MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA CATARINENSE
CAMPUS LUZERNA

Sistemas Digitais

Professor: Ricardo Kerschbaumer

Aluno: _____

12 de outubro de 2023

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CATARINENSE

Sumário

Lista de figuras

Aula 1 - Conceitos introdutórios

Nesta aula serão apresentados alguns conceitos necessários ao entendimento dos conteúdos de sistemas digitais. A princípio serão apresentadas as diferenças entre sinais analógicos e sinais digitais. Em seguida serão apresentados os fundamentos das representações dos sinais digitais e as vantagens e desvantagens de sua utilização. Finalmente serão discutidos os circuitos digitais, seus aspectos elétricos, a conversão dos sinais entre o ambiente analógico e o digital e as formas de onda relacionadas.

1.1. Sistemas digitais

Os Sistemas digitais ou circuitos digitais, ou ainda circuitos lógicos são definidos como circuitos eletrônicos que empregam a utilização de sinais elétricos em apenas dois níveis de tensão de forma a definir uma representação de valores binária.

Os sinais elétricos transportados por estes circuitos são chamados de **sinais digitais** ou **sinais binários**.

1.1.1. Sinais digitais e sinais analógicos

Os sistemas digitais são sistemas no qual os sinais têm um número finito de valores discretos, normalmente dois, se contrapondo a sistemas analógicos onde os sinais têm valores pertencentes a um conjunto contínuo ou infinito de valores.

A Figura 1 apresenta um gráfico da tensão em função do tempo, representando um sinal analógico.

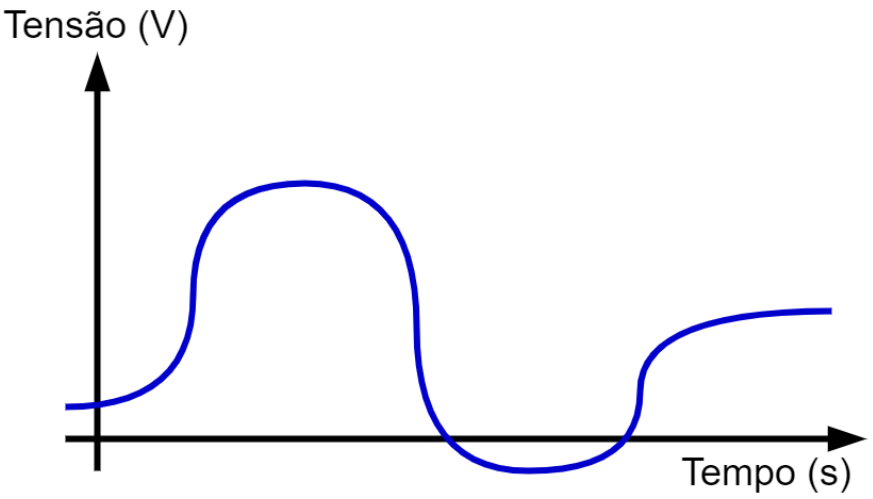


Figura 1 - Sinal analógico.

Nesta figura é possível observar que a amplitude do sinal pode assumir qualquer valor.

A Figura 2 por sua vez apresenta um gráfico da tensão em função do tempo, representando um sinal digital.

