Instituto Federal Catarinense

Câmpus Luzerna

Engenharia de controle e automação

apostila de sistemas digitais

rafael garlet de oliveira

Luzerna – SC

2013

Sumário

[Sumário 2](#_Toc361927749)

[Prefácio 6](#_Toc361927750)

[1. Introdução 7](#_Toc361927751)

[1.1 Sistemas de Numeração 8](#_Toc361927752)

[1.2 Representação das Quantidades Binárias 8](#_Toc361927753)

[1.3 Circuitos Lógicos e Digitais 9](#_Toc361927754)

[1.4 Transmissão Série e Paralela 9](#_Toc361927755)

[1.5 Memórias e Computadores Digitais. 10](#_Toc361927756)

[2. Códigos e Sistemas de Numeração 12](#_Toc361927757)

[2.1 Bases Numéricas 12](#_Toc361927758)

[2.1.1 Sistema Decimal 12](#_Toc361927759)

[2.1.2 Sistema Binário 13](#_Toc361927760)

[2.1.3 Sistema Octal e Hexadecimal 13](#_Toc361927761)

[2.2 Conversão entre Bases; 14](#_Toc361927762)

[2.2.1 Método das Divisões Sucessivas 14](#_Toc361927763)

[2.2.2 Conversão de Decimal para uma Base Qualquer 15](#_Toc361927764)

[2.2.3 Conversão de uma Base Qualquer para Decimal 15](#_Toc361927765)

[2.2.4 Conversão entre Binário, Octal e Hexadecimal 16](#_Toc361927766)

[2.2.5 Conversão de Número Fracionário 16](#_Toc361927767)

[2.3 Representação de números com sinal 18](#_Toc361927768)

[2.3.1 Módulo e Sinal (MS) 18](#_Toc361927769)

[2.3.2 Complemento 1 19](#_Toc361927770)

[2.3.3 Complemento 2 19](#_Toc361927771)

[2.4 Codificação de Números Binários 21](#_Toc361927772)

[2.4.1 Código BCD 21](#_Toc361927773)

[2.4.2 Código Gray 21](#_Toc361927774)

[2.4.3 Código 7 Segmentos 22](#_Toc361927775)

[3. Álgebra Booleana e Portas Lógicas 23](#_Toc361927776)

[3.1 Tabela Verdade e Funções Lógicas 23](#_Toc361927777)

[3.1.1 Função Não (NOT) – Inversão 23](#_Toc361927778)

[3.1.2 Função E (AND) 23](#_Toc361927779)

[3.1.3 Função Ou (OR) 24](#_Toc361927780)

[3.1.4 Função Não-E (NAND) 24](#_Toc361927781)

[3.1.5 Função Não-Ou (NOR) 24](#_Toc361927782)

[3.1.6 Função Ou Exclusivo (X-OR) 24](#_Toc361927783)

[3.1.7 Função Não Ou Exclusivo (X-NOR) 25](#_Toc361927784)

[3.2 Teoremas Booleanos 25](#_Toc361927785)

[3.2.1 Teoremas de uma variável: 25](#_Toc361927786)

[3.2.2 Teoremas de mais variáveis: 25](#_Toc361927787)

[3.3 Dualidade 27](#_Toc361927788)

[3.4 Formas Padrão Para as Expressões Lógicas 27](#_Toc361927789)

[3.4.1 Forma Padrão de Mintermos 27](#_Toc361927790)

[3.4.2 Forma Padrão de Maxtermos 28](#_Toc361927791)

[3.5 Simplificação de Funções Booleanas 29](#_Toc361927792)

[3.6 Minimização por Mapa de Karnaugh 29](#_Toc361927793)

[4. Circuitos Lógicos Combinacionais 33](#_Toc361927794)

[4.1 Circuitos SSI 33](#_Toc361927795)

[4.1.1 Análise 33](#_Toc361927796)

[4.1.2 Síntese 34](#_Toc361927797)

[4.1.3 Gerador de Paridade 36](#_Toc361927798)

[*4.1.4* Circuitos *Enable/Disable* 37](#_Toc361927799)

[4.1.5 Circuito Somador 38](#_Toc361927800)

[4.2 Circuitos MSI 40](#_Toc361927801)

[4.2.1 Circuito Somador 40](#_Toc361927802)

[4.2.2 Decodificador 40](#_Toc361927803)

[4.2.3 Codificador 45](#_Toc361927804)

[4.2.4 Multiplexador 45](#_Toc361927805)

[4.2.5 Demultiplexador 47](#_Toc361927806)

[4.2.6 Comparador de Magnitudes 48](#_Toc361927807)

[5. Circuitos Lógicos Sequenciais 50](#_Toc361927808)

[*5.1* *Flip-flops* 50](#_Toc361927809)

[5.1.1 Flip-flop RS (*Reset-Set*) 51](#_Toc361927810)

[5.1.2 Flip-flop D (Data) 56](#_Toc361927811)

[5.1.3 Flip-flop T (*Toggle*) 57](#_Toc361927812)

[5.1.4 Flip-flop JK 58](#_Toc361927813)

[5.1.5 Conversão entre *flip-flops* 59](#_Toc361927814)

[5.1.6 Parâmetros Operacionais 59](#_Toc361927815)

[5.2 Diagramas de Estado 60](#_Toc361927816)

[5.2.1 Estrutura Básica 60](#_Toc361927817)

[5.2.2 Exemplo: Somador Serial 61](#_Toc361927818)

[5.2.3 Tabela de Estados 63](#_Toc361927819)

[5.2.4 Diagrama de estados dos *flip-flops* 63](#_Toc361927820)

[5.3 Análise de circuitos sequenciais síncronos 64](#_Toc361927821)

[5.3.1 Objetivo da análise 64](#_Toc361927822)

[5.3.2 Exemplo 1 65](#_Toc361927823)

[5.3.3 Exemplo 2 67](#_Toc361927824)

[5.4 Projeto de Circuitos Sequenciais Síncronos 69](#_Toc361927825)

[5.4.1 Exemplo 1 70](#_Toc361927826)

[5.4.2 Exemplo 2 72](#_Toc361927827)

[5.5 Principais circuitos sequenciais síncronos 74](#_Toc361927828)

[5.5.1 Contadores 74](#_Toc361927829)

[5.5.2 Registradores de deslocamento 79](#_Toc361927830)

[6. Dispositivos de Memória 83](#_Toc361927831)

[6.1 Introdução 83](#_Toc361927832)

[6.2 Conceitos básicos e terminologia 83](#_Toc361927833)

[6.3 Estrutura básica de um computador 84](#_Toc361927834)

[6.4 Sistemas de memória em um computador 86](#_Toc361927835)

[6.4.1 Capacidade de uma memória 87](#_Toc361927836)

[6.4.2 Conceitos sobre dispositivos de memória 88](#_Toc361927837)

[6.4.3 Princípio de operação de uma memória 89](#_Toc361927838)

[6.5 Classificação das memórias semicondutoras 92](#_Toc361927839)

[6.5.1 Memórias de leitura - ROM 92](#_Toc361927840)

[6.5.2 Memórias de Acesso Randômico (RAM) 100](#_Toc361927841)

[6.5.3 Bancos de memória 103](#_Toc361927842)

[7. Circuitos Schmitt-Trigger e Multivibradores 105](#_Toc361927843)

[7.1 Circuitos *Schmitt-Trigger* 105](#_Toc361927844)

[7.2 Circuitos Multivibradores 107](#_Toc361927845)

[7.2.1 Multivibrador Monoestável 107](#_Toc361927846)

[7.2.2 Multivibrador Astável 110](#_Toc361927847)

[8. Famílias Lógicas de Circuitos Integrados 112](#_Toc361927848)

[8.1 Parâmetros de Circuitos Integrados 112](#_Toc361927849)

[8.2 A família TTL 115](#_Toc361927850)

[8.2.1 Classificação das Séries TTL 116](#_Toc361927851)

[8.3 *Fan-out* em dispositivos TTL 119](#_Toc361927852)

[8.4 As famílias MOS 119](#_Toc361927853)

[8.4.1 O Transistor MOSFET 119](#_Toc361927854)

[8.4.2 Vantagens e desvantagens dos MOSFETs sobre a tecnologia bipolar 120](#_Toc361927855)

[8.5 A família CMOS 120](#_Toc361927856)

[8.5.1 Classificação das séries CMOS 120](#_Toc361927857)

[8.6 Compatibilidade entre TTL e CMOS 121](#_Toc361927858)

# Prefácio

Esta apostila serve como material para a disciplina de Sistemas Digitais, que se encontra no currículo do curso de Engenharia de Controle e Automação do IFC - câmpus Luzerna. Por ser uma cadeira da segunda fase, subentende-se que os conceitos relacionados à Informática Para Automação, onde foi vista lógica de programação e linguagens de alto nível, já sejam de conhecimento do aluno.

Nesta disciplina, a interface entre o mundo real, que é predominantemente analógico, e os sistemas computacionais será introduzida. Os microcomputadores, em seu nível de abstração mais baixo, somente compreendem linguagem de máquina, que é formada por sistemas de numeração Binário ou Hexadecimal, que são a saída dos compiladores das linguagens de usuário, como C, Delphi, etc.

Conceitos que envolvem sistemas de numeração serão devidamente detalhados nesta apostila, onde será também apresentada a Lógica Booleana, componentes de circuitos lógicos digitais e os elementos que constituem a arquitetura dos sistemas microprocessados.

Espera-se que com esta disciplina o aluno conheça os princípios de sistemas eletrônicos digitais, domine o processo de análise e projeto de circuitos eletrônicos digitais, compreenda dispositivos lógicos combinacionais e sequenciais e conheça as várias famílias lógicas, seu desempenho e interfaces.

Esta disciplina serve como pré-requisito para a cadeira de microcontroladores, que vai retomar e utilizar alguns conceitos vistos sobre elementos de circuitos digitais e arquitetura de *hardware*.

**Objetivos da Disciplina**

* Compreender os princípios de sistemas numéricos e bases;
* Compreender os princípios de álgebra e lógica booleana;
* Conhecer e identificar os principais componentes de circuitos lógicos digitais;
* Conhecer as técnicas de análise e síntese de circuitos lógicos digitais;
* Conhecer as principais famílias lógicas;
* Conhecer os conceitos de memórias semicondutoras e arquitetura de computadores.

# Introdução

Na natureza as grandezas são predominantemente analógicas, cuja variação é dada de forma contínua. Logo, a forma mais comum de representar valores numéricos é a **analógica**. Nesta representação, entre dois valores há uma quantidade infinita de valores aceitáveis e geralmente são utilizados indicadores variáveis. Como exemplos conhecidos podem-se citar os termômetros de mercúrio, velocímetro de automóvel, relógio com ponteiros.

A forma de representar valores numéricos utilizada pelos sistemas digitais é a representação **digital**. Nesta representação os números variam de forma discreta, ou seja, passo a passo. Entre dois valores, há uma quantidade finita de valores aceitáveis. Em um relógio digital, por exemplo, a variação dos valores é dada de um em um segundo.

Para que um sistema digital seja capaz de trabalhar com sinais analógicos é necessária a utilização de conversores entre os sinais analógicos e digitais, como representa a Figura 1.

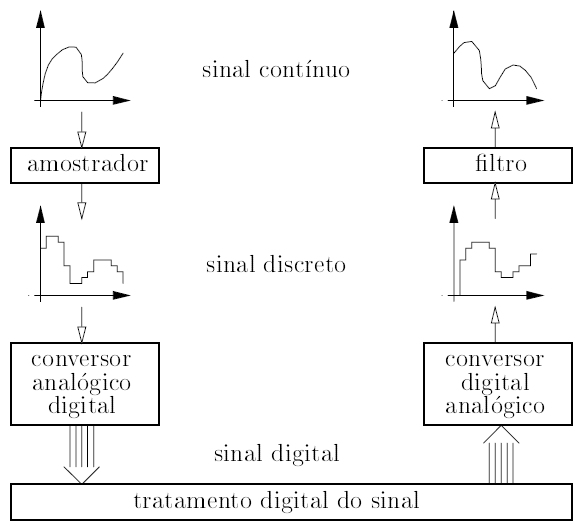


Figura : Conversão entre sinais analógicos e digitais.

Um dígito pode ser entendido como um sinal gráfico que representa um número, mas é também um termo derivado do Latim que se refere aos dedos do corpo humano. Desde sempre o ser humano utiliza os dedos nos processos de contagem, o que deu origem ao sistema de numeração **decimal**, formado por 10 números.

Os sistemas digitais fazem uso do sistema de numeração binário, que é formado por 2 números: 0 e 1. Estes sistemas consideram somente dois valores para a codificação das informações que utilizam: *ligado/desligado, verdadeiro/falso*. A grande vantagem da representação binária está na facilidade de implementação de circuitos eletrônicos para armazenar e realizar operações sobre informações binárias.

Vantagens:

* Sistemas digitais são mais fáceis de projetar;
* Fácil armazenamento da informação;
* Maior precisão e exatidão;
* As operações podem ser programadas;
* Circuitos digitais são menos afetados por ruídos;
* Circuitos digitais são mais adequados para integração.

## Sistemas de Numeração

Os sistemas numéricos mais utilizados pela eletrônica digital são: decimal, binário, octal e hexadecimal. Seus nomes se originam da quantidade de símbolos que utilizam para representar os valores, como apresentado na Tabela 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decimal: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  |  |  |  |  |  |
| Binário: | 0 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Octal: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  |  |  |  |  |  |  |  |
| Hexadecimal: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |

Tabela : Símbolos utilizados pelas bases numéricas.

O sistema decimal é o mais comumente utilizado pelas pessoas. É empregado tanto na entrada de um sistema, por meio do teclado, sensores, etc., quanto na sua saída, por meio de *displays*, atuadores, etc.

O sistema binário é utilizado internamente a um sistema digital, por sua memória e sistema de processamento. Os sistemas octal e hexadecimal são utilizados para fornecer um meio mais eficiente para representar números binários que contêm grandes quantidades de bits, devido à fácil conversão entre estas bases.

## Representação das Quantidades Binárias

Quantidades binárias podem ser representadas por qualquer dispositivo que apresente apenas dois estados de operação ou condições possíveis (ligado/desligado, aberto/fechado). Pode-se definir que uma chave aberta represente o dígito 0 e uma chave fechada, o dígito 1. Assim, no exemplo da Figura 2 o número que está representado é 100102.

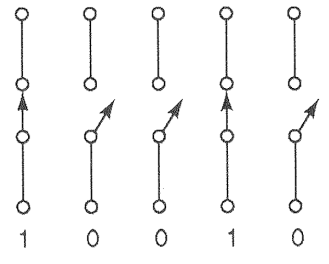


Figura : Conjunto de chaves representando um número binário.

Em sistemas digitais eletrônicos, a informação binária é tipicamente representada por faixas de tensões. Normalmente, a faixa de 0 V a 0,8 V representa o binário 0 e a faixa de 2 V a 5 V, o binário 1, como mostra a Figura 3.

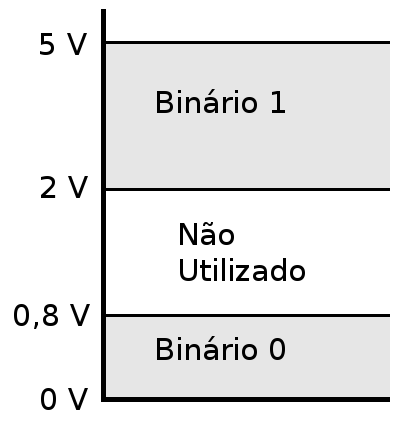


Figura : Tensões típicas em um sistema digital.

## Circuitos Lógicos e Digitais

O objetivo de circuitos digitais é responder a tensões de entrada que variem entre a faixa de valores determinada para os níveis 0 e 1, da forma como é apresentado na Figura 3. O modo como um circuito digital responde a uma entrada é a lógica do circuito. Cada tipo de circuito digital obedece a um determinado conjunto de regras lógicas, por isso os circuitos digitais são também chamados de circuitos lógicos.

Atualmente os circuitos digitais são normalmente implementados usando Circuitos Integrados (CIs). Existem diversas tecnologias de fabricação para produzir os CIs digitais, as mais comuns são o CMOS (*Complementary Metal-Oxide Semiconductor*) e o TTL (*Transistor/Transistor Logic*). A tecnologia CMOS emprega transistores de efeito de campo em sua tecnologia, e a família TTL emprega transistores bipolares. As diferenças básicas entre estas famílias se referem ao tamanho dos componentes, velocidade de resposta, consumo de corrente, tensão utilizada, etc.

## Transmissão Série e Paralela

A comunicação entre diferentes circuitos digitais é feita através da transmissão de dados. Existem duas formas para se transmitirem as informações digitais na forma de dados binários: transmissão serial e paralela. Nas duas formas, os dados transmitidos são representados na forma de tensões na saída de um circuito emissor.

Na transmissão serial, somente um bit é transmitido de cada vez, representado na forma de tensões que variam periodicamente em intervalos de tempo regulares. Utiliza somente uma linha de conexão para todos os bits. Na transmissão paralela, um determinado número de bits é transmitido simultaneamente. Utiliza uma linha de conexão por bit. A Figura 4 ilustra a transmissão do número 101102 nestas duas formas.

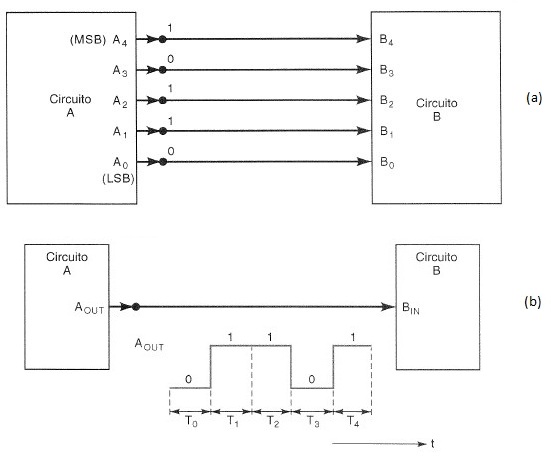


Figura : (a) Transmissão paralela. (b) Transmissão Serial.

## Memórias e Computadores Digitais.

Um computador é um sistema de *hardware* que realiza operações aritméticas, manipula dados e toma decisões. Geralmente é projetado para executar conjuntos de tarefas definidas. Para isto, deve ser dado um conjunto de instruções que lhe diga exatamente o que fazer. Estes conjuntos de instruções são denominados **programas**, que são armazenados na unidade de memória e codificados em forma binária.

**Partes principais de um computador:**

A Figura 5 mostra as cinco unidades principais de um computador digital, juntamente com as suas interações. As principais funções de cada unidade são:

1. **Unidade de Entrada**

Permite que um conjunto de instruções (programa) e dados seja fornecido ao sistema.

1. **Unidade de Memória**

Armazena as instruções e dados recebidos da unidade de entrada e os resultados das operações aritméticas. Fornece informações à unidade de saída.

1. **Unidade de Controle**

Busca as instruções na unidade de memória, uma de cada vez, e as interpreta, gerando sinais apropriados para as outras unidades, de acordo com cada instrução.

1. **Unidade Lógica/Aritmética**

Realiza os cálculos aritméticos e toma as decisões lógicas, enviando os resultados para a unidade de memória.

1. **Unidade de Saída**

Recebe os dados da unidade de memória e os exibe ao operador, ou ainda atua em outros dispositivos.

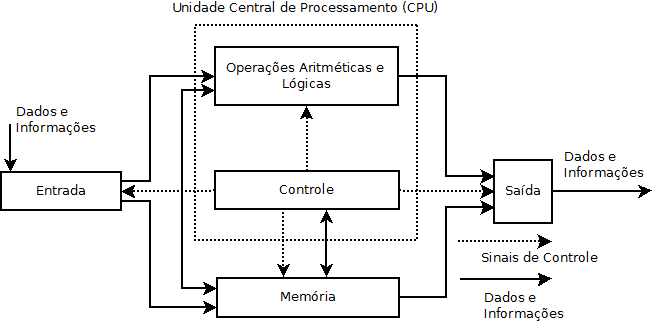


Figura : Diagrama funcional de um computador digital

**Memória:**

Nos circuitos digitais sem memória, quando um sinal é aplicado, a saída muda em resposta à entrada, mas quando o sinal de entrada é removido, a saída volta ao estado original.

Entretanto existem dispositivos que possuem a propriedade de retenção de resposta a uma entrada momentânea, que é denominada de **memória**. Nestes dispositivos, a saída muda de estado conforme a entrada aplicada, mas mesmo que o sinal de entrada seja removido, a saída se mantém no seu estado anterior.

**Unidade Central de Processamento (CPU):**

A unidade de controle, e a unidade de operações aritméticas e lógicas formam a Unidade Central de Processamento (CPU – *Central Processing Unit*). Esta unidade é responsável por executar a busca na memória por instruções e realizar a sua interpretação, assim como controlar e realizar as várias operações requeridas pelas instruções.

# Códigos e Sistemas de Numeração

## Bases Numéricas

O sistema de numeração decimal é normalmente utilizado externamente a um sistema digital, tanto em sua entrada, por usuários ou sensores, quanto em sua saída por meio de *displays* ou atuadores. Contudo, internamente a um sistema digital, o sistema binário de numeração é o mais importante. Com isso, em muitos casos deseja-se que o sistema digital seja capaz de realizar a conversão entre estas duas bases, como ilustra a Figura 6.



Figura : Necessidade de conversão entre as bases binária e decimal.

Os sistemas numéricos octal e hexadecimal são também utilizados para representar internamente quantidades binárias, devido à facilidade de conversão entre estas bases. Com isso podem ser representados números binários de uma maneira mais eficiente.

Outra maneira de representação é por meio de códigos, que não são realmente bases numéricas, mas servem para representar as quantidades binárias. Sua vantagem é a facilidade de conversão entre as bases binária e decimal.

### Sistema Decimal

Este é o sistema de numeração comumente utilizado pelo operador humano, formado pelos símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. O incremento de uma unidade em um dígito faz o número avançar ao dígito seguinte. Quando o maior dígito (9) é incrementado, o dígito a esquerda deste deve ser incrementado, como exemplificado abaixo:

02 + 1 = 03

19 + 1 = 20

099 + 1 = 100

100 + 1 = 101

O dígito mais a esquerda de todos é o dígito mais significativo (MSD: *More Significative Digit*), enquanto que o mais a direita é o dígito menos significativo (Less Significative Digit). Como exemplo, considere o número 517, o dígito 5 é o MSD e o 7 é o LSD. Qualquer número na base 10 pode ser decomposto como uma soma de potências da base, da seguinte maneira:

517 = 5×100 + 1×10 + 7

517 = 5×102 + 1×101 + 7×100

### Sistema Binário

O sistema binário emprega somente dois símbolos para representar qualquer quantidade numérica: 0 e 1. As operações de incremento funcionam da mesma forma que para o sistema decimal, como exemplificado abaixo:

00 + 1 = 01

01 + 1 = 10

10 + 1 = 11

11 + 1 = 100

Cada dígito na base 2 é chamado de *bit* e o conjunto de 8 *bits* é conhecido como *byte.* O dígito mais a esquerda é denominado de bit mais significativo (MSB: *More Significative Bit*), o dígito mais a direita é o bit menos significativo (LSB: *Less Significative Bit)*. Considere o número 1012 (101 na base 2). Este número pode ser decomposto em soma de potências da base:

1012 = 1×22 + 0×21 + 1×20 = 510

### Sistema Octal e Hexadecimal

O sistema octal emprega 8 símbolos para representar as quantidades numéricas: 0, 1, 2, 3, 4, 5, 6 e 7; enquanto que o sistema hexadecimal emprega 16 símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Devido à facilidade de conversão entre estes sistemas numéricos e o sistema binário (que é feita de forma direta), estas bases são utilizadas para representar quantidades binárias de uma forma mais compacta e legível.

A conversão entre números nestes sistemas e o sistema binário é bastante simples. Apesar da possibilidade de ser feita por meio do método das divisões sucessivas, apresentado na próxima seção, pode ser empregada uma forma bem mais simples. Para o sistema octal, basta agrupar os dígitos em grupos de 3, como no exemplo:

1010011002 = 101 001 1002 = 5148

5

1

4

Para o sistema hexadecimal, basta agrupar os dígitos em grupos de 4, como no exemplo:

0101100011012 = 0101 1000 11012 = 58D16

5

8

D

A conversão entre octal e hexadecimal também pode ser realizada por meio do método das divisões sucessivas, mas em alguns casos pode ser preferível converter o número para a base binária e depois converter para a base desejada.

A relação entre os números nestas quatro bases é dada pela Tabela 2:

|  |  |  |  |
| --- | --- | --- | --- |
| Base 10 | Base 2 | Base 8 | Base 16 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 10 | 2 | 2 |
| 3 | 11 | 3 | 3 |
| 4 | 100 | 4 | 4 |
| 5 | 101 | 5 | 5 |
| 6 | 110 | 6 | 6 |
| 7 | 111 | 7 | 7 |
| 8 | 1000 | 10 | 8 |
| 9 | 1001 | 11 | 9 |
| 10 | 1010 | 12 | A |
| 11 | 1011 | 13 | B |
| 12 | 1100 | 14 | C |
| 13 | 1101 | 15 | D |
| 14 | 1110 | 16 | E |
| 15 | 1111 | 17 | F |

Tabela : relação entre os símbolos das principais bases numéricas.

## Conversão entre Bases;

### Método das Divisões Sucessivas

Para fazer a conversão entre quaisquer bases numéricas, pode-se utilizar o Método das Divisões Sucessivas. Considere um número inteiro *N* expresso em sua base *a*. Deseja-se converter este número para uma determinada base *b*. Neste método, utilizam-se operações de divisão de números inteiros tomando-se os restos *ri* das divisões para formar os resultados.



Resultado: ***(N)a=(ri ... r2 r1 r0)b***

Os restos *ri* são os dígitos do número convertido na base *b*. As operações de divisão seguem até o momento em que o dividendo *Ni* é menor do que o divisor *b*, resultando no quociente *Ni+1=0* e resto *r1=Ni*.

Este método é geralmente utilizado quando se deseja converter um número expresso na **base decimal para uma base qualquer**. A conversão de um número expresso em uma **base qualquer para a base decimal** utiliza outro método, como será visto na sequência. A conversão entre números expressos na base **binária, octal e hexadecimal** utiliza um método mais simples, como apresentado no decorrer desta seção.

### Conversão de Decimal para uma Base Qualquer

Para converter um número expresso na base decimal para uma base qualquer, utiliza-se o Método das Divisões Sucessivas. Divide-se o número pela base tantas vezes quanto for necessário e os restos das divisões são utilizados para a obtenção do número desejado. Para ilustrar este conceito, apresentam-se a seguir dois exemplos:

* **Exemplo 1: Converter (51)10 para base binária**



Resultado: (51)10 = (110011)2

As divisões continuam até o ponto em que o quociente é menor do que a base desejada. Para se obter o número convertido, deve-se tomar os restos das divisões, considerando o resto da última operação como o bit mais significativo.

* **Exemplo 2: Converter (538)10 para base hexadecimal**



Resultado: (538)10 = (21A)16

### Conversão de uma Base Qualquer para Decimal

Para converter um número de uma base qualquer para a Base 10, basta aplicar a Lei de Formação, levando em consideração a base com que se está trabalhando. Para deixar mais claro, seguem os exemplos:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (10011)2 = | 1 x 24 + | 0 x 23 + | 0 x 22 + | 1 x 21 + | 1 x 20 |
| = | 16 + | 0 + | 0 + | 2 + | 1 |
| = | (19)10 |  |  |  |  |
|  |  |  |  |  |  |
| (1321B)16 = | 1 x 164 + | 3 x 163 + | 2 x 162 + | 1 x 161 + | 11 x 160 |
| = | 65536 + | 12288 + | 512 + | 16 + | 11 |
| = | (78363)10 |  |  |  |  |

### Conversão entre Binário, Octal e Hexadecimal

A conversão entre as bases binária, octal e hexadecimal é dada de forma direta, como apresentado na seção 2.1.3. Para a conversão de binário para octal, deve-se tomar os dígitos em grupos de 3 e fazer a relação com o correspondente em octal. Para a conversão entre binário e hexadecimal, deve-se tomar os dígitos em grupos de 4 e fazer a relação com o correspondente em hexadecimal. A conversão entre octal e hexadecimal pode ser realizada por meio do método das divisões sucessivas, mas na maioria dos casos é preferível converter primeiro para binário e depois para a base desejada. Abaixo seguem dois exemplos de conversão entre estes sistemas numéricos:

F7A16 =1111 0111 10102

F

7

A

7658 = 111 110 1012

7

6

5

### Conversão de Número Fracionário

Para fazer a conversão de um número fracionário, deve-se separar o número em duas partes, sua parte inteira e sua parte fracionária: N = Ni + Nf. Da mesma forma, o número N pode ser decomposto em somas de potência da base:

N = Ni + Nf

Ni = An.bn + ... + A1.b1 + A0.b0

Nf = A-1.b-1 + A-2.b-2 + ...

A conversão de um número fracionário de uma base qualquer **para a base decimal** pode ser realizada por meio da Lei de Formação, como visto anteriormente. Para exemplificar, considere o número N = 11,012.

N = 11,10102 = Ni + Nf; Ni = 112; Nf = 0,10102

Ni = 112 = 1x21 + 1x20= 310

Nf = 0,10102 = 1x2-1 + 0x2-2 + 1x2-3 + 0x2-4 =  0,62510

N = 11,10102 = 3,62510

A conversão de um número fracionário expresso na base decimal **para uma base qualquer** considera também as duas partes separadas (inteira e racional). Nesta seção somente será vista a técnica para conversão da parte fracionária, pois a conversão da parte inteira foi vista nas seções anteriores. A conversão da parte fracionária é feita da seguinte forma.

Considerando a expressão:

Nr = A-1.b-1 + A-2.b-2 + ...

Deve-se multiplicar ambos os lados da equação pela base b para encontrar o coeficiente A-1:

b.Nr = A-1 + A-2.b-1 + ...

Assim, o coeficiente A-1 é compreendido como a parte inteira do termo b.Nr, ou seja, se este termo for > 1, A-1 = 1, se este termo for < 1, A-1 = 0.

Esta equação deve então ser reorganizada da seguinte forma:

(b.Nr - A-1)= + A-2.b-1 + ...

E multiplicada novamente pela base b, para encontrar o coeficiente A-2, da mesma forma que foi feito para A-1:

b.(b.Nr - A-1)= + A-2 + ...

E assim por diante, até atingir o número de dígitos desejados.

**Exemplo: C**onsidere o número 3,67110, converta para a base binária com quatro casas depois da vírgula.

N = 3,67110

Ni = 310 = 112

Nr = 0,67110



Nr = A1A2A3A4...= 10102 (Até o número desejado de casas decimais.)

Assim, N = 3,67110 = 11,10102

## Representação de números com sinal

Utilizando um número qualquer d de bits, é possível representar até 2d valores distintos. Por exemplo, um registrador de 4 bits é capaz de representar 16 valores distintos. Se considerar somente valores positivos, é possível representar números inteiros positivos de 0 a 15, como mostrado na Figura 7.

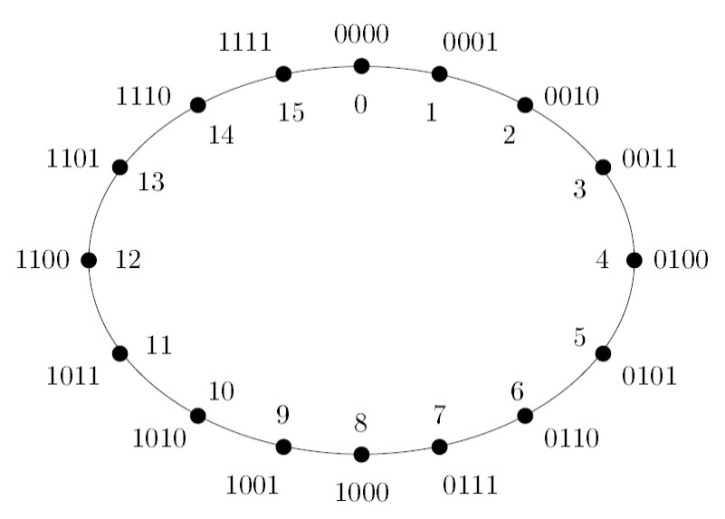


Figura : Representação de números inteiros positivos.

Para a representação de números negativos, podem ser empregadas as técnicas vistas nas próximas seções.

### Módulo e Sinal (MS)

Com esta abordagem, utiliza-se o bit mais significativo (MSB) para indicar o sinal, como apresentado na Figura 8. O bit 0 indica que o número é positivo, enquanto que o bit 1 indica que o número é negativo.

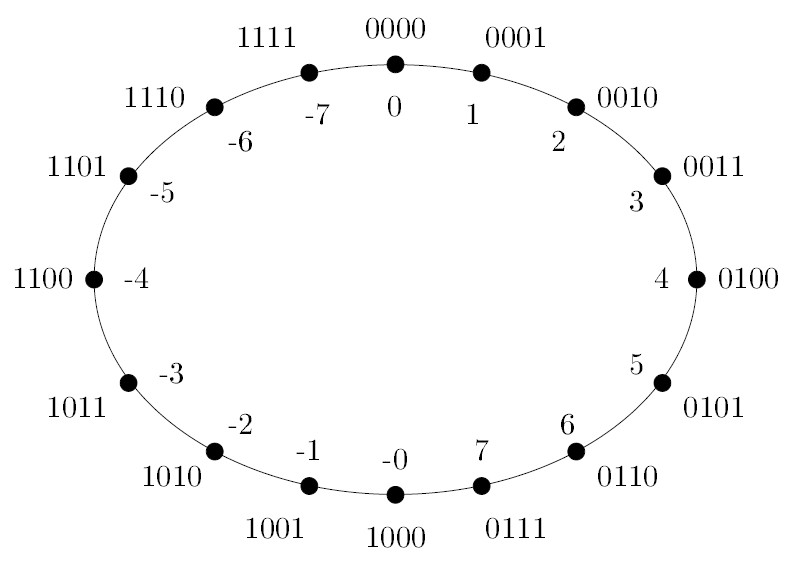


Figura : Representação de números com sinal utilizando MS.

Como exemplo, o número 7 é representado por 0111 e o número -7 é representado por 1111. A desvantagem desta abordagem é a dupla representação do número 0 (0000 e 1000). Com isso a faixa de representação é -7  7. Como alternativa podem ser utilizados os números complementares.

### Complemento 1

Existem dois tipos de complementos: o complemento 1 e o complemento 2. Os complementos servem para facilitar as operações aritméticas do ponto de vista do processador e auxiliar a representação de números com sinal.

