Capítulo 2

Portas lógicas I

Uma das bases da técnica de projeto digital é a percepção de que qualquer informação ou valor pode ser representado de forma binária, seja por meio de zeros e uns, de falsos e verdadeiros ou de ligados e desligados. Tudo do mundo externo ao circuito que vamos representar dentro do nosso projeto, usando as técnicas de projeto digital, será feito com valores binários. Isso se aplica mesmo a valores de variáveis tipicamente contínuas, como temperatura, pressão etc. — com as quais temos de usar as técnicas de conversão analógica para digital e conversão digital para analógica —; ainda assim, estamos falando de valores e informações que são representados usando valores binários.

Neste capítulo, vamos começar a entender como podemos manipular essas informações e esses valores para que eles sejam úteis dentro de um produto que foi feito com as técnicas de projeto digital. Vamos iniciar com as operações mais básicas, que são os blocos fundamentais para a construção física de qualquer dispositivo que utiliza as técnicas de projeto digital.

Qualquer operação para tratar os valores representados em binário, do ponto de vista físico, será implementada por meio dos circuitos mais básicos, que vamos descrever neste capítulo — e isso inclui todas as operações que qualquer software pode fazer. Por exemplo, para que um computador possa fazer a soma de dois números, a operação matemática será feita por um circuito digital, que é composto pelos circuitos básicos descritos a seguir.

1 Portas lógicas AND, OR, NOT

Na técnica de projeto digital, as portas lógicas são definidas como um circuito eletrônico que tem por função executar operações lógicas.

Atualmente, essas funções lógicas são tipicamente implementadas usando-se transistores em circuitos integrados, e podemos encontrá-los sob diversos tipos de encapsulamento. Podemos encontrar chips que implementam apenas algumas poucas portas lógicas, que são acessíveis diretamente pelos terminais, ou chips com muitos milhões de portas lógicas, que podem se apresentar na forma de processadores, microcontroladores, chips lógicos programáveis em campo etc.

Entretanto, esses circuitos podem ser implementados de diversas formas, inclusive com o uso de tecnologias que estão disponíveis desde a década de 1930; nesse caso, estamos falando dos relês como elemento ativo.

Um relê é um dispositivo eletromecânico composto por uma alavanca metálica com uma mola e um eletroímã. A alavanca metálica liga dois (ou mais) contatos elétricos, que podem ser conectados ou desconectados pela ação do eletroímã. Para que isso fique mais claro, vamos explicar como funciona um relê normalmente fechado. Quando não há corrente elétrica passando pelo eletroímã, os contatos elétricos desse dispositivo ficam conectados, ou seja, trata-se de uma chave elétrica fechada; por isso, como o normal do relê é operar sem corrente elétrica na bobina, ele é um dispositivo normalmente fechado. No entanto, quando passamos corrente elétrica pela bobina do relê, a alavanca é acionada e o contato elétrico é desfeito, ou seja, a chave elétrica do nosso relê se torna aberta.

Em suma, um relê normalmente aberto opera no inverso de um relê normalmente fechado, ou seja, os contatos do relê se encontram na forma de uma chave elétrica aberta quando não há corrente elétrica passando pela bobina, e os contatos são fechados quando há corrente elétrica pela bobina.

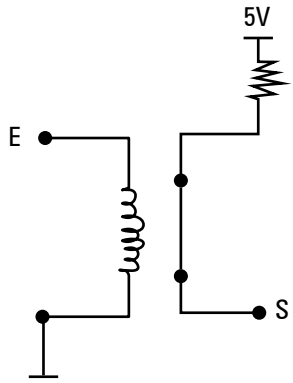
Fundamentalmente, qualquer circuito eletrônico que implemente uma das funções lógicas básicas pode ser considerado uma porta lógica, não importa como tenha sido construído do ponto de vista físico. Isso tornou a técnica de projeto digital perene em meio à tremenda evolução tecnológica das últimas décadas.

IMPORTANTE

Por mais que a tecnologia de fabricação dos dispositivos tenha evoluído, as equações e a matemática que eles implementam são as mesmas.