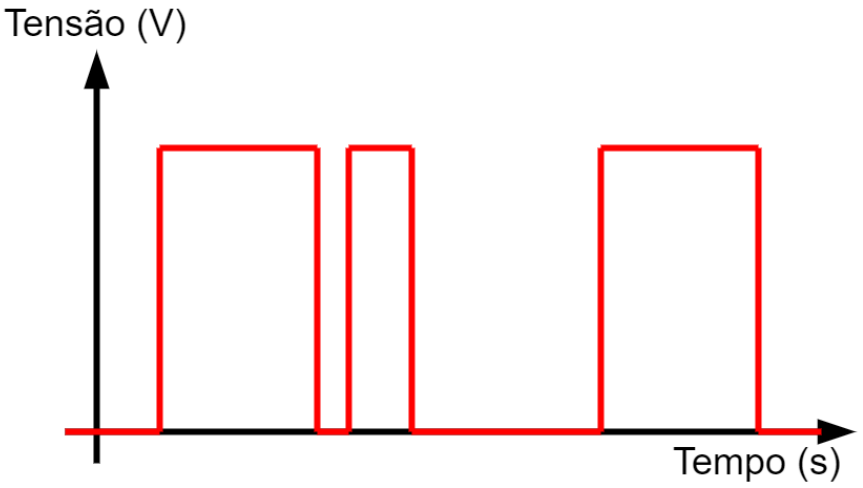


Figura 2 - Sinal digital.

Nesta figura é possível observar que o sinal pode assumir apenas dois valores distintos, um com tensão mais baixa e outro com tensão mais alta.

1.1.2. Representação de sinais digitais

Como já mencionado, sinais digitais costumas assumir dois estados. Assim é possível utilizar sinais digitais para representar fenômenos com as mesmas características. Por exemplo, um sinal digital pode ser utilizado para representar o estado de uma lâmpada, ligada ou desligada.

É comum dizermos que um sinal digital, ou seja, um sinal binário, pode assumir dois estados lógicos. Estes estados têm várias denominações, ligado e desligado, ou alto e baixo, ou ainda verdadeiro e falso. Utilizaremos a seguinte notação para representar os estados de um sinal digital.

O digito 0 (zero) representa o valor falso ou baixo, enquanto o digito 1 (um) representa o valor verdadeiro ou alto.

1.1.3. Porque usar sinais digitais

Os sinais digitais e consequentemente os sistemas digitais são utilizados por apresentarem algumas vantagens sobre os sinais analógicos. As principais vantagens dos sinais digitais são.

- Os sinais digitais são muito mais imunes a distorções, ruídos e interferências.
- Os circuitos digitais são mais confiáveis e robustos.
- Os circuitos digitais são fáceis de projetar e mais baratos.
- A implementação de hardware em circuitos digitais é mais flexível.

1.2. Grandezas digitais

Um único sinal digital pode assumir apenas dois estados, como já mencionado. Porém para a maioria das aplicações, dois estados não são suficientes. Imagine que se deseja contar o número de pessoas em uma sala, seria necessária uma quantidade muito maior de estados para representar este número. Para contornar este tipo de situação utiliza-se não apenas um, mas vários sinais

digitais para representar uma determinada grandeza. Neste contexto surgem diferentes nomenclaturas para determinados números de sinais binários agrupados. A nomenclatura utilizada para os sinais digitais é o **bit** (***Bi**nary digit*). A seguir são apresentadas as principais notações utilizadas para grupos de bits.

- Um sinal composto por apenas um **bit** é um sinal binário único.
- Um sinal composto por quatro Bits é chamado de **nibble**.
- Um sinal composto por oito Bits é chamado de **byte**.
- Um sinal composto por dezesseis Bits é chamado de **word**.

A Figura 3 mostra graficamente a relação entre os diferentes grupos de Bits.

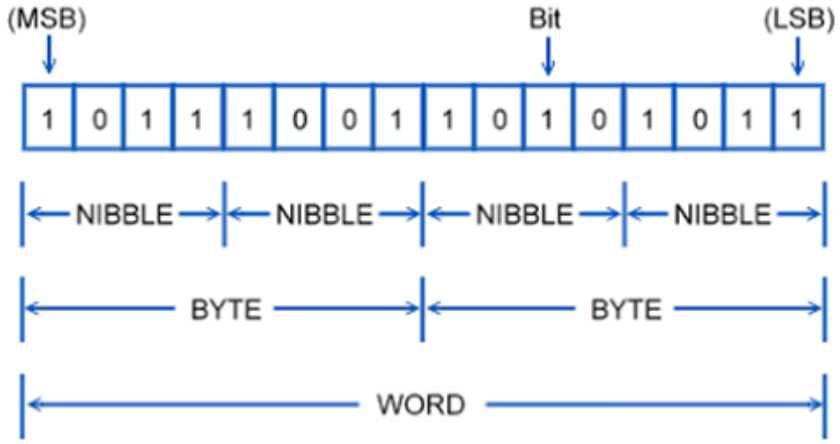


Figura 3 - Grandezas digitais

Quando se trata do armazenamento de informações digitais é comum a utilização de múltiplos para representar grandes quantidades de bits. A seguir são apresentados alguns múltiplos comumente utilizados.