O complemento 1 de um número N é definido da seguinte maneira:

, onde

 complemento 1;

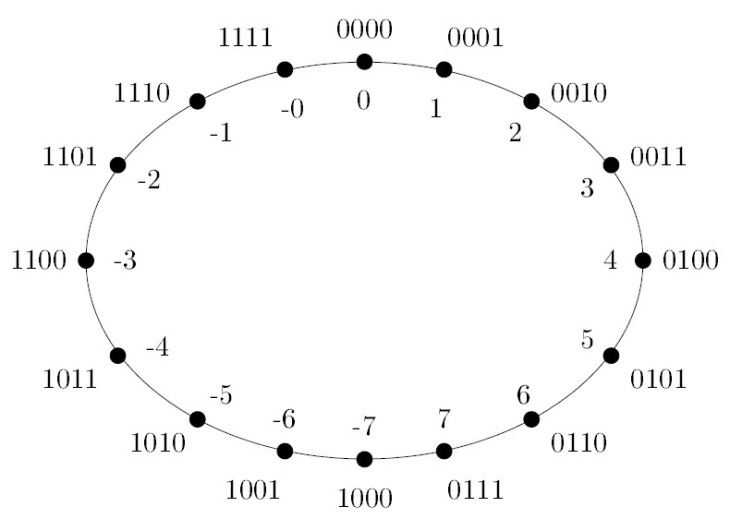
K: módulo ou número de dígitos.

**Exemplo:** N = 01112 = 710



Uma maneira mais simples de encontrar o complemento 1 é inverter todos os bits do número. Ou seja, trocar todos os 0 por 1 e trocar todos os 1 por 0.

Por meio do complemento 1, os números negativos são representados pelo complemento do número positivo correspondente. Com o complemento 1, o bit MSB também indica o sinal do número. Mas, com o complemento 1 o número 0 tem também duas representações possíveis, e a faixa de representação é -7  7.



### Complemento 2

O método mais utilizado para representar números negativos é com o complemento 2. O complemento 2 de um número N é definido da seguinte maneira:

, onde

: complemento 2;

K: módulo ou número de dígitos.

**Exemplo:** N = 01112 = 710



Nota-se que ,

Assim, basta encontrar o complemento 1 através da inversão de bits e somar o dígito 1, para encontrar o complemento 2.

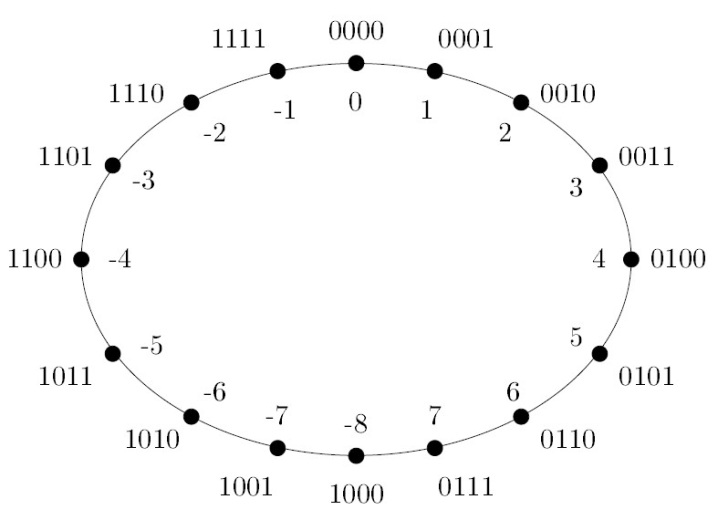
**Exemplo:** 

Outra maneira de encontrar o complemento 2 é a seguinte: começando da direita para esquerda os números vão sendo copiados até que seja encontrado o primeiro número 1. Depois de copiar este número, os números seguintes devem ser invertidos.

**Exemplo:** N = 1101002



Com este método, os números negativos são representados pelo complemento 2 do número positivo em questão. O bit MSB indica o sinal do número. Por meio desta abordagem, o número zero tem somente uma representação e a faixa de representação é -8  7.



## Codificação de Números Binários

Outra forma de representação de números binários é por meio de códigos, que não são realmente bases numéricas, mas servem para representar as quantidades binárias.

### Código BCD

A sigla deste código significa: decimal codificado em binário (*binary-coded-decimal*). Sua vantagem é a facilidade de conversão entre as bases decimal e binária. Com este código, cada dígito de um número decimal é representado separadamente em seu equivalente em binário, como no seguinte exemplo:

91510 = 1001 0001 0101 (BCD)

5

1

9

### Código Gray

A vantagem deste código é minimizar o risco de perdas devido à variação de muitos bits ao mesmo tempo em um circuito. Com este código, apenas um bit muda entre dois números sucessivos na sequência, como apresentado na Tabela 3:

|  |  |  |  |
| --- | --- | --- | --- |
| Decimal | Binário | BCD | Gray |
| 0 | 0000 | 000 | 0000 |
| 1 | 0001 | 0001 | 0001 |
| 2 | 0010 | 0010 | 0011 |
| 3 | 0011 | 0011 | 0010 |
| 4 | 0100 | 0100 | 0110 |
| 5 | 0101 | 0101 | 0111 |
| 6 | 0110 | 0110 | 0101 |
| 7 | 0111 | 0111 | 0100 |
| 8 | 1000 | 1000 | 1100 |
| 9 | 1001 | 1001 | 1101 |
| 10 | 1010 | 0001 0000 | 1111 |
| 11 | 1011 | 0001 0001 | 1110 |
| 12 | 1100 | 0001 0010 | 1010 |
| 13 | 1101 | 0001 0011 | 1011 |
| 14 | 1110 | 0001 0100 | 1001 |
| 15 | 1111 | 0001 0101 | 1000 |

Tabela : Representação nos Códigos BCD e Gray

### 

### Código 7 Segmentos

Este código serve para facilitar a apresentação de um número em um mostrador de 7 segmentos. Este dispositivo consiste em um display formado por sete segmentos luminosos (Figura 9), que são construídos utilizando várias tecnologias, como LEDs por exemplo. Cada display pode representar qualquer dígito do sistema hexadecimal (0 a F).

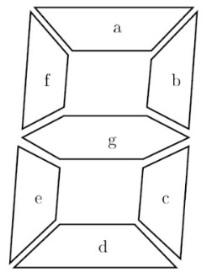


Figura : Display de 7 segmentos

Para um display que considera o sinal ativo alto para ligar um segmento, o código utilizado é apresentado na Tabela 4.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Números em Hexadecimal | | | | | | | | | | | | | | | |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| Segmentos | a | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| b | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| c | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| d | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| e | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| f | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| g | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

Tabela : Código 7 segmentos

# Álgebra Booleana e Portas Lógicas

Quando se estuda lógica de uma maneira geral, consideram-se variáveis que podem assumir apenas dois possíveis valores mutuamente exclusivos: *verdadeiro* ou *falso*. Este tipo de variável é chamado de **variável lógica**. Uma **função lógica** é uma relação entre dois conjuntos de variáveis lógicas, B = f(A), onde A e B são conjuntos de variáveis lógicas.

Exemplo: B = f(A) =  é a função inversora, ou seja, B é o inverso de A.

Os circuitos digitais são desenvolvidos para assumirem apenas dois possíveis valores lógicos, considerando as faixas de tensão, como visto na seção 1.2. A álgebra Booleana é uma ferramenta matemática que permite descrever a relação entre as saídas e as entradas de um circuito lógico por meio de equações (expressões booleanas ou expressões lógicas). As implementações, das funções lógicas fundamentais são denominadas portas lógicas.

## Tabela Verdade e Funções Lógicas

A tabela verdade permite mostrar a forma como as saídas dos circuitos lógicos respondem às combinações dos níveis lógicos nas entradas. A partir das tabelas verdade é possível descobrir ou projetar a lógica de um circuito. Abaixo são apresentadas as funções lógicas fundamentais, com sua tabela verdade, sua notação em circuitos digitais e sua função em álgebra Booleana.

### Função Não (NOT) – Inversão

A função inversão é uma porta lógica que possui somente uma entrada e a sua saída é o complemento da entrada.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | A | B | | 0 | 1 | | 1 | 0 |   Tabela Verdade | Símbolo | Função |

### Função E (AND)

A função *AND* tem a saída em nível lógico 1 somente quando todas as entradas estiverem em nível lógico 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | A | B | C | | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 |   Tabela Verdade | Símbolo | Função |
|  |  |  |

### Função Ou (OR)

A função *OR* tem a saída em nível lógico 1 quando alguma das entradas estiver em nível lógico 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | A | B | C | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 1 |   Tabela Verdade | Símbolo | Função |

### Função Não-E (NAND)

A função *NAND* é o complemento da função *AND*.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | A | B | C | | 0 | 0 | 1 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 |   Tabela Verdade | Símbolo | Função |

### Função Não-Ou (NOR)

A função *NOR* é o complemento da função *OR*.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | A | B | C | | 0 | 0 | 1 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 0 |   Tabela Verdade | Símbolo | Função |

### Função Ou Exclusivo (X-OR)

A função *X-OR* tem a saída em nível lógico 1 quando as entradas estiverem em níveis lógicos diferentes.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | A | B | C | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 |   Tabela Verdade | Símbolo | Função |

### Função Não Ou Exclusivo (X-NOR)

A função *X-NOR* tem a saída em nível lógico 1 quando as entradas estiverem em níveis lógicos iguais.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | A | B | C | | 0 | 0 | 1 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 |   Tabela Verdade | Símbolo | Função |

## Teoremas Booleanos

Existem alguns teoremas envolvendo as operações básicas na álgebra booleana, que podem ajudar a simplificar as expressões lógicas.

### Teoremas de uma variável:

|  |  |  |  |
| --- | --- | --- | --- |
| (1a) |  | (3a) |  |
| (1b) |  | (3b) |  |
| (2a) |  | (4a) | C:\Users\rafa\Dropbox\IFC\Sistemas Digitais - Automacao\Apostila de Sistemas Digitais\teorema 4.jpg |
| (2b) |  | (4b) |  |

### Teoremas de mais variáveis:

|  |  |  |
| --- | --- | --- |
| (9) | Comutatividade: |  |
| (10) | Comutatividade: |  |
| (11) | Associatividade: |  |
| (12) | Associatividade: |  |
| (13a) | Distributividade: |  |
| (13b) | Distributividade: |  |
| (14) |  |  |
| (15) |  |  |
| (16a) | DeMorgan: |  |
| (16b) | DeMorgan: |  |

|  |  |  |
| --- | --- | --- |
|  |  | Usando o teorema 13 |
|  |  | Usando o teorema 1 |
|  |  | Usando o teorema 2 |

**Prova do Teorema 14:**

|  |
| --- |
|  |

**Exemplo de Simplificação:**

**Universalidade das Portas *NAND* e *NOR:***

As figuras Figura 10 e Figura 11 demonstram como as portas lógicas *NAND* e *NOR* podem ser utilizadas para formar as funções lógicas básicas e, assim, serem empregadas para implementar qualquer circuito lógico.

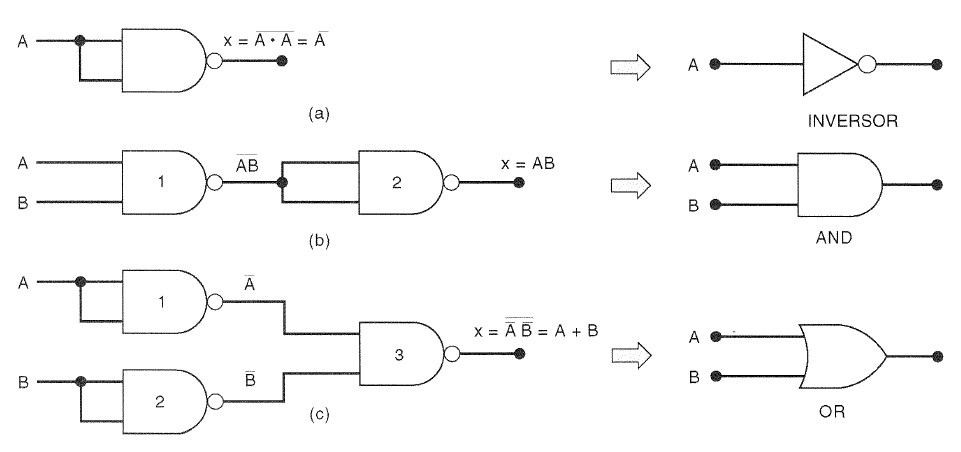


Figura : Universalidade das portas *NAND*

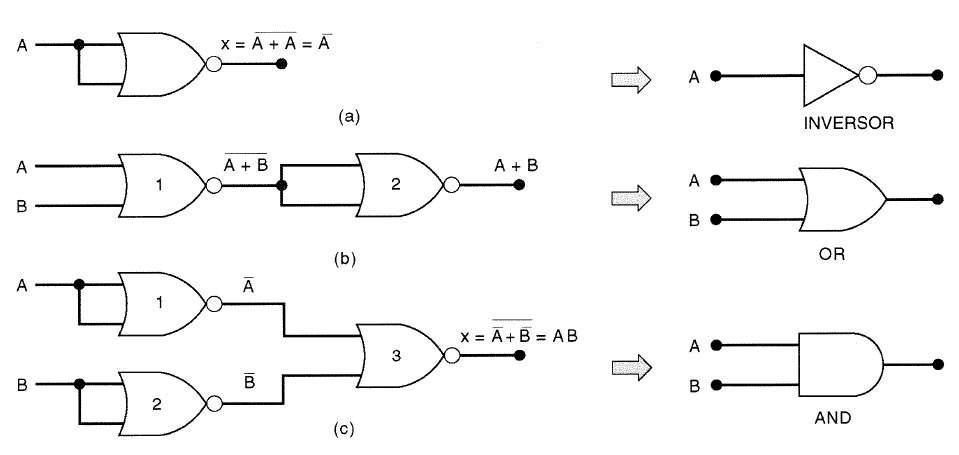


Figura : Universalidade das portas NOR

## Dualidade

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Nota-se que os teoremas apresentados anteriormente são agrupados aos pares. Cada um dos teoremas tem o seu correspondente dual. A dualidade geralmente é utilizada para a simplificação de expressões booleanas. O dual pode ser obtido trocando-se 0 por 1 e 1 por 0, ‘+’ por ‘.’ e ‘.’ por ‘+’. Para se fazer uma simplificação, primeiramente encontra-se o dual e depois de terminada a manipulação algébrica retorna-se a forma anterior encontrando novamente o dual.

**Exemplo:**

## 

## Formas Padrão Para as Expressões Lógicas

Qualquer expressão lógica pode ser escrita em uma das formas padrões que serão vistas nesta seção. Estas formas padrões são constituídas por soma de produtos e produtos de somas. São usadas para auxiliar a análise e simplificação das expressões, utilizando os métodos que serão vistos nas seções seguintes.

É denominada forma canônica de uma função lógica toda soma de produtos (**mintermos**) e todo produto de somas (**maxtermos**) em que aparecem todas as variáveis em cada um dos termos que constituem a expressão.

### Forma Padrão de Mintermos

A expressão é formada por uma soma-padrão de produtos. Cada termo da expressão é chamado de mintermo.

**Exemplo:** Encontrar a forma padrão de mintermos da função .

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | Distributiva |
|  |  |  |

Esta forma apresenta a propriedade de f(A, B, C)=1 se ao menos um dos mintermos for igual a 1, e um mintermos será 1 somente se todas as suas variáveis forem 1. Assim, na tabela verdade desta função, cada linha que corresponder ao nível f(A, B, C)=1, é considerada um mintermo.

|  |  |  |  |
| --- | --- | --- | --- |
| A B C | f(A, B, C) | Mintermos |  |
| 0 0 0 | 0 |  | m0 |
| 0 0 1 | 1 |  | m1 |
| 0 1 0 | 0 |  | m2 |
| 0 1 1 | 0 |  | m3 |
| 1 0 0 | 1 |  | m4 |
| 1 0 1 | 1 |  | m5 |
| 1 1 0 | 1 |  | m6 |
| 1 1 1 | 1 |  | m7 |

### Forma Padrão de Maxtermos

A expressão é formada por um produto-padrão de somas. Cada termo da expressão é chamado de maxtermo.

**Exemplo:** Encontre a forma canônica de maxtermos para a mesma função do exemplo anterior.

|  |  |  |
| --- | --- | --- |
|  |  | Dual |
| Dual |
|  |
|  |
| Distributiva |

Esta forma apresenta a propriedade de f(A, B, C)=0 se ao menos um dos maxtermos for 0, e um maxtermo será 0 somente se todas as suas variáveis forem 0. Assim, na tabela verdade desta função, cada linha que corresponder ao nível f(A, B, C)=0 é considerada um maxtermo.

|  |  |  |  |
| --- | --- | --- | --- |
| A B C | f(A, B, C) | Maxtermos |  |
| 0 0 0 | 0 |  | M0 |
| 0 0 1 | 1 |  | M1 |
| 0 1 0 | 0 |  | M2 |
| 0 1 1 | 0 |  | M3 |
| 1 0 0 | 1 |  | M4 |
| 1 0 1 | 1 |  | M5 |
| 1 1 0 | 1 |  | M6 |
| 1 1 1 | 1 |  | M7 |

## Simplificação de Funções Booleanas

Considerando os teoremas da álgebra booleana apresentados anteriormente e a forma dual das expressões, devem-se simplificar as expressões antes de executar a sua implementação em circuitos lógicos. Ao simplificar uma expressão lógica, é possível minimizar os custos de implementação em circuitos digitais e aumentar a sua velocidade de resposta. Algumas expressões lógicas podem ter uma forma muito mais simples do que a expressão original, como no seguinte exemplo.

|  |  |
| --- | --- |
|  | Utiliza 5 portas |
| Distributiva |
|  |
| Distributiva |
| Apenas uma chave |
|  |  |

## Minimização por Mapa de Karnaugh

O mapa de Karnaugh é uma representação gráfica utilizada para simplificar expressões lógicas ou converter uma tabela verdade em seu circuito lógico correspondente. Por meio dos mapas de Karnaugh é possível simplificar ao máximo uma expressão lógica, o que é chamado de minimização.

É formado a partir da tabela verdade de uma dada expressão e oferece uma representação das formas canônicas de mintermos e maxtermos. Estes termos são representados pelas células do mapa, que são ordenadas segundo o código *gray*, de modo que as células adjacentes difiram entre si de apenas uma variável.

Abaixo está colocado como é a construção do mapa de Karnaugh para duas variáveis com a forma de mintermos.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | A B | X |  | | 0 0 |  | m0 | | 0 1 |  | m1 | | 1 0 |  | m2 | | 1 1 |  | m3 | | X = m0 + m1 + m2 + m3 | |  |  |  |  | | --- | --- | --- | --- | |  | **B** |  |  | | **A** |  | **0** | **1** | |  | **0** | **m0** | **m1** | |  | **1** | **m2** | **m3** | |
| **Tabela Verdade** | **Função** | **Mapa de Karnaugh** |

A minimização é executada pela aplicação do teorema 4b () entre células adjacentes, como será visto no exemplo a seguir.

**Exemplo:** para exemplificar o processo de minimização será apresentada uma tabela verdade para 3 variáveis. A partir da tabela verdade é construído o mapa de Karnaugh, inserindo-se os valores de f(A, B, C). A função é construída a partir do mapa, selecionando-se os mintermos cujo valor correspondente de f(A, B, C) seja 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | A B C | f(A, B, C) | Mintermos |  | | 0 0 0 | 1 |  | m0 | | 0 0 1 | 1 |  | m1 | | 0 1 0 | 1 |  | m2 | | 0 1 1 | 0 |  | m3 | | 1 0 0 | 0 |  | m4 | | 1 0 1 | 0 |  | m5 | | 1 1 0 | 1 |  | m6 | | 1 1 1 | 0 |  | m7 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **BC** |  |  |  |  | | **A** |  | **00** | **01** | **11** | **10** | |  | **0** | **1** | **1** | **0** | **1** | |  | **1** | **0** | **0** | **0** | **1** | |

Podem-se também escolher os termos cujo valor correspondente de f(A, B, C) seja 0, mas desta forma devem-se selecionar os maxtermos, como mostrado abaixo para o mesmo exemplo.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | A B C | f(A, B, C) | Maxtermos |  | | 0 0 0 | 1 |  | M0 | | 0 0 1 | 1 |  | M1 | | 0 1 0 | 1 |  | M2 | | 0 1 1 | 0 |  | M3 | | 1 0 0 | 0 |  | M4 | | 1 0 1 | 0 |  | M5 | | 1 1 0 | 1 |  | M6 | | 1 1 1 | 0 |  | M7 | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **BC** |  |  |  |  | | **A** |  | **00** | **01** | **11** | **10** | |  | **0** | **1** | **1** | **0** | **1** | |  | **1** | **0** | **0** | **0** | **1** | |

**Desafio:** Provar que as duas expressões  e são equivalentes.

**Agrupando as células adjacentes:** Com este exemplo, nota-se que é possível simplificar a expressão lógica aplicando a propriedade distributiva nos mintermos (ou maxtermos) adjacentes (neste exemplo: m0 e m1, m2 e m6). Este processo pode ser feito de maneira direta, economizando a manipulação algébrica. Para isto, devem-se marcar no mapa quais são as células adjacentes e **eliminar a variável que tiver o seu valor normal agrupado com o seu complementar**.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **BC** |  |  |  |  | | **A** |  | **00** | **01** | **11** | **10** | |  | **0** | **1** | **1** | **0** | **1** | |  | **1** | **0** | **0** | **0** | **1** | | No agrupamento da esquerda, elimina-se a variável C, no da direita, a variável A: |

O agrupamento das células pode ser feito de diversas formas, mas deve ser feito sempre aos pares, quartetos ou octetos (múltiplos de 2). Nas figuras Figura 12 e Figura 13 são apresentadas alguns exemplos retirados do livro texto (TOCCI) onde podem ser vistos mapas para 3 e 4 variáveis e algumas formas diferentes de agrupamento.

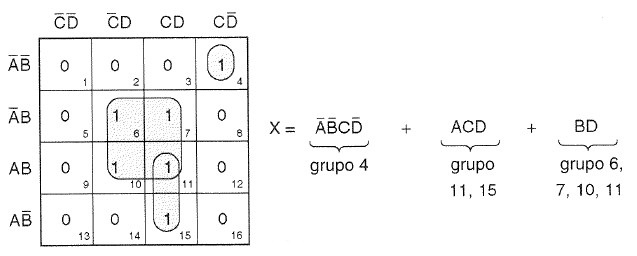
****

Figura : Exemplo de mapas de Karnaugh. Fonte: Tocci (1998, p. 75, 76 e 77).

|  |  |  |
| --- | --- | --- |
| **C:\Users\rafa\Desktop\mapa1.jpg** | **C:\Users\rafa\Desktop\mapa5.jpg** | **C:\Users\rafa\Desktop\mapa.jpg** |
| **C:\Users\rafa\Desktop\mapa3.jpg** | **C:\Users\rafa\Desktop\mapa4.jpg** | **C:\Users\rafa\Desktop\mapa2.jpg** |
| **Figura 13: Exemplos de mapas de Karnaugh. Fonte: Tocci (1998, p. 75, 76 e 77).** | | |

**Funções Incompletamente Especificadas:**

Uma função lógica é dita incompletamente especificada quando para uma dada combinação das variáveis de entrada a saída é irrelevante, podendo assumir 0 ou 1. Isso ocorre quando certas combinações das variáveis de entrada nunca ocorrem; ou certa combinação das entradas não influencia no comportamento global do sistema.

Nestas condições a função pode assumir 0 ou 1 na tabela verdade ou no mapa de Karnaugh, sendo então indicada por um X. Podemos considerar a função nessas condições como 0 ou 1 dependendo da conveniência (por exemplo, para auxiliar os agrupamentos no mapa de Karnaugh). **Exemplo:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **BC** |  |  |  |  | | **A** |  | **00** | **01** | **11** | **10** | |  | **0** | **1** | **1** | **X** | **1** | |  | **1** | **0** | **0** | **X** | **0** | | Neste caso, considera-se o X da linha superior como 1 e o X da linha inferior como 0. |

**Exercício Resolvido:** O exemplo 4.15 do livro texto (TOCCI, 11ª edição), demonstra o processo completo de síntese de um circuito lógico partindo da especificação de um sistema para projeto a partir da expressão lógica e mapa de Karnaugh.

# Circuitos Lógicos Combinacionais

Um circuito lógico é dito combinacional quando suas saídas em um determinado instante *t* são uma função, unicamente, de suas entradas naquele instante, ou seja, suas saídas são **combinações lógicas** de suas entradas naquele instante (como ilustra a Figura 14). Isso significa que qualquer modificação nas entradas será imediatamente considerada pelas saídas.

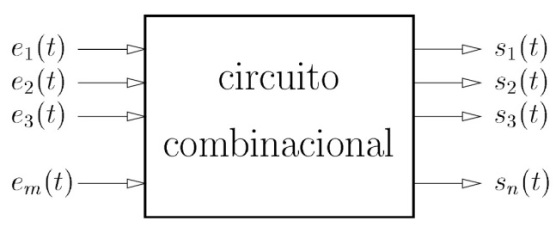


Figura : Modelo de circuitos lógicos combinacionais.

Um circuito lógico pode possuir diversas variáveis de entrada e diversas variáveis de saída. No capítulo anterior, foram vistas as ferramentas para projeto de sistemas digitais utilizando circuitos SSI (portas lógicas básicas) com uma variável de saída. Este capítulo irá abordar o projeto para sistemas com mais de uma variável de saída, assim como a utilização de circuitos lógicos MSI.

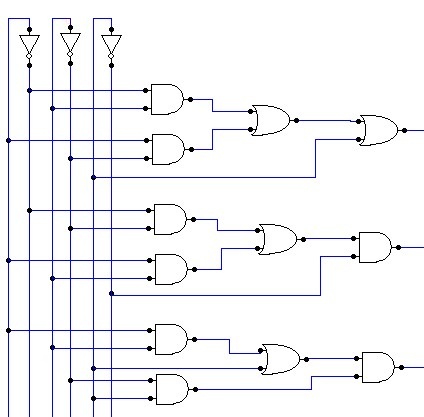
A tecnologia de CIs digitais avançou rapidamente em relação à sua escala de integração. A integração em pequena escala (SSI – *small scale integration*) se refere a CIs que contam com menos de 12 portas por chip. Correspondem às portas lógicas básicas vistas no capítulo anterior. A integração em média escala (MSI – *médium scale integration*) conta com um número de 12 a 99 portas por chip. Corresponde aos chips que serão apresentados no decorrer deste capítulo, como decodificador, codificador, multiplexador, etc. Além destas, existem outras escalas de integração, como larga escala e escala muito ampla (LSI e VLSI) com dezenas de milhares de portas por chip, escala ultra larga (ULSI) com mais de 100 mil portas por chip e escala giga (GSI) com 1 milhão de portas ou mais por chip.

Os chips SSI são geralmente utilizados para funções simples, ou ainda como interface para circuitos mais complexos construídos com outros CIs. Os chips na categoria MSI executam funções um pouco mais complexas e são construídos na forma de blocos básicos com funções como codificação, seleção de dados, conversão de código, comparação, etc.

## Circuitos SSI

### Análise

A **análise** de um circuito lógico construído com chips SSI deve ser feita observando-se os níveis de propagação do sinal digital na estrutura do circuito para construir a expressão lógica (função booleana) de cada saída, como apresentado no seguinte exemplo.



3ºNível de

propagação

2ºNível de

propagação

1ºNível de

propagação









C























B

A

As expressões lógicas resultantes em cada saída são as seguintes:



### Síntese

Para se executar a **síntese** de um circuito lógico combinacional, quando os níveis de saída são conhecidos para todas as possíveis combinações das variáveis de entrada, esses resultados podem ser agrupados em uma tabela verdade. A metodologia de projeto pode-se resumir nos seguintes passos:

1. Montar a tabela verdade a partir da especificação do problema;
2. Traduzir a tabela verdade na forma de um mapa de Karnaugh;
3. Realizar os agrupamentos e a minimização da expressão a partir do mapa de Karnaugh, obtendo a equação simplificada com portas básicas NOT, AND e OR;
4. Observar se é possível modificar a expressão utilizando as funções derivadas XOR e X-NOR, ou, quando necessário, as funções universais NAND e NOR.
5. Montar o circuito lógico correspondente à expressão obtida.

Como exemplo será construído um circuito lógico para detecção de números primos para quatro bits. Primeiramente se constrói a tabela verdade com todas as possibilidades de entrada.

Tabela : Tabela verdade para detecção de números primos com 4 bits.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | A | B | C | D | f(.) |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 |
| 10 | 1 | 0 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 0 |
| 15 | 1 | 1 | 1 | 1 | 0 |

A seguir, traduz-se a tabela verdade na forma de um mapa de Karnaugh e encontra-se a expressão lógica correspondente.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  | **CD** |  |  |  |  | | **AB** |  | **00** | **01** | **11** | **10** | |  | **00** | **0** | **0** | **1** | **1** | |  | **01** | **0** | **1** | **1** | **0** | |  | **11** | **0** | **1** | **0** | **0** | |  | **10** | **0** | **0** | **1** | **0** | | Existem quatro agrupamentos possíveis para a construção da expressão lógica: |

O circuito lógico referente à expressão encontrada é apresentado na Figura 15.

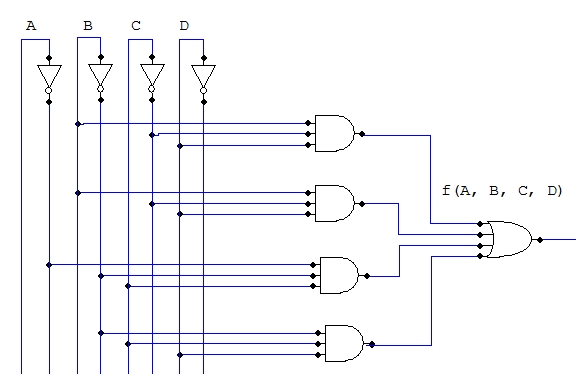


Figura : Circuito lógico para detecção de números primos.

Para se transformar a expressão lógica utilizando-se somente portas NAND, deve barrar a expressão duas vezes e aplicar o teorema de DeMorgan, como mostrado a seguir.



A fim de encontrar uma expressão lógica que utilize somente as portas universais NOR, deve-se escolher os maxtermos do mapa de Karnaugh para a construção da expressão lógica e barrar a expressão duas vezes para a aplicação do teorema de DeMorgan.

### Gerador de Paridade

Como foi visto no capítulo 1, podem existir duas formas de transmissão de dados: série e paralela. Entretanto, qualquer tipo de transmissão está sujeito à perda de dados por diversos motivos. Uma das técnicas desenvolvidas para detectar erros de transmissão é o bit de paridade, que permite verificar se os dados recebidos são fiéis aos dados transmitidos.

Seu funcionamento consiste no envio de um bit de paridade, além dos dados a serem transmitidos, que indica se o número de 1’s contidos nos dados é par ou é ímpar, como ilustra a Figura 16.

D­1

Sistema Digital

Rx

Sistema Digital

Tx

D­n

D­0

Bit de Paridade

Figura : Transmissor e receptor com bit de paridade.

Assim, o emissor calcula o número de 1’s contidos nos dados e, no caso de resultar em um número par, envia o dígito 0 como bit de paridade, mas no caso de resultar em um número ímpar, envia o dígito 1. O emissor, ao receber os dados, recalcula o número de 1‘s contidos e, no caso de ser ímpar, checa se o bit de paridade vale 1, mas no caso de ser par, checa se o bit de paridade vale 0.

A Figura 17 ilustra um circuito gerador de paridade construído com portas XOR no transmissor de dados e a Figura 18 ilustra o equivalente circuito detector de paridade no receptor.

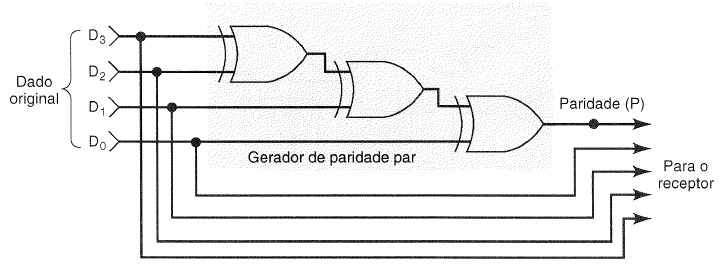


Figura : Circuito gerador de paridade no transmissor.

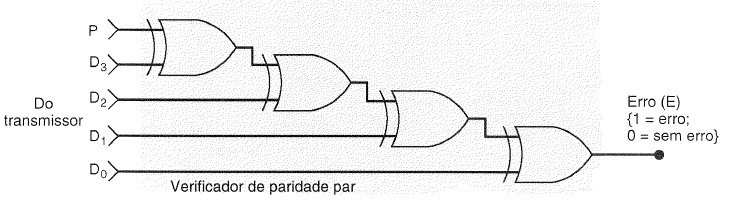


Figura : Circuito verificador de paridade no receptor.

### Circuitos *Enable/Disable*

Uma utilização prática para os teoremas booleanos vistos na seção 3.2 é o controle de passagem de um sinal lógico em um circuito. As portas lógicas básicas podem ser utilizadas para habilitar ou desabilitar o envio ou a recepção de dados.

Na Figura 18, a variável A serve como entrada de dados, enquanto que a variável B serve como entrada de controle que habilita ou desabilita a transmissão dos dados. Para a porta AND, quando B=1 a transmissão de A é habilitada, quando B=0, é desabilitada. O mesmo vale para a porta OR utilizando B=0 para habilitar e B=1 para desabilitar. As portas NAND e NOR funcionam da mesma maneira para habilitar a transmissão, porém o sinal enviado é o de .

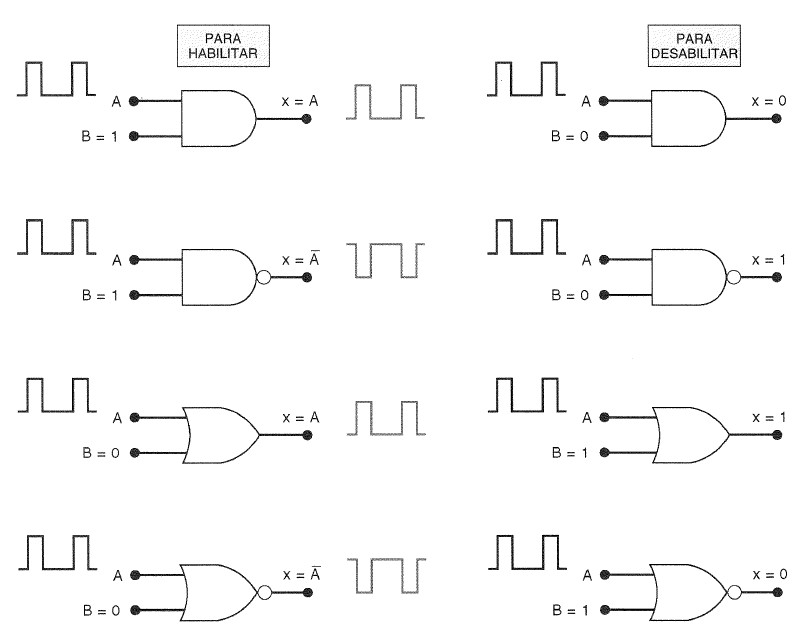
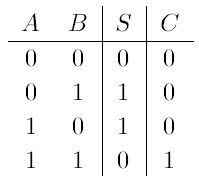


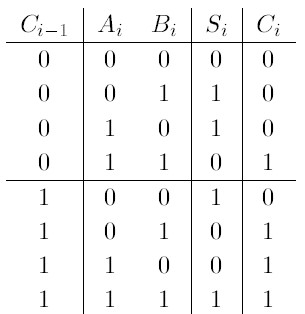
Figura : Portas lógicas básicas utilizadas como circuitos *enable/disable*.

