

série  
Eixos

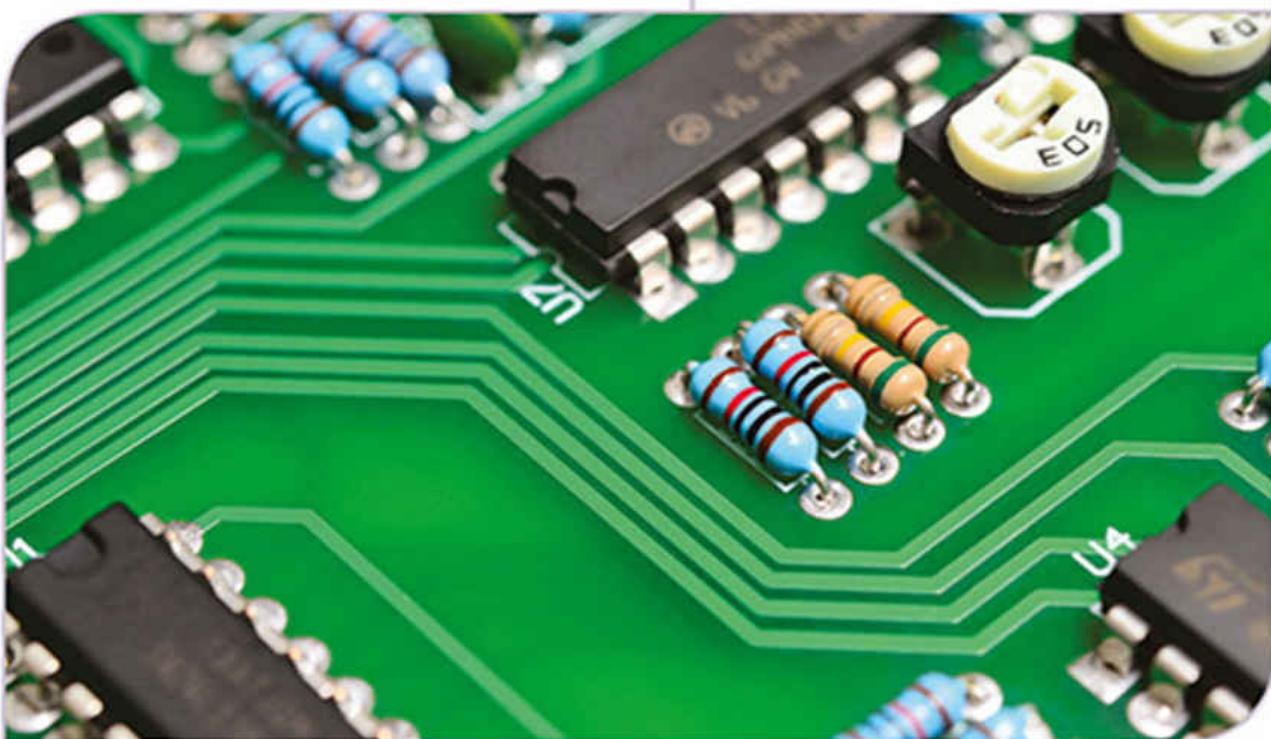
Francisco Gabriel Capuano

Informação e Comunicação

# SISTEMAS DIGITAIS

Circuitos Combinacionais e Sequenciais

 **érica | Saraiva**



Francisco Gabriel Capuano

# Sistemas Digitais

## Circuitos Combinacionais e Sequenciais

**1<sup>a</sup> Edição**

 **érica | Saraiva**

## Agradecimentos

---

À minha esposa Cristina Maria, pelo incentivo e apoio de sempre.

Aos excelentes profissionais Prof. Dr. Alexandre Brincalepe Campo e Caio Novaes Filipini, pela ajuda prestada.

## Sobre o autor

---

**Francisco Gabriel Capuano** Professor de Ensino Superior desde 1982, atuou na coordenação e montagem de diversos cursos. Professor dos Cursos de Engenharia de Controle e Automação, Tecnologia em Sistemas Eletrônicos, Técnicos em Eletrônica e Telecomunicações nas disciplinas: Eletrônica Digital, Eletricidade, Eletrônica Aplicada e Laboratório de Eletricidade do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo.

Professor da Faculdade de Engenharia da Fundação Armando Alvares Penteado (FAAP) desde o ano 2012, nas disciplinas: Sistemas Digitais, Microcontroladores, Energia e Meio Ambiente e Geração de Energia Elétrica.

Coordenador do Ensino Técnico do Liceu de Artes e Ofícios de São Paulo no período de 2008 a 2012.

Atuou como Coordenador dos Cursos Técnicos em Eletrônica e Telecomunicações, Gerente Educacional de Eletromecânica, Gerente Educacional de Telecomunicações e Informática e, ainda, Diretor da Unidade Sede do Centro Federal de Educação Tecnológica de São Paulo.

Participou dos grupos de trabalho do Ministério da Educação, responsável pela confecção das Diretrizes Nacionais Curriculares da Área Industrial e Compatibilização do Perfil Profissional dos Técnicos da Área de Eletrônica dos Países do MERCOSUL.

Especialista (*Lato Sensu*) e Engenheiro em Eletrônica formado pela Faculdade de Engenharia Industrial (FEI).

Autor dos livros: “*Elementos de Eletrônica Digital*” em parceria com Ivan Valeije Idoeta, “*Laboratório de Eletricidade e Eletrônica*” em parceria com Maria Aparecida Mendes Marino e “*Exercícios de Eletrônica Digital*”, publicados pela Editora Érica.

# Sumário

---

## Capítulo 1 - Funções e Portas Lógicas - Circuitos Lógicos

- 1.1 Circuitos lógicos
- 1.2 Função e porta lógica E ou AND
- 1.3 Função e porta lógica OU ou OR
- 1.4 Função e porta lógica NÃO ou NOT
- 1.5 Função e porta lógica NÃO E, NE ou NAND
- 1.6 Função e porta lógica NÃO OU, NOU ou NOR
- 1.7 Expressões booleanas obtidas de circuitos lógicos
- 1.8 Circuitos lógicos obtidos de expressões booleanas
- 1.9 Tabelas-verdade obtidas de expressões booleanas
- 1.10 Expressões booleanas obtidas de tabelas-verdade
- 1.11 Blocos lógicos OU-Exclusivo e Coincidência
  - 1.11.1 Função e porta lógica OU-Exclusivo ou Exclusive OR
  - 1.11.2 Função e porta lógica Coincidência ou NOU Exclusivo ou Exclusive NOR

Agora é com você!

## Capítulo 2 - Álgebra de Boole e Simplificação de Circuitos Lógicos

- 2.1 Simplificação de circuitos lógicos utilizando a álgebra de Boole
- 2.2 Variáveis e expressões na álgebra de Boole
- 2.3 Postulados
  - 2.3.1 Postulados da complementação
  - 2.3.2 Postulado da adição
  - 2.3.3 Postulado da multiplicação
- 2.4 Propriedades

- 2.4.1 Propriedade comutativa
- 2.4.2 Propriedade associativa
- 2.4.3 Propriedade distributiva
- 2.5 Teoremas de De Morgan
  - 2.5.1 1º Teorema de De Morgan
  - 2.5.2 2º Teorema de De Morgan
- 2.6 Identidades auxiliares
- 2.7 Simplificação de expressões utilizando a álgebra de Boole
- 2.8 Simplificação de expressões utilizando os Mapas de Karnaugh
  - 2.8.1 Mapa de Karnaugh para duas variáveis
  - 2.8.2 Mapas de Karnaugh para três variáveis
  - 2.8.3 Mapa de Karnaugh para quatro variáveis
  - 2.8.4 Mapas com condições irrelevantes

Agora é com você!

## Capítulo 3 - Circuitos Combinacionais

- 3.1 Definição e aplicações dos circuitos combinacionais
- 3.2 Códigos
  - 3.2.1 Código BCD 8421
  - 3.2.2 Código 9876543210
- 3.3 Codificadores e Decodificadores
  - 3.3.1 Codificador decimal/binário
  - 3.3.2 Decodificador binário/decimal
  - 3.3.3 Decodificador para display de 7 segmentos
- 3.4 Circuitos aritméticos
  - 3.4.1 Meio Somador
  - 3.4.2 Somador Completo
  - 3.4.3 Meio Subtrator
  - 3.4.4 Subtrator Completo

Agora é com você!

## Capítulo 4 - Circuitos Sequenciais

4.1 Definição geral de circuitos sequenciais

4.2 *Flip-flops*

4.2.1 *Flip-flop RS* básico

4.2.2 *Flip-flop RS* com entrada *clock*

4.2.3 *Flip-flop JK* mestre-escravo

4.2.4 *Flip-flop JK* mestre-escravo com entradas *Preset* e *Clear*

4.2.5 *Flip-Flop T*

4.2.6 *Flip-flop D*

4.3 Contadores

4.3.1 Contadores Assíncronos

4.3.2 Contadores Síncronos

4.4 Registradores de Deslocamento

4.4.1 Conversor Série-Paralelo

4.4.2 Conversor Paralelo-Série

4.4.3 Registrador de entrada série e saída série

4.4.4 Registrador de entrada paralela e saída paralela

4.5 Projetos de sistemas sequenciais

Agora é com você!

## Capítulo 5 - Memórias

5.1 Definição geral e aplicações das memórias eletrônicas

5.2 Classificação das memórias

5.3 Estrutura geral e organização de uma memória

5.4 Memórias ROM

5.4.1 Arquitetura interna das memórias ROM

5.5 Memórias PROM

5.6 Memórias EPROM

5.7 Memórias EEPROM

5.8 Memórias *Flash*

5.9 Memórias RAM

5.9.1 Arquitetura interna das memórias RAM

Agora é com você!

## Capítulo 6 - Introdução aos Computadores Digitais

6.1 Computador digital

6.2 Estrutura geral e organização de um computador

6.2.1 Barramentos internos

6.2.2 Unidade central de processamento

6.2.3 Dispositivos de memórias

6.2.4 Dispositivos de entrada e saída de dados

6.3 Software dos computadores

6.3.1 Linguagens de programação

Agora é com você!

## Bibliografia

# Prefácio

---

Este trabalho tem como proposta apresentar os conceitos básicos dos sistemas digitais com uma linguagem didática e acessível, a fim de serem utilizados na formação técnica de profissionais de diversas áreas que englobem tais sistemas. Ele é o resultado de minha atuação como professor ao longo dos anos, em disciplinas com esses conteúdos para diversos cursos.

O livro apresenta e desenvolve os assuntos que vão desde as portas lógicas simples até as memórias digitais, passando por importantes conceitos e circuitos básicos, seja de lógica combinacional ou sequencial, os quais são partes integrantes e essenciais dos sistemas eletrônicos digitais atuais. O último capítulo, de maneira introdutória é dedicado aos computadores digitais, permitindo ao leitor ou estudante, uma base sólida para um maior aprofundamento nos tópicos abordados, utilizando a vasta bibliografia específica existente no mercado ou ainda, as ferramentas de pesquisa encontradas nos dias atuais.

Além de tudo, considero o entendimento desses assuntos básicos, muito importante, pois, é necessário para a formação do repertório e raciocínio lógico, indispensáveis para os profissionais das áreas envolvidas, trazendo boa capacidade de interpretação, desenvolvimento ou manipulação de sistemas mais avançados.

Espero com isso, contribuir para a formação de profissionais qualificados em diversas áreas, onde se enquadram os sistemas digitais e, sobretudo, como mensagem: estimulá-los a não pararem de buscar novos conhecimentos, pois conforme a experiência cotidiana nos mostra, esse é o caminho para uma trajetória de sucesso e excelência na carreira profissional de qualquer indivíduo.

*O autor*

# 1

## Funções e Portas Lógicas - Circuitos lógicos

### Para começar

Este capítulo tem como objetivo proporcionar os conhecimentos básicos dos sistemas digitais e suas aplicações em circuitos lógicos. Serão estudadas as funções e as portas lógicas utilizadas na formação de circuitos lógicos diversos, bem como a manipulação de expressões booleanas e tabelas-verdade, necessárias para obtê-los.

### 1.1 Circuitos lógicos

São denominados de **circuitos lógicos**, o arranjo de um pequeno grupo de circuitos básicos padronizados conhecidos como portas lógicas, que realizam funções de **lógica digital** dentro do campo, também conhecido como **eletrônica digital**.

Na prática, as portas lógicas são encontradas dentro de circuitos integrados comerciais específicos e, ainda fazem parte da estrutura interna de dispositivos mais complexos, tais como: microprocessadores, microcontroladores e outros circuitos integrados digitais utilizados em computadores e aparelhos eletrônicos.

Através da utilização conveniente dessas portas, podemos implementar todas as expressões geradas pela álgebra de Boole, que é um estudo onde estão todas as leis fundamentais da lógica digital.

A seguir, é feito o estudo das principais funções lógicas que na realidade derivam dos postulados da álgebra de Boole, sendo as variáveis e expressões envolvidas denominadas de booleanas.

Nas funções lógicas, temos apenas dois níveis lógicos distintos:

O nível lógico 0 (zero) e o nível lógico 1 (um).

O nível lógico 0 representa, por exemplo, ausência de tensão, chave aberta, não etc. O nível lógico 1 representa, presença de tensão, chave fechada, sim etc.

Note que se representarmos por 0 uma situação, representamos por 1 a situação contrária. Deve-se salientar que cada variável booleana da função lógica pode assumir somente duas situações distintas, 0 ou 1.

### Amplie seus conhecimentos

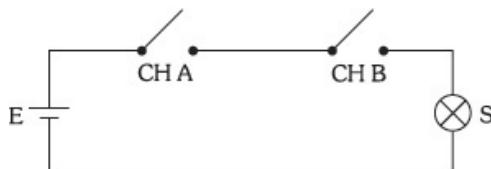
Em 1854, o matemático inglês George Boole (1815-1864), através da obra intitulada *An Investigation of the Laws of Thought*, apresentou um sistema matemático de análise lógica conhecido como álgebra de Boole.

Apenas em 1938, o engenheiro americano Claude Elwood Shannon utilizou as teorias da álgebra de Boole para a solução de problemas de circuitos de telefonia com relés, tendo publicado um trabalho denominado *Symbolic Analysis of Relay and Switching*, praticamente introduzindo na área tecnológica o campo da eletrônica digital.

## 1.2 Função e porta lógica E ou AND

A função **E** é aquela que executa a multiplicação de duas ou mais variáveis booleanas. É também conhecida como função **AND**. Sua representação algébrica para duas variáveis é  $S = A \cdot B$ , onde se lê  $S = A$  e  $B$ .

Para melhor compreensão, vamos utilizar e analisar o circuito representativo da função E visto na Figura 1.1.



Convenções: chave aberta = 0  
chave fechada = 1

lâmpada apagada = 0  
lâmpada acesa = 1

Figura 1.1 - Circuito representativo da função E.

Situações possíveis:

- 1) Se tivermos a chave A aberta (0) e a chave B aberta (0), neste circuito não circula corrente; logo, a lâmpada permanece apagada (0):  $A = 0, B = 0 \Rightarrow S = A \cdot B = 0$ .
- 2) Se tivermos a chave A aberta (0) e a chave B fechada (1), a lâmpada permanece apagada (0):  $A = 0, B = 1 \Rightarrow S = 0$ .
- 3) Se tivermos a chave A fechada (1) e a chave B aberta (0), a lâmpada permanece apagada:  $A = 1, B = 0 \Rightarrow S = 0$ .
- 4) Se tivermos, agora, a chave A fechada (1) e a chave B fechada (1), a lâmpada acende, pois circula corrente:  $A = 1, B = 1 \Rightarrow S = 1$ .

Analizando as situações, concluímos que só teremos a lâmpada acesa quando as chaves A e B estiverem fechadas.

Chamamos **tabela-verdade** um mapa onde colocamos todas as possíveis situações com seus respectivos resultados. Na tabela encontramos o modo como a função comporta-se. A seguir, vamos apresentar a tabela-verdade de uma função E ou AND para duas variáveis de entrada como mostra a Tabela 1.1.

Tabela 1.1 - Tabela-verdade da função E de duas variáveis

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

A porta lógica E é um circuito que executa a função E, sendo representada na prática através do símbolo mostrado na Figura 1.2.

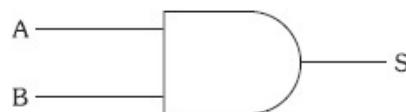


Figura 1.2 - Porta E de duas entradas.

Como já dissemos, a porta E executa a tabela-verdade da função E, ou seja, teremos a saída no estado 1 se, e somente se, as duas entradas forem iguais a 1, e teremos a saída igual a 0 nos demais casos.

Até agora, descrevemos a função E para duas variáveis de entrada. Podemos estender esse conceito para qualquer número de entradas. Para exemplificar, mostraremos uma porta E de três variáveis de entrada, sua tabela-verdade e ainda sua expressão booleana. Veja a Tabela 1.2.

Notamos que a tabela-verdade mostra as oito possíveis combinações das variáveis de entrada e seus respectivos resultados na saída. O número de situações possíveis é igual a  $2^N$ , em que N é o número de variáveis de entrada. No exemplo:  $N = 3 \therefore 2^3 = 8$ .

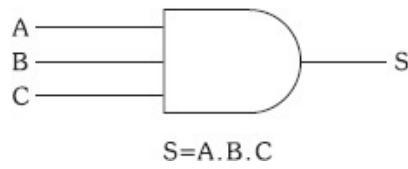


Figura 1.3 - Porta E de três entradas e sua expressão booleana.

Tabela 1.2 - Tabela-verdade da Porta E de três entradas

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

### 1.3 Função e porta lógica OU ou OR

A função OU é aquela que assume valor 1 quando uma ou mais variáveis da entrada forem iguais a 1 e assume valor 0 se, e somente se, todas as variáveis de entrada forem iguais a 0. Sua representação algébrica para duas variáveis de entrada é  $S = A + B$ , em que se lê S = A **ou** B.

O termo **OR** é também utilizado na prática.

Para entendermos melhor a função OU, vamos representá-la pelo circuito da Figura 1.4 e analisar as situações possíveis.

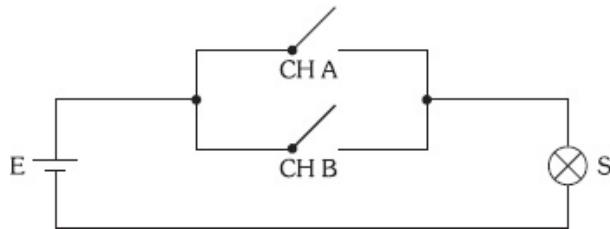


Figura 1.4 - Circuito representativo da função OU.

Usaremos as mesmas convenções do circuito representativo da função E, visto anteriormente.

Situações possíveis:

- 1) Se tivermos a chave A aberta (0) e a chave B aberta (0), no circuito não circula corrente; logo, a lâmpada permanece apagada (0):  $A = 0, B = 0 \Rightarrow S = A + B = 0$ .
- 2) Se tivermos a chave A aberta (0) e a chave B fechada (1), circula corrente pela chave B e a lâmpada acende (1):  $A = 0, B = 1 \Rightarrow S = 1$ .
- 3) Se tivermos a chave A fechada (1) e a chave B aberta (0), circula corrente pela chave A e a lâmpada acende (1):  $A = 1, B = 0 \Rightarrow S = 1$ .
- 4) Se tivermos a chave A fechada (1) e a chave B fechada (1), circula corrente pelas duas chaves e a lâmpada acende (1):  $A = 1, B = 1 \Rightarrow S = 1$ .

Para o caso  $A = 1$  e  $B = 1$ , a soma  $A + B = 1$ , a princípio estranha, é verdadeira, pois, como veremos mais à frente, trata-se de uma soma booleana derivada do postulado da adição da álgebra de Boole.

Notamos pelas situações que temos a lâmpada ligada quando chA ou chB ou ambas as chaves estiverem ligadas.

A Tabela 1.3 apresenta a tabela-verdade da função OU ou OR para duas variáveis de entrada. Nesta tabela-verdade temos as situações possíveis com os respectivos valores que a função OU assume.

Tabela 1.3 - Tabela-verdade da função OU de duas variáveis

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

Representaremos a porta OU pelo símbolo da Figura 1.5. É a porta que executa a função OU.



Figura 1.5 - Porta OU de duas entradas.

A porta OU executa a tabela-verdade de função OU, ou seja, teremos a saída igual a 1 quando uma ou mais variáveis de entrada forem iguais a 1 e 0 se, e somente se, todas as variáveis de entrada forem iguais a 0.

Podemos estender o conceito para mais de duas variáveis de entrada. Como exemplo, vamos mostrar uma porta OU, sua tabela-verdade e a expressão booleana com quatro variáveis de entrada. Veja a Figura 1.6.

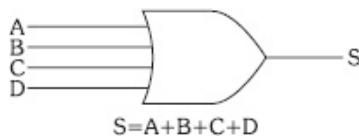


Figura 1.6 - Porta OU de quatro entradas e sua expressão booleana.

Na Tabela 1.4, as quatro variáveis de entrada possibilitam 16 combinações ( $2^4 = 16$ ).

Tabela 1.4 - Tabela da porta OU de quatro entradas

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

## 1.4 Função e porta lógica NÃO ou NOT

A função NÃO é aquela que inverte ou complementa o estado da variável, ou seja, se a variável estiver em 0, a saída vai para 1, e se estiver em 1, a saída vai para 0. É representada algebraicamente da seguinte forma:  $S = \bar{A}$  ou  $S = A'$ , em que se lê **A barra** ou **NÃO A**.

Esta barra ou apóstrofo sobre a letra que representa a variável significa que ela sofre uma inversão. Também podemos dizer que  $\bar{A}$  significa a negação de A.

Para entendermos melhor a função NÃO, vamos representá-la pelo circuito da Figura 1.7. Analisaremos utilizando as mesmas convenções dos casos anteriores.

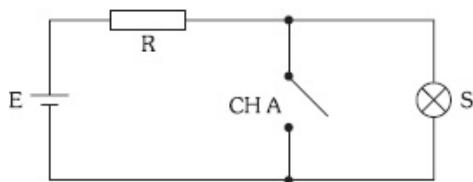


Figura 1.7 - Circuito representativo da função NÃO.

Situações possíveis:

- 1) Quando a chave A estiver aberta (0), passa corrente pela lâmpada e ela acende (1):  $A = 0 \Rightarrow S = \bar{A} = 1$ .
- 2) Quando a chave A estiver fechada (1), curto-circuitamos a lâmpada e ela se apaga (0):  $A = 1 \Rightarrow S = \bar{A} = 0$ .

A Tabela 1.5 apresenta casos possíveis da função NÃO.

Tabela 1.5 - Tabela-verdade da função NÃO

A	S
0	1
1	0

O **inversor** é o bloco lógico que executa a função NÃO. Suas representações simbólicas encontram-se na Figura 1.8.

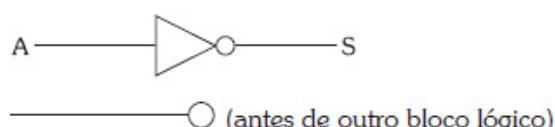


Figura 1.8 - Formas de representação do inversor.

A função NÃO ou complementar também é conhecida como NOT.

## 1.5 Função e porta lógica NÃO E, NE ou NAND

Como o próprio nome “NÃO E” diz, essa função é uma composição da função E com a função NÃO, ou seja, temos a função E invertida. É representada algebricamente da seguinte forma:

$$S = (\overline{A \cdot B}) \Rightarrow \text{O traço indica que temos a inversão do produto } A \cdot B.$$

A Tabela 1.6 apresenta a função NE para duas variáveis de entrada.

Tabela 1.6 - Tabela-verdade da função NE para duas variáveis

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

Pela tabela-verdade, podemos notar que essa função é o inverso da função E.

A porta NE é o bloco lógico que executa a função NE. Sua representação simbólica é vista na Figura 1.9.

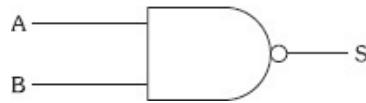


Figura 1.9 - Porta NE de duas entradas.

Podemos também formar uma porta NE pela composição de uma porta E com um inversor ligado à sua saída formando um circuito. A Figura 1.10 mostra essa situação.

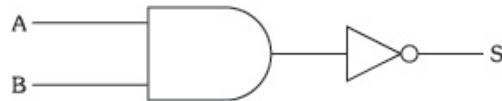


Figura 1.10 - Porta E com inversor formando uma porta NE.

A porta NE como outros blocos lógicos, pode ter duas ou mais entradas.

## 1.6 Função e porta lógica NÃO OU, NOU ou NOR

Analogamente à função NE, a função NOU é a composição da função NÃO com a função OU, ou seja, a função NOU será o inverso da função OU. É representada da seguinte forma:

$$S = (\overline{A + B}) \Rightarrow \text{O traço indica a inversão da soma booleana } (A + B).$$

A Tabela 1.7 apresenta a função NOU para duas variáveis de entrada.

Tabela 1.7 - Tabela-verdade da função NOU para duas variáveis

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

Podemos notar pela tabela-verdade que a função NOU representa a função OU invertida.

A porta NOU é o bloco lógico que executa a função NOU. Sua representação simbólica está na Figura 1.11.



Figura 1.11 - Porta NOU de duas entradas.

De maneira análoga, podemos formar uma porta NOU utilizando uma OU e um inversor ligado à sua saída formando um circuito. Essa situação é vista na Figura 1.12.

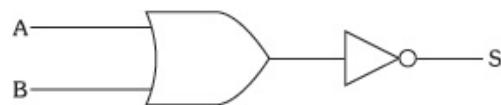


Figura 1.12 - Porta OU e inversor formando uma Porta NOU.

### Amplie seus conhecimentos

Na prática as portas lógicas são encontradas dentro de circuitos integrados comerciais específicos, pertencentes a uma determinada **família** de circuitos lógicos, como ilustra a

Figura 1.13.

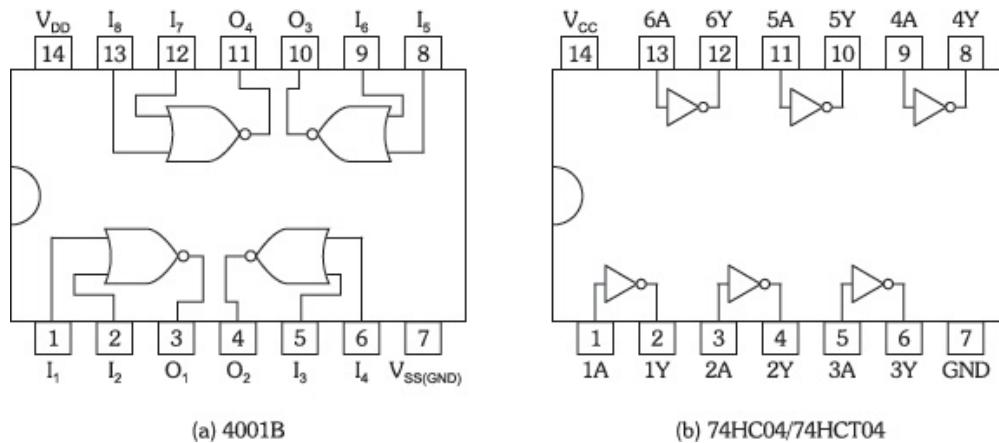


Figura 1.13 - Pinagem e configuração interna de circuitos integrados da família CMOS: (a) 4001B: 4 portas NOU de 2 entradas e (b) 74HC04/74HCT04: 6 inversores CMOS.

Podemos ter portas NOU com mais de duas entradas.

## 1.7 Expressões booleanas obtidas de circuitos lógicos

Todo circuito lógico executa uma expressão booleana e, por mais complexo que seja, é formado pela interligação das portas lógicas básicas. Podemos obter a expressão booleana que é executada por um circuito lógico qualquer. Para mostrar o procedimento, vamos obter a expressão que o circuito da Figura 1.14 executa.

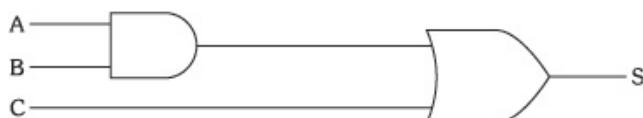


Figura 1.14 - Exemplo de circuito lógico.

Para facilitar, vamos dividir o circuito em duas partes como mostra a Figura 1.15.

Na saída S<sub>1</sub>, temos o produto A · B, sendo este bloco uma porta E, sua expressão de saída será S<sub>1</sub> = A · B. Como S<sub>1</sub> é injetada em uma das entradas da porta OU pertencente à segunda parte do circuito e na outra entrada está a variável C, a expressão de saída será S = S<sub>1</sub> + C.

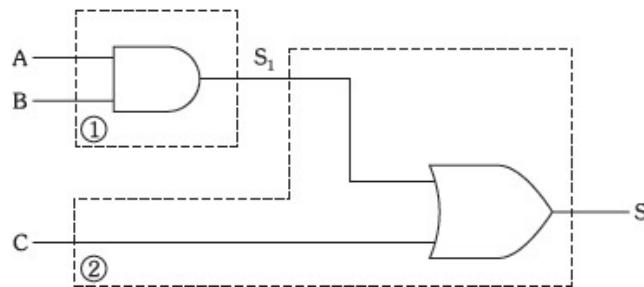


Figura 1.15 - Circuito dividido em duas partes.

Para determinarmos a expressão final, basta substituirmos a expressão de  $S_1$  na expressão anterior, obtendo então:  $S = A \cdot B + C$ , que é a expressão que o circuito executa.

Outra maneira mais simples de resolvemos o problema é escrevermos nas saídas dos diversos blocos básicos do circuito as expressões por eles executadas. A Figura 1.16 ilustra esse procedimento.



Figura 1.16 - Outro modo de obtenção da expressão de saída.  $\therefore S = A \cdot B + C$ .



## Exercícios resolvidos

- 1.1) Escreva a expressão booleana executada pelo circuito da Figura 1.17.

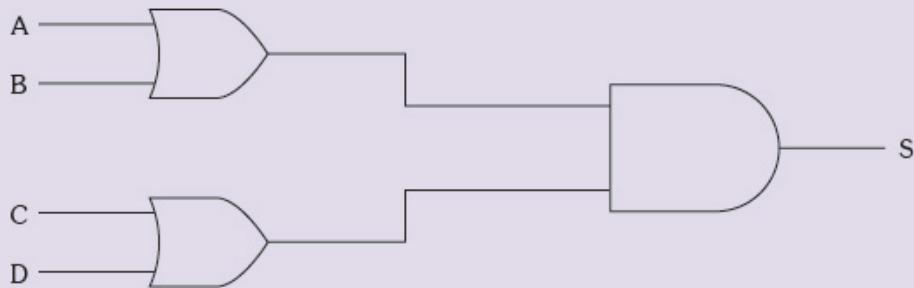


Figura 1.17 - Circuito do exercício 1.

Vamos, agora, escrever as expressões de saída de cada bloco básico do circuito:

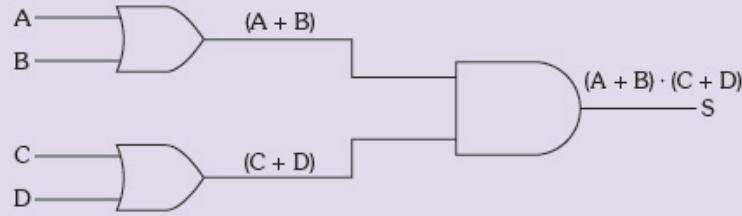


Figura 1.18 - Circuito com as expressões colocadas.  $\therefore S = (A + B) \cdot (C + D)$ .

- 1.2) Determine a expressão booleana característica do circuito da Figura 1.19.

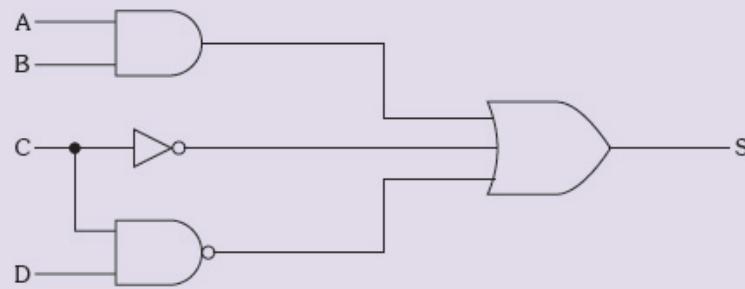


Figura 1.19 - Circuito do exercício 2.

Seguindo o processo descrito, temos:

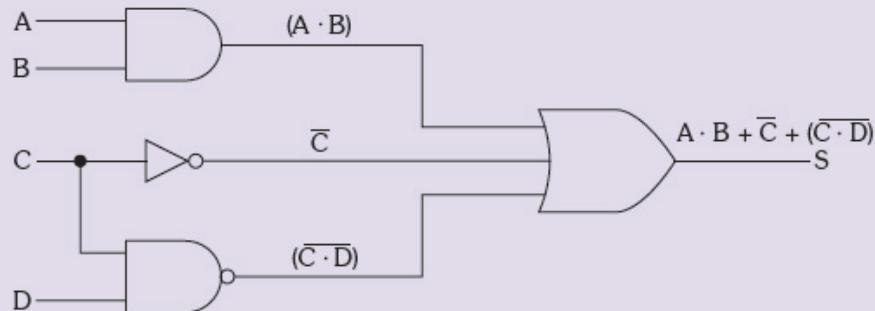


Figura 1.20 - Circuito com as expressões colocadas.  $\therefore S = A \cdot B + \bar{C} + (\bar{C} \cdot D)$ .

- 1.3) Idem, para o circuito da Figura 1.21.

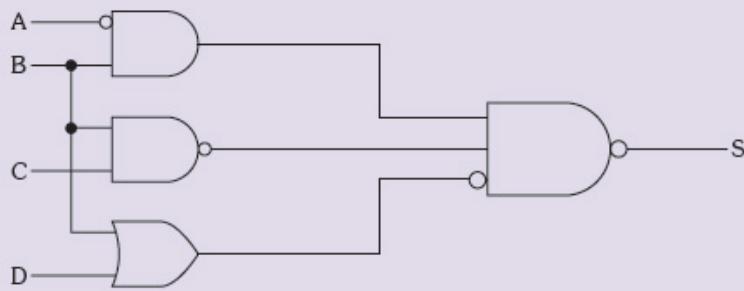


Figura 1.21 - Circuito do exercício 3.

Antes da solução, convém recordarmos que os círculos colocados nas terminais de entrada junto das portas representam também inversores. Solucionando, temos:

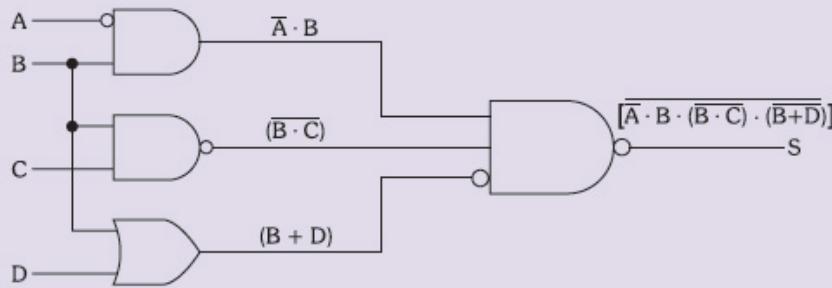


Figura 1.22 - Circuito com as expressões colocadas.  $\therefore S = [\bar{A} \cdot B \cdot (B \cdot C) \cdot (\bar{A} + D)]$ .

## 1.8 Circuitos lógicos obtidos de expressões booleanas

Vimos, no tópico anterior, que podemos obter uma expressão booleana que um circuito lógico executa. Podemos também desenhar um circuito lógico que executa uma expressão booleana qualquer, ou seja, podemos desenhar um circuito a partir de sua expressão característica.

O método para a resolução consiste em identificar as portas lógicas na expressão e desenhá-las com as respectivas ligações a partir das variáveis de entrada. Para exemplificar, vamos obter o circuito que executa a expressão:  $S = (A + B) \cdot C \cdot (B + D)$ .

Vamos solucionar respeitando a hierarquia das funções da aritmética elementar, ou seja, iniciaremos a solução primeiramente pelos parênteses.

Para o primeiro parêntese, temos a soma booleana  $A + B$ ; logo, o circuito que o executa é uma porta OU. Para o segundo existe a soma booleana  $B + D$ ; logo, o circuito é uma porta OU. Até o momento temos:

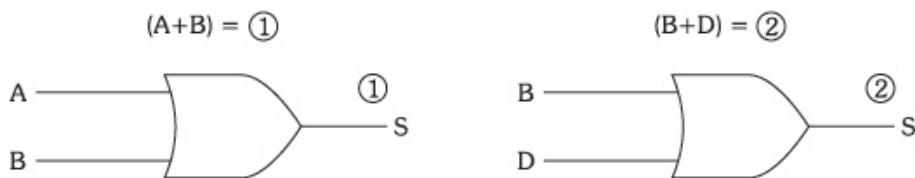


Figura 1.23 - Resolução do exemplo por partes.

A seguir, ocorre uma multiplicação booleana dos dois parênteses, juntamente com a variável C, sendo o circuito que executa esta multiplicação uma porta E. Veja a Figura 1.24.

Substituindo as saídas ① e ② no bloco da Figura 1.24, obtemos o circuito completo, visto na Figura 1.25.

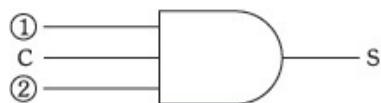


Figura 1.24 - Composição parcial do circuito.

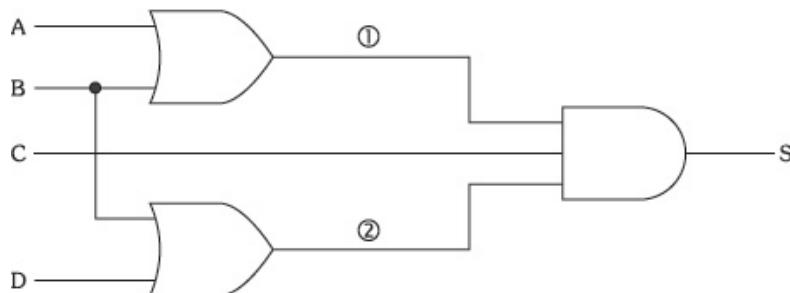


Figura 1.25 - Circuito completo.



## Exercícios resolvidos

- 1.1) Desenhe o circuito responsável por executar a expressão booleana  $S = A \cdot B \cdot C + (A+B) \cdot C$ .

Primeiramente, para identificar as portas lógicas, vamos numerar cada termo da expressão:

$$S = \underbrace{A \cdot B \cdot C}_{(1)} + \underbrace{(A+B) \cdot C}_{\underbrace{\begin{array}{c} (2) \\ (3) \end{array}}_{(4)}}$$

Assim sendo, temos:

- ① porta E com A, B e C.
- ② porta OU com A e B.
- ③ porta E com ② e C.
- ④ porta OU com ① e ③.

Para facilitar as ligações, podemos utilizar uma rede ou barra de variáveis de entrada. A Figura 1.26 mostra o circuito final, composto pela ligação das portas envolvidas com uma rede de variáveis de entrada.

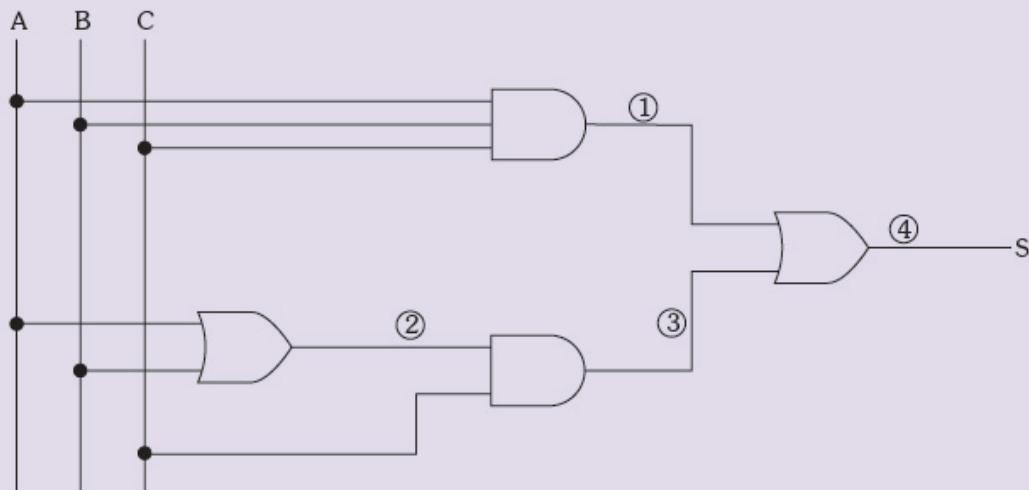


Figura 1.26 - Circuito final obtido.

1.2) Idem para a expressão:  $S = [(\overline{A} + B) + (\overline{C} \cdot D)] \cdot \overline{D}$

Solucionando, conforme já explicado, temos:

$$S = \underbrace{[(\overline{A} + B) + (\overline{C} \cdot D)]}_{\underbrace{\begin{array}{c} (1) \\ (2) \\ (3) \end{array}}_{(4)}} \cdot \overline{D}$$

Desenhando e interligando as portas a partir de uma rede de variáveis de entrada, obtemos o circuito da Figura 1.27.

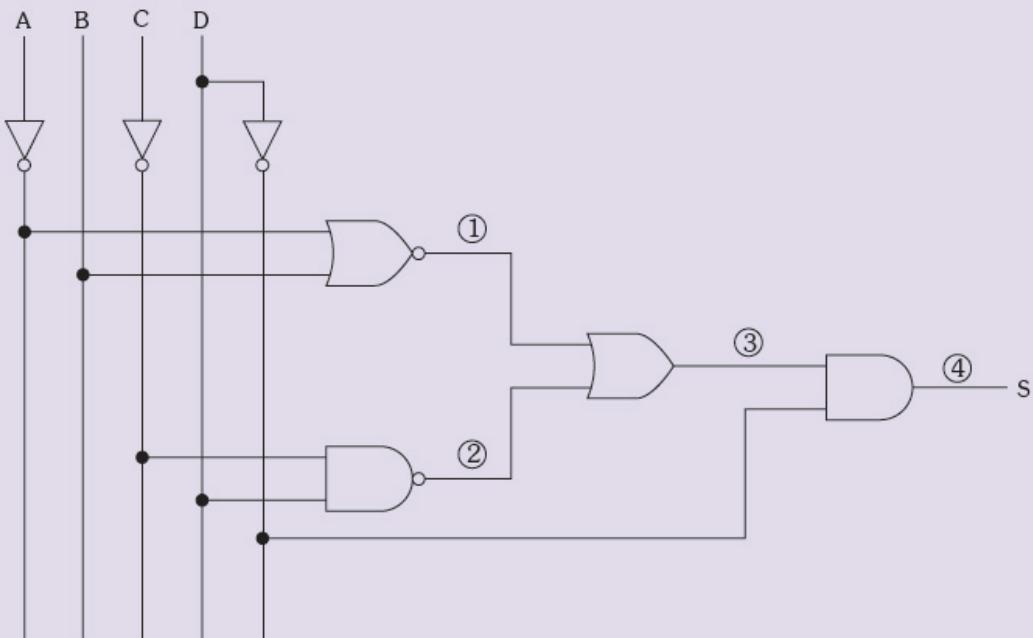


Figura 1.27 - Circuito final obtido.

## 1.9 Tabelas-verdade obtidas de expressões booleanas

Uma maneira de fazer o estudo de uma função booleana é a utilização da tabela-verdade que, como vimos anteriormente, é um mapa onde se colocam todas as situações possíveis de uma dada expressão, juntamente com o valor por essa assumido.

Para extrairmos a tabela-verdade de uma expressão, acompanhamos o seguinte procedimento:

- 1) montamos o quadro de possibilidades.
- 2) montamos colunas para os vários membros da expressão.
- 3) preenchemos essas colunas com seus resultados.
- 4) montamos uma coluna para o resultado final.
- 5) preenchemos essa coluna com os resultados finais.

Para esclarecer este processo, vamos utilizar a seguinte expressão:

$$S = A \cdot \bar{B} \cdot C + A \cdot \bar{D} + \bar{A} \cdot B \cdot D$$

A expressão possui quatro variáveis: A, B, C e D; logo, temos  $2^4$  possibilidades de combinação de entrada.

Vamos montar o quadro de possibilidades com quatro variáveis de entrada, três colunas auxiliares, sendo uma para cada membro da expressão, e uma coluna para o resultado final (S):

**Tabela 1.8 - Quadro de possibilidades da expressão**

A	B	C	D	1º membro $A \cdot \bar{B} \cdot C$	2º membro $A \cdot \bar{D}$	3º membro $\bar{A} \cdot B \cdot D$	Resultado final S
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	0
0	1	1	1	0	0	1	1
1	0	0	0	0	1	0	1
1	0	0	1	0	0	0	0
1	0	1	0	1	1	0	1
1	0	1	1	1	0	0	1
1	1	0	0	0	1	0	1
1	1	0	1	0	0	0	0
1	1	1	0	0	1	0	1
1	1	1	1	0	0	0	0

Convém observar que na coluna relativa ao 1º membro colocamos os resultados da multiplicação de A com o inverso da variável B e C ( $A \cdot \bar{B} \cdot C$ ), na do 2º membro, o resultado de  $A\bar{D}$  e na do 3º, o resultado de  $\bar{A} \cdot B \cdot D$ . Na coluna de resultado final (S) colocamos a soma booleana (porta OU) dos três termos da expressão.

Outro modo de resolução, porém, mais prático, consiste no preenchimento direto da coluna com o resultado final. Para isso, basta montar o quadro de possibilidades conforme o número de variáveis, reconhecer na expressão operações notáveis que permitem a conclusão do resultado final de imediato e, por exclusão, ir executando as operações até o preenchimento total da tabela.

Para melhor entendimento, vamos levantar a tabela da expressão  $S = \bar{A} + B + A \cdot \bar{B} \cdot \bar{C}$ , utilizando esse modo mais prático.

Primeiramente, vamos montar o quadro de possibilidades para as três variáveis envolvidas na expressão como mostra a Tabela 1.9.

Tabela 1.9 - Quadro de possibilidades da expressão

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

Logo após, vamos preencher a tabela utilizando os casos notáveis, que permitem a conclusão do resultado final imediato:

- 1) Nos casos em que  $A = 0$  ( $\bar{A} = 1$ ), temos  $S = 1$ , pois sendo  $\bar{A} = 1$ , temos na expressão  $S = 1 + B + A \cdot \bar{B} \cdot \bar{C} = 1$  (qualquer que sejam os valores assumidos pela variável B ou pelo termo  $A \cdot \bar{B} \cdot \bar{C}$ ).
- 2) Nos casos remanescentes em que  $B = 1$ , temos  $S = 1$ , pois, da mesma forma que nos casos anteriores,  $S = \bar{A} + 1 + A \cdot \bar{B} \cdot \bar{C} = 1$ .
- 3) O termo  $A \cdot \bar{B} \cdot \bar{C}$  será igual a 1, somente no caso remanescente 100, levando para este caso a saída da expressão para 1 ( $S = 1$ ).
- 4) Por exclusão, ou ainda, por substituição dos valores, concluímos que no último caso (101), temos na saída  $S = 0$ .

A Tabela 1.10 apresenta o resultado com todos os casos preenchidos e assinalados conforme a análise efetuada.

Tabela 1.10 - Resultados obtidos

A	B	C	S	
0	0	0	1	①
0	0	1	1	
0	1	0	1	
0	1	1	1	
1	0	0	1	③
1	0	1	0	④
1	1	0	1	②
1	1	1	1	



## Exercícios resolvidos

1.1) Prove as identidades relacionadas a seguir:

- a)  $\overline{A} \cdot \overline{B} \neq \overline{A \cdot B}$
- b)  $\overline{A} + \overline{B} \neq \overline{A + B}$
- c)  $\overline{A} \cdot \overline{B} = \overline{A + B}$
- d)  $\overline{A} + \overline{B} = \overline{A \cdot B}$

Podemos provar essas identidades levantando as respectivas tabelas-verdade. Para facilitar, vamos colocar um quadro de possibilidades (duas variáveis: A e B) e colunas para os termos comuns entre as expressões. Ver Tabela 1.11:

Tabela 1.11 - Tabela-verdade com os termos das expressões e resultados

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

Como podemos notar na tabela, os termos  $\overline{A} \cdot \overline{B}$  e  $\overline{A \cdot B}$  assumem resultados diferentes (a) para as mesmas possibilidades de entrada, o mesmo ocorrendo com  $\overline{A} + \overline{B}$  e  $\overline{A + B}$  (b), porém, o termo  $A \cdot B$  assume resultado idêntico a  $\overline{A} + \overline{B}$  (c), o mesmo acontecendo entre  $\overline{A} + \overline{B}$  e  $\overline{A \cdot B}$  (d).

1.2) Levante a tabela-verdade da expressão:

$$S = (A + B) \cdot (\overline{B} \cdot C)$$

Seguindo o processo já visto, temos:

- 1) Pela análise do termo  $(A + B)$ , concluímos que nos casos da tabela em que  $A = B = 0$ , temos  $S = 0$ , pois  $S = (0 + 0) \cdot (\overline{B} \cdot \overline{C})$ . Para qualquer outro caso remanescente, esse termo será igual a 1, sendo necessária a análise de  $(\overline{B} \cdot \overline{C})$ , para a conclusão dos outros resultados finais.
- 2) Nos casos remanescentes da tabela em que  $B = 0$  ou  $C = 0$ , temos  $S = 1$ , pois  $(\overline{0} \cdot \overline{C})$  ou  $(\overline{B} \cdot 0)$  são iguais a 1, que multiplicado por  $(A + B) = 1$ , resulta em  $S = 1$ .
- 3) Nos dois últimos casos em que  $B = C = 1$ , temos  $S = 0$ , pois  $S = (A + B) \cdot (\overline{1} \cdot \overline{1}) = 0$ .

