Sistemas digitais

Hugo Tanzarella Teixeira
Marley Fagundes Tavares
Rodrigo Vinícius Mendonça Pereira

Dados Internacionais de Catalogação na Publicação (CIP)

Teixeira, Hugo Tanzarella
T266s Sistemas digitais / Hugo Tanzarella Teixeira, Marley Fagundes Tavares, Rodrigo Vinícius Mendonça Pereira. – Londrina : Editora e Distribuidora Educacional S.A. 2017.
184 p.

ISBN 978-85-522-0299-8

1. Computadores digitais. I. Tavares, Marley Fagundes. II. Pereira, Rodrigo Vinicius Mendonça. III. Título.

CDD 004

2017
Editora e Distribuidora Educacional S.A.
Avenida Paris, 675 – Parque Residencial João Piza
CEP: 86041-100 — Londrina — PR
e-mail: editora.educacional@kroton.com.br
Homepage: http://www.kroton.com.br/

Sumário

Palavras do autor

Caro aluno, estamos vivendo momentos significativos na história da humanidade. Se você nasceu entre a década de 1980 e 1990, pertence a geração **Y**, também conhecida como geração do **milênio**. Essa geração foi a primeira a nascer necessariamente em um mundo completamente tecnológico. Esse avanço somente foi possível devido aos significativos avanços da ciência, em especial aos avanços na área de microeletrônica. Ano após ano, observamos saltos tecnológicos em dispositivos como celulares, tablets ou computadores pessoais, tanto em capacidade de processamento quanto nas aplicações que funcionam nesses sistemas.

Dessa forma, este livro traz uma análise básica, porém contemporânea, de um tema cada vez mais relevante: sistemas digitais.

O conteúdo aqui exposto guia você para um entendimento aprofundado sobre eletrônica digital e seus conceitos básicos; aspectos do sistema de numeração utilizado em sistemas computacionais e suas operações e codificações; uma visão geral sobre funções lógicas e portas lógicas; e detalhes sobre álgebra booleana e sua simplificação lógica. Além disso, serão abordados assuntos fundamentais, como o diagrama de Veitch-Karnaugh; flip-flops; contadores; e máquina de estados finitos.

Todo esse estudo possibilitará que você compreenda pontualmente detalhes sobre os sistemas digitais. Isso vai permitir que você ganhe confiança para tratar desse assunto com seus colegas. Saiba que o tema é relevante, atual e ao mesmo tempo instigante. Dessa forma, não se limite a este livro, mas extrapole procurando outras fontes de informação e, principalmente, experimentando as abordagens aqui propostas. No momento em que você está cursando essa disciplina é evidente o chamado ***boom do hardware***, um movimento caracterizado por grandes empresas que inundam o mercado com sistemas completos para desenvolvedores extrapolarem a imaginação. Existe um investimento significativo por parte das grandes empresas em ambientes de programação gratuitos, com diversas bibliotecas gratuitas e dispositivos dos mais variados preços, processamento e aplicação. Nós, brasileiros, somos um povo criativo. Já embarcamos nessa nova onda, e com boas ideias. De leitura fácil e instigante, o presente texto é um convite ao estudo, sem deixar de lado o contexto atual e a criatividade da juventude. Bem-vindo ao mundo dos sistemas digitais!

Introdução aos sistemas digitais

Convite ao estudo

Estimado aluno, a partir de agora, começaremos a conhecer os circuitos digitais. Inicialmente, em linhas gerais, definiremos os conceitos de circuitos digitais, ou seja, a construção desse conhecimento se fundamenta em alguns conceitos básicos dos circuitos digitais. Em seguida, entenderemos os sistemas numéricos e sua importância para os sistemas digitais. Por fim, falaremos sobre as portas lógicas e sua aplicação.

Todo esse conhecimento justifica-se se observarmos o nosso dia a dia. No momento em que você lê este texto, bilhões de pessoas estão interagindo com algum circuito digital, seja ligando o celular, seja usando um relógio ou assistindo a um vídeo na internet. A quantidade de informação gerada somente na internet rompeu a casa dos **zettabytes** (equivalente a um **sextilhão** de bytes). Portanto, circuitos digitais e toda a eletrônica envolvida nesse volume de informação faz parte da revolução dos bytes nos últimos cem anos. Você já pensou que faz parte dessa revolução? E que você vive um momento da história no qual não existem mais barreiras entre idiomas, culturas e conhecimento? Todo esse cenário só foi possível porque milhões de cientistas levaram a ciência a um novo patamar, criando sistemas e incrementando as tecnologias de forma a facilitar o dia a dia do homem. Mas o que é um sistema digital? Como ele interage como um sistema analógico? Como um circuito com alguns sensores pode ser capaz de tomar uma decisão e acionar um motor? Como se dá essa interação?

Todas essas perguntas podem ser respondidas considerando-se os circuitos digitais. Portanto, gradativamente, podemos construir o seu pensamento trabalhando conceitos básicos sobre circuitos digitais e evoluindo-o até culminarmos em circuitos mais complexos, como para acionamento de motores, por exemplo.

Para tanto, vamos iniciar esta unidade de ensino contextualizando um cenário prático: considere que você está iniciando sua carreira na área de microeletrônica. Você acaba de ser contratado para fazer parte do time da linha de montagem de uma empresa de desenvolvimento de sistemas embarcados. Você é extremamente curioso e não se limita a aprender somente o básico. Até onde você pretende chegar? Será possível subir de cargo ou mudar de setor? Quem sabe, em breve, você não estará em um time mais desafiador, como em um time de P&D (pesquisa e desenvolvimento)? Todas essas conquistas são alcançadas com pesquisa, estudo e conhecimento da área e dos elementos que compõem seu ambiente de trabalho.

Como mencionado, circuitos digitais fazem parte da nossa vida. Conhecer, identificar e solucionar problemas em circuitos digitais não é uma tarefa fácil. Isso exige muito estudo e dedicação. Para chegar ao time de P&D você deve seguir uma trilha que passa pelo conhecimento, dedicação e profissionalismo. Esteja atento a todos os conceitos que serão apresentados aqui. Não se contente somente com o conhecimento apresentado neste material. Extrapole-o.

Boa sorte, mãos à obra e bons estudos!

Seção 1.1

Conceitos de circuitos digitais

Rodrigo Vinícius Mendonça Pereira

Diálogo aberto

Caro aluno, esta seção tem o objetivo de direcionar o seu aprendizado de forma mobilizadora através de um contexto muito próximo da sua realidade. Todo esse contexto é para conduzir você o mais próximo possível dos conceitos de circuitos digitais. Esses conceitos são a base para que, futuramente, você comece a projetar sistemas digitais. Por isso, iremos compreender a diferença entre sinais digitais e sinais analógicos. Também veremos os conceitos fundamentais sobre circuitos digitais, como eles se enquadram em um circuito digital e como toda essa tecnologia evoluiu até os presentes dias. Vamos, em especial, compreender os níveis lógicos e as formas de ondas geradas por um circuito digital básico. Por fim, analisaremos os conceitos sobre circuitos integrados, circuitos lógicos e, finalmente, sobre sistemas digitais. Esta seção, apesar de introdutória, é fundamental. Ela explica conceitos básicos e necessários para o bom entendimento da disciplina.

Retomando o nosso contexto, imagine o seguinte cenário: você foi contratado por uma empresa de desenvolvimento de sistemas embarcados para trabalhar com a montagem de produtos. O seu trabalho consiste somente em soldar componentes desses produtos. À sua frente existe uma mesa muito grande, uma iluminação perfeita, uma bela estação de solda SMD e diversos componentes e placas de circuitos impressos prontos para serem soldados. Mas você está curioso por saber sobre os fundamentos dos circuitos digitais: como uma sequência de pulsos elétricos é transformada em código digital? Como funcionam essas peças pretas que todos chamam de microchips? As perguntas são muitas.

Além disso, você percebe duas caixas de componentes sobre a sua mesa e decide saber mais sobre eles. Uma forma é pesquisando na internet sobre a folha de dados (ou *datasheet*). Um dos componentes é um conversor analógico para digital e o outros são *flip-flops* tipo D e portas lógicas AND e OR. Você agora precisa lidar com alguns questionamentos que lhe vêm à mente.

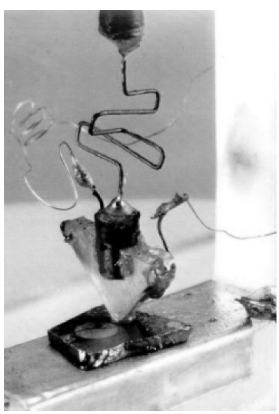
Querendo saber mais, você pesquisa sobre os componentes *flip-flops* tipo D e as portas lógicas AND e OR que estão sobre a sua mesa – pesquisa sobre encapsulamentos de circuitos integrados (ou CI). Qual a diferença de um CI DIP (do inglês *Dual In-line Package*) e um CI BGA (do inglês *Ball Grid Array*)? Todo esse questionamento vai ajudá-lo a entender muito como esses circuitos são feitos e como você irá soldá-los. Ao fim da sua pesquisa, organize as novas informações que você adquiriu como uma base, de forma a lhe ajudar nas suas pesquisas futuras.

Lembre-se de que esse cenário é um aprofundamento de seu conhecimento sobre circuitos digitais. Todo o conhecimento será feito com base em bastante leitura e pesquisa, e pode ser usado no futuro no seu dia a dia. Nas seções a seguir serão apresentadas a base para esse conhecimento, mas, conforme já dissemos, não fique limitado somente a essa fonte de conhecimento. Pesquise, converse e troque informações com outras pessoas. O mercado está favorável para funcionários proativos e esforçados. Boa sorte e bom trabalho.

Não pode faltar

Em 1947, John Bardeen, Walter Brattain e William Shockley (Figura 1.1(a)) inventavam, no Bell Labs, um pequeno dispositivo semicondutor chamado **transistor** (Figura 1.1(b)). Isso mudaria o mundo (MACNEIL, 2016).

Figura 1.1 | (a) Bardeen, Brattain e Shockley na capa da Electronics Magazine; (b) primeiro transistor de ponto de contato



Fonte: <http://www.computerhistory.org/siliconengine/invention-of-the-point-contact-transistor/>. Acesso em: 25 set. 2017.

Hoje, há trilhões de transistores espalhados pela Terra, em cada lugar onde um aparelho eletrônico possa ser encontrado, e existem ainda outros bilhões no espaço, em satélites e espaçonaves. O transistor é o carro-chefe da eletrônica e foi o dispositivo que anunciou o início da era digital. Indústrias inteiras foram criadas com base nos semicondutores. As telecomunicações, como as conhecemos, não teriam sido possíveis se não fosse pelo transistor (BELL, [s.d.]).

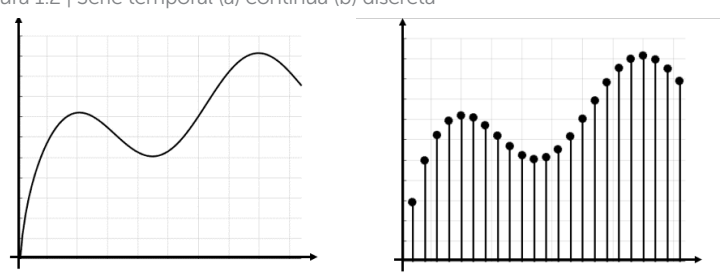
Podemos dividir os circuitos eletrônicos em duas categorias principais: circuitos analógicos e circuitos digitais. Enquanto na eletrônica analógica lida-se com grandezas de valores contínuos, na eletrônica digital nosso objeto de estudo nesta unidade, lida com grandezas de valores discretos. É importante, no entanto, que você tenha conhecimento das duas áreas, uma vez que diversas aplicações exigem conhecimento de ambas.

Assimile

Uma **grandeza analógica** pode assumir infinitos valores de amplitude dentro de um intervalo. Já uma **grandeza digital** somente pode assumir determinados valores de amplitude bem definidos dentro de um intervalo. Em geral, a maior parte do que podemos medir quantitativamente na natureza são exemplos de grandezas analógicas, como a temperatura, o tempo, a pressão, a distância e o som.

Na Figura 1.2(a), temos o exemplo de uma série temporal contínua no tempo. Ao representar uma série contínua no tempo amostrando o sinal em instantes específicos, conforme podemos ver na Figura 1.2(b), estaremos convertendo uma grandeza analógica em um formato no qual é possível digitalizar, representando cada valor por um código digital.

Figura 1.2 | Série temporal (a) contínua (b) discreta



(a) (b)

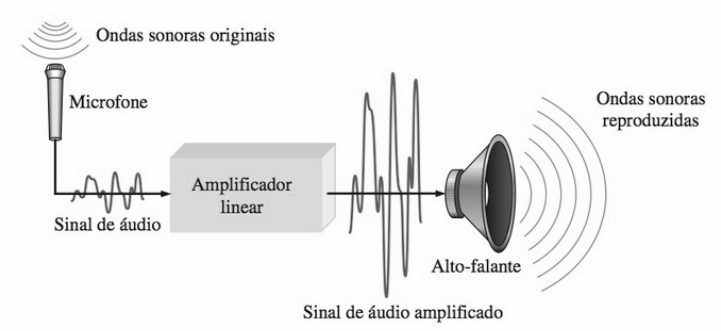
Fonte: elaborada pelo autor.

É importante perceber, no entanto, que a Figura 1.2(b) não é, de fato, uma representação digital de uma grandeza analógica.

Representar um sinal no formato digital pode ser vantajoso em relação à representação analógica em aplicações eletrônicas, uma vez que os dados digitais podem ser processados e transmitidos de forma mais eficiente e confiável que os dados analógicos e são consideravelmente mais simples de armazenar. Por exemplo, um arquivo de áudio no formato digital pode ser armazenado de forma mais compacta e reproduzido com maior precisão e pureza do que quando está no formato analógico, além disso, os dados digitais estão menos sujeitos ao ruído.

Podemos citar, como exemplo de aplicação de eletrônica analógica, um sistema de amplificação de áudio. O diagrama básico da Figura 1.3 ilustra as ondas sonoras, que são de natureza analógica, sendo captadas por um microfone e convertidas em uma pequena tensão analógica, denominada sinal de áudio. Essa tensão varia continuamente de acordo com as variações na amplitude e frequência do som, e é aplicada na entrada de um amplificador linear. A saída do amplificador, que é uma reprodução amplificada da tensão de entrada, é enviada para o alto-falante, que converte o sinal de áudio de volta para o formato de ondas sonoras com um volume muito maior que o das ondas sonoras originais (FLOYD, 2007).

Figura 1.3 | Esquema básico de um sistema de amplificação de áudio

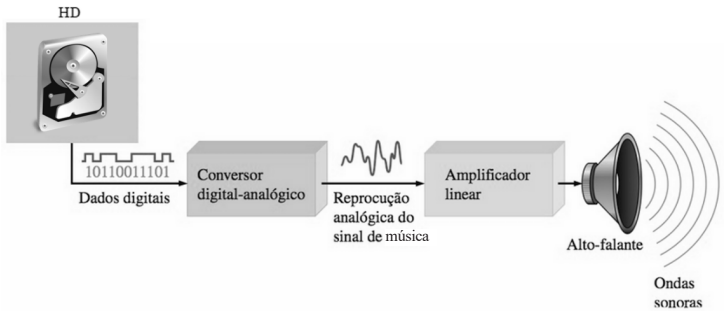


Fonte: Floyd (2007, p. 21).

Já ao tocar um mp3 em seu computador ou em um tocador de mp3, são usados circuitos digitais e analógicos. O diagrama de blocos simplificado da Figura 1.4 ilustra o seu princípio básico. A música no formato digital é armazenada no disco rígido (HD, do inglês *hard disk*). Ao tocar a música, os dados digitais são transferidos para um conversor digital-analógico que converte os dados digitais em um

sinal analógico que é uma reprodução elétrica da música original. Esse sinal é, então, amplificado e enviado para o alto-falante.

Figura 1.4 | Diagrama de blocos básico de um tocador de mp3



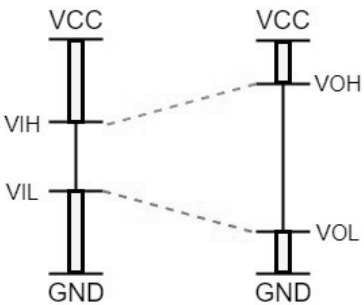
Fonte: adaptada de Floyd (2007, p. 21).

Ainda que a lógica multinível venha sendo investigada há muito tempo, a lógica de dois níveis continua sendo a mais viável. Assim, a eletrônica digital opera com apenas dois estados possíveis, representados por níveis lógicos diferentes, um ALTO e um BAIXO. Os dois estados também podem ser representados por níveis de corrente, bits e ressaltos num CD, DVD etc. Mesmo que os sinais verdadeiros (físicos) que correspondem a ALTO ou BAIXO sejam de fundamental importância para os desenvolvedores de tecnologia, eles são irrelevantes para os usuários do sistema (programadores, por exemplo). Nos sistemas digitais, esses dois estados são combinados formando códigos usados para representar números, símbolos, caracteres alfabéticos e outros tipos de informações, os bits são representados por dois níveis de tensão diferentes. Em geral, o bit 1 é representado por uma tensão maior, a qual chamamos por nível lógico ALTO, e o bit 0 é representado por uma tensão menor, a qual chamamos de nível lógico BAIXO. Quando em um sistema o bit 1 é representado por uma tensão menor, ou nível BAIXO, e o bit 0 é representado por uma tensão maior, ou nível ALTO, esse sistema opera no que chamamos de lógica negativa.

Na teoria, um nível único de tensão é usado para representar um nível lógico ALTO e o outro nível de tensão é usado para representar um nível BAIXO. Na prática, no entanto, em um circuito digital, os níveis lógicos são representados por um intervalo de tensão entre um valor mínimo e um valor máximo especificados, não podendo haver sobreposição entre as faixas aceitáveis para os níveis ALTO e BAIXO.

Considere como nosso padrão 0 volts (ou GND ou terra) para o dígito 0 e VCC para o dígito um 1. Você percebeu que existe uma infinidade de valores possíveis entre 0 volts e VCC? Dessa forma, é preciso saber como se dá essa condição intermediária (entre 0 e VCC). Vamos começar entendendo os níveis de tensão possíveis em circuitos digitais. Para tensões de circuitos digitais na entrada, é adotada a nomenclatura **VIH** (V *Input High*) para a menor tensão na qual ainda é possível representar o valor 1. A maior tensão na qual ainda é representado o valor 0 na entrada é chamada **VIL** (V *Input Low*). Fato similar ocorre na saída de um circuito digital. Os níveis de tensão podem variar entre **VCC** e **VOH** (V *Output High*) para indicar o valor 1, e entre **VOL** (V *Output Low*) e GND para indicar 0. A Figura 1.5 ilustra todo esse conceito de representação de uma entrada e uma saída digital e seus respectivos níveis de tensão V *High* e V *Low*. O valor intermediário (entre VIH e VIL, entre VOH e VOL) são tensões que não são usadas e não são símbolos válidos, ou seja, um circuito digital deve desconsiderar esses valores.

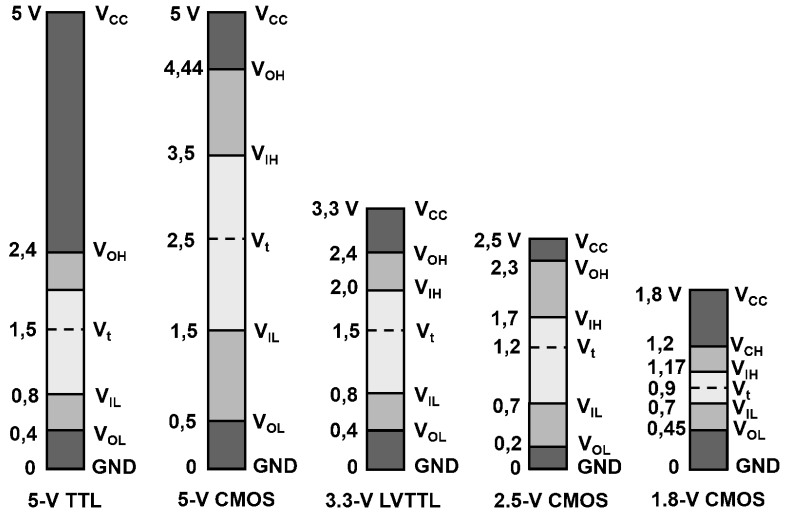
Figura 1.5 | Entrada e saída digitais



Fonte: elaborada pelo autor.

Ainda a respeito de níveis de tensão, existem diferentes tecnologias que se aplicam aos circuitos digitais. Existem, por exemplo, tanto circuitos TTL e CMOS quanto o LTTL (*Low Power TTL*), BiCMOS, LVC (*Low Voltage CMOS*), entre outros. Para cada tecnologia, existe um padrão de nível de tensão. A Figura 1.6 escalona as tecnologias com relação aos níveis de tensão (VCC, GND, V *High* e V *Low*).

Figura 1.6 | Níveis de tensão digitais



Fonte: elaborada pelo autor.

Definido como os níveis de tensão representam dígitos binários, basta agora entender que podemos gerar esses níveis de tensão em forma de onda, consistidas em níveis de tensão que comutam entre os níveis lógicos ALTO e BAIXO. Na Figura 1.7(a), podemos ver um pulso positivo, que ocorre quando a tensão ou a corrente passa do nível BAIXO normal para o nível ALTO, e, em seguida, retorna para o nível BAIXO. Já na Figura 1.7(b), temos um pulso negativo, gerado quando a tensão passa do nível ALTO normal para o nível BAIXO e retorna para o nível ALTO. Uma forma de onda digital consiste em uma série desses pulsos.

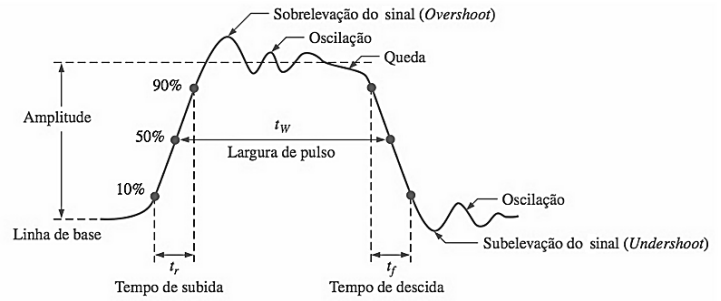
Figura 1.7 | Pulso ideal (a) positivo (b) negativo

Fonte: elaborada pelo autor.

As transições em um pulso são classificadas como positiva ou borda de subida (na Figura 1.7(a), ocorre em t_0), ou como negativa ou borda de descida (na Figura 1.7(a) ocorre em t_1).

Os pulsos vistos na Figura 1.7 são ideais, porque se considera que as bordas de subida e descida comutam instantaneamente. Na prática, essas transições não ocorrem no mesmo instante. Na Figura 1.8 estão ilustradas as características que um pulso real pode exibir, embora no projeto da maioria dos circuitos digitais consideramos os pulsos ideais.

Figura 1.8 | Características de um pulso não ideal



Fonte: adaptada de Floyd (2007, p. 24).

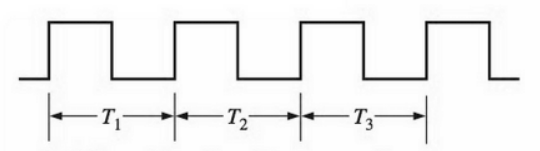
A sobre-elevação do sinal (*overshoot*) e oscilações são produzidas algumas vezes por efeitos de indutância e capacitância parasitas. A inclinação pode ser causada por capacitância parasita e circuitos resistivos que formam um circuito RC com uma pequena constante de tempo. O tempo necessário para um pulso passar do nível BAIXO para o ALTO é denominado tempo de subida (t_r – *rise time*), e o tempo necessário para a transição do nível ALTO para o nível BAIXO é denominado tempo de descida (t_f – *fall time*). A largura de pulso (t_w – *pulse width*) é a medida da duração do pulso, e é frequentemente definida como o intervalo de tempo entre os pontos de 50% das bordas de subida e descida (FLOYD, 2007).

Segundo Floyd (2007), a maioria das formas de onda encontradas em sistemas digitais são compostas de uma série de pulsos, algumas vezes denominados trem de pulsos, podendo ser classificadas como periódicas ou não periódicas. Uma forma de onda periódica é aquela que se repete num intervalo fixo, denominado de período (*T*). A frequência (*f*) é a taxa com que ela se repete, e é medida em hertz (Hz). Uma forma de onda não periódica, é claro, não se repete em intervalos fixos, e pode ser composta de pulsos com larguras aleatórias e/ou intervalos aleatórios de tempo entre os pulsos.

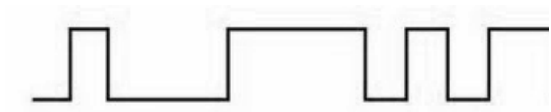
Exemplificando

Um exemplo de cada tipo é mostrado na Figura 1.9.

Figura 1.9 | Formas de onda digital (a) periódica (b) não periódica



(a)



(b)

Fonte: Floyd (2007, p. 24).

Na Figura 1.9(a) o período é fixo, ou seja, $T_1 = T_2 = T_3 = \ldots = T_n$, e a frequência é dada por $f = \frac{1}{T}$.

Podemos dizer que cada circuito digital pode ser descrito por uma **função lógica** que processa os bits que o circuito recebe. Por exemplo, considere ***a*** e ***b*** dois bits recebidos por um certo circuito, o qual produz o bit ***y*** na saída. A seguir, damos alguns exemplos de funções binárias básicas.

$y = \text{NOT } a$ (também representado por $y = a'$).

$y = a \text{ OU } b$ (ou $y = a + b$, em que "+" representa o OU (OR) lógico, e não deve ser confundido com o sinal de adição aritmética).

$y = a \text{ E } b$ (ou $y = a \cdot b$, em que "$\cdot$" representa o E (AND) lógico, e não deve ser confundido com o sinal de multiplicação aritmética).

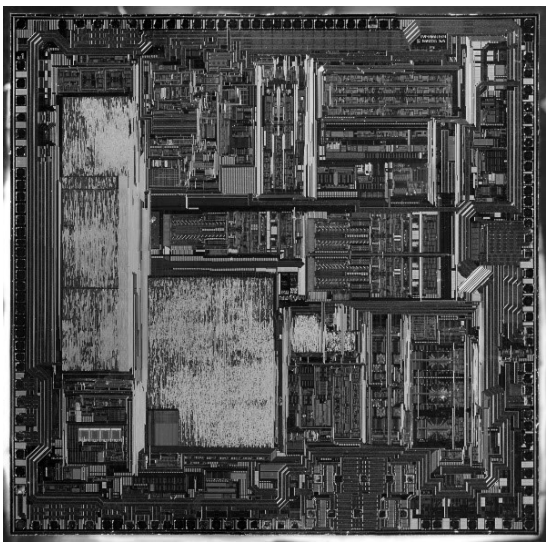
Na Seção 1.3 aprofundaremos esse assunto. Porém, podemos adiantar que a primeira função é denominada **inversão** ou **negação**, pois ***y*** é o oposto de ***a***. A segunda é denominada função OU, porque basta ter uma entrada alta para que a saída seja alta. Por fim, a terceira função é denominada E, porque a saída é alta somente quando todas as entradas forem altas.

Reflita

No seu cotidiano, o que pode ser considerado uma função lógica? Por exemplo, no café da manhã, se você tiver leite e café, você toma café com leite; se você só tiver café, toma somente o café. Podemos considerar essas decisões como funções lógicas? A vida é feita de circuitos lógicos, ou seja, de decisões. Saiba tomá-las com cautela.

Por fim, segundo Vahid (2008), um circuito integrado (ou CI, microchip ou chip), como o ilustrado na Figura 1.10, consiste em um circuito eletrônico miniaturizado, feito basicamente de silício, do tamanho de uma unha.

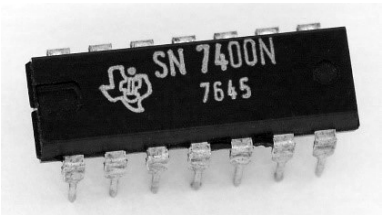
Figura 1.10 | Circuito integrado com encapsulamento removido



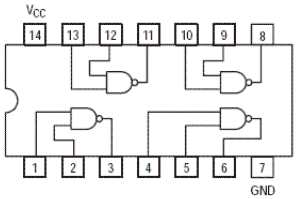
Fonte: <https://pixabay.com/pt/circuito-integrado-dispositivo-chip-876099/>. Acesso em: 26 set. 2017.

Foi a invenção dos circuitos integrados que permitiu o início de toda essa revolução que ocorreu no século XX e ainda ocorre nos dias atuais. Já os circuitos lógicos são os circuitos que fazem uso dos blocos lógicos, explicados anteriormente. A Figura 1.11 exemplifica uma porta lógica NAND.

Figura 1.11 | Porta NAND – (a) circuito integrado e (b) esquemático do circuito



(a)



(b)

Fonte: National (1989, p. 1).

Esses circuitos foram amplamente utilizados no início da eletrônica. Atualmente, ainda podem ser utilizados, mas é mais fácil programar um microcontrolador. Portanto, eles são importantes, mas não muito utilizados no dia a dia. Para Tocci (2011), sistemas digitais são uma combinação de dispositivos projetados para manipular informações lógicas ou valores físicos representados no formato digital. Podemos citar como sistemas digitais: computadores pessoais, celulares, calculadoras avançadas, sistemas de áudio e vídeo.

Pesquise mais

Segundo o *MIT Technology Review* (2017), existem tendências que poderão afetar diretamente a nossa economia, medicina e cultura nos próximos quatro a 15 anos. Listamos algumas a seguir:

1. Reversão de paralisias: através de implantes, cientistas pretendem reestabelecer movimentos de pessoas com lesões graves – disponível em: dez a 15 anos.

2. Pagamento através de reconhecimento facial: detecção de padrões faciais que autorizam o pagamento de contas – já disponível.

3. Computadores quânticos na prática: Google, IBM, Intel, Microsoft, entre outras, investem massivamente em sistemas quânticos completos – disponível em: quatro a cinco anos.

4. Selfie 360: câmeras permitem a gravação de imagens em 360° – já disponível.

5. Terapia genética 2.0: através de manipulações nos genes das células é possível tratar câncer, problemas do coração, entre outras doenças – já disponível.

O mais interessante está no fato que essas tecnologias possuem, em algum momento, alguma relação com a eletrônica digital. Pesquise sobre as inovações tecnológicas envolvendo sistemas digitais, o assunto é fascinante.

Prezado aluno, o assunto não se encerra com o conteúdo apresentado aqui. Como mencionamos anteriormente, pesquise outras fontes e questione os especialistas no assunto. Bons estudos!

Sem medo de errar

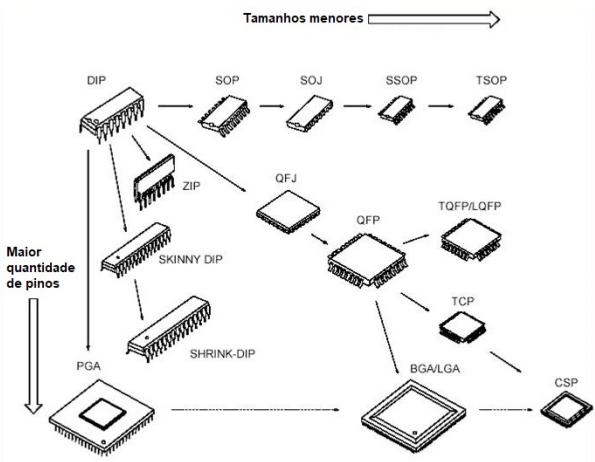
Relembrando a situação problema-proposta, você foi contratado por uma empresa de desenvolvimento de sistemas embarcados para trabalhar com a montagem/solda de produtos. Na sua bancada de trabalho existem diversos componentes e placas de circuitos impressos prontos para serem soldados.

Todo esse contexto leva você a se questionar sobre os componentes que estão sobre a sua mesa. É fundamental saber como encontrar informações sobre eles. Na internet, pesquise sobre a folha de dados de um CI o 74LS74. Procure da seguinte forma: 74LS74 *datasheet*; ou pesquise sobre *flip-flops* tipo D. Fique atento a detalhes como a pinagem do CI e as condições de operação.

Pesquise detalhes como as definições de tempo do componente e as condições de operação recomendadas.

Para finalizar, entenda os vários encapsulamentos possíveis para os componentes que você trabalha. Observe a Figura 1.12, ela descreve muito bem a tecnologia de encapsulamento pela quantidade de pinos do microchip.

Figura 1.12 | Diferentes tipos de encapsulamento



Fonte: adaptada de <http://bga.blog.tartanga.eus/files/2015/06/Imagen1.jpg>. Acesso em: 26 set. 2017.

Pesquise o custo entre um mesmo CI mas com diferentes encapsulamentos. Quando se usa um e quando se usa outro encapsulamento?

Relembrando: esse cenário é um aprofundamento ao seu conhecimento sobre circuitos digitais. Dessa forma, todo o conhecimento será feito com base em bastante leitura e pesquisa. A ideia aqui é criar uma base sólida, e que esse conhecimento seja empregado no futuro no seu dia a dia. Não fique limitado somente a esta fonte de conhecimento. Pesquise, converse e troque informações com outras pessoas.

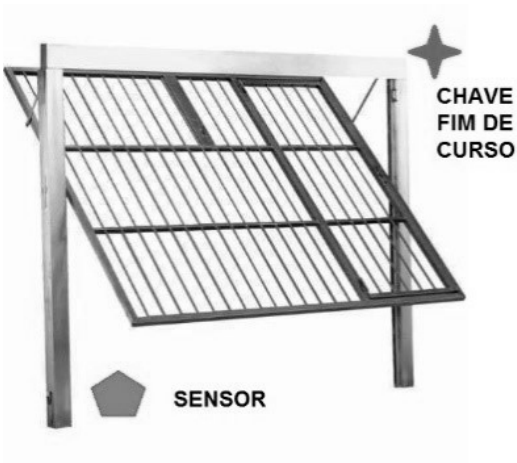
Avançando na prática

Portão automático

Descrição da situação-problema

Caro aluno, você está adquirindo um conhecimento muito amplo sobre circuitos digitais. Inclusive, você já começa a observar soluções para problemas cotidianos. Você reparou que na casa da sua avó o portão automático pode ser melhorado. Ocorre que esse portão não possui um sensor de presença, ou seja, caso exista alguém debaixo dele, ele ignora isso e fecha. Observe a Figura 1.13. Ela possui um sensor e uma chave de fim de curso.

Figura 1.13 | Portão automático



Fonte: elaborada pelo autor.

Qual seria uma lógica para a seguinte situação?

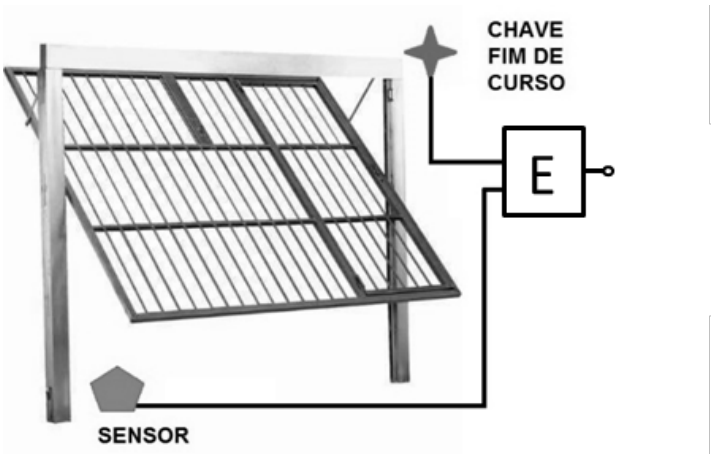
1. Portão no alto, chave ativa (em 1).
    a) Sensor sem pessoas embaixo do portão (em 0).
    b) Portão pode fechar.
2. Portão no alto, chave ativa (em 1).
    a) Sensor com pessoas embaixo do portão (em 1).
    b) Portão não pode fechar.
3. Portão embaixo, chave desativada (em 0).
    a) Portão pode abrir.

Utilize conceitos das funções lógicas trabalhadas na unidade.

Resolução da situação-problema

Caro aluno, uma solução aparentemente simples é utilizar uma função lógica AND (ou E) tendo como entradas a CHAVE DE FIM DE CURSO e o SENSOR de presença. A saída seria utilizada na entrada do circuito que controla a subida e descida do portão, como mostra a Figura 1.14.

Figura 1.14 | Portão automático com circuito para controle



Fonte: elaborada pelo autor.

Existem diversas soluções. A princípio, esta solução pode resolver o problema. Pesquise como fazer esse teste fisicamente.

Faça valer a pena

1. Analise a notícia:

TV Digital: veja cronograma de desligamento do sinal analógico nas cidades

Até o fim de 2017, oito capitais, além do DF, passam a receber apenas sinal digital. Quem tem TV analógica precisa ter conversor e antena digital.

Fonte: <https://goo.gl/Kq7LzS>. Acesso em: 13 out. 2017.

Com relação a um sinal digital e um sinal analógico, podemos afirmar que:

a) O sinal analógico assume valores finitos (ou discretos) no tempo;
o sinal digital pode assumir diversos valores no tempo.

b) O sinal analógico sempre assume valores inconstantes no tempo;
o sinal digital pode assumir diversos valores no tempo.

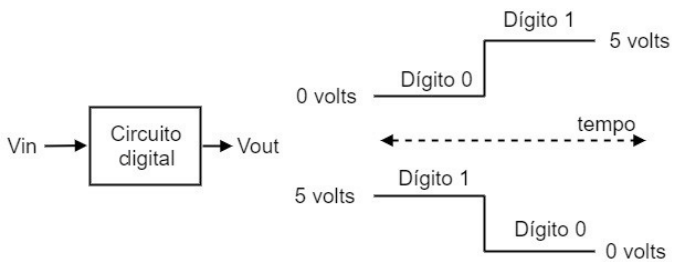
c) O sinal digital sempre assume valores indeterminados no tempo;
o sinal analógico é um sinal que pode assumir somente um valor no tempo.

d) O sinal digital nunca assume valores finitos (ou discretos) no tempo;
o sinal analógico não pode assumir diversos valores no tempo.

e) O sinal digital sempre assume valores finitos (ou discretos) no tempo;
o sinal analógico pode assumir diversos valores no tempo.

2. Segundo Tocci (2011), circuitos digitais são dispositivos projetados para produzir tensões de saída entre as faixas VCC – VOH e VOL – GND para os níveis 1 e 0, respectivamente. Também são dispositivos projetados para responder a tensões de entrada entre as faixas VCC – VIH e VIL – GND para os níveis 1 e 0, respectivamente. Para Tocci (2011), o modo como cada circuito digital interage a uma entrada é denominado lógica do circuito, ou seja, cada circuito segue regras determinadas pelas suas funções lógicas (ou portas lógicas) implementadas internamente.

Figura 1.15 | Circuitos digitais e os níveis binários (0 e 1)

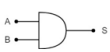
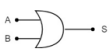


Fonte: adaptada de Tocci (2011).

A figura representa um circuito digital cuja entrada Vin produz uma saída Vout. Tanto o sinal na entrada quanto a resposta na saída podem ser representados pelas duas formas de onda ao lado. Dentro desse circuito existem funções lógicas. Essas funções podem representar dois estados, somente: ligado ou desligado; 0 ou 1; aberto ou fechado; ou seja, tanto na entrada quanto na saída desses blocos lógicos são permitidos somente esses valores.
Considerando o texto, qual será a mensagem (dígitos) enviada pela porta Vout se for lida na porta a seguinte sequência: GND (mais significativo) – VCC – VCC – GND – GND – VCC – GND - VCC (menos significativo)? Considere GND e VCC padronizados no texto.
a) 1 (mais significativo) – 1 – 1 – 0 – 0 – 0 – 0 – 1 (menos significativo).
b) 0 (mais significativo) – 1 – 1 – 0 – 0 – 1 – 0 – 1 (menos significativo).
c) 1 (mais significativo) – 0 – 0 – 1 – 1 – 0 – 1 – 0 (menos significativo).
d) 0 (mais significativo) – 1 – 0 – 1 – 0 – 1 – 0 – 0 (menos significativo).
e) 1 (mais significativo) – 1 – 1 – 0 – 0 – 1 – 0 – 1 (menos significativo).

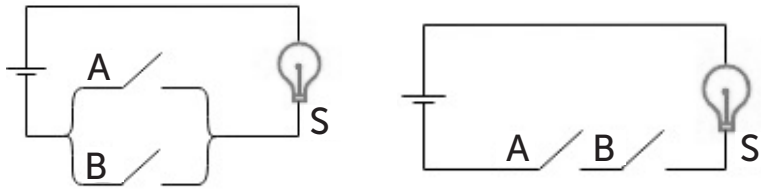
3. As três principais portas lógicas utilizadas em circuitos digitais são: porta AND (lógica E), OR (lógica OU) e NOT (lógica NÃO ou INVERSORA). Considere a tabela a seguir:

Tabela 1.1 | Blocos lógicos básicos

Porta	Símbolo	Tabela verdade	Função lógica	Expressão
E **AND**		**A** **B** **S** 0 0 0 0 1 0 1 0 0 1 1 1	Assume valor 1 quando todas a entradas forem 1 e 0 nos outros casos.	$S = A \cdot B$
OU **OR**		**A** **B** **S** 0 0 0 0 1 1 1 0 1 1 1 1	Assume valor 0 quando todas a entradas forem 1 e 1 nos outros casos.	$S = A + B$
NÃO (ou INVERSORA) **NOT**		**A** **S** 0 1 1 0	Inverte o valor da variável de entrada A.	$S = A'$

Analise a Figura 1.16 considerando o texto-base e assinale a alternativa correta:

Figura 1.16 | Circuito com chaves eletromecânicas



(a) (b)

Fonte: elaborada pelo autor.

a) A Figura 1.16(**a**) corresponde a uma porta lógica **OR**. A lâmpada ficará acesa se e somente se ambas as entradas forem acionadas.
A Figura 1.16(**b**) corresponde a uma porta lógica **AND**. A lâmpada ficará acesa eventualmente quando ambas as entradas forem acionadas.
b) A Figura 1.16(**a**) corresponde a uma porta lógica **AND**. A lâmpada ficará acesa se ambas as entradas forem acionadas.
A Figura 1.16(**b**) corresponde a uma porta lógica **OR**. A lâmpada ficará acesa se A ou B ou ambas entradas forem acionadas.
c) A Figura 1.16(**a**) corresponde a uma porta lógica **OR**. A lâmpada ficará acesa se A ou B ou ambas forem acionadas.
A Figura 1.16(**b**) corresponde a uma porta lógica **AND**. A lâmpada ficará acesa se e somente se ambas entradas forem acionadas.
d) Tanto a Figura 1.16(**a**) quanto a Figura 1.16(**b**)podem ligar a lâmpada, independentemente da posição na qual estão as portas A e B.
e) Não é possível abstrair portas lógicas observando o desenho. O que se conclui é que, independentemente da posição das chaves, as lâmpadas não serão acesas.

