inet

INTRODUÇÃO AOS CIRCUITOS LÓGICOS

NOTAS DE AULA

*" **- Comece pelo começo** - disse o rei gravemente - , **siga até o fim; então pare**."*

Lewis Carroll

Prof.: Heliomar

1. SISTEMAS DE NUMERAÇÃO

Sistemas de numeração que mais se destacam: DECIMAL, BINÁRIO, OCTAL E HEXADECIMAL

São identificados de acordo com a sua base e são sempre ponderados .

O sistema de numeração é dito PONDERADO quando, na determinação do valor do número a posição do dígito no número possui um peso.

1.1. SISTEMA DE NUMERAÇÃO DECIMAL

Base: 10

Exemplo: $5422 = 5 \times 10^3 + 4 \times 10^2 + 2 \times 10^1 + 2 \times 10^0$

posição 0
posição 1
posição 2
posição 3

Cada dígito do número acima tem um peso correspondente a posição que ele ocupa. No caso do sistema decimal o peso é expresso em potência de 10 (que é a base do sistema).

Números decimais menores 1 que têm seus pesos com expoente negativo crescendo a medida em que o dígito se afasta da virgula

EXEMPLO : $0,479 = 4 \times 10^{-1} + 7 \times 10^{-2} + 9 \times 10^{-3}$

SISTEMA DECIMAL é dito ser um sistema de base 10, porque possui 10 dígitos diferentes: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9.

1.2. SISTEMA DE NUMERAÇÃO BINÁRIO

Usado em SISTEMAS DIGITAIS os quais trabalham com dois estados discretos.

Base: 2

Dígitos que compõem o sistema : 0 e 1

O valor do número é encontrado usando as mesmas regras do sistema decimal, apenas que agora a base vale 2.

Exemplo: $1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11_{10}$

Com este exemplo já demonstramos como é feita a conversão de um número na base 2 para a base 10.

Sendo o número fracionário o valor é calculado conforme abaixo.

Exemplo: $0,101_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 0,5 + 0 + 0,125 = 0,625_{10}$

1.3. SISTEMA DE NUMERAÇÃO OCTAL

Possui base (também chamada de RAIZ) igual a 8 e oito dígitos: 0, 1, 2, 3, 4, 5, 6 e 7. Na sua conversão para binário observa-se que os dígitos em octal preenchem todas as combinações possíveis de três dígitos binários, daí a relação deste sistema de numeração com processadores digitais.

Exemplo: $177_8 = 1 \times 8^2 + 7 \times 8^1 + 7 \times 8^0 = 64 + 56 + 7 = 127_{10}$

O valor de números fracionários segue a mesma regra usada pelo sistema acima.

Exemplo: $0,45_8 = 4 \times 8^{-1} + 5 \times 8^{-2} = 0,5 + 0,078125 = 0,578125_{10}$

1.4. SISTEMA DE NUMERAÇÃO HEXADECIMAL

A base deste sistema é 16 e portanto possui 16 dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F.
As letras A, B, C, D, E e F valem respectivamente em decimal 10, 11, 12, 13, 14 e 15.
Como no caso do octal a conversão para binário dos dígitos hexadecimais ocupa todas as combinações possíveis de quatro dígitos binários.

Exemplos: $38AF_{16} = 3 \times 16^3 + 8 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 = 12288 + 2048 + 160 + 15 = 14511_{10}$

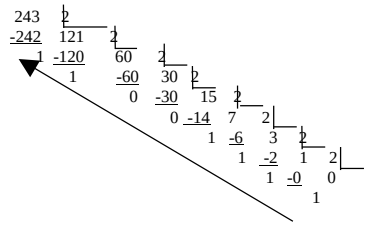
$$0,32_{16} = 3 \times 16^{-1} + 2 \times 16^{-2} = 0,1875 + 0,0078125 = 0,1953125_{10}$$

1.5. CONVERSÃO ENTRE SISTEMAS DE NUMERAÇÃO

A conversão para o decimal foi feita acima para cada um dos outros s istema. O inverso, ou seja, a conversão do decimal para o binário, octal e hexadecimal é feita através de divisões sucessivas pela base tomando-se o resto na seqüência da última para primeira divisão. Os exemplos abaixo ajudarão a esclarecer o que foi dit o.

1.5.1. Conversão do Decimal para Binário

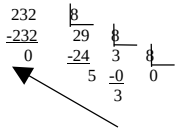
Exemplo: Converter 243_{10} para binário



Assim, $243_{10} = 11110011_2$

1.5.2. Conversão de Decimal para Octal

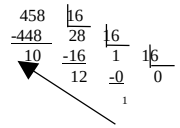
Exemplo: Converter 232_{10} para octal



Assim, $232_{10} = 350_8$

1.5.3. Conversão de Decimal para Hexadecimal

Exemplo: Converter 458_{10} para hexadecimal



Na representação hexadecimal $10_{10} = A_{16}$ e $12_{10} = C_{16}$. Portanto

$458_{10} = 1CA_{16} = 1CA_H$

O "H" também é usado para indicar um número hexadecimal

Conversão de Octal para Binário

A regra consiste em converter intermediáriamente de octal para decimal e, após converter de decimal para binário. Isto no entanto é desnecessário se tomarmos a representação binária (em três dígitos) de cada dígito octal.

Dígito em Octal	Representação Binária
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Exemplo: $37_8 = 011\ 111 = 11111_2$

$3_8\ \ 7_8$

1.5.4. Conversão de Hexadecimal para Binário

Mesma regra usada na conversão de octal para binário. O dígi to hexadecimal é representado binariamente como uma combinação de 4 dígitos 0's e 1's, conforme abaixo.

Exemplo: Converter $2AF_H$ em binário

$2AF_H = 0010\ 1010\ 1111 = 1010101111_2$

$2_H\ \ A_H\ \ F_H$

Dígito em Hexadecimal	Representação Binária
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

1.5.5. Conversão de Binário para Octal

Divide-se o número binário em grupos de 3 e pega-se a representação octal de cada um deles. Para número inteiro a seqüência é da direita para esquerda. Para número fracionário a seqüência é da esquerda para a direita.

Exemplos: Converter 101001110_2 para octal

$101001110_2 = 101\ 001\ 110 = 516_8$

Converter $0{,}11011_2$ para octal

$0{,}11011_2 = 0{,}110\ 110 = 0{,}66_8$

Introduzido para completar a representação octal do dígito.

1.5.6. Conversão de Binário para Hexadecimal

Mesma regra que a da conversão de binário para octal mudando-se o número de dígitos para cada grupo que agora é de 4.

Exemplos: Converter 11010011011_2 para hexadecimal

$11010011011_2 = 0110\ 1001\ 1011 = 69B_H$

Introduzido para completar a representação hexadecimal do dígito.

Converter $0{,}100100102$ para hexadecimal

$0{,}10010010_2 = 0{,}1001\ 0010 = 0{,}92_H$

1.6. REPRESENTAÇÃO EM BCD

BCD = Binary Coded Decimal (Decimal Codificado em Binário)

O dígito decimal é convertido diretamente em binário (em grupo de 4 dígitos). A tabela de conversão é parte da tabela Número Binário x Número Hexadecimal apresentada acima.

Vantagem: Facilidade na conversão de binário para decimal, pois pôr este código a conversão é DIRETA.

Exemplo: Converter 6758_{10} para binário em BCD

$6758_{10} = 0110\ 0111\ 0101\ 1000 = 110011101011000_2$

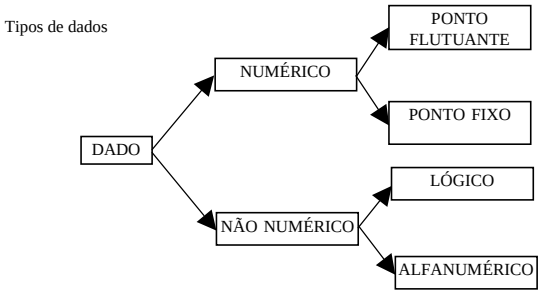
6 7 5 8

OBSERVAÇÃO: Na conversão de binário para decimal e vice versa é importante saber se está usando conversão em BCD ou binária direta, pois os resultados são diferentes. Veja que o número 243_{10} quando convertido em binário direto dá 11110011_2. Sua conversão em BCD dá $0010\ 0100\ 0011_2$, ou seja 1001000011_2.

Para esclarecer apresentamos abaixo a tabela para conversão em BCD

Dígito Decimal	Representação Binária
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

1.7. REPRESENTAÇÃO DE NÚMEROS EM PONTO FLUTUANTE



A representação em ponto fixo pode ser derivada diretamente das representações que temos até agora adotado, ou seja, o ponto que separa a parte inteira da fracionária é alocado implicitamente ou explicitamente numa posição fixa na fo rmatação do número. Ela no entanto não é adequada para representação de números muito grandes ou muito pequenos. Para isto usamos a representação em PONTO FLUTUANTE, conforme veremos.

Em geral um número em ponto flutuante consiste de um par de números em ponto fixo. Um desses pares se constitui a MANTISSA (M) e o outro o EXPOENTE (E).

Assim,

$$N = M \times B^{E}$$

onde:

N = valor do número na base decimal

M = mantissa

B = base em que o número está de representado

E = expoente

1.7.1. Representação em ponto flutuante conforme IEE 754 (32 bits)

$$N = (-1)^{S} . 2^{E-127}(1,M)$$

N = valor do número na base decimal

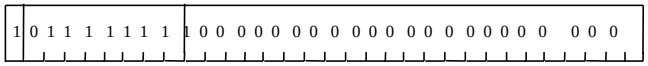
M = mantissa

B = base em que o número está de represent ado

E = expoente

S = 0 se N é positivo 1 se N é negativo

Exemplo: O número $-1,5_{10}$ é representado pôr



$$N = (-1)^1 . 2^{127-127}.(1,1_2) = (-1).2^0.(1,5) = -1,5_{10}$$

O padrão acima permite representar números até cerca de $3,4 \times 10^{38}$ enquanto que a representação em ponto fixo no formato inteir o com a mesma quantidade de bits (32) não pode passar de $2^{32} - 1$ (cerca de $2,1 \times 10^9$).

2. ARITMÉTICA BINÁRIA

2.1. SOMA E SUBTRAÇÃO

Regras para a soma :

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 e vai 1 (carry)

Regras para a subtração:

0 - 0 = 0
0 - 1 = 1 e pede emprestado (borrow)
1 - 0 = 1
1 - 1 = 0

Exemplo: Somar 0111_2 com 0010_2

vai 11
0111
\+ 0010
1001_2

Exemplo : Subtrair 0010_2 de 1001_2

pede emprestado 11
1001
\- 0010
0111_2

2.1.1. Subtração pôr Complemento

Complemento da base (2)

Complemento de 2 de um número binário é o que somado ao número dá o valor da base .

Exemplo: O complemento de 2 de 0101_2 é 1011_2 pois,

$$0101_2 + 1011_2 = 1\,0000_2$$

Observe que aqui, um conjunto de 4 bits, na verdade está representando binariamente um número hexadecimal (cuja base é 16) e que o número $1\,0000_2 = 2^4 = 16$

A regra simples para se achar o complemento de 2 de um número binário e inverter os bits do número original e somar 1ao resultado. Vejamos o caso acima.

Exemplo: Achar o complemento de 2 de 0101_2

$\overline{0101_2} = 1010_2$ — Esta barra significa inversão (negação) do número

$1010_2 + 1_2 = 1011_2$

O resultado da subtração entre dois números pode ser encontrado fazendo -se a soma do primeiro com o complemento de 2 do segundo.

Assim podemos calcular $1100_2 - 1001_2$ conforme abaixo

Complemento de 2 de 1001_2 é 0111_2

```
  1100
+ 0111
1 0011
```

Este 1 é desprezado

Complemento de 1

Neste caso o complemento é o que falta para se chegar a BASE - 1

Encontra-se o complemento de 1 de um número invertendo -se todos os seus bits.

Exemplo: Achar o complemento de 1 de 1101_2

$\overline{1101_2} = 0010_2$

Também podemos calcular a subtração entre dois números através da soma do primeiro com o complemento de 1 do segundo, conforme abaixo.

Exemplo: Calcular $1100_2 - 1001_2$

Complemento de 1 de 1001_2 é 0110_2

Assim

```
   1
  1100
+ 0110
  0010
+ 1
  0011
```

Acrescentado ao resultado pois não foi considerado na conversão para o complemento de 1

VANTAGENS DA OPERAÇÃO PÔR COMPLEMENTO:

Reduz as operações de soma e subtração à operações de soma (um tipo apenas de circuito ou programa)

Operações de inversão e acréscimo de 1 (+1) são fáceis de se implementar, tanto a nível de circuito como de programação (b uilt in functions).

