



Técnicas Digitais

Saul Azzolin Bonaldo



**Santa Maria - RS
2011**

Presidência da República Federativa do Brasil

Ministério da Educação

Secretaria de Educação a Distância

© Colégio Técnico Industrial de Santa Maria

Este material didático foi elaborado em parceria, entre o Colégio Técnico Industrial de Santa Maria e a Universidade Federal de Santa Catarina para o Sistema Escola Técnica Aberta do Brasil – e-Tec Brasil.

Comissão de Acompanhamento e Validação
Universidade Federal de Santa Catarina/UFSC

Coordenação Institucional
Araci Hack Catapan/UFSC

Coordenação do Projeto
Sílvia Modesto Nassar/UFSC

Coordenação de Design Instrucional
Beatriz Helena Dal Molin/UNIOESTE

Designers Intrucionais
Helena Maria Maullmann/UFSC
Jorge Luiz Silva Hermenegildo/CEFET-SC

WEB Designers
Beatriz Helena Dal Molin/UNIOESTE
Mércia Freire Rocha Cordeiro Machado/ETUFPR

Supervisão de Projeto Gráfico
Ana Carine García Montero/UFSC

Diagramação
João Ricardo Zattar/UFSC
Luís Henrique Lindler/UFSC

Revisão
Lúcia Locatelli Flôres/UFSC

Comissão de Acompanhamento e Validação
Colégio Técnico Industrial de Santa Maria/CTISM

Coordenador Institucional
Paulo Roberto Colusso/CTISM

Professor-autor
Saul Azzolin Bonaldo/CTISM

Coordenação Técnica
Iza Neuza Teixeira Bohrer/CTISM

Coordenação de Design
Erika Goellner/CTISM

Revisão Pedagógica
Andressa Rosemárie de Menezes Costa/CTISM
Francine Netto Martins Tadielo/CTISM
Marcia Migliore Freo/CTISM

Revisão Textual
Lourdes Maria Grotto de Moura/CTISM
Vera da Silva Oliveira/CTISM

Revisão Técnica
Alex Martins/CTISM
Eduardo Lehnhart Vargas/CTISM

Diagramação e Ilustração
Leandro Felipe Aguiar Freitas/CTISM
Marcel Jacques/CTISM
Rafael Cavalli Viapiana/CTISM
Ricardo Antunes Machado/CTISM

Ficha catalográfica elaborada por Maristela Eckhardt – CRB 10/737
Biblioteca Central – UFSM

B697c Bonaldo, Saul Azzolin
Curso técnico em automação industrial : técnicas digitais /
Saul Azzolin Bonaldo. – 3. ed. – Santa Maria : Universidade Federal
de Santa Maria, Colégio Técnico Industrial de Santa Maria, Curso
Técnico em Automação Industrial, Universidade Federal de Santa
Catarina, 2011.
86 p. : il. ; 21 cm.

1. Informática 2. Tecnologia digital 3. Circuitos 4. Álgebra de
Boole 5. Portas lógicas 6. Programa Escola Aberta do Brasil I.
Universidade Federal de Santa Maria. Curso Técnico em Automação
Industrial. II. Universidade Federal de Santa Catarina III. Título.

CDU 004

Apresentação e-Tec Brasil

Prezado estudante,

Bem-vindo ao e-Tec Brasil!

Você faz parte de uma rede nacional pública de ensino, a Escola Técnica Aberta do Brasil, instituída pelo Decreto nº 6.301, de 12 de dezembro 2007, com o objetivo de democratizar o acesso ao ensino técnico público, na modalidade a distância. O programa é resultado de uma parceria entre o Ministério da Educação, por meio das Secretarias de Educação a Distância (SEED) e de Educação Profissional e Tecnológica (SETEC), as universidades e escolas técnicas estaduais e federais.

A educação a distância no nosso país, de dimensões continentais e grande diversidade regional e cultural, longe de distanciar, aproxima as pessoas ao garantir acesso à educação de qualidade, e promover o fortalecimento da formação de jovens moradores de regiões distantes dos grandes centros geograficamente ou economicamente.

O e-Tec Brasil leva os cursos técnicos a locais distantes das instituições de ensino e para a periferia das grandes cidades, incentivando os jovens a concluir o ensino médio. Os cursos são ofertados pelas instituições públicas de ensino e o atendimento ao estudante é realizado em escolas-polo integrantes das redes públicas municipais e estaduais.

O Ministério da Educação, as instituições públicas de ensino técnico, seus servidores técnicos e professores acreditam que uma educação profissional qualificada – integradora do ensino médio e educação técnica, – é capaz de promover o cidadão com capacidades para produzir, mas também com autonomia diante das diferentes dimensões da realidade: cultural, social, familiar, esportiva, política e ética.

Nós acreditamos em você!

Desejamos sucesso na sua formação profissional!

Ministério da Educação
Janeiro de 2010

Nosso contato
etecbrasil@mec.gov.br



Indicação de ícones

Os ícones são elementos gráficos utilizados para ampliar as formas de linguagem e facilitar a organização e a leitura hipertextual.



Atenção: indica pontos de maior relevância no texto.



Saiba mais: oferece novas informações que enriquecem o assunto ou “curiosidades” e notícias recentes relacionadas ao tema estudado.



Glossário: indica a definição de um termo, palavra ou expressão utilizada no texto.



Mídias integradas: sempre que se desejar que os estudantes desenvolvam atividades empregando diferentes mídias: vídeos, filmes, jornais, ambiente AVEA e outras.



Atividades de aprendizagem: apresenta atividades em diferentes níveis de aprendizagem para que o estudante possa realizá-las e conferir o seu domínio do tema estudado.



Sumário

Palavra do professor-autor	9
Apresentação da disciplina	11
Projeto instrucional	13
Aula 1 – Portas lógicas	15
1.1 Histórico	15
1.2 Tipos de portas lógicas	16
Aula 2 – Álgebra de Boole	29
2.1 Definições preliminares	29
2.2 Propriedades ou leis da álgebra de Boole	30
2.3 Teoremas da álgebra de Boole	31
2.4 Quadro resumo	33
2.5 Expressões lógicas	33
Aula 3 – Mapas de <i>Karnaugh</i>	45
3.1 Métodos de minimização	45
3.2 Diagrama (ou mapa) de <i>Karnaugh</i> para duas variáveis	45
3.3 Diagrama (ou mapa) de <i>Karnaugh</i> para três variáveis	48
3.4 Diagrama (ou mapa) de <i>Karnaugh</i> para quatro variáveis	51
3.5 Diagramas com condições irrelevantes	55
Aula 4 – Circuitos combinacionais	59
4.1 Definição de circuitos combinacionais	59
4.2 Projetos de circuitos lógicos combinacionais	59
4.3 Projeto de circuitos codificadores e decodificadores	63
Aula 5 – Circuitos sequenciais	71
5.1 Definição de circuitos sequenciais	71
5.2 <i>Flip-flops</i> (ou biestáveis)	71
5.3 Diagramas temporais com <i>flip-flops</i>	79
Referências	84
Currículo do professor-autor	85



Palavra do professor-autor

"Em meu fim está meu princípio."

T. S. Eliot, "East Coker"

Câmeras digitais, aparelhos de áudio digital, telefones celulares com recursos avançados, lentes de contato interativas, dispositivos de identificação por radiofrequência, todos esses equipamentos estão deixando nosso mundo muito mais interessante.

Então, muitos sonhos aparentemente impossíveis estão se tornando realidade rapidamente, graças ao avanço da tecnologia digital. O universo digital está se espalhando por áreas inimagináveis, e o limite é dado pela imaginação.

O objetivo principal do estudo das Técnicas Digitais é conhecer os princípios e as técnicas que são comuns a todos os sistemas digitais, partindo da mais simples chave, liga-desliga, ao mais complexo computador. Tendo o domínio desta poderosa ferramenta, você também poderá realizar algum sonho aparentemente impossível.

Steve Jobs, co-fundador da *Apple Computers* e um dos principais responsáveis pela transformação do mundo em um grande universo digital, deixou um recado a uma turma de formandos da Universidade de Stanford: *Stay Hungry, Stay Foolish*. "Continue esfomeado, continue tolo". Nunca se satisfaça totalmente, nem desista por medo de fazer besteira ou de ver que seus sonhos são ilusórios.

Continue "esfomeado", continue "tolo", afinal, qualquer sonho vale a pena.

Saul Azzolin Bonaldo



Apresentação da disciplina

Na disciplina de Tecnologia da Informática, vimos que os circuitos internos dos computadores modernos trabalham no modo digital, utilizando o sistema binário de numeração.

Circuitos e sistemas digitais são encontrados não somente em computadores, mas em muitos outros equipamentos, como em videogames, dispositivos de eletrônica embarcada (automotivos), equipamentos de medição, controladores de temperatura, entre outros.

Os circuitos digitais estão também sendo utilizados em aplicações tipicamente analógicas, como aparelhos de áudio e vídeo, por exemplo.

Nesta disciplina, serão estudados os princípios e as técnicas comuns a todos os sistemas digitais, partindo da mais simples chave liga-desliga ao mais complexo computador. O entendimento destes princípios é fundamental para se compreender o funcionamento de sistemas digitais, comandos lógico-processados e microprocessadores, de modo que possamos aplicar esses conhecimentos na análise e reparo de muitos desses sistemas.



Projeto instrucional

Disciplina: Técnicas Digitais (carga horária: 60h).

Ementa: Portas lógicas. Álgebra de Boole. Mapas de *Karnaugh*. Circuitos Combinacionais. Circuitos Sequenciais.

AULA	OBJETIVOS DE APRENDIZAGEM	MATERIAIS	CARGA HORÁRIA (horas)
1. Portas lógicas	Entender o funcionamento dos diferentes tipos de portas lógicas. Conhecer a simbologia, o círculo equivalente e a tabela verdade de cada uma das portas lógicas estudadas.	Ambiente virtual. Apostila didática. Recursos de apoio: <i>links</i> , exercícios, textos complementares, videoconferência.	12
2. Álgebra de Boole	Compreender a álgebra de Boole, seus postulados, propriedades e teoremas. Trabalhar com circuitos lógicos, expressões lógicas e tabelas verdade. Identificar as equivalências entre os diferentes tipos de portas lógicas.	Ambiente virtual. Apostila didática. Recursos de apoio: <i>links</i> , exercícios, textos complementares, videoconferência.	12
3. Mapas de <i>Karnaugh</i>	Entender a importância da simplificação de circuitos lógicos. Aprender o método de simplificação através de mapas de <i>Karnaugh</i> para duas, três e quatro variáveis.	Ambiente virtual. Apostila didática. Recursos de apoio: <i>links</i> , exercícios, textos complementares, videoconferência.	12
4. Circuitos combinacionais	Conhecer e identificar os circuitos lógicos combinacionais. Projetar circuitos lógicos combinacionais com a finalidade de resolver problemas que envolvam variáveis lógicas de entrada e saída.	Ambiente virtual. Apostila didática. Recursos de apoio: <i>links</i> , exercícios, textos complementares, videoconferência.	12
5. Circuitos sequenciais	Construir e analisar o funcionamento de <i>flip-flops</i> com portas <i>NAND</i> e <i>NOR</i> . Conhecer os diferentes tipos de <i>flip-flops</i> .	Ambiente virtual. Apostila didática. Recursos de apoio: <i>links</i> , exercícios, textos complementares, videoconferência.	12

Aula 1 – Portas lógicas

Objetivos

Entender o funcionamento dos diferentes tipos de portas lógicas.

Conhecer a simbologia, o circuito equivalente e a tabela verdade de cada uma das portas lógicas estudadas.

1.1 Histórico

Em 1854, o matemático inglês George Boole apresentou um sistema matemático de análise lógica conhecido como **álgebra de Boole**.

Somente em 1938, um engenheiro americano (Claude Shannon) utilizou as teorias da álgebra de Boole para a solução de problemas de circuitos de telefonia com relés, tendo publicado um artigo que praticamente introduziu na área tecnológica o campo da eletrônica digital.

Os sistemas digitais são formados por circuitos lógicos denominados portas lógicas que, utilizados de forma conveniente, podem implementar todas as expressões geradas pela álgebra de Boole.

A aula 2 nos trará uma boa explicação sobre álgebra de Boole.

As portas lógicas são circuitos digitais com uma ou mais tensões de entrada que podem ser construídos com diodos, transistores e resistores conectados de tal forma, que o sinal de saída do circuito corresponde ao resultado de uma função lógica básica (*AND*, *OR*, *NOT*). Os valores possíveis das tensões de entrada e da tensão de saída são somente dois: tensão de alimentação do circuito (V_{cc}) ou tensão nula (terra ou GND). Por convenção, considera-se a tensão de alimentação como sinal lógico "1" e a tensão nula como sinal lógico "0".

Uma vez associados os sinais elétricos com níveis lógicos (“0” e “1”), pode-se concluir que através de portas lógicas se pode implementar qualquer um dos operadores lógicos (*AND*, *OR*, *NOT*).

Existem três portas básicas (*AND*, *OR* e *NOT*) que podem ser conectadas de várias maneiras, formando sistemas que vão de simples relógios digitais aos computadores de grande porte. Derivadas dessas portas lógicas, outras portas lógicas foram concebidas como a “Porta *NAND*”, a “Porta *NOR*”, a “Porta *XOR*” e a “Porta *XNOR*”.



Para melhor compreensão, para cada porta lógica básica serão apresentados um circuito elétrico equivalente e um circuito eletrônico capaz de implementar a função lógica equivalente. Será apresentada, também, a tabela verdade, a expressão lógica que define a função/porta lógica em questão e a simbologia tradicionalmente utilizada para representá-la. Os sinais de entrada das portas serão representados com as letras iniciais do alfabeto e o sinal de saída pela letra “S”. Convém enfatizar que os sinais de entrada e de saída são tensões elétricas.

1.2 Tipos de portas lógicas

1.2.1 Porta “*AND*” (E)

A função ***AND*** é aquela que executa a multiplicação de duas ou mais variáveis booleanas. Sua representação algébrica para duas variáveis é:

$$S = A \cdot B$$

Lê-se:
 $S = A \text{ e } B$

A porta lógica “*AND*” possui dois ou mais sinais de entrada, mas somente um sinal de saída. Nessa porta lógica, todas as entradas devem estar no nível lógico “1” (V_{cc}) para que se obtenha um nível lógico “1” (V_{cc}) na saída da mesma. Caso contrário, o sinal de saída será de nível lógico “0”.

a) Expressão lógica da porta *AND*

$$S = A \cdot B$$

b) Circuito elétrico equivalente da porta *AND*

Para compreender a função ***AND*** da álgebra booleana, deve-se analisar o circuito da Figura 1.1, para o qual se adotam as seguintes convenções:

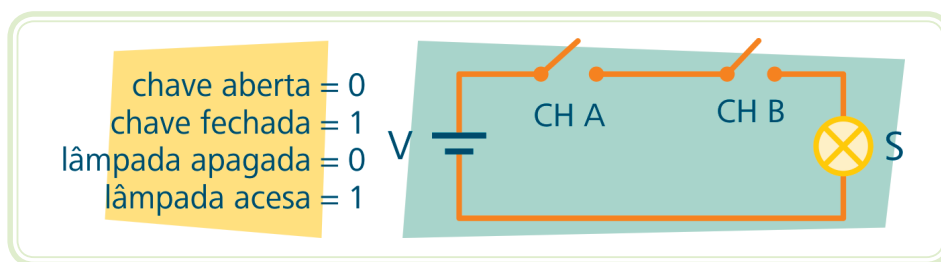


Figura 1.1: Circuito elétrico equivalente da porta AND

Fonte: CTISM

A análise da Figura 1.1 revela que a lâmpada somente acenderá se ambas as chaves estiverem fechadas e, seguindo a convenção, tem-se:

c) Tabela verdade da porta AND

Para o caso específico da porta lógica AND, a tabela verdade fica sendo:

CH A = 1, CH B = 1, resultante em S = 1

Tabela 1.1: Tabela verdade da porta AND

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

Na tabela verdade, escrevem-se todas as possíveis combinações de operação das chaves. Dessa forma, verifica-se que a tabela verdade é um mapa no qual são depositadas todas as possíveis situações com seus respectivos resultados. O número de combinações possíveis é igual a 2^N , onde N é o número de variáveis de entrada.

d) Simbologia da porta AND

A **porta lógica AND** é, portanto, um circuito eletrônico que executa a **função AND** da álgebra de Boole, sendo representada, na prática, pelo símbolo visto na Figura 1.2 que segue:

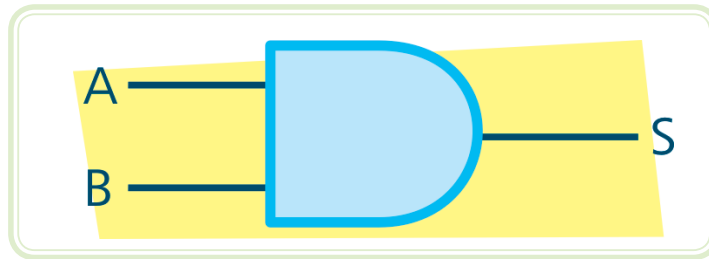


Figura 1.2: Simbologia da porta lógica AND

Fonte: CTISM

1.2.2 Porta "OR" (OU)

A função **OR** é aquela que assume valor 1 quando uma ou mais variáveis de entrada forem iguais a 1 e assume 0 se, somente se, todas as variáveis de entrada forem iguais a zero. Sua representação algébrica para duas variáveis de entrada é:

$$S = A + B \quad \text{Lê-se:} \\ S = A \text{ ou } B$$

Portanto, nessa porta lógica, pelo menos uma das entradas deve estar no nível lógico "1" (Vcc) para que se obtenha um nível lógico "1" (Vcc) na saída da porta lógica.

a) Expressão lógica da porta OR

$$S = A + B$$

b) Circuito elétrico equivalente da porta OR

Para entender melhor a função OR da álgebra booleana, analisam-se todas as situações possíveis de operação das chaves do circuito da Figura 1.3. A convenção é a mesma adotada anteriormente:

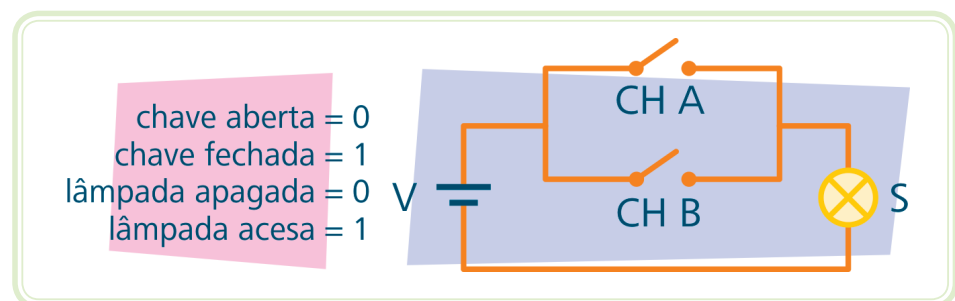


Figura 1.3: Circuito elétrico equivalente da porta OR

Fonte: CTISM

O circuito anterior mostra que a lâmpada acende quando qualquer uma das chaves estiver fechada e permanece apagada se ambas estiverem abertas, ou seja:

c) Tabela verdade da porta OR

Para o caso específico da porta lógica OR, a tabela verdade fica sendo:

Tabela 1.2: Tabela verdade da porta OR		
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

d) Simbologia da porta OR

A **porta lógica OR** é, portanto, um circuito eletrônico que executa a **função OR** da álgebra de Boole, sendo representada, na prática, através do símbolo visto na Figura 1.4 a seguir:

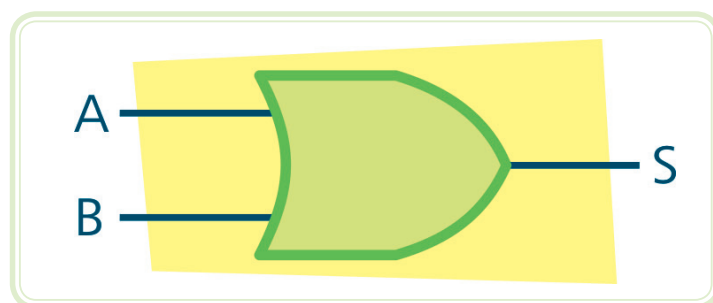


Figura 1.4: Simbologia da porta lógica OR

Fonte: CTISM

1.2.3 Porta inversora "NOT" (NÃO)

A função **NOT** é aquela que inverte ou complementa o estado da variável de entrada, ou seja, se a variável estiver em 0, a saída vai para 1; se estiver em 1, a saída vai para 0. É, portanto, uma porta com apenas um sinal de entrada e um sinal de saída o qual assumirá sempre valores lógicos inversos (complementares) ao sinal de entrada. Executa a função lógica da inversão booleana.

a) Expressão lógica da porta NOT

A porta lógica NOT é representada algebricamente da seguinte forma:

$$S = \overline{A}$$

Lê-se:
S é igual a A barra ou NÃO A

b) Circuito elétrico equivalente da porta NOT

A análise do circuito da Figura 1.5 ajuda a compreender melhor a função **NOT** da álgebra Booleana.

