Introdução aos Sistemas de Digitais

O ser humano lida continuamente com quantidades, nos mais diversos segmentos da ciência, da tecnologia, do negócio e do comércio. Quando quantidades em geral são medidas, monitoradas, armazenadas, processadas aritmeticamente ou observadas, é necessário que a unidade de medida escolhida seja coerente com a magnitude da medida e que sua representação numérica da medida seja feita de forma precisa e eficiente.

Existem, basicamente, duas formas de representar numericamente quantidades: a analógica e a digital. Fenômenos físicos tais como tempo, velocidade, temperatura, tensão, corrente, entre outras, são variáveis analógicas por variarem continuamente dentro de faixas de valores; porém suas representações numéricas dependem da aplicação da medida, da precisão desejada e das formas de armazenamento e de visualização.

Na representação analógica, a medida varia continuamente segundo a variação da variável física ou da medida de um valor médio. Na representação digital, as medidas variam de forma não contínua. Um relógio digital, por exemplo, apresenta informação de horas, minutos e segundos na forma de dígitos decimais, atualizando estas informações em saltos a cada segundo, a cada minuto e a cada hora, embora o tempo varie continuamente. A informação, então, é dita mudar em passos discretos. São possíveis, então, as associações: analógica com medida contínua e digital com medida discreta ou passo a passo.

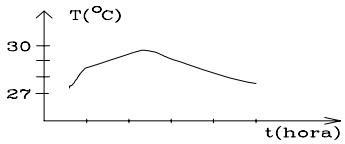
A Eletrônica, uma das ciências mais utilizadas na medição dos fenômenos físicos, pode ser classificada em duas grandes áreas: a Analógica e a Digital. Na Eletrônica Analógica os sinais elétricos possuem comportamento analógico. Na Eletrônica Digital os sinais elétricos assumem apenas dois níveis, ou valores de tensão de sinais elétricos, um dito “alto” e outro “baixo”. Dependendo da tecnologia empregada na confecção dos circuitos, os valores de tensão elétrica utilizados entre os níveis alto e baixo são valores bem definidos, assim como as faixas em que podem variar (nas quais um sinal elétrico é considerados "alto" ou "baixo"), entre outros parâmetros que caracterizam a família de circuitos integrados (CI's) empregada. Sinônima de padronização, a especificação de famílias de CI´s permitiu que vários fabricantes de componentes digitais surgissem e que seus produtos fossem conectáveis entre si.

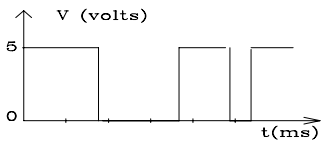
A Eletrônica Digital causou um grande impacto no desenvolvimento tecnológico. Pode-se dizer que, somente através dela, foi possível transformar o computador em um equipamento eletrodoméstico. Este crescimento resultou do vertiginoso estudo e aplicação dos materiais semicondutores e das técnicas de microprogramação. Aplicada inicialmente nos computadores, a Eletrônica Digital está presente na maioria dos equipamentos de comunicação e de controle atualmente existentes. Procedimentos de tratamento analógico de sinais e de informação estão continuamente sendo substituídos por técnicas digitais ... e tudo começou com os sistemas de numeração, centenas de anos A.C.

Em um Curso de Eletrônica Digital são apresentadas as ferramentas básicas para a análise e projeto de Sistemas Digitais.

Grandezas e Variáveis Lógicas

As variáveis de um sistema real podem ser classificadas como analógicas e digitais. As variáveis analógicas, geralmente variam de forma contínua e, teoricamente, possuem infinitos valores entre uma medida e outra (tais como as medidas da temperatura, do peso, da velocidade, entre outras). Uma variável digital assume valores discretos e possui número de estados finitos (tal como uma chave que pode estar ligada ou desligada, ou uma lâmpada (idem) ... uma porta (aberta ou fechada), etc.). As figuras abaixo apresentam os gráficos de dois sinais que representam uma variável analógica e uma variável / sinal digital:





Temperatura, variável analógica. Sinal elétrico pulsado, variável digital.

Variáveis Digitais

As variáveis digitais assumem sempre estados discretos. Geralmente, estes estados são em número de dois, um oposto ao outro. O estado de uma lâmpada, por exemplo, pode estar ligada ou desligada (não ligada). A energia elétrica pode estar presente ou ausente (não presente) nos contatos das tomadas. O estudo das variáveis lógicas é bastante antigo, o que torna difícil dizer quando surgiu o computador, finalmente. Em verdade, o computador é o resultado de estudos que tiveram início anos antes de Cristo, tais como o da álgebra e o da geometria, e ganhou impulso nos séculos XVIII com Leibniz e, posteriormente com os trabalhos de Boole, DeMorgan entre outros pensadores; tornando-se viável com a tecnologia dos semicondutores e a integração em muito larga escala dos componentes eletrônicos.

Boole estabeleceu, 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 verdadeira ou falsa.

Na Eletrônica Digital, um circuito só estar em um entre dois estados possíveis, onde há presença de tensão elétrica ou ausência do sinal; adequando-se perfeitamente aos princípios da álgebra de Boole (álgebra booleana), que classifica as informações em dois tipos: verdadeira e falsa. Atribui-se às informações verdadeiras o símbolo matemático 1 e às falsas o símbolo 0. Isto facilita o manuseio matemático das informações.

Assim, uma variável lógica assume apenas os valores 1 ou 0. As variáveis lógicas são normalmente representadas por letras e seu uso permite escrever expressões algébricas, que podem ser manipuladas matematicamente dentro das regras da álgebra booleana.

A álgebra booleana tem como base três funções lógicas: E, OU e NÃO (em inglês AND, OR e NOT), das quais derivam várias outras. A partir dessas três operações básicas é possível implementar desde o mais simples circuito eletrônico até o mais sofisticado computador digital. Na prática, as variáveis lógicas são utilizadas para descrever o funcionamento de um sistema lógico. Em outras palavras, pode-se traduzir em expressões da álgebra booleana um circuito digital.

Para ajudar na compreensão da evolução dos circuitos lógicos e dos computadores, enfim, vamos conhecer mais sobre a evolução desde segundo.

Níveis Lógicos

No corrente estudo dos circuitos digitais a presença de uma tensão é indicada como 1 ou H (de HIGH, Alto) enquanto que a ausência de uma tensão é indicada por 0 ou L (de LOW, Baixo). O 0 é sempre uma tensão próxima de zero, enquanto que o nível lógico 1 assume uma tensão próxima de 5 volts. Estes valores são características das famílias TTL e CMOS, descritas adiante.

Durante muito tempo, os circuitos construídos a partir da Álgebra booleana foram implementados utilizando-se dispositivos eletromecânicos: os relês; depois com válvulas. Nestes, os níveis de tensão correspondentes aos níveis lógicos variavam de fabricante para fabricante, o que dificultava o uso de circuitos eletrônicos de diferentes fabricantes em um mesmo projeto. À partir do surgimento do transistor, procurou-se padronizar os sinais elétricos correspondentes aos níveis lógicos. Esta padronização ocasionou o surgimento das famílias de componentes digitais.

As famílias lógicas diferem basicamente pelo componente principal utilizado por cada uma em seus circuitos. A família TTL (Transistor-Transistor Logic) usa transistores bipolares como seu principal componente, enquanto as famílias que utilizam a tecnologia MOS (Metal Oxide Semicondutor), usam os transistores unipolares MOSFET (transistor de efeito de campo construído segundo a técnica MOS) como seu elemento principal de circuito. Essas famílias lógicas serão discutidas com maior aprofundamento, posteriormente.

1. Sistemas de Numeração

Introdução

A Eletrônica Digital é a base para a análise e projeto de Sistemas Digitais. O exemplo mais comum destes sistemas é o computador digital, o qual é largamente utilizado no processamento da informação numérica e textual codificada por códigos numéricos binário. Estes códigos consistem no sistema de numeração binário, propriamente dito e por diversos códigos binários. Compreender os computadores digitais requer o domínio destes, que possuem dois símbolos apenas: 0 e 1. Devido ao uso costumeiro do sistema de numeração decimal é comum, no início do curso, alguma dificuldade no uso de outros sistemas de numeração.

Compreender como os sistemas digitais tratam os sistemas de numeração requer o estudo de como estes sistemas funcionam e que soluções são adotadas para a representação dos milhares de símbolos utilizados na vida real, tais como os números, as letras, os caracteres gráficos e semigráficos, acentos, pontuações, etc.. , como são armazenados e processados.

Histórico

Tribos aborígenes emitiram mensagens complexas à distância usando sinais com apenas dois dígitos, centenas de anos A C., porém estavam sujeitos às ambigüidades da língua, da mesma forma que o código Morse que também utiliza dois dígitos.

Gottfried Wilhelm Von Leibniz (1646 - 1716), idealista e matemático alemão, acreditava que pela lógica as leis do pensamento, que estavam sujeitas às diversas modificações da língua, poderiam passar de um estado verbal para uma condição matemática absoluta. As idéias de Leibniz eram avançadas, sendo logo ignoradas pela comunidade cientifica de seu tempo. Mesmo assim Leibniz prosseguiu com seus estudos através da lógica desenvolvendo uma coletânea de ambigüidades tais como: dia e noite; claro e escuro; sim e não; entre outras. Leibniz atribuía à vida propósitos discretos, fortalecendo assim o pensamento à lógica e, animado com suas idealizações, começou a refinar seu sistema binário rudimentar (sim ou não; claro ou escuro, etc.), utilizando numerosas combinações de uns e zeros, mesmo sem encontrar aplicação no seu tempo.

Leibniz faleceu em 1716 sem realizar o sonho de uma língua universal matemática e lógica. Deixou, porém, a principal idéia do sistema numérico binário, aperfeiçoada por George Boole 125 anos depois com a Lógica Booleana. Somente no ano de 1930 foi encontrada uma utilidade para o sistema binário: John Atanasoff, professor de física, propôs que a redução dos dez símbolos do sistema decimal para apenas dois, trariam mais eficiência e velocidade aos computadores eletrônicos; e no ano de 1939 criou o protótipo do computador utilizando o sistema binário.

Em aritmética binaria, uma quantidade existe ou não existe, e este tipo de decisão foi relativamente fácil de implementar com circuitos a transistor, onde tensão existe ou não na saída, *podendo mudar do estado de condução ao de corte em menos de um nanosegundo*. Na Eletrônica, é possível o desenvolvimento de circuitos que possam manipular diretamente a informação em decimal, porém os dispositivos físicos possuidores de dez diferentes estados ou diferentes níveis de tensão, seriam extremamente complexos e onerosos.

Os Sistemas de Numeração Básicos

A operação básica de um sistema de numeração é a contagem. Ao se contar o número de objetos de uma coleção, divide-se o conjunto em grupos com um determinado número de objetos, denominado base ou raiz do sistema de numeração. Neste estudo, são de interesse, *a priori*, os sistemas numéricos: Decimal, Binário, Octal, Hexadecimal e o BCD.

Existem várias formas para se representar os números associados às bases. As mais comuns são:

- Utilizar uma letra após o número para indicar a base;
- Colocar o número entre parênteses e a base como um índice do número.

Exemplo: 2763D ou $(2763)_{10}$ ou 2763_{10}

Sistema Decimal (Base 10)

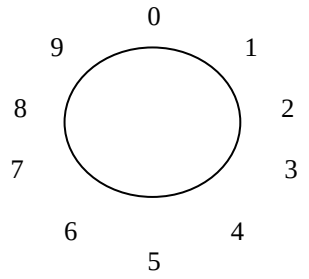
Muitos anos atrás, o homem sentiu a necessidade de contar, como exemplo, o número de animais em seu rebanho. Então, o homem começou a usar algo que estava "mais à mão": seus dedos. O que serviu, logo, como base para a contagem sempre que o que contava ultrapassava nove unidades. É possível imaginar que contava algo como: "... 5 mãos e 4 dedos em animais" ... ou o que corresponderia a 54_{10} animais.

O sistema decimal, então, utiliza 10 símbolos (algarismos) para representar qualquer quantidade:

0 1 2 3 4 5 6 7 8 9

Para representar quantidades maiores que a base são utilizados pesos para algarismos associadas à suas posições ou casas decimais. Quanto mais à esquerda for a posição do algarismo, maior seu peso, o qual é 10 vezes maior que a posição anterior, uma vez que a base é 10.

Uma idéia aplicada de sistema contagem é a do disco do sistema de numeração. Quando contamos, a cada unidade assumimos uma nova posição no sentido horário. Quando passamos pelo **zero** devemos incrementar uma unidade em outro circulo, o das dezenas, depois o das centenas, e assim sucessivamente. Baseados nesta idéia foram desenvolvidas as primeiras máquinas aritméticas.



Círculo de Contagem Decimal

Com a idéia de peso dos algarismos surgiram os nomes unidade, dezena, (dez unidades), centena (cem unidades), milhar (mil unidades), dezena de milhar, centena de milhar, milhão, etc. A utilização do Sistema de Numeração Decimal está tão presente no pensamento humano que, ao trabalhar com números, se age mecanicamente sem a percepção das convenções usadas: o número 2574, por exemplo, é composto por 2 milhares, 5 centenas, 7 dezenas e 4 unidades ou 2000 + 500 + 70 + 4 = 2574.

Em resumo:

Decimal
baseado em 10 dígitos (0,1,2,3,4,5,6,7,8.9)

Exemplo:
5264 = (5x1000) + (2x100) + (6x10) + 4x1
$5264 = (5x10^3) + (2x10^2) + (6x10^1) + (4x10^0)$

Valores fracionários:
$75{,}32 = (7x10^1) + (5x10^0) + (3x10^{-1}) + (2x10^{-2})$

De uma forma geral:

$$\mathbf{X} = \sum_i x_i 10^i$$

onde: x_i = (0,1,2,3,4,5,6,7,8,9) e i corresponde a posição do dígito.

Sistema Binário (Base 2)

O sistema binário utiliza dois símbolos (algarismos) para representar qualquer quantidade:

0 1

O sistema binário segue as mesmas regras do sistema decimal e utiliza os conceitos de peso e posição dos algarismos. Os termos unidades, dezenas, centenas, etc, são exclusivos do sistema decimal e não há não há semelhantes no sistema binário. Cada algarismo ou dígito de um número binário é chamado de bit, ou a abreviação de binary digit (dígito binário).

Exemplo: 101_2 (lê-se um-zero-um)

O caracter 1, mais à esquerda, corresponde ao caracter mais significativo (*Most-Significative-Bit*,) e é denominado *MSB.* O caracter 1, mais à direita, corresponde ao caracter menos significativo (*Least-Significative-Bit)* e é denominado "LSB".

Os símbolos manuseados pelo homem podem ser classificados como numéricos e não numéricos ou alfanuméricos. Estes símbolos, também conhecidos como caracteres, quando são introduzidos em um sistemas digital, são convertidos para sistemas binários os que obedecem às regras básicas da matemática. Quando alfanuméricos, são traduzidos mediante códigos binários, de tal modo que o processo inverso restaure a informação armazenada fidedigmamente.

Em resumo:

Sistema Binário
baseado em dois dígitos: 0 e 1 (base 2)

Exemplos:
$11 = (1x2^1) + (1x2^0)$
$110 = (1x2^2) + (1x2^1) + (0x2^0)$
$100.101 = 2^2 + 2^{-1} + 2^{-3}$

Sistema Hexadecimal (Base 16)

O sistema hexadecimal possui 16 símbolos, dos quais os dez primeiros são velhos conhecidos, e os demais são adotados as letras A, B, C, D, E e F:

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

Exemplo: $5A3_{16}$ (lê-se cinco-a-três)

Sistema Octal (Base 8)

O sistema octal apresentausa 8 símbolos:

0 1 2 3 4 5 6 7

Exemplo: 563_8 (lê-se cinco-seis-três)

Em resumo:

- O número de dígitos usado no sistema é igual à base.
- O maior dígito é sempre menor que a base.
- O algarismo mais significativo está à esquerda, e o menos significativo à direita.
- Em geral se toma a base decimal como referência.

Sistema BCD (*Binary Code Decimal*)

O BCD trata-se da codificação binária dos números decimais dígito a dígito. Neste sistema, associa-se 4 bits para cada dígito como mostram os exemplos abaixo. Esta codificação é utilizada em grandes operações aritméticas com números reais representados em ponto flutuante ou notação científica. Nesta representação, um número possue uma parte fracionária ou mantissa (que é sempre menor do 1 e maior do que 0,1) e uma potência (de base decimal).

Exemplos de números BCD:

$345_{10} = 0011_2\ 0100_2\ 0101_2 = 001101000101_{bcd}$
$10000010_{bcd} = 1000_2\ 0010_2 = 82_{10}$

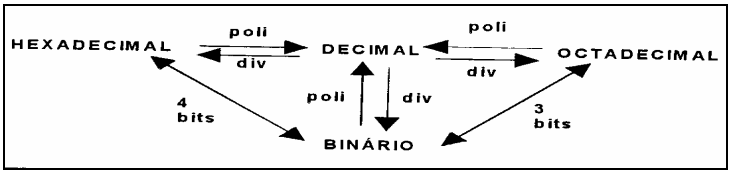
Conversão entre Sistemas Numéricos

Para converter uma informação representada em dado sistema de numeração para outro faz-se necessário aplicar uma ou duas regras de equivalência. A tabela a seguir apresenta a equivalência entre os sistemas já conhecidos:

Decimal	Binário	Octal	Hexadecimal	BCD
0	0	0	0	00000000
1	1	1	1	00000001
2	10	2	2	00000010
3	11	3	3	00000011
4	100	4	4	00000100
5	101	5	5	00000101
6	110	6	6	00000110
7	111	7	7	00000111
8	1000	10	8	00001000
9	1001	11	9	00001001
10	1010	12	A	00010000
11	1011	13	B	00010001
12	1100	14	C	00010010
13	1101	15	D	00010011
14	1110	16	E	00010100
15	1111	17	F	00010101
16	10000	20	10	00010110
17	10001	21	11	00010111
...	...	...	...	...

Sistemas de Numeração mais Comuns

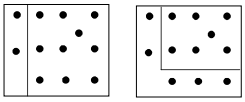
Através da tabela, percebe-se que é impraticável montá-la para todos os números ou realizar conversões de uma base para outra sem o uso de uma regra aritmética. Existem, portanto, três procedimentos básicos úteis na conversão de um sistema em um outro: a divisão, o polinômio e o agrupamento de bits. A figura abaixo mostra quando devem ser utilizados:



Métodos de Conversão entre Bases Numéricas

A Divisão

A divisão é utilizada quando desejado encontrar o equivalente de um número decimal em um outro sistema. Sejam os quadros abaixo:

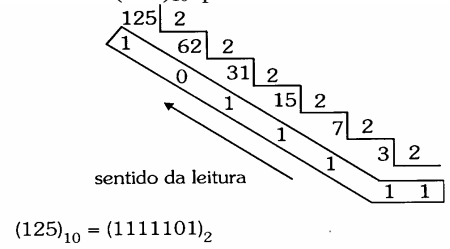


12_{10} bolas 15_7 bolas

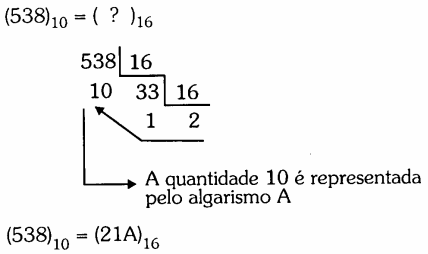
No primeiro quadro há 12_{10} bolas porque há um conjunto de 10 (que é a base) e sobram 2 unidades. No segundo, há 15_7 pelo mesmo motivo. O que foi feito foi expressar o mesmo número de bolas em dois sistemas de numeração diferentes. Então, na conversão de um número decimal em um sistema de base b, divide-se este número por b e seus resultados consecutivas vezes.

Exemplos:

Converter $(125)_{10}$ para a base 2.



b) Converter $(538)_{10}$ em hexadecimal



Método da Divisão

A Notação Polinomial ou Posicional (Lei de Formação)

Desde a infância aprende-se a ler os números segundo o peso do dígito (ou casa decimal) que ocupam os numerais. O número 234, por exemplo, é lido como "duzentos, trinta e quatro", que corresponde a duas centenas, três dezenas e quatro unidades, exatamente. De outro modo, pode-se dizer que 234 é igual $2x10^2 + 3x10^1 + 4x10^0$, que os numerais 2, 3 e 4 são elementos do sistema decimal e que 10 é a sua base. Sempre lê-se ou narra-se um número da esquerda para a direita ou do seu maior peso (ou dígito mais significativo) ao de menor peso (ou dígito menos significativo).

O número 2574 pode ser decomposto da forma:

$$\begin{aligned} 2574 &= 2000 + 500 + 70 + 4 \\ &= 2 \times 1000 + 5 \times 100 + 7 \times 10 + 4 \times 1 \\ &= 2 \times \mathbf{10}^3 + 5 \times \mathbf{10}^2 + 7 \times \mathbf{10}^1 + 4 \times \mathbf{10}^0 \end{aligned}$$

BASE

Decomposição Polinomial de um Número

Esta forma de decompor um número é chamada de Lei de Formação e é válida para qualquer base numérica. Genericamente, a Lei de Foação é escrita na forma:

$$N = a_n b^n + a_{n-1} b^{n-1} + a_{n-2} b^{n-2} + ... + a_0 b^0,$$

Onde:
a_n = algarismo
b = base do número
n = quantidade de algarismo - 1

Para se converter um número de um sistema numérico qualquer para o decimal, basta aplicar a lei de formação substituindo **b** pela base do número a ser convertido e $\mathbf{a_n}$ por seus respectivos algarismos:

Exemplos:

a) conversão do número $(1101)_2$ para a base 10
$(1101)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (13)_{10}$

b) conversão do número $(3AF7)_{16}$ para a base 10
$(3AF6)_{16} = 3 \times 16^3 + A \times 16^2 + F \times 16^1 + 6 \times 16^0 = (15095)_{10}$

c) Conversão de decimal para binário:

Parte inteira:

$N = 2 \text{ x } N_1 + R_1$ $\quad R_1 = 0$ ou 1
$N_1 = 2 \text{ x } N_2 + R_2$ $\quad R_2 = 0$ ou 1
$N_2 = 2 \text{ x } N_3 + R_3$ $\quad R_3 = 0$ ou 1

Assim:
$N = 2 \text{ x } (2 \text{ x } (2 \text{ x } N_3 + R_3) + R_2) + R_1$ $\quad (R1 = 2^0 \text{ x } R_1)$
$N = (2^3 \text{ x } N_3) + (2^2 \text{ x } R_3) + (2^1 \text{ x } R_2) + R_1$
e assim sucessivamente até $N_k = 1$.

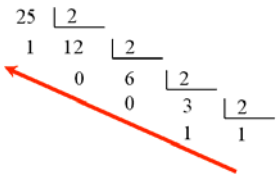
Parte fracionária:
$0.101 = 1x2^{-1} + 0x2^{-2} + 1x2^{-3} = 0.625$
$F = a_{-1} x\, 2^{-1} + a_{-2} \text{ x } 2^{-2} + a_{-3} \text{ x } 2^{-3} + \ldots$

onde $0 < F < 1$ e a_{-i} é 0 ou 1.
$2xF = a_{-1} + a_{-2} \text{ x } 2^{-1} + a_{-3} \text{ x } 2^{-2} + \ldots$

A parte inteira dos dois lados da equação devem ser iguais e como $0 < 2F < 2$, a parte inteira do lado esquerdo da equação deve ser 0 ou 1, ou seja, é a_{-1}.

Repetindo o processo podemos encontrar a notação binária de F.

d)



$25 = 1x\,2^4 + 1 \text{ x } 2^3 + 0 \text{ x } 2^2 + 0 \text{ x } 2^1 + 1 \text{ x } 2^0$

11001

e)

0.56 x 2 = 1.12
0.12 x 2 = 0.24
0.24 x 2 = 0.48
0.48 x 2 = 0.96
0.96 x 2 = 1.92

$0.56 = 1 \text{ x } 2^{-1} + 0 \text{ x } 2^{-2} + 0 \text{ x } 2^{-3} + 0 \text{ x } 2^{-4} + 1 \text{ x } 2^{-5} \ldots$

.10001 (aproximadamente)

Agrupamento de Bits

O método do agrupamento é utilizado nas conversões entre sistemas octal e hexa para o binário (e vice versa).

Exemplos:

a) $(1011110010100111)_2 = (\ ?\ )_{16}$

1011 1100 1010 0111
↓ ↓ ↓ ↓
B C A 7

$(1011110010100111)_2 = (BCA7)_{16}$

b) $(A79E)_{16} = (\ ?\ )_2$

A 7 9 E
↓ ↓ ↓ ↓
1010 0111 1001 1110

$(A79E)_{16} = (1010011110011110)_2$

Agrupamento de Bits

Conversão com o binário

Costuma-se dizer que os sistemas octal e hexa são subconjuntos do binário. Em verdade, são *múltiplos* e são utilizados para facilitar a leitura e a escrita dos números binários, pois é mais fácil (e rápido) falar $FADA_{16}$ (efe-a-dê-a) do que 1111101011011010_2 (um-um...zero).

Como $8 = 2^3$ e $16 = 2^4$, fazer conversões rápidas destes sistemas para o binário, significa associar 3 bits ou 4 bits (quando octal ou hexadecimal, respectivamente) e vice-versa. Nos exemplos a seguir, o polinômio é utilizado para mostrar/comprovar estas conversões para o binário:

Exemplos:

a) $201_8 = 2x8^2 + 0x8^1 + 1x8^0 = 2x\ (2x2x2)^2 + 1$
$= 2^7 + 1 = 1x2^7 + 0x2^5 + 0x2^4 + 0x2^3 + 0x2^2 + 0x2^1 + 1x2^0$
$= 10000001_2$

b) $FADA_{16} = 15x16^3 + 10x16^2 + 13x16^1 + 10x16^0$
$= (8+4+2+1)x(2x2x2x2)^3 + (8+2)x(2x2x2x2)^2 + (8+4+1)x2x2x2x2 + (8+2)$
$= 2^{15} + 2^{14} + 2^{13} + 2^{12} + 2^{11} + 2^9 + 2^7 + 2^6 + 2^4 + 2^3 + 2^1$
$= 1111101011011010_2$

Como visto, os cálculos comprovam as relações entre os sistemas octal e hexa com o binário. Agilizando:

201_8 = 010 000 001 = 10000001_2
$FADA_{16}$ = 1111 1010 1101 1010 = 1111101011011010_2

Em resumo, converte-se da direita para a esquerda para o binário porções de 3 bits (se octal) ou 4 bits (se hexa).

A conversão octal - hexadecimal não pode ser realizada diretamente, uma vez que não há relação de potência inteira entre as bases octal e haxa. Este tipo de conversão é semelhante à conversão entre duas bases quaisquer, onde deve ser usada uma base intermediária. Entretanto, ao invés de usar a base decimal, é mais simples usar a base binária como base de referência para uma conversão em duas etapas:

- Primeiramente, o número é convertido da base octal (ou hexadecimal) para a base binária.
- Em seguida, o resultado intermediário é convertido da base binária para a base hexadecimal (ou octal).

Exemplo: Converter o número $(11647)_8$ para a base 16.

Convertendo cada dígito octal separadamente, obtém-se
$11647_8 \Leftrightarrow 001\ 001\ \ 110\ \ 100\ 111_2$

Dividindo o número em grupos de quatro bits a partir da direita, e convertendo cada grupo separadamente, obtém-se:
$001\ 001\ \ 110\ \ 100\ 111_2 \Leftrightarrow 0001\ \ 0011\ 1010\ \ 0111_2 \Leftrightarrow 13A7_{16}$

Concluindo, uma das principais aplicações dos sistemas octal e hexadecimal é simplificar o uso de números binários. De fato, números binários grandes são difíceis de serem lidos o que leva a erros de leitura pela facilidade da troca ou esquecimento de um bit, em uma grande seqüência de bits. Assim, o conteúdo de registradores ou porções de memória são lidos em hexadecimal ou octal. Mesmo quando se escreve um número binário, é prática dar um espaço entre grupos de quatro ou três bits, para facilitar a leitura.

Conversão de Números Fracionários

Até aqui foram vistos os números inteiros em diversas bases. É possível utilizar toda esta teoria para os números fracionários. Pode-se, portanto, ampliar a aplicação da **Lei de Formação** para números fracionários utilizando-se, para os algarismos à direita da vírgula, expoentes negativos em ordem crescente. Assim, é a Lei de Formação ampliada :

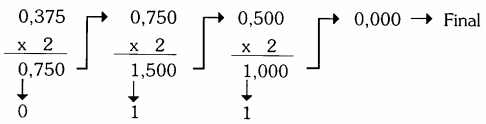
$$\text{Número} = \underbrace{a_n.b^n + a_{n-1}.b^{n-1} + a_{n-2}.b^{n-2} + \ldots + a_0.b^0}_{\text{parte inteira}} + \underbrace{a_{-1}.b^{-1} + a_{-2}.b^{-2} + \ldots + a_{-m}.b^{-m}}_{\text{parte fracionária}}$$

Exemplos:

a) Base 2 para a base 10:
$(101{,}110)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} = (5{,}75)_{10}$

b) Base 10 para a base 2:
$(8{,}375)_{10} = (\ ?\ )_2$

- parte inteira: $(8)_{10} = (1000)_2$
- parte fracionária:



$(8{,}375)_{10} = (1000{,}011)_2$

Conversão para BCD

Neste caso, associa-se 4 bits binários para cada dígito, como apresentou a tabela no início deste capítulo. Sejam os exemplos abaixo:

$345_{10} = 0011_2\ 0100_2\ 0101_2 = 001101000101_{bcd}$
$10000010_{bcd} = 1000_2\ 0010_2 = 82_{10}$

2. Operações Aritméticas

Introdução

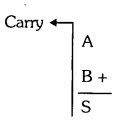
As operações aritméticas na base decimal são facilmente realizáveis por dependerem da capacidade da memorização das tabuadas de cada operação, ou os conjuntos de relações que definem as operações aritméticas de soma (+), subtração (-), multiplicação (×) e divisão (÷).

As operações aritméticas são operações ou relações binárias definidas para dois argumentos. O computador, por exemplo, realiza internamente várias operações no sistema binário. Para a compreensão de como isso é feito, é necessário entender o mecanismo das operações básicas, adição e subtração no sistema binário.

As operações aritméticas no sistema binário constituem uma parte do estudo da numeração binária a qual facilitará a compreensão dos circuitos lógicos aritméticos somadores, subtratores, entre outros, apresentados adiante.

Adição - Sistema Numérico Binário

Na adição entre dois números de um algarismo cada, pode-se obter resultados com um ou dois dígitos. Em algumas situações esta adição leva a um *estouro* (o maior algarismo do sistema é ultrapassado). A este estouro dá-se o nome de *carry* (transporte) ou *vai-um.* No sistema binário o estouro ocorre apenas quando se adiciona uma unidade ao algarismo 1. Este mesmo mecanismo, portanto, pode ser adotado para números com vários dígitos. A figura a seguir apresenta operações de adição entre dois algarismos A e B na base 2.



Operandos		Resultado	Estouro
A	B	S	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Soma Binária

Quando não há estouro, entende-se que carry = 0, caso contrário carry =1.

Exemplos:

```
a)  0
    0 1
    1 0 +
    -----
    1 1

b)  1 1 1
      1 0 1
      0 1 1 +
    -------
    1 0 0 0

c)  1 0 1 1 1 1 0 1
      1 0 0 1 1 1 0 1
      1 0 1 0 0 1 0 1 +
    -----------------
    1 0 1 0 0 0 0 1 0
```

Se feita a conversão destes números para o sistema decimal, obtém-se, respectivamente:

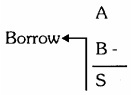
```
a)   1
     2 +
     ---
     3

b)   5
     3 +
     ---
     8

c)   1 5 7
     1 6 5 +
     -------
     3 2 2
```

a) $(3)_{10}=(11)_2$ **b)** $(8)_{10}=(1000)_2$ **c)** $(322)_{10}=(101000010)_2$

Subtração - Sistema Numérico Binário

A operação subtração, tem como mecanismo o inverso do utilizado na adição. Contudo, quando o minuendo é menor que o subtraendo há também um estouro e, neste caso, deve-se subtrair uma unidade do minuendo ou somar uma unidade ao subtraendo da casa seguinte. A este estouro dá-se o nome de borrow (empréstimo) ou vem-um.

A operação de subtração entre dois algarismos no sistema binário é apresentada pela figura abaixo:



Operandos		Resultado	Estouro
A	B	S	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Subtração Binária

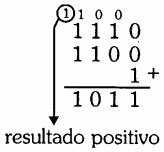
Nesta tabela, pode-se observar que a coluna subtração tem os mesmos resultados da coluna soma. A diferença entre os resultados das operações adição e subtração no sistema binário está nas colunas *carry* e *borrow*.

Subtração em Complemento de 2

O método do complemento de 2 (o que falta para 2), que corresponde a **somar 1 ao complemento de** 1 (o que falta para 1), é o mais indicado para a operação subtração no sistema binário, devido à sua simplicidade. Exemplos:

a)

$$1110 - 11 = 1110 + \overline{0011} + 1 = 1110 + 1100 + 1 = 1011$$

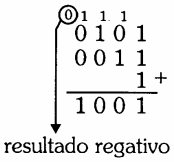


Fazendo a verificação no sistema decimal: 14 – 3 = + 11

Nota-se que, para a complementação do número $(11)_2$, foi necessário acrescentar dois zeros à sua esquerda $(0011)_2$, para que os dois números tenham a mesma quantidade de algarismos para que a operação fique correta.

b)

$$0101-1100 = 0101+\overline{1100}+1 = 0101+0011+1 = 1001$$



Logo, o resultado é $-(\overline{1001}+1) = -0111$

Fazendo a verificação no sistema decimal: 5 – 12 = -7

Com os números negativos não é válida a expressão polinomial utilizada para a obtenção do correspondente em decimal. Na realidade, para um computador ou um sistema digital qualquer, não interessa a complementação final para representar o resultado negativo, pois, pelo *carry* final ele sabe se o resultado é positivo ou negativo. Esta convenção, mostrando o sinal + ou - , só é necessária quando o computador for apresentar o resultado para o usuário, por exemplo, numa tela ou numa impressora.

Bit de Sinal:

Várias tentativas de codificar números negativos (que podem ser ditos complementos dos positivos e vice-versa) foram feitas. Uma delas foi a de colocar um bit à esquerda do número para indicar que é positivo ou negativo, o que foi complicado para certas aplicações, devido à baixa velocidade de processamento e a dificuldade de se localizar situações de estouro ou overflow. No caso de operações realizadas por software, o programa teria que testar continuamente a ocorrência ou não de estouro ou mudança do bit de sinal**.**

Seja A = 00001010 = $+10_{10}$, então – A = 10001010 = -10_{10} .

Complemento de 1

Outra solução é o complemento de 1: se A é um número, ~A é o seu complemento.

Seja A = 00001010 = $+10_{10}$, então – A = 11110101 = -10_{10}
Esta solução também exige algumas avaliações quanto às situações de overflow e quanto ao valor zero:

Seja A = 00001010, então A – A = A + (-A) = 00001010 + 11110101 = 11111111, que é diferente de 00000000, a menos que esta igualdade seja admitida (e corrigida pelo hardware ou software).

Complemento de 2:

O complemento de 2, pela sua velocidade e eficiência é o mais utilizado. Ao hardware ou ao software significa verificar situações em que as operações de soma e subtração apresentam "falsos overflows". Sejam os exemplos a seguir:

Seja A = 00001010 = $+10_{10}$,
então – A = -10_{10} = 11110101 (complemento de 1) + 1 = 11110110.
Vamos verificar?

```
A – A = A + (-A)  =      00001010
                        11110110
                         --------------  +
                        10000000          ( igual a   00000000 ?)
```

Nos computadores os números (como todos os dados) são limitados segundo regras associadas com os tipos de dados, o que significa dizer que existem limites e tamanhos em bits para cada tipo de variável, ou elemento associado com o dado. No exemplo anterior, foi utilizado um sistema de oito bits onde a passagem de um "1" para um nono bit não se trata de *overflow*, porque um número positivo somado a um negativo (ou vice-versa) será sempre um número entre eles. A seguir é apresentado o conjunto dos números inteiros codificados em binário de oito bits em um sistema que utiliza complemento de 2:

{ 10000001, ..., 11111110, 11111111, 00000000, 00000001, 00000010, ..., 01111111 }
(-127, ..., -2. -1, 0, +1, +2,..., +127)

Neste caso, o bit mais significativo contém a informação de sinal (0 = positivo e 1 = negativo). Exemplo:

1) Dados os números nas bases decimal e binária em 8 bits, calcule o que se pede:
a) Se X_2 = 11110000 - X_2 = ? X_{10} = ?

b) Se Y_{10} = 4 - Y_2 = ?

Solução:

a) $-X_2 = \sim X_2 + 1 = 00001111_2 + 1 = 00010000_2$
$00010000_2 = 16_{10}$, então $X_{10} = -16$

b) $Y_{10} = 4_{10} = 00000100_2$
$-Y_{10} = -4_{10} = -Y_2 = \sim Y_2 = \sim 00000100_2 + 1 = 11111011 + 1 = 11111100_2$

2) Realizar as operações aritméticas a seguir em binário com 8 bits. Verificar em decimal se os resultados encontrados são coerentes. No caso (ou não) de *overflow*, justificar sua resposta:
a) $32_{10} - 54_{10}$ = ? b) $67_{10} - 42_{10}$ = ?

Solução

a) $32_{10} - 54_{10} = 32_{10} + (-54_{10})$ = 00100000 + ~00110110 + 1 = 00100000

```
11001001 +
        1
 --------
11101010
```

$32_{10} – 54_{10} = 11101010_2 = - 22_{10}$ (?)

Prova:
Se $11101010_2 + 1 = 00010101 + 1 = 00010110_2 = 1x2^4 + 1x2^2 + 1x2^1 = 22_{10}$
Então $11101010_2 = - 22_{10}$ (!)

b) $67_{10} – 42_{10} = 01000011 + \sim00101010 + 1 = 01000011$

```
   11010101 +
          1
 ----------
 100011001
```

$67_{10} – 42_{10}$ = 00011001 (porque não pode haver overflow na soma de números de sinais opostos)
$= 2^4 + 2^3 + 2^0 = 25_{10}$

Deslocamento (Multiplicação e Divisão)

Multiplicação pela Base

Deslocando-se os algarismos de um número para a esquerda ou a sua vírgula para a direita, o resultado corresponde à multiplicação do número pela sua base. Exemplos:

a) $(101{,}11)_2 \times 2 = (1011{,}1)_2$ (lembre-se de que $2 = 2_{10} = 10_2$)

Verificar este resultado em decimal com $(101{,}11)_2 = (5{,}75)_{10}$ e $(1011{,}1)_2 = (11{,}5)_{10}$

b) $(11{,}1011)_2 \times 4 = (11{,}1011)_2 \times 2 \times 2 = (1110{,}11)_2$

Verificar este resultado em decimal com $(11{,}1011)_2 = (3{,}6875)_{10}$ e $(1110{,}11)_2 = (14{,}75)_{10}$

Multiplicação Binária

Na multiplicação sabe-se que.
$0 \times 0 = 0$ $0 \times 1 = 1$ $1 \times 0 = 1$ $1 \times 1 = 1$

Sejam os cálculos abaixo em binário:

```
a)      101        b)  1001        c)     1101        d)   101 e)    110
      x  100             x 11            x  101           x 101      x 111
     --------           ------          --------          ------     ------
   1010000             1001             1101              101        110
                      1001             0000             101        110
                     ---------        1101             -------    110
                     11011           -----------       11001     ---------
                                      1000001                    101010
```

Em a) uma das parcelas da multiplicação é uma potência exata da base. Então, repetida a outra parcela e acrescentado o número de 0's presentes na parcela que é potência exata; em b) e em c) procede-se igualmente como com decimais; em d) foi excluído um resultado parcial nulo (por ser desnecessário); e em e) a ordem das parcelas não alterou o produto.

