Introdução a funções lógicas e circuitos lógicos

Última revisão em 12 de março de 2016

Este documento é parte das notas de aula da disciplina MAC0329 (Álgebra Booleana e Circuitos Digitais) e apresenta uma introdução aos conceitos e terminologias importantes no estudo de circuitos digitais.

Para a leitura deste texto, alguma noção sobre representação de dados em computador, especialmente a representação binária de números, é importante. Dente os tópicos relacionados e previamente vistos, citamos:

- sistemas de representação numérica: representações de números em diferentes bases (especialmente a base 2); conversão de representação de uma base para outra;
- representação de números no computador: palavras "binárias" com número fixo de bits (tipicamente 64 ou 32); representação de números sem sinal e com sinal; representação complemento de 2; intervalo dos números que podem ser representados em n bits;
- algoritmo de adição de números binários; adição e subtração de números sem sinal e com sinal (considerando a representação na forma complemento de 2); detecção de overflow nas operações de adição e subtração.

1 Funções lógicas

No estudo de circuitos digitais, comumente deparamos com os termos **função lógica**, **função binária**, **função booleana**, **função de chaveamento**. Todos eles referem-se a funções que mapeiam entradas binárias em saídas binárias. O termo lógica está predominantemente associado aos valores **verdadeiro** e **falso**, o binário aos valores **1** e **0**, e chaveamento aos estados **ligado** e **desligado**. Ao removermos o "aspecto semântico" dessas funções, temos uma estrutura matemática abstrata denominada **álgebra booleana** (nome derivado de George Boole, um matemático inglês), útil no estudo dessas funções. Uma definição formal de álgebra booleana será apresentada em outro documento, mas o conteúdo aqui descrito já utiliza algumas notações algébricas a serem vistas posteriormente.

Consideremos um computador. A menor unidade de armazenamento de informação no computador é o bit (contração de *Binary Digit*), capaz de representar dois estados que

podem ser interpretados como 0 e 1. Assim, computadores adotam a base binária para representar dados. Grupos de k bits são usados para armazenar dados no computador (os computadores populares modernos usam $k = 64$ bits). Portanto, podemos pensar que qualquer processamento de dados no computador é uma transformação de entradas em binário para saídas também em binário.

Transformações de dados podem ser descritas por uma função. No caso de dados binários, estamos considerando funções que mapeiam um certo número n de entradas binárias, que denotaremos $x_1, x_2, \ldots, x_n$, para m saídas binárias $y_1, \ldots, y_m$. Matematicamente temos uma função $f : \{0,1\}^n \to \{0,1\}^m$. Podemos pensar também que $f = (f_1, f_2, \ldots, f_m)$, com $f_i : \{0,1\}^n \to \{0,1\}$ e $y_i = f_i(x_1, x_2, \ldots, x_n)\ \forall i$.

Breve pausa: $\{0,1\}^n$ denota o produto cartesiano de conjuntos. Por exemplo, $\{0,1\}^3$ corresponde ao produto cartesiano $\{0,1\}\times\{0,1\}\times\{0,1\}$. Um elemento de $\{0,1\}\times\{0,1\}\times\{0,1\}$ é uma trinca (a,b,c), tal que $a,b,c \in \{0,1\}$. De forma geral, dados por exemplo dois conjuntos A e B, temos que o produto cartesiano de A e B é o conjunto definido por $A \times B = \{(a,b) : a \in A\ e\ b \in B\}$ (isto é, pares (a,b) tais que o primeiro elemento do par pertence a A e o segundo elemento do par pertence a B).

Pergunta: Se supormos que A contém n_a elementos e B contém n_b elementos, qual é o número de elementos (pares) no conjunto $A \times B$?

Exemplo: um exemplo de transformação, ou processamento, de dados é a adição de dois números. Supondo que o computador é de 8 bits, cada número a ser somado corresponderia a um conjunto de 8 bits. Considerando os dois números a serem somados, teríamos portanto 16 entradas. O resultado da adição deverá ser um número armazenado também em 8 bits. Assim, a função de adição seria da forma $f : \{0,1\}^{16} \to \{0,1\}^8$.

