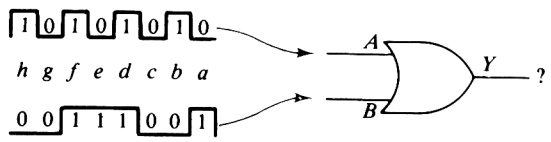
SENAI Espírito Santo

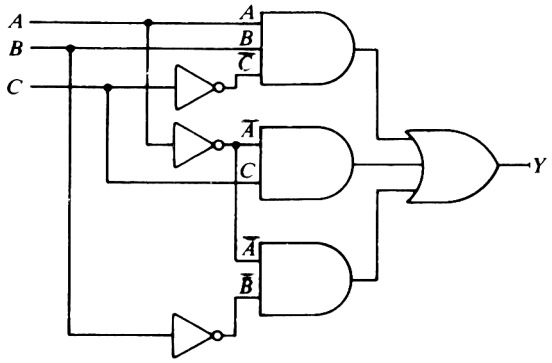
COMPANHIA SIDERÚRGICA DE TUBARÃO

CPM – Programa de Certificação do Pessoal de Manutenção

Instrumentação

Eletrônica Digital

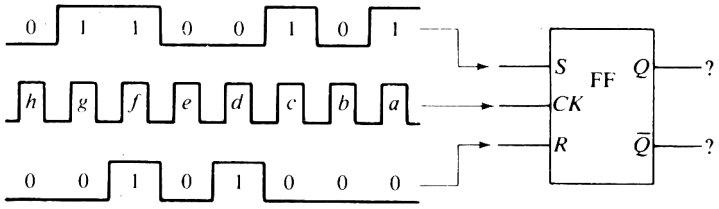




$$\bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} = Y$$

Entradas			Saída	Entradas			Saída
C	*B*	*A*	*Y*	*C*	*B*	*A*	*Y*
0	0	0	1	1	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	1	1	1	1	0

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



Eletrotécnica Básica – Instrumentação

Trabalho realizado em parceria SENAI / CST (Companhia Siderúrgica de Tubarão)

Coordenação Geral	Evandro de Figueiredo Neto (CST) Robson Santos Cardoso (SENAI)
Supervisão	Rosalvo Marcos Trazzi (CST) Fernando Tadeu Rios Dias (SENAI)
Elaboração	Jader de Oliveira (SENAI)
Aprovação	Alexandre Kalil Hanna (CST) Carlos Athico Prates (CST) Robisley Silva Braga (CST) Wenceslau de Oliveira (CST)

SENAI – Serviço Nacional de Aprendizagem Industrial
CTIIAF – Centro Técnico de Instrumentação Industrial Arivaldo Fontes
Departamento Regional do Espírito Santo
Av. Marechal Mascarenhas de Moraes, 2235
Bento Ferreira – Vitória – ES
CEP
Telefone: (027) 334 - 5200
Telefax: (027) 334 - 5212

CST – Companhia Siderúrgica de Tubarão
Departamento de Recursos Humanos
Av. Brigadeiro Eduardo Gomes, s/n, Jardim Limoeiro – Serra – ES
CEP 29160-972
Telefone: (027) 348-1286
Telefax: (027) 348-1077

SENAI
Espírito Santo

COMPANHIA SIDERÚRGICA DE TUBARÃO

Índice

Assunto **Página**

ESTUDO BÁSICO DE CIRCUITO DIGITAL

1.1 - DIFERENÇAS ENTRE CIRCUITOS DIGITAIS E ANALÓGICOS

Os circuitos analógicos utilizam no seu funcionamento grandezas continuamente variáveis, em geral tensões e corrente elétrica.
Os circuitos digitais produzem sua saída, respondendo a incrementos fixos. A entrada no circuito analógico nunca constitui um número absoluto: é uma posição aproximada numa escala contínua. Por exemplo: um relógio analógico possui os ponteiros que estão em constante movimento; não possui um valor determinado para o intervalo de tempo.
O relógio digital tem sua indicação das horas através de números que mudam de intervalo em intervalo.

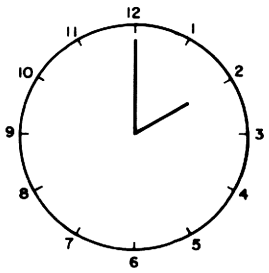


Fig. 1 – Sistema analógico
Relógio analógico

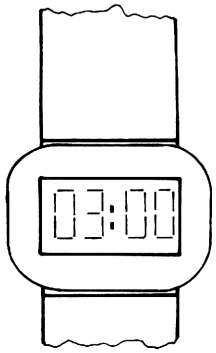


Fig. 2 – Sistema digital
Relógio digital

Outro exemplo, seria você estar subindo uma rampa ou escada. Subindo uma rampa, você está a cada instante em movimento para cima. Já na escada não, você, em cada instante está em um degrau.
Assim podemos então entender que um circuito analógico tem suas variáveis em contínua variação no tempo, e o circuito digital possui suas variáveis fixas em períodos de tempo.

1.2. SISTEMAS DE NUMERAÇÃO

Todos nós, quando ouvimos pronunciar a palavra números, automaticamente a associamos ao sistema decimal com o qual estamos acostumados a operar. Este sistema está fundamentado em certas regras que são base para qualquer outro. Vamos, portanto, estudar estas regras e aplicá-las aos sistemas de numeração binária, octal e hexadecimal. Estes sistemas são utilizados em computadores digitais, circuitos lógicos em geral e no processamento de informações dos mais variados tipos. O número decimal 573 pode ser também representado da seguinte forma:

$$573 = 500 = 70 + 3 \text{ ou } 573 = 5 \times 10^2 + 7 \times 10^1 + 3 \times 10^0$$

Isto nos mostra que um dígito no sistema decimal tem na realidade dois significados.
Um, é o valor propriamente dito do dígito, e o outro é o que está relacionado com a posição do dígito no número (peso). Por exemplo: o dígito 7 no número acima representa 7 x 10, ou seja 70, devido a posição que ele ocupa no número. Este princípio é aplicável a qualquer sistema de numeração onde os dígitos possuem "pesos", determinados pelo seu posicionamento. Sendo assim, um sistema de numeração genérico pode ser expresso da seguinte maneira:

$$N = dn \cdot B^n + \ldots + d3 \cdot B^3 + d2 \cdot B^2 + d1 \cdot B^1 + d0 \cdot B^0$$

Onde:

N = representação do número na base B
dn = dígito na posição n
B = base do sistema utilizado
n = valor posicional do dígito

Por exemplo, o número 1587 no sistema decimal é representado como:

$$N = d3 \cdot B^3 + d2 \cdot B^2 + d1 \cdot B^1 + d0 \cdot B^0$$

$$1587 = 1 \cdot 10^3 + 5 \cdot 10^2 + 8 \cdot 10^1 + 7 \cdot 10^0$$

1.2.1 - SISTEMA DE NUMERAÇÃO BINÁRIO

O sistema binário utiliza dois dígitos (base 2) para representar qualquer quantidade. De acordo com a definição de um sistema de numeração qualquer, o número binário 1101 pode ser representado da seguinte forma:

$$1101 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$
$$1101 = 8 + 4 + 0 + 1 = 13$$

Note que os índices foram especificados em notação decimal, o que possibilita a conversão binária-decimal como descrito acima.

Através do exemplo anterior, podemos notar que a quantidade de dígitos necessário para representar um número qualquer, no sistema binário, é muito maior quando comparada ao sistema decimal. A grande vantagem do sistema binário reside no fato de que, possuindo apenas dois dígitos, estes são facilmente representados por uma chave aberta e uma chave fechada ou, um relé ativado e um relé desativado, ou, um transistor saturado e um transistor cortado; o que torna simples a implementação de sistemas digitais mecânicos, eletromecânicos ou eletrônicos.
Em sistemas eletrônicos, o dígito binário (0 ou 1) é chamado de BIT, enquanto que um conjunto de 8 bits é denominado *BYTE*.

1.2.1.1 - Conversão Binário Decimal

A conversão de um número do sistema binário para o sistema decimal é efetuada simplesmente adicionando os pesos dos dígitos binários 1, como mostra o exemplo a seguir:

a) 11010 (B) b) 1100100 (B)

Solução:

a) $11010 = 1 . 2^4 + 1 . 2^3 + 0 . 2^2 + 1 . 2^1 + 0 . 2^0$
$11010 = 16 + 8 + 0 + 2 + 0$
$11010 = 26$ (D)

b) $1100100 = 1 . 2^6 + 1 . 2^5 + 0 . 2^4 + 0 . 2^3 + 1 . 2^2 + 0 . 2^1 + 0 . 2^0$
$1100100 = 64 + 32 + 0 + 0 + 4 + 0 + 0$
$1100100 = 100$ (D)

1.2.1.2 - Conversão Decimal Binário

Para se converter um número decimal em binário, divide-se sucessivamente o número decimal por 2 (base do sistema binário), até que o último quociente seja 1. Os restos obtidos das divisões e o último quociente compõem um número binário equivalente, como mostra o exemplo a seguir.

Exemplo:

Converter os seguintes números decimais em binário.

a) 23 (D) b) 52 (D)

Solução:

a) 23 |2
1 11 |2
1 5 |2
1 2 |2
0 1 - bit mais significativo

logo :

23 (D) = 10111 (B)

a) 52 |2
0 26 |2
0 13 |2
1 6 |2
0 3 |2
1 1

logo :

52 (D) = 110100 (B)

1.2.1.3 - Adição com números binários

A adição no sistema binário é efetuada de maneira idêntica ao sistema decimal. Devemos observar, entretanto, que o transporte (*vai um*) na adição em binário, ocorre quando temos 1+1 . A tabela abaixo ilustra as condições possíveis para adição de Bits.

A + B	Soma	Transporte
0 + 0	0	-
0 + 1	1	-
1 + 0	1	-
1 + 1	0	ocorre

Observe, nos exemplos seguintes, como é efetuada uma adição em binário.
Exemplo:
Adicionar os seguintes números binários.
a) 101110 + 100101
b) 1001 + 1100

Solução:
a)
```
    1   1 1
      1 0 1 1 1 0
+     1 0 0 1 0 1
  1 0 1 0 0 1 1
```

b)
```
    1
      1 0 0 1
+   1 1 0 0
  1 0 1 0 1
```

OBSERVAÇÃO:
O termo transporte, (vai um) utilizado para indicar o envio de um dígito para a posição imediatamente superior do número é chamado de CARRY em inglês. Este termo será utilizado a partir de agora, em lugar de "transporte", por ser encontrado na literatura técnica.

1.2.1.4 - Subtração em números binários

As regras básicas para subtração são equivalentes à subtração decimal, e estão apresentadas na tabela a seguir.

A - B	Diferença	Transporte
0 - 0	0	-
0 - 1	1	[ocorre]
1 - 0	1	-
1 - 1	0	-

Exemplo:
Subtrair os seguintes números binários.
a) 111 - 101

b) 1101 - 1010

Solução:

a)
$$\begin{array}{r} 1\ 1\ 1 \\ -\ 1\ 0\ 1 \\ \hline 0\ 1\ 0 \end{array}$$

b)
$$\begin{array}{r} 1\ 1\ 0\ 1 \\ -\ 1\ 0\ 1\ 0 \\ \hline 0\ 0\ 1\ 1 \end{array}$$

OBSERVAÇÃO:

- O termo transporte (pede um), utilizado para indicar a requisição de um dígito da posição imediatamente superior do número, é chamado Borrow em inglês. Este termo será utilizado, a partir de agora, em lugar de transporte, por ser o encontrado na literatura técnica.

O processo de subtração efetuado na maioria dos computadores digitais é realizado através da representação de números negativos. Por exemplo, a operação 7 - 5 pode ser representada como sendo 7 + (-5). Observe que, na segunda representação, a operação efetuada é uma adição de um número positivo com um negativo.

Os números binários negativos são representados através do 2º complemento. Vejamos como isto é feito. O segundo complemento de um número binário é obtido adicionando-se 1 ao primeiro complemento do mesmo. O primeiro complemento é obtido simplesmente, complementando os dígitos que formam o número.

Exemplo:

Calcule o 2° complemento dos seguintes números binários.

a) 1001 b) 1101

Solução:

a)
$$\begin{array}{r} 1\ 0\ 0\ 1 \\ 0\ 1\ 1\ 0 \rightarrow 1^{\circ}\text{ complemento} \\ +\quad\ 1 \\ \hline 0\ 1\ 1\ 1 \rightarrow 2^{\circ}\text{ complemento} \end{array}$$

b)
$$\begin{array}{r} 1\ 1\ 0\ 1 \\ 0\ 0\ 1\ 0 \rightarrow 1^{\circ}\text{ complemento} \\ +\quad\ 1 \\ \hline 0\ 0\ 1\ 1 \rightarrow 2^{\circ}\text{ complemento} \end{array}$$

No exemplo anterior (a), o número 9 (1001) tem como segundo complemento 0111. O segundo complemento é a representação negativa do número binário, ou seja, -9 é representado como sendo 0111.

A subtração binária através do 2° complemento, é realizada somando o subtrator com o 2º complemento do subtraendo, como mostra o exemplo a seguir.

Exemplo:

Subtraia os seguintes números em binários.

a) 13 - 7 b) 6 -9

Solução:

a) 13 = 1101
7 = 0111

Calculando o 2° complemento de 7 (0111), temos:

```
  0 1 1 1                          logo:
  1 0 0 0→ 1° complemento            13 =   1 1 0 1
+       1                          - 7 = + 1 0 0 1
  1 0 0 1→ 2° complemento              6     0 1 1 0
```

OBSERVAÇÃO:
Sempre que houver carry do bit mais significativo, ele deverá ser desprezado.

b) 6 = 0110
9 = 1001

Calculando o 2° complemento de 9 (1001), temos:

```
  1 0 0 1                          0 1 1 0
  0 1 1 0→ 1° complemento         + 0 1 1 1
+       1                          1 1 0 1
  0 1 1 1→ 2° complemento
```

Se no resultado da soma (1101) não existe carry, devemos achar o 2° complemento deste número e acrescentar o sinal negativo (-).

```
  1 1 0 1                          então:
  0 0 1 0→ 1° complemento          6 - 9 = - 3, ou seja: - 0011
+       1
(-) 0 0 1 1→ 2° complemento
```

OBSERVAÇÃO:

Podemos achar o 2° complemento de um binário pela seguinte regra: conserva o 1° (primeiro) bit um (1) menos significativo e faz-se o 1° complemento dos bits mais significantes (bits da esquerda).
Exemplos:

1) 1 0 0 1→ 9
0 1 1 1→ 2° complemento
conserva
1° complemento

2) 1 0 0 0 → 8
1 0 0 0→ 2° complemento
conserva
1° complemento

3) 0 1 1 0 → 6
1 0 1 0 → 2° complemento
conserva
1° complemento

1.2.3 - SISTEMA DE NUMERAÇÃO HEXADECIMAL

O sistema hexadecimal, ou sistema de base 16, é largamente utilizado nos computadores de grande porte e vários microcomputadores. Neste sistema são utilizados 16 símbolos para representar cada um dos dígitos hexadecimais, conforme a tabela a seguir:

Nº DECIMAL	DÍGITO HEXADECIMAL	Nº BINÁRIO
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Note que as letras A, B, C, D, E, F representam dígitos associados às quantidades, 10, 11, 12, 13, 14, 15, respectivamente.

1.2.3.1 - Conversão Hexadecimal Decimal

Novamente aplicamos para o sistema hexadecimal a definição de um sistema de numeração qualquer. Assim temos:

$$N = dn \cdot 16^n + \ldots + d2 \cdot 16^2 + d1 \cdot 16^1 + do \cdot 16^0$$

Para se efetuar a conversão, basta adicionar os membros da segunda parcela da igualdade, como ilustra o exemplo a seguir:
Exemplo:
Converter em decimal os seguintes números hexadecimais.
a) 23 (H) b) 3B (H)

Solução:
a) $23\ (H) = 2 \cdot 16^1 + 3 \cdot 16^0$
$23\ (H) = 2 \cdot 16 + 3 \cdot 1$
$23\ (H) = 35\ (D)$

b) $3B\ (H) = 3 \cdot 16^1 + B \cdot 16^0$

3B (H) = 3 . 16 + 11
3B (H) = 59 (D)

Observe que o dígito hexadecimal "B", no exemplo (b), equivalente ao número 11 decimal, como mostra a tabela apresentada anteriormente.

1.2.3.2 - Conversão Decimal Hexadecimal

A conversão decimal hexadecimal é efetuada através das divisões sucessivas do número decimal por 16, como demostrado no exemplo a seguir.
Exemplo:
Converter em hexadecimal os seguintes números:
a) 152 (D) b) 249 (D)

Silução:
a) 152 | 16 — logo:
8 9 — 152 (D) = 98 (H)

b) 249 | 16 — logo:
9 15 — 249 (D) = F9 (H)

1.2.4 - NÚMEROS DECIMAIS CODIFICADOS EM BINÁRIO (BCD)

Como já foi discutido anteriormente, os sistemas digitais em geral, trabalham com números binários. Com o intuito de facilitar a comunicação homem-máquina, foi desenvolvido um código que representa cada dígito decimal por um conjunto de 4 dígitos binários, como mostra a tabela seguinte:

Nº DECIMAL	REPRESENTAÇÃO BINÁRIA
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Este tipo de representação é denominado de código BCD (Binary-Coded Decimal).
Desta maneira, cada dígito decimal é representado por grupo de quatro bits, como ilustrado a seguir:

527 = 0101 0010 0111
527 = 010100100111

Observe que a conversão decimal-BCD e BCD-decimal é direta, ou seja, separando-se o dígito BCD em grupos de 4 bits, cada grupo representa um dígito decimal.
Exemplo:
Converter os seguintes números decimais em BCD.
a) 290 (D) b) 638 (D)

Solução:
a) 290 = 0010 1001 0000
290 = 001010010000

b) 638 = 0110 0011 1000
638 = 011000111000

Exemplo:
Converter os seguintes números em decimal.
a) 1001010000001000 = 1001 0100 0000 1000
1001010000001000 = 9 4 0 8
= 9408

b) 001001101001 = 0010 0110 1001
001001101001 = 2 6 9
= 269

1.3.- CIRCUITOS DIGITAIS BÁSICOS

Os sistemas digitais são formados por circuitos lógicos denominados Portas Lógicas.
Existem 3 portas básicas que podem ser conectadas de maneiras variadas, formando sistemas que vão de simples relógios digitais aos computadores de grande porte.
Veremos as características das 3 portas básicas, bem como seus símbolos e circuitos equivalentes.

1.3.1 - *Porta* AND (E)

Esta porta pode ter duas ou mais entradas e uma saída e funciona de acordo com a seguinte definição:
"A saída de uma porta AND será 1, somente se todas as entradas forem 1".

Na figura 3, temos o símbolo de uma porta AND de 2 entradas (A e B) juntamente com um quadro que mostra todas as possibilidades de níveis de entrada com a respectiva saída.
Este quadro é chamado de Tabela Verdade.

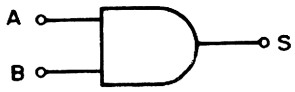


Fig. 3

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

-Representação:

S = A . B

AND

(e)

Analisemos agora o circuito da figura 4. Este circuito executa a função AND.
Considere o nível lógico 1 igual a "Chave fechada" e nível lógico 0 (zero) igual a chave aberta.

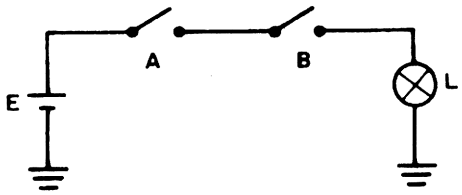


Fig. 4

Quando tivermos a condição de chave A aberta (0) e chave B aberta (0), não circulará corrente e a lâmpada L fica apagada (0).
Na condição de termos a chave A aberta (0) e a chave B fechada (1), ainda assim não circula corrente e a lâmpada está apagada (0).
É fácil observar que a condição inversa [chave A(1) e chave B(0)], também implica em a lâmpada estar apagada (0).
Agora temos a condição em que a chave A fechada (1) e a chave B fechada (1). Desta maneira a corrente pode circular e a lâmpada acende (1).
Verifique portanto que a análise acima descrita confirma a tabela verdade da figura 3.
Para o circuito AND portanto, podemos afirmar que qualquer 0 (zero) na entrada leva a saída para o 0 (zero).

1.3.2 - Porta OR (ou)

Esta porta também possui duas ou mais entradas, e uma saída, funcionando de acordo com a seguinte definição:
"A saída de uma porta OR será 1 se uma ou mais entradas forem 1".

Na figura 5 temos o símbolo de uma porta OR de 2 entradas (A e B) juntamente com a respectiva tabela verdade.

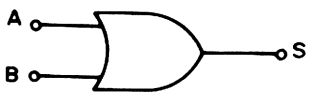


Fig. 5

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

- Representação:

S = A + B

OR

(ou)

Para a análise do circuito da porta OR (figura 6), vamos manter as mesmas considerações utilizadas da porta AND, ou seja:
Chave aberta = nível lógico 0 (zero)
Chave fechada = nível lógico 1 (um)
Quando tivermos chave A fechada e chave B aberta, teremos corrente circulando e consequentemente a lâmpada L estará acesa.
A lâmpada fica acesa também com as condições:

- Chave A = Aberta e Chave B = Fechada
- Chave A = Fechada e Chave B = Fechada.

A lâmpada somente estará apagada quando as duas chaves (A e B) estiverem abertas.

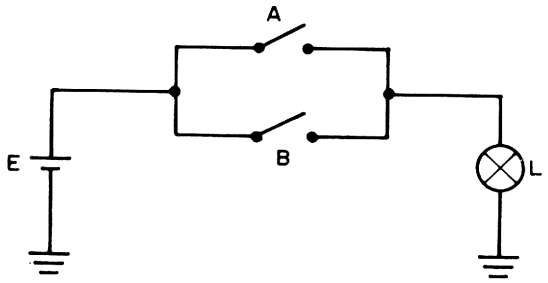


Fig. 6

Analise o circuito, novamente comparando-o com a tabela verdade da figura 5.
Podemos afirmar, portanto, que para um circuito OR, qualquer 1 na entrada leva a saída para 1.

1.3.3 - Porta NOT (não)

A porta NOT possui somente uma entrada e uma saída e obedece à seguinte definição:
"A saída de uma porta NOT assume o nível lógico 1 somente quando sua entrada é 0 (zero) e vice-versa".
Isto significa que a porta NOT é um inversor lógico, ou seja, o nível lógico da sua saída será sempre o oposto do nível lógico de entrada. A figura 7 apresenta o símbolo da porta lógica NOT e sua tabela verdade.

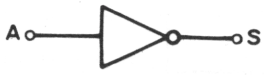


Fig. 7

A	S
0	1
1	0

- Representação

$S = \overline{A}$ not (não)

O circuito que executa a função NOT é mostrado na figura 8.
Observe que o circuito se resume a uma chave ligada para o terra. Quando a chave está aberta, a corrente circula pela lâmpada que fica acesa. Quando a chave A fecha , a corrente circula agora pela chave. Com isso a lâmpada se apaga.
Verifica-se portanto a tabela verdade da figura 7.