### Circuito Somador

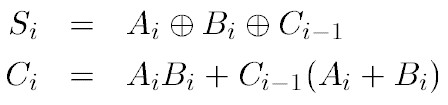
Os circuitos somadores permitem efetuar operações de soma entre duas palavras de n bits, levando em conta o número "vai um" ou *carry*. Para a soma de duas palavras A e B de 1 bit teremos a seguinte tabela verdade para a soma S e o *carry* C.



Pode-se então concluir que e . O circuito definido por essas funções é chamado de **meio somador** e sua implementação é simples, usando apenas duas portas lógicas. Um circuito somador completo deve levar em conta o *carry* de um eventual antecessor para poder ser associado em cascata a outros somadores e assim implementar somadores para palavras mais longas que 1 bit. A tabela-verdade para um **somador completo** de 1 bit é dada por:



Seguindo a metodologia de projeto, constrói-se o mapa de Karnaugh e o circuito lógico equivalente. As expressões lógicas resultantes são as seguintes:



O circuito lógico que implementa estas expressões é apresentado na Figura 20.

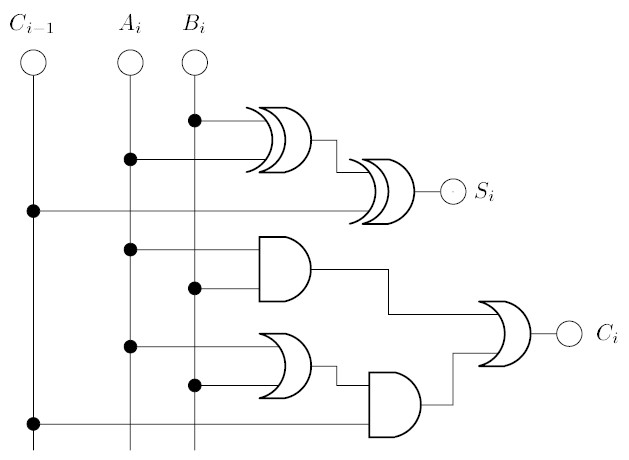


Figura : Circuito de um somador completo de 1 bit.

Utilizando um módulo de somador completo para 1 bit, podem-se construir somadores para palavras de *n* bits. Como exemplo apresenta-se um somador para 4 bits:

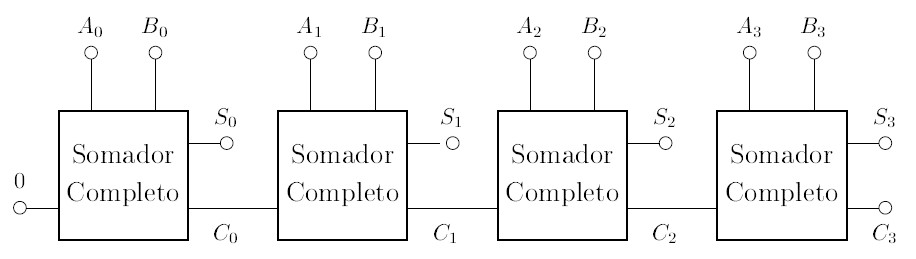


Figura : Somadores completos de 1 bit em cascata.

## Circuitos MSI

### Circuito Somador

Os circuitos somadores, apresentados na seção anterior, são também implementados em circuitos MSI, que reúnem as funções em um só bloco. Na Figura 22 apresenta-se o circuito TTL 7483 que executa a função de somador de quatro bits.

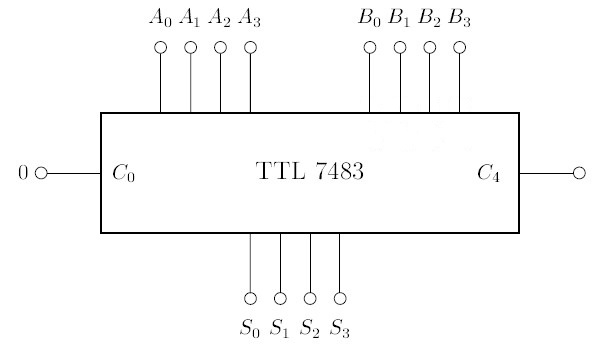


Figura : Circuito somador de 4 bits.

### Decodificador

Um decodificador é um circuito lógico com *n* entradas e 2­­n saídas. Sua função é a conversão de um código de entrada em um código de saída, sendo que cada linha de saída será ativada por uma das possíveis combinações de bits de entradas. Através do código nas suas entradas, podem-se endereçar quaisquer dispositivos conectados nas suas saídas.

Como exemplo toma-se um decodificador de 3 entradas por 8 saídas. Somente uma das saídas será ativada ao mesmo tempo, conforme o código de entrada. Por exemplo, com a entrada A = 0, B = 0 e C = 0, somente a saída Q0 estará em nível lógico 1.

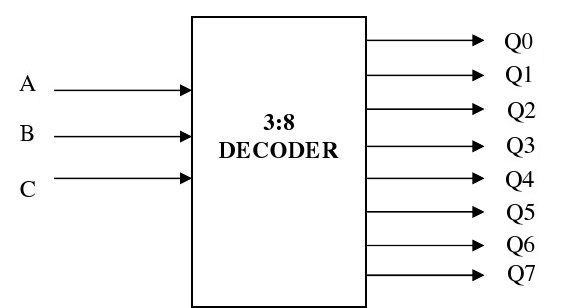


Figura : Decodificador 3:8.

A tabela verdade da saída deste circuito é dada pela Tabela 6, que relaciona as três entradas A, B e C, onde A é o LSB, e as saídas Q0 a Q7.

Tabela : Tabela Verdade para o decodificador 3:8.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | B | A | O0 | O1 | O2 | O3 | O4 | O5 | O6 | O7 |
| 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** |

O circuito lógico que compõe este decodificador é apresentado na Figura 24, onde para cada saída Qi são tomados os mintermos correspondentes da tabela verdade.

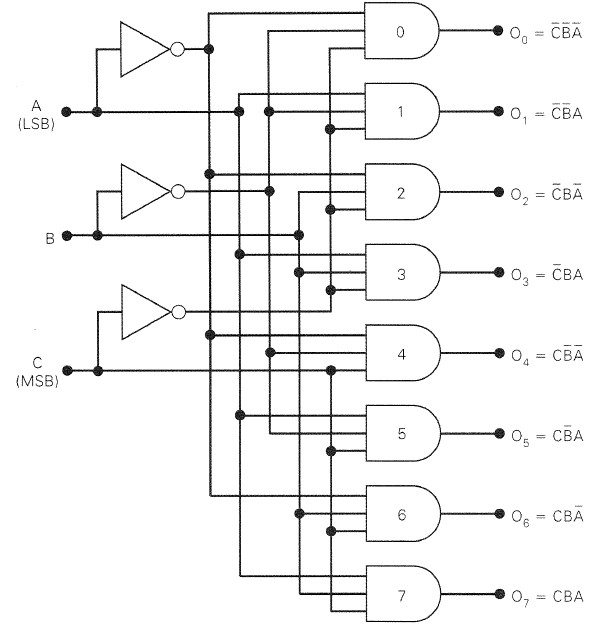


Figura : Circuito lógico do decodificador 3:8.

#### Síntese de circuitos combinacionais com decodificadores

Observa-se que em um decodificador todos os mintermos possíveis são gerados. Com isso se torna possível utilizar este componente para a síntese de circuitos combinacionais. Muitos decodificadores são projetados para gerar saídas ativas em nível baixo (conhecidos como saída em **ativo baixo**), nas quais apenas a selecionada estará em nível baixo, enquanto todas as outras estarão em nível alto. Isto é indicado por pequenos círculos nas saídas do decodificador. Uma das vantagens de projetar circuitos combinacionais com decodificadores é a simplicidade. Para exemplificar, pode-se tomar o mesmo exemplo de um detector de números primos apresentado anteriormente. A tabela verdade deste exemplo é apresentada na Tabela 5.

O circuito lógico para este exemplo é apresentado na Figura 25. É construído utilizando-se o CI 74154, que implementa um decodificador 4:16, com duas entradas *enable* (E0 e E1). Para este circuito, ambas as 16 saídas são ativo baixo, assim como as entradas E0 e E1 (os pequenos círculos indicam que o circuito será ativado quando estas entradas forem iguais a 0).

Assim, cada palavra binária na entrada DCBA que corresponde a um número primo habilita apenas uma das saídas em ativo baixo (saídas 1, 2, 3, 5, 7, 11, e 13). Por exemplo, DCBA = 0011, faz com que a saída 3 seja 0 e todas as outras sejam iguais a 1. Todas as saídas que correspondem a números primos são conectadas à entrada de uma porta NAND, gerando assim a saída Y do detector de números primos. Assim, a tabela verdade da saída Y corresponde à Tabela 5, o que pode também ser provado algebricamente por meio de expressões lógicas.

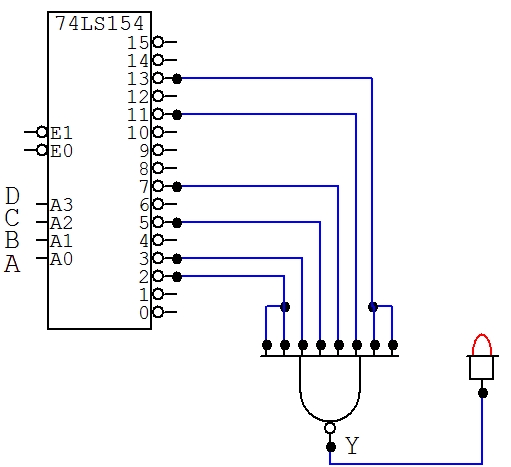


Figura : Implementação de um detector de números primos utilizando decodificador.

Uma saída em nível lógico 0 drena uma corrente bem maior do que uma saída em nível lógico 1. Por isso os decodificadores em ativo baixo (maioria das saídas iguais a 1) geram uma economia de energia se comparados aos decodificadores em ativo alto (maioria das saídas iguais a 0).

#### Decodificador BCD/Decimal

Este circuito faz a conversão entre um número em código BCD para um número em decimal. A Figura 26 apresenta o circuito 74LS42, um decodificador BCD com saída em ativo baixo. As entradas que não representam nenhum número em BCD não habilitam nenhuma saída.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D | C | B | A | Saída em ativo baixo |
| 0 | 0 | 0 | 0 | S0 |
| 0 | 0 | 0 | 1 | S1 |
| 0 | 0 | 1 | 0 | S2 |
| 0 | 0 | 1 | 1 | S3 |
| 0 | 1 | 0 | 0 | S4 |
| 0 | 1 | 0 | 1 | S5 |
| 0 | 1 | 1 | 0 | S6 |
| 0 | 1 | 1 | 1 | S7 |
| 1 | 0 | 0 | 0 | S8 |
| 1 | 0 | 0 | 1 | S9 |
| 1 | 0 | 1 | 0 | Nenhuma |
| 1 | 0 | 1 | 1 | Nenhuma |
| 1 | 1 | 0 | 0 | Nenhuma |
| 1 | 1 | 0 | 1 | Nenhuma |
| 1 | 1 | 1 | 0 | Nenhuma |
| 1 | 1 | 1 | 1 | Nenhuma |

Tabela : Tabela verdade do decodificador BCD/Decimal.

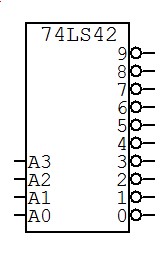


Figura : Decodificador BCD/Decimal

#### Decodificador BCD para 7 Segmentos

Os decodificadores para 7 segmentos são utilizados nos instrumentos digitais que possuem uma saída numérica, como multímetros, multimedidores, etc. Nestes equipamentos é comum encontrar displays de 7 segmentos, como o apresentado na Figura 27. Estes displays são construídos com LEDs, onde as duas estruturas possíveis são conexões em catodo comum e em anodo comum. Para a primeira, a saída do decodificador deve ser em ativo alto, e para a segunda, deve ser em ativo baixo.

Estes decodificadores produzem uma combinação das saídas conforme a combinação dos bits de entrada em BCD, ou seja, uma entrada em BCD gera as saídas que acionam os bits correspondentes conforme o código para 7 segmentos visto na Seção 0.

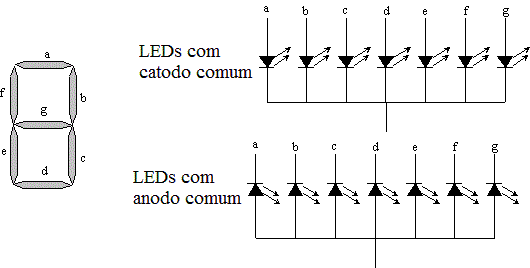


Figura : Display de 7 segmentos.

Na **Figura 28** é apresentado o circuito do decodificador BCD/7 segmentos 7447, que possui as saídas em ativo baixo. A tabela verdade para este CI é apresentada na Tabela 8.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | A3 | A2 | A1 | A0 | a | b | c | d | e | f | g | Display | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 2 | | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 3 | | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 4 | | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 5 | | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 6 | | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 7 | | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 8 | | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 9 | | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |  | | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |  | | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |  | | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |  | | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |  | | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |   Tabela : Tabela verdade para o decodificador BCD/7segmentos 7447. | C:\Users\rafa\Desktop\bcd 7 segmentos.jpg  **Figura 28: Decodificador BCD/7 segmentos.** |

### Codificador

O codificador possui um conjunto de 2n linhas de entrada e um conjunto de n linhas de saída. Somente uma das entradas pode estar ativa ao mesmo tempo. Conforme a entrada que estiver ativa, este dispositivo irá produzir um código nos bits de saída.

Um exemplo é o codificador 74148, que converte uma entrada octal em binário, possui 8 bits de entrada e 3 bits de saída, ambos em ativo baixo. Este codificador implementa uma lógica para evitar conflito quando mais de uma entrada estiver ativa ao mesmo tempo (prioridade das entradas). Nestes casos, a saída corresponderá à entrada mais significativa. Por exemplo, se as entradas I1 e I2 estiverem ativas ao mesmo tempo (nível baixo), a saída será 010, ao invés de 001. A entrada EI corresponde ao sinal de *enable*. A tabela verdade deste dispositivo é representada na Tabela 9.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C:\Users\rafa\Desktop\codificador.jpg  **Figura 29: Codificador octal para binário.** | |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | I0 | I1 | I2 | I3 | I4 | I5 | I6 | I7 | A2 | A1 | A0 | | x | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | x | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | | x | x | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | | x | x | x | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | | x | x | x | x | 0 | 1 | 1 | 1 | 0 | 1 | 1 | | x | x | x | x | x | 0 | 1 | 1 | 0 | 1 | 0 | | x | x | x | x | x | x | 0 | 1 | 0 | 0 | 1 | | x | x | x | x | x | x | x | 0 | 0 | 0 | 0 |   Tabela : Tabela verdade do codificador octal para binário com saída ativo baixo. |

### Multiplexador

Um multiplexador é um dispositivo com 2n entradas de dados, uma saída e n entradas de seleção que escolhem qual das entradas será transferida para a saída. Um multiplexador de 4 entradas é apresentado na Figura 30, onde Di são as entradas de dados, Ei são as entradas de seleção e S é a saída.

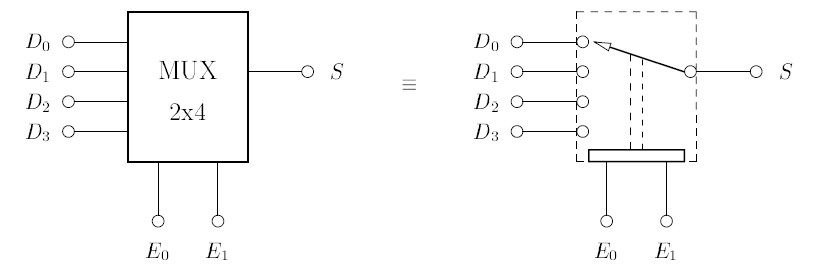


Figura : Multiplexador de 4 entradas.

A tabela verdade deste multiplexador é apresentada na Tabela 10 e seu circuito lógico pode ser implementado como mostra a Figura 31.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | E1 | E0 | D0 | D1 | D2 | D3 | S | | 0 | 0 | 1 | 0 | 0 | 0 |  | | 0 | 1 | 0 | 1 | 0 | 0 |  | | 1 | 0 | 0 | 0 | 1 | 0 |  | | 1 | 1 | 0 | 0 | 0 | 1 |  |   Tabela : Tabela verdade do multiplexador de 4 entradas. | C:\Users\rafa\Desktop\mux 4 entradas circuito.jpg  Figura : Circuito lógico do multiplexador de 4 entradas. |

#### Aplicações dos Multiplexadores

Entre as diversas aplicações dos multiplexadores podem ser citadas seleção de dados, roteamento de dados, sequenciamento de operações, conversões paralelo/série, implementação de tabelas verdade de funções lógicas. O roteamento de dados consiste em escolher quais das entradas estarão ativas na saída, assim como em um relógio digital, que possui várias funções como alarme, cronômetro, entre outras, que devem ser representadas na mesma saída.

Um exemplo de síntese de um circuito utilizando multiplexador é o detector de números primos apresentado na Figura 32. As entradas Di recebem o sinal 1 quando i for um número primo e 0 quando não for. As entradas seletoras Ei recebem o número em código binário. A saída irá apresentar o valor da entrada Di correspondente, conforme o valor das entradas Ei.

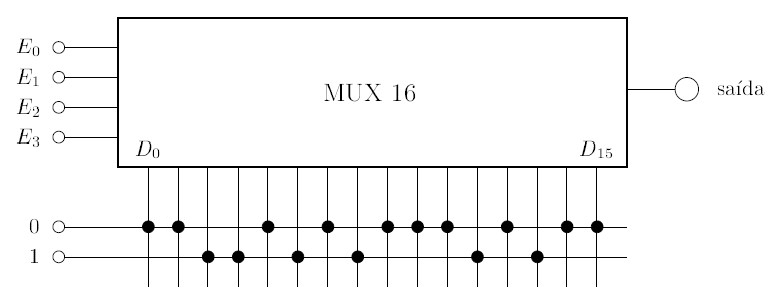


Figura : Detector de números primos com multiplexador.

Outra solução para o mesmo problema pode ser construída utilizando um multiplexador mais simples. Observando a tabela verdade da saída, pode-se constatar que as entradas E2, E1, e E0 têm as mesmas combinações quando E3 vale 0 ou 1 (metade superior e inferior da tabela). Assim, pode-se definir uma saída Y2 que dependa também de E3, como mostrado na Tabela 11 e Figura 33.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| E3 | E2 | E1 | E0 | Y | Y2 |
| 0 | 0 | 0 | 0 | 0 | Gnd |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |  |
| 0 | 0 | 1 | 1 | 1 | Vcc |
| 0 | 1 | 0 | 0 | 0 | Gnd |
| 0 | 1 | 0 | 1 | 1 | Vcc |
| 0 | 1 | 1 | 0 | 0 | Gnd |
| 0 | 1 | 1 | 1 | 1 |  |
| 1 | 0 | 0 | 0 | 0 | Gnd |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 |  |
| 1 | 0 | 1 | 1 | 1 | Vcc |
| 1 | 1 | 0 | 0 | 0 | Gnd |
| 1 | 1 | 0 | 1 | 1 | Vcc |
| 1 | 1 | 1 | 0 | 0 | Gnd |
| 1 | 1 | 1 | 1 | 0 |  |

Tabela : Tabela verdade do detector de números primos

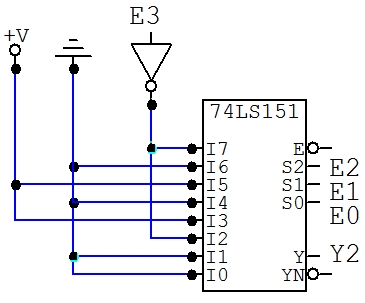


Figura : Detector de números primos.

### Demultiplexador

O demultiplexador realiza a operação inversa do multiplexador, ou seja, uma única entrada de dados (serial) é distribuída (endereçada) entre uma das várias saídas. As entradas de dados serão transmitidas para uma das saídas, conforme os níveis das entradas seletoras.A Figura 34 representa um demultiplexador de 4 vias, onde Si são as saídas e Ei são as entradas seletoras.

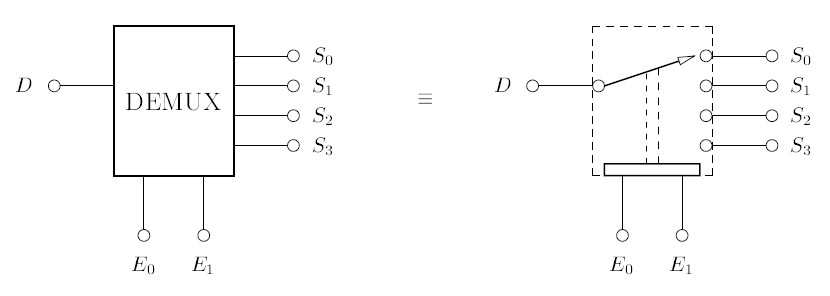


Figura : Demultiplexador de 4 vias.

Para este demultiplexador temos a seguinte tabela verdade representada na Tabela 12 e o circuito lógico representado na Figura 35.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| E1 | E0 | S0 | S1 | S2 | S3 |
| 0 | 0 | D | 0 | 0 | 0 |
| 0 | 1 | 0 | D | 0 | 0 |
| 1 | 0 | 0 | 0 | D | 0 |
| 1 | 1 | 0 | 0 | 0 | D |

Tabela : Tabela verdade do demultiplexador de 4 vias

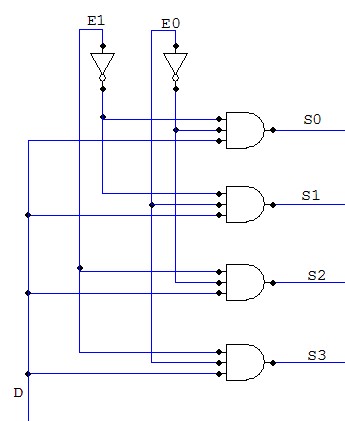


Figura : Circuito lógico do demultiplexador de 4 vias.

Pode-se reparar que para um demultiplexador, se a entrada de dados D = 1, este dispositivo funciona como um decodificador. Por isso muitas vezes estes dois dispositivos são tratados como o mesmo.

### Comparador de Magnitudes

O comparador de magnitudes compara duas quantidades binárias (A e B) e indica nas suas o resultado desta comparação. Para um comparador completo de 1 bit, é feita a comparação entre A e B, e as possíveis saídas são A > B, A = B, A < B, como apresentado na tabela verdade da Tabela 13 e circuito lógico na Figura 36.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | A > B | A = B | A < B |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |

Tabela : Tabela verdade do comparador de magnitudes de 1 bit.

Desta tabela verdade pode-se concluir:



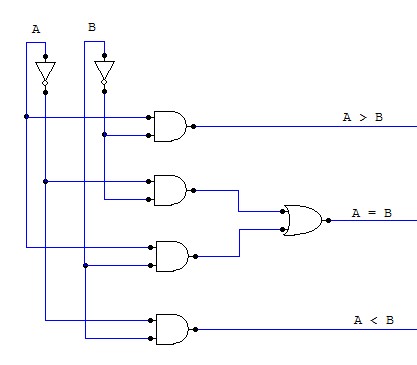


Figura : Circuito lógico do comparador de magnitudes de 1 bit

O circuito 7485 implementa a um comparador de magnitudes de 4 bits, como apresentado na Figura 37. As entradas chamadas de *condição anterior* permitem associar dispositivos em cascata, formando comparadores de 8 bits, como apresentado na Figura 38.

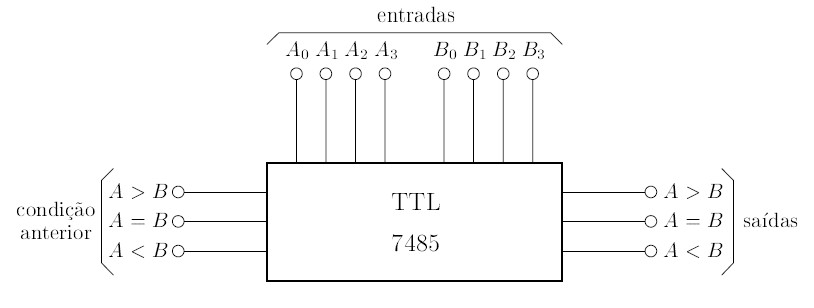


Figura : Comparador de 4 bits.

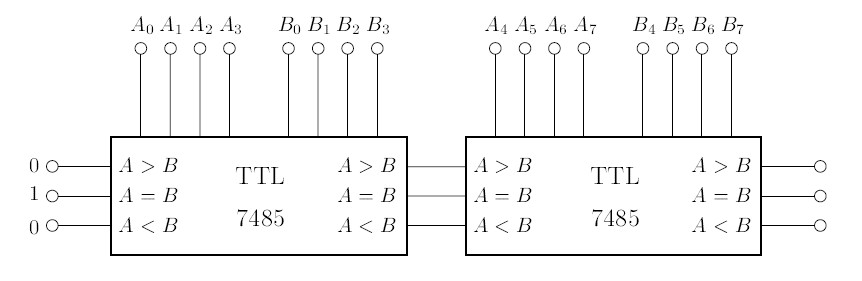


Figura : Comparador de 8 bits.

# Circuitos Lógicos Sequenciais

Nos circuitos convencionais, as saídas em um instante t dependem única e exclusivamente das entradas do circuito naquele mesmo instante, ou seja, do estado atual de suas entradas. Nos circuitos que serão estudados neste capítulo, as saídas não dependem somente do estado atual das entradas, mas também de estados anteriores do circuito.

Pode-se representar um circuito sequencial como sendo composto de um circuito combinacional associado a uma unidade de memória, capaz de armazenar estados anteriores do sistema:

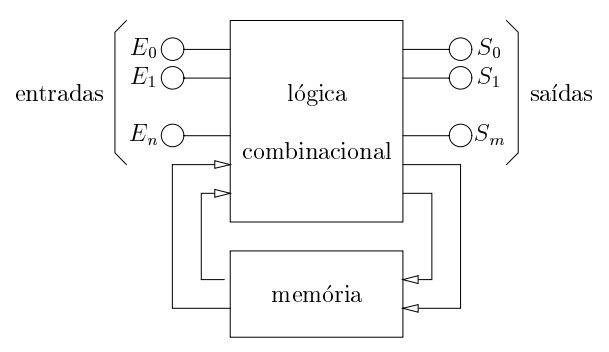


Figura : Modelo de circuitos lógicos sequenciais.

Neste capítulo serão estudados somente os circuitos sequenciais síncronos. Neste tipo de circuito todas as mudanças de estado internas são estimuladas por um sinal externo de sincronização chamado *clock*, que geralmente é uma onda quadrada de frequência constante. O sinal de *clock* deve ser comum e único para todas as partes do circuito. Desta forma, se considera que as mudanças de estado em um circuito sequencial são provocadas pelos pulsos do sinal de *clock*, em instantes de tempo discretos: t, t + 1, t + 2, t + 3...

A saída de um circuito sequencial é função de um número limitado de estados anteriores, que devem, portanto, ser armazenados em algum lugar do circuito sequencial. As unidades básicas de memória empregadas em circuitos sequenciais são os *flip-flops*, que serão estudados a seguir.

## *Flip-flops*

Um *flip-flop* (gangorra, em inglês) é um dispositivo que possui **dois** estados internos estáveis e complementares, sendo por isso também chamado de **biestável**. O estado interno de um *flip-flop* vale "0" ou "1", e pode permanecer indefinidamente em um deles, em função de suas entradas.

Desta forma, um flip-flop pode armazenar 1 bit de informação digital, e por isso constitui a unidade elementar de memória. Um flip-flop normalmente apresenta um conjunto de entradas capaz de alterar seu estado interno, e duas saídas apresentando respectivamente seu estado interno e o complemento deste. Existem diversos tipos de *flip-flops* com comportamentos distintos, mas que podem ser transformados entre si. A seguir serão vistos os tipos mais conhecidos.

### Flip-flop RS (*Reset-Set*)

O flip-flop RS (*Reset-Set*) é o mais simples desses dispositivos, e pode ser facilmente construído a partir de duas portas NAND, como mostra a Figura 40:

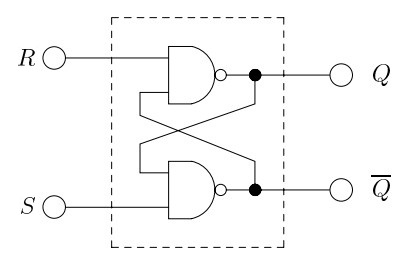


Figura : *Flip-flop* RS.

Os *flip-flops* possuem a característica de estabilizar em um determinado estado, conforme o nível das entradas aplicadas. Ao manter as entradas R e S ativas (nível lógico alto), existem dois estados possíveis de equilíbrio para essa estrutura, que podem ser observados na Figura 41. As saídasede um *flip-flop* são, por definição, complementares.

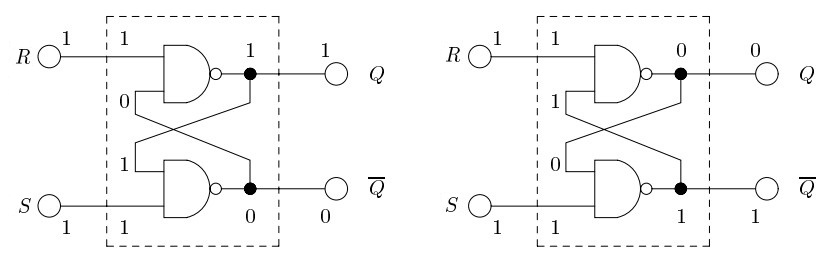
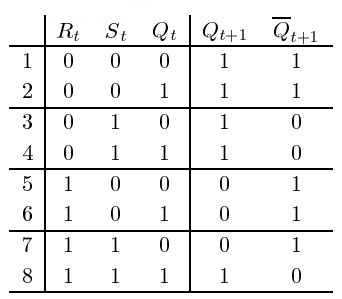


Figura : Possíveis estados estáveis do *flip-flop* RS.

**Tabela de Transição:** A determinação do estado das saídas de um *flip-flop* RS em um instante futuro ( e ) deve levar em conta suas entradas atuais ee o estado interno atual  do mesmo. Assim pode-se construir a tabela verdade apresentada na Tabela 14.

Tabela : Tabela verdade para o *flip-flop* RS (Tabela de Transição).

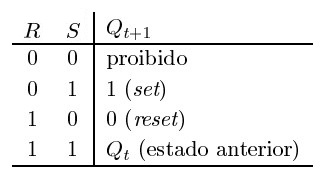


Todas as situações possíveis para as entradas do flip-flop são as seguintes:

* Nas linhas 1 e 2, temos R = S = 0 (ambas inativas), e por consequência  =  = 1, o que corresponde a uma situação proibida no funcionamento normal do flip-flop (as duas saídas sempre devem ser complementares).
* As linhas 3 e 4, nas quais R = 0 e S = 1 correspondem a uma situação de *Set*, ou seja, impõe-se o valor "1" ao estado seguinte  do *flip-flop*, pouco importando seu estado atual .
* As linhas 5 e 6, nas quais R = 1 e S = 0 correspondem a uma situação de Reset, ou seja, impõe-se o valor "0" ao estado seguinte  do *flip-flop*, pouco importando seu estado atual .
* Na situação das linhas 7 e 8 temos R = 1 e S = 1 e o *flip-flop* armazena seu estado anterior: = .

**Tabela de Função:** Após estas constatações, a tabela anterior pode ser resumida como apresentado na Tabela 15. A tabela de função é obtida diretamente da tabela de transição e mostra as características básicas de funcionamento do *flip-flop*.

Tabela : Tabela verdade do *flip-flop* RS baseada no estado anterior (Tabela de Função).



Por meio da tabela de transição pode ser construído o mapa de Karnaugh e se obter a função lógica que representa o comportamento do *flip-flop* RS. Como os estados onde R = S = 0 são proibidos, pode-se considerar a função nestes pontos como sendo irrelevante (X). Assim se obtém o mapa e a expressão lógica associada ao *flip-flop* (equação de transição), apresentados na Figura 42**.**

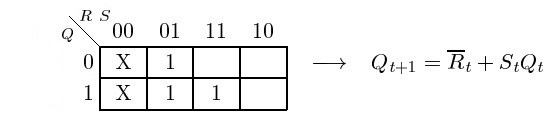


Figura : Equação de transição do *flip-flop* RS.

**Tabela de excitação:** A tabela de excitação possibilita saber os valores das excitações (R(t) e S(t)) quando da ocorrência de uma transição Q(t) → Q(t+1). Para gerar a tabela de excitação, usa-se a equação de transição.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  | | --- | --- | --- | --- | | Q(t) → | Q(t+1) | R(t) | S(t) | | 0 | 0 | 1 | X | | 0 | 1 | 0 | X | | 1 | 0 | 1 | 0 | | 1 | 1 | X | 1 | |

Esta tabela é usada na síntese dos circuitos sequenciais, pois se conhecendo as transições que deverão ocorrer, pode-se determinar os valores das excitações (entradas dos *flip-flops*). De posse dos valores nas entradas de excitação pode-se determinar os circuitos combinacionais que irão propiciar as transições desejadas nos *flip-flops*.

#### Sincronização por sinal de *clock*

Pode-se modificar o flip-flop RS para incluir uma entrada de controle, que pode servir para sincronizá-lo em relação a um sinal de relógio externo (*clock*), como mostra a Figura 43.

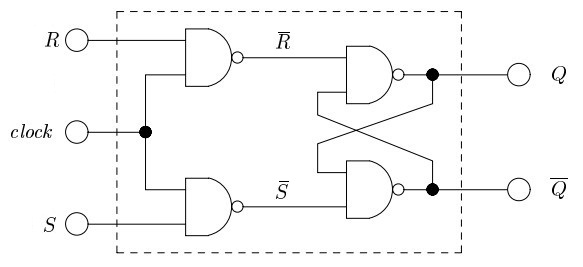


Figura : *Flip-flop* RS sincronizado com sinal de *clock*.

Neste novo circuito as entradas R e S são ditas síncronas em relação ao sinal de *clock* Ck. A tabela-verdade para esta nova configuração é apresentada na Tabela 16, onde S e R são agora entradas ativo baixo.

