Organização de Computadores
Prof. Robson de Souza

Nível Lógico Digital

Álgebra Booleana

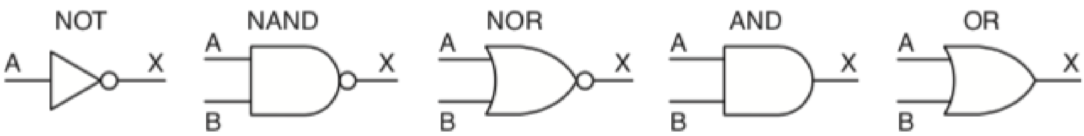
O Nível Lógico Digital se refere ao real hardware do computador. A partir daqui serão examinados alguns aspectos da lógica digital, como um fundamento para o estudo de níveis mais altos da organização de computadores. Esse assunto está no limiar entre a ciência da computação e a engenharia elétrica, mas o material é independente. Algo muito interessante que é possível perceber ao estudar essa área é que os elementos básicos que fazem parte de todos os computadores digitais são surpreendentemente simples.

Portas e álgebra booleana

Circuitos digitais podem ser construídos com um pequeno número de elementos primitivos combinando-os de inúmeras maneiras. Um circuito digital é aquele em que estão presentes somente dois valores lógicos. O normal é que um sinal entre 0 e 0,5 volt represente um valor (por exemplo, 0 binário) e um sinal entre 1 e 1,5 volt represente o outro valor (por exemplo, 1 binário). Minúsculos dispositivos eletrônicos, denominados **portas (gates)**, podem calcular várias funções desses sinais de dois valores. Essas portas formam a base do hardware sobre a qual todos os computadores digitais são construídos.

Os detalhes do funcionamento interno das portas pertencem ao nível de dispositivo, que está abaixo do nível 0. É possível montar as portas utilizando um conjunto de transistores por exemplo.
De todas as portas lógicas possíveis, existem cinco que são os principais elementos de construção do nível lógico digital, que são as portas NOT, NAND, NOR, AND e OR.

As portas NAND e NOR requerem dois transistores cada, ao passo que as portas AND e OR requerem três cada. Por essa razão, muitos computadores são baseados em portas NAND e NOR em vez das portas mais conhecidas, AND e OR. Vale a pena observar que as portas podem perfeitamente ter mais de duas entradas. A figura abaixo mostra a representação gráfica dessas portas, com o seu conjunto de entradas e saídas.



A	X
0	1
1	0

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

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

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

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

Fonte: (Tanenbaum e Austin, 2013)

Para descrever os circuitos que podem ser construídos combinando portas, é necessário um novo tipo de álgebra, no qual variáveis e funções podem assumir somente os valores 0 e 1. Essa álgebra é denominada álgebra booleana, nome que se deve a seu descobridor, o matemático inglês George Boole (1815–1864).

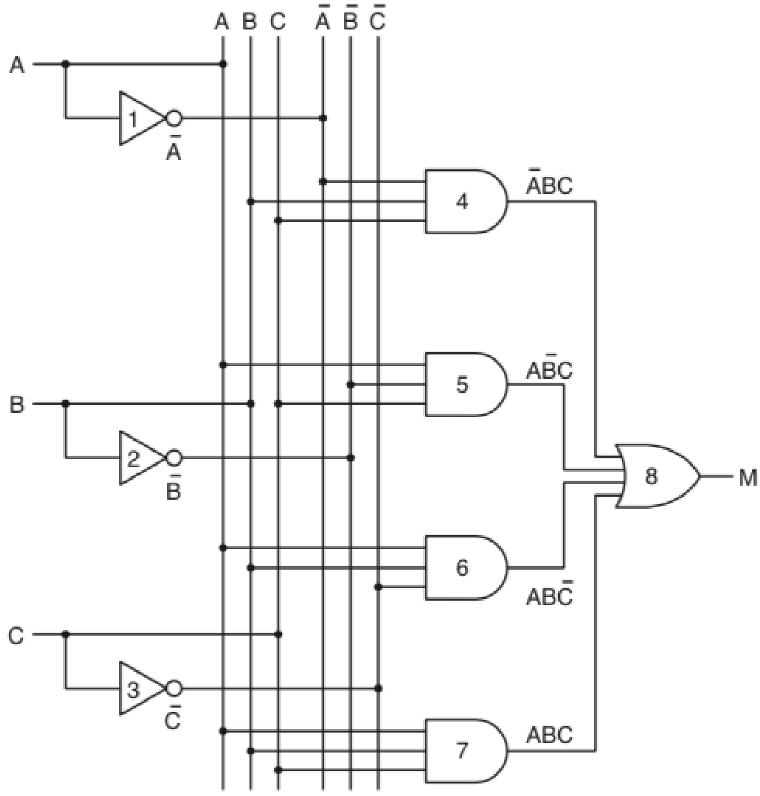
Assim como há funções na álgebra "ordinária", também há funções na álgebra booleana. Uma função booleana tem uma ou mais variáveis de entrada e produz um resultado que depende somente dos valores dessas variáveis. Uma função simples, f, pode ser definida ao se dizer que f(A) é 1 se A for 0 e f(A) é 0 se A for 1. Essa função é a função not da Figura acima.