O algoritmo de adição de binários é similar à adição usual de números em base 10 (ou seja, os números são somados coluna a coluna, da direita para a esquerda) com a diferença de que há apenas dois possíveis dígitos (0 e 1). Em cada coluna, pode haver um *carry-in* c_{in} (vai-um que veio da coluna anterior) e um *carry-out* c_{out} (vai-um para a próxima coluna). Usando essas notações, podemos escrever o comportamento da soma referente a uma coluna, conforme a tabela 1.

Na tabela, a e b indicam os dígitos em uma certa coluna dos dois números a serem somados e s o dígito da soma nesta mesma coluna. Pode-se ver que esta tabela define, para todas as possíveis atribuições de valores às variáveis c_{in}, a e b, o valor do bit soma s e do carry-out c_{out}. Isto é, temos duas funções binárias, $s(c_{in}, a, b)$ e $c_{out}(c_{in}, a, b)$, ambas com 3 entradas. Usando a notação anteriormente mencionada, poderíamos também considerar

Entrada			Saída	
c_{in}	a	b	s	c_{out}
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 que define o somador de bits.

que temos uma função $f(c_{in}, a, b) = (\, s(c_{in}, a, b), c_{out}(c_{in}, a, b)\,)$.

A "transformação" de entrada para saída da operação de adição de dois números de 8 bits pode também ser descrita em uma tabela como a acima (porém é melhor não tentar fazer isso, pois o número de linhas da tabela é muito grande. Quantas linhas terá a tabela?) De forma similar, qualquer mapeamento de entradas binárias para saídas binárias pode ser descrita por uma tabela.

Se supormos que qualquer processamento de dados em computadores pode ser descrita por uma função binária, então podemos construir computadores usando dispositivos que implementam essas funções. Vamos então estudar quais seriam esses dispositivos e quais funções podem ser implementadas.

2 Circuitos lógicos

Visando entender um pouco como funções tais como as funções s e c_{out} definidas na tabela 1 poderiam ser implementadas, vamos "escrever" a função s de uma forma especial:

$$\begin{aligned} s(c_{in}, a, b) = 1 \iff\ & (c_{in} = 0\ e\ a = 0\ e\ b = 1)\ ou\ (c_{in} = 0\ e\ a = 1\ e\ b = 0)\ ou \\ & (c_{in} = 1\ e\ a = 0\ e\ b = 0)\ ou\ (c_{in} = 1\ e\ a = 1\ e\ b = 1) \end{aligned}$$

Note que no lado direito da equivalência, há uma expressão que inclui os **conectivos lógicos** E e OU, que podem ser "interpretados" da forma que estamos acostumados. A expressão "enumera" as entradas para as quais a função toma valor 1. Não é preciso muito esforço para verificar que para as demais entradas a função s toma valor 0.

Esses conectivos, que correspondem a operações lógicas básicas, definem funções

lógicas, que podem ser descritas por tabelas conforme apresentadas a seguir, juntamente com a definição de outras funções lógicas básicas.

		Função lógica					
		E	OU	NÃO	XOR	NÃO-E	NÃO-OU
x_1	x_2	$x_1\,x_2$	$x_1 + x_2$	$\overline{x}_1$	$x_1 \oplus x_2$	$\overline{x_1\,x_2}$	$\overline{x_1 + x_2}$
0	0	0	0	1	0	1	1
0	1	0	1	1	1	1	0
1	0	0	1	0	1	1	0
1	1	1	1	0	0	0	0

Tabela 2: Tabela-verdade das funções lógicas básicas.

