Eletrônica Digital Básica

Jurandyr S. Nogueira

EDUFBA

Eletrônica Digital Básica

Eletrônica Digital Básica

Jurandyr S. Nogueira

Salvador
EDUFBA
2011

CAPA E ARTE FINAL
Amanda Santana da Silva

NORMALIZAÇÃO
Adriana Caxiado

Sistema de Biblioteca da UFBA

Nogueira, Jurandyr Santos.
Eletrônica digital básica / Jurandyr Santos Nogueira. - Salvador: EDUFBA, 2011.
170 p.

ISBN 978-85-232-0836-3

1. Eletrônica digital. I. Título.

CDD - 621.3815

Editora filiada à

Associação Brasileira das Editoras Universitárias

ASOCIACION DE EDITORIALES UNIVERSITARIAS DE AMERICA LATINA Y EL CARIBE

CBaL Câmara Bahiana do Livro

EDUFBA

Rua Barão de Jeremoabo, s/n, Campus de Ondina,

40170-115 Salvador-BA Brasil

Tel/fax: (71)3283-6160/3283-6164
www.edufba.ufba.br | edufba@ufba.br

SUMÁRIO

1. INTRODUÇÃO AOS SISTEMAS NUMÉRICOS

Todo sistema numérico é constituído por um conjunto ordenado de símbolos ou dígitos, com regras definidas para as operações de adição, subtração, multiplicação, divisão e outras operações matemáticas. O número de símbolos pertencentes a um dado sistema é denominado **base** ou **raiz**.
Um sistema numérico permite representar uma grandeza qualquer através de símbolos pertencentes ao sistema, podendo tal representação constar de uma parte inteira e outra fracionária, as quais são separadas, entre si, por uma vírgula.
Um número escrito numa base qualquer (**r**) pode ser expresso por sua forma polinomial:

$$(N)_r = \sum_{j=-m}^{n-1} a_j \, r^j$$

ou

$$(N)_r = a_{n-1}\, r^{n-1} + a_{n-2}\, r^{n-2} + \ldots + a_1\, r^1 + a_0\, r^0 + \ldots \Rightarrow \{\text{Parte Inteira}\}$$

$$+ a_{-1}\, r^{-1} + a_{-2}\, r^{-2} + \ldots + a_{-m}\, r^{-m} \Rightarrow \{\text{Parte Fracionária}\}$$

onde:

- **r** ⇒ (base ou raiz do sistema);
- **a** ⇒ (dígito do sistema);
- **n** ⇒ (número de dígitos da parte inteira);
- **m** ⇒ (número de dígitos da parte fracionária);
- a_{n-1} ⇒ (dígito mais significativo da parte inteira);
- a_0 ⇒ (dígito menos significativo da parte inteira);
- a_{-1} ⇒ (dígito mais significativo da parte fracionária);
- a_{-m} ⇒ (dígito menos significativo da parte fracionária).

O Sistema Decimal é o mais utilizado nas operações matemáticas e aplicações gerais da engenharia, mas alguns outros Sistemas Numéricos são de grande importância, sobretudo no estudo dos circuitos lógicos, sistemas digitais e computadores, a exemplo dos Sistemas: Binário, Octal e Hexadecimal, os quais são assim denominados em virtude de apresentarem, respectivamente, as bases dois, oito e dezesseis.

Assim, os mencionados Sistemas Numéricos possuem os dígitos abaixo discriminados:

- **decimal: {0,1,2,3,4,5,6,7,8,9};**
- **binário: {0,1};**
- **octal: {0,1,2,3,4,5,6,7} e**
- **hexadecimal:{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}.**

Qualquer dígito do Sistema Binário é comumente denominado ***bit***, pela contração das palavras ***binary digit***, do idioma inglês. O Sistema Hexadecimal (**Hex**), é considerado como um sistema alfanumérico, por ser constituído por letras do alfabeto e algarismos. As letras **A**, **B**, **C**, **D**, **E** e **F** do Hexadecimal correspondem aos decimais: **10**, **11**, **12**, **13**, **14** e **15**, respectivamente.
A representação polinomial de um número, expresso nesses sistemas de interesse, pode ser exemplificada conforme a indicação abaixo:

$$(N)_{10} = (94,72)_{10} = 9 \times 10^{1} + 4 \times 10^{0} + 7 \times 10^{-1} + 2 \times 10^{-2}$$

$$(N)_{2} = (1011,11)_{2} = 1 \times 2^{3} + 0 \times 2^{2} + 1 \times 2^{1} + 1 \times 2^{0} + 1 \times 2^{-1} + 1 \times 2^{-2}$$

$$(N)_{8} = (1374,25)_{8} = 1 \times 8^{3} + 3 \times 8^{2} + 7 \times 8^{1} + 4 \times 8^{0} + 2 \times 8^{-1} + 5 \times 8^{-2}$$

$$(N)_{16} = (F8E,3D)_{16} = F \times 16^{2} + 8 \times 16^{1} + E \times 16^{0} + 3 \times 16^{-1} + D \times 16^{-2}$$

1.1 CONVERSÕES ENTRE SISTEMAS NUMÉRICOS

Um número expresso num dado sistema numérico tem o seu equivalente em qualquer outro sistema. Assim, um número decimal, mediante a aplicação de algoritmos adequados, poderá ter o seu equivalente binário, octal, hexadecimal ou em qualquer outra base. Do mesmo modo, um número escrito, por exemplo num sistema cuja base é igual a 3 (ternário), também terá os seus equivalentes em binário, decimal, octal, hexadecimal ou em qualquer outra base ou raiz.
A obtenção de um número equivalente a outro numa base distinta, denomina-se conversão numérica. Essencialmente, três casos devem ser considerados:

a) conversão de um número numa base qualquer para decimal: $(N)_r \Rightarrow (N)_{10}$;

b) conversão de um número decimal para uma base qualquer: $(N)_{10} \Rightarrow (N)_r$;

c) conversão de um número numa base r_1 para a base r_2: $(N)_{r_1} \Rightarrow (N)_{r_2}$.

1.1.1 Conversão de um número numa Base qualquer para Decimal: $(N)_r \Rightarrow (N)_{10}$

Quando se deseja encontrar o decimal equivalente a um número expresso em qualquer outro sistema, representa-se o número em questão sob a sua forma polinomial e, em seguida, encontra-se o decimal correspondente a cada uma das parcelas. A soma resultante das diversas parcelas corresponderá ao número decimal equivalente. Por exemplo: se uma determinada grandeza se encontra representada pelo número binário $(N)_2 = (1101,1001)_2$, a sua expressão polinomial será:

$$(N)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} .$$

Encontrando-se o decimal equivalente a cada uma das parcelas, tem-se:

$$8 + 4 + 0 + 1 + 0,5 + 0 + 0 + 0,0625 .$$

Efetuando-se a soma das parcelas acima, encontra-se o número $(13,5625)_{10}$ como resultado, o qual corresponderá ao decimal equivalente ao número binário $(1101,1001)_2$.

Ou seja:

$$\mathbf{(13,5625)_{10} \cong (1101,1001)_2}$$

Do mesmo modo, se o hexadecimal $\mathbf{(N)_{16} = (F8E,3D)_{16}}$ tem por representação polinomial:

$$\mathbf{(N)_{16} = F \times (16)^2 + 8 \times (16)^1 + E \times (16)^0 + 3 \times (16)^{-1} + D \times (16)^{-2} ,}$$

então, sabendo-se que as letras **A**, **B**, **C**, **D**, **E**, e **F** do Sistema Hexadecimal correspondem respectivamente aos números **10**, **11**, **12**, **13**, **14**, e **15** do Sistema Decimal, como dito acima, a sua representação polinomial será:

$$15 \times (16)^2 + 8 \times (16)^1 + 14 \times (16)^0 + 3 \times (16)^{-1} + 13 \times (16)^{-2}. \text{ ou}$$

$$3840 + 128 + 14 + 0,1875 + 0,0508 ,$$

cuja soma resulta em $(3982,2383)_{10}$.

Logo,

$$\mathbf{(3982,2383)_{10} \cong (F8E,3D)_{16}}$$

Do mesmo modo, se o número $\mathbf{(N)_7}$ = $\mathbf{(2146{,}523)_7}$ tem por representação polinomial:

$$2 \times 7^3 + 1 \times 7^2 + 4 \times 7^1 + 6 \times 7^0 + 5 \times 7^{-1} + 2 \times 7^{-2} + 3 \times 7^{-3} ,$$

e, sendo os decimais equivalentes a cada uma das parcelas:

$$686 + 49 + 28 + 6 + 0{,}7143 + 0{,}0408 + 0{,}0087 ,$$

a soma dessas parcelas é igual a $(769{,}7638)_{10}$.

Portanto,

$$\mathbf{(769{,}7638)_{10} \cong (2146{,}523)_7.}$$

O procedimento acima, envolvendo o desenvolvimento polinomial (**DP**) e o somatório das diversas parcelas, convertidas ao Sistema Decimal, deverá ser aplicado cada vez que um número expresso numa **base r** qualquer deva ser convertido à **base 10**.
Esquematicamente:

$$\mathbf{(N)_r \Rightarrow \{ \Sigma(D\ P) \} \Rightarrow (N)_{10} .}$$

Observação:
No caso específico de se converter um número binário, inteiro, em seu equivalente decimal, existe um algoritmo muito simples, conhecido como o **método de dobrar** *e* **somar**, que permite a realização de tal conversão praticamente por inspeção, sem se utilizar, explicitamente, o desenvolvimento polinomial, o qual muitas vezes se torna enfadonho, quando da sua aplicação sistemática.

O método em questão consiste em dobrar o **bit** mais significativo, e somá-lo com o imediatamente menos significativo, dobrando-se novamente o resultado, repetindo-se o processo até o **bit** menos significativo da parte inteira. O resultado, ao final, será o decimal equivalente.

Exemplo:
Encontrar o decimal equivalente ao número binário:

$$\mathbf{(N)_2 = 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1}$$

Solução:

dobra-se o **MSB**	=	1	X	2	=	2	$\Rightarrow$	+	bit seguinte:	2	+	1	=	3
dobra-se o resultado	=	3	X	2	=	6	$\Rightarrow$	+	bit seguinte:	6	+	0	=	6
dobra-se o resultado	=	6	X	2	=	12	$\Rightarrow$	+	bit seguinte:	12	+	1	=	13
dobra-se o resultado	=	13	X	2	=	26	$\Rightarrow$	+	bit seguinte:	26	+	0	=	26
dobra-se o resultado	=	26	X	2	=	52	$\Rightarrow$	+	**LSB**	52	+	1	=	53

Decimal equivalente: $(\mathbf{53})_{\mathbf{10}}$.

Isto é:

$$\mathbf{(1\ 1\ 0\ 1\ 0\ 1)_2 \cong (53)_{10}}$$

Outro exemplo:

Encontrar o decimal equivalente ao número binário $(N)_2$ = 1 0 0 1 1 1 0 1 0.

Solução:

dobra-se o **MSB**:	1	x	2	=	2	$\Rightarrow$	+	bit seguinte:	2	+	0	=	2
dobra-se o resultado	2	x	2	=	4	$\Rightarrow$	+	bit seguinte:	4	+	0	=	4
dobra-se o resultado	4	x	2	=	8	$\Rightarrow$	+	bit seguinte:	8	+	1	=	9
dobra-se o resultado	9	x	2	=	18	$\Rightarrow$	+	bit seguinte:	18	+	1	=	19
dobra-se o resultado	19	x	2	=	38	$\Rightarrow$	+	bit seguinte:	38	+	1	=	39
dobra-se o resultado	39	x	2	=	78	$\Rightarrow$	+	bit seguinte:	78	+	0	=	78
dobra-se o resultado	78	x	2	=	156	$\Rightarrow$	+	bit seguinte:	156	+	1	=	157
dobra-se o resultado	157	x	2	=	314	$\Rightarrow$	+	**LSB:**	314	+	0	=	**314**

Decimal equivalente: $(\mathbf{314})_{\mathbf{10}}$.

Logo,

$$\mathbf{(1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0)_2 \cong (314)_{10}}$$

Observa-se que a aplicação de algoritmo similar à parte fracionária, bem como aos números em Octal ou Hexadecimal, não reduz suficientemente o trabalho de se efetuar a conversão pelos métodos anteriormente descritos, não se justificando, portanto, a sua utilização generalizada!

1.1.2 Conversão de um Número Decimal para uma Base qualquer: $(N)_{10} \Rightarrow (N)_r$

Freqüentemente o número conhecido encontra-se em decimal $\mathbf{(N)_{10}}$ e deseja-se obter o seu equivalente numa base **r** qualquer $\mathbf{(N)_r}$. Neste caso, o método anterior não é utilizado diretamente, sendo aplicados distintos algoritmos tanto

para a parte inteira quanto para a fracionária do decimal conhecido, procedendo-se do seguinte modo:

a) separa-se a parte inteira da fracionária, caso esta última exista;

b) toma-se a parte inteira do número decimal e divide-se pela base (**r**) do sistema no qual se deseja encontrar o número equivalente. O resultado apresentará um quociente (**Q**) e um resto (**R**). Toma-se o primeiro quociente encontrado e divide-se novamente pela base (**r**). O resultado apresentará um segundo quociente e um segundo resto. Aplica-se sucessivamente tal processo até que o quociente resultante seja nulo, indicando-se também o último resto obtido. Os restos obtidos ao longo desse procedimento se constituirão nos dígitos da parte inteira do número na base **r** , sendo o primeiro deles o dígito menos significativo (**LSB**) e o último, o mais significativo (**MSB**).

c) toma-se a parte fracionária do número decimal (**F**), caso a mesma exista, e multiplica-se pela base (**r**) do sistema no qual se deseja encontrar o número equivalente. O resultado apresentará um produto (**P**) que estará constituído por uma parte inteira (**I**), eventualmente nula, e outra fracionária (**f**). Toma-se a primeira parte fracionária encontrada e multiplica-se novamente pela base (**r**). O resultado apresentará um segundo produto, também constituído por uma segunda parte inteira, eventualmente nula, e outra fracionária.
Repete-se o procedimento, sucessivamente, até o número de dígitos atender à aproximação necessária para representar com precisão o equivalente ao decimal a ser convertido, ou até a parte fracionária resultar nula, ou se configurar como em sendo uma dízima periódica!

Os inteiros obtidos ao longo desse procedimento se constituirão nos dígitos da parte fracionária do número na base (**r**) , sendo o primeiro deles o dígito mais significativo (**MSB**) e o último, o menos significativo (**LSB**)!

Após tal procedimento, a soma do inteiro com o fracionário equivalentes, resulta no número representado na base (**r**).

Exemplo: Converter o decimal $(N)_{10}$ = 100,39 para o sistema binário.

Solução:

a) Separando-se a parte inteira da fracionária, temos que o número decimal acima $(N)_{10} = 100,39$ pode ser escrito como:

$$(N)_{10} = (100)_{10} + (0,39)_{10}$$

b) Tomando-se a parte inteira $(100)_{10}$ e aplicando-se o procedimento *b*, indicado anteriormente, tem-se:

DEC	÷	Base	=	Quociente	+	Resto		Status
100	÷	2	=	50	+	0	⇒	**[LSB]**
50	÷	2	=	25	+	0		
25	÷	2	=	12	+	1		
12	÷	2	=	6	+	0		
6	÷	2	=	3	+	0		
3	÷	2	=	1	+	1		
1	÷	2	=	0	+	1	⇒	**[MSB]**

Assim, o número binário equivalente ao inteiro $(100)_{10}$ será constituído pelos dígitos binários: **1 1 0 0 1 0 0**.

Ou seja: $(\ 1100100\ )_2 \cong (\ 100\ )_{10}$

c) Tomando-se agora a parte fracionária $(0,39)_{10}$ e aplicando-se o procedimento *c*, indicado acima, tem-se:

DEC	x	Base	=	Produto	=	Fração	+	Inteira		Status
0,39	x	2	=	0,78	=	0,78	+	0	⇒	**[MSB]**
0,78	x	2	=	1,56	=	0,56	+	0		
0,56	x	2	=	1,12	=	0,12	+	1		
0,12	x	2	=	0,24	=	0,24	+	1		
0,24	x	2	=	0,48	=	0,48	+	0		
0,48	x	2	=	1,92	=	0,96	+	0		
0,96	x	2	=	1,92	=	0,92	+	0		
0,92	x	2	=	1,84	=	0,84	+	1		
0,84	x	2	=	1,68	=	0,68	+	1		
0,68	x	2	=	1,36	=	0,36	+	1	⇒	**[LSB]**

Assim, a parte fracionária será constituída pelos dígitos binários:

0 1 1 0 0 0 1 1 1 1

Observa-se que a fração resultante igual a $\mathbf{(0{,}0110001111)_2}$, corresponde na verdade ao decimal $\mathbf{(0{,}3896)_{10}}$ - o que pode ser verificado através do seu desenvolvimento polinomial; ou seja, uma aproximação do número $\mathbf{(0{,}39)_{10}}$.
O erro na conversão é inversamente proporcional ao número de *bits* da parte fracionária.

Concluída a conversão da parte fracionária, tem-se então que:

$$\mathbf{(100{,}39)_{10} \cong (1100100{,}0110001111)_2}$$

Evidentemente, a verificação desse resultado pode ser realizada pela conversão do binário encontrado ao decimal correspondente, através do método do desenvolvimento polinomial, conforme citado acima, considerando as aproximações pertinentes.

Vale ressaltar que o resultado da conversão será exato somente quando a parte fracionária resultar nula.

Por exemplo: Converter o decimal 0,5 para a base 2.

Aplicando-se o mesmo procedimento, anteriormente descrito, para a parte fracionária, uma vez que neste caso a parte inteira é nula, tem-se:

DEC	**x**	**Base**	**=**	**Produto**	**=**	**Fração**	**+**	**Inteira**	**Status**
0,5	x	2	=	1,0	=	0,0	+	**1**	⇒**[MSB]**
0,0	x	2	=	0,0	=	0,0	+	**0**	⇒**[LSB]**

O binário equivalente é constituído pelos bits **1** e **0**.
Isto é:

$$\mathbf{(0{,}5)_{10} \cong (0{,}10)_{2.}}$$

A conversão de um número na base dez, no seu equivalente em qualquer outra base pode ser encontrada utilizando-se o método acima exposto. Este algoritmo se aplica a qualquer caso, lembrando-se apenas que, quando o número é composto, as partes inteira e fracionária recebem tratamentos distintos e duais.

Esquematicamente, para efeito mnemônico,

$$(N)_{10} \Rightarrow \left\{ \begin{array}{c} \{ I \div r = Q + [R]^{LSB}_{MSB} \} \\ + \\ \{ F \times r = f + [I]^{MSB}_{LSB} \} \end{array} \right\} \Rightarrow (N)_r$$

1.1.3 Conversão de um Número numa Base r_1 para a Base r_2: $(N)r_1 \Rightarrow (N)r_2$

Genericamente, quando se deseja converter um número expresso num sistema cuja base ou raiz é definida por $\mathbf{r_1}$ em outro sistema cuja base é definida por $\mathbf{r_2}$, procede-se da seguinte maneira:

Encontra-se o equivalente decimal do número expresso em um dos sistemas através do seu desenvolvimento polinomial, como indicado no item I.1.1.
Após a identificação do decimal correspondente, encontra-se o equivalente ao número expresso na outra base, através do método indicado no item I.1.2.

Esquematicamente:

$$(N)r_1 \Rightarrow\Rightarrow (N)_{10} \Rightarrow\Rightarrow (N)r_2$$

Exemplo: Encontrar o equivalente ao número $(123,45)_6$, num sistema de base 3.

Solução: O equivalente decimal de $(123,45)_6$ é, como já vimos pelo método do desenvolvimento polinomial, encontrado por:

$$1 \times 6^2 + 2 \times 6^1 + 3 \times 6^0 + 4 \times 6^{-1} + 5 \times 6^{-2} \quad \text{ou,}$$

$$1 \times 36 + 2 \times 6 + 3 \times 1 + 4 \times 0{,}1666 + 5 \times 0{,}0277 \quad \text{ou ainda,}$$

$$36 + 12 + 3 + 0{,}6664 + 0{,}1385 = (51{,}8049)_{10}.$$

Então:

$$(123,45)_6 \cong (51,8049)_{10}$$

Para se encontrar o equivalente de $(51,8049)_{10}$ na base 3, aplica-se o algoritmo indicado no item I.1.2, separando-se o procedimento relativo à parte inteira do da fracionária, conforme visto anteriormente:

Parte inteira :

DEC	**÷**	**Base**	**=**	**Quociente**	**+**	**Resto**		**Status**
51	÷	3	=	17	+	**0**	⇒	**[LSB]**
17	÷	3	=	5	+	**2**		
5	÷	3	=	1	+	**2**		
1	÷	3	=	0	+	**1**	⇒	**[MSB]**

ou,

$$\mathbf{(51)_{10} \cong (1220)_3}$$

Parte Fracionária :

DEC.	**x**	**Base**	**=**	**Produto**	**=**	**Fração**	**+**	**Inteira**		**Status**
0,8049	x	3	=	2,4147	=	0,4147	+	**2**	⇒	**[MSB]**
0,4147	x	3	=	1,2441	=	0,2441	+	**1**		
0,2441	x	3	=	0,7323	=	0,7323	+	**0**		
0,7323	x	3	=	2,1969	=	0,1969	+	**2**		
0,1969	x	3	=	0,5907	=	0,5907	+	**0**		
0,5907	x	3	=	1,7721	=	0,7721	+	**1**		
0,7721	x	3	=	2,3163	=	0,3163	+	**2**	⇒	**[LSB]**

ou,

$$\mathbf{(0,8049)_{10} \cong (0,2102012...)_3}$$

Logo,

$$\mathbf{(123,45)_6 \cong (51,8049)_{10} \cong (1220,2102012...)_3}$$

1.1.4 Conversões entre Binário, Octal e Hexadecimal

A conversão entre os sistemas binário, octal e hexadecimal, pode ser realizada a partir do método geral indicado no item I.1.3. Contudo, devido a determinadas peculiaridades desses sistemas, a conversão entre os mesmos torna-se muito simples, desde que se considere o seguinte: Os dígitos do sistema octal podem ser representados por grupos de 3 *bits* do sistema binário, como a seguir:

OCTAL	BINÁRIO
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Do mesmo modo, os dígitos do sistema hexadecimal podem ser representados por grupos de 4 bits do sistema binário:

HEXADECIMAL	BINÁRIO
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

Pela razão acima citada, qualquer número binário pode ser convertido diretamente em octal ou hexadecimal, bastando para isso separar-se no primeiro caso (**octal**) grupos de 3 *bits* e, no segundo (**hexadecimal**), grupos de 4 *bits* do binário em questão. A partir daí, identifica-se a representação equivalente, encontrando-se, por inspeção, o resultado num ou noutro sistema.

Quando o grupo de dígitos não pode ser subdividido em subgrupos de 3 ou 4 *bits*, acrescentam-se zeros à esquerda da parte inteira ou à direita da parte fracionária.

Exemplo: Encontrar o octal e o hexadecimal equivalentes ao número

$$(N)_2 = 111101000{,}0111$$

Solução: Separando-se a informação em grupos de 3 *bits,*

$$(N)_2 = 111\ \ 101\ \ 000,\ \ 011\ \ 100 \text{ ou,}$$

$$(N)_8 = 7\ \ 5\ \ 0\ ,\ \ 3\ \ 4$$

Do mesmo modo, no caso do hexadecimal, separando-se em grupos de 4 *bits,*

$$(N)_2 = 0001\ \ 1110\ \ 1000,\ \ 0111 \text{ tem-se:}$$

$$(N)_{16} = 1\ \ E\ \ 8\ ,\ \ 7.$$

Logo,

$$(111101000{,}0111)_2 \cong (750{,}34)_8 \cong (1E8{,}7)_{16}$$

Observa-se então que, o número de dígitos necessários para se representar uma determinada grandeza varia de acordo com a base do sistema. No exemplo acima, a grandeza é representada em binário por 13 dígitos, em octal por 5 dígitos e por apenas 4 dígitos em hexadecimal. Essa é uma das vantagens de se utilizar os sistemas octal e hexadecimal!

Evidentemente, o processo de conversão de octal ou hexadecimal para binário também é possível, bastando aplicar o processo anterior de modo inverso.

Exemplo:

Encontrar o Bnário equivalente ao Octal $(N)_8$ = 274,71

Como $(N)_8$ = 2 7 4 , 7 1

tem-se $(N)_2$ = 010 111 100 , 111 001.

Do mesmo modo, o hexadecimal $(N)_{16}$ = FAE,BC terá como equivalente o binário:

$(N)_{16}$ = F A E , B C ou

$(N)_2$ = 1111 1010 1110, 1011 1100

Observações*:*

Quando se deseja encontrar os equivalentes numéricos em binário, octal e hexadecimal de um determinado número explicitado em decimal, deve-se iniciar o processo de conversão a partir do hexadecimal, e não do binário como é muito comum se proceder. A razão dessa recomendação nasce do fato de as operações de conversão para a base 16 convergir mais rapidamente para o resultado. Após a determinação do hexadecimal, deve-se encontrar, por inspeção, o seu equivalente binário e, através desse mesmo processo, a partir do binário, encontrar-se o octal equivalente.

Devido à sua grande utilização, vale dizer que os binários, quando agrupados em determinado número de dígitos, constituíndo-se numa informação completa, recebe designações especiais a saber:

- ***BIT*** $\Rightarrow$ **um dígito binário 0 ou 1**
- ***NIBBLE*** $\Rightarrow$ **um grupo de 4 *bits***
- ***BYTE*** $\Rightarrow$ **um grupo de 8 *bits***
- ***WORD*** $\Rightarrow$ **um grupo de 2 *bytes* ou 4 *nibbles*, ou múltiplos desses.**

Apenas como ilustração, apresenta-se a seguir algumas equivalências numéricas entre os sistemas decimal, binário, octal e hexadecimal:

DECIMAL	BINÁRIO	OCTAL	HEXADECIMAL
0	00000	0	0
1	00001	1	1
2	00010	2	2
3	00011	3	3
4	00100	4	4
5	00101	5	5
6	00110	6	6
7	00111	7	7
8	01000	10	8
9	01001	11	9
10	01010	12	A
11	01011	13	B
12	01100	14	C
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F
32	100000	40	20
50	110010	62	32
60	111100	74	3C
64	1000000	100	40
100	1100100	144	64
255	11111111	377	FF
1000	1111101000	1750	3E8
1024	1000000000	2000	400

1.2 CÓDIGOS BINÁRIOS

A representação dos números binários nem sempre é estabelecida a partir da sua equivalência direta com os números decimais, octais ou hexadecimais, conforme indicado anteriormente. Freqüentemente, por razão de praticidade, aumento de velocidade das operações lógicas e simplificação dos circuitos digitais (*hardware*) que executam operações lógicas diversas, sobretudo no que diz respeito a circuitos aritméticos de computadores, torna-se conveniente expressar determinadas informações de uma forma codificada, sem que a mesma guarde qualquer relação com o valor numérico, propriamente dito, que se encontra simbolizado por um determinado número em binário. Assim, existem vários códigos numéricos dentro do próprio sistema binário. Entre esses códigos destacam-se:

- **Código binário natural (8 4 2 1)**
- **BCD (Binary Coded Decimal)**
- **Código de Gray**
- **Código Excesso de 3 (XS3)**
- **Código ASCII**
- **Códigos de paridade, deteção e correção de erros, etc.**

A aplicação de cada um desses códigos depende de aspectos operacionais e da filosofia utilizada em projetos de circuitos aritméticos, por exemplo. As justificativas para a utilização de cada um deles, evidentemente, fogem ao escopo de uma abordagem introdutória sobre sistemas numéricos. Contudo, o código **BCD (8421)** e o **Código de Gray**, devido a aplicações que serão vistas a seguir, merecem algum destaque:

O código **BCD (8421)** representa os decimais de **0** a **9** sob a forma de 4 bits, de modo que a conversão de decimal para binário é realizada dígito a dígito.

Exemplo:

Encontrar o equivalente BCD do número decimal $(1990,47)_{10}$.

$$(N)_{10} = (1990,47)_{10} = \quad 1 \quad 9 \quad 9 \quad 0 \quad , \quad 4 \quad 7$$

$$\text{Daí,} \qquad (N)_{BCD} = \quad 0001 \;\; 1001 \;\; 1001 \;\; 0000 \;\; , \;\; 0100 \;\; 0111$$

Observa-se que os decimais acima do dígito **9**, não têm correspondência direta no código BCD, sendo representados dígito a dígito nesse código. Assim, por exemplo, os decimais **10** e **15** terão os seus equivalentes BCD, conforme indicado abaixo:

$$(10)_{10} \cong (0001\ 0000)_{BCD} \quad \textbf{e} \quad (15)_{10} \cong (0001\ 0101)_{BCD}$$

Os estados lógicos que não têm correspondência direta nos códigos binários, são denominados **estados indiferentes** e, como será visto na abordagem sobre Minimização de Circuitos Combinacionais e Seqüenciais, tais estados são de grande utilidade na simplificação de circuitos lógicos. O Código de Gray, por seu turno, tem a peculiaridade, como pode ser observado no quadro correspondente, de representar os equivalentes decimais na ordem crescente ou decrescente de modo que, de uma representação à outra, somente um único **bit** assume valor diferente com relação à configuração anterior. Essa característica é de grande importância em transmissão de dados e também em codificadores e conversores digitais, como será mostrado oportunamente.

A tabela apresentada abaixo mostra alguns códigos utilizados, com o objetivo de ilustrar a total inexistência da relação entre determinados códigos e o valor numérico decimal da configuração binária propriamente dito. A razão técnica de diversas codificações poderá ser encontrada na literatura especializada, e em vários trabalhos indicados na bibliografia citada.

DEC	BIN	BCD	XS3	GRAY	BIQUINARY
0	0000	0000	0011	0000	0100001
1	0001	0001	0100	0001	0100010
2	0010	0010	0101	0011	0100100
3	0011	0011	0110	0010	0101000
4	0100	0100	0111	0110	0110000
5	0101	0101	1000	0111	1000001
6	0110	0110	1001	0101	1000010
7	0111	0111	1010	0100	1000100
8	1000	1000	1011	1100	1001000
9	1001	1001	1100	1101	1010000
10	1010			1111	
11	1011			1110	
12	1100			1010	
13	1101			1011	
14	1110			1001	
15	1111			1000	

A seguir apresenta-se uma série de exercícios propostos, objetivando-se uma revisão geral sobre o assunto aqui discutido, enfatizando-se os aspectos conceituais relativos aos diversos sistemas numéricos.

1.3 EXERCÍCIOS I:

I.1 Escrever os números abaixo sob a forma de representação polinominal:

a) $(1001)_2$; b) $(0,101)_2$; c) $(101,02)_3$; d) $(1010,01)_2$; e) $(715)_8$;
f) $(157,75)_8$; g) $(1400,320)_7$; h) $(100)_8$; i) $(78E5,A)_{16}$; j) $(0,A27)_{16}$

I.2 Converter os números abaixo para o sistema numérico decimal:

a) $(101)_2$; b) $(1101,101)_2$; c) $(1110,01)_2$; d) $(0,0101)_2$; e) $(07,602)_8$;
f) $(27,36)_8$; g) $(0,004)_8$; h) $(7,77)_8$; i) $(76FA,6B)_{16}$; j) $(6BA,3A)_{16}$

I.3 Converter os seguintes números decimais em binários:

a) 0,50; b) 0,25; c) 14,776; d) 9,25; e) 10,4;
f) 512,64; g) 734; h) 1010,01; i) 7836; j) 832,17.

I.4 Quantos algarismos binários necessitamos para representar os decimais:

a) 256; b) 512; c) 2748; d) 10844; e) 1023; f) 1024

I.5 Converter os seguintes decimais para as bases octal e hexadecimal.

a) 10,84; b) 36; c) 74; d) 700; e) 2700; f) 8742,34;
g) 10000 h) 278,50; i) 0,0522; j) 888 k) 386; l) 222

I.6 Converter diretamente da base binária para a base octal e hexadecimal os seguintes números:

a) 1011101; b) 110001; c) 111000; d) 111110; e) 1101101;
f) 1001,1110; g) 1101001; h) 11,001; i) 0,1101; j) 110011,10110

I.7 Em que base r o número 106 equivale a 577 ?

I.8 Em que base r o número decimal 79 equivale ao número 142?

I.9 Em que base r o decimal 22213 equivale a 106 ?

I.10 Converter os números abaixo, conforme indicado:

a) $(543,21)_6$ para a base 9;

b) $(42,21)_5$ para a base 3;

c) $(52)_7$ para a base 6

d) $(0,00125)_{10}$ para as bases 2, 8 e 16;

e) $(FEA6,FE2)_{16}$ para a base 7

2. INTRODUÇÃO À ÁLGEBRA BOOLEANA

Os operadores, postulados e teoremas da álgebra tradicional (não booleana), que são aplicados de uma maneira sistemática na engenharia, têm o sistema numérico decimal como referência, utilizando as suas regras básicas para adição, subtração, multiplicação, divisão e outras operações matemáticas.

A álgebra booleana, que estuda as leis e processos formais de operações aplicados ao sistema binário, é assim denominada em homenagem ao matemático inglês George Boole, o qual em 1849 publicou o livro *An Investigation of the Laws of Thought, on Which are Founded the Mathematical Theories of Logic and Probability*, apresentando os princípios básicos dessa álgebra, através de uma investigação eminentemente teórica.

Conforme citação de diversos autores, somente em 1938 Shannon, C.F. veio a publicar *Symbolic Analysis of Relay and Switching Circuits* (Trans. AIEE, 57 (1938) p 713-723), implementando aplicações práticas para o trabalho teórico de George Boole. As aplicações discernidas por Shannon transformaram radicalmente a abordagem lógica das interconexões de circuitos operados por relés, sendo tais aplicações rapidamente extendida aos componentes eletrônicos tais como válvulas termoiônicas, logo em seguida aos circuitos a transistores e, modernamente, aos circuitos integrados.

A Álgebra Booleana, por se fundamentar no Sistema Binário, o qual possui apenas os dígitos **0** e **1**, é aplicável genericamente a qualquer situação lógica envolvendo variáveis que podem assumir somente dois valores discretos, mutuamente exclusivos entre sí, tais como: [baixo, alto]; [verde, azul]; [direita, esquerda]; [0,1]; [aberto, fechado]; [-5, +12]; [falso, verdadeiro]; [morto, vivo], etc.

Assim, qualquer operação lógica realizada através da álgebra booleana que associe, ou codifique, uma ou mais variáveis como em sendo **0** ou **1**, não significa, necessariamente, que a variável ou variáveis em questão apresentem o valor numérico 0 ou 1. Tais associações, ou codificações, têm uma conotação de carater eminentemente lógico, para permitir a aplicação da álgebra booleana.

Cada vez que uma dada variável ou uma dada situação é **falsa** ou **verdadeira**, por exemplo, pode-se associar a primeira condição a **0** e a segunda a **1**.

O estabelecimento dessa convenção ou codificação é chamada de lógica positiva, por associar o **0** a uma situação "desfavorável", e o **1** a uma situação "favorável". A convenção oposta é, portanto, definida como lógica negativa. O fato de se convencionar de uma forma ou doutra, evidentemente, não altera a sistemática algébrica definida por Boole. Deve-se lembrar que, o que é definido como "desfavorável" para um observador, pode ser altamente "favorável" para outro...

2.1 OPERADORES DA ÁLGEBRA BOOLEANA

Os operadores fundamentais da Álgebra Booleana são basicamente:: NOT, AND e o OR.

A aplicação do operador **NOT** a uma variável "**X**" é simbolizada por $\overline{\mathbf{X}}$ ou **X'**. Tal operador tem a propriedade de negar o estado lógico atribuído à variável, como ilustra a tabela abaixo:

X	**X'**
0	1
1	0

Esta tabela é denominada *tabela-verdade*.