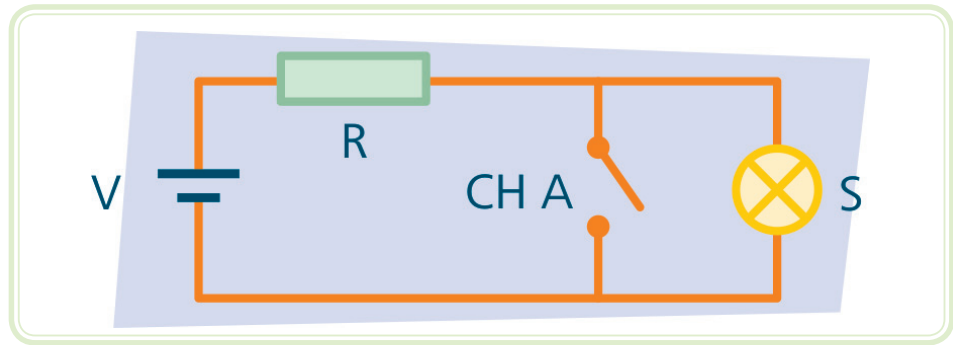


Figura 1.5: Circuito elétrico equivalente da porta NOT

Fonte: CTISM

Observando o circuito da Figura 1.5, pode-se concluir que a lâmpada estará acesa somente se a chave estiver aberta ($CH A = 0$, $S = 1$). Quando a chave fecha, a corrente desvia por ela, e a lâmpada se apaga ($CH A = 1$, $S = 0$).

c) Tabela verdade da porta NOT

Para o caso específico da porta lógica NOT, a tabela verdade fica sendo:

Tabela 1.3: Tabela verdade da porta NOT	
A	S
0	1
1	0

d) Simbologia da porta NOT

A porta lógica NOT é, portanto, um circuito eletrônico que executa a função NOT da álgebra de Boole, sendo representada, na prática, através do símbolo visto na Figura 1.6 que segue:

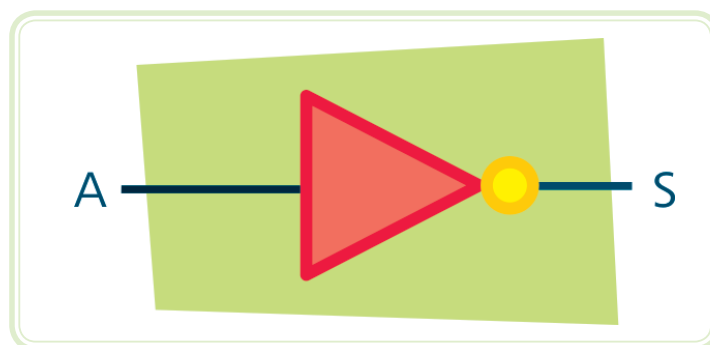


Figura 1.6: Simbologia da porta lógica NOT

Fonte: CTISM

1.2.4 Porta "NAND" (NÃO E)

Esta função é uma composição das funções **AND** e **NOT**, ou seja, é a função **AND** invertida. A porta lógica **NAND** tem dois ou mais sinais de entrada e

apenas um de saída que só será baixo se todos os sinais de entrada forem altos. Como o próprio nome diz, a porta *NAND* é uma porta lógica *AND* com saída negada, isto é, uma *AND* seguida de uma *NOT*.

a) Expressão lógica da porta lógica *NAND*

A expressão algébrica da porta *NAND* é dada por:

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

O traço indica uma inversão do produto booleano $A \cdot B$ que define a porta lógica *AND*.

b) Circuito elétrico equivalente da porta *NAND*

O circuito da Figura 1.7 esclarece o comportamento da função “*NAND*”.

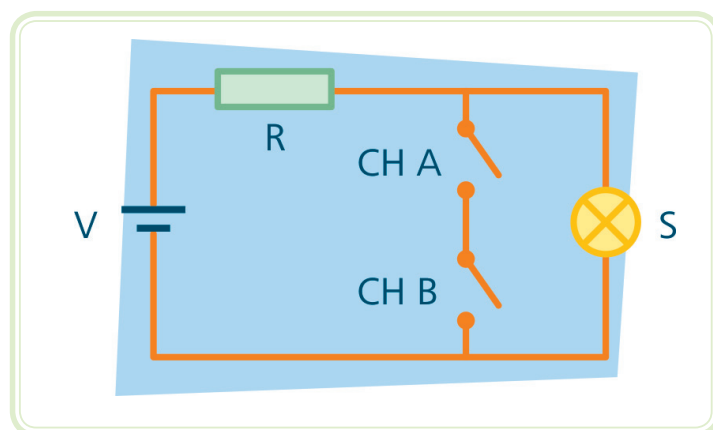


Figura 1.7: Circuito elétrico equivalente da porta *NAND*

Fonte: CTISM

Observa-se que a lâmpada apaga somente quando ambas as chaves são fechadas, ou seja:

$$CH A = 1, CH B = 1, \text{ implica em } S = 0$$

c) Tabela verdade da porta lógica *NAND*

Para o caso específico da porta lógica *NAND*, a tabela verdade fica sendo:

Tabela 1.4: Tabela verdade da porta lógica <i>NAND</i>		
A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

d) Simbologia da porta **NAND**

A porta lógica **NAND** é, portanto, um circuito eletrônico que executa a função **AND** da álgebra de Boole e a inverte, de acordo com a Figura 1.8 que segue:

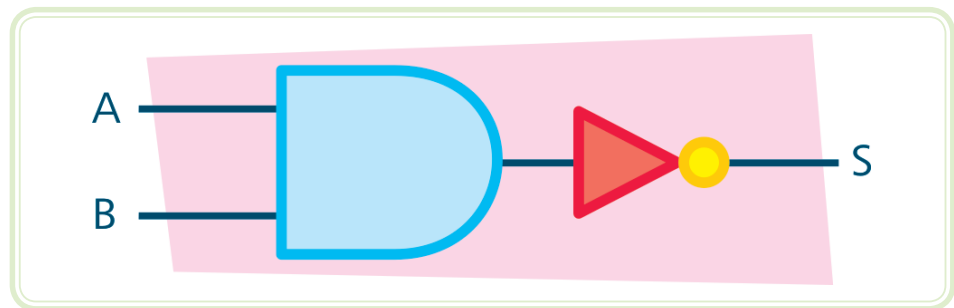


Figura 1.8: Simbologia da porta **NAND**

Fonte: CTISM

A porta lógica **NAND** é representada, na prática, pelo símbolo visto na Figura 1.9 a seguir:

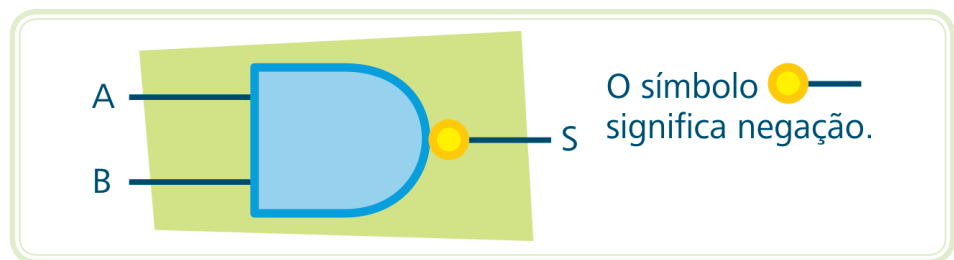


Figura 1.9: Simbologia usual da porta **NAND**

Fonte: CTISM

1.2.5 Porta “**NOR**” (NÃO OU)

Esta função é uma composição das funções **OR** e **NOT**, ou seja, é a função **OR** invertida. A porta lógica **NOR** tem dois ou mais sinais de entrada e apenas um de saída que só será alto se todos os sinais de entrada forem baixos. Como o próprio nome diz, a porta **NOR** é uma porta lógica **OR** com saída negada, isto é, uma **OR** seguida de uma **NOT**.

a) Expressão lógica da porta lógica **NOR**

A expressão algébrica da porta **NOR** é dada por:

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

O traço indica que há uma inversão da soma booleana $A + B$ que define a porta lógica **OR**

b) Circuito elétrico equivalente da porta **NOR**

O circuito da Figura 1.10 esclarece o comportamento da função **NOR**.

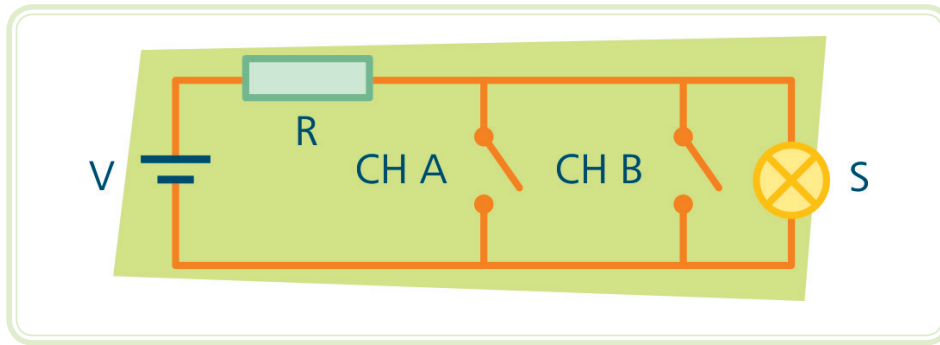


Figura 1.10: Circuito elétrico equivalente da porta *NOR*

Fonte: CTISM

Observa-se que a lâmpada fica acesa somente quando as duas chaves estão abertas.

CH A = 0, CH B = 0, resultante em S = 1

c) Tabela verdade da porta lógica *NOR*

Para o caso específico da porta lógica *NOR*, a tabela verdade fica sendo:

Tabela 1.5: Tabela verdade da porta lógica <i>NOR</i>		
A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

d) Simbologia da porta *NOR*

A porta lógica *NOR* é, portanto, um circuito eletrônico que executa a função *OR* da álgebra de Boole e a inverte, de acordo com a Figura 1.11 que segue:

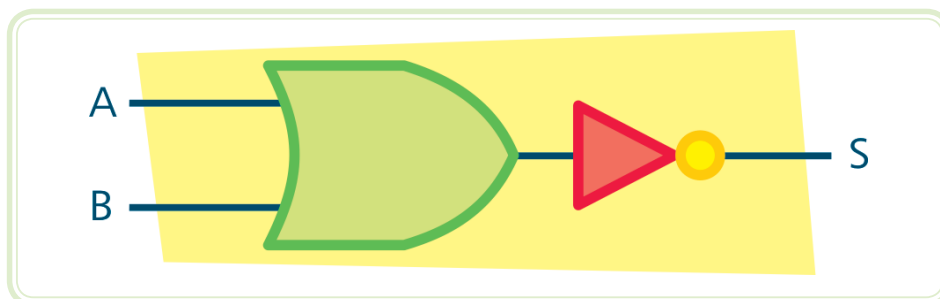


Figura 1.11: Simbologia da porta *NOR*

Fonte: CTISM

A porta lógica *NOR* é representada, na prática, pelo símbolo visto na Figura 1.12.

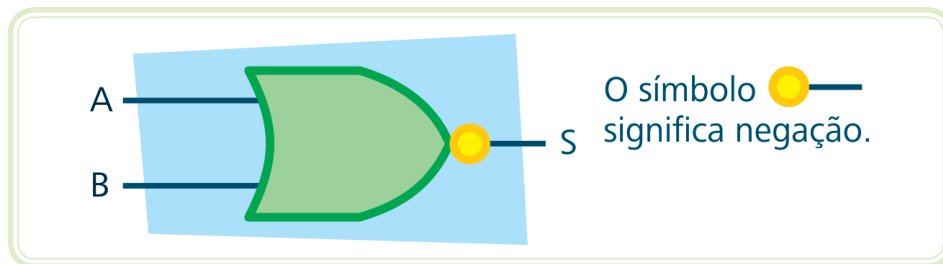


Figura 1.12: Simbologia usual da porta NOR

Fonte: CTISM

1.2.6 Porta "XOR" (OU EXCLUSIVO)

Essa função, como o próprio nome diz, apresenta saída com valor 1, quando as variáveis de entrada forem diferentes entre si. Portanto, a porta lógica **XOR** é um circuito lógico tal que, para cada combinação dos sinais de entrada, o sinal de saída será nível lógico "1" (alto) se, somente se, houver um NÚMERO ÍMPAR de entradas em nível lógico "1" (alto).

a) Expressão lógica da porta lógica XOR

A notação algébrica que representa a função XOR é dada por:

$$S = A \oplus B$$

Lê-se: A OU EXCLUSIVO B

O símbolo " $\oplus$ " é chamado de OPERADOR XOR

b) Circuito elétrico equivalente da porta XOR

O circuito da Figura 1.13 esclarece o comportamento da função XOR.

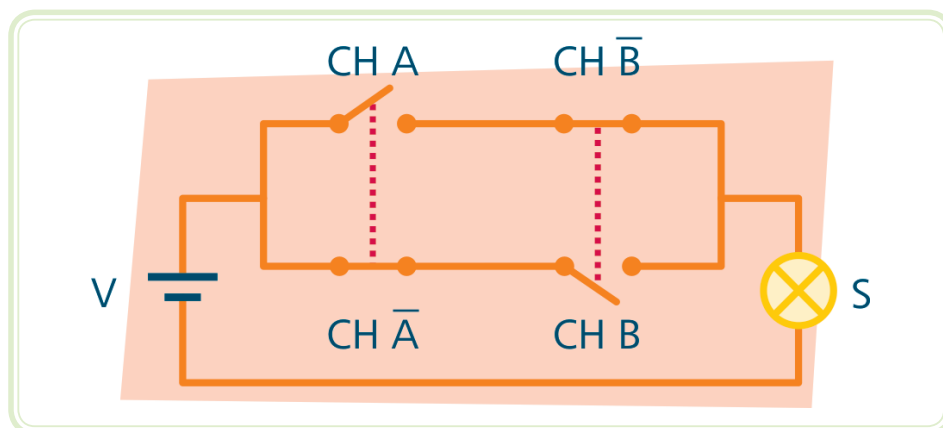


Figura 1.13: Circuito elétrico equivalente da porta XOR

Fonte: CTISM

Observa-se que a lâmpada fica acesa, ou seja, este bloco somente terá nível lógico 1 na saída, quando suas entradas forem diferentes.

CH A = 0, CH B = 1, resultante em S = 1
 CH A = 1, CH B = 0, resultante em S = 1
 Para os demais casos, a saída será 0

c) Tabela verdade da porta lógica XOR

Para o caso específico da porta lógica XOR, a tabela verdade fica sendo:

Tabela 1.6: Tabela verdade da porta lógica XOR		
A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

d) Simbologia da porta XOR

A porta lógica XOR é um circuito relativamente complexo formado por diversas portas lógicas básicas que e pode ser verificado na Figura 1.14 que segue:

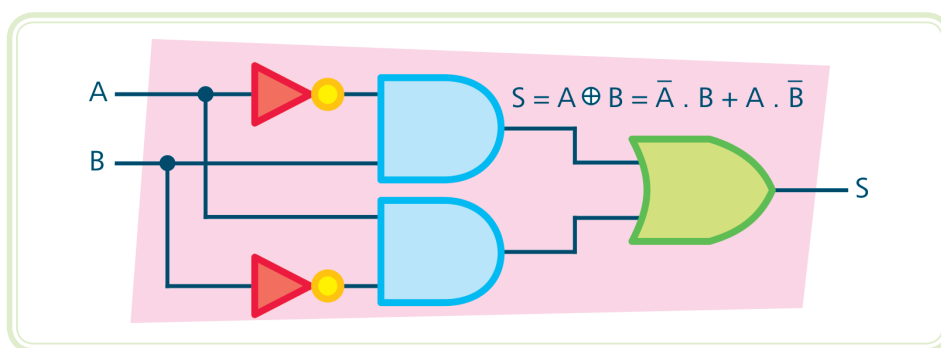


Figura 1.14: Diagrama da porta lógica XOR

Fonte: CTISM

A porta lógica XOR é representada, na prática, pelo símbolo visto na Figura 1.15 que segue:

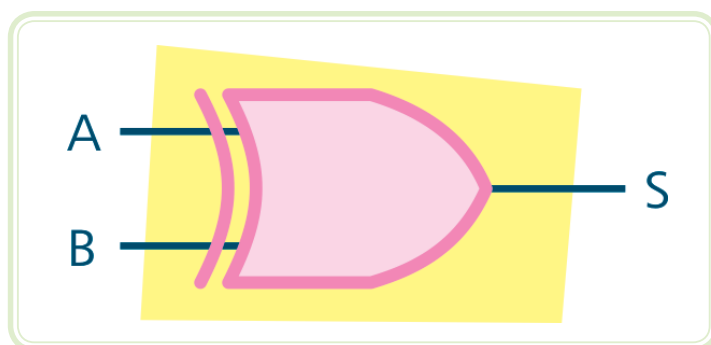


Figura 1.15: Simbologia da porta lógica XOR

Fonte: CTISM

1.2.7 Porta "XNOR" (NÃO OU EXCLUSIVA ou COINCIDÊNCIA)

Esta função, como o próprio nome diz, apresenta saída com valor "1", quando as variáveis de entrada forem iguais entre si. Portanto, a porta lógica XNOR é um circuito lógico tal, que para cada combinação dos sinais de entrada, o sinal de saída será nível lógico "1" (alto) se, somente se, houver um NÚMERO PAR de entradas em nível lógico "1" (alto). A porta lógica XNOR é a porta XOR invertida.

a) Expressão lógica da porta lógica XNOR

A notação algébrica que representa a função XNOR é dada por:

$$S = A \odot B$$

Lê-se: A COINCIDÊNCIA B. O símbolo " $\odot$ " é chamado de OPERADOR COINCIDÊNCIA ou OPERADOR XNOR

b) Circuito elétrico equivalente da porta XNOR

O circuito da Figura 1.16 esclarece o comportamento da função XNOR.

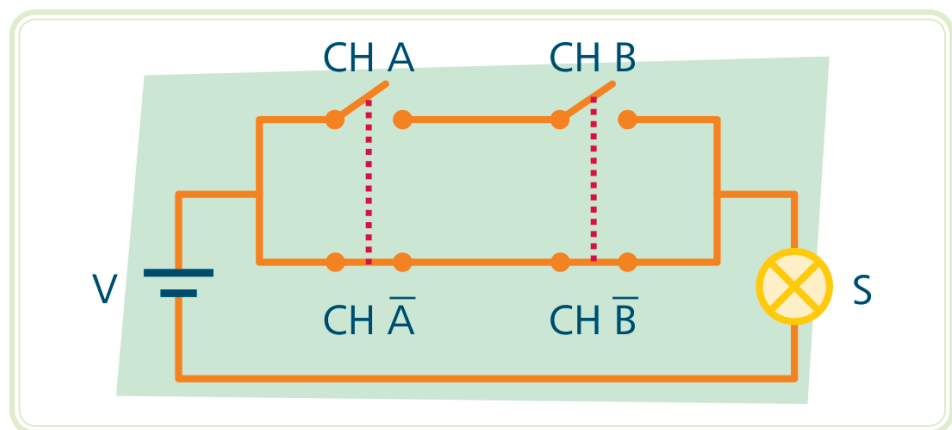


Figura 1.16: Circuito elétrico equivalente da porta XNOR

Fonte: CTISM

Observa-se que a lâmpada fica acesa, ou seja, este bloco somente terá nível lógico 1 na saída (lâmpada acesa), quando suas entradas forem iguais (coincidentes).

c) Tabela verdade da porta lógica XNOR

Para o caso específico da porta lógica XNOR, a tabela verdade fica sendo:

Tabela 1.7: Tabela verdade da porta lógica <i>XNOR</i>		
A	B	S
0	0	1
0	1	0
1	0	0
1	1	1

d) Simbologia da porta *XNOR*

A porta lógica *XNOR* é um circuito relativamente complexo, formado por diversas portas lógicas básicas que pode ser observado na Figura 1.17 a seguir:

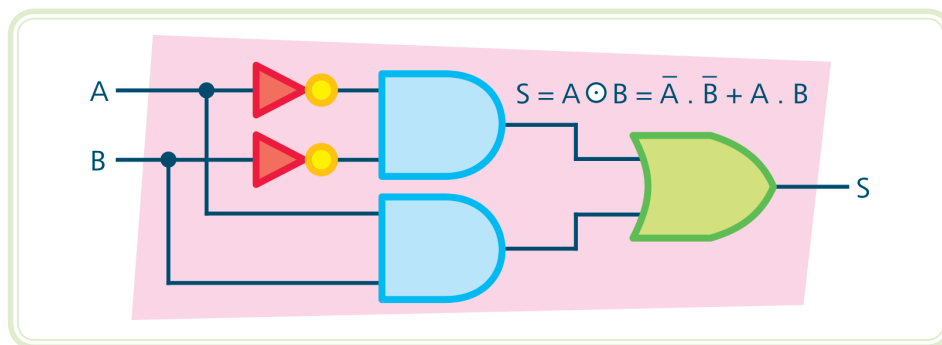


Figura 1.17: Diagrama da porta lógica *XNOR*

Fonte: CTISM

A porta lógica *XNOR* é representada, na prática, pelo símbolo visto na Figura 1.18 a seguir:

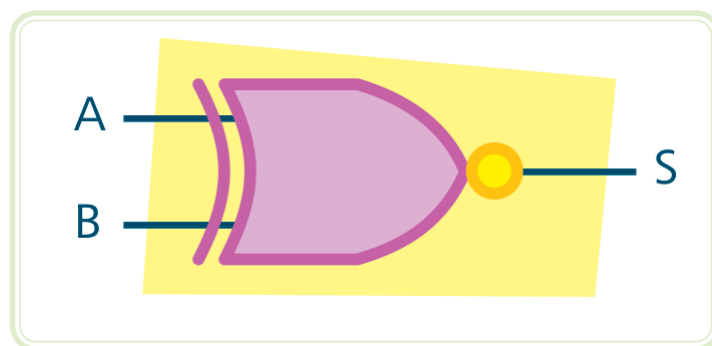


Figura 1.18: Simbologia da porta lógica *XNOR*

Fonte: CTISM

Resumo

Nesta aula, foram estudados os diferentes tipos de portas lógicas, suas características, simbologia, tabela verdade e expressão algébrica equivalente.



Atividades de aprendizagem

1. Elabore um quadro-resumo das portas lógicas, especificando o tipo a simbologia usual, a tabela verdade, a descrição da função lógica e expressão algébrica equivalente.