As colunas x_1, x_2 denotam as entradas e as demais colunas definem funções lógicas. XOR corresponde ao OU EXCLUSIVO. Por exemplo, a terceira coluna define o E lógico que toma valor 1 se e somente se $x_1 = 1$ e $x_2 = 1$, e isto pode ser expresso algebricamente por $x_1\,x_2$. A expressão algébrica das demais funções aparece também no cabeçalho de cada coluna. Note que o símbolo $+$, usado para a operação OR (ou lógico), não é o mesmo da álgebra elementar; na álgebra elementar temos $x + x = 2x$, mas na álgebra booleana temos $x + x = x$.

Tabela-verdade é o nome que se dá à definição de uma função lógica escrita na forma de uma tabela como a tabela acima. Os valores são denotados pelos binários 0 e 1, mas poderiam ser equivalentemente denotados pelos valores lógicos V (verdadeiro) e F (falso).

A expressão da função s, apresentada acima, em termos de conectivos lógicos E e OU, sugere que se tivéssemos dispositivos físicos que implementassem o comportamento desses conectivos, seria possível implementar fisicamente a função s.

No caso de computadores digitais, os dispositivos que implementam as funções lógicas básicas acima são denominados **portas lógicas**. A figura 1 mostra as **portas lógicas** mais comumente usadas. Essas portas recebem sinais de entrada à esquerda e produzem um sinal de saída à direita.

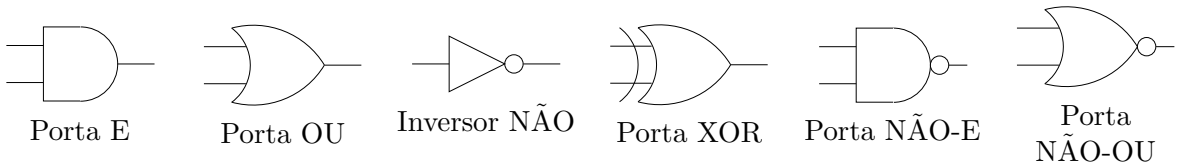


Figura 1: Representação gráfica de algumas portas lógicas.

Ao conectarmos a saída de um dispositivo nas entradas de outros, podemos construir uma rede interconexa de dispositivos. No caso de computadores digitais, a interconexão

define o que é chamado de **circuito digital**. Sem nos atermos à característica física da implementação, podemos usar genericamente o termo **circuito lógico** para nos referirmos a essas redes interconexas.

Seja um circuito, como por exemplo o mostrado na figura 2, que usa três inversores e uma porta E. A saída do circuito, expressa como $f(A, B)$, indica que o valor da saída depende das duas entradas A e B, que podem ser vistas como variáveis da função.

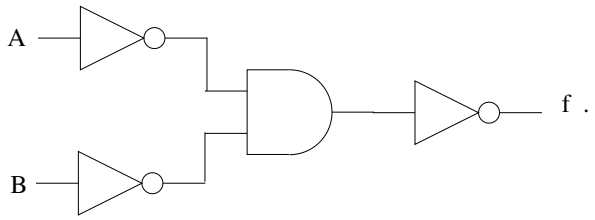


Figura 2: Um circuito simples.

Dizemos que **um circuito realiza uma função**. Podemos descrever seu funcionamento em uma tabela-verdade. Cada linha da tabela corresponde a uma das possíveis atribuições de valor às variáveis de entrada do circuito. No caso do circuito da figura 2, a tabela-verdade é mostrada na tabela 3, juntamente com os valores do circuito em pontos intermediários entre a entrada e a saída. Observe também que uma expressão algébrica que define a função pode ser obtida diretamente do circuito: primeiramente ambas as entradas são negadas e em seguida o resultado do E lógico entre elas é também negado, ou seja, temos $f(A, B) = \overline{\overline{A}\,\overline{B}}$.

A	B	$\overline{A}$	$\overline{B}$	$\overline{A}\,\overline{B}$	$f(A,B) = \overline{\overline{A}\,\overline{B}}$
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1

Tabela 3: Tabela-verdade da função $f(A, B) = \overline{\overline{A}\,\overline{B}}$.