A tabela-verdade do operador **AND** aplicado às variáveis "**X**" e "**Y**", cuja operação é simbolizada por **(X . Y)** , ou apenas **XY**, é definida por:

X Y	**X . Y**
0 0	0
0 1	0
1 0	0
1 1	1

Ou seja, o operador "**AND**" faz com que o resultado da sua aplicação se torne verdadeiro (estado lógico **1**), somente quando todas as variáveis envolvidas assumem valores lógicos verdadeiros.

A *tabela-verdade* do operador "**OR**" aplicado às variáveis "**X**" e "**Y**", cuja operação é simbolizada por (**X + Y**) , é definida por:

X Y	X + Y
0 0	0
0 1	1
1 0	1
1 1	1

ou seja, o operador "**OR**" faz com que o resultado da sua aplicação se torne verdadeiro quando qualquer uma das variáveis assume valor lógico verdadeiro.

Vale ressaltar que as representações (**X . Y**) para a aplicação do operador **AND** e também (**X + Y**) para o operador **OR**, não têm o mesmo significado das operações de multiplicação e adição, respectivamente, como acontece no sistema decimal.
Os três operadores acima mencionados são os operadores básicos da álgebra de Boole. Entretanto, devido à grande freqüência com que aparecem nas operações booleanas, outros operadores ou funções derivados deles, a partir da combinação dos operadores **NOT** com os operadores **AND** e **OR**, merecem destaque, a exemplo dos operadores **NAND**, **NOR**, **XOR** e **XNOR**, cujas tabelas-verdade são definidas abaixo:

X Y	NAND	NOR	XOR	XNOR
0 0	1	1	0	1
0 1	1	0	1	0
1 0	1	0	1	0
1 1	0	0	0	1

Cada um desses operadores são representados da maneira indicada a seguir, e são o resultado da aplicação dos operadores fundamentais, conforme mostrado:

NAND(X,Y) = $\overline{\mathbf{XY}}$ ou **(XY)'** :

Ao contrário do operador **AND**, a função "**NAND**" faz com que o resultado da sua aplicação se torne **falso** (estado lógico **0**) somente quando as variáveis envolvidas assumem valores lógicos **verdadeiros**.

NOR(X,Y) = $\overline{\mathbf{X+Y}}$ ou **(X+Y)'** :

De modo análogo, a função "**NOR**" faz com que o resultado da sua aplicação se torne **falso** quando qualquer uma das variáveis assume valor lógico **verdadeiro**.

XOR(X,Y) = $(\mathbf{X} \oplus \mathbf{Y}) = \overline{\mathbf{X}}\,\mathbf{Y} + \mathbf{X}\,\overline{\mathbf{Y}}$ ou **X'Y + XY'** :

A função **XOR** (ou **OR**-exclusivo) é verdadeira quando uma das variáveis é **falsa** e a outra verdadeira, ou seja, quando as variáveis envolvidas assumem valores lógicos **diferentes**, simultaneamente.

XNOR(X,Y) = $\mathbf{X} \odot \mathbf{Y} = \overline{\mathbf{X}}\,\overline{\mathbf{Y}} + \mathbf{X}\,\mathbf{Y}$ ou **X'Y' + X Y** :

Ao contrário da função **XOR**, a **XNOR** (ou **NOR-exclusivo**) é verdadeira somente quando as variáveis envolvidas assumem o mesmo valor lógico, simultaneamente. Esta função, por este motivo, é também conhecida como *função coincidência*.
Observar que a função **XNOR** é o resultado da aplicação do operador **NOT** à função XOR; isto é**:**

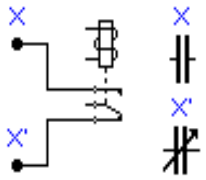
$$\mathbf{X} \odot \mathbf{Y} = \overline{\mathbf{X} \oplus \mathbf{Y}}$$

2.2 CIRCUITOS EQUIVALENTES E SIMBOLOGIA C.I.'s

Antes do estudo dos postulados e teoremas da Álgebra de Boole, torna-se extremamente interessante identificar-se os circuitos equivalentes inerentes a cada uma das funções anteriormente apresentadas. O conhecimento de tais circuitos equivalentes auxilia sobremaneira o entendimento das relações algébricas, permitindo uma visão conceitual dos teoremas, postulados e propriedades da álgebra em questão.

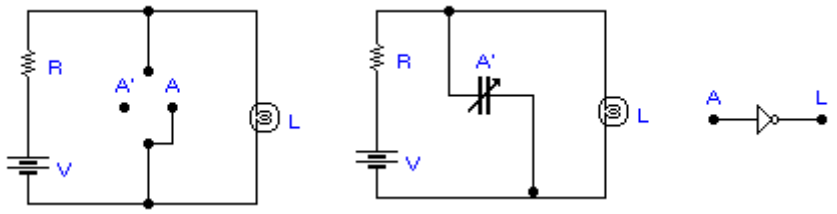
Para o desenvolvimento dos circuitos equivalentes, utilizam-se chaves convencionais ou contatos de relés (**X** e **X'**), uma fonte de tensão (**V**), um resistor (**R**) e uma carga, simbolizada por uma lâmpada (**L**). O estado lógico **1** estará associado a: contato fechado ou bobina do relé energizada ou lâmpada acesa. O estado lógico **0** estará associado às situações: contato aberto ou bobina do relé desenergizada ou lâmpada apagada!

O relé representado abaixo, como se vê, possui dois contatos. Um deles encontra-se permanentemente fechado (estado lógico **1**) na condição de repouso, ou seja, quando a bobina não se encontra energizada (estado lógico **0**); este contato está sendo definido como **X'.** O outro contato (**X**) atua de modo oposto: encontra-se aberto (estado lógico **0**) na condição de repouso. Quando a bobina do relé passa a estar energizada (estado lógico **1**), os dois contatos mudam de estado: **X'** que se encontrava em estado lógico **1** (contato fechado), muda para o estado lógico **0** (contato aberto), enquanto **X** que se encontrava no estado lógico **0**, muda para o estado **1**. O funcionamento desses contatos encontra-se representado pelos símbolos indicados ao lado dos mesmos, no diagrama abaixo

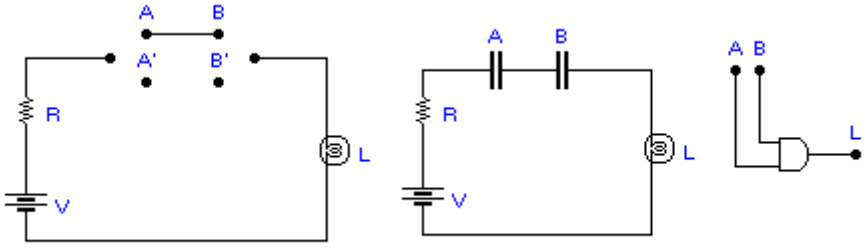


Os operadores básicos e os seus derivados mais utilizados, acima definidos sob a forma de *tabelas-verdade*, são simulados a seguir por circuitos elétricos muito simples, tendo ao lado o diagrama correspondente em contatos de relés e ainda a simbologia própria dos circuitos lógicos integrados (**C.I**.'s), para cada uma das portas lógicas, conforme aparecem nos diagramas eletrônicos dos circuitos lógicos ou digitais.

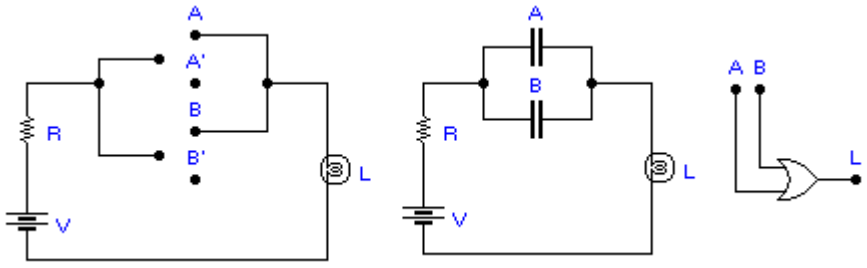
NOT: L = $\overline{A}$ ou L = A'



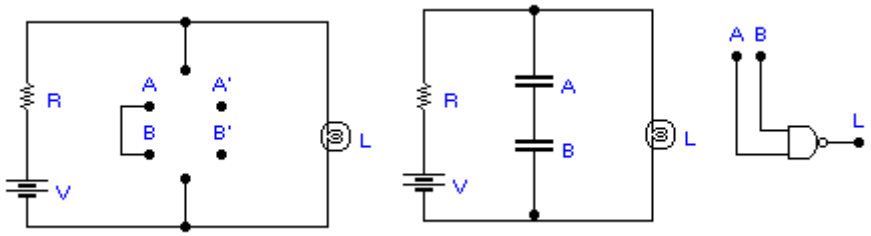
AND: L = A . B



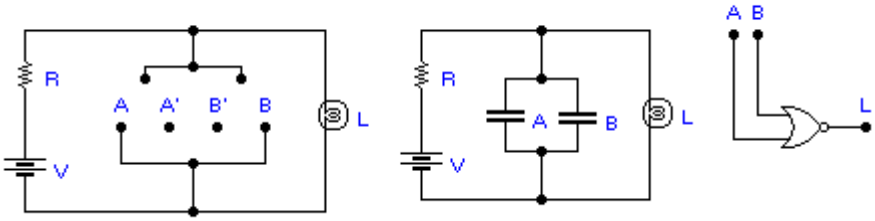
OR: L = A + B



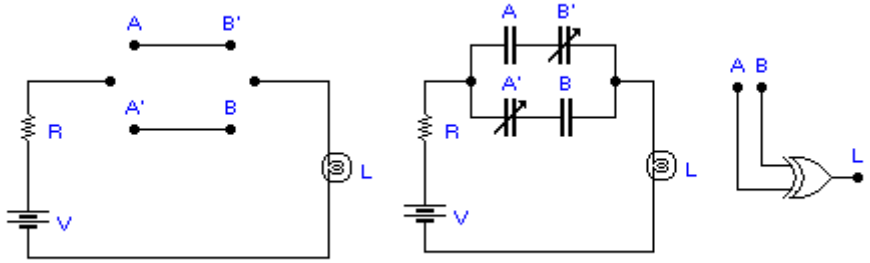
NAND: L = $\overline{A \cdot B}$ ou L = (A . B)'



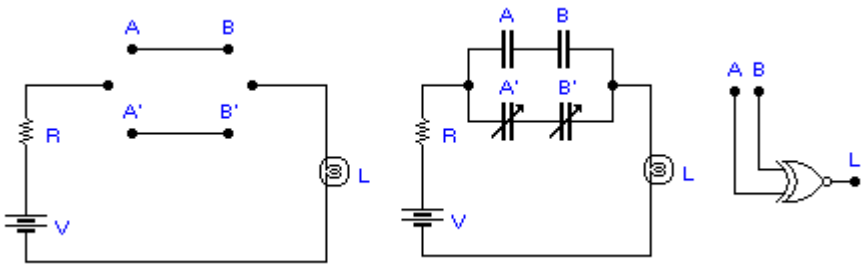
NOR: L = $\overline{A + B}$ ou L= (A + B)'



XOR : L = A ⊕ B = $\overline{A}$. B + A . $\overline{B}$ ou L = (A'. B + A . B')



XNOR: $L = A \odot B = \bar{A} \cdot \bar{B} + A \cdot B$ **ou** $L = (A' \cdot B' + A \cdot B)$



2.3 POSTULADOS E TEOREMAS DA ÁLGEBRA BOOLEANA

Após a associação de cada um dos operadores ou funções com os seus circuitos equivalentes, torna-se muito simples a interpretação dos postulados, teoremas e propriedades apresentados abaixo:

Postulados:

NOT	AND	OR
$\bar{0} = 1$	**0 . 0 = 0**	**0 + 0 = 0**
	0 . 1 = 0	**0 + 1 = 1**
$\bar{1} = 0$	**1 . 0 = 0**	**1 + 0 = 1**
	1 . 1 = 1	**1 + 1 = 1**

Teoremas e Propriedades:

1. Sobre "1" e "0"

$0 + X = X$ $\qquad$ $0 \cdot X = 0$

$1 + X = 1$ $\qquad$ $1 \cdot X = X$

2. Comutativas

$X + Y = Y + X$ $\qquad$ $X \cdot Y = Y \cdot X$

3. Associativas

$X+(Y + Z) = (X + Y) + Z$ $\qquad$ $X.(Y \cdot Z) = (X \cdot Y).Z$

4. Distributivas

$(X + Y).(Z + W) = X \cdot Z + X \cdot W + Y \cdot Z + Y \cdot W$

5. Idempotência

$X + X = X$ $\qquad$ $X \cdot X = X$

6. Complementares

$X + \bar{X} = 1$ $\qquad$ $X \cdot \bar{X} = 0$

7. Absorção

$X + X.Y = X$ e $X(X + \bar{Y}) = X$ e $X + \bar{X}Y = X + Y$

8. Inversão ou Teorema de Morgan

$\overline{(X + Y)} = \bar{X} . \bar{Y}$ e $\overline{(X . Y)} = \bar{X} + \bar{Y}$

9. Involução

$\bar{\bar{X}} = X$

10. Adjacência

$X . Y + X . \bar{Y} = X$

11. Dualidade

Uma relação booleana pode ser transformada na sua dual da seguinte maneira:

* **Troca-se OR por AND e vice-versa e**
* **Troca-se 0 por 1 e vice-versa.**

X + 0 = X	**X . 1 = X**
$\mathbf{X + \bar{X} = 1}$	$\mathbf{\bar{X} . X = 0}$
X + Y = Y + X	**X . Y = Y . X**
X + X Y = X	**X .(X + Y) = X**
$\mathbf{\overline{\bar{X} + \bar{Y}} = X . Y}$	$\mathbf{\overline{X . Y} = \bar{X} + \bar{Y}}$
$\mathbf{X + \bar{X} Y = X + Y}$	$\mathbf{X.(\bar{X} + Y) = X . Y}$

Observa-se que relações ou expressões lógicas duais não são, necessariamente, equivalentes do ponto de vista lógico!

Exemplos & Aplicações:

Exemplo 1:

Aplicar o teorema de De Morgan à função:

$$F = A.\overline{B}.C + \overline{A}.B + \overline{A}.\overline{B}.C$$

1^0 Passo: Negar a função:

$$\overline{F} = A.\overline{B}.C + \overline{A}.B + \overline{A}.\overline{B}.C$$

2^0 Passo: Trocar operadores externos e negar as variáveis

$$\overline{F} = (A.\overline{B}.C) \,.\, (\overline{A}.B) \,.\, (\overline{A}.\overline{B}.C)$$

3^0 Passo: Trocar operadores internos e negar variáveis internas

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

4^0 Passo: Complementar a função $\overline{F}$, revertendo-se à função F (verdadeira)

$$F = \overline{(\overline{A}+B+\overline{C}) \,.\, (A+\overline{B}) \,.\, (A+B+\overline{C})} = \mathbf{F}$$

Pode-se observar que a aplicação do teorema de De Morgan permitiu a obtenção da mesma função lógica, através de diferentes operadores!

Tais manipulações algébricas podem vir a ser extremamente útil na implementação de circuitos lógicos, como será mostrado oportunamente!!

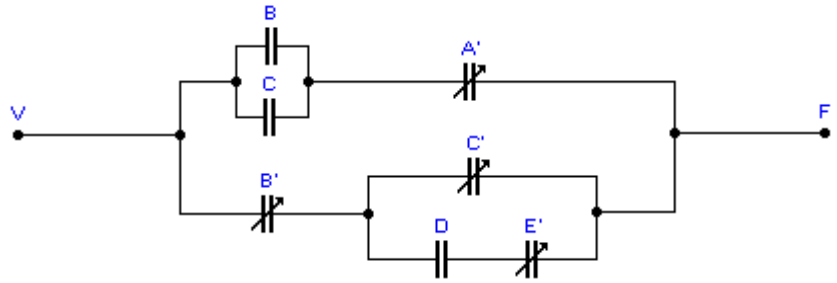
Exemplo 2:

Projetar um circuito a relés para implementar a função:

$$F = \overline{A}.(B + C) + \overline{B}.(\overline{C} + D.\overline{E})$$

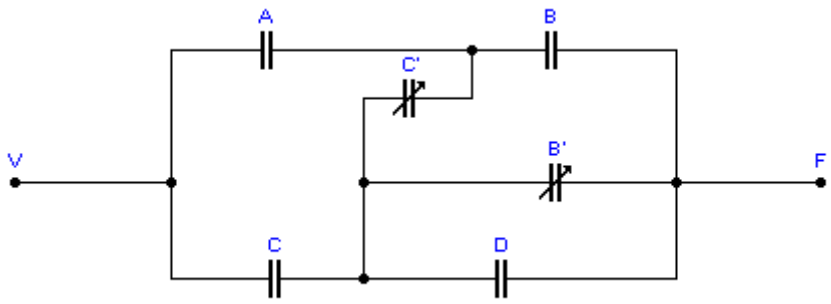
Solução:

Aplicando-se os circuitos equivalentes, identificados anteriormente, para implementar as funções **AND** e **OR**, observa-se que a equação proposta representa dois ramos em paralelo. Um deles está descrito por $\overline{A}$**(B+C),** o que corresponde ao contato **A** em série com dois contatos em paralelo definidos por **(B+C)**. Aplicando-se as mesmas considerações para o outro ramo, tem-se o circuito abaixo como resultado:



Exemplo 3:

Escrever a expressão lógica para o circuito abaixo, simplificar a expressão resultante e esboçar o circuito equivalente:



Solução:

Considerando-se todos os ramos e caminhos, sejam de contatos em série ou em paralelo, para que o estado lógico presente em **V** seja comunicado ao ponto **F**, pode-se escrever:

$$F = A.B + A.\overline{C}.\overline{B} + A.\overline{C}.D + C.D + C.\overline{B} + C.\overline{C}.B$$

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

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

$$= A.B + A.\overline{B} + \overline{B}.C + A.\overline{C}.D + C.D$$

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

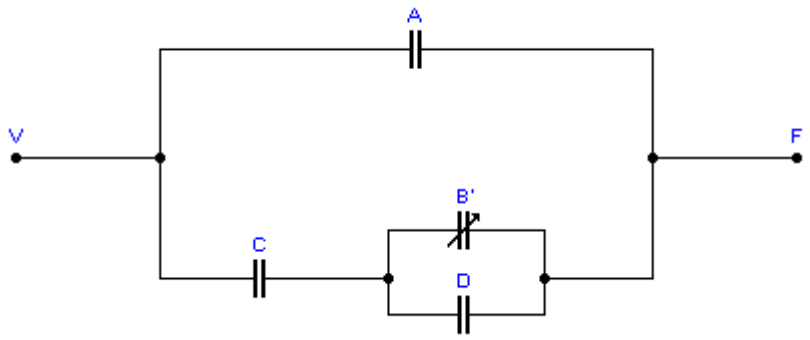
$$= A + \overline{B}.C + A.\overline{C}.D + C.D$$

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

$$= A + \overline{B}.C + C.D + A.D$$

$$= A.(1 + D) + C.(\overline{B} + D) \quad \text{ou} \quad F = A + C.(\overline{B}+ D)$$

Pelo resultado acima, o circuito equivalente pode ser representado por:



Comparando-se os dois circuitos, chega-se à conclusão de que esta segunda versão possui apenas 4 contatos de relés, em lugar de 6, conforme o circuito anterior. Isto demonstra que a manipulação algébrica de uma dada equação lógica pode promover maior economia na implementação da função final.

Exemplo 4:

Provar que $X + \bar{X}.Y = X + Y$. Sabe-se que:

$X + Y = (X + Y) . (\bar{X} + X)$ logo,

$= X.\bar{X} + X.X + \bar{X}.Y + X.Y$

$= X + \bar{X}.Y + X.Y$ ou

$= X + X.Y + \bar{X}.Y$

$= X.(1 + Y) + \bar{X}.Y = X + \bar{X}.Y$ c.q.d.

Exemplo 5:

Mostrar que $X.Y + \bar{Y}.Z + X.Z = X.Y + \bar{Y}.Z$

Sabe-se que:

$X.Y + \bar{Y}.Z = X.Y.(1 + Z) + \bar{Y}.Z.(1 + X)$

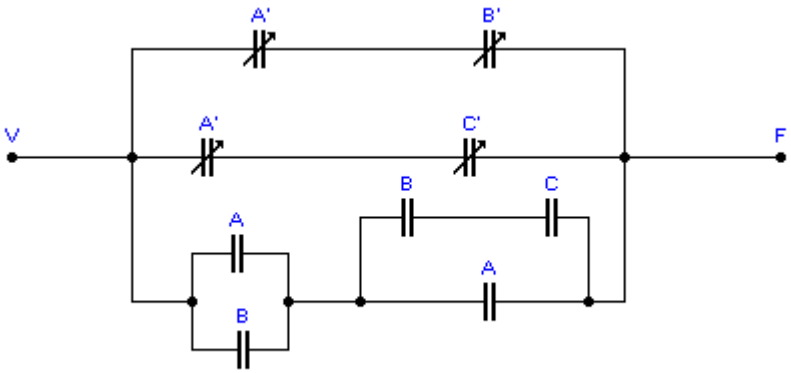
$= X.Y + X.Y.Z + \bar{Y}.Z + X.\bar{Y}.Z$

$= X.Y + \bar{Y}.Z + X.Z.(Y + \bar{Y})$

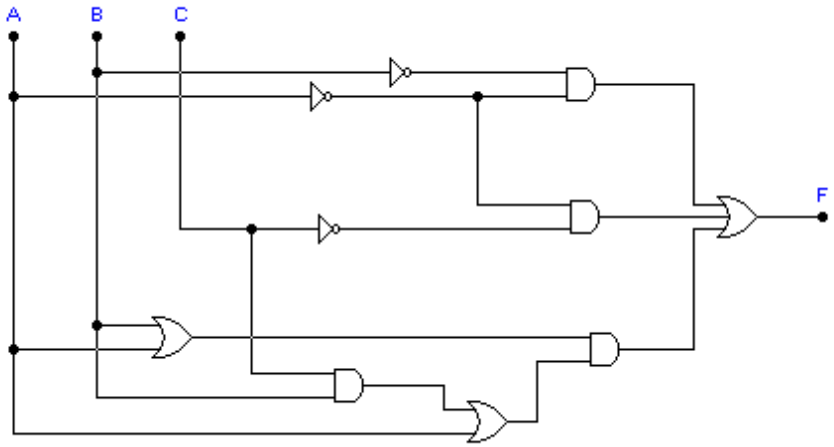
$= X.Y + \bar{Y}.Z + X.Z$ c.q.d.

Exemplo 6:

Simplificar a função descrita pelas seguintes alternativas de circuitos, cujos diagramas são apresentados tanto sob a forma de relés quanto sob a forma de portas lógicas integradas:



A alternativa do circuito acima, em diagrama de circuitos lógicos integrados, seria:



A partir de qualquer um dos circuitos acima, chega-se à seguinte equação lógica:

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

Aplicando-se inicialmente a propriedade distributiva da Álgebra booleana, e logo após diversas outras propriedades indicadas anteriormente, tem-se:

$$= A.A + A.B.C + A.B + B.B.C + \overline{A} \cdot \overline{B} + \overline{A} \cdot \overline{C}$$

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

$$= A.(1 + B.C + B) + B.C + \overline{A}.\overline{B} + \overline{A}.\overline{C}$$

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

$= A + \overline{B} + \overline{A}.\overline{C} + B.C \quad = A + \overline{A}\,\overline{C} + \overline{B} + B\,C$

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

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

$\mathbf{= A + \overline{B} + 1 = 1.}$

Ou seja: A função resultante equivale ao nível lógico 1.

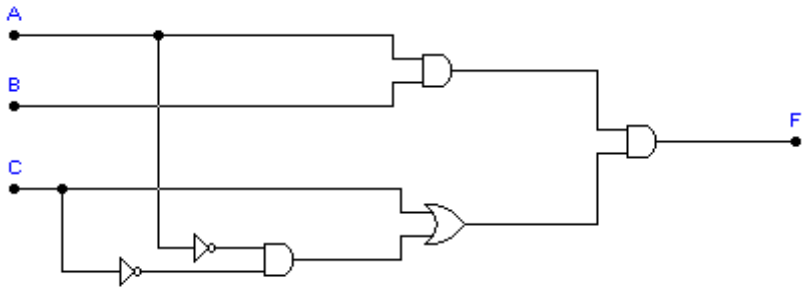
Como deve ser interpretado este resultado?

Exemplo 7:

Utilizando a simbologia dos circuitos integrados para representar as portas lógicas, esboçar o diagrama que corresponde à função:

$$\mathbf{F = A.B\,(\,C + \overline{A}.\overline{C})}$$

Solução:



Existe um circuito equivalente mais simples?

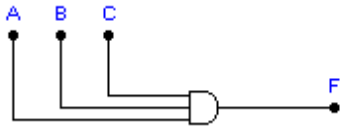
Solução:

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

$= A.B.C + A.\overline{A}.B.\overline{C} = A.B.C + 0$

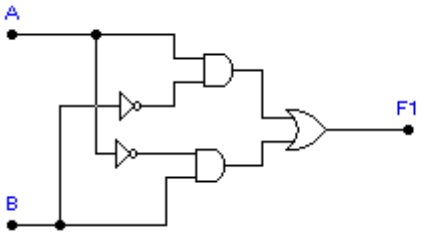
$= A\,B\,C$

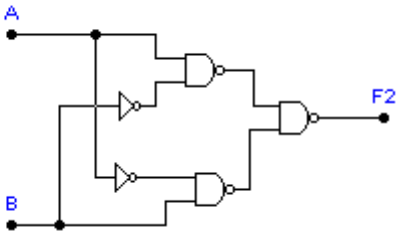
Ou seja, existe um circuito mais simples, constituído apenas por uma porta lógica **AND**, com três entradas:



Exemplo 8:

Escrever as equações lógicas que descrevem o funcionamento dos circuitos a seguir e, através da Álgebra, provar que os mesmos são equivalentes!





Solução:

Por inspeção, vê-se que

$$F_1 = A.\overline{B} + \overline{A}.B \qquad \text{e} \qquad F_2 = (A.\overline{B}) \,.\, (\overline{A}.B).$$

Ainda, noutra alternativa de notação,

$$\mathbf{F_1 = A.B' + A'.B} \qquad \mathbf{e} \qquad \mathbf{F_2 = (\,(A.B')' \,.\, (A'.B)'\,)'.}$$

Aplicando-se o teorema de De Morgan à função F_1,

$$\overline{F}_1 = A.\overline{B} + \overline{A}.B \qquad = \qquad \overline{F}_1 = (A.\overline{B}) \,.\, (\overline{A}.B)$$

Daí,

$$F_1 = \overline{\overline{F_1}} = \overline{(A.\overline{B}) \,.\, (\overline{A}.B)} = F_2.$$

Conclusão: Os circuitos são equivalentes.

Exemplo 9:

Prova de equivalência entre funções através de "Tabelas Verdade".

Provar que:

$$C + A.\overline{C} = A + C$$

Para a comprovação da igualdade acima, pode-se construir uma tabela verdade como a mostrada abaixo, contendo as expressões de interesse.

A	C	A+C	$A \,.\, \overline{C}$	$C+A\overline{C}$
0	0	0	0	0
0	1	1	0	1
1	0	1	1	1
1	1	1	0	1

Por inspeção, observa-se que as colunas relativas a $(C + A.\overline{C})$ e (A + C) são equivalentes, comprovando-se portanto a equivalência entre as funções.

2.4 PROPRIEDADES DAS FUNÇÕES "NAND"

a) $\overline{A.B.C} = \overline{B.A.C} = \overline{C.B.A}$

b) $\overline{(A.B)}.C \neq A.\overline{(B.C)}$

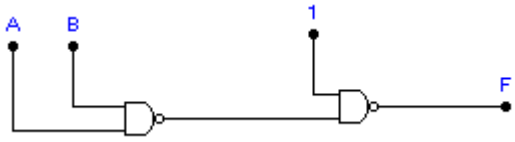
2.4.1 Expressões que envolvem apenas "NAND'S".

1. NOT

$F = \overline{A} = \overline{A \cdot 1} = \overline{A \cdot A}$

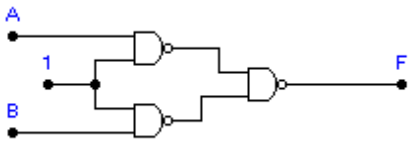
2. AND

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



3. OR

$F = \overline{\overline{A + B}} \quad F = \overline{\overline{(A \cdot 1)} \cdot \overline{(B \cdot 1)}}$



2.5 PROPRIEDADES DAS FUNÇÕES "NOR"

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

b) $\overline{\overline{A + B} + C} \neq \overline{A + \overline{B + C}} \neq \ldots$

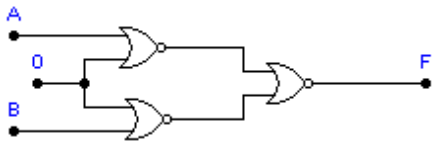
2.5.1 Expressões que envolvem apenas "NOR'S".

1. NOT

$$F = \overline{A} = \overline{A + 0} = \overline{A + A}$$

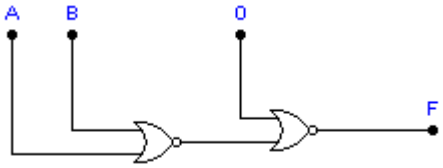
2. AND

$$F = A.B \qquad F = \overline{A.B} = \overline{A} + \overline{B}$$



3. OR

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



2.6 PROPRIEDADES DAS FUNÇÕES “XOR” E “XNOR”

As funções **XOR** e **XNOR** apresentam algumas propriedades, mostradas abaixo, as quais facilitam a implementação dos circuitos lógicos:

X ⊕ Y	**X ⊙ Y**
0 ⊕ 0 = 0	0 ⊙ 0 = 1
0 ⊕ 1 = 1	0 ⊙ 1 = 0
1 ⊕ 0 = 1	1 ⊙ 0 = 0
1 ⊕ 1 = 0	1 ⊙ 1 = 1
0 ⊕ X = X	0 ⊙ X = X’
1 ⊕ X = X’	1 ⊙ X = X
X’⊕ X = 1	X’ ⊙ X = 0
X ⊕ X = 0	X ⊙ X = 1
X’⊕ Y = X⊙Y	X’ ⊙ Y = X ⊕ Y
X ⊕ Y’ = X⊙Y	X ⊙ Y’ = X ⊕ Y
(X ⊕ Y)’ = X⊙Y	(X ⊙ Y)’ = X ⊕ Y
X⊕Y⊕X.Y = X+Y	X⊙Y⊙X.Y=X+Y
X ⊕ X.Y = X.Y’	X ⊙ X.Y = (X.Y’)’
X⊕(X + Y) = X’.Y	X⊙(X+Y)=(X’.Y)’
X ⊕ X’.Y = X + Y	X ⊙ X’.Y = (X + Y)’
X⊕(X’+Y)=(X.Y)’	X⊙(X’+Y)=X.Y

Recomenda-se, como exercício, a demonstração algébrica das relações do quadro acima, onde as variáveis negadas estão representadas por **X’** , **Y’**, **(X + Y)’** , etc.

Devido à grande vantagem da aplicação do OR exclusivo ou da Coincidência na implementação dos circuitos lógicos, deve-se sempre procurar verificar se existe a possibilidade de se expressar equações lógicas mais complexas, através desses operadores. Vale observar que existem circuitos integrados que efetuam as operações **XOR** e **XNOR** diretamente.

2.7 FORMAS CANÔNICAS

As expressões booleanas, envolvendo os diversos operadores, podem ser escritas na sua forma canônica, seja em soma de produtos (**S.O.P.**) ou em produto de somas (**P.O.S.**).

A vantagem de se exprimir uma função booleana nessa forma reside no fato desse tipo de expressão permitir a representação numérica da função, facilitando assim os procedimentos de simplificação.

As reduções ou simplificações sistemáticas das funções, sobretudo através dos mapas de Karnaugh e do método Quine McCluskey, se baseiam nas formas canônicas das funções a serem simplificadas. Esses procedimentos permitem maior independência de artifícios algébricos, e facilitam a identificação da expressão mais simples para representar a mesma função lógica.

Uma função **F(A,B,C,)** escrita em **S.O.P**. ou **P.O.S.** tem a forma geral abaixo:

- **S.O.P. F(A,B,C) = (A.B.C) + (A.B.C) + ...**

ou

- **P.O.S F(A,B,C) = (A + B + C) . (A + B + C) ...**

Quando cada termo da expressão em **S.O.P.** ou **P.O.S.** contém todas as variáveis da função, diz-se que a mesma encontra-se escrita sob a forma **standard** ou **completa**.

2.7.1 **Funções em S.O.P.**

A função

$$F(A.B.C.D) = A.B.\overline{C}.D + A.\overline{B}.C.D + A.\overline{B}.\overline{C}.D + A.\overline{B}.\overline{C}.D + A.B.C.D,$$

encontra-se escrita sob a forma *standard* ou completa.

Os termos ou parcelas da expressão são denominados termos mínimos (**MINTERM's**) da função; isto porque a função é verdadeira para o número mínimo de parcelas verdadeiras presentes na expressão.

Toda **S.O.P.** completa pode ser escrita na sua representação numérica, convencionando-se:

variável falsa = **0** e variável verdadeira = **1**

A partir dessa convenção, atribuem-se valores **0** ou **1** às variáveis presentes em cada Termo Mínimo (**MINTERM**), e encontra-se o decimal correspondente a cada um deles. Esses decimais definem a representação numérica da função, conforme indicado a seguir.

Exemplo:

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

0 0 0 1 0 0 1 0 1 1 1 1

MINTERM's: (0) (4) (5) (7)

Representação numérica da função:

$$F(A,B,C) = \Sigma\ m(0,4,5,7)$$

É importante observar que a primeira variável (A) é a mais significativa (MSB), e a última (C), a menos significativa (LSB). Esta convenção deve ser obedecida, para que a representação numérica decimal se mantenha coerente com o seu equivalente binário.

Outro exemplo:

$$F(A,B,C,D) = \overline{A}.\overline{B}.\overline{C}.\overline{D} + \overline{A}.\overline{B}.C.D + \overline{A}.B.\overline{C}.D + A.B.\overline{C}.D$$

0 0 0 0 0 0 1 1 0 1 0 1 1 1 0 1

MINTERM's: (0) (3) (5) (13)

Tem-se então a representação numérica da função como em sendo:

$$\mathbf{F(A,B,C,D) = \Sigma\ m(0,3,5,13)}$$

Evidentemente, quando a função está sob a forma numérica, a sua expressão algébrica pode ser encontrada pelo processo inverso.

Exemplo:

Escrever em soma de produtos a função representada por:

$$F(A,B,C,D) = \Sigma\ m(1,5,7,13,15)$$

Processo inverso:

Sabe-se que cada MINTERM tem os correspondentes binários indicados a seguir:

MINTERM's: **(1)** **(5)** **(7)** **(13)** **(15)**

Binário: **0 0 0 1** **0 1 0 1** **0 1 1 1** **1 1 0 1** **1 1 1 1**

logo,

$$\bar{A}.\bar{B}.\bar{C}.D \quad \bar{A}.B.\bar{C}.D \quad \bar{A}.B.C.D \quad A.B.\bar{C}.D \quad A.B.C.D$$

ou

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

Quando a S.O.P. encontra-se representada na forma incompleta, a sua representação numérica não é possível. Para tornar possível esta representação, a S.O.P. incompleta deve ser transformada em completa, seja através de artifícios algébricos, ou mesmo através de tabelas verdade, sem que o valor lógico da função venha a sofrer alteração.

Exemplo de S.O.P. incompleta:

$$\mathbf{F(A,B,C,D) = \bar{A}.\bar{B}.C}$$

Vê-se que a variável "**D**" não está presente no único termo da função.