Divisão pela Base

Deslocando-se os algarismos de um número para a direita ou a sua vírgula para a esquerda, o resultado corresponde à divisão do número pela sua base.

Exemplos:

- $(1100{,}1)_2 \div 2 = (110{,}01)_2$

Conferindo este resultado em decimal com $(1100{,}1)_2 = (12{,}5)_{10}$ e $(110{,}01)_2 = (6{,}25)_{10}$

- $(11{,}01)_2 \div 8 = (0{,}01101)_2$

Verificar este resultado em decimal com $(11{,}01)_2 = (3{,}25)_{10}$, $(0{,}01101)_2 = (0{,}40625)_{10}$

Divisão Binária

Ocorre igualmente como nos decimais:

a) 1000 / 10 = 100 b) 1110 / 10 = 111
c) 101010 / 1000 = 101,01 d) 1111001 / 1011 =?

a) e b) apresentam divisões por potências de 10_2 (2_{10}). Em c) uma vírgula foi utilizada para separar a parte inteira da parte fracionária do resultado; em d) que :

```
1111001   | 1011
010000    | 1011
   01011
   (0000)
```

Conferindo ... $1011_2 = 11_{10}$
$1111001_2 = 121_{10}$

Notação dos Números Positivos e Negativos

Nesta Seção busca-se resumir a notação utilizada nos sistemas digitais quanto aos números positivos e negativos:

- Números com sinal-magnitude: usa-se o bit mais significativo para expressar um número positivo (0) ou número negativo (1); exemplos:

  $+5_{10} = 0000\ 0101_2$
  $-16_{10} = 1001\ 0000_2$
  $-73_{10} = 1100\ 1001_2$

  Variação dos números com sinal magnitude:

  - Números de 8 bits sem sinal → 0 a 255
  - Números de 8 bits com sinal → -127 a + 127
  - Números de 16 bits sem sinal → 0 a 65.535
  - Números de 16 bits com sinal → -32.767 a + 32.767

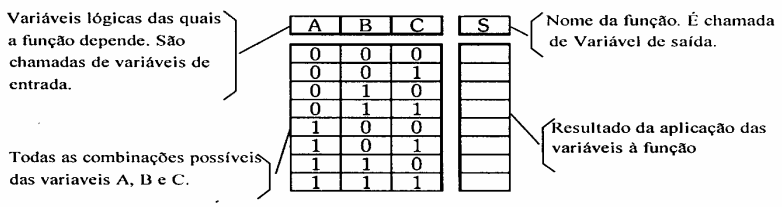
- Representação em complemento de dois: bastante utilizada nos circuitos que realizam somas e subtrações; A tabela a seguir mostra que nesta representação, o número negativo mais alto tem uma magnitude uma unidade maior que o número positivo mais alto.

Magnitude	Positivo	Negativo
1	0 0 0 1	1 1 1 1
2	0 0 1 0	1 1 1 0
3	0 0 1 1	1 1 0 1
4	0 1 0 0	1 1 0 0
5	0 1 0 1	1 0 1 1
6	0 1 1 0	1 0 1 0
7	0 1 1 1	1 0 0 1
8	-	1 0 0 0

3. As Funções Lógicas Básicas, Tabela-Verdade e Simbologia

As funções lógicas são operadores que possuem como entrada pelo menos uma variável lógica e uma saída. O resultado lógico de uma operação também possui um comportamento discreto. Uma função lógica é uma operação da álgebra booleana aplicada a uma ou mais variáveis lógicas.

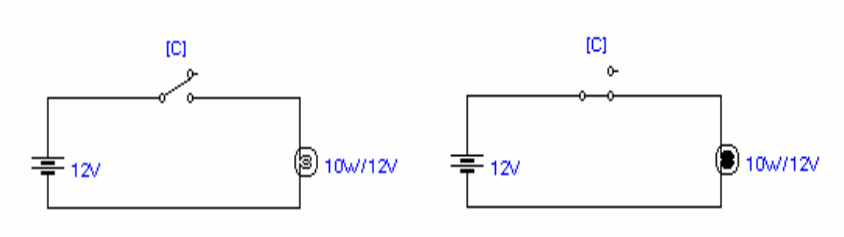
Em alguns casos, as funções lógicas são extremamente complexas e de difícil análise. A tabela-verdade é uma representação, em forma de tabela, das funções lógicas que facilita sua representação e análise.



A	B	C	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	

Tabela-Verdade para três variáveis: A, B e C.

A função lógica mais elementar é a função identidade ou igualdade, a qual é exemplificada pelo circuito abaixo:



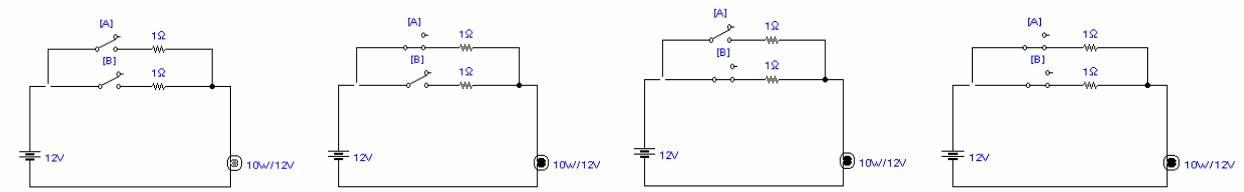
Função Identidade: Modelo Elétrico

Seja L a variável que representa o estado da lâmpada e C a chave de liga/desliga do circuito. A relação L = C significa que quando a chave está ligada (ON) a lâmpada está iluminando um ambiente (ON); quando a chave está desligada (OFF), a lâmpada não o está iluminando (OFF).

Serão estudadas, também, as funções lógicas básicas OU, E e INVERSORA (OR, AND e NOT ou INVERTER) e outras derivadas destas (NOR, NAND, XOR e XNOR). As portas lógicas constituem os dispositivos básicos dos circuitos digitais e têm como função a implementar as funções lógicas.

A Função OU (OR)

Na figura abaixo, um circuito elétrico apresenta quatro estados que podem assumir as duas chaves utilizadas para controlar uma lâmpada:



Função OR: Modelo Elétrico

A Tabela-Verdade

As situações mostradas nas figuras podem ser representadas por uma tabela, a qual expressa o comportamento da função, lembrando que L trata-se da lâmpada. Se associados os termos "não" a "0" (zero) e os termos "sim" a "1" (um), encontra-se a tabela verdade da função OR.

Para memorizar a função OR basta lembrar de um avião (não planador), que possui dois motores (entradas), que permitem o vôo (saída) se pelo menos um dos motores estiver funcionando (estado "1") ... caso contrário, este não permanecerá no ar. Em outras palavras a saída da função OR é FALSA ("0") se e somente se todas as entradas forem FALSAS.

B fechada	A fechada	L Fechada
não	não	não
não	sim	sim
sim	não	sim
sim	sim	sim

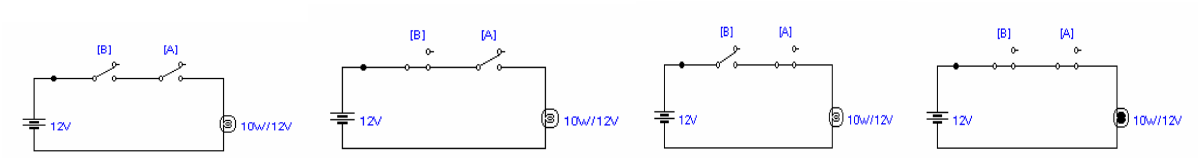
B	A	L
0	0	0
0	1	1
1	0	1
1	1	1

Função OR: Tabela-Verdade

A porta lógica OR (OU) é aquela cujo circuito executa a função lógica OR (OU).

A Função E (AND)

Da mesma forma, há circuitos com chaves que representam a função AND. Sejam os quatro estados que podem assumir duas chaves ligadas em série usadas no controle de uma lâmpada:



Função AND: Modelo Elétrico

A Tabela-Verdade

Da mesma forma, é possível representar esta função por tabelas. No caso da função AND a saída é SIM (verdadeira, "1"), se e somente se todas as entradas são SIM.

A função AND pode ser encontrada no seguinte anunciado: "... se todas as portas e janelas de um avião estiverem fechadas (em "1"), então este poderá ser pressurizado e levantar vôo (saída).

B fechada	A fechada	L Fechada
não	não	não
não	sim	não
sim	não	não
sim	sim	sim

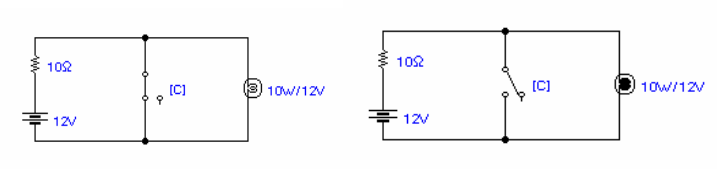
B	A	L
0	0	0
0	1	0
1	0	0
1	1	1

Função AND: Tabela-Verdade

A porta lógica AND (E) é aquela cujo circuito executa a função lógica AND (E).

A Função NÃO (NOT) ou INVERSORA (INVERTER)

A função INVERSORA complementa o resultado da Função Identidade. No circuito a seguir, quando a chave está aberta (NÃO) a bateria fornece a corrente que ilumina a lâmpada (SIM), ou saída. Quando ligada a chave (SIM) a corrente que antes alimentava a lâmpada é desviada pelo caminho elétrico de menor resistividade, forçando ao à NÃO iluminação da lâmpada.



Função NOT: Modelo Elétrico

C Fechada	L Ligada
Não	sim
Sim	não

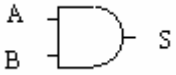
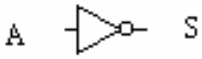
C	L
0	1
1	0

Função NOT: Tabela-Verdade

A porta lógica NOT (NÃO) é aquela cujo circuito executa a função lógica NOT (NÃO).

Simbologia das Funções Lógicas

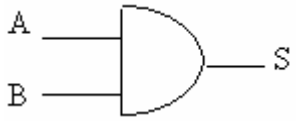
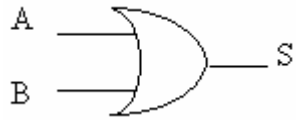
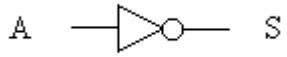
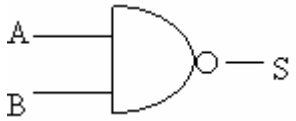
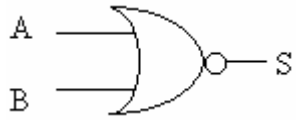
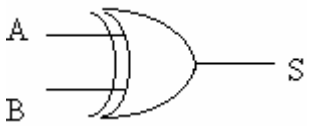
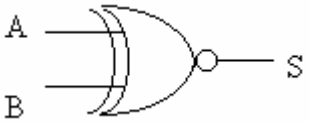
Para a representação de uma função lógica pode ser utilizada uma expressão lógica que utiliza símbolos não gráficos ou diagramas lógicos que utilizam símbolos gráficos. A tabela abaixo apresenta as simbologias adotadas para as funções já conhecidas:

Função	**OR**	**AND**	**NOT**
Expressões	S = A + B S = A $_v$ B	S = A.B S = AB S = A $^\wedge$ B	S = A* S = $\overline{A}$ S = ~A
Símbolo Lógico			

Simbologia das Portas Lógicas Principais

Funções Lógicas NE (Não E, NAND), NOU (Não OU, NOR), OU Exclusivo (XOR, Exclusive OR) e NOU Exclusivo (NXOR, Exclusive NOR)

A tabela a seguir apresenta tabelas e símbolos as portas lógicas conhecidas e outras implementadas à partir destas primeiras que, pelo elevado número de aplicações, possuem denominações e podem ser encontradas em circuitos integrados digitais.

Porta Símbolo Usual	Tabela Verdade	Função Lógica	Expressão
Porta E, AND  	A B S 0 0 0 0 1 0 1 0 0 1 1 1	Assume 1 Quando Todas as entradas forem 1	S = A.B
Porta OU, OR  	A B S 0 0 0 1 0 1 0 1 1 1 1 1	Assume 0 Quando as entradas forem 0	S = A + B
Porta Não, Not, Inversora, Inverter  	A S 0 1 1 0	Complementa o estado da variável aplicada à entrada	$S = \overline{A}$
Porta NE, NAND  	A B S 0 0 1 0 1 1 1 0 1 1 1 0	Complemento da Função E	$S = \overline{A.B}$
Porta NOU, NOR  	A B S 0 0 1 0 1 0 1 0 0 1 1 0	Complementoda Função OU	$S = \overline{A + B}$
Porta OU Exclusivo, Exclusive OR, XOR  	A B S 0 0 0 0 1 1 1 0 1 1 1 0	Assume 1 Quando as entradas forem diferentes entre si	$S=\overline{A}.B+A.\overline{B}$ $S=A \oplus B$
Porta NOU Exclusivo, Exclusive NOR, NXOR  	A B S 0 0 1 0 1 0 1 0 0 1 1 1	Assume 1 Quando Houver <u>Igualdade</u> entre as entradas	$S=A.B+\overline{A}.\overline{B}$ $S=A \otimes B$

Portas Lógicas Principais e destas Derivadas mais Comuns

Obs.: Há duas maneiras de indicar a negação ou inversão. Uma é usar o símbolo do inversor, o que deve ser feito sempre que se usar o próprio dispositivo físico. A outra é usar um pequeno círculo na entrada ou na saída de um bloco funcional para indicar a negação.

Diagramas de Tempo

Costuma-se estudar o comportamento das portas lógicas, também, à partir da variação temporal de suas saídas, em função dos estados de suas entradas. As figuras abaixo apresentam exemplos de diagramas de tempo para as funções OR e AND. A avaliação deve ser feita sempre na vertical.

Diagrama de Tempo (OR)

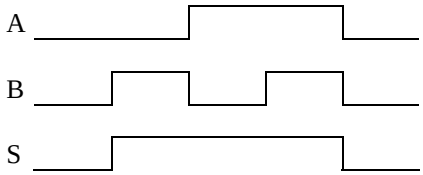


Tabela Verdade

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

Função OR: Diagrama de Tempo

Diagrama de Tempo (AND)

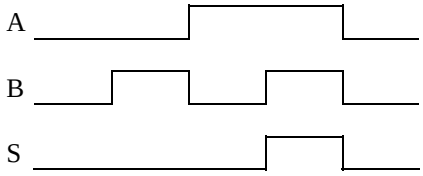


Tabela Verdade

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

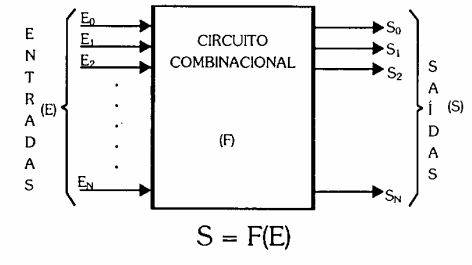
Função AND: Diagrama de Tempo

4. Formação e Simplificação de Expressões Lógicas

Neste Capítulo são abordados os circuitos combinacionais; como é possível obter o circuito lógico a partir da função lógica e vice-versa; e são apresentados postulados, propriedades e teoremas que regem a Álgebra Booleana, fundamentais nas operações de simplificação de expressões lógicas.

Circuitos Combinacionais

Um circuito combinacional é aquele que executa uma expressão booleana através da interligação das várias portas lógicas existentes, de modo que as saídas dependem única e exclusivamente das entradas. Um circuito combinacional pode ser representado por um modelo genérico, como mostra a figura abaixo.



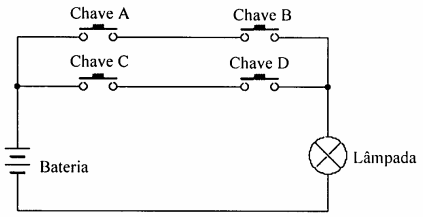
Modelo de Circuitos Combinacionais

O circuito combinacional constitui um subsistema digital ou parte de um sistema maior e mais complexo. Normalmente, o circuito combinacional é formado à partir de expressões lógicas.

Expressões Lógicas

Uma expressão lógica descreve uma função ou uma operação a ser concretizada por um sistema lógico (um circuito eletrônico ou um programa), de forma a resolver um determinado problema.

Exemplo: Dado o circuito a seguir e a definição das variáveis, qual é a expressão lógica que descreve seu funcionamento ?



L → Lâmpada (acesa = 1)

A, B, C, D → Chaves A, B, C e D (fechada =1)

Logo, **L = A . B + C. D**

Expressão Lógica L = A . B + C. D: Modelo Elétrico

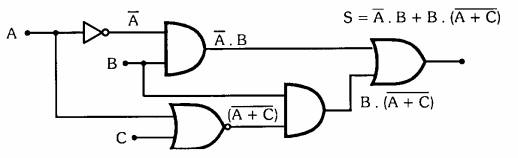
Observa-se, pelo exemplo, que a obtenção da expressão lógica que representa um sistema é muito simples. Mas, se o número de variáveis de entrada aumenta, a análise do problema fica mais complexal. Se o número de saídas também aumenta fica claro a necessidade do uso de outros métodos para se obter a expressão lógica. Esses métodos serão descritos a seguir.

Obtenção do Circuito Combinacional a partir da Função Lógica

Dada uma expressão booleana qualquer, pode-se implementar o circuito combinacional correspondente associando portas lógicas de acordo com as operações lógicas envolvidas na expressão. Para obtermos o circuito a partir da expressão utilizamos algumas regras semelhantes às existentes na matemática: os parênteses representam funções ou conjunto de funções e estão no topo da hierarquia de privilégios; a função NOT está em segundo lugar; uma função totalmente invertida (tal como a NAND) somente poderá ter a função inversora removida aplicando-se o teorema de De Morgan (a ser conhecido); em terceiro lugar, a função AND; e, finalmente, a função OR.

Exemplo: Dada a expressão booleana: $S = \overline{A} \cdot B + B \cdot (\overline{A + C})$, obter o circuito correspondente.

Solução:



Circuito Lógico $S = \overline{A} \cdot B + B \cdot (\overline{A + C})$

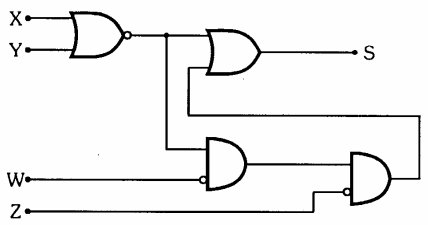
Obtenção da Função Lógica a partir do Circuito Combinacional

Dado um circuito combinacional qualquer, pode-se obter sua função lógica ou expressão booleana correspondente, associando as variáveis de entrada entre si através das operações lógicas envolvidas no circuito.

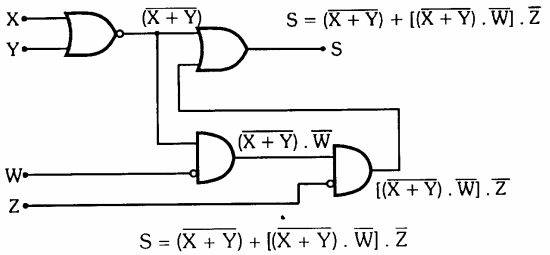
Para obtermos a expressão lógica a partir do circuito devemos nomear os sinais presentes nas saídas das portas lógicas, montar a expressão da saída mais à direita e depois substituir passo a passo até que a expressão geral esteja completamente escrita em função das entradas. Como exemplo, observar o circuito da figura abaixo e os passos dados para a obtenção da expressão geral.

Exemplo:

A partir do circuito combinacional, obter a expressão booleana resultante:



Solução:

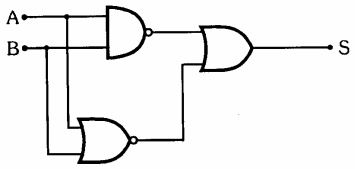


Obtenção da Função Lógica a partir do Circuito Combinacional

Obtenção da Tabela-Verdade a partir do Circuito Combinacional

Atribuindo níveis lógicos às variáveis de um circuito combinacional, pode-se obter a sua tabela-verdade completa.

Exemplo: Seja o circuito a seguir, obter sua tabela-verdade.



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

Solução:
Circuito com 2 variáveis de entrada, sua tabela é composta por 4 linhas, cada uma com uma situação de entrada.

Obtenção da Tabela-Verdade a partir do Circuito Combinacional

Simplificação de Expressões Booleanas

Foi visto, que é possível obter um circuito lógico através de uma expressão booleana. No entanto, o resultado obtido nem sempre é satisfatório visto que, às vezes, o circuito resultante pode ser muito complexo ou muito denso. Neste tópico, serão vistos os métodos de simplificação de expressões booleanas com o propósito de minimizar o circuito lógico equivalente.

A simplificação é um processo de manipulação algébrica das funções lógicas com a finalidade de reduzir o número de variáveis e operações necessárias para sua realização.

Postulados, Propriedades e Teoremas

O objetivo do estudo da álgebra booleana é a manipulação algébrica das funções lógicas. Em eletrônica digital e em informática, esta manipulação visa a simplificação das expressões lógicas. A manipulação algébrica das expressões é feita tomando-se como base os postulados, teoremas e propriedades da teoria desenvolvida por Boole e Shannon. A seguir, são apresentados estes postulados, propriedades e teoremas.

Se A = 0 então $\overline{A} = 1$	
Se A = 1 então $\overline{A} = 0$	
$\overline{\overline{A}} = A$	
0 + 0 = 0	0 . 0 = 0
0 + 1 = 1	0 . 1 = 0
1 + 0 = 1	1 . 0 = 0
1 + 1 = 1	1 . 1 = 1
A + A = A	A . A = A
A + 0 = A	A . 0 = 0
A + 1 = 1	A . 1 = A
$A + \overline{A} = 1$	$A . \overline{A} = 0$

Postulados Booleanos

Todos estes postulados podem ser provados de forma imediata, baseados nas funções lógicas básicas: AND, OR e NOT.

Comutativa
A . B = B . A
A + B = B + A
Associativa
A . (B . C) = (A . B) . C
A + (B + C) = (A + B) + C
Distributiva
A . (B + C) = A . B + A . C
A + (B . C) = (A + B) . (A + C)

Propriedades Booleanas

Teoremas de De Morgan
$\overline{A + B} = \overline{A} . \overline{B}$
$\overline{A . B} = \overline{A} + \overline{B}$
Teoremas da Absorção
A + A . B = A
$A + \overline{A} . B = A + B$

Teoremas Booleanos

Estes teoremas são genéricos e válidos para qualquer número de variáveis. Eles podem ser facilmente demonstrados com a ajuda de suas tabelas-verdade.

Simplificação de Expressões através dos Postulados, Propriedades e Teoremas

Não existe um método ou "receita de bolo" para realizar simplificações desta forma. É necessário "visão lógica" e a utilização correta dos postulados, propriedades e teoremas de acordo com a necessidade, visando sempre reduzir ao máximo o número de variáveis e operações lógicas da expressão. Exemplo: Simplificar a expressão: $S = \overline{X}.\overline{Y.Z}.W + X.\overline{Y.Z}.\overline{W} + X.\overline{Y.Z}.W$

Solução:

$$S = \overline{X}.\overline{Y.Z}.W + X.\overline{Y.Z}.\overline{W} + X.\overline{Y.Z}.W$$
$$S = \overline{Y.Z}.(\overline{X}.W + X.\overline{W} + X.W)$$
$$S = \overline{Y.Z}.[X.\overline{W} + W.(\overline{X} + X)]$$
$$S = \overline{Y.Z}.(X.\overline{W} + W)$$

Pelo teorema da absorção, obtém-se: $S = \overline{Y.Z}.(W + X)$

A vantagem de se simplificar as expressões lógicas está na simplificação dos circuitos , resultando em menores custo, consumo, maiores velocidade, facilidade de manutenção e confiabilidade.

Variáveis Relevantes e Irrelevantes

Nos sistemas digitais existem situações em que pelo menos uma variável de entrada se torna irrelevante na determinação da saída, isso devido ao estado que pelo menos uma outra variável assume. Observar o exemplo da expressão Y = A + B*:*

a) se a variável A assume o estado "1", a saída Y está em "1" independentemente do estado da variável B, pode-se, então, dizer que nesta situação o estado de B é irrelevante. Em outras palavras:

Y = A + B = "1" + B = "1" + X = "1" (X significa estado "0" ou "1") ;

b) se A assume o estado "0", então, a saída Y dependerá de B. Em outras palavras:

Y = A + B = "0" + B = B (se A = "0").

A tabela-verdade da função OR pode ser apresentada, então, em função dos estados de suas entradas:

A	B	Y = A + B
0	X	B
1	X	1
X	0	A
X	1	1

Conceitos de Relevância

Soma de Produtos e Produto de Somas

Como visto, a função AND é representada por um ponto. Por isso, em muitos casos, a função AND é referida como PRODUTO. A palavra produto perde seu sentido original quando usada para representar a operação AND, e serve apenas como um símbolo matemático para esta operação. Da mesma forma, a função OR é representada por um sinal de SOMA, mas é apenas um símbolo. Os termos PRODUTO e SOMA foram introduzidos na álgebra booleana para facilitar a discussão e a descrição das expressões lógicas.

São exemplos de expressões lógicas:

- Uma "SOMA DE PRODUTOS": $A.B + \overline{A}.C + B.\overline{C}$
- Um "PRODUTO DE SOMAS": $(M + N).(\overline{A}).(R + \overline{W})$

Alguns autores utilizam estes termos para expressar as técnicas de se obter as expressões lógicas à partir da tabela-verdade. Obter uma expressão em termos de *soma de produtos* é obter uma expressão que consiste do OR de todos os *minitermos* que levam a saída para "1", onde o minitermo é uma combinação AND que a torna verdadeira. O *minitermo* é encontrado segundo a seguinte convenção: se a variável de entrada é "1", então ela é uma entrada da função AND direta, caso contrário (se "0") ela precisa ser negada (invertida). Seja a função "OR":

A	B	Y	Minitermos
0	0	0	Nenhum
0	1	1	$Y_1 = \overline{A}.B$
1	0	1	$Y_2 = A.\overline{B}$
1	1	1	$Y_3 = A.B$

$Y = Y_1 + Y_2 + Y_3 = \overline{A}.B + A.\overline{B} + A.B = A + B$ (prova abaixo)

$Y = Y_1 + Y_2 + Y_3 = \overline{A}.B + A.\overline{B} + A.B + A.B$ (repetindo A.B)

$Y = A.(\overline{B} + B) + B(\overline{A} + A) = A.\text{"1"} + B.\text{"1"} = A + B$

O *produto de somas* é a expressão formada pelo AND de todos os *maxitermos* que levam a saída para "O". O *maxitermo* é dado pela função OR de todas as variáveis cujos estados a levam a "0" e, consequentemente, levam a função principal (0.X = 0). Para encontrar os maxitermos da função, utiliza-se a seguinte convenção: se a variável de entrada é "0", então ela é uma entrada da função OR direta, caso contrário ela precisa ser negada. Seja a função AND:

A	B	Y	Maxitermos
0	0	0	$Y_0 = A + B$
0	1	0	$Y_1 = A + \overline{B}$
1	0	0	$Y_2 = \overline{A} + B$
1	1	1	Nenhum

$$Y = Y_0 . Y_1 . Y_2 = (A + B) . (A + \overline{B}) . (\overline{A} + B)$$

Prova:

$Y = Y_0 . Y_1 . Y_2 = (A + B) . (A + \overline{B}) . (\overline{A} + B) = (A + B) . (A.\overline{A} + A.B + \overline{B}.\overline{A} + \overline{B}.B)$

$= (A + B)(\text{"0"} + A.B + \overline{B}.\overline{A} + \text{"0"}). = (A.A.B + A.\overline{B}.\overline{A} + B.A.B + B.\overline{B}.\overline{A})$

$= A.B + 0 + A.B + 0 = A.B$

Mapas de Karnaugh

O mapa de Veitch-Karnaugh, ou simplesmente mapa de Karnaugh, é uma tabela que visa facilitar o processo de minimização das expressões lógicas. Ele é formado por 2^n células, onde n é o número de variáveis de entrada; tendo, portanto, tantas células quanto o número de linhas da tabela-verdade. Em um Mapa de Karnaugh, a representação da relação entre as variáveis de entrada e suas saídas correspondentes é feita de modo que cada célula corresponda a uma condição de entrada:

- As saídas são indicadas dentro das células correspondentes;
- A disposição das células entre si é tal que facilita o enlace entre células adjacentes.

Os conceitos de adjacência e enlace são de fundamental importância para a compreensão e aplicação do mapa de Karnaugh:

- Adjacência: duas células são adjacentes entre si quando apenas uma de suas variáveis de entrada muda de valor. Exemplo: A tabela verdade de duas variáveis (porta OR) pode ser representada por quatro células:

  AB = 00 [0] AB = 01 [1] AB = 10 [1] AB = 11 [1]

  Pode-se afirmar que:

  As células AB = 00 e AB = 01 são adjacentes (apenas B muda de valor);
  As células AB = 00 e AB = 10 são adjacentes (apenas A muda de valor);
  As células AB = 01 e AB = 10 não são adjacentes (A e B mudam de valor)

- Enlace: é o agrupamento de células adjacentes com saídas iguais, do qual se pode extrair diretamente uma expressão booleana simplificada. Esta simplificação advém da aplicação do teorema da absorção. Assim, num enlace entre duas células adjacentes, pode-se extrair uma expressão booleana simplificada já que a variável que muda de valor desaparece.

Os mapas de Karnaugh consistem de estruturas tabulares e o uso do conceito de vizinhança associada a estados relevantes e irrelevantes das entradas que tornam uma função verdadeira. Observar a tabela verdade da função OR e suas transformações:

B	A	Y = B + A
0	0	0
0	1	1
1	0	1
1	1	1

B	A	Y = B + A
0	0	0
0	1	1
1	1	1
1	0	1

B	A	Y = B + A
0	X	A
X	1	1
1	X	1
X	0	B

		(B) 0	(B) 1
(A)	0		1
	1	1	1

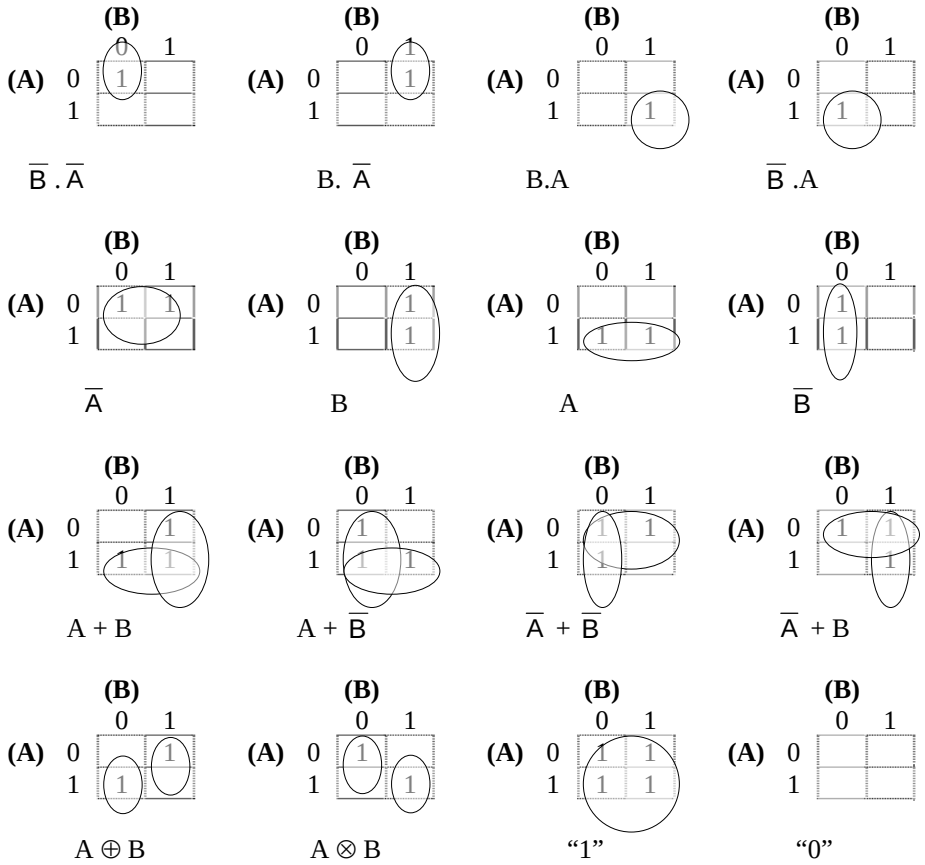
Construção do Mapa de Karnaugh de 2 Variáveis

Na segunda tabela foi modificada a seqüência dos estados. Na terceira, estabelece-se a saída em função de valores de A e B (relevâncias e não). Na quarta, encontra-se o mapa de Karnaugh para duas variáveis preenchido de modo que sua solução resulta em Y = B + A. Nos mapas não há necessidade de marcar as células em "0".

O conceito de vizinhança está associado com a montagem do mapa, de modo que cada célula está associada à um minitermo da função e entre as ordens binárias adjacentes apenas variar um bit; a variável que variar não entra na expressão. No mapa da função OR há dois enlaces (ou ilhas): "A em 1"' (onde B assume "0" e "1) e "B em 1"' (onde A assume "0" e "1).

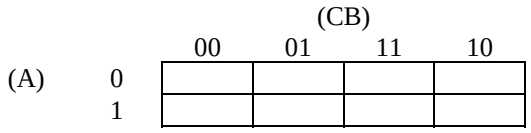
Mapas de Karnaugh de 2 Variáveis

Com duas variáveis são possíveis 16 mapas de Karnaugh:

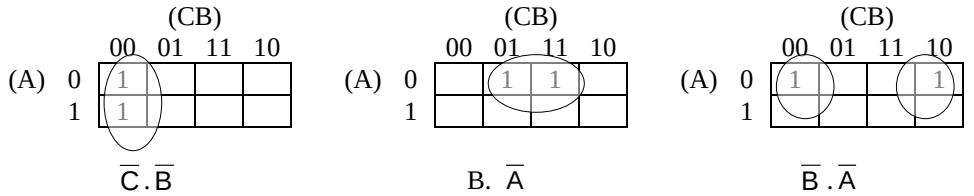


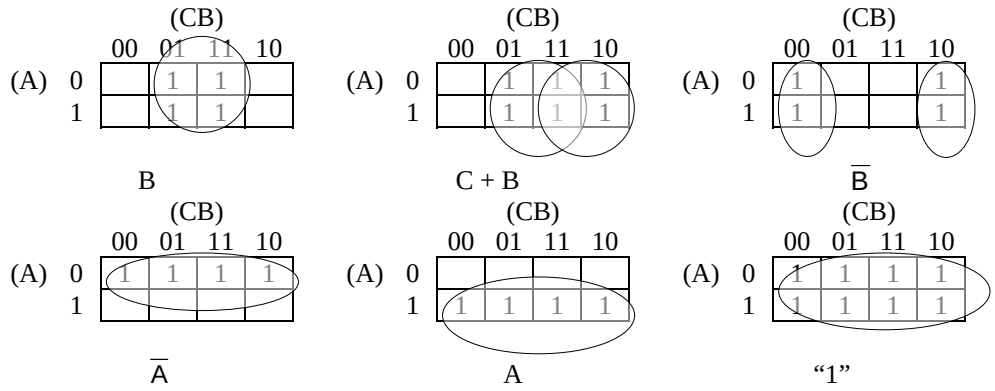
Mapa de Karnaugh de 3 Variáveis

Os mapas com três variáveis possuem a estrutura abaixo: as variáveis mais significativas são colocada à esquerda e acima (no exemplo C e B) e a menos significativa ao lado (A). Os exemplos, a seguir, apresentam alguns possíveis enlaces:



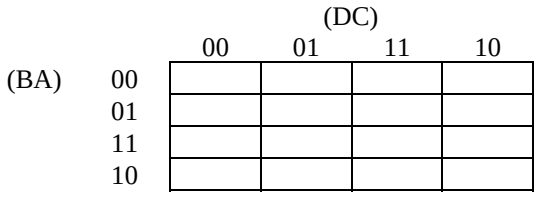
Mapas de Karnaugh de 3 Variáveis



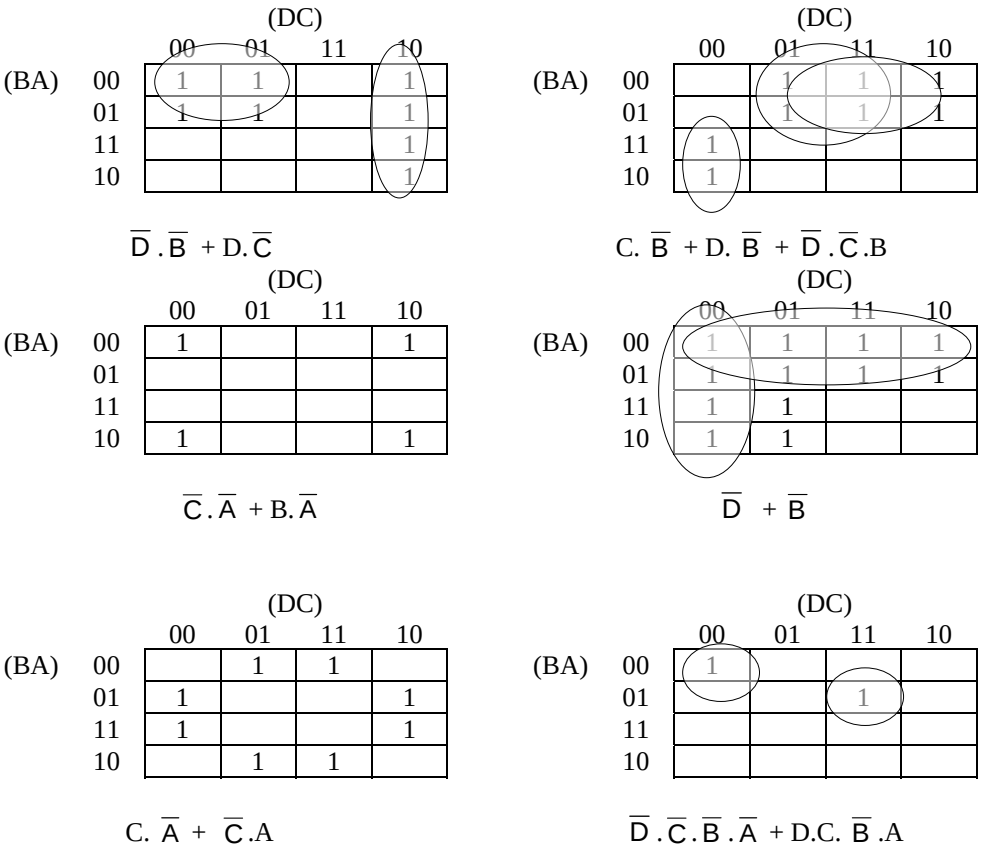


Mapas de Karnaugh de 4 Variáveis

Os mapas com quatro variáveis possuem a estrutura abaixo; as duas variáveis mais significativas ficam acima (nas colunas, no exemplo, D e C) e as menos significativas ao lado (linhas, B e A). Os exemplos abaixo apresentam algumas das possíveis vizinhanças



Mapa de Karnaugh de 4 Variáveis

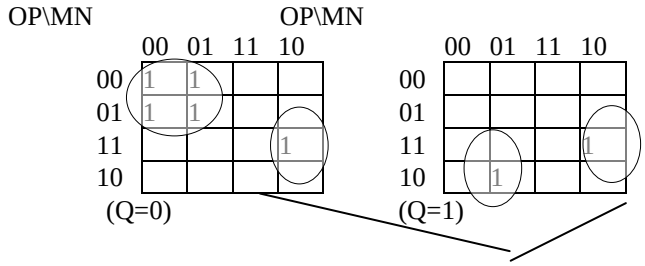


Mapas de Karnaugh acima de 4 variáveis

Como visto os mapas de Karnaugh possuem estruturas simples, porém para número de variáveis acima de quatro, faz-se necessário a construção de múltiplos mapas de quatro variáveis e a visualização de ilhas entre mapas.