Seção 1.2

Sistemas numéricos e códigos binários

Hugo Tanzarella Teixeira

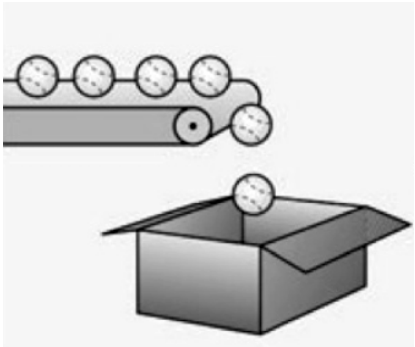
Diálogo aberto

Na seção anterior, vimos que nos sistemas digitais o sistema de **numeração binário** e os códigos digitais são de fundamental importância. Por exemplo, quando pressionamos um número, como, o 2, no teclado de uma calculadora, duas coisas acontecem: o número é enviado ao display, para que o usuário tenha a certeza de que apertou a tecla correta; e o número também é enviado ao **circuito responsável pelos cálculos**. Como sabemos, nos sistemas digitais somente são permitidos símbolos com dois valores (binários). Então, como o número 2 é realmente representado nesse sistema binário?

Um numeral é um símbolo ou grupo de símbolos que representa um número. Em nosso cotidiano, estamos acostumados a quantificar nossas medidas com números decimais entre 0 e 9, mas será que este é o sistema de numeração mais adequado a todas as medições e sistemas? Será que existem outros além do binário e do decimal? Quais são as vantagens de cada sistema existente?

Para entendermos um pouco melhor essa situação, devemos retomar nosso contexto: você trabalha em uma empresa de desenvolvimento de sistemas embarcados, e seu chefe notou sua curiosidade e entusiasmo pelo assunto. Por isso, pediu-lhe que você participasse de uma reunião com a equipe de desenvolvimento. Eles precisam desenvolver um sistema de contagem de bolas de tênis em uma linha de montagem. O sistema é bastante simples: de fato, as bolas de tênis, ao final da produção, são colocadas em uma caixa a partir de uma correia transportadora, como pode ser visto na Figura 1.15. Cada caixa é preenchida com nove bolas.

Figura 1.15 | Problema da contagem de bolas de tênis



Fonte: adaptada de Floyd (2007, p. 67).

Você está entusiasmado com a sua oportunidade e quer surpreender a todos na reunião com suas ideias. Para ajudá-lo, nesta seção, conheceremos o sistema de numeração binário, vamos aprender as suas relações com o sistema decimal, e também com os outros sistemas de numeração importantes para os sistemas digitais, como hexadecimal e octal. Com o intuito de fornecer uma base de conhecimento para o entendimento de como os computadores e muitos outros sistemas digitais funcionam, vamos aprender algumas operações aritméticas com números binários. Por fim, introduziremos o conceito de codificação binária.

Mantenha o foco e bons estudos.

Não pode faltar

O **sistema de numeração** consiste de um sistema em que um conjunto de números é representado por numerais de forma consistente. Você com certeza já está familiarizado com o sistema de numeração **decimal** porque usa os números decimais todos os dias. Embora muito comum, a estrutura de pesos dos números decimais não é frequentemente compreendida. Nesta seção, revisaremos a estrutura dos números decimais, o que ajudará você a compreender mais facilmente a estrutura dos sistemas de numeração **binário**, **octal** e **hexadecimal**, fundamentais no estudo de computadores e eletrônica digital.

Em um **sistema numérico genérico** podemos representar um número *N* qualquer por uma equação geral:

$$N = d_{p-1}b^{p-1} + d_{p-2}b^{p-2} + \ldots + d_1b^1 + d_0b^0 + d_{-1}b^{-1} + \ldots + d_{-(q-2)}b^{-(q-2)} + d_{-(q-1)}b^{-(q-1)} + d_{-q}b^{-q},$$

(1.1)

Nesta equação, *b* é a base numérica ou raiz do sistema, *d* corresponde a dígitos singulares possíveis no sistema, *p* é a potência da base correspondente à sua posição relativa aos números inteiros e *q* é a potência da base correspondente à sua posição relativa aos números fracionários. A base ou raiz de um sistema numérico é definida como um número que pode ocorrer em cada posição desse sistema, composto de dígitos diferentes (SZAJNBERG, 2014). Os dígitos de um sistema numérico devem satisfazer a seguinte relação:

$$(b-1) \geq d \geq 0 \qquad (1.2)$$

O **sistema de numeração decimal** tem base ou raiz **10** e, portanto, dez dígitos diferentes, de 0 a 9, e cada um pode ser usado em qualquer posição do número. No sistema decimal, *p* corresponde às unidades, dezenas, centenas, milhares, e assim por diante.

$$10^0 = 1; 10^1 = 10; 10^2 = 100; 10^3 = 1000\ldots$$

E *q* corresponde a décimos, centésimos, milésimos, e assim por diante.

$$10^{-1} = 0,1; 10^{-2} = 0,01; 10^{-3} = 0,001\ldots$$

Exemplificando

Por exemplo, podemos representar o número decimal 3012,84 pela equação geral (1.1), de modo que

$$3012,84 = 3\times10^3 + 0\times10^2 + 1\times10^1 + 2\times10^0 + 8\times10^{-1} + 4\times10^{-2}.$$

Podemos interpretar os elementos da soma usando pesos. Começando da posição de unidades, teremos, à esquerda, os pesos (1000), (100), (10) e (1) para a parte inteira e, na parte fracionária, à direita, os pesos (0,1) e (0,001), conforme ilustrado na Tabela 1.2.

Tabela 1.2 | Pesos do sistema decimal

Parte inteira				Parte fracionária		
1000	100	10	1	0,1	0,01	0,001
3	0	1	2	8	4	0

Fonte: elaborada pelo autor.

O **sistema de numeração binário** tem como base o número 2 e, portanto, apenas dois dígitos, **0** e **1**, como podemos confirmar resolvendo a equação (1.2) para $b = 2$: $(b-1) \geq d_1 \geq 0 \rightarrow (2\text{-}1) \geq d_1 \geq 0$.

Os pesos em um número binário são baseados em potência de dois. Portanto, conforme a fórmula genérica (1.1), um número na forma binária é apresentado como:

$$N = d_{p-1}2^{p-1} + d_{p-2}2^{p-2} + \ldots + d_1 2^1 + d_0 2^0 + d_{-1}2^{-1} + \ldots$$

$$+ d_{-(q-2)}2^{-(q-2)} + d_{-(q-1)}2^{-(q-1)} + d_{-q}2^{-q}.$$

(1.3)

Exemplificando

De maneira semelhante ao exemplo anterior, um número binário, por exemplo, 1101,011, será representado por um somatório.

$$1011{,}01_2 = 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2},$$

$$= 1\times 8 + 0\times 4 + 1\times 2 + 1\times 1 + 0\times 1/2 + 1\times 1/4,$$

$$= 8 + 2 + 1 + 1\times 1/4 = 11{,}25_{10}.$$

O valor de cada dígito é determinado pela sua posição em relação à vírgula binária. O procedimento é idêntico ao realizado no sistema decimal. No entanto, aqui os pesos correspondem a potências de 2, conforme ilustrado na Tabela 1.3.

Tabela 1.3 | Pesos do sistema binário

Parte inteira				Parte fracionária		
8	4	2	1	0,5	0,25	0,125
1	0	1	1	0	1	0

Fonte: elaborada pelo autor.

Como você já deve ter notado, a numeração binária requer uma grande quantidade de dígitos para representar números relativamente pequenos. Uma análise de estados numéricos na escrita pode se tornar laboriosa e tediosa. Para simplificar essa tarefa, foram desenvolvidos os sistemas **octal** e **hexadecimal**.

Reflita

Números binários longos são difíceis de serem lidos e escritos porque é fácil omitir ou trocar um bit. Como os computadores entendem apenas os números **0** e **1**, é necessário usar esses dígitos quando se programa em "linguagem de máquina" (FLOYD, 2007). Portanto, tente imaginar como seria escrever uma instrução de dezesseis bits para um sistema microprocessado usando apenas **zeros** e **uns**.

O sistema **octal** é composto de oito dígitos, de 0 a 7. Por isso, podemos considerar a contagem em octal bastante similar à contagem em decimal, com a exceção de que os dígitos 8 e 9 não são usados. Já um número no sistema **hexadecimal** tem base **16**, logo, usa 16 dígitos. Já que no sistema decimal convencional existem somente 10 dígitos singulares (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), usamos as primeiras seis letras do alfabeto latino (A, B, C, D, E, F) para completar a sequência dos 16 dígitos.

Exemplificando

Por exemplo, $235,2_8$, em octal, equivale a 157_{10} em decimal. Para entendermos essa conversão, basta resolver a equação (1.1),

$$235,2_8 = 2\times8^2 + 3\times8^1 + 5\times8^0 + 2\times8^{-1} = 128 + 24 + 5 + 0,25 = 157,25_{10}.$$

Os seus pesos correspondem a potências de 8, conforme ilustrado na Tabela 1.4.

Tabela 1.4 | Pesos do sistema octal

Parte inteira			Parte fracionária	
64	8	1	0,125	0,0156
2	3	5	2	0

Fonte: elaborada pelo autor.

E o número hexadecimal $D10,A2_{16}$ é $3344,6328125_{10}$,

$$D10,A2_{16} = 13\times16^2 + 1\times16^1 + 0\times16^0 + 10\times16^{-1} + 2\times16^{-2} = 3344,6328125_{10}$$

Os seus pesos, por sua vez, correspondem a potências de 16, conforme pode ser visto na Tabela 1.5.

Tabela 1.5 | Pesos do sistema hexadecimal

Parte inteira			Parte fracionária	
256	16	1	0,0625	0.00390625
13	1	0	10	2

Fonte: elaborada pelo autor.

A **conversão** dos sistemas binário, octal e hexadecimal para o sistema decimal é simples, basta aplicar a equação geral (1.1) utilizando a potência adequada e resolvendo a somatória, como vimos nos exemplos dados até agora.

Um dos métodos de **conversão do sistema decimal para o binário** é chamado método de divisões sucessivas, e consiste em dividir o número decimal por 2 sucessivamente. Como resultado, são obtidos quocientes *Q* e restos *R*. Uma vez que estamos dividindo sempre por 2, o resto será sempre igual a 1 ou a 0.

O primeiro resto obtido corresponde ao último dígito do número binário; a divisão seguinte fornece o penúltimo; e assim sucessivamente até o quociente ser nulo. Os restos em sequência formam o número binário procurado, como podemos ver a seguir, na conversão do número decimal 84 para a base binária:

$$84 \div 2 = 42 \rightarrow R_0 = 0$$

$$42 \div 2 = 21 \rightarrow R_1 = 0$$

$$21 \div 2 = 10 \rightarrow R_2 = 1$$

$$10 \div 2 = 5 \rightarrow R_3 = 0$$

$$5 \div 2 = 2 \rightarrow R_4 = 1$$

$$2 \div 2 = 1 \rightarrow R_5 = 0$$

$$1 \div 2 = 0 \rightarrow R_5 = 1$$

O número binário é formado pela sequência $R_5, R_4, R_3, R_2, R_1, R_0 = 1010100$. Para converter um número decimal fracionário em seu equivalente binário, basta multiplicá-lo por 2. O produto obtido consiste de uma parte fracionária X e uma parte inteira I, que somente será igual a 1 ou 0. Uma multiplicação subsequente por 2 produz uma nova parte fracionária e uma nova parte inteira, e assim sucessivamente até que a parte inteira se iguale a zero. Os dígitos da parte inteira irão compor o desejado número binário.

Reflita

Converta o número decimal 0,215 para seu equivalente binário utilizando a técnica descrita. Perguntamos o seguinte: esse número pode ser representado por um número binário finito exato?

As conversões do sistema decimal para os sistemas octal e hexadecimal seguem o mesmo método da conversão decimal-binária. No entanto, ao invés de dividir por 2, dividimos por 8, no caso octal, e por 16, no caso hexadecimal.

Faça você mesmo

Agora chegou a sua vez de praticar: converta 45_{10} para o sistema octal e 1015_{10} para o sistema hexadecimal.

A conversão de octal para binário e de hexadecimal para binário é direta: cada dígito no sistema octal equivale a uma sequência de três dígitos binários, ao passo que cada dígito hexadecimal equivale a uma sequência de quatro dígitos binários, conforme vemos na Tabela 1.6.

Tabela 1.6 | Sistemas de numeração

Sistema			
Decimal	Binário	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Fonte: elaborada pelo autor.

Assimile

O uso dos números hexadecimais é bastante conveniente na programação de sistemas digitais, uma vez que cada dígito hexadecimal representa um número binário de 4 bits e a maioria dos sistemas digitais processa dados binários em grupos múltiplos de quatro bits (FLOYD, 2007).

O estudo das **operações aritméticas no sistema binário** é de extrema importância no âmbito da eletrônica digital e dos microcontroladores, em que operações são constantemente realizadas. Assim, saber como elas são realizadas possibilita a otimização e o melhor aproveitamento dos componentes.

A **adição** binária envolve a adição de dois ou mais números binários. A Tabela 1.7 mostra as regras para duas entradas.

Tabela 1.7 | Adição binária

Entrada 1	Entrada 2	Soma	Vai um
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fonte: elaborada pelo autor.

A operação de **subtração binária** também oferece quatro combinações possíveis, que podem ser vistas na Tabela 1.8.

Tabela 1.8 | Subtração binária

Entrada 1	Entrada 2	Subtração	Vem um
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Fonte: elaborada pelo autor.

A **multiplicação** no sistema binário é executada conforme o sistema decimal, portanto:

$$0 \times 0 = 0; \quad 0 \times 1 = 0; \quad 1 \times 0 = 0; \quad 1 \times 1 = 1.$$

Assim como a **divisão** binária, em que:

$$0 \div 1 = 0; \quad 1 \div 1 = 1.$$

A operação de divisão por zero também não é permitida aqui.

Pesquise mais

Para saber mais sobre as operações aritméticas nos sistemas numéricos, leia a Seção 1.3 no livro *Eletrônica digital: teoria, componentes e aplicações,* de Szajnberg (2014), disponível na nossa biblioteca virtual em: <https://biblioteca-virtual.com/detalhes/parceiros/5> Acesso em: 24 set. 2017.

Sistemas digitais que representam dados numéricos, alfabéticos, alfanuméricos, enfim, todo tipo de dados, formam combinações de bits em grupos de 4, 6, 8, 16, 32... Um código é construído por uma combinação de bits em uma palavra binária prefixada. Alguns códigos são muito conhecidos em sistemas digitais, tais como:

No **código numérico BCD** (decimal codificado em binário, do inglês, *binary coded decimal*), cada dígito decimal é representado como um código binário. Por exemplo, os dígitos decimais, de 0 a 9, podem ser representados por um código binário de quatro bits. Esse tipo de código, BCD, é chamado de **código 8421**, a designação 8421 indica os pesos binários dos quatro bits (2^3, 2^2, 2^1, 2^0). A facilidade de conversão entre os números decimais em código 8421 é sua principal vantagem.

No **código Gray**, no entanto, os bits não têm peso, de modo que ele não é considerado um código aritmético. A característica que torna o código Gray importante é que nele há a mudança de um único bit entre dois números adjacentes. Essa propriedade é importante em muitas aplicações, como em codificadores de posição de eixo, em que a possibilidade de ocorrerem erros aumenta com o número de mudanças de bits entre números adjacentes em uma sequência. A Tabela 1.9 traz as codificações dos decimais de 0 a 15 em binário, Gray e BCD.

Tabela 1.9 | Códigos numéricos

Decimal	Binário	Gray	BCD
0	0000	0000	0000
1	0001	0001	0001
2	0010	0011	0010
3	0011	0010	0011
4	0100	0110	0100
5	0101	0111	0101
6	0110	0101	0110
7	0111	0100	0111
8	1000	1100	1000
9	1001	1101	1001
10	1010	1111	0001 0000
11	1011	1110	0001 0001
12	1100	1010	0001 0010
13	1101	1011	0001 0011
14	1110	1001	0001 0100
15	1111	1000	0001 0101

Fonte: elaborada pelo autor.

Segundo Floyd (2007), para nos comunicarmos, não usamos apenas números, mas também letras e outros símbolos. Os códigos alfanuméricos representam números e caracteres alfabéticos (letras). O código ASCII, pronunciado "askii", é a abreviação de *American Standard Code for Information Interchange* (Código Padrão Americano para Troca de Informações), é um código alfanumérico aceito universalmente e usado na maioria dos computadores e outros equipamentos eletrônicos. Em geral, os teclados de computadores são padronizados com o código ASCII. Quando digitamos uma letra, um número ou um comando de controle, o código ASCII correspondente é enviado para o computador.

O ASCII tem 128 caracteres e símbolos representados por um código de 7 bits. Na prática, o código ASCII é considerado um código de 8 bits com o bit mais significativo sempre 0. Esse código de 8 bits vai de 00 até 7F em hexadecimal.

Pesquise mais

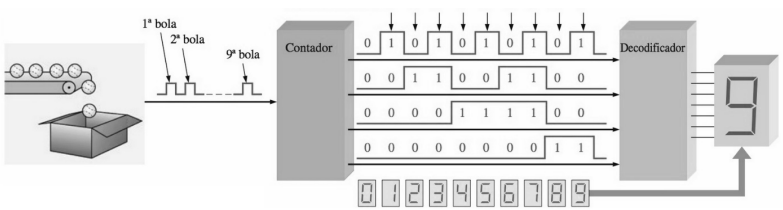
Para saber mais sobre códigos digitais, leia as Seções 2.10, 2.11 e 2.13 no livro *Sistemas digitais: fundamentos e aplicações,* de Floyd (2007), disponível na nossa biblioteca virtual: <https://biblioteca-virtual.com/detalhes/parceiros/5> Acesso em: 24 set. 2017.

Sem medo de errar

Caro aluno, seu momento chegou: seu chefe, impressionado com a sua curiosidade e empenho, convidou-lhe para participar da reunião com a equipe de desenvolvimento. O assunto da reunião é o desenvolvimento de um sistema de contagem da quantidade de bolas de tênis em cada caixa em uma linha de produção.

Depois de uma pesquisa sobre o assunto, você tem a oportunidade de apresentar sua ideia. Você propõe o uso de um sistema com um sensor de presença, um contador de quatro bits, um decodificador BCD – sete segmentos e um display de sete segmentos para apresentar a contagem. No seu sistema, há um contador de quatro bits que conta os pulsos de um sensor que detecta a passagem de uma bola e gera uma sequência de níveis lógicos em cada uma das suas quatro saídas paralelas. Cada conjunto de níveis lógicos representa um número binário de quatro bits (nível ALTO = 1 e nível BAIXO = 0), conforme indicado no esquema da Figura 1.18.

Figura 1.18 | Problema da contagem de bolas de tênis



Fonte: adaptada de Floyd (2007, p. 67).

À medida que o decodificador recebe essas formas de onda, ele decodifica cada conjunto de quatro bits, converte no número decimal correspondente e mostra em um display de sete segmentos. Quando o contador chega no estado binário 1001, é porque ele contou nove bolas de tênis: o display mostra o decimal 9 e uma nova caixa é posicionada sob o transportador. Então, o contador retorna ao estado zero (0000) e o processo começa novamente.

Você ressalta ainda que, caso se deseje aumentar o número de bolas em uma caixa, basta utilizar um sistema semelhante a esse para fazer a contagem das dezenas. A cada vez que o sistema (agora para contar as unidades) chega a 9 e é reiniciado, o sistema das dezenas é incrementado.

Seu chefe está muito feliz com o seu desempenho. Continue assim!

Avançando na prática

Controle de posição de um braço robótico

Descrição da situação-problema

É muito comum atualmente na indústria a aplicação de técnicas de automação com o objetivo de aumentar a sua eficiência e maximizar a produção, com um menor consumo de energia e de matérias-primas. O uso de robôs industriais já é uma realidade, como podemos ver na Figura 1.19.

Figura 1.19 | Robô industrial em uma linha de produção

Fonte: <https://commons.wikimedia.org/wiki/File:FANUC_R2000iB_AtWork.jpg>. Acesso em: 26 set. 2017.

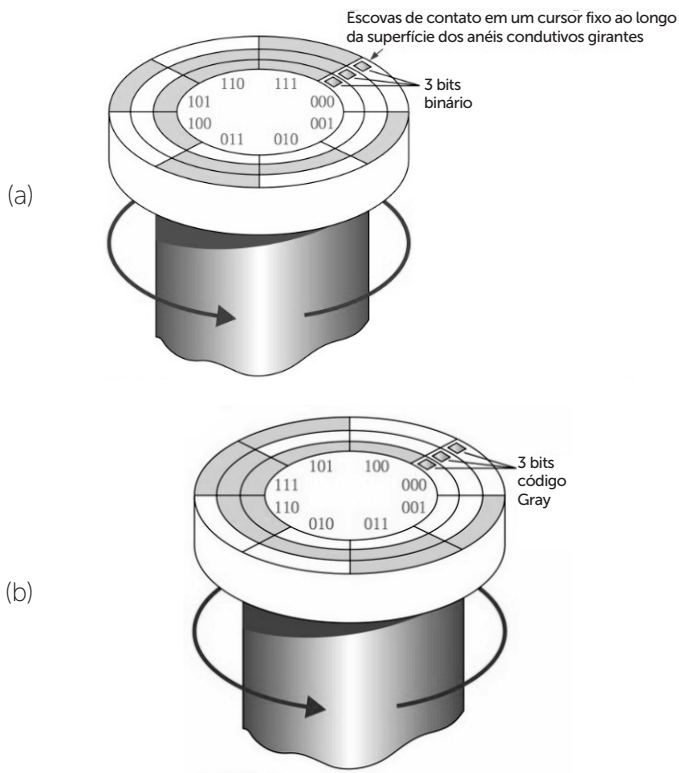
Você trabalha na equipe de manutenção do processo de montagem de uma grande montadora de carros. Na manutenção preventiva, você terá que propor a troca dos sensores de posição. Justifique sua decisão.

Resolução da situação-problema

O controle multieixos de um braço robótico é possível com a aplicação direta dos *encoders* ou codificador de posição de eixo. Atualmente, os braços da sua linha usam *encoders* com código binário. Para aumentar a precisão, sua proposta é trocá-los por novos *encoders* com código Gray.

Um diagrama simplificado de um mecanismo codificador de posição de eixo de 3 bits é mostrado na Figura 1.20. No *encoder*, existem três anéis segmentados em oito setores. Você explica que a precisão do posicionamento é proporcional à quantidade de setores, ou seja, quanto mais setores existirem, maior será a precisão do posicionamento, porém, apenas para fins de ilustração, está usando nesse exemplo um *encoder* com oito setores.

Figura 1.20 | *Encoder* de três bits (a) binário (b) Gray



Fonte: Floyd (2007, p. 105).

Cada setor de cada anel é fixado a uma tensão de nível ALTO ou a uma tensão de nível BAIXO para representar uns e zeros. 1 é indicado por um setor hachurado e 0, por um setor branco. À medida que o eixo gira no sentido anti-horário, os oito setores se movem sob três escovas produzindo uma saída binária de 3 bits indicando a posição do eixo.

A Figura 1.20(a) representa os *encoders* usados atualmente. Neles, os setores são organizados de forma a produzir um padrão binário direto, gerando na passagem das escovas a sequência 000, 001, 010, 011, e assim por diante. Quando as escovas estão nos setores coloridos, a saída é 1; e quando elas estão nos setores brancos, a saída é 0. Se uma escova estiver um pouco à frente das outras durante a transição de um setor para o próximo, podem ocorrer erros na saída. Considere o que acontece quando as escovas estão no setor 111 e entram no setor 000. Se a escova relativa ao bit mais significativo estiver um pouco adiantada, a posição 011 seria indicada incorretamente em vez da transição direta de 111 para 000. Nesse tipo de aplicação, é praticamente impossível manter um alinhamento mecânico preciso para todas as escovas. Portanto, alguns erros podem ocorrer em muitas das transições entre setores.

Sua proposta é a de substituir os atuais *encoders* para novos que utilizam o código Gray para eliminar problemas de erro que são inerentes ao código binário. Conforme podemos ver na Figura 1.20(b), o código Gray garante que apenas um bit mude entre setores adjacentes. Assim, mesmo que as escovas não tenham um alinhamento preciso, serão evitados erros na transição. Por exemplo, vamos considerar o mesmo caso anterior, quando as escovas estão no setor 111 e se movem para o próximo setor, 101. As duas únicas saídas possíveis durante a transição são 111 e 101, não importando como as escovas estão alinhadas. Uma situação similar ocorre na transição de cada um dos outros setores.

Faça valer a pena

1. O **sistema de numeração** consiste em um sistema em que um conjunto de números é representado por numerais de forma consistente.
Converta os números do sistema decimal para o sistema binário:

I. 13.
II. 356.
III. 0,54.
IV. 7,625.

Assinale a alternativa correta:
a) 1001; 110011011; 0,011101; 101,111.
b) 1101; 101101001; 110110; 111.
c) 1101; 101100100; 110110; 111.
d) 1101; 101100100; 0,10001010001; 111,101.
e) 1110; 1100100; 010001, 111,101.

2. O estudo das **operações aritméticas no sistema binário** é de extrema importância no âmbito da eletrônica digital e dos microcontroladores, em que operações são constantemente realizadas. Assim, saber como elas são realizadas possibilita a otimização e melhor aproveitamento dos componentes.
Efetue as seguintes operações aritméticas no sistema binário:
I. 10001 + 10,001.
II. 1100 + 111.
III. 1101 - 111.
IV. 11110 - 1100.
Assinale a opção correta:
a) 10011,001; 10011; 110; 10010.
b) 11011,100; 1011; 111; 11010.
c) 10011; 10011; 101; 11010.
d) 10101,010; 10011; 101; 10110.
e) 10011,001; 10011; 101; 10110.

3. Os sistemas digitais representam dados numéricos, alfabéticos, alfanuméricos, enfim, todo tipo de dados. Para isso códigos são construídos por uma combinação de bits em uma palavra binária prefixada.
Nesse contexto, avalie as afirmativas a seguir:
I. O código numérico BCD é uma forma de expressar cada dígito decimal com um código binário, em que cada dígito decimal, de 0 a 9, é representado por um código binário de quatro bits.
II. A designação 8421 indica os pesos binários dos quatro bits (2^8, 2^4, 2^2, 2^1), e a facilidade de conversão entre os números em código 8421 é a principal vantagem desse código.
III. A característica importante do código Gray é que ele apresenta uma mudança de um único bit quando se passa de uma palavra do código para a seguinte na sequência.
É correto o que se afirma em:
a) I, apenas.
b) I e III, apenas.
c) I e II, apenas.
d) II, apenas.
e) I, II e III.

Seção 1.3

Circuitos lógicos

Hugo Tanzarella Teixeira

Diálogo aberto

Na seção anterior, tratamos de diferentes sistemas numéricos, bem como a conversão entre esses sistemas. Tratamos também sobre a aritmética dos sistemas binários e dos códigos binários. Conhecer esses assuntos é muito importante quando lidamos com sistemas digitais. Além disso, um conhecimento de álgebra booleana é fundamental para o estudo e análise dos circuitos lógicos.

A álgebra de Boole tem como base o sistema de numeração binário, o que permite fazer operações lógicas e aritméticas usando apenas dois dígitos (ou dois estados). Por isso, a álgebra booleana é a ferramenta ideal para representar os circuitos eletrônicos digitais (portas lógicas), os números, caracteres e realizar operações lógicas e aritméticas nos sistemas digitais.

Sendo assim, na Unidade 2, nos aprofundaremos na álgebra booleana e nas técnicas de simplificação de circuitos lógicos, mas, antes disso, nesta seção trataremos de alguns assuntos fundamentais para iniciarmos nossos estudos. São eles: variáveis lógicas, tabela verdade, níveis lógicos e portas lógicas.

Assim, para pôr em prática todo esse aprendizado, você deve lembrar que trabalha em uma empresa de desenvolvimento de sistemas embarcados. Seu empenho em saber mais sobre os sistemas digitais chamou a atenção do seu chefe, que lhe convidou para participar de uma reunião da equipe de desenvolvimento. Você se preparou e participou ativamente da reunião, inclusive dando ideias. Um colega da equipe de desenvolvimento, notando sua curiosidade e criatividade, lhe lançou um desafio: criar um somador binário usando apenas portas lógicas.

Mas, o que são portas lógicas? Vamos descobrir?

Não pode faltar

Segundo Capuano (2014), é denominado de **circuito lógico** o arranjo de um pequeno grupo de circuitos básicos padronizados,

conhecidos como **portas lógicas**, que realizam **funções de lógica digital** dentro da eletrônica digital. Na prática, as portas lógicas são encontradas dentro de circuitos integrados comerciais específicos ou fazem parte da estrutura interna de dispositivos mais complexos, tais como microprocessadores, microcontroladores e outros circuitos integrados digitais.

Saiba mais

Em 1938, o engenheiro americano Claude Elwood Shannon utilizou as teorias da álgebra booleana para resolver problemas de circuitos de telefonia com relés, no seu trabalho ***Symbolic analysis of relay and switching***. Esse trabalho praticamente introduziu o campo da eletrônica digital na área tecnológica.

Uma **função lógica** admite uma ou mais entradas, mas apenas uma saída. As **variáveis lógicas**, normalmente representadas por letras, podem assumir apenas dois valores mutuamente excludentes, chamados **níveis lógicos**, e seu uso permite que se escrevam expressões algébricas, que podem ser manipuladas matematicamente dentro da álgebra booleana.

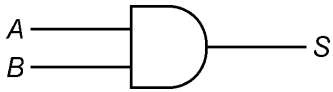
Exemplificando

Na eletrônica digital, é comum representarmos os níveis lógicos pelos dígitos binários **0** e **1**. O nível lógico 0 pode, por exemplo, representar a ausência de tensão ou uma chave aberta. Nesses casos, obrigatoriamente, o nível lógico 1 representará a presença de tensão, ou uma chave fechada, respectivamente.

Existem três funções lógicas básicas (AND, OR e NOT) e outras que são derivadas destas (NAND, NOR, XOR e XNOR). Nesta seção, estudaremos as funções lógicas juntamente com suas portas lógicas correspondentes.

Uma **porta AND** (ou E) de duas entradas é mostrada na Figura 1.22. A saída de uma porta AND é **1** *apenas* quando *todas* as entradas forem **1**. Quando qualquer uma das entradas for **0**, a saída será **0**. A Tabela 1.10 mostra uma tabela verdade da porta lógica AND de duas entradas, mas ela pode ser expandida para qualquer número de entradas.

Figura 1.22 | Porta lógica AND



Fonte: elaborada pelo autor.

Tabela 1.10 | Tabela verdade da porta AND

A	*B*	*S*
0	0	0
0	1	0
1	0	0
1	1	1

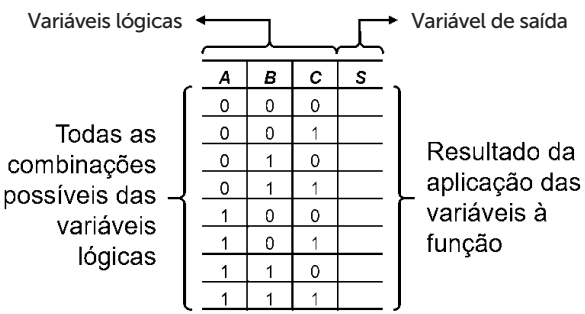
Fonte: elaborada pelo autor.

A porta AND é, portanto, usada para determinar quando certas condições são simultaneamente verdadeiras.

Assimile

A operação lógica de uma porta pode ser expressa com uma **tabela verdade** que apresenta uma coluna para cada entrada mais uma coluna para a saída. Nas linhas da tabela verdade são listadas todas as combinações de entrada com as saídas correspondentes de uma porta lógica, facilitando a representação e a análise delas. A Figura 1.21 mostra como uma tabela verdade é organizada.

Figura 1.21 | Como montar uma tabela verdade para três variáveis



Fonte: adaptada de Lourenço et al. (1997, p. 52).

A função lógica AND de duas variáveis é escrita como:

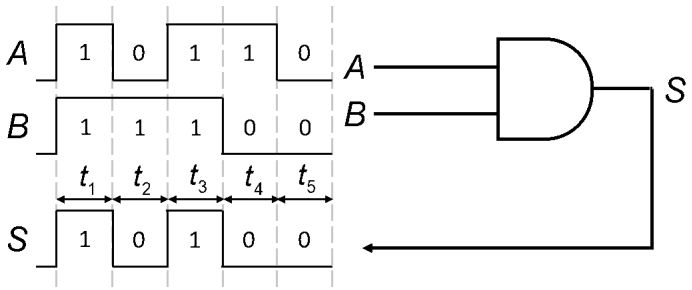
$$S = A \cdot B \text{ ou } S = AB. \quad (1.4)$$

Exemplificando

Nas aplicações práticas, em geral, as entradas de uma porta são formas de onda de tensão que variam frequentemente entre os níveis lógicos **1** e **0**. Vamos analisar, então, a operação das portas AND com formas de onda de pulsos nas entradas, tendo em mente que uma porta obedece à operação de uma tabela verdade independentemente se as entradas dela são níveis constantes ou níveis que variam entre 1 e 0.

Vamos examinar a operação com formas de onda nas entradas de uma porta AND, mostrada na Figura 1.23, observando as entradas uma relativa à outra para determinar o nível de saída num determinado instante.

Figura 1.23 | Exemplo de operação de uma porta AND

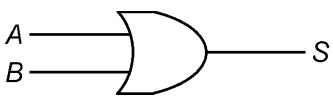


Fonte: adaptada de Floyd (2007, p. 135).

Ao lado da porta, montamos um diagrama de temporização que mostra as relações entre entradas e a saída. Durante os intervalos de tempo t_1 e t_3 ambas as entradas são 1, portanto, a saída também é 1 nesses intervalos. Durante os intervalos t_2, t_4 e t_5, pelo menos uma das entradas é 0 e, portanto, a saída é 0 nesses instantes.

Uma **porta OR** (ou OU) de duas entradas é mostrada na Figura 1.24. A **porta OR** (ou OU) produz uma saída **1** quando *qualquer* uma das entradas for 1. Apenas quando *todas* as entradas forem **0**, a saída será **0**. A Tabela 1.11 mostra a tabela verdade da porta lógica OR. A tabela verdade pode ser expandida para qualquer número de entradas.

Figura 1.24 | Porta lógica OR



Fonte: elaborada pelo autor.

Tabela 1.11 | Tabela verdade da porta OR

A	*B*	*S*
0	0	0
0	1	1
1	0	1
1	1	1

Fonte: elaborada pelo autor.

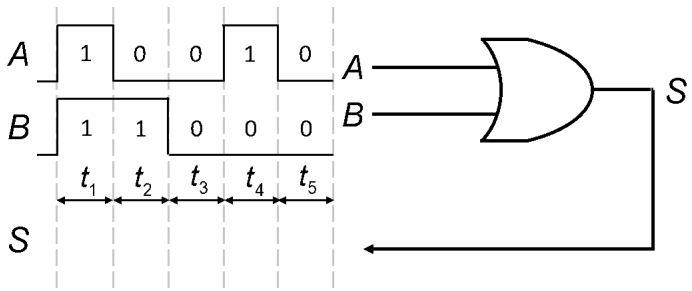
A função lógica OR de duas variáveis é representada matematicamente por:

$$S = A + B \qquad (1.5)$$

Faça você mesmo

Agora que você já conhece o princípio de funcionamento da porta OR, analise a operação de uma porta OR com as formas de onda digitais nas entradas mostradas na Figura 1.25 e desenhe a forma de onda da saída.

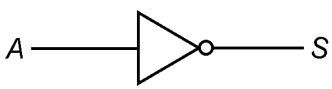
Figura 1.25 | Exemplo de operação de uma porta OR



Fonte: adaptada de Floyd (2007, p. 141).

A **porta NOT** (ou NÃO) realiza a operação denominada *inversão lógica*, e por isso pode ser chamada também de **porta inversora**. A porta NOT troca um nível lógico para o nível lógico oposto. Ou seja, em termos de bits, ele troca **1** por **0** e **0** por **1**. A porta NOT tem apenas uma entrada, e seu símbolo lógico é mostrado na Figura 1.26, enquanto que sua tabela verdade está montada na Tabela 1.12.

Figura 1.26 | Porta lógica NOT



Fonte: elaborada pelo autor.

Tabela 1.12 | Tabela verdade da porta NOT

A	*S*
0	1
1	0

Fonte: elaborada pelo autor.

A operação de um inversor pode ser expressa como a seguir:

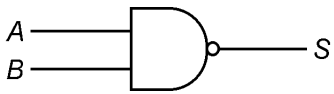
$$S = \overline{A} \text{ ou } S = A' \qquad (1.6)$$

A variável invertida pode ser lida como "A barra" ou "A negado".

A **porta NAND** (ou NÃO-E) é um elemento lógico muito popular e importante, pois é considerada uma porta universal, ou seja, as portas NAND podem ser usadas em diferentes combinações para realizarem operações básicas AND, OR ou NOT. O termo NAND é

uma contração da NOT-AND, portanto, essa porta funciona como uma porta AND com sua saída negada (invertida). O símbolo lógico dessa porta é, portanto, uma combinação entre as portas NOT e AND e está representado na Figura 1.27 (o pequeno círculo na saída indica a inversão do sinal). A saída da porta NAND é **0** apenas quando todas as entradas forem **1**, a Tabela 1.13 mostra uma tabela verdade para uma porta NAND de duas entradas.

Figura 1.27 | Porta lógica NAND



Fonte: elaborada pelo autor.

Tabela 1.13 | Tabela verdade da porta NAND

A	*B*	*S*
0	0	1
0	1	1
1	0	1
1	1	0

Fonte: elaborada pelo autor.

A função lógica para uma porta NAND de duas entradas é

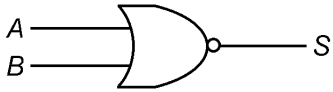
$$S = \overline{AB} \tag{1.7}$$

Reflita

É inerente à porta NAND o fato que uma ou mais entradas em nível lógico 0 produz uma saída em nível lógico 1. A partir dessa análise, é possível concluir que uma porta NAND pode ser usada para implementar de uma certa forma uma operação OR. Analise a Tabela 1.11 e Tabela 1.13 e conclua como isso é feito.

Assim como a porta NAND, a **porta NOR** (ou NÃO-OU) é um elemento lógico bastante útil, porque ela também é considerada uma porta universal e pode ser usada em determinadas combinações para realizar as operações básicas AND, OR e NOT. A propriedade universal das portas NAND e NOR será analisada melhor na Unidade 2 deste livro. O termo NOR é a contração de NOT e OR, e indica que essa porta funciona como uma porta OR com sua saída invertida. O símbolo lógico padrão para essa porta está representado na Figura 1.28. Para uma porta NOR, a saída será nível 0 quando pelo menos uma das entradas estiver em nível lógico 1, como pode ser visto na sua tabela verdade na Tabela 1.14.

Figura 1.28 | Porta lógica NOR



Fonte: elaborada pelo autor.

Tabela 1.14 | Tabela verdade da porta NOR

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

Fonte: elaborada pelo autor.

A função lógica para uma porta NOR de duas entradas é

$$S = \overline{A + B} \qquad (1.8)$$

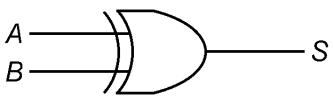
Reflita

Na porta NOR, assim como na NAND, há um aspecto de sua operação que é inerente à forma como ela funciona logicamente. Na porta NOR, um nível lógico 1 é produzido na saída apenas quando todas as entradas estiverem no nível 0. A partir dessa análise, é possível concluir que uma porta NOR pode ser usada para implementar de uma certa forma uma operação AND. Analise a Tabela 1.10 e Tabela 1.14 e conclua como isso é feito.

As portas XOR (ou OU exclusivo) e XNOR (ou NÃO-OU exclusivo) são formadas pela combinação de algumas das portas já estudadas, mas, devido à sua importância em diversas aplicações, elas são tratadas como um elemento básico, e, por isso, têm seu próprio símbolo lógico (FLOYD, 2007).

A saída de uma **porta XOR** é o nível lógico **1** *apenas* quando as duas entradas estão em níveis lógicos opostos. A operação de uma porta XOR está resumida na tabela verdade mostrada na Tabela 1.15. O símbolo lógico padrão para essa porta está representado na Figura 1.29.

Figura 1.29 | Porta lógica XOR



Fonte: elaborada pelo autor.