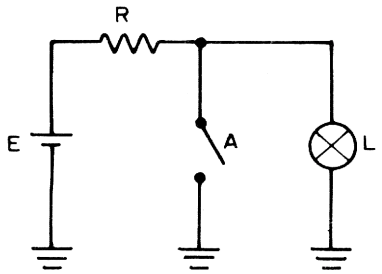


Fig. 8

1.3.4 - Portas NAND e NOR (NE e NOU)

As portas lógicas NAND e NOR são na realidade combinações das três portas básicas AND, OU e NOT. Entretanto, por fatores que serão discutidos posteriormente, estas portas são tomadas como portas básicas das famílias lógicas. Vamos portanto, analisar cada uma delas:
A figura 9 apresenta uma porta NAND de duas entradas com o símbolo e a tabela verdade.
Note que a porta NAND é constituída de uma AND seguida de um inversor (NOT).

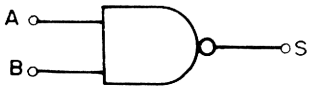


Fig.9

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

- Representação:

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

O circuito de uma porta NAND é visto na figura 10 onde é fácil verificar a tabela verdade.
Podemos afirma que para uma NAND que qualquer 0 (zero) na entrada, leva a saída para 1.

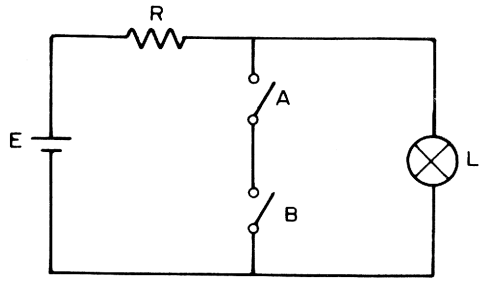


Fig. 10

A figura 11 apresenta o símbolo de uma porta NOR que é uma combinação de uma OR e um inversor (NOT). Segundo a tabela da figura 11, podemos afirmar para uma NOR que: "Qualquer 1 na entrada leva a saída para 0 (zero)."

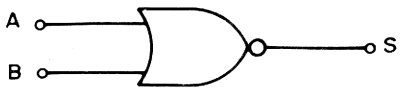


Fig. 11

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

-Representação:

$S = \overline{A + B}$

Analisando o circuito da figura 12. É fácil concluir que quando qualquer uma das entradas (Chave A ou Chave B) estiverem com 1(fechada) e saída S (lâmpada L) estará com 0 (zero) (lâmpada apagada).

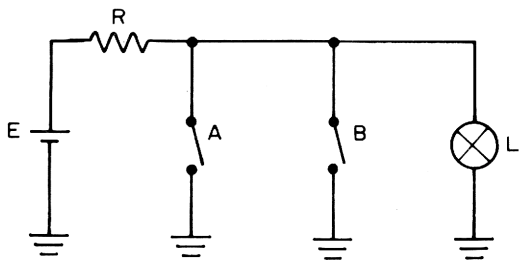


Fig. 12

1.*3:5 - Porta Exclusive-OR* (ou exclusiva)
A função que esta porta executa, como o próprio nome diz; consiste em fornecer a saída quando as variáveis de entrada forem diferentes entre si. A figura 13 apresenta o símbolo de uma porta exclusive-OR e sua tabela verdade.

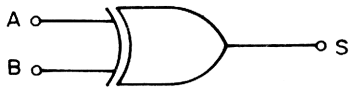


Fig. 13

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

$S = A \oplus B$

O circuito da figura 14 verifica a tabela, utilizando as chaves A e B.
Na condição em que as chaves A e B estão abertas, não há caminho para a corrente circular e a lâmpada não acende. Com a condição das chaves A e B fechadas, também não se tem corrente circulando e a' lâmpada não se acende.
Portanto, concluímos que esta porta só terá nível 1 na saída quando suas entradas forem diferentes.

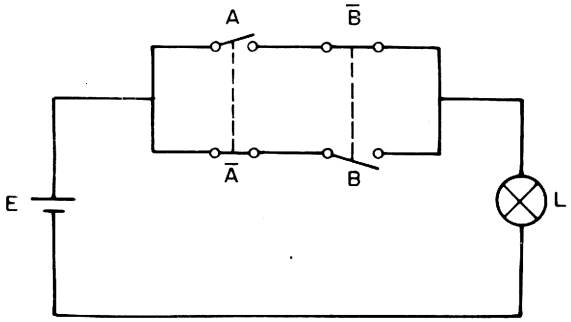


Fig. 14

1.3.6 - Porta Exclusive-NOR (Não-Exclusiva ou circuito coincidência)
Esta porta tem como função, fornecer 1 na saída somente quando suas entradas forem iguais.
A figura 15 mostra o símbolo de uma porta exclusive-NOR e sua tabela verdade.

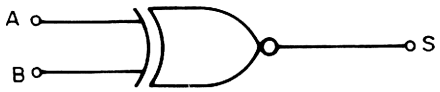


Fig. 15

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

$S = A \odot B$

No circuito da figura 16 existem agora as chaves $\overline{A}$ e $\overline{B}$; que funcionam de maneira inversa às chaves A e B, isto é; quando a chave A está aberta, a chave $\overline{A}$ está fechada o mesmo acontecendo com as chaves B e $\overline{B}$.
Desta maneira podemos verificar a tabela verdade da figura 15 através da seguinte análise. Quando as chaves A e B estão abertas (chaves $\overline{A}$ e $\overline{B}$ estão fechadas) circula corrente pela lâmpada e ela estará acesa. Quando a chave A está fechada e a chave B aberta (chave $\overline{B}$ fechada) não circula corrente pela lâmpada, o que implica em lâmpada apagada.. Na situação inversa chave A aberta (chave $\overline{A}$ fechada) e chave B fechada ocorre a mesma coisa e a lâmpada estará apagada.
Com as duas chaves A e B fechadas (Chave $\overline{A}$ e $\overline{B}$ abertas) circulará corrente pela lâmpada e esta estará acesa.
Portanto, pode-se afirmar que a porta exclusive-NOR terá 0 (zero) em sua saída quando as entradas forem diferentes.

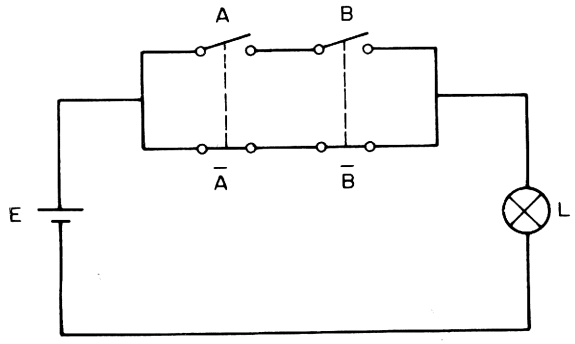
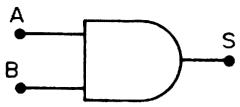
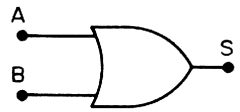
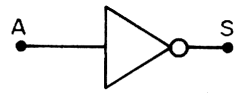
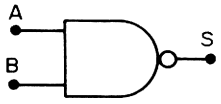
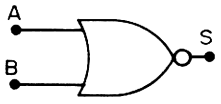
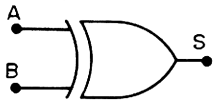
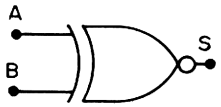


Fig. 16

1.3.7 - QUADROS RESUMO.

BLOCOS LÓGICOS BÁSICOS

Porta	Símbolo Usual	Tabela Verdade	Função Lógica
E AND		A B S 0 0 0 0 1 0 1 0 0 1 1 1	Função E: assume valor 1 quando todas as variáveis forem iguais a 1 e assume valor zero nos outros casos possíveis.
OU OR		A B S 0 0 0 0 1 1 1 0 1 1 1 1	Função OU: assume valor zero quando todas variáveis forem iguais a zero e assume valor 1 nos outros casos.
NÃO NOT INVERSOR		A $\overline{A}$ 0 1 1 0	Função Não: inverte a variável aplicada à sua entrada.

NE NAND		A 0 0 1 1	B 0 1 0 1	S 1 1 1 0	Função NE: Inverso da função E.
NOU NOR		A 0 0 1 1	B 0 1 0 1	S 1 0 0 0	Função NOU: Inverso da função OU.
EX. OU EX.OR		A 0 0 1 1	B 0 1 0 1	S 0 1 1 0	Função EX-OU: Assume valor 1 quando as variáveis forem diferentes e zero quando forem iguais
EX.NOU EX.NOR		A 0 0 1 1	B 0 1 0 1	S 1 0 0 1	Função Ex.NOU: inversa em função Ex.OU

1.4 - ÁLGEBRA DE BOOLE

Álgebra Booleana é uma técnica matemática usada quando consideramos problemas de natureza lógica. Em 1847, um matemático inglês chamado George Boole, desenvolveu as leis básicas e regras matemáticas que poderiam ser aplicadas em problemas de lógica dedutiva. Até 1938, estas técnicas se limitaram a serem usadas no campo matemático. Nesta época, Claude Shammon, um cientista do Be1 Laboratories, percebeu a utilidade de tal álgebra quando aplicada no equacionamento e análise de redes de multicontatos. Com o desenvolvimento dos computadores, o uso da álgebra de Boole no campo da eletrônica cresceu, de modo que ela é hoje ferramenta fundamental, para engenheiros e matemáticos no desenvolvimento de projetos lógicos. Originalmente a álgebra de Boole foi baseada em proposições que teriam como resultado serem falsas ou verdadeiras. Shammon usou a álgebra de Boole para equacionar uma malha de contatos que poderiam estar abertos ou fechados.
No campo de computadores é usada na descrição de circuitos, podendo assumir os estágios lógicos 1 ou 0. É fácil perceber que a lógica de Boole é extremamente interrelacionada com o sistema de numeração binária, já que ambos trabalham com duas variáveis.

1.4.1 - Postulados e Teoremas Booleanos

Toda teoria de Boole está fundamentada 7 postulados apresentados a seguir:

P1 - X = 0 ou X = 1
P2 - 0 . 0 = 0
P3 - 1 . 1 = 1
P4 - 0 + 0 = 0
P5 - 1 + 1 = 1
P6-1 . 0 = 0 . 1 = 0
P7 - 1 + 0 = 0 + 1 = 1

Compare estes postulados com as definições de adição lógica e multiplicação lógica, apresentadas anteriormente.
Fundamentado nos postulados Booleanos, um número de teoremas pode agora ser apresentado.
O teorema em álgebra de Boole é uma relação fundamental entre as variáveis Booleanas. O uso dos teoremas irá permitir simplificações nas equações lógicas e manipulações em circuitos lógicos das mais variadas formas. Analisemos cada um dos teoremas.

T1 - Lei comutativa
(a) A + B = B + A
(b) A . B = B . A

T2- Lei Associativa
(a) (A + B) + C = A + (B + C)
(b) (A . B) . C = A . (B . C)

T3 - Lei distribuitiva
(a) A . (B + C) = A . B + A . C
(b) A + (B . C) = (A + B) . (A + C)

T4 - Lei da identidade
(a) A + A = A
(b) A . A = A

T5 - Lei da Negação
(a) $(\overline{A}) = A$
(b) $\overline{(A)} = A$

T6 - Lei de redundância
(a) A + A . B = A
(b) A . (A + B) = A

T7 -(a) 0 + A = A
(b) 1 . A = A
(c) 1 + A = 1
(d) 0 . A = 0

T8 - (a) $\overline{A} + A = 1$
(b) $\overline{A} \cdot A = 0$

T9 -(a) $A + \overline{A} . B = A + B$
(b) $A . (\overline{A} + B) = A . B$

T10 - Teorema de Morgan
(a) $\overline{A + B} = \overline{A} . \overline{B}$
(b) $\overline{A} . B = \overline{A} + \overline{B}$

Observe que todos os teoremas são divididos em duas partes, portanto, são duais entre si.
O termo dual significa que as operações OR e AND são intercambiáveis.
Para se obter o dual de um teorema, basta substituir os "1" por "0" e vice-versa, e substituir a função lógica AND por OR e vice-versa. Observe o exemplo a seguir:

T1 - Lei comutativa
(a) A + B = B + A
(b) A . B = B . A

T6 - (a) A + A . B = A
(b) A . (A + B) = A

T8 - (a) $\overline{A} + A = 1$
(b) $\overline{A} . A = 0$

Os três primeiros teoremas mostram que as leis básicas de comutação, associação e distribuição de álgebra convencional são também válidas para as variáveis Booleanas. A lei da navegação só é aplicável à lógica de duas variáveis, como é o caso da álgebra de Boole. A lei redundância pode ser facilmente comprovada da seguinte maneira:

(a) $A + A . B = A$ Colocando A em evidência
$A . (1+ B) = A$
$A = A$ [T7 (b)]

(b) $A . (A + B) = A$
$A . A + A . B = A$
$A + A . B = A$
$A . (1 + B) = A$ [T7 (b)]
$A . 1 = A$
$A = A$

Os teoremas T7 e T8 são regras da álgebra Booleana.
T9 pode ser demonstrado como a seguir:

$A + \overline{A} . B = A + B$ Expandindo a Equação
$(A + \overline{A}) . (A + B) = A + B$ [T3(b)] (Fatoração)
$1 . (A + B) = A + B$ [T8(a)]
$A + B = A + B$ [T7(b)]

O teorema T10 é conhecido como teorema de Morgan e é uma das mais importantes ferramentas na manipulação de circuitos lógicos.

1.4.2 - Simplificação Lógica

Aplicando-se os teoremas e postulados Booleanos podemos simplificar equações lógicas, e com isto minimizar a implementação de circuitos lógicos. Vamos analisar como pode ser feita a simplificação lógica na série de exemplos a seguir:

Exemplo 1

Considere que a saída de um circuito lógico deve obedecer à seguinte equação:

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

Se este circuito fosse implementado desta forma através de portas lógicas, teríamos o circuito da figura 17.

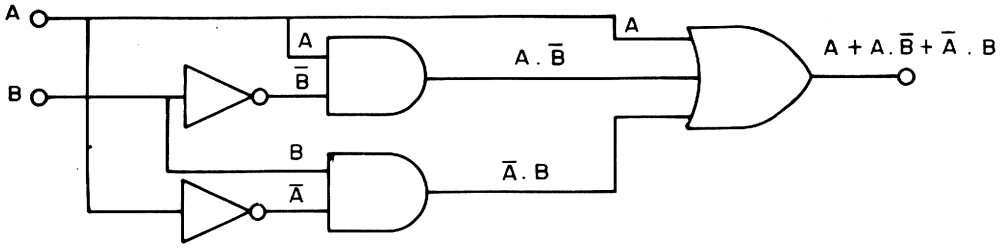


Fig. 17

Utilizando-se teoremas de Boole, vamos simplificar a equação dada.

$$A + A \cdot \overline{B} + \overline{A} \cdot B = (A + A \cdot \overline{B}) + \overline{A} \cdot B$$
$$= A + \overline{A} \cdot B \qquad [T6\ (a)]$$
$$= A + B \qquad [T9\ (a)]$$

A equação resultante pode ser implementada através do circuito da figura 18, ou seja, uma simples porta OR. Isto significa que os dois circuitos representam a mesma função lógica.

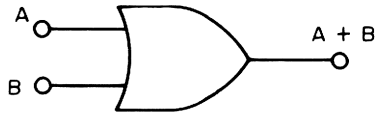


Fig. 18

Naturalmente o circuito simplificado é o ideal, visto que executa a mesma função lógica com um número reduzido de portas lógicas.

Exemplo 2

Simplifique a expressão A . (A . B + C)

Solução:

$$A \cdot (A \cdot B + C) = A \cdot A \cdot B + A \cdot C \qquad [T3(a)]$$
$$= A \cdot B + A \cdot C \qquad [T4(b)]$$
$$= A \cdot (B + C) \qquad [T3(a)]$$

1.4.3 - Manipulações Lógicas

Os teoremas de Boole são mais úteis na manipulação de variáveis lógicas do que propriamente na simplificação. Isto porque, um circuito após simplificado pode não estar em sua forma minimizada, e este processo de minimização se torna trabalhoso, em determinados casos, quando feito através de simplificações lógicas. Considere a seguinte equação lógica: $S=\overline{A} + B$. Suponha que seja necessário implementá-la através de portas lógicas NAND.

Aplicando o teorema de de Morgan na equação acima e negando duplamente o resultado, temos:

$\overline{A + B} = \overline{A} \cdot \overline{B}$ [De Morgan]

$A + B = \overline{\overline{A} \cdot \overline{B}}$ [Dupla negação]

Observe a figura:

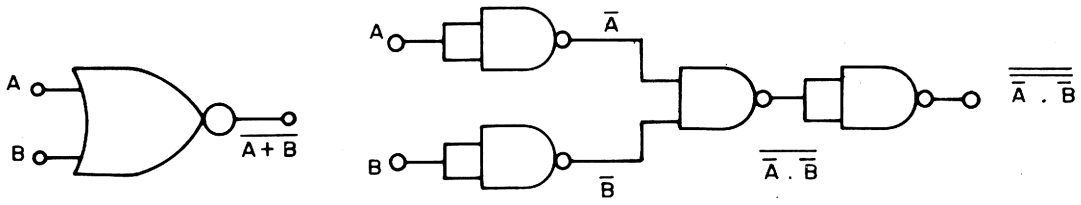
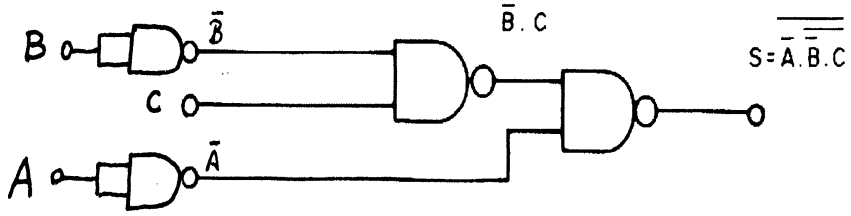


Fig. 19

Na realidade, qualquer expressão lógica pode ser manipulada de forma a ser totalmente implementada através de portas NAND ou NOR, como mostrado nos seguintes exemplos:
Exemplo 3
1) Implemente as seguintes expressões lógicas
a)D = A + $\overline{B}$. C
b) W = X . $\overline{Y}$ + $\overline{X}$. Z
Solução:

a) $A + \overline{B} \cdot C = \overline{\overline{A + \overline{B} \cdot C}}$ (Dupla negação)

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



b) $X\overline{Y} + \overline{X}\,Z = \overline{\overline{X\overline{Y}} + \overline{\overline{X}}\,Z}$ [Dupla negação]

$\overline{X\overline{Y}} + \overline{\overline{X}}Z \equiv \overline{X\overline{Y}} \cdot \overline{\overline{X}Z}$ [De Morgan]

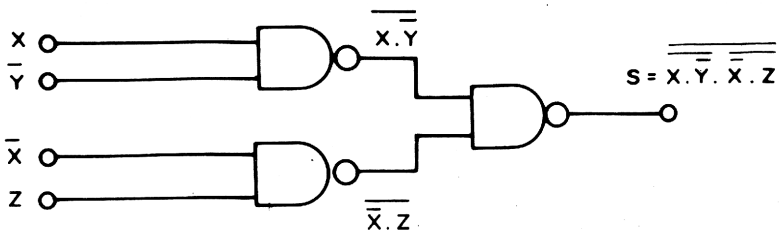


Fig. 21

2) implemente as seguintes expressões lógicas com portas NOR.

a) $X \cdot \overline{Y} + Z$
b) $(U + V) \cdot (X + \overline{Y} + \overline{Z})$

Solução:
a) Utilizando uma dupla negação no primeiro membro da equação, teremos:

$$X \cdot \overline{Y} + Z = \overline{\overline{X \cdot \overline{Y}} + Z}$$

Aplicando De Morgan no termo complemento em (1) temos:

$$\overline{\overline{X \cdot \overline{Y}}} + Z = \overline{X} + Y + Z$$

Finalmente, negando duplamente a equação completa temos:

$$\overline{\overline{X} + Y + Z} = \overline{\overline{X} + Y + Z}$$

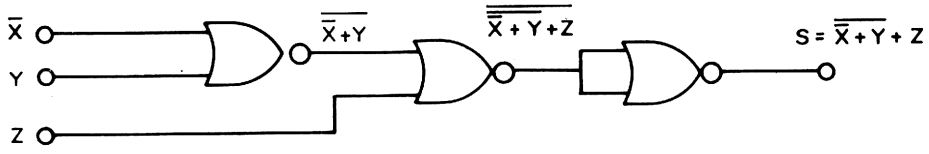


Fig. 22

b) Utilizando uma dupla negação na equação temos:

$$(U + V) \cdot (X + \overline{Y} + \overline{Z}) = \overline{\overline{(U + V) \cdot (X + \overline{Y} + \overline{Z})}}$$

Aplicando De Morgan em (2) temos:

$$\overline{\overline{(U + V) \cdot (X + Y + Z)}} = \overline{\overline{(U + V)} \; \overline{(X + \overline{Y} + \overline{Z})}}$$

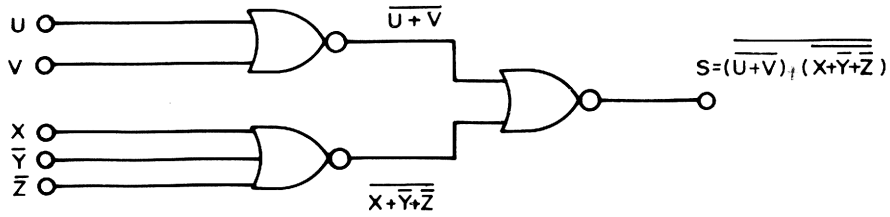


Fig. 23

1.5 - MAPA DE KARNAUGH

Quando utilizamos os teoremas e postulados Booleanos para simplificação de uma expressão lógica qualquer não podemos afirmar, em vários casos, que a equação resultante está na sua forma minimizada.
Existem métodos de mapeamento das expressões lõgicas, que possibilitam a minimização de expressões com N variáveis. O método descrito neste capitulo chama-se Mapa de Karnaugh e é o indicado para a minimização de expressões de até 4 variáveis.

1.5.1 - Expressões Booleanas de Soma de Produtos

Consideremos as combinações de variáveis que irão gerar uma saída 1 na tabela verdade abaixo (linhas 1 e 3).
Da linha 1, dizemos que “um não A E um não B irá gerar uma saída 1”. Da linha 3, dizemos que “um A E um não B irá gerar uma saída 1”.
Estas duas combinações possíveis são depois submetidas juntas a uma operação OU para formar a expressão booleana completa da tabela verdade ($S = \overline{A} . B + A . \overline{B}$)

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

$\overline{A} . \overline{B}$

$A . \overline{B}$

logo:
$S = \overline{A} . \overline{B} + A . \overline{B}$

A expressão resultante é chamada de soma de produtos ou forma de termos mínimos = (mintermos).
Considere agora a equação S = AB + BC + AC. Vamos colocá-la sob a forma de termos mínimos (soma de produtos). Para isto, devemos proceder como indicamos a seguir.

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

Note que os termos (A + A), (B + B) e (C + C) são iguais a 1 e, multiplicados por cada parcela da equação não a alteram.
Retirando-se os parênteses e ordenando as parcelas, temos a equação sob a forma de termos mínimos.

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

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

É importante ressaltar que as duas equações possuem a mesma Tabela Verdade e, portanto, a mesma função lógica.