Aula 2 – Álgebra de Boole

Objetivos

Compreender a álgebra de Boole, seus postulados, propriedades e teoremas.

Trabalhar com circuitos lógicos, expressões lógicas e tabelas verdade.

Identificar as equivalências entre os diferentes tipos de portas lógicas.

2.1 Definições preliminares

Na aula anterior, os circuitos lógicos foram tratados sem preocupação com a simplificação o que, na prática, deve ser evitada visando minimizar a quantidade de portas lógicas do circuito.

Dessa forma, deve-se realizar um breve estudo da álgebra de Boole, pois é através de seus postulados, propriedades, teoremas fundamentais e identidades que se efetuam as simplificações. Na álgebra de Boole, estão todos os fundamentos da Eletrônica Digital.

Durante séculos, os matemáticos percebiam que havia uma conexão entre a matemática e a lógica, mas ninguém antes do matemático inglês George Boole descobriu essa ligação. Em 1854, Boole estabeleceu a teoria da lógica simbólica, onde cada variável lógica pode assumir somente valores discretos. Até meados de 1938, a álgebra booleana praticamente não teve aplicação no mundo real. Nessa época, Claude E. Shannon, pesquisador do *Bell Labs* (USA), demonstrou como adaptar a álgebra booleana para analisar e descrever o desempenho de circuitos de comutação telefônica construídos a base de relés. Ele fez com que as variáveis booleanas representassem relés FECHADOS ou ABERTOS. A partir desta aplicação, começou, então, a difusão da tecnologia digital que temos disponível na atualidade.

A grande diferenciação que há entre a álgebra booleana e a álgebra linear é que as constantes e variáveis booleanas podem assumir somente dois valores, 0

ou 1. Esses valores podem representar duas condições distintas, normalmente indicando “**Verdadeiro**” ou “**Falso**”. Contudo, também podem representar condições ambíguas, tais como “Aberto” ou “Fechado”, “Alto” ou “Baixo”, entre outras.

Em geral, utilizam-se o valor **0** para indicar a condição **falsa** e **1** para indicar a condição **verdadeira**. Essa lógica é conhecida por lógica normal ou convencional. Entretanto, em muitos casos utilizam-se **0** para **verdadeiro** e **1** para **falso**. Nesses casos, diz-se que se utiliza a **lógica inversa**.

Em função dos valores que as variáveis booleanas podem assumir, três operações básicas são possíveis de ser executadas:

SOMA BOOLEANA: operador *OR* (OU) “+”
PRODUTO BOOLEANO: operador *AND* (E) “.”
INVERSÃO BOOLEANA: operador *NOT* (NÃO) “—”

Como em qualquer teoria matemática, a álgebra de Boole possui POSTULADOS, LEIS E TEOREMAS que a definem. O conhecimento dessas definições é necessário para o correto entendimento dos princípios da Eletrônica Digital, o que nos permitirá desenvolver nossos próprios projetos de sistemas digitais.

2.2 Propriedades ou leis da álgebra de Boole

Serão estudadas as principais propriedades algébricas úteis principalmente no manuseio e nas simplificações de expressões e, conseqüentemente, de circuitos lógicos.

2.2.1 Propriedade comutativa

A propriedade comutativa é válida tanto na adição booleana quanto na multiplicação booleana, e é definida por:

$$A + B = B + A \qquad A \cdot B = B \cdot A$$

2.2.2 Propriedade associativa

Da mesma forma que a propriedade comutativa, a propriedade associativa é válida tanto na adição booleana quanto na multiplicação booleana. Dessa forma, temos:

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

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

2.2.3 Propriedade distributiva

Da mesma forma que na álgebra linear, a propriedade distributiva nos mostra que:

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

2.3 Teoremas da álgebra de Boole

Os teoremas da álgebra de Boole são definidos para uma variável booleana qualquer, ou seja, seu valor pode ser "0" ou "1". Esses teoremas são divididos em três grupos de acordo com as funções lógicas básicas.

2.3.1 Teoremas da adição lógica

$$A + 0 = A$$

$$A + 1 = 1$$

$$A + \overline{A} = 1$$

$$A + A = A$$

2.3.2 Teoremas do produto lógico

$$A \cdot 0 = 0$$

$$A \cdot 1 = A$$

$$A \cdot \overline{A} = 0$$

$$A \cdot A = A$$

2.3.3 Teorema do complemento ou da inversão lógica

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

2.3.4 Teoremas de "De Morgan"

Os teoremas de *De Morgan* são muito importantes quando se necessita simplificar um circuito lógico, ou mesmo eliminar o complemento de uma função

lógica. O primeiro teorema converte uma operação “OR” em uma operação “AND”. O segundo teorema realiza a operação inversa, isto é, converte uma operação “AND” em uma operação “OR”.

a) Primeiro teorema de De Morgan

O complemento de uma função lógica na forma de um produto lógico de qualquer número de variáveis pode ser transformado em uma soma lógica, complementando cada variável em separado e trocando o operador “.” pelo operador “+”.

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

De maneira geral, ou seja, estendendo para mais variáveis de entrada, temos:

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

b) Segundo teorema de De Morgan

O complemento de uma função lógica na forma de uma soma lógica de qualquer número de variáveis pode ser transformado em um produto lógico, complementando-se cada variável em separado e trocando o operador “+” por “.”.

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

De maneira geral, ou seja, estendendo para mais variáveis de entrada, temos:

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

2.3.5 Identidades auxiliares

Algumas identidades auxiliares, úteis para a simplificação de expressões, são descritas a seguir:

$$A + A \cdot B = A$$

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

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

2.4 Quadro resumo

Identidades		
Complementação $A = 0 \rightarrow \bar{A} = 1$ $\bar{A} = 0 \rightarrow A = 1$ $\overline{\bar{A}} = A$	Adição $A + 0 = 0$ $A + 1 = 1$ $A + A = A$ $A + \bar{A} = 1$	Multiplicação $A \cdot 0 = 0$ $A \cdot 1 = A$ $A \cdot A = A$ $A \cdot \bar{A} = 0$
Propriedades		
Comutativa	$A + B = B + A$	$A \cdot B = B \cdot A$
Associativa	$A + (B + C) = (A + B) + C = A + B + C$ $A \cdot (B \cdot C) = (A \cdot B) \cdot C = A \cdot B \cdot C$	
Distributiva	$A \cdot (B + C) = A \cdot B + A \cdot C$	
Teoremas “De Morgan”		
$\overline{(A \cdot B)} = \bar{A} + \bar{B}$		$\overline{(A + B)} = \bar{A} \cdot \bar{B}$
Identidades Auxiliares		
$A + A \cdot B = A$		$A + \bar{A} \cdot B = A + B$
$(A + B) \cdot (A + C) = A + B \cdot C$		

Figura 2.1: Quadro resumo das propriedades e teoremas da álgebra de Boole
 Fonte: CTISM

2.5 Expressões lógicas

Uma expressão lógica ou booleana é uma função formada por variáveis binárias, ou seja, variáveis que podem assumir somente os valores 0 e 1 por operadores lógicos *OR*, *AND* e *NOT*, por coeficientes numéricos de valor 0 ou 1 e por um sinal de igualdade. Uma expressão booleana pode ter como resultado somente os valores 0 e 1.

Da mesma forma que na álgebra tradicional, nas expressões booleanas podem ser utilizados parênteses, colchetes e chaves para se exprimir em prioridades.

Se o circuito lógico apresentar mais de uma saída, para cada uma delas haverá uma expressão booleana correspondente, ou seja, deve-se criar uma função lógica exclusiva para cada saída em função das entradas.

2.5.1 Expressões booleanas obtidas de circuitos lógicos

Todo o circuito lógico executa uma função booleana e, por mais complexo que seja, é formado pela interligação das portas lógicas básicas. Assim, pode-se obter a expressão booleana que é executada por um circuito lógico qualquer. Dessa forma, analisemos o circuito que segue:

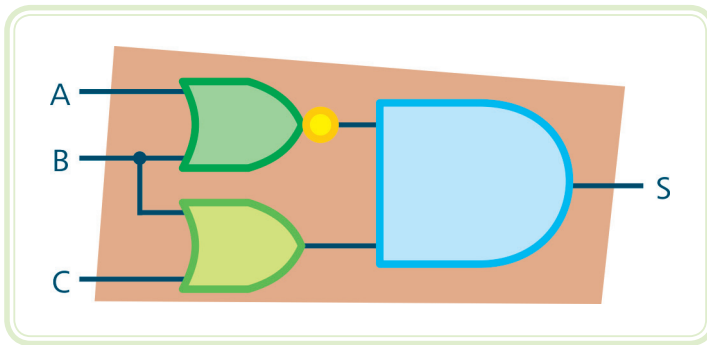


Figura 2.2: Exemplo de circuito lógico

Fonte: CTISM

A maneira mais simples de se descrever a expressão lógica do circuito anterior é escrever na saída das diversas portas lógicas do circuito a expressão lógica por elas executada. Dessa forma, temos:

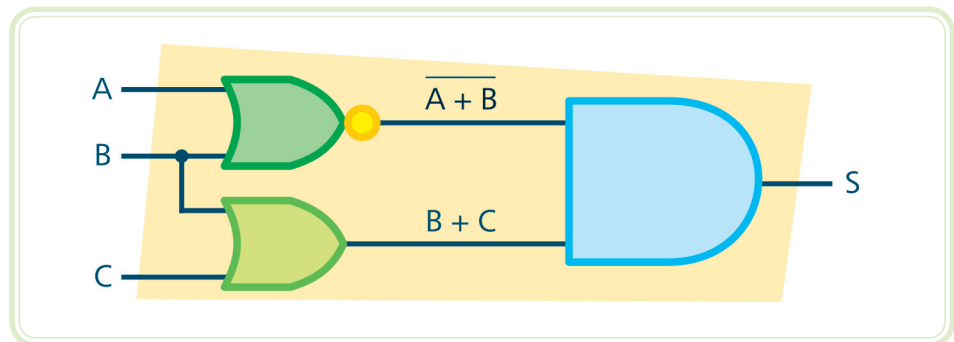


Figura 2.3: Exemplo anterior com as expressões de saída de cada porta lógica

Fonte: CTISM

E, portanto, o resultado final fica sendo a seguinte expressão:

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

2.5.2. Circuitos lógicos obtidos de expressões booleanas

Consiste em desenhar um circuito lógico que executa uma função booleana qualquer, ou seja, pode-se desenhar um circuito a partir de sua expressão característica.

Como método de resolução, deve-se identificar as portas lógicas na expressão e desenhá-las com as respectivas ligações, a partir das variáveis de entrada. Da mesma forma que na aritmética elementar, deve-se sempre respeitar a hierarquia das funções, ou seja, a solução inicia-se pelos parênteses, quando houver.

Vejamos, por exemplo, para a expressão booleana a seguir:

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

No primeiro parêntese, tem-se uma soma booleana $A + B$ negada. Portanto, a porta lógica que a representa é uma *NOR*.

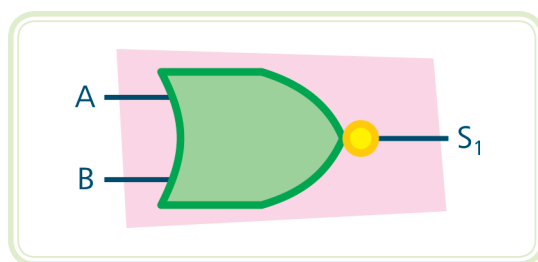


Figura 2.4: Porta lógica NOR

Fonte: CTISM

Para o segundo parêntese, temos uma soma booleana $B + C$. Portanto, a porta lógica que a representa é uma *OR*.

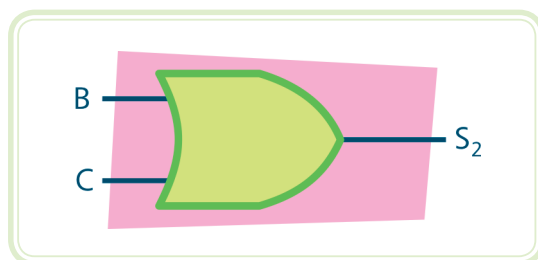


Figura 2.5: Porta lógica OR

Fonte: CTISM

Por fim, temos a multiplicação booleana dos dois parênteses, o que significa que se tem uma porta lógica *AND*, cujas entradas são as saídas S_1 e S_2 dos parênteses.

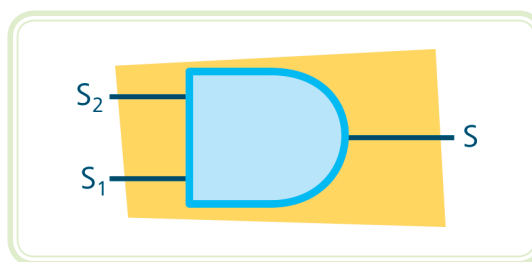


Figura 2.6: Multiplicação booleana das saídas S_1 e S_2

Fonte: CTISM

Dessa forma, o circuito final fica sendo o da Figura 2.7 que segue:

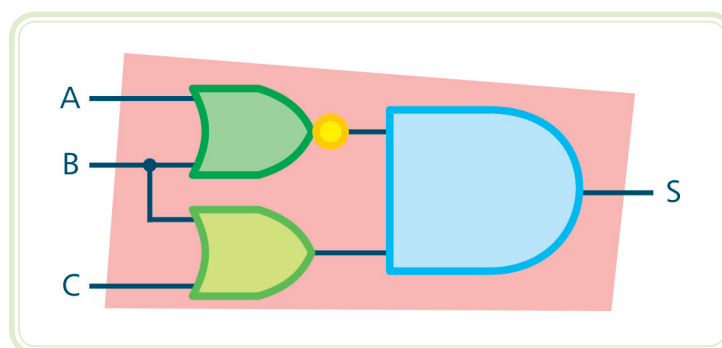


Figura 2.7: Circuito final que representa a expressão dada

Fonte: CTISM

2.5.3 Tabelas da verdade obtidas de expressões booleanas

A representação das combinações na forma de uma tabela a qual chamamos de **Tabela Verdade**, permite uma visão completa do comportamento da função. A tabela verdade, como já foi visto, é um mapa ou representação tabular em que se colocam todas as situações possíveis de uma dada expressão, juntamente com os valores por ela assumidos.

Um método prático para extrair a tabela verdade de uma expressão booleana pode ser o que segue:

- Montar o quadro de possibilidades: costuma-se arranjar as combinações dos sinais de entrada em linhas na ordem crescente de sua representação binária.
- Montar colunas para os vários membros da expressão lógica e preenchê-las com os respectivos resultados.
- Montar uma coluna para o resultado final e preenchê-la com os respectivos resultados.

O número de combinações possíveis entre as variáveis de entrada é dado por:

$$N = 2^n$$

sendo N o número de combinações possíveis e n o número de variáveis lógicas de entrada

Assim, para um sistema com três variáveis de entrada, teremos 23, ou seja, 8 combinações possíveis. Por exemplo, dada a seguinte expressão lógica:

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

Vemos que há três variáveis de entrada e, portanto, a tabela verdade terá 8 combinações possíveis, ou seja, 8 linhas. Poderemos ainda agregar duas colunas auxiliares, uma para cada membro da expressão e teremos, obrigatoriamente, uma coluna para o resultado final.

Tabela 2.1: Tabela referente ao exemplo dado

A	B	C	S_1	S_2	S
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	0	1	0

2.5.4 Expressões booleanas obtidas de tabelas verdade

Nesta seção, será estudada a forma de se obter em expressões booleanas a partir de tabelas verdade.

Esse método é muito simples e para se obter a expressão lógica a partir de uma tabela verdade, basta montar os termos relativos aos casos em que a expressão for verdadeira, ou seja, tiver saída igual a 1, e somá-los.

Por exemplo, analisando a tabela verdade que segue:

Tabela 2.2: Tabela verdade exemplo

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

Temos S = 1 somente quando:

$$A = 0, B = 1 \text{ e } C = 0 \rightarrow \bar{A} \cdot B \cdot \bar{C}$$

$$A = 1, B = 1 \text{ e } C = 1 \rightarrow A \cdot B \cdot C$$

Portanto, a expressão lógica resultante será obtida pela simples soma booleana de cada termo descrito, ou seja:

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

Nota-se que o método permite obter de qualquer tabela uma expressão padrão formada sempre pela **soma de produtos**.

2.5.5 Equivalência entre blocos lógicos

Nesta seção, veremos como se podem obter portas lógicas equivalentes, utilizando outros tipos de portas lógicas, ou seja, como realizar as mesmas funções lógicas de uma determinada porta lógica, utilizando somente outros tipos de portas lógicas.

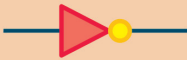
Bloco Lógico	Bloco Equivalente
	   
 $S = \overline{A+B}$	 $S = \overline{A} \cdot \overline{B}$
 $S = A+B$	 $S = \overline{\overline{A} \cdot \overline{B}}$
 $S = \overline{A \cdot B}$	 $S = \overline{A} + \overline{B}$
 $S = A \cdot B$	 $S = \overline{\overline{A} + \overline{B}}$

Figura 2.8: Equivalência entre portas lógicas

Fonte: CTISM

Essas equivalências são muito importantes na prática, ou seja, na montagem de sistemas digitais, pois possibilitam maior otimização na utilização dos circuitos integrados comerciais, assegurando, principalmente, a redução de componentes e a consequente minimização do custo do sistema.



2.5.6 Universalidade das portas lógicas **NAND** e **NOR**

Sabemos que toda expressão lógica é composta de diversas combinações dos operadores lógicos *AND*, *OR* e *NOT*. Veremos, nesta seção, que esses operadores lógicos podem ser implementados utilizando-se unicamente portas lógicas *NAND* ou *NOR*. Por esse fato, essas portas lógicas *NAND* e *NOR* são conhecidas como portas lógicas universais.

Muitas vezes, quando implementamos uma função lógica formada pela combinação de diversas portas lógicas, podemos não estar utilizando todas as portas lógicas que compõem os Circuitos Integrados (CI's) empregados na implementação. Caso todas as portas lógicas utilizadas fossem substituídas por portas *NAND* ou *NOR*, pode haver necessidade de um menor número de circuitos integrados para realizar a mesma função lógica.

A seguir, mostra-se como implementar as funções lógicas *AND*, *OR*, *NOT* e *NOR* a partir de portas lógicas *NAND*.

As portas lógicas estão disponíveis em circuitos integrados – CI's, que possuem em seu interior algumas portas lógicas de uma mesma espécie. Por exemplo, os CI's da família TTL, 7408 e 7432 são compostos por 4 portas lógicas *AND* de 2 entradas (7408) e 4 portas lógicas *OR* de 2 entradas (7432), respectivamente. Já o CI 7400 é composto de 4 portas *NAND* de 2 entradas.

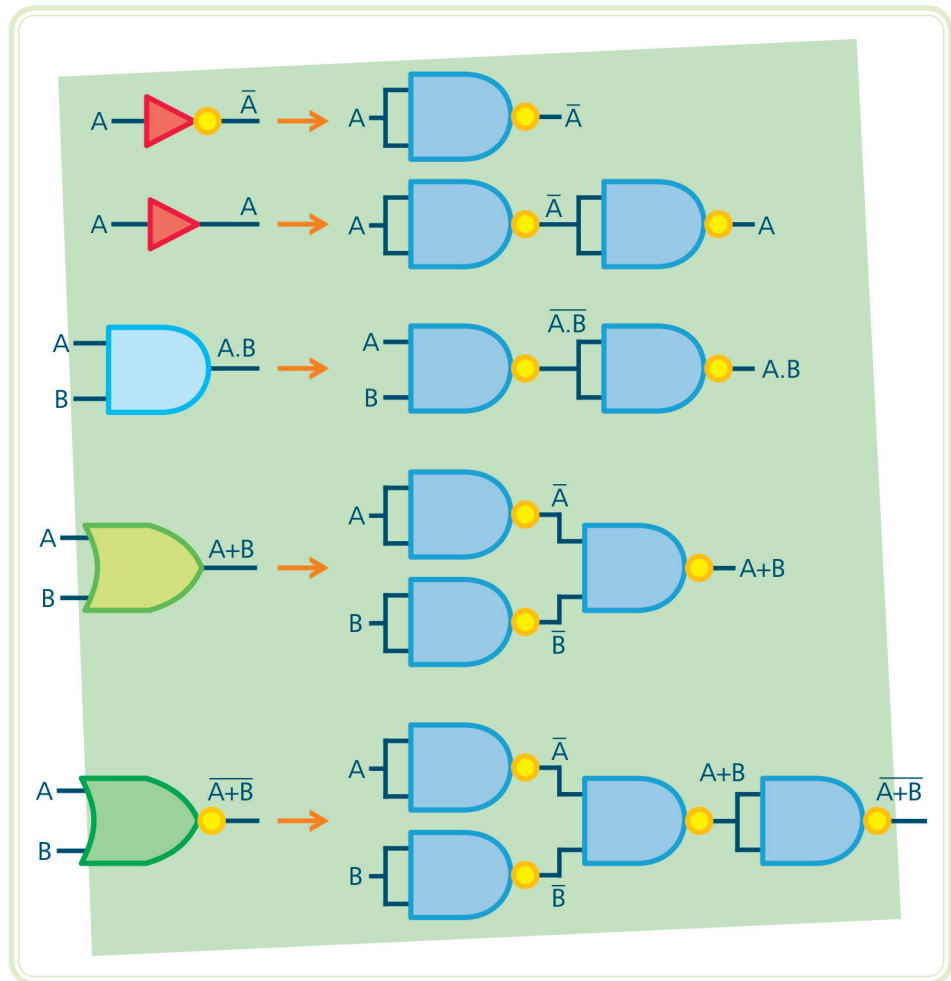


Figura 2.9: Universalidade da porta lógica "NAND"

Fonte: CTISM

Da mesma forma, a Figura 2.10 demonstra como se devem implementar as funções lógicas *AND*, *OR*, *NAND* e *NAND* a partir de portas lógicas *NOR*.

