CURSO DE ELETRÔNICA DIGITAL

INTRODUÇÃO

Os circuitos equipados com processadores, cada vez mais, estão fazendo parte do cotidiano do técnico e/ou engenheiro, tanto de campo como de desenvolvimento.

Hoje, dificilmente encontramos um equipamento, seja ele de consumo ou de produção, que não possua pelo menos um processador (DSP, microprocessador, ou microcontrolador).

É fato também que vários profissionais encontram muitas dificuldades na programação e desenvolvimento de projetos com esses componentes, simplesmente por terem esquecido alguns conceitos fundamentais da eletrônica digital clássica.

A intenção desse "especial" é justamente essa, ou seja, cobrir possíveis lacunas sobre essa tecnologia de modo simples e objetivo. Procuramos complementar a teoria com circuitos práticos e úteis, e dividimos o trabalho em doze capítulos:

- Sistemas de numeração
- Álgebra de Boole e portas lógicas
- Família TTL
- Família CMOS
- Funções lógicas
- Flip-Flops
- Funções lógicas integradas
- Multivibradores
- Contadores
- Decodificadores
- Registradores de deslocamento
- Displays

Tivemos o cuidado de elaborar alguns testes, para que o leitor possa acompanhar melhor sua percepção.

Newton C. Braga

LIÇÃO 1

ELETRÔNICA ANALÓGICA E DIGITAL
SISTEMAS DE NUMERAÇÃO

1.1- ANALÓGICO E DIGITAL

Por que digital? Esta é certamente a primeira pergunta que qualquer leitor que está "chegando agora" e tem apenas alguma base teórica sobre Eletrônica faria ao encontrar o nosso curso.

Por este motivo, começamos justamente por explicar as diferenças entre as duas eletrônicas, de modo que elas fiquem bem claras. Devemos lembrar que em muitos equipamentos, mesmo classificados como analógicos ou digitais, encontraremos os dois tipos de circuitos. É o caso dos computadores, que mesmo sendo classificados como "máquinas estritamente digitais" podem ter em alguns pontos de seus circuitos configurações analógicas.

Uma definição encontrada nos livros especializados atribui o nome de Eletrônica Digital aos circuitos que operam com quantidades que só podem ser incrementadas ou decrementadas em passos finitos.

Um exemplo disso é dado pelos circuitos que operam com impulsos. Só podemos ter números inteiros de pulsos sendo trabalhados em qualquer momento em qualquer ponto do circuito. Em nenhum lugar encontraremos "meio pulso" ou "um quarto de pulso".

A palavra digital também está associada a dígito (do latim digitu, dedo) que está associado à representação de quantidades inteiras. Não podemos usar os dedos para representar meio pulso ou um quarto de pulso.

Na Eletrônica Analógica trabalhamos com quantidades ou sinais que podem ter valores que variam de modo contínuo numa escala. Os valores dos sinais não precisam ser inteiros. Por exemplo, um sinal de áudio, que é analógico, varia suavemente entre dois extremos, enquanto que um sinal digital só pode variar aos saltos, observe a **figura 1**.

Conforme o leitor pode perceber, a diferença básica entre os dois tipos de eletrônica está associada inicialmente ao tipo de sinais com que elas trabalham e no que elas fazem com os sinais.

De uma forma resumida podemos dizer que:

A Eletrônica Digital trabalha com sinais que só podem assumir valores discretos ou inteiros.

A Eletrônica Analógica trabalha com sinais que podem ter qualquer valor entre dois limites.

1.2 - LÓGICA DIGITAL

Os computadores e outros equipamentos que usam circuitos digitais funcionam obedecendo a um tipo de comportamento baseado no que se denomina Lógica.

Diferentemente dos circuitos amplificadores comuns que simplesmente amplificam, atenuam ou realizam algum tipo de processamento simples dos sinais, os circuitos digitais usados em computadores e outras máquinas não processam os sinais baseados em uma finalidade simples determinada quando são fabricados.

Os circuitos digitais dos computadores e outros equipamentos são capazes de combinar os sinais tomando decisões segundo um comportamento lógico.

É evidente que se o leitor deseja realmente entender como as coisas acontecem nos circuitos digitais, deve partir exatamente do aprendizado do comportamento lógico. Podemos dizer que a lógica nos permite tirar

COMPUTADORES: os computadores atuais são digitais em sua totalidade e praticamente não é usado outro tipo de configuração. No entanto, nem sempre foi assim. Nas primeiras décadas deste século, quando os circuitos eram ainda valvulados, os primeiros computadores eram máquinas analógicas. A imprecisão e algumas outras dificuldades técnicas que estes computadores apresentavam fizeram com que logo fossem substituídos pelos circuitos digitais hoje usados.

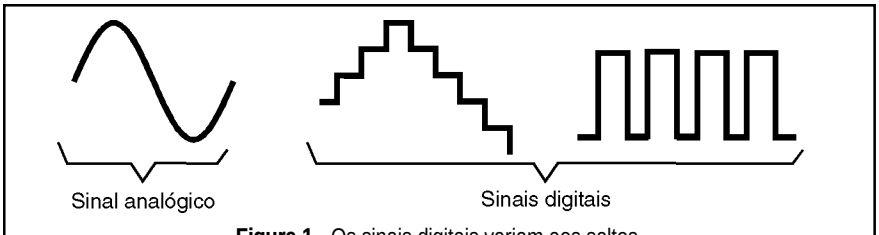


Figura 1 - Os sinais digitais variam aos saltos.

conclusões ou tomar decisões a partir de fatos conhecidos.

Por exemplo, a decisão de "acender uma lâmpada quando está escuro" é uma decisão lógica, pois a proposição e a conclusão são fatos relacionados.

Ao contrário, a decisão de "acender uma lâmpada, porque está chovendo" não é uma decisão lógica, pois os fatos envolvidos não têm relação.

Evidentemente, os fatos relacionados acima são simples e servem para exemplificar como as coisas funcionam.

Na eletrônica dos computadores, o que temos é a aplicação da lógica digital, ou seja, de circuitos que operam tomando decisões em função de coisas que acontecem no seu próprio interior. É claro que os computadores e seus circuitos digitais não podem entender coisas como está escuro ou está chovendo e tomar decisões.

Os circuitos lógicos digitais trabalham com sinais elétricos.

Assim, os circuitos lógicos digitais nada mais fazem do que receber sinais com determinadas características e em função destes tomar decisões que nada mais são do que a produção de um outro sinal elétrico.

Mas, se os sinais elétricos são digitais, ou seja, representam quantidades discretas e se a lógica é baseada em tomada de decisões, o próximo passo no entendimento da Eletrônica Digital, é partir para o modo como as quantidades discretas são representadas e entendidas pelos circuitos eletrônicos.

1.3 - SISTEMAS DE NUMERAÇÃO

O modo como contamos as quantidades vem do fato de possuirmos 10 dedos. Assim, tomando os dedos das mãos podemos contar objetos com facilidade até certo ponto.

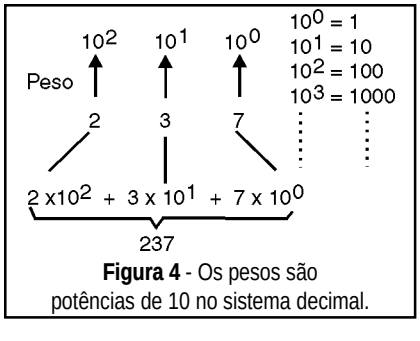


Figura 4 - Os pesos são potências de 10 no sistema decimal.

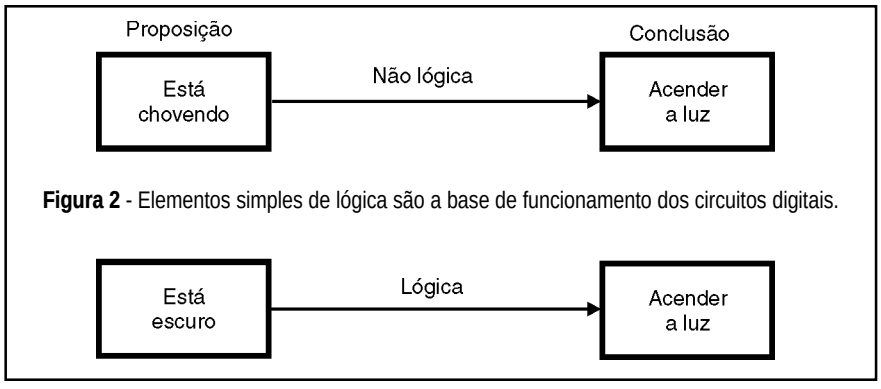


Figura 2 - Elementos simples de lógica são a base de funcionamento dos circuitos digitais.

O ponto crítico ocorre quando temos quantidades maiores do que 10. O homem resolveu o problema passando a indicar também a quantidade de mãos ou de vezes em que os dez dedos eram usados.

Assim, quando dizemos que temos 27 objetos, o 2 indica que temos "duas mãos cheias" ou duas dezenas mais 7 objetos. O 2 tem peso 10.

Da mesma forma, quando dizemos que temos 237 objetos, o 2 indica que temos "duas dezenas de mãos cheias" ou duas centenas, enquanto o 3 indica que temos mais 3 mãos cheias e finalmente o 7, mais 7 objetos, **figura 3**. Em outras palavras, a posição dos algarismos na representação dos números tem um peso e em nosso sistema de numeração que é decimal este peso é 10, veja a **figura 4**.

O que aconteceria se tivéssemos um número diferente de dedos, por exemplo 2 em cada mão?

Isso significaria, em primeiro lugar, que em nosso sistema de base 4 (e não base 10) só existiriam 4 algarismos para representar os números: 0, 1, 2 e 3, confira a **figura 5**.

Para representar uma quantidade maior do que 4 teríamos de usar mais de um algarismo.

Assim, para indicar 7 objetos na base 4, teríamos "uma mão cheia com 4" e mais 3. Isso daria 13, **figura 6**.

Veja então que no "13" na base 4, o 1 tem peso 4, enquanto que o 3 tem o seu valor normal.

De uma forma generalizada, dizemos que dependendo da base do sistema os algarismos têm "pesos" que correspondem à sua posição no número e que estes pesos são potências da base. Por exemplo, para a base 10, cada algarismo a partir da direita tem um peso, que é uma potência de 10 em ordem crescente, o que nos leva à unidade (dez elevado a zero), à dezena (dez elevado ao expoente um), à centena (dez elevado ao quadrado), ao milhar (dez elevado ao cubo) e assim por diante, conforme a **figura 7**.

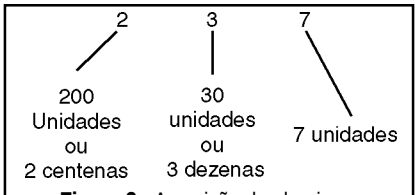


Figura 3 - A posição do algarismo dá seu valor relativo.

Em Eletrônica Digital costumamos dizer que o dígito mais à direita, por representar a menor potência ou ter menor peso, é o dígito ou bit* menos significativo ou LSB (*Less Significant Bit*) enquanto que o mais à esquerda é o mais significativo ou MSB (*Most Significant Bit*). Para a base 4, conforme observamos na figura 8, os dígitos têm potências de 4.

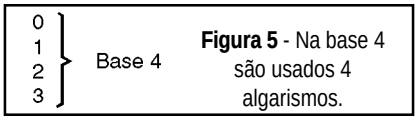


Figura 5 - Na base 4 são usados 4 algarismos.

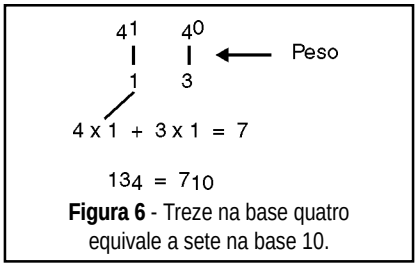


Figura 6 - Treze na base quatro equivale a sete na base 10.

*O bit que é o dígito binário (na base 2) será estudado mais adiante.

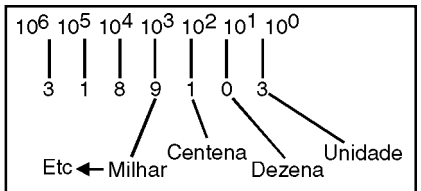


Figura 7 - Os pesos aumentam da direita para a esquerda.

1.4 - NUMERAÇÃO BINÁRIA

Os circuitos eletrônicos não possuem dedos.

É evidente também que não seria muito fácil projetar circuitos capazes de reconhecer 10 níveis de uma tensão ou de outra grandeza elétrica sem o perigo de que qualquer pequeno problema fizesse-os causar qualquer confusão.

Muito mais simples para os circuitos eletrônicos é trabalhar com um sistema de numeração que esteja mais de acordo com o seu princípio de funcionamento e isso realmente é feito.

Um circuito eletrônico pode ter ou não corrente, ter ou não tensão, pode receber ou não um pulso elétrico.

Ora, os circuitos eletrônicos são mais apropriados para operar com sinais que tenham duas condições possíveis, ou seja, que representem dois dígitos ou algarismos.

Também podemos dizer que as regras que regem o funcionamento dos circuitos que operam com apenas duas condições possíveis são muito mais simples.

Assim, o sistema adotado nos circuitos eletrônicos digitais é o sistema binário ou de base 2, onde são usados apenas dois dígitos, correspondentes a duas condições possíveis de um circuito: 0 e 1.

Mas, como podemos representar qualquer quantidade usando apenas dois algarismos?

A idéia básica é a mesma usada na representação de quantidades no sistema decimal: atribuir pesos aos dígitos conforme sua posição no número. Assim, vamos tomar como exemplo o valor 1101 que em binário representa o número 13 decimal e ver como isso ocorre.

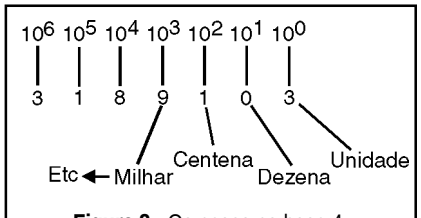


Figura 8 - Os pesos na base 4.

O primeiro dígito da direita nos indica que temos uma vez o peso deste dígito ou 1.

O zero do segundo dígito da direita para a esquerda indica que não temos nada com o peso 2.

Agora o terceiro dígito da direita para a esquerda e que tem peso 4 é 1, o que indica que temos “uma vez quatro”.

Finalmente, o primeiro dígito da esquerda que é 1 e está na posição de peso 8, nos diz que temos “uma vez oito”.

Somando uma vez oito, com uma vez quatro e uma vez um, temos o total, justamente a quantidade que conhecemos em decimal como treze.

Veja então, conforme indica a **figura 9**, que na numeração binária, os dígitos vão tendo pesos da direita para a esquerda que são potências de 2, ou seja, dois elevado ao expoente zero que é um, dois elevado ao expoente 1 que é 2, dois ao quadrado que é 4 e assim por diante.

Basta lembrar que a cada vez que nos deslocamos para a esquerda, o peso do dígito dobra, **figura 10**.

Como não existe um limite para os valores dos pesos, isso significa que é posível representar qualquer quantidade em binário, por maior que seja, simplesmente usando o número apropriado de dígitos.

Para 4 dígitos podemos representar números até 15; para 8 dígitos podemos ir até 255; para 16 dígitos até 65 535 e assim por diante.

O leitor deve lembrar-se desses valores limites para 4, 8 e 16 dígitos de um número binário, pois eles têm uma grande importância na Informática.

A seguir damos a representação binária dos números decimais até 17 para uma melhor ilustração de como tudo funciona:

Figura 9 - Pesos na numeração binária.

Decimal	Binário	Decimal	Binário
0	0	9	1001
1	1	10	1010
2	10	11	1011
3	11	12	1100
4	100	13	1101
5	101	14	1110
6	110	15	1111
7	111	16	10000
8	1000	17	10001

Para o leitor que pretende entender de Eletrônica Digital aplicada aos computadores há momentos em que é preciso saber converter uma indicação em binário para o decimal correspondente.

Podemos dar como exemplo o caso de certas placas que são usadas no diagnóstico de computadores e que possuem um conjunto de LEDs que acende indicando um número correspondente a um código de erros. Os LEDs apagados indicam o algarismo 0 e os LEDs acesos, o algarismo 1.

Vamos supor que num diagnóstico a sequência de acendimento dos LEDs seja 1010110. É preciso saber por onde começar a leitura ou seja, se o de menor peso é o da direita ou da esquerda.

Nas indicações dadas por instrumentos ou mesmo na representação da valores binários, como por exemplo na saída de um circuito, é preciso saber qual dos dígitos tem maior peso e qual tem menor peso.

Isso é feito com uma sigla adotada normalmente e que se refere ao dígito, no caso denominado bit.

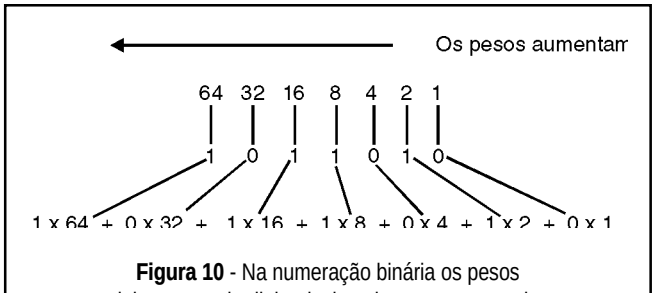


Figura 10 - Na numeração binária os pesos dobram a cada digito deslocado para a esquerda.

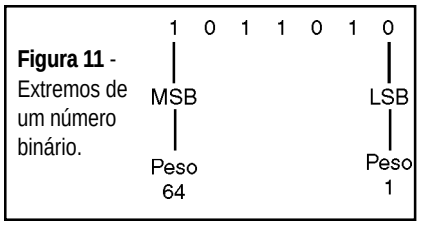


Figura 11 - Extremos de um número binário.

Assim, conforme citado anteriormente, para o dígito de menor peso ou bit menos significativo é adotada a sigla LSB (*Less Significant Bit*) e para o mais significativo é adotada a sigla MSB (*Most Significant Bit*), figura 11.

O que fazemos é somar os valores dados pelos dígitos multiplicados pelo peso de sua posição. No caso do valor tomado como exemplo, 1010110, temos:

Dígito		Peso		Valor
1	x	64	=	64
0	x	32	=	0
1	x	16	=	16
0	x	8	=	0
1	x	4	=	4
1	x	2	=	2
0	x	1	=	0

Somando os valores teremos:
64 + 16 + 4 + 2 = 86

O valor decimal de 1010110 é 86.

Assim, tudo que o leitor tem de fazer é lembrar que a cada dígito que saltamos para a esquerda seu peso dobra na sequência 1, 2, 4, 8, 16, 32, 64, 128, etc.

Na prática também pode ocorrer o problema inverso, transformação de um valor expresso em decimal (base 10) para a base 2 ou binário.

Para esta transformação podemos fazer uso de algoritmo muito simples que memorizado pelo leitor pode ser de grande utilidade, dada sua praticidade.

Para os que não sabem, algoritmo nada mais é do que uma sequência de operações que seguem uma determinada regra e permitem realizar uma operação mais complexa. Quando você soma os números um sobre o outro (da mesma coluna) e passa para cima os dígitos que excedem o 10, fazendo o conhecido "vai um", você nada mais está fazendo do que usar um algoritmo.

Os computadores usam muitos tipos de algoritmos quando fazem suas operações, se bem que a maioria não precise ser conhecida dos leitores.

Assim, para a conversão de um decimal para binário, como por exemplo o 116, o que fazemos é uma série de divisões sucessivas, **figura 12**.

Vamos dividindo os números por 2 até o ponto em que chegamos a um valor menor que 2 e que portanto, não pode mais ser dividido.

O resultado desta última divisão, ou seja, seu quociente é então o primeiro dígito binário do número convertido. Os demais dígitos são obtidos lendo-se os restos da direita para a esquerda da série de divisões que realizamos. Tudo muito simples e rápido.

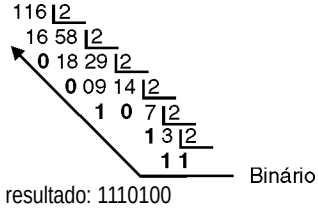


Figura 12 - Conversão de um decimal em binário por divisões sucessivas.

1.5 - BINÁRIOS MENORES QUE 1

Para o leitor talvez seja difícil entender como usando quantidades que só podem ser inteiras, como dado pela definição de digital no início desta lição, seja possível representar quantidades menores que um, ou seja, números "quebrados" ou fracionários.

É claro que isso é possível na prática, pois se assim não fosse os computadores e as calculadoras não poderiam realizar qualquer operação com estes números e sabemos que isso não é verdade.

O que se faz é usar um artifício que consiste em empregar potências negativas de um número inteiro para representar quantidades que não são inteiras.

Assim é possível usar dígitos binários para representar quantidades fracionárias sem problemas.

Vamos dar um exemplo tomando o número 0,01101 em binário.

A própria existência de um "0," já nos sugere que se trata de um número menor que 1 e portanto, fracionário.

Ocorre que os dígitos deste número têm pesos que correspondem a potências de 2 negativas, que nada mais são do que frações, conforme a seguinte sequência:

Dígito		Peso		Valor
0,	x	1	=	0
0	x	1/2	=	0
1	x	1/4	=	0,25
1	x	1/8	=	0,0625
0	x	1/16	=	0
1	x	1/32	=	0,03125

Somando os valores relativos teremos:
0,25 + 0,0625 + 0,03125 = 0,625

O número decimal representado é portanto 0,625.

Veja que usando tantos dígitos quantos sejam necessários podemos representar com a precisão desejada um número decimal.

1.6 - FORMAS DIFERENTES DE UTILIZAR O SISTEMA BINÁRIO

A utilização de circuitos eletrônicos com determinadas características e a própria necessidade de adaptar o sistema binário à representação de valores que sejam convertidos rapidamente para o decimal e mesmo outros sistemas, levou ao aparecimento de algumas formas diferentes de utilização dos binários.

Estas formas são encontradas em diversos tipos de equipamentos digitais, incluindo os computadores.

Sistema BCD (Decimal Codificado em Binário)

BCD é a abreviação de *Binary Coded Decimal* e se adapta melhor aos circuitos digitais.

Permite transformar cada dígito decimal de um número numa representação por quatro dígitos binários (bits) independentemente do valor total do número que será representado.

Assim, partimos da seguinte tabela:

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

Se quisermos representar em BCD o número 23,25 não o convertemos da forma convencional por divisões sucessivas mas sim, tomamos cada dígito e o convertemos no BCD equivalente, conforme segue:

2 3, 2 5
0010 0011 0010 0101

Veja então que para cada dígito decimal sempre teremos quatro dígitos binários ou bits e que os valores 1010, 1011, 1100, 1101 e 1111 não existem neste código.

Esta representação foi muito interessante quando as calculadoras se tornaram populares, pois era possível usá-las para todas as operações com números comuns e os 5 códigos não utilizados dos valores que não existiam foram adotados para indicar as operações! (figura 13)

O leitor também perceberá que usando representações desta forma, operavam os primeiros computadores, apropriadamente chamados de computadores de “4 bits”.

Outros Códigos

Outros códigos binários, mas não tão importantes neste momento, são o Código Biquinário, em que cada dígito tem um peso e são sempre usados 7 bits para sua representação e o Código Gray que aparece em diversas versões.

O Código Gray se caracteriza pelo fato da passagem de qualquer número para o seguinte sempre ser feita com a mudança de um único dígito.

Assim, por exemplo, quando passamos de 0111 (7 em decimal) para 1000 (8 em decimal) os quatro dígitos mudam. No Código Gray a passagem do 7 para 8 muda apenas um dígito, pois o 7 é 0100 e o 8 é 1100.

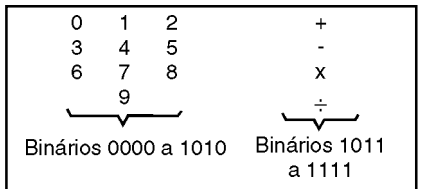


Figura 13 - Uso dos valores de 0000 a 1111.

Podemos ainda citar os Códigos de Paridade de Bit e o Código de Excesso 3 (XS3) encontrados em aplicações envolvendo circuitos digitais.

1.7 - SISTEMA HEXADECIMAL

Os bits dos computadores são agrupados em conjuntos de 4, assim temos os computadores de 4, 8, 16 e 32 bits. Também observamos que com 4 bits podemos obter representações binárias de 16 números e não somente de 10. Vimos que os 5 excedentes poderiam ser usados para representar operações nas calculadoras.

Isso significa que a representação de valores no sistema hexadecimal ou de base 16 é mais compatível com a numeração binária ou operação binária dos computadores.

E de fato isso é feito: abrindo muitos programas de um computador, vemos que suas características como posições de memória ou quantidade de memória são feitas neste sistema.

Isso significa que o técnico precisa conhecer este sistema e mais do que isso, deve saber como fazer conversões dele para o decimal e vice-versa, além de conversões para o sistema binário. Na tabela abaixo damos as representações dos dígitos deste sistema com equivalentes decimais e binários:

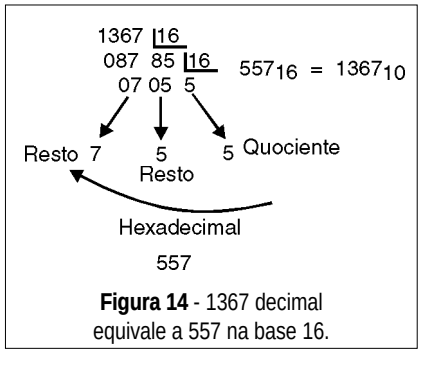


Figura 14 - 1367 decimal equivale a 557 na base 16.

Decimal	Binário	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Observe que como não existem símbolos para os dígitos 10, 11, 12, 13, 14 e 15, foram usadas as letras A,B,C,D,E e F.

Como fazer as conversões: os mesmos procedimentos que vimos para o caso das conversões de decimal para binário e vice-versa são válidos para o caso dos hexadecimais, mudando-se apenas a base.

Vamos dar exemplos:

Como converter 4D5 em decimal:

Os pesos no caso são: 256, 16 e 1. (a cada dígito para a esquerda multiplicamos o peso do anterior por 16 para obter novo peso).

Temos então:

4D5 = (4 x 256)+(13x16)+(1x5) = 1237

Observe que o “D” corresponde ao 13. O número decimal equivalente ao 4D5 hexadecimal ou “hex”, como é muitas vezes representado, é 1237.

4D5 (hex) = 1237 (dec)

A conversão inversa, ou seja, de decimal para hexadecimal é feita por divisões sucessivas. Tomemos o caso de 1256, apresentado na figura 14.

Veja que basta ler o quociente final e depois os restos das divisões sucessivas, sempre lembrando que os que excederem 10 devem ser “trocados” pelas letras equivalentes.

EXERCÍCIOS

a) Converter 645 em BCD
b) Converter 45 em binário puro
c) Converter 11001 (binário) em decimal
d) Converter 1101 0011 1011 (BCD) em decimal
e) Converter 1745 (decimal) em hexadecimal.
f) Converter FFF (hex) em decimal.
g) Converter F4D (hex) em decimal.

LIÇÃO 2

A ÁLGEBRA DE BOOLE

Na primeira lição do nosso curso aprendemos o significado das palavras **Digital** e **Lógica** empregadas na Eletrônica e nos computadores. Vimos que os computadores são denominados digitais quando trabalham com sinais discretos, ou seja, sinais que não variam continuamente entre dois valores, mas que assumem determinados valores inteiros. Também vimos que os computadores são máquinas lógicas, porque tomam decisões a partir de certos fatos, segundo regras muito bem estabelecidas. Vimos que no caso dos circuitos digitais, como os usados nos computadores, a base 10 não é a mais apropriada e que estes equipamentos usam principalmente o sistema binário e hexadecimal. Aprendemos ainda como fazer as conversões de base e ler os números binários e hexadecimais.

Nesta lição veremos de que modo os circuitos digitais podem tomar decisões lógicas. Todas essas decisões são baseadas em circuitos muito simples e configurações que operam na base 2 e que portanto, são fáceis de entender, porém muito importantes para os leitores que pretendam trabalhar com computadores, ou pelo menos entender melhor seu princípio de funcionamento.

2.1 - A álgebra de Boole

Em meados do século passado George Boole, um matemático inglês, desenvolveu uma teoria completamente diferente para a época, baseada em uma série de postulados e operações simples para resolver uma infinidade de problemas.

Apesar da algebra de Boole, como foi chamada, poder resolver problemas práticos de controle e fabricação de produtos, na época não havia Eletrônica e nem as máquinas eram suficientemente avançadas para utilizar seus princípios.

A álgebra de Boole veio a se tornar importante com o advento da Eletrônica, especificamente, da Eletrônica Digital, que gerou os modernos computadores.

Boole estabelece em sua teoria que só existem no universo duas condições possíveis ou estados, para qualquer coisa que se deseje analisar e estes dois estados são opostos.

Assim, uma lâmpada só pode estar acesa ou apagada, uma torneira só pode estar aberta ou fechada, uma fonte só pode ter ou não ter tensão na sua saída, uma pergunta só pode ter como resposta verdadeiro ou falso. Dizemos de maneira simples que na álgebra de Boole as variáveis lógicas só podem adquirir dois estados:

0 ou 1
Verdadeiro ou Falso
Aberto ou Fechado
Alto ou Baixo (HI ou LO)
Ligado ou Desligado

Na Eletrônica Digital partimos justamente do fato de que um circuito só pode trabalhar com dois estados possíveis, ou seja, encontraremos presença do sinal ou a ausência do sinal, o que se adapta perfeitamente aos princípios da álgebra de Boole.

Tudo que um circuito lógico digital pode fazer está previsto pela álgebra de Boole. Desde as mais simples operações ou decisões, como acender um LED quando dois sensores são ativados de uma determinada maneira ou quando uma tecla é pressionada, até girar no espaço uma imagem tridimensional.

2.2 - Os níveis lógicos

Partimos então do fato de que nos circuitos digitais só encontraremos duas condições possíveis: presença ou ausência de sinal, para definir alguns pontos importantes para o nosso entendimento.

Nos circuitos digitais a presença de uma tensão será indicada como 1 ou HI (de *HIGH* ou Alto) enquanto que a ausência de uma tensão será indicada por 0 ou LO (de *LOW* ou baixo).

O 0 ou LO será sempre uma tensão nula, ou ausência de sinal num ponto do circuito, mas o nível lógico 1 ou HI pode variar de acordo com o circuito considerado (**figura 1**). Nos PCs de mesa, a tensão usada para a alimentação de todos os circuitos lógicos, por exemplo, é de 5 V. Assim, o nível 1 ou HI de seus circuitos será

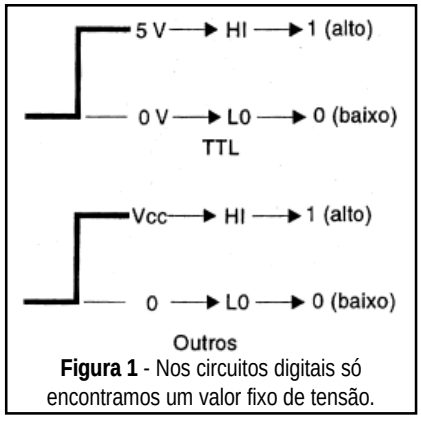


Figura 1 - Nos circuitos digitais só encontramos um valor fixo de tensão.

sempre uma tensão de 5 V. Nos *laptops* é usada uma tensão de alimentação menor, da ordem de 3,2 V, portanto, nestes circuitos um nível 1 ou HI sempre corresponderá a uma tensão desse valor.

Existem ainda circuitos digitais que empregam componentes de tecnologia CMOS e que são alimentados tipicamente por tensões entre 3 e 15 V. Nestes casos, conforme vemos na **figura 2,** um nível lógico 1 ou HI poderá ter qualquer tensão entre 3 e 15 V, dependendo apenas da tensão de alimentação usada.

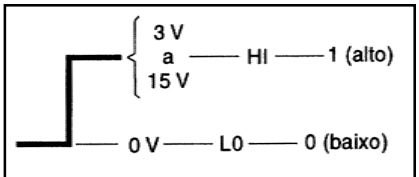


Figura 2 - A tensão encontrada nos circuitos CMOS terá um valor fixo entre 3 e 15 V.

Na verdade, a idéia de associar a presença de tensão ao nível 1 e a ausência ao nível 0, é mera questão de convenção.

Nada impede que adotemos um critério inverso e projetemos os circuitos, pois eles funcionarão perfeitamente.

Assim, quando dizemos que ao nível alto (1) associamos a presença de tensão e ao nível baixo a ausência de tensão (0), estamos falando do que se denomina "lógica positiva".

Se associarmos o nível baixo ou 0 a presença de tensão e o nível alto ou 1 a ausência de tensão, estaremos falando de uma "lógica negativa", conforme ilustra a **figura 3**.

Para não causar nenhum tipo de confusão, todo o nosso curso tratará exclusivamente da lógica positiva, o mesmo acontecendo com os dispositivos eletrônicos tomados como exemplos.

Portanto, em nossa lógica, é possível associar os seguintes estados de um circuito aos valores 0 e 1:

0 V
Falso
Desligado
Nível baixo ou LO

1 - 5 V (ou outra tensão positiva, conforme o circuito)
Verdadeiro
Ligado
Nível alto ou HI

3.1 - Operações Lógicas

No dia-a-dia estamos acostumados a realizar diversos tipos de operações lógicas, as mais comuns são as que envolvem números, ou seja, quantidades que podem variar ou variáveis.

Assim, podemos representar uma soma como:

$$Y = A + B$$

Onde o valor que vamos encontrar para Y depende dos valores atribuídos às letras A e B.

Dizemos que temos neste caso uma função algébrica e que o valor Y é a variável dependente, pois seu valor dependerá justamente dos valores de A e B, que são as variáveis independentes.

Na Eletrônica Digital, entretanto, existem operações mais simples do que a soma, e que podem ser perfeitamente implementadas levando em conta a utilização da álgebra booleana.

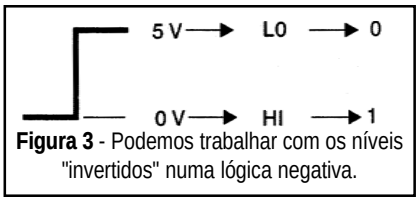


Figura 3 - Podemos trabalhar com os níveis "invertidos" numa lógica negativa.

É interessante observar que com um pequeno número destas operações conseguimos chegar a uma infinidade de operações mais complexas, como por exemplo, as utilizadas nos computadores e que, repetidas em grande quantidade ou levadas a um grau de complexidade muito grande, nos fazem até acreditar que a máquina seja "inteligente"!

Na verdade, é a associação, de determinada forma das operações simples que nos leva ao comportamento muito complexo de muitos circuitos digitais, conforme ilustra a **figura 4**.

Assim, como observamos na **figura 5**, um computador é formado por

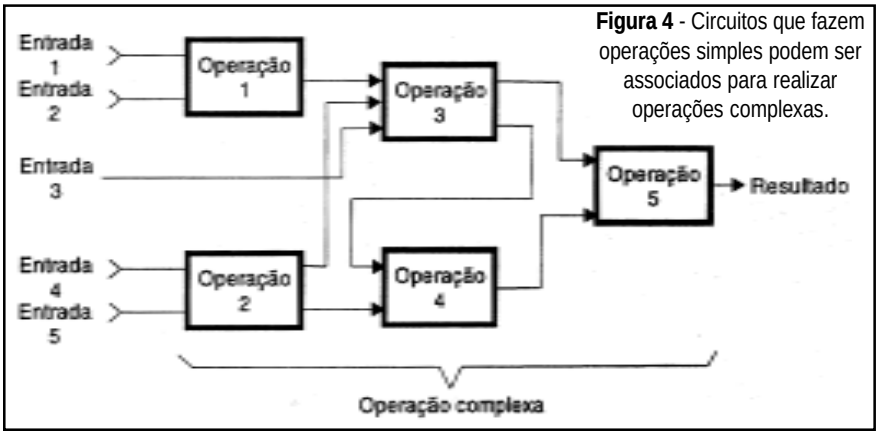


Figura 4 - Circuitos que fazem operações simples podem ser associados para realizar operações complexas.

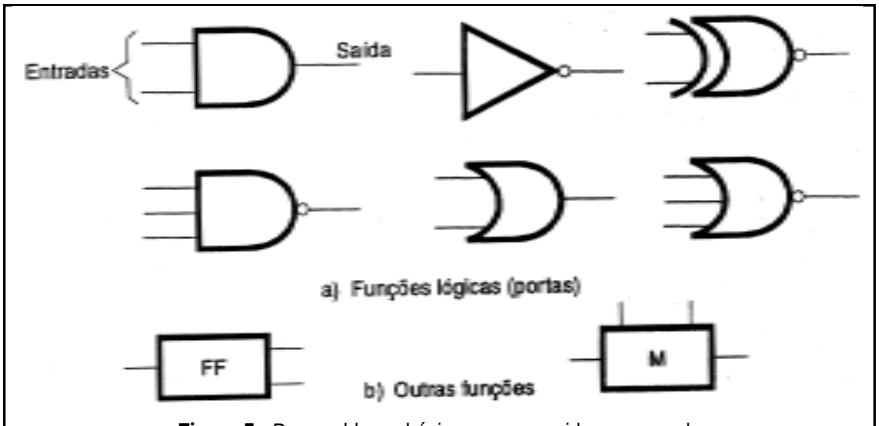


Figura 5 - Poucos blocos básicos, mas reunidos em grande quantidae podem realizar operações muito complexas.

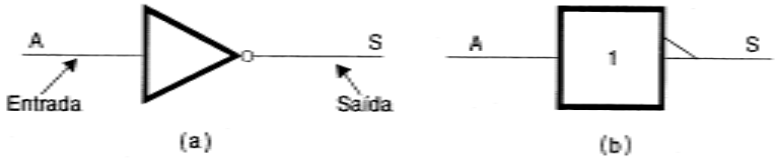


Figura 6 - Em (a) o simbolo mais comum e em (b) o simbolo IEEE usado em muitas publicações técnicas mais modernas dos Estados Unidos e Europa.

um grande número de pequenos blocos denominados portas ou funções em que temos entradas e saídas.

O que irá aparecer na saída é determinado pela função e pelo que acontece nas entradas. Em outras palavras, a resposta que cada circuito lógico dá para uma determinada entrada ou entradas depende do que ele é ou de que "regra booleana" ele segue.

Isso significa que para entender como o computador realiza as mais complexas operações teremos de começar entendendo como ele faz as operações mais simples com as denominadas portas e quais são elas.

Por este motivo, depois de definir estas operações lógicas, associando-as à álgebra de Boole, vamos estudá-las uma a uma.

2.4 - Função Lógica NÃO ou Inversora

Nos manuais também encontramos a indicação desta função com a palavra inglesa correspondente, que é NOT.

O que esta função faz é negar uma afirmação, ou seja, como em álgebra booleana só existem duas respostas possíveis para uma pergunta, esta função "inverte" a resposta, ou seja, a resposta é o "inverso" da pergunta. O circuito que realiza esta operação é denominado inversor.

Levando em conta que este circuito diz sim, quando a entrada é não, ou que apresenta nível 0, quando a entrada é 1 e vice-versa, podemos associar a ele uma espécie de tabela que será de grande utilidade sempre que estudarmos qualquer tipo de circuito lógico.

Esta tabela mostra o que ocorre com a saída da função quando colocamos na entrada todas as combinações possíveis de níveis lógicos.

Dizemos que se trata de uma "tabela verdade" (nos manuais em Inglês esta tabela aparece com o nome de *Truth Table)*. A seguir apresentamos a tabela verdade para a porta NOT ou inversora:

Entrada	Saída
0	1
1	0

Os símbolos adotados para representar esta função são mostrados na **figura 6**.

O adotado normalmente em nossas publicações é o mostrado em (a), mas existem muitos manuais técnicos e mesmo diagramas em que são adotados outros e os leitores devem conhecê-los.

Esta função pode ser simulada por um circuito simples e de fácil entendimento apresentado na **figura 7**.

Neste circuito temos uma lâmpada que, acesa, indica o nível 1 na saída e apagada, indica o nível 0. Quando a chave está aberta indicando que a entrada é nível 0, a lâmpada está acesa, indicando que a saída é nivel 1. Por outro lado, quando a chave é fechada, o que representa uma entrada 1, a lâmpada apaga, indicando que a saída é zero.

Esta maneira de simular funções lógicas com lâmpadas indicando a saída e chaves indicando a entrada, é bastante interessante pela facilidade com que o leitor pode entender seu funcionamento.

Basta então lembrar que:

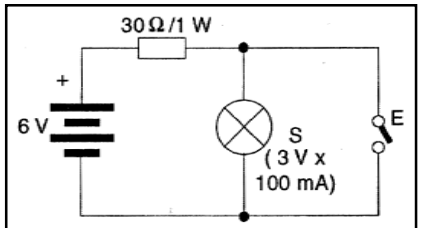


Figura 7 - Circuito simples para simular a função NÃO (NOT) ou inversor.

Entrada: chave aberta = 0
chave fechada = 1
Saída: lâmpada apagada = 0
lâmpada acesa = 1

2.5 - Função Lógica E

A função lógica E também conhecida pelo seu nome em inglês AND pode ser definida como aquela em que a saída será 1 se, e somente se, **todas** as variáveis de entrada forem 1.

Veja que neste caso, as funções lógicas E podem ter duas, três, quatro ou quantas entradas quisermos e é representada pelos símbolos mostrados na **figura 8**.

As funções lógicas também são chamadas de "portas" ou *"gates"* (do inglês) já que correspondem a circuitos que podem controlar ou deixar passar os sinais sob determinadas condições.

Tomando como exemplo uma porta ou função E de duas entradas, escrevemos a seguinte tabela verdade:

Entradas		Saída
A	B	
0	0	0
0	1	0
1	0	0
1	1	1

Na **figura 9** apresentamos o modo de simular o circuito de uma porta E

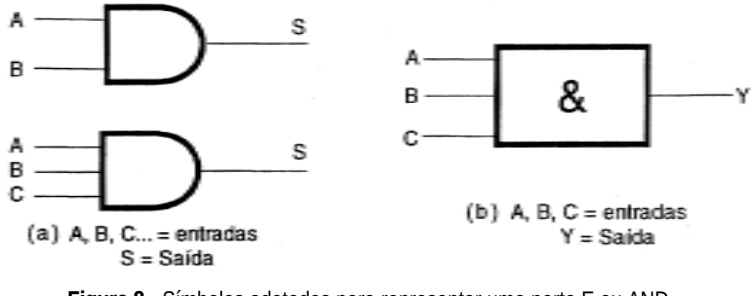


Figura 8 - Símbolos adotados para representar uma porta E ou AND.

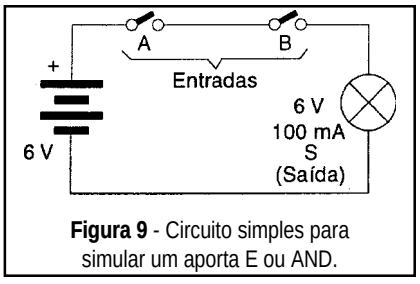


Figura 9 - Circuito simples para simular um aporta E ou AND.

usando chaves e uma lâmpada comum. É preciso que S_1 e S_2 estejam fechadas, para que a saída (lâmpada) seja ativada.

Para uma porta E de três entradas tabela verdade será a seguinte:

Entradas			Saída
A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Para que a saída seja 1, é preciso que todas as entradas sejam 1.

Observamos que para uma porta E de 2 entradas temos 4 combinações possíveis para os sinais aplicados. Para uma porta E de 3 entradas temos 8 combinações possíveis para o sinal de entrada.

Para uma porta de 4 entradas, teremos 16 e assim por diante.

2.6 - Função lógica OU

A função OU ou ainda OR (do inglês) é definida como aquela em que a saída estará em nível alto se uma ou mais entradas estiver em nível alto. Esta função é representada pelos símbolos mostrados na **figura 10**.

O símbolo adotado normalmente em nossas publicações é o mostrado em (a).

Para uma porta OU de duas entradas podemos elaborar a seguinte tabela verdade:

Entradas		Saída
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

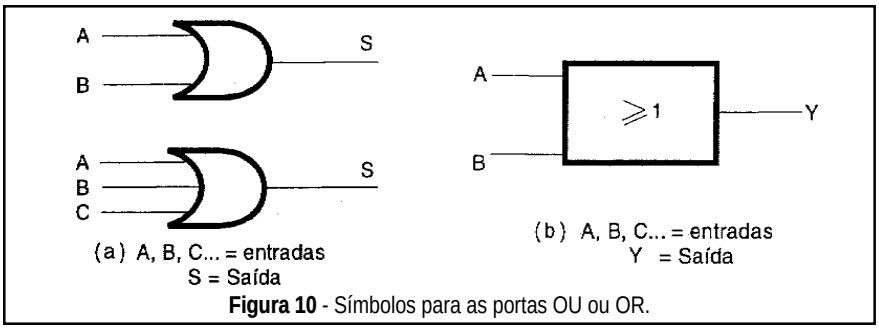


Figura 10 - Símbolos para as portas OU ou OR.

Vemos que a saída estará no nível 1 se uma das entradas estiverem no nível 1.

Um circuito simples com chaves e lâmpada para simular esta função é dado na **figura 11**.

Quando uma chave estiver fechada (entrada 1) a lâmpada receberá corrente (saída 1), conforme desejarmos. Para mais de duas variáveis podemos ter portas com mais de duas entradas. Para o caso de uma porta OU de três entradas teremos a seguinte tabela verdade:

Entradas			Saída
A	B	C	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

2.7 - Função NÃO-E

As funções E, OU e NÃO (inversor) são a base de toda a álgebra booleana e todas as demais podem ser consideradas como derivadas delas. Vejamos:

Uma primeira função importante derivada das anteriores é a obtida pela associação da função E com a função NÃO, ou seja, a negação da função E que é denominada NÃO-E ou em inglês, NAND.

Na **figura 12** temos os símbolos adotados para representar esta função.

Observe a existência de um pequeno círculo na saída da porta para indicar a negação.

Podemos dizer que para a função NAND a saída estará em nível 0 se, e somente se, **todas** as entradas estiverem em nível 1.

A tabela verdade para uma porta NÃO-E ou NAND de duas entradas é a seguinte:

Entradas		Saída
A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

Na **figura 13** temos um circuito simples com chaves, que simula esta função.

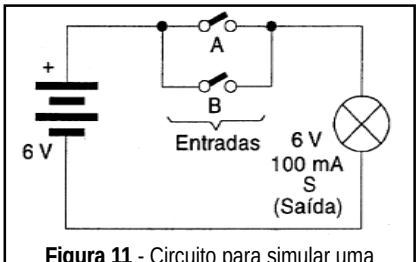


Figura 11 - Circuito para simular uma porta OU ou OR de duas entradas.

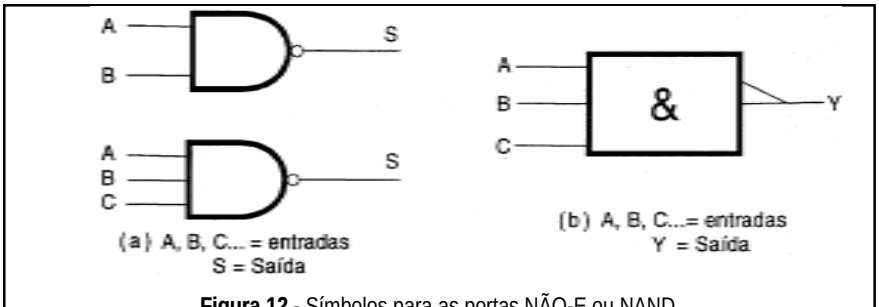


Figura 12 - Símbolos para as portas NÃO-E ou NAND.

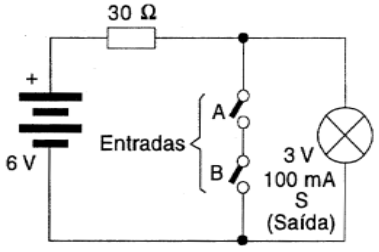


Figura 13 - Circuito que simula uma porta NAND ou NÃO-E de duas entradas.

Veja que a lâmpada só apagará (saída 0) quando as duas chaves estiverem fechadas (1), curto-circuitando assim sua alimentação. O resistor é usado para limitar a corrente da fonte.

Também neste caso podemos ter a função NAND com mais de duas entradas. Para o caso de 3 entradas teremos a seguinte tabela verdade:

Entradas			Saída
A	B	C	S
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

2.8 - Função NÃO-OU

Esta é a negação da função OU, obtida da associação da função OU com a função NÃO ou inversor. O termo inglês usado para indicar esta função é NOR e seus símbolos são apresentados na **figura 14**.

Sua ação é definida da seguinte forma: a saída será 1 se, e somente se, **todas** as variáveis de entrada forem 0.

Uma tabela verdade para uma função NOR de duas entradas é mostrada a seguir:

Entradas		Saída
A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

Um circuito simples usando chaves e lâmpada para simular esta função é mostrado na **figura 15**.

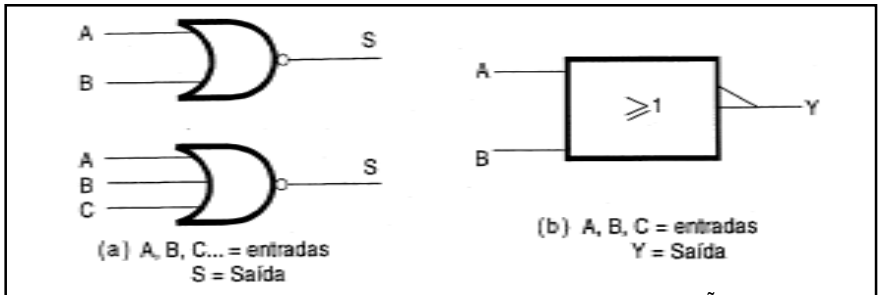


Figura 14 - Símbolo usados para representar a função NOR ou NÃO-E

Observe que a lâmpada só se mantém acesa (nível 1) se as duas chaves (S_1 e S_2) estiverem abertas (nível 0).

Da mesma forma que nas funções anteriores, podemos ter portas NOR com mais de duas entradas. Para o caso de três entradas teremos a seguinte tabela verdade:

Entradas			Saída
A	B	C	S
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

2.9 - Função OU-exclusivo

Uma função de grande importância para o funcionamento dos circuitos lógicos digitais e especificamente para os computadores é a denominada OU-exclusivo ou usando o termo inglês, *"exclusive-OR".* Esta função tem a propriedade de realizar a soma de valores binários ou ainda encontrar o que se denomina "paridade" (o que será visto futuramente).

Na **figura 16** temos os símbolos adotados para esta função.

Podemos definir sua ação da seguinte forma: a saída será 1 se, e somente se, as variáveis de entrada forem diferentes. Isso significa que, para uma porta *Exclusive-OR* de duas entradas teremos saída 1 se as entradas forem 0 e 1 ou 1 e 0, mas a saída será 0 se as entradas forem ambas 1 ou ambas 0, conforme a seguinte tabela verdade:

Entradas		Saída
A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Esta função é derivada das demais, pois podemos "montá-la" usando portas conhecidas (**figura 17**).

Assim, se bem que esta função tenha seu próprio símbolo e possa ser considerada um "bloco" independente nos projetos, podemos sempre implementá-la com um circuito equivalente como o ilustrado nessa figura.

2.10 - Função NÃO-OU exclusivo ou coincidência

Podemos considerar esta função como o "inverso" do OU-exclusivo. Sua denominação em inglês é *Exclusive*

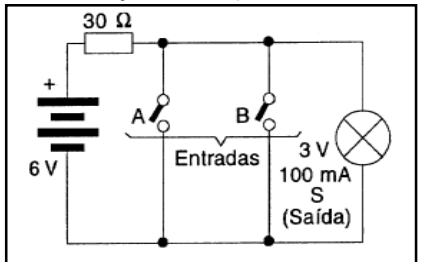


Figura 15 - Circuito usado para simular uma porta NOR de duas entradas.

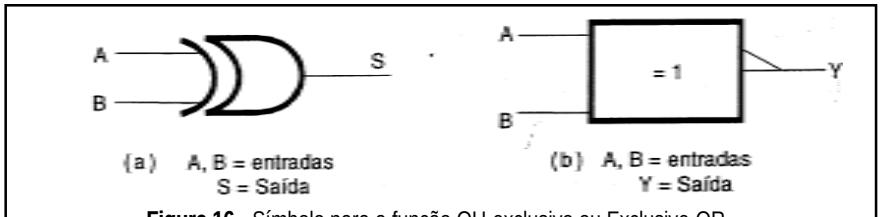


Figura 16 - Símbolo para a função OU-exclusivo ou Exclusive-OR.

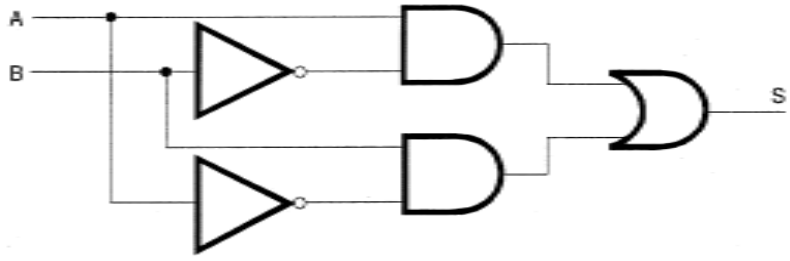


Figura 17 - Elaboração da função OU-exclusivo com inversores, portas AND e uma porta OR.

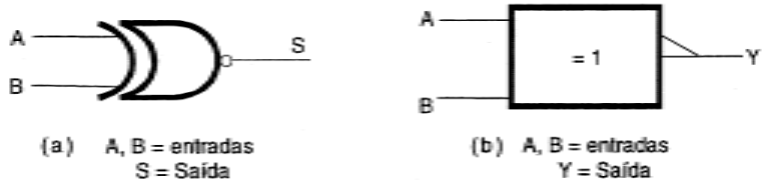


Figura 18 - Símbolos da função Não-OU-Exclusive ou Exclusive NOR também chamada função coincidência.

NOR e é representada pelo símbolo mostrado na **figura 18**.

Observe o círculo que indica a negativa da função anterior, se bem que essa terminologia são seja apropriada neste caso.

Esta função pode ser definida como a que apresenta uma saída igual a 1 se, e somente se as variáveis de entrada forem iguais.

Uma tabela verdade para esta função é a seguite:

Entrada		Saída
A	B	S
0	0	1
0	1	0
1	0	0
1	1	1

Podemos implementar esta função usando outras já conhecidas, conforme a **figura 19**.

2.11 - Propriedades das operações lógicas

As portas realizam operações com os valores binários aplicados às suas entradas. Assim, podemos representar estas operações por uma simbologia apropriada, facilitando o projeto dos circuitos e permitindo visualizar melhor o que ocorre quando associamos muitas funções.

No entanto, para saber associar as diversas portas e com isso realizar operações mais complexas, é preciso conhecer as propriedades que as operações apresentam.

Exatamente como no caso das operações com números decimais, as operações lógicas com a álgebra Booleana se baseiam numa série de postulados e teoremas algo simples.

Os principais são dados a seguir e prová-los fica por conta dos leitores que desejarem ir além. Para entender, entretanto, seu significado não é preciso saber como provar sua validade, mas sim memorizar seu significado.

Representações

As operações E, OU e NÃO são representadas por símbolos da seguinte forma:

a) Operação E

A operação E é representada por um ponto final(.). Assim, para uma porta E de duas entradas (A e B) e saída S podemos fazer a representação:

A . B = S

b) Operação OU

Esta operação é representada pelo sinal (+).

A operação de uma porta OU de entradas A e B e saída S pode ser representada como:

A + B = S

c) Operação NÃO

Esta operação é indicada por uma barra da seguinte forma:

A\ = S

Partindo destas representações, podemos enumerar as seguintes propriedades das operações lógicas:

1. Propriedade comutativa das operações E e OU:

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

2. Propriedade associativa das operações E e OU:

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

3. Teorema da Involução:

(A negação da negação é a própria afirmação)

A\\ = A

4. A operacão E é distributiva em relação à operação OU:

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

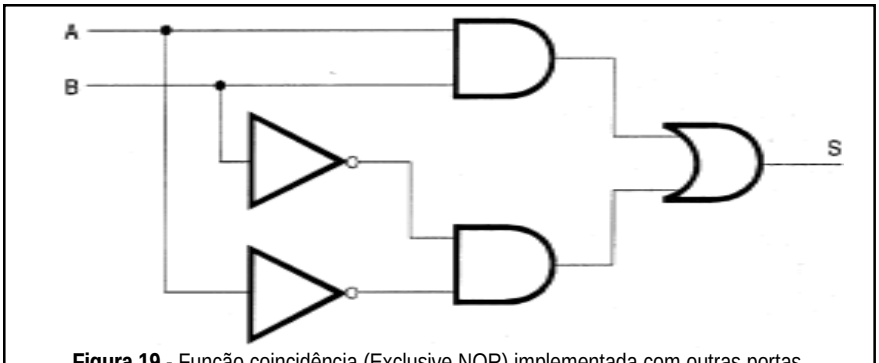


Figura 19 - Função coincidência (Exclusive NOR) implementada com outras portas.

5. Propriedades diversas:

A.A = A
A+A = A
A.0 = 0
A.1 = A
A+0 = A
A+1 = 1
A.A\= 0
A+A\= 1
A+A.B = A

6. Teoremas de De Morgan:

Aplicando a operação NÃO a uma operação E, o resultado obtido é igual ao da operação OU aplicada aos complementos das variáveis de entrada.

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

Aplicando a operação NÃO a uma operação OU o resultado é igual ao da operação E aplicada aos complementos das variáveis de entrada.

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

2.12 - Fazendo tudo com portas NAND

As portas NÃO-E, pelas suas características, podem ser usadas para obter qualquer outra função que estudamos. Esta propriedade torna essas portas blocos universais nos projetos de circuitos digitais já que, na forma de circuitos integrados, as funções NAND são fáceis de obter e baratas.

A seguir vamos mostrar de que modo podemos obter as funções estudadas simplesmente usando portas NAND.

Inversor

Para obter um inversor a partir de uma porta NAND basta unir suas entradas ou colocar uma das entradas no nível lógico 1, conforme **figura 2**0.

Uma porta E (AND) é obtida simplesmente agregando-se à função NÃO-E (NAND) um inversor em cada entrada, (**figura 21**).

A função OU (OR) pode ser obtida com o circuito mostrado na **figura 22**. O que se faz é inverter a saída depois de aplicá-la a uma porta NAND.

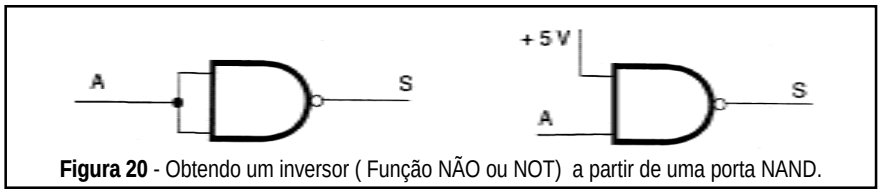


Figura 20 - Obtendo um inversor (Função NÃO ou NOT) a partir de uma porta NAND.

2.13 - Conclusão

Os princípios em que se baseiam os circuitos lógicos digitais podem parecer algo abstratos, pois usam muito de Matemática e isso talvez desestimule os leitores. No entanto, eles são apenas o começo. O esforço para entendê-los certamente será recompensado, pois estes princípios estão presentes em tudo que um computador faz. Nas próximas lições, quando os princípios estudados começarem a tomar uma forma mais concreta, aparecendo em circuitos e aplicações práticas será fácil entendê-los melhor.

Nas próximas lições, o que foi estudado até agora ficará mais claro quando encontrarmos sua aplicação prática.

QUESTIONÁRIO

1. Se associarmos à presença de uma tensão o nível lógico 1 e à sua ausência o nível 0, teremos que tipo de lógica:
   a) Digital b) Positiva
   c) Negativa d) Booleana

2. Na entrada de uma função lógica NÃO aplicamos o nível lógico 0. A saída certamente será:
   a) 0 b) 1
   c) Pode ser 0 ou 1
   d) Estará indefinida

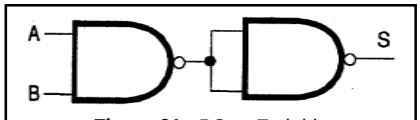


Figura 21 - POrta E obtida com duas NÀO-E (NAND).