A obtenção da forma' de termos mínimos é necessária na simplificação por Mapa de Karnaugh e na construção da Tabela Verdade, a partir da equação Booleana. Observe mais uma vez através do exemplo a seguir, as etapas para obtenção da equação na forma de termos mínimos.

$$T = \overline{XY} + XYZ + Z$$

$$T = \overline{XY}.(Z + \overline{Z}) + XYZ + Z.(X + \overline{X}).(Y + \overline{Y})$$

$$T = \overline{X}\overline{Y}Z + X\overline{Y}\,\overline{Z} + XYZ + ZXY + Z\overline{XY} + Z\overline{X}Y + Z\overline{X}\,\overline{Y}$$

$$T = \overline{X}\overline{Y}Z + X\overline{Y}\,\overline{Z} + XYZ + \overline{X}YZ + \overline{X}\,\overline{Y}Z$$

Na forma de termos, cada parcela da equação é equivalente a 1 na tabela verdade.
Logo, a tabela verdade da equação acima pode ser facilmente obtida, como mostrado a seguir.

$$T = \overline{X}\overline{Y}Z + X\overline{Y}\,\overline{Z} + XYZ + \overline{X}YZ + \overline{X}\,\overline{Y}Z$$

X	Y	Z	T	
0	0	0	0	
0	0	1	1	$\longrightarrow \overline{X}\,\overline{Y}Z$
0	1	0	0	
0	1	1	1	$\longrightarrow \overline{X}YZ$
1	0	0	1	$\longrightarrow X\overline{YZ}$
1	0	1	1	$\longrightarrow X\overline{Y}Z$
1	1	0	0	
1	1	1	1	$\longrightarrow XYZ$

1.5.2 - EXPRESSÕES BOOLEANAS DE PRODUTOS DE SOMAS.

As equações lógicas podem também ser implementadas sob a forma de produto de somas, também chamado de termos máximos (maxterm).

A obtenção da equação de termos máximos a partir da tabela verdade, é feita como mostrado no exemplo a seguir.

X	Y	S	
0	0	1	
0	1	1	
1	0	0	
1	1	0	$\longleftarrow (\overline{X} \pm Y)$

⟵ (X + Y) logo, $S = (X + Y) . (X + Y)$

Note que cada termo da equação é a representação de um "0" da tabela verdade. Vejamos agora como é possível obter-se a equação de termos máximos a partir de uma equação Booleana qualquer.

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

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

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

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

Note que os termos $(C . \overline{C})$, $(B . \overline{B})$, $(A . \overline{A})$ são iguais a 0 e somados a cada parcela da equação não alteram a mesma.
A tabela verdade pode ser facilmente construída, através da forma de termos máximos, como mostrado a seguir.

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

A	**B**	**C**	**S**	
0	0	0	0	⟵ (A + B + C)
0	0	1	0	⟵ (A + B + $\overline{C}$)
0	1	0	0	⟵ (A + $\overline{B}$ + C)
0	1	1	1	
1	0	0	1	
1	0	1	1	
1	1	0	0	⟵ ($\overline{A}$ + $\overline{B}$ + C)
1	1	1	1	

Naturalmente a tabela acima é válida tanto para a forma de produto de somas, quanto para a equação original.
O exemplo a seguir apresenta, mais uma vez, as etapas necessárias para a obtenção da equação de termos máximos.

$G = (E + \overline{F} + D) . \overline{F}$

$G = (E + \overline{F} + D) . (\overline{F} + (E . \overline{E}) + (D . \overline{D}))$

$G = (E + \overline{F} + D) . [(\overline{F} + E + D) . (\overline{F} + E + \overline{D}) . (\overline{F} + \overline{E} + D) . (\overline{F} + \overline{E} + \overline{D})]$

$G = (E + \overline{F} + D). (E + \overline{F} + \overline{D}). (\overline{E} + \overline{F} + D). (\overline{E} + \overline{F} + \overline{D})$

E	F	D	G	
0	0	0	1	
0	0	1	1	⟵ $(E + \overline{F} + D)$
0	1	0	0	⟵ $(E + \overline{F} + \overline{D})$
0	1	1	0	
1	0	0	1	
1	0	1	1	⟵ $(\overline{E} + \overline{F} + D)$
1	1	0	0	⟵ $(\overline{E} + \overline{F} + \overline{D})$
1	1	1	0	

OBSERVAÇÃO:
Em alguns livros, as expressões de soma de produtos e produtos de soma são encontradas com o nome de forma canônica.

1.5.3 - MAPA DE KARNAUGH DE DUAS VARIÁVEIS
Considere a tabela verdade abaixo:

X	Y	Z	
0	0	0	
0	1	0	
1	0	1	⟵ $X\overline{Y}$
1	1	1	⟵ XY

A equação de termos_ mínimos será:

$$Z = XY + X\overline{Y}$$

Iremos preencher a tabela abaixo com os valores da variável de saída Z, para uma das combinações de entrada.

Mapa de Karnaugh de 2 variáveis

	X	$\overline{X}$
Y		
$\overline{Y}$		

Da tabela verdade, obtemos o seguinte:

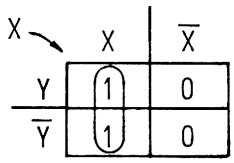
$\overline{X} \cdot \overline{Y} = 0$
$X \cdot \overline{Y} = 0$
$X \cdot Y = 1$
$X \cdot Y = 1$

Logo, preenchendo o mapa, obtemos:

	X	$\overline{X}$
Y	1	0
$\overline{Y}$	1	0

Quaisquer dos quadrados que possuam nível lógico 1 podem ser combinados de forma a se obter uma variável simples. No exemplo, temos:
O conjunto de 1's representa a variável X, pois:

$X \cdot Y + X \cdot \overline{Y} = X(Y + \overline{Y}) = X \cdot 1 = X$
Logo temos, que: $Z = X$
Podemos afirmar que a equação está completa e minimizada, porque todos os 1's do Mapa de Karnaugh foram equacionados.

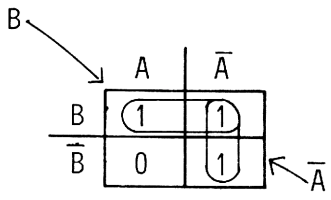


Considere agora o mapa da equação a seguir:

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

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

Podemos reunir os 1's em 2 grupos de 2, como mostrado a seguir:



Logo temos:

$Y = \overline{A} + B$

Novamente temos a equação completa e minimizada, uma vez que todos os 1's foram equacionados.

Observemos o próximo exemplo:

$K = I\,J + \overline{I}\,\overline{J}$

	I	$\overline{I}$
J	1	0
$\overline{J}$	0	1

Nesta situação não é possível formar grupos, uma vez que não existem 1's adjacentes. Sendo assim, cada 1 constitui seu próprio grupo e, portanto, a equação já está em sua forma minimizada.

1.5.4 - MAPA DE KARNAUGH DE TRÊS VARIÁVEIS.

Considere a equação :

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

Representando no Mapa de Karnaugh, temos:

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

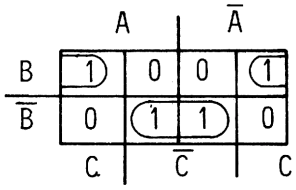
A princípio, consideremos os grupos apresentados a seguir:

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

Equacionando-se cada grupo, temos:

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

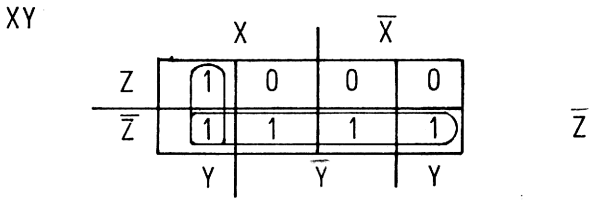
Esta equação está completa porque todos os 1's foram agrupados e equacionados.
Entretanto, a equação não está minimizada porque é possível diminuir o número de grupos feitos.
O Mapa de Karnaugh de 3 variáveis é na realidade montado no espaço como se fosse um cilindro. Portanto, o que vemos no papel é a planificação do cilindro. Sendo assim, a lateral direita do mapa está na realidade conectada à lateral esquerda, de modo que podemos formar os seguintes grupos:



$Y = BC + \overline{B}\,\overline{C}$

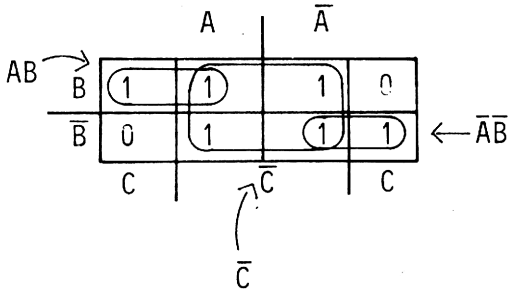
Esta é a equação minimizada.
Procure analisar os exemplos dados a seguir:
Exemplos:

a) $T = XYZ + XY\overline{Z} + X\overline{Y}\,\overline{Z} + \overline{X}YZ + \overline{X}\,\overline{Y}Z$

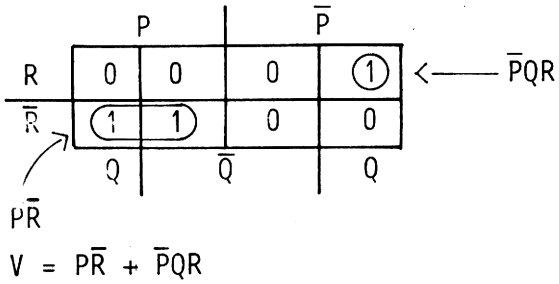


$T = XY + \overline{Z}$

b) $S = ABC + AB\overline{C} + \overline{A}BC + A\overline{B}C + \overline{A}\,\overline{B}C + \overline{A}B\overline{C}$



c) V = PQ$\overline{R}$ + P$\overline{Q}$$\overline{R}$ + $\overline{P}$QR

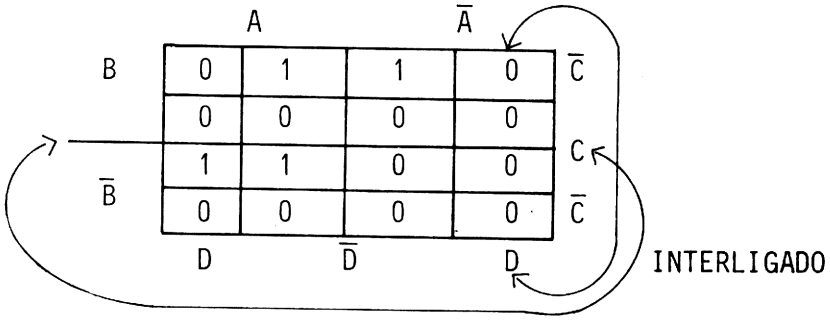


Com 3 variáveis, os grupos de 1's podem conter 1, 2 ou 4 elementos adjacentes, como apresentado nos exemplos anteriores:
Para se obter o circuito correto e minimizado deve-se observar o seguinte:
1 - Cada 1 do mapa deve pertencer a pelo menos 1 grupo.
2 - Cada grupo deve conter o maior número de elementos possível.
3 - O número de grupos deve ser o menos possível.
4 - Não são permitidos grupos transversais.
5 - Grupos de 4 "1's "representam um termo de 1 variável.
6 - Grupos de 2 "1's "representam um termo de 2 variáveis.
7 - Grupos de 1 "1 "representam um termo de 3 variáveis.

1.5.5 - MAPA DE KARNAUGH DE QUATRO VARIÁVEIS.

Para simplificação de 4 variáveis, todas as regras mencionadas até agora são válidas. Nesta situação os grupos de 1's podem conter 1, 2, 4 ou 8 elementos adjacentes.
As faces laterais direita e esquerda do mapa estão conectadas, assim como as faces superior e inferior, como demonstrado a seguir:

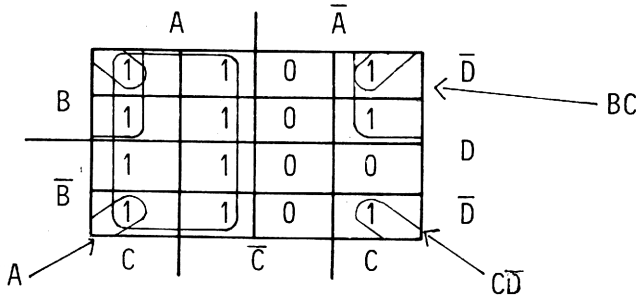
INTERLIGADO



Analise os exemplos dados a seguir:

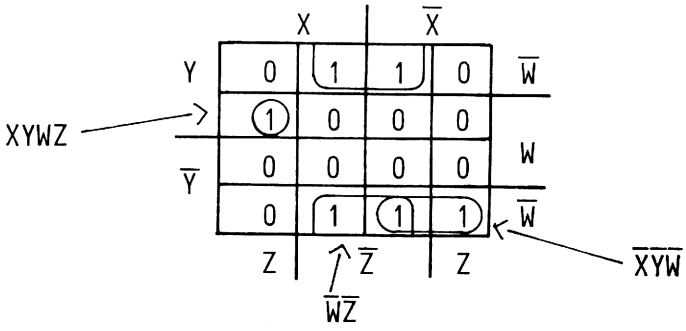
Exemplos:

a)



$Y = A + BC + C\bar{D}$

b)



Observe que o Mapa de Karnaugh de 4 variáveis temos a seguinte relação entre o número de elementos de cada grupo e o número de variáveis de cada termo:

- Grupos de 8 "1's" representam 1 termo de 1 variável.
- Grupos de 4 "1's" representam 1 termo de 2 variáveis.
- Grupos de 2 "1's" representam 1 termo de 3 variáveis.
- Grupos de 1 "1" representam 1 termo de 4 variáveis.

Vejamos agora um exemplo completo de simplificação através do Mapa de Karnaugh.
Exemplo: minimize a equação dada a seguir através do Mapa de Karnaugh.

$$Y = AC + ABD + B\bar{C}\bar{D} + \bar{A}BD$$

Solução:
Primeiramente devemos obter a equação de termos mínimos. Para isto, procedemos da seguinte maneira:

$Y = AC \cdot (B + \overline{B}) \cdot (D + \overline{D}) + ABD \cdot (C + \overline{C}) + B\overline{CD} \cdot (A + \overline{A}) + (\overline{A}BD) \cdot (C + \overline{C})$

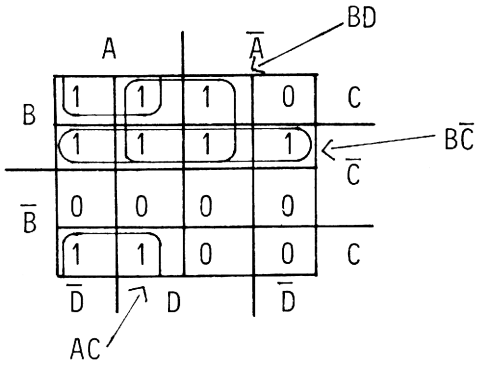
$Y = ACBD + AC\overline{B}D + AC\overline{B}D + AC\overline{BD} + ABDC + ABD\overline{C} + AB\overline{CD} + \overline{A}B\overline{CD} + \overline{A}BDC + \overline{A}BD\overline{C}$

$Y = ABCD + AB\overline{C}D + A\overline{B}CD + A\overline{B}C\overline{D} + AB\overline{C}D + AB\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + \overline{A}B\overline{C}D$

Com a equação na forma de soma de produtos podemos preencher o mapa.

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

O próximo passo é a obtenção dos grupos de "1's", lembrando-se das regras já descritas.

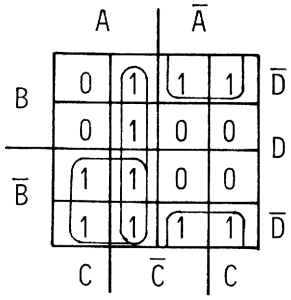


Finalmente, equacionamos cada grupo que foi somado para obtermos a equação final.

$Y = AC + BD + B\overline{C}$

1.5.6 - RESOLUÇÃO DO MAPA DE KARNAUGH ATRAVÉS DE PRODUTOS DE SOMAS.

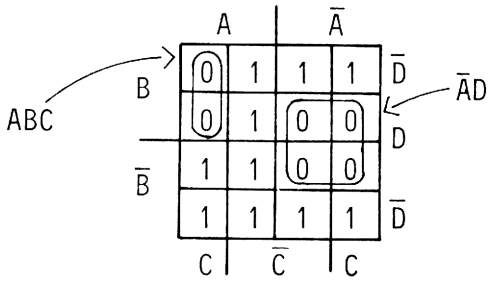
Até agora todas as equações minimizadas foram transformadas primeiramente em soma de produtos.
Considere entretanto, o Mapa de Karnaugh a seguir:



A equação minimizada é:

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

Vamos agora formar grupos de "0's" no lugar de grupos de "1's".



Como os grupos de "0's" foram equacionados, temos o complemento da saída ou seja:

$$Y = ABC + \bar{A}D$$

Para se obter a equação original, basta completar os dois lados da equação, como apresentado a seguir:

$$\bar{\bar{Y}} = \overline{\overline{ABC + \bar{A}D}}$$

$$Y = \overline{ABC} + \overline{\bar{A}D}$$

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

Observe que a equação final é um produto de somas.
Não é possível determinar qual das duas equações finais (soma de produto ou produto de somas), será mais fácil de implementar. Normalmente se resolve pelos dois métodos e seleciona-se a equação mais conveniente para a dada aplicação.

Temos então que, para obter a equação final sob a forma de produto de somas, primeiramente equacionam-se os grupos de "0's", seguindo as mesmas regras. A seguir, complementa-se o resultado obtido.

1.5.7 - MINIMIZAÇÃO EM EXPRESSÕES COM SAÍDAS NÃO ESPECIFICADAS (DON'T CARES)

É comum necessitarmos de circuitos lógicos em que algumas das combinações de saída são irrelevantes, ou seja, podem assumir qualquer um dos dois estados lógicos.
Considere por exemplo, que necessitamos de projetar um circuito que, a cada número binário de entrada (de 0 a 4), acione duas saídas consecutivas de suas cinco saídas.

Note que são necessários 3 bits de entrada para se representar de 0 a 4. Como se pode representar até "7" (111B) com bits, os números 5(101B), 6(110B) e 7(111B) não estão considerados no circuito. Portanto, não interessa o estado que as saídas vão assumir, quando ocorrer alguma destas situações.
Para desenvolvermos o circuito, devemos primeiramente montar a Tabela Verdade.

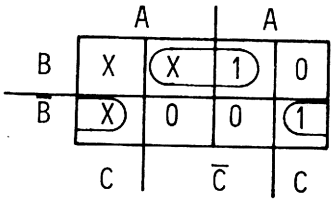
A	**B**	**C**	**0**	**1**	**2**	**3**	**4**
0	0	0	1	0	0	0	1
0	0	1	1	1	0	0	0
0	1	0	0	1	1	0	0
0	1	1	0	0	1	1	0
1	0	0	0	0	0	1	1
1	0	1	X	X	X	X	X
1	1	0	X	X	X	X	X
1	1	1	X	X	X	X	X

Para as condições de entrada (101B), (110B) e (111B) foi colocado um X em cada uma das saídas, indicando que estas podem assumir qualquer estado.
Como este circuito contém várias saídas, devemos minimizar cada uma delas em separado.
Vamos, como exemplo, minimizar o circuito para a saída "1". Repetindo a tabela para esta saída, temos:

A	**B**	**C**	**"1"**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	X
1	1	0	X
1	1	1	X

$\longleftarrow \overline{A}B\overline{C}$
$\longleftarrow ABC$

Cada uma das situações de saída será agora transportada para o Mapa de Karnaugh, como mostrado a seguir:



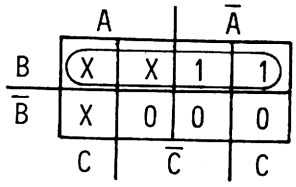
Como as saídas "X" podem, assumir qualquer estado lógico, os grupos foram feitos de tal forma que a equação total fosse simplificada, ou seja, o menor número de grupos do maior tamanho possível. A equação de saída será portanto:

$$"1" = \overline{B}C + B\overline{C}$$

Vamos repetir o processo para a saída "2".

A	B	C	"2"	
0	0	0	0	
0	0	1	0	
0	1	0	1	← $\overline{A}B\overline{C}$
0	1	1	1	← $\overline{A}BC$
1	0	0	0	
1	0	1	X	
1	1	0	X	
1	1	1	X	

Mapa de Karnaugh



Equação de Saída

"2" = B

Note que as saídas não especificadas são feitas 1 ou 0 de modo a completar um ou mais grupos no mapa.

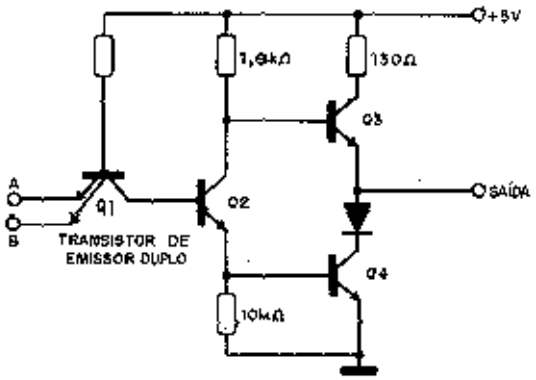
FAMÍLIAS LÓGICAS

1. INTRODUÇÃO

As famílias lógicas correspondem a grupos de tecnologias empregadas na construção dos circuitos integrados (CI) digitais.
Os CIs numa família são ditos compatíveis e podem ser facilmente conectados pois possuem características comuns como: faixa de tensão de alimentação, velocidade de operação, níveis de tensão de entrada, potência de dissipação, fan-out (fator de carga de saída = limitação de quantas portas podem ser excitadas por uma única saída).
Um grupo de famílias é produzido usando a tecnologia bipolar. Estes CIs contém partes comparáveis a transistores bipolares, diodos e resistores. Um outro grupo de famílias de CI digital usa tecnologia de semicondutor metal- óxido (MOS).

2. FAMÍLIA TTL (Transistor- Transistor- Logic)

Internamente, os componentes desta série são elaborados com a integração de transístores bipolares e na entrada observamos a presença de um transístor com emissor múltiplo.



Porta Nand TTL

É apresentada em duas séries: 54 e 74 .
A série 54 tem uma faixa maior de temperatura (-55ºC a +125º) e segue especificações militares.
A série 74 é de uso geral, operando na faixa de temperatura de 0ºC a +70ºC.
Os circuitos integrados da família TTL se caracterizam por exigir uma tensão de alimentação de 5V.
Para que a entrada reconheça o nível lógico baixo, é preciso que a tensão seja de 0 a 0,8V.
Analogamente, uma entrada alta deve estender-se de 2 a 5V.
Encontramos dentro da família de integrados TTL centenas de funções lógicas, desde portas lógicas, flip flops, decodificadores, comparadores, etc.

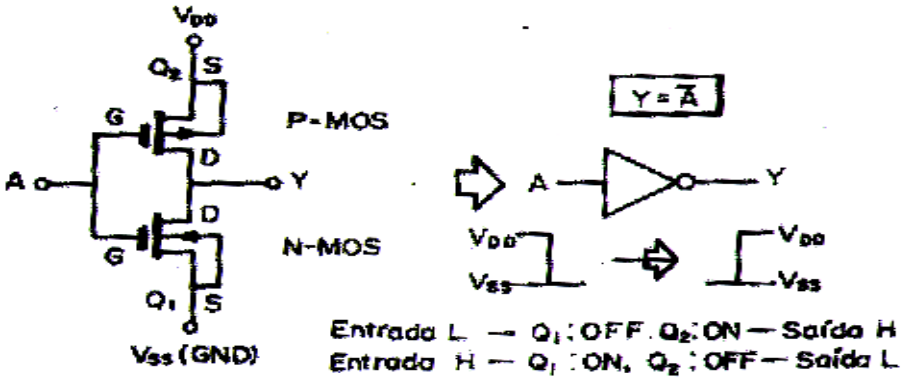
Podemos classificar um componente da família TTL em quatro categorias, conforme o número de portas que o mesmo integra:

- SSI (Small Scale Integration) - integração em pequena escala: com cerca de doze portas lógicas incorporadas.
- MSI (Medium Scale Integration) - integração em média escala: até 99 portas lógicas incorporadas.
- LSI (Large Scale Integration) - integração em grande escala: de 100 até 1000 portas lógicas.
- VLSI (Very Large Scale Integration) - integração em escala muito alta: acima de 1000 portas lógicas.