Para ilustrar isso de uma forma bastante simples, na figura 1, a seguir, temos uma porta NOT implementada com um relê normalmente fechado.

Figura 1 — Porta lógica a relê



Analisando esse circuito, identificamos o terminal E como entrada e o terminal S como saída. Dado que a tensão para o acionamento da bobina é de 5 V, quando esse nível de tensão for aplicado no terminal E, a chave interna do relê será aberta, e o terminal S ficará sem alimentação da fonte, o que vai levar a sua tensão para 0 V (em uma aplicação real, o terminal S estaria ligado na bobina de outro relê, o que levaria o terminal para 0 V). Se aplicarmos uma tensão de 0 V na entrada E do circuito, a bobina deixará de ser acionada, e a tensão no terminal S será de 5 V.

Isso nos leva à nossa primeira tabela-verdade, expressa na tabela 1, a seguir.

Tabela 1 – Tabela-verdade 1

E	S
5 V	0 V
0 V	5 V

Entretanto, para simplificar e padronizar a notação dos projetos digitais, se convencionou que os símbolos a serem usados são 0 e 1, ou falso e verdadeiro, respectivamente.

Assim, é comum considerarmos que 0 V corresponde a falso (ou desligado) e 5 V corresponde a verdadeiro (ou ligado), porém, como já exposto no capítulo anterior, cada tecnologia de fabricação adota uma padronização diferente de tensões para representar 0 e 1 (falso e verdadeiro).

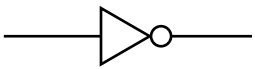
Remodelando nossa tabela-verdade para utilizar a simbologia tradicional, chegamos à tabela 2.

Tabela 2 – Tabela-verdade 2

E	S
1	0
0	1

A tabela 2 é a primeira forma de representação da função lógica de negação (NOT). Além dessa tabela-verdade, podemos representar a porta lógica NOT pelo símbolo usado em esquemas elétricos, apresentado na figura 2.

Figura 2 – Porta NOT (negação)



Outra forma de representar a função lógica da negação é por meio da representação algébrica:

$$S = \overline{E}$$

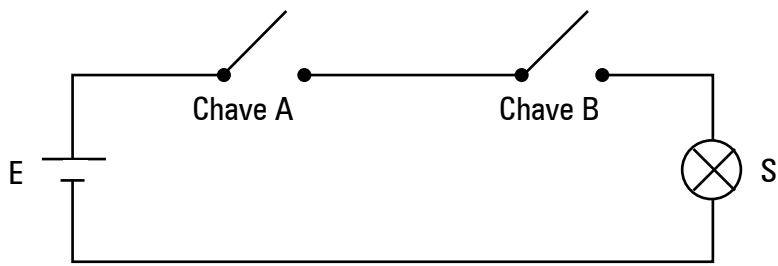
Note que, nessa representação, a negação é simbolizada por uma linha sobre a expressão que ela está negando. Mais adiante, vai ficar

mais claro como podemos usar essa notação para fazer as simplificações das expressões lógicas.

A segunda porta lógica básica que vamos estudar é a porta lógica AND ("e"). Podemos descrever a função lógica AND como uma multiplicação de duas ou mais variáveis booleanas, em que todas as entradas da porta lógica devem estar acionadas para que a saída seja verdadeira.

O circuito mais simples que podemos usar para representar a função lógica AND seria o de chaves em série, em que as chaves são as entradas. Quando a chave está fechada, ela representa 1 (verdadeiro), e quando está aberta, representa 0 (falso). Já a saída pode ser representada pela presença ou pela ausência de tensão para acionar uma lâmpada, por exemplo, como na figura 3, a seguir.

Figura 3 – Implementando uma porta AND



No circuito da figura 3, podemos notar que a lâmpada só ficará acesa se a chave A e a chave B estiverem acionadas (fechadas), ou seja, em estado verdadeiro.

Podemos descrever esse circuito por meio de uma tabela-verdade, conforme expresso na tabela 3.