- 1 kilobyte (KB) = 1024 bytes
- 1 megabyte (MB) = 1024 kilobytes
- 1 gigabyte (GB) = 1024 megabytes
- 1 terabyte (TB) = 1024 gigabytes
- 1 petabyte (PB) = 1024 terabytes

Apesar de muito utilizada esta notação não é totalmente correta, pois foi desenvolvida para numeração de base 10 e não para representação binária. Assim, o múltiplo k (kilo), por exemplo, deveria representar 1000 e não 1024. Para evitar confusões foi desenvolvida uma outra nomenclatura especificamente para sistemas digitais. A Figura 4 apresenta os múltiplos de byte.

Profixo Binário (IEC)			Profixo do SI		
Nome	**Símbolo**	**Múltiplo**	**Nome**	**Símbolo**	**Múltiplo**
kibibyte	KiB	2^{10}	kilobyte	kB	10^{3}
mebibyte	MiB	2^{20}	megabyte	MB	10^{6}
gibibyte	GiB	2^{30}	gigabyte	GB	10^{9}
tebibyte	TiB	2^{40}	terabyte	TB	10^{12}
pebibyte	PiB	2^{50}	petabyte	PB	10^{15}

Figura 4 - Múltiplo de byte

1.2.1. Contando em binário

Existem várias formas de representar as informações em formato digital, apenas para ilustrar este conceito a Tabela 1 apresenta a representação dos números de 0 a 9 na forma de bits.

Tabela 1 - Números na forma binária

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

Estas representações serão assunto das próximas aulas.

1.3. Circuitos digitais

Circuitos digitais são a implementação de sistemas digitais na forma de circuitos eletrônicos, onde os sinais digitais são representados por sinais elétricos.

1.3.1. Aspectos elétricos

Para a construção de circuitos eletrônicos digitais é necessário implementar os sinais lógicos 0 e 1 na forma de sinais elétricos. Existem várias formas de fazer isso, as principais serão estudadas mais a frente. A título de exemplo a Figura 5 apresenta os níveis de tensão dos estados lógicos em um sistema digital de 5 V.

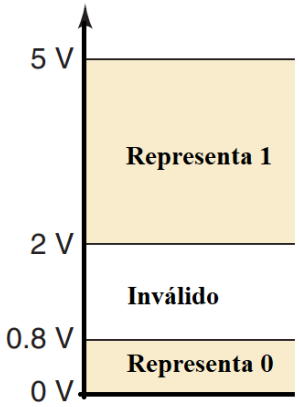


Figura 5 - Níveis de tensão

É comum dizermos que o nível lógico 0 é representado por 0 V e o nível lógico 1 é representado por 5 V, mas na prática é um pouco mais complicado. Como pode ser observado na

Figura 5, existem faixas de tensão que representam cada um dos níveis. Isso torna os sistemas digitais mais robustos e imunes a interferências.

Para simplificar os sinais são representados como na Figura 6.

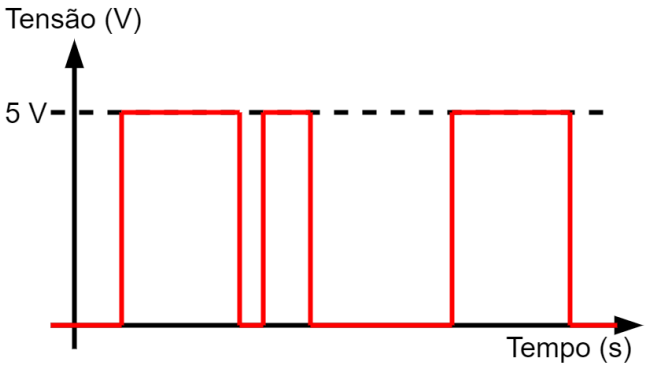


Figura 6 - Sinal digital típico.