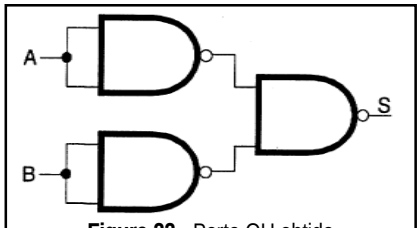


Figura 22 - Porta OU obtida com duas NÃO-E (NAND).

3. O circuito que realiza a operação lógica NÃO é denominado:
   a) Porta lógica b) Inversor
   c) Amplificador digital
   d) Amplificador analógico

4. Se na entrada de uma porta NAND aplicarmos os níveis lógicos 0 e 1, a saída será:
   a) 0
   b) 1
   c) Pode ser 0 ou 1
   d) Estará indefinida

5. Em qual das seguintes condições de entrada a saída de uma porta OR será 0:
   a) 0,0 b) 0,1
   c) 1,0 d) 1,1

6. Qual é o nome da função lógica em que obtemos uma saída 1 quando as entradas tiverem níveis lógicos diferentes, ou seja, forem 0 e 1 ou 1 e 0.
   a) NAND
   b) NOR
   c) AND
   d) Exclusive OR

7. Qual é a porta que pode ser utilizada para implementar qualquer função lógica:
   a) Inversor (NÃO)
   b) AND
   c) NAND
   d) OR

Respostas da lição nº 1
a) 0110 0100 0101
b) 101101
c) 25
d) Sem resposta (1101 não existe)
e) 131
f) 131
g) 334

LIÇÃO 3

FAMÍLIAS DE CIRCUITOS LÓGICOS DIGITAIS

Na lição anterior conhecemos os princípios simples da Álgebra de Boole que regem o funcionamento dos circuitos lógicos digitais encontrados nos computadores e em muitos outros equipamentos. Vimos de que modo umas poucas funções simples funcionam e sua importância na obtenção de funções mais complexas. Mesmo sendo um assunto um pouco abstrato, por envolver princípios matemáticos, o leitor pode perceber que é possível simular o funcionamento de algumas funções com circuitos eletrônicos relativamente simples, usando chaves e lâmpadas.

Os circuitos eletrônicos modernos, entretanto, não usam chaves e lâmpadas, mas sim, dispositivos muito rápidos que podem estabelecer os níveis lógicos nas entradas das funções com velocidades incríveis e isso lhes permite realizar milhões de operações muito complexas a cada segundo.

Nesta edição veremos que tipo de circuitos são usados e como são encontrados na prática em blocos básicos que unidos podem levar a elaboração de circuitos muito complicados como os encontrados nos computadores.

O leitor irá começar a tomar contato com componentes práticos das famílias usadas na montagem dos equipamentos digitais. São estes os componentes básicos que podem ser encontrados em circuitos digitais, computadores e muitos outros.

3.1 - O transistor como chave eletrônica

Um transistor pode funcionar como um interruptor deixando passar ou não uma corrente, conforme a aplicação de uma tensão em sua entrada.

Assim, na simulação dos circuitos que estudamos e em que usamos chaves, é possível utilizar transistores com uma série de vantagens.

No caso das chaves, o operador era responsável pela entrada do sinal, pois, atuando com suas mãos sobre a chave, deveria estabelecer o nível lógico de entrada, mantendo esta chave aberta ou fechada conforme desejasse 0 ou 1.

Se usarmos um transistor teremos uma vantagem importante: o transistor poderá operar com a tensão ou nível lógico produzido por uma outra função e não necessariamente por uma pessoa que acione uma chave.

Assim, as funções lógicas implementadas com transistores têm a vantagem de poderem ser interligadas umas nas outras, pois o sinal que aparece na saída de cada uma pode ser usado como entrada para outra, conforme a **figura 1**.

Na figura 1 damos um exemplo interessante de como podemos obter um inversor usando um transistor.

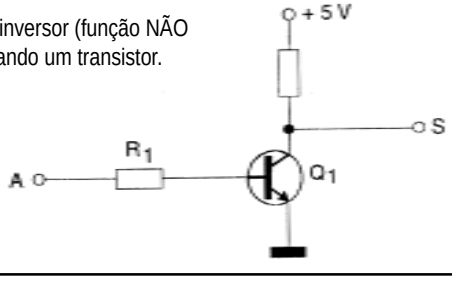
Aplicando o nível 1 na base do transistor ele conduz até o ponto de saturar, o que faz, com que a tensão no seu coletor caia a 0. Por outro lado, na ausência de tensão na sua base, que corresponde ao nível 0 de entrada, o transistor se mantém cortado e a tensão no seu coletor se mantém alta, o que corresponde ao nível 1.

Conforme observamos na **figura 2**, outras funções podem ser conseguidas com transistores.

Isso significa que a elaboração de um circuito lógico digital capaz de realizar operações complexas usando transistores é algo que pode ser conseguido com relativa facilidade.

3.2 - Melhorando o desempenho

No entanto, usar transistores em circuitos que correspondam a cada função de uma maneira não padronizada pode trazer algumas dificuldades.



E	S
0	5V
5V	0

Figura 1 - Um inversor (função NÃO ou NOT) usando um transistor.

Dessa forma, se bem que nos primeiros tempos da Eletrônica Digital cada função era montada com seus transistores, diodos e resistores na sua plaquinha para depois serem todas interligadas, este procedimento se revelou inconveniente por diversos motivos.

O primeiro deles é a complexidade que o circuito adquiria se realizasse muitas funções.

O segundo, é a necessidade de padronizar o modo de funcionamento de cada circuito ou função. Seria muito importante estabelecer que todos os circuitos operassem com a mesma tensão de alimentação e fornecessem sinais que os demais pudessem reconhecer e reconhecessem os sinais gerados pelos outros.

O desenvolvimento da tecnologia dos circuitos integrados, possibilitando a colocação num único invólucro de diversos componentes já interligados, veio permitir um desenvolvimento muito rápido da Eletrônica Digital.

Foi criada então uma série de circuitos integrados que continham numa única pastilha as funções lógicas digitais mais usadas e de tal maneira projetadas que todas eram compatíveis entre si, ou seja, operavam com as mesmas tensões e reconheciam os mesmos sinais.

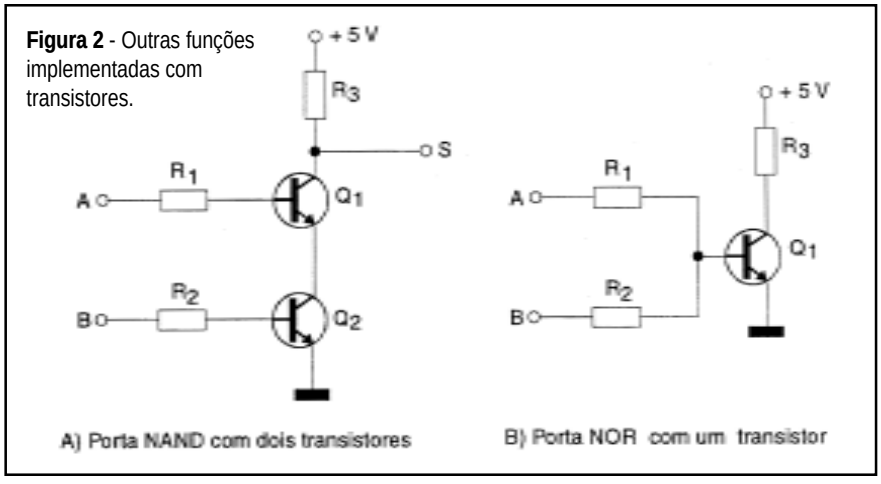


Figura 2 - Outras funções implementadas com transistores.

Estas séries de circuitos integrados formaram então as Famílias Lógicas, a partir das quais os projetistas tiveram facilidade em encontrar todos os blocos para montar seus equipamentos digitais.

Assim, conforme a **figura 3**, precisando montar um circuito que usasse uma porta AND duas NOR e inversores, o projetista teria disponíveis componentes compatíveis entre si contendo estas funções e de tal forma que poderiam ser interligadas das maneiras desejadas.

O sucesso do advento dessas famílias foi enorme, pois além do menor tamanho dos circuitos e menor consumo de energia, havia ainda a vantagem do menor custo e obtenção de maior velocidade de operação e confiabilidade.

Diversas famílias foram criadas desde o advento dos circuitos integrados, recebendo uma denominação conforme a tecnologia empregada.

As principais famílias lógicas desenvolvidas foram:

- *RTL* ou *Resistor Transistor Logic*
- *RCTL* ou *Resistor Capacitor Transistor Logic*
- *DTL* ou *Diode Transistor Logic*
- *TTL* ou Transistor Transistor Logic
- *CMOS* ou *Complementary Metal Oxid Semiconductor*
- *ECL* ou *Emitter Coupled Logic*

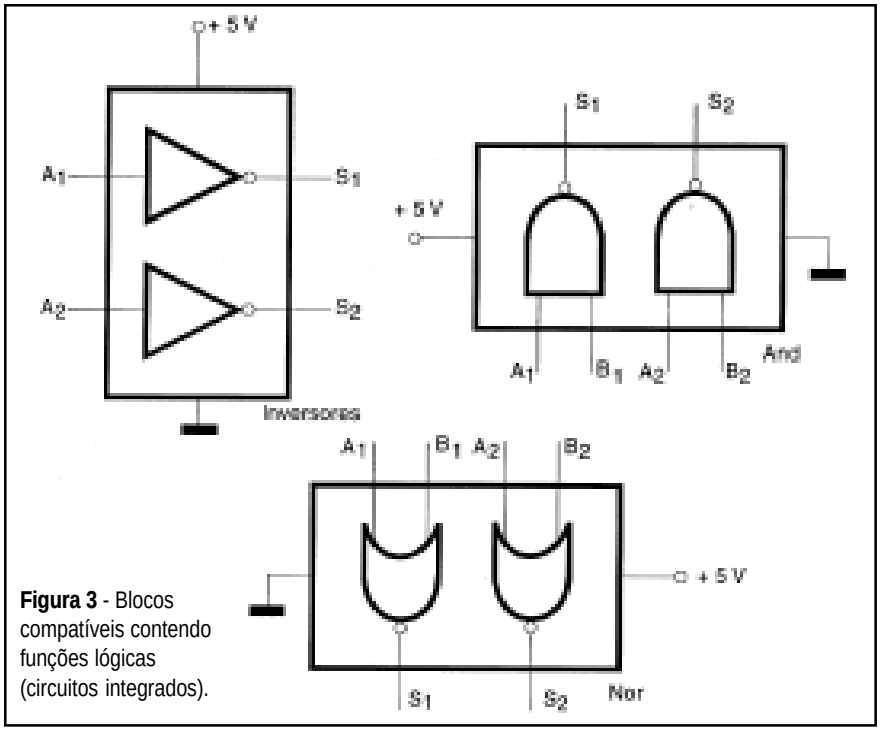


Figura 3 - Blocos compatíveis contendo funções lógicas (circuitos integrados).

Atualmente a Família TTL e a CMOS são as mais usadas, sendo empregadas em uma grande quantidade de equipamentos digitais e também nos computadores e periféricos.

3.3 - A família TTL

A família TTL foi originalmente desenvolvida pela Texas Instruments, mas hoje, muitos fabricantes de semicondutores produzem seus componentes.

Esta família é principalmente reconhecida pelo fato de ter duas séries que começam pelos números 54 para os componentes de uso militar e 74 para os componentes de uso comercial.

Assim, podemos rapidamente associar qualquer componente que comece pelo número "74" à família TTL.

Na **figura 4** mostramos uma porta típica TTL. Trata-se de uma porta NAND de duas entradas que logo

chama a atenção pelo fato de usar um transistor de dois emissores.

A característica mais importante desta família está no fato de que ela é alimentada por uma tensão de 5 V.

Assim, para os componentes desta família, o nível lógico 0 é sempre a ausência de tensão ou 0 V, enquanto que o nível lógico 1 é sempre uma tensão de +5 V.

Para os níveis lógicos serem reconhecidos devem estar dentro de faixas bem definidas.

Conforme verificamos na **figura 5**, uma porta TTL reconhecerá como nível 0 as tensões que estiverem entre 0 e 0,8 V e como 1 os que estiverem numa outra faixa entre 2,4 e 5 V.

Entre essas duas faixas existe uma região indefinida que deve ser evitada.

Há centenas de circuitos integrados TTL disponíveis no mercado para a realização de projetos. A maioria deles está em invólucros DIL de 14 e 16 pinos, conforme exemplos da **figura 6.**

As funções mais simples das portas disponíveis numa certa quantidade em cada integrado usam circuitos integrados de poucos pinos.

No entanto, à medida que novas tecnologias foram sendo desenvolvidas permitindo a integração de uma grande quantidade de componentes, surgiu a possibilidade de colocar num integrado não apenas umas poucas portas e funções adicionais que serão estudadas futuramente como *flip-flops*, decodificadores e outros mas, também interligá-los de diversas formas e utilizá-los em aplicações específicas.

Diversas etapas no aumento da integração foram obtidas e receberam nomes que hoje são comuns quando

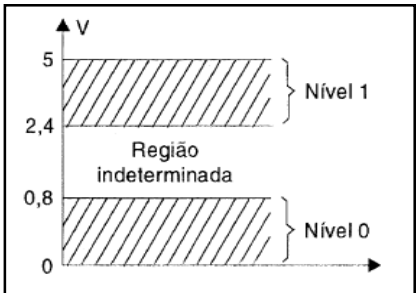


Figura 5 - Faixas de tensão reconhecidas como 0 e 1 (nível alto e baixo).

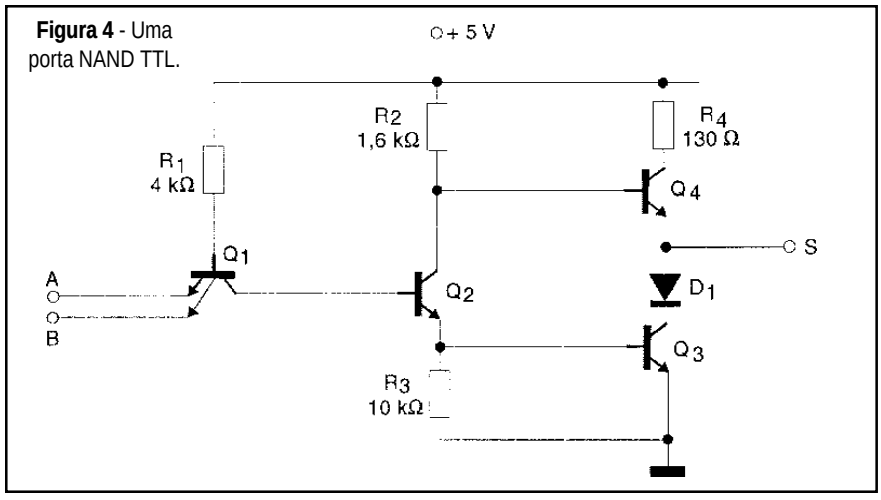


Figura 4 - Uma porta NAND TTL.

falamos de equipamentos digitais e computadores em geral. Temos as seguintes classificações para os graus de integração dos circuitos digitais:

***SSI - Small Scale Integration* ou Integração em Pequena Escala** que corresponde a série normal dos primeiros TTL que contém de 1 a 12 portas lógicas num mesmo componente ou circuito integrado.

***MSI - Medium Scale Integration* ou Integração de Média Escala** em que temos num único circuito integrado de 13 a 99 portas ou funções lógicas.

***LSI - Large Scale Integration* ou Integração em Grande Escala** que corresponde a circuitos integrados contendo de 100 a 999 portas ou funções lógicas.

***VLSI - Very Large Scale Integration* ou Integração em Escala Muito Grande** que corresponde aos circuitos integrados com mais de 1000 portas ou funções lógicas.

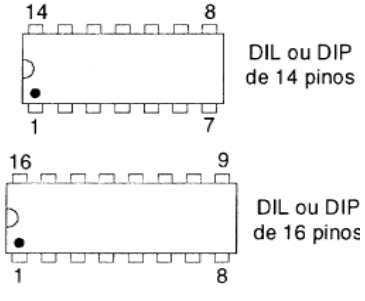


Figura 6 - As funções mais simples TTL são encontradas nestes invólucros.

3.4 - Outras Características da Família TTL

Para usar corretamente os circuitos integrados TTL e mesmo saber como testá-los, quando apresentam algum problema de funcionamento, é importante conhecer algumas de suas características adicionais.

Analisemos as principais características lembrando os níveis lógicos de entrada e saída admitidos:

- Correntes de entrada:

Quando uma entrada de uma função lógica TTL está no nível 0, flui uma corrente da base para o emissor do transistor multiemissor da ordem de 1,6 mA, **figura 7**.

Esta corrente deve ser levada em conta em qualquer projeto, pois, ela deve ser suprida pelo circuito que excitará a porta.

Quando a entrada de uma porta lógica TTL está no nível alto, figura 8, flui uma corrente no sentido oposto da ordem de 40 µA.

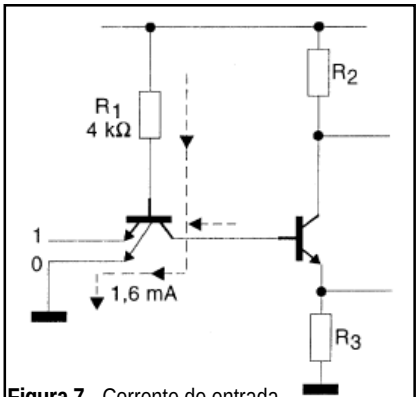


Figura 7 - Corrente de entrada no nível baixo (0).

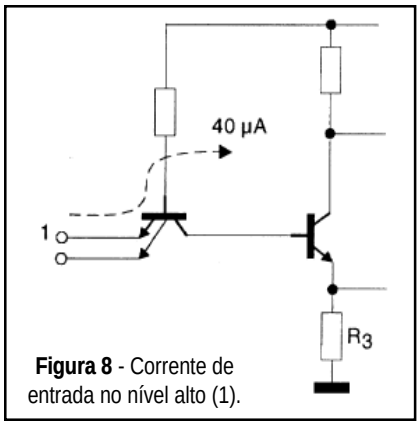


Figura 8 - Corrente de entrada no nível alto (1).

Esta corrente vai circular quando a tensão de entrada estiver com um valor superior a 2,0 V.

- Correntes de saída

Quando a saída de um circuito TTL vai ao nível 0 (ou baixo), flui uma corrente da ordem de 16 mA, conforme observamos no circuito equivalente da **figura 9**.

Isso significa que uma saída TTL no nível 0 ou baixo pode drenar de uma carga uma corrente máxima de 16 mA, ou seja, pode "absorver" uma corrente máxima desta ordem.

Por outro lado, quando a saída de uma função TTL está no nível 1 ou alto, ela pode fornecer uma corrente máxima de 400 µA, **figura 10**.

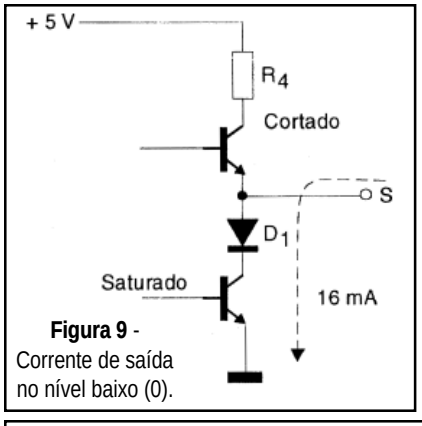


Figura 9 - Corrente de saída no nível baixo (0).

Veja então que podemos obter uma capacidade muito maior de excitação de saída de uma porta TTL quando ela é levada ao nível 0 do que ao nível 1.

Isso justifica o fato de que em muitas funções indicadoras, em que ligamos um LED na saída, fazemos com que ele seja aceso quando a saída vai ao nível 0 (e portanto, a corrente é maior) e não ao nível 1, conforme a **figura 11**.

- Fan In e Fan Out

Estes são termos técnicos que especificam características de extrema importância quando usamos circuitos integrados da família TTL.

A saída de uma porta não precisa estar obrigatoriamente ligada a uma entrada de outra porta. A mesma saída pode ser usada para excitar diversas portas.

Como a entrada de cada porta precisa de uma certa corrente e a saída da porta que irá excitar tem uma capacidade limitada de fornecimento ou de drenar a corrente, é preciso estabelecer um limite para a quantidade de portas que podem ser excitadas, veja o exemplo da **figura 12**.

Assim, levando em conta as correntes nos níveis 1 e 0 das entradas e saídas, definimos o *FAN OUT* como o número máximo de entradas que podemos ligar a uma saída TTL.

Para os componentes da família TTL normal ou *Standard* que estamos estudando, o *FAN OUT* é 10.

Por outro lado, também pode ocorrer que na entrada de uma função lógica TTL precisemos ligar mais de uma saída TTL.

Considerando novamente que circulam correntes nestas ligações e que os circuitos têm capacidades limitadas de condução, precisamos saber até que quantidade de ligações podemos fazer.

Desta forma o *FAN-IN* indica a quantidade máxima de saídas que podemos ligar a uma entrada, **figura 13**.

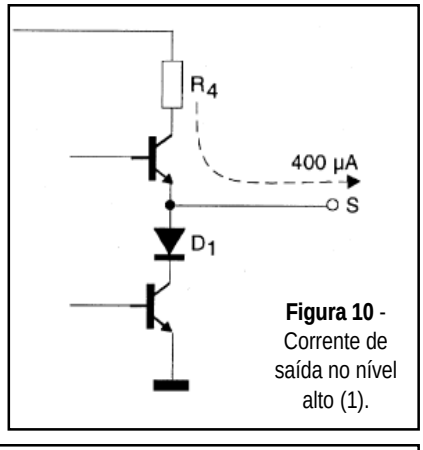


Figura 10 - Corrente de saída no nível alto (1).

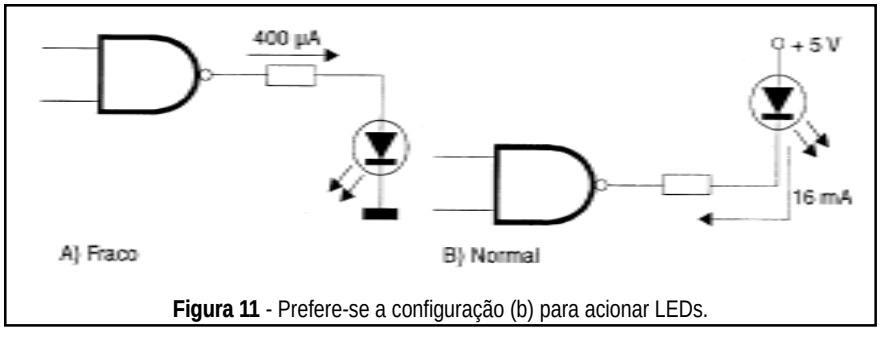


Figura 11 - Prefere-se a configuração (b) para acionar LEDs.

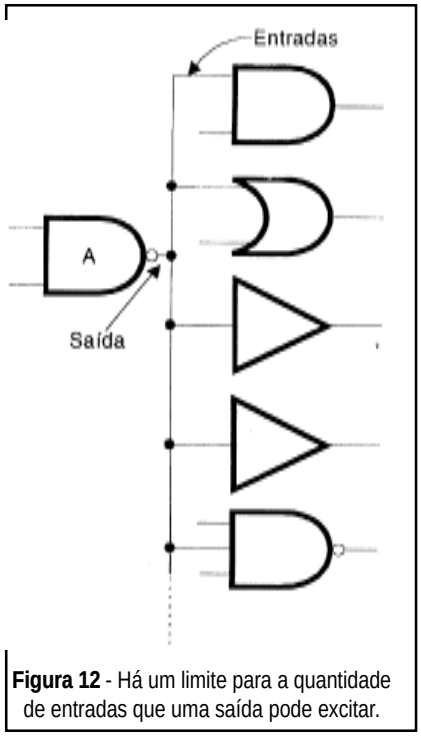


Figura 12 - Há um limite para a quantidade de entradas que uma saída pode excitar.

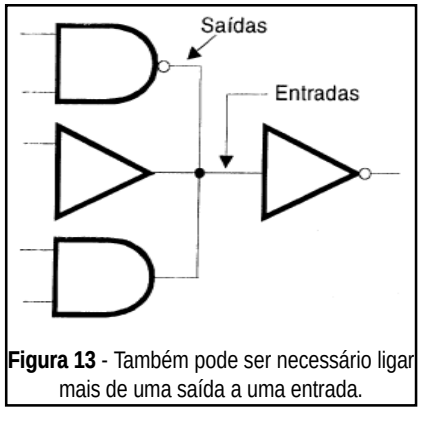


Figura 13 - Também pode ser necessário ligar mais de uma saída a uma entrada.

- Velocidade

Os circuitos eletrônicos possuem uma velocidade limitada de operação que depende de diversos fatores.

No caso específico dos circuitos TTL, temos de considerar a própria configuração das portas que apresentam indutâncias e capacitâncias parasitas que influem na sua velocidade de operação.

Assim, levando em conta a configuração típica de uma porta, conforme observamos no circuito da **figura 14**, veremos que se for estabelecida uma transição muito rápida da tensão de entrada, a tensão no circuito não subirá com a mesma velocidade.

Este sinal terá antes de carregar as capacitâncias parasitas existentes de modo que a tensão de entrada suba gradualmente, demorando um certo tempo que deve ser considerado.

Da mesma forma, à medida que o sinal vai passando pelas diversas etapas do circuito, temos de considerar os tempos que os componentes demoram para comutar justamente em função das capacitâncias e indutâncias parasitas existentes.

O resultado disso é que para os circuitos integrados TTL existe um retardo entre o instante em que o sinal passa do nível 0 para o 1 na entrada e o instante em que o sinal na saída responde a este sinal, passando do nível 1 para o 0 no caso de um inversor.

Da mesma forma, existe um retardo entre o instante em que o sinal de entrada passa do nível 1 para o 0 e o instante em que o sinal de saída passa do nível 0 para o 1, no caso de um inversor.

Mostramos esses dois tempos na **figura 15**, eles são muito importantes nas especificações dos circuitos

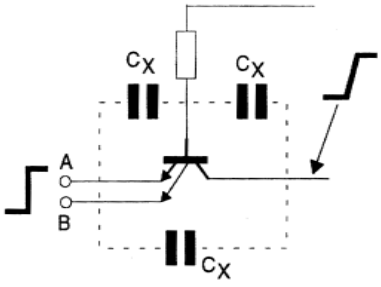


Figura 14 - Capacitâncias parasitas que influem na velocidade de resposta dos circuitos.

TTL, principalmente quando trabalhamos com o projeto de dispositivos muito rápidos. Basicamente podemos adiantar para o leitor que se dois sinais que devam chegar ao mesmo tempo a um certo ponto do circuito não o fizerem, porque um se retarda mais do que o outro ao passar por determinadas funções, isso pode gerar interpretações erradas do próprio circuito que funcionará de modo anormal.

Os primeiros circuitos TTL que foram desenvolvidos logo se mostraram inapropriados para certas aplicações.

3.5 - Subfamílias TTL

Os primeiros circuitos TTL que foram desenvolvidos logo se mostraram inapropriados para certas aplicações, quando é necessária maior velocidade, ou menor consumo de energia ou ainda os dois fatores reunidos.

Isso fez com que, mantendo as características originais de compatibilidade entre os circuitos e mantendo as mesmas funções básicas, fossem criadas sub-famílias que tivessem uma característica adicional diferenciada.

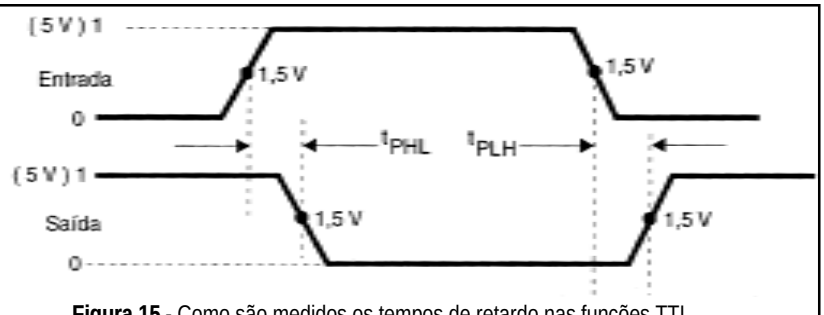


Figura 15 - Como são medidos os tempos de retardo nas funções TTL.

Assim, a partir da família original denominada *"Standard"* surgiram diversas subfamílias. Para diferenciar essas subfamílias, foram adicionadas ao número que identifica o componente (depois do 54 ou 74 com que todos começam), uma ou duas letras.

Temos então a seguinte tabela de subfamílias e da família TTL standard:

Indicação: 54/74
Família/Subfamília: Standard
Característica: nenhuma

Indicação: 54L/74L
Família/Subfamília: Low Power
Característica: Baixo consumo

Indicação: 54H/74H
Família/Subfamília: High Speed
Característica: Alta velocidade
Indicação: 54S/74S
Família/Subfamília: Schottky
Característica: nenhuma

Indicação: 54LS/74LS
Família/Subfamília: Low Power Schottky
Característica: nenhuma

A versão *standard* apresenta componentes com o custo mais baixo e também dispõe da maior quantidade de funções disponíveis.

No entanto, a versão LS se adapta mais aos circuitos de computadores, pois tem a mesma velocidade dos components da família *Standard* com muito menor consumo.

Algumas características podem ser comparadas, para que os leitores verifiquem as diferenças existentes.

- Velocidade

A velocidade de operação de uma função TTL normalmente é especificada pelo tempo que o sinal demora para propagar através do circuito. Em uma linguagem mais simples, trata-se do tempo entre o instante em que aplicamos os níveis lógicos na entrada e o instante em que obtemos a resposta, conforme verificamos através da forma de onda que vimos na **figura 15**.

Para os circuitos da família TTL é comum especificar estes tempos em nanossegundos ou bilionésimos de segundo.

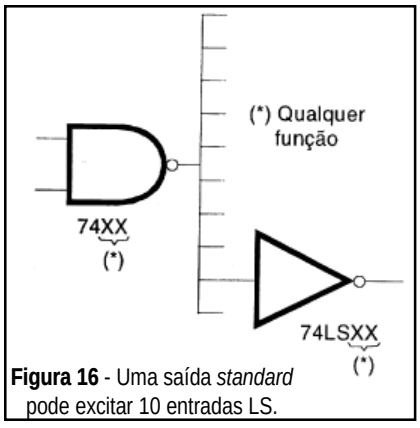


Figura 16 - Uma saída *standard* pode excitar 10 entradas LS.

Assim, temos:

Família/Subfamília: TTL Standart
Tempo de programação (ns): 10

Família/Subfamília: Low Power
Tempo de programação (ns): 33

Família/Subfamília: Low Power Schottlky
Tempo de programação (ns): 10

Família/Subfamília: High Speed
Tempo de programação (ns): 6

Família/Subfamília: Schottkly
Tempo de programação (ns): 3

- Dissipação

Outro ponto importante no projeto de circuitos digitais é a potência consumida e portanto, dissipada na forma de calor. Quando usamos uma grande quantidade de funções, esta característica se torna importante tanto para o dimensionamento da fonte como para o próprio projeto da placa e do aparelho que deve ter meios de dissipar o calor gerado.

Podemos então comparar as dissipações das diversas famílias, tomando como base uma porta ou *gate:*

Família/SubFamília: *Standard*
Dissipação por Gate (mW): 10

Família/SubFamília: *Low Power*
Dissipação por Gate (mW): 1

Família/SubFamília: *Low Power Schottky*
Dissipação por Gate (mW): 2

Família/SubFamília: *High Speed*
Dissipação por Gate (mW): 22

Família/Subfamília: *Schottky*
Dissipação por Gate (mW): 20

O leitor já deve ter percebido um problema importante: quando aumentamos a velocidade, o consumo também aumenta. O projetista deve portanto, ser cuidadoso em escolher a sub- família que una as duas características na medida certa de sua precisão, incluindo o preço.

3.6 - Compatibilidade entre as subfamílias

Um ponto importante que deve ser levado em conta quando trabalhamos com a família *Standard* e as subfamílias TTL é a possibilidade de interligarmos os diversos tipos.

Isso realmente ocorre, já que todos os circuitos integrados da família TTL e também das subfamílias são alimentados com 5 V.

Devemos observar, e com muito cuidado, que as correntes que circulam nas entradas e saídas dos componentes das diversas subfamílias são completamente diferentes, logo, quando passamos de uma para outra, tentanto interligar os seus componentes, as regras de *Fan-In* e *Fan-Out* mudam completamente.

Na verdade, não podemos falar de *Fan-in* e *Fan-out* quando interligamos circuitos de famílias diferentes.

O que existe é a possibilidade de elaborar uma tabela, a partir das características dos componentes, em que a quantidade máxima de entradas de determinada subfamília possa ser ligada na saída de outra subfamília.

Esta tabela é dada a seguir:

Saída

	74L	74	74LS	74H	74S
74L	20	40	40	50	100
74LS	2,5	10	51	2,5	12,5

Entrada

74	10	20	20	25	50
74H	2	8	4	10	10
74S	2	8	4	10	10

Observamos por esta tabela que uma saída 74 *(Standard)* pode excitar convenientemente 10 entradas 74LS *(Low Power Schottky)*.

Na figura 16 mostramos como isso pode ser feito.

3.7 - *Open Collector* e *Totem-Pole*

Os circuitos comuns TTL estudados até agora e que têm a configuração mostrada na **figura 14** são denominados *Totem Pole*.

Nestes circuitos temos uma configuração em que um ou outro transistor conduz a corrente, conforme o nível estabelecido na saída seja 0 ou 1.

Este tipo de circuito apresenta um inconveniente se ligarmos duas portas em paralelo, conforme a **figura 17**.

Se uma das portas tiver sua saída indo ao nível alto (1) ao mesmo tempo que a outra vai ao nível baixo (0),um curto-circuito é estabelecido na saída e pode causar sua queima.

Isso significa que os circuitos integrados TTL com esta configuração nunca podem ter suas saídas interligadas da forma indicada.

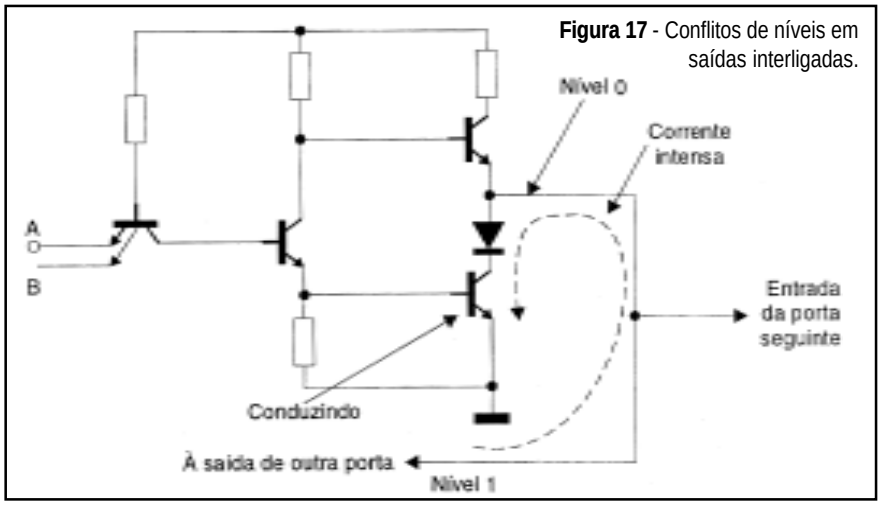


Figura 17 - Conflitos de níveis em saídas interligadas.

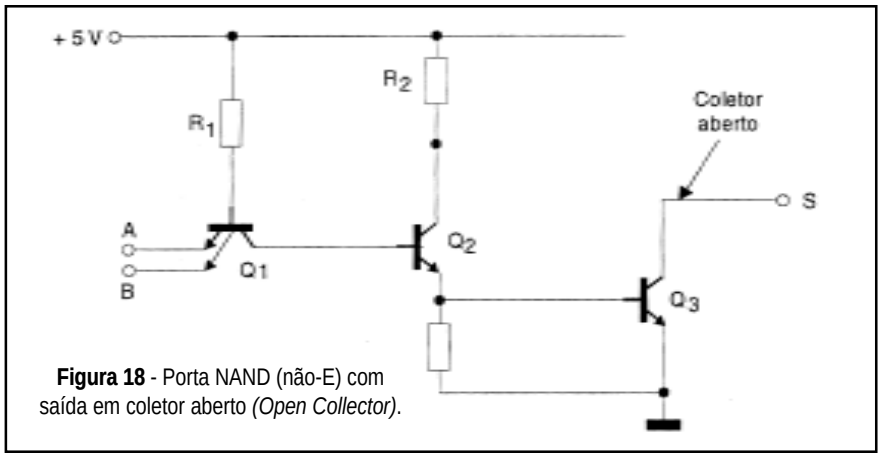


Figura 18 - Porta NAND (não-E) com saída em coletor aberto *(Open Collector)*.

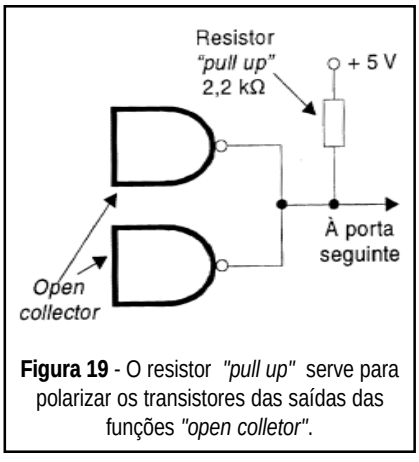


Figura 19 - O resistor *"pull up"* serve para polarizar os transistores das saídas das funções *"open colletor"*.

No entanto, existe uma possibilidade de elaborar circuitos em que as saídas de portas sejam interligadas. Isso é conseguido com a configuração denominada *Open Collector* mostrada na figura 18.

Os circuitos integrados TTL que possuem esta configuração são indicados como *"open collector"* e quando são usados, exigem a ligação de um resistor externo denominado *"pull up"* normalmente de 2000 Ω ou próximo disso.

Como o nome em inglês diz, o transistor interno está com o "coletor aberto" *(open collector)* e para funcionar precisa de um resistor de polarização.

A vantagem desta configuração está na possibilidade de interligarmos portas diferentes num mesmo ponto, **figura 19**.

A desvantagem está na redução da velocidade de operação do circuito que se torna mais lento com a presença do resistor, pois ele tem uma certa impedância que afeta o desempenho do circuito.

3.8 - *Tri-State*

Tri-state significa terceiro estado e é uma configuração que também pode ser encontrada em alguns circuitos integrados TTL, principalmente usados em Informática. Na **figura 20** temos um circuito típico de uma porta NAND *tri-state* que vai servir como exemplo. Podem existir aplicações em que duas portas tenham suas saídas ligadas num mesmo circuito, **figura 21**.

Uma porta está associada a um primeiro circuito e a outra porta a um segundo circuito. Quando um circuito envia seus sinais para a porta, o outro deve ficar em espera.

Ora, se o circuito que está em espera ficar no nível 0 ou no nível 1, estes níveis serão interpretados pela porta seguinte como informação e isso não deve ocorrer.

O que deve ocorrer é que quando uma porta estiver enviando seus sinais, a outra porta deve estar numa situação em que na sua saída não tenhamos nem 0 e nem 1, ou seja, ela deve ficar num estado de circuito desligado, circuito aberto ou terceiro estado. Isso é conseguido através de uma entrada de controle denominada "habilitação" em inglês *"enable"* abreviada por EN.

Assim, quando EN está no nível 0, no circuito da **figura 20**, o transistor não conduz e nada acontece no circuito que funciona normalmente.

No entanto, se EN for levada ao nível 1, o transistor satura, levando ao corte, ou seja, os dois passam a se comportar como circuitos abertos, independentemente dos sinais de entrada. Na saída Y teremos então um estado de alta impedância.

Podemos então concluir que a função *tri-state* apresenta três estados possíveis na sua saída:

Nível lógico 0
Nível lógico 1
Alta Impedância

As funções *tri-state* são muito usadas nos circuitos de computadores, nos denominados barramentos de dados ou "data bus", onde diversos circuitos devem aplicar seus sinais ao mesmo ponto ou devem compartilhar a mesma linha de transferência desses dados. O circuito que está funcionando deve estar habilitado e os que não estão funcionando, para que suas saídas não influenciem nos demais, devem ser levados sempre ao terceiro estado.

Na **figura 22** temos um exemplo de aplicação em que são usados circuitos *tri-state*. Uma unidade de processamento de um computador envia e recebe dados para/de diversos periféricos usando uma única linha *(bus)*. Todos os circuitos ligados a estas linhas devem ter saídas do tipo *tri-state*.

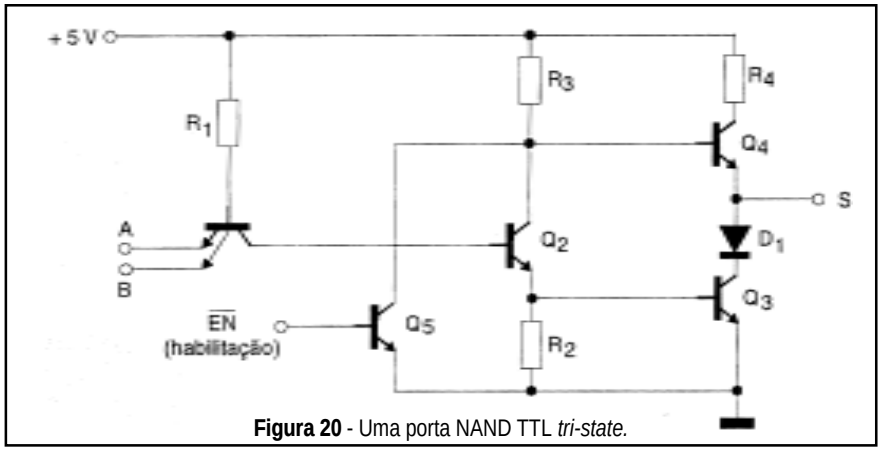


Figura 20 - Uma porta NAND TTL *tri-state.*

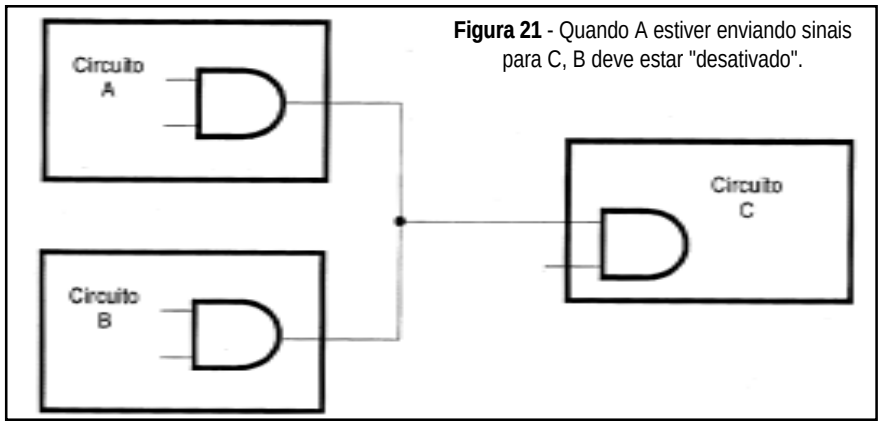


Figura 21 - Quando A estiver enviando sinais para C, B deve estar "desativado".

QUESTIONÁRIO

1. Quais são as duas principais famílias de circuitos lógicos digitais obtidas na forma de circuitos integrados?
   a) CMOS e TTL
   b) *Schottky* e LS
   c) AO e *Solid State*
   d) FET e Bipolar

2. Qual é a tensão de alimentação dos circuitos integrados da família TTL *Standard?*
   a) 3 a 15 V b) 1,5 V
   c) 5 V d) 12 V

3. Circuitos integrados que contenham grande quantidade de funções, mais de 1 000, usados principalmente nos modernos computadores são denominados:
   a) SSI b) MSI
   c) LSI d) VLSI

4. Um circuito integrado tem uma capacidade maior de corrente na sua saída quando:
   a) No nível 1 b) No nível 0
   c) As capacidades são iguais nos dois níveis
   d) A capacidade depende da função

5. A família TTL de alta velocidade tem seus componentes com a sigla:
   a) 74L b) 74H
   c) 74S d) 74LS

6. Para que tipos de configuração de saída não podemos ligar duas portas juntas?
   a) Todas
   b) *Totem pole*
   c) *Open Collector*
   d) Nenhuma delas

7. Que estado encontramos numa saída de uma função TTL *Tri-state* quando a entrada de habilitação não está ativada?
   a) Nível 0
   b) Nível 1
   c) Nível 0 ou 1
   d) Alta impedância

Respostas da lição nº 2
1 - b)
2 - b)
3 - a)
4 - a)
5 - a)
6 - d)
7 - c)

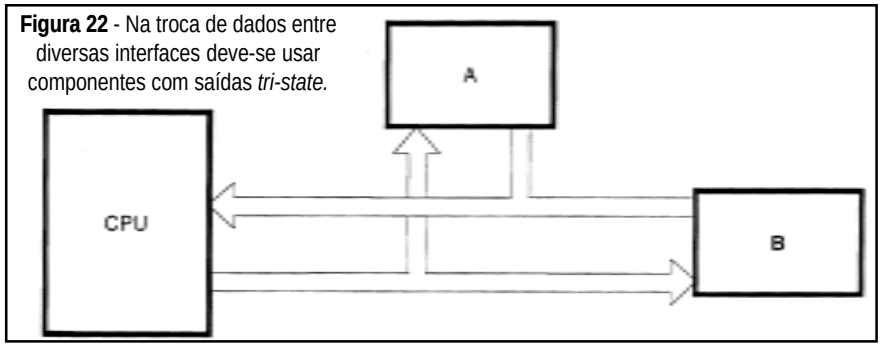


Figura 22 - Na troca de dados entre diversas interfaces deve-se usar componentes com saídas *tri-state.*

LIÇÃO 4

FAMÍLIAS DE CIRCUITOS INTEGRADOS CMOS

Na lição anterior mostramos aos leitores que os circuitos integrados digitais são organizados em famílias de modo a manter uma compatibilidade de características que permita sua interligação direta sem a necessidade de qualquer componente adicional. Vimos na ocasião que as famílias contam com dezenas ou mesmo centenas de funções que atuam como blocos ou tijolos a partir dos quais podemos "construir" qualquer circuito eletrônico digital, por mais complexo que seja. Na verdade, os próprios blocos tendem a ser cada vez mais completos, com a disponibilidade de circuitos integrados que contenham milhares ou mesmo dezenas de milhares de funções já interligadas de modo a exercer uma tarefa que seja muito utilizada. É o caso dos circuitos integrados VLSI de apoio encontrados nos computadores, em que milhares de funções lógicas já estão interligadas para exercer dezenas ou centenas de funções comuns nestes equipamentos.

Na lição anterior estudamos a família TTL e suas subfamílias muito comuns na maioria dos equipamentos eletrônicos, analisando as principais funções disponíveis e também suas características elétricas.

No entanto, existem outras famílias e uma muito utilizada é justamente a que vamos estudar nesta lição: a família CMOS. Se bem que as duas famílias CMOS e TTL tenham características diferentes, não são incompatíveis. Na verdade, conforme veremos, elas podem ser interligadas em determinadas condições que o leitor deve conhecer e que também serão abordadas nesta lição. Como estas duas famílias correspondem praticamente a tudo que pode ser feito em matéria de circuitos digitais, o seu conhecimento dará as bases necessárias ao trabalho com este tipo de componente.

OS CIRCUITOS INTEGRADOS CMOS

CMOS significa *Complementary Metal-Oxide Semiconductor* e se refere a um tipo de tecnologia que utiliza transistores de efeito de campo ou *Field Effect Transistor* (FET) em lugar dos transistores bipolares comuns (como nos circuitos TTL) na elaboração dos circuitos integrados digitais.

Existem vantagens e desvantagens no uso de transistores de efeito de campo, mas os fabricantes conseguem pouco a pouco eliminar as diferenças existentes entre as duas famílias com o desenvolvimento de tecnologias de fabricação, aumentando ainda a sua velocidade e reduzindo seu consumo. De uma forma geral, podemos dizer que existem aplicações em que é mais vantajoso usar um tipo e aplicações em que o outro tipo é melhor.

Os transistores de efeito de campo usados nos circuitos integrados CMOS ou MOSFETs têm a estrutura básica mostrada na **figura 1** onde também aparece seu símbolo.

Conforme podemos ver, o eletrodo de controle é a comporta ou *gate* (g) onde se aplica o sinal que deve ser amplificado ou usado para chavear o circuito. O transistor é polarizado de modo a haver uma tensão entre a fonte ou *source* (s) e o dreno ou *drain* (d). Fazendo uma analogia com o transistor bipolar, podemos dizer que a comporta do MOSFET equivale à base do transistor bipolar, enquanto que o dreno equivale ao coletor e a fonte ao emissor, figura 4.2.

Observe que entre o eletrodo de comporta, que consiste numa placa de alumínio e a parte que forma o substrato ou canal por onde passa a corrente, não existe contato elétrico e nem junção, mas sim uma finíssima camada de óxido de alumínio ou óxi-

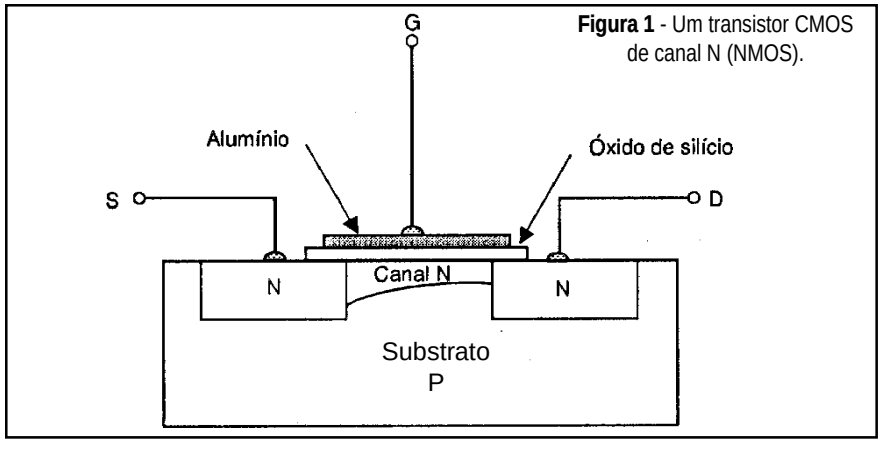


Figura 1 - Um transistor CMOS de canal N (NMOS).

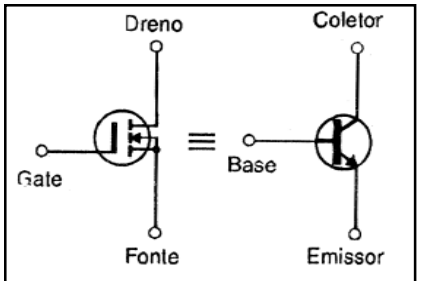


Figura 2 - Equivalência de funções dos eletrodos para transistores MOS e bipolares.

do metálico, que dá nome ao dispositivo (*metal-oxide*).

A polaridade do material semicondutor usado no canal, que é a parte do transistor por onde circula a corrente controlada, determina seu tipo e também a polaridade da tensão que a controla.

Assim, encontramos na prática transistores de efeito de campo tipo MOS de canal N e transistores de efeito de campo tipo MOS de canal P.

Na verdade, os próprios transistores MOS podem ainda ser divididos em dois tipos: enriquecimento e empobrecimento que levam a dois tipos de representação. Para nosso curso é mais importante lembrar que existem transistores MOS tipo P e tipo N. Na **figura 3** temos os símbolos adotados para representar os dois tipos de transistores.

Podemos dizer, de maneira geral, que estes transistores são equivalentes aos tipos NPN e PNP bipolares.

A corrente que circula entre a fonte e o dreno pode ser controlada pela tensão aplicada à comporta. Isso significa que, diferentemente dos transistores bipolares em que a corrente de coletor depende da corrente de base, no transistor de efeito de campo, a corrente do dreno depende da tensão de comporta.

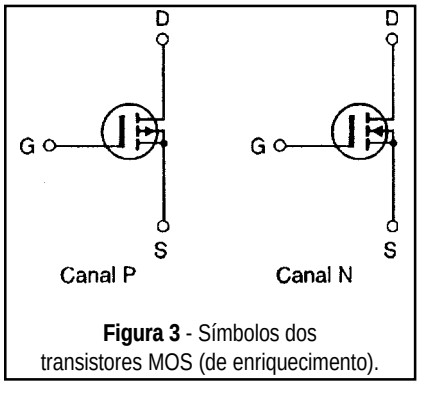


Figura 3 - Símbolos dos transistores MOS (de enriquecimento).

Assim, no tipo P uma tensão positiva de comporta aumenta sua condução, ou seja, faz com que ele sature e no tipo N, uma tensão negativa de comporta é que o leva à saturação.

Mais uma vez fazendo uma comparação com os tipos bipolares, podemos dizer então que enquanto os transistores bipolares são típicos amplificadores de corrente, os FETs ou transistores de efeito de campo MOS são típicos amplificadores de tensão.

Esta diferença leva o transistor de efeito de campo MOS a apresentar características muito interessantes para aplicações em Eletrônica Digital ou Analógica.

Uma delas está no fato de que a impedância de entrada do circuito é extremamente elevada, o que significa que precisamos praticamente só de tensão para controlar os dispositivos CMOS.

Assim, é preciso uma potência extremamente baixa para o sinal que vai excitar a entrada de um circuito integrado CMOS, já que praticamente nenhuma corrente circula por este elemento.

A outra está no fato de que, diferentemente dos transistores bipolares que só começam a conduzir quando uma tensão da ordem de 0,6 V vence a barreira de potencial de sua junção base-emissor, os FETs não têm esta descontinuidade de características, o que os torna muito mais lineares em qualquer aplicação que envolva amplificação de sinais.

Na **figura 4** temos as curvas características de um MOSFET de canal N.

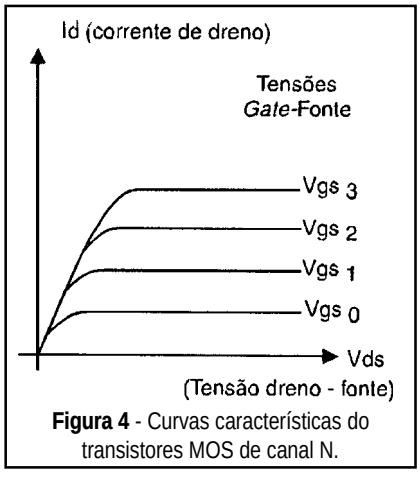


Figura 4 - Curvas características do transistores MOS de canal N.

APLICAÇÕES DIGITAIS

Da mesma forma que podemos elaborar funções lógicas básicas usando transistores bipolares comuns, também podemos fazer o mesmo com base nos transistores de efeito de campo MOS. A tecnologia CMOS (*Complementary MOS*) permite que os dispositivos tenham características excelentes para aplicações digitais.

CMOS significa que em cada função temos configurações em que transistores de canal N e de canal P são usados ao mesmo tempo, ou seja, usamos pares complementares, conforme diagrama do inversor lógico mostrado na **figura 5**. Conforme explicamos no item anterior, a polaridade da tensão que controla a corrente principal nos transistores de efeito de campo MOS depende justamente do tipo de material usado no canal, que pode ser do tipo P ou do tipo N.

Assim, se levarmos em conta que nos circuitos digitais temos dois níveis de sinal possíveis, podemos perceber que dependendo do nível deste sinal aplicado à comporta dos dois transistores ao mesmo tempo, quando um deles estiver polarizado no sentido de conduzir plenamente a corrente (saturado), o outro estará obrigatoriamente polarizado no sentido de cortar esta corrente (corte).

No circuito indicado, quando a entrada A estiver no nível baixo (0) o transistor Q_2 conduz, enquanto Q_1 permanece no corte. Isso significa que Vdd, que é a tensão de alimentação positiva, é colocada na saída, o que corresponde ao nível alto ou 1.

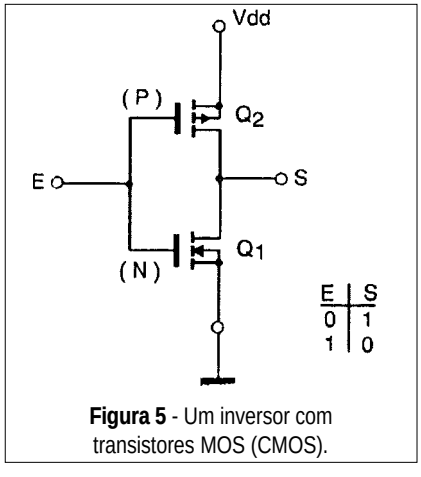


Figura 5 - Um inversor com transistores MOS (CMOS).

Por outro lado, quando na entrada aplicamos o nível alto, que corresponde ao Vdd (tensão de alimentação), é o transistor Q_1 que conduz e com isso o nível baixo ou 0 V é que será colocado na saída.

Conforme sabemos, estas características correspondem justamente a função inversora.

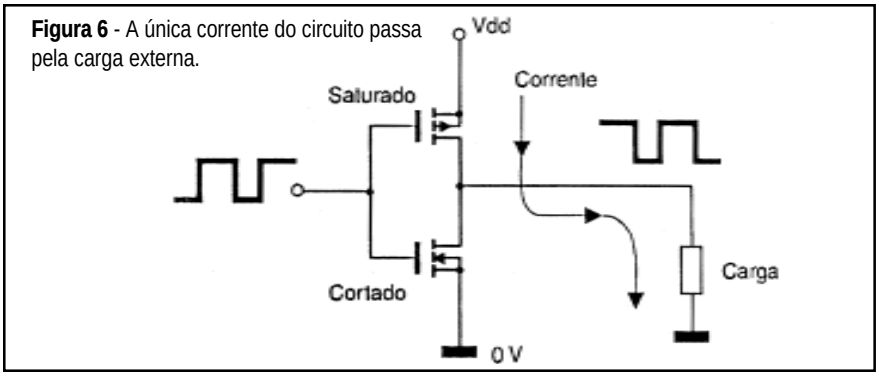


Figura 6 - A única corrente do circuito passa pela carga externa.

CONSUMO E VELOCIDADE

Analisando o circuito inversor tomado como base para nossas explicações, vemos que ele apresenta duas características importantes.

A primeira é que sempre um dos transistores estará cortado, qualquer que seja o sinal de entrada (alto ou baixo) logo, praticamente não circula corrente alguma entre o Vdd e o ponto de terra (0 V). A única corrente que irá circular será eventualmente a de um circuito externo excitado pela saída, **figura 6**.

Isso significa um consumo extremamente baixo para este par de transistores em condições normais, já que na entrada a impedância é elevadíssima e praticamente nenhuma corrente circula. Este consumo é da ordem de apenas 10 nW (nW = nanowatt = 0,000 000 001 watt).

É fácil perceber que se integrarmos 1 milhão de funções destas num circuito integrado, ele irá consumir apenas 1 mW! Na prática temos fatores que tornam maior este consumo, como por exemplo, eventuais fugas, a necessidade de um ou outro componente especial de excitação que exija maior corrente, etc.

Mas, ao lado das boas características, ele também tem seus problemas: um deles está no fato de que o eletrodo de controle (comporta) que é uma placa de metal fixada no material semicondutor e isolada por meio de uma camada de óxido, funciona como a armadura ou placa de um capacitor, verifique a **figura 7**.

Isso significa que, ao aplicarmos um sinal de controle a uma função deste tipo, a tensão não sobe imediatamente até o valor desejado, mas precisa de um certo tempo necessário para carregar o "capacitor" representado pelo eletrodo de comporta. Se bem que o eletrodo tenha dimensões extremamente pequenas, se levarmos em conta as impedâncias envolvidas no processo de carga e também a própria disponibilidade de corrente dos circuitos excitadores, o tempo envolvido no processo não é desprezível e um certo atraso na propagação do sinal ocorre.