Tabela 1.15 | Tabela verdade da porta XOR

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

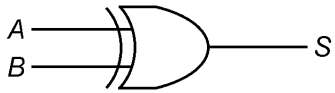
Fonte: elaborada pelo autor.

A função lógica para uma porta XOR de 2 entradas é

$$S = A \oplus B \quad (1.9)$$

A **porta XNOR** funciona de maneira oposta à porta XOR, ou seja, sua saída terá nível lógico **1** quando todas as entradas estiverem no mesmo nível lógico, como podemos ver na Tabela 1.16. O símbolo lógico padrão para essa porta está representado na Figura 1.30.

Figura 1.30 | Porta lógica XNOR



Fonte: elaborado pelo autor.

Tabela 1.16 | Tabela verdade da porta XNOR

A	*B*	*S*
0	0	1
0	1	0
1	0	0
1	1	1

Fonte: elaborada pelo autor.

A função lógica para uma porta XNOR de duas entradas é

$$S = \overline{A \oplus B} \quad (1.10)$$

É através da utilização das portas lógicas que podemos implementar todas as expressões geradas pela álgebra de Boole, que é um estudo onde estão todas as leis fundamentais da lógica digital, mas isso é assunto para a próxima unidade.

Pesquise mais

Usamos os símbolos característicos de acordo com o Padrão 91-1984 (IEEE, 1984) para representar as portas lógicas. Existem outras maneiras gráficas de representá-las. Para conhecer mais algumas, leia o capítulo 3 do livro *Sistemas digitais: fundamentos e aplicações*, de Floyd (2007), disponível na nossa biblioteca virtual: <https://biblioteca-virtual.com/detalhes/parceiros/5>. Acesso em: 16 out. 2017.

Sem medo de errar

Agora que já está familiarizado com as portas lógicas, você se dedica a resolver o desafio lançado pelo seu colega da equipe de desenvolvimento. Ele sugeriu que você montasse um circuito lógico que realizasse uma operação de soma binária.

Você já sabe como a aritmética binária funciona. Seu circuito terá duas entradas, uma para cada parcela da soma, e duas saídas, uma para o bit que representa o resultado da soma e outra para o bit que

representa o "vai um". A sua abordagem para resolver o problema começa por escrever uma tabela verdade para a soma binária. Na Tabela 1.17, *A* e *B* são as entradas; *S* é a saída que representa a soma; e *C* é a saída que representa o "vai um".

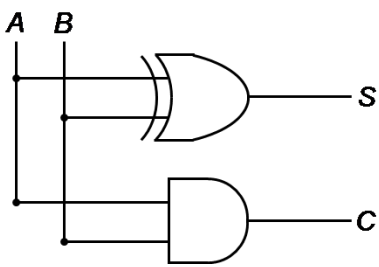
Tabela 1.17 | Tabela verdade da soma binária

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

Fonte: elaborada pelo autor.

Depois de estudar a fundo as portas lógicas, você percebe que o problema, na verdade, é bastante simples, e foi possível resolvê-lo usando apenas duas portas lógicas. Você notou que a tabela verdade da porta lógica XOR (Tabela 1.15) é exatamente igual à tabela verdade para a saída S, e que a tabela verdade para a porta AND (Tabela 1.10) é o que você precisa pra montar o circuito para o "vai um". Agora, você monta o esquemático do circuito lógico (Figura 1.31).

Figura 1.31 | Circuito lógico para o somador binário



Fonte: elaborada pelo autor.

Depois, você escreve as funções lógicas para cada saída:

$$S = A \oplus B \quad (1.11)$$

$$C = AB \quad (1.12)$$

Todo empolgado, você apresenta sua solução para o seu colega e ele o parabeniza, mas faz uma ressalva: esse é um "meio somador binário". Você conseguiria desenvolver um somador completo?

Continue sua jornada em busca de conhecimento. Faça mais pesquisas, estude mais e discuta com seus colegas sobre esse novo problema que lhe foi apresentado.

Avançando na prática

Sensor de porta e janela aberta

Descrição da situação-problema

Você trabalha em uma empresa de alarmes e precisa instalar um alarme em uma pequena loja comercial. O ambiente em questão consiste em duas janelas e uma porta. Seu sistema de alarme, quando ligado, deve disparar quando uma das janelas ou a porta for aberta.

Proponha um sistema baseado em portas lógicas para o alarme dessa loja.

Resolução da situação-problema

Primeiro, você precisa instalar os sensores em cada uma das janelas e na porta. Você optou pelo uso de chaves magnéticas (Figura 1.32), baseadas em efeito Hall. Elas produzem uma saída em nível ALTO quando abertas e em nível BAIXO quando fechadas. Você já possui um módulo para o alarme, que é ativado com nível ALTO.

Figura 1.32 | Chave magnética de efeito Hall

Fonte: <http://www.oneproject.com.br/media/user/images/original/sensoresg_s7.png>. Acesso em: 29 set. 2017.

Para montar o seu circuito, você precisa construir uma tabela verdade para ele. A Tabela 1.18 mostra tal feito. Nela, *S* é a saída que indica a necessidade de disparar o alarme e as entradas *P, J1* e *J2* se referem à porta e a cada uma das janelas, respectivamente.

Tabela 1.18 | Tabela verdade do sistema de alarme

P	*J1*	*J2*	*S*
BAIXO	BAIXO	BAIXO	BAIXO
BAIXO	BAIXO	ALTO	ALTO
BAIXO	ALTO	BAIXO	ALTO
BAIXO	ALTO	ALTO	ALTO
ALTO	BAIXO	BAIXO	ALTO
ALTO	BAIXO	ALTO	ALTO
ALTO	ALTO	BAIXO	ALTO
ALTO	ALTO	ALTO	ALTO

Fonte: elaborada pelo autor.

Observando a Tabela 1.18 fica bem claro que uma porta OR de três entradas seria suficiente para montar o seu circuito. Agora, é preciso criar uma parte do circuito para habilitar o alarme somente quando conveniente, afinal, o alarme não pode ficar ativado o tempo todo. A tabela verdade para essa parte do circuito está montada na Tabela 1.19. As entradas são *H*, que é o sinal que habilita o circuito; e *S*, que é a saída do módulo anterior. A saída *C* irá, de fato, disparar o alarme, quando em nível ALTO.

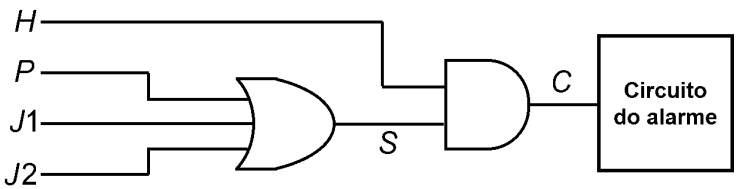
Tabela 1.19 | Tabela verdade do sistema que habilita o alarme

H	*S*	*C*
BAIXO	BAIXO	BAIXO
BAIXO	ALTO	BAIXO
ALTO	BAIXO	BAIXO
ALTO	ALTO	ALTO

Fonte: elaborada pelo autor.

Observando a Tabela 1.19, você logo percebe que essa etapa do circuito é facilmente obtida com uma porta AND de duas entradas. Agora, só resta montar o esquema do circuito, que pode ser visto na Figura 1.33.

Figura 1.33 | Esquema simplificado do sistema de alarme



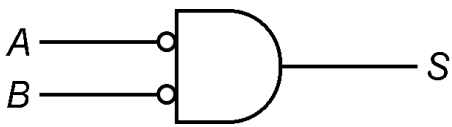
Fonte: elaborada pelo autor.

Para finalizar, fique com um questionamento para reflexão: se você somente tivesse acesso a portas lógicas OR de duas entradas, como montaria esse circuito?

Faça valer a pena

1. A operação lógica de uma porta pode ser expressa com uma tabela verdade que apresenta uma coluna para cada entrada mais uma coluna para a saída.
Monte a tabela verdade para o circuito da Figura 1.34, lembrando que o pequeno círculo em cada entrada indica a inversão do sinal.

Figura 1.34 | Circuito lógico



Fonte: elaborada pelo autor.

a)

A	*B*	*S*
0	0	1
0	1	0
1	0	0
1	1	0

b)

A	*B*	*S*
0	0	1
0	1	0
1	0	0
1	1	1

c)

A	*B*	*S*
0	0	0
0	1	0
1	0	0
1	1	1

d)

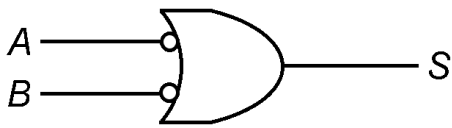
A	*B*	*S*
0	0	1
0	1	1
1	0	1
1	1	0

e)

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

2. É possível que alguns circuitos lógicos sejam equivalentes. Uma das maneiras para provar essa equivalência é comparando suas tabelas verdades: se elas forem iguais, os circuitos são equivalentes.
O circuito lógico da Figura 1.35 é equivalente a qual porta lógica? Lembrando que o pequeno círculo em cada entrada indica a inversão do sinal.

Figura 1.35 | Circuito lógico

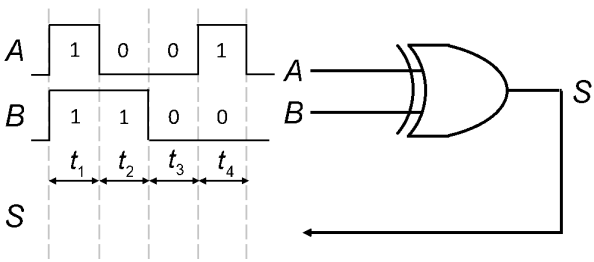


Fonte: elaborada pelo autor.

a) AND.
b) OR.
c) XOR.
d) NOR.
e) NAND.

3. Na maioria das aplicações, as entradas de uma porta não apresentam níveis estacionários, e sim formas de onda de tensão que variam frequentemente entre os níveis lógicos 1 e 0.
Analise a operação de uma porta XOR com as formas de onda digitais nas entradas mostradas na Figura 1.36 e assinale a alternativa que possui a sequência de níveis lógicos na ordem t_1, t_2, t_3 e t_4.

Figura 1.36 | Exemplo de operação de uma porta XOR



Fonte: elaborada pelo autor.

a) 0100.
b) 1110.
c) 1001.
d) 0101.
e) 1010.

Referências

BELL Labs. 1956 Nobel Prize in Physics: The Transistor. **Nokia Bell Labs**, [s.d.]. Disponível em: <https://www.bell-labs.com/our-people/recognition/1956-transistor/>. Acesso em: 25 set. 2017.

CAPUANO, Francisco Gabriel. **Sistemas digitais**: circuitos combinacionais e sequenciais. São Paulo: Érica, 2014. 144 p.

FLOYD, Thomas. **Sistemas digitais**: fundamentos e aplicações. 9. ed. São Paulo: Bookman, 2007. 888 p.

IEEE STANDARTS. IEEE Standard Graphic Symbols for Logic Functions. **IEEE Xplore**, p. 1-160, 1984.

LOURENÇO, Antônio Carlos et al. **Circuitos digitais**: estude e use. São Paulo: Érica, 1997. 336 p.

MACNEIL, Jessica. **First successful test of the transistor**: December 16, 1947. EDN Network Blog, 16 dez. 2016. Disponível em: <http://www.edn.com/electronics-blogs/edn-moments/4426086/1st-successful-test-of-the-transistor--December-16--1947>. Acesso em: 25 set. 2017.

MIT Technology Review. 10 Breakthorough Tecnologies 2017, **MIT**, mar./abr. 2017. Disponível em: <https://www.technologyreview.com/lists/technologies/2017/> Acesso em: 18 set. 2017.

NATIONAL Semiconductor. 54LS03/DM54LS03/DM74LS03 Quad 2-Input NAND Gates with Open-Collector Outputs datasheet. **National Semiconductor Corparation**, 1989. 6 p. Disponível em: <http://rtellason.com/chipdata/dm74ls03.pdf>. Acesso em: 18 out. 2017.

SHANNON, Claude. **Symbolic analysis of relays and switching circuits**. Dissertação de mestrado. Boston: Massachusetts Institute of Technology (MIT), 1937. 69 p.

SZAJNBERG, Mordka. **Eletrônica digital**: teoria, componentes e aplicações. Rio de Janeiro: LTC, 2014. 455 p.

TOCCI, Ronald J. **Sistemas digitais**: princípios e aplicações. 11. ed. São Paulo: Pearson, 2011. 840 p.

VAHID, Frank. **Sistemas digitais**: projeto, otimização e HDLs. 1. ed. Porto Alegre: Bookman, 2008. 560 p.

Álgebra booleana e simplificação de circuitos lógicos

Convite ao estudo

Vimos, até aqui que os circuitos lógicos executam expressões lógicas. Na Unidade 1, determinamos esses circuitos através de expressões características extraídas de tabelas verdade. Os circuitos gerados por esse processo, apesar de corretos, geralmente admitem simplificações e, consequentemente, diminuição de blocos lógicos utilizados na prática.

Para iniciarmos o estudo de simplificação de circuitos lógicos, é preciso conhecer razoavelmente a álgebra de Boole, e através das suas leis, teoremas e axiomas efetuaremos as mencionadas simplificações. Além disso, é importante destacar que a teoria de sistemas digitais é fundamentada na álgebra de Boole.

Vamos imaginar o seguinte contexto: você e seus amigos acabaram de se formar e decidem abrir uma empresa de segurança automotiva. Vocês sabem que para conquistar o mercado terão que ser criativos e, sobretudo, econômicos em seus projetos. Você está pronto para esse desafio?

Nesta unidade, para auxiliá-lo, apresentaremos os conceitos básicos da álgebra de Boole. Além disso, por meio de manipulação algébrica é permitida a simplificação de expressões e circuitos lógicos derivados. Outra forma de simplificação mais usual também será estudada, trata-se do uso dos mapas de Karnaugh, que, na essência, utilizam os conceitos de Boole, porém, arranjados por um método gráfico que permite a simplificação de expressões e tabelas verdade de maneira imediata e menos trabalhosa.

E então, preparado para aprender um pouco mais sobre esse universo digital?

Mãos à obra e um ótimo estudo!

Seção 2.1

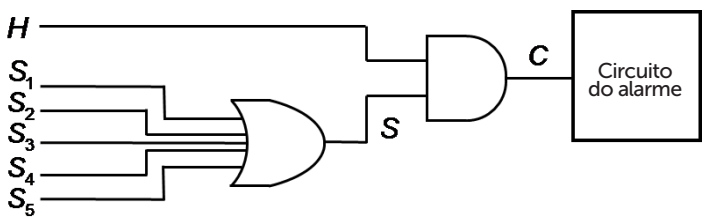
Leis e teoremas da álgebra booleana

Diálogo aberto

A álgebra booleana é a matemática dos sistemas digitais, conhecê-la é fundamental para a análise e síntese de circuitos lógicos. No fim da unidade anterior tivemos um primeiro contato com as operações booleanas através de suas relações com as portas NOT, AND, OR, NAND e NOR.

Nesta seção, aprenderemos um pouco mais sobre como aplicar e simplificar os circuitos lógicos que compõem os sistemas digitais. Para isso, retomaremos o contexto apresentado no início da unidade: você acabou de fundar uma empresa de segurança automotiva com seus amigos. Atualmente, vocês estão se dedicando a um novo projeto de alarme para carros e se depararam com a seguinte situação: inicialmente, o projeto previa cinco sensores distribuídos pelo carro (entre portas, janelas e capôs) e a entrada que habilita o alarme (botão de liga e desliga). O primeiro circuito lógico projetado por vocês consistia em uma porta OR de cinco entradas e uma porta AND de duas entradas, como pode ser visto na Figura 2.1.

Figura 2.1 | Primeiro projeto do circuito lógico para o alarme



Fonte: elaborada pelo autor.

O problema é que no estoque de peças só restaram CIs do tipo SN74HC00. Como lidar com essa situação?

Para ajudá-lo a resolver esse impasse, nesta seção conheceremos as principais leis e teoremas da álgebra booleana, bem como o teorema de DeMorgan. Fique atento, esse conhecimento será muito importante para novos projetos de circuitos lógicos.

Não pode faltar

Em 1854, no trabalho *Investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities* (em português, "Uma investigação das leis do pensamento, sobre as quais são fundadas as teorias matemáticas de lógica e probabilidades"), Georg Boole fundamentou a "álgebra lógica", hoje conhecida como **álgebra booleana**.

Segundo Floyd (2007), a álgebra booleana traz uma forma conveniente de expressar e analisar a operação de circuitos lógicos de forma sistemática. Antes de entrarmos no assunto, devemos nos familiarizar com os termos *variável*, *complemento* e *literal*.

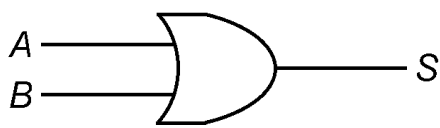
Uma **variável** é um símbolo (geralmente uma letra maiúscula em itálico) usado para representar uma grandeza lógica. Uma variável booleana simples admite apenas dois valores, em circuitos lógicos: 1 ou 0. O **complemento** é o inverso de uma variável e é indicado por uma barra sobre a variável. Por exemplo, o complemento da variável A é $\overline{A}$. Se $A = 1$. Então, $\overline{A} = 0$; e se $A = 0$, então, $\overline{A} = 1$. Como vimos, o complemento de uma variável *A* é lido como "A negado" ou "A barrado". Pode ser usado, também, para representar um complemento outro símbolo. Por exemplo, B' indica o complemento de *B*. Uma **literal** é a variável ou o complemento de uma variável.

Tradicionalmente, utilizamos a álgebra booleana para simplificarmos os circuitos lógicos digitais. Os métodos que veremos para simplificação e projetos de circuitos lógicos requerem que a expressão esteja na forma de soma-de-produtos. Sendo assim, é fundamental aprendermos um pouco mais sobre as operações de adição e multiplicação booleana.

A **adição booleana** é equivalente à operação OR (Figura 2.2), e as regras básicas são ilustradas com suas relações da seguinte forma:

Figura 2.2 | Porta OR

$A + B = S$
$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 1$



Fonte: elaborada pelo autor.

Na álgebra booleana, um **termo soma** é uma soma de literais. Em circuitos lógicos, um termo soma é produzido por uma operação OR sem o envolvimento de operações AND, por exemplo: $A+B$, $A+\bar{B}$, $A+B+\bar{C}$ e $\bar{A}+B+C+\bar{D}$.

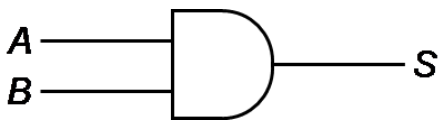
Assimile

Um termo soma será igual a 1 quando uma ou mais das literais no termo for 1. Um termo soma será igual a 0 somente se cada uma das literais for 0.

A **multiplicação booleana** é equivalente à operação AND (Figura 2.3), e as regras básicas são ilustradas com suas relações da seguinte forma:

Figura 2.3 | Porta AND

A . B = S
0 . 0 = 0
0 . 1 = 0
1 . 0 = 0
1 . 1 = 1



Fonte: elaborada pelo autor.

Na álgebra booleana, um **termo produto** é o produto de literais. Em circuitos lógicos, um termo produto é produzido por uma operação AND sem o envolvimento de operações OR, por exemplo: AB, $A\bar{B}$, ABC e $\bar{A}BC\bar{D}$.

Assimile

Um termo produto será igual a 1 apenas se cada uma das literais no termo for 1. Um termo produto será igual a 0 quando uma ou mais literais for 0.

Começaremos o nosso estudo da álgebra boolena pelas suas leis básicas. Elas são as mesmas que para a álgebra comum e estão resumidas no Quadro 2. 1.

Quadro 2.1 | Leis da álgebra booleana

Lei	Enunciado	Representação
Comutativa da adição	A ordem das variáveis na qual a função OR é aplicada não faz diferença.	$A+B=B+A$
Comutativa da multiplicação	A ordem das variáveis na qual a operação AND é aplicada não faz diferença.	$AB=BA$
Associativa da adição	Quando uma operação OR é aplicada em mais de duas variáveis, o resultado é o mesmo, independentemente da forma de agrupar as variáveis.	$A+(B+C)=(A+B)+C$
Associativa da multiplicação	Quando uma operação AND é aplicada em mais de duas variáveis, o resultado é o mesmo, independente da forma de agrupar as variáveis.	$A(BC)=(AB)C$
Distributiva	Uma operação AND entre uma única variável com o resultado de uma operação OR aplicada a duas ou mais variáveis é equivalente a uma operação OR entre os resultados das operações AND entre uma única variável e cada uma das duas ou mais variáveis.	$A(B+C)=AB+AC$

Fonte: adaptado de Floyd (2007, p. 202-203).

A álgebra booleana conta com uma série de axiomas e teoremas que são importantes na manipulação e simplificação de expressões booleanas. Os axiomas são proposições que não têm demonstração, por serem considerados evidentes. Os axiomas da álgebra booleana são:

1. **Fechamento**: dado o conjunto de valores binários, $C=\{0,1\}$,

$$A,B\in C\Rightarrow\begin{cases}A+B\in C\\A\cdot B\in C\end{cases} \qquad (2.1)$$

Ou seja, as operações OR e AND resultam em valores que também são binários.

2. **Identidade**: dado o conjunto $C=\{0,1\}$,

$$A\in C\Rightarrow\begin{cases}A+0=A\\A\cdot 1=A\end{cases} \qquad (2.2)$$

Esse axioma introduz os elementos neutros das operações OR e AND.

Existem ainda os axiomas da **comutatividade**, **distributividade** e **complemento**, que foram apresentados no Quadro 2.1 como leis da álgebra booleana.

Os teoremas, por sua vez, são afirmações que necessitam de uma prova, ou seja, todo teorema está associado a uma sequência lógica, que leva à sua conclusão. Como não estamos interessados no estudo matemático da álgebra booleana, e sim na sua aplicação aos sistemas digitais, não apresentaremos as provas dos teoremas a seguir.

1. Indepotência

$$\begin{cases} A + A = A \\ A \cdot A = A \end{cases} \quad (2.3)$$

2. Aniquilação

$$\begin{cases} A + 1 = 1 \\ A \cdot 0 = 0 \end{cases} \quad (2.4)$$

3. Dupla negação

$$\bar{\bar{A}} = A \quad (2.5)$$

4. DeMorgan

$$\begin{cases} \overline{A + B} = \bar{A} \cdot \bar{B} \\ \overline{A \cdot B} = \bar{A} + \bar{B} \end{cases} \quad (2.6)$$

Além desses teoremas, existe ainda o teorema da **associatividade**, que já foi apresentado no Quadro 2.1 como duas das leis da álgebra booleana. Para facilitar os seus estudos, esses axiomas e teoremas bem como algumas combinações importantes deles estão compiladas no Quadro 2. 2.

Quadro 2.2 | Regras da álgebra booleana

1.	$A + 0 = A$	7.	$A \cdot A = A$
2.	$A + 1 = 1$	8.	$A \cdot \bar{A} = 0$
3.	$A \cdot 0 = 0$	9.	$\bar{\bar{A}} = A$
4.	$A \cdot 1 = A$	10.	$A + AB = A$
5.	$A + A = A$	11.	$A + \bar{A}B = A + B$
6.	$A + \bar{A} = 1$	12.	$(A + B)(A + C) = A + BC$

Fonte: Floyd (2007, p. 203).

Exemplificando

A regra 10 do Quadro 2.2 pode ser provada aplicando-se a lei distributiva e as regras 2 e 4, como feito a seguir:

$$\begin{aligned} A + B &= A(1 + B) && \text{Fatorando (lei distributiva)} \\ &= A \cdot 1 && \text{Regra 2: } (1 + B) = 1 \\ &= A && \text{Regra 4: } A \cdot 1 = A \end{aligned}$$

É possível, ainda, demonstrar a prova em termos da tabela verdade na Tabela 2.1.

Tabela 2.1 | Tabela verdade da regra 10

A	B	$A \cdot B$	$A + A \cdot B$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Fonte: elaborada pelo autor.

A prova das regras 11 e 12 fica como exercício para que você possa exercitar suas habilidades em álgebra booleana.

DeMorgan foi um matemático que conheceu Boole e propôs dois teoremas que representam uma parte importante na álgebra booleana (FLOYD, 2007). Devido à sua importância nos circuitos digitais, daremos uma atenção a mais aos seus teoremas. Em termos práticos, os teoremas de DeMorgan provêm uma verificação de equivalências entre as portas NAND e OR negativa e as equivalências entre as portas NOR e AND negativa, já discutidas na Seção 1.3.

Um dos teoremas de DeMorgan afirma que:

O complemento de um produto de variáveis é igual à soma dos complementos das variáveis.

Ou, em termos de operações lógicas:

O complemento de duas ou mais variáveis submetidas a uma operação AND é equivalente a uma operação OR entre os complementos das variáveis individuais.

Podemos expressar esse teorema para duas variáveis com a expressão

$$\overline{A \cdot B} = \overline{A} + \overline{B}. \qquad (2.7)$$

O segundo teorema de DeMorgan afirma que:

O complemento de uma soma de variáveis é igual ao produto do complemento das variáveis.

Ou, em termos de operações lógicas:

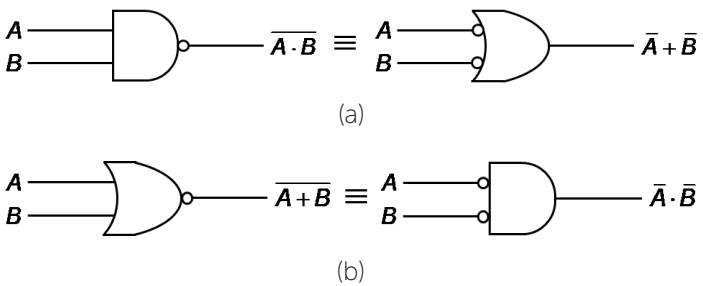
O complemento de duas ou mais variáveis submetidas a uma operação OR é equivalente a uma operação AND entre os complementos das variáveis individuais.

A fórmula para a expressão desse teorema para duas variáveis é:

$$\overline{A+B} = \bar{A} \cdot \bar{B}. \qquad (2.8)$$

A Figura 2.4(a) mostra a equivalência de portas para a Equação 2.7. A Figura 2.4(b) mostra a equivalência de portas para a Equação 2.8, e suas tabelas verdade estão escritas na Tabela 2.2.

Figura 2.4 | Equivalência entre portas lógicas (a) NAND e OR negativa; (b) NOR e AND negativa



Fonte: elaborada pelo autor.

Tabela 2.2 | Tabela verdade (a) NAND e OR negativa; (b) NOR e AND negativa

A	B	$\overline{A \cdot B}$	$\bar{A} + \bar{B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

(a)

A	B	$\overline{A+B}$	$\bar{A} \cdot \bar{B}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

(b)

Fonte: elaborada pelo autor.

É importante salientar que os teoremas de DeMorgan também se aplicam a expressões nas quais existem mais do que duas variáveis, por exemplo:

$$\overline{ABC} = \bar{A} + \bar{B} + \bar{C},$$

e

$$\overline{A+B+C}=\bar{A}\bar{B}\bar{C}$$

Outro aspecto importante é que cada variável nos teoremas de DeMorgan podem representar uma combinação de outras variáveis. Por exemplo, *A* na Equação 2.7 pode ser igual ao termo $XY+Z$, e *B* pode ser igual ao termo $X+YZ$. Assim, aplicando o teorema de DeMorgan para a expressão $\overline{(XY+Z)(X+YZ)}$, obtemos o seguinte resultado:

$$\overline{(XY+Z)(X+YZ)}=\overline{(XY+Z)}+\overline{(X+YZ)} \quad (2.9)$$

Ainda podemos aplicar (2.8) no resultado, de modo que

$$\overline{(XY+Z)}+\overline{(X+YZ)}=(\overline{XY})\bar{Z}+\bar{X}(\overline{YZ}) \quad (2.10)$$

Podemos, ainda, aplicar o teorema de DeMorgan aos termos $\overline{XY}$ e $\overline{YZ}$, o que resulta na seguinte expressão:

$$(\overline{XY})\bar{Z}+\bar{X}(\overline{YZ})=(\bar{X}+\bar{Y})\bar{Z}+\bar{X}(\bar{Y}+\bar{Z}) \quad (2.11)$$

A Expressão 2.11 ainda pode ser simplificada usando-se as leis e regras da álgebra booleana, mas não é mais possível aplicar o teorema de DeMorgan.

Reflita

Em certo momento desta seção, destacamos a importância dos teoremas de DeMorgan para a análise e projeto de circuitos lógicos. Mas a que se deve tamanha importância?

Pesquise mais

A Seção 4.3 do livro *Sistemas digitais: fundamentos e aplicações* (FLOYD, 2007), disponível na nossa biblioteca virtual, traz diversos exemplos de aplicação do teorema de DeMorgan. Consulte e se aprofunde nesse conteúdo.

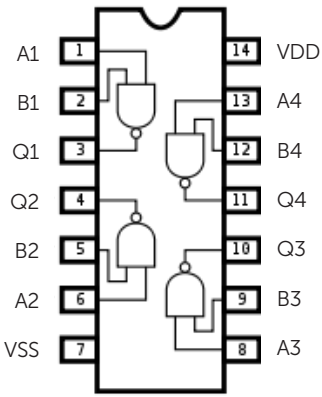
FLOYD, Thomas L. **Sistemas digitais**: fundamentos e aplicações. Tradução de José Lucimar do Nascimento. 9. ed. Porto Alegre : Bookman, 2007.

Sem medo de errar

Você e seus amigos acabaram de abrir uma empresa de segurança automotiva e, naturalmente, já se depararam com um problema: o circuito que projetaram previamente é composto de portas AND e OR e, para implementar esse circuito vocês só contam com o CI SN74HC00.

O seu primeiro passo é saber como esse CI funciona. Para isso, você pesquisa o seu *datasheet* e lembra que se trata de um encapsulamento com quatro portas lógicas NAND de duas portas, como pode ser visto na Figura 2.5.

Figura 2.5 | Configuração do CI SN74HC00



Fonte: adaptada de Texas Instruments (2016, p. 3).

Portanto, agora você sabe que o seu problema consiste em transformar o projeto inicial num equivalente usando apenas portas NAND.

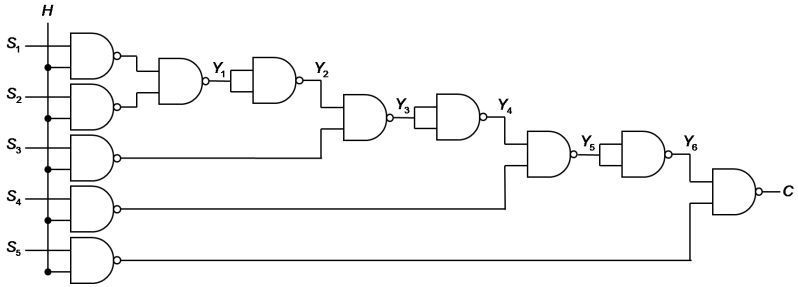
Da Figura 2.1 você sabe que $C = S \cdot H$ e que $S = S_1 + S_2 + S_3 + S_4 + S_5$, portanto,

$$C = (S_1 + S_2 + S_3 + S_4 + S_5)H,$$

$$C = HS_1 + HS_2 + HS_3 + HS_4 + HS_5.$$

Pelo teorema de DeMorgan, você sabe que $\overline{AB} = \bar{A} + \bar{B}$, que é o mesmo que dizer que a porta NAND é equivalente à porta OR negativa. Você sabe também que $\overline{AA} = \bar{A}$. Com essas possibilidades, você tem uma ideia para um arranjo utilizando apenas as portas NAND em cascata, como pode ser visto na Figura 2.6, que você apresentou para os seus amigos.

Figura 2.6 | Projeto para o circuito lógico do alarme apenas comportas NAND



Fonte: elaborada pelo autor.

Para entender melhor o funcionamento desse circuito, você o separou ele em partes. Assim,

$$Y_1 = \overline{\overline{HS_1} \cdot \overline{HS_2}} = \overline{\overline{HS_1}} + \overline{\overline{HS_2}} = HS_1 + HS_2 .$$

$$Y_2 = \overline{Y_1} = \overline{HS_1 + HS_2}.$$

$$Y_3 = \overline{\overline{Y_1} \cdot \overline{HS_3}} = \overline{\overline{Y_1}} + \overline{\overline{HS_3}} = HS_1 + HS_2 + HS_3 .$$

$$Y_4 = \overline{Y_3} = \overline{HS_1 + HS_2 + HS_3} .$$

$$Y_5 = \overline{\overline{Y_3} \cdot \overline{HS_3}} = \overline{\overline{Y_3}} + \overline{\overline{HS_4}} = HS_1 + HS_2 + HS_3 + HS_4 .$$

$$Y_6 = \overline{Y_5} = \overline{HS_1 + HS_2 + HS_3 + HS_4} .$$

$$C = \overline{\overline{Y_4} \cdot \overline{HS_5}} = \overline{\overline{Y_4}} + \overline{\overline{HS_5}} = HS_1 + HS_2 + HS_3 + HS_4 + HS_5 .$$

Dessa forma, foi possível construir um circuito equivalente ao primeiro projeto utilizando apenas portas NAND, usando três Cis SN74HC00. Lembre-se de que, em alguns momentos, ser mais econômico não significa usar o menor número de material possível, mas sim usar o que você tem disponível de maneira inteligente.

Faça valer a pena

1. Na álgebra booleana, um termo produto é o produto de literais. Em circuitos lógicos, um **termo produto** é produzido por uma operação AND sem o envolvimento de operações OR.

Determine os valores de A, B, C e D que tornam o termo produto $\bar{A}BC\bar{D}$ igual a 1:

a) 1001.
b) 0110.
c) 1111.
d) 0000.
e) 1010.

2. DeMorgan propôs dois teoremas que representam uma parte importante na álgebra booleana que, em termos práticos, provêm uma verificação de equivalências entre as portas NAND e OR negativa e as equivalências entre as portas NOR e AND negativa.
Aplique o teorema de DeMorgan na expressão $\overline{ABC + DEF}$. Depois, assinale a alternativa que contém a expressão equivalente:
a) $\left(\overline{A+B+C}\right)\left(\overline{D+E+F}\right)$.
b) $\bar{A}\bar{B}\bar{C}\bar{D}\bar{E}\bar{F}$.
c) $\overline{ABC} + \overline{DEF}$.
d) $\left(\bar{A}+\bar{B}+\bar{C}\right)+\left(\bar{D}+\bar{E}+\bar{F}\right)$.
e) $(\bar{A}+\bar{B}+\bar{C})(\bar{D}+\bar{E}+\bar{F})$.

3. Uma porta XOR de duas entradas realiza uma operação lógica que resulta em 1 se, e somente se, uma das entradas for 1. A expressão booleana para uma porta XOR é $A\bar{B} + \bar{A}B$.
Assinale a alternativa que apresenta a expressão booleana para uma porta **XNOR**:
a) $\bar{A}\bar{B} + AB$.
b) $\bar{A} + \bar{B}$.
c) $\bar{A}B + A\bar{B}$.
d) $\bar{A}A + \bar{B}B$.
e) $\bar{A} + B$.

Seção 2.2

Análise booleana e síntese de circuitos lógicos

Diálogo aberto

Como vimos, podemos usar a álgebra Booleana para expressar a operação de uma porta lógica. O mesmo pode ser feito para um circuito lógico, que se constitui em uma combinação de portas lógicas, de forma que as saídas sejam determinadas como combinações dos valores de entrada. Para pôr em prática os conhecimentos desta seção, devemos relembrar o nosso contexto: você e seus amigos acabaram de se formar e fundaram uma empresa de segurança automotiva. Seus projetos envolvem desenvolver dispositivos eletrônicos que melhoram a segurança e o desempenho dos automóveis de alguma forma.

Em um dos seus novos projetos foi proposto um sistema para monitorar a tensão contínua da bateria de 12 V de um carro. O sistema é dividido em dois módulos: no primeiro módulo, um conversor analógico digital monitora a tensão da bateria e fornece na saída de 4 bits um número binário, que corresponde à tensão da bateria, em degraus de 1 V; o módulo seguinte é um circuito lógico que recebe como entrada as saídas do conversor e fornece uma saída igual a 1 para toda tensão menor ou igual a 6 V. Você é capaz de projetar esse circuito lógico?

Para lhe ajudar com esse projeto, nesta seção veremos algumas maneiras de analisar e sintetizar circuitos lógicos através da álgebra booleana.

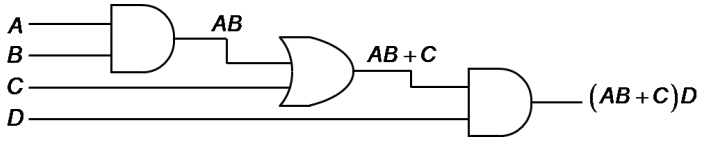
Espero que esteja animado para mais este conhecimento.

Bons estudos!

Não pode faltar

Como você já deve ter notado nesse momento, um circuito lógico pode ser descrito por uma equação booleana. Para circuitos simples, é relativamente fácil obter a expressão booleana de um certo circuito lógico. Devemos começar a escrever as expressões lógicas para cada porta a partir das entradas mais à esquerda até a saída final do circuito. Para entender melhor esse procedimento, considere o circuito lógico da Figura 2.7.

Figura 2.7 | Circuito lógico



Fonte: adaptada de Floyd (2007, p. 211).

Para a porta AND P_1 a expressão de saída é AB. Sua saída é uma das entradas da porta OR P_2, de modo que a expressão para P_2 é $AB + C$. Que por sua vez é entrada da porta AND P_3, de maneira que, a expressão para essa porta, $(AB + C)D$, é a expressão final para o circuito completo. Determinada a expressão booleana para o circuito lógico, é possível preencher uma tabela verdade para ele.

Exemplificando

Para o cálculo da expressão $(AB + C)D$, primeiro determine os valores das variáveis que levam a expressão para 1, usando as regras da álgebra booleana vistas na Seção 2.1. Essa expressão somente é igual a 1 quando $AB + C = 1$ e $D = 1$. Pois, assim,

$(AB + C)D = 1 \cdot 1 = 1$.

Agora, determine quando o termo $AB + C$ é igual a 1. Isso acontece se $AB = 1$ ou $C = 1$ ou se ambos forem iguais a 1, porque

$AB + C = 1 + C = 1$,

$AB + C = AB + 1 = 1$.

O termo *AB* é igual a 1 apenas se *A* e *B* forem iguais a 1.

Ou seja, a expressão $(AB + C)D = 1$ quando $C = 1$ e $D = 1$, independentemente dos valores de *A* e *B*, ou quando $A = 1$, $B = 1$ e $D = 1$, independentemente do valor de *B*. Para todas as outras combinações, $(AB + C)D = 0$.

O circuito da Figura 2.7 possui quatro entradas e, portanto, 16 ($2^4 = 16$) combinações de valores. O primeiro passo é escrever todas as combinações possíveis dos números 0 e 1 das variáveis de entrada em uma sequência binária, conforme pode ser visto na Tabela 2.3. Em seguida, coloque 1 na coluna da saída em cada linha em que a combinação das variáveis de entrada resulte em 1, de acordo com a avaliação prévia. Por fim, preencha as demais células com 0.

Tabela 2.3 | Tabela verdade do circuito lógico da Figura 2.7

A	*B*	*C*	*D*	*S*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Fonte: elaborada pelo autor.

Pesquise mais

Para aplicar a álgebra Booleana, em geral, é necessário simplificar a expressão. Esse procedimento depende do conhecimento completo da álgebra booleana e uma considerável prática na sua aplicação, além de habilidade e inteligência. A Seção 4.5 do livro *Sistemas digitais: fundamentos e aplicações* (FLOYD, 2007), disponível na nossa biblioteca virtual, traz diversos exemplos de simplificação de expressões booleanas. Leia e refaça os exemplos para aumentar sua habilidade. FLOYD, Thomas L. **Sistemas digitais**: fundamentos e aplicações. Tradução de José Lucimar do Nascimento. 9. ed. Porto Alegre : Bookman, 2007.

Qualquer expressão booleana pode ser convertida em qualquer uma das formas padrão, chamadas: **soma-de-produtos** e **produto-de-somas**. Essa padronização torna a análise e síntese de expressões booleanas mais sistemática e fácil.

O termo-produto, ou mintermo, consiste em um produto de literais, em que cada literal aparece apenas uma vez no termo. Quando dois ou mais termos-produto são somados, temos uma **soma-de-produtos**. Como os termos mostrados a seguir:

$$AB + \bar{A}BC$$

$$ABC + A\bar{B}C + \bar{A}B\bar{C}$$

Uma única variável também pode ser considerada como um termo-produto, portanto a expressão $A\bar{B}C + A\bar{B}\bar{C} + C$ também é classificada como uma soma-de-produtos. É importante salientar que em uma soma-de-produtos, o produto de duas variáveis não pode ser negado, mas mais de uma variável em um termo pode. Por exemplo, uma soma-de-produtos pode conter um termo $\bar{A}\bar{B}\bar{C}$, mas não um termo $\overline{ABC}$.

Assimile

O domínio de uma expressão booleana é definido como o conjunto de todas as variáveis contidas nessa expressão.

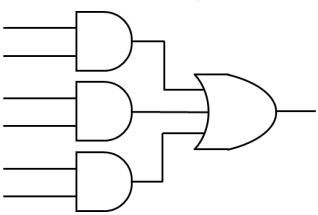
Por exemplo, a expressão $A\bar{B} + ABC$ tem como domínio o conjunto das variáveis *A*, *B* e *C*, já a expressão $AB + \bar{A}C\bar{D} + \bar{B}CE$ tem como domínio o conjunto das variáveis *A, B, C, D* e *E*.

Como foi dito, podemos transformar qualquer expressão lógica para o formato de soma-de-produtos. Por exemplo, a expressão $(AB + C)D$ pode ser convertida para uma soma-de-produto aplicando a lei distributiva:

$$(AB + C)D = ABD + CD.$$

Uma expressão de soma-de-produtos é implementada usando uma porta OR, cujas entradas são as saídas de duas ou mais portas AND. Esse tipo de projeto é chamado de lógica AND/OR e uma implementação genérica é vista na Figura 2.8.