Tabela 3 – Tabela-verdade 3

ENTRADAS		SAÍDA
Chave A	**Chave B**	**S**
Aberta	Aberta	Apagada
Aberta	Fechada	Apagada
Fechada	Aberta	Apagada
Fechada	Fechada	Acesa

Se adotarmos que fechado corresponde a 1 (verdadeiro) e aberto corresponde a 0 (falso), bem como que o estado "aceso" equivale a verdadeiro e "apagado", a falso, podemos refazer a tabela de forma mais padronizada, conforme a tabela 4.

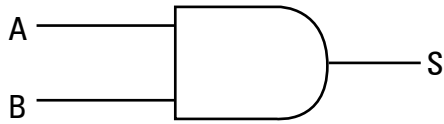
Tabela 4 – Tabela-verdade 4

ENTRADAS		SAÍDA
Chave A	**Chave B**	**S**
0	0	0
0	1	0
1	0	0
1	1	1

Ou seja, a função lógica AND só gera uma saída verdadeira quando todas as entradas são verdadeiras, independentemente do número de entradas da porta.

A forma gráfica de representar a função lógica AND de duas entradas é a da figura 4, a seguir.

Figura 4 – Representação gráfica de uma porta AND



O número de entradas, em termos teóricos, é ilimitado, porém, na prática, existe um limite, em parte por questões econômicas e em parte em decorrência da tecnologia que se usa para implementar a porta lógica. Entretanto, se uma função exigir muitas entradas, pode-se facilmente combinar várias portas AND menores, que são equivalentes a uma maior. Esse tópico será aprofundado mais adiante.

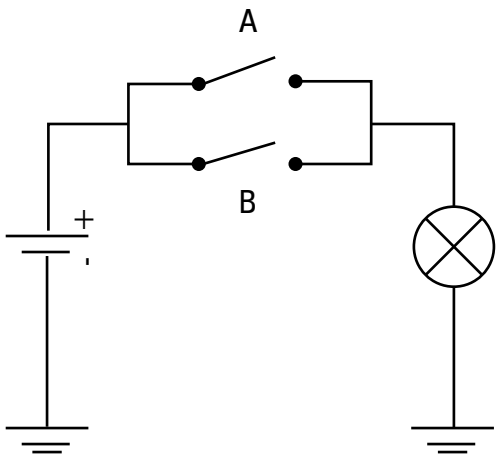
Do ponto de vista de expressão algébrica, a operação AND é representada por um ponto (.) entre duas variáveis booleanas, como exemplificado na expressão a seguir:

$$S = A \,.\, B$$

A terceira porta lógica que vamos analisar é a porta lógica OR ("ou"). Podemos descrevê-la como uma soma de duas ou mais variáveis booleanas, em que, se qualquer uma das entradas for verdadeira, a saída será verdadeira.

Como forma de ilustrar essa função lógica, vamos analisar o circuito da figura 5.

Figura 5 – Construindo uma porta OR



No circuito da figura 5, podemos notar que a lâmpada ficará acesa se qualquer uma das chaves estiver acionada. Ou seja, se a chave A ou a chave B estiverem acionadas (fechadas), a lâmpada ficará acesa.

Podemos descrever esse circuito por meio de uma tabela-verdade, conforme a tabela 5, a seguir.

Tabela 5 – Tabela-verdade 5

ENTRADAS		SAÍDA
Chave A	**Chave B**	**S**
Aberta	Aberta	Apagada
Aberta	Fechada	Acesa
Fechada	Aberta	Acesa
Fechada	Fechada	Acesa

Se adotarmos que fechado corresponde a 1 (verdadeiro) e aberto corresponde a 0 (falso), bem como que o estado "aceso" é equivalente a verdadeiro e "apagado", a falso, podemos refazer a tabela de uma forma mais padronizada, conforme a tabela 6.