2.2. MULTIPLICAÇÃO E DIVISÃO

MULTIPLICAÇÃO

Regras para multiplicação binária:

$1 \times 1 = 1$
$1 \times 0 = 0$
$0 \times 1 = 0$
$0 \times 0 = 0$

A multiplicação entre números binários é feita da mesma maneira que em números decimais

Exemplo: Calcular $1011_2 \times 101_2$ ($11_{10} \times 5_{10} = 55_{10}$)

$1011_2 = 11_{10}$
$\times \quad 101_2 = 5_{10}$
1011
000
1011
$110111_2 = 55_{10}$

DIVISÃO

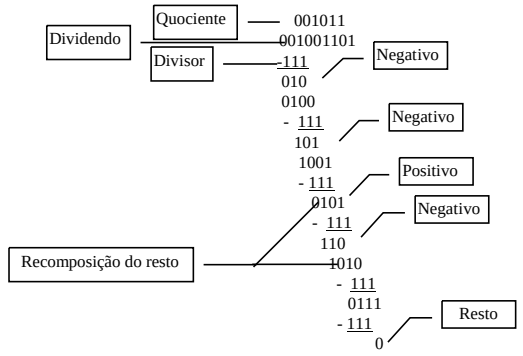
O quociente é encontrado da seguinte forma:

Subtrai-se o dividendo do divisor

a) Se o resto for negativo o quociente é zero. Desloca -se o dividendo de uma casa para a direita e novamente se faz a subtração. Se o resto continua negativo o processo se repete.

b) Se o resto for positivo o quociente é 1. Mant ém-se o resto e a direita do mesmo vai o próximo bit do dividendo. Subtrai -se em seguida do divisor. Se o resto for negativo recompõe-se o resto parcial, o quociente é zero, e à direita do resto parcial vai o próximo dígito do dividendo. Subtrai -se em seguida o divisor.

Exemplo: Calcular $1001101_2 \div 111_2$ ($77_{10} \div 7_{10} = 11_{10}$ resto 0_{10})



3. ÁLGEBRA BOOLEANA

Trabalha com funções lógicas ou funções binárias.

BINÁRIAS : Suas variáveis podem assumir dois valores **Ü** 0 e 1, ACIMA E ABAIXO, ALTO E BAIXO, NEGATIVO E POSITIVO ETC

É usando a Álgebra de BOOLE que projetamos , construímos e programamos circuitos lógicos e digitais.

3.1. REPRESENTAÇÃO DE FUNÇÕES LÓGICAS

Como tanto as variáveis de entrada como a saída de uma função lógica atingem somente dois valores, uma representação consiste na forma de tab ela em são mostradas todas as combinações possíveis das entradas ,com cada combinação de entrada em uma linha da tabela e nesta mesma linha , na coluna da saída, o valor correspondente ao valor de saída da função. A esta tabela denominamos **TABELA VERDADE.**

Exemplo: Construir a tabela verdade da função lógica "NÚMERO DE ENTRADAS PAR".

ENTRADAS				SAÍDA
D	C	B	A	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Outra forma é representação através de EQUAÇÕES BOOLEANAS padronizadas chamadas CANÔNICAS.

Temos duas formas: SOMA DE PRODUTOS e PRODUTO DE SOMAS

SOMA DE PRODUTOS

Tomam-se as linhas da TABELA VERDADE que corr espondem as saídas em 1.

Exemplo: Para a TABELA VERDADE acima escrever a função booleana na forma de SOMA DE PRODUTOS canônicos.

$$Y = \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + \underline{AB\bar{C}\bar{D}} + \bar{A}\bar{B}C\bar{D} + A\bar{B}C\bar{D} + \bar{A}BC\bar{D} + ABC\bar{D}$$

MINTERMO

SIMPLIFICAÇÃO DA REPRESENTAÇÃO

As combinações de variáveis podem ser substituídas pôr números pôr conversão binária considerando que, estando na sua forma natural seu valor é 1. Estando na forma barrada se u valor é 0.

Assim,

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

$$Y = 0000 + 0011 + 0101 + 0110 + 1001 + 101\,0 + 1100 + 1111$$

$$Y = \Sigma\, m(0, 3, 5, 6, 9, A, C, F)$$

PRODUTO DE SOMAS

Tomam-se agora as linhas da TABELA VERDADE que correspondem as saídas em 0. Aqui também a variável é natural quando seu valor é 0 e barrada quando seu valor é 1

Exemplo: Para a TABELA VERDADE acima escrever a função booleana na forma de PRODUTO DE SOMAS canônicos

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

MÁXTERMO

SIMPLIFICAÇÃO DA REPRESENTAÇÃO

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

$(0 + 0 + 0 + 1)\ (0 + 0 + 1 + 0)\ (0 + 1 + 0 + 0)\ (0 + 1 + 1 + 1)\ (1 + 0 + 0 + 0)$

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

$(1 + 0 + 1 + 1)\ (1 + 1 + 0 + 1)\ (1 + 1 + 1 + 0)$

$Y = \Pi M\ (1, 2, 4, 7, 8, B, D, E)$

3.2. POSTULADOS DA ÁLGEBRA DE BOOLE

3.2.1. Definição do conjunto

P1A - Se A ≠ 0 então A = 1

P1B - Se A ≠ 1 então A = 0

3.2.2. Definição dos Operadores

P2A - 0 . 0 = 0 - PRODUTO LÓGICO

P2B - 1 + 1 = 1 - SOMA LÓGICA

P3A - 1 . 1 = 1 - PRODUTO LÓGICO

P3B - 0 + 0 = 0 - SOMA LÓGICA

P4A - 1 . 0 = 0 . 1 = 0

P4B - 0 + 1 = 1 + 0 = 1

LEI COMUTATIVA

A . B = B . A ou A + B = B + A

3.2.3. Existência do Complemento

P5A - $\overline{0}$ = 1

P5B - $\overline{1}$ = 0

DUPLO COMPLEMENTO

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

3.3. TEOREMAS DA ÁLGEBRA DE BOOLE

T1A - A + 0 = A

T1B - A . 1 = A

Provar que A + 0 = A

Se A = 1 1 + 0 = 1
Se A = 0 0 + 0 = 0

Provar que A . 1 = A

Se A = 1 1 . 1 = 1
Se A = 0 0 . 1 = 0

T2A - A + 1 = 1
T2B - A . 0 = 0

A prova decorre direto dos postulados P2A a P4B.

T3A - A + A = A

T3B - A . A = A

T4A - A + $\bar{A}$ = 1

T4B - A . $\bar{A}$ = 0

T5A - A . B = B . A

Teorema Relativo à Lei Comutativa

T5B - A + B = B + A

T6A - A . B . C = A . (B . C) = (A . B) . C

Teorema relativo à Lei Associativa

T6B - A + B + C = A + (B + C) = (A + B) + C

T7A - A . B + A . C = A . (B + C)

Teorema relativo à Lei Distributiva

T7B - (A + B) . (A + C) = A + B . C

T8A - A . B + A . $\bar{B}$ = A

T8B - (A + B) . (A + $\bar{B}$) = A

T9A - A + A . B = A

T9B - A . (A + B) = A

Provar que A + A . B = A

A . (1 + B) = A . 1 = A

Provar que A . (A + B) = A

A . A + A . B = A + A . B = A . (1 + B) = A . 1 = A

T10A - A + $\bar{A}$. B = A + B

T10B - A . ($\bar{A}$ + B) = A . B

Provar que A + $\bar{A}$. B = A + B

A + $\bar{A}$. B = A + $\underline{A . B}$ + $\bar{A}$. B = A + B . (A + $\bar{A}$) = A + B

Introduzido sem alterar o termo já que A + A .B = A . (B + 1) = A . 1 = A

T11A - A . B + A . $\bar{B}$. C = A . B + A . C

T11B - (A + B) . (A + $\bar{B}$ + C) = (A + B) . (A + C)

Provar que A . B + A . $\bar{B}$. C = A . B + A . C

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

T12A - A . B + $\bar{A}$. C + B . C = A . B + $\bar{A}$. C

T12B- (A + B) . ($\bar{A}$ + C) . (B + C) = (A + B) . ($\bar{A}$ + C)

T13A - $\overline{A . B . C . \ldots\ldots\ldots . Z}$ = $\bar{A}$ + $\bar{B}$ + $\bar{C}$ + ... + $\bar{Z}$

T13B - $\overline{A + B + C + \ldots\ldots\ldots + Z}$ = $\bar{A}$. $\bar{B}$. $\bar{C}$ . ... $\bar{Z}$

Os teoremas T13A e T13B acima são co nhecidos como **TEOREMAS DE " DE MORGAN"**

T14A - A . B + $\bar{A}$. C = (A + C) . ($\bar{A}$ + B)

T14B - (A + B) . ($\bar{A}$ + C) = A . C + $\bar{A}$. B

Os Teoremas T14A e T14B acima permitem a transformação de uma função produto lógico de somas lógicas em outra função soma lógica de produtos lógicos , e vice versa, desde que um termo possua uma variável no seu estado natural e outro com a variável no estado barrado.

4. CIRCUITOS LÓGICOS

Todos os sistema digitais são construídos com base somente em três portas lógicas básicas.

Estas portas lógicas básicas são: AND (E), OR (OU) E NOT (NÃO)

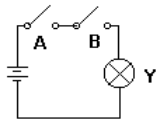
Quando for atribuído o nível H o estado lógico 1 e ao nível L o estado lógico 0 diz -se que a lógica usada é LÓGICA POSITIVA. Se, ao contrário, for atribuído ao nível H o estado lógico 0 e ao nível L o estado lógico 1, diz-se que se trata de LÓGICA NEGATIVA.

Trabalharemos com a lógica **POSITIVA.**

4.1. PORTA AND (E)

DEFINIÇÃO: A SAÍDA ASSUME O NÍVEL LÓGICO L SEMPRE QUE, PELO MENOS, UMA DAS ENTRADAS ASSUME L. A SAÍDA SÓ ASSUMIRÁ H QUANDO TODAS AS ENTRADAS FOREM H.

Como exemplo de porta AND seja o circuito elétri co abaixo



CONVENÇÃO: Entradas: Chave aberta - LOW (L)
Chave Fechada - HIGH (H)

Saída: Lâmpada apagada - LOW (L)
Lâmpada acesa - HIGH (H)

A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H

TABELA DE NÍVEIS LÓGICOS

Observar pela tabela acima (também chamada de TABELA DE NÍVEIS LÓGICOS) que o circuito série obedece à definição de p orta lógica AND (ou E).

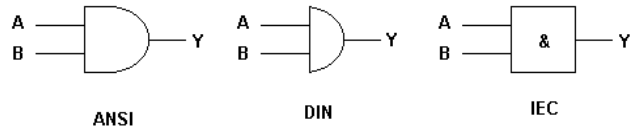
Aqui vale fazer mais um passo que é transformar os níveis H e L ,aqui ligados a implementação física do circuito, nos níveis 0 e 1 de conceituação da álgebra de Boole.

Se estamos trabalhando com lógica positiva, conforme acima, a tabela verdade para a porta AND é a mostrada abaixo.

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

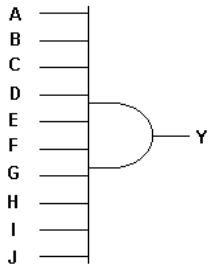
TABELA VERDADE

Simbologia



Y = A . B

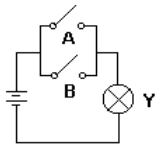
O símbolo abaixo é usado quando o número de entradas é grande (simbologia ANSI)



4.2. PORTA OR (OU)

DEFINIÇÃO: A SAÍDA ASSUME O NÍVEL LÓGICO H SEMPRE QUE, PELO MENOS, UMA DAS ENTRADAS ASSUME H. A SAÍDA SÓ ASSUMIRÁ L QUANDO TODAS AS ENTRADAS FOREM L.

Como exemplo de porta OR seja o circuito elétrico abaixo



A convenção é a mesma da porta AND.

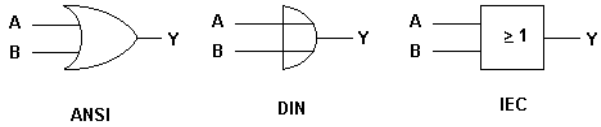
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	H

TABELA DE NÍVEIS LÓGICOS

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

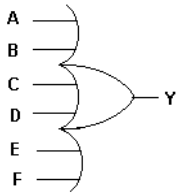
TABELA VERDADE

Simbologia



Y = A + B

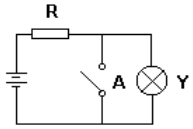
O símbolo abaixo é usado quando o número de entradas para uma porta OR é grande.



4.3. PORTA NOT (NÃO)

DEFINIÇÃO: A SAÍDA ASSUME O NÍVEL LÓGICO H SOMENTE SE ENTRADA FOR L E VICE VERSA

Como exemplo de porta NOT seja o circuito elétrico abaixo



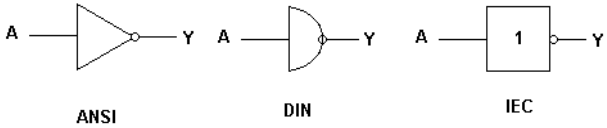
A	Y
L	H
H	L

TABELA DE NÍVEIS LÓGICOS

A	Y
0	1
1	0

TABELA VERDADE

Simbologia

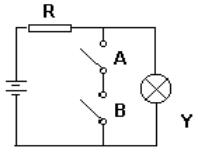


$$Y = \overline{A}$$

4.4. PORTA NAND (NÃ0-E)

A PORTA NAND PODE SER INTERPRETADA COMO UMA PORTA AND EM CUJA SAÍDA É LIGADA UMA PORTA NOT.

Como exemplo de porta NAND seja o circuito elétrico abaixo



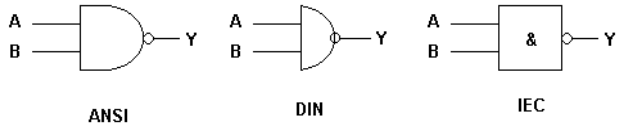
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

TABELA DE NÍVEIS LÓGICOS

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

TABELA VERDADE

Simbologia

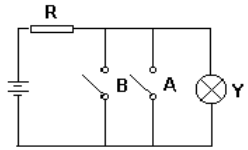


$$Y = \overline{A \cdot B}$$

4.5. PORTA NOR (NÃO-OU)

A PORTA NOR PODE SER INTERPRETADA COMO UMA PORTA OR EM CUJA SAÍDA É LIGADA UMA PORTA NOT.