Passando os casos analisados para a tabela, temos:

Tabela 1.12 - Tabela-verdade com os resultados

A	B	C	S	
0	0	0	0	①
0	0	1	0	
0	1	0	1	②
0	1	1	0	
1	0	0	1	③
1	0	1	1	
1	1	0	1	③
1	1	1	0	

- 1.3) Monte a tabela-verdade da expressão:

$$S = [(\overline{A + B}) \cdot \overline{C}] + [\overline{D} \cdot (\overline{B + C})].$$

Solução:

- 1) Nos casos em que  $C = 0$ , temos  $S = 1$ , pois o 1º termo da expressão será igual a 1:

$$S = [(\overline{A + B}) \cdot \overline{0}] + \dots = 1.$$

- 2) O mesmo ocorre nos casos remanescentes em que  $D = 0$ , pois temos que  $S = \dots + [(\overline{0} \cdot (\overline{B + C}))] = 1$ .
- 3) Nos casos remanescentes em que  $B = 1$ , temos  $S = 0$ , pois estando B presente nos dois termos, temos  $S = [(\overline{A + 1})\overline{1}] + [\overline{1} \cdot (\overline{1 + C})] = 0 + 0 = 0$  (para esses casos restantes  $C = 1$  e  $D = 1$ ).

- 4) Sendo  $B = 0$  para os casos restantes, em que  $A = 0$ , temos  $S = 1$ , pois  
 $S = [(0 + 0) \cdot 1] + \dots = 1$ .
- 5) No último caso (1011), por simples substituição, concluímos que  $S = 0$ .

A Tabela 1.13 mostra o resultado final com todos esses casos preenchidos e assinalados.

Tabela 1.13 - Tabela-verdade com os resultados

A	B	C	D	S	
0	0	0	0	1	①
0	0	0	1	1	
0	0	1	0	1	②
0	0	1	1	1	
0	1	0	0	1	①
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	0	③
1	0	0	0	1	
1	0	0	1	1	①
1	0	1	0	1	
1	0	1	1	0	⑤
1	1	0	0	1	
1	1	0	1	1	①
1	1	1	0	1	
1	1	1	1	0	③

## 1.10 Expressões booleanas obtidas de tabelas-verdade

Até aqui aprendemos como obter expressões a partir de circuitos, circuitos a partir de expressões e tabelas-verdade a partir de circuitos ou expressões. Esse item ensina como obter expressões e circuitos a partir de tabelas-verdade, sendo este o caso mais comum em projetos práticos, pois geralmente necessitamos representar situações através de tabelas-verdade e a partir delas obter a expressão booleana e, consequentemente, o circuito lógico.

Para demonstrar este procedimento, vamos obter a expressão da Tabela 1.14.

Tabela 1.14 - Tabela-verdade exemplo

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

Observando a tabela, notamos que a expressão é verdadeira ( $S = 1$ ) nos casos em que  $A = 0$  e  $B = 0$  ou  $A = 1$  e  $B = 0$  ou  $A = 1$  e  $B = 1$ . Para obter a expressão, basta montar os termos relativos aos casos em que a expressão for verdadeira e somá-los:

Caso 00:  $S = 1$  quando  $A = 0$  e  $B = 0$  ( $\bar{A} = 1$  e  $\bar{B} = 1$ )  $\Rightarrow \bar{A} \cdot \bar{B}$

Caso 10:  $S = 1$  quando  $A = 1$  e  $B = 0$  ( $A = 1$  e  $\bar{B} = 1$ )  $\Rightarrow A \cdot \bar{B}$

Caso 11:  $S = 1$  quando  $A = 1$  e  $B = 1$   $\Rightarrow A \cdot B$

$$\therefore S = \bar{A} \cdot \bar{B} + A \cdot \bar{B} + A \cdot B$$

Notamos que o método permite obter, qualquer que seja a tabela, uma expressão padrão formada sempre pela soma de produtos.

### Fique de olho!

O Capítulo 2, relativo à álgebra de Boole, estuda o processo de simplificação de expressões booleanas, possibilitando a obtenção de expressões e consequentemente circuitos equivalentes mais simplificados. A simplificação da expressão obtida no exemplo, utilizando as técnicas a serem estudadas é:  $S = A + \bar{B}$ .



## Exercícios resolvidos

- 1.1) Determine a expressão que executa a Tabela 1.15 e desenhe o circuito lógico.

Tabela 1.15 - Tabela-verdade do exercício 1

A	B	C	s
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Para solucionar, extraímos os casos em que a expressão é verdadeira ( $S = 1$ ): 000 ou 010 ou 110 ou 111.

$$\therefore S = \overline{A} \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C} + A \cdot B \cdot \overline{C} + A \cdot B \cdot C$$

Da expressão extraímos o circuito lógico mostrado na Figura 1.28.

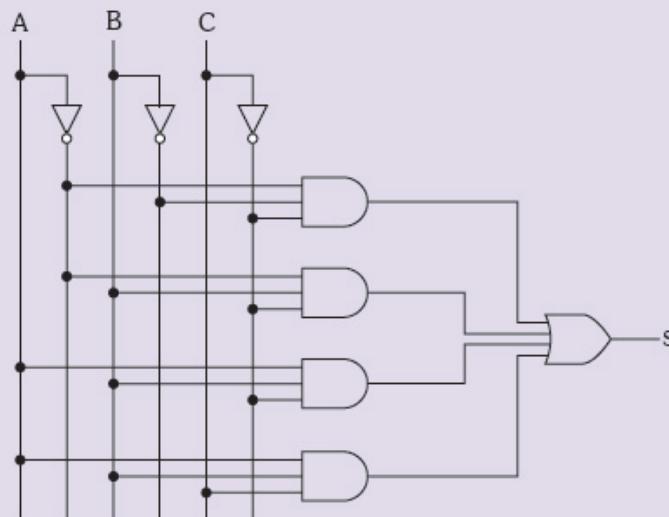


Figura 1.28 - Circuito obtido da tabela.

1.2) Obtenha a expressão booleana da Tabela 1.16.

Tabela 1.16 - Tabela-verdade do exercício 2

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

Da tabela extraímos os casos em que a expressão é verdadeira ( $S = 1$ ): 0100 ou 0110 ou 1000 ou 1011.

$$\therefore S = \overline{A}B\overline{C}\overline{D} + \overline{A}B C \overline{D} + A\overline{B}\overline{C}\overline{D} + A\overline{B} C D$$

## 1.11 Blocos lógicos OU Exclusivo e Coincidência

Estudaremos, neste tópico, os blocos OU Exclusivo e Coincidência que são muito importantes na área de Eletrônica Digital, juntamente com as outras portas lógicas formam outros circuitos elementares dentro dos sistemas digitais.

Embora estes circuitos sejam blocos lógicos básicos, podemos considerá-los também como circuitos combinacionais, pois sua obtenção provém de uma tabela-verdade (situação), que gera uma expressão característica, da qual esquematizamos o circuito.

### 1.11.1 Função e Porta Lógica OU Exclusivo ou Exclusive OR

A função que ele executa, como o próprio nome diz, consiste em fornecer 1 à saída quando as variáveis de entrada forem diferentes entre si. Com essa

pequena apresentação podemos montar sua tabela-verdade e obter pelo mesmo processo visto até aqui a expressão característica e, posteriormente esquematizar o circuito.

Tabela 1.17 - Tabela-verdade do bloco Ou Exclusivo

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

Da Tabela 1.17 obtemos sua expressão característica:

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

Da expressão esquematizamos o circuito representativo da função OU Exclusivo, visto na Figura 1.29.

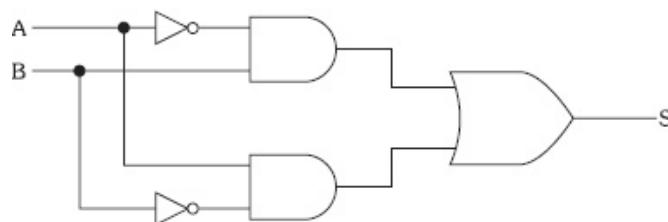


Figura 1.29 - Circuito da função OU Exclusivo.

A notação algébrica que representa a função OU Exclusivo é  $S = A \oplus B$ , onde se lê A OU Exclusivo B, sendo  $S = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$ . O circuito OU Exclusivo pode ser representado tanto pelo circuito da Figura 1.29 como pelo símbolo visto na Figura 1.30, sendo este mais simples.



Figura 1.30 - Símbolo do OU Exclusivo.

Uma importante observação é que, ao contrário de outros blocos lógicos básicos, o circuito OU Exclusivo só pode ter duas variáveis de entrada, fato devido à sua definição básica. O circuito OU Exclusivo também é conhecido como **Exclusive OR (EXOR)**.

## 1.11.2 Função e porta lógica Coincidência ou NOU Exclusivo ou Exclusive NOR

A função que ele executa, como seu próprio nome diz, é a de fornecer 1 à saída quando houver uma coincidência nos valores das variáveis de entrada. Vamos, agora, montar sua tabela-verdade:

Tabela 1.18 - Tabela-verdade do bloco Coincidência

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

A Tabela 1.18 gera a expressão  $S = \overline{A} \cdot \overline{B} + A \cdot B$ .

A partir desta expressão, vamos esquematizar o circuito mostrado na Figura 1.31.

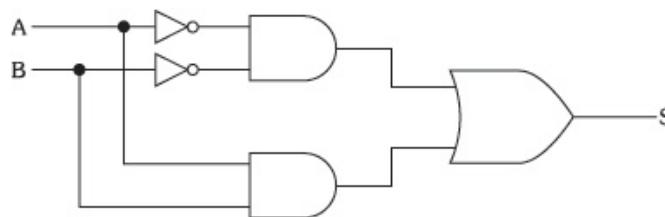


Figura 1.31 - Circuito da função Coincidência.

A notação algébrica que representa a função Coincidência é  $S = A \odot B$ , onde se lê: A Coincidência B, sendo  $S = A \odot B = \overline{A} \cdot \overline{B} + A \cdot B$ . O símbolo do circuito Coincidência é visto na Figura 1.32.



Figura 1.32 - Símbolo do Coincidência.

Se compararmos as tabelas-verdade dos blocos OU Exclusivo e Coincidência, vamos concluir que eles são complementares, ou seja, teremos a saída de um invertida em relação à saída do outro. Assim sendo, podemos escrever:

$$A \oplus B = \overline{A \odot B}$$

O bloco Coincidência é também denominado Exclusive NOR. Da mesma forma que o OU Exclusivo, o bloco Coincidência é definido apenas para duas variáveis de entrada.



## Exercícios resolvidos

- 1.1) A partir dos sinais aplicados às entradas da porta da Figura 1.33 desenhe a forma de onda na saída S.

Para a solução, devemos desenhar o sinal de saída, efetuando a operação OU Exclusivo entre os níveis, colocando  $S = 1$  nos casos  $A = 0$  e  $B = 1$  ou  $A = 1$  e  $B = 0$ . Assim sendo, temos a Figura 1.34.

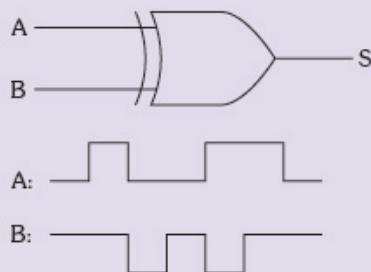


Figura 1.33 - Porta lógica e sinais aplicados.

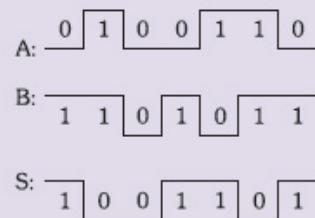


Figura 1.34 - Sinais de entrada e saída S resultante.

- 1.2) Determine a expressão e a tabela-verdade do circuito da Figura 1.35.

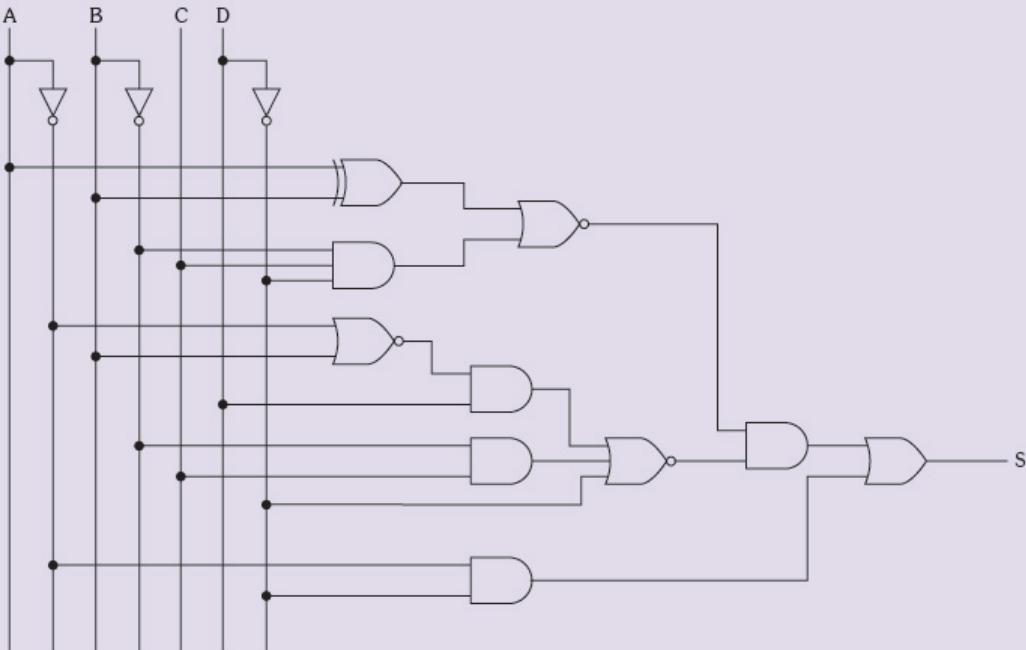


Figura 1.35 - Circuito do exercício 2.

Vamos primeiramente obter a expressão que o circuito executa:

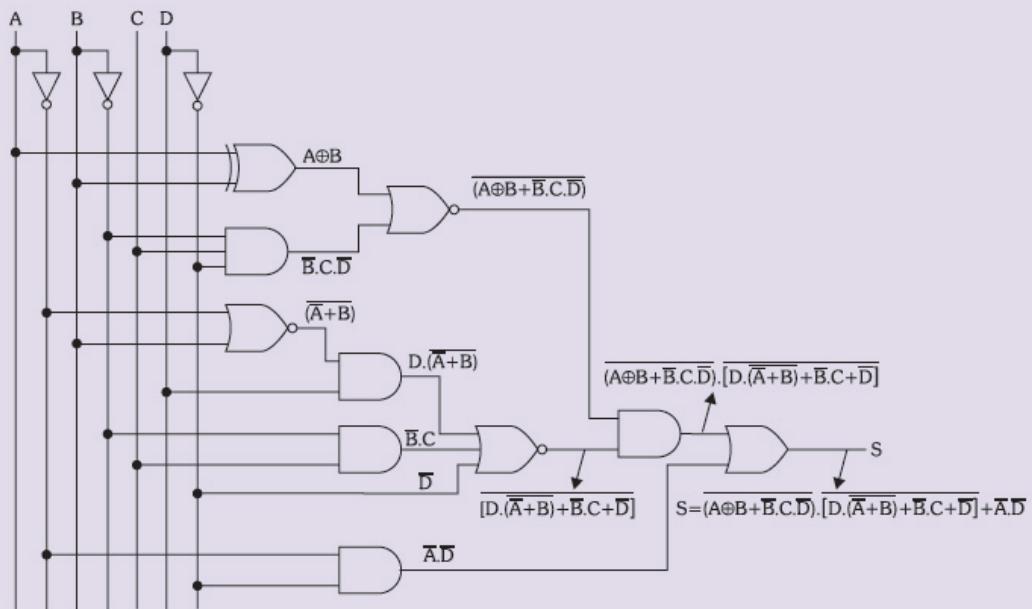


Figura 1.36 - Sinais de entrada e saída S resultante.

Logo após, a partir da expressão, vamos levantar sua tabela-verdade:

- 1) Nos casos em que  $A = D = 0$ , temos  $S = 1$ , o termo  $\bar{A} \cdot \bar{D}$  será igual a 1:  $S = \dots + 1 = 1$ .

- 2) Nos casos remanescentes em que  $A = 1$  e  $B = 0$  ou  $A = 0$  e  $B = 1$  A  $\oplus$   $B = 1$ , temos  $S = 0$ , o termo  $(A \oplus B + \overline{B}C\overline{D})$  será 0 e este multiplica o outro termo da expressão.
- 3) Nos casos remanescentes em que  $D = 0$ , temos  $S = 0$ , o termo  $[... + \overline{D}]$  da mesma forma será 0.
- 4) Para o caso restante, em que  $\overline{B} \cdot C = 1$  ( $B = 0$  e  $C = 1$ ), da mesma forma  $S = 0$ .
- 5) Para os casos restantes em que  $B = 1$ , temos  $S = 1$ , o parêntese  $(\overline{A} + B)$  será 0, sendo  $S = 1 \cdot [0 + 0 + 0] + 0 = 1$ .
- 6) No último caso  $S = 1$ , sendo  $A = 0$  valem as mesmas considerações do caso ⑤.

A Tabela 1.19 mostra a tabela obtida da expressão com os casos analisados assinalados.

Tabela 1.19 - Tabela-verdade obtido com os casos colocados

A	B	C	D	S	
0	0	0	0	1	①
0	0	0	1	1	⑥
0	0	1	0	1	①
0	0	1	1	0	④
0	1	0	0	1	①
0	1	0	1	0	
0	1	1	0	1	①
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	0	③
1	1	0	1	1	
1	1	1	0	0	③
1	1	1	1	1	⑤

### Vamos recapitular?

Neste capítulo, você conheceu as funções e portas lógicas básicas, as expressões booleanas características e tabelas-verdade de cada bloco.

Aprendeu também a obter expressões a partir de circuitos lógicos diversos, desenhar circuitos a partir das expressões, levantar tabelas-verdade das expressões e obter expressões a partir de tabelas-verdade.

Conheceu também os blocos OU Exclusivo e Coincidência, definidos apenas para duas variáveis de entrada, suas expressões e circuitos representativos, sendo inseridos em circuitos lógicos diversos para a realização das operações já citadas.



## Agora é com você!

- 1) De forma análoga aos circuitos das Figuras 1.1, 1.4 e 1.7, esquematize os circuitos representativos das funções NE e NOU.
- 2) Determine a expressão característica de cada circuito.
  - a)

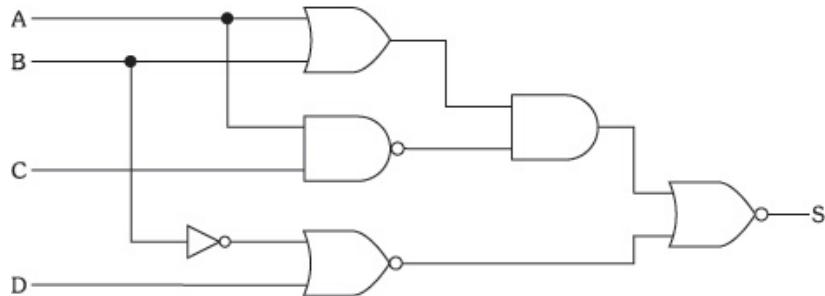


Figura 1.37 - Circuito lógico (a).

- 2) b)

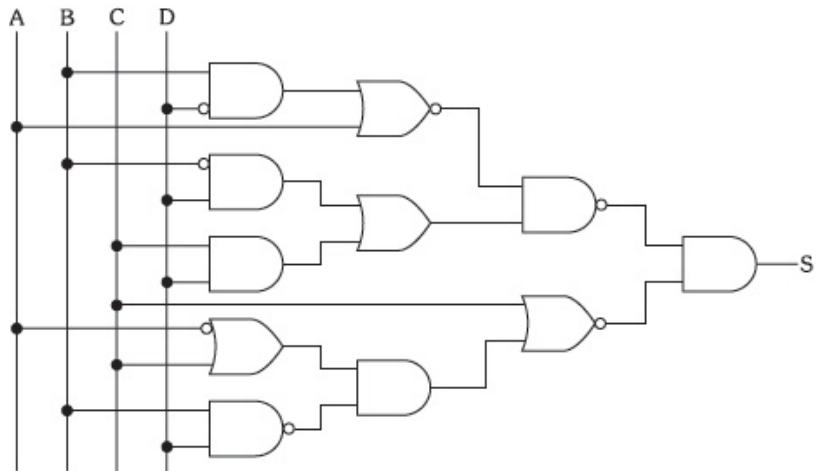


Figura 1.38 - Circuito lógico (b).

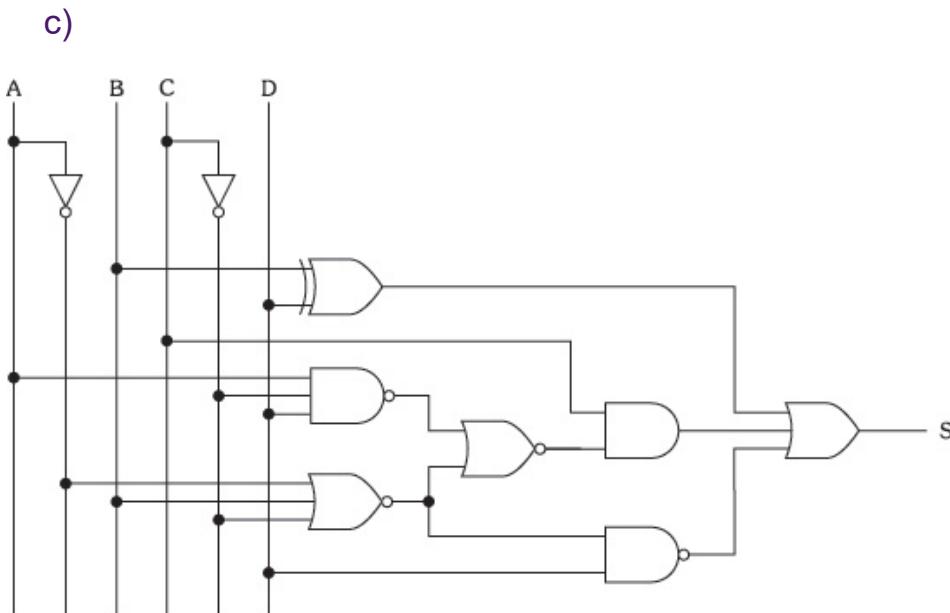


Figura 1.39 - Circuito lógico (c).

3) Desenhe o circuito que executa cada expressão:

a)  $S = \overline{A} \cdot [\overline{B} \cdot C + A \cdot (\overline{C} + \overline{D}) + B \cdot \overline{C} \cdot D] + B \cdot \overline{D}$

b)  $S = (A \odot B) \cdot [A \cdot \overline{B} + (\overline{B} + \overline{D}) + C \cdot \overline{D} + (\overline{B} \cdot C)] + \overline{A} \cdot B \cdot \overline{C} \cdot D$

4) Levante a tabela-verdade de cada expressão:

a)  $S = \overline{C} \cdot [A \cdot \overline{B} + B \cdot (\overline{A} + C)]$

b)  $S = (B \oplus D) \cdot [\overline{A} + \overline{B} \cdot (\overline{C} + \overline{D}) + A \cdot \overline{B} \cdot \overline{C}]$

- 5) Escreva a expressão característica do circuito mostrado na Figura 1.40 e levante a respectiva tabela-verdade.

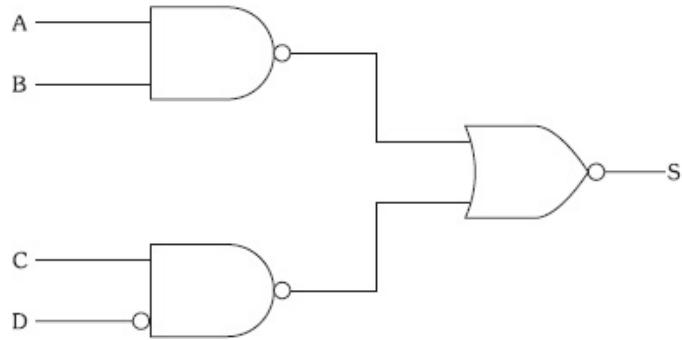


Figura 1.40 - Circuito do exercício 5.

- 6) Desenhe o circuito a partir da expressão e levante sua tabela-verdade:

$$S = [(\overline{\overline{B}} + \overline{C} + \overline{D}) \cdot (\overline{A} + B + C) + C] + A \cdot \overline{B} \cdot C + \overline{B} \cdot \overline{(A + C)}$$

- 7) Determine a expressão booleana a partir da Tabela 1.20.

Tabela 1.20 - Tabela do exercício 7

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

- 8) Desenhe o circuito que executa a Tabela 1.21.

Tabela 1.21 - Tabela do exercício 8

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

9) Desenhe o sinal na saída S do circuito da Figura 1.41.

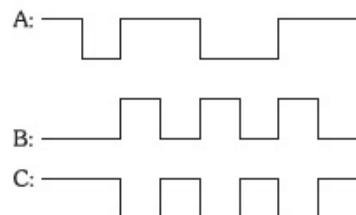
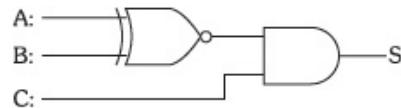


Figura 1.41 - Circuito e formas de onda do exercício 9.

10) Mostre que o circuito a seguir é um OU Exclusivo.

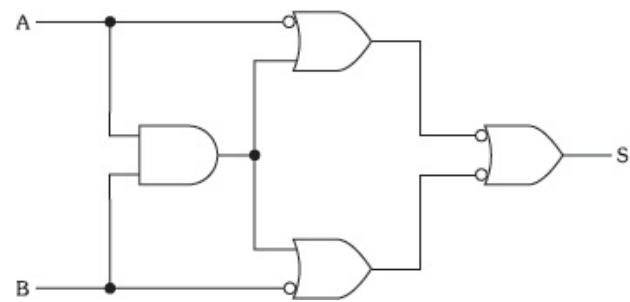


Figura 1.42 - Circuito do exercício 10.

- 11) Mostre que o circuito a seguir é um circuito Coincidência.

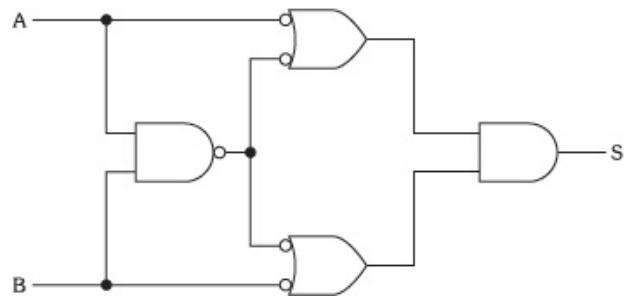


Figura 1.43 - Circuito do exercício 11.

- 12) Levante a tabela-verdade e esquematize o circuito que executa a expressão:

$$S = \{[A \cdot B + C] \oplus [A + B]\} \odot C$$

# 2

## Álgebra de Boole e Simplificação de Circuitos Lógicos

### Para começar

Este capítulo tem por objetivo apresentar os conceitos referentes à álgebra de Boole, que dá origem aos sistemas digitais elementares estudados no Capítulo 1. Além disso, por meio de manipulação algébrica permite a simplificação de expressões e circuitos lógicos derivados.

Outra forma de simplificação mais usual, também será estudada, se trata do uso dos mapas de 'Veitch-Karnaugh' ou simplesmente 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.

### 2.1 Simplificação de circuitos lógicos utilizando a álgebra de Boole

Vimos, até aqui, que os circuitos lógicos executam expressões. No Capítulo 1, determinamos esses circuitos através de expressões características extraídas de tabelas-verdade. Os circuitos gerados por esse processo, apesar de corretos, admitem geralmente simplificações, e consequente diminuição de blocos lógicos utilizados na prática.

Para entrarmos no estudo da simplificação dos circuitos lógicos, é preciso fazer uma breve pesquisa da álgebra de Boole, pois, através de seus postulados,

propriedades, teoremas fundamentais e identidades é que efetuaremos as mencionadas simplificações. Além disso, notamos que é na álgebra de Boole que estão todos os fundamentos dos sistemas digitais.

## 2.2 Variáveis e expressões na álgebra de Boole

---

As variáveis booleanas são representadas por letras, podendo assumir apenas dois valores distintos: 0 ou 1. Denominamos expressão booleana a sentença matemática composta de termos cujas variáveis são booleanas, da mesma forma, podendo assumir como resultado final 0 ou 1.

## 2.3 Postulados

---

A seguir, apresentamos os postulados da complementação, adição e multiplicação da álgebra de Boole e as respectivas identidades resultantes.

### 2.3.1 Postulados da complementação

Este postulado mostra como são as regras da complementação na álgebra de Boole. Chamaremos de  $\bar{A}$  o complemento de A:

$$\text{Se } A = 0 \rightarrow \bar{A} = 1 \text{ e se } A = 1 \rightarrow \bar{A} = 0$$

Por meio do postulado da complementação podemos estabelecer a seguinte identidade:

$$\overline{\bar{A}} = A$$

Se  $A = 1$ , temos  $\bar{A} = 0$  e se  $\bar{A} = 0 \rightarrow \overline{\bar{A}} = 1$ .

Se  $A = 0$ , temos  $\bar{A} = 1$  e se  $\bar{A} = 1 \rightarrow \overline{\bar{A}} = 0$ .

Assim sendo, podemos escrever  $\overline{\bar{A}} = A$ .

O bloco lógico que executa o postulado da complementação é o inversor.

### 2.3.2 Postulado da adição

Este postulado mostra como são as regras da adição na álgebra de Boole.

$$1^{\underline{o}}) 0 + 0 = 0$$

$$2^{\underline{o}}) 0 + 1 = 1$$

$$3^{\underline{o}}) 1 + 0 = 1$$

$$4^{\underline{o}}) 1 + 1 = 1$$

Através desse postulado, podemos estabelecer as seguintes identidades:

**A + 0 =** A pode ser 0 ou 1. Vejamos todas as possibilidades:

A.

$$A = 0 \rightarrow 0 + 0 = 0 \text{ e } A = 1 \rightarrow 1 + 0 = 1$$

Notamos que o resultado será sempre igual à variável A.

$A + 1 =$  Vejamos todas as possibilidades:

1.

$$A = 0 \rightarrow 0 + 1 = 1 \text{ e } A = 1 \rightarrow 1 + 1 = 1$$

Notamos que se somarmos 1 a uma variável, o resultado será sempre 1.

$A + A =$  Vejamos todas as possibilidades:

A.

$$A = 0 \rightarrow 0 + 0 = 0 \text{ e } A = 1 \rightarrow 1 + 1 = 1$$

Notamos que se somarmos a mesma variável, o resultado será ela mesma.

$A + \bar{A} =$  Vejamos todas as possibilidades:

1.

$$A = 0 \rightarrow \bar{A} = 1 \rightarrow 0 + 1 = 1 \text{ e } A = 1 \rightarrow \bar{A} = 0 \rightarrow 1 + 0 = 1$$

Notamos que sempre que somarmos a uma variável o seu complemento, teremos como resultado 1.

O bloco lógico que executa o postulado da adição é o OU.

### 2.3.3 Postulado da multiplicação

É o postulado que determina as regras da multiplicação booleana:

1º)  $0 \cdot 0 = 0$

2º)  $0 \cdot 1 = 0$

3º)  $1 \cdot 0 = 0$

4º)  $1 \cdot 1 = 1$

Com esse postulado podemos estabelecer as seguintes identidades:

$A \cdot 0$  Podemos confirmar, verificando todas as possibilidades:  
= 0.

$$A = 0 \rightarrow 0 \cdot 0 = 0 \text{ e } A = 1 \rightarrow 1 \cdot 0 = 0$$

Notamos que todo número multiplicado por 0 é 0.

$A \cdot 1$  Analisando todas as possibilidades, temos:  
= A.

$$A = 0 \rightarrow 0 \cdot 1 = 0 \text{ e } A = 1 \rightarrow 1 \cdot 1 = 1$$

Notamos que o resultado destas expressões numéricas é sempre igual a A.

$A \cdot A$  Esta identidade, à primeira vista estranha, é verdadeira, como podemos confirmar pela análise de todas as possibilidades:

$$A = 0 \rightarrow 0 \cdot 0 = 0 \text{ e } A = 1 \rightarrow 1 \cdot 1 = 1$$

Notamos que os resultados serão sempre iguais a A.

$A \cdot \bar{A}$  Vamos analisar todas as possibilidades:  
 $= 0$ .

$$A = 0 \rightarrow 0 \cdot 1 = 0 \text{ e } A = 1 \rightarrow 1 \cdot 0 = 0$$

Notamos que para os ambos valores possíveis que a variável pode assumir, o resultado da expressão será sempre 0.

O bloco lógico que executa o postulado da multiplicação é o E.

## 2.4 Propriedades

---

A seguir descrevemos as principais propriedades algébricas, úteis principalmente no manuseio e na simplificação de expressões. Tal como na matemática comum, vale na álgebra de Boole as propriedades comutativa, associativa e distributiva.

### 2.4.1 Propriedade comutativa

Esta propriedade é válida tanto na adição como na multiplicação:

Adição:  $A + B = B + A$

Multiplicação:  $A \cdot B = B \cdot A$

### 2.4.2 Propriedade associativa

Da mesma forma que na anterior, temos a propriedade associativa válida na adição e na multiplicação:

Adição:  $A + (B + C) = (A + B) + C = A + B + C$

Multiplicação:  $A \cdot (B \cdot C) = (A \cdot B) \cdot C = A \cdot B \cdot C$

### 2.4.3 Propriedade distributiva

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

Vamos verificar esta propriedade na tabela-verdade, analisando todas as possibilidades:

Tabela 2.1 - Comprovação da propriedade distributiva

A	B	C	$A(B+C)$	$AB + AC$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Notamos, pela Tabela 2.1, que as expressões equivalem-se.

## 2.5 Teoremas de De Morgan

Os teoremas de De Morgan são muito empregados na prática, em simplificações de expressões booleanas e ainda no desenvolvimento de circuitos digitais, como veremos em tópicos posteriores.

### 2.5.1 1º Teorema de De Morgan

O complemento do produto é igual à soma dos complementos:  $(\overline{A \cdot B}) = \overline{A} + \overline{B}$

Para provar este teorema, vamos montar a tabela-verdade de cada membro e comparar os resultados, Tabela 2.2.

Tabela 2.2 - Comprovação do 1º Teorema de De Morgan

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

Notamos a igualdade de ambas as colunas. O teorema pode ser estendido para mais de duas variáveis:

$$(\overline{A \cdot B \cdot C \dots N}) = \overline{A} + \overline{B} + \overline{C} + \dots + \overline{N}$$

### 2.5.2 2º Teorema de De Morgan

O complemento da soma é igual ao produto dos complementos.

Este teorema é uma extensão do primeiro:

$$(\overline{A + B}) = \overline{A} \cdot \overline{B} \leftarrow 2º \text{ Teorema}$$

Podemos reescrevê-lo da seguinte maneira:  $A \cdot B = (\overline{\overline{A} + \overline{B}})$

Notamos que  $A$  é o complemento de  $\bar{A}$  e que  $B$  é o complemento de  $\bar{B}$ . Vamos chamar  $\bar{A}$  de  $X$  e  $\bar{B}$  de  $Y$ . Assim sendo, temos:

$$\bar{X} \cdot \bar{Y} = (\bar{X} + \bar{Y})$$

Reescrevendo, em termos de  $A$  e  $B$ , temos:

$$\bar{A} \cdot \bar{B} = (\bar{A} + \bar{B}) \leftarrow 2^{\text{o}} \text{ Teorema}$$

Da mesma forma que no anterior, o teorema pode ser estendido para mais de duas variáveis:

$$(\bar{A} + \bar{B} + \bar{C} + \dots + \bar{N}) = \bar{A} \cdot \bar{B} \cdot \bar{C} \dots \bar{N}$$

## 2.6 Identidades auxiliares

---

A seguir, vamos deduzir três identidades úteis para a simplificação de expressões:

**1º)  $A + A \cdot B = A$**

Provamos esta identidade utilizando a propriedade distributiva. Vamos evidenciar  $A$  no 1º termo:

$$A(1 + B) = A$$

Do postulado da soma temos  $1 + B = 1$ ; logo, podemos escrever:  $A \cdot 1 = A$  ∴  $A + AB = A$

**2º)  $(A + B) \cdot (A + C) = A + B \cdot C$**

Vamos agora provar esta identidade:

$$(A + B) \cdot (A + C)$$

$$= A \cdot A + A \cdot C + A \cdot B + B \cdot C \quad \rightarrow \text{Propriedade distributiva}$$

$$= A + A \cdot C + A \cdot B + B \cdot C \quad \rightarrow \text{Identidade } A \cdot A = A$$

$$= A \cdot (1 + B + C) + B \cdot C \quad \rightarrow \text{Propriedade distributiva}$$

$$= A \cdot 1 + B \cdot C \quad \rightarrow \text{Identidades: } 1 + X = 1 \text{ e } A \cdot 1 = A$$

$$\therefore (A + B) \cdot (A + C) = A + BC$$

**3º)  $A + \bar{A} \cdot B = A + B$**

Vamos provar esta identidade:

$$\begin{aligned}
A + \overline{A} \cdot B &= \overline{(A + \overline{A} \cdot B)} && \rightarrow \text{Identidade } \overline{\overline{X}} = X \\
&= \overline{[\overline{A} \cdot (\overline{A} \cdot B)]} && \rightarrow 2^{\text{o}} \text{ Teorema de De Morgan: } \overline{(X + Y)} = \overline{X} \cdot \overline{Y} \\
&= \overline{[\overline{A} \cdot (A + \overline{B})]} && \rightarrow 1^{\text{o}} \text{ Teorema de De Morgan aplicado no parênteses:} \\
&&& \quad \overline{X \cdot Y} = (\overline{X} + \overline{Y}) \\
&= (\overline{A} \cdot A + \overline{A} \cdot \overline{B}) && \rightarrow \text{propriedade distributiva e identidade } \overline{A} \cdot A = 0 \\
&= (\overline{A} \cdot \overline{B}) \\
&= (A + B) && \rightarrow 1^{\text{o}} \text{ Teorema de De Morgan} \\
\therefore (A + \overline{A} \cdot B) &= A + B
\end{aligned}$$

## 2.7 Simplificação de expressões utilizando a álgebra de Boole

---

Para exemplificar uma simplificação, vamos utilizar a expressão:

$$S = ABC + A\overline{C} + A\overline{B}$$

Vamos simplificá-la utilizando a álgebra de Boole. Primeiramente é preciso evidenciar o termo A:

$$S = A(BC + \overline{C} + \overline{B})$$

Agora, aplicando a propriedade associativa, temos:

$$S = A [BC + (\overline{C} + \overline{B})]$$

Aplicando a identidade  $\overline{\overline{X}} = X$ , temos:

$$S = A [BC + (\overline{\overline{C}} + \overline{\overline{B}})]$$

Aplicando o Teorema de De Morgan, temos:

$$S = [BC + (\overline{B}\overline{C})] \cdot A$$

Chamando BC de Y; logo  $(\overline{B}\overline{C}) = \overline{Y}$ . Temos:

$$S = A(Y + \overline{Y})$$

Como  $Y + \overline{Y} = 1$ , logo  $S = A \cdot 1 = A \therefore S = A$

Esta expressão mostra a importância da simplificação e a consequente minimização do circuito, pois os resultados são idênticos aos valores assumidos pela variável A. Assim sendo, todo o circuito pode ser substituído por um único fio ligado à variável A.

Como outro exemplo, vamos simplificar a expressão:

$$S = \overline{A} \overline{B} \overline{C} + \overline{ABC} + A\overline{B}C$$

Tirando  $\overline{A} \cdot \overline{C}$  em evidência nos dois primeiros termos, temos:

$$S = \overline{A} \cdot \overline{C} \cdot (\overline{B} + B) + A\overline{B}C$$

Aplicando a identidade  $B + \overline{B} = 1$ , temos:

$$S = \overline{A} \cdot \overline{C} \cdot (\overline{B} + B) + A\overline{B}C = \overline{A} \cdot \overline{C} + A\overline{B}C \therefore S = \overline{A} \overline{C} + A\overline{B}C$$



## Exercícios resolvidos

2.1) Simplifique as expressões booleanas apresentadas a seguir:

a)  $S = \overline{A} \overline{B} \overline{C} + \overline{ABC} + \overline{AB}\overline{C} + A \overline{B} \overline{C} + A\overline{B}\overline{C}$

Evidenciando  $\overline{C}$ , temos:

$$S = \overline{ABC} + \overline{C}(\overline{A} \overline{B} + \overline{AB} + A\overline{B} + AB)$$

Evidenciando  $\overline{A}$  e  $A$ , temos:

$$S = \overline{ABC} + \overline{C}[\overline{A}(\overline{B} + B) + A(\overline{B} + B)]$$

$$S = \overline{ABC} + \overline{C}(\overline{A} \cdot 1 + A \cdot 1) \rightarrow \text{identidade } \overline{X} + X = 1$$

$$S = \overline{ABC} + \overline{C}(\overline{A} + A)$$

$$S = \overline{ABC} + \overline{C} \cdot 1 \rightarrow \text{identidade } \overline{X} + X = 1$$

$$S = (\overline{\overline{ABC}} + \overline{\overline{C}}) \rightarrow \text{identidade } \overline{\overline{X}} = X$$

$$S = \overline{[(\overline{ABC}).C]} \rightarrow \text{Teorema de De Morgan: } (X + Y) = \overline{X} \cdot \overline{Y}$$

$$S = \overline{[(A + \overline{B} + \overline{C}).\overline{C}]} \rightarrow \text{Teorema de De Morgan: } (\overline{X} \cdot \overline{Y} \cdot \overline{Z}) = \overline{X} + \overline{Y} + \overline{Z}$$

$$S = \overline{(AC + \overline{B}C + \overline{C}.C)} \rightarrow \text{propriedade distributiva}$$

$$S = \overline{[C.(A + \overline{B})]} \rightarrow \text{propriedade distributiva e identidade } X \cdot \overline{X} = 0$$

$$S = \overline{[\overline{C} + (A + \overline{B})]} \rightarrow \text{Teorema de De Morgan: } (\overline{X} \cdot \overline{Y}) = \overline{X} + \overline{Y}$$

$$S = (\overline{C} + \overline{A} \cdot B) \rightarrow \text{Teorema de De Morgan: } (\overline{X} + \overline{Y}) = \overline{X} \cdot \overline{Y}$$

$$\therefore S = \overline{C} + \overline{A} \cdot B$$

b)  $S = (A + B + C) \cdot (\overline{A} + \overline{B} + C)$

Aplicando a propriedade distributiva, temos:

$$S = A\overline{A} + A\overline{B} + AC + \overline{A}B + B\overline{B} + BC + \overline{A}C + BC + CC$$

Vamos usar as identidades  $X \cdot \bar{X} = 0$  e  $X \cdot X = X$  e reescrever:

$$S = A\bar{B} + AC + \bar{A}\bar{B} + BC + \bar{A}C + \bar{B}C + C$$

Colocando C em evidência, temos:

$$S = A\bar{B} + C(A + B + \bar{A} + \bar{B} + 1) + \bar{A}B$$

Usando as identidades  $X + 1 = 1$  e  $X \cdot 1 = X$ , obtemos o resultado final:

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

c)  $S = [(\overline{AC}) + B + D] + C(\overline{ACD})$

Aplicando o Teorema de De Morgan ao 1º e 2º termos, obtemos:

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

Agora, aplicando o Teorema de De Morgan ao 1º termo e a propriedade distributiva ao 2º termo, temos:

$$S = ACD\bar{B} + \bar{A}C + CC\bar{C} + C\bar{D}$$

Reescrevendo, aplicando a identidade  $X \cdot \bar{X} = 0$ , temos:

$$S = A\bar{B}C\bar{D} + \bar{A}C + C\bar{D}$$

Evidenciando o termo  $C\bar{D}$ , vamos ter:

$$S = \bar{C}\bar{D}(A\bar{B} + 1) + \bar{A}C$$

$$S = \bar{C}\bar{D} \cdot 1 + \bar{A}C \rightarrow \text{identidade } X + 1 = 1$$

$$\therefore S = \bar{C}\bar{D} + \bar{A}C$$

2.2) A partir da expressão ) obtenha  $S = \overline{(A \odot B)}$  obtenha  $S = A \oplus B$ .

O primeiro passo é substituir a expressão do circuito coincidência pela sua equivalente:

$$S = \overline{(A \odot B)}$$

$$S = \overline{(\bar{A}\bar{B} + AB)}$$

Aplicando Teorema De Morgan  $(\bar{X} + \bar{Y}) = \bar{X} \cdot \bar{Y}$ , temos:

$$S = (\bar{\bar{A}}\bar{\bar{B}}) \cdot (\bar{AB})$$

Aplicando o outro Teorema,  $(\bar{X} \cdot \bar{Y}) = \bar{X} + \bar{Y}$ , em cada parênteses, temos:

$$S = (A + B) \cdot (\bar{A} + \bar{B})$$

Aplicando a propriedade distributiva, temos:

$$S = A\bar{A} + \bar{A}B + A\bar{B} + \bar{B}B$$

Como  $A\bar{A} = 0$  e  $\bar{B}B = 0$ , temos:

$$S = \bar{A}B + A\bar{B} \Rightarrow S = A \oplus B$$

2.3) Obtenha o circuito simplificado que executa a expressão:

$$S = (A \oplus B) [ \overline{B(A + \bar{C})} + \bar{D}(\bar{A} + B + \bar{C}) ]$$

Aplicando a propriedade distributiva e De Morgan respectivamente aos termos do colchete, obtém-se:

$$S = (A \oplus B)(AB + B\bar{C} + \bar{D}A\bar{B}\bar{C})$$

Reescrevendo o último termo, em ordem, surge:

$$S = (A \oplus B)(AB + B\bar{C} + A\bar{B}\bar{C}\bar{D})$$

Aplicando De Morgan ao 2º parêntese, temos:

$$S = (A \oplus B)(\bar{A}B)(\bar{B}\bar{C})(\bar{A}\bar{B}\bar{C}\bar{D})$$

Aplicando novamente em cada parêntese e substituindo o 1º pela expressão equivalente do OU Exclusivo, obtemos:

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

Efetuando a multiplicação entre os dois primeiros parênteses, eliminando os termos resultantes, em que  $\bar{A} \cdot A = 0$  e  $\bar{B} \cdot B = 0$ , obtemos:

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

Como  $X \cdot X = X$ , temos:

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

Da mesma forma, efetuando a multiplicação entre os dois últimos, obtemos:

$$S = (\bar{A}B + A\bar{B})(\bar{A}\bar{B} + \bar{B}\bar{C} + \bar{B}D + \bar{A}C + BC + CD)$$

Novamente multiplicando, temos:

$$S = \bar{A}BC + \bar{A}BC + \bar{A}BCD + \bar{A}\bar{B}\bar{C} + A\bar{B}D + A\bar{B}CD$$

Tirando em evidência  $\bar{A}BC$  para os três primeiros termos e  $A\bar{B}$  para os últimos, temos:

$$S = \bar{A}BC(1 + 1 + D) + A\bar{B}(\bar{C} + D + CD)$$

Fazendo  $(1 + D) = 1$  e colocando em evidência D no 2º parêntese, temos:

$$S = \bar{A}BC + A\bar{B}[\bar{C} + D(1 + C)]$$

Como  $1 + C = 1$ , temos:

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

$$\therefore S = \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}D$$

A partir da expressão, desenhamos o circuito simplificado visto na Figura 2.1.

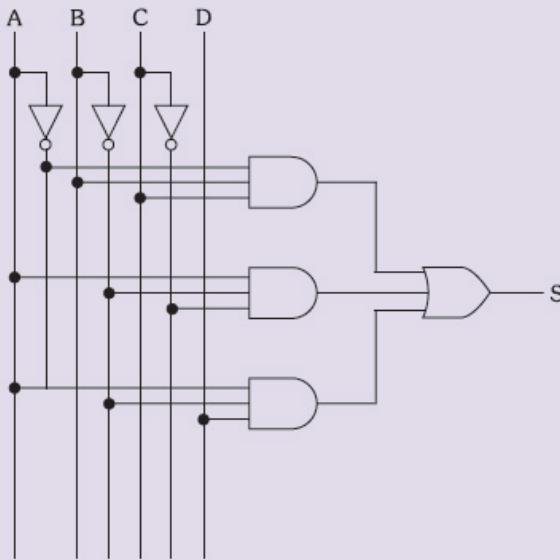


Figura 2.1 - Circuito simplificado obtido.

## 2.8 Simplificação de expressões utilizando os mapas de Karnaugh

Vimos a simplificação de expressões mediante a utilização da álgebra de Boole. Nestes itens, vamos tratar da simplificação de expressões por meio dos mapas ou diagramas de Karnaugh.

Estes mapas permitem a simplificação de maneira mais rápida dos casos extraídos de tabelas-verdade, obtidas de situações quaisquer. Serão estudados os mapas para duas, três e quatro variáveis.

### 2.8.1 Mapa de Karnaugh para duas variáveis

A Figura 2.2 apresenta o mapa de Karnaugh para duas variáveis:

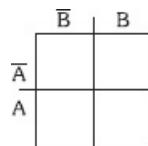
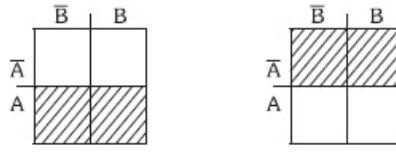
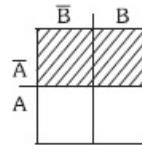


Figura 2.2 - Mapa de Karnaugh para duas variáveis.

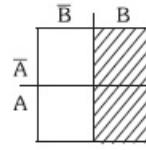
No mapa encontramos todas as possibilidades assumidas entre as variáveis A e B. A Figura 2.3 mostra todas as regiões do mapa.