Mesmo dentro desta família existem “sub-famílias”, versões, que mantém as especificações de tipos,ou seja, das funções que encontramos no circuito integrado, mas variam de velocidade e corrente consumida.
Dentre todas as versões TTL disponíveis, a versão Standard (padrão) é a de mais baixo custo e a que possui a maior variedade de funções disponíveis.
A versão Low Power (indicada pela letra L) é a de mais baixo consumo de corrente, e também a mais lenta.
A versão High Speed (letra H) apresenta uma velocidade maior, com consumo bem elevado.
A versão Schottky (letra S) é a mais veloz de todas e com consumo médio (substitui a versão H).
A versão Low Power Schottky (letra LS) oferece a mesma velocidade da versão Standard, com menor consumo. É a mais utilizada para quase todas as aplicações.

3. FAMÍLIA CMOS (Complementary MOS)

A letra C do nome CMOS significa complementar (complementary) e é um circuito estruturado pelo MOS-FET de canal N em oposição a um MOS-FET de canal P.



Circuito fundamental do CMOS (inversor)

Os circuitos integrados da família CMOS são encontrados em duas séries principais:

- 54C / 74C
- 4000

A série 54C / 74C compreende os circuitos integrados que correspondem diretamente a seus homönimos da série 54 / 74 TTL. Assim, um CI 74C00 é uma versão CMOS do CI 7400.

A série 4000 é encontrada nas versões A (Standard) e B (bufferizada) , esta última com mais potëncia. A série B se caracteriza por ter um isolamento maior entre a entrada e a saída (buffer) e assim possibilitar melhor desempenho nas aplicações conjuntas (resistëncia de entrada na faixa de $10^{12}\Omega$)

O CI CMOS possui várias vantagens, tais como:

- podem ser alimentados com tensões de 3 a 15V para a série A e de 3 a 18V para a série B;
- apresentam excelente imunidade a ruídos;
- consumo de potëncia extremamente baixo;
- fan-out (número de portas acionadas por uma única porta): no TTL é igual a 10 e no CMOS é tipicamente em torno de 50.

MULTIVIBRADORES BIESTÁVEIS (FLIP-FLOP)

1. INTRODUÇÃO

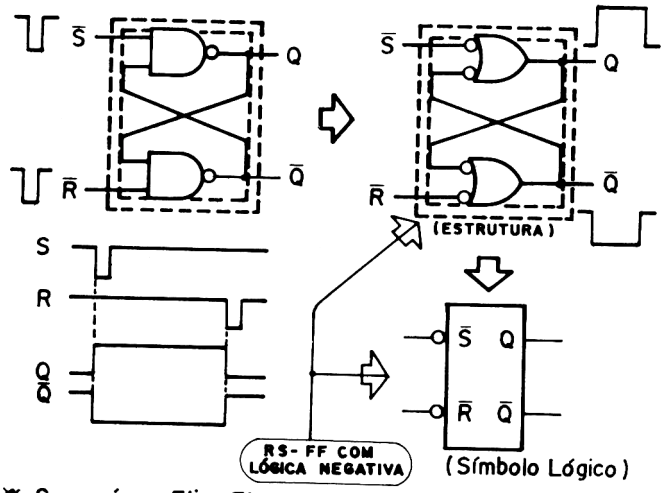
FLIP-FLOP é também chamado de multivibrador BIESTÁVEL, e como possui 2 estados de estabilidade, pode memorizar informações de 1 BIT.
O FLIP-FLOP (abreviado temos FF), são interligados para formar circuitos lógicos para armazenamento, temporização, contagem e sequenciação.
O FLIP-FLOP pode ser classificado, do ponto de vista da função lógica, conforme segue:

- FLIP-FLOP RS (FF - RS)
- FLIP-FLOP RST (FF - RST)
- FLIP-FLOP JK (FF - JK)
- FLIP-FLOP D (D - FF) (FF – D)
- FLIP-FLOP T (T - FF) (FF – T)

Neste capítulo, iremos estudar o funcionamento de cada um dos tipos mostrados acima.

2 - FLIP-FLOP R. S

Na figura 3 apresentamos a estrutura do FLIP-FLOP RS utilizando 2 portas NAND.
Como sabemos, a saída de uma porta NAND só será L quando todas as entradas forem H. E ainda, as entradas são $\overline{S}$ e $\overline{R}$, o que significa que este circuito funciona com pulso negativo (ou nível L).
Obs.: - L (do inglês LOW, baixo) = nível lógico 0(zero);
- H (do inglês HIGH, alto) = nível lógico 1;
- $\overline{Q}$ = saída normal;
- Q = saída complementar.



* Como é um Flip-Flop com entrada ativa em L é chamado de $\bar{R}\bar{S}$ ($\bar{R}\bar{S}$-FF).

Fig. 3 - Flip-Flop RS com Porta Nand.

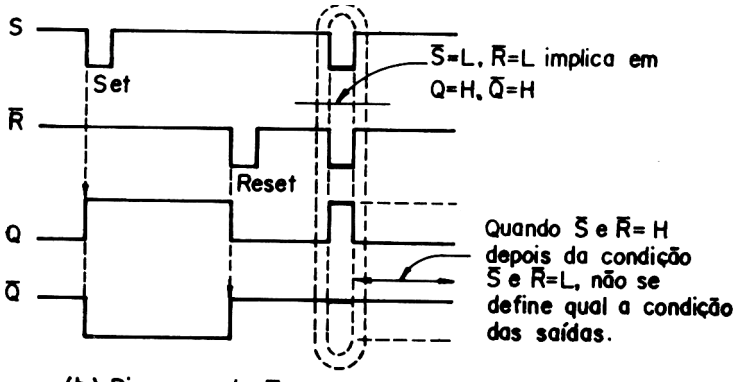
Vamos, agora, estudar o funcionamento lógico deste circuito observando a figura 4, que mostra o diagrama de tempo e a tabela verdade, e a figura 5 que indica o seu estado de funcionamento.

ENTRADA		SAÍDA		
$\bar{R}$	$\bar{S}$	Q_{n+1}	$\bar{Q}_{n+1}$	
H	H	Q_n	$\bar{Q}_n$	→ Mantem estado anterior.
H	L	H	L	→ Set.
L	H	L	H	→ Reset.
L	L	*	*	→ Indefinido.

Q_n : Estado anterior a entrada de RS. * Indefinido.

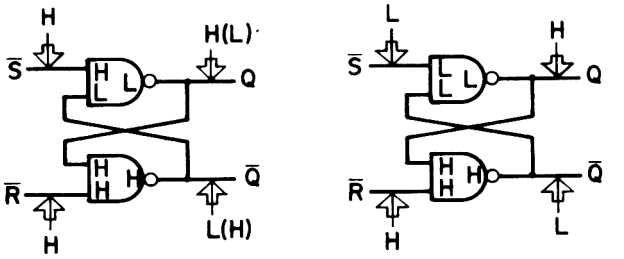
Q_{n+1} : Estado posterior a entrada de RS.

Fig. 4 - (a) Tabela Verdade

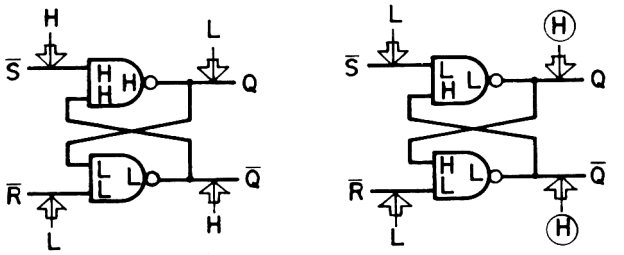


(b) Diagrama de Tempo.

Quando as entradas $\overline{S}$ e $\overline{R}$ estiverem ambas com nível H, teremos a condição de estabilidade, isto é, a saída manterá o estado anterior. Quando L é aplicado à entrada S (e H na entrada R), a saída Q passará para H (e Q passará para L), e o estado de SET será estável. Em seguida, se L é aplicado à entrada R (e H em S), a saída Q será L (e Q será H) , e o estado RESET será estável. Se for aplicado nível L às 2 entradas, teremos a condição de entrada proibida, pois a saída será indeterminada.



(1) $\overline{S}$=H, $\overline{R}$=H (Conserva-se o estado anterior). (2) $\overline{S}$=L, $\overline{R}$=H (Estado de set).



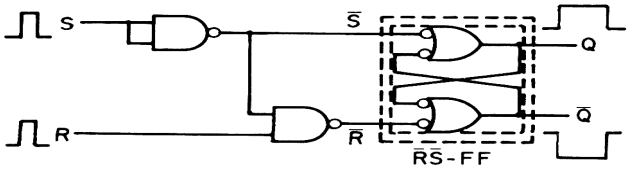
(3) $\overline{S}$=H, $\overline{R}$=L (Estado de reset). (4) $\overline{S}$=L, $\overline{R}$=L (Entrada proibida).

Fig. 5- Estado de funcionamento do RS-FF com Portas Nand.

Conforme podemos observar , o FLIP-FLOP funciona com lógica negativa. Portanto o FLIP-FLOP é denominado de R.S-FF. No caso de querer que um FLIP-FLOP de lógica positiva funcione com lógica negativa, basta colocarmos 1 inversor em cada entrada.

3.2 - FLIP-FLOP com preferência SET

Em todos os FLIP-FLOPS explicados até agora, sempre houve uma condição de entrada proibida que não identificava uma saída estável. A figura 6 apresenta a estrutura do circuito e a tabela verdade do FLIP-FLOP com preferência SET. Este FLIP-FLOP tem a característica de assumir o estado de SET de saída, quando for aplicado à entrada uma condição de entrada proibida. Isso acontece porque quando é aplicado nível H à entrada S, será inserido L na porta NAND do lado da entrada R, impedindo que a entrada RESET atue no FLIP-FLOP . Desta forma, haverá uma preferência pelo estado SET, e o RESET só irá ocorrer quando não houver sinal de SET (S com nível H).



(a) Estrutura

ENTRADA		SAÍDA		
R	S	Q_{n+1}	$\bar{Q}_{n+1}$	
L	L	Q_n	$\bar{Q}_n$	→ Mantem o estado anterior.
L	H	H	L	→ Set.
H	L	L	H	→ Reset.
H	H	H	L	→ Set preferencial.

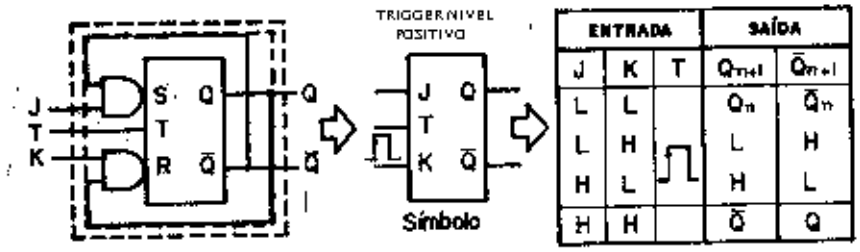
Q_n : Estado anterior a entrada R e S.
Q_{n+1}: Estado posterior a entrada R e S.

(b) Tabela Verdade

Fig. 6 - Estrutura do circuito e Tabela Verdade do RS - FF com preferência set.

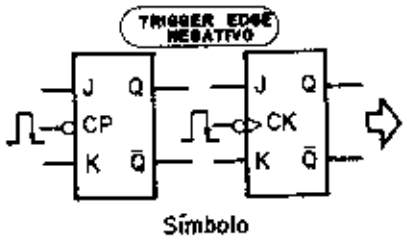
4 - FLIP-FLOP JK (JK - FF)

Conforme demonstrado na figura 7(a), o FLIP-FLOP JK é formado pelas entradas J, K e de Clock T (também representada por C, CK ou CP), e as saídas Q e Q. As entradas J e K correspondem respectivamente às entradas S e R do RS-FF. O FLIP-FLOP JK tem, além da função do RST-FF, a função de inverter o estado das saídas através do pulso de Clock, quando as 2 entradas estiverem com nível H. Isto é, não há condição de entrada proibida para o FLIP-FLOP JK.



ENTRADA			SAÍDA	
J	K	T	Q_{n+1}	$\bar{Q}_{n+1}$
L	L		Q_n	$\bar{Q}_n$
L	H		L	H
H	L		H	L
H	H		$\bar{Q}$	Q

* O terminal de clock T, tambem é denominado de C, CK ou CP.
(a) Funciona na subida do pulso de ckock.



ENTRADA			SAÍDA		
J	K	CK	Q_{n+1}	$\bar{Q}_{n+1}$	
L	L		Q_n	$\bar{Q}_n$	→ Mantem o estado anterior.
L	H		L	H	→ Reset
H	L		H	L	→ Set
H	H		$\bar{Q}$	Q	→ Inversão.

(b) Funciona na descida do pulso de ckock.

Fig. 7- Símbolo Lógico e Tabela Verdade do JK - FF.

Devido ao fato do FLIP-FLOP JK possuir todas as funções dos outros FLIP-FLOPS é chamado, também de Rei dos FLIP-FLOPS, e diz-se que sua denominação provém do J de Jack (forma popular de JOHN ou JACOB - homem) e K de KING (Rei).
O FLIP-FLOP, normalmente comercializado é o tipo TRIGGER EDGE NEGATIVO que funciona na descida do pulso de Clock, conforme indicado na figura 7 (b). Os que funcionam na subida do pulso de Clock é chamado de TRIGGER EDGE POSITIVO.
A figura 8 apresenta o diagrama de tempo do FLIP-FLOP JK do tipo TRIGGER EDGE NEGATIVO. Vejamos então o seu funcionamento lógico.

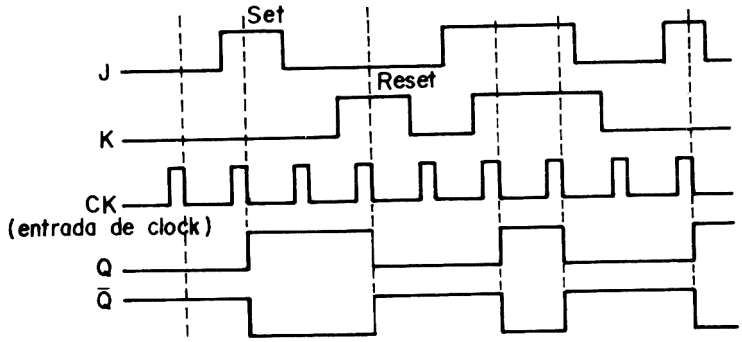


Fig. 8 - Diagrama de Tempo do JK- FF.

Quando as entradas J e K estiverem com nível L, e for inserido um pulso de Clock, a saída não se altera, mantendo o estado anterior. Se quando for aplicado um pulso de Clock, a entrada J estiver com L e a entrada K estiver com H, então a saída Q vai para nível L e $\overline{Q}$ para nível H.

Se agora, tivermos a entrada J com H e a entrada K com L, quanto for aplicado o pulso de Clock a saída Q vai para nível H e a $\overline{Q}$ vai para nível L. Na situação que tivermos as 2 entradas, J e K, com nível H, cada pulso de Clock aplicado fará com que as saídas invertam de nível, isto é, passem para o estado inverso do estado anterior ao pulso de Clock.

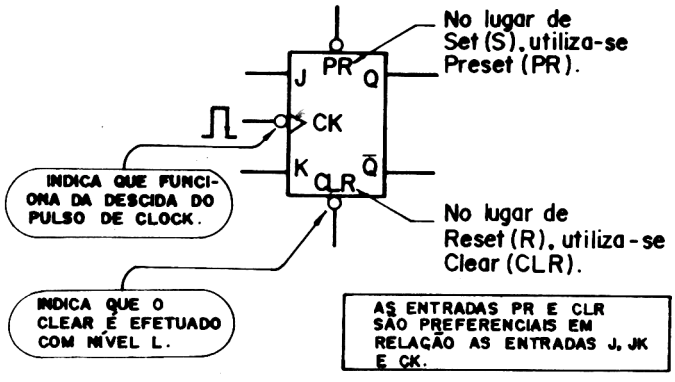


Fig. 9 - Terminais PR e CLR do JK- FF.

Resumindo o funcionamento lógico do FLIP-FLOP JK, teremos:

1) J = L, K = L: mesmo inserindo CK, Q e $\overline{Q}$ invariável
2) J = L, K = H: se CK é inserido, Q = L e $\overline{Q}$ = H RESET
3) J = H, K = L: se CK é inserido, Q = H e $\overline{Q}$ = L SET
4) J = H, K = H: se CK é inserido, Q e $\overline{Q}$ invertem o estado

O FLIP-FLOP JK possui ainda o terminal de entrada de PRESET (PR), e o terminal de entrada de CLEAR (CLR), conforme mostrado na figura 9. Utiliza-se a representação PRESET (PR) e CLEAR (CLR), ao invés de SET (S) e RESET (R), nas o significado é o mesmo. Isso quer dizer que para que funcione as entradas J, K e CK as entradas PR e CLR devem estar com nível H.

5 - FLIP-FLOP D (D-FF)

O FLIP-FLOP é do tipo de atraso (Delay) e conforme mostrado na figura 10, possui a entrada de dados D e a entrada de Clock CK. A saída apresenta um atraso de 1 pulso de Clock em relação à entrada, isto é, o sinal de entrada passa para a saída com certo tempo de atraso (no máximo de 1 ciclo de Clock), por isso é também chamado de FLIP-FLOP do tipo atraso.

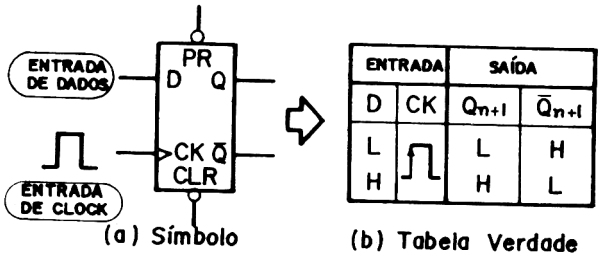


Fig. 10 - Símbolo Lógico e Tabela Verdade do D-FF

A figura 11 representa o diagrama de tempo. O FLIP-FLOP passa para a saída, o sinal de entrada anterior a subida do pulso de Clock. Isto é, o sinal de entrada é memorizado na subida do pulso de Clock, a saída Q passa para nível L. Se a entrada estiver com nível H quando for inserido um pulso de Clock, a saída Q terá nível H. Quando às entradas CLR e PR, estas são preferenciais a exemplo do que ocorre no FLIP-FLOP JK. Isto é, para que as entradas D e CK estejam liberadas, as entradas de PR e CLR devem estar com nível H.

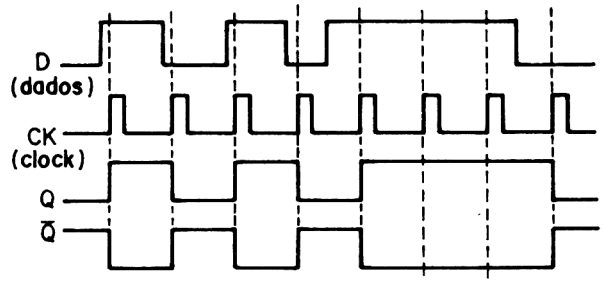


Fig. 11 - Diagrama de Tempo do D-FF.

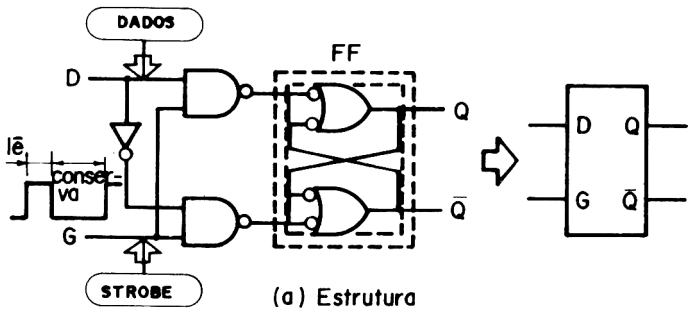
Resumindo o funcionamento lógico do FLIP-FLOP D, teremos:

1) D = L : se for inserido CK, Q = L e $\overline{Q}$ = H
2) D = H: se for inserido CK, Q = H e $\overline{Q}$ = L

Devido as características do FLIP-FLOP tipo D, de transferir a entrada para a saída com um pulso de Clock, e manter essa saída até o próximo pulso de Clock (memoriza o sinal de entrada), faz com que ele seja utilizado no circuito "LATCH" e em registradores de deslocamento (SHIFT REGISTER), que memorizam por algum tempo a informação da entrada.

5.1 - Circuito LATCH

O circuito "LATCH", é um circuito que memoriza durante determinado tempo, o sinal, ou partes do sinal, de entrada, o que em microcomputação é denominado de fazer "LATCH" de um sinal.
A figura 12 apresenta o circuito LATCH estruturado com portas NAND e a tabela verdade. O teminal D é o terminal de entrada de dados, e o teminal G é a entrada do sinal STROBE (significa sinal de "pinçamento"), que definirá o tempo de memória do sinal na saída.



(a) Estrutura

ENTRADA		SAÍDA		
D	G	Q_{n+1}	$\overline{Q}_{n+1}$	
X	L	Q_n	$\overline{Q}_n$	→ Mantem o estado anterior
L	H	L	H	
H	H	H	L	X : H ou L

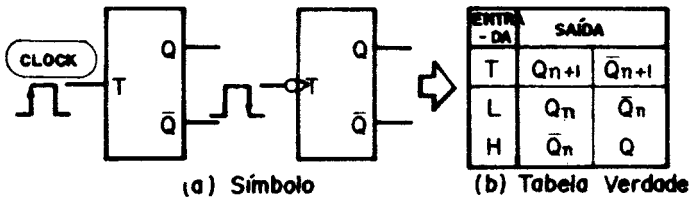
(b) Tabela Verdade

Fig. 12 - Estrutura e Tabela Verdade do circuito Latch.

Vejamos como funciona este circuito. Quando a entrada G for L, saída Q mantém seu estado anterior. Se G passar para H, o sinal de entrada (D) passa para a saída. Parece que o funcionamento é o mesmo do FLIP-FLOP D mas na realidade o D-FF faz a leitura da entrada D na subida do pulso de Clock e esse valor é mantido até que ocorra outra subida do pulso de Clock. Já o "LATCH", lê o sinal de entrada quando G e H mantém esse sinal enquanto G for L.

6 - FLIP-FLOP T (T-FF)

O FLIP-FLOP T é chamado de FLIP-FLOP TRIGGER ou FLIP-FLOP TOGGLE. Na figura 13, apresentamos o símbolo lógico e a tabela verdade. A figura 14 apresenta o diagrama de tempo.



ENTRA-DA	SAÍDA	
T	Q_{n+1}	$\bar{Q}_{n+1}$
L	Q_n	$\bar{Q}_n$
H	$\bar{Q}_n$	Q

(b) Tabela Verdade

Fig. 13 Símbolo Lógico e Tabela Verdade do T-FF.