Existem muitas outras tecnologias de implementação de circuitos digitais, mas geralmente o nível 0 é representado por uma tensão baixa (perto de 0 V) e o nível 1 é geralmente representado por uma tensão mais alta.

1.3.2. Conversores A/D e D/A

Uma vez que os sinais do mundo físico são analógicos, é necessário convertê-los para sinais digitais e vice-versa sempre que os sinais digitais tenham que interagir com os sinais do meio físico.

A conversão de sinais analógicos em sinais digitais é realizada por um dispositivo eletrônico chamado conversor analógico digital (A/D). A Figura 7 mostra o diagrama de um conversor analógico digital de 8 bits.

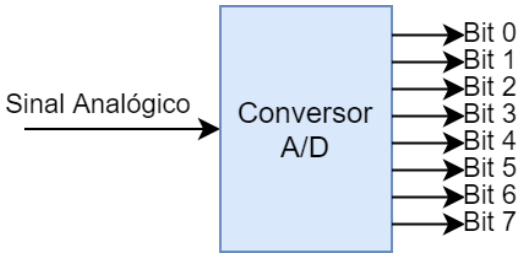


Figura 7 - Conversor Analógico Digital.

Da mesma forma, a conversão de sinais digitais em sinais analógicos é realizada por um dispositivo eletrônico chamado conversor digital analógico (D/A). A Figura 8 mostra o diagrama de um conversor digital analógico de 8 bits.

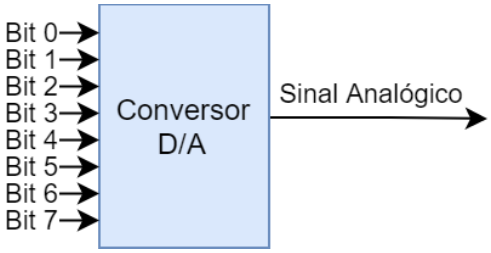


Figura 8 - Conversor Digital Analógico

A maioria dos sistemas digitais é utilizado para processar sinais oriundos de sensores analógicos, como por exemplo, microfones, sensores de temperatura, sensores de luminosidade etc. Assim, estes sinais são primeiro convertidos de analógicos para digitais para só então serem processados. Este processamento é então realizado por circuitos digitais projetados para este fim. Os resultados deste processamento são também sinais digitais, que devem então ser convertidos em sinais analógicos, utilizando um conversor digital analógico. Os sinais analógicos resultantes podem então ser enviado para atuadores, como por exemplo, alto falantes, motores etc.

A Figura 9 apresenta um diagrama de blocos de um sistema de processamento digital de sinais analógicos.

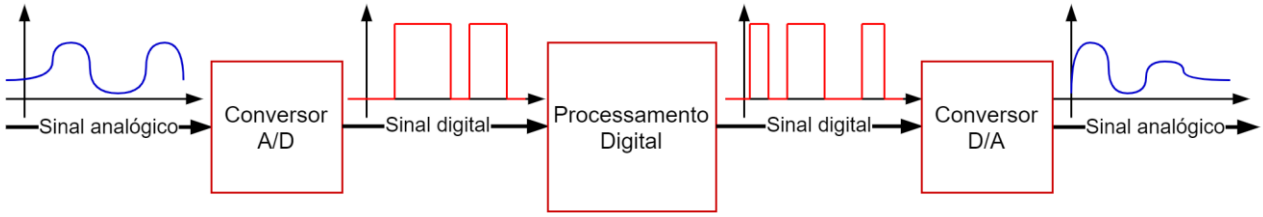


Figura 9 - Processamento digital de sinais

1.3.3. Formas de onda digitais

Nos estudos de sistemas digitais é comum a utilização de gráficos que representam as formas de onda de sinais digitais. Destes gráficos é possível obter várias informações importantes, como a amplitude do sinal, o período e a frequência. A Figura 10 apresenta um gráfico típico de um sinal digital.