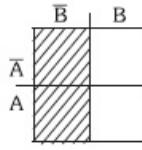
(a)



(b)



(c)



(d)

- (a) região onde  $A = 1$ .    (b) região onde  $A = 0 (\bar{A} = 1)$ .  
 (c) região onde  $B = 1$ .    (d) região onde  $B = 0 (\bar{B} = 1)$ .

Figura 2.3 - Regiões do mapa de Karnaugh.

Com duas variáveis podemos obter quatro possibilidades:

Tabela 2.3 - Tabela para colocação no mapa

A	B	
0	0	caso 0
0	1	caso 1
1	0	caso 2
1	1	caso 3

No caso 0, temos  $A = 0$  e  $B = 0$ . A região do mapa que mostra esta condição é a da intersecção das regiões em que  $A = 0$  e  $B = 0$ :

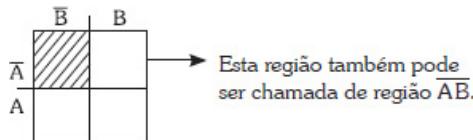


Figura 2.4 - Região do caso 0 no mapa de duas variáveis.

No caso 1, temos  $A = 0$  e  $B = 1$ . A região do mapa que mostra esta condição é a da intersecção das regiões em que  $A = 0 (\bar{A} = 1)$  e  $B = 1$ :



Figura 2.5 - Região do caso 1 no mapa de duas variáveis.

No caso 2 temos a intersecção das regiões em que  $A = 1$  e  $B = 0 (\bar{B} = 1)$ . Fazendo esta intersecção, temos:



Figura 2.6 - Região do caso 2 no mapa de duas variáveis.

No caso 3, temos a intersecção das regiões em que  $A = 1$  e  $B = 1$ . Fazendo esta intersecção, temos:

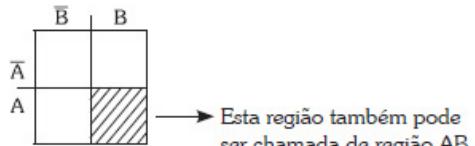


Figura 2.7 - Região do caso 3 no mapa de duas variáveis.

Podemos distribuir as quatro possibilidades neste mapa da seguinte forma:

	$\bar{B}$	B
$\bar{A}$	Caso 0 A B 0 0	Caso 1 A B 0 1
A	Caso 2 A B 1 0	Caso 3 A B 1 1

Figura 2.8 - Mapa de duas variáveis com todos os casos colocados.

Notamos que cada linha da tabela-verdade possui sua região própria no mapa.

Essas regiões são, portanto, os locais onde devem ser colocados os valores que a expressão assume nas diferentes possibilidades.

Para entendermos melhor o significado destes conceitos, vamos utilizar os exemplos:

- 1) A tabela-verdade mostra o estudo de uma função de duas variáveis. Vamos colocar seus resultados no mapa de Karnaugh.

Tabela 2.4 - Tabela com exemplo para colocação no mapa

A	B	S	
0	0	0	caso 0
0	1	1	caso 1
1	0	1	caso 2
1	1	1	caso 3

Utilizando o método desenvolvido anteriormente, obtemos a expressão característica da função:

$$S = \bar{A}\bar{B} + A\bar{B} + AB$$

Passando para o mapa os casos da tabela-verdade, conforme o esquema de colocação da Figura 2.9, obtém-se:

	$\bar{B}$	B
$\bar{A}$	0	1
A	1	1

Figura 2.9 - Mapa com os casos colocados.

Entendida a colocação dos valores assumidos pela expressão em cada caso no mapa de Karnaugh, vamos verificar como podemos efetuar as simplificações.

Para obter a expressão simplificada do mapa, utilizamos o seguinte método:

Tentamos agrupar as regiões onde S é igual a 1, no menor número possível de agrupamentos. As regiões onde S é 1, que não puderem ser agrupadas, serão consideradas isoladamente. Para um mapa de duas variáveis, os agrupamentos possíveis são os seguintes:

a) Quadra:

Conjunto de quatro regiões, onde S é igual a 1. No mapa de duas variáveis, é o agrupamento máximo proveniente de uma tabela onde todos os casos valem 1. Assim sendo, a expressão final simplificada obtida é S = 1. A Figura 2.10 ilustra esta situação:

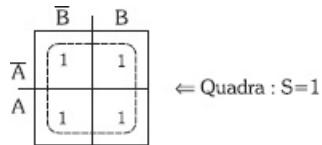


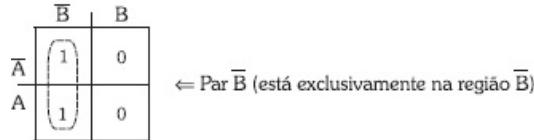
Figura 2.10 - Agrupamento formando uma quadra

b) Pares:

Conjunto de duas regiões onde S é 1, as quais têm um lado em comum, ou seja, são vizinhas. A Figura 2.11 mostra exemplos de dois pares agrupados e suas respectivas expressões, dentre os quatro possíveis em duas variáveis:



(a)



(b)

Figura 2.11 - Mapas (a) e (b) demonstrando exemplos de agrupamentos de pares.

c) Termos isolados:

Regiões onde S é 1, sem vizinhança para agrupamentos. São os próprios casos de entrada, sem simplificação. A Figura 2.12 exemplifica dois termos isolados, sem possibilidades de agrupamento.

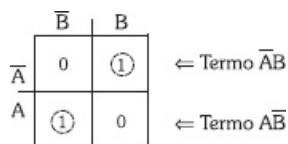


Figura 2.12 - Mapa demonstrando exemplos de termos isolados.

Para o exemplo da Tabela 2.4, efetuando os agrupamentos, temos a Figura 2.13.

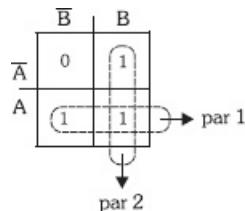


Figura 2.13 - Casos da Tabela 2.4 colocados no mapa e agrupamentos.

Feito isto, escrevemos a expressão de cada par, ou seja, a região que o par ocupa no mapa.

O par 1 ocupa a região onde A é igual a 1, então sua expressão será Par 1 = A.

O par 2 ocupa a região onde B é igual a 1, então sua expressão será Par 2 = B.

Notamos ainda que nenhum 1 ficou fora dos agrupamentos e que o mesmo 1 pode pertencer a mais de um agrupamento.

Para obter a expressão simplificada, basta somarmos os termos obtidos nos agrupamentos:

$$S = \text{Par 1} + \text{Par 2} \therefore S = A + B$$

Como podemos notar, esta é a expressão de uma porta OU, pois a tabela-verdade também é a da porta OU. Outro fato a ser notado é que a expressão obtida é visivelmente menor do que a extraída diretamente da tabela-verdade, acarretando um circuito mais simples, diminuindo, consequentemente, a dificuldade de montagem e o custo do sistema.

- 2) Vamos simplificar o circuito que executa a tabela-verdade a seguir:

Tabela 2.5 - Tabela com exemplo para colocação no mapa

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

Obtendo a expressão diretamente da tabela, temos:  $S = \bar{A}\bar{B} + \bar{A}B + A\bar{B}$

Transportando a tabela para o mapa, mediante processo já visto, temos:

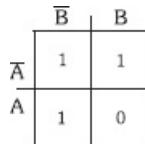


Figura 2.14 - Casos da Tabela 2.5 colocados no mapa.

Agora, vamos agrupar os pares:

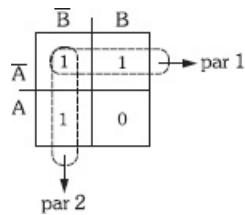


Figura 2.15 - Casos da Tabela 2.5 colocados no mapa e agrupados.

Vamos escrever as expressões dos pares:

$$\text{par 1} \rightarrow \bar{A} \quad \text{e} \quad \text{par 2} \rightarrow \bar{B}$$

Somando as expressões dos pares, temos a expressão simplificada:

$$S = \bar{A} + \bar{B}$$

Notamos que a tabela-verdade é a de uma porta NE. Aplicando o teorema de De Morgan à expressão, após a simplificação, encontramos a expressão de uma porta NE:  $S = \overline{AB}$

## 2.8.2 Mapas de Karnaugh para três variáveis

O mapa de Karnaugh para três variáveis é apresentado na Figura 2.16.

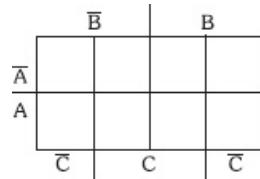


Figura 2.16 - Mapa de Karnaugh para três variáveis.

No mapa encontramos todas as possibilidades assumidas entre as variáveis A, B e C. A Figura 2.17 mostra as regiões deste mapa.

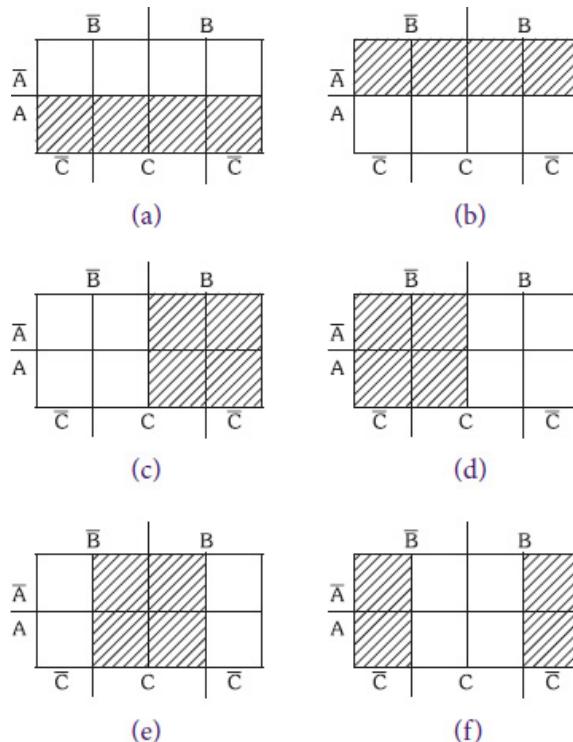


Figura 2.17 - Regiões do mapa de Karnaugh de três variáveis.

Neste mapa também há uma região para cada caso da tabela-verdade. A Tabela 2.6 e a Figura 2.18 mostram casos para três variáveis e as respectivas localizações no mapa.

Tabela 2.6 - Tabela-verdade de três variáveis

Caso	A	B	C
0	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

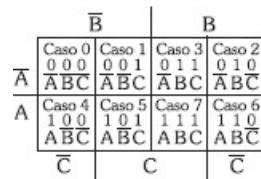


Figura 2.18 - Localização dos casos de três variáveis no mapa de Karnaugh.

Vamos analisar a localização somente de uma das possibilidades, visto que as outras são de maneira análoga. Assim sendo, vamos localizar no mapa o caso 3:

Caso	A	B	C
3	0	1	1

No mapa, será a intersecção das regiões que  $A = 0$  ( $\bar{A} = 1$ ),  $B = 1$  e  $C = 1$ . Esta pode ser chamada de região  $\bar{A}BC$ . A Figura 2.19 mostra esta localização no mapa, para a colocação do respectivo caso de entrada da coluna S.

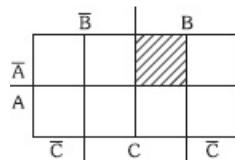


Figura 2.19 - Colocação do caso 3 no mapa de Karnaugh de três variáveis.

Para melhor compreensão, como exemplo, vamos transpor para o mapa as situações de saída da Tabela 2.7.

Tabela 2.7 - Tabela exemplo com as situações de saída para colocação em um mapa de Karnaugh

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

Expressão extraída da tabela-verdade:

$$S = \overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC} + ABC$$

Transpondo a tabela para o mapa, temos:

		$\overline{B}$		B		
		Caso 0	Caso 1	Caso 3	Caso 2	
$\overline{A}$	1	0	1	1		
	A	1	0	0	1	
		$\overline{C}$	C	C	$\overline{C}$	

Figura 2.20 - Mapa com os casos do exemplo colocados.

Para efetuarmos a simplificação, seguimos o mesmo processo visto anteriormente, somente que, para três variáveis, os agrupamentos possíveis são os seguintes:

a) Oitava:

Agrupamento máximo, em que todas as localidades valem 1. A Figura 2.21 apresenta esta situação.

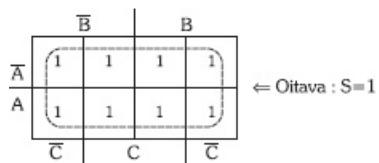


Figura 2.21 - Oitava: agrupamento máximo em um mapa de Karnaugh de três variáveis.

b) Quadras:

Quadras são agrupamentos de quatro regiões, sendo S igual a 1, adjacentes ou em sequência. Vamos agora formar algumas quadras possíveis em um mapa de três variáveis, a título de exemplo:

	$\bar{B}$	B	
$\bar{A}$	1 1 1 1	1 1 1 1	1 1 1 1
A	0 0 0 0	0 0 0 0	0 0 0 0
	$\bar{C}$	C	$\bar{C}$

(a)

	$\bar{B}$	B	
$\bar{A}$	1 1	0 0	0 0
A	1 1	0 0	0 0
	$\bar{C}$	C	$\bar{C}$

(b)

	$\bar{B}$	B	
$\bar{A}$	1 0 0 1	0 0 0 1	0 0 1 1
A	1 0 0 1	0 0 0 1	0 0 1 1
	$\bar{C}$	C	$\bar{C}$

(c)

Figura 2.22 - (a) Quadra  $\bar{A}$ ; (b) Quadra  $\bar{B}$ ; (c) Quadra  $\bar{C}$ .

c) Pares:

A Figura 2.23 apresenta como exemplo, dois pares entre os 12 possíveis em um mapa de três variáveis:

	$\bar{B}$	B	
$\bar{A}$	1 0 0 1	0 0 0 1	0 0 1 1
A	0 1 1 0	1 1 1 0	1 0 0 0
	$\bar{C}$	C	$\bar{C}$

↑

Par  $\bar{AC}$  (está localizado na intersecção das regiões  $\bar{A}$  e  $\bar{C}$ )

Par  $AC$  (está localizado na intersecção das regiões A e C)

Figura 2.23 - Exemplos de agrupamentos de pares.

d) Termos isolados:

Vejamos na Figura 2.24 alguns exemplos de termos isolados, que, como já dissemos, são os casos de entrada sem simplificação.

	$\bar{B}$		B
$\bar{A}$	0	(1)	0
A	0	0	(1)
$\bar{C}$	C	C	$\bar{C}$

↑  
Termo  $\bar{ABC}$

Figura 2.24 - Mapa demonstrando exemplos de termos isolados.

Para o exemplo da Tabela 2.7, agrupamos primeiramente uma quadra e, logo após, um par, conforme mostra a Figura 2.25.

	$\bar{B}$		B
$\bar{A}$	1	0	(1)
A	1	0	(1)
$\bar{C}$	C	C	$\bar{C}$

← Par  $\bar{AB}$   
← Quadra  $\bar{C}$

Figura 2.25 - Casos da Tabela 2.7 colocados no mapa e agrupados.

Notamos que o par não depende de C, pois está localizado tanto em C como em  $\bar{C}$ , sendo sua expressão independente de C, ou seja, resultando no termo  $\bar{A}B$ .

O passo final é somarmos as expressões referentes aos agrupamentos. A expressão final minimizada será  $S = \bar{A}B + \bar{C}$ .

Como outro exemplo, vamos minimizar o circuito que executa a Tabela 2.8.

Tabela 2.8 - Tabela exemplo

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	1
1	1	0	1
1	1	1	0

Transpondo para o mapa, temos:

	$\bar{B}$		B
$\bar{A}$	0	1	1
A	1	1	0
$\bar{C}$	C	C	$\bar{C}$

Figura 2.26 - Casos da Tabela 2.8 colocados no mapa.

Efetuando os agrupamentos, notamos que obtemos apenas três pares:

		$\bar{B}$	B
$\bar{A}$	0	(1)	(1)
A	(1)	(1)	0
$\bar{C}$	C	C	$\bar{C}$

$\Leftarrow$  Par  $\bar{A}C$

$\Leftarrow$  Pares:  $A\bar{C}$  e  $A\bar{B}$

Figura 2.27 - Agrupamentos.

A expressão minimizada será  $S = \bar{A}\bar{C} + A\bar{B} + \bar{A}\bar{C}$ .

Poderíamos também ter agrupado de outra maneira, conforme mostra a Figura 2.28.

		$\bar{B}$	B
$\bar{A}$	0	(1)	(1)
A	(1)	(1)	0
$\bar{C}$	C	C	$\bar{C}$

Figura 2.28 - Outra maneira de agrupar.

A expressão gerada seria, então,  $S = \bar{A}\bar{C} + A\bar{C} + \bar{B}C$ .

Estas duas expressões, aparentemente diferentes, possuem o mesmo comportamento em cada possibilidade, fato este comprovado, levantando-se as respectivas tabelas-verdade.

### 2.8.3 Mapa de Karnaugh para quatro variáveis

O mapa para quatro variáveis é apresentado na Figura 2.29.

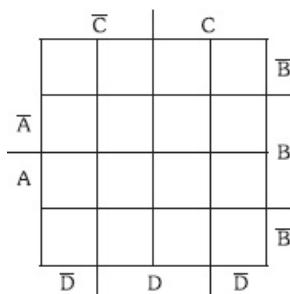
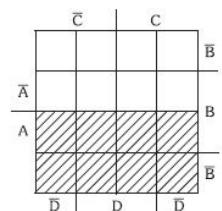
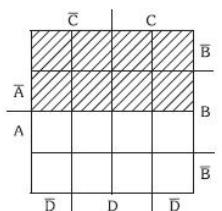


Figura 2.29 - Mapa de Karnaugh para quatro variáveis.

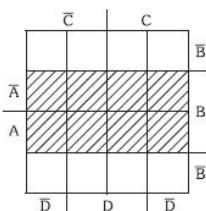
A Figura 2.30 mostra as regiões assumidas pelas variáveis A, B, C e D neste mapa.



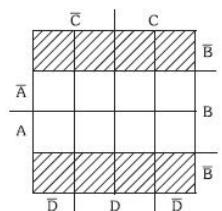
(a) Região onde  $A = 1$ .



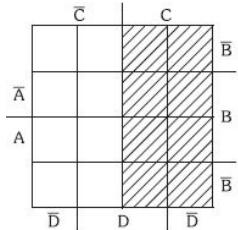
(b) Região onde  
 $\bar{A} = 1$  ( $A = 0$ ).



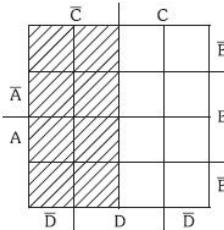
(c) Região onde  $B = 1$ .



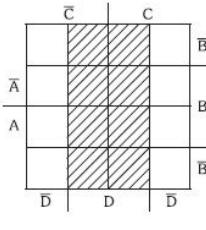
(d) Região onde  
 $\bar{B} = 1$  ( $B = 0$ ).



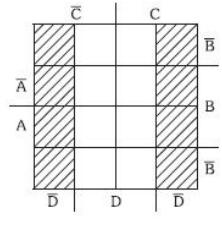
(e) Região onde  $C = 1$ .



(f) Região onde  
 $\bar{C} = 1$  ( $C = 0$ ).



(g) Região onde  $D = 1$ .



(h) Região onde  
 $\bar{D} = 1$  ( $D = 0$ ).

Figura 2.30 - Regiões do mapa de Karnaugh para quatro variáveis (a) a (h).

Neste tipo de mapa, também temos uma região para cada caso da tabela-verdade, como podemos verificar no mapa completo na Figura 2.31.

Tabela 2.9 - Tabela-verdade de quatro variáveis

Casos	A	B	C	D
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
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

		$\bar{C}$	C		
		Caso 0 0 0 0 0 $\bar{A} \bar{B} C D$	Caso 1 0 0 0 1 $\bar{A} \bar{B} C D$	Caso 3 0 0 1 1 $\bar{A} \bar{B} C D$	Caso 2 0 0 1 0 $\bar{A} \bar{B} C \bar{D}$
$\bar{A}$		Caso 4 0 1 0 0 $A \bar{B} \bar{C} \bar{D}$	Caso 5 0 1 0 1 $A \bar{B} \bar{C} D$	Caso 7 0 1 1 1 $A \bar{B} C D$	Caso 6 0 1 1 0 $A \bar{B} C \bar{D}$
		Caso 12 1 1 0 0 $A \bar{B} C \bar{D}$	Caso 13 1 1 0 1 $A \bar{B} C D$	Caso 15 1 1 1 1 $A \bar{B} C D$	Caso 14 1 1 1 0 $A \bar{B} C \bar{D}$
$A$		Caso 8 1 0 0 0 $A \bar{B} \bar{C} \bar{D}$	Caso 9 1 0 0 1 $A \bar{B} \bar{C} D$	Caso 11 1 0 1 1 $A \bar{B} C D$	Caso 10 1 0 1 0 $A \bar{B} C \bar{D}$
		$\bar{D}$	D	$\bar{D}$	

Figura 2.31 - Localização dos casos de quatro variáveis no mapa de Karnaugh.

Vamos analisar a colocação de uma das possibilidades, visto que as outras são análogas. Tomemos como exemplo o caso 8:

$$A\bar{B}\bar{C}\bar{D} \rightarrow 1000$$

$$A = 1, B = 0 (\bar{B} = 1), C = 0 (\bar{C} = 1) \text{ e } D = 0 (\bar{D} = 1)$$

Da intersecção dessas regiões obtemos a região  $A \bar{B} \bar{C} \bar{D}$ , que é a referente ao caso 8:

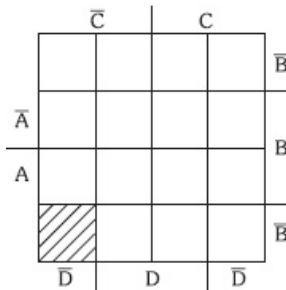


Figura 2.32 - Colocação do caso 8 no mapa de Karnaugh de quatro variáveis.

Para esclarecermos melhor a colocação do mapa e analisarmos outros casos, vamos transpor para ele a Tabela 2.10.

Tabela 2.10 - Tabela exemplo com situações para quatro variáveis

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

Expressão de S, extraída da tabela-verdade:

$$\begin{aligned}
 S = & \overline{ABC}\overline{D} + \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}CD \\
 & + \overline{A}\overline{B}C\overline{D} + A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D \\
 & + A\overline{B}C\overline{D} + AB\overline{C}\overline{D} + AB\overline{C}D + ABCD
 \end{aligned}$$

Transpondo a tabela para o mapa, temos:

		$\overline{C}$		C		$\overline{B}$
		0	1	1	1	
$\overline{A}$		0	1	1	0	B
A	1	1	1	0	$\overline{B}$	$\overline{B}$
	1	1	1	0		
$\overline{D}$		D	D	D	$\overline{D}$	

Figura 2.33 - Colocação dos casos da Tabela 2.10.

Para efetuarmos a simplificação, seguimos o mesmo processo para os mapas de três variáveis, somente que neste caso, o principal agrupamento será a oitava.

Devemos ressaltar que no mapa os lados extremos opostos se comunicam, ou seja, é possível formar oitavas, quadras e pares com os termos localizados nos lados extremos opostos.

Vamos, por exemplo, verificar alguns desses casos no mapa:

- a) Exemplos de pares:

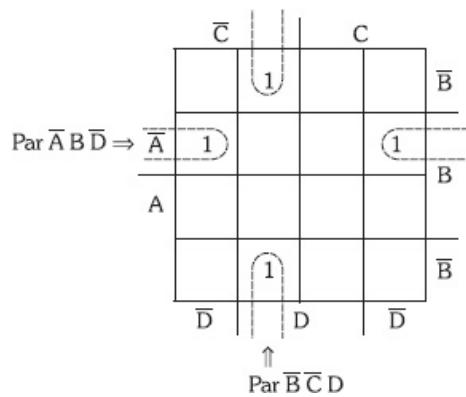


Figura 2.34 - Exemplos de pares no mapa de quatro variáveis.

b) Exemplos de quadras:

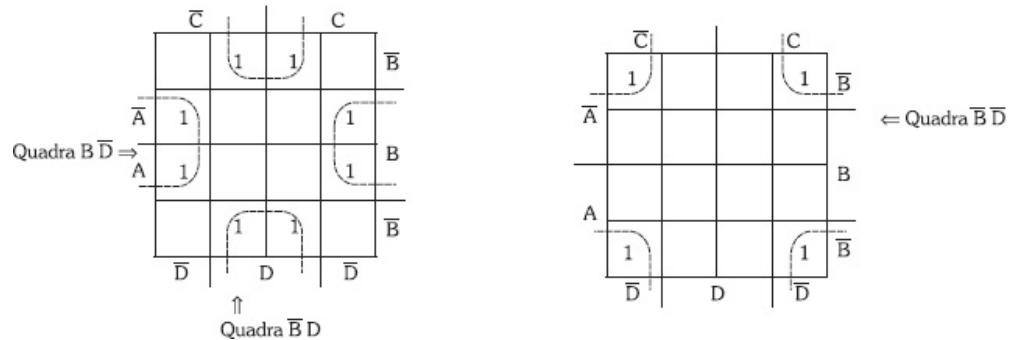


Figura 2.35 - Exemplos de quadras no mapa de quatro variáveis.

c) Exemplos de oitavas:

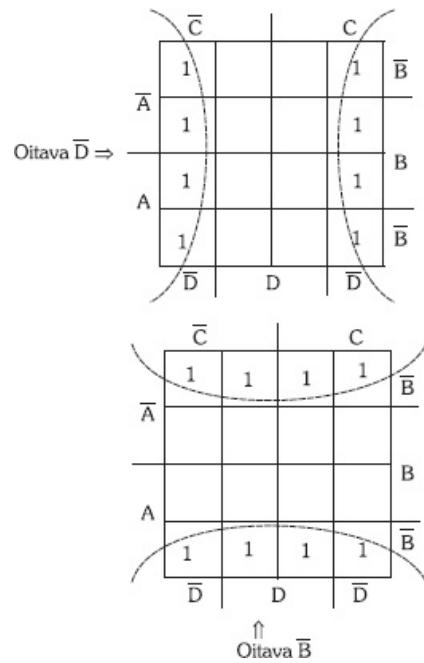


Figura 2.36 - Exemplos de oitavas no mapa de quatro variáveis.

Convém observar que, neste mapa, as oitavas representam as próprias regiões A,  $\bar{A}$ , B,  $\bar{B}$ , C,  $\bar{C}$ , D e  $\bar{D}$  e que o agrupamento máximo (mapa totalmente preenchido com 1) constitui-se em uma **hexa**, ou seja, agrupamento com 16 regiões valendo 1.

Após essa ressalva, vamos minimizar a expressão do nosso exemplo. Inicialmente, agrupamos as oitavas, em seguida as quadras, a seguir os pares e, por último, os termos isolados, se existirem.

Expressões dos agrupamentos:

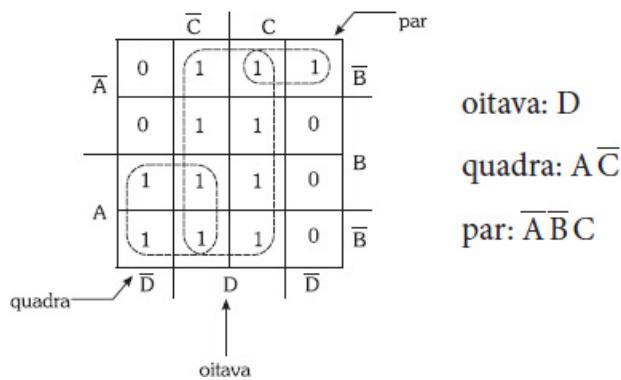


Figura 2.37 - Agrupamentos dos casos da Tabela 2.10.

Somando as expressões, temos a expressão final minimizada, que é:  $S = D + A\bar{C} + \bar{A}\bar{B}C$ .

Como outro exemplo, vamos minimizar o circuito que executa a Tabela 2.11.

Tabela 2.11 - Outro exemplo com situações para quatro variáveis

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

Transpondo a tabela-verdade para o mapa, temos:

Figura 2.38 - Casos da Tabela 2.11 colocados no mapa, agrupamentos e termos extraídos.

No mapa, temos duas quadras, um par e um termo isolado. A expressão minimizada de S será a soma de todos esses agrupamentos:  $S = A\bar{B}C\bar{D} + B\bar{C}D + \bar{A}B + \bar{A}\bar{D}$

### Fique de olho!

É importante ressaltar que uma oitava agrupada representa maior simplificação que uma quadra, e uma quadra agrupada maior simplificação que um par, e, este, maior simplificação que um termo isolado. Assim sendo, deve-se preferir agrupar em oitavas, e se não for possível, em quadras e também se não for possível, em pares, mesmo que em alguns casos já tenham sido considerados em outros agrupamentos, lembrando sempre que devemos ter o menor número de agrupamentos possível.



## Exercícios resolvidos

- 2.1) Simplifique as expressões obtidas das tabelas a seguir, utilizando os mapas de Karnaugh.

a)

Tabela 2.12 - Tabela do item (a) - Exercício 1

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

Transpondo para o mapa de três variáveis e reconhecendo os agrupamentos, temos:

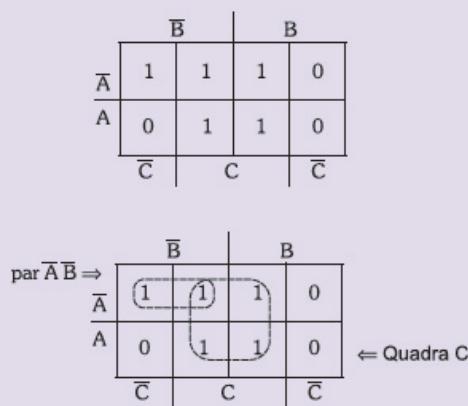


Figura 2.39 - Tabela transposta para o mapa, agrupamentos e termos obtidos.

A expressão minimizada será:

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

b)

Tabela 2.13 - Tabela do item (b) - Exercício 1

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

Transpondo para o diagrama e agrupando, temos:

		$\bar{B}$		B	
		0	0	1	0
		1	0	0	1
$\bar{A}$					
A					
	$\bar{C}$	C			$\bar{C}$

		$\bar{B}$		B	
		0	0	①	0
		1	0	0	①
$\bar{A}$					
A					
	$\bar{C}$	C			$\bar{C}$

Figura 2.40 - Tabela transposta para o mapa, agrupamentos e termos obtidos.

$$\therefore S = A\bar{C} + \bar{A}BC$$

c)

Tabela 2.14 - Tabela do item (c) - Exercício 1

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

Transpondo da tabela para o diagrama, temos:

	$\bar{C}$	$C$			
	1	0	0	1	$\bar{B}$
$\bar{A}$	1	1	1	1	$B$
$A$	1	0	1	1	$B$
	1	0	0	1	$\bar{B}$
$\bar{D}$	D	D	$\bar{D}$		

	$\bar{C}$	$C$			
	1	0	0	1	$\bar{B}$
$\bar{A}$	(1)	1	(1)	(1)	$B$
$A$	1	0	(1)	(1)	$B$
	1	0	0	1	$\bar{B}$
$\bar{D}$	D	D	$\bar{D}$		

↔ oitava  $\bar{D}$

↔ quadras:  $\bar{A}B$  e  $BC$

Figura 2.41 - Tabela transposta para o mapa, agrupamentos e termos obtidos.

A expressão minimizada será  $S = \bar{A}\bar{B} + BC + \bar{D}$ .

2.2) Minimize a expressão a seguir usando o mapa de Karnaugh:

a)  $S = \bar{A}\bar{B}\bar{C} + \bar{ABC} + \bar{ABC} + ABC$

Colocando os termos diretamente no mapa, temos:

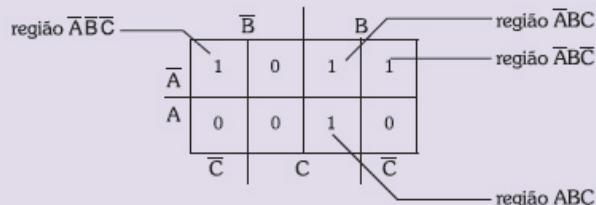


Figura 2.42 - Termos da expressão transpostos diretamente para o mapa.

Existem neste diagrama dois pares:

	$\bar{B}$	$B$			
$\bar{A}$	1	0	1	1	
$A$	0	0	1	0	
$\bar{C}$		$C$		$\bar{C}$	
					região ABC

Figura 2.43 - Agrupamentos e termos obtidos.

A expressão minimizada será:

$$S = \bar{A}\bar{C} + BC.$$

### Fique de olho!

Notamos que as expressões inseridas diretamente nos mapas possuem formato básico, caracterizado por uma soma de produtos. Para a colocação de expressões mais complexas, como as utilizadas nos itens de simplificação por álgebra de Boole, recomenda-se primeiramente levantar a tabela-verdade para extrair a expressão básica e posteriormente inserir cada termo no mapa.

## 2.8.4 Mapas com condições irrelevantes

Chamamos de condição irrelevante (X) a situação de entrada em que a saída pode assumir 0 ou 1 indiferentemente. Esta condição ocorre principalmente pela impossibilidade prática do caso de entrada acontecer, sendo utilizada em várias situações. Para a sua utilização em mapas de Karnaugh, devemos, para cada condição irrelevante, adotar 0 ou 1, dos dois, aquele que possibilitar melhor agrupamento e, consequentemente, maior simplificação.

Para esclarecer este processo, vamos utilizar a Tabela 2.15.

Tabela 2.15 - Exemplo de tabela verdade com condições irrelevantes

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

Transpondo esta tabela para o mapa, temos:

	$\bar{B}$	B	
$\bar{A}$	X	1	1
A	0	0	0
$\bar{C}$		C	$\bar{C}$

Figura 2.44 - Tabela transposta para o mapa.

O símbolo (X) indica que neste caso a saída pode assumir 0 ou 1, indiferentemente, pois a situação de entrada é impossível de acontecer ou, ainda, possibilita qualquer dos dois valores na saída. Para fins de simplificação, devemos

adotar  $X = 1$ , assim sendo, agrupamos uma quadra em vez de dois pares (no caso de  $X = 0$ ), representando maior simplificação da expressão de saída:  $S = \overline{A}$ .

Convém ressaltar que em uma tabela-verdade podemos ter várias condições irrelevantes que devem ser consideradas independentemente, conforme agrupamento em que se encontram.

Para exemplificar, vamos simplificar a expressão extraída da Tabela 2.16.

Tabela 2.16 - Outro exemplo de tabela-verdade com condições irrelevantes

A	B	C	D	S
0	0	0	0	X
0	0	0	1	0
0	0	1	0	1
0	0	1	1	X
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	0
1	1	0	0	0
1	1	0	1	X
1	1	1	0	0
1	1	1	1	X

Passando para o mapa de quatro variáveis, temos:

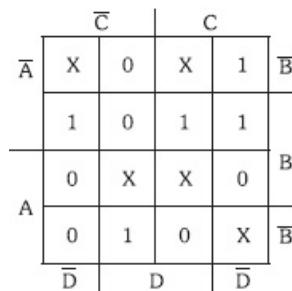


Figura 2.45 - Tabela transposta para o mapa.

O próximo passo é agrupar as regiões que valem 1, utilizando a condição irrelevante (X) para completar o agrupamento. Convém lembrar que, para maior simplificação, devemos ter um número mínimo de agrupamentos, cada um deles, porém com o maior número de células possível. Assim sendo, temos:

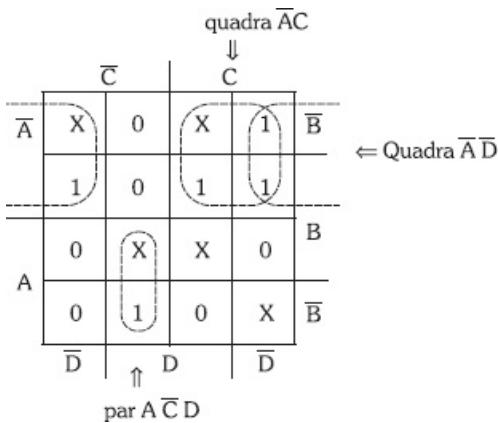


Figura 2.46 - Agrupamentos e expressões extraídas.

Notamos, na Figura 2.48, que as condições irrelevantes pertencentes aos agrupamentos receberam valor 1, enquanto as deixadas de fora, valor 0.

A expressão simplificada será composta por duas quadras e um par:  $S = \overline{AC} + \overline{AD} + A\overline{CD}$ .



## Exercícios resolvidos

- 2.1) A Tabela 2.17 representa as possibilidades de saída obtidas de um projeto envolvendo três variáveis A, B e C. Determine a expressão simplificada.

Tabela 2.17 - Tabela-verdade do exercício 1

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

A Figura 2.47 apresenta a colocação dos valores da tabela no diagrama (a) e os respectivos agrupamentos para a obtenção da expressão simplificada (b).

	$\bar{B}$	B	
$\bar{A}$	1	X	1
A	X	1	X
	$\bar{C}$	C	$\bar{C}$

(a)

	$\bar{B}$	B	
$\bar{A}$	(1)	(X)	1)
A	X	(1)	X
	$\bar{C}$	C	$\bar{C}$

(b)

Figura 2.47 - Casos da tabela colocados no mapa e agrupamentos.

A expressão simplificada será composta pelas duas quadras obtidas:  $S = \bar{B} + C$ .

## 2.2) Simplifique a expressão representativa da Tabela 2.18.

Tabela 2.18 - Tabela-verdade do exercício 2

A	B	C	D	S
0	0	0	0	1
0	0	0	1	X
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	X
1	0	1	1	1
1	1	0	0	X
1	1	0	1	1
1	1	1	0	X
1	1	1	1	0

Passando os valores da tabela para o mapa e efetuando os agrupamentos visando obter a expressão simplificada de forma máxima, temos:

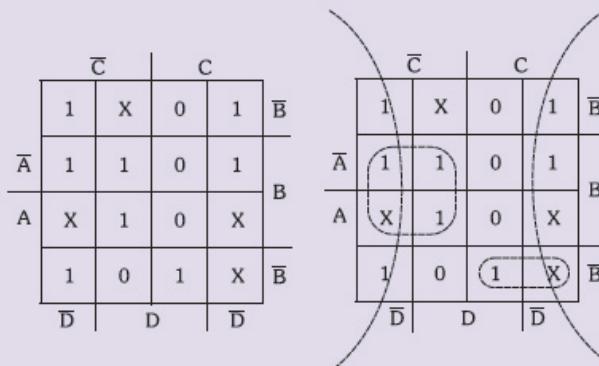


Figura 2.48 - Casos da tabela colocados no mapa e agrupamentos.

Como resultado, obtemos a oitava  $\bar{D}$ , a quadra  $B\bar{C}$  e o par  $A\bar{B}C$ , gerando a expressão:

$$S = \bar{D} + B\bar{C} + A\bar{B}C.$$

É importante observar que, se tivéssemos agrupado precipitadamente ao início do exercício a quadra  $\bar{A}\bar{C}$ , geraríamos erradamente um termo a mais na expressão final. Para melhor condução do processo de agrupamento, devemos iniciar sempre pelos obrigatórios e bem definidos.

### Vamos recapitular?

Neste capítulo, você conheceu os postulados, as propriedades, teoremas fundamentais e identidades que compõe a álgebra de Boole.

Aprendeu a simplificar expressões booleanas a partir da manipulação algébrica utilizando esses conceitos.

Conheceu outra forma de simplificação de expressões booleanas de maneira imediata e menos trabalhosa, por meio da utilização dos mapas de Karnaugh.

Verificou também que o método de Karnaugh pode ser aplicado extraindo-se os casos diretamente da tabela-verdade ou, conforme a expressão, com a colocação direta de cada termo no mapa.



### Agora é com você!

- 1) Simplifique cada expressão, utilizando a álgebra de Boole:

a)  $S = ABC + \overline{A}\overline{B}C + ABC + \overline{A}\overline{B}C + \overline{ABC}$

b)  $S = ABCD + \overline{A}\overline{B}CD + A\overline{B}\overline{C}D + \overline{ABC}\overline{D} + ABC\overline{D} + A\overline{B}\overline{C}D + ABCD$

2) Simplifique utilizando a álgebra de Boole:

$$S = [(\overline{\overline{B}} + \overline{\overline{C}} + \overline{\overline{D}})(\overline{\overline{A}} + B + C) + C] + \overline{A}\overline{B}C + \overline{B}(\overline{A} + C)$$

3) Idem para a expressão:

$$S = A[\overline{B}(\overline{C} + D) + \overline{A}(B + C)] + C\overline{D} + A\overline{B}C + AB$$

4) Idem para a expressão:

$$S = (A \oplus B + \overline{B}CD)[\overline{D} + \overline{B}C + D(\overline{\overline{A}} + B)] + \overline{A}\overline{D}$$

5) Desenhe o circuito que executa a expressão, simplificado:

$$S = (\overline{\overline{B}} + \overline{\overline{D}})\{\overline{\overline{B}} + C \odot D + \overline{A}[\overline{B}\overline{C} + \overline{B}C + A + B(\overline{\overline{C}} + \overline{\overline{D}})]\}$$

6) Através dos mapas de Karnaugh, determine a expressão simplificada de  $S_1$  e  $S_2$  da Tabela 2.19:

Tabela 2.19 - Tabela-verdade do exercício 6

A B	$S_1$	$S_2$
0 0	1	1
0 1	0	1
1 0	1	0
1 1	1	0

7) Simplifique as expressões de  $S_1$ ,  $S_2$ ,  $S_3$  e  $S_4$  da Tabela 2.20, utilizando os mapas de Karnaugh:

Tabela 2.20 - Tabela-verdade do exercício 7

A	B	C	$S_1$	$S_2$	$S_3$	$S_4$
0	0	0	1	1	0	0
0	0	1	0	1	1	1
0	1	0	1	1	0	1
0	1	1	1	0	0	0
1	0	0	1	1	1	1
1	0	1	1	1	1	0
1	1	0	0	1	1	1
1	1	1	1	0	0	1

8) Idem ao anterior para a Tabela 2.21:

Tabela 2.21 - Tabela-verdade do exercício 8

A	B	C	D	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	S <sub>4</sub>
0	0	0	0	1	1	0	0
0	0	0	1	1	0	0	0
0	0	1	0	1	1	1	0
0	0	1	1	1	0	0	1
0	1	0	0	1	1	1	1
0	1	0	1	0	1	1	1
0	1	1	0	0	1	1	0
0	1	1	1	1	1	0	1
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	0	1	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	0	1	1	1
1	1	1	0	0	0	0	1
1	1	1	1	1	1	0	1

9) Simplifique as expressões utilizando mapas de Karnaugh:

- a)  $S = \overline{ABC} + A\overline{B}C + \overline{A}BC + \overline{ABC} + ABC$
- b)  $S = \overline{A}\overline{B}CD + \overline{A}\overline{B}CD + \overline{A}\overline{B}\overline{C}D + AB\overline{C}D + \overline{ABC}D + A\overline{B}\overline{C}D + ABCD + A\overline{B}\overline{C}\overline{D}$
- c)  $S = \overline{BD} + \overline{A} + A\overline{B}\overline{C}D + A\overline{B}CD + \overline{AC}$

10) Simplifique as expressões de S<sub>1</sub> e S<sub>2</sub> da Tabela 2.22:

Tabela 2.22 - Tabela-verdade do exercício 10

A	B	C	S <sub>1</sub>	S <sub>2</sub>
0	0	0	X	1
0	0	1	0	X
0	1	0	1	0
0	1	1	X	0
1	0	0	1	0
1	0	1	X	1
1	1	0	X	X
1	1	1	1	X

11) Determine as expressões simplificadas de S<sub>1</sub>, S<sub>2</sub>, S<sub>3</sub> e S<sub>4</sub> da Tabela 2.23:

Tabela 2.23 - Tabela-verdade do exercício 12

A	B	C	D	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	S <sub>4</sub>
0	0	0	0	1	X	0	X
0	0	0	1	X	X	0	0
0	0	1	0	X	1	0	X
0	0	1	1	X	0	1	1
0	1	0	0	1	X	X	1
0	1	0	1	0	1	X	X
0	1	1	0	X	0	1	0
0	1	1	1	X	1	0	1
1	0	0	0	X	1	X	0
1	0	0	1	1	0	1	1
1	0	1	0	X	X	0	0
1	0	1	1	1	1	0	X
1	1	0	0	X	0	1	1
1	1	0	1	X	1	0	1
1	1	1	0	1	1	X	1
1	1	1	1	0	X	1	X

# 3

## Circuitos Combinacionais

### Para começar

Os sistemas digitais são basicamente divididos em dois campos: lógica combinacional e lógica sequencial. Neste capítulo estudaremos o desenvolvimento de circuitos digitais combinacionais para aplicações específicas.

Um dos temas importantes dos sistemas digitais é o que trata dos circuitos combinacionais. Pelo seu estudo podemos compreender o funcionamento de circuitos, tais como: codificadores, decodificadores, circuitos aritméticos: somadores, subtraidores e outros sistemas derivados.

A partir daí, utilizando esses elementos, outros circuitos serão estudados com aplicações em sistemas digitais e, ainda,

na arquitetura interna de circuitos integrados utilizados em computadores.

### 3.1 Definição e aplicações dos circuitos combinacionais

O circuito combinacional é aquele cuja saída depende única e exclusivamente das combinações entre as variáveis de entrada.

Podemos utilizar um circuito lógico combinacional para solucionar problemas em que necessitamos de uma resposta, quando acontecerem determinadas situações representadas pelas variáveis de entrada. Para construirmos esses circuitos, necessitamos de suas expressões características que são obtidas das tabelas-verdade que representam as situações já mencionadas.

A Figura 3.1 ilustra a sequência do processo em que, a partir da situação, obtemos a tabela-verdade e a partir dela, através das técnicas já conhecidas, a expressão simplificada e o circuito final.

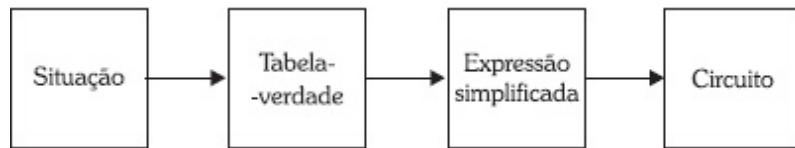


Figura 3.1 - Sequência típica de projeto de um circuito combinacional.

Este capítulo aborda os circuitos combinacionais destinados principalmente a aplicações específicas, empregados, sobretudo na arquitetura interna de circuitos integrados e, ainda, em sistemas digitais.

Entre os circuitos destinados a estas finalidades destacamos os codificadores, decodificadores e os circuitos aritméticos. Eles serão abordados de forma básica como projetos combinacionais, para melhor entendimento, sendo encontrados na prática, disponíveis em circuitos integrados comerciais ou internos a circuitos integrados dedicados.

Para a construção dos codificadores e decodificadores, vamos inicialmente conhecer dois códigos digitais, que serão muito úteis nos itens de execução dos projetos já referidos.

## 3.2 Códigos

São vários os códigos dentro dos sistemas digitais. Vamos, neste tópico, descrever o código BCD 8421, que é um código binário e o Código 9876543210, que é um código decimal.

### Lembre-se

O **Sistema Binário de Numeração** possui apenas dois algarismos: **0** e **1** e sua base é **2**.

A Tabela 3.1 apresenta a regra de formação do sistema binário até o valor 9 no sistema decimal.

Tabela 3.1 - Formação do sistema binário até 9 no sistema decimal

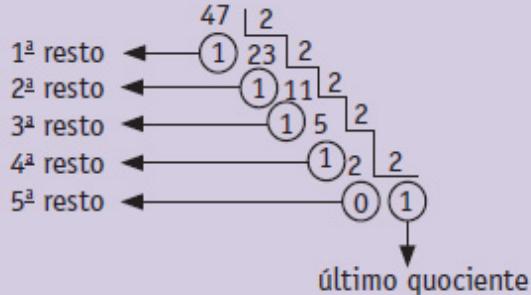
Decimal	Binário
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

A conversão para o sistema decimal se dá pelo somatório de cada algarismo correspondente multiplicado pela base elevada por um índice, conforme o posicionamento do algarismo no número.

Exemplo:  $1001_2 \Rightarrow 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 1 \times 8 + 0 + 0 + 1 \times 1 = 9_{10}$

A conversão contrária, ou seja, do sistema decimal para o binário se dá efetuando-se sucessivas divisões pela base a ser convertida (no caso o 2) até o último quociente possível. O número transformado será composto por este último quociente (algarismo mais significativo) e todos os restos, na ordem inversa às divisões.

Exemplo:  $47_{10} \Rightarrow 101111_2$



Na prática, cada dígito binário recebe a denominação de ***bit*** (*binary digit*). O conjunto de 4 bits é denominado ***nibble*** e o de 8 bits de ***byte***.

### 3.2.1 Código BCD 8421

Vamos iniciar explicando que no nome deste código, a sigla BCD representa as iniciais de Binary Coded Decimal, que significa uma codificação do sistema decimal em binário. Os termos seguintes (8421) significam os valores dos algarismos em um dado número binário, que representam, respectivamente,  $2^3$ ,  $2^2$ ,  $2^1$  e  $2^0$ .

A formação deste código é encontrada na Tabela 3.2.

Tabela 3.2 - Código BCD 8421

Decimal	BCD 8421			
	A	B	C	D
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

O número de *bits* de um código é o número de dígitos binários que ele possui. Notamos, que o código BCD 8421 é de 4 *bits* e, ainda, que é válido de 0 a 9<sub>10</sub>.

### 3.2.2 Código 9876543210

Notamos no código que em dez saídas somente uma vale 1 em cada caso, conforme o algarismo correspondente. A formação desse código é vista na Tabela 3.3.

Tabela 3.3 - Código 9876543210

Decimal	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0

### 3.3 Codificadores e decodificadores

Vamos tratar de circuitos que efetuam a passagem de um determinado código para outro. Primeiramente, vamos fazer uma análise do significado das palavras codificador e decodificador.