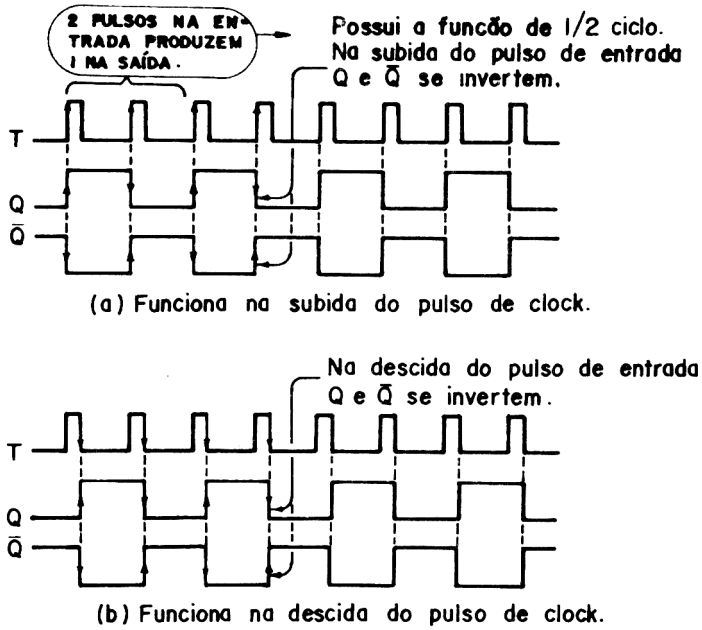


Fig. 14 - Diagrama de Tempo do T-FF.

Vejamos seu funcionamento : o FLIP-FLOP T inverte o estado de saída Q toda vez que é inserido um pulso de Clock. Conforme podemos observar no diagrama de tempo, são necessários 2 pulsos de entrada para se obter 1 pulso na saída. Isso significa que a saída é a metade da freqüência dos pulsos de entrada. Devido a isso, é utilizado no circuito de cálculo e no circuito divisor.
Não existe atualmente, um CI digital específico do FLIP-FLOP T. Para se obter o FLIP-FLOP T, devemos efetuar algumas modificações em outros FLIP-FLOPS, conforme demonstra a figura 15.

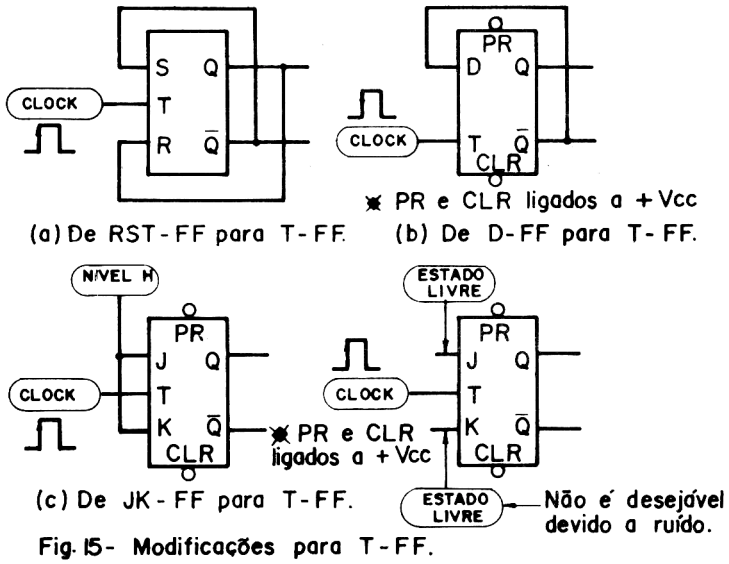


Fig. 15 - Modificações para T - FF.

A figura 15(a) representa um FLIP-FLOP RST onde a saída $\overline{Q}$ realimenta a entrada S, a saída $\overline{Q}$ realimenta a entrada R, e o sinal de entrada é aplicado em T, obtendo-se assim o T-FF.

A figura 15(b) representa um FLIP-FLOP T a partir de um D-FF onde a saída $\overline{Q}$ realimenta a entrada D e o sinal de entrada é aplicado em T.
Outra forma de se obter um FLIP-FLOP T, é apresentada na figura 15 (c) onde um FLIP-FLOP JK tem as entradas J e K ligadas com nível H, e a entrada é aplicada em T. As entradas J e K também podem ficar sem serem conectadas, que o FLIP-FLOP JK funciona como T-FF, mas isso torna o circuito mais vulnerável a ruído.

7 - FLIP-FLOP MASTER SLAVE (MASTER SLAVE FF)

O FLIP-FLOP MASTER SLAVE, conforme apresentado na figura 16, é formado por um FLIP-FLOP MASTER (mestre) e um FLIP-FLOP SLAVE (escravo).

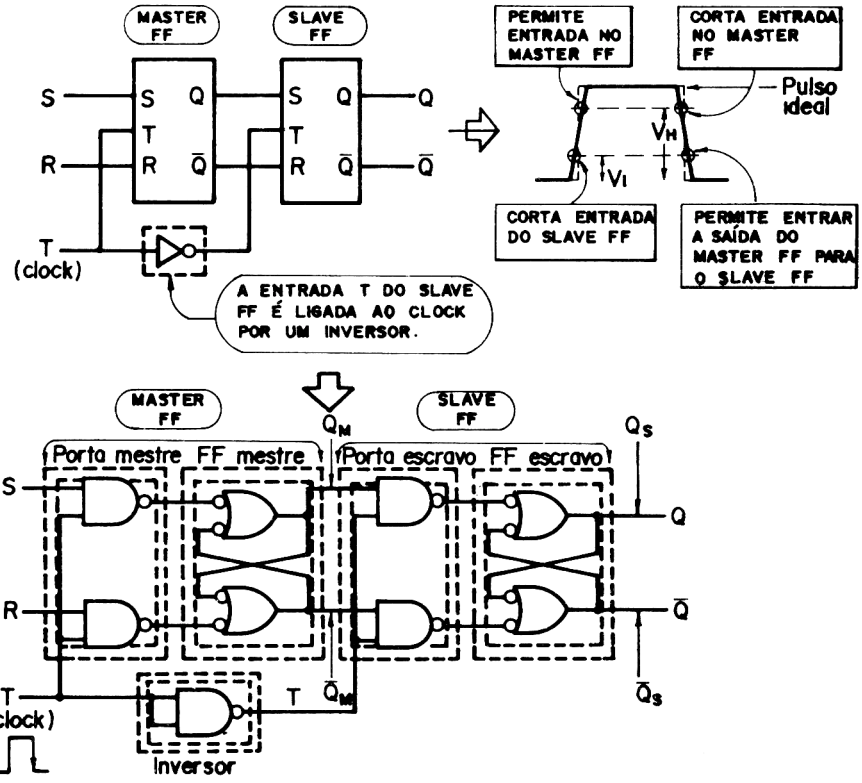


Fig. 16 - Símbolo Lógico e estrutura do Flip-Flop master-slaver.

O funcionamento do FLIP-FLOP obedece ao seguinte raciocínio: na subida do pulso de Clock (de L para H) o FLIP-FLOP MASTER lê o sinal de entrada e na descida do Clock (de H para L), o conteúdo da saída do FLIP-FLOP MASTER passa para a saída do FLIP-FLOP SLAVE.

Baseado no diagrama de tempo da figura 17, vamos agora estudar o funcionamento do FLIP-FLOP MASTER e do FLIP-FLOP SLAVE.
Quando o pulso de Clock T_1 é inserido (S = H e R = L), a entrada T passa para H fazendo com que o circuito NAND mestre fique ON (permite que o sinal das entradas passe). Isso faz que a saída Qm do FLIP-FLOP mestre passe de L para H, e Qm de H para L.

Neste instante T', que é a saída do inversor, está L e as saídas Qs e $\overline{Qs}$ do FLIP-FLOP escravo não se alteram. Quando a entrada T volta para o nível L, o circuito NAND mestre passa para OFF, e o circuito NAND escravo passa para ON pois T' está com nível H. Assim sendo, o conteúdo das saídas do FLIP-FLOP mestre (Qm e $\overline{Qm}$) passam para o FLIP-FLOP escravo fazendo com que Qs passe de L para H e $\overline{Qs}$ de H para L. Aplicando agora novo pulso de Clock, T_2, para as condições de entrada S = L e R = H, teremos T = H permitindo que o sinal de entrada passe, fazendo com que a saída Qm fique com o nível L e $\overline{Qm}$ fique com o nível H. Nesse instante T' está com nível L o que indica que o FLIP-FLOP escravo mantém a sua saída sem alteração. Assim que a entrada T volta para o nível L, o circuito NAND mestre passa para OFF e T' passa para H, fazendo com que Qm e $\overline{Qm}$ passem para Qs e $\overline{Qs}$. Desta forma, podemos observar que a cada pulso de Clock toda a operação se repete.

Baseado, então, no funcionamento e no diagrama de tempo, podemos afirmar que há um defasamento entre a leitura e memorização da entrada, pelo FLIP-FLOP mestre, e a recepção e saída pelo FLIP-FLOP escravo.

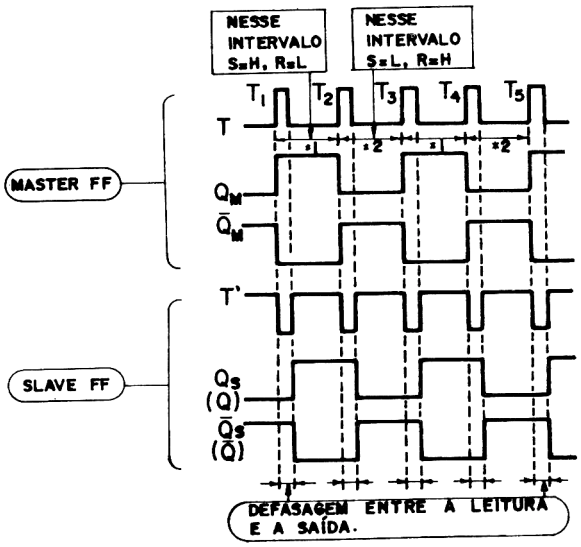


Fig. 17 - Diagrama de Tempo do Flip-Flop master - slave.

As figuras 18 e 19 representam a estrutura do FLIP-FLOP JK MASTER SLAVE com portas NAND.

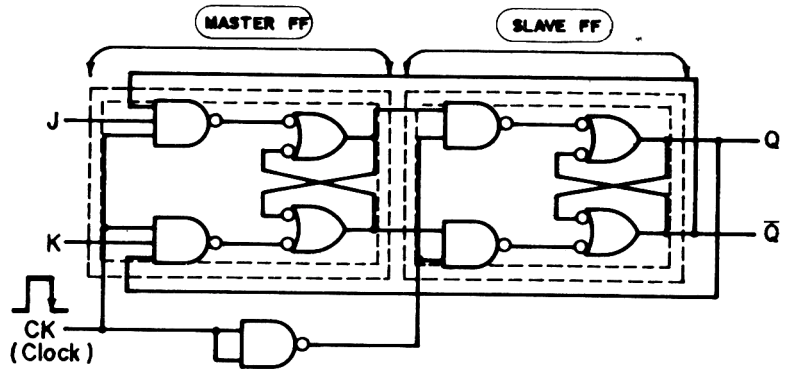


Fig. 18 - Master slave JK - FF.

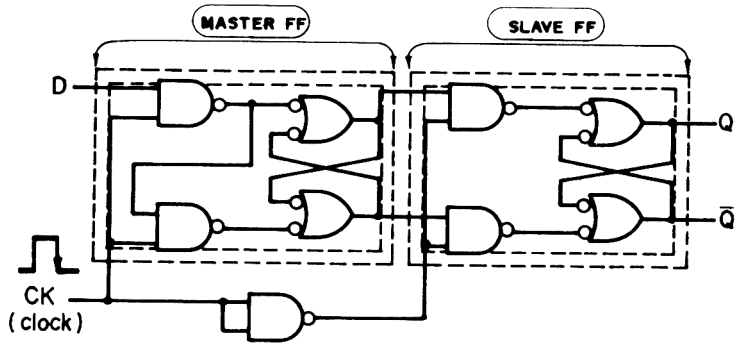


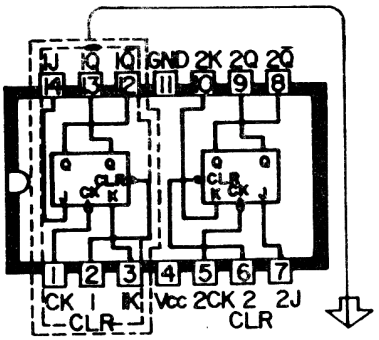
Fig. 19 - Master slave D - FF.

8 - CI'S DIGITAIS COM FLIP-FLOPS

Vamos, agora, escolher 2 CI's da Texas para explicar o funcionamento. O primeiro será o SN 7473 que é um FLIP-FLOP JK MASTER SLAVE e o segundo, o CI SN 7474 que é um FLIP-FLOP D com EDGE TRIGGER positivo.

8.1 - CI 7473 (Dual JK FLIP-FLOP Whith Clear)

Na figura 20, apresentamos a distribuição dos pinos, a tabela verdade e o circuito equivalente do 7473. Este CI é um FLIP-FLOP JK MASTER SLAVE com entrada de CLEAR. Se a entrada CLR estiver com nível L, a saída Q passa para L e Q para H, independente das entradas J, K e Ck. Com a entrada CLR em nível H, o FLIP-FLOP JK funciona como já foi explicado anteriormente.
Dentre os TTL que podem substituir o 7473, temos o M53273 da MITSUBISHI, TD3473 da TOSHIBA e HD2515 da HITACHI. Existe também o 7476 que possui 2 FLIP-FLOP JK com PRESET e CLEAR. A figura 21 apresenta a distribuição dos pinos e a tabela verdade. A entrada PRESET está indicada no retângulo pontilhado no circuito equivalente da figura 20.



ENTRADA				SAÍDA	
CLR	CK	J	K	Q	$\bar{Q}$
L	X	X	X	L	H
H	↑	L	L	Q_0	$\bar{Q}_0$
H	↑	H	L	H	L
H	↑	L	H	L	H
H	↑	H	H	TOGGLE	

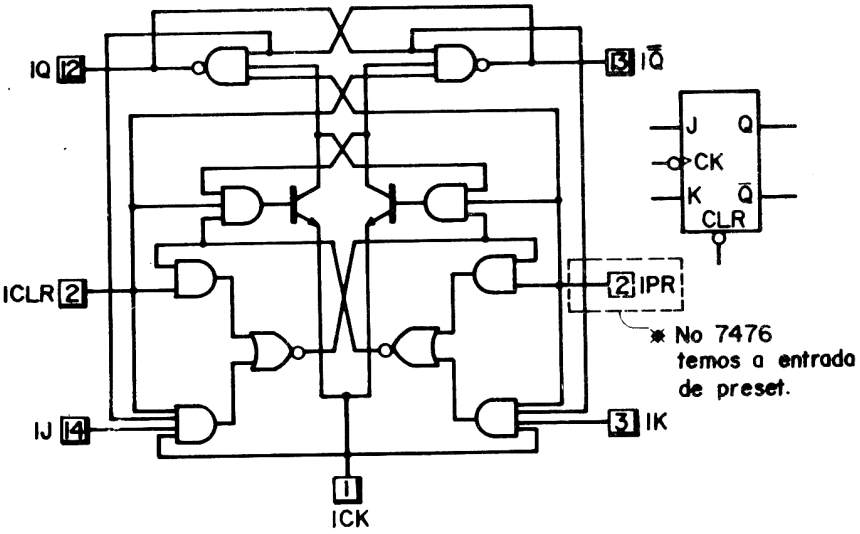
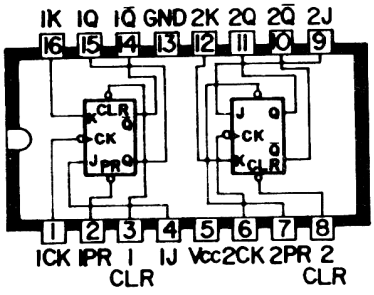


Fig. 20 - CI 7473 (2 Flip-Flops JK com clear).

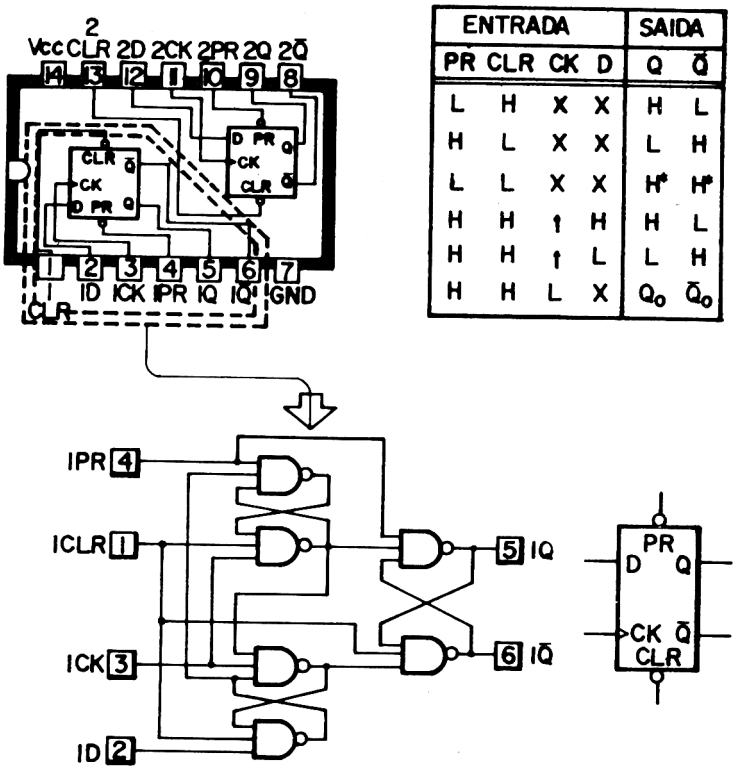


ENTRADA					SAÍDA	
PR	CLR	CK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↑	L	L	Q_0	$\bar{Q}_0$
H	H	↑	H	L	H	L
H	H	↑	L	H	L	H
H	H	↑	H	H	TOGGLE	

Fig. 21 - CI 7476 (2 JK - FF com preset e clear).

8.2 - CI 7474 - (Dual D - Tipe Positive - Edge - Trigger o Flip-Flops White Preset and Clear)

Na figura 22, apresentamos a distribuição dos pinos, o circuito equivalente e a tabela verdade do 7474. Este CI possui 2 FLIP-FLOPS tipo D com TRIGGER EDGE positivo com entradas de PRESET e CLEAR. O seu funcionamento é o mesmo já explicado anteriormente.
Dentre os CI's TTL que podem substituí-lo temos o M53274 da MITSUBISHI, o TD 3474 da TOSHIBA e o HD 2515 da HITASHI.



ENTRADA				SAIDA	
PR	CLR	CK	D	Q	Q̄
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\bar{Q}_0$

Fig. 22 - CI 7474

CONTADORES

1. INTRODUÇÃO

Os contadores são importantes circuitos eletrônicos digitais. Eles são circuitos lógicos seqüenciais porque a *temporização* é obviamente importante e porque eles necessitam de uma característica de *memória*. Os *contadores digitais* tem as seguintes características importantes:

1. Número máximo de contagens (módulo do contador).
2. Contagem para cima ou para baixo.
3. Operação assíncrona ou síncrona.
4. Funcionamento livre ou autoparada.

Como nos outros circuitos seqüenciais, são usados biestáveis (*flip-flop*) para construir contadores.

2. CONTADORES ASSÍNCRONOS (RIPPLE)

Os contadores digitais contarão apenas em binário ou em códigos binários. A Fig. 1 mostra a seqüência da contagem em binário desde 0000 até 1111 (0 a 15 em decimal). Um contador digital que contaria desde o binário 0000 até 1111 como indicado na tabela pode ser chamado de *contador de módulo 16*. O *módulo* de um contador é o número de contagens que ele completa. O termo "módulo" às vezes é abreviado para "mod". Este contador pode ser chamado de *contador mod-16*.

Contagem decimal	Contagem binária 8s 4s 2s 1s (D C B A)	Contagem decimal	Contagem binária 8s 4s 2s 1s (D C B A)
0	0 0 0 0	8	1 0 0 0
1	0 0 0 1	9	1 0 0 1
2	0 0 1 0	10	1 0 1 0
3	0 0 1 1	11	1 0 1 1
4	0 1 0 0	12	1 1 0 0
5	0 1 0 1	13	1 1 0 1
6	0 1 1 0	14	1 1 1 0
7	0 1 1 1	15	1 1 1 1

Fig. 1 - Seqüência da contagem de um contador de 4 bits.

Um diagrama lógico do contador mod-16 usando biestáveis *JK* é mostrado na Fig. 2. Notar primeiro que as entradas de dados J e K dos biestáveis são unidas a um 1 lógico. Isso significa que cada biestável está em seu modo chave (*toggle*). Cada pulso de clock (relógio) fará então com que o biestável chaveie para seu estado oposto. Notar também que a saída *Q* do FF1 (biestável 1) está conectada diretamente à entrada (*CK*) de clock da unidade seguinte (FF2) e

assim por diante. Os indicadores de saída (lâmpadas ou LEDs) são mostrados à direita na parte superior para monitorar a saída binária do contador. O indicador *A* é o LSB (bit menos significativo), enquanto *D* é o MSB (bit mais significativo).
O contador mod-16 na Fig.2 conta de acordo com a tabela na Fig. 1. Costuma-se analisar uma operação do contador usando *formas-de-onda* (diagramas de temporização). A Fig.3 é uma forma-de-onda do contador mod-16. A linha superior representa a entrada (*CK*) de clock no FF1. A linha inferior mostra a contagem binária nos indicadores. Notar que o contador binário está limpo, ou reajustado (*reset*) para 0000, à esquerda. Cada pulso de clock aumentará a contagem binária de 1 quando nos deslocarmos para a direita sobre o diagrama.

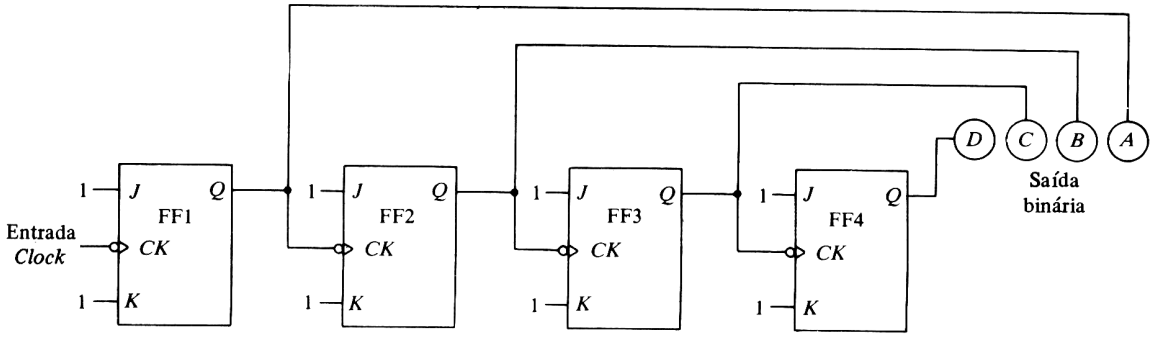


Fig. 2 - Um contador assíncrono de 4 bits.