Figura 2.8 | Implementação de uma soma-de-produtos



Fonte: elaborada pelo autor.

Reflita

Uma expressão de soma-de-produtos pode ser implementada usando apenas portas NAND, como isso é possível?

Uma expressão em que todas as variáveis do seu domínio aparecem em todos os seus termos-produto é classificada como uma **soma-de-produtos padrão**, como em $A\bar{B}C\bar{D} + AB\bar{C}\bar{D} + \bar{A}BCD$, para um domínio de quatro variáveis. Esse tipo de expressão, útil na construção de tabelas verdade, é muito importante para o método de simplificação utilizando mapa de Karnaugh, que será tema da próxima seção.

É possível transformar qualquer expressão de soma-de-produtos em uma forma padrão. Os termos-produto, em uma expressão, que não contém todas as variáveis do domínio, podem ser expandidos para a forma padrão, isso pode ser feito em dois passos.

1. Uma vez $A + \bar{A} = 1$, podemos multiplicar cada um dos termos-produto, que não estão na forma padrão, pela soma de uma variável, que não aparece no termo, com o seu complemento, resultando em dois termos-produto sem que altere o seu valor, uma vez que estamos apenas multiplicando o termo-produto por 1.
2. O passo 1 deve ser repetido até que todos os termos produtos resultantes contenham todas as variáveis do domínio na forma complementada ou não complementada.

Exemplificando

Vamos converter a expressão $ABD + CD$ para a sua forma padrão de soma-de-produtos.

O seu domínio é A, B, C e D. Para facilitar, vamos trabalhar com um termo por vez.

$$ABD\left(C + \bar{C}\right) = ABCD + AB\bar{C}D \qquad (2.12)$$

A conversão do termo ABD resultou em dois termos-produto padrão.

No segundo termo não aparecem as variáveis A e B. Mas precisamos fazer essa conversão por partes. Primeiro, multiplicamos o termo CD por $(A+\bar{A})$:

$$CD(A+\bar{A}) = ACD + \bar{A}CD.$$

Nos dois termos resultantes não aparece a variável B. Portanto, devemos multiplicar os dois termos por $(B+\bar{B})$.

$$ACD(B+\bar{B}) = ABCD + A\bar{B}CD \quad (2.13)$$

$$\bar{A}CD(B+\bar{B}) = \bar{A}BCD + \bar{A}\bar{B}CD \quad (2.14)$$

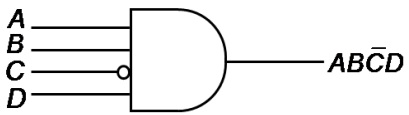
Note que o primeiro termo de 2.12 e o primeiro termo de 2.13 são idênticos. Portanto, não precisam aparecer duas vezes na expressão. Assim,

$$(AB+C)D = ABD + CD = ABCD + AB\bar{C}D + A\bar{B}CD + \bar{A}BCD + \bar{A}\bar{B}CD \quad (2.15)$$

Um termo-produto padrão somente será igual a 1 para uma certa combinação de valores das variáveis. Por exemplo, o termo $AB\bar{C}D$ é igual a 1 quando $A=1$, $B=1$, $C=0$ e Uma vez que $AB\bar{C}D = 1\cdot 1\cdot \bar{0}\cdot 1 = 1$, é dito que esse termo-produto tem um valor binário 1101 (decimal treze). Um termo-produto é implementado utilizando uma combinação entre uma porta AND e inversores, usados quando é necessário o complemento de uma variável.

A implementação do termo $AB\bar{C}D$ pode ser vista na Figura 2.9.

Figura 2.9 | Implementação do termo $AB\bar{C}D$



Fonte: elaborada pelo autor.

O termo-soma, ou maxtermo, consiste em uma soma de literais, em que cada literal aparece apenas uma vez no termo. Quando dois ou mais termos-soma são multiplicados, temos um **produto-de-somas**. Como os mostrados a seguir:

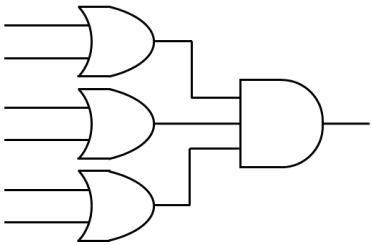
$$(A+\bar{B})(A+\bar{B}+C),$$

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

Em um termo-soma, cada variável só pode ser complementada individualmente. Por exemplo, $\bar{A}+\bar{B}+\bar{C}$ é um termo-soma, mas $\overline{A+B+C}$ não. Uma expressão de produto-de-somas é implementada usando uma porta AND, cujas entradas são as saídas de duas ou mais

portas OR. Assim como na implementação da soma-de-produtos, esse tipo de projeto também usa a lógica AND/OR e uma implementação genérica é vista na Figura 2.10.

Figura 2.10 | Implementação de um produto de somas



Fonte: elaborada pelo autor.

Reflita

Uma expressão de produto-de-somas pode ser implementada usando apenas portas NOR, como isso é possível?

O produto de somas também possui uma forma padrão. Por exemplo, na expressão

$$(A+\bar{B}+C)(A+B+C+\bar{D})(A+\bar{B}+\bar{C}+D)$$

as variáveis *A, B, C* e *D* constituem o domínio da expressão. Note que a variável *D* não aparece no primeiro termo-soma da expressão. Em uma expressão de **produto-de-somas padrão**, cada termo-soma da expressão deve conter todas as variáveis do domínio, como em:

$$(A+\bar{B}+C+D)(A+\bar{B}+C+\bar{D})(A+B+C+\bar{D})(A+\bar{B}+\bar{C}+D),$$

que é uma expressão de produto de somas padrão equivalente à expressão anterior. É possível converter qualquer expressão de produto-de-somas para a forma padrão usando a álgebra booleana, isso pode ser feito em três passos:

1. Uma vez que $A \cdot \bar{A} = 0$, podemos somar a cada um dos termo-soma, que não estão na forma padrão, um termo formado pelo produto da variável que não aparece no termo com o seu complemento, resultando em dois termos-soma, sem alterar o valor da expressão, uma vez que podemos somar 0 com qualquer coisa sem alterar o seu valor.

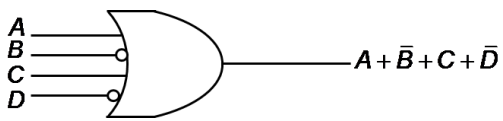
2. Em seguida, podemos aplicar a regra $A+BC=(A+B)(A+C)$.
3. Agora, basta repetir os passos anteriores até que todos os termos somas resultantes contenham todas as variáveis do domínio na forma complementada ou não complementada.

Faça você mesmo

Converta a expressão $(A+C)(A+D)D$ para a forma de produto-de-somas padrão.

Um termo-soma padrão é igual a 0 apenas para uma certa combinação de valores das suas variáveis. Por exemplo, o termo-soma $A+\bar{B}+C+\bar{D}$ é 0 quando $A=0$, $B=1$, $C=0$ e $D=1$, uma vez que, $A+\bar{B}+C+\bar{D}=0+\bar{1}+0+\bar{1}=0$. Nesse caso, o termo-soma tem valor binário 0101 (decimal 5). Um termo-soma pode ser obtido com uma porta OR. Por exemplo, o termo $A+\bar{B}+C+\bar{D}$ é implementado como visto na Figura 2.11.

Figura 2.11 | Implementação do termo $A+\bar{B}+C+\bar{D}$



Fonte: elaborada pelo autor.

É muito simples preencher uma tabela verdade para uma expressão booleana padrão, ou determinar uma expressão booleana padrão a partir de uma tabela verdade.

A expressão de soma-de-produtos deve ser convertida para a forma padrão, caso já não esteja nesse formato. E em seguida, colocamos 1 na coluna da saída em cada linha em que o valor binário leva um termo-produto da soma para 1.

Vamos utilizar como exemplo a Expressão 2.15, os valores binários dos termos-produto iguais a 1 são $\bar{A}\bar{B}CD:0011$, $\bar{A}BCD:0111$, $A\bar{B}CD:1011$, $AB\bar{C}D:1101$, $ABCD:1111$. Na linha referente a cada um desses termos, coloque 1 na coluna da saída e coloque 0 nas demais linhas. De fato, essa tabela já foi construída no início da seção, portanto, compare seu resultado com o da Tabela 2.3.

Para determinar a expressão de soma-de-produtos padrão representada por uma tabela verdade, basta fazer o caminho inverso. Considere todas as linhas cujo valor de saída seja 1 e

converta o valor binário de cada linha para o termo-produto correspondente. Por exemplo, o valor binário 1010 é convertido para o termo-produto $A\bar{B}C\bar{D}$.

Também é possível determinar uma expressão de poduto-de-somas a partir de uma tabela verdade. Para isso, deve-se considerar as linhas cujo valor de saída seja 0 e converter o seu valor binário para o termo-soma correspondente. Agora, cada 1 no valor binário corresponde ao complemento da variável, por exemplo, o valor binário 1010 é convertido para o termo-soma $\bar{A}+B+\bar{C}+D$.

Faça você mesmo

A partir da tabela verdade na Tabela 2.4, determine a expressão de soma-de-produtos padrão e a expressão de produto de somas padrão equivalente.

Tabela 2.4 | Tabela verdade

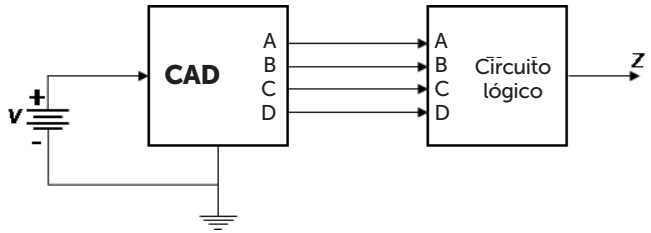
A	*B*	*C*	*S*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Fonte: elaborada pelo autor.

Sem medo de errar

Você está animado como nunca com os novos projetos da sua empresa. Você precisa projetar um circuito lógico que gere uma saída em nível ALTO quando a tensão da bateria for menor ou igual a 6 V. As entradas desse circuito lógico são fornecidas por um conversor analógico-digital de quatro bits, que fornece como saída um número binário de quatro bits que corresponde à tensão da bateria em degraus de 1 V, como pode ser visto no esquema da Figura 2.12.

Figura 2.12 | Esquema de funcionamento do circuito



Fonte: elaborada pelo autor.

Seu primeiro passo para projetar o circuito lógico foi escrever a tabela verdade, vista na Tabela 2.5, em que o X foi usado para quando a saída não importa, pois, nesse caso, não se espera que o circuito receba essas combinações de bits na entrada.

Tabela 2.5 | Tabela verdade do circuito lógico

	A	*B*	*C*	*D*	*Z*
(0)	0	0	0	0	1
(1)	0	0	0	1	1
(2)	0	0	1	0	1
(3)	0	0	1	1	1
(4)	0	1	0	0	1
(5)	0	1	0	1	1
(6)	0	1	1	0	1
(7)	0	1	1	1	0
(8)	1	0	0	0	0
(9)	1	0	0	1	0
(10)	1	0	1	0	0
(11)	1	0	1	1	0
(12)	1	1	0	0	0
(13)	1	1	0	1	X
(14)	1	1	1	0	X
(15)	1	1	1	1	X

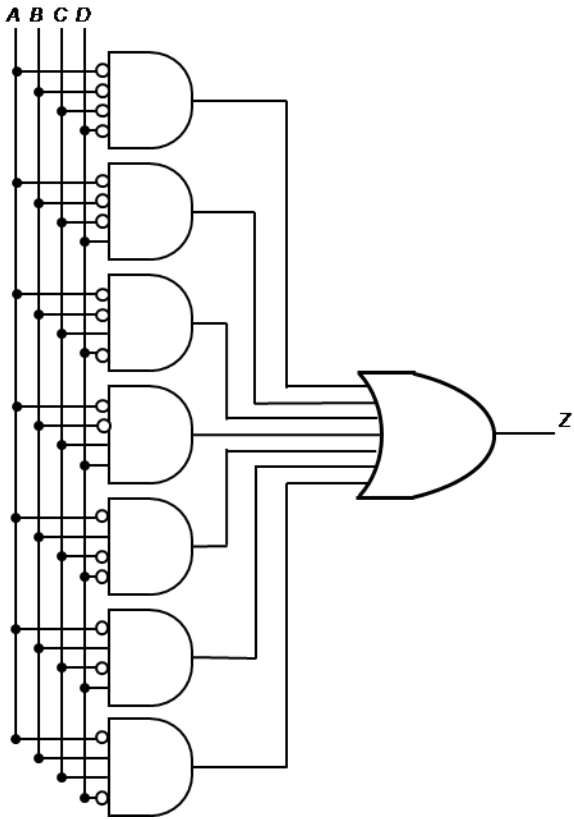
Fonte: elaborada pelo autor.

Uma maneira de implementar esse circuito é com uma soma-de-produtos. Para isso, você seleciona todos os termos produtos em que a saída seja 1. Assim, a expressão booleana para o circuito lógico do seu projeto é:

$$Z = \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + \bar{A}B\bar{C}D + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D + A\bar{B}CD.$$

Ela está sintetizada no circuito lógico da Figura 2.13.

Figura 2.13 | Circuito lógico



Fonte: elaborada pelo autor.

Pronto, a partir do circuito lógico mostrado na Figura 2.13, você pode implementar um dispositivo de monitoramento da tensão em corrente contínua de uma bateria. Excelente! Sua empresa está indo muito bem!

Faça valer a pena

1. A tabela verdade para uma expressão booleana padrão pode ser facilmente obtida usando valores binários para cada termo na expressão. Desenvolva uma tabela verdade para a expressão $\bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$ e assinale a alternativa correta:

a)

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

b)

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

c)

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

d)

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

e)

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

2. Qualquer expressão lógica pode ser convertida para o formato de soma-de-produtos usando a álgebra booleana. Por exemplo, a expressão $(AB+C)D$ é convertida para o formato de soma-de-produto aplicando a lei distributiva: $(AB+C)D = ABD+CD$.

Considerando esse contexto, avalie as seguintes asserções e a relação proposta entre elas:

I. A expressão $ABD + CD$ é uma soma-de-produtos padrão
PORQUE
II. Todas as variáveis do domínio aparecem em cada um dos termos-produto na expressão.
A respeito dessas asserções, assinale a alternativa correta:
a) As asserções I e II são proposições verdadeiras, e a II é uma justificativa da I.
b) As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa da I.
c) A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.
d) A asserção I e uma proposição falsa, e a II é uma proposição verdadeira.
e) As asserções I e II são proposições falsas.

3. O termo-produto é um termo que consiste em um produto de literais. Quando dois ou mais termos-produtos são somados por uma adição booleana, temos uma **soma-de-produtos**.
Nesse contexto, avalie as afirmações a seguir:
I. A expressão $\overline{AB} + AB$ é um exemplo de soma-de-produtos.
II. Uma expressão na forma de soma-de-produtos pode conter um termo de uma única variável, como em $A\bar{B}C + A\bar{B}\bar{C} + C$.
III. Uma expressão de **soma-de-produtos padrão** é uma expressão na qual todas as variáveis do domínio aparecem em cada um dos termos produtos na expressão, como em $A\bar{B}C\bar{D} + AB\bar{C}\bar{D} + \bar{A}BCD$.
É correto o que se afirma em:
a) I, apenas.
b) III, apenas.
c) I e III, apenas.
d) II e III, apenas.
e) I, II e III.

Seção 2.3

Simplificação de circuitos lógicos

Diálogo aberto

Como pudemos notar na Seção 2.2, expressões booleanas diferentes podem produzir o mesmo resultado. A consequência disso é que circuitos lógicos diferentes acabam sendo equivalentes. Em geral, devemos ter em mente que ao implementar uma função lógica é desejável que tanto o número de portas lógicas quanto o número de literais sejam mínimos. As expressões booleanas com menor número de operadores lógicos requerem um menor número de portas e um menor número de entradas por porta. Isso nos permite, portanto, projetar circuitos lógicos com menor custo.

Retomando o nosso contexto, você e seus amigos acabaram de se formar e fundaram uma empresa de segurança automotiva. Seus projetos envolvem desenvolver dispositivos eletrônicos que melhorem a segurança e o desempenho dos automóveis de alguma forma. No seu último projeto, você desenvolveu um circuito lógico que recebe como entrada um código binário de quatro bits de um conversor analógico-digital e fornece como saída um sinal em nível ALTO sempre que o valor binário for menor ou igual a 6 (0110), ou seja, quando a tensão da bateria for menor ou igual a 6 V. Um dos seus sócios questiona se a solução apresentada é a mais eficiente. Você pode garantir isso?

Para lhe ajudar a responder essa pergunta, nesta seção apresentaremos um método mais fácil e rápido de minimização. O método requer a representação gráfica dos termos-produtos da função em um mapa bidimensional.

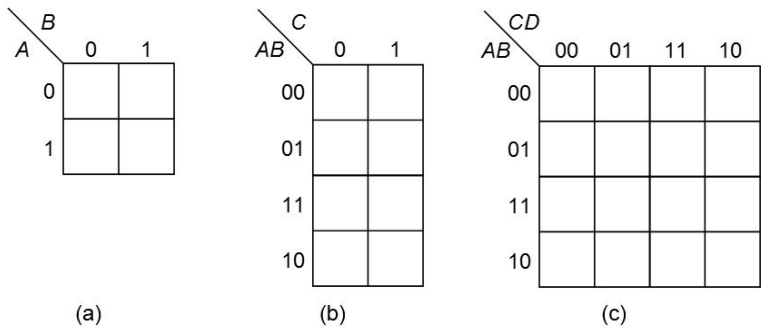
Aproveite este momento para adquirir mais conhecimento.

Não pode faltar

O mapa de Karnaugh é uma representação gráfica bidimensional de uma tabela verdade, e pode ser encarado como um método de simplificação de expressões booleanas, ou circuitos lógicos, mais simples que a simplificação pela álgebra booleana. É possível garantir, ainda, que as expressões obtidas com esse método sejam as mais simples possíveis, por isso chamadas de expressões mínimas.

Enquanto a tabela verdade é organizada em linhas e colunas, o mapa de Karnaugh é um arranjo de células no qual cada célula representa um valor binário das variáveis de entrada. Por exemplo, considere a célula mais à esquerda e acima de cada mapa na Figura 2.14: ela representa $AB = 00$ no mapa para expressões com duas variáveis em (a), $ABC = 000$ no mapa para expressões com três variáveis em (b) e $ABCD = 0000$ no mapa para expressões com quatro variáveis em (d). O mapa de Karnaugh possui 2^n células, em que *n* é número de variáveis de entrada. Esse número é igual ao número de linhas de uma tabela verdade.

Figura 2.14 | Mapa de Karnaugh (a) duas variáveis; (b) três variáveis; (c) quatro variáveis



Fonte: elaborada pelo autor.

Assimile

A quantidade de células em um mapa de Karnaugh é dada por 2^n, em que *n* é a quantidade de variáveis.

Por exemplo, em um mapa de três variáveis a quantidade de células é $2^3 = 8$. Para o mapa de quatro variáveis, o número de células é $2^4 = 16$; e para o mapa de cinco variáveis, o número de células é $2^5 = 32$.

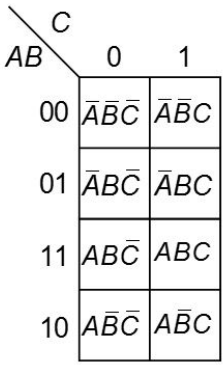
Cada linha da tabela verdade possui sua região própria no mapa, como podemos ver na comparação entre a Tabela 2.6 e a Figura 2.15, para o caso de três variáveis.

Tabela 2.6 | Tabela verdade para três variáveis

A	*B*	*C*	*Termo*
0	0	0	$\bar{A}\bar{B}\bar{C}$
0	0	1	$\bar{A}\bar{B}C$
0	1	0	$\bar{A}B\bar{C}$
0	1	1	$\bar{A}BC$
1	0	0	$A\bar{B}\bar{C}$
1	0	1	$A\bar{B}C$
1	1	0	$AB\bar{C}$
1	1	1	ABC

Fonte: elaborada pelo autor.

Figura 2.15 | Mapa de Karnaugh de três variáveis



Fonte: elaborada pelo autor.

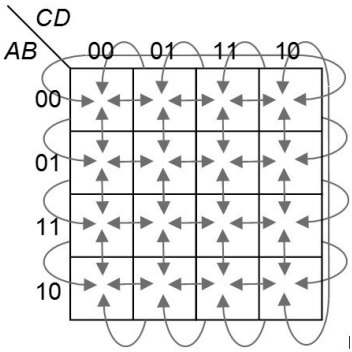
Os valores binários de *A* e *B* possíveis são distribuídos pelo lado esquerdo do mapa enquanto que os valores de *C* são distribuídos na parte superior. Observe a sequência utilizada, ela usa o código Gray, de modo que apenas 1 bit varie entre células adjacentes.

Assimile

Duas células são **adjacentes** quando entre elas há mudança de bit em apenas uma das suas variáveis.

Na prática, cada célula é adjacente às suas células vizinhas pelos quatro lados. E nesse momento é importante salientar que as células representadas nos limites do mapa são adjacentes às células no outro extremo, por exemplo, uma célula na coluna mais à esquerda é adjacente à célula mais à direita e uma célula na linha inferior é adjacente à célula correspondente na linha superior. Mas uma célula **não** é adjacente às células que a tocam diagonalmente em qualquer um dos vértices (FLOYD, 2007). Observe o exemplo da Figura 2.16, que ilustra a adjacência de células em um mapa de quatro variáveis.

Figura 2.16 | Células adjacentes em um mapa de Karnaugh de quatro variáveis



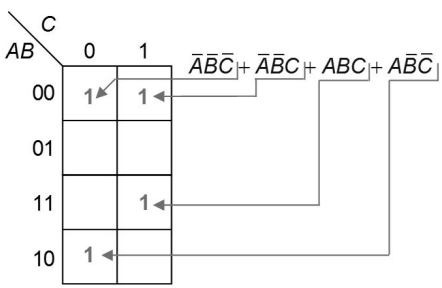
Fonte: elaborada pelo autor.

As mesmas regras se aplicam a qualquer mapa de Karnough, independentemente do número de variáveis. O preenchimento de um mapa de Karnough se dá colocando um bit 1 em cada célula correspondente ao valor de termo-produto na expressão.

Exemplificando

Quando uma expressão de soma-de-produtos é completamente inserida no mapa, existirá uma quantidade de números 1 no mapa igual ao número de termos produtos na expressão da soma-de-produtos padrão. Por exemplo, para a expressão $\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + AB\bar{C} + A\bar{B}\bar{C}$, temos o mapa da Karnaugh da Figura 2.17.

Figura 2.17 | Inserção de uma expressão de soma-de-produtos



Fonte: elaborada pelo autor.

As demais células são os número 0 mas, geralmente, é interessante suprimir sua representação para não poluir o mapa visualmente.

Agora é a sua vez, faça um mapa de Karnaugh de quatro variáveis para a expressão $\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} + A\bar{B}C\bar{D}$.

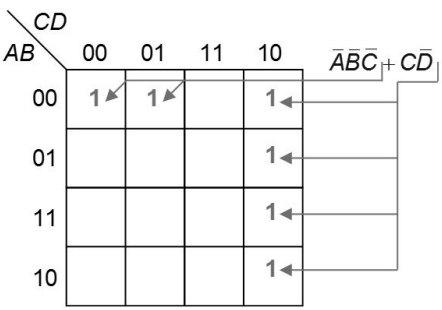
Como você já deve ter notado, é importante que a expressão booleana esteja na forma padrão para que seja possível montar o mapa de Karnaugh. Se expressão não estiver na forma padrão, então, ela deve ser convertida usando o procedimento indicado na Seção 2.2, ou através de uma **expansão numérica**, que pode ser mais eficiente, uma vez que é preciso avaliar a expressão antes de colocá-la no mapa.

Lembre-se de que em um termo-produto não padrão falta uma ou mais variáveis do domínio. Por exemplo, seja o termo $\bar{A}\bar{B}C$ em uma expressão de soma-de-produtos de quatro variáveis, podemos expandir esse termo numericamente para a forma padrão escrevendo o valor binário das variáveis do termo: $\bar{A}\bar{B}C : 000$, e, em seguida, escrevendo todas as combinações possíveis para as variáveis que não aparecem. Nesse caso, são apenas duas combinações, $\bar{A}\bar{B}C\bar{D} : 0000$ e $\bar{A}\bar{B}C\bar{D} : 0001$.

Exemplificando

Para que o procedimento de inserir uma expressão de soma-de-produtos não padrão em um mapa de Karnaugh fique mais claro, vamos montar o mapa para a expressão $\bar{A}\bar{B}C + C\bar{D}$ na Figura 2.18. Como vimos, para o termo $\bar{A}\bar{B}C$, temos duas combinações possíveis: $\bar{A}\bar{B}C\bar{D} : 0000$ e $\bar{A}\bar{B}C\bar{D} : 0001$. Já para o termo $C\bar{D}$, são quatro combinações possíveis: $\bar{A}\bar{B}C\bar{D} : 0000$, $\bar{A}BC\bar{D} : 0100$, $ABC\bar{D} : 1100$ e $A\bar{B}C\bar{D} : 1000$.

Figura 2.18 | Inserção de uma expressão de soma-de-produtos não padrão



Fonte: elaborada pelo autor.

Teste suas habilidades e faça um mapa da Karnaugh para a seguinte expressão de soma-de-produtos: $A + \bar{A}B + \bar{A}BC$. Considere um domínio de três variáveis para essa expressão.

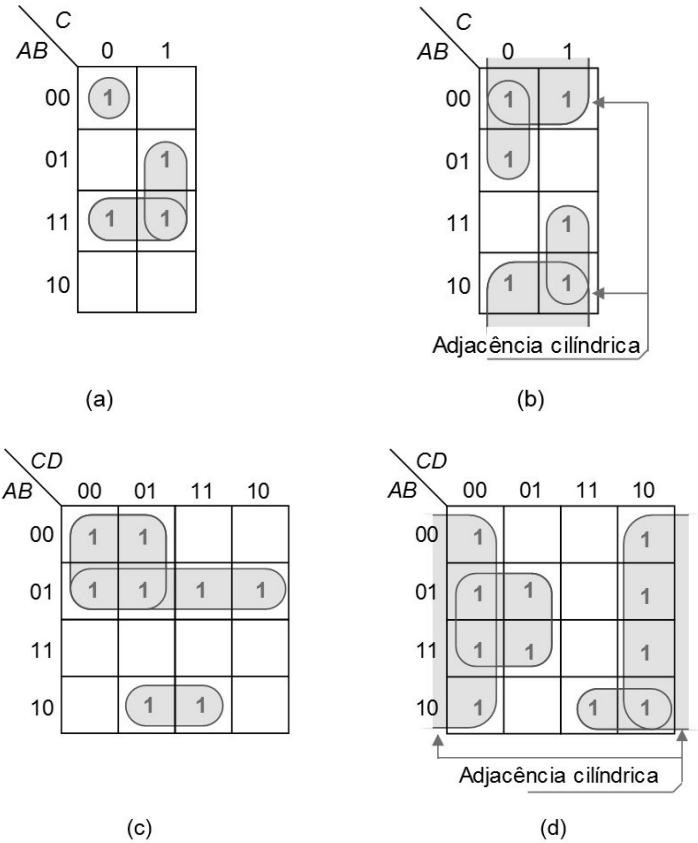
Agora que já sabemos como preencher um mapa de Karnaugh adequadamente, veremos como **simplificar uma expressão booleana utilizando o mapa de Karnaugh**. Segundo Floyd (2007), esse processo resulta em uma expressão que contém o menor número de termos possível com o menor número de variáveis possíveis e é chamado, por razões óbvias, de **minimização**. Após a expressão de soma-de-produtos ser inserida no mapa, uma expressão de soma-de-produtos mínima é obtida agrupando-se, de acordo com certas regras, os números 1 e determinando a expressão de soma-de-produtos mínima a partir do mapa.

O objetivo é maximizar o tamanho dos grupos e minimizar a quantidade de grupos. As regras de agrupamento de 1s em um mapa são as seguintes:

1. Um grupo só pode conter uma quantidade de células que seja uma potência inteira de 2, por exemplo: 1, 2, 4 ou 16 células. No caso de um mapa de três variáveis, $2^3 = 8$ é o tamanho do maior grupo possível.
2. Cada célula em um grupo deve ser adjacente a uma ou mais células do mesmo grupo.
3. Sempre inclua o maior número de algarismos 1 em um grupo, respeitando a regra 1.
4. Cada número 1 no mapa tem que ser incluído em pelo menos um grupo. Os números 1 que já fazem parte de um grupo podem ser incluídos num outro grupo, enquanto os grupos sobrepostos incluem números 1 não comuns.

Na Figura 2.19 vemos alguns exemplos de agrupamentos de células em mapas de três e quatro variáveis. Um destaque importante para as Figuras 2.19(b) e (d): ocorrem agrupamentos entre células das fronteiras, devido à adjacência cilíndrica.

Figura 2.19 | Exemplos de agrupamentos no mapa de Karnaugh (a) uma e duas células em um mapa de três variáveis; (b) duas e quatro células em um mapa de três variáveis; (c) duas e quatro células em um mapa de quatro variáveis; (d) duas, quatro e oito células em um mapa de quatro variáveis



Fonte: adaptada de Floyd (2007, p. 232).

Reflita

Você consegue notar alguma relação entre os conceitos **expansão numérica** de um termo-produto e o **agrupamento de células** descritos?

Uma expressão de soma-de-produtos minimizada contém a menor quantidade possível de termos com a menor quantidade possível de variáveis por termos, e, geralmente, uma **soma-de-produtos mínima** pode ser implementada com menos portas lógicas que uma expressão padrão (FLOYD, 2007).

Após preencher adequadamente o mapa de Karnaugh e criar o menor número possível de grupos e que eles englobem o maior número de células possível, podemos determinar a expressão de soma-de-produtos mínima equivalente conforme as seguintes regras:

1. Cada grupo é equivalente a um termo-produto composto por todas as variáveis que o valor não varia dentro do grupo.

2. Determine o termo produto mínimo para cada grupo:

a. Para um mapa de três variáveis:

- Um grupo de uma célula resulta em um termo produto de três variáveis.
- Um grupo de duas células resulta em um termo produto de duas variáveis.
- Um grupo de quatro células resulta em um termo de uma variável.
- Um grupo de oito células resulta no valor 1 para a expressão.

b. Para um mapa de quatro variáveis:

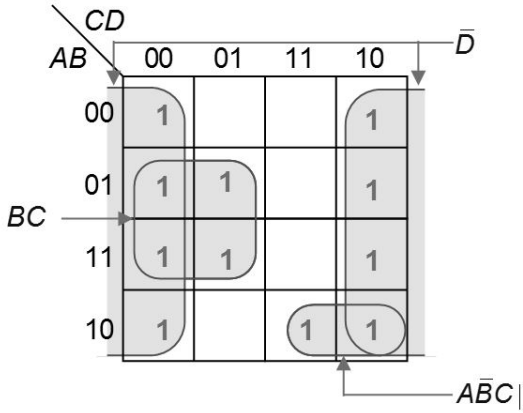
- Um grupo de uma célula resulta em um termo produto de quatro variáveis.
- Um grupo de duas células resulta em um termo produto de três variáveis.
- Um grupo de quatro células resulta em um termo produto de duas variáveis.
- Um grupo de oito células resulta em um termo de uma variável.
- Um grupo de 16 células resulta no valor 1 para a expressão.

3. Quando se obtém todos os termos-produtos mínimos com o mapa de Karnaugh, eles são somados formando uma expressão de soma-de-produtos mínima.

Exemplificando

Vamos considerar o mapa da Figura 2.19(d) e determinar a expressão de soma-de-produtos mínima para ele.

Figura 2.20 | Inserção de uma expressão de soma-de-produtos não padrão



Fonte: elaborada pelo autor.

Como foi possível criar três grupos, teremos uma expressão com três termos-produto, sendo um termo com três variáveis (devido ao grupo com duas células), um termo de duas variáveis (devido ao grupo com quatro células) e um termo com uma variável (devido ao grupo com oito células). Assim, a expressão em soma-de-produtos mínima resultante é dada por:

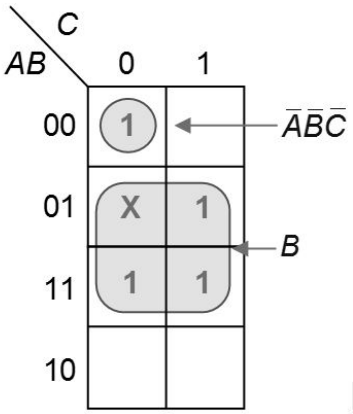
$$A\bar{B}C + BC + \bar{D}$$

Pratique também, encontre a expressão em soma-de-produtos mínima para os demais mapas da Figura 2.19.

Existem alguns casos em que uma determinada combinação de entrada não é permitida. Esses estados podem ser tratados como bits "não importa" (é comum também usar a expressão em inglês, *don't care*) em relação aos seus efeitos na saída. Na prática, a existência desses termos permite que os usemos como 0 ou 1, conforme seja conveniente em uma minimização.

Por exemplo, na Figura 2.21 existe uma saída "não importa" na posição 010. Se não houvesse a saída "não importa" nessa célula, a expressão mínima resultante desse mapa seria $\bar{A}\bar{B}\bar{C} + AB + BC$. No entanto, como podemos considerá-lo como 1 para formar um agrupamento de quatro células, a expressão mínima resultante é $\bar{A}\bar{B}\bar{C} + B$.

Figura 2.21 | Exemplo do uso do bit "não importa"



Fonte: elaborada pelo autor.

Pesquise mais

Embora seja muito mais comum lidarmos com termos-produto e expressões em soma-de-produtos, todos os resultados e análises feitos aqui podem ser estendidos de maneira análoga aos termos-produto e expressões em produto-de-somas. Para saber mais sobre a técnica de minimização de produto-de-somas usando o mapa de Karnaugh, leia a seção 4.10 do do livro *Sistemas digitais: fundamentos e aplicações* (FLOYD, 2007), disponível na nossa biblioteca virtual.

FLOYD, Thomas L. **Sistemas digitais**: fundamentos e aplicações. Tradução de José Lucimar do Nascimento. 9. ed. Porto Alegre : Bookman, 2007.

Sem medo de errar

No seu último projeto, você desenvolveu um circuito lógico que recebe como entrada um código binário de quatro bits de um conversor analógico-digital e fornece como saída um sinal em nível ALTO sempre que o valor binário for menor ou igual a 6 (0110), ou seja, quando a tensão da bateria for menor ou igual a 6 V. O circuito lógico desenvolvido possui a seguinte expressão em soma-de-produtos padrão:

$$Z = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \bar{A}BCD.$$

A tabela verdade utilizada por você naquele momento pode ser vista na Tabela 2.7.

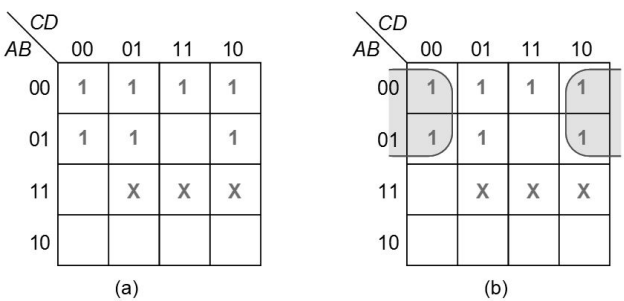
Tabela 2.7 | Tabela verdade do circuito lógico

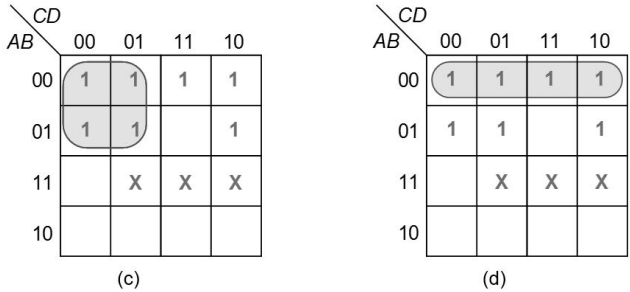
	A	*B*	*C*	*D*	*Z*
(0)	0	0	0	0	1
(1)	0	0	0	1	1
(2)	0	0	1	0	1
(3)	0	0	1	1	1
(4)	0	1	0	0	1
(5)	0	1	0	1	1
(6)	0	1	1	0	1
(7)	0	1	1	1	0
(8)	1	0	0	0	0
(9)	1	0	0	1	0
(10)	1	0	1	0	0
(11)	1	0	1	1	0
(12)	1	1	0	0	0
(13)	1	1	0	1	X
(14)	1	1	1	0	X
(15)	1	1	1	1	X

Fonte: elaborada pelo autor.

Agora, você já sabe que é possível minimizar esse circuito de modo a diminuir o custo do projeto, utilizando o mapa de Karnaugh. Seu primeiro passo é transpor as informações da tabela para o mapa, como pode ser visto na Figura 2.22(a). Depois, você deve agrupar os números 1 convenientemente. Você decidiu fazer três agrupamentos de quatro células, como pode ser visto na Figura 2.22(b), (c) e (d).

Figura 2.22 | (a) Mapa de Karnaugh; (b) agrupamento $\bar{A}\bar{D}$; (c) agrupamento $\bar{A}\bar{C}$; (d) agrupamento $\bar{A}\bar{B}$





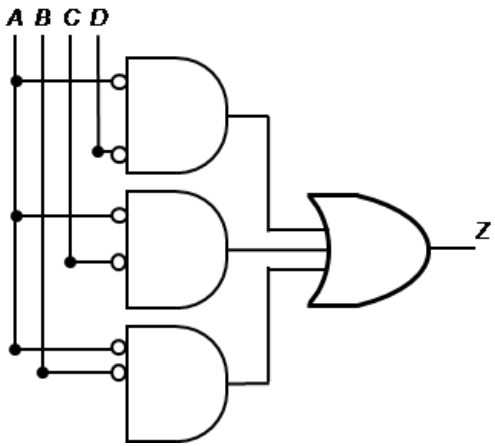
Fonte: elaborada pelo autor.

Com isso, é possível reduzir o circuito para a seguinte expressão de soma-de-produtos:

$$Z = \overline{A}\overline{D} + \overline{A}\overline{C} + \overline{A}\overline{B}\,.$$

Assim, compare o circuito minimizado, visto na Figura 2.23, com o circuito anterior, produzido na Figura 2.13, na seção anterior.

Figura 2.23 | Circuito lógico minimizado



Fonte: elaborada pelo autor.

Note que o uso das saídas "não importa" não foi feito nesta minimização.

Faça valer a pena

1. O mapa de Karnough pode ser usado para simplificar expressões booleanas, obtendo sua forma mínima.

Uma etapa importante para o processo de minimização através do mapa de Karnaugh é o seu preenchimento correto. Assinale a alternativa que contém o mapa equivalente à expressão $\bar{A}\bar{B}\bar{C} + AB\bar{C} + \bar{A}BC + ABC + A\bar{B}C$:

a)

AB \ C	0	1
00		1
01	1	
11	1	1
10	1	

b)

AB \ C	0	1
00	1	
01		1
11	1	1
10		1

c)

AB \ C	0	1
00	1	
01	1	1
11	1	
10		1

d)

AB \ C	0	1
00	1	
01	1	1
11	1	
10		1

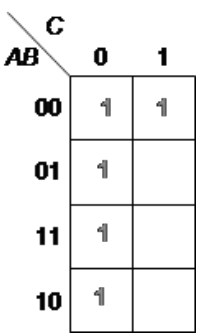
e)

AB \ C	0	1
00		1
01	1	1
11	1	
10		1

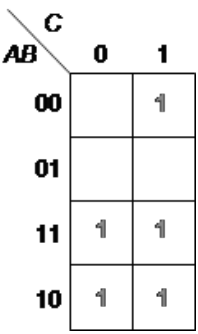
2. Ao preencher um mapa de Karnaugh, é importante que a expressão booleana esteja na forma padrão. Caso isso não ocorra, ela deve ser convertida para a forma padrão ou o mapa deve ser preenchido através de uma **expansão numérica**.

Marque a alternativa que apresenta o mapa de Karnaugh para a expressão $AB + C$ em um domínio de três variáveis:

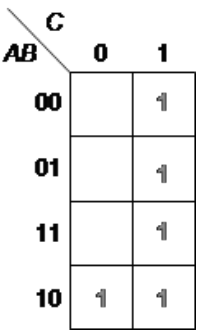
a)



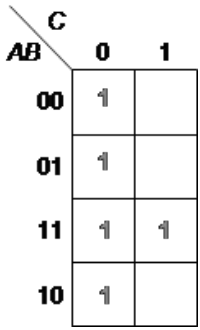
b)



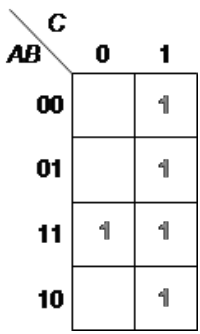
c)



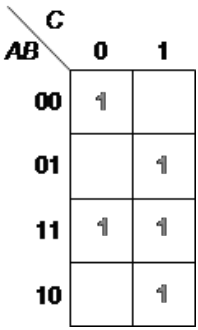
d)



e)



3. Uma expressão de soma-de-produtos minimizada contém a menor quantidade possível de termos com a menor quantidade possível de variáveis por termos e, geralmente, uma **soma-de-produtos mínima** pode ser implementada com menos portas lógicas que uma expressão padrão. Assinale a alternativa que contém a expressão em **soma-de-produtos mínima** para o seguinte mapa de Karnaugh:



a) $\bar{A}\bar{B}\bar{C} + AB\bar{C} + \bar{A}BC + ABC + A\bar{B}C$.
b) $\bar{A}\bar{B}\bar{C} + AB + BC + AC$.
c) $ABC + \bar{A}\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{C}$.
d) $ABC + \bar{A}\bar{B}C + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C}$.
e) $\bar{A}\bar{B}\bar{C} + AB + \bar{A}BC + A\bar{B}C$.