Chamamos de codificador o circuito combinacional que torna possível a passagem de um código conhecido para um desconhecido. Como exemplo, podemos citar o circuito inicial de uma calculadora que transforma uma entrada decimal, através do sistema de chaves de um teclado, em saída binária para que o circuito interno processe e faça a operação.

Chamamos de decodificador o circuito que faz o inverso, ou seja, passa um código desconhecido para um conhecido. No exemplo citado é o circuito que recebe o resultado da operação em binário e o transforma em saída decimal, na forma compatível para um mostrador digital apresentar os algarismos.

A Figura 3.2 ilustra o exemplo utilizado.

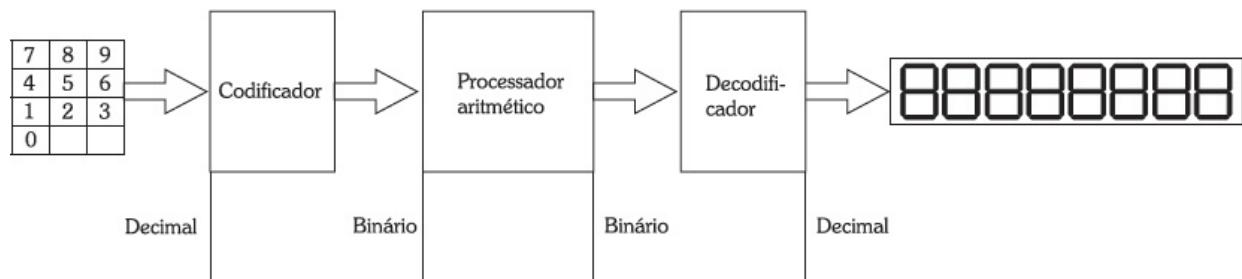


Figura 3.2 - Estrutura genérica de uma calculadora.

Os termos codificador e decodificador, porém, diferenciam-se em função do referencial. Se para o usuário da calculadora o sistema de entrada é um codificador, para o processador será um decodificador, pois passa de um código desconhecido para ele (decimal), para um conhecido (binário). Na prática, é comum utilizar a denominação de decodificador para o sistema que passa de um código para outro, quaisquer que sejam.

### Amplie seus conhecimentos

Conforme o tipo de circuito interno das portas lógicas, o terminal de entrada em vazio (aberto, sem ligação) é equivalente ao nível lógico 1. Essa característica ocorre principalmente na família de circuito denominada TTL (*Transistor-Transistor Logic*) e foi considerada neste projeto do codificador.

### 3.3.1 Codificador decimal/binário

Neste item vamos elaborar um codificador para transformar um código decimal em binário (BCD 8421). A entrada do código decimal é feita através de um conjunto de chaves numeradas de 0 a 9 e a saída por quatro fios, para fornecer um código binário de 4 bits, correspondente à chave acionada. A Figura 3.3 mostra a estrutura geral desse sistema, sendo convencionado que a chave fechada (ligada ao fio terra) equivale a nível 0, e aberta equivalente a nível lógico 1.

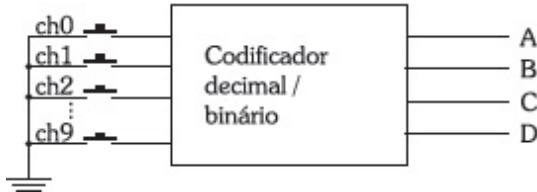


Figura 3.3 - Estrutura geral de um codificador decimal/binário.

Na Tabela 2.4, vamos construir a tabela-verdade do codificador que relaciona cada chave de entrada decimal com a respectiva saída em binário.

Tabela 3.4 - Código BCD 8421

Chave	A	B	C	D
Ch0	0	0	0	0
Ch1	0	0	0	1
Ch2	0	0	1	0
Ch3	0	0	1	1
Ch4	0	1	0	0
Ch5	0	1	0	1
Ch6	0	1	1	0
Ch7	0	1	1	1
Ch8	1	0	0	0
Ch9	1	0	0	1

Pela tabela concluímos que a saída A vale 1 quando Ch8 ou Ch9 for acionada. A saída B quando Ch4, Ch5, Ch6 ou Ch7 for acionada. A saída C quando Ch2, Ch3, Ch6 ou Ch7 for acionada. A saída D quando Ch1, Ch3, Ch5, Ch7 ou Ch9 for acionada.

Usaremos para a construção do circuito uma porta NE em cada saída, ela fornece nível 1 quando qualquer uma de suas entradas assumir nível 0, situação compatível com a convenção adotada para o conjunto de chaves. A ligação das entradas de cada porta será feita, conforme a análise efetuada, às chaves responsáveis pelos níveis 1 de cada saída.

O circuito, assim constituído, é visto na Figura 3.4.

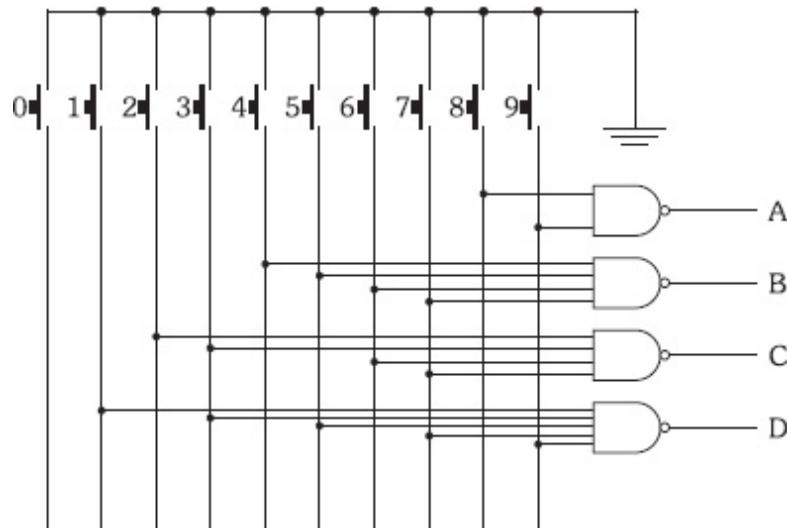


Figura 3.4 - Circuito codificador decimal/binário.

Pela figura, notamos que a chave Ch0 não está ligada a nenhuma das entradas das portas, sendo irrelevante o seu acionamento, a saída também será igual a 0 ( $A = B = C = D = 0$ ) quando nenhuma das chaves for acionada.

### 3.3.2 Decodificador binário/decimal

A estrutura geral desse decodificador é vista na Figura 3.5.

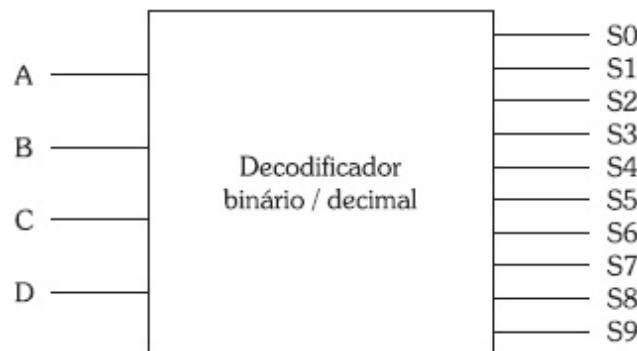


Figura 3.5 - Estrutura geral de um decodificador binário/decimal.

Vamos montar a tabela-verdade do circuito no qual as entradas são *bits* do código BCD 8421 e as saídas são os respectivos *bits* do código decimal 9876543210.

Tabela 3.5 - Decodificador BCD 8421/9876543210

BCD 8421				Código 9876543210									
A	B	C	D	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0

O código BCD 8421 não possui números maiores que 9, logo, tanto faz o valor assumido nas possibilidades excedentes, visto que, quando passarmos do código BCD 8421 para o código 9876543210, estas não vão ocorrer. Nos mapas de Karnaugh, consequentemente, consideraremos esses casos como condições irrelevantes. A Figura 3.6 mostra os diagramas de todas as saídas do decodificador ( $S_9$  a  $S_0$ ) e suas respectivas simplificações.

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

$$(S_9) S_9 = AD$$

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

$$(S_8) S_8 = \overline{AD}$$

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

$$(S_7) S_7 = BCD$$

	$\bar{C}$	$C$	
$\bar{A}$	0 0	0 0	$\bar{B}$
A	X X	X X	B
$\bar{D}$	0 0	X X	$\bar{B}$

$$(S_6) S_6 = B\bar{C}\bar{D}$$

	$\bar{C}$	$C$	
$\bar{A}$	0 0	0 0	$\bar{B}$
A	X X	X X	B
$\bar{D}$	0 0	X X	$\bar{B}$

$$(S_5) S_5 = B\bar{C}D$$

	$\bar{C}$	$C$	
$\bar{A}$	0 0	0 0	$\bar{B}$
A	X X	X X	B
$\bar{D}$	0 0	X X	$\bar{B}$

$$(S_4) S_4 = B\bar{C}\bar{D}$$

	$\bar{C}$	$C$	
$\bar{A}$	0 0	0 0	$\bar{B}$
A	X X	X X	B
$\bar{D}$	0 0	X X	$\bar{B}$

$$(S_3) S_3 = \bar{B}CD$$

	$\bar{C}$	$C$	
$\bar{A}$	0 0	0 0	$\bar{B}$
A	X X	X X	B
$\bar{D}$	0 0	X X	$\bar{B}$

$$(S_2) S_2 = \bar{B}\bar{C}\bar{D}$$

Figuras 3.6 - Mapas com simplificações das saídas ( $S_9$  a  $S_0$ ) do decodificador BCD 8421/9876543210.

$\bar{C}$	$C$		$\bar{B}$
0	(1)	0	0
$\bar{A}$	0	0	0
A	X	X	X
0	0	X	X
$\bar{D}$	D	D	$\bar{D}$

$$(S_1) S_1 = \overline{ABC}\bar{D}$$

$\bar{C}$	$C$		$\bar{B}$
(1)	0	0	0
$\bar{A}$	0	0	0
A	X	X	X
0	0	X	X
$\bar{D}$	D	D	$\bar{D}$

$$(S_0) S_0 = \overline{ABC}\bar{D}$$

Figuras 3.6 - Mapas com simplificações das saídas ( $S_9$  a  $S_0$ ) do decodificador BCD 8421/9876543210 (Continuação).

A partir das expressões simplificadas obtemos o circuito do decodificador da Figura 3.7.

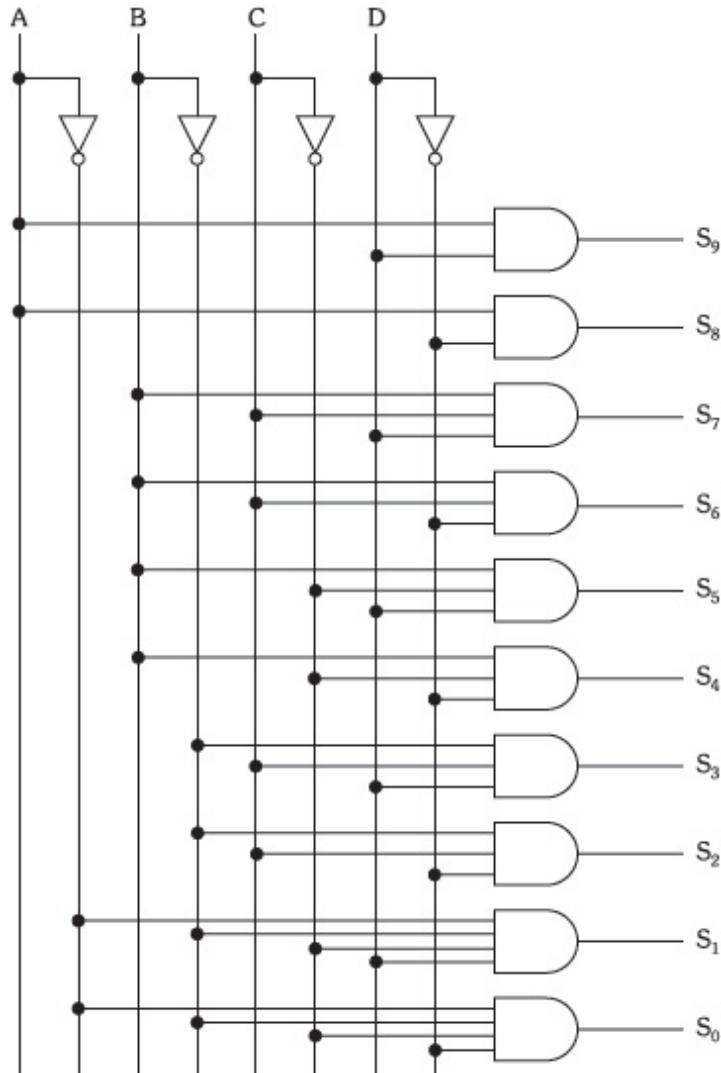


Figura 3.7 - Circuito de um decodificador binário/decimal.

### 3.3.3 Decodificador para display de 7 segmentos

O *display* de 7 segmentos possibilita escrevermos números decimais de 0 a 9 e alguns outros símbolos que podem ser letras ou sinais. A Figura 3.8 representa uma unidade do *display* genérica com a nomenclatura de identificação dos segmentos mais usual entre os fabricantes.

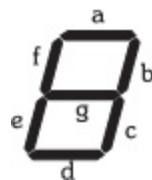


Figura 3.8 - *Display* genérico com os segmentos identificados.

Entre as tecnologias de fabricação das unidades de *display* usaremos o mais comum que é o *display* a LED, que possui cada segmento composto por um LED, que acende quando aplicado nível 1 (LED do tipo **catodo comum**).

### Amplie seus conhecimentos

O LED (*Light Emitting Diode*) é um diodo emissor de luz. E por ser um diodo possui um terminal ligado ao anodo e outro ligado ao seu catodo. O *display* denominado catodo comum é aquele que possui todos os catodos dos LEDs interligados, sendo necessário aplicar nível 1 no anodo respectivo para acender cada segmento. Já o de anodo comum possui todos os anodos interligados, sendo preciso aplicar nível 0 ao catodo respectivo.

Outro ponto a ser realçado é que em uma montagem prática, a ligação do *display* se faz com resistores para respeitar os limites máximos de corrente nos LEDs.

A título de exemplo vamos elaborar um decodificador para a partir de um código binário (BCD 8421) escrever a sequência de 0 a 9 em um *display* de 7 segmentos catodo comum. O esquema geral desse decodificador é visto na Figura 3.9.

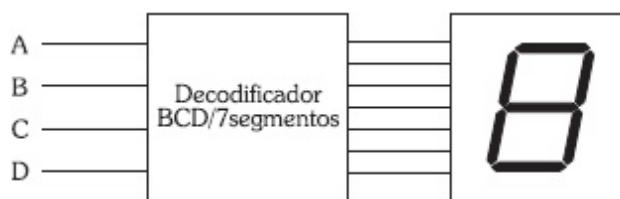
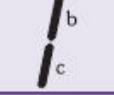
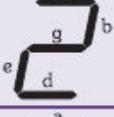
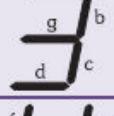
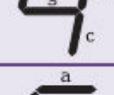
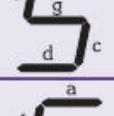
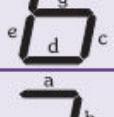
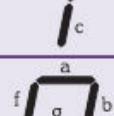
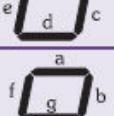
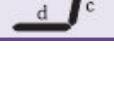


Figura 3.9 - Esquema geral do decodificador BCD8421/7 segmentos.

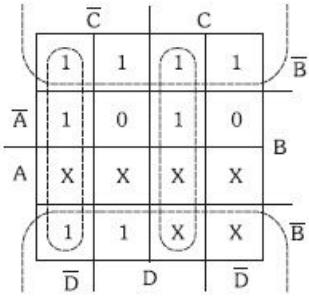
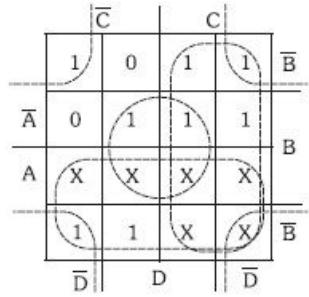
Para efetuar o projeto desse decodificador, devemos verificar em cada caractere os segmentos que devem ser acesos e atribuir o nível

1 (no caso do catodo comum), em função da respectiva entrada no código binário. A Tabela 3.6 apresenta a sequência de caracteres, o respectivo código de entrada e os níveis aplicados em cada segmento para que tal ocorra.

Tabela 3.6 - Sequência de caracteres do decodificador BCD8421/7segmentos

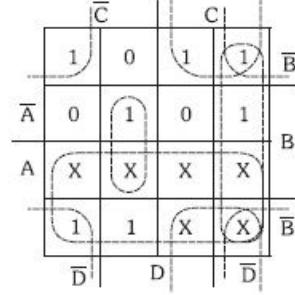
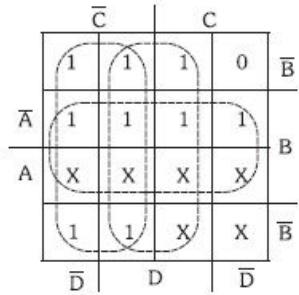
Caracteres	Display	BCD 8421				Código para 7 segmentos						
		A	B	C	D	a	b	c	d	e	f	g
0		0	0	0	0	1	1	1	1	1	1	0
1		0	0	0	1	0	1	1	0	0	0	0
2		0	0	1	0	1	1	0	1	1	0	1
3		0	0	1	1	1	1	1	1	0	0	1
4		0	1	0	0	0	1	1	0	0	1	1
5		0	1	0	1	1	0	1	1	0	1	1
6		0	1	1	0	1	0	1	1	1	1	1
7		0	1	1	1	1	1	1	0	0	0	0
8		1	0	0	0	1	1	1	1	1	1	1
9		1	0	0	1	1	1	1	1	0	1	1

Para fins de simplificação, vamos considerar os casos fora da sequência como irrelevantes. Transpondo as saídas para os diagramas, temos:



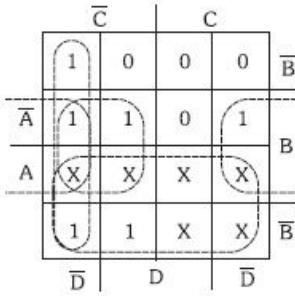
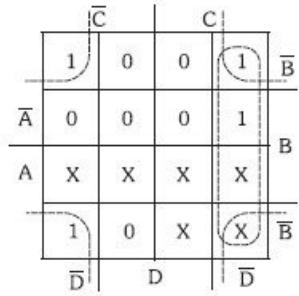
$$(a) a = A + C + BD + \bar{B}\bar{D} \text{ ou } a = A + C + B \odot D$$

$$(b) b = \bar{B} + \bar{C}\bar{D} + CD \text{ ou } b = \bar{B} + C \odot D$$



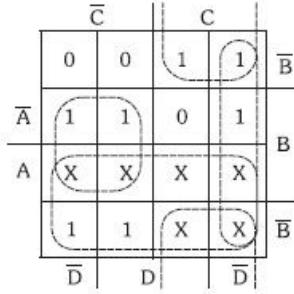
$$(c) c = B + \bar{C} + D$$

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



$$(e) e = \bar{B}\bar{D} + CD$$

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



$$(g) g = A + \bar{B}\bar{C} + \bar{B}C + \bar{C}\bar{D} \text{ ou } g = A + B \oplus C + \bar{C}\bar{D}$$

Figura 3.10 - Mapas com as simplificações.

O circuito do decodificador BCD 8421 para *display* de 7 segmentos obtido encontra-se na Figura 3.11.

Os *displays* de 7 segmentos podem ainda escrever outros caracteres, que são frequentemente utilizados em sistemas digitais para representar outras funções, bem como formar palavras-chave em *software* de programação. A Tabela 3.7 mostra como exemplo outras possibilidades de caracteres.

Tabela 3.7 - Outras possibilidades de caracteres

A	b	C c	d	E e	F
G g	H h	I i	J j	L l	N n
n	0 o	P	9	r	S s
t	u	U	y	-	z

Para efetuar o projeto, basta verificar caso a caso quais segmentos devem acender e montar, a tabela-verdade.

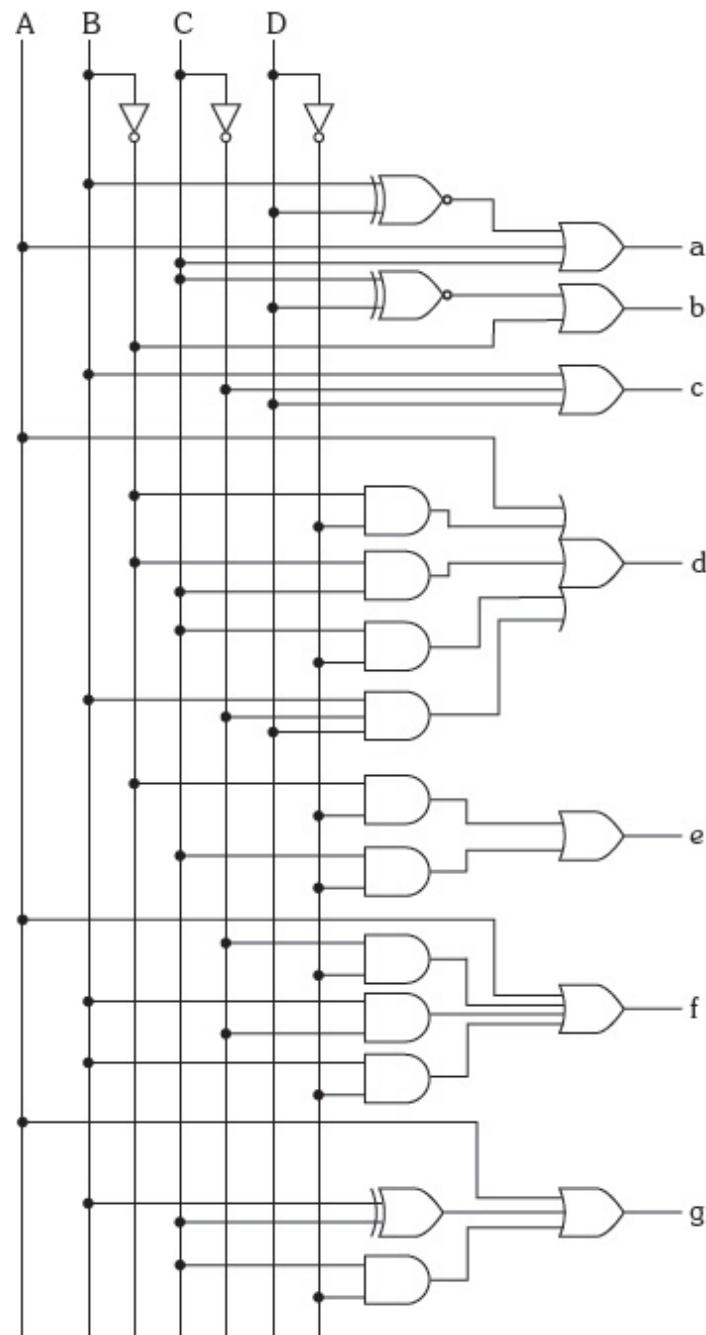


Figura 3.11 - Circuito do decodificador BCD842/7segmentos.



## Exercício resolvido

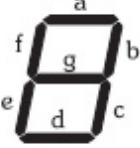
- 3.1) Projete um decodificador para, a partir de um código binário, escrever a sequência da Figura 3.12 em um *display* de 7 segmentos catodo comum.

Caractere	<i>S</i>	<i>E</i>	<i>o</i>	<i>P</i>	-	<i>E</i>	<i>r</i>	<i>B</i>
Caso	0	1	2	3	4	5	6	7

Figura 3.12 - Sequência do projeto.

Para escrever os oito símbolos mostrados na Figura, um código binário de 3 *bits* é suficiente. A Tabela 3.8 apresenta o código binário de entrada e os níveis aplicados em cada segmento para escrever a sequência de caracteres.

Tabela 3.8 - Código binário de entrada e níveis dos segmentos

	A	B	C	a	b	c	d	e	f	g
	0	0	0	1	0	1	1	0	1	1
	0	0	1	0	0	0	1	1	1	1
	0	1	0	0	0	1	1	1	0	1
	0	1	1	1	1	0	0	1	1	1
	1	0	0	0	0	0	0	0	0	1
	1	0	1	1	0	0	1	1	1	1
	1	1	0	0	0	0	0	1	0	1
	1	1	1	1	1	1	1	1	1	1

A Figura 3.13 apresenta o diagrama e a simplificação do circuito de saída para cada segmento.

	$\bar{B}$		$B$
$\bar{A}$	1	0	1
A	0	1	1
	$\bar{C}$	C	$\bar{C}$

(a)  $a = \bar{A}\bar{B}\bar{C} + BC + AC$

	$\bar{B}$		$B$
$\bar{A}$	0	0	1
A	0	0	1
	$\bar{C}$	C	$\bar{C}$

(b)  $b = BC$

	$\bar{B}$		$B$
$\bar{A}$	1	0	0
A	0	0	1
	$\bar{C}$	C	$\bar{C}$

(c)  $c = \bar{A}\bar{C} + ABC$

	$\bar{B}$		$B$
$\bar{A}$	1	1	0
A	0	1	1
	$\bar{C}$	C	$\bar{C}$

(d)  $d = \bar{A}\bar{C} + AC + \bar{B}C$  ou  
 $d = A \odot C + \bar{B}C$

	$\bar{B}$		$B$
$\bar{A}$	0	1	1
A	0	1	1
	$\bar{C}$	C	$\bar{C}$

(e)  $e = B + C$

	$\bar{B}$		$B$
$\bar{A}$	1	1	1
A	0	1	0
	$\bar{C}$	C	$\bar{C}$

(f)  $f = \bar{A}\bar{B} + C$

	$\bar{B}$		$B$
$\bar{A}$	0	1	1
A	0	1	1
	$\bar{C}$	C	$\bar{C}$

(g)  $g = 1$

Figura 3.13 - Mapas com as simplificações.

O circuito extraído das expressões simplificadas está na Figura 3.14.

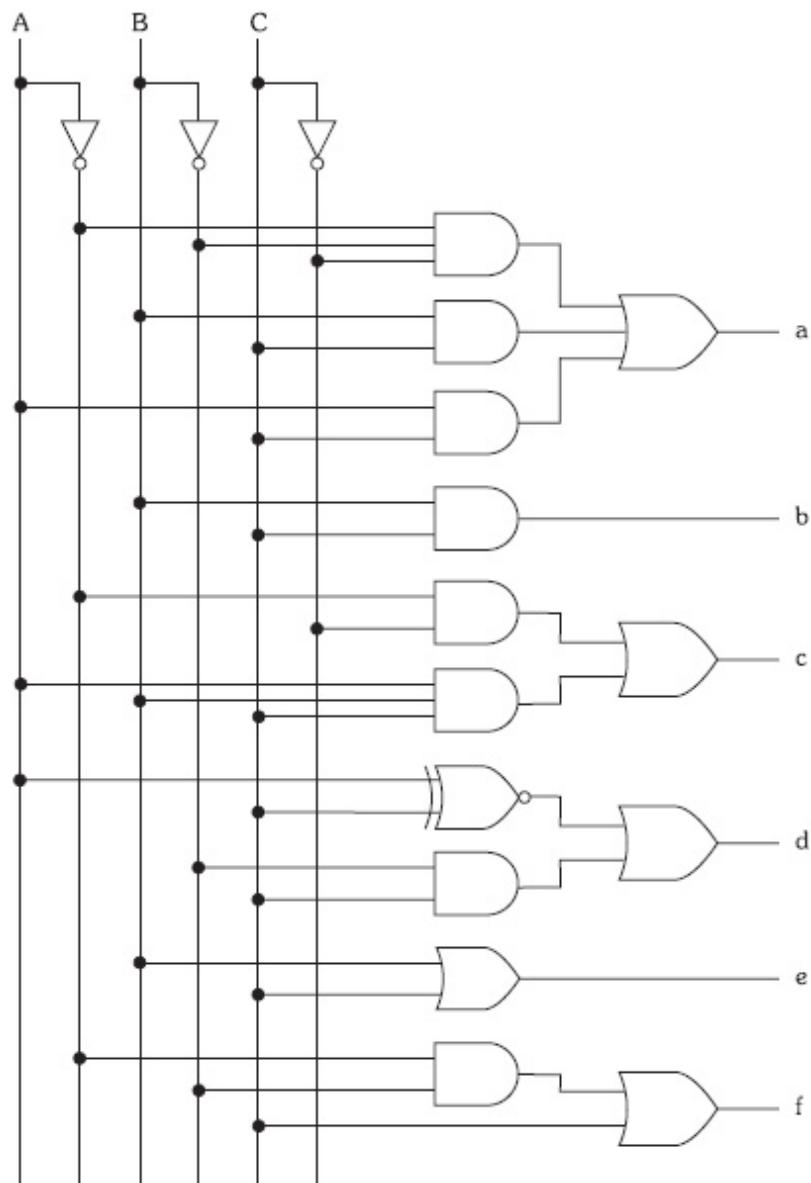


Figura 3.14 - Circuito final.

### 3.4 Circuitos aritméticos

Dentro do conjunto de circuitos combinacionais aplicados para finalidade específica nos sistemas digitais, destacam-se os circuitos aritméticos. São utilizados, principalmente, para construir a ULA (Unidade Lógica e Aritmética) dos microprocessadores e, ainda, encontrados disponíveis em circuitos integrados comerciais. Este tópico aborda os principais circuitos aritméticos e seus subsistemas derivados.

### 3.4.1 Meio Somador

Antes de iniciarmos o assunto, vamos verificar a operação de soma entre dois números binários:

A diagram showing the addition of two binary numbers. On the left, there are two rows of digits: the top row has digits 0, 0, 1 and the bottom row has digits + 0, + 1. Below these are the results: 0 and 1. To the right of the digits is a vertical column of additions: 0 + 0 = 0, 0 + 1 = 1, 0 + 0 = 0, and 0 + 1 = 1. Above the result 1 is a small box containing '11' with a downward arrow pointing to it. Below the result 10 is a bracket labeled 'transporte' (carry) with a line pointing to the digit 1 in the result.

Figura 3.15 - Adição de 2 bits.

Após esta breve introdução, vamos montar uma tabela-verdade da soma de dois números binários de um algarismo:

Tabela 3.9 - Soma de 2 números binários

A	B	S	Ts	Ts → transporte de saída
0	0	0	0	(0 + 0 = 0 → Ts = 0)
0	1	1	0	(0 + 1 = 1 → Ts = 0)
1	0	1	0	(1 + 0 = 1 → Ts = 0)
1	1	0	1	(1 + 1 = 0 → Ts = 1)

Representando cada número por 1 bit, podemos montar um circuito que possui como entradas A e B, e como saída, a soma dos algarismos (S) e o respectivo transporte de saída (Ts). As expressões características do circuito, extraídas da tabela, são:

$$S = A \oplus B$$

$$Ts = AB$$

O circuito a partir destas expressões é ilustrado na Figura 3.16.

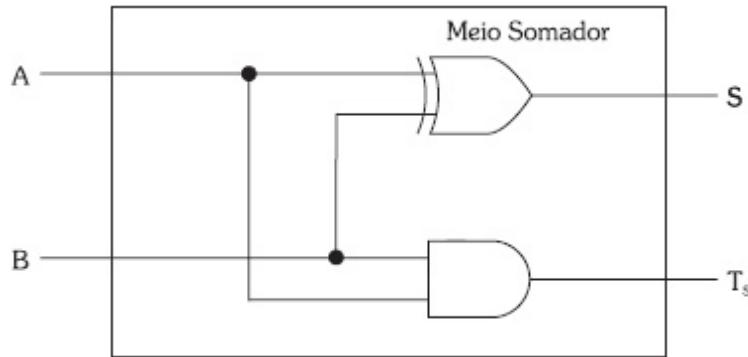


Figura 3.16 - Meio Somador.

A representação em bloco desse circuito é vista na Figura 3.17.

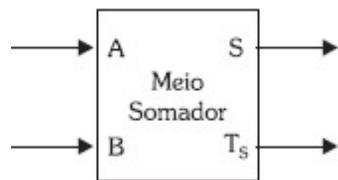


Figura 3.17 - Bloco Meio Somador.

Esse circuito Meio Somador é também conhecido como *Half Adder*, sendo a saída de transporte denominada *carry out*, ambos os termos derivados do inglês.

### 3.4.2 Somador Completo

O Meio Somador possibilita efetuar a soma de números binários com um algarismo. Para fazer a soma de números binários de mais algarismos, esse circuito torna-se insuficiente, pois não possibilita a introdução do transporte de entrada proveniente da coluna anterior. Para melhor compreensão, vamos analisar o caso da soma  $1110_2 + 110_2$ . Assim sendo, temos:

The diagram illustrates the binary addition of  $1110_2 + 110_2$ . The numbers are aligned vertically by their least significant bits. The result is  $10010_2$ . Below the addition, three arrows point to the third column from the left, indicating the propagation of the carry bit. The carry bit is labeled  $T_s = 1$  under each arrow.

Figura 3.18 - Exemplo de soma em binário.

A coluna 1 tem como resultado um transporte de saída igual a 0. A coluna 2 tem como resultado 0 e um transporte de saída igual a 1. A coluna 3 tem um transporte de entrada igual a 1 (Ts da coluna anterior), possui resultado 1 e transporte de saída igual a 1. A coluna 4 tem transporte de entrada igual a 1, resultado 0 e transporte de saída 1. A coluna 5 possui apenas um transporte de entrada (Ts da coluna 4) e, obviamente, seu resultado será igual a 1.

Para fazermos a soma de dois números binários de mais algarismos, basta somarmos coluna a coluna, levando em conta o transporte de entrada que nada mais é do que o Ts da coluna anterior.

O Somador Completo é um circuito para efetuar a soma completa de uma coluna, considerando o transporte de entrada. Vamos, agora, montar a tabela-verdade desse circuito:

Tabela 3.10 - Tabela-verdade de um Somador Completo

A	B	T <sub>E</sub>	S	T <sub>S</sub>	T <sub>E</sub> → transporte de entrada
0	0	0	0	0	(0 + 0 + 0 = 0 → Ts = 0)
0	0	1	1	0	(0 + 0 + 1 = 1 → Ts = 0)
0	1	0	1	0	(0 + 1 + 0 = 1 → Ts = 0)
0	1	1	0	1	(0 + 1 + 1 = 0 → Ts = 1)
1	0	0	1	0	(1 + 0 + 0 = 1 → Ts = 0)
1	0	1	0	1	(1 + 0 + 1 = 0 → Ts = 1)
1	1	0	0	1	(1 + 1 + 0 = 0 → Ts = 1)
1	1	1	1	1	(1 + 1 + 1 = 1 → Ts = 1)

Vamos escrever as expressões características, sem simplificação, de um Somador Completo:

$$S = \overline{A}\overline{B}T_E + \overline{A}B\overline{T}_E + A\overline{B}\overline{T}_E + ABT_E$$

$$Ts = \overline{A}BT_E + A\overline{B}T_E + AB\overline{T}_E + ABT_E$$

Transpondo  $S$  e  $T_s$  para os mapas de Karnaugh, temos:

	$\bar{B}$		$B$
$\bar{A}$	0	(1)	0
A	(1)	0	(1)
	$\bar{T}_E$	$T_E$	$\bar{T}_E$

Figura 3.19 - Mapa da saída S.

Verificamos que há apenas termos isolados, não havendo possibilidade de simplificação.

Extraindo a expressão da tabela inicial ou do diagrama, temos:

$$S = AB\bar{T}_E + \bar{A}\bar{B}T_E + ABT_E + \bar{A}\bar{B}\bar{T}_E$$

Evidenciando A e A, temos:

$$S = \bar{A}(\bar{B}T_E + B\bar{T}_E) + A(\bar{B}\bar{T}_E + BT_E)$$

Substituindo os parênteses respectivamente por  $B \oplus T_E$  e  $B \odot T_E$ , temos:

$$S = \bar{A}(B \oplus T_E) + A(B \odot T_E)$$

Como  $B \odot T_E = \overline{B \oplus T_E}$ , reescrevemos:

$$S = \bar{A}(B \oplus T_E) + A(\overline{B \oplus T_E})$$

Chamando  $(B \oplus T_E)$  de X, obtém-se:  $S = \bar{A}X + A\bar{X} = A \oplus X$

Substituindo X, temos:  $S = A \oplus B \oplus T_E$

	$\bar{B}$		$B$
$\bar{A}$	0	0	(1)
A	0	(1)	(1)
	$\bar{T}_E$	$T_E$	$\bar{T}_E$

$$T_S = BT_E + AT_E + AB$$

Figura 3.20 - Mapa da saída  $T_S$ .

Vamos, pelas expressões, esquematizar o circuito Somador Completo:

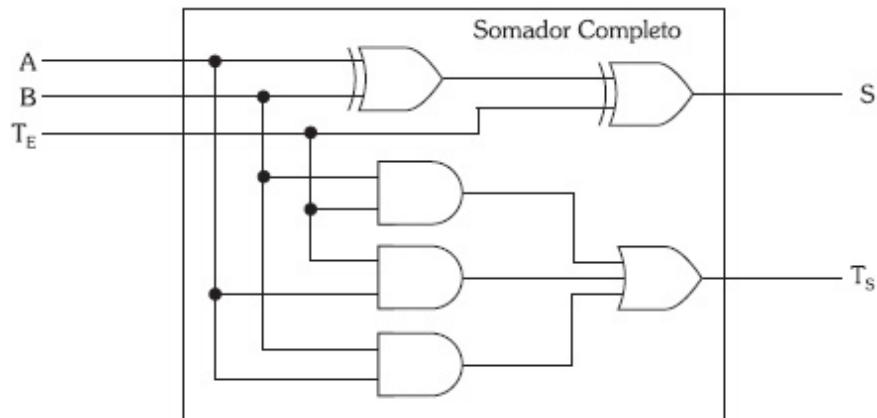


Figura 3.21 - Somador Completo.

Da mesma forma, o circuito apresentado em bloco é visto na Figura 3.22.

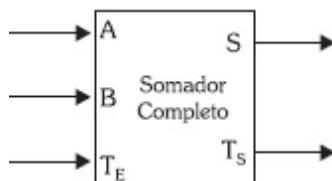


Figura 3.22 - Bloco Somador Completo.

O circuito Somador Completo é também conhecido como *Full Adder*, sendo a entrada de transporte denominada *carry in*, ambos os termos derivados do inglês.

Vamos, para exemplo de aplicação, montar um sistema em blocos que efetua a soma de dois números de 4 bits, conforme o esquema a seguir:

$$\begin{array}{r}
 & A_3 & A_2 & A_1 & A_0 \\
 + & B_3 & B_2 & B_1 & B_0 \\
 \hline
 S_4 & S_3 & S_2 & S_1 & S_0
 \end{array}$$

Para efetuar a soma dos bits A<sub>0</sub> e B<sub>0</sub> dos números (1<sup>a</sup> coluna), vamos utilizar um Meio Somador, pois não existe transporte de

entrada, mas para as outras colunas utilizaremos Somadores Completos, pois precisamos considerar os transportes provenientes das colunas anteriores. O sistema montado está na Figura 3.23.

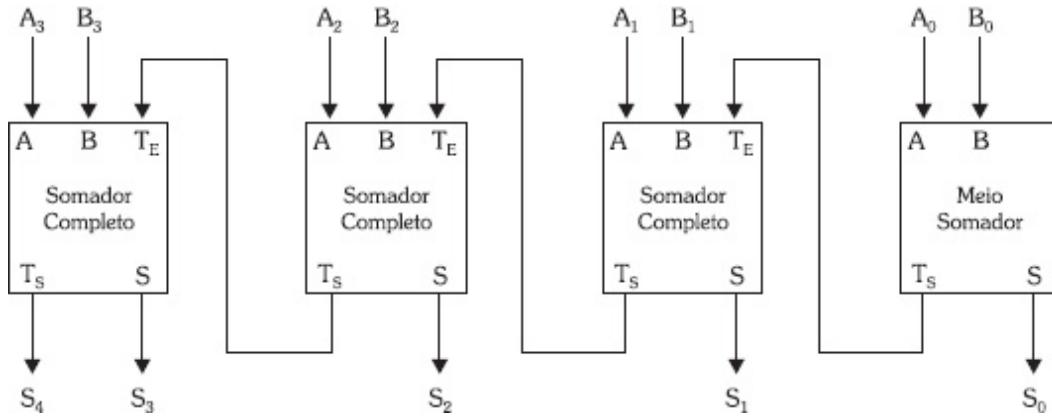


Figura 3.23 - Sistema Somador para 2 números de 4 bits.

Generalizando para um sistema que efetua a soma de dois números de  $m$  bits ( $m = n + 1$ ), temos:

$$\begin{array}{r}
 & A_n & A_{n-1} & \dots & A_1 & A_0 \\
 + & B_n & B_{n-1} & \dots & B_1 & B_0 \\
 \hline
 S_{n+1} & S_n & S_{n-1} & \dots & S_1 & S_0
 \end{array}$$

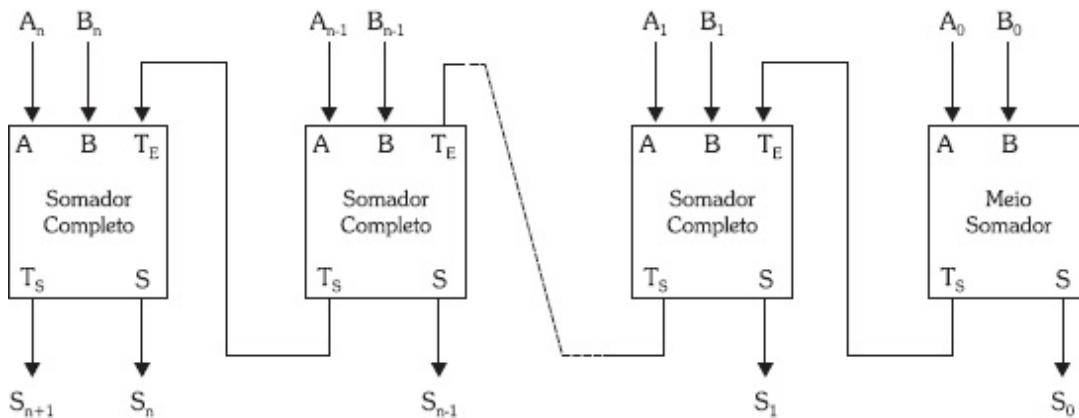


Figura 3.24 - Sistema Somador genérico.

### 3.4.3 Meio Subtrator

Antes de iniciar o assunto, vamos verificar a operação de subtração de números binários:

$$0 - 0 = 0$$

$$0 - 1 = 1 \text{ e transporta } 1 \text{ ("empresta" } 1)$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

Vamos montar a tabela-verdade de uma subtração de dois números binários de um algarismo:

Tabela 3.11 - Subtração de 2 números binários

A	B	S	Ts	
0	0	0	0	$(0 - 0 = 0 \rightarrow Ts = 0)$
0	1	1	1	$(0 - 1 = 1 \rightarrow Ts = 1)$
1	0	1	0	$(1 - 0 = 1 \rightarrow Ts = 0)$
1	1	0	0	$(1 - 1 = 0 \rightarrow Ts = 0)$

Representando cada número por 1 *bit*, podemos montar um circuito com as entradas A e B, e como saída, a subtração (S) e o transporte de saída (Ts).

As expressões características do circuito, extraídas da tabela, são:

$$\begin{aligned} S &= A \oplus B \\ Ts &= \overline{A}B \end{aligned}$$

O circuito a partir destas é mostrado na Figura 3.25.

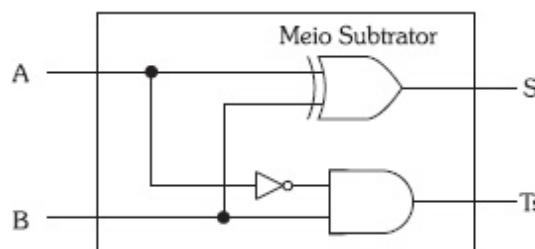


Figura 3.25 - Meio Subtrator.

Em bloco, o circuito recebe a representação da Figura 3.26.

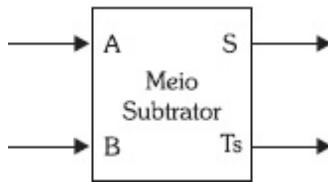


Figura 3.26 - Bloco Meio Subtrator.

Do inglês, o circuito recebe a denominação *Half Subtractor*.

### 3.4.4 Subtrator Completo

O Meio Subtrator possibilita efetuar a subtração de números binários de um algarismo. Para fazer uma subtração com números de mais algarismos, esse circuito torna-se insuficiente, pois não possibilita a entrada do transporte ( $T_E$ ), proveniente da coluna anterior.

Para compreendermos melhor, vamos analisar a subtração  $1100_2 - 11_2$ . Assim sendo, temos:

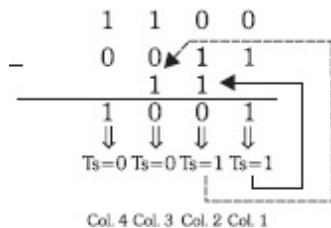


Figura 3.27 - Exemplo de subtração em binário.

A coluna 1 tem como resultado de saída 1 e apresenta um transporte de saída igual a 1. A coluna 2 tem um transporte de entrada igual a 1 ( $T_E$  da coluna anterior), um resultado igual a 0 e um  $T_E = 1$ . A coluna 3 tem  $T_E = 1$ , resultado igual a 0 e  $T_E = 0$ . A coluna 4 tem  $T_E = 0$ , resultado igual a 1 e  $T_E = 0$ .

Para fazermos a subtração de números binários de mais algarismos, basta subtrairmos coluna a coluna, levando em conta o transporte de entrada, que nada mais é do que o  $T_E$  da coluna anterior.

O Subtrator Completo é um circuito que efetua a subtração completa de uma coluna, ou seja, considera o transporte de entrada proveniente da coluna anterior. Vamos agora montar a tabela-verdade desse circuito:

Tabela 3.12 - Tabela-verdade de um Subtrator Completo

A	B	$T_E$	S	$T_S$
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

As expressões características extraídas da tabela são:

$$S = \overline{A}\overline{B}T_E + \overline{A}B\overline{T}_E + A\overline{B}T_E + ABT_E$$

$$T_S = \overline{A}\overline{B}T_E + \overline{A}B\overline{T}_E + \overline{A}BT_E + ABT_E$$

Vamos simplificar estas expressões:

S:

	$\overline{B}$	B	
$\overline{A}$	0	(1)	0
A	(1)	0	(1)
$\overline{T}_E$		$T_E$	$\overline{T}_E$

$T_S$ :

	$\overline{B}$	B	
$\overline{A}$	0	(1)	(1)
A	0	0	(1)
$\overline{T}_E$		$T_E$	$\overline{T}_E$

$$(a) S = A \oplus B \oplus T_E$$

$$(b) T_S = \overline{A}B + \overline{A}T_E + BT_E$$

Figura 3.28 - Diagramas com as simplificações.

O circuito derivado das expressões é visto na Figura 3.29.

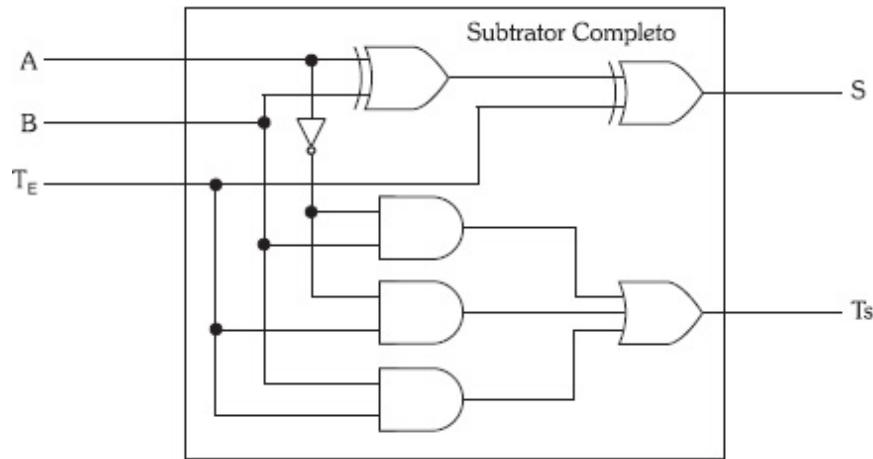


Figura 3.29 - Subtrator Completo.

Em bloco, recebe a representação da Figura 3.30.

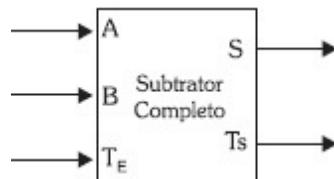


Figura 3.30 - Bloco Subtrator Completo.

A denominação derivada do inglês é *Full Subtractor*.

Da mesma forma, podemos esquematizar um sistema subtrator para dois números de  $m$  bits ( $m = n + 1$ ). A Figura 3.31 mostra um sistema subtrator genérico para dois números de  $m$  bits.

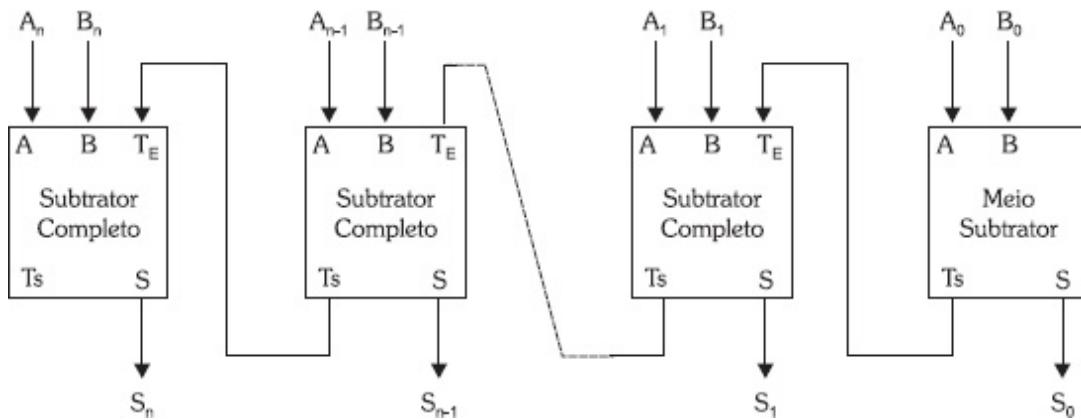


Figura 3.31 - Bloco subtrator genérico.