Mapas de Karnaugh de 5 variáveis Y = f (M, N, O, P, Q):

Um mapa de 5 variáveis pode ser implementado com dois mapas de quatro, geralmente formados por quatro das variáveis e identificados pela quinta variável. Seja o exemplo abaixo:



Mapa de Karnaugh de 5 Variáveis

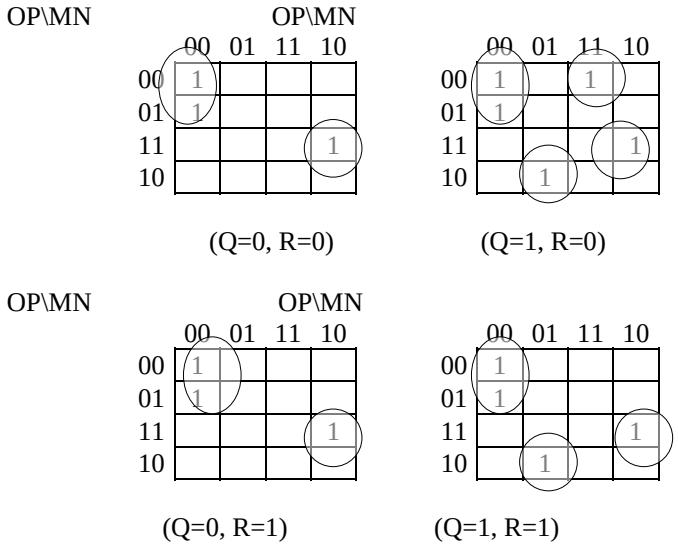
Neste mapa obtém-se um enlace de quatro células no primeiro mapa (Q=0), um enl;ace de duas células presente em ambos os mapas e um enlace de uma célula apenas no segundo (Q=1). Considerando esta ordem, a solução encontrada é dada pela expressão:

$$Y = \overline{M}.\overline{O}.\overline{Q} + M.\overline{N}.O.P + \overline{M}.N.O.\overline{P}.Q$$

Mapas de Karnaugh de 6 variáveis Z = f (M, N, O, P, Q, R):

No exemplo com cinco variáveis, a variável Q foi utilizada para identificar os mapas de quatro variáveis Q=0 e Q=1. Se um enlace existe nos dois mapas em enlaces correspondentes às mesmas combinações, então a variável de identificação de mapas (Q) não está presente na expressão correspondente ao enlace.

No mapa de seis variáveis são construídos quatro mapas de quatro variáveis os quais são endereçados pelas duas outras variáveis:



Mapa de Karnaugh de 6 Variáveis

Neste mapa há um enlace com oito células formado por enlaces de 2 células presentes em todos os mapas de quatro; ainda, da mesma forma, um enlace de quatro células formado por células de tamanho 1; um enlace de 2

células, onde uma se encontra em (Q,R) = (0,1) e outra em (Q, R) = (1, 1) e; finalmente, um enlace de uma célula presente em (Q,R) = (0,1). Considerando-se esta ordem, a solução para este papa é dada pela expressão:

$$Z = \overline{M}\,\overline{N}\,\overline{O} + M\overline{N}OP + \overline{M}NO\overline{P}R + MN\overline{O}\,\overline{P}\,\overline{Q}\,\overline{R}$$

Observa-se que quanto maior a ilha menor a expressão simplificada e esta é a idéia principal do uso dos mapas de Karnaugh. Ainda, que:

- Um enlace envolvendo uma única célula não permite simplificação, salvo se houver correspondentes em outros mapas auxiliares (se houver vizinhança entre mapas, no caso dos mapas com mais de quatro variáveis);
- Dois enlaces podem ter uma ou mais células em comum;
- Quanto menor o número de enlaces, menos termos tem a expressão booleana resultante;
- A solução de um mapa com enlaces maiores do que possível ou com enlaces desnecessários resulta, também, em uma expressão booleana correta, porém, não totalmente simplificada;
- Nas aplicações em que estado de uma célula pode assumir qualquer valor (adota-se a letra X) a esta deve atribuído um (1 ou 0) de modo que os enlaces formados sejam os maiores possíveis, resultando, então, em maior simplificação.

5. Representação Convencional e ABNT/IEEE

As portas lógicas são implementadas em circuitos integrados (CI's) com tecnologias diversas tais como a TTL e a CMOS. Portas lógicas AND, OR, NAND e NOR podem ser encontradas comercialmente com duas, três, quatro ou oito entradas.. A tabela abaixo apresenta alguns CI's antigos da família tecnológica TTL:

TTL	CMOS	Especificações
7400	4011	4 portas NAND de 2 entradas
7402	4001	4 portas NOR de 2 entradas
7404	4009	6 portas INVERSORAS
7408	4081	4 portas AND de 2 entradas
7432	4071	4 portas OR de 2 entradas
7486	4030	4 portas XOR de 2 entradas
7410	4023	3 portas NAND de 3 entradas
7427	4002	2 portas NOR de 4 entradas
7430	74C30	1 porta NAND de 8 entradas

Uma tecnologia de circuitos integrados estabelece diversas especificações que definem famílias de CI's, tais como tensão elétrica de alimentação, correntes e tensões mínimas e máximas de operação, velocidades de resposta da saída em função dos níveis lógicos de entrada, disposição das portas lógicas nos CI's, entre outras.

As figuras, a seguir, apresentam alguns exemplos de CI's nas representações convencional e nas adotadas pela ABNT - Associação Brasileira de Normas Técnicas. Os diagramas ABNT são recomendações IEEE e adotadas em mais de uma centena de países membros do Instituto dos Engenheiros Elétricos e Eletrônicos (IEEE), órgão internacional composto por mais de 300.000 membros entre organizações, profissionais e estudantes de Engenharia Elétrice e Eletrônica e das Ciências da Computação. Ser um membro IEEE é importante se se deseja atualização e competitividade no mundo globalizado atual. Maiores informações podem ser encontradas em http:\\www.ieee.org.

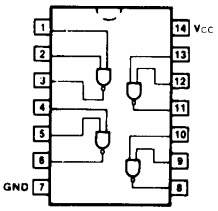


DIAGRAMA DE CONEXÃO – 74H00PC

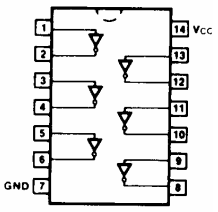
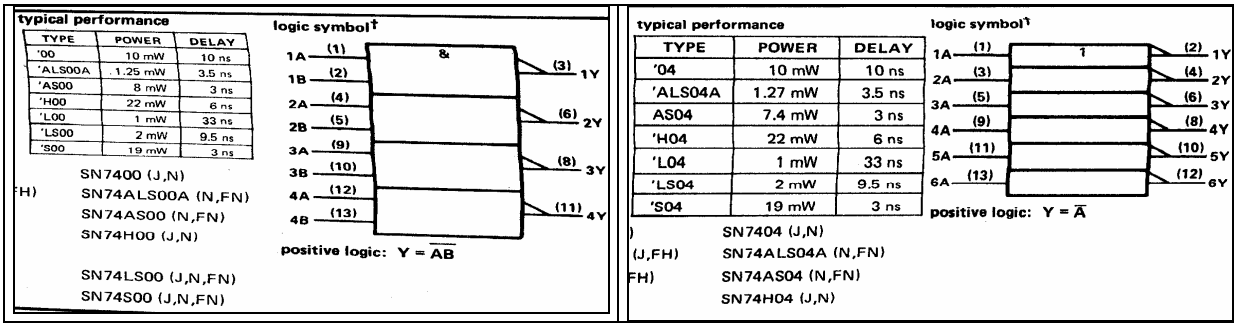


DIAGRAMA DE CONEXÃO – 7404PC



typical performance

TYPE	POWER	DELAY
'00	10 mW	10 ns
'ALS00A	1.25 mW	3.5 ns
'AS00	8 mW	3 ns
'H00	22 mW	6 ns
'L00	1 mW	33 ns
'LS00	2 mW	9.5 ns
'S00	19 mW	3 ns

SN7400 (J,N)
SN74ALS00A (N,FN)
SN74AS00 (N,FN)
SN74H00 (J,N)

SN74LS00 (J,N,FN)
SN74S00 (J,N,FN)

logic symbol†

positive logic: $Y = \overline{AB}$

typical performance

TYPE	POWER	DELAY
'04	10 mW	10 ns
'ALS04A	1.27 mW	3.5 ns
AS04	7.4 mW	3 ns
'H04	22 mW	6 ns
'L04	1 mW	33 ns
'LS04	2 mW	9.5 ns
'S04	19 mW	3 ns

SN7404 (J,N)
SN74ALS04A (N,FN)
SN74AS04 (N,FN)
SN74H04 (J,N)

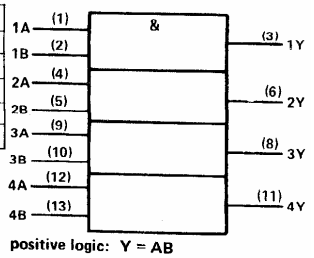
logic symbol†

positive logic: $Y = \overline{A}$

logic symbol†

TYPE	POWER	DELAY
'08	19 mW	15 ns
'ALS08	2.19 mW	6.5 ns
AS08	13 mW	4 ns
'LS08	4.25 mW	12 ns
'S08	32 mW	4.75 ns

SN7408 (J,N)
H) SN74ALS08 (N,FN)
) SN74AS08 (N,FN)
) SN74LS08 (J,N,FN)
SN74S08 (J,N,FN)



positive logic: Y = AB

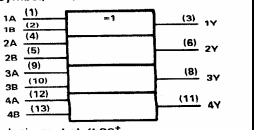
86 QUADRUPLE 2-INPUT EXCLUSIVE-OR GATES

typical performance

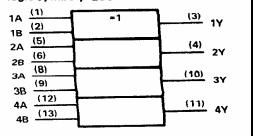
TYPE	POWER	DELAY
'86	150 mW	14 ns
'ALS86		
'L86	15 mW	55 ns
'LS86	30 mW	10 ns
'S86	250 mW	7 ns

SN5486 (J,FH) SN7486 (J,N)
SN54ALS86 (J,FH) SN74ALS86 (N,FN)
SN54L86 (J)
SN54LS86 (J,FH) SN74LS86 (J,N,FN)
SN54S86 (J,FH) SN74S86 (J,N,FN)

logic symbol, '86, 'ALS86, 'LS86, 'S86† pin



logic symbol, 'L86†



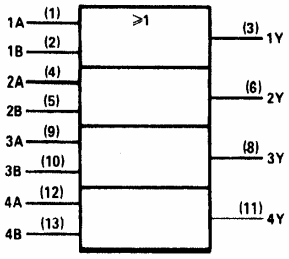
positive logic: $Y = A \oplus B = \bar{A}B + A\bar{B}$

typical performance

TYPE	POWER	DELAY
'32	24 mW	12 ns
'ALS32	2.81 mW	5.5 ns
'AS32	14.5 mW	4.5 ns
'LS32	5 mW	12 ns
'S32	35 mW	4 ns

) SN7432 (J,N)
J,FH) SN74ALS32 (N,FN)
FH) SN74AS32 (N,FN)
FH) SN74LS32 (J,N,FN)
H) SN74S32 (J,N,FN)

logic symbol†



positive logic: Y = A+B

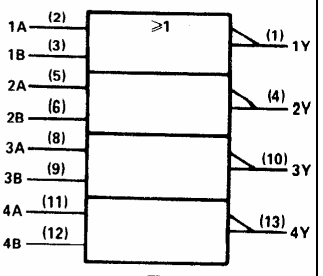
typical performance

TYPE	POWER	DELAY
'02	14 mW	10 ns
'ALS02	1.89 mW	5.5 ns
'AS02	12 mW	3 ns
'L02	1.5 mW	33 ns
'LS02	2.75 mW	10 ns
'S02	29 mW	3.5 ns

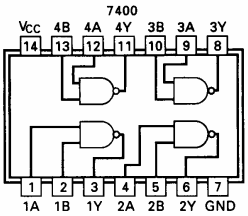
SN7402 (J,N)
SN74ALS02 (N,FN)
SN74AS02 (N,FN)

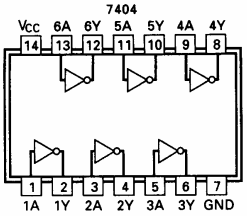
SN74LS02 (J,N,FN)
SN74S02 (J,N,FN)

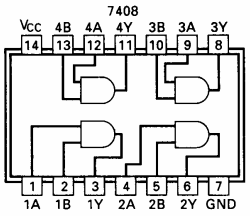
logic symbol†

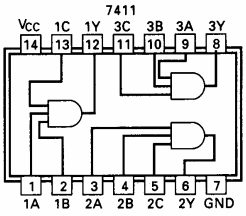


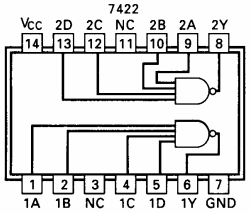
positive logic: $Y = \overline{A + B}$

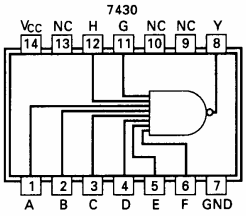


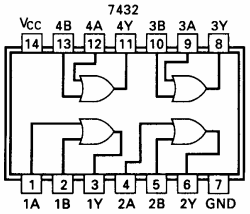


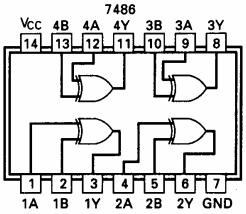












Algumas pastilhas SSI. *Layouts* de pinos de *The TTL Data Book for Design Engineers*, direitos reservados de Texas Instruments Incorporated, 1976.

6. Circuitos Aritméticos Básicos

Neste Capítulo são abordados os conceitos de Soma Binária, Subtração Binária e de ULA e principais expressões e circuitos que implementam estas funções binárias.

Somadores Binários

Os somadores binários são circuitos digitais das Unidades Lógicas e Aritméticas (ULA). O processo da soma binária é semelhante ao aplicado com decimais:

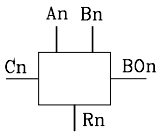
- primeiramente alinha-se as parcelas à direita para que as ordens destes números fiquem coincidentes;
- depois soma-se da direita para a esquerda, numeral a numeral, obtendo um resultado a cada ordem;
- quando o resultado ultrapassa a base (dez) faz-se o resultado igual ao valor encontrado subtraindo-se a base e diz-se que, para a ordem seguinte, "vai um"; e assim prossegue-se até a última ordem.

Somadores em cascata

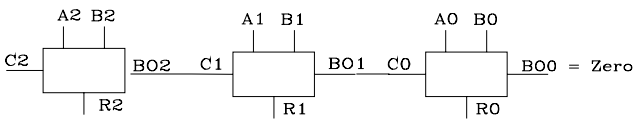
A soma é um processo repetitivo:

- na soma dos numerais de uma ordem é considerado o "vai um" da ordem anterior (se diferente de zero, que na ordem atual é chamado de "vem um" ou *Borrow*); e
- é encontrado um resultado e o "vai um" (ou *Carry*) para a ordem seguinte, se o resultado ultrapassa a base.

Pode-se, então, representar um somador de numerais genérico pelos símbolos abaixo que, quando ligados em cascata, possibilitam a geração de um sistema que soma números de infinitas ordens:



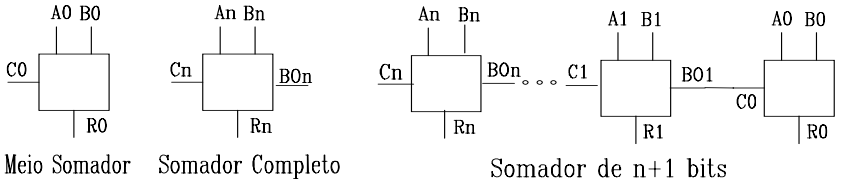
Somador Genérico



Somador Genérico de 2 números de 3 ordens

Com a tecnologia digital e os números binários, foram desenvolvidos circuitos que realizam todas as operações aritméticas conhecidas. No caso da soma, pode-se definir dois tipos de somadores genéricos:

- meio somador (*half adder*) que não tem "vem um" e é utilizado quando na soma de números de apenas 1 bit ou quando a parcela do número é a de menor ordem; e
- somador completo (*full adder*), que considera o "vem um" da parcela anterior e é utilizado para somar todas as parcelas do número com exceção das de menor ordem. A figura a seguir apresenta estes somadores e o modelo de um somador de n bits.



Em um sistema de n+1 bits o último "vai um" (Cn), quando "1" indica que o resultado da soma ultrapassou o maior número que o sistema pode representar. Esta situação é dita estouro ou *overflow*

Os circuitos meio somador e somador completo são encontrados à partir da solução dos mapas de Karnaugh dados abaixo:

Meio Somador

B0\A0

	0	1
0		1
1	1	

$R0 = A0 \oplus B0$

B0\A0

	0	1
0		
1		1

$C0 = A0.B0$

Somador Completo

BOn\AnBn

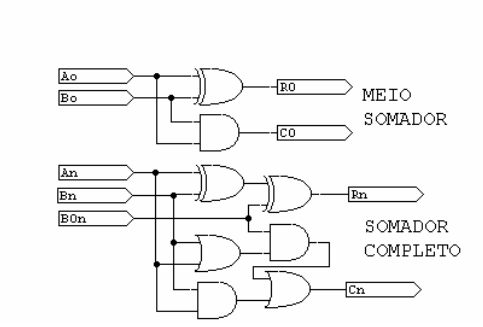
	00	01	11	10
0		1		1
1	1		1	

$Rn = An \oplus Bn \oplus BOn$

BOn\AnBn

	00	01	11	10
0			1	
1		1	1	1

$Cn = An.Bn + (An+Bn).BOn$



Circuitos Meio Somador e Somador Completo

7. Comparadores

Os comparadores são circuitos utilizados na verificação das relações existentes entre dois números, que podem ser igualdade, desigualdade, maior e menor.

Comparador de 2 bits

O comparador mais simples é o de dois bits, por exemplo A e B. Há quatro possíveis relações entre eles ou saídas:

- IGUAL, que é “1” se A = B ou “0”, caso contrário;
- DIFERENTE, que é “1” se A ≠ B, ou “0”, caso contrário;
- MAIOR, que é “1” se A > B, ou “0”, caso contrário; e
- MENOR, que é “1” se A < B, ou “0”, caso contrário.

As expressões e circuitos do comparador são encontrados à partir da solução dos mapas de Karnaugh à seguir:

B\A

	0	1
0	1	
1		1

A=B

B\A

	0	1
0		1
1	1	

A≠B

B\A

	0	1
0		1
1		

A>B

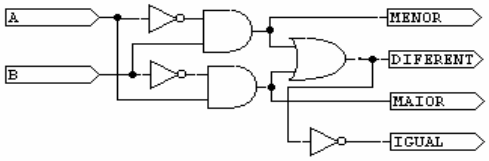
B\A

	0	1
0		
1	1	

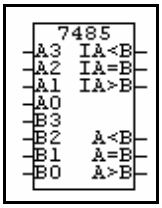
A<B

IGUAL = $A \otimes B$ DIFERENTE = $A \oplus B$

MAIOR = $A.\overline{B}$ MENOR = $\overline{A}.B$



CI 7485 é circuito integrado comparador mais conhecido na família TTL. Ele compara de números binários de 4 bits cada:



- A3, A2, A1 e A0 são as entradas do número binário A;
- B3, B2, B1 e B0 são as entradas do número binário B.
- IA>B, IA<B e IA=B são entradas de controle de cascata: se uma dela for "1" a saída correspondente (A>B, A<B ou A=B) também será "1"independentemente dos números binários de A e B.

Sua Tabela-Verdade é mostrada abaixo.

Entradas de comparação				Entradas de conexão em cascata			Saídas		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	$A > B$	$A < B$	$A = B$	$A > B$	$A < B$	$A = B$
$A_3 > B_3$	X	X	X	X	X	X	H	L	L
$A_3 < B_3$	X	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 > B_2$	X	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 < B_2$	X	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	X	X	X	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	X	X	X	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	H	L	L	H	L	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	H	L	L	H	L
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	L	L	H	L	L	H

H = Nível ALTO (HIGH) L = Nível BAIXO (LOW) *X* = Irrelevante

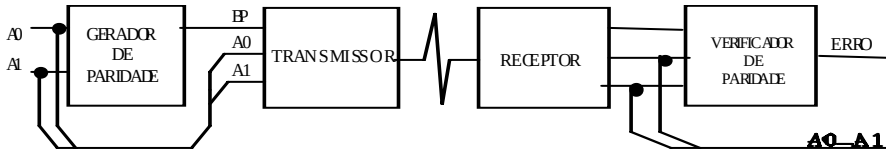
Figura 4.29 Tabela-verdade do 74LS85.

8. Paridade

A paridade é o mecanismo de segurança de dados binários mais simples de ser implementado e (por este motivo) é largamente utilizada nos computadores pessoais no armazenamento de informação em sua memória e na comunicação entre sistemas. Consiste, então, no acréscimo de um bit à informação (de tamanho pré-definido) chamado *bit de paridade*. Seu valor depende do número de bits em "1" da informação original e do tipo de paridade em uso.

A paridade pode ser *NULA*, ou sem paridade, onde não há acréscimo do bit de paridade; *PAR*, se a soma dos bits em '1" de informação + bit de paridade é par; e *ÍMPAR*, se a soma dos bits em '1" de informação + bit de paridade é ímpar.

O processo da paridade compreende em gerar e verificar a paridade. Quando uma informação (8 bits, por exemplo) é transferida seja para uma memória ou um outro sistema e faz-se necessária a integridade dos dados, um bit de paridade é gerado, anexo (no fim ou no começo) e transferido com a informação. No seu destino é verificado. Como os sistemas digitais estão continuamente sujeitos a distúrbios eletromagnéticos e mecânicos, o sistema receptor ou armazenador verifica se os dados estão de acordo com a paridade comprometida. Em uma comunicação, por exemplo, quando o receptor de dados detecta um erro (número de bits em "1" ímpar quando paridade comprometida par ou vice-versa), ele solicita ao transmissor a retransmissão da informação ou inicializa um outro procedimento. Se não detectado um erro, o bit de paridade é desprezado e a informação aceita.



Exemplo de Geração e Detecção de Paridade

Existem outros métodos que, utilizados em conjunto com a paridade, melhoram significativamente a segurança dos dados. A utilização de um mecanismo de segurança de dados depende da importância da informação no processo e do custo de geração, verificação e de restauração da mesma (se for o caso), entre outros fatores.

Gerador e Detector de Paridade de 2 bits

Seja um sistema de dois bits e paridade par. Os circuitos gerador e verificador de paridade são obtidos à partir da solução dos Mapas de Karnaugh dados a seguir.

A1\A0

	0	1
0		1
1	1	

BP\A1A0

	00	01	11	10
0		1		1
1	1		1	

$BP = A1 \oplus A0$ $ERRO = BP \oplus A1 \oplus A0$

Gerador e Verificador de Paridade Par para 2 bits de Informação

Pelos mapas verifica-se que os circuitos geradores e verificadores de paridade utilizam portas Oo-exclusivo.

9. Codificadores e Decodificadores

Códigos - Histórico

"Foi na Índia que nos deu o método engenhoso de exprimir todos os números por meios de dez símbolos, cada símbolo recebendo um valor absoluto; uma idéia consistente e importante que agora nos parece tão simples a ponto de ignorarmos seu verdadeiro mérito."

Marquês de Laplace

Como matemático, Laplace teve oportunidade de apreciar muito bem o sistema de numeração decimal. Nosso atual sistema de numeração proporciona aos modernos matemáticos e cientistas uma grande vantagem em relação aos das civilizações anteriores e constitui um fator importante em nosso rápido progresso.

O sistema decimal de contagem tem sido tão amplamente adotado por toda a nossa atual civilização que dificilmente consideramos as possibilidades de adotar-se outro sistema de numeração. Apesar disso, não é razoável esperar que um sistema baseado no número de dedos que possuímos seja o mais eficiente para a construção de máquinas. O fato é que um sistema pouco utilizado, porém muito simples, o sistema de numeração binário, é comprovadamente o mais natural e eficiente sistema para uso em máquinas.

Um matemático alemão do século XVII, Gottfried Wilhelm von Leibniz, foi o advogado do sistema binário de numeração, usando apenas os símbolos "0" e "1".

Se parece estranho que um eminente matemático advogue tal sistema de numeração muito simples, deve ser notado que ele foi também um filósofo. As razões de Leibniz para advogar o sistema binário parecem ter sido místicas. Ele sentiu que havia grande beleza na analogia entre zero, representando o vazio, e um, representando a Divindade.

Independente de quão boas fossem as razões de Leibniz para advogá-lo, o sistema binário tornou-se muito popular, pois os computadores digitais da atualidade são construídos para operar em sistemas binários ou de números codificados em binário.

Códigos Binários

Uma grande parte dos sistemas digitais utiliza informações codificadas binárias. Em uma calculadora, por exemplo, há informações numéricas e alfanuméricas para atender à representação de números, letras, símbolos, etc.. Quando um número é codificado em binário, ele é feito corresponder a uma seqüência de "0"s e "1"s e o código resultante é classificado como BCD(Binary-Coded Decimal).

Código BCD8421

O mais comum dos códigos BCD é o BCD8421, no qual cada algarismo decimal corresponde a seu equivalente binário. O nome BCD8421 deriva do peso atribuído a cada bit de código; como cada bit possui um valor posicional, o código BCD é dito com pesos(Weighted Code).

A vantagem do código BCD8421 é a facilidade da conversão de decimal para binário ou vice-versa. Uma desvantagem, contudo, é que as regras da adição binária não são diretamente aplicáveis a números codificados em BCD8421.

DECIMAL	BCD8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Código BCD excesso-3

Outro código BCD relativamente comum é o chamado código excesso-3. A vantagem desse código é que ele é auto-complementar, significando que um número decimal e o seu complemento 9(o que falta para 9) corresponde a binários também complementares.

Essa propriedade é útil na realização de operações aritméticas com números decimais codificados em BCD excesso-3. Atualmente, com o desenvolvimento dos circuitos integrados que executam operações lógicas e aritméticas, o código excesso 3 deixou de ter aplicação prática.

DECIMAL	BCD EXCESSO-3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

Código BCH

Código BCH (Binary Coded Hexadecimal), que significa hexadecimal codificado em binário, é análogo ao código BCD, porém representa os algarismos do sistema hexadecimal através das combinações possíveis com quatro bits.

DECIMAL	BCD8421	BCD EXCESSO-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

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

Outros Códigos BCD de 4 bits

Além dos códigos BCD8421 e BCD excesso-3, há diversos outros códigos BCD de 4 bits, alguns são mostrados na tabela a seguir, e que possuem aplicação mais restrita.

DECIMAL	7421	6311	5421	5311	5211	4221	3321	2421
0	0000	0000	0000	0000	0000	0000	0000	0000
1	0001	0001	0001	0001	0001	0001	0001	0001
2	0010	0011	0010	0011	0011	0010	0010	0010
3	0011	0100	0011	0100	0101	0011	0011	0011
4	0100	0101	0100	0101	0111	1000	0101	0100
5	0101	0111	1000	1000	1000	0111	1010	1011
6	0110	1000	1001	1001	1001	1100	1100	1100
7	1000	1001	1010	1011	1011	1101	1101	1101
8	1001	1011	1011	1100	1101	1110	1110	1110
9	1010	1100	1100	1101	1111	1111	1111	1111

Códigos BCD de 5 bits

Embora 4 bits sejam suficientes para representar todos os algarismos decimais, existem códigos BCD de 5 bits ou mais. Os bits adicionais facilitam a decodificação em alguns casos e permitem a detecção de erros com mais facilidade. O código 2-em-5, por exemplo, faz cada número decimal corresponder a um binário de 5 bits, dos quais dois são iguais a 1. Isso torna a paridade(# de "1"s na palavra) sempre par e facilita a detecção de erros. O código 2-em-5 é bastante utilizado em telefonia.

Outro código BCD de 5 bits importante é o código Johnson. Este código é facilmente decodificável. A mesma característica é apresentada no código BCD51111.

DECIMAL	2-EM-5	63210	JOHNSON	86421	51111
0	00011	00110	00000	00000	00000
1	00101	00011	00001	00001	00001
2	00110	00101	00011	00010	00011
3	01001	01001	00111	00011	00111
4	01010	01010	01111	00100	01111
5	01100	01100	11111	00101	10000
6	10001	10001	11110	01000	11000
7	10010	10010	11100	01001	11100
8	10100	10100	11000	10000	11110
9	11000	11000	10000	10001	11111

Códigos BCD de mais de 5 bits

Dos códigos BCD de mais de 5 bits os mais importantes são o biquinário ou 50 43210, o 543210 e o 9876543210 ou código anel. Os dois primeiros permitam fácil detecção de erros; o código anel é o de maior facilidade de decodificação, uma vez que a contagem já é feita decodificada.

DECIMAL	50 43210	543210	9876543210
0	01 00001	000001	0000000001
1	01 00010	000010	0000000010
2	01 00100	000100	0000000100
3	01 01000	001000	0000001000
4	01 10000	010000	0000010000
5	10 00001	100000	0000100000
6	10 00010	100001	0001000000
7	10 00100	100010	0010000000
8	10 01000	100100	0100000000
9	10 10000	110000	1000000000

Código Gray

Um código bastante utilizado é o código Gray. Esse não é um código BCD e sua característica notável é que entre dois números sucessivos apenas um bit varia. Esta particularidade é útil em certas aplicações, como conversores analógicos-digitais, controle de servomecanismos, etc.

A característica do código Gray é chamada de adjacência, muito útil no processo de simplificação nos mapas de Karnaugh.

DECIMAL	BINÁRIO	GRAY
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Códigos Alfanuméricos

Há códigos chamados alfanuméricos que permitem representar caracteres alfabéticos maiúsculos e minúsculos, algarismos, caracteres especiais(?,!,%, etc.) e caracteres de controle (CR - CARRIAGE RETURN, LF - LINE FEED, para controle de um teletipo ou terminal de vídeo). Dois desses códigos são padrões utilizados universalmente nos computadores digitais e equipamentos periféricos. São eles os códigos EBCDIC (Extended Binary-Coded Decimal Interchange Code) e ASCII (American Standard Code for Information Interchange).

O código EBCDIC é um código binário de 8 bits, utilizado principalmente para representação interna de caracteres em computadores digitais e o código ASCII é um código binário de 7 bits, utilizado principalmente em comunicação de dados.

Exemplos:

- Pressionando-se a tecla M do teclado de um computador, internamente é gerado o código (1001101);
- Pressionando-se a barra de espaços do teclado de um computador, internamente é gerado o código (0100000), que corresponde ao sinal de controle SP (Space).

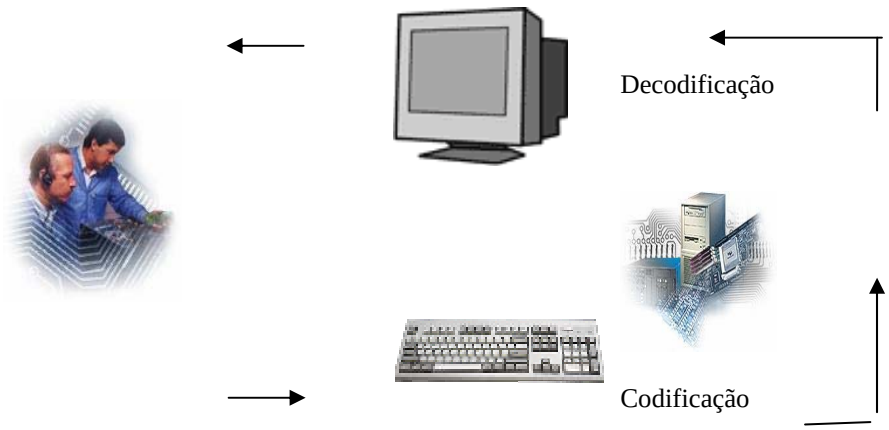
				B_7	0	0	0	0	1	1	1	1
				B_6	0	0	1	1	0	0	1	1
				B_5	0	1	0	1	0	1	0	1
B_4	B_3	B_2	B_1	HEXADECIMAL	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	(TC7) DLE	SP	0	@	P	'	p
0	0	0	1	1	(TC1) SDH	DC1	!	1	A	Q	a	q
0	0	1	0	2	(TC2) STX	DC2	"	2	B	R	b	r
0	0	1	1	3	(TC3) ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	(TC4) EDT	DC4	$	4	D	T	d	t
0	1	0	1	5	(TC5) ENQ	(TC8) NAK	%	5	E	U	e	u
0	1	1	0	6	(TC6) ACK	(TC9) ETB	&	6	F	V	f	v
0	1	1	1	7	BEL	(TC10) ETB	'	7	G	W	g	w
1	0	0	0	8	FE0 (BS)	CAN	(	8	H	X	h	x
1	0	0	1	9	FE1 (HT)	EM	)	9	I	Y	i	y
1	0	1	0	A	FE2 (LF)	SUB	*	:	J	Z	j	z
1	0	1	1	B	FE3 (VT)	ESC	+	;	K	[	k	{
1	1	0	0	C	FE4 (FF)	IS4 (FS)	,	<	L	\	l	|
1	1	0	1	D	FE5 (CR)	IS3 (6S)	-	=	M	]	m	}
1	1	1	0	E	SD	IS2 (RS)	.	>	N	^	n	~
1	1	1	1	F	SI	IS1 (GS)	/	?	O	_	o	DEL

Código ASCII

Codificadores e Decodificadores

Os codificadores são operadores que realizam a transformação da informação representada e manipulada pelo homem para um código binário. É, então, uma operação de entrada. Os decodificadores realizam a operação inversa, ou de saída

Nos sistemas digitais, circuitos codificadores convertem códigos para binário e os decodificadores fazem o inverso. É um circuito que recebe uma informação codificada de alguma forma e traduz para outra. A informação pode ser um número decimal codificado em binário, um endereço de uma posição de memória, etc.

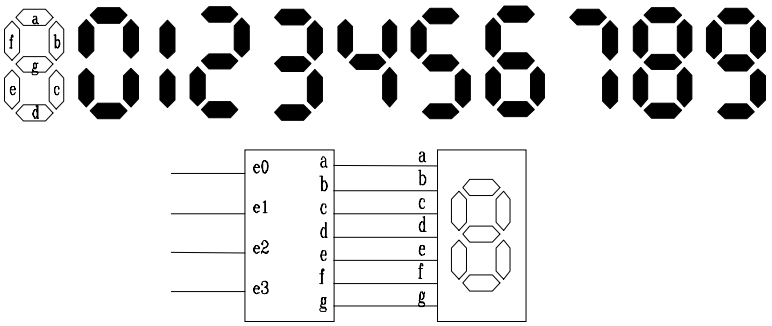


Decodificador BCD para displays de 7 segmentos

Um display de 7 segmentos é um dispositivo utilizado para apresentar informações numéricas de forma inteligível para o ser humano. Nos displays a informação é apresentada com a energização de segmentos de modo que seja visualizado um número decimal. Estes segmentos podem ser filamentos incandescentes, diodos emissores de luz ou cristais líquidos. Destes, o maior consumo de corrente ocorre no caso de filamentos incandescentes e o menor nos cristais líquidos.

Um display de sete segmentos baseado em LEDs (light emitting diode ou diodo emissor de luz) é constituído por sete leds denominados a, b, c, d e f e g, dispostos como apresenta a figura abeixo. Estes dispositivos podem ser catodo comum ou anodo comum. No tipo catodo comum todos os LED'S estão interligados pelo catodo (vide anexo A) e, geralmente, ao Terra (0 Volt) diretamente ou através de um transistor. Neste tipo, deve-se aplicar um nível lógico "1" ao segmento para que ele "acenda". No tipo anodo comum, todos os LED'S estão interligados pelo anodo e a VCC (+5 Volts) diretamente ou através de um transistor. Neste tipo, deve-se então aplicar um nível lógico "0" ao segmento para que ele "acenda".

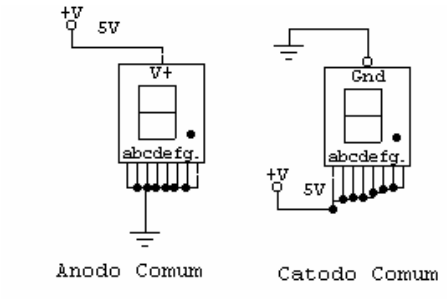
Construir um decodificador BCD de 7-segmentos significa desenvolver um circuito que à partir do código binário (geralmente o BCD) gere um código abcdefg que faça äcender corretamente os segmentos a, b, c, d, e, f, ou g que formarão o número decimal:



decodificador display

Os displays podem ser anodo comum ou catodo comum, cujos segmentos acendem quando recebem nível lógico 0 ou 1, respectivamente. A Tabela-Verdade para um display catodo comum é mostrada a seguir.

Anodo Comum Catodo Comum



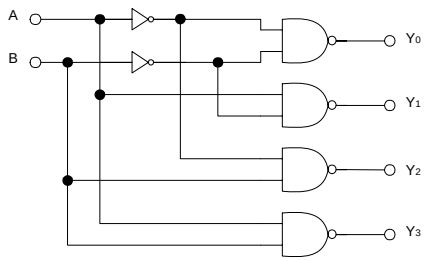
As tabelas, a seguir, apresentam o mapa de ativação das saídas do decodificador BCD para 7 segmentos e o Mapa de Karnaugh cuja solução corresponde ao circuito que acende o led a.

Entradas				Saídas						
e3	**e2**	**e1**	**e0**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x

e1e0 \ e3e2	00	01	11	10
00	1		x	1
01		1	x	1
11	1	1	x	x
10	1	1	x	x

Decodificador de n para 2^n linhas

Uma classe importante de decodificadores são os decodificadores de n para 2^n linhas (2 para 4, 3 para 8, 4 para 16, etc.). Estes decodificadores possuem n linhas de entrada e 2^n linhas de saída, das quais somente uma ficará ativa para cada combinação das linhas de entrada. A figura abaixo mostra um decodificador de 2 linhas para 4 linhas e sua tabela da verdade:



B	A	Y_0	Y_1	Y_2	Y_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Neste circuito somente a saída correspondente à combinação das entradas A e B estará ativa(neste caso ativa no estado "0") em cada instante.

O conceito apresentado neste circuito pode ser facilmente estendido para 3, 4 ou n linhas de entrada. Na prática, não é necessário construir estes decodificadores a partir de portas lógicas, pois há uma grande variedade de CI' s que realizam estas funções.

Codificador de 2^n linhas para n linhas

De forma análoga os codificadores de 2^n linhas para n (4 para 2, 8 para 3, 16 para 4, etc.). possuem papel importante na codificação da informação humana para a binária. Como exercício desenvolver o circuito de um codificador 4:2, seguindo a tabela verdade abaixo:

E_3	E_2	E_1	E_0	S_1	S_0
0	1	1	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	0	1	1

Neste circuito S_1S_0 deverá apresentar o código binário correspondente à ordem da entrada ativa. A tabela - verdade abaixo apresenta uma outra solução para o codificador admitindo-se que as entradas são ativas em "0"". Repitir, então, o exercício anterior considerando este estado de ativação

E_3	E_2	E_1	E_0	S_1	S_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

O CI 74138

Conceito. Os codificadores e decodificadores são circuitos cujos princípios de funcionamento e aplicações são bastante semelhantes aos multiplexadores e demultiplexadores, de forma tal que estes se confundem. Portanto, existem CI's que podem atender a várias aplicações tal como o CI 74138, largamente utilizado como decodificador 3 linhas para 8 endereços, como demultiplexador 1:8 e como codificador octal para decimal.

Finalmente, os circuitos codificadores são circuitos utilizados para converter um código em um outro (geralmente um código para binário), enquanto que os decodificadores geralmente bfazem a função inversa (de binário para outro código).