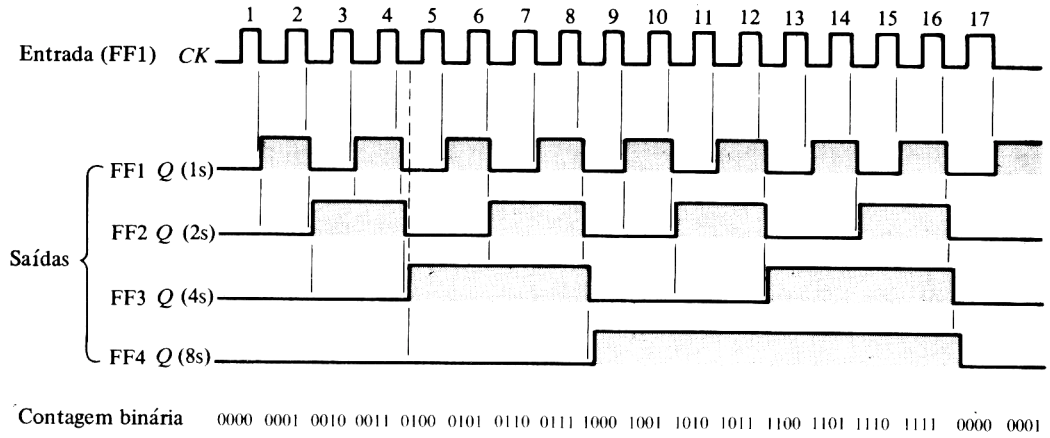


Fig. 3 - Diagrama de temporização de um contador assíncrono mod-16.

A bola na entrada (*CK*) de clock do biestável *JK* na Fig. 2 significa que a unidade chaveará na transição H-para-L (bordo posterior ou transição final) do pulso de clock. Observemos o pulso 1 de clock na Fig. 3. A transição H-pata-L chaveia FF1. A saída *Q* de FF1 vai de BAIXA para ALTA. A contagem binária é agora 0001.
Observemos o pulso 2 de relógio. O edge (bordo) posterior do pulso de relógio dispara FF1. Este FF1 chaveia , e a saída *Q* vai de ALTA para BAIXA. Quando a saída *Q* de FF1 vai de ALTA para BAIXA, ela por sua vez chaveia FF2 (a saída *Q* de FF1 está ligada à entrada *CK* de FF2). FF2 chaveia de BAIXA para ALTA. Depois do pulso 2 de clock, a contagem binária aumentou para 0010.

Observemos o pulso 3 de clock na Fig. 3. O edge posterior ou transição final dispara FF1, que chaveia (muda de estado). A saída *Q* de FF1 chaveia de BAIXA para ALTA. A contagem binária (ver linha inferior) aumentou para 0011.
Observemos o pulso 4 de clock na Fig. 3. O edge posterior ou transição final dispara FF1, que chaveia (muda de estado), com *Q* indo de ALTA para BAIXA. Esta transição H-para-L em *Q* de FF1 por sua vez chaveia FF2. A saída *Q* de FF2 chaveia de ALTA para BAIXA. Esta transição H-para-L em Q de FF2 por sua vez faz com que FF3 chaveie. A saída Q de FF3 chaveia de BAIXA para ALTA. A contagem binária agora permanece em 0100.
Observemos a linha pontilhada após o pulso 4 na forma-de-onda ALTA em *Q* de FF3. Notar que um grande espaço de tempo passa antes que FF3 finalmente chaveie para seu estado ALTO. Isto é porque FF1 chaveia, que por sua vez chaveia FF2, que por sua vez chaveia FF3. Tudo isto consome tempo. Este tipo de contador é chamado *contador com ondulação*. O disparo de biestável a biestável efetivamente ondula através do contador. Este contador é também denominado *contador assíncrono*, porque nem todos os biestáveis chaveiam exatamente ao mesmo tempo com o pulso de relógio.
Observemos o restante da forma-de-onda na Fig. 3 para nos certificarmos de que compreendemos sua operação. Notar particularmente que no pulso 16 a transição H-para-L chaveia FF1. A saída de FF1 vai de ALTA para BAIXA. FF2 é chaveado por FF1. A saída de FF2 vai de ALTA para BAIXA. FF3 é chaveado (comutado) por FF2 e assim por diante. Notar que todos os biestáveis comutam (chaveiam ou mudam de estado) alternadamente e vão desde seu estado ALTO até seu estado BAIXO. A contagem binária é depois voltada para 0000. O contador não pára em sua contagem máxima, mas continua contando enquanto os pulsos de clock são introduzidos na entrada *CK* de FFl.
Contar cuidadosamente o número de pulsos ALTOS sob os 16 primeiros pulsos de clock (na linha de saída de FF1). Acharemos oito pulsos. Dezesseis pulsos entram em FF1 e somente oito pulsos saem. Este biestável é portanto um *divisor de freqüências*. 16 dividido por 8 é igual a 2. FF1 pode, portanto, ser também considerado um *contador de dividir-por-2*.
Contemos os pulsos de saída ALTA em FF2. Para 16 pulsos de clock, somente quatro pulsos aparecem sia saída de FF2. Então, dividindo 16 por 4 é igual a 4. A saída *Q* de FF2 pode ser considerada um *contador de dividir por-4*. Acha-se que a saída de FF3 é um contador de dividir-por-8. A saída de FF4 é um contador de dividir-por-16. Em alguns dispositivos, tais como clocks digitais, a divisão de freqüência é uma tarefa muito importante dos contadores.
A forma-de-onda confirma que um contador é um dispositivo lógico seqüencial. A característica de memória é também importante, pois o biestável deve "lembrar" quantos pulsos de clock chegaram à entrada *CK*. O contador assíncrono é o tipo mais simples de contador. Sua dificuldade é o *retardo de tempo*, pois um biestável dispara o seguinte e assim por diante.

3. CONTADORES SÍNCRONOS (PARALELOS)

O contador com ondulação assíncrono tem a limitação do atraso de tempo no disparo de todos, biestáveis. Para eliminai este problema, podem ser usados contadores paralelos. O diagrama de símbolos lógicos de um *contador paralelo* de 3 bits está ilustrado na Fig. 4a. Notar que todas as entradas *CK* estão ligadas diretamente à entrada de clock. Elas estão montadas em paralelo. Notar também que são usados biestáveis *JK* FF1 é o contador de casas ls e está sempre no modo chave (*toggle*). FF2 tem suas entradas *J* e *K* ligadas à saída de FF1 e estará no modo de retenção (*hold*) ou no modo chave (*toggle*): As saídas de FF1 e FF2 são introduzidas numa porta E. A porta E controla o modo de operação de FF3. Quando a porta E é ativada pelos ls em *A* e *B*, FF3 estará em seu modo chave (*toggle*). Com a porta E ' desativada, FF3 estará em seu modo de retenção (*hold*). FF2 é o contador de casas 2s e FF3 é o contador de casas 4s.

A seqüência de contagem deste contador paralelo de 3 bits é mostrada na Fig. 4b. Notar que este é um contador de módulo 8 (mod-8). O contador iniciará a contagem em binário 000 e contará até 111. Depois ele reciclará de volta a 000 para iniciar a contagem novamente.

A forma-de-onda (diagrama de temporização) do contador paralelo mod-8 está desenhada na Fig. 5. A linha superior representa as entradas (*CK*) de relógio para todos os três biestáveis. As saídas (em *Q*) de cada um dos biestáveis são mostradas nas três linhas do meio. A linha inferior dá a contagem binária indicada.

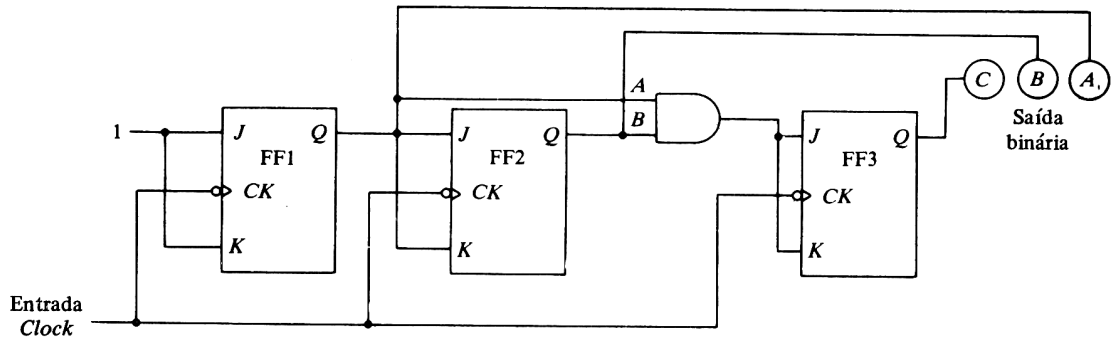
Consideremos o pulso 1 na Fig. 5. O pulso 1 chega em cada um dos três biestáveis. FF1 comuta . (chaveia) de BAIXO para ALTO. FF2 e FF3 não chaveiam porque eles estão no modo de retenção (J e $K = 0$). A contagem binária é agora 001.

Observemos o pulso 2 na Fig. 5. O pulso 2 chega em todos os biestáveis. FF1 e FF2 comutam porque eles estão no modo chave (J e $K = 1$). FF1 vai de ALTO para BAIXO, enquanto FF2 vai de BAIXO para ALTO. FF3 está ainda no modo de retenção, e portanto não comuta (chaveia). A contagem agora é 010.

O pulso 3 chega em todos os biestáveis ao mesmo tempo. Somente FF1 comuta. FF2 e FF3 estão no modo de retenção devido a ter J e $K = 0$. A contagem binária é agora 011.

Consideremos o pulso 4 na Fig. 5. Notar que a porta E é ativada pouco antes de o pulso de relógio ir de ALTO para BAIXO. A porta E colocará FF3 no modo chave (J e $K = 1$). Na transição H-para-L do ' pulso 4 de relógio, todos os biestáveis comutam. FF1 e FF2 vão de ALTO para BAIXO. FF3 comuta de ' BAIXO para ALTO. A contagem binária é agora 100. Notar a linha pontilhada abaixo do bordo posterior do pulso 4 de clock. Dificilmente qualquer atraso de tempo é evidente de FF1 a FF3, porque todos os biestáveis são sincronizados (clocados) exatamente ao mesmo tempo. Esta é a vantagem do contador ' ' do tipo paralelo. Os contadores paralelos são também chamados *contadores síncronos*, porque todos os . biestáveis chaveiam exatamente em tempo com o relógio. Os contadores paralelos são mais complicados ; (ver as linhas acrescentadas e a porta E) mas são usados onde o problema do atraso do tempo com um contador com ondulação (assíncrono) causaria dificuldades.

Observemos o restante da forma-de-onda na Fig. 5. Entendamos que cada biestável é sincronizado em cada pulso de relógio. FF1 sempre comuta. FF2 e FF3 podem estar ou no modo chave ou no modo de retenção.



(*a*) Diagrama de símbolos lógicos

Contagem decimal	Contagem binária 4s 2s 1s — *C B A*
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

(*b*) Seqüência de contagem

Fig. 4 - Um condutor paralelo de 3 bits.

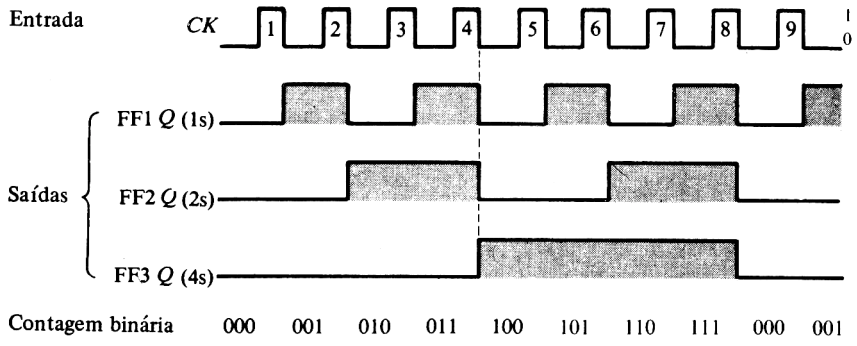


Fig. 5 - Diagrama de temporização de um condutor paralelo de 3 bits.

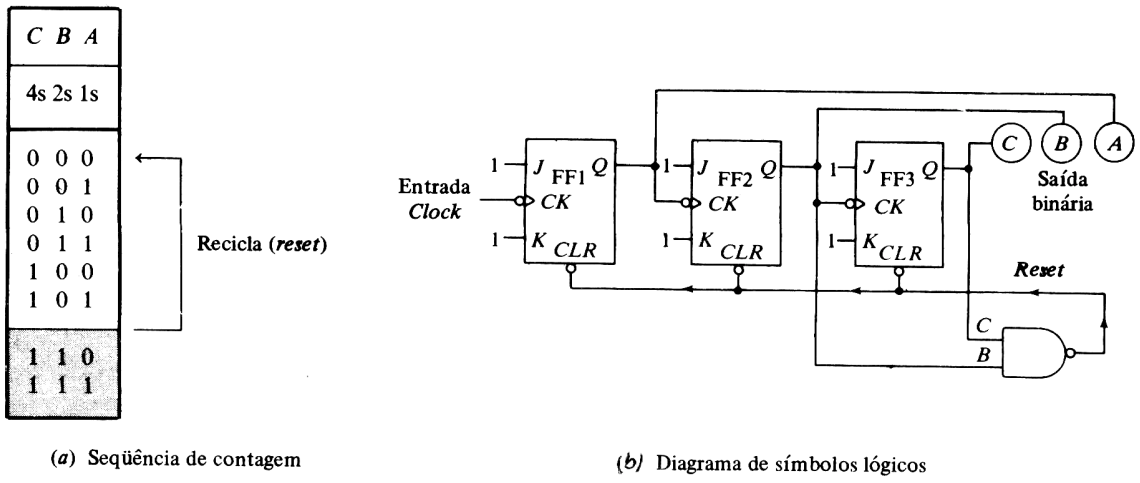
4. OUTROS CONTADORES

Suponhamos que estamos precisando de um contador assíncrono de módulo-6. O que pareceria ele? A primeira etapa na construção de um contador assíncrono mod-6 consiste em listar a seqüência da contagem mostrada na Fig. 6a. A seqüência da contagem do contador de mod-6 é de 000 a 101. Notar que um contador de 3 bits é necessário com um contador (*C*) de 4s, um contador (*B*) de 2s, e um contador (*A*) de ls. Conforme mostrado na Fig. 6a, o contador de 3 bits normalmente conta de 000 a 111. As duas últimas contagens na carta (110 e 111) devem ser omitidas.

O artifício para este problema de projeto de mod-6 consiste em observar a contagem binária *imediatamente após a mais alta contagem do contador*. Esta é 110 neste caso. Introduzir o 110 num circuito lógico que produzirá um pulso de limpar (*clear*), ou reajustar (*reset*). O pulso de limpar volta para uma entrada de limpar assíncrona em cada biestável *JK*, limpando, ou reajustando, portanto, o contador em 000.

O circuito lógico necessário para limpar ou restabelecer (reajustar) os biestáveis *JK* de volta a 0 é mostrado na Fig. 6b. A porta NE de 2 entradas fará o serviço quando as saídas de FF2 e FF3 forem introduzidas nela. Notar da tabela de contagem na Fig. 6a que a primeira vez que tanto *C* quanto *B* são 1 é imediatamente após a contagem mais alta. Assim, quando o contador tenta ir para 110, ele imediatamente será limpo ou reajustado em 000.

O contador de mod-6 na Fig. 6b é um contador assíncrono que é reajustado ou limpo exatamente duas contagens antes de sua contagem máxima normal de 111. A porta NE faz o serviço de reajustar ou restabelecer os biestáveis em 0 ativando as entradas *CLR*.



(*a*) Seqüência de contagem

(*b*) Diagrama de símbolos lógicos

Fig. 6 - Contador de ondulações de mod-6.

As formas-de-onda do contador assíncrono em anel mod-6 estão em diagrama na Fig. 7. A entrada de relógio para FF1 é mostrada no topo. As três linhas do meio mostram o estado das saídas *Q*. A linha inferior dá a contagem binária.

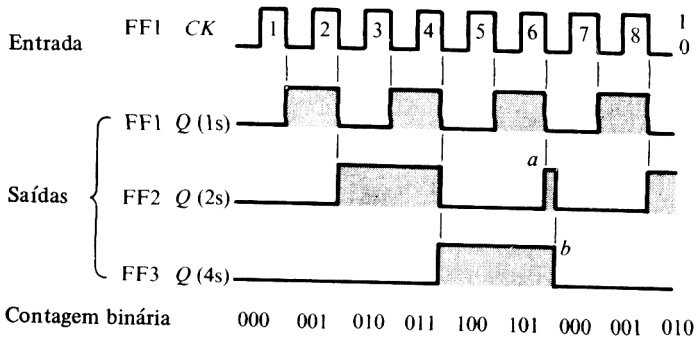


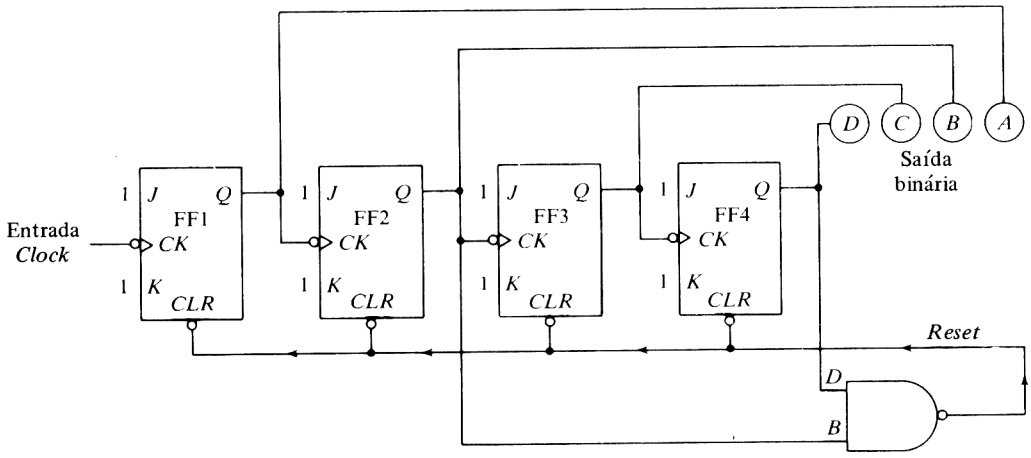
Fig. 7 - Diagrama de temporização de um contador de ondulações mod-6.

O contador mod-6 representado no diagrama na Fig. 7 opera como um contador assíncrono normal até o pulso 6. A contagem binária antes do pulso 6 é 101. Esta é a contagem máxima desta unidade. Na transição H-para-L do pulso 6 de clock, FF1 comuta de ALTO para BAIXO. A transição H-para-L dos FF1 dispara FF2, que comuta de BAIXO para ALTO. No ponto a na Fig. 8.9, ambas as saídas de FF2 e FF3 estão em 1. Estes dois ls são aplicados à porta NE (ver Fig. 6b). A porta NE é ativada, produzindo um 0. Este 0 ativa a entrada *CLR* assíncrona para todos os biestáveis, reajustando-os todos em 0. O reajuste, ou limpeza para 000, é mostrado no ponto b na Fig. 7. O pequeno pulso no ponto a na Fig. 7 é tão curto que nem mesmo chega a iluminar os indicadores de saída. O contador está livre para contar para cima normalmente outra vez a partir do binário 000.

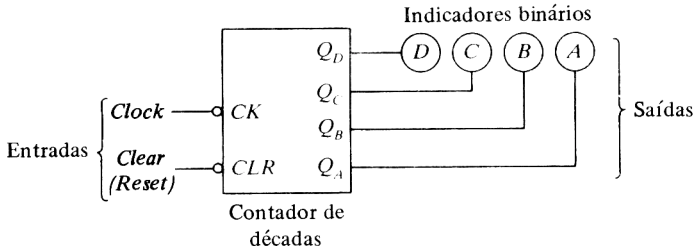
Observemos o edge posterior do pulso 6 novamente na Fig. 7. Notemos novamente o atraso entre o tempo em que o pulso 6 vai de ALTO para BAIXO e o tempo em que FF2 e FF3 finalmente são reajustados em 0 no ponto b. Os engenheiros se referem a este atraso de tempo como o tempo de propagação, e ele é baseado no retardo de propagação do biestável e da porta que estão sendo usados. O retardo de propagação de um biestável TTL típico é muito curto - cerca de 30 ns (nanossegundos). Algumas famílias lógicas têm retardos de propagação muito mais longos.

Um *contador de décadas* é provavelmente o contador mais amplamente usado. Um contador de décadas poderia também ser descrito como um *contador de módulo 10*. A Figura 8a mostra em diagrama um contador assíncrono de mod-10. São usados quatro biestáveis *JK* mais uma porta NE para montar o contador de décadas. A unidade conta exatamente como o contador de mod-16 até 1001. O binário 1001 é a contagem máxima desta unidade. Quando a contagem tenta avançar para 1010, os dois ls ($D = 1$ e $B = 1$) são introduzidos na porta NE. A porta NE é ativada, reajustando o indicador visual em 0000.

Um símbolo lógico geral às vezes é usado num contador quando comprado em forma de CI. O símbolo lógico na Fig. 8b pode ser substituído no diagrama de contador de décadas na Fig. 8a. Uma entrada limpar (*clear*) (ou reajustar - *reset*) foi acrescentada ao contador de décadas na Fig. 8b. Esta entrada de limpar não aparece no contador de décadas na Fig. 8a. Um 0 lógico ativa o reajuste e limpa a saída em 0000.



(*a*) Diagrama de símbolos lógicos para um contador de décadas assíncrono



(*b*) Símbolo lógico simplificado para um contador de décadas

Figura 8

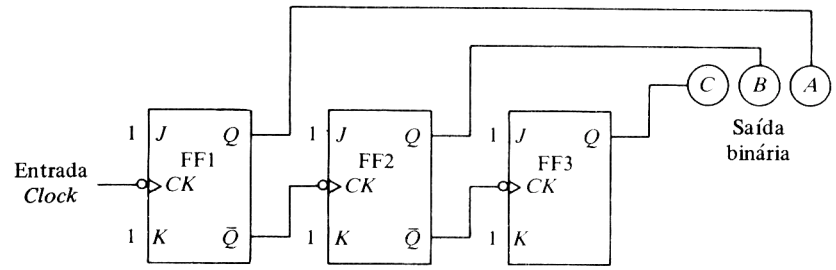
Foi mencionado que alguns contadores contém para baixo. A Fig. 9 mostra em diagrama um contador decrescente. Esta unidade é um contador para baixo assíncrono de 3 bits. A contagem binária seria 111, 110, 101, 100, 011, 010, 001, 000, seguida de uma reciclagem para 111, e assim por diante. Notar na Fig. 9a que o contador para baixo assíncrono é muito semelhante ao contador para cima. A "linha de disparo" de FF1 a FF2 vai desde a saída *Q* até a entrada de clock em vez de desde a saída *Q* para o clock. De outro modo, a montagem do contador para cima e do contador para baixo é a mesma. Notar também que cada biestável *JK* está em seu modo chave (*J* e *K* iguais a 1).

A forma-de-onda na Fig. 9b ajuda a entender a operação do contador para baixo. A linha de cima é a entrada *CK* para FF1. A linha de baixo é a contagem binária. Notar que a contagem binária começa em 111 à esquerda. Duas saídas (Q e Q) são mostradas tanto para FF1 quanto para FF2. A saída *Q* é mostrada em FF3. As saídas ligadas aos indicadores binários são mostradas sombreadas no diagrama de temporização.

