



SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL
Escola de Educação Profissional Senai “Plínio Gilberto Kröeff”

ELETRÔNICA DIGITAL

Professor: Carlos Ricardo dos Santos Barbosa
Unidade Curricular: Manutenção Eletrônica
Curso: Técnico em Eletrônica

São Leopoldo
2009

SUMÁRIO

1 ELETRÔNICA ANALÓGICA E DIGITAL	5
2 SISTEMA DE NUMERAÇÃO	7
2.1 SISTEMA DECIMAL.....	7
2.2 SISTEMA BINÁRIO	8
2.2.1 Conversão do sistema binário para o sistema decimal	9
2.2.2 Conversão do sistema decimal para o sistema binário	10
2.3 SISTEMA OCTAL.....	10
2.3.1 Conversão do sistema octal para o sistema decimal	11
2.3.2 Conversão do sistema decimal para o sistema octal	11
2.3.3 Conversão do sistema octal para o sistema binário	12
2.3.4 Conversão do sistema binário para o sistema octal	12
2.4 SISTEMA HEXADECIMAL	13
2.4.1 Conversão do sistema hexadecimal para o sistema decimal	14
2.4.2 Conversão do sistema decimal para o sistema hexadecimal	14
2.4.3 Conversão do sistema hexadecimal para o sistema binário	14
2.4.4 Conversão do sistema binário para o sistema hexadecimal	15
3 ARITMÉTICA BINÁRIA	16
3.1 ADIÇÃO BINÁRIA	16
3.2 SUBTRAÇÃO BINÁRIA.....	17
4 FUNÇÕES LÓGICAS	19
4.1 VARIÁVEIS LÓGICAS	19
4.1.1 Variável lógica de entrada	20
4.1.2 Variável lógica de saída	20
4.2 Função E ou AND	21
4.2.1 Tabela da verdade de uma função E ou AND	22
4.2.2 Porta E ou AND	22
4.3 FUNÇÃO OU OU OR	23
4.3.1 Tabela da verdade da função OU ou OR	24
4.3.2 Porta OU ou OR	24
4.4 FUNÇÃO NÃO OU NOT.....	25
4.4.1 Tabela da verdade da função NÃO ou NOT	25

4.4.2 Porta Inversor	26
4.5 FUNÇÃO NÃO E, NE OU NAND.....	26
4.5.1 Tabela da verdade da função NE ou NAND.....	26
4.5.2 Porta NE ou NAND.....	27
4.6 FUNÇÃO NÃO OU, NOU OU NOR.....	27
4.6.1 Tabela da verdade da função NOU ou NOR	27
4.6.2 Porta NOU ou NOR	28
4.7 FUNÇÃO OU EXCLUSIVO	28
4.8 FUNÇÃO COINCIDÊNCIA	29
4.9 QUADRO RESUMO.....	30
5 ALGEBRA BOOLE	32
5.1 POSTULADOS.....	32
5.1.1 Postulados da complementação.....	32
5.1.2 Postulado da adição.....	32
5.1.3 Postulado da multiplicação	32
5.2 TEOREMA DA ABSORÇÃO (IDENTIDADES AUXILIARES).....	33
5.3 TEOREMA DE DE MORGAN.....	34
5.4 TABELA RESUMO.....	36
6 MAPAS DE VEITCH – KARNAUGH	37
6.1 TÉCNICAS DE SIMPLIFICAÇÃO POR MAPAS	38
6.1.1 Por minitermos	38
6.1.2 Por maxitermo	38
6.2 DIAGRAMA DE VEITCH-KARNAUGH PARA 2 VARIÁVEIS	40
6.2.1 Transferência da tabela para o mapa	40
6.2.2 Formas de agrupamento.....	41
6.3 DIAGRAMA DE VEITCH-KARNAUGH PARA 3 VARIÁVEIS	42
6.3.1 Transferência da tabela para o mapa	43
6.3.2 Formas de agrupamento.....	43
6.4 DIAGRAMA DE VEITCH-KARNAUGH PARA 4 VARIÁVEIS	45
6.4.1 Transferência da tabela para o mapa	47
6.4.2 Formas de agrupamento.....	48
7 PARÂMETROS DOS CIRCUITOS LÓGICOS	50
7.1 ATRASO DE PROPAGAÇÃO	50
7.2 ATRASO DE TRANSIÇÃO.....	51

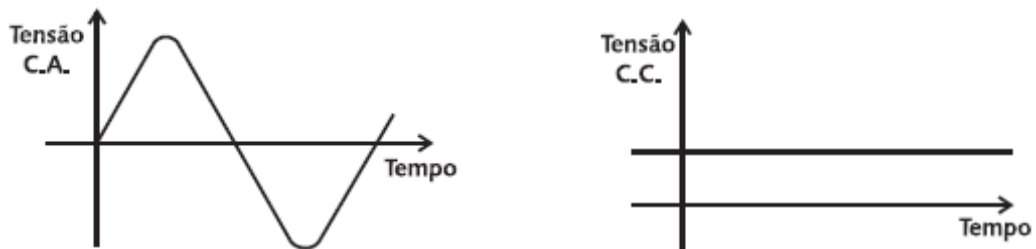
7.3 MARGEM DE RUÍDO.....	52
8 FAMÍLIAS LÓGICAS DE 1º GRUPO	53
9 FAMÍLIAS LÓGICAS DE 2º GRUPO	54
9.1 FAMÍLIA TTL.....	54
9.1.1 Saída TOTEM POLEM.....	55
9.1.2 Saída OPEN – COLLECTOR	55
9.1.3 Saída THREE – STATE	56
9.2 FAMÍLIA CMOS.....	56
10 CÓDIGOS NUMÉRICOS.....	58
10.1 CÓDIGO BCD 8421	58
10.2 CÓDIGO OCTAL.....	58
10.3 CÓDIGO HEXADECIMAL	59
10.4 CÓDIGO ASC II	60
10.4.1 Tabela ASCII	60
10.5 CÓDIGOS EXCESSO 3 (EX 3, XS3)	61
10.6 CÓDIGO GRAY.....	61
11 CODIFICADORES E DECODIFICADORES	63
11.1 CODIFICADOR DECIMAL/BINÁRIO.....	64
11.2 DECODIFICADOR BINÁRIO/DECIMAL.....	66
11.3 DECODIFICADOR PARA DISPLAY DE 7 SEGMENTOS.....	67
12 CIRCUITOS ARITMÉTICOS	72
12.1 MEIO SOMADOR.....	72
12.2 SOMADOR COMPLETO.....	73
12.3 MEIO SUBTRATOR	77
12.4 SUBTRATOR COMPLETO	78
12.5 SOMADOR / SUBTRATOR COMPLETO.....	80
13 FLIP-FLOP. REGISTRADORES E CONTADORES	82
13.1 FLIP-FLOPS.....	82
13.1.1 Flip-Flop RS básico	83
13.1.2 Flip-Flop RS com entrada clock.....	84
13.1.3 Flip-Flop JK.....	86
13.1.3.1 Flip-Flop JK com Entradas Preset e Clear	87
13.1.3.2 Flip-Flop JK mestre-escravo	88
13.1.3.3 Flip-Flop JK mestre-escravo com entrada preset e clear	89

13.1.4 Flip-Flop Tipo T	90
13.1.5 Flip-Flop Tipo D	91
13.2 REGISTRADORES DE DESLOCAMENTO	92
13.2.1 Conversor série-paralelo	92
13.2.2 Conversor paralelo-série	94
13.3 CONTADORES	95
13.3.1 Contadores assíncronos	95
13.3.1.1 Contador de pulsos	96
13.3.1.2 Contador de década	97
13.3.1.3 Contador assíncrono crescente/decrescente	98
13.3.2 Contadores síncronos	99
13.3.2.1 Contador gerador de uma seqüência qualquer	101
14 CIRCUITOS MULTIPLEX E DEMULTIPLEX	103
14.1 MULTIPLEX	103
14.1.1 Projeto do circuito de um multiplex	104
14.1.2 Ampliação da capacidade de um sistema multiplex	106
14.2 DEMULTIPLEX	108
14.2.1 Projeto do circuito de um demultiplex	108
14.2.2 Ampliação da capacidade de um circuito demultiplex	110
14.3 Multiplex e Demultiplex Utilizados na Transmissão de Dados	111
14.3.1 Transmissão Paralela	112
14.3.2 Transmissão Série	112
REFERÊNCIAS	114

1 ELETRÔNICA ANALÓGICA E DIGITAL

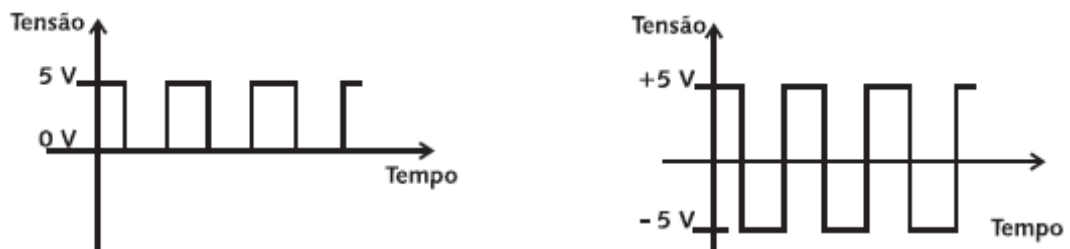
A diferença entre eletrônica analógica e digital é devido ao tipo de sinal processado. O sinal analógico tem como principal característica a de que ele não tem descontinuidades no seu valor, ou seja, não varia bruscamente no tempo. Normalmente um circuito analógico responde a múltiplos níveis de tensão.

A figura abaixo apresenta um sinal analógico variando continuamente no tempo (corrente alternada) e um sinal sem variação no tempo (corrente contínua).



Já o sinal digital apresenta variações descontínuas no tempo, ou seja, normalmente o sinal varia bruscamente entre níveis definidos e conhecidos. Os circuitos digitais baseiam-se na representação de números (dígitos) binários; Portanto, normalmente respondem a apenas dois níveis de tensão, representativos destes números. Os gráficos abaixo demonstram dois sinais digitais: O primeiro varia entre 0 e 5V e o segundo entre -5 e +5 V.

Observe que o sinal não mantém-se entre os dois níveis por tempos que sejam consideráveis.



Os circuitos analógicos e os digitais tem a mesma finalidade, qual seja: processar os sinais de entrada e fornecer sinais de saída. O que varia de um para outro é a maneira de funcionamento. Cada tipo tem suas vantagens e desvantagens.

Atualmente, os circuitos digitais tem avançado em áreas antes dominadas por dispositivos analógicos (como áudio e vídeo, por exemplo), avanço este proporcionado pelo aumento do poder de processamento do circuitos integrados.

2 SISTEMAS DE NUMERAÇÃO

O homem, através dos tempos, sentiu a necessidade da utilização de sistemas numéricos. Existem vários sistemas numéricos, dentre os quais se destacam: o *sistema decimal*, o *binário*, o *octal* e o *hexadecimal*. O sistema decimal é utilizado por nós no dia-a-dia e é, sem dúvida, o mais importante dos sistemas numéricos. Os sistemas: binário, octal e hexadecimal são muito importantes na área de técnicas digitais e computação.

2.1 SISTEMA DECIMAL

O sistema decimal de numeração é composto por 10 símbolos ou dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9; usando tais símbolos, podemos expressar qualquer quantidade. O sistema decimal, também chamado de sistema de base 10, pois ele usa 10 dígitos, evoluiu naturalmente como resultado do fato de os seres humanos, terem 10 dedos. O sistema decimal é um sistema de valor posicional, no qual o valor de um dígito depende de sua posição. Por exemplo, o número 594 significa:

$$\begin{array}{ccccccc} 5 \times 100 & + & 9 \times 10 & + & 4 \times 1 & = & 594 \\ \downarrow & & \downarrow & & \downarrow & & \\ \text{centena} & & \text{dezena} & & \text{unidade} & & \\ \uparrow & & \uparrow & & \uparrow & & \\ 5 \times 10^2 & + & 9 \times 10^1 & + & 4 \times 10^0 & = & 594 \end{array}$$

Neste exemplo podemos notar que o algarismo menos significativo (4) multiplica a unidade (1 ou 10^0), o segundo algarismo (9) multiplica a dezena (10 ou 10^1) e o mais significativo (5) multiplica a centena (100 ou 10^2). A soma desses resultados irá representar o número. Podemos notar ainda, que de maneira geral, a regra básica de formação de um número consiste no somatório de cada algarismo

correspondente multiplicado pela base (no exemplo 10) elevada por um índice conforme o posicionamento do algarismo no número.

2.2 SISTEMA BINÁRIO

No sistema binário de numeração, existem apenas 2 algarismos: **0** (zero) e **1**(um). Por isso sua base é dois. Cada dígito ou algarismo binário é chamado de bit (do inglês “binary digit”, ou seja dígito binário). Um bit é, a menor unidade de informação nos circuitos digitais.

A tabela 01, mostra a correspondência entre números decimais e binários:

DECIMAL	BINÁRIO	DECIMAL	BINÁRIO
0	0000	10	1010
1	0001	11	1011
2	0010	12	1100
3	0011	13	1101
4	0100	14	1110
5	0101	15	1111
6	0110	16	10000
7	0111	17	10001
8	1000	18	10010
9	1001	19	10011

Tabela 01 – Binário x Decimal

Empregando a propriedade do valor de posição do dígito, podemos representar qualquer valor numérico com os dígitos 0 e 1. Como a base de numeração binária é 2, o valor de posição é dado pelas potências de base 2, como mostra a tabela a seguir:

Potências de base 2	2^4	2^3	2^2	2^1	2^0
Valor de posição	16	8	4	2	1

O valor da posição é indicado pelo expoente da base do sistema numérico. Esse valor aumenta da direita para a esquerda. O valor da posição do bit mais significativo (de maior valor) será a base elevada a $m-1$ (m = número de dígitos).

Por exemplo, 101011 é um número binário de 6 bits. Ao aplicar a fórmula, temos $6-1=5$. Assim, o bit mais significativo terá como valor de posição 2^5 .

Valor de posição	2^5	2^4	2^3	2^2	2^1	2^0
Binário	1	0	1	0	1	1

MSB – do inglês “most significant bit” ou seja, bit mais significativo

LSB – do inglês “least significant bit” ou seja, bit menos significativo

2.2.1 Conversão do Sistema Binário para o Sistema Decimal

Para converter um número binário em decimal, deve-se multiplicar cada bit pelo seu valor de posição (que é indicado pelo valor da base) e somar os resultados.

Exemplo:

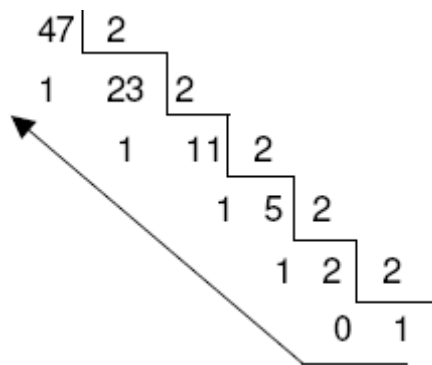
Na conversão de 1010_2 para o sistema decimal, procede-se da seguinte forma:

Potência de 2	2^3	2^2	2^1	2^0	
Binário	1	0	1	0	
Valor de posição	1×8	0×4	1×2	0×1	
Nº decimal	$8 + 0 + 2 + 0 = 10$				

2.2.2 Conversão do Sistema Decimal para o Sistema Binário

A conversão de números do sistema decimal para o sistema binário é realizada efetuando-se divisões sucessivas do número decimal pela base a ser convertida (no caso 2) até o último quociente possível. O número transformado será composto por este último quociente (algarismo mais significativo) e, todos os restos, na ordem inversa às divisões.

Exemplo:



O último quociente será o algarismo mais significativo e ficará colocado à esquerda. Os outros algarismos seguem-se na ordem até o 1º resto:

1	0	1	1	1	1
último	5º	4º	3º	2º	1º
quociente	resto	resto	resto	resto	resto
$101111_2 = 47_{10}$					

2.3 SISTEMA OCTAL

O sistema octal de numeração é um sistema de base 8 no qual existem 8 algarismos: 0, 1, 2, 3, 4, 5, 6 e 7. Para representarmos a quantidade oito, agimos do mesmo modo visto anteriormente para números binários e decimais, colocamos o algarismo 1 seguido do algarismo 0, significando que temos um grupo de oito adicionados a nenhuma unidade.

A tabela 02 mostra a correspondência entre números decimais e octais.

DECIMAL	OCTAL	DECIMAL	OCTAL
0	0	9	11
1	1	10	12
2	2	11	13
3	3	12	14
4	4	13	15
5	5	14	16
6	6	15	17
7	7	16	20
8	10	17	21

Tabela 02 – OCTAL x Decimal

2.3.1 Conversão do Sistema Octal para o Sistema Decimal

Para convertermos um número octal em decimal, utilizamos o conceito básico de formação de um número.

Vamos por exemplo converter o número 144_8 em decimal:

$$\begin{array}{r}
 8^2 \qquad \qquad 8^1 \qquad \qquad 8^0 \\
 \mathbf{1} \qquad \qquad \mathbf{4} \qquad \qquad \mathbf{4} \\
 1 \times 8^2 \quad + \quad 4 \times 8^1 \quad + \quad 4 \times 8^0 = \\
 1 \times 64 \quad + \quad 4 \times 8 \quad + \quad 4 \times 1 = 64 + 32 + 4 = 100_{10}
 \end{array}$$

2.3.2 Conversão do Sistema Decimal para o Sistema Octal

O processo é análogo à conversão do sistema decimal para binário, somente que neste caso, utilizaremos a divisão por 8, pois sendo o sistema octal, sua base é igual a 8.

Para exemplificar, vamos converter o número 92_{10} para o sistema octal:

	92	8		
1º resto	4	11	8	
2º resto		3	1	
último quociente				

$92_{10} = 134_8$

2.3.3 Conversão do Sistema Octal para o Sistema Binário

Vamos usar um número octal qualquer, por exemplo, 27_8 . A regra consiste em transformar cada algarismo diretamente no correspondente em binário,

respeitando-se o número padrão de bits do sistema, sendo para o octal igual a três ($2^3 = 8$, base do sistema octal). Assim sendo, temos:

2	7	
010	111	$27_8 = 10111_2$

Convém lembrar que a regra só é válida entre sistemas numéricos de base múltipla de 2^n , sendo n um número inteiro.

2.3.4 Conversão do Sistema Binário para o Sistema Octal

Para efetuar esta conversão, vamos aplicar o processo inverso ao utilizado na conversão de octal para binário. Como exemplo, vamos utilizar o número 110010_2 . Para transformar este número em octal, vamos primeiramente separá-lo em grupos de 3 bits a partir da direita, e efetuar a conversão de cada grupo de bits diretamente para o sistema octal:

110	010
6	2

O número convertido será composto pela união dos algarismos obtidos.

$$110010_2 = 62_8$$

No caso do último grupo se formar incompleto, adicionamos zeros à esquerda, até completá-lo com 3 bits. Para exemplificar, vamos converter o número 1010_2 em octal:

$$\begin{array}{cc} 001 & 010 \\ 1 & 2 \end{array} \quad 1010_2 = 12_8$$

2.4 SISTEMA HEXADECIMAL

O sistema hexadecimal tem a base 16. Os 16 símbolos que constituem a numeração hexadecimal são os seguintes algarismos e letras: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F.

A tabela 03 a seguir mostra relação entre numeração decimal e hexadecimal

Decimal	Hexa	Decimal	Hexa	Decimal	Hexa
0	0	11	B	22	16
1	1	12	C	23	17
2	2	13	D	24	18
3	3	14	E	25	19
4	4	15	F	26	1A
5	5	16	10	27	1B
6	6	17	11	28	1C
7	7	18	12	29	1D
8	8	19	13	30	1E
9	9	20	14	31	1F
10	A	21	15	32	20

Tabela 03 - Hexadecimal x Decimal

Este sistema é muito utilizado na área dos microprocessadores e também no mapeamento de memórias em sistemas digitais, tratando-se de um sistema numérico muito importante, sendo aplicado em projetos de software e hardware.

2.4.1 Conversão do Sistema Hexadecimal para o Sistema Decimal

A regra de conversão é análoga à de outros sistemas, somente neste caso, a base é 16. Como exemplo, vamos utilizar o número $3F_{16}$ e convertê-lo em decimal:

16^1	16^0
3	F
3×16^1	$F \times 16^0$

Sendo $F_{16} = 15_{10}$, substituindo temos:

$$3 \times 16 + 15 \times 1 = 63_{10} \qquad 3F_{16} = 63_{10}$$

2.4.2 Conversão do Sistema Decimal para o Sistema Hexadecimal

Da mesma forma que nos casos anteriores, esta conversão se faz através de divisões sucessivas pela base do sistema a ser convertido. Para exemplificar vamos transformar o número 1000_{10} em hexadecimal:

	1000	16	
1º resto	8	62	16
2º resto	14	3	
último quociente			

Sendo $14_{10} = E_{16}$, temos: $3E8_{16}$ $1000_{10} = 3E8_{16}$

2.4.3 Conversão do Sistema Hexadecimal para o Sistema Binário

É análoga à conversão do sistema octal para o sistema binário, somente, neste caso, necessita-se de 4 bits para representar cada algarismo hexadecimal. Como exemplo, vamos converter o número $C13_{16}$ para o sistema binário:

$$\begin{array}{r}
 \mathbf{C} \quad (\mathbf{C}_{16} = 12_{10}) \quad \mathbf{1} \quad \quad \quad \mathbf{3} \\
 1100 \quad \quad \quad 0001 \quad \quad \quad 0011 \\
 \\
 \mathbf{C}_{13}_{16} = 110000010011_2
 \end{array}$$

2.4.4 Conversão do Sistema Binário para o Sistema Hexadecimal

É análoga à conversão do sistema binário para o octal, somente que neste caso, agrupamos de 4 em 4 bits da direita para a esquerda. A título de exemplo, vamos transformar o número 10011000_2 em hexadecimal:

$$\begin{array}{r}
 1001 \quad 1000 \quad \quad \quad 10011000_2 = 98_{16} \\
 9 \quad \quad 8
 \end{array}$$

3 ARITMÉTICA BINÁRIA

As operações aritméticas podem ser realizadas com números binários, exatamente da mesma forma como com números decimais. Em alguns casos, porém, certas operações binárias são feitas de modo diferente das suas equivalentes decimais por causa de considerações de hardware.

3.1 ADIÇÃO BINÁRIA

A adição de dois números binários é executada exatamente da mesma maneira que a adição de números decimais. De fato, a adição binária é mais simples, já que há menos casos para aprender. Existem apenas quatro casos que podem ocorrer na adição dos dígitos binários (bits) em qualquer posição.

São eles:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \text{ Vai 1 para a próxima posição (10)}$$

$$1 + 1 + 1 = 1 \text{ Vai 1 para a próxima posição (11)}$$

Este último caso ocorre quando os dois bits em uma dada posição são iguais a 1 e existe um vai- um da posição anterior. Aqui temos exemplos da adição de dois números binários:

$$\begin{array}{r} 011(3) \\ + 110(6) \\ \hline 1001(9) \end{array} \qquad \begin{array}{r} 1001(9) \\ + 1111(15) \\ \hline 11000(24) \end{array}$$

A adição é a operação aritmética mais importante nos sistemas digitais, pois as operações de subtração, multiplicação e divisão, da forma como elas são

Subtração Normal

$$\begin{array}{r} \text{Minuendo} \quad 1100 \\ \text{Subtraendo} \quad - 1001 \\ \hline \text{Diferença} \quad 0011 \end{array}$$

Subtração em complemento de 2

$$\begin{array}{r} \text{Minuendo} \quad 1100 \\ \text{Complemento de 2} \quad + 0111 \\ \hline \text{Soma} \quad 0011 \end{array}$$

Assim, o resultado final é 0011 (decimal 3).