Tabela : Tabela verdade para o *flip-flop* RS sincronizado por *clock*.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ck | R(t) | S(t) |  |  | Q(t+1) |
| 0 | X | X | 1 | 1 | Q(t) |
| 1 | 0 | 0 | 1 | 1 | Q(t) |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | proibido |

#### Entradas Assíncronas

Pode-se ainda acrescentar à estrutura do *flip-flop* RS duas entradas, do tipo *clear* e *preset* (ambas ativo baixo), como apresentado na Figura 44, para alterar imediatamente o estado do *flip-flop* sem necessidade das entradas R e S (ou seja, de forma assíncrona em relação ao sinal de clock). A entrada *clear* tem a função de forçar a saída Q(t+1) = 0 e a entrada *preset*, Q(t+1) = 1.

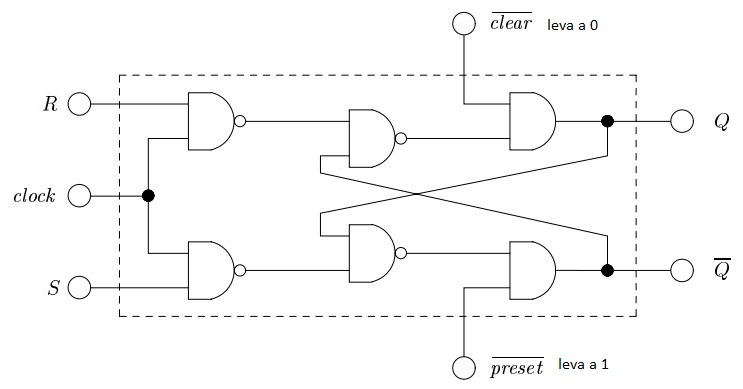
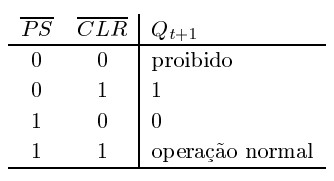


Figura : *Flip-flop* RS modificado com entradas *clear* e *preset*.

A partir do circuito apresentado, monta-se a tabela verdade apresentada na Tabela 17, na qual se pode observar que as entradas *preset* e *clear* não podem operar simultaneamente, pois neste caso .

Tabela : Tabela verdade para o *Flip-flop* RS modificado com entradas *clear* e *preset*.



#### Níveis e transições

Os circuitos vistos até o momento têm entradas sensíveis a nível, cujo funcionamento é baseado em níveis lógicos constantes e bem definidos (0 ou 1). Este comportamento é desejável em circuitos combinacionais, mas pode provocar problemas em circuitos sequenciais síncronos. Por exemplo, o circuito do flip-flop RS com entrada de sincronização (*clock*) vai estar ativo durante toda a duração de cada pulso de clock. Com isto, ele pode mudar de estado diversas vezes em cada pulso de *clock*, caso as entradas mudem durante esse intervalo. Para contornar este problema foram criadas as entradas sensíveis a transição, que são consideradas ativas somente durante as transições de nível logico do sinal aplicado. Com isso uma entrada pode ser sensível a quatro diferentes tipos de excitação:

* Nível logico alto: o sinal aplicado vale 1.
* Nível logico baixo: o sinal aplicado vale 0.
* Transição positiva: o sinal aplicado passa de 0 a 1.
* Transição negativa: o sinal aplicado passa de 1 a 0.

O sinal de *clock* é normalmente aplicado a entradas sensíveis a transição, positiva ou negativa. Transições positivas e negativas são representadas em tabelas-verdade respectivamente pelos símbolos ↑ e ↓. A Figura 45 indica os pontos de sensibilidade de uma entrada em relação ao sinal de *clock*, e indica os símbolos usados para representar cada tipo de entrada.

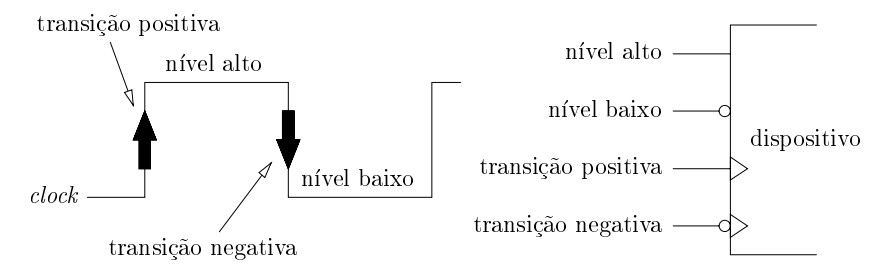


Figura : Diferentes tipos de transição.

#### Flip-flop Mestre escravo

Antes da disponibilidade de *flip-flops* com entradas sensíveis a transição, uma estrutura especial denominada *flip-flop* mestre-escravo era usada para isolar as entradas instáveis e sincronizar o sistema em relação ao *clock*. O circuito normalmente usado para implementar *flip-flops* mestre-escravo é composto por dois *flip-flops* RS em cascata, e pode ser visto na Figura 46.

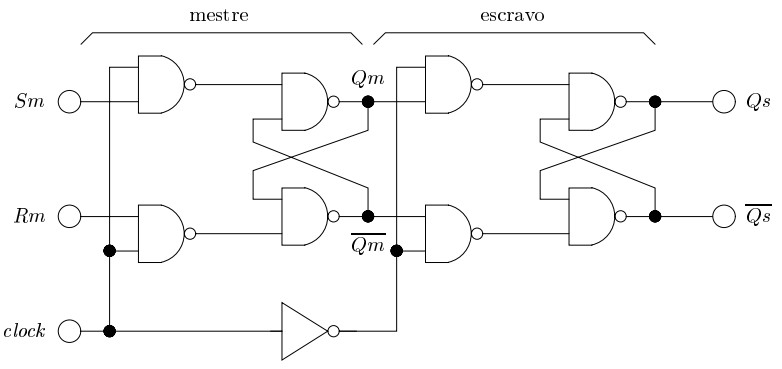


Figura : Construção do *flip-flop* mestre-escravo.

Para fazer a análise do que ocorre em um *flip-flop* mestre-escravo deve-se observar um pulso completo de *clock*. No início, com a descida do sinal de *clock* no escravo, este é desconectado do mestre. Ao final da subida, as entradas do mestre são habilitadas e este começa a atualizar seu estado interno em função das entradas. O mestre permanece assim até o início da descida do pulso de *clock*, quando suas entradas são desabilitadas, e ao final da descida seu estado é transferido ao escravo. Com isso, a saída do *flip-flop* escravo é completamente isolada de eventuais oscilações nas entradas Sm e Rm. A Figura 47 ilustra esse comportamento.

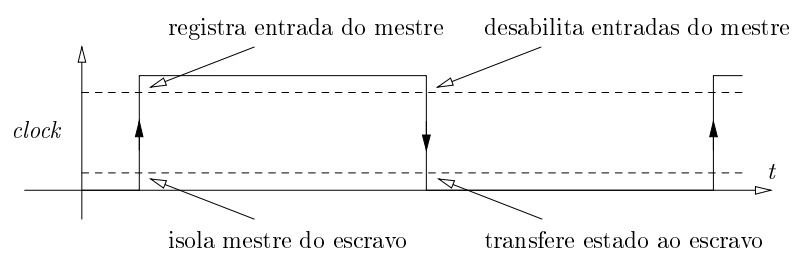


Figura : Análise do diagrama de tempo em *flip-flop* mestre-escravo.

O flip-flop de tipo mestre-escravo caiu em desuso com o surgimento de entradas sensíveis a transição, sendo raramente usado hoje em dia.

### Flip-flop D (Data)

Neste tipo de *flip-flop*, a saída Q assume o valor de uma entrada de dados D sempre que este for habilitado pelo sinal de *clock*. Este tipo de dispositivo é bastante empregado na construção de registradores de deslocamento e no armazenamento de dados (*buffer*). Sua representação e sua tabela-verdade são apresentadas na Figura 48. A partir da tabela pode-se constatar que na transição do sinal de *clock*, a saída vale .

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C:\Users\rafael\Desktop\figuras modificadas\ff d representacao e tva.jpg | **Tabela de transição:**   |  |  |  | | --- | --- | --- | | D(t) | Q(t) | Q(t+1) | | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 1 | | 1 | 1 | 1 | |

Figura : Representação e tabela de transição do *flip-flop* do tipo D.

Com isso pode-se construir a tabela de função deste *flip-flop* e obter a sua equação de transição. A equação de transição é obtida diretamente a partir da tabela de transição, de onde são retirados os mintermos, como apresentado a seguir.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela de função:**   |  |  | | --- | --- | | D(t) | Q(t+1) | | 0 | 0 | | 1 | 1 | | **Equação de transição:** |

**Tabela de excitação:** É obtida a partir da tabela de transição, por rearranjo das linhas.

|  |  |  |
| --- | --- | --- |
| Q(t) → | Q(t+1) | D(t) |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

### Flip-flop T (*Toggle*)

Neste tipo de *flip-flop* a saída Q é invertida (*toggled*) quando a entrada T está ativa e o sinal de *clock* sofre uma transição. Este tipo de dispositivo é empregado em contadores e divisores de frequência, pois seu comportamento permite a divisão por dois da frequência do sinal de entrada. Sua representação e sua tabela de transição são apresentadas na Figura 49.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C:\Users\rafa\Dropbox\IFC\Sistemas Digitais - Automacao\Apostila de Sistemas Digitais\figuras modificadas\ff t representacao e tva.jpg | **Tabela de transição:**   |  |  |  | | --- | --- | --- | | T(t) | Q(t) | Q(t+1) | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 | |

Figura : Representação e tabela de transição para o *flip-flop* do tipo T.

A tabela de função deste *flip-flop* e a sua equação de transição são apresentados a seguir. A equação de transição é obtida diretamente a partir da tabela de transição, de onde são retirados os mintermos, como apresentado a seguir.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela de função:**   |  |  | | --- | --- | | T(t) | Q(t+1) | | 0 | Q(t) | | 1 |  | | **Equação de transição:** |

**Tabela de Excitação:** É obtida a partir da tabela de transição, por rearranjo das linhas.

|  |  |  |
| --- | --- | --- |
| Q(t) → | Q(t+1) | T(t) |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Para compreender melhor o funcionamento deste tipo de flip-flop, deve-se observar seu comportamento temporal apresentado na Figura 50. Durante o período em que T = 1, a cada transição positiva do *clock* o nível da saída Q é invertido. Com isso, o sinal da saída tem a metade da frequência do sinal de *clock*.

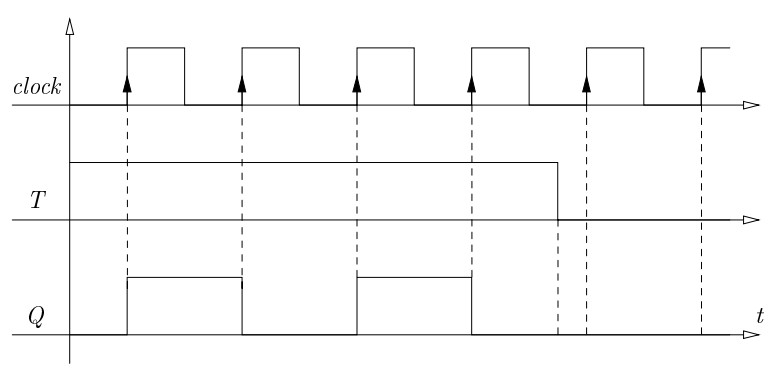


Figura : Comportamento temporal do *flip-flop* do tipo T.

### Flip-flop JK

O *flip-flop* de tipo JK apresenta um comportamento misto entre os *flip-flops* RS e T. Este dispositivo possui duas entradas J e K (que equivalem respectivamente às entradas S e R dos *flip-flops* RS) e uma entrada de *clock*. Para os estados normais, o comportamento é o mesmo do flip-flop RS, mas quando J = K = 1 (definido como estado proibido no *flip-flop* RS), a saída é complementada, como ocorre no flip-flop T. Com isso, pode-se definir a representação e tabela-verdade para o flip-flop JK apresentadas na Figura 51.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C:\Users\rafa\Dropbox\IFC\Sistemas Digitais - Automacao\Apostila de Sistemas Digitais\figuras modificadas\ff jk representacao e tva.jpg | **Tabela de transição:**   |  |  |  |  | | --- | --- | --- | --- | | J(t) | K(t) | Q(t) | Q(t+1) | | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 1 | | 0 | 1 | 0 | 0 | | 0 | 1 | 1 | 0 | | 1 | 0 | 0 | 1 | | 1 | 0 | 1 | 1 | | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 0 | |

Figura : Representação e tabela de transição para o *flip-flop* do tipo JK.

A tabela de função deste *flip-flop* e a equação de transição são apresentados. A equação de transição é obtida do um mapa de Karnaugh feito com a tabela de transição.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabela de função:**   |  |  |  | | --- | --- | --- | | J(t) | K(t) | Q(t+1) | | 0 | 0 | Q(t) | | 0 | 1 | 0 | | 1 | 0 | 1 | | 1 | 1 |  | | **Equação de transição:**  C:\Users\rafael\Desktop\ff jk mapa k e expressao.jpg |

**Tabela de Excitação:** Obtida por meio da equação de transição. O flip-flop JK é bastante popular, pois a partir dele podem-se obter os demais *flip-flops* sem a necessidade de circuitos adicionais, como será visto na seção 5.1.5.

|  |  |  |  |
| --- | --- | --- | --- |
| Q(t) → | Q(t+1) | J(t) | K(t) |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

### Conversão entre *flip-flops*

Os *flip-flops* apresentados têm comportamentos similares, e podem ser facilmente convertidos entre si através de conexões simples e do uso de algumas portas adicionais, como apresentado nos seguintes exemplos:

|  |  |
| --- | --- |
| C:\Users\rafael\Desktop\conversao jk d.jpg  Figura : Conversão de flip-flop JK em flip-flop D. | C:\Users\rafael\Desktop\conversao jk t.jpg  Figura : Conversão de flip-flop JK em flip-flop T. |
| C:\Users\rafael\Desktop\conversao rs t.jpg  Figura : Conversão de flip-flop RS em flip-flop T. | C:\Users\rafael\Desktop\conversao rs jk.jpg  Figura : Conversão de flip-flop RS em flip-flop JK. |
| C:\Users\rafael\Desktop\conversao d t.jpg  Figura : Conversão de flip-flop D em flip-flop T. |  |

### Parâmetros Operacionais

Para a operação correta dos *flip-flops* alguns parâmetros devem ser respeitados, sobretudo no que diz respeito às características temporais dos sinais de entrada. Os parâmetros mais importantes são:

* **Frequência máxima fmax:** é a máxima frequência admitida para o sinal de *clock*, ou seja, a máxima frequência de operação do dispositivo.
* **Tempo de setup tsetup:** é o tempo mínimo de presença do sinal em uma entrada de dados antes da ocorrência do pulso de *clock*.
* **Tempo de permanência thold:** é o tempo mínimo que o sinal deve permanecer em uma entrada de dados apos a transição do *clock*.
* **Tempo de *preset* tpreset:** é o tempo mínimo que uma entrada do tipo *preset* ou *clear* precisa estar ativa para efetuar sua função.
* **Largura de pulso tw:** é o tempo mínimo que o *clock* precisa permanecer em um nível alto (caso a porta seja sensível a ↑) ou baixo (caso a porta seja sensível a ↓) para que possa ser confiável.

A Figura 57 ilustra esses parâmetros, que são fortemente dependentes da tecnologia empregada para a construção do dispositivo. Estes valores são apresentados em detalhes nos manuais de dados técnicos dos dispositivos (*Data Sheets* dos fabricantes).

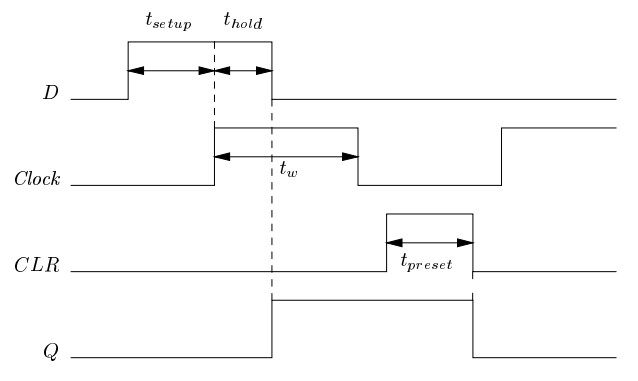


Figura : Parâmetros operacionais.

## Diagramas de Estado

Pode-se representar o comportamento de um circuito sequencial por meio de equações lógicas ou de tabelas-verdade. Essa forma de representação somente é viável para circuitos simples, contendo apenas um flip-flop. Quando a complexidade do circuito aumenta, torna-se necessário empregar outras formas de representação para definir seu comportamento de maneira fiel e sem inconsistências ou ambiguidades. Uma ferramenta muito útil para a representação do comportamento de circuitos sequenciais complexos é o diagrama de estados, também chamado de autômato finito, que veremos nesta seção.

### Estrutura Básica

Um diagrama de estados é uma construção gráfica composta por um conjunto de estados (indicados por círculos) e de transições (indicadas por arcos com setas). Os estados representam todas as situações possíveis para o sistema, e as transições indicam as mudanças de estado possíveis, e em que condições elas são provocadas (os valores das entradas que as provocam). O estado inicial do sistema é indicado por um círculo duplo.

A Figura 58 apresenta o exemplo de um diagrama de estados de um semáforo. Esse diagrama possui os estados verde (estado inicial), amarelo, vermelho e parado. As transições entre estados estão indicadas pelas setas, com os nomes dos eventos que as provocam.

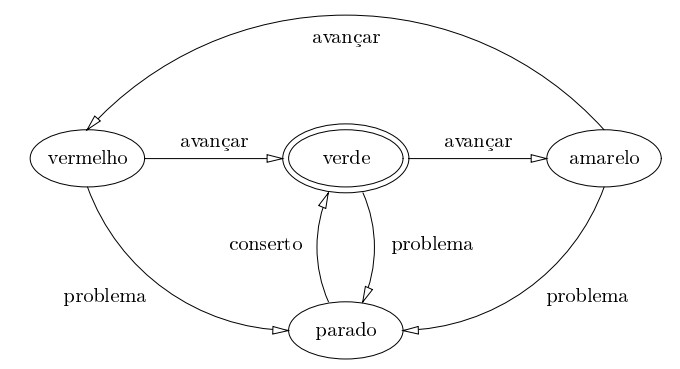


Figura : Diagrama de estados de um semáforo.

No caso específico dos circuitos sequenciais síncronos, o diagrama de estados possui algumas características importantes que devem ser levadas em conta em sua interpretação. A primeira é o seu aspecto síncrono: o diagrama deve indicar as mudanças possíveis no sistema no próximo pulso de *clock*, em função do valor das entradas do circuito naquele momento.

Outra característica diz respeito aos rótulos dos arcos que representam as transições: um determinado arco leva de um estado atual ao seu próximo estado, e seu rótulo indica a combinação de entrada que ativa aquela transição e o valor de saída que ela irá provocar. A Figura 59 ilustra esse funcionamento:

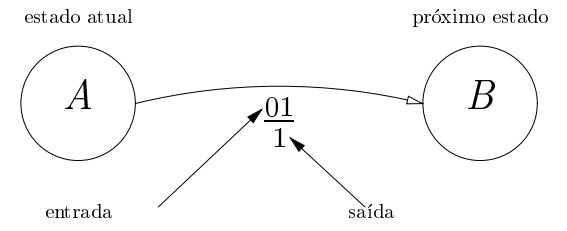


Figura : Simplificação de uma transição de diagramas de estado de circuitos sequenciais.

A interpretação da transição acima é a seguinte: no próximo pulso de *clock*, se o sistema estiver no estado interno A e ocorrer a entrada 01, então o sistema passará ao estado B e a saída passará a valer 1.

### Exemplo: Somador Serial

Inicialmente, utiliza-se o diagrama de estados para representar o comportamento de um circuito um pouco mais complexo: um somador completo serial. Este circuito tem duas entradas At e Bt que recebem os dois dígitos binários a somar, e uma saída St, que apresenta a soma obtida. Além disso, o circuito armazena o excesso Ct-1 (*carry*) da soma anterior e considera esse valor na soma atual. Desta forma, podemos somar números binários longos, processando um bit por pulso de *clock*.

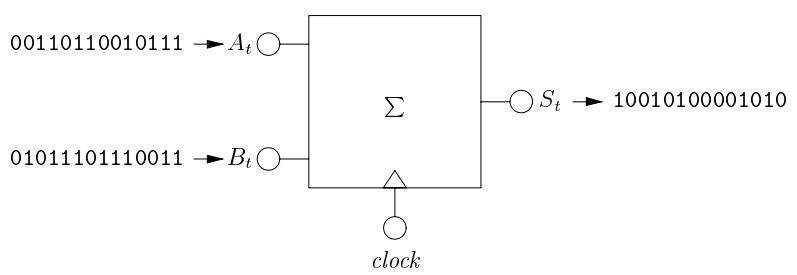
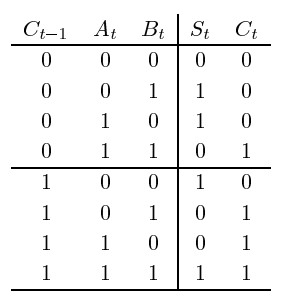


Figura : Representação do somador serial.

A partir das entradas At e Bt e do excesso da última soma Ct-1 podemos construir a seguinte tabela-verdade para a soma atual St e seu excesso Ct:



O valor a ser memorizado é o último excesso, que constitui, portanto, o estado interno do sistema. Com isso podemos deduzir que o sistema possui dois estados internos: um quando Ct-1 = 0, que será chamado q0, e outro quando Ct-1 = 1, que será chamado q1. Pode-se então construir o diagrama de estados que representa seu comportamento, considerando como entradas o par At e Bt e como saída a soma St como apresentado na Figura 61.

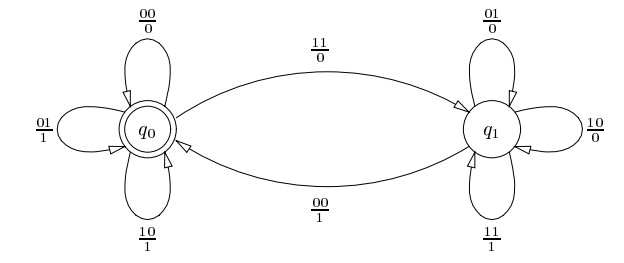


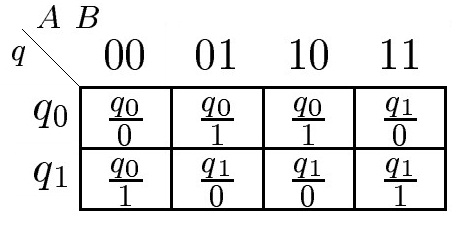
Figura : Diagrama de estados do somador serial.

O diagrama acima permite representar de forma sucinta e sem ambiguidades o comportamento esperado para o somador serial. Esses diagramas serão importantes para a análise e projeto de circuitos sequenciais síncronos, por isso sua estrutura deve ser perfeitamente compreendida.

### Tabela de Estados

Uma forma alternativa de representação do comportamento de um circuito sequencial é sob a forma de uma tabela, que indica para cada estado e para cada combinação das entradas, o próximo estado e o valor da saída (sob a forma de fração). Para o somador serial podemos construir a tabela de estados apresentada na Tabela 18 (que pode ser obtida da tabela-verdade ou do diagrama de estados). O diagrama de estados e a tabela de estados contém exatamente a mesma informação e por isso são equivalentes.

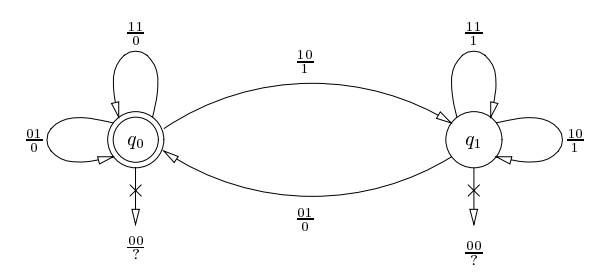
Tabela : Tabela de estados do somador serial.



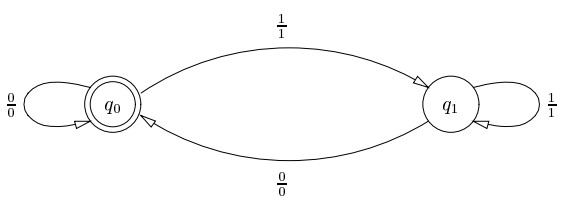
### Diagrama de estados dos *flip-flops*

Os diagramas de estado dos *flip-flops* podem ser construídos a partir das tabelas-verdade e das equações lógicas vistas ate o momento para os mesmos. O diagrama de estado de um *flip-flop* é bastante simples, porque a saída se confunde com o próximo estado interno do sistema (Q é ao mesmo tempo saída e estado). A seguir serão apresentados os diagramas de estado dos principais *flip-flops*:

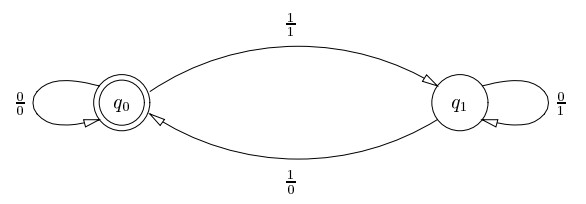
***Flip-flop* RS:** Considerando como entrada o par S R e como saída Q:



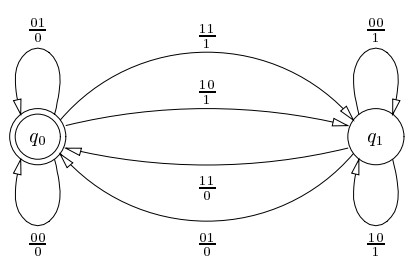
***Flip-flop* D:** Considerando como entrada D e como saída Q:



***Flip-flop* T:** Considerando como entrada T e como saída Q:



***Flip-flop* JK:** considerando como entrada o par JK e como saída Q:



## Análise de circuitos sequenciais síncronos

O comportamento de um circuito sequencial é definido por uma sequência de estados que evolui em função de seu estado atual e das entradas do circuito; essa evolução é estimulada por um sinal de *clock*.

### Objetivo da análise

O objetivo da análise dos circuitos com flip-flops é a obtenção do diagrama de estados do circuito, que determina todas as relações entre entradas, estados atuais, saídas e próximos estados, possibilitando a completa compreensão do seu funcionamento.

Para a análise de um circuito sequencial deve ser seguido o seguinte procedimento:

**1.** Identificar as variáveis lógicas de entrada, de saída e de controle dos *flip-flops* (excitação);

**2.** Escrever as equações de próximo estado Q(t+1) de cada *flip-flop*, em função do tipo de *flip-flop* (RS, D, T ou JK), de seu estado atual Q(t) e suas entradas de controle;

**3.** Montar uma tabela-verdade (tabela de transição) relacionando cada uma das entradas X(t), estados atuais Q(t), saídas S(t), e próximos estados Q(t + 1), da seguinte forma:

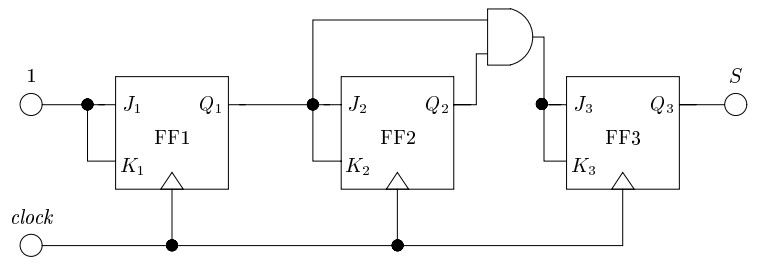
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Entradas | Estado atual | | Saídas | Próximo estado | |
| X(t) | Q1(t) | Q2(t) | S(t) | Q1(t+1) | Q2(t+1) |
| 0 | 0 | 0 | ... | ... | ... |
| 0 | 0 | 1 | ... | ... | ... |
| 0 | 1 | 0 | ... | ... | ... |
| ... | ... | ... | ... | ... | ... |

**4.** Determinar a tabela de estados do sistema (por exemplo, estado q0 corresponde a Q1(t)=0 e Q2(t)=0, estado q1 corresponde a Q1(t)=0 e Q2(t)=1, etc.);

**5.** A partir da tabela de estados, construir o diagrama de estados do circuito.

### Exemplo 1

O circuito da figura seguinte será analisado para se obter seu diagrama de estados.



O primeiro passo na análise consiste em identificar todas as variáveis envolvidas:

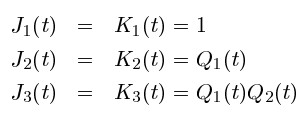
* Variáveis de entrada: nenhuma;
* Variáveis de saída: S;
* Variáveis de controle: J1, K1, J2, K2, J3, K3;
* Variáveis de estado: Q1, Q2, Q3;

A seguir devem ser identificadas e refinadas as equações:

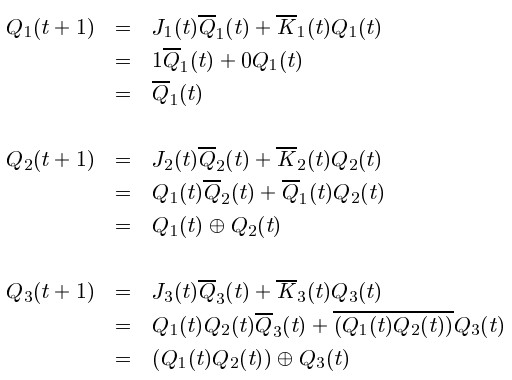
Saída:

C:\Users\rafael\Desktop\equacao saida 442.jpg

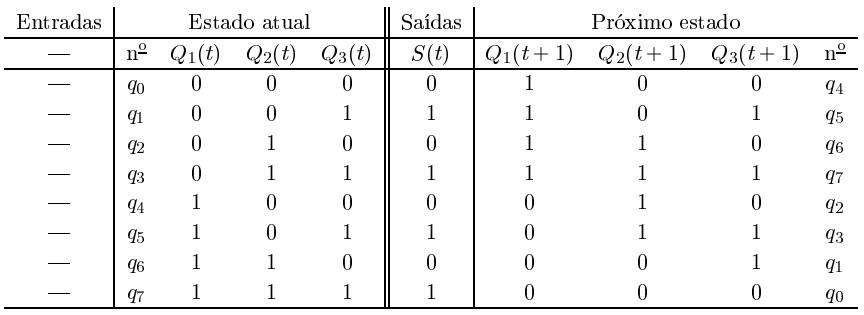
Controle:



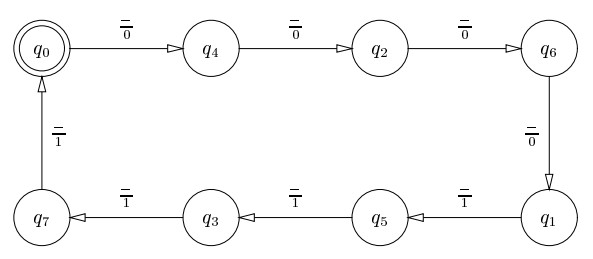
Próximo estado:



A partir desses dados pode-se construir a tabela de transição para esse circuito. Para sua construção, devem ser enumeradas todas as combinações possíveis para as entradas X(t) e estados dos *flip-flops* Q(t). A partir desses dados e das equações obtidas, podem ser determinadas as saídas do sistema S(t) e os estados futuros dos *flip-flops* Q(t + 1). Deve-se observar que cada combinação de estados dos *flip-flops* corresponde a um estado interno qi diferente para o sistema.

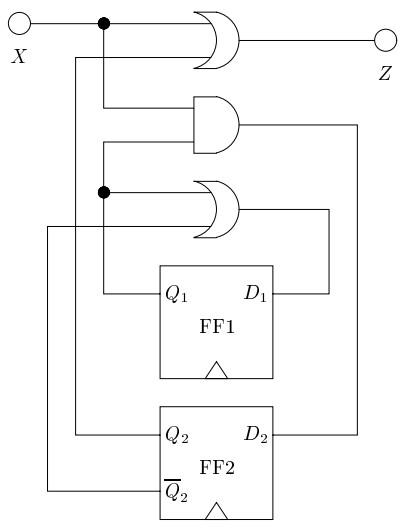


A partir da tabela-verdade obtida pode-se montar a tabela de estados do circuito, ou diretamente o diagrama de estados do circuito, que representa seu comportamento temporal. O diagrama de estados do circuito permite observar facilmente que a sequência de valores da saída do circuito será composta alternadamente por quatro "0" e quatro "1", sendo esta a função deste circuito: 000011110000111100001111...



### Exemplo 2

Seja o circuito da figura a seguir:



Primeiramente deve-se identificar todas as variáveis envolvidas:

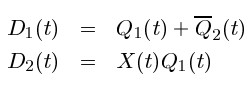
* Variáveis de entrada: X
* Variáveis de saída: Z
* Variáveis de controle: D1 D2
* Variáveis de estado: Q1 Q2

A seguir deve-se identificar e refinar as equações:

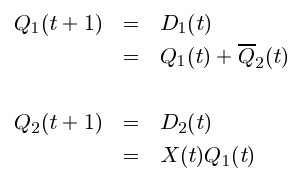
Saída:

C:\Users\rafael\Desktop\equacao saida 443.jpg

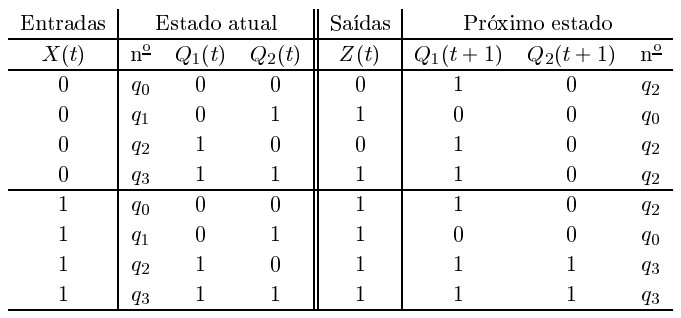
Controle:



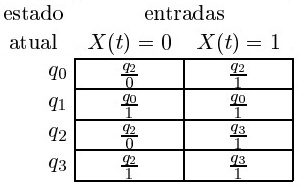
Próximo estado:



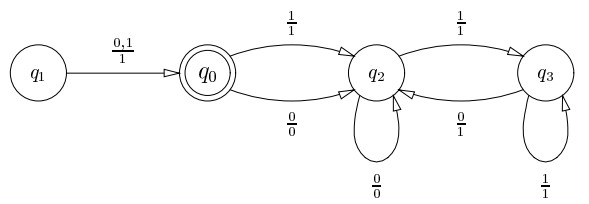
A partir desses dados deve-se construir a tabela de transição para esse circuito:



Como visto anteriormente, pode-se também representar o diagrama de estados na forma de uma tabela de estados:



Finalmente deve-se construir o diagrama de estados do circuito, para determinar o seu comportamento:

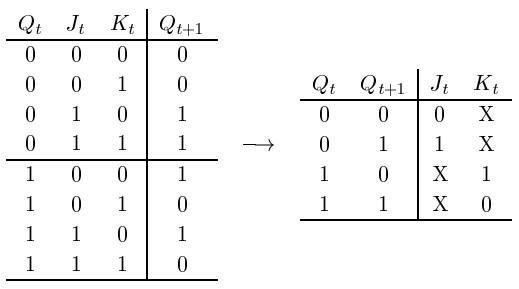


## Projeto de Circuitos Sequenciais Síncronos

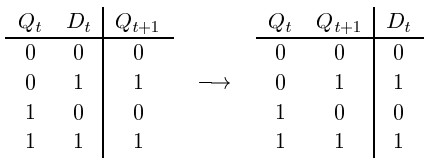
O projeto de circuitos sequenciais síncronos segue uma abordagem análoga à utilizada no processo de análise, mas em sentido inverso. Desta forma, o projeto pode ser realizado com o seguinte procedimento:

1. Descrição completa da operação desejada para o circuito, envolvendo:
   1. Identificação das entradas e saídas;
   2. Identificação dos estados internos (n *flip-flops* para 2n estados);
   3. Definição do comportamento desejado, através de um diagrama de estados.
2. Determinação da tabela de estados e da tabela de transição para o circuito, com base nos valores possíveis para as entradas e estados internos do circuito.
3. Determinação do tipo de *flip-flop* que será utilizado. Pode ser escolhido qualquer tipo, independente do problema. As entradas dos sinais de excitação dos *flip-flops* devem ser inseridas na tabela de transição anterior.
4. Minimização das funções correspondentes à parte combinacional do circuito, ou seja, saídas e controles dos *flip-flops*.
5. Construção do circuito final.

