez.
    c. Tem apenas uma entrada e várias saídas.
    d. Tem a função de selecionar dados.
4. Qual é a utilização de um circuito comparador?
5. Pesquise algum equipamento que utilize um decodificador BCD de 7 segmentos.

Referências

COSTA, Cesar da. Projetos de Circuitos Digitais com FPGA. Editora Érica, 2009.

DATASHEETCATALOG.COM. Disponível em: <http://www.datasheetcatalog.com/datasheets_pdf/S/N/7/4/SN74LS47N.shtml>. Acesso em: 14 ago. 2012.

MSPC: informações técnicas. Eletrônica digital VI-10: Decodificador para display de sete segmentos. Disponível em: <http://www.mspc.eng.br/eledig/eldg0610.shtml>. Acesso em: 14 ago. 2012.

PEDRONI, Volnei. Eletrônica Digital moderna e VHDL. Rio de Janeiro: Elsevier, 2010.

RONALD, Tocci; NEAL, S. Widner; GREGORY, L. Moss. **Sistemas digitais**: princípios e aplicações. 10. ed. São Paulo: Prentice Hall, 2007