Como exemplo de porta NOR seja o circuito elétrico abaixo



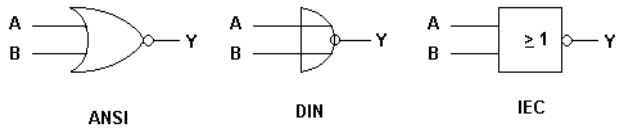
A	B	Y
L	L	H
L	H	L
H	L	L
H	H	L

TABELA DE NÍVEIS LÓGICOS

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

TABELA VERDADE

Simbologia



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

OBSERVAÇÃO: AS PORTAS NAND E NOR SÃO CHAMADAS DE PORTAS UNIVERSAIS, POIS COM ELAS PODEMOS IMPLEMENTAR QUALQUER UMA DAS PORTAS BÁSICAS (AND, OR OR OU NOT).
ASSIM PODEMOS IMPLEMENTAR CIRCUITOS LÓGICOS USANDO SOMENTE PORTAS NAND E NOR E DESTA FORMA TORNANDO-OS MAIS ECONÔMICOS.

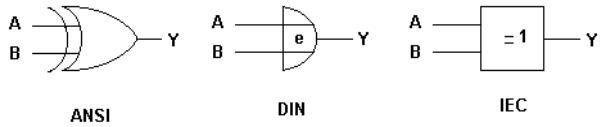
4.6. PORTA EXCLUSIVE OR (OU EXCLUSIVO)

O comportamento de uma porta EXCLUSIVE OR ou porta XOR pode ser visto através da Tabela Verdade, logo abaixo.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

TABELA VERDADE

Simbologia



$$Y = A \oplus B$$

Observando a tabela verdade da porta XOR podemos ver que o número de 1s é sempre par se considerarmos as entradas e saída. Portanto a porta XOR é um GERADOR DE PARIDADE PAR.
Podemos ver também que a saída é a soma dos sinais de entrada senão levamos em conta o "vai um". Portanto a porta XOR também é usada em circuitos somadores.

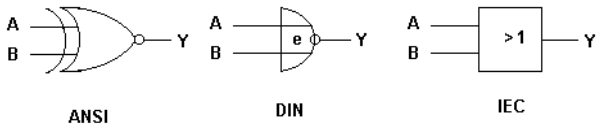
4.7. PORTA EXCLUSIVE NOR (NÃO OU EXCLUSIVO)

É a porta EXCLUSIVE OR complementada. Também chamada de porta NXOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

TABELA VERDADE

Simbologia



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

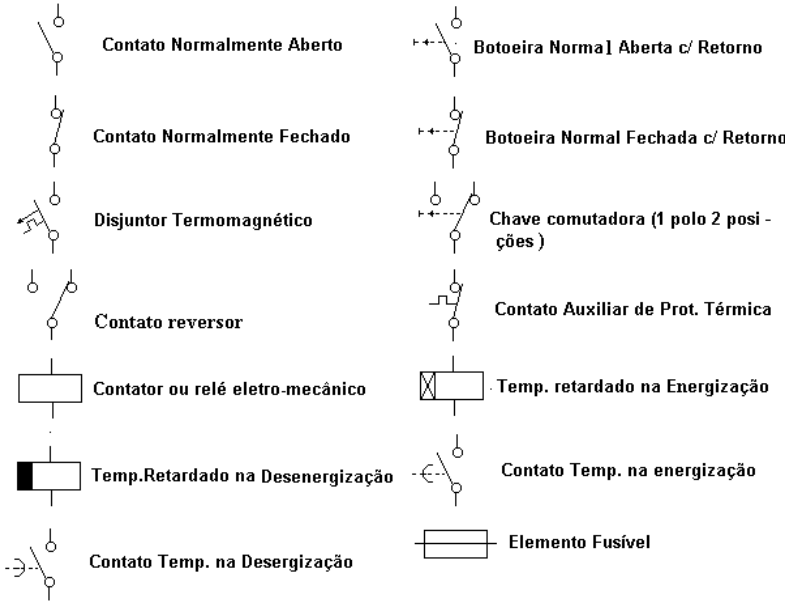
A porta NXOR é um **GERADOR DE PARIDADE ÍMPAR** como pode ser visto pela sua tabela verdade.

4.8. LÓGICA A RELÉS

Muito embora a eletrônica tenha possibilitado grandes avanços na área de sistemas digitais, o comando e controle usando dispositivos eletro -mecânicos (relés) ainda são bastante usados na indústria dada a sua simplicidade e robustez e porque não, boa confiabilidade.
Vale ressaltar que mesmo equipamentos que utilizam eletrônica avançada, simulam o funcionamento de relés eletro-mecânicos para comando e controle industriais, permitindo que o usuário se adapte com rapidez à nova tecnologia e sem necessidade de mão de obra especializada. Exemplo disso são os CONTROLADORES PROGRAMÁVEIS que trabalham com linguagem do tipo LADDER DIAGRAM, uma linguagem típica de relés.
Assim mesmo diminuindo a participação de relés, subsistirão seus projetos.

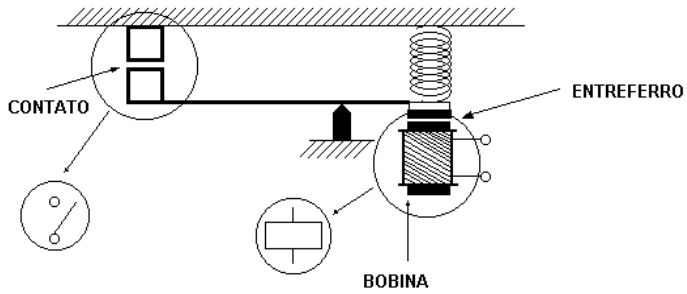
Faremos a seguir um estudo sobre circuito lógicos usando relés.

SIMBOLOGIA

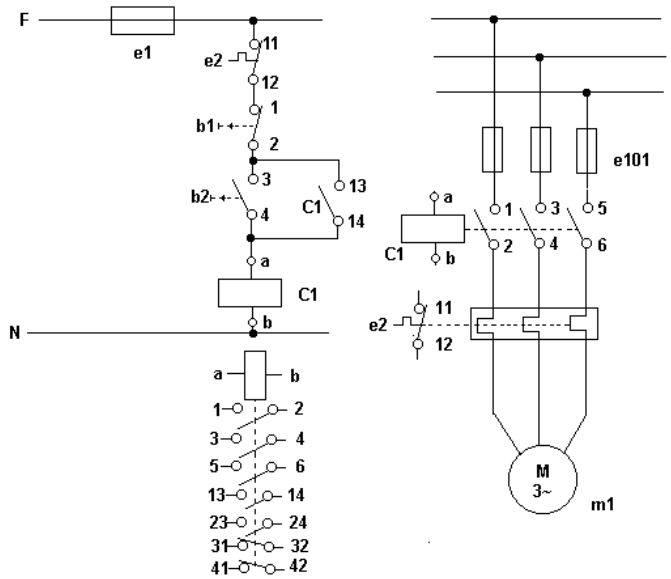


A simbologia acima visa apenas o essencial ao nosso estudo. A compl eta se encontra na norma de símbolos gráficos de Eletricidade P -SB-13 da ABNT.

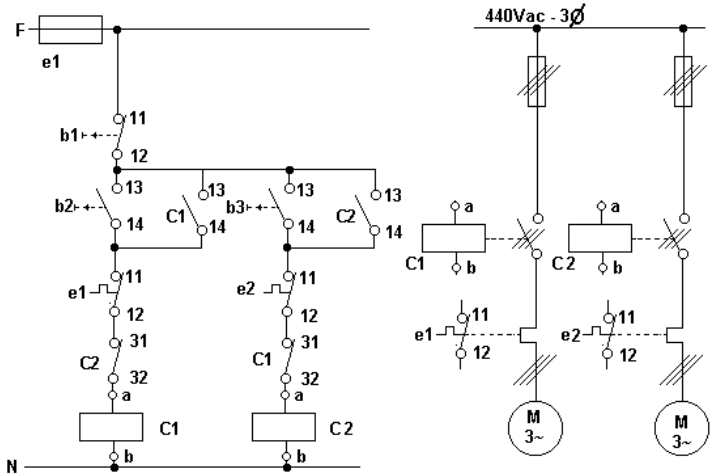
Um relé eletro-mecânico efetua a comutação de seus contatos através da ação de um indutor magnetizado pôr uma corrente elétrica. Um relé industrial pode assumir as mais va riadas formas. Sua estrutura básica é apresentada na figura abaixo.



A) Comando liga desliga com botoeira

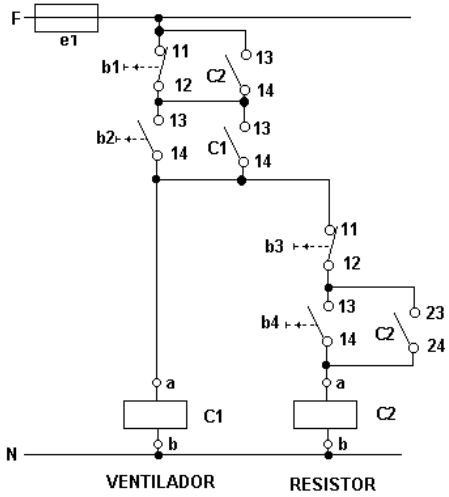


B) Comando Liga/desliga com intertravamento

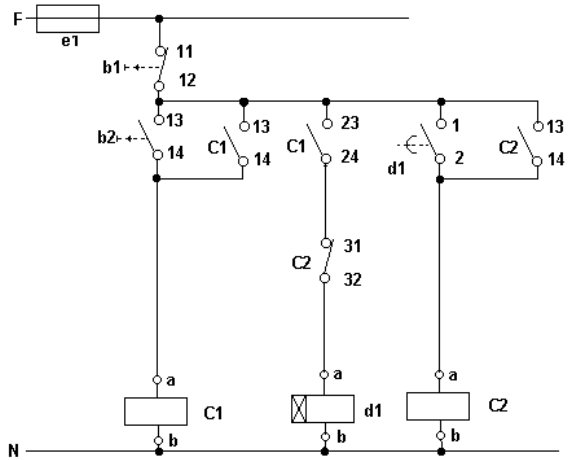


Exemplo 1:Três resistores de aquecimento em 220 Vac - 60 Hz são ligados em triângulo a rede através de um contator tripolar. Associados aos resistores é ligado um ventilador cujo motor trifásico também é comandado pôr um contator. O circuito deve ter o seguinte funcionamento:

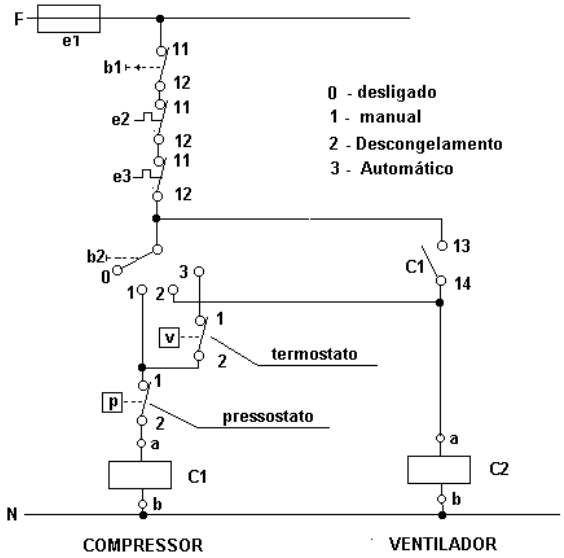
1- Os resistores podem apenas ser ligados após o motor do ventilador estar em funcionamento.
2- Quando o ventilador e resistores estiverem ligados, os resistores podem ser desligados mas o ventilador precisa continuar funcionando.
3- Se o ventilador for desligado os resistores também o serão.



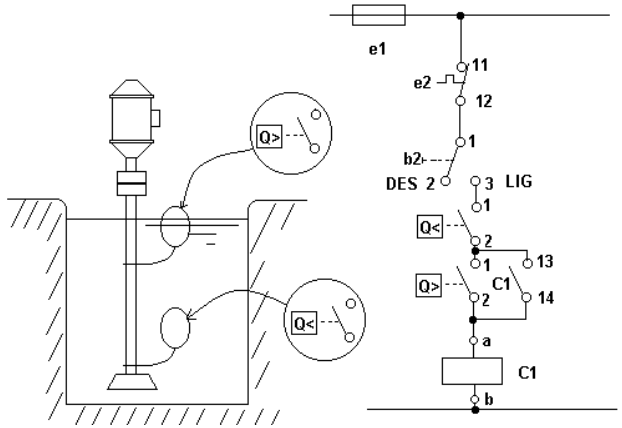
C) Comando Temporizado



D) Controle de temperatura (Sistema de refrigeração)



E) CONTROLE DE NÍVEL

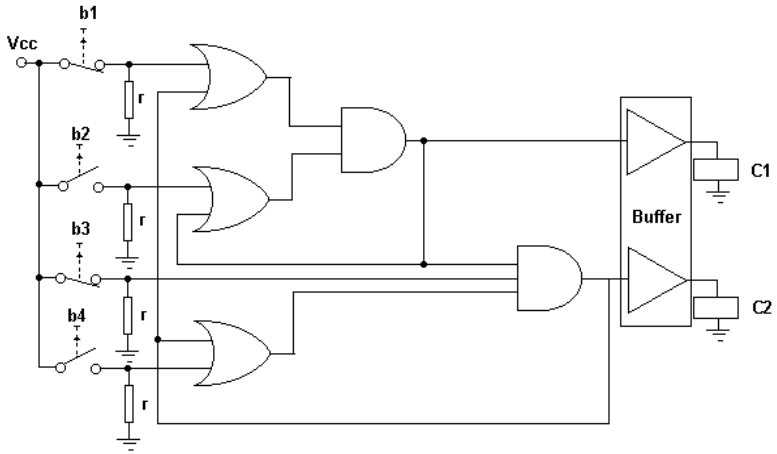


Podemos transformar os diagramas e létricos de comando mostrados anteriormente em portas lógicas e desta forma implementar circuito lógicos eletrônicos que executam as mesmas funções. É bom lembrar que as entradas deste circuito, sendo dispositivos de contato apresentam vibração (bouncin g) que pode ser captada pôr circuitos eletrônicos.