Endereçamento Geral, Seleção e Habilitação.

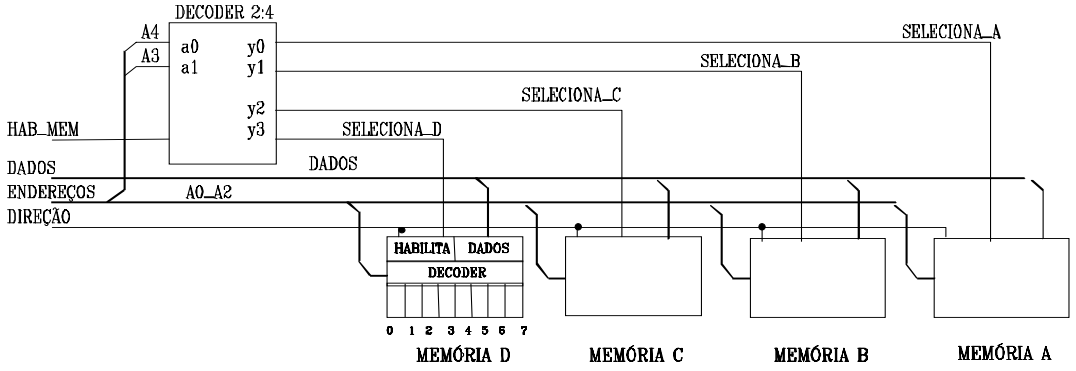
Nos computadores digitais as memórias são acessadas pelo processador central ou microprocessador (uP) através de sinais de identificação, chamados linhas de endereçamento.

Quando o uP deseja se comunicar com a memória, ele ativa as linhas de endereçamento, onde parte delas são responsáveis pela seleção da memória (linhas de endereço mais alto, que são entradas em um decodificador) e o restante são responsáveis pela identificação da <u>locação</u> da memória que será <u>acessada</u>.

A figura, a seguir, apresenta o modelo de uma memória que possui 32 locações de memória divididas em 4 CI's A, B, C e D. Cada CI possui então 8 locações (0, 1, 2, 3, 4, 5, 6 e 7) que são selecionadas pelas linhas de endereço A0, A1 e A2, <u>se</u> o sinal de seleção do CI <u>estiver ativado</u>. No exemplo, conforme o código binário em A4 e A3 (nas entradas a0 e a1 do decodificador) será ativada um dos 4 CI's se o sinal HAB_MEM estiver em "1" (estado ativo, ou de ativação).

Este sinal é importante, pois indica quando o acesso é à memória em questão, lembrando que existem muitos outros dispositivos em um computador que utilizam o mesmo princípio de endereçamento. A tabela, abaixo apresenta o mapa de endereçamento do exemplo.

A4	A3	A2	A1	A0	SE HAB_MEM = "1" SELECIONA
0	0	X	X	X	MEMÓRIA A
0	1	X	X	X	MEMÓRIA B
1	0	X	X	X	MEMÓRIA C
1	1	X	X	X	MEMÓRIA D



10. Multiplexadores e Demultiplexadores

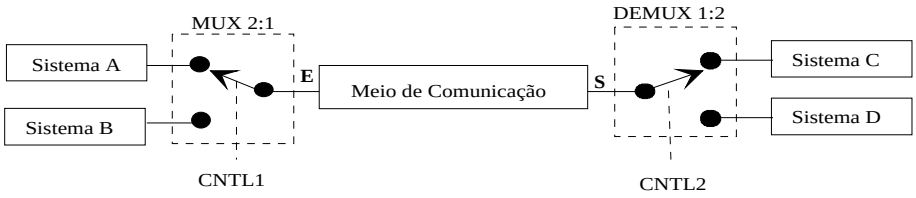
Os multiplexadores e demultiplexadores são circuitos largamente utilizados nos sistemas digitais, funcionando como chaves digitais de múltiplos contatos com os quais são realizadas conexões elétricas entre sinais digitais ou canais de comunicação.

O multiplexador ou MUX é um circuito combinacional que, através das variáveis de seleção ou de controle, conecta uma em m entradas em uma única saída. Esta operação é denominada multiplex ou multiplexação.

O demultiplexador ou DEMUX realiza a função inversa ou, através das variáveis de seleção, conecta sua entrada à uma de suas n entradas.

MUX 2:1 e DEMUX 1:2

Na figura abaixo, dois sistemas se conectam com outros dois utilizando um único meio de comunicação a cada instante. No início um multiplexador 2:1 (mux dois para um) permite que apenas um sistema faça uso do meio a cada instante e no fim, um demultiplexador 1:2 (demux um para dois) determina que sistema recebe a informação em trânsito.



Comunicação usando MUX e DEMUX

Sejam E e S, os sinais na entrada e na saída do meio de comunicação, respectivamente. Se desejado transmitir o sinal do sistema A, deve-se fazer E = A, caso contrário E = B. A função do MUX 2:1 é transferir para a sua saída o sinal de A ou de B, sob o comando do sinal de controle (CNTL1). Como CNTL1 assume dois estados, pode-se fazer com que se igual a "0" deixe o sinal A fluir para E, e se "1" deixe de B para E. A expressão lógica, então, do mux 2:1 é dada por:

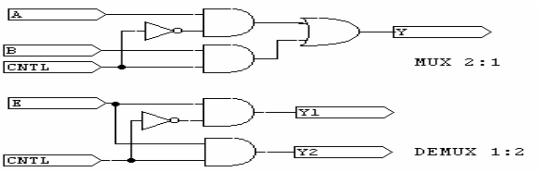
$E = A \cdot \overline{CNTL1} + B \cdot CNTL1,$

Na saída do meio de comunicação, o circuito DEMUX 1:2 transfere a informação para o sistema C quando CNTL2 = "0", ou para o sistema D, quando CNTL2 = "1". As expressões do demux 1:2 são dadas por:

$C = S \cdot \overline{CNTL\ 2}$ $\qquad$ $D = S.\ CNTL2$

As figuras, a seguir, apresentam os diagramas de bloco do mux 2:1 e do demux 1:2, expressões e circuitos lógicos:

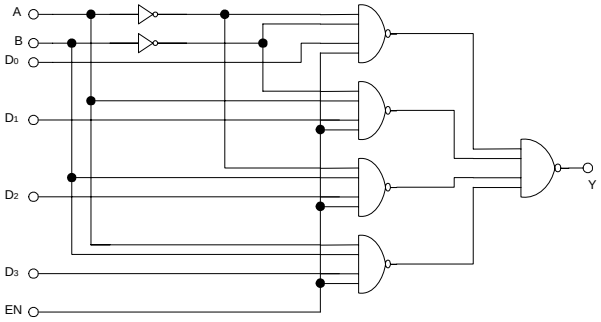
Diagramas de Blocos do MUX 2:1 e DEMUX 1:2



MUX 2:1 E DEMUX 1;2

MUX de 4 para 1 linha

É de grande utilidade um circuito que execute a função contrária do DEMUX, ou seja, selecione uma entre 2^n entradas e a direcione para uma linha de saída única. Este circuito é denominado MULTIPLEX, DATA SELECTOR ou simplesmente MUX, de 4 linhas para uma linha.

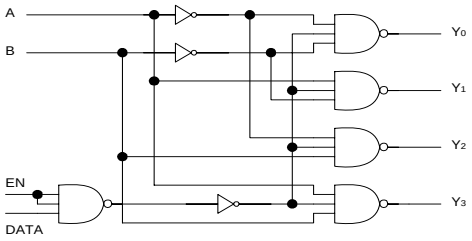


EN	B	A	Y
0	X	X	0
1	0	0	D_0
1	0	1	D_1
1	1	0	D_2
1	1	1	D_3

DEMUX de 1 para 4 linhas

O circuito decodificador de 2 para 4 linhas pode ser utilizado com algumas modificações para executar uma função muito útil, qual seja, direcionar uma informação a uma dentre 4 linhas de saída conforme o controle nas linhas A e B. O circuito resultante é denominado demultiplexador, demultiplexer, data distribuidor, ou simplesmente DEMUX, de 1 para 4 linhas.

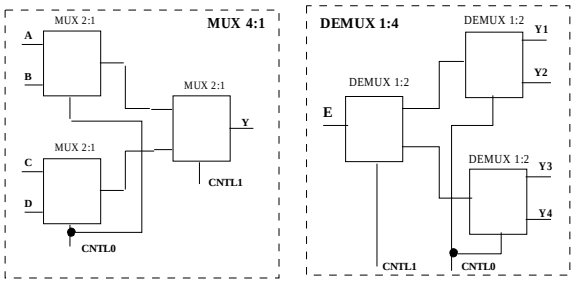
Neste circuito é acrescentada uma linha de habilitação ou ENABLE e pode-se verificar que se EN = "0" todas as saídas ficam inativas, independente das linhas de dados(DATA). Se EN = "1", o conteúdo da linha de dados é transferido invertido para a saída endereçada pelas linhas A e B. Há diversos tipos de DEMUX disponíveis na forma de CI' s e este circuito tem grande aplicação prática.



EN	B A D	Y0	Y1	Y2	Y3
0	X X X	1	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	0	1	1	1
1	0 1 1	1	0	1	1
1	1 0 1	1	1	0	1
1	1 1 1	1	1	1	0

Associação de MUX e de DEMUX

Vários outros circuitos multiplexadores e demultiplexadores são obtidos à partir da associações, tais como os apresentados pela figura abaixo:



M UX 4:1

CNTL1	CNTL0	Y
0	0	A
0	1	B
1	0	C
1	1	D

DEMUX 1:4

CNTL1	CNTL0	Y1	Y2	Y3	Y4
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

Vários CI's realizam as funções de mux e demux, tais como os da família TTL abaixo:

MUX

74151	74153	74157
74158	74251	74253
74257	74258	74298
74352	74353	

DEMUX

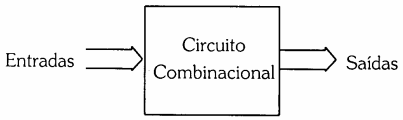
7442	7445	7447
7448	74138	74139
74154	74155	74247
74248		

MUX e DEMUX Comerciais da Família TTL

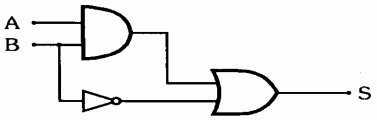
11. Lógica Seqüencial: Flip-Flops, Registradores e Contadores

Introdução

Os circuitos digitais podem ser classificados em combinacionais e seqüenciais. O circuito combinacional é aquele cujas saídas dependem unicamente das entradas. As saídas desses circuitos, em um certo tempo *t*, dependem apenas dos valores das entradas (excitações ocorridas durante t). A Figura, abaixo, representa um circuito combinacional genérico:



Seja o circuito combinacional:



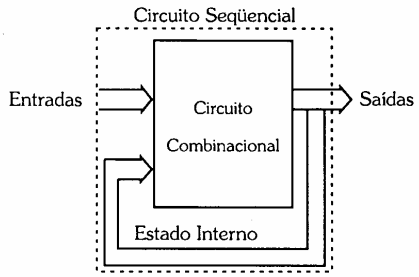
$S = AB + \overline{B}$

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

Neste circuito, S (a variável de saída) depende apenas de A e B (variáveis de entrada) como mostram a expressão lógica e a tabela-verdade.

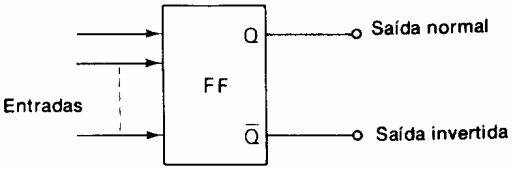
O circuito seqüencial é aquele que possui realimentação da saída para a entrada, denominada estado atual, de modo que as condições atuais da entrada e do estado atual determinam a condição futura da saída.

Nos circuitos seqüenciais, as saídas num tempo *t*, dependem não apenas das entradas em *t*, mas também da história das entradas, ou seja, do comportamento das entradas antes do tempo *t*. Em outras palavras, estes circuitos lógicos podem produzir saídas dependentes dos estados anteriores das entradas. A Figura abaixo representa um circuito seqüencial genérico:



Flip-Flops

O circuito de memória mais largamente nos Sistemas Digitais é o flip-flop (FF). O flip-flop tem como função armazenar temporariamente um nível lógico, como um elemento de memória.



Símbolo geral de um Flip-Flop

São características gerais dos flip-flops:

- Os flip-flops podem ter vários tipos de configurações e geralmente apresentam duas saídas complementares chamadas Q e $\overline{Q}$;
- A saída Q é chamada de saída normal do FF e $\overline{Q}$ é a saída invertida do FF.
- Quando se diz que o FF está no estado alto (1) ou no estado baixo (0), esta é a condição da saída Q.
- Existem dois estados possíveis de operação para o FF: $Q = 0$ e $\overline{Q} = 1$; e $Q = 1$ e $\overline{Q} = 0$
- O FF possui uma ou mais entradas, que são usadas para causar o chaveamento do FF ou a mudança entre os estados de suas saídas.

Tipos de Flop-Flops

Os flip-flops podem ser classificados pelo tipo de gatilhamento: em Flip-Flops Síncronos ou com gatilhamento e Flip-Flops Assíncronos ou sem gatilhamento.

Nos Flip-Flops Síncronos, um pulso de gatilhamento, ou “clock”, é usado para determinar o instante da mudança de estado do FF ou da sincronização. Nos Flip-Flops Assíncronos, as transições ocorrem após terem sido completadas as mudanças nas entradas normais, sem a necessidade de pulso de sincronização.

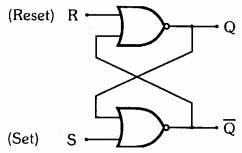
O gatilhamento pode ocorrer de duas formas: a) por nível (*Level-Triggered*) ou por transição (Edge Triggered). No FF gatilhável por nível, a informação é armazenada à partir de sua entrada enquanto a entrada de "clock" ou "trigger" estiver no estado ativo (ou de armazenamento), o qual pode se alto ou baixo. Durante o armazenamento os dados nas entradas não devem ser mudados até a mudança do estado ativo do “clock”.

Os flip-flops gatilháveis por transição (edge) podem ser: Edge Positivo e Edge Negativo. No FF Gatilhável por Edge Positivo a transferência da informação da entrada para a saída ocorre na subida do pulso de clock (edge positivo). No FF Gatilhável por Edge Negativo esta transferência ocorre na descida do pulso de clock (edge negativo).

Os FF podem ser ainda classificados segundo o comportamento ou função de suas entradas. Os mais comuns são: RS, T, D e JK, descritos a seguir.

RS Assíncrono

Este é o mais simples dispositivo de memória, também chamado de “Flip-Flop Set-Reset”. Este FF possui duas entradas denominadas Reset (R) e Set (S): A saída Q vai para baixo sempre que a entrada reset é ativada e para alto sempre que a entrada Set é ativada. Quando o nível da saída Q é igual a "1", a saída é dita “setada”; e quando "0", é dita “resetada”. Este FF pode ser implementado com portas NOR ou com portas NAND, como mostra a figura abaixo:



Flip-Flop RS Assíncrono

Devido à realimentação das saídas complementares Q e $\overline{Q}$ para as entradas das portas lógicas, só é possível conhecer os níveis lógicos das saídas num instante futuro (t+Δt), conhecendo-se os níveis lógicos das entradas R e S e das saídas Q e $\overline{Q}$ no instante atual (t). A tabela a seguir descreve o princípio de armazenamento do FF SR:

$$Q(t + \Delta t) = \overline{R(t) + \overline{Q}(t)}$$

$$\overline{Q}(t + \Delta t) = \overline{S(t) + Q(t)}$$

Entradas Atuais		Saídas Atuais		Saídas Futuras		Comentários	
R(t)	S(t)	$Q(t)$	$\overline{Q}(t)$	$Q(t+\Delta t)$	$\overline{Q}(t+\Delta t)$		
0	0	0	1	0	1	Saídas futuras = saídas atuais	
		1	0	1	0		
0	1	0	1	1	0	saída futura Q=1 independente de seu valor atual	Região de operação
		1	0	1	0		
1	0	0	1	0	1	saída futura Q=0 independente de seu valor atual	
		1	0	0	1		
1	1	0	1	0	0	erro lógico $Q(t+\Delta t) = \overline{Q}(t+\Delta t)$	
		1	0	0	0		

Tanto no circuito como nas expressões, devido à propagação dos sinais, as saídas $Q(t)$ e $\overline{Q}(t)$ são atualizadas constantemente até a estabilização das mesmas. As Figuras, a seguir, apresentam o símbolo lógico do FF RS e sua tabela-verdade simplificada, em função de suas variáveis de entrada R, S, onde Q_a diz respeito ao estado atual e Q_f ao estado futuro.

R	S	Q_f
0	0	Q_a
0	1	1
1	0	0
1	1	*

* Erro Lógico

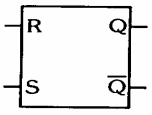


Tabela-verdade e símbolo lógico do Flip-Flop RS Assíncrono

Diagrama no Tempo

A figura a seguir apresenta os diagramas no tempo nos terminais de saída em função das entradas R e S para a condição inicial Q = 1. Os atrasos de propagação nas portas foram considerados iguais a 0. Tais diagramas refletem a tabela de combinações da figura anterior.

Observando o diagrama no tempo, verifica-se que quando na região normal de operação, a saída Q = 0 informa que o último pulso positivo ocorreu na entrada R.

Observa-se, ainda que antes de t0, e entre os instantes t1 e t2, as entradas R e S estão em nível 0. Contudo, a saída Q assume níveis lógicos diferentes, ou seja, para uma mesma entrada a saída pode ser 0 ou 1, dependendo da história das entradas.

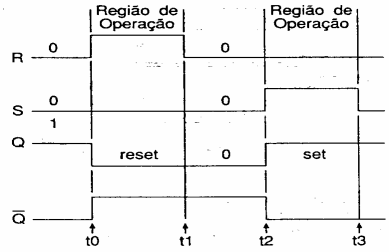
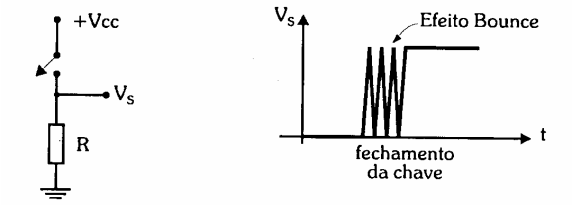


Diagrama no tempo do Flip-Flop RS

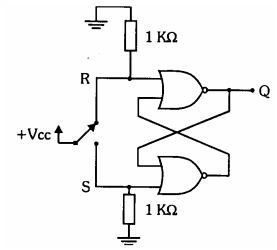
Exemplo de Aplicação: Eliminador de Ruído (Debouncing)

Muitas vezes, o acionamento ou o controle de sistemas digitais é feito através de dispositivos mecânicos que, devido às suas características físicas de construção, apresentam vibrações ao serem acionados, gerando um ruído denominado efeito *bounce*, que pode ser prejudicial ao funcionamento do sistema, como mostra a figura a seguir.



Por isso, muitos sistemas digitais precisam de circuitos eliminadores de ruídos (*debouncing*), como mostrado a seguir. O circuito *debouncing* é formado por um flip-flop RS cujas entradas estão ligadas ao terra através de resistores denominados *pull-down*.

A chave ligada ao Vcc ativa as entradas R ou S, levando a saída Q para 0 (chave na posição R) ou para 1 (chave na posição S). O ruído gerado pela vibração da chave é eliminado, pois, quando ela não está ligada a nenhuma das entradas, R e S ficam em nível lógico 0 devido aos resistores de pull-down, mantendo a saída Q inalterada, como mostram as figuras a seguir:



Circuito Eliminador de Ruído (debouncing)

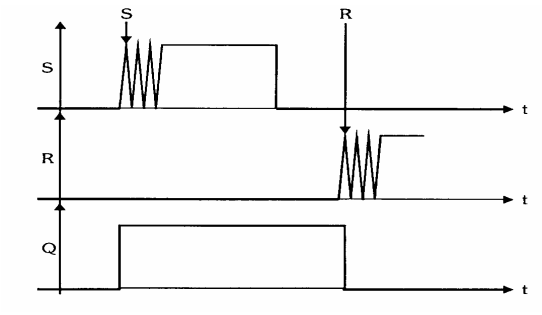
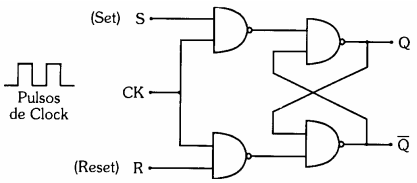


Gráfico de Saída do Eliminador de Ruído

Flip-Flop RS Síncrono

Este flip-flop apresenta, além das entradas reset (R) e set (S), uma terceira entrada denominada CK que, através de um sinal externo chamado pulso de clock (relógio), determina o instante de atualização das saídas Q e $\overline{Q}$, como mostra a figura abaixo.



Flip-Flop RS Síncrono

CK	R	S	Q_f
0	X	X	Q_a
1	0	0	Q_a
	0	1	1
	1	0	0
	1	1	*

* Erro Lógico

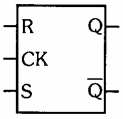
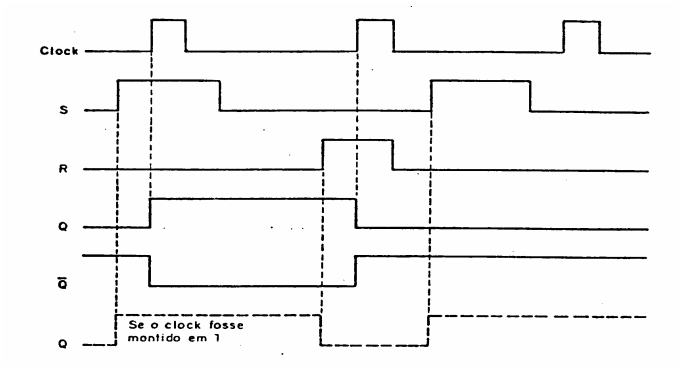


Tabela-Verdade e Símbolo Lógico do Flip-Flop RS Síncrono

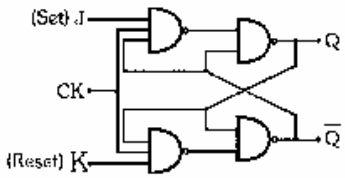
Neste circuito, o pulso de clock determina o instante da ação das entradas R e S. Neste sinal, os tempos dos níveis 0 e 1 devem ser maiores que o tempo de atraso das portas lógicas do circuito, para que a atualização das saídas seja completada. Neste circuito permanece o problema do erro lógico ($Q = \overline{Q} = 1$) ocorrido quando R=1 e S=1.



Forma de onda do funcionamento de um Flip-Flop RS Síncrono

Flip-Flop JK

O circuito abaixo representa um flip-flop JK que é uma variação do RS síncrono, no qual foi incluída uma nova realimentação das saídas Q e $\overline{Q}$ às portas lógicas de entrada.



Flip-Flop JK

Neste flip-flop, as entradas J e K executam, respectivamente, as funções set e reset. Seu funcionamento é similar ao do flip-flop RS síncrono com exceção da condição de entrada J=1 e K=1 na qual, logo que o pulso de clock muda de 0 para 1, as saídas Q e $\overline{Q}$ se complementam. Esta complementação das saídas e a realimentação às portas lógicas de entrada provocam sucessivas complementações (oscilação) enquanto o pulso de clock encontra-se em nível lógico 1.

CK	J	K	Q_f
0	X	X	Q_a
1	0	0	Q_a
	0	1	0
	1	0	1
	1	1	*

* - oscilação

Tabela-verdade do Flip-Flop JK (síncrono à nível)

Esta oscilação na condição J=1 e K=1 também não é desejável, pois, trata-se de uma instabilidade do circuito. A solução é utilizarmos flip flops sensíveis 'a transição:

A tabela-verdade deste flip-flop, bem como o seu símbolo lógico, estão mostrados a seguir.

CK	J	K	Q_f
0	X	X	Q_a
↓	0	0	Q_a
	0	1	0
	1	0	1
	1	1	$\overline{Q}_a$

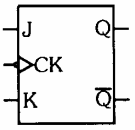
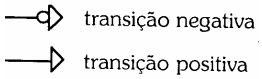


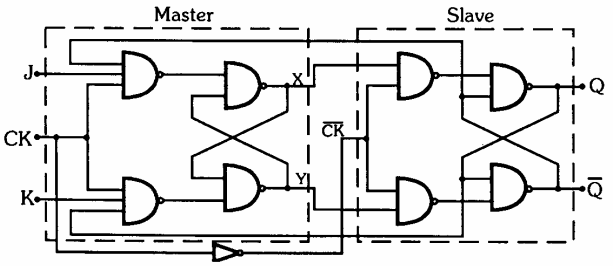
Tabela-Verdade e Símbolo Lógico do Flip-Flop JK (síncrono à transição)

Além de resolver o problema da oscilação, neste flip-flop as saídas se atualizam somente na descida do pulso de clock, sendo, por isso, chamado de sensível à borda de descida ou transição negativa. Para transformá-lo num flip-flop sensível à borda de subida ou transição positiva, basta acrescentar um inversor na entrada CK. Os símbolos utilizados para representar uma entrada de clock sensível às transições negativa e positiva são:



Flip-Flop JK Master-Slave (Mestre-Escravo)

O circuito da figura a seguir apresenta o flip-flop JK *master-slave* (mestre-escravo), o qual é formado por dois flip-flops RS síncronos ligados em cascata com um inversor entre a entrada de clock do primeiro (master ou mestre) e a entrada de clock do segundo (slave ou escravo), além de uma outra realimentação que vem das saídas Q e $\overline{Q}$ às portas lógicas de entrada.

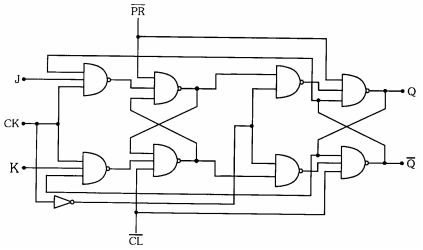


Flip-Flop JK Master-Slave

No flip-flop JK *master slave*, para J=1 e K=1, na subida do pulso de clock, X e Y complementam-se apenas uma vez e, na descida do pulso de clock, as saídas Q e $\overline{Q}$ complementam-se também apenas uma vez, pe+rmanecendo estáveis até que um novo pulso de clock completo (subida e descida) seja aplicado à entrada CK.

Flip-Flop JK Master Slave com Preset e Clear

Este FF JK apresenta duas entradas assíncronas muito úteis, preset (PR) e clear (CL). Estas entradas atuam diretamente nas saídas Q e $\overline{Q}$, portanto, independente do pulso de clock e do nível lógico das entradas J e K, como mostra a figura a seguir.



Flip-Flop JK Master-Slave com Preset e Clear

$\overline{PR}$	$\overline{CL}$	CK	J	K	Q_f
1	0	X	X	X	0
0	1	X	X	X	1
1	1	↓	0	0	Q_a
			0	1	0
			1	0	1
			1	1	$\overline{Q}_a$

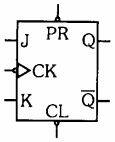
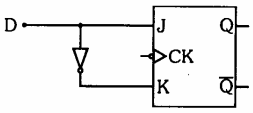


Tabela-Verdade e Símbolo Lógico do Flip-Flop JK com Preset e Clear

Neste circuito, as entradas $\overline{PR}$ e $\overline{CL}$ são ativas em 0 e têm a função de forçar a saída Q para 1 (preset ativo) ou para 0 (clear ativo). Com as entradas preset e clear desativadas ($\overline{PR} = 1$ e $\overline{CL} = 1$), o flip-flop funciona normalmente, com suas saídas dependendo de J, K e CK. Ë importante lembrar que nstes circuitos, as entradas preset e clear não podem ser ativas simultaneamente ($\overline{PR} = 0$ e $\overline{CL} = 0$), caso contrário, tem-se um novo erro lógico nas saídas.

Flip-Flops D e T

Os FFs D e T são FFs obtidos à partir do FF JK. A figura a seguir representa um flip-flop JK master-slave com um inversor entre suas entradas, formando um flip-flop D:



Flip-Flop D

Deste modo, tem-se $J = \overline{K}$, ou seja, se D = 0, então J = 0 e K = 1 (reset ativado) e, portanto, as saídas futuras do flip-flop serão $Q_f = 0$ e $\overline{Q}_f = 1$; se D = 1, então J = 1 e K = 0 (set ativado) e, portanto, as saídas futuras do flip-flop serão $Q_f = 1$ e $\overline{Q}_f = 0$. A figura a seguir mostra a tabela-verdade do flip-flop D, bem como o seu símbolo lógico.

CK	D	Q_f
↓	0	0
↓	1	1

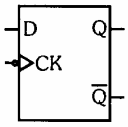
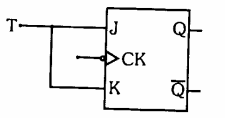


Tabela-Verdade e Símbolo Lógico do Flip-Flop D

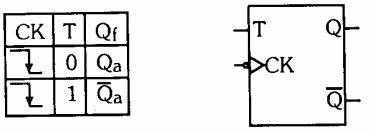
O FF D, pela sua simplicidade de operação, é largamente empregado como elemento de diversos tipos de registradores entre outros sistemas de memória, a serem estudados adiante. A sigla D vem de Data (dado), termo original em inglês.

A figura a seguir representa um flip-flop JK master-slave com as entradas J e K ligadas, formando o flip-flop T.



Flip-Flop T

Deste modo, tem-se $J = K$, ou seja: se T = 0, então J = 0 e K = 0 e, portanto, as saídas futuras do flip-flop permanecerão iguais às atuais ($Q_f = Q_a$ e $\overline{Q}_f = \overline{Q}_a$); se T = 1, então J = 1 e K=1, as saídas futuras do flip-flop serão o complemento das atuais ($Q_f = \overline{Q}_a$ e $\overline{Q}_f = Q_a$). A figura a seguir mostra a tabela-verdade do flip-flop T, bem como o seu símbolo lógico:

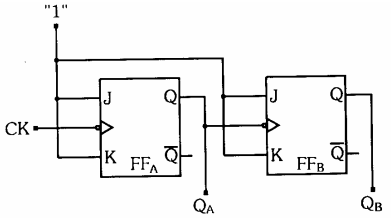


CK	T	Q_f
↓	0	Q_a
↓	1	$\overline{Q}_a$

Tabela-Verdade e Símbolo Lógico do Flip-Flop T

Exemplo de Aplicação – Divisor de Freqüência

O circuito a seguir apresenta dois flip-flops JK master-slave ligados em cascata, funcionando como um divisor de freqüência.



Divisor de freqüência

Estando os dois flip-flops com as entradas J e K em nível lógico 1, o primeiro (FF_A) complementa sua saída Q_A a cada transição negativa do pulso de clock e o segundo (FF_B) complementa sua saída Q_B a cada transição negativa da saída Q_A, como mostra o diagrama de tempo da figura a seguir.

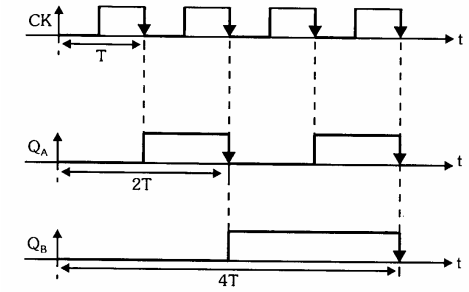


Diagrama de Tempos do Divisor de Freqüência

Através do diagrama pode-se observar a relação entre as freqüências dos sinais CK, Q_A e Q_B:

$$f_{QB} = \frac{f_{QA}}{2} = \frac{f_{CK}}{4}$$

Parâmetros Temporais dos Flip-Flops

Como em todos os circuitos digitais, faz-se importante conhecer as características temporais dos FFs. Entre estas destacam-se:

- Tempo *set-up* (ou de estabilização) – T_E é o intervalo de tempo mínimo (em ns) durante o qual as entradas (R, S, D, J ou K) não devem mudar antes da transição do *clock* ou *enable*;
- Tempo *Hold* (de sustentação ou de manutenção) - T_{HOLD} é o intervalo de tempo mínimo (em ns) durante o qual as entradas (R, S, D, J ou K) não devem mudar após a transição do *clock* ou *enable*;
- Largura de pulso (T_L): é a menor largura de pulso aceitável (mínima, em ns) para a entrada *clock*;
- Freqüência máxima de operação ($F_{máx}$) : é a maior freqüência dos pulsos de *clock* que pode ser aplicado ao dispositivo.

Registradores

Um registrador é um circuito compostos por vários flip-flops utilizados para armazenar um bit de informação cada. Arranjados em conjuntos de tamanhos diversos (dados pelo número de flip-flops existentes), estes circuitos possuem várias aplicações e tipos de entradas:

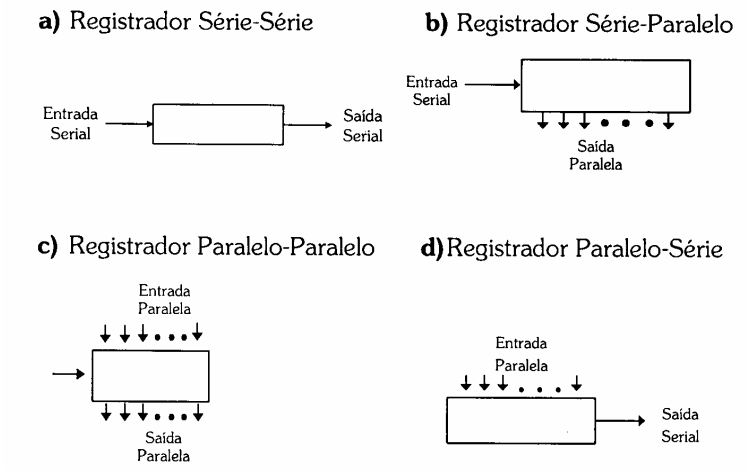
- suas entradas de controle (tais como preset, clear, CK, entre outras), geralmente, são comuns a todos os FFs internos;
- os registradores podem ser encontrados organizados em 4, 6 e 8 bits;
- além da função de armazenamento paralelo da informação binária há registradores que permitem operações de deslocamento e de rotação, úteis nas operações aritméticas e nos procedimentos de paralelização e de serialização da informação, existentes nos circuitos de comunicação digital.

Configurações Básicas

Os registradores podem ter diferentes configurações as quais dependem de como os dados são tratados:

- Serial: a informação é recebida ou transmitida bit a bit em uma única linha;
- Paralelo: os bits da informação são recebidos ou transmitidos, simultaneamente.

As figuras, a seguir, mostram as configurações básicas dos registradores.



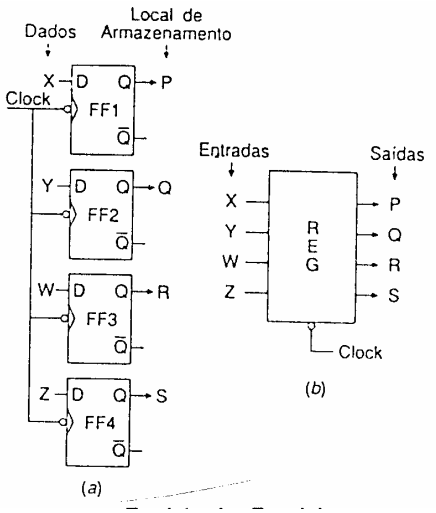
Configurações Básicas dos Registradores

O número de bits que pode ser armazenado num registrador depende do número de flip-flops que o compõe. O número de flip-flops envolvido em um registrador é, no mínimo, igual ao número de bits da informação a ser armazenada.

Nos registradores que utilizam entrada e/ou saída seriais, os dados movimentam-se internamente bit a bit a cada transição do clock. Estes circuitos são, portanto, chamados de registradores de deslocamento (*shift-register*s).

Registrador com Entrada Paralela

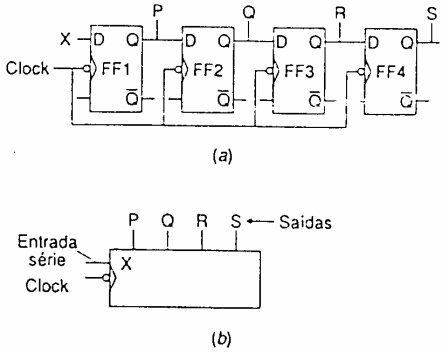
Este circuito permite o armazenamento simultâneo de vários bits de informação em uma única transição de clock. A Figura a seguir apresenta um exemplo constituído por quatro FFs. Quando ocorre a transição negativa na linha de *clock*, os níveis lógicos presentes nas entradas X, Y, W e Z são copiados para os FFs, onde ficam armazenados e presentes nas saídas P, Q, R e S, respectivamente.



Registrador Paralelo: a) Circuito; b) Símbolo

Registrador Série

O circuito registrador série é implementado ligando-se vários FFs em cascata, conforme apresenta a Figura abaixo: a saída do primeiro é ligada à entrada do segundo, a saída deste na entrada do terceiro, até o último deles. Neste registrador a informação binária é transferida flip-flop a flip-flop a cada transição do clock, realizando o armazenamento seqüencial da informação.



Registrador série: a) Circuito; b) Símbolo

Quando ocorre a transição negativa na linha *clock* da figura, o nível lógico presente em cada entrada é transferido para a entrada seguinte. O nível presente em S é sempre perdido a cada pulso de clock.

Dependendo da ordem adotada para os bits de saídas dos FFs, podem ser obtidos registradores de deslocamento à direita ou à esquerda. No exemplo dado, se o bit S é mais significativo do que P, então o registrador em questão realiza um deslocamento à esquerda. Caso P seja mais significativo, o deslocamento ocorre à direita.

Em resumo, o deslocamento pode ocorrer para a esquerda (*shift-left*), para a direita *(shift-right*) e bidirecional (*shift right/left*). Neste último caso, uma entrada de controle estabelece o funcionamento (como *right* ou *left*).

A Tabela a seguir apresenta o mecanismo de armazenamento e deslocamento da informação 1010 introduzindo-a pelo bit menos significativo, realizando, assim, um deslocamento à direita.

Entrada de *clock*	Nível Lógico em X, antes do pulso *clock*	Saídas P Q R S	Comentário
"0" ou "1"	*	* * * *	estados iniciais quaisquer (* = 1 ou 0)
"0" ou "1"	0	* * * *	
Transição 1 -> 0	0	0 * * *	P = X = 0
"0" ou "1"	1	0 * * *	X = 1 (Atualização)
Transição 1 -> 0	1	1 0 * *	Q = 0; P = X = 1
"0" ou "1"	0	1 0 * *	X = 0 (Atualização)
Transição 1 -> 0	0	0 1 0 *	R = 0; Q = 1; P = X = 0
"0" ou "1"	1	0 1 0 *	X = 1 (Atualização)
Transição 1 -> 0	1	1 0 1 0	P = 1; Q = 0; R = 1; S = 0

Classificação dos Registradores

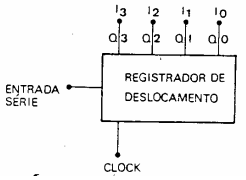
Os registradores podem, ainda, ser classificados pela forma de manipulação dos dados em registradores de entrada paralela/saída paralela (*parallel-in/parallel-out*); b) registradores de entrada série/saída paralela (*serial-in/parallel-out*); e registradores de entrada série/saída série (*serial-in/serial-out*).

Aplicação de Registradores

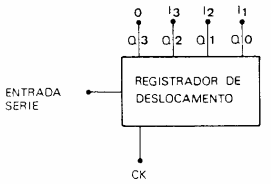
A seguir são apresentadas algumas das aplicações mais comuns de registradores.

Divisor por 2, 4, 8 ...