Note que $f(A, B) = \overline{\overline{A}\,\overline{B}} = A + B$. Isto mostra que uma mesma função pode ser realizada por diferentes circuitos. Quando dois circuitos realizam uma mesma função, eles são ditos equivalentes. O circuito da figura 2 é equivalente à porta OR.

Obter a tabela-verdade correspondente a um circuito é uma tarefa mecânica. Obviamente, as tabelas-verdade (e portanto as funções realizadas) por circuitos equivalentes são exatamente iguais. Por outro lado, o problema inverso de desenhar (projetar) um circuito que realiza uma dada função parece ser muito complexa. Porém, há uma forma sistemática que permite desenhar circuitos correspondentes a uma função binária qualquer. Essa forma sistemática baseia-se na mesma ideia da expressão escrita anteriormente para a saída s em termos dos conectivos E e OU, como veremos adiante.

3 Expressões booleanas

Já vimos que uma função lógica pode ser descrita por meio de uma tabela-verdade (apesar dessa representação só ser praticável quando o número de entradas é pequeno ...). Similarmente, a recíproca é verdadeira (isto é, uma tabela-verdade define uma função).

Reforçaremos aqui a ideia de que uma função (ou, equivalentemente, uma tabela-verdade) pode ser expressa por uma expressão, e que expressões por sua vez podem ser "traduzidas" para um circuito lógico. Buscaremos entender como esses elementos se relacionam.

Para tanto, voltemos para o exemplo do somador de bits, e em especial à função s. Por conveniência, abaixo reescrevemos a tabela-verdade do somador de bits e a expressão da saída s apresentada anteriormente:

Entrada			Saída	
c_{in}	a	b	s	c_{out}
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

$$s(c_{in}, a, b) = 1 \iff (c_{in} = 0 \ e \ a = 0 \ e \ b = 1) \ ou \ (c_{in} = 0 \ e \ a = 1 \ e \ b = 0) \ ou \ (c_{in} = 1 \ e \ a = 0 \ e \ b = 0) \ ou \ (c_{in} = 1 \ e \ a = 1 \ e \ b = 1)$$

Qual seria um circuito que realiza a função s ? Note que a expressão s é verdade (i.e., vale 1) se ao menos uma das quatro conjunções ($(c_{in} = 0 \ e \ a = 0 \ e \ b = 1)$, $(c_{in} = 0 \ e \ a = 1 \ e \ b = 0)$, $(c_{in} = 1 \ e \ a = 0 \ e \ b = 0)$ ou $(c_{in} = 1 \ e \ a = 1 \ e \ b = 1)$) for verdade. Em outras palavras, a função que define s é tal que seu valor deve ser 1 para as entradas 011, 101, 110 e 111, e deve ser 0 para as demais entradas.

Observe que, por exemplo, quando $c_{in} = 1$, $a = 1$ e $b = 1$, a expressão $c_{in}\, a\, b$ (produto das três variáveis, ou o E lógico das três variáveis) vale 1. Para qualquer outra atribuição de valor a essas três variáveis, o produto $c_{in}\, a\, b$ toma valor 0. Podemos, portanto, fazer um raciocínio inverso e determinar qual é o produto que toma valor 1 para uma dada entrada específica. Por exemplo, qual é o produto que toma valor 1 quando $c_{in} = 0$, $a = 1$

e $b = 1$? A variável c_{in} precisa aparecer barrada ($\overline{c}_{in}$) no produto (pois caso contrário, a conjunção (E lógico) de c_{in} com qualquer outro valor seria necessariamente 0). O produto que queremos nesse caso é $\overline{c}_{in}\, a\, b$.