Para a determinação da tabela-verdade deve-se empregar as "tabelas de excitação" para os *flip-flops*, que permitem determinar os valores das entradas de controle em função da transição de estado desejada. Essas tabelas podem ser facilmente deduzidas a partir das tabelas de transição dos *flip-flops*. Por exemplo, para o *flip-flop* JK:



De maneira similar pode-se obter a tabela de excitação para o flip-flop D:



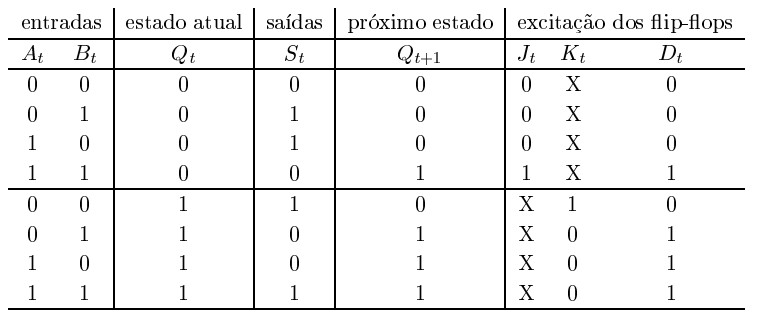
### Exemplo 1

Neste exemplo vamos projetar um circuito para implementar o somador serial cujo comportamento foi apresentado na seção 5.2.2. Nosso primeiro passo é determinar a operação desejada para o circuito, e para isso vamos determinar os seguintes dados:

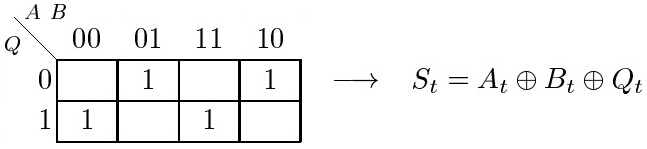
* Entradas: os bits a somar A(t) e B(t);
* Saída: a soma acumulada: S(t);
* Estados internos: dois estados indicando o bit “vai um” da ultima operação efetuada, sem *carry* (q0) e com *carry* (q1).
* Comportamento: indicado no diagrama de estados apresentado na seção 5.2.2.

Como há dois estados internos, é necessário apenas uma variável de estado Q(t) e por consequência apenas um flip-flop.

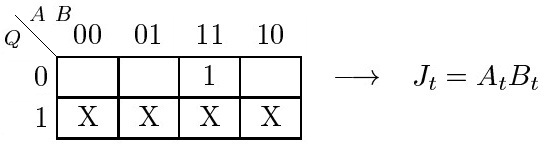
Neste ponto pode-se montar a tabela-verdade do circuito. Já são conhecidas At, Bt, St, Qt e Qt+1, apresentados no diagrama de estados. Deve-se então escolher um tipo de flip-flop para a implementação e obter as colunas correspondentes à sua excitação. Essas colunas serão determinadas a partir de cada par "estado atual -> estado futuro" para cada flip-flop, usando a tabela de excitação do flip-flop escolhido. Escolhendo um flip-flop JK ou D, a tabela de estados assume a seguinte forma:



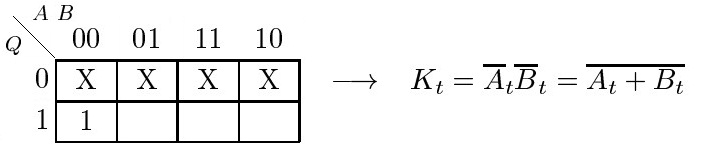
Com a tabela construída, pode-se passar à determinação das funções combinacionais necessárias à geração da saída e das excitações do flip-flop. Para St pode-se construir o seguinte mapa de Karnaugh:



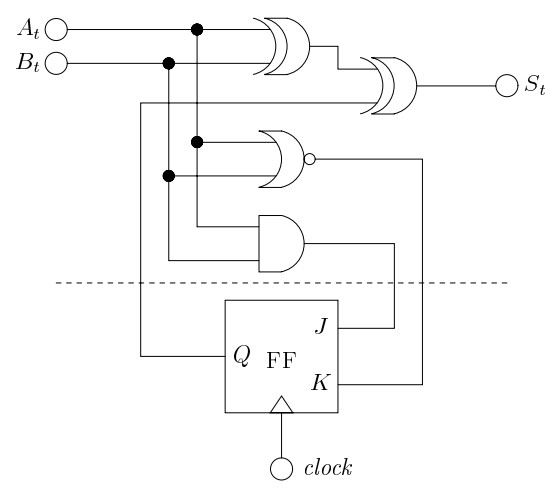
Para Jt temos:



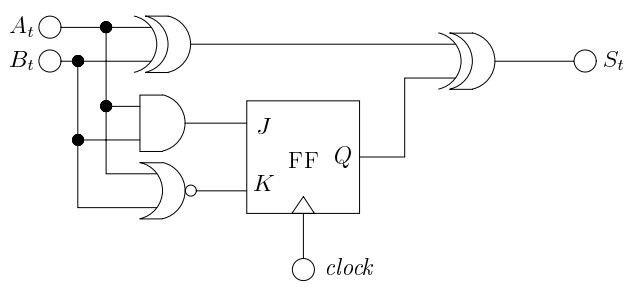
Para Kt temos:



A partir das funções mínimas obtidas pode-se sintetizar a parte combinacional do circuito, que acoplada ao *flip-flop* resultará o circuito final:

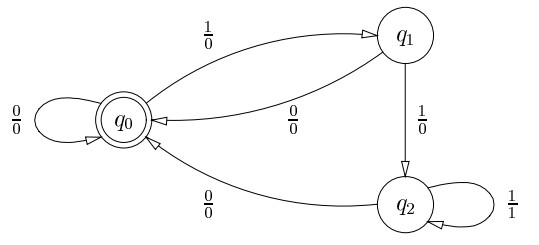


O circuito acima pode ser apresentado de uma forma mais intuitiva:



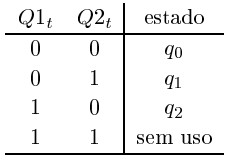
### Exemplo 2

Deseja-se construir um circuito para implementar o comportamento descrito através do seguinte diagrama de estados:



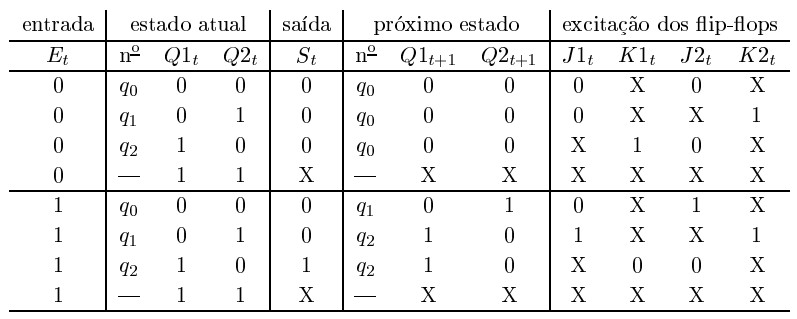
O primeiro passo é determinar a operação desejada para o circuito, e para tal determinam-se os seguintes dados:

* Entrada: uma entrada Et;
* Saída: uma saída St;
* Estados internos: o sistema possui três estados (q0, q1 e q2), o que leva a um mínimo de dois *flip-flops* e, portanto, duas variáveis de estado Q1t e Q2t, cuja combinação permitirá a indicação do estado do sistema:

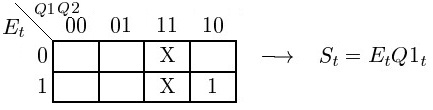


* Comportamento: indicado no diagrama de estados apresentado acima.

Assim, pode-se construir a tabela de transição do circuito. Já são conhecidas Et, St, Q1t, Q2t, Q1t+1 e Q2t+1, apresentados no diagrama de estados. Escolhendo um flip-flop JK, deve-se então determinar J1t , K1t ,J2t e K2t:



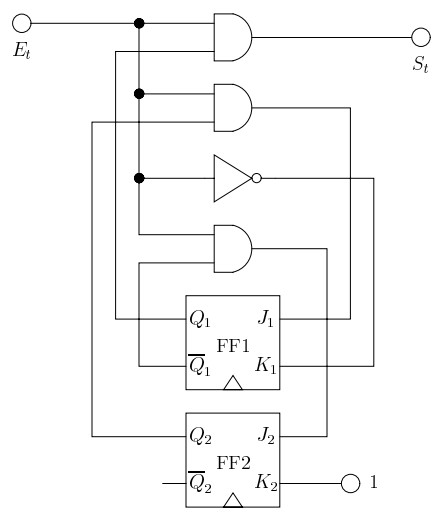
Com a tabela assim construída, pode-se passar à determinação das funções combinacionais necessárias a geração da saída e das excitações dos *flip-flops*, em função da entrada Et e dos estados Q1t e Q2t. Para St pode-se construir o seguinte mapa de Karnaugh:



Para as excitações J1t , K1t , J2t e K2t tem-se:

|  |  |
| --- | --- |
| C:\Users\rafael\Desktop\mk j1t 552.jpg | C:\Users\rafael\Desktop\mk k1t 552.jpg |
| C:\Users\rafael\Desktop\mk j2t 552.jpg | C:\Users\rafael\Desktop\mk k2t 552.jpg |

A partir das funções mínimas obtidas pode-se sintetizar a parte combinacional do circuito, que acoplada aos *flip-flops* resultará o circuito final:



## Principais circuitos sequenciais síncronos

Nesta seção serão vistos alguns circuitos sequenciais síncronos de uso bastante frequente, e, portanto, facilmente encontrados na forma de chips completos. Esses circuitos podem também ser sintetizados através da técnica de projeto vista anteriormente. Serão abordados aqui os contadores e os registradores de deslocamento.

### Contadores

Contadores são circuitos sequenciais que permitem contar pulsos de uma entrada, apresentando a contagem sob a forma de um número binário, em uma saída com n bits. Os contadores têm muitas aplicações, dentre as quais a contagem de eventos, a divisão de frequência, o sequenciamento de operações, etc. Podem ser classificados segundo alguns parâmetros:

* Sincronismo: um contador pode ser síncrono, quando todos os seus *flip-flops* estão sob o comando de um mesmo *clock*, ou assíncrono, quando os *flip-flops* podem ser excitados por *clocks* distintos.
* Sentido: um contador pode contar de forma crescente ou decrescente, ou ambas.
* Programação: um contador pode efetuar uma contagem entre dois extremos fixos, ou podem ser estabelecidos os valores inicial e final para a contagem.

#### Contadores Assíncronos

O contador de implementação mais simples é o assíncrono crescente, que pode ser facilmente obtido através da associação de flip-flops tipo T em cascata. Nos contadores assíncronos, o sinal de clock aplica-se à entrada do primeiro FF, a saída deste à entrada do próximo e assim sucessivamente.

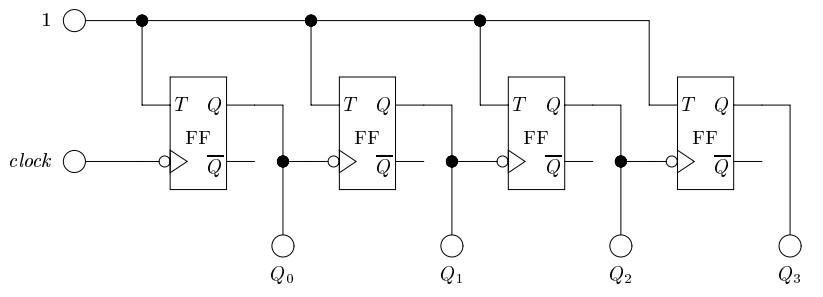


Figura : Contador assíncrono crescente com *flip-flops* do tipo T.

Este mesmo circuito pode ser construído com *flip-flops* do tipo JK, utilizando a conversão entre *flip-flops* (JK para T). Neste caso, basta conectar as duas entradas J e K para formar a entrada T, como mostra a figura.

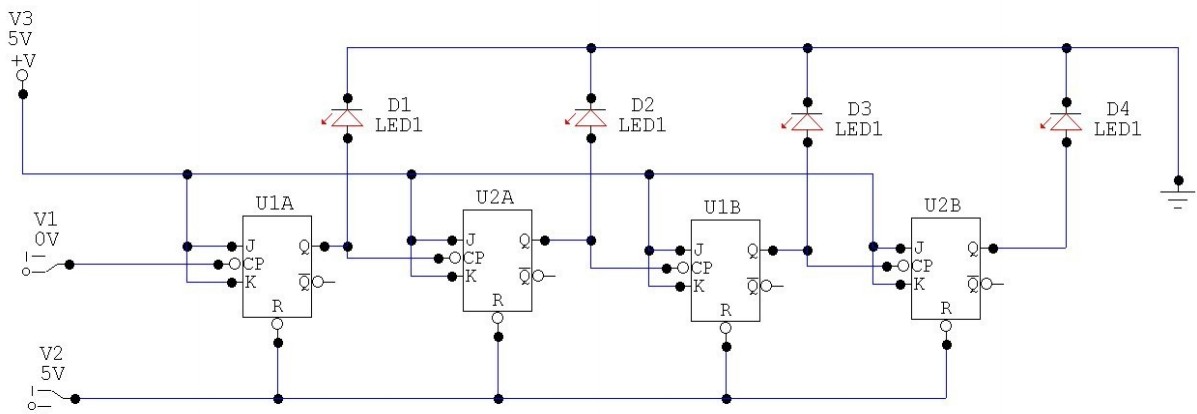
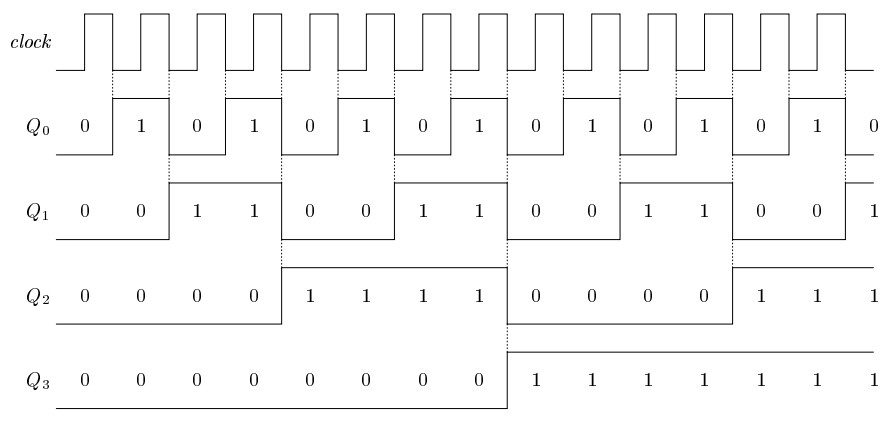


Figura : Contador assíncrono crescente com *flip-flops* do tipo JK.

O circuito acima permite contar 16 passos, de 0000 a 1111, em sentido ascendente, como mostra o diagrama temporal a seguir:



O circuito anterior pode ser visto como um divisor de frequência: para um sinal de *clock* de frequência *fc*, a saída Q0 tem frequência *fc*/2, a saída Q1 tem frequência fc/4 e assim por diante.

Para a construção de um contador assíncrono decrescente basta usar as saídas  para associar os *flip-flops*. O contador da figura abaixo efetua a contagem do ciclo 0000 -> 1111 -> 1110 -> 1101 -> 1100...

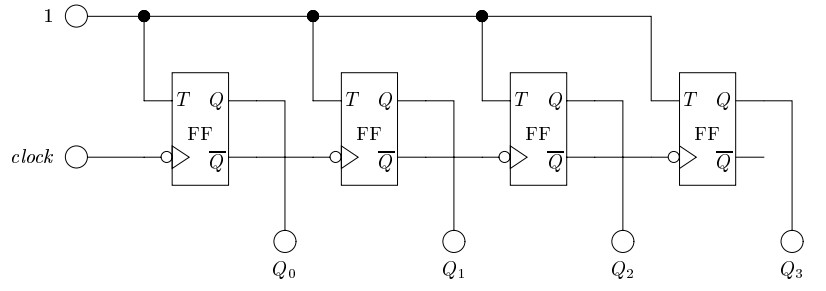


Figura : Contador assíncrono decrescente com *flip-flops* T.

Outra alternativa para a construção é tomar as saídas como saída da contagem, utilizando as saídas Q na ligação das entradas de *clock*. Na figura abaixo é dado o exemplo construído com *flip-flops* JK.

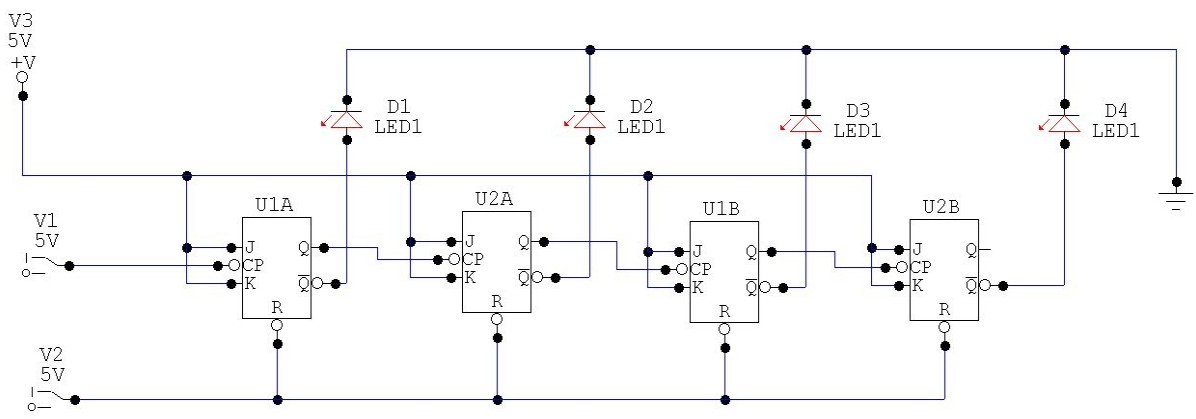
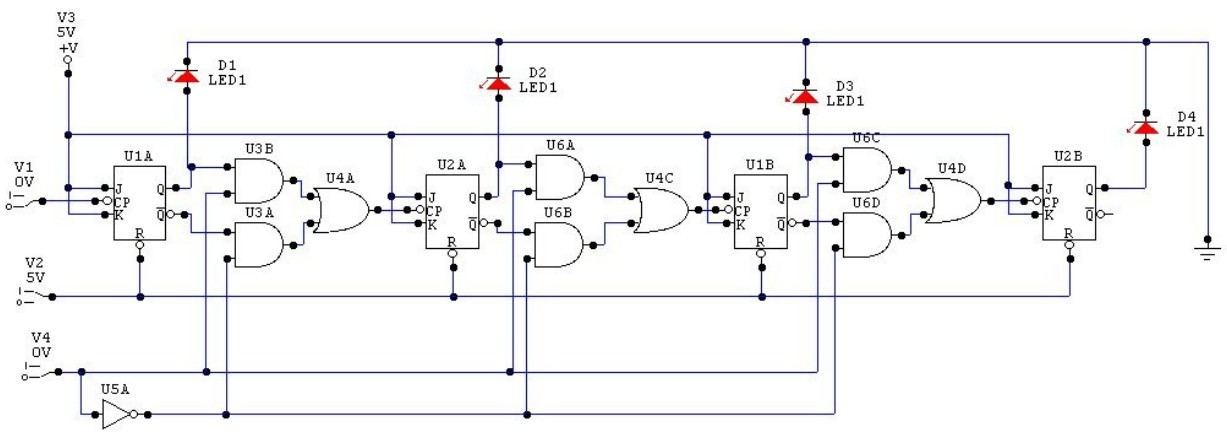


Figura : Contador assíncrono decrescente.

Podem ser associados os dois circuitos destes dois contadores para se obter um contador crescente-decrescente, sob o comando de uma porta de controle, representada neste caso pela chave V4. Caso V4 = 1 a contagem será crescente, e caso V4 = 0 ela será decrescente:



#### Contadores Síncronos

O projeto de contadores síncronos pode ser feito facilmente através da técnica de projeto descrita neste capítulo. Duas estruturas de contadores síncronos são bastante conhecidas: o contador síncrono com transporte série ou com transporte paralelo. Estes nomes indicam a forma como a evolução dos bits menos significativos do contador são consideradas na evolução dos bits mais significativos. Ambas as estruturas são apresentadas a seguir (para 4 bits):

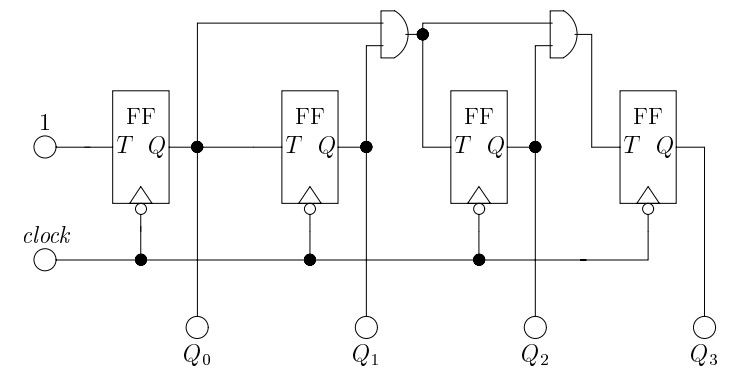


Figura : Contador síncrono com transporte série.

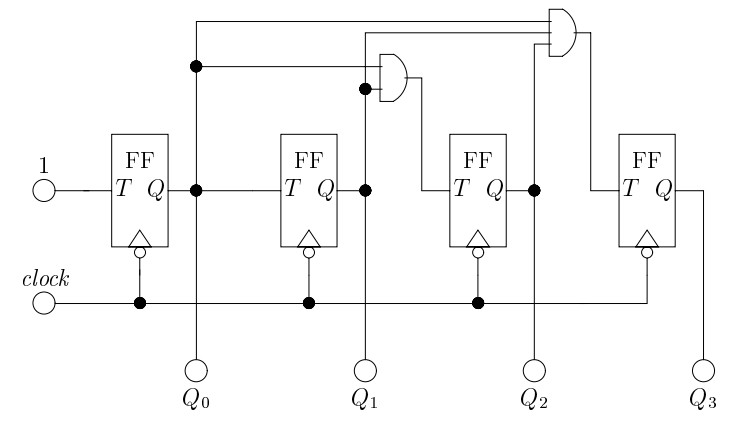


Figura : Contador síncrono com transporte paralelo.

Para a construção de contadores síncronos decrescentes ou bidirecionais podem ser seguidos os mesmos procedimentos apresentados para os contadores assíncronos, ou seja, usar ao invés de Q ou associar uma lógica de controle para selecionar entre Q e .

Os contadores vistos ate o momento permitem contar M = 2n passos, onde n é o número de *flip-flops* usados. No entanto podem ser construídos contadores para operar com um valor M qualquer, que serão chamados contadores em módulo M. Por exemplo, um contador em módulo 5 permite contar 5 passos, percorrendo o ciclo 000 -> 001 -> 010 -> 011 -> 100 -> 000...

O primeiro passo para a construção de um contador em módulo M é a determinação do número de *flip-flops* necessários para a contagem. Isso pode ser feito com base no número de dígitos binários necessários para representar os estados do contador. Por exemplo, um contador em modulo 6 (610 = 1102) irá precisar de 3 *flip-flops*, enquanto um contador em modulo 26 irá necessitar de 5 *flip-flops* (pois 2610 = 110102).

A seguir deve-se construir um contador assíncrono usando os *flip-flops* necessários. Para obter a contagem até M, devem-se conectar todas as saídas ativas no estado M a uma porta AND que irá ativar as entradas *clear* de todos os *flip-flops* (no caso das entradas *clear* serem ativo alto). Assim, quando a contagem atingir M, a porta AND será ativada e os *flip-flops* voltarão a zero, reiniciando a contagem.

Se as entradas *clear* forem ativo baixo, devem ser utilizadas portas NAND para fazer a escolha do módulo M da contagem. No exemplo a seguir, o circuito executa a contagem de 10 passos (contagem de década). Quando o resultado da contagem for 1010, as saídas assumem os seguinte valores: D4 = 1, D3 = 0, D2 = 1, D1 = 0. Isso faz com que a saída da porta NAND seja 0, ativando a entrada *clear* e zerando os *flip-flops*.

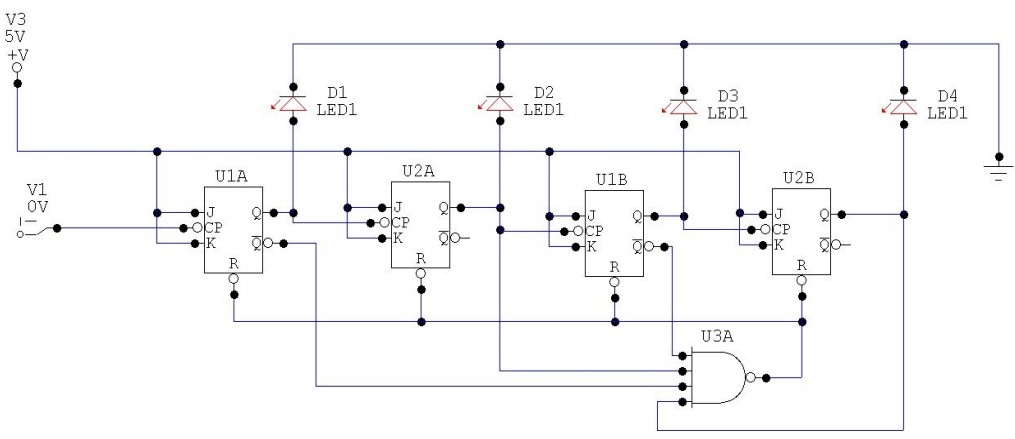


Figura : Contador assíncrono ascendente de módulo 10.

#### Exercícios:

* Projetar um contador síncrono de 3 bits (FF tipo T).
* Projetar um contador síncrono de 3 bits, de módulo M.
* Projetar um contador síncrono crescente/decrescente de 2 bits.
* Projetar um contador síncrono que conte a seguinte sequência: 0, 3, 7, 1, 5, 6 (FF tipo JK).

### Registradores de deslocamento

Um registrador de deslocamento é um arranjo linear de *n* *flip-flops* capaz de armazenar *n* bits de informação. A cada pulso de *clock* os dados podem ser deslocados uma posição para a direita ou para a esquerda, de acordo com a implementação do registrador, o que justifica o nome desse dispositivo. Os dados podem ser carregados no registrador de forma paralela (todos simultaneamente, através de entradas especiais) ou sequencial, através de um dos extremos do registrador e fazendo uso do mecanismo de deslocamento de bits. A figura a seguir mostra o diagrama simplificado de um registrador de deslocamento para a direita com 8 bits (foram omitidos o sinal de *clock* e as entradas de controle):

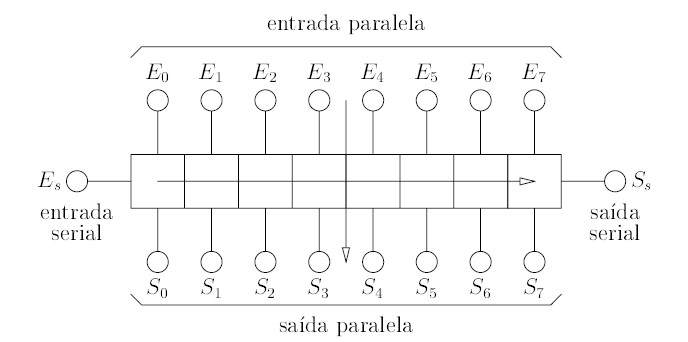


Figura : Diagrama de um registrador de deslocamento.

As duas setas indicam os sentidos possíveis do fluxo de dados no interior do registrador de deslocamento:

* **Fluxo sequencial:** os dados fluem para a direita sincronizados por *clock*. O primeiro *flip-flop* assume o valor da entrada serial Es, e o dado do outro extremo, que já atravessou o registrador, é apresentado na saída serial Ss. Portanto, a cada pulso de *clock* o dado mais antigo (à direita) é descartado e um novo dado entra no registrador (à esquerda). Desta forma, um dado demora n pulsos de *clock* para atravessar um registrador de deslocamento de *n* bits.
* **Fluxo paralelo:** o conteúdo do registrador pode ser acessado através das saídas paralelas, e pode ser totalmente alterado fazendo-se uso das entradas paralelas. Essas operações normalmente podem ser efetuadas de maneira assíncrona.

Pode-se compreender melhor o funcionamento de um registrador de deslocamento analisando seu comportamento temporal. O diagrama de tempo a seguir mostra o comportamento de um registrador de deslocamento de 4 bits com o conteúdo inicial 1000 (apenas um bit ativo no primeiro registro), e com Es = 0:

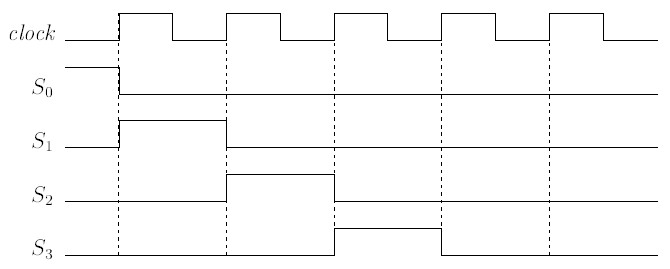


Figura : Diagrama de tempo de um registrador de 4 bits.

Segundo a maneira como a informação flui no interior de um registrador de deslocamento, pode-se classificá-lo em quatro grupos:

* **Série-série:** como cada bit demora n pulsos de clock para atravessar o registrador, este dispositivo é usado em linhas de retardo digitais, para criar atrasos no sinal digital.
* **Série-paralelo:** os dados são carregados em série e retirados através da saída paralela. Esta estrutura é normalmente usada para em comunicação de dados, para converter sinais seriais (na linha telefônica) em sinais paralelos (no interior do computador), por exemplo.
* **Paralelo-série:** exerce a função inversa do anterior, sendo por isso também empregado em comunicação de dados.
* **Paralelo-paralelo:** os dados são carregados e descarregados do registrador são feitos através das portas paralelas. Pode ser usado para deslocar (shift) valores binários, ou para armazená-los temporariamente (buffer).

A seguir serão vistas algumas estruturas simples para a implementação dos registradores de deslocamento. São empregados *flip-flops* do tipo D por serem os que melhor se enquadram neste tipo de aplicação, e por sua simplicidade. Para construir um registrador de deslocamento **série-série**, basta acoplar *flip-flops* tipo D em série:

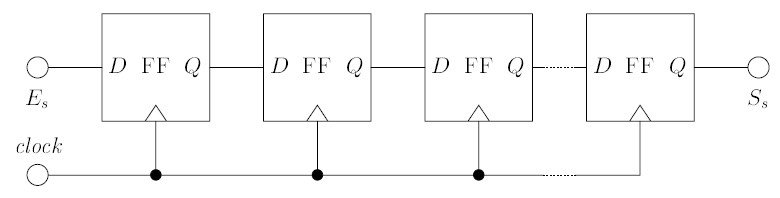


Figura : Registrador série-série com FFs tipo D.

O registrador série-paralelo tem uma implementação similar, bastando extrair a saída individual de cada flip-flop para compor a saída paralela. No exemplo a seguir é mostrado um circuito com esta funcionalidade, mas construído com *flip-flops* JK, implementando a sua conversão para *flip-flops* do tipo D.

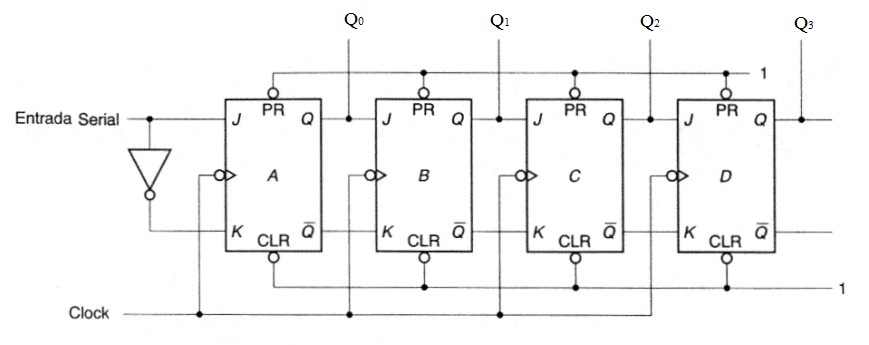


Figura : Registrador série-paralelo com FFs do tipo JK.

Para construir os registradores paralelo-série e paralelo-paralelo, são usadas as entradas *preset* e *clear* de cada *flip-flop* para carregar o registrador com os valores presentes na entrada paralela.