Se Inserido um zero à esquerda de um número, deslocando os demais para a direita divide-se este número por 2. Se inserido um segundo 0 e deslocado mais um bit à direita a divisão dada será por 4, e assim por adiante. As Figuras a seguir mostram estas operações na saída:



No. Presente nas saídas O_3_O_0



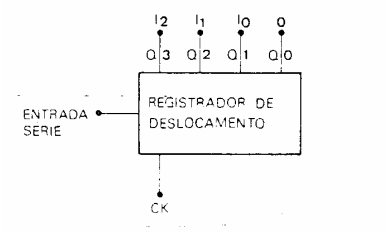
No. Anterior Dividido por 2 Presente nas saídas O_3_O_0

Exemplo:

Seja I = 1110 (14_{10})
Registrador: Q_3 = 1, Q_2 = 1, Q_1=1 e Q_0 = 0
Quando realizado o deslocamento à direita com a inserção de "0" no bit mais significativo obtêm-se na saída Q_3 = 0, Q_2 = 1, Q_1=1 e Q_0 = 1 ou I = 0111 (7_{10}) , ou o número anterior dividido por 2.

Multiplicador por 2, 4, 8 ...

Se Inserido um zero à esquerda de um número, deslocando os demais para a esquerda multiplica-se este número por 2. Se inserido um segundo 0 e deslocado mais um bit à esquerda a multiplicação dada será por 4, e assim por adiante. As Figuras a seguir mostram estas operações na saída:

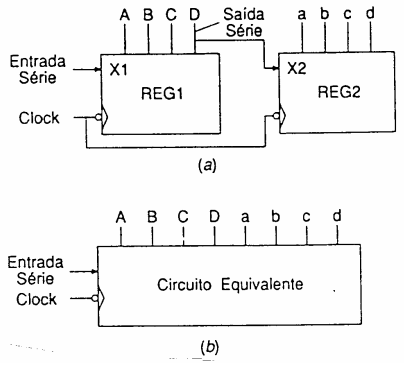


Exemplo:
Seja I = 0101 (5_{10})
Registrador: $Q_3 = 0$, $Q_2 = 1$, $Q_1=0$ e $Q_0 = 1$
Quando realizado o deslocamento à esquerda com a inserção de "0" no bit menos significativo, obtêm-se na saída $Q_3 = 1$, $Q_2 = 0$, $Q_1=1$ e $Q_0 = 0$ ou I = 1010 (10_{10}) , ou o número anterior multiplicado por 2.

"Cascateamento" e Paralelismo

Interligando registradores é possível obter um registrador com maior capacidade. Dois registradores de 4 bits formam um de 8 bits de duas formas: ligando-os em série para um *shift register* de 8 bits, ou um registrador paralelo de 8 bits como apresenta as figuras abaixo:

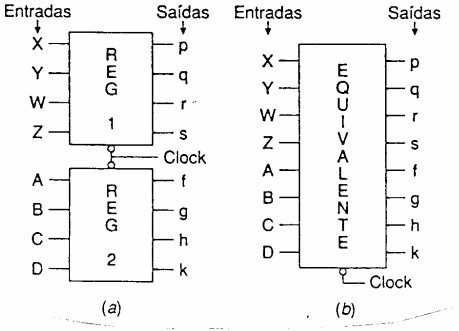
Na figura a seguir dois registradores série foram cascateados para a obtenção de um circuito equivalente a um registrador série com 8 bits.



"Cascateamento" de Registradores de Deslocamento.

Na Figura acima, a saída série do registrador REG1 foi conectada à entrada série do registrador REG2 e foram interligados os *clocks* dos dois registradores. A entrada série do circuito equivalente coincide com a entrada série do REG1. A sua saída série, com a saída série de REG2.

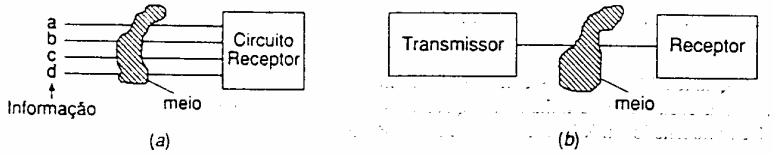
Na figura, a seguir, dois registradores paralelos foram colocados lado a lado para a obtenção de um circuito equivalente a um registrador paralelo com 8 bits. As entradas similares de controle, set, reset a entrada *clock* são, respectivamente, interligadas. As entradas e saídas de dados fornecem as entradas e saídas do circuito equivalente.



Paralelismo de Registradores.

Conversão Série-Paralelo e Paralelo-Série

A conversão de dados entre as formas série e paralelo é muito comum quando a transmissão da informação em uso é do tipo serial, o que não é necessário quando a mesma é do tipo paralela, pois se dá no formato em que os dados são armazenados ou processados.

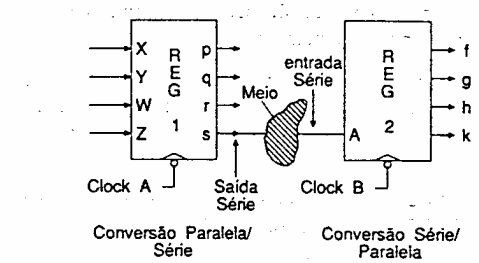


(a)Transmissão paralela; (b)Transmissão série.

A transmissão da informação, então, pode ocorrer com diferentes custos que, geralmente, são determinados pelo tipo de comunicação, pelas taxas (ou velocidades), pelos meios e pela tecnologia empregada.

Na comunicação bit a bit ou serial, um bit de informação trafega por vez no meio de comunicação. Durante a transmissão, um registrador de deslocamento é utilizado, realizando a carga (*load*) ou armazenamento em paralelo da informação e a transmitindo (*shift*) a cada pulso de clock, bit a bit.

Na Figura a seguir, a informação presente nas entradas XYWZ, é armazenada em REG1 após um pulso na linha *clock* A. Neste momento, o REG1 realiza a carga paralela (operação de *load*). Em seguida, a informação armazenada é transmitida para REG2, bit a bit. REG1 realiza uma conversão paralelo-série e REG2, uma conversão série-paralela).



Conversão Paralela-Série e Série-Paralela.

Registradores Integrados

Na família TTL existem registradores na forma de CI's, tais como:

- 74164, que é um registrador de deslocamento entrada série, saída paralelo;
- 74165, que é um registrador de deslocamento, entrada paralela, saída série;
- 74194, que é um registrador de deslocamento universal, no qual os dados podem entrar em série ou paralelo e sair em paralelo ou série;
- entre outros.

Contadores

O contador é um sistema seqüencial que fornece em suas saídas um conjunto de níveis lógicos numa seqüência predeterminada.

A este conjunto de níveis lógicos dá-se o nome de estados internos do contador.

O contador é formado basicamente por flip-flops e, portanto, a velocidade da seqüência gerada é determinada pela freqüência dos pulsos de clock.

Os contadores são utilizados principalmente para contagens, geradores de palavras, divisores de freqüência, medidas de freqüência e tempo, geradores de forma de onda, conversão analógico/digital, seqüenciamento de operações de máquinas, etc.

Configurações Básicas

Os contadores podem ser classificados segundo alguns critérios:

Tipo de controle:
- Assíncrono
- Síncrono

Tipo de contagem:
- Crescente (up)
- Decrescente (down)

Tipo de código:
- Hexadecimal
- Decimal (Década)
- Outros

Circuitos contadores são aqueles cujas saídas, para uma condição inicial 0, assumem uma combinação binária cujo equivalente decimal é igual ao número de pulsos recebidos na entrada *clock*.

O diagrama de blocos de um contador está apresentado na figura a seguir.

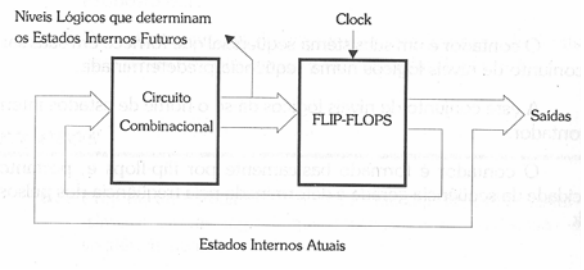


Diagrama de blocos de um contador genérico

Neste circuito a realimentação das saídas do circuito combinacional paras suas entradas passa por flip-flops que funcionam controlados por pulsos de clock. Desta forma, as saídas do contador fornecem os estados internos atuais e as saídas do circuito combinacional fornecem níveis lógicos que, atuando nas entradas dos flip-flops, determinam os estados internos futuros.

A seqüência das combinações binárias assumidas pelas saídas do contador em função dos pulsos na linha *clock* pode ser natural ou não. Por exemplo, podemos projetar um contador para assumir uma seqüência de estados que corresponde a decimais consecutivos crescentes, (0, 1, 2, 3, 4, ...) ou decrescentes (..., 4, 3, 2, 1, 0), ou ainda, pode assumir uma seqüência não natural (3, 6, 4, 8, 2, 5, 1).

Obs.: Salvo especificação em contrário, consideraremos sempre a seqüência natural.

Contadores Assíncronos

Um contador assíncrono é aquele no qual os flip-flops são controlados por pulsos de clock não simultâneos. A figura a seguir mostra um diagrama de blocos deste tipo de montagem.

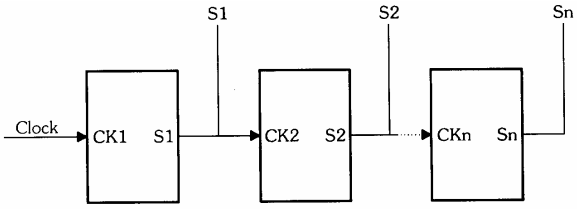


Diagrama de blocos de um contador assíncrono genérico

A entrada *clock* do contador corresponde à entrada *clock* do flip-flop mais à esquerda. A saída deste flip-flop corresponde ao bit menos significativo do contador.

Contador Assíncrono Crescente

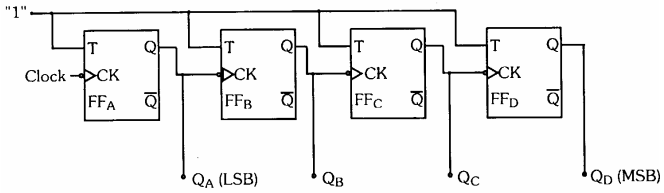
O projeto desse contador assíncrono crescente é muito simples: as saídas e entradas são, respectivamente, ligadas e polarizadas conforme a rotina:

A saída de cada flip-flop deve ser conectada a entrada clock dos flip-flops seguintes;

As entradas T de todos os flip-flops (pode utilizar o flip-flop JK) devem ser polarizadas com nível 1;

sinal clock do contador deve acionar a entrada clock do primeiro flip-flop.

O circuito do contador hexadecimal assíncrono crescente pode ser obtido da forma apresentada na figura a seguir.



Contador Hexadecimal Assíncrono Crescente

As saídas do primeiro e último flip-flop correspondem, respectivamente, aos bits menos e mais significativos do contador.

Para análise do funcionamento deste circuito, é preciso destacar duas características do flip-flop T:

Atua na transição negativa (borda de descida do clock);

Para T = 1, $Q_f = \overline{Q_a}$.

Como este contador é assíncrono, os pulsos de clock externos atuam apenas no flip-flop A complementando sua saída Q_A a cada transição negativa. Mas Q_A é quem fornece o sinal de clock para o flip-flop B, e portanto, quando esta saída corresponder à passagem de nível lógico 1 para 0 (transição negativa), a saída Q_B é complementada, e assim sucessivamente, ou seja, Q_B fornece o sinal de clock para o flip-flop C que complementa Q_C que fornece o sinal de clock para o flip-flop T que complementa Q_D.

As figuras a seguir mostram o diagrama de tempos e a tabela de combinações deste contador partindo do estado inicial $Q_DQ_CQ_BQ_A$ = 0 0 0 0.

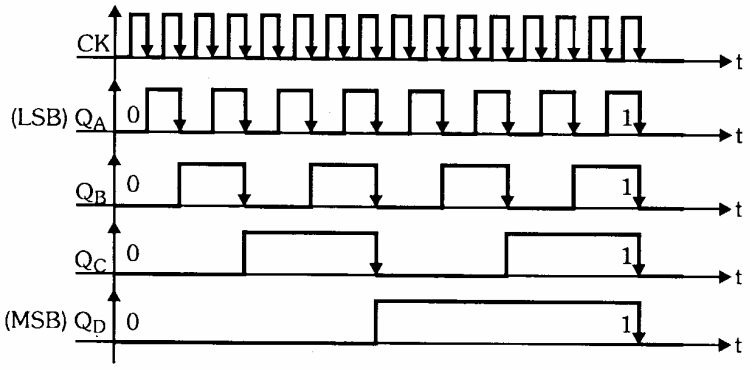


Diagrama de Tempos do Contador Assíncrono Hexadecimal Crescente

Número de pulsos na Entrada clock	Saídas $Q_DQ_CQ_BQ_A$
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

← Estado inicial

Contador Assíncrono Hexadecimal Crescente: Tabela de combinações

Considerando-se a saída Q_A como o bit menos significativo (LSB) e a saída Q_D como o bit mais significativo (MSB), verifica-se que este circuito gera em seqüência o código 0000 até terminar em 1111, recomeçando em seguida em 0000 enquanto houver pulsos de clock externos.

Por isto, este cicuito é denominado contador hexadecimal assíncrono crescente.

Um fato bastante relevante pode ser observado através do diagrama de tempos que mostra, também, que este contador divide a freqüência de clock por dois (Q_A), por quatro (Q_B), por oito (Q_C) e por dezesseis (Q_D). Assim, os estágios do contador dividem a freqüência de entrada *f*, respectivamente, por 2, 4, 8, e 16. Desta forma um sinal com freqüência igual a $f/2^n$ pode ser obtido na saída do estágio mais significativo de um contador assíncrono com n bits.

Para representar a seqüência de estados gerada por um contador, costuma-se utilizar o diagrama de estados como mostrado na figura a seguir, referente ao circuito analisado.

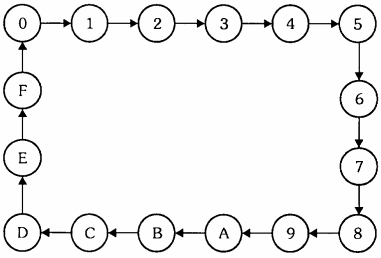


Diagrama de Estados do Contador Hexadecimal Crescente

Após 16 pulsos, o contador retornará ao estado 0000. Na verdade, após um número de pulsos múltiplos de 16, o contador assume o estado 0000. Ou seja, o estado 0000 não implica necessariamente que nenhum pulso acessou a entrada clock.

Analogamente, após um número de pulsos múltiplo de 17 o contador assume o estado 0001. Ou seja, o estado 0001 não implica necessariamente a chegada de um pulso na entrada clock.

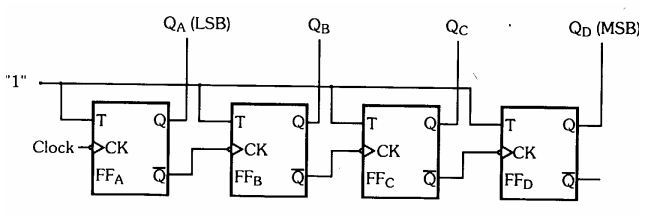
O número (n) de saídas do contador deve ser tal que a máxima combinação binária (2^n) corresponda, pelo menos, ao número máximo de pulsos previstos na entrada clock. Por exemplo, se um contador deve receber e registrar corretamente 30 pulsos, o valor de n deve ser, no mínimo, igual a 5. Isto porque o máximo estado corresponde à combinação 11111 (equivalente ao decimal 31).

Generalizando, o número máximo de pulsos que um contador pode registrar corretamente é dado por (2^n –1), em que *n* é igual ao número de flip-flops do contador.

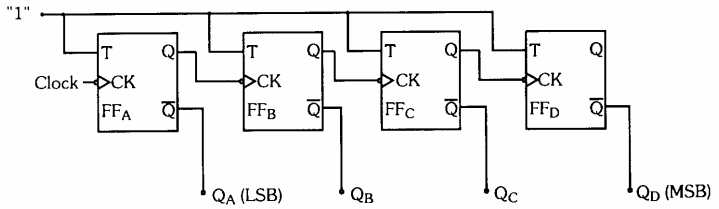
Contador Assíncrono Decrescente

Um contador assíncrono decrescente pode ser obtido de dois arranjos diferentes:

a) Modificando-se o arranjo mostrado na figura anterior de forma tal que as entradas clock dos estágios mais significativos sejam obtidos nas saídas Q barradas e as saídas do contador sejam tiradas das saídas Q dos flip-flops, conforme figura a seguir.



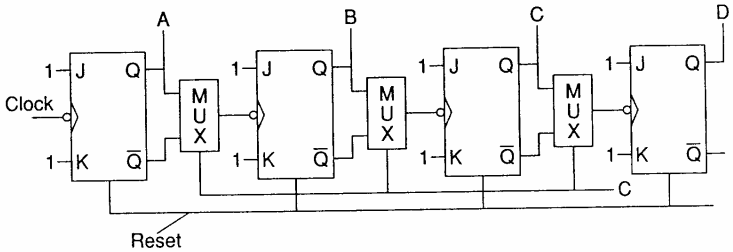
b) Organizando os flip-flops de forma similar ao da figura inicial contudo, considerando-se como saídas do contador, as saídas Q barradas dos flip-flops, conforme mostra a figura a seguir.



Contador Assíncrono Reversível (up/down)

Uma superposição dos contadores crescente e decrescente fornece o contador da figura abaixo (implementado, por exemplo, com flip-flop JK). Este contador pode assumir a seqüência decrescente ou crescente e, por esta razão, é chamado contador bidirecional.

Uma entrada de controle (C) é responsável pela conversão da forma de contagem crescente (up) para decrescente (down), e vice-versa.



Contador assíncrono reversível/4 bits.

Interliga-se circuitos multiplexadores entre os vários estágios do contador que selecionam para a linha *clock* do próximo estágio a saída do flip-flop ou o seu complemento.

Se a entrada (C) de seleção do multiplexador for igual a 0, (seleção dos canais 0's) os sinais clock dos flip-flops são obtidos, respectivamente, das saídas A, B e C. Neste caso a seqüência de contagem é crescente.

Se a entrada (C) de seleção do multiplexador for igual a 1 (seleção dos canais 1's) os sinais *clock* dos flip-flops são obtidos, respectivamente, dos complementos das saídas A, B e C. Neste caso a seqüência de contagem é decrescente.

Contadores Síncronos

São aqueles cujas entradas clock de todos os flip-flops do contador são interligadas ao sinal externo clock. Desta forma, opera sincronizado com a linha externa clock e com maior velocidade do que os equivalentes assíncronos.

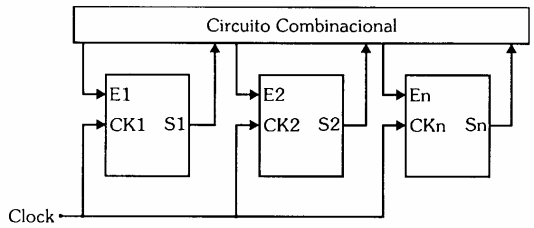


Diagrama de blocos de um contador síncrono genérico

O projeto do contador síncrono pode ser obtido, por exemplo, com flip-flop D ou flip-flop JK. Nesta disciplina será abordado apenas o projeto de contador síncrono com flip-flop D.

Projeto de um Contador Síncrono com flip-flop D

O projeto de contadores síncronos não é tão simples como os dos contadores assíncronos. No caso dos síncronos as entradas dos flip-flops não podem ter polarização fixa, mas devem ser convenientemente modificadas após cada pulso na linha clock

Exemplo de Aplicação: Contador Módulo 5 Síncrono Crescente

Como se trata de um contador módulo 5 (cinco estados), três flip-flops são suficientes para o projeto, já que a seqüência é formada apenas pelos estados 0 a 4 (000 a 100).

A figura abaixo mostra o diagrama de estados deste contador, que, como se pode notar, não apresenta em sua malha principal todos os estados possíveis (estão faltando os estados 5, 6 e 7).

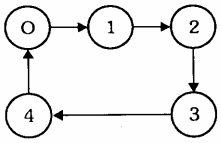


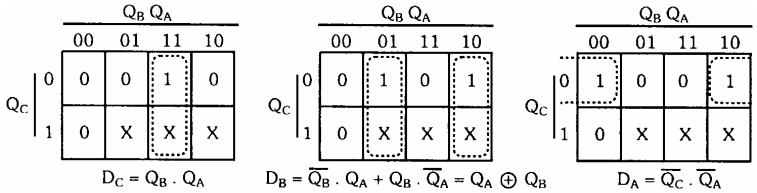
Diagrama de Estados do Contador Módulo 5 Síncrono Crescente

O projeto deste contador será desenvolvido utilizando flip-flop D. Levando-se em conta a característica do flip-flop D (Q_f =0 para D=0 e Q_f=1 para D=1), pode-se construir a tabela-verdade do circuito combinacional deste contador, como mostra a figura a seguir.

		Entradas do Circuito Combinacional						Saídas do Circuito Combinacional		
Estado Atual	Estado Futuro	Saídas Atuais dos Flip-Flops			Saídas Futuras dos Flip-Flops			Entradas dos Flip-Flops		
		Q_C	Q_B	Q_A	Q_C	Q_B	Q_A	D_C	D_B	D_A
0	1	0	0	0	0	0	1	0	0	1
1	2	0	0	1	0	1	0	0	1	0
2	3	0	1	0	0	1	1	0	1	1
3	4	0	1	1	1	0	0	1	0	0
4	0	1	0	0	0	0	0	0	0	0
5	X	1	0	1	X	X	X	X	X	X
6	X	1	1	0	X	X	X	X	X	X
7	X	1	1	1	X	X	X	X	X	X

Tabela-Verdade do Contador Módulo 5 Síncrono Crescente (Flip-Flop D).

Desta tabela-verdade pode-se obter as expressões lógicas das saídas do circuito combinacional através de mapas de Karnaugh.



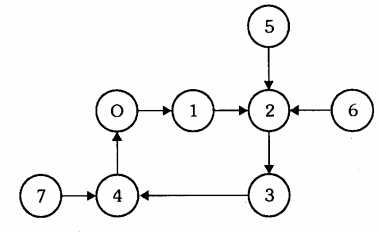
Para implementação do circuito, deve-se fazer a análise dos estados secundários (5, 6 e 7) para saber como se comportam em relação aos estados da malha principal. Para isso, é considerado o fato dos níveis lógicos irrelevantes terem sido definidos durante a resolução dos mapas de Karnaugh, pois, se eles fizeram parte do enlace, passaram a valer 1, caso contrário a valer 0.

Pela substituição das variáveis Q_C, Q_B e Q_A obtidas por seus valores correspondentes nos estados secundários 5, 6 e 7, obtém-se a tabela-verdade destes estados, descobrindo-se, assim, os estados futuros, como mostra a figura a seguir.

Estado Atual	Saídas Atuais dos Flip-Flops			Entradas dos Flip-Flops			Saídas Futuras dos Flip-Flops			Estado Futuro
	Q_C	Q_B	Q_A	D_C	D_B	D_A	Q_C	Q_B	Q_A	
5	1	0	1	0	1	0	0	1	0	2
6	1	1	0	0	1	0	0	1	0	2
7	1	1	1	1	0	0	1	0	0	4

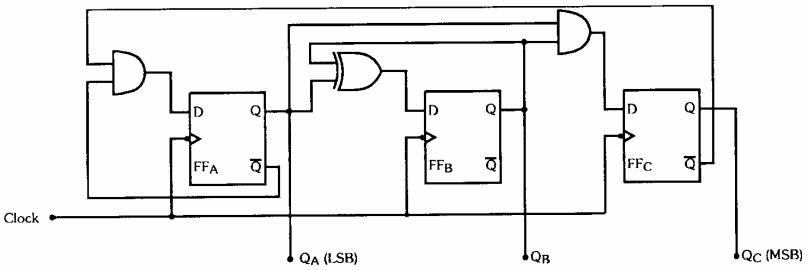
Tabela-Verdade dos Estados Secundários do Contador Módulo 5 Síncrono Crescente (Flip-Flop D).

O diagrama de estados completo do contador é dado por:



Obs.: Verifica-se pelo diagrama que é necessário um pulso de clock para que o contador volte à malha principal, independente de em qual estado secundário ele venha a cair.

A figura a seguir mostra o circuito do contador módulo 5 síncrono.

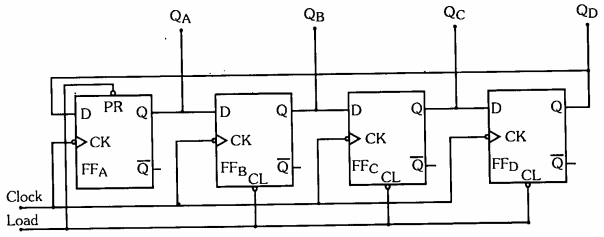


Contador Módulo 5 Síncrono Crescente (Flip-Flop D)

Obs.: Podemos construir contadores que geram uma seqüência qualquer. Para isso é necessário estabelecermos a seqüência e seguirmos o método descrito anteriormente.

Exemplo de Aplicação: Contador em Anel

Este circuito pode ser construído a partir de um registrador de deslocamento no qual a saída serial é realimentada à entrada serial. A figura, a seguir, mostra um contador em anel de 4 bits com uma única saída em nível lógico 1.



Circuito do Contador em Anel

A entrada de controle load é utilizada para inicializar o contador através de um nível lógico 0, carregando as saídas com $Q_AQ_BQ_CQ_D$ = 1000. Em seguida, esta entrada deve permanecer em nível lógico 1, fazendo com que os pulsos de clock desloquem as saídas da esquerda para a direita com exceção da saída Q_D que é realimentada para o primeiro flip-flop, gerando a seqüência mostrada a seguir.

Q_A	Q_B	Q_C	Q_D
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Seqüência de saída do Contador em Anel

Este circuito tem grande aplicação quando se deseja o acionamento seqüencial de máquinas em uma fábrica (evitando, assim, um pico de corrente elevado), de lâmpadas coloridas (para efeitos luminosos), acionar circuitos de alarmes ou de outros sistemas.

Obs.: Pelo que foi estudado, conclui-se que existem duas vantagens principais do contador síncrono em relação ao assíncrono, a saber:

Velocidade – A freqüência máxima de clock do contador síncrono é limitada apenas pela freqüência máxima de clock de um flip-flop, pois não há propagação de atraso;

Versatilidade – É possível o projeto de um contador síncrono de qualquer seqüência prevendo, inclusive, a eliminação de problemas causados pelos estados secundários.

Ligação de Contadores em Cascata

Pode-se acoplar contadores a fim de se obter um contador de módulo maior. Se dois contadores de módulos M_1 e M_2 forem acoplados corretamente pode-se obter um contador de módulo $M = M_1 \times M_2$.

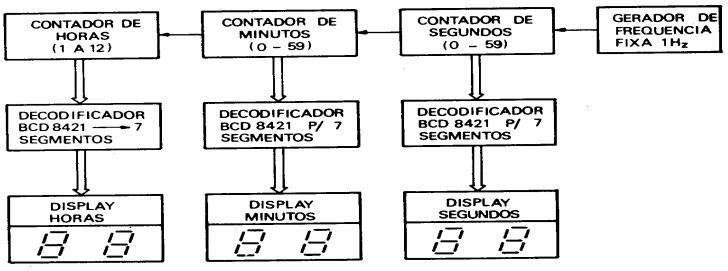
Devido ao uso generalizado dos circuitos contadores, os fabricantes desenvolveram pastilhas de circuitos integrados MSI com 14 e 16 pinos, contendo no seu interior contadores síncronos ou assíncronos com 4 ou mais bits.

Contadores Integrados mais comuns

7490	contador de “décadas”
7492	divisor por 12
7493	Divisor por 16
74193	Contador up/down

Com os elementos vistos nesta disciplina, podemos esquematizar o diagrama em blocos de um relógio digital básico. Esse é visto abaixo.

Analisando esse diagrama de blocos, nota-se que a cada pulso do gerador de freqüência o contador de segundos apresentará sua contagem num display de 7 segmentos, gerando também um pulso de clock para o contador de minutos, que também apresentará sua contagem num display de minutos e este contador por sua vez gerará um pulso de clock para o contador de horas, e assim poderemos ver no display geral a contagem que representará as horas, os minutos e os segundos.



12. Bibliografia

IDOETA, IVÃ VALEIJE e CAPUNO, FRANCISCO GABRIEL:
Elementos de eletrônica digital, 23ª edição revisada, atualizada e aplicada,
São Paulo: ÉRICA, 1984.

AZEVEDO JÚNIOR, JOÃO BATISTA DE:
TTL/CMOS: Teoria e aplicação em circuitos digitais, 2ª edição, volume 2
São Paulo: ÉRICA,1984.

BARTEE, THOMAS C.:
Fundamentos de computadores digitais, 4ª edição
Rio de Janeiro: Guanabara Koogam S.A.

Anexo 1 – Famílias Lógicas

Famílias lógicas

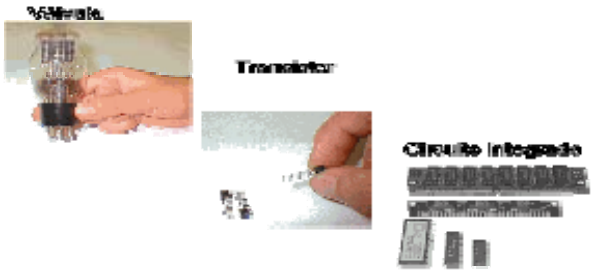
- Quando desejamos desenvolver um novo produto devemos observar suas características técnicas, as quais nos ajudam a decidir qual melhor tecnologia a ser usada, que satisfaça tais exigências, como:
  - Potência
  - Velocidade
  - Temperatura
  - .....
- Assim, deveríamos a princípio observar várias *tecnologias*, ou *famílias lógicas,* antes de decidirmos qual a melhor para a implementação de nosso projetos.

Mas, o que vem a ser uma família lógica?

Famílias lógicas

- **Família Lógica**
  - É um conjunto de circuitos digitais que implementam funções lógicas e que possuem características próprias de funcionamento, quanto a velocidade de chaveamento, alimentação (tensão e correntes elétricas) e temperatura.
- **Tipos de Famílias lógicas**
  - Transistor-Transistor Logic (TTL),
  - Emitter-Coupled Logic (ECL).
  - MOS - Metal- Oxide Semiconductor logic
  - Complementary Metal- Oxide Semiconductor logic (CMOS)

Evolução da Microeletrônica



- Os CHIPS podem ser divididos em várias categorias, dependendo da quantidade de transistores que existem em seu interior:
- SSI (Small Scale of Integration) - Contém em seu interior apenas algumas dezenas de transistores.
- MSI (Medium Scale of Integration) - Contém algumas dezenas de transistores.
- LSI (Large Scale of Integration) - Contém em seu interior, alguns milhares de transistores.
- VLSI (Very Large Scale of Integration) - Contém da ordem de dezenas de milhares a milhões de transistores.

De que são feitos os CHIPS ?

Princípios de Eletrônica

O Diodo na Comutação

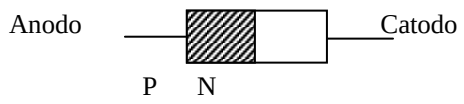


Figura 1 – O diodo "PN"

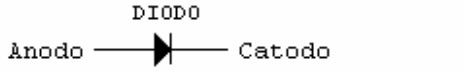


Figura 2 - O símbolo do diodo

O

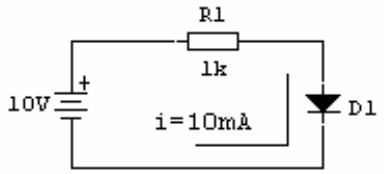


Figura 3 - diodo conduzindo

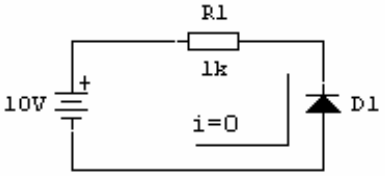


Figura 4 - diodo cortado

Transistor na Comutação

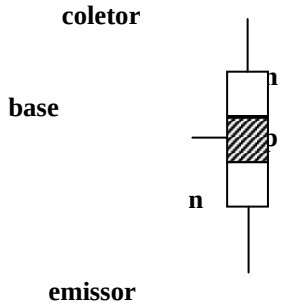


Figura 5 – o sanduíche "NPN"

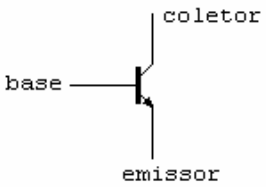


Figura 6 - símbolo de um transistor NPN

transistor cortado → chave aberta transistor saturado → chave fechada

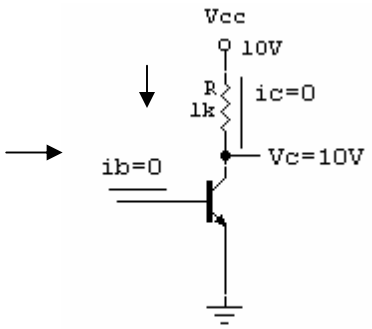


Figura 7 - Transistor Cortado

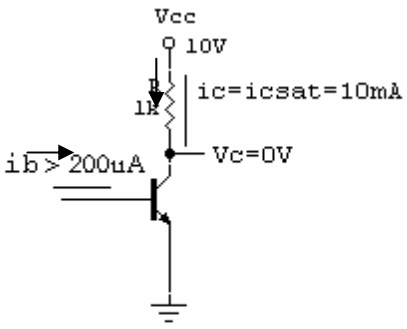


Figura 8 - Transistor Saturado

Conceitos e Parâmetros das Famílias Lógicas

Níveis de Tensão e de Corrente

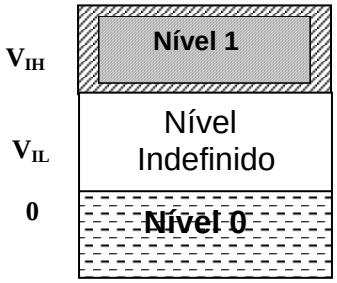


Figura 10 (a) – Entrada

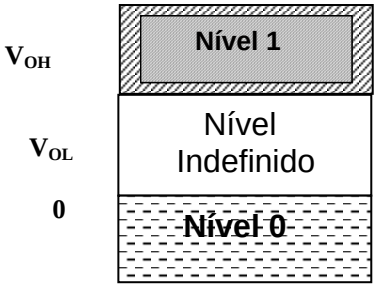


Figura 10 (b) – Saída

V_{IL} (Low–level Input Voltage) → Valor de Tensão (máxima) que garante o nível 0 na entrada;

V_{OL} (Low–level Output Voltage) → Valor de Tensão (máxima) que garante o nível 0 na saída;

V_{IH} (High–level Input Voltage) → Valor de Tensão (mínima) que garante o nível 1 na entrada;

V_{OH} (High–level Output Voltage) → Valor de Tensão (mínima) que garante o nível 1 na saída;

I_{IL} (Low–level Input Current) → Valor de corrente (máxima) no terminal de entrada (no sentido do bloco para o terminal), quando é aplicado o nível 0;

I_{OL} (Low–level Output Current) →Valor de corrente (máxima) que a saída pode receber quando em nível 1;

I_{IH} (High–level Input Current) → Valor de corrente de entrada (máxima) quando é aplicado o nível 1;

I_{OH} (High–level Output Current) → Valor de corrente de saída (máxima) quando em nível 1;

Nos manuais, além dos limites de mínimo e máximo, conforme a definição do parâmetro, são encontrados os valores típicos de trabalho.

Fan-Out : Número máximo de blocos lógicos que pode ser ligado à saída de outro da mesma família e/ou versões compatíveis.

$$\text{Fan-Out}_{(\text{nível } 0)} = \frac{\mathbf{I_{OL}}}{\mathbf{I_{IL}}} \quad \text{e} \quad \text{Fan-Out}_{(\text{nível } 1)} = \frac{\mathbf{I_{OH}}}{\mathbf{I_{IH}}}$$

Tempo de Atraso e Propagação: O tempo que um bloco lógico leva para mudar de estado desde a aplicação de um nível lógico para tanto, ou seja, é o tempo de resposta para que um bloco lógico possa passar de 0 para 1 (**t_{PLH}**), ou de 1 para 0 (**t_{PHL}**). Valor da ordem de nano segundos;

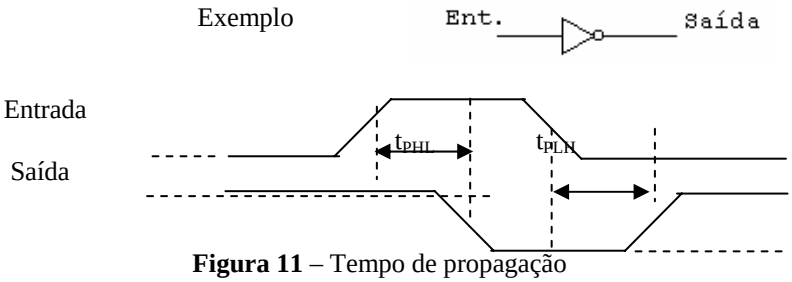
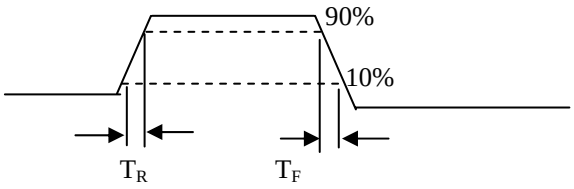


Figura 11 – Tempo de propagação

- tempo de subida (rise time – T_R);
- tempo de descida (fall time – T_F);



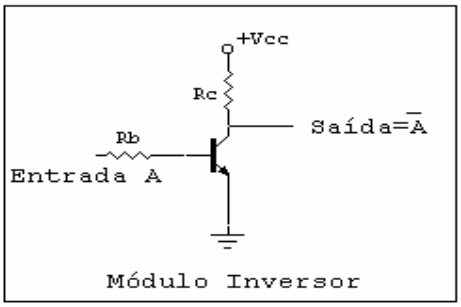
Imunidade ao Ruído. A capacidade que um bloco lógico de uma determinada família possui de não receber influências parasitas elétricas ou magnéticas, denominadas ruído, típicas dentro dos sistemas eletrônicos ou sob determinadas condições do ambiente em que estão situados;

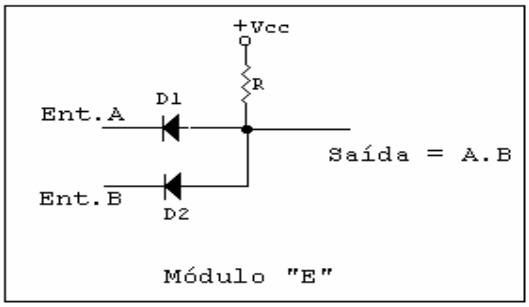
Margem de ruído : o quanto de tolerância irá haver sobre os limites dos níveis lógicos, sem que haja alteração na sua funcionalidade.