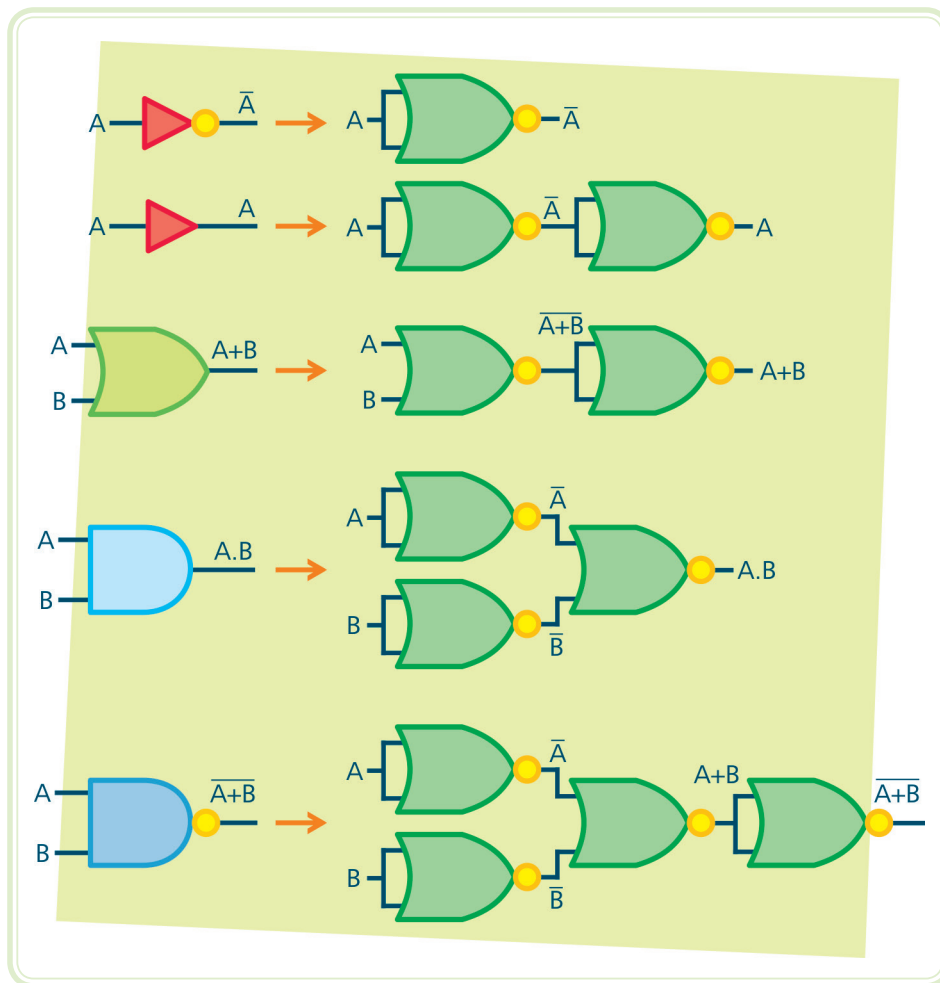


Figura 2.10: Universalidade da porta lógica "NOR"

Fonte: CTISM

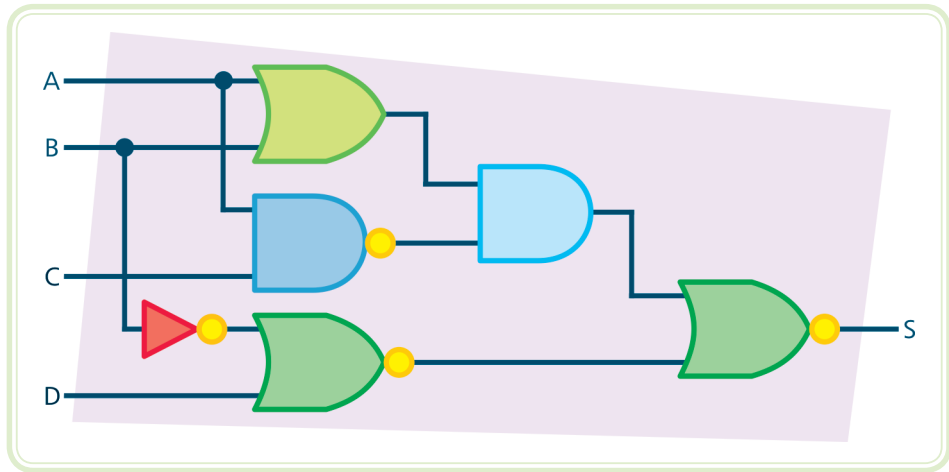
Resumo

Nessa aula, trabalhamos com a álgebra de Boole, seus postulados, propriedades e teoremas. Também trabalhamos com circuitos lógicos, expressões lógicas e tabelas verdade de circuitos lógicos. Finalmente, identificamos as equivalências entre os diferentes tipos de portas lógicas, bem como a universalidade das portas *NAND* e *NOR*. Dessa forma, estamos aptos a prosseguir os estudos, passando à próxima aula que tratará da simplificação dos circuitos lógicos.

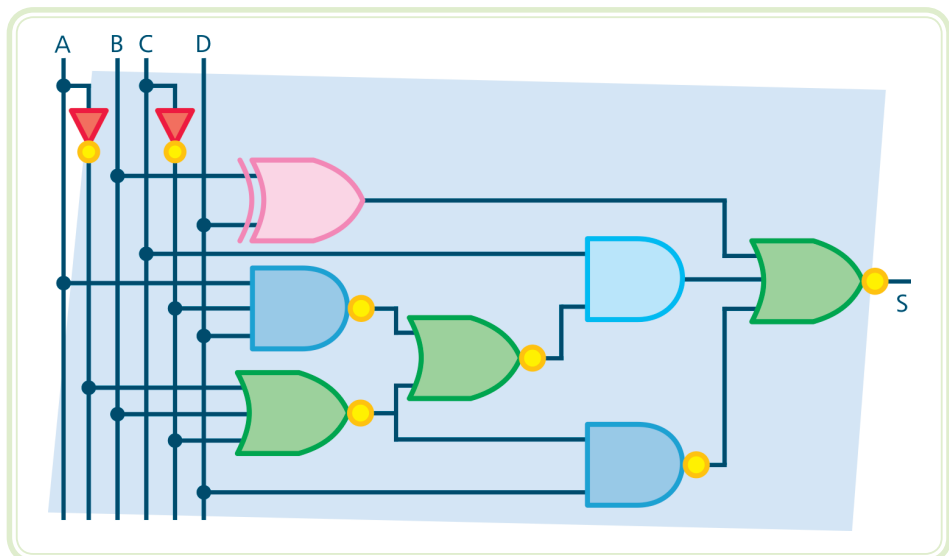


Atividades de aprendizagem

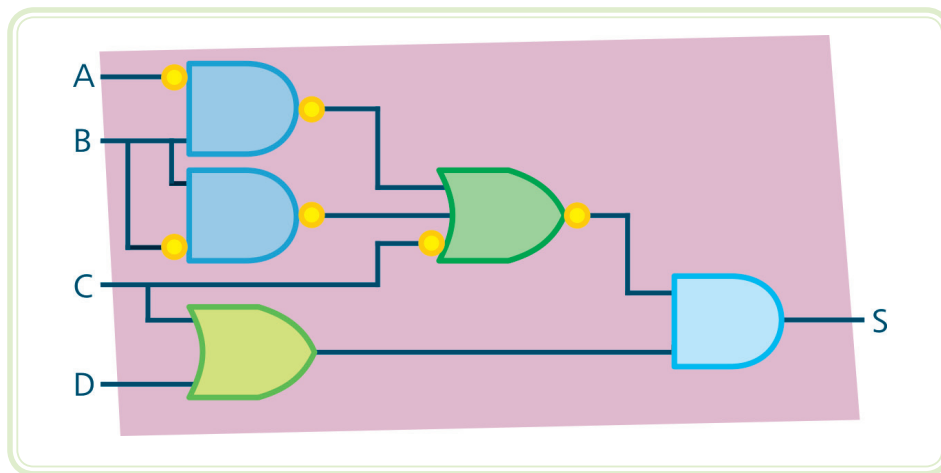
1. Determine as expressões dos circuitos que seguem e levante suas respectivas tabelas verdade.



Fonte: CTISM



Fonte: CTISM



Fonte: CTISM

2. Determine as expressões booleanas a partir das tabelas verdade que seguem.

Tabela verdade: Exercício 1

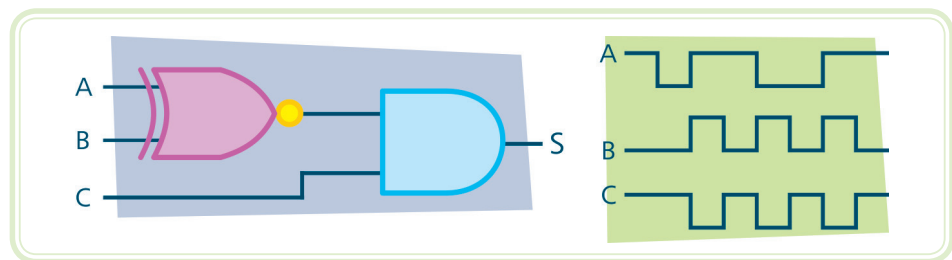
A	B	C	S
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Tabela verdade: Exercício 2

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

A	B	C	D	S
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

3. Desenhe o sinal de saída do circuito que segue.



Fonte: CTISM

4. Desenhe os circuitos que executam as expressões obtidas no exercício "2", utilizando:

- a) Somente portas *NAND*.
- b) Somente portas *NOR*.

Aula 3 – Mapas de *Karnaugh*

Objetivos

Entender a importância da simplificação de circuitos lógicos.

Aprender o método de simplificação através de mapas de *Karnaugh* para duas, três e quatro variáveis.

3.1 Métodos de minimização

Quando são utilizados os teoremas e postulados booleanos para a simplificação de expressões lógicas, não se pode afirmar, em vários casos, que a equação resultante está na sua forma minimizada.

A expressão lógica resultante pode ser escrita de diversas formas. A utilização da simplificação algébrica para a minimização de funções lógicas não segue regras claras e sequenciais para a correta manipulação algébrica, fazendo desta técnica um procedimento ineficaz e fortemente dependente da experiência do projetista.

Existem métodos de mapeamento das expressões lógicas que possibilitam a simplificação de expressões de N variáveis. O diagrama ou mapa de *Karnaugh* é um desses métodos que permite a simplificação mais rápida dos casos extraídos diretamente de tabelas verdade, obtidas de situações quaisquer.

O método de *Karnaugh* consiste em uma representação gráfica que permite a percepção visual dos termos fundamentais que compõem a função lógica, de modo a combiná-los para formar a função lógica simplificada.

3.2 Diagrama (ou mapa) de *Karnaugh* para duas variáveis

A Figura 3.1 mostra um diagrama de *Veitch-Karnaugh* para 2 variáveis.

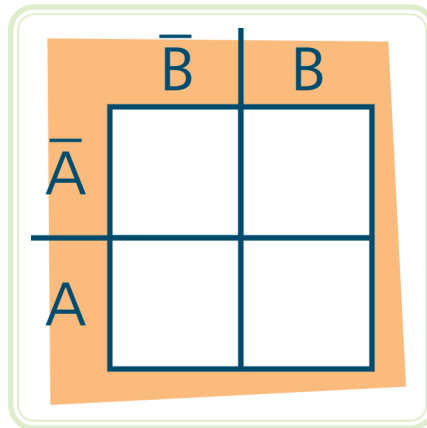


Figura 3.1: Diagrama para 2 variáveis

Fonte: CTISM

Cada linha da tabela verdade possui uma região definida no diagrama de *Karnaugh*. Essas regiões são os locais onde devem ser colocados os valores que a expressão assume nas diferentes possibilidades. Veja a Figura 3.2 que segue:

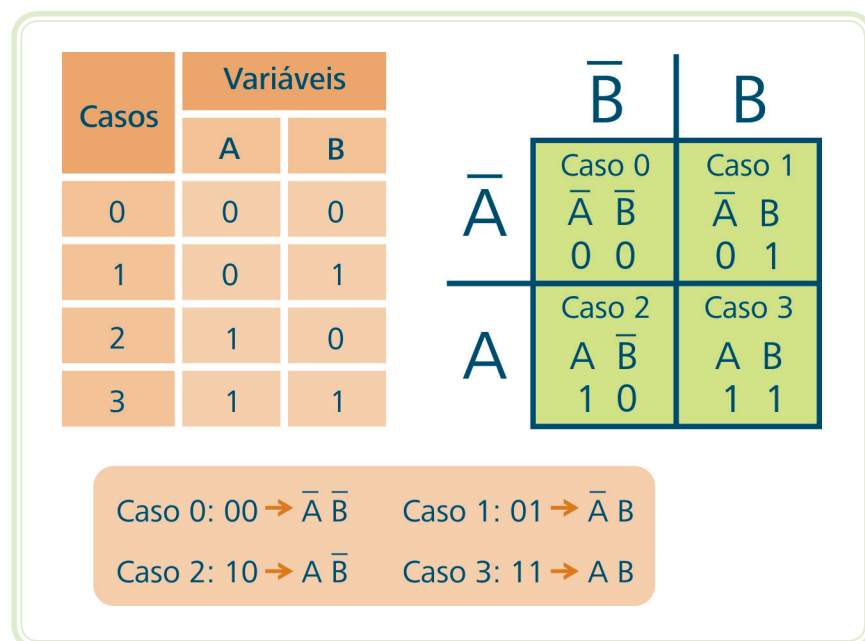


Figura 3.2: Mapa de Karnaugh para duas variáveis e correspondência com tabela verdade

Fonte: CTISM

No entanto, para que se possa obter a expressão simplificada de um diagrama de *Karnaugh*, devemos agrupar as regiões onde $S = 1$, no menor número possível de agrupamentos.

Para o caso específico de um diagrama de 2 variáveis, os agrupamentos possíveis são: quadra, pares e termos isolados.

3.2.1 Quadra

Conjunto de 4 regiões onde $S = 1$. No diagrama de 2 variáveis é o agrupamento máximo proveniente de uma tabela onde todos os casos valem 1. Dessa forma, a expressão final simplificada obtida é $S = 1$, conforme Figura 3.3 que segue:

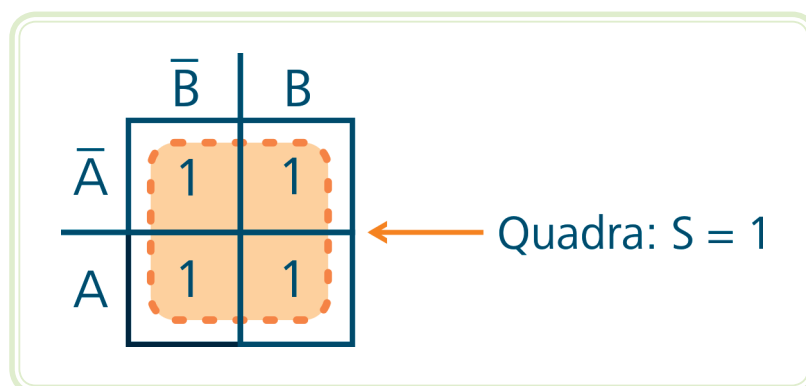


Figura 3.3: Agrupamento do tipo quadra em um mapa de duas variáveis

Fonte: CTISM

3.2.2 Pares

É o conjunto de duas regiões, onde $S = 1$ que não podem ser agrupadas na diagonal. As Figuras 3.4 e 3.5 mostram exemplos de agrupamentos pares e suas respectivas equações.

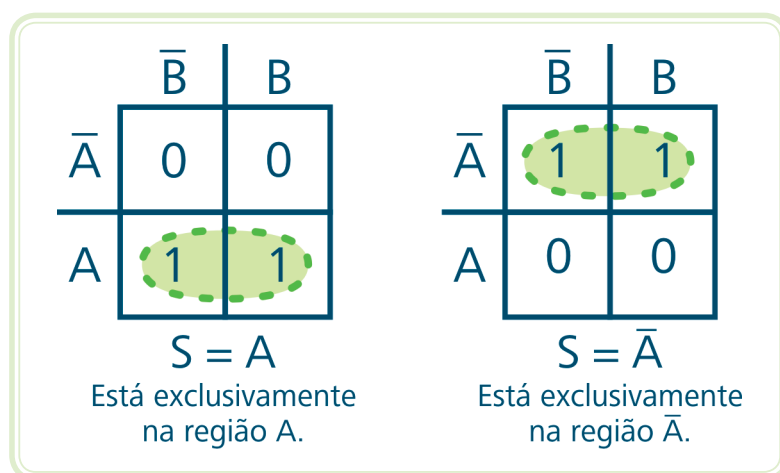


Figura 3.4: Agrupamentos do tipo dupla em um mapa de duas variáveis

Fonte: CTISM

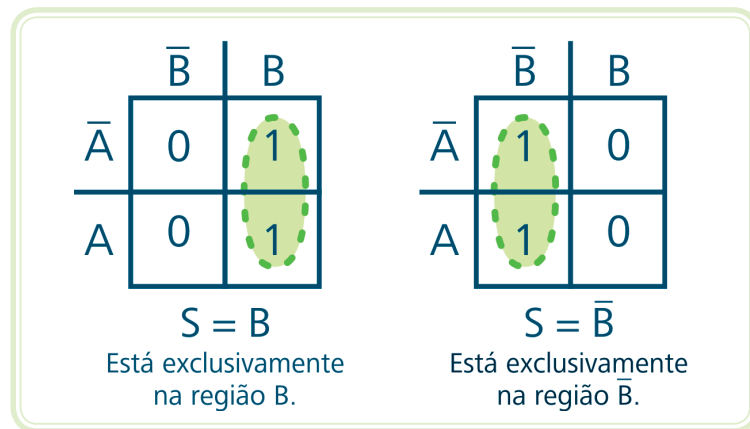


Figura 3.5: Agrupamentos do tipo dupla em um mapa de duas variáveis
Fonte: CTISM

3.2.3 Termos isolados

Região onde $S = 1$, sem vizinhança para agrupamento. Os termos isolados são os próprios casos de entrada sem simplificação. A Figura 3.6 mostra alguns exemplos e suas respectivas equações:

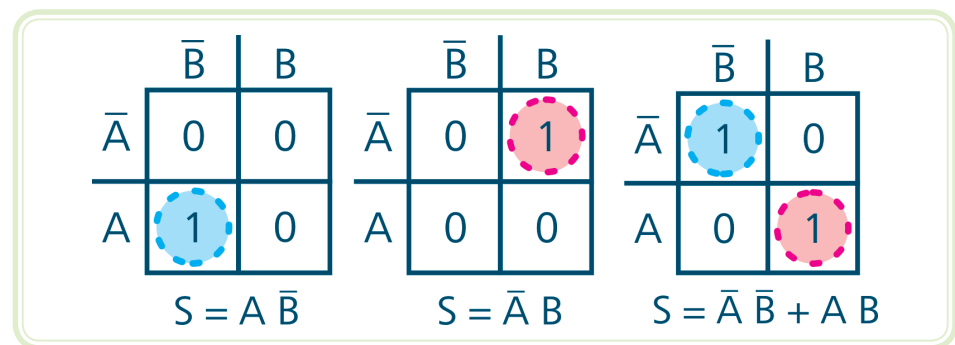


Figura 3.6: Exemplos de termos isolados em um mapa de duas variáveis
Fonte: CTISM

3.3 Diagrama (ou mapa) de Karnaugh para três variáveis

A Figura 3.7 mostra um diagrama de *Veitch-Karnaugh* para 3 variáveis:

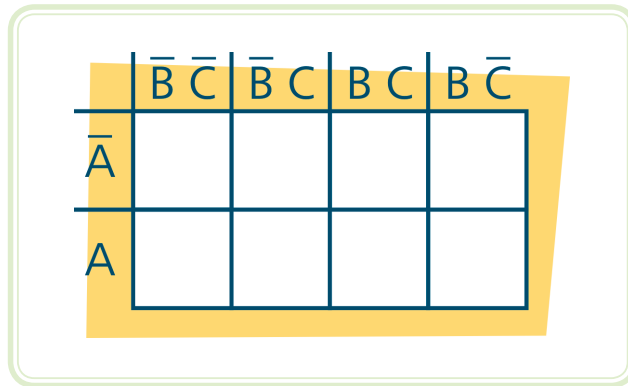


Figura 3.7: Diagrama para 3 variáveis

Fonte: CTISM

Da mesma forma que para duas variáveis, neste caso, cada linha da tabela verdade possui uma região definida no diagrama de *Karnaugh*, essas regiões são os locais onde devem ser colocados os valores que a expressão assume nas diferentes possibilidades. Veja a Figura 3.8 a seguir:

Casos	Variáveis				$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
	A	B	C					
0	0	0	0	$\bar{A}$	Caso 0 $\bar{A}\bar{B}\bar{C}$ 0 0 0	Caso 1 $\bar{A}\bar{B}C$ 0 0 1	Caso 3 $\bar{A}BC$ 0 1 1	Caso 2 $\bar{A}B\bar{C}$ 0 1 0
1	0	0	1					
2	0	1	0	A	Caso 4 $A\bar{B}\bar{C}$ 1 0 0	Caso 5 $A\bar{B}C$ 1 0 1	Caso 7 ABC 1 1 1	Caso 6 $AB\bar{C}$ 1 1 0
3	0	1	1					
4	1	0	0					
5	1	0	1					
6	1	1	0					
7	1	1	1					

Figura 3.8: Mapa de Karnaugh para três variáveis e correspondência com tabela verdade

Fonte: CTISM

No entanto, para que se possa obter a expressão simplificada de um diagrama de *Karnaugh*, devemos agrupar as regiões onde $S = 1$, no menor número possível de agrupamentos.