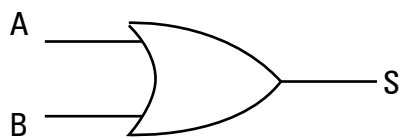
Tabela 6 – Tabela-verdade 6

ENTRADAS		SAÍDA
Chave A	**Chave B**	**S**
0	0	0
0	1	1
1	0	1
1	1	1

Ou seja, a função lógica OR gera uma saída verdadeira quando qualquer uma das entradas estiver verdadeira, independentemente do número de entradas da porta.

A forma gráfica de representar a função lógica OR de duas entradas é a da figura 6, a seguir.

Figura 6 – Representação gráfica de uma porta OR



Novamente, o número de entradas, em termos teóricos, é ilimitado, porém, na prática, os limites são os mesmos observados para a porta AND. Além disso, similarmente, se uma função exigir muitas entradas,

pode-se facilmente combinar várias portas OR menores, que são equivalentes a uma maior.

Do ponto de vista da expressão algébrica, a operação OR é representada por um sinal de soma + entre duas variáveis booleanas, como exemplificado na expressão:

$$S = A + B$$

2 Tabelas-verdade

Como você já deve ter percebido, uma das formas de descrever as portas lógicas simples é por meio de suas tabelas-verdade. As tabelas-verdade, entretanto, podem descrever quaisquer circuitos digitais combinacionais. Elas consistem em uma coluna que simboliza a saída do circuito combinacional e outra coluna que remete a cada variável booleana do circuito.

O número de linhas desse tipo de tabela depende do número de combinações possíveis das variáveis booleanas de entrada. Assim, se tivermos um circuito com três entradas, teremos 2^3 combinações, e se tivermos quatro entradas, teremos 2^4 combinações.

O procedimento básico para usar as tabelas-verdade consiste em gerar uma tabela com todas as combinações possíveis de entradas e, em seguida, anotar, para cada combinação, se o valor da saída é falso ou verdadeiro.

Um primeiro exemplo de circuito com três entradas é mostrado na tabela 7.

Tabela 7 – Tabela-verdade 7

A	B	C	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Na tabela-verdade apresentada, podemos ler que a saída será verdadeira em quatro situações:

- quando A = 0, B = 0 e C = 1;
- quando A = 0, B = 1 e C = 0;
- quando A = 1, B = 0 e C = 0; e
- quando A = 1, B = 0 e C = 1.

No próximo subcapítulo, vamos explorar como transformamos a tabela-verdade em expressões lógicas e em circuitos lógicos.

Quanto à transformação de uma lista de requisitos de funcionalidade em um circuito lógico, uma das formas mais fáceis de fazê-lo é por meio de tabelas-verdade. Basicamente, a cada linha da lista de requisitos, anotamos se a saída do circuito combinatório deve ser falsa ou verdadeira.

Como exemplo, vamos analisar o texto a seguir, que é uma maneira informal de descrever o que queremos que um circuito combinatório que vamos projetar faça.

> "Estamos projetando um forno de micro-ondas e precisamos de um circuito que proteja o proprietário de acidentes.
>
> Se o proprietário colocasse a mão dentro do forno ligado, haveria um acidente, portanto, o forno não pode ligar se a porta estiver aberta. Além disso, precisamos que a luz interna fique acesa quando a porta estiver aberta. Também precisamos que a lâmpada fique ligada quando o forno estiver ligado, e que ela fique desligada se a porta estiver fechada e o forno, desligado.
>
> Quando o proprietário pressionar o botão 'ligar', se a porta estiver fechada, o forno vai ligar. Quando o proprietário pressionar 'ligar', se a porta estiver aberta, o forno não vai ligar."

Nesse texto, fizemos uma descrição parcial do projeto de um forno de micro-ondas. Agora, vamos transformar o texto em variáveis booleanas, começando pelas entradas:

- temos o botão para acionar o forno, que vamos chamar de A;
- por simplicidade, vamos adotar que o botão A, quando pressionado, fica ligado; para ser desligado, ele precisa ser pressionando para o outro lado;
- vamos adotar que, quando quisermos o forno ligado, a variável A estará em verdadeiro (1); quando quisermos o forno desligado, a variável estará em falso (0);
- temos o sensor da porta, que vamos chamar de P; quando a porta estiver aberta, estará em verdadeiro, e quando estiver fechada, estará em falso.