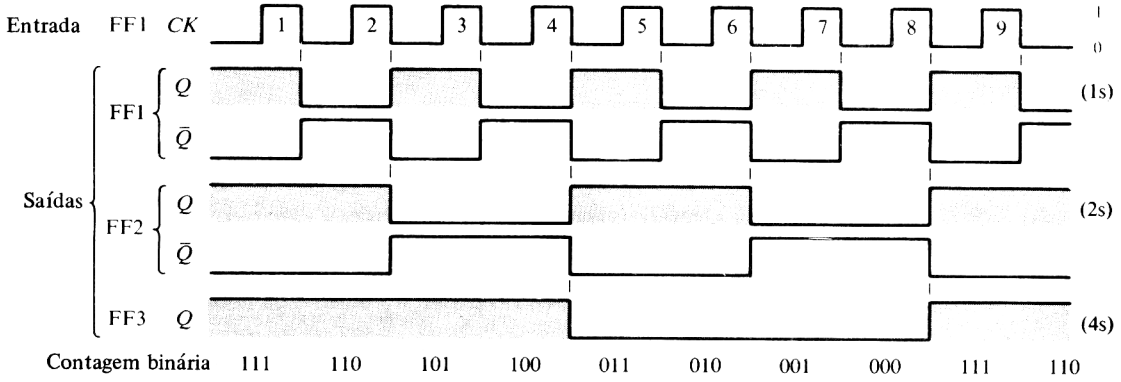
Consideremos o pulso 1 na fig. 9b. Todos os biestáveis estão ajustados (*set*). A saída binária nos indicadores é 111. Na transição H-para-L do pulso 1, FF1 comuta. A saída *Q* vai de ALTA para BAIXA (*Q* vai de BAIXA para ALTA). A contagem binária é agora 101.

Consideremos o pulso 3 na Fig. 9b. Na transição H-para-L do pulso de clock, FF1 comuta. Isto faz com que a saída *Q* vá de BAIXA para ALTA. A saída *Q* vai de ALTA para BAIXA, fazendo desse modo com que FF2 comuta, e a saída *Q* vai de ALTA para BAIXA (*Q* vai de BAIXA para ALTA). A contagem binária é agora 101.

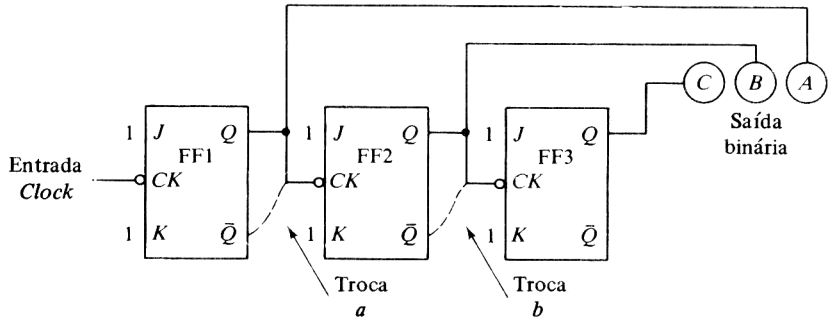
Consideremos o pulso 3 na Fig. 9b. O pulso 3 dispara FF1. A saída *Q* de FF1 torna-se BAIXA, enquanto *Q* torna-se ALTA. A saída binária é agora 100.



(*a*) Diagrama de símbolos lógicos



(*b*) Diagrama de temporização



(*c*) Duas trocas necessárias para converter um contador para baixo em um contador para cima de 3 bits.

Fig. 9 - Um contador para baixo assíncromo de 3 bits.

Observemos o pulso 4 na Fig. 9b. O pulso 4 dispara FF1. FF1 está ajustado, e a saída $\overline{Q}$ vai de ALTA para BAIXA. Isto faz com que FF2 comute. FF2 está ajustado, e a saída *Q* vai de

ALTA para BAIXA. Isto por sua vez faz com que FF3 comute e se reajuste. A saída binária após o pulso 4 é então 011.

Examinemos o restante da forma-de-onda. Particularmente, notar as linhas verticais iluminadas que mostram o disparo do próximo biestável. Lembrar que as saídas *Q* se conectam aos indicadores de saída mas as saídas *Q* de FF1 e FF2 disparam o próximo biestável.

TIPOS DE MEMÓRIAS SEMICONDUTORAS

As memórias são locais onde armazenam-se dados e programas em um sistema de computação. As memórias são as partes mais ativas de um computador, armazenando programas e dados antes, durante e após a execução. Pode-se afirmar que a memória é equivalente a milhares de registradores; cada um armazenando uma palavra binária.

Há duas únicas ações que podem ser realizadas em uma memória. A primeira é a ação de guardar um elemento (ou um grupo de elementos) - em computação, esta ação genericamente denominada de *armazenar* e a operação em si, que é realizada para a consecução dessa ação de armazenamento, é chamada de *escrita* ou *gravação* ("write"). A segunda é a ação de recuperação do elemento guardado (ou grupo de elementos) para um uso qualquer - em computação esta ação se denomina *recuperar* (“retrieve") e a operação para realizá-la chama-se *leitura* ("read").

Até o final da década de 60, as memórias dos computadores eram magnéticas. As mais antigas eram de "*tambor*": um cilindro magnético, girando a alta velocidade, com cabeças de gravação e leitura escrevendo e lendo dados e instruções em sua superfície. Outras eram construídas com núcleo de ferrite: minúsculos toróides de ferrite, costurados por fios de acesso de dados e de endereçamento. Em 1969, a IBM introduziu em seu processador modelo 360/85, uma pequena (pelos padrões atuais) memória de 16 Kbytes, construída com transistores. Surgiam ali as memórias monolíticas, ou *memórias a semicondutor*. Desde então diferentes tipos de memórias tornaram-se disponíveis no mercado. Como conseqüência, o projetista tem muito onde escolher, mas a escolha é mais difícil, e deve se basear na adequação das características da memória às necessidade da aplicação.

Conceitos Preliminares

Quando se fala em Sistemas Digitais de um ponto de vista genérico, as memórias são vistas em dois grandes grupos: as **lê-escreve** e as de **apenas-leitura**. As memórias do tipo lê-escreve são as tradicionais RAM´s e como seu nome diz, têm seu conteúdo alterado durante o funcionamento do circuito. Em contrapartida, as memórias de apenas-leitura têm conteúdo fixo e os circuitos conseguem apenas ler o que ali está escrito. Como metáfora pode-se fazer a analogia de uma memória do tipo lê-escreve com um caderno e das memórias do tipo apenas-leitura com um livro.

No caso de uma memória de computador, o elemento a ser manipulado fisicamente é o *bit*, o qual, em grupo de **n** bits (n pode variar consideravelmente, dependendo daquilo a que se está referindo), corresponde a unidade de informação a ser armazenada, transferida, recuperada, etc. Ou seja, a memória serve para guardar (armazenar) informações (na forma de bits) e recuperá-las quando desejado. Para isso, realizam-se de *escrita* (transferência de bits de outro componente do sistema de computação; por exemplo: da UCP, de disco) e de *leitura* (transferência de bits da memória para a UCP, disco, etc.). Por *informação,* entendem-se as instruções e os dados de um programa.

Para que a informação possa ser armazenada em uma memória (operação de escrita) é necessário que seja definido um local disponível identificado de alguma forma precisa e única (um número, por exemplo). O número ou código que é associado ao local é o *endereço* ("address") e irá permitir que a informação possa ser localizada, assim como nossa residência é localizada pelo seu endereço, como o livro é localizado pelo "endereço" da prateleira / estante e a correspondência é manipulada pelo seu endereço.

Antes de analisar os aspectos práticos do uso de memórias como elementos de projeto, é preciso definir e tornar mais precisos alguns termos como *endereço, posição, modo de acesso,* entre outros, e apresentar algumas classes distintas de memória.

Posição e Endereço

Costuma-se chamar de memória os circuitos utilizados para armazenar uma alta quantidade de bits, organizados em *palavras* e *posições* de memória.

A **palavra** de memória é o conjunto de bits que pode ser, simultaneamente, lido ou escrito (gravado) na memória. Tipicamente, as memórias têm palavra de 8, 16 ou 32 bits (1, 2 ou 4 *bytes);* existem pastilhas de memórias de 1 bit e de 4 bits (meio byte ou um *nibble),* que podem ser associadas para construir memórias maiores.

A **posição** de memória é o local onde se armazena uma palavra. A cada posição está associado um **endereço**, que é um código binário utilizado para acessar a posição. A quantidade de endereços define, portanto, a quantidade de posições.

A **capacidade** da memória é a medida do total de bits que ela armazena. Calcula-se a capacidade da memória multiplicando-se a quantidade de posições pela largura da palavra. A capacidade da memória é normalmente expressa em tantas posições de tantos bits.

Exemplo: Uma memória de 1024 posições, com palavras de 8 bits, tem uma capacidade de 8 x 1024 8192 bits.

Ao expressar a capacidade de memória em bits mascara-se a sua organização interna (ou seja, o tamanho da palavra), e uma vez que essa organização é fundamental para a escolha e a associação das memórias num projeto, é comum referir-se à capacidade das memórias em termos do número de posições x tamanho da palavra.

Exemplo: Memória de 1024 x 8 bits, ou 1024 bytes, ou ainda 1 Kbyte (1K = 1024 posições)

Os endereços, codificados em binário, são apresentados ao módulo de memória através da **via de endereço** *(address bus).* Portanto, a cada posição de memória corresponde um endereço, em binário. Uma memória de 1024 posições terá 10 bits de endereço ($1024 = 2^{10}$); por outro lado, uma via de endereços de 16 bits poderá endereçar uma memória com $2^{16} = 65536$ posições (64K).

As Memórias do tipo Lê-Escreve

Inúmeras são as características de acordo com as quais pode-se classificar as memórias em categorias, tais como:

- Modos de Acesso;
- Volatilidade;
- Tipo de Armazenamento.

Obs. Ressalte-se, entretanto, que o “universo" das memórias é vasto e muito rico em técnicas e idéias.

Modos de Acesso

Existem memórias de **acesso seqüencial**. Nelas, as posições de memória estão fisicamente organizadas em lugares sucessivos, e a leitura ou escrita de uma certa posição requer que se “caminhe" sobre todas as posições anteriores. Como ocorre com as fitas magnéticas: só se consegue atingir trechos no final da memória passando sobre os trechos iniciais. Dessa forma, o tempo necessário para se ter acesso a uma dada posição de memória

(chamado de *tempo de acesso*), será tanto maior quanto mais distante a posição desejada estiver da posição inicial. Os discos magnéticos e as fitas magnéticas são dois típicos exemplos de memórias com acesso seqüencial.

Outro tipo de memória é o de **acesso direto ou aleatório** (**RAM** - *Random Access Memory)*. Nessas é possível realizar-se a seleção direta de qualquer posição sem ter que passar sobre posições anteriores. O que caracteriza a memória de acesso direto é que o tempo de acesso é o mesmo para todas as posições, independentemente da posição inicial.

O termo *Random Access Memory* - RAM que deveria ser aplicado para diferenciar memórias de acesso direto ou "aleatório", das seqüenciais, tem sido usado inadequadamente para descrever as memórias do tipo lê-escreve. As memórias apenas de leitura, ROM *(Read-Only Memory),* também são de acesso direto, e não são comercialmente referidas por RAM.

Volatilidade

Com relação à capacidade de reter os dados armazenados, os dispositivos de memórias podem ser divididos em duas categorias: **voláteis e não-voláteis**. As memórias voláteis mantém o seu conteúdo armazenado apenas enquanto estiverem “alimentadas" com energia elétrica. As memórias não-voláteis mantém seu conteúdo mesmo que falte energia. Tipicamente as memórias magnéticas são não-voláteis. As memórias do tipo lê-escreve de acesso direto são em geral voláteis, enquanto que as memórias de apenas-leitura, as ROM *(Read-Only Memory)* e seus derivados PROM, EPROM, EPROM (E^2PROM) e EAROM são memórias não-voláteis.

Anteriormente ao advento das memórias a semicondutor, as memórias de núcleo de ferrite, *core memory,* cumpriam essa função de memórias tipo lê-escreve com a vantagem de serem não-voláteis, mas com os enormes inconvenientes de seu grande tamanho e consumo.

Tipo de Armazenamento

De acordo com a forma como a informação é armazenada, existem dois tipos de memórias lê-escreve: **as estáticas e as dinâmicas**.

As *memórias estáticas* são velozes e simples de serem utilizadas: a célula básica (a unidade que armazena um bit de informação) é constituída por um flip-flop tradicional que armazena "**0**" ou "**1**" (conteúdo daquela posição). Seu inconveniente é que a célula tem dimensões grandes, o que limita a quantidade de posições que se consegue integrar em uma pastilha.

Na figura 2 vemos um sistema de memória RAM estática de 1024 por 4. Isto significa que ela está formada por 64 linhas e 18 colunas (64 x 16 = 1024) e o tamanho da palavra (byte) é de 4 bits. Como esta memória é formada por uma palavra de 4 bits, teremos 4 planos com arranjo de 1024 endereços para cada um, formando a memória 1024 x 4. Veja a figura 3.

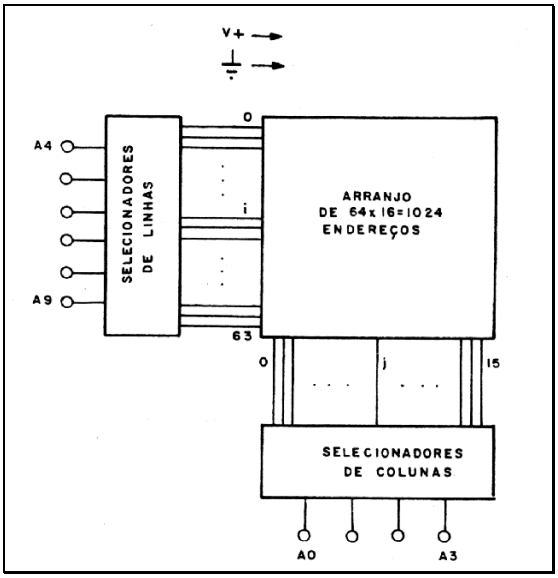


Figura 1

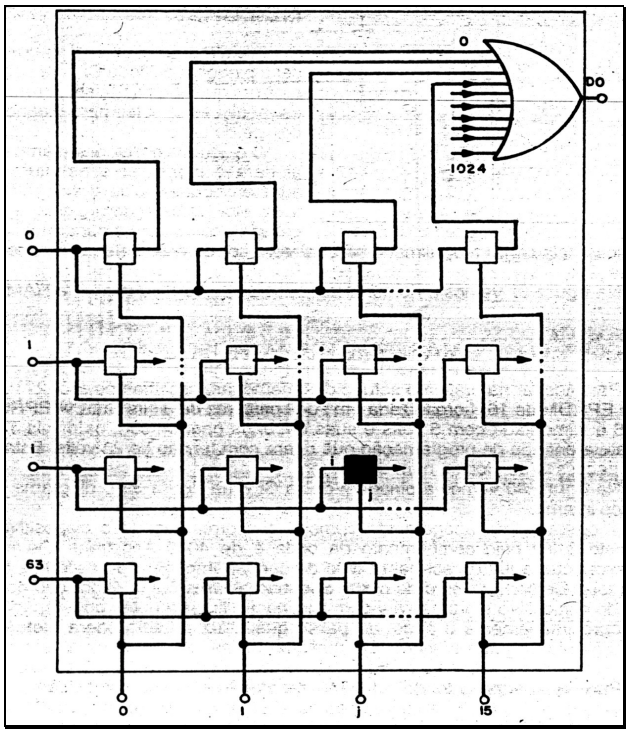


Figura 2

As *memórias dinâmicas* apresentam alta capacidade, velocidade moderada e baixo consumo. Sua célula é um capacitor para o armazenamento de carga, associado a um transistor. A presença de carga no capacitor é interpretada pelos circuito sensores da memória como nível UM; a ausência de carga, como nível ZERO lógico. A figura 4 mostra um exemplo dessa célula.

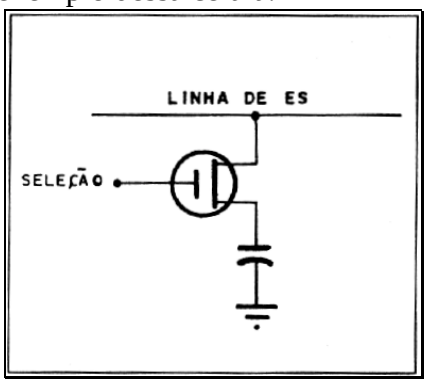


Figura 3

A RAM dinâmica utilizada como célula o elemento MOS, armazenamento da informação é feito nas capacitâncias parasitas que se formam entre a porta e a fonte do transistor FET (figura 4). Uma célula de memória dinâmica consiste em um transistor MOSFET e um único capacitor MOS de alguns picofarads.

O transistor atua como chave e o capacitor armazena a tensão correspondente ao nível zero ou um. A operação de leitura é feita ativando-se o gatilho ou porta do transistor.

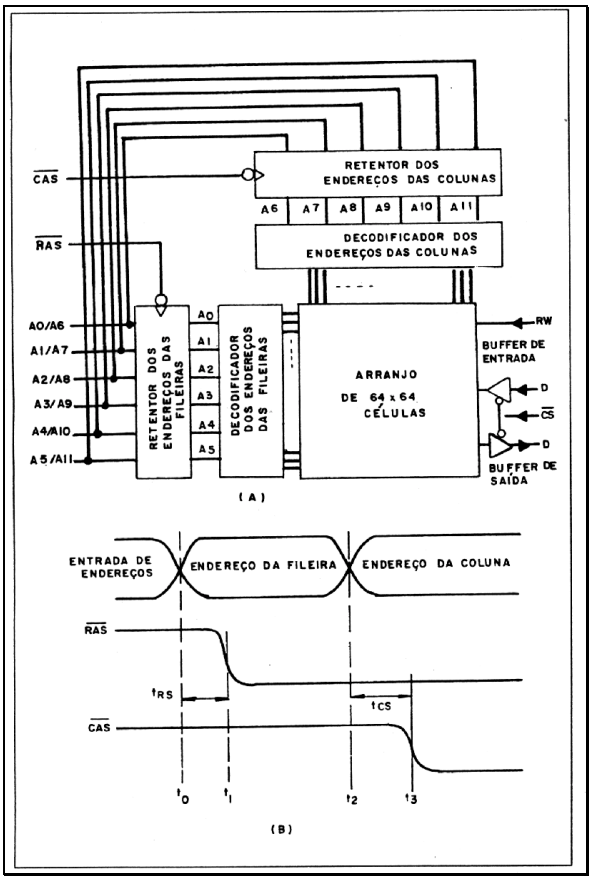


Figura 4

Como a tendência da carga é de se dissipar, as memórias dinâmicas necessitam de um reforço periódico na carga, ou refrescamento. Esse requisito implica na necessidade de circuitos adicionais para realizar tal reforço de cargas periodicamente, o que torna mais complexa sua utilização. Além disso, na ocasião do refrescamento a memória não permite a realização de escritas ou leituras, reduzindo sua disponibilidade. Existem circuitos especialmente fabricados para controlar as memórias dinâmicas, os DRAM Controllers *(Dynamic RAM ControlIers),* integrados em pastilhas LSI.

Devido a sua simples estrutura, a RAM dinâmica possibilita concentrar em sua área 4 vezes mais informações do que a RAM estática. A figura 5 ilustra a arquitetura de uma DRAM (dinâmica RAM) onde temos um arranjo de 64 x 64 células.

Para diminuir o número de pinos de endereçamento, os fabricantes de chips de memória utilizam o sistema de multiplexação de endereços. Nesse caso, o endereço da fileira é aplicado em primeiro lugar, para depois aparecer ó endereço da coluna.

Uma alternativa intermediária, que pode ser utilizada em pequenos sistemas são as memórias pseudo - estáticas, as iRAM *(Integrated R*AM). Elas são dispositivos que contêm, num só encapsulamento, a unidade de memória dinâmica e a unidade de controle. Do ponto de vista do usuário, a memória iRAM se comporta como se fosse estática. Suas características de consumo, capacidade e velocidade, todavia, são as de uma memória dinâmica.

As Memórias de Apenas-Leitura (ROM)

As memórias de apenas leitura (ROM - *Read-OnIy Memory)* tratadas aqui, são do tipo RAM *(Random-Access Memory)*: acessam diretamente cada posição de armazenamento. Como o nome diz, as memórias de apenas-leitura não permitem a gravação dos dados. Em uso normal, pode-se apenas ler seu conteúdo. O conteúdo de uma ROM é gravado no momento da fabricação da memória, ou mais tarde, em aparelho especial. Dos vários tipos de ROM, a maioria deles tem a gravação de seu conteúdo feita fora do circuito.

Por isso, as ROM são usadas basicamente para o armazenamento de informações que não estão sujeitas a mudanças ao longo do processamento. Memórias do tipo ROM surgiram a partir da necessidade de armazenar informação (programação, tabelas, constantes, etc.) em equipamentos microprogramados. Antes do surgimento e consolidação das memórias a semicondutor, diversas outras tecnologias foram utilizadas como memórias capacitivas e memórias indutivas. E a partir do advento dos microprocessadores que as memórias ROM a semicondutor integrados, passaram a ser muito utilizadas. Nos microprocessadores, o programa básico (BIOS - *Basic Input/Output System)* é fixo e armazenado numa ROM, enquanto que os programas aplicativos e os dados manipulados são armazenados em memórias do tipo lê-escreve.

Os aspectos tecnológicos ligados à construção das memórias de apenas-leitura são diversos, e dependem do particular tipo de ROM. Esses detalhes estão apresentados a seguir.

Mask-ROM - ou ROM Programada por Máscara

As Mask-ROM são o tipo mais antigo de memória ROM. Ela é composta de uma matriz de células de diodos ou transistores. A programação da memória com 1 ou 0 é feita pela conexão ou não de dispositivos à grade da memória, o que é feito durante a etapa de fabricação da memória, na fase de metalização das conexões.

Veja a figura 6. Ela mostra uma matriz de uma memória com 4 posições de 4 bits. Os endereços E_1E_0 selecionam uma das quatro posições, ativando um das linhas S_0 a S_3 com um nível 1. A linha ativada, por exemplo S_0 provoca a condução dos diodos que a une às saídas. O conteúdo armazenado na posição selecionada, ou seja 1101, surge então na salda. Para evitar problemas de *fan-out* nas linhas de seleção, e diminuir as correntes envolvidas, as ROM utilizam transistores ao invés de diodos em cada um dos nós. Dessa forma, quem deve fornecer as maiores correntes, não é mais o decodificador, mas a própria fonte de alimentação, cabendo às linhas de seleção o fornecimento da reduzida corrente de base dos transistores que estiverem a ela ligados (figura 7). No caso de dispositivos MOS essas correntes são ainda menores.

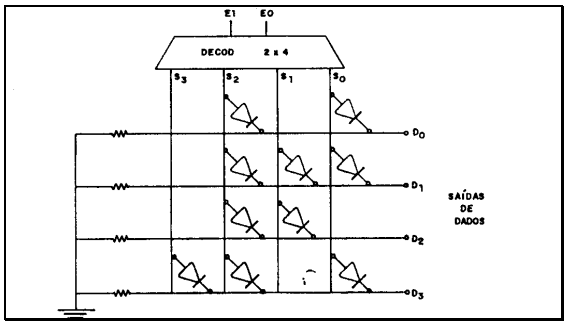


Figura 6

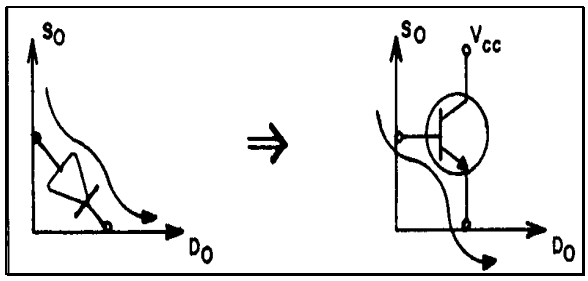


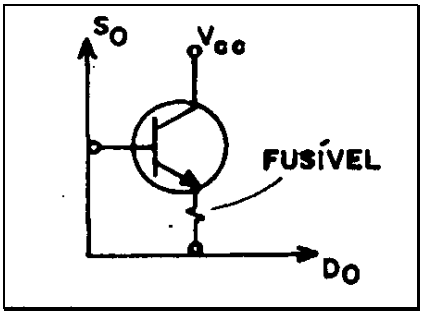
Figura 7

Um detalhe importante aqui, é que o conteúdo da memória deve ser fornecido pelo consumidor ao fabricante de memórias para que, na fabricação de memória, a máscara de metalização seja tal que faça as conexões corretas. É claro que essa programação (metalização) uma vez feita, não pode ser desfeita. Por isso, as ROM só devem ser encomendadas ao fabricante após ter seu conteúdo muito testado, para ser pequena a probabilidade de conter erros, e apenas em casos de grande volume, porque o custo dessa produção sob medida só se justifica quando se prevê a utilização em altas quantidades.