Para o caso específico de um diagrama de 3 variáveis, os agrupamentos possíveis são: oitava, quadra, pares e termos isolados.

3.3.1 Oitava

Agrupamento máximo onde todas as variáveis assumem o valor 1. Veja a Figura 3.9:

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

← Oitava: $S = 1$

Figura 3.9: Agrupamento do tipo oitava em um mapa de três variáveis

Fonte: CTISM

3.3.2 Quadra

Agrupamentos de 4 regiões onde $S = 1$ adjacentes ou em sequência. Seguem alguns exemplos de possíveis quadras, num diagrama de 3 variáveis, e as respectivas expressões.

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

$S = \bar{A}$

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

$S = \bar{B}$

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

$S = \bar{C}$

Figura 3.10: Agrupamento do tipo quadra em um mapa de três variáveis

Fonte: CTISM

3.3.3 Pares

Conjunto de duas regiões onde $S = 1$. Estes não podem ser agrupados na diagonal. A Figura 3.11 mostra exemplos de agrupamentos pares e sua respectiva equação.

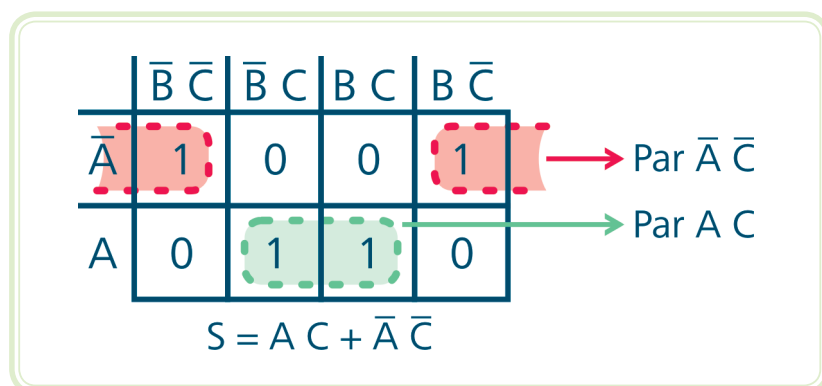


Figura 3.11: Agrupamentos do tipo dupla em um mapa de três variáveis

Fonte: CTISM

3.3.4 Termos isolados

Região onde $S = 1$, sem vizinhança para agrupamento. Os termos isolados são os próprios casos de entrada, sem simplificação. A Figura 3.12 que segue mostra exemplos e suas respectivas equações:

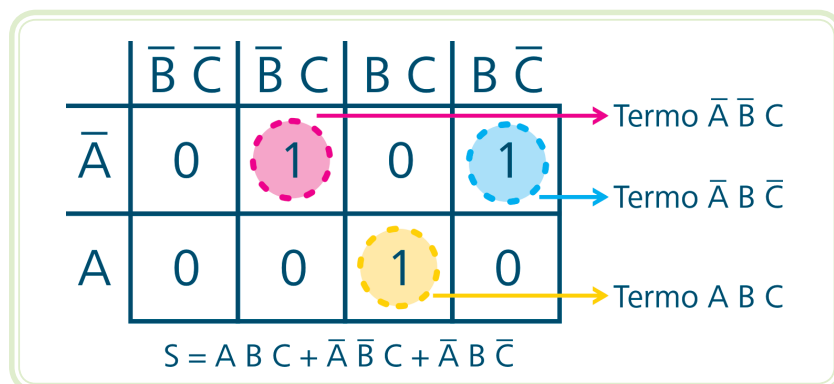


Figura 3.12: Exemplos de termos isolados em um mapa de três variáveis

Fonte: CTISM

3.4 Diagrama (ou mapa) de Karnaugh para quatro variáveis

A Figura 3.13 mostra um diagrama de Veitch-Karnaugh para 4 variáveis:

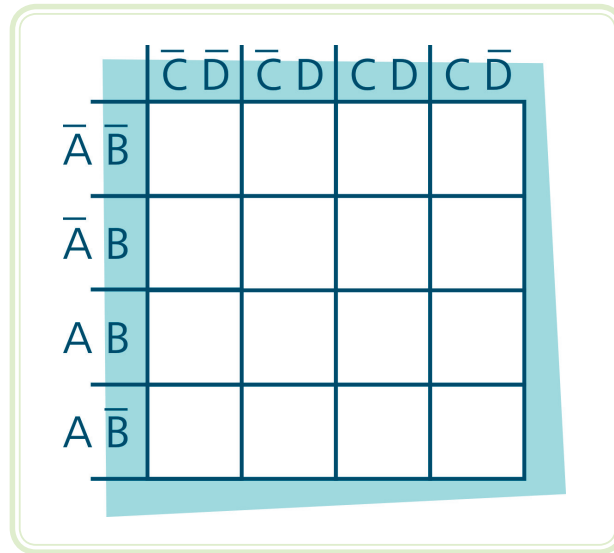


Figura 3.13: Diagrama de Karnaugh para 4 variáveis

Fonte: CTISM

Da mesma forma que para duas e três variáveis, neste caso, cada linha da tabela verdade possui uma região definida no diagrama de Karnaugh. Essas regiões são os locais onde devem ser colocados os valores que a expressão assume nas diferentes possibilidades. Veja a Figura 3.14 que segue:

Casos	Variáveis					$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
	A	B	C	D					
0	0	0	0	0	$\bar{A}\bar{B}$	Caso 0 $\bar{A}\bar{B}\bar{C}\bar{D}$ 0 0 0 0	Caso 1 $\bar{A}\bar{B}\bar{C}D$ 0 0 0 1	Caso 3 $\bar{A}\bar{B}CD$ 0 0 1 1	Caso 2 $\bar{A}\bar{B}C\bar{D}$ 0 0 1 0
1	0	0	0	1					
2	0	0	1	0	$\bar{A}B$	Caso 4 $\bar{A}B\bar{C}\bar{D}$ 0 1 0 0	Caso 5 $\bar{A}B\bar{C}D$ 0 1 0 1	Caso 7 $\bar{A}BCD$ 0 1 1 1	Caso 6 $\bar{A}BC\bar{D}$ 0 1 1 0
3	0	0	1	1					
4	0	1	0	0					
5	0	1	0	1					
6	0	1	1	0	AB	Caso 12 $AB\bar{C}\bar{D}$ 1 1 0 0	Caso 13 $AB\bar{C}D$ 1 1 0 1	Caso 15 $ABCD$ 1 1 1 1	Caso 14 $ABC\bar{D}$ 1 1 1 0
7	0	1	1	1					
8	1	0	0	0	$A\bar{B}$	Caso 8 $A\bar{B}\bar{C}\bar{D}$ 1 0 0 0	Caso 9 $A\bar{B}\bar{C}D$ 1 0 0 1	Caso 11 $A\bar{B}CD$ 1 0 1 1	Caso 10 $A\bar{B}C\bar{D}$ 1 0 1 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					

Figura 3.14: Mapa de Karnaugh para quatro variáveis e correspondência com tabela verdade

Fonte: CTISM

No entanto, para que se possa obter a expressão simplificada de um diagrama de *Karnaugh*, devemos agrupar as regiões onde $S = 1$ no menor número possível de agrupamentos.

Para o caso específico de um diagrama de 4 variáveis, os agrupamentos possíveis são: oitava, quadra, pares e termos isolados

3.4.1 Oitava

Agrupamento de oito regiões onde $S = 1$ adjacentes ou em sequência. Veja exemplos na Figura 3.15 a seguir:

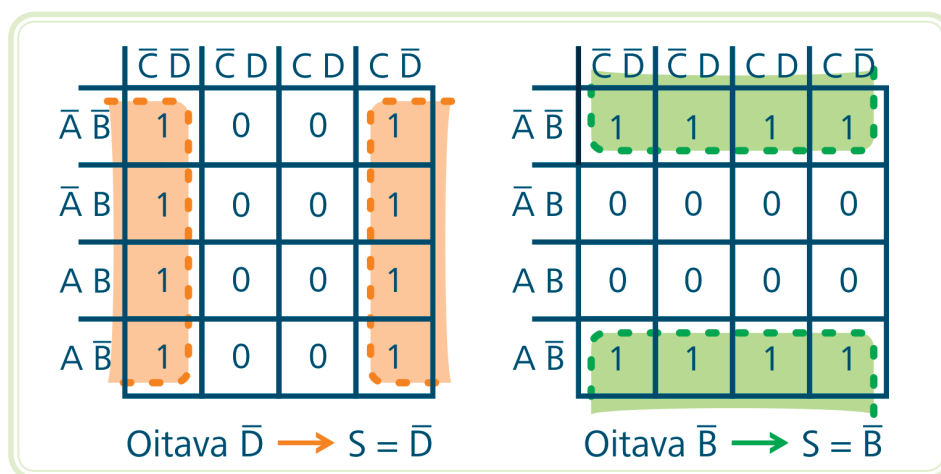


Figura 3.15: Agrupamento do tipo oitava em um mapa de quatro variáveis

Fonte: CTISM

3.4.2 Quadra

Agrupamentos de 4 regiões onde $S = 1$ adjacentes ou em sequência. Seguem alguns exemplos de possíveis quadras, num diagrama de 4 variáveis, e as respectivas expressões.

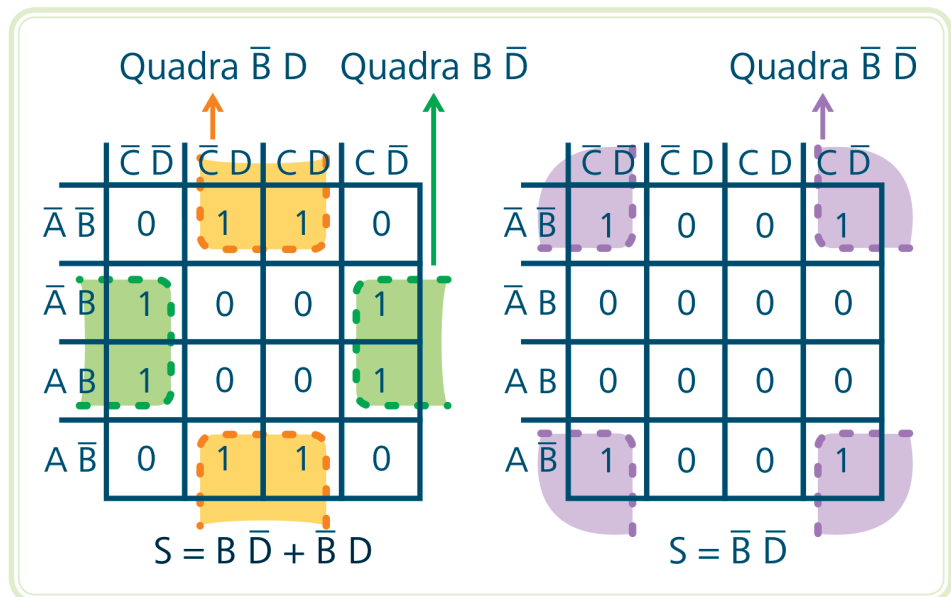


Figura 3.16: Agrupamento do tipo quadra em um mapa de quatro variáveis

Fonte: CTISM

3.4.3 Pares

Conjunto de duas regiões onde $S = 1$. Não podem ser agrupadas na diagonal. A Figura 3.17 mostra exemplos de agrupamentos pares e suas respectivas equações.

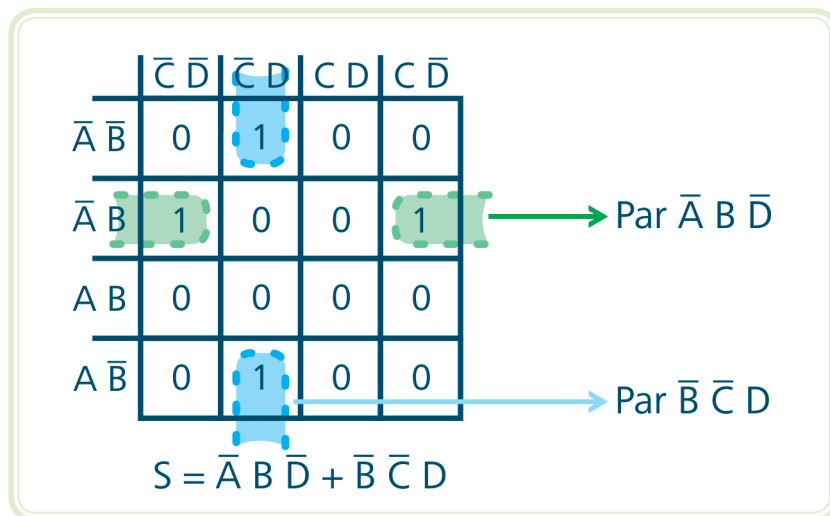


Figura 3.17: Agrupamentos do tipo dupla em um mapa de quatro variáveis

Fonte: CTISM

3.4.4 Termos isolados

Região onde $S=1$ sem vizinhança para agrupamento. São os próprios casos de entrada, sem simplificação.

3.5 Diagramas com condições irrelevantes

Condição irrelevante ocorre quando a saída pode assumir 0 ou 1, indiferentemente, para uma dada situação de entrada. Na prática, essa condição ocorre, principalmente, pela impossibilidade de a situação de entrada acontecer.

Dessa forma, os valores irrelevantes da tabela verdade devem ser transportados para o diagrama de *Karnaugh*. Assim, para efetuar as simplificações, a condição irrelevante pode ser utilizada para completar um agrupamento, minimizando a expressão característica e, conseqüentemente, o circuito lógico.

Utilizando o método de *Karnaugh*, obtenha a expressão simplificada que executa a tabela verdade a seguir:

Tabela 3.1: 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

Transpondo para o mapa de *Karnaugh* de 4 variáveis, obtem-se:

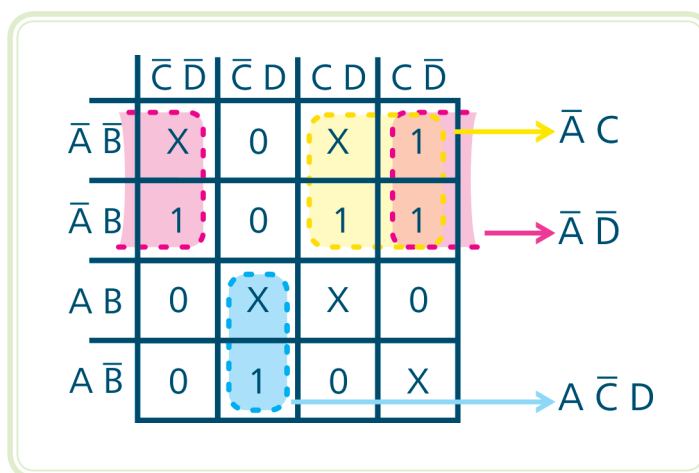


Figura 3.18: Agrupamento para a tabela verdade com condições irrelevantes

Fonte: CTISM

Utilizando-se 2 valores irrelevantes e abandonando outros 2, podem-se agrupar duas quadras e um par, gerando a seguinte expressão:

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

Logo, pode-se observar que, para simplificação de uma equação através do mapa de *Karnaugh*, é possível adotar o X tanto como nível alto “1”, quanto como nível baixo “0”, a fim de reduzi-lá ao máximo a mesma.

Resumo

Nessa aula percebemos a importância da simplificação de circuitos lógicos e aprendemos a trabalhar com mapas de *Karnaugh* para duas, três e quatro variáveis. Estamos, portanto, aptos a passar para a próxima aula que tratará de circuitos combinacionais.



Atividades de aprendizagem

1. Considerando os diagramas de *Karnaugh*, determine a expressão simplificada de S_1 e S_2 da tabela a seguir:

Tabela verdade: Exercício 1			
A	B	S_1	S_2
0	0	1	1
0	1	0	1
1	0	1	0
1	1	1	0

2. Simplifique as expressões de S_1 , S_2 , S_3 e S_4 da tabela verdade a seguir, utilizando os mapas de *Karnaugh*.

Tabela verdade: Exercício 2						
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

3. Simplifique as expressões de S_1 , S_2 , S_3 e S_4 da tabela verdade a seguir, utilizando os mapas de *Karnaugh*.

Tabela verdade: Exercício 3							
A	B	C	D	S_1	S_2	S_3	S_4
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

4. Determine as expressões simplificadas de S_1 , S_2 , S_3 e S_4 da tabela a seguir.

Tabela verdade: Exercício 4							
A	B	C	D	S_1	S_2	S_3	S_4
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

Aula 4 – Circuitos combinacionais

Objetivos

Conhecer e identificar os circuitos lógicos combinacionais.

Projetar circuitos lógicos combinacionais com a finalidade de resolver problemas que envolvam variáveis lógicas de entrada e de saída.

4.1 Definição de circuitos combinacionais

Sabemos que a linguagem com a qual nos expressamos não é a mesma que os computadores e demais circuitos digitais entendem. Dessa forma, é necessária a utilização de codificadores e decodificadores, a fim de se converter em informações de um determinado código de numeração para outro.

Os circuitos que executam essas e outras atividades muito importantes na eletrônica digital são agrupados em uma categoria de circuitos denominados circuitos combinacionais.

Um circuito combinacional é aquele em que sua saída depende única e exclusivamente das combinações entre as diversas variáveis de entrada.

4.2 Projetos de circuitos lógicos combinacionais

Além dos casos já citados de conversão de unidades, como a conversão de binário para decimal, podemos utilizar um circuito lógico combinacional para solucionar um problema em que se necessita de uma resposta diante de determinadas situações representadas pelas variáveis de entrada.

Na Figura 4.1 a seguir, verificamos a sequência dos procedimentos necessários para se construir um circuito lógico combinacional.

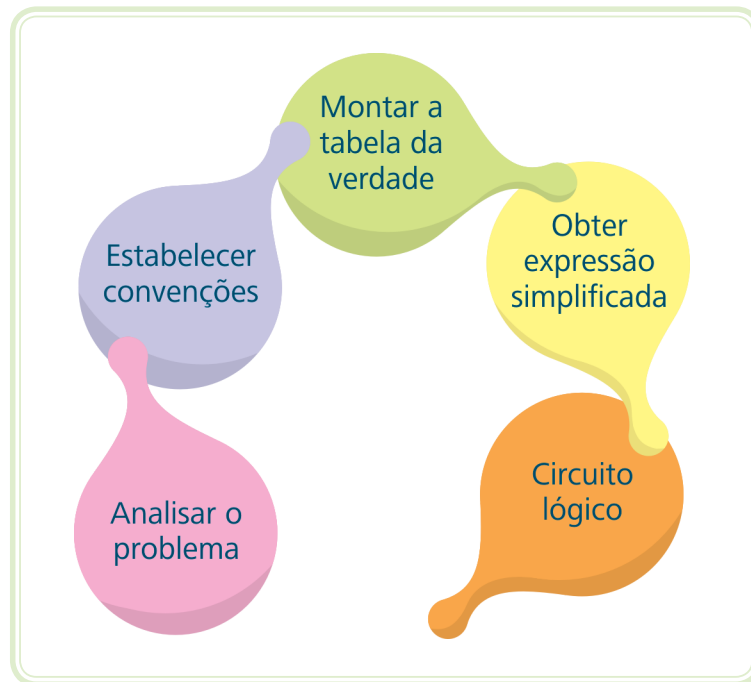


Figura 4.1: Sequência de projeto de um circuito lógico combinacional
Fonte: CTISM

Vamos supor que um grande fabricante de aparelhos eletrônicos deseja fabricar um equipamento de áudio que compartilhe um único amplificador de som para ligar três aparelhos: um toca-CDs, um toca-fitas e um rádio AM/FM.

O fabricante determinou ainda que o circuito lógico combinacional deverá funcionar da seguinte maneira: se o toca-CDs e o toca fitas estiverem desligados, o rádio AM/FM, se ligado, será conectado à entrada do amplificador. Caso o toca-fitas seja ligado, o circuito deverá conectá-lo à entrada do amplificador, pois possui prioridade sobre o rádio e assim por diante. A Figura 4.2 apresenta os seguintes passos:

a) Analisar o problema

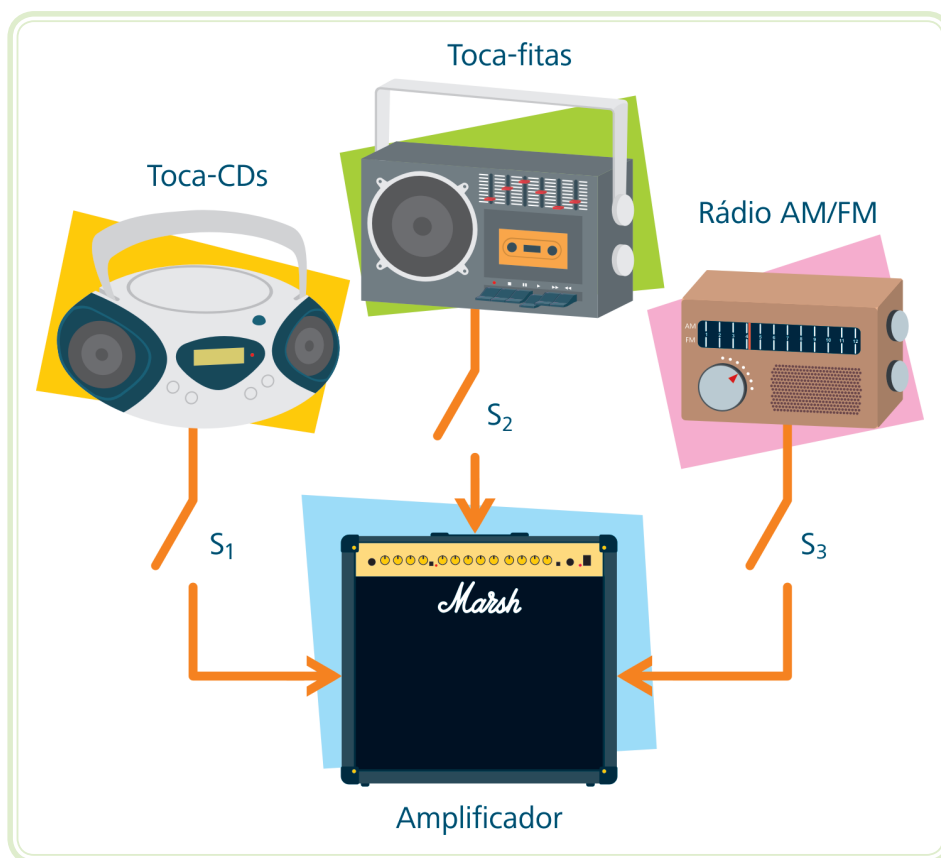


Figura 4.2: Circuito analisado

Fonte: CTISM

O circuito lógico deverá ligar os aparelhos obedecendo às seguintes prioridades:

1ª prioridade: Toca-CDs

2ª prioridade: Toca-fitas

3ª prioridade: Rádio AM/FM

b) Estabelecer convenções

Variáveis de entrada: aparelho desligado = 0 e aparelho ligado = 1.