Ao fazermos o OU (disjunção) de todos os termos produtos associados a cada entrada que toma valor 1 na tabela, teremos uma forma de expressar a função descrita pela tabela. No caso da função s, temos que:

$$s(c_{in}, a, b) = \overline{c}_{in}\, \overline{a}\, b + \overline{c}_{in}\, a\, \overline{b} + c_{in}\, \overline{a}\, \overline{b} + c_{in}\, a\, b$$

Similarmente, temos que a função c_{out} pode ser escrita como:

$$c_{out}(c_{in}, a, b) = \overline{c}_{in}\, a\, b + c_{in}\, \overline{a}\, b + c_{in}\, a\, \overline{b} + c_{in}\, a\, b$$

Admitindo-se portas lógicas com múltiplas entradas, precisamos então de quatro portas E, uma porta OU e três inversores para implementar a função s, como mostrado na figura 3.

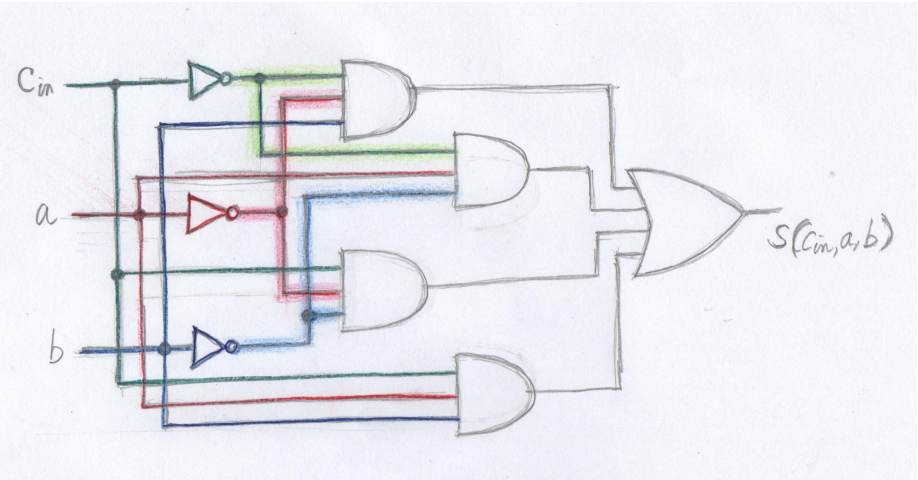
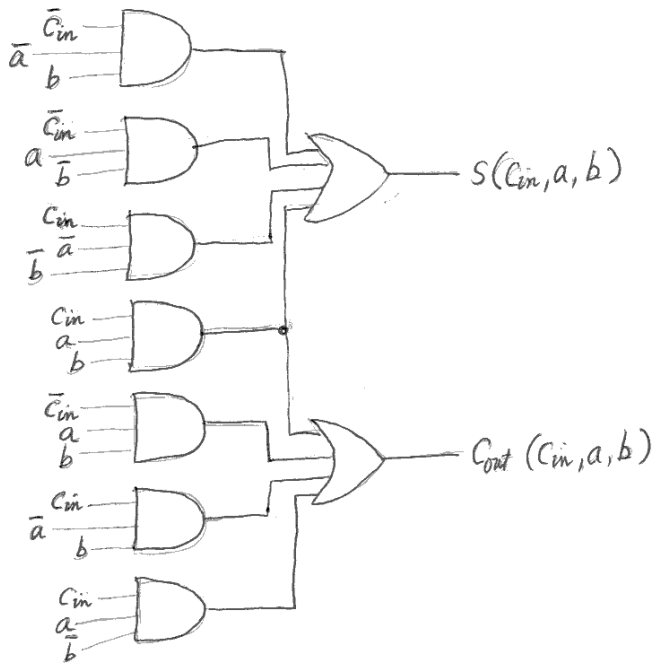


Figura 3: Circuito correspondente à expressão do bit soma $s(c_{in}, a, b) = \overline{c}_{in}\, \overline{a}\, b + \overline{c}_{in}\, a\, \overline{b} + c_{in}\, \overline{a}\, \overline{b} + c_{in}\, a\, b$.