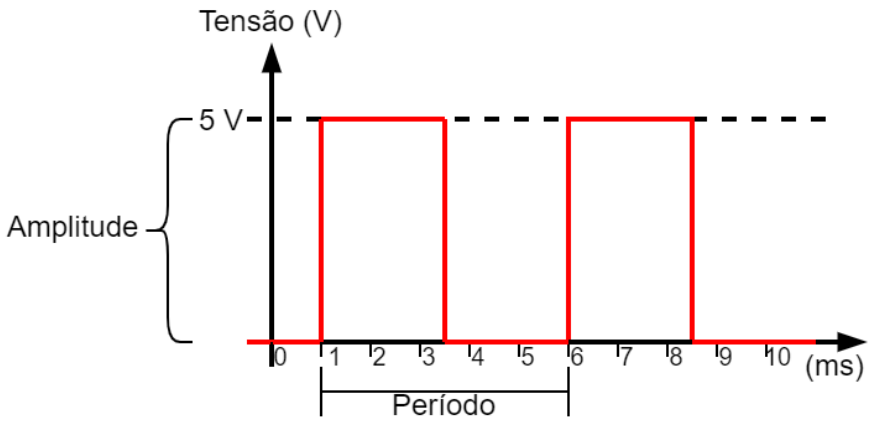


Figura 10 - Forma de onda de sinal digital

No eixo vertical é possível verificar a **amplitude** deste sinal, neste caso 5 V. No eixo horizontal é apresentada a escala de tempo, neste caso em milissegundos. Se o sinal apresentado no gráfico for um sinal repetitivo é possível observar também seu **período**. O período de um sinal é o tempo que ele leva para se repetir. No gráfico da figura p período é de 5 ms ou 0,005 s. Em função do período pode-se calcular a **frequência** do sinal através da seguinte formula.

$$F = \frac{1}{P}$$

Onde *F* é a frequência e *P* o período. No exemplo da figura a frequência é de 200 Hz.

1.4. Exercícios

1) Dado o sinal digital da Figura 11 determine a amplitude, o período e a frequência.

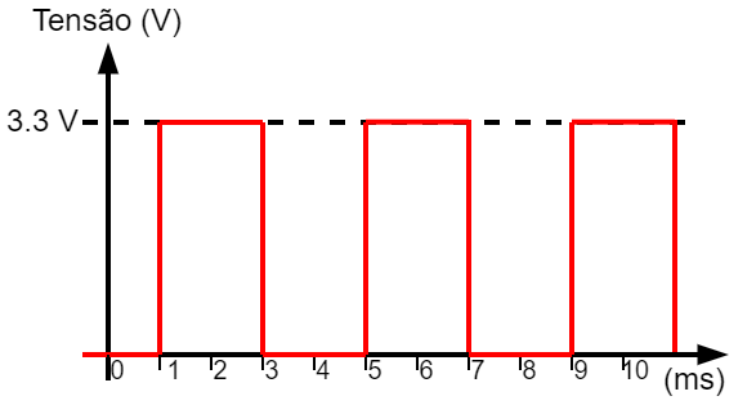


Figura 11 - Forma de onda do exercício 1.

a) Amplitude = _____
b) Período = _____
c) Frequência = _____

Aula 2 - Códigos e sistemas numéricos

Nesta aula serão abordados assuntos relacionados a forma com que os sistemas digitais codificam as informações. Inicialmente serão abordados os sistemas de numeração, enfatizando o sistema binário, o octal, o hexadecimal e o BCD. Na sequência estudaremos os métodos de conversão entre os diversos sistemas de numeração. O assunto seguinte é a representação de números negativos. Por fim estudaremos a representação de números de ponto flutuante e os códigos alfanuméricos.