O atraso nada mais é do que a diferença de tempo entre o instante em que aplicamos o sinal na entrada e o instante em que obtemos um sinal na saída.

Nos circuitos integrados CMOS típicos como os usados nas aplicações digitais, para um inversor como o do exemplo, este atraso é da ordem de 3 nanossegundos (3 ns).

Isso pode parecer pouco nas aplicações comuns, mas se um sinal tiver de passar por centenas de portas antes de chegar a um certo ponto em que ele seja necessário, e a soma dos atrasos não for prevista poderá haver diversos problemas de funcionamento.

Veja, entretanto, que a carga de um capacitor num circuito de tempo, como o na **figura 8** até um determinado nível de tensão depende também da tensão de alimentação.

Assim, com mais tensão, a carga é mais rápida e isso nos leva a uma característica muito importante dos circuitos CMOS digitais que deve ser levada em conta em qualquer aplicação: **com maior tensão de alimentação, os circuitos integrados CMOS são mais rápidos.**

Assim, enquanto que nos manuais de circuitos integrados TTL encontramos uma velocidade máxima única de operação para cada tipo (mesmo porque sua tensão de alimentação é fixa de 5 V), nos manuais CMOS encontramos as velocidades associadas às tensões de alimentação (já que os circuitos integrados CMOS podem ser alimentados por uma ampla faixa de tensões).

Um exemplo disso pode ser observado nas características de um circuito integrado CMOS formado por seis inversores *(hex inverter)* onde temos as seguintes frequências máximas de operação:

4049 - Seis inversores
Frequência máxima de operação:
Com Vdd = 5 V - 1,66 MHz (tip)
Vdd = 10 V - 4,00 MHz (tip)
Vdd = 15 V - 5,00 MHz (tip)

Veja então que o circuito é muito mais rápido quando o alimentamos com uma tensão de 15 V do que quando o alimentamos com uma tensão

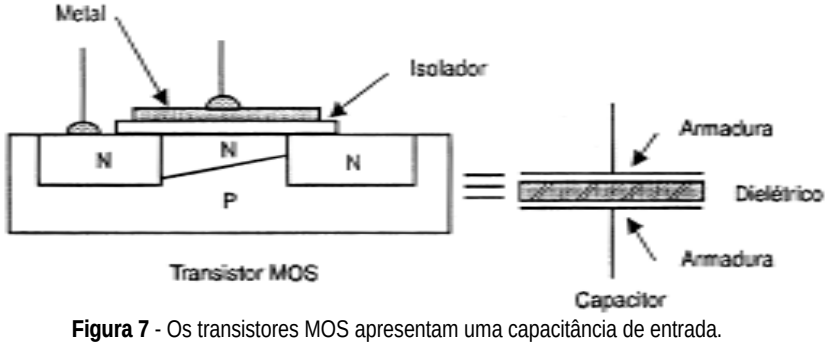


Figura 7 - Os transistores MOS apresentam uma capacitância de entrada.

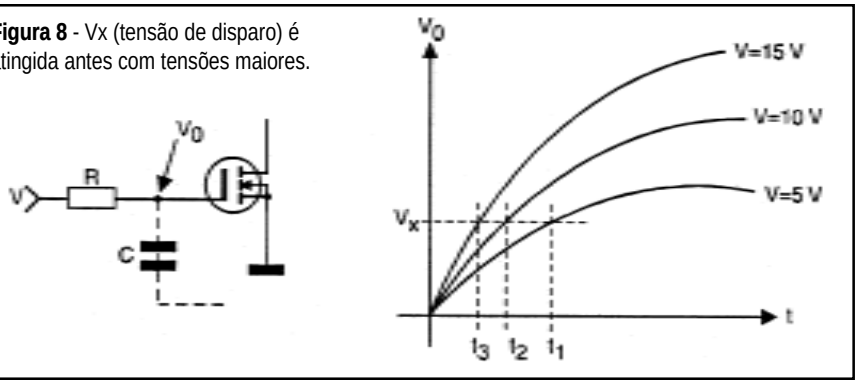


Figura 8 - Vx (tensão de disparo) é atingida antes com tensões maiores.

de apenas 5 V. Este fato é muito importante, por exemplo, na elaboração de um oscilador com circuito integrado CMOS que opere no seu limite de velocidade.

SENSIBILIDADE AO MANUSEIO

O fato de que existe uma finíssima camada de óxido isolando a comporta do substrato e esta camada é extremamente sensível a descargas elétricas torna os dispositivos que usam transistores MOS muito delicados.

De fato, a própria carga elétrica acumulada em ferramentas ou em nosso corpo quando caminhamos num tapete num dia seco ou ainda atritamos objetos em nossa roupa pode ser suficiente para danificar de modo irreversível dispositivos CMOS. Para que o leitor tenha uma idéia, caminhando num carpete num dia seco, seu corpo pode acumular uma carga estática que atinge potenciais de até 10 000 V.

Se você tocar numa torneira, a descarga de seu corpo neste percurso de terra pode lhe causar um forte choque.

Se, da mesma forma, você tocar num terminal de um dispositivo CMOS, a carga do seu corpo que escoa por este dispositivo pode facilmente destruir a finíssima camada de óxido que separa a comporta do substrato e o componente estará inutilizado.

Em outras palavras, os dispositivos que usam transistores CMOS são extremamente sensíveis a descargas estáticas, **figura 9**.

Assim, a primeira preocupação no uso e manuseio destes componentes

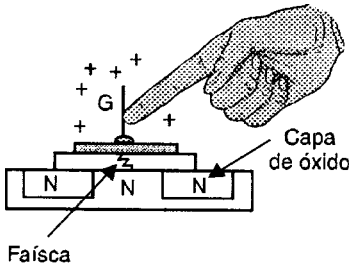


Figura 9 - Descargas estáticas destroem os transistores MOS.

é evitar de qualquer modo que apareçam tensões perigosas capazes de causar danos entre os terminais dos componentes.

Para os transistores MOS existe a possibilidade de dotá-los de um pequeno anel de metal que curto-circuita seus terminais, conforme **figura 10**, e que somente é retirado depois que o componente é soldado na placa de circuito impresso.

Existem diversas formas de fazer transporte de circuitos integrados sem o perigo de que cargas estáticas acumuladas em objetos possam lhes causar danos.

Uma delas consiste no uso de uma esponja condutora onde os terminais dos circuitos integrados são enfiados e assim mantidos em curto, **figura 11**.

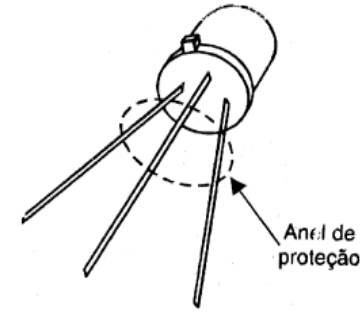


Figura 10 - Transistores MOS podem ser protegidos por um anel de metal que coloca em curtos seus terminais.

Os circuitos integrados CMOS devem ser mantidos nestas esponjas até o momento de serem usados, sob pena de que algum toque acidental com o dedo carregado de estática provoque danos.

Outra possibilidade consiste em transportar os circuitos integrados CMOS em embalagens de plástico anti-estático **figura 12**.

De qualquer forma, a regra geral é: **NUNCA toque com os dedos nos terminais de componentes CMOS sejam eles circuitos integrados ou transistores.**

Num laboratório onde são efetuados trabalhos com circuitos integrados CMOS é importante observar precauções especiais para que em nenhum ponto ocorram acúmulos de cargas estáticas. As bancadas de trabalhos com computadores devem ter partes metálicas aterradas e os próprios técnicos devem usar recursos que permitam descarregar cargas do seu corpo. Em empresas de trabalhos com circuitos CMOS é comum os técnicos usarem pulseiras metálicas, sendo estas pulseiras ligadas a um fio terra.

Para o técnico comum é apenas necessário lembrar-se de que não deve tocar nos terminais dos componentes e com isso já haverá uma boa garantia da integridade dos circuitos.

Um outro ponto importante é nunca deixar nenhuma entrada de um circuito integrado CMOS desligada.

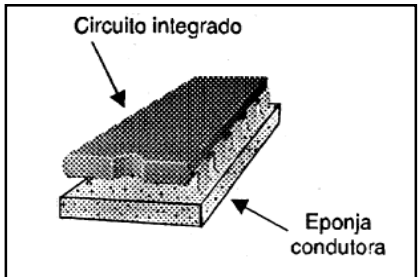


Figura 11 - Uma esponja condutora é usada no transporte de CIs sensíveis.

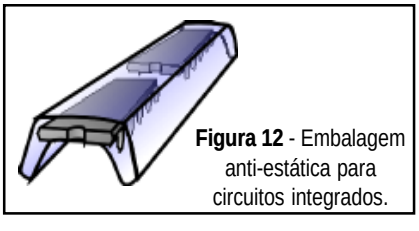


Figura 12 - Embalagem anti-estática para circuitos integrados.

A sensibilidade destas entradas é suficientemente alta para que tensões induzidas no próprio circuito sejam captadas, levando os dois transistores a um estado intermediário entre o corte e a saturação ou ainda fazendo com que entrem em oscilação na frequência do sinal captado. Isso, além de elevar o consumo do circuito integrado, pode causar instabilidades que afetem o funcionamento geral do circuito.

Uma regra prática consiste em levar as entradas das funções não usadas num integrado a níveis definidos de tensão, ou seja, ligar ao Vdd ou ainda ao ponto de 0 V.

AS CONFIGURAÇÕES CMOS

Na **figura 13** temos a configuração usada para uma porta NOR de 2 entradas CMOS em que temos quatro transistores.

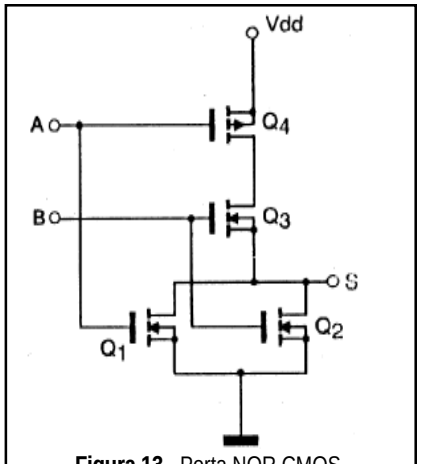


Figura 13 - Porta NOR CMOS.

Observe a simplicidade dos circuitos CMOS quando comparados a funções equivalentes TTL. Com os circuitos CMOS precisamos apenas de transistores para obter a função desejada, enquanto que na equivalente TTL precisamos de transistores e muitos resistores e em alguns casos até de diodos.

Na **figura 14** temos a configuração usada para uma porta NAND de duas entradas CMOS onde também usamos apenas 4 transistores.

Neste circuito, quando as entradas ou uma delas estiver no nível baixo (0) um ou os dois transistores de canal P estarão em condução e a saída ficará no nível alto.

Quando as duas entradas estiverem no nível 1, entretanto, os dois transistores de canal N irão conduzir ao mesmo tempo, levando a saída para o nível baixo.

Para as outras funções lógicas temos configurações do mesmo tipo, mudando apenas a disposição e a quantidade de transistores usados. Tomando estas duas funções como exemplo, achamos que o leitor terá uma idéia de como elas são feitas e como funcionam.

ESPECIFICAÇÕES

A principal família de circuitos integrados CMOS é a 4000, onde todos os componentes são designados por números como 4001, 4011, 4017, 4096, etc.

Os circuitos integrados CMOS comuns funcionam com tensões de alimentação de 3 a 15 V. Lembramos que existem séries CMOS mais antigas com o sufixo A em que a tensão de alimentação fica na faixa de 3 a 12 V.

De qualquer forma, em caso de dúvida sobre qualquer característica de um circuito integrado CMOS que tenha algum sufixo que possa indicar variações nas especificações normais, é sempre bom consultar seu manual.

Da mesma forma que no caso dos circuitos integrados TTL, é preciso saber interpretar algumas das principais especificações que são:

a) Tensão de saída - no nível lógico baixo (0) a tensão de saída se aproxima de 0 V sendo no máximo de 0,01 V para os tipos comuns com alimentação na faixa de 5 a 10 V. No nível lógico alto, a tensão de saída é praticamente a tensão de alimentação Vdd ou no máximo 0,01 V menor.

b) Corrente de saída - diferentemente dos circuitos integrados TTL em que temos uma capacidade maior de drenar corrente na saída do que de fornecer, para os circuitos integrados CMOS a capacidade de drenar e de fornecer corrente de saída é praticamente a mesma.

Assim, para uma alimentação de 5 V as saídas podem fornecer (quando no nível alto) ou drenar (quando

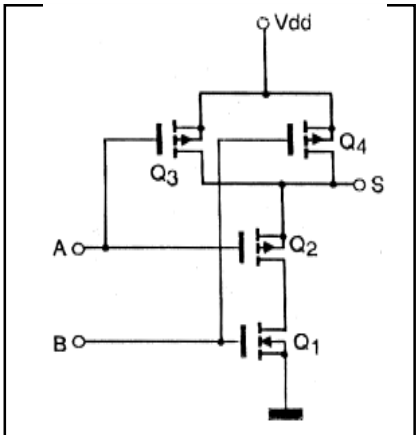


Figura 14 - Porta NAND CMOS.

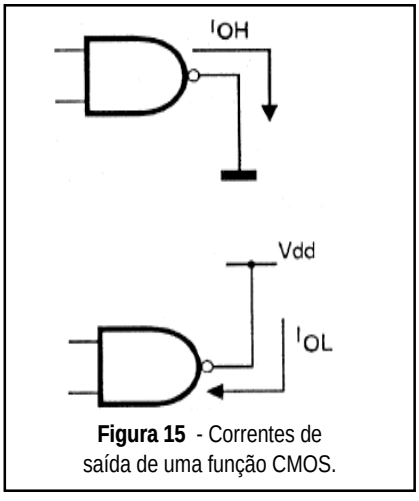


Figura 15 - Correntes de saída de uma função CMOS.

no nível baixo) uma corrente de até 1 mA e essa corrente sobe para 2,5 mA quando a alimentação é de 10 V.

Estas correntes, conforme a figura 4.15 são designadas por IOL e IOH nas folhas de especificações dos circuitos integrados CMOS.

c) Corrente de fuga na entrada - se bem que a comporta esteja isolada do circuito dreno-fonte, com uma resistência que teoricamente seria infinita, na prática pode ocorrer uma pequena fuga.

Esta, da ordem de 10 pA (1 picoampère = 0,000 000 000 001 ampère) para uma alimentação de 10 V deve ser considerada quando precisamos calcular a corrente de entrada de um circuito CMOS numa aplicação mais crítica.

d) Potência - os circuitos integrados CMOS consomem muito menos energia que os circuitos integrados TTL. Para os tipos comuns a corrente de alimentação Idd é normalmente da ordem de 1 nA tipicamente com um

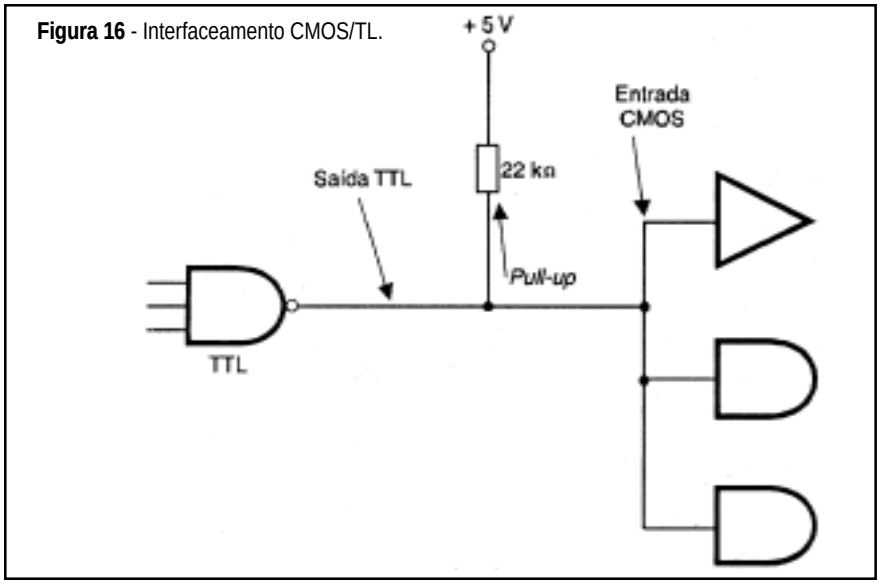


Figura 16 - Interfaceamento CMOS/TL.

máximo de 0,05 µA para alimentação de 5 V, o que corresponde a uma dissipação de 5 nW em média para alimentação de 5 V e 10 nW para alimentação de 10 V.

e) Velocidade - os tipos comuns CMOS são muito mais lentos que os TTL, mas famílias especiais estão aparecendo com velocidades cada vez maiores e em muitos casos estas se aproximam dos mais rápidos TTLs.

As frequências máximas, conforme já explicamos, dependem das tensões de alimentação e das funções, já que maior número de componentes para atravessar significa um atraso maior do sinal. Assim, nos manuais encontramos a especificação de velocidade dada tanto em termos de frequência quanto em termos de atraso do sinal. Para o caso do atraso do sinal, observamos que ele pode estar especificado para uma transição do nível alto para o nível baixo ou viceversa e em alguns circuitos ou tensões de alimentação podem ocorrer diferenças.

INTERFACEANDO

Conforme explicamos, mesmo tendo uma faixa de tensões ampla e características diferentes dos circuitos integrados TTL, existe a possibilidade de interfacear circuitos dos dois tipos. Há duas possibilidades de interfaceamento entre circuitos digitais TTL e circuitos digitais CMOS.

a) A saída TTL deve excitar a entrada CMOS.

Se os dois circuitos operarem com uma tensão de alimentação de 5 V não há problema e a interligação pode ser direta.

Como as entradas CMOS têm uma impedância muito alta (não exigindo praticamente corrente alguma) da saída TTL, não existe perigo do circuito CMOS "carregar" a saída TTL. No entanto, existe um problema a ser considerado: as entradas CMOS só reconhecem como nível 1 uma tensão de pelo menos 3,5 V, enquanto que no nível alto, a tensão mínima que o TTL pode fornecer nestas condições é de 3,3 V.

Isso significa que é preciso assegurar que a entrada CMOS reconheça o nível alto TTL, o que é conseguido com a adição de um resistor externo de *pull-up*, observe a figura 4.16.

Este resistor de 22 kΩ é ligado ao positivo da alimentação de 5 V.

Se o circuito CMOS a ser excitado por um TTL for alimentado com tensão maior que 5 V, por exemplo 12 V, deve ser usado um circuito intermediário de casamento de características.

Este circuito intermediário deve manter o sinal, ou seja, deve ser simplesmente um *buffer* não inversor, como por exemplo, o de coletor aberto 7406 ou 7407 com um resistor de *pull-up* externo, conforme a figura 4.17. O valor deste resistor dependerá da tensão de alimentação.

b) CMOS excitando uma entrada TTL

Neste caso, devemos considerar que uma saída CMOS no nível baixo pode drenar uma corrente de aproximadamente 0,5 mA e no estado alto, a mesma intensidade.

No entanto, uma entrada TTL fornece uma corrente de 1,6 mA no nível baixo, o que não pode ser absorvido pela saída CMOS. Isso significa que entre as duas devemos intercalar um *buffer* CMOS, como por exemplo, os 4049 e 4050 que permitem a excitação de até duas entradas TTL a partir de uma saída CMOS.

FONTE DE ALIMENTAÇÃO

Os circuitos integrados TTL precisam de uma tensão contínua na faixa de 4,5 a 5,5 V para poderem funcionar e são bastante sensíveis a alterações que saiam desta faixa.

Já os circuitos CMOS são muito menos sensíveis e podem operar numa faixa mais larga de tensões, conforme vimos, o que facilita bastante o projeto das fontes e até permite a alimentação direta a partir de pilhas ou baterias.

Veja que o fato dos circuitos integrados CMOS funcionarem perfeitamente com tensões como 3, 6 , 9 e 12 V, que são facilmente obtidas de pilhas e bateria, os torna ideais para aplicações em que este tipo de fonte é usada.

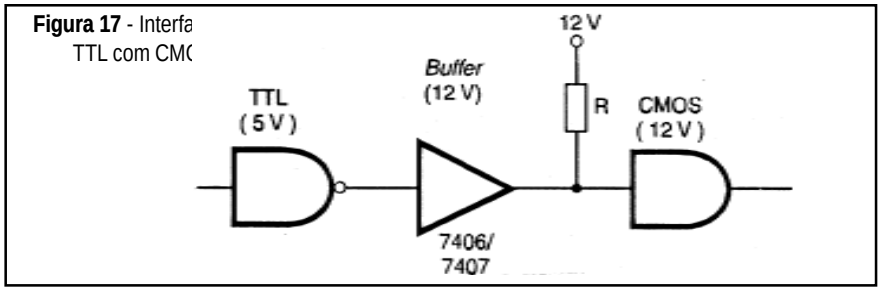


Figura 17 - Interfa
TTL com CMC

QUESTIONÁRIO

1. O elemento de controle do sinal de um transistor de efeito de campo é denominado:
a) base
b) dreno
c) comporta
d) canal

2. Qual o tipo de material que separa o elemento de controle de um MOSFET do canal?
a) Uma junção PN
b) Um substrato condutor
c) Uma camada de material isolante
d) Um terminal de cobre

3. Num inversor CMOS encontramos na etapa de saída:
a) dois FETs de canal N
b) dois FETs de canal P
c) Um par de transistores bipolares
d) Um FET de canal N e outro de canal P

4. A faixa de tensões de alimentação dos circuitos integrados CMOS tem valores entre:
a) 4,5 e 5,5 V
b) 3 e 15 V
c) 0 e 6 V
d) 5 e 18 V

5. O perigo maior do manuseio dos circuitos integrados CMOS se deve a:
a) descargas estáticas
b) aquecimento da pastilha semicondutora
c) perigo de quebra dos terminais
d) contaminação radioativa

6. O que devemos fazer com as entradas não usadas de um circuito integrados CMOS.
a)cortá-las
b)aterrá-las
c)ligá-las a um nível lógico apropriado
d) ligar a um resistor de 100 kΩ

Respostas da Lição nº 2:
1-b 2-b 3-a 4-a 5-a 6-d 7-c
Respostas da Lição nº3:
1-a 2-c 3-d 4-b 5-d 6-c 7-d
Respostas desta edição:
1-c 2-c 3-d 4-b 5-a 6-c

LIÇÃO 5

COMBINANDO FUNÇÕES LÓGICAS

Nas duas lições anteriores estudamos as famílias lógicas CMOS e TTL, analisando suas características elétricas principais e a maneira como os componentes são fabricados através de alguns circuitos típicos.

Nesta lição continuaremos a estudar as funções lógicas, agora de uma forma mais completa. Analisaremos o que ocorre quando juntamos diversas funções lógicas, prevendo o que acontece com suas saídas. Os circuitos complexos, como os usados nos computadores, por exemplo, se aproveitam das operações complicadas que muitas portas lógicas podem realizar em conjunto. Assim, é de fundamental importância para nosso estudo saber analisar estas funções.

5.1 - As tabelas verdade

Os diversos sinais de entrada aplicados a uma função lógica, com todas as suas combinações possíveis, e a saída correspondente podem ser colocados numa tabela.

Nas colunas de entradas colocamos todas as combinações possíveis de níveis lógicos que as entradas podem assumir. Na coluna correspondente à saída colocamos os valores que esta saída assume em função dos níveis lógicos correspondentes na entrada.

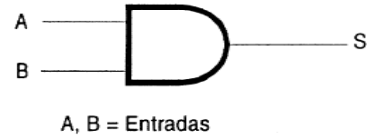


Figura 1 - Funções ou porta AND (E) de duas entradas.

Vimos, desta forma, que a tabela verdade para uma função AND de duas entradas, como a representada na **figura 1**, pode ser dada por:

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

Veja que nas colunas de entrada (A e B) para termos todas as combinações possíveis, fazemos o equivalente à numeração binária de 0 a 3, já que:

0	0	=	0
0	1	=	1
1	0	=	2
1	1	=	3

Para uma tabela verdade feita para uma porta AND de 3 entradas teremos:

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

Neste caso, as combinações de níveis lógicos na entrada correspondem à numeração binária de 0 a 7 já que:

0	0	0	=	0
0	0	1	=	1
0	1	0	=	2
0	1	1	=	3
1	0	0	=	4
1	0	1	=	5
1	1	0	=	6
1	1	1	=	7

O conhecimento da contagem binária facilita bastante a elaboração de tabelas verdades, quando todas as combinações possíveis de níveis lógicos em 2, 3 ou 4 entradas devam ser estudadas.

Assim, uma vez que o leitor conheça o comportamento das principais funções, sabendo o que ocorre na saída de cada uma quando temos determinadas entradas e sabendo elaborar tabelas verdades, fica fácil combinar funções e saber o que acontece em suas saídas.

5.2 - Lógica Combinacional

Vamos partir de um exemplo simples de lógica combinacional usando tabelas verdades para saber o que ocorre na sua saída, com o circuito da **figura 2**.

Este circuito faz uso de uma porta AND, um inversor e uma porta OR. O resultado desta configuração é uma função combinacional com três entradas e uma saída.

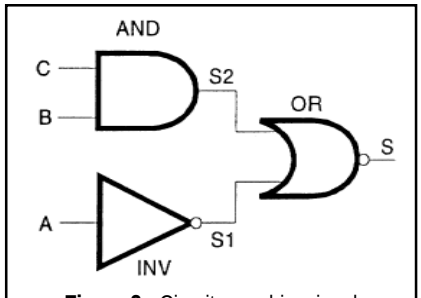


Figura 2 - Circuito combinacional simples com três entradas.

Para elaborar a tabela verdade para este circuito e assim determinarmos todas as saídas possíveis em função das entradas, devemos levar em conta que ele é formado por duas etapas.

Na primeira etapa temos a porta AND e o inversor, enquanto que na segunda etapa temos a porta OR. Isso significa que as saídas dos circuitos da primeira etapa, que chamaremos de S_1 e S_2 são a entrada da segunda etapa.

Temos então de levar em conta estas saídas na elaboração da tabela verdade que terá no seu topo as seguintes variáveis:

A	B	C	S_1	S_2	S

A,B e C são as entradas dos circuitos. S_1 e S_2 são pontos intermediários do circuito que precisam ser analisados para a obtenção de S, que é a saída final do circuito.

Começamos por colocar em A, B e C todas as suas condições possíveis, ou todas as combinações de níveis lógicos que podem ser aplicadas ao circuito:

A	B	C	S_1	S_2	S
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

O passo seguinte é colocar os valores possíveis de S_1, que corresponde à saída do inversor.

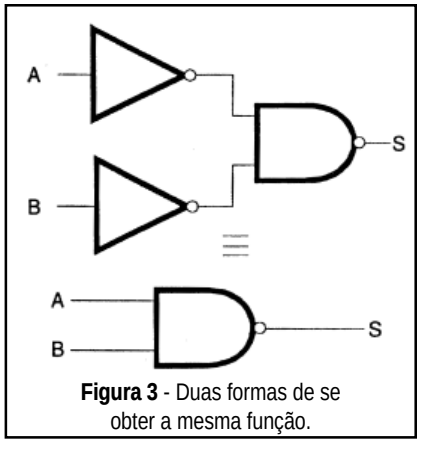


Figura 3 - Duas formas de se obter a mesma função.

Sabemos que a tabela verdade para o inversor é:

A	S
0	1
1	0

Ora, como em nosso caso A é a entrada do inversor e S_1 é sua saída, podemos partir para a determinação de toda a coluna S_1 simplesmente invertendo os valores de A, da seguinte forma:

A	B	C	S_1	S_2	S
0	0	0	1		
0	0	1	1		
0	1	0	1		
0	1	1	1		
1	0	0	0		
1	0	1	0		
1	1	0	0		
1	1	1	0		

Para encontrar os valores da coluna S_2 devemos observar que ela corresponde à tabela verdade da função AND onde as entradas são B e C e a saída é S_2.

B	C	S_2
0	0	0
0	1	0
1	0	0
1	1	1

Temos então:

A	B	C	S_1	S_2	S
0	0	0	1	0	
0	0	1	1	0	
0	1	0	1	0	
0	1	1	1	1	
1	0	0	0	0	
1	0	1	0	0	
1	1	0	0	0	
1	1	1	0	1	

Finalmente, levando em conta que S_1 e S_2 são entradas de uma porta OR de duas entradas cuja saída é S, podemos elaborar a coluna final de saídas (S)

S_1	S_2	S
0	0	0
0	1	1
1	0	1
1	1	1

Resultando na seguinte tabela:

A	B	C	S_1	S_2	S
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	1	1

Trata-se de uma função bastante interessante que pode ser definida como “a que fornece uma saída alta somente quando a entrada A estiver no nível baixo, não importando as demais entradas ou ainda quando as três entradas estiverem no nível alto”.

5.3 - Como Projetar Um Circuito Combinacional

O problema de saber o que acontece com a saída de um circuito formado por muitas funções lógicas quando suas entradas recebem diversas combinações de sinais não é o mais importante para o projetista de equipamentos digitais. Na verdade, muito mais importante que este procedimento é justamente fazer o contrário, ou seja, projetar um circuito que, em função de determinados sinais de entrada, forneça exatamente na saída o que se deseja.

O projeto de um circuito que tenha uma determinada função envolve um procedimento de síntese em algumas etapas.

Na primeira etapa deve ser definido o problema, estabelecendo-se exatamente qual a função a ser executada, ou seja, quais as entradas e quais as saídas.

Numa segunda etapa, coloca-se o problema numa tabela verdade ou ainda na forma de equações lógicas.

O procedimento que abordaremos neste curso será basicamente o da obtenção das funções a partir das tabelas verdade e das equações lógicas.

Finalmente, numa terceira etapa, obtemos o circuito que exercerá as funções desejadas.

Na terceira etapa, um ponto importante consiste na minimização do circuito, já que na maioria dos casos

pode-se implementar a mesma função de muitas formas diferentes como atesta o circuito simples apresentado na **figura 3**.

Veja que podemos ter o mesmo circuito com quantidades de portas diferentes, na prática devemos sempre levar este fato em conta. Não é apenas o número de portas que determinará a configuração final, mas sim, seu custo e a eventual utilização em outras partes do circuito.

Por exemplo, se o circuito já estiver usando dois inversores dos seis disponíveis num circuito integrado e a nossa função tiver uma solução um pouco maior, mas que use estes inversores, será interessante adotá-la para aproveitar os inversores ociosos.

A seguir daremos um exemplo de como obter os circuitos a partir de uma tabela verdade.

a) Passo 1 - Determinação das equações lógicas

Lembramos que para as funções estudadas temos as seguintes representações:

Função E (AND)

$Y=A.B$

Função Não E (NAND)

$Y=\overline{A.B}$

Função OU (OR)

$Y=A+B$

Função Não OU (NOR)

$Y=\overline{A+B}$

Função Não (NOT) ou inversor

$Y=\overline{A}$

Função ou exclusivo
(Exclusive OR)

$Y=A(+)B$

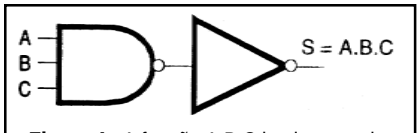


Figura 4 - A função A.B.C implementada.

Vamos tomar como exemplo a tabela verdade abaixo para determinar a função lógica correspondente:

A	B	C	Y	linha
0	0	0	0	1
1	0	0	1	2
0	1	0	1	3
1	1	0	0	4
0	0	1	1	5
1	0	1	0	6
0	1	1	0	7
1	1	1	1	8

Indicamos a linha na última coluna de modo a facilitar as explicações seguintes.

Observamos que temos saídas no nível 0 para as linhas 0, 3, 5 e 6, enquanto para as linhas 1, 2, 4 e 7 temos saídas 1.

Isso quer dizer que teremos a função OU para as linhas cuja saída é 1 que podem ser encaradas como operações OR com tabelas que teriam 1 na saída apenas nas linhas 1, 2, 4 e 7, conforme mostrado a seguir:

A B C Y		A B C S1		A B C S2		A B C S3		A B C S4
0 0 0 0		0 0 0 0		0 0 0 0		0 0 0 0		0 0 0 0
0 0 1 1		0 0 0 1		0 0 1 0		0 0 1 0		0 0 1 0
0 1 0 1		0 1 0 0		0 1 0 1		0 1 0 0		0 1 0 0
0 1 1 0	=	0 1 1 0	+	0 1 1 0	+	0 1 1 0	+	0 1 1 0
1 0 0 1		1 0 0 0		1 0 0 0		1 0 0 1		1 0 0 0
1 0 1 0		1 0 1 0		1 0 1 0		1 0 1 0		1 0 1 0
1 1 0 0		1 1 0 0		1 1 0 0		1 1 0 0		1 1 0 0
1 1 1 1		1 1 1 0		1 1 1 0		1 1 1 0		1 1 1 1

Isso nos permite escrever as equações lógicas para cada uma das quatro tabelas da seguinte forma:

$S_1 = \overline{A} . \overline{B} . C$ que
corresponde a A=0, B=0 e C=1

$S_2 = \overline{A} . B . C$ que
corresponde a A=0, B=1 e A=0

$S_3 = A . \overline{B} . \overline{C}$ que
corresponde a A=1, B=0 e C=0

$S_4 = A . B . C$ que
corresponde a A=1, B=1 e C=1

Como a saída S é a combinação das quatro funções temos:

$S = S_1 + S_2 + S_3 + S_4$

Substituindo pelos valores encontrados teremos:

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

Esta é então a função lógica que representa a tabela verdade que propusemos como parte inicial do problema e para a qual devemos encontrar um circuito equivalente.

Passo 2 - Implementação dos Circuitos Combinacionais

Conforme estudamos em lições anteriores, é possível usar as portas NAND e NOR como blocos lógicos universais a partir dos quais podemos elaborar qualquer outra função ou mesmo funções mais complexas.

Para exemplificar vamos analisar uma função um pouco mais simples do que a obtida no passo anterior. Tomemos a expressão:

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

Podemos tentar implementá-la usando portas NAND e eventualmente inversores, já que a barra sobre cada letra indica sua negativa, conforme estudamos.

A operação (.) pode ser realizada utilizando-se uma porta NAND que ligada a um inversor nos fornece uma porta AND.

Assim, conforme a **figura 4**, podemos implementar A.B.C usando uma porta NAND de 3 entradas e um inversor.

Veja na **figura 5** como a operação A.B.C pode ser implementada.

A soma (+) pode ser implementada com uma porta OR ligada a dois inversores, **figura 6**.

Figura 5 - Implementação da função A.B.C

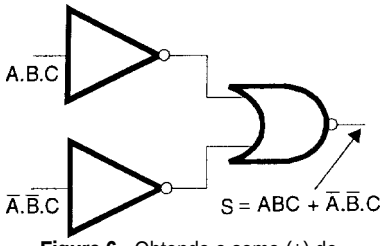


Figura 6 - Obtendo a soma (+) de duas expressão lógicas.

Combinando os três circuitos podemos chegar à configuração final desejada, **figura 7**.

Veja que a inversão da inversão usada no circuito anterior nos leva ao circuito original. Isso significa que podemos simplificar a configuração eliminando as duplas inversões em série. Isso nos leva à configuração final do circuito mostrada na **figura 8**.

Logo, quando temos uma expressão formada pela soma de produtos, podemos usar portas NAND sem a necessidade de inversores, bastando apenas lembrar duas propriedades:

As combinações de entrada podem ser aplicadas a portas NAND.

As saídas das portas NAND podem ser aplicadas à entrada de uma segunda porta NAND obtendo-se na saída a função desejada.

Vamos agora fazer uma tentativa de implementar uma função usando portas NOR, o que será escolhido quando tivermos um produto de somas.

Tomemos como exemplo a função:

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

As somas podem ser obtidas facilmente a partir de portas NOR com

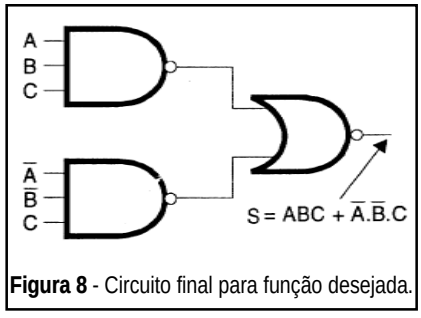


Figura 8 - Circuito final para função desejada.

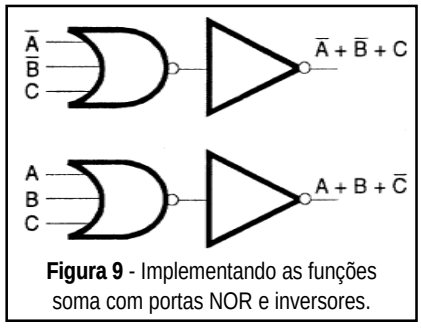


Figura 9 - Implementando as funções soma com portas NOR e inversores.

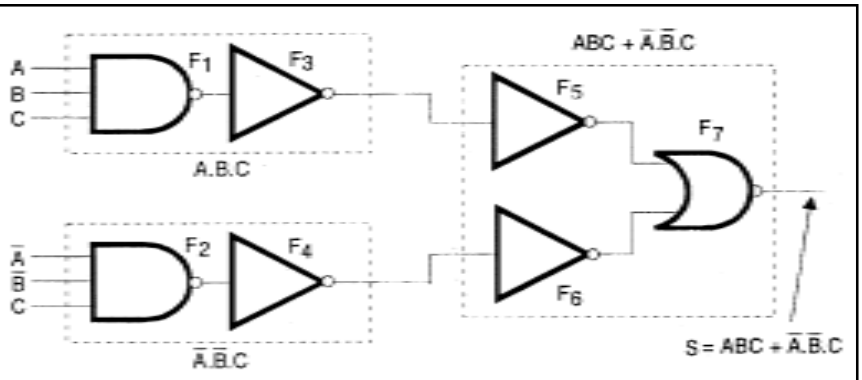


Figura 7 - A função S=$\overline{A}\overline{B}$C+ABC de forma que ainda pode ser minimizada.

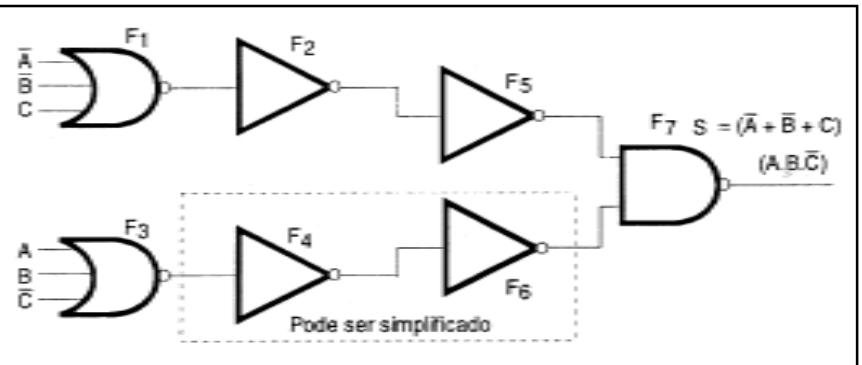


Figura 10 - A implementação quase final da função.

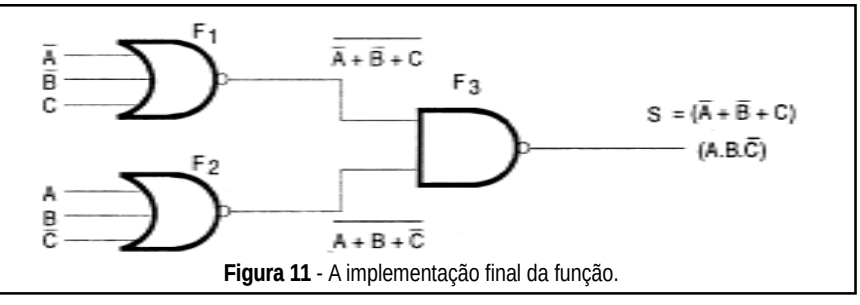


Figura 11 - A implementação final da função.

as saída aplicadas a um inversor. A negação de NOR é OR. O circuito equivalente para três entradas é mostrado na **figura 9**.

O produto das duas somas é obtido com dois inversores aplicando os sinais a uma outra porta OR, ou seja, a uma outra configuração NOR.

Como nas duas linhas de sinais temos inversores em série, e o inversor do inverso de um nível lógico é ele mesmo, podemos simplificar o circuito eliminando todos os inversores.

Isso nos permite chegar à configuração final que é mostrada na **..**

Assim, se quisermos implementar uma função que consiste num produto de somas, basta seguir dois procedimentos básicos:

Aplicar as entradas correspondentes a cada soma a uma porta OR que pode ser obtida associando-se uma porta NOR a uma inversor.

Aplicar as saídas obtidas nas funções que devem ser multiplicadas a inversores que são ligados às entradas de uma porta OR final, também obtida com a associação de um inversor a uma porta NOR.

Como os inversores em série se anulam, eles podem ser eliminados e o circuito implementado utilizando-se apenas portas NOR.

É possível resolver o problema de implementar circuitos combinacionais reduzindo as funções a produtos de somas ou ainda a soma de produtos,

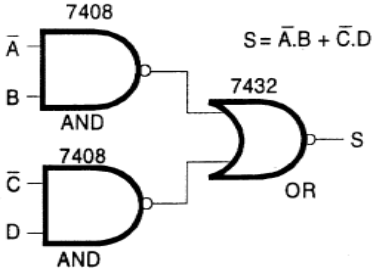


Figura 12 - Dois tipos diferentes de portas são usados neste circuito.

casos em que podemos trabalhar com funções NAND ou NOR.

Como as duas soluções levam aos mesmos resultados, num projeto prático é interessante analisar as configurações obtidas para um problema nos dois casos. Adota-se então a solução que utilizar menos circuitos ou que for mais conveniente, por exemplo, aproveitando portas ociosas de um circuito integrado já utilizado no mesmo projeto com outras finalidades.

5.4 - SIMPLIFICANDO E MINIMIZANDO

Uma consequência da possibilidade de construir funções complexas a partir de portas básicas como OR e AND (OU e E) é a otimização de um projeto aproveitando poucos tipos de circuitos integrados básicos.

Assim, se tivermos uma função que seja obtida utilizando-se portas AND e OR como a mostrada na figura 12, ela terá o inconveniente de precisar de dois tipos diferentes de circuitos integrados.

Se quisermos esta função com circuitos TTL, por exemplo, aproveitaremos três das três portas de três entradas de um circuito 7411 e também precisaremos aproveitar uma das quatro portas OR de duas entradas de um circuito integrado 7432.

Evidentemente, estaremos usando dois circuitos integrados, desperdiçando 1/3 de um e 3/4 do outro.

Podemos simplificar consideravelmente este circuito se usarmos apenas portas NAND com a configuração equivalente mostrada na figura 13.

Este circuito, que apresenta a mesma função do anterior, usa as três portas de um circuito integrado 7410. Utilizamos apenas um circuito integrado que é totalmente aproveitado, sem nenhuma parte ociosa.

5.5 - DIAGRAMAS DE KARNAUGH

Um processo bastante interessante para representar uma tabela verdade e a partir dela obter uma simplificação dos circuitos utilizados para sua implementação é o que faz uso dos chamados diagramas ou mapas de Karnaugh.

O diagrama de Karnaugh consiste numa tabela retangular com número de quadros que corresponde a 2 elevado ao expoente N, onde N é o número de variáveis do circuito.

Cada variável lógica ocupa no gráfico metade da sua extensão e seu complemento ocupa a outra metade.

Na **figura 13** temos o modo como são elaborados os diagramas de Karnaugh para 1, 2 e 3 variáveis, com as expressões lógicas correspondentes a cada caso.

Estas expressões são obtidas de uma forma muito semelhante à usada no conhecido joguinho de "batalha naval" onde a posição de cada "tiro" é dada por duas coordenadas, uma correspondente às linhas e outra às colunas.

Na **figura 15** mostramos, como exemplo, de que modo um diagrama de Karnaugh de 4 variáveis pode ser obtido com a inclusão dentro de cada quadro da expressão correspondente. No diagrama (b) da **figura 14** os quadros foram preenchidos com os valores 0 e 1 correspondentes às entradas. Este diagrama é chamado também de diagrama de Veitch. Uma observação importante em relação a esta representação por 0 e 1 é que

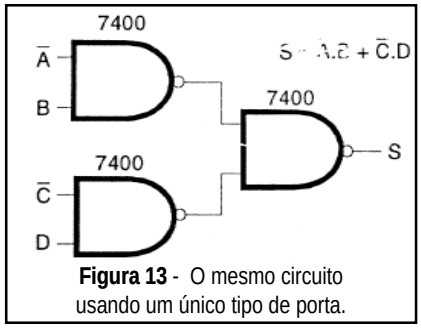


Figura 13 - O mesmo circuito usando um único tipo de porta.

cada quadro difere do adjacente em apenas um dígito.

Dizemos que são adjacentes os termos que estão à direita e à esquerda de cada quadro e também os que estão acima e abaixo. Também são adjacentes os que estiverem na mesma fila, mas um na primeira coluna e outro na última.

Na **figura 16** temos um mapa com a identificação das adjacências.

Assim, o que fazemos é plotar a tabela verdade da função que desejamos implementar num mapa de Karnaugh com o que será possível identificar melhor as adjacências e assim fazer as simplificações.

Para que o leitor entenda como "funciona" o mapa de Karnaugh numa simplificação de uma função, vamos tomar como exemplo a função que é dada pela seguinte tabela verdade:

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

Desejamos expressar esta tabela como a soma de produtos, o que significa que os valores adjacentes que

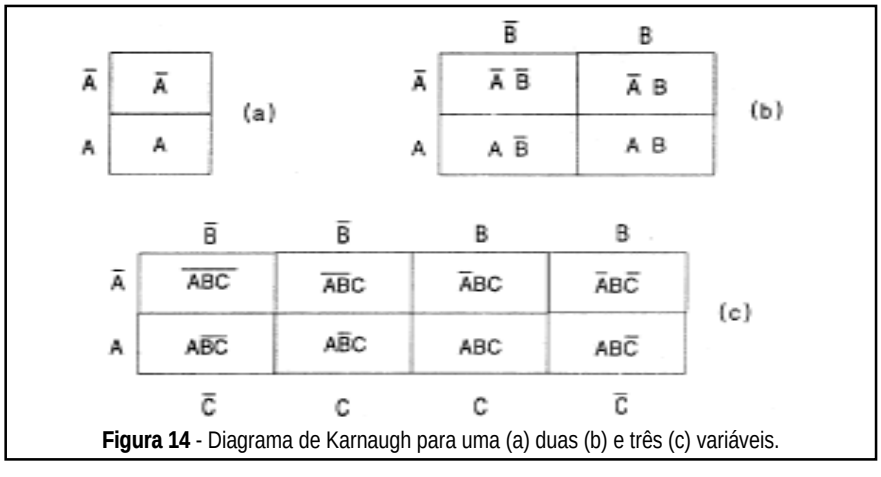


Figura 14 - Diagrama de Karnaugh para uma (a) duas (b) e três (c) variáveis.

devemos procurar na tabela são os "1". Se fôssemos expressar esta função como o produto de uma soma, os valores considerados seriam os "0" e o procedimento final seria o mesmo.

Construímos então o Diagrama de Karnaugh para esta tabela conforme a **figura 17**.

A partir deste diagrama nosso próximo passo consiste em tentar fazer simplificações que possam levar a circuitos mais simples na implementação.

A idéia é agrupar os termos adjacentes iguais, havendo para isso diversas possibilidades que são apresentadas na **figura 18**.

A primeira possibilidade mostrada em (a) nos leva a uma soma de três produtos, cada qual obtido pela intersecção da linha com a coluna em que está o "1" correspondente.

Assim, o primeiro está na coluna que intercepta A-0 com B-0. Ora, o valor zero na indexação indica inversão, portanto, isso significa que o primeiro fator de nosso produto será:

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

O segundo "1" a ser considerado está na coluna A=1 e B=0, portanto, temos A invertido e B sem inversão, o que nos leva ao segundo fator de nosso produto:

$$\overline{A}.B$$

Finalmente, o terceiro "1" a ser considerado está na linha A=1 e B=1, o que significa um fator com A multiplicado por B sem inversões ou:

$$A.B$$

Como devemos expressar a função na forma de uma soma de produtos fazemos:

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

Para o segundo caso (b) temos uma simplificação maior, já que agrupamos os dois "1" da primeira linha de modo que podemos adotar para ele:

$$\overline{A}$$

Para o outro valor "1" que está na casa que corresponde à intersecção de A-1 com B-1 vale a soma (sem inversão):

$$A + B$$

A expressão final na forma de um produto de somas será então:

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

Da mesma forma chegamos à simplificação (b) que permite a expressão mais simples, pois conseguimos juntar três casas adjacentes.

Raciocinando da mesma forma chegamos à expressão:

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

O procedimento que vimos como exemplo envolveu uma função simples com apenas duas variáveis de entrada.

No entanto, o mesmo procedimen-

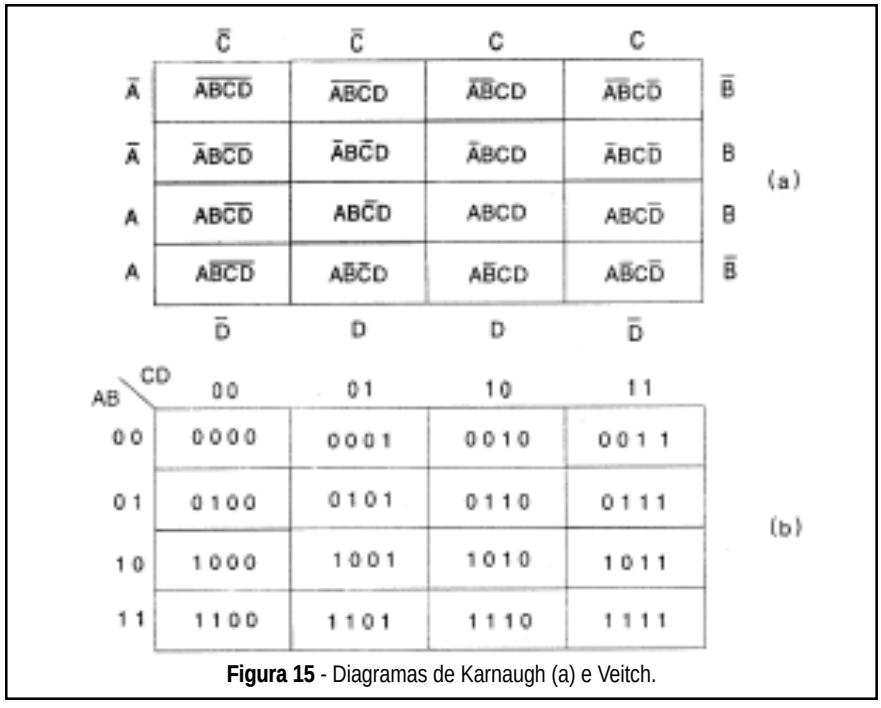


Figura 15 - Diagramas de Karnaugh (a) e Veitch.

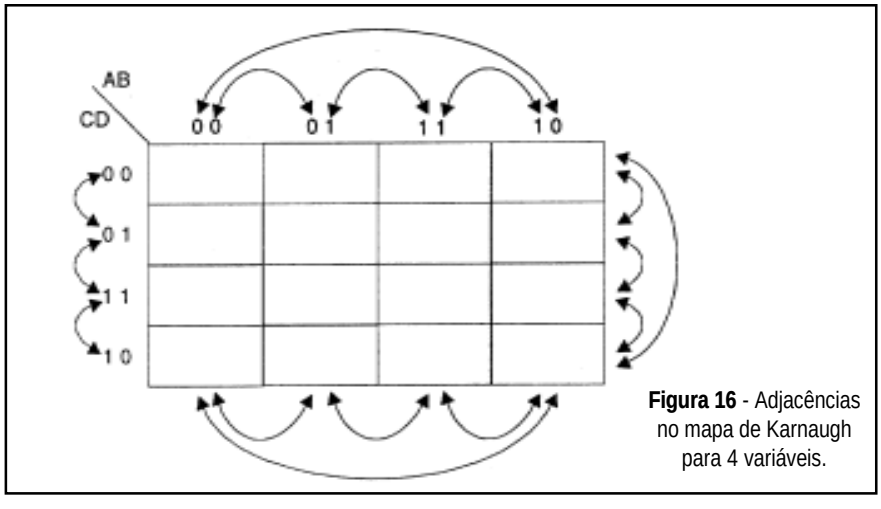


Figura 16 - Adjacências no mapa de Karnaugh para 4 variáveis.

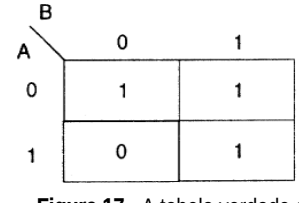


Figura 17 - A tabela verdade é plotada no Mapa de Karnaugh.

to é válido para qualquer número de variáveis. Os leitores interessados em aprofundar-se neste estudo devem procurar treinar os procedimentos indicados, trabalhando com funções cada vez mais complexas.

CONCLUSÃO

O espaço disponível para nosso curso não permite um aprofundamento maior neste assunto e um certo treino se faz necessário para o domínio das técnicas envolvidas. Assim, para os leitores interessados no tema, sugerimos a procura de literatura complementar. Mostramos os procedimentos lógicos que permitem trabalhar com as funções de modo a chegar aos circuitos.

Assim, uma tabela verdade que tenha qualquer combinação de entradas que nos leve a qualquer combinação de saída pode ser elaborada na prática com funções básicas (NOR e NAND) e isso não exige que se "quebre a cabeça".

Conhecendo os procedimentos para resumir tudo em produto de somas e soma de produtos e também o uso dos mapas de Karnaugh para simplificação, obteremos configurações simples que facilitam qualquer projeto.

QUESTIONÁRIO

1. Os valores combinados de todas as entradas e a saída correspondente podem ser colocados numa tabela denominada:

a) Mapa de Karnaugh
b) Diagrama de Veitch
c) Tabela verdade
d) Produto de somas

2. A tabela verdade abaixo, corresponde à qual função:

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

a) AND (E)
b) NAND (Não-E)
c) OR (OU)
d) NOR (Não-OU)

3. Qualquer circuito lógico pode ser implementado utilizando-se que funções básicas?

a) NAND e inversores
b) NAND e NOR
c) OR e Inversores
d) AND e Inversores

4. Para implementar um circuito que corresponda a uma função dada por uma soma de produtos usamos quais funções lógicas?

a) Portas NAND
b) Inversores
c) Portas OR
d) Não é possível fazer isso

5. Se numa implementação lógica precisarmos usar inversores em série, o que podemos fazer com eles?

a) Ligá-los à portas AND
b) Colocá-los em paralelo
c) Inverter suas saídas
d) Eliminá-los ■

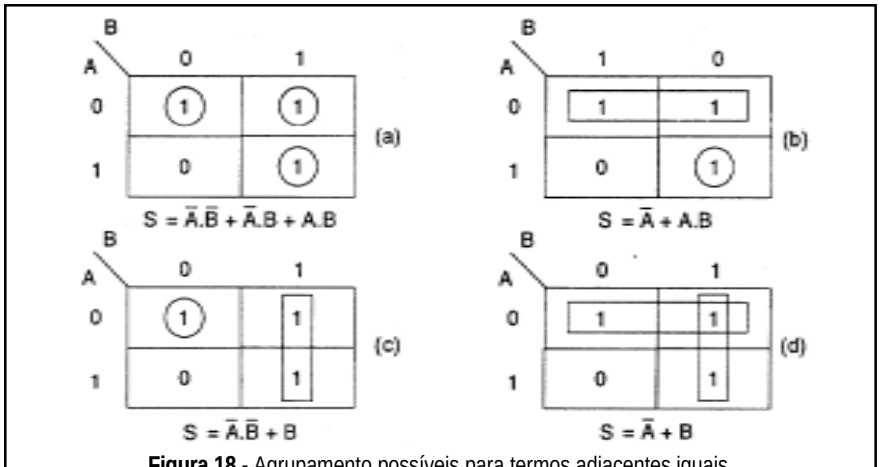


Figura 18 - Agrupamento possíveis para termos adjacentes iguais.

Respostas: 1-C, 2-B, 3-B, 4-A, 5-D

LIÇÃO 6

OS ELEMENTOS BIESTÁVEIS

Na lição anterior analisamos os modos segundo os quais podemos saber o que acontece quando combinamos funções lógicas. Vimos os procedimentos utilizados para implementar um circuito a partir de uma tabela verdade ou ainda da expressão da função lógica. No entanto, as funções lógicas não consistem nos únicos blocos básicos usados nos projetos de circuitos digitais. Além dessas funções, existem outras e um grupo delas que executa funções de relevante importância nos equipamentos são as formadas pelos elementos biestáveis. Nesta lição veremos como funcionam estes elementos, os seus tipos e onde podem ser usados.

6.1 - OS *FLIP-FLOPS*

Os *flip-flops* são elementos de circuito que podem apresentar em seu funcionamento apenas dois estados estáveis. Não existem estados intermediários entre estes dois estados.

A aplicação de um sinal de entrada pode mudar o dispositivo de um estado para outro e como a qualquer momento podemos saber qual é o estado em que ele se encontra, é possível considerar este circuito como uma memória capaz de armazenar um bit.

O *flip-flop* é o elemento básico das chamadas memórias estáticas.

Existem diversos tipos de *flip-flops* encontrados nos circuitos digitais e que analisaremos a partir de agora.

6.2 - *FLIP-FLOP R-S*

O *Flip-Flop R-S* (de *Reset e Set*) tem sua configuração com transistores mostrada na **figura 1** e funciona da seguinte maneira:

Quando alimentamos o circuito, dada as mínimas diferenças que podem existir entre as características dos dois transistores, um deles conduzirá mais do que o outro. Supondo que este transistor seja Q_1, há uma queda de tensão no seu coletor que reduz em consequência a corrente que polariza a base de Q_2 via R_2.

Nestas condições, a tensão do coletor de Q_2 se mantém alta, realimentando a base de Q_1 via R_3 e a situação final do circuito é estabelecida: Q_1 satura e Q_2 fica no corte. O *flip-flop* encontra seu estado estável inicial.

O *flip-flop R-S* tem duas saídas representadas por Q e /Q, assim, na condição inicial estável, com Q_1 conduzindo, Q estará no nível baixo (0) e /Q estará no nível alto (1).

O processo que leva o *flip-flop* a este estado inicial pronto para funcionar é muito rápido, não demorando mais do que alguns microssegundos.

Quando o *flip-flop* se encontra na situação indicada, com Q=0 e /Q=1, dizemos que ele se encontra "setado" ou armado.

A mudança de estado do *flip-flop* pode ser obtida aplicando-se um sinal conveniente na entrada. Como usamos transistores NPN para comutar o *flip-flop,* temos de fazer conduzir por um instante o transistor que está cortado, ou seja, devemos aplicar um pulso positivo na entrada correspondente.

Assim, estando o *flip-flop* na condição indicada, se desejarmos mudar o estado, aplicamos o pulso na entrada SET. O transistor Q_2 conduz por um instante, realimentando via R_3 a base de Q_1 que é cortado.

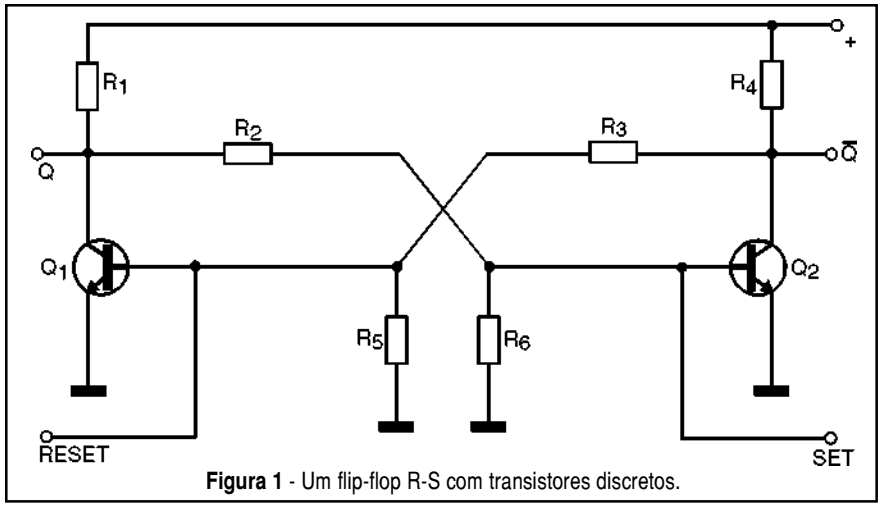


Figura 1 - Um flip-flop R-S com transistores discretos.