4 FUNÇÕES LÓGICAS

As funções lógicas derivam dos postulados da álgebra de Boole, sendo as variáveis e expressões envolvidas denominadas de booleanas. Nas funções lógicas, temos apenas dois estados distintos:

O estado **0** (zero) e

O estado **1** (um).

O estado 0 representará, Por exemplo: portão fechado, aparelho desligado, ausência de tensão, chave aberta, não, etc. O estado 1 representará, então: portão aberto, aparelho ligado, presença de tensão, chave fechada, sim, etc.

As funções lógicas se dividem em dois grupos:

Funções lógicas básicas (1º grupo):

Função “E” ou “AND”

Função “OU” ou “OR”

Função “NÃO” ou “INVERSORA” ou “NOT” ou “INVERTER”

Funções lógicas derivadas (2º grupo):

Função “NÃO E” ou “NAND”

Função “NÃO OU” ou “NOR”

Função “OU EXCLUSIVO” ou “EXCLUSIVE OR” ou “XOR”

Função “NÃO OU EXCLUSIVO” ou “EXCLUSIVE NOR” ou “XNOR”

4.1 VARIÁVEIS LÓGICAS

As variáveis lógicas são todas as variáveis envolvidas em um circuito digital, podem ser de dois tipos: de entrada e de saída.

4.1.1 Variável lógica de entrada

É uma variável que pode assumir apenas dois valores e pode ser proveniente de uma chave, sensores etc. São injetadas no circuito para serem processadas. São representadas por letras maiúsculas.

4.1.2 Variável lógica de saída

Também pode assumir apenas dois valores lógicos. É resultado das variáveis de entrada processadas pelo circuito lógico. São representadas por letras minúsculas.

Um circuito lógico deve processar os valores lógicos fornecidos por suas entradas e acionar a saída, dependendo das combinações das variáveis de entrada. Estas combinações são demonstradas em uma tabela chamada *tabela verdade*. Cada linha desta tabela corresponde a uma das possíveis combinações das variáveis de entrada. Para determinar o número de combinações possíveis, com determinado número de variáveis de entrada, utilizamos a seguinte equação:

$$C = 2^n$$

Onde:

C = número de combinações

n = número de variáveis de entradas

Exemplo: em um circuito com três variáveis de entrada, será possível 2^3 , combinações.

$$C = 2^3 = 8$$

A tabela verdade a seguir representará as possibilidades de combinações:

A	B	C	y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

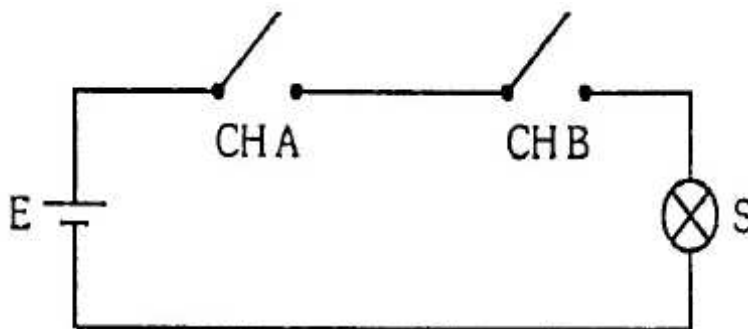
A saída y depende da função que o bloco lógico executa.

4.2 FUNÇÃO E OU AND

A função **E** é aquela que executa a multiplicação de 2 ou mais variáveis booleanas.

É também conhecida como função **AND**, nome derivado do inglês. Sua representação algébrica para 2 variações é $S = A \cdot B$, onde se lê $S = A$ e B .

Para melhor compreensão, vamos utilizar e analisar o circuito representativo da função E visto na figura.



Convenções: chave aberta = 0 chave fechada = 1

Lâmpada apagada = 0 lâmpada acesa = 1

Analisando as situações, concluímos que só teremos a lâmpada acesa quando as chaves A e B estiverem fechadas.

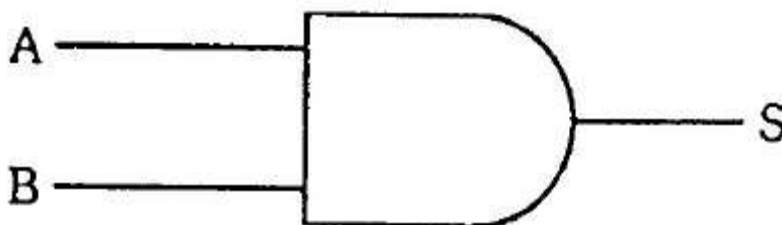
4.2.1 Tabela da Verdade de uma Função E ou AND

Chamamos **Tabela da Verdade** um mapa onde colocamos todas as possíveis situações com seus respectivos resultados. Na tabela, iremos encontrar o modo como a função se comporta. A seguir, iremos apresentar a tabela da verdade de uma função E ou AND para 2 variáveis de entrada:

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

4.2.2 Porta E ou AND

A porta E é um circuito que executa a função E, sendo representada na prática, através do símbolo visto na figura.



Como já dissemos, a porta E executa a tabela da verdade da função E, ou seja, teremos a saída no estado 1 se, e somente se, as 2 entradas forem iguais a 1, e teremos a saída igual a 0 nos demais casos.

Notamos que a tabela da verdade mostra as 4 possíveis combinações das variáveis de entrada e seus respectivos resultados na saída.

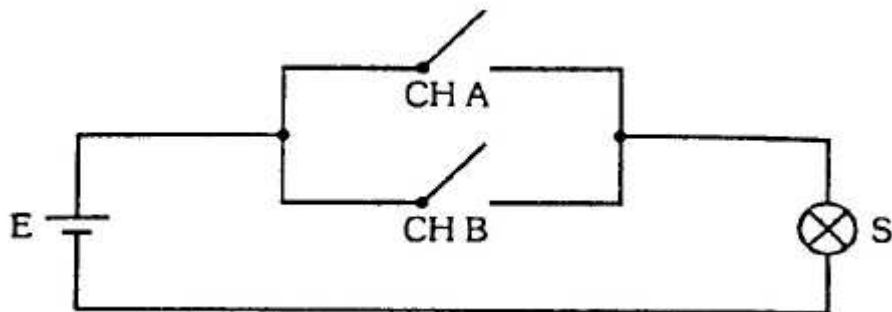
O número de situações possíveis é 2^n , onde n é o número de variáveis de entrada.

Exemplo: $n = 3$ $2^3 = 8$.

4.3 FUNÇÃO OU OU OR

A função OU é aquela que assume valor 1 quando uma ou mais variáveis da entrada forem iguais a 1 e assume valor 0 se, e somente se, todas as variáveis de entrada forem iguais a 0. Sua representação algébrica para 2 variáveis de entrada é $S = A + B$, onde se lê $S = A$ ou B . O termo OR, também utilizado, é derivado do inglês.

Para entendermos melhor a função OU, vamos representá-la através do circuito da figura.



Usaremos as mesmas convenções do circuito representativo da função E, visto anteriormente.

Notamos pelas situações que teremos a lâmpada ligada quando **chA ou chB ou** ambas as chaves estiverem ligadas.

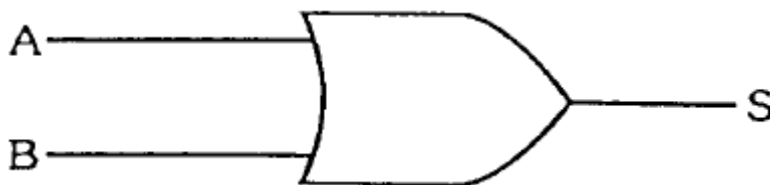
4.3.1 Tabela da Verdade da Função OU ou OR

Nesta tabela da verdade, teremos todas as situações possíveis com os respectivos valores que a função OU assume. A tabela apresenta a tabela da verdade da função OU ou OR para 2 variáveis de entrada.

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

4.3.2 Porta OU ou OR

É a porta que executa a função OU. Representaremos a porta OU através do símbolo visto na figura.



A porta OU executa a tabela da verdade de função OU, ou seja, teremos a saída igual a 1 quando uma ou mais variáveis de entrada forem iguais a 1 e 0 se, e somente se, todas as variáveis de entrada forem iguais a 0.

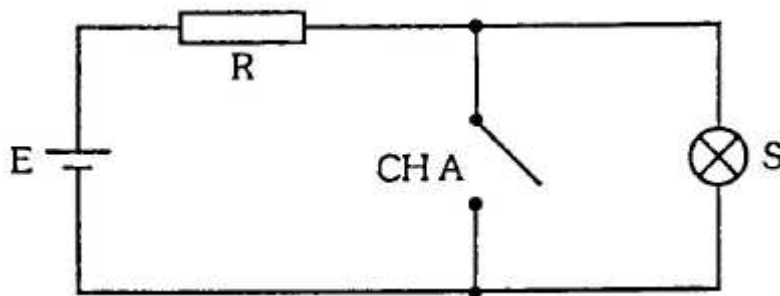
4.4 FUNÇÃO NÃO OU NOT

A função NÃO é aquela que inverte ou complementa o estado da variável, ou seja, se a variável estiver em 0, à saída vai para 1, e se estiver em 1, à saída vai para 0. É representada algebricamente da seguinte forma: $S = \bar{A}$, onde se lê **A barra** ou **NÃO A**.

Esta barra ou apóstrofo sobre a letra que representa a variável significa que esta sofre uma inversão. Também, podemos dizer que \bar{A} significa a negação de A.

Para entendermos melhor a função NÃO vamos representá-la pelo circuito da figura.

Analisaremos utilizando as mesmas convenções dos casos anteriores.



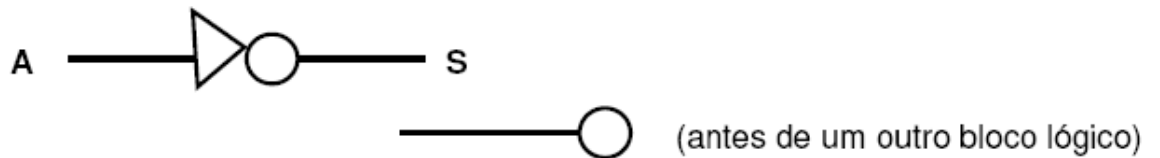
4.4.1 Tabela da Verdade da Função NÃO ou NOT

A tabela apresenta casos possíveis da função NÃO.

A	S
0	1
1	0

4.4.2 Porta Inversora

O inversor é o bloco lógico que executa a função NÃO.
Suas representações simbólicas são vistas na figura.



4.5 FUNÇÃO NÃO E, NE OU NAND.

Como o próprio nome “NÃO E” diz: essa função é uma composição da função E com a função NÃO, ou seja, teremos a função E invertida. É representada algebricamente da seguinte forma:

$$S = (\overline{A \cdot B}), \text{ onde o traço indica que temos a inversão do produto } A \cdot B.$$

4.5.1 Tabela da Verdade da Função NE ou NAND

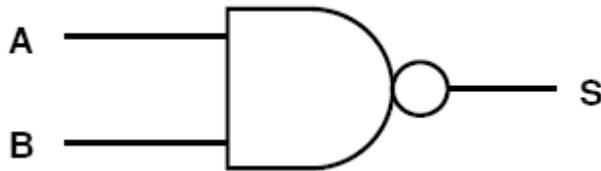
A tabela apresenta a função NE para 2 variáveis de entrada.

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

Pela tabela da verdade, podemos notar que esta função é o inverso da função E.

4.5.2 Porta NE ou NAND

A porta NE é o bloco lógico que executa a função NE. Sua representação simbólica é vista na figura.



4.6 FUNÇÃO NÃO OU, NOU OU NOR.

Analogamente à função NE, a função NOU é a composição da função, NÃO com a função OU, ou seja, a função NOU será o inverso da função OU. É representada da seguinte forma:

$$S = (\overline{A + B}), \text{ onde o traço indica a inversão da soma booleana } A + B.$$

4.6.1 Tabela da Verdade da Função NOU ou NOR

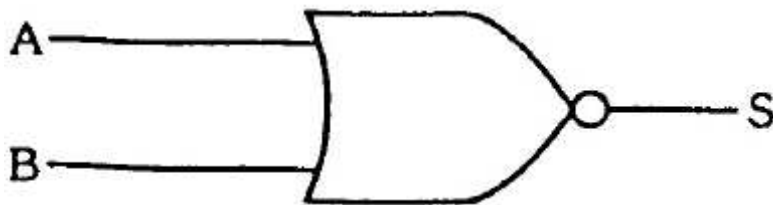
A tabela apresenta a função NOU para 2 variáveis de entrada.

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

Podemos notar pela tabela da verdade que a função NOU representa a função ou invertida.

4.6.2 Porta NOU ou NOR

A porta NOU é o bloco lógico que executa a função NOU. Sua representação simbólica é vista na figura.



4.7 FUNÇÃO OU EXCLUSIVO

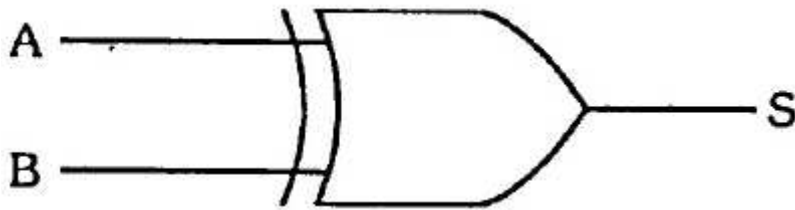
A função que ele executa, como o próprio nome diz, consiste em fornecer 1 à saída quando as variáveis de entrada forem diferentes entre si.

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

Da tabela obtemos sua expressão característica:

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

A notação algébrica que representa a função OU Exclusivo é $S = A \oplus B$, onde se lê A OU Exclusivo B, sendo $S = A \oplus B = \bar{A} \cdot B + A \cdot \bar{B}$. O circuito OU Exclusivo pode ser representado também pelo símbolo visto na figura.



Uma importante observação é que, ao contrário de outros blocos lógicos básicos, o circuito OU Exclusivo só pode ter 2 variáveis de entrada, fato este devido à sua definição básica. O circuito OU Exclusivo também é conhecido como Exclusive OR (XOR), termo derivado do inglês.

4.8 FUNÇÃO COINCIDÊNCIA

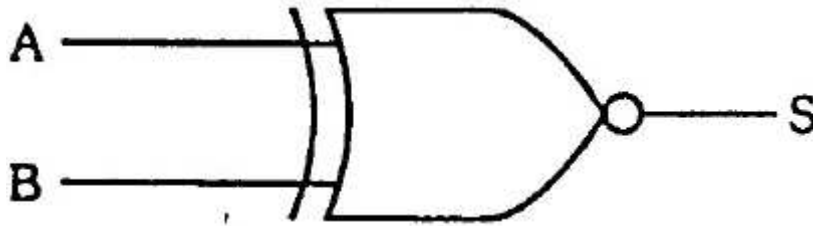
A função que ele executa, como seu próprio nome diz, é a de fornecer 1 à saída quando houver uma coincidência nos valores das variáveis de entrada.

Vamos, agora, montar sua tabela da verdade:

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

A tabela gera a expressão $S = \bar{A} \cdot \bar{B} + A \cdot B$.

A notação algébrica que representa a função Coincidência é $S = A \odot B$, onde se lê A Coincidência B, sendo $S = \bar{A} \cdot \bar{B} + A \cdot B$. O símbolo do circuito Coincidência é visto na figura abaixo:



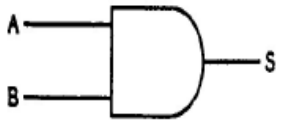

Se compararmos as tabelas da verdade dos blocos OU Exclusivo e Coincidência, iremos concluir que estes são complementares, ou seja, teremos a saída de um invertido em relação à saída do outro. Assim sendo, podemos escrever:

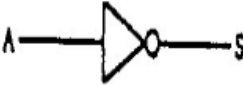


$$A \oplus B = \overline{A \odot B}$$

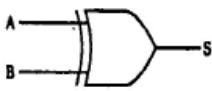
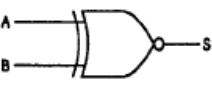
O bloco Coincidência é também denominado de NOU Exclusivo e do inglês Exclusive NOR.

Da mesma forma que o OU Exclusivo, o bloco Coincidência é definido apenas para 2 variáveis de entrada.

4.9 QUADRO RESUMO

BLOCOS LÓGICOS BÁSICOS																			
Porta	Símbolo Usual	Tabela da verdade	Função Lógica	Expressão															
E AND		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	S	0	0	0	0	1	0	1	0	0	1	1	1	Função E: Assume 1 quando todas as variáveis forem 1 e 0 nos outros casos.	$S = AB$
A	B	S																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
OU OR		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	S	0	0	0	0	1	1	1	0	1	1	1	1	Função OU: Assume 0 quando todas as variáveis forem 0 e 1 nos outros casos.	$S = A + B$
A	B	S																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	1																	

NÃO NOT INVERSOR		<table border="1"> <thead> <tr> <th>A</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	S	0	1	1	0	Função NÃO: Inverte a variável aplicada à sua entrada.	$S = \bar{A}$									
A	S																		
0	1																		
1	0																		
NE NAND		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	S	0	0	1	0	1	1	1	0	1	1	1	0	Função NE: Inverso da função E.	$S = \overline{AB}$
A	B	S																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	
NOU NOR		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	0	Função NOU: Inverso da função OU.	$S = \overline{A + B}$
A	B	S																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	

OU EXCLUSIVO EXCLUSIVE OR		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	S	0	0	0	0	1	1	1	0	1	1	1	0	Função OU: Exclusivo – assume 1 quando as variáveis assumirem valores diferentes entre si	$S = \bar{A} \cdot B + A \cdot \bar{B}$ $S = A \oplus B$
A	B	S																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	
NOU EXCLUSIVO EXCLUSIVE NOR COINCIDÊNCIA		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	1	Função Coincidência: Assume 1 quando houver coincidência entre os valores das variáveis	$S = \bar{A} \cdot \bar{B} + A \cdot B$ $S = A \odot B$
A	B	S																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	1																	

5 ÁLGEBRA DE BOOLE

Consiste na aplicação de um conjunto de postulados e teoremas com os quais desenvolvemos a simplificação de uma expressão lógica.

5.1 POSTULADOS

5.1.1 Postulado da complementação: \bar{A} é o complemento de A.

$\overline{\bar{A}} = A$, o inverso da negação é a própria expressão.

5.1.2 Postulado da adição:

$$A + 0 = A$$

$$A + A = A$$

$$A + 1 = 1$$

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

5.1.3 Postulado da multiplicação:

$$A \cdot 0 = 0$$

$$A \cdot A = A$$

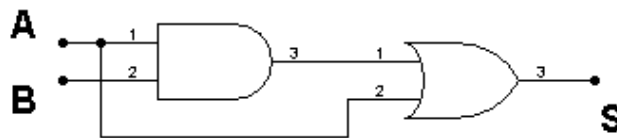
$$A \cdot 1 = A$$

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

5.2 TEOREMA DA ABSORÇÃO (IDENTIDADES AUXILIARES)

$$1 - A + (A \cdot B) = A$$

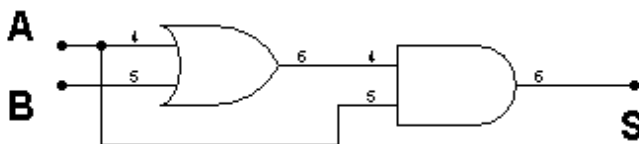
$$1A + A \cdot B = A \cdot (1 + B) = A \cdot 1 = A$$



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

$$2 - A \cdot (A + B) = A$$

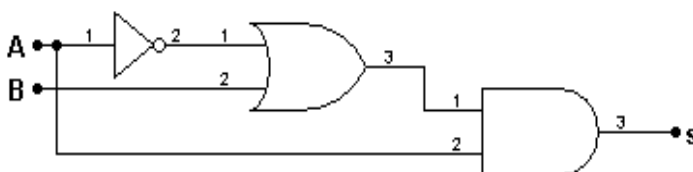
$$A \cdot A + A \cdot B = A + A \cdot B = A$$



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

$$3 - A \cdot (\bar{A} + B) = A \cdot B$$

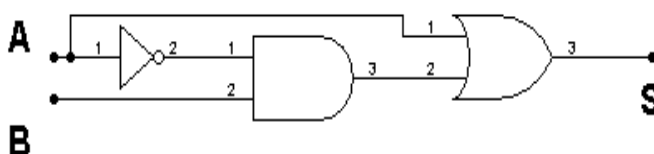
$$A \cdot \bar{A} + A \cdot B = 0 + A \cdot B = A \cdot B$$



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

$$4 - A + (\bar{A} \cdot B) = A + B$$

$$(A + \bar{A}) \cdot (A + B) = 1 \cdot (A + B) = A + B$$

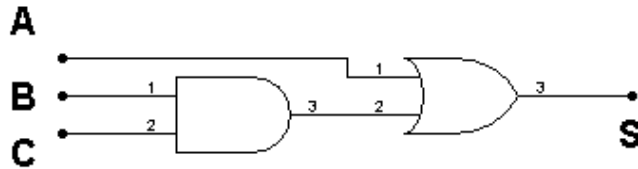


A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

$$5 - (A + B) \cdot (A + C) = A + B \cdot C$$

$$A \cdot A + A \cdot C + A \cdot B + B \cdot C = A + A \cdot C + A \cdot B + B \cdot C = A(1 + B + C) + B \cdot C =$$

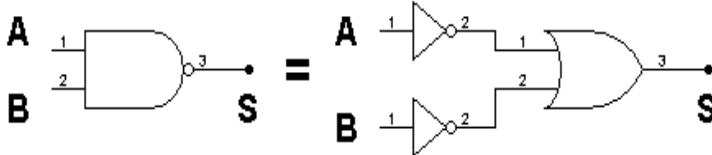
$$A \cdot 1 + B \cdot C = A + B \cdot C$$



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

5.3 TEOREMAS DE DE MORGAN

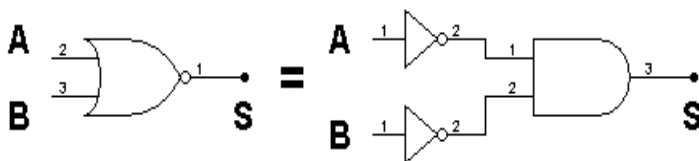
$$1 - \overline{(A \cdot B)} = \bar{A} + \bar{B}$$



A	B	\bar{S}	S
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

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

$$2 - \overline{(A + B)} = \bar{A} \cdot \bar{B}$$



A	B	\bar{S}	S
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

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

Exemplo: A expressão abaixo será simplificada através da álgebra de Boole.