PROM - ROM Programável pelo usuário

Para superar o problema de custo de programação, foi desenvolvida a PROM - *Programmable ROM*, que é semelhante à ROM programável por máscara, exceto pelo fato de ser o usuário quem faz a programação do seu conteúdo, utilizando programadores de PROM. Internamente as PROM são também constituídas por uma matriz de diodos ou transistores como as ROM; com a diferença que esses dispositivos encontram-se previamente ligados à grade da memória, o que significa ter valor lógico 1 armazenado ou todos os bits. O procedimento de programação consiste em gravar 0 nas posições desejadas o que é conseguido “queimando-se" minúsculos fusíveis existentes na grade, para cada bit, de forma a separar os transistores da grade. Veja a figura 8.

EPROM - ROM Programável e Apagável



O fusível é queimado selecionado-se na PROM o endereço e a linha de dados desejados, e aplicando-se um pulso de alta tensão, tipicamente de 10 Volts a 30 Volts, através de um pino especial da pastilha. Esse procedimento também é feito uma só vez, o que significa que se a programação foi errada, a pastilha estará perdida. Entretanto o custo da pastilha é bem menor que o de uma ROM já que não é fabricada sob medida.

Figura 8

O passo seguinte da indústria foi o das EPROM - *Erasable Programmable ROM*, que são programáveis pelo usuário, mas que ainda podem ter seu conteúdo apagado. O processo de apagamento as faz retornar ao estado inicial, com conteúdo 1 em todos os bits.

Ao invés da tecnologia de "fusíveis" utilizados nas PROM, as EPROM baseiam-se em programação por armazenamento de carga. Cada bit da memória possui um transistor MOS com dois *gates*, um deles flutuante, não conectado à grade da memória, e isolado por material de altíssima impedância. Em estado "apagado", como ao sair da fábrica, esses transistores não conduzem quando selecionados e o conteúdo das posições de memória é levado a 1 por resistores chamados de *pull-up.* Para gravar um valor 0 numa determinada posição, aplica-se uma alta tensão no gate flutuante, o que causa uma ruptura *(break-down)* no material isolante e permite o acúmulo de cargas no *gate* flutuante, as quais ali permanecem mesmo após o término do pulso de tensão, devido à alta impedância do material isolante. A presença dessas cargas no *gate* do transistor provoca a condução quando a posição daquele bit for selecionada, Com isso aquela linha de bit é levada para 0 (figura 9).

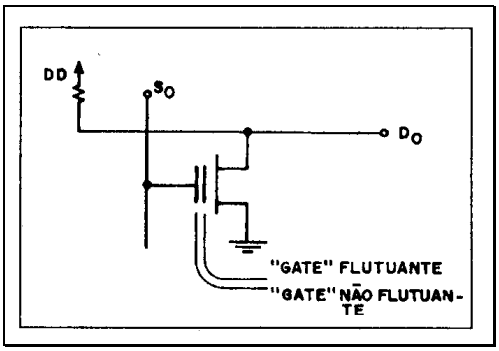


Figura 9

A alta impedância do material isolante, permite que uma EPROM mantenha sua programação por no mínimo 10 anos, se bem programada pelo aparelho programador de EPROM. Para reprogramá-la, é necessário, antes, apagar a programação anterior, o que é conseguido expondo-a à luz ultravioleta (dai elas serem chamadas de UV-EPROM). Os fótons de alta energia da luz UV colidem com os elétrons armazenados no *gate* flutuante e provocam o seu espalhamento, desfazendo o acúmulo de cargas e, portanto, a programação. Existe uma quantidade limite de vezes que a EPROM pode ser apagada e regravada.

Dados dos fabricantes mostram que a exposição constante à luz fluorescente de ambientes de trabalho pode apagar uma EPROM em cerca de 3 anos, enquanto que a exposição ao sol desfaria o conteúdo de uma pastilha EPROM em cerca de uma semana. Para evitar isso, costuma-se cobrir a janela de quartzo de pastilha com uma etiqueta opaca.

Exemplo de uma Memória EPROM (2716)

Para ilustrar na prática o estudo das memórias, escolhemos o CI 2716. É uma EPROM de 16K organizada em 2048 palavras de 8 bits cada. A EPROM 2716 é alimentada com 5 volts e suas entradas operam com os níveis TTL, exceto a entrada de programação que opera com tensão de 25 volts. Este Cl não necessita de clock e nem de refresh. Na figura 10 vemos a pinagem desse CI e na figura 11 o diagrama em blocos interno.

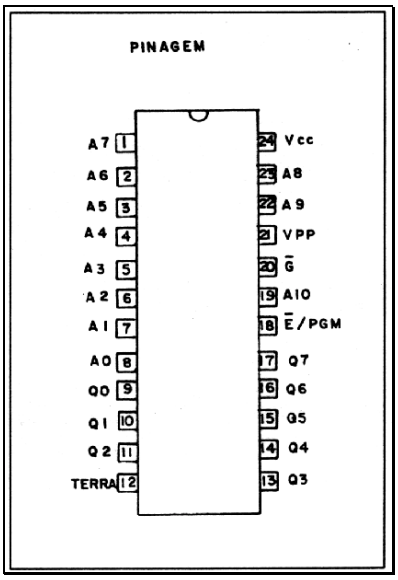


Figura 10

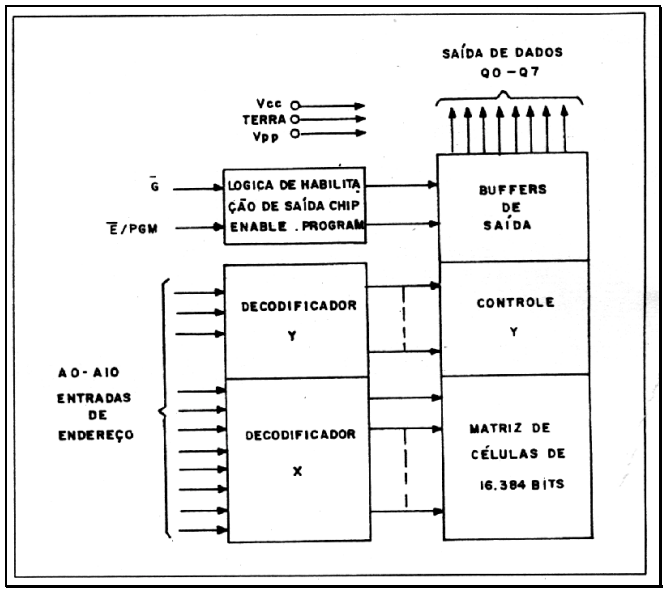


Figura 11

A operação de apagamento começa a ocorrer quando o dispositivo é exposto à luz cujo comprimento de onda é de 4000 Angstrons. Deve-se observar que a luz do sol bem como de certas lâmpadas fluorescentes tem esta faixa de comprimento de onda. Segundo estimativas, a exposição contínua do dispositivo em uma sala com iluminação fluorescentes pode apagá-lo em aproximadamente 3 anos, ao passo que a luz do sol poderá efetuar o apagamento em uma semana. De qualquer forma, é bom cobrir a janela do dispositivo com um selo escuro a fim de evitar o apagamento indesejável.

Para o apagamento do CI 2716 recomenda-se sua exposição à luz ultravioleta com comprimento de onda de 2537 Angstrons, devendo se distanciar uns 3 cm do tubo de luz e permanecer ali por cerca de 20 minutos.

E^2PROM - PROM Eletricamente Apagável

As E^2PROM são apagáveis e reprogramáveis como as EPROM, com a diferença de que isso pode ser feito no próprio circuito, eletricamente, sem necessidade de uso de equipamentos adicionais (apagadores e programadores). Além disso pode-se programar apenas um bit, ou um *byte*, sem ter que apagar a memória inteira.

A E^2PROM também tem sua célula constituída de transistores com *gate* flutuante, embora o material isolante seja de uma espessura muito menor. Desta forma cada bit pode ser apagado pela aplicação de uma tensão no *gate* não flutuante, oposta à que gerou as cargas na gravação.

A E^2PROM, apesar do nome ROM, permite leitura e escrita; todavia, ela não substitui uma autêntica" memória do tipo lê-escreve pois ela tem tempos de escrita muito superiores, tem custo muito maior e aceita um número limitado (10 mil) de ciclos de apagamento/gravação.

Com essas caraterísticas, e lembrando que a E^2PROM não é volátil, ela é muito útil para o armazenamento de dados que devem ser preservados quando o equipamento for desligado, ou se ficar sem alimentação. Tipicamente esses dados podem ser alterados, desde

que não freqüentemente, ao longo da operação. Esses seriam os casos dos dados de configuração de um equipamento, dos dados de tabelas, etc.

Flash Memory

As memórias *flash* são uma outra alternativa para aplicações de memórias não-voláteis que requerem reprogramação no circuito e maiores capacidades/densidades. Tais memórias são um misto de EPROM e E^2PROM: são eletricamente apagáveis como a última, mas o apagamento necessariamente é feito em toda a memória, como na primeira. Não é possível apagar byte a byte. O tempo de apagamento e regravação é bastante
pequeno, cerca de 5 segundos para 1 Megabit, mas pode exigir uma tensão adicional de 12 Volts, conforme o fabricante.

A vantagem das *flash memories* é o seu custo que tende a ser menor que o da E^2PROM em função de sua maior densidade; entretanto como as tecnologias estão por se consolidar no mercado, é conveniente conferir as várias opções oferecidas pelos fabricantes.

EXERCÍCIOS

1. Converter os seguintes números binários em seus equivalentes decimais:
a. 001100 c. 011100 e. 101010 g. 100001
b. 000011 d. 111100 f. 111111 h. 111000

2. Converter os seguintes números decimais em seus equivalentes binários:
a) 64 b) 100 c) 111 d) 145 e) 225 f) 500

3. Converter os seguintes números inteiros hexadecimais em seus equivalentes decimais:
a) C b) 9F c) D52 d) 67E e) ABCD

4. Converter os seguintes números decimais inteiros em seus equivalentes hexadecimais:
a) 8 b) 10 c) 14 d) 16 e) 80 f) 2560 g) 3000 h) 62 500

5. Como deve aparecer na saída da figura 1 o trem de pulsos indicado na entrada? Observar que dois trens de pulsos estão submetidos a uma operação E.

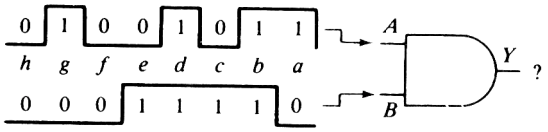


Figura 1 - Problema do trem de pulsos.

6. Como deve aparecer na saída o trem de pulsos na figura 2? Observar que dois trens de pulsos estão sendo submetidos a uma operação OU.

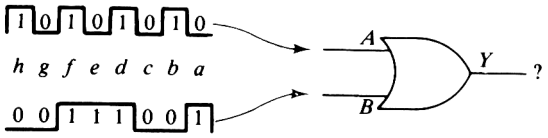


Figura 2 - Problema do trem de pulsos

7. Qual seria o aspecto do trem de pulsos na saída da porta XOU na figura 3?

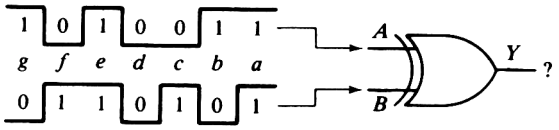


Figura 3 - Problema do trem de pulsos

8. Qual é a expressão booleana do diagrama lógico E-OU na figura 4?

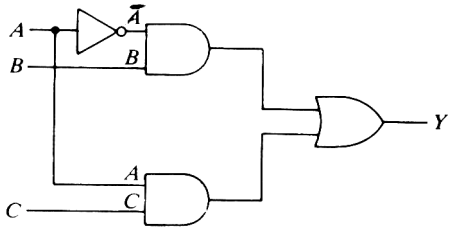


Figura 4 - Problema de circuito lógico E-OU

9. Qual é a tabela verdade para o diagrama lógico mostrado na figura 4?

10. Qual é a expressão booleana do diagrama lógico E-OU na figura 5?

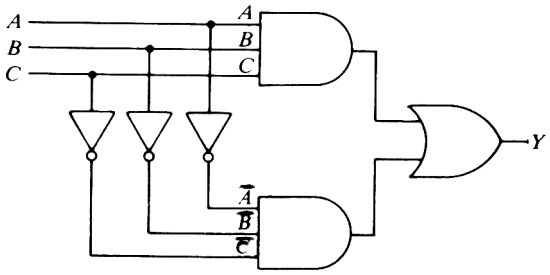


Figura 5 - Problema de circuito lógico E-OU

11. Qual é a expressão booleana do diagrama lógico E-OU na figura 6?

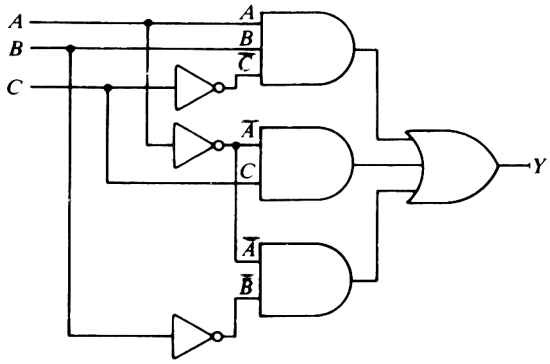


Figura 6 - Problema de circuito lógico E-OU

12. Qual é a tabela verdade para o diagrama lógico mostrado na figura 6?

13. Escrever uma expressão booleana de soma-de-produtos para a tabela verdade mostrada na figura 7.

Entradas			Saída	Entradas			Saída
C	*B*	*A*	*Y*	*C*	*B*	*A*	*Y*
0	0	0	1	1	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	1	1	1	1	0

Figura 7

14. Desenhar um diagrama de símbolos lógicos que executará a lógica mostrada na tabela verdade na figura 7.

15. Escrever uma expressão booleana de termos mínimos para a tabela verdade mostrada na figura 8.

Entradas			Saída	Entradas			Saída
C	*B*	*A*	*Y*	*C*	*B*	*A*	*Y*
0	0	0	1	1	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	1	1	1	1	0

Figura 8

16. Desenhar um diagrama de símbolos lógicos que executará a lógica especificada pela expressão booleana desenvolvida no problema anterior.

17. Escrever a expressão booleana simplificada baseada no mapa de Karnaugh dos itens abaixo:.

a) $Y = AB + A\overline{B} + \overline{A}B$
b) $Y = AB\overline{C} + A\overline{B}C + \overline{A}BC + \overline{A}B\overline{C} + \overline{A}\overline{B}C$
c) $Y = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}C + ABC$

18. Listar a saída *binária* em Q para o biestável na figura 9 durante os oito pulsos de clock.

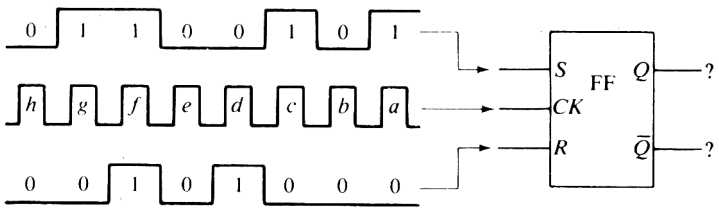
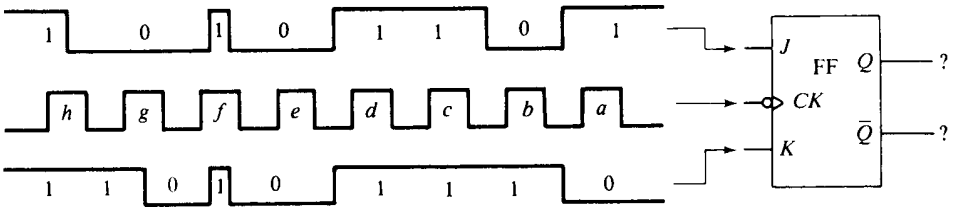


Figura 9 - Problema do trem de pulsos do biestável RS sincronizado

19. Listar a saída *binária* (em Q) para o biestável *JK* mestre-escravo na figura abaixo depois de cada um dos oito pulsos de clock.



20. Desenhar um diagrama de símbolos lógicos de um contador assíncrono de mol-8 usando três biestáveis *JK*.

21. Listar a seqüência de contagens binárias que o contador do problema anterior deve executar.

22. Ver a figura 10. Qual é a contagem binária após o pulso 2?

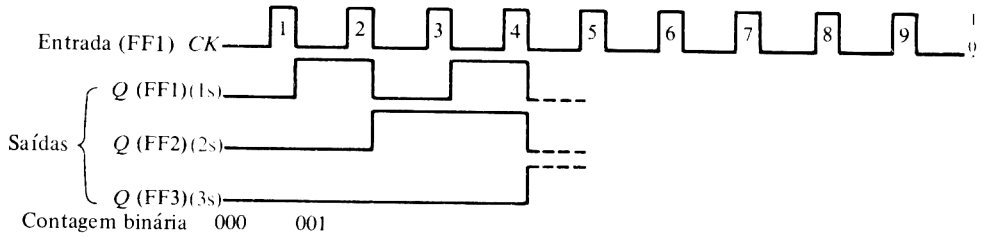


Figura 10 - Problema do trem de pulsos do biestável JK

RESPOSTAS DOS EXERCÍCIOS

1) a) 12 b) 3 c) 28 d) 60
e) 42 f) 63 g) 33 h) 56

2) a) 1000000 b) 1100100 c)1101111
d) 10010001 e) 11111111 f)111110100

3) a) 12 b) 159 c) 3410 d) 1662 e) 43981

4) a) 8 b) A c) E d) 10
e) 50 f) A00 g) BB8 h) F424

5) pulso: a = 0 b = 1 c = 0 d = 1
e = 0 f = 0 g = 0 h = 0

6) pulso: a = 1 b = 1 c = 0 d = 1
e = 1 f = 1 g = 0 h = 1

7) pulso: a = 0 b = 1 c = 1 d = 0
e = 0 f = 1 g = 1

8) $Y = \overline{A}B + AC$

9)

Entradas			Saída	Entradas			Saída
A	B	C	Y	A	B	C	Y
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	1
0	1	0	1	1	1	0	0
0	1	1	1	1	1	1	1

10) $Y = ABC + \overline{A}\overline{B}\overline{C}$

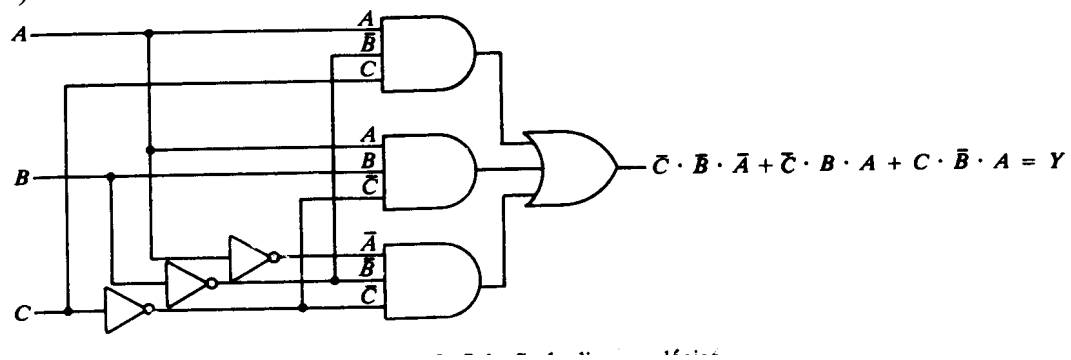
11) $Y = AB\overline{C} + \overline{A}C + \overline{A}\overline{B}$

12)

Entradas			Saída	Entradas			Saída
A	B	C	Y	A	B	C	Y
0	0	0	1	1	0	0	0
0	0	1	1	1	0	1	0
0	1	0	0	1	1	0	1
0	1	1	1	1	1	1	0

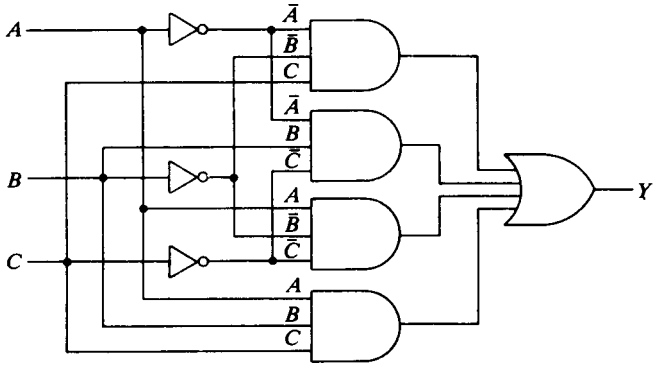
13) $Y = \overline{C}\overline{B}\overline{A} + \overline{C}BA + C\overline{B}A$

14)



15) $Y = \overline{A}\overline{B}C + \overline{A}B\overline{C} + \overline{A}B\overline{C} + ABC$

16)

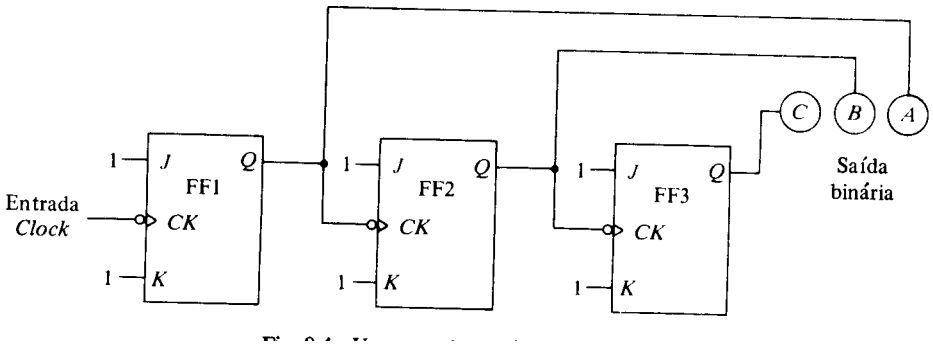


17) a) y = A + B
b) y = C + $\overline{A}B$
c) y = AC + $\overline{A}\overline{C}$

18) pulso: a = 1 b = 1 c = 1 d = 0 e = 0
f = 1(condição proibida) g = 1 h = 1

19) pulso: a = 1 b = 0 c = 1 d = 0
e = 0 f = 1 g = 0 h = 1

20)



21) O contador mod- 8 contaria em binário como se segue: 000, 001, 010, 011, 100, 101, 110, 111, e depois retorna a 000, e assim por diante.

22) A contagem binária após o pulso 2 é 010.