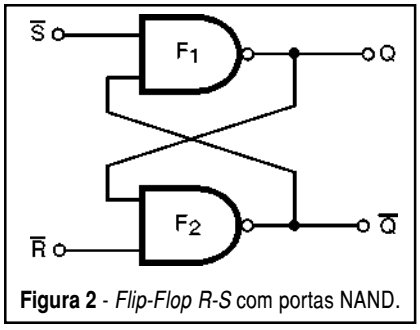


Figura 2 - *Flip-Flop R-S* com portas NAND.

Com o corte, a tensão na base de Q_2 sobe via polarização de R_2 e mesmo que o pulso de disparo desapareça, o circuito se mantém no novo estado graças à realimentação.

Sua saída Q vai ao nível (1) e a saída /Q vai ao nível (0).

Para trocar novamente de estado o *flip-flop R-S,* aplicamos um pulso positivo na entrada RESET, levando Q_1 à saturação e Q_2 ao corte, situação que se firma mesmo depois de desaparecido o pulso graças à realimentação proporcionada pelos resistores.

Veja que um pulso aplicado à entrada SET, o que corresponde a um bit 1, faz com que a saída Q que estava em 0 passe a 1, armazenando este bit. O *flip-flop* funciona realmente como uma memória para este bit.

Da mesma forma como utilizamos transistores bipolares NPN para obter um *flip-flop,* podemos também empregar outros tipos de componentes em configurações semelhantes. Podemos, por exemplo, elaborar *flip-flops* usando transistores PNP, caso em que a polaridade dos sinais de disparo vai ser invertida.

Da mesma forma, podemos usar transistores de efeito de campo, tanto de canal N como canal P (bipolares ou JFETs) como também transistores de efeito de campo MOS com os dois tipos de canal (N ou P). O que mudará em cada caso é o sentido de circulação das correntes e as polaridades dos sinais aplicados.

Conforme veremos na última parte desta lição, os *flip-flops* também podem ser feitos com válvulas e na realidade os primeiros que existiram eram justamente montados com estes componentes. Naquela época não existiam transistores e nem circuitos integrados.

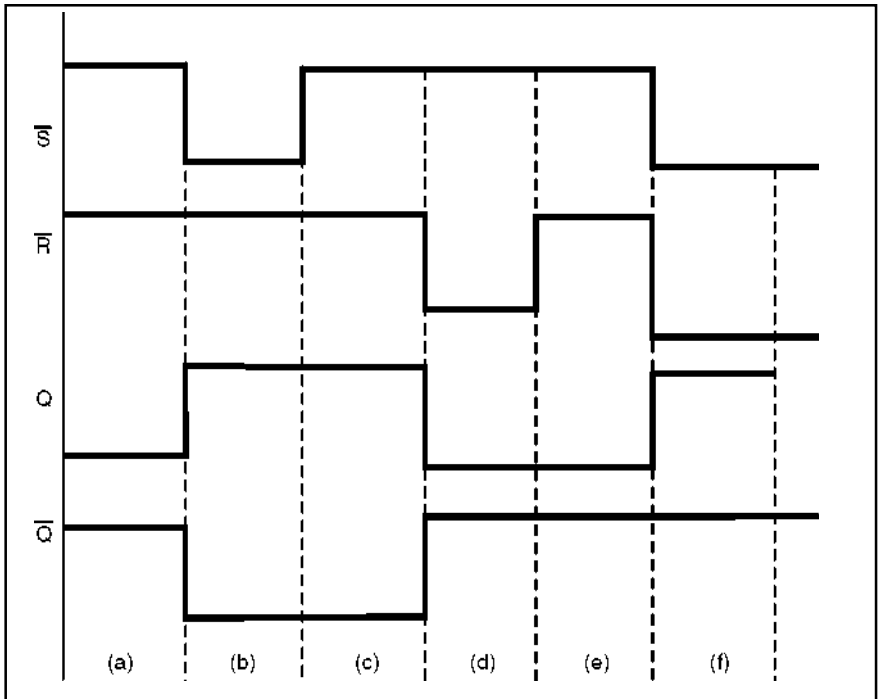


Figura 3 - Em (f) temos uma condição não permitida com Q e $\overline{Q}$ no nível alto $\overline{(S \text{ e } R}$ no nível baixo).

Os *flip-flops* podem ser elaborados com portas lógicas e o R-S que estudamos pode ser facilmente obtido a partir de duas portas NAND de duas entradas, **figura 2**.

Levando em conta as tabelas verdade das portas NAND veremos que a saída da primeira porta realimenta a segunda e vice-versa, garantindo assim a manutenção dos estados obtidos quando o *flip-flop* comuta.

No entanto, a comutação deste circuito ocorre quando as entradas passam do nível alto para o baixo, ou seja, de 1 para 0. Esta condição é indicada pelos símbolos /R e /S na entradas.

O leitor pode então perceber que, quando as entradas estão ambas no nível baixo, o *flip-flop* se mantém no estado em que foi colocado por ser ligado ou por uma comutação anterior.

Por outro lado, se as entradas forem levadas simultaneamente ao nível alto, o *flip-flop* irá para um estado indeterminado que deve ser evitado. Na prática, a aplicação de níveis altos (1) nas duas entradas pode destruir o dispositivo.

O diagrama de tempos da **figura 3** mostra o que ocorre no funcionamento de um *flip-flop* por etapas que podemos analisar da seguinte forma:

a) *Flip-flop* resetado

b) /S vai ao nível baixo e o *flip-flop* é setado

c) /S vai ao nível alto e o *flip-flop* permanece setado

d) /R vai ao nível baixo e o *flip-flop* é ressetado

e) /R volta ao nível alto e o *flip-flop* permanece ressetado

Tudo isso pode ser representado por uma tabela verdade, da mesma forma que fazemos com as funções lógicas. Nesta tabela temos alguns novos símbolos com os quais o leitor deve começar a familiarizar-se e que são amplamente usados em Eletrônica Digital, a saber:

a) Primeira possibilidade

Qn-1 = representa o estado da saída Q ANTES da aplicação dos sinais.

Qn = representa o estado da saída Q DEPOIS da aplicação dos sinais.

b) Segunda possibilidade

Q = representa o estado da saída Q ANTES da aplicação dos sinais.

Qn+1 = representa o estado da saída Q DEPOIS da aplicação dos sinais.

Obs: em lugar de n em alguns livros encontramos a letra t.

Os dois tipos de representação são usados.

Nas colunas e linhas em que são colocados os níveis lógicos 0 e 1 , quando aparece o termo Qn ou /Qn significa que a saída vai para um estado indeterminado.

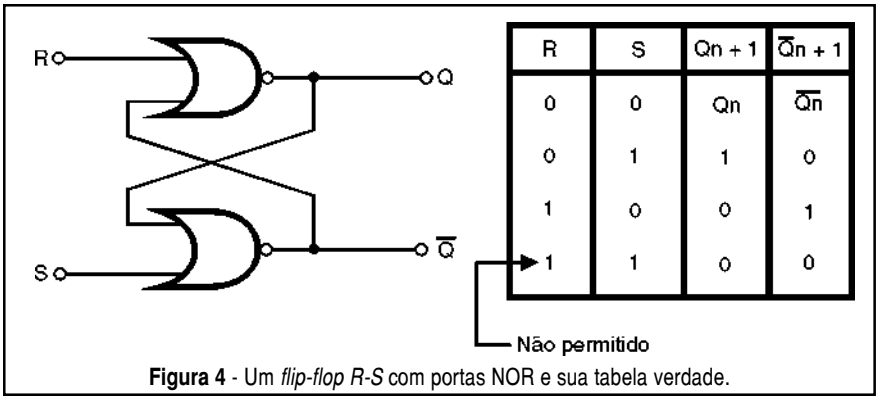
A tabela verdade do *flip-flop R-S* com portas NAND fica então:

R	S	Qn+1	Qn+1
0	0	1	1
0	1	0	1
1	0	1	0
1	1	Qn	Qn

Para obtermos um *flip-flop R-S* também podemos usar portas NOR, conforme a **figura 4**.

Na **figura 5** temos os símbolos adotados para representar este tipo de *flip-flop*.

Este circuito também é chamado de *R-S NOR LATCH* da mesma forma que o circuito anterior é denominado *R-S NAND LATCH*.



R	S	Qn + 1	$\overline{Q}$n + 1
0	0	Qn	$\overline{Q}$n
0	1	1	0
1	0	0	1
1	1	0	0

Figura 4 - Um *flip-flop R-S* com portas NOR e sua tabela verdade.

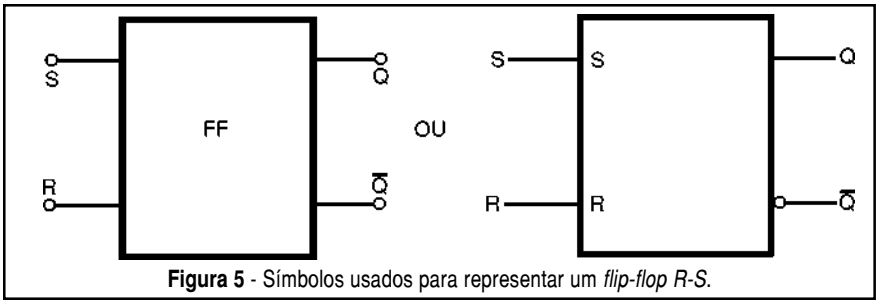


Figura 5 - Símbolos usados para representar um *flip-flop R-S*.

6.3 - *FLIP-FLOP RS* COM *CLOCK* E MESTRE-ESCRAVO

Estes circuitos chamados de *flip-flop R-S* controlados por *clock* e mestre escravo encontram uma gama de aplicações muito grande nos circuitos digitais mais complexos, já que estes são sempre comandados por um *clock*, ou seja, são circuitos lógicos sincronizados.

O uso de um circuito de controle (mestre) que determina quando o *flip-flop* (escravo) muda de estado é importante para permitir que as mudanças de estado do *flip-flop* só ocorram em determinados instantes.

Usando portas NAND podemos inicialmente implementar um *flip-flop R-S* controlado por *clock (Master-Slave),* conforme a figura 6.

Analisemos seu funcionamento:

Partindo da situação em que a entrada de *clock* (relógio) esteja no nível baixo, as saídas Q e /Q permanecerão no estado inicial em que se encontravam e insensíveis a qualquer variação que ocorra nas entradas S e R.

Quando a entrada de *clock* for levada ao nível 1, o circuito passa a responder aos sinais das entradas R e S.

No entanto, conforme o diagrama de tempos da **figura 7**, este circuito tem um inconveniente.

Como as saídas acompanham as entradas, durante o tempo em que o *clock* as habilita, estas saídas podem mudar de estado mais de uma vez, voltando assim ao estado inicial, o que não é desejado de forma alguma.

Um modo de contornar este proble|ma consiste na utilização de duas etapas numa configuração mais complexa, que é apresentada na **figura 8**.

Este circuito é denominado *Flip-Flop R-S* Mestre-Escravo ou *Flip-Flop R-S Master-Slave* e faz uso de portas NAND e de um inversor, cuja finalidade é inverter o pulso de *clock*.

Neste caso, quando a entrada de *clock* for ao nível 1, o *flip-flop* mestre mudará de estado, mas o *flip-flop* escravo permanecerá insensível, mantendo seu estado.

Quando a entrada de *clock* passar para o nível lógico 0, a saída do *flip-flop* mestre será levada para o escravo.

Isso significa que o *flip-flop* em seu todo não é sensível ao nível do sinal de *clock*, ou seja, se ele é 0 ou 1, mas sim à sua transição. As saídas Q e /Q só vão mudar de estado no instante em que ocorrer a transição do sinal de *clock* do nível alto para o nível baixo. Com esta configuração é possível garantir que só vai ocorrer uma mudança de estado na presença de um pulso de *clock*.

Os *flip-flops* que funcionam desta forma são denominados *"Edge Triggered"* ou "Disparados pela Bor-

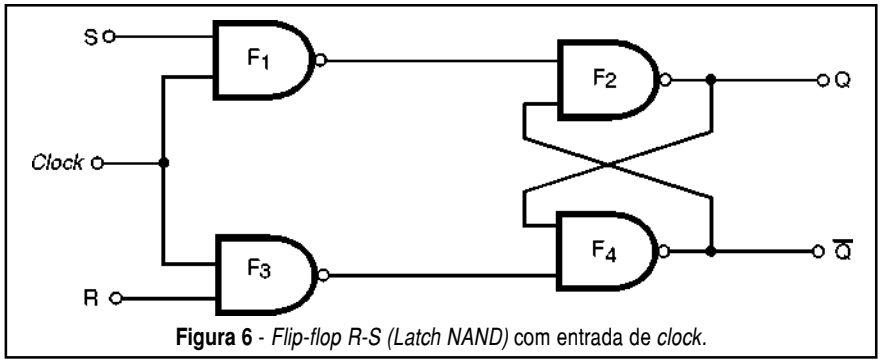


Figura 6 - *Flip-flop R-S (Latch NAND)* com entrada de *clock*.

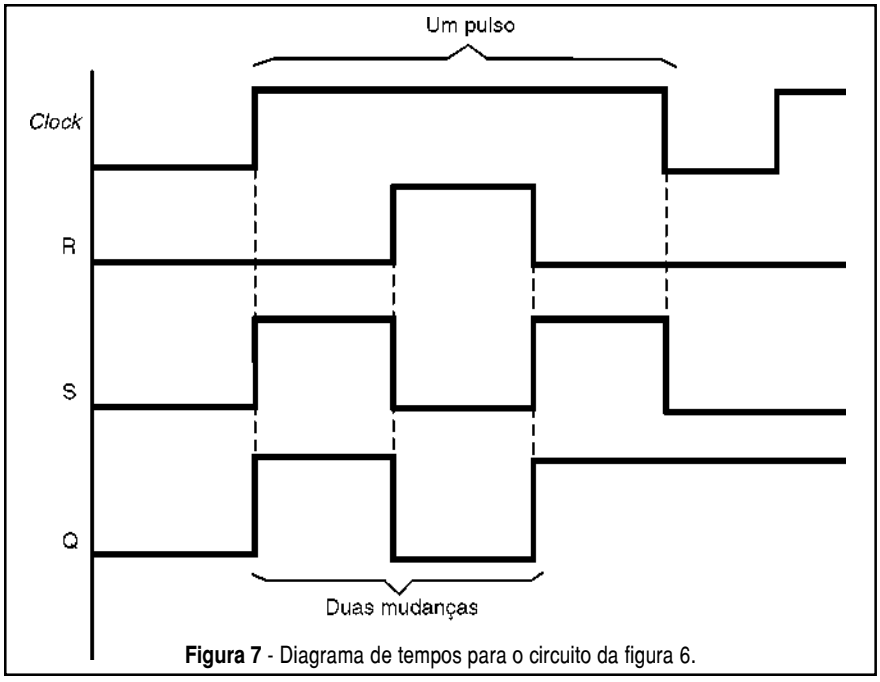


Figura 7 - Diagrama de tempos para o circuito da figura 6.

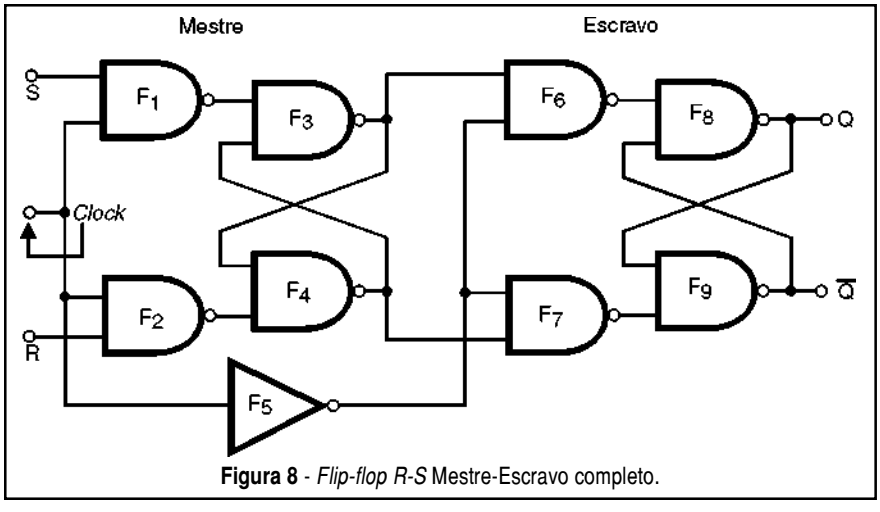


Figura 8 - *Flip-flop R-S* Mestre-Escravo completo.

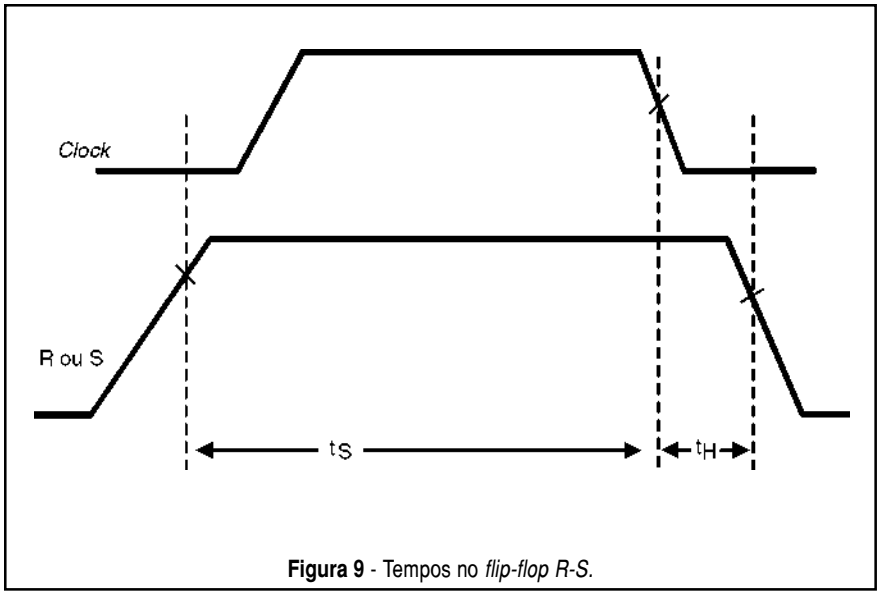


Figura 9 - Tempos no *flip-flop R-S*.

da".

Se a mudança de estado ou disparo (gatilhamento) ocorre quando o sinal de *clock* passa de 0 para 1, os *flip-flops* são denominados *"positive edge-triggered",* enquanto que, se o disparo ocorre quando o *clock* vai do nível 1 para 0, ou seja, na queda do nível lógico, os *flip-flops* são chamados de *"negative edge-triggered"*.

Neste tipo de circuito é muito importante levar em conta, num projeto de maior velocidade, os tempos em que todo o processo ocorre.

Assim, partindo do diagrama de tempos da **figura 9**, vemos que a saída do *flip-flop* só completa sua mudança de estado depois de um certo tempo, do pulso de *clock* ter sido aplicado.

Dois tempos são importantes neste tipo de circuito.

a) tH: *Hold Time* ou Tempo de Manutenção é o tempo em que a entrada deve permanecer ainda no circuito para que seu nível lógico seja reconhecido pelo *flip-flop*.

b) tS: *Setup Time* ou tempo em que a entrada do *flip-flop* deve permanecer no estado desejado antes da transição do *clock* que vai provocar a mudança de estado do circuito.

Duas entradas podem ser acrescentadas neste circuito, verifique a **figura 10**, dotando-o de recursos importantes para aplicações práticas.

Uma das entradas é denominada *PRESET (/PR)* ou pré-ajuste e tem por função levar imediatamente as saídas do circuito a um estado determinado (Q=1 e /Q=0), independentemente do que esteja acontecendo nas demais entradas.

Sua ativação ocorre quando /PR estiver em 0 e /CLR em 1, no caso apresentado, pois a / sobre a identificação indica que ela está ativa no nível baixo.

A outra entrada denominada *CLEAR* ou apagamento tem por função levar as saídas aos estados Q=0 e /Q=1, independentemente do que estiver ocorrendo nas demais entradas.

É importante observar que estas duas entradas não podem ser ativadas ao mesmo tempo, pois isso levaria o circuito a um estado indeterminado que inclusive poderia

causar problemas aos seus componentes.

A tabela verdade para este circuito nos mostra três novos símbolos que normalmente são usados em Eletrônica Digital.

X representa uma condição irrelevante qualquer que ela seja, não haverá influência no que ocorre na saída.

A seta para cima indica a transição do nível baixo para o nível do sinal na entrada ou saída representadas, enquanto que a seta apontando para baixo indica uma transição do nível baixo para o nível alto do sinal correspondente.

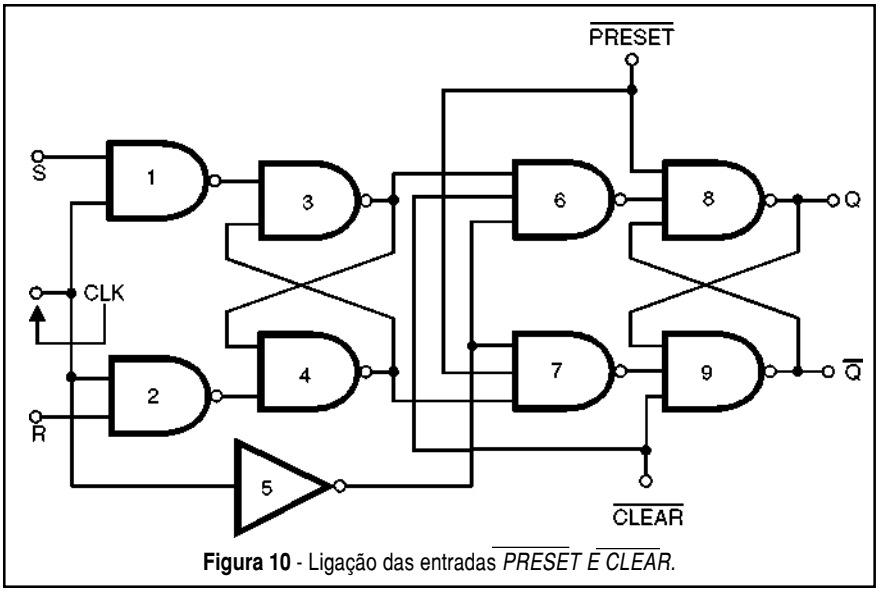


Figura 10 - Ligação das entradas *PRESET* E *CLEAR*.

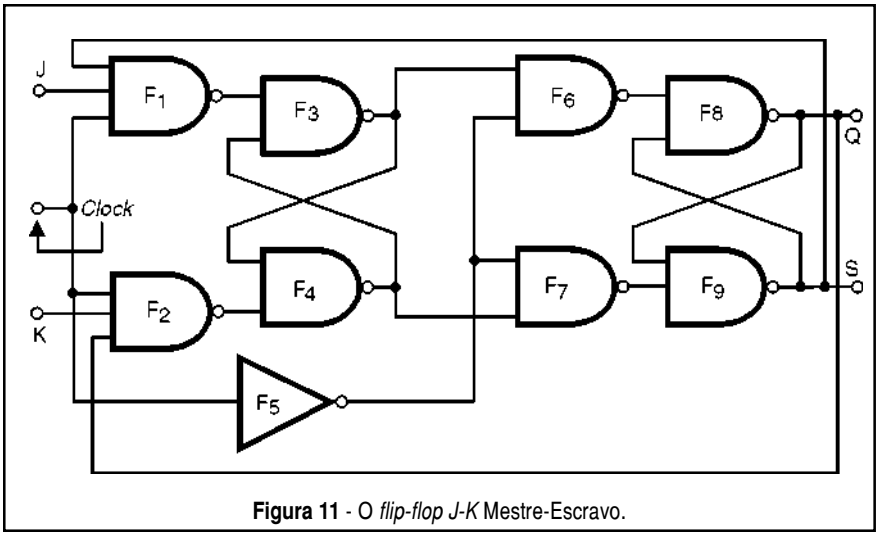


Figura 11 - O *flip-flop J-K* Mestre-Escravo.

6.4 - O *FLIP-FLOP J-K* MESTRE-ESCRAVO

O *flip-flop J-K* mestre-escravo ou *"master-slave"* pode ser implementado por funções lógicas comuns, adquirindo a configuração básica mostrada na **figura 11**.

Um problema que observamos nos *flip-flops R-S* é que temos uma situação "proibida" que ocorre quando as entradas R e S vão ao nível alto ao mesmo tempo e que pode levar o circuito a um estado indeterminado. Esta situação acontece principalmente nas aplicações em computação, quando uma parte do sinal de saída é usada para realimentar a entrada. Nestas condições podem ocorrer as situações de conflito com a produção de oscilações indesejadas.

Esta situação pode ser contornada com a utilização de uma nova configuração, que é justamente a do *flip-flop J-K* utilizada nas aplicações práticas e que analisaremos a seguir.

Podemos ter quatro combinações possíveis para os sinais aplicados nas entradas J e K, conforme observamos na tabela abaixo.

J	K
0	0
1	0
0	1
1	1

Analisemos cada uma das combinações:

a) J=0 e K=0

Quando a entrada de *clock (CLK)* passa por uma transição negativa do sinal, o *flip-flop* mantém sua condição original, ou seja, não muda de estado.

b) J=1 e K=0

Quando a entrada de *clock (CLK)* passa por uma transição negativa, o *flip-flop* é "setado". Se já estiver setado, ele permanece nesta condição.

c) J=0 e K=1

Quando a entrada de *clock (CLK)* passa por uma transição negativa, o *flip-flop* é "ressetado". Se já estiver nesta condição, ele permanece.

d) J=1 e K=1

Nesta condição, ao receber uma transição negativa na entrada de *clock (CLK),* o *flip-flop* muda de estado *(TOGGLE)*. Se estiver setado, ele resseta e se estiver ressetado, ele é setado.

Podemos elaborar a tabela verdade da **figura 12** para indicar o que ocorre com este *flip-flop*.

Observe o uso das setas para indicar as transições de sinal na entrada de *clock* que comandam o funcionamento deste tipo de circuito.

Da mesma forma que nas outras configurações estudadas, podemos também incluir as entradas de *PRESET* e *CLEAR* neste circuito que ficará da maneira apresentada na **figura 13**.

Uma tabela verdade incluindo as entradas de *PRESET (PR)* e *CLEAR (CLR)* é mostrada na **figura 14**.

Uma maneira melhor de analisar-

CLK	J	K	Qn + 1	$\overline{Q}$n + 1	
↓	Ø	Ø	Qn	$\overline{Q}$n	Não muda
↓	Ø	1	Ø	1	Reseta
↓	1	Ø	1	Ø	Seta
↓	1	1	$\overline{Q}$n	Qn	Complementa (Toggle)

Figura 12 - Tabela verdade para o *flip-flop J-K* Mestre-Escravo.

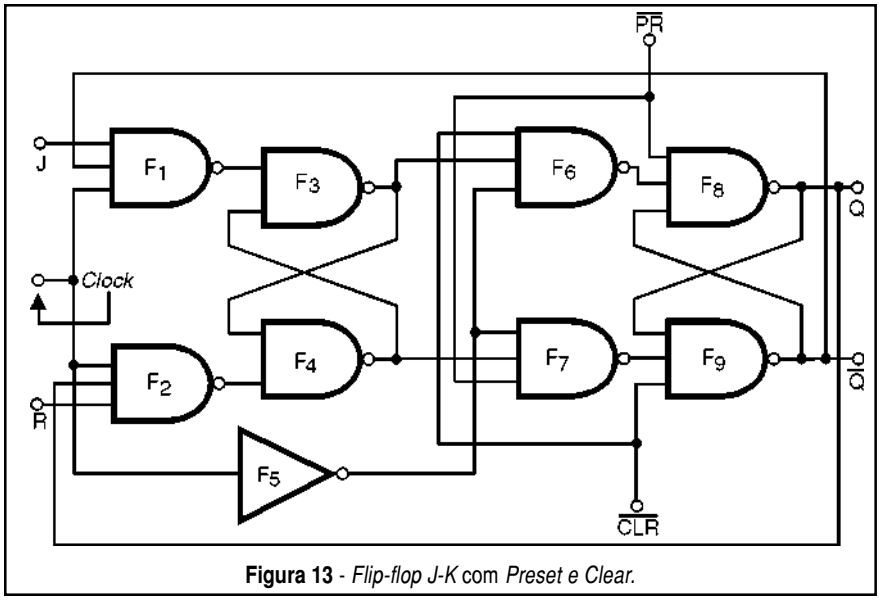


Figura 13 - *Flip-flop J-K* com *Preset e Clear.*

$\overline{CLR}$	$\overline{PR}$	J	K	CLK	Qn + 1	$\overline{Q}$n + 1	
Ø	Ø	X	X	X	1	1	(*)
Ø	1	X	X	X	Ø	1	
1	Ø	X	X	X	1	Ø	
1	1	Ø	Ø	↓	Qn	$\overline{Q}$n	
1	1	Ø	1	↓	Ø	1	
1	1	1	Ø	↓	1	Ø	
1	1	1	1	↓	$\overline{Q}$n	Qn	

Figura 14 - Tabela verdade para o *flip-flop J-K* com *Preset e Clear.*

mos o funcionamento deste circuito é através de um diagrama de tempos, em que observamos as formas de onda nos diversos pontos de entrada e saída. Este diagrama de tempos para o *flip-flop J-K* é mostrado na **figura 15**.

Analisemos alguns trechos importantes deste diagrama mostrando o que acontece:

a) Neste instante CLR e PR estão no nível baixo, Q e /Q estão no nível alto, que é uma condição não permitida.

b) Aplica-se então o sinal PR, que indo ao nível alto, faz com que o *flip-flop* seja ressetado.

c) A aplicação de um pulso na entrada CLR que vai ao nível alto, e a ida de PR ao nível baixo fazem agora com que o *flip-flop* seja setado.

d) CLR e PR são mantidos no nível alto a partir deste instante. Com J=0 neste trecho e K indo ao nível alto, o *flip-flop* será ressetado na próxima transição negativa do sinal de *clock*.

e) Ainda com CLR e PR no nível alto (esta condição se manterá daqui por diante) e a saída J=0 e k=1, o *flip-flop* permanecerá ressetado.

f) Com J=1 e K=0, o *flip-flop* é setado na transição seguinte do pulso de *clock*.

g) Com J=1 e K=0, não ocorrem mudanças de estado.

h) Com J=1 e K=1 na transição seguinte do pulso de *clock*, o *flip-flop* muda de estado (complementa ou *"toggle"*). Se estiver ressetado, como neste caso, ele é setado.

i) Mantendo J=1 e K=1 com nova transição do pulso de *clock,* o *flip-flop* muda de estado outra vez, ou seja, complementa.

Veja que quando as entradas J e K estão no nível alto, o circuito se comporta como um disparador, mudando de estado a cada transição negativa do pulso de *clock*.

6.5 - O *FLIP-FLOP* TIPO D

Este é também um circuito de *flip-flop* muito usado, cujo símbolo é mostrado na **figura 16**.

Este *flip-flop* possui uma única entrada que comanda todo o circuito. Esta entrada é que lhe dá nome. De-

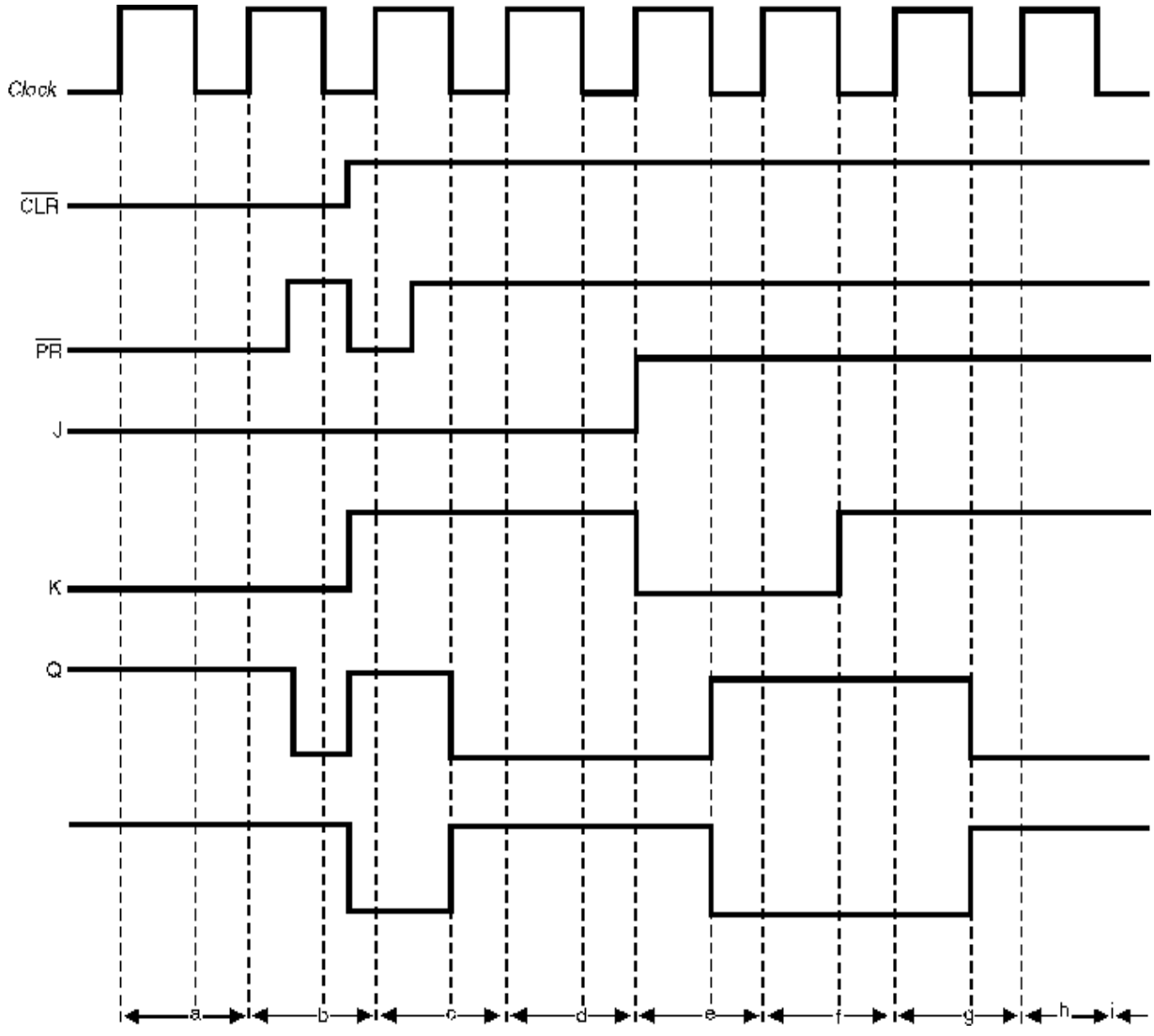


Diagrama de tempo do *flip-flop J-K*.

nominada “Data” (dados), é abreviada por D, daí o nome do dispositivo.

Este *flip-flop* opera de uma maneira muito simples: no pulso de *clock,* ele assume o estado da entrada, conforme podemos ver pela sua tabela verdade:

D	Qn+1
0	0
1	1

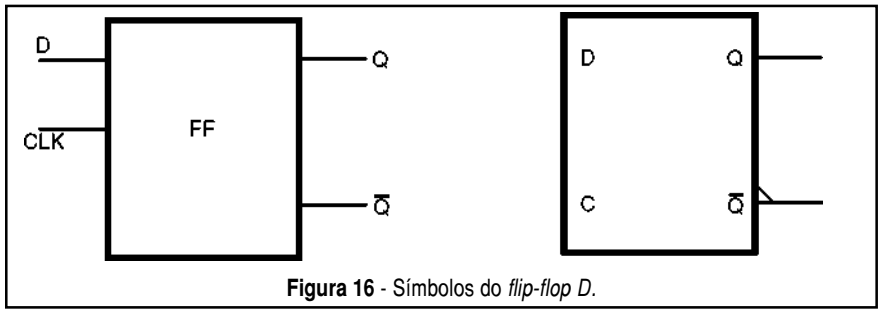


Figura 16 - Símbolos do *flip-flop D*.

6.6 - *FLIP-FLOP* TIPO *T*

O nome vem de *“Toggle”* ou complementação, seu símbolo é mostrado na **figura 17**.

O que este circuito faz pode ser entendido facilmente pelo diagrama de tempos mostrado na **figura 18**. Quando a entrada T deste circuito está no nível baixo, o *flip-flop* se mantém em seu estado anterior, mesmo com a aplicação do pulso de *clock*. No entanto, quando a entrada T está no nível alto, o *flip-flop* muda de estado. Se estava setado, ele resseta e se estava ressetado, ele seta.

Este comportamento significa na realidade a divisão da frequência de *clock* por dois. Em outras palavras, este circuito se comporta como um divisor de frequência, encontrando aplicações práticas bastante importantes em Eletrônica Digital.

Um exemplo de aplicação é dado na **figura 19** em que associamos di-

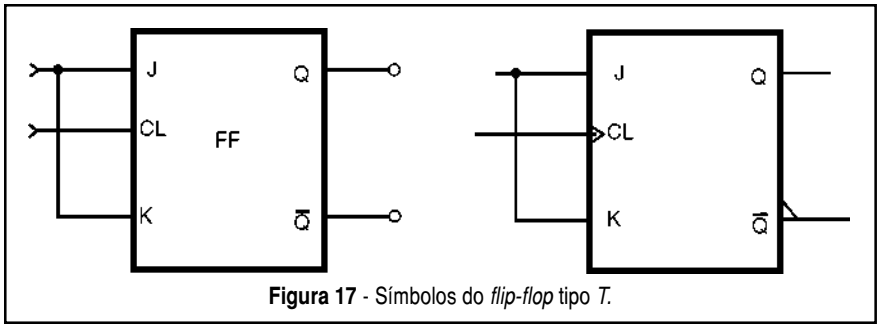


Figura 17 - Símbolos do *flip-flop* tipo *T.*

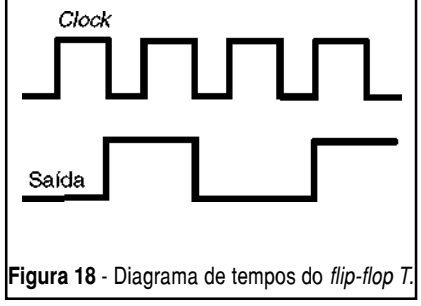


Figura 18 - Diagrama de tempos do *flip-flop T.*

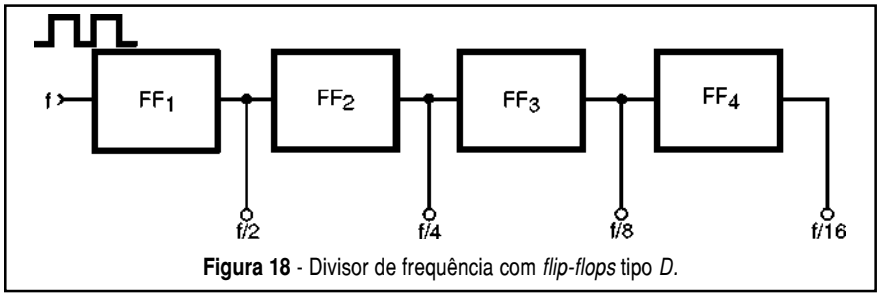


Figura 18 - Divisor de frequência com *flip-flops* tipo *D.*

versos *flip-flops* do tipo T em série, de modo que passando através de cada um, a frequência do sinal de entrada é divida por 2.

Usando 4 *flip-flops,* podemos dividir a frequência por 2, 4, 8 e 16.

Este tipo de divisor de frequência é muito usado, existindo até circuitos integrados que possuem sequências de mais de 10 *flip-flops* ligados desta forma.

Na prática não temos os *flip-flops* tipo D como componentes prontos para uso. Estes *flip-flops* podem ser obtidos a partir de outros e isso será visto no item seguinte.

6.7 - TRANSFORMANDO *FLIP-FLOPS*

Da mesma maneira como podemos obter qualquer função lógica complexa a partir de funções simples, o que foi visto em lições anteriores, também podemos "brincar" com os *flip-flops,* obtendo outros tipos a partir de um tipo básico.

Assim, usando um *flip-flops R-S* ou *J-K* que são comuns e algumas portas lógicas, podemos obter *flip-flops* de outros tipos.

Na **figura 20** temos algumas con-

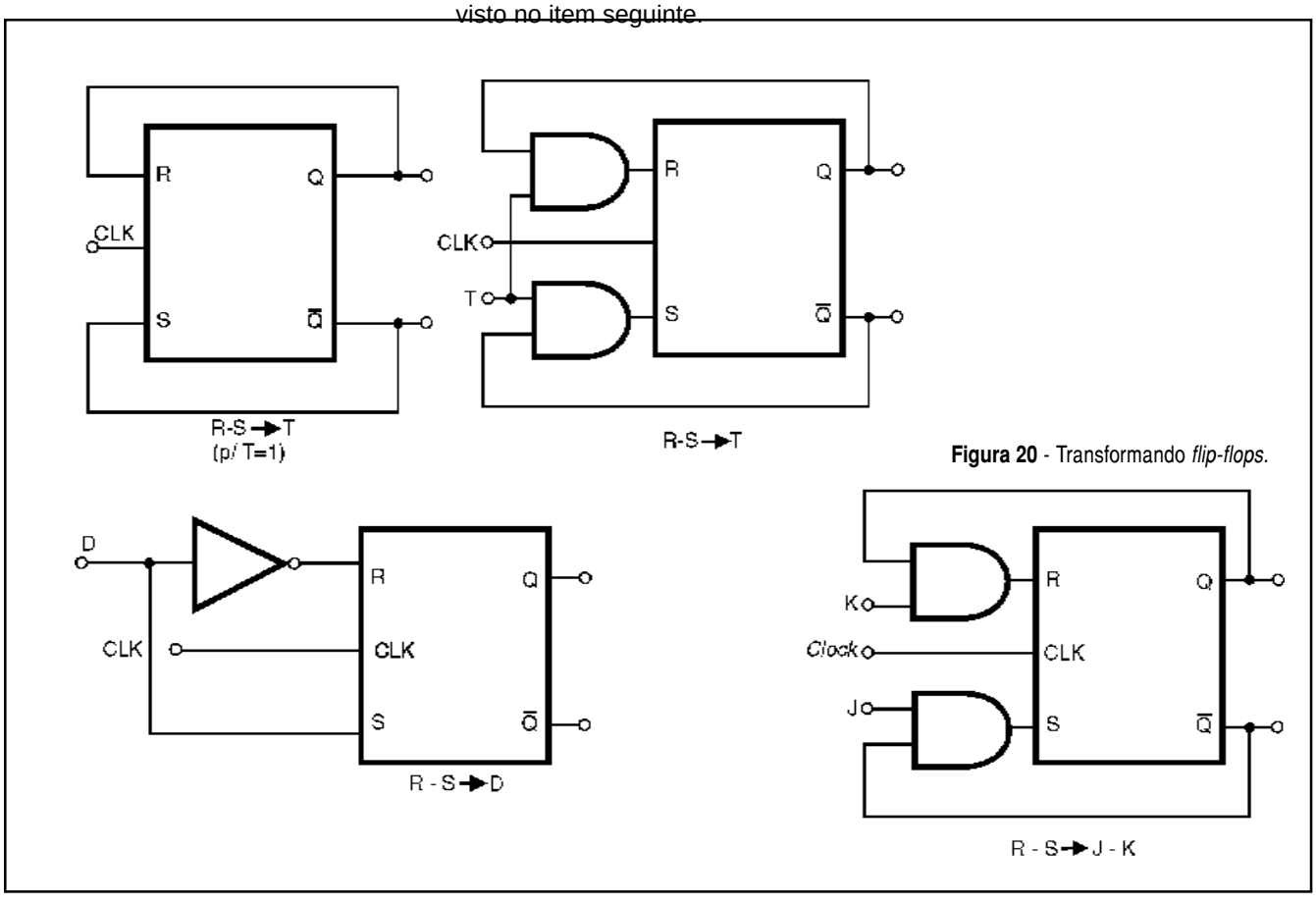


Figura 20 - Transformando *flip-flops.*

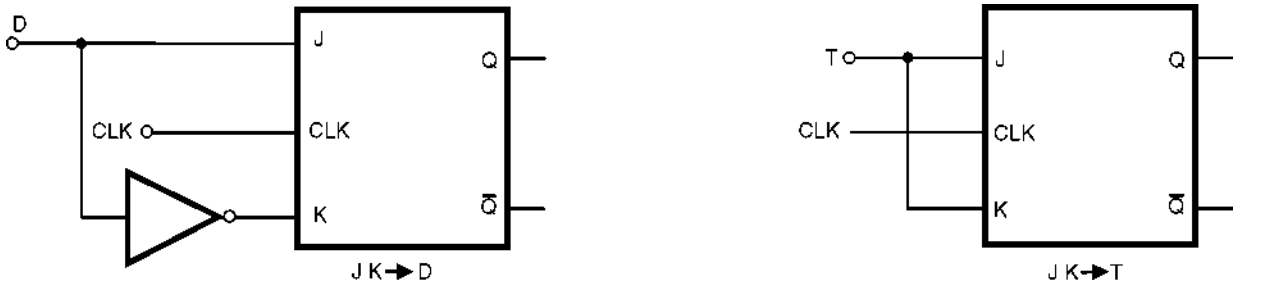


Figura 21 - Transformando FF tipo JK em tipo D e tipo T.

versões que podem ser feitas utilizando-se *flip-flops* do tipo *R-S*.

O modo de funcionamento de cada um pode ser facilmente entendido se o leitor tentar associar as tabelas verdade dos *flip-flops* que foram estudados nesta lição às tabelas verdade das portas agregadas, considerando os sinais de realimentação.

Na **figura 21** temos o modo de obter *flip-flops* tipo *D e T* a partir de *flip-flops* do tipo *J-K*.

Veja que a simples conexão da entrada *K* ao *J* no *flip-flop* do tipo *J-K* o transforma em um *flip-flop* tipo *T*. Esta possibilidade é muito interessante, já que *flip-flops J-K* são disponíveis em tecnologia TTL e CMOS e podem ser usados em circuitos divisores de frequência. Na verdade, já utilizamos esta configuração em diversos projetos práticos que publicamos. Finalmente, temos outras duas transformações importantes de *flip-flops* mostradas na **figura 22**.

No primeiro caso temos uma transformação de um *flip-flop* tipo *D* em *flip-flop* tipo *T,* bastando para isso que a saída complementar /Q seja ligada à entrada *D*, realimentando o circuito.

A segunda transformação, que leva um *flip-flop* tipo *D* a funcionar como tipo *T*, exige o emprego de uma porta AND adicional na realimentação

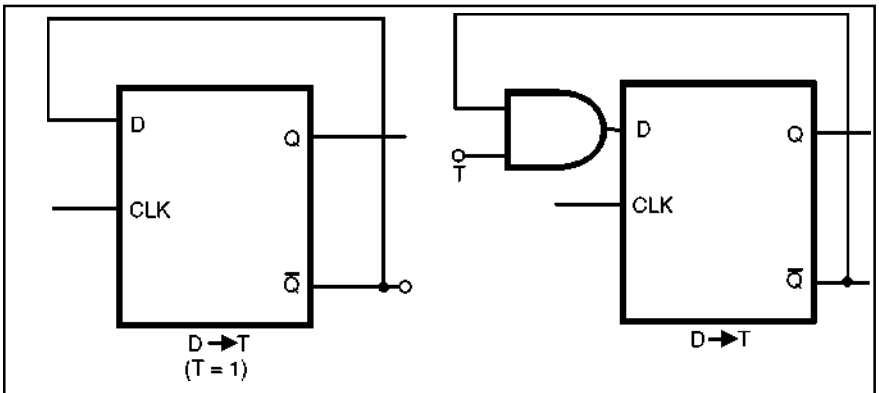


Figura 22 - Outras transformações de *flip-flops*.

do sinal que é retirado da saída complementar /Q.

6.8 - NOS COMPUTADORES

Encontramos os *flip-flops* nos computadores como elementos fundamentais de muitos circuitos.

Uma aplicação é na própria divisão de frequência dos *clocks*. Conforme o leitor sabe, existem setores de um PC que devem operar com velocidades menores que a fornecida pelo *clock* principal. É o caso dos barramentos onde são ligadas as placas de expansão, os modems e as saídas de dados paralela e serial.

Assim, em lugar de usar um *clock* para cada frequência desejada, o que se faz é empregar um *clock* único e dividir sua frequência conforme as exigências de frequências mais baixas, observe a **figura 23**.

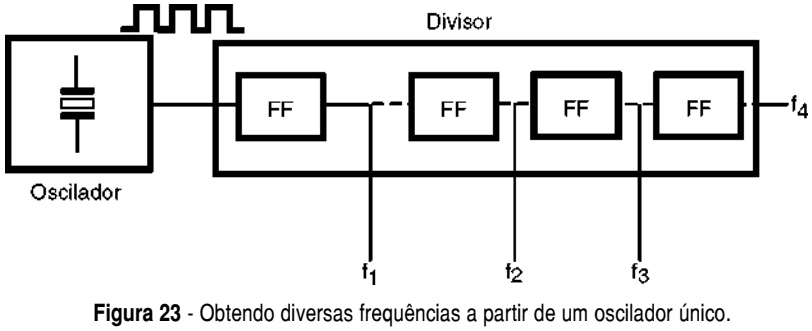


Figura 23 - Obtendo diversas frequências a partir de um oscilador único.

No caso dos computadores, tanto o próprio *clock* como a sequência de *flip-flops* divisores podem ser obtidos num único circuito integrado.

Um ponto importante que deve ser levado em conta e que estudaremos nas lições futuras é a possibilidade de ligar os *flip-flops* em conjunto com outras funções, de modo que a frequência possa ser dividida por qualquer número e não somente por potências de (2,4,8,16,32,64, etc).

Outra aplicação importante é como célula de memória. Oito *flip-flops* ligados lado a lado podem armazenar um *byte* inteiro. Cada *flip-flop* armazena um bit. Existem diversas memórias internas de um PC que nada mais são do que *flip-flops* que podem ser habilitados tanto para a leitura de dados como para introdução (gravação de dados). Existem ainda outras funções importantes implementadas a partir de *flip-flops* e que serão estudadas futuramente.

6.9. OS *FLIP-FLOPS* ANTIGOS

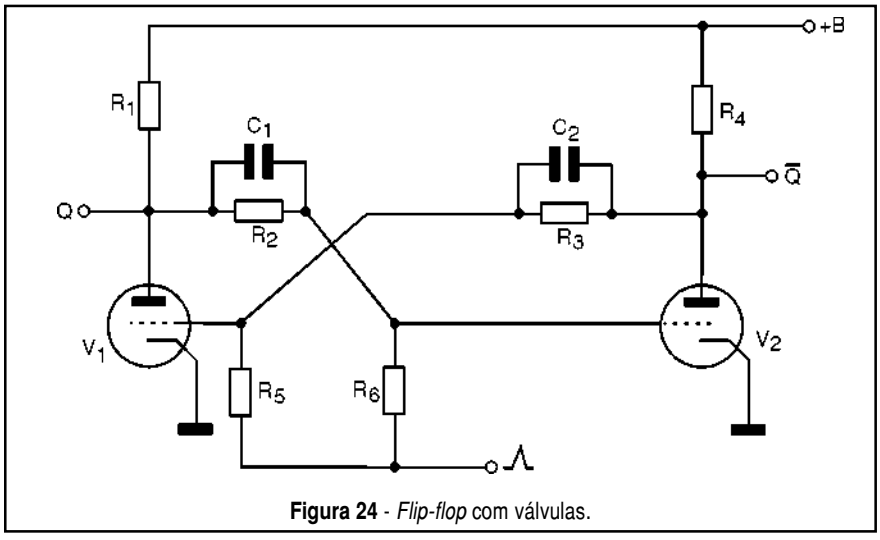


Figura 24 - *Flip-flop* com válvulas.

A configuração do *flip-flop* não é nova. Na verdade, foi em 1919 que dois pesquisadores americanos chamados **Eccles** e **Jordan** apresentaram o primeiro circuito de *flip-flop* usando válvulas, confira na figura 24.

Por este motivo, muitos ainda chamam os *flip-flops* de "Básculas ou Circuitos Eccles-Jordan".

Em 1930, os físicos já usavam estes circuitos ligados em série para dividir a contagem dos pulsos de contadores Geiger de radiação e obterem valores menores mais facilmente totalizados nas pesquisas.

Um contador binário usando uma lâmpada neon ligada às válvulas foi desenvolvido usando estes *flip-flops* em 1940, mas foi somente depois disso que os primeiros computadores digitais passaram a usar estes circuitos de uma forma mais intensa, até o advento do transistor e depois dos circuitos integrados.

QUESTIONÁRIO

1. Os elementos biestáveis de um circuito:

a) Possuem apenas um estado estável

b) Possuem dois estados instáveis

c) Possuem dois estados estáveis

d) Possuem um número indeterminado de estado estáveis

2. Usado como elemento de memória, um *flip-flop* pode armazenar:

a) 1 bit

b) 1 byte

c) meio byte

d) 2 bits

3. Um *flip-flop R-S* "setado" apresenta que níveis lógicos em suas saídas:

a) Q=0 e /Q=0

b) Q=0 e /Q=1

c) Q=1 e /Q=0

d) Q=1 e /Q=1

4. Os *flip-flops "negative edge-triggered"* mudam de estado quando:

a) O pulso *clock* vai do nível baixo para o nível alto

b) O pulso de *clock* vai do nível alto para o nível baixo

c) O pulso de *clock* estabiliza no nível baixo

d) O pulso de *clock* estabiliza no nível alto

5. Para que um *flip-flop J-K* Mestre escravo tenha a condição *"toggle",* quais são os níveis lógicos que devem ser colocados na entrada J e K?

a) J=0 e K=0

b) J=0 e K=1

c) J=1 e K=0

d) J=1 e K=1

6. Quatro *flip-flops* do tipo *T* ligados um após o outro (em série) recebem uma frequência de 1 600 Hz na sua entrada. Qual é a frequência obtida na saída do último *flip-flop?* (o sinal deve ser retangular).

a) 800 Hz

b) 400 Hz

c) 200 Hz

d) 100 Hz

Respostas:

1-C, 2-A, 3-C, 4-B, 5-D. 6-D

LIÇÃO 7

OS *FLIP-FLOPS* E FUNÇÕES LÓGICAS EM CIRCUITOS INTEGRADOS

Na lição anterior aprendemos como funcionam os principais tipos de *flip-flops,* verificando que, dependendo dos recursos que cada um possua, podem ser empregados de diversas formas. Também vimos as entradas que estes dispositivos podem conter para melhorar seu desempenho em determinadas aplicações, como por exemplo, nos computadores. Estudamos ainda nas primeiras lições do curso as funções lógicas usadas em diversos circuitos. Tudo isso nos leva à necessidade de contarmos com estas funções na forma de circuitos integrados. De fato, existem muitos circuitos integrados TTL e CMOS contendo *flip-flops* dos tipos estudados e todas as funções lógicas (portas e inversores e amplificadores) e será justamente deles que falaremos nesta lição.

6.1 - OS *FLIP-FLOPS* TTL

A família de circuitos integrados digitais TTL conta com uma grande quantidade de *flip-flops* usados numa infinidade de aplicações práticas.

A diferença de cada tipo de circuito integrado não está apenas no tipo de *flip-flop* que contém como também nos seus recursos e na sua quantidade. Também devemos observar que um fator importante na escolha de um *flip-flop* para uma determinada aplicação é a sua velocidade. Para as diversas famílias TTL podemos especificar as máximas velocidades dos seus *flip-flops* da seguinte maneira:

Standard (74) - 35 MHz
Low Power (74L) - 3 MHz
Low Power Shottky (74LS) - 45 MHz
High Speed (74H) - 50 MHz
(74S) - 125 MHz

É importante observar que para os *flip-flops* TTL é preciso alguns cuidados, como por exemplo, manter sempre as entradas *CLEAR* e *PRESET* em níveis definidos. Deixando estas entradas abertas, podem ocorrer instabilidades de funcionamento.

O nível em que elas devem ser deixadas, ou seja, sua conexão no Vcc ou 0 V depende da aplicação.

a) 7473 - DUPLO *FLIP-FLOP* J-K COM *CLEAR*

Num único invólucro de 14 pinos *Dual in Line* temos 2 *flip-flops* do tipo J-K com entrada de *Clear*. A pinagem deste circuito integrado é mostrada na **figura 1**.

Os *flip-flops* são sensíveis ao nível de *clock (Level Triggered)* com entrada de *Clear* assíncrono. O funcionamento dos *flip-flops* deste circuito integrado pode ser melhor entendido pela tabela verdade da **figura 2.**

Nesta tabela, o símbolo com a forma de um pulso de sinal representa um pulso de *clock* positivo aplicado à entrada correspondente.

Observe que quando J e K estão aterradas, o *clock* não tem efeito sobre o circuito. Na operação normal, a entrada *Clear* deve ser mantida no nível alto. Se a entrada *Clear* for aterrada, o *flip-flop* resseta. A frequência máxima de operação destes *flip-flops* é de 20 MHz com um consumo por circuito integrado da ordem de 20 mA.

b) 7474 - DUPLO *FLIP-FLOP* TIPO D COM *PRESET* E *CLEAR*

Os *flip-flops* contidos no invólucro DIL de 14 pinos disparam com a transição positiva do sinal de *clock (Positive-Edge Triggered)*. A pinagem deste circuito integrado é mostrada na **figura 3**.

A tabela verdade que apresenta o funcionamento dos *flip-flops* deste

$\overline{CLR}$	CLK	J	K	Qn+1	$\overline{Qn+1}$
Ø	X		Ø	Ø	1
1	⊓	Ø	Ø	Qn	$\overline{Qn}$
1	⊓	Ø	1	Ø	1
1	⊓	1	Ø	1	Ø
1	⊓	1	1	$\overline{Qn}$	Qn

X = não importa

Figura 2 - Tabela verdade que descreve o funcionamento do 7473.

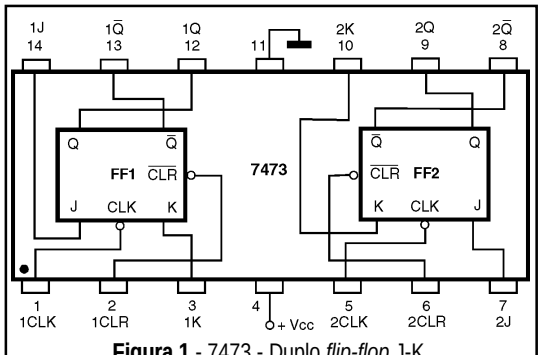


Figura 1 - 7473 - Duplo *flip-flop* J-K.

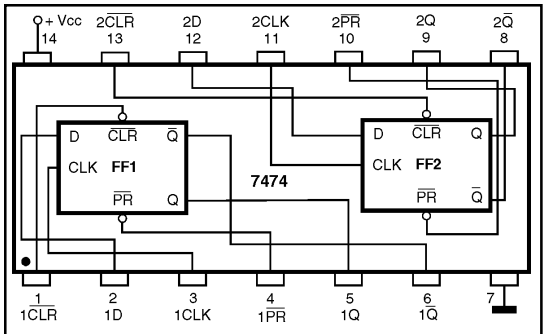


Figura 3 - Duplo *flip-flop* D - 7474.

$\overline{CLR}$	$\overline{PR}$	D	CLK	Qn+1	$\overline{Q}$n+1	
Ø	Ø	X	X	1	1	(*)
Ø	1	X	X	Ø	1	
1	Ø	X	X	1	Ø	
1	1	Ø	↑	Ø	1	
1	1	1	↑	1	Ø	

(*) = não permitido
X = não importa

Figura 4 - Tabela verdade que descreve o funcionamento do 7474.

circuito integrado é dada na **figura 4**. Pela tabela, concluímos que a condição em que as entradas *Clear* e *Preset* estão simultaneamente ativas não deve ser usada, pois teremos uma condição não permitida para os *flip-flops*.