A figura 4 mostra o circuito de ambas as funções, s e c_{out}. Como o último termo é o mesmo em s e em c_{out}, ele é compartilhado. São utilizados portanto 7 portas E (com três entradas cada) e 2 portas OU (com quatro entradas cada). Além disso, seriam necessários inversores, um para cada entrada. Daqui em diante, porém, em vez de desenhar inversores para as entradas barradas, iremos escrever diretamente a variável barrada na entrada do circuito.



$$S(c_{in}, a, b) = \bar{c}_{in}\,\bar{a}\,b + \bar{c}_{in}\,a\,\bar{b} + c_{in}\,\bar{a}\,\bar{b} + c_{in}\,a\,b$$

$$C_{out}(c_{in}, a, b) = \bar{c}_{in}\,a\,b + c_{in}\,\bar{a}\,b + c_{in}\,a\,\bar{b} + c_{in}\,a\,b$$

Figura 4: Circuito somador de bits, implementando as funções s e c_{out} na forma soma de produtos.

4 Circuito somador

Sejam $A = a_3\,a_2\,a_1\,a_0$ e $B = b_3\,b_2\,b_1\,b_0$ dois números binários de 4 bits (onde o subscrito 0 e 3 representam, respectivamente, o bit menos e mais significativo). Podemos projetar um circuito somador, isto é, um circuito que calcula a soma dos dois números, compondo-se somadores de bits em série. Denotando as entradas dos somadores de bits por a_i, b_i e c_i (vai-um da coluna anterior) e as saídas por s_i e c_{i+1}, a ligação em série pode ser feita de forma que a saída vai-um de uma coluna i alimente a entrada vai-um da coluna $i+1$, como mostrado na figura 5.

Cada uma das "caixas" corresponde a um somador de bits, que conforme já discutido acima, trata-se de um circuito de duas saídas (e, portanto, realiza duas funções). Do ponto de vista de função, podemos enxergar o somador de bits como uma função da forma $f : \{0,1\}^3 \to \{0,1\}^2$ (ou duas funções da forma $f : \{0,1\}^3 \to \{0,1\}$).

No diagrama, a entrada c_0, isto é, o *carry-in* na coluna do bit menos significativo,

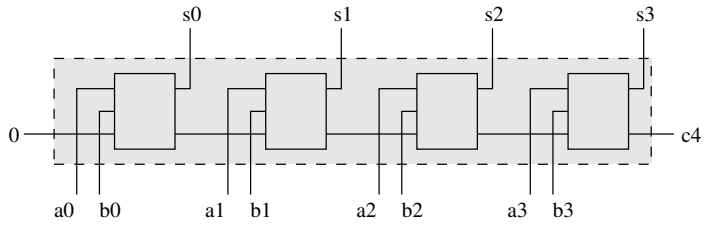


Figura 5: Esquema de um somador de 4 bits.

é zero. Este valor é apropriado para a realização das operações de adição e subtração na representação complemento de dois. No caso da representação sem sinal, a adição pode ser efetuada por esse circuito sem alteração alguma; já a subtração pode ser realizada complementando-se o termo a ser subtraído e fazendo $c_0 = 1$ (complementar e somar 1 é a forma de se obter o negativo de um número na representação complemento de dois). Conforme visto antes, embora o mesmo circuito possa ser usado para as operações de adição e subtração no caso de representação sem sinal e no caso de representação com sinal (na forma complemento de dois), a forma de detecção de *overflow* difere de uma representação para a outra.

Exercício: Para cada uma das representações, escreva uma função f que toma valor 1 no caso de *overflow* e toma valor zero caso contrário. Como poderia ser a implementação dessas funções ?

Na figura 6 é mostrado um outro circuito que também realiza as duas funções s e c_{out}, porém com compartilhamento de subcircuitos.