Para transformar tal equação em representação completa ou *standard*, deve-se incluir a variável "**D**" , sem alterar o valor lógico da função.

Assim, sabendo-se que $\overline{D} + D = 1$, e que F.1 = F, faz-se:

$$F(A,B,C,D) = \overline{A}.\overline{B}.C$$

$$F(A,B,C,D) = \overline{A}.\overline{B}.C\ .(\overline{D} + D)$$

$$F(A,B,C,D) = A.\overline{B}.\overline{C}.\overline{D} + A.\overline{B}.\overline{C}.D$$

ou

$$F(A,B,C,D) = \Sigma\ m\ (8,9)$$

Um modo prático de se chegar ao mesmo resultado é exemplificado abaixo:

Seja a função incompleta:

$$\mathbf{F(A,B,C,D) = A.\overline{C}.\overline{D}}$$

Como a variável "**B**" encontra-se ausente, tem-se: **1 X 0 0** onde "**X**" representa a variável "**B**", a qual pode assumir apenas os valores **0** e **1** .

Assim, dois termos possíveis são:

1 0 0 0 = 8 e 1 1 0 0 = 12

Logo, **$F(A,B,C,D) = \Sigma\ m(8,12)$,** tendo como representação algébrica :

$$\mathbf{F(A,B,C,D) = A.\overline{B}.\overline{C}.\overline{D} + A.B.\overline{C}.\overline{D}}$$

Quando o número de variáveis ausentes é grande, o método mais prático para se encontrar a função completa, e a correspondente representação numérica, é o da tabela verdade como se mostra a seguir:

Seja:

$$\mathbf{F(A,B,C,D,E) = \overline{B}.E}$$

Vê-se que as variáveis "A", "C" e "D" estão ausentes. Assim, cria-se então a tabela-verdade conforme demonstrado abaixo, fazendo-se constar todas as configurações binárias possíveis para tais variáveis e, na coluna seguinte, acrescentam-se as variáveis presentes, atribuíndo-se os valores lógicos pertinentes. Os decimais equivalentes às configurações completas, definem os MINTERM's da função.

Variáveis Ausentes	**Termos Completos**	**Decimais Equivalentes**
ACD	**ABCDE**	**MINTERMS**
0 0 0	0 **0** 0 0 **1**	1
0 0 1	0 **0** 0 1 **1**	3
0 1 0	0 **0** 1 0 **1**	5
0 1 1	0 **0** 1 1 **1**	7
1 0 0	1 **0** 0 0 **1**	17
1 0 1	1 **0** 0 1 **1**	19
1 1 0	1 **0** 1 0 **1**	21
1 1 1	1 **0** 1 1 **1**	23

ou seja, $\mathbf{F(A,B,C,D,E) = \Sigma\ m(1,3,5,7,17,19,21,23).}$

2.7.2 Funções em P.O.S.

A função abaixo encontra-se escrita sob a forma completa, em Produtos de Somas (P.O.S.):

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

Os termos ou parcelas da expressão são denominados Termos Máximos (**MAXTERMS**) da função; isto porque a função é verdadeira para o número máximo de parcelas verdadeiras, presentes na expressão!

Todo **P.O.S.** completo pode ser escrito na sua representação numérica, segundo a convenção usada anteriormente. Contudo, o procedimento para se encontrar a representação numérica de um **P.O.S.** requer que se encontre o complemento de cada uma das variáveis, antes de se encontrar o decimal equivalente, como indicado a seguir:

$$F(A,B,C,D) = (A+\overline{B}+C+\overline{D}).(\overline{A}+B+C+D).(A+B+\overline{C}+\overline{D}).(\overline{A}+B+C+\overline{D}).$$

COMPLEMENTOS: (0 1 0 1).(1 0 0 0).(0 0 1 1).(1 0 0 1)

MAXTERMS: (5) (8) (3) (9)

Daí, ordenando em ordem crescente, $F(A,B,C,D) = \Pi M(3,5,8,9)$.

Onde Π representa o produto dos Termos Máximos da função.

Outro exemplo:

$$F(W,X,Y,Z) = (\overline{W} + X + Y + \overline{Z})$$

COMPLEMENTOS: 1 0 0 1

MAXTERM: 9

Ou seja: $F(W,X,Y,Z) = \Pi M (9)$

Também, como no caso da **S.O.P.**, pode-se escrever a expressão algébrica do **P.O.S.** a partir da representação numérica, lembrando-se de encontrar o complemento antes de se reverter à variável original, como nos exemplos abaixo:

Escrever em **P.O.S.** a função:

$F(A,B,C,D) = \prod M\ (0, 4, 7, 11, 14)$. Então,

F(A,B,C,D) = (0 4 7 11 14)

0 0 0 0 0 1 0 0 0 1 1 1 1 0 1 1 1 1 1 0

COMPLEMENTOS:

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

Quando a função está expressa sob a forma incompleta, procede-se de modo similar ao caso da S.O.P., conforme os exemplos:

Exemplo 1:

$F(A,B,C,D) = (A+\overline{C}+\overline{D}).(A+\overline{B})$

F1 . F2

Pela expressão F1,

$F1 = A+\overline{C}+\overline{D}+(B.\overline{B})$

$F1 = (A+\overline{C}+\overline{D}+B).(A+\overline{C}+\overline{D}+\overline{B})$

$F1 = (A+B+\overline{C}+\overline{D}).(A+\overline{B}+\overline{C}+\overline{D}).$

Do mesmo modo,

$F2 = A+\overline{B}+(C.\overline{C})+(D.\overline{D})$

$F2 = (A+\overline{B}+C).(A+\overline{B}+\overline{C})+(D.\overline{D})$

$F2 = (A+\overline{B}+C+D).(A+\overline{B}+C+\overline{D}).(A+\overline{B}+\overline{C}+D).(A+\overline{B}+\overline{C}+\overline{D})$ ou seja,

$F(A,B,C,D) = F1 \,.\, F2$ ou

$F(A,B,C,D) =$

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

Exemplo 2:

Encontrar a representação numérica do P.O.S.

$F(W,X,Y,Z) = (\overline{W}+Y).(W+\overline{Y}+Z).(X+Y+\overline{Z})$

Considerando-se todos os valores lógicos possíveis para as variáveis ausentes em cada termo, tem-se:

$\overline{W}+Y \quad : \quad \Rightarrow \quad [\,1 + X + 0 + Z\,] \quad \Rightarrow \quad \{\,8, 9, 12, 13\,\}$

$W+\overline{Y}+Z : \quad \Rightarrow \quad [\,0 + X + 1 + 0\,] \quad \Rightarrow \quad \{\,2, 6\,\}$

$X+Y+\overline{Z} : \quad \Rightarrow \quad [\,W + 0 + 0 + 1\,] \quad \Rightarrow \quad \{\,1, 9\,\}$

$$\mathbf{F(W,X,Y,Z) = \Pi\, M\,(1,2,6,8,9,12,13)}$$

Exemplo 3:

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

Pelo método da tabela-verdade:

BD	A B C' D	MINTERM
0 0	**0** 0 **1** 0	2
0 1	**0** 0 **1** 1	3
1 0	**0** 1 **1** 0	6
1 1	**0** 1 **1** 1	7

↑ ↑ **Complementos !:**

Pelos termos mínimos resultantes tem-se

$F(A,B,C,D) = \Pi\, M\,(2,3,6,7)$

2.8 Relação entre P.O.S. / S.O.P. e Tabelas-Verdade

Seja a tabela verdade da função XOR abaixo:

DEC	A B	F
0	0 0	0
1	0 1	1
2	1 0	1
3	1 1	0

Como nos casos anteriores, a tabela-verdade especifica a função F como sendo verdadeira para as situações definidas pela equação:

$F(A,B) = \bar{A}.B + A.\bar{B}$, cuja representação numérica é:

Binário 0 1 1 0

MINTERM's 1 2 ou

$$F(A,B) = \bar{A}.B + A.\bar{B} = \sum m(1,2).$$

A mesma tabela-verdade especifica a função **F** como falsa para as situações definidas por:

$$\overline{F(A,B)} = \bar{A}.\bar{B} + A.B.$$

Aplicando-se o teorema de De Morgan à expressão acima tem-se:

$$\overline{\overline{F(A,B)}} = \overline{\bar{A}.\bar{B} + A.B} \quad \text{ou}$$

$$F(A,B) = (\bar{A}.\bar{B}) \,.\, (A.B)$$

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

Esta última expressão é a representação da função sob a forma de P.O.S.!
Encontrando-se a sua representação numérica, tem-se:

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

Complemento 0 0 . 1 1

MAXTERM's **0** **3** ou

$$F(A,B) = (A+B).(\overline{A}+\overline{B}) = \prod M\,(0,3).$$

Observa-se, entretanto, que a expressão $F(A,B) = (A+B).(\overline{A}+\overline{B})$ pode ser desenvolvida algebricamente do seguinte modo, a partir da propriedade distributiva:

$$F(A,B) = (A+B).(\overline{A}+\overline{B}) = A.\overline{A} + A.\overline{B} + \overline{A}.B + B.\overline{B}$$

$$= 0 + A.\overline{B} + \overline{A}.B + 0 .$$

Daí,

$$F(A,B) = (A+B).(\overline{A}+\overline{B}) = \overline{A}.B + A.\overline{B}.$$

Ou seja:

$$\mathbf{F(A,B) = \prod M\,(0,3) = \sum m(1,2).}$$

Isto significa que a expressão definida a partir da função verdadeira, em **S.O.P**., é equivalente à definida a partir da função falsa, em **P.O.S.**

Logo, tanto faz se exprimir uma mesma função a partir dos **1's** em **S.O.P.**, quanto a partir dos **0's** em **P.O.S.**

Vale dizer que as expressões são equivalentes entre sí, podendo uma delas ser transformada na outra, através da aplicação do teorema de De Morgan e outras propriedades algébricas!

Observa-se ainda que os números decimais que não constam da **S.O.P**. são os que definem o **P.O.S.** e vice-versa. Isto porque, quando todas as configurações entre as variáveis são examinadas e definem os valores lógicos de determinada função, os termos que não são **1's**, necessariamente serão **0's** e vice-versa. Ou seja, o que não corresponde a termo mínimo (**MINTERM**) é, necessariamente, termo máximo (**MAXTERM**) da função.

Outro exemplo:

- Seja a função F(X,Y,Z) definida pela tabela verdade a seguir:

DEC	X Y Z	F
0	0 0 0	1
1	0 0 1	0
2	0 1 0	1
3	0 1 1	0
4	1 0 0	1
5	1 0 1	1
6	1 1 0	0
7	1 1 1	1

Provar que as expressões em **S.O.P** e **P.O.S.** da função, assim definida, são equivalentes entre sí.

Solução:

A tabela-verdade indica as expressões em S.O.P. e P.O..S., respectivamente:

$$F(X,Y,Z)_{SOP} = \sum m(0,2,4,5,7) \quad \text{e} \quad F(X,Y,Z)_{POS} = \prod M(1,3,6)$$

Partindo-se da primeira expressão, pode-se escrever que:

$$F(X,Y,Z) = \sum m(0,2,4,5,7)$$

$$F(X,Y,Z)_{SOP} = \bar{X}.\bar{Y}.\bar{Z} + \bar{X}.Y.\bar{Z} + X.\bar{Y}.\bar{Z} + X.\bar{Y}.Z + X.Y.Z$$

$$F(X,Y,Z)_{SOP} = \bar{X}.\bar{Y}.\bar{Z} + \bar{X}.Y.\bar{Z} + X.\bar{Y}.\bar{Z} + (X.\bar{Y}.Z + X.\bar{Y}.Z) + X.Y.Z$$

$$= \bar{X}\,\bar{Z}.(\bar{Y}+Y) + X.\bar{Y}(\bar{Z} + Z) + X.Z(\bar{Y} + Y)$$

$$= \bar{X}.\bar{Z} + X.\bar{Y} + X.Z$$

$F(X,Y,Z)_{SOP} = \bar{X}.\bar{Z} + X.\bar{Y} + X.Z$

Da mesma maneira,

$$F(X,Y,Z)_{POS} = (X+Y+\bar{Z}).(X+\bar{Y}+\bar{Z}).(\bar{X}+\bar{Y}+Z)$$

$$= (X+\bar{Z}).(Y+\bar{Y}).(\bar{X}+\bar{Y}+Z)$$

$$= (X+\bar{Z}).(\bar{X}+\bar{Y}+Z)$$

$$= X.\bar{X} + X.\bar{Y} + X.Z + \bar{X}.\bar{Z} + \bar{Y}.Z + Z.\bar{Z}$$

$$= X.\bar{Y} + \bar{X}.\bar{Z} + \bar{Y}.\bar{Z} + X.Z$$

$$= (X.\bar{Y} + \bar{X}.\bar{Z}) + X.Z$$

$F(X,Y,Z)_{POS} = \bar{X}.\bar{Z} + X.\bar{Y} + X.Z$

Pode-se observar, então, que as expressões em **S.O.P**. e **P.O.S** resultam na mesma equação lógica, provando-se a equivalência entre as mesmas.

Exercício:

Seja a função definida pela tabela verdade abaixo:

DEC	A B C D	F
0	0 0 0 0	0
1	0 0 0 1	0
2	0 0 1 0	0
3	0 0 1 1	0
4	0 1 0 0	0
5	0 1 0 1	1
6	0 1 1 0	0
7	0 1 1 1	1
8	1 0 0 0	0
9	1 0 0 1	0
10	1 0 1 0	0
11	1 0 1 1	0
12	1 1 0 0	0
13	1 1 0 1	1
14	1 1 1 0	0
15	1 1 1 1	1

A função é verdadeira para:

S.O.P.

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

0 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1

(5) (7) (13) (15)

. .

. . $F(A,B,C,D) = \sum m(5,7,13,15)$

A partir dos zeros tem-se:

P.O.S.

$$F(A,B,C,D) = \prod M\,(0,1,2,3,4,6,8,9,10,11,12,14)$$

Como exercício, provar que as expressões em **S.O.P** e **P.O.S**. são equivalentes. Apenas para fixar idéias, apresenta-se o quadro a seguir, para uma função F(A,B,C,D), onde especificam-se os valores lógicos e as formas de representar os MINTERM's e os MAXTERM's de uma função de quatro variáveis:

DEC	A B C D	F	MINTERMS	MAXTERMS
0	0 0 0 0	0	$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$	$A+B+C+D$
1	0 0 0 1	0	$\overline{A}\,\overline{B}\,\overline{C}\,D$	$A+B+C+\overline{D}$
2	0 0 1 0	1	$\overline{A}\,\overline{B}\,C\,\overline{D}$	$A+B+\overline{C}+D$
3	0 0 1 1	1	$\overline{A}\,\overline{B}\,C\,D$	$A+B+\overline{C}+\overline{D}$
4	0 1 0 0	1	$\overline{A}\,B\,\overline{C}\,\overline{D}$	$A+\overline{B}+C+D$
5	0 1 0 1	0	$\overline{A}\,B\,\overline{C}\,D$	$A+\overline{B}+C+\overline{D}$
6	0 1 1 0	1	$\overline{A}\,B\,C\,\overline{D}$	$A+\overline{B}+\overline{C}+D$
7	0 1 1 1	0	$\overline{A}\,B\,C\,D$	$A+\overline{B}+\overline{C}+\overline{D}$
8	1 0 0 0	0	$\overline{A}\,B\,C\,D$	$\overline{A}+B+C+D$
9	1 0 0 1	0	$A\,\overline{B}\,\overline{C}\,D$	$\overline{A}+B+C+\overline{D}$
10	1 0 1 0	1	$A\,\overline{B}\,C\,\overline{D}$	$\overline{A}+B+\overline{C}+D$
11	1 0 1 1	1	$A\,\overline{B}\,C\,D$	$\overline{A}+B+\overline{C}+\overline{D}$
12	1 1 0 0	0	$A\,B\,\overline{C}\,\overline{D}$	$\overline{A}+\overline{B}+C+D$
13	1 1 0 1	1	$A\,B\,\overline{C}\,D$	$\overline{A}+\overline{B}+C+\overline{D}$
14	1 1 1 0	0	$A\,B\,C\,\overline{D}$	$\overline{A}+\overline{B}+\overline{C}+D$
15	1 1 1 1	0	$A\,B\,C\,D$	$\overline{A}+\overline{B}+\overline{C}+\overline{D}$

A função **"F"** definida como acima será representada por:

$F(A,B,C,D) = \sum m(2,3,4,6,10,11,13) = \prod M\,(0,1,5,7,8,9,12,14,15)$, ou seja:

$$F = \overline{A}.\overline{B}.C.\overline{D}+\overline{A}.\overline{B}.C.D+\overline{A}.B.\overline{C}.\overline{D}+\overline{A}.B.\overline{C}.D+\overline{A}.B.C.\overline{D}+\overline{A}.B.C.D+A.\overline{B}.\overline{C}.D$$

$F(A,B,C,D) = \prod M\,(0,1,5,7,8,9,12,14,15)$.

Ou seja:

$$F(A,B,C,D) = (A+B+C+D).(A+B+C+\overline{D}).(A+\overline{B}+C+\overline{D}).$$

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

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

Através de manipulações algébricas, conforme procedimentos anteriormente mostrados, pode-se provar que as expressões em S.O.P e P.O.S são equivalentes; ou seja:

$$F(A,B,C,D) = \sum m(2,3,4,6,10,11,13) = \prod M\,(0,1,5,7,8,9,12,14,15)$$

Observa-se, mais uma vez, que os decimais ausentes na representação em MINTERM's são os que constam da representação em MAXTERM's.

A seguir, uma série de exercícios propostos é apresentada, objetivando a aplicação dos diversos conceitos relativos ao presente capítulo.

2. 9 EXERCÍCIOS II

1. Verificar se as igualdades abaixo são verdadeiras ou falsas:

a) $(X+Y).(X+Z) = X+Y.Z$; b) $X + \overline{X}.Y = X + Y$;

c) $X.Y+\overline{Y}.Z+X.Z = X.\overline{Y}+Y.Z$; d) $\overline{X}.\overline{Y} + X+\overline{Y} = X$;

e) $(\overline{X}+\overline{Y}).(X.\overline{Y}) = X.\overline{Y}$; f)$(X.Y+\overline{X}.\overline{Y})+(\overline{X}.Y+X.\overline{Y})= X.Y+\overline{X}.\overline{Y}$

2. Simplificar algebricamente as expressões:

a) $(X+Z).\overline{(Z+W.Y)} + (VZ+W.\overline{X}).\overline{(Y+Z)}$;

b) $(\overline{X}+\overline{Y}).(\overline{X}+\overline{Z})$;

c) $X.Y + Y(W.\overline{Z} + W.Z)$;

d) $X.Y + ZW + Z + XW(Y+\overline{Z})$

3. Transformar as expressões abaixo nas equivalentes apenas em NAND'S.

a) $F = A\,\overline{B}\,C + \overline{A}\,B + \overline{A}.\overline{B}.\overline{C}$;

b) $F = A\,\overline{B}\,\overline{C}\,\overline{D} + A\,B\,\overline{C}\,D + A\,B\,\overline{C}\,\overline{D} + A\,\overline{B}\,\overline{C}\,D$;

c) $F = \overline{A}\,\overline{B}\,(\overline{C}\,\overline{D} + C\,D) + (A + \overline{B})\overline{C}\,\overline{D} + \overline{A}\,C$;

d) $F = A\,\overline{B} + \overline{A}\,B + A\,B$

4. Obter a tabela verdade para as funções:

a) $F = XY + X\overline{Y} + \overline{Y}Z$; b) $M = X.Y + X+Y$

5. Encontrar o complemento das expressões abaixo e simplificá-los:

a) $P = (B\overline{C} + \overline{A}D)\,(A\overline{B} + C\overline{D})$;

b) $Q = (\overline{AB}.A).(\overline{AB}.B)$

6. Verificar se são verdadeiras as expressões:

a) $A \oplus B \oplus C = A \odot B \odot C$; b) $A \oplus B \oplus C \oplus D = A \odot B \odot C \odot D$;

c) $A \oplus B \odot C = A \odot B \oplus C$; d) $\overline{A \odot B} \odot C = \overline{A} \oplus \overline{B \odot C}$

7. Encontrar as expressões mais simples em S.O.P. e P.O.S. para:

a) $F(A,B,C) = \sum m(1,3,7)$;
b) $F(A,B,C) = \prod M(1,3,4,7)$;
c) $F(A,B,C,D) = \sum m(0,1,2,3,4,6,12)$;
d) $F(A,B,C,D) = \sum m(0,1,2,3,4,11,12,13,14,15)$;
e) $F(A,B,C,D,E) = \sum m(0,1,2,9,13,16,18,24,25)$.
f) $F(A,B,C,D,E) = \prod M(3,5,6,8,9,12,13,14,19,22,24,25,30)$;

8. Simplificar as expressões indicadas abaixo:

a) $F(A,B,C,D) = \overline{A}\,\overline{B}\,\overline{D} + \overline{A}\,C\,D + \overline{A}\,B\,C + A\,\overline{B}\,C\,\overline{D} + A\,C\,D$

b) $F(A,B,C,D,E) = \overline{A}\,C\,\overline{D}\,\overline{E} + \overline{A}\,\overline{C}\,D\,E + D\,\overline{E} + \overline{A}\,\overline{D}\,E$;
c) $F(W,X,Y,Z) = \sum m(0,2,8,10)$;
d) $F(W,X,Y,Z) = \sum m(1,5,9,13)$;
e) $F(W,X,Y,Z) = \sum m(0,1,2,3,8,9,10,11)$.

9. Construir a tabela verdade para as seguintes funções:

a) $F(A,B) = \sum m(0,1,3)$;
b) $F(A,B,C) = \sum m(2,5,6,7)$;
c) $F(A,B,C,D) = \sum m(0,1,4,6,9,11,13,14)$;
d) $F(A,B,C,D,E) = \prod M(0,2,9,14,15,17,23,26,28,30,31)$.

10. Transformar as expressões abaixo nas equivalentes apenas em NOR'S.

a) $F = A\,\overline{B}\,C + \overline{A}\,B + \overline{A}.\overline{B}.\overline{C}$;

b) $F = A\,\overline{B}\,\overline{C}\,\overline{D} + A\,B\,\overline{C}\,D + A\,B\,\overline{C}\,\overline{D} + A\,\overline{B}\,\overline{C}\,D$;

c) $F = \overline{A}\,\overline{B}\,(\overline{C}\,\overline{D} + C\,D) + (A + \overline{B})\overline{C}\,\overline{D} + \overline{A}\,C$;

11. A partir das tabelas-verdade abaixo, expressar a função "F" sob as formas de Soma de Produtos e Produtos de Somas, e provar a equivalência entre as mesmas:

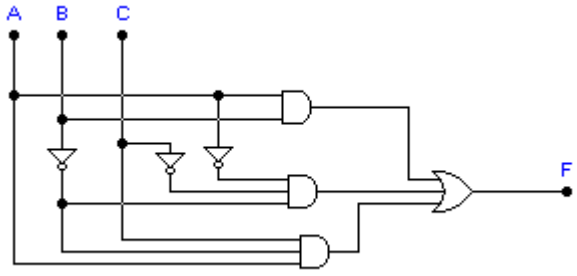
a)

X	0	0	0	0	1	1	1	1
Y	0	0	1	1	0	0	1	1
Z	0	1	0	1	0	1	0	1
F	0	0	1	1	0	1	1	0

b)

W	**X**	**Y**	**Z**	**F**
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

12. Para o circuito abaixo, encontrar o seu equivalente com portas lógicas do tipo **NAND** e, depois, repetir o exercício para portas do tipo **NOR**.



3 MINIMIZAÇÃO DE CIRCUITOS LÓGICOS

Como se depreende do capítulo anterior, os circuitos lógicos realizam fisicamente as equações booleanas, ou equações lógicas. Tais equações são determinadas a partir das relações que as variáveis binárias guardam entre sí para determinarem uma função específica. Essas relações são normalmente definidas pelo enunciado de um problema ou por tabelas-verdade.

A minimização dos circuitos que executam, na prática, as funções assim definidas, é de grande importância por propiciar a redução do número de dispositivos a serem utilizados, reduzir o tempo necessário ao desempenho previsto, aumentar a confiabilidade e, obviamente, a economia, na implementação dos circuitos ou sistemas lógicos.

Os métodos mais usados na minimização de funções são:

* **método algébrico;**
* **mapas de Karnaugh (MK);**
* **método tabular de Quine McCluskey e**
* **algoritmos computacionais.**

O método algébrico aplica-se sobretudo quando o número de variáveis é pequeno e as relações entre as mesmas são facilmente identificáveis. Neste caso, aplicam-se os postulados e teoremas da álgebra de Boole para se encontrar expressões mais simples, como vimos anteriormente.

A minimização através dos mapas de Karnaugh, baseada no teorema da adjacência (XY + XY' = X), é muito eficiente, simples e prática, permitindo a eliminação de grande número de manipulações algébricas. A sua utilização é recomendada quando a função depende de até cinco variáveis, embora seja possível a sua aplicação a funções que dependam de até seis variáveis.

O algoritmo devido a Quine & McCluskey, permite a busca exaustiva de MINTERM's ou MAXTERM's adjacentes, e se presta à identificação da função mais simples para grande número de variáveis. Embora não exista uma limitação formal quanto ao número de variáveis da função, para a sua aplicação, os seus procedimentos repetitivos torna-o enfadonho para a sua execução de modo não mecanizado.

Os métodos computacionais seriam os preferidos para a minimização de funções de grande número de variáveis. Contudo, à medida que o grau de complexidade de um circuito aumenta, deve ser considerada a alternativa da utilização de circuitos integrados de mais elevado grau de integração (MSI, LSI, ou VLSI), a exemplo de memórias programáveis, multiplexadores, decodificadores e microprocessadores, os quais permitem maior eficiência em menor espaço físico, sem depender diretamente da simplificação de funções lógicas elementares.

À excessão da simplificação algébrica, os demais procedimentos só podem ser aplicados a funções definidas de forma *standard* ou completa, em S.O.P. ou P.O.S.

O processo mais utilizado para a minimização dos circuitos de reduzido grau de integração (SSI), os quais são extremamente necessários inclusive para a interligação dos circuitos de maior grau, é conhecido como Mapas de Karnaugh, cuja análise é o principal objetivo do presente capítulo.

3.1 MAPAS DE KARNAUGH (MK)

M. Karnaugh criou, em 1953, uma representação gráfica que ordena e mostra os MINTERM's e os MAXTERM's das funções lógicas de uma forma geométrica tal que a aplicação do teorema da adjacência, citado anteriormente, se torna óbvia por inspeção.

Os Mapas de Karnaugh são aplicados sobretudo aos circuitos lógicos combinacionais, ou àqueles cuja função de saída depende única e exclusivamente dos estados lógicos das variáveis de entrada, definidos em termos de **0's** e **1's**, embora também possam ser aplicados a circuitos seqüenciais.

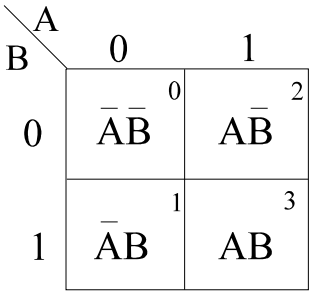
Esse procedimento se caracteriza pela representação gráfica de funções S.O.P. ou P.O.S., quando escritas de modo *standard* ou completo, conforme as definições apresentadas no capítulo anterior.

Os números internos aos lugares geométricos, representados nos MK's, correspondem aos MINTERM's ou aos decimais equivalentes às configurações binárias indicadas pelos níveis lógicos das variáveis, respeitando-se a hierarquia da variável mais significativa. Os estados lógicos das variáveis, que estão representados em todos os mapas, obedecem a uma distribuição típica do código de Gray. Por esta razão, os lugares geométricos se distribuem de modo

contínuamente adjacente tal que, somente uma única variável muda o seu estado lógico, em relação aos lugares geométricos vizinhos. Esses lugares geométricos, por sua vez, são definidos como estados adjacentes, identificando as variáveis às quais o teorema da adjacência pode ser aplicado.

Os mapas criados para representar as funções até 4 variáveis são mostrados a seguir, com os seus MINTERM's e respectivas adjacências:

Mapa de Karnaugh para 2 variáveis:



MINTERM	**ADJACÊNCIAS**
0	1,2
1	0,3
2	0,3
3	1,2

As adjacências apontadas para cada termo mínimo significam que o MINTERM em questão tem os outros termos mínimos como adjacentes, ou aqueles com os quais o dado MINTERM satisfaz a equação XY + XY' = X. Por exemplo, o MINTERM 0 mantém a condição de adjacência com os termos mínimos 1 e 2. Isto quer dizer que as seguintes expressões são válidas:

Entre os MINTERM's 0 e 1 : $\bar{\mathbf{A}}\ \bar{\mathbf{B}} + \bar{\mathbf{A}}\ \mathbf{B} = \bar{\mathbf{A}}$

Entre os MINTERM's 0 e 2 : $\bar{\mathbf{A}}\ \bar{\mathbf{B}} + \mathbf{A}\ \bar{\mathbf{B}} = \bar{\mathbf{B}}$

Do mesmo modo, o MINTERM 3 mantém a condição de adjacência com os termos mínimos 1 e 2. Isto quer dizer que as seguintes expressões também são válidas:

Entre os MINTERM's 3 e 1 : $\mathbf{A\ B + \overline{A}\ B = B}$

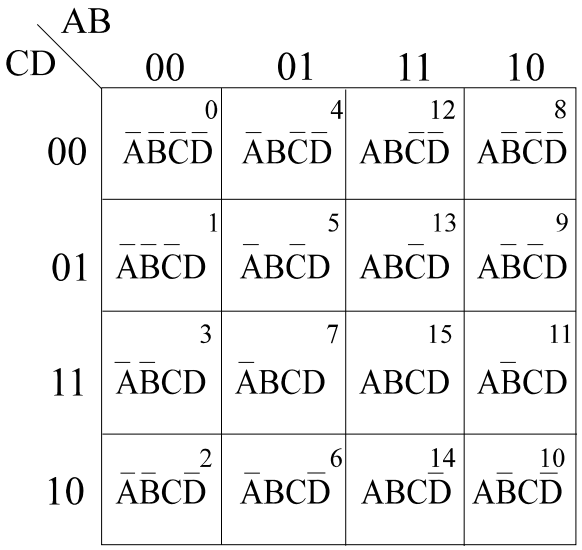
Entre os MINTERM's 3 e 2 : $\mathbf{A\ B + A\ \overline{B} = A}$

Naturalmente, o mesmo se aplica a todos os outros casos, e para qualquer número de variáveis constantes do Mapa de Karnaugh, desde que os termos envolvidos apresentem configurações de bits que diferem entre sí por apenas um único bit, como é o caso, por exemplo, das expressões A B' C D' e A B' C D que diferem apenas quanto à variável D.
Mapa de Karnaugh para 3 variáveis:

C \ AB	00	01	11	10
0	0 $\overline{A}\overline{B}\overline{C}$	2 $\overline{A}B\overline{C}$	6 $AB\overline{C}$	4 $A\overline{B}\overline{C}$
1	1 $\overline{A}\overline{B}C$	3 $\overline{A}BC$	7 ABC	5 $A\overline{B}C$

MINTERM	ADJACÊNCIAS
0	1, 2, 4
1	0, 3, 5
2	0, 3, 6
3	1, 2, 7
4	0, 5, 6
5	1, 4, 7
6	2, 4, 7
7	3, 5, 6

Mapa de Karnaugh para 4 variáveis:



MINTERM	ADJACÊNCIAS	MINTERM	ADJACÊNCIAS
0	1, 2, 4, 8	8	0, 9, 10, 12
1	0, 3, 5, 9	9	1, 8, 11, 13
2	0, 3, 6, 10	10	2, 8, 11, 14
3	1, 2, 7, 11	11	3, 9, 10, 15
4	0, 5, 6, 12	12	4, 8, 13, 14
5	1, 4, 7, 13	13	5, 9, 12, 15
6	2, 4, 7, 14	14	6, 10, 12, 15
7	3, 5, 6, 15	15	7, 11, 13, 14

Os Mapas de Karnaugh para 5 e 6 variáveis são indicados a seguir, deixando-se ao leitor o exercício da identificação das adjacências aos diversos termos mínimos do MK para 5 variáveis, e a identificação inclusive da expressão algébrica correspondente a cada um deles, no caso de seis variáveis.

Mapa de Karnaugh para 5 variáveis:

A=0

DE \ BC	00	01	11	10
00	0 $\overline{A}\overline{B}\overline{C}\overline{D}\overline{E}$	4 $\overline{A}\overline{B}C\overline{D}\overline{E}$	12 $\overline{A}BC\overline{D}\overline{E}$	8 $\overline{A}B\overline{C}\overline{D}\overline{E}$
01	1 $\overline{A}\overline{B}\overline{C}\overline{D}E$	5 $\overline{A}\overline{B}C\overline{D}E$	13 $\overline{A}BC\overline{D}E$	9 $\overline{A}B\overline{C}\overline{D}E$
11	3 $\overline{A}\overline{B}\overline{C}DE$	7 $\overline{A}\overline{B}CDE$	15 $\overline{A}BCDE$	11 $\overline{A}B\overline{C}DE$
10	2 $\overline{A}\overline{B}\overline{C}D\overline{E}$	6 $\overline{A}\overline{B}CD\overline{E}$	14 $\overline{A}BCD\overline{E}$	10 $\overline{A}B\overline{C}D\overline{E}$

A=1

DE \ BC	00	01	11	10
00	16 $A\overline{B}\overline{C}\overline{D}\overline{E}$	20 $A\overline{B}C\overline{D}\overline{E}$	28 $ABC\overline{D}\overline{E}$	24 $AB\overline{C}\overline{D}\overline{E}$
01	17 $A\overline{B}\overline{C}\overline{D}E$	21 $A\overline{B}C\overline{D}E$	29 $ABC\overline{D}E$	25 $AB\overline{C}\overline{D}E$
11	19 $A\overline{B}\overline{C}DE$	23 $A\overline{B}CDE$	31 $ABCDE$	27 $AB\overline{C}DE$
10	18 $A\overline{B}\overline{C}D\overline{E}$	22 $A\overline{B}CD\overline{E}$	30 $ABCD\overline{E}$	26 $AB\overline{C}D\overline{E}$

Neste caso, utilizam-se dois mapas de 4 variáveis fazendo com que um deles seja integralmente adjacente ao outro. A variável **A,** que é a mais significativa, é falsa no primeiro MK e verdadeira no segundo. Assim, cada lugar geométrico do segundo mapa é adjacente ao lugar correspondente do primeiro e vice-versa, uma vez que todos eles têm apenas uma única variável (**A**) que muda de valor lógico. Pelo exposto, então, o MINTERM 2, por exemplo, é adjacente ao 18, o 31 ao 15, o 8 ao 24, e assim por diante.

Mapa de Karnaugh para 6 variáveis:

O Mapa de Karnaugh para 6 variáveis é construído por 4 MK's de 4 variáveis. A exemplo do caso para 5 variáveis, os mapas são distribuídos de modo que cada um deles se apresenta como adjacente ao outro. Observa-se que o primeiro MK tem a condição das variáveis A e B serem ambas falsas (A'B'). O MK vizinho já apresenta a condição A falsa e B verdadeira (A'B) e, logo a seguir, (AB) e (AB'), tal que a seqüência completa faz com que todos os MK's sejam adjacentes entre sí, excetuando-se aqueles que se encontram em diagonal, justamente como acontece no caso dos MINTERM's 1, 2 e 0, 3 no MK para 2 variáveis. Daí, os MINTERM's 7, 23, 55 e 39 - por exemplo - serem adjacentes entre sí.