A frequência máxima de operação deste circuito integrado é de 25 MHz e o consumo é da ordem de 17 mA.

EN	D	Qn+1	$\overline{Q}$n+1
Ø	X	Qn	$\overline{Q}$n
1	Ø	Ø	1
1	1	1	Ø

Figura 6 - Tabela verdade para o 7475.

c) 7475 - QUATRO *LATCHES* TIPO D

Os *latches* são como chaves que armazenam uma informação digital presente em sua entrada. A aplicação mais comum é justamente como memória, cada circuito integrado 7475 pode armazenar 4 bits de informação.

Na **figura 5** temos a pinagem deste circuito integrado. Quando o circuito é habilitado, o que é conseguido levando a linha "*ENABLE*" ao nível alto, as saídas Q e /Q seguem a entrada D. O latch é do tipo "transparente", logo, se as entradas forem modificadas, as saídas também se alterarão.

Quando a entrada "*ENABLE*" é levada ao nível baixo, as saídas não respondem aos sinais de entrada D.

Veja que o *LATCH* armazena a informação que estava na entrada D imediatamente antes da ocorrência de uma transição do nível alto para o nível baixo da linha de habilitação (Nível 1 para o nível 0).

O funcionamento de cada *flip-flop* do 7475 pode ser colocado na tabela verdade da **figura 6**.

Este circuito integrado não serve para aplicações onde se deseja mudanças de estado a cada pulso de *clock*. Dizemos que este circuito não pode ser usado como um registrador de deslocamento *(shift-register)* que será estudado nas próximas lições.

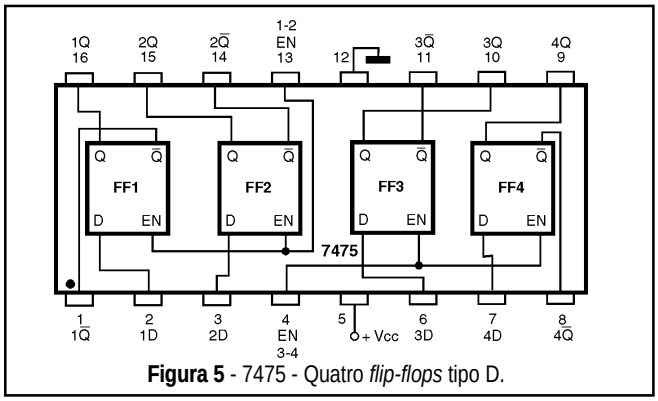


Figura 5 - 7475 - Quatro *flip-flops* tipo D.

O tempo de propagação do sinal é da ordem de 24 ns e o consumo típico por circuito integrado é de 32 mA.

d) 7476 - DOIS *FLIP-FLOPS* J-K COM *PRESET* E CLEAR

Os dois *flip-flops* deste circuito integrado têm funcionamento independente e disparam com nível do sinal de *clock (level triggered)*.

O invólucro é DIL de 16 pinos, veja a **figura 7**. O funcionamento de cada um dos *flip-flops* pode ser melhor analisado através da tabela verdade da **figura 8**. Observe o símbolo adotado para representar um pulso de *clock*.

Da mesma forma que nos demais circuitos integrados desta série, as entradas *CLEAR E PRESET* devem ser mantidas em níveis lógicos definidos, para que não ocorra o funcionamento errático do circuito.

Também observamos pela tabela verdade que não se pode ativar as duas entradas de *CLOCK* E *CLEAR* ao mesmo tempo, pois isso levaria os *flip-flops* a uma condição não permitida.

$\overline{CLR}$	$\overline{PR}$	J	K	CLK	Qn+1	$\overline{Q}$n+1	
Ø	Ø	X	X	X	1	1	(*)
Ø	1	X	X	X	Ø	1	
1	Ø	X	X	X	1	Ø	
1	1	Ø	Ø	⊓	Qn	$\overline{Q}$n	
1	1	Ø	1	⊓	Ø	1	
1	1	1	Ø	⊓	1	Ø	
1	1	1	1	⊓	$\overline{Q}$n	Qn	

(*) não permitido X = não importa

Figura 8 - Tabela verdade do 7476.

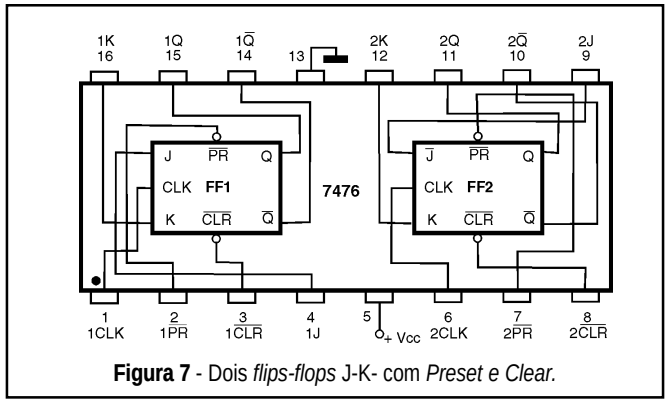


Figura 7 - Dois *flips-flops* J-K- com *Preset e Clear.*

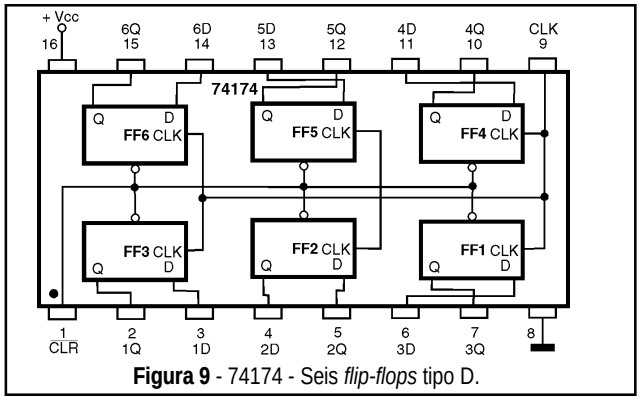


Figura 9 - 74174 - Seis *flip-flops* tipo D.

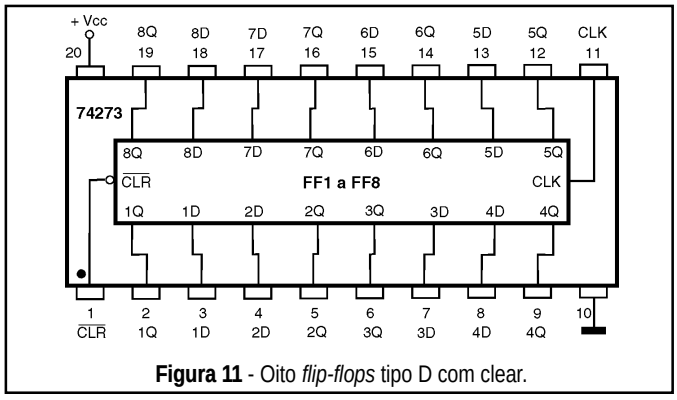


Figura 11 - Oito *flip-flops* tipo D com clear.

$\overline{CLR}$	D	CLK	$\overline{Q}n+1$
Ø	X	X	Ø
1	Ø	↑	Ø
1	1	↑	1
1	X	Ø	Qn

X = não importa

Figura 10 - Tabela verdade para os *flip-flops* do 74174.

Um ponto interessante que deve ser observado neste circuito integrado é a pinagem diferente, já que normalmente nos circuitos desta série a alimentação positiva é sempre nos pino 14 ou 16 e a negativa no pino 7 ou 8, quando os invólucros são de 14 ou 16 pinos. A frequência máxima de operação destes *flip-flops* para a série normal é de 20 MHz e o consumo de 20 mA.

e) 74174 - SEIS *FLIP-FLOPS* TIPO D COM *CLEAR*

Este circuito integrado contém seis *flip-flops* do tipo D que são disparados na transição positiva do sinal de *clock*. A entrada de *CLEAR* é comum a todos os *flip-flops*. O invólucro é de 16 pinos com a identificação feita segundo mostra a **figura 9**.

A tabela verdade que descreve o funcionamento de cada *flip-flop* deste circuito integrado está na **figura 10**.

Observe que nestes *flip-flops* temos acesso a apenas uma das saídas, assim, as saídas complementares não podem ser usadas.

A frequência máxima dos *flip-flops* da série *stardard* (comum) é de 35 MHz com um consumo típico de 45 mA por circuito integrado.

f) 74273 - OITO *FLIP-FLOPS* TIPO D COM *CLEAR*

Este circuito é semelhante ao anterior com a diferença de que existem oito em lugar de seis *flip-flops* tipo D. Cada um dos *flip-flop* pode operar com um bit, assim, esta configuração se torna ideal para aplicações em computadores, pois opera com 8 bits que correspondem a um byte.

A pinagem do circuito integrado 74273 é mostrada na **figura 11**.

A tabela verdade para cada *flip-flop* é a mesma do circuito integrado anterior apresentada na **figura 10**.

A frequência máxima de operação para os circuitos integrados deste tipo da série normal é de 30 MHz com um consumo de 62 mA para cada um.

Veja que o invólucro usado é *Dual In Line* de 20 pinos e que a entrada de CLEAR é comum a todos os integrados. Também observamos que não existe acesso às saídas complementares dos *flip-flops*.

g) 74LS373 - *LATCH* OCTAL TRANSPARENTE TIPO D

O tipo LS é importante neste caso, já que se trata de circuito compatível com as portas paralelas dos computadores e portanto, pode ser excitado diretamente pelos níveis lógicos existentes num PC.

Uma vez que o circuito integrado 74LS373 contém 8 *latches* com saída *tri-state,* ele pode ser usado para trabalhar com um byte inteiro, sem problemas.

A pinagem deste circuito integrado é mostrada na **figura 12**.

Quando a entrada /OE está no nível alto (1), as saídas de todos os *flip-flops* vão para o estado de alta impedância. Isso significa que estas saídas podem ser ligadas a um barramento comum a outros circuitos integrados, sem o problema de conflitos que possam carregar os circuitos causando problemas de funcionamento, conforme já estudamos nas lições iniciais deste curso.

Quando a entrada /OE está ativada, o que é feito levando-a ao nível baixo (0), o estado das saídas vai depender da entrada EL. Se EL estiver no nível alto (1), o *latch* estará aberto "transparente". O que estiver na entrada D vai passar pelo circuito e aparecer na saída Q.

Se EL estiver no nível baixo (0), a saída Q não mais responde ao que ocorre nas entradas D. Nestas condições

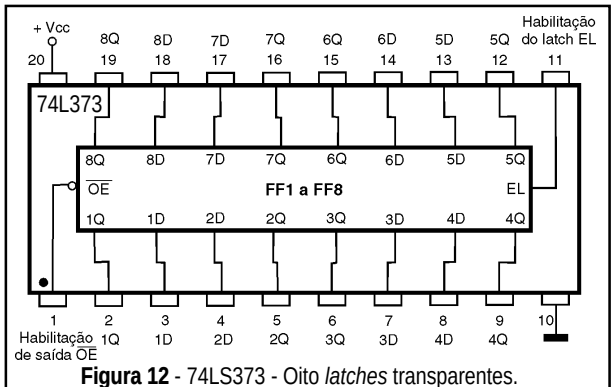


Figura 12 - 74LS373 - Oito *latches* transparentes.

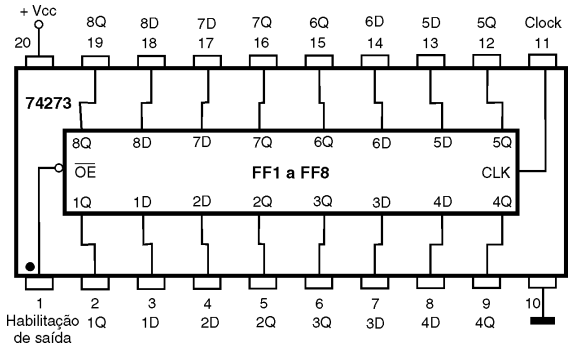


Figura 13 - 74LS374 - Oito *flip-flops* tipo D.

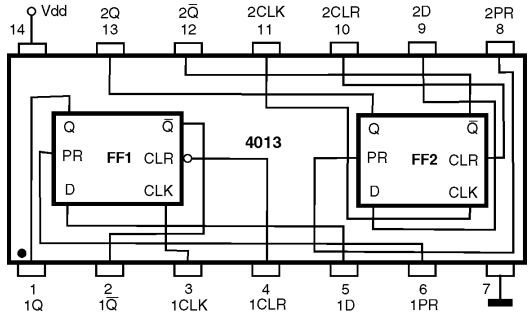


Figura 15 - 4013 - Dois *flip-flops* tipo D.

dizemos que o *latch* está fechado e a saída Q será o conteúdo das entradas D que foi armazenado imediatamente antes da transição das entradas EL do nível alto para o nível baixo. Em outras palavras, podemos dizer que os *flip-flops* são gatilhados na transição negativa da entrada EL.

Observe a condição de alta impedância obtida com /OE no nível alto. A frequência máxima de operação para os *latches* deste circuito integrado é de 50 MHz com um consumo de 24 mA.

$\overline{OE}$	D	CLK	Qn+1
1	X	X	alta-Z
Ø	Ø	↑	Ø
Ø	1	↑	1
Ø	X	1	Qn

Figura 14 - Tabela verdade de cada *flip-flop* do 74LS374.

CLR	$\overline{PR}$	D	CLK	Qn+1	$\overline{Q}$n+1
Ø	Ø	Ø	↑	Ø	1
Ø	Ø	1	↑	1	Ø
1	Ø	X	X	Ø	1
Ø	1	X	X	1	Ø
1	1	X	X	1	1

x = não importa

Figura 16 - Tabela verdade para os *flip-flops* do 4013.

h) 74LS374 - OITO *FLIP-FLOPS* TIPO D COM SAÍDAS *TRI-STATE*

Temos neste circuito integrado TTL em invólucro DIL de 20 pinos 8 *flip-flops* do tipo D que são disparados na transição positiva do sinal de *clock*. As saídas são *tri-state* e a pinagem é mostrada na **figura 13**.

Quando a entrada /OE está no nível alto, as saídas de todos os *flip-flops* vão para o estado de alta impedância. Veja que neste circuito integrado também não temos acesso às saídas complementares dos *flip-flops*. A tabela verdade que descreve o funcionamento de cada um dos *flip-flops* é mostrada na **figura 14**.

A frequência máxima de operação deste circuito integrado é de 50 MHz com um consumo típico de 27 mA.

7.2 - OS *FLIP-FLOPS* CMOS

Temos diversos *flip-flops* disponíveis na família CMOS que serão analisados a seguir. Uma recomendação importante relativa ao uso destes *flip-flops*, assim como das demais funções CMOS, é que as entradas não usadas, pela sua sensibilidade devida à alta impedância, nunca devem ser mantidas abertas.

Nos *flip-flops* CMOS, diferentemente dos TTL, as entradas assíncronas são ativadas no nível alto, o que significa que devem ser mantidas no nível baixo para a operação normal.

a) 4013 - DOIS *FLIP-FLOPS* TIPO D COM *PRESET* E *CLEAR*

Os dois *flip-flops* contidos neste circuito integrado são disparados na transição positiva do sinal de *clock*.

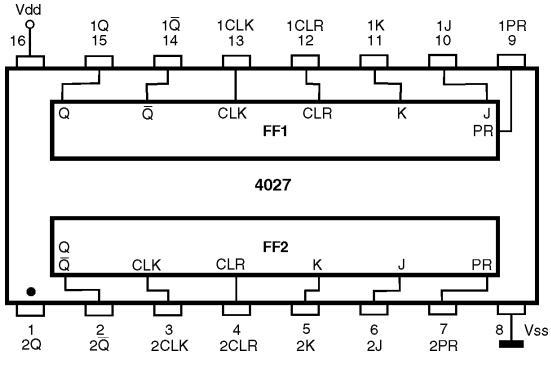


Figura 17 - 4027 - Dois *flip-flops* J-K.

$\overline{CLR}$	$\overline{PR}$	J	K	CLK	Qn+1	$\overline{Q}$n+1	
Ø	Ø	Ø	Ø	↑	Qn	$\overline{Q}$n	
Ø	Ø	Ø	1	↑	Ø	1	
Ø	Ø	1	Ø	↑	1	Ø	
Ø	Ø	1	1	↑	$\overline{Q}$n	Qn	
1	Ø	X	X	X	Ø	1	
Ø	1	X	X	X	1	Ø	
1	1	X	X	X	1	1	(*)

(*) não permitido X = não importa

Figura 18 - Tabela verdade para os *flip-flops* do 4027.

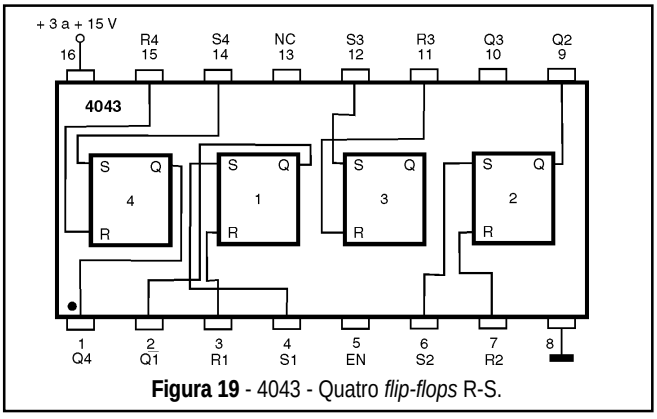


Figura 19 - 4043 - Quatro *flip-flops* R-S.

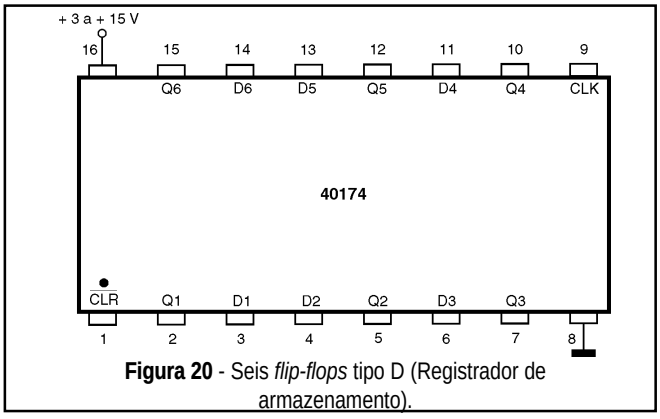


Figura 20 - Seis *flip-flops* tipo D (Registrador de armazenamento).

O invólucro é o DIL de 14 pinos da **figura 15**.

A tabela verdade para este circuito integrado está na **figura 16**.

Pela tabela verdade vemos que as entradas *CLEAR* E *PRESET* são ativas no nível alto, mas que somente uma delas pode estar nesta condição de cada vez. Se as duas entradas *PRESET* e *CLEAR* forem colocadas no nível alto ao mesmo tempo, o *flip-flop* vai para uma condição não permitida.

A informação presente na entrada D é transferida para a saída, quando as entradas assíncronas *PRESET* E *CLEAR* estão inativas.

É importante observar que a velocidade de operação dos circuitos CMOS depende da tensão de alimentação, como já estudamos nas lições anteriores.

Nos manuais de circuitos integrados CMOS os leitores poderão encontrar tabelas que trazem os diversos tempos de propagação dos sinais e as frequências de operação em função desta tensão de alimentação. Podemos dizer apenas que, para uma alimentação de 10 V, a frequência máxima de *clock* será de 7 MHz.

b) 4027 - DUPLO *FLIP-FLOP* J-K COM *PRESET* E *CLEAR*

Neste circuito integrado encontramos dois *flip-flops* tipo J-K com entradas de *PRESET* E *CLEAR*. O invólucro é DIL de 16 pinos, mostrado na **figura 17**.

Nos *flip-flops*, as entradas *PRESET* e *CLEAR* são independentes. A tabela verdade para os *flip-flops* é mostrada na **figura 18**.

Observe que temos acesso tanto as saídas normais como complementares de cada um dos *flip-flops* e que as saídas *CLEAR* E *PRESET* estão ativas no nível alto. No entanto, como nos demais *flip-flops*, estas saídas não podem ser ativadas ao mesmo tempo, pois levariam os *flip-flops* a uma condição não permitida.

Como no caso anterior, a frequência depende da tensão de alimentação. Para uma tensão de alimentação de 10 V, a frequência máxima de operação é da ordem de 8 MHz.

c) 4043 - QUATRO *FLIP-FLOPS S* R-S (Lógica NOR)

Este circuito integrado contém quatro *flip-flops* R-S independentes com saídas *tri-state*. O invólucro DIL de 16 pinos é mostrado na **figura 19**.

Em cada um dos *flip-flops,* as entradas *SET* e *RESET* podem normalmente ficar no nível baixo. Se a entrada *SET* for levada ao nível alto, a saída irá e permanecerá no nível alto. Se a entrada *RESET* for levada ao nível alto a saída irá e permanecerá no nível baixo. As duas saídas não podem ser levadas ao mesmo tempo ao nível alto, pois isso representa um estado não permitido.

As saídas vão ao estado de alta impedância com a entrada EN (habilitação ou *ENABLE*) levada ao nível baixo. Quando o nível da entrada EN é alto, as saídas são conectadas aos *flip-flops,* transferindo seus estados para os circuitos externos.

Como estes circuitos não usam *clocks,* eles não devem ser ligados em cascata para formar contadores ou *shift-registers*.

d) 40174 - SEIS *FLIP-FLOPS* TIPO D

Este circuito integrado contém seis *flip-flops* tipo D disparados pela transição positiva do sinal de *clock*. Apenas uma das saídas de cada *flip-flop* é acessível externamente e o *CLEAR* é comum a todos eles. O invólucro é DIL de 16 pinos com a pinagem mostrada na **figura 20**. Todos os *flip-flops* são controlados por uma entrada comum de *clock*. A tabela verdade para os *flip-flops* deste circuito integrado é mostrada na **figura 21**.

CLR	CLK	D	Q_{n+1}	$\overline{Q}_{n+1}$
Ø	X	X	Ø	1
1	↑	1	1	Ø
1	↑	Ø	Ø	1
↑	1	X	Q_n	$\overline{Q}_n$
1	Ø	X	Q_n	$\overline{Q}_n$

x = não importa

Figura 21 - Tabela verdade para os *flip-flops* do 40174.

e) 40175 - QUATRO *FLIP-FLOPS* TIPO D

Trata-se de um circuito integrado que contém quatro *flip-flops* semelhantes ao anterior com a diferença de que as duas saídas (normal e complementar) podem ser acessadas.

O invólucro deste circuito integrado é apresentado na **figura 22**.

A tabela verdade para os circuitos integrados é a mesma do 40174. Para

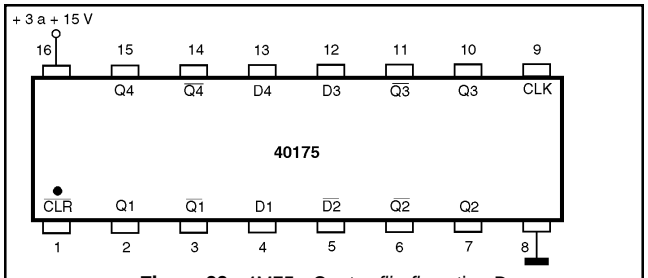


Figura 22 - 4M75 - Quatro *flip-flops* tipo D (Registrador de armazenamento).

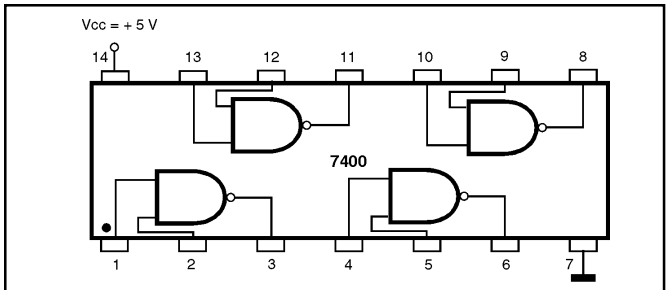


Figura 23 - Quatro portas NAND de duas entradas.

uma alimentação de 10 V, a frequência máxima de *clock* é de 10 MHz.

7.3 - FUNÇÕES LÓGICAS TTL

Podemos contar com uma boa quantidade de circuitos integrados contendo as principais funções lógicas em tecnologia TTL. Damos a seguir alguns dos mais importantes, já que para obter informações sobre a totalidade será interessante contar com um manual TTL.

a) 7400 - Quatro Portas NAND de duas entradas

Num invólucro DIL de 14 pinos contamos com quatro portas NAND de duas entradas de funcionamento independente.

Veja na **figura 23** a pinagem deste circuito integrado.

O consumo médio por circuito integrado é da ordem de 12 mA.

b) 7402 - Quatro Portas NOR de duas entradas

Este circuito integrado em invólucro DIL de 14 pinos tem a pinagem mostrada na **figura 24** e cada unidade exige uma corrente de 12 mA.

c) 7404 - Seis Inversores *(Hex Inverter)*

Os seis inversores deste circuito integrado podem ser usados de forma independente. A pinagem está na **figura 25**.

d) 7408 - Quatro Portas AND de duas entradas

Este circuito integrado tem a pinagem da **figura 26** e cada unidade exige uma corrente de 16 mA.

e) 7410 - Três portas NAND de três entradas

Cada uma das três portas NAND deste circuito integrado pode ser usada de forma independente. A corrente exigida pelo circuito é de 6 mA.

f) 7420 - Duas portas NAND de quatro entradas

Este circuito integrado contém duas portas NAND que podem ser usadas de forma independente. A pinagem é mostrada na **figura 28**.

O consumo por unidade é de aproximadamente 4 mA.

g) 7432 - Quatro portas OR de duas entradas

As portas OR deste circuito integrado podem ser usadas de modo independente e a corrente total exigida é da ordem de 19 mA. A pinagem está na **figura 29**.

h) 7486 - Quatro Portas *OR-Exclusivo*

As portas OU-exclusivo ou Exclusive OR deste circuito integrado podem ser usadas de forma independente. O consumo é de 30 mA e a pinagem está na figura 30.

7.4 - FUNÇÕES LÓGICAS CMOS

Também podemos contar com uma boa quantidade de circuitos integrados CMOS contendo funções lógicas. Evidentemente, não temos espaço para colocar todas estas funções nesta lição, assim recomendamos ao leitor que adquira um manual

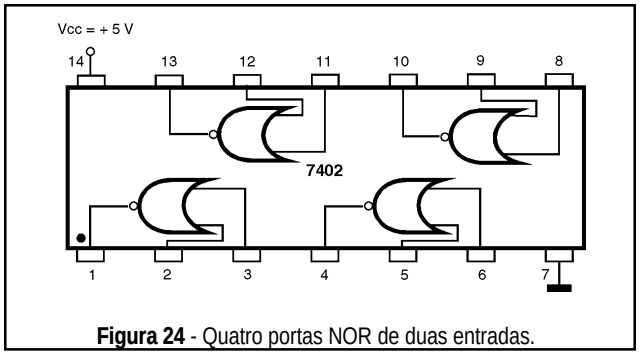


Figura 24 - Quatro portas NOR de duas entradas.

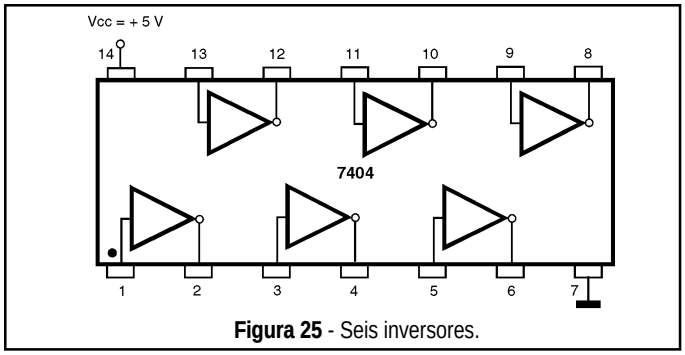


Figura 25 - Seis inversores.

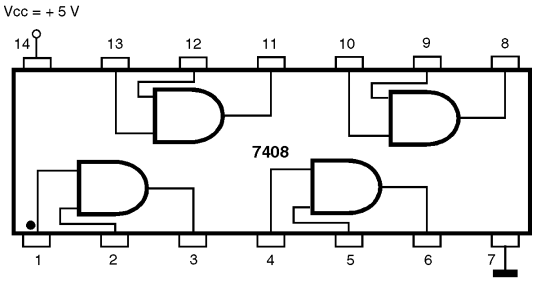


Figura 26 - Quatro portas AND de duas entradas.

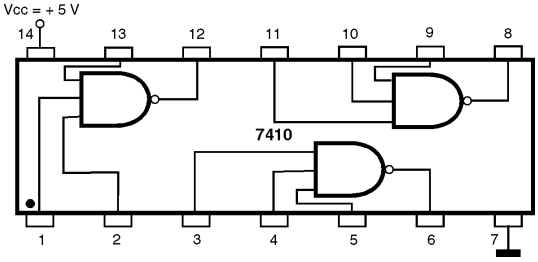


Figura 27 - Três portas NAND de três entradas.

CMOS. Daremos a seguir algumas das mais usadas.

a) 4001 - Quatro Portas NOR de duas entradas

Este circuito integrado contém quatro portas NOR em invólucro DIL de 14 pinos com a pinagem mostrada na **figura 31**.

O consumo por circuito integrado é da ordem de 10 nW.

b) 4011 - Quatro portas NAND de duas entradas

Em invólucro DIL de 14 pinos encontramos quatro portas NOR de duas entradas de funcionamento independente. O invólucro com a identificação dos terminais é mostrado na **figura 32**.

c) 4012 - Duas portas NAND de quatro entradas

As quatro portas NOR de duas entradas deste circuito integrado podem ser usadas de forma independente. A identificação dos terminais deste circuito integrado está na **figura 33**.

d) 4023 - Três portas NAND de três entradas

As três portas NAND deste circuito integrado podem ser usadas de maneira independente. A pinagem é mostrada na **figura 34**.

e) 4025 - Três portas NOR de três entradas

Encontramos neste circuito integrado três funções NOR que podem ser usadas de forma independente. A pinagem é mostrada na **figura 35**.

7.5 - A FUNÇÃO *TRI-STATE* EXPANSÍVEL DO 4048

O circuito integrado 4048 tem características muito interessantes para projetos CMOS envolvendo funções lógicas. Conforme estudamos, usando combinações apropriadas de funções simples, é possível simular qualquer outra função mais complexa. É justamente isso que faz o 4048 que tem a pinagem mostrada na **figura 36**.

Este circuito possui 8 entradas, uma saída e três entradas de "programação".

Dependendo dos níveis lógicos aplicados nestas entradas de programação, o circuito se comporta como funções NOR, OR, NAND ou AND com 8 entradas ou ainda de forma combinada, realizando ao mesmo tempo funções de portas OR e AND cada um de 4 entradas e outras que são mostradas na **figura 37**.

Assim, por exemplo, se colocarmos todas as três entradas de programação no nível alto (Ka,Kb e Kc= 111), o circuito se comporta como duas portas AND de quatro entradas ligadas a uma porta OR de duas entradas.

Veja então que esta interessante função pode servir de "coringa" em muitos projetos, pois consegue simular a operação de diversas combinações de outros circuitos integrados CMOS.

Internamente, o 4048 é bastante complexo contendo 32 funções independentes programadas pelos níveis lógicos aplicados às entradas correspondentes.

QUESTIONÁRIO

1. Qual é o conjunto de funções que o leitor provavelmente não encontrará na forma de um circuito

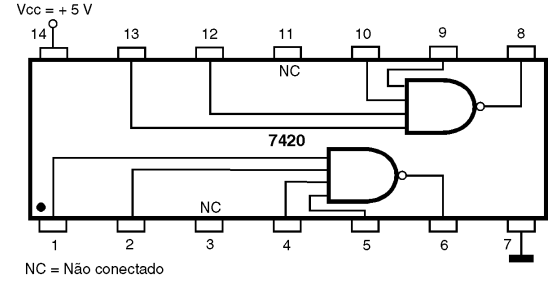


Figura 28 - Duas portas NAND de quatro entradas.

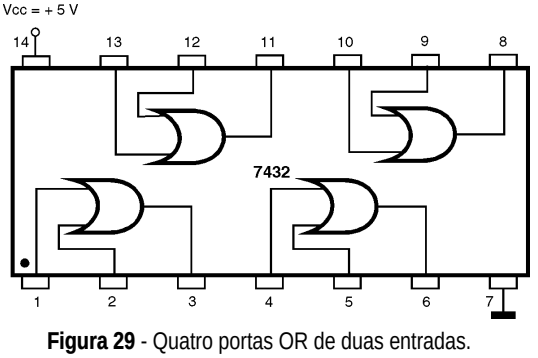


Figura 29 - Quatro portas OR de duas entradas.

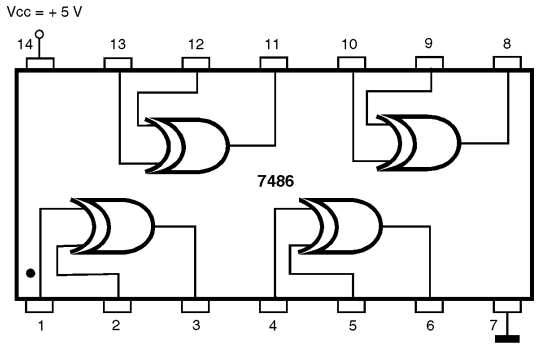


Figura 30 - Quatro portas Exclusive-OR de duas entradas.

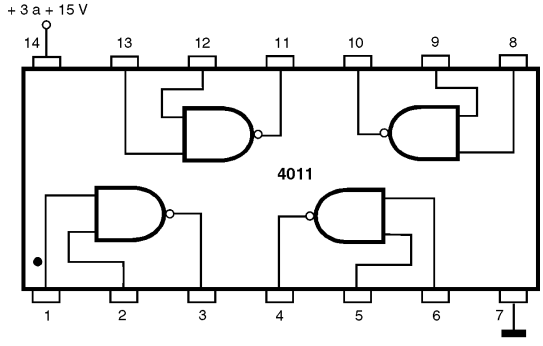


Figura 32 - 4011 - Quatro portas NAND de duas entradas.

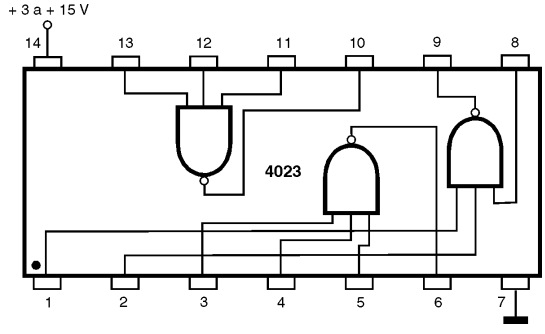


Figura 34 - 4023 - Três portas NAND de três entradas.

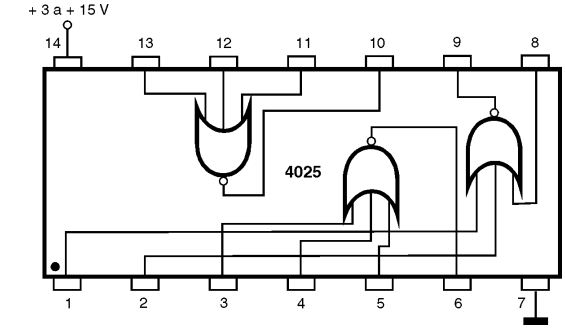


Figura 35 - 4025 - Três portas NOR de três entradas.

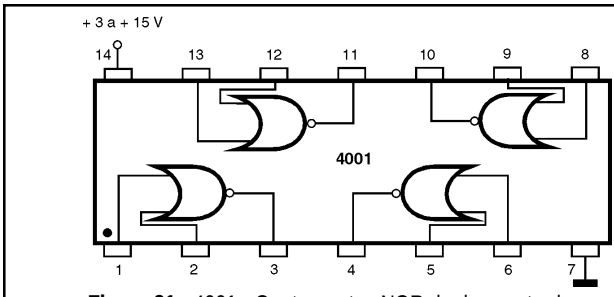


Figura 31 - 4001 - Quatro portas NOR de duas entradas.

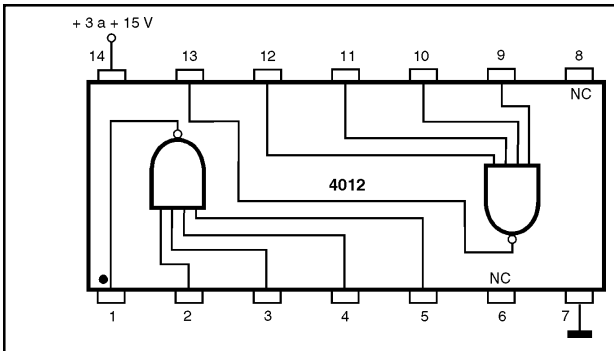


Figura 33 - 4012 - Duas portas NAND de quatro entradas.

integrado TTL ou CMOS?
a) Seis portas AND de 3 entradas
b) Seis inversores
c) Quatro portas AND de duas entradas
d) Quatro portas Exclusive OR

2. As quatro portas NAND de um circuito integrado TTL 7400 têm:
a) Alimentação independente
b) Quatro entradas
c) Funcionamento independente
d) *Reset* comum

3. Os *flip-flops* do circuito integrado 4027 são:
a) Do tipo R-S b) Do tipo D
c) Do tipo J-K d) Do tipo T

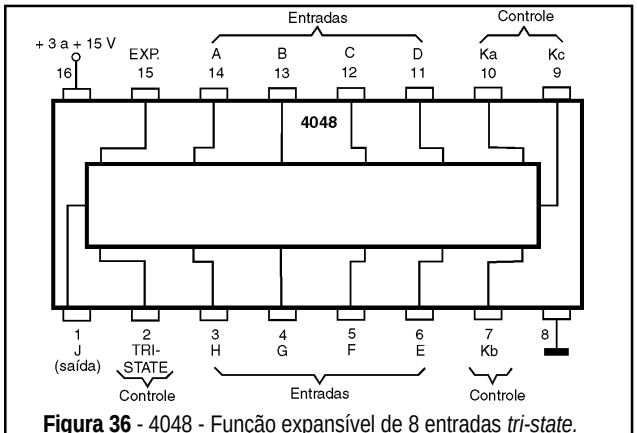


Figura 36 - 4048 - Função expansível de 8 entradas *tri-state*.

4. Um "*latch*" como o circuito TTL 7475 é usado para:
a) Contagem binária
b) Divisão de frequência
c) Operação como porta AND
d) Armazenamento de informação digital

5. Qual é a condição proibida nos f*lip-flops* CMOS e TTL?
a) Entradas J e K ligadas em paralelo
b) *Preset* e *Clear* ao mesmo tempo ativos
c) Preset e Clear ao mesmo tempo desativados
d) Saídas ligadas às entradas D ou *Clear* ■

Respostas:
1-a, 2-c, 3-c, 4-d, 5-b

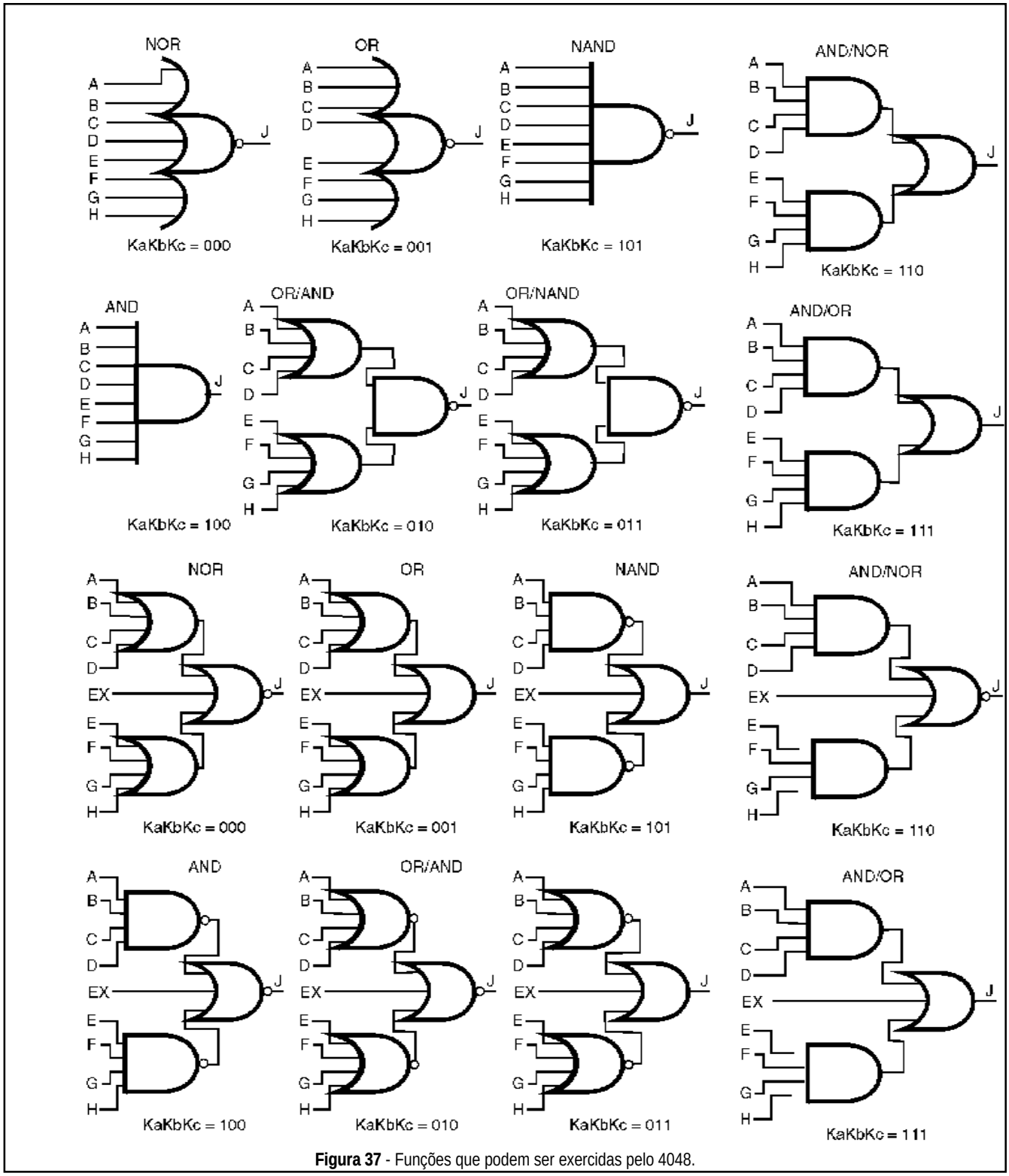


Figura 37 - Funções que podem ser exercidas pelo 4048.

LIÇÃO 8

OS MULTIVIBRADORES ASTÁVEIS E MONOESTÁVEIS

Na lição anterior aprendemos como funcionam os principais tipos de *flip-flops,* verificando que dependendo dos recursos de cada um, eles podem ser empregados de diversas formas. Também vimos as entradas que estes dispositivos podem conter para melhorar seu desempenho em determinadas aplicações, como por exemplo, nos computadores. Vimos também que os *flip-flops* são usados como divisores de frequência ou células de memória. Tudo isso nos leva à necessidade de contar com esta função na forma de circuitos integrados. De fato, existem muitos circuitos integrados TTL e CMOS contendo *flip-flops* dos tipos estudados e será justamente deles que falaremos nesta lição. Também enfocaremos algumas configurações que em lugar de dois estados estáveis possuem apenas um, além das configurações que não possuem nenhum estado estável. Estes circuitos denominados multivibradores astáveis e monoestáveis também são muito importantes em aplicações relacionadas com a Eletrônica Digital.

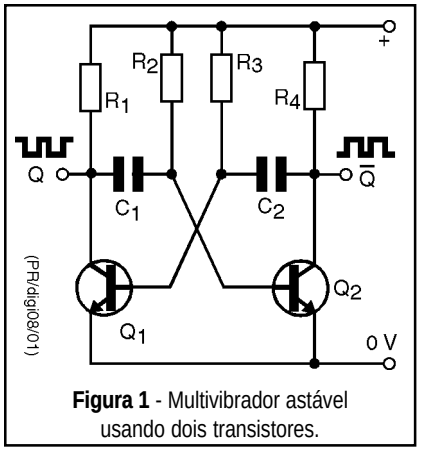


Figura 1 - Multivibrador astável usando dois transistores.

8.1 - MULVIBRADORES ASTÁVEIS

Os circuitos digitais trabalham sincronizados, em sua maioria, por sinais retangulares que precisam ser produzidos por algum tipo de oscilador. O oscilador, que produz o sinal de "*CLOCK*" ou "relógio" deve ter características especiais e para isso podem ser usadas diversas configurações.

Uma das configurações mais interessantes é justamente aquela que parte de um circuito bastante semelhante aos *flip-flops* que estudamos na lição anterior.

Este circuito recebe o nome de multivibrador astável e se caracteriza por não ter dois, nem um estado estável. Este circuito muda constantemente de estado, numa velocidade que depende dos valores dos componentes usados e que, portanto gera um sinal retangular.

Da mesma forma que estudamos os *flip-flops* partindo da configuração básica com transistores, vamos estudar o multivibrador astável.

Assim, caso tenhamos a configuração mostrada na **figura 1**, usando transistores, os capacitores proporcionam uma realimentação que leva o circuito à oscilação.

O multivibrador astável é um circuito em que a frequência é determinada por um capacitor e um resistor, ou seja, por uma constante de tempo RC. Dizemos que este tipo de oscilador é do tipo RC. Analisemos melhor como funciona a configuração mostrada na figura 1.

Quando a alimentação é estabelecida, um dos transistores conduz mais do que outro e inicialmente podemos ter, por exemplo, Q_1 saturado, e Q_2 cortado. Com Q_1 saturado o capacitor C_1 carrega-se via R_1 de modo que a tensão no capacitor sobe gradualmente até o ponto em que, estando carregado, o transistor Q_2 é polarizado no sentido de conduzir. Quando isso ocorre, Q_2 tem um dos seus terminais aterrado e descarrega-se. Nestas condições Q_1 vai ao corte e Q_2 satura. Agora é a vez de C_2 carregar-se até que ocorra novamente uma comutação dos transistores e um novo ciclo de funcionamento.

As formas de onda geradas neste circuito são mostradas na **figura 2**, observando-se o ciclo de carga e descarga dos capacitores.

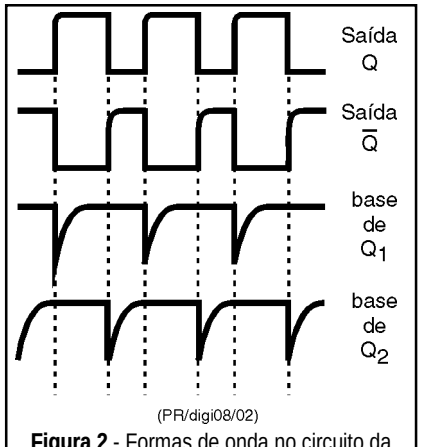


Figura 2 - Formas de onda no circuito da figura 1.

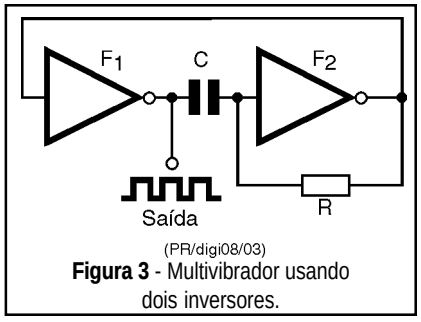


Figura 3 - Multivibrador usando dois inversores.

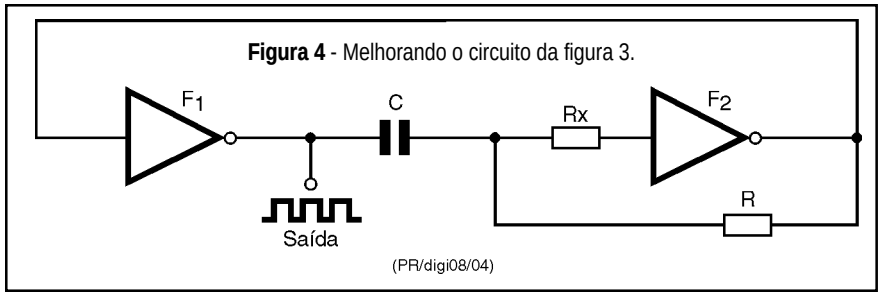


Figura 4 - Melhorando o circuito da figura 3.

O leitor pode perceber então que o tempo de carga e descarga dos capacitores e portanto, das oscilações geradas por este circuito, dependem tanto dos valores dos capacitores como dos resistores de base através dos quais ocorrem as descargas.

Também podemos observar que os sinais gerados são retangulares, pois ocorre uma comutação rápida dos transistores de tal forma que a tensão em seus coletores sobe e desce muito rapidamente.

Da mesma forma que no caso dos *flip-flops*, podemos elaborar multivibradores astáveis, tanto usando válvulas como transistores de efeito de campo.

Podemos também ter osciladores RC que geram sinais com boa estabilidade com menos componentes. Estes osciladores podem ser elaborados com funções lógicas e para isso temos diversas possibilidades.

7.4 - ASTÁVEIS COM FUNÇÕES LÓGICAS

a) Astável usando inversores

Um primeiro tipo de oscilador RC ou astável pode ser elaborado com base em dois inversores, utilizando a configuração mostrada na figura 3.

Neste circuito, R e C determinam a frequência de operação. Resumimos o princípio de funcionamento da seguinte forma: quando o inversor F-2 está com a saída no nível alto, a saída de F-1 estará no nível baixo, o que faz com que o capacitor carregue via R. Quando a tensão em C atinge o valor que provoca a comutação de F-2, ele troca de estado e sua saída vai ao nível baixo. Nestas condições a saída de F-1 vai ao nível alto, o capacitor é "invertido" começando sua carga, mas com polaridade oposta até que novamente tenhamos o reconhecimento do nível de comutação e um novo ciclo se inicie.

A frequência de operação deste circuito é dada com aproximação pela fórmula:

$$f = 1/(2 \times 3{,}14 \times R \times C)$$

3,14 é o famoso "PI" que é constante.

C deve ser expresso em farads, R em ohms para que tenhamos a frequência em hertz.

Nos circuitos integrados CMOS costuma-se agregar nas entradas diodos de proteção com a finalidade de protegê-los contra descargas estáticas. Estes diodos afetam o funcionamento dos osciladores, podendo dificultar sua operação. Uma maneira de contornar o problema causado pela presença dos diodos consiste em modificar o circuito da **figura 3**, agregando um resistor adicional da forma indicada na **figura 4**.

Este resistor Rx deve ser pelo menos 10 vezes maior que R. Valores da ordem de 1 MΩ são os mais usados na prática de modo a não afetar a frequência de operação determinada pela fórmula que vimos.

Podemos controlar a frequência deste tipo de oscilador colocando um resistor variável no circuito de realimentação, verifique a **figura 5**.

Como o resistor variável é 10 vezes maior do que o resistor que está em série, a faixa de frequências obtida variará numa razão de 10 para 1. Assim, se a frequência mínima for de 100 Hz, a máxima será de 1000 Hz. Veja que não é recomendável que o resistor em série seja muito pequeno, menor que 10 kΩ, dadas as características do circuito.

Como o tempo de carga e descarga do capacitor é o mesmo, o sinal produzido é retangular com um ciclo ativo de 50%, ou seja, o tempo em que ele permanece no nível alto é o mesmo do nível baixo, **figura 6**.

Na maioria das aplicações que envolvem o uso de circuitos digitais são necessários circuitos de *clock* que tenham ciclos ativos de 50%, no entanto, existem aplicações especiais em que um ciclo ativo diferente pode ser necessário.

Para modificar o ciclo ativo, o recurso mais comum consiste em ter percursos diferentes para as correntes de carga e descarga do capacitor,

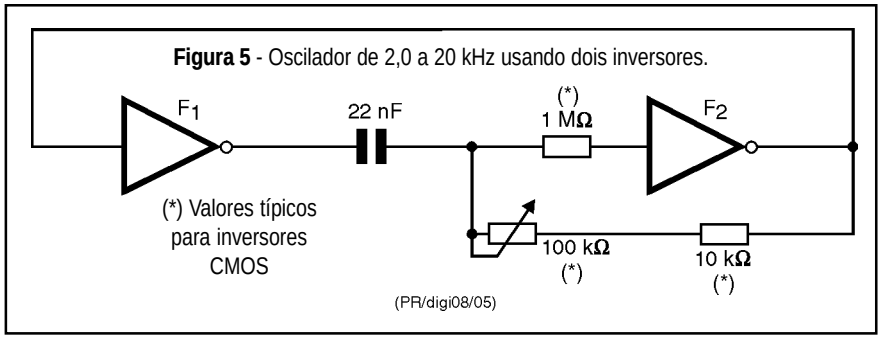


Figura 5 - Oscilador de 2,0 a 20 kHz usando dois inversores.

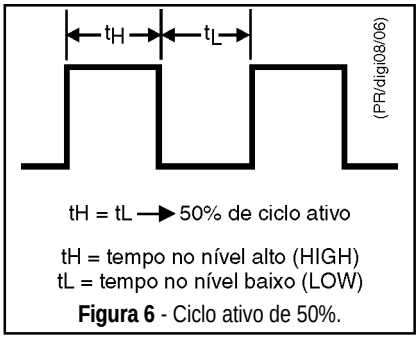


Figura 6 - Ciclo ativo de 50%.

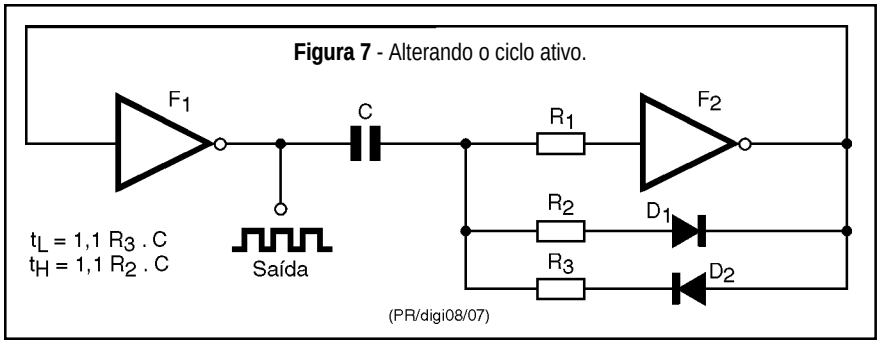


Figura 7 - Alterando o ciclo ativo.

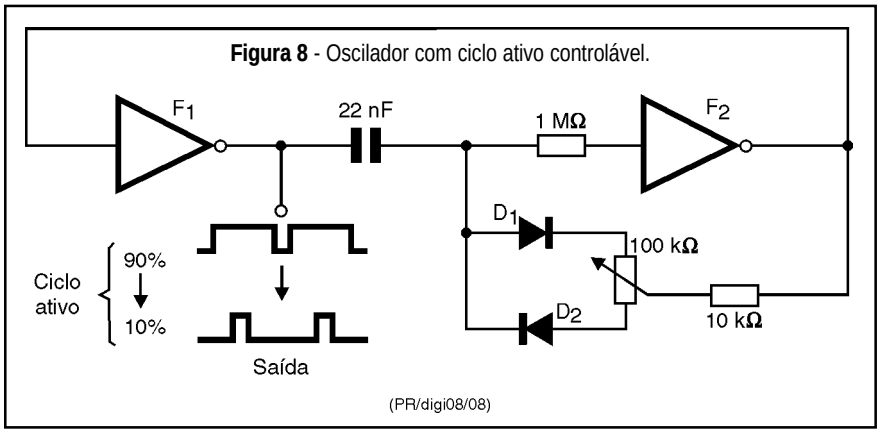


Figura 8 - Oscilador com ciclo ativo controlável.

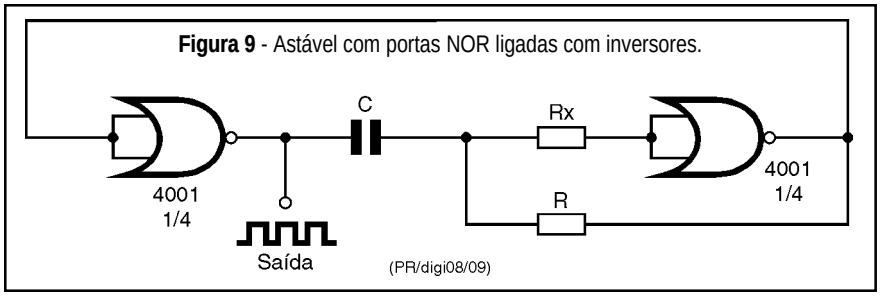


Figura 9 - Astável com portas NOR ligadas com inversores.

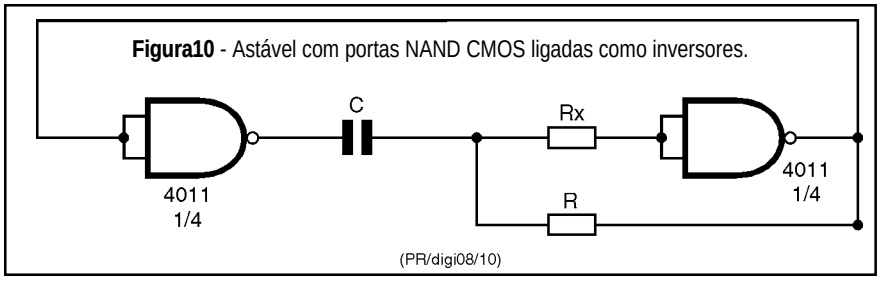


Figura10 - Astável com portas NAND CMOS ligadas como inversores.

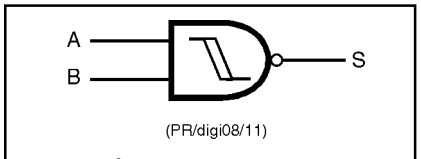


Figura 11 - Porta NAND disparadora (Schimitt trigger).

Um tipo de função lógica importante, que possui tempos reduzidos de comutação, é a formada por circuitos disparadores ou *"triggers",*

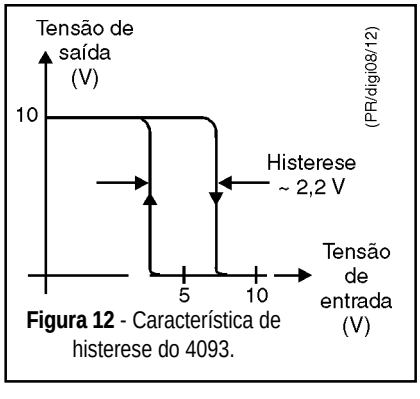


Figura 12 - Característica de histerese do 4093.

o que pode ser conseguido com o uso de diodos.

Assim, para o circuito que tomamos como exemplo é possível modificar o ciclo ativo da maneira indicada na **figura 7**.

O capacitor vai carregar-se via R_1 e descarregar via D_2, o que significa tempos diferentes para a saída no nível alto e baixo.

Estes tempos dependem dos capacitores e são dados pelas fórmulas junto ao diagrama.

Para um ajuste do ciclo ativo podemos agregar um potenciômetro ou *trimpot* ao circuito que vai determinar os percursos para as correntes de carga e descarga do capacitor, conforme **figura 8**.

A posição do cursor determina o ciclo ativo, observando-se que na posição central este ciclo será de 50%.

Observamos finalmente que inversores podem ser obtidos com a ligação de portas NOR com as entradas em paralelo, confira na **figura 9**.

Ou ainda, a configuração indicada pode ser elaborada com portas NAND, ficando com a disposição da **figura 10**.

b) Oscilador com disparador

Uma característica não muito desejada quando se pretende usar uma função como osciladora, é o tempo de comutação quando o nível lógico é reconhecido na entrada.

Um tipo de função lógica importante, que possui tempos reduzidos de comutação, é a formada por circuitos disparadores ou *"triggers",* como por exemplo, do circuito integrado 4093, ver na **figura 11**.

Estas portas possuem uma característica de histerese que é mostrada na **figura 12**.

Esta característica mostra que, quando o circuito reconhece o nível lógico necessário à comutação, a saída passa de um nível a outro numa velocidade muito grande, ou seja, há uma comutação muito rápida.