A transformação é feita considerando o seguinte:

1- Contatos em série são representados pôr portas AND.
2- Contatos em paralelo são representados pôr portas OR
3- Contatos normalmente fechados, pôr portas NOT.
4- Cada bobina de relé eqüivale a uma saída de circuito lógico
5- O contato normalmente aberto pertencente a um relé eqüivale a saída do circuito lógico correspondente a bobina deste relé.
6- O contato normalmente fechado eqüivale a saída negada do circuito lógico .

Exemplo: Transformar o circuito do exemplo 1 acima para portas lógicas



Observar que as chaves b1 e b3 não foram seguidas portas inversoras (NOT) como poderia sugerir o que foi dito acima. A razão é que, da forma como foram ligad as, estas já contém a função inversora. Poderíamos substituir a tipo da chave b1 (pôr exemplo) pelo tipo da b2,e aí sim, a porta NOT deveria ser usada.

4.9. PROJETO DE CIRCUITOS LÓGICOS COMBINACIONAIS

Começa com um conjunto de ESPECIFICAÇÕES OPERACIONAIS que podem ser escritas, verbais e imaginadas e termina com um CIRCUITO FUNCIONAL BEM DOCUMENTADO.

PASSOS E PROCEDIMENTOS RECOMENDADOS

1- Analise as especificações e desenvolva um entendimento global do que o seu circuito vai fazer. Certifique-se que alguma variedade de circuito combinacional é a solução.
2- Faça um diagrama de blocos (se necessário) de seu sistema e ilustre a relação com os sistemas de entrada e de saída. Documente com clareza os níveis (0/1) das entradas e os necessários para as suas saídas. Esta documentação também implica no uso apropriado de mneumônicos para as entradas e saídas.
3- Determine a magnitude do seu projeto. Em resumo, determine a quantidade de entradas e saídas. Pode ser necessário modularizar seu projeto em subsistemas se o pro jeto tornar-se muito grande.
4- Desenvolva a TABELA VERDADE definindo as exigências do seu projeto.
5- Utilize-se de técnicas de minimização para simplificação do circuito.
6- Desenvolva o diagrama do circuito ótimo levando em consideração:
   a) Custo da implementação
   b) Disponibilidade de portas lógicas dentro do sistema.
   c) Critério de projeto (Pôr exemplo: Uso preferencial de um determinado tipo de porta lógica (NAND/NOR)).

7- Monte o circuito, depure-o e após atualize a documentação de forma que esta retrate fielmente o circuito conforme foi construído (documentação chamada agora de AS BUILT).

Exemplo: Projete um circuito combinacional com duas entradas e uma saída em que a saída sempre fica no nível lógico 1 quando as entradas estão em níveis lógicos diferentes .

Passo 1:

Verificado que a solução pode ser implementada através de circuito combinacional, pois o valor da saída depende apenas das combinações das entradas.
Entendido de forma global como deve funcionar o c ircuito.

Passo 2:

As entradas serão acionadas através de chaves botoeiras e a saída acenderá um LED de corrente de 10 mA.

A tensão de alimentação das entradas, saídas e do circuito é 5 Vcc com capacidade de 1 A de corrente.

Definição dos níveis lógicos das entradas

	Nível Lógico 1	Nível Lógico 0
Tensão de entrada	>= 2,0 Vcc	<= 0,8 Vcc
Corrente de Entrada	<= 40 µAcc (+)	<= 1,6 mAcc (-)

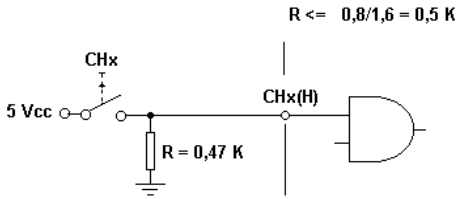
(*) (+) - corrente entrando no circuito
(-) - corrente saindo do circuito

Definição dos níveis lógicos da saída

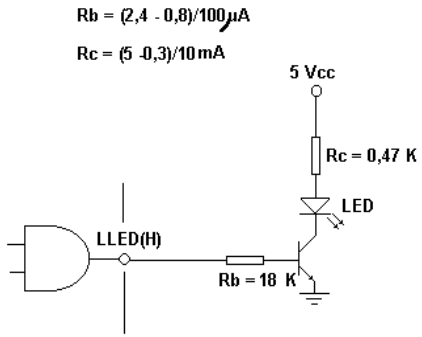
	Nível Lógico 1	Nível Lógico 0
Tensão de entrada	>= 2,4 Vcc	<= 0,4 Vcc
Corrente de Entrada	<= 400 µAcc (-)	<= 16 mAcc (+)

(*) (+) - corrente entrando no circuito
(-) - corrente saindo do circuito

Circuito de entrada



Circuito de saída



Passo 3:

Número de entradas: 2
Número de saídas: 1
O projeto é pequeno. Não há necessidade de su bsistemas.

Passo 4:

TABELA VERDADE

CH1	CH2	LLED
0	0	0
0	1	1
1	0	1
1	1	0

Passo 5:

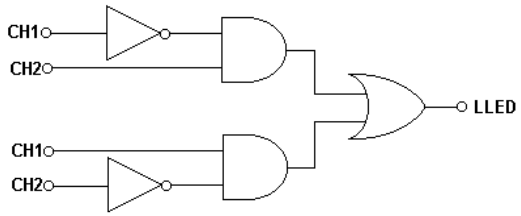
O circuito obedece a seguinte equação booleana

$LLED = \overline{CH1} \cdot CH2 + CH1 \cdot \overline{CH2}$

Aplicaremos mais na frente técnicas de minimização de circuitos lógicos

Passo 6:

O circuito abaixo é a implementação da equação lógica encontrada no passo 5.



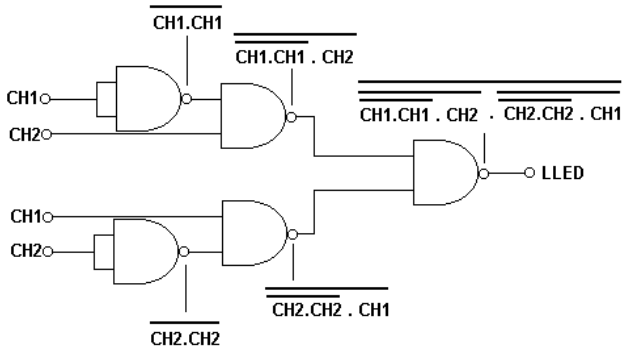
O circuito acima utiliza três tipos de porta lógicas (AND, OR e NOT). Um critério de otimização pode ser a utilização de apenas um tipo de porta como pôr exemplo o uso de portas NAND.
Para isto temos que transformar a equação booleana original, conforme abaixo.

$$LLED = \overline{CH1} \cdot CH2 + CH1 \cdot \overline{CH2}$$

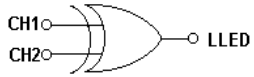
$$\overline{LLED} = \overline{\overline{CH1} \cdot CH2 + CH1 \cdot \overline{CH2}} = \overline{(CH1 \cdot CH2)} \cdot \overline{(CH1 \cdot \overline{CH2})}$$

$$LLED = \overline{\overline{LLED}} = \overline{(CH1 \cdot CH2)} \cdot \overline{(CH1 \cdot \overline{CH2})} = (CH1.CH1.CH2) \cdot (CH1 \cdot \overline{CH2}.CH2)$$

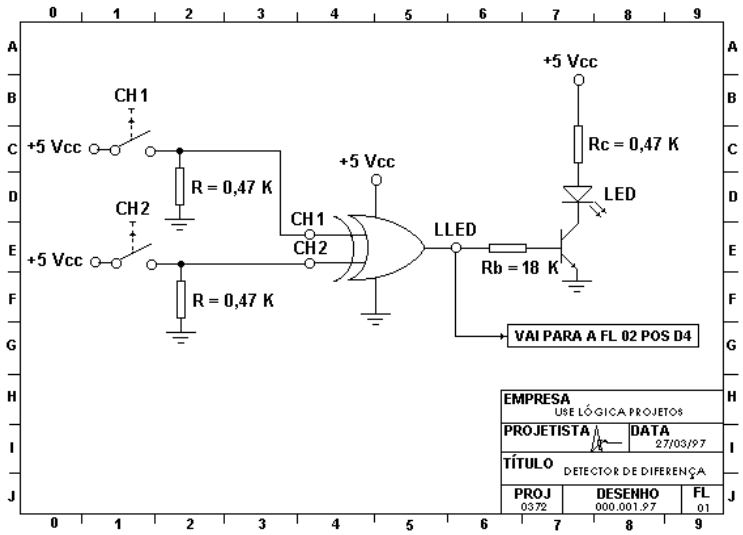
Implementação do circuito



Finalmente e para este caso específico podemos ver que a TABELA VERDADE do circuito em questão é a mesma de uma porta EXCLUSIVE OR e portanto esta é a sua implementação mais simples.

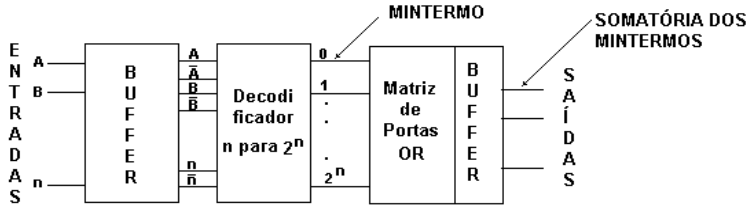


Apresentamos abaixo o documento final após montagem e depuração

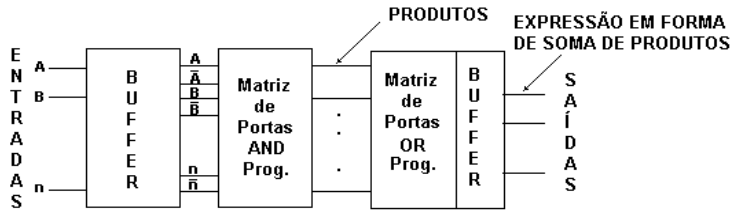


4.10. OUTRAS FORMAS DE PROJETO

Para projeto de circuitos lógicos combinacionais podemos nos utilizar de circuitos integrados de tecnologia LSI (Large Scale Integration) como ROMs (Read Only memory) e suas derivadas PROMs , EPROMs e EEPROMs, PLAs (Proggrammed Logic Arrays) e FPLAs (Field Programmed Logic Arrays).
Para ilustrar apresentamos a abaixo as estruturas simplificadas da ROM e do PLA



ESTRUTURA DA ROM



ESTRUTURA DO PLA

Como pode ser visto a diferença está na decodificação das entradas. Na ROM é um decodificador n x 2^n enquanto no PLA é uma matriz de portas AND programável. Esta diferença elimina o ineficiente armazenamento de mintermos desnecessários que se fazem presentes em face do decodificador que decodifica todas as combinações possíveis das entradas. O PLA permite programar expressões lógicas simplificadas melhor do que as formas canônicas requeridas quando se usam ROMs. Em resumo o PLA é mais eficiente que a ROM.

Exemplo: Projetar a função lógica dada pela tabela da verdade abaixo usando ROM e PLA.

C	B	A	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

A expressão booleana na sua forma canônica (soma de produtos) é :

$Y = \overline{C}\overline{B}\overline{A} + \overline{C}\overline{B}A + \overline{C}B\overline{A} + C\overline{B}\overline{A} + CBA = \Sigma(0, 1,4,6,7)$ **Ü EXPRESSÃO PARA A ROM**

Podemos simplificar a expressão acima usando os teoremas da álgebra de Boole.