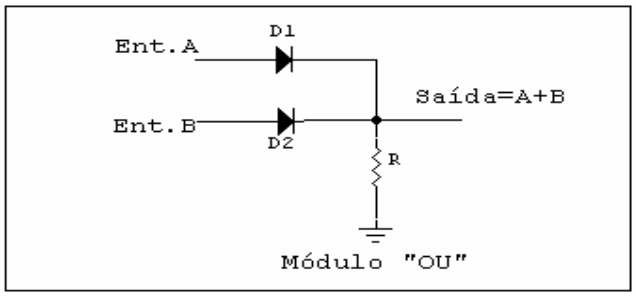
As Principais Famílias Lógicas

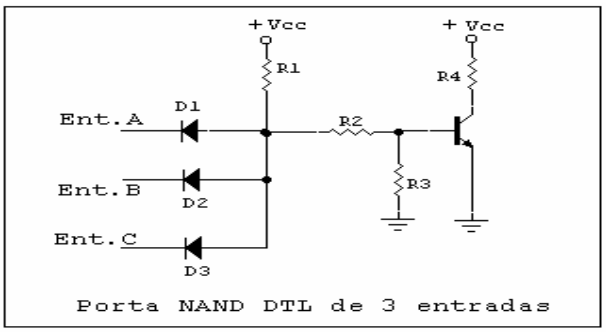
DTL – Diode Transistor Logic, uma evolução da RTL – Resistor Transistor Logic

Os módulos básicos DTL:

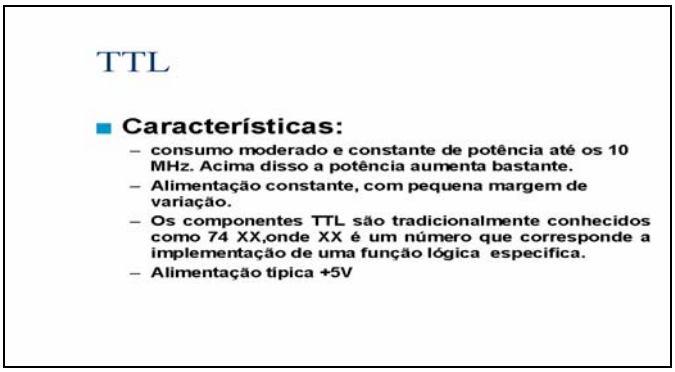




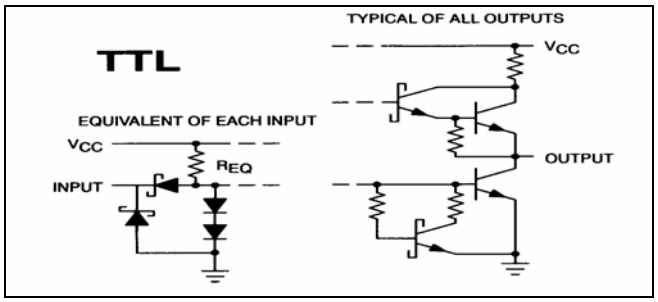


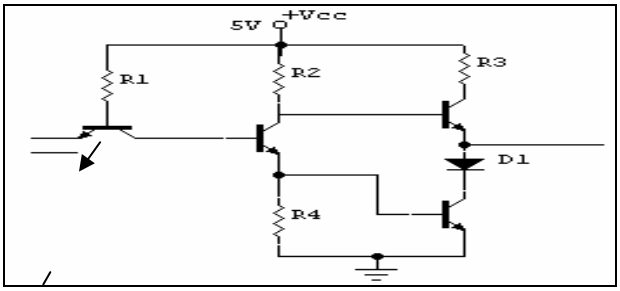


A Família TTL – Lógica Transistor –Transistor



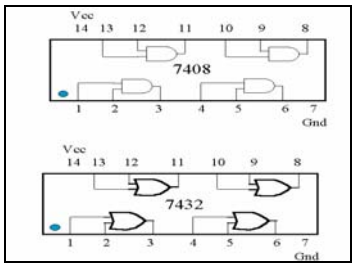
Transistores bipolares multi-emissores



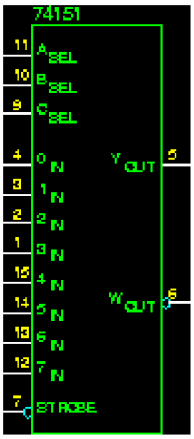


Exemplo: Uma porta NAND TTL de 2 entradas com Active Pull-Up e Totem-pole.

Exemplo de Circuitos
- 7400 - 4 portas NAND de 2-Entradas
- 7406 - 6 inversores (Open Collector)
- 7408 - 4 portas AND de 2-entradas
- 7432 - 4 portas OR de 2 - entradas
- 7486 - 4 portas XOR de 2 - entradas

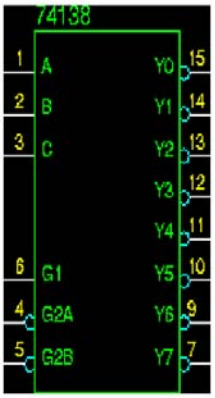


74151 - Multiplexador 8:1



Select inputs C B A	strobe S	output Y	W
X X X	H	L	H
L L L	L	D0	$\overline{D0}$
L L H	L	D1	$\overline{D1}$
L H L	L	D2	$\overline{D2}$
L H H	L	D3	$\overline{D3}$
H L L	L	D4	$\overline{D4}$
H L H	L	D5	$\overline{D5}$
H H L	L	D6	$\overline{D6}$
H H H	L	D7	$\overline{D7}$

74138 -Decodificador/Demultiplexador 3:8



Inputs Enable G1	G2	Select C	B	A	Outputs Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

A grande maioria dos circuitos integrados TTL pertencem às séries 54 e 74, introduzidos originalmente pela *Texas Instruments* e que são hoje um padrão da indústria, fornecidos por diversos fabricantes.

A série 54 é de uso militar e opera na faixa de temperatura de –55 °C a +125 °C, com uma tensão de alimentação de 5 ± 0,5 V.

A série 74 é de uso geral, operando na faixa de temperatura de 0 °C a +70 °C, com alimentação de 5 ± 0,25 V.

Há centenas de funções disponíveis nas séries 54/74, abrangendo portas lógicas, flip-flops, decodificadores, contadores, etc.

Conforme o número de portas contidas em um CI (circuito integrado), ele se classifica:

- **SSI** (Small Scale Integration) ou integração em pequena escala – de 1 a 12 portas lógicas.
- **MSI** (Medium Scale Integraton) ou integração em média escala – de 13 a 99 portas lógicas.
- **LSI** (Large Scale Integration) ou integração em grande escala – de 100 a 1.000 portas lógicas.
- **VLSI** (Very Large Scale Integration) ou integração em escala muito grande – acima de 1.000 portas lógicas.

Além da série 54/74, que é a mais importante e que possui o maior número de funções disponíveis, existem algumas outras séries como a 4000 MTLL da Motorola e a 8200 da Signetics.

Características Gerais e parâmetros da Família TTL (portas Nand)

VERSÃO	IDENTIFICA-ÇÃO DA SÉRIE	ATRASO DE PROPAGAÇÃO POR PORTA	CONSUMO DE POTÊNCIA	FREQ. CLOCK MÁX. (FF)	OBSERVA-ÇÕES
Standard	54/74	10 ns	10 mW	35 MHz	comum
Low Power	54L/74L	33 ns	1 mW	3 MHz	baixíssimo consumo
High Speed	54H/74H	6 ns	22 mW	50 MHz	Alta velocidade
Schottky	54S/74S	3 ns	19 mW	125 MHz	Altíssima velocidade
Advanced Schottky	54AS/74AS	1,5 ns	8,5 mW	200 MHz	Altíssima velocidade e baixo consumo
Low Power Schottky	54LS/74LS	10 ns	2 mW	45 MHz	Baixíssimo consumo

Advanced Low Power Schottky	54ALS/74ALS	4 ns	1 mW	70 MHz	Altíssima velocidade e baixíssimo consumo
Fast	54F/74F	2 ns	5 mW	≈ 200 MHz	Altíssima velocidade e baixo consumo

74: Versão comercial **54:** Versão militar

Diferenças básicas:

Versão comercial: $0 < T < +70ºC$; $V_{cc} = 5\ V \pm 5\%$

Versão militar: $-55 < T < +125ºC$; $V_{cc} = 5\ V \pm 10\%$

A Família TTL Standard (54/74)

- Maior opções de circuitos
- Características adequadas para a maioria das aplicações

A Família Low Power (54L/74L)

- Resistores com valores cerca de 10 vezes superiores aos da porta padrão
- Redução da potência, com tempo de retardo
- Circuitos lentos e de baixa potência

A Família TTL Fast (54F/74F)

- Mais recente
- Entradas: com diodo, transistores NPN ou PNP;
- Elevada impedância de entrada , logo pouca corrente drenada e elevado FAN-OUT;
- Saídas: PULL UP (coletor aberto ou configuração darlington ou three state) e a PULL DOWN (mais comum);

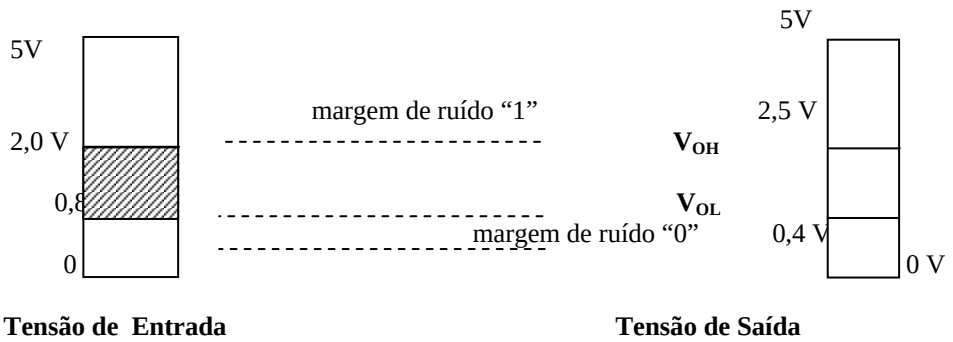
Escolhendo a família:

- Repertório Padrão e LS
- Velocidade
- Consumo de potência
- Custo

Menos onerosas → TTL padrão e LS
Mais elevado → AS, S e F

Além da série 54/74, que é a mais importante e que possui o maior número de funções disponíveis, existem algumas outras séries como a 4000 MTLL da Motorola e a 8200 da Signetics.

Características Gerais dos circuitos TTL



Tensões e Correntes Máximas e Margem de Ruído

- V_{OH} = 2.5 V
- V_{OL} = 0.25 V
- I_{OH} = - 0.4 mA
- I_{OL} = 4 mA

■ Margem de ruído
400 mV (standard)
300 mV (LS e S)

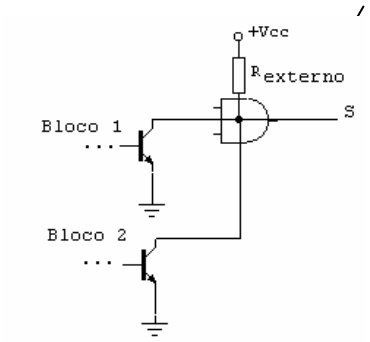
FAN-OUT de Várias Famílias de Circuitos Integrados

VERSÃO	FAN-OUT
Standard	**10**
Low Power	**20**
Schottky	**10**
Advanced Schottky	**40**
Low Power Schottky	**20**
Advanced Low Power Schottky	**20**
Fast	**40 (ent. Diodo)** **50 (ent. Transistor)**

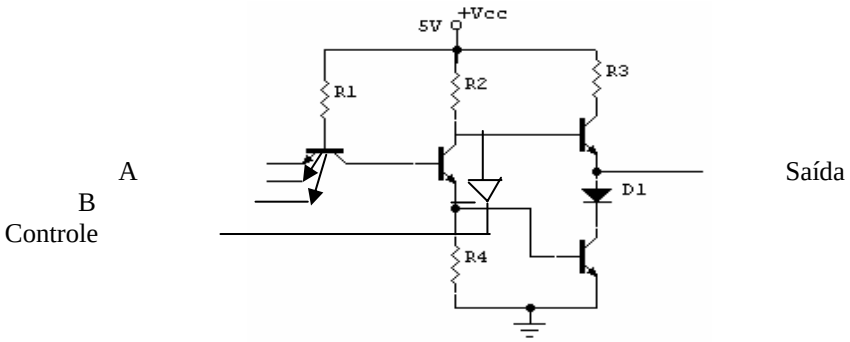
Tipos de saída (Aplicações especiais)

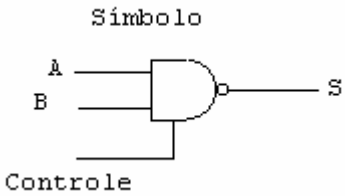
Saída Coletor Aberto:

- Permite o controle externo da corrente de coletor
- Aumento do fan-out
- Permite ligação em conjunto de várias saídas através de um único resistor de coletor (Wired-AND)



Portas em Three-state (ou Tri-state)

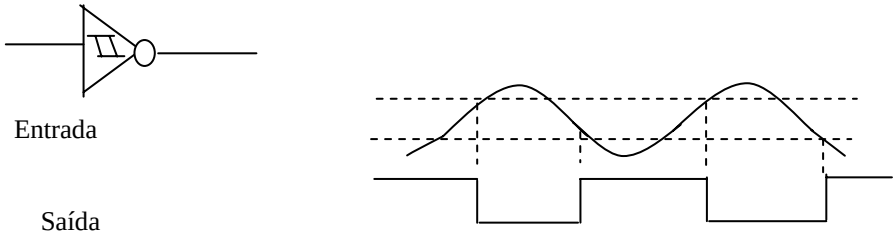




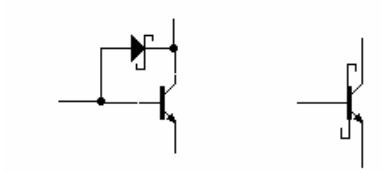
- Aplicação: sistemas com microprocessadores (via de dados ou endereço – bus ou barramento)
- Controle: 1 ou aberto, Porta NE normal
- Controle: 0 - Os três transistores à direita cortados, logo saída em alta impedância

Schimitt-Trigger

- Torna rápidas as variações lentas dos níveis de tensão de determinados sinais aplicados à sua entrada, causando na saída o aparecimento de uma onda quadrada bem definida
  → além de realizar sua função lógica, quadra o sinal aplicado à entrada, observando-se os limiares de tensão.



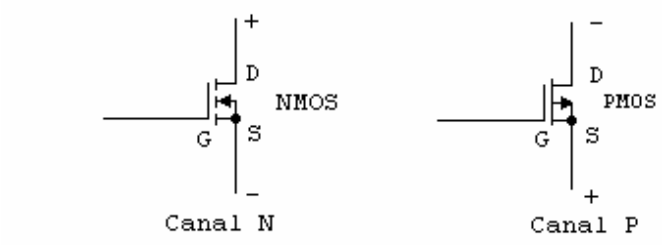
- Relógios digitais → sinal da rede → onda quadrada (60 Hz) → divisor de freqüência → Clock de 1 Hz
- Onde encontramos? - Portas Inversoras, Nand, buffers/drivers, registradores, outros



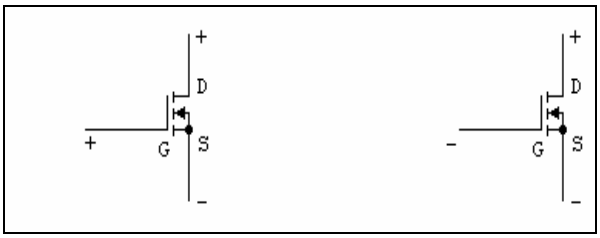
Transistor Schottky - Simbolo

A Família CMOS

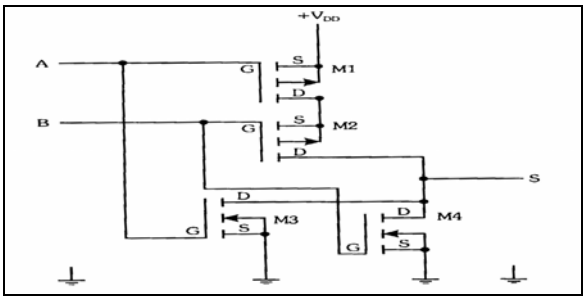
O transistor MOS-FET



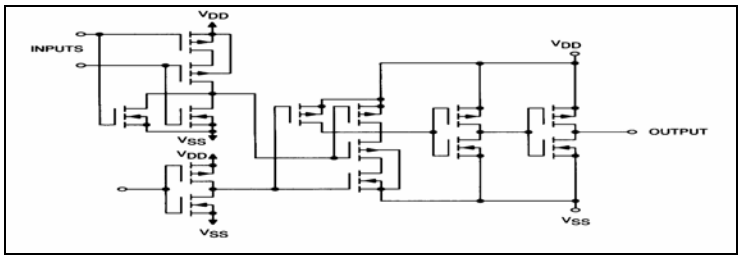
- Alto fan-out
- Alta imunidade ao ruído
- Baixíssimo consumo



Condução: Não condução:



Exemplo: Porta NOU



Que porta é esta?

Família CMOS

Características

- Baixa potência em baixas freqüência. A potência é altamente dependente da freqüência de operação do circuito.
- Uma vantagem importante do CMOS é a sua larga margem de alimentação, permitindo circuitos operarem a tensão altas como 10 a 15 V até tensões da ordem 2 V.
- Esta tecnologia apresenta vantagens em relação a família TTL, tais como baixa potência e menor tamanho físico dos circuitos
- Devido a estas características esta tecnologia é mais apropriada para projetos de circuito integrados.
- A mais recente evolução neste tipo de tecnologia é a redução do valor da tensão de alimentação destes componentes sem comprometer sua performance. Exemplo deste tipo de componente é a família lógica LCX que funciona a 3 V para suas entradas e saídas e permite conexão com dispositivos que trabalham a 5 V.

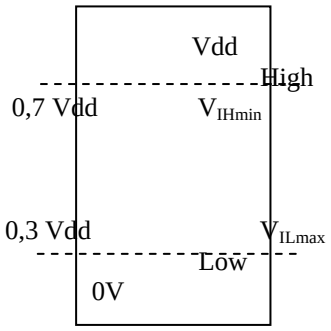
Séries: 4000A, 4000B e 54/74C (semelhante a TTL → pinos, blocos lógicos disponíveis)
74HC/74HCT: High Speed CMOS

faixa de temperatura: -40 < T < +85ºC (comum)
-55 < T < +125ºC (militar)

Principais parâmetros:

Alimentação: série 4000 e 74C: 3 V a 15 V
HC: 2V a 6V
HCT: 4,5 a 5,5 V
Séries de baixa voltagem: LV → 1 V a 3,6 V
LVC: 1,2 V a 3,6 V (3,3 V, uso mais comum)

Faixas de tensão para níveis lógicos alto e baixo para CMOS



Avanços tecnológicos em projeto e fabricação têm trazio CMOS para velocidades de operação e capacidade de drenar cargas compatíveis com a a família TTL.

- Tipos de componentes CMOS:
- MG (Metal-Gate CMOS)
- HC (High-speed silicon-gate CMOS)
- FACTTM (Advanced CMOS).

Alimentação
2 V a 15 V
Margem de ruído
1000 mV (standard-CMOSB)
800 mV (HCMOS)
1250 mV (ACMOS)

Comparação entre famílias lógicas

Typical Commercial Parameter (0°C to +70°C)	Logic Families												
	TTL				CMOS					ECL			
	LS	ALS	ABT	FAST	MG	HC	FACT	LVC	LCX	10H	100K	ECL in PS(3)	E-Lite
Speed													
Gate Prop Delay (ns)	9	7	2.7	3	65	8	5	3.3	3.5	1	0.75	0.33	0.22
Flip-Flop Toggle Rate (MHz)	33	45	200	125	4	45	160	200	200	330	400	1,000	2800
Output Edge Rate (ns)	6	3	3	2	50	4	2	3.7	3.6	1	0.7	0.5	0.25
Power Consumption Per Gate (mW)													
Quiescent	5	1.2	0.005	12.5	0.0006	0.003	0.0001	0.003	1E-04	25	50	25	73
Operating (1 MHz)	5	1.2	1.0	12.5	0.04	0.6	0.6	0.8	0.3	25	50	25	73
Supply Voltage (V)	+4.5 to +5.5	+4.5 to +5.5	+4.5 to +5.5	+4.5 to +5.5	+3 to +18	+2 to +6	+1.2 to +3.6	+2 to +3.6	+2 to +6	-4.5 to -5.5	-4.2 to -4.8	-4.2 to -5.5	-4.2 to -5.5
Output Drive (mA)	8	8	32/64	20	1	4	24	24	24	50 ohm load	50 ohm load	50 ohm load	50 ohm load
5V Tolerant													
Inputs	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Yes	Yes	N/A	N/A	N/A	N/A
Outputs	N/A	N/A	N/A	N/A	N/A	N/A	N/A	No	Yes	N/A	N/A	N/A	N/A
DC Noise Margin (1)													
High Input %	22	22	22	22	30	30	30	30	30	27	41	28/41	33
Low Input %	10	10	10	10	30	30	30	30	30	31	30	31/31	33
Packaging(4)													
DIP	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No
SO	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes
LCC	No	Yes	No	Yes	No	No	Yes	No	No	Yes	No	Yes	No
SSOP	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	No	No	No
TSSOP	No	No	No	No	Yes	Yes	Yes	Yes	Yes	No	No	No	No
Functional devices Types	190	210	50	110	125	103	80	35	27(2)	64	44	48	40
Relative 1-25 Gate Price U.S. $	.90	1.00	1.60	1.00	.90	.90	1.50	1.80	1.80	2.00	10.00	28.00	32.00

Anexo 2 – Circuitos Integrados dos Anos 90

CMOS

- CD4001 - Quad 2-In NOR Gate
- CD4007 - Dual Complementary Pair + Inverter
- CD4010 - Hex Buffer/Converter
- CD4011 - Quad 2-In NAND Gate
- CD4013 - Dual D Flip Flop
- CD4014 - 8-Bit Static Shift Register
- CD4015 - Dual 4-Bit Static Shift Register
- CD4016 - Quad Analog Switch Quad Multiplexer
- CD4017 - Decade Counter/Divider
- CD4020 - 14-Bit Binary Counter
- CD4021 - 8-Bit Static Shift Register
- CD4022 - Octal Divider Counter
- CD4023 - Triple 3-In NAND Gate
- CD4024 - 7-Bit Binary Counter
- CD4027 - Dual JK Flip Flop
- CD4028 - BCD to Decimal Decoder
- CD4029 - Presettable Up/Down Cntr
- CD4040 - 12-Bit Binary Counter
- CD4043 - Quad NOR R-S Latch
- CD4044 - Quad NAND R-S Latch
- CD4046 - Phase Locked Loop
- CD4047 - Monostable Multivibrator
- CD4049 - Hex Buffer, Inverter
- CD4050 - Hex Buffer Noninverter
- CD4051 - 8-Channel Analog Multiplexer
- CD4052 - Dual 4-Channel Analog Multiplexer
- CD4066 - Quad Analog Switch
- CD4069 - Hex Buffer, Inverter
- CD4070 - Quad XOR Gate
- CD4071 - Quad 2-In OR Gate
- CD4081 - Quad 2-In AND Gate
- CD4093 - Quad 2-In NAND Schmitt Trigger
- CD4503 - Hex 3-State Buffer
- CD4511 - BCD to 7-Segment Latch Decoder Driver
- CD4512 - 8 Channel Data Select
- CD4515 - 4-Bit Latch 4-to-16 Line Decoder (Low)
- CD4528 - Dual Monostable Multivibrator
- CD4538 - Dual Precision Monostable Multivibrator
- CD4541 - Programmable Oscillator Timer

TTL

- 74LS00 - Quad 2-In NAND Gate
- 74LS01 - Quad 2-In NAND Gate, Open Coll
- 74LS02 - Quad 2-In NOR Gate
- 74LS03 - Quad 2-In NAND Gate, Open Coll
- 74LS04 - Hex Inverter
- 74LS05 - Hex Inverter Open Coll
- 74LS08 - Quad 2-In AND Gate
- 74LS09 - Quad 2-In AND Gate Open Coll
- 74LS10 - Triple 3-In NAND Gate
- 74LS107A - Dual JK Flip Flop
- 74LS109A - Dual JK Edge Triggered Flip Flop
- 74LS11 - Triple 3-In AND Gate
- 74LS112A - Dual JK Edge Triggered Flip Flop

- 74LS113A - Dual JK Edge Triggered Flip Flop
- 74LS114A - Dual JK Edge Triggered Flip Flop
- 74LS122 - Retriggerable Monostable Multivibrator
- 74LS123 - Retriggerable Monostable Multivibrator
- 74LS125A - Quad 3 State Buffer, Lo-Enable
- 74LS126A - Quad Buffer 3 State Hi-Enable
- 74LS13 - Dual 4-Dual Schmitt Trigger
- 74LS132 - Quad 2-In Schmitt Trigger
- 74LS136 - Quad Exclusive OR Gate Open Coll
- 74LS138 - 1-of-8 Dcdr Demultiplexer
- 74LS139 - Dual 1-of-4 Dcdr Demultiplexer
- 74LS14 - Hex Schmitt Trigger
- 74LS145 - 1-of-10 Dcdr Drvr Open Coll
- 74LS151 - 8-In Multiplexer
- 74LS153 - Dual 4-In Multiplexer
- 74LS155 - Dual 1-of-4 Dcdr Demultiplexer
- 74LS156 - Dual 1-of-4 Dcdr Demultiplexer Open Coll
- 74LS157 - Quad 2-In Multiplexer Noninv
- 74LS158 - Quad 2-In Multiplexer Inv
- 74LS160A - BCD Decade Cntr Asynch Reset (9310 Type)
- 74LS161A - 4-Bit Binary Cntr Asynch Reset (9316 Type)
- 74LS162A - BCD Decade Cntr Synch Reset
- 74LS163A - 4-Bit Binary Cntr Synch Reset
- 74LS164 - 8-Bit Shift Register Ser In/Para Out
- 74LS165 - 8-Bit Shift Register Para In/Ser Out
- 74LS166 - 8-Bit Shift Register Para In/Ser Out
- 74LS169 - BCD Decade (Module Bi-Directional Cntr)
- 74LS173 - 4-Bit Type Register 3 State
- 74LS174 - Hex D-Type Flip Flop w/Clear
- 74LS175 - Quad D-Type Flip Flop w/Clear
- 74LS181 - 4-Bit ALU
- 74LS191 - Up/Down Binary Cntr
- 74LS193 - Up/Down Binary Cntr
- 74LS194A - 4-Bit Right/Left Shift Register
- 74LS195A - 4-Bit Shift Register (9300 Type)
- 74LS20 - Dual 4-In NAND Gate
- 74LS21 - Dual 4-In AND Gate
- 74LS240 - Octal Inv Bus Ln Drvr
- 74LS243 - Quad Bus Transceiver Noninv
- 74LS244 - Octal 3 State Drvr Noninv
- 74LS245 - Octal Bus Transceiver Noninv
- 74LS249 - BCD to 7 Seg Dcdr Drvr Open Coll
- 74LS253 - Dual 4-In Multiplexer, 3 State
- 74LS256 - Dual 4-Bit Addressable Latch
- 74LS257A - Quad 2-In Multiplexer 3 State
- 74LS26 - Quad 2-In NAND Buffer Open Coll
- 74LS27 - Triple 3-In NOR Gate
- 74LS273 - Octal D-Type Flip Flop w/Clear
- 74LS279 - Quad Set/Reset Latch
- 74LS283 - 4-Bit Full Adder (Rotated LS83A)
- 74LS30 - 8-In NAND Gate
- 74LS32 - Quad 2-In OR Gate
- 74LS352 - Dual 4-In Multiplexer
- 74LS365A - Hex Buffer w/Common Enable 3 State
- 74LS366A - Hex Inverter w/Common Enable 3 State
- 74LS367A - Hex Buffer 4-Bit and 2-Bit 3 State
- 74LS368A - Hex Inverter 4-Bit and 2-Bit 3 State
- 74LS373 - Octal Transparent Latch 3 State
- 74LS374 - Octal D-Type Flip Flop 3 State
- 74LS379 - 4-Bit D-Type Flip Flop w/Enable
- 74LS38 - Quad 2-In NAND Buffer Open Coll

- 74LS390 - Dual Decade Cntr
- 74LS42 - 1-of-10 Dcdr
- 74LS47 - BCD to 7 Seg Dcdr Drvr Open Coll
- 74LS49 - BCD to 7 Seg Dcdr Drvr Open Coll
- 74LS51 - Dual AND-OR-INVERT Gate
- 74LS670 - 4x4 Register File, 3 State
- 74LS73A - Dual JK Flip Flop
- 74LS74A - Dual D Flip Flop
- 74LS75 - 4-Bit Bi-Stable Latch w/Q&Q
- 74LS76A - Dual JK Flip Flop
- 74LS83A - 4-Bit Full Adder
- 74LS85 - 4-Bit Magnitude Comparator
- 74LS86 - Quad Exclusive OR Gate
- 74LS90 - Decade Cntr
- 74LS93 - 4-Bit Binary Cntr

Anexo 3 – Sites de Eletrônica Digital

Páginas Nacionais

http://www.terravista.pt/FerNoronha/2604/	Este endereço tem artigos diversos

Páginas Internacionais

http://www.williamson-labs.com/480_logic.htm#pos-neg-logic	Elementos Lógicos com Animação
http://www.ee.surrey.ac.uk/Projects/Labview/index.html	Curso de Sistemas Digitais
http://www.play-hookey.com/digital/	Lógica Digital
http://elwww.cc.purdue.edu/~garrods/DIGITAL/index.html	Eletrônica Digital
http://unix.cc.wmich.edu/~johnson/ece250	Idem
http://playpen.cs.cornell.edu/cs314/f98/Lectures/	Notas em PDF
http://classes.colgate.edu/phys282/	Livro de Eletrônica em PDF
http://www.ee.vt.edu/~ee2504sh/notes.html	Notas de Digitais
http://kingfisher.cms.shu.ac.uk/cm126/tutorials/	Artigos
http://spigot.anu.edu.au/people/mat/engn2211/notes/notes.html	Livro de Eletrônica
http://www.hkstar.com/~hkiedsci	Eletrônica Digital
http://www.4qd.co.uk/index.html http://www.4qd.co.uk/ccts/index.html#tra	Circuitos Diversos
http://www.ee.ualberta.ca/~charro/cookbook/	Circuitos Diversos

Anexo 4 – Exercícios Resolvidos

Sistemas de Numeração

ER.1 - Converter o número 28_{10} em binário, octal e hexadecimal.

Solução:

```
28 |2                          28 |8          26 |16
(0) 14 |2                       (4) (3)        (12) (1)
    (0) 7 |2
        (1) 3 |2
            (1) (1)
```

$28_{10} = 11100_2$ $28_{10} = 34_8$ $28_{10} = 1C_{16}$

ER.2 - Converter para decimal os números apresentados na primeira coluna da tabela abaixo:

Sistema	Solução	Respostas
$[1010]_2$	$1x2^3 + 0x2^2 + 1x2^1 + 0x2^0$	10_{10}
$[325]_8$	$3x8^2 + 2x8^1 + 5x8^0$	213_{10}
$[AB]_{16}$	$10x16^1 + 11x16^0$	171_{10}
$[201]_4$	$2x4^2 + 0x4^1 + 1x4^0$	33_{10}
$[201]_3$	$2x3^2 + 0x3^1 + 1x3^0$	19_{10}

Como apresenta a tabela, 201_4 é igual a $33_{10.}$ Já 201_3 é igual a 19_{10}. Simbolicamente, 201 é igual a 201, porém seu valor depende do sistema numérico adotado. Nos computadores a informação 01000001_2, pode representar vários tipos e valores de informação e isso depende de como o programa trata a variável associada. Se a informação faz parte de um nome, então 010000001 corresponde à letra "A" no código ASCII, a ser conhecido. Neste caso, um programa trata esta informação sempre como o caracter alfanumérico "A" e nunca como um número (que pode ser operado aritmeticamente). Caso contrário 01000001 pode representar 65_{10} se está em binário. Pode, ainda, se tratar de um caracter gráfico.

ER.3 - Realizar as conversões de base pedidas abaixo:

$(10110111)_2$	=	$(\qquad\qquad)_{16}$
$(CB)_{16}$	=	$(\qquad\qquad)_2$
$(42)_{10}$	=	$(\qquad\qquad)_{16}$
$(DE)_{16}$	=	$(\qquad\qquad)_{10}$
$(111111)_2$	=	$(\qquad\qquad)_{10}$
$(52)_{10}$	=	$(\qquad\qquad)_8$
$(45)_8$	=	$(\qquad\qquad)_{10}$
$(1110111)_2$	=	$(\qquad\qquad)_8$

Solução:

$10110111_2 = 1011_2\ 0111_2 = B7_{16}$ (por agrupamento)
$CB_{16} = 1100_2\ 1011_2 = 11001011_2$ (por agrupamento)
$42_{10} = 2 \times 16^1 + 10 = 2A_{16}$ (divisão por 16)
$DE_{16} = 13x16^1 + 14 = 222_{10}$ (polinômio)
$111111_2 = 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 63_{10}$ (polinômio)
$52_{10} = 6x8 + 4 = 64_8$ (divisão por 8)
$45_8 = 4x8^1 + 5 = 37_{10}$ (polinômio)
$1110111_2 = 1_2\ 110_2\ 111_2 = 167_8$ (agrupamento)

Operações Aritméticas

ER.4 - Realizar as operações aritméticas abaixo em binário:

a) 1001111 + 101010 b) 1001010 – 101111
c) 10011000000 x 10010000 d) 111111 / 1100

Solução:

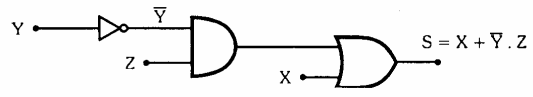
```
a)   1001111
   +  101010
   --------------
      1111001

b)   1001010
   -  101111
   -----------
    0011011

c)    10011000000
    x 10010000
  -----------------------
      10011
    10011
 ---------------------------
    101010110000000000

d)   111111   | 1100
     001111   | 101,01
       001100
         (0000)
```

Obs.: Para saber se os resultados estão corretos, converter para decimal.

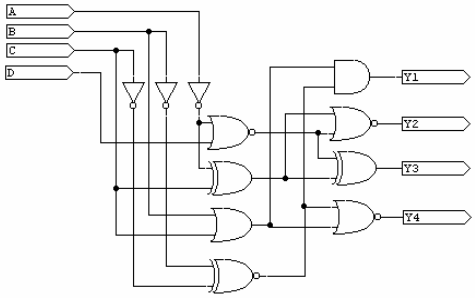
Formação e Simplificação de Expressões Lógicas

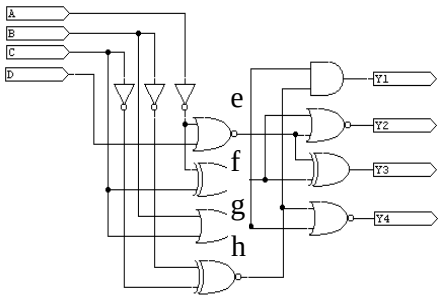
ER.5 – Dada a expressão booleana: $S = X + \overline{Y} \,.\, Z$, obter o circuito correspondente.

Solução:



ER.6 - A partir do circuito combinacional, obter a expressão booleana resultante para a saída Y1:

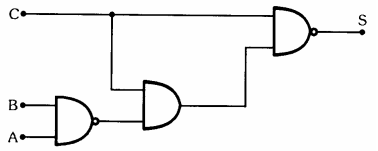




Y1 = g.h porém g = B + C e

h = $\overline{B} \otimes \overline{C}$ então Y1 = (B + C) . ($\overline{B} \otimes \overline{C}$)

ER.7 – Dado o circuito combinacional a seguir, obter sua tabela-verdade:



Solução:

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

ER.8 – Simplificar a expressão: $F = \overline{(\overline{K.M} + L + N)} + M.\overline{(K.M.N)}$

Solução:
Aplicando o teorema de De Morgan aos dois termos, obtém-se:

$$F = (\overline{\overline{K} + \overline{M} + L + N}) + M.(\overline{K} + \overline{M} + \overline{N})$$

Aplicando De Morgan no primeiro termo e a propriedade distributiva no segundo termo, obtém-se:

$$F = K.M.\overline{L}.\overline{N} + M.\overline{K} + M.\overline{M} + M.\overline{N}$$
$$F = K.M.\overline{L}.\overline{N} + M.\overline{K} + M.\overline{N}$$
$$F = M.\overline{N}.(K.\overline{L} + 1) + M.\overline{K}$$
$$F = M.\overline{N}.1 + M.\overline{K}$$
$$F = M.(\overline{N} + \overline{K})$$

Formação e Simplificação de Expressões Lógicas

ER.9 - Dada a tabela verdade abaixo, encontrar as expressões lógicas equivalentes em termos de: a) soma de produtos b) produto de somas:

Solução:

C	B	A	Y	Minitermos	Maxitermos
0	0	0	1	$Y_0 = \overline{C}.\overline{B}.\overline{A}$	
0	0	1	0		$Y_1 = C+B+\overline{A}$
0	1	0	1	$Y_2 = \overline{C}.B.\overline{A}$	
0	1	1	1	$Y_3 = \overline{C}.B.A$	
1	0	0	1	$Y_4 = C.\overline{B}.\overline{A}$	
1	0	1	1	$Y_5 = \overline{C}.B.A$	
1	1	0	0		$Y_6 = \overline{C}+\overline{B}+A$
1	1	1	0		$Y_7 = \overline{C}+\overline{B}+\overline{A}$

Maxi e Minitermos da Função Exemplo

$Y = Y_0 + Y_2 + Y_3 + Y_4 + Y_5 = \overline{C}.\overline{B}.\overline{A} + \overline{C}.B.\overline{A} + \overline{C}.B.A + C.\overline{B}.\overline{A} + C.\overline{B}.A$

$Y = Y_1.Y_6.Y_7 = (C+B+\overline{A}).(\overline{C}+\overline{B}+A).(\overline{C}+\overline{B}+\overline{A})$

ER.10 - Dada a tabela verdade abaixo, encontrar as expressões lógicas equivalentes em termos de: soma de produtos e produto de somas. Simplificar uma das expressões encontradas e apresentar o circuito simplificado.

Solução:

M	**N**	**O**	**Y**	**Minitermos**	**Maxitermos**
0	0	0	0		$Y_0 = M + N + O$
0	0	1	1	$Y_1 = \overline{M}.\overline{N}.O$	
0	1	0	1	$Y_2 = \overline{M}.N.\overline{O}$	
0	1	1	0		$Y_3 = M + \overline{N} + \overline{O}$
1	0	0	1	$Y_4 = M.\overline{N}.\overline{O}$	
1	0	1	0		$Y_5 = \overline{M} + N + \overline{O}$
1	1	0	0		$Y_6 = \overline{M} + \overline{N} + O$
1	1	1	1	$Y_7 = M.N.O$	

$Y = (M+N+O).(M+\overline{N}+\overline{O}).(\overline{M}+N+\overline{O})(M+\overline{N}+O) = \overline{M}\,\overline{N}O + \overline{M}N\overline{O} + M\overline{N}O + MNO$

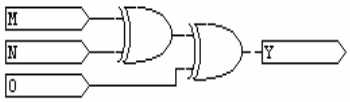
Simplificando a soma de produtos ...

$Y = \overline{M}\,\overline{N}O + \overline{M}N\overline{O} + M\overline{N}O + MNO$

$Y = \overline{M}.(\overline{N}O + N\overline{O}) + M.(\overline{N}\,\overline{O} + NO) = \overline{M}.(N \oplus O) + M.(\overline{N \oplus O})$

fazendo $N \oplus O = P$ obtém-se $Y = \overline{M}.P + M.\overline{P} = M \oplus P$ substituindo P

$Y = M \oplus N \oplus O$



Circuito Equivalente

ER.11 - Apresentar as tabelas-verdade das funções *AND, NOR e NAND* em função de variáveis relevantes e irrelevantes.

Solução

A	**B**	**Y = AB**
0	X	0
1	X	B
X	0	0
X	1	A

A	**B**	**Y =** $\overline{AB}$
0	X	1
1	X	$\overline{B}$
X	0	1
X	1	$\overline{A}$

A	**B**	**Y =** $\overline{A+B}$
0	X	$\overline{B}$
1	X	0
X	0	$\overline{A}$
X	1	0

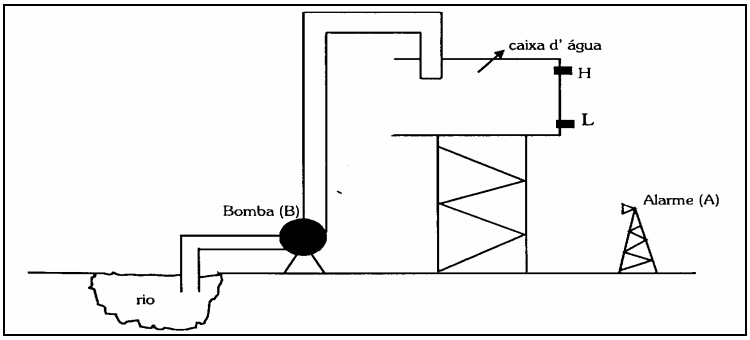
ER.12 - Repetir o exercício anterior com a expressão lógica P = M + N.Q

Solução:

M	**N**	**Q**	**P**
0	X	X	N.Q
1	X	X	1
X	0	X	M
X	1	X	M + Q
X	X	0	M
X	X	1	M + N

Mapas de Karnaugh

ER.13 - Controle de Bombeamento de Água. O desenho, a seguir, apresenta um processo simples para encher uma caixa d'água a partir do bombeamento da água de um rio próximo, utilizando-se sensores de nível (H e L), uma bomba (B) e um alarme (A).