Por outro lado, o nível lógico de entrada que faz novamente a comutação para que a saída volte ao estado anterior não ocorre com a mesma tensão "de ida".

Em outras palavras, o sinal de saída oscila do nível alto para o baixo

e vice-versa com tensões diferentes de entrada. Estas diferentes tensões determinam uma faixa denominada "histerese" e que é mostrada na curva da figura 12.

Esta característica é muito importante pois garante que o circuito comute com segurança, tanto "na ida" como "na volta" dos sinais, e que além disso, possam ser usados em osciladores de bom desempenho.

Para termos um oscilador com uma porta NAND disparadora como a do circuito integrado CMOS 4093, precisamos de apenas dois componentes externos na configuração mostrada na **figura 13**.

Neste circuito, o capacitor se carrega através do resistor quando a saída da porta (ligada como inversor) está no nível alto e descarrega-se quando está no nível baixo, produzindo um sinal com ciclo ativo de 50%.

A entrada do circuito, ligada entre o capacitor e o resistor, não drena nem fornece corrente, já que é de alta impedância, apenas sensoriando o nível de tensão neste ponto para fazer a comutação.

As formas de onda obtidas neste circuito são mostradas na figura 14.

Da mesma forma que nos circuitos anteriores também podemos modificar o ciclo ativo do sinal gerados modificando o percurso das correntes de carga e descarga do capacitor, o que pode ser conseguido através de diodos.

Temos então na figura 15 um circuito com ciclo ativo diferente de 50% usando diodos.

Neste circuito, quando a saída do disparador está no nível alto, o capacitor carrega-se via D_1 e R_1. Estes componentes determinam então o tempo de saída alto.

Quando o circuito comuta e a saída do disparador vai ao nível baixo, o capacitor descarrega-se via D_2 e R_2, sendo estes os componentes responsáveis pelo tempo baixo do sinal de saída.

Também podemos controlar o ciclo ativo deste circuito colocando um potenciômetro ou *trimpot,* conforme a figura 16.

A posição do cursor determina a resistência do circuito nos percursos de carga e descarga do capacitor.

O 4093, na verdade, corresponde a um grupo de circuitos denominados "disparadores de Schmitt" que será estudado nas próximas lições na sua real função, que é a de modificar formas de onda de um circuito.

O disparador pode transformar um sinal de qualquer forma de onda num sinal retangular, conforme veremos mais adiante.

c) Oscilador TTL com Inversores com saída de coletor aberto

Um outro tipo de circuito astável, que pode ser usado para gerar sinais retangulares num equipamento digital, é o que faz uso de três dos seis inversores disponíveis num circuito integrado 7406. Este circuito é mostrado na figura 17.

O sinal é realimentado da saída do último inversor para a entrada do primeiro e pelo resistor variável temos o ajuste da frequência e do ponto de funcionamento.

Este oscilador pode gerar sinais na faixa de 1 MHz a 10 MHz para TTLs normais e frequências mais elevadas com TTL LS.

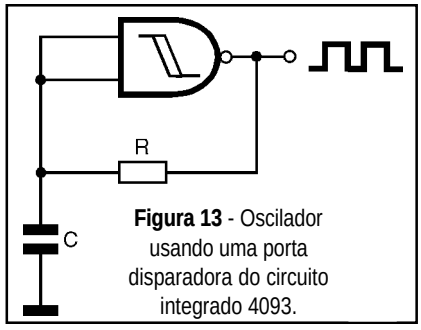


Figura 13 - Oscilador usando uma porta disparadora do circuito integrado 4093.

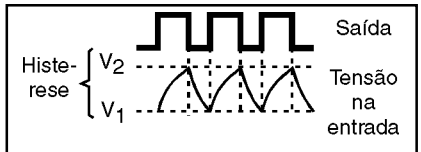


Figura 14 - Formas de onda num oscilador com o 4093.

d) Oscilador com cristal

O cristal é um elemento importante no controle de frequência de um circuito. Os cristais oscilam em frequências determinadas pelo seu corte. Assim, eles podem ser usados para manter a frequência fixa dentro de estreitos limites.

Seu uso mais comum é justamente em circuitos em que a precisão da frequência seja importante, tais como relógios, cronômetros e em

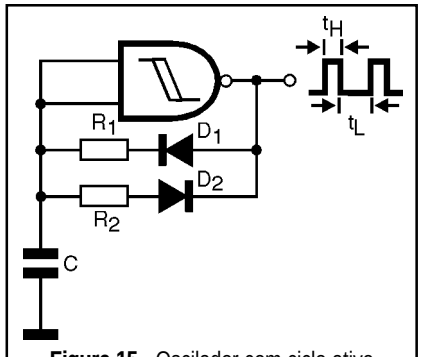


Figura 15 - Oscilador com ciclo ativo determinado pela relação entre R_1 e R_2.

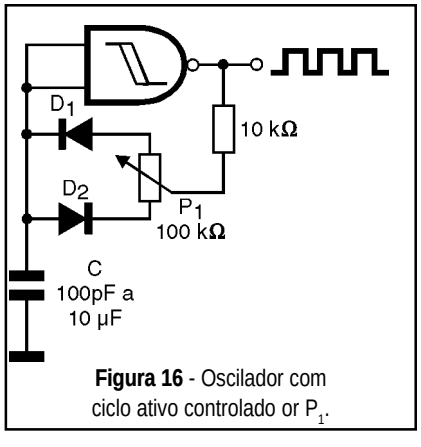


Figura 16 - Oscilador com ciclo ativo controlado or P_1.

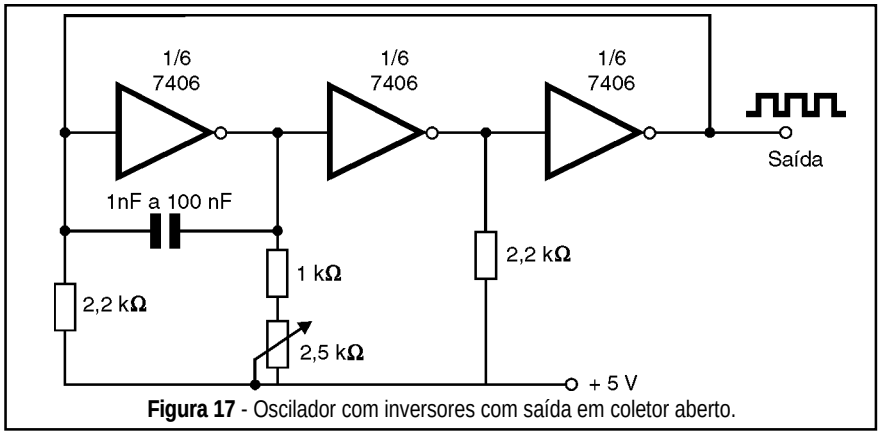


Figura 17 - Oscilador com inversores com saída em coletor aberto.

instrumentação. Existem diversas formas de obter um oscilador com cristal para aplicações em circuitos digitais. Um primeiro circuito que pode ser dado como exemplo é apresentado na **figura 18** e faz uso de duas das quatro portas NOR disponíveis num circuito integrado CMOS 4001.

O cristal serve de elemento de realimentação devendo haver um capacitor ajustável que permite variar a frequência levemente em torno do valor estabelecido pelas características do circuito.

Uma porta serve como elemento ativo do circuito (amplificador), enquanto a outra serve de *"buffer",* ou seja, isola a saída do circuito oscilador.

Os *buffers* são importantes em muitas aplicações, pois impedem que variações ocorridas no circuito que recebe o sinal afetem a frequência do oscilador.

Um outro oscilador a cristal com inversores CMOS é o da **figura 19**.

A saída do último inversor fornece o sinal de realimentação do circuito através do cristal que então determina a sua frequência.

Versão equivalente com inversores e circuitos integrados TTL para osciladores controlados a cristal é mostrada na **figura 20**.

QUESTIONÁRIO

1. Quantos estados estáveis têm um multivibrador monoestável?
a) 1 b) 2 c) nenhum
d) todos os estados são estáveis

2. Qual dos circuitos abaixo indicados não pode ser usado como astável ou monoestável?
a) 7474 b) 555
c) 74122 d) 4011

3. O tempo de carga de um capacitor é diferente do tempo de descarga quando usado num astável. Podemos dizer que o ciclo ativo do sinal gerado é:
a) 50%
b) maior que 50%
c) menor que 50%
d) diferente de 50%

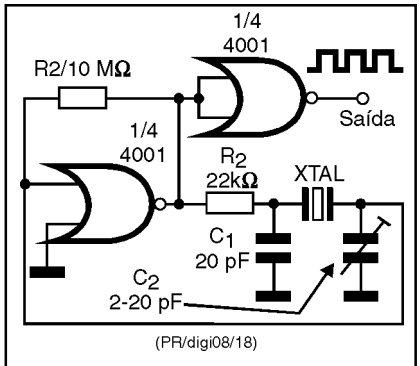


Figura 18 - Oscilador CMOS a cristal.

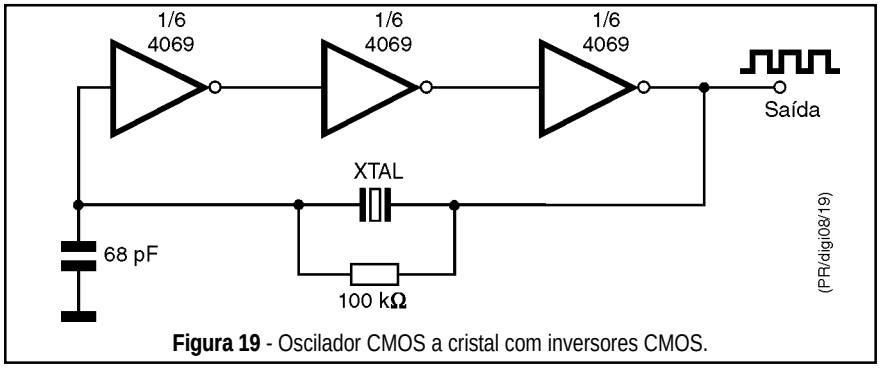


Figura 19 - Oscilador CMOS a cristal com inversores CMOS.

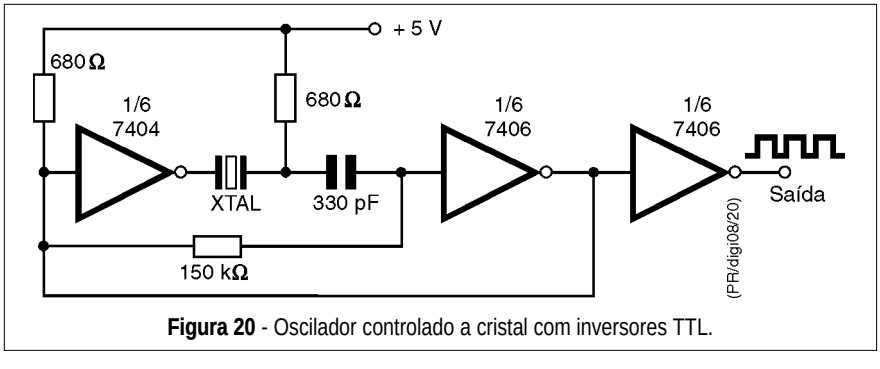


Figura 20 - Oscilador controlado a cristal com inversores TTL.

LIÇÃO 9

OS CONTADORES DIGITAIS

Na lição anterior analisamos o princípio de funcionamento de um dos mais importantes blocos da Eletrônica Digital, o *flip-flop*. Vimos que estes blocos poderiam ter diversos tipos de comportamento e que, quando reunidos, poderiam apresentar comportamentos interessantes como, por exemplo, a capacidade de dividir frequências, de armazenar informações (bits), além de outras. Nesta lição vamos nos dedicar justamente a uma das funções mais importantes dos *flip-flops* que é a de fazer a contagem do número de pulsos, o que corresponde em última análise a contagem de bits. A partir desta contagem podemos usar estes circuitos para a realização de operações mais complexas como somas, manipulação de dados etc.

9 - OS TIPOS DE CONTADORES

Em Eletrônica Digital devemos separar os circuitos lógicos sem sincronismo daqueles que possuam algum tipo de sincronismo externo, ou seja, que usam um sinal de *CLOCK*.

Existem aplicações em que tudo o que importa para o circuito é fazer uma operação com determinados níveis lógicos aplicados à sua entrada, quando eles estão presentes, não importando quando isso ocorra. Tais circuitos não precisam de sincronismo algum e são mais simples de serem utilizados.

No entanto, com circuitos muito complexos, como os utilizados em computadores e em muitos outros casos, o instante em que uma operação deve ser realizada é muito importante e isso implica em que os circuitos devam ser habilitados no instante em que os níveis lógicos são aplicados em sua entrada.

Isso significa que tais circuitos devem ser sincronizados por algum tipo de sinal vindo de um circuito externo. E este circuito nada mais é do que um oscilador que produz um sinal de *clock* ou relógio.

Os circuitos que operam com estes sinais são denominados circuitos com lógica sincronizada.

Para os contadores temos então diversas classificações que levam em conta estes e outros fatores, por exemplo:

a) Classificação quanto ao sincronismo:

Os contadores podem ser ASSÍNCRONOS, quando existe o sinal de *clock* aplicado apenas ao primeiro estágio. Os estágios seguintes utilizam como sinal de sincronismo a saída de cada estágio anterior. Estes contadores também são denominados *Ripple Counters*.

Os contadores também podem ser SÍNCRONOS, quando existe um sinal de *clock* único externo aplicado a todos os estágios ao mesmo tempo.

b) Classificação quanto ao modo de contagem

Os contadores podem ser PROGRESSIVOS ou CRESCENTES, quando contam numa sequência de números crescentes, ou seja, dos valores mais baixos para os mais altos, como (1,2,3,4...). São também chamados pelo termo inglês de *UP COUNTERS*.

Os contadores podem ser REGRESSIVOS ou DECRESCENTES, quando a contagem é feita dos valores mais altos para os mais baixos como (4,3,2,1...).O termo inglês é *DOWN COUNTERS*.

Se bem que possamos fazer contadores usando funções lógicas comuns e mesmo *flip-flops* discretos, podemos contar na prática com circuitos integrados em lógica TTL ou CMOS que já possuam contadores completos implementados.

9.1 - CONTADOR ASSÍNCRONO

Conforme explicamos, neste tipo de contador, o sinal de *clock* é aplicado apenas ao primeiro estágio, ficando os demais sincronizados pelos estágios anteriores.

Na figura 1 temos a estrutura básica de um contador deste tipo usando *flip-flops* do tipo J-K.

Usamos três estágios ou três *flip-flops* ligados de tal forma que a saída Q do primeiro serve de *clock* para o segundo, e a saída Q do segundo serve de *clock* para o terceiro.

Sabemos que os *flip-flops* ligados da forma indicada funcionam como divisores de frequência.

Assim, o sinal de *clock* aplicado ao primeiro tem sua frequência dividida por 2.

A frequência estará dividida por 4

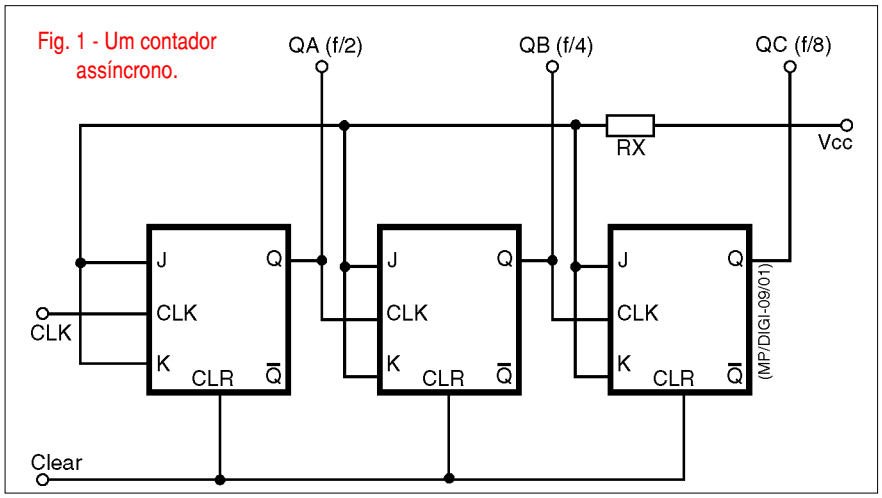


Fig. 1 - Um contador assíncrono.

na saída do segundo e por 8 na saída do terceiro.

Tudo isso pode ser visualizado pelo diagrama de tempos mostrado na figura 2.

Mas, se elaborarmos uma tabela verdade com os níveis lógicos obtidos na saída de cada um dos *flip-flops,* a cada pulso do *clock* aplicado, a partir do instante em que todas as saídas sejam zero, teremos algo interessante a considerar:

Entrada	QC	QB	QA
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

Veja que a sequência de valores obtidos 000, 001, 010, 011, 100, 101, 110 e 111 corresponde justamente à contagem em binário dos pulsos de 0 a 7! Em outras palavras, este circuito conta os pulsos de entrada e fornece saídas que são a representação binária desta contagem.

Veja também que ele faz a contagem crescente, ou seja, de 0 até 7.

Se, em lugar de três *flip-flops,* usarmos quatro, no circuito mostrado na figura 3, teremos a contagem de 0000 a 1111, ou seja, uma contagem crescente de 0 a 15 pulsos.

Oito desses *flip-flops* ligados em série podem contar até 256 pulsos e com isso fornecer uma saída de 8 bits ou 1 byte.

O circuito apresentado comuta na transição negativa do sinal de *clock*.

Vamos supor agora que em lugar de usarmos como saídas de contagem as saídas Q de cada *flip-flop*, usássemos as saídas complementares /Q, conforme a figura 4. É fácil perceber que, partindo da situação em que todos os *flip-flops* estejam ressetados, a tabela verdade obtida terá nas saídas os complementos da tabela anterior. Esta tabela será:

Entrada	QA	QB	QC	Valor Binário
0	1	1	1	7
1	1	1	0	6
2	1	0	1	5
3	1	0	0	4
4	0	1	1	3
5	0	1	0	2
6	0	0	1	1
7	0	0	0	0

Portanto, este contador fornece em sua saída valores binários que correspondem à contagem decrescente dos pulsos de entrada, partindo de 7. Trata-se de um contador decrescente ou *DOWN COUNTER*.

Como no caso anterior, se tivermos mais *flip-flops,* podemos contar a partir de valores mais altos. Com 4 *flip-flops* podemos partir a contagem de 15 e com 8 *flip-flops,* de 255. Veja que a quantidade máxima que podemos contar com um contador deste tipo depende da quantidade de *flip-flops* usados.

Um problema que ocorre com este tipo de *flip-flop* é que cada um precisa de um certo tempo para mudar de

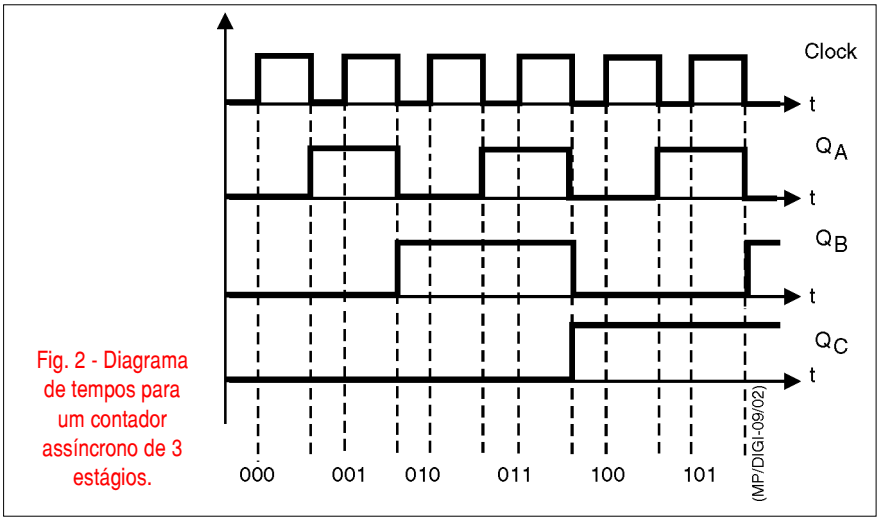


Fig. 2 - Diagrama de tempos para um contador assíncrono de 3 estágios.

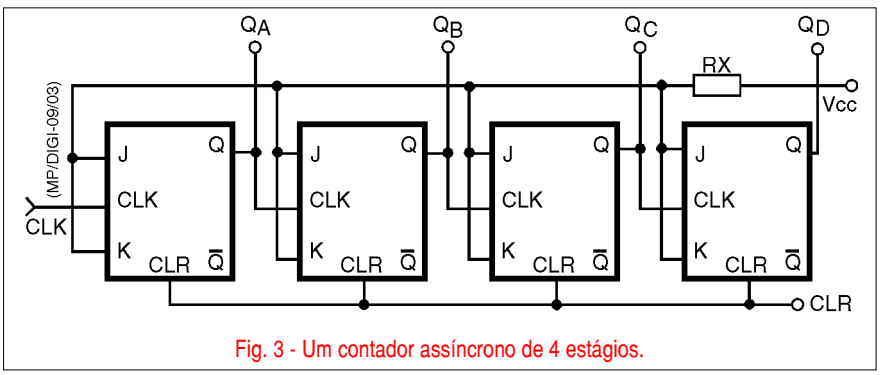


Fig. 3 - Um contador assíncrono de 4 estágios.

estado. Isso significa que à medida que usamos mais *flip-flops* em sequência num contador, os tempos de mudança de estado são somados e o conjunto precisa cada vez de mais tempo para chegar ao estado final desejado.

Se aplicarmos um novo pulso de *clock* para contagem à entrada do circuito, antes de ocorrer a mudança de estado do conjunto, pode ocorrer um funcionamento errático. Assim, a frequência máxima de operação de um contador é dada pelo tempo necessário para cada estágio mudar de estado multiplicado pelo número de estágios usados no contador.

9.2 - CONTAGEM PROGRAMADA

Conforme vimos, os ciclos de contagem dos circuitos dados como exemplos no item anterior são sempre potências de 2, ou seja, são circuitos que contam até 2, 4, 8, 16, 32 etc. O que fazer se precisarmos de um circuito que tenha um ciclo de contagem diferente desses valores, que não seja uma potência de 2?

Devemos levar em conta dois fatores:

Podemos usar a entrada *CLEAR* para reiniciar a contagem, zerando-a, quando chegar ao valor desejado. Por exemplo, podemos reiniciar a contagem depois do 5 se quisermos um contador que conte de 0 a 5, ou seja, que tenha 6 estados de saída, conforme a tabela verdade dada a seguir:

Entrada	QC	QB	QA
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	0	0	**0**

estado instável

No sexto pulso que corresponde ao estado 110, o circuito vai a um estado que ativa a entrada *CLEAR* e leva todos os *flip-flops* a serem ressetados.

Para este circuito a solução é simples. Veja que a situação em que devemos ter a volta a zero da contagem

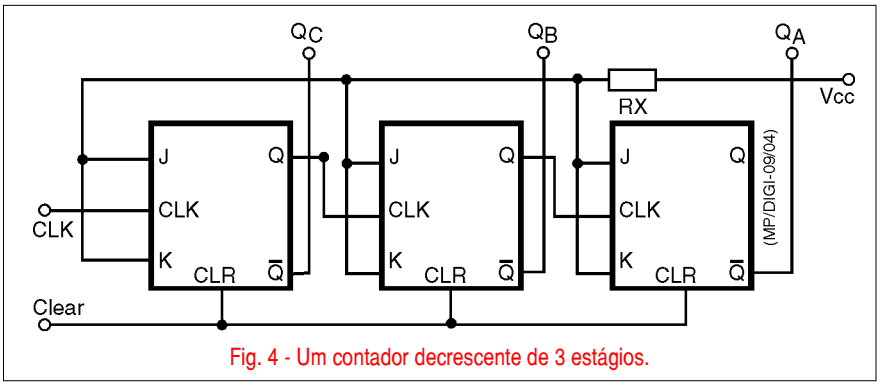


Fig. 4 - Um contador decrescente de 3 estágios.

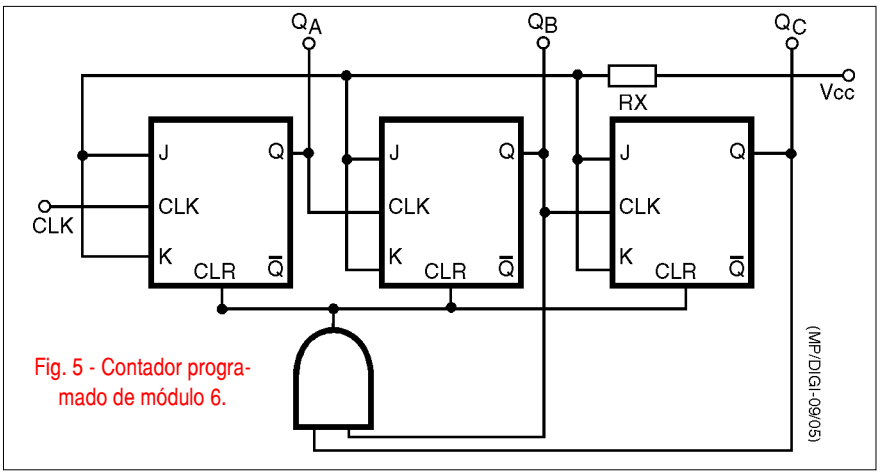


Fig. 5 - Contador programado de módulo 6.

e portanto, a ativação da linha CLR *(clear)* ocorre com uma única combinação de sinais: QA e QB no nível alto.

Se usarmos *flip-flops* que tenham entradas *CLEAR* ativadas pelo nível alto, basta usar uma porta AND de duas entradas com as entradas ligadas nas saídas QB e QC e a saída na linha comum de *CLEAR* de todos os *flip-flops*, conforme a figura 5. Se os *flip-flops* usados tiverem um *CLEAR* ativado no nível baixo como o 7476 (TTL), basta usar uma porta NAND em lugar de AND.

Se quiséssemos um contador até 4, por exemplo, o estado em que deveria ocorrer a ativação da entrada *CLEAR* ocorreria com a quinta combinação de saídas, ou seja, 101, o que significa QC=1 e QA=1. Bastaria então ligar as entradas da porta AND nessas saídas, conforme a figura 6.

Um diagrama de tempos pode mostrar exatamente o que ocorre com o contador elaborado desta maneira. Este diagrama é apresentado na figura 7. Observe que, quando as saídas chegarem ao estado 110, que

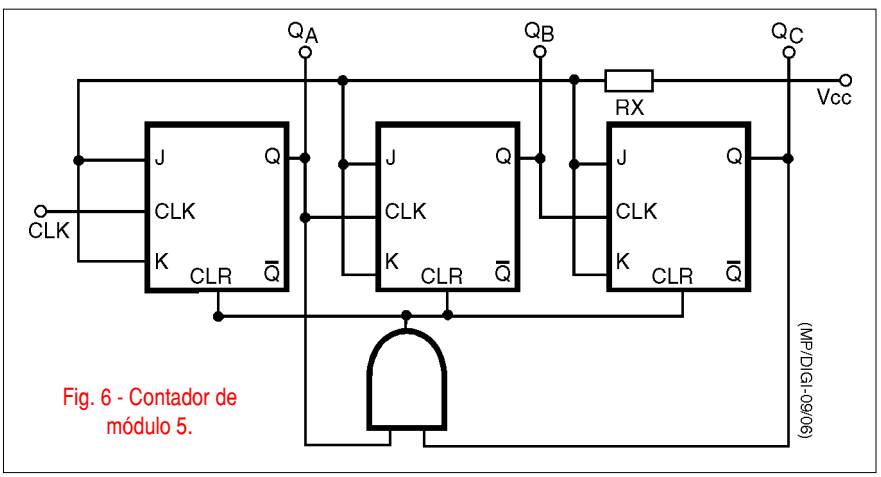


Fig. 6 - Contador de módulo 5.

seria a contagem do quinto pulso no circuito da figura 6, um pulso de reset de curta duração é produzido. Esta curta duração é dada justamente pelo tempo que os *flip-flops* demoram para mudar de estado ressetando, pois eles "realimentam" as entradas da porta AND.

Nos exemplos dados fizemos a programação da contagem usando as entradas de *CLEAR* de cada *flip-flop*.

Uma outra maneira de projetarmos um contador consiste em usarmos as entradas *PRESET* em lugar de *CLEAR*.

Para isso fazemos com que, no momento em que for atingida a contagem do valor imediatamente anterior àquele em que deve ocorrer a volta a zero, ou seja, n-1, em lugar de termos a comutação dos *flip-flops*, tenhamos a ativação das entradas de *PRESET*. Desta forma, no pulso seguinte de *clock* (n) teremos a volta a zero (reset) do contador.

Para um contador de 6 estados, que depois do quinto pulso resseta, teremos a seguinte tabela verdade.

Pulsos	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
o PRESET é ativado			
x	x	x	x
volta a zero na transição do *clock*			
6	0	0	0
7	0	0	1
8	0	1	0

Um circuito usando uma porta NAND é mostrado na figura 9.

Veja que a detecção da condição de produção do pulso de PRESET deve ser reconhecida com os níveis 101 nas saídas dos estágios dos contadores e com o pulso indo ao nível alto na entrada de contagem.

Para obtermos a configuração 1111 que nos permitiria usar uma porta AND de quatro entradas, basta levar em conta a saída /QB em lugar de QB.

Assim, basta usar a porta *AND* e ligá-la nas entradas de *PRESET* (PR) dos *flip-flops*.

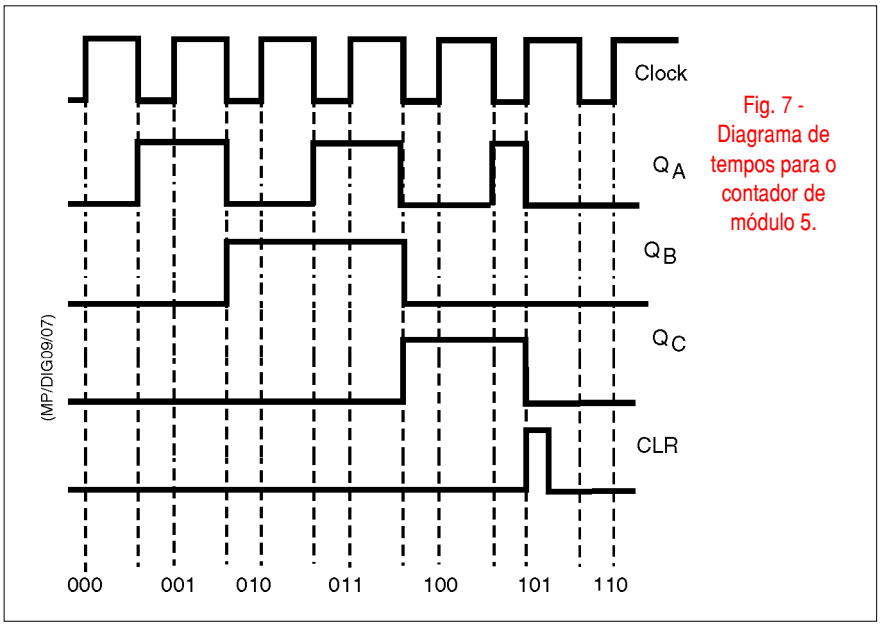


Fig. 7 - Diagrama de tempos para o contador de módulo 5.

Se as entradas forem ativadas no nível baixo (/PR), basta trocar a porta *AND* por uma porta *NAND* de quatro entradas.

9.3 - CONTADORES *UP/DOWN* (PROGRESSIVOS E REGRESSIVOS)

Usando alguns artifícios, como por exemplo, portas apropriadas, é possível programar um contador de modo que ele tanto conte progressivamente como regressivamente.

Usando 3 estágios, podemos ter um contador *UP/DOWN,* conforme a figura 10. Uma entrada *(UP/DOWN)* pode ser usada para determinar o sentido da contagem.

Trata-se de uma entrada seletora de dados ou DATA SELECTOR, que pode ser usada para mudar o modo de funcionamento dos estágios deste circuito.

Funcionamento: conforme vimos nesta lição, se usarmos as saídas Q dos *flip-flops* de um contador, a contagem será crescente, mas se usarmos as saídas /Q, a contagem será

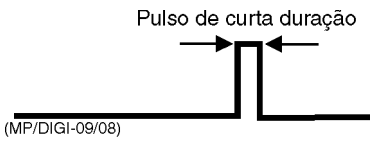


Fig. 8 - Na saída da porta AND temos um pulso de curta duração que resseta o contador.

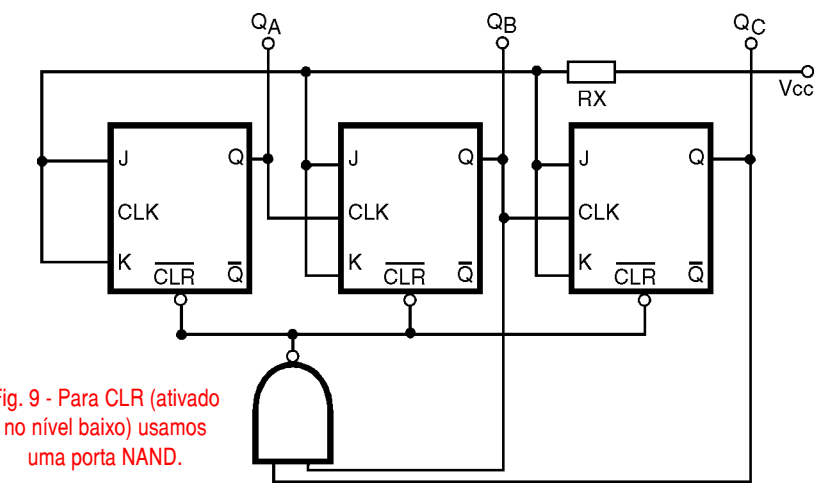


Fig. 9 - Para CLR (ativado no nível baixo) usamos uma porta NAND.

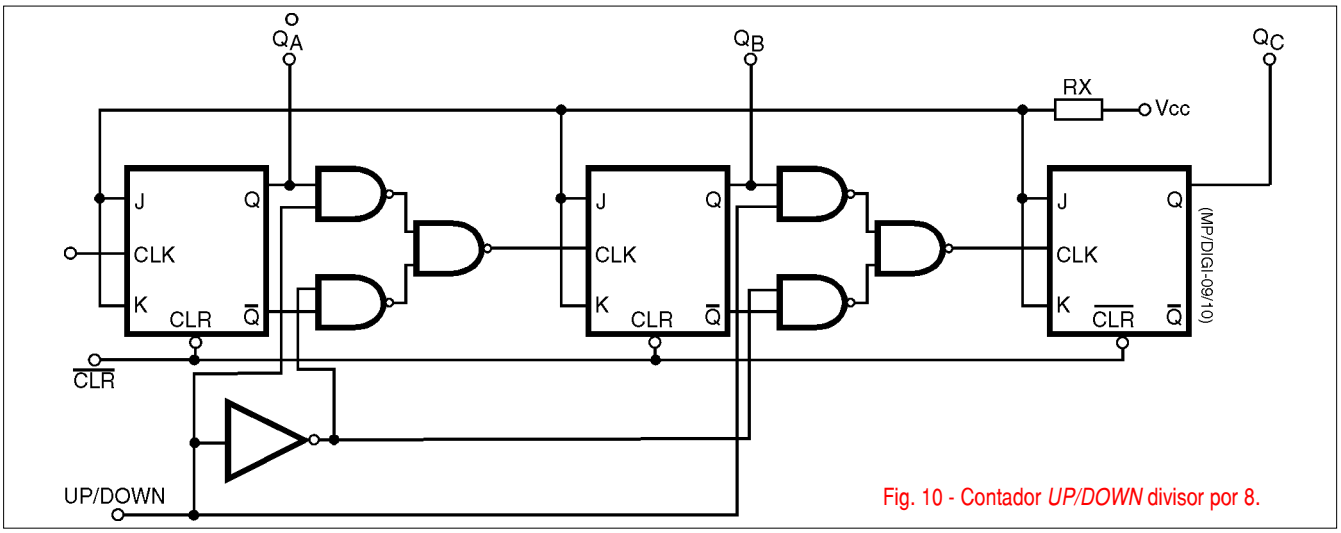


Fig. 10 - Contador *UP/DOWN* divisor por 8.

decrescente. Assim, o que fazemos é colocar um circuito seletor nessas saídas, de tal modo que ele coloque a saída Q de cada *flip-flop* na entrada de *clock* do seguinte, quando a contagem deve ser progressiva, e coloque a saída /Q na entrada do seguinte, quando na contagem descrescente. Três portas *NAND* para cada estágio podem fazer isso a partir do sinal de comando *UP/DOWN*.

9.4 - CONTADORES SÍNCRONOS

Sincronizar a contagem por um *clock* único aplicado a todos os estágios não é apenas uma necessidade dos circuitos mais complexos, principalmente, os usados em Informática e Instrumentação.

O sincronismo de todos os estágios pelo mesmo *clock* tem ainda vantagens operacionais importantes. Conforme vimos, nos contadores assíncronos, os tempos de comutação dos *flip-flops* influem no funcionamento final do circuito, pois eles são cumulativos.

Em outras palavras, cada estágio precisa esperar o anterior completar a operação antes de iniciar a sua.

Usando lógica sincronizada, ou seja, um contador em que todos os estágios são sincronizados por um *clock* único, este problema não existe e podemos ter contadores muito mais rápidos, na verdade, contadores cuja velocidade independe do número de etapas.

Para mostrar como isso pode ser feito, vamos tomar como exemplo o circuito da figura 11.

Este circuito utiliza *flip-flops* tipo J-K ligados de uma forma denominada *PARALLEL CARRY*.

Nesta forma de ligação, J e K do primeiro *flip-flop* são mantidas no nível alto por meio de um resistor ligado ao positivo da alimentação (Vcc).

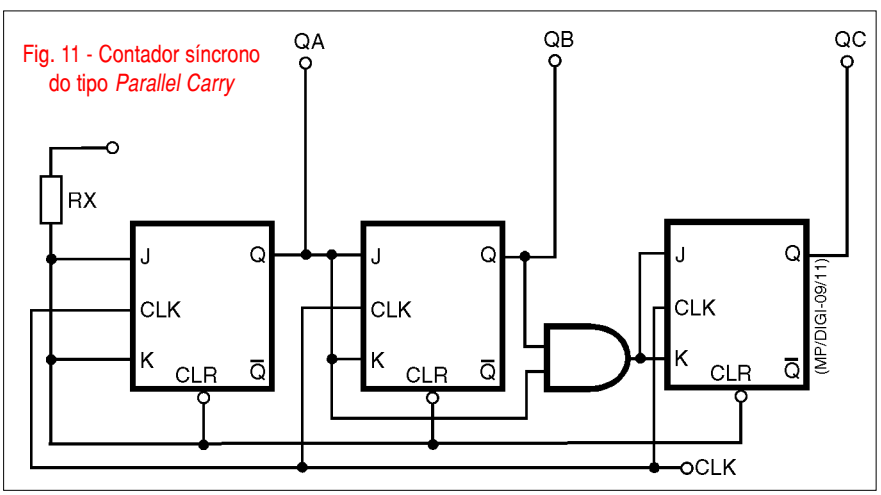


Fig. 11 - Contador síncrono do tipo *Parallel Carry*

Assim, o primeiro *flip-flop* muda de estado a cada pulso de *clock*. No entanto, J do segundo *flip-flop* está ligado à saída Q do primeiro. Isso significa que o segundo *flip-flop* só mudará de estado quando o primeiro *flip-flop* for ressetado, ou seja, a cada dois pulsos de *clock*.

Da mesma forma, com o uso de uma porta AND, o terceiro *flip-flop* só vai mudar de estado quando as saídas Q do primeiro e segundo *flip-flop* forem ao nível 1, ou seja, a cada 4 pulsos de *clock*.

Para 4 bits, utilizando 4 estágios, podemos usar o circuito mostrado na figura 12.

Um problema que ocorre com este tipo de configuração é que a partir de 3 estágios, a cada estágio que acrescentamos no contador devemos adicionar uma porta *AND* cujo número de entradas vai aumentando.

Assim, para 4 estágios, a porta deve ter três entradas, para 5 estágios, 4 entradas e assim por diante.

Uma maneira de não termos este problema consiste em usar uma configuração diferente de contador apresentada na figura 13 e denominada *RIPPLE CARRY*.

Neste circuito as portas usadas sempre precisam ter apenas duas entradas, o que é importante para a implementação prática do contador.

No entanto, como desvantagem deste circuito, temos uma limitação da velocidade de operação, pois como o sinal para os estágios vem da porta anterior, temos de considerar seu atraso.

9.5 - CONTADORES SÍNCRONOS PROGRAMÁVEIS

Da mesma forma que no caso dos contadores assíncronos, também é necessário, em determinadas aplicações, fazer a contagem até valores que não sejam potências de 2.

A divisão ou contagem por outros valores pode ser feita com a ajuda de portas ligadas de modo a "sentir" quando um determinado valor é alcançado, ressetando então todos os *flip-flops*.

9.6 - CONTADORES TTL

Utilizando portas lógicas e *flip-flops* podemos implementar contadores que contenham ou façam a divisão de um sinal de entrada por qualquer valor. No entanto, na prática, podemos contar com muitos circuitos integrados em tecnologia TTL que já contenham estes circuitos completos num único chip e até com recursos que permitam alterar seu funcionamento de modo a ser obtida a contagem até um determinado valor.

A seguir veremos alguns dos principais circuitos integrados contadores em tecnologia TTL.

a) 7490 - Contador de Década

Este é um dos mais populares dos contadores TTL e contém em seu interior quatro *flip-flops* já interligados de modo a funcionar como divisores por 2 e por 5. Isso significa que esses divisores podem ser usados para resultar num contador até 2 e num contador até 5, e em conjunto, num contador até 10.

Na figura 14 temos a disposição dos terminais deste circuito integrado.

Este circuito pode ser usado de três formas diferentes, sempre com as entradas R0(1), R0(2), R9(1) e R9(2) aterradas:

Quando ligamos a entrada B à saída QA e aplicamos o sinal de *clock* à entrada A, o circuito funciona como um contador BCD, ou seja, conta até 10, com as saídas em decimal codificado em binário apresentadas nos pinos QA, QB, QC e QD. Esta ligação é mostrada na figura 15.

A tabela verdade para os pulsos aplicados na entrada neste modo de funcionamento será:

Pulso	QD	QC	QB	QA
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

Quando ligamos a saída QD à entrada A e aplicamos o sinal de *clock* à entrada B, teremos o circuito funcionando como um divisor de frequência por 10 simétrico. Teremos na saída QA um sinal quadrado (ciclo ativo de 50%) com 1/10 da frequência do *clock*.

Este modo de funcionamento tem as ligações mostradas na figura 16.

Finalmente, quando quisermos usar o circuito como divisor por 2 ou por 5, independentes, não é preciso ligação externa alguma.

O sinal aplicado em CLK1 tem a frequência dividida por 2 e o sinal

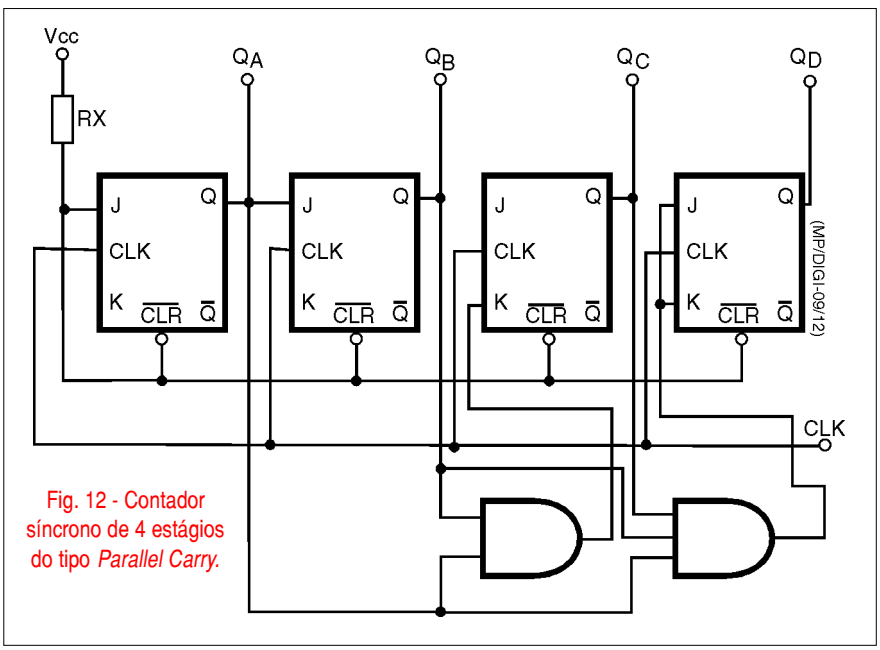


Fig. 12 - Contador síncrono de 4 estágios do tipo *Parallel Carry*.

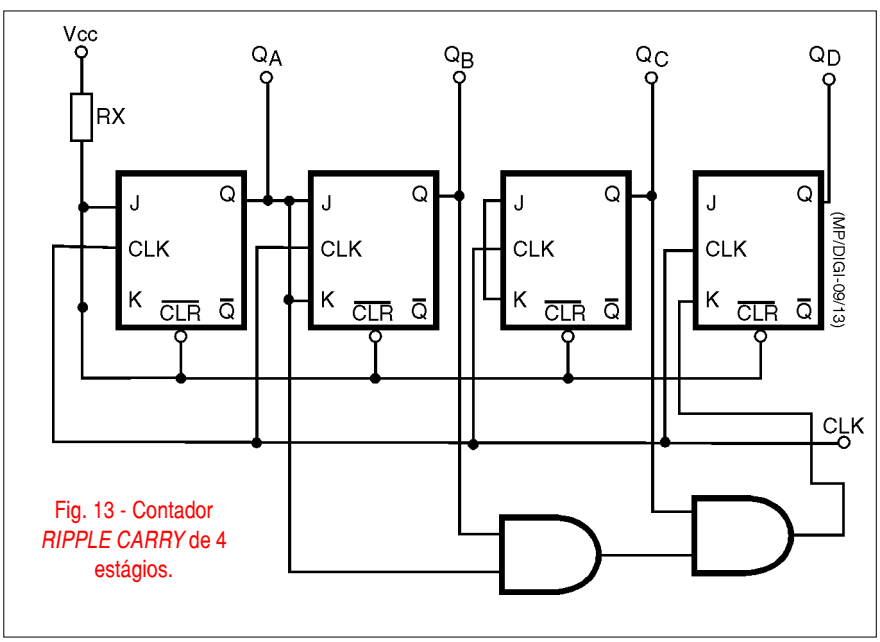


Fig. 13 - Contador *RIPPLE CARRY* de 4 estágios.

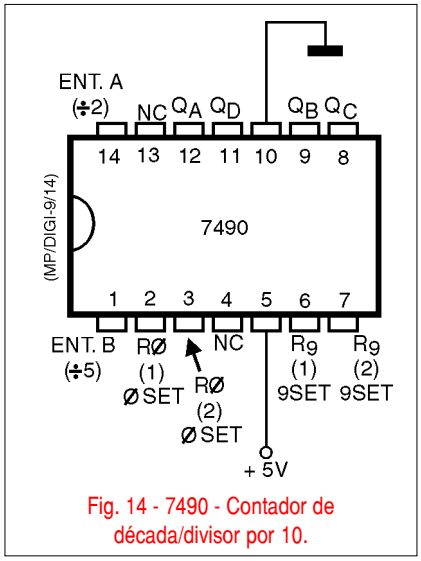


Fig. 14 - 7490 - Contador de década/divisor por 10.

aplicado no CLK2 tem a frequência dividida por 5. Na operação normal as entradas R0(1) e R0(2) devem ser mantidas no nível baixo.

b) 7492 - Contador-Divisor por 12

Este circuito integrado contém quatro *flip-flops* ligados como um divisor por 2 e um divisor por 6 que podem ser usados de maneira independente.

A pinagem deste circuito integrado TTL é mostrada na figura 17.

O disparo dos *flip-flops* ocorre na transição do sinal de *clock* do nível alto para o nível baixo. Para ressetar o contador para 0000, basta aplicar o nível lógico 1 nas entradas R0.

Existem três modos de operação para este circuito integrado:

Como contador até 12, basta ligar a saída QA à entrada B. O sinal de *clock* é aplicado à entrada A.

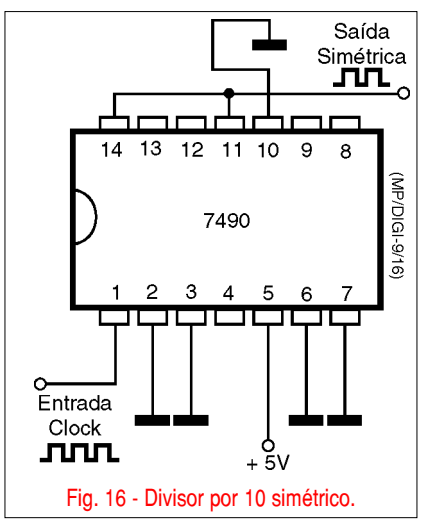


Fig. 16 - Divisor por 10 simétrico.

A tabela verdade para este modo de operação será:

Entrada	QD	QC	QB	QA
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

Na segunda forma de operação, ligamos a saída QD à entrada A. O circuito funcionará como um divisor simétrico de frequência. A frequência do sinal de *clock* aplicado à entrada B será dividida por 12 e o sinal terá um ciclo ativo de 50%.

Na operação sem nenhuma ligação externa, o sinal aplicado à entrada A terá sua frequência dividida por 2 e o sinal aplicado na entrada B terá sua frequência dividida por 6.

9.7 - CONTADORES E DIVISORES CMOS

Temos ainda diversos circuitos integrados em tecnologia CMOS contendo contadores e divisores.

A seguir veremos um dos mais importantes.

Na operação normal, contando até 10, as entradas RST e EN devem ser mantidas no nível baixo.

Levando-se a entrada RST ao nível alto, o contador é ressetado. Se a entrada EN for levada ao nível alto, a contagem é paralisada.

Na figura 18 temos as formas de onda deste contador, mostrando de que forma em cada instante temos sempre apenas uma saída no nível alto.

Como em todos os circuitos CMOS, a frequência máxima de contagem depende da tensão de alimentação. Para 10 V, a frequência máxima é da ordem de 5 MHz.

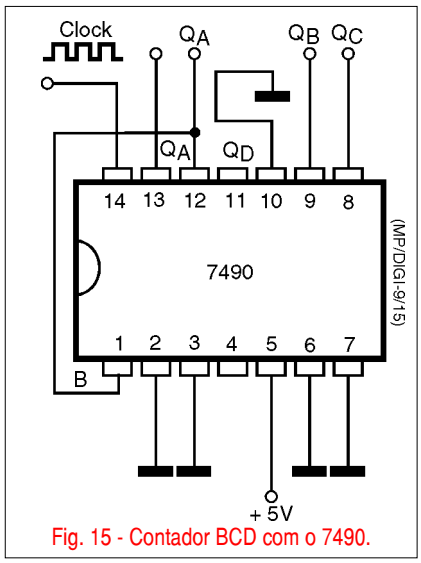


Fig. 15 - Contador BCD com o 7490.

4018 - Contador/Divisor Por N

Este circuito integrado, que será melhor analisado na próxima lição, pode fazer a divisão ou contagem de pulsos em valores até 10 programados pelas ligações externas.

Sua pinagem é mostrada na figura 19 e seu uso será visto posteriormente.

QUESTIONÁRIO

1. Que tipo de contador tem cada estágio controlado pelo anterior, com o sinal de *clock* aplicado apenas ao primeiro estágio?

a) Síncrono
b) Assíncrono
c) *Ripple Counter*
d) Contador de década

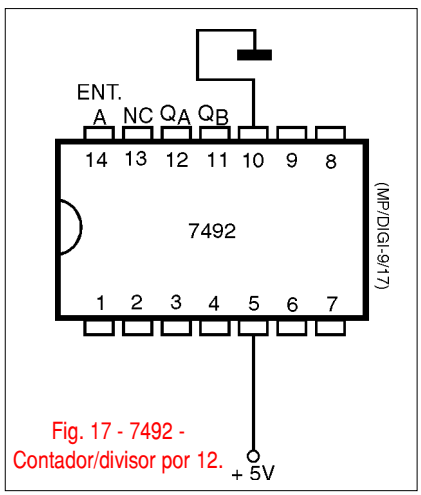


Fig. 17 - 7492 - Contador/divisor por 12.

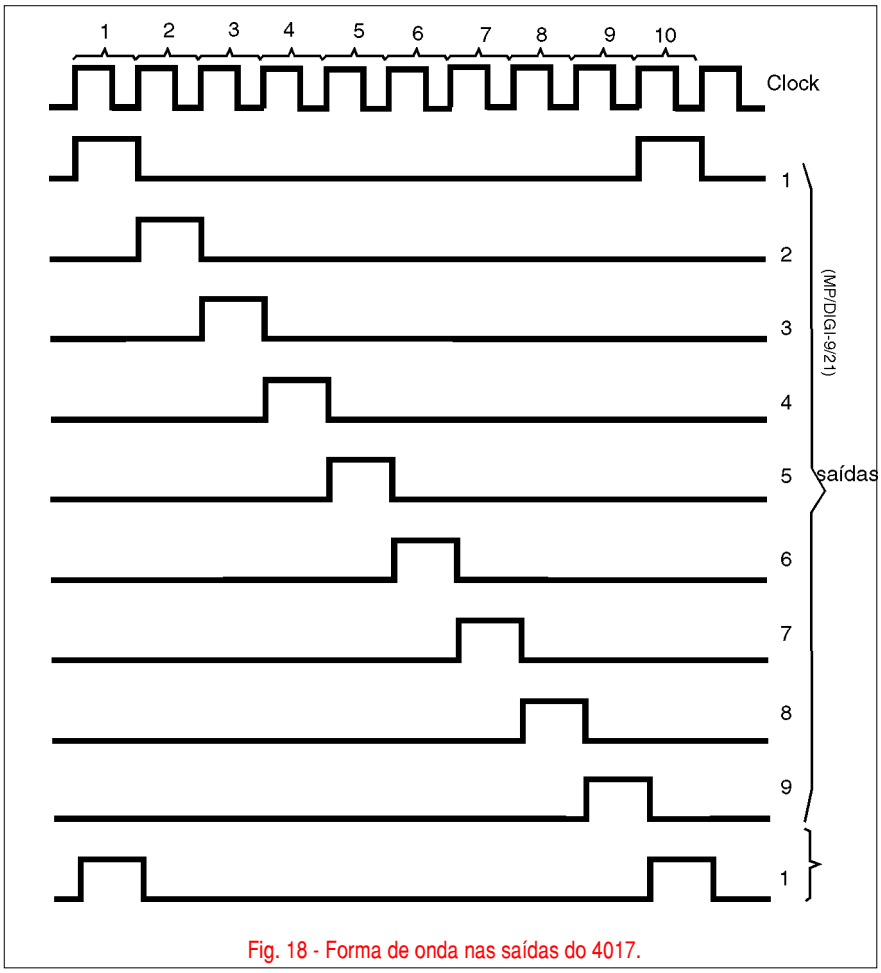


Fig. 18 - Forma de onda nas saídas do 4017.

2. Qual é o valor máximo de contagem de um contador que use 4 *flip-flops*?
a) 4
b) 8
c) 16
d) 10

3. Para um contador de 4 estágios, um do tipo síncrono e outro assíncrono, qual é o mais rápido?
a) O contador síncrono
b) O contador assíncrono
c) Ambos têm a mesma velocidade
d) Depende do modo como são usados

4. Podemos fazer a contagem até valores que não sejam potências de 2 usando que tipos de circuitos?
a) Contadores comuns sozinhos
b) Contadores comuns e funções lógicas
c) Somente funções lógicas complexas
d) Não é possível fazer isso

5. Qual dos contadores/divisores abaixo relacionados tem saídas do tipo 1-de- 10?
a) 7400 b) 7490
c) 74190 d) 4017

Resp.:1-b, 2-c, 3-a, 4-b, 5-d

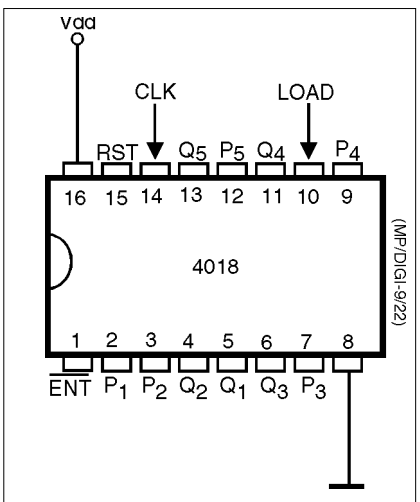


Fig. 19 - 4018 - Divisor por n programável.

LIÇÃO 10

APLICAÇÃO PARA OS CONTADORES DIGITAIS/DECODIFICADORES

Na lição anterior estudamos os contadores e divisores de frequências que consistem em blocos digitais utilizando *flip-flops,* elementos fundamentais para o projeto de circuitos. Na mesma lição vimos o funcionamento dos contadores em detalhes, analisando os diversos tipos possíveis e algumas alterações que podem ser feitas no seu modo de ligação e na própria utilização, de grande importância para os projetos práticos.

Nesta lição continuaremos a explorar o assunto, com a análise de alguns circuitos práticos que podem ser elaborados com base nos circuitos integrados TTL e CMOS que consistem em contadores e divisores de frequência.

Será muito importante o leitor prestar bastante atenção nestes blocos pela sua utilidade no projeto de grande quantidade de circuitos digitais e para o entendimento de circuitos equivalentes encontrados em computadores e outras aplicações semelhantes.

10.1 - CONTADORES/DIVISORES POR N

Dividir uma frequência por um valor qualquer (n) é um problema cuja solução pode ser muito importante para a implementação de um projeto digital.

Conforme vimos na lição anterior, a divisão natural de circuitos que usam *flip-flops* é por valores que sejam de potências de 2, conforme a **figura 1**.

No entanto, usando recursos simples como portas, podemos alterar este comportamento e assim obter a divisão por qualquer número inteiro que seja menor que o valor n da divisão final do módulo ou contador, **figura 2**.

Na prática, temos contadores e divisores na forma de circuitos integrados digitais que podem ser usados na divisão por determinados números fixados por elementos internos do circuito e também podem ser usados na divisão por qualquer outro valor, quer seja por meio de programação, quer seja pelo uso de elementos externos, ou ainda pelos dois recursos. A programação consiste na interligação de determinados pinos, enquanto que o uso de portas consiste na ligação de funções lógicas determinadas entre pinos previamente fixados para esta finalidade.

Nesta lição veremos alguns circuitos práticos que podem ser usados na divisão de frequência, sendo, entretanto, interessante definir dois termos importantes que usaremos muitas vezes na definição das características destes circuitos.

a) Módulo - é o valor n ou valor máximo que um contador pode con-

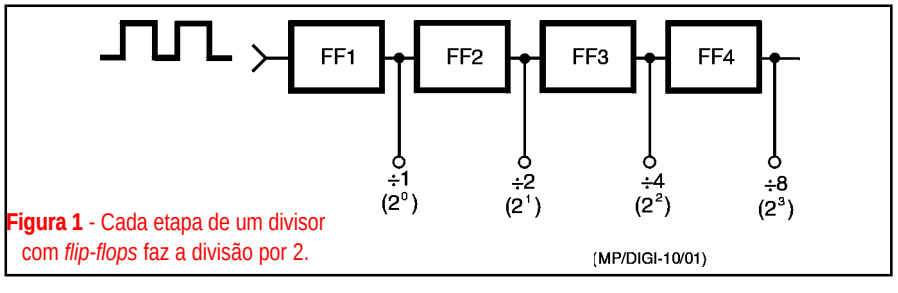


Figura 1 - Cada etapa de um divisor com *flip-flops* faz a divisão por 2.