Nesse sistema, a saída de transporte ( $T_s$ ) do último bloco torna-se desnecessária se o número  $A_n...A_0$  (minuendo) for maior ou igual a  $B_n...B_0$  (subtraendo).



## Exercícios resolvidos

- 3.1) Desenhe um sistema somador para dois números de 2 *bits* apenas com blocos de Somadores Completos.

Para obter esse sistema, necessitamos de um Meio Somador e um Somador Completo. A solução é obtida aplicando nível 0 (terra) à entrada de transporte do ( $T_E$ ) do somador relativo ao *bit* menos significativo, transformando-o em Meio Somador, pois essa entrada fica eliminada. A Figura 3.32 apresenta esse sistema composto apenas de Somadores Completos.

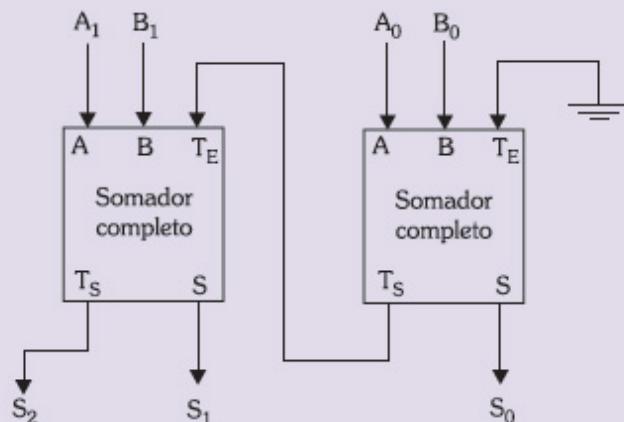


Figura 3.32 - Sistema Somador composto apenas com Somadores Completos.

- 3.2) Esquematize, em blocos, um sistema subtrator para dois números com 2 *bits*.

O sistema proposto realiza a subtração do número  $A_1A_0$  com o número  $B_1B_0$ . Assim sendo, temos:

$$\begin{array}{r}
 & A_1 & A_0 \\
 - & B_1 & B_0 \\
 \hline
 S_1 & & S_0
 \end{array}$$

Para a primeira coluna da operação, vamos utilizar um Meio Subtrator, pois não há transporte de entrada. Para a segunda coluna, porém, utilizamos um Subtrator Completo, pois ele possui entrada para o *bit* proveniente da coluna anterior. O circuito, assim esquematizado, está na Figura 3.33.

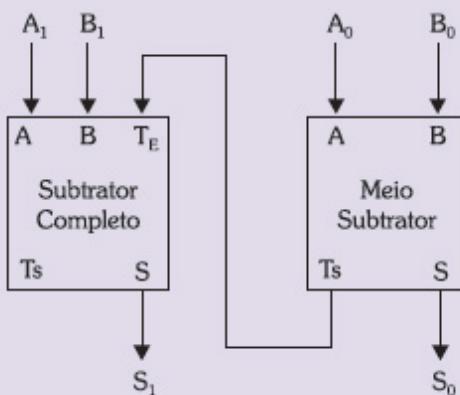


Figura 3.33 - Sistema Subtrator para dois números com 2 bits.

### Vamos recapitular?

Neste capítulo, você aprendeu como se desenvolve o projeto de um circuito combinacional para aplicações específicas.

Compreendeu o funcionamento de codificadores, decodificadores e circuitos aritméticos básicos, obtidos pelo desenvolvimento de circuitos combinacionais.

Conheceu também as formas de acender os *displays* de 7 segmentos para mostrar uma sequência de caracteres, a partir de um código binário.

Utilizando esses conceitos, você poderá entender o modo de operação de outros circuitos com aplicações em sistemas digitais e, ainda, na arquitetura interna de computadores.



## Agora é com você!

- 1) Elabore um codificador decimal/binário para, a partir de um teclado com chaves numeradas de 0 a 3, fornecer nas saídas o código correspondente. Considere que as entradas das portas em vazio equivalem à aplicação de nível lógico 1.
- 2) Projete um circuito combinacional para, em um conjunto de quatro fios, fornecer nível 0 em apenas um deles por vez (estando os demais em nível 1), conforme seleção binária aplicada às entradas digitais.
- 3) Elabore um decodificador 3 para 8 em que, conforme as combinações entre os três fios de entrada, um entre os oito fios de saída é ativado (nível 1).
- 4) Projete um decodificador para, a partir de um código binário, escrever a sequência de 1 a 5 em um *display* de 7 segmentos catodo comum.
- 5) Idem ao anterior, para escrever a sequência da Figura 3.34 em um *display* de 7 segmentos. Considere que o *display* seja do tipo anodo comum (o segmento acende com nível 0 aplicado).

Caractere	C	d	P	L	A	Y	E	r
Caso	0	1	2	3	4	5	6	7

Figura 3.34 - Sequência do display do exercício 5.

- 6) Qual é a diferença fundamental entre um Meio Somador e um Somador Completo?
- 7) Mostre como um bloco Somador Completo pode ser utilizado para efetuar a soma de três números de 1 *bit*.
- 8) Esquematize, em blocos, um Sistema Somador para 2 números de 2 *bits*.
- 9) Esquematize, em blocos, um Sistema Subtrator para dois números de 4 *bits*.

# 4

## Circuitos Sequenciais

**Para começar**

Dedicaremos este capítulo ao estudo dos circuitos sequenciais que se caracterizam por ter as saídas dependendo das variáveis de entrada e, ainda, dos próprios estados anteriores das saídas que podem realimentar as entradas. São sistemas que trabalham sob regime de tempo, sendo acionados sequencialmente por pulsos, ou seja, por variação de tensão elétrica no tempo.

Esta categoria de circuitos desempenha um papel essencial na arquitetura interna dos sistemas computadorizados. São compostos por contadores, registradores, memórias eletrônicas e outros sistemas que atuam de maneira pulsada.

### 4.1 Definição geral de circuitos sequenciais

Os circuitos sequenciais têm as saídas dependentes das variáveis de entrada e/ou de seus estados anteriores que permanecem armazenados, sendo, geralmente, sistemas pulsados, ou seja, operam sob comando de uma sequência de pulsos denominada *clock*.

Na lógica sequencial, os *flip-flops* são os principais elementos, devidamente interligados, formam contadores síncronos e assíncronos, registradores de deslocamento e memórias eletrônicas.

Vamos estudar os principais tipos de *flip-flops* existentes. Para entendimento das estruturas básicas desses elementos, vamos analisar primeiramente o circuito interno do *flip-flop* RS básico, composto por portas lógicas já conhecidas. Em seguida será colocada a entrada *clock*, que faz o controle do sistema, permitindo sua atuação no regime do tempo.

## 4.2 *Flip-flops*

---

De forma geral, podemos representar o *flip-flop* como um bloco com duas saídas: Q e  $\bar{Q}$ , entradas para as variáveis e uma entrada de controle (*clock*). A saída Q é a principal do bloco. A Figura 4.1 ilustra um *flip-flop* genérico.

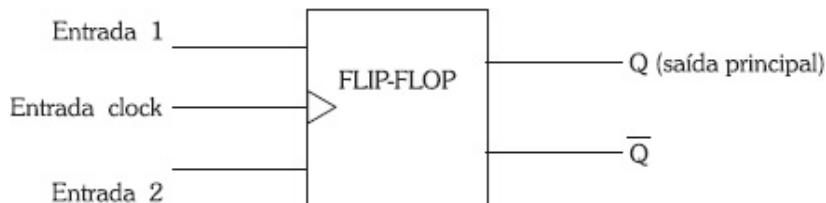


Figura 4.1 - *Flip-flop* genérico com entradas e saídas.

Esse dispositivo possui basicamente dois estados de saída. Para o *flip-flop* assumir um desses estados, é necessário que haja uma combinação das variáveis e do pulso de controle (*clock*). Após esse pulso, o *flip-flop* permanece neste estado até a chegada de um novo pulso de *clock* e, de acordo com as variáveis de entrada, muda ou não de estado.

Os dois estados possíveis são:  $Q = 0 \rightarrow \bar{Q} = 1$  ou  $Q = 1 \rightarrow \bar{Q} = 0$

Vamos, a seguir, analisar alguns circuitos de *flip-flops* e suas respectivas operações.

### 4.2.1 *Flip-flop* RS básico

Primeiramente, vamos analisar o *flip-flop* RS básico, construído a partir de portas NE e inversores, cujo circuito é visto na Figura 4.2.

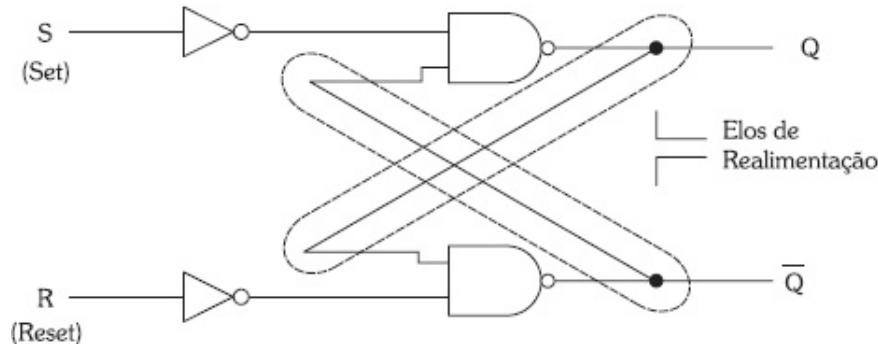


Figura 4.2 - Circuito básico do *flip-flop* RS.

Notamos que os elos de realimentação fazem com que as saídas sejam injetadas juntamente com as variáveis de entrada, ficando claro que os estados que as saídas vão assumir dependem de ambas.

Para analisarmos o comportamento do circuito, vamos construir a tabela-verdade, levando em consideração as duas variáveis de entrada (S e R) e a saída Q anterior (Qa) à aplicação das entradas:

Tabela 4.1 - Tabela-verdade para levantamento dos estados futuros

	S	R	Qa	Qf
0	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	

→ Estado anterior da saída Q.  
→ Estado que a saída deve assumir (estado futuro) após a aplicação das entradas.

A saída que o *flip-flop* vai assumir (Qf), portanto, será em função das entradas S, R e da saída anterior (Qa).

Vamos, agora, analisar cada caso possível:

Caso 0:  $S = 0$ ,  $R = 0$  e  $Q_a = 0 \rightarrow \overline{Q_a} = 1$

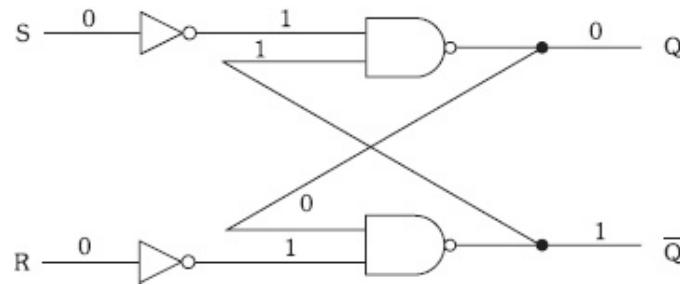


Figura 4.3

Podemos notar que este estado é estável, logo, o valor que a saída Q vai assumir será igual ao seu valor anterior à aplicação das entradas:  $Q_f = Q_a = 0$

Caso 1:  $S = 0$ ,  $R = 0$  e  $Q_a = 1 \rightarrow s \overline{Q_a} = 0$

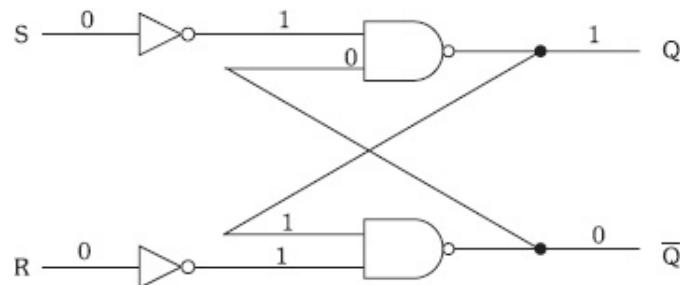


Figura 4.4

Este também será um estado estável, logo, o valor que a saída Q do *flip-flop* vai assumir será igual ao seu valor anterior:  $Q_f = Q_a = 1$

Caso 2:  $S = 0$ ,  $R = 1$  e  $Q_a = 0 \rightarrow \overline{Q_a} = 1$

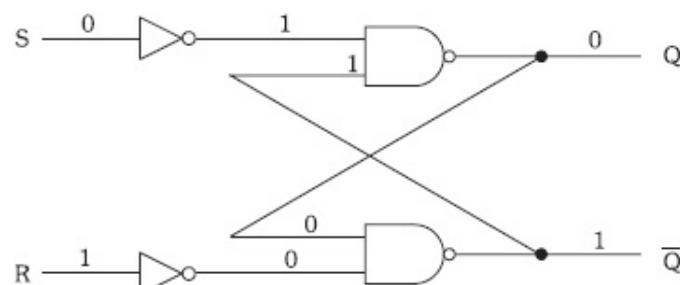


Figura 4.5

Este estado é estável, logo, Q vai assumir o valor 0:  $Q_f = 0$ .

Caso 3:  $S = 0$ ,  $R = 1$  e  $Q_a = 1 \rightarrow \overline{Q_a} = 0$

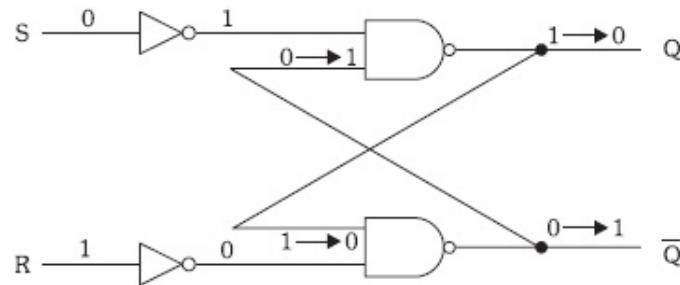


Figura 4.6

Notamos, agora, que a saída  $Q$  está em um estado instável, pois  $\overline{Q}$  vai mudar para 1, forçando assim que  $Q$  assuma valor 0 e aí sim, temos um estado estável, logo, podemos escrever para este caso  $Q_f = 0$  (pois  $Q$  vai assumir valor 0).

Caso 4:  $S = 1$ ,  $R = 0$  e  $Q_a = 0 \rightarrow \overline{Q_a} = 1$

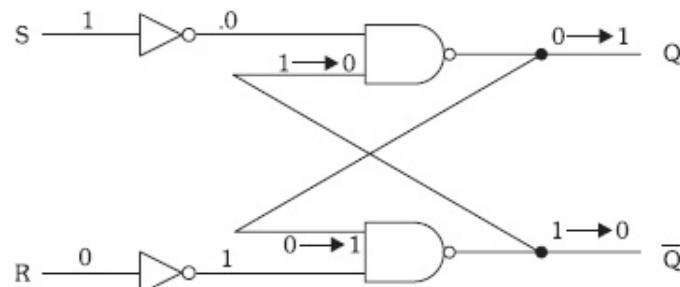


Figura 4.7

Notamos que este é um estado instável, pois  $Q$  vai assumir forçosamente valor 1 e, por conseguinte,  $\overline{Q}$  assumirá valor 0, logo, podemos escrever  $Q_f = 1$ .

Caso 5:  $S = 1$ ,  $R = 0$  e  $Q_a = 1 \rightarrow s \overline{Q_a} = 0$

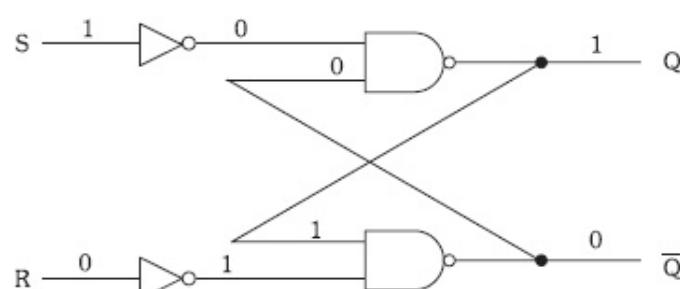


Figura 4.8

Notamos que este é um estado estável, logo, podemos escrever para este caso  $Q_f = 1$ .

Caso 6:  $S = 1$ ,  $R = 1$  e  $Q_a = 0 \rightarrow \overline{Q_a} = 1$

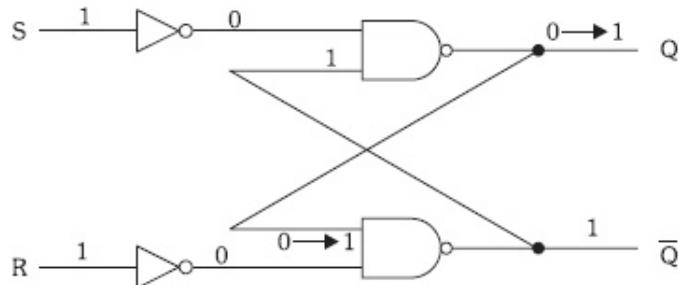
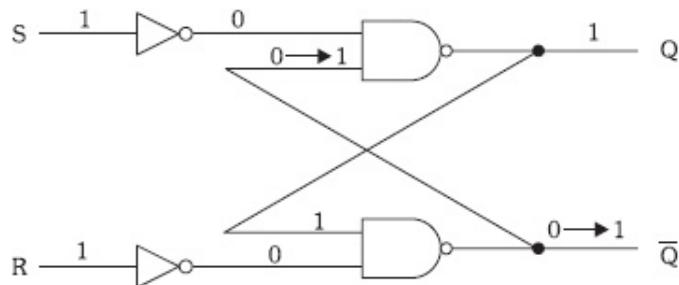


Figura 4.9

Este é um estado instável, pois  $Q$  forçosamente vai assumir valor 1. Notamos, também, que  $\overline{Q}$  vai assumir valor 1. Podemos escrever para este caso  $Q_f = \overline{Q}_f = 1$ .

Este caso não pode ser permitido na entrada, pois força o *flip-flop* a assumir um estado de saída, no qual a saída  $Q$  será igual à saída complementar  $\overline{Q}$ .

Caso 7:  $S = 1$ ,  $R = 1$  e  $Q_a = 1 \rightarrow \overline{Q_a} = 0$



Figuras 4.3 a 4.10 - Análise dos casos no *flip-flop* RS básico.

Notamos que esta é uma situação instável e análoga ao caso 6, logo, também será uma situação não permitida.

A Tabela 4.2 apresenta o resultado da análise de todos esses casos e a Tabela 4.3 os casos resumidos.

Tabela 4.2 - Tabela-verdade do *flip-flop* RS básico

S	R	Qa	Qf	Qf	
0	0	0	0	1	Fixa Qf = Qa
0	0	1	1	0	
0	1	0	0	1	Fixa Qf em 0
0	1	1	0	1	
1	0	0	1	0	Fixa Qf em 1
1	0	1	1	0	
1	1	0	1	1	
1	1	1	1	1	Não permitido

Tabela 4.3 - Tabela-verdade resumida

S	R	Qf
0	0	Qa
0	1	0
1	0	1
1	1	X

A entrada S é denominada *Set*, pois quando acionada (nível 1), passa a saída para 1 (estabelece ou fixa 1), e a entrada R é denominada *Reset*, pois quando acionada (nível 1), passa a saída para 0 (recompõe ou zera o *flip-flop*). Estes termos são muito usuais, sendo provenientes do idioma inglês.

Esse circuito vai mudar de estado apenas no instante em que mudam as variáveis de entrada. Veremos em seguida como é o circuito de um *flip-flop RS* que tem sua mudança de estado controlada pela entrada de *clock*.

#### 4.2.2 *Flip-flop RS* com entrada *clock*

Para que o *flip-flop RS* básico seja controlado por uma sequência de pulsos de *clock*, basta trocarmos os dois inversores por portas NE, e às outras entradas dessas portas, injetarmos o *clock*. O circuito, com essas modificações, é ilustrado na Figura 4.11.

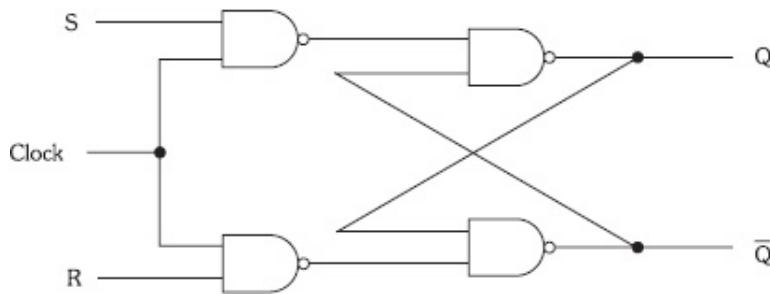


Figura 4.11 - *Flip-flop RS* com entrada *clock*.

Neste circuito, quando a entrada de *clock* for igual a 0, o *flip-flop* vai permanecer no seu estado, mesmo que variem as entradas S e R. Isso pode ser confirmado pela análise do circuito, pois para *clock* = 0, as saídas das portas NE de entrada serão sempre iguais a 1, independentemente dos valores assumidos por S e R.

Quando a entrada *clock* assumir valor 1, o circuito vai comportar-se como um *flip-flop RS* básico, pois as portas NE de entrada funcionarão como os inversores do circuito anteriormente visto. A Tabela 4.4 resume a operação desse *flip-flop* em função da entrada *clock*.

Tabela 4.4 - Tabela da atuação do *clock*

CK	Qf
0	Qa
1	RS básico

De maneira geral, podemos concluir que o circuito vai funcionar quando a entrada *clock* assumir valor 1 e manterá travada essa saída quando a entrada *clock* passar para 0. O *flip-flop RS* pode ser representado pelo bloco da Figura 4.12.

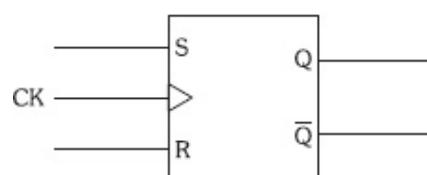


Figura 4.12 - Bloco do *flip-flop RS* com entrada *clock*.

#### 4.2.3 *Flip-flop JK mestre-escravo*

Como vimos *flip-flop* RS apresenta uma situação não permitida quando as entradas S e R assumirem simultaneamente o valor 1. O *flip-flop* JK Mestre-Escravo (*JK Master-Slave*), cujo circuito é apresentado na Figura 4.13, soluciona a indeterminação possibilitando a utilização desse caso, como veremos na análise do circuito.

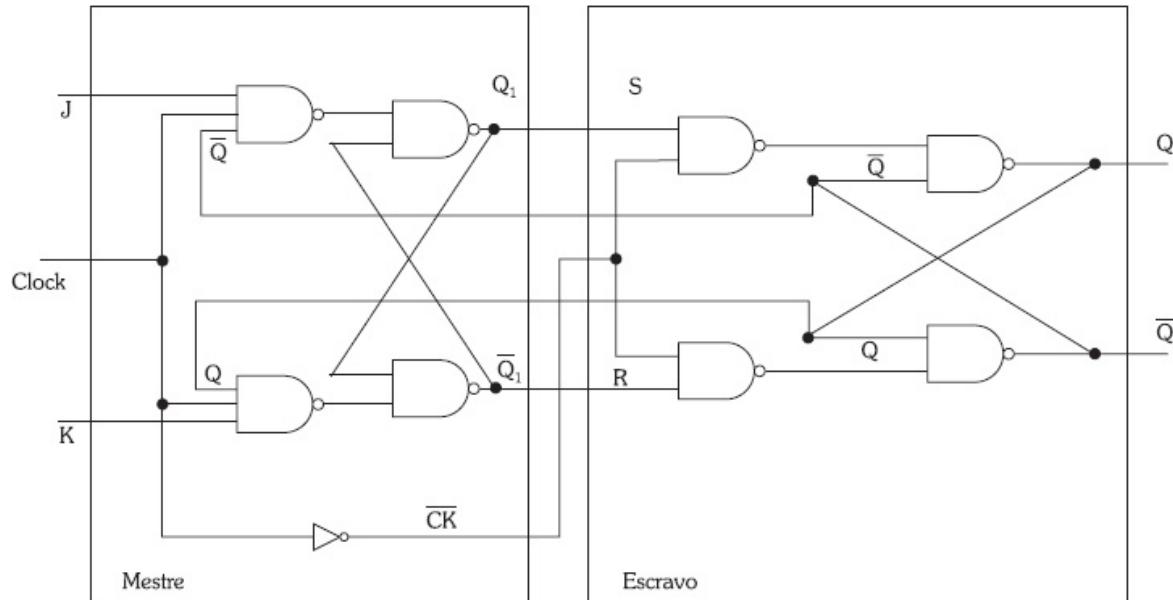


Figura 4.13 - *Flip-flop* JK Mestre-Escravo.

Primeiramente, devemos notar que quando o *clock* for igual a 1, há a passagem das entradas J e K (circuito mestre), porém não há passagem das saídas  $Q_1$  e  $\bar{Q}_1$  (entradas S e R do circuito escravo), enquanto o *clock* do circuito mestre for igual a 1, no circuito escravo será 0, bloqueando suas entradas. Quando o *clock* passar para 0, as saídas  $Q_1$  e  $\bar{Q}_1$  ficam bloqueadas no último estado assumido e entram em R e S desbloqueadas, mudando o estado do circuito escravo e, consequentemente, das saídas Q e  $\bar{Q}$ .

Enquanto o *clock* permanece em 0, notamos que J e K podem variar à vontade que o *flip-flop* mantém a saída constante, pois  $Q_1$  e  $\bar{Q}_1$  permanecem fixos. No momento em que o *clock* passa para 1, os pontos  $Q_1$  e  $\bar{Q}_1$  mudam de estado conforme as entradas J e K, porém a saída Q permanece constante, pois a entrada de *clock* do circuito escravo ( $\bar{CK}$ ) fica em 0. O circuito mestre assume o estado que for imposto pelas entradas J e K no momento em que o *clock* mudar para 0, permanecendo nesse estado até que o *clock* volte a mudar. A saída assumida pelo

círcuito mestre vai impor ao circuito escravo o seu estado, e este só vai mudar na próxima vez em que o *clock* mudar de 1 para 0.

No caso  $J = 1$  e  $K = 1$ , obtém-se  $Q_f = \bar{Q}a$ , conforme mostra o circuito da Figura 4.14, em razão da influência dos fios de realimentação aplicados ao bloco mestre provenientes das saídas  $Q$  e  $\bar{Q}$ , haverá mudança dessas saídas provocando a inversão de  $Q$  e  $\bar{Q}$ . Na descida de *clock*, por atuação do bloco escravo, esta inversão será passada às saídas do *flip-flop*.

A Tabela 4.5 resume a operação do *flip-flop* JK Mestre-Escravo:

Tabela 4.5 - Tabela resumida

J	K	$Q_f$
0	0	$Qa$
0	1	0
1	0	1
1	1	$\bar{Q}a$

A saída  $Q$  vai assumir valores, conforme a situação das entradas JK, somente após a passagem do *clock* para 0. Assim sendo, o circuito se denomina JK Mestre-Escravo **sensível à descida de *clock***. Para obter um circuito **sensível à subida de *clock***, basta colocarmos um inversor interno à entrada *clock*.

A Figura 4.14 mostra o bloco JK Mestre-Escravo e a simbologia para identificar o circuito sensível à descida de *clock* (a) e à subida de *clock* (b).

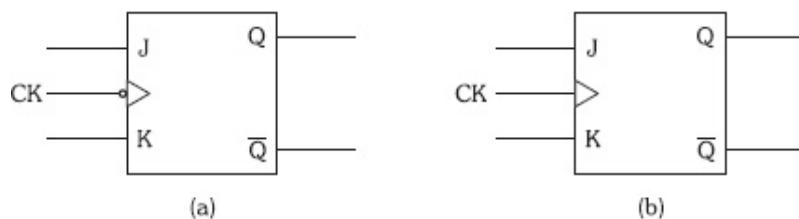


Figura 4.14 - (a) Entrada *clock* sensível à descida. (b) Entrada *clock* sensível à subida.

O círculo no bloco da Figura 4.14 (a) indica que o *clock* é ativo quando passa de 1 para 0.

#### 4.2.4 Flip-flop JK mestre-escravo com entradas Preset e Clear

O *flip-flop* JK pode assumir valores  $Q = 1$  ou  $Q = 0$  mediante utilização das entradas *Preset* (PR) e *Clear* (CLR). Essas entradas são inseridas no circuito, conforme mostra a Figura 4.15.

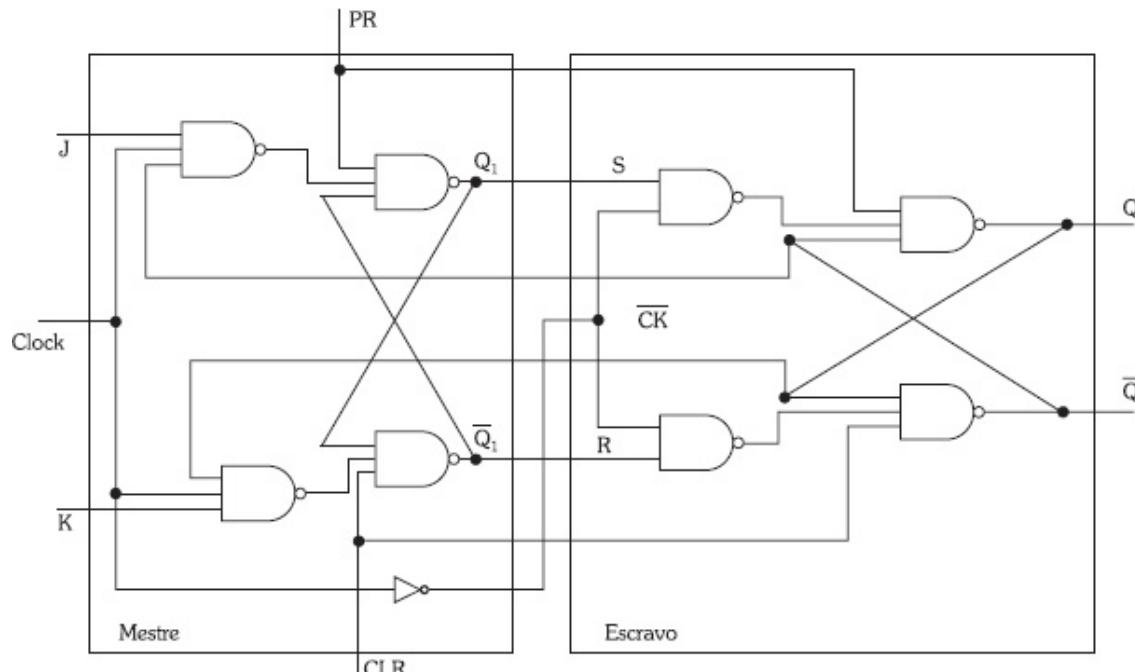


Figura 4.15 - JK Mestre-Escravo com *Preset* e *Clear*.

O controle de *Preset*, quando assumir valor 0, faz com que a saída do circuito ( $Q$ ) assuma valor 1. O mesmo ocorre com o controle de *Clear*, fazendo com que a saída assuma valor 0. Notamos que ambos, por estarem ligados simultaneamente aos circuitos Mestre e Escravo, atuam independentemente da entrada *clock*. As entradas *Preset* e *Clear* não podem assumir valor 0, simultaneamente, pois isso acarretaria à saída uma situação não permitida. Todas as situações possíveis são descritas na Tabela 4.6.

A entrada *Clear* é também denominada *Reset*, termo este, da mesma forma que os outros, derivado do inglês. A Figura 4.16 mostra o bloco representativo do *flip-flop* JK Mestre-Escravo com as entradas *Preset* e *Clear* ativas em 0.

Tabela 4.6 - Atuação do *Preset* e *Clear*

CLR	PR	Qf
0	0	Não permitido
0	1	0
1	0	1
1	1	Funcionamento normal

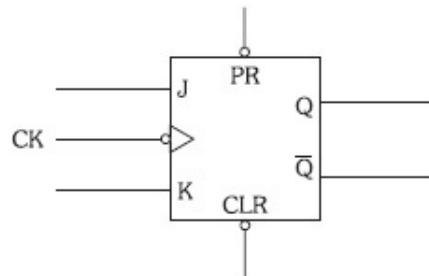


Figura 4.16 - Bloco do *flip-flop* JK Mestre-Escravo com *Preset* e *Clear*.

#### 4.2.5 *Flip-Flop T*

Esse *flip-flop* é obtido a partir de um JK Mestre-Escravo com as entradas J e K curto-circuitadas (uma ligada à outra); logo, quando J assumir valor 1, K também assume valor 1, e quando J assumir valor 0, K também assume valor 0. Obviamente, no caso desta ligação, não vão ocorrer nunca entradas como J = 0 e K = 1; J = 1 e K = 0. A Figura 4.17 mostra a ligação e o bloco representativo do *flip-flop* T obtido.

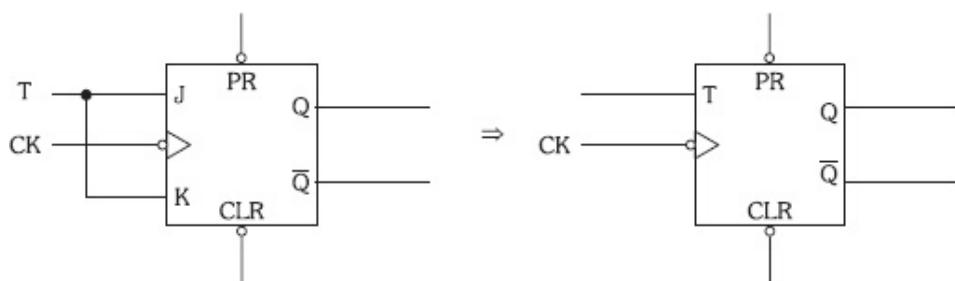


Figura 4.17 - *Flip-flop* configurado como T e bloco representativo.

A tabela-verdade completa será:

Tabela 4.7 - Tabela-verdade do *flip-flop* T completa

J	K	T	Qf
0	0	0	Qa
0	1	não existe	/
1	0	não existe	/
1	1	1	$\bar{Q}a$

Eliminando os casos não existentes, obtemos a tabela-verdade do *flip-flop T*.

Devido ao fato de o *flip-flop T*, com a entrada T igual a 1, complementar a saída ( $\bar{Q}a$ ) a cada descida de *clock*, este será utilizado como célula principal dos Contadores Assíncronos que serão estudados adiante. A sigla **T** vem de **Toggle** (comutado).

Tabela 4.8 - Tabela verdade do *flip-flop T*

T	Qf
0	Qa
1	$\bar{Q}a$

#### Lembre-se

O *flip-flop T* não é encontrado na série de circuitos integrados comerciais, sendo na prática montado a partir de um JK Mestre-Escravo, conforme a ligação apresentada.

#### 4.2.6 *Flip-flop D*

É obtido a partir de um *flip-flop JK* Mestre-Escravo com a entrada K invertida (por inverter) em relação a J. Logo, neste *flip-flop*, temos as seguintes entradas possíveis: J = 0 e K = 1; J = 1 e K = 0. Obviamente, não vão ocorrer os casos J = 0 e K = 0; J = 1 e K = 1. A Figura 4.18 mostra como ele é obtido e seu bloco representativo.

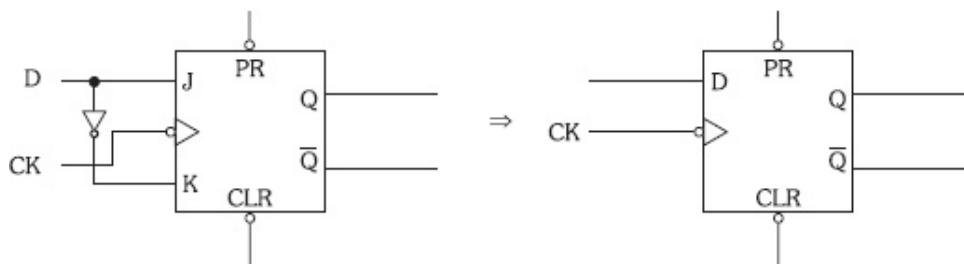


Figura 4.18 - *Flip-flop* configurado como D e bloco representativo.

A Tabela 4.9 apresenta a tabela-verdade completa e eliminando os casos não existentes, obtemos a Tabela 4.10.

Tabela 4.9 - Tabela-verdade completa

J	K	D	Qf
0	0	não existe	/
0	1	0	0
1	0	1	1
1	1	não existe	/

Tabela 4.10 - Tabela-verdade do *flip-flop* D

D	Qf
0	0
1	1

Pela capacidade de passar para a saída (Qf) e armazenar o dado aplicado na entrada D, esse *flip-flop* será empregado como célula de Registradores de Deslocamento e em outros sistemas de memória, a serem estudados adiante. A sigla D vem de *Data (dado)*, termo original em inglês.



## Exercícios resolvidos

- 1) Levante a tabela-verdade do *flip-flop* da Figura 4.19 e identifique as entradas S e R.

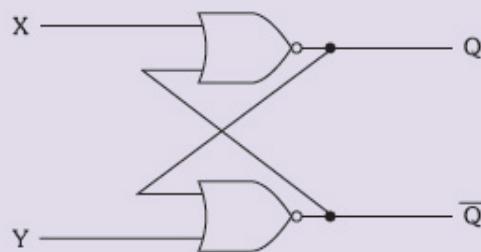


Figura 4.19 - *Flip-flop* a ser analisado.

Para solucionar, vamos levantar a tabela em função das entradas X e Y:

<b>1º)</b> X = 0 e Y = 0:	Para $Q_a = 0$ ( $\bar{Q}_a = 1$ ), a saída não se alterará ( $Q_f = 0$ ), pois os níveis são concordantes com a operação das portas NOU envolvidas, sendo uma situação estável. O mesmo ocorre com $Q_a = 1$ ( $\bar{Q}_a = 0$ ), em que $Q_f = 1$ .
<b>2º)</b> X = 0 e Y = 1:	O nível 1 aplicado em Y faz com que a respectiva porta NOU apresente saída igual a 0 ( $\bar{Q}_f = 0$ ) e, consequentemente, $Q_f = 1$ , pois o nível 0 de $\bar{Q}_f$ é aplicado juntamente ao nível 0 da entrada X, na outra porta NOU, forçando a saída $Q_f = 1$ , independente dos estados de saída anteriores.
<b>3º)</b> X = 1 e Y = 0:	Esse caso será exatamente oposto ao anterior, pois a aplicação do nível 1 em X faz com que a saída assuma valor 0 ( $Q_f = 0$ ), independente das situações das saídas antes da aplicação das entradas.
<b>4º)</b> X = 1 e Y = 1:	Esses níveis aplicados forçam ambas as saídas a assumir valor 0 ( $Q_f = \bar{Q}_f = 0$ ), caracterizando-se um caso não permitido.

A Tabela 4.11 apresenta todos os resultados, conforme a análise efetuada.

De posse da tabela, concluímos que a entrada X, quando ativa (nível 1), fixa a saída em 0, tendo a função de Reset ( $X = R$ ), e a entrada Y, quando ativa (nível 1), fixa a saída em 1, tendo a função de Set ( $Y = S$ ). O *flip-flop* com as entradas identificadas é encontrado na Figura 4.20.

Tabela 4.11 - Tabela resumida com os resultados

X	Y	Qf
0	0	$Q_a$
0	1	1
1	0	0
1	1	

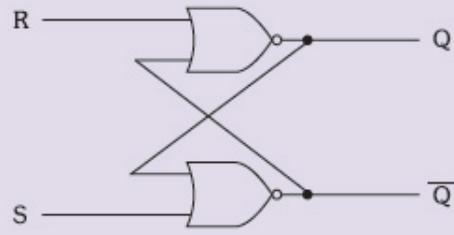


Figura 4.20 - *Flip-flop* com as entradas identificadas.

- 2) Complete a tabela, referente à saída Q do *flip-flop* JK/Mestre-Escravo da Figura 4.21, em função das entradas aplicadas.

<i>Clock</i>	<i>J</i>	<i>K</i>	<i>Preset</i>	<i>Clear</i>	<i>Qa</i>	<i>Qf</i>
$1 \rightarrow 0$	0	0	1	1	1	
$1 \rightarrow 0$	0	1	1	1	1	
1	1	1	1	0	1	
0	1	0	0	1	0	
$1 \rightarrow 0$	1	0	1	1	0	
$1 \rightarrow 0$	0	1	1	1	1	
$0 \rightarrow 1$	0	1	1	0	1	
$0 \rightarrow 1$	1	1	0	1	1	

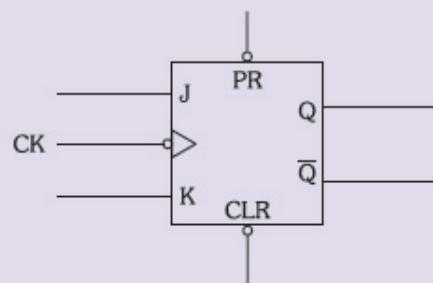


Figura 4.21 - *Flip-flop* e tabela de entradas aplicadas.

Tabela 4.12 - Tabela com os resultados

<i>Clock</i>	<i>J</i>	<i>K</i>	<i>Preset</i>	<i>Clear</i>	<i>Qa</i>	<i>Qf</i>
$1 \rightarrow 0$	0	0	1	1	1	1
$1 \rightarrow 0$	0	1	1	1	1	0
1	1	1	1	0	1	0
0	1	0	0	1	0	1
$1 \rightarrow 0$	1	0	1	1	0	1
$1 \rightarrow 0$	0	1	1	1	1	0
$0 \rightarrow 1$	0	1	1	0	1	0
$0 \rightarrow 1$	1	1	0	1	1	1

## 4.3 Contadores

---

Contadores são circuitos digitais que variam os seus estados de saída, sob o comando de um pulso de *clock*, de acordo com uma sequência de contagem predeterminada no projeto. Basicamente esses sistemas são divididos em duas categorias, que são **Contadores Assíncronos e Síncronos**.

### 4.3.1 Contadores assíncronos

Seus *flip-flops* funcionam de maneira assíncrona (sem sincronismo), não tendo entradas *clock* em comum. Nesse tipo de circuito, a entrada *clock* se faz apenas no primeiro *flip-flop*, sendo as outras derivadas das saídas dos blocos anteriores.

Vamos analisar os principais contadores assíncronos.

#### 4.3.1.1 Contador de pulsos

A principal característica de um contador de pulsos é apresentar nas saídas o sistema binário em sequência. Seu circuito básico apresenta um grupo de quatro *flip-flops* T ou JK Mestre-Escravo, os quais possuem a entrada T ou, no caso, J e K iguais a 1, originando na saída  $Q_f = \bar{Q}_a$ , a cada descida de *clock*.

A entrada dos pulsos se faz pela entrada *clock* do primeiro *flip-flop*, sendo as entradas *clock* dos *flip-flops* seguintes conectadas às saídas Q dos respectivos antecessores, conforme circuito da Figura 4.22.

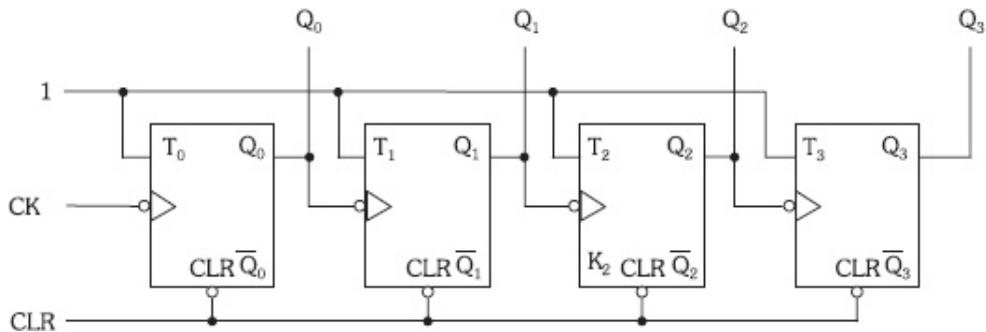


Figura 4.22 - Contador de pulsos assíncrono.

Vamos fazer, inicialmente, com que todos os *flip-flops* assumam saídas iguais a 0, através da aplicação de um nível 0 à entrada *clear*. A cada descida do pulso de *clock*, o primeiro *flip-flop* muda de estado, sendo esta troca aplicada à entrada do segundo *flip-flop*, fazendo com que este, troque de estado a cada descida da saída Q<sub>0</sub>, assim sucessivamente.

Vamos analisar este comportamento na Tabela 4.13.

Considerando Q<sub>0</sub> como *bit* menos significativo (LSB) e Q<sub>3</sub> como mais significativo (MSB), temos nas saídas o sistema binário em sequência (0000 a 1111).

#### Lembre-se

Na prática, o *bit* menos significativo de um número binário recebe a notação de *LSB* (*Least Significant Bit*) e o *bit* mais significativo de *MSB* (*Most Significant Bit*).

Notamos ainda que após a 16<sup>a</sup> descida de *clock*, o contador reinicia a contagem. A Figura 4.23 apresenta toda a sequência obtida graficamente, a partir da variação aplicada à entrada *clock* do sistema.

Tabela 4.13 - Comportamento do contador de pulsos assíncrono

Descidas de <i>clock</i>	Saídas				
	$Q_0$	$Q_1$	$Q_2$	$Q_3$	
1 <sup>a</sup>	0	0	0	0	(Estado inicial, imposto por CLR = 0)
2 <sup>a</sup>	1	0	0	0	(Após a 1 <sup>a</sup> descida de <i>clock</i> : Q = 1)
3 <sup>a</sup>	0	1	0	0	(Após a 2 <sup>a</sup> descida: Q = 0 e Q = 1, obtido pela descida de $Q_0$ )
4 <sup>a</sup>	1	1	0	0	( $Q_0 = 1$ e Q, permanece igual a 1)
5 <sup>a</sup>	0	0	1	0	( $Q_0 = 0 \Rightarrow Q_1 = 0 \Rightarrow Q_2 = 1$ )
6 <sup>a</sup>	1	0	1	0	( $Q_0 = 1$ , $Q_1$ e $Q_2$ permanecem)
7 <sup>a</sup>	0	1	1	0	( $Q_0 = 0 \Rightarrow Q_1 = 1$ )
8 <sup>a</sup>	1	1	1	0	( $Q_0 = 1$ )
9 <sup>a</sup>	0	0	0	1	( $Q_0 = 0 \Rightarrow Q_1 = 0 \Rightarrow Q_2 = 0 \Rightarrow Q_3 = 1$ )
10 <sup>a</sup>	1	0	0	1	( $Q_0 = 1$ )
11 <sup>a</sup>	0	1	0	1	( $Q_0 = 0 \Rightarrow Q_1 = 1$ )
12 <sup>a</sup>	1	1	0	1	( $Q_0 = 1$ )
13 <sup>a</sup>	0	0	1	1	( $Q_0 = 0 \Rightarrow Q_2 = 0 \Rightarrow Q_2 = 1$ )
14 <sup>a</sup>	1	0	1	1	( $Q_0 = 1$ )
15 <sup>a</sup>	0	1	1	1	( $Q_0 = 0 \Rightarrow Q_1 = 1$ )
16 <sup>a</sup>	1	1	1	1	( $Q_0 = 1$ )
17 <sup>a</sup>	0	0	0	0	( $Q_0 = 0 \Rightarrow Q_1 = 0 \Rightarrow Q_2 = 0 \Rightarrow Q_3 = 0$ )

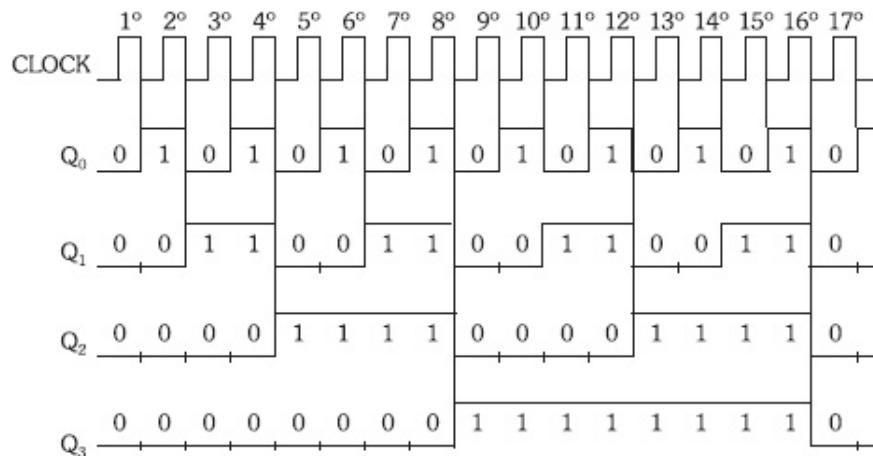


Figura 4.23 - Sequência das saídas do contador de pulsos.

Analisando os gráficos, notamos que o período de  $Q_0$  é o dobro do período do *clock*; logo, a frequência de  $Q_0$  será a metade da frequência do *clock*, pois  $f = 1/T$ . Analisando a saída  $Q_1$ , vemos que seu período é o dobro de  $Q_0$  e o quádruplo do *clock*; logo, sua frequência será a metade de  $Q_0$  e um quarto da frequência do pulso de *clock*. Isso se estende sucessivamente aos demais *flip-flops*. Assim sendo, uma das aplicações

dos contadores será a de dividir a frequência de sinais (onda quadrada) aplicados à entrada *clock*. No caso desse contador, a divisão será por um número múltiplo de  $2^N$ , sendo **N** o número de *flip-flops* utilizados.