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

Evidenciando o termo A:

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

Aplicando a propriedade associativa:

$$S = A [BC + (\bar{C} + \bar{B})]$$

Aplicando o teorema de De Morgan, temos:

$$S = A [BC + \overline{(\overline{BC})}]$$

Chamando BC de Y, logo $\overline{(\overline{BC})} = \bar{Y}$, temos então:

$$S = A (Y + \bar{Y})$$

Como $Y + \bar{Y} = 1$, logo: $S = A \cdot 1 = A$

5.4 TABELA RESUMO

POSTULADOS		
COMPLEMENTAÇÃO	ADIÇÃO	MULTIPLICAÇÃO
$A = 0 \rightarrow \bar{A} = 1$	$0 + 0 = 0$	$0 \cdot 0 = 0$
$A = 1 \rightarrow \bar{A} = 0$	$0 + 1 = 1$	$0 \cdot 1 = 0$
	$1 + 0 = 1$	$1 \cdot 0 = 0$
	$1 + 1 = 1$	$1 \cdot 1 = 1$

IDENTIDADES		
COMPLEMENTAÇÃO	ADIÇÃO	MULTIPLICAÇÃO
$A = \bar{\bar{A}}$	$A + 0 = A$	$A \cdot 0 = 0$
	$A + 1 = 1$	$A \cdot 1 = A$
	$A + A = A$	$A \cdot A = A$
	$A + \bar{A} = 1$	$A \cdot \bar{A} = 0$

PROPRIEDADES	
COMUTATIVA:	ADIÇÃO: $A + B = B + A$
	MULTIPLICAÇÃO: $A \cdot B = B \cdot A$
ASSOCIATIVA:	ADIÇÃO: $A + (B + C) = (A + B) + C = A + B + C$
	MULTIPLICAÇÃO: $A \cdot (B \cdot C) = (A \cdot B) \cdot C = A \cdot B \cdot C$
DISTRIBUTIVA:	$A \cdot (B + C) = A \cdot B + A \cdot C$

TEOREMA de DE MORGAN	
1°TEOREMA	$\overline{(A \cdot B)} = (\bar{A} + \bar{B})$
2°TEOREMA	$\overline{(A + B)} = \bar{A} \cdot \bar{B}$

IDENTIDADES AUXILIARES	
	$A + A \cdot B = A$
	$A + \bar{A} \cdot B = A + B$
	$(A + B) \cdot (A + C) = A + B \cdot C$

6 MAPAS DE VEITCH – KARNAUGH

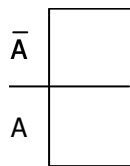
São dispositivos práticos para simplificação de expressões lógicas com até cinco variáveis.

Constitui – se numa forma diferente de escrever uma tabela verdade.

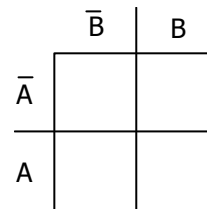
É constituído por 2^n células, onde “n” é o número de variáveis. Cada célula corresponde a uma das alternativas das combinações possíveis das variáveis, e em seu interior indicamos o nível alto ou baixo.

A seguir as configurações dos mapas para até quatro variáveis:

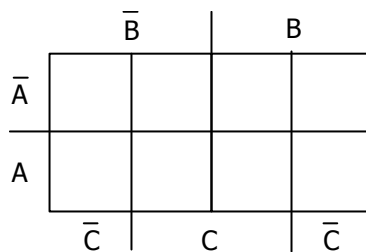
1 – uma variável



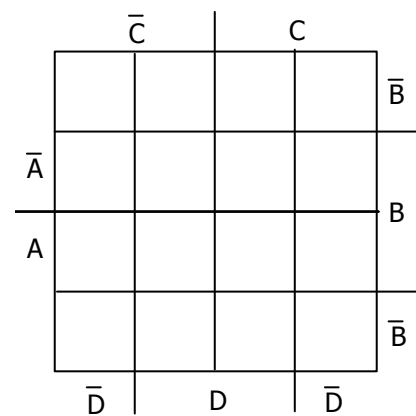
2 – duas variáveis



3 – três variáveis



4 – quatro variáveis



6.1 TÉCNICAS DE SIMPLIFICAÇÃO POR MAPAS

6.1.1 Por Minitermos

Uma vez mapeados os dados, começa-se a “laçar” o maior número de células com “1” formando laços que contenham um número de células potência de 2 e que sejam vizinhas. Esgotados os laços maiores, parte-se para os laços de ordem imediatamente inferior até que todos os “1” tenham sido laçados. Considera-se célula vizinha aquela em que apenas uma variável mudou. A expressão simplificada será dada por uma função “ou” de funções “E” das variáveis que não mudaram.

	\bar{B}	B	
\bar{A}	0	1	$Y_1 = A$ $+$ $Y_2 = B$
A	1	1	

$\left. \begin{array}{l} Y_1 = A \\ + \\ Y_2 = B \end{array} \right\} A + B$

	\bar{B}	B	
\bar{A}	0	1	$Y_1 = \bar{A} \cdot B$ $+$ $Y_2 = A \cdot \bar{B}$
A	1	0	

$\left. \begin{array}{l} Y_1 = \bar{A} \cdot B \\ + \\ Y_2 = A \cdot \bar{B} \end{array} \right\} \bar{A} \cdot B + A \cdot \bar{B} = A \oplus B$

6.1.2 Por Maxitermo

O processo é semelhante ao anterior, ou seja, os “laços” são de “0” e não de 1. Neste caso considera-se verdadeira os “0” e falsos os “1”. Porém desta forma obtemos o complemento da função, ou seja, a saída \bar{S} .

	\bar{B}	B	
\bar{A}	0	0	$S = \bar{B}$ $+$ $S = \bar{A}$
A	0	1	

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

Se formos considerar a simplificação por minitermos, obteremos a expressão:

$$S = A \cdot B$$

Exemplo:

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

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

\bar{C} C \bar{C}

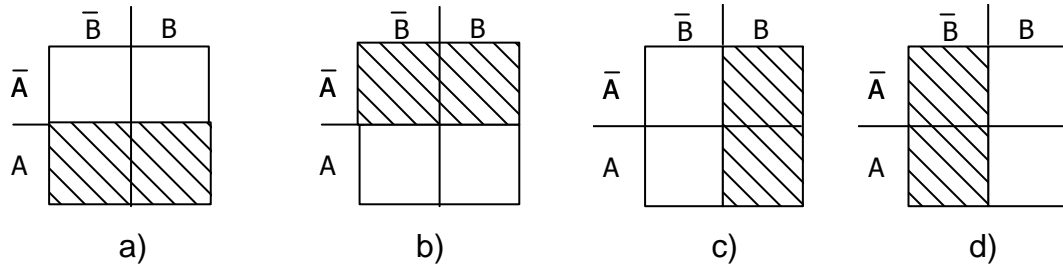
POR MINITERMOS: $S = A+C$

POR MAXITERMOS: $\bar{S} = \bar{A} \cdot \bar{C}$

- Utilizando o teorema de De Morgan temos: $\bar{S} = \overline{A+C}$ $S = \overline{\bar{A} \cdot \bar{C}}$ $S = A + C$

6.2 DIAGRAMA DE VEITCH-KARNAUGH PARA 2 VARIÁVEIS

As possibilidades assumidas pelas variáveis A e B.



- (a) região onde $A=1$.
- (b) região onde $A=0$ ($\bar{A}=1$).
- (c) região onde $B=1$.
- (d) região onde $B=0$ ($\bar{B}=1$).

6.2.1 Transferência da tabela para o mapa

Veremos a forma correta de transferência da tabela verdade para o mapa, devemos observar caso a caso, onde cada linha corresponde a uma combinação possível.

CASO	A	B
0	0	0
1	0	1
2	1	0
3	1	1

	\bar{B}	B
\bar{A}	Caso 0 0 0	Caso 1 0 1
A	Caso 2 1 0	Caso 3 1 1

6.2.2 Formas de agrupamento

a) Quadra

	\bar{B}	B
\bar{A}	1	1
A	1	1

Quadra: $S=1$

b) Pares

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

Par A (está exclusivamente na região A)

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

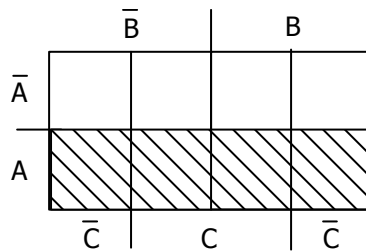
Par \bar{B} (está exclusivamente na região \bar{B})

c) Termos isolados

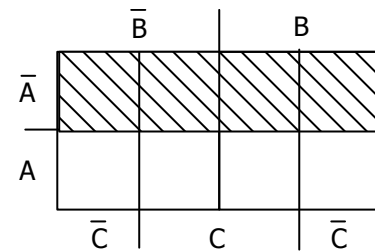
	\bar{B}	B	
\bar{A}	0	1	Termo $\bar{A}B$
A	1	0	Termo $A\bar{B}$

6.3 DIAGRAMA DE VEITCH-KARNAUGH PARA 3 VARIÁVEIS

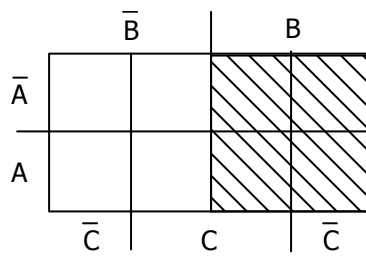
As possibilidades assumidas pelas variáveis A, B e C.



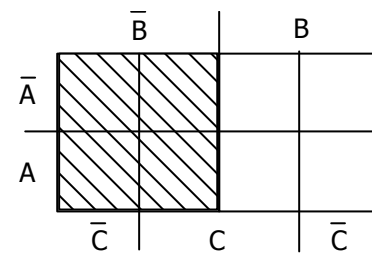
a)



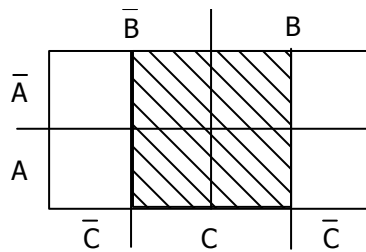
b)



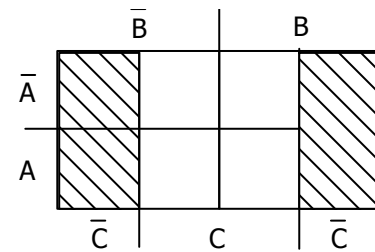
c)



d)



e)



f)

(a) região na qual $A = 1$.

(b) região na qual $\bar{A} = 1$ ($A = 0$).

(c) região na qual $B = 1$.

(d) região na qual $\bar{B} = 1$ ($B = 0$).

(e) região na qual $C = 1$.

(f) região na qual $\bar{C} = 1$ ($C = 0$).

6.3.1 Transferência da tabela para o mapa

CASO	A	B	C
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

	\bar{B}		B	
\bar{A}	Caso 0 0 0 0	Caso 1 0 0 1	Caso 3 0 1 1	Caso 2 0 1 0
A	Caso 4 1 0 0	Caso 5 1 0 1	Caso 7 1 1 1	Caso 6 1 1 0
	\bar{C}	C	\bar{C}	

6.3.2 Formas de agrupamento

a) Oitava

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

Oitava: $S = 1$

b) Quadras

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

Quadra \bar{A} .

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

Quadra \bar{B} .

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

Quadra \bar{C} .

c) Pares

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

Par $\bar{A}\bar{C}$ (está localizado na intersecção das regiões \bar{A} e \bar{C})

Par AC (está localizado na intersecção das regiões A e C)

d) Termos isolados

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

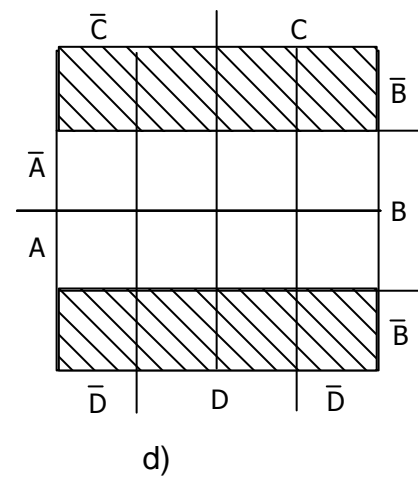
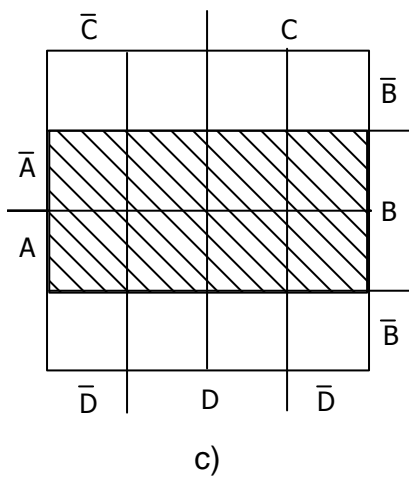
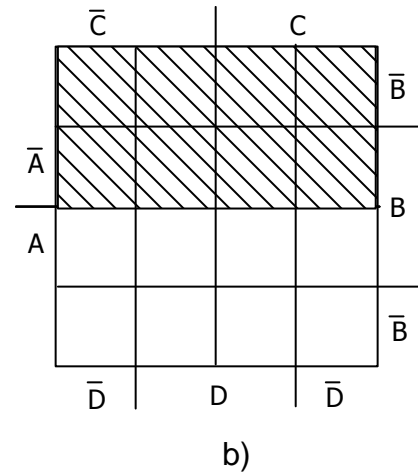
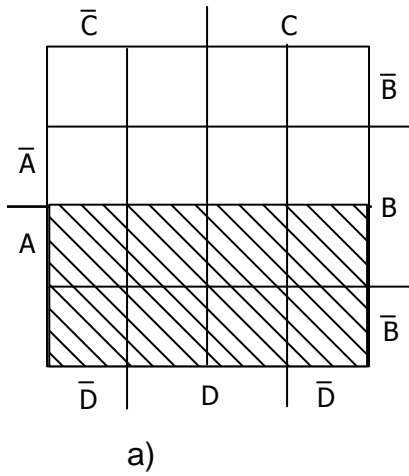
Termo $\bar{A}B\bar{C}$

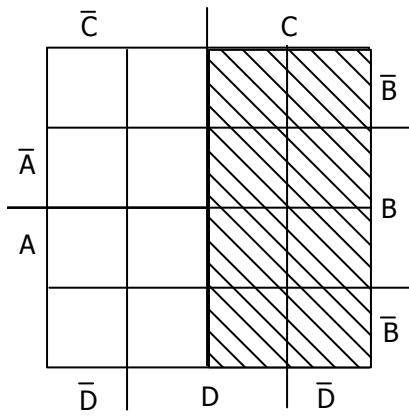
Termo $A\bar{B}C$

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

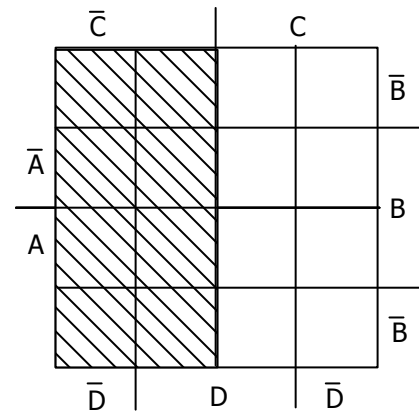
6.4 DIAGRAMA DE VEITCH-KARNAUGH PARA 4 VARIÁVEIS

As possibilidades assumidas pelas variáveis A, B, C e D.

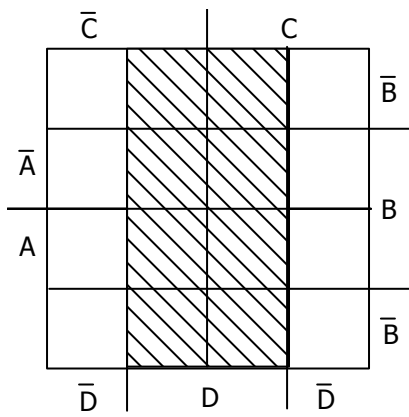




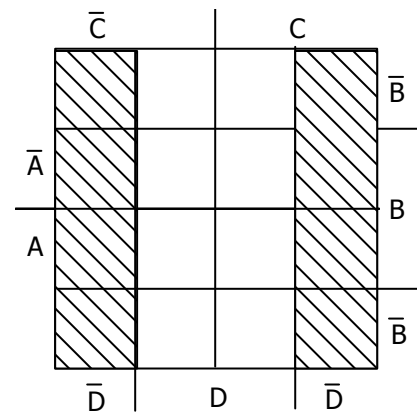
e)



f)



g)



h)

- a) região onde $A = 1$.
- b) região onde $\bar{A} = 1$ ($A = 0$).
- c) região onde $B = 1$.
- d) região onde $\bar{B} = 1$ ($B = 0$).
- e) região onde $C = 1$.
- f) região onde $\bar{C} = 1$ ($C = 0$).
- g) região onde $D = 1$.
- h) região onde $\bar{D} = 1$ ($D = 0$).

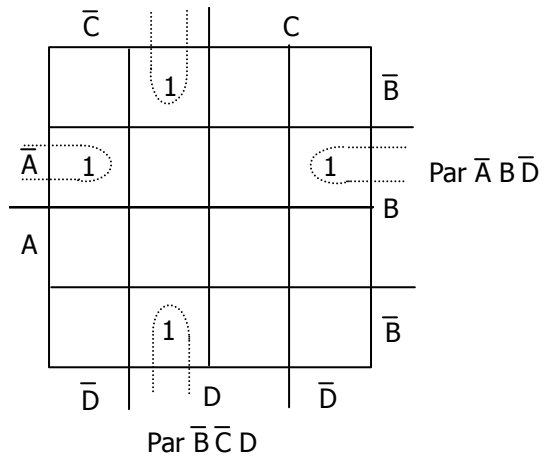
6.4.1 Transferência da tabela para o mapa

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

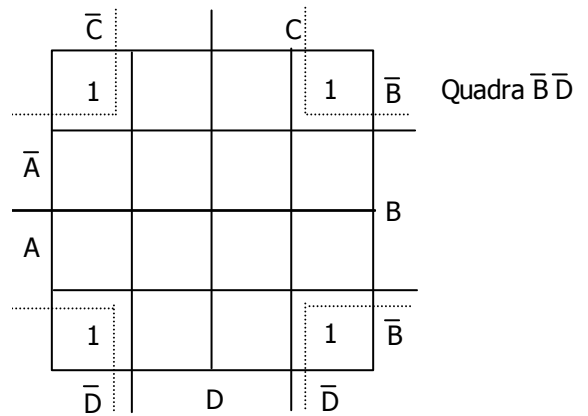
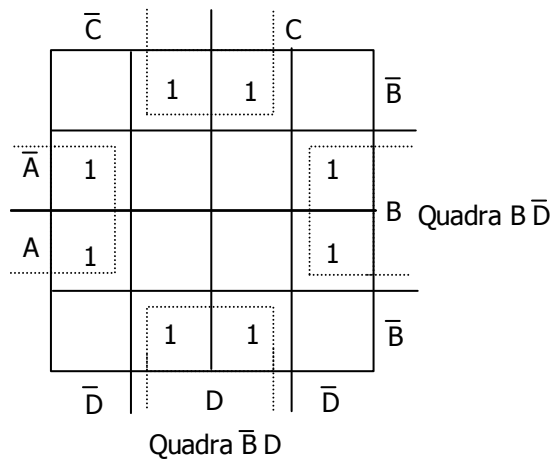
		\bar{C}	C		
\bar{A}	Caso 0 0 0 0 0	Caso 1 0 0 0 1	Caso 3 0 0 1 1	Caso 2 0 0 1 0	\bar{B}
	Caso 4 0 1 0 0	Caso 5 0 1 0 1	Caso 7 0 1 1 1	Caso 6 0 1 1 0	B
A	Caso 12 1 1 0 0	Caso 13 1 1 0 1	Caso 15 1 1 1 1	Caso 14 1 1 1 0	
	Caso 8 1 0 0 0	Caso 9 1 0 0 1	Caso 11 1 0 1 1	Caso 10 1 0 1 0	\bar{B}
		\bar{D}	D	\bar{D}	

6.4.2 Formas de agrupamento

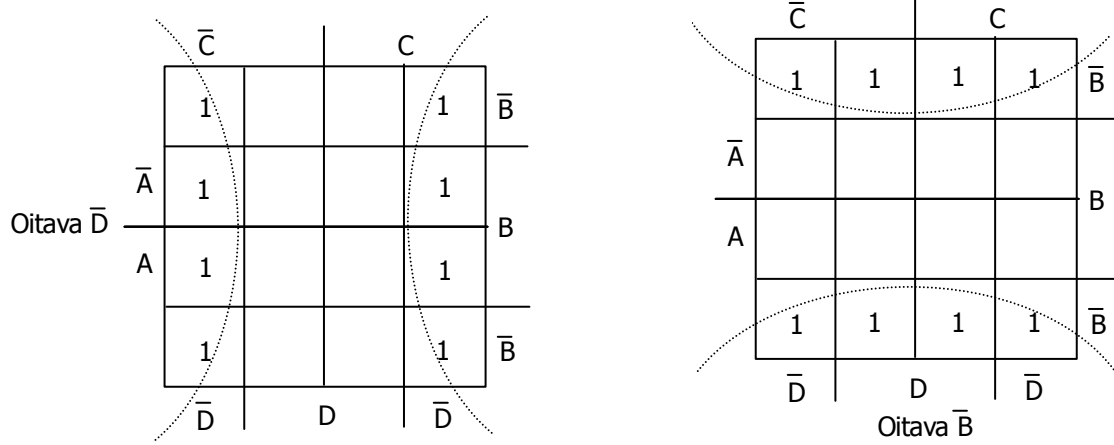
a) Pares



b) Quadras

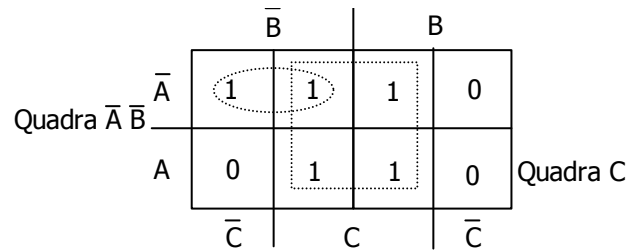


c) Oitavas



Exemplo:

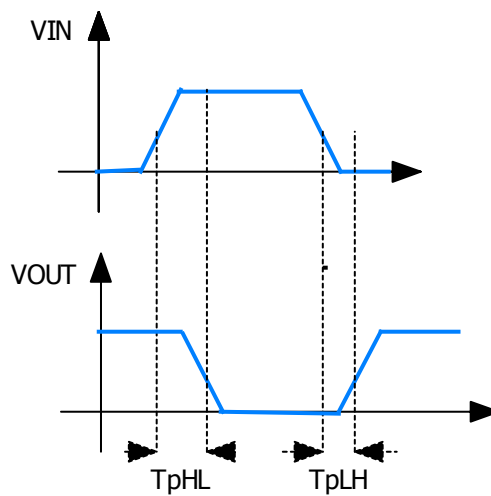
A	B	C	S
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A expressão minimizada será: $S = \bar{C} + \bar{A} B$.

7 PARÂMETROS DOS CIRCUITOS LÓGICOS

7.1 ATRASO DE PROPAGAÇÃO

A tensão de saída de uma porta nunca responde instantaneamente às variações de Entrada. Há sempre um, certo atraso associado à porta lógica.



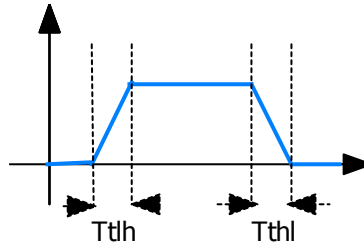
TpHL – Tempo de Propagação de Alto – Baixo: é o tempo de transição do estado lógico 1 para o estado lógico 0, considerando sinais de entrada e saída. O TpHL é tipicamente 7,5ns.

TpLH – Tempo de Propagação de Baixo – Alto: é o tempo de transição do estado lógico 0 para estado lógico 1, considerando sinais de entrada e saída. O TpLH é tipicamente igual a 11ns.

O atraso de propagação é especificado como a média aritmética entre TpHL e TpLH, é um fator que limita a aplicação de um CI, pois se a entrada varia de modo excessivamente rápido, a saída não consegue acompanhar a entrada.

7.2 ATRASO DE TRANSIÇÃO

É aquele causado pela troca do estado lógico 0 para o estado lógico 1 e vice-versa.