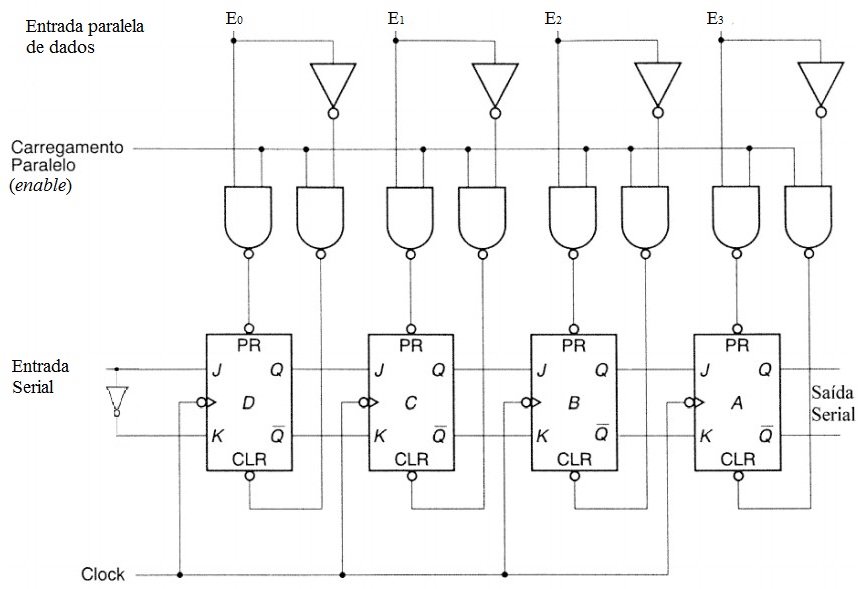


Figura : Registrador paralelo-série com FFs do tipo JK.

Pode-se também construir um registrador bidirecional, cujo sentido de deslocamento (da esquerda para a direita ou vice-versa) é estabelecido através de uma porta de controle M.

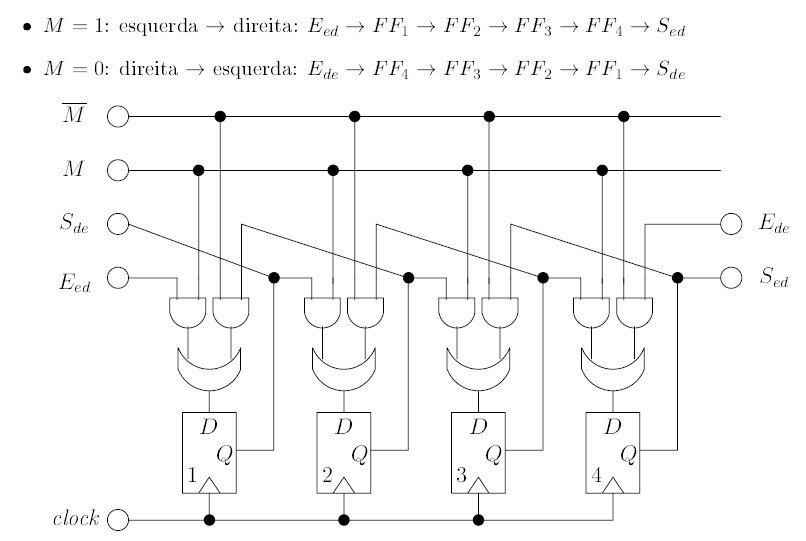


Figura : Registrador bidirecional.

Na figura abaixo é apresentado um registrador de deslocamento comercialmente disponível na família TTL: 74LS165.

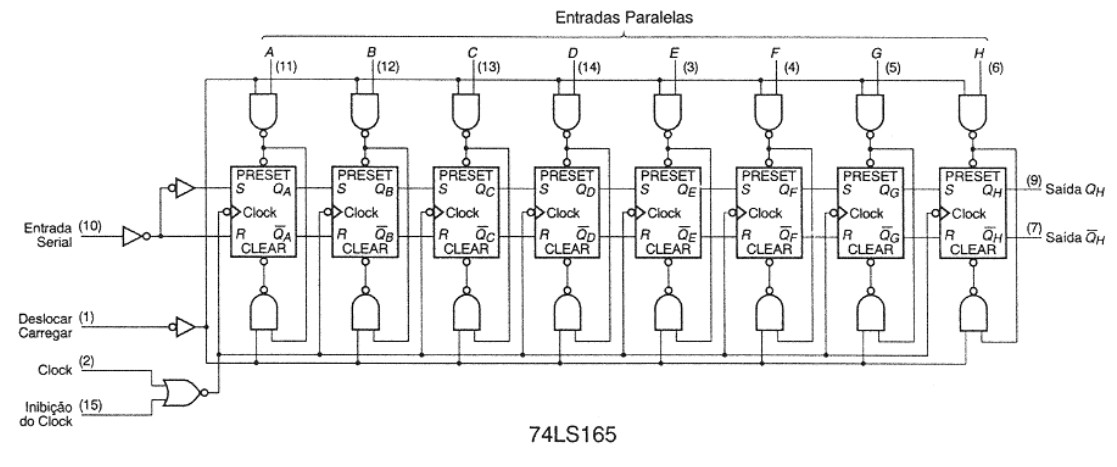


Figura : Registrador TTL 74165.

# Dispositivos de Memória

## Introdução

Um dos elementos da estrutura básica de um computador são os dispositivos de memória. Os circuitos básicos para a construção destes dispositivos são os **registradores de deslocamento**. Um registrador de deslocamento de 4 bits, por exemplo, pode armazenar uma *palavra de memória* de 4 bits, como será visto no decorrer deste capítulo.

Serão considerados como memória os **dispositivos semicondutores** capazes de armazenar dados, de forma temporária ou permanente. Os objetivos deste capítulo são: estudar os tipos de memória; entender e usar corretamente a terminologia associada aos sistemas de memória; descrever as diferenças entre memórias de leitura (ROM) e de leitura/escrita (RWM); distinguir os vários tipos de ROM e RWM; combinar CIs de memória para formar banco de memórias.

## Conceitos básicos e terminologia

**Computador:** É todo dispositivo capaz de processar dados e controlar processos externos, automaticamente, de acordo com instruções sequenciais fornecidas previamente.

**Dados:** Podem ser números, letras, palavras, símbolos gráficos quaisquer ou ainda, sinais elétricos que traduzem a ocorrência de eventos físicos do seu mundo exterior. Processar dados significa manipular ou tratar esses dados de modo a alcançar um resultado desejado.

**Instrução:** Se denomina*instrução* cada ordem fornecida ao computador para uma operação específica com os dados.

**Memória:** É todo dispositivo capaz de armazenar informações. Em uma memória, toda a informação é armazenada na forma binária.

**Célula de memória:** É um dispositivo ou circuito elétrico capaz de armazenar um único bit (0 ou 1). Como célula de memória podem ser citados o *flip-flops*.

**Palavra de memória:** É um conjunto de bits que representa instruções ou dados. Por exemplo, um registrador composto por 8 *flip-flops* pode ser considerado uma memória com capacidade de armazenar uma palavra de 8 bits.

**Byte:** É o conjunto de 8 bits.

**Endereço:** É um número que identifica a posição de uma palavra na memória, sendo expresso sempre em número binário, mas, em alguns casos por conveniência pode ser expresso em octal e hexadecimal.

**Operação de leitura:** É a operação em que uma palavra binária armazenada em uma posição específica na memória (endereço) é identificada e transferida para outro dispositivo qualquer do sistema. Essa operação muitas vezes é denominada *busca*.

**Operação de escrita:** É a operação na qual uma palavra é colocada em determinada posição da memória. Essa operação muitas vezes é denominada *armazenamento.*

A tabela a seguir representa uma pequena memória composta por 5 palavras, cada uma delas com um endereço específico.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Endereço | Palavra de memória |  |  |  |
|  |  | **000** | Palavra 0 |  |  |  |
|  |  | **001** | Palavra 1 |  |  |  |
|  |  | **010** | Palavra 2 |  |  |  |
|  |  | **011** | Palavra 3 |  |  |  |
|  |  | **100** | Palavra 4 |  |  |  |
|  |  | **101** | Palavra 5 |  |  |  |
|  |  | **110** | Palavra 6 |  |  |  |
|  |  | **111** | Palavra 7 |  |  |  |

## Estrutura básica de um computador

O computador é um sistema sequencial complexo que pode realizar diferentes operações indicadas por meio de instruções. A estrutura básica dos computadores atuais segue o modelo proposto por Von Neumann, representado na Figura 76, composta pelos seguintes elementos:

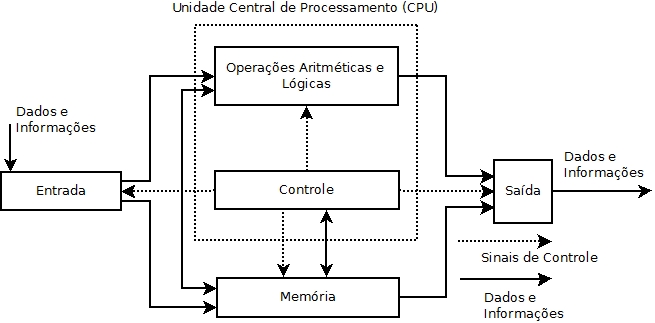


Figura : Unidades básicas de um computador.

**Unidade de Controle (UC):** Sequencia e supervisiona as tarefas que estão sendo executadas pelos demais elementos do computador. Por exemplo, durante a execução de um programa, ela controla o fluxo de dados entre os dispositivos de entrada/saída, a memória e a unidade lógica e aritmética.

**Unidade lógica e aritmética (ULA):** Executa operações lógicas e aritméticas com os dados presentes na memória. Em conjunto com a unidade de controle constitui o processador (Central Processing Unit - CPU).

**Memória:** Armazena os programas (instruções) e dados em forma binária. Cada palavra (dado ou instrução) está contida em um endereço (posição) distinto na memória.

**Dispositivos de entrada e saída:** Permitem a ligação do computador com o usuário, como teclado, monitor, mouse, etc. Nesta categoria também estão incluídos os dispositivos de armazenamento de massa, como discos rígidos, disquetes, CD-ROMs, etc.

Esses diferentes elementos se comunicam entre si através de três barramentos distintos, aos quais todos estão conectados.

**Barramento de dados:** permite transferir informações (dados) entre processador, memória e dispositivos de E/S. A largura (numero de vias) desse barramento depende do processador usado. As larguras de barramento de dados mais comuns são:

* 8 bits;
* 16 bits;
* 32 bits;
* 64 bits.

**Barramento de endereços:** permite ao processador selecionar o endereço (posição) de memória ou de E/S que deseja acessar. As larguras mais comuns para esse barramento são:

* 20 bits (220 = 1 Mbyte): 8086, 8088
* 24 bits (224 = 16 Mbytes): 286
* 32 bits (232 = 4 Gbytes): 386, 486, Pentium

**Barramento de controle:** veicula sinais de controle entre o processador e os demais dispositivos, dentre os quais:

* Operação de escrita ou leitura (R/): indica se na operação atual o processador deseja ler ou escrever o dado presente do barramento de dados no endereço indicado pelo barramento de endereços.
* Operação em memória ou dispositivo de E/S (IO/): indica se na operação atual o processador deseja acessar um endereço (indicado pelo barramento de endereços) em memória ou em um dispositivo de entrada/saída.
* Sinalização de interrupções de hardware.

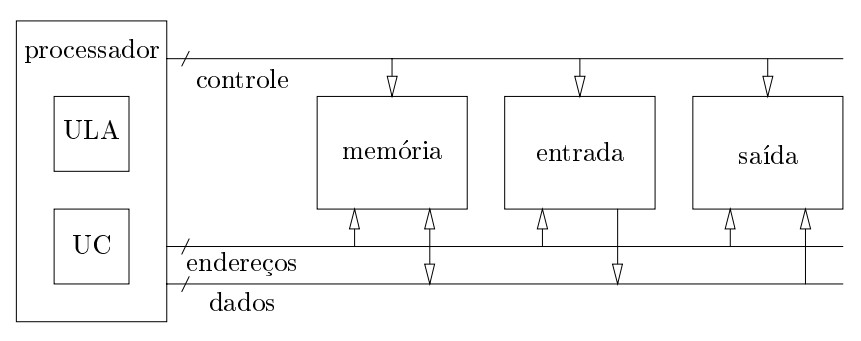


Figura : Tipos de barramentos.

## Sistemas de memória em um computador

**Memória principal**

* Armazena instruções e dados usados pelo processador;
* Memória mais rápida do sistema de memória da máquina;
* Tecnologia à base de semicondutores;
* Custo por bit bastante elevado.

**Memória de massa**

* Também chamada de auxiliar ou secundária, armazena grande quantidade de dados;
* Mais lenta que a memória principal;
* Memória não volátil;
* Custo por bit menor;

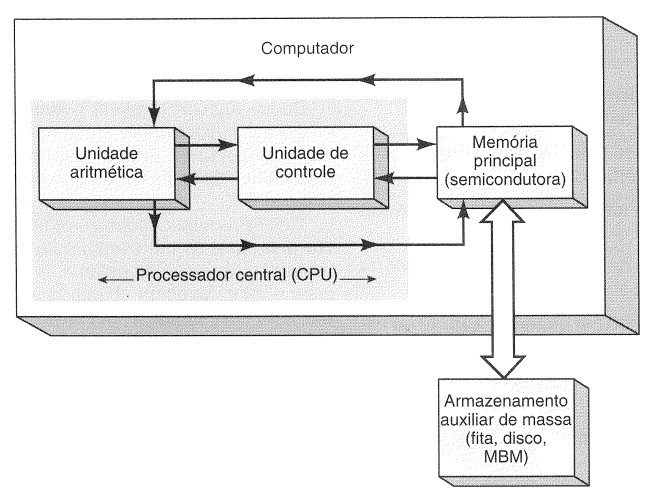


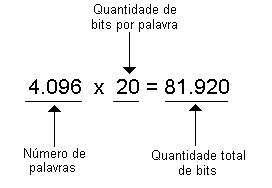
Figura : Sistemas de memória em um computador (Tocci).

Os elementos básicos de memória, com a capacidade de armazenamento de 1 bit de informação, são os *flip-flops*. Com estes elementos podem ser construídos os registradores de deslocamento, capazes de armazenar *informação estruturada*, composta por dados ou instruções, onde tais *informações* podem ser recebidas ou transferidas para outros dispositivos de armazenamento.

As memórias a semicondutor são usadas como memória principal de um computador, onde é exigida uma velocidade de operação bastante alta. A memória principal é a parte integrante do próprio computador e sem ela o computador simplesmente não funciona. Pode ser também denominada memória residente ou memória interna, estando em constante comunicação com o processador enquanto as instruções básicas estiverem sendo executadas.

### Capacidade de uma memória

É muito importante especificar quantos bits podem ser armazenados em uma determinada memória. Como exemplo, uma memória pode armazenar 4.096 palavras de 20 bits. Isto representa uma capacidade total de armazenamento de 81.920 bits (4.096 x 20), onde 4.096 é o número de palavras e 20 é a quantidade de bits por palavra.



Comumente se representa o número de palavras da memória como múltiplo de 1.024, sendo comum a designação 1K (1 kilo) para representar 1.024 bits, que é igual a 210. Por exemplo, uma memória que tenha uma capacidade de armazenamento de 8K x 20 é na verdade uma memória de 8.192 x 20. Memórias de grande capacidade de armazenamento utiliza a designação 1M (1 mega), que representa 220 que é igual a 1.048.576 bits. Dessa forma uma memória com capacidade de 2M x 8, possui uma capacidade de 2.097.152 x 8.

**Exemplos:**

1) Um chip de memória é especificado tendo a capacidade de 4K x 8. Quantas palavras podem ser armazenadas nesse chip? Qual é o tamanho da palavra? Quantos bits no total esse chip pode armazenar?

*Solução:*

4K = 4 x 1.024 = 4.096 palavras

Cada palavra tem 8 bits ou l byte

O número total de bits é 32.768 (4.096 x 8)

2) Qual das memórias armazena mais bits?

a) 2M x 8

b) 1M x 16

*Solução:*

a) 2 x 1.048.576 x 8 = 16.777.216

b) 1 x 1.048.576 x 16 = 16.777.216

As duas memórias têm a mesma capacidade de armazenamento de bits.

### Conceitos sobre dispositivos de memória

Nesta seção serão apresentados alguns conceitos referentes aos dispositivos de memória. Um chip de memória possui entradas de dados para serem gravados, saídas de dados lidos, entradas de endereço para selecionar a posição correta da memória e entradas *enable*, que são, em geral, *chip select* (CS) e *read/write* (R/).

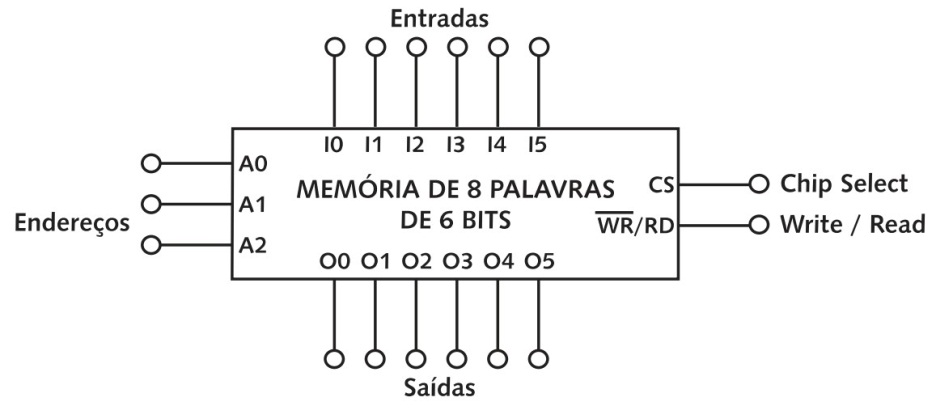


Figura : Chip de memória de 8 palavras de 6 bits.

**Memória volátil:** Qualquer tipo de memória que necessite de energia elétrica para reter informações armazenadas é denominada memória volátil. Quando a energia elétrica for interrompida toda a informação armazenada será perdida. Muitas memórias a semicondutor são voláteis, enquanto que todas as memórias magnéticas são não voláteis.

**Memória de acesso randômico (RAM):** Memória onde a localização física real de uma palavra de memória não tem efeito sobre o tempo que se leva para ler ou escrever nesta posição, isto é, o tempo de acesso é constante para qualquer endereço da memória. A grande maioria das memórias e semicondutor são randômicas.

**Memória de Leitura/Escrita (RWM):** Qualquer memória que possa ser lida ou escrita com facilidade.

**Memória de leitura (ROM):** São memórias a semicondutor onde a taxa de operações de leitura é infinitamente maior do que a escrita. Tecnicamente uma ROM pode ser gravada ou programada apenas uma vez, o que na maioria das vezes é feita na fábrica. Depois disso, a informação somente poderá ser lida.

**Dispositivos de memória estática:** Nessas memórias as informações armazenadas permanecerão armazenadas enquanto houver energia elétrica aplicada à memória, sem que haja necessidade da informação ser rescrita periodicamente na memória.

**Dispositivos de memória dinâmica:** Nessas memórias as informações armazenadas não permanecerão armazenadas, mesmo com a presença da energia elétrica necessária para alimentar o circuito, a não ser que as informações sejam rescritas periodicamente na memória com determinada frequência. Esta operação é denominada *refresh* (recarga).

**Tempo de acesso:** O tempo de acesso mede a velocidade de acesso à memória. É a quantidade de tempo necessária à efetivação de uma operação de leitura ou, mais especificamente, é o tempo decorrido entre o momento da recepção pela memória de um novo endereço, e o instante em que a informação daquele endereço fica disponível.

### Princípio de operação de uma memória

As memórias de uma forma geral, apesar dos diversos tipos de implementação, apresentam algumas operações básicas:

**a)** selecionar o endereço que está sendo acessado, para leitura ou escrita;

**b)** selecionar a operação a ser realizada (leitura ou escrita);

**c)** fornecer os dados de entrada para operação de escrita;

**d)** manter estáveis as informações de saída da memória. resultantes de uma operação de escrita, durante um tempo determinado;

**e)** habilitar ou desabilitar a memória, para fazê-la responder ou não ao endereço na entrada e ao comando leitura/escrita.

A Figura 80 mostra como exemplo o diagrama simplificado de uma memória 32x4 (32 palavras de 4 bits), isto é, cada palavra tem o tamanho de 4 bits.

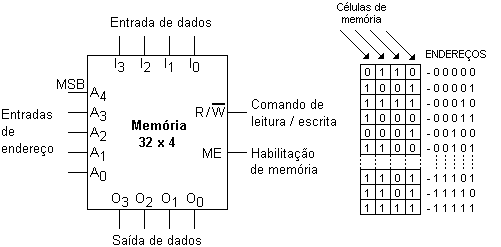


Figura : Memória 32x4.

Como o tamanho da palavra é de 4 bits, existe nesta memória 4 linhas de entrada de dados e 4 linhas de saída de dados.

Durante a operação de escrita, os dados a serem armazenados na memória devem ser colocados nas linhas de entrada de dados (I0 a I3) e durante a operação de leitura, a palavra lida aparece nas linhas de saída de dados (O0 a O3).

A memória mostrada no exemplo acima possui 32 posições diferentes de armazenamento e portanto, 32 endereços diferentes, começando por 00000 e terminando em 11111 (0 a 31 decimal).

Desta forma essa memória deve ter 5 entradas de endereçamento (25 = 32). Em geral são necessárias N linhas de entrada de endereço para uma memória com capacidade de 2N palavras.

Voltando ao exemplo da Figura 80, observa-se que cada posição possui 4 células de memória que armazenam 0s e 1s, formando assim as palavras de memória. Dessa forma, no endereço 00001 está armazenada a palavra 1001; no endereço 00010 está armazenada a palavra 1111 e assim por diante.

O comando de leitura/escrita determina qual das operações a memória deverá executar. Alguns sistemas utilizam linhas separadas para leitura e escrita. Quando se usa uma única linha as seguintes condições são respeitadas:



A Figura 81 ilustra o processo de leitura e escrita na memória 32 x 4, onde em A ocorre a escrita de uma palavra de dados 0001 na posição da memória cujo endereço é 00100, enquanto que em B ocorre a leitura de uma palavra de dados 1101 na posição de memória, cujo endereço é 11110.

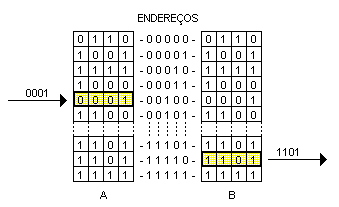


Figura : Processo de leitura e escrita em memória.

**Exemplos:**

**a)** Descreva as condições de cada uma das linhas de entrada e saída, quando deve ser lido o conteúdo do endereço 11110.

*Solução:*

Entradas de endereço: 11110;

Entradas de dados: não utilizadas;

R/: nível alto;

Habilitação da memória: nível alto;

Saídas de dados: 1101.

**b)** Descreva as condições de cada uma das linhas de entrada e saída, quando deve ser escrita a palavra 1001 no endereço 10001.

*Solução:*

Entradas de endereço: 10001;

Entradas de dados: 1001;

R/: nível baixo;

Habilitação da memória: nível alto;

Saídas de dados: não utilizadas (normalmente em alta impedância).

**c)** Uma memória tem uma capacidade de 8K x 4. Pergunta-se:

a) Quantas linhas para entrada e saída de dados esta memória deve ter?

b) Quantas linhas de endereço deve ter?

c) Qual é a sua capacidade total em bytes?

*Solução:*

a) 4 linhas de entrada e 4 linhas de saída, pois o tamanho da palavra é de 4 bits.

b) A memória armazena 8.192 palavras, devendo então existir 8.192 endereços de memória. Como 8.192 = 213 , a mesma precisa de 13 bits para especificar qualquer um dos seus endereços.

c) Cada byte corresponde a 8 bits. Logo, tem 4.096 bytes de capacidade.

## Classificação das memórias semicondutoras

As memórias semicondutoras podem ser classificadas em duas famílias: ROM e RAM.

As memórias ROM podem ser divididas em:

* ROM (ROM programada por máscara);
* PROM (ROM programável pelo usuário);
* EPROM (ROM programável/apagável);
* EEPROM (ROM programável/apagável eletricamente).

As memórias RAM podem ser divididas em:

* SRAM (RAM estática);
* DRAM (RAM dinâmica).

### Memórias de leitura - ROM

As memórias ROM (*Read-Only Memory*) são dispositivos de armazenamento de dados cujo conteúdo é fixo (não pode ser mudado facilmente) e não-volátil (mantido mesmo quando o dispositivo é desligado). As memórias ROM são normalmente empregadas para armazenar informações permanentes ou de longa duração, como programas de inicialização (boot) de sistemas ou tabelas de dados fixos.

As aplicações destas memórias se baseiam em armazenamento permanente de valores binários, como em Firmwares, tabelas de dados, dados de configuração (BIOS de computadores), etc.

#### Estrutura básica

Um chip de memória ROM pode ser visto como um circuito combinacional, cuja entrada é o endereço interno a acessar e sua saída é o valor (fixo) armazenado naquele endereço. A entrada de endereços possui *n* vias para 2n endereços distintos, enquanto que a saída de dados pode possuir *m* bits. O exemplo da Figura 82 indica uma memória ROM com 256 posições de 8 bits cada, somando um total de 2048 bits armazenados.

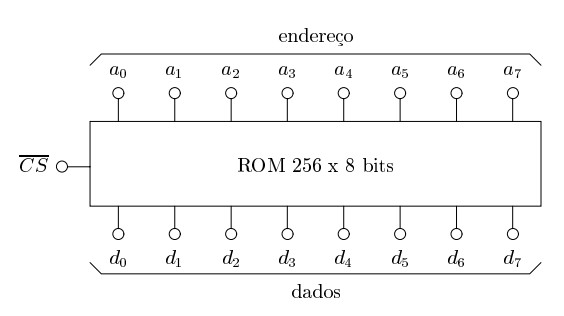


Figura : chip de memória de 256 posições de 8 bits.

No exemplo acima, a entrada CS (*Chip Select*) controla o estado das saídas do chip. Caso esta entrada seja ativada, os dados do endereço selecionado são colocados na saída da memória. Caso contrário, a saída é desconectada da estrutura interna da memória e mantida em um estado de alta impedância. Esse mecanismo é necessário porque todas as saídas de memórias e dispositivos de E/S são conectadas junto ao barramento de dados do computador. As entradas *chip select* das memórias do sistema devem ser adequadamente controladas para garantir que em um determinado instante apenas o conteúdo de uma posição de memória seja colocado no barramento de dados, sem a possibilidade de conflitos. A Figura 83 ilustra esse mecanismo:

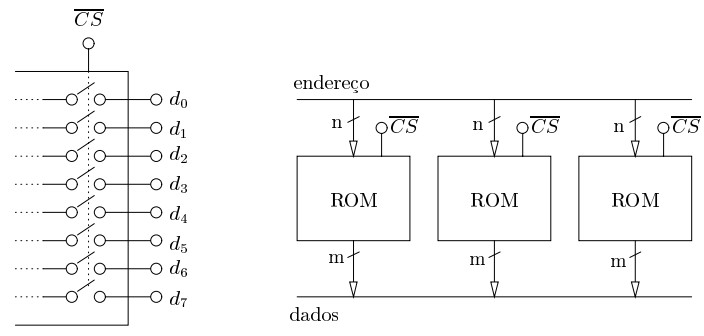


Figura : Funcionamento do bit *chip select* (CS).

#### Arquitetura interna

A Figura 84 mostra a arquitetura interna de uma ROM 16 x 8, onde verifica-se a existência de dois decodificadores, um serve para seleção de linha e outro para seleção de coluna.

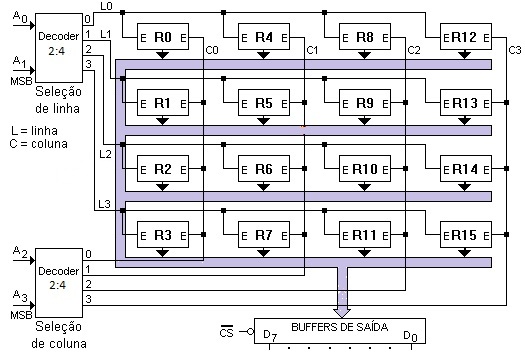


Figura : Arquitetura interna de uma memória ROM.

Observa-se no exemplo acima a existência de 16 registradores (R0 a R15) que armazenam todas as informações na ROM. Cada registrador tem um número de células de memória igual 8 bits neste caso, além de 2 entradas para habilitação (E), ativas em nível alto, sendo uma interligada à linha e outra interligada à coluna.

Os registradores são arranjados na forma de uma matriz quadrada (comum na maioria dos chips), identificáveis através das linhas e colunas por eles ocupadas na matriz.

Por exemplo, o registrador 7 está na linha 3 e coluna 1; o registrador 11 está na linha 3 e coluna 2 e assim por diante. O código de endereço aplicado, A3A2A1A0, determina qual dos registradores da matriz deve ser habilitado.

Seguem alguns exemplos:

**a)** Qual dos registradores será habilitado para uma entrada de endereço 1110?

*Solução:*

A3A2A1A0 = 1110

A3A2 = 11: coluna 3

A1A0 = 10: linha 2

Então o registrador 14 é devidamente habilitado

**b)** Qual é o endereço que habilitará o registrador 6?

*Solução:*

As entradas de habilitação deste registrador são conectadas aos sinais da linha 2 e coluna 1 respectivamente.

Logo, A3A2A1A0 = 0110

O registrador habilitado pelas entradas de endereço deverá colocar seus dados no barramento de dados, que serão entregues ao *buffer de saída* desde que  esteja em nível baixo, caso contrário as saídas do buffer estarão em alta impedância e as linhas D0 a D7 estarão em flutuação.

#### Ampliação da capacidade de uma ROM

Pode se considerar uma ROM organizada na forma 8x1, conforme ilustra a Figura 85. Onde A, B e C são entradas de endereçamento e S é a saída para leitura de dados. Cada palavra de memória, neste exemplo, é capaz de armazenar 1 bit.

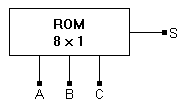


Figura : ROM 8x1 para ser usada numa expansão de memória.

Se forem disponíveis dois destes chips de capacidade 8x1, pode-se ampliar sua capacidade para 16x1 (**ampliação do número de palavras**), conforme ilustra a Figura 86. As entradas de endereço ABCD dividem-se em duas partes: a mais significativa, no caso A, é aplicada ao multiplexador, enquanto BCD (menos significativa) é aplicada às ROMs.

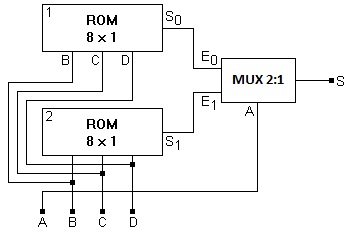


Figura : Expansão de memória para 16x1, com dois chips 8x1.

De acordo com o valor assumido por A, o multiplexador selecionará qual saída, S0 ou S1 será transportada à saída S do multiplexador, através das entradas E0 ou E1 do multiplexador.

Como outro exemplo, é apresentada a ampliação da capacidade de uma ROM, a partir de 2 ROMs organizadas na forma 512x8, com 4096 bits cada. Com isto será possível ampliar sua capacidade de tal forma a organizá-la na forma 1024x8, totalizando 8.192 bits.

Deve-se então dispor de 8 multiplexadores, uma vez que, o tamanho da palavra para essa ROM é de 8 bits, como ilustra a Figura 87.

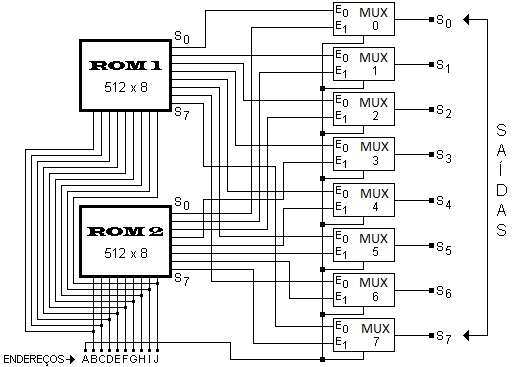


Figura : Ampliação de memória para 1024x8, com chips 512x8.

Os endereços de A até J admitem 1024 possibilidades (210). A é o endereço mais significativo, conectado aos MUX (0 a 7), que permitirá obter na saída as localidades correspondentes às ROMs 1 ou 2.

As entradas E0 de cada multiplexador estão ligadas na ROM 1, enquanto que as entradas E1 estão ligadas na ROM 2. A ROM 1 abrange as localidades 0 a 511, que correspondem aos endereços 0000000000 a 0111111111, enquanto que a ROM 2 abrange as localidades 512 a 1023, que correspondem aos endereços 1000000000 a 1111111111.

Quando A = 0, nas saídas dos multiplexadores estarão os valores da ROM 1, enquanto que quando A = 1, nas mesmas saídas estarão os valores da ROM 2. O endereço 0100011001, por exemplo, indica a localidade 281, correspondente a ROM 1, enquanto que o endereço 1000101011 indica a localidade 43, correspondente a ROM 2.

**Exercício:** Utilize os dois chips de memória ROM de 512x8 do exemplo anterior e construa um dispositivo de memória de 512x16 (**ampliação do tamanho de palavra**).

#### Tecnologias de fabricação

Existem diversas tecnologias de fabricação de memórias ROM, nesta seção serão destacadas as mais conhecidas.

**ROM programada por máscara (MROM)**

Os dados são armazenados durante a fabricação do circuito e não podem mais ser alterados. É uma tecnologia rápida e barata, usada na produção de sistemas em larga escala industrial.

A ROM programada por máscara tem suas posições de memória escritas (programadas) pelo fabricante de acordo com as especificações do cliente (usuário). Um negativo fotográfico, denominado **máscara**, é utilizado para especificar as conexões elétricas do chip.

Esta memória é viável economicamente se for produzido um número muito grande de ROMs com a mesma máscara. A grande desvantagem é que os dados não podem ser apagados e reprogramados. Além disso, cada modificação no projeto de uma memória deste tipo exige que seja feita uma nova máscara para nova gravação.

A Figura 88 abaixo mostra a estrutura interna de uma MROM de pequena capacidade, constituída de 16 células arranjadas em 4 linhas e 4 colunas de células, construídas com transistores bipolares. Dessa forma, cada célula é constituída por um transistor bipolar. Cada linha constitui um registrador de 4 bits. Uma conexão aberta na base do transistor armazena um bit “0” enquanto que, uma conexão da base à saída do decodificador armazena um bit “1”. A condição de cada uma das conexões da base é controlada através de uma máscara fotográfica durante o processo de fabricação, de acordo com os dados fornecidos pelo cliente.

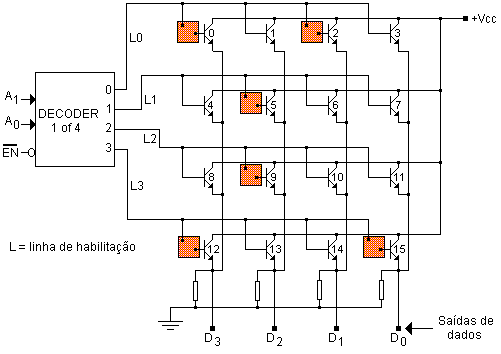


Figura : MROM de 16 células

**PROM (Programable ROM)**

Pode ser programada uma única vez pelo usuário, através da queima de micro-fusíveis no interior do chip. A Figura 89 mostra a estrutura interna de uma PROM construída com transistores MOSFET, onde uma conexão aberta armazena um bit 0 e uma conexão fechada, um bit 1.