$\overline{A}\,\overline{B}$

EF \ CD	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

$\overline{A}\,B$

EF \ CD	00	01	11	10
00	16	20	28	24
01	17	21	29	25
11	19	23	31	27
10	18	22	30	26

$A\,\overline{B}$

EF \ CD	00	01	11	10
00	32	36	44	40
01	33	37	45	41
11	35	39	47	43
10	34	38	46	42

$A\,B$

EF \ CD	00	01	11	10
00	48	52	60	56
01	49	53	61	57
11	51	55	63	59
10	50	54	62	58

3.2 RELAÇÃO ENTRE TABELAS-VERDADE E MK's

As tabelas-verdade, definidas a partir das relações que as variáveis guardam com determinada função lógica, definem os MINTERM's e MAXTERM's da função. Em se conhecendo a tabela-verdade, ou mesmo apenas a representação numérica de qualquer função, pode-se representá-la diretamente no Mapa de Karnaugh, conforme mostram os exemplos abaixo:

3.2.1 **Para duas variáveis:**

Uma função definida pela tabela-verdade:

DEC	A	B	F
0	0	0	1
1	0	1	0
2	1	0	0
3	1	1	0

gera a expressão em S.O.P

F (A . B) = $\overline{\mathbf{A}}\ \overline{\mathbf{B}}$, com a representação numérica:

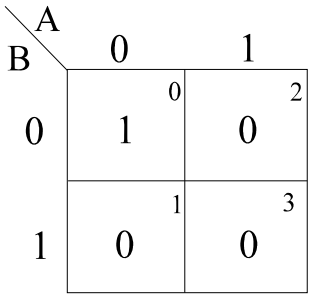
$$F(A \cdot B) = \sum m(0)$$

A mesma tabela-verdade pode gerar ainda uma função P.O.S. do tipo:

$$F(A, B) = (\mathbf{A} + \overline{\mathbf{B}})(\overline{\mathbf{A}} + \mathbf{B})(\overline{\mathbf{A}} + \overline{\mathbf{B}})$$

ou, $F(A, B) = \prod M(1,2,3)$.

Daí, o Mapa de Karnaugh correspondente pode ser facilmente construído:



3.2.2 **Para três variáveis:**

Uma função, definida diretamente sob a forma numérica, por exemplo:

F (A B C) = $\sum$ m (0, 7), gera a equação lógica: F (A B C.) = $\overline{\mathbf{A}}\ \overline{\mathbf{B}}\ \overline{\mathbf{C}} + \mathbf{A}\ \mathbf{B}\ \mathbf{C}$

A equação correspondente em P.O.S é: F (A, B, C) = $\prod$ M(1, 2, 3, 4, 5, 6), tendo por expressão algébrica:

$$\mathbf{F(A,B,C)=(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+B+C)(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+C)}$$

De qualquer uma das equações acima chega-se facilmente tanto à tabela-verdade quanto ao Mapa de Karnaugh correspondentes:

	A	**B**	**C**	**F**
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

e

C \ AB	00	01	11	10
0	0: 1	2: 0	6: 0	4: 0
1	1: 0	3: 0	7: 1	5: 0

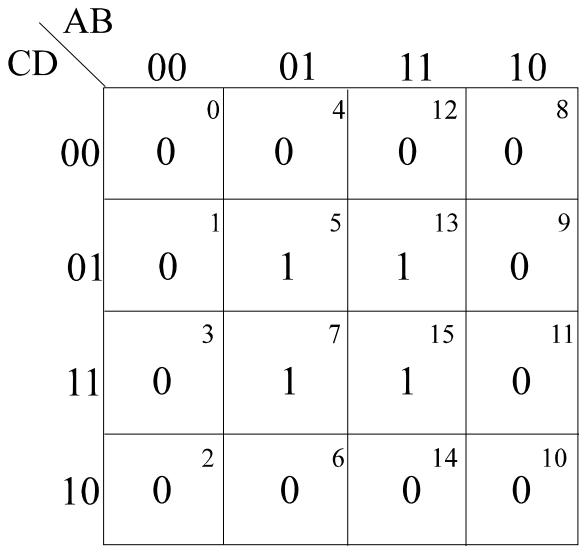
3.2.3 **Para quatro variáveis:**

Para a função F(A,B,C,D) = Π M (0, 1, 2, 3, 4, 6, 8, 9, 10, 11, 12, 14) ou

F(A,B,C,D) = Σ m (5, 7, 13, 15), tem-se, respectivamente, a tabela-verdade:

DEC	A	B	C	D	F
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

e o MK correspondente:



Obviamente, basta que se possa representar uma função qualquer na sua forma numérica em S.O.P., para se ter a indicação de em que termos mínimos do MK deve-se colocar o valor lógico **1,** que corresponde às configurações binárias para as quais a função é verdadeira. Quando se tem a expressão numérica em P.O.S., os termos máximos indicam onde se deve colocar o valor lógico **0** que corresponde às configurações binárias para as quais a função é falsa.
As tabelas-verdade e os respectivos mapas de Karnaugh para cinco e seis variáveis deixam de ser mostrados, devido à grande extensão dos mesmos. As suas aplicações serão vistas oportunamente, a partir das suas representações numéricas, por serem muito mais compactas.

3.3 MINIMIZAÇÃO OU SIMPLIFICAÇÃO DE FUNÇÕES ATRAVÉS DE MK's

Conhecendo-se o teorema da adjacência e a filosofia da construção dos Mapas de Karnaugh, as simplificações mostradas a seguir se tornam óbvias, por inspeção, sem a necessidade de desenvolvimentos ou manipulações algébricas:

Exemplo 1: Simplificar a função abaixo, definida pelas expressões em S.O.P e P.O.S.:

S. O. P.: $\mathbf{F_{SOP}(A,B) = \Sigma\, m(0,1)}$

$$F = \overline{A}\ \overline{B} + \overline{A}\ B = \overline{A}(\overline{B}+B) = \overline{A}$$

Em **P.O.S.:** $\mathbf{F_{POS}(A,B) = \Pi\, M(2,3)}$,

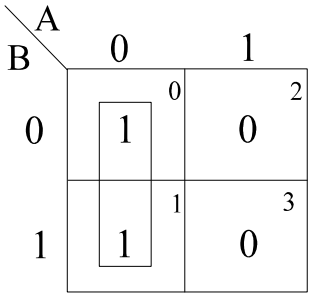
resultando em:

$$F = (\overline{A} + B)(\overline{A} + \overline{B}) = \overline{A}\ \overline{A} + \overline{A}\ \overline{B} + \overline{A}\ B + B\ \overline{B}$$

$$F = \overline{A}(\ 1 + \overline{B} + B\ ) = \overline{A}$$

Evidentemente, a simplificação realizada acima, tanto em S.O.P. quanto em P.O.S., foi levada a termo através de manipulações algébricas, apresentando o resultado F= $\overline{A}$ em qualquer dos dois casos.

O Mapa da Karnaugh correspondente à função dada, pode ser construído conforme indicado:



Vale observar, no MK em questão, que os termos adjacentes 0 e 1 estão indicados por um enlace para facilitar a sua identificação. Este enlace quer dizer que entre os elementos ou termos mínimos 0 e 1, pode-se aplicar o teorema da adjacência. Vê-se que a função em soma de produtos (S.O.P.) é verdadeira para os termos mínimos adjacentes m{0 e 1}, situação em que apenas a variável B muda de estado. Considerando-se que a mudança de estado de B não afeta o valor lógico da função, conclui-se que a mesma não depende desta variável. Assim, como A se mantém constante com o valor lógico 0, na situação identificada pelo enlace, o resultado da simplificação é:

$$F_{SOP} = \overline{A}$$

Tal conclusão, diretamente do MK, equivale à aplicação do teorema da adjacência, por inspeção.

Evidentemente, o mesmo procedimento pode ser utilizado a partir dos termos máximos para a construção da expressão simplificada sob a forma de produtos de somas (P.O.S.), a partir dos zeros da função, lembrando-se de complementar as variáveis, como visto no capítulo anterior. Neste caso específico, a função se mantém falsa para os termos máximos adjacentes M{2 e 3}, quando apenas a variável B muda de estado. Considerando-se que a mudança de estado de B não afeta o valor lógico da função, conclui-se, como no caso acima, que a mesma não

depende desta variável. Assim, como A se mantém constante com o valor lógico **1**, esta varivel é complementada, e o resultado da simplificação é igualmente:

$$F_{POS} = \overline{\mathbf{A}}$$

Mais uma vez, tal conclusão, diretamente do MK, equivale à aplicação do teorema da adjacência, por inspeção.

Exemplo 2: Simplificar a função: $F_{SOP}(A,B,C) = \Sigma\, m(0,1,4,5)$

ou $$F = \overline{\mathbf{A}}\ \overline{\mathbf{B}}\ \overline{\mathbf{C}} + \overline{\mathbf{A}}\ \overline{\mathbf{B}}\ \mathbf{C} + \mathbf{A}\ \overline{\mathbf{B}}\ \overline{\mathbf{C}} + \mathbf{A}\ \overline{\mathbf{B}}\ \mathbf{C}\ ,$$

cujo MK pode ser construído a partir apenas da expressão numérica como:

C \ AB	00	01	11	10
	0	2	6	4
0	1	0	0	1
	1	3	7	5
1	1	0	0	1

A partir de considerações similares às do exemplo anterior, nota-se no MK acima que a função S.O.P. se mantém verdadeira para os termos mínimos adjacentes m{0,1,4 e 5}, como indicado pelo enlace. Assim, as variáveis A e C mudam os seus valores lógicos de **0** para **1** sem afetarem o valor lógico da função, enquanto a variável B se mantém falsa. Conclui-se, então, que a função depende apenas da variável B falsa. Daí, a expressão simplificada corresponde simplesmente a:

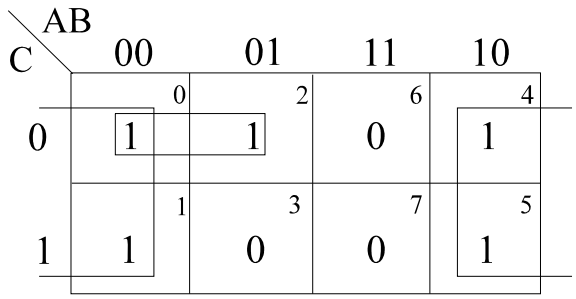
$$F_{SOP}(A,B,C) = \overline{\mathbf{B}}$$

Quanto à expressão em P. O. S., da mesma função, $F_{POS}(A,B,C) = \prod M(2,3,6,7)$,
os termos máximos adjacentes mostram apenas a variável B como verdadeira a qual, após complementada, resulta em:

$$\mathbf{F_{POS}(A.B.C) = \overline{B}}$$

Exemplo 3: Simplificar a função: $F_{POS}(A,B,C) = \prod M(3,6,7)$

Os **MAXTERM's** situados em 3, 6 e 7 indicam o MK:



A partir de considerações similares às anteriores, têm-se os seguintes termos adjacentes, com as suas respectivas simplificações:

$M\{3, 7\} = (\overline{\mathbf{B}}+\overline{\mathbf{C}})$ e $M\{6, 7\} = (\overline{\mathbf{A}}+\overline{\mathbf{B}})$

ou seja:

$$F_{POS} = (\overline{\mathbf{B}}+\overline{\mathbf{C}}).(\overline{\mathbf{A}}+\overline{\mathbf{B}}).$$

Em termos de **S.O.P.**, tem-se:

$m\{0, 2\} = (\ \overline{\mathbf{A}}\ \overline{\mathbf{C}}\ )$ ou

$m\{0,1,4 \text{ e } 5\} = (\ \overline{\mathbf{B}}\ )$

correspondendo então a:

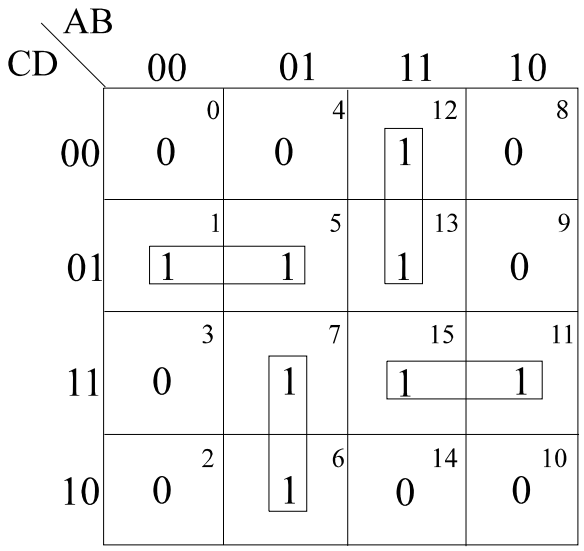
$$\mathbf{F_{SOP} = \overline{A}\ \overline{C} + \overline{B}}$$

É sempre interessante estar-se atento para o fato de que as expressões em S.O.P. são equivalentes àquelas em P.O.S., para a mesma função, mesmo na forma simplificada, como pode ser mostrado algebricamente.

Exemplo 4: Simplificar a função:

$$\mathbf{F_{SOP}(A,B,C,D) = \Sigma\, m(1,5,6,7,11,12,13,15)}$$

A partir do **MK** abaixo, observando-se os termos mínimos adjacentes, conforme os enlaces indicados, é possível chegar-se, por inspeção, às simplificações explicitadas:



tem-se:

m{1 e 5} = $\overline{\mathbf{A}}\ \overline{\mathbf{C}}\ \mathbf{D}$ e m{6 e 7} = $\overline{\mathbf{A}}\ \mathbf{B}\ \mathbf{C}$

m{12 e 13} = $\mathbf{A}\ \mathbf{B}\ \overline{\mathbf{C}}$ e m{11 e 15} = $\mathbf{A}\ \mathbf{C}\ \mathbf{D}$, resultando em:

$\mathbf{F_{SOP}(A,B,C,D)} = \overline{\mathbf{A}}\ \overline{\mathbf{C}}\ \mathbf{D} + \overline{\mathbf{A}}\ \mathbf{B}\ \mathbf{C} + \mathbf{A}\ \mathbf{B}\ \overline{\mathbf{C}} + \mathbf{A}\ \mathbf{C}\ \mathbf{D}$.

A análise da expressão em P.O.S. pode ser realizada a partir dos termos máximos adjacentes:

M{10 e 14} = $\overline{\mathbf{A}}+\overline{\mathbf{C}}+\mathbf{D}$; M{2 e 3} = $\mathbf{A}+\mathbf{B}+\overline{\mathbf{C}}$
M{8 e 9} = $\overline{\mathbf{A}}+\mathbf{B}+\mathbf{C}$; M{0 e 4} = $\mathbf{A}+\mathbf{C}+\mathbf{D}$, resultando em:

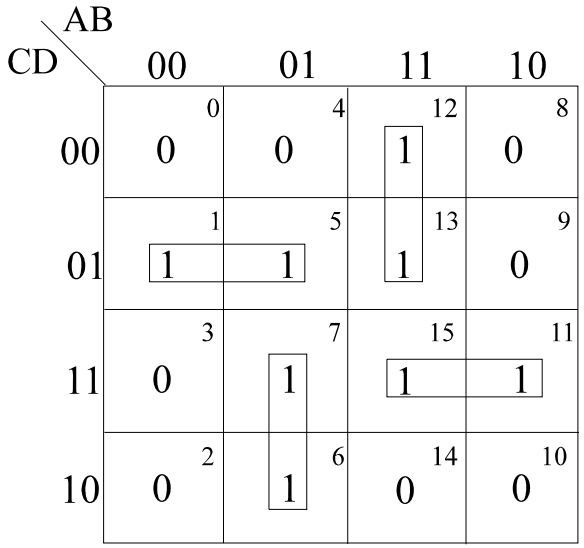
$$F_{POS} = (\overline{\mathbf{A}}+\overline{\mathbf{C}}+\mathbf{D}).(\mathbf{A}+\mathbf{B}+\overline{\mathbf{C}}).(\overline{\mathbf{A}}+\mathbf{B}+\mathbf{C}).(\mathbf{A}+\mathbf{C}+\mathbf{D})$$

Observação importante: Todas as vezes em que se procura a identificação de adjacências nos Mapas de Karnaugh, deve-se iniciar o processo a partir dos termos que se encontram isolados (sem termos adjacentes). Depois, procuram-se enlaces entre termos que sejam adjacentes apenas a um único outro termo. Em seguida procura-se identificar os que se combinam 2 a 2, 4 a 4, 8 a 8 etc. Esta regra é importante para se evitar a inclusão de termos não essenciais ou termos

redundantes. Para ilustrar uma situação típica de geração de termo redundante, pode-se tomar o exemplo recém-analisado:

$F_{SOP}(A,B,C,D) = \Sigma\, m(1,5,6,7,11,12,13,15)$

Observando-se mais uma vez o **MK** correspondente:



Nota-se que os termos **m(5, 7, 13, 15)** são todos adjacentes entre si, gerando um termo simplificado reconhecido como **BD** o qual, por sua vez, é um termo redundante; isso porque, absolutamente todos os **MINTERM's** considerados nessa simplificação já se encontravam, anteriormente, com enlaces ou adjacências com outro termo. Embora um mesmo termo mínimo ou máximo possa participar de diversos enlaces ou adjacências, sempre que todos os termos envolvidos já se encontram participando de enlaces outros, essa simplificação gerará um termo redundante, desnecessário à composição da equação lógica**.**

Para demonstrar tal afirmativa, pode-se tomar a equação simplificada anteriormente:

$F_{SOP}(A,B,C,D) = \overline{A}\ \overline{C}\ D + \overline{A}\ B\ C + A\ B\ \overline{C} + A\ C\ D$.

e acrescentar-se então o termo **BD:**

$F_{SOP}(A,B,C,D) = \overline{A}\ \overline{C}\ D + \overline{A}\ B\ C + A\ B\ \overline{C} + A\ C\ D + B\ D$.

A inclusão desse termo na equação permite o seguinte desenvolvimento algébrico:

$\mathbf{F_{SOP}(A,B,C,D) =. (\overline{A}\ C + A\ \overline{C})B + (\overline{A}\,\overline{C} + A\ C)D + B\ D.}$

Tomando-se como referência o que foi definido no Capitulo II, esta expressão equivale evidentemente a:

$\mathbf{F_{SOP}(A,B,C,D) =. B(A \oplus C) + \overline{(A \oplus C)}D + BD.}$

Lembrando-se que a relação $\mathbf{XY + \overline{Y}Z + XZ = XY + \overline{Y}Z}$ é verdadeira, conforme demonstrado ainda no Capítulo II - Exemplo 5 -, reconhece-se então que o termo **B** associado a **X**, e o termo **D** associado a **Z**, permite provar que o termo **BD** não é necessário para definir a equação lógica em questão.

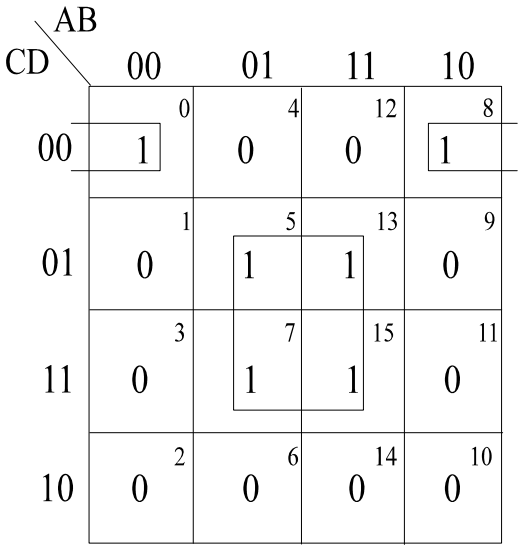
É sempre possível demonstrar-se algebricamente que um termo redundante não contribui para o estabelecimento de uma equação lógica.

Exemplo 5: Simplificar a função:

$F_{SOP}(A,B,C,D) = \sum m(0,5,7,8,13,15)$

$= F_{POS}(A,B,C) = \prod M(1,2,3,4,6,9,10,11,12,14)$

O MK para a função em questão é:



Os MINTERM's apresentam as seguintes adjacências com as simplificações resultantes:

$m\{5,7,13,15\} = \mathbf{B.D}$ **e** $m\{0 \text{ e } 8\} = \overline{\mathbf{B}}\ \overline{\mathbf{C}}\ \overline{\mathbf{D}}$

$$\mathbf{F_{SOP}(A,B,C,D) = B.D + \overline{B}\ \overline{C}\ \overline{D}}$$

Em termos de P.O.S.:

$M\{1,3,9,11\} = (\mathbf{B} + \overline{\mathbf{D}})$

$M\{4,6,12,14\} = (\overline{\mathbf{B}} + \mathbf{D})$

$M\{2,6,14,10\} = (\overline{\mathbf{C}} + \mathbf{D})$

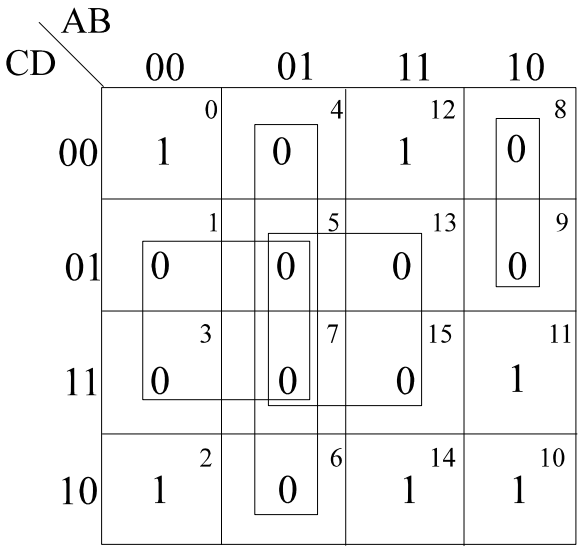
$$F_{POS}(A,B,C) = (\mathbf{B} + \overline{\mathbf{D}}) + (\overline{\mathbf{B}} + \mathbf{D}) + (\overline{\mathbf{C}} + \mathbf{D}).$$

Exemplo 6: Simplificar a função:

$F_{POS}(A,B,C,D) = \prod M(1,3,4,5,6,7,8,9,13,15)$

$$= F_{SOP}(A,B,C,D) = \Sigma\ m(0,2,10,11,12,14)$$

Partindo-se das definições chega-se ao MK:



Daí,

M{8,9} = $\overline{\mathbf{A}}$ **+ B +C**

M{1,3,5,7} = **A +** $\overline{\mathbf{D}}$

M{5,7,13,15} = $\overline{\mathbf{B}}$ + $\overline{\mathbf{D}}$

M{4,5,6,7} = **A +** $\overline{\mathbf{B}}$ ou

$$\mathbf{F_{POS} = (\overline{A} + B + C) + (A + \overline{D}) + (\overline{B} + \overline{D}) + (A + \overline{B})}$$

Quanto à $F_{SOP}(A,B,C,D)$,

m{12,14} = **A B** $\overline{\mathbf{D}}$

m{0, 2} = $\overline{\mathbf{A}}$ $\overline{\mathbf{B}}$ $\overline{\mathbf{D}}$

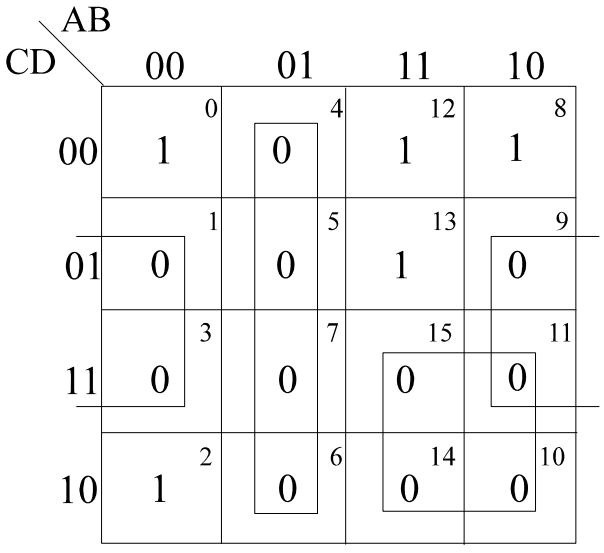
m{10,11} = **A** $\overline{\mathbf{B}}$ **C**

$$\mathbf{F_{SOP}(A,B,C,D) = A\ B\ \overline{D} + \overline{A}\ \overline{B}\ \overline{D} + A\ \overline{B}\ C}$$

Exemplo 7: Simplificar a função:

$F_{POS}(A,B,C,D) = \prod M(1,3,4,5,6,7,9,10,11,14,15)$ ou

$F_{SOP}(A,B,C,D) = \Sigma$ m(0,2,8,12,13). Daí o MK:



Os termos adjacentes são:

$M\{10,11,14,15\} = \overline{A} + \overline{C}$

$M\{4,5,6,7\} = A + \overline{B}$

$M\{1,3,9,11\} = B + \overline{D}$ isto é:

$F_{POS}(A,B,C,D) = (\overline{A} + \overline{C}).(A + \overline{B}).(B + \overline{D})$

Para a $F_{SOP}(A,B,C,D)$,

$m\{0,2\} = A\ B\ \overline{D}$

$m\{12,13\} = A\ B\ \overline{C}$

Neste caso particular, observa-se que o termo mínimo 8 é adjacente ao 0 e também ao 12, podendo gerar duas simplificações distintas, embora somente uma delas venha a ser considerada pois, as duas expressões são igualmente válidas na equação, permitindo, assim, duas soluções para o problema.
As adjacências m{0 e 8} resultam na simplificação $\overline{B}\ \overline{C}\ \overline{D}$, enquanto as adjacências m{8 e 12} em $A\ \overline{C}\ \overline{D}$, possibilitando:

$F_{SOP}(A,B,C,D) = \overline{A}\ \overline{B}\ \overline{D} + A\ B\ \overline{C} + \overline{B}\ \overline{C}\ \overline{D}$ ou

$F_{SOP}(A,B,C,D) = \overline{A}\ \overline{B}\ \overline{D} + A\ B\ \overline{C} + A\ \overline{C}\ \overline{D}$

Nos casos em que mais de uma solução se torne possível, no processo de simplificação, deve-se escolher a solução que apresente menor número de inversores, ou que contenha portas lógicas convenientes, levando-se em consideração a disponibilidade das mesmas em laboratório, ou outras considerações ditadas pelo bom senso do projetista.

Exemplo 8: Simplificar a função:

$F_{SOP}(A,B,C,D,E) = \Sigma\, m(3,5,6,8,9,12,13,14,19,22,24,25,30)$ ou

$F_{POS}(A,B,C,D,E) = \Pi\, M(0,1,2,4,7,10,11,15,16,17,18,20,21,23,26,27,28,29,31)$

A=0

DE \ BC	00	01	11	10
00	0 (0)	0 (4)	1 (12)	1 (8)
01	0 (1)	1 (5)	1 (13)	1 (9)
11	1 (3)	0 (7)	0 (15)	0 (11)
10	0 (2)	1 (6)	1 (14)	0 (10)

A=1

DE \ BC	00	01	11	10
00	0 (16)	0 (20)	0 (28)	1 (24)
01	0 (17)	0 (21)	0 (29)	1 (25)
11	1 (19)	0 (23)	0 (31)	0 (27)
10	0 (18)	1 (22)	1 (30)	0 (26)

Como nos casos anteriores, podem ser identificadas as seguintes adjacências com as respectivas simplificações:

Para $F_{SOP}(A,B,C,D,E)$:

$m\{5,13\} = \overline{A}\ C\ \overline{D}\ E$ $\qquad$ $m\{3,19\} = \overline{B}\ \overline{C}\ D\ E$

$m\{8,9,12,13\} = \overline{A}\ B\ \overline{D}$ $\quad m\{6,14,22,30\} = C\ D\ \overline{E}$ $\quad m\{8,9,24,25\} = B\ \overline{C}\ \overline{D}$

Isto resulta na expressão:

$F_{SOP}(A,B,C,D,E) = \overline{A}\ B\ \overline{D} + \overline{A}\ C\ \overline{D}\ E + \overline{B}\ \overline{C}\ D\ E + C\ D\ \overline{E} + B\ \overline{C}\ \overline{D}$

Considerações similares aplicadas à F_{POS} (A,B,C,D,E) permitem identificar as adjacências e simplificações a seguir:

$M\{0,4,16,20\} = (B + D + E)$ $\qquad$ $M\{0,1,16,17\} = (B + C + D)$

$M\{2,10,18,26\} = (C + \overline{D} + E)$ $\qquad$ $M\{11,15,27,31\} = (\overline{B}+\overline{D}+\overline{E})$

$M\{20,21,28,29\} = (\overline{A}+C+\overline{D})$ $\qquad$ $M\{7,15,23,31\} = (\overline{C}+\overline{D}+\overline{E})$ $\qquad$ ou

$F_{POS}(A,B,C,D,E)$ =

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

Exemplo 9: Simplificar a função:

$F_{SOP}(A,B,C,D,E) = \Sigma\, m(0,1,4,5,6,11,12,14,16,20,22,28,30,31)$ ou

$F_{POS}(A,B,C,D,E) = \Pi\, M(2,3,7,8,9,10,13,15,17,18,19,21,23,24,25,26,27,29)$

O Mapa de Karnaugh, para a função definida acima corresponde a:

A=0

DE \ BC	00	01	11	10
00	0: 1	4: 1	12: 1	8: 0
01	1: 1	5: 1	13: 0	9: 0
11	3: 0	7: 0	15: 0	11: 1
10	2: 0	6: 1	14: 1	10: 0

A=1

DE \ BC	00	01	11	10
00	16: 1	20: 1	28: 1	24: 0
01	17: 0	21: 0	29: 0	25: 0
11	19: 0	23: 0	31: 1	27: 0
10	18: 0	22: 1	30: 1	26: 0

Para $F_{SOP}(A,B,C,D,E)$, é possível a identificação das adjacências e suas respectivas simplificações:

$m\{11\} = \overline{\mathbf{A}}\ \mathbf{B}\ \overline{\mathbf{C}}\ \mathbf{D}\ \mathbf{E}$

A presença de todas as variáveis envolvidas neste termo significa que o MINTERM 11 encontra-se isolado, sem qualquer outro termo adjacente ao mesmo, aparecendo portanto na sua forma completa ou *standard*.

$m\{0,4,16,20\} = \overline{\mathbf{B}}\ \overline{\mathbf{D}}\ \overline{\mathbf{E}}$ $\qquad$ $m\{30,31\} = \mathbf{A\,B\,C\,D}$

$m\{0,1,4,5\} = \overline{\mathbf{A}}\ \overline{\mathbf{B}}\ \overline{\mathbf{D}}$ $\qquad$ $m\ \{4,6,12,14;20,22,28,30\} = \mathbf{C}\ \overline{\mathbf{E}}$

resultando em:

$F_{SOP}(A.B.C.D.E) = \overline{A}\ B\ \overline{C}\ D\ E + \overline{B}\ \overline{D} + A\,B.C.D. + \overline{A}\ \overline{B}\ \overline{D} + C\ \overline{E}$

Similarmente, para $\mathbf{F_{POS}\ (A,B,C,D,E)}$ tem-se:

$M\{2,10,18,26\} = C + \overline{D} + E$ $\quad$ $M\{8,9,24,25\} = \overline{B} + C + D$

$M\{9,13,25,29\} = \overline{B} + D + \overline{E}$ $\quad$ $M\{17,19,21,23\} = \overline{A} + B + \overline{E}$

$M\{3,7,19,23\} = B + \overline{D} + \overline{E}$ $\quad$ $M\{13,15\} = A + \overline{B} + \overline{C} + \overline{E}$ $\quad$ e

$M\{24,25,26,27\} = \overline{A} + \overline{B} + C$

Isto é:

$\mathbf{F_{POS}} =$

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

O exemplo a seguir, mostra a aplicação dos Mapas de Karnaugh para o caso de 6 variáveis, onde já se verifica maior dificuldade para a identificação ou visualização dos estados lógicos adjacentes:

Exemplo 10: Simplificar a função: $\mathbf{F_{POS}(A,B,C,D,E,F)} =$

$$= \prod M\ (0,5,7,8,9,12,13,23,24,25,28,29,37,40,42,44,46,55,56,57,60,61)$$

Da expressão em P.O.S. pode-se construir o conjunto de mapas para 6 variáveis, a seguir resultando nas equações:

$\overline{A}\,\overline{B}$

EF \ CD	00	01	11	10
00	0 (0)	1 (4)	0 (12)	0 (8)
01	1 (1)	0 (5)	0 (13)	0 (9)
11	1 (3)	0 (7)	1 (15)	1 (11)
10	1 (2)	1 (6)	1 (14)	1 (10)

$\overline{A}\,B$

EF \ CD	00	01	11	10
00	1 (16)	1 (20)	0 (28)	0 (24)
01	1 (17)	1 (21)	0 (29)	0 (25)
11	1 (19)	0 (23)	1 (31)	1 (27)
10	1 (18)	1 (22)	1 (30)	1 (26)

$A\,\overline{B}$

EF \ CD	00	01	11	10
00	1 (32)	1 (36)	0 (44)	0 (40)
01	1 (33)	0 (37)	1 (45)	1 (41)
11	1 (35)	1 (39)	1 (47)	1 (43)
10	1 (34)	1 (38)	0 (46)	0 (42)

$A\,B$

EF \ CD	00	01	11	10
00	1 (48)	1 (52)	0 (60)	0 (56)
01	1 (49)	1 (53)	0 (61)	0 (57)
11	1 (51)	0 (55)	1 (63)	1 (59)
10	1 (50)	1 (54)	1 (62)	1 (58)

Para F_{POS}(A,B,C,D,E,F), tem-se:

M{0, 8} = **A+B+D+E+F**

M{5, 37} = **B+C+$\overline{D}$+E+$\overline{F}$**

M{5, 7 } = **A+B+C+$\overline{D}$+$\overline{F}$**

M{23, 55} = **$\overline{B}$+C+$\overline{D}$+$\overline{E}$+$\overline{F}$**

M{40, 42, 44, 46} = **$\overline{A}$+B+$\overline{C}$+F**

M{24, 25, 28, 29, 56, 57, 60, 61} = $\overline{B}+\overline{C}+E$

M{8, 9, 12, 13, 24, 25, 28, 29} = $A+\overline{C}+E$

Ou

$F_{POS}(A,B,C,D,E,F) = (A+B+D+E+F).(B+C+\overline{D}+E+\overline{F}).(A+B+C+\overline{D}+\overline{F}).$

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

Para $F_{SOP}(A,B,C,D,E) =$

Σ m (1, 2, 3, 4, 6, 10, 11, 14, 15, 16, 17, 18, 19, 20, 21, 22, 26, 27, 30, 31, 32, 33, 34, 35, 36, 38, 39, 41, 43, 45, 47, 48, 49, 50, 51, 52, 53, 54, 58, 59, 62, 63),

tem-se:

m{4,6,20,22;36,38,52,54} = $\overline{C}\ D\ \overline{F}$ m{1,3,17,19,33,35,49,51} = $\overline{C}\ \overline{D}\ F$
m{16,17,20,21,48,49,52,53} = $B\ \overline{C}\ \overline{E}$ m{2,6,14,10,18,22,26,30} = $\overline{A}\ E\ \overline{F}$
m{18,22,26,30,50,54,58,62} = $B\ E\ \overline{F}$ m{32,34,36,38,48,50,52,54} = $A\ \overline{C}\ \overline{F}$
m{11,15,27,31,43,47,59,63} = $C\ E\ F$ m{35,39,43,47} = $A\ \overline{B}\ E\ F$
m{41,43,45,47} = $A\ \overline{B}\ C\ F$.

Naturalmente, o somatório de todas as parcelas simplificadas, definem a função em soma de produtos:

$$F_{SOP}(A,B,C,D,E) = \overline{C}\ D\ \overline{F} + \overline{C}\ \overline{D}\ F + B\ \overline{C}\ \overline{E} + \overline{A}\ E\ \overline{F} + B\ E\ \overline{F} + A\ \overline{C}\ \overline{F} + C\ E\ F + A\ \overline{B}\ E\ F + A\ \overline{B}\ C\ F.$$

O método de minimização, ilustrado acima, se constitui numa poderosa ferramenta para simplificar os circuitos lógicos capazes de resolver problemas relativos tanto à lógica combinacional quanto à seqüencial, como será demonstrado nos próximos capítulos. Contudo, os últimos exemplos servem para ilustrar que a partir de 6 variáveis esse processo de simplificação não deve ser utilizado.