Os sensores de nível alto (H) e de nível baixo (L) são utilizados para determinar o acionamento da bomba (B) e do alarme (A). São características dos sensores:

H = L = 0 $\rightarrow$ sensor desacionado, se a água está abaixo dele.
H = L = 1 $\rightarrow$ sensor acionado, se a água está sobre ou acima dele.

A bomba é acionada sempre que o nível da água da caixa atinge o sensor L. O alarme é acionado sempre que houver uma irregularidade. Um sensor é utilizado para indicar o estado da bomba e impedir que a bomba fique ligando/desligando sucessivas vezes, sempre que o nível da água estiver nas proximidades dos níveis H e L de monitorização.

Desenvolvimento:
1- Determinar as entradas e saídas;
2- Tabela-verdade

3- Expressões lógicas; e
4- Circuito lógico que controla o sistema.

Solução:

1. Variáveis de Entrada: H , L e sensor da Bomba (SB)
   Variáveis de saída: Bomba B e Alarme A

2. Tabela-verdade:

SB	**H**	**L**	**B**	**A**	Comentários
0	0	0	1	0	Caixa vazia: liga bomba
0	0	1	0	0	Caixa em uso, motor desligado)
0	1	0	0	1	Erro: possivelmente sensor com defeito
0	1	1	0	0	Caixa Cheia: desliga motor
1	0	0	1	0	Caixa vazia: motor ligado
1	0	1	1	0	Caixa enchendo
1	1	0	0	1	Erro: possivelmente sensor com defeito
1	1	1	0	0	Caixa Cheia: desliga motor

3) Expressões Lógicas

Bomba

L \ Sb H	00	01	11	10
0	1			1
1				1

$$B = Sb\,\overline{H} + \overline{H}\,\overline{L}$$

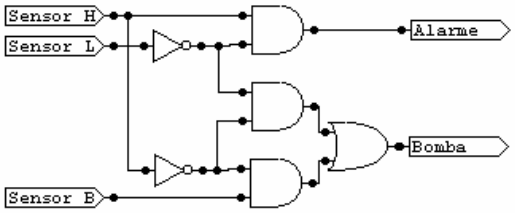
Alarme

L \ Sb H	00	01	11	10
0		1	1	
1				

$$A = H\,.\overline{L}$$

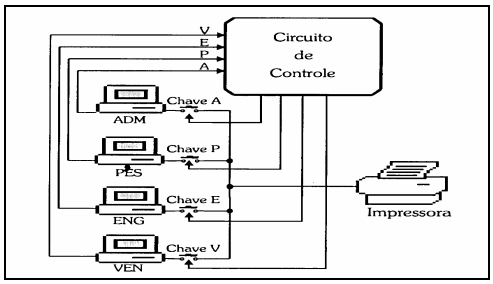
4) Circuito lógico

O circuito de controle pode ser implementado como segue:



ER.14. Controle de Uso de uma Impressora. A figura abaixo mostra, de forma esquemática, a conexão de 4 computadores de uma determinada empresa a uma impressora. Esta conexão é feita através de um circuito de controle que obedece às seguintes prioridades:

- Computador do setor administrativo (ADM) → 1ª prioridade
- Computador do setor pessoal (PES) → 2ª prioridade
- Computador do setor de engenharia (ENG) → 3ª prioridade
- Computador do setor de vendas (VEN) → 4ª prioridade



A conexão de computador à impressora é feita através de 4 chaves. Estas chaves devem ser abertas ou fechadas pelo circuito de controle. Isto significa que devem ser obtidas 4 expressões lógicas, cada uma descrevendo o funcionamento de uma chave do circuito de controle. Deve-se, então, determinar as entradas e saídas, montar a tabela-verdade e obter a expressão lógica e o circuito lógico correspondente que descreve o funcionamento do sistema.

1. Variáveis de entrada:
   Computador da administração = entrada A
   Computador do setor pessoal = entrada P
   Computador do setor de engenharia = entrada E
   Computador do setor de vendas = entrada V
   De modo que se A, P, V, E = 1 significa solicitação da impressora.

2. Variáveis de saída:
   Chave A = variável de saída CH_A
   Chave P = variável de saída CH_P
   Chave E = variável de saída CH_E
   Chave V = variável de saída CH_V

3. Montagem da tabela-verdade:

Linhas	**A**	**P**	**E**	**V**	CH_A	CH_P	CH_E	CH_V
1ª	0	0	0	0	0	0	0	0
2ª	0	0	0	1	0	0	0	1
3ª	0	0	1	0	0	0	1	0
4ª	0	0	1	1	0	0	1	0
5ª	0	1	0	0	0	1	0	0
6ª	0	1	0	1	0	1	0	0
7ª	0	1	1	0	0	1	0	0
8ª	0	1	1	1	0	1	0	0
9ª	1	0	0	0	1	0	0	0
10ª	1	0	0	1	1	0	0	0
11ª	1	0	1	0	1	0	0	0
12ª	1	0	1	1	1	0	0	0
13ª	1	1	0	0	1	0	0	0
14ª	1	1	0	1	1	0	0	0
15ª	1	1	1	0	1	0	0	0
16ª	1	1	1	1	1	0	0	0

4. Análise de cada combinação:
   A análise é feita linha a linha da tabela, observando-se quem está solicitando a impressora e quem tem prioridade, para, então, estabelecer que chave deve ser acionada.

   - 1ª linha: nenhum computador está solicitando a impressora, portanto, nenhum chave deve ser acionada;
   - 2ª linha: somente o computador do setor de vendas está solicitando a impressora, portanto, a chave CH_V deve ser fechada;
   - 3ª linha: somente o computador da engenharia está solicitando a impressora, portanto, a chave CH_E deve ser acionada;
   - 4ª linha: os computadores de vendas e da engenharia estão solicitando a impressora. Pela lista de prioridades, a engenharia vem em primeiro lugar, logo, somente a chave CH_E deve ser fechada.

- E assim por diante, até preencher toda a tabela.

5. Obtenção das expressões lógicas das saídas

a) SOMA DE PRODUTOS

$$CH_V = \overline{A}.\overline{P}.\overline{E}.V$$
$$CH_E = \overline{A}.\overline{P}.E.\overline{V} + \overline{A}.\vec{P}.E.V$$
$$CH_P = \overline{A}.P.\overline{E}.\overline{V} + \overline{A}.P.\overline{E}.V + \overline{A}.P.E.\overline{V} + \overline{A}.P.E.V$$
$$CH_A = A.\overline{P}.\overline{E}.\overline{V} + A.\overline{P}.\overline{E}.V + A.\overline{P}.E.\overline{V} + A.\overline{P}.E.V + A.P.\overline{E}.\overline{V} + A.P.\overline{E}.V + A.P.E.\overline{V} + A.P.E.V$$

b) Expressões simplificadas

c.1) Aplicando as propriedades e o teorema de De Morgan

$$CH_V = \overline{A}.\overline{P}.\overline{E}.V$$
$$CH_E = \overline{A}.\overline{P}.E$$
$$CH_P = \overline{A}.P$$
$$CH_A = A$$

c.2) Utilizando o Mapa de Karnaugh

(EV) \ (A P)	00	01	11	10
00				
01	1			
11				
10				

$CH_V = \overline{A}.\overline{P}.\overline{E}.V$

(EV) \ (A P)	00	01	11	10
00				
01				
11	1			
10	1			

$CH_E = \overline{A}.\overline{P}.E$

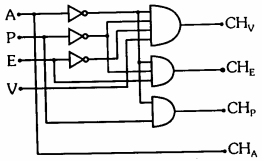
(EV) \ (A P)	00	01	11	10
00		1		
01		1		
11		1		
10		1		

$CH_P = \overline{A}.P$

(EV) \ (A P)	00	01	11	10
00			1	1
01			1	1
11			1	1
10			1	1

$CH_A = A$

6. Obtenção do circuito lógico



Este circuito, da forma como está, utilizaria, para sua implementação, 4 circuitos integrados: 7404, 7408, 7411 e 7421, ficando todos eles subutilizados.

Circuitos Aritméticos Básicos

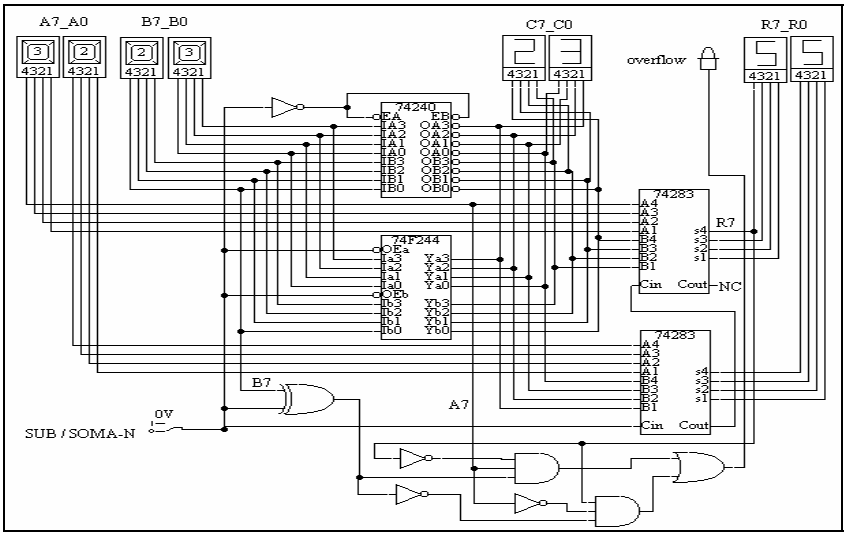
ER15. Somador / Subtrator

O circuito abaixo realiza operações de soma e subtração dos números A e B, de 8 bits cada (editados via teclas hexadecimais), onde o mais significativo e' de sinal, de forma que os números entre 00_{16} a $7F_{16}$ (0_{10} a 127_{10}) compreendam os números não-negativos, e entre 81_{16} a FF_{16} (-127_{10} a -1_{10}) os números negativos.

O circuito soma A e B sempre que o sinal SUB/SOMA-N esta' em "0". Nesta operação os circuitos integrados 74283 (somadores de nos. de 4 bits) recebem os sinais de A e B fielmente (A diretamente e B via o buffer 74244 que, quando habilitado se comporta como um conjunto de 8 portas não inversoras). Neste caso, R = A + B + "0" (pelo fato do C_{in} do primeiro 74283 receber o sinal SUB/SOMA-N = "0").

O circuito subtrai B de A sempre que o sinal SUB/SOMA-N esta' em "1". Nesta operação os circuitos integrados 74283 recebem os sinais de A diretamente, e B complementados via o buffer 74240 que, quando habilitado, age como um conjunto de 8 portas inversoras). Neste caso, R = A + complemento de B + "1" (pelo fato do C_{in} do primeiro 74283 receber o sinal SUB/SOMA-N = "1"), resultando em A + (- B) ou A - B.

Como os circuitos 74283 são somadores nos quais os bits mais significativos não são tratados como bits de sinal, um circuito lógico fez-se necessário para que falsas condições de "overflow" fossem detectadas, entre outras incoerências. Este circuito pode ser obtido através da solução do MVK_5 dado abaixo, a qual resulta na expressão:



$$\text{overflow} = \overline{A_7}.R_7.(B_7 \oplus \text{SUB/SOMA-N}) + \overline{A_7}.R_7.\overline{(B_7 \oplus \text{SUB/SOMA-N})}$$

$C_{out}\ R_7$ \ A_7B_7	00	01	11	10
00			x	
01	1			
11	1			
10	x		1	

(SUB/SOMA-N = "0")

$C_{out}\ R_7$ \ A_7B_7	00	01	11	10
00				x
01		1		
11		1		
10				1

(SUB/SOMA-N = "1")

No mapa, "x" e' um estado que nunca ocorre.
No circuito, o display C apresenta o valor de B (quando na soma) ou seu complemento (quando na subtração).

Anexo 5 – Exercícios Propostos

Sistemas de Numeração

EP.1 – Sucintamente, descrever diferenças existentes entre quantidades digital e analógica.

EP.2 – Indicar quais das seguintes proposições são quantidades digitais, e quais são analógicas:

a) Chave de 10 posições
b) Temperatura
c) Grãos de areia na praia

EP.3 – Realizar as conversões de base abaixo:

$(10010011)_2$	= (	$)_{16}$	e)	$(BC)_{16}$	= (	$)_2$
$(11011100)_2$	= (	$)_{10}$	f)	$(57,6)_8$	= (	$)_{10}$
$(34)_{10}$	= (	$)_{16}$	f)	$(43)_8$	= (	$)_{10}$
$(ED)_{16}$	= (	$)_{10}$	g)	$(495,375)_{10}$	= (	$)_2$

EP.4 – Preencher os quadros em branco, realizando as conversões de base pedidas, de modo que cada coluna possua o mesmo valor numérico.

Binário					01110011
Octal				247	
Hexadecimal			5E		
Decimal		65			
BCD	10000101				

Operações Aritméticas

EP.5 - Fazer um estudo sobre *Estouro de Capacidade*, explicando para que casos nas operações de soma e subtração podem ocorrer estouros, produzindo uma resposta incorreta. Descrever o que programador deve fazer para corrigir esse tipo de erro.

EP.6 - Realizar as operações aritméticas abaixo em binário:

a) 10101 + 01101 =
b) 1010 x 11000 =
c) 1010 + 11101 =
d) 1001 x 1010 =
e) 1110 - 1001 =
f) 10101 / 11 =
g) 10101 - 1010 =
h) 11110 / 11 =

EP.7 - Realizar as operações aritméticas a seguir em binário com 8 bits. Verificar em decimal se os resultados encontrados são coerentes. No caso (ou não) de overflow, justificar sua resposta.

a) $89_{10} + 45_{10} = ?$ b) $94_{10} + 67_{10} = ?$ c) $94_{10} – 67_{10} = ?$

As Funções Lógicas Básicas, Tabela-Verdade e Simbologia

EP.8 – Se associado à presença de uma tensão o nível lógico 1 e à ausência o nível 0, que tipo de lógica está-se adotando?

EP.9 – O que são as funções lógicas e as portas lógicas ?

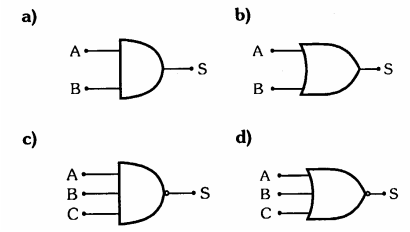
EP.10 – Qual é o nome da função lógica em que obtemos uma saída 1 quando as entradas se encontram em níveis lógicos diferentes ?

EP.11 – Em qual condição de entrada a saída de uma porta OR é 0 ?

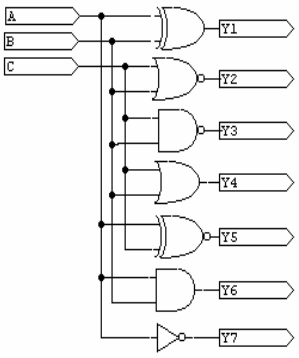
EP.12 – Determinar a tabela-verdade de uma porta AND de 3 entradas.

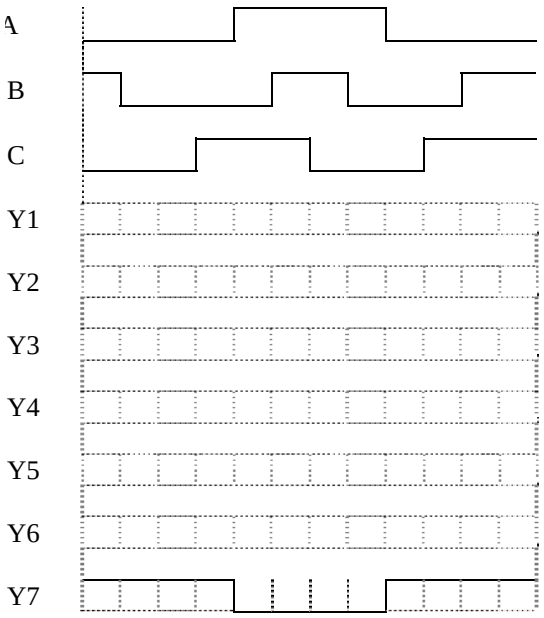
EP.13 – Determinar a tabela-verdade de uma porta OR de 3 entradas.

EP.14 – Dados os blocos lógicos abaixo, apresentar suas tabelas-verdade correspondentes.



EP.15 - Dado o circuito ao lado, apresentar as expressões lógicas de Y1, Y2, Y3, Y4, Y5 e Y6. Completar o diagrama de tempo a seguir em função dos níveis lógicos que assumem as entradas A, B e C (observar o exemplo apresentado em Y7):





Formação e Simplificação de Expressões Lógicas

EP.16 - Marcar as alternativas corretas:

1.1) O resultado da operação A.A é:

a) 0 b) 1 c) A d) $\overline{A}$ e) n.r.a.

1.2) O resultado da operação $\overline{A}$ + A é:

a) 0b) 1 c) A d) $\overline{A}$ e) n.r.a.

1.3) O resultado da operação A + A.B é:

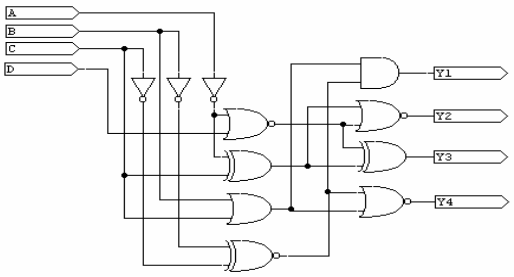
a) 1b) A c) B d) $\overline{A}$ + B e) n.r.a.

1.4) O resultado da operação (AC) ⊕ (BC) é:

a) A$\overline{B}$C + AB$\overline{C}$ b) $\overline{A}$B + BC c) ABC d) A+BC e) n.r.a.

EP.17 - Dado o circuito abaixo

1.1- Encontrar as expressões de Y2, Y3 e Y4 em função das entradas A, B, C e D.



1.2 - Na expressão de Y1 se B = 0, encontra-se Y1 igual a:

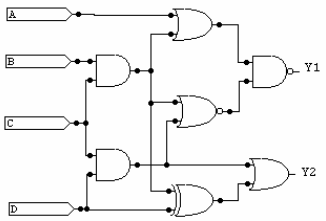
a) 0 b) 1 c) C d)A e)D

1.3 - Na expressão de Y3 se A = 1, encontra-se Y3 igual a:

a) 0 b) $C \otimes D$ c) C d) 1 e) D

1.4 - Simplificar as expressões de Y1, Y2, Y3 e Y4 do circuito apresentado em 1.1 e apresentar o circuito equivalente.

1.5 - Simplificar as expressões de Y1 e Y2 do circuito a seguir e apresentar o circuito equivalente:



EP.18 - Apresentar os resultados das operações lógicas abaixo:

0.1 =	A + 1 =
1.1 =	1 + 0 =
A.1 =	A + 0 =
B. 0 =	0 + 0 =

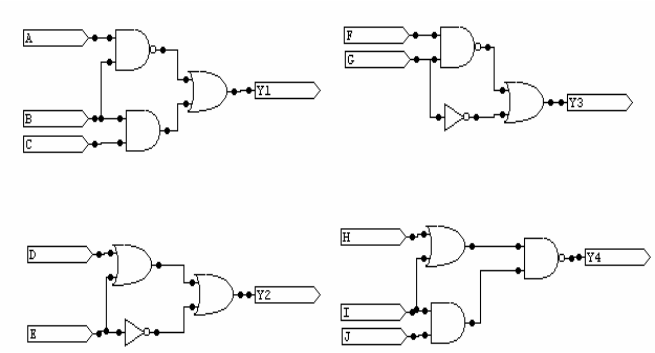
EP.19 - Desenhe os circuitos lógicos definidos pelas expressões abaixo:

$Y = \overline{A.B} + CD$	$M = \overline{N + P.Q}$
$X = \overline{U} + \overline{V}.\overline{Z}$	$K = \overline{L.(O + T)}$

EP.20 - Simplifique as expressões abaixo:

$Y = \overline{A.B} + \overline{A}.\overline{B}$	$M = \overline{N + P.N}$
$X = \overline{U} + \overline{Z}.U$	$K = \overline{L}.\overline{(O + L)}$

EP.21 - Apresentar as expressões lógicas correspondentes aos circuitos dados abaixo:



Formação e Simplificação de Expressões Lógicas

EP.22 - Para as expressões lógicas a seguir, fazer o que se pede:

1.1 Desenhar o diagrama lógico correspondente;

1.2 Simplificar utilizando os teoremas de Boole e de DeMorgan;

1.3 Desenhar os circuitos lógicos correspondentes às expressões simplificadas

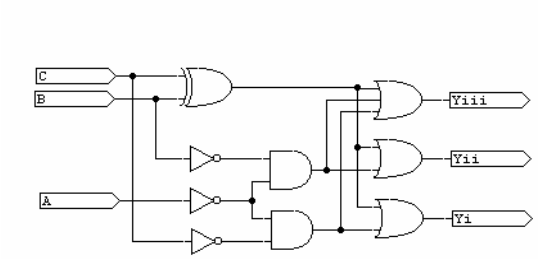
i) $P = [((M.\ \overline{Q} + \overline{M}.Q).N)] + [(Q\ .(\ \overline{N + Q.M})]$

ii) $K = \overline{X.Y} + Y.\overline{Z} + \overline{X}.Z$

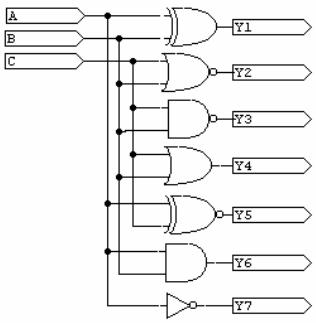
iii) $L = (A.B.\overline{C}) + BC(\overline{A + C})$

iv) $I = JM\overline{N} + \overline{JNM} + \overline{J + NM}$

EP.23 - Simplificar as expressões encontradas no exercício resolvido ER.2 e identificar que saídas / circuitos abaixo estão associadas às expressões encontradas.



EP.24 - Apresentar o circuito lógico (sem simplificação) das saídas M, N, O e P em função das entradas A, B e C.



Y = $(Y1+\overline{Y4}).Y6$
N = $Y2.\overline{Y4}+Y4$
O = $(Y3+\overline{Y6}).Y7$
P = $\overline{Y2}+Y5.\overline{Y4}$

EP.25 - Apresentar a solução simplificada para cada uma das expressões do item anterior e o circuito equivalente.

EP.26 - Para as tabelas verdade dadas a seguir, encontrar os minitermos e as expressões lógicas reduzidas a partir da soma de produtos e desenhar os circuitos lógicos equivalentes. Discutir os resultados encontrados.

a)

1	0	

b)

1	0	

c)

d)

EP.27 - Para as tabelas-verdade dadas a seguir, apresentar as expressões lógicas usando o produto de somas, os circuitos lógicos correspondentes e as expressões simplificadas.

a)

1	0	a

b)

			b

Mapas de Karnaugh

EP.28 - Para as expressões lógicas abaixo:

a) simplificar utilizando Mapas de Karnaugh.

b) desenhar os circuitos lógicos correspondentes às expressões encontradas em a)

i) P = $\overline{M}\,\overline{Q}R + \overline{M}QR + \overline{M}\,\overline{Q}\,\overline{R} + \overline{Q}R$

ii) $K = \overline{X}\,\overline{Y}ZW + XY\overline{W} + \overline{Z}Y\overline{W} + ZW$

iii) $L = A\,\overline{B}\,\overline{C}\,\overline{D} + AB\overline{D} + \overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}\,\overline{D}$

iv) $I = JN\overline{O} + JM\overline{N}\,\overline{O} + \overline{J}\,\overline{N}MO + \overline{J}\,\overline{N}M$

EP.29 - Dada a tabela verdade abaixo, encontrar o mapa de Karnaugh correspondente.

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

BA \ DC	00	01	11	10
00				
01				
11				
10				

EP.30 - Dados os mapas de Karnaugh abaixo, encontrar as expressões simplificadas e desenhar os circuitos lógicos correspondentes.

(MN) (XY)

(O)	00	01	11	10
0	1	1	1	
1	1	1		

(Z)	00	01	11	10
0		1	1	
1		1	1	

(BC) (PQ)

(A)	00	01	11	10
0		1		1
1	1		1	

(R)	00	01	11	10
0	1			1
1	1			1

MN\OP

	00	01	11	10
00	1	1	1	
01	1	1		
11	1	1		1
10	1	1		1

XY\ZW

	00	01	11	10
00		1	1	
01	1	1	1	1
11	1	1	1	1
10		1	1	

AB\CD

	00	01	11	10
00	1	1	1	1
01	1		1	
11	1		1	
10	1	1	1	1

RS\PQ

	00	01	11	10
00	1			1
01	1			1
11		1	1	
10		1	1	

MN\OP

	00	01	11	10
00	1	1		1
01				1
11		1	1	
10	1			1

XY\ZW

	00	01	11	10
00	1	1	1	1
01	1		1	1
11				
10			1	1

EP.31 - Dadas as tabelas-verdade abaixo, simplificá-las, apresentar as expressões lógicas resultantes e os circuitos lógicos correspondentes:

M	N	O	Y
x	0	1	1
0	1	0	x
0	1	1	x
1	1	0	1

D	C	B	A	Y
0	0	x	0	1
1	1	x	1	1
0	1	x	x	x
1	0	x	x	1

EP.32 - Dada a tabela verdade abaixo, encontrar as expressões simplificadas e marcar as alternativas corretas:

Entradas			Saídas		
C	B	A	K	L	M
0	0	0	1	0	1
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	0	0	1

EP.32.1. Na expressão simplificada de K encontra-se, se A = 1, K resulta em:

a) $\overline{B+C}$ b) B c) 1 d) $\overline{B}C$ e) n.r.a.

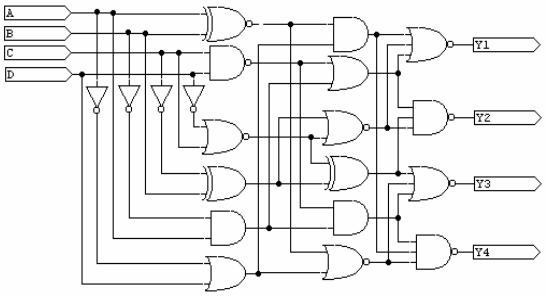
EP.32.2. Simplificando a expressão de L, L é igual a:

a)$A \oplus C$ b)AB+C c) $A \oplus B$ d) $A \otimes C$ e) n.r.a

EP.32.3. Na expressão de M, se C = 0, M é igual a

a) $\overline{A}$ b) B c) $\overline{A}$ d) B e) n.r.a

EP.33 - Encontrar a expressão lógica das saídas do circuito ao lado, simplificá-las até obter uma soma de produtos (usando De Morgan), depois simplificá-las à partir dos mapas de Karnaugh correspondentes e apresentar as expressões reduzidas e os respectivos circuitos:

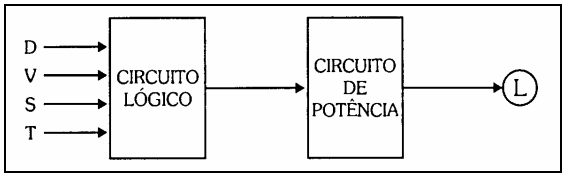


EP.34 – Determinar as entradas e saídas, montar a tabela-verdade, obtenha a expressão lógica e o circuito lógico correspondente que descreve o funcionamento do sistema abaixo.

Sistema de Votação. Uma escola tem sua diretoria constituída pelos seguintes elementos: Diretor, Vice-Diretor, Secretário e Tesoureiro. Uma vez por mês esta diretoria se reúne para decidir sobre diversos assuntos, sendo que as propostas são aceitas ou não através de votação. Devido ao número de elementos da diretoria ser par; o sistema adotado assim se comporta:

- Maioria absoluta – a proposta é aceita ou não se no mínimo três elementos são, respectivamente, a favor ou contra;
- Empate – vence o voto dado pelo diretor.

A figura a seguir mostra o diagrama de bloco desse sistema de votação.



Projetar um circuito que acenda uma lâmpada caso a proposta seja aprovada pela diretoria.

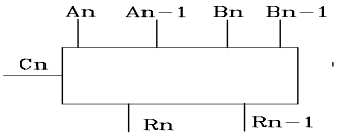
Circuitos Aritméticos Básicos

EP.35. Apresentar o diagrama de blocos de um sistema que soma 3 números de 2 bits cada, utilizando circuitos meio somador e somador completo.

EP.36. Desenvolver um circuito que soma dois números de 2 bits, utilizando os circuitos meio somador e somador completo já conhecidos.

EP.37. De forma semelhante aos circuitos somadores, é possível o desenvolvimento de circuitos meio subtradores e subtradores completos. Na prática, na ULA, os circuitos que somam são os mesmos que subtraem, porque a subtração consiste de uma soma de um número com um complementar (A – B = A + (-B), onde (-B) é obtido do complemento de dois de B. Desenvolver os circuitos meio subtrator e subtrator completo.

EP.38 - Desenvolver um meio somador que soma duas ordens (dois bits) de dois números, conforme sugere a figura abaixo.



Meio Somador de 2 bits

EP. 39. Desenvolver, também, o somador completo de 2 bits.

EP.40 - Conceitos de ULA. A ULA de um microprocessador consiste de diversos circuitos lógicos e aritméticos que realizam operações determinadas pelos estados de sinais de controle. Baseado nesta idéia, desenvolver um circuito que possui duas aplicações selecionadas por um sinal de controle:

- quando o sinal de controle está em “0” o circuito funciona como um somador completo;
- quando o sinal de controle está em “1” o circuito funciona como um subtrador completo.

Comparadores

EP41. Usando o 74LS85, compare os seguintes números. A = 0100 e B = 0101; A= 1000 e B = 0101; e A = 1010 e B = 1010. Use Led' de cores diferentes (da sua preferência) para cada uma das saídas.

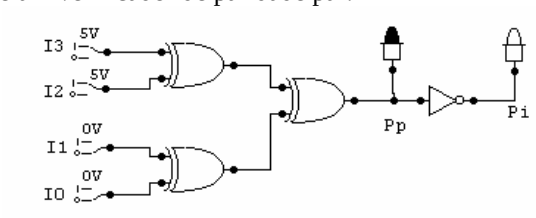
Paridade

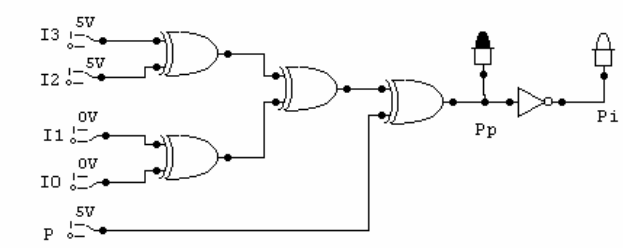
EP.42 - Para um sistema com 3 bits, desenvolver um circuito que gera o bit de paridade (par) do conjunto de sinais C, B e A; e um circuito que verifica a paridade (par) do conjunto de sinais BP, C, B e A.

EP.43 - Repetir o EP.1 para paridade ímpar.

EP.44 - Utilizando os circuitos encontrados nos exercícios anteriores desenvolva um circuito que gera e verifica paridades, cujos tipos são determinado por um sinal de controle de nome CNTL, de forma que: se CNTL = "0" o circuito gerador gera paridade par e o verificador verifica paridade par; e se CNTL = "1" o circuito gerador gera paridade ímpar e o verificador verifica paridade ímpar.

EP.45 - Utilizando o programa simulador: Montar/editar os circuitos dados ao lado, sabendo que tratam-se de um gerador de paridade par/ímpar e um verificador de paridade par.

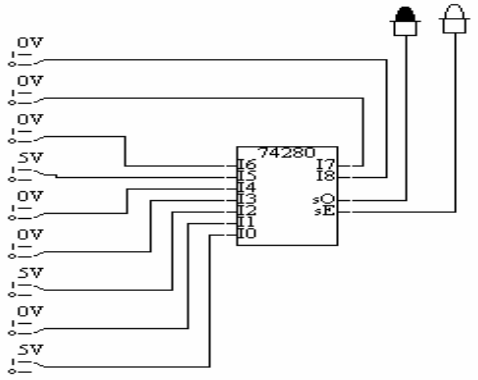




- Verificar o funcionamento do gerador de bit de paridade, estabelecendo diferentes níveis lógicos nas entradas A0_A3. Anotar seus dados e apresentar conclusões.
- Verificar o funcionamento do verificador de paridade par, estabelecendo diferentes níveis lógicos nas entradas R0_R3 e R_par. Anote seus dados e tire as suas conclusões.

EP.46 - Apresentar o diagrama lógico de um gerador de paridade ímpar de 8 bits para um circuito de transmissão digital e o diagrama lógico do verificador de paridade necessário para o circuito de recepção digital.

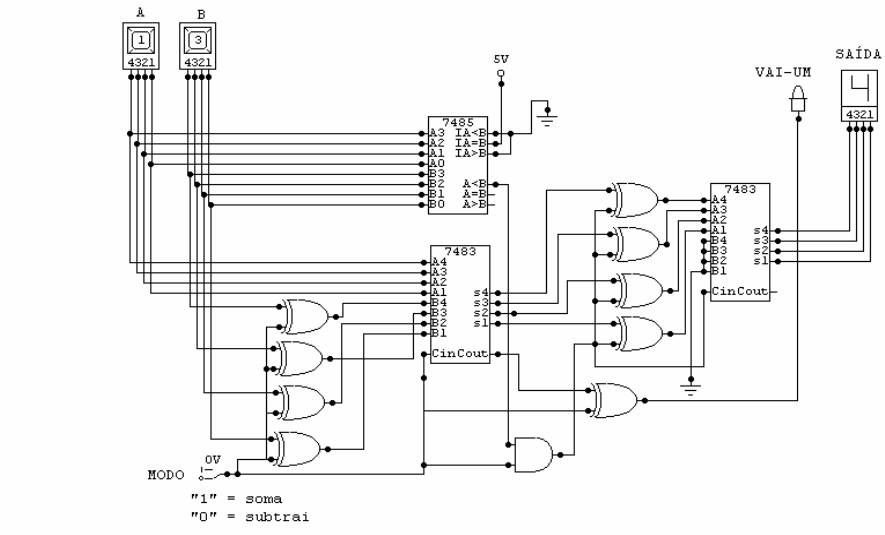
EP.47 - Utilizando o programa simulador, verificar e descrever o funcionamento do circuito abaixo, sabendo tratar-se de um gerador e verificador de paridade de oito bits. Apresentar suas conclusões.



GERADOR/VERIFICADOR
DE PARIDADE

Circuitos Aritméticos

EP48. Monte o circuito abaixo e verifique o seu funcionamento.

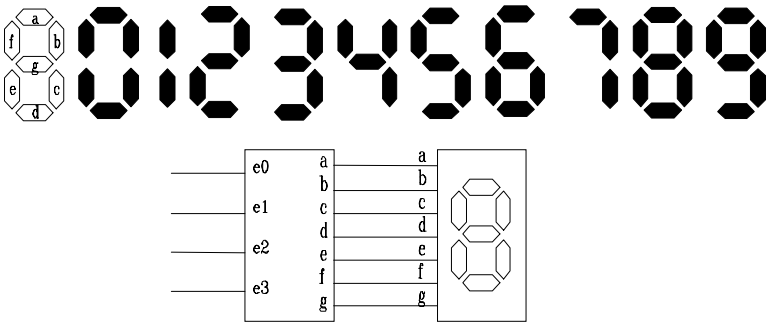


Responda:

- Para que serve a chave MODO?
- Para que serve a porta lógica AND?
- Qual a utilidade das portas XOR que estão em grupos de 4, nas entradas dos CI's 7483?
- Como fazer para implementar um circuito que execute a mesma função para números A e B de oito bits? (Desenhe o diagrama de blocos).

Codificadores e Decodificadores

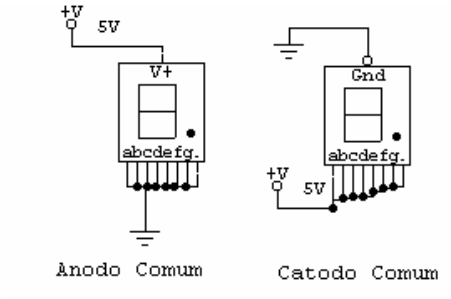
EP49. Construir um decodificador BCD de 7-segmentos significa desenvolver um circuito que à partir do código binário (geralmente o BCD) gere um código abcdefg que faça ächender corretamente os segmentos a, b, c, d, e, f, ou g que formarão o número decimal:



decodificador display

Os displays podem ser anodo comum ou catodo comum, cujos segmentos acendem quando recebem nível lógico 0 ou 1, respectivamente. A Tabela-Verdade para um display catodo comum é mostrada a seguir.

Anodo Comum Catodo Comum



Construa o mapa de Karnaugh para cada um dos segmentos e desenhe o circuito lógico simplificado para cada um deles.

EP50. Construa a Tabela-Verdade e o circuito lógico simplificado pelo mapa de Karnaugh de um decodificador que, a partir de um código binário, escreva a sequência Cd PLAYEr.

EP51. Projete um decodifcador BCD 8421 para Excesso 3.

EP52. Projete um decodifcador BCD 8421 para o código 2 entre 5.

EP53. Faça um projeto que, a partir de um código binário, escrevA a seqüência do sistema hexadecimal em um display de 7 segmentos anodo comum.

EP54. Apresente a tabela verdade de um decodificador binário para display de 7 segmentos para um display onde cada segmento emite luz apenas quando na sua entrada correspondente é mantido o nível; lógico “O”

EP55. Apresente o M.K., para a saída a do decodificador encontrado no item anterior, a expressão resultante e o circuito correspondente.

EP56. Descreva as principais características do CI’s 74247 e 74248.

EP.57 – Responder sucintamente:

- Qual o objetivo dos códigos em um sistema digital ?
- Qual o objetivo dos códigos alfanuméricos ?
- Para que são utilizados os códigos *EBCDIC* e *ASCII*, respectivamente ?
- O que são os codificadores e decodificadores ?
- Para o decodificador de 7 segmentos em 8.3.1., qual o M K. e a expressão lógica da saída g?

EP57.1 - Complete os espaços a seguir:

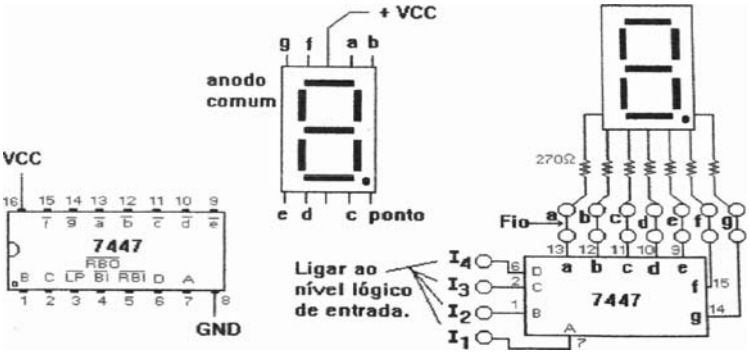
A principal função dos codificadores e' converter códigos ou sinais _____ para o código _____. Já os decodificadores fazem o papel _____. Como exemplo, podemos citar o código _____ que é largamente utilizado no mundo ocidental para troca de informações entre equipamentos, tais como computador e impressora.