Antes porém vamos acrescentar à expressão o termo $\overline{C}\overline{B}\overline{A}$ que de forma alguma a altera. Assim:

$$Y = \overline{C}\overline{B}\overline{A} + \overline{C}\overline{B}A + \overline{C}B\overline{A} + C\overline{B}\overline{A} + CBA + \overline{C}\overline{B}\overline{A}$$

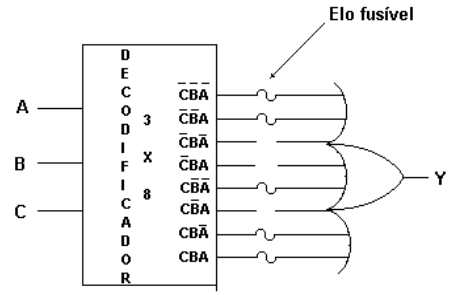
Evidenciando-se $\overline{A}\overline{B}$, $C\overline{B}$ e $\overline{C}\overline{B}$ teremos:

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

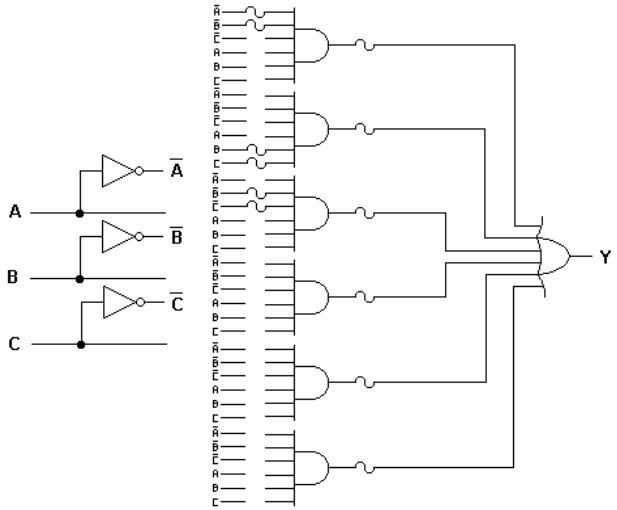
e

$Y = \overline{A}\overline{B} + C\overline{B} + \overline{C}\overline{B}$ **Ü EXPRESSÃO PARA O PLA**

Implementação com ROM



Implementação com PLA



5. MINIMIZAÇÃO DE CIRCUITOS LÓGICOS

Minimização de circuitos lógicos á **DIMINUIÇÃO DE CUSTOS**

Métodos de Minimização : 1 - Uso dos postulados e Teoremas da Álgebra de Boole
2- Mapas de Karnaugh
3- Método Tabular de Quine-McCluskey

Postulados e Teoremas da Álgebra de Boole á Capítulo 3

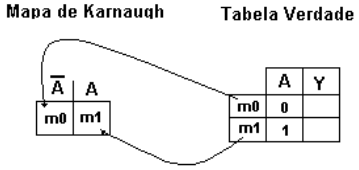
5.1. MAPAS DE KARNAUGH

O Mapa de Karnaugh é uma outra forma de representação de uma equação booleana, com a vantagem de que induz à minimização dos seus termos e portanto do circuito que implementará a função.

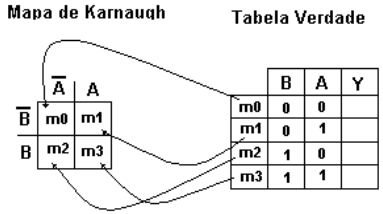
O Mapa de Karnaugh é uma figura geométrica que contém uma região para cada linha de uma dada Tabela Verdade.

Vejamos os exemplos abaixo

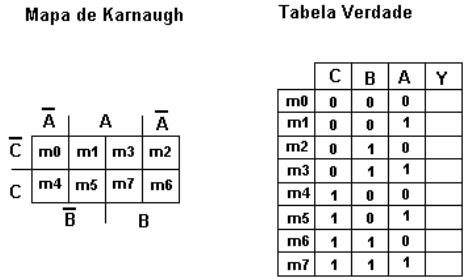
MAPA DE KARNAUGH PARA UMA VARIÁVEL



MAPA DE KARNAUGH PARA DUAS VARIÁVEIS



MAPA DE KARNAUGH PARA TRÊS VARIÁVEIS



Observando o mapa de Karnaugh vemos que regiões vizinhas possuem a propriedade de diferirem em apenas uma variável, uma relação a outra.

Tomemos como exemplo os mintermos vizinhos m5 e m7 do Mapa de Karnaugh de três variáveis acima.

$$m5 = C\overline{B}A \qquad m7 = CBA$$

Assim, se numa expressão Booleana estes termos aparecem, a variável que difere pode ser eliminada pois,

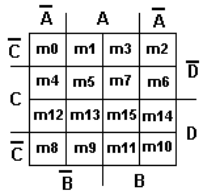
$$Y = CBA + C\overline{B}A + ... = CA(B + \overline{B}) + ... = CA + ...$$

Podemos então dizer que qualquer par de mintermos vizinhos pode ser combinado em um único termo que possui uma variável a menos que os mintermos originais. A variável que desaparece é a que não é comum no par de mintermos.

Interessante observar que m0 e m2 são vizinhos assim como m4 e m6, pois ambos diferem pela variável B.

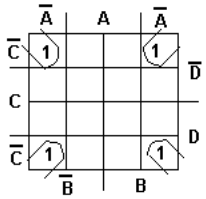
No **Mapa de Karnaugh para quatro variáveis** abaixo encontraremos os seguintes pares de mintermos vizinhos que aparentemente não parecem ser:

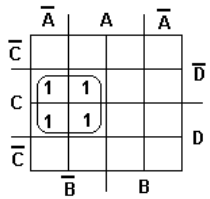
Mintermos Vizinhos	Variável que Difere
m0 e m8	D
m1 e m9	D
m3 e m11	D
m2 e m10	D
m0 e m2	B
m4 e m6	B
m12 e m14	B
m8 e m10	B

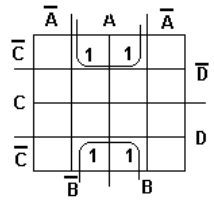


Agrupamentos maiores que um par.

Vejamos os Mapas de Karnaugh abaixo.







No primeiro Mapa a expressão Booleana é : $Y = \bar{D}\bar{C}\bar{B}A + \bar{D}\bar{C}B\bar{A} + \bar{D}C\bar{B}\bar{A} + D\bar{C}\bar{B}\bar{A} =$

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

Desta forma vemos a formação de um grupo de quatro regiões, a que chamamos de QUARTETO permite eliminar duas variáveis.

No segundo Mapa as variáveis eliminadas são A e D e a expressão Booleana é formada pelas variáveis que são comuns no quarteto. Portanto,

$$Y = \bar{B}C$$

No terceiro Mapa as variáveis eliminadas são B e D e a expressão booleana é formada pelas variáveis que são comuns no quartet o. Portanto,

$$Y = C\bar{A}$$

Algoritmo para se chegar a expressão mínima de uma função lógica, usando Mapa de Karnaugh:

1- Assinalar e considerar como essencial qualquer região que não po ssa ser combinada com nenhuma outra.

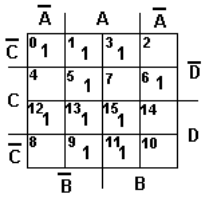
2- Identificar grupos que podem ser formados com uma única região apenas de uma maneira.
Regiões que podem ser combinadas em pares de mais de uma maneira são deixados temporariamente em segundo plano.

3- Identificar regiões que podem ser combinadas com outras três de apenas uma maneira.
Se as quatro regiões de tais combinações ainda são estiverem já sido incluídas na formação de pares assinalar o quarteto. Novamente uma região que pode ser combinada num quarteto de mais de uma maneira deve ser deixada temporariamente em segundo plano.

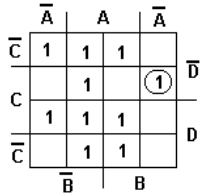
4- Repetir o processo acima para grupo de oito (OCTETOS) etc.

5- Se no final do processo sobrarem regiões não incluídas em grupo, elas podem ser combinadas umas com as outras ou com regiões já incluídas em outros grupos, lembrando que o objetivo é obter o menor número de grupos possível.

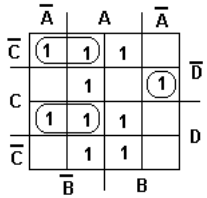
Exemplo 1: Uma função de quatro variáveis é dada pôr Y = f(A,B,C,D) = Σm(0,1,3,5,6,9,11,12,13,15). Use o Mapa de Karnaugh para minimizar a função.



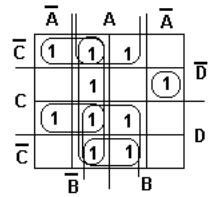
Aplicação do passo 1



Aplicação do passo 2 (m0 e m12 só formam pares de uma só maneira)



Aplicação do passo 3 (m3, m5 e m15 só formam quartetos de uma só maneira)



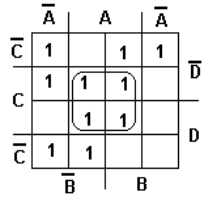
A aplicação dos passos 4 e 5 não foi necessária.

Pelo Mapa de Karnaugh acima vemos que : $Y = \bar{D}\bar{C}BA + \bar{D}\bar{C}\bar{B} + D\bar{C}\bar{B} + DA + \bar{B}A + \bar{C}A$

Exemplo 2: Uma função de quatro variáveis é dada pôr Y =f(A,B,C,D) = Σm(0,2,3,4,5,7,8,9,13,15). Use o Mapa de Karnaugh para minimizar a função.

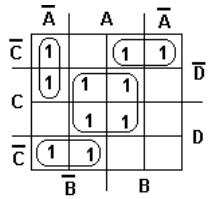
Passos 1 e 2 não se aplicam pois não há implicante primo essencial (não se combina com nenhum outro e nem pares que se combinam de forma única).

Aplicação do passo 3



O passo 4 não se aplica neste caso.

Passo 5



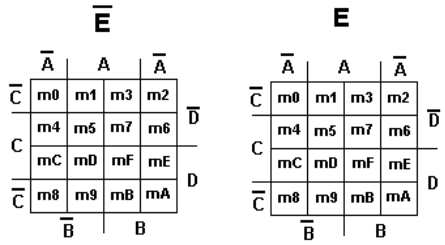
E

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

5.2. MAPAS DE KARNAUGH PARA 5 E 6 VARIÁVEIS

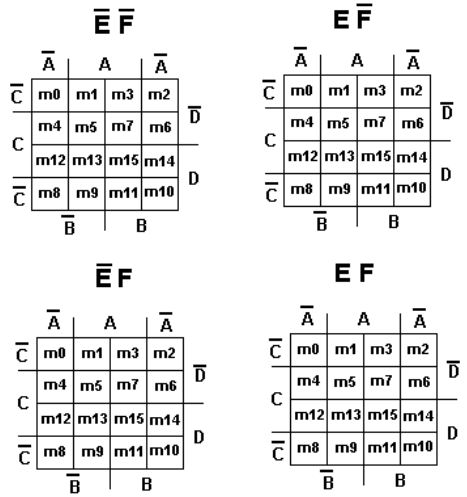
Mapa de Karnaugh para 5 variáveis

Neste caso dispomos dois mapas de 4 variáveis lado a lado, um correspondendo a variável $\overline{E}$ e outro a variável E. As mesmas regiões de cada mapa são vizinhas pois diferem uma da outra apenas pela variável E



Mapa de Karnaugh para 6 variáveis

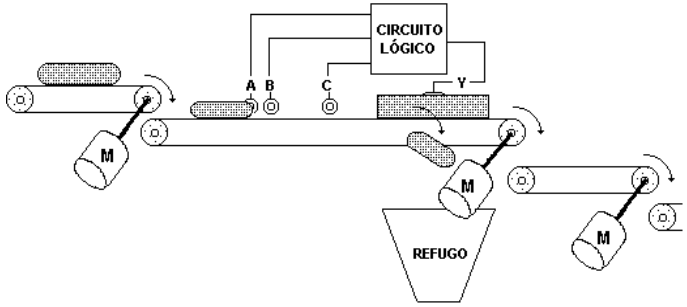
Dispomos 4 mapas de 4 variáveis nos vértices de um quadrado (imaginário). Cada mapa corresponde a uma combinação das variáveis E e F, conforme desenho abaixo.



Para um número de variáveis maiores que 5 ou 6 os mapas ficam muito grandes e de difícil manipulação. Assim outros métodos são aconselháveis tal como veremos no item 5.4 abaixo.

5.3. CONDIÇÕES INDIFERENTES

Consideremos o problema de projetar um circuito lógico que classifique o tamanho de peças em formato cilíndrico tendo como entrada sensores que detectam a presença da peça conforme desenho esquemático abaixo.



Y aciona um pistão que desvia a peça fora do tamanho requerido para a caixa de refugo, ou seja, a peça menor que a distância entre os sensores B e C e maior que a distância entre A e C. Se a peça está dentro da tolerância, Y = 0 e a peça passa p ara a correia seguinte para próximo processamento.
O sensor C quando em 1 mantém a última posição do pistão inibindo a lógica para valores seguintes de A e B. A lógica porém funciona para a última combinação de ABC tendo C = 1.

>AC	=AC	>BC<AC	=BC	<BC	=AB	<AB
CBA	CBA	CBA	CBA	CBA	CBA	CBA
001	001	001	001	001	001	001
011	011	011	011	011	011	000
111	111	110	110	000	000	010
(i)	(i)	(i)	(i)	100	100	000
(i)	(i)	(i)	(i)	(i)	(i)	100
(i)	(i)	(i)	(i)	(i)	(i)	(i)
REFUGO	REFUGO	OK	OK	REFUGO	REFUGO	REFUGO

TABELA VERDADE

C	B	A	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	X
1	1	0	0
1	1	1	1

Observe na tabela acima que a entrada CBA = 101 nunca vai ocorrer numa situação normal de funcionamento o que implica que ao circuito lógico não vai ser a plicada esta combinação e portanto o resultado da saída é irrelevante. Em função disto a saída para esta combinação das entradas recebe a letra X caracterizando a situação citada.

Durante o projeto do circuito pode ser conveniente considerar as saídas par a condições irrelevantes em 1. Isto porque estas podem estar em regiões do mapa de Karnaugh que propiciem agrupamentos envolvendo um maior número de variáveis e portanto simplificando a função lógica.