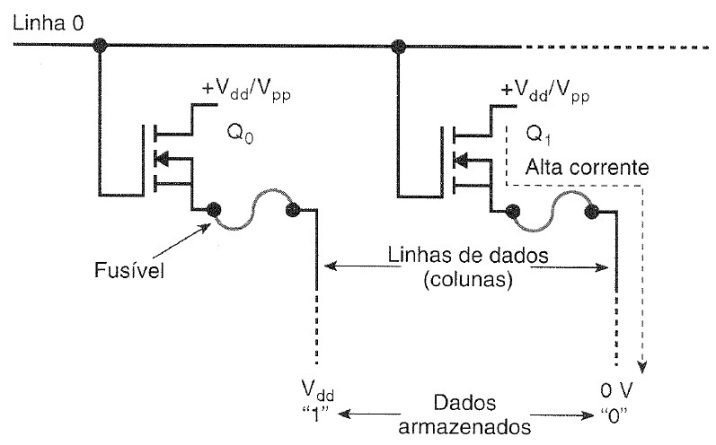


Figura : Estrutura básica de uma célula de memória PROM.

A PROM tem um custo menor do que uma MROM. O usuário através de um gravador próprio introduz os dados. Uma vez gravados os dados, não pode ser alterada.

**EPROM (Erasable PROM)**

Esta memória pode ser programada diversas vezes através da aplicação de tensão elevada em pontos especiais. Para apagar os dados, devem ser inseridos raios ultravioletas em um local específico do chip de memória. O processo de programação é realizado por um dispositivo microprocessado (gravador de EPROM). A Figura 90 mostra uma EPROM de 4kx8, onde as células de memória são construídas de transistores MOS.

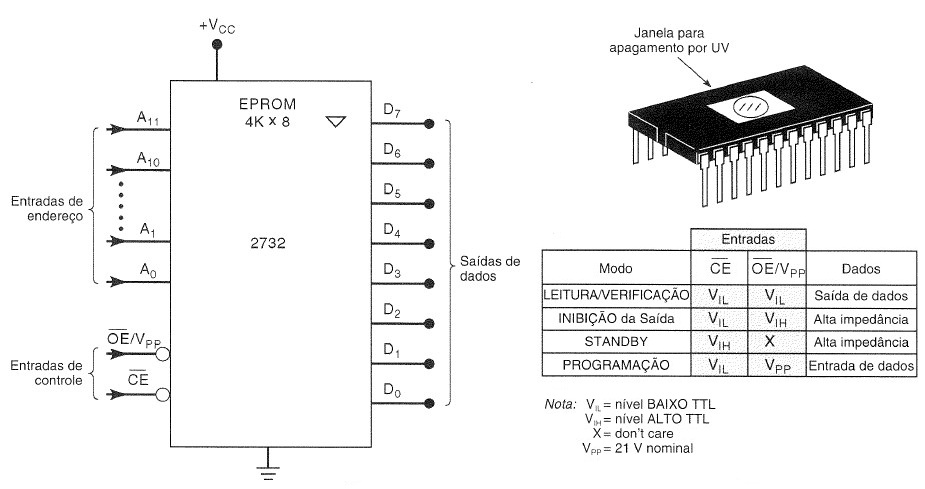


Figura : Chip de memória de uma EPROM.

**EEPROM (Eletrically Erasable PROM)**

Pode ser escrita e apagada (lentamente) através de sinais elétricos. É bastante usada para armazenar configurações de hardware em computadores (BIOS). Este tipo de memória soluciona o problema das EPROM, que precisam ser retiradas do soquete para ser apagadas. A desvantagem das EEPROM é a necessidade de um conversor externo para gerar as tensões de programação (21V e -21V), no entanto, os dispositivos mais recentes já incorporam este conversor. As células de memória das EEPROM são também construídas com transistores MOSFET, porém com uma estrutura diferente.

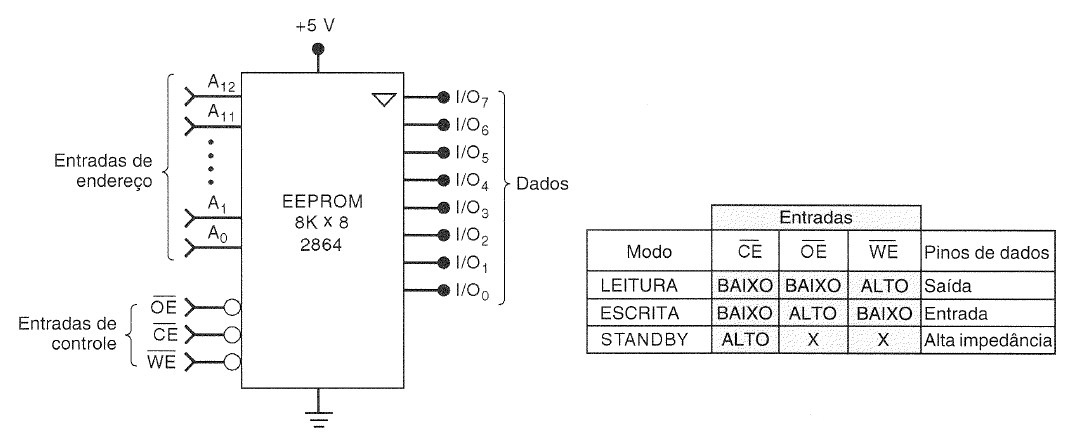


Figura : Chip de memória EEPROM.

Modo de Operação:

* : Habilita ou desabilita o chip;
* : Habilita ou desabilita os buffers de saída;
* : Seleciona o modo de escrita ou leitura da EEPROM.

**Memória *Flash***

Uma célula de memória *flash* se assemelha às células de memória EPROM, sendo mais simples do que as células de memória EEPROM. As memórias *flash* possuem um tempo de leitura/escrita pequeno, comparado com os outros tipos de memória. As memórias *flash* podem ser apagadas e regravadas sem ser retiradas do circuito. Outra vantagem destas memórias é o baixo custo e complexidade por bit. A Figura 92 apresenta um CI de memória *flash* típico, com capacidade 32k x 8, cujos modos de operação são os mesmos do chip de memória EEPROM apresentado. A Figura 93 apresenta um comparativo entre os diferentes tipos de memória ROM apresentados.

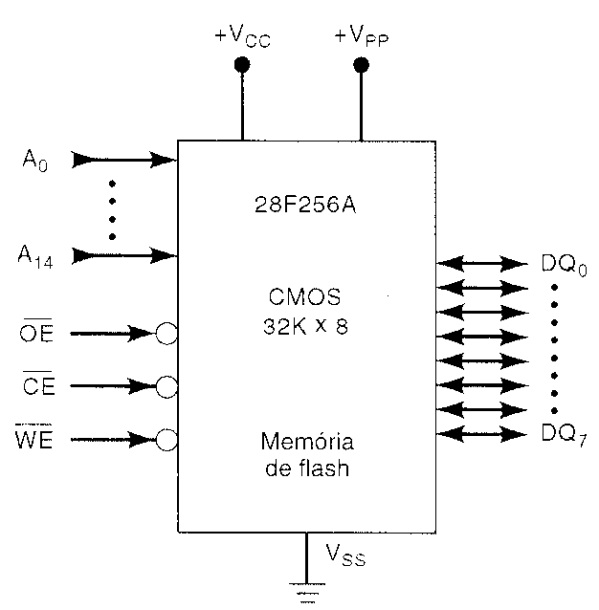


Figura : Chip de memória *flash.*

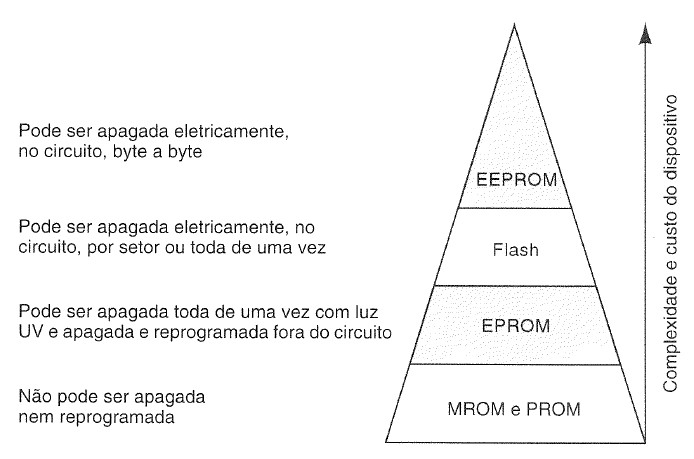


Figura : Comparativo entre os diferentes tipos de memória.

### Memórias de Acesso Randômico (RAM)

As memórias RAM, também conhecidas como memórias de escrita e leitura (RWM) são utilizadas para armazenamento temporário de dados. O termo RAM (Random Acess Memory) é usado para designar uma memória que tem igual facilidade de acesso para todos os endereços, sendo também chamadas de memória de acesso direto por esse motivo.

As memórias RAM permitem armazenar dados e modifica-los continuamente, servindo como uma área de trabalho para a execução de programas (variáveis, dados temporários, pilhas de execução). O conteúdo de uma RAM é normalmente volátil, ou seja, desaparece se o circuito for desligado.

#### Tecnologia de fabricação

* RAM estáticas (SRAM): A célula de armazenamento da informação binária é um *flip-flop*, que armazena 1 bit; ele permanece nesse estado até ser explicitamente alterado.
* RAM dinâmicas (DRAM): A célula básica de armazenamento da informação binária é um capacitor que armazena 1 bit. Este capacitor precisa ser periodicamente recarregado (*refresh*) para permanecer em 1, caso armazene um bit ativo.

As RAM estáticas são mais rápidas e simples de construir que as dinâmicas, pois não precisam de circuitos auxiliares para percorrer toda a memória e efetuar o *refresh* dos capacitores carregados (bits ativos). Todavia as RAM dinâmicas são mais baratas e muito mais compactas, além de consumirem menos energia. Desta forma, as memórias RAM estáticas são empregadas em sistemas que necessitam de pouca memória mas com alta velocidade, como microcontroladores, processamento de sinais em tempo real, memórias de cache e de vídeo, etc. As RAM dinâmicas são normalmente empregadas quando se tenta otimizar o volume de memória e o baixo consumo, como no caso da memória principal dos computadores.

**Arquitetura da SRAM**

A Figura 94 mostra a arquitetura de uma SRAM que armazena 64 palavras de 4 bits cada (64x4).

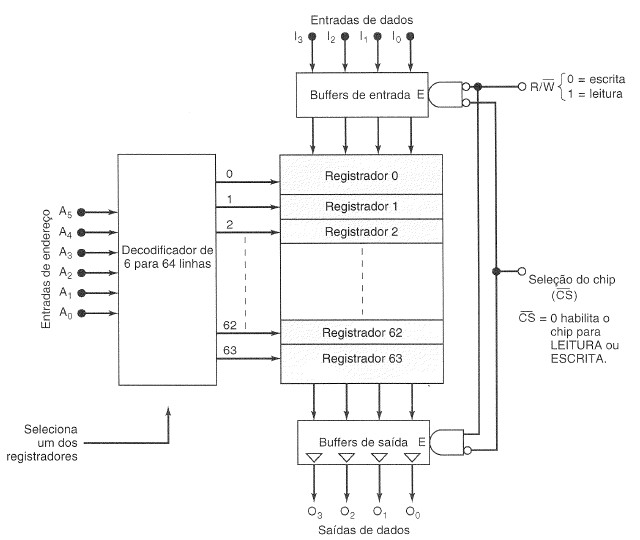


Figura : Arquitetura interna de uma SRAM.

**Arquitetura da DRAM**

As RAMs dinâmicas utilizam tecnologia CMOS, o que confere alta capacidade de armazenamento, baixo custo de energia e velocidade de operação moderada. O armazenamento dos dados é realizado através de cargas em microcapacitores MOS. Por esse motivo as RAMs dinâmicas necessitam recarregar periodicamente as células de memória (operação de *refresh*). Para esta operação é utilizado um circuito externo, chamado de controlador de RAM dinâmica. Na Figura 95 é apresentada a estrutura de uma célula de memória de uma DRAM.

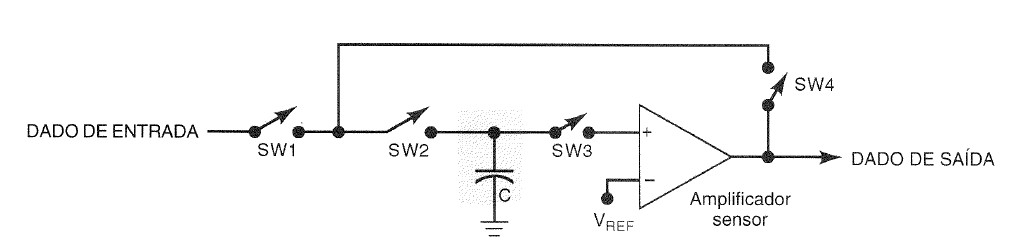


Figura : Célula de memória de uma DRAM.

Na Figura 96 é apresentado um diagrama de funcionamento de uma memória DRAM, que possui multiplexação de endereços, sinal de seleção de linha (RAS – strobe do endereço de linha) e sinal de seleção de colunas (CAS – strobe do endereço de coluna). Esta memória possui registradores de endereço de linha e de coluna, que armazenam qual endereço será acessado em determinado momento. O multiplexador escolhe 8 dentre os bits A0 – A15 para enviar às entradas de endereço da DRAM. Quando a entrada de seleção MUX\* = 0, são enviados os bits referentes ao endereço da linha (A8 – A15). Este sinal deve ser sincronizado com os bits RAS e CAS. Quando RAS = 0, os registradores de endereço de linha armazenam os 8 bits Ai, quando CAS = 0, os registradores de endereço de coluna armazenam os 8 bits Ai.

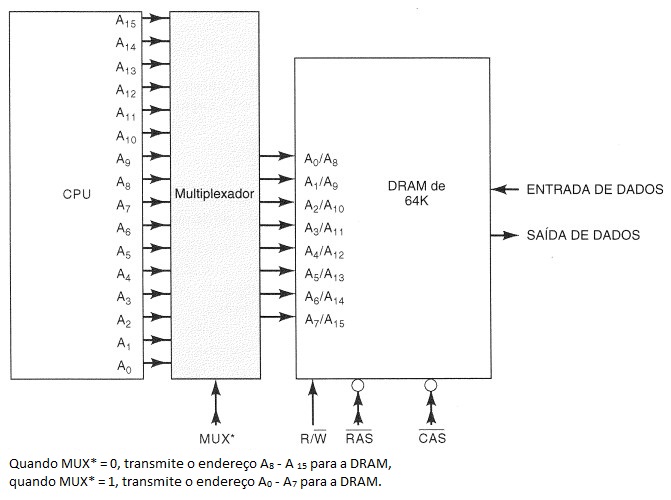


Figura : Estrutura básica de uma DRAM.

**Operação de *refresh* de uma DRAM**

Como cada célula de memória de uma DRAM é constituída de um capacitor, estes precisam ser recarregados periodicamente, para não perder seus dados por motivo de descarga dos capacitores. Para esta operação existe um dispositivo controlador de memória dinâmica, que periodicamente executa esta operação de *refresh*.

### Bancos de memória

1. Expansão da capacidade de endereçamento: Como exemplo pode ser citada a necessidade de uma memória com capacidade 32 x 4 (32 palavras de 4 bits), mas dispomos apenas de chips 16 x 4.

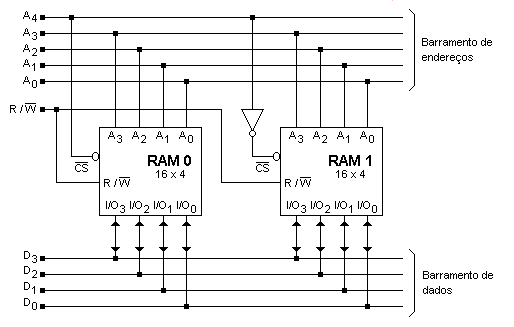


Figura : Expansão de número de palavras de memória RAM.

O arranjo acima ilustrado, normalmente denominado *módulo de memória*, abrange os endereços:

RAM 0 - 00000 a 01111

RAM 1 - 10000 a 11111

TOTAL - 00000 a 11111 (32 palavras = 25 = 5 linhas de endereços)

Observa-se que o endereço A4 tem por função selecionar uma das duas RAMs através da entrada  a partir dos níveis 1 ou 0.

Quando  = 1, habilita a RAM 0, desabilitando a RAM 1. Desta forma, a faixa de endereços para as RAMs 0 e 1 ficará por conta das entradas A3A2A1A0.

1. Expansão da capacidade de armazenamento: Supondo que se dispõem de chips de memória RAM 16 x 4, mas precisa-se apenas dobrar o tamanho da palavra, mantendo a mesma quantidade de palavras.

Isto resultaria uma organização 16 x 8.

Neste caso cada chip seria utilizado para armazenar a metade de bits de cada palavra, como ilustra a Figura 98.

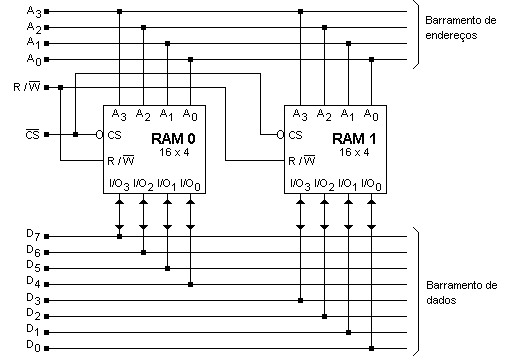


Figura : Expansão da capacidade de armazenamento.

Para o arranjo acima, a faixa de endereços compreende: 0000 a 1111, que corresponde a 16 palavras.

A RAM 0 armazena os 4 bits da mais significativos da palavra, enquanto que a RAM 1 armazena os 4 bits menos significativos da mesma palavra.

As entradas R/ e  controlam a operação escrita/leitura.

Para leitura R/= 1 e  = 0; para escrita R/= 1 e = 0.

O barramento de dados funciona como entrada e saída, podendo ser interligado a qualquer outro dispositivo, como por exemplo, um processador.

# Circuitos Schmitt-Trigger e Multivibradores

No Capítulo 5, foram estudados os *flip-flops*, ou multivibradores biestáveis. Estes circuitos se caracterizam por possuírem dois estados estáveis, podendo assumir, em sua saída, o nível lógico ‘0’ ou ‘1’. Neste capítulo serão estudados os circuitos multivibradores **monoestáveis** e **astáveis**, que se caracterizam por possuírem, respectivamente, somente um estado estável e nenhum estado estável. Os circuitos multivibradores monoestáveis e astáveis possuem várias aplicações, como temporização e geração de *clock*.

Outros circuitos que podem ser utilizados em aplicações com temporização e geração de *clock* são os circuitos com entradas disparadoras ou *Schimitt-Trigger*. Estes circuitos apresentam características importantes de entrada-saída, como será visto na próxima seção.

## Circuitos *Schmitt-Trigger*

Em um circuito *Schimitt-Trigger*, a função de transferência entre sua entrada e sua saída apresenta uma característica de memória, devido à histerese que este circuito apresenta. A histerese pode ser definida como um retardo da entrada em relação a mudanças na saída, como ilustra a Figura 99. Ou seja, a saída *vo* apresenta valor ativo alto somente quando a entrada assumir uma tensão acima de *v2* e assume valor ativo baixo somente quando a entrada assumir uma tensão abaixo de *v1­*.

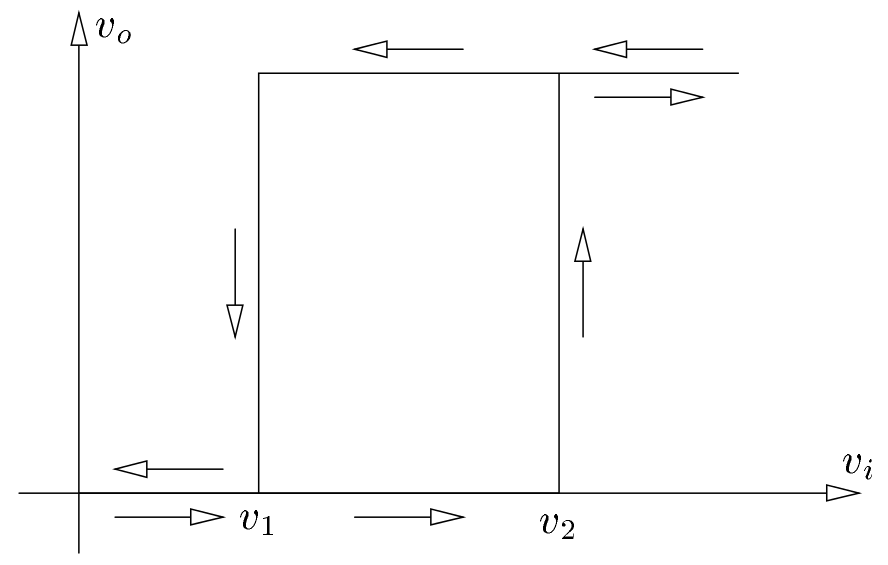


Figura : Histerese em um circuito *Schmitt-Trigger*.

Um circuito com a característica de histerese da figura anterior pode apresentar a curva de resposta apresentada na Figura 100.

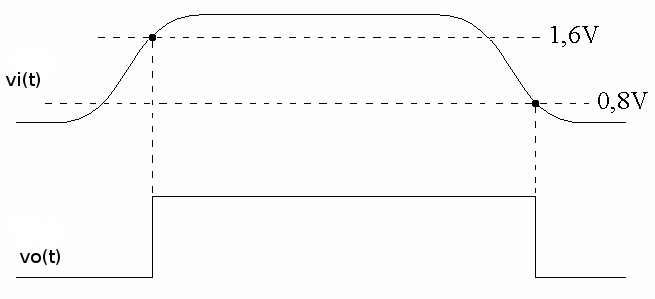


Figura : Curva de resposta em um circuito *Schmitt-Trigger*.

Como exemplo será analisado um circuito inversor. A forma de onda da saída de um inversor comum é apresentada na Figura 101, onde pode-se perceber que no intervalo de transição entre os níveis alto e baixo o sinal de saída apresenta uma oscilação indesejada.

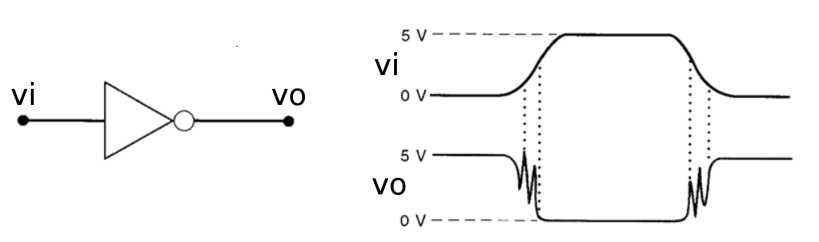


Figura : Inversor comum.

O inversor *Schmitt-Trigger*, por sua vez, apresenta a forma de onda da saída como na Figura 102, onde pode-se perceber que o seu valor não apresenta as oscilações indesejadas.

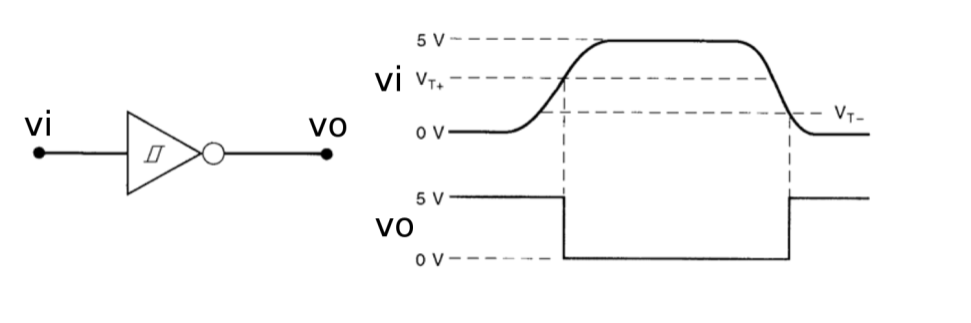


Figura : Inversor disparador.

Este exemplo mostra que estes dispositivos podem ser utilizados para o acoplamento entre circuitos lentos e circuitos rápidos. Em circuitos lentos, uma transição de nível pode demorar bastante para ocorrer e, sendo conectados a circuitos *Schmitt-Trigger*, esta variação lenta é transformada em uma variação bem mais rápida. Da mesma forma, estes dispositivos podem ser utilizados como conformadores de pulso, como apresentado na Figura 103.

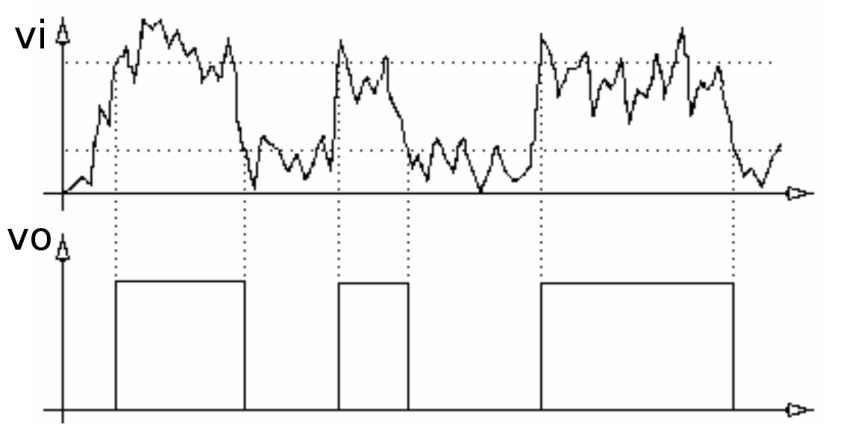


Figura : Conformador ou limitador de pulsos.

Outra aplicação para circuitos *Schmitt-Trigger* é trabalhar como oscilador de onda quadrada. Neste caso, devem ser conectados à sua entrada um resistor e um capacitor, que definem a frequência de oscilação deste sinal, respeitando as condições da Figura 104.

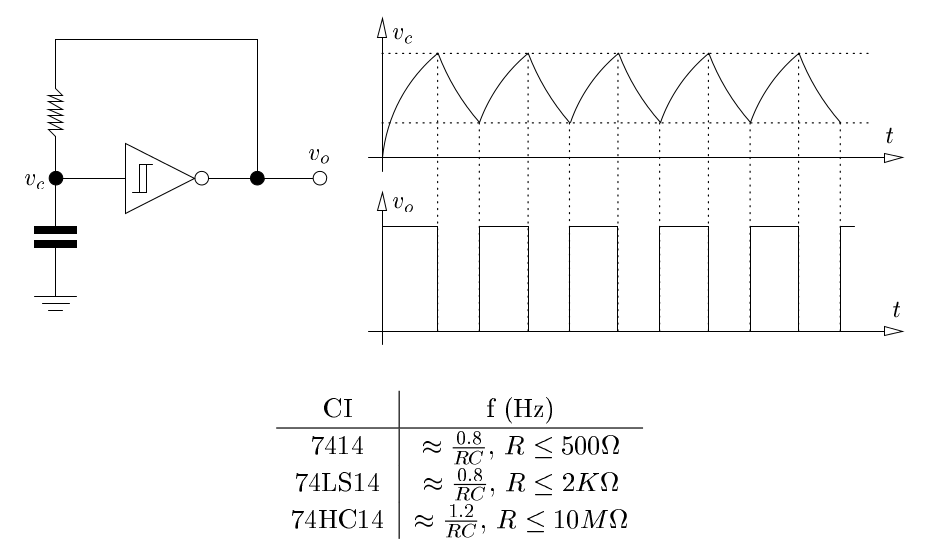


Figura : Oscilador com *Schmitt-Trigger*.

## Circuitos Multivibradores

No capítulo 5 foram estudados os *flip-flops*, que são circuitos multivibradores biestáveis, ou seja, com dois estados estáveis. Nesta seção serão estudados os circuitos multivibradores monoestáveis e os astáveis. Os monoestáveis se caracterizam por possuírem apenas um estado estável, enquanto que os astáveis possuem somente estados estáveis, como será mais bem visto nas próximas seções.

### Multivibrador Monoestável

O circuito multivibrador monoestável possui apenas um estado estável e um estado instável, denominado na literatura por estado *quase estável.* No funcionamento normal, o circuito permanece em seu estado estável e, quando aplicado um pulso, o circuito atinge o seu estado quase estável, permanecendo neste estado por um intervalo de tempo **tw**. O tempo **tw**é determinado por um circuito RC externo ao dispositivo, como mostra a Figura 105.

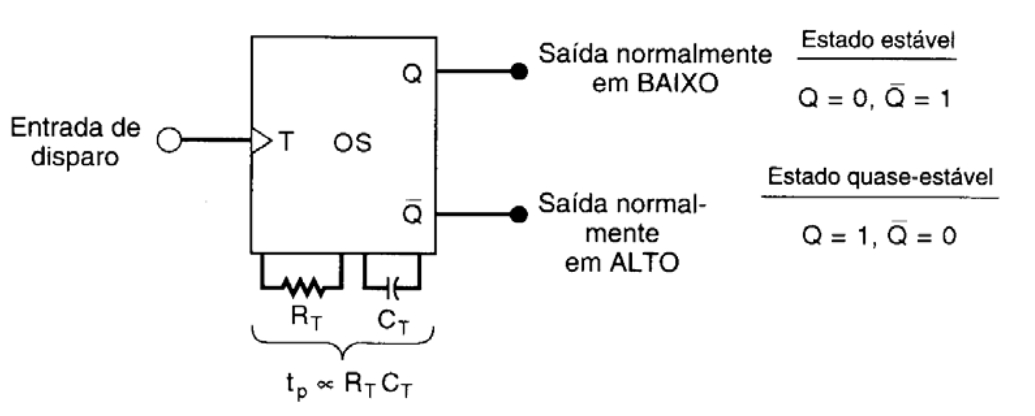


Figura : Circuito Multivibrador Monoestável.

A passagem do estado estável para o quase estável se realiza unicamente se o circuito for disparado (triggered) externamente, enquanto o retorno ao estado estável se dá automaticamente. O seu funcionamento é ilustrado na figura seguinte:

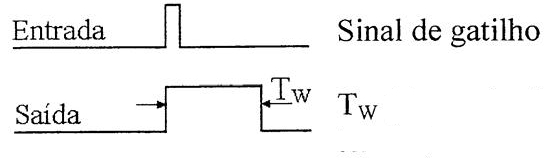


Figura : Disparo em um circuito Monoestável.

Existem dois tipos de multivibradores monoestáveis: os redisparáveis e os não-redisparáveis.

#### Não Redisparável

Para um circuito não redisparável, qualquer sinal de gatilho na sua entrada, durante um pulso quase estável **tw**, não traz nenhuma modificação na largura deste pulso, ou seja, é ignorado, como ilustrado na Figura 107.

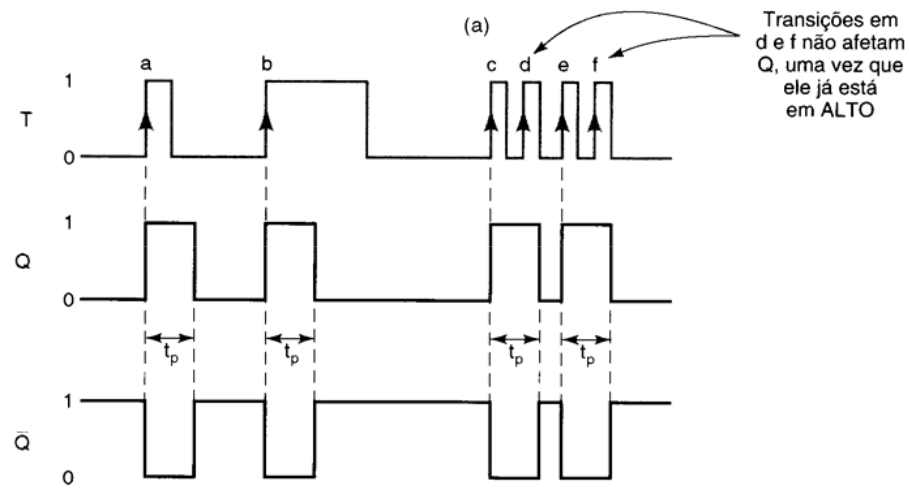


Figura : Monoestável não-redisparável.

#### Redisparável

Para um circuito redisparável, novos pulsos de disparo, mesmo durante o estado quase estável **tw**, incrementam a largura do pulso de saída, como mostra a Figura 108.

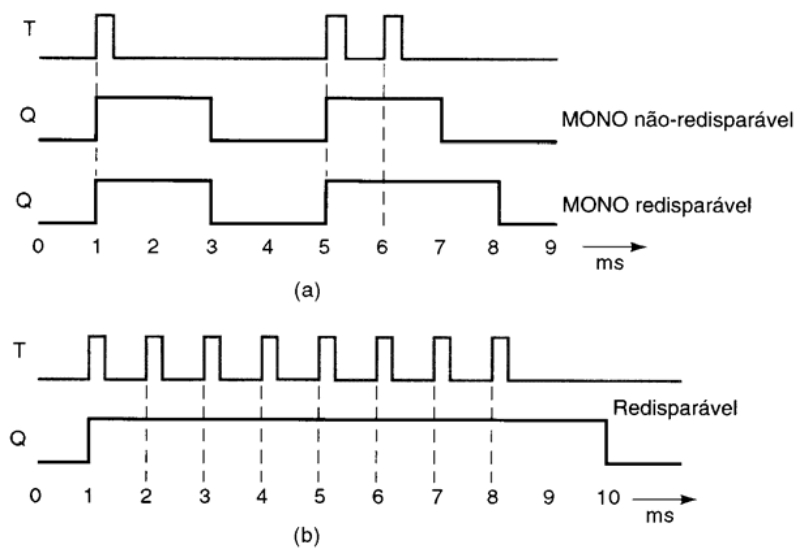


Figura : Monoestável redisparável.

#### Construção do circuito

Os circuitos monoestáveis podem ser utilizados para: gerar pulsos de largura fixa conhecida, aumentar ou diminuir o tamanho de pulsos, filtrar sinais de entrada com ruído, etc.

Comercialmente, o circuito que desempenha esta função é o TTL 74122 e o TTL 74123, que pode ser implementado como na Figura 109, onde A e B servem como pulsos de disparo.