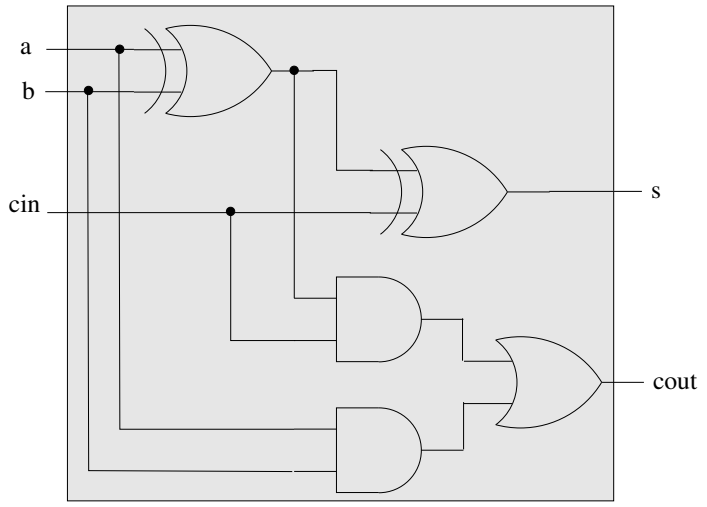


Figura 6: Esquema de um somador de bits, com compartilhamento de subcircuito.

Exercício: Escreva a tabela-verdade correspondente ao circuito da figura 6, incluindo uma coluna para a saída de cada uma das cinco portas lógicas (não apenas para as portas que geram as saídas do circuito). Verifique que as saídas s e c_{out} são conforme esperadas.

A tabela está apresentada a seguir, na figura 7.

c_{in}	a	b	y_1 $a \oplus b$	y_2 $y_1 \cdot c_{in}$	y_3 ab	s	c_{out}
0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0
0	1	0	1	0	0	1	0
0	1	1	0	0	1	0	1
1	0	0	0	0	0	1	0
1	0	1	1	1	0	0	1
1	1	0	1	1	0	0	1
1	1	1	0	0	1	1	1

$$s = y_1 \oplus c_{in} = (a \oplus b) \oplus c_{in}$$

$$c_{out} = y_2 + y_3 = y_1 c_{in} + ab = (a \oplus b) c_{in} + ab$$

Figura 7: Tabela-verdade do circuito da figura 6.

Como podemos ver, obtemos uma tabela igual ao da tabela 1. Isso significa que ambos os circuitos realizam a mesma função e portanto são equivalentes. As expressões de s e c_{out}, obtidas diretamente do circuito, são:

$$s = (a \oplus b) \oplus c$$

$$c_{out} = (a \oplus b)\, c_{in} + ab$$

Logo, é interessante investigar "Como podemos saber se duas expressões são equivalentes". Será necessário escrever a tabela-verdade delas e compará-las ?

Resumo: Neste documento foram introduzidos terminologias e conceitos relacionados a funções lógicas e circuitos lógicos. Considerando computadores como dispositivos que transformam dados binários em dados binários, podemos descrever o comportamento funcional do sistema computacional por meio de funções binárias (lógicas). Uma forma simples de especificação dessas funções são as tabelas-verdade. A partir da tabela-verdade pode-se deduzir uma expressão correspondente à função definida por ela. Conforme visto acima, qualquer expressão escrita em termos dos conectivos lógicos E, OU e a negação NÃO pode ser realizada por meio de um circuito (conectando-se as portas lógicas e inversores conforme a expressão). Similarmente, pode-se facilmente obter a expressão da

função realizada por um circuito. Expressões distintas (e, portanto, circuitos distintos) podem realizar uma mesma função (como é o caso do somador de bits visto acima).

Na sequência, serão consideradas questões como:

- Quais funções podem ser realizadas por um circuito?
- Como verificar se dois circuitos são equivalentes?
- Dada uma função realizável, qual a realização que utiliza o menor número de dispositivos?
- Dado um circuito, existe circuito equivalente e mais simples?
- Se nos restringirmos ao uso de determinados tipos de porta apenas (e não todos), quais tipos de funções são realizáveis?
- Como otimizar o compartilhamento de subcircuitos para a realização de várias funções ?

Não percam o próximo capítulo !