3.4 COMPLEMENTAÇÕES DE FUNÇÕES ATRAVÉS De MK's

Além da utilização como método de minimização de funções lógicas, os MK's podem ser aplicados para completar funções S.O.P. ou P.O.S. que se apresentem sob a forma incompleta, evitando desenvolvimentos algébricos, conforme abordado, anteriormente, no Capítulo II.

Para ilustrar esse tipo de aplicação, propõe-se, por exemplo, o exercício de se completar a seguinte função incompleta, sem os artifícios algébricos apresentados no Capítulo II :

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

Como se vê, a função de 4 variáveis apresenta 3 termos incompletos. Como o MK completo é necessariamente o indicado abaixo, para todos os MINTERM's, basta considerar que a função em pauta será verdadeira para qualquer das 3 parcelas : $\overline{A}\ \overline{B}$ ou $\overline{C}\ D$ ou ainda $A\ \overline{C}\ D$,. que apareçam no MK, no termo mínimo correspondente.
:

CD \ AB	00	01	11	10
00	0 $\overline{A}\overline{B}\overline{C}\overline{D}$	4 $\overline{A}B\overline{C}\overline{D}$	12 $AB\overline{C}\overline{D}$	8 $A\overline{B}\overline{C}\overline{D}$
01	1 $\overline{A}\overline{B}\overline{C}D$	5 $\overline{A}B\overline{C}D$	13 $AB\overline{C}D$	9 $A\overline{B}\overline{C}D$
11	3 $\overline{A}\overline{B}CD$	7 $\overline{A}BCD$	15 $ABCD$	11 $A\overline{B}CD$
10	2 $\overline{A}\overline{B}C\overline{D}$	6 $\overline{A}BC\overline{D}$	14 $ABC\overline{D}$	10 $A\overline{B}C\overline{D}$

Assim, em todas as posições do MK, em que tais cofigurações venham a aparecer, a função deverá ser lançada como verdadeira. Isso faz com que o MK se apresente, para a função completa, sob a forma:

CD \ AB	00	01	11	10
00	0: 1	4: 0	12: 0	8: 0
01	1: 1	5: 1	13: 1	9: 1
11	3: 1	7: 0	15: 0	11: 0
10	2: 1	6: 0	14: 0	10: 0

Evidentemente, a partir deste MK, cada um dos termos que aparece (sem simplificação, já que se deseja encontrar a expressão completa), reconstitui a expressão algébrica *standard* indicada a seguir:

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

3.5 TERMOS INDIFERENTES NA MINIMIZAÇÃO POR MK's

Em diversas situações reais, alguns termos da função completa não fazem parte das possibilidades de configurações binárias de determinadas inter-relações lógicas. Um exemplo clássico disso é o Código BCD, no qual só existem configurações binárias possíveis até o decimal 9, ficando sem representação válida todas as configurações binárias correspondentes aos termos mínimos **10, 11, 12, 13, 14 e 15.**

Para efeito do sistema combinacional que gera tal código, então, todas as configurações não pertencentes ao código não têm qualquer influência nas saídas do circuito, uma vez que as mesmas jamais aparecem na entrada. Tais configurações, que não aparecem na entrada, e, por isso, não têm como afetar o resultado da saída, interpretam-se como termos indiferentes. Nesta situação, tais termos podem ser considerados indiferentemente como **0** ou **1**, uma vez que não influenciam o resultado da saída e, assim, podem ser usados no MK, a partir das eventuais adjacências que venham a criar, desde que se revelem vantajosas para a simplificação que se deseja obter. No capítulo IV, apresentam-se alguns exemplos que ilustram tal tipo de aplicação.

3.6 EXERCÍCIOS III

1. Utilizar os mapas de Karnaugh para simplificar as funções abaixo, encontrando as expressões correspondentes em S.O.P. e P.O.S.

   a) $F(A,B,C) = \Sigma\, m(1,3,7)$;

   b) $F(A,B,C) = \Pi\, M(1,3,4,7)$;

   c) $F(A,B,C,D) = \Sigma\, m(0,1,2,3,4,6,12)$;

   d) $F(A,B,C,D,E) = \Pi\, M(3,5,6,8,9,12,13,14,19,22,24,25,30)$;

2. Utilizar os mapas de Karnaugh para simplificar as expressões indicadas abaixo:

   a) $F(W,X,Y,Z) = \Sigma\, m(0,8,2,10)$;

   b) $F(W,X,Y,Z) = \Sigma\, m(1,5,13,9)$;

   c) $F(W,X,Y,Z) = \Sigma\, m(0,1,2,3,8,9,10,11)$

3. Utilizar os mapas de Karnaugh para simplificar as expressões indicadas abaixo:

   a) $F(A,B) = \Sigma\, m(0,1,3)$;

   b) $F(A,B,C,D) = \Sigma\, m(0,1,4,6,9,11,13,14)$;

   c) $F(A,B,C,D,E) = \Pi\, M(0,2,9,14,15,17,23,26,28,30,31)$

4. Simplificar, pelo método do mapa de Karnaugh, as seguintes expressões:

a) $(B+\overline{D}).(D+A+C).(E+B).(\overline{C}+D)$;
b) $(\overline{B}+\overline{A}).(\overline{B}+\overline{C})$;
c) $B.C + C.A.\overline{D} + A.D$;
d) $B.C + D.A + D + B.A.\overline{C}$

5. Simplificar a função:

   $F = \Sigma\, m(0,5,7,8,9,12,13,23,24,25,28,29,37,40,42,44,46,55,56,57,60,61)$.

6. Simplificar, pelo método do mapa de Karnaugh, as seguintes expressões:

a) $\mathbf{A\ \overline{B}\ C + \overline{A}\ B + \overline{A}\ \overline{B}\ \overline{C};}$
b) F = $\mathbf{A\ \overline{B}\ \overline{C}\ \overline{D} + A\ B\ \overline{C}\ D + A\ B\ \overline{C}\ \overline{D};}$
c) F = $\mathbf{\overline{A}\ \overline{B}(\overline{C}\ \overline{D} + C\ D) + (A + \overline{B})\ \overline{C}\ \overline{D} + (\overline{A}\ C + A\ \overline{C})\ .D}$;
d) F = $\mathbf{A\ \overline{B} + \overline{A}\ B + A\ B}$
e) P = $\mathbf{(B\overline{C} + \overline{A}D)\ (A\overline{B} + C\overline{D})}$;

7. Provar que:

$\Sigma\ m(2,7,8,13) = \Pi\ M(0,1,3,4,5,6,9,10,11,12,14,15)$

8. Encontrar a expressão mais simples, em S.O.P. e P.O.S. para:

$F(A,B,C,D,E) = \Sigma\ m(2,7,8,13,18,23,24,29)$

9. Encontrar a expressão mais simples, em S.O.P. e P.O.S. para:

$F(A,B,C,D,E) = \Pi\ M(0,5,10,15,16,21,26,31)$

10. Simplificar em termos de P.O.S e S.O.P a expressão:

$F = \Pi\ M(0,2,8,10,16,18,24,26,32,34,40,42,50,56,58)$

4 CIRCUITOS COMBINACIONAIS

Os circuitos combinacionais são circuitos lógicos cujas saídas dependem única e exclusivamente da configuração das variáveis de entrada, em termos dos seus estados lógicos, ou seja, das variáveis que são falsas ou verdadeiras em determinado instante, independendo do estado anterior das saídas. O número de saídas de um circuito combinacional pode ser menor, igual ou maior do que o número de entradas, dependendo apenas das especificações contidas no problema.

As técnicas desenvolvidas nos capítulos anteriores são satisfatórias para se estabelecer as equações lógicas, as quais descrevem o comportamento das funções, e ainda possibilitar a implementação dos circuitos de forma mais simples e econômica.

O estabelecimento das funções lógicas é realizado a partir das condições que definem as relações entre as diversas variáveis e, de um modo geral, tais relações são explicitadas por meio de um texto ou de uma tabela-verdade.

Algumas relações entre as variáveis são muito claras no próprio texto e permitem a identificação da equação resultante diretamente do mesmo, sem a necessidade de uma abordagem sistemática. Entretanto, a maior parte das interrelações exige um procedimento mais elaborado para se encontrar as equações que definem as funções das saídas.

Em geral, a determinação sistemática de um circuito combinacional é realizada em quatro passos:

- leitura e interpretação do texto que define as funções lógicas das saídas, a partir das interrelações das variáveis de entrada, especificadas segundo as condições pertinentes ao problema em foco;

- representação das variáveis e das funções resultantes sob a forma de uma ou mais tabelas-verdade;

- expressão das funções resultantes, na sua forma mais simples, em S.O.P e P.O.S.;

- implementação dos circuitos lógicos correspondentes às equações, em função das disponibilidades de circuitos integrados e demais considerações que possibilitem a otimização do circuito lógico final a ser adotado.

À medida que um circuito seja classificado como combinacional, o procedimento acima deve ser aplicado para se encontrar a sua equação e a configuração geral de interligação das diversas portas lógicas que permitem a implementação da solução do problema.

Assim, inúmeros circuitos lógicos podem ser implementados, recebendo diversas classificações tais como:

- circuitos aritméticos;
- codificadores;
- decodificadores;
- comparadores;
- geradores e detetores de paridade;
- multiplexadores;
- demultiplexadores e
- outros circuitos destinados a sistemas de comando ou controle.

Independentemente da classificação que um circuito combinacional venha a receber, a rotina para o estabelecimento da equação lógica obedece à mesma seqüência indicada acima.

A seguir, apresenta-se uma série de exemplos ilustrativos de projetos de circuitos combinacionais, previlegiando-se - por simplicidade - a implementação em S.O.P., objetivando-se uma abordagem sistemática para a solução de problemas aplicados, independentemente da classificação recebida pelos diversos circuitos.

4.1 PROJETO DE CIRCUITO COMBINACIONAL ELEMENTAR:

Projetar um circuito combinacional para permitir ligar ou desligar uma lâmpada a partir de dois pontos distintos, a exemplo do circuito de interruptores *three-way*.

Solução :
Admitindo-se dois interruptores A e B situados nos pontos de interesse, e a lâmpada F a ser ligada ou desligada, e ainda que os estados ligado ou desligado

correspondem aos estados lógicos **1** ou **0** respectivamente, têm-se a tabela-verdade e o mapa de Karnaugh indicados abaixo. Tal tabela-verdade é elaborada partindo-se das seguintes considerações:

a) considerando-se inicialmente que os dois interruptores e a lâmpada se encontrem desligados, qualquer interruptor que venha a mudar de estado lógico ligará a lâmpada.

b) do mesmo modo, quando a lâmpada se encontrar ligada, qualquer mudança de estado em qualquer dos interruptores, a desligará.
Examinando-se todas as alternativas possíveis, pode-se construir então a tabela-verdade, bem como o correspondente Mapa de Karnaugh, os quais descrevem a situação para qualquer caso:

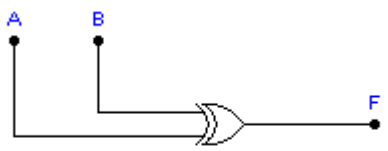
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

B \ A	0	1
0	0 (0)	1 (2)
1	1 (1)	0 (3)

A equação resultante pode ser estabelecida tanto diretamente da tabela quanto do Mapa de Karnaugh, o qual revela não haver estados adjacentes.
Pelas definições anteriores, a equação lógica que representa a função verdadeira corresponde a um XOR, ou seja:

$$F(A,B) = \overline{A}.B + A.\overline{B} \quad \text{ou} \quad F(A,B) = A \oplus B$$

O circuito equivalente é :



4.2 CIRCUITO ELEMENTAR DE ALARME 1:

Uma máquina refrigerada a água e lubrificada a óleo, sob pressão, necessita de um sinal de alarme a ser acionado quando uma ou ambas as situações abaixo se apresentem:

- a temperatura da máquina se encontra acima da máxima permitida pelo fabricante e
- a temperatura se encontra abaixo da máxima, mas o nível da água e a pressão do óleo estão abaixo dos valores médios recomendados no manual de manutenção.

Projetar um circuito lógico para acionar o mencionado alarme.

Solução :

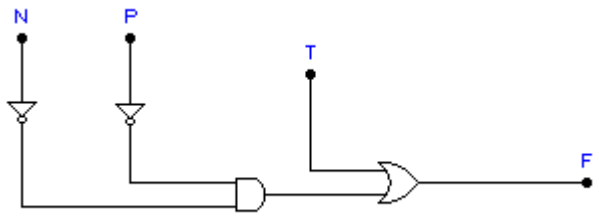
O texto do problema é extremamente claro e, apenas pelas defições das funções OR e AND pode-se chegar à equação lógica, bastando para isso convencionar-se:

F = acionar alarme;
T = temperatura da máquina acima da máxima permitida;
N = nível da água igual ou acima do recomendado;
P = pressão do óleo igual ou acima da recomendada.

A partir do texto, pode-se escrever diretamente:

$F = T + \overline{T}.\overline{N}.\overline{P}$ ou, por simplificação algébrica,

$F = T + \overline{N}.\overline{P}$, resultando no circuito lógico abaixo:



4.3 CIRCUITO ELEMENTAR DE ALARME 2:

Uma máquina funciona com 4 sensores controlando o seu estado de operação. Se a máquina estiver operando adequadamente, pelo menos duas das variáveis de controle devem estar presentes ao mesmo tempo. Quando a máquina não estiver operando corretamente, um alarme deverá soar num painel de controle. Projetar o circuito lógico do alarme.
Convencionando-se a presença da variável de controle como estado lógico **1**, a tabela-verdade abaixo demonstra a condição para soar o alarme (F=**1**), segundo as condições definidas no texto. A partir desta tabela-verdade, obtem-se o mapa de Karnaugh correspondente, conforme se observa abaixo:

DEC	A	B	C	D	F
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

Pela tabela-verdade, $F(A,B,C,D) = \Sigma\, m(0,1,2,4,8)$, gerando o MK:

CD \ AB	00	01	11	10
00	0: 1	4: 1	12: 0	8: 1
01	1: 1	5: 0	13: 0	9: 0
11	3: 0	7: 0	15: 0	11: 0
10	2: 1	6: 0	14: 0	10: 0

Observando-se o MK acima, vê-se que os termos mínimos adjacentes definem as simplificações que resultam nas seguintes expressões em S.O.P e P.O.S, respectivamente:

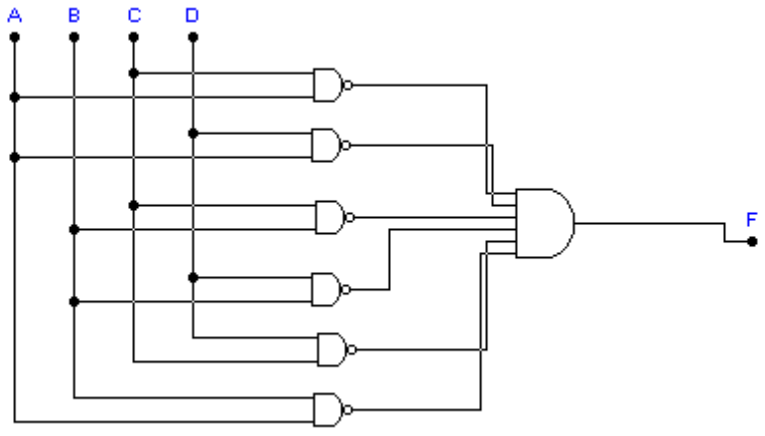
$F_{SOP} = \overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{D} + \overline{A}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}$ e

$F_{POS} = (\overline{A} + \overline{B})\,(\overline{C} + \overline{D})\,(\overline{B} + \overline{D})\,(\overline{B} + \overline{C})\,(\overline{A} + \overline{D})\,(\overline{A} + \overline{C})$

Visando-se a implementação do circuito, através de portas lógicas do tipo NAND, tem-se pela aplicação do teorema de De Morgan:

$F = (\overline{AB} \,.\, \overline{CD} \,.\, \overline{BD} \,.\, \overline{BC} \,.\, \overline{AD} \,.\, \overline{AC})$.

Desta última expressão, pode-se implementar o circuito:



4. 4 CIRCUITO ELEMENTAR DE ALARME 3:

A porta de emergência de uma aeronave deverá operar nas seguintes condições:

- se o piloto indicar a necessidade de sua operação;
- se o teor de O^2 estiver abaixo do mínimo permitido;
- se a temperatura estiver acima da máxima permitida e
- o engenheiro de vôo fizer operar "situação de emergência".

Encontrar o correspondente circuito de controle, sabendo-se que a mesma deverá operar se a primeira condição ocorrer, simultaneamente com pelo menos duas das outras três.

A tabela-verdade a seguir representa a situação lógica estabelecida no texto:

DEC	A	B	C	D	F
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

Pela tabela-verdade, pode-se escrever:

$F(A,B,C,D) = \sum m(11,13,14,15)$ e

$F(A,B,C,D) = \prod M(0,1,2,3,4,5,6,7,8,9,10,12)$

O MK correspondente resulta em:

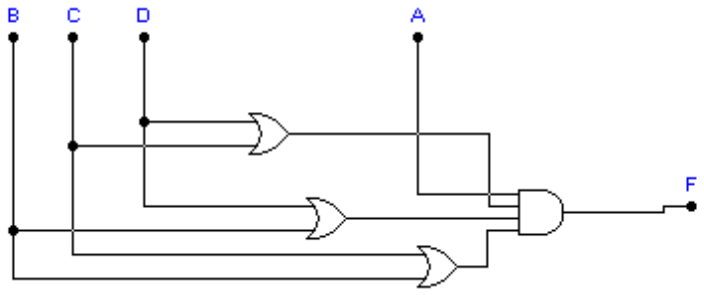
CD \ AB	00	01	11	10
00	0 (0)	0 (4)	0 (12)	0 (8)
01	0 (1)	0 (5)	1 (13)	0 (9)
11	0 (3)	0 (7)	1 (15)	1 (11)
10	0 (2)	0 (6)	1 (14)	0 (10)

As adjacências identificadas no MK acima, permitem escrever :

$$F_{SOP} = A\,(BC + CD + BD)$$

$$F_{POS} = A\,(C + D).(B + D).(B + C)$$

Tomando-se a segunda expressão para a implementação do circuito, tem-se:



4.5 CODIFICADOR ELEMENTAR X/X^2:

A partir de números binários de 3 digitos, projetar um circuito para gerar um código de saída que corresponda ao quadrado dos números binários apresentados na entrada.

a) Encontrar o circuito lógico mais simples e
b) esboçar o diagrama do circuito.

Solução :

Pelas condições estabelecidas no problema pode-se construir a tabela-verdade abaixo :

X	A	B	C	X^2	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	1
2	0	1	0	4	0	0	0	1	0	0
3	0	1	1	9	0	0	1	0	0	1
4	1	0	0	16	0	1	0	0	0	0
5	1	0	1	25	0	1	1	0	0	1
6	1	1	0	36	1	0	0	1	0	0
7	1	1	1	49	1	1	0	0	0	1

A partir da tabela acima, obtem-se as equações a seguir, as quais, devido à sua grande simplicidade, podem ser minimizadas algebricamente, conforme demonstrado :

$$Y_5 = A\,B\,\overline{C} + A\,B\,C = A\,B.(\overline{C} + C) \quad \text{ou} \quad \mathbf{Y_5 = A.B}$$

$$Y_4 = A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + A\,B\,C$$

$$= A\,\overline{B} + A\,B\,C$$

$$= A\,(B + C) \quad \text{ou} \quad \mathbf{Y_4 = A\,\overline{(B\,\overline{C})}}$$

$$Y_3 = \overline{A}\,B\,C + A\,\overline{B}\,C$$

$$= C.(\overline{A}\,B + A\,\overline{B}) \quad \text{ou} \quad \mathbf{Y_3 = C.(A \oplus B)}$$

$$Y_2 = \overline{A}\,B\,\overline{C} + A\,B\,\overline{C}$$

$$= B\,\overline{C}\,(\overline{A} + A) \quad \text{ou} \quad \mathbf{Y_2 = B\,\overline{C}}$$

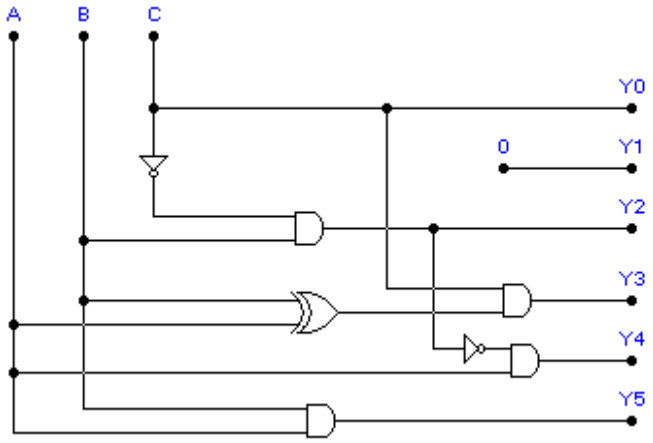
$Y_1 = 0$, por inspeção : **$Y_1 = 0$**

$Y_0 = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + A\,\overline{B}\,C + A\,B\,C$

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

$= C\,(\overline{A} + A)$ ou **$Y_0 = C$**

Daí,



4.6 COMANDO DE CIRCUITO ELÉTRICO:

Projetar um circuito lógico para controlar um sistema de iluminação de uma quadra de esportes, tal que um conjunto de refletores possa ser ligado ou desligado de 4 pontos distintos.

- elaborar a tabela-verdade e encontrar a equação correspondente;
- simplificar a equação e esboçar o circuito lógico utilizando o menor número possível de CI's.

Solução :

Convencionando-se os 4 pontos distintos como A,B,C e D e o conjunto de refletores como F, e ainda assumindo tanto os refletores quanto os interruptores situados nos diversos pontos com valor lógico **1** na condição de ligado, pode-se elaborar a tabela-verdade abaixo, partindo-se do pressuposto de que o conjunto

de refletores encontra-se desligado quando todos os interruptores estiverem nessa mesma condição.

DEC	A	B	C	D	F
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

Logo,

$$F(A,B,C,D) = \Sigma\ m(1,2,4,7,8,11,13,14)$$

O mapa de Karnaugh corresponde a:

CD \ AB	00	01	11	10
00	0 (0)	1 (4)	0 (12)	1 (8)
01	1 (1)	0 (5)	1 (13)	0 (9)
11	0 (3)	1 (7)	0 (15)	1 (11)
10	1 (2)	0 (6)	1 (14)	0 (10)

Como se vê, o mapa de Karnaugh resulta numa função para a qual não é possível se encontrar MINTERM's ou MAXTERM's adjacentes.
Vale observar que todas as vezes que os termos não adjacentes se apresentam no MK segundo uma diagonal, a função resultante estará relacionada com expressões envolvendo XOR ou XNOR, conforme demonstrado a seguir:

A função F(A,B,C,D) é definida então por :

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

$$+ A\,B\,\bar{C}\,D + A\,B\,C\,\bar{D} + A\,\bar{B}\,\bar{C}\,\bar{D} + A\,\bar{B}\,C\,D$$

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

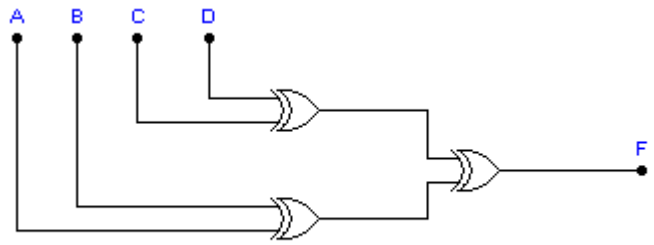
$$+ A\,B\,(\bar{C}\,D + C\,\bar{D}) + A\,\bar{B}\,(\bar{C}\,\bar{D} + C\,D)$$

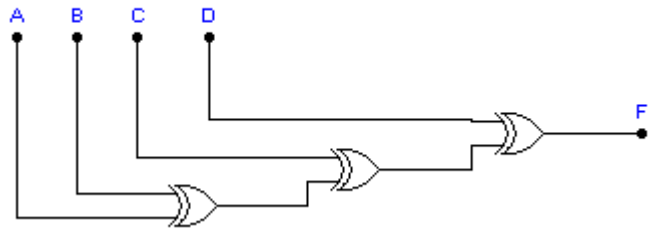
$$F = (\bar{C}\,D + C\,\bar{D}).(\bar{A}\,\bar{B} + A\,B) + (\bar{C}\,\bar{D} + C\,D)(\bar{A}\,B + A\,\bar{B})$$

$$F = (C \oplus D).\overline{(A \oplus B)} + \overline{(C \oplus D)}.(A \oplus B) \quad \text{ou}$$

$$F(A,B,C,D) = (A \oplus B) \oplus (C \oplus D) = A \oplus B \oplus C \oplus D$$

Circuitos correspondentes :





Qualquer um dos circuitos sintetiza a função em questão. O primeiro circuito, entretanto, apresenta a vantagem de utilizar o mesmo número de portas lógicas, envolvendo apenas 2 estágios, o que representa menor constante de tempo na transição de **1** para **0** ou vice-versa.

4.7 SOMADOR COMPLETO OU *FULL-ADDER*:

Projetar um somador "*Full-Adder*"

A tabela-verdade que define as relações "entradas *versus* saídas" de um somador "Full-Adder" pode ser observada abaixo:

Cin	**A**	**B**	**S**	**Cout**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

C_{in} representa o *carry* ou o transporte (vai-um) do somador anterior, que pode ser **0** ou **1**. A e B correspondem aos *bits* a serem somados; **S** e **C_{out}** correspondem à soma e ao *carry* seguinte.

Os MKs correspondentes são:

S

B \ C_{in}A	00	01	11	10
0	0 (0)	1 (2)	0 (6)	1 (4)
1	1 (1)	0 (3)	1 (7)	0 (5)

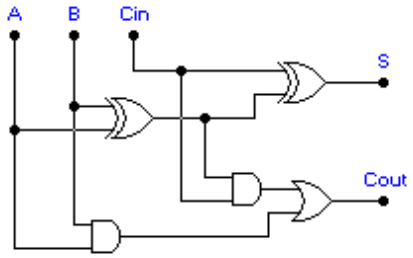
C_{out}

B \ C_{in}A	00	01	11	10
0	0 (0)	0 (2)	1 (6)	0 (4)
1	0 (1)	1 (3)	1 (7)	1 (5)

As equações resultantes, por sua vez , são:

$S = A \oplus B \oplus C_{in}$ **e**

$C_{out} = A.B + C_{in} (A \oplus B)$, correspondendo ao circuito:



Vale observar que optou-se pela simplificação da função C_{out} através das adjacências m{3 e 7}, mantendo-se os termos não adjacentes m{5 e 6}, na condição de guardarem entre sí a relação **(A ⊕ B).** Tal artifício permite a utilização desse XOR, já pertencente à equação anterior da função **S** , aproveitando-a também para a implementação da função C_{out} .

4.8 CIRCUITO COMPARADOR:

Projetar um circuito combinacional para comparar a magnitude relativa de dois números binários de 2 Bits.

Solução:

Considerando-se os números em questão como A(A_1 A_0) e B(B_1 B_0), e elaborando-se a tabela-verdade abaixo, que corresponde a uma comparação entre todas as configurações possíveis entre os 4 bits em análise, tem-se:

DEC	$A_1A_0B_1B_0$	A<B	A=B	A>B
0	0 0 0 0	0	1	0
1	0 0 0 1	1	0	0
2	0 0 1 0	1	0	0
3	0 0 1 1	1	0	0
4	0 1 0 0	0	0	1
5	0 1 0 1	0	1	0
6	0 1 1 0	1	0	0
7	0 1 1 1	1	0	0
8	1 0 0 0	0	0	1
9	1 0 0 1	0	0	1
10	1 0 1 0	0	1	0
11	1 0 1 1	1	0	0
12	1 1 0 0	0	0	1
13	1 1 0 1	0	0	1
14	1 1 1 0	0	0	1
15	1 1 1 1	0	1	0

Pela tabela-verdade acima pode-se estabelecer os mapas de Karnaugh:

A<B

B_1B_0 \ A_1A_0	00	01	11	10
00	0 (0)	0 (4)	0 (12)	0 (8)
01	1 (1)	0 (5)	0 (13)	0 (9)
11	1 (3)	1 (7)	0 (15)	1 (11)
10	1 (2)	1 (6)	0 (14)	0 (10)

A=B

B_1B_0 \ A_1A_0	00	01	11	10
00	1 (0)	0 (4)	0 (12)	0 (8)
01	0 (1)	1 (5)	0 (13)	0 (9)
11	0 (3)	0 (7)	1 (15)	0 (11)
10	0 (2)	0 (6)	0 (14)	1 (10)

A>B

B_1B_0 \ A_1A_0	00	01	11	10
00	0 (0)	1 (4)	1 (12)	1 (8)
01	0 (1)	0 (5)	1 (13)	1 (9)
11	0 (3)	0 (7)	0 (15)	0 (11)
10	0 (2)	0 (6)	1 (14)	0 (10)

De cada uma dessas funções, ou seja:

$[A<B] = \Sigma\, m(1,2,3,6,7,11);$

$[A=B] = \Sigma m(0,5,10,15)$ e $[A>B] = \Sigma\, m(4,8,9,12,13,14)$,

pode-se chegar às equações simplificadas:

$[A<B] = \overline{A}_1\overline{A}_0B_0 + \overline{A}_0B_1B_0 + \overline{A}_1B_1$

$[A=B] = (A_0 \odot B_0).(A_1 \odot B_1)$

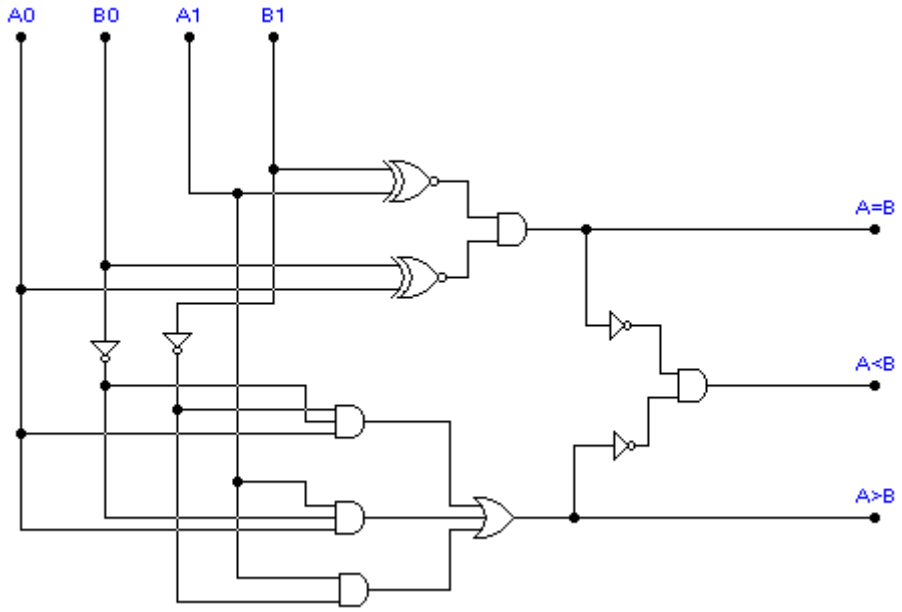
$[A>B] = A_0\overline{B}_1\overline{B}_0 + A_1A_0\overline{B}_0 + A_1\overline{B}_1$

Observa-se entretanto que as três condições são exclusivas entre sí, já que dois números só podem atender a apenas uma das hipóteses previstas no problema. Logo, uma das condições pode ser testada a partir da negação das outras duas.

Por exemplo: se A não é maior do que B e também não é igual a B, então, A é menor do que B! Ou seja, a equação para a condição [A<B] poderia ser simplesmente a equação alternativa:

$$[\,A<B\,] = \overline{[\,A=B\,]}.\overline{[\,A>B\,]}$$

Tais considerações podem resultar numa simplificação do circuito final a ser implementado, como se torna evidente no diagrama abaixo:



4.9 CIRCUITO GERADOR DE BIT DE PARIDADE:

Projetar um circuito gerador e detetor de paridade par e ímpar para uma palavra de 4 bits.

O *bit* de paridade é um dígito binário de teste que é adicionado à palavra ou informação, o qual, através de adequada codificação, pode possibilitar a deteção de um erro numa informação transmitida.

Diz-se que o código gerado é do tipo **paridade par**, quando a soma dos **1's** presentes na informação total, incluindo o bit de teste, resulta em um número par!

A recíproca é verdadeira, para o caso do código do tipo **paridade ímpar**.

Assim, tomando-se os bits A, B, C e D como relativos à informação, e P e I como os de paridade par e ímpar, a serem gerados, pode-se construir a tabela-verdade:

DEC	A B C D	P	I
0	0 0 0 0	0	1
1	0 0 0 1	1	0
2	0 0 1 0	1	0
3	0 0 1 1	0	1
4	0 1 0 0	1	0
5	0 1 0 1	0	1
6	0 1 1 0	0	1
7	0 1 1 1	1	0
8	1 0 0 0	1	0
9	1 0 0 1	0	1
10	1 0 1 0	0	1
11	1 0 1 1	1	0
12	1 1 0 0	0	1
13	1 1 0 1	1	0
14	1 1 1 0	1	0
15	1 1 1 1	0	1

A S.O.P resultante para P (paridade par) será:

$$F(A,B,C,D) = \sum m(1,2,4,7,8,11,13,14).$$

Por inspeção, a expressão para I (paridade ímpar) corresponde ao complemento da anterior, não sendo necessária a elaboração do MK correspondente. Tal fato pode ser demonstrado a partir do mapa elaborado para a saída I.

P

CD \ AB	00	01	11	10
00	0 (0)	1 (4)	0 (12)	1 (8)
01	1 (1)	0 (5)	1 (13)	0 (9)
11	0 (3)	1 (7)	0 (15)	1 (11)
10	1 (2)	0 (6)	1 (14)	0 (10)

I

CD \ AB	00	01	11	10
00	1 (0)	0 (4)	1 (12)	0 (8)
01	0 (1)	1 (5)	0 (13)	1 (9)
11	1 (3)	0 (7)	1 (15)	0 (11)
10	0 (2)	1 (6)	0 (14)	1 (10)

Os Mapas de Karnaugh acima indicam a inexistência de termos adjacentes.

Daí, as expressões para P e I são:

$$P(A,B,C,D) = (B \oplus D)AC + (B \oplus D)\overline{A}\,\overline{C} + \overline{(B \oplus D)}A\overline{C} + \overline{(B \oplus D)}\,\overline{A}C$$

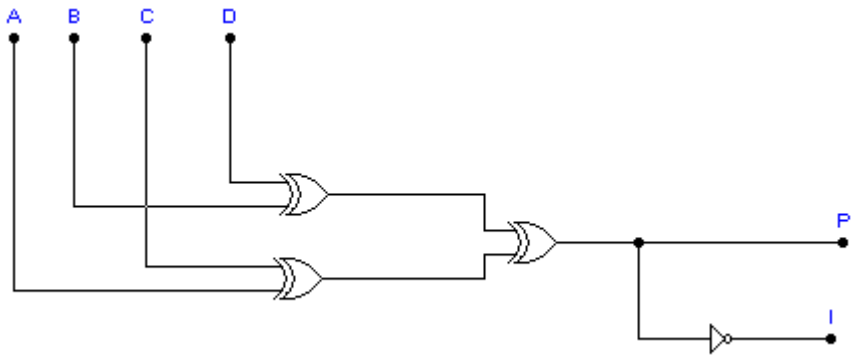
$$P(A,B,C,D) = (B \oplus D).\overline{(A \oplus C)} + \overline{(B \oplus D)}.(A \oplus C)$$

$$P(A,B,C,D) = (B \oplus D) \oplus (A \oplus C)$$

Evidentemente,

$$I(A,B,C,D) = \overline{P(A,B,C,D)}$$

O circuito corresponde a:



4.10 CONVERSOR DE CÓDIGO BCD/ XS3:

Projetar um coversor do Código BCD para o Código Excesso de Três.