5.4. MÉTODO TABULAR DE QUINE-MCCLUSKEY

O Mapa de Karnaugh é poderoso mais tem seus inconvenientes. Primeiro, não oferece garantia de que realmente produz a melhor implementação. Segundo, seu processo não é adequado para ser "automatizado", pôr exemplo, através de um programa de computador e, em terceiro, seu grau de dificuldade é grande para implementações com mais de 5 ou 6 variáveis de entrada . Assim lançaremos mão do processo de minimização abaixo que não possui os problemas acima citados.

MÉTODO TABULAR DE QUINE-MCCLUSKEY

Condição: A função deverá estar na sua forma canônica com soma de produtos (somatória de Mintermos).

5.4.1. Procedimentos (para funções completamente especificadas)

1- Formar grupos com os termos canônicos da função que contiverem o mesmo número de bits 1.

2- Formar novos grupos comparando sucessivamente mintermos de um grupo com o seguinte, sendo que os elementos de cada novo grupo ocorrerão quando:

a) Na comparação de um mintermo de um grupo com o seu seguinte a diferença for de uma potência de 2 (1, 2, 4, 8 ...)

b) Na comparação de um mintermo de um grupo com o seu seguinte o primeiro mintermo for menor que o segundo.

Os grupos que se combinarem deverão ser marcados com 4 ao lado.

O passo 2 termina quando não for mais possível formar novos grupos, ou seja quando na comparação entre grupos não ocorrer as condições (a) e (b).

3- Montar o Mapa dos Implicantes Primos
Este mapa ilustra a cobertura dos mintermos da função a ser minimizada feita pôr cada mintermo do último grupo formado. Cada mintermo deste grupo é denominado de IMPLICANTE PRIMO.

4- Subtrair sucessivamente do Mapa dos Implicantes Primos as linhas correspondentes aos IMPLICANTES PRIMOS ESSENCIAIS, ou seja, aqueles que cobrem unicamente mintermos da função a ser minimizada . As colunas correspondentes aos mintermos cobertos são também retiradas e novo mapa é formado .

O passo 4 termina quando todos os mintermos da função a ser minimizada são considerados.

A expressão final é a soma dos implicantes primos cujas linha foram sucessivamente subtraídas na formação dos Mapas dos Implicantes Primos.

Exemplo: Minimizar a função $Y = f(A,B,C,D) = \Sigma m(0,2,3,7,8,10,11,12,14)$

DECIMAL	DCBA
0	0000
2	0010
3	0011
7	0111
8	1000
10	1010
11	1011
12	1100
14	1110

Passo 1:

GRUPO 0

ÍNDICE	Nº DE 1s	MINTERMO	DCBA
0	0	0 4	0000
1	1	2 4	0010
		8 4	1000
2	2	3 4	0011
		10 4	1010
		12 4	1100
3	3	7 4	0111
		11 4	1011
		14 4	1110

Passo 2:

GRUPO 1

ÍNDICE	DECIMAL	DCBA
0	0,2 (2) 4	00-0
	0,8 (8) 4	-000
1	2,3 (1) 4	001-
	2,10 (8) 4	-010
	8,10 (2) 4	10-0
	8,12 (4) 4	1-00
2	3,7 (4)	0-11
	3,11 (8) 4	-011
	10,11 (1) 4	101-
	10,14 (4) 4	1-10
	12,14 (2) 4	11-0

OBS: O número entre parênteses indica o peso da variável que foi eliminada e é repres entada pôr um traço na coluna DCBA.

GRUPO 2

ÍNDICE	DECIMAL	DCBA
0	0,2,8,10 (2,8)	-0-0
	0,8,2,10 (8,2)	-0-0
1	2,3,10,11 (1,8)	-01-
	2,10,3,11 (8,1)	-01-
	8,1012,14 (2,4)	1--0
	8,12,10,14 (4,2)	1--0

Passo 3:

	MINTERMOS								
IMPLICANTES PRIMOS	0	2	3	7	8	10	11	12	14
0,2,8,10 (2,8)	x	x			x	x			
2,3,10,11 (1,8)		x	x			x	x		
8,10,12,14 (2,4)					x	x		x	x
3,7 (4)			x	x					

Passo 4:

	MINTERMOS								
IMPLICANTES PRIMOS	0	2	3	7	8	10	11	12	14
0,2,8,10 (2,8)	[x]	x			x	x			
2,3,10,11 (1,8)		x	x			x	[x]		
8,10,12,14 (2,4)					x	x		[x]	[x]
3,7 (4)			x	[x]					

Neste caso todos os implicantes primos são essenciais.

Traduzindo da coluna BINÁRIO das tabelas dos GRUPOS 1 e 2 para a forma de expressão booleana temos que :

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

5.4.2. Procedimentos (para funções com condições irrelevantes)

O procedimento é o mesmo do item 5.4.1, a menos de que no Mapa dos Implicantes Primos os mintermos correspondentes às condições irrelevantes podem ser ignorados pois não necessitam de ser obrigatoriamente cobertos. A consideração destes mintermos pode ajudar na simplificação da função.

Exemplo: Simplificar a função Y = f(A,B,C,D) = Σm(1,5,8,9,13,14,15) + ΣIr(3,7,10,11,12).

DECIMAL	BINÁRIO
1	0001
3	0011
5	0101
7	0111
8	1000
9	1001
13	1101
14	1110
15	1111

GRUPO 0

ÍNDICE	N^0 DE 1s	MINTERMO	DCBA
0	1	1 4	0001
		8 4	1000
1	2	3 4	0011
		5 4	0101
		9 4	1001
		10 4	1010
		12 4	1100
2	3	7 4	0111
		11 4	1011
		13 4	1101
		14 4	1110
3	3	15 4	1111

GRUPO 1

ÍNDICE	DECIMAL	DCBA
0	1,3 (2) 4	00-1
	1,5 (4) 4	0-01
	1,9 (8) 4	-001
	8,9 (1) 4	100-
	8,10 (2) 4	10-0
	8,12 (4) 4	1-00
1	3,7 (4) 4	0-11
	3,11 (8) 4	-011
	5,7 (2) 4	01-1
	5,13 (8) 4	1-01
	9,11 (2) 4	10-1
	9,13 (4) 4	1-01
	10,11 (1) 4	101-
	10,14 (4) 4	1-10
	12,13 (1) 4	110-
	12,14 (2) 4	11-0
2	7,15 (8) 4	-111
	11,15 (4) 4	1-11
	13,15 (2) 4	11-1
	14,15 (1) 4	111-

GRUPO 2

ÍNDICE	DECIMAL	DCBA
0	1,3,5,7 (2,4) 4	0—1
	1,3,9,11 (2,8) 4	-0-1
	1,5,3,7 (4,2) 4	0—1
	1,5,9,13 (4,8) 4	--0
	1,9,3,11 (8,2) 4	--01
	1,9,5,13 (8,4) 4	10—
	8,9,10,11 (1,2) 4	1-0-
	8,9,12,13 (1,4) 4	10—
	8,10,9,11 (2,1) 4	1—0
	8,10,12,14 (2,4) 4	1--0
	8,12,9,13 (4,1) 4	1—0
	8,12,10,14 (4,2) 4	
1	3,7,11,15 (4,8) 4	--11
	3,11,7,15 (8,4) 4	--11
	5,7,13,15 (2,8) 4	-1-1
	5,13,7,15 (8,2) 4	-1-1
	9,11,13,15 (2,4) 4	1—1
	9,13,11,15 (4,2) 4	1—1
	10,11,14,15 (1,4) 4	1-1-
	10,14,11,15 (4,1) 4	1-1-
	12,13,14,15 (1,2) 4	11--
	12,14,13,15 (2,1) 4	11--

GRUPO 3

ÍNDICE	DECIMAL	DCBA
0	1,3,5,7,9,11,13.15 (2,4,8) 4	---1
	1,3,9,11,5,7,13,15 (2,8,4) 4	---1
	1,5,9,13,3,7,11,15 (4,2,8) 4	---1
	8,9,10,11,12,13,14,15 (1,2,4) 4	1---
	8,9,12,13,10,11,14,15 (1,4,2) 4	1---
	8,10,12,14,9,11,13,15 (2,4,1) 4	1---

	MINTERMOS ESPECIFICADOS							MINTERMOS IRRELEV.				
IMPLICANTES PRIMOS	1	5	8	9	13	14	15	3	7	10	11	12
1,3,5,7,9,11,13,15 (2,4,8)	[x]	[x]		x	x		x	[x]	[x]		x	
8,9,10,11,12,13,14,15 (1,2,4)			[x]	x	x	[x]	x			[x]	x	[x]

A expressão da função simplificada é Y = A + D

6. CIRCUITOS SEQUENCIAIS

O campo da Eletrônica Digital pod e ser dividido em duas áreas: a lógica combinacional e a lógica seqüencial.

Os circuitos combinacionais, como vimos até aqui, apresentam saídas que dependem única e exclusivamente dos valores das variáveis de entrada.

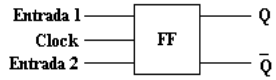
Os circuitos seqüenciais têm suas sa ídas dependentes das variáveis de entrada e/ou de seus estados anteriores.

Os blocos lógicos básicos na implementação de circuitos combinacionais foram apresentados anteriormente e são as portas OR, AND e NOT. Os circuitos seqüenciais utilizam outros blocos lógicos básicos: os Flip-Flops.

Vários circuitos seqüenciais são sistemas pulsados, ou seja, operam sob comando de uma seqüência de pulsos denominada clock.

O flip-flop é um dispositivo que possui dois estados estáveis. Para que o flip -flop assuma um desses estados é necessário que haja uma combinação das variáveis e um pulso de controle a que chamamos de **clock**. Após este pulso o estado não se alterará até a chegada de um novo pulso e, então, de acordo com as variáveis de entrada e com o estado prese nte, permanecerá ou mudará de estado.

O flip-flop pode ser representado pôr um bloco com duas saídas Q e $\overline{Q}$, uma entrada de controle (clock) e as entradas para as variáveis.

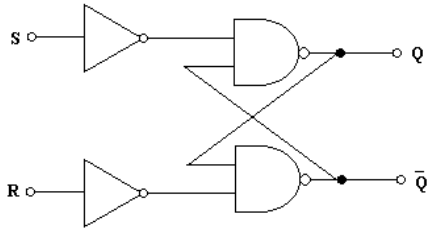


Os dois estados possíveis são:

$$Q = 0 \rightarrow \overline{Q} = 1$$
$$Q = 1 \rightarrow \overline{Q} = 0$$

6.1. FLIP-FLOP RS

Primeiramente vamos analisar o flip-flop RS básico, construído a partir de portas NAND.



Note que as saídas são realimentadas e injetadas junto com as variáveis de entrada.

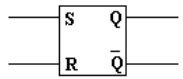
Para analisarmos o circuito vamos construir a tabela verdade, levando em consideração as variáveis de entrada (R e S) e a saída Q.

S R Qa	Qf $\overline{Qf}$	
0 0 0	0 1	Qf = Qa
0 0 1	1 0	Qf = Qa
0 1 0	0 1	Qf = 0
0 1 1	0 1	Qf = 0
1 0 0	1 0	Qf = 1
1 0 1	1 0	Qf = 1
1 1 0	1 1	não permitido
1 1 1	1 1	não permitido

Podemos então resumir a tabela verdade de um flip-flop RS básico como:

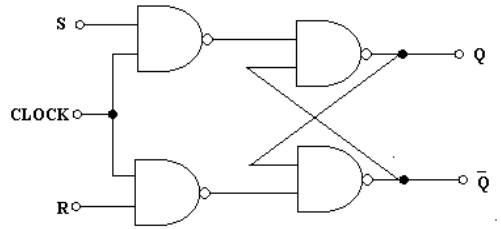
S R	Qf
0 0	Qa
0 1	0
1 0	1
1 1	não permitido

O símbolo do circuito é mostrado abaixo:

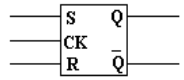


Pode-se fazer um flip-flop RS comandado pôr uma seqüência de pulsos de clock. Para isto basta trocar os dois inversores pôr portas NAND, e injetar o sinal de clock nas outras entradas destas portas.

O circuito resultante é mostrado abaixo.

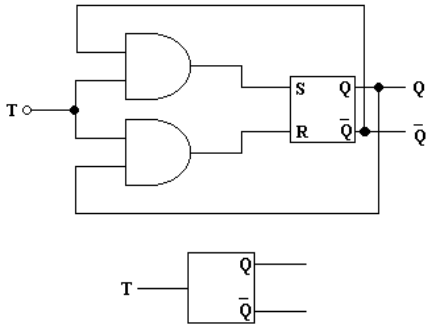


Neste circuito, quando a entrada do clock for 0, o flip-flop irá permanecer no mesmo estado mesmo que as entradas R e S variem. Quando a entrada clock for 1, o circuito irá se comportar como um flip-flop RS básico. Este flip-flop é portanto sensível a nível e seu símbolo é mostrado abaixo.



6.2. FLIP-FLOP TRIGGER OU T

O flip-flop trigger ou T pode ser obtido do flip-flop RS como mostrado na figura abaixo.