Tth- é o atraso de subida e Tthl - é o atraso de descida.

A soma dos dois tempos caracteriza um dado importante para determinação de frequência máxima de operação.

Fan – in: é um parâmetro utilizado para definir o número de saídas que podem ser ligadas a uma determinada entrada. Neste caso as saídas precisam ser obrigatoriamente OPEN – COLLECTOR (COLETOR ABERTO).

Fan – out: indica o número máximo de entradas de outras portas que podem ser ligadas na saída de uma porta lógica.

Frequência máxima de operação: é definida como a máxima frequência que um circuito suporta em sua entrada sem distorcer o sinal de saída.

Seu cálculo é feito baseado nos atrasos por transição:

$$f_{m\acute{a}x} = \frac{1}{(T_{th} + T_{thl})}$$

7.3 MARGEM DE RUÍDO

É um termo designado para indicar o máximo ruído (em mV) que uma porta lógica suporta em sua entrada sem degenerar o sinal.

8 FAMÍLIAS LÓGICAS DE 1º GRUPO

São diferentes tipos de estruturas internas utilizadas na formação de um bloco lógico. Analisando a estrutura interna de um CI, iremos nos deparar com arranjos físicos equivalentes aos componentes discretos.

Dependendo de como é feito este arranjo físico, o circuito irá pertencer a uma ou outra família lógica apresentada:

RTL – LÓGICA COM RESISTORES E TRANSISTORES

DL – LÓGICA COM DIODOS

DTL – LÓGICA COM DIODOS E TRANSISTORES

HTL – LÓGICA COM ALTA IMUNIDADE A RUÍDOS

RCTL – LÓGICA COM RESISTORES, CAPACITORES e TRANSISTORES

Este grupo de famílias lógicas caiu em desuso por ter sido superada pelo segundo grupo.

Cada família lógica possui características peculiares, por isto sua classificação é importante.

A tecnologia de concepção e produção dos CI's está em constante evolução, o que faz com que uma família seja substituída por outra após alguns anos em evidência.

9 FAMÍLIAS LÓGICAS DE 2º GRUPO

TTL – LÓGICA TRANSISTOR – TRANSISTOR

MOS – METAL – ÓXIDE – SEMICONDUCTOR

CMOS – MOS COMPLEMENTAR

ECL – LÓGICA COM ACOPLAMENTO PELO EMISSOR

IIL – LÓGICA COM INJEÇÃO INTEGRADA

HCT MOS – CMOS DE ALTA VELOCIDADE

Dos anos 70 em diante as famílias mais utilizadas são a TTL, CMOS e HCT MOS.

9.1 FAMÍLIA TTL

Os circuitos TTL são produzidos em duas séries comerciais: a série 74XX e a 54 XXX, sendo esta última denominada série militar ou profissional, devido à maior margem de variação nas especificações de alimentação e temperatura, assegurando a confiabilidade no desempenho em condições máximas.

Alimentação: temos para todos os blocos uma alimentação de 5V. Para a série 54 temos V_{cc} mínimo = 4,5V e V_{cc} máximo = 5,5V que são valores dentro da especificação militar de 10% de tolerância. Para a série 74, temos V_{cc} mínimo de 4,75V e V_{cc} máximo = 5,25V que são valores dentro da especificação comum de 5% de tolerância.

Fan-out: Na versão padrão, o Fan-out é igual a 10, ou seja, pode-se ligar à saída destes blocos, no máximo outros 10 blocos.

Tempo de atraso de propagação: Em média de 10ns.

Imunidade ao ruído: De maneira geral é igual a 0,4V, e é considerada baixa

em relação à CMOS.

Potência Dissipada: É da ordem de 10mW por porta.

Tipos: Podemos destacar os blocos open-collector, tri-state e schmitt-trigger.

9.1.1 Saída TOTEM POLEM

Características: alta velocidade de comutação, não permite ligar saídas simultaneamente ao mesmo ponto.

Vantagens: alta velocidade, custo reduzido, não necessita de resistor externo.

Desvantagens: não permite conexão simultânea de saídas.

9.1.2 Saída OPEN – COLLECTOR

Características: permite ligação de mais de uma saída ao mesmo ponto, necessita de um resistor externo, menor velocidade de comutação, WIRE – AND, reduz o número de portas lógicas, estado lógico de saída igual ao estado na conexão.

Vantagens: permite a conexão de várias saídas ao mesmo ponto, reduz o número de portas lógicas e custo;

Desvantagens: menor velocidade de comutação em relação a TOTEM – POLEM, necessita de resistor externo, é necessário dimensionar o resistor externo

9.1.3 Saída THREE – STATE

Características: apresenta três estados lógicos, possui um transistor para habilitação (ENABLE), velocidade elevada, aplicado para barramento de dados, saídas podem ser conectadas juntas, custo elevado e menor número de portas no CI, só é utilizada quando apresenta nítida vantagem.

Vantagens: oferece estado de alta impedância, velocidade de comutação elevada, saídas podem ser conectadas juntas.

Desvantagens: menor número de portas de saída, custo elevado, sincronismo: as saídas podem ser conectadas juntas desde que só uma delas esteja habilitada a cada instante.

9.2 FAMÍLIA CMOS

Os circuitos CMOS são construídos por transistores MOS-FET complementares do tipo canal N e canal P. Suas configurações básicas permitem obter-se uma série de vantagens, tais como: alto Fan-Out, alta margem de imunidade ao ruído e baixíssimo consumo, sendo esta uma das principais características.

Alimentação: temos para as séries 4000 e 74C, a faixa de 3V a 15V, para a versão HC, a faixa de 2V a 6V e para a HCT de 4,5V a 5,5V. Para as séries de baixa tensão, a faixa de 1V a 3,6V para a LV, e 1,2V a 3,6V para a LVC.

Fan-out: Na versão padrão, o Fan-out é igual a 50, porém varia conforme as versões empregadas.

Tempo de atraso de propagação: Em média de 90ns.

Imunidade ao ruído: De maneira geral é igual 45% de V_{dd} (tensão de alimentação).

Potência Dissipada: É da ordem de 1nW por porta da série 4000 e 2,5nW na

série 74HC, a uma tensão de alimentação de 5V.

Manuseio: Esta família necessita, ao contrário da TTL, um cuidado extra no manuseio dos circuitos integrados, que devido à eletricidade estática, provoca a degradação das junções internas dos chips, comprometendo sua vida útil.

10 CÓDIGOS NUMÉRICOS

Os DISPOSITIVOS DIGITAIS podem processar somente níveis “0” e “1”, é difícil ao homem interpretar estas longas séries. Por esta razão os conversores de código foram criados para converter em uma linguagem acessível às pessoas para a linguagem de Máquina e vice-versa.

Dentre os vários códigos existentes podemos citar os principais: CÓDIGO BCD 8421, CÓDIGO OCTAL, CÓDIGO HEXADECIMAL, CÓDIGO ASCII, CÓDIGO EXCESSO 3, CÓDIGO GRAY.

10.1 CÓDIGO BCD 8421 (BINARY CODE DECIMAL)

É o mais utilizado entre os códigos existentes, tem como base a utilização de 4 bits (NIBBLE) para representar cada dígito decimal. Existem 16 combinações possíveis, sendo utilizadas as primeiras 10 combinações.

BCD	DECIMAL	BCD	DECIMAL
0000	0	0101	5
0001	1	0110	6
0010	2	0111	7
0011	3	1000	8
0100	4	1001	9

A conversão dá-se pela troca do número decimal por quatro bits.

EX: 187= 0001 1000 0111

10.2 CÓDIGO OCTAL

Faz-se por agrupamento de 3 bits estes agrupamentos são feitos sempre à direita para a esquerda. Após feito agrupamentos no sentido correto executa-se a leitura de cada grupo transformando-o em um dígito decimal.

No caso inverso, cada dígito decimal corresponde a um grupo de 3 bits.

10.3 CÓDIGO HEXADECIMAL

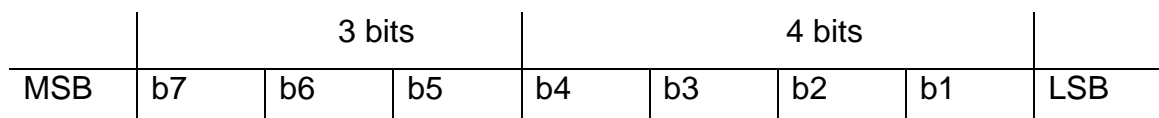
Aplica-se o código HEXADECIMAL com agrupamento de 4 bits utilizando-se as letras: A, B, C, D, E e F, para representar os números binários de 1010 até 1111.

10.4 CÓDIGO ASC II “AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE”

Código padrão Americano para intercâmbio de informações. Permite a troca de informações entre computadores e seus sistemas periféricos. É constituído por um conjunto de caracteres, que podem ser números, símbolos especiais, letras, ou símbolos de controle, codificados em sete bits.

Cada caracter é codificado em dois grupos de bits: Um grupo de três bits e outro de quatro bits.

Desta maneira o formato do caracter do ASC II fica assim estabelecido:



Observe que os sete bits permitem a representação de até 128 caracteres.

$$2^7 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 128$$

10.4.1 Tabela ASCII

A Tabela ASCII (American Standard Code for Information Interchange) é usada pela maior parte da indústria de computadores para a troca de informações. Cada caracter é representado por um código de 8 bits (um byte). Abaixo mostramos a tabela ASCII de 7 bits. Existe uma tabela estendida para 8 bits que inclui os caracteres acentuados.

CÓDIGO ASC II				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1									
0	0	0	0	NUL	DLE	SP	0	@	P	`	p	
0	0	0	1	SOH	DC1	!	1	A	Q	a	q	
0	0	1	0	STX	DC2	"	2	B	R	b	r	
0	0	1	1	ETX	DC3	#	3	C	S	c	s	
0	1	0	0	EOT	DC4	\$	4	D	T	d	t	
0	1	0	1	ENQ	NAR	%	5	E	U	e	u	
0	1	1	0	ACK	SYN	&	6	F	V	f	v	
0	1	1	1	BEL	ETB	'	7	G	W	g	w	
1	0	0	0	BS	CAN	(8	H	X	h	x	
1	0	0	1	HT	EM)	9	I	Y	i	y	
1	0	1	0	LF	SUB	*	:	J	Z	j	z	
1	0	1	1	VT	ESC	+	;	K	[k	{	
1	1	0	0	FF	FS	,	<	L	\	l		
1	1	0	1	CR	GS	-	=	M]	m	}	
1	1	1	0	SO	RS	.	>	N	^	n	~	
1	1	1	1	SI	US	/	?	O	_	o	DEL	

10.5 CÓDIGOS EXCESSO 3 (EX 3, XS3)

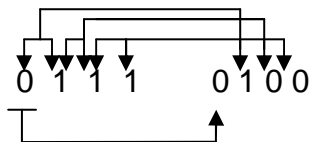
Parte do código binário acrescido na primeira linha da tabela com o valor três. É convertido, normalmente, para o decimal.

A	B	C	D	DECIMAL
0	0	1	1	0
0	1	0	0	1
0	1	0	1	2
0	1	1	0	3
0	1	1	1	4
1	0	0	0	5
1	0	0	1	6
1	0	1	0	7
1	0	1	1	8
1	1	0	0	9

10.6 CÓDIGO GRAY

Este código não representa um valor direto. Sua característica principal é a mudança de apenas um caracter de uma linha para a outra.

É obtido a partir do binário, executando a adição binária, conforme o exemplo:



O código não considera o transporte.

Este código é utilizado na conversão analógica para digital e sua característica de variação de um bit minimiza o erro durante o processo de conversão.

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

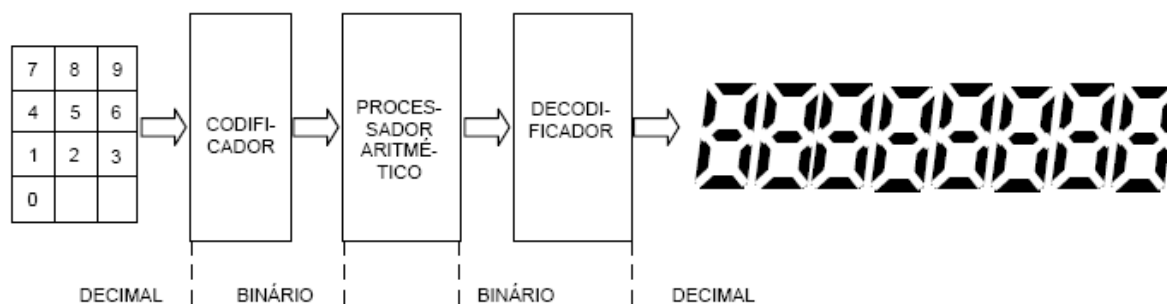
11 CODIFICADORES E DECODIFICADORES

Vamos, agora, tratar de circuitos que efetuam a passagem de um determinado código para outro. Primeiramente, vamos fazer uma análise do significado das palavras codificador e decodificador.

Chamamos de **codificador** o circuito combinacional que torna possível a passagem de um código conhecido para um desconhecido. Como exemplo, podemos citar o circuito inicial de uma calculadora que transforma uma entrada decimal, através do sistema de chaves de um teclado, em saída binária para que o circuito interno processe e faça a operação.

Chamamos de **decodificador** o circuito que faz o inverso do codificador, ou seja, passa um código desconhecido para um conhecido. No exemplo citado é o circuito que recebe o resultado da operação em binário e o transforma em saída decimal, na forma compatível para um mostrador digital apresentar os algarismos.

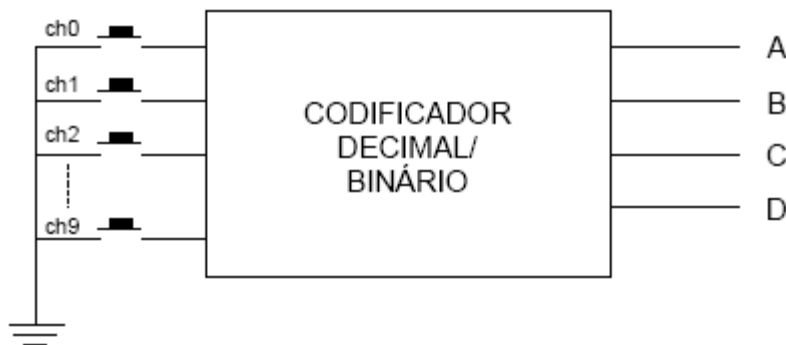
A figura ilustra o exemplo utilizado.



Os termos codificador e decodificador, porém, diferenciam-se em função do referencial. Se para o usuário da calculadora o sistema de entrada é um codificador, para o processador será um decodificador, pois passa de um código desconhecido para ele (decimal), para um conhecido (binário). Na prática, é comum se utilizar a denominação de decodificador para o sistema que passa de um código para outro, quaisquer que sejam.

11.1 CODIFICADOR DECIMAL/BINÁRIO

Vamos, neste item, elaborar um codificador para transformar um código decimal em binário (BCD8421). A entrada do código decimal vai ser feita através de um conjunto de chaves numeradas de 0 a 9 e a saída por 4 fios, para fornecer um código binário de 4 bits, correspondente à chave acionada. A figura a seguir mostra a estrutura geral deste sistema, sendo convencionado que a chave fechada equivale a nível 0, para evitar o problema prático, principalmente da família TTL, que um terminal de entrada em vazio é equivalente a nível lógico 1.



A seguir, vamos construir a tabela da verdade do codificador que relaciona cada chave de entrada decimal com a respectiva saída em binário:

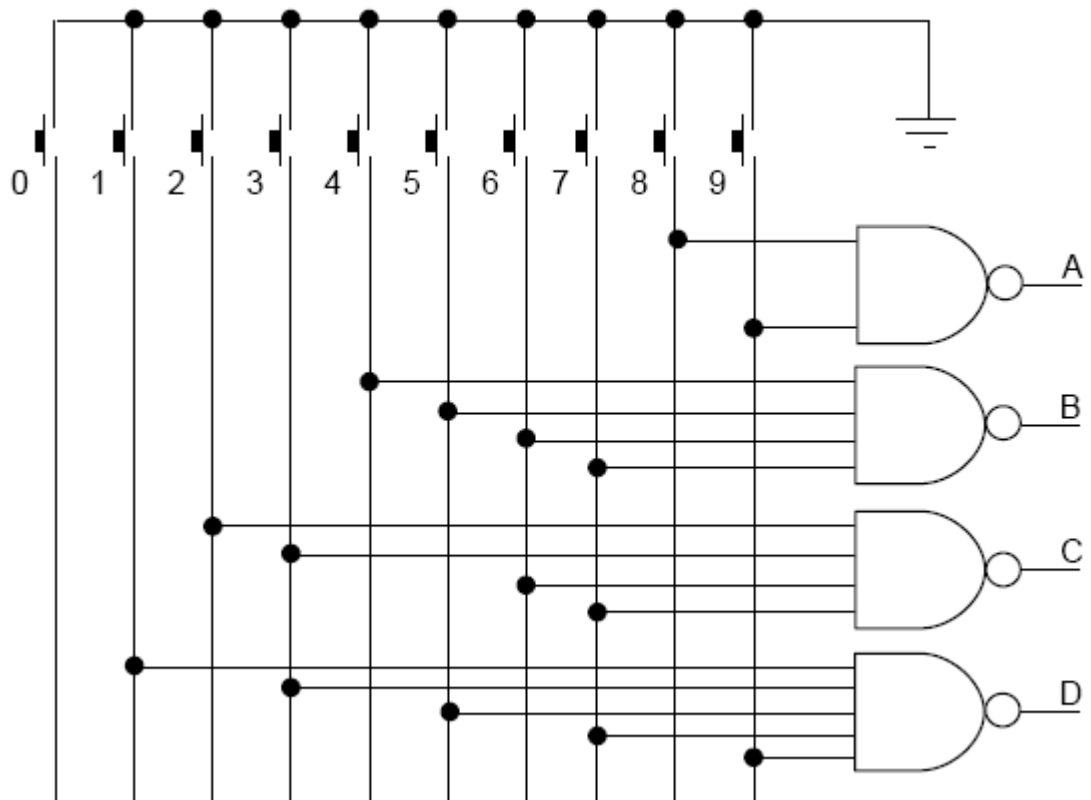
Chave	A	B	C	D
Ch0	0	0	0	0
Ch1	0	0	0	1
Ch2	0	0	1	0
Ch3	0	0	1	1
Ch4	0	1	0	0
Ch5	0	1	0	1
Ch6	0	1	1	0
Ch7	0	1	1	1
Ch8	1	0	0	0
Ch9	1	0	0	1

Através da tabela, concluímos que a saída A valerá 1 quando Ch8 ou Ch9 for acionada. A saída B quando Ch4, Ch5, Ch6 ou Ch7 for acionada. A saída C quando

Ch2, Ch3, Ch6 ou Ch7 for acionada. A saída D quando Ch1, Ch3, Ch5, Ch7 ou Ch9 for acionada.

Usaremos para a construção do circuito, uma porta NE em cada saída, pois esta fornece nível 1 quando qualquer uma de suas entradas assumir nível 0, situação compatível com a convenção adotada para o conjunto de chaves. A ligação das entradas de cada porta será feita, conforme a análise efetuada, às chaves responsáveis pelos níveis 1 de cada saída.

Pela figura, notamos que a chave Ch0 não está ligada a nenhuma das entradas das portas, sendo irrelevante o seu acionamento, pois a saída também será igual a 0 ($A = B = C = D = 0$) quando nenhuma das chaves for acionada.



11.2 DECODIFICADOR BINÁRIO/DECIMAL

A estrutura geral deste decodificador é vista na figura abaixo:



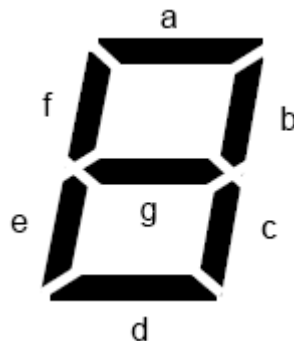
Vamos montar a tabela da verdade do circuito no qual as entradas são bits do código BCD 8421 e as saídas são os respectivos bits do código decimal 9876543210.

<i>BCD 8421</i>				<i>Código 9876543210</i>									
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>S9</i>	<i>S8</i>	<i>S7</i>	<i>S6</i>	<i>S5</i>	<i>S4</i>	<i>S3</i>	<i>S2</i>	<i>S1</i>	<i>S0</i>
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0

O código BCD 8421 não possui números maiores que 9, logo, tanto faz o valor assumido nas possibilidades excedentes, visto que, quando passarmos do código BCD 8421 para o código decimal estas não irão ocorrer.

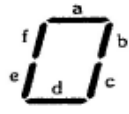

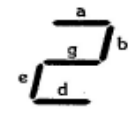

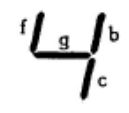
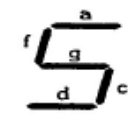
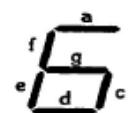
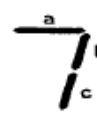
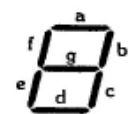
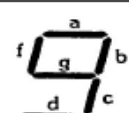
11.3 DECODIFICADOR PARA DISPLAY DE 7 SEGMENTOS

O *display de 7 segmentos* possibilita escrevermos números decimais de 0 a 9 e alguns outros símbolos que podem ser letras ou sinais. A figura a seguir representa uma unidade do display genérica, com a nomenclatura de identificação dos segmentos usual em manuais práticos.



Entre as tecnologias de fabricação das unidades de display usaremos o mais comum que é o display a led, que possui cada segmento composto por um led, existindo um tipo denominado *catodo comum* e outro *anodo comum*.

O display tipo catodo comum é aquele que possui todos os catodos dos led's interligados, sendo necessário aplicar nível 1 no anodo respectivo para acender cada segmento. Já o de anodo comum possui todos os anodos interligados, sendo preciso aplicar nível 0 ao catodo respectivo.

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

Vamos a título de exemplo, elaborar um decodificador para a partir de um código binário (BCD 8421) escrever a seqüência de 0 a 9 em um display de 7 segmentos catodo comum. O esquema geral deste decodificador é visto na figura abaixo:



Para efetuar o projeto deste decodificador, devemos verificar em cada caracter, os segmentos que devem ser acesos e atribuir o nível 1 (no caso do catodo comum), em função da respectiva entrada no código binário. A tabela a seguir apresenta a seqüência de caracteres, o respectivo código de entrada, e os níveis aplicados em cada segmento para que tal ocorra.