Como se sabe (vide Capítulo I), o Código Excesso de Três é gerado somando-se 3 ao código BCD, como indicado na tabela-verdade logo a seguir. Considerando-se que o BCD tem a sua representação válida apenas para os dígitos do sistema decimal (0 a 9), as configurações que correspondem a 10, 11, 12, 13, 14 e 15 não têm uma representação simples em $E_3E_2E_1E_0$. Essas configurações são compostas e representadas dígito a dígito. Por essa razão, as saídas $E_3E_2E_1E_0$, na tabela-verdade, assumem estados considerados indiferentes. Esses estados são representados por (**x**), (**d**) ou (∅), tanto na tabela-verdade quanto nos mapas de Karnaugh.

A tabela-verdade e o correspondente Mapa de Karnough, neste caso, são construídos a partir da definição do código em questão, conforme consta no Capítulo I:

DEC	B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	x	x	x	x
11	1	0	1	1	x	x	x	x
12	1	1	0	0	x	x	x	x
13	1	1	0	1	x	x	x	x
14	1	1	1	0	x	x	x	x
15	1	1	1	1	x	x	x	x

Os Mk's para o conversor são:`

E_3

B_1B_0 \ B_3B_2	00	01	11	10
00	0 (0)	0 (4)	x (12)	1 (8)
01	0 (1)	1 (5)	x (13)	1 (9)
11	0 (3)	1 (7)	x (15)	x (11)
10	0 (2)	1 (6)	x (14)	x (10)

E_2

B_1B_0 \ B_3B_2	00	01	11	10
00	0 (0)	1 (4)	x (12)	0 (8)
01	1 (1)	0 (5)	x (13)	1 (9)
11	1 (3)	0 (7)	x (15)	x (11)
10	1 (2)	0 (6)	x (14)	x (10)

E_1

B_1B_0 \ B_3B_2	00	01	11	10
00	1 (0)	1 (4)	x (12)	1 (8)
01	0 (1)	0 (5)	x (13)	0 (9)
11	1 (3)	1 (7)	x (15)	x (11)
10	0 (2)	0 (6)	x (14)	x (10)

E_0

B_1B_0 \ B_3B_2	00	01	11	10
00	1 (0)	1 (4)	x (12)	1 (8)
01	0 (1)	0 (5)	x (13)	0 (9)
11	0 (3)	0 (7)	x (15)	x (11)
10	1 (2)	1 (6)	x (14)	x (10)

Os estados indiferentes, quando representados nos mapas de Karnaugh, podem ser usados para efeito de simplificação das funções, desde que estejam em posições adjacentes estratégicas.

Tais estados podem ser considerados indistintamente como **0** ou **1**, de acordo com a conveniência ocasionada pela adjacência eventualmente produzida.

Observa-se neste exemplo que no MK(E_3), todos os estados indiferentes podem ser considerados como **1**, possibilitando as simplificações decorrentes dos termos mínimos adjacentes:

(m_8, m_9, m_{10}, m_{11}, m_{12}, m_{13}, m_{14} e m_{15}), resultando em (B_3);

(m_5, m_7, m_{13},e m_{15}), resultando em ($B_2.B_0$) e

(m_6, m_7, m_{14}, e m_{15}), resultando em ($B_2.B_1$)

Ou seja,

$$\mathbf{E_3 = B_3 + B_2B_0 + B_2B_1.}$$

Similarmente, no MK(E2), os termos indiferentes m_{10}, m_{11} e m_{12} podem ser considerados como **1** e os demais como **0**, possibilitando as simplificações entre as adjacências:

(m_4 e m_{12}), resultando em $B_2.\overline{B}_1.\overline{B}_0$;

(m_1, m_3, m_9, m_{11}), resultando em $\overline{B}_2.B_0$ e

(m_2, m_3, m_{10} e m_{11}), resultando em $\overline{B}_2.B_1$.

Assim,

$$E_2 = B_2.\overline{B}_1.\overline{B}_0 + \overline{B}_2.B_0 + \overline{B}_2.B_1$$

No MK(E_1), os termos indiferentes m_{11}, m_{12} e m_{15} podem ser considerados como **1** e os demais como **0**, possibilitando as simplificações:

(m_0, m_4, m_8, m_{12}), resultando em $\overline{B}_1.\overline{B}_0$

(m_3, m_7, m_{11}, m_{15}), correspondendo a $B_1.B_0$. ou,

$$\mathbf{E_1 = \overline{B}_1.\overline{B}_0 + B_1.B_0 = B_1 \odot B_0.}$$

Quanto ao MK(E_0), considerando-se os estados indiferentes m_{10}, m_{12}, e m_{14} como lógico **1** e os demais como **0**, tem-se:

$$\mathbf{E_0 = \overline{B}_0}$$

Determinadas considerações e equivalências podem contribuir para a otimização da implementação do circuito resultante, como mostrado a seguir.

$E_3 = B_3 + B_2.(B_1 + B_0)$

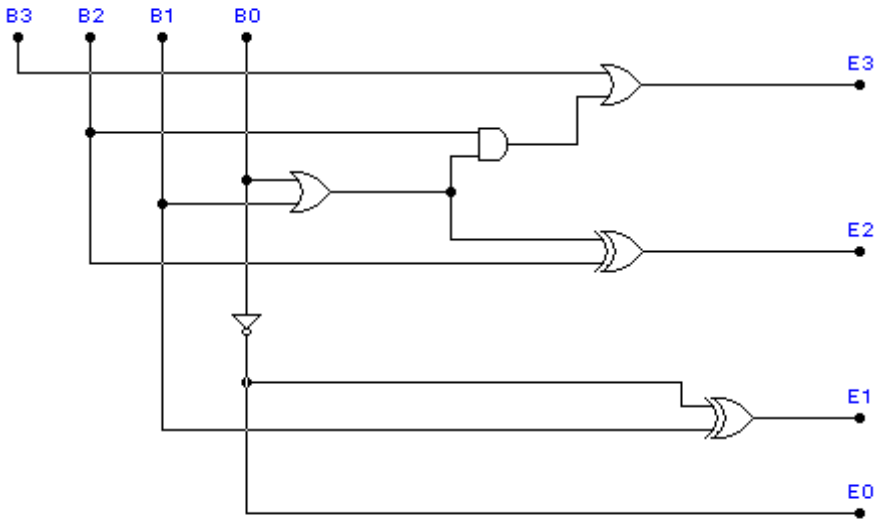
$E_2 = B_2(\overline{B}_1.\overline{B}_0) + \overline{B}_2(B_1 + B_0) = B_2(\overline{B_1 + B_0}) + \overline{B}_2(B_1 + B_0)$

ou $E_2 = B_2 \oplus (B_1 + B_0)$

$E_1 = \overline{B}_1.\overline{B}_0 + B_1.B_0 = B_1 \odot B_0 = B_1 \oplus \overline{B}_0$

$E_0 = \overline{B}_0.$

O circuito correspondente é mostrado abaixo:



4.11 SELECIONADOR - SOMADOR/SUBTRATOR:

Projetar um circuito selecionador, de tal sorte que se um dado bit de seleção K for igual a **0**, o circuito combinacional deverá se comportar como um somador *Full-Adder*, entre os bits $\mathbf{C_{IN}}$, **A** e **B**, para as saídas $\mathbf{S_1}$ e $\mathbf{C_{1OUT}}$. No caso de a variável K vir a assumir o valor lógico **1**, as saídas em questão deverão apresentar, respectivamente, a subtração simples entre A e B, e ao transporte da operação de subtração.

Daí, a tabela verdade que corresponde às operações mencionadas:

DEC	K	C_{IN}	A	B	S_1	C_{1OUT}
0	0	0	0	0	0	0
1	0	0	0	1	1	0
2	0	0	1	0	1	0
3	0	0	1	1	0	1
4	0	1	0	0	1	0
5	0	1	0	1	0	1
6	0	1	1	0	0	1
7	0	1	1	1	1	1
8	1	0	0	0	0	0
9	1	0	0	1	1	1
10	1	0	1	0	1	0
11	1	0	1	1	0	0
12	1	1	0	0	0	0
13	1	1	0	1	1	1
14	1	1	1	0	1	0
15	1	1	1	1	0	0

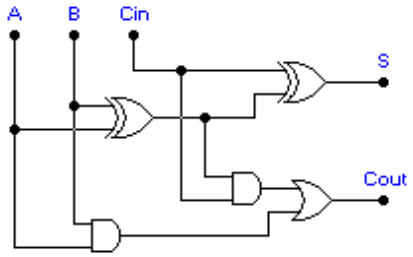
Logo,

$S(K,C_{IN},A,B) = \Sigma\, m(1,2,4,7,9,10,13,14)$

$C_{OUT}(K,C_{IN},A,B) = \Sigma\, m(3,5,6,7,9,13)$

A partir das expressões acima pode-se construir os Mapas de Karnaugh para as funções $S(K,C_{IN},A,B)$ e $C_{OUT}(K,C_{IN},A,B)$, conforme os procedimentos anteriormente mostrados, chegando-se então às minimizações pertinentes. Contudo, este problema, aparentemente de maior complexidade, pode ser resolvido por mera inspeção, desde que se observe o seguinte:

O texto do problema diz que para **K=0** a função deve ser um somador completo. Assim, sabendo-se que as equações e o circuito para o somador completo são aquelas definidas no ítem IV.7: $S = A \oplus B \oplus C_{in}$ e $C_{out} = A.B + C_{in} (A \oplus B)$, correspondendo ao circuito:



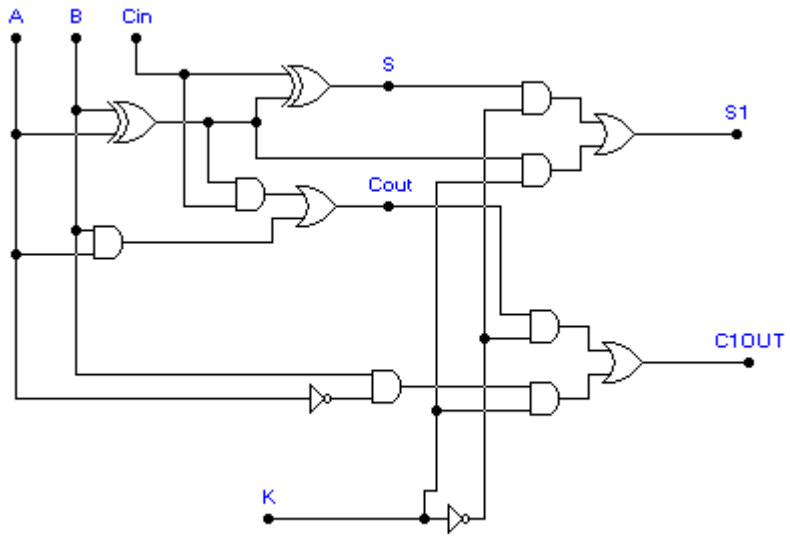
Acrescentando-se, para a primeira condição, a variável K:

$$S_1 = \{A \oplus B \oplus C_{in}\}.\overline{K}... \quad e \quad C_{1out} = \{A.B + C_{in} (A \oplus B)\}.\overline{K}...$$

Como na outra condição, para **K=1,** as funções $\mathbf{S_1}$ e $\mathbf{C_{1out}}$ deverão apresentar o resultado da subtração simples entre as variáveis **A** e **B**, definidas pela tabela verdade, basta reconhecer que a tabela verdade para $\mathbf{S_1}$ quando **K=1**, equivale ao **XOR** entre **A** e **B**, enquanto $\mathbf{C_{1out}}$ equivale à função **AND** entre **A**' e **B**. Esta segunda condição deverá então fazer parte da equação geral, através da inclusão dessa alternativa nas expressões de $\mathbf{S_1}$ e $\mathbf{C_{1out}}$, definidas acima. Logo,

$$\mathbf{S_1 = \{A \oplus B \oplus C_{in}\}.\overline{K} + \{A \oplus B\}.K \quad e}$$

$$\mathbf{C_{1out} = \{A.B + C_{in} (A \oplus B)\}.\overline{K} + \{\overline{A.B}\}.K \quad ou}$$



O circuito apresentado admite tais funções como as adequadas para a implementação, aproveitando-se o circuito anteriormente encontrado. Vale observar que o circuito anexado às saídas S e Cout do circuito anterior (IV.7), atua como mero selecionador das funções que deverão ser efetuadas entre as variáveis de entrada, tendo a variável K como variável de controle ou seleção. Circuitos selecionadores como esse são de grande aplicação nos sistemas lógicos e digitais.

A solução desse problema, via minimização por MK's, certamente traria circuitos combinacionais mais simples, mas o tipo de solução aquí utilizada, ilustra a aplicação de uma estrutura selecionadora de operações lógicas.

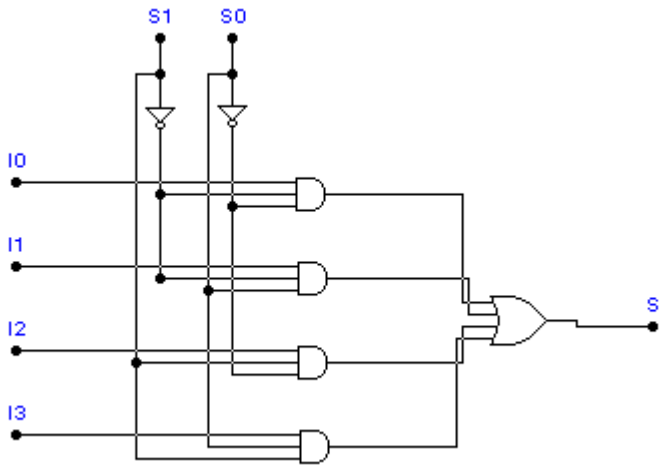
4.12 MULTIPLEXADOR/DEMULTIPLEXADOR:

Projetar um circuito selecionador que obedeça à tabela-verdade abaixo, onde $\mathbf{S_1}$ e $\mathbf{S_0}$ representam determinadas variáveis de seleção que definem a função F como assumindo o valor lógico das variáveis ou funções I_0 ou I_1 ou I_2 ou ainda I_3, de acordo com os valores lógicos das variáveis de seleção:

S_1	S_0	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$\mathbf{F(S_1,S_0)} = \mathbf{(\overline{S}_1\ \overline{S}_0)}\ I_0 + \mathbf{(\overline{S}_1\ S_0)}\ I_1 + \mathbf{(S_1\ \overline{S}_0)}\ I_2 + \mathbf{(S_1\ S_0)}\ I_3$$

O circuito correspondente a esta equação lógica será naturalmente:



Este circuito mostra claramente que é possível selecionar-se uma determinada informação, a partir da codificação empregada para as variáveis de seleção. Caso as variáveis de seleção sejam escolhidas de uma maneira seqüencial, torna-se possível então transmitir as informações I_0, I_1, I_2, e I_3, em momentos distintos, através de apenas uma saída. A aplicação prática de um circuito como este é evidente, pois, através de um único canal ou através de um único condutor pode-se transmitir grande número de informações. Este circuito é denominado MULTIPLEXADOR (MUX), e é de enorme aplicação prática. A equação definida acima pode ser também explicitada da seguinte forma:

$$\mathbf{F(S_1,S_0)} = \mathbf{(m_0)}\, I_0 + \mathbf{(m_1)}\, I_1 + \mathbf{(m_2)}\, I_2 + \mathbf{(m_3)}\, I_3 .$$

Evidentemente, $\mathbf{m_0}$, $\mathbf{m_1}$, $\mathbf{m_2}$ e $\mathbf{m_3}$ representam os termos mínimos relativos a $\mathbf{S_1}$ e $\mathbf{S_0}$.

O fato de a equação poder ser expressa desta forma, indica outra importante aplicação do MULTIPLEXADOR:

Como cada código de $\mathbf{S_1}$ e $\mathbf{S_0}$ encontra-se associado ao correspondente termo mínimo da função, neste caso de duas variáveis, qualquer função lógica que venha a ser escrita sob a forma dos seus MINTERM's poderá ser implementada através de um multiplexador, bastando-se para isso atribuir-se os valores **1** ou **0**, às entradas I_0, I_1, I_2, e I_3, de acordo com a definição dos termos mínimos e máximos da função desejada.

Por exemplo:

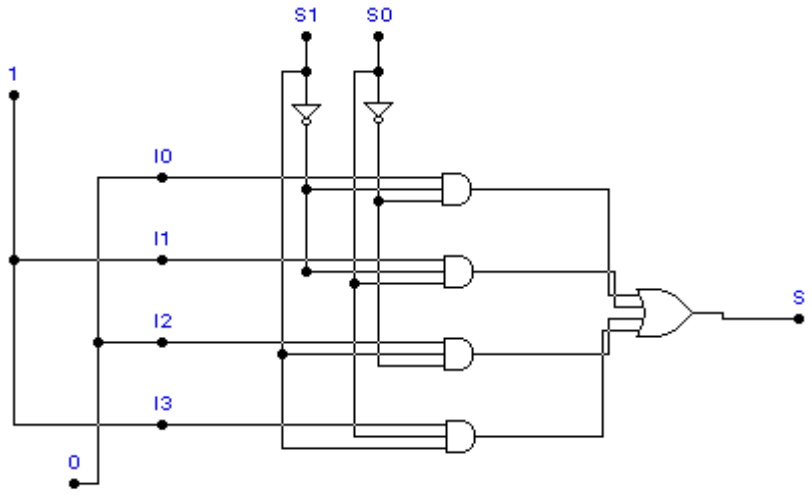
Usar o circuito acima para implementar a função: $\mathbf{F(S_1,S_0) = \Sigma\, m(1,3).}$

Esta expressão define os termos mínimos 1 e 3 como aqueles em que a função é verdadeira. Portanto, a função será falsa para os termos 0 e 2.

$\mathbf{F(S_1,S_0)} = \mathbf{(m_0)}\, I_0 + \mathbf{(m_1)}\, I_1 + \mathbf{(m_2)}\, I_2 + \mathbf{(m_3)}\, I_3$ se apresenta como:

$\mathbf{F(S_1,S_0) = (m_0)\, 0 + (m_1)\, 1 + (m_2)\, 0 + (m_3)\, 1}$

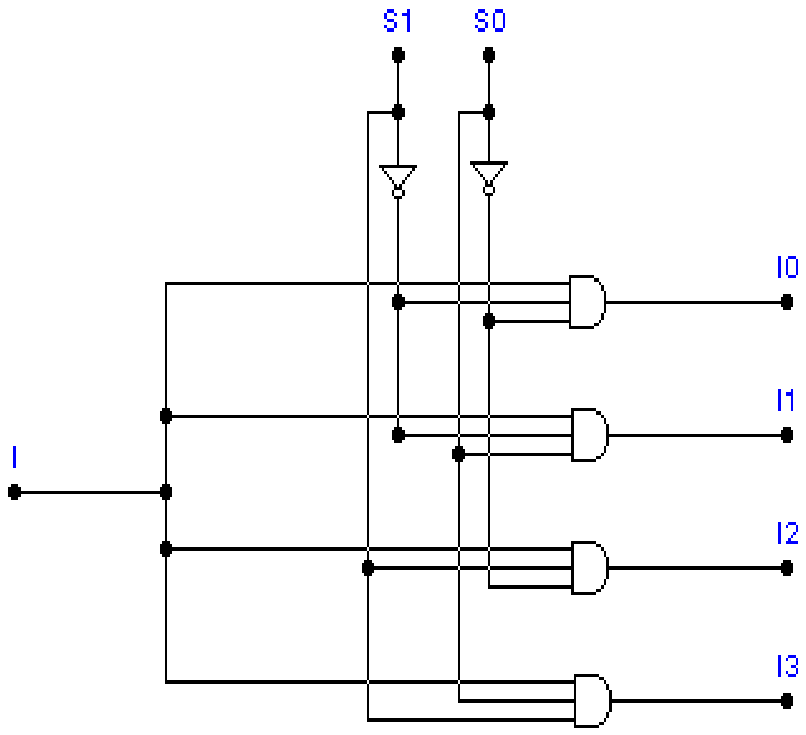
Isto implica em se conectar as entradas I_0 e I_2 a **0** e I_1 e I_3 a **1**, como indicado abaixo, para se ter o circuito implementado, sem qualquer preocupação inclusive quanto à simplificação do mesmo.



Outra vantagem extraordinária dessa estrutura é a de que a configuração lógica das diversas portas não muda para as diferentes funções. Basta que as entradas sejam ligadas a **0** ou a **1**, conforme os termos mínimos selecionados, para se ter outras funções devidamente implementadas.

Existem inúmeros tipos de MULTIPLEXADORES integrados e para grande número de variáveis de entrada, permitindo maior flexibilidade na implementação de funções lógicas combinacionais. Os manuais dos fabricantes trazem maiores detalhes e sugestões de uso para os diversos dispositivos desse tipo, os quais, apesar da simplicidade na sua lógica, permite a solução de problemas complexos de modo elegante, além de programável.

O circuito que executa a tarefa de receber as informações por um único condutor e recompor a informação completa, denomina-se DEMULTIPLEXADOR (DEMUX). Este circuito faz o trabalho inverso do anterior, apresentando a seguinte configuração:



Tal configuração é evidente por sí mesma, onde **I** representa a informação, geralmente fornecida por um MULTIPLEXADOR, para cada código $\mathbf{S_1}$, $\mathbf{S_0}$. As informações $\mathbf{I_0}$, $\mathbf{I_1}$, $\mathbf{I_2}$ e $\mathbf{I_3}$ são reconstituídas para o código específico, coincidente com o do MULTIPLEXADOR, sendo geralmente armazenadas em registros ou memórias a FLIP-FLOP, por exemplo, como será visto no próximo Capítulo.

4.13 CONVERSOR BINÁRIO/GRAY:

Projetar um conversor Binário/Gray de 4 Bits.

Pela definição do código de Gray, apresentada anteriormente, pode-se construir a tabela-verdade a seguir, onde $B_3B_2B_1B_0$ equivalem aos bits em binário e $G_3G_2G_1G_0$ aos seus correspondentes no código de Gray.

Código Binário x Gray

DEC	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

- As S.O.P.'s para cada caso são:

$G_3 = \sum m(8,9,10,11,12,13,14,15)$;

$G_2 = \sum m(4,5,6,7,8,9,10,11)$;

$G_1 = \sum m(2,3,4,5,10,11,12,13)$;

$G_0 = \sum m(1,2,5,6,9,10,13,14)$.

- Os MK's se apresentam como:

G_3

B_1B_0 \ B_3B_2	00	01	11	10
00	0 (0)	0 (4)	1 (12)	1 (8)
01	0 (1)	0 (5)	1 (13)	1 (9)
11	0 (3)	0 (7)	1 (15)	1 (11)
10	0 (2)	0 (6)	1 (14)	1 (10)

G_2

B_1B_0 \ B_3B_2	00	01	11	10
00	0 (0)	1 (4)	0 (12)	1 (8)
01	0 (1)	1 (5)	0 (13)	1 (9)
11	0 (3)	1 (7)	0 (15)	1 (11)
10	0 (2)	1 (6)	0 (14)	1 (10)

G_1

B_1B_0 \ B_3B_2	00	01	11	10
00	0 (0)	1 (4)	1 (12)	0 (8)
01	0 (1)	1 (5)	1 (13)	0 (9)
11	1 (3)	0 (7)	0 (15)	1 (11)
10	1 (2)	0 (6)	0 (14)	1 (10)

G_0

B_1B_0 \ B_3B_2	00	01	11	10
00	0 (0)	0 (4)	0 (12)	0 (8)
01	1 (1)	1 (5)	1 (13)	1 (9)
11	0 (3)	0 (7)	0 (15)	0 (11)
10	1 (2)	1 (6)	1 (14)	1 (10)

As equações resultantes são:

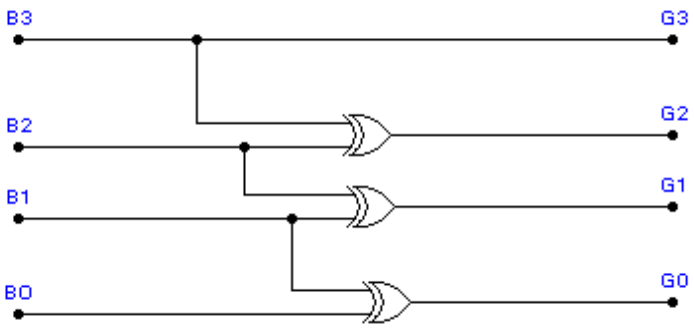
$$G_3 = B_3$$

$$G_2 = \bar{B}_3B_2 + B_3\bar{B}_2 = B_3 \oplus B_2$$

$$G_1 = \bar{B}_2B_1 + B_2\bar{B}_1 = B_2 \oplus B_1$$

$$G_0 = \bar{B}_1B_0 + B_1\bar{B}_0 = B_1 \oplus B_0$$

Para o problema específico acima, o circuito correspondente será:



Observe-se que é possível a generalização para qualquer número de bits, conforme indicado:

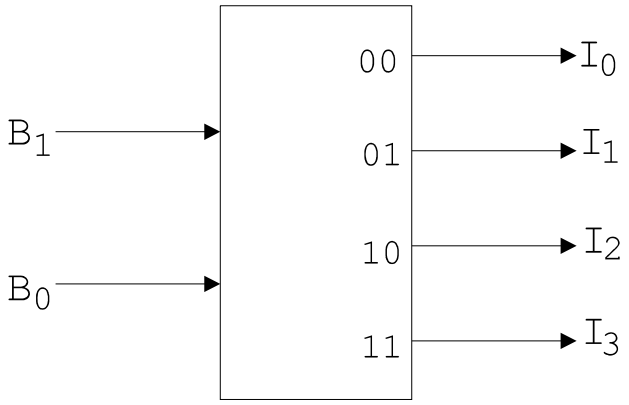
$$G_i = B_{i+1} \oplus B_i$$

4.14 DECODIFICADORES:

A exemplo das informações lógicas que deverão estar disponíveis na entrada dos multiplexadores, ou de qualquer outra estrutura como reconstituidora de linhas de atuação de circuitos de controle, ou de endereçamento de memórias e outras aplicações, os decodificadores exercem, apesar da sua simplicidade, um papel muito importante no interfaceamento de circuitos lógicos e digitais.

Por exemplo, no caso do endereçamento da informação para uma única linha, entre várias outras, ou barramento, as variáveis de entrada podem exercer o papel de selecionadoras de linha, similarmente ao que foi visto no caso dos demultiplexadores.

A filosofia de operação dos decodificadores pode ser ilustrada a partir do diagrama de blocos mostrado a seguir, onde $\mathbf{B_1}$ e $\mathbf{B_0}$ representam as variáveis de entrada, enquanto $\mathbf{I_0}$, $\mathbf{I_1}$, $\mathbf{I_2}$ e $\mathbf{I_3}$, as diversas linhas a serem energizadas, ou acessadas.



O diagrama de blocos sintetiza a situação em que apenas 2 variáveis podem gerar 4 (2^2) saídas, em consonância com as configurações das variáveis de entrada, conforme a representação indicada pela tabela-verdade abaixo:

$\mathbf{B_1B_0}$	$\mathbf{I_3}$	$\mathbf{I_2}$	$\mathbf{I_1}$	$\mathbf{I_0}$
0 0	0	0	0	1
0 1	0	0	1	0
1 0	0	1	0	0
1 1	1	0	0	0

A tabela-verdade é auto explicativa, indicando simplesmente que somente uma das saídas assume o valor lógico verdadeiro, a cada distinta configuração das variáveis de entrada.

Representando o que foi dito acima sob a forma de equações lógicas, tem-se:

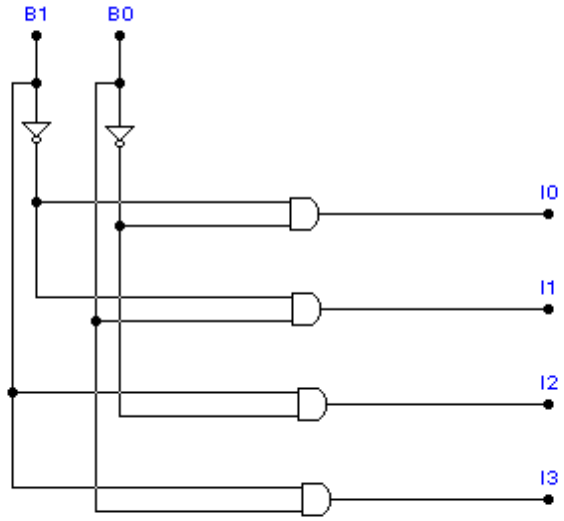
$$\mathbf{I_0 = \overline{B_1}\,\overline{B_0}}$$

$$\mathbf{I_1 = \overline{B_1}\,B_0}$$

$$\mathbf{I_2 = B_1\,\overline{B_0}}$$

$$\mathbf{I_3 = B_1\,B_0}$$

O circuito correspondente é:



Este circuito muito simples, para apenas duas variáveis de entrada, pode ser expandido para **n** variáveis, podendo então apresentar na saída até $\mathbf{2^n}$ linhas, associadas às diversas configurações.

Os decodificadores se prestam a variadas aplicações, a exemplo de geração de formas de onda, conversores BCD x 7 segmentos, geração aleatória de funções lógicas, desempenhando também um importante papel nos circuitos de endereçamento de memórias semicondutoras.

4.15 CONSIDERAÇÕES GERAIS:

Evidentemente, inúmeros outros circuitos são passíves de implementação, a exemplo de multiplexadores e demultiplexadores para grande número de entradas, codificadores e decodificadores para funções especiais e também circuitos aritméticos e outros de maior grau de complexidade. Contudo, todos os conceitos aquí apresentados se aplicam integralmente, tanto na análise quanto na síntese de circuitos lógicos, e são essenciais para permitir uma visão crítica quando da utilização de circuitos integrados interconectados entre sí, visando a implementação de soluções efetivas de problemas reais e de aplicação prática.

O fato de determinados circuitos se destinarem a aplicação mais geral, contribui para que grande variedade deles seja projetada e implementada monoliticamente, em circuitos integrados ou CI's, permitindo a sua utilização diretamente, sem a necessidade de se interconectar portas básicas do tipo SSI, ou de baixo grau de integração.

Assim, vários circuitos cujos projetos foram aqui elaborados, podem ser encontrados para pronta utilização, como pode ser visto nos manuais dos fabricantes, onde são mostrados os diagramas lógicos e de pinagem, as características elétricas gerais, além de sugestões para o seu uso e aplicações várias, ou *application notes*.

Evidentemente, qualquer projeto a nível profissional requer o conhecimento do que se encontra à disposição no mercado, características desses circuitos, além de um detido exame do problema específico a resolver, para se evitar esforços desnecessários no desenvolvimento e implementação de circuitos que já se encontrem disponíveis no mercado.

O fato de se dispor de funções mais complexas já integradas, facilita o trabalho e aumenta a confiabilidade do sistema implementado. Recomenda-se a qualquer profissional que deseje projetar circuitos lógicos deter o conhecimento das famílias de circuitos lógicos existentes, e detalhes a respeito das suas características eletro-eletrônicas e limitações.

Considerando que o presente trabalho visa unicamente dar os subsídios básicos para o entendimento da filosofia e conceituação sobre circuitos lógicos, não se apresenta aquí os dados tecnológicos propriamente ditos sobre os componentes a serem utilizados na implementação desses circuitos e sistemas.

Como deve ter sido notado, as soluções apresentadas neste capítulo privilegiaram a resolução dos problemas propostos a partir das expressões em soma de produtos (S.O.P.) mas, evidentemente, todos os problemas podem ser solucionados sob a forma de P.O.S.

O desenvolvimento de tais expressões deve ser realizado como exercício por parte do leitor, objetivando maior segurança no entendimento dos procedimentos aquí discutidos.

4. 16 EXERCíCIOS IV

1. O Código BCD (5311), é definido pela tabela-verdade indicada abaixo, em relação ao código binário natural (8421). Projetar um circuito combinacional para converter o binário representado em BCD (5311), utilizando o menor número possível de portas lógicas. Esboçar o circuito a ser implementado.

BINÁRIO	**5 3 1 1**
A B C D	W X Y Z
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 1
0 0 1 1	0 1 0 0
0 1 0 0	0 1 0 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 1
1 0 0 0	1 1 0 0
1 0 0 1	1 1 0 1

2. Um codificador de posição angular, associado ao eixo de um motor, indica a posição angular do mesmo em intervalos de 30 graus, usando o código de Gray, conforme a tabela abaixo. Projetar um circuito lógico para fazer soar um alarme quando a posição angular detectada se encontre no segundo ou no quarto quadrante.

Posição	**Código**
00-30	0 0 1 1
30-60	0 0 1 0
60-90	0 1 1 0
90-120	0 1 1 1
120-150	0 1 0 1
150-180	0 1 0 0
180-210	1 1 0 0
210-240	1 1 0 1
240-270	1 1 1 1
270-300	1 1 1 0
300-330	1 0 1 0
330-360	1 0 1 1

Deve-se assumir que as configurações inexistentes, na tabela-verdade, não colaboram no estabelecimento do sistema de codificação.

3. Projetar um circuito a 5 variáveis tal que a saída seja verdadeira quando mais de cinqüenta por cento dessas variáveis apresentem valor lógico **0**.

    - simplificar a expressão encontrada usando o mapa de Karnaugh;
    - esboçar o diagrama do circuito lógico.

4. Projetar um circuito comparador, tal que um *LED* seja ativado num painel de controle, a cada vez que o estado lógico de determinada máquina ***A*** seja equivalente ao da máquina ***D*** , e o estado da máquina ***B*** diferente do da máquina ***C***.

5. Projetar um circuito lógico para ligar um motor para bombeamento de combustível de um reservatório inferior para outro superior. A bomba será controlada por quatro detetores de nível que acionam respectivamente 4 variáveis lógicas a saber:

    - reservatório inferior vazio;
    - reservatório inferior cheio;
    - reservatório superior vazio;
    - reservatório superior cheio.

O motor deverá partir sempre que o reservatório superior estiver vazio e o inferior não estiver vazio, ou quando o reservatório inferior estiver cheio e o superior não estiver cheio. Prever um circuito de alarme, para indicar eventuais condições de mau funcionamento dos detetores de nível do sistema.

6. Em determinado computador, as letras do alfabeto e os dígitos do sistema decimal são codificados em termos dos números octais:

Letras:
A=30; B=23; C=16; D=22; E=20; F=26; G=13; H=05; I=14;
J=32; K=36; L=11; M=07; N=06; O=03; P=15; Q=35;
R=12; S=24; T=01; U=34; V=17; W=31; X=27; Y=25; Z=21.
Números:
0=37; 1=52; 2=74; 3=70; 4=64; 5=62; 6=66; 7=72; 8=60 9=33

Projetar um circuito para receber esse código na entrada, e apresentar uma saída falsa a cada vez que a informação de entrada corresponder a uma letra do

alfabeto, e uma saída verdadeira, quando a informação for numérica. Observar que os demais códigos, pertencentes ao sistema, se referem a símbolos especiais, cujas configurações não serão utilizadas pelo circuito combinacional em questão.