2.1. Sistemas de numeração

Os sistemas de numeração são formas de representar os números. Cada sistema de numeração possui um conjunto específico de caracteres. O número de caracteres de cada conjunto é chamado de base do sistema de numeração. A seguir serão apresentados os principais sistemas de numeração relacionados aos sistemas digitais.

2.1.1. Sistema decimal

O sistema decimal é o mais conhecido sistema de numeração e é amplamente empregado em todo o mundo. Este sistema utiliza um conjunto de 10 caracteres ou símbolos (0, 1, 2, 3, 4, 5, 6, 7, 8 e 9) para representar ou números. Estes caracteres são também chamados de dígitos. Como são 10 caracteres que compõe a base do sistema decimal, este sistema é também conhecido como sistema de base 10.

Neste sistema cada dígito possui um peso relacionado a uma potência de sua base, neste caso, a base 10. A Figura 12 apresenta a representação do número 5432,789 na base 10.

10^3	10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-3}
5	4	3	2 ,	7	8	9

Figura 12 - Sistema de numeração de base 10.

É possível observar que a direita da vírgula as potências da base são negativas, sinalizando pesos menores do que 1. Já a direita da vírgula as potências da base são positivas. O peso de cada uma das posições ocupadas pelos dígitos é a base elevada a potência relacionada a sua posição.

Para a determinação do valor de um número na base 10 deve se fazer a soma de cada um de seus dígitos multiplicado pelo peso da posição onde ele se encontra. Veja o exemplo a seguir.

$$5 * 10^3 + 4 * 10^2 + 3 * 10^1 + 2 * 10^0 + 7 * 10^{-1} + 8 * 10^{-2} + 8 * 10^{-3} = 5432,789$$

2.1.2. Sistema binário

O sistema binário segue a mesma lógica, porém ele utiliza um conjunto de apenas 2 caracteres ou símbolos (0 e 1) para representar ou números. Assim este sistema de numeração é chamado de sistema de base 2.

Neste sistema de numeração os pesos dos dígitos são todos potências de 2. A Figura 13 apresenta a representação do número 1000110 na base 2. Quando a base de um número é diferente da base 10 é comum colocar-se o número da base subscrita no final do número, assim o número anterior seria 1000110_2.

2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	16	8	4	2	1
1	0	0	0	1	1	0

Figura 13 - Sistema de numeração de base 2.

Para a determinação do valor de um número em na base 2 se deve fazer a soma de cada um de seus dígitos (0 ou 1) multiplicado pelo peso da posição onde ele se encontra. Veja o exemplo a seguir.

$$1000110_2 = 1 * 2^6 + 0 * 2^5 + 0 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0 = 70_{10}$$

Quando se utiliza apenas números inteiros, sem a parte fracionária, o maior número que pode ser representado por um conjunto de bits pode ser determinado pela seguinte expressão.

$$Maior\ Valor = 2^{número\ de\ bits} - 1$$

Assim como no sistema decimal, o sistema binário pode utilizar dígitos a direita da vírgula. A Figura 14 apresenta um exemplo.

2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}
16	8	4	2	1	0,5	0,25
1	0	1	0	1 ,	1	1

Figura 14 - Número fracionário na base 2

A determinação do valor de um número nestas condições segue a mesma lógica dos anteriores, veja a seguir.

$$10101,11_2 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 1 * 2^{-1} + 1 * 2^{-2} = 21,75_{10}$$

O sistema de numeração binário é o sistema utilizado nos sistemas digitais, daí sua grande importância para nossos estudos.

2.1.3. Sistema octal

O sistema octal possui este nome porque utiliza um conjunto de 8 caracteres ou símbolos (0, 1, 2, 3, 4, 5, 6 e 7) para representar ou números. Assim este sistema de numeração é chamado de sistema de base 8.