Agora, vamos definir as saídas:

- a primeira saída é a lâmpada, que vamos chamar de L; quando L estiver em verdadeiro, a lâmpada estará ligada, e quando estiver em falso, a lâmpada estará desligada;
- a segunda saída é o magnétron – o elemento que aquece os alimentos dentro do forno –, que vamos chamar de M; quando M estiver verdadeiro, o forno estará aquecendo, e quando M estiver em falso, o forno estará desligado.

Como temos duas variáveis de entrada e quatro possibilidades de entrada, isso vai nos dar a tabela 8, a seguir.

Tabela 8 – Tabela-verdade 8

A	P	L	M
Não acionado	Fechado	Desligado	Desligado
Não acionado	Aberto	Ligado	Desligado
Acionado	Fechado	Ligado	Ligado
Acionado	Aberto	Ligado	Desligado

Observe que a tabela consegue expressar, de forma muito resumida, o que precisamos que o circuito faça.

Como temos duas saídas, podemos fazer duas tabelas separadas para cada saída, ou podemos resumir as informações em uma única tabela, a fim de facilitar o entendimento.

IMPORTANTE

Muitas vezes, é possível aproveitar circuitos lógicos em mais de uma saída.

Trocando as palavras pelos significados que demos para elas, em termos de verdadeiro e falso, temos a tabela 9.

Tabela 9 – Tabela-verdade 9

A	P	L	M
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	0

3 Expressões booleanas

Podemos entender as expressões booleanas como uma forma algébrica de representar os circuitos digitais. Desse modo, temos uma ferramenta para manipular e otimizar os circuitos.

As operações básicas da álgebra booleana são fundamentalmente as mesmas que as portas lógicas já apresentadas fazem, ou seja, as das operações lógicas AND, OR e NOT.

Como já destacamos anteriormente, as operações lógicas podem ser representadas por:

- AND ⟶ A AND B = A . B = B . A = B AND A
- OR ⟶ A OR B = A + B = B + A = B OR A
- NOT ⟶ NOT A = A = $\overline{A}$

Assim, temos:

$0 . 0 = 0$

$1 . 1 = 1$

$0 . 1 = 1 . 0 = 0$

$0 + 0 = 0$

$1 + 1 = 1$

$0 + 1 = 1 + 0 = 1$

Não $1 = 1' = \overline{1} = 0$

Não $0 = 0' = \overline{0} = 1$

Isso nos leva aos principais postulados de Boole (IDOETA; CAPUANO, 1999), conforme descrição a seguir.

Considere X, Y e Z variáveis lógicas distintas.

$0*X = 0$

$1*X = X$

$X*X = X$

$\overline{X}*X = 0$

$0 + X = X$

$1 + X = 1$

$X + X = X$

$\overline{X} + X = 1$

$\overline{\overline{X}} = X$

Comutativas:

$X + Y = Y + X$

$X*Y = Y*X$

Associativas:

$$X + (Y + Z) = (X + Y) + Z$$

$$X*(Y*Z) = (X*Y)*Z$$

Distributivas:

$$X*(Y + Z) = (X*Y) + (X*Z)$$

Veja que esses postulados são bastante semelhantes à álgebra tradicional, porém, têm suas particularidades.

Além desses postulados, é muito importante ter em mente e treinar o uso dos dois teoremas de De Morgan (IDOETA; CAPUANO, 1999):

- 1° Teorema: $\overline{A} + \overline{B} = \overline{A \cdot B}$
- 2° Teorema: $\overline{A} \cdot \overline{B} = \overline{A + B}$

Podemos demonstrá-los por meio da tabela 10, a seguir.