Referências

BOOLE, George. **An investigation of the laws of thought on which are founded the mathematical theories of logic and probabilities**. Londres: Macmillan, 1954.

FLOYD, Thomas. **Sistemas digitais**: fundamentos e aplicações. 9. ed. São Paulo: Bookman, 2007. 888 p.

TEXAS Instruments. **SNx4HC00 Quadruple 2-Input Positive-NAND Gates datasheet**. Texas Instruments. 2016. Disponível em: <http://www.ti.com/lit/ds/symlink/sn74hc00.pdf>. Acesso em: 8 out. 2017.

Lógica combinacional

Convite ao estudo

Nas unidades anteriores, estudamos o funcionamento das principais portas lógicas básicas e usamos a álgebra booleana para descrever e analisar os circuitos lógicos. Além disso, vimos também o mapa de Karnaugh, que nos permitiu aprender como simplificar os circuitos lógicos que compõem os sistemas digitais.

Nesta unidade, continuaremos nosso estudo sobre os circuitos combinacionais. Veremos que as portas lógicas básicas podem ser combinadas para realizar atividades que são importantes para o funcionamento dos computadores que tendem a facilitar nossas atividades. Sendo assim, na Seção 3.1, conheceremos os circuitos somadores: meio somador e somador completo, além dos circuitos subtratores. Na Seção 3.2, aprenderemos sobre os circuitos comparadores, codificadores e decodificadores. Por fim, na Seção 3.3, descobriremos os multiplexadores (MUX) e demultiplexadores (DEMUX).

Para pôr todo esse conhecimento em prática, considere que você é o responsável técnico de uma empresa de sistemas embarcados e terá que desenvolver alguns projetos simples e eficiente solicitados pelo seu chefe imediato, que está avaliando suas competências técnicas. E então, pronto para mais um desafio?

Bem-vindo a mais uma etapa de conhecimentos dos sistemas digitais.

Bons estudos e ótimo trabalho!

Seção 3.1

Circuito somador

Diálogo aberto

Frequentemente, em nossas tarefas do cotidiano, nos cálculos científicos ou nos negócios, é fundamental realizarmos operações aritméticas como soma e subtração. Muitas vezes, utilizamos as calculadoras digitais ou computadores para realizar essa atividade. Ao fazermos isso, estamos utilizando os circuitos aritméticos com lógica combinacional.

Esses são tipos de circuitos dedicados que realizam operações de adição e subtração com números binários e fazem parte do subsistema chamado Unidade Lógica Aritmética (ULA) dos computadores e microcontroladores (presentes nas calculadoras digitais).

Nesta seção, aprenderemos como realizar as operações de soma e subtração binária utilizando as portas digitais que conhecemos nas unidades anteriores. Para pôr em prática esse conhecimento, você será o responsável técnico de uma pequena empresa de sistemas embarcados. Nessa empresa, ocorrerá a eleição entre dois representantes de uma das áreas. A fim de tornar ágil o processo de votação e testar suas habilidades técnicas, seu chefe solicitou a sua ajuda para a implementação de um sistema automático de contagem de votos. Após checar os componentes disponíveis, você percebeu que possui apenas componentes lógicos somadores. E então, como desenvolver esse sistema? Será que é possível desenvolver o sistema utilizando apenas os circuitos somadores?

Vamos descobrir?

Não pode faltar

Nas unidades anteriores, estudamos o funcionamento das portas lógicas básicas e empregamos a álgebra booleana para descrever e analisar os circuitos feitos a partir das diferentes combinações que utilizam as portas lógicas. Esses circuitos podem ser classificados como *circuitos combinacionais*, pois em qualquer instante de tempo o nível lógico da saída depende apenas da combinação dos níveis lógicos presentes na entrada do circuito. Em outras palavras, esses

circuitos não possuem envolvimento com memória ou realimentação da saída para entrada.

Segundo Szanjberg (2014), a inexistência de realimentação proporciona estabilidade ao circuito e elimina qualquer incerteza nos resultados obtidos, que podem ser previstos usando-se somente as proposições da álgebra booleana.

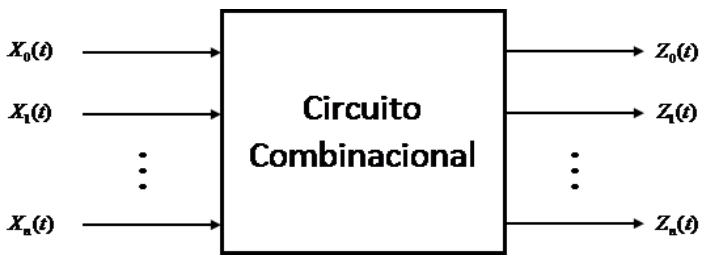
As principais características dos circuitos com a lógica combinacional são:

a) Reconhecimento de existência de pulsos. Esse tipo de circuito detecta se houve mudança de nível no sinal de entrada, uma vez que a saída depende única e exclusivamente dos níveis lógicos presentes na entrada. A cada combinação dos valores da entrada pode ser vista como uma informação diferente na saída do sistema.

b) Discriminação de pulsos. Esse tipo de circuito detecta se houve mudança na largura do pulso por meio do tempo de duração.

A Figura 3.1 ilustra um modelo genérico para o circuito combinacional.

Figura 3.1 | Modelo genérico do circuito combinacional



Fonte: adaptada de Szanjberg (2014, p. 228).

Os circuitos combinacionais são responsáveis pelas operações aritméticas dentro de um sistema digital, como computadores e calculadoras digitais, que são conhecidos como circuitos aritméticos. Esse tipo de circuito implementa operações como adição, por meio dos circuitos somadores; e subtração, por meio dos circuitos subtratores.

Os somadores são importantes circuitos para os computadores digitais ou sistemas microprocessados, visto que umas das tarefas

mais importantes executadas por esses sistemas é a operação da adição em números binários. Os somadores podem ser do tipo meio somador ou somador completo.

O circuito meio somador aceita dois dígitos binários em suas entradas e produz dois dígitos em suas saídas, um bit soma e um bit *carry*, ou "vai um".

Lembre-se

As quatro operações possíveis para adição binária são:

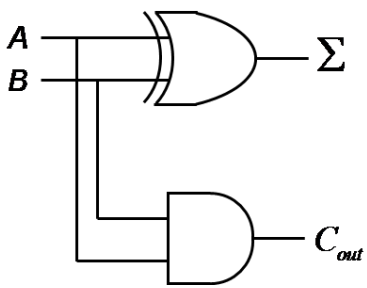
$$0+0=0$$
$$0+1=1$$
$$1+0=1$$
$$1+1=10$$

Tabela 3.1 | Tabela verdade circuito meio somador

A	B	Σ	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fonte: elaborada pelo autor.

Figura 3.2 | Diagrama lógico circuito meio somador



Fonte: elaborada pelo autor.

A partir da tabela verdade para soma binária, ilustrada na Tabela 3.1, podemos deduzir as expressões para o bit soma (Σ), resultado do circuito somador, e para o bit *carry* de saída (C_{out}) como funções das entradas. Note que o bit C_{out} é 1 apenas quando as entradas A e

B são 1. Logo, podemos representar o bit *carry* de saída como uma operação lógica AND entre as variáveis de entrada. Já a saída soma é 1 apenas se os valores de A e B forem diferentes, portanto, essa saída pode ser expressa como operação XOR (ou OU-Exclusivo) entre as variáveis de entrada. Sendo assim, as expressões para o bit soma (Σ) e bit *carry* (C_{out}) são:

$$\Sigma = A \oplus B$$
$$C_{out} = AB$$

A partir destas equações, podemos representar o diagrama lógico para o circuito meio somador, como mostra a Figura 3.2(b).

Já o circuito somador completo aceita dois bits (A, B) e um *carry de entrada* (C_{in}), e gera um bit soma (Σ) e um *carry* de saída (C_{out}) como saída do circuito aritmético. A tabela verdade de um circuito somador completo é mostrada na Tabela 3.2.

Assimile

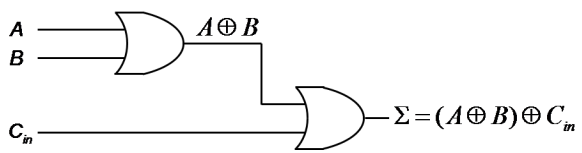
A diferença básica entre um somador completo e um meio somador é que o somador completo aceita um bit *carry* de entrada, o que aumenta as possibilidades de números a serem somados.

Tabela 3.2 | Tabela verdade circuito somador

A	B	C_{in}	Σ	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

Fonte: elaborada pelo autor.

Figura 3.3 | Circuito somador: lógica necessária para o somador de três bits



Fonte: elaborada pelo autor.

A partir do meio somador, sabemos que a soma de dois bits de entrada é obtida por meio do uso de uma XOR dessas duas variáveis. Se queremos somar mais o bit *carry de entrada*, devemos realizar uma XOR dos bits de entrada, A e B, representando parte de uma adição, com o bit C_{in} representando a outra parte da adição, como mostra a lógica representada na Figura 3.3 e conforme é expresso por:

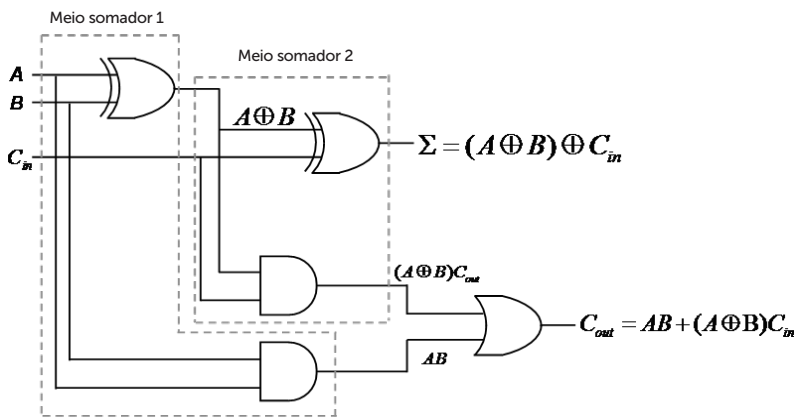
$$\Sigma = (A \oplus B) \oplus C_{in} \qquad (3.1)$$

O *carry de saída* (C_{out}) é 1 apenas quando os bis A e B forem 1, ou seja, quando as entradas para a primeira porta XOR (Figura 3.3 (b)) forem 1; ou quando um dos bits A ou B for 1 e o *carry de entrada* também for 1, isto é, quando as duas entradas para a segunda porta XOR forem 1, como pode ser analisado na tabela verdade para o circuito somador completo (Figura 3.3(a)). Dessa forma, podemos representar o *carry de saída* do somador completo como uma operação AND entre os bits da primeira porta, A com B, e pela operação AND do C_{in} com resultado do XOR da primeira porta, $A \oplus B$, como determina a expressão:

$$C_{out} = AB + (A \oplus \mathrm{B})C_{in} \qquad (3.2)$$

Dadas as Expressões 3.1 e 3.2, podemos implementar a lógica representativa para o circuito somador completo (Figura 3.4). Podemos notar que para implementar essa função é necessário utilizar dois meios somadores conectados por uma porta OR.

Figura 3.4 | Diagrama lógico para o circuito somador completo



Fonte: elaborada pelo autor.

Reflita

Vimos que o somador completo pode ser utilizado para adição de números binários com dois bits e um *carry* de entrada. Mas e se quisermos realizar a adição entre quatro bits utilizando somadores completos, como seria? Como seria a relação C_{out} de um somador com C_{in} de outro somador? Esses bits teriam alguma relação?

Exemplificando

Quais são as saídas de um somador completo para A = 1, B = 0 e $C_{in} = 0$?

Um somador completo é formado por três bits de entrada. Logo, podemos considerar que:

$$1 + 0 + 0 = 1 \text{ sem } carry$$

Considerando a tabela verdade para o circuito somador completo, podemos considerar a saída soma como $\Sigma = 1$ e o bit *carry* de saída como $C_{out} = 0$.

Outra operação igualmente importante em nossas atividades é a subtração. O circuito meio subtrator aceita dois dígitos binários em suas entradas e produz dois dígitos em suas saídas, um bit diferença e um bit de "falta um" ou "empresta um" (*borrow*).

Lembre-se

As quatro operações possíveis para subtração binária são:

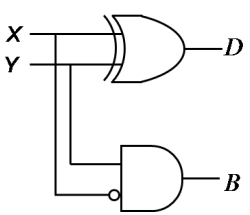
$$0 - 0 = 0$$
$$0 - 1 = 1$$
$$1 - 0 = 1$$
$$1 - 1 = 0$$

Tabela 3.3 | Tabela verdade circuito subtrator

X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Fonte: elaborada pelo autor.

Figura 3.5 | Diagrama lógico circuito meio subtrator



Fonte: elaborada pelo autor.

Assim como no circuito somador, a tabela verdade do circuito meio subtrator (Tabela 3.3) nos faz deduzir que a expressão para a saída diferença (D) pode ser expressa como operação XOR, visto que o valor de D é 1 apenas quando os bits de entrada forem diferentes. Já o bit *borrow* (B) tem valor de 1 quando a entrada X for 0 e a entrada Y for 1. Logo, podemos representar o bit de saída B como uma operação lógica AND cuja entrada X deverá ser negada, ou barrada, como mostra a Figura 3.5.

As expressões para as saídas do circuito meio subtrator são dadas por:

$$D = X \oplus Y$$

$$B = \overline{X}Y$$

Já o circuito subtrator completo possibilita a subtração de dois bits (X e Y) e um bit "empresta um" de entrada (B_{in}), e gera os bits diferença (D) e "empresta um" de saída (B_{out}). Portanto, esse circuito possui três bits de entrada (X, Y e B_{in}) e dois bits de saída (D e B_{out}). A tabela verdade de um circuito subtrator completo é mostrada na Tabela 3.4.

Tabela 3.4 | Tabela verdade circuito subtrator completo

A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Fonte: elaborada pelo autor.

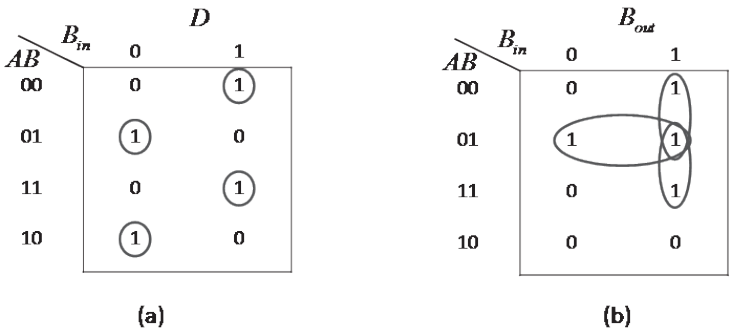
A fim de treinarmos uma forma distinta de obter a expressão para o sistema, para a obtenção da expressão lógica de saída para D e B_{out} utilizaremos a análise do mapa de Karnaugh. Sendo assim, considerando o mapa apresentado na Figura 3.6(a), notamos que a expressão pode ser dada para saída diferença (D) como:

$$D = X \oplus Y \oplus B_{in} \qquad (3.3)$$

Já para saída (B_{out}), podemos considerar o mapa da Figura 3.6 (b), cuja expressão lógica de saída é dada como:

$$B_{out} = \overline{X}Y + \overline{X}B_{in} + YB_{in} \quad (3.4)$$

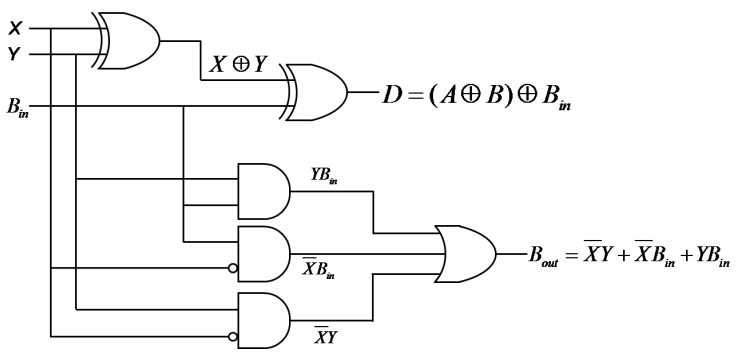
Figura 3.6 | Análise do mapa Karnaugh para: (a) saída D; e (b) saída B_{out}



Fonte: elaborada pelo autor.

Dadas as Expressões 3.3 e 3.4, podemos implementar a lógica representativa para o circuito subtrator completo (Figura 3.7). Note que o subtrator completo pode ser apresentado pela combinação de um par de meios subtratores.

Figura 3.7 | Diagrama lógico para o circuito subtrator completo



Fonte: elaborada pelo autor.

Na prática, o circuito somador pode ser considerado como subtrator, considerando o método de subtração por complemento de 2. Dessa forma, a saída da subtração pode ser produzida pelo circuito somador, uma vez que uma subtração pode ser considerada como a soma de um número com o complemento de 2 de outro número.

Pesquise mais

Além das operações de adição e subtração, podemos realizar as operações de multiplicação e divisão por meio dos circuitos lógicos combinacionais.

Para saber um pouco mais sobre os circuitos aritméticos combinacionais que realizam essas operações, acesse o link disponível em: <http://www.inf.ufsc.br/~j.guntzel/isd/isd3.pdf>. Acesso em: 15 out. 2017.

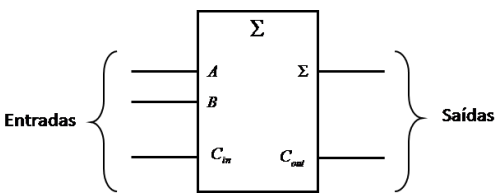
Sem medo de errar

Você é o responsável técnico de uma pequena empresa de sistemas embarcados. Nessa empresa, ocorrerá a eleição entre dois representantes de uma das áreas. A fim de tornar ágil o processo de votação e testar suas habilidades técnicas, seu chefe solicitou a sua ajuda para implementação de um sistema automático de contagem de votos. Após checar os componentes disponíveis, você percebeu que possui apenas componentes lógicos somadores.

Depois de uma revisão das funções da lógica combinacional aritmética, circuito somador e componente disponível para o desenvolvimento do sistema solicitado, você teve uma brilhante ideia para a implementação do sistema de votação.

Um circuito somador completo possui três bits de entrada e dois bits de saída, e está representado de forma genérica pelo diagrama em blocos na Figura 3.8.

Figura 3.8 | Representação genérica do circuito somador completo em bloco

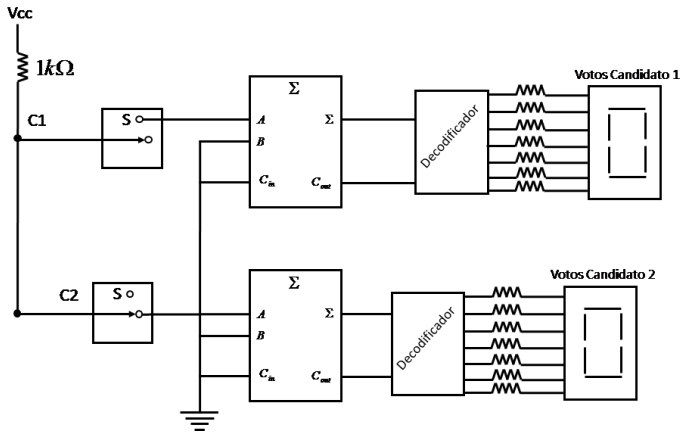


Fonte: adaptada de Floyd (2007, p. 314).

Podemos utilizar um circuito somador para adicionar os votos para cada candidato. Sendo assim, o eleitor que optar pelo candidato 1 deverá apertar a chave C1. Isso faz com que o bit A tenha valor lógico 1, enquanto que os níveis lógicos B e C_{in} permanecem em 0. Sendo assim, a saída Σ do somador completo passará a ser 1 e contará um

voto para o candidato 1. A mesma lógica é realizada para o candidato 2: o eleitor que preferir votar no candidato 2 deverá apertar a chave ou (botão) C2, que passará o bit A2 para o nível lógico 1, enquanto que os níveis lógicos B2 e C_{in2} permanecem em 0. A Figura 3.9 mostra o circuito completo para o sistema desenvolvido.

Figura 3.9 | Sistema de votação utilizando somadores completos



Fonte: adaptada de Floyd (2007, p. 323).

Vale salientar que o resistor da entrada de cada somador completo para o GND garante que cada entrada terá nível BAIXO quando a chave estiver na posição neutra (lógica CMOS é usada). Quando a chave é comutada para a posição S, um nível ALTO (Vcc) é aplicado na entrada do somador completo.

Faça valer a pena

1. Os somadores são importantes circuitos para os computadores digitais ou sistemas microprocessados, visto que umas das tarefas mais importantes executadas por esses sistemas é a operação da adição em números binários.

Para o somador completo, qual é o nível lógico da saída de cada de cada porta se o nível lógico das entradas forem A = 1, B = 0 e $C_{in} = 1$?

a) $\Sigma = 0$ e $C_{out} = 0$.

b) $\Sigma = 0$ e $C_{out} = 1$.

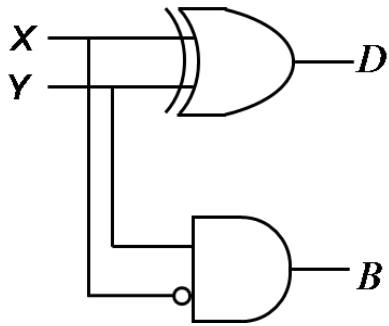
c) $\Sigma = 1$ e $C_{out} = 0$.

d) $\Sigma = 1$e $C_{out} = 1$.

e) $\Sigma = 1$ e $C_{out} = X$.

2. Os circuitos combinacionais são responsáveis pelas operações aritméticas dentro de um sistema digital, como computadores e calculadoras digitais, que são conhecidos como circuitos aritméticos.
Dado o seguinte diagrama lógico:

Figura 3.10 | Diagrama lógico



Fonte: elaborada pelo autor.

Podemos afirmar que o diagrama representa:
a) Circuito meio somador.
b) Circuito meio subtrator.
c) Circuito somador completo.
d) Circuito subtrator completo.
e) Circuito multiplicador.

3. Frequentemente, em nossas tarefas do cotidiano, nos cálculos científicos ou nos negócios, é fundamental realizarmos operações aritméticas como a soma e a subtração. Muitas vezes, utilizamos as calculadoras digitais ou computadores para realizar essa atividade. Ao fazermos isso, estamos utilizando os circuitos aritméticos com lógica combinacional.
A respeito dos circuitos lógicos combinacionais, podemos afirmar que:
I. O nível lógico de saída depende apenas da combinação dos níveis lógicos presentes na entrada do circuito.
II. Apresentam estabilidade ao circuito e elimina qualquer incerteza em relação aos resultados obtidos.
III. Podem ser formados por portas lógicas básicas e *flip-flops*.
Dadas as sentenças, é correto afirmar que:
a) Apenas I e II estão corretas.
b) Apenas I e III estão corretas.
c) Apenas II e III estão corretas.
d) Todas estão corretas.
e) Apenas I está correta.

Seção 3.2

Circuito comparador, codificador e decodificador

Diálogo aberto

Na seção anterior, demos início aos estudos dos diversos circuitos aplicados às nossas atividades utilizando os circuitos combinacionais. Vimos que esse tipo de circuito pode ser empregado para realizar operações aritméticas como soma e subtração a partir das portas digitais que conhecemos na Unidade 2.

Nesta seção, daremos continuidade ao nosso estudo da lógica combinacional. Veremos como os circuitos digitais podem ser aplicados para comparação entre os números binários, especificando as desigualdades ou a igualdade entre eles. Além disso, conheceremos os circuitos classificados como codificadores e decodificadores, que são empregados quando é necessário realizar a conversão, ou passagem, de um tipo de código para outro. Esse tipo de circuito é amplamente utilizado para a lógica dos computadores digitais atuais.

Sendo assim, para pôr esse conhecimento em prática, devemos lembrar que você é o responsável técnico de uma pequena empresa de sistemas embarcados. Após a implementação bem-sucedida do sistema de votação, seu chefe pediu que você desenvolvesse um sistema simplificado em que o usuário entre com um código de quatro bits, que ele receberá em casa, via e-mail. Esse código irá gerar a posição dele na fila de espera para que seja atendido em uma determinada empresa. Além disso, o código deve aparecer em um display, para confirmação do usuário. A empresa contratante tem por objetivo que cada funcionário tenha até nove atendimentos, a serem realizados em espera, em um determinado período de tempo. Como podemos desenvolver esse sistema utilizando apenas os circuitos combinacionais? Qual tipo de circuito devemos utilizar para esse desenvolvimento?

Pronto para mais este desafio? Esperamos que esteja animado! Desejamos bons estudos e um ótimo trabalho!

Não pode faltar

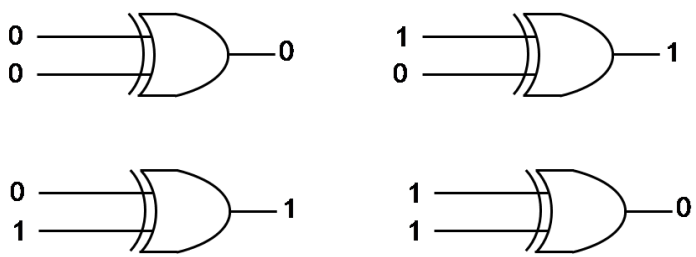
Frequentemente, em nosso cotidiano, utilizamos o método de comparação para realizar alguma atividade. Por exemplo: para sabermos o peso de algum produto, podemos realizar uma comparação entre o valor de um peso já conhecido com o que ainda não conhecemos e identificar se eles são iguais, diferentes, se são maiores ou menores. Além disso, nos sistemas cujo controle de temperatura é importante, como nas indústrias químicas e aviárias, é fundamental a comparação dos valores de temperatura medidos pelos sensores digitais, a fim de manter o bom funcionamento dos processos.

A comparação entre as magnitudes de dois números binários para determinar a relação comparativa entre eles pode ser realizada através dos circuitos combinacionais do tipo comparador. Em sua forma mais simples, um circuito comparador determina se dois números são iguais. Para isso, é necessária a utilização da porta XOR (ou OU-Exclusiva).

Lembre-se

A porta lógica XOR tem o nível lógico da saída 1 se os dois bits de entrada forem diferentes. Contudo, se os bits de entrada forem iguais, o nível lógico de saída será 0, como mostra a Figura 3.11.

Figura 3.11 | Lógica combinacional porta XOR

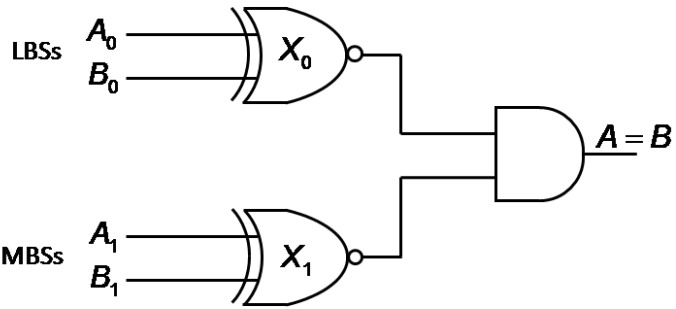


Fonte: adaptada de Floyd (2007, p. 328).

Dessa forma, se quisermos comparar "n" números binários de dois bits de cada é necessário utilizarmos "n" portas XOR. Por exemplo, se quisermos comparar dois números binários (A e B) de dois bits cada um (A_1A_0 e B_1B_0), é necessária a utilização de duas portas XOR. Os dois bits menos significativos, LSBs, (B_0 e B_0) dos dois números

são comparados pela porta X_0; e os bits mais significados, MSBs, (A_1 e B_1), são comparados pela porta X_1, como mostra Figura 3.12 (FLOYD, 2007).

Figura 3.12 | Diagrama lógico comparador de igualdade entre dois números de dois bits



Fonte: adaptada de Floyd (2007, p. 328).

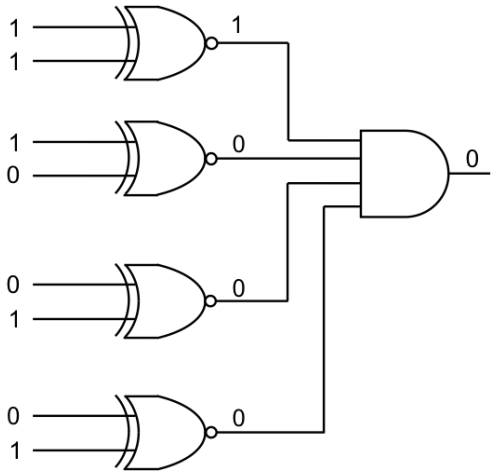
Na Figura 3.12, note que a saída da porta XOR possui dois inversores. Perceba que essa combinação de portas também é conhecida como porta XNOR, como vimos na Unidade 1. Sendo assim, se os dois números forem iguais, ou seja, se os bits correspondentes forem iguais, o nível lógico da saída da porta lógica XOR será 1, e a porta AND terá sua saída também com nível lógico 1. Todavia, quando os números forem diferentes, ou se um dos pares dos bits correspondentes não forem iguais, a saída lógica para uma das portas XNOR será 0 e, por sua vez, a saída para a porta AND também terá nível lógico 0. Portanto, a saída da porta AND indica a igualdade, caso o nível lógico da saída for 1, ou desigualdade, caso o nível lógico for 0, entre os dois números binários de dois dígitos.

Vale ressaltar que esse comparador básico pode ser expandido para o número de bits que for necessário para comparação entre dois números. O único ponto de atenção é que a quantidade de bits dos números a serem comparados devem ser iguais entre si.

Exemplificando

Dados os conjuntos numérico binários $A = 1011$ e $B = 0011$, determine o diagrama lógico comparador básico de igualdade e a saída desse circuito.

Figura 3. 13 | Diagrama lógico do comparador básico de quatro bits



Fonte: elaborada pelo autor.

Além de comparar se os binários são iguais ou não, ainda é possível por meio dos circuitos combinacionais verificar qual grandeza, entre as comparadas, é maior. Isso é possível pelo uso dos circuitos combinacionais do tipo meio comparador binário, ou circuito de desigualdade.

Esse tipo de comparador compara duas entradas com quantidades binárias, como A e B, e gera saídas para indicar qual delas é maior, $A_0 > B_0$ ou $B_0 > A_0$, ou se são iguais. Na prática, nunca teremos dois desses casos sendo verdadeiros ao mesmo tempo. Sendo assim, a tabela verdade para esse tipo de circuito é representada pela Tabela 3.5.

Tabela 3.5 | Tabela verdade do circuito meio comparador binário

A	B	A>B	A<B	A=B
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Fonte: elaborada pelo autor.

A partir da Tabela 3.5, podemos deduzir as seguintes expressões lógicas:

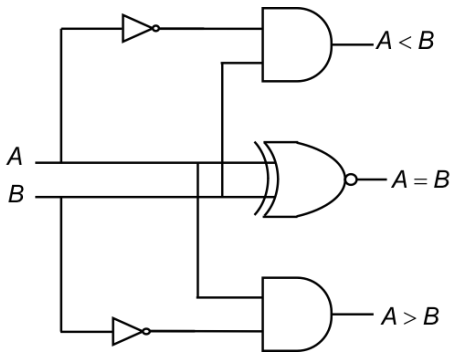
$$A > B = A\overline{B}$$

$$A < B = \overline{A}B$$

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

A partir dessas expressões lógicas, podemos deduzir o diagrama lógico para o circuito meio comparador, como mostra a Figura 3.14.

Figura 3.14 | Diagrama lógico do circuito meio comparador



Fonte: elaborada pelo autor.

Pesquise mais

O CI 74HC85 é um circuito integrado formado pelo circuito combinacional do tipo meio comparador. Para saber mais sobre o diagrama de pinos e símbolos lógico desse circuito integrado, basta acessar o *datasheet*, ou folha de dados, disponível em: <https://www.eng.tau.ac.il/~shavitt/courses/DigLogSys/74LS85.pdf>. Acesso em: 16 out. 2017.

Outro tipo de circuito lógico combinacional muito utilizado é o decodificador. Esse tipo de circuitos digitais determina uma saída específica para cada combinação única de bits ou código de entradas.

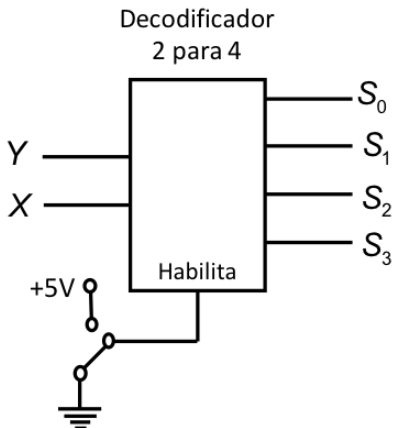
De uma forma geral, um decodificador converte as informações binárias de n linhas de entradas para um máximo de 2^n linhas únicas de saída. Para um decodificador 2 para 4, por exemplo, com apenas 2 linhas bits de entrada (Y, X), é possível termos, no máximo, 4 linhas de saída (S_3, S_2, S_1, S_0), como mostra a Tabela 3.6.

Tabela 3.5 | Tabela verdade para o circuito decodificador 2 para 4

Habilita	Y	X	S_0	S_1	S_2	S_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Fonte: elaborada pelo autor.

Figura 3.15 | Diagrama lógico de blocos do circuito decodificador 2 para 4



Fonte: adaptada de Szanjberg (2014, p. 263).

Assimile

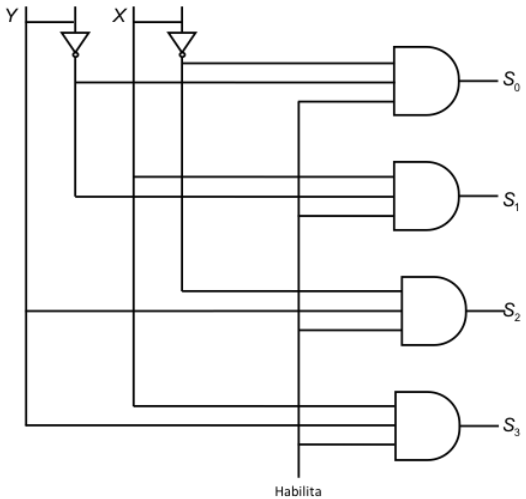
Um decodificador converte as informações binárias de *n* linhas de entradas para um máximo de 2^n linhas únicas de saída.

Note na Figura 3.15 a presença da entrada adicional "habilita" (ou *Enable*). Esta, quando em nível alto, ou igual 1, determina que a entrada do decodificador esteja no estado ativado ou, em outras palavras, quando o habilita = 1 (ou E = 1), a entrada está ativada, ou habilitada, para determinar as saídas especificadas pelo código dos bits de entrada. Porém, quando o habilita = 0 (ou E = 0), a entrada do decodificador estará no estado desativado, ou simplesmente

desabilitada, e não importa quais forem os bits de entrada (Y, X), pois as saídas estarão sempre em nível baixo.

A partir da tabela verdade (Tabela 3.6), é possível determinar o diagrama lógico desse decodificador, como mostra a Figura 3.16.

Figura 3.16 | Diagrama lógico para o decodificador 2 para 4



Fonte: adaptada de Szanjberg (2014, p. 264).

Reflita

Em um CI decodificador de 2 para 4, é possível codificar apenas dois bits. Contudo, como seria se precisássemos codificar três bits? Quantos CIs seriam necessários para essa situação?

Pesquise mais

Diversas combinações de n linhas de entradas para um máximo de 2^n linhas únicas de saída são possíveis a depender da aplicabilidade do circuito combinacional requerido, por exemplo: decodificador de 3 para 8 ou BCD-decimal. Este último se caracteriza por ter quatro entradas e dez linhas de saída cuja aplicação é uma conversão binário-decimal.

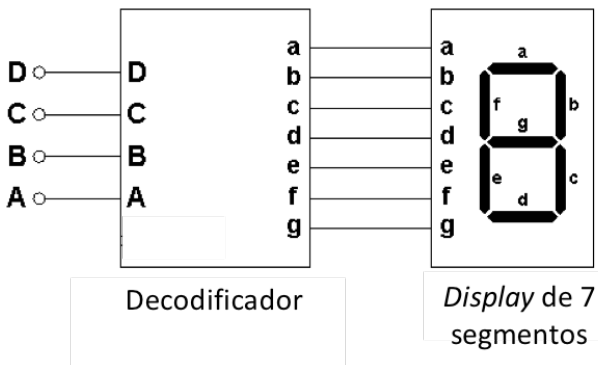
Para saber um pouco mais sobre esse tipo de decodificador, acesse a seção 6.5.5 do livro:

SZAJNBERG, Mordka. **Eletrônica digital**: teoria, componentes e aplicações. 1. ed. Rio de Janeiro: LTC, 2014. O livro também está disponível em nossa biblioteca virtual, no site:

<https://biblioteca-virtual.com/detalhes/parceiros/5>. Acesso em: 20 out. 2017.

Dentre as possíveis aplicações dadas aos decodificadores, uma das mais utilizadas é o decodificador de BCD para sete segmentos. Este tipo de circuito aceita o código binário, ou BCD, como entrada, podendo ativar um display de até sete segmentos de LEDs, identificados como *a, b, c, d, e, f, g*, como ilustrado na Figura 3.17.

Figura 3.17 | Decodificador BCD – Display de sete segmentos



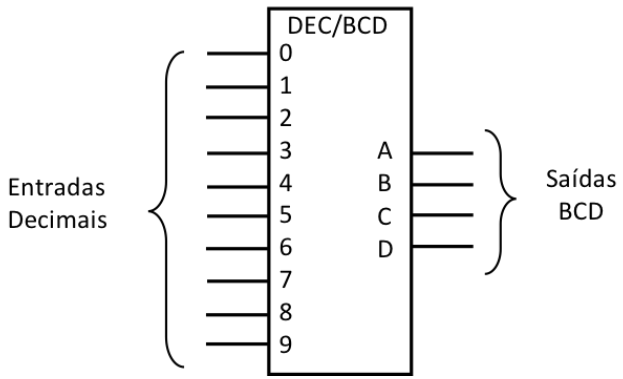
Fonte: adaptada de <http://macao.communications.museum/por/exhibition/secondfloor/MoreInfo/Displays.html>. Acesso em: 16 out. 2017.

Nesse tipo de circuito, as entradas BCD (A, B, C, D da Figura 3.17) são convertidas nas entradas dos setes segmentos (a, b, c, d, e, f, g) pelo uso do decodificador. Cada segmento de saída representa um LED, que pode ser acionado através da ligação do ânodo com uma fonte 5 volts, permitindo, assim, escrever números de 0 a 9 e algumas letras ou sinais neste display, ou mostrador.

Já um circuito codificador é um circuito lógico que realiza a operação "inversa" ao decodificador, ou seja, esse tipo de circuito aceita um nível ativo em uma de suas entradas representando um dígito, tal como dígito decimal ou octal, e o converte em uma saída codificada, tal como binário. Em outras palavras, os codificadores convertem um determinado tipo de código em suas entradas para um outro formato codificado.

Dentre os tipos mais utilizados, temos o codificador de decimal para BCD. Esse tipo de codificador possui dez entradas, uma para cada dígito decimal, e quatro saídas para o código BCD. Logo, esse codificador é classificado como codificador 10 para 4, como mostra a Tabela 3.7.

Figura 3.18 | Símbolo lógico para o circuito codificador decimal para BCD



Fonte: elaborada pelo autor.

Tabela 3.7 | Tabela verdade circuito codificador de decimal para BCD

	Código BCD			
Dígito decimal	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Fonte: adaptada de Floyd (2007, p. 341).

O código para esse tipo de codificador é apresentado na Figura 3.18. Segundo Floyd (2007), a partir desse código, podemos relacionar cada dígito decimal com o código BCD e verificar a lógica utilizada. Por exemplo, os dígitos 8 e 9 decimais possuem sempre o nível 1 para

o bit mais significativo do código BCD. Logo, podemos representar a expressão OR para o bit A_3 em termos dos dígitos decimais como:

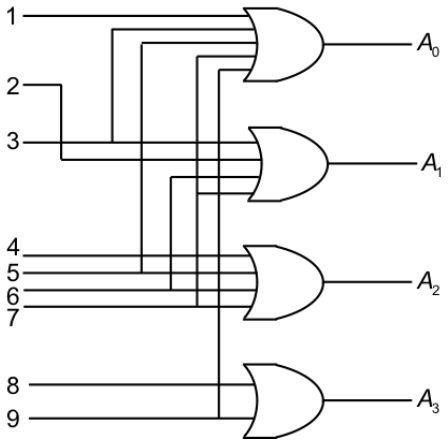
$$A_3 = 8 + 9$$

O A_0, por sua vez, pode ser escrito como:

$$A_0 = 1 + 3 + 5 + 7 + 9$$

E assim sucessivamente para os outros códigos BCD, resultando no diagrama lógico apresentado na Figura 3.19.

Figura 3.19 | Diagrama lógico decodificador decimal BCD



Fonte: adaptada de Floyd (2007, p. 341).