A = Toca-CDs

B = Toca-fitas

C = Rádio AM/FM

Chaves S_1 , S_2 e S_3 : chave aberta = 0 e chave fechada = 1.

c) Montar a tabela verdade

Entradas			Saídas		
A	B	C	S_1	S_2	S_3
0	0	0	X	X	X
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0

d) Obter a expressão simplificada

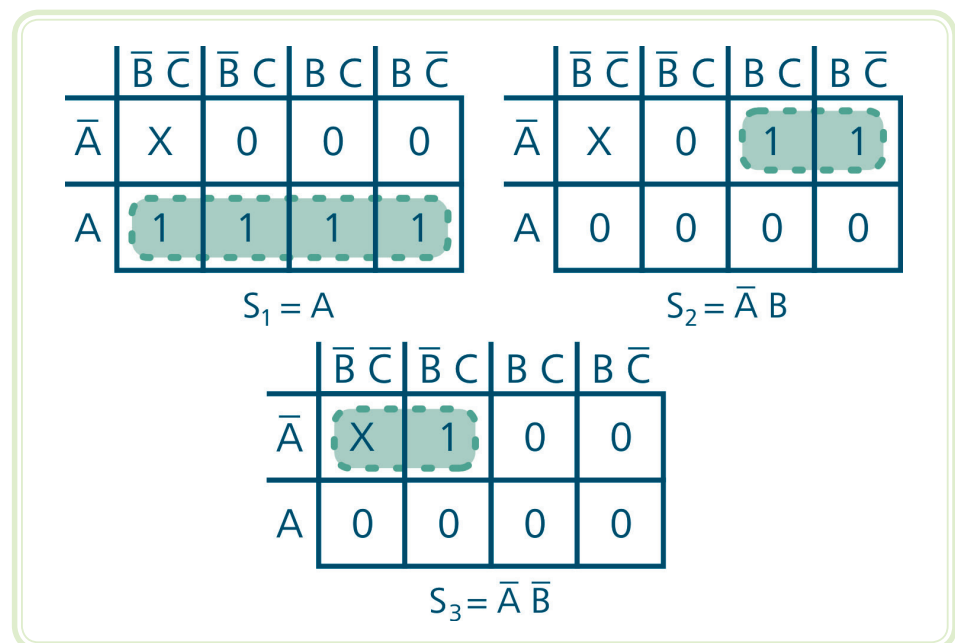


Figura 4.3: Mapas de Karnaugh para o circuito analisado

Fonte: CTISM

e) Circuito lógico

Dessa forma, temos o circuito lógico combinacional desejado que é obtido das expressões simplificadas e fica sendo:

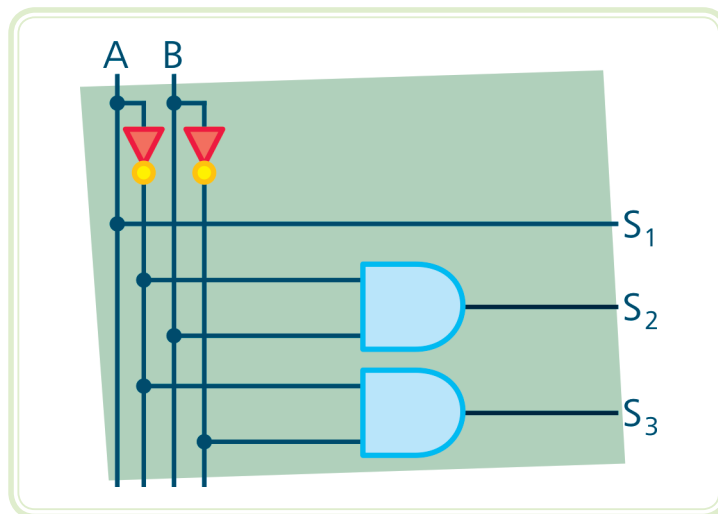


Figura 4.4: Diagrama final do circuito analisado

Fonte: CTISM

4.3 Projeto de circuitos codificadores e decodificadores

Os codificadores são circuitos combinacionais que possibilitam a passagem de um código conhecido para um desconhecido. Os circuitos decodificadores fazem o inverso, ou seja, passam um código desconhecido para um conhecido.

Os equipamentos digitais e alguns sistemas de computação têm seus dados de entrada expressos em decimal, facilitando o trabalho do operador. Entretanto, esses dados são processados internamente em binário, e o trabalho de conversão é realizado pelos circuitos codificadores. Os dados já processados são novamente convertidos em decimal na forma compatível para um mostrador digital apresentar os algarismos. Este trabalho é feito pelos circuitos decodificadores. Verifique a Figura 4.5 que segue:

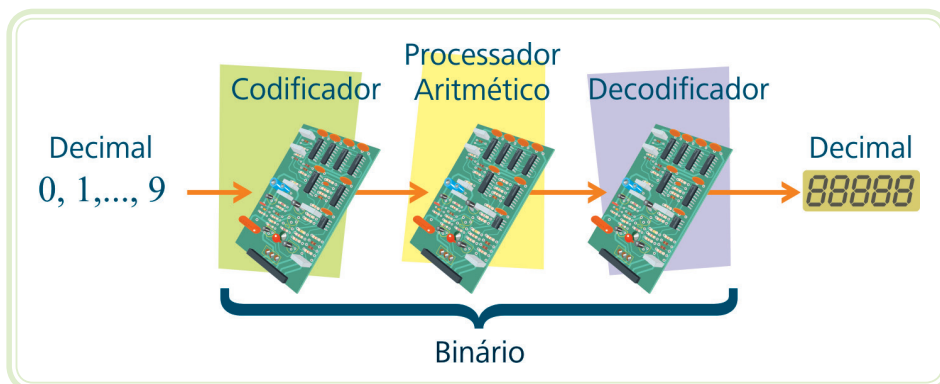


Figura 4.5: Codificadores e decodificadores

Fonte: CTISM

Para exemplificar, vamos desenvolver o circuito lógico combinacional de um decodificador de binário para *display* de *led* de sete segmentos, de acordo com a Figura 4.6:

Consulte o *link* que segue e informe-se sobre o tema:
http://pt.wikipedia.org/wiki/Display_de_sete_segmentos

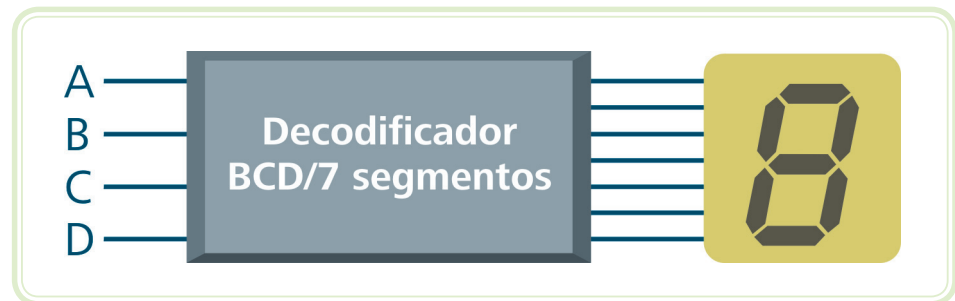


Figura 4.6: Decodificador BCD-7 segmentos

Fonte: CTISM

O *display* de 7 segmentos possibilita a escrita de números decimais de 0 a 9 e alguns outros símbolos que podem ser letras ou sinais. A Figura 4.7 a seguir ilustra um *display* genérico com a nomenclatura de identificação dos segmentos.

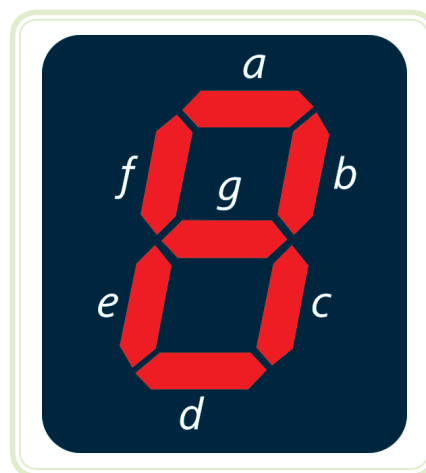


Figura 4.7: Display de *led* de sete segmentos

Fonte: CTISM

Descreveremos o código binário de entrada e o código de saída correspondentes a cada uma das entradas, considerando o nível lógico 1 como segmento (*led*) aceso, e o nível lógico zero como segmento (*led*) apagado. Dessa forma, temos para os números de 0 a 9, a seguinte tabela:

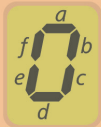
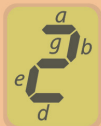
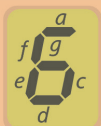
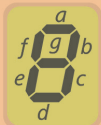
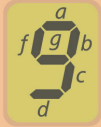
Caracteres	Display	BCD 8421				Código para 7 segmentos						
		A	B	C	D	a	b	c	d	e	f	g
0		0	0	0	0	1	1	1	1	1	1	0
1		0	0	0	1	0	1	1	0	0	0	0
2		0	0	1	0	1	1	0	1	1	0	1
3		0	0	1	1	1	1	1	1	0	0	1
4		0	1	0	0	0	1	1	0	0	1	1
5		0	1	0	1	1	0	1	1	0	1	1
6		0	1	1	0	1	0	1	1	1	1	1
7		0	1	1	1	1	1	1	0	0	0	0
8		1	0	0	0	1	1	1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

Figura 4.8: Códigos para 7 segmentos
 Fonte: CTISM

Para simplificar o circuito de saída basta utilizar o diagrama de *Karnaugh*. Os termos que não são representados na tabela serão considerados irrelevantes.

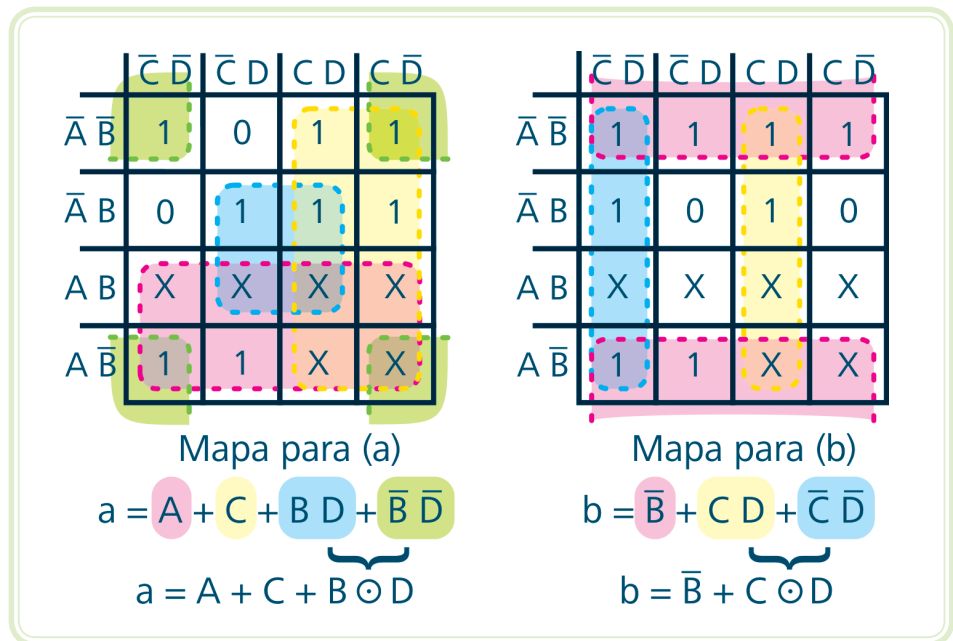


Figura 4.9: Mapas de Karnaugh e expressões finais para segmentos (a) e (b)
Fonte: CTISM

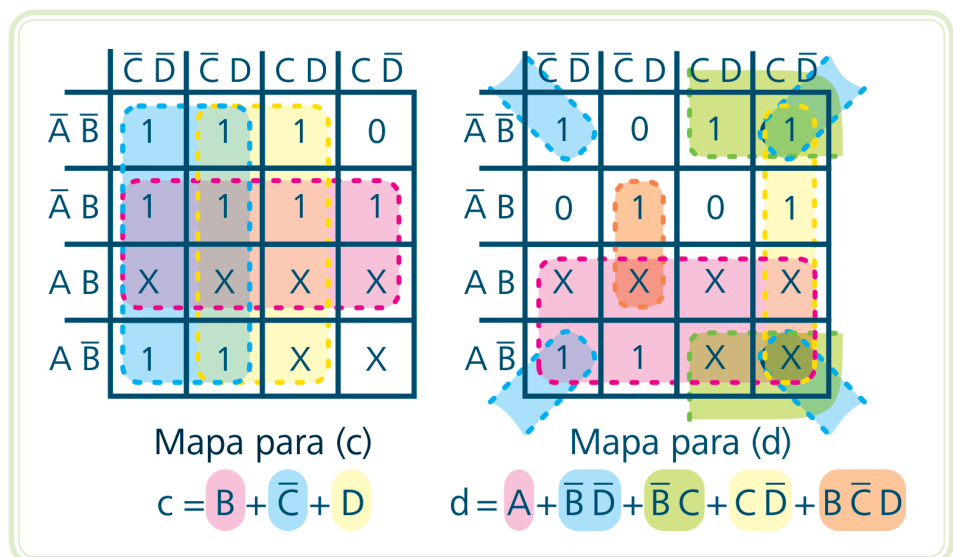


Figura 4.10: Mapas de Karnaugh e expressões finais para segmentos (c) e (d)
Fonte: CTISM

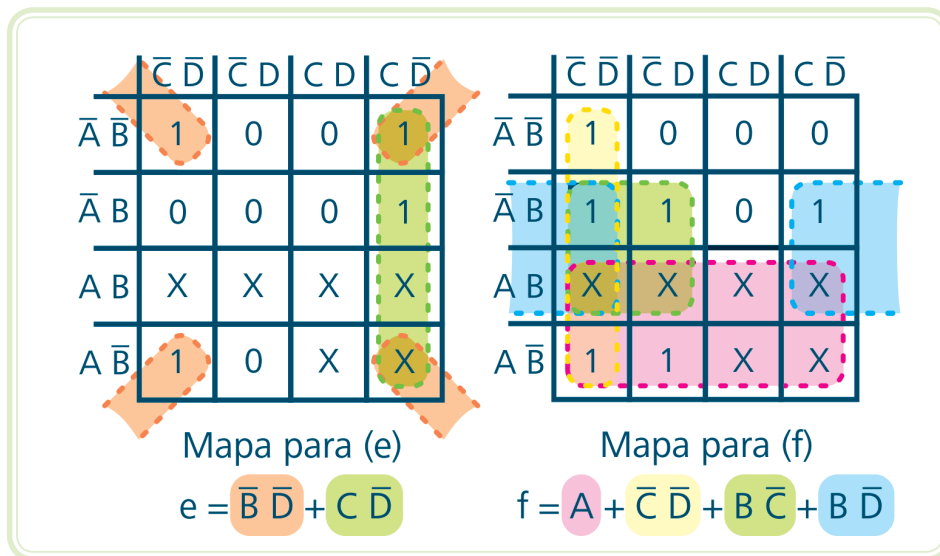


Figura 4.11: Mapas de Karnaugh e expressões finais para segmentos (e) e (f)
Fonte: CTISM

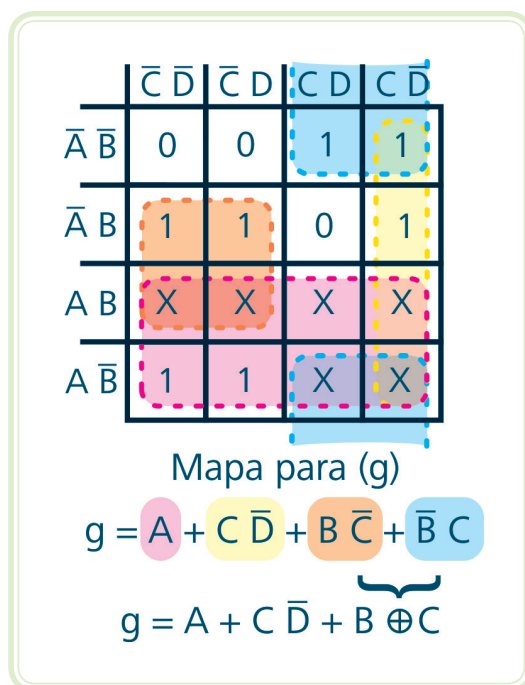


Figura 4.12: Mapas de Karnaugh e expressões finais para segmento (g)
Fonte: CTISM

O circuito lógico obtido nas expressões simplificadas pode ser visto na Figura 4.13 a seguir:

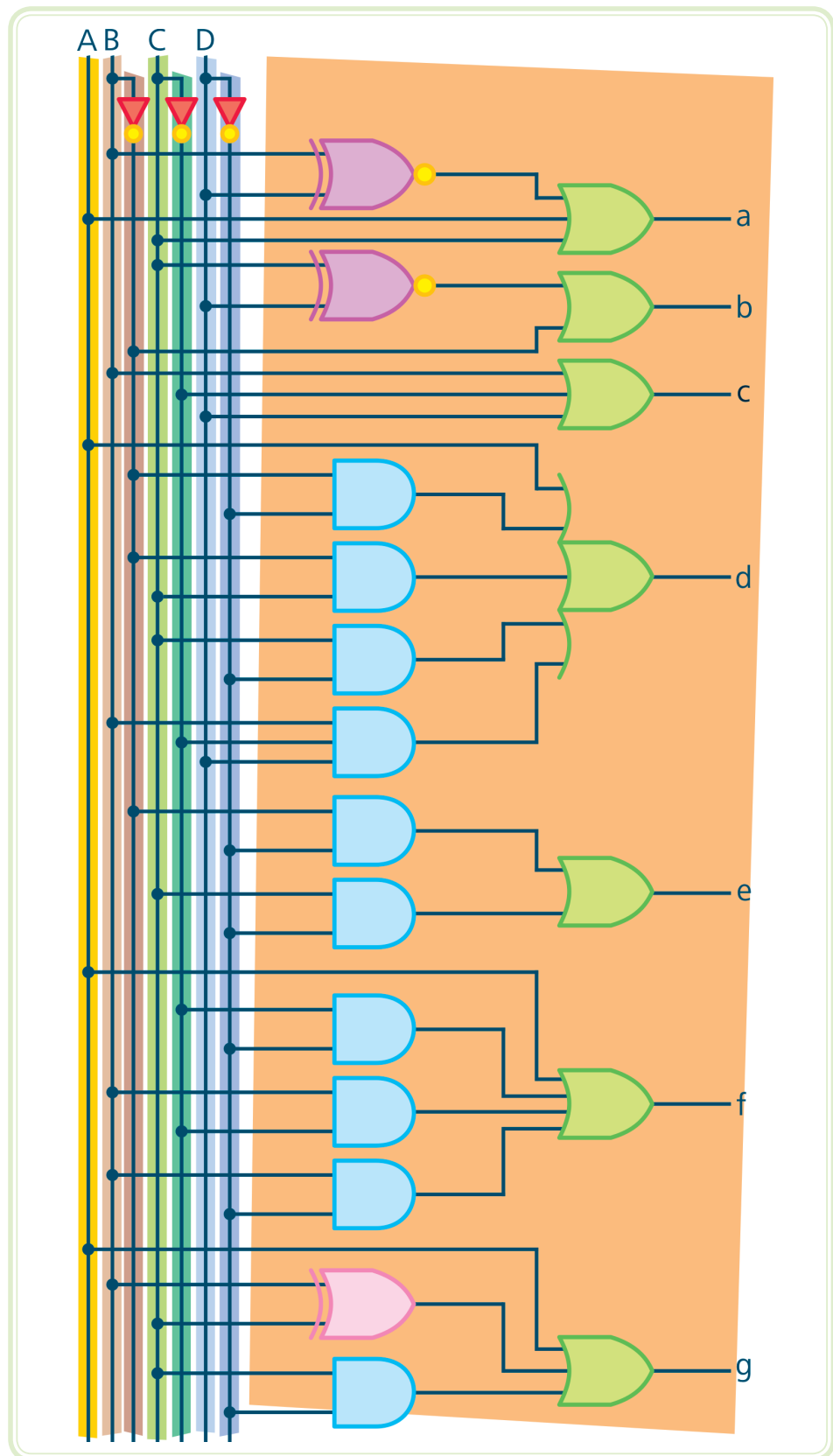


Figura 4.13: Circuito lógico combinacional equivalente de um decodificador de binário para 7 segmentos

Fonte: CTISM

Resumo

Nessa aula aprendemos a conhecer e a identificar os circuitos lógicos combinacionais. Também verificamos que, seguindo a sequência dos procedimentos necessários, podemos projetar circuitos lógicos combinacionais com a finalidade de resolver problemas que envolvam variáveis lógicas de entrada e de saída, de maneira simples, rápida e eficiente.