#### 4.3.1.2 Contador de década

O contador de década é o circuito que efetua a contagem em números binários de 0 a 9 (dez algarismos). Isso significa acompanhar a sequência do código BCD 8421 de 0000 até 1001.

Para construir este circuito, utilizamos o contador de pulsos, interligando as entradas *clear* dos *flip-flops*.

Para que o contador conte somente de 0 a 9, deve-se jogar um nível 0 na entrada *clear* assim que surgir o caso 10 (1010), ou seja, no décimo pulso. Para isso utilizamos uma porta NE de duas entradas, ligando apenas  $Q_3$  e  $Q_1$  nesta, pois só serão iguais a 1 simultaneamente nesse caso, zerando as saídas. O circuito de um contador de década assíncrono é mostrado na Figura 4.24.

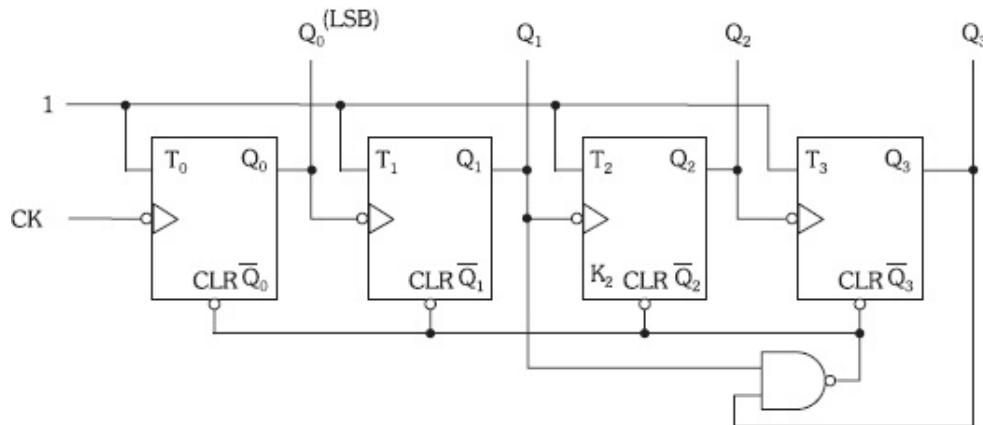


Figura 4.24 - Contador de década assíncrono.

A Tabela 4.14 apresenta a sequência da contagem:

Tabela 4.14 - Tabela-verdade de um contador de década assíncrono

Descidas de <i>clock</i>	$Q_3$	$Q_2$	$Q_1$	$Q_0$	CLR
1 <sup>a</sup>	0	0	0	0	1
2 <sup>a</sup>	0	0	0	1	1
3 <sup>a</sup>	0	0	1	0	1
4 <sup>a</sup>	0	0	1	1	1
5 <sup>a</sup>	0	1	0	0	1
6 <sup>a</sup>	0	1	0	1	1
7 <sup>a</sup>	0	1	1	0	1
8 <sup>a</sup>	0	1	1	1	1
9 <sup>a</sup>	1	0	0	0	1
10 <sup>a</sup>	1	0	0	1	1
	1	0	1	0	0

Após a décima descida de *clock*, o contador tende a assumir o estado  $Q_0 = 0$ ,  $Q_1 = 1$ ,  $Q_2 = 0$ ,  $Q_3 = 1$  ( $1010_2$ ), porém, neste instante, a entrada *clear* vai para 0, zerando o contador, ou seja, fazendo com que assuma o estado 0 (0000), reiniciando a contagem.

Este contador pode ser utilizado como divisor de frequência por 10 para uma onda quadrada aplicada à entrada *clock*, pois possui dez estados de saída.

#### 4.3.1.3 Contador sequencial de 0 a n

Vimos no item anterior um contador que faz a contagem de 0 até 9. Utilizando o mesmo processo, podemos fazer um contador contar de 0 até um número  $n$  qualquer. Para isso, basta verificar quais as saídas do contador para o caso seguinte a  $n$ , colocá-las em uma porta NE e à saída desta ligar as entradas *clear* dos *flip-flops*.

Para exemplificar, vamos elaborar o circuito de um contador de 0 a 5. Nesse caso, desejamos que o contador recomece a contagem após o estado 5, ou seja, passe para 0 todos os *flip-flops*.

Nesse caso, o estado seguinte a  $n$  será o 6, ocasionando nas saídas  $Q_2 = 1$ ,  $Q_1 = 1$  e  $Q_0 = 0$  ( $110$ ). Quando ocorrer deve haver um 0 nas entradas *clear* interligadas, levando o contador a 0. Devemos, para tanto, ter na entrada da porta NE a ligação de  $Q_2$  e  $Q_1$ , pois na sequência da contagem, estas vão assumir níveis 1 simultaneamente apenas no caso 110.

A Figura 4.25 mostra o circuito assim configurado.

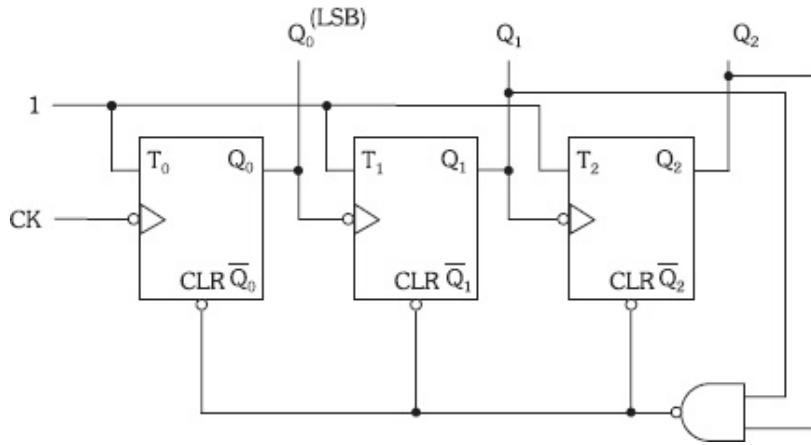


Figura 4.25 - Contador de 0 a 5 assíncrono.

No circuito, utilizamos somente três *flip-flops*, pois são suficientes para a contagem de 0 a  $7_{10}$  ( $2^3 = 8$ ).

### 4.3.2 Contadores síncronos

Esses contadores possuem entradas *clock* curto-circuitadas, ou seja, o *clock* entra em todos os *flip-flops* simultaneamente, fazendo todos atuarem de forma sincronizada.

Para que haja mudanças de estado, devemos estudar o comportamento das entradas J e K dos vários *flip-flops*, para que tenhamos nas saídas, as sequências desejadas.

Para estudarmos os contadores síncronos devemos sempre escrever a tabela-verdade, estudando quais devem ser as entradas J e K dos vários *flip-flops*, para que eles assumam o estado seguinte. Para isso, vamos utilizar a tabela-verdade do *flip-flop JK*.

A partir da Tabela 4.15 construímos a Tabela 4.16, relacionando os estados de saída e as entradas J e K:

Tabela 4.15 - Tabela do *flip-flop JK*

J	K	Qf	
0	0	Qa	(mantém o estado)
0	1	0	(fixa 0)
1	0	1	(fixa 1)
1	1	$\bar{Q}a$	(inverte o estado)

Tabela 4.16 - Tabela de projeto dos contadores síncronos

	Qa	Qf	J	K
1)	0	0	0	X
2)	0	1	1	X
3)	1	0	X	1
4)	1	1	X	0

Vamos, a seguir, analisar cada caso:

- 1) Se o *flip-flop* estiver em 0 ( $Q_a = 0$ ) e quisermos que o estado a ser assumido seja 0 ( $Q_f = 0$ ), podemos tanto manter o estado do *flip-flop* ( $J = 0, K = 0 \rightarrow Q_f = Q_a$ ) como fixar 0 ( $J = 0, K = 1 \rightarrow Q_f = 0$ ); logo, se  $J = 0$  e  $K = X$ , temos a passagem de  $Q_a = 0$  para  $Q_f = 0$ .
- 2) Se o *flip-flop* estiver em 0 ( $Q_a = 0$ ) e quisermos que o estado a ser assumido seja 1 ( $Q_f = 1$ ), podemos tanto inverter o estado ( $J = 1, K = 1 \rightarrow Q_f = \bar{Q}_a$ ) como fixar 1 ( $J = 1, K = 0 \rightarrow Q_f = 1$ ); logo, se  $J = 1$  e  $K = X$ , temos a passagem de  $Q_a = 0$  para  $Q_f = 1$ .
- 3) Quando o *flip-flop* estiver em 1 ( $Q_a = 1$ ) e quisermos que ele vá para 0 ( $Q_f = 0$ ), podemos inverter o estado ( $J = 1, K = 1 \rightarrow Q_f = \bar{Q}_a$ ) ou fixar 0 ( $J = 0, K = 1 \rightarrow Q_f = 0$ ); logo, se  $J = X$  e  $K = 1$ , temos a passagem de  $Q_a = 1$  para  $Q_f = 0$ .
- 4) Quando o *flip-flop* estiver em 1 ( $Q_a = 1$ ) e quisermos que ele permaneça em 1 ( $Q_f = 1$ ), podemos manter o estado ( $J = 0, K = 0 \rightarrow Q_f = Q_a$ ) ou fixar 1 ( $J = 1, K = 0 \rightarrow Q_f = 1$ ); logo, se  $J = X$  e  $K = 0$ , temos a passagem de  $Q_a = 1$  para  $Q_f = 1$ .

De posse dos resultados das entradas J e K dos *flip-flops* para a sequência desejada, obtidos da tabela, efetuamos as simplificações e montamos um circuito combinacional que em função das saídas dos *flip-flops* vai atuar nessas entradas para processar as mudanças de estado.

Genericamente, um contador síncrono possui o esquema da Figura 4.26.

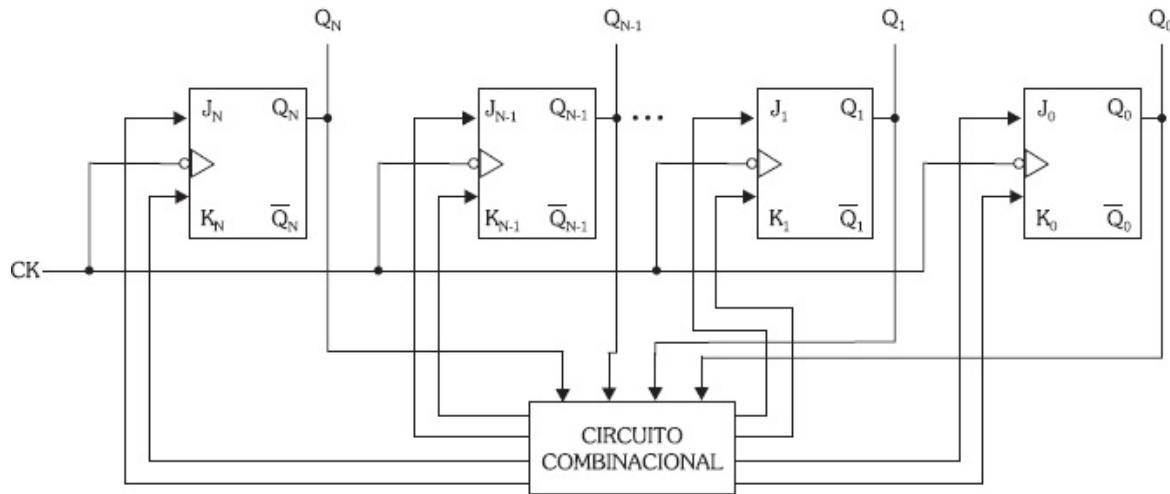


Figura 4.26 - Esquema genérico de um contador síncrono.

Nos próximos itens vamos, a título de exemplo, efetuar projetos de contadores síncronos para gerar sequências conhecidas ou de grande aplicabilidade na prática.

#### 4.3.2.1 Contador síncrono gerador de código binário de 4 bits

Para gerar esse código, necessitamos de quatro *flip-flops* JK Mestre-Escravo, ou seja, um *flip-flop* para cada bit do código. A Tabela 4.17 mostra a sequência proposta.

Tabela 4.17 - Sequência do gerador de código binário de 4 bits

ck	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
1 <sup>a</sup>	0	0	0	0
2 <sup>a</sup>	0	0	0	1
3 <sup>a</sup>	0	0	1	0
4 <sup>a</sup>	0	0	1	1
5 <sup>a</sup>	0	1	0	0
6 <sup>a</sup>	0	1	0	1
7 <sup>a</sup>	0	1	1	0
8 <sup>a</sup>	0	1	1	1
9 <sup>a</sup>	1	0	0	0
10 <sup>a</sup>	1	0	0	1
11 <sup>a</sup>	1	0	1	0
12 <sup>a</sup>	1	0	1	1
13 <sup>a</sup>	1	1	0	0
14 <sup>a</sup>	1	1	0	1
15 <sup>a</sup>	1	1	1	0
16 <sup>a</sup>	1	1	1	1

Esta tabela apresenta a sequência que as saídas dos *flip-flops* devem assumir em função da presença de pulsos de *clock*. Para o projeto, devemos estudar, para cada caso, o comportamento das entradas J e K dos *flip-flops* e levantar o circuito necessário para gerar a sequência.

Vamos supor que, ao ligarmos o contador, ele assuma o seguinte estado inicial:

Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
0	0	0	0

Ele deve, após o primeiro pulso de *clock*, passar para o estado seguinte:

Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
0	0	0	1

Sob a presença do primeiro pulso de *clock*, temos:

- » Q<sub>3</sub>: estava em 0, deve passar para 0; logo, antes do primeiro pulso de *clock*, devemos ter as seguintes entradas nesse *flip-flop*: J<sub>3</sub> = 0 e K<sub>3</sub> = X (J = 0 e K = X → Q<sub>a</sub> = 0 passa para Q<sub>f</sub> = 0).

- »  $Q_2$ : caso análogo a  $Q_3$ ; logo,  $J_2 = 0$  e  $K_2 = X$ .
- »  $Q_1$ : idem; logo,  $J_1 = 0$  e  $K_1 = X$ .
- »  $Q_0$ : estava em 0, após o primeiro pulso de *clock* deve mudar para 1; logo, antes do primeiro pulso de *clock*, devemos ter as seguintes entradas nesse *flip-flop*:  $J_0 = 1$  e  $K_0 = X$  ( $J = 1$  e  $K = X \rightarrow Q_a = 0$  passa para  $Q_f = 1$ ).

A partir da análise podemos escrever a primeira linha da tabela-verdade.

Descida de <i>clock</i>	$Q_3$	$Q_2$	$Q_1$	$Q_0$	$J_3$	$K_3$	$J_2$	$K_2$	$J_1$	$K_1$	$J_0$	$K_0$
1 <sup>a</sup>	0	0	0	0	0	X	0	X	0	X	1	X
	0	0	0	1								

O contador está, agora, no estado:

$Q_3$	$Q_2$	$Q_1$	$Q_0$
0	0	0	1

E deve, após o segundo pulso de *clock*, passar para:

$Q_3$	$Q_2$	$Q_1$	$Q_0$
0	0	1	0

Vamos analisar as entradas J e K para este caso:

- »  $Q_3$ : estava em 0 e deve permanecer em 0; logo, antes do segundo pulso de *clock* devemos ter a seguinte situação de entrada:  $J_3 = 0$  e  $K_3 = X$ .
- »  $Q_2$ : possui caso análogo a  $Q_3$ ; logo,  $J_2 = 0$  e  $K_2 = X$ .
- »  $Q_1$ : estava em 0 e deve passar para 1; logo, antes do segundo pulso de *clock*, devemos ter a seguinte situação de entrada no *flip-flop*:  $J_1 = 1$  e  $K_1 = X$ .
- »  $Q_0$ : estava em 1 e deve passar para 0; logo, antes do segundo pulso de *clock*, devemos ter a seguinte situação de entrada:  $J_0 = X$  e  $K_0 = 1$ .

Podemos, a partir disso, escrever a segunda linha da tabela-verdade:

Descidas de <i>clock</i>	$Q_3$	$Q_2$	$Q_1$	$Q_0$	$J_3$	$K_3$	$J_2$	$K_2$	$J_1$	$K_1$	$J_0$	$K_0$
1ª	0	0	0	0	0	X	0	X	0	X	1	X
2ª	0	0	0	1	0	X	0	X	1	X	X	1
	0	0	1	0								

Para fixarmos melhor o procedimento, vamos analisar mais uma mudança do contador, ou seja, após a descida do 3º pulso de *clock*, a passagem do estado 2 para o estado 3:

	$Q_3$	$Q_2$	$Q_1$	$Q_0$	
Estado 2 →	0	0	1	0	» $Q_3$ : vai de 0 para 0 → $J_3 = 0$ e $K_3 = X$
Estado 3 →	0	0	1	1	» $Q_2$ : vai de 0 para 0 → $J_2 = 0$ e $K_2 = X$ » $Q_1$ : vai de 1 para 1 → $J_1 = X$ e $K_1 = 0$ » $Q_0$ : vai de 0 para 1 → $J_0 = 1$ e $K_0 = X$

A tabela-verdade, até a terceira linha, será:

Descidas de <i>clock</i>	$Q_3$	$Q_2$	$Q_1$	$Q_0$	$J_3$	$K_3$	$J_2$	$K_2$	$J_1$	$K_1$	$J_0$	$K_0$
1ª	0	0	0	0	0	X	0	X	0	X	1	X
2ª	0	0	0	1	0	X	0	X	1	X	X	1
3ª	0	0	1	0	0	X	0	X	X	0	1	X
	0	0	1	1								

Utilizando o mesmo procedimento para os outros casos, obtemos a tabela completa:

Tabela 4.18 - Tabela para o projeto completa

Descidas de <i>clock</i>	$Q_3$	$Q_2$	$Q_1$	$Q_0$	$J_3$	$K_3$	$J_2$	$K_2$	$J_1$	$K_1$	$J_0$	$K_0$
1 <sup>a</sup>	0	0	0	0	0	X	0	X	0	X	1	X
2 <sup>a</sup>	0	0	0	1	0	X	0	X	1	X	X	1
3 <sup>a</sup>	0	0	1	0	0	X	0	X	X	0	1	X
4 <sup>a</sup>	0	0	1	1	0	X	1	X	X	1	X	1
5 <sup>a</sup>	0	1	0	0	0	X	X	0	0	X	1	X
6 <sup>a</sup>	0	1	0	1	0	X	X	0	1	X	X	1
7 <sup>a</sup>	0	1	1	0	0	X	X	0	X	0	1	X
8 <sup>a</sup>	0	1	1	1	1	X	X	1	X	1	X	1
9 <sup>a</sup>	1	0	0	0	X	0	0	X	0	X	1	X
10 <sup>a</sup>	1	0	0	1	X	0	0	X	1	X	X	1
11 <sup>a</sup>	1	0	1	0	X	0	0	X	X	0	1	X
12 <sup>a</sup>	1	0	1	1	X	0	1	X	X	1	X	1
13 <sup>a</sup>	1	1	0	0	X	0	X	0	0	X	1	X
14 <sup>a</sup>	1	1	0	1	X	0	X	0	1	X	X	1
15 <sup>a</sup>	1	1	1	0	X	0	X	0	X	0	1	X
16 <sup>a</sup>	1	1	1	1	X	1	X	1	X	1	X	1

No projeto, o estado 0 foi considerado após o estado 15, pois ao final, o contador deve reiniciar a contagem.

Para obter as expressões de  $J_3$ ,  $K_3$ ,  $J_2$ ,  $K_2$ ,  $J_1$ ,  $K_1$ ,  $J_0$  e  $K_0$  simplificadas, vamos utilizar mapas de Karnaugh:

$\bar{Q}_3$	$\bar{Q}_1$	$Q_1$	$Q_3$
$\bar{Q}_3$	0	0	0
$\bar{Q}_3$	0	0	1
$Q_3$	X	X	X
$Q_3$	X	X	X
$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_2$

$$(J_3) J_3 = Q_2 Q_1 Q_0$$

$\bar{Q}_3$	$\bar{Q}_1$	$Q_1$	$Q_3$	$\bar{Q}_2$
$\bar{Q}_3$	X	X	X	X
$\bar{Q}_3$	X	X	X	X
$Q_3$	0	0	1	0
$Q_3$	0	0	0	0
$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_2$	$Q_2$

$$(K_3) K_3 = Q_2 Q_1 Q_0 \quad \therefore J_3 = K_3 = Q_2 Q_1 Q_0$$

$\bar{Q}_1$		$Q_1$		$\bar{Q}_2$
$\bar{Q}_3$	0	0	1	
	X	X	X	X
$Q_3$	X	X	X	X
	0	0	1	0
$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_2$	

$$(J_2) J_2 = Q_1 Q_0$$

$\bar{Q}_1$		$Q_1$		$\bar{Q}_2$
$\bar{Q}_3$	X	X	X	
	0	0	1	0
$Q_3$	0	0	1	0
	X	X	X	X
$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_2$	

$$(K_2) K_2 = Q_1 Q_0 \quad \therefore J_2 = K_2 = Q_1 Q_0$$

$\bar{Q}_1$		$Q_1$		$\bar{Q}_2$
$\bar{Q}_3$	0	1	X	X
	0	1	X	X
$Q_3$	0	1	X	X
	0	1	X	X
$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_2$	

$$(J_1) J_1 = Q_0$$

$$(K_1) K_1 = Q_0 \quad \therefore J_1 = K_1 = Q_0$$

$\bar{Q}_1$		$Q_1$		$\bar{Q}_2$
$\bar{Q}_3$	1	X	X	1
	1	X	X	1
$Q_3$	1	X	X	1
	1	X	X	1
$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_2$	

$$(J_0) J_0 = 1$$

$$(K_0) K_0 = 1 \quad \therefore J_0 = K_0 = 1$$

Figura 4.27 - Mapas de Karnaugh com as simplificações de  $J_3$  a  $K_0$ .

O circuito completo desse contador é visto na Figura 4.28.

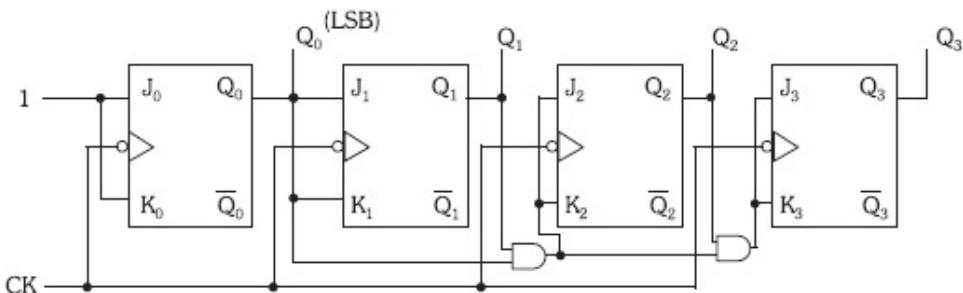


Figura 4.28 - Contador síncrono gerador de código binário de 4 bits.

As entradas *clear* e *preset* dos *flip-flops* poderiam, da mesma forma que nos contadores assíncronos, ser utilizadas para estabelecer o caso inicial, zerar o contador, ou ainda, fixar qualquer caso no decorrer da contagem.

#### 4.3.2.2 Contador síncrono gerador de uma sequência qualquer

Podemos construir um contador que gere uma sequência qualquer. Para isso, basta estabelecermos a sequência e seguirmos o método já conhecido, ou seja, o da determinação das entradas J e K. Os estados que não fizerem parte da sequência devem ser considerados condições irrelevantes, ou serem encadeados objetivando atingir o estado inicial.

O *loop* que o contador deve efetuar para acompanhar a sequência é denominado **Diagrama de Estados**. A Figura 4.29 apresenta um diagrama de estados genérico.

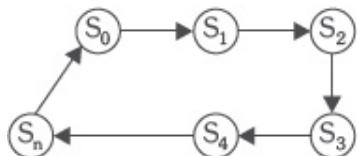


Figura 4.29 - Diagrama de estados genérico.



#### Exercícios resolvidos

- 4.1) Elabore um Contador em Anel, cujo diagrama de estados é visto na Figura 4.30.

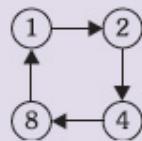


Figura 4.30 - Diagrama de estados do Contador em Anel.

De acordo com o diagrama de estados, este contador, gera a sequência da Tabela 4.19.

Tabela 4.19 - Tabela-verdade do Contador em anel

$Q_3$	$Q_2$	$Q_1$	$Q_0$
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

Vamos levantar, de modo análogo aos anteriores, o comportamento das entradas  $J$  e  $K$ , conforme a sequência apresentada. Para isso, montamos a tabela-verdade:

Tabela 4.20 - Tabela-verdade do projeto

$Q_3$	$Q_2$	$Q_1$	$Q_0$	$J_3$	$K_3$	$J_2$	$K_2$	$J_1$	$K_1$	$J_0$	$K_0$
0	0	0	1	0	X	0	X	1	X	X	1
0	0	1	0	0	X	1	X	X	1	0	X
0	1	0	0	1	X	X	1	0	X	0	X
1	0	0	0	X	1	0	X	0	X	1	X

Se obtivermos o estado inicial através das entradas *preset* e *clear*, faremos o contador permanecer sempre no *loop* da sequência; logo, os outros estados tornam-se irrelevantes. Podemos transcrever a tabela-verdade para os diagramas:

	$\bar{Q}_1$		$Q_1$		
$\bar{Q}_3$	X	0	X	0	$\bar{Q}_2$
$Q_3$	(1)	X	X	X	$Q_2$
	X	X	X	X	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_0$	

$$(J_3) J_3 = Q_2$$

	$\bar{Q}_1$		$Q_1$		
$\bar{Q}_3$	X	X	X	X	$\bar{Q}_2$
$Q_3$	X	X	X	X	$Q_2$
	(1)	X	X	X	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_0$	

$$(K_3) K_3 = \bar{Q}_2^*$$

	$\bar{Q}_1$		$Q_1$		
$\bar{Q}_3$	X	0	(X)	1	$\bar{Q}_2$
$Q_3$	X	X	X	X	$Q_2$
	0	X	(X)	X	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_0$	

$$(J_2) J_2 = Q_1$$

	$\bar{Q}_1$		$Q_1$		
$\bar{Q}_3$	(X)	X	X	X	$\bar{Q}_2$
$Q_3$	1	X	X	X	$Q_2$
	X	X	X	X	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_0$	

$$(K_2) K_2 = \bar{Q}_1^*$$

	$\bar{Q}_1$		$Q_1$		
$\bar{Q}_3$	X	(1)	X	X	$\bar{Q}_2$
$Q_3$	0	X	X	X	$Q_2$
	X	X	X	X	$\bar{Q}_2$
	0	(X)	X	X	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_0$	

$$(J_1) J_1 = Q_0$$

	$\bar{Q}_1$		$Q_1$		
$\bar{Q}_3$	X	X	X	(1)	$\bar{Q}_2$
$Q_3$	X	X	X	X	$Q_2$
	X	X	X	(X)	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_0$	

$$(K_1) K_1 = \bar{Q}_0^*$$

	$\bar{Q}_1$		$Q_1$		
$\bar{Q}_3$	X	X	X	0	$\bar{Q}_2$
$Q_3$	0	X	X	X	$Q_2$
	(X)	X	X	X	$\bar{Q}_2$
	1	X	X	X	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_0$	

$$(J_0) J_0 = Q_3$$

	$\bar{Q}_1$		$Q_1$		
$\bar{Q}_3$	(X)	1	X	X	$\bar{Q}_2$
$Q_3$	X	X	X	X	$Q_2$
	X	X	X	X	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$\bar{Q}_0$	

$$(K_0) K_0 = Q_3^*$$

Figura 4.31 - Mapas de Karnaugh com as simplificações de  $J_3$  a  $K_0$ .

Após obter as expressões, vamos esquematizar o circuito do Contador em Anel:

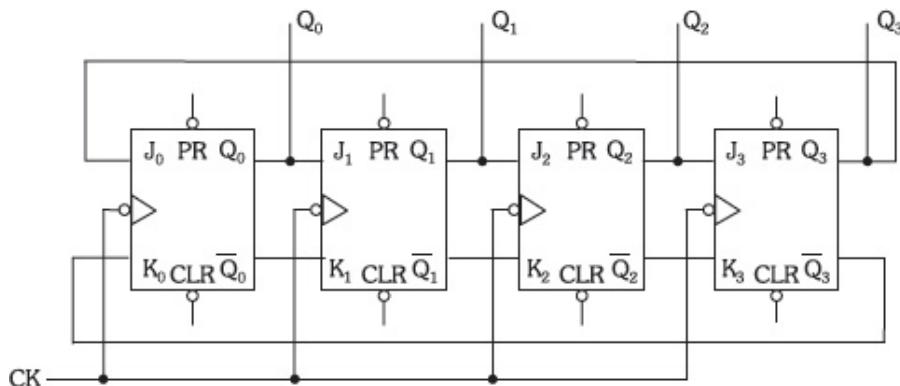


Figura 4.32 - Circuito do Contador em Anel.

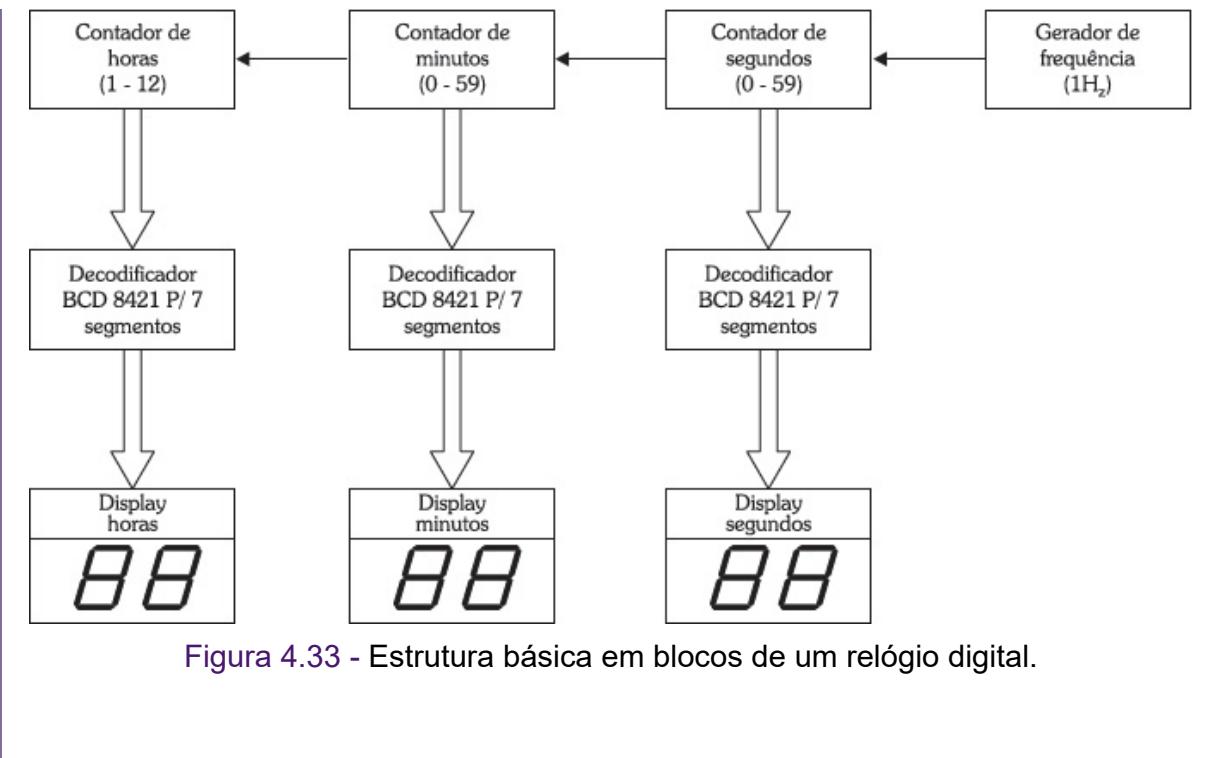
### Lembre-se

\*Embora pudéssemos agrupar  $K_3$ ,  $K_2$ ,  $K_1$  e  $K_0$  em 1 (agrupamento máximo), em razão de facilidade de implementação do circuito, agrupamos as oitavas  $Q_3$ ,  $Q_2$ ,  $Q_1$  e  $Q_0$ .

### Amplie seus conhecimentos

Com os elementos estudados até aqui podemos esquematizar a estrutura básica de um relógio digital. Na prática essa estrutura é desenvolvida e colocada em circuitos integrados comerciais ou dentro de circuitos mais complexos para funcionar como relógio ou outros sistemas correlacionados. Observe a Figura 4.33.

Perceba que a ativação de cada contador é obtida a partir do gerador de frequência sucessivamente. Esta prática se faz pela saída do terminal mais significativo (MSB) de cada contador ligado na entrada *clock* do seguinte.



## 4.4 Registradores de Deslocamento

Durante o período em que sua entrada *clock* for igual a 0, o *flip-flop* pode armazenar um *bit* apenas (saída Q). Porém, se necessitarmos guardar uma informação de mais de um *bit*, o *flip-flop* torna-se insuficiente. Para isso utilizamos um sistema denominado **Registrador de Deslocamento (Shift Register)**. Trata-se de um certo número de *flip-flops* D ligados de tal forma que as saídas de cada bloco sejam aplicadas às entradas do *flip-flop* seguinte. A Figura 4.34 apresenta um Registrador de Deslocamento generalizado para N + 1 bits.

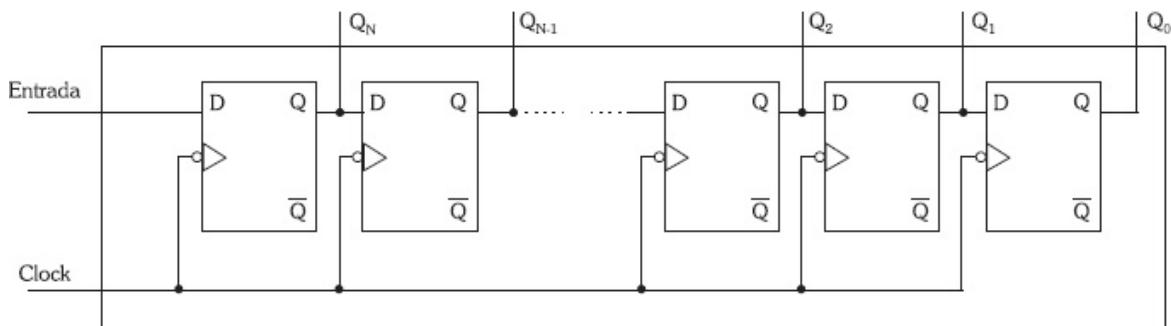


Figura 4.34 - Registrador de Deslocamento genérico.

O funcionamento deste sistema, juntamente com suas aplicações, é tratado nos itens subsequentes.

#### 4.4.1 Conversor Série-Paralelo

Antes de estudarmos o comportamento do Registrador de Deslocamento como Conversor Série-Paralelo, vamos explicar o que significa informação série e informação paralela.

Chamamos de informação paralela aquela na qual todos os *bits* se apresentam simultaneamente. Uma informação paralela precisa de tantos fios quantos forem os *bits* contidos nela, além, logicamente, do fio referencial do sistema (terra). Para exemplificar, vamos utilizar uma informação de 4 *bits*:

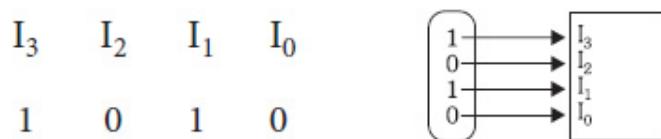


Figura 4.35 - Informação paralela.

Essa informação necessita de quatro fios para ser transmitida ou inserida no bloco.

Informação série é aquela que utiliza apenas um fio, e os *bits* de informação vêm sequencialmente, um após o outro. Como exemplo, vamos utilizar a mesma informação, porém em série:

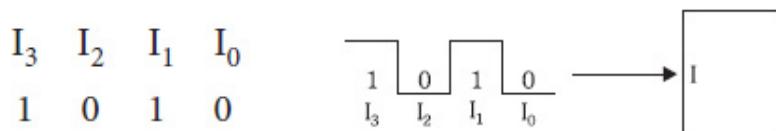


Figura 4.36 - Informação série.

Notamos que essa informação necessita de um fio para ser transmitida ou inserida no bloco.

O Registrador de Deslocamento pode ser usado para converter uma informação série em paralela, ou seja, funcionar como Conversor Série-Paralelo. A configuração básica nessa situação, para uma informação de 4 *bits*, é vista na Figura 4.37.

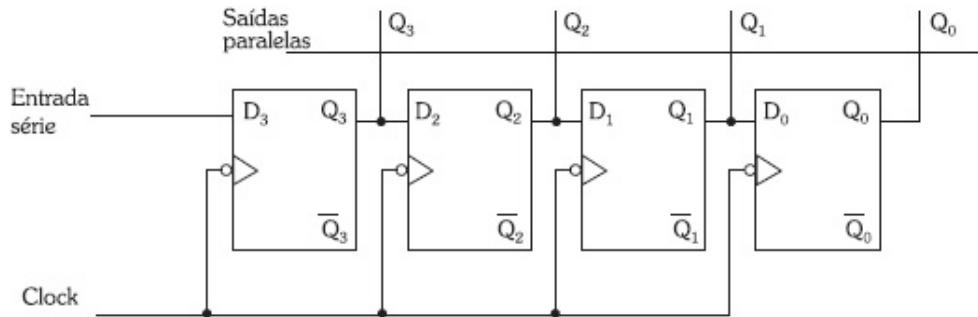


Figura 4.37 - Conversor Série-Paralelo.

Como exemplo, vamos aplicar a informação série  $I = 1010$  ( $I_3 I_2 I_1 I_0$ ) à entrada série do registrador e analisar as saídas  $Q_0$ ,  $Q_1$ ,  $Q_2$  e  $Q_3$ , após os pulsos de *clock*. Deve-se ressaltar que esses *flip-flops* atuam como mestre-escravo e têm sua comutação no instante da descida do pulso de *clock*.

Assim sendo, temos:

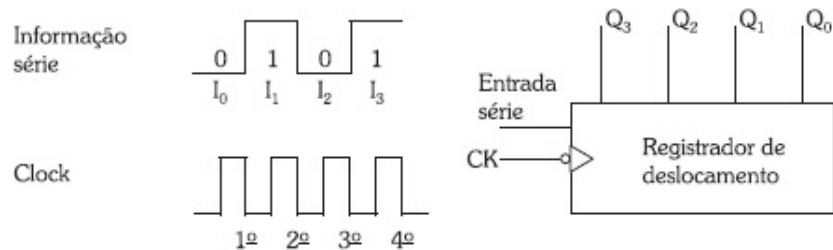


Figura 4.38 - Exemplo de entrada série.

Entraremos com a informação (1010), como mostrado na Figura 4.38, na entrada série e os pulsos de *clock* na respectiva entrada (CK).

Vamos supor que, inicialmente, as saídas  $Q_3$ ,  $Q_2$ ,  $Q_1$  e  $Q_0$  do registrador estejam em nível 0. Ao ser injetado na entrada o 1º bit de informação ( $I_0 = 0$ ) e ocorrer a descida do pulso de *clock*, o *flip-flop* 3 vai apresentar na saída 0 ( $D_3 = 0 \rightarrow Q_3 = 0$ ). Após este pulso de *clock*, aparece na entrada o bit seguinte de informação ( $I_1 = 1$ ) e na descida do 2º pulso de *clock* temos a passagem de  $I_0$  para o *flip-flop* 2 ( $D_2 = 0 \rightarrow Q_2 = 0$ ) e  $Q_3$  assume o valor do bit de informação  $I_1$  (entrada série =  $D_3 = 1 \rightarrow Q_3 = 1$ ).

Após a descida do 3º pulso de *clock*, ficamos com a seguinte situação:

- »  $Q_1 = 0$  ( $D_1 = Q_2 = 0 \rightarrow Q_1 = 0$ ),
- »  $Q_2 = 1$  ( $D_2 = Q_3 = 1 \rightarrow Q_2 = 1$ ) e
- »  $Q_3 = 0$  ( $D_3 = I_2 = 0 \rightarrow Q_3 = 0$ ).

Após o 4º pulso de *clock*, ocorre a seguinte situação:

- »  $Q_0 = 0$  ( $D_0 = Q_1 = 0 \rightarrow Q_0 = 0$ )
- »  $Q_1 = 1$  ( $D_1 = Q_2 = 1 \rightarrow Q_1 = 1$ )
- »  $Q_2 = 0$  ( $D_2 = Q_3 = 0 \rightarrow Q_2 = 0$ )
- »  $Q_3 = 1$  ( $D_3 = I_3 = 1 \rightarrow Q_3 = 1$ )

Notamos que, após o 4º pulso de *clock*, a informação I estará armazenada no registrador de deslocamento e aparecerá nas saídas  $Q_3$ ,  $Q_2$ ,  $Q_1$  e  $Q_0$  como uma informação paralela.

Para resumir, vamos representar toda a sequência sob a forma da tabela-verdade:

Tabela 4.21 - Sequência do exemplo resumida

Informação	Descidas de <i>clock</i>	$Q_3$	$Q_2$	$Q_1$	$Q_0$
$I_0 = 0$	1ª	0	0	0	0
$I_1 = 1$	2ª	1	0	0	0
$I_2 = 0$	3ª	0	1	0	0
$I_3 = 1$	4ª	1	0	1	0

É pelo motivo de deslocar a informação a cada pulso de *clock* que esse dispositivo é denominado Registrador de Deslocamento.

#### 4.4.2 Conversor Paralelo-Série

Para entrarmos com uma informação paralela, necessitamos de um registrador que apresente entradas *Preset* e *Clear*, pois com elas fazemos com que o registrador armazene a informação paralela. O registrador com essas entradas é exibido na Figura 4.39.

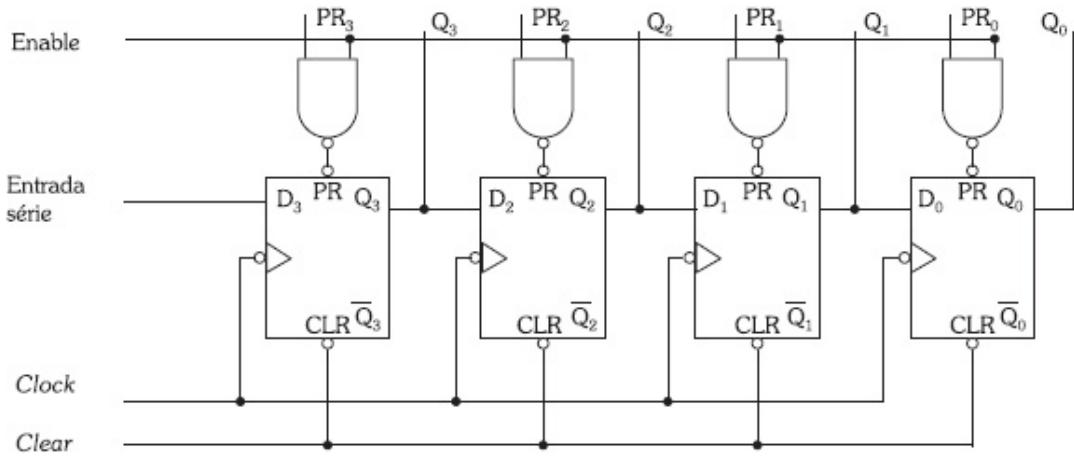


Figura 4.39 - Conversor Paralelo-Série.

Primeiramente, vamos estudar o funcionamento da entrada **ENABLE**. Quando a entrada enable estiver em 0, as entradas *preset* (PR) dos *flip-flops* assumem, respectivamente, níveis 1, fazendo com que o registrador atue normalmente. Quando a entrada enable for igual a 1, as entradas *preset* dos *flip-flops* assumem os valores complementares das entradas PR<sub>3</sub>, PR<sub>2</sub>, PR<sub>1</sub> e PR<sub>0</sub>; logo, os *flip-flops* assumem os valores que estiverem, respectivamente, em PR<sub>3</sub>, PR<sub>2</sub>, PR<sub>1</sub> e PR<sub>0</sub>.

Após essa análise, concluímos que, se zerarmos o registrador (aplicando 0 à entrada *clear*), e logo após introduzirmos a informação paralela (I<sub>3</sub>, I<sub>2</sub>, I<sub>1</sub> e I<sub>0</sub>) pelas entradas PR<sub>3</sub>, PR<sub>2</sub>, PR<sub>1</sub> e PR<sub>0</sub>, as saídas Q<sub>3</sub>, Q<sub>2</sub>, Q<sub>1</sub> e Q<sub>0</sub> assumirão, respectivamente, os valores da informação. Essa maneira de entrarmos com a informação no registrador é chamada de entrada paralela de informação, sendo a entrada enable responsável pela habilitação dela.

Para que o registrador de deslocamento funcione como **Conversor Paralelo-Série**, necessitamos zerá-lo e em seguida introduzir a informação como já descrito, recolhendo na saída Q<sub>0</sub> a mesma informação de modo série. É fácil de notar que a saída Q<sub>0</sub> assume primeiramente o valor I<sub>0</sub> e a cada descida do pulso de *clock* vai assumir sequencialmente os valores I<sub>1</sub>, I<sub>2</sub> e I<sub>3</sub>.

#### 4.4.3 Registrador de entrada série e saída série

Podemos utilizar o registrador de deslocamento com entrada série e o consequente armazenamento da informação nele, e recolher a informação também de modo série. Notamos que nessa aplicação, após a entrada da informação, se inibirmos a entrada de *clock*, a informação permanece no registrador até que haja uma nova entrada. Assim sendo, é fácil observar que o registrador funcionou como uma memória. A entrada de informação série se faz na entrada série do registrador e pode ser recolhida na saída  $Q_0$  do registrador.

#### 4.4.4 Registrador de entrada paralela e saída paralela

A entrada paralela, como já visto, se faz através dos terminais *preset* e *clear*. Se inibirmos a entrada de *clock*, a informação contida no registrador pode ser acessada pelos terminais de saída  $Q_3$ ,  $Q_2$ ,  $Q_1$  e  $Q_0$ .



#### Exercício resolvido

- 4.1) A partir dos sinais aplicados às entradas, esboce as formas de onda das saídas para o Registrador de Deslocamento de 4 bits, visto na Figura 4.40.

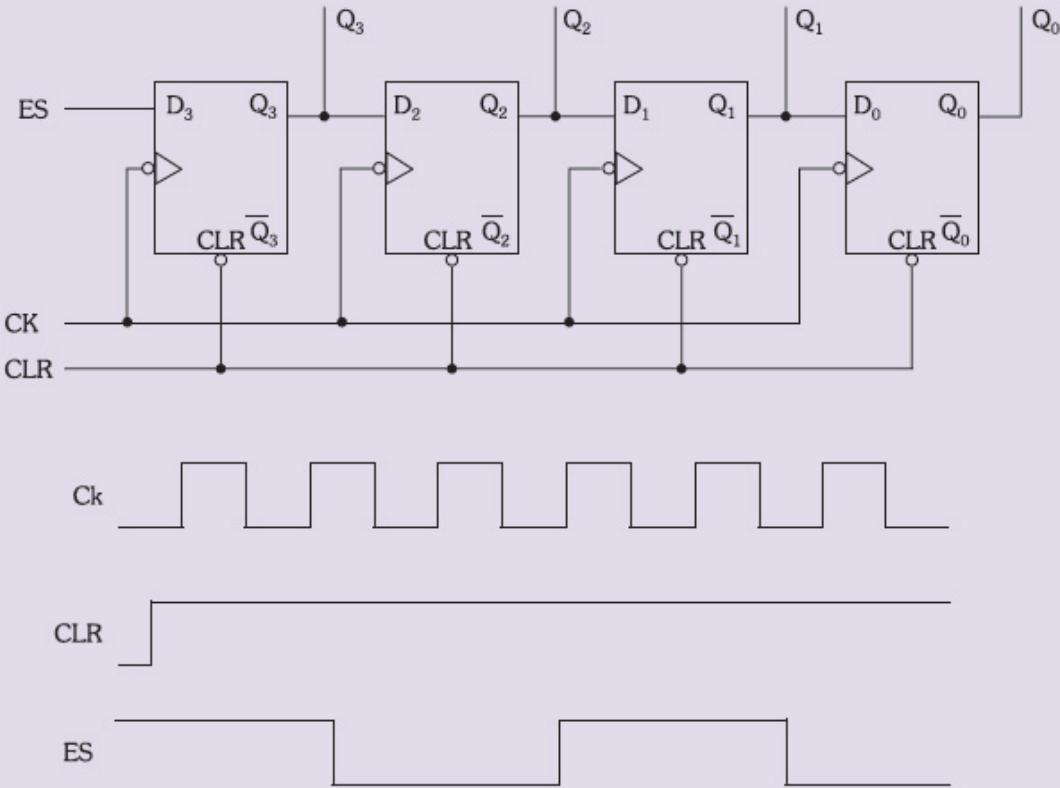


Figura 4.40 - Registrador e sinais aplicados.

Para solucionar, vamos considerar cada descida de *clock* e verificar, a partir da entrada série (ES), o nível de saída registrado em cada bloco anterior. A Figura 4.41 apresenta os sinais de saída resultantes deste processo, sendo o registrador inicialmente zerado pela forma de onda da entrada *clear*.