Pesquise mais

Outro codificador também muito utilizado nas atividades práticas é o codificador binário-Gray. Para saber um pouco mais sobre esse tipo de codificador, acesse a seção 6.5.5, capítulo 6 do livro:

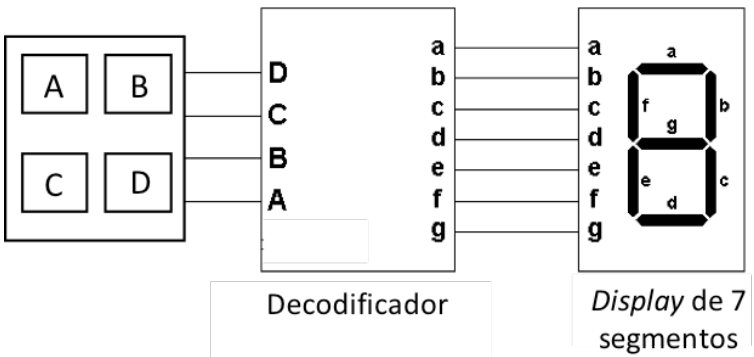
SZAJNBERG, Mordka. **Eletrônica digital**: teoria, componentes e aplicações. 1. ed. Rio de Janeiro: LTC, 2014. Esse livro está disponível em nossa biblioteca virtual, no site: <https://biblioteca-virtual.com/detalhes/parceiros/5>. Acesso em: 20 out. 2017.

Sem medo de errar

Você é o responsável técnico de uma empresa de sistemas embarcados e seu chefe solicitou que você desenvolvesse um sistema simplificado em que o usuário entre com um código de

quatro bits, que ele receberá em casa, via e-mail. Esse código irá gerar a posição dele na fila de espera para que seja atendido em uma determinada empresa. Além disso, esse código deve aparecer em um display, para confirmação do usuário. A empresa contratante tem por objetivo que cada funcionário tenha até nove atendimentos, a serem realizados em espera, em um determinado período de tempo. Como podemos desenvolver esse sistema utilizando apenas os circuitos combinacionais? Qual tipo de circuito devemos utilizar para esse desenvolvimento?

Figura 3.20 | Circuito mostrador de posição



Fonte: elaborada pelo autor.

Após uma revisão nos conceitos de decodificadores, você percebeu que o problema é simples de ser solucionado ao utilizar um decodificador e um display de sete segmentos. Como o código que o usuário recebe é formado por quatro bits, representados no teclado por A, B, C e D, estes são codificados, ou convertidos, em um código de sete bits (de *a* até *g*) que acenderão os LEDs em um display que mostra o número decimal da posição em que o usuário está na fila de espera. Como cada funcionário pode acumular até nove atendimentos, apenas um display é o suficiente para implementação desse sistema.

Logo, utilizando apenas circuitos combinacionais, você conseguiu resolver mais um simples problema.

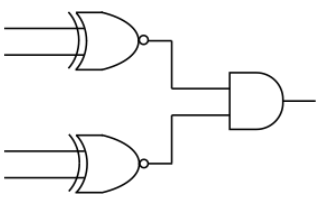
Faça valer a pena

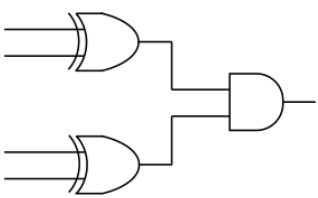
1. Os *circuitos combinacionais* são tipos de circuitos caracterizados por apresentar, em qualquer instante de tempo, o nível lógico da saída depende apenas da combinação dos níveis lógicos presentes na entrada do circuito.

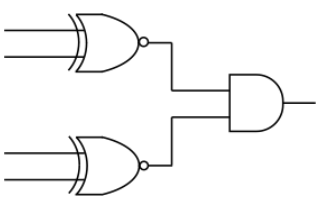
Sobre os circuitos combinacionais, analise as seguintes afirmações:
I. Podem ser utilizados para converter um tipo de código em suas entradas para um outro formato codificado.
II. Podem ser utilizados para determinar se um número binário é maior ou menor que outro.
III. Quando utilizados como decodificadores, convertem as informações binárias de *n* linhas de entradas para um máximo de $A_3 = 8 + 9$ linhas únicas de saída.
Dadas as sentenças, é correto afirmar que:

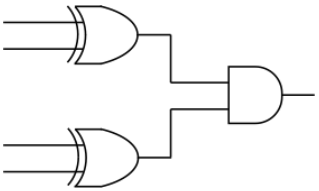
a) Apenas I e II estão corretas.
b) Apenas I e III estão corretas.
c) Apenas II e III estão corretas.
d) Todas estão corretas.
e) Apenas a I está correta.

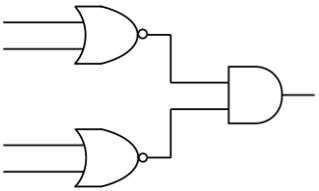
2. A comparação entre as magnitudes de dois números binários para determinar a relação comparativa entre eles pode ser realizada através dos circuitos combinacionais do tipo comparadores.
Dados os números binários 01 e 10, determine qual circuito lógico combinacional pode realizar uma comparação entre os números e qual é o nível lógico presente na saída:

a)  , nível lógico de saída 0.

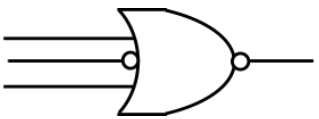
b)  , nível lógico de saída 0.

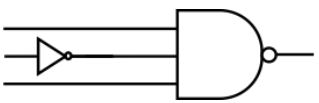
c)  , nível lógico de saída 1.

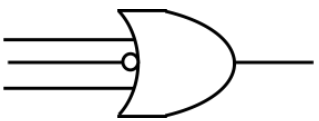
d)  , nível lógico de saída 1.

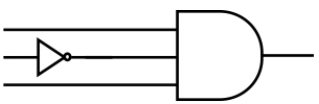
e)  , nível lógico de saída 0.

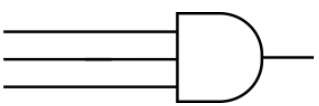
3. Os decodificadores são tipos de circuitos combinacionais que determinam uma saída especificada para cada combinação única de bits, ou código, de entradas.
Determine a melhor lógica necessária para decodificar o número binário 101 produzindo um nível alto na saída:

a) 

b) 

c) 

d) 

e) 

Seção 3.3

Circuitos multiplexado e demultiplexado

Diálogo aberto

Na seção anterior, vimos os circuitos combinais aplicados para comparação entre números binários, especificando as desigualdades ou igualdade entre eles. Além disso, conhecemos os circuitos classificados como codificadores e decodificadores, empregados quando é necessário realizar a conversão, ou passagem, de um tipo de código para outro.

Nesta última seção referente aos circuitos combinacionais, estudaremos detalhadamente sobre os circuitos multiplexadores e demultiplexadores. Veremos como são formados e algumas aplicabilidades desse tipo de circuito digital, muito utilizado em sistemas de comunicação.

Sendo assim, para pôr este conhecimento em prática devemos lembrar que você é o responsável técnico de uma pequena empresa de sistemas embarcados. Após a implementação bem-sucedida de alguns projetos iniciais, seu chefe solicitou que você desenvolvesse um sistema simples e financeiramente rentável de monitoração de segurança de um banco em que o estado aberto ou fechado de oito portas deve ser monitorado. Cada porta controla o estado de uma chave, e é necessário mostrar o estado de cada chave por meio de LEDs montados em um painel remoto de monitoração na sala de segurança. Ao checar o almoxarifado, você percebe que possui apenas CIs 74HC151 e 74HC138, que são referentes aos circuitos multiplexadores e demultiplexadores. E agora, como resolver mais essa situação? Será possível utilizar a lógica combinacional para implementar esse sistema? Pronto para mais esse desafio?

Desejamos bons estudos e um ótimo trabalho.

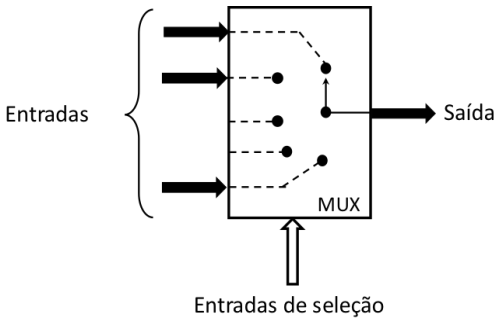
Não pode faltar

Algumas vezes, temos múltiplas entradas em nossos dispositivos eletrônicos e apenas uma única saída. Por exemplo, sistemas de amplificador de som modernos podem possuir chaves que

selecionam músicas por diferentes fontes, como: *bluetooth*; televisão; sintonizador de rádio; áudio de DVD; uma ligação telefônica na qual há duas pessoas falando, duas fontes de dados e apenas uma linha de transmissão. Isso só é possível pelo uso dos multiplexadores. Esse tipo de circuito combinacional, também conhecido como seletor de dados, tem a função de associar uma ou mais entradas de dados a uma saída singular.

Segundo Tocci (2011), um multiplexador (ou Mux) digital é um circuito lógico que recebe diversos dados digitais de entrada e seleciona um, em determinado instante, para transferi-lo para a saída. O envio do dado de entrada desejado para a saída é controlado pelas entradas de *seleção ou controle*. A Figura 3.21 mostra o diagrama funcional genérico de um circuito multiplexador.

Figura 3.21 | Diagrama funcional do mutilplexador



Fonte: adaptada de Tocci (2011, p. 520).

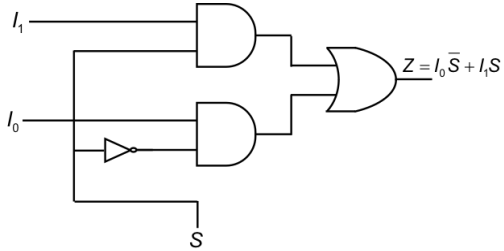
O circuito multiplexador básico possui duas entradas (I0 e I1) de dados. Logo, é necessário apenas uma linha de seleção (S). O nível lógico aplicado à chave seletora determina a porta AND a ser habilitada. A resposta dessa porta lógica é enviada a uma porta OR para então o canal de dado selecionado ser expresso na saída (Z), como mostra a Figura 3.22.

Tabela 3.8 | Tabela verdade do circuito Mux básico

S	Saída
0	I_0
1	I_1

Fonte: elaborada pelo autor.

Figura 3.22 | Diagrama lógico do Circuito MUX básico



Fonte: adaptada de Tocci (2011, p. 521).

Outro tipo de circuito multiplexador que encontramos na prática é o circuito com quatro entradas. Para esse circuito, são necessárias duas linhas de seleção pois, com dois bits, qualquer uma das quatro linhas de entrada de dados pode ser selecionada.

Assimile

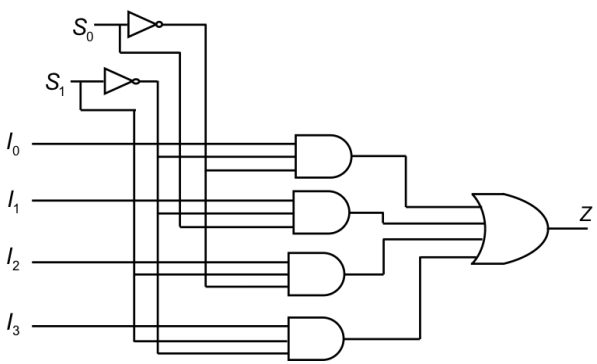
São necessárias n linhas de seleção para 2^n dados de entrada em um circuito Mux.

Tabela 3.9 | Tabela verdade para o circuito Mux de 1 para 4

S_1	S_0	Entrada Selecionada
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Fonte: elaborada pelo autor.

Figura 3.23 | Diagrama lógico para o circuito Mux de 1 para 4



Fonte: elaborada pelo autor.

Ao analisar a tabela apresentada na Figura 3.23(a), percebe-se que, assim como no circuito anterior, a entrada de seleção permite que uma determinada entrada seja selecionada. Além disso, a partir dessa tabela, podemos deduzir as expressões lógicas para saída em termos da entrada de dados e das entradas de seleção. A saída de dados é igual I_0, apenas se $S_1 = 0$ e $S_0 = 0$, logo: $I_0\overline{S_1}\overline{S_0}$. Já para I_1, apenas se $S_1 = 0$ e $S_0 = 1$, logo: $I_1\overline{S_1}S_0$. Para I_2, apenas se $S_1 = 1$ e $S_0 = 0$, logo: $I_2S_1\overline{S_0}$ e por fim para I_3, apenas se $S_1 = 1$ e $S_0 = 1$, logo: $I_3S_1S_0$.

Esses termos podem então ser relacionados por uma porta OR, resultando na expressão:

$$Z = I_0\overline{S_1}\overline{S_0} + I_1\overline{S_1}S_0 + I_2S_1\overline{S_0} + I_3S_1S_0$$

A implementação dessa equação requer quatro portas AND de três entradas, uma porta OR de quatro entradas e dois inversores para girar a chave seletora, como ilustra a Figura 3.23(b).

Pesquise mais

Os circuitos multiplexadores podem apresentar diferentes quantidades de entrada de dados. Por exemplo, além desses dois já apresentados, existem circuitos multiplexadores com oito entradas de dados, também muito aplicados na prática. Para saber mais sobre esses circuitos acesse o capítulo 9 do livro:

TOCCI, Ronald J. **Sistemas digitais**: princípios e aplicações. 11. ed. São Paulo: Pearson Prentice Hall, 2011.

Além das entradas de dados, de seleção e saída, os CIs que contêm os mutiplexadores também são formados por uma entrada denominada habilitação (E). Esta habilita a saída, ou seja, quando é aplicado o nível lógico correto a essa entrada, é permitido que o dado da entrada selecionada passe para a saída. Caso contrário, nenhum dado é visto na saída do circuito multiplexador, independentemente do código na entrada de seleção, isto é, a saída está desabilitada.

Exemplificando

Dado um sistema com quatro entradas, determine o diagrama lógico para um circuito multiplexador com a entrada de habilitação (E), que é habilitada com nível lógico 0 em sua entrada.

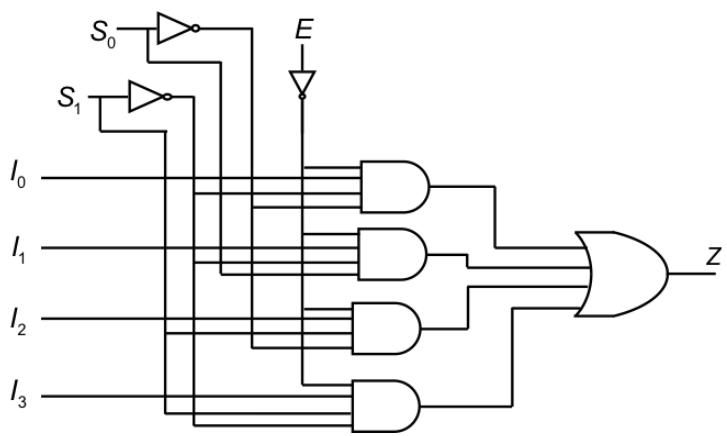
Para um sistema com quatro entradas são necessários dois níveis lógicos para a chave seletora. Como a entrada de habilitação tem nível lógico 0, a tabela verdade para esse sistema é ilustrada na Tabela 3.10.

Tabela 3. 10 | Tabela de seleção para o circuito Mux 1 de 4

E	S_1	S_0	Entrada Selecionada
0	0	0	I_0
1	0	0	0
0	0	1	I_1
1	0	1	0
0	1	0	I_2
1	1	0	0
0	1	1	I_3
1	1	1	0

Fonte: elaborada pelo autor.

Figura 3.24 | Diagrama lógico de circuito Mux 1 de 4



Fonte: elaborada pelo autor.

Assim como para o circuito mux de 1 de 4 sem entrada "Habilita", o diagrama lógico do processo será formado por quatro portas AND, mas agora serão necessárias quatro entradas, sendo uma a mais para representação da entrada E; e uma porta OR também de quatro entradas. Esse diagrama lógico é apresentado na Figura 3.24(b), e a equação é dada por:

$$Z = \overline{E}I_0\overline{S_1}\,\overline{S_0} + \overline{E}I_1\overline{S_1}S_0 + \overline{E}I_2S_1\overline{S_0} + \overline{E}I_3S_1S_0$$

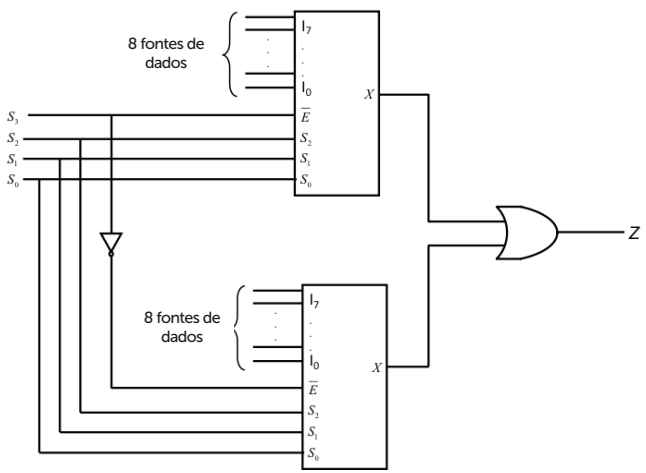
Em algumas situações práticas, temos mais entradas de dados do que permite um multiplexador real. Nesses casos, é possível combinar os multiplexadores para ajustar a quantidade de dados às entradas, por meio da entrada habilita.

Exemplificando

Suponha que você tem 16 fontes de dados diferentes como entrada e apenas CIs multiplexadores com oito entradas. Como você poderia enviar esses dados em apenas um canal de comunicação?

Como os multiplexadores são de 16 entradas, temos quatro chaves seletoras. Além disso, pela quantidade de fontes de dados que pode transmitir informações, será necessário utilizar dois CIs multiplexadores, como mostra Figura 3.25.

Figura 3.25 | Multiplexadores 8 para 1 combinados para formar multiplexador de 16 entradas



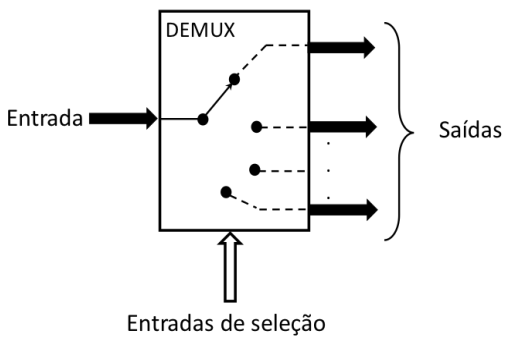
Fonte: elaborada pelo autor.

Nesta figura, o circuito funciona como um multiplexador de 16 entradas. As quatro entradas de seleção ($S_3S_2S_1S_0$) selecionam uma das 16 fontes de dados para ser enviadas para a saída Z.

A entrada S_3 determinará o multiplexador que será habilitado. Quando $s_3 = 0$, o multiplexador da parte superior é habilitado e $S_2S_1S_0$ determinam a entrada de dados que será transmitida para a saída passando pela porta OR até Z. Quando $S_3 = 1$, o multiplexador da parte inferior é habilitado, e $S_2S_1S_0$ selecionam uma das entradas de dados que serão enviadas à saída. Assim, é possível ajustar os multiplexadores à quantidade de fontes de entrada necessárias.

Os circuitos combinacionais demultiplexador (Demux) realizam a operação inversa ao mux. Os circuitos classificados como demux recebem uma única entrada e a distribuem para várias entradas. Em outras palavras, esse tipo de circuito recebe informações digitais a partir de uma linha e as distribui para um determinado número de linhas de saída. Por essa razão, o demultiplexador também é conhecido como distribuidor de dados. A Figura 3.26 mostra o diagrama funcional genérico de um circuito demultiplexador.

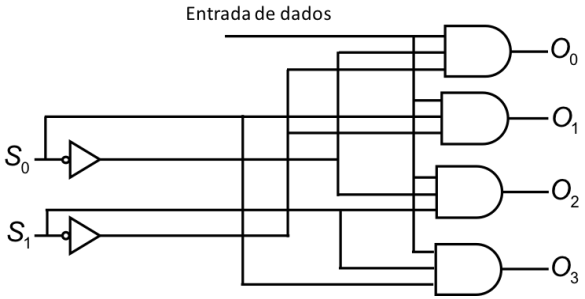
Figura 3.26 | Diagrama funcional do demutilplexador



Fonte: elaborada pelo autor.

A Figura 3.27 ilustra um diagrama lógico de um demultiplexador de uma linha para quatro linhas. A linha de entrada de dados está conectada nas quatro portas AND. As duas linhas de seleção de dados habilitam uma porta de cada vez, e os dados que aparecem na linha de entrada de dados passam, através da porta selecionada, para a linha de saída de dados associada.

Figura 3.27 | Demultiplexador de uma para quatro linhas

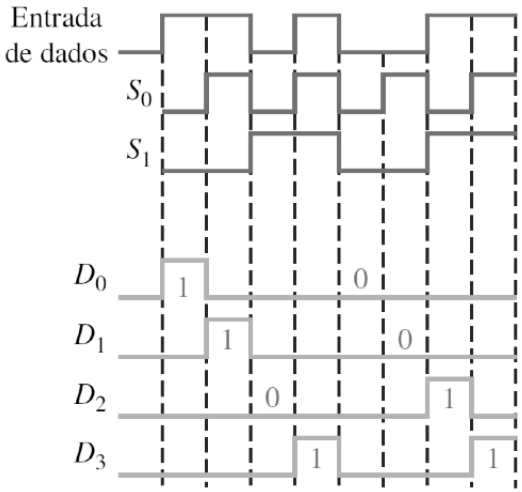


Fonte: elaborada pelo autor.

Exemplificando

Considerando a forma de onda de entrada de dados em série e as entradas de seleção de dados (S_0 e S_1), determine as formas de onda de saída de dados para um demutiplexador de quatro linhas de saída.

Figura 3.25 | Multiplexadores 8 para 1 combinados para formar multiplexador de 16 entradas



Fonte: Floyd (2011).

Note que as linhas das entradas de seleção obedecem a uma sequência binária de forma que cada bit sucessivo de entrada é direcionado para D_0, D_1, D_2 e D_3 na sequência, como mostra a Figura 3.28.

Reflita

Agora que já conhecemos os demux e os decodificadores, apresentados na Seção 2 desta unidade, podemos dizer que um decodificador pode ser usado como um demultiplexador? De que forma?

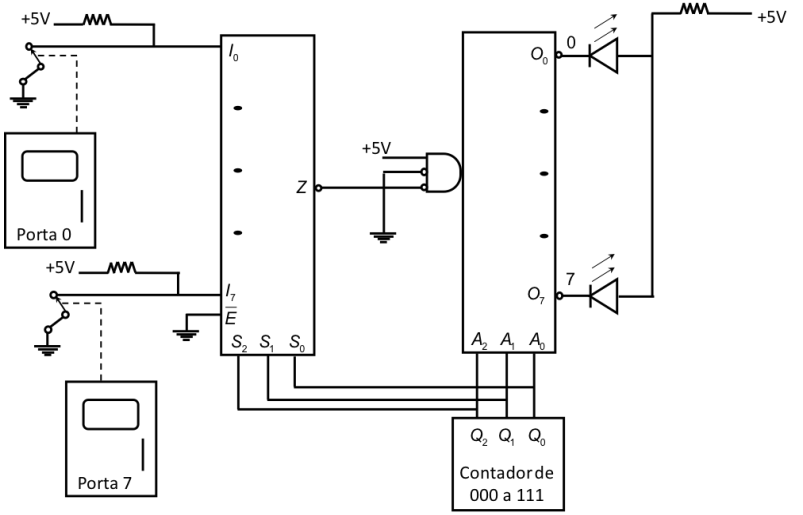
Sem medo de errar

Você é o responsável técnico de uma empresa de sistemas embarcados e seu chefe solicitou que você desenvolvesse um sistema simples e financeiramente rentável de monitoração de segurança de um banco em que o estado aberto ou fechado de oito portas deve ser monitorado. Cada porta controla o estado de uma chave, e é necessário mostrar o estado de cada chave por meio de LEDs montados em um painel remoto de monitoração na sala de segurança. Ao checar o almoxarifado, você percebe que possui apenas CIs 74HC151 e 74HC138, que são referentes aos circuitos multiplexadores e demultiplexadores. E agora, como resolver mais essa situação? Será possível utilizar a lógica combinacional para implementar esse sistema?

Existem algumas formas de implementar o sistema. Uma delas seria levar o sinal da chave de cada porta até o LED no painel de monitoração na sala de segurança. Contudo, isso exigiria a instalação de grande quantidade de fios por uma longa distância, o que tornaria o seu projeto financeiramente inviável, dadas as condições do projeto.

Uma outra solução, mais adequada e financeiramente mais atrativa, seria utilizar uma combinação multiplexador/demultiplexador interligado por uma porta AND, como mostra a Figura 3.29.

Figura 3.29 | Sistema de monitoração de segurança



Fonte: adaptada de Tocci (2011, p. 532).

Nesse, circuito as oito portas do banco são as entradas do circuito Mux. Elas irão gerar nível lógico 1 quando abertas e nível lógico 0 quando fechadas. O contador será responsável por gerar as entradas das chaves seletoras do Mux e Demux. Cada saída do circuto demux estará ligada ao painel de LEDs indicador, que acenderá quando a saída estiver em nível baixo. Cada número do contador será invertido pelo Mux e passará para a saída invertida Z. Essa saída será transmitida para a entrada Demux, que passará a saída correspondente.

Supondo que o contador esteja na contagem 001 e considerando que a porta está fechada, o nível baixo da entrada I_1 passará pelo Mux e será invertido para produzir $\overline{Z}$. Este, em nível alto, passará pelo Demux de modo que o LED 1 apagará, sinalizando que a porta está fechada.

Faça valer a pena

1. Os *circuitos combinacionais* são tipos de circuitos caracterizados por em qualquer instante de tempo. O nível lógico da saída depende apenas da combinação dos níveis lógicos presentes na entrada do circuito.
Sobre os circuitos combinacionais, podemos afirmar que:
I. Podem ser utilizados para transmitir dados de várias fontes em um único canal de transmissão.

II. Quando utilizados como multiplexadores, são necessários n números de entrada de seleção para 3^n número de linhas de entrada.
III. Quando utilizados como demultiplexadores, o circuito recebe informações digitais a partir de uma linha e as distribui para um determinado número de linhas de saída.
Dada as sentenças, é correto afirmar que:
a) Apenas I e II estão corretas.
b) Apenas I e III estão corretas.
c) Apenas II e III estão corretas.
d) Todas estão corretas.
e) Apenas I está correta.

2. Um multiplexador (ou Mux) digital é um circuito lógico que recebe diversos dados digitais de entrada e seleciona um, em determinado instante, para transferi-lo para a saída.
Em um circuito multiplexador de 1 para 4, sendo $I_0 = 0, I_1 = 0, I_2 = 1, I_3 = 1, S_0 = 1$ e $S_1 = 0$, qual é o nível lógico da saída?
a) 10.
b) 01.
c) Não importa.
d) 1.
e) 0.

3. Um multiplexador (ou Mux) digital é um circuito lógico que recebe diversos dados digitais de entrada e seleciona um, em determinado instante, para transferi-lo para a saída. O envio do dado de entrada desejado para a saída é controlado pelas entradas de seleção ou controle.
Se um multiplexador pode comutar 64 entradas de dados para sua saída, quantas entradas diferentes tem esse MUX?
a) 32.
b) 6.
c) 5.
d) 16.
e) 4.

Referências

FLOYD, Thomas L. **Sistemas digitais**: fundamentos e aplicações. 9. ed. Porto Alegre: Bookman, 2007.

GÜNTZEL, José Luís A.; NASCIMENTO, Francisco Assis do. **Circuitos combinacionais**. Universidade Federal de Santa Catarina (UFSC). 2001. Disponível em: <http://www.inf.ufsc.br/~j.guntzel/isd/isd3.pdf>. Acesso em: 15 out. 2017.

SZANJBERG, Mordka. **Eletrônica digital**: teoria, componentes e aplicações. Rio de Janeiro: LTC, 2014.

TEXAS Instruments. **54LS85/DM54LS85/DM74LS85 4-Bit magnitude comparators**. Texas Instruments. 1989. Disponível em: <https://www.eng.tau.ac.il/~shavitt/courses/DigLogSys/74LS85.pdf>. Acesso em: 16 out. 2017.

TOCCI, Ronald J. **Sistemas digitais**: princípios e aplicações. 11. ed. São Paulo: Pearson Prentice Hall, 2011.

Lógica sequencial

Convite ao estudo

Na unidade anterior, estudamos circuitos lógicos combinacionais. Nesta unidade, conheceremos os circuitos lógicos sequenciais. Um circuito sequencial difere de um combinacional pela presença de memória e de uma ou mais malhas de realimentação. Os sinais de controle nos circuitos sequenciais são gerados sequencialmente e se relacionam não somente com as funções de entrada presentes, mas também com os estados anteriores do sistema armazenado pela memória. Os estados são fornecidos justamente pelas vias de realimentação saída-entrada. Logo, a saída de um sistema sequencial, em qualquer instante, depende da entrada presente e das condições dos sinais da entrada no passado.

Nesse contexto, vamos pensar na seguinte situação: você é proprietário de uma empresa de desenvolvimento de projetos de sistemas de automação. No seu dia a dia, você costuma criar novos projetos e também realiza a manutenção de produtos que você já vendeu para os clientes. Para que você continue desempenhando essa tarefa com qualidade e domínio, fique atento aos conceitos que serão apresentados nesta unidade.

Pronto para mais um desafio? Esperamos que sim!

Bons estudos.

Seção 4.1

Latches e *flip-flops*

Diálogo aberto

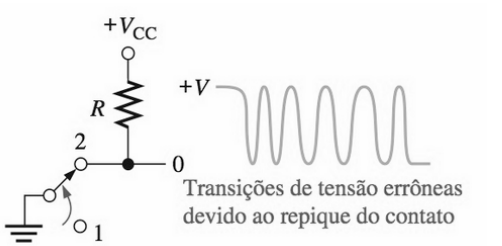
Esta seção começa com o estudo dos fundamentos da lógica sequencial. Nela, serão abordados os dispositivos lógicos biestáveis, que são o *latch* e o *flip-flop*. Os dispositivos biestáveis são dispositivos de armazenamento temporário que têm dois estados estáveis, chamados SET e RESET. A diferença básica entre *latches* e *flip-flops* é a forma com que eles comutam de um estado para o outro. O *flip-flop* é a unidade básica para construção de contadores, registradores e outras lógicas de controle sequencial, e é usado em certos tipos de memórias.

Retomando o contexto apresentado no início da unidade, você é o dono de uma empresa de desenvolvimento de projetos de sistemas de automação. Você costuma trabalhar tanto criando quanto analisando outros projetos, e recebeu para análise um projeto de um controle que não está funcionando adequadamente.

Em uma primeira análise, você isolou a parte do circuito que está com defeito: trata-se de um *latch* $\bar{S}$-$\bar{R}$ utilizado para eliminar oscilações devido ao contato de uma chave mecânica do controle.

Segundo Floyd (2007), quando a chave é acionada fechando um contato, esse contato vibra fisicamente, oscilando por algum tempo antes de estabilizar. Embora isso ocorra em um intervalo relativamente curto, as oscilações produzem picos de tensão em geral não aceitáveis em circuitos digitais. Essa situação é ilustrada na Figura 4.1.

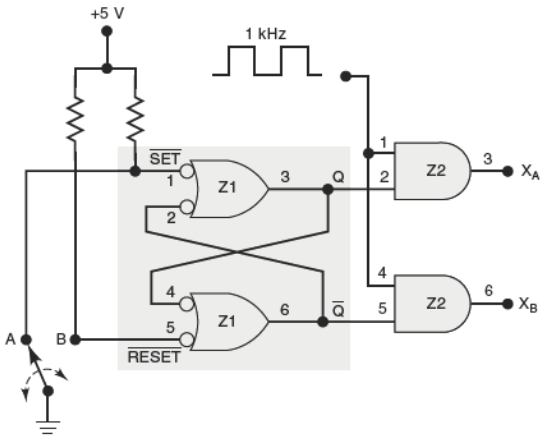
Figura 4.1 | Situação de repique



Fonte: Floyd (2007, p. 391).

Um *latch* feito com portas NAND (ou OR negativa) pode ser usado para eliminar os efeitos do repique de uma chave, como mostra a Figura 4.2.

Figura 4.2 | Circuito para eliminar o repique



Fonte: Tocci, Widmer e Moss (2011, p. 183).

Ao testar o circuito, você percebe que ele funciona corretamente quando a chave está na posição B, porém, quando a chave é colocada na posição A, a saída Q não vai para o estado 1, conforme era esperado. Quais são as possíveis causas para esse mau funcionamento?

Para auxiliá-lo com essa análise, vamos aprender sobre *latches* e *flip-flops*. Está preparado?

Desejamos bons estudos e ótimo trabalho!

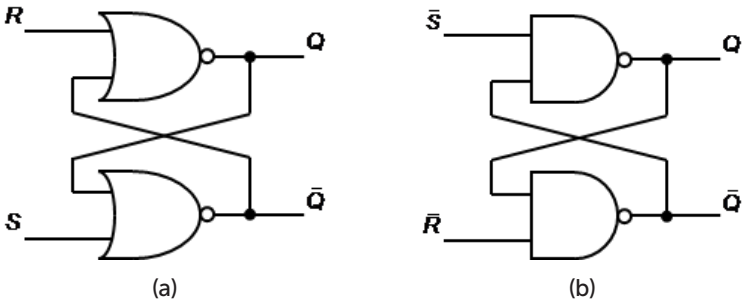
Não pode faltar

Os circuitos lógicos sequenciais são dispositivos biestáveis, isto é, podem ter suas saídas alocadas em dois estados básicos, uma no nível lógico ALTO, ou 1; e outra no nível lógico BAIXO, ou 0, permanecendo trancados (*latched*) indefinidamente até que outro sinal ou pulso aplicado na entrada apropriada cause a mudança desses estados: nível 1 para 0, e nível 0 para 1.

Segundo Floyd (2007), um *latch* é um tipo de dispositivo lógico de armazenamento temporário que tem dois estados estáveis, e por isso é chamado de **biestável** ou **multivibrador**. Um ***latch* S-R** (SET-RESET) com entrada ativa em nível ALTO é composto de duas portas NOR com acoplamento cruzado, conforme pode ser visto na Figura

4.3(a). Um ***latch*** $\bar{S}$-$\bar{R}$ com entrada ativa em nível BAIXO, por sua vez, é composto por duas portas NAND com acoplamento cruzado, conforme pode ser visto na Figura 4.3(b).

Figura 4.3 | *Latch* S-R (a) com entrada ativa em nível ALTO (b) com entrada ativa em nível BAIXO



Fonte: elaborada pelo autor.

Para entendermos o funcionamento de um *latch*, vamos considerar o *latch* S-R composto por portas NOR da Figura 4.3(a). De acordo com o funcionamento de uma porta NOR de duas entradas, podemos dizer que o nível lógico 0 em uma das entradas faz com que a saída seja a outra entrada complementada. Já o nível lógico 1 em uma das entradas força a saída a permanecer no estado lógico 0.

Lembre-se

Vamos recordar a tabela verdade da porta NOR na Tabela 4.1.

Tabela 4.1 | Tabela verdade da porta NOR

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Fonte: elaborada pelo autor.

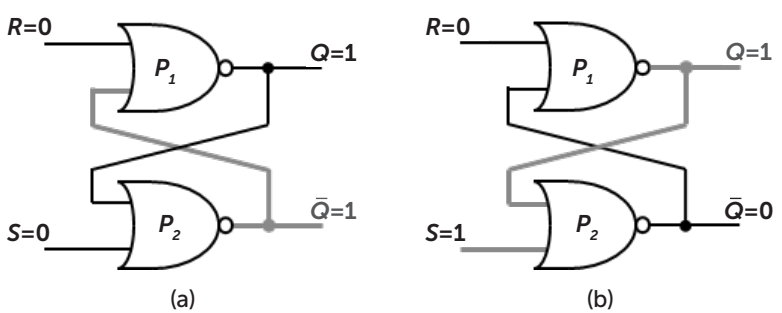
O *latch* mostrado na Figura 4.3(a) tem duas entradas, *S* e *R*; e duas saídas, *Q* e $\bar{Q}$. Vamos iniciar considerando que as duas entradas e a saída *Q* estão no nível lógico 0. Como a saída *Q* é conectada de volta na entrada da porta P_2 e a entrada *S* está no nível 0, a saída $\bar{Q}$ vai para o nível 1. Essa saída é acoplada de volta na entrada da porta P_1 garantindo que sua saída seja 0, como mostrado na Figura 4.4(a). Quando a saída *Q* é 0, o *latch* está no estado de *RESET*, e permanece nesse estado indefinidamente até que um nível 1 seja aplicado na entrada *S*.

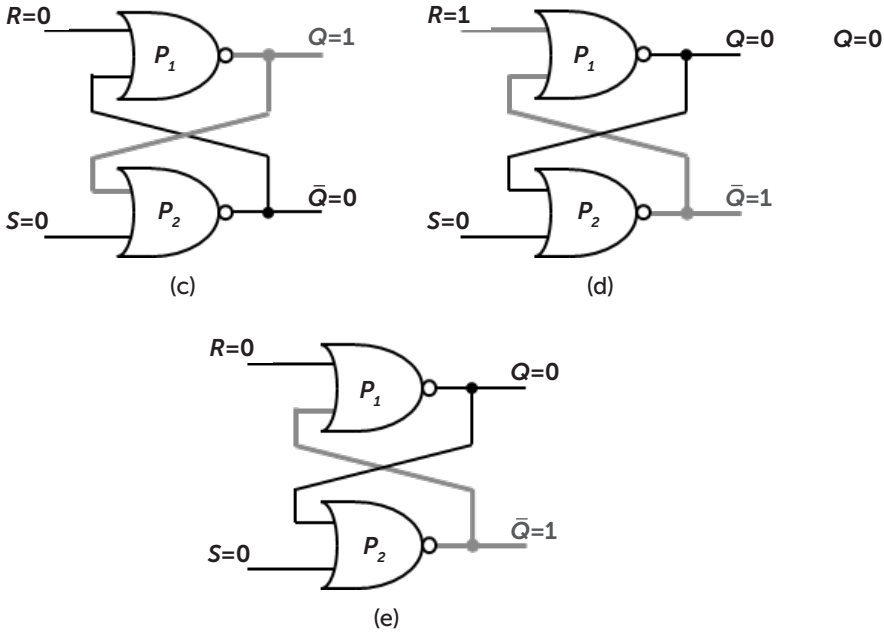
Com a entrada *S* igual a 1, a saída da porta P_2 é forçada para 0. O acoplamento entre as portas faz com que uma entrada da porta P_1 seja 0. Como a entrada *R* está no nível 0, então, a saída *Q* é levada para o nível 1 e o acoplamento entre as portas garante que a saída $\bar{Q}$ seja 0, como pode ser visto na Figura 4.4(b).

O nível 1 na saída *Q*, de fato, garante que a saída $\bar{Q}$ seja 0 mesmo quando o nível 1 da entrada *S* seja removido, como pode ser visto na Figura 4.4(c). Quando a saída *Q* é 1, o *latch* está no estado de *SET* até que um nível 1 seja aplicado na entrada *R*.

Fazendo a entrada *R* igual a 1, a saída *Q* é levada novamente para 0, o acoplamento entre as portas e o fato de que a entrada *S* está no nível 0 fazem com que a saída $\bar{Q}$ seja levada para 1, como vemos na Figura 4.4(d). Como podemos notar, o *latch* foi levado novamente para o estado *RESET*, pois mesmo quando a entrada *R* volta para o nível 0, a saída *Q* mantém-se no nível 0, como pode ser visto na Figura 4.4(e), que é exatamente igual à Figura 4.4(a).

Figura 4.4 | Modos de operação do *latch* S-R básico

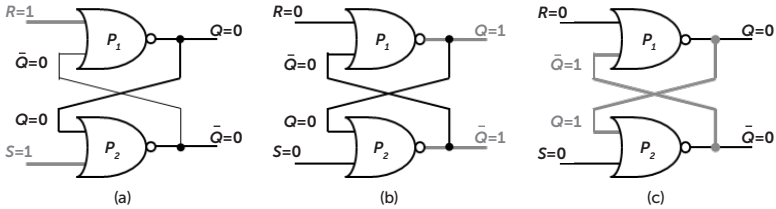




Fonte: elaborada pelo autor.

Quando as duas entradas *S* e *R* são colocadas no nível lógico 1 ao mesmo tempo, tanto a saída Q quanto $\bar{Q}$ são forçadas para 0 (Figura 4.5(a)), violando a condição básica de complementaridade entre essas duas saídas. Mas o problema mais sério ocorre quando tornamos as entradas *S* e *R* 0 novamente. Nesse caso, é possível que as duas entradas retornem ao valor 0 simultaneamente. Quando isso ocorre, as portas P_1 e P_2 terão em todas as suas entradas o nível lógico 0, fazendo com que suas saídas mudem para 1, como mostrado na Figura 4.5(b). No entanto, as portas são realimentadas com as saídas, que estão agora no nível lógico 1, forçando as saídas para 0, como vemos na Figura 4.5(c). Os níveis 1 nas portas levarão as saídas para 1 e esse ciclo se repetirá, criando um estado oscilatório no *latch*. No entanto a oscilação não é uma característica desejável em um bloco de armazenamento de memória.

Figura 4.5 | Oscilação devido ao estado inválido no *latch* S-R básico



Fonte: elaborada pelo autor.

Assimile

De modo similar à tabela verdade do sistema combinacional, a **tabela de estado** enumera as entradas e saídas com colunas adicionais para os estados presentes e seguintes do circuito sequencial, como podemos ver na Tabela 4.2 para o *latch* S-R.

Tabela 4.2 | Tabela de estado do *latch* S-R

S	R	$Q(t)$	$Q(t+1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	Indefinido
1	1	1	Indefinido

Fonte: elaborada pelo autor.

A tabela excitação, por sua vez, lista os valores das entradas do circuito sequencial, bem como as mudanças entre os estados presentes e seguintes, como é visto na Tabela 4.3 para o *latch* S-R.