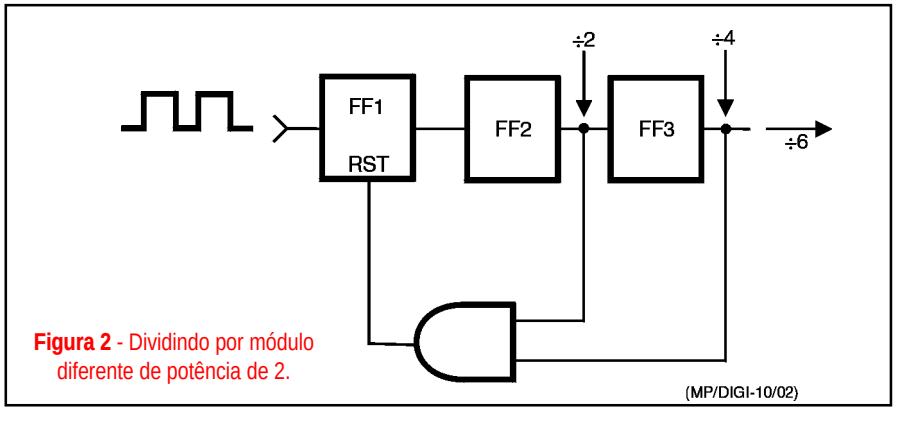


Figura 2 - Dividindo por módulo diferente de potência de 2.

tar. Por exemplo, um contador de módulo 8 é um contador que pode contar até 8 ou dividir uma frequência por valores até 8.

Se o contador tiver um módulo fixo, ele só pode dividir por este valor. No entanto, se o contador tiver um módulo variável, poderá dividir ou contar valores de 2 até este valor n. Conforme estudamos na lição anterior, o valor máximo até onde um contador pode ir é dado pelo número de *flip-flops* usados, verifique a **figura 3**.

b) Peso - num contador com saídas nos diversos *flip-flops*, a saída de cada um tem um certo "peso" na determinação do valor binário obtido na contagem.

Assim, para o circuito da **figura 4**, a saída QA tem peso 1, pois ela só pode variar entre 0 e 1. A saída QB por outro lado, tem peso 2, pois representa valores entre 0 e 2. A terceira saída (QC) tem peso 4, podendo significar valores 0 ou 4 da contagem, enquanto que QD tem peso 8, significando valores 0 ou 8, conforme esteja no nível baixo ou alto.

Assim, conforme vimos pelas tabelas verdade dos contadores, os níveis destas saídas dão o valor em binário da quantidade de pulsos de entrada contados.

c) Decodificação - alguns contadores que estudamos, como o 4017, possuem saídas decodificadas, pois elas não correspondem a valores em binário, mas sim representados de outra forma.

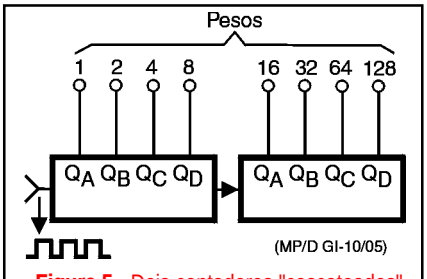


Figura 5 - Dois contadores "cascateados" para obtenção de módulo maior. Por este circuito o módulo é 16 x 16 = 256.

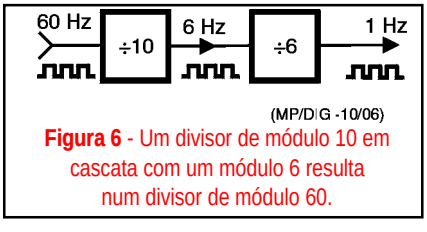


Figura 6 - Um divisor de módulo 10 em cascata com um módulo 6 resulta num divisor de módulo 60.

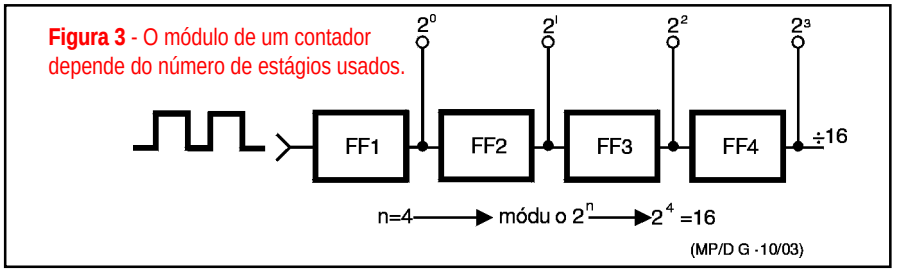


Figura 3 - O módulo de um contador depende do número de estágios usados.

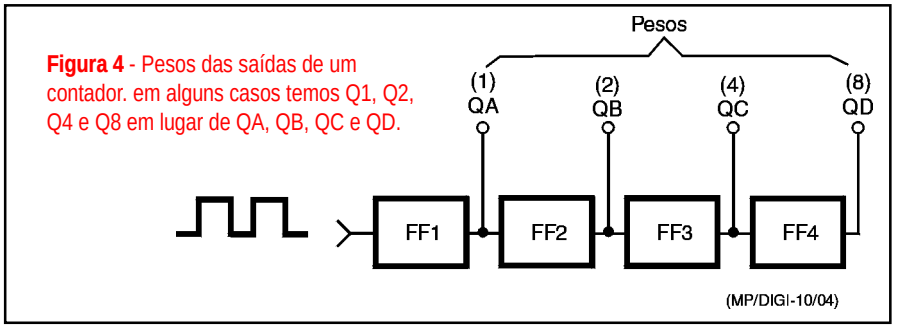


Figura 4 - Pesos das saídas de um contador. em alguns casos temos Q1, Q2, Q4 e Q8 em lugar de QA, QB, QC e QD.

No caso do 4017, a saída é decodificada para 1 de 10, no sentido de que apenas uma delas está no nível alto para cada número da contagem.

d) Cascateável - A ligação em cascata ou um após outro é importante quando desejamos fazer a contagem até valores que um único circuito integrado não alcance.

Assim, dizemos que os contadores são "cascateáveis" quando podem ser ligados da forma indicada, mostrada na **figura 5**.

Quando ligamos contadores em cascata, o módulo final obtido passa a ser o produto dos módulos dos contadores associados. Por exemplo, ligando um contador/divisor de módulo 10 em cascata com um de módulo 6, obtemos um contador/divisor de módulo 60, **figura 6**.

Esta é uma configuração muito usada em relógios digitais que produzem um pulso por segundo (1 Hz), dividindo a frequência da rede (60 Hz) por 60.

10.2 - CIRCUITOS PRÁTICOS

Daremos a seguir uma série de circuitos práticos de divisores usando circuitos integrados TTL e CMOS, que podem ser usados em projetos em que se deseja fazer a divisão ou contagem em diversos módulos a partir de 2.

a) Divisor por 2

Os dois circuitos mostrados na figura 7, com base nos circuitos integrados TTL 74107 e 7474, que contém *flip-flops* J-K e tipo D, fazem a divisão da frequência de entrada por 2.

Observe que o primeiro circuito dispara na transição negativa do sinal de *clock,* enquanto o segundo dispara na transição positiva do sinal de *clock*.

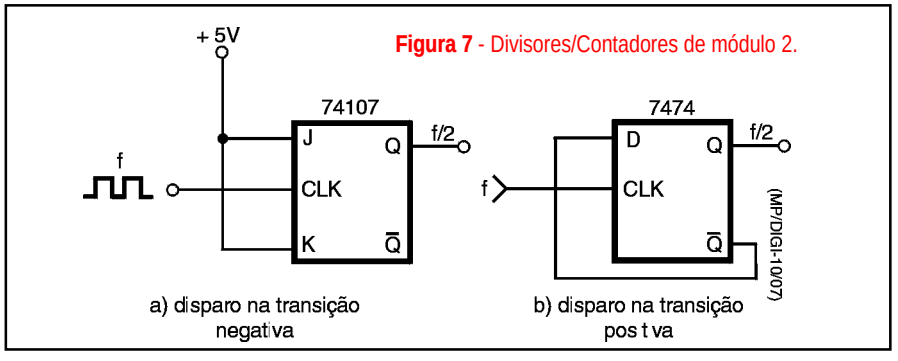


Figura 7 - Divisores/Contadores de módulo 2.

b) Divisores por 3

Divisores por 3 com base em *flip-flops* TTL e portas são mostrados a seguir. O primeiro, mostrado na figura 8, usa dois *flip-flops* do 74107 e uma porta NAND 7400.

Este circuito foi estudado na lição anterior, consistindo num contador decodificado com saída 1-de-3.

O segundo é mostrado na figura 9 e faz uso do mesmo circuito integrado 74107 e duas portas NOR do 7402.

Este circuito se caracteriza por ter uma saída simétrica, ou seja, com ciclo ativo de 50%.

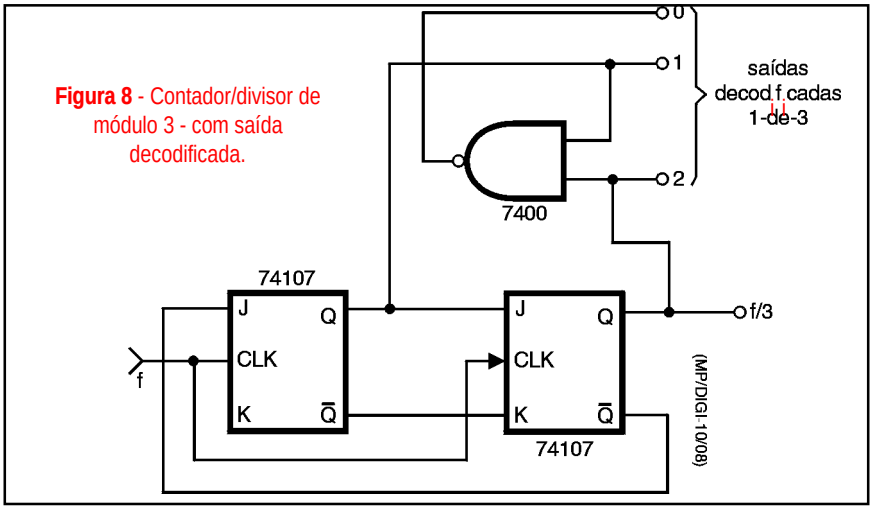


Figura 8 - Contador/divisor de módulo 3 - com saída decodificada.

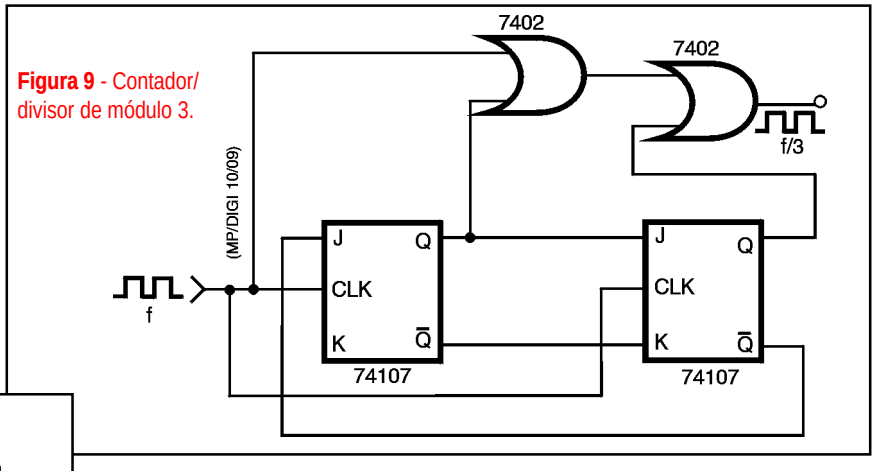


Figura 9 - Contador/divisor de módulo 3.

c) Divisores por 4

Na figura 10 temos três circuitos práticos que permitem fazer a divisão ou contagem até 4. Todos eles se baseiam em circuitos integrados TTL comuns, que já estudamos na lição anterior.

d) Divisores por 5

Usando circuitos integrados TTL e CMOS, temos diversas possibilidades de implementar divisores de frequência ou contadores de módulo 5.

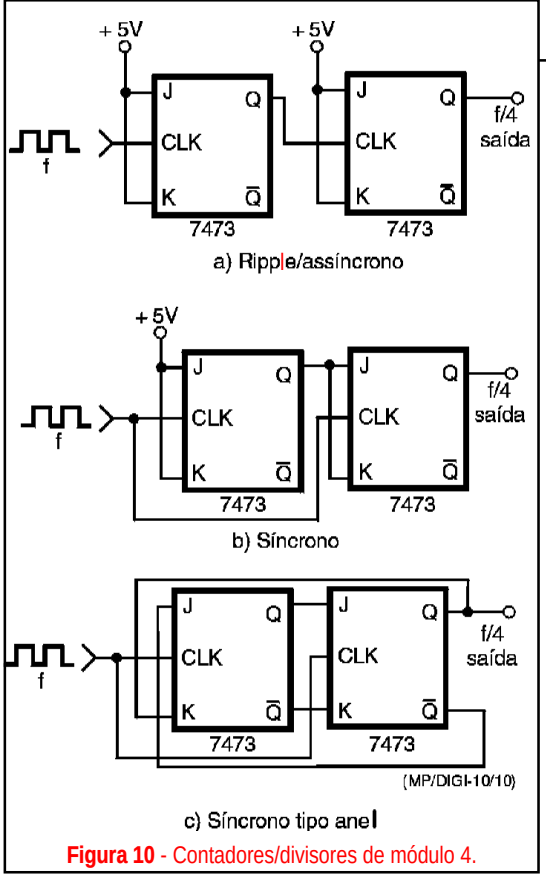


Figura 10 - Contadores/divisores de módulo 4.

Quatro destes circuitos são mostrados na **figura 11**.

Observe que o circuito 7490 é usado de forma direta, pois, como vimos, ele já possui internamente um divisor por 5. Este circuito tem algumas desvantagens que podem ser superadas com o uso de versões mais modernas como o 74290 e 74293.

O circuito com o 4018 é interessante, pois este componente é um contador "programado". Basta aplicar nas entradas de programação (L) o número na forma binária para o qual se deseja fazer a divisão.

Por exemplo, para dividir por 5 (0101), basta levar as entradas L_2 e L_4 ao nível baixo e as entradas L_1 e L_3 ao nível alto, pois este circuito é um *"down counter"*.

Observe no caso do 8281, que é necessário o uso de um par de resistores na entrada para a sua polarização.

e) Divisores por 6

Na **figura 12** damos quatro configurações com apenas um circuito integrado cada uma, que podem ser usadas para fazer a contagem de módulo 6.

Novamente encontramos o 4018, que apenas recebe a programação apropriada nas entradas L, conforme vimos no caso anterior e o 7490, que é bastante versátil neste tipo de aplicação. As características obtidas em cada caso são especificadas junto ao circuito correspondente.

Observe também os tipos de sinais usados para fazer o chaveamento de cada configuração, já que algumas disparam com a transição positiva do sinal de *clock*, enquanto outras disparam com a transição negativa do sinal de *clock*.

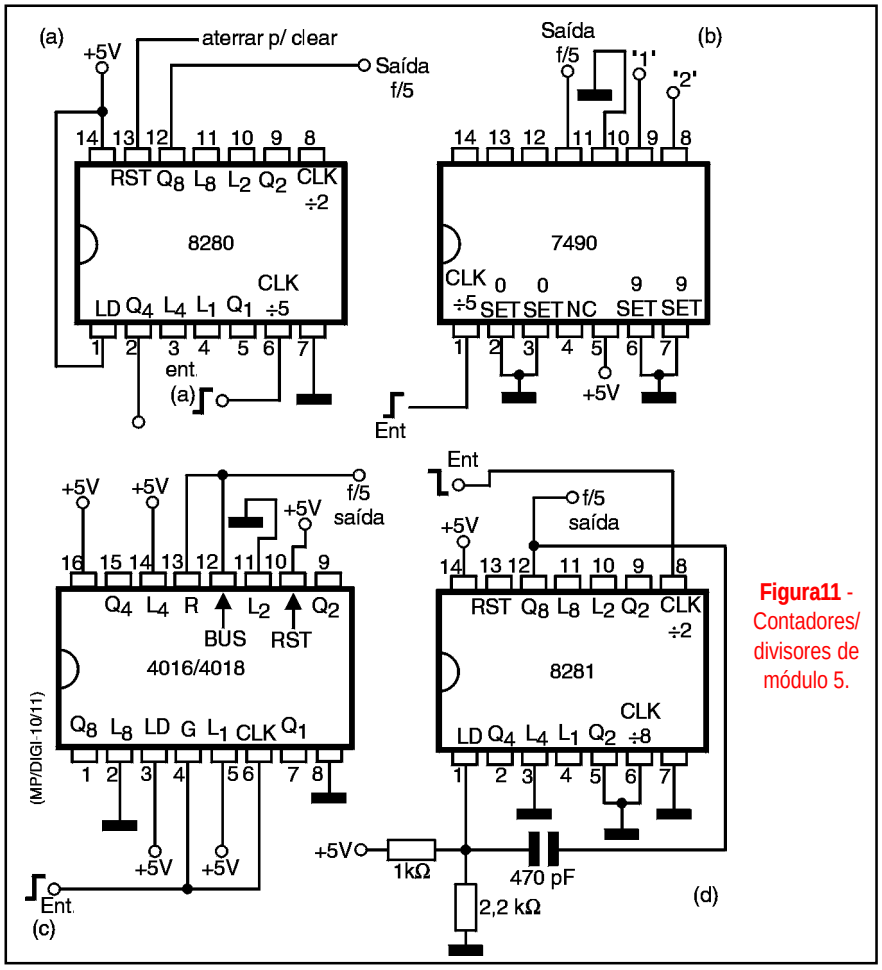


Figura11 - Contadores/ divisores de módulo 5.

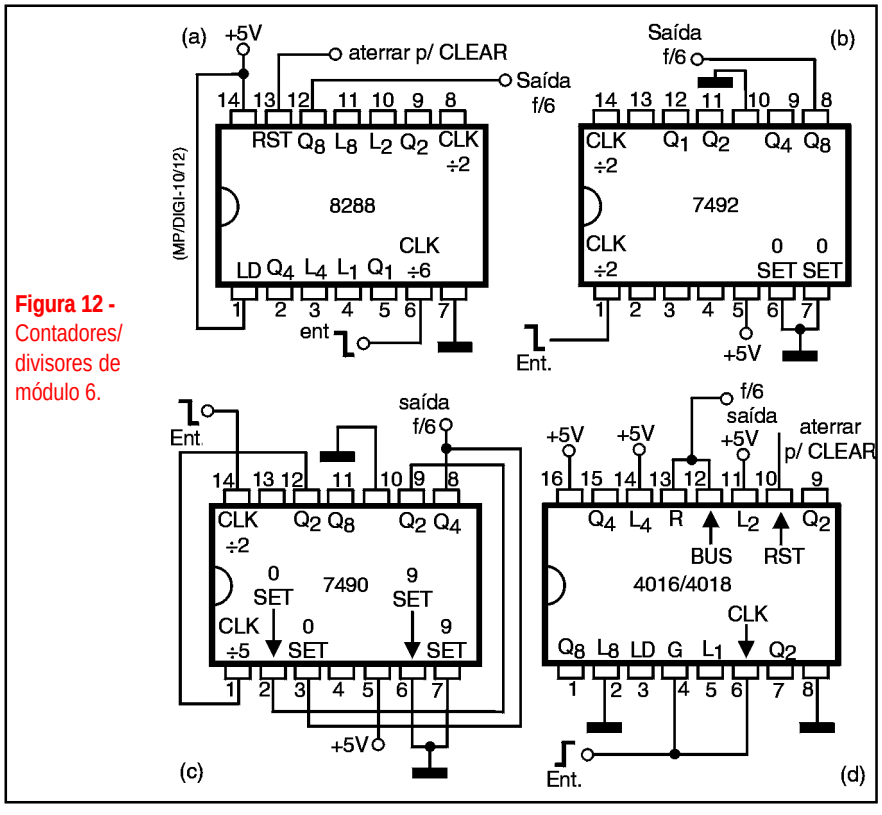


Figura 12 - Contadores/ divisores de módulo 6.

f) Divisores por 7

A divisão ou contagem em módulo 7 pode ser feita basicamente com os mesmos circuitos que usamos para o caso do módulo 6. Estes circuitos são mostrados na **figura 13**.

Veja que neste caso, em dois deles, precisamos usar portas externas para obter a divisão pelo módulo desejado.

Um tipo de funcionamento interessante é o usado no caso do 4018, que conta regressivamente. Neste circuito ele conta a partir de 7 até 0 e quando chega ao zero, salta novamente para 7, recomeçando a contagem.

Para o 74161, temos também uma modalidade de funcionamento bastante interessante: este circuito começa a contagem em 8 e vai até 15. Quando ele chega a esta contagem, o circuito recomeça, mas do pulso 8, de modo que no fundo temos a divisão por 7 como desejado.

Observe também o tipo de sinal de disparo de cada um dos tipos e as principais características indicadas junto a cada configuração.

g) Divisores por 8

Na figura 14 temos quatro circuitos de contadores/divisores de módulo 8 usando circuitos integrados TTL e CMOS.

Em cada bloco temos o tipo de disparo do circuito.

Assim, temos três configurações em que o disparo ocorre na transição negativa do sinal de *clock* e um circuito em que esse disparo ocorre na transição positiva.

Nas aplicações práticas, é muito importante observar qual é o tipo de sinal que fará o disparo, principalmente, nas que operam com lógica sincronizada.

Para os circuitos integrados 8281 e 7493, a contagem até 8 é normal, pois esses consistem em divisores com este módulo. No entanto, para o 8280 é preciso fazer uma programação. Assim, ele conta de 0 até 8 e quando chega em 8, volta novamente a zero.

h) Divisores por 9

Os circuitos contadores/divisores com módulo 9 são mostrados na **figura 15**.

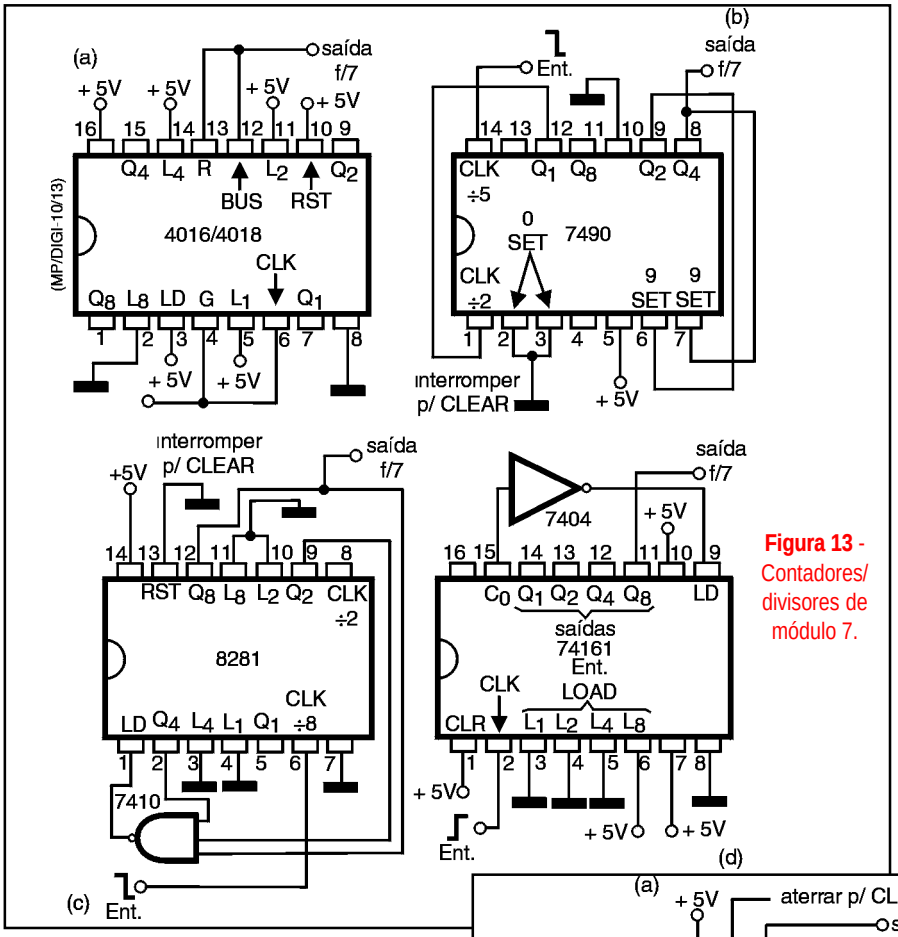


Figura 13 - Contadores/ divisores de módulo 7.

A solução mais simples para obter um divisor por 9 consiste em ligar em cascata dois divisores por 3, como os que já vimos nesta lição.

No entanto, também podemos contar com alguns circuitos integrados que podem ser programados de modo relativamente simples para fazer isso, como os apresentados na **figura 15**.

Observe que dois circuitos comutam na transição positiva do sinal e dois circuitos comutam na transição negativa.

Veja também que em duas das configurações precisamos usar portas externas para obter o módulo desejado de contagem ou divisão.

Em todos os circuitos, o princípio de operação é o já visto na lição anterior: detecta-se o estado de contagem 9 para fazer o zeramento da contagem.

i) Divisores por 10

Na figura 16 temos 5 circuitos de divisores/contadores de módulo 10 usando integrados TTL e CMOS.

Em nenhum deles é preciso usar portas ou outros componentes externos. Observe que devemos distinguir os simples divisores que fornecem uma saída com a frequência dividida por 10, dos contadores que possuem saídas com pesos 1,2,4,8 e que podem ser usados em muitas aplicações importantes, conforme veremos nas lições posteriores.

A contagem até 10 pode ser feita no sentido progressivo ou regressivo e isso é indicado em cada uma das configurações.

j) Divisores por 11

Divisores/contadores com módulo 11 podem ser elaborados com certa facilidade usando circuitos integrados comuns. Na **figura 17** temos quatro exemplos de como isso pode ser feito, destacando-se o que faz uso do 4018, que é o único que não precisa de nenhum componente externo. Conforme vimos, o 4018 é contador regressivo e basta programar sua entrada para que ele faça a divisão pelo módulo desejado, o que simplifica

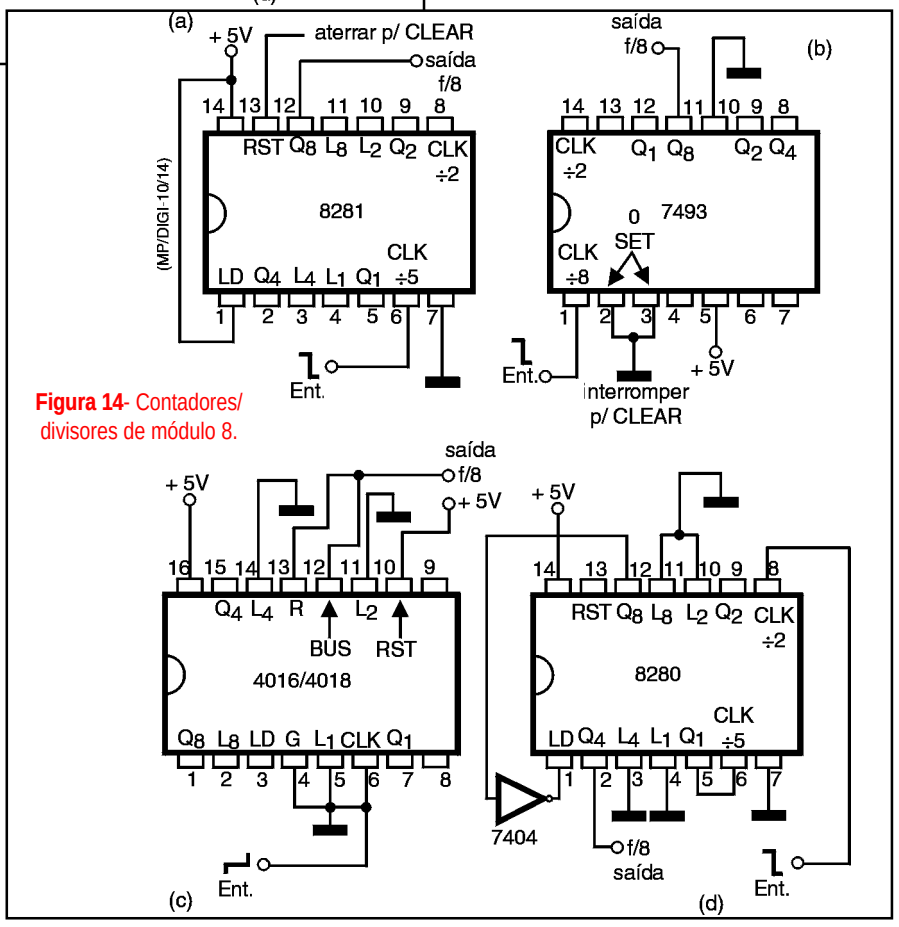


Figura 14- Contadores/ divisores de módulo 8.

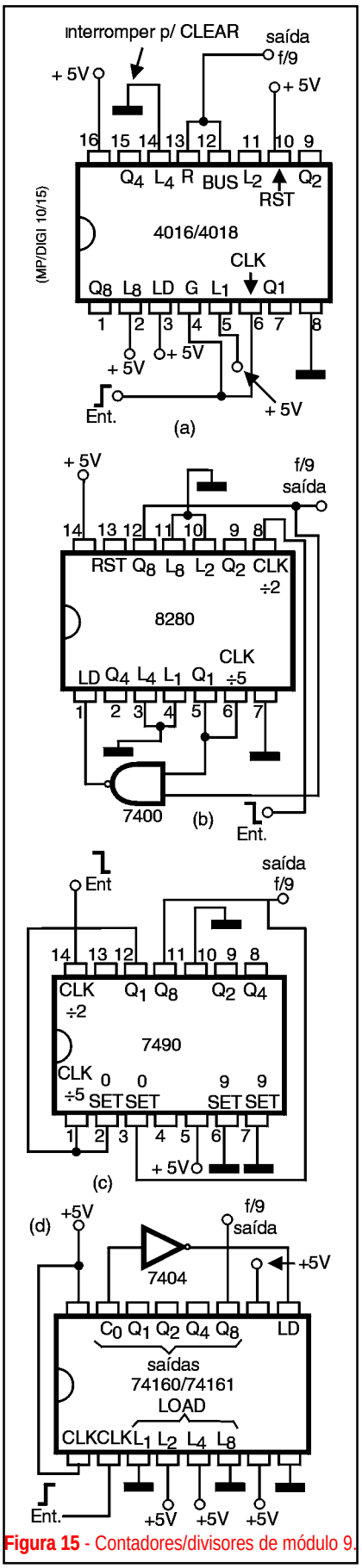


Figura 15 - Contadores/divisores de módulo 9.

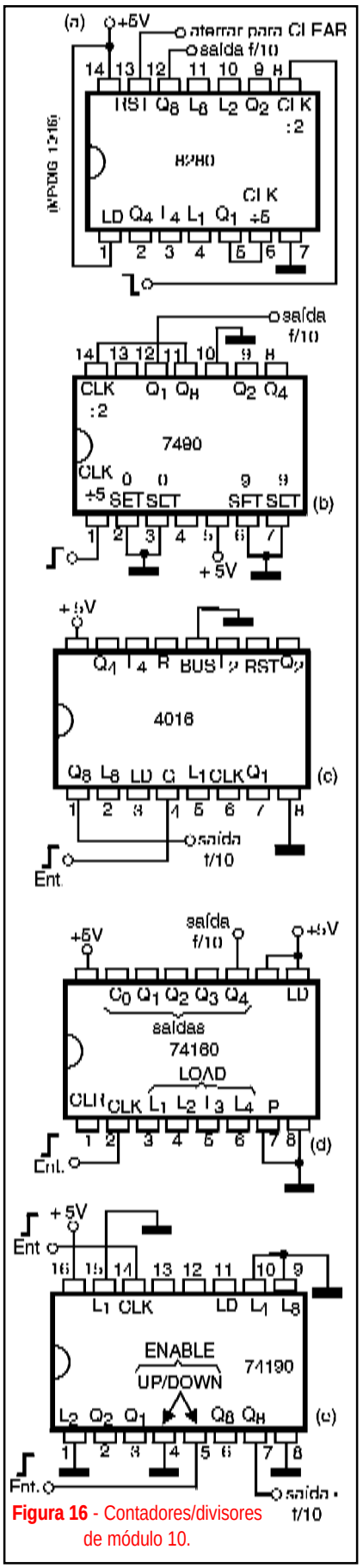


Figura 16 - Contadores/divisores de módulo 10.

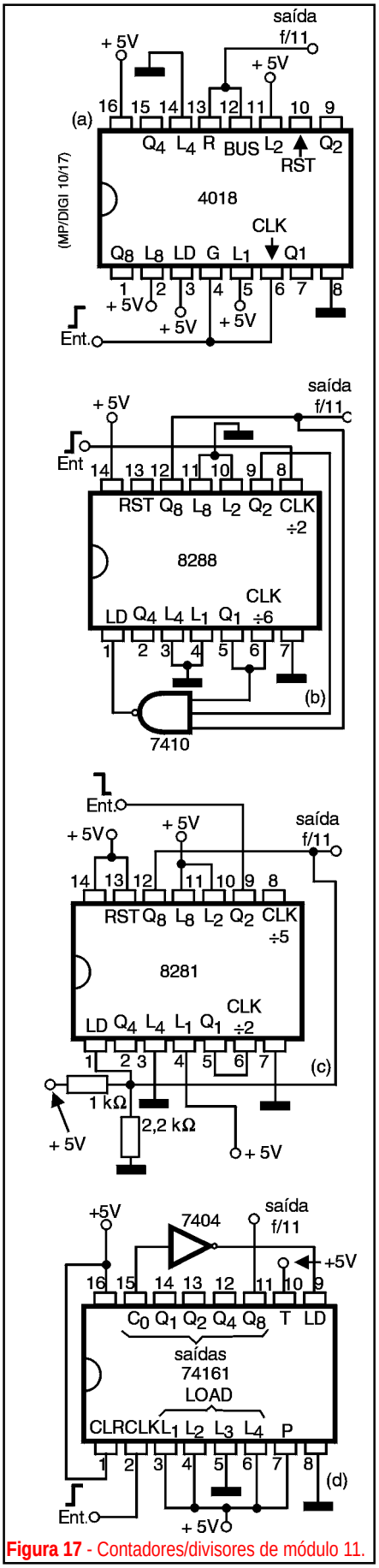


Figura 17 - Contadores/divisores de módulo 11.

bastante os projetos que fazem seu uso.

Para os demais, temos como destaque o que faz uso do 74161 e 8288 que necessitam de portas externas.

k) Divisores por 12

Quatro configurações de divisores por 12 são mostradas na **figura 18**.

Duas delas comutam na transição negativa do sinal de *clock*, enquanto que outras duas comutam na transição positiva. Observe que apenas uma delas, a que faz uso do circuito integrado 74161, necessita de um inversor externo.

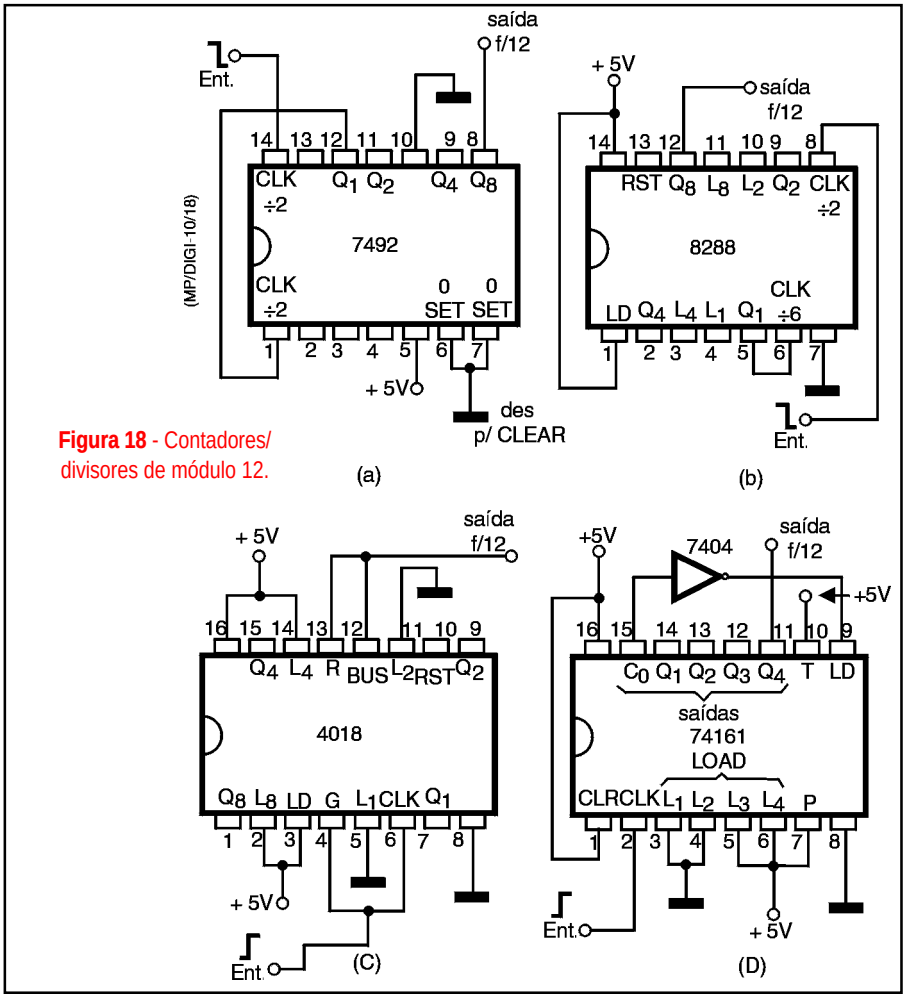


Figura 18 - Contadores/ divisores de módulo 12.

l) Divisor por 13

A divisão pelo módulo 13 pode ser feita com os dois circuitos mostrados na **figura 19**.

A mais simples é a que faz uso do contador regressivo 4018, que tem a programação digital para este valor nas entradas correspondentes. A utilização do 8281 tem por desvantagem a necessidade de alguns componentes externos adicionais.

m) Divisor por 14

A divisão por 14 pode ser feita pelos circuitos integrados 8281 e 74161 na configuração mostrada na **figura 20**.

Veja que nos dois casos precisamos usar duas funções externas para

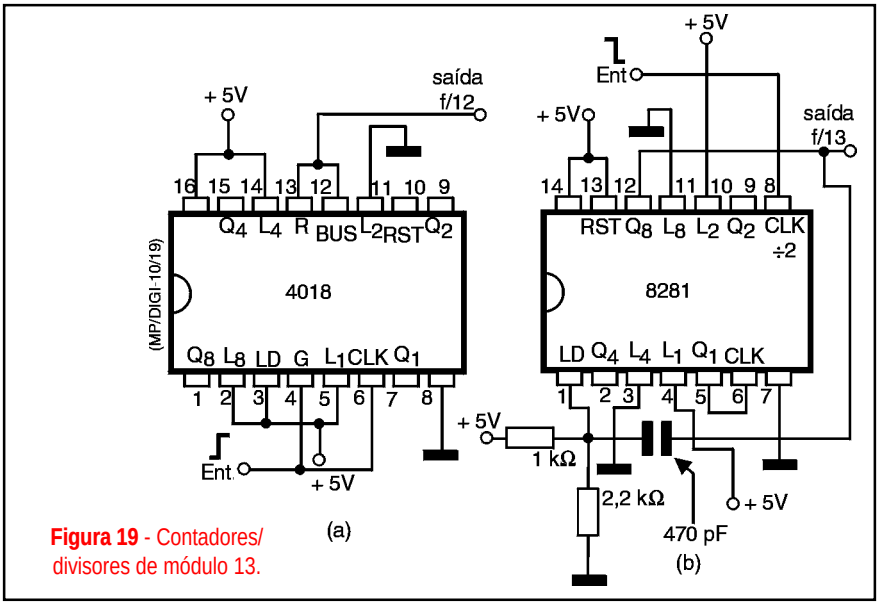


Figura 19 - Contadores/ divisores de módulo 13.

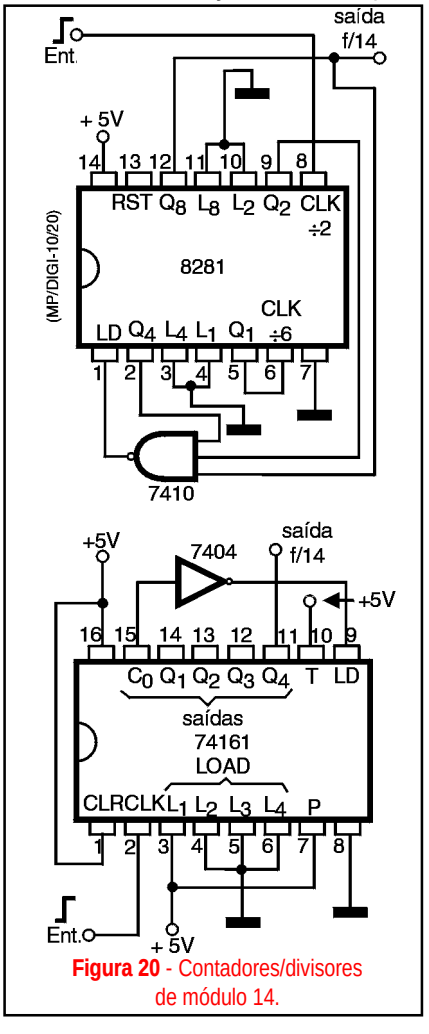


Figura 20 - Contadores/divisores de módulo 14.

obter o módulo desejado. Um dos circuitos opera com a transição positiva do sinal de *clock*, enquanto o outro opera com a transição negativa do sinal de *clock*.

n) Divisão por 15

A divisão/contagem até 15 pode ser feita com os circuitos mostrados na **figura 21**.

Com o uso do 4018 temos a configuração mais simples, já que não precisamos de nenhum componente externo, mas tão somente programar as entradas de programação para dividir pelo módulo desejado. Já com o uso do 74161 (TTL) precisamos usar um inversor externo.

Os dois circuitos operam com a transição positiva do sinal de *clock*. Em se necessitando de uma operação com a transição negativa, basta agregar um inversor na entrada.

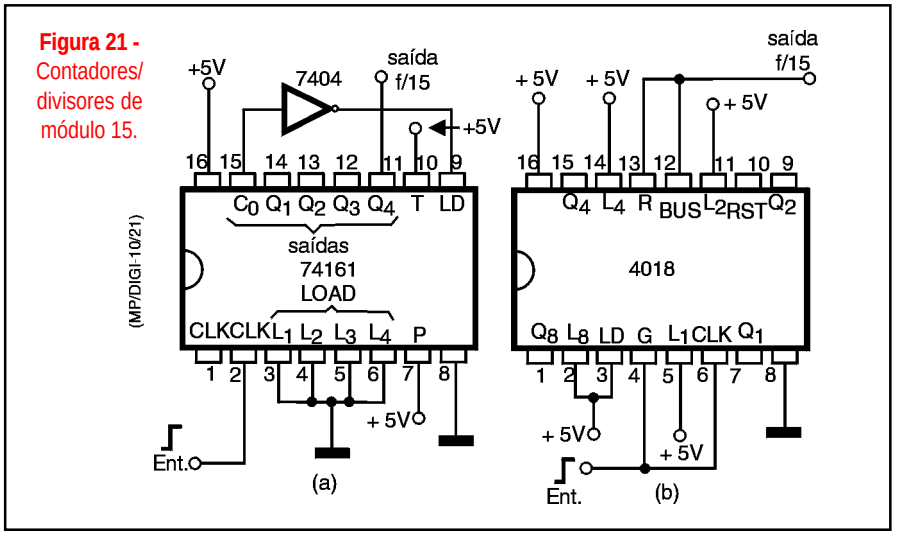
o) Divisão por 16

A divisão pelo módulo 16 é relativamente simples, pois se trata de valor normal para 4 *flip-flops* ligados em cascata. Assim, conforme observamos na **figura 22**, as configurações de divisores/contadores com este módulo são relativamente simples.

Os quatro divisores/contadores possuem saídas com pesos 1-2-4-8 acessíveis, o que pode ser muito importante nas aplicações em que se deseja a função de contador.

Dois dos circuitos operam com a transição positiva do sinal de *clock*, enquanto que outros dois operam com a transição negativa do sinal de *clock*.

Figura 21 - Contadores/divisores de módulo 15.



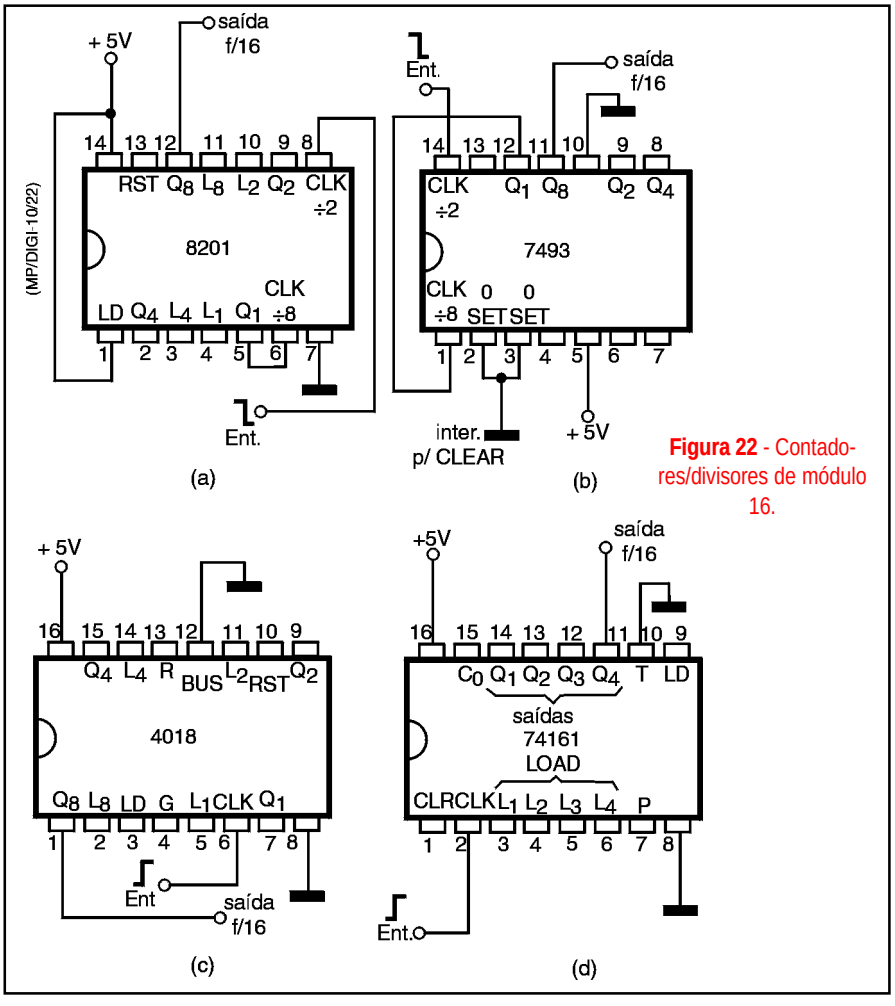


Figura 22 - Contadores/divisores de módulo 16.

QUESTIONÁRIO

1. Um contador binário tem 4 estágios. Seu módulo de contagem é:
   a) 2
   b) 4
   c) 8
   d) 16

2. Ligando em cascata um divisor de frequência por 4 e um divisor por 12 obtemos um circuito capaz de dividir a frequência por:
   a) 8
   b) 16
   d) 48
   e) 24

3. Num contador temos saídas de pesos 1-2-4-8. Aplicando um sinal de 160 Hz na entrada deste contador, qual será a frequência do sinal obtido na saída de peso 4?
   a) 20 Hz
   b) 40 Hz
   c) 80 Hz
   d) 160 Hz

Respostas:
1-d, 2-d, 3-a,

■ (digi-10)

LIÇÃO 11

COMO FUNCIONAM OS REGISTRADORES DE DESLOCAMENTO *(SHIFT-REGISTERS)*

Na lição anterior estudamos alguns divisores/contadores binários especiais capazes de fazer a divisão por qualquer módulo fixo ou programável. Vimos na ocasião que cada módulo permitia ter diversas configurações usando circuitos integrados comuns. Também estudamos divisores programáveis capazes de dividir uma frequência ou fazer a contagem em qualquer módulo, circuitos de grande utilidade em muitos projetos de Eletrônica Digital. Um elemento de grande importância nos projetos de equipamentos digitais é o registrador de deslocamento ou *shift-register*. Os *shift-registers* nada mais são do que o resultado da utilização de *flip-flops* de uma forma especial, eles são o tema desta lição.

11.1 - O QUE É UM REGISTRADOR DE DESLOCAMENTO

Um registrador de deslocamento ou *"shift-register"*, como também é chamado pelo termo em inglês, consiste num conjunto de *flip-flops* que podem ser interligados de diversas formas, como, por exemplo, as apresentadas na figura 1.

Estes circuitos podem deslocar uma informação (bit) aplicada na entrada de uma posição a cada pulso de *clock*. Por exemplo, o bit 1 aplicado na entrada aparece na saída do primeiro *flip-flop* no primeiro pulso de *clock*, depois desloca-se, aparecendo na saída do segundo *flip-flop* no segundo pulso de *clock* e assim por diante, até aparecer na saída do final da sequência, figura 2.

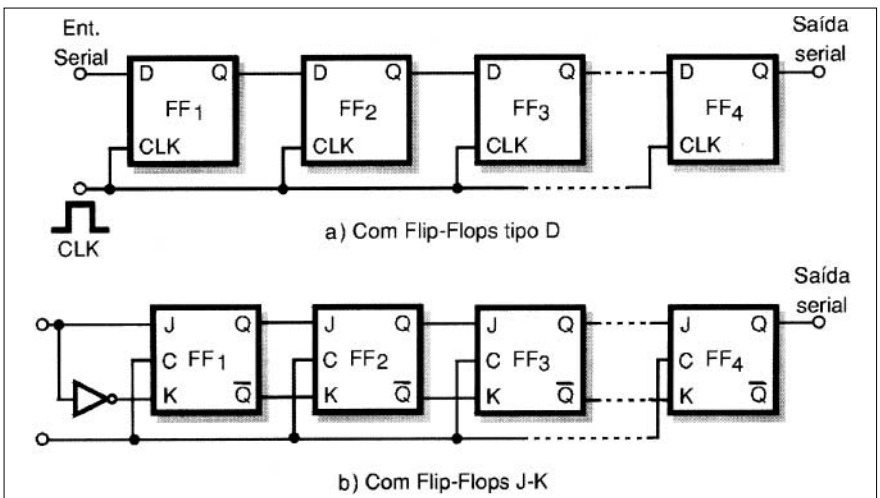


Fig. 1 - Registradores de deslocamento com *flip-flops* D e J-K.

Na configuração mostrada na figura 1 (a), cada *flip-flop* tipo D tem sua saída conectada à entrada do *flip-flop* seguinte e todos eles são controlados pelo mesmo *CLOCK*.

Para entender como funciona este circuito, vamos partir da situação inicial em que todos eles estejam desativados ou com suas saídas Q no nível baixo.

Inicialmente vamos aplicar à entrada de dados um nível alto (1). Conforme podemos ver, esta entrada é feita pela entrada J do primeiro *flip-flop* (FF_1).

Com a chegada do pulso de *clock* a este *flip-flop*, ele muda de estado e com isso "armazena" o pulso aplicado à entrada, o qual aparece em sua saída depois de um curto intervalo de tempo.

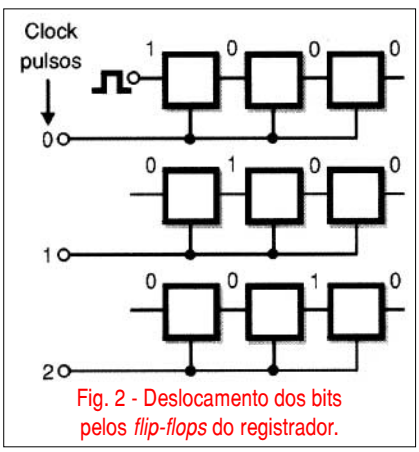


Fig. 2 - Deslocamento dos bits pelos *flip-flops* do registrador.

Veja que este sinal é armazenado com o flanco positivo do sinal de *clock*, quando então o nível alto deve estar presente na entrada do *flip-flop*. O intervalo de tempo que decorre entre a

aplicação do sinal na entrada de dados e seu aparecimento na saída do *flip-flop* é da ordem de alguns nanossegundos nos integrados das famílias lógicas comuns, mas é importante que em muitas aplicações mais rápidas ele seja levado em conta.

No próximo pulso de *clock*, ocorre algo interessante: a entrada do primeiro *flip-flop* já não tem mais o nível alto, e portanto FF_1 não muda de estado. No entanto, na saída de FF_1, temos nível alto, e esta saída está ligada à entrada do segundo *flip-flop* (FF_2). Isso significa que, com a chegada do segundo pulso de *clock*, o nível lógico da saída do primeiro se transfere para a saída do segundo, depois é claro, de um pequeno intervalo de tempo, veja a tabela I.

A sequência de bits aplicados à entrada (a) aparece na saída (b) depois de certo número de *clock*.

Isso significa que o bit 1 aplicado na entrada se "deslocará" mais um pouco no circuito, passando para a saída do segundo *flip-flop*.

É claro que, se nessa segunda passagem, tivermos aplicado um novo nível 1 na entrada do circuito, ao mesmo tempo que o primeiro se transfere para o segundo *flip-flop*, o segundo se transfere para a saída do primeiro *flip-flop*, veja a figura 3.

Chegando agora um terceiro pulso de *clock,* teremos nova transferência e o nível alto ou bit 1 se transfere para a saída do *flip-flop* seguinte, ou seja FF_3. Em outras palavras, a cada pulso de *clock*, os níveis existentes nas saídas dos *flip-flops*, sejam eles 0 ou 1, se transferem para o *flip-flop* seguinte.

Assim, supondo que apliquemos, em sequência, na entrada de um *shift-register* como o indicado, os níveis 0101, teremos a seguinte sequência de condições de saída para os *flip-flops* de um *shift-register* que use 4 deles:

Clock	Entrada	FF1	FF2	FF_3	FF_4
início	0	0	0	0	0
0	1	0	0	0	0
1	0	1	0	0	0
2	1	0	1	0	0
3	0	1	0	1	0
4	0	0	1	0	1

Tabela I

clock	entrada	FF1	FF2	FF3	FF4	Saída
0	1	0	0	0	0	0
1	1	1	0	0	0	0
2	A 0	1	1	0	0	0
3	0	0	1	1	0	0
4	1	0	0	1	1	0
5	0	1	0	0	1	1
6	0	0	1	0	0	1
7	0	0	0	1	0	B 0
8	0	0	0	0	1	0
9	0	0	0	0	0	1

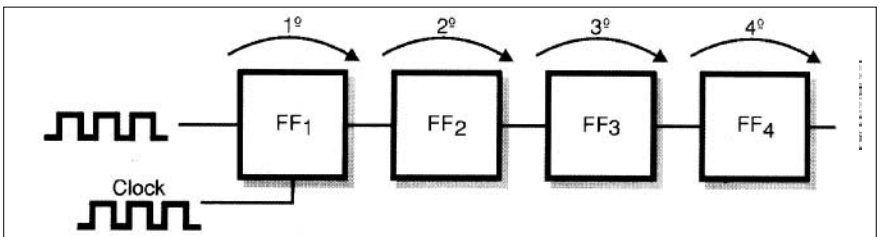


Fig. 3 - Sequência de transferências dos bits conforme os pulsos de *clock*.

Veja então que no quinto pulso de *clock*, o primeiro pulso de *clock*, o primeiro nível lógico, aparece na saída do último *flip-flop* (FF_4) e se lermos a saída dos *flip-flops* teremos registrado os níveis aplicados na entrada: 0101.

O leitor já deve ter percebido que aplicando um dado binário num *shift-register*, depois do número apropriado de pulsos de *clock*, ele pode armazenar este dado.

Para retirar o dado em sequência, basta continuar aplicando pulsos de *clock* ao circuito, conforme a seguinte tabela:

Clock	FF_1	FF_2	FF_3	FF_4	saída
início(4)	0	1	0	1	1
5	0	0	1	0	0
6	0	0	0	1	1
7	0	0	0	0	0

A figura 4 mostra o que ocorre em pormenores:

Veja então que para armazenar um dado de 4 bits num registrador devemos aplicar 4 pulsos de *clock* e para ler em sequência, mais 4 pulsos de *clock*.

Para "apagar" os dados registrados num *shif-register*, como o indicado, basta aplicar um pulso na entrada *CLEAR*. Todos os *flip-flops* terão suas saídas levadas ao nível baixo ou 0.

11.2 - TIPOS DE REGISTRADORES DE DESLOCAMENTO

Dependendo da maneira como a informação entra e como ela pode ser obtida num registrador de deslocamento, podemos ter diversas configurações que nos levam a muitos tipos de circuitos. Assim, existem circuitos em que temos uma entrada serial ou duas, e também podemos ter uma ou duas linhas de saída.

A seguir, veremos os principais tipos como suas denominações.

a) SISO - *Serial-in/Serial-out*

No exemplo, os dados foram aplicados à entrada do registrador na forma de níveis lógicos um atrás do outro, acompanhando o sinal de *clock*. Dizemos que este registrador opera com a carga de dados *"serial"* ou em

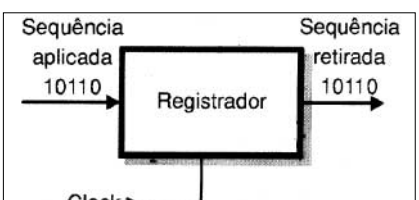
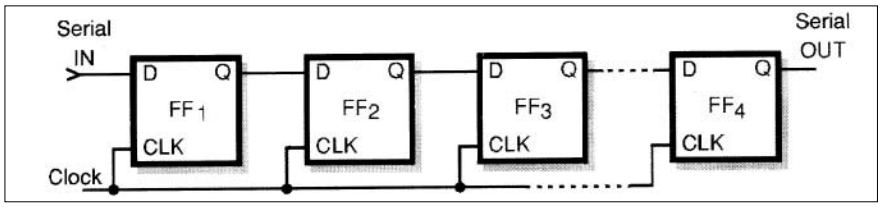


Fig. 4 - Nos registradores de deslocamento a entrada e saída podem ser serial.

série. Em outras palavras, este circuito tem entrada *serial* ou *serial-in*.

Exatamente como ocorre com a porta serial de um computador, os dados são "enfileirados" e entram um após outro e vão sendo armazenados em *flip-flops*, conforme o circuito da figura 5.



b) PISO - *Parallel-in/Serial out*

No entanto, existe uma segunda possibilidade de operação para os *shift-registers,* que é a de operar com a entrada de dados em paralelo e sair com estes mesmos dados em série. Dizemos que se trata de um *shift-register* com entrada paralela e saída serial.

Na figura 6 temos um diagrama que usa 4 *flip-flops* tipo D e que tem entrada de dados paralela e saída serial.

Analisemos como ele funciona:

Os dados são colocados ao mesmo tempo na entrada, pois ela opera em paralelo. Por exemplo, se vamos armazenar o dado 0110, esses dados são aplicados ao mesmo tempo nas entradas correspondentes (S) dos *flip-flops*.

No primeiro pulso de *clock*, os *flip-flops* "armazenam" esses dados. Assim, os *flip-flops* que possuem nível 1 em sua entrada S passam esse nível à saída (FF_2, FF_3). Por outro lado, os que possuem nível 0 na sua entrada, mantém este nível na saída (FF_1 e FF_4).

Isso significa que, após o pulso de *clock*, as saídas dos *flip-flops* apresentarão os níveis 0110.

c) SIPO - *Serial-In/Parallel-out*

Da mesma forma, como verificamos na figura 7, podemos carregar os dados em série e fazer sua leitura em paralelo.

Os registradores que operam desta forma podem ser também denominados conversores série-paralelo ou paralelo-série, conforme o modo de funcionamento.

d) PIPO - *Parallel-in/Parallel-out*

Estes são circuitos em que os dados são carregados ao mesmo tempo e depois lidos ao mesmo tempo pelas saídas dos *flip-flops*, veja a figura 8. Os registradores de deslocamento podem ainda ser classificados quanto à direção em que os dados podem ser deslocados.

Dizemos que se trata do tipo *Shift-Right,* quando os dados são deslocados para a direita e que se trata de um tipo *Shift-Left*, quando os dados são deslocados somente para a esquerda.