Atividades de aprendizagem



1. Elabore um circuito lógico combinacional para controlar o nível de um tanque industrial por meio de duas eletroválvulas, uma para a entrada e outra para a saída do líquido. O circuito deverá atuar nas eletroválvulas para encher o tanque totalmente, assim que for ligado o botão de comando, e deverá esvaziá-lo no momento em que o mesmo botão for desligado. Há um sensor de nível na parte superior do tanque que indica quando o tanque está cheio, e outro, na parte inferior que indica quando o tanque está totalmente vazio.
2. Desenhe um circuito para, em um conjunto de três chaves, detectar um número ímpar de chaves ligadas. Convencionar que chave fechada equivale a nível 0.

Aula 5 – Circuitos sequenciais

Objetivos

Construir e analisar o funcionamento de *flip-flops* com portas *NAND* e *NOR*.

Conhecer os diferentes tipos de *flip-flops*.

5.1 Definição de circuitos sequenciais

Os circuitos que compõem a eletrônica digital são divididos em dois grandes grupos: os lógicos combinacionais e os lógicos sequenciais.

Vimos que os circuitos combinacionais apresentam as saídas dependentes exclusivamente de suas variáveis de entrada. Os circuitos sequenciais, por sua vez, têm as suas saídas dependentes não somente de suas variáveis de entrada, mas também do estado anterior de suas saídas que permanece armazenado e opera sob o comando de uma sequência de pulsos denominada “*clock*”.

5.2 Flip-flops (ou biestáveis)

Os *flip-flops* são os circuitos sequenciais mais elementares. Possuem a capacidade de armazenar a informação neles contida. Representam a unidade elementar de memória de 1 *bit* (*binary digit*), ou seja, funcionam como um elemento de memória por armazenar níveis lógicos temporariamente. São chamados de biestáveis, porque possuem dois estados lógicos estáveis, geralmente representados por “0” e “1”.

De forma geral, podemos representar o *flip-flop* como um bloco que tem duas saídas: Q e $\bar{Q}$ (barrado), entradas para as variáveis e uma entrada de controle (*clock*).

Este dispositivo possui basicamente dois estados de saída:

$$Q = 0 \text{ e } \bar{Q} = 1; Q = 1 \text{ e } \bar{Q} = 0$$



O link abaixo ajudará a complementar informações:
<http://pt.wikipedia.org/wiki/flip-flop>

Para o *flip-flop* assumir um desses estados, é necessário que haja uma combinação das variáveis e do pulso de controle. Após esse pulso, o *flip-flop* permanecerá nesse estado até a chegada de um novo pulso de *clock* e, de acordo com as variáveis de entrada, mudará ou não seus estados de saída.

5.2.1 Latch

A forma mais básica de implementar um circuito lógico de memória é conhecida como ***latch***, que significa, em português, **trinco**, **ferrolho**. Sua arquitetura é composta de duas portas lógicas inversoras, possuindo duas saídas: a variável lógica **Q** e o seu complemento lógico, **Q barra**. Veja a Figura 5.1:

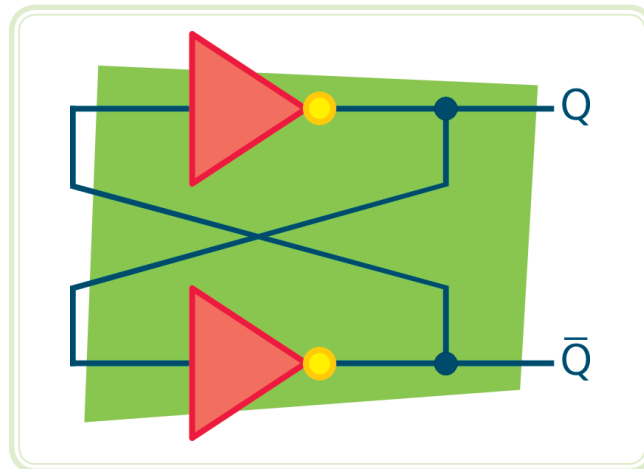


Figura 5.1: Latch básico

Fonte: CTISM

Note que, se você impõe nível lógico alto (1) em **Q**, seu complemento vai para o nível lógico baixo (0). Esse estado (**Q barra = 0**) permanecerá até que você imponha nível lógico baixo a **Q**.

Concluimos que um *latch* é um dispositivo de memória com capacidade de armazenar um único *bit*. Se você precisar armazenar uma palavra de mais de um *bit*, você precisará de um *latch* para cada *bit* (por exemplo, uma palavra de 32 *bits* precisa de um dispositivo de memória de 32 *latch*'s para ser armazenada).

5.2.2 Latch-SR

Pode-se, também, construir um *latch* com outras portas lógicas (*OR* e *AND*) e, de quebra, pode-se disponibilizar entradas para o *latch*. Um *latch* construído dessa forma é chamado *LATCH-SR*. Veja o *latch-SR* construído com porta *NAND*:

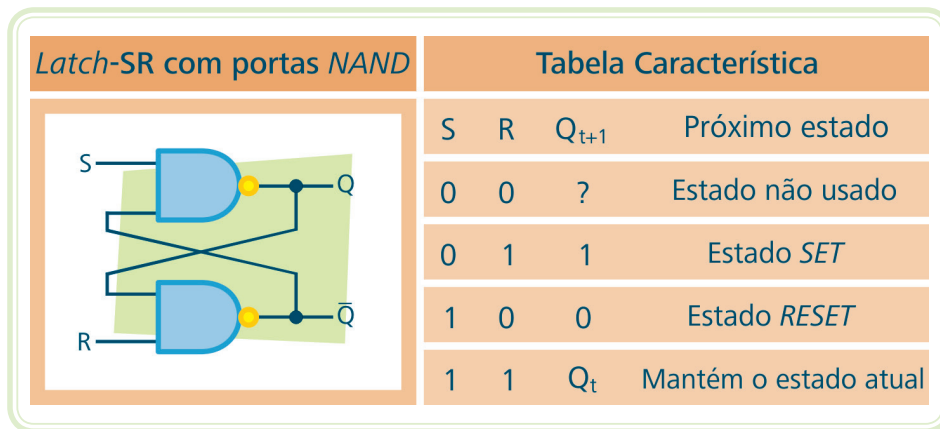


Figura 5.2: Latch-SR com portas NAND e sua tabela verdade característica
 Fonte: CTISM

Note que esse *latch*-SR possui duas portas NAND entrelaçadas com duas entradas, **S** e **R**. Também possui duas saídas, uma denominada **Q**, e a outra o complemento de **Q**. Independentemente dos valores lógicos atribuídos a **S** e a **R**, essas variáveis são referências aos valores da variável de estado do *latch*-SR. Em primeiro lugar, especifica-se o estado do *latch*-SR através do par **Q** e seu complemento.

A outra implementação de *latch* com duas entradas é realizada com o uso de portas **NOR**. Veja a Figura 5.3:

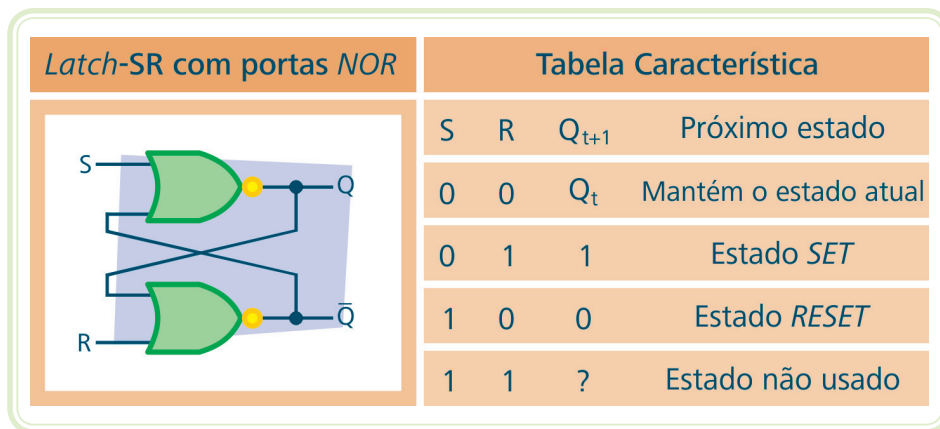


Figura 5.3: Latch-SR com portas NOR e sua tabela verdade característica
 Fonte: CTISM

Note que a diferença entre as duas implementações está na combinação SR que leva ao estado indefinido.

Para entender o funcionamento de um *latch*-SR, vamos considerar o *flip-flop* RS básico, construído a partir de portas NAND e inversores, conforme Figura 5.4 a seguir:

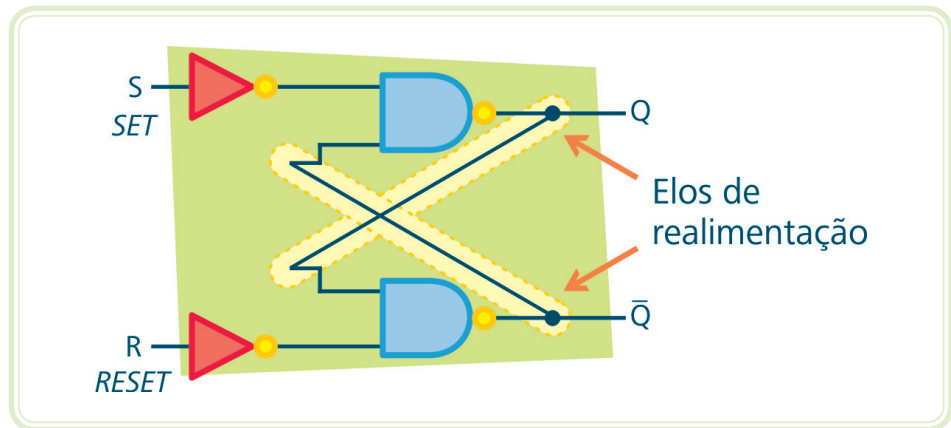


Figura 5.4: Latch-SR com portas NAND e portas inversoras

Fonte: CTISM

Notamos que os elos de realimentação fazem com que as saídas sejam injetadas juntamente com as variáveis de entrada, ficando claro, então, que os estados que as saídas irão assumir dependerão de ambas. A entrada S é denominada *Set*, pois, quando acionada (nível 1), passa a saída para 1 (estabelece ou fixa 1); a entrada R é denominada *Reset*, pois, quando acionada (nível 1), passa a saída para 0 (recompõe ou zera o *flip-flop*). Esses termos, são provenientes da língua inglesa, usuais na área de eletrônica digital. Esse circuito mudará de estado apenas no instante em que mudam as variáveis de entrada.

5.2.3 Latch-SR com entrada de clock

É claro que o aparecimento de estado indefinido representa uma desvantagem dos *latches*-SR. Um avanço possível na direção da eliminação desse problema é a inclusão de uma terceira entrada de controle, **C**. Seu diagrama lógico e a respectiva tabela característica são mostrados na Figura 5.5. Podemos verificar esta entrada de controle (*clock*) responsável por “habilitar” o *latch*. O objetivo principal do *clock* em um *latch*-SR é restringir entradas que possam afetar o estado do *latch*.

Portanto, nessa configuração o *flip-flop* RS passa a ser controlado por uma sequência de pulsos de *clock*. Para que isso seja possível, basta substituímos os inversores por portas NAND e, às outras entradas destas portas, conectarmos o *clock*.

No circuito da Figura 5.5, quando a entrada do *clock* for igual a 0, o *flip-flop* irá permanecer no seu estado, mesmo que variem as entradas S e R. A partir de uma análise do circuito, podemos concluir que para *clock*=0, as saídas das portas NAND de entrada serão sempre iguais a 1, independentemente dos valores assumidos por S e R. Quando a entrada *clock* assumir valor 1, o

circuito irá comportar-se como um *flip-flop* RS básico, pois as portas *NAND* de entrada funcionarão como os inversores do circuito RS básico. De maneira geral, podemos concluir que o circuito irá funcionar quando a entrada do *clock* assumir valor 1 e manterá travada esta saída quando a entrada *clock* passar para 0.

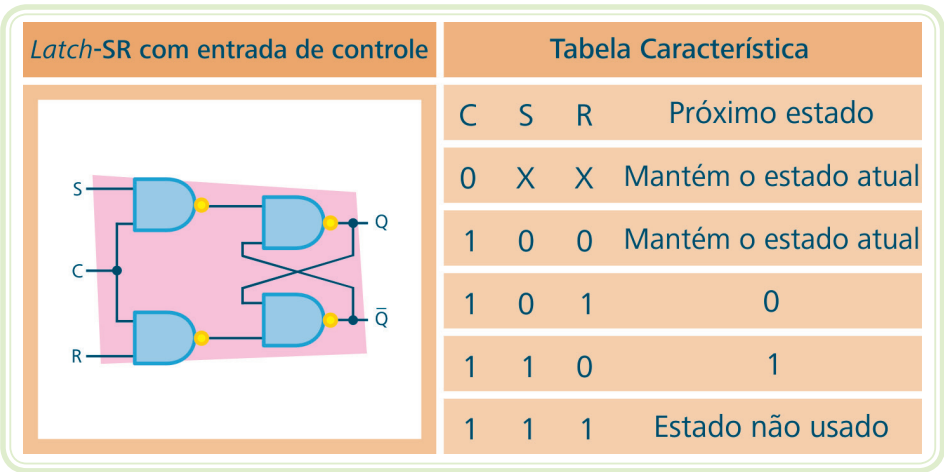


Figura 5.5: Latch-SR com entrada de clock
Fonte: CTISM

5.2.4 Flip-flop JK

Até agora, temos evitado fazer S e R tal que $S = R = 1$, pois tal procedimento tentaria ajustar (*set*) e reajustar (*reset*) o *flip-flop* ao mesmo tempo, e o resultado seria ambíguo. Vamos modificar o *flip-flop* para permitir $S = R = 1$ e observaremos que o *flip-flop* modificado possui a propriedade que, quando $S = R = 1$, ele chaveia, isto é, muda de estado a cada transição de gatilho do relógio. A modificação consiste em prover terminais adicionais nas portas de entrada e em fazer ligações entre as saídas e as entradas, conforme o que mostra na Figura 5.6. O *flip-flop* JK nada mais é que um *flip-flop* RS realimentado, como mostra a Figura 5.6. O terminal de dados anteriormente chamado S é, agora, chamado J, e o terminal de dados R é chamado K.

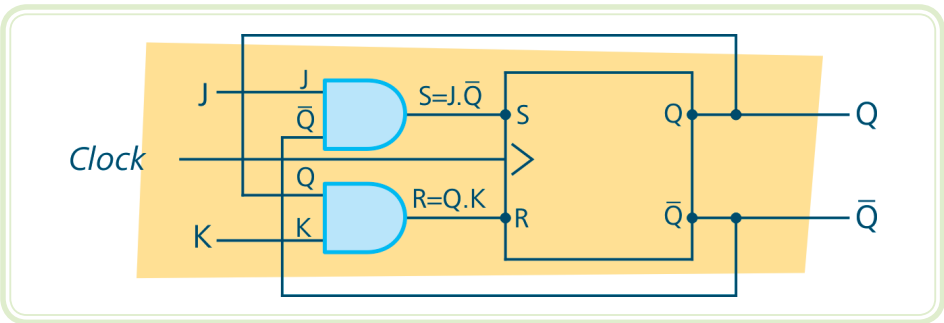


Figura 5.6: Flip-flop JK
Fonte: CTISM

Na ausência dessa modificação, os níveis lógicos em S e R dirigiam o sinal de *clock*, isto é, dependendo de S e R, uma ou outra das portas de entrada era habilitada, e o *clock* ajustava ou reajustava (*set* ou *reset*) o *flip-flop*. A razão da modificação é fazer com que a direção do sinal de *clock* seja determinada não só por S e R, mas também pelo estado do *flip-flop*. O *flip-flop* JK é uma configuração em que a saída de um *flip-flop* é ligada à entrada de um *flip-flop*. Nesse caso, acontece que a saída e a entrada pertencem ao mesmo *flip-flop*.

A tabela verdade de um **flip-flop** JK é mostrada a seguir:

Tabela 5.1: Tabela verdade do <i>flip-flop</i> JK		
J	K	Q
0	0	Q_a
0	1	0
1	0	1
1	1	$\overline{Q}_a$

5.2.5 Flip-flop JK mestre-escravo

A tabela verdade é exatamente a mesma que a do *flip-flop* JK comum. A única diferença está no funcionamento interno do circuito que, nesse caso, sempre funcionará por borda de transição de CLK. É um circuito bastante usado comercialmente. Pode possuir, além das entradas mencionadas, as entradas PR (*PRESET*) e CLR (*CLEAR*).

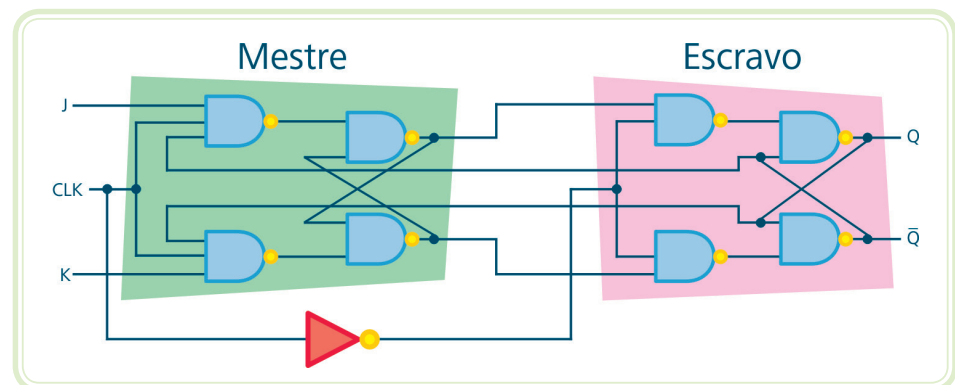


Figura 5.7: Flip-flop JK mestre-escravo
Fonte: CTISM

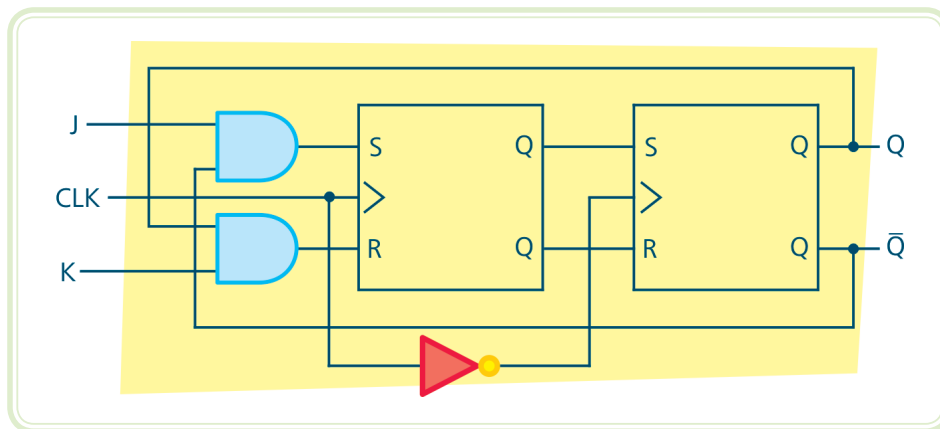


Figura 5.8: Simbologia usual do *flip-flop* JK mestre-escravo

Fonte: CTISM

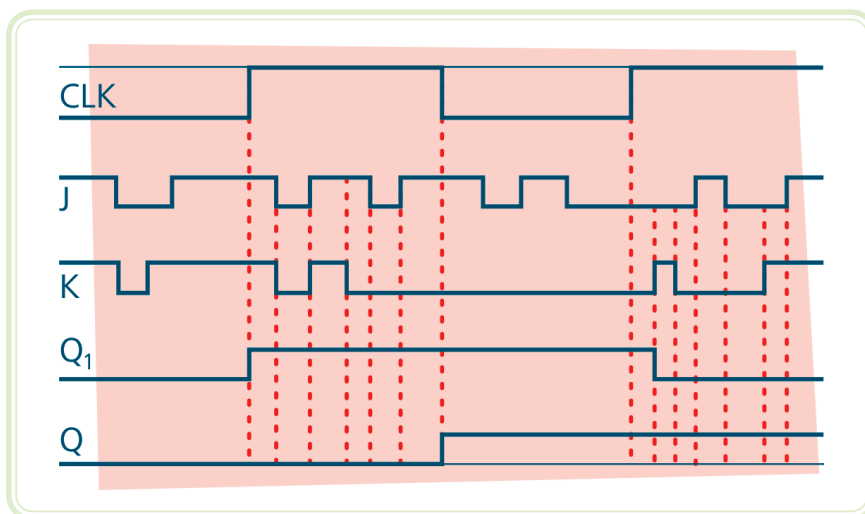


Figura 5.9: Diagrama de tempos de um *flip-flop* JK mestre-escravo

Fonte: CTISM

A Figura 5.9 refere-se a diagramas temporais que serão explicados no item 5.3.



5.2.6 Flip-flop tipo T

Este *flip-flop* é obtido a partir de um JK com as entradas J e K curto-circuitadas, logo, quando J assumir valor 1, K também assumirá valor 1 e, quando J assumir valor 0, K também assumirá valor 0. Obviamente, no caso desta ligação, não irão ocorrer nunca entradas como:

$$J = 0 \text{ e } K = 1; J = 1 \text{ e } K = 0$$

Devido ao fato de o *flip-flop* tipo T, com a entrada T igual a 1, complementar a saída (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). O *flip-flop* tipo T não é encontrado na série de circuitos integrados comerciais, sendo na prática montado a partir de um JK.