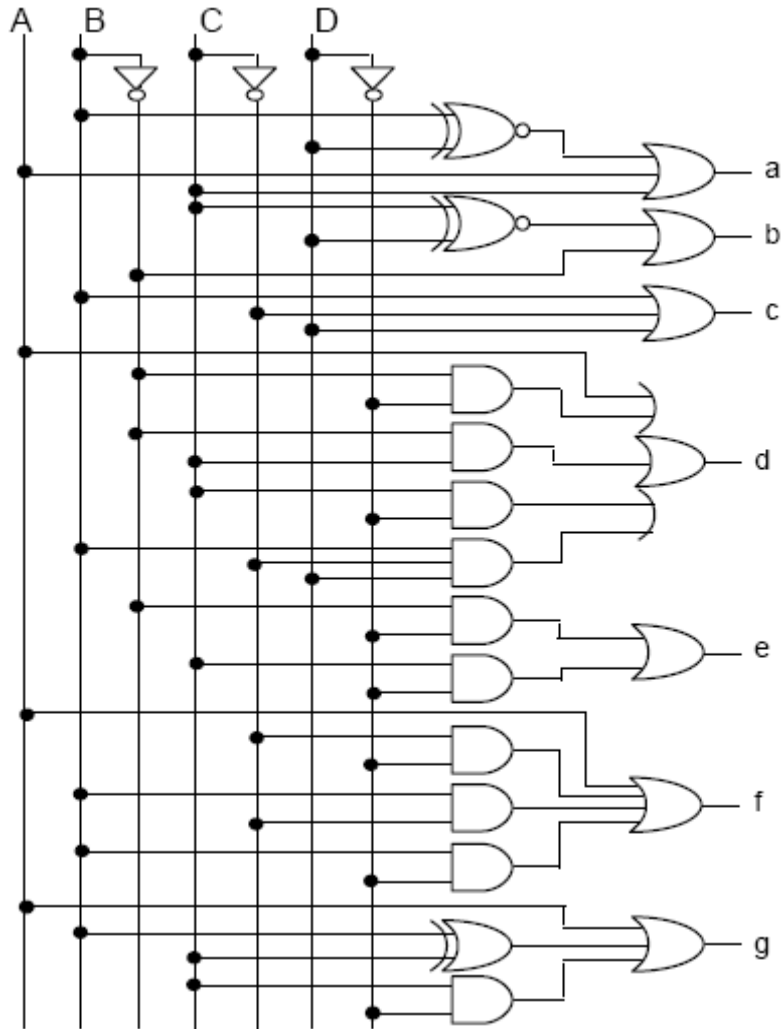
Carac- teres	BCD 8421				Código para 7 segmentos						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0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

Para fins de simplificação, vamos considerar os casos fora da seqüência como irrelevantes. Transpondo as saídas para os diagramas de Karnaugh, temos após simplificação:

- a) $A + C + B \odot D$
- b) $\bar{B} + C \odot D$
- c) $B + \bar{C} + D$
- d) $A + \bar{B} \bar{D} + \bar{B} C + C \bar{D} + B \bar{C} D$
- e) $\bar{B} \bar{D} + C \bar{D}$
- f) $A + \bar{C} \bar{D} + B \bar{C} + B \bar{D}$
- g) $A + B \oplus C + C \bar{D}$

O circuito do decodificador BCD 8421 para display de 7 segmentos obtido, é visto na figura abaixo:

Convém observar que o circuito poderia ser otimizado, pois as expressões dos segmentos possuem vários termos em comum, resultando no emprego de um menor número de portas. Porém, para melhor clareza didática, este foi deixado na sua forma original de acordo com as expressões extraídas dos diagramas.



Circuito Equivalente do Decodificador para Display de 7 segmentos

Outro ponto a ser realçado é que numa montagem prática, a ligação do display se faz, conforme a família lógica, através de resistores para observar os limites máximos de corrente nos led's. Os displays de 7 segmentos podem ainda escrever outros caracteres, que são freqüentemente utilizados em sistemas digitais para representar outras funções, bem como formar palavras-chave em software de programação.

Para efetuar o projeto, basta verificar caso a caso quais segmentos, devem acender e montar assim a tabela da verdade.

12 CIRCUITOS ARITMÉTICOS

Dentro do conjunto de circuitos combinacionais aplicados para finalidades específica nos sistemas digitais, destacam-se os circuitos aritméticos. São utilizados, principalmente, para construir a **ULA (Unidade Lógica Aritmética)** dos microprocessadores e, ainda, encontrados disponíveis em circuitos integrados comerciais. Neste tópico, abordamos os principais circuitos aritméticos e seus subsistemas derivados.

12.1 MEIO SOMADOR

Antes de iniciarmos o assunto, vamos relembrar alguns tópicos importantes da soma de 2 números binários:

$$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array} \quad \begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array} \quad \begin{array}{r} 11 \\ + 1 \\ \hline 10 \end{array}$$

↳ Transporte

Após essa breve introdução, vamos montar uma tabela da verdade da soma de 2 números binários de 1 algarismo:

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

Ts → transporte de saída

$$(0 + 0 = 0 \rightarrow Ts = 0)$$

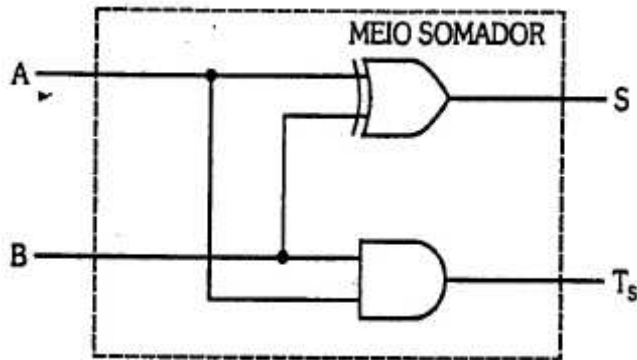
$$(0 + 1 = 1 \rightarrow Ts = 0)$$

$$(1 + 0 = 1 \rightarrow Ts = 0)$$

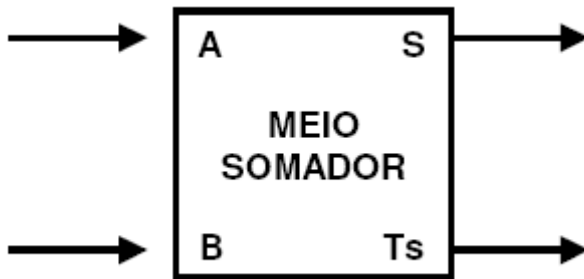
$$(1 + 1 = 0 \rightarrow Ts = 1)$$

Representando cada número por 1 bit, podemos, então, montar um circuito que possui como entradas A e B, e como saída, a soma dos algarismos (S) e o respectivo transporte de saída (T_s). As expressões características do circuito, extraídas da tabela, são: $S = A \oplus B$ $T_s = AB$

O circuito a partir destas expressões é visto na figura.



A representação em bloco deste circuito é vista na figura.

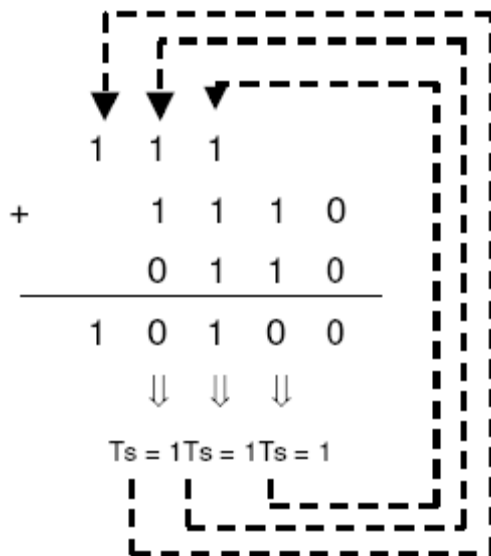


12.2 SOMADOR COMPLETO

O Meio Somador possibilita efetuar a soma de números binários com 1 algarismo.

Para se fazer a soma de números binários de mais algarismos, esse circuito torna-se insuficiente, pois não possibilita a introdução do transporte de entrada proveniente da coluna anterior. Para melhor compreensão, vamos analisar o caso da soma:

$1110_2 + 110_2$. Assim sendo, temos:



A coluna 1 tem como resultado um transporte de saída igual a 0. A coluna 2 tem como resultado 0 e um transporte de saída igual a 1. A coluna 3 tem um transporte de entrada igual a 1 (T_s da coluna anterior), possui resultado 1 e transporte de saída igual a 1. A coluna 4 tem transporte de entrada igual a 1, resultado 0 e transporte de saída 1. A coluna 5 possui apenas um transporte de entrada (T_s da coluna 4) e, obviamente, seu resultado será igual a 1.

Para fazermos a soma de 2 números binários de mais algarismos, basta somarmos coluna a coluna, levando em conta o transporte de entrada que nada mais é do que o T_s da coluna anterior.

O somador Completo é um circuito para efetuar a soma completa de uma coluna, considerando o transporte de entrada. Vamos, agora, montar a tabela da verdade deste circuito:

A	B	T _E	S	T _S
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

T_E → transporte de entrada

$$(0 + 0 + 0 = 0 \rightarrow T_s = 0)$$

$$(0 + 0 + 1 = 1 \rightarrow T_s = 0)$$

$$(0 + 1 + 0 = 1 \rightarrow T_s = 0)$$

$$(0 + 1 + 1 = 0 \rightarrow T_s = 1)$$

$$(1 + 0 + 0 = 1 \rightarrow T_s = 0)$$

$$(1 + 0 + 1 = 0 \rightarrow T_s = 1)$$

$$(1 + 1 + 0 = 0 \rightarrow T_s = 1)$$

$$(1 + 1 + 1 = 1 \rightarrow T_s = 1)$$

Vamos, então, escrever as expressões características, sem simplificação, de um Somador Completo:

$$S = \bar{A}\bar{B}T_E + \bar{A}B\bar{T}_E + A\bar{B}\bar{T}_E + ABT_E$$

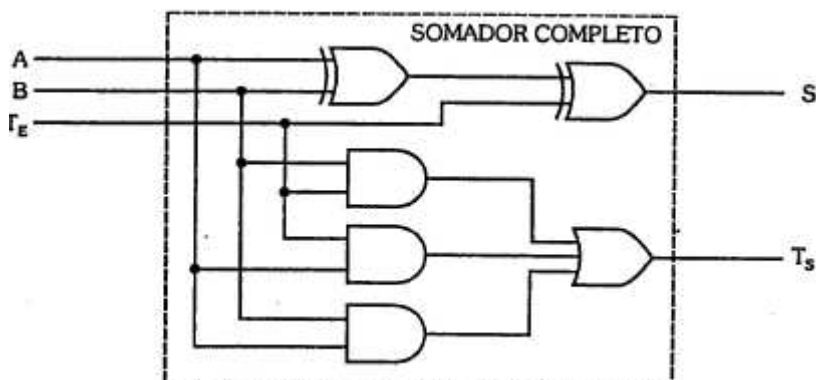
$$T_S = \bar{A}BT_E + A\bar{B}T_E + AB\bar{T}_E + ABT_E$$

Transpondo para diagramas de Veitch-Karnaugh, temos:

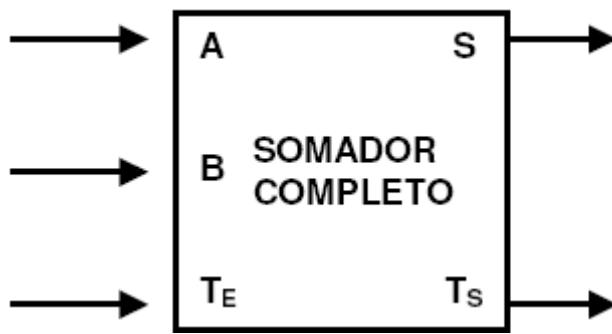
$$S = A \oplus B \oplus T_E$$

$$T_S = BT_E + AT_E + AB$$

Vamos, através das expressões, esquematizar o circuito Somador Completo:



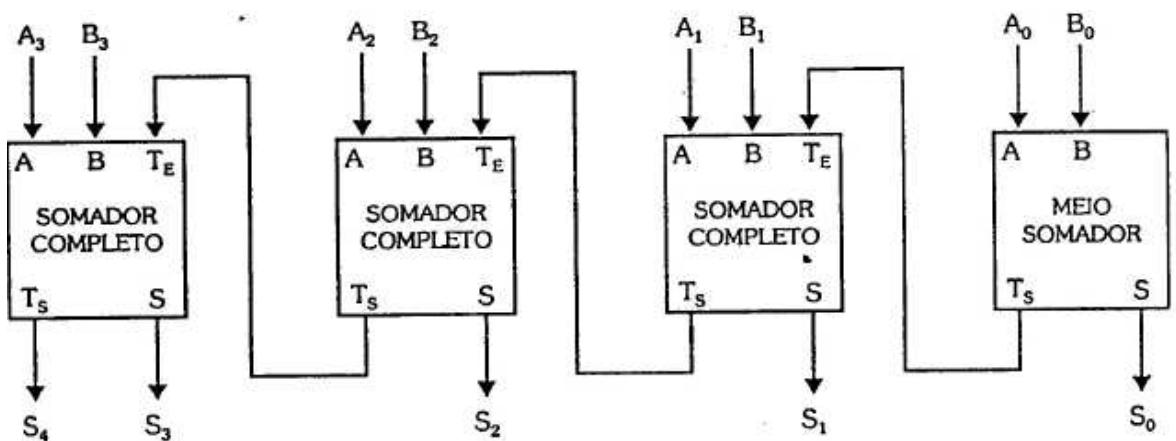
Da mesma forma, o circuito apresentado em bloco, é visto na figura.



Vamos, para exemplo de aplicação, montar um sistema em blocos que efetua soma de 2 números de 4 bits, conforme o esquema a seguir:

$$\begin{array}{r}
 A_3 \quad A_2 \quad A_1 \quad A_0 \\
 + \quad B_3 \quad B_2 \quad B_1 \quad B_0 \\
 \hline
 S_4 \quad S_3 \quad S_2 \quad S_1 \quad S_0
 \end{array}$$

Para efetuar a soma dos bits A_0 e B_0 dos números (1ª coluna), vamos utilizar um Meio Somador, pois não existe transporte de entrada, mas para as outras colunas utilizaremos Somadores Completos, pois necessitaremos considerar os transportes provenientes das colunas anteriores. O sistema montado é visto na figura.



12.3 MEIO SUBTRATOR

Antes de iniciarmos o assunto, vamos relembrar alguns tópicos importantes da subtração de números binários:

$$0 - 0 = 0$$

$$0 - 1 = 1 \quad \text{e transporta 1 ("empresta" 1)}$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

Vamos montar a tabela da verdade de uma subtração de 2 números binários de 1 algarismo:

A	B	S	T _s
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$$(0 - 0 = 0 \rightarrow T_s = 0)$$

$$(0 - 1 = 1 \rightarrow T_s = 1)$$

$$(1 - 0 = 1 \rightarrow T_s = 0)$$

$$(1 - 1 = 0 \rightarrow T_s = 0)$$

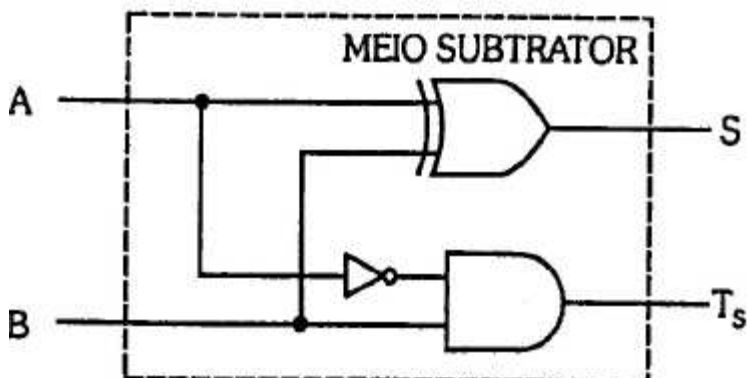
Representando cada número por 1 bit, podemos montar um circuito com as entradas A e B, e como saída, a subtração (S) e o transporte de saída (T_s).

As expressões características do circuito, extraídas da tabela, são:

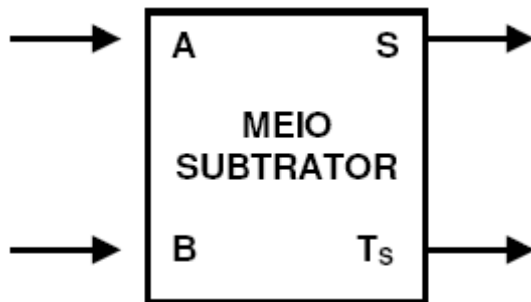
$$S = A \oplus B$$

$$T_s = \bar{A}B$$

O circuito a partir destas, é visto na figura.



Em bloco, o circuito recebe a representação da figura.



12.4 SUBTRATOR COMPLETO

O Meio Subtrator possibilita-nos efetuar a subtração de números binários de 1 algarismo. Para se fazer uma subtração com números de mais algarismos, este circuito torna-se insuficiente, pois não possibilita a entrada do transporte (T_E), proveniente da coluna anterior.

Para compreendermos melhor, vamos analisar a subtração:

$1100_2 - 11_2$. Assim sendo, temos:

$$\begin{array}{r}
 1 \quad 1 \quad 0 \quad 0 \\
 - \quad 0 \quad 0 \quad 1 \quad 1 \\
 \hline
 1 \quad 0 \quad 0 \quad 1 \\
 \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
 T_s = 0 \quad T_s = 0 \quad T_s = 1 \quad T_s = 1
 \end{array}$$

Col. 4 Col. 3 Col. 2 Col. 1

A coluna 1 tem como resultado de saída 1 e apresenta um transporte de saída igual a 1. A coluna 2 tem um transporte de entrada igual a 1 (T_s da coluna

anterior), um resultado igual a 0 e $T_s = 0$. A coluna 4 tem: $T_E = 0$, resultado igual a 1 e $T_s = 0$.

Para fazermos a subtração de números binários de mais algarismos, basta subtrairmos coluna a coluna, levando em conta o transporte de entrada, que nada mais é do que o T_s da coluna anterior.

O Subtrator Completo é um circuito que efetua a subtração completa de uma coluna, ou seja, considera o transporte de entrada proveniente da coluna anterior. Vamos, agora, montar a tabela da verdade deste circuito:

A	B	T_E	S	T_s
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

As expressões características extraídas da tabela são:

$$S = \overline{A}\overline{B}T_E + \overline{A}B\overline{T}_E + A\overline{B}\overline{T}_E + ABT_E$$

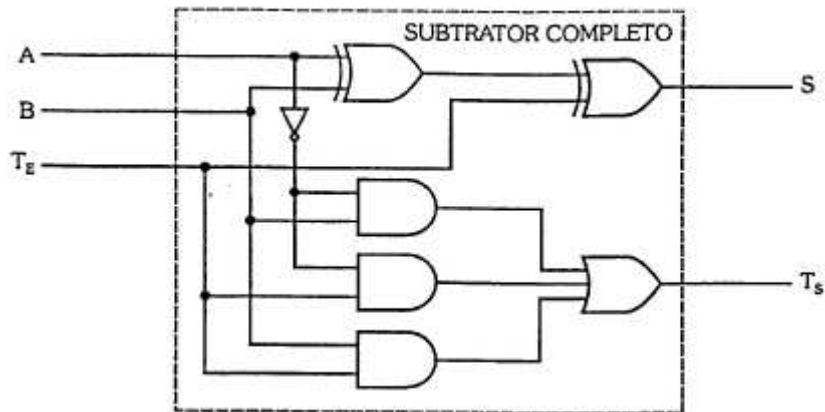
$$T_s = \overline{A}\overline{B}T_E + \overline{A}B\overline{T}_E + \overline{A}BT_E + ABT_E$$

Vamos simplificar estas expressões:

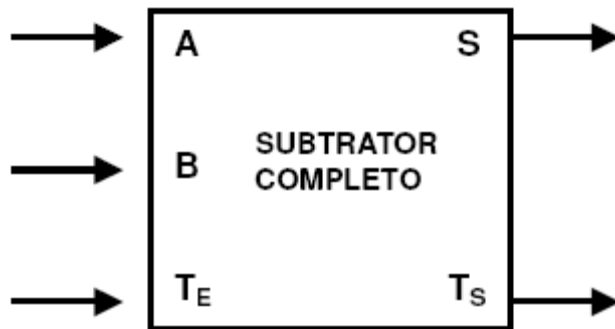
$$S = A \oplus B \oplus T_E$$

$$T_s = \overline{A}B + \overline{A}T_E + BT_E$$

O circuito derivado das expressões é visto na figura.



Em bloco, recebe a representação da figura.



12.5 SOMADOR / SUBTRATOR COMPLETO

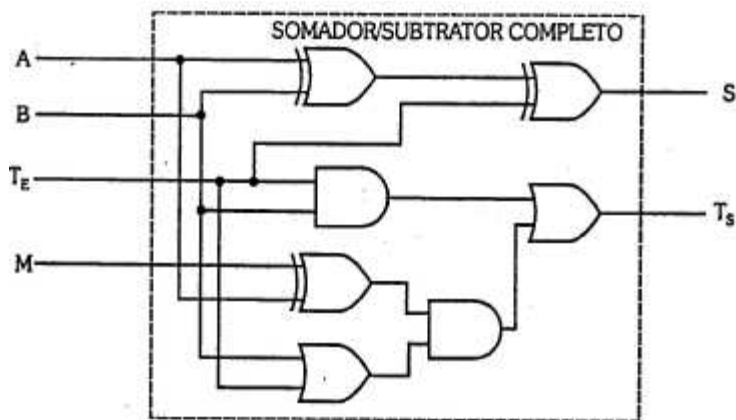
Podemos esquematizar um circuito que efetue as duas operações. Para isso, vamos introduzir outra entrada que permanecendo em nível 0, faz o circuito efetuar uma soma completa, e permanecendo em nível 1, faz efetuar uma subtração completa.

Vamos, agora, montar a tabela da verdade do circuito, sendo M a variável de controle ($M = 0 \rightarrow$ soma e $M = 1 \rightarrow$ subtração):

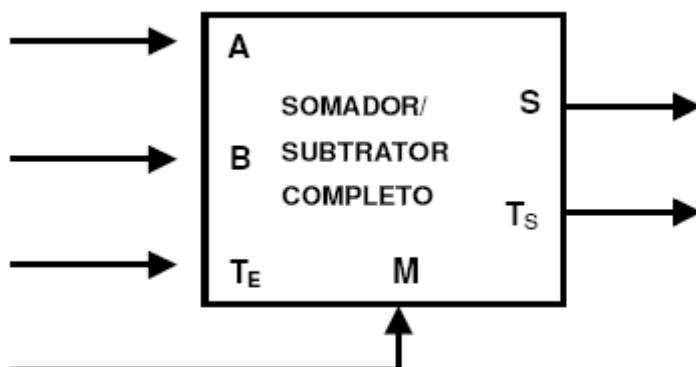
M	A	B	T _E	S	T _S
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	1	1

Soma Completa (M = 0)
 Subtração Completa (M = 1)

Vamos, então, esquematizar o circuito:



A figura mostra a representação deste circuito Somador/Subtrator Completo, em bloco:



13 FLIP-FLOP, REGISTRADORES E CONTADORES

O campo da Eletrônica Digital é basicamente dividido em duas áreas: lógica combinacional e lógica seqüencial.

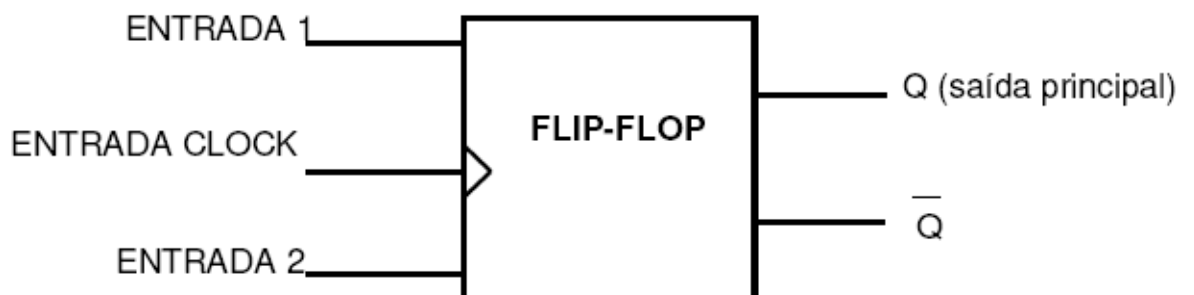
Os circuitos combinacionais, como vimos até aqui, apresentam as saídas, única e exclusivamente, dependentes das variáveis de entrada.

Os circuitos seqüenciais têm as saídas dependentes das variáveis de entrada e/ou de seus estados anteriores que permanecem armazenados, sendo, geralmente, sistemas pulsados, ou seja, operam sob o comando de uma seqüência de pulsos denominada **clock**.