Existem ainda os tipos bidirecionais como o mostrado na figura 9, em que os dados podem ser deslocados nas duas direções. Este é um registrador do tipo SISO.

Veja que o sentido de deslocamento é determinado por uma entrada que atua sobre portas que modificam o ponto de aplicação dos sinais em cada *flip-flop,* exatamente como estudamos nos contadores *up* e *down* das lições anteriores.

Com a aplicação de um nível lógico conveniente na entrada *LEFT/RIGHT,* podemos determinar o sentido de deslocamento dos dados no circuito.

11.3 - OPERANDO COM BINÁRIOS

Conforme o leitor já percebeu, os registradores de deslocamento podem memorizar números binários, recebendo-os em série ou paralelo e

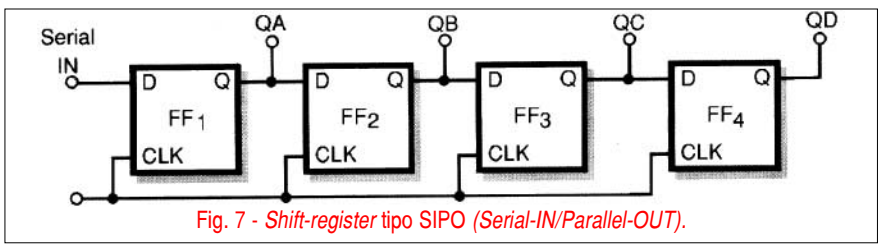


Fig. 7 - *Shift-register* tipo SIPO *(Serial-IN/Parallel-OUT).*

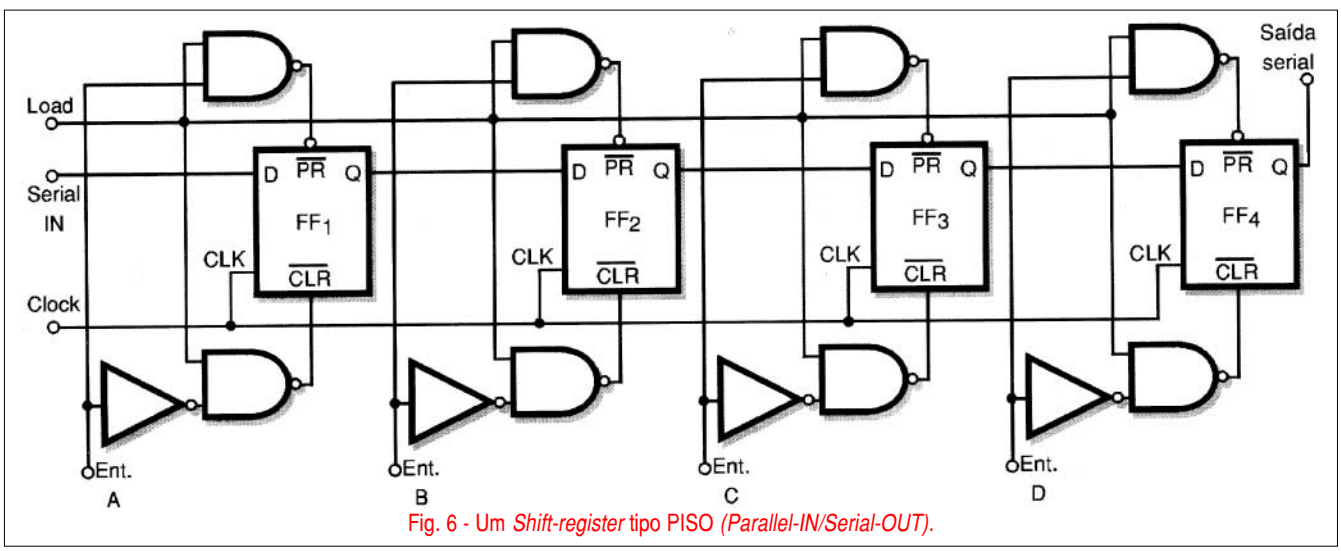


Fig. 6 - Um *Shift-register* tipo PISO *(Parallel-IN/Serial-OUT).*

entregando-os depois em série ou paralelo.

Nos computadores, esta configuração é bastante usada tanto na conversão de dados de portas como nas próprias memórias e outros circuitos internos.

É interessante observar que na configuração que tomamos como exemplo, em que são usados 4 *flip-flops*, os bits armazenados seguem uma determinada ordem.

Assim, quando representamos o número 5 (0101), cada um dos bits tem um valor relativo, que depende da sua posição no dado, conforme já estudamos em lições anteriores.

bit	0	1	0	1	
valor	8	4	2	1	
no dado	8x0	4x1	2x0	1x1	
total	0 +	4 +	0 +	1 =	5
	MSB		LSB		

MSB significa bit mais significativo, ou seja, de maior peso, enquanto que LSB significa bit menos significativo ou de menor peso.

Estamos trabalhando com dados de 4 bits, e não 8, como é comum nos computadores, obtendo assim o "byte", para maior facilidade de entendimento.

Ligando então 4 *flip-flops* de modo a obter um *shitf-register*, como observamos na figura 10, entrando com os dados de tal forma que o bit menos significativo (LSB) seja o primeiro, depois de 4 pulsos de *clock,* ele vai aparecer, na saída do último *flip-flop*.

Da mesma forma, se o *shift-register* for carregado em paralelo, o bit menos significativo (LSB) deve entrar no último, de modo que na leitura ele seja o primeiro a sair.

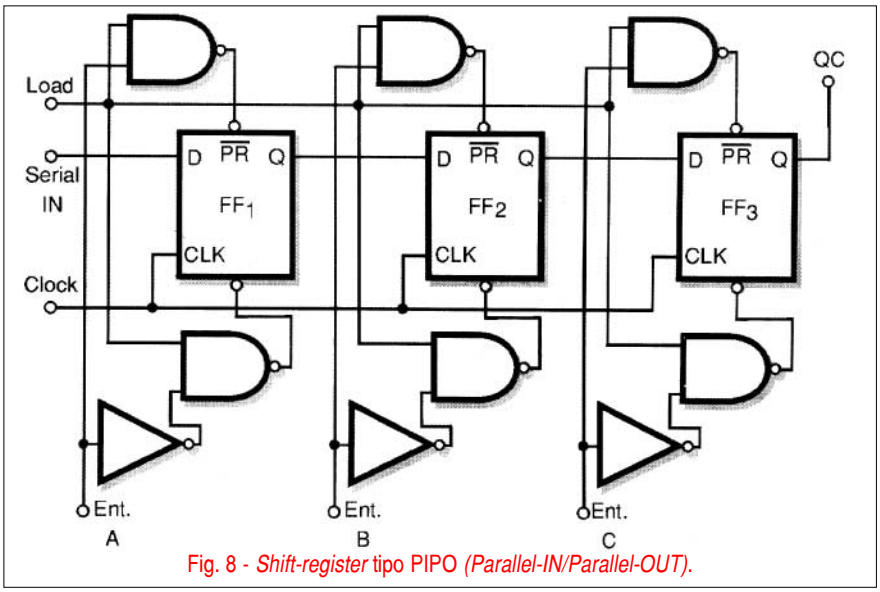


Fig. 8 - *Shift-register* tipo PIPO *(Parallel-IN/Parallel-OUT).*

11.4 - *SHIFT-REGISTERS* OU REGISTRADORES DE DESLOCAMENTO INTEGRADOS

Podemos encontrar registradores de deslocamento nas famílias TTL ou CMOs. Vamos dar alguns exemplos de circuitos integrados comuns que podem ser usados em projetos, analisando suas principais características.

7495 - *SHIFT-REGISTER* DE 4 BITS

(Da esquerda para a direita - entrada e saída em paralelo)

Este circuito integrado TTL pode operar de duas formas: *Shift* ou *Load*. Na figura 11 temos sua pinagem.

Para operar no modo *shift,* basta colocar a entrada Mode no nível baixo. Uma transição do nível alto para o nível baixo na entrada de *clock* SRT movimenta os dados de uma etapa para a direita.

Uma transição do nível alto para o baixo na entrada SLT movimenta o dado no sentido inverso.

É interessante observar que este circuito usa dois *clocks*, um para

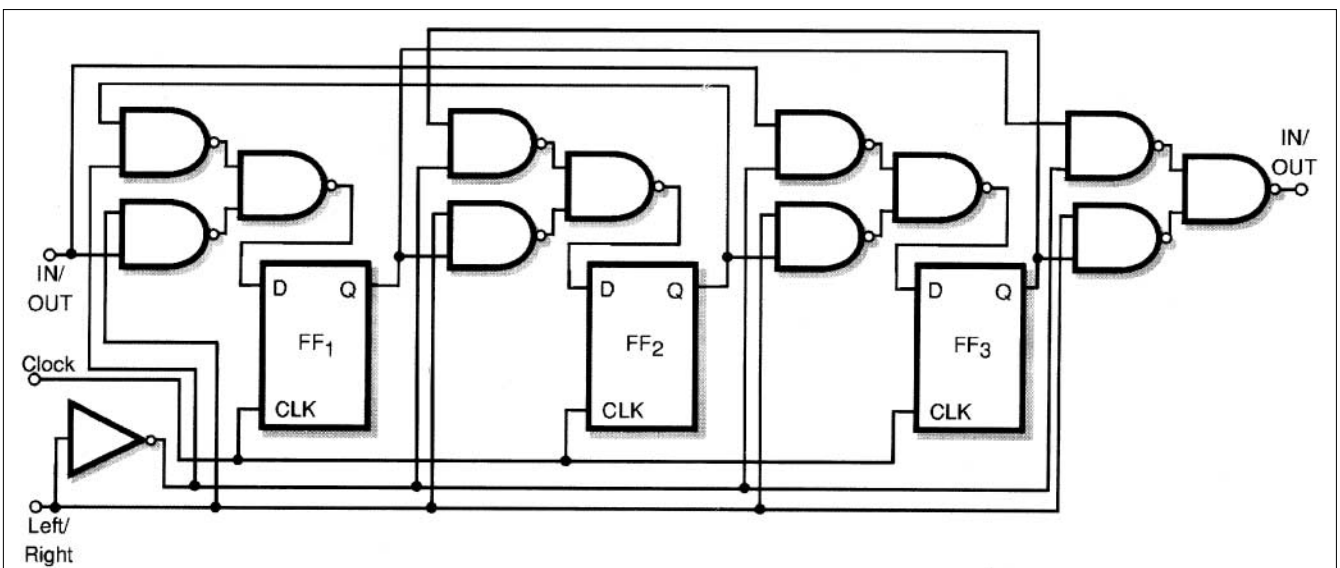


Fig. 9 - *Shift-register* bidirecional.

Fig. 10 - A ordem de entrada é a ordem de saída.

movimentar os dados para a direita e outro para a esquerda.

No modo *Load*, esta entrada deve ir ao nível alto, e a informação carregada nas entradas LA, LB, LC e LD entram no circuito na transição do nível alto para o baixo da entrada de comando na entrada *shift-left* (SLT). A frequência máxima de operação de um 7495 *standard* é de 36 MHz. Velocidades maiores de operação podem ser conseguidas com os tipos LS.

74164 - *SHIFT-REGISTER* DE 8 BITS (Entrada serial, saída paralela)

Na figura 12 temos a pinagem deste *shift register* TTL.

Este circuito pode ser usado na configuração de *serial-in/serial-out* ou *serial in/parallel-out* ou seja, entrada e saída de dados em série, ou entrada de dados em série e saída em paralelo.

Na operação normal, uma das saídas seriais é mantida no nível alto e os dados são aplicados à segunda entrada serial. A entrada *Clear* é mantida no nível alto e a cada pulso do nível baixo para o alto do *clock,* os dados movem-se de um estágio no circuito.

O conteúdo do *shift* pode ser zerado levando-se a entrada *clear* por um instante ao nível baixo.

A frequência máxima de operação deste circuito na série *Standard* é de 36 MHz.

74165 - *SHIFT-REGISTER* DE 8 BITS (Entrada Paralela, saída serial)

Este circuito integrado TTL contém um *shift-register* de 8 bits com entrada paralela e saída de dados serial. A pinagem é mostrada na figura 13.

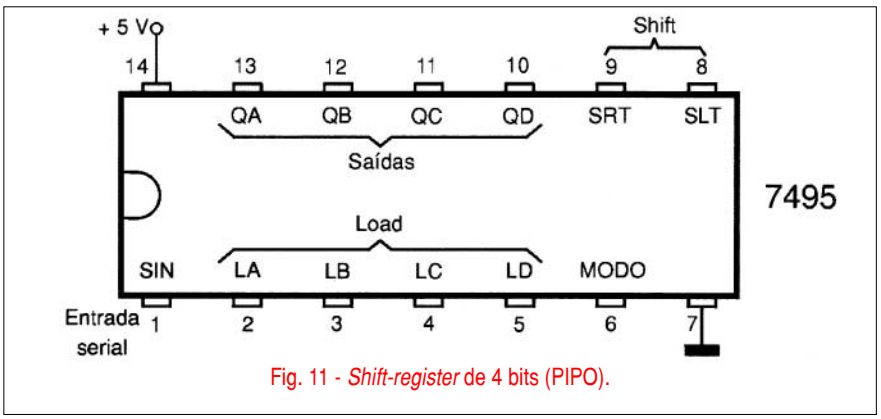


Fig. 11 - *Shift-register* de 4 bits (PIPO).

Para operação normal EN deve ficar no nível baixo e LOAD no nível alto. Nestas condições, os dados são deslocados um estágio na transição positiva do sinal de *clock*.

Quando a entrada *LOAD* é levada ao nível baixo, o conteúdo das entradas de A até H é carregado no registrador.

Fazendo *EN*=0 e *LOAD*=1 os dados são deslocados uma etapa no circuito a cada transição positiva do sinal de *clock*. A última etapa do circuito dispõe de um acesso para a saída complementar.

Damos a seguir alguns registradores de deslocamento da família CMOS.

4014 - *SHIFT-REGISTER* ESTÁTICO DE 8 BITS (Entrada paralela e saída em série)

Este circuito integrado CMOS tem a pinagem mostrada na figura 14.

Um controle série/paralelo controla a entrada e habilita as etapas individuais de cada um dos 8 estágios. As saídas Q são disponíveis nos estágios 6, 7 e 8. Todas as saídas podem fornecer ou drenar a mesma intensidade de corrente.

Quando a entrada de controle paralelo/série está no nível baixo, os dados são deslocados pelo circuito a cada transição positiva do sinal de

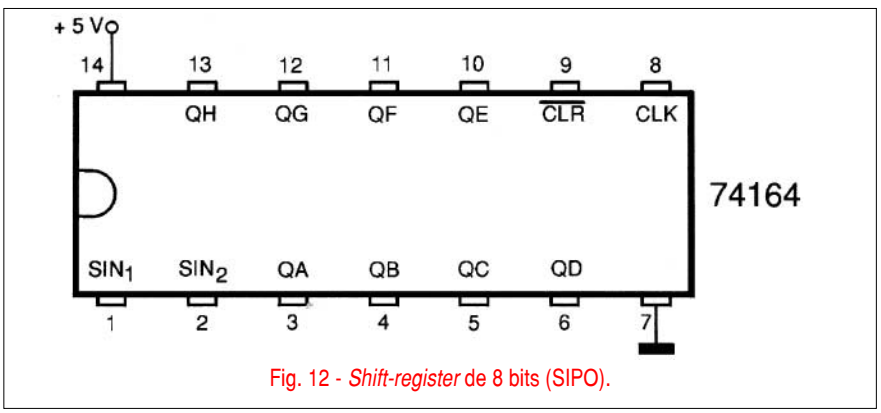


Fig. 12 - *Shift-register* de 8 bits (SIPO).

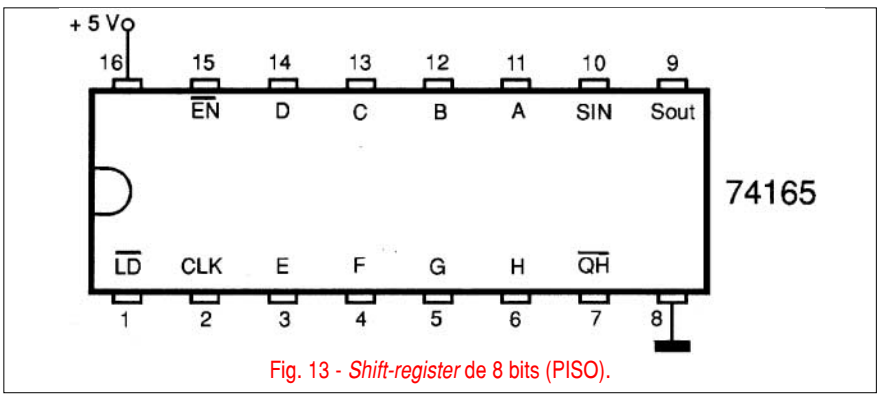


Fig. 13 - *Shift-register* de 8 bits (PISO).

clock. Quando a entrada de controle está no nível alto, os dados são aplicados a cada etapa do *shift-register* com a transição positiva do *clock*.

A frequência máxima de operação deste tipo de circuito depende da tensão de alimentação. Para uma alimentação de 10 V, esta frequência é da ordem de 5 MHz, caindo para 2,5 MHz com uma alimentação de 5 V.

4015 - DOIS *SHIFT-REGISTERS* DE 4 BITS (Entrada serial, Saída paralela)

A pinagem deste circuito fornecido em invólucro DIL de 16 pinos é mostrada na figura 15.

Neste circuito integrado encontramos dois *shift-registers* que podem ser usados de modo independente.

Na operação normal RST deve ser colocado no nível baixo. Levando esta entrada ao nível alto, o circuito resseta o *shift-register* correspondente, levando todas suas saídas ao nível lógico 0.

Os dados são deslocados a cada transição positiva do pulso de *clock*.

Para uma alimentação de 10 V, a frequência máxima de operação é de 5 MHz, caindo para metade com alimentação de 5 V.

4021 - *SHIFT-REGISTER* DE 8 BITS *(Parallel in, Serial out)*

Este circuito integrado, cuja pinagem é mostrada na figura 16, é semelhante ao 4014.

A diferença está no fato de que a carga *(LOAD)* pode ser feita de forma assíncrona. Isso significa que esta entrada independe do sinal de *clock*.

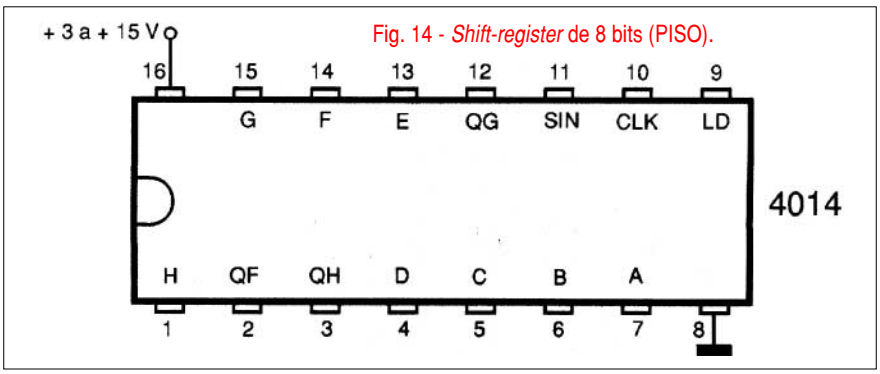


Fig. 14 - *Shift-register* de 8 bits (PISO).

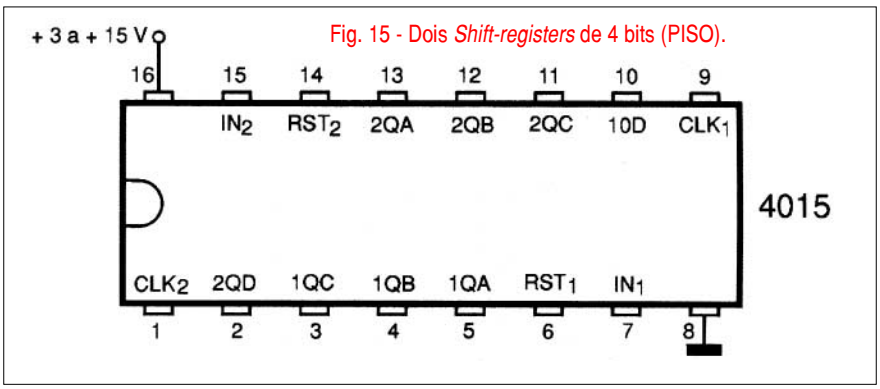


Fig. 15 - Dois *Shift-registers* de 4 bits (PISO).

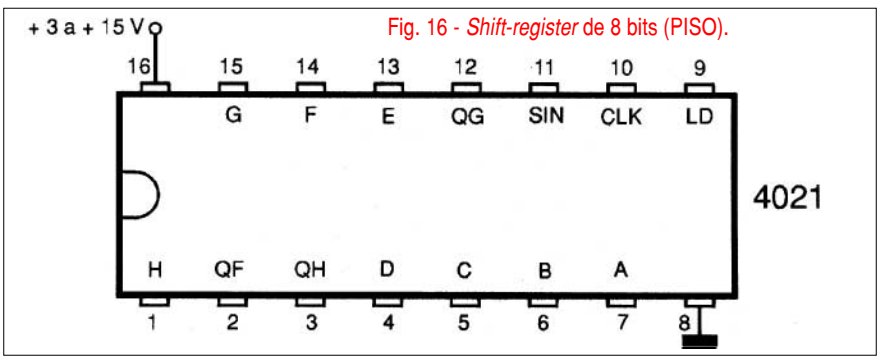


Fig. 16 - *Shift-register* de 8 bits (PISO).

QUESTIONÁRIO

1. Para obter um registrador de deslocamento, o que devemos fazer com um circuito divisor/contador digital?

a) Aterrar suas saídas complementares
b) Inverter suas saídas normais
c) Ligar sua saída à entrada
d) Não utilizar o sinal de *clock*

2. Num *shift-register* do tipo SISO temos que característica:

a) A entrada e a saída são seriais

b) A entrada e a saída são paralelas

c) A entrada é serial e a saída paralela

d) A entrada é paralela e a saída serial

3. A conversão de sinais Serial/Paralela pode ser feita por qual tipo de *shift-register*?

a) SISO
b) SIPO
c) PISO
d) PIPO

4. Para obter um contador Johnson que tipo de ligação fazemos num registrador de deslocamento?

a) Aterramos suas saídas complementares.

b) Ligamos a saída complementar do último estágio à entrada do primeiro.

c) Ligamos o *CLEAR* à entrada do primeiro estágio.

d) Ligamos o *CLEAR* à saída complementar do último estágio. ■

Respostas
1-c, 2-a, 3-b, 4-b

LIÇÃO 12

DECODIFICADORES E *DISPLAYS*

Na lição anterior estudamos os registradores de deslocamento ou *shift-registers*, analisando seu princípio de funcionamento e principais aplicações. Vimos também as pinagens e características de alguns circuitos integrados de registradores de deslocamento nas tecnologias TTL e CMOS. Nesta última lição de nosso curso, analisaremos dois blocos fundamentais para o projeto de equipamentos digitais, pois eles são responsáveis pelo interfaceamento destes circuitos com o usuário e com outros circuitos. Falaremos dos decodificadores e dos *displays*.

12.1 - OS DECODIFICADORES

As informações que os circuitos digitais produzem estão na forma binária ou em outras formas que nem sempre podem ser visualizadas facilmente pelo usuário, ou ainda que não podem ser utilizadas pelos circuitos seguintes do equipamento.

Isso implica na necessidade de termos circuitos que trabalhem uma informação codificada de modo a transformá-la em outra que possa ser usada por dispositivos ou circuitos.

Podemos ter, por exemplo, a necessidade de apresentar um valor numérico na forma decimal a partir de um valor binário ou produzir um impulso em determinado endereço numa memória a partir de uma informação binária deste endereço.

Nas aplicações digitais encontramos diversos tipos de circuitos decodificadores, estudaremos os principais nesta lição.

a) Decodificador de n para 2 elevado a n linhas

Temos nesta categoria circuitos que decodificam um sinal binário de n dígitos para uma saída de 2 elevado ao expoente n. Por exemplo, para 2 dígitos ou linhas de entrada, temos 2 x 2 linhas de saída. Para 3 linhas de entrada, temos 2 x 2 x 2 linhas de saída ou 8, e assim por diante, conforme figura 1.

Para entendermos como funciona este tipo de circuito vamos pegar sua configuração mais simples com 2 linhas de entrada e 4 de saída, usando quatro portas NAND do 7400 e dois inversores do 7404, que é mostrado na figura 2.

Este circuito ativa apenas uma das saídas a partir das quatro combinações possíveis do sinal de entrada, conforme verificamos na seguinte tabela verdade:

Entradas		Saídas			
A	B	S1	S2	S3	S4
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Veja que a saída ativada vai ao nível baixo quando o valor binário correspondente é aplicado à entrada.

Na prática não é preciso implementar circuitos decodificadores como este a partir de portas lógicas, pois existem circuitos integrados que já realizam estas funções. Daremos exemplos no final do artigo.

Aplicações possíveis para este circuito podem ser facilmente imaginadas pelos leitores.

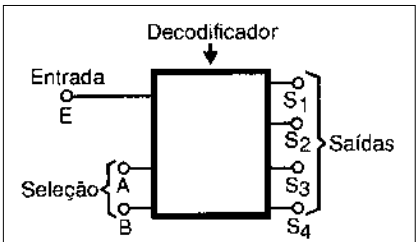


Fig. 1 - Um decodificador 1 para 4.

Na figura 3 temos um circuito em que um contador binário é ligado a um destes decodificadores de modo a fazer o acionamento sequencial de lâmpadas.

Basta ajustar a velocidade do oscilador que funciona como *clock* para determinar a velocidade do corrimento das lâmpadas, que acendem quando cada saída correspondente for ativada.

b) Demultiplexador ou DEMUX

A configuração lógica estudada no item anterior pode ser usada para realizar uma função muito interessante

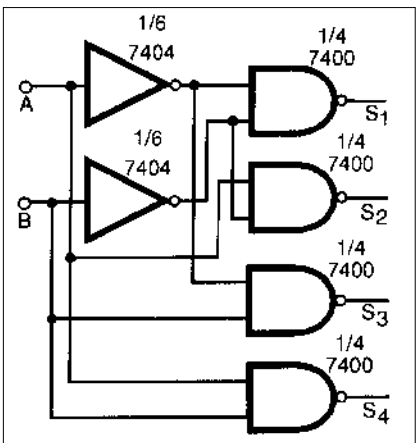


Fig. 2 - Decodificador 1 para 4 ou 1 de 4 com portas TTL.

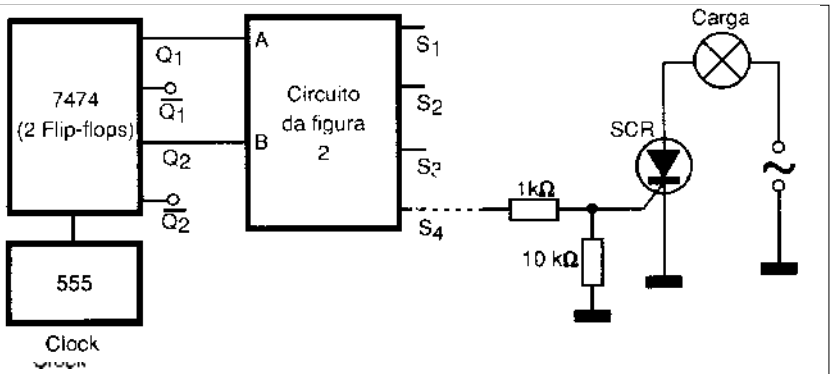


Fig. 3 - Acionando lâmpadas sequencialmente.

e útil: o direcionamento de dados num circuito.

O bloco mostrado na figura 4 ilustra o que dizemos.

O fluxo de informações (tanto analógicas como digitais) aplicado a uma entrada pode ser direcionado para qualquer uma das saídas, conforme o comando aplicado à linha de seleção de dados.

Por exemplo, se na linha de seleção de dados ou controle for aplicado o valor 10, os dados de entrada serão encaminhados para a terceira linha de saída.

Na figura 5 mostramos um circuito deste tipo implementado com portas TTL e que portanto, só funciona com dados digitais.

Neste DEMUX os dados aplicados na entrada DADOS (DATA) são encaminhados para uma das saídas (S1 a S3), conforme o "endereço" aplicado nas entradas A e B.

No entanto, os dados só podem "passar" no momento em que a entrada de habilitação EN (de *enable*) for levada ao nível alto.

A tabela verdade para este circuito é dada a seguir:

End. (AB)		Dados	EN	S1	S2	S3	S4
X	X	X	0	1	1	1	1
0	0	0	1	1	1	1	1
0	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1
1	1	0	1	1	1	1	1
0	0	1	1	0	1	1	1
0	1	1	1	1	0	1	1
1	0	1	1	1	1	0	1
1	1	1	1	1	1	1	0

X = não importa

Também é possível encontrar diversos circuitos integrados em tecnologia CMOS ou TTL que contêm estas funções, alguns operando até com sinais analógicos.

c) Multiplexadores ou MUX

Um tipo de circuito que encontra aplicações práticas importantes em Eletrônica Digital é o que realiza a função inversa a que vimos no item anterior.

Este circuito, conforme observamos na figura 6, seleciona os sinais

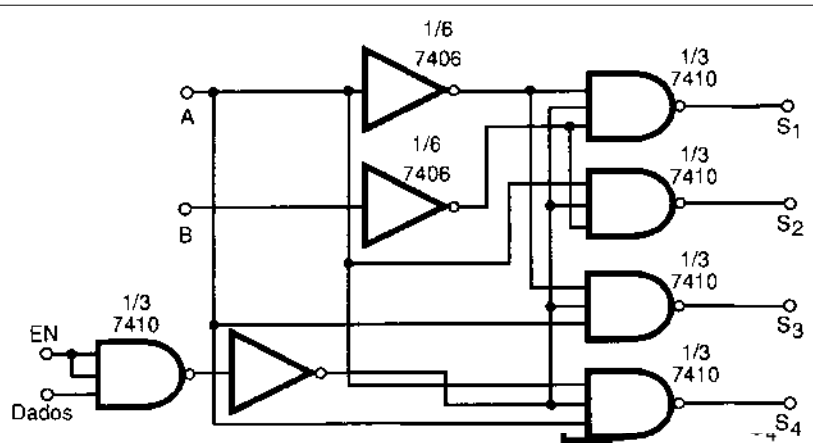


Fig. 5 - Demux com portas TTL.

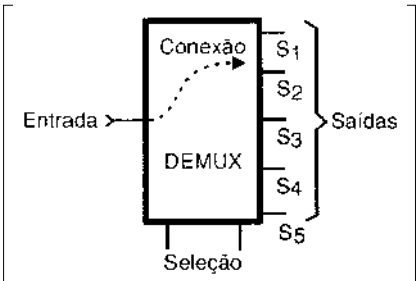


Fig. 4 - Um Demux em bloco.

de uma única entrada e aplica o nível lógico nela existente a uma saída. Em outras palavras, este circuito "lê" a informação digital presente numa saída programa e a transfere para a saída.

Este circuito recebe o nome de multiplexador ou multiplexer (MUX).

Na figura 7 temos um exemplo de aplicação implementado com funções lógicas comuns e que trabalha com 4 entradas e uma saída.

Novamente o nível lógico existente numa das entradas é transferido para a saída selecionada pelos níveis lógicos aplicados em A e B, quando a entrada de habilitação (EN) é levada ao nível alto.

Podemos elaborar a seguinte tabela verdade para este circuito:

EN	A	B	S
0	X	X	0
1	0	0	E1
1	0	1	E2
1	1	0	E3
1	1	1	E4

X = não importa

Este tipo de função também pode ser encontrada com facilidade na forma de circuitos integrados TTL e CMOS, com número de entradas que pode variar bastante conforme a aplicação desejada.

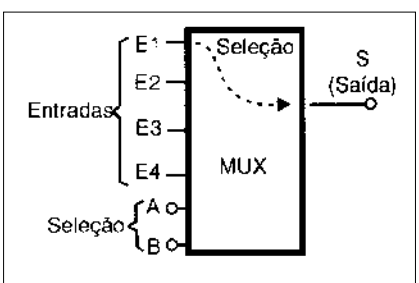


Fig. 6 - Um mux de 4 entradas (4 para 1).

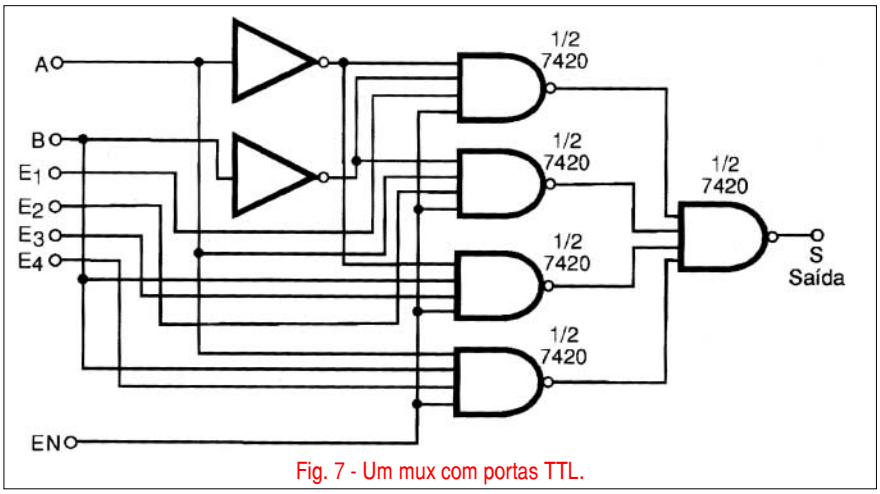


Fig. 7 - Um mux com portas TTL.

d) Decodificador BCD para 7 segmentos

Um tipo de decodificador muito usado nos projetos que envolvem Eletrônica Digital é o que faz a conversão dos sinais BCD (Decimais Codificados em Binário) para acionar um mostrador de 7 segmentos.

Podemos formar qualquer algarismo de 0 a 9 usando uma combinação de 7 segmentos de um mostrador, observe a figura 8.

Assim, se quisermos fazer surgir o algarismo 5, bastará "acender" os segmentos a, c, d, f, g, veja a figura 9.

Como os sinais codificados em binário não servem para alimentar diretamente os mostradores, é preciso contar com um circuito que faça a conversão, verifique a figura 10.

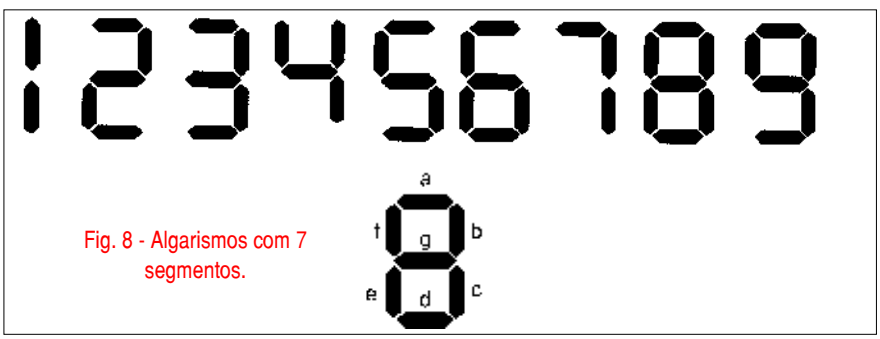


Fig. 8 - Algarismos com 7 segmentos.

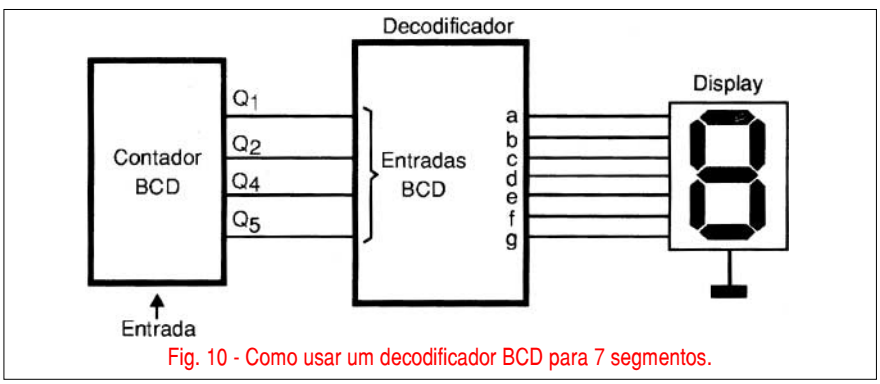


Fig. 10 - Como usar um decodificador BCD para 7 segmentos.

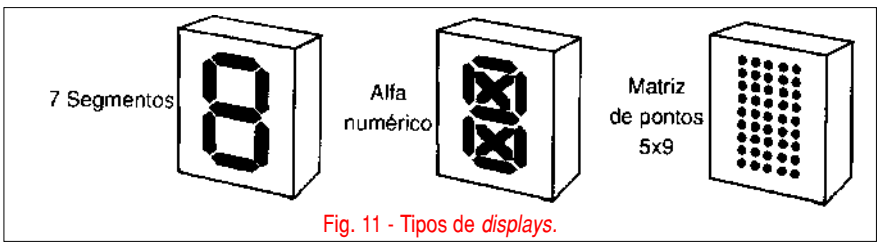


Fig. 11 - Tipos de *displays*.

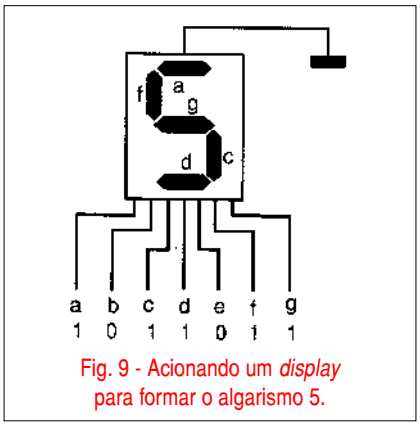


Fig. 9 - Acionando um *display* para formar o algarismo 5.

Este tipo de circuito decodificador conta com 4 entradas, por onde entra a informação BCD e 7 saídas que correspondem aos 7 segmentos de um mostrador que irá apresentar o dígito correspondente.

A combinação de níveis lógicos aplicada às entradas produzirá níveis lógicos de saída que, aplicados aos segmentos de um mostrador, fazem aparecer o dígito correspondente.

É preciso levar em conta que neste tipo de circuito, os segmentos de um mostrador podem ser ativados quando a saída vai ao nível alto ou quando a saída vai ao nível baixo. Isso dependerá do tipo de *display*, o que será estudado no item seguinte.

12.2 - *DISPLAYS*

Um *display* é um dispositivo que tem por finalidade apresentar uma informação numa forma que possa ser lida por um operador.

Podemos ter *displays* simples que operam na forma digital como sequências de LEDs, *displays* que apresentam números (numéricos), e *displays* que apresentam também símbolos gráficos (letras e sinais) denominados alfa-numéricos semelhantes aos mostrados na figura 11.

Alguns mais sofisticados podem até apresentar imagens de objetos ou formas, como os usados em equipamentos informatizados. O tipo mais comum de *display* usado nos projetos básicos de Eletrônica Digital é o numérico de 7 segmentos, de que já falamos no item anterior.

A combinação do acionamento de 7 segmentos possibilita o aparecimento dos algarismos de 0 a 9 e

também de alguns símbolos gráficos semelhantes aos apresentados na figura 12.

O tipo mais comum usado nos projetos digitais é o mostrador de LEDs, onde cada segmento é um diodo emissor de luz, sua aparência e símbolo interno são mostrados na figura 13.

Os LEDs podem ser ligados de modo a ter o anodo conectado ao mesmo ponto, caso em que dizemos que se trata de um *display* de anodo comum, ou podem ter os catodos interligados, caso em que dizemos que se trata de um *display* de catodo comum.

As correntes nos segmentos variam tipicamente entre 10 e 50 mA conforme o tipo, o que nos leva a concluir que o consumo máximo ocorre quando o dígito 8 é projetado (todos os segmentos acesos) e pode chegar a 400 mA por dígito. Alguns fabricantes podem juntar mais de um dígito num único bloco, facilitando assim os projetos, pois, na maioria dos projetos os números apresentados são maiores que 9, ver figura 14.

Outro tipo de *display* também utilizado com certa frequência nos projetos é o de cristal líquido.

Este *display* não "acende" quando excitado. Eletrodos transparentes ao serem excitados eletricamente pelo sinal do circuito fazem com que o líquido com que ele está em contato torne-se opaco, deixando assim de refletir a luz. Desta forma, o fundo branco do material deixa de ser visto, aparecendo em seu lugar uma região preta, veja a figura 15.

Fig. 12 - Símbolos gráficos em *displays* de 7 segmentos.

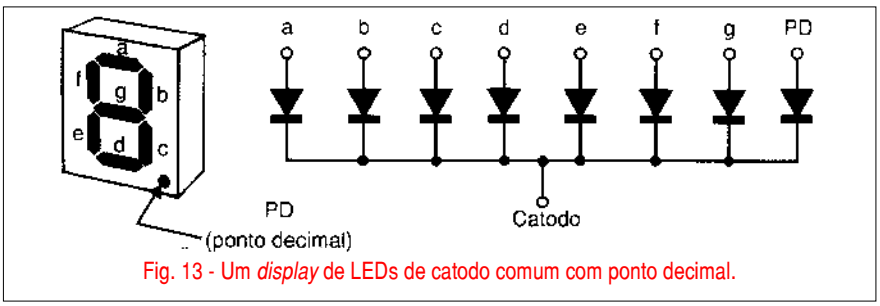


Fig. 13 - Um *display* de LEDs de catodo comum com ponto decimal.

Fig. 14 - Tipos de *displays* múltiplos.

As regiões formam os segmentos e conforme sua combinação temos o aparecimento dos dígitos.

No entanto, é mais difícil trabalhar com estes mostradores, pois eles exigem circuitos de excitação especiais que também são mais caros.

A principal vantagem do mostrador de cristal líquido (LCD) é seu consumo, que é centenas de vezes menor do que o de um mostrador de LEDs. Para as aplicações em que o aparelho deve ser alimentado através de pilhas ou ficar permanentemente ligado, é muito vantajoso usar o mostrador LCD.

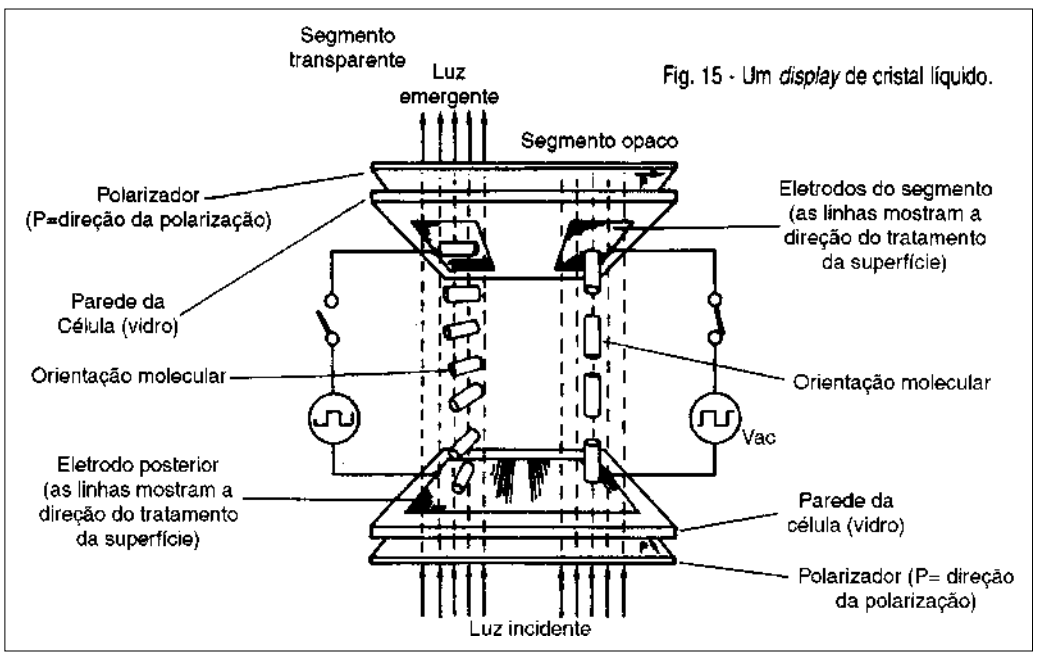


Fig. 15 - Um *display* de cristal líquido.

12.3 DECODIFICADORES E CODIFICADORES TTL E CMOS

Podemos contar com uma boa quantidade de decodificadores, multiplexadores e demultiplexadores na forma de circuitos integrados TTL ou CMOS. Será interessante para qualquer praticante de Eletrônica Digital contar com um desses manuais.

No entanto, para facilitar, decreveremos alguns circuitos integrados que contêm estas funções e são mais utilizados nos projetos e aplicações práticas.

a) 7442 - Decodificador BCD para decimal

Este circuito integrado tem a pinagem mostrada na figura 16.

Conforme a combinação de níveis lógicos das entradas (codificadas em BCD), apenas uma das saídas irá para o nível lógico baixo. Todas as demais permanecerão no nível alto.

Se os níveis lógicos aplicados às entradas tiverem a combinação 1010 até 1111 (que correspondem de 11 a 15) nenhuma das saídas será ativada. Quando ativada, cada saída pode drenar uma corrente de 16 mA.

O circuito integrado TTL 7445 tem a mesma função, com a diferença de que possui transistores na configuração de coletor aberto na saída, podendo com isso trabalhar com tensões de até 30 V e drenar correntes de até 80 mA. A pinagem é a mesma do 7442.

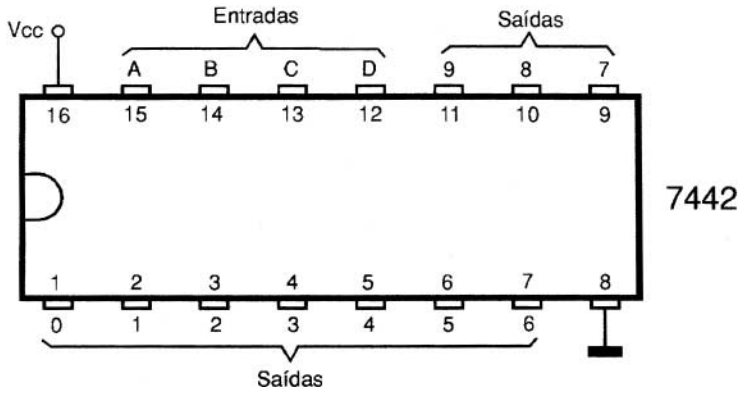


Fig. 16 - BCD para decimal - decodificador.

b) 7447 - Decodificador BCD para 7 Segmentos

Este é um circuito TTL que possui saídas em coletor aberto capazes de drenar correntes de até 40 mA, sendo portanto indicado para excitar *displays* de LEDs de anodo comum.

Na figura 17 temos a sua pinagem.

Algumas características importantes devem ser observadas neste circuito.

Uma delas é o terminal *Lamp Test* ou teste do *display*. Colocando esta saída no nível lógico baixo (em funcionamento normal ela deve ser mantida no nível alto) todas as saídas vão ao nível baixo, fazendo com que todos os segmentos do *display* acendam. Com isso é possível verificar se ele está em bom estado.

Outra saída importante é a RBI *(Ripple Blank Input)* que faz com que os zeros à esquerda sejam apagados quando são usados diversos contadores, figura 18.

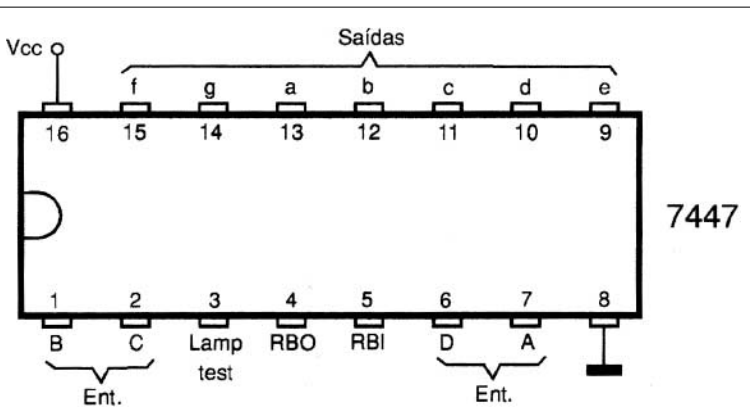


Fig. 17 - Decodificador BCD para 7 segmentos.

Assim, em lugar de aparecer o valor 008, numa contagem aparece apenas 8.

Observe que a saída RB0 *(Ripple Blank Output)* serve para a ligação em série de diversos blocos contadores de modo a ser obtido um conjunto com vários dígitos.

c) 74150 - Seletor de dados 1-de-16

Este circuito integrado TTL consiste num multiplexador que possui 16 linhas de entrada e uma saída selecionadas pelas Linhas de Seleção. Na figura 19 temos a pinagem deste circuito integrado.

Para operação normal, a entrada de habilitação (EN) deve ser mantida no nível alto até o momento em que

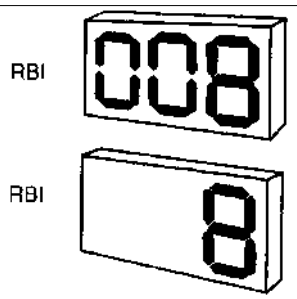


Fig. 18 - Usando a função RBI *(Ripple Blank Input).*

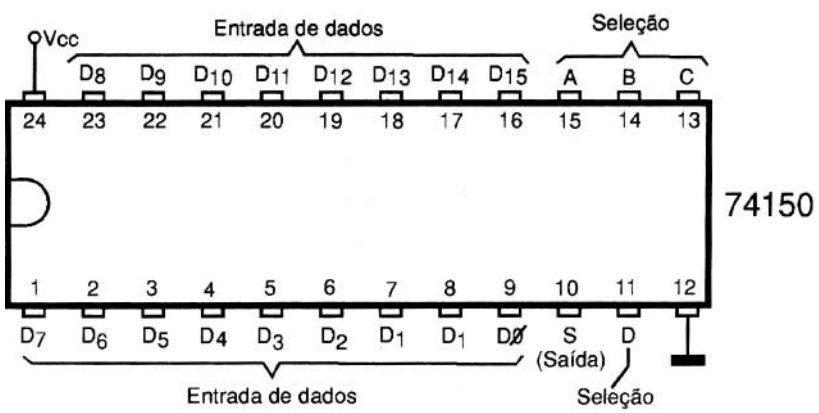


Fig. 19 - Seletor de dados 1 de 16.

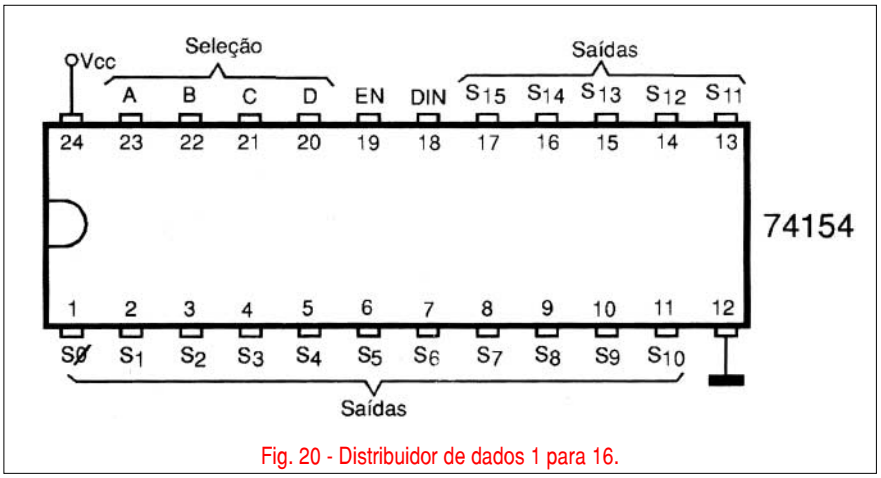


Fig. 20 - Distribuidor de dados 1 para 16.

os dados de uma determinada entrada devam ser levados para a saída. Qual entrada será ativada depende do código aplicado à linha de seleção. O circuito possui duas saídas. Numa delas aparece o sinal da entrada selecionada e na outra, o sinal complementar.

Circuitos semelhantes da mesma família são o 74151 que consiste num seletor 1 de 8 e o 74153 que consiste num seletor 1 de 4.

d) 74154 - Distribuidor de Dados 1-de-16

Este curcuito integrado contém um DEMUX ou Demultiplexador 1 de 16 em tecnologia TTL. Sua pinagem é mostrada na figura 20.

A entrada da habilitação (EN) deve ser mantida no nível alto até o momento em que os dados da entrada devam ser transferidos para a saída selecionada.

Os circuitos integrados 74157 são distribuidores semelhantes, mas 1-de-2 e o 74155 1-de-4.

e) 4028 - Decodificador BCD para Decimal

Este é um circuito integrado CMOS com 10 saídas, onde aquela que vai ao nível alto depende da combinação dos níveis de entrada. As demais saídas permanecerão no nível baixo. A pinagem deste circuito integrado é mostrada na figura 21.

As combinações de entrada entre 1010 e 1111 que correspondem aos números de 11 a 15 não serão reconhecidas e todas as saídas permanecerão no nível baixo.

f) 4051 - Chave 1-de-8

Este circuito integrado CMOS pode chavear sinais analógicos ou digitais e tem a pinagem mostrada na figura 22.

Para utilizar este circuito com sinais digitais, a tensão de alimentação positiva pode ficar entre 5 e 12 V, enquanto que o pino 7 é aterrado.

No entanto, para operar com sinais analógicos, o pino 7 deve ser conectado a uma fonte de -5 V (fonte negativa) e o pino 8 aterrado.

Nestas condições os sinais a serem chaveados podem variar entre -5 e +5 V, enquanto os sinais de seleção podem ter nível baixo (0 V) ou nível alto (5 V).

Tanto na operação com sinais digitais como analógicos, as chaves fechadas representam uma resistência de 120 Ω e não devem ser usadas cargas com resistências inferiores a 100 Ω. A corrente máxima chaveada para os sinais não deve superar os 25 mA.

Semelhantes a este circuito em características são os:

4052 - Duas chaves 1 de 4
4053 - Três chaves 1 de 2
4067 - Uma chave 1 de 16

Este último circuito integrado pode funcionar como multiplexador ou demultiplexador para sinais analógicos e digitais de modo similar aos anteriores.

g) 4026 - Contador de Década com Saída de 7 Segmentos

Este importante circuito integrado CMOS tem um contador divisor por 10 e suas saídas são decodificadas.

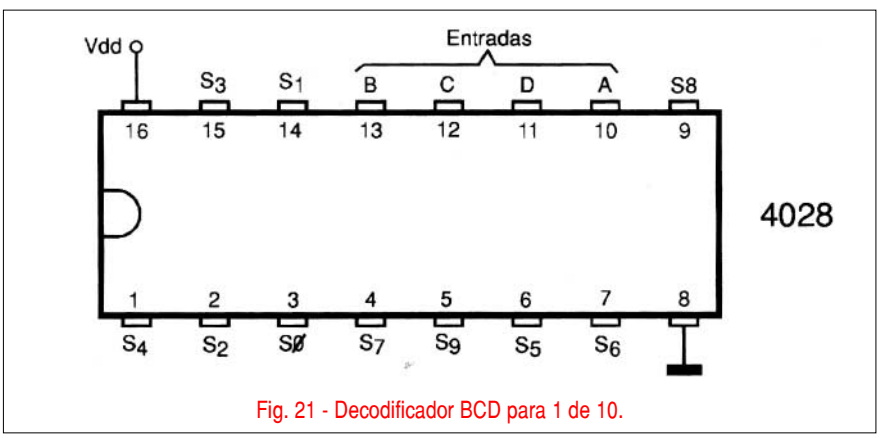


Fig. 21 - Decodificador BCD para 1 de 10.

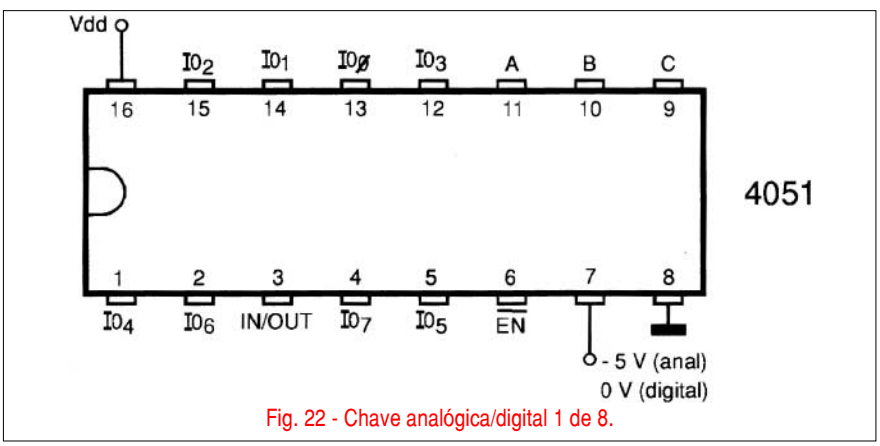
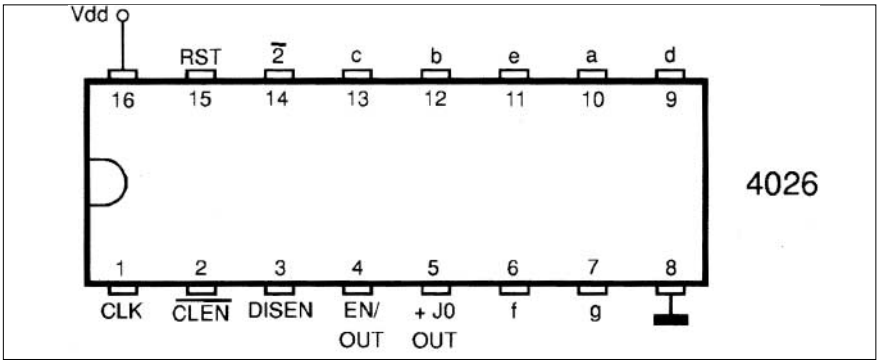


Fig. 22 - Chave analógica/digital 1 de 8.



A pinagem deste circuito integrado é mostrada na figura 23.

Na operação normal, as entradas RST *(Reset)* e *CLEN* devem ser mantidas no nível baixo. Um nível alto aplicado em RST resseta o contador, levando o valor da saída a 0 e ao mesmo tempo impede a contagem.

Um nível alto aplicado em *CLEN* (Habilitação do *Clock* ou *Clock Enable*) inibe a entrada dos sinais de *clock*. O contador é gatilhado nas transições positivas do sinal de *clock*.

No pino 5 é possível obter um sinal quadrado de 1/10 da frequência de *clock* e no pino 14 temos um sinal que permanece no nível alto até o momento em que a contagem chega a 0010, quando passa ao nível baixo.

A entrada *DISEN* serve para habilitar o *display,* devendo permanecer no nível alto na operação normal. Quando esta linha vai ao nível baixo, as saídas vão todas ao nível baixo.

Este circuito é indicado para operar com *displays* de catodo comum e a corrente de saída máxima é de 1,2 mA para uma tensão de alimentação de 5 V, e 5 mA para 10 V.

A frequência máxima de operação é de 5 MHz para 10 V de tensão de alimentação e 2,5 MHz para 5 V.

QUESTIONÁRIO

1. Um circuito que joga o sinal de uma entrada em uma de 4 saídas é denominado:
a) Multiplexador 1 de 4
b) Demultiplexador 1 de 4
c) Decodificador 4 por 4
d) Decodificador BCD para 1 de 4

2. Que tipo de decodificador tem apenas uma de 10 saídas ativadas a partir de sinais BCD de entrada?
a) Decodificador 1 de 10
b) Demux 1 de 10
c) Contador Johnson
d) Decodificador BCD para 1 de 10

3. Em que tipo de *display* os catodos de todos os LEDs dos segmentos são interligados e conectados a um ponto comum?
a) Anodo comum
b) Cristal líquido ou LCD
c) Catodo comum
d) Duplo

Resposta: 1.b 2.d 3.c