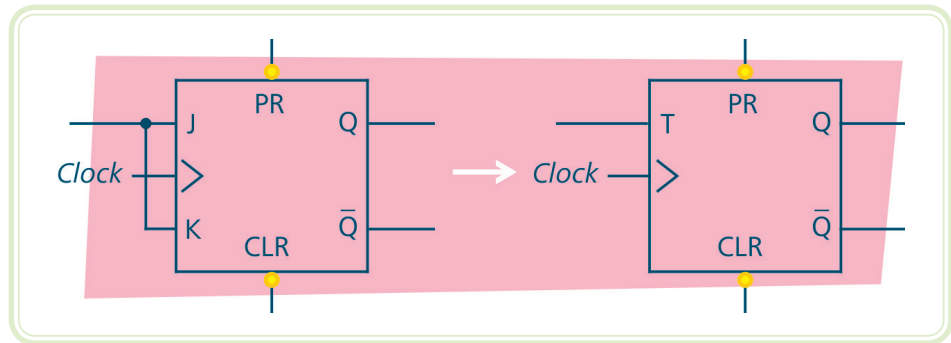


Figura 5.10: Flip-flop tipo T

Fonte: CTISM

Tabela 5.2: Tabela verdade do *flip-flop* tipo T

T	Q
0	Q_a
1	Q_a

5.2.7 Flip-flop tipo D

É obtido a partir de um flip-flop JK com a entrada K invertida (por inversor) em relação a J. Logo, neste *flip-flop*, teremos as seguintes entradas possíveis:

$J = 0$ e $K = 1$; $J = 1$ e $K = 0$

Obviamente não irão ocorrer os casos:

$J = 0$ e $K = 0$; $J = 1$ e $K = 1$

A Figura 5.11 que segue mostra como este é obtido e seu bloco representativo.

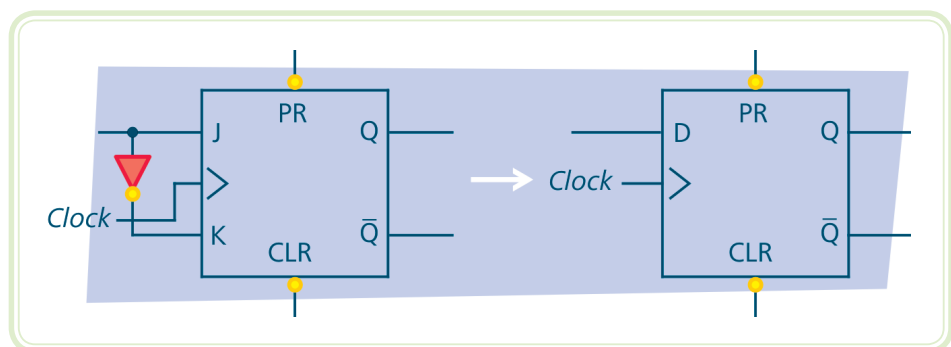


Figura 5.11: Flip-flop tipo D

Fonte: CTISM

Pela capacidade de passar para a saída (Q) e armazenar o dado aplicado na entrada D, este *flip-flop* será empregado como célula de registradores do deslocamento e de outros sistemas de memória. A sigla D vem de *Data* (dado), termo original em inglês.

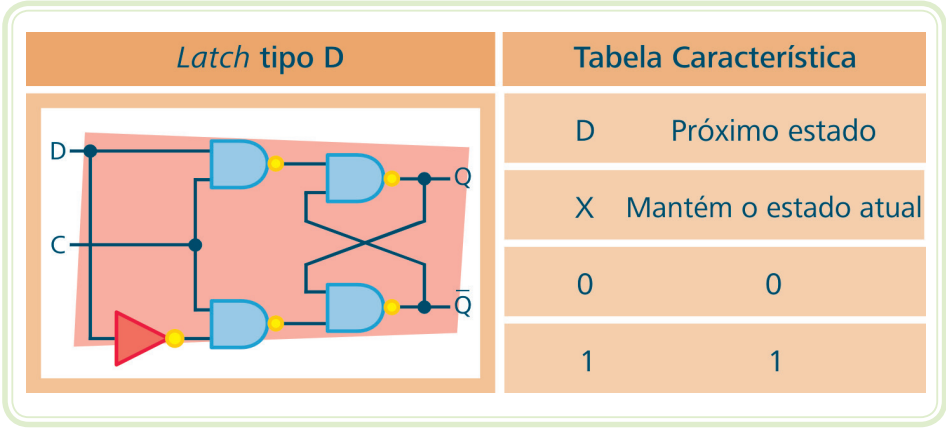


Figura 5.12: *Flip-flop* tipo D e sua tabela verdade

Fonte: CTISM

5.3 Diagramas temporais com *flip-flops*

Para entender o funcionamento dos *flip-flops* ao longo do tempo, utilizam-se diagramas temporais. Veja a Figura 5.13 onde temos um *flip-flop* S-R.

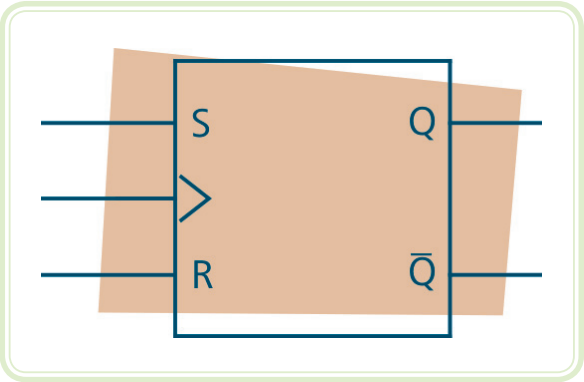


Figura 5.13: *Flip-flop* S-R

Fonte: CTISM

O diagrama temporal das saídas $\bar{Q}$ deste *flip-flop*, em função das entradas S, R e *clock*, fica sendo:

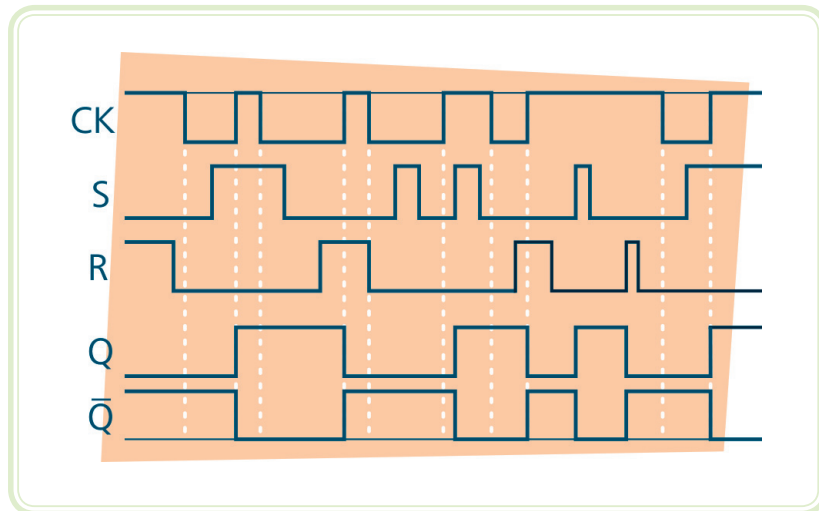


Figura 5.14: Diagrama temporal das saídas Q e $\bar{Q}$ em função das entradas

Fonte: CTISM

Observe que as saídas são sincronizadas com o *clock*, ou seja, mesmo que a entrada S passe de zero a um, as saídas Q e $\bar{Q}$ somente sofrerão variação no próximo pulso de *clock*.

Observe na Figura 5.15 o comportamento quando o *flip-flop* tiver entradas *Preset* e *Clear*.

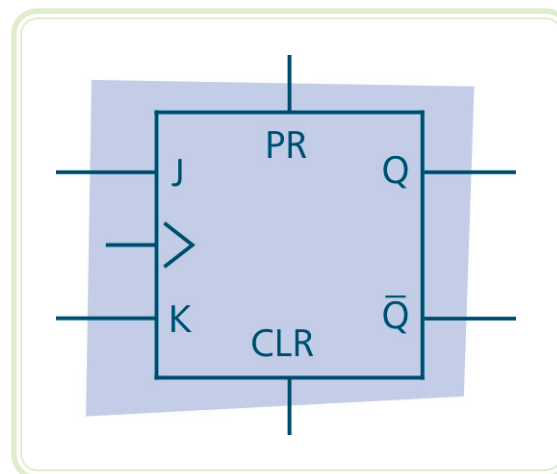


Figura 5.15: Flip-flop JK com entradas preset e clear

Fonte: CTISM

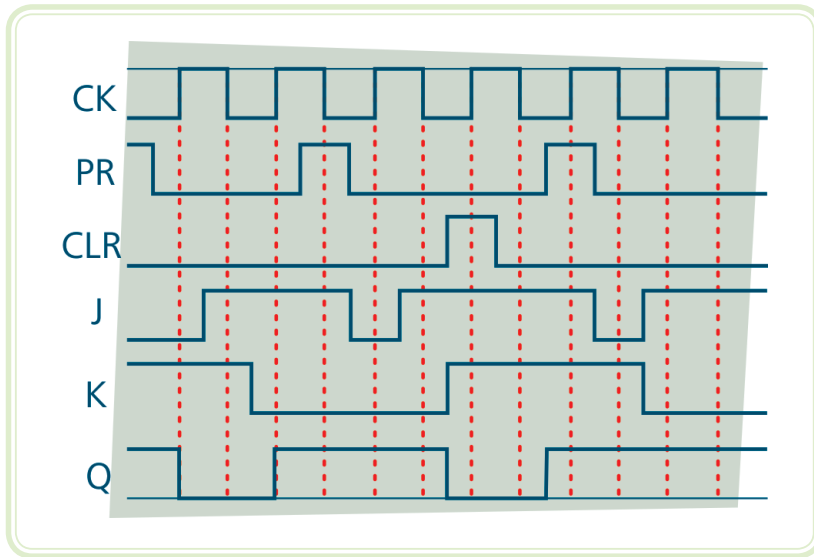


Figura 5.16: Diagrama temporal da saída Q em função das entradas

Fonte: CTISM

Observe que as saídas são sincronizadas com o *clock*, ou seja, mesmo que a entrada S passe de zero a um, a saída Q somente irá variar no próximo pulso de *clock*. No entanto, o “*preset*” e o “*clear*” não obedecem ao *clock*, ou seja, seus efeitos são sentidos na saída Q assim que estes sinais aparecem, e eles se sobrepõem às entradas J e K.

Podemos ter ainda *flip-flops* com entradas invertidas, conforme a Figura 5.17 seguinte:

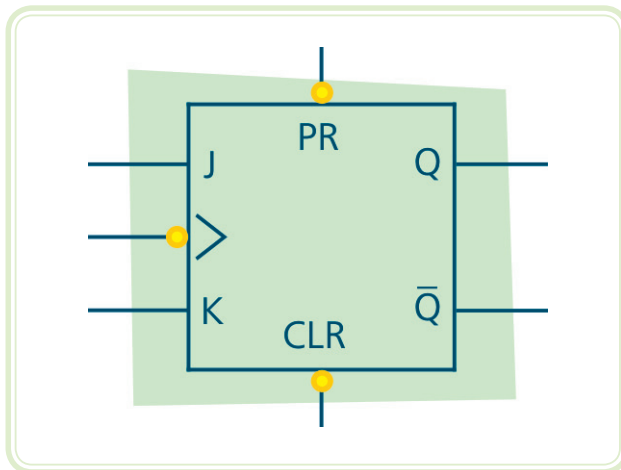


Figura 5.17: Flip-flop JK com entradas preset, clear e clock invertidas

Fonte: CTISM

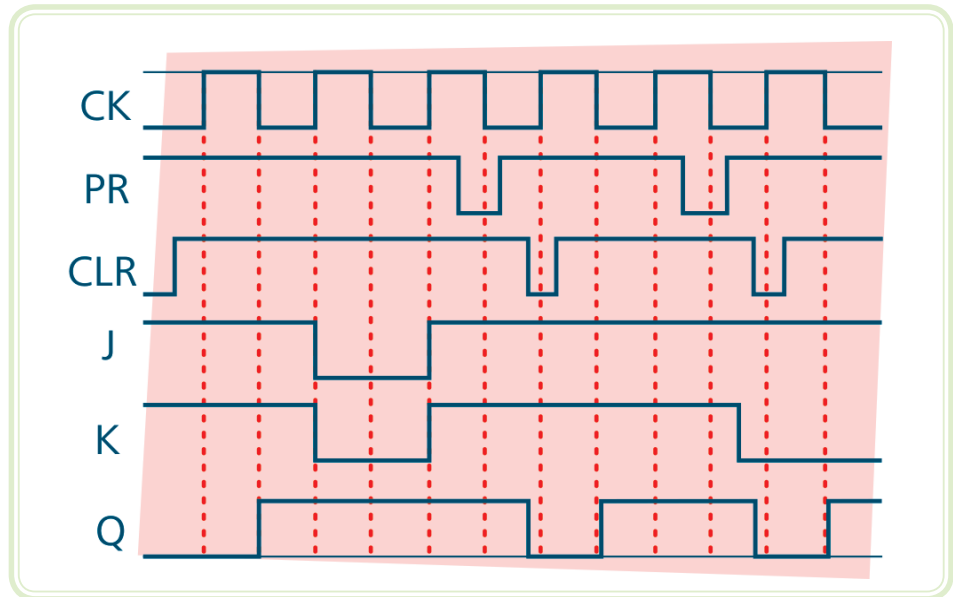


Figura 5.18: Diagrama temporal da saída Q em função das entradas

Fonte: CTISM

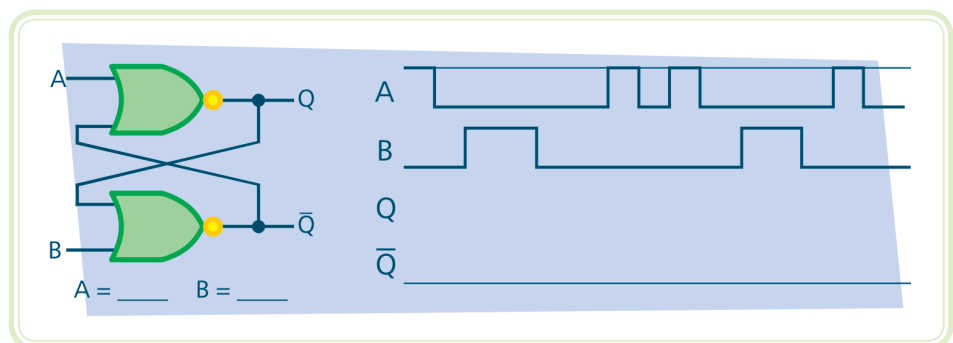
Resumo

Nessa aula aprendemos como construir e analisar o funcionamento de *flip-flops* com portas *NAND* e *NOR*. Conhecemos os diferentes tipos de *flip-flops*, bem como aprendemos a verificar seu comportamento ao longo do tempo. Dessa forma, encerramos esta disciplina, e estamos aptos a passar para o estudo de circuitos mais complexos, como microcontroladores e dispositivos lógico-programáveis.



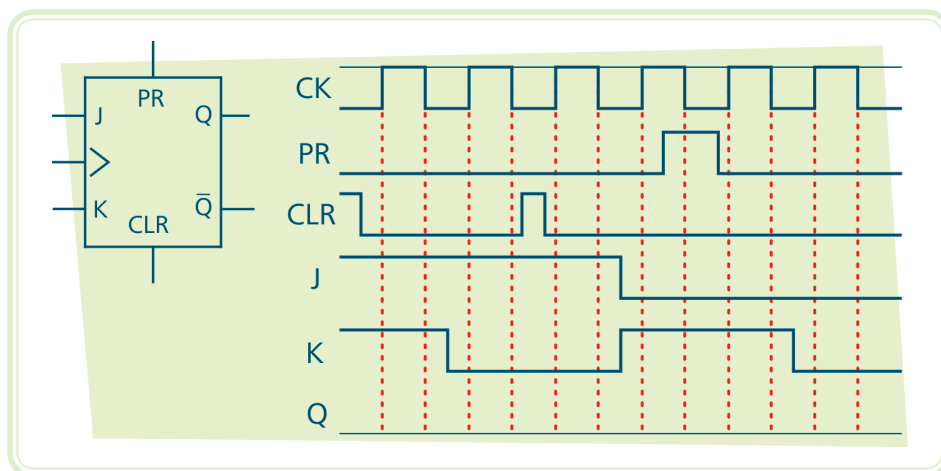
Atividades de aprendizagem

1. Para o *flip-flop* RS, identifique as entradas R e S e desenhe as formas de onda nas saídas em função dos sinais aplicados.

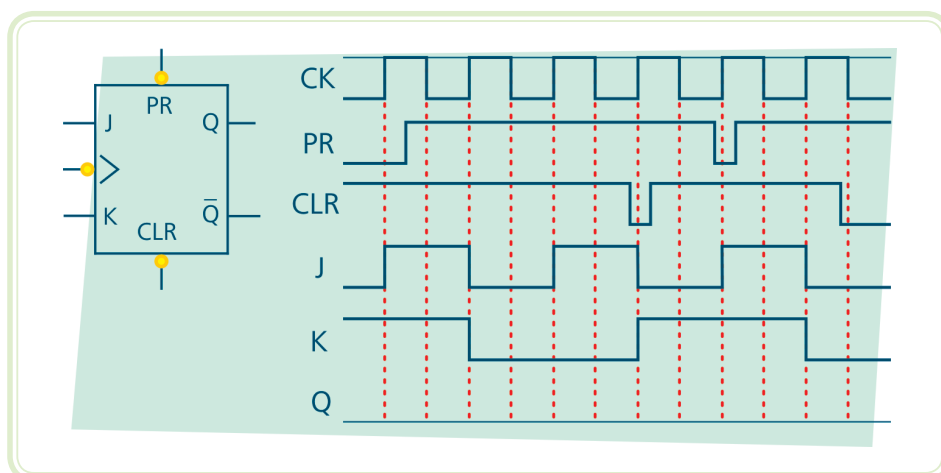


Fonte: CTISM

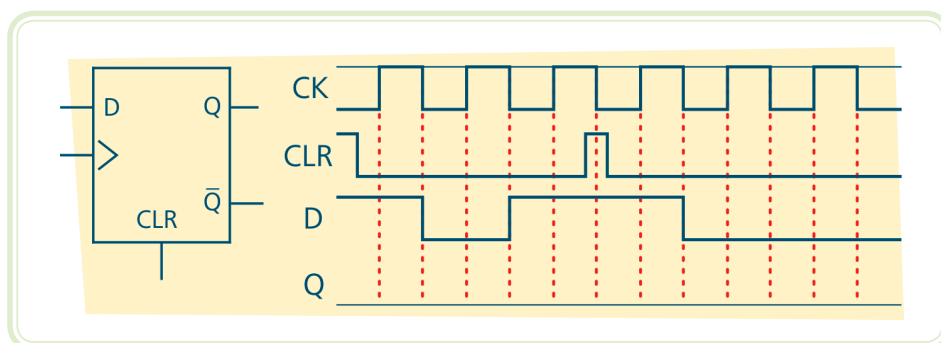
2. Para as *flip-flops* das figuras a seguir, desenhe a forma de onda na saída em função dos sinais aplicados:



Fonte: CTISM



Fonte: CTISM



Fonte: CTISM

Referências

GARUE, S. **Eletrônica Digital**: circuitos e tecnologias. São Paulo: Hemus, 2004.

IDOETA, I.; CAPUANO, F. **Elementos de Eletrônica Digital**. 34. ed. São Paulo: Erica, 2002.

PADILLA, Antônio J. G. **Sistemas Digitais**. Lisboa: McGraw-Hill de Portugal, 1993.

PATTERSON, D. A; HENNESSY, J. L. **Computer Organization & Design**: the hardware/software interface. New York: Morgan Kaufmann Publishers, Inc., 1998.

STALLINGS, William. **Arquitetura e Organização de Computadores**. 5. ed. São Paulo: Makron Books, 2002.

TANENBAUM, Andrew S. **Organização Estruturada de Computadores**. 5. ed. Rio de Janeiro: Pearson Prentice Hall, 2007.

TAUB, H. **Circuitos Digitais e Microprocessadores**. São Paulo: McGraw-Hill Ltda, 1984.

TOCCI, R. J. **Sistemas Digitais**: princípios e aplicações. 10. ed. São Paulo: Pearson Prentice Hall, 2007.

Currículo do professor-autor

Saul Azzolin Bonaldo é professor do Colégio Técnico Industrial (CTISM) da Universidade Federal de Santa Maria (UFSM). Graduado em Engenharia Elétrica e mestre em Engenharia Elétrica pela UFSM. Trabalhou por vários anos na iniciativa privada, especialmente no projeto e execução de instalações elétricas em baixa tensão, redes lógicas e sistemas de segurança eletrônica, adquirindo boa experiência em gestão empresarial e no acompanhamento e execução de obras. Foi Inspetor do CREA-RS. No CTISM, ministra as disciplinas de Eletrônica, Circuitos Digitais, Máquinas Elétricas e Manutenção Eletromecânica. Atua também como Coordenador do Curso Técnico em Automação Industrial. É Membro do IEEE (*The Institute of Electrical and Electronics Enginners*) e filiado ao IAS (*Industry Application Society*), ao PELS (*Power Electronics Society*) e ao IES (*Industrial Electronics Society*). É revisor da revista *Potentials*, publicada pelo IEEE, e da *Industrial Electronic Magazine*, publicada pelo IES-IEEE.