Tabela 10 – Tabela-verdade 10

1° TEOREMA			
A	**B**	**Termo 1**	**Termo 2**
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

2° TEOREMA			
A	**B**	**Termo 1**	**Termo 2**
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

3.1 Usando as expressões booleanas

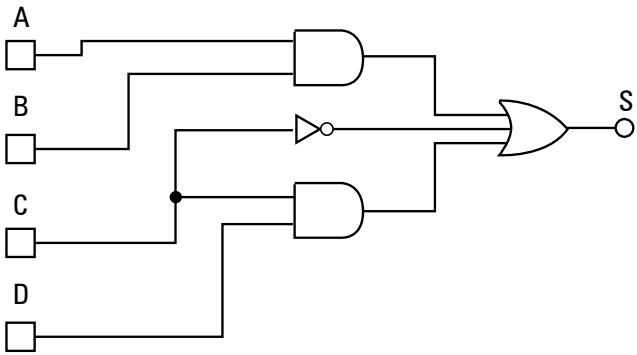
O uso das expressões booleanas exige treino e certamente não é algo simples à primeira vista. Por isso, vamos usar as expressões e

suas representações em termos de circuitos lógicos, assim conseguiremos mostrar o uso real dessa ferramenta de projeto digital.

O primeiro exemplo pode ser conferido na figura 7, cuja equação algébrica é:

$$S = (A \,.\, B) + \overline{C} + (C \,.\, D)$$

Figura 7 – Primeiro exemplo de expressão booleana



3.2 Criando expressões booleanas a partir de tabelas-verdade

Até este ponto do capítulo, trabalhamos para gerar as tabelas-verdade a partir das expressões booleanas, porém, em geral, o que ocorre é o processo inverso: a partir da descrição do problema a ser resolvido geramos a tabela-verdade, e da tabela-verdade geramos as expressões booleanas.

O processo de conversão, entretanto, é bastante simples, embora possa ser trabalhoso, dependendo do número de variáveis de entrada que o circuito possui.

Basicamente, o processo consiste, primeiro, em escolher se vamos trabalhar com lógica positiva ou lógica negativa.

Essa escolha se dá pela observação, por parte da pessoa que está fazendo o projeto, da quantidade de combinações em que a saída é verdadeira e da quantidade de combinações em que a saída é falsa.

Uma vez feita a escolha, identificamos os casos em que apenas uma a situação escolhida ocorre, e escrevemos a expressão lógica correspondente àquela combinação.

Como exemplo, vamos analisar a tabela 11.

Tabela 11 – Tabela-verdade 11

LINHA	A	B	S
0	0	0	1
1	0	1	0
2	1	0	1
3	1	1	1

Para fins didáticos, vamos analisar os casos em que a saída é verdadeira, embora seja muito mais vantajoso fazer o processo com a situação em que a saída é falsa.

A primeira situação em que a saída é verdadeira está na linha 0, cuja expressão lógica seria:

$$S = \overline{A} \cdot \overline{B}$$

A segunda situação ocorre na linha 2, que nos dá a expressão:

$$S = A \cdot \overline{B}$$

E a última situação ocorre na linha 3:

$$S = A \cdot B$$

Se fizermos um OU entre as três expressões obtidas até agora, vamos obter a expressão que descreve a tabela 11:

$$S = (\overline{A} \,.\, \overline{B}) + (A \,.\, \overline{B}) + (A \,.\, B)$$

Podemos notar que, independentemente do número de entradas, é possível obter as expressões booleanas que descrevem a tabela-verdade, e manipulando as expressões podemos simplificar qualquer circuito ao seu mínimo para a implementação.

Considerações finais

Neste capítulo, conhecemos as portas lógicas básicas, detalhamos o comportamento de cada uma delas – expressando-os na forma de equações – e começamos a entender como podemos combinar as portas lógicas para que elas tenham o comportamento desejado. Conferimos, também, que a matemática por trás dessas portas lógicas não depende de como elas estão sendo implementadas, e que essa mesma matemática pode ser utilizada com circuitos baseados em transistores, relês ou outras formas de implementação.

Referências

IDOETA, Ivan V.; CAPUANO, Francisco G. **Elementos de eletrônica digital**. São Paulo: Érica, 1999.