Assim como nos sistemas de numeração anteriores, os pesos dos dígitos são todos potências da base, neste caso, 8. A Figura 15 apresenta a representação do número 4205,47 na base 8.

8^3	8^2	8^1	8^0	8^{-1}	8^{-2}
4	2	0	5 ,	4	7

Figura 15 - Sistema de numeração de base 8.

Para a determinação do valor de um número em na base 8 deve-se, assim como nos anteriores, fazer a soma de cada um de seus dígitos multiplicado pelo peso da posição onde ele se encontra. Veja o exemplo a seguir.

$$4205{,}47_8 = 4 * 8^3 + 2 * 8^2 + 0 * 8^1 + 5 * 8^0 + 4 * 8^{-1} + 7 * 8^{-2} = 2181{,}609375_{10}$$

O sistema octal é importante para os estudos de sistemas digitais pois os números (0 a 7) podem ser representados por 3 bits, assim a representação binária de números em formato octal é facilitada. A Figura 16 apresenta como cada número em octal pode ser representado por 3 bits.

4			2			0			5			4			7		
1	0	0	0	1	0	0	0	0	1	0	1	1	0	0	1	1	1

Figura 16 - Relação entre octal e binário.

2.1.4. Sistema hexadecimal

O sistema hexadecimal é parecido com o sistema octal, porém utiliza 4 bits cara cada caractere. Assim, o sistema hexadecimal utiliza um conjunto de 16 caracteres ou símbolos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F) para representar ou números. Assim este sistema de numeração é chamado de sistema de base 16. Neste sistema as letras A, B, C, D, E e F correspondem respectivamente aos números decimais 10, 11, 12, 13, 14 e 15, ou seja, A=10, B=11, C=12, D=13, E=14 e F=15.

Assim como nos sistemas de numeração anteriores, os pesos dos dígitos são todos potências da base, neste caso, 16. A Figura 17 apresenta a representação do número 3A4,F2 na base 16.

16^2	16^1	16^0	16^{-1}	16^{-2}
3	A	4 ,	F	2

Figura 17 - Sistema de numeração de base 16.

Para a determinação do valor de um número em na base 16 deve-se, assim como nos anteriores, fazer a soma de cada um de seus dígitos multiplicado pelo peso da posição onde ele se encontra. Veja o exemplo a seguir.

$$3A4{,}F2_{16} = 3 * 16^2 + A * 16^1 + 4 * 16^0 + F * 16^{-1} + 2 * 16^{-2} = 932{,}9453125_{10}$$

O sistema hexadecimal é importante para os estudos de sistemas digitais pois os números (0 a F) podem ser representados por 4 bits, assim a representação binária de números em formato hexadecimal é facilitada. A Figura 18 apresenta como cada número em hexadecimal pode ser representado por 4 bits.

Hexadecimal	3				A				4				F				2			
Decimal	3				10				4				15				2			
Binário	0	0	1	1	1	0	1	0	0	1	0	0	1	1	1	1	0	0	1	0

Figura 18 - Relação entre hexadecimal, decimal e binário.

Para relacionar os diferentes sistemas de numeração, a Figura 19 mostra uma tabela comparativa dos números de 0 a 20.

Decimal	Binário	Octal	Hexa
0	0	0	0
1	1	1	1
2	1 0	2	2
3	1 1	3	3
4	1 0 0	4	4
5	1 0 1	5	5
6	1 1 0	6	6
7	1 1 1	7	7
8	1 0 0 0	1 0	8
9	1 0 0 1	1 1	9
10	1 0 1 0	1 2	A
11	1 0 1 1	1 3	B
12	1 1 0 0	1 4	C
13	1 1 0 1	1 5	D
14	1 1 1 0	1 6	E
15	1 1 1 1	1 7	F
16	1 0 0 0 0	2 0	1 0
17	1 0 0 0 1	2 1	1 1
18	1 0 0 1 0	2 2	1 2
19	1 0 0 1 1	2 3	1 3
20	1 0 1 0 0	2 4	1 4
Pesos →	16 8 4 2 1	8 1	16 1

Figura 19 - Comparativo entre as bases.