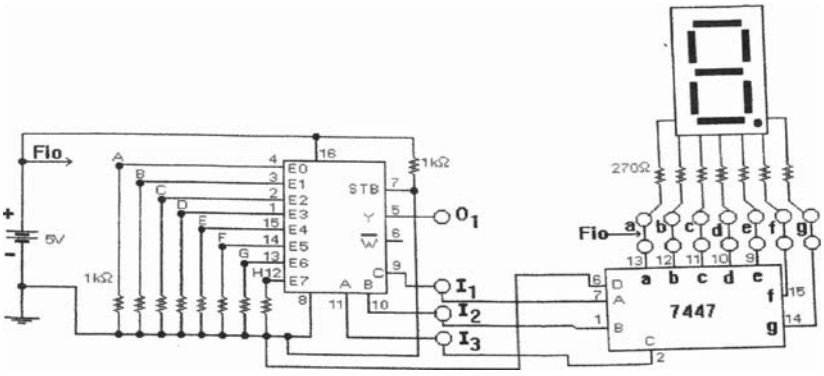
Em um circuito com um display de 7 segmentos, quando está visível o número 5, significa que somente os leds do display _____ estão acesos. Se o decodificador para este display tem suas saídas ativas em nível baixo, então o display em questão é _____ comum.

EP.58 - Faça o que se pede:

1. Diferenciar o código *BCD8421* do código *Excesso 3*.
2. Apresentar a tabela verdade de um decodificador binário para display de 7 segmentos para um display onde cada segmento emite luz apenas quando na sua entrada correspondente é mantido o nível lógico "O".

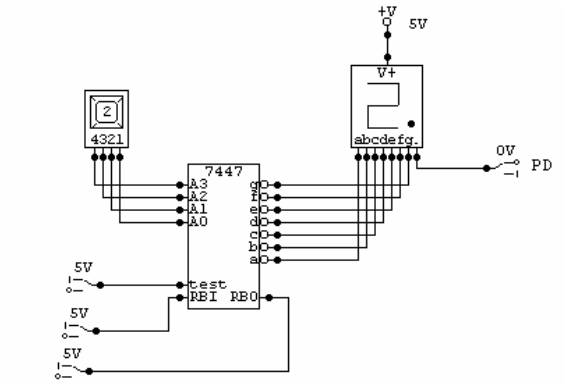
EP59. Laboratorio: usando o 7447





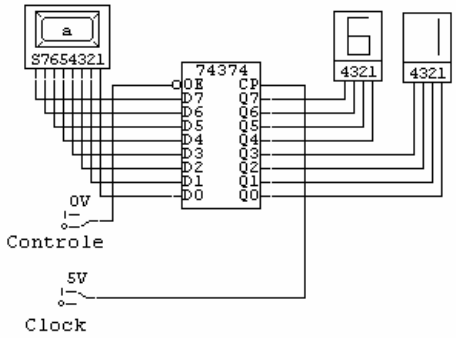
Indicador de chave/operação selecionada

EP60. Laboratório: A figura abaixo mostra um Decodificador BCD para 7 Segmentos. Faça o teste, preenchendo a Tabela e verifique o funcionamento do mesmo. Anote o que foi observado.



test	RBI	RBO	Observações (*o que ocorre com o display*)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

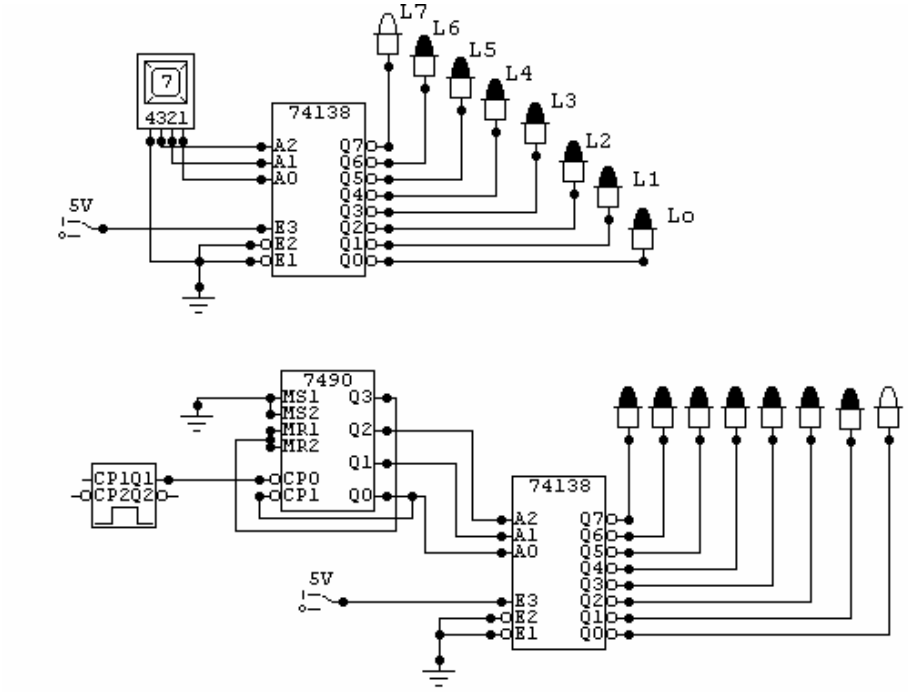
EP61. Laboratório: Decodificador ASCII – Hexadecimal



Escreva o código ASCII em hexa do seu nome, usando o circuito dado. Anote-o.

O CI 74138

EP62. Laboratório: Decodificador Como Selecionador De Dispositivos (74138): Implemente os circuitos abaixo e verifique seu funcionamento. Use LED's de cores diferentes para cada saída.

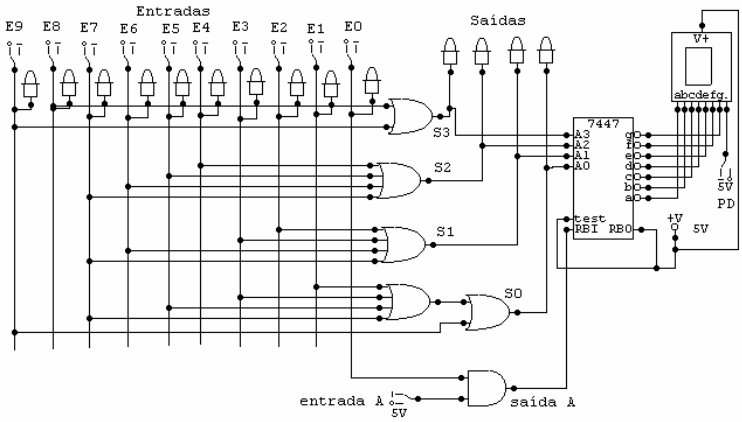


a)

Responda:

- Explique o principio de funcionamento do oscilador do circuito ponta lógica. Que componente garante que, quando na presença de sinal lógico, os leds não piscarão"? Porque?
- No ensaio do 74138, se desconectado o pino 4 do GND e ligado ao pino 6, o que ocorrerá? Porque?
- No ensaio do display: Qual a função dos pinos 3,4 e 5 do decodificador? Cite uma aplicação para cada um.

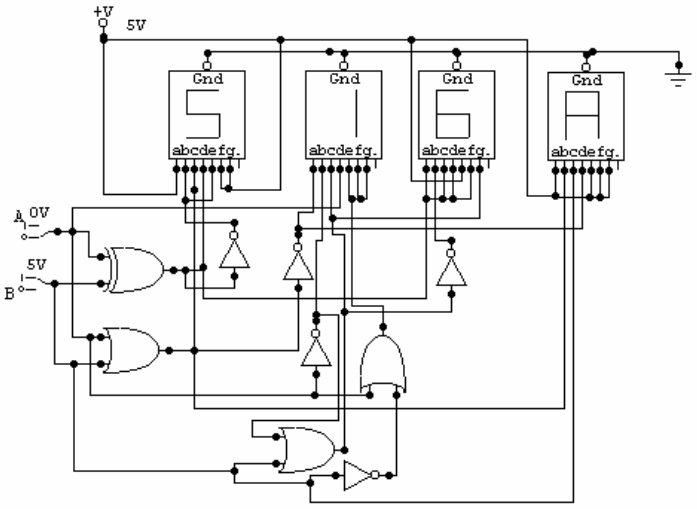
EP63. Laboratório: Decodificador 9876543210 - BCD – Display de 7 segmentos



Responda:

- Qual a função da entrada A?
- O que aconteceria se RBI fosse ligado diretamente a 5V ?
- Quais as expressões lógicas para S0, S1, S2 e S3?

EP64. Laboratório: Verifique o funcionamento do circuito abaixo:



Responda:

- Quais as palavras que aparecem nos displays, seguindo as seqüências para as entradas de controle A e B?
- Que letras aparecem no displays 1, 2, 3 e 4, respectivamente?
- Escreva a expressão para cada segmento dos displays, baseado na Tabela-Verdade, a seguir:

Entradas Display 1 Display 2 Display 3 Display 4

A	**B**	**a1**	**b 1**	**c1**	**d 1**	**e1**	**f1**	**g1**	**a2**	**b 2**	**c2**	**d 2**	**e2**	**f2**	**g2**	**a3**	**b 3**	**c3**	**d 3**	**e3**	**f3**	**g3**	**a4**	**b 4**	**c4**	**d 4**	**e4**	**f4**	**g4**
0	0	1	1	0	0	1	1	1	1	1	1	0	1	1	1	0	0	0	0	1	0	1	1	0	0	1	1	1	1
0	1	1	0	1	1	0	1	1	0	1	1	0	0	0	0	1	0	1	1	1	1	1	1	1	1	0	1	1	1
1	0	1	0	1	1	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	1
1	1	1	1	1	0	1	1	1	0	0	1	1	1	1	1	0	0	0	0	1	0	1	1	1	1	0	1	1	1

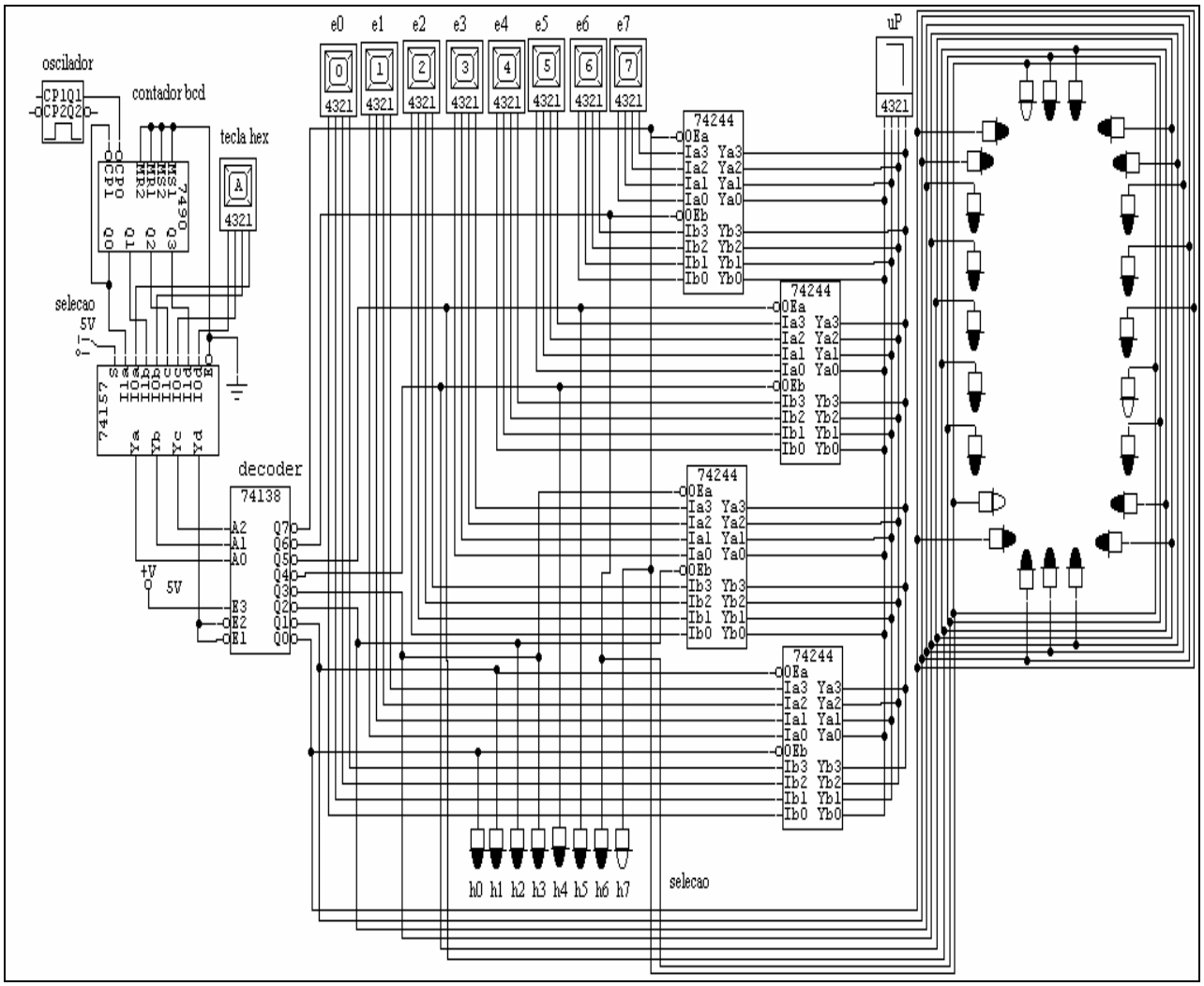
- Quantas palavras e quantas letras diferentes podemos formar se tivermos 4 entradas de controle?

EP65. Laboratório: Aplicações do decodificador CI 74138

A figura ao lado apresenta um circuito composto por:

- um contador que conta bcd ciclicamente;
- uma tecla hexa, que gera 4 bits;
- um mux 4 x 2:1 que através de chave permite que os sinais gerados pelo contador ou da chave hexa sejam apresentados em suas saídas;
- um decodificador CI 74138 que, quando habilitado, converte o código binário de 3 bits em um código octal ativo baixo;
- oito chaves hexa, as quais simulam dispositivos que fornecem dados de 4 bits cada;
- oito conjuntos de 4 x buffers, formados pelos CI's 74244);
- um display hexa, que simula um microprocessador ou dispositivo que recebe os dados;
- uma seqüencial composta de 3 x 8 lâmpadas.

Monte-o e descreva seu funcionamento



Multiplexadores e Demultiplexadores

EP.66 - Será possível desenvolver um mux de 8 entradas e 1 saída à partir de muxs de 2 entradas e 1 saída ? Justifique sua resposta.

EP.67 - Utilizando mux 4:1 (como "caixa preta") desenvolver um mux 8:1.

EP.68 - Apresente o circuito de um mux 2:1.

EP.69 - Utilizando-se circuitos tipo mux de 2:1, desenvolver um mux 4:1

EP.70 - Utilizando-se circuitos tipo mux de 2:1, desenvolver um mux 8:1

EP.71 - A partir de um mux 4:1 desenvolver um mux 2:1

EP.72 - A partir de um mux 8:1 desenvolver um mux 4:1

EP.73 - A partir de um mux 8:1 desenvolver um mux 2:1

EP.74 - Desenvolver um demux 1:2

EP.75 - Utilizando circuitos demux 1:2, desenvolver um demux 1:8

EP.76 - Conceitos de ULA. Utilizando circuitos somadores completos e multiplexadores 2:1 (como "caixa preta") desenvolver um circuito que soma e subtrai números inteiros com sinal de quatro bits cada. São componentes do circuito:

- sinal de entrada SOMA que comanda a operação de soma quando em "1" (B<0:3> + A<0:3>) e de subtração quando em "0" (B<0:3> - A<0:3>);
- a saída OVER que será "1" sempre que a soma ou subtração ocasionar "overflow" tanto na subtração quanto na adição.

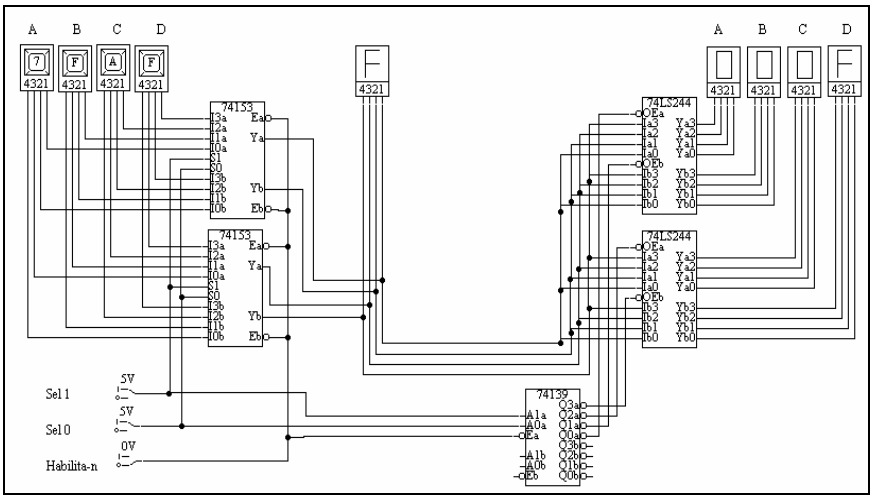
EP77. Laboratório. Utilizando o simulador, verificar o funcionamento do *Hex Display* estabelecendo diferentes níveis lógicos nas entradas. Apresentar dados e conclusões.

EP78. Laboratório. Implementar, utilizando o simulador, os circuitos Meio Somador e Somador Completo. Verificar o funcionamento dos circuitos, estabelecendo diferentes níveis lógicos nas entradas. Apresentar dados e conclusões.

EP79. Laboratório. Implementar, utilizando o simulador, os circuitos comparadores vistos. Verificar o funcionamento dos circuitos, estabelecendo diferentes níveis lógicos nas entradas. Apresentar dados e conclusões.

EP80. Laboratório. Implementar, utilizando o simulador, os circuitos Geradores e Verificadores de Paridade

EP81. Laboratório. Laboratório: MUX 16:4



Responda:

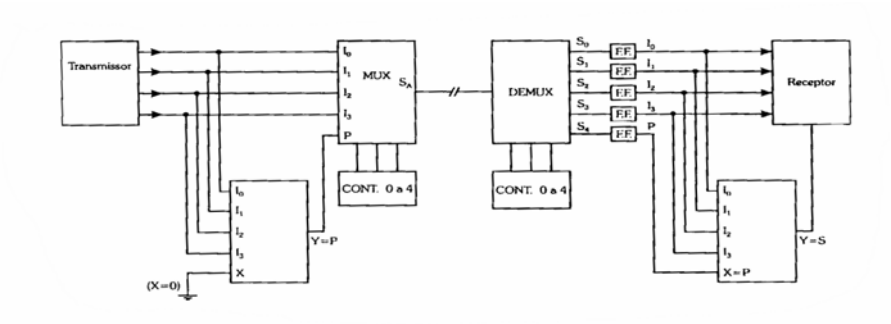
a) Qual o princípio de funcionamento do circuito que compreende os CI's 74153?

b) Qual o principio de funcionamento do circuito que compreende os CI's 74139 e os CI's 74244? Que tipo de circuito realizam?

c) o que realizam as chaves Sel_1 e Sel_0?

EP82. Laboratório. Projeto – Sistema De Transmissão E Recepção De Dados: A figura, a seguir, apresenta um sistema de transmissão e recepção de dados,, utilizando mux, demux, gerador / verificador de paridade e contadores para uma transmissão de uma palavra de 4 bits (Informação + bit de paridade).

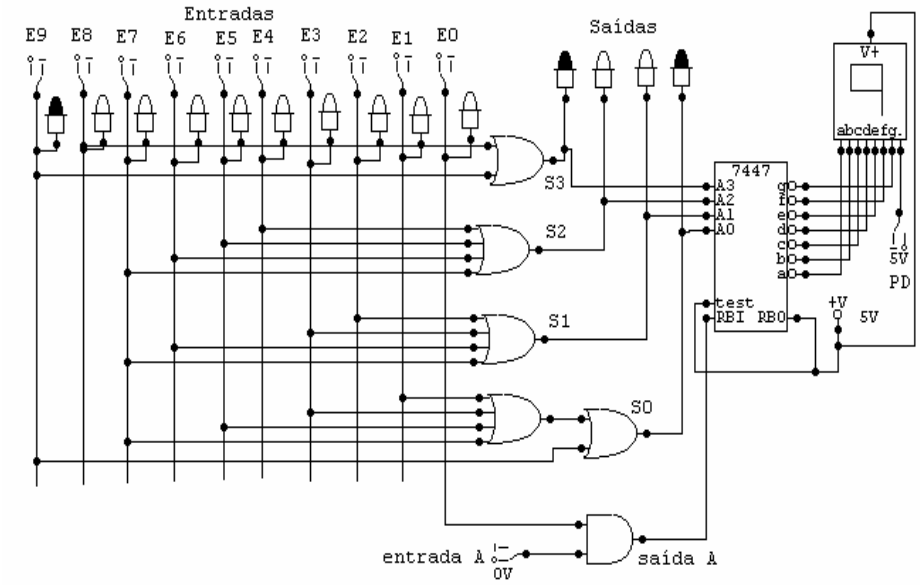
Para que o sistema funcione corretamente, é necessário que os contadores atuem de forma sincronizada e, ainda, que os FF's nas saídas do demux, através da devida sincronização dos pulsos de clock, armazenem gradativamente a informação recebida para ser entregue ao receptor e ao verificador de paridade. Na prática, além do bit de paridade, outros bits são transmitidos para a toda a sincronização do sistema. Baseado neste, implemente um sistema de transmissão e recepção de dados de 8 bits (7 bits de informação + 1 bit de paridade)



EP83. Multiplexadores e Demultiplexadores:

- Conceitue multiplexadores e demultiplexadores. Cite uma aplicação destes circuitos
- Utilizando MUX 2:1 e DEMUX 1:2 como diagramas de bloco, apresente:
  - MUX 4:1
  - MUX 8:1
  - MUX 16:1
  - DEMUX 1:4
  - DEMUX 1:8
  - DEMUX 1:16

EP84. Laboratório. Decodificador Decimal – BCD – 7 segmentos: Implemente, no simulador, o circuito abaixo.



1. Explique o funcionamento do circuito.
2. Encontre as expressões para as saídas S3, S2, S1 e S0.
3. Apresente a Tabela-Verdade que represente o funcionamento do circuito.
4. Associe a 1a. coluna com a 2a.

	codificação			conversão de "saída"
	saída 3-T			transistor bipolar aberto
	multiplexador			uma entrada, várias saídas
	decodificaão			várias entradas, uma saída
	saída "open collector"			controlada por "enable"
	demultiplexação			conversão de "entrada"
				saída "open drain"

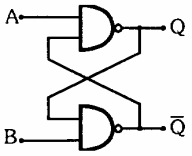
Lógica Seqüencial: Flip-Flops, Registradores e Contadores

EP.85 – Qual a diferença básica entre um circuito combinacional e um seqüencial ?

EP.86 - Responder:

- Como funciona um *flip flop* SR que não possui *clock* ?
- Como funciona um *flip flop* JK ?
- O que é um *flip flop master slave* (mestre escravo) ?
- Como funciona um *flip flop* D ?

EP.87 – Analisar o circuito do flip-flop abaixo e construir sua tabela-verdade, identificando a função das entradas A e B (utilize o programa simulador).



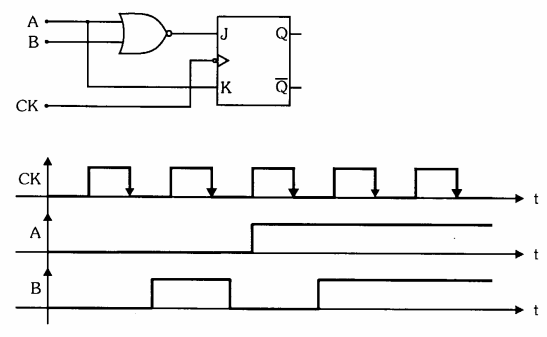
EP.88 – Qual a diferença básica entre um flip-flop assíncrono e um síncrono ?

EP.89 – É possível obter-se um flip-flop D a partir do flip-flop RS síncrono ? Justifique sua resposta.

EP.90 - A partir de um *flip flop* T desenvolva um *flip flop* JK

EP.91 - Apresentar um *FF master slave* (mestre escravo). Descrever seu princípio de funcionamento.

EP.92 – Determinar as formas de onda das entradas J e K e das saídas Q e $\overline{Q}$ do flip-flop do circuito seguinte, dadas as formas de onda de CK, A e B. Considerar inicialmente Q = 0.



EP.93 – Utilizando o programa simulador, verificar e descrever o funcionamento dos circuitos integrados TTL 7474 e 7476.

Registradores

EP.94 – Qual o objetivo de um circuito registrador ?

EP.95 – O que é um registrador de deslocamento ?

EP.96 – O que diferencia um registrador com entrada paralela de um registrador série ?

EP.97 – Utilizando o programa simulador, projetar um circuito registrador com 8 bits, entradas paralelas e entrada série à partir do CI 7495.

EP.98 - Utilizando o programa simulador, implementar circuitos que realizam a divisão e a multiplicação de um número por 2.

Contadores

EP.99 – Descreva, em linhas gerais, o que são contadores e ilustre aplicações para os mesmos.

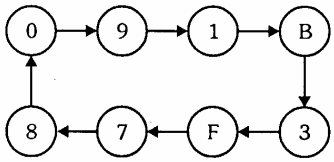
EP.100 – Diferencie contador assíncrono e síncrono ?

EP.101 – Quais as principais vantagens de um contador síncrono em relação ao assíncrono ?

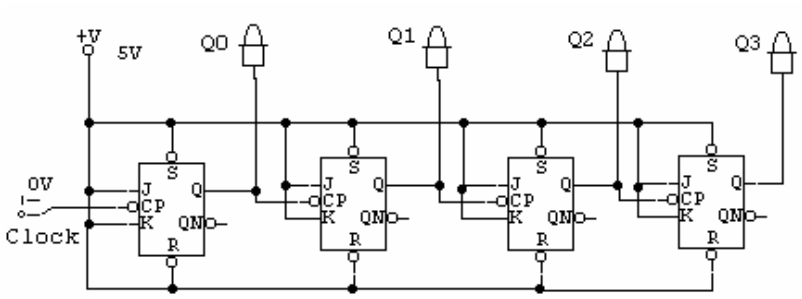
EP.102 – Para obtenção de um sinal com freqüência 32 KHz a partir de um sinal com 256 KHz, é necessário utilizar um contador com quantos bits ?

EP.103 – Desenhe o diagrama de tempos e o circuito correspondente de um Contador Hexadecimal Assíncrono Decrescente.

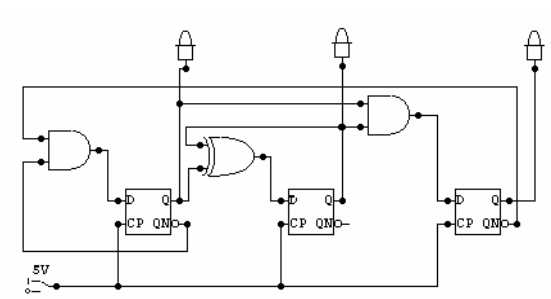
EP.104 – O diagrama de estados abaixo representa o funcionamento de um contador de números de quatro bits. Obter o seu circuito nos modos síncrono e assíncrono, utilizando apenas três flip-flops T.



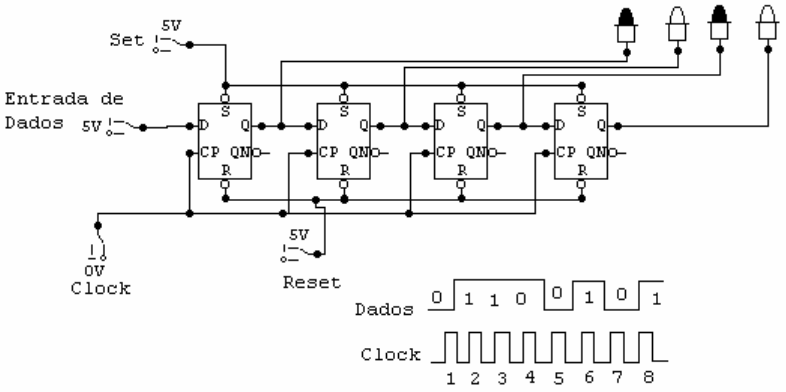
EP.105. Laboratório – Utilizando o programa simulador descreva o funcionamento do circuito abaixo.



EP.106. Laboratório – Utilizando o programa simulador descreva o funcionamento do circuito abaixo.

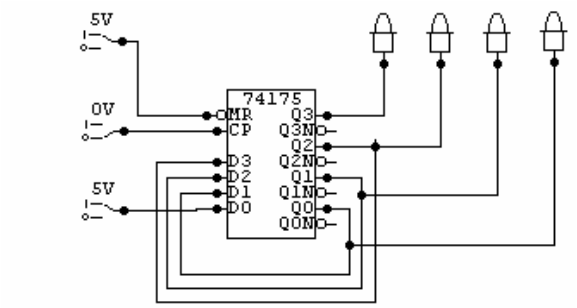


EP.107. Laboratório. Registrador De Deslocamento – Conversor Série-Paralelo. Implemente o circuito, a seguir, e verifique o que ocorre com a seqüência de pulsos quando dada.



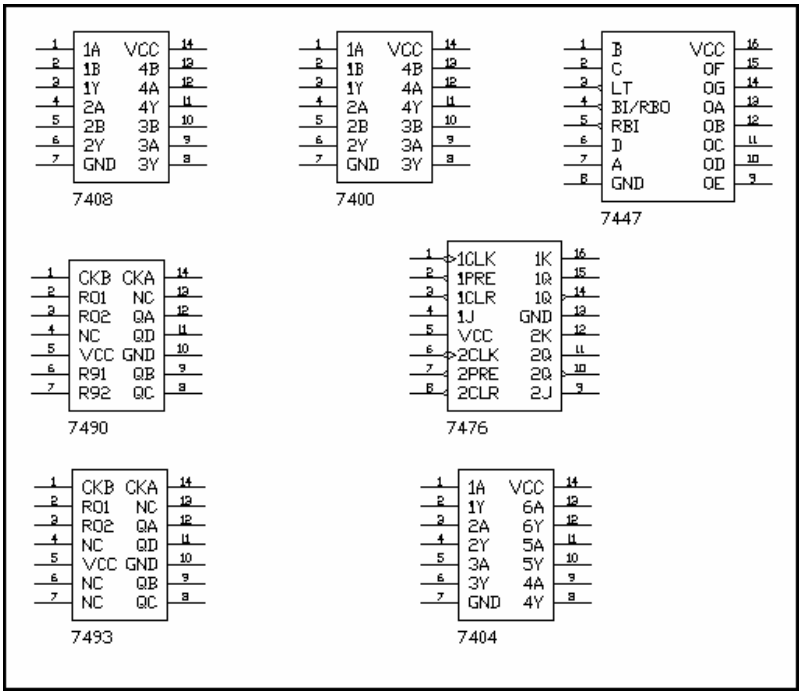
O que podemos observar nas saídas após o 4º pulso de clock? E após o 8º pulso?

EP108. Laboratório. Implemente o circuito abaixo e verifique o seu funcionamento. Aplique os pulsos do circuito anterior e veja o que acontece nas saídas. Compare com o circuito anterior.

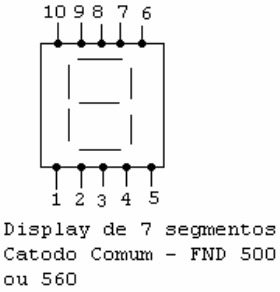


EP109. Laboratório. Flip-Flops Jk (Tipo T) – Contador Assíncrono: Implemente um contador de 00-59 e um de 00-23. Interligue-os para formar um relógio digital. Através do guia prático de como usar o CI-7490, use-o para implementar um relógio digital, contendo horas e minutos.

CIs úteis:

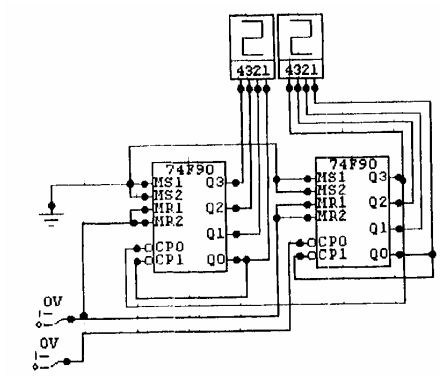


CI 7408 → Portas and de duas entradas (A e B – entradas e Y saída)
CI 7400 → Portas Nand de duas entradas (A e B – entradas e Y saída)
CI 7447 → Decodificador BCD para 7 segmentos
CI 7490 → Contador de Década CI 7493 → Contador de 0 até 15
CI 7476 → Flip- Flops JK (2 FF's no mesmo CI)
CI 7404 → Portas Inversoras (A – entrada e Y – saída)

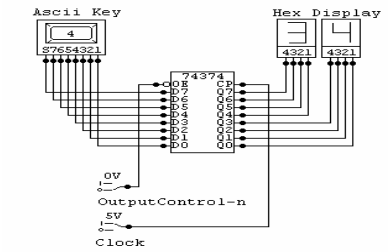


Pinos: 1, 2, 4, 6, 7, 9 e 10 → Segmentos E, D, C, B, A, F e G, respectivamente. 3 e 8 → Comum (Terra), 5 → Ponto Decimal

EP.110. Laboratório – Utilizando o programa simulador descreva o funcionamento do circuito abaixo.



EP111. Laboratório. Monte o circuito da Figura abaixo e descreva seu comportamento.



Verifique e apresente o código ASCII correspondente ao seu primeiro nome.

EP112. Laboratório. Descreva o princípio de funcionamento dos circuitos abaixo:

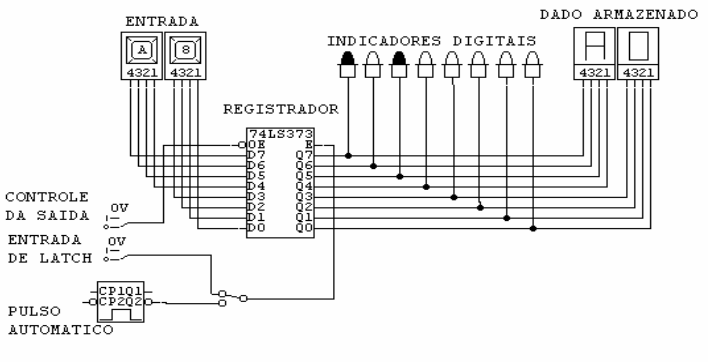


Fig. 1. Registrador com controle de saída (tri-state, ativo baixo) com armazenamento a nível (alto)

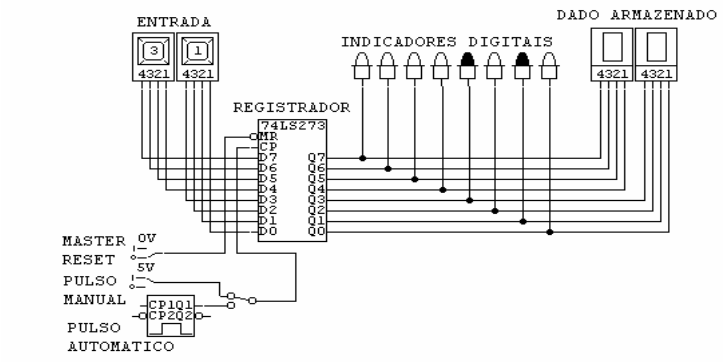
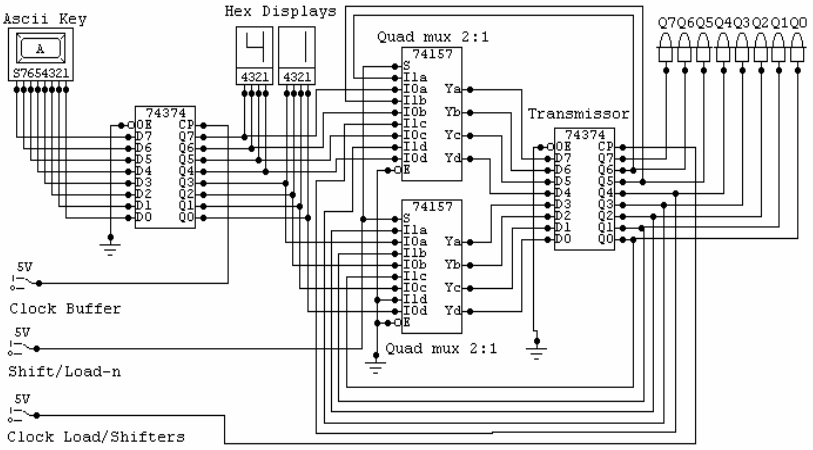


Fig.2. Registrador com controle de (master) reset (ativo baixo) com armazenamento à transição (positiva)
.

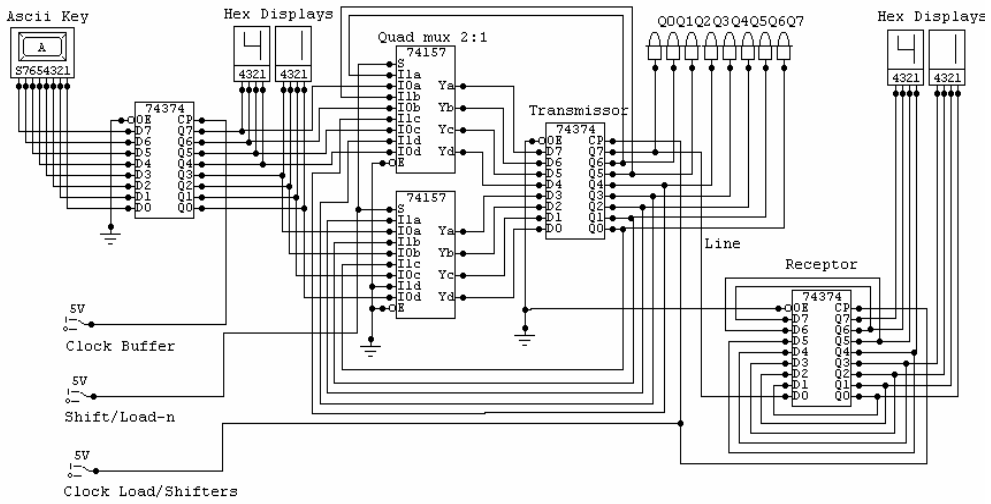
EP 113. Laboratório. Registrador de Deslocamento. Seja o circuito abaixo:



Sabendo que os CI's 784157 são compostos por 4 mux 2:1 controlados simultaneamente pelo sinal/chave Shift/Load-n :

a) Armazene no primeiro 74374 a primeira letra do seu primeiro nome, aplicando um pulso no "clock buffer"
b) Transfira esta informação para o outro 74374 (transmissor), colocando a chave Shift/Load-n em "0" (modo Load) e aplicando um pulso apenas em "Clock Load/Shifters"
c) Coloque a chave Shift/Load-n em "1" (modo Shift) e aplique <u>oito pulsos lentamente</u>. Observe o que ocorre a <u>após cada transição positiva</u>
d) Apresente suas conclusões.

EP114. Laboratório. Acrescente ao circuito anterior os novos componentes presentes na Figura, a seguir, e repita os passos a), b), c) e d) do item anterior.


Ascii Key
A
S7654321
Hex Displays
4321
4321
Quad mux 2:1
74157
S
I1a
I0a
I1b
I0b
I1c
I0c
I1d
I0d
OE
Ya
Yb
Yc
Yd
74374
OOE
D7
D6
D5
D4
D3
D2
D1
D0
CP
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0
Transmissor
Q0Q1Q2Q3Q4Q5Q6Q7
Hex Displays
Line
Receptor
5V
Clock Buffer
Shift/Load-n
Clock Load/Shifters