Tabela 4.3 | Tabela excitação do *latch* S-R

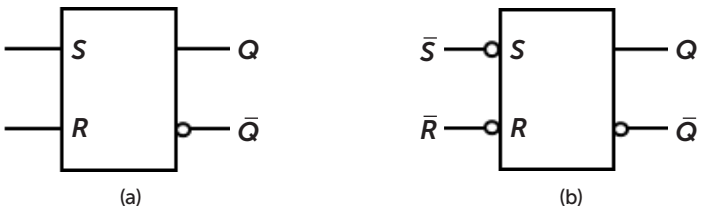
S	R	$Q(t+1)$	Estado
0	0	$Q(t)$	Repouso
0	1	0	***RESET***
1	0	1	***SET***
1	1	?	Inválido

Fonte: elaborada pelo autor.

A tabela excitação é geralmente usada no projeto e síntese de circuitos sequenciais.

Os símbolos lógicos para os *latches* com entradas ativas em nível ALTO e entradas ativas em nível BAIXO são mostrados na Figura 4.6.

Figura 4.6 | Símbolo lógico para o latch: (a) S-R com entradas ativas em nível ALTO; (b) $\bar{S}$-$\bar{R}$ com entradas ativas em nível BAIXO



Fonte: elaborada pelo autor.

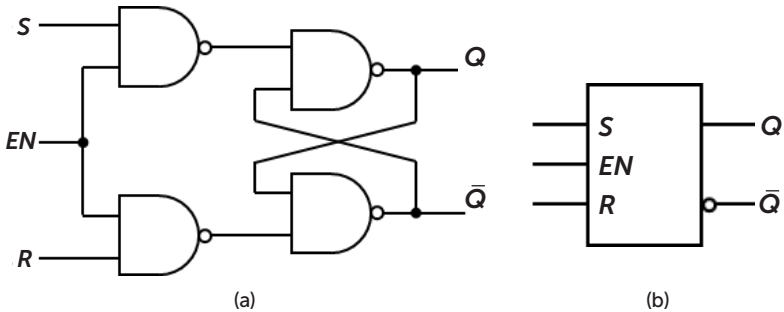
Pesquise mais

O *latch* $\bar{S}$-$\bar{R}$ com portas NAND tem funcionamento idêntico ao do *latch* S-R, porém, sua entrada é ativada em nível BAIXO. Para uma análise de funcionamento, semelhante à feita aqui para o *latch* $\bar{S}$-$\bar{R}$, consulte a seção 7.1 do livro *Sistemas digitais: Fundamentos e aplicações* (FLOYD, 2007), disponível na nossa biblioteca virtual: <https://biblioteca-virtual.com/detalhes/parceiros/5>. Acesso em: 30 out. 2017.

Em um ***latch* controlado**, a transição entre os estados ocorre somente sob a ação da subida de um pulso em uma entrada de habilitação, *EN*, (essa entrada também pode ser chamada de *gatilho* e, portanto, a letra *G* também pode ser usada para indicar essa entrada).

O diagrama lógico e o símbolo lógico para um ***latch*** **S-R controlado** podem ser vistos na Figura 4.7.

Figura 4.7 | *Latch* S-R controlado (a) diagrama lógico (b) símbolo lógico

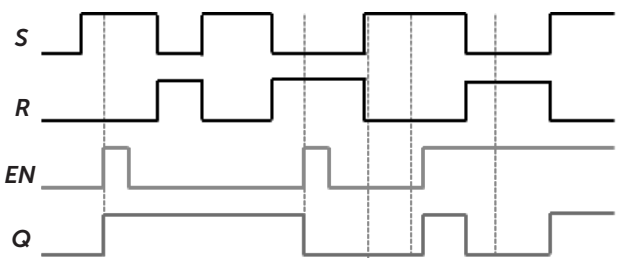


Fonte: elaborada pelo autor.

Exemplificando

As entradas *S* e *R* controlam o estado para o qual o *latch* irá quando um nível ALTO é aplicado na entrada *EN*. O *latch* não mudará de estado enquanto *EN* estiver em nível BAIXO. Porém, durante todo o tempo em que essa entrada permanecer em nível ALTO, a saída é determinada pelos estados das entradas *S* e *R*. Nesse circuito, o estado inválido ocorre quando S e R tiverem simultaneamente nível ALTO. Um exemplo de diagrama de tempo do *latch* S-R controlado pode ser visto na Figura 4.8.

Figura 4.8 | Funcionamento do *latch* S-R controlado

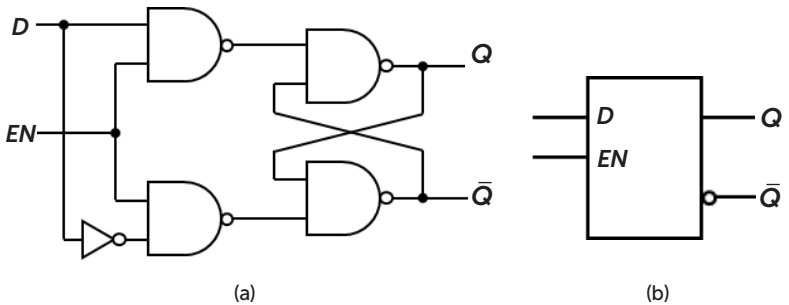


Fonte: elaborada pelo autor.

Os gráficos com "nível lógico versus tempo" fornecem não só uma representação visual do desenvolvimento de sinais no tempo, mas também uma demonstração gráfica de comparação entre sinais em vários pontos de um circuito de chaveamento. Os diagramas de tempo são muito usados em sistemas sequenciais.

O ***latch* D** é um outro tipo de *latch* controlado. Ele teve sua origem devido à necessidade de se evitar, no *latch* S-R, a ocorrência do estado proibido. Por isso, o *latch* D tem apenas uma entrada além da *EN*. A entrada mencionada é denominada de entrada *D* (dado). O diagrama lógico e o símbolo lógico para um *latch* D são mostrados na Figura 4.9.

Figura 4.9 | *Latch* D controlado (a) diagrama lógico (b) símbolo lógico



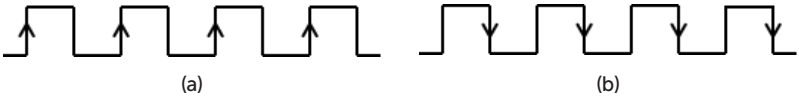
Fonte: elaborada pelo autor.

Quando a entrada *D* for de nível ALTO e a entrada *EN* for de nível ALTO, o *latch* será levado para o estado *SET*. Quando a entrada *D* for de nível BAIXO e a entrada *EN* for de nível ALTO, o *latch* será levado para o estado *RESET*. Dito de outra forma, a saída *Q* segue a entrada *D* quando *EN* for de nível ALTO.

Em lógica sequencial, temos dois modos básicos de operação: **assíncrono** e **síncrono**. No modo assíncrono, os circuitos que constituem o sistema funcionam com tempos independentes entre si. Os estados de saída são gerados imediatamente após a aplicação dos sinais de entrada, controlados pela realimentação direta, usando estritamente os retardos de propagação dos decodificadores do estado seguinte. Os *latches* são um exemplo de dispositivo assíncrono.

No modo síncrono, os ***flip-flops*** são chaveados por um trem de pulsos denominado relógio do sistema, ou, como é internacionalmente conhecido, "*clock* do sistema" (*system clock* ou simplesmente *clock*). O *clock* representa o sinal de comando de um sistema sequencial e consiste em um sinal digital permanente, geralmente de alta frequência. A Figura 4.10 traz um exemplo de um sinal de *clock*. A seta evidencia qual borda é utilizada para acionar o *flip-flop*.

Figura 4.10 | Trem de pulsos de um sinal de *clock*: (a) borda positiva; (b) borda negativa



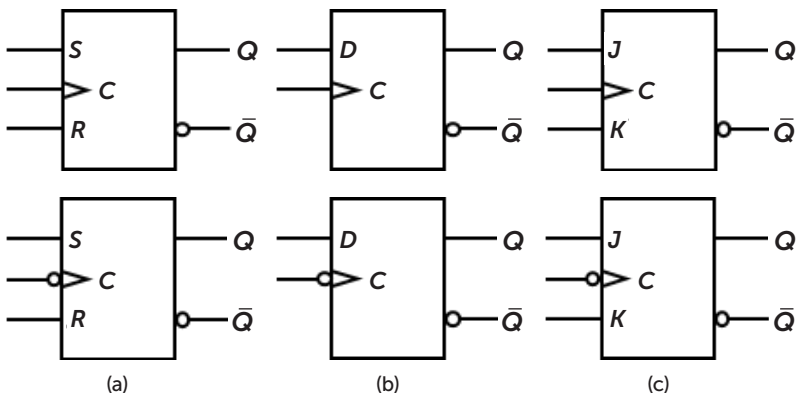
Fonte: elaborada pelo autor.

Quando um um *latch* controlado é adaptado para funcionar de maneira síncrona, ele passa a ser chamado de *flip-flop* S-R disparado por borda.

Apresentaremos, a seguir, três tipos de *flip-flops* disparados por borda: S-R, D e J-K. Na prática, o *flip-flop* S-R não está disponível na forma de CI, mas ele é a base para os *flip-flops* D e J-K, por isso é importante entendermos o seu funcionamento.

A Figura 4.11 traz os símbolos lógicos para esses *flip-flops*. O pequeno triângulo dentro do bloco indica a entrada de *clock* (C), e é denominado indicador de entrada dinâmica. Os *flip-flops* podem ser disparados tanto na borda positiva (parte superior da figura) quanto na negativa (parte inferior da figura), indicados pelo pequeno círculo na entrada *C* (FLOYD, 2007).

Figura 4.11 | Símbolos lógicos de *flip-flops* disparados por borda (a) S-R (b) D (c) J-K



Fonte: adaptada de Floyd (2007, p. 394).

No ***flip-flop* S-R disparado por borda**, (Figura 4.11(a)), as entradas *S* e *R* são denominadas síncronas, uma vez que os bits nas entradas são transferidos para a saída do *flip-flop* apenas na borda de disparo do pulso de *clock*.

Quando $SR = 10$, somente na borda de disparo do pulso de *clock* é que o *flip-flop* será levado para o estado *SET*, com a saída $Q = 1$.

O mesmo ocorre quando $SR = 01$: a saída somente será $Q = 0$ na próxima borda de disparo do pulso de *clock*, estando o *flip-flop* no estado *RESET*. Quando $SR = 00$, a saída não muda de estado, permanecendo no estado anterior. O mesmo fenômeno oscilatório descrito para o *latch* S-R ocorre para o *flip-flop* quando $SR = 11$, por isso essa condição é considerada inválida.

Reflita

Agora que você conhece o funcionamento do *latch* S-R controlado e do *flip-flop* S-R, qual é exatamente a diferença entre esses dois dispositivos?

A Tabela 4.4 traz a tabela de excitação do *flip-flop* S-R disparado por borda positiva. Lembre-se, o *flip-flop* somente muda de estado na borda de disparo de um pulso de *clock*, portanto, qualquer alteração nas entradas do *flip-flop* não alteram o seu estado, com exceção do pequeno intervalo em torno da transição do *clock*.

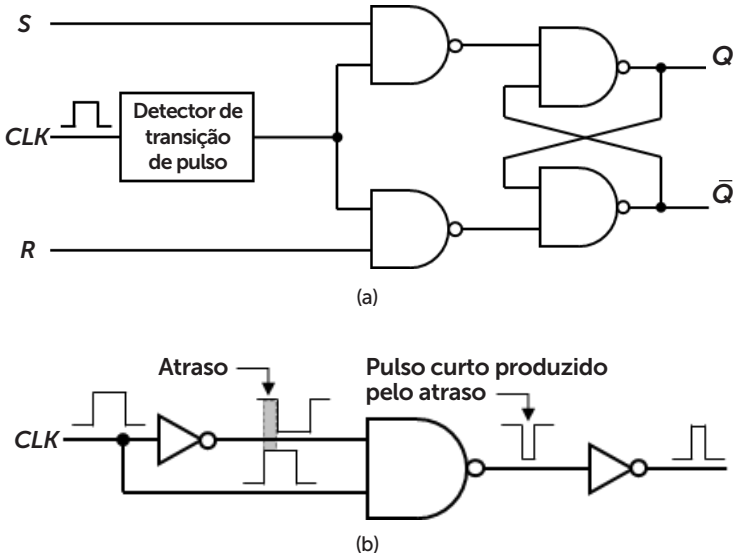
Tabela 4.4 | Tabela excitação do *flip-flop* S-R disparado por borda positiva

S	R	C	$Q(t+1)$	Estado
0	0	X	$Q(t)$	Repouso
0	1	↑	0	***RESET***
1	0	↑	1	***SET***
1	1	↑	?	Inválido

Fonte: elaborada pelo autor.

Na Figura 4.12(a) é possível ver uma implementação simplificada de um *flip-flop* S-R disparado por borda. Usaremos essa implementação para demonstrar o conceito de disparo por borda. O *flip-flop* S-R difere do *latch* S-R controlado apenas pelo fato de que o primeiro possui um detector de transição de pulso (FLOYD, 2007). Um exemplo básico de detector de transição de pulso é mostrado na Figura 4.12(b).

Figura 4.12 | Disparo por borda: (a) diagrama lógico simplificado do *flip-flop* S-R disparado por borda positiva; (b) exemplo de detector de transição de pulso

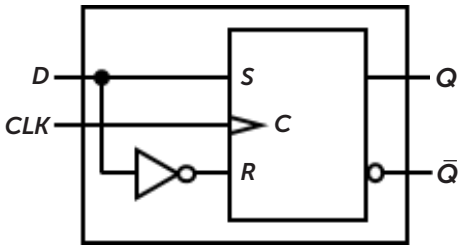


Fonte: adaptada de Floyd (2007, p. 397).

Esse tipo de detector de transição de pulso faz uso do atraso ocasionado pela primeira porta inversora. Assim, o pulso de *clock* invertido chega alguns nanosegundos depois na porta NAND em relação ao pulso original. Esse circuito produz um pico de duração muito curta na transição positiva do pulso de *clock*. Em um *flip-flop* disparado pela borda negativa, o sinal do *clock* é invertido primeiro, produzindo um pico na borda negativa.

Os *flip-flops* D e J-K são comercializados na forma de CI e são mais amplamente usados que o tipo S-R. No entanto, eles são derivados do *flip-flop* S-R. O *flip-flop* D (Figura 4.11(b)) é usado quando um único bit de dado (1 ou 0) deve ser armazenado. Um ***flip-flop* D** básico é obtido através de um *flip-flop* S-R, invertendo a entrada S e conectando esse sinal à entrada *R*, como pode ser visto na Figura 4.13. Observe que esse tipo de *flip-flop* possui apenas uma entrada além do *clock*. Caso a entrada *D* seja 1 quando um pulso de *clock* é aplicado, o *flip-flop* irá para o estado *SET* e o nível 1 da entrada será armazenado pelo *flip-flop*. Caso exista um nível 0 na entrada *D* quando um pulso de *clock* seja aplicado, o *flip-flop* irá para o estado *RESET* e o nível 0 da entrada será armazenado pelo *flip-flop*.

Figura 4.13 | *Flip-flop* D disparado por borda positiva



Fonte: elaborada pelo autor.

A tabela de excitação do *flip-flop* D disparado por borda positiva pode ser vista na Tabela 4.5. A operação de um dispositivo disparado por borda negativa é evidentemente a mesma, exceto que o disparo ocorre na borda de descida do pulso de *clock*. Lembre-se: a saída *Q* segue a entrada *D* na borda ativa ou de disparo do *clock*.

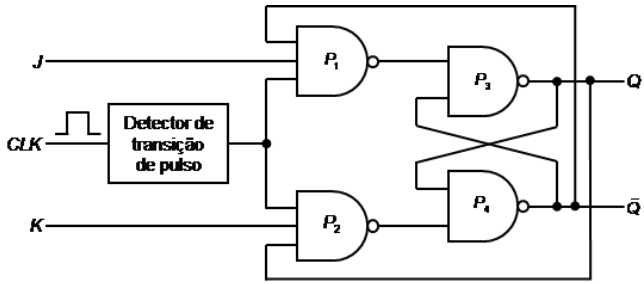
Tabela 4.5 | Tabela excitação do *flip-flop* D disparado por borda positiva

D	C	$Q(t+1)$	Estado
0	↑	0	*RESET*
1	↑	1	*SET*

Fonte: elaborada pelo autor.

O ***flip-flop* J-K** (Figura 4.11(c)) é versátil e amplamente usado. O *flip-flop* J-K funciona da mesma forma que o *flip-flop* S-R nas condições de operação de *SET*, *RESET* e repouso. A diferença entre os dois é que o *flip-flop* J-K não possui um estado inválido, como acontece com o *flip-flop* S-R. A Figura 4.14 mostra a lógica interna básica para um *flip-flop* J-K disparado por borda. Note que a saída *Q* é conectada de volta na entrada da porta P_2 e a saída $\bar{Q}$ é conectada de volta na entrada da porta P_1. As duas entradas de controle são denominadas J e K em homenagem a Jack Kilby, inventor do circuito integrado (FLOYD, 2007).

Figura 4.14 | Diagrama lógico simplificado para um *flip-flop* J-K disparado por borda positiva



Fonte: elaborada pelo autor.

Um *flip-flop* J-K também pode ser do tipo disparado por borda negativa, caso no qual a entrada de *clock* é invertida.

A operação lógica do *flip-flop* J-K é a mesma que a do tipo S-R para as condições de *SET* e *RESET*. A diferença ocorre quando as entrada *J* e *K* estiverem ambas em nível ALTO. Para entender isso, considere que o *flip-flop* esteja no estado de *RESET*. O nível lógico 1 em $\bar{Q}$ habilita a porta P_1, assim, uma transição no *clock* leva o *flip-flop* para o estado *SET*, de modo que agora existe um nível 1 em Q, permitindo que a próxima transição do *clock* passe através da porta P_2 e "*resete*" o *flip-flop*. Podemos notar que a cada transição sucessiva do *clock* o *flip-flop* muda para o estado oposto ao anterior. Esse estado é denominado ***toggle*** (comutação). A Tabela 4.6 traz a tabela de excitação do *flip-flop* J-K disparado por borda positiva. Observe que não existe estado inválido, como ocorre com o *flip-flop* S-R. A tabela de excitação, para um dispositivo disparado por borda negativa, é idêntica a essa tabela, exceto que o *flip-flop* é disparado na borda de descida do pulso de *clock*.

Tabela 4.6 | Tabela excitação do *flip-flop* J-K disparado por borda positiva

J	*K*	*C*	$Q(t+1)$	Estado
0	0	↑	$Q(t)$	Repouso
0	1	↑	0	*RESET*
1	0	↑	1	*SET*
1	1	↑	$\bar{Q}(t)$	*Toggle*

Fonte: elaborada pelo autor.

Um *flip-flop* J-K conectado para o modo *toggle* é denominado algumas vezes de ***flip-flop*** **T**, e tem a tabela de excitação mostrada na Tabela 4.7.

Tabela 4.7 | Tabela excitação do *flip-flop* T disparado por borda positiva

T	*C*	$Q(t+1)$	Estado
0	↑	$Q(t)$	Repouso
1	↑	$\bar{Q}(t)$	*Toggle*

Fonte: elaborada pelo autor.

Pesquise mais

> **O desempenho, os requisitos de operação e as limitações dos flip-flops são especificados por diversas características de operação ou parâmetros encontrados nas folhas de dados dos dispositivos. Geralmente, essas especificações são aplicáveis a todos os flip-flops CMOS e TTL. (FLOYD, 2007, p. 406)**

Para saber mais sobre isso, leia a Seção 7.3 do livro *Sistemas digitais: fundamentos e aplicações* (FLOYD, 2007).

Para mais detalhes sobre o funcionamento dos *latches* e *flip-flops*, leia a seção 7.1 do livro *Eletrônica digital – teoria, componentes e aplicações* (SZAJNBERG, 2014).

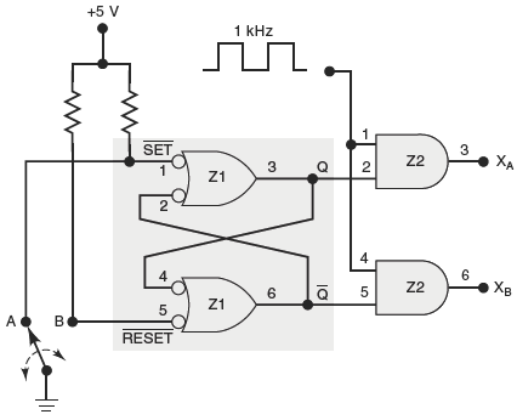
Ambos os materiais estão disponíveis na nossa biblioteca virtual: <https://biblioteca-virtual.com/detalhes/parceiros/5>. Acesso em: 30 out. 2017.

Sem medo de errar

Você é o dono de uma empresa de desenvolvimento de projetos de sistemas de automação e recebeu para análise um projeto de um controle que não está funcionando adequadamente.

Você já isolou a parte defeituosa, trata-se de um latch $\bar{S}$-$\bar{R}$ para eliminar o repique em uma chave eletrônica. O circuito em questão é repetido na Figura 4.15.

Figura 4.15 | Circuito para eliminar o repique



Fonte: Tocci, Widmer e Moss (2011, p. 183).

As saídas desse *latch* controlam a passagem de um sinal formado por pulsos retangulares com frequência de 1 kHz por meio das saídas X_A e X_B das portas AND. Quando a chave é colocada na posição A, o *latch* é "setado" ($Q = 1$). Isso habilita os pulsos de 1 kHz a chegarem à saída X_A, enquanto o nível BAIXO em *Q* mantém $X_B = 0$. Quando a chave é colocada na posição *B*, o *latch* é "resetado" ($Q = 0$), mantendo $X_A = 0$, enquanto o nível ALTO em *Q* habilita a passagem dos pulsos para X_B.

Você percebeu que quando a chave está na posição B o circuito funciona corretamente, porém, com a chave na posição A, a saída Q não vai para o estado 1, conforme era esperado. Existem algumas possibilidades de problema.

1. Circuito aberto internamente em Z1-1, que impediria que a saída *Q* respondesse à entrada $\overline{SET}$.
2. Falha em um componente interno da porta NAND Z1, que impediria que ela funcionasse corretamente.
3. A saída *Q* está fixa em nível BAIXO. Isso pode ser causado por:
   a) Z1-3 em curto interno com GND.
   b) Z1-4 em curto interno com GND.
   c) Z2-2 em curto interno com GND.
   d) A saída *Q* em curto externo com GND.

Uma verificação com ohmímetro, medindo-se entre a saída *Q* e GND, determinará a existência de qualquer uma dessas condições

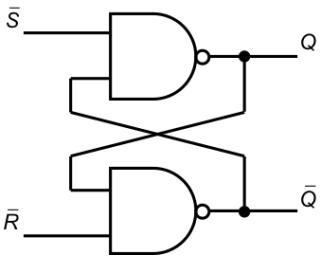
apresentadas; e uma verificação visual pode revelar algum curto externo.

Um curto interno ou externo entre $\bar{Q}$ e V_{cc} é descartado, pois se $\bar{Q}$ estivesse em curto com V_{cc}, isso não evitaria que a saída *Q* fosse levada para nível ALTO, quando a entrada *SET* fosse para nível BAIXO. Como a saída *Q* não está indo para nível ALTO, esse não é o defeito. O motivo que faz com que a saída $\bar{Q}$ esteja fixa em nível ALTO é a saída *Q* estar fixa em nível BAIXO, mantendo a saída $\bar{Q}$ em nível ALTO por meio da porta NAND, na parte inferior do diagrama.

Faça valer a pena

1. Um ***latch*** é um tipo de dispositivo lógico de armazenamento temporário que tem dois estados estáveis, e por isso é chamado de **biestável** ou **multivibrador**. Um ***latch*** $\bar{S}$-$\bar{R}$ com entrada ativa em nível BAIXO é composto por duas portas NAND com acoplamento cruzado, conforme indicado na Figura 4.16.

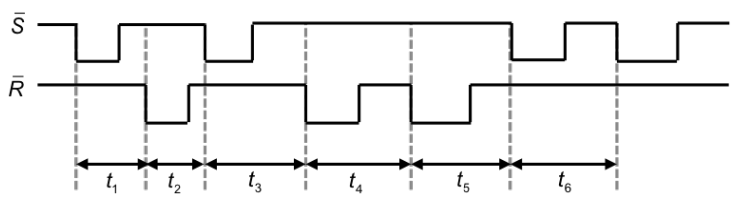
Figura 4.16 | Aspecto construtivo do *Latch* $\bar{S}$-$\bar{R}$



Fonte: elaborada pelo autor.

Considere que as formas de onda de $\bar{S}$ e $\bar{R}$, mostradas na Figura 4.17, são aplicadas nas entradas do *latch* visto na Figura 4.16. Determine os valores da saída *Q*, nos instantes t_1, t_2, t_3, t_4, t_5 e t_6, e assinale a alternativa que contém a sequência correta.

Figura 4.17 | Formas de onda de entrada

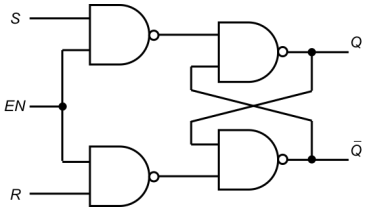


Fonte: elaborada pelo autor.

a) 010110.
b) 110011.
c) 101010.
d) 101001.
e) 010101.

2. Em um ***latch* controlado**, a transição entre os estados ocorre somente sob a ação da subida de um pulso em uma entrada de habilitação, EN. O diagrama lógico para um ***latch* S-R** controlado é mostrado na Figura 4.18.

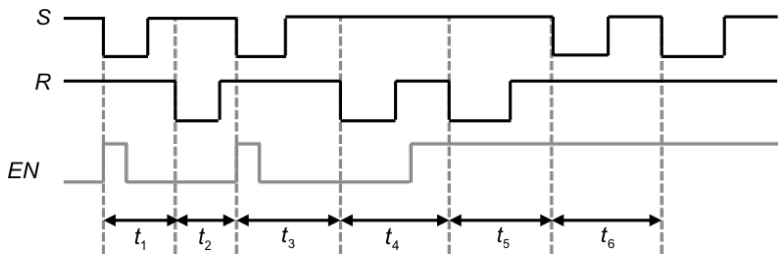
Figura 4.18 | Diagrama lógico para o *latch* S-R controlado



Fonte: elaborada pelo autor.

Considere que as formas de onda de ***S, R*** e ***EN***, mostradas na Figura 4.19, são aplicadas nas entradas do *latch* visto na Figura 4.18. Determine os valores da saída *Q* nos instantes t_1, t_2, t_3, t_4, t_5 **e** t_6 e assinale a alternativa que contém a sequência correta.

Figura 4.19 | Formas de onda de entrada



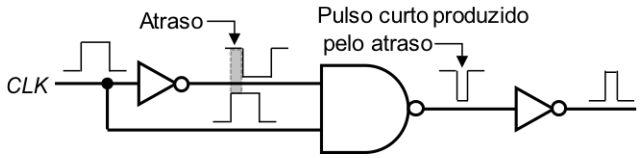
Fonte: elaborada pelo autor.

a) 010110.
b) 000010.
c) 111101.
d) 101001.
e) 101010.

3. No ***flip-flop* S-R disparado por borda**, as entradas são denominadas síncronas, uma vez que os dados nas entradas são transferidos para a saída do *flip-flop* apenas na borda de disparo do pulso de *clock*.

Um exemplo básico de detector de transição de pulso é mostrado na Figura 4.20.

Figura 4.20 | Detector de transição de pulso



Fonte: adaptada de Floyd (2007, p. 397).

Considerando esse contexto, avalie as seguintes asserções e a relação proposta entre elas.

I. Esse tipo de detector de pulso faz uso do atraso ocasionado pela primeira porta inversora para produzir um pico de duração muito curta na transição positiva do pulso de *clock*.

PORQUE

II. O pulso de *clock* invertido chega alguns nanosegundos depois na porta NAND em relação ao pulso original.

A respeito dessas asserções, assinale a alternativa correta:

a) As asserções I e II são proposições verdadeiras, e a II é uma justificativa da I.

b) As asserções I e II são proposições verdadeiras, e a II não é uma justificativa da I.

c) A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.

d) A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.

e) As asserções I e II são proposições falsas.

Seção 4.2

Contadores e registradores

Diálogo aberto

Os *flip-flops* podem ser utilizados para realizar operações de contagem. Esse arranjo com *flip-flops* recebe o nome de contador. A quantidade de *flip-flops* usada e a forma como eles são combinados determinam o número de estados e a sequência específica de estados que o contador percorre durante um ciclo completo. Os registradores de deslocamento são um tipo de circuito lógico muito parecido com os contadores digitais. Os registradores são geralmente usados para armazenamento de dados digitais e não possuem uma característica interna de sequência de estado como os contadores.

Relembrando: você é o dono de uma empresa de desenvolvimento de projetos de sistemas de automação e, dessa vez, sua empresa foi recentemente contratada por um pequeno shopping local para desenvolver um sistema de controle de estacionamento de veículos. O problema consiste em fazer um projeto de monitoração dos espaços disponíveis numa centena de vagas num estacionamento e prover a sinalização da condição de lotado através de uma indicação luminosa e abaixando a cancela na entrada.

Para auxiliá-lo com esse projeto, vamos aprender mais sobre o uso de *flip-flops* como contadores e registradores de deslocamento.

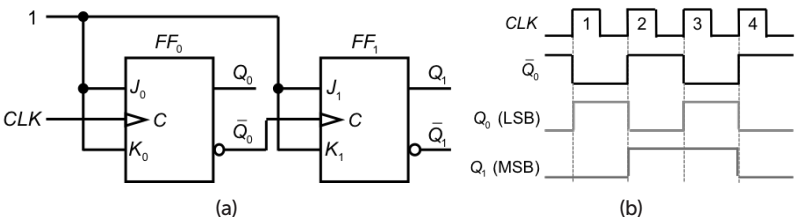
Não pode faltar

Os contadores podem ser classificados como **assíncronos** ou **síncronos**, de acordo com a maneira como eles recebem os pulsos de *clock*. Dentro dessas categorias, os contadores podem ainda ser classificados pela sequência com a qual percorrem os estados, pelo número de estados ou pelo número de *flip-flops* no contador.

Nos **contadores assíncronos**, o primeiro *flip-flop* recebe um sinal de *clock* externo e cada *flip-flop* subsequente recebe o *clock* através da saída do *flip-flop* anterior.

A Figura 4.20(a) mostra um contador binário de dois bits conectado para operação assíncrona. Os *flip-flops* são conectados para operação *toggle* ($J = 1$ e $K = 1$) e o *clock* (*CLK*) é aplicado na entrada de *clock* (*C*) apenas do primeiro *flip-flop* (FF_0), que é o bit menos significativo (LSB). O segundo *flip-flop* (FF_1) é disparado pela saída $\bar{Q}_0$ do FF_0. Enquanto o FF_0 muda de estado na borda positiva de cada pulso de *clock*, o FF_1 muda apenas quando disparado por uma transição positiva da saída $\bar{Q}_0$ do FF_0. Uma vez que há o tempo de atraso de propagação inerente dos *flip-flops*, a transição do pulso de *clock* (*CLK*) de entrada e a transição da saída $\bar{Q}_0$ do FF_0 nunca podem mudar exatamente ao mesmo tempo. Portanto, os dois *flip-flops* nunca são disparados simultaneamente (FLOYD, 2007). Por isso, a operação do contador é assíncrona.

Figura 4.20 | Contador binário assíncrono de dois bits: (a) esquema de montagem; (b) diagrama de temporização



Fonte: adaptada de Floyd (2007, p. 444).

A Figura 4.20(b) ilustra as mudanças de estado nas saídas dos *flip-flops* em resposta aos pulsos de *clock*. Por questões de simplicidade, as transições de Q_0 e Q_1 para os pulsos do *clock* são mostradas como eventos simultâneos, mas sabemos que existe um pequeno atraso entre as transições do clock e Q_0, e entre as transições de $\bar{Q}_0$ e Q_1. A sequência dos estados do contador realiza uma sequência de números binários, conforme listada na Tabela 4.8.

Tabela 4.8 | Sequência de estados binários para o contador de dois bits assíncrono

CLK	Q_1	Q_0
Valor inicial	0	0
1	0	1
2	1	0
3	1	1
4 (recicla)	0	0

Fonte: elaborada pelo autor.

O contador da Figura 4.20(a) conta, de fato, os pulsos de *clock* até três, e no quarto pulso ele recicla seu estado para o estado inicial ($Q_0 = 0$, $Q_1 = 0$). O termo reciclagem, usualmente empregado na operação de contadores, se refere à transição do contador do seu estado final para o original.

Faça você mesmo

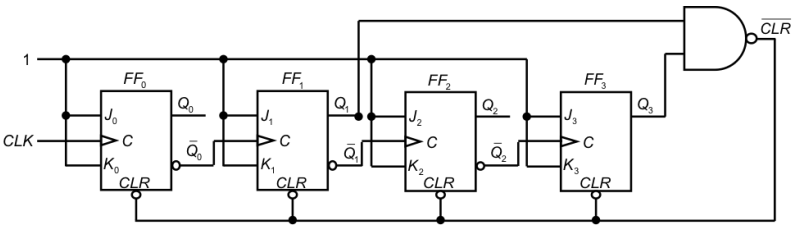
Como era de se esperar, com dois *flip-flops* ($2^2 = 4$), o contador de dois bits exibe quatro estados diferentes. É possível criar contadores capazes de contar qualquer quantia de potências de 2. Faça o esquema de montagem do diagrama de temporização e a tabela com a sequência de estados binários para um contador de três bits. Quantos estados esse contador possui?

Os contadores também podem ser projetados de modo a ter um número de estados menor em sua sequência que o valor máximo de 2^n. Esse tipo de sequência é chamado de **sequência truncada**. Para se obter uma sequência truncada, é preciso forçar o contador a reciclar seu estado antes que ele complete um ciclo passando por todos os estados possíveis.

Um contador de dez estados, chamado contador de década, tem sua sequência de contagem de zero (0000) a nove (1001). Além disso, ele é um contador de década BCD porque a sua sequência de dez estados produz o código BCD. Esse tipo de contador é útil em aplicações com display nas quais o código BCD é necessário para conversões com leituras em decimal.

Por exemplo, em um contador de década BCD, o estado é reciclado para o estado 0000 após o estado 1001. Um contador de década requer quatro *flip-flops* (três *flip-flops* são insuficientes, pois $2^3 = 8$). Uma forma de fazer um contador reciclar após a contagem do nove (1001) é decodificar a contagem dez (1010) com uma porta NAND e conectar a saída da porta NAND nas entradas de *clear* ($\overline{CLR}$) dos *flip-flops*, como mostra a Figura 4.21.

Figura 4.21 | Contador de década assíncrono

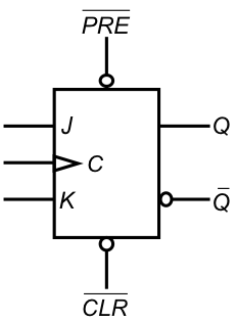


Fonte: adaptada de Floyd (2007, p. 449).

Assimile

A maioria dos *flip-flops* em circuitos integrados também tem entradas assíncronas. Essas são entradas que afetam o estado do *flip-flop* independentemente do *clock*. Elas são normalmente denominadas ***preset*** (*PRE*) e ***clear*** (*CLR*). Um nível ativo na entrada *preset* irá levar o *flip-flop* para o estado *SET*, e um nível ativo na entrada *clear* irá levar o *flip-flop* para o estado *RESET*. O símbolo lógico para um *flip-flop* J-K com entradas *preset* e *clear* é mostrado na Figura 4.22. Essas entradas são ativas em nível BAIXO, conforme indicado pelos pequenos círculos. Essas entradas de *preset* e *clear* devem ser mantidas em nível ALTO para a operação síncrona.

Figura 4.22 | Símbolo lógico para um *flip-flop* J-K com entradas *preset* e *clear*



Fonte: elaborada pelo autor.

Observe que apenas as saídas Q_1 e Q_3 são necessárias para reciclar a contagem. Esse tipo de arranjo é chamado de **decodificação parcial**, pois não foram usadas todas as saídas para reciclar a contagem. Quando o contador chega no estado dez (1010), a saída da porta de decodificação vai para 0 e assincronamente "reseta" todos os *flip-flops*.

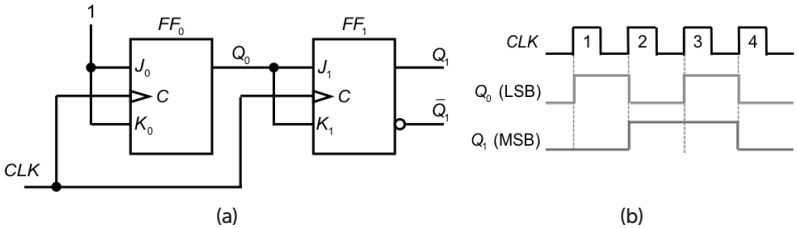
Nos **contadores síncronos**, o mesmo sinal de *clock* é fornecido simultaneamente para todos os *flip-flops*. Na Figura 4.23(a), temos como exemplo um contador binário síncrono de dois bits. Note que foi preciso um arranjo diferente na ligação das entradas J_1 e K_1 para que o FF_1 consiga uma sequência binária.

Considere que os dois *flip-flops* estão "resetados", de modo que o contador esteja inicialmente no estado binário 0. Quando a borda positiva do primeiro pulso de *clock* é aplicada, FF_0 comuta e Q_0 passa para o nível lógico 1. No FF_1 as entradas J_1 e K_1 estão em nível baixo porque Q_0, saída na qual as entradas de FF_1 estão conectadas, ainda não foi para o nível lógico 1. Lembre-se de que existe um atraso de propagação entre a borda de disparo do *clock* até que a saída *Q* realmente comute. Assim, $J_1 = 0$ e $K_1 = 0$ quando é aplicada a borda de subida do primeiro pulso de *clock*. Essa é uma condição de repouso, e por isso FF_1 não muda de estado (FLOYD, 2007).

Quando a borda de subida do segundo pulso de *clock* ocorre, FF_0 comuta e Q_0 vai para o nível baixo. Como FF_1 tem um nível 1 nas entradas J_1 e K_1 na borda de disparo desse pulso de *clock*, o *flip-flop* comuta e Q_1 vai para o nível lógico 1. Quando a borda de subida do terceiro pulso de clock ocorre, FF_0 comuta novamente para o estado *SET* e FF_1 permanece no estado *SET*.

Por fim, na quarta borda de subida do *clock*, Q_0 e Q_1 vão para o nível lógico 0, porque os dois estão na condição *toggle* em suas entradas *J* e *K*. Assim, o contador recicla para o seu estado original. A Figura 4.23(b) mostra o diagrama de temporização desse contador.

Figura 4.23 | Contador binário síncrono de dois bits: (a) esquema de montagem; (b) diagrama de temporização



Fonte: adaptada de Floyd (2007, p. 452-453).

Observe que os atrasos de propagação não são indicados na Figura 4.23(b). Embora os atrasos sejam um fator importante na operação de um contador síncrono, eles normalmente são omitidos no diagrama de temporização geral por questão de simplicidade de representação. Entretanto, em circuitos digitais de alta velocidade, esses pequenos atrasos são considerações importantes no projeto e na análise de defeito (FLOYD, 2007).

Exemplificando

Um contador binário de três bits tem a sua sequência de estados representada na Tabela 4.9. Vamos projetar um contador binário **síncrono** de três bits?

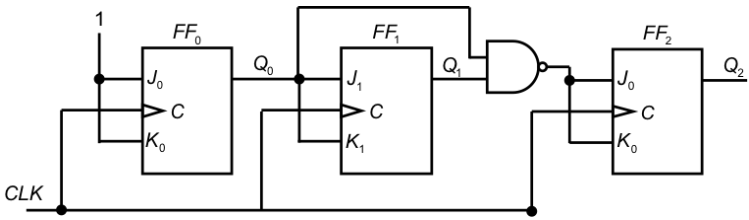
Tabela 4.9 | Sequência de estados binários para o contador de dois bits assíncrono

CLK	Q_2	Q_1	Q_0
Valor inicial	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (recicla)	0	0	0

Fonte: elaborada pelo autor.

Observe que Q_0 alterna os bits a cada pulso de *clock*. Para produzir essa operação, o FF_0 deve ser mantido no modo *toggle*. Note que Q_1 vai para o estado oposto cada vez que Q_0 for nível 1. Segundo Floyd (2007), para produzir essa operação, Q_0 é conectada diretamente nas entradas J_1 e K_1 do FF_2. Por fim, no FF_2 a saída Q_2 muda de estado somente quando Q_0 e Q_1 são nível lógico 1. Essa condição pode ser detectada por uma porta AND e aplicada nas entradas J_2 e K_2 do FF_2. O esquema de montagem desse contador pode ser visto na Figura 4.24.

Figura 4.24 | Contador binário síncrono de três bits



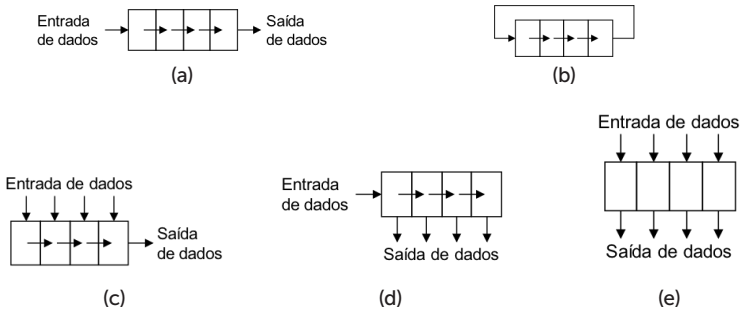
Fonte: adaptada de Floyd (2007, p. 454).