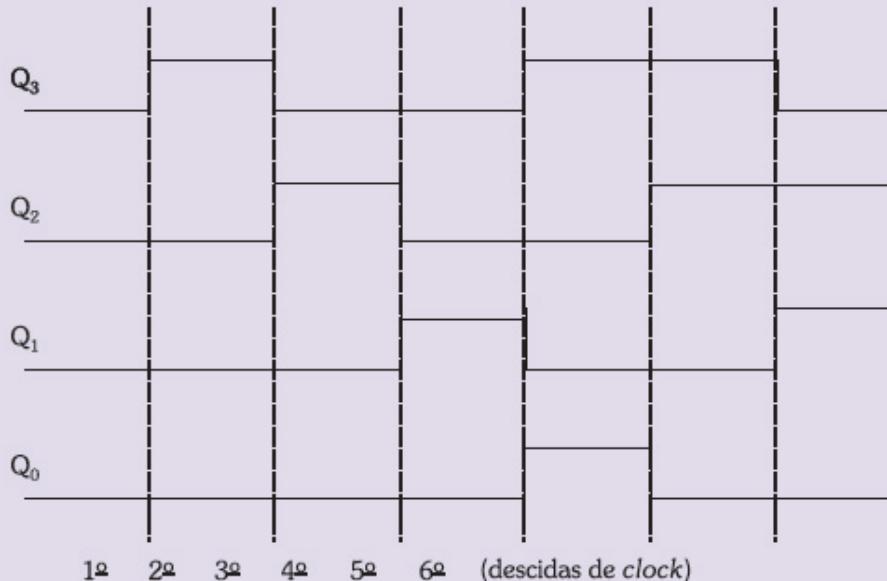


Figura 4.41 - Sinais de saída obtidos a partir dos sinais de entrada.

## 4.5 Projetos de sistemas sequenciais

Os sistemas sequenciais podem ser utilizados em projetos para diversas finalidades. São caracterizados por circuitos que executam sequências ou estados predefinidos sob o comando de pulsos de *clock*, juntamente com outros sinais de entrada. Os contadores já estudados, são circuitos sequenciais, destinados a executar sequências numéricas (contagens, como o nome diz), a partir de pulsos de *clock*.

Os sistemas sequenciais projetados para aplicações específicas, também são denominados de modo geral de **Máquinas de Estado** ou, ainda, **Máquinas de Estados Finitos**, pois teremos um número finito de estados possíveis de saída, a serem obtidos nos projetos. Os contadores são casos particulares de máquinas de estado. Existem dois modelos tradicionais para a implementação de máquinas de estados, são os modelos de *Moore* e o de *Mealy*.

No modelo de *Moore*, as saídas dependem do estado atual registrado internamente no sistema. As mudanças das saídas ocorrem apenas na próxima descida de *clock*. A Figura 4.42 apresenta em blocos a estrutura um sistema sequencial utilizando o modelo de *Moore*.

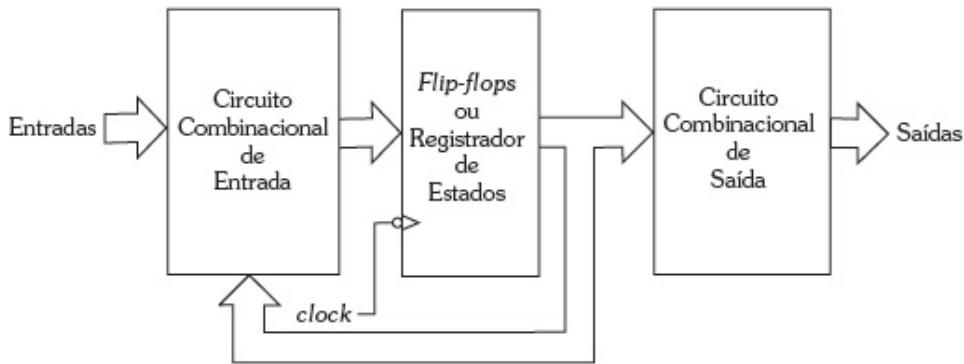


Figura 4.42 - Sistema sequencial utilizando o modelo de *Moore*.

Podemos notar, pela figura, que o *clock* comanda o registrador que envia os estados armazenados resultantes das entradas e estado anteriormente existente, para o para estágio de saída. As saídas irão assumir um novo estado, somente na próxima descida de *clock*.

No modelo de *Mealy*, as saídas dependem das entradas e do estado atual registrado internamente no sistema. Se houver mudanças nas entradas, as saídas também se alterarão. A Figura 4.43 apresenta em blocos a estrutura um sistema sequencial utilizando o modelo de *Mealy*.

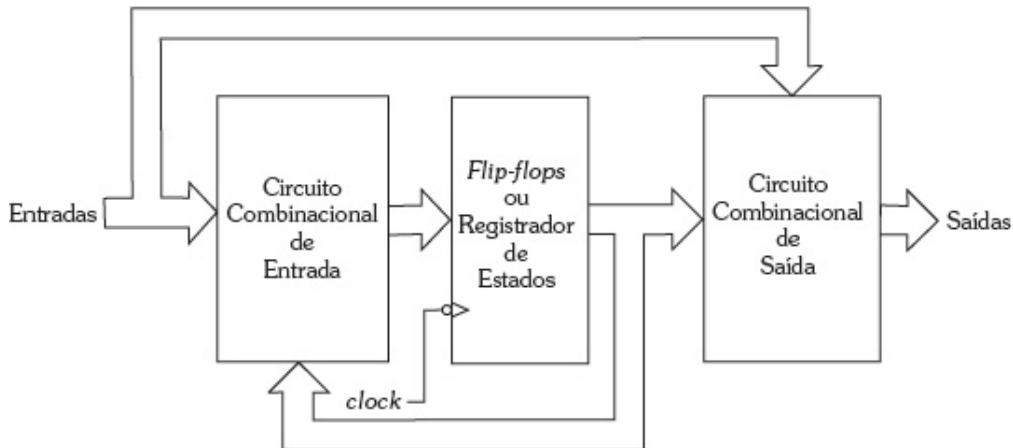


Figura 4.43 - Sistema sequencial utilizando o modelo de *Mealy*.

Pela figura, 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*.

As máquinas de estado também são 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 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 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 que no modelo de *Mealy*, as saídas podem atuar assincronamente.

Para efetuarmos projetos de sistemas sequenciais, de um modo geral, devemos seguir o mesmo procedimento utilizado na elaboração dos contadores síncronos, 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; passar os casos de saída para os mapas de Karnaugh, para simplificá-los e assim elaborar o circuito com as expressões simplificadas. Como exemplo, vamos elaborar um projeto no item a seguir.



## Exercício resolvido

- 4.1) Projete uma máquina de estados para atuar como contador de 3 bits para efetuar a contagem crescente ( $X = 0 \rightarrow 0$  a 7) ou decrescente ( $X = 1 \rightarrow 7$  a 0), através de uma variável de controle X. O circuito deverá também possuir uma saída S sinalizadora da contagem,  $S = 1$  para a contagem crescente e  $S = 0$  para a decrescente.

A introdução da variável X como entrada, faz com que o contador efetue a contagem crescente quando está em 0, e decrescente quando está em 1. A Figura 4.44 mostra o diagrama de estados e a Tabela 4.22 apresenta a sequência proposta.

Tabela 4.22 - Sequência de contagem proposta

$x$	$Q_2$	$Q_1$	$Q_0$	
Contagem crescente	0	0	0	0
	0	0	0	1
	0	0	1	0
	0	0	1	1
	0	1	0	0
	0	1	0	1
	0	1	1	0
	1	1	1	1
Contagem decrescente	1	1	1	0
	1	1	0	1
	1	1	0	0
	1	0	1	1
	1	0	1	0
	1	0	0	1
	1	0	0	0

Utilizando a tabela de projetos, levantamos a tabela-verdade das entradas J e K dos *flip-flops* e colocamos, também, a coluna S conforme solicita o enunciado. Ver a Tabela 4.23.

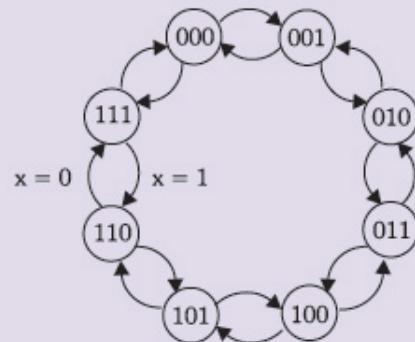


Figura 4.44 - Diagrama de estados.

Tabela 4.23 - Tabela de projeto dos contadores síncronos

Qa	Qf	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Tabela 4.24 - Tabela-verdade do projeto completa

X	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	J <sub>2</sub>	K <sub>2</sub>	J <sub>1</sub>	K <sub>1</sub>	J <sub>0</sub>	K <sub>0</sub>	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

A seguir, vamos simplificar o circuito das entradas J e K dos *flip-flops* e a saída S, através dos mapas de Karnaugh:

	$\bar{Q}_1$		$Q_1$		
$\bar{X}$	0	0	(1)	0	$\bar{Q}_2$
	X	X	(X)	X	
X	(X)	X	X	X	$Q_2$
	(1)	0	0	0	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$

$$(J_2) J_2 = X\bar{Q}_1\bar{Q}_0 + \bar{X}Q_1Q_0$$

	$\bar{Q}_1$		$Q_1$		
$\bar{X}$	X	X	(X)	X	$\bar{Q}_2$
	0	0	(1)	0	
X	(1)	0	0	0	$Q_2$
	(X)	X	X	X	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$

$$(K_2) K_2 = X\bar{Q}_1\bar{Q}_0 + \bar{X}Q_1Q_0$$

	$\bar{Q}_1$		$Q_1$		
$\bar{X}$	0	(1)	(X)	X	$\bar{Q}_2$
	0	(1)	(X)	X	
X	(1)	0	X	(X)	$Q_2$
	(1)	0	X	(X)	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$

$$(J_1) J_1 = X\bar{Q}_0 + \bar{X}Q_0$$

	$\bar{Q}_1$		$Q_1$		
$\bar{X}$	X	(X)	(1)	0	$\bar{Q}_2$
	X	(X)	(1)	0	
X	(X)	X	0	(1)	$Q_2$
	(X)	X	0	(1)	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$

$$(K_1) K_1 = X\bar{Q}_0 + \bar{X}Q_0 \quad \therefore J_1 = K_1 = X \oplus Q_0$$

	$\bar{Q}_1$		$Q_1$		
$\bar{X}$	1	X	X	1	$\bar{Q}_2$
	1	X	X	1	
X	1	X	X	1	$Q_2$
	1	X	X	1	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$

$$(J_0) J_0 = 1$$

	$\bar{Q}_1$		$Q_1$		
$\bar{X}$	X	1	1	X	$\bar{Q}_2$
	X	1	1	X	
X	(X)	1	1	X	$Q_2$
	(X)	1	1	X	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$

$$(K_0) K_0 = 1$$

	$\bar{Q}_1$		$Q_1$		
$\bar{X}$	1	1	1	1	$\bar{Q}_2$
	1	1	1	1	
X	0	0	0	0	$Q_2$
	0	0	0	0	$\bar{Q}_2$
	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$	$Q_0$	$\bar{Q}_0$

$$(S) S = \bar{X}$$

Figura 4.45 - Mapas de Karnaugh com as simplificações de  $J_2$  a  $S$ .

A partir das expressões simplificadas, montamos o circuito.

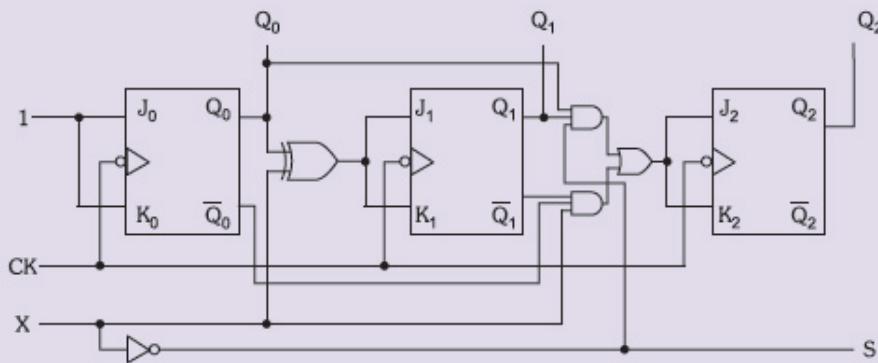


Figura 4.46 - Circuito final.

### Vamos recapitular?

Neste capítulo, você conheceu os *flip-flops*, que são os elementos básicos dos circuitos sequenciais.

Entendeu a estrutura e funcionamento dos contadores assíncronos e como implementá-los para contagens sequenciais até determinados valores numéricos.

Você aprendeu como se desenvolve projetos de contadores síncronos.

Compreendeu o funcionamento de registradores de deslocamento e as entradas e saídas de informações série e paralela.

Conheceu os principais modelos de máquinas de estado, suas características e aprendeu a elaborar um projeto.

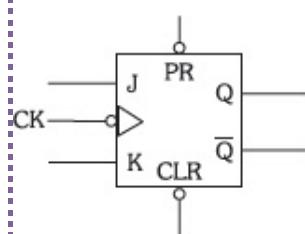


### Agora é com você!

- 1) Esquematize um *flip-flop RS* com entrada *clock* apenas com portas NOU. Para o circuito obtido, escreva as

tabelas, mostrando a atuação de R, S e *clock*.

- 2) Complete a tabela, referente à saída Q do *flip-flop* JK/Mestre-Escravo da Figura 4.47, em função das entradas aplicadas.



Clock	J	K	Preset	Clear	Q <sub>a</sub>	Q <sub>f</sub>
1 → 0	0	1	1	1	1	
1 → 0	1	1	1	1	1	
0	1	1	0	1	1	
1	1	0	1	0	0	
1 → 0	1	0	1	1	0	
1 → 0	0	0	1	1	1	
0 → 1	0	0	1	0	1	
0 → 1	1	1	0	1	0	

Figura 4.47 - *Flip-flop* e tabela de entradas aplicadas.

- 3) Elabore um contador assíncrono de 0 a  $8_{10}$ .
- 4) Altere o circuito obtido no exercício anterior, colocando uma entrada *clear* no contador para utilização externa.
- 5) Desenhe um contador assíncrono de 1 a 12. O circuito deve possuir uma entrada para estabelecer o caso inicial, através do nível 0 aplicado.
- 6) Elabore o projeto de um contador de década síncrono.
- 7) Esquematize um contador para trabalhar como divisor de frequência por 50.
- 8) Elabore o circuito de um contador síncrono que execute a sequência mostrada no diagrama da Figura 4.48. Considere os casos não pertencentes ao diagrama como irrelevantes.

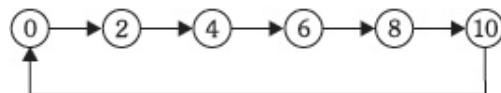


Figura 4.48 - Diagrama de estados do exercício 8.

- 9) Obtenha as expressões simplificadas dos *flip-flops* de um contador síncrono para gerar a sequência de 9 a 0.
- 10) Esboce as formas de onda para o registrador de deslocamento da Figura 4.49, em função dos sinais aplicados, considerando a entrada enable igual a 0.

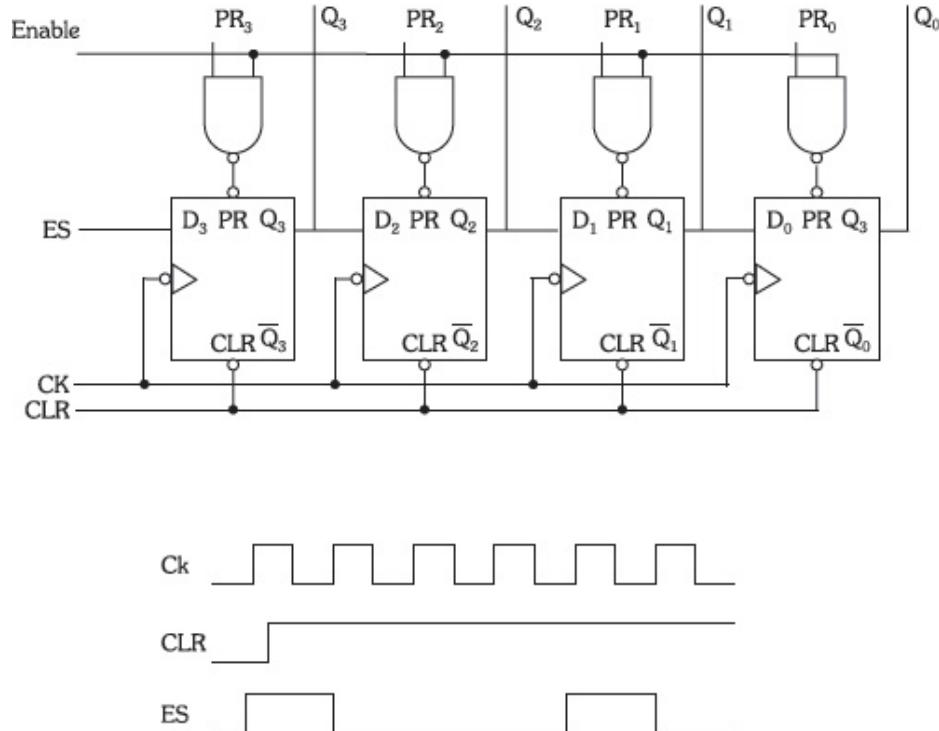


Figura 4.49 - Registrador e formas de onda do exercício 10.

- 11) Determine a situação das saídas Q<sub>3</sub>, Q<sub>2</sub>, Q<sub>1</sub> e Q<sub>0</sub> para o circuito do exercício anterior, após três descidas de *clock*, sabendo que PR<sub>3</sub> = 1, PR<sub>2</sub> = 0, PR<sub>1</sub> = 0, PR<sub>0</sub> = 0 e ES = 0, que inicialmente houve a passagem do *clear* de 0 para 1, que o enable passou de 0 para 1 e logo após de 1 para 0.
- 12) No exercício anterior, o que aconteceria se ligássemos a saída Q<sub>0</sub> à entrada ES e, logo após, aplicássemos à entrada *clock* sucessivas descidas de pulsos?
- 13) Projete uma máquina de estados para atuar como contador de 2 *bits* para efetuar a contagem crescente (X = 0 → 0 a 3) ou decrescente (X = 1 → 3 a 0), através de

uma variável de controle X. O circuito deverá também possuir uma entrada M para manter o valor da contagem, mesmo no acionamento sucessivo de *clock* ( $M = 0 \rightarrow$  as saídas mantêm o valor e  $M = 1 \rightarrow$  funcionamento normal conforme X).

- 14) Que modelo a máquina de estado o exercício resolvido, no tópico referente aos projetos de sistemas síncronos, obedece? E o projeto do exercício anterior? Justifique as respostas.

# 5

## Memórias

### Para começar

Este capítulo trata das principais memórias utilizadas e os principais conceitos envolvidos. As tecnologias de fabricação, por fugir da finalidade do livro, não serão abordadas. No campo de desenvolvimento tecnológico há uma evolução constante dos circuitos e materiais empregados, objetivando, principalmente, maiores capacidades de armazenamento e velocidades na transferência de dados.

As memórias encontram um grande emprego no campo da informática, sendo utilizadas principalmente em sistemas computadorizados e periféricos. Nesses tipos de equipamentos, as memórias armazenam dados, programas externos e, ainda, para constituir o conjunto de programas internos para funcionalidade do sistema operacional.

---

## 5.1 Definição geral e aplicações das memórias eletrônicas

---

Memórias são os dispositivos que armazenam informações. Dividem-se basicamente em dois grupos: as memórias de escrita e leitura e as memórias apenas de leitura. São utilizadas em sistemas eletrônicos com microprocessadores e microcontroladores, para a armazenagem de programas e dados em projetos específicos.

Os próximos itens tratam das memórias que armazenam informações codificadas digitalmente que podem representar números, letras, caracteres quaisquer, comandos de operações, endereços ou ainda qualquer outro tipo de dado.

## 5.2 Classificação das memórias

---

Antes de estudarmos os diversos tipos de memórias, vamos conhecer sua classificação. Podemos classificar as memórias em vários itens diferentes. A seguir, vamos relacionar os principais:

- 1) acesso.
- 2) volatilidade.
- 3) troca de dados.
- 4) tipo de armazenamento.

Vamos definir cada item:

### 1) Acesso

As memórias acessam informações em lugares denominados **localidades de memória**. Cada uma das localidades de memória possui um conjunto de *bits* que permite o seu acesso. A esse conjunto de *bits* damos o nome de **endereço**. Esse conceito é de fácil compreensão, pois como o próprio nome diz, o conjunto de *bits* representa o endereço da localidade onde está armazenada uma informação.

O tempo de acesso de uma memória é o tempo necessário desde a entrada de um endereço até o momento em que a informação apareça na saída. Para as memórias de escrita/leitura é também o tempo necessário para a informação ser gravada.

Podemos ter acesso a uma dada localidade de memória de duas maneiras diferentes: **acesso sequencial** e **acesso aleatório**.

As memórias que utilizam o acesso sequencial, dado o endereço de uma determinada localidade, permitem que se chegue até esta, passando por todas as localidades intermediárias. Uma característica importante desse tipo de acesso, é que o tempo de acesso depende do lugar onde a informação está armazenada.

As memórias que utilizam o acesso aleatório, dado um endereço de uma certa localidade, permitem que se chegue até esta diretamente, sem que necessitemos passar pelas localidades intermediárias. As principais memórias com esse tipo de acesso são também conhecidas como **RAM** (*Random-Access Memory*). Possuem a grande vantagem de ter um tempo de acesso pequeno e igual para qualquer uma das localidades de memória. Analisaremos mais adiante o circuito da memória RAM.

## 2) Volatilidade

Quanto à volatilidade, as memórias podem ser **voláteis** ou não **voláteis**.

As memórias voláteis são aquelas que, ao ser cortada a alimentação, perdem as informações armazenadas. São memórias feitas, geralmente, a partir de semicondutores e na maioria das vezes, possuem como elemento de memória o *flip-flop*. Um exemplo típico já citado é o da memória RAM.

As memórias não voláteis são aquelas que mesmo sem alimentação continuam com as informações armazenadas. Entre elas se destacam as memórias eletrônicas: **ROM**, **PROM** e **EPROM**.

## 3) Troca de dados

No que se refere à troca de dados com outros componentes do sistema, as memórias podem ser de escrita/leitura ou memórias

apenas de leitura.

As memórias de escrita/leitura são aquelas que permitem acesso a uma localidade qualquer para armazenar a informação desejada. Além disso, permitem o acesso também para a leitura do dado. As memórias RAM também enquadram-se nessa situação.

As memórias apenas de leitura, como o próprio nome diz, são aquelas em que a informação é fixa, só podendo efetuar-se a leitura. São também conhecidas como ROM (*Read-Only Memory*). A análise desse tipo de memória será feita adiante.

#### 4) Tipos de armazenamento

Quanto ao tipo de armazenamento, as memórias classificam-se em **estáticas** e **dinâmicas**.

As memórias de armazenamento estático são aquelas em que, uma vez inserido o dado em uma dada localidade, ele lá permanece.

As memórias de armazenamento dinâmico são aquelas em que é preciso inserir a informação de tempos em tempos, pois de acordo com as características de seus elementos internos, perdem as informações após um determinado tempo.

As memórias de armazenamento estático apresentam como vantagem a fácil utilização em relação às dinâmicas.

### 5.3 Estrutura geral e organização de uma memória

---

Como vimos, uma memória armazena ou acessa as informações digitais mediante endereçamento, em lugares denominados localidades de memórias. Para o acesso a essas localidades o bloco possui uma série de terminais de entrada de endereços que são ligados a um conjunto de fios denominado **barra de endereços** (*address bus*), sendo este responsável por todo o endereçamento de um sistema típico com microprocessador, por exemplo. Para a entrada e saída dos dados, da mesma forma, o bloco possui uma série de terminais ligados à **barra de dados** (*data bus*). Além disso, o

bloco possui terminais de controle ligados à **barra de controle** (*control bus*). A Figura 5.1 apresenta a esquematização de uma memória eletrônica típica com a ligação dos **barramentos** mencionados mais os terminais de alimentação e terra.

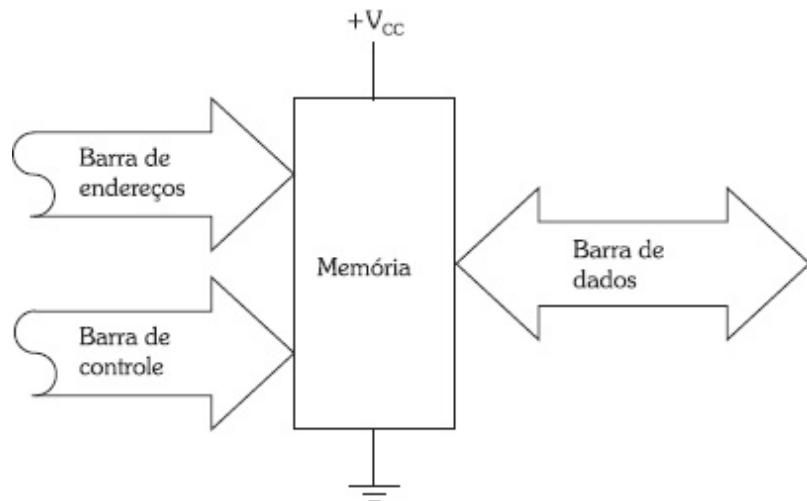


Figura 5.1 - Memória eletrônica típica.

A simbologia utilizada na Figura 5.1 mostra que a barra de dados é **bidirecional**, ou seja, pode ser usada tanto como saída ou entrada de dados, isso nos tipos mais comuns de memórias de escrita/leitura (RAM), sendo um terminal apropriado da barra de controle responsável por este procedimento, estudado detalhadamente mais adiante.

As memórias, de maneira geral, no que se refere à quantidade de dados armazenados, são especificadas pela notação **Nxm**. A primeira letra indica o número de localidades da memória e a segunda indica o número de *bits* da informação armazenada por localidade. Essas especificações caracterizam toda a organização de estrutura da memória. Para exemplificar, vamos relacionar algumas estruturas de memórias usuais na prática:

- » 128x8
- » 1Kx4
- » 64Kx8
- » 2Mx16

Notamos, por estas especificações, que o número de localidades é sempre múltiplo de  $2^n$ , fato derivado da possibilidade total de endereçamento por um determinado número de fios ou terminais ( $n$ ), em situação binária.

A designação **K (Kilo)** que significa um fator  $2^{10} = 1024$ , a **M (Mega)** que significa  $2^{20} = 1048576$  e a **G (Giga)** que significa  $2^{30} = 1073741824$  são muito usuais, principalmente esta última, devido à grande quantidade de memória exigida pelos sistemas digitais. Por exemplo, a memória mencionada anteriormente de  $64\text{ Kx }8$  possui  $64 \times 1024 = 65536$  localidades, com  $8\text{ bits}$  ( $1\text{ byte}$ ) em cada uma, necessitando de 16 terminais para endereçamento. A de  $2\text{M} \times 16$  possui  $2 \times 1048576 = 2097152$  localidades com  $16\text{ bits}$ , necessitando de 21 terminais para endereçamento.

Outro parâmetro a ser definido é o da **capacidade de memória**, que significa o número total de *bits* que podem ser armazenados em uma memória. Para seu cálculo, basta efetuarmos o produto  $N \times m$ , multiplicando o número de localidades pelo número de *bits* por localidade, obtendo assim, a capacidade total em *bits* dessa memória. Por exemplo, uma memória de  $1\text{Kx}4$  possui na totalidade  $4096\text{ bits}$  de capacidade.

Outro ponto importante a ser abordado é em relação à **palavra de endereço**, que é definida como o conjunto de níveis lógicos ou *bits* necessários para o endereçamento de uma determinada localidade de memória para o acesso ao dado. Para facilitar a escrita da palavra de endereço relativa a cada localidade, bem como a sua utilização em programação, é comum transcrever esse conjunto de *bits* diretamente para hexadecimal, principalmente no caso de memórias de alta capacidade, pois esse sistema de numeração permite a representação de cada 4 *bits* utilizando apenas um dígito hexadecimal pela conversão direta.

### Lembre-se

O Sistema Hexadecimal de Numeração possui 16 algarismos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F e sua base é 16.

A Tabela 5.1 apresenta o sistema hexadecimal até o valor 15 no sistema decimal.

Tabela 5.1 - Sistema hexadecimal e equivalência no sistema decimal

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

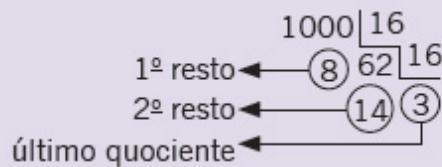
A conversão para o sistema decimal se dá pelo somatório de cada algarismo correspondente multiplicado pela base elevada por um índice, conforme o posicionamento do algarismo no número.

Exemplo:  $1B5_{16} \Rightarrow 1 \times 16^2 + B (11) \times 16^1 + 5 \times 16^0 = 256 + 176 + 5 = 437_{10}$

A conversão contrária do sistema decimal para o binário se dá efetuando-se sucessivas divisões pela base a ser convertida (no caso o 16) até o último quociente possível. O número transformado será composto por este último

quociente (algarismo mais significativo) e todos os restos, na ordem inversa às divisões.

Exemplo:  $1000_{10} \Rightarrow 3E8_{16}$



Conversão direta do sistema hexadecimal para o sistema binário:

A regra consiste em transformar cada algarismo diretamente no correspondente em binário, respeitando-se o número padrão de *bits* do sistema, sendo para o hexadecimal igual a quatro ( $2^4 = 16 \Rightarrow$  base do sistema).

Exemplo:  $C13_{16} \Rightarrow 110000010011_2$

$\underbrace{C}_{1100} \quad \underbrace{1}_{0001} \quad \underbrace{3}_{0011}$

Conversão direta do sistema binário para o sistema hexadecimal:

Para efetuar esta conversão, vamos aplicar o processo inverso ao utilizado na conversão de hexadecimal em binário. Neste caso agrupamos de 4 em 4 *bits* da direita para a esquerda.

Exemplo:  $10011000_2 \Rightarrow 98_{16}$

$\underbrace{1001}_{9} \underbrace{1000}_{8}$

Toda a estrutura com endereços e dados armazenados é frequentemente colocada em uma tabela denominada mapeamento de memória. Para exemplificar, a Tabela 5.2 apresenta o mapeamento de uma memória genérica de 256 localidades.

Notamos que uma memória com 256 localidades precisa de oito fios para endereçamento ( $2^8 = 256$ ), identificados de  $A_7$  até  $A_0$ ,

sendo o endereço da localidade inicial  $00_{16}$  ( $00000000_2$ ) e da final  $FF_{16}$  ( $11111111_2$ ). Supondo que a referida memória possua 8 bits por localidade, ou seja,  $256 \times 8$ , sua esquematização em bloco, com a barra de dados identificada de  $D_7$  até  $D_0$ , é mostrada na Figura 5.2.

Tabela 5.2 - Tabela-verdade de uma memória genérica de 256 localidades

Endereço das localidades em binário								Endereço das localidades em hexadecimal	Localidade	Conteúdo	
$A_7$	$A_6$	$A_5$	$A_4$	$A_3$	$A_2$	$A_1$	$A_0$				
0	0	0	0	0	0	0	0	00	$L_0$	$I_0$	
0	0	0	0	0	0	0	1	01	$L_1$	$I_1$	
0	0	0	0	0	0	1	0	02	$L_2$	$I_2$	
0	0	0	0	0	0	1	1	03	$L_3$	$I_3$	
⋮								⋮	⋮	⋮	⋮
1	0	1	0	0	1	1	1	A7	$L_{167}$	$I_{167}$	
1	0	1	0	1	0	0	0	A8	$L_{168}$	$I_{168}$	
1	0	1	0	1	0	0	1	A9	$L_{169}$	$I_{169}$	
⋮								⋮	⋮	⋮	⋮
1	1	1	1	1	1	0	1	FD	$L_{253}$	$I_{253}$	
1	1	1	1	1	1	1	0	FE	$L_{254}$	$I_{254}$	
1	1	1	1	1	1	1	1	FF	$L_{255}$	$I_{255}$	

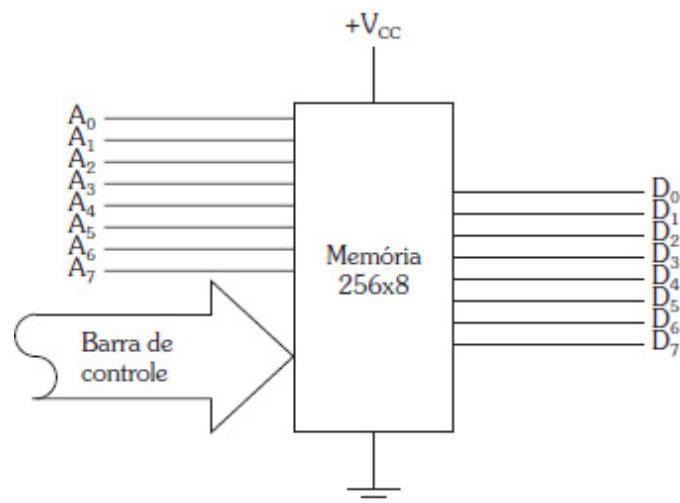


Figura 5.2 - Esquematização de uma memória de 256 x 8.

Nos itens subsequentes, vamos estudar os principais tipos de memórias e suas respectivas arquiteturas.

## 5.4 Memórias ROM

As memórias ROM, conforme já visto no item relativo à classificação, apresentam como característica principal permitir somente a leitura dos dados nelas gravados previamente em sua fabricação. Vem daí o nome ROM (Read-Only Memory), que significa memória apenas de leitura. Além disso, possuem acesso aleatório e são não voláteis, pois não perdem seus dados armazenados com o desligamento da alimentação. Na realidade, as memórias ROM podem ser consideradas circuitos combinacionais, pois apresentam as saídas de dados em função das combinações entre as variáveis de entrada (endereçamento).

Dentre as diversas aplicações destacamos a sua utilização para armazenar programas de sistemas operacionais em computadores e outros sistemas digitais. A Figura 5.3 mostra o bloco representativo de uma memória ROM, com terminais e barramentos conhecidos e mais um terminal de controle ( $\overline{CS}$ ), para **habilitação da pastilha ou chip**.

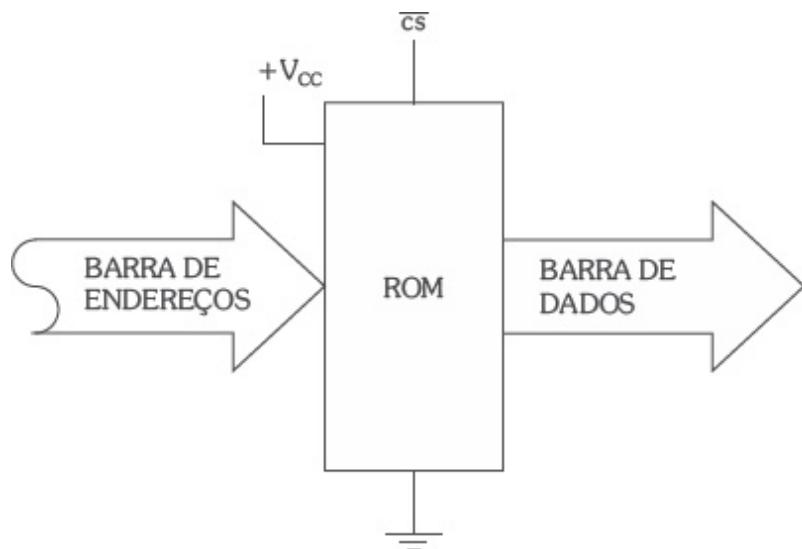


Figura 5.3 - Memória ROM.

O terminal de controle para habilitação ou seleção da pastilha,  $\overline{CS}$  (*chip select*), é na realidade uma entrada de nível lógico para ativar ou não as saídas da ROM. Se aplicarmos a essa entrada um nível 0, as saídas são habilitadas, ou seja, são internamente comutadas para fornecer os dados, conforme funcionamento normal de endereçamento, porém se aplicarmos um nível 1, elas são desabilitadas, ficando as saídas desativadas (em *tri-state*), liberando a barra de dados para utilização por outros dispositivos presentes no sistema controlado normalmente por microprocessador. O traço sobre CS ( $\overline{CS}$ ) indica que a habilitação da pastilha é feita com nível 0 (ativa em 0), sendo esta uma forma de nomenclatura muito utilizada na prática. Na série de circuitos integrados comerciais é também encontrado o termo *chip enable* ( $\overline{CE}$ ), tendo a mesma funcionalidade.

A escolha da ativação por nível 0 deve-se também ao fato de ela proporcionar maior **imunidade ao ruído**, pois em situação contrária, haveria maior susceptibilidade para o acionamento dos blocos dentro do sistema, frente a esse fator transiente indesejável.

### Amplie seus conhecimentos

É denominado *tri-state* (*3-state*) o estado caracterizado por uma desativação do circuito interno, fazendo a saída do bloco comportar-se como circuito aberto ou de *alta impedância* (conceito utilizado em eletrônica), ficando esta saída desligada do sistema ao qual está inserida.

## 5.4.1 Arquitetura interna das memórias ROM

Uma memória ROM pode ser construída de inúmeras maneiras, porém vamos estudar a arquitetura básica utilizada, principalmente, nos processos de fabricação dos circuitos integrados atuais. A Figura 5.4 apresenta em blocos a arquitetura básica de uma ROM genérica, com os respectivos terminais e barramentos de entrada e saída.

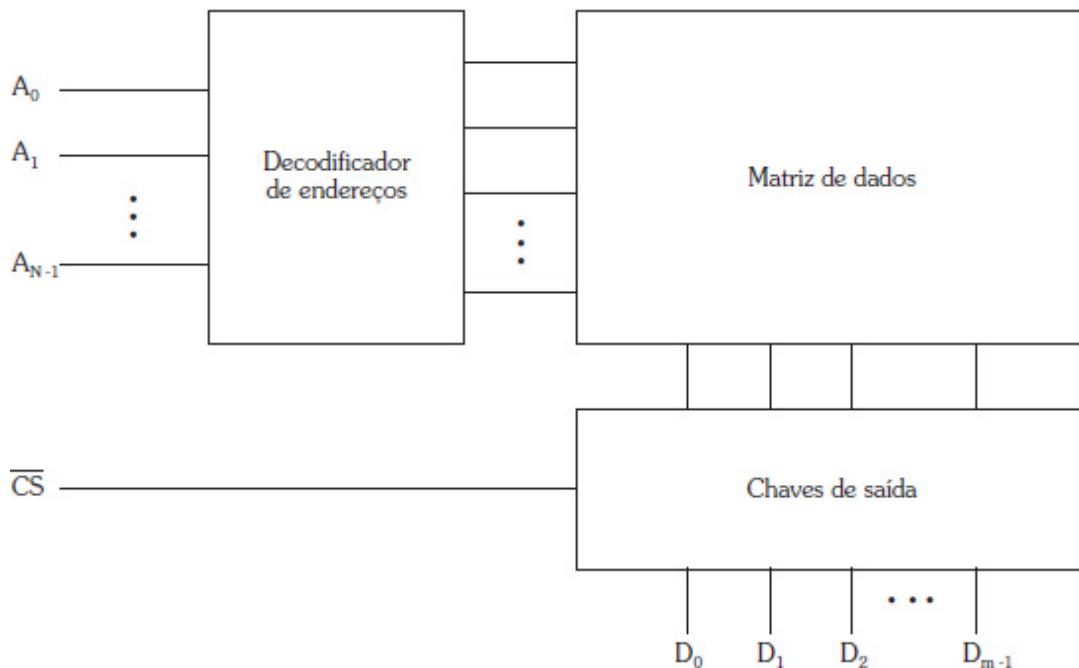


Figura 5.4 - Arquitetura básica de uma ROM genérica.

O primeiro bloco consiste em um **decodificador de endereços**, que nada mais é que um circuito decodificador responsável por ativar (fornecer nível 1) um fio de saída por vez, em função do endereçamento.

O segundo bloco é constituído por uma **matriz de dados**, que é um arranjo de linhas e colunas que, através de um elo, possibilita a gravação dos dados pelo fabricante e consequente leitura pelo usuário. Na prática, dentre as várias tecnologias de construção, utilizam-se para a formação desses elos elementos semicondutores, que vão se constituir na estrutura de dados propriamente dita.

Para a saída dos dados, a memória possui um conjunto de chaves (*buffers*) que, conforme habilitação através do terminal  **$\overline{CS}$** , possibilita a conexão das saídas (terminal  $\overline{CS} = 0$ ), ou as deixa em aberto ou desligadas (terminal  $\overline{CS} = 1$ ), desconectando-as da barra de dados do sistema.

Para facilitar o processo de programação pelo fabricante, ele utiliza um gabarito fotográfico das ligações elétricas chamado '**máscara**', sendo as memórias assim confeccionadas denominadas

ROMs Programadas por Máscara (*Mask-Programmed Read-Only Memory*).

As memórias ROM são produzidas com programações fixas para aplicações determinadas e sob encomenda, apenas em grande quantidade, normalmente para clientes específicos e fabricantes de equipamentos. Uma solução para o pequeno usuário é a utilização das ROMs programáveis (PROM e EPROM), estudadas nos itens a seguir.

## 5.5 Memórias PROM

---

As memórias PROM (*Programmable Read-Only Memory*) permitem o armazenamento dos dados pelo próprio usuário, porém feito de modo definitivo. Após essa programação, a memória PROM transforma-se em uma ROM, devendo, portanto, ser utilizada como tal.

O princípio básico da programação ou armazenamento de dados em uma PROM é destruir, através de nível de tensão conveniente especificado pelo fabricante, as pequenas ligações semicondutoras existentes internamente nas localidades onde se quer armazenar a palavra de dados, conforme endereçamento feito. O roteiro para tanto é fornecido pelo fabricante. Na prática, existem disponíveis sistemas apropriados (*kits* ou placas) para realizá-lo conforme o tipo de pastilha, com maior eficiência e rapidez. Devemos realçar que após a programação o processo é irreversível, não sendo possível nenhuma alteração.

Esse tipo de memória recebe, da mesma forma que a ROM, a classificação de não volátil, acesso aleatório e de apenas de leitura, pois apesar da programação prévia para a funcionalidade do sistema no qual vai ser utilizada, só vai permitir a leitura de dados.

## 5.6 Memórias EPROM

---

Com o avanço da tecnologia foram criadas as memórias EPROM (*Erasable Programmable Read-Only Memory*), ROM programável e

apagável, que permitem a programação de modo semelhante à das PROMs, com a vantagem de poderem ser normalmente apagadas mediante banho de luz ultravioleta, efetuado pela exposição da pastilha por uma janela existente em seu encapsulamento e, ainda, serem reprogramadas. São também conhecidas como UV PROM (*Ultraviolet PROM*). Da mesma forma, após a programação, essa memória transforma-se em uma ROM, recebendo os mesmos itens de classificação.

Convém ressaltar que o apagamento dos dados ocorre de maneira simultânea e compacta para o programa inteiro, sendo necessária a regravação total do programa em caso de modificações por mais simples que sejam.

Existem disponíveis comercialmente vários tipos de EPROMs com diversas capacidades de armazenamento. Para exemplificar, mostrar a terminologia e a função dos terminais dos barramentos, sobretudo os básicos de controle, a Figura 5.5 apresenta o bloco de uma EPROM do tipo mais comum, estruturada em 2Kx8.

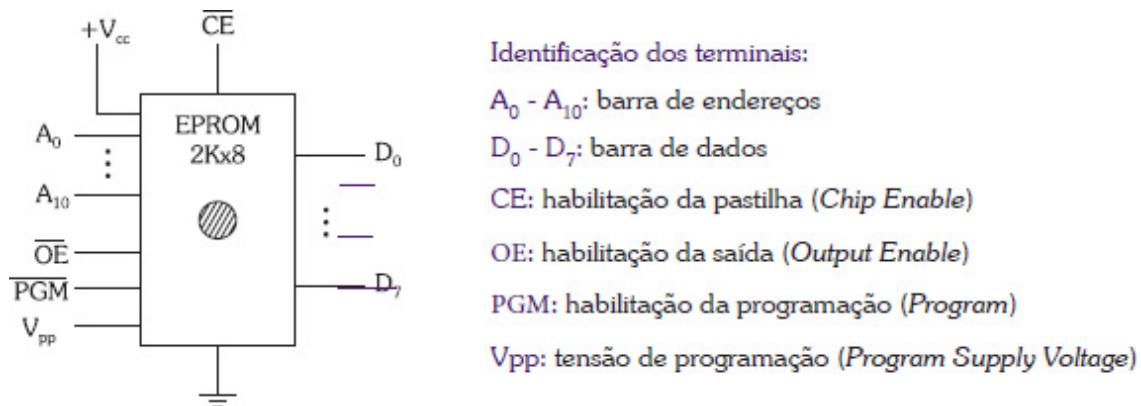


Figura 5.5 - Memória EPROM 2Kx8.

Conforme a capacidade dessa memória (2Kx8), para o acesso das localidades é necessário 11 fios ( $2^{11} = 2048 = 2K$ ) e oito para a barra de dados.

O terminal de habilitação  $\overline{CE}$  tem a função de ativar o bloco através de nível 0, e quando em nível 1 o deixa desativado, na situação de baixo consumo de potência (*standby*). A entrada de controle  $\overline{OE}$  tem a função de habilitar ou desabilitar apenas o

barramento de saída, sendo da mesma forma a habilitação em nível 0.

Para a programação dos dados, o bloco dispõe de um terminal ( $V_{pp}$ ) que, recebendo uma tensão específica, sendo o valor fornecido pelo fabricante, é responsável juntamente com o terminal  $\overline{PGM}$  pelo armazenamento das informações. O processo se realiza mediante a aplicação da tensão em  $V_{pp}$  (tipicamente um valor maior que  $V_{cc}$ ), da habilitação da programação ( $\overline{PGM}$ ) através de nível 0, do endereçamento e da aplicação das respectivas palavras de dados ao bloco, sequencialmente, conforme a listagem do programa a ser armazenado.

O apagamento do programa pode ser feito pela exposição do bloco à luz ultravioleta durante 15 a 50 minutos, também conforme especificação do fabricante, em função da potência da lâmpada utilizada. Após o apagamento, todas as localidades assumem níveis 1, podendo o processo de regravação e apagamento repetir-se por inúmeras vezes.

Na prática, da mesma forma, existem disponíveis sistemas apropriados (*kits* ou placas) para realizar o processo de gravação conforme o tipo de EPROM, com maior eficiência e rapidez. Existem também sistemas denominados apagadores de EPROM, constituindo uma caixa vedada, com lâmpada ultravioleta e sistema de cronometragem programada, que conforme o fabricante e a especificação da EPROM efetuam o processo de apagamento automaticamente.

## 5.7 Memórias EEPROM

---

As memórias de nome **EEPROM** ou **E<sub>2</sub>PROM** (*Electrically Erasable Programmable Read-Only Memory*) são um avanço tecnológico em relação às EPROMs estudadas, pois permitem que o apagamento dos dados seja feito eletricamente e, ainda, isoladamente por palavra de dados, sem necessidade de reprogramação total. Esse fato faz com que as alterações de programação sejam efetuadas pelo próprio sistema no qual a memória esteja inserida, sem necessidade de desconexão do circuito.

integrado, como no caso da EPROM. Para ilustrar esta apresentação, a Figura 5.6 apresenta o bloco de uma E<sup>2</sup> PROM de tipo comum, estruturada em 8Kx8.

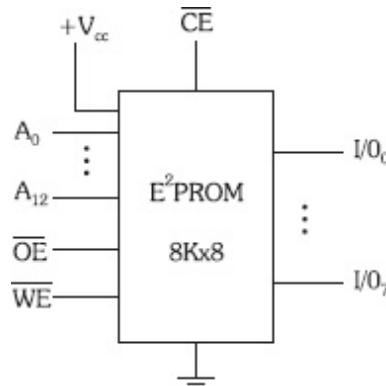


Figura 5.6 - Memória E<sup>2</sup>PROM 8Kx8.

Notamos pela figura que para o acesso das localidades dessa memória é necessário 13 fios ( $2^{13} = 8192 = 8K$ ). Notamos ainda que devido à possibilidade de escrita e leitura pelos mesmos terminais, a barra de dados passa a ter a característica de bidirecional, recebendo a terminologia de I/O (*Input/Output*), muito comum nestes casos.

A escrita de uma palavra de dados, alterando a programação, é obtida pelo endereçamento e a respectiva aplicação da palavra nos terminais da barra de dados, isso com o terminal OE em nível 1, e o de habilitação da escrita WE (*Write Enable*) em nível 0, dentro de um ciclo de tempo mínimo, especificado pelo fabricante do circuito integrado. Esta memória também é não volátil e possui acesso aleatório.

#### Lembre-se

Em nível de classificação, a memória **E<sup>2</sup>PROM** pode causar polêmica em um item, pois apesar de permitir a escrita e leitura de dados, faz parte da família das memórias apenas de leitura (ROM). O nome EEPROM, no entanto, deve ter sido atribuído por questões históricas do desenvolvimento tecnológico na área, o mesmo ocorrendo com outras memórias.

## 5.8 Memórias Flash

---

Com o avanço tecnológico no campo dos semicondutores, foram desenvolvidos outros arranjos das estruturas internas das memórias, visando possibilitar o armazenamento de um maior número de dados. Isso é obtido devido à maior densidade (concentração) de componentes internos, responsáveis por esse armazenamento, para a mesma área utilizada por *chip*. As memórias *flash*, por possuírem estas características, são definidas como sendo de alta densidade. Além disso, são memórias que possibilitam que as operações de escrita/leitura sejam feitas pelo próprio sistema em que elas estão inseridas e não são voláteis, ou seja, podem manter seus dados armazenados, mesmo sem alimentação.

Em termos de funcionalidade dos terminais para a escrita/leitura e apagamento, as memórias *flash* assemelham-se às E<sup>2</sup>PROM, porém com maiores velocidades de acesso para a transferência de dados.

As memórias *flash* são utilizadas em computadores, periféricos e aparelhos eletrônicos de áudio, fotografia, vídeo e telefones celulares. Devido às suas características, têm grande emprego em sistemas de armazenamento de dados removíveis, tais como cartões de memória e *pen-drives*.

## 5.9 Memórias RAM

---

As memórias RAM, conforme já mencionado, permitem escrita e leitura dos dados e possuem acesso aleatório ou randômico. Vem daí o nome RAM (*Random-Access Memory*). Além disso, são voláteis, pois perdem seus dados armazenados com o desligamento da alimentação. Possuem ainda um tempo de acesso muito reduzido, sendo usadas em equipamentos digitais principalmente como memórias de programas e dados para armazenamento de forma temporária, pois em função de volatilidade, estes são perdidos no desligamento ou interrupção da energia.