Neste capítulo, trataremos do estudo dos flip-flops e de circuitos nos quais fazem o papel de elemento principal.

13.1 FLIP-FLOPS

De forma geral, podemos representar o flip-flop como um bloco onde temos 2 saídas: Q e \bar{Q} , entradas para as variáveis e uma entrada de controle (clock). A saída Q será a principal do bloco. A figura ilustra um flip-flop genérico.



Este dispositivo possui basicamente dois estados de saída. Para o flip-flop assumir um destes estados é necessário que haja uma combinação das variáveis e do pulso de controle (clock). Após este pulso, o flip-flop permanecerá neste estado até a chegada de um novo pulso de clock e, então, de acordo com as variáveis de entrada, mudará ou não de estado.

Os dois estados possíveis são:

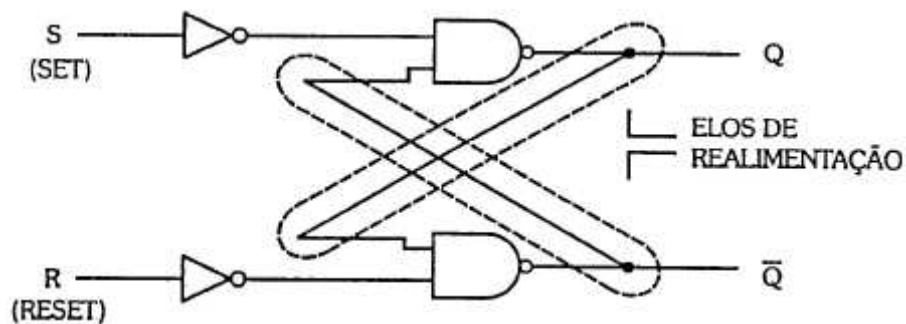
$$1) Q = 0 \rightarrow \overline{Q} = 1$$

$$2) Q = 1 \rightarrow \overline{Q} = 0$$

Vamos, a seguir, analisar alguns circuitos de flip-flops e suas respectivas operações.

13.1.1 Flip-Flop RS Básico

Primeiramente, vamos analisar o flip-flop RS básico, construído a partir de portas NE e inversores, cujo circuito é visto na figura.



Notamos que estes elos de realimentação fazem com que as saídas sejam injetadas juntamente com as variáveis de entrada, ficando claro, então, que os estados que as saídas irão assumir dependerão de ambas.

Para analisarmos o comportamento do circuito, vamos construir a tabela da verdade, levando em consideração as 2 variáveis de entrada (S e R) e a saída Q anterior (Q_a) à aplicação das entradas:

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

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

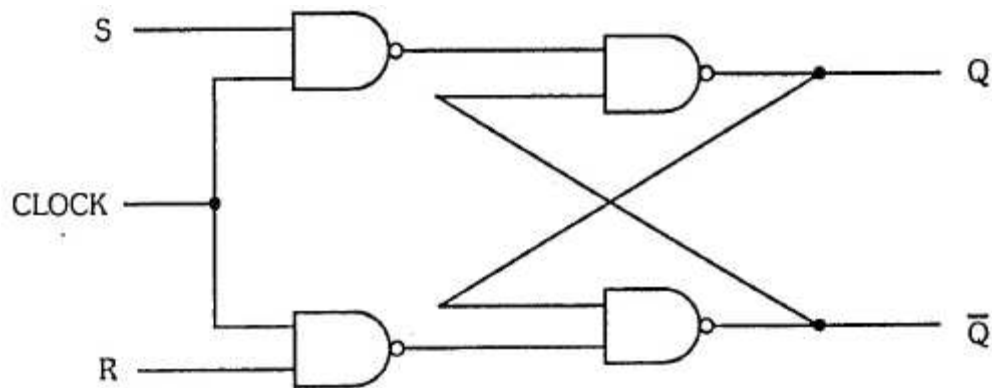
S	R	Qf
0	0	Qa
0	1	0
1	0	1
1	1	

A entrada S é denominada **Set**, pois quando acionada (nível 1), passa a saída para 1 (estabelece ou fixa 1), e a entrada R é denominada **Reset**, pois quando acionada (nível 1), passa a saída para 0 (recompõe ou zera o flip-flop). Estes termos são muito usuais na área de eletrônica digital, sendo provenientes do idioma inglês.

Este circuito irá mudar de estado apenas no instante em que mudam as variáveis de entrada. Veremos em seguida, como é o circuito de um flip-flop RS que tem sua mudança de estado controlada pela entrada de clock.

13.1.2 Flip-Flop RS com Entrada Clock

Para que o flip-flop RS básico seja controlado por uma seqüência de pulsos de clock, basta trocarmos os 2 inversores por portas NE, e as outras entradas destas portas, injetarmos o clock. O circuito, com estas modificações é visto na figura.

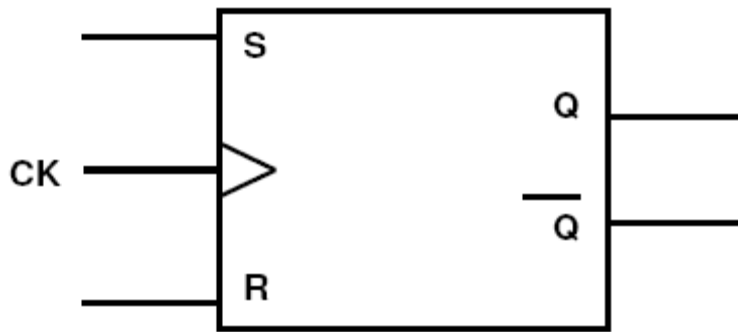


Neste circuito, quando a entrada do clock for igual a 0, o flip-flop irá permanecer no seu estado, mesmo que variem as entradas S e R. Isso pode ser confirmado pela análise do circuito, onde concluímos que para $\text{clock} = 0$, as saídas das portas NE de entrada serão sempre iguais a 1, independentemente dos valores assumidos por S e R.

Quando a entrada clock assumir valor 1, o circuito irá comportar-se como um flip-flop RS básico, pois as portas NE de entrada funcionarão como os inversores do circuito anteriormente visto. A tabela resume a operação deste flip-flop em função da entrada clock.

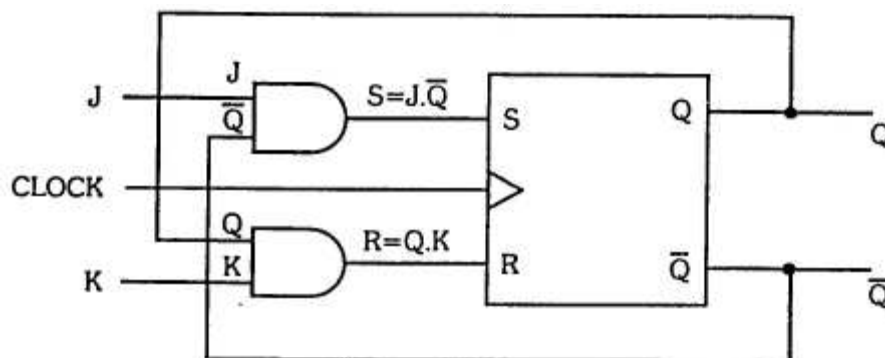
CK	Qf
0	QA
1	RS básico

De maneira geral, podemos concluir que o circuito irá funcionar quando a entrada clock assumir valor 1 e manterá travada esta saída quando a entrada clock passar para 0. O flip-flop pode ser representado pelo bloco visto na figura.



13.1.3 Flip-Flop JK

O flip-flop JK nada mais é que um flip-flop RS realimentado da maneira mostrada na figura.



A tabela simplificada resultante será:

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

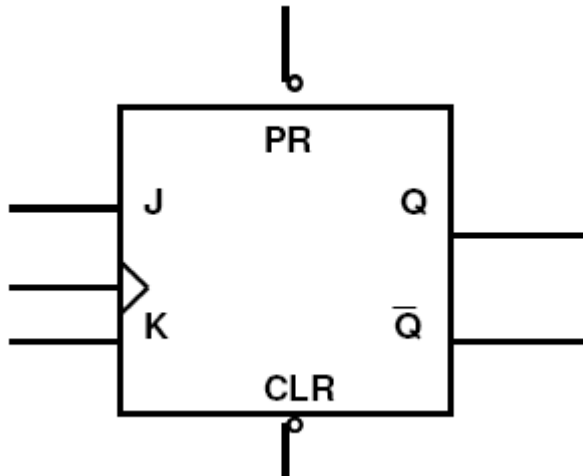
13.1.3.1 Flip-Flop JK com Entradas Preset e Clear

O flip-flop JK poderá assumir valores $Q = 1$ ou $Q = 0$ mediante a utilização das entradas **Preset (PR)** e **Clear (CLR)**.

A tabela resume a atuação das entradas Preset e Clear.

CLR	PR	Qf
0	0	Não permitido
0	1	0
1	0	1
1	1	Funcionamento normal

Podemos, para facilitar, utilizar um bloco representativo como o mostrado na figura.

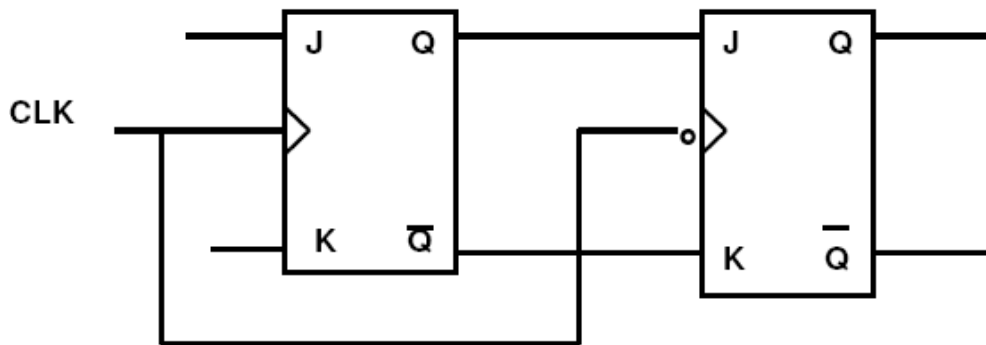


Os circuitos na simbologia do bloco indicam que as entradas Preset e Clear são ativas em 0, ou seja, funcionam respectivamente com nível 0 aplicado.

13.1.3.2 Flip-Flop JK Mestre-Escravo

O flip-flop JK apresenta uma característica indesejável. Quando o clock for igual a 1, teremos o circuito funcionando como sendo um circuito combinacional, pois haverá a passagem das entradas J, K e também da realimentação. Nessa situação, se houver uma mudança nas entradas J e K, o circuito apresentará uma nova saída, podendo alterar seu estado tantas vezes quantas alterarem os estados das entradas J e K.

Para resolver esse problema, foi criado o flip-flop **JK Mestre-Escravo (JK Master- Slave)**.



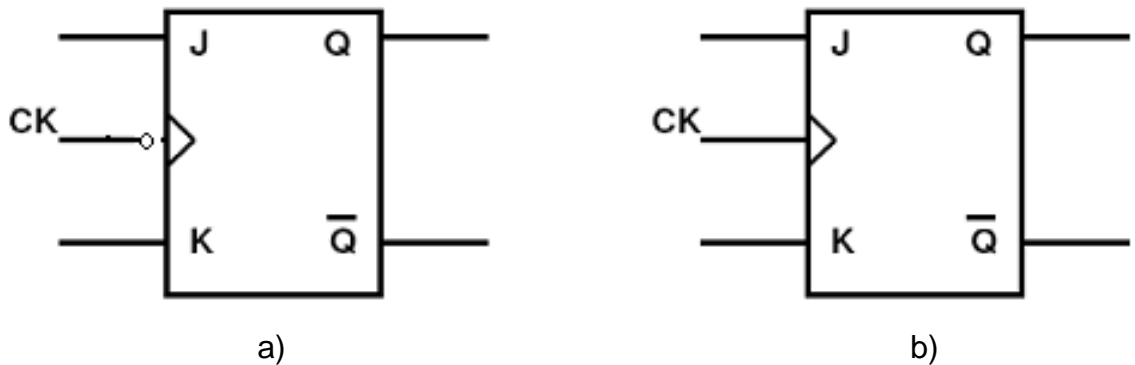
Flip-flop Mestre-Escravo.

A tabela resume a operação do flip-flop JK Mestre-Escravo:

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

Notamos que esta tabela é idêntica à de um flip-flop JK básico, porém a saída Q irá assumir valores, conforme a situação das entradas JK, somente após a passagem do clock para 0. Assim sendo, o circuito é denominado JK Mestre-Escravo **sensível à descida de clock**. Para obter um circuito **sensível à subida de clock** basta colocarmos um inversor interno à entrada clock.

A figura mostra o bloco JK Mestre-Escravo e a simbologia para identificar o circuito sensível à descida de clock (a) e à subida de clock (b).

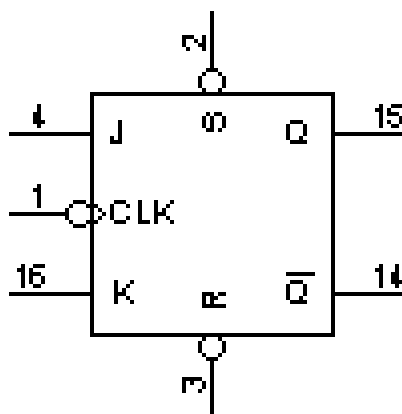


O círculo na entrada de clock, indica que o clock é ativo quando passa de 1 para 0.

13.1.3.3 Flip-Flop JK Mestre-Escravo com Entrada Preset e Clear

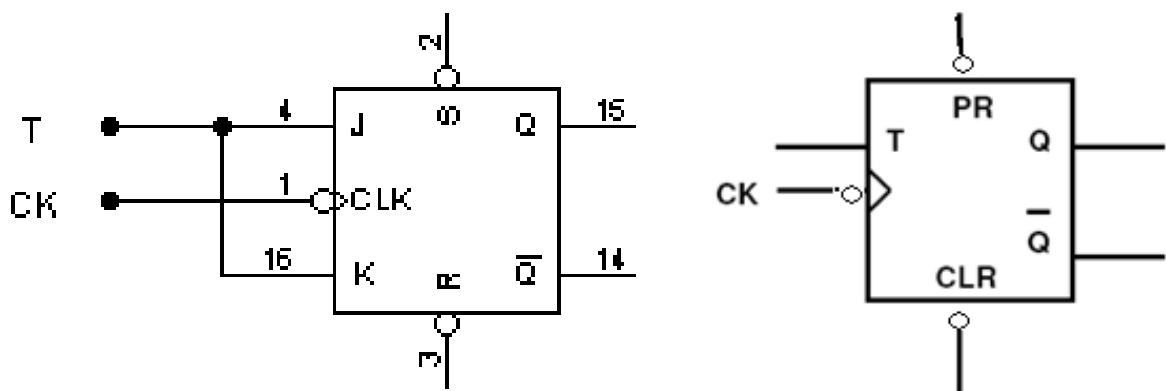
O controle de Preset, quando assumir valor 0, fará com que a saída do circuito (Q) assumira valor 1. O mesmo ocorre com o controle de Clear, fazendo com que a saída assumira valor 0.

A figura mostra o bloco representativo do flip-flop JK Mestre-Escravo com as entradas Preset e Clear ativas em 0.



13.1.4 Flip-Flop Tipo T

Este flip-flop é obtido de um JK Mestre-escravo com as entradas J e K curto-circuitadas (uma ligada à outra), logo quando J assumir valor 1, K também assumirá valor 1, e quando J assumir valor 0, K também assumirá valor 0. Obviamente, no caso desta ligação, não irão ocorrer nunca entradas como: $J = 0$ e $K = 1$; $J = 1$ e $K = 0$. A figura mostra a ligação e o bloco representativo do flip-flop tipo T obtido.



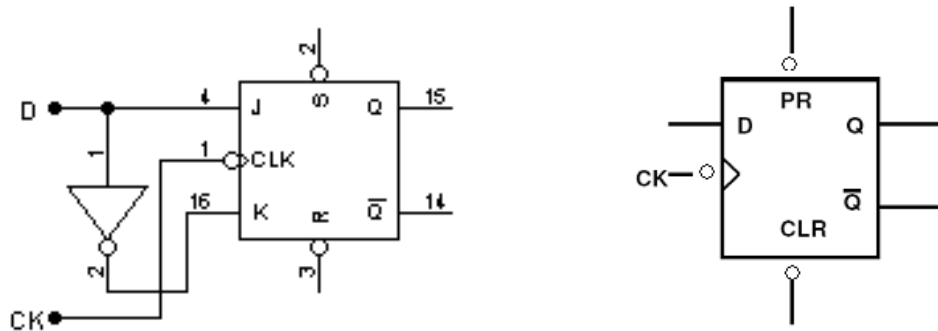
Eliminando os casos não existentes, obtemos a tabela da verdade do flip-flop do tipo T:

T	Qf
0	Qa
1	$\bar{Q}a$

Devido ao fato de o flip-flop tipo T, com a entrada T igual a 1, complementar a saída ($\bar{Q}a$) a cada descida de clock, este será utilizado como célula principal dos **contadores assíncronos** que serão estudados adiante. A sigla **T** vem de **Toggle** (comutado).

13.1.5 Flip-Flop Tipo D

É obtido a partir de um flip-flop JK Mestre-Escravo com a entrada K invertida (por inversor) em relação a J. Logo, neste flip-flop, teremos as seguintes entradas possíveis: $J = 0$ e $K = 1$; $J = 1$ e $K = 0$. Obviamente, não irão ocorrer os casos: $J = 0$ e $K = 0$; $J = 1$ e $K = 1$. A figura mostra como este é obtido e seu bloco representativo.



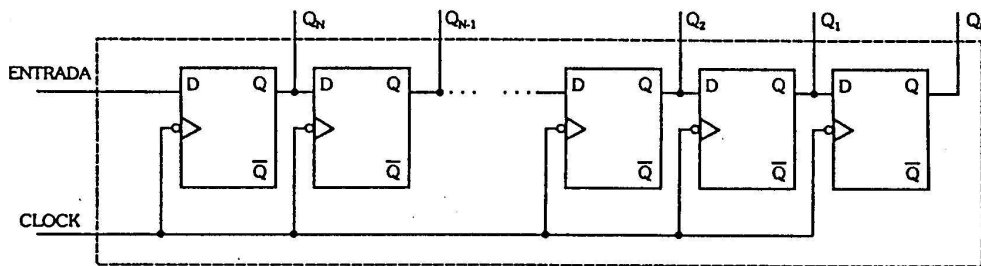
Eliminando os casos não existentes, obtemos a tabela do flip-flop tipo D.

D	Qf
0	0
1	1

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

13.2 REGISTRADORES DE DESLOCAMENTO

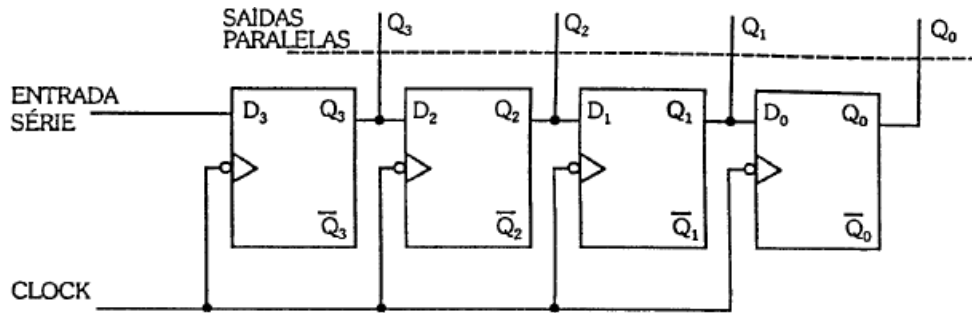
Como vimos, o flip-flop pode armazenar durante o período em que sua entrada clock for igual a 0, um bit apenas (saída Q). Porém, se necessitarmos guardar uma informação de mais de um bit, o flip-flop irá tornar-se insuficiente. Para isso utilizamo-nos de um sistema denominado **Registrador de Deslocamento (Shift Register)**. Trata-se de um, certo número de flip-flops tipo JK Mestre-Escravo ligado de tal forma que as saídas de cada bloco sejam aplicadas nas entradas J e K respectivas do flip-flop tipo D. A figura representa um registrador de Deslocamento.



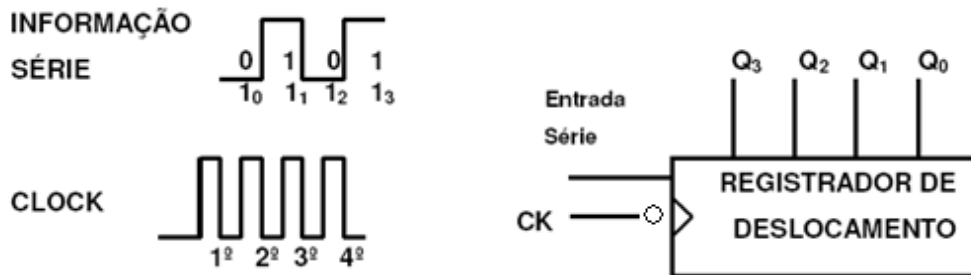
O funcionamento deste sistema, juntamente com suas aplicações, será visto nos itens subseqüente.

13.2.1 Conversor Série-Paralelo

O Registrador de Deslocamento pode ser usado para converter uma informação série em paralela, ou seja, funcionar como Conversor Série-Paralelo. A configuração básica nessa situação, para uma informação de 4 bits, é vista na figura.



Como exemplo, vamos aplicar a informação série $I = 1010$ ($I_3 I_2 I_1 I_0$) à entrada série do registrador e analisar as saídas Q_0 , Q_1 , Q_2 e Q_3 , após os pulsos de clock. Deve-se ressaltar que estes flip-flops atuam como mestre-escravo e têm sua comutação no instante da descida do pulso de clock. Assim sendo, temos:



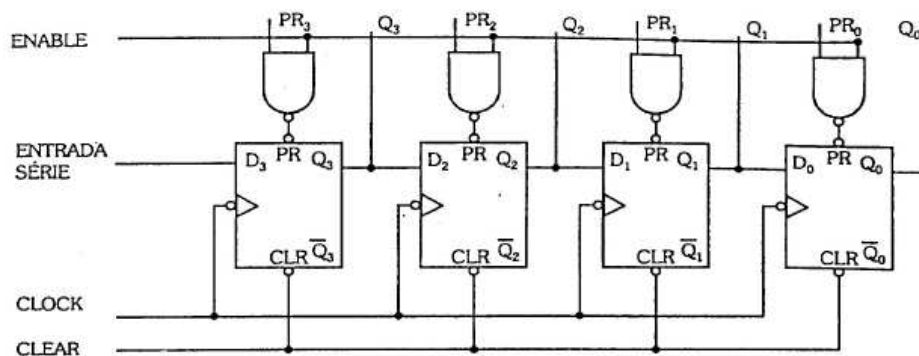
Para resumir, vamos representar toda a seqüência sob a forma da tabela verdade:

Informação	Descidas de clock	Q_3	Q_2	Q_1	Q_0
$I_0 = 0$	1ª	0	0	0	0
$I_1 = 1$	2ª	1	0	0	0
$I_2 = 0$	3ª	0	1	0	0
$I_3 = 1$	4ª	1	0	1	0

É pelo motivo de deslocar a informação a cada pulso de clock que esse dispositivo é denominado Registrador de Deslocamento.

13.2.2. Conversor Paralelo-Série

Para entrarmos com uma informação paralela, necessitamos de um registrador que apresente entradas Preset e Clear, pois é através destas que fazemos com que o Registrador armazene a informação paralela. O registrador com estas entradas é visto na figura.