7. Em determinado computador, as letras do alfabeto e os dígitos do sistema decimal são codificados em termos dos números hexadecimais, equivalentes à configuração binária, conforme demonstra a tabela abaixo:

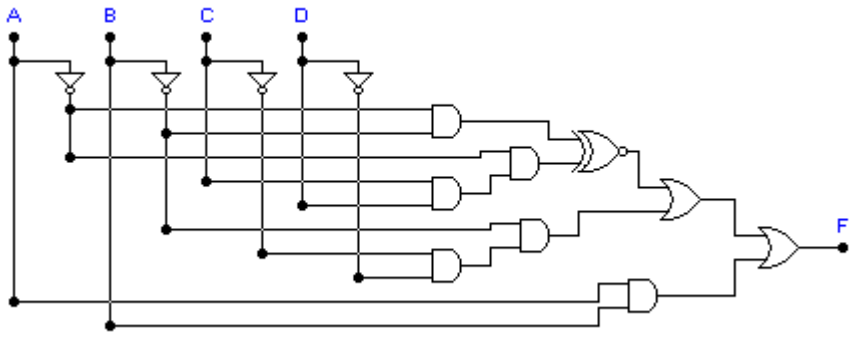
Letras:
A=18; B=13; C=0E; D=12; E=10; F=16; G=0B; H=05; I=0C;
J=1A; K=1E; L=09; M=07; N=06; O=03; P=0D; Q=1D;
R=0A; S=14; T=01; U=1C; V=0F; W=19; X=17; Y=15; Z=11
Números:
0=1F; 1=2A; 2=3C; 3=38; 4=34; 5=32; 6=36; 7=3A; 8=30; 9=1B.

Projetar um circuito para receber tal código na entrada e apresentar uma saída verdadeira, a cada vez que a informação de entrada corresponder a uma letra do alfabeto, e uma saída falsa, quando a informação for numérica.

Observar que os demais códigos, pertencentes ao sistema, se referem a símbolos especiais, cujas configurações não estarão disponíveis na entrada do circuito combinacional em questão.

8. Projetar um circuito de controle para desarmar um disjuntor de proteção, quando ocorrer a energização de uma prensa hidráulica, ao mesmo tempo em que um sensor indique a presença de um operador, e determinada esteira rolante se encontre funcionando, enquanto o sistema de empacotamento automático de determinado produto se encontre desligado.

9. Para o circuito abaixo, encontrar a tabela-verdade e a equação lógica em Soma de Produtos e Produto de Somas. Implementar o mesmo circuito através de um multiplexador.



5 FLIP-FLOP's

Os Flip-Flop's são dispositivos especiais que permitem armazenar uma dada informação do sistema binário e, por essa razão, são também conhecidos como *elementos de memória* ou *células de memória.*

Para descrever o comportamento das funções elementares dos circuitos combinacionais, estudados anteriormente, as tabelas-verdade representavam uma fonte de informações que permitiam o estabelecimento das equações lógicas. Devido aos aspectos dinâmicos do funcionamento dos FLIP-FLOP's, um outro tipo de representação se torna mais indicado e é designado como *diagrama de estados*.

Os diagramas de estados permitem uma descrição bastante compacta das transições (mudança de estado lógico de 0 para 1, ou vice-versa) que são realizadas pelos FLIP-FLOP's, em função das variáveis de entrada e de saída, além de facilitarem a elaboração de uma tabela de estados, ou transição, a qual caracteriza a dinâmica do circuito.

Os FLIP-FLOP's são geralmente comandados por trens de pulsos (*clock*), que definem com que freqüência as ações ou transições devem ocorrer, para que uma determinada seqüência de operações seja executada.

Cada Flip-Flop é definido através de um *diagrama de estados* que dá origem a uma tabela-verdade denominada *Tabela Característica*, da qual resultam outras duas, conhecidas como *Tabela de Estados (*ou *Tabela de Transição)* e *Tabela de Excitação.*

A Tabela Característica explicita o que sucede na saída do dispositivo, em função das variáveis de entrada, após a ocorrência de um pulso de comando ou também denominado pulso *clock*.

A Tabela de Estados ou Transição é uma versão expandida da Tabela Característica, constando o estado lógico do momento da observação (ou atual) em que o dispositivo se encontra, além das variáveis de entrada nesse mesmo momento, e a determinação do estado lógico seguinte da saída, em função das mencionadas variáveis.

A Tabela de Excitação, por sua vez, determina quais são os valores lógicos necessários na entrada dos Flip-Flop's para se programar uma transição desejada no estado lógico da saída.

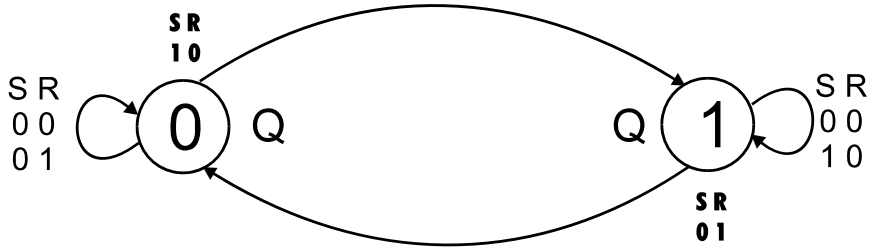
A saída desse dispositivo é geralmente indicada na literatura técnica e nos manuais dos fabricantes pela letra Q. Tal saída, num determinado momento de observação, é identificada por Q_n e, no momento seguinte, por $Q_{n+1.}$ Vale a pena registrar que as saídas Q_n e Q_{n+1} são a mesma saída física, em momentos distintos de observação.
Quando tal saída se apresenta no estado lógico 1, diz-se que o Flip-Flop encontra-se *setado* ou em estado *SET. No* caso oposto (estado lógico 0), diz-se que o mesmo encontra-se *resetado* ou em estado *RESET*.

Os Flip-Flop's mais utilizados são os denominados SR, JK, T e D.

A seguir, apresentam-se os mencionados dispositivos, com os seus respectivos diagramas de estado, tabelas-verdade e as suas correspondentes equações lógicas.

5.1 FLIP-FLOP SR:

5.1.1 Diagrama de Estados FF-SR:



O diagrama de estados mostrado acima indica esquematicamente o que ocorre com a saída Q do Flip-Flop SR (ou RS) na presença dos valores lógicos das entradas S e R, dependendo do estado em que a própria saída se encontra. Por exemplo: se o dispositivo se encontrar no estado lógico 0, num dado momento de observação, caso as entradas SR estejam, respectivamente, com os valores 00 ou 01, o dispositivo será mantido no estado em que se encontra (0); caso as entradas apresentem os valores lógicos SR = 10, o Flip-Flop deverá fazer a transição do estado lógico 0 para o estado 1: ($0 \rightarrow 1$). Este diagrama de estados também indica que, no caso de o Flip-Flop se encontrar no estado lógico 1, se as entradas S e R apresentarem valores lógicos 00 ou 10, o Flip-Flop continuará no estado em que se encontra (1); entretanto, se as entradas assumirem os valores SR = 01,

o Flip-Flop mudará do estado lógico 1 para o estado 0: ($1 \rightarrow 0$). Essas mudanças de estado são denominadas *transição*. Evidentemente, o diagrama de estados fornece os elementos essenciais para que as diversas tabelas-verdade sejam construídas.

5.1.2 Tabela-Característica FF-SR:

Como mencionado anteriormente, esta tabela-verdade descreve o que ocorre na saída Q do dispositivo, no momento de observação (n+1) - após a ocorrência de um dos pulsos de comando (*clock*) -, em função dos valores lógicos das entradas S e R, no momento de observação (n).

Assim,

S_n	R_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	x

Em consonância com o Diagrama de Estados, esta tabela indica que o estado seguinte de observação da saída (Q_{n+1}) corresponde ao estado atual de observação (Q_n), quando S_nR_n= 00; que ocorre *reset* para S_nR_n=01 e *set* para S_nR_n=10.

O estado indiferente (x), representado na tabela-verdade, decorre do fato de que neste dispositivo não se permite que as entradas S e R apresentem, simultaneamente, níveis lógicos verdadeiros. Tal situação, devido à própria concepção lógica deste Flip-Flop, resultaria numa ambigüidade pois, tais entradas estariam sendo apresentadas de sorte que o dispositivo teria que *setar* e *resetar* simultaneamente. Na prática, esta situação provocará uma instabilidade no circuito eletrônico, não sendo possível prever-se o estado lógico da saída Q_{n+1}, quando R=S=1. Por este motivo, o projetista deverá evitar que as entradas S e R assumam, concomitantemente, o valor lógico 1.

5.1.3 Tabela de Estados FF-SR:

Esta tabela-verdade, que nada mais é do que uma expansão da tabela-característica, registra qual será o estado lógico previsto para a saída no momento seguinte ao de observação (Q_{n+1}), em função dos estados lógicos das entradas e da própria saída, no momento atual de observação (n).

Tabela de Estados ou Transição FF-SR:

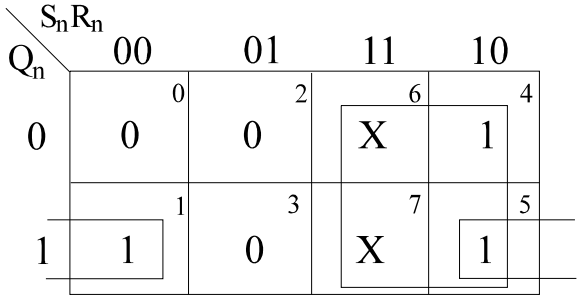
DEC	S_n	R_n	Q_n	Q_{n+1}
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	x
7	1	1	1	x

Observação: *DEC*, na tabela, corresponde aos MINTERM's ou aos valores em decimal equivalentes à configuração em binário.
Conforme as técnicas de abordagem apresentadas no capítulo anterior, a equação que define a função a partir da tabela-verdade acima pode ser escrita como:

$$\mathbf{Q_{n+1} = \sum m\,(1,4,5) + d\,(6,7)}$$

O termo d(6,7), define os MINTERM's indiferentes pertencentes à função. Como se sabe, tais valores lógicos representados no Mapa de Karnaugh da função, podem ser úteis na simplificação da equação a ser implementada.

Neste caso específico, tem-se o seguinte Mapa de Karnaugh:



Logo, a equação simplificada através dos MINTERM's assinalados se apresenta como:

$$\mathbf{Q_{n+1} = S_n + Q_n \overline{R}_n}$$

Para SR ≠ 11

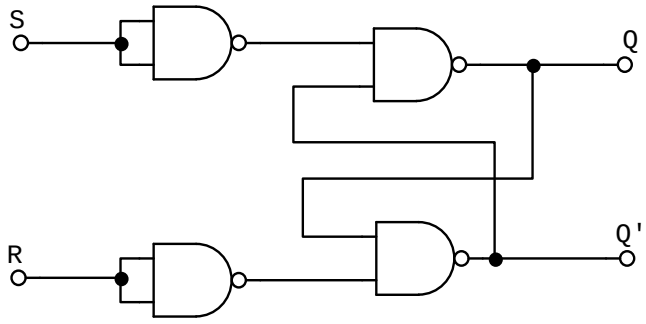
Por uma questão de conveniência de implementação, a partir da utilização de apenas um tipo de porta lógica, esta equação pode ser representada por portas **NAND** ou por portas **NOR**, indiferentemente.
Caso deseje-se implementar esta função a partir de portas do tipo **NAND**, a equação acima, após manipulações algébricas bastante óbvias, assume a seguinte configuração:

$$\overline{Q}_{n+1} = S_n + Q_n.\overline{R}_n$$

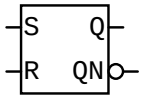
Daí, pelo teorema de De Morgan,

$$Q_{n+1} = \overline{(\overline{S}_n)} \; . \; \overline{(Q_n.\overline{R}_n)}$$

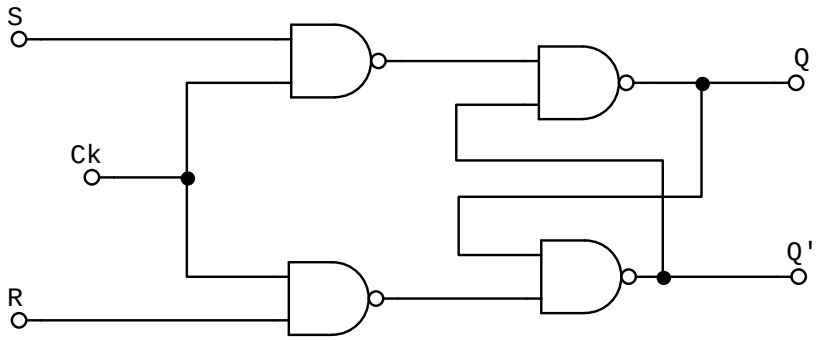
O circuito resultante utilizando somente portas lógicas do tipo **NAND**, é:



Como pode ser mostrado através da tabela de estados ou transição, todas as vezes que a saída **Q** assume um dado valor lógico (no momento **n+1**), a saída do **NAND** associada a **Q'**, apresenta o valor lógico complementar ao da saída **Q**. Por uma questão de praticidade e flexibilidade na utilização dos Flip-Flops, as saídas complementares quase sempre também se encontram disponíveis nos Circuitos Integrados (CI´s) como parte integrante dos circuitos lógicos integrados. O circuito mostrado acima é geralmente denominado *latch* e se constitui na célula básica de memória ou no Flip-Flop essencial, sendo representado em diagramas lógicos pelo bloco indicado abaixo:



Do ponto de vista conceitual, o pulso *clock* (*Cp* ou *Ck*) que comanda o dispositivo pode ser implementado pela anexação de circuitos do tipo AND ao circuito anterior, resultando na configuração abaixo:



Vale observar que os valores lógicos das entradas **S** e **R** são comunicados ou não à célula básica de memória, dependendo do nível lógico da entrada Ck, que representa o pulso *clock*. A equação originalmente apresentada pode ser implementada também a partir de portas lógicas do tipo **NOR**, conforme as manipulações algébricas a seguir:

Sendo

$$Q_{n+1} = S_n + Q_n \overline{R}_n$$

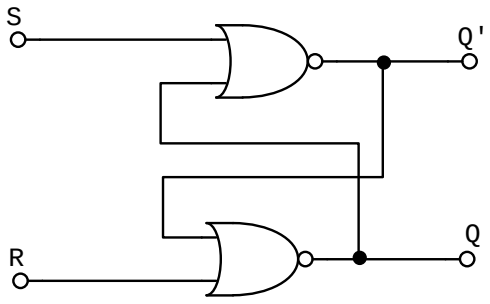
Logo, pelo teorema de De Morgan,

$$Q_{n+1} = S_n + \overline{(\overline{Q}_n + R_n)}$$

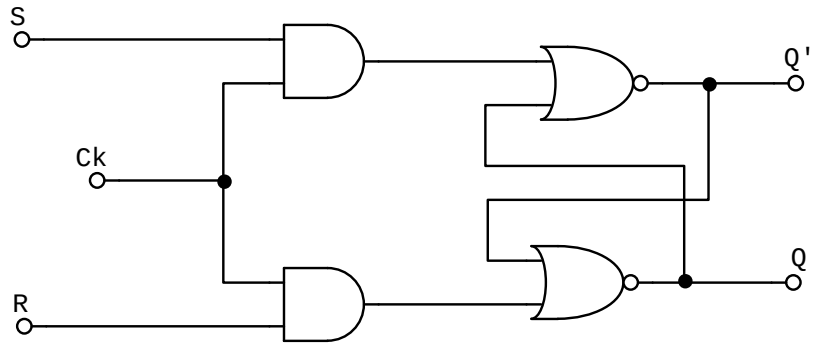
Daí,

$$\overline{Q}_{n+1} = \overline{(S_n)} + \overline{(\overline{Q}_n + R_n)}$$

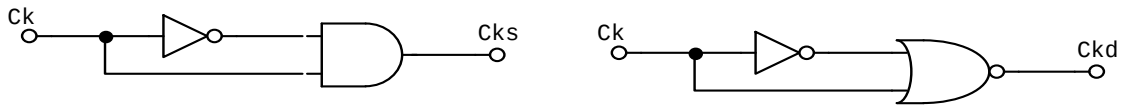
Esta expressão pode ser implementada, então, pelo circuito básico:



A inclusão do circuito que permite a atuação do *clock,* resulta na configuração:



Nos Flip-Flop's para uso profissional, circuitos detectores de subida ou descida de pulsos são anexados, a exemplo dos circuitos indicados abaixo:



Estes circuitos fazem com que o pulso *clock* permita a atuação da célula de memória na subida (do nível lógico 0 para o nível lógico 1) ou na descida do mesmo (do nível lógico 1 para o nível 0). Este artifício permite que as transições de um estado lógico para outro possam ocorrer com maior velocidade.

5.1.4 Tabela de Excitação FF-SR:

A Tabela de Excitação desse Flip-Flop é representada por:

$Q_n \rightarrow Q_{n+1}$	S_n R_n
0 → 0	0 x
0 → 1	1 0
1 → 0	0 1
1 → 1	x 0

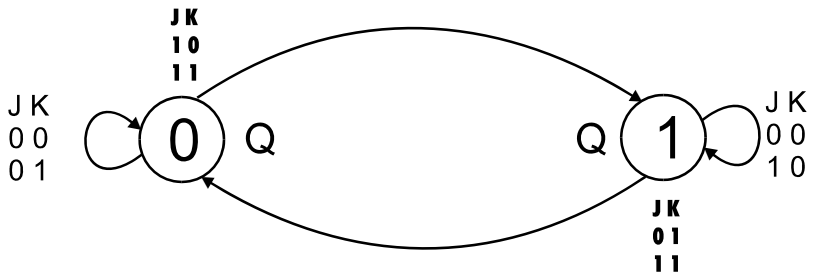
Naturalmente, esta tabela-verdade define os estados lógicos necessários nas entradas do **FF-SR** para que as transições aí representadas possam ocorrer.
Por exemplo: se o dispositivo estiver no estado lógico **0** (**no momento de observação n**) e se deseja, após a ocorrência do pulso *clock*, que o mesmo se

mantenha nesse mesmo estado lógico, torna-se necessário que a entrada **S**, no momento de observação **n**, seja **0** (para não *setar* o dispositivo), mas a entrada **R**, no mesmo momento de observação, pode apresentar qualquer valor lógico pois, em sendo nível lógico **0**, a tabela característica diz que o estado seguinte é sempre igual ao estado anterior e, em sendo nível lógico **1**, ocorrerá um *reset* do dispositivo. Considerando que o mesmo já se encontra *resetado*, o valor lógico dessa entrada será *irrelevante* para a obtenção da operação pretendida. Do mesmo modo, analisando a situação para o caso de o dispositivo se encontrar no estado lógico **0**, e vir a fazer a sua transição para o estado **1**, necessário se torna, no momento de observação **n**, que as entradas **S** e **R** sejam respectivamente **1** e **0**, e assim por diante (***vide Tabela Característica***).

5.2 FLIP-FLOP JK

A exemplo do Flip-Flop **SR**, o **FF-JK** apresenta o diagrama de estados, as tabelas e as equações lógicas, como indicados a seguir:

5.2.1 Diagrama de Estados FF-JK:



Como no caso anterior, o diagrama de estados indica o que ocorre com a saída Q do Flip-Flop JK na presença dos valores lógicos das entradas J e K, dependendo do estado em que a saída se encontra. Em outras palavras, se o dispositivo se encontrar no estado lógico 0, num dado momento de observação, caso as entradas JK estejam, respectivamente, com os valores 00 ou 01, o dispositivo deverá ser mantido no estado em que se encontra (0); caso as entradas apresentem os valores lógicos JK = 10 ou 11, o Flip-Flop deverá fazer a transição do estado lógico 0 para o estado 1 ($0 \rightarrow 1$). Este diagrama de estados também indica que, no caso de o Flip-Flop se encontrar no estado lógico 1, se as entradas S e R apresentarem valores lógicos 00 ou 10, o Flip-Flop continuará no estado

em que se encontra (1); entretanto, se as entradas assumirem os valores JK = 01 ou 11, o Flip-Flop mudará do estado lógico 1 para o estado 0 ($1 \rightarrow 0$).
Como se sabe, o Diagrama de Estados fornece os elementos essenciais para que as diversas tabelas-verdade sejam construídas. Vale ressaltar que o Flip-Flop JK permite que as entradas assumam simultaneamente o estado lógico 1, apresentando como estado seguinte (Q_{n+1}), o complemento do estado atual de observação ($\overline{Q}n$), não havendo portanto a restrição estabelecida no caso do FF-SR.

5.2.2 Tabela-Característica FF-JK:

Esta Tabela-Característica difere da anterior (FF-SR), apenas pela condição das entradas poderem apresentar simultaneamente o estado lógico 1, sendo o comportamento das entradas J e K similar ao das entradas S e R, respectivamente. Assim,

J_n K_n	Q_{n+1}
0 0	Q_n
0 1	0
1 0	1
1 1	$\overline{Q}n$

5.2.3 Tabela de Estados FF-JK:

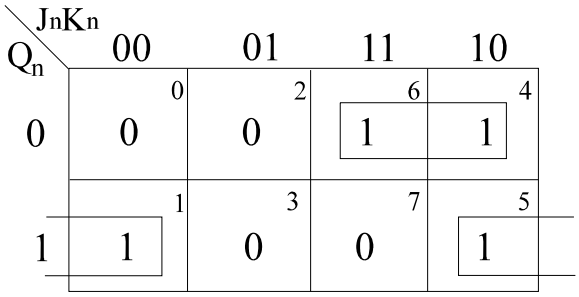
Ainda como no caso anterior, expandindo-se a Tabela-Característica, chega-se à Tabela de Estados ou de Transição a seguir, onde as entradas J e K podem assumir concomitantemente o valor lógico 1 , resultando no complemento do estado lógico Qn.

DEC	J_n	K_n	Q_n	Q_{n+1}
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Da Tabela de Estados ou de Transição acima, define-se a função:

$$\mathbf{Q_{n+1} = \sum m(1,4,5,6).}$$

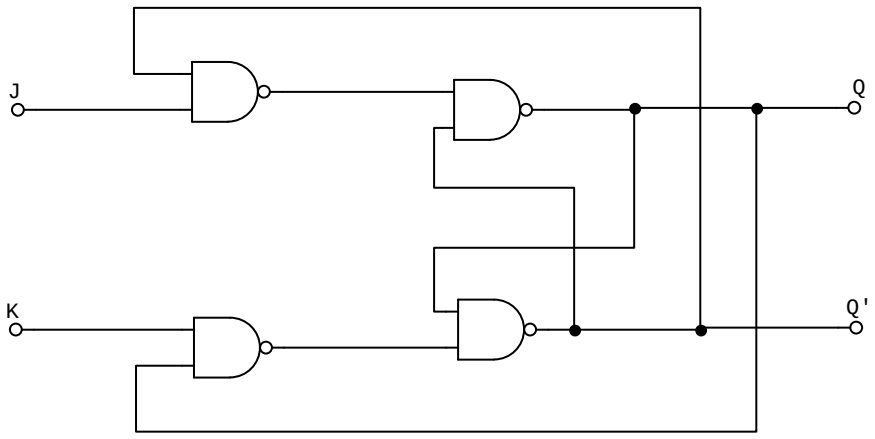
O Mapa de Karnaugh pode então ser construído:



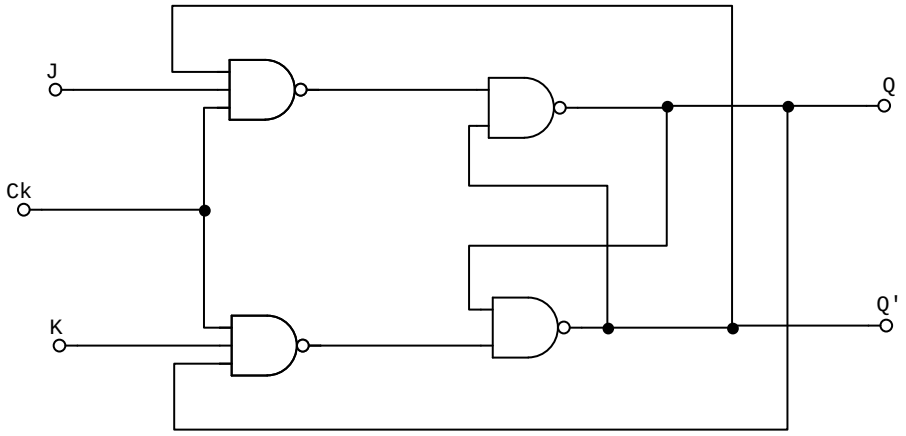
Pelas adjacências indicadas, chega-se então à expressão:

$$\mathbf{Q_{n+1} = J_n\overline{Q}_n + \overline{K}_nQ_n}$$

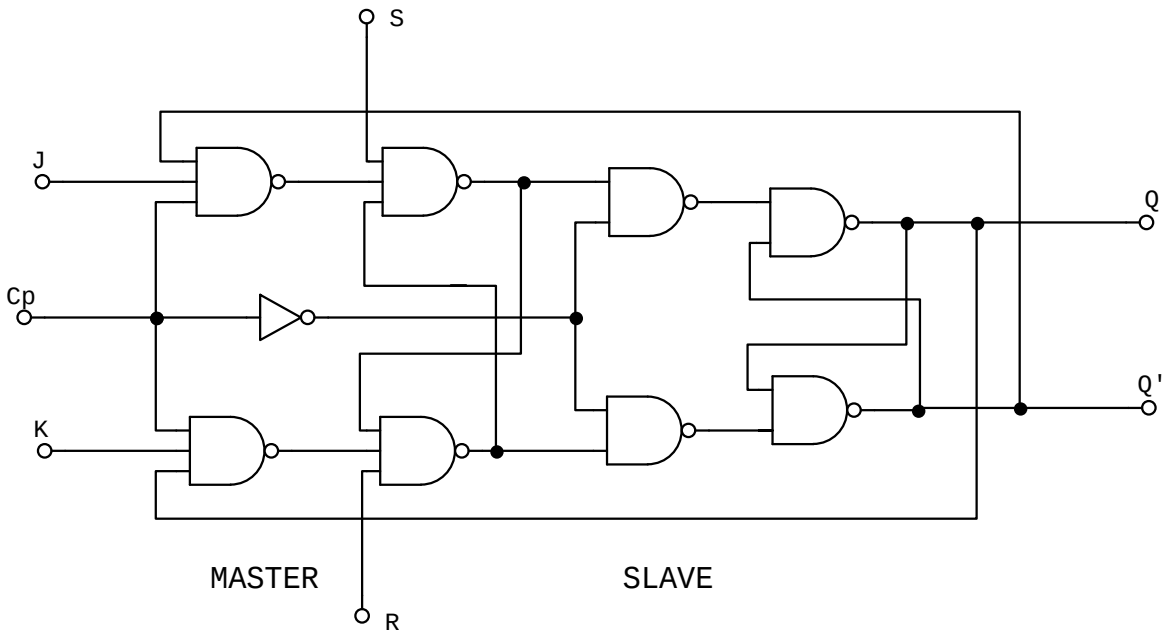
Esta expressão, como no caso do FF-SR, pode ser implementada também através de portas lógicas NAND ou NOR, conforme indicado pela equação lógica. Entretanto, a implementação do FF JK, por razão de praticidade, é realizada através da célula básica do Flip-Flop SR, cujo circuito resultante é mostrado abaixo, utilizando-se o método de conversão explicado no item V.2.



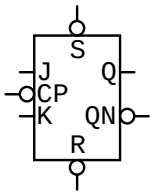
A versão de tal circuito, incluindo a entrada do *clock*, é portanto:



Devido à necessidade de se dispor de circuitos seqüenciais melhor sincronizados e com a propriedade de serem setados ou resetados, independentemente tanto do *clock* quanto das entradas, tornou-se muito popular o circuito mostrado abaixo, denominado de Flip-Flop JK - MASTER-SLAVE. Do ponto de vista lógico, não há qualquer diferença da síntese anteriormente realizada. Contudo, esta concepção de montagem permite que, de acordo com os valores lógicos S ou R sejam executadas as funções de PRESET ou PRERESET. Observa-se também que o inversor presente no circuito somente libera o segundo estágio (SLAVE), para efetuar eventuais transições, após o término do *clock*, quando as saídas da célula básica do primeiro estágio (MASTER) já se encontram estabilizadas.



ou



Embora esta versão do Flip-Flop JK ainda se encontre em uso, as modernas tecnologias permitem que as transições dos Flip-Flop's ocorram para pulsos extremamente estreitos tanto na subida quanto na descida dos mesmos, não havendo problemas mais significativos quanto à sincronização de operações, desde que as características inerentes aos pulsos de comando indicadas pelos fabricantes de tais circuitos, sejam obedecidas. Tais características são indicadas nos manuais pertinentes.

5.2.4 Tabela de Excitação FF-JK:

$Q_n \rightarrow Q_{n+1}$	$J_n\ K_n$
$0 \rightarrow 0$	0 x
$0 \rightarrow 1$	1 x
$1 \rightarrow 0$	x 1
$1 \rightarrow 1$	x 0

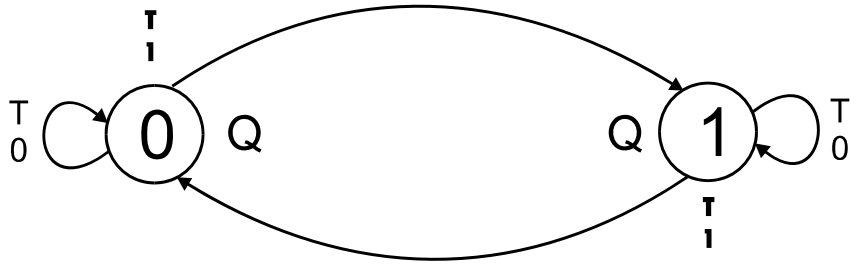
Como já se sabe, esta tabela-verdade define os estados lógicos necessários nas entradas do FF-JK, para que as transições aí representadas possam ocorrer.

Se o dispositivo estiver no estado lógico 0 (no momento de observação n) e se deseja, após a ocorrência do pulso *clock*, que o mesmo continue nesse mesmo estado, torna-se necessário que a entrada J, no momento de observação n, seja 0 (para não *setar* o dispositivo), mas a entrada K, no mesmo momento de observação, pode apresentar qualquer valor lógico pois, em sendo nível lógico 0, a tabela característica diz que o estado seguinte é igual ao estado anterior e, em sendo nível lógico 1, ocorrerá um *reset* do dispositivo, mas, como ele já se encontra *resetado*, o valor lógico dessa entrada é irrelevante para a obtenção da operação pretendida. Do mesmo modo, analisando a situação para o caso de o dispositivo se encontrar no estado lógico 0, e fazer a sua transição para o estado 1, necessário se torna, no momento de observação n, que a entrada J seja 1, podendo a entrada K assumir qualquer valor lógico pois, sendo K=0, o Flip-Flop é conduzido ao estado SET e sendo K=1, o Flip-Flop muda de estado, ou seja: faz a transição de 0 para 1, resultando na mesma operação. No caso de as duas

entradas apresentarem valor lógico 1, o Flip-Flop simplesmente muda de estado quando ocorre o pulso *clock* (vide Tabela Característica).

5.3 FLIP-FLOP T

5.3.1 Diagrama de Estados FF-T:



O Diagrama de Estados indica que este Flip-Flop apresenta apenas uma entrada denominada T. Quando a entrada T apresenta valor lógico 0, o Flip-Flop não muda de estado e, quando tal entrada assume valor lógico 1, o Flip-Flop faz a transição para o estado complementar ao que se encontrava no momento de observação (n). As tabelas verdade resultantes apresentam-se a seguir:

5.3.2 Tabela Característica FF-T:

T_n	Q_{n+1}
0	Qn
1	$\overline{Q}$n

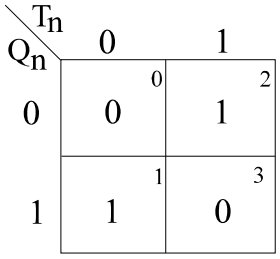
5.3.3 Tabela de Estados FF-T:

DEC	T_n	Q_n	Q_{n+1}
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

A equação sob a forma numérica:

$$\mathbf{Q_{n+1} = \sum m(1,2)}.$$

O MK correspondente:



Evidentemente, a expressão resultante é a de definição de um OR-Exclusivo (XOR):

$$\mathbf{Q_{n+1} = \overline{Q}_n T_n + Q_n \overline{T}_n}$$

ou

$$\mathbf{Q_{n+1} = Q_n \oplus T_n .}$$

A implementação do circuito eletrônico deste Flip-Flop é realizada normalmente através da célula básica do Flip-Flop SR ou do JK, conforme será mostrado no item V.2.

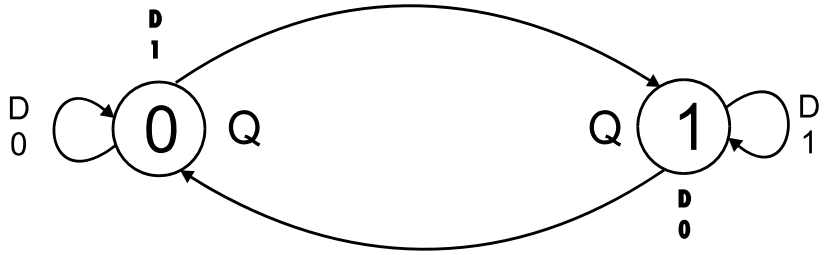
5.3.4 Tabela de Excitação FF-T:

$Q_n \rightarrow Q_{n+1}$	T_n
$0 \rightarrow 0$	0
$0 \rightarrow 1$	1
$1 \rightarrow 0$	1
$1 \rightarrow 1$	0

Esta tabela de excitação apenas indica que, no caso do Flip-Flop T, sempre que se deseja promover uma transição ($0 \rightarrow 1$ ou $1 \rightarrow 0$), deve-se apresentar na entrada T o valor lógico 1. Naturalmente, o valor 0 na entrada T faz com que o Flip-Flop mantenha-se no estado em que se encontrava no momento de observação n.

5.4 FLIP-FLOP D

5.4.1 Diagrama de Estados FF-D:



Similarmente ao caso anterior, o Diagrama de Estados indica que este Flip-Flop apresenta apenas uma entrada denominada D. Quando o Flip-Flop se encontra *resetado* ($Q_n = 0$) e a entrada D apresenta valor lógico 0, o Flip-Flop não muda de estado. Quando essa entrada assume valor lógico 1, o Flip-Flop faz a transição para o estado lógico 1 ($Q_{n+1} = 1$). Caso o Flip-Flop se apresente *setado* ($Q_n = 1$), e a entrada D apresente valor lógico 1, o Flip-Flop não muda de estado. Quando a mencionada entrada assume o valor lógico **0**, o Flip-Flop faz a transição para o estado lógico **0** ($\mathbf{Q_{n+1} = 0}$).

Na prática, o diagrama de estados informa apenas que a estado seguinte do Flip-Flop **D** ($\mathbf{Q_{n+1}}$) é sempre determinado pelo valor lógico da entrada **D**, no momento de observação **n** ($\mathbf{D_n}$), conforme fica demonstrado pelas tabelas-verdade a seguir:

5.4.2 Tabela Característica FF-D:

D_n	Q_{n+1}
0	0
1	1

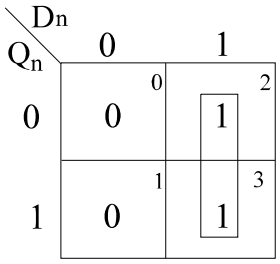
5.4.3 Tabela de Estados FF-D:

DEC	D_n	Q_n	Q_{n+1}
0	0	0	0
1	0	1	0
2	1	0	1
3	1	1	1

Daí,

$$\mathbf{Q_{n+1} = \sum m(2,3).}$$

Mapa de Karnaugh:



Logo,

$$\mathbf{Q_{n+1} = D_n}$$

Assim como no caso do Flip-Flop **T**, o Flip-Flop **D** é implementado a partir da célula básica do Flip-Flop **SR** ou do **JK**, conforme será mostrado adiante.