Você é capaz, agora, de projetar um contador binário síncrono de quatro bits? E um contador de década síncrono de quatro bits? Temos a certeza de que sim! Exercite seus conhecimentos projetando esses contadores.

Um registrador é um circuito digital com duas funções básicas: **armazenar** e **movimentar dados**.

Segundo Floyd (2007), a **capacidade de armazenamento** de um registrador é o número total de bits de dados digitais que ele pode reter. A capacidade de armazenamento de um registrador o torna um importante tipo de dispositivo de memória. Ela pode ser aumentada aumentando-se o número de *flip-flops* no registrador, cada *flip-flop* equivale a um bit de capacidade a mais. E é a **capacidade de deslocamento** de um registrador que permite o movimento de dados de um *flip-flop* para outro com a aplicação de pulsos de *clock*. Na Figura 4.25 temos exemplos dos tipos de movimentos de dados em registradores de deslocamento.

Figura 4.25 | Movimentos básicos de dados em registradores de deslocamento: (a) entrada serial/saída serial; (b) rotação; (c) entrada paralela/saída serial; (d) entrada serial/saída paralela; (e) entradas paralela/saída paralela



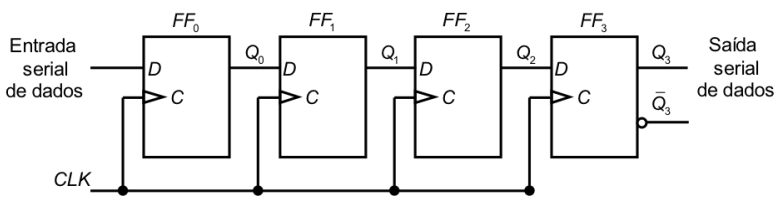
Fonte: elaborada pelo autor.

Reflita

Em geral, qual é a diferença entre um contador e um registrador de deslocamento? Qual *flip-flop* seria mais adequado para ser utilizado em um registrador?

O registrador de deslocamento com **entrada serial/saída serial** recebe um bit de cada vez numa única linha, ou seja, de forma serial. Ele disponibiliza na sua saída a informação armazenada também um bit de cada vez, de forma serial. A Figura 4.26 mostra um dispositivo de quatro bits implementado com *flip-flops* D. Com quatro estágios é possível armazenar até quatro bits de dados nesse registrador de deslocamento.

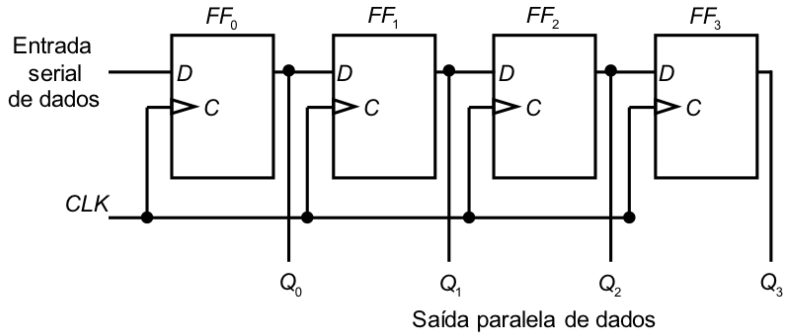
Figura 4.26 | Registrador com entrada serial/saída serial de quatro bits



Fonte: adaptada de Floyd (2007, p. 511).

Em um registrador de deslocamento com **entrada serial/saída paralela**, os bits de dados são inseridos de forma serial, porém, uma vez que os dados são armazenados, cada bit é disponibilizado simultaneamente na saída de forma paralela. A Figura 4.27 mostra um registrador de deslocamento com entrada serial/saída paralela de quatro bits.

Figura 4.27 | Registrador com entrada serial/saída paralela de quatro bits

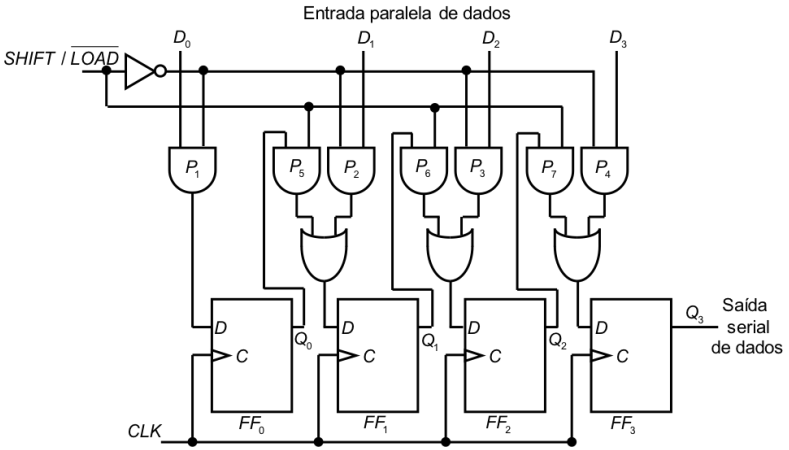


Fonte: adaptada de Floyd (2007, p. 515).

Para um registrador de deslocamento com **entrada paralela/saída serial**, os bits são inseridos todos ao mesmo tempo nos seus respectivos *flip-flops* em vez de bit a bit numa única linha, como acontece com a entrada serial de dados. A saída se dá bit a bit. A Figura 4.28 mostra um registrador de deslocamento com entrada paralela/saída serial de quatro bits. Há, além das quatro linhas de entradas de dados (D_0, D_1, D_2 e D_3), uma entrada $SHIFT / \overline{LOAD}$, que permite carregar os quatros bits em paralelo no registrador.

As portas de P_1 a P_4 são habilitadas quando $SHIFT / \overline{LOAD}$ é 0, permitindo que os bits de dados sejam aplicados na entrada *D* do seu respectivo *flip-flop*. Quando um pulso de *clock* é aplicado, os estados desses *flip-flops* são atualizados adequadamente, de modo que todos os quatro bits são armazenados simultaneamente. Quando $SHIFT / \overline{LOAD}$ for igual a 1, as portas de P_1 a P_4 são desabilitadas e as portas de P_5 a P_7 são habilitadas, deslocando os bits de dados de um *flip-flop* para o outro.

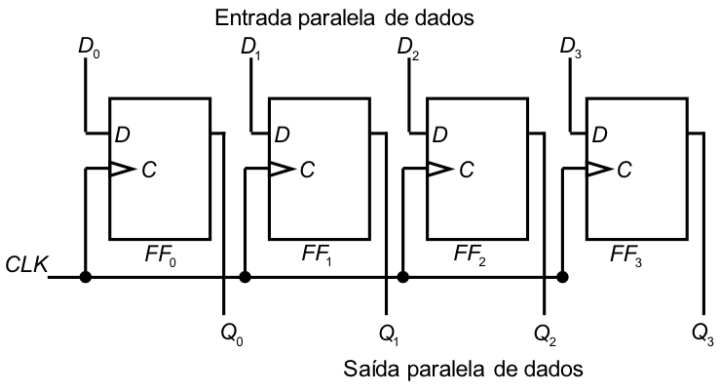
Figura 4.28 | Registrador com entrada paralela/saída serial de quatro bits



Fonte: adaptada de Floyd (2007, p. 518).

O registrador de deslocamento com entrada **paralela/saída paralela** possui o aspecto construtivo, e o seu entendimento é muito mais simples. No instante seguinte à entrada de todos os bits de dados, eles são disponibilizados nas saídas. A Figura 4.29 mostra um registrador com entrada paralela/saída paralela de quatro bits.

Figura 4.29 | Registrador com entrada paralela/saída paralela de quatro bits



Fonte: adaptada de Floyd (2007, p. 522).

Pesquise mais

Os registradores de deslocamento são importantes em aplicações que envolvem o armazenamento e a transferência de dados em sistemas digitais. Nesta seção, apenas introduzimos o assunto. Para saber mais sobre eles, leia o capítulo 9 do *Sistemas digitais: fundamentos e aplicações* (FLOYD, 2007), disponível na nossa biblioteca virtual: <https://biblioteca-virtual.com/detalhes/parceiros/5>. Acesso em: 30 out. 2017.

Sem medo de errar

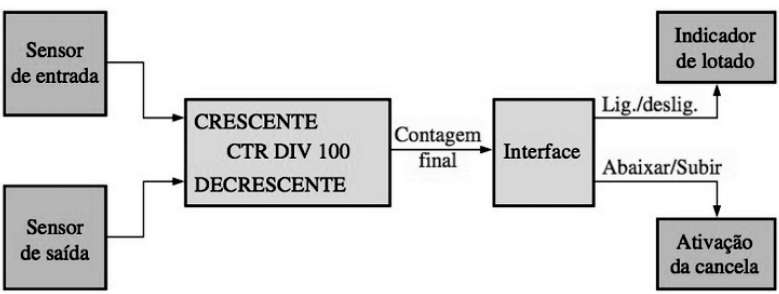
Sua empresa foi recentemente contratada para desenvolver um sistema para monitorar a quantidade de vagas disponíveis em um estacionamento com uma centena de vagas de um pequeno shopping local e prover uma indicação de quando o estacionamento estiver lotado através de um sinal luminoso e abaixando a cancela na entrada.

Você propôs um sistema para resolver esse problema, que consiste em:

1. Dois sensores de presença, um na entrada e outro na saída do estacionamento.
2. Um contador crescente/decrescente.
3. Um circuito lógico para ligar ou desligar o sinal de LOTADO e baixar ou subir a cancela na entrada.

A Figura 4.30 mostra o diagrama de blocos funcional desse sistema.

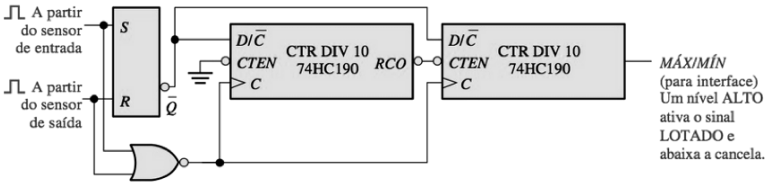
Figura 4.30 | Diagrama de blocos funcional para o controle de vagas do estacionamento



Fonte: Floyd (2007, p. 483).

O diagrama lógico do contador crescente/decrescente é mostrado na Figura 4.31. Ele consiste em dois CIs contadores de década 74HC190 (TEXAS, 2003) conectados em cascata.

Figura 4.31 | Diagrama lógico para um contador crescente/decrescente de módulo 100 para o controle de estacionamento de veículos



Fonte: Floyd (2007, p. 483).

Cada veículo que entra no estacionamento aciona o sensor de entrada que leva o *latch* S-R para o estado *SET*, de modo que $\bar{Q} = 0$ levando o contador a funcionar no modo *crescente*. O sinal do sensor também é levado para uma porta NOR que aciona os contadores pela entrada do *clock* quando a saída da porta muda de 0 para 1. Cada vez que isso ocorre, o contador incrementa uma unidade.

Quando cem veículos entrarem no estacionamento, a saída MÁX/MÍN do contador vai para 1, ativando um circuito que acende o sinal de *lotado* e abaixa a cancela.

Quando um veículo sai, o sensor de saída é acionado, levando o *latch* S-R para o estado *RESET*. Agora $\bar{Q} = 1$, leva o contador para o modo *decrescente*. O sinal desse sensor também é levado para uma porta NOR, que aciona os contadores pela entrada do *clock* quando a saída da porta muda de 0 para 1. Cada vez que isso ocorre, o contador decrementa uma unidade.

Se o estacionamento estiver lotado e um veículo sair, a saída MÁX/MÍN vai para nível BAIXO, desligando o sinal *lotado* e subindo a cancela.

Faça valer a pena

1. Os contadores são classificados em duas grandes categorias de acordo com a forma que eles recebem os pulsos de o *clock*: **assíncronos** e **síncronos**.
Sobre os contadores, avalie as afirmações a seguir:

I. Nos contadores síncronos, o primeiro *flip-flop* recebe o *clock* por meio de um pulso de *clock* externo e cada *flip-flop* sucessivo recebe o *clock* através da saída do *flip-flop* anterior.
II. Nos contadores assíncronos, a entrada de *clock* é conectada a todos os *flip-flops* de forma que eles recebem o *clock* simultaneamente.
III. Os contadores somente podem ser projetados de modo a ter 2^n estados, em que *n* é a quantidade de *flip-flops* do contador.
IV. Um contador de década com uma sequência de contagem de zero (0000) a nove (1001) é um contador de década BCD porque a sua sequência de dez estados produz o código BCD.
É correto o que se afirma em:
a) I e III, apenas.
b) I, II e IV, apenas.
c) I, II e III, apenas.
d) IV, apenas.
e) I, II, III e IV.

2. Os contadores também podem ser projetados para ter um número de estados em sua sequência que é menor que o valor máximo de 2^n. Esse tipo de sequência é denominado de **sequência truncada**. Um módulo comum para contadores com sequências truncadas é dez. Um contador de década BCD tem que reciclar para o estado 0000 após o estado 1001.
Considerando esse contexto, avalie as seguintes asserções e a relação proposta entre elas.
I. Uma forma de fazer um contador reciclar após a contagem do nove (1001) é ligar uma porta NAND nas saídas *Q* do primeiro e do último *flip-flops* e conectar a saída da porta NAND nas entradas de *clear* ($\overline{CLR}$) de todos os *flip-flops*.
PORQUE
II. Apenas no estado nove (1001) ocorre o nível lógico 1 simultaneamente nesses dois *flip-flops*.
a) As asserções I e II são proposições verdadeiras, e a II é uma justificativa da I.
b) As asserções I e II são proposições verdadeiras, e a II não é uma justificativa da I.
c) A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.
d) A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.
e) As asserções I e II são proposições falsas.

3. Um registrador é um circuito digital com duas funções básicas: **armazenamento de dados** e **movimentação de dados**. A capacidade de armazenamento de um registrador o torna um importante tipo de dispositivo de memória.

Sobre os registradores, avalie as afirmações a seguir:

I. A capacidade de armazenamento de um registrador é o número total de bits de dados digitais que ele pode reter.

II. A capacidade de deslocamento de um registrador permite o movimento de dados de um estágio para outro dentro do registrador ou ainda para dentro ou para fora do registrador com a aplicação de pulsos de *clock*.

III. O registrador de deslocamento com entrada serial/saída serial aceita dados seriais, ou seja, um bit de cada vez numa única linha.

IV. Em um registrador de deslocamento com entrada serial/saída paralela, os bits de dados são inseridos serialmente, porém, a saída de cada estágio está disponível.

V. Em um registrador de deslocamento com entrada paralela/saída serial, os bits são inseridos simultaneamente nos seus respectivos estágios em linhas paralelas. A saída se dá bit a bit.

VI. No registrador de deslocamento com entrada paralela/saída paralela, os bits de dados aparecem na saída no próximo pulso de *clock* seguinte à entrada de todos os bits de dados.

É correto o que se afirma em:

a) I, II e III.
b) I, II, IV, e VI.
c) III, IV, e VI.
d) II, III, V, e VI.
e) I, II, III, IV, V, e VI.

Seção 4.3

Máquinas de estado finitos

Diálogo aberto

Os sistemas sequenciais podem ser utilizados em projetos para diversas finalidades. Eles são caracterizados por circuitos que executam sequências de estados predefinidos de acordo com os pulsos de *clock*, e com os outros sinais de entrada. Os contadores já estudados são máquinas de estado, destinados a executar sequências numéricas regulares.

Relembrando, você é o dono de uma empresa de desenvolvimento de projetos de sistemas de automação. Em seu último trabalho, você foi contratado por um pequeno shopping local para desenvolver um sistema de controle de estacionamento de veículos. O problema consistia em fazer um projeto de monitoração dos espaços disponíveis numa centena de vagas em um estacionamento e prover a indicação da condição de "lotado", através de um sinal luminoso e abaixando a cancela que dá acesso à entrada. O cliente, embora satisfeito com sua solução, decidiu mudar o conceito do seu pedido. Nessa nova situação, solicitou que seja adicionado na entrada do estacionamento um painel indicando o número de vagas, além do que já havia sido feito.

Para auxiliá-lo com esse projeto, vamos aprender mais sobre o uso e implementação das máquinas de estado finito. Preparado?

Bons estudos.

Não pode faltar

Os sistemas sequenciais também podem ser projetados para aplicações específicas. Nesses casos, geralmente são chamados de **máquinas de estados finitos**. Os contadores são casos particulares de máquinas de estado. A máquina de estados finitos, de forma geral, é um conceito abstrato que, por meio de um método estruturado, auxilia a análise e a síntese de sistemas sequenciais sincronizados. Representa um sistema digital que, quando acionado por um pulso de um *clock*, move-se de um estado lógico para outro.

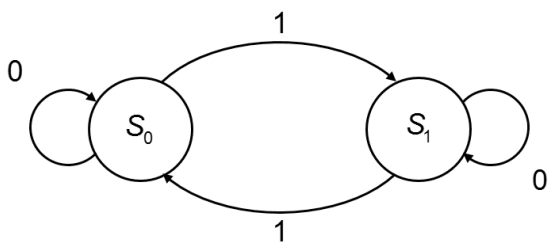
As máquinas de estado finito têm um nome que pode causar alguma confusão. O termo "finito" está presente para diferenciar as máquinas de estado finito de uma representação similar usada na matemática, que pode ter um número infinito de estados. O termo "máquina" é usado no sentido matemático, ou de ciência da computação, sendo um objeto conceitual que pode executar uma linguagem abstrata. Nesse caso, esse sentido de máquina não é o de hardware.

Em tese, todos os circuitos síncronos, incluindo os registradores e contadores movidos por um *clock*, podem ser representados por máquinas de estados finitos. É sempre recomendável solucionar um problema em um nível funcional mais alto, valendo-se da metodologia das máquinas de estados finitos. O projeto de um circuito sequencial pode ser dividido em cinco etapas:

1. Desenhar o diagrama de estados do circuito.

   Um **diagrama de estado** representa um modelo gráfico de um sistema sequencial e especifica as transições entre os estados do sistema, as quais, por sua vez, descrevem como um estado particular passa para outro. Um sistema sequencial pode convenientemente ser representado por um diagrama de estado. Um diagrama é traçado na forma de um conjunto de círculos, em que cada um deles corresponde a um estado. A Figura 4.32 traz como exemplo um diagrama de estado genérico para um sistema com uma única entrada e dois estados possíveis, S_0 e S_1. As setas indicam as transições entre os estados para cada ação (entrada) em um determinado estado.

Figura 4.32 | Diagrama de estado genérico



Fonte: elaborada pelo autor.

2. Identificação dos estados e cálculo da quantidade de *flip-flops* necessária, dada por

$$n_{ff} = \text{int}\left(\log_2 n_s\right) \tag{4.1}$$

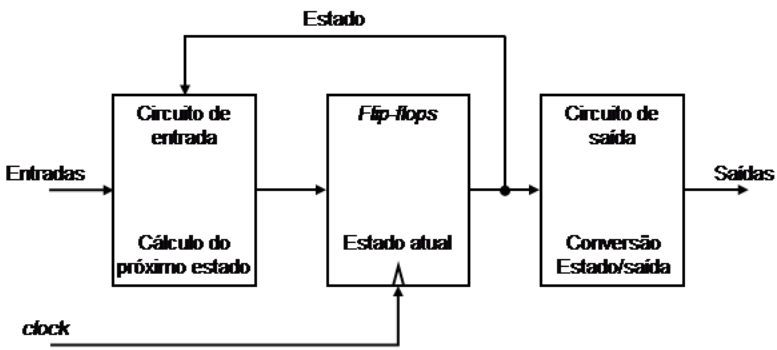
em que n_{ff} é o número de *flip-flops* e n_s é o número de estados. A função $\text{int}(\cdot)$ indica o número inteiro menor ou igual à variável.

3. Obter as expressões lógicas, com o uso de tabelas verdade e/ou mapas de Karnaugh, das variáveis de entrada dos *flip-flops* em função do estado atual e das variáveis de entrada.
4. Obter as expressões lógicas, com o uso de tabelas verdade e/ou mapas de Karnaugh, das saídas dos *flip-flops* em função do estado atual.
5. Desenhar o circuito.

Antes de praticarmos, é importante sabermos que existem dois modelos tradicionais para a implementação de máquinas de estados, são eles: as **máquinas Moore** e as **máquinas Mealy**.

Nas **máquinas Moore**, as saídas dependem **apenas** do estado atual registrado internamente no sistema. As mudanças das saídas ocorrem apenas na próxima transição de borda do *clock*. A Figura 4.33 apresenta um diagrama de blocos de uma máquina Moore.

Figura 4.33 | Diagrama de blocos de uma máquina Moore

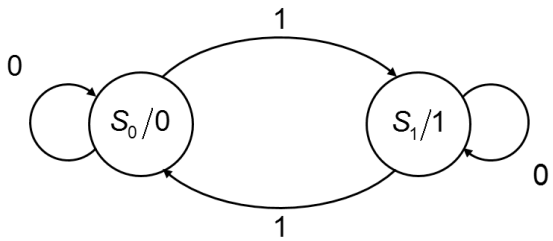


Fonte: elaborada pelo autor.

É possível notar, pela figura, que o *clock* comanda a memória que envia para o circuito de saída o estado atual, que é resultado da combinação entre as entradas e o estado anterior. Assim, as saídas assumem um novo estado somente na próxima transição de borda

do *clock*. A Figura 4.34 traz um diagrama de estado genérico de uma máquina Moore para um sistema com uma entrada, uma saída e dois estados possíveis. Note que a saída é indicada em cada estado.

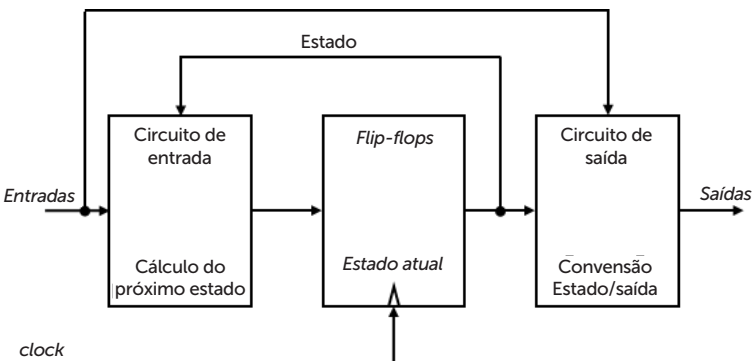
Figura 4.34 | Diagrama de estado genérico para uma máquina Moore



Fonte: elaborada pelo autor.

Nas **máquinas Mealy**, as saídas dependem tanto das entradas quanto do estado atual registrado internamente no sistema. Se houver mudanças nas entradas, as saídas também se alterarão. A Figura 4.35 apresenta em blocos a estrutura de um sistema sequencial que utiliza o modelo de Mealy.

Figura 4.35 | Diagrama de blocos de uma máquina Mealy



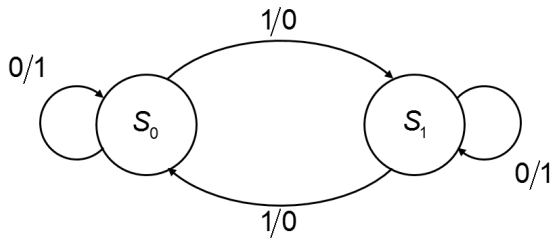
Fonte: elaborada pelo autor.

Assimile

Nas **máquinas Moore**, as saídas dependem **apenas** do estado atual registrado internamente no sistema. Já nas **máquinas Mealy**, as saídas dependem tanto das entradas quanto do estado atual registrado internamente no sistema.

Agora, podemos notar que o *clock* comanda o registrador que envia os estados armazenados resultantes das entradas e estado anteriormente existente para estágio de saída, porém, as saídas também recebem as informações diretamente das entradas. Nesse caso, as saídas podem ser alteradas pela mudança dos estados de entradas, mesmo antes da atuação do *clock*. A Figura 4.36 traz um diagrama de estado genérico de uma máquina Mealy para um sistema com uma entrada, uma saída e dois estados possíveis. Note que agora a saída é indicada em cada transição de estado.

Figura 4.36 | Diagrama de estado genérico para uma máquina Mealy



Fonte: elaborada pelo autor.

As máquinas de estado também podem ser classificadas como sistemas síncronos ou assíncronos, conforme as características de aplicação do sistema de *clock* nos circuitos. Como já vimos, os sistemas sequenciais síncronos são aqueles em que os circuitos operam de maneira sincronizada com a entrada *clock*, ou seja, todos os elementos, *flip-flops* ou registradores de estado são comandados pelo mesmo pulso de *clock* em função do acionamento no tempo, pois estes têm ligação dessa entrada em comum. Já os sistemas sequenciais assíncronos são aqueles que atuam de maneira assíncrona, ou seja, sem sincronismo entre os elementos do circuito, sendo que as variações internas e de saída ocorrem em instantes diferenciados com relação ao pulso de *clock*. Notamos pelas estruturas estudadas que a atuação do modelo de Moore é síncrona, e no modelo de Mealy as saídas podem atuar assincronamente (CAPUANO, 2014).

Reflita

Você é capaz de pensar em aplicações em que é mais conveniente utilizar um dos modelos de máquina de estado finito apresentado (Moore ou Mealy) a despeito do outro?

Para efetuarmos projetos de sistemas sequenciais, de um modo geral, devemos seguir o procedimento descrito, ou seja, determinar o diagrama de estados; levantar a tabela verdade da sequência de estados proposta; colocar na tabela os estados a serem assumidos nas entradas dos *flip-flops*, utilizando a tabela de projetos já desenvolvida; utilizar os mapas de Karnaugh para simplificá-los e assim elaborar o circuito com as expressões simplificadas. Para entender melhor o procedimento, vamos elaborar um projeto no exemplo a seguir.

Exemplificando

Vamos projetar uma máquina de estados para atuar como contador de três bits para efetuar a contagem crescente ($E = 0 \rightarrow$ 0 a 7) ou decrescente ($E = 1 \rightarrow$ 7 a 0). O circuito deverá possuir também uma saída *S* que sinaliza se a contagem é crescente ($S = 1$) ou decrescente ($S = 0$). Para isso, use *flip-flops* do tipo J-K. A Tabela 4.10 traz a sequência de contagem proposta e a Tabela 4.11 traz a tabela verdade do *flip-flop* J-K.

Tabela 4.10 | Sequência de contagem proposta

E	Q_2	Q_1	Q_0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	1	1	1	7
1	1	1	0	6
1	1	0	1	5
1	1	0	0	4
1	0	1	1	3
1	0	1	0	2
1	0	0	1	1
1	0	0	0	0

Fonte: elaborada pelo autor.

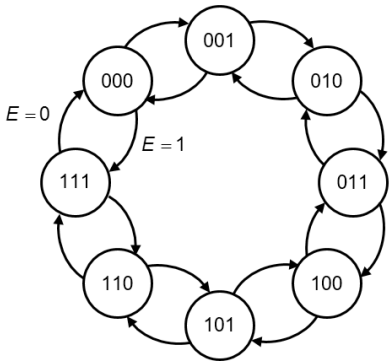
Tabela 4.11 | Tabela verdade do *flip-flop* J-K

$Q(t)$	$Q(t+1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Fonte: elaborada pelo autor.

Passo 1: a Figura 4.37 mostra o diagrama de estados para o problema proposto. Note que para evitar que o diagrama ficasse visualmente poluído, as entradas $E = 0$ e $E = 1$ foram representadas apenas nas transições entre os estados 000 e 111, mas elas se repetem de forma idêntica nas transições dos demais estados.

Figura 4.37 | Diagrama de estados



Fonte: elaborada pelo autor.

Passo 2: a partir da máquina de estados, sabemos que existem oito estados, e que serão necessários três *flip-flops*.

Passo 3: a Tabela 4.12 reproduz a máquina de estados, apresentando os estados e sua tradução para a saída dos *flip-flops* J-K.

Tabela 4.12 | Tabela verdade completa do projeto

E	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0	S
0	0	0	0	0	X	0	X	1	X	1
0	0	0	1	0	X	1	X	X	1	1
0	0	1	0	0	X	X	0	1	X	1
0	0	1	1	1	X	X	1	X	1	1
0	1	0	0	X	0	0	X	1	X	1
0	1	0	1	X	0	1	X	X	1	1
0	1	1	0	X	0	X	0	1	X	1
0	1	1	1	X	1	X	1	X	1	1
1	1	1	1	X	0	X	0	X	1	0
1	1	1	0	X	0	X	1	1	X	0
1	1	0	1	X	0	0	X	X	1	0
1	1	0	0	X	1	1	X	1	X	0

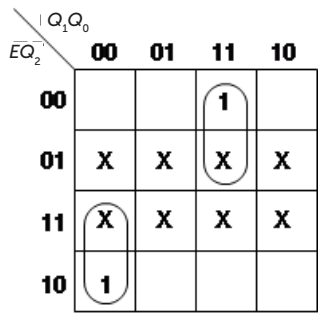
1	0	1	1	0	X	X	0	X	1	0
1	0	1	0	0	X	X	1	1	X	0
1	0	0	1	0	X	0	X	X	1	0
1	0	0	0	1	X	1	X	1	X	0

Fonte: elaborada pelo autor.

A Figura 4.38 apresenta os mapas de Karnaugh correspondentes a cada entrada dos *flip-flops* e da saída *S*.

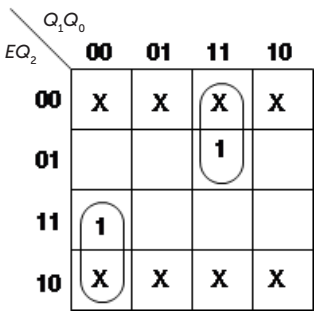
Passo 4: na Figura 4.38 também estão listadas as expressões simplificadas para cada entrada dos *flip-flops* e da saída *S*.

Figura 4.38 | Mapas de Karnaugh com simplificações (a) J_2 (b) K_2 (c) J_1 (d) K_1 (e) J_0 (f) K_0 (g) S



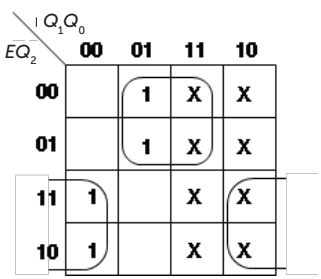
$J_2 = E\bar{Q}_1\bar{Q}_2 + \bar{E}Q_1Q_2$

(a)



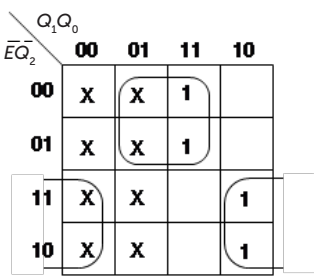
$K_2 = E\bar{Q}_1\bar{Q}_2 + \bar{E}Q_1Q_2$

(b)



$J_1 = E\bar{Q}_0 + \bar{E}Q_0 = E \oplus Q_0$

(c)



$K_1 = E\bar{Q}_0 + \bar{E}Q_0 = E \oplus Q_0$

(d)

$\overline{E}Q_2$ \ Q_1Q_0	00	01	11	10
00	1	X	X	1
01	1	X	X	1
11	1	X	X	1
10	1	X	X	1

$J_0 = 1$
(e)

$\overline{E}Q_2$ \ Q_1Q_0	00	01	11	10
00	X	1	1	X
01	X	1	1	X
11	X	1	1	X
10	X	1	1	X

$K_0 = 1$
(f)

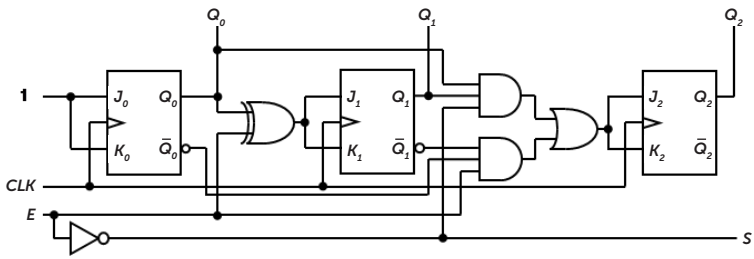
$\overline{E}Q_2$ \ Q_1Q_0	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11				
10				

$S = \overline{E}$
(g)

Fonte: elaborada pelo autor.

Passo 5: o circuito final é apresentado na Figura 4.39.

Figura 4.39 | Circuito final



Fonte: adaptada de Capuano (2014, p. 116).

Pesquise mais

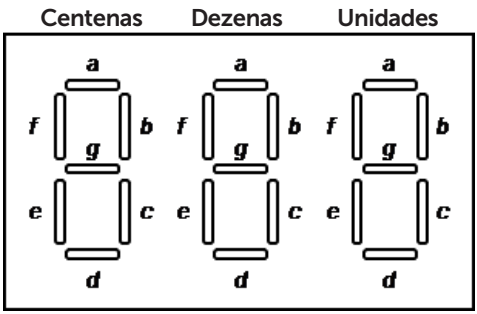
Caro aluno, esperamos que tenha gostado dessa jornada de aprendizado pelo mundo dos sistemas digitais. O que vimos até aqui mal "arranha a superfície" desse tema tão desafiador e atual. De fato, podemos dizer que apenas introduzimos o assunto. Portanto, não pare por aqui. Continue sua pesquisa. Um bom ponto de partida para se aprofundar no assunto é o livro *Sistemas digitais: projetos, otimização e HDLs* (VAHID, 2008), disponível na nossa biblioteca virtual: <https://biblioteca-virtual.com/detalhes/parceiros/5>. Acesso em: 30 out. 2017.

Sem medo de errar

Sua empresa foi recentemente contratada para desenvolver um sistema de monitoração dos espaços disponíveis numa centena de vagas num estacionamento de um pequeno shopping local e prover a sinalização da condição de lotado através de uma indicação luminosa e abaixando a cancela na entrada. Em uma atualização do projeto, o cliente pediu que seja adicionado na entrada do estacionamento um painel indicando o número de vagas.

Para o painel, você usará um display de sete segmentos triplo (um para cada dígito de uma centena), como pode ser visto na Figura 4.40.

Figura 4.40 | Display de sete segmentos triplo



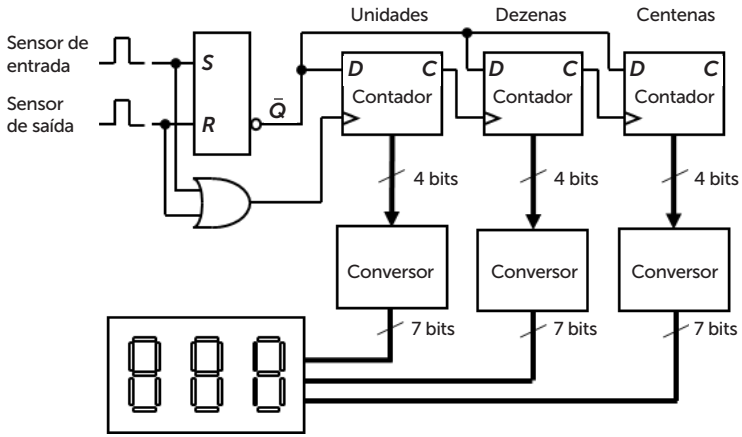
Fonte: elaborada pelo autor.

Cada dígito no display é acionado individualmente e, para facilitar o seu projeto, você decide usar três contadores de década de quatro bits (conta de 0 a 9) crescente/decrescente encadeados. Cada contador deve ter uma entrada para indicar a direção da contagem, crescente para $D = 0$ e decrescente para $D = 1$. O contador deve

ter também uma saída, C, que indica com um nível lógico 1 que a contagem foi reciclada (1001 para 0000 ou 0000 para 1001).

Os quatro bits de saída de cada contador são ligados a decodificadores BCD-7 segmentos, e as saídas do decodificador alimentam o display. A Figura 4.41 traz o diagrama lógico para o circuito proposto.

Figura 4.41 | Diagrama lógico para o circuito contador de carros no estacionamento



Fonte: elaborada pelo autor.

Embora exista no mercado esse tipo de contador encapsulado em um único circuito integrado (CI), projete o circuito lógico de cada contador utilizando o procedimento apresentado nesta seção.

Faça valer a pena

1. Todos os circuitos síncronos, incluindo os registradores e contadores movidos por um *clock*, podem ser representados por máquinas de estados finitos. É sempre recomendável solucionar um problema em um nível funcional mais alto, valendo-se da metodologia das máquinas de estados finitos.

O projeto de um circuito sequencial pode ser dividido em cinco etapas:

I. Obter as expressões lógicas das entradas em função do estado atual.

II. Obter as expressões lógicas das saídas em função do estado atual.

III. Desenhar o diagrama de estados do circuito.

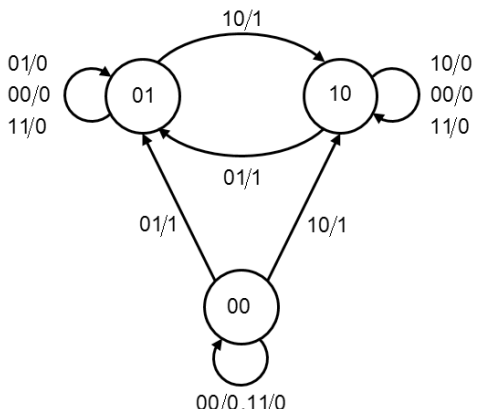
IV. Identificar os estados e calcular a quantidade de *flip-flops* necessários.

V. Desenhar o circuito.

Assinale a opção que apresenta a ordem correta dos procedimentos realizados.
a) I; II; III; IV; V.
b) V; III; I; II; IV.
c) III; IV; I; II; V.
d) V; IV; I; II; III.
e) IV; I; I; III; V.

2. Um **diagrama de estado** representa um modelo gráfico de um sistema sequencial e especifica as transições entre os estados do sistema, as quais, por sua vez, descrevem como um estado particular passa para outro. Considere o diagrama de estado hipotético da Figura 4.42.

Figura 4.42 | Diagrama de estado hipotético



Fonte: elaborada pelo autor.

Sobre este diagrama de estado, avalie as afirmações a seguir:
I. A máquina de estado representada por esse diagrama possui quatro estados, sendo que o estado 11 foi suprimido do diagrama.
II. Essa máquina de estado possui duas entradas, e todas as possibilidades foram representadas para cada estado.
III. A máquina de estado apresenta uma saída que indica um nível lógico ALTO quando há transição de um estado para outro.
IV. No estado 01, as entradas 01 e 10 produzem o mesmo efeito.
Quais afirmações estão corretas?
a) I, II e III.
b) I, III e IV.
c) II e III.
d) III.
e) II, III e IV.

3. Existem dois modelos tradicionais para a implementação de máquinas de estados, são eles: as **máquinas Moore** e as **máquinas Mealy**.

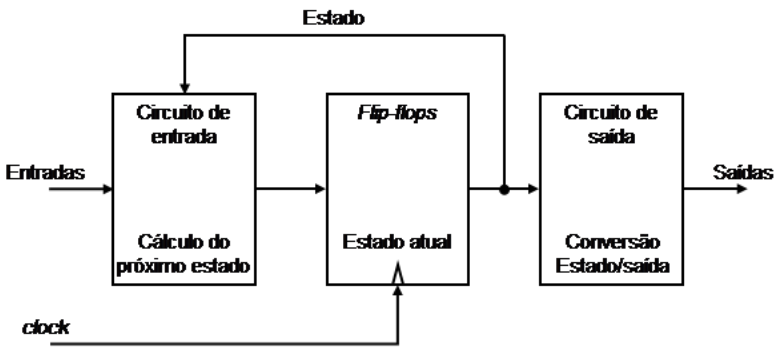
Sobre os modelos de máquinas de estado finito, avalie as afirmações a seguir:

I. Nas máquinas Moore, as saídas dependem apenas do estado atual registrado internamente no sistema.

II. Nas máquinas Mealy, as saídas dependem das entradas e do estado atual registrado internamente no sistema.

III. O diagrama de blocos da Figura 4.43 corresponde a uma máquina Mealy.

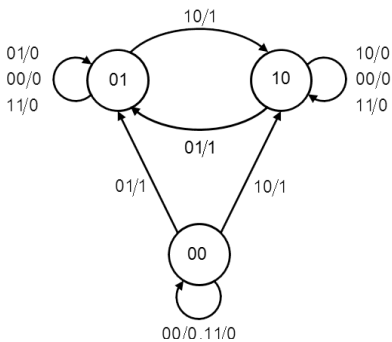
Figura 4.43 | Diagrama de blocos



Fonte: elaborada pelo autor.

IV. O diagrama de estado da Figura 4.44 é referente a uma máquina Moore.

Figura 4.44 | Diagrama de estado



Fonte: elaborada pelo autor.

É correto o que se afirma em:

a) I e II.
b) III e IV.
c) I e III.
d) II e IV.
e) I, II, III e IV.

Referências

CAPUANO, Francisco Gabriel. **Sistemas digitais**: circuitos combinacionais e sequenciais São Paulo: Érica, 2014. 144 p.

FLOYD, Thomas L. **Sistemas digitais**: fundamentos e aplicações. 9. ed. Porto Alegre: Bookman, 2007.

SZANJBERG, Mordka. **Eletrônica digital**: teoria, componentes e aplicações. Rio de Janeiro: LTC, 2014.

TEXAS Instrument. **CD54HC190, CD7HC190, CD54HC191, CD74HC191, CD54HCT191, CD74HCT191 Synchronous up/down counters with down/up mode control datasheet**. Dallas: Texas Instrument, 2003. 31 p. Disponível em: <http://www.ti.com/lit/ds/symlink/cd74hc190.pdf>. Acesso em: 30 out. 2017.

TOCCI, Ronald J.; WIDMER, Neal S.; MOSS, Gregory L. **Sistemas digitais**: princípios e aplicações. 11. ed. São Paulo: Pearson Prentice Hall, 2011.

VAHID, Frank. **Sistemas digitais**: projetos, otimização e HDLs. Porto Alegre: Bookman, 2008. 560 p.

Anotações

Anotações