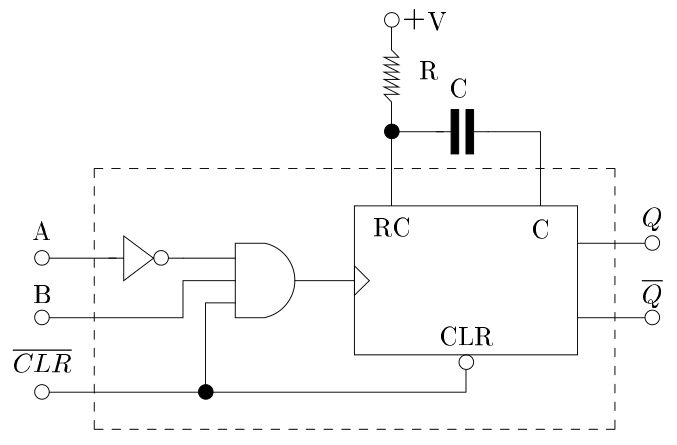


Figura : Implementação do TTL 74122.

Um circuito multivibrador monoestável também pode ser montado a partir do circuito integrado 555, como mostrado Figura 110. Neste caso, tw ≅ 1,1RC (respeitando R ≥ 1KΩ).

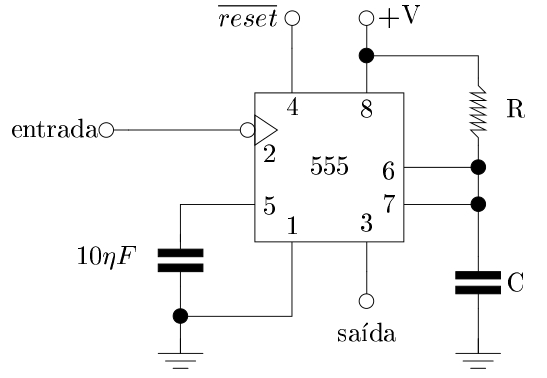


Figura : Multivibrador monoestável montado com CI 555.

### Multivibrador Astável

O circuito multivibrador astável se caracteriza por não possuir estados estáveis. Estes circuitos possuem dois estados e a sua saída oscila continuamente entre eles com intervalos bem definidos. São muito utilizados para a geração de onda quadrada em circuitos sincronizados por um sinal de *clock*. Uma maneira de montar um circuito astável é utilizando o CI 555, como apresentado na Figura 111.

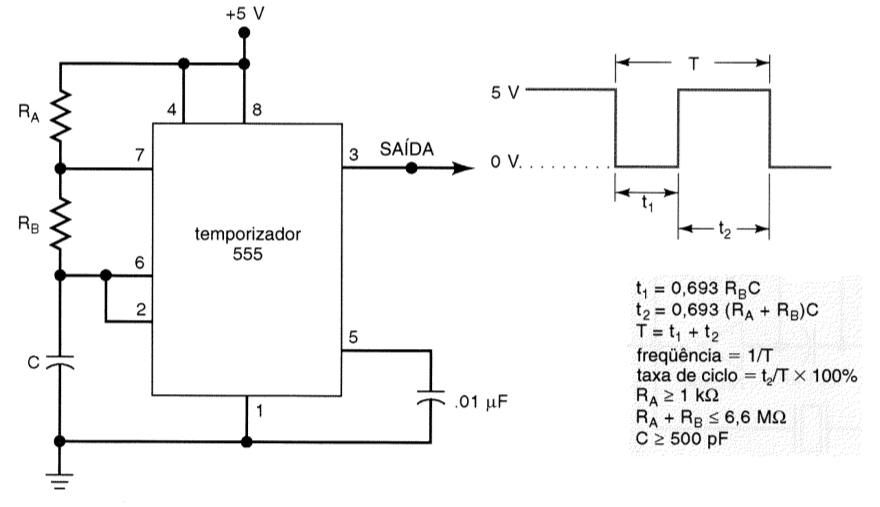


Figura : Multivibrador astável montado com o CI 555.

**Exemplo:** Cálculo para circuito gerador de *clock*.

Serão apresentados os cálculos para o projeto de um circuito astável que gere uma onda com o tempo em alto **th=1s** e o tempo em baixo **tl=0,2s**.

(1)

(2)

Utilizando um capacitor de 1µF na equação (2):

Aplicando este valor em (1):

**Obs 1:** Na prática deve-se utilizar um potenciômetro em associação a um dos resistores para ajustar adequadamente o valor de th e tl.

**Obs 2:** Pode-se perceber que matematicamente não é possível obter th = tl. Para obter um resultado próximo disso deve-se escolher RB ≫ RA.

**Exercícios:**

1. Projetar um circuito astável usando o CI 555, para operar com frequência de 10KHz e ciclo de trabalho de 30%.
2. Projetar um circuito monoestável com o CI 555, para operar com tw = 50ms.
3. Determine a frequência de onda gerada por um oscilador usando inversor Schmitt-Trigger implementado com o circuito 74LS14, com R = 1k e C = 10ηF.

# Famílias Lógicas de Circuitos Integrados

Os circuitos integrados são classificados por sua complexidade, de acordo com o número de transistores que compõem o chip, e pelo tipo de dispositivo semicondutor empregado em sua fabricação. Referente à complexidade dos CIs, estes se dividem em SSI (menos de 12 portas lógicas por chip), MSI (12 a 99 portas/chip), LSI (100 a 999 portas/chip), VLSI (1000 a 99999 portas/chip) e ULSI (mais de 100000 portas/chip).

Quanto ao tipo de dispositivo semicondutor empregado no chip, os CIs podem ser fabricados com a tecnologia MOS – *Metal Oxide Semiconductor* (família CMOS) e com a tecnologia bipolar (família TTL, ECL). Dentro de uma mesma família de CIs existem diversas categorias, que se diferenciam em potência consumida, velocidade de resposta, etc.

A tecnologia bipolar baseia-se na associação de transistores convencionais de junção PNP e NPN. Desta tecnologia resultam dispositivos rápidos mas com um consumo de corrente significativo. Como cada transistor bipolar ocupa uma área significativa na pastilha semicondutora o uso desta tecnologia limita-se a dispositivos SSI e MSI, ou seja, na construção de sistemas lógicos simples ou na interligação de componentes mais complexos. As famílias TTL e ECL são os resultados mais difundidos desta tecnologia.

Os dispositivos de tecnologia MOS são constituídos quase que exclusivamente de transistores de efeito de campo que contribuem para um consumo de corrente extremamente baixo. Entretanto esses dispositivos são normalmente mais lentos que seus equivalentes bipolares TTL e ECL. A maior vantagem da tecnologia MOS é a alta capacidade de integração, dispositivos MOS podem ocupar menos de 5%­ da área de seus equivalentes bipolares. Isto faz desta tecnologia a mais usada na implementação de circuitos VLSI como processadores grandes, memórias, etc.

## Parâmetros de Circuitos Integrados

Nesta seção serão apresentados inicialmente a **nomenclatura** padrão de tensão e corrente que se utiliza para representar os parâmetros dos circuitos integrados. A Figura 112 ilustra como é representado cada termo em nível lógico alto e baixo respectivamente.

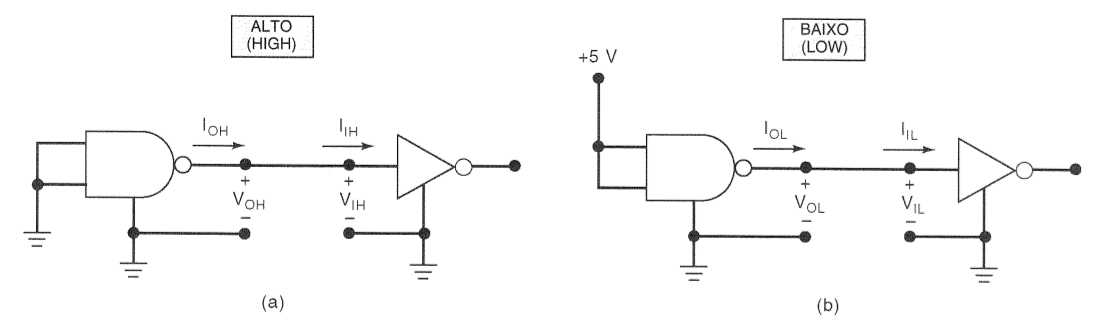


Figura : Terminologia de tensões e correntes.

* **VIH(mín):** tensão de entrada em nível alto (*high-level input voltage*). A mínima tensão para o nível lógico 1 em uma entrada. Abaixo deste nível a tensão não é considerada nível lógico alto.
* **VIL(máx):** tensão de entrada em nível baixo (*low-level input voltage*). A máxima tensão para o nível lógico 0 em uma entrada. Acima deste valor a tensão não é considerada nível lógico baixo.
* **VOH(mín):** tensão de saída em nível alto (*high-level output voltage*). O mínimo de tensão na saída para nível lógico 1.
* **VOL(máx):** tensão de saída em nível baixo (*low-level output voltage*). O máximo de tensão na saída para nível lógico 0.
* **IIH:** corrente de entrada em nível alto (*high level input current*). Corrente que flui para uma entrada quando aplicado nível lógico alto.
* **IIL:** corrente de entrada em nível baixo (*low-level input current*). Corrente que flui para uma entrada quando aplicado nível lógico baixo.
* **IOH:** corrente de saída em nível alto (*high-level output current*). Corrente que flui de uma saída em nível lógico 1.
* **IOL:** corrente de saída em nível baixo (*low-level output current*). Corrente que flui de uma saída em nível lógico 0.

***Fan-out*:** conhecido também como fator de carga. É o número máximo de entradas de circuitos lógicos que uma saída pode alimentar de forma confiável.

**Atraso de propagação:** é o tempo requerido pela saída de uma porta para responder a uma mudança no nível lógico da entrada da porta. Podem ser considerados dois tipos de atraso, como ilustra a Figura 113:

* **tPLH:** atraso correspondente a passagem de 0 para 1.
* **tPHL:** atraso correspondente a passagem de 1 para 0

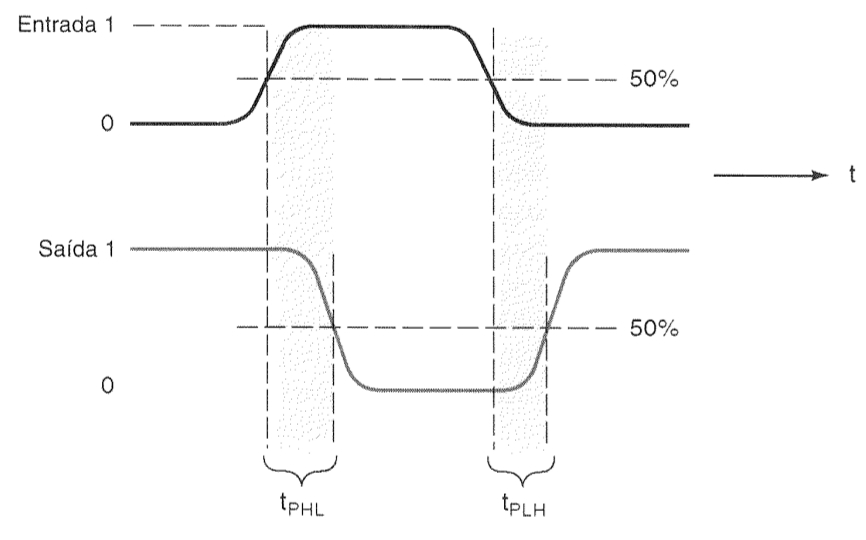
****

Figura : Atrasos de propagação.

**Requisitos de potência:** É a quantidade de potência elétrica necessária para o CI operar. A alimentação para a família TTL é denominada de VCC e para a família MOS é denominada VDD. A potência que um CI necessita é dada pelo produto: ICCxVCC (ou IDDxVDD), onde ICC, ou IDD, é a corrente fornecida pela fonte. Entretanto, a potência real consumida por um CI depende do nível lógico de suas entradas e saídas. Desta forma, a corrente fornecida pela fonte (ICC) é dada pela média das correntes em nível lógico alto e baixo, ou seja:

**ICC(médio) = (ICCH + ICCL)/2**

E a potência média consumida pelo CI é:

**PD(média) = ICC(média)xVCC**

**Margem de Ruído:** É a diferença entre a tensão de saída e a tensão de entrada de uma porta para um mesmo nível lógico. A Figura 114 apresenta um diagrama com as tensões de entrada e saída para um mesmo CI. Os valores VNH e VNL são os valores de margem de ruído para os níveis lógicos alto e baixo, respectivamente.

VNH = VOH - VIH

VNL = V­IL - VOL

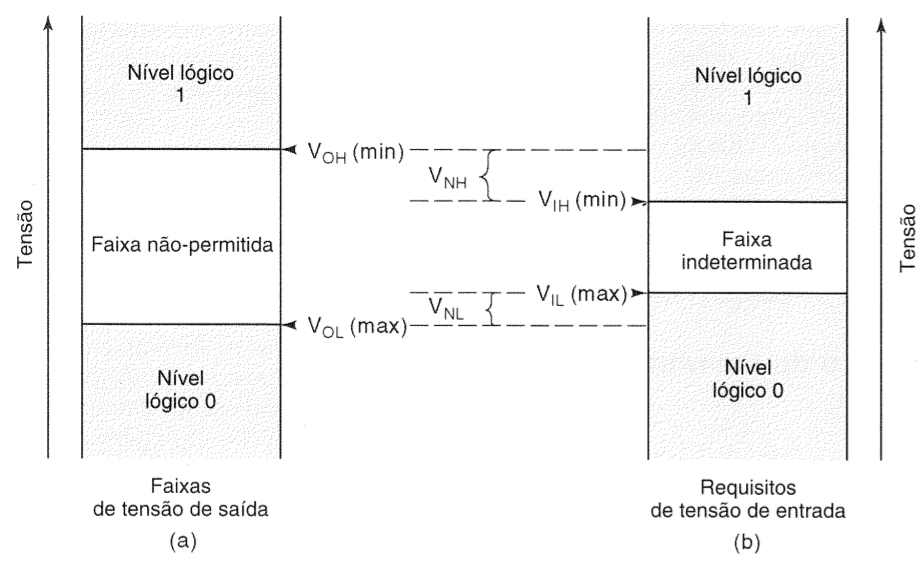


Figura : Margem de ruído.

Por meio da figura pode-se perceber que VNH e VNL permitem que haja um ruído máximo na saída de um determinado CI, pois mesmo com essa condição, a entrada do CI apresenta uma tolerância para o nível alto e baixo. Para as famílias TTL e CMOS, as respectivas faixas de tensão são representadas na Tabela 19.

Tabela : Margem de ruído para famílias TTL e CMOS.

|  |  |  |
| --- | --- | --- |
| Família | Parâmetro | Tensão |
| TTL | VOH | 2,4V |
| VIH | 2,0V |
| VNH | 0,4V |
| VIL | 0,8V |
| VOL | 0,4V |
| VNL | 0,4V |
| CMOS | VOH | 4,9V |
| VIH | 3,5V |
| VNH | 1,5V |
| VIL | 1,5V |
| VOL | 0,1V |
| VNL | 1,5V |

## A família TTL

A família lógica TTL existe desde meados da década de 60. Como já mencionado, internamente suas portas lógicas são construídas com transistores bipolares. A Figura 115 apresenta a estrutura interna de uma porta NAND da família TTL. A utilização dos CIs desta família caiu muito atualmente, sendo utilizado somente para executar a interface entre circuitos mais desenvolvidos. Entretanto esta família de circuitos serviu como base para a construção das tecnologias mais atuais.

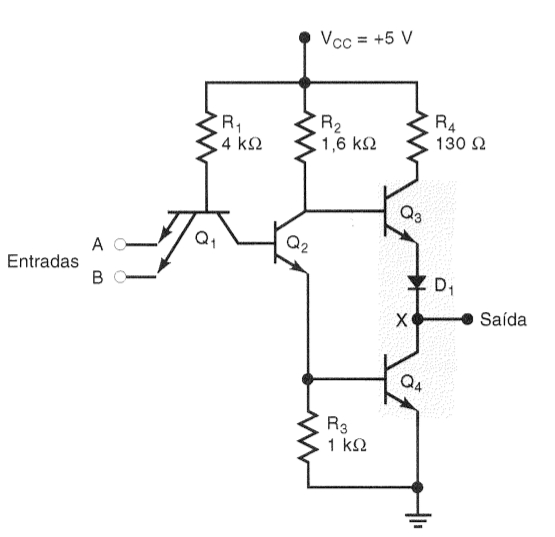
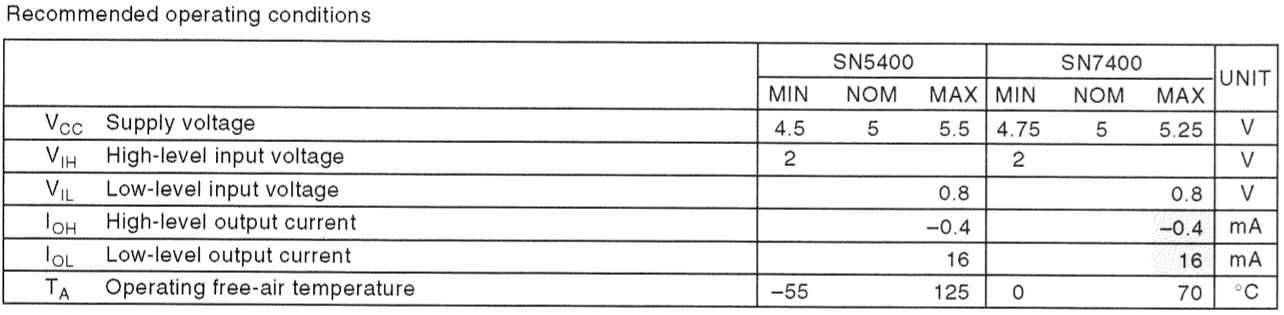


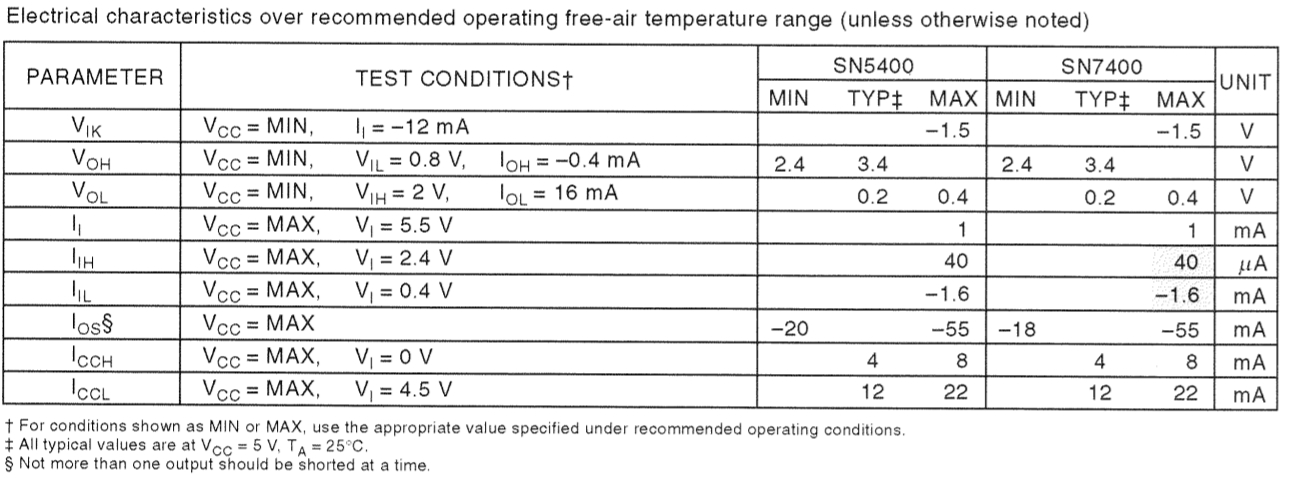
Figura : Estrutura interna de uma porta NAND TTL básica.

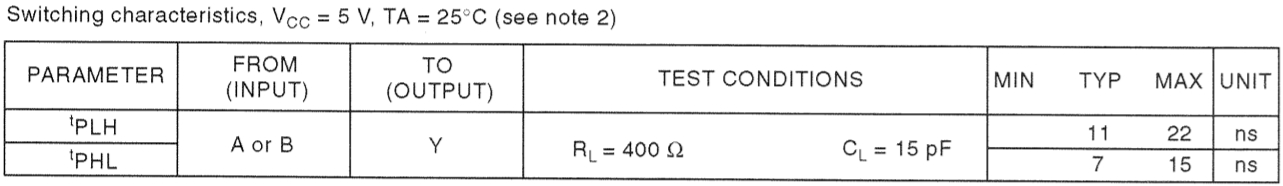
Dentro da família lógica TTL existem alguns tipos diferentes de construção, dentre as quais variam parâmetros de temperatura de operação, valores de tensão de alimentação, potência consumida, velocidade de resposta, etc.

Os circuitos da família TTL podem ser identificados pelas iniciais 74 ou 54. Os CIs 74 contêm as especificações comerciais, podendo operar de 0°C a 75°C. Os CIs 54 contêm especificações militares, podendo operar de -50°C a 125°C. Os demais parâmetros para uma porta NAND 7400 desta família estão apresentados na Tabela 20.

Tabela : Parâmetros para uma porta NAND 7400.







### Classificação das Séries TTL

As séries TTL fornecem uma ampla variedade de escolha dos parâmetros de velocidade e potência consumida.

* 54/74: Padrão;
* 54/74 L: Baixa potência;
* 54/74 H: Alta velocidade;
* 54/74 S: Schottky;
* 54/74 LS: Schottky de baixa potência;
* 54/74 AS: Schottky avançada;
* 54/74 ALS: Schottky avançada de baixa potência.

#### Série de Baixa Potência (54/74 L)

Os circuitos desta série são essencialmente os mesmos da série TTL padrão, exceto pelo fato de que todos seus resistores são de valor maior. Resistores de maior valor reduzem a potência, à custa de um aumento nos retardos de propagação (menor velocidade).

* Retardo de propagação: 33 ns;
* Consumo: 1mW.

#### Série de Alta Velocidade (54/74 H)

Os circuitos desta série, ao contrário da anterior, têm todos os resistores diminuídos. Como resultado, todas as constantes de tempo serão diminuídas (alta velocidade), à custa de um aumento de potência (consumo).

* Retardo de propagação: 6 ns;
* Consumo: 23 mW.

#### Série Schottky (54/74 S)

As séries 74, 74H e 74L usam transistores que, quando estão conduzindo, operam na região de saturação. Esta forma introduz um retardo de tempo de armazenamento sempre que o transistor mude de estado, ou seja, passe de *condução* para *corte*. Com isso há uma limitação de velocidade.

A série Schottky evita a saturação clássica do transistor usando um diodo de junção de barreira Schottky (SBD) – ilustrado na Figura 116 - entre a base e o coletor do transistor, com tensão de corte de 0,25V. Quando o transistor entra em condução, parte da corrente é desviada pelo diodo SBD evitando a saturação e diminuindo os atrasos de propagação.

* Retardo de propagação: 3 ns;
* Consumo: 20 mW.

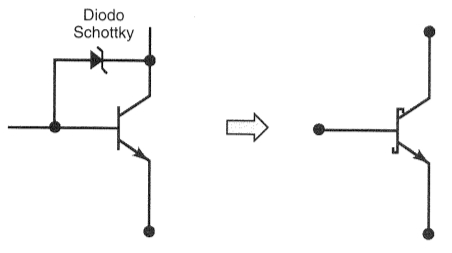


Figura : Transistor com diodo de junção de barreira Schottky (SBD).

#### Série Schottky de Baixa Potência (54/74 LS)

Esta série é uma versão da série 54/74S, cujos resistores são aumentados em valor para diminuir o consumo de potência. Como consequência o retardo de propagação se torna maior.

* Retardo de propagação: 9,5 ns;
* Consumo: 2mW.

#### Série Schottky Avançada (54/74 AS)

Esta série apresenta velocidade de operação bem menor, assim como a potência consumida. Outra vantagem é a necessidade de correntes de entrada (IIL, IIH) extremamente baixas, o que resulta em *fan-outs* maiores.

* Retardo de propagação: 1,7 ns;
* Consumo: 8mW.

#### Série Schottky Avançada de Baixa Potência (54/74 ALS)

Esta série apresenta um retardo de propagação um pouco maior do que a série anterior, mas consome uma potência bem menor. Isso torna a relação custo-benefício desta série a melhor dentre todas as demais séries da família TTL.

* Retardo de propagação: 4 ns;
* Consumo: 1,2 mW.

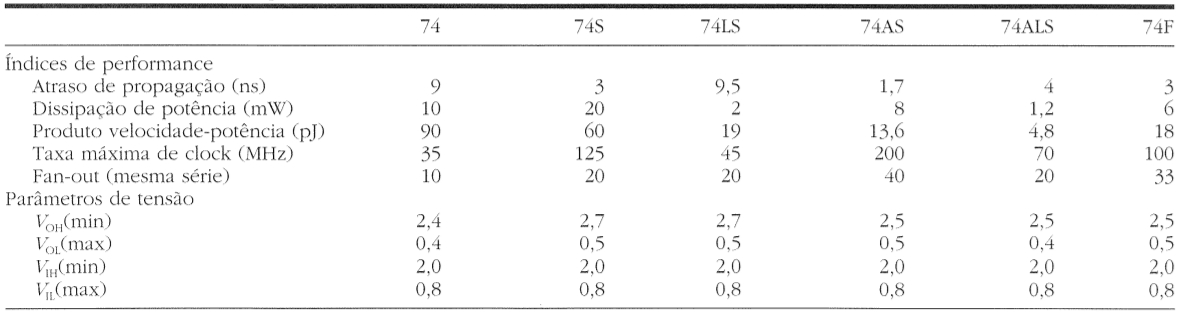
#### Série Fast (54/74 F)

Esta é a série TTL mais nova. Utiliza uma técnica para a fabricação dos circuitos integrados que reduz as capacitâncias entre os dispositivos internos para alcançar atrasos de propagação reduzidos.

* Retardo de propagação: 3 ns;
* Consumo: 6 mW.

Na Tabela 21 consta o comparativo entre alguns parâmetros de algumas das séries apresentadas. Os valores da tabela são referentes a uma porta NAND, exceto o valor de *clock* máximo, que se refere a um *flip-flop* JK.

Tabela : Comparativo de alguns parâmetros das séries TTL.



**Questionário:**

* + 1. Qual das séries TTL é a mais recomendada para operar em alta frequência?
    2. Qual das séries TTL tem a maior margem de ruído para o nível lógico alto?
    3. Assumindo o mesmo custo para os três componentes abaixo citados. Por que se escolheria um contador 74ALS193, ao invés de um 74LS193 ou um 74AS193, para ser usado em um circuito que opere com um *clock* de 40 MHz?

## *Fan-out* em dispositivos TTL

*Fan-out* de uma determinada saída de um CI TTL significa o número de entradas que tal saída pode alimentar de maneira confiável. Para esta análise são consideradas as seguintes variáveis: **IOL** e **IIL** são as correntes de saída e de entrada, respectivamente, em baixa tensão; **IOH** e **IIH** são as correntes de saída e de entrada, respectivamente, em alta tensão.

Em **nível lógico baixo**, considera-se que as correntes fluem das entradas dos CIs para as saídas que as conectam, como mostrado na **Figura 117** (a).

*Fan-out* (baixo) =

Em **nível lógico alto**, considera-se que as correntes fluem das saídas dos CIS para as entradas que estas alimentam, como mostrado na **Figura 117** (b).

*Fan-out* (baixo) =

|  |  |
| --- | --- |
| C:\Users\Rafael\Dropbox\IFC\Sistemas Digitais 12_2\Apostila de Sistemas Digitais\figuras cap 8\fan out nivel baixo.jpg | C:\Users\Rafael\Dropbox\IFC\Sistemas Digitais 12_2\Apostila de Sistemas Digitais\figuras cap 8\fan out nivel alto.jpg |

**Figura 117: Correntes de saída e de entrada em nível lógico baixo (a) e nível lógico alto (b).**

## As famílias MOS

A tecnologia MOS (*metal oxide semiconductor*) tem seu nome extraído do fato de sua estrutura básica ser formada por um eletrodo de metal conectado a uma camada de óxido isolante, que, por sua vez, é depositada sobre um substrato de silício. Os transistores MOS são transistores por efeito de campo (*field effect transistor*), por isso chamados de MOSFET.

### O Transistor MOSFET

Os CIs MOS utilizam apenas o MOSFET por indução. Podem ser de dois tipos:

* MOSFET de canal N
* MOSFET de canal P

O MOSFET pode ser visto como uma chave eletrônica controlada pela tensão da porta. A tensão entre a porta e a fonte, VGS, é a tensão de entrada, usada para controlar a resistência entre dreno e a fonte e determinar se o dispositivo está ligado ou desligado:

* VGS ≤ 0 V: chave aberta;
* VGS ≥ 1,5 V: chave fehada.

### Vantagens e desvantagens dos MOSFETs sobre a tecnologia bipolar

|  |  |
| --- | --- |
| Vantagens:   * Construção simples; * Baixo custo de fabricação; * Área pequena utilizada no CI (50 vezes menor); * Baixo consumo de potência; * Alta densidade de integração: permite LSI e VLSI; | Desvantagens:   * Baixa velocidade em relação à tecnologia bipolar; * Sensibilidade à eletricidade estática. |

## A família CMOS

A família CMOS (*Complementary MOS*) é a que apresenta o menor consumo e a melhor velocidade de operação entre as famílias da tecnologia MOS. Devido a estes fatores, esta é a tecnologia que domina o mercado atual, apesar de ter uma construção complexa.

### Classificação das séries CMOS

Assim como a família TTL, esta família apresenta uma diversidade de formas construtivas, sendo divididas em séries:

#### Série 4000/14000

Esta é a série mais antiga, possuem baixo consumo e podem operar em uma larga faixa de tensões. Apesar disso são dispositivos mais lentos e possuem baixa capacidade de corrente.

* Tensão de alimentação: 3 V a 15 V;
* Margem de ruído: 1,5 V (VDD = 5 V);
* *Fan-out*: 50;
* Consumo: varia com a frequência de trabalho;
* Velocidade de comutação: 50 ns (VDD = 5V)

#### Série 74C (compatível com TTL)

Esta série é compatível, considerando a pinagem e as funções, com os dispositivos TTL com a mesma numeração, possibilitando a substituição entre CIs destas duas famílias.

* Tensão de alimentação: 3V a 15V;
* Margem de ruído: 1,5 V (VDD = 5 V);
* *Fan-out*: 50;
* Consumo: varia com a frequência de trabalho;
* Velocidade de comutação: 50 ns.

#### Série 74HC (CMOS de alta velocidade)

É uma versão melhorada da série 74 C, tendo uma velocidade 10 vezes superior e sendo compatível em velocidade com os dispositivos TTL LS. Pode ainda fornecer uma alta corrente de saída.

* Tensão de alimentação: 2 V a 6 V;
* Margem de ruído: 0,9V (VDD = 5V);
* *Fan-out*: 50;
* Consumo: varia em função da frequência;
* Velocidade de comutação: 8 ns.

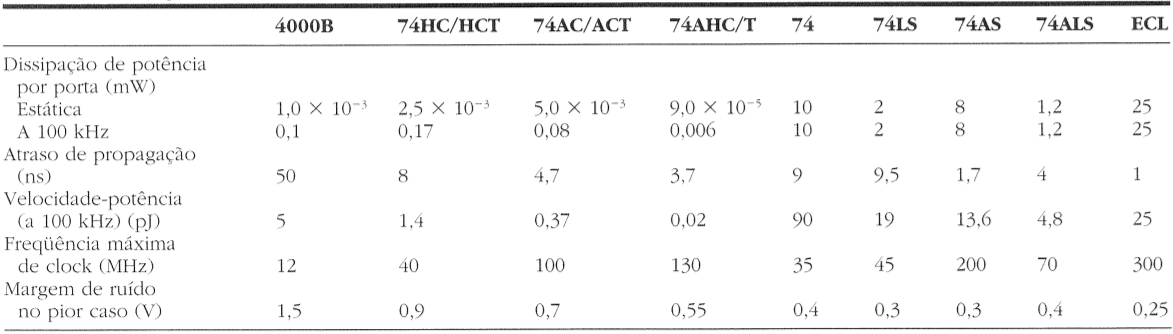
#### Série 74HCT (alta velocidade e compatível com TTL)

A exemplo da séria 74 HC, esta séria apresenta também uma alta velocidade de comutação, com a característica de ter sido construída para ser compatível em termos de tensões com a família TTL.

* Tensão de alimentação: 2 V a 6 V;
* Margem de ruído: 0,7 V (VDD = 5 V);
* *Fan-out*: 50;
* Consumo: varia em função da frequência;
* Velocidade de comutação: 8 ns.

Na Tabela 22 é mostrado um quadro comparativo entre as principais séries CMOS e TTL.

Tabela : Comparação entre as séries de CIs digitais (considerando VDD = 5V).



## Compatibilidade entre TTL e CMOS

O acoplamento entre dispositivos CMOS e TTL padrão não pode ser feito diretamente devido às diferençaas entre as tensões de nível lógico e as correntes de entrada e de saída das duas famílias mesmo sendo providas as mesmas tensões de alimentação. Além disso a diferença de velocidades de comutação pode causar problemas.

Para conectar uma saída TTL a uma entrada CMOS precisa-se elevar a tensão de nível lógico alto da saída que é muito baixa no padrão TTL (3,5 V). Isto pode ser obtido através de um resistor de *pull-up* como indica a Figura 118:

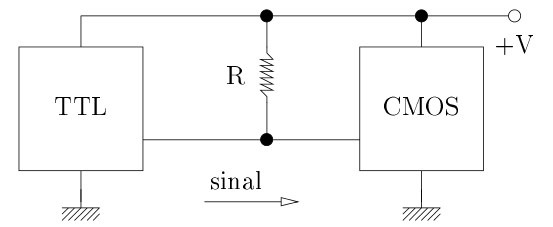


Figura : Conexão entre TTL e CMOS por meio de um resistor de *pull-up*.

Para acoplar uma saída CMOS a uma entrada TTL é necessário aumentar a capacidade de corrente da saída, o que pode ser feito usando um buffer. Além disso, no caso de dispositivos CMOS lentos é necessário aumentar a velocidade de comutação do sinal da saída através de um schmitt-trigger, como mostra a Figura 119:

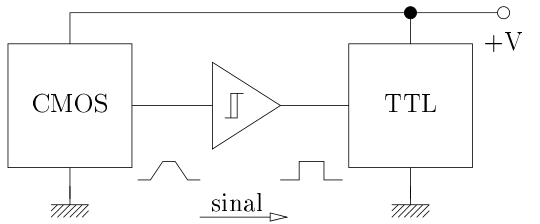


Figura : Conectando CMOS (lento) a uma entrada TTL (rápido).

As novas séries de dispositivos TTL e CMOS permitem uma maior flexibilidade de interconexão devido a melhoria nas características elétricas das duas famílias, sobretudo na família CMOS (maior velocidade de comutação e maior capacidade de corrente nas saídas). As regras acima continuam, no entanto válidas, como caso geral,  
 antes de interligar circuitos digitais de famílias distintas, é necessário certificar-se de que as tensões de nível lógico, a velocidade de operação e as correntes envolvidas nas entradas e saídas são compatíveis entre si.