Como uma função booleana de n variáveis só tem 2^n combinações possíveis de valores de entrada, ela pode ser completamente descrita por uma tabela com 2^n linhas, na qual cada linha informa o valor da função para uma combinação diferente de valores de entrada. Ela é denominada **tabela verdade**.

A figura abaixo mostra a tabela verdade para uma função booleana de três variáveis: M = f(A, B, C). Essa função é a de lógica majoritária, isto é, ela é 0 se a maioria de suas entradas for 0, e 1 se a maioria de suas entradas for 1. Embora qualquer função booleana possa ser completamente especificada dada sua tabela verdade, à medida que aumenta o número de variáveis, essa notação fica cada vez mais trabalhosa. Portanto, costuma-se usar outra notação no lugar dela.

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(a)



(b)

Fonte: (Tanenbaum e Austin, 2013)

Para ver como ocorre essa outra notação, observe que qualquer função booleana pode ser especificada ao se dizer quais combinações de variáveis de entrada dão um valor de saída igual a 1. Para a função da figura acima, há quatro combinações de variáveis de entrada que fazem com que M seja 1. Por convenção, marcaremos a variável de entrada com uma barra para indicar que seu valor é invertido. A ausência de uma barra significa que o valor não é invertido.

Além disso, usaremos a multiplicação implícita ou um ponto para representar a função booleana AND e + (sinal de adição) para representar a função booleana OR. Assim, por exemplo, $A\overline{B}C$ assume o valor 1 somente quando A = 1 e B = 0 e C = 1. Além disso, $A\overline{B} + B\overline{C}$ é 1 somente quando (A = 1 e B = 0) ou (B = 1 e C = 0).

As quatro linhas da figura acima que produzem bits 1 na saída são: $\overline{A}BC$, $A\overline{B}C$, $AB\overline{C}$ e ABC. A função, M, é verdadeira (isto é, 1) se qualquer uma dessas quatro condições for verdadeira; daí, podemos escrever

$$M = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

como um modo compacto de dar a tabela verdade. Essa formulação é de especial importância, pois leva diretamente a uma execução da função que usa portas padronizadas.
É importante ter em mente a distinção entre uma função booleana abstrata e sua execução por um circuito

eletrônico. Uma função booleana consiste em variáveis, como A, B e C, e operadores booleanos, como AND, OR e NOT.

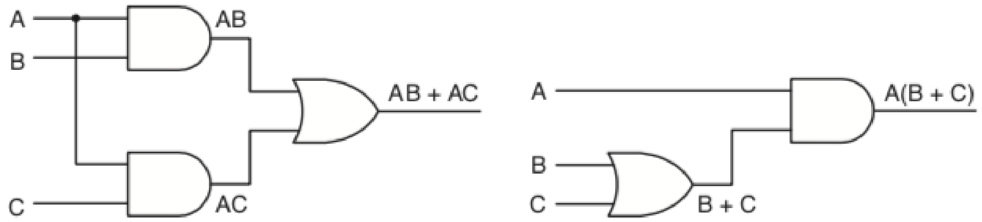
Uma função booleana pode ser executada por um circuito eletrônico (muitas vezes de vários modos diferentes) usando sinais que representam as variáveis de entrada e saída e portas como AND, OR e NOT.

<u>* Equivalência de circuito</u>

Projetistas de circuitos muitas vezes tentam reduzir o número de portas em seus produtos para reduzir a área da placa de circuito interno necessária para executá-las, diminuir o consumo de potência e aumentar a velocidade. Para reduzir a complexidade de um circuito, o projetista tem de encontrar outro circuito que calcule a mesma função que o original, mas efetue essa operação com um número menor de portas (ou talvez com portas mais simples, por exemplo, com duas em vez de com quatro entradas).

Como exemplo de quão útil a álgebra booleana pode ser usada para essa finalidade, considere o circuito e a tabela verdade para AB + AC mostrados na figura abaixo parte (a). Embora ainda não as tenhamos discutido, muitas das regras da álgebra comum também são válidas para a booleana.

Em particular, a expressão AB + AC pode ser fatorada para A(B + C) usando a lei distributiva. A figura abaixo parte (b) mostra o circuito e a tabela verdade para A(B + C). Como duas funções são equivalentes **se, e somente se**, elas tiverem a mesma saída para **todas** as entradas possíveis, é fácil ver pelas tabelas verdade da figura abaixo que A(B + C) é equivalente a AB + AC.



A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

(a)

A	B	C	A	B + C	A(B + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

(b)

Fonte: (Tanenbaum e Austin, 2013)

Em geral, um projetista de circuitos começa com uma função booleana e depois aplica a ela as leis da álgebra booleana na tentativa de achar uma função mais simples, porém equivalente. Um circuito pode ser construído com base na função final.

Referências bibliográficas:

TANENBAUM, Andrew S. Organização Estruturada de Computadores, 2007, 5ª Edição.

TANENBAUM, Andrew S. AUSTIN, Todd; Organização Estruturada de Computadores, 2013, 6ª Edição.