Quanto ao armazenamento, são encontradas as estáticas (**SRAM: Static RAM**) ou dinâmicas (**DRAM: Dynamic RAM**). As RAMs

estáticas utilizam como célula básica de memória o *flip-flop*, possuindo em sua arquitetura vários elementos. Já as dinâmicas possuem circuitos mais simples, porém necessitam de reinserção de dados periódica em ciclo, na prática denominada *refresh* (termo em inglês que significa “refrescar”), sendo essa operação controlada pelo microprocessador do sistema. A célula básica da RAM dinâmica armazena cada dado por efeito capacitivo do pequeno semicondutor formado internamente. Por este motivo, apresenta como vantagem a alta capacidade de armazenamento por circuito integrado.

A Figura 5.7 apresenta o bloco representativo de uma memória RAM estática, com terminais e barramentos já conhecidos e mais um terminal de controle  $R/\bar{W}$  (Read/Write) de dupla função para possibilitar a leitura ( $R/\bar{W} = 1$ ) ou escrita ( $R/\bar{W} = 0$ ) dos dados nas localidades endereçadas.

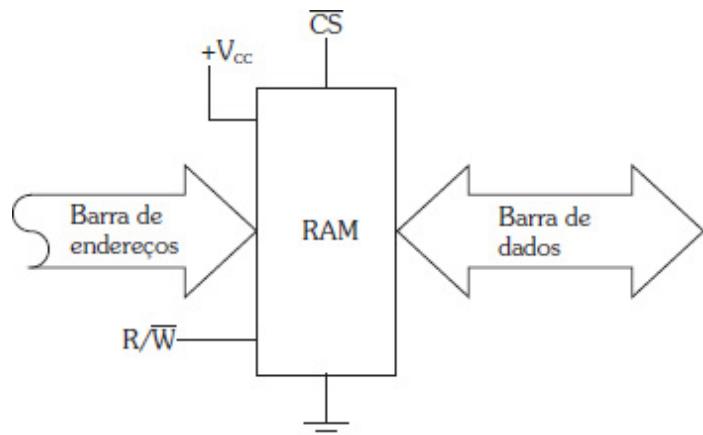


Figura 5.7 - Memória RAM estática.

Para entendermos o funcionamento básico de uma RAM estática, vamos inicialmente analisar o circuito de uma célula básica que permite a escrita e leitura de 1 *bit* de informação. Esse circuito é visto na Figura 5.8.

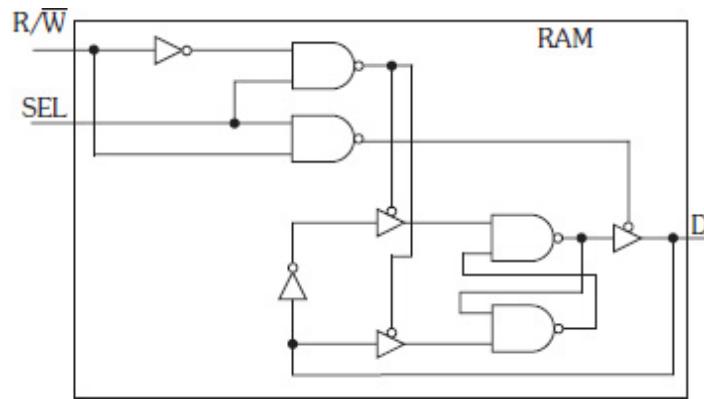


Figura 5.8 - Célula básica de uma RAM estática.

No circuito, além de portas NE e inversores, notamos pela simbologia utilizada, a presença de chaves controladas (*buffers*). Para melhor esclarecimento da atuação desses elementos, a Figura 5.9 mostra um *buffer* (a) e seu circuito equivalente (b), sendo sua atuação transcrita para a Tabela 5.3.

Tabela 5.3 - Tabela de atuação do buffer

M	Ch	E	S
0	Fechada	0	0
		1	1
1	Aberta	X	Desativada

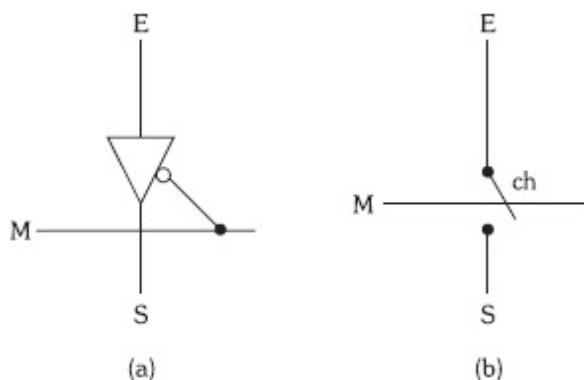


Figura 5.9 - Buffer (a) e circuito equivalente (b).

Pela tabela, notamos que a saída do dado aplicado a entrada E é controlado pelo nível presente no terminal M (se  $M = 0 \Rightarrow S = E$  e  $M$

$= 1 \Rightarrow S$  é desativada). A presença do círculo no símbolo do *buffer*, significa que sua habilitação com chave fechada ocorre em nível 0 (ativo em 0).

Para efetuar a escrita de um dado, devemos primeiramente selecionar a célula ( $SEL=1$ ) e passar o controle de leitura/escrita ( $R/W$ ) para 0. Logo após, aplicamos o dado no terminal D, agora configurado como entrada.

A Figura 5.10 mostra a célula básica com todas essas situações colocadas e, ainda como exemplo, a aplicação para armazenamento de nível 1 na entrada D.

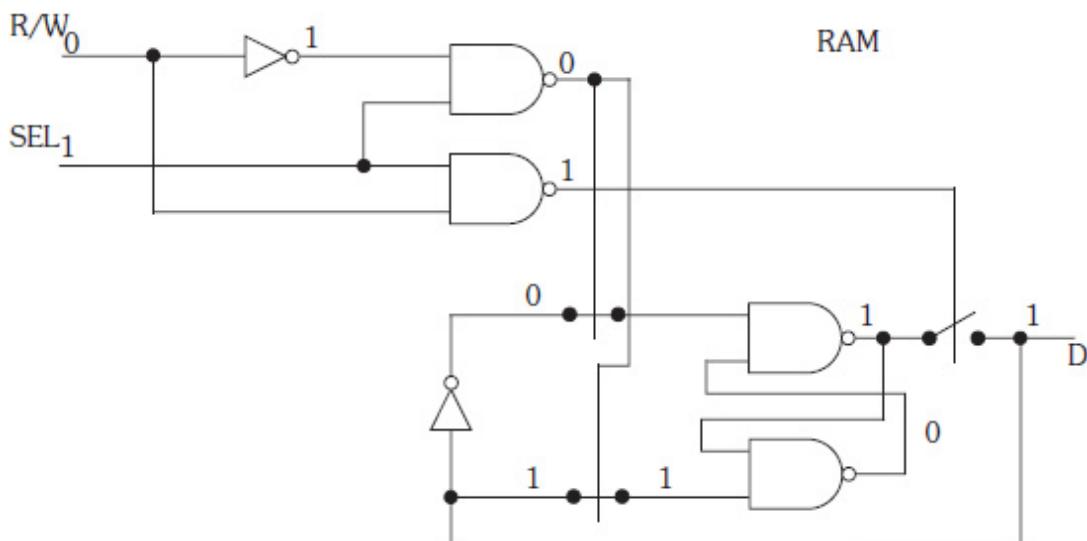


Figura 5.10 - Célula básica de RAM estática na situação de escrita de um dado.

Pela figura notamos que a porta NE superior, através de nível 0 em sua saída, ativa as duas chaves (*buffers*), aqui substituídas pelos circuitos equivalentes, fazendo o dado ser aplicado ao *flip-flop* e, consequentemente, ser armazenado na saída. Enquanto isso, a outra porta, através de nível 1 em sua saída, desativa a chave de saída, permitindo a escrita ou entrada do dado.

Para efetuar a leitura, devemos também selecionar a célula ( $SEL=1$ ) e passar o controle  $R/W$  para 1, sendo obtido o dado armazenado pelo terminal D, agora configurado como saída. A Figura 5.11 mostra essa situação colocada no esquema.

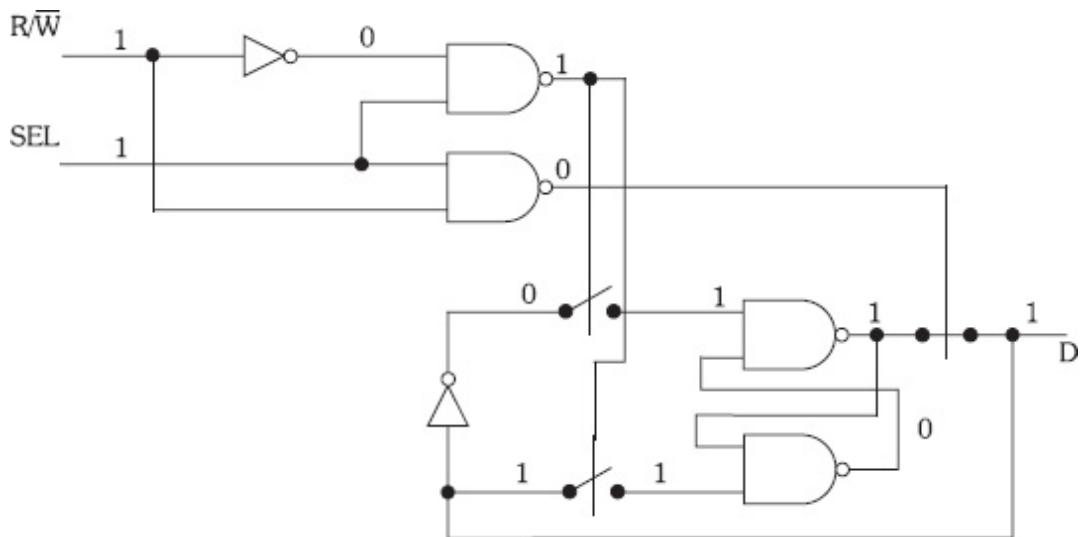


Figura 5.11 - Célula básica de RAM estática na situação de leitura de um dado.

Na realidade, dentro dos circuitos integrados, são construídas células básicas com diversas configurações tecnológicas e de circuitos, sendo esta apresentada devido ao seu enorme caráter didático. Nos itens seguintes, para facilitar, utilizamos para desenvolver a arquitetura interna desse tipo de memória células genéricas representadas em blocos. A Figura 5.12 mostra o bloco padrão representativo da célula da memória RAM, sendo sua atuação resumida na Tabela 5.4.

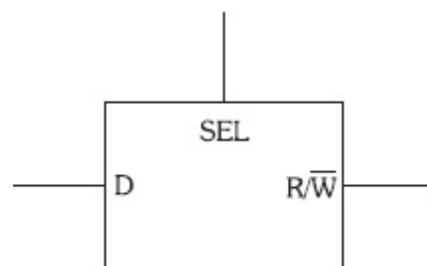


Figura 5.12 - Bloco representativo.

Tabela 5.4 - Atuação da célula da RAM

SEL	R/W	D
0	X	Desativada
1	0	Escrita

## 5.9.1 Arquitetura interna das memórias RAM

Utilizando a célula básica padrão analisada no item anterior, podemos construir arquiteturas de memórias RAM estáticas no formato  $N \times m$ . Para exemplificar, a Figura 5.13 apresenta a arquitetura de uma RAM de estrutura  $4 \times 4$ .

Uma RAM assim especificada possui quatro localidades com 4 *bits* cada.

O circuito, como se observa, é constituído por um decodificador de endereços com dois fios ( $A_1$  e  $A_0$ ), responsável pelo endereçamento de cada localidade definida pelo conjunto das quatro células interligadas horizontalmente. Os terminais de dados (D) estão também interligados, porém por posicionamento do *bit* na palavra de dados, pois no endereçamento de cada conjunto através das entradas SEL, os outros não endereçados adquirirem nos terminais D a situação de desativação, sendo desconectados do fio em comum. Além disso, todas as entradas R/W encontram-se interligadas para propiciar um controle simultâneo da escrita ou leitura para todas as localidades.

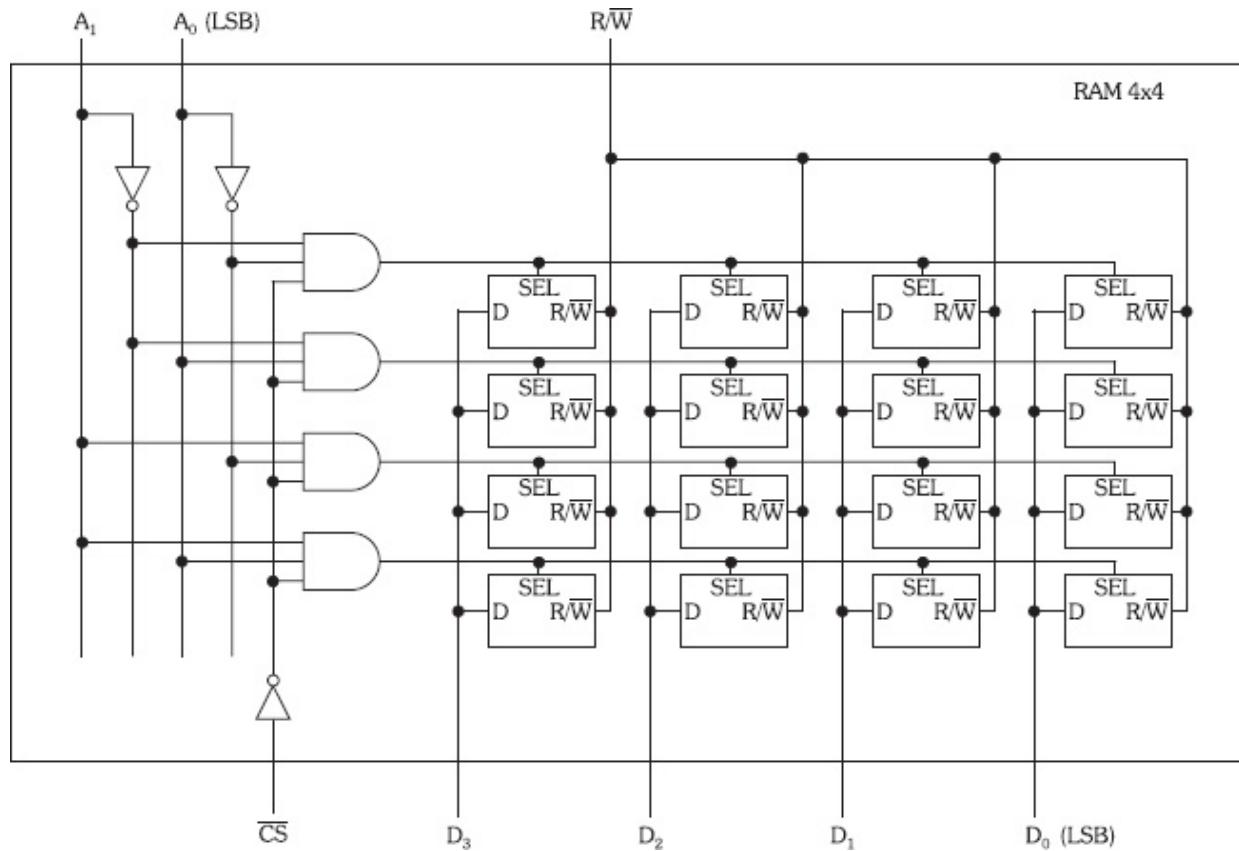


Figura 5.13 - Arquitetura interna de uma RAM 4 x 4.

Para mostrar o funcionamento dessa memória, vamos primeiramente efetuar o armazenamento (escrita) do dado  $5_{16}$  ( $0101_2$ ) na localidade  $1_{16}$ , endereçada por 01.

Inicialmente, estando a pastilha não selecionada ( $\overline{CS} = 1$ ), o nível 0 na entrada das portas E após o inverter ocasiona o aparecimento de nível 0 na saída delas, fazendo todas as células de memória ficarem desativadas ( $SEL = 0 \Rightarrow D$  desativado).

Feita a seleção da pastilha ( $\overline{CS} = 0$ ) e o endereçamento da localidade ( $A_1 = 0$  e  $A_0 = 1$ ), o segundo fio superior na saída da porta E, mediante nível 1, seleciona todas as células da linha ( $SEL = 1$ ). Com o controle  $R/W$  em 0 (escrita), aplicamos os dados nos respectivos terminais, agora configurados como entradas ( $D_3 = 0$ ,  $D_2 = 1$ ,  $D_1 = 0$  e  $D_0 = 1$ ), sendo armazenados pelas células.

Com a passagem de  $R/W$  para 1, para posterior leitura, os dados vão permanecer armazenados, mesmo na reversão da seleção da

célula com  $\overline{CS}$  passando para nível 1. Devemos ressaltar ainda que a informação será perdida, caso se desligue a tensão de alimentação.

O mesmo processo de escrita pode ser estendido para outras localidades, bastando endereçar, passando  $R/W$  para 0 e aplicando respectivamente a informação de dados nas entradas D.

Para a leitura de uma informação, devemos selecionar a pastilha ( $\overline{CS} = 0$ ) e com  $R/W$  igual a 1, endereçar a localidade, obtendo a informação de dados nos terminais D, agora configurados como saídas.

A Figura 5.14 mostra a representação em bloco da RAM utilizada no exemplo.

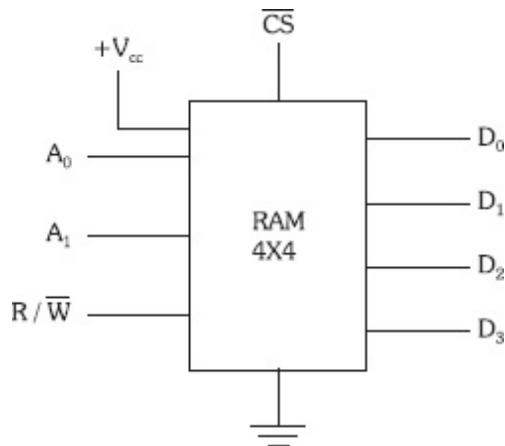


Figura 5.14 - Bloco da RAM 4 x 4.



## Exercícios resolvidos

- 5.1) Indique a palavra de endereço inicial e final de uma memória de 1Mx16.

Com esta especificação, a memória vai possuir  $2^{20} = 1048576$  localidades com 16 bits, endereçadas de 0 a  $1048575_{10}$ . Em binário, temos:  $1048576 = 2^{20} = 1000000000000000000000000$

A última localidade é  $1048575 = 111111111111111111111111$ .

Transformando diretamente em hexadecimal, temos  $FFFF_{16}$ .

∴ A palavra de endereço inicial é  $00000_{16}$ , e a final  $FFFFF_{16}$ .

- 5.2) Determine o mapeamento de uma memória ROM com o seguinte conteúdo:  $1E_{16}$ ,  $8A_{16}$ ,  $0D_{16}$  e  $76_{16}$ . Esquematize o bloco e calcule, ainda, a capacidade de memória.

Nesta situação, a ROM deverá possuir 4 localidades com 8 bits em cada uma. A partir das informações, vamos montar a tabela com o mapeamento da memória e esquematizar o bloco:

Tabela 5.5 - Mapeamento da memória ROM a partir do conteúdo solicitado

Endereço		Hex	Dados							
A <sub>1</sub>	A <sub>0</sub>		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	0	1E	0	0	0	1	1	1	1	0
0	1	8A	1	0	0	0	1	0	1	0
1	0	0D	0	0	0	0	1	1	0	1
1	1	76	0	1	1	1	0	1	1	0

A Figura 5.15 mostra o bloco desta ROM esquematizado.

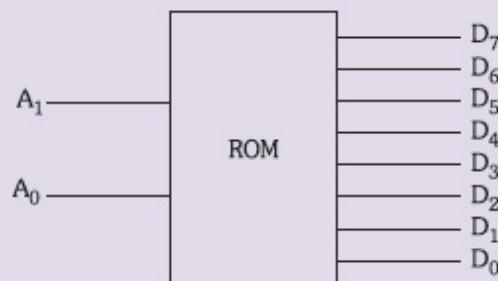


Figura 5.15 - Representação do bloco da ROM do exercício 2.

Por possuir 4 localidades de 8 bits, esta memória é especificada por 4x8 e sua capacidade será de 32 bits.

Vamos recapitular?

Neste capítulo, você conheceu os principais tipos de memórias eletrônicas existentes.

Conheceu os conceitos referentes à classificação: acesso, volatilidade, troca de dados e tipo de armazenamento, para estabelecer uma comparação entre os tipos apresentados e aplicações.

Você conheceu a arquitetura interna de uma ROM e as características da PROM, EPROM e E<sup>2</sup>PROM, e suas formas de escrita e leitura de dados.

As vantagens e aplicações das memórias *Flash*.

Compreendeu o funcionamento de célula básica de uma RAM e sua arquitetura interna para compor uma memória estática. As operações de escrita e leitura de dados.



## Agora é com você!

- 1) Determine a capacidade de memória e a palavra de endereço inicial e final para cada memória especificada a seguir:
  - a) ROM 512 x 4
  - b) EPROM 4K x 8
  - c) RAM 128K x 8
  - d) RAM 2M x 16
- 2) Determine o mapeamento de uma ROM com o seguinte conteúdo:  $01_{16}$ ,  $3F_{16}$ ,  $23_{16}$ ,  $4B_{16}$ ,  $56_{16}$ ,  $88_{16}$ ,  $9C_{16}$  e  $ED_{16}$ . Esquematize o bloco e calcule, ainda, a capacidade de memória.

- 3) Determine o mapeamento de uma PROM para atuar como gerador de caracteres para hexadecimal, ou seja, a partir de um código binário, forneça os níveis para fazer um *display* de 7 segmentos catodo comum apresentar a sequência do sistema hexadecimal. Especifique a memória e determine sua capacidade.
- 4) Utilizando apenas portas NOU e inversores, modifique o circuito da célula básica da memória RAM estudada para executar as mesmas funções. Considere que nas portas utilizadas cada terminal em vazio equivale a nível lógico 1.
- 5) Desenhe a arquitetura interna de uma RAM 8 x 1.
- 6) Elabore um quadro comparativo entre as todas as memórias apresentadas nos quesitos referentes a acesso, volatilidade, troca de dados e tipo de armazenamentos.

# 6

## Introdução aos Computadores Digitais

### Para começar

Este capítulo traz uma abordagem introdutória e os principais elementos e conceitos envolvidos no assunto referente aos computadores digitais ou microcomputadores.

Dentro da área de sistemas digitais, caracterizada como *hardware* de computadores, o assunto é bem complexo, envolvendo a interligação de muitos sistemas, tais como microprocessadores, memórias e outros circuitos integrados periféricos que, para atuar nos sistemas computadorizados, requerem maiores especificações nas características de capacidade e velocidade.

Além disso, os computadores usam *softwares* próprios, que trazem a operacionalidade necessária para o controle e

transferência de informações internas e possibilitam a instalação de programas aplicativos para utilização externa.

## 6.1 Computador digital

---

Podemos definir um computador como sendo um sistema digital sequencial programável. Basicamente um computador é um dispositivo que recebe informações de entrada, efetua um processamento interno e as entrega nas saídas, sob vários formatos, adequados conforme a finalidade na qual o sistema se destina.

Na essência, um computador digital, é um manipulador de sinais elétricos no domínio do tempo, que devidamente arranjados, trafegam pelos circuitos internos por vias de comunicação entre os blocos, podendo ser alterados em função das operações próprias, armazenados e transmitidos.

A própria evolução da computação nos mostra que de uma máquina abstrata, para uso apenas de profissionais qualificados, o computador e os sistemas eletrônicos correlacionados, passaram a fazer parte do cotidiano, com uma alta interação com os seres humanos e suas vidas sociais.

Os itens seguintes abordam os itens relativos à organização interna dos computadores.

## 6.2 Estrutura geral e organização de um computador

---

Em suas arquiteturas internas, todos os computadores apresentam blocos e sistemas funcionais básicos, que interligados propiciam a transferência de dados entre eles devidamente controlados por uma unidade central de processamento.

A **Unidade Central de Processamento (CPU)**, termo utilizado em inglês, *Central Processing Unit*), é responsável pelo controle de todo o sistema. Sua função elementar é buscar as instruções dos

programas armazenados e executá-las sequencialmente, acessando e transferindo dados para outros blocos funcionais, sendo os principais as memórias e os dispositivos que cuidam das entradas e saídas de dados (E/S) com o meio exterior.

A Figura 6.1 mostra a estrutura básica de um computador digital.

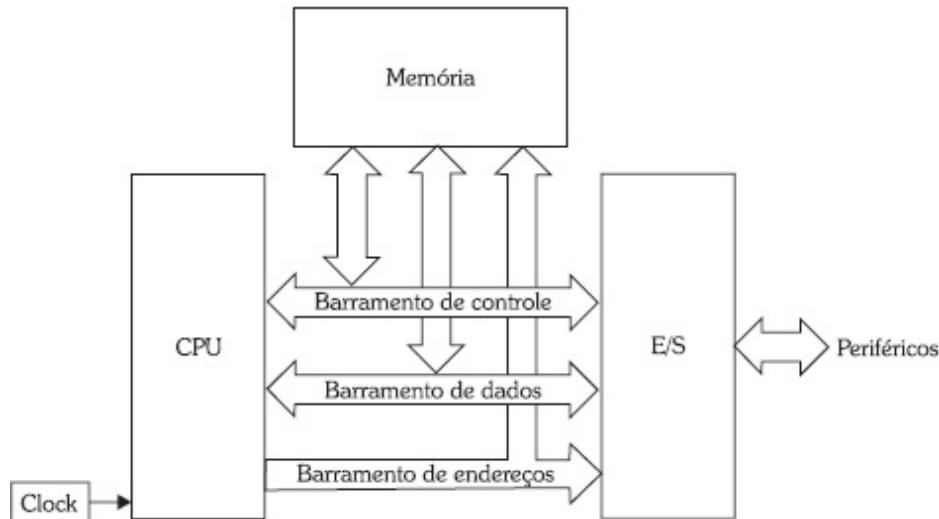


Figura 6.1 - Estrutura básica de um computador.

Pela figura, notamos que sob o comando de um sinal de *clock*, a CPU se comunica com os outros blocos, via barramentos, que, conforme visto no Capítulo 5, nada mais são que um certo número de fios que dispostos em conjunto, que fazem a conexão física para a transferência dos sinais dos blocos internos.

Em razão dos terminais de entradas e saídas dos blocos ligados aos barramentos, possibilitarem a situação de *tri-state* (desativação, fazendo as saídas dos blocos comportarem-se como circuito aberto), a CPU pode, através do barramento de controle habilitar um bloco selecionado, utilizando o terminal  $\overline{CS}$  (*chip select*) ou *chip enable* ( $\overline{CE}$ ) presente em cada bloco, trocar dados de modo unidirecional ou bidirecional com um dispositivo por vez. Diante disso, de maneira geral, os barramentos são utilizados como vias de acesso comum a todos os blocos do sistema, sob a coordenação da CPU. Este fato simplifica muito o traçado do circuito, em caso contrário, teríamos um barramento específico para cada bloco integrante do sistema e

consequentemente mais conexões e terminais nos circuitos integrados.

Nos itens a seguir, vamos abordar cada parte integrante da estrutura básica apresentada e, compreender a funcionalidade geral do sistema.

### 6.2.1 Barramentos internos

Os barramentos internos, também denominados de barramentos locais, são as vias físicas internas utilizadas para a troca de dados entre os blocos do sistema.

O barramento de controle (*control bus*) é utilizado pela CPU para fornecer ou receber sinais controladores necessários à funcionalidade do sistema. Por meio dele, ela pode, por exemplo, habilitar o bloco de memória para transferir dados e comandar as operações de escrita/leitura. Além destas, o barramento pode receber sinais provenientes dos blocos. Assim sendo, o barramento de controle trabalha de modo bidirecional com a CPU (veja a simbologia deste barramento utilizada na Figura 6.1), ou seja, ela pode enviar ou receber sinais.

O barramento de endereços (*address bus*) é utilizado para a CPU endereçar uma localidade (endereço) de um determinado bloco habilitado. Este barramento é unidirecional o endereçamento da localidade requerida é determinado pela CPU.

O barramento de dados (*data bus*) é utilizado pela CPU para trocar dados com o bloco selecionado, portanto é um barramento bidirecional. É também nesse barramento que a CPU vai receber/enviar as informações aos componentes externos, via dispositivos de entrada/saídas de dados (I/O).

### 6.2.2 Unidade central de processamento

A CPU, sob a ação do pulso de *clock*, comanda todas as operações dentro de um sistema computadorizado. Ela busca as instruções dos programas armazenados e as executa sequencialmente para realizar uma determinada tarefa. O processo

consiste em buscar uma instrução e executar, partir para a próxima posição de memória que contém outra instrução e executar, assim por diante, até que o programa termine. Assim sendo, a CPU realiza ciclos de busca, interpretação e execução da instrução. Para cada ciclo destes, a CPU utiliza um número determinado de pulsos de *clock* adequados a cada instrução processada.

Para realizar todo este processamento, a CPU possui internamente uma Unidade Lógica e Aritmética (ULA), um Decodificador de Instruções, uma Unidade de Controle e Registradores, sendo todos conectados por barramentos próprios existentes. Os circuitos integrados processadores ou microprocessadores, citados neste trabalho várias vezes, são a CPU dos atuais microcomputadores e outros sistemas eletrônicos computadorizados. A Figura 6.2 apresenta a estrutura básica de uma CPU.

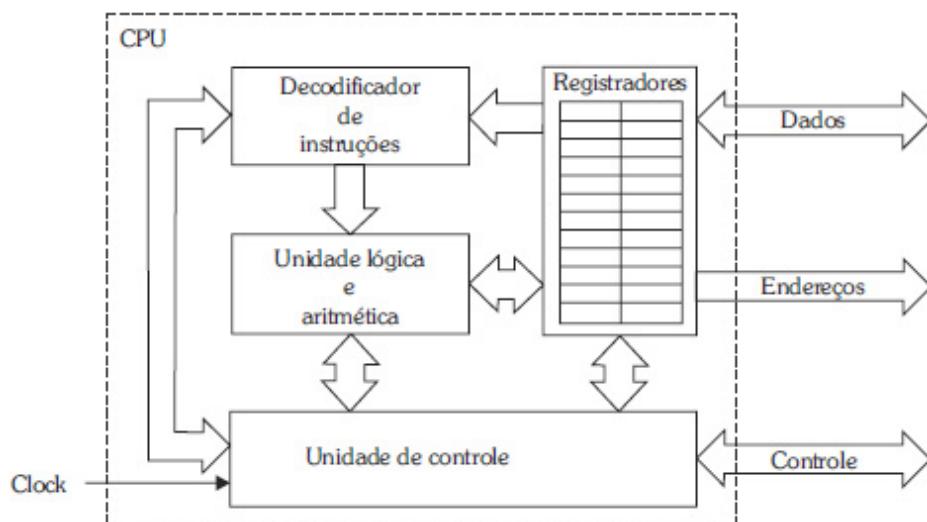


Figura 6.2 - Estrutura básica de uma CPU.

A seguir, vamos descrever a atuação dos principais elementos internos à CPU.

### 6.2.2.1 Unidade lógica e aritmética

A Unidade Lógica Aritmética (ULA ou ALU, *Arithmetic Logic Unit*) é o principal circuito de processamento dentro da CPU. Sua função é

a de realizar todas as operações aritméticas contidas nas instruções (adição, subtração, multiplicação e divisão), operações lógicas convencionais (E, OU, NÃO e OU Exclusivo) e outras operações, tais como incremento, decremento e de deslocamento de *bits*.

Conforme veremos, os *bits* de entrada para que se realizem todas essas operações, são provenientes do decodificador de instruções e dos registradores.

### 6.2.2.2 Decodificador de instruções

O decodificador de instruções tem como atribuição receber cada instrução, na ordem do programa armazenado na memória e decodificá-las (interpretá-las), possibilitando o acionamento da unidade de controle para o processamento e demais operações a serem executadas pela unidade lógica e aritmética.

### 6.2.2.3 Unidade de controle

A unidade de controle estabelece os sinais de controle necessários para a movimentação de dados no sistema a partir do sinal de *clock*. Ela recebe as instruções decodificadas e se encarrega do processamento de dados, sob um regime de temporização imposto pela frequência de *clock*, para a parte exterior da CPU, com a devida sincronização de todas as unidades na movimentação. Além disso, esta unidade se encarrega das operações de controle da ULA e, através do envio de sinais ao barramento de controle, de toda a transferência de dados às memórias e demais dispositivos de entrada e saída.

### 6.2.2.4 Registradores

A CPU possui internamente um conjunto de registradores, que são memórias RAM para registro temporário de endereços e dados utilizados no processamento das instruções do programa. Essa área de armazenamento de informações é utilizada pelas outras unidades internas da CPU, seja para dados referentes aos resultados

temporários ou, ainda, de controle para determinada finalidade. A área permite acesso imediato, possibilitando velocidade e eficiência na execução de um determinado programa.

Entre os registradores existentes, alguns são destinados a uso geral dos programas armazenados como o **Acumulador**, que é utilizado para realizar as operações aritméticas e lógica, registrando, operando e o próprio resultado após a instrução ser realizada. Outros possuem finalidades específicas, dentre os quais destacamos o **Registrador de Instrução (IR, Instruction Register)**, que armazena o código da instrução que está sendo decodificada e o **Contador de Programa (PC, Program Counter)**, que registra os endereços das instruções durante o processamento do programa, definindo a próxima instrução a ser executada.

### 6.2.3 Dispositivos de memórias

O computador para realizar todo o processamento dos dados, necessita armazená-los em vários momentos durante a execução dos programas. Para isso, são utilizados diversos tipos de memórias, sendo cada tipo utilizado de maneira adequada, conforme a classificação requerida, onde o tempo de acesso, tipo de troca de dados (escrita/leitura ou apenas de leitura), volatilidade e volume de dados são itens bastante considerados.

Como vimos no Capítulo 5, as memórias armazenam na essência conjuntos de *bits* (*bytes* ou palavras), que representam endereços, instruções e dados. A organização das memórias internas ao sistema computadorizado obedece basicamente a dois tipos: o de armazenamento permanente de informações para uso do próprio sistema operacional elementar ou de determinados programas instalados, e o armazenamento de dados temporário, que são resultados de operações que podem ou não serem salvas.

O sistema operacional elementar do computador necessita estar armazenado permanentemente em uma memória não volátil. Esta característica é necessária também para fazer o sistema entrar em operação no momento de ligar e cumprir as primeiras etapas de

operações básicas internas. Para isso utilizam-se memórias do tipo ROM.

Essa categoria de memória armazena o sistema operacional, denominado *BIOS (Basic Input/Output System)*, que é o sistema de entrada e saída básico, que atua assim que o computador é ligado, contendo as instruções de inicialização do sistema que são o autoteste e o carregamento do sistema operacional que está no disco rígido (*HD, Hard-Disk*). Outra função do BIOS é o controle do armazenamento dos endereços das rotinas que controlam a participação de unidades periféricas, onde a execução de um programa principal é retida, por um processo denominado de *interrupção*.

Para o armazenamento dos dados temporários que os programas utilizam ou, ainda, outros casos assemelhados, a melhor solução é a utilização de uma área de memórias RAM. Nos sistemas, esta área de memória é normalmente dimensionada com uma capacidade menor em relação à utilizada pelo disco rígido (*HD*). A Figura 6.3 apresenta o esquema simplificado da ligação da CPU às memórias do sistema via barramentos de dados.

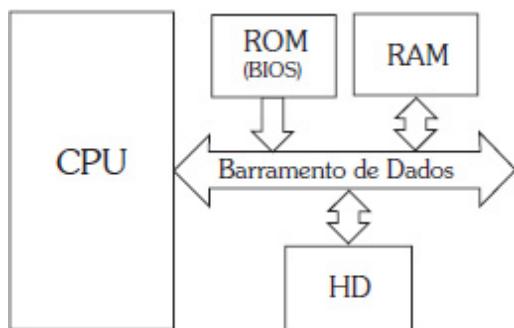


Figura 6.3 - Ligação de uma CPU às memórias via barramento de dados.

As RAMs mais utilizadas são as dinâmicas (DRAM), que conforme já visto, possuem alta capacidade de armazenamento por circuito integrado. Na prática, as DRAMs são arranjadas em conjunto, permitindo de modo compacto, a formação dessa área de RAMs de maior capacidade. As DRAM necessitam de operação de *refresh* que é a reinserção periódica de dados, tratando-se de mais uma rotina realizada pela CPU.

Para ter um acesso mais rápido, pois normalmente o acesso da CPU a essa área de RAMs (memória principal) é mais lento, os sistemas possuem dentro da CPU uma subsecção de RAMs estáticas (SRAM), denominada de '**memória cache**'. Para essa parte limitada de memória, conforme a capacidade, são denominadas de cache de nível 1 ou L1, de nível 2 ou L2 (maior capacidade) ou, ainda, de nível 3 ou L3, podendo, estas últimas, estarem arranjadas em circuitos integrados externos à CPU.

## 6.2.4 Dispositivos de entrada e saída de dados

Como vimos, os dispositivos de entradas e saída (E/S) são responsáveis por trocar dados com o meio externo. A designação também utilizada de *I/O* é proveniente do termo *Input/Output*, da mesma forma que outros, do inglês. Normalmente, como dispositivos de entradas e saídas, encontramos **interfaces padronizadas** com conectores, onde são ligados os **periféricos** básicos obrigatórios mais comuns, que são o teclado, mouse e monitor. Além disso, internamente, são conectadas unidades para receber os dispositivos de armazenamento removíveis, tais como: *CD*, *CD-RW* e *DVD*.

Em sistemas digitais de *hardware* mais simplificados, os dispositivos de E/S recebem, através de circuitos específicos, dados de sensores e outros sistemas, visando a utilização em outras aplicações, controle industrial por exemplo. Um microcontrolador difere do microprocessador, por já incorporar no circuito integrado dispositivos de E/S (*I/O Ports*) e outros elementos, sendo mais apropriado para aplicações dedicadas.

A conexão dos periféricos ao *hardware* dos computadores se dá por meio de interfaces padronizadas. Estas interfaces têm a função de adequar os sinais digitais dos dispositivos e níveis elétricos, possibilitando conexão entre os dispositivos externos e os barramentos internos do sistema.

Existem dois modos de conexão via interfaces padronizadas, por barramento paralelo, cuja entrada/saída dos sinais é feita de maneira

paralela, e por **barramento serial**, onde as entradas/saídas são feitas em modo serial.

Como exemplos tradicionais de utilização do barramento paralelo, podemos citar a ligação de uma impressora ou um monitor de vídeo. No serial, a ligação de um *mouse*.

Um tipo de barramento serial tem substituído as formas barramentos já citadas. Trata-se da **USB** (*Universal Serial Bus*), que significa barramento serial universal. Esta entrada/saída presente nos diversos sistemas computadorizados existentes, possibilita a conexão de periféricos de maneira prática, com um ótimo desempenho no aspecto referente à velocidade de transmissão/recepção. Consiste de um conector característico de fácil acoplamento e a tomada ou porta pode ser instalada em diversos pontos dos gabinetes dos equipamentos, permitindo a ligação de vários dispositivos periféricos. O dispositivo tem também uma saída de alimentação (5 V), permitindo o fornecimento de energia a um elemento conectado.

Outra categoria de conectores, com funcionalidade semelhante ao USB, é o *FireWire*, utilizado por alguns fabricantes de equipamentos e aparelhos de áudio e vídeo, sobretudo, devido as altas taxas de velocidades permitidas.

## **6.3 Software dos computadores**

---

Para comandar os circuitos (*hardware*), um computador necessita de um conjunto de instruções previamente programadas denominadas *software*. De maneira geral, existem dois tipos de *softwares*, os utilizados para finalidade de controle do próprio sistema e os instalados como programas aplicativos.

Os *software* de controle do sistema tem a finalidade de gerenciar o *hardware* nas operações essenciais do sistema, como por exemplo, fracionar o HD para receber programas e dados. Como já se sabe, existem vários sistema operacionais, por exemplo o *Windows*, que são instalados nos computadores servindo como uma interface entre o usuário e o sistema, gerenciando todo o *hardware*.

de maneira prática por ícones e janelas, e ainda apresentando uma série de recursos. O programa também permite o uso simultâneo de vários programas, dando ao sistema uma característica denominada de ‘multitarefa’.

Os softwares que têm funções específicas de aplicação são instalados a partir do gerenciamento do sistema operacional existente. Estes programas também ocupam áreas do HD. Dentre os tipos de aplicações existentes, podemos citar os editores de texto, as planilhas eletrônicas, os gerenciadores de bancos de dados, editores de imagens e outros específicos para as áreas do conhecimento humano e finalidades profissionais.

#### Lembre-se

O sistema operacional (*Windows*, por exemplo) fica instalado no HD e a parte aplicativa para o usuário é carregada na RAM, durante a inicialização do sistema.

Os programas aplicativos também ficam armazenados no HD e quando abertos, também, a parte aplicativa é carregada na RAM.

### 6.3.1 Linguagens de programação

Na essência de um sistema computadorizado, um *software* é traduzido como uma sequência de conjuntos de *bits*, que representam operações, possibilitando o controle de um sistema digital.

No circuito, conforme já estudado, um *bit* é um nível de tensão elétrica que pode assumir dois estados lógicos distintos, o nível 0 (ausência de tensão) e o nível 1 (presença de tensão).

A linguagem que atua diretamente no *hardware*, formada por estes níveis elementares é denominada de ‘*linguagem de máquina*’. Acima desta linguagem, para facilitar a programação pelos projetistas evitando o uso de sequências de *bits* incompreensíveis, está a linguagem *Assembly*, que é constituída de palavras chave adequadas (mnemônicos), conforme o processador utilizado. Esta linguagem é muito usada pelos projetistas de equipamentos

eletrônicos com microcontroladores, para as mais diversas finalidades.

Em um nível acima, sempre em relação ao *hardware*, encontramos outras linguagens trabalhadas para uso nos sistemas de modo geral, denominadas de **linguagens de alto nível**. Como exemplo de linguagens de alto nível, temos as linguagens C e C++, que são aplicadas para o desenvolvimento de programas e projetos. A Figura 6.4 mostra os níveis das linguagens em relação ao *hardware*.

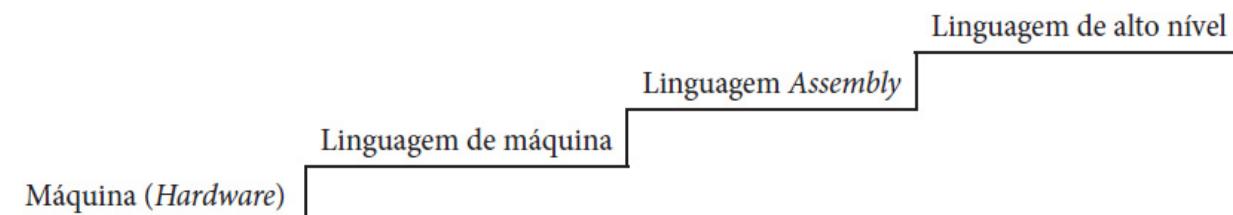


Figura 6.4 - Níveis das linguagens em relação ao hardware.

Na prática, o programa que converte a linguagem de alto nível para a linguagem de máquina é denominado de **compilador** e o programa que converte a linguagem *assembly* para a linguagem de máquina de *assemblador*. O programa em linguagem de alto nível ou *assembly* é denominado de '**programa fonte**' e o convertido em linguagem de máquina é denominado de '**programa objeto**'.

#### Lembre-se

Pelo fato da linguagem *Assembly* estar muito próxima à linguagem de máquina, ela é denominada de '**linguagem de baixo nível**'.

#### Vamos recapitular?

Neste capítulo, você conheceu a estrutura básica interna de um computador digital. Entendeu as definições e finalidades dos barramentos internos do sistema.

Conheceu a estrutura básica de uma CPU e a funcionalidade seus blocos internos.

Entendeu quais são as unidades de memórias utilizadas em um sistema computacional e suas funcionalidades.

Compreendeu as características dos dispositivos de entrada e saída utilizados no sistema e suas interligações com outros periféricos. Verificou também, os tipos de barramentos para efetuar estas ligações.

Verificou os tipos de *software* e as características das linguagens de programação, de alto nível, de baixo nível e de máquina.

Foram definidos compiladores, assembladores, programa fonte e programa objeto.



## Agora é com você!

- 1) Quais são e qual a finalidade de cada barramento interno de um computador?
- 2) Como é possível em um computador todos os blocos internos usarem os mesmos barramentos?
- 3) Quais são as unidades internas de um CPU? Escreva a finalidade de cada uma.
- 4) Escreva a finalidade no sistema:
  - a) BIOS
  - b) HD
  - c) Memória cache
- 5) Cite uma diferença entre microprocessador e microcontrolador.
- 6) O que são interfaces padronizadas?

- 7) Quais os tipos de barramentos existentes? Qual a diferença entre eles?
- 8) Quais as vantagens de utilização do USB?
- 9) O que é multitarefa?
- 10) O que caracteriza uma linguagem de alto nível? E uma linguagem de baixo nível? Cite exemplos.
- 11) Qual a diferença entre compilador e assemblador?
- 12) Qual a diferença entre programa fonte e programa objeto?
- 13) Baseado nos textos, escreva a sequência de operação interna, quando ligamos um computador.
- 14) Em qual memória são alojados os arquivos de um determinado programa aplicativo quando são salvos?

## Bibliografia

---

CAPUANO, F. G. **Exercícios de eletrônica digital.** 3. ed. São Paulo: Érica, 1996.

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

GARCIA, P. A.; MARTINI, J. S. C. **Eletrônica digital.** São Paulo: Érica, 2006.

IDOETA, I. V.; CAPUANO, F. G. **Elementos de eletrônica digital.** 41. ed. São Paulo: Érica, 2012.

NICOLOSI, D. E. C. **Microcontrolador:** detalhado. 8. ed. São Paulo: Érica, 2013.

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **Sistemas digitais:** princípios e aplicações. 10 ed. São Paulo: Pearson, 2008.

TOCCI, R. J.; WIDMER, N. S. **Sistemas digitais:** princípios e aplicações. 8. ed. São Paulo: Pearson, 2006.

## Marcas Registradas

---

Todos os nomes registrados, marcas registradas ou direitos de uso citados neste livro pertencem aos respectivos proprietários.



Av. das Nações Unidas, 7221, 1º Andar, Setor B  
Pinheiros – São Paulo – SP – CEP: 05425-902

**SAC** | 0800-0117875  
De 2ª a 6ª, das 8h00 às 18h00  
[www.editorasaraiva.com.br/contato](http://www.editorasaraiva.com.br/contato)

<b>Vice-presidente</b>	Claudio Lensing
<b>Gestora do ensino técnico</b>	Alini Dal Magro
<b>Coordenadora editorial</b>	Rosiane Ap. Marinho Botelho
<b>Editora de aquisições</b>	Rosana Ap. Alves dos Santos
<b>Assistente de aquisições</b>	Mônica Gonçalves Dias
<b>Editoras</b>	Márcia da Cruz Nóbrega Leme Silvia Campos Ferreira
<b>Assistentes editoriais</b>	Paula Hercy Cardoso Craveiro Raquel F. Abrantes
<b>Editor de arte</b>	Rodrigo Novaes de Almeida Kleber de Messias
<b>Assistentes de produção</b>	Fabio Augusto Ramos Katia Regina
<b>Produção gráfica</b>	Sergio Luiz P. Lopes

<b>Preparação de texto e Diagramação</b>	Triall Composição Editorial Ltda
<b>Capa</b>	Maurício S. de França
<b>Impressão e acabamento</b>	

## DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP) (CÂMARA BRASILEIRA DO LIVRO, SP, BRASIL)

Capuano, Francisco Gabriel

Sistemas digitais : circuitos combinacionais e sequenciais / Francisco Gabriel Capuano. -- 1. ed. -- São Paulo : Érica, 2014.

Bibliografia.

ISBN 978-85-365-2576-1

1. Eletrônica digital I. Título.

14-00543

CDD-621.381

Índices para catálogo sistemático:

1. Sistemas digitais : Engenharia eletrônica :  
Tecnologia 621.381

Copyright© 2014 Saraiva Educação

Todos os direitos reservados.

**1a edição**  
5a tiragem: 2017

Autores e Editora acreditam que todas as informações aqui apresentadas estão corretas e podem ser utilizadas para qualquer fim legal. Entretanto, não existe qualquer garantia, explícita ou implícita, de que o uso de tais informações conduzirá sempre ao resultado desejado. Os nomes de sites e empresas, porventura mencionados, foram utilizados apenas para ilustrar os exemplos, não tendo vínculo nenhum com o livro, não garantindo a sua existência nem divulgação.

A Ilustração de capa e algumas imagens de miolo foram retiradas de <[www.shutterstock.com](http://www.shutterstock.com)>, empresa com a qual se mantém contrato ativo na data de publicação do livro. Outras foram obtidas da Coleção MasterClips/MasterPhotos© da IMSI, 100 Rowland Way, 3rd floor Novato, CA 94945, USA, e do CorelDRAW X6 e X7, Corel Gallery e Corel Corporation Samples. Corel Corporation e seus licenciadores. Todos os direitos reservados.

Todos os esforços foram feitos para creditar devidamente os detentores dos direitos das imagens utilizadas neste livro. Eventuais omissões de crédito e copyright não são intencionais e serão devidamente solucionadas nas próximas edições, bastando que seus proprietários contatem os editores.

Nenhuma parte desta publicação poderá ser reproduzida por qualquer meio ou forma sem a prévia autorização da Saraiva Educação. A violação dos direitos autorais é crime

estabelecido na lei nº 9.610/98 e punido pelo artigo 184 do Código Penal.

---

CL 640485 CAE 585045

---

Edição digital: maio 2018

---

Arquivo ePub produzido pela **Simplíssimo Livros**

---