Primeiramente, vamos estudar o funcionamento da entrada **ENABLE**.

Quando a entrada enable estiver em 0, as entradas preset (PR) dos flip-flops assumirão, respectivamente, níveis 1, fazendo com que o registrador atue normalmente.

Quando a entrada enable for igual a 1, as entradas preset dos flip-flops assumirão os valores complementares das entradas PR₃, PR₂, PR₁ e PR₀, logo, os flip-flops irão assumir os valores que estiverem, respectivamente, em PR₃, PR₂, PR₁ e PR₀.

Para entendermos melhor, vamos analisar uma célula do registrador. Para zerar (clear) o flip-flop ($Q_3 = 0$), vamos inicialmente, aplicar nível 0 à entrada clear. Com enable = 0, a entrada PR do flip-flop irá assumir nível 1 e este irá ter um funcionamento normal como célula do registrador de deslocamento em questão, mantendo a saída no estado em que se encontra.

Com enable = 1 e PR₃ = 0, a entrada PR do flip-flop assumirá nível 1, logo, a saída Q₃ manterá o seu estado ($Q_3 = 0$). Com enable = 1 e PR₃ = 1, a entrada PR do flip-flop assumirá nível 0, forçando a saída a assumir nível 1 ($Q_3 = 1$).

Após essa análise, concluímos que, se zerarmos o registrador (aplicando 0 à entrada clear), e logo após introduzirmos a informação paralela (I₃, I₂, I₁ e I₀) pelas

entradas PR3, PR2, PR1 e PR0, as saídas Q3, Q2, Q1 e Q0 assumirão respectivamente os valores da informação.

Essa maneira de entrarmos com a informação no registrador é chamada entrada paralela de informação, sendo a entrada responsável pela habilitação da mesma.

Para que o registrador de deslocamento funcione como **Conversor Paralelo-Série**, necessitamos zerá-lo e em seguida, introduzir a informação como já descrito, recolhendo na saída Q0 a mesma informação de modo série.

É fácil de notar que a saída Q0 assume primeiramente o valor I_0 e a cada descida do pulso de clock, irá assumir seqüencialmente os valores I_1 , I_2 e I_3 .

13.3 CONTADORES

Contadores são circuitos digitais que variam os seus estados, sob o comando de um clock, de acordo com uma seqüência predeterminada. São utilizadas principalmente para contagens diversas, divisões de freqüência, medição de freqüência e tempo, geração de formas de onda e conversão de analógico para digital.

Basicamente, estes sistemas, são divididos em duas categorias: **Contadores Assíncronos e Síncronos**.

13.3.1 Contadores Assíncronos

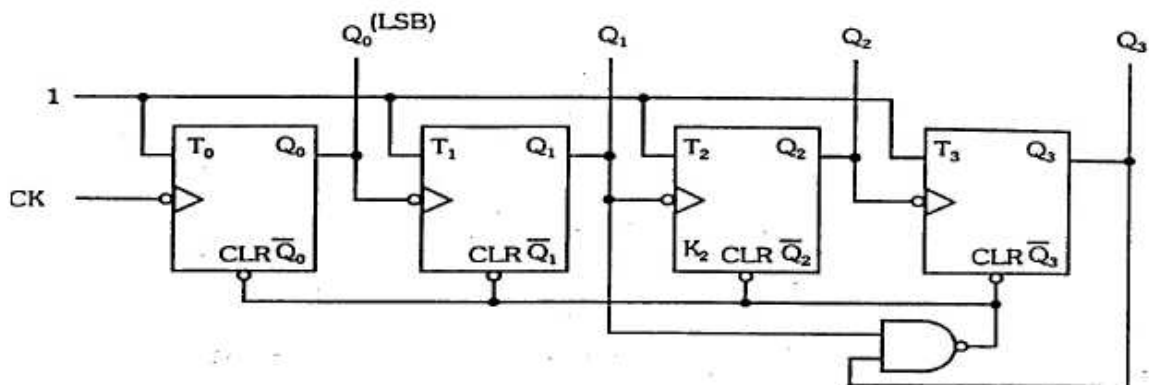
São caracterizados por seus flip-flops funcionarem de maneira assíncrona (sem sincronismo), não tendo entradas clock em comum. Neste tipo de circuito, a entrada clock se faz apenas no primeiro flip-flop, sendo as outras derivadas das saídas dos blocos anteriores.

Vamos, a seguir, analisar os principais contadores assíncronos:

13.3.1.2 Contador de Década

O contador de década é o circuito que efetua a contagem em números binários de 0 a 9_{10} (10 algarismos). Isso significa acompanhar a seqüência do código BCD 8421 de 0000 até 1001.

Para que o contador conte somente de 0 a 9, deve-se jogar um nível 0 na entrada clear assim que surgir o caso 10 (1010), ou seja, no 10º pulso. O circuito de um contador de década assíncrono é visto na figura.



Temos, neste caso, a seguinte tabela da verdade:

Descidas de clock	Q_3	Q_2	Q_1	Q_0	CLR
1ª	0	0	0	0	1
2ª	0	0	0	1	1
3ª	0	0	1	0	1
4ª	0	0	1	1	1
5ª	0	1	0	0	1
6ª	0	1	0	1	1
7ª	0	1	1	0	1
8ª	0	1	1	1	1
9ª	1	0	0	0	1
10ª	1	0	0	1	1
	1	0	1	0	0

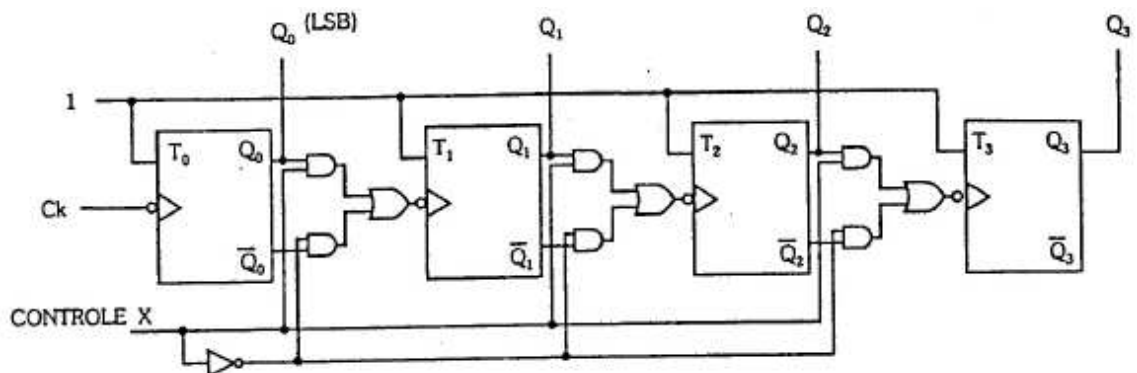
Este contador poderá ser utilizado como divisor de frequência por 10 para uma onda quadrada aplicada à entrada clock, pois possui 10 estados de saída.

13.3.1.3 Contador Assíncrono Crescente/Decrescente

Podemos construir um contador que execute a contagem crescente ou decrescente.

Para isso, utilizamos uma variável de controle que quando assume 1, faz o circuito executar contagem crescente e quando assume 0, faz a contagem decrescente.

Este circuito é mostrado na figura.



Notamos que, no circuito, quando o controle X estiver em 1, às saídas \bar{Q}_0 , \bar{Q}_1 e \bar{Q}_2 estarão bloqueadas, fazendo com que entrem as saídas Q_0 , Q_1 e Q_2 nas entradas clock dos flip-flops respectivamente. Isto fará com que o contador conte crescentemente.

Quando o controle X estiver em 0, a situação invertir-se-á e, por conseguinte, o contador contará decrescentemente.

Notamos, ainda, que Q_0 será a saída do bit menos significativo (LSB).

O contador crescente/decrescente é também denominado **Up/Down Counter**, que é o termo designativo em inglês.

13.3.2 Contadores Síncronos

Estes contadores possuem entradas clock curto-circuitadas, ou seja, o clock entra em todos os flip-flops simultaneamente, fazendo todos atuarem de forma sincronizada.

Para que haja mudanças de estado, devemos então estudar o comportamento das entradas J e K dos vários flip-flops, para que tenhamos nas saídas, as seqüências desejadas.

Para estudarmos os contadores síncronos devemos sempre escrever a tabela da verdade, estudando quais devem ser as entradas J e K dos vários flip-flops, para que estes assumam o estado seguinte. Para isso, vamos utilizar a tabela da verdade do flip-flop JK.

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

A partir desta tabela, construímos outra relacionando os estados de saída e as entradas J e K:

	Qa	Qf	J	K
1)	0	0	0	X
2)	0	1	1	X
3)	1	0	X	1
4)	1	1	X	0

Vamos, a seguir analisar cada caso:

1) Se o flip-flop estiver em 0 ($Qa = 0$) e quisermos que o estado a ser assumido seja 0 ($Qf = 0$), podemos tanto manter o estado do flip-flop ($J = 0, K = 0$)

→ $Q_f = Q_a$), como fixar 0 ($J = 0, K = 1 \rightarrow Q_f = 0$), logo, se $J = 0$ e $K = X$, teremos a passagem de $Q_a = 0$ para $Q_f = 0$.

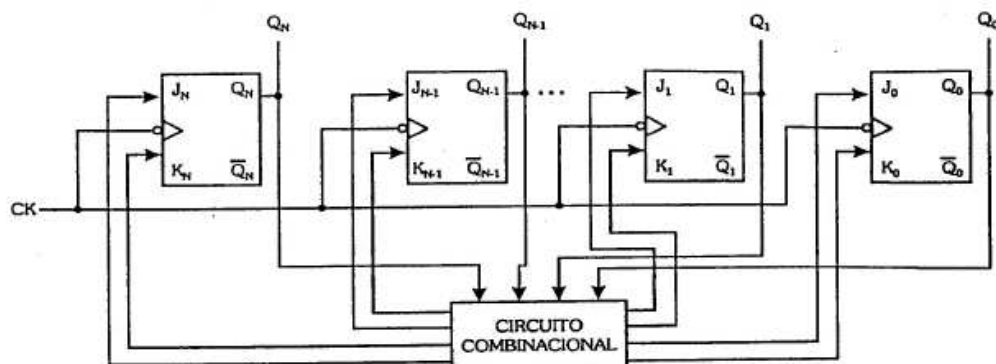
2) Se o flip-flop estiver em 0 ($Q_a = 0$) e quisermos que o estado a ser assumido seja 1 ($Q_f = 1$), podemos tanto inverter o estado ($J = 1, K = 1 \rightarrow Q_f = \bar{Q}_a$), como fixarmos 1 ($J = 1, K = 0 \rightarrow Q_f = 1$), logo, se $J = 1$ e $K = X$, teremos a passagem de $Q_a = 0$ para $Q_f = 1$.

3) Quando o flip-flop estiver em 1 ($Q_a = 1$) e quisermos que ele vá para 0 ($Q_f = 0$), podemos inverter o estado ($J = 1, K = 1 \rightarrow Q_f = \bar{Q}_a$) ou fixar 0 ($J = 0, K = 1 \rightarrow Q_f = 0$), logo, se $J = X$ e $K = 1$, teremos a passagem de $Q_a = 1$ para $Q_f = 0$.

4) Quando o flip-flop estiver em 1 ($Q_a = 1$) e quisermos que ele permaneça em 1 ($Q_f = 1$), podemos manter o estado ($J = 0, K = 0 \rightarrow Q_f = Q_a$) ou fixarmos 1 ($J = 1, K = 0 \rightarrow Q_f = 1$), logo, se $J = X$ e $K = 0$, teremos a passagem de $Q_a = 1$ para $Q_f = 1$.

De posse dos resultados das entradas J e K dos flip-flops para a seqüência desejada, obtidos da tabela, efetuamos as simplificações e montamos um circuito combinacional que em função das saídas dos flip-flops irá atuar nestas entradas para processar as mudanças de estado.

Genericamente, um contador síncrono possui o esquema visto na figura.

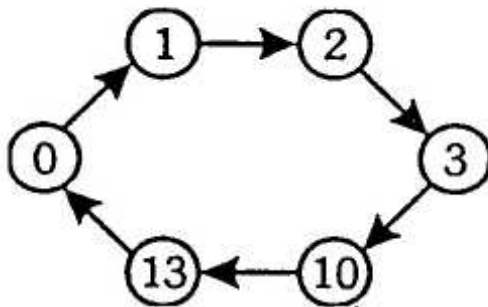


13.3.2.1 Contador Gerador de uma Seqüência Qualquer

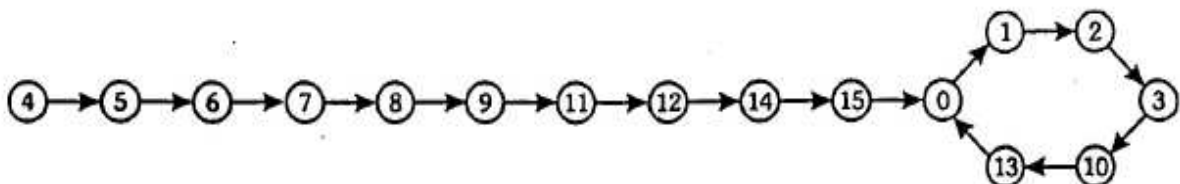
Podemos construir um contador que gere uma seqüência qualquer. Para isso, basta estabelecermos a seqüência e seguirmos o método já conhecido, ou seja, o da determinação das entradas J e K. os estados que não fizerem parte da seqüência deverão ser considerados como condições irrelevantes, ou ser encadeados objetivando atingir o estado inicial.

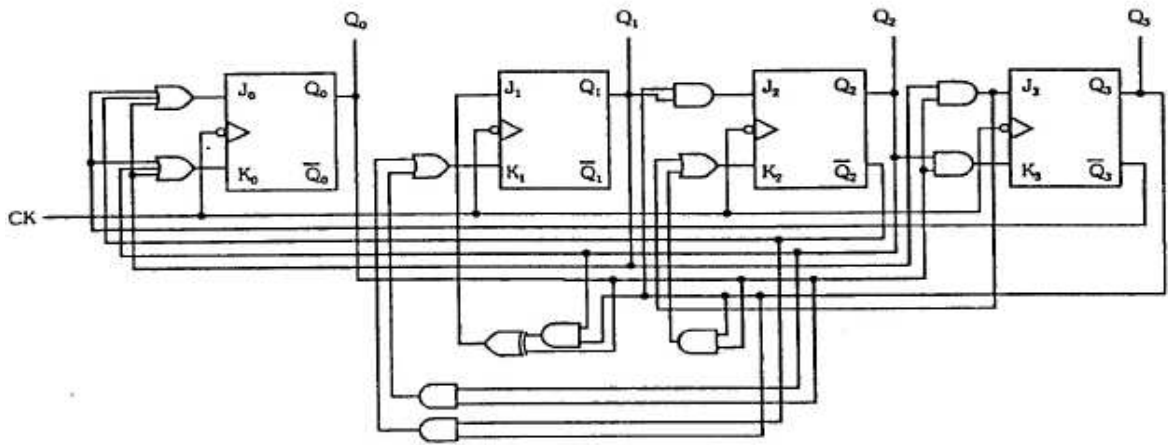
Para exemplificarmos, vamos construir um contador que gere a seguinte seqüência: $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 10 \rightarrow 13 \rightarrow 0$.

O **loop** que o contador deve efetuar para acompanhar a seqüência é visto no **diagrama de estados** visto na figura.



Notamos que os estados que não pertencem à seqüência são: 4, 5, 6, 7, 8, 9, 11, 12, 14 e 15. Vamos fazer, então, com que o contador, estando no estado 4, após o pulso de clock, vá para o estado 5, deste para o 6 e assim sucessivamente, até que o estado 15 vá para 0 que inicia a seqüência. Esquemáticamente, temos:





14 CIRCUITOS MULTIPLEX E DEMULTIPLEX

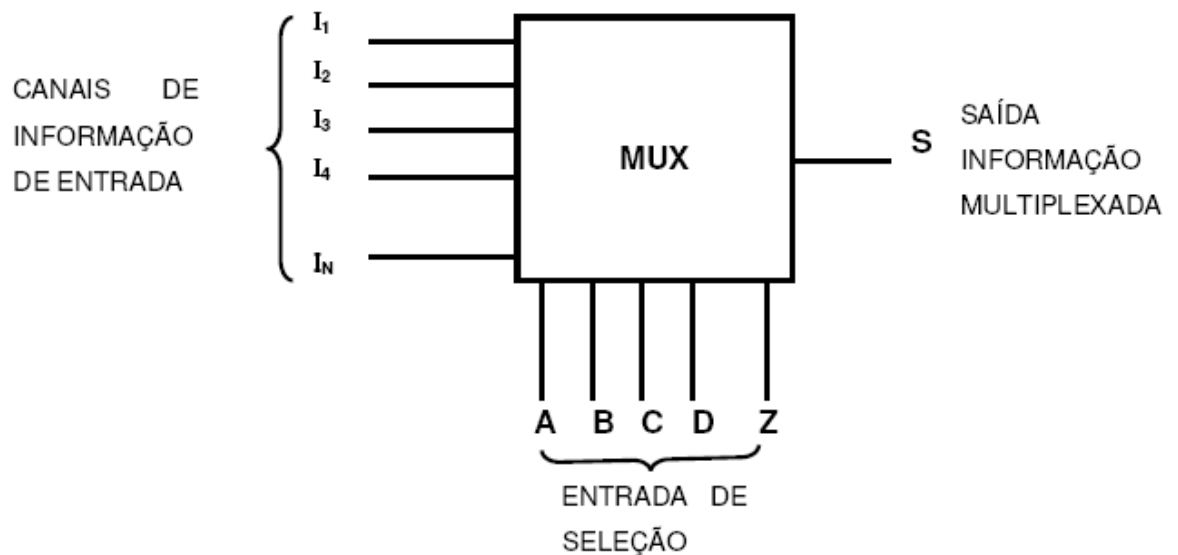
Os circuitos multiplex são utilizados nos casos em que necessitamos enviar em certo número de informações, contidas em vários canais, a um só canal.

Os circuitos demultiplex efetuam a função inversa à dos multiplex, ou seja, enviam as informações, vindas de um único canal, a vários canais.

Ambos os circuitos são largamente empregados dentro de sistemas digitais, bem como na área de Transmissão de dados.

14.1 MULTIPLEX

Como dissemos no início deste capítulo, o circuito multiplex é utilizado para enviarmos as informações contidas em vários canais (fios), a um só canal (fio). Esquematisando o bloco multiplex, temos:



14.1.1 Projeto do Circuito de um Multiplex

Para projetarmos um multiplex, devemos relacionar, principalmente, a possibilidade de que as entradas de seleção irão assumir com a informação de entrada que deve ser conectada à saída. Para isso, montamos uma tabela da verdade onde serão colocadas todas as possibilidades de seleção e as respectivas informações que devem aparecer na saída.

Para mostrarmos passo a passo a elaboração de multiplex, vamos iniciar, efetuando o projeto de um multiplex de 4 canais ou entradas de informações.

Para que possamos conectar aleatoriamente 4 entradas à saída, necessitamos de 2 variáveis de seleção. Com isso, podemos montar a tabela da verdade:

Variáveis de seleção		Saída
A	B	S
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Montando a tabela, relacionamos os valores assumidos pela saída para cada possibilidade das variáveis de seleção, obtendo, a partir disso, o respectivo produto canônico.

Variáveis de Seleção:

Caso 00 ($P_0 = \bar{A} \cdot \bar{B}$)

Caso 01 ($P_1 = \bar{A} \cdot B$)

Caso 10 ($P_2 = A \cdot \bar{B}$)

Caso 11 ($P_3 = A \cdot B$)

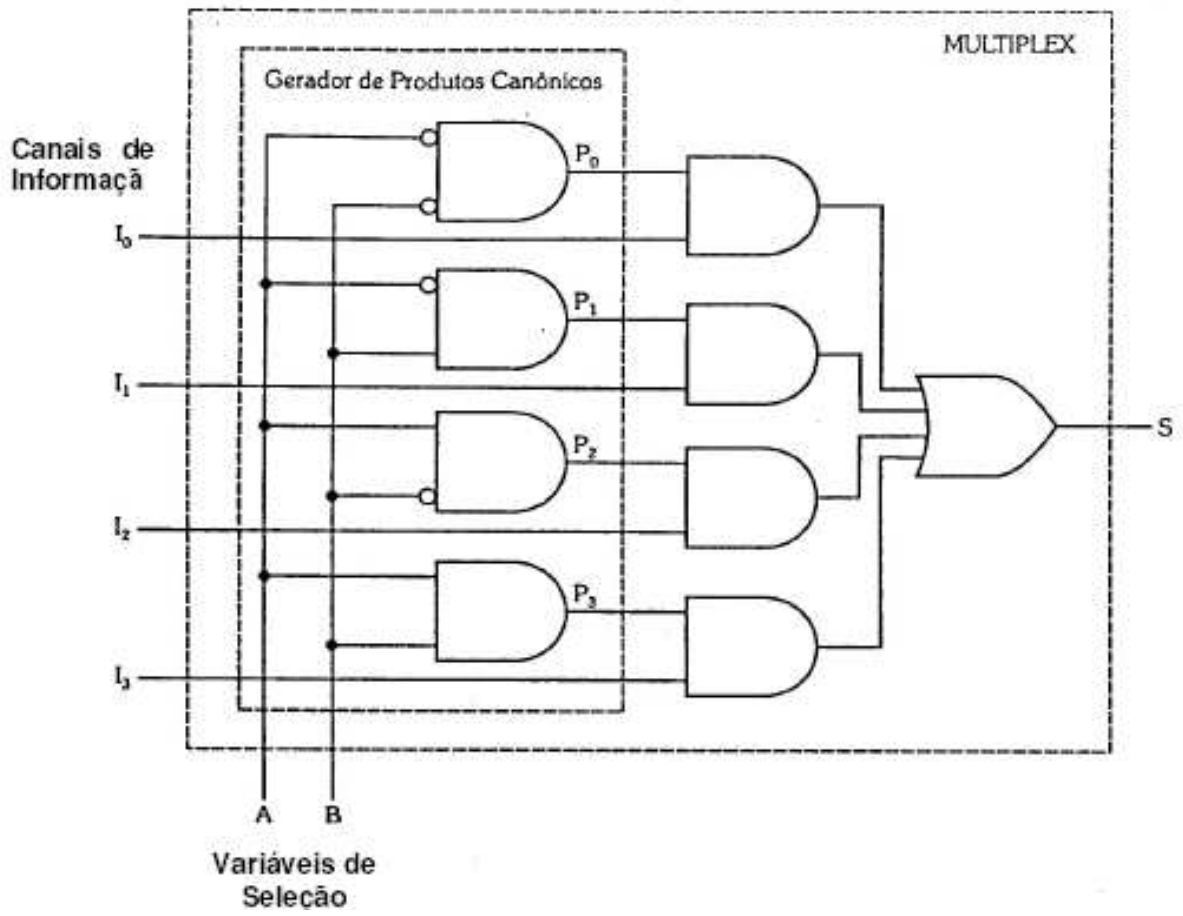
Situação na Saída:

$S = I_0$

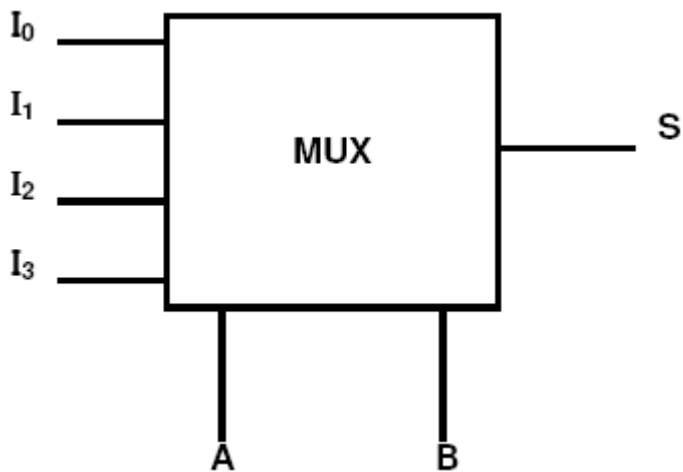
$S = I_1$

$S = I_2$

$S = I_3$

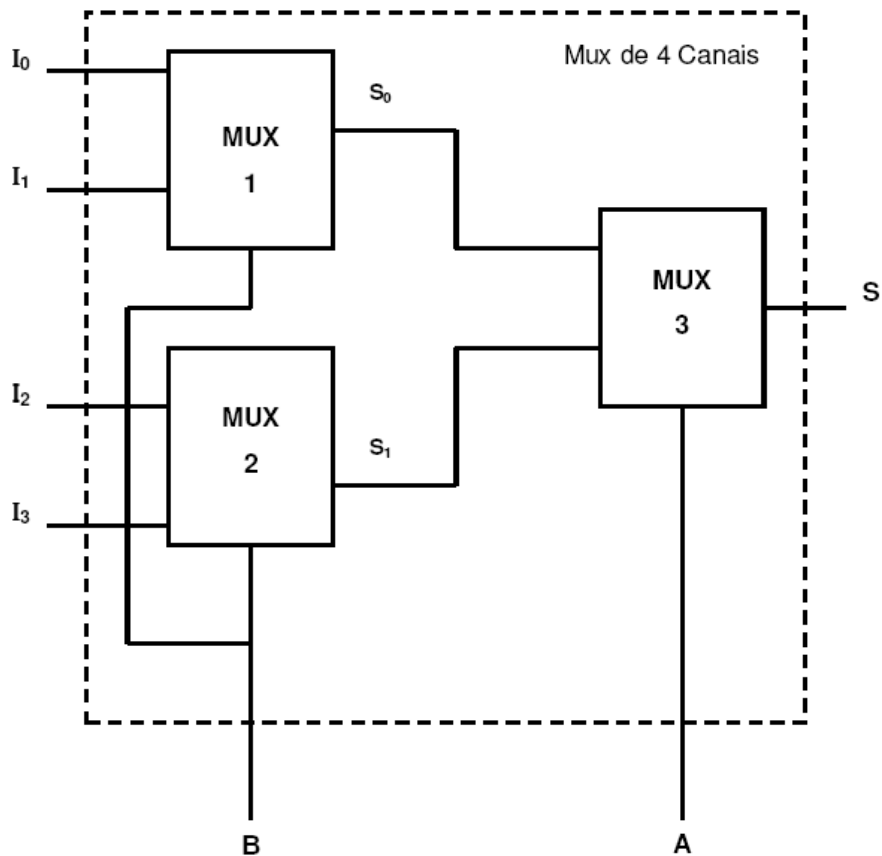


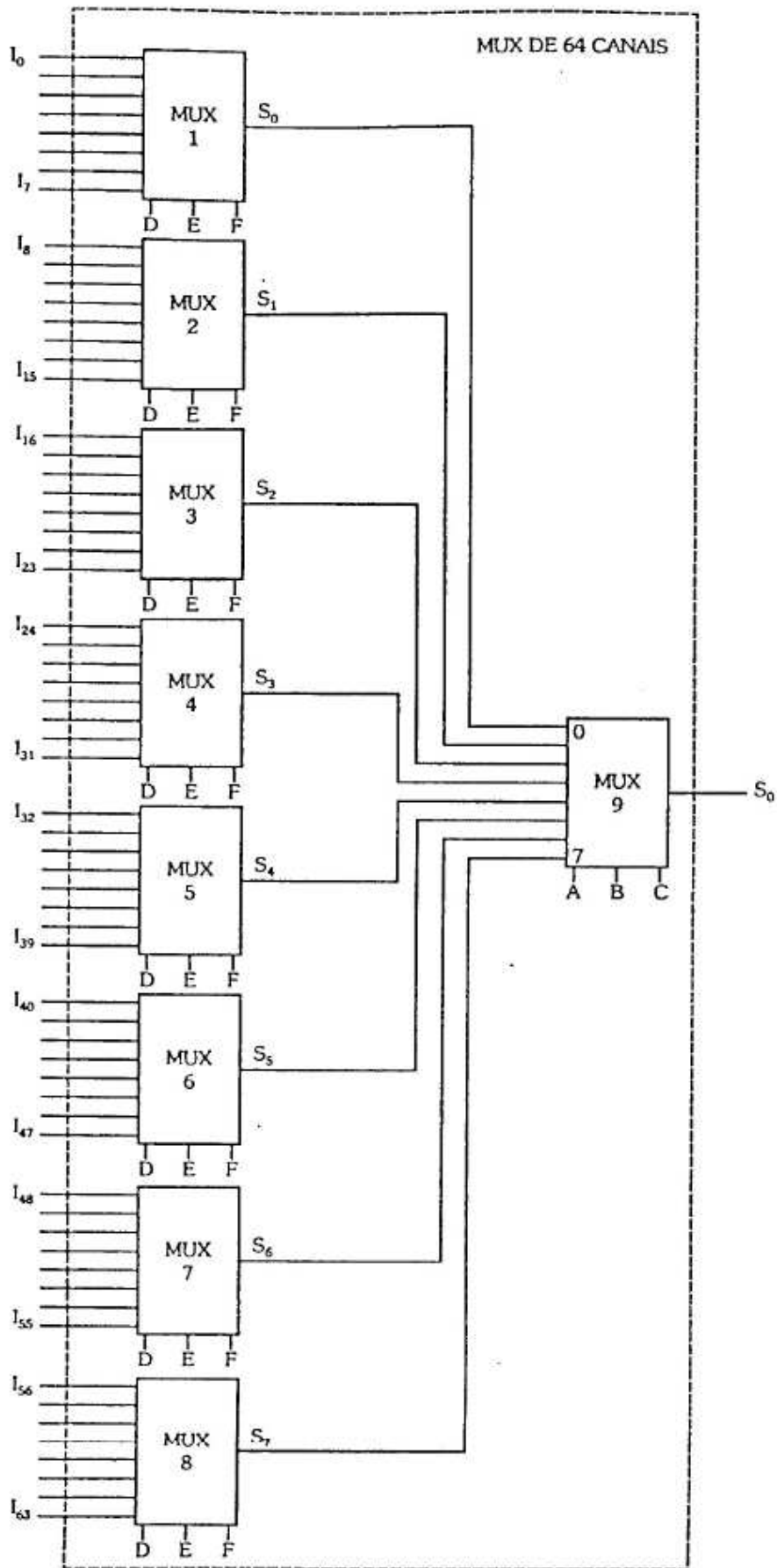
Representando o multiplex obtido em bloco, temos:



14.1.2 Ampliação da capacidade de um Sistema Multiplex

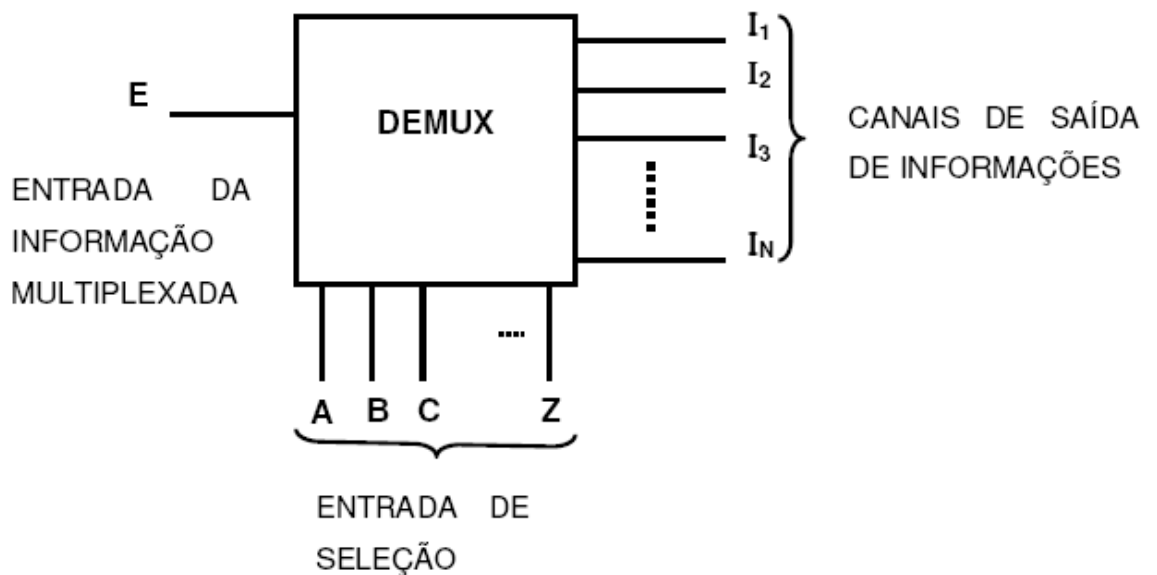
Podemos, a partir de circuitos multiplex de baixa capacidade, formar outros para um maior número de informações de entrada. Para entendermos o processo, vamos montar um multiplex de 4 canais de informação, a partir de outros de apenas 2 canais de informação. A figura mostra, em blocos, o multiplex obtido.





14.2 DEMULTIPLEX

Entende-se por demultiplex como sendo o bloco que efetua a função inversa ao multiplex, ou seja, a de enviar informações contidas em um canal a vários canais de saída. A figura mostra um bloco demultiplex genérico.



As entradas de seleção têm como finalidade escolher qual o canal de informação de saída que deve ser conectado à entrada, ou seja, devem endereçar o canal de saída, ao qual a informação deve se dirigir.

14.2.1 Projeto do Circuito de um Demultiplex

Para projetarmos um demultiplex devemos relacionar, primeiramente, a possibilidade que as variáveis de seleção irão assumir (endereço), com o canal de saída de informação que deve ser conectado à entrada. Para isso, montamos uma tabela da verdade onde são considerados todas as possibilidades de seleção e os respectivos canais de informação.

Como exemplo, vamos elaborar um demultiplex de 4 canais. Para que possamos conectar aleatoriamente uma entrada a 4 canais de saída, necessitamos,

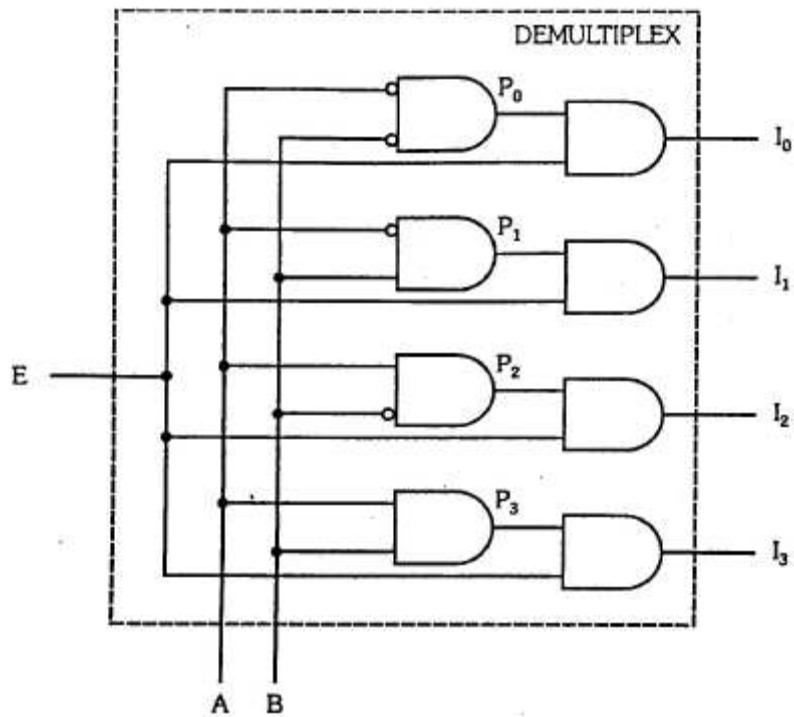
como já visto, de 2 variáveis de seleção. Com isso, podemos montar a tabela verdade:

Variáveis		Canais de Saída			
A	B	I ₀	I ₁	I ₂	I ₃
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

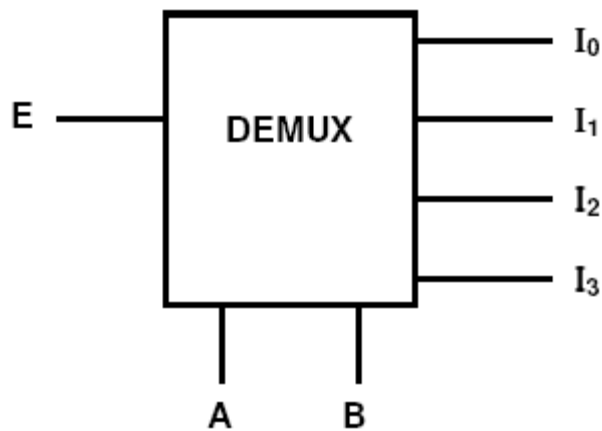
Através de uma tabela, notamos que, quando as variáveis de seleção assumirem:

- ⇒ 00 ($P_0 = \bar{A} \cdot \bar{B}$) : teremos o valor de E no canal de saída I₀.
- ⇒ 01 ($P_1 = \bar{A} \cdot B$) : teremos o valor de E no canal de saída I₁.
- ⇒ 10 ($P_2 = A \cdot \bar{B}$) : teremos o valor de E no canal de saída I₂.
- ⇒ 11 ($P_3 = A \cdot B$) : teremos o valor de E no canal de saída I₃.

O circuito para executar esta função é visto na figura.



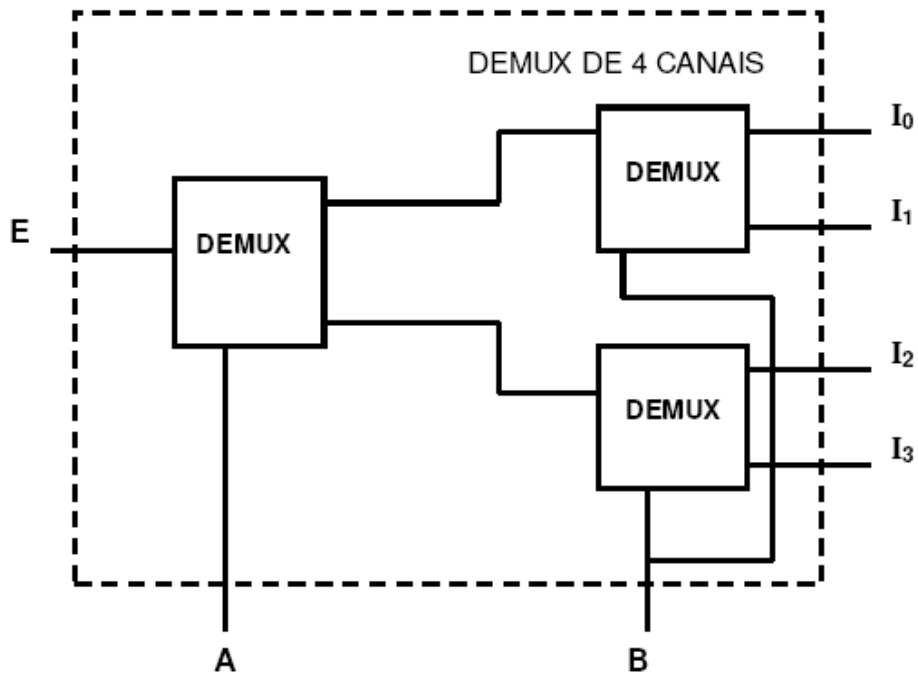
Em bloco, o circuito fica representado:



14.2.2 Ampliação da Capacidade de um Circuito Demultiplex

Como nos circuitos multiplex, podemos montar a partir de demultiplexadores de menor capacidade, outros de maior capacidade, ou seja, maior número de canais de saída.

Para entendermos o processo, vamos iniciar com um caso simples, onde vamos montar um demultiplex de 4 canais a partir de outros de apenas 2 canais de saída. A figura apresenta esta montagem.



14.3 MULTIPLEX E DEMULTIPLEX UTILIZADOS NA TRANSMISSÃO DE DADOS

Os circuitos Multiplex e Demultiplex são muito utilizados em transmissão de dados.

Para isso, basta que tenhamos um bloco no transmissor e um outro no receptor executando a função inversa.

Para que haja uma perfeita recepção, é necessário também que as variáveis de seleção estejam sincronizadas, ou seja, tanto na transmissão como na recepção, as variáveis de controle devem enviar o mesmo endereço. Basicamente, temos dois processos de transmissão:

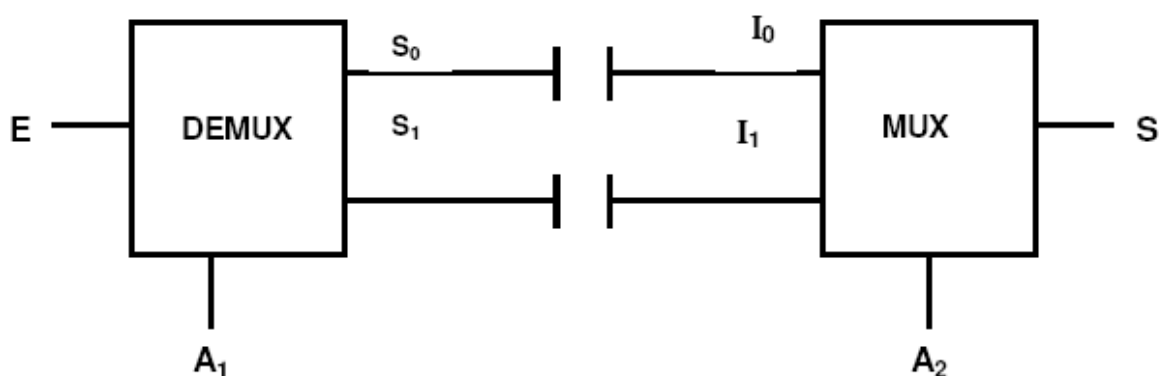
- 1 – **Transmissão paralela:** através de múltiplos fios.
- 2 – **Transmissão série:** através de 1 fio.

Vamos, para analisar os processos, exemplificar a transmissão de dados de 2 bits nos dois modos:

14.3.1 Transmissão Paralela

A configuração do circuito neste tipo de transmissão é vista na figura.

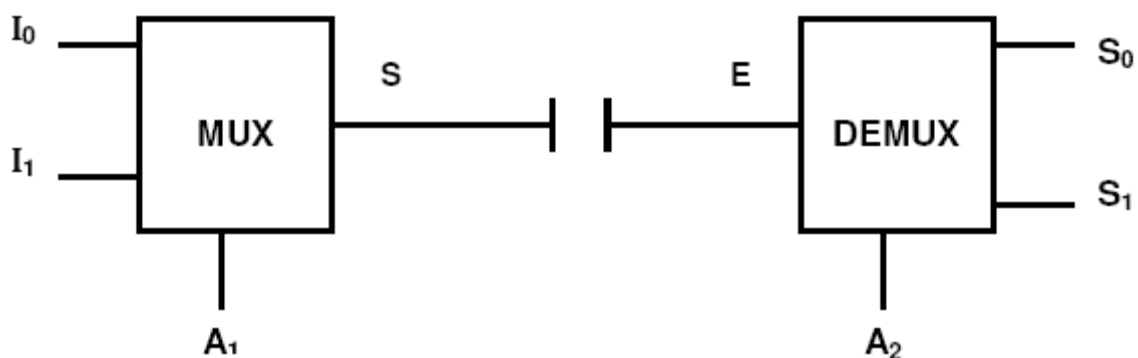
LINHA DE TRANSMISSÃO



14.3.2 Transmissão Série

A configuração do circuito é vista na figura.

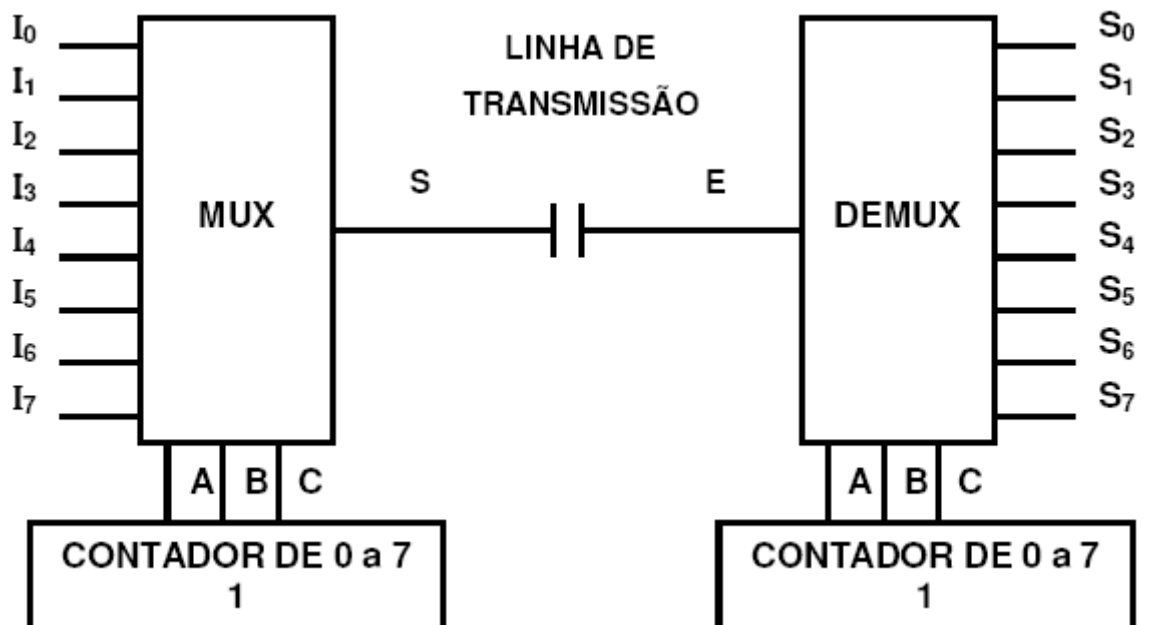
LINHA DE TRANSMISSÃO



Neste caso, a entrada da informação é feita por 2 fios (2 bits de informação) e é transmitida através de um único fio. Na recepção, teremos a conversão para saída em 2 fios, como na entrada.

O processo apresenta a vantagem de transmitir a informação de modo série. Este fato é muito importante quando temos uma grande distância entre o transmissor e o receptor, pois a linha de transmissão poderá ser simplesmente um par de fios, linha telefônica ou, ainda, um sistema mais complexo utilizando **fibras ópticas**.

Vejamos a seguir, um sistema de transmissão de dados, utilizando multiplex e demultiplex de 8 canais de informação, ambos com endereçamento seqüencial:



REFERÊNCIAS

CAPUANO, Francisco Gabriel; IDOETA, Ivan Valeije. **Elementos de Eletrônica Digital**. 33. Ed. São Paulo, Érica, 2002, 526p.

MALVINO, Albert Paul, LEACH, Donald P. **Eletrônica Digital Princípios e Aplicações**; lógica seqüencial. São Paulo, McGraw-Hill, 1987, 2 v.

PHILIPS SEMICONDUCTORES. **Fast TTL Logic Series**. Data Handbook, 1992.

CAPUANO, F.G. Exercícios de Eletrônica Digital. São Paulo: Érica, 1996.