2.1.5. Sistema BCD

O sistema ou código BCD (do inglês ***B**inary **C**oded **D**ecimal*) é uma forma diferente de codificar números decimais em binário. Nesta codificação, cada dígito de um número decimal é codificado separadamente por uma combinação de 4 bits. A seguir são apresentados dois exemplos de codificação no sistema BCD.

37 = “0011” “0111”

459 = “0100” “0101” “1001”

É importante salientar que o sistema BCD se diferencia do sistema binário convencional porque separa os bits em grupos de 4.

2.1.6. Código Gray

O código Gray é também uma forma binária de representa números, porém tem o diferencial de possuir uma distância unitária entre os números. A Figura 20 apresenta os números de 0 a 9 representados através do código Gray.

Decimal	Gray
0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 0 1 0
4	0 1 1 0
5	0 1 1 1
6	0 1 0 1
7	0 1 0 0
8	1 1 0 0
9	1 1 0 1

Figura 20 - Código Gray

Observando a figura é possível notar que de um número para outro apenas um bit muda. Este código é mais utilizado em sistemas eletromecânicos, onde a comutação das chaves consome mais energia e produz ruídos. Assim, o uso do código Gray garante que qualquer mudança altera apenas um bit, minimizando o consumo de energia e o ruído.

2.2. Conversão entre bases

É comum que necessitemos converter um número em uma determinada base em seu valor equivalente em outra base. Nas seções anteriores foi apresentado que para converter um número em qualquer base para a base decimal, deve-se fazer a soma de cada um de seus dígitos multiplicado pelo peso da posição onde ele se encontra. As seções a seguir apresentas os procedimentos para realizar as conversões de números na base decimal para outras bases.

2.2.1. Converter inteiro decimal para outras bases

Para converter números inteiros da base decimal para qualquer outra base se utiliza um método chamado método de divisões sucessivas (DS). O funcionamento deste método é simples, basta ir dividindo sucessivamente o número inteiro decimal pela base (b) que se deseja utilizar. O resultado é a composição de todos os restos parciais das divisões. A seguir são apresentados os passos para a aplicação do método.

1) Efetue a divisão do número inteiro decimal (N) pela base (b) de forma a obter o quociente (Q_1) e o resto (R_1). O resto (R_1) e o quociente (Q_1) devem ser colocados respectivamente embaixo do número inteiro decimal (N) e da base (b), veja a Figura 21.

$$\begin{array}{c|c} N & b \\ \hline R_1 & Q_1 \end{array}$$

Figura 21 - Primeiro passo da divisão sucessiva.

2) Repita o passo anterior dividindo o quociente (Q_1) pela base (b) para obter o quociente (Q_2) e o resto (R_2), veja a Figura 22.

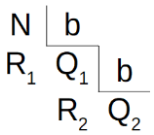


Figura 22 - Segundo passo da divisão sucessiva.

Este processo deve ir se repetindo até que seja encontrado um quociente (Q_n) com valor menor do que a base (b), veja a Figura 23.

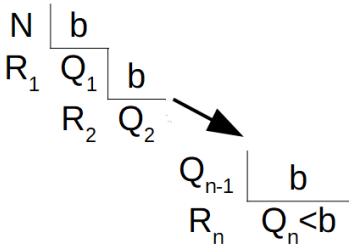


Figura 23 - Conclusão do método da divisão sucessiva.

A obtenção do resultado acontece da seguinte forma. O número inteiro na base b (I_b) é obtido do último quociente (Q_n) e dos restos obtidos nas divisões sucessivas conforma a expressão a seguir.

$$\mathbf{I_b = (Q_n\ R_n\ R_{n-1}\ R_{n-2}\ \ldots\ R_1)_b}$$

Para facilitar o entendimento do método, vejamos alguns exemplos. A Figura 24 apresenta um exemplo de conversão de um inteiro decimal para a base 2 (binário).