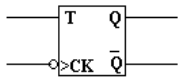


Aplicando um pulso à sua entrada causa-se uma variação no estado da saída. Portanto, se Q = 0 um pulso aplicado em T pulsa S e causa uma variação na saída para Q = 1; se Q = 1, um pulso aplicado a T pulsa R e causa uma variação na saída para Q = 0. A fim de oper ar propriamente, este tipo de flip-flop deve apresentar um "delay" (atraso) entre a aplicação do pulso em R ou S e a variação no estado da saída Q. O pulso deve terminar antes que a saída varie de estado. Assim a saída não variará até que um novo pulso che gue. Se isto não acontecer a saída Q será realimentada na entrada causando uma nova variação no estado do flip-flop.

A tabela verdade do flip-flop é dada abaixo.

T Q	Qf
0 0	0
0 1	1
1 0	1
1 1	0

A figura abaixo mostra um flip-flop T com sinal de clock de entrada.

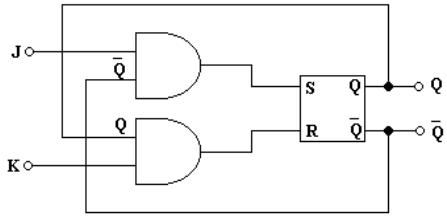


O sinal de maior identifica que o flip-flop é sensível a transição do pulso de clock e o pequeno círculo (símbolo de inversão) indica que o flip-flop varia de estado na descida deste pulso (transição negativa). Assim este flip-flop muda de estado apenas se na descida do pulso de clock a entrada T estiver em 1. A tabela verdade neste caso é como a do flip-flop T sem clock.

6.3. FLIP-FLOP JK

O flip-flop JK une as características dos flip-flops RS e T. Assim quando uma entrada 1 é aplicada a J ou K age como se um 1 estivesse aplicado em S ou R, respectivamente. Mas se J e K forem simultaneamente 1, o comportamento será como no flip-flop T.

O flip-flop JK pode ser obtido do flip-flop RS realimentando suas saídas como na figura abaixo.



A tabela verdade para este flip-flop com a entrada clock igual a 1 é:

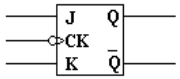
J K Qa	$\overline{Qa}$	S R	Qf	
0 0 0	1	0 0	Qa	+Qa
0 0 1	0	0 0	Qa	+
0 1 0	1	0 0	Qa (Qa = 0)	+0
0 1 1	0	0 1	0	+
1 0 0	1	1 0	1	+1
1 0 1	0	0 0	Qa (Qa = 1)	+
1 1 0	1	1 0	$\overline{Qa}$	+ $\overline{Qa}$
1 1 1	0	0 1	$\overline{Qa}$	+

A tabela simplificada é:

J K	Qf
0 0	Qa
0 1	0
1 0	1
1 1	$\overline{Qa}$

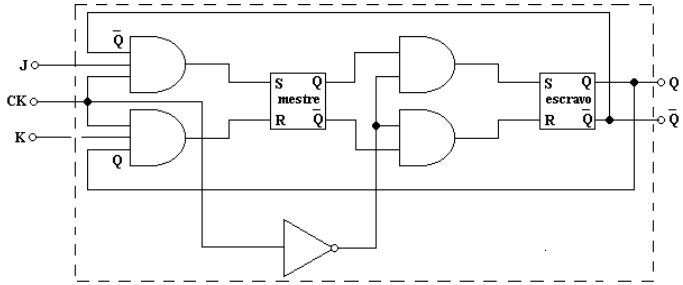
No caso em que J = 1 e K = 1, para obter-se Qf = $\overline{Qa}$ é necessário que J=K=1 permaneça um tempo menor que o necessário a variação na saída do flip-flop RS, senão a saída entrará em constante mudança (oscilação) provocando uma indeterminação.

Pode-se construir um flip-flop JK com sinal de clock como o apresentado na figura abaixo.



Este flip-flop varia de estado apenas na descida do pulso de clock para determinados valores de J e K. Portanto se J = 1 durante o pulso de clock a saída mudará para 1 assim que o pulso de clock for para zero. Novamente, o sinal de maior indica que o flip-flop é sensível a transição do pulso de clock e o pequeno círculo indica que a variação de es tado ocorre após a descida do clock.

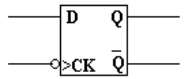
O flip-flop JK com sinal de clock pode ser construído usando flip-flops RS em um arranjo mestre-escravo (master-slave) como mostrado na figura abaixo.



Quando o clock for 1, as entradas J e K terão passagem (circuito Mestre), mas Q_1 e $\overline{Q_1}$ (entradas S e R do circuito Escravo) não terão passagem, pois o clock em zero bloqueará suas entradas. Quando o clock for a zero, as saídas Q_1 e $\overline{Q_1}$ ficarão bloqueadas no último estado e entrarão em R e S desbloqueadas, mudando o estado do circuito Escravo e as saídas Q e $\overline{Q}$. Para uma operação própria deste circuito, suas entradas J e K devem variar entre os pulsos de clock.

6.4. FLIP-FLOP D

O flip-flop tipo D (delay) é mostrado na figura abaixo.



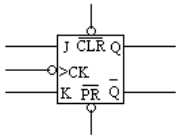
O estado deste flip-flop após um pulso de clock é igual à entrada D antes do pulso de clock. Como se vê na figura este flip-flop também varia na descida do clock. Se D varia apenas uma vez após cada pulso de clock a saída é como D, exceto pôr um delay.

A tabela verdade deste flip-flop é mostrada abaixo.

D Qa	Qf
0 0	0
0 1	0
1 0	1
1 1	1

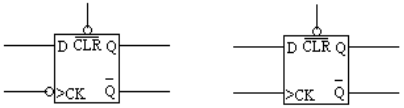
6.5. FLIP-FLOPS COM PRESET E CLEAR

Os flip-flops com sinal de clock geralmente apresentam duas entradas adicionais que podem ser usadas para estabelecer um estado inicial no flip-flop independente do clock. A figura abaixo mostra um flip-flop JK com preset (PRE) e clear (CLR).



Os pequenos círculos indicam que é necessário um valor lógico 0 (e não 1) para que seja estabelecido um 0 ou 1 na saída Q do flip-flop através do uso de clear e preset, respectivamente. Essas entradas suprimem as entradas clock, J e K. Sob condições normais, clear e preset não devem ser 0 simultaneamente . Se clear e preset forem 1 o circuito funciona normalmente.

A figura abaixo mostra dois flip-flops D com entrada clear.



No primeiro flip-flop ocorrerão variações nas saídas após a descida do pulso de clock e no segundo após a subida do clock. Nos dois casos a entrada clear força um valor Q = 0 se um valor zero for aplicado nesta.

6.6. EQUAÇÕES CARACTERÍSTICAS

A equação característica para um flip-flop pode ser derivada como segue:
Primeiro, faça uma tabela verdade que dá o estado futuro Qf como uma função do estado atual Qa e das entradas. Qualquer combinação ilegal deve ser tratada como estado irrelevante (don't care). Então desenhe o mapa para Qf e leia a equação característica do mapa. A equação dos flip-flops vistos até aqui são:

$$Qf = S + \overline{R}Qa \qquad \text{flip-flop RS}$$

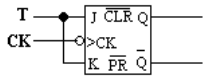
$$Qf = T \oplus Q = T\overline{Q} + \overline{T}Q \qquad \text{flip-flop T}$$

$$Qf = J\overline{Q} + \overline{K}Q \qquad \text{flip-flop JK}$$

$$Qf = D \qquad \text{flip-flop D}$$

Estas equações são válidas unicamente se as condições de entrada forem observadas.

Geralmente se pode obter um flip-flop de outro adicionando portas extras. Um flip-flop T, pôr exemplo, pode ser obtido de um flip-flop JK como na figura abaixo.



6.7. CIRCUITOS CONTADORES

Contam segundo uma determinada seqüência o número de pulsos (clock) que recebe na sua entrada.
São utilizados principalmente para contagens, geração de palavras, divisão de freqüência, medição de freqüência e tempo, geração de formas de onda e conversão de analógico para digital.

Contadores Assíncronos: o clock é aplicado no primeiro estágio. Os estágios seguintes utilizam como clock os estágios anteriores.

Contadores Síncronos: o clock é aplicado simultaneamente nos dois estágios.

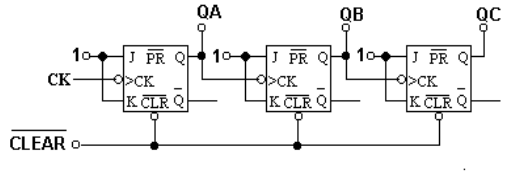
Contadores para cima (UP COUNTERS): contam numa seqüência crescente.
Contadores para baixo (DOWN COUNTERS): contam numa seqüência para baixo.

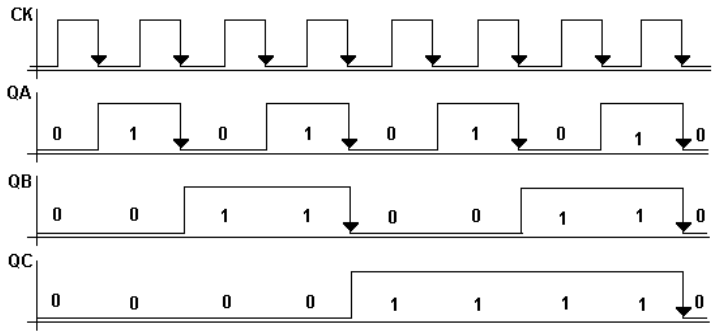
Módulo de um contador é o número de estados pelo qual o contador passa. Num contador assíncrono podemos fazer contadores de módulo diferente de 2^n usando as entradas PRESET e CLEAR dos flip-flops.

LOCKOUT = situação em que um contador pôr alguma razão entra num estado (número) fora da sua seqüência de contagem permanecendo numa o utra seqüência fora da original, não mais retornando a esta.

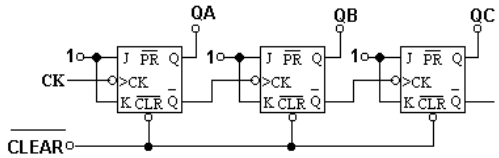
6.7.1. Contadores Assíncronos

UP COUNTER





DOWN COUNTER



6.7.2. Contadores Síncronos

Para estudarmos contadores síncronos usando flip -flops JK , pôr exemplo, devemos nos lembrar da sua tabela verdade, para que estes assumam o estado desejado.

J K	Qf	
0 0	Qa	(mantém o estado)
0 1	0	fixa 0
1 0	1	fixa 1
1 1	$\overline{Qa}$	inverte o estado

Podemos construir outra tabela a partir desta da seguinte forma:

Qa	Qf	J K
0	0	0 x
0	1	1 x
1	0	x 1
1	1	x 0

Onde x pode ser 0 ou 1. Nesta tabela vemos que se a saída estiver em zero e quisermos que o novo estado seja zero, podemos tanto manter o estado do flip -flop quanto fixar a saída em zero. Seguindo este raciocínio podemos obter o estado desejado na saída de cada flip -flop JK.

Gerador de Código BCD 8421

Para gerarmos este código necessitamos de quatro flip -flops JK Mestre-Escravo, um para cada bit do código. As saídas de cada um devem seguir a tabela verdade mostrada abaixo.

BCD 8421
Q_3 Q_2 Q_1 Q_0
0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0
0 1 0 1
0 1 1 0
0 1 1 1
1 0 0 0
1 0 0 1
1 0 1 0
1 0 1 1
1 1 0 0
1 1 0 1
1 1 1 0
1 1 1 1

Devemos então estudar as entradas JK em cada caso.

Se a condição inicial for

Q_3 Q_2 Q_1 Q_0
0 0 0 0

O estado seguinte ao primeiro pulso de clock deve ser

Q_3 Q_2 Q_1 Q_0
0 0 0 1

Portanto:

- Q_3 que estava em zero deve passar para zero, ou seja $J_3 = 0$ e $K_3 = x$.
- Q_2 que estava em zero deve passar para zero, ou seja $J_2 = 0$ e $K_2 = x$.
- Q_1 que estava em zero deve passar para ze ro, ou seja $J_1 = 0$ e $K_1 = x$.
- Q_0 que estava em zero deve passar para um, ou seja $J_0 = 1$ e $K_0 = x$.

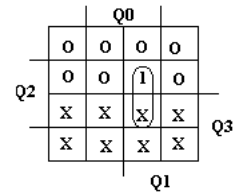
Fazendo isto para todos os estados:

	BCD 8421	
Pulso de clock	$Q_3\ Q_2\ Q_1\ Q_0$	$J_3\ K_3\ J_2\ K_2\ J_1\ K_1\ J_0\ K_0$
1º	0 0 0 0	0 x 0 x 0 x 1 x
2º	0 0 0 1	0 x 0 x 1 x x 1
3º	0 0 1 0	0 x 0 x x 0 1 x
4º	0 0 1 1	0 x 1 x x 1 x 1
5º	0 1 0 0	0 x x 0 0 x 1 x
6º	0 1 0 1	0 x x 0 1 x x 1
7º	0 1 1 0	0 x x 0 x 0 1 x
8º	0 1 1 1	1 x x 1 x 1 x 1
9º	1 0 0 0	x 0 0 x 0 x 1 x
10º	1 0 0 1	x 0 0 x 1 x x 1
11º	1 0 1 0	x 0 0 x x 0 1 x
12º	1 0 1 1	x 0 1 x x 1 x 1
13º	1 1 0 0	x 0 x 0 0 x 1 x
14º	1 1 0 1	x 0 x 0 1 x x 1
15º	1 1 1 0	x 0 x 0 x 0 1 x
16º	1 1 1 1	x 1 x 1 x 1 x 1

O contador reiniciará a contagem após chegar ao estado 15.