5.4.4 Tabela de Excitação FF-D:

$Q_n \rightarrow Q_{n+1}$	D_n
$0 \rightarrow 0$	0
$0 \rightarrow 1$	1
$1 \rightarrow 0$	0
$1 \rightarrow 1$	1

Esta Tabela de Excitação informa apenas que, quando se deseja que o estado seguinte do **Flip-Flop (Q_{n+1})** apresente um determinado valor lógico, basta

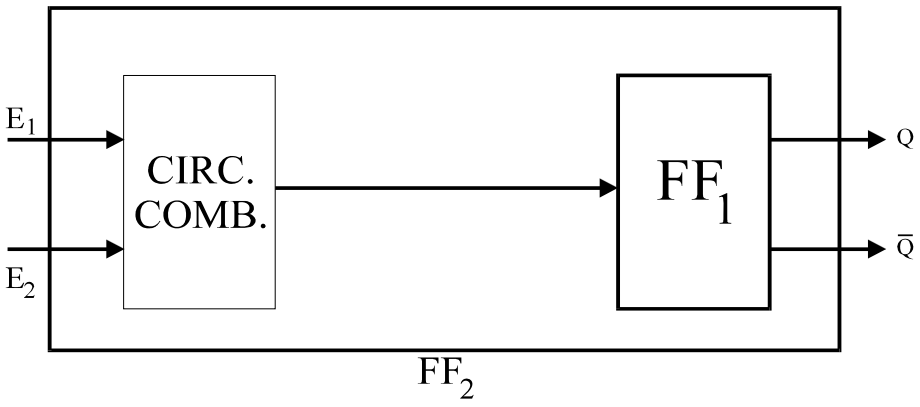
assegurar que esse valor lógico esteja presente na entrada **D**, no momento de observação **n**.

Um Flip-Flop que apresente a característica de a saída apenas seguir o valor lógico da entrada, mediante a ocorrência de um pulso *clock*, é muito utilizado como armazenador de uma informação binária, e a sua utilização em momentos adequados pode ser extremamente importante dentro de uma dada seqüência de operações, como será mostrado oportunamente.

5.5 CONVERSÃO DE FLIP-FLOP'S

Qualquer tipo de Flip-Flop pode ser transformado em qualquer outro, inclusive num Flip-Flop especial, cuja tabela característica venha a ser definida pelo usuário. Para tanto, utilizam-se as tabelas de excitação do Flip-Flop que se encontra disponível, em contraposição com a do Flip-Flop desejado. Em seguida, encontra-se a equação, ou equações, das variáveis de entrada do Flip-Flop disponível, em função de Q_n e da entrada, ou entradas, do Flip-Flop desejado. A equação, ou equações, resultantes definem os circuitos combinacionais capazes de programar a transição do Flip-Flop disponível, tal que o mesmo se comporte segundo a tabela de transição do desejado.

Esquematicamente, pode-se indicar o procedimento usado da seguinte maneira:



O diagrama acima representa o **FF1** (**disponível**) que deverá ser transformado no **FF2** (**desejado ou necessário**), através do circuito combinacional (**Circ.Comb**).

Para facilitar a compreensão do método exposto, vários exemplos de conversão entre tais dispositivos são apresentados a seguir.

5.5.1 Conversão do FF SR em FF JK

A conversão ou a implementação de um Flip-Flop JK através de um Flip-Flop SR, como explicada anteriormente, é realizada através da contraposição entre as Tabelas de Excitação do SR (disponível) e JK (desejado), tal que se possa determinar as equações:

$$\mathbf{S_n = \varphi\,(\,Q_n\,,\,J_n\,,\,K_n\,)} \quad \mathbf{e} \quad \mathbf{R_n = \varphi\,(\,Q_n\,,\,J_n\,,\,K_n\,)}$$

Como já se sabe, as tabelas em questão são as indicadas abaixo, onde as entradas do Flip-Flop desejado aparecem em negrito:

$Q_n \rightarrow Q_{n+1}$	$S_n\ R_n$	$J_n\ K_n$
$0 \rightarrow 0$	0 x	0 x
$0 \rightarrow 1$	1 0	1 x
$1 \rightarrow 0$	0 1	x 1
$1 \rightarrow 1$	x 0	x 0

Partindo-se, pois, das tabelas acima, vê-se que os valores lógicos de $\mathbf{S_n}$ e $\mathbf{R_n}$ podem ser inseridos nos Mapas de Karnaugh correspondentes, em função dos valores lógicos de $\mathbf{Q_n}$, $\mathbf{J_n}$ e $\mathbf{K_n}$, para todos os MINTERM's constantes dos citados Mapas.

Assim,

S_n

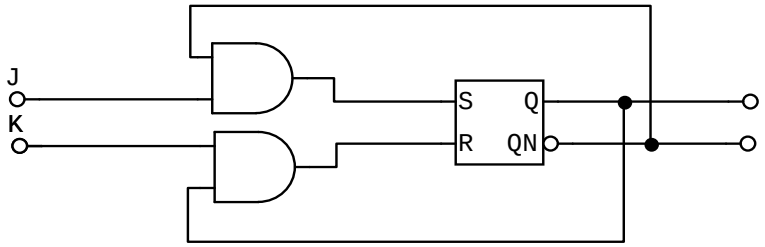
K_n \ Q_nJ_n	00	01	11	10
	0	2	6	4
0	0	1	X	X
	1	3	7	5
1	0	1	0	0

R_n

K_n \ Q_nJ_n	00	01	11	10
	0	2	6	4
0	X	0	0	0
	1	3	7	5
1	X	0	1	1

$$\mathbf{S_n = \overline{Q}_n\,J_n} \qquad\qquad \mathbf{R_n = Q_n\,K_{n\text{-}}}$$

ou



5.5.2 Conversão do FF-SR em FF-T

A conversão ou a implementação de um Flip-Flop T, através de um SR, pode ser realizada do mesmo modo, desde que as equações lógicas para S e R sejam encontradas em função do estado Q_n do FF disponível e da entrada T do desejado.

As Tabelas de Excitação dos dois FF são apresentadas a seguir:

$Q_n \rightarrow Q_{n+1}$	S_n R_n	T_n
$0 \rightarrow 0$	0 x	0
$0 \rightarrow 1$	1 0	1
$1 \rightarrow 0$	0 1	1
$1 \rightarrow 1$	x 0	0

As equações de S_n e R_n em função de Q_n e T_n são encontradas a partir dos Mapas de Karnaugh a seguir, onde os estados lógicos de S_n e R_n são inseridos nos respectivos Mapas para todos os MINTERM'S relativos às variáveis Q_n e T_n:

S_n

T_n \ Q_n	0	1
0	0 (0)	X (2)
1	1 (1)	0 (3)

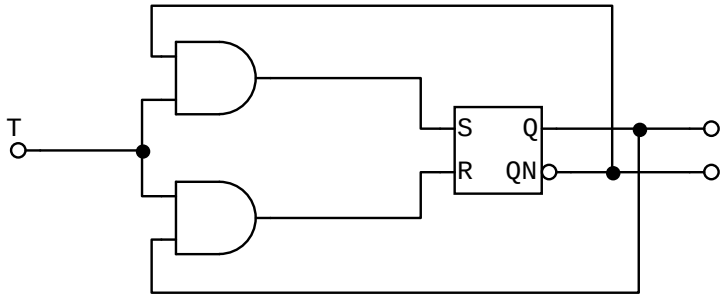
R_n

T_n \ Q_n	0	1
0	X (0)	0 (2)
1	0 (1)	1 (3)

Logo,

$$S_n = \overline{Q_n}\, T_n \qquad\qquad R_n = Q_n\, T_n$$

Assim, tendo-se o FF-SR mostrado abaixo e à direita, anexando-se os AND's indicados pelas equações acima, obtém-se um circuito cujas transições correspondem àquelas de um FF do tipo T.



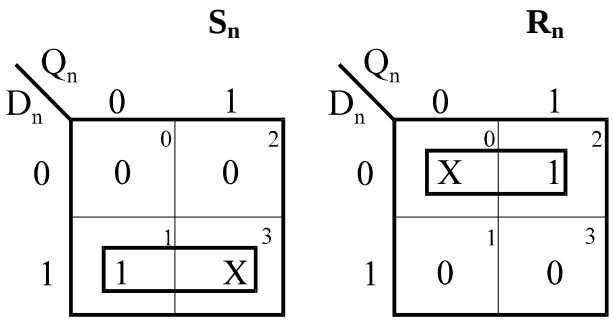
5.5.3 Conversão do FF-SR em FF-D

Repetindo-se o procedimento para o caso de se converter um FF-SR em FF-D, tem-se as tabelas de excitação:

$Q_n \rightarrow Q_{n+1}$	$S_n\ R_n$	D_n
0 → 0	0 x	0
0 → 1	1 0	1
1 → 0	0 1	0
1 → 1	x 0	1

e

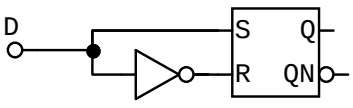
Os Mapas de Karnaugh correspondentes:



Logo,

$$S_n = D_n \qquad R_n = \overline{D_n}$$

ou



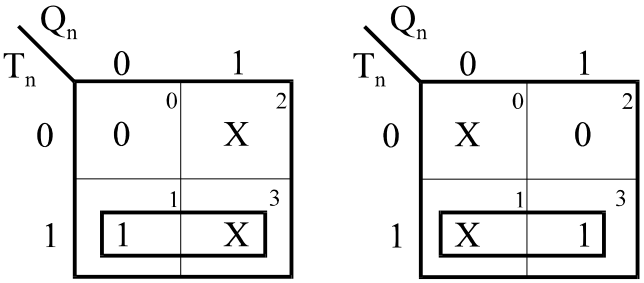
5.5.4. Conversão do FF-JK em FF-T

Ainda segundo o mesmo processo:

$Q_n \rightarrow Q_{n+1}$	$J_n\ K_n$	T_n
$0 \rightarrow 0$	0 x	0
$0 \rightarrow 1$	1 x	1
$1 \rightarrow 0$	x 1	1
$1 \rightarrow 1$	x 0	0

Mapas de Karnaugh:

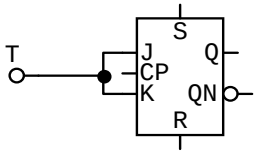
J_n **K_n**



ou

$J_n = T_n$ **$K_n = T_n$**

Apesar de os Mapas de Karnaugh indicarem tal resultado, uma observação um pouco mais atenta às tabelas de excitação acima revela, por inspeção, que, tanto a entrada $\mathbf{J_n}$ quanto a $\mathbf{K_n}$ coincidem com o valor lógico da entrada $\mathbf{T_n}$, resultando em $\mathbf{J_n} = \mathbf{K_n} = \mathbf{T_n}$, ou no circuito equivalente:



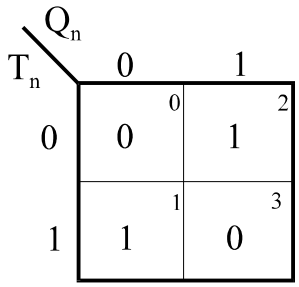
5.5.5 Conversão do FF-D em FF-T

Pelo mesmo processo, temos as Tabelas de Excitação e o Mapa de Karnaugh correspondentes:

Tabelas de Excitação:

$Q_n \rightarrow Q_{n+1}$	D_n	T_n
$0 \rightarrow 0$	0	0
$0 \rightarrow 1$	1	1
$1 \rightarrow 0$	0	1
$1 \rightarrow 1$	1	0

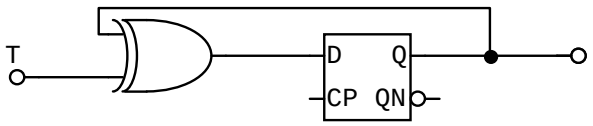
Mapa de Karnaugh:



Equação :

$$\mathbf{D_n = Q_n \oplus T_n}$$

ou

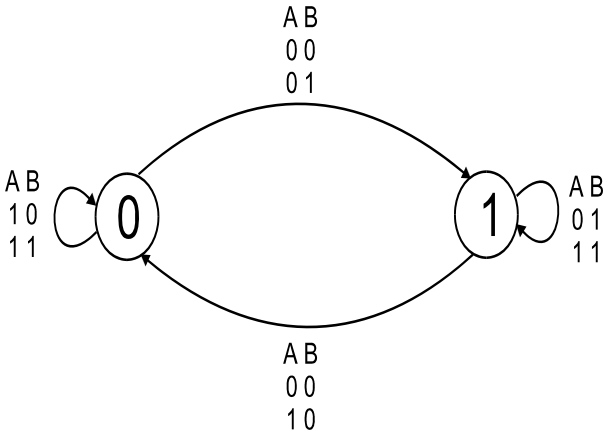


5.5.6 Conversão do FF-SR num FF qualquer

Seja um Flip-Flop qualquer, de entradas **A** e **B**, definido pela seguinte Tabela-Característica:

A_n B_n	Operação em Q_{n+1}
0 0	mudar de estado
0 1	setar o dispositivo
1 0	resetar o dispositivo
1 1	manter o estado anterior

A Tabela acima pode ser facilmente representada pelo diagrama de estados:



Expandindo-se a Tabela-Característica, ou mesmo através do próprio diagrama de estados, chega-se à Tabela de Estados ou de Transição:

DEC	A_n	B_n	Q_n	Q_{n+1}
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Da Tabela de Estados ou de Transição acima, define-se a função:

$$\mathbf{Q_{n+1} = \sum m(0,2,3,7).}$$

O Mapa de Karnaugh pode então ser construído:

Q_n \ A_nB_n	00	01	11	10
0	0: 1	2: 1	6: 0	4: 0
1	1: 0	3: 1	7: 1	5: 0

Pelas adjacências indicadas, chega-se então à expressão:

$$Q_{n+1} = \overline{A}_n \overline{Q}_n + B_n Q_n$$

Como nos casos anteriores, este Flip-Flop hipotético, de entradas A e B , poderia ser sintetizado a partir de portas lógicas NAND's ou NOR's. Contudo, realizando a implementação do mesmo através da célula básica do FF-SR, conforme solicitado acima, chega-se às tabelas de excitação a seguir, e aplica-se o procedimento de conversão já demonstrado em diversos exemplos anteriores:

Tabelas de Excitação:

$Q_n \rightarrow Q_{n+1}$	S_n R_n	A_n B_n
0 → 0	0 x	1 x
0 → 1	1 x	0 x
1 → 0	x 1	x 0
1 → 1	x 0	x 1

Mapas de Karnaugh correspondentes:

S_n

B_n \ Q_nA_n	00	01	11	10
	0	2	6	4
0	1	0	0	0
	1	3	7	5
1	1	0	X	X

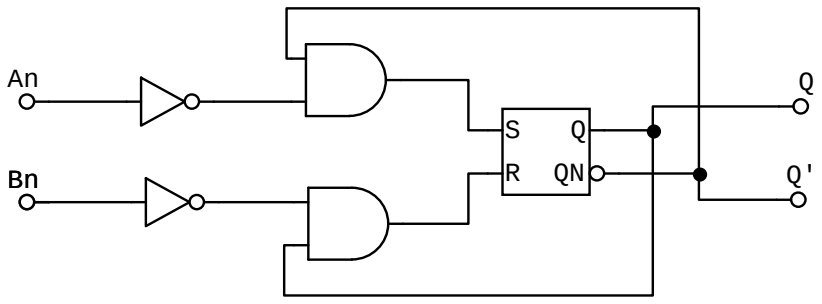
R_n

B_n \ Q_nA_n	00	01	11	10
	0	2	6	4
0	0	X	1	1
	1	3	7	5
1	0	X	0	0

Equações simplificadas:

$$S_n = \overline{Q}_n \overline{A}_n \qquad R_n = Q_n \overline{B}_n$$

Vale dizer então que, o circuito que realiza o FF-AB, através do FF-SR, é:



6 IMPLEMENTAÇÃO DE CIRCUITOS LÓGICOS & FIELD PROGRAMMABLE GATE ARRAY - FPGA's

6.1 CONSIDERAÇÕES GERAIS

As portas lógicas integradas são fabricadas de modo a que num único bloco de material semicondutor ("***chip***"), um ou mais circuitos completos para realizar determinadas funções são implementados, agrupando-se de forma compacta e indissociável diversos dispositivos e portas lógicas básicas, além de circuitos de larga utilização, tais como: contadores, codificadores, decodificadores, e variado número de funções lógicas de interesse aplicado.

O material semicondutor que serve como substrato, e as técnicas utilizadas para a fabricação dos circuitos integrados são de fundamental importância na definição das características elétricas distintas entre os circuitos sintetizados. Tais características, e o grau de compactação ou densidade de integração dos circuitos, são fatores preponderantes no estabelecimento das suas propriedades gerais e, conseqüentemente, no grau de aplicabilidade de cada um deles. A grande diversidade de características e a densidade de integração dos circuitos integrados permitem classificá-los em *linhas* ou *famílias,* as quais são caracterizadas segundo o seu grau de integração e o tipo de componentes ativos e passivos utilizados no processo da sua fabricação. Assim, os circuitos são classificados como:

- **-SSI ("*Small Scale Integration*")**
  Correspondendo àqueles circuitos que possuem até 12 portas lógicas básicas integradas.

- **-MSI ("*Medium Scale Integration*")**
  Correspondendo àqueles circuitos que possuem até 99 portas lógicas mais sofisticadas, sintetizando algumas funções lógicas mais utilizadas, objetivando reduzir as conexões externas entre vários circuitos SSI que realizariam a mesma função lógica.

- **-LSI ("*Large Scale Integration*")**
  Estes circuitos possibilitam a realização de funções mais complexas que suscitariam a interconexão entre vários outros classificados como MSI e SSI, a exemplo de microprocessadores, microcontroladores, memórias, relógios digitais e outros.

- **-VLSI ("*Very Large Scale Integration*")**
  Por extensão, estes circuitos desempenham funções que necessitam de vários CI's do tipo MSI ou LSI para implementá-los, a exemplo de unidades aritméticas, e memórias de maior porte.

Quanto ao tipo de componentes que são utilizados para a implementação física dos circuitos, várias linhas ou famílias de circuitos lógicos são fabricadas, a saber:

- **RTL ("Resistor Transistor Logic")**
- **DTL ("Diode Transistor Logic")**
- **TTL ("Transistor Transistor Logic")**
- **ECL ("Emitter Coupled Logic")**
- **CMOS ("Complementary Metal Oxide Semiconductor")**
- **VMOS ("Vertical Metal Oxide Semiconductor")**
- **I^2L ("Integrated Injection Logic")**

As famílias de circuitos lógicos **RTL** e **DTL**, de grande valor histórico, são consideradas obsoletas, sobretudo em decorrência do desenvolvimento das famílias **TTL**, **CMOS**, **ECL** e outras, as quais apresentam hoje grande diversificação de portas lógicas e funções, melhor resposta em freqüência, sendo a família **ECL** a de escolha natural especialmente quando se necessita adequado desempenho em elevadas freqüências pois, enquanto nas demais famílias os semicondutores trabalham entre a região de corte e a de saturação, na família **ECL** os transistores operam entre a região de corte e a linear, sem entrar em saturação, o que reduz o tempo de resposta, permitindo a sua utilização em mais elevadas freqüências.

A tecnologia dos circuitos integrados faculta ao projetista dos circuitos lógicos, ou digitais, a opção dos ***Field Programmable Gate Arrays*** ou **FPGA's**, os quais possibilitam a implementação de circuitos com maior flexibilidade, especialmente por apresentarem uma lógica programável, propiciando a síntese de configurações distintas através de 'softwares', permitindo que considerável

número de funções possam vir a ser configuradas num mesmo bloco de circuito integrado, a exemplo de codificadores, decodificadores, interfaces, controladores, contadores, memórias semicondutoras, *shift-registers*, etc.

Assim como nos circuitos digitais integrados sob a forma de portas lógicas do tipo **AND**, **NAND**, **OR**, **NOR**, **OREx**, **NOREx**, **AOI**, **Flip-Flops** e outras, várias Famílias de **FPGAs** surgiram, sendo as suas características e aplicações apresentadas nos manuais dos respectivos fabricantes, mediante "***Application Notes***" correspondentes.

Dentre as diversas Linhas ou Famílias de **FPGAs** destacam-se as seguintes **Famílias: XC4000**, **Spartan**, **Spartan XL**, **Spartan-II**, **Virtex, Virtex-E**, **Virtex-II/Virtex-II PRO** e **Spartan 3**, além de outras desenvolvidas por indústrias especializadas, cada uma delas objetivando apresentar elevada performance ou desempenho dos circuitos eletrônicos que utilizam tais dispositivos.

As considerações aqui apresentadas visam apenas mostrar algumas opções relativas à **síntese dos circuitos lógicos** que são desenvolvidos a partir dos procedimentos discutidos nos capítulos anteriores, assunto este que se encontra em permanente evolução, enquanto tecnologia aplicada, e foge ao escopo do presente trabalho, o qual procura principalmente apresentar os métodos de análise e síntese dos circuitos lógicos, enfatizando a sua metodologia ou filosofia de abordagem, em lugar dos aspectos estritamente tecnológicos.

Naturalmente, os leitores interessados nos aspectos eminentemente técnicos quanto à implementação dos circuitos dessa natureza, terão acesso às informações adequadas nos manuais dos fabricantes, buscando compatibilizar a síntese dos circuitos específicos a serem desenvolvidos com a disponibilidade das portas lógicas, ou circuitos integrados dedicados, à disposição nos seus laboratórios.

Vale ressaltar ainda que, grande número de referências a livros, manuais e artigos sobre esse tema encontra-se disponível também através de '***Application Notes***' em artigos especializados, inclusive de fácil acesso por meio eletrônico via *internet (Wikipédia ou similares)*.

As considerações abaixo, por exemplo, de caráter mais aplicado aos dispositivos em pauta, são provenientes de informações fornecidas diretamente dos manuais dos fabricantes ou ***sites*** especializados, para efeito de ilustração quanto às

disponibilidades de recursos tecnológicos, e às suas aplicações na solução de problemas concretos na elaboração de circuitos e projetos aplicados.

A grande maioria dos ***'chips'*** utilizados nos equipamentos eletrônicos de uso geral, tais como: rádios, televisores, telefones celulares, e outros, são pré-programados; ou seja, têm as suas funções específicas previamente definidas no ato da sua fabricação.

A evolução na fabricação dos circuitos integrados permitiu então uma nova categoria de ***hardwares reconfiguráveis,*** os quais têm as suas funções finais definidas pelos próprios usuários, a exemplo dos ***FPGAs (Field Programmable Gate Arrays)***, dispositivos largamente utilizados especialmente no processamento de informações digitais.

O ***FPGA*** é composto basicamente por três tipos de componentes**: Blocos de Entrada** e **Saída (IOB), Blocos Lógicos Configuráveis (CLB)** e **Chaves de Interconexão (Switch Matrix).**

Os mencionados B**locos Lógicos** são dispostos de forma bidimensional; as chaves de interconexão são dispostas em formas de trilhas verticais e horizontais entre as linhas e as colunas dos blocos lógicos. Assim, alguns termos comuns e presentes nos manuais de tais dispositivos suscitam esclarecimentos, a exemplo de:

- **CLB (*Configuration Logical Blocks*):**
  Circuitos idênticos, construído pela reunião de **flip-flops** (entre 2 e 4) e a utilização de lógica combinacional. Utilizando os **CLBs**, o usuário pode construir diversos elementos funcionais lógicos.

- **IOB (*Input/Output Block*):**
  São circuitos responsáveis pelo interfaceamento das saídas provenientes das combinações de **CLBs**. São basicamente ***buffers***, que funcionarão como um **acoplamento** bidirecional entrada-saída do **FPGA**.

- ***Switch Matrix*** **(chaves de interconexões):**
  Trilhas utilizadas para conectar os **CLBs** e **IOBs**. O terceiro grupo é composto pelas interconexões. Os recursos de interconexões possuem trilhas para conectar as entradas e saídas dos **CLBs** e **IOBs** para as redes apropriadas.

Geralmente, a configuração é estabelecida por programação interna das células de memória estática, que determinam funções lógicas e conexões internas implementadas no **FPGA** entre os **CLBs** e os **IOBs.** O processo de escolha das interconexões é chamado de roteamento.

6.1.1 Considerações sobre a tecnologia do *FPGA*:

O ***FPGA*** é um ***chip*** que possibilita a implementação de circuitos lógicos relativamente complexos, consistindo de um grande arranjo de células lógicas ou blocos lógicos configuráveis, contidos em um único circuito integrado, conforme anteriormente mencionado. Vale observar que, cada célula contém capacidade computacional para implementar funções lógicas e efetuar o ***roteamento*** para comunicação entre elas.

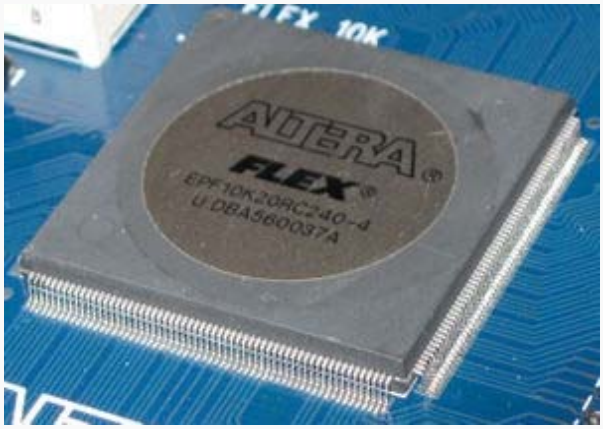
O primeiro ***FPGA*** disponível comercialmente foi desenvolvido pela empresa Xilinx Inc, em 1983. Os **FPGAs** não possuem planos **OR** ou **AND**, e consistem de um grande arranjo de células configuráveis que podem ser utilizadas para a implementação de funções lógicas.

Um ***FPGA*** é basicamente constituído por blocos lógicos, blocos de entrada e saída, e chaves de interconexão. Os blocos lógicos formam uma matriz bidimensional, e as chaves de interconexão são organizadas como canais de roteamento horizontal e vertical entre as linhas e colunas dos blocos lógicos. Os canais de roteamento possuem chaves de interligação programáveis que permitem conectar os blocos lógicos de maneira conveniente, em função das necessidades de cada projeto. No interior de cada bloco lógico do ***FPGA*** existem vários modos possíveis para a implementação de funções lógicas. O mais utilizado pelos seus fabricantes, a exemplo da Empresa **Altera Corp.**, é o bloco de memória **LUT** (**Look-Up Table**). Esse tipo de bloco lógico contém células de armazenamento que são utilizadas para implementar pequenas funções lógicas. Cada célula é capaz de armazenar um único valor lógico: ***zero*** ou ***um***.

Nos FPGAs disponíveis comercialmente, tais blocos lógicos 'LUTs' possuem geralmente quatro ou cinco entradas, o que permite endereçar 16 ou 32 células de armazenamento. Quando um circuito lógico é implementado em um FPGA, os blocos lógicos são programados para realizar as funções desejadas, e os canais de roteamento são estruturados de forma a efetuar as interconexões necessárias entre tais blocos lógicos. As células de armazenamento dos LUTs de um FPGA são voláteis, o que implica na perda do conteúdo armazenado, no caso de falta de suprimento de energia elétrica. Dessa forma, tal dispositivo deve ser

reprogramado toda vez que for energizado. Assim, geralmente utiliza-se uma pequena memória FLASH EEPROM (*Electrically Erasable Programmable Read Only Memory*), cuja função é carregar automaticamente as células de armazenamento, a cada vez que o FPGA é re- energizado.

6.1.2 Para efeito de ilustração, apresenta-se abaixo o aspecto de um FPGA da Altera



Um FPGA da **Altera** com 20.000 células.

Objetivando-se classificar os FPGAs quanto ao bloco lógico pertinente, foram criadas algumas categorias, a saber:

- **Grão grande: podem possuir como grão unidades lógicas e aritméticas, pequenos microprocessadores e memórias.**

- **Grão médio: freqüentemente contêm duas ou mais LUTs e dois ou mais flip-flops.**

- **Grão pequeno: contêm um grande número de blocos lógicos simples. Os blocos lógicos normalmente contêm uma função lógica de duas entradas ou um multiplexador 4x1 e um flip-flop.**

A arquitetura de roteamento de um **FPGA** é a forma pela qual os seus barramentos e chaves de comutação são posicionados para permitir a interconexão entre as células lógicas. Essa arquitetura deve permitir que se obtenha um roteamento completo e, ao mesmo tempo, uma alta densidade de portas lógicas. Para uma melhor compreensão dessa arquitetura, torna-se necessária a definição de nomenclaturas e alguns conceitos básicos tais como:

- **Pinos: entradas e saídas dos blocos lógicos;**
- **Conexão: ligação elétrica de um par de pinos;**
- **Rede: um conjunto de pinos que estão conectados;**
- **Segmento de trilha não interrompido por chaves programáveis;**
- **Bloco de Comutação: utilizado para conectar 2 segmentos de trilha;**
- **Canal de roteamento: grupo de duas ou mais trilhas paralelas.**
- **Bloco de conexão: permite a conectividade das entradas e saídas de um bloco lógico com os segmentos de trilhas nos canais.**

As chaves programáveis de roteamento apresentam algumas propriedades, tais como: tamanho, resistência, capacitância e tecnologia de fabricação, que afetam principalmente a velocidade e o tempo de propagação dos sinais, e definem características como volatilidade e capacidade de reprogramação. Na escolha de um dispositivo reconfigurável, esses fatores devem ser avaliados. Basicamente existem três tipos de tecnologias de programação das chaves de roteamento:

- **SRAM (*Static Random Access Memory*):** nesta tecnologia, a chave de roteamento ou comutador é um transistor de passagem ou um multiplexador controlado por uma memória estática de acesso aleatório SRAM. Devido à volatilidade dessas memórias, os FPGAs que se utilizam dessa tecnologia precisam de uma memória externa tipo FLASH EEPROM. Essa tecnologia ocupa muito espaço no circuito integrado, entretanto é rapidamente reprogramável.

- **Antifuse:** esta tecnologia baseia-se num dispositivo de dois terminais, que no estado não programado apresenta uma alta impedância (circuito aberto). Aplicando-se uma tensão, por exemplo, entre 11 e 20 Vdc, o dispositivo forma um caminho de baixa impedância entre seus terminais.

- **Gate flutuante:** baseia-se em transistores **MOS (*Metal Oxide Semiconductor*)**, especialmente construído com dois gates flutuantes semelhantes aos usados nas memórias **EPROM (*Erasable Programmable Read Only Memory*)** e **EEPROM (*Electrical EPROM*)**. A maior vantagem dessa tecnologia é a sua capacidade de programação e a retenção dos dados. Além disso, da mesma forma que uma memória **EEPROM**, os dados podem ser programados com o circuito integrado instalado na placa, característica denominada **ISP (*In System Programmability*)**.

Encontram-se atualmente no mercado três tipos importantes de FPGA's, os quais apresentarão os desempenhos pertinentes, dependendo das aplicações para as quais os mesmos venham a ser destinados:

RAM Estática: FPGA na qual suas conexões entre as portas são feitas entre blocos lógicos por meio de portas de transmissão ou multiplexadores controladas por células SRAM. Tem como vantagem a possibilidade de ser rapidamente configurada, porém exige hardware externo auxiliar que deve ser montado junto com os blocos lógicos.

Transistores de Passagem: Esta é uma opção de menor custo que a opção anteriormente citada, sendo composta por uma grande concentração de transistores configurados em modo de corte ou modo de condução.

EPROM/EEPROM: Baseada na tecnologia de criação de memórias EPROM e EEPROM, sendo a sua principal vantagem a de permitir a reprogramação sem que se precise armazenar a configuração externa.

6.1.3 Considerações necessárias

Além das funções lógicas ou digitais, alguns FPGAs dispõem de recursos analógicos, permitindo, portanto, a implementação de Conversores Analógico/Digital (ADC) e Digital/Analógico (DAC), oferecendo grande flexibilidade ao projetista. Tais circuitos são definidos como estrutura de interconexão programável ou (FPAA), e também permitem valores analógicos nesta interconexão.

Para efeito da implementação de circuitos mais complexos, vale ressaltar a possibilidade da aplicação dos Circuitos Integrados CPLDs (Complex Programmable Logic Devices), os quais podem promover a síntese de circuitos lógicos/digitais programáveis de muito maior grau de complexidade!

Evidentemente, o estudo mais aprofundado de tais dispositivos extrapolaria completamente a intenção inerente ao presente trabalho!.

Algumas referências sobre esses temas também podem ser encontradas na bibliografia indicada. Contudo, devido à dinâmica inerente a essas tecnologias, os manuais dos fabricantes e suas 'Application Notes' constituem-se na melhor e mais adequada fonte de referência, no sentido de se promover a otimização de um projeto específico a ser levado a termo de modo econômico e eficaz.

6.2 PESQUISAS EM FPGA

- Navarre AsyncArt. Asynchronous-SOPC research. Universidade de Navarre
- Reconfigurable Network Group da Universidade de Washington
- Circuits and Systems Group, Imperial College, London
- MEANDER FPGA Design Framework from the Democritus University of Thrace (Greece)
- Pesquisas em FPGA da Universidade de British Columbia
- Pesquisas em FPGA da Universidade de Toronto
- Pesquisas em FPGA da Northeastern University
- Pesquisas em FPGA da Universidade de Sao Paulo - LCR/ICMC/USP
- FPGA Reliability Studies, Brigham Young University
- The Virginia Tech Configurable Computing Laboratory
- Computer System Design Lab da Universidade de Kansas
- GRECO-Grupo de Engenharia da Computação - Centro de Informática da UFPE
- LAD - Laboratório de Arquiteturas Dedicadas - Grupo de Arquiteturas Dedicadas - Centro de Engenharia Elétrica e Informática - UFCG
- FPGA Database.

REFERÊNCIAS

- BARNA, A.; PORAT, D. I. *Integrated circuits in digital electronics*. York: John Wiley, 1973.

- BARTELT, Terry L. M. *Digital electronics*: an integrated laboratory approach. New Jersey: Prentice Hall, 2002.

- BURGER, Peter. *Digital design*: a practical course. New York: John Wiley, 1988.

- COSTA, César da. *Projetando controladores digitais com FPGA*. São Paulo:

- DEWEY, Allen. *Analysis and design of digital systems with VHDL*. Andover-UK: International Thomson, 1997.

- FLETCHER, W. I. *An engineering approach to digital design*. New Jersey: Prentice Hall, 1980.

- FLOYD, Thomas L. *Sistemas digitais*: fundamentos e aplicações. 9. ed. São Paulo: Artmed, 2007.

- GREENFIELD, J. D. *Practical digital design using IC's*. 2th ed. New York: John Wiley, 1983.

- HARTMUT, F.-W.; SADROZINSKI; JINYUAN ,Wu. *Aplicações de programmable gate Arrays de Campo em Investigação Científica* . London: Taylor & Francis, 2010.

- HILL, F. J.; PETERSON, G. R. *Digital logic and microprocessors*. New York: John Wiley, 1984.

- HILL, F. J.; PETERSON, G. R. *Introduction to switching theory and logical design*. 2th ed. New York: John Wiley, 1974.

- MELO, Mairton. *Circuitos digitais e microprocessadores*. São Paulo: McGraw Hill do Brasil, 1993.

- MORENO, Edward David; PENTEADO, Cesar Giacomini; RODRIGUES, Alexandre César. *Microcontroladores e FPGAs - Aplicações em Automação*. São Paulo: Novatec, 2005.
- SARAIVA, Fregni. *Engenharia do projeto lógico e digital*: circuitos e Prática. São Paulo: Edgard Blücher, 1995.

- TOCCI, Ronald J. *Sistemas digitais*: princípios e aplicações. 5. ed. Rio de Janeiro: Prentice-Hall do Brasil, 1994.

COLOFÃO

Formato
21 X 29,7 cm

Tipologia
Times New Roman

Papel
Alcalino 75 g/m^2 (miolo)
Cartão Supremo 300 g/m^2 (capa)

Impressão
Setor de Reprografia da EDUFBA

Capa e Acabamento
Cian

Tiragem
300