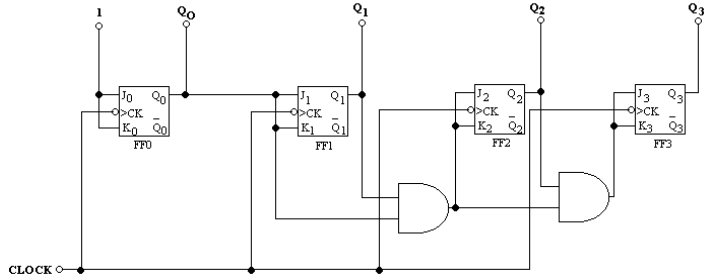
Para obter a expressão de J_3, K_3, J_2, K_2, J_1, K_1, J_0 e K_0 usando mapas de Karnaugh:

J_3



$J_3 = Q_2Q_1Q_0$

Fazendo para os outros obtemos:
$K_3 = Q_2Q_1Q_0$
$J_2 = Q_1Q_0$
$K_2 = Q_1Q_0$
$J_1 = Q_0$
$K_1 = Q_0$
$J_0 = 1$
$K_0 = 1$
O circuito será então:



Para fazer o mesmo contador usando flip -flops D é necessário um flip -flop para cada variável $Q_3Q_2Q_1Q_0$. Suponhamos que o estado inicial de cada um seja 0. Quando um pulso é recebido, a contagem deve mudar para 0001; quando o segundo pulso chegar, deve mudar para 0010; assim pôr diante. E quando a contagem chegar a 1111 o contador deve ser receber um reset e ir para 0000.

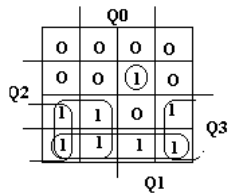
O problema então é determinar as entradas dos flip -flops T para que a contagem aconteça nesta seqüência.

	BCD 8421	
Pulso de clock	$Q_3\ Q_2\ Q_1\ Q_0$	$D_3\ D_2\ D_1\ D_0$
1º	0 0 0 0	0 0 0 1
2º	0 0 0 1	0 0 1 0
3º	0 0 1 0	0 0 1 1
4º	0 0 1 1	0 1 0 0
5º	0 1 0 0	0 1 0 1
6º	0 1 0 1	0 1 1 0
7º	0 1 1 0	0 1 1 1
8º	0 1 1 1	1 0 0 0
9º	1 0 0 0	1 0 0 1
10º	1 0 0 1	1 0 1 0
11º	1 0 1 0	1 0 1 1
12º	1 0 1 1	1 1 0 0
13º	1 1 0 0	1 1 0 1
14º	1 1 0 1	1 1 1 0
15º	1 1 1 0	1 1 1 1
16º	1 1 1 1	0 0 0 0

Note na tabela que se o estado futuro em um flip -flop é 0 a entrada D deste flip-flop deve ser 0 antes da chegada de um pulso e se a saída tiver que ser 1 a entrada deste flip -flop deve ser 1.

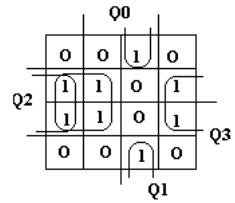
Podemos então derivar os valores de D_3, D_2, D_1 e D_0 usando mapas de Karnaugh.

D_3



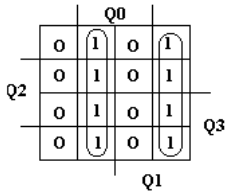
$$D_3 = \overline{Q_3}Q_2Q_1Q_0 + Q_3\overline{Q_2} + Q_3\overline{Q_0} + Q_3\overline{Q_1}$$

D_2



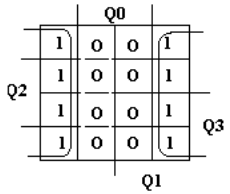
$$D_2 = \overline{Q_2}Q_1Q_0 + Q_2\overline{Q_0} + Q_2\overline{Q_1}$$

D_1



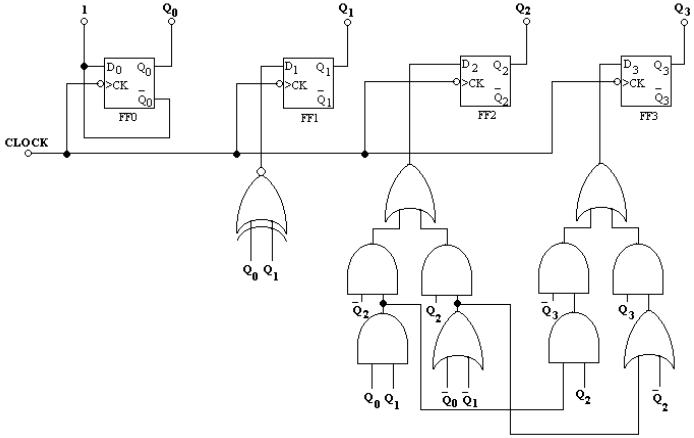
$$D_1 = \overline{Q_1}Q_0 + Q_1\overline{Q_0}$$

D_0



$D_0 = \overline{Q_0}$

O circuito é apresentado na figura abaixo.

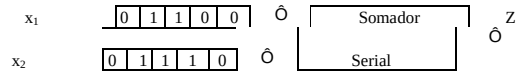


Pode-se usar os outros flip-flops para implementar o contador acima. Pode-se além disto fazer outros contadores bastando para isto escrever a tabela verdade dos estados presentes e futuros na seqüência e adequar as entradas dos flip -flops para que gerem as saídas necessárias.

6.8. MÁQUINAS DE ESTADO

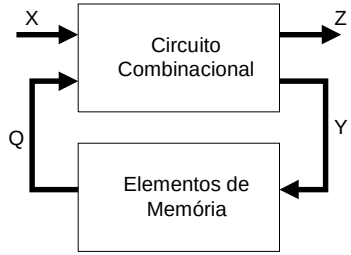
Máquina seqüencial ou máquina de estados finitos, é o modelo abstrato usado para representar o circuito seqüencial real. Seu comportamento é descrito como uma seqüência de eventos que ocorrem em instantes de tempo discretos.

Suponha que uma máquina M receba alguns sinais de entrada e responda produzindo alguns sinais de saída. Se no instante t for aplicado um sinal x(t) a M sua resposta depende de x(t) bem como das entradas nos instantes anteriores. Trataremos aqui apenas de máquinas cujo passado pode ser resumido em um conj unto finito de variáveis. Pôr exemplo, no somador binário serial da figura abaixo a resposta aos sinais de entrada em t depende apenas destes sinais e do valor do transporte gerado no instante anterior. Assim, embora o somador possa ter um número infinito de passados, eles podem ser agrupados em duas classes, aquelas que resultam num transporte 1 e as que resultam num transporte 0 num instante t.



As máquinas estudadas aqui são as que podem dis tinguir entre um número finito de classes de entradas passadas e estas classes são chamadas estados internos destas máquinas.

Uma máquina seqüencial pode ser representada esquematicamente pelo circuito da figura abaixo.



As entradas do circuito formam o conjunto X com r variáveis. O circuito tem m saídas que compõem o conjunto Z. O valor contido em cada elemento de memória é chamado variável de estado, e o conjunto Q constitui o conjunto de k variáveis de estado. As m saídas Z e as s funções de transição interna Y são funções das entradas e dos estados internos da máquina. Os valores em Y no instante t definem as variáveis de estado no instante t+1.

As relações entre entradas, estados presentes, saídas e estados futuros pode m ser descritas pôr um diagrama de estados ou pôr uma tabela de estados, ou ainda pôr uma tabela de transição.

Uma tabela de estados tem $p = 2^r$ colunas, uma para cada ocorrência do vetor de entrada, e $n = 2^k$ linhas, uma para cada estado. Cada combinação de entradas e estados presentes determina a saída produzida e o próximo estado para a máquina.

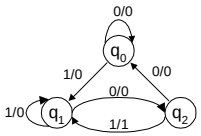
O diagrama de estados é um grafo orientado, onde cada estado da máquina corresponde a um nó. De cada nó emanam p arcos orientados, correspondendo às transições de estados causadas pela ocorrência da entrada. Cada arco orientado é rotulado com a entrada que determina aquela transição e com a saída gerada.

As máquinas que determinam o próximo estado Q(t+1) com base apenas em Q(t) e na entrada X(t) são chamadas máquinas determinísticas ou markovianas. Nas máquinas determinísticas Q(t+1) = f[Q(t),X(t)] onde f é a função de transição de estados. O valor da saída é função do estado presente e às vezes das entradas presentes. Uma máquina determinística cuja saída depende apenas do estado interno é chamada máquina de Moore. Uma máquina de Mealy é uma máquina determinística cuja saída depende do estado interno e das entradas presentes.

Exemplo:
Projete um circuito cuja saída Z seja um se a seqüência de pulsos numa cadeia de símbolos 0 e 1 for 101 a partir do estado inicial q0.

Expressando o circuito como uma máquina de Mealy temos:

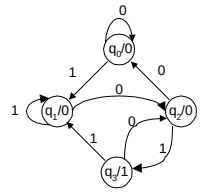
q(t)	x	q(t+1)	Z(t)
q0	0	q0	0
q0	1	q1	0
q1	0	q2	0
q1	1	q1	0
q2	0	q0	0
q2	1	q1	1



O estado q0 representa o estado inicial, e se um 0 for encontrado o estado não se alterará e a saída será 0; um 1 leva ao estado q1 que indica que o primeiro 1 na seqüência desejada já foi encontrado, mas a saída produzida é ainda 0. No estado q1, um 1 não provocará mudança de estado, mas um 0 levará ao estado q2, que indica que o 0 depois do primeiro 1 na seqüência já foi encontrado e a saída será 0. Se o próximo número na seqüência for um 1 o sistema terá uma saída 1 e o próximo estado será q1, pois uma nova seqüência 101 pode estar começando. Se no estado q2 a seqüência apresentar um 0, o circuito dará saída 0 e retornará para o estado q0, indicando que nenhuma parte da seqüência desejada foi ainda encontrada.

Expressando como máquina de Moore temos:

q(t)	x	q(t+1)	Z(t)
q0	0	q0	0
q0	1	q1	0
q1	0	q2	0
q1	1	q1	0
q2	0	q0	0
q2	1	q3	0
q3	0	q2	1
q3	1	q1	1



Neste caso o estado q0 é o estado inicial, o estado q1 indica que o primeiro 1 na seqüência desejada foi encontrado, o estado q2 indica que o 0 depois do primeiro 1 já foi encontrado e o estado q3 indica que a seqüência inteira foi encontrada. A saída é mostrada nos nós do grafo.

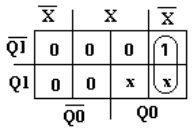
Pode-se usar flip-flops para implementar máquinas de estado associando os estados dos flip-flops aos estados da máquina pôr um processo chamado alocação de estados.

Uma tabela de transição de estados especifica o próximo estado dos elementos de memória para cada combinação de entrada e variáveis de estado. Tomando-se a máquina de Mealy do exemplo anterior, com a seguinte alocação de estados: q0 = 00, q1 = 01 e q2 = 10, temos:

$Q_1Q_0(t)$	X	$Q_1Q_0(t+1)$	Z	D_1D_0
00	0	00	0	00
00	1	01	0	01
01	0	10	0	10
01	1	01	0	01
10	0	00	0	00
10	1	01	1	01
11	0	xx	x	xx
11	1	xx	x	xx

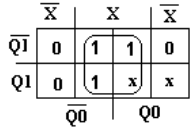
Para gerar os estados futuros como especificado, os elementos de memória devem ser supridos com as entradas necessárias. Assim, para implementar o circuito usando flip-flops tipo D deve-se ter:

D_1



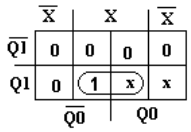
$D_1 = \overline{X}Q_0$

D_0

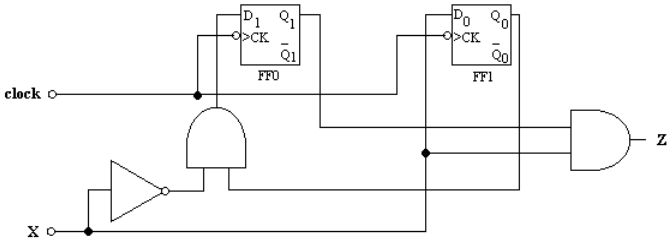


$D_0 = X$

Z



$Z = XQ_1$

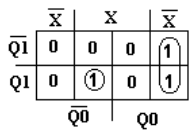


Agora vejamos a máquina de Moore apresentada anteriormente.

q(t)	x	q(t+1)	Z(t)
q0	0	q0	0
q0	1	q1	0
q1	0	q2	0
q1	1	q1	0
q2	0	q0	0
q2	1	q3	0
q3	0	q2	1
q3	1	q1	1

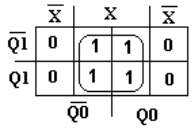
$Q_1Q_0(t)$	X	$Q_1Q_0(t+1)$	Z	D_1D_0
00	0	00	0	00
00	1	01	0	01
01	0	10	0	10
01	1	01	0	01
10	0	00	0	00
10	1	11	1	11
11	0	10	0	10
11	1	01	0	01

D_1



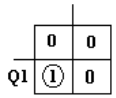
$D_1 = X\overline{Q_0}Q_1 + \overline{X}Q_0$

D_0



$D_0 = X$

Z



$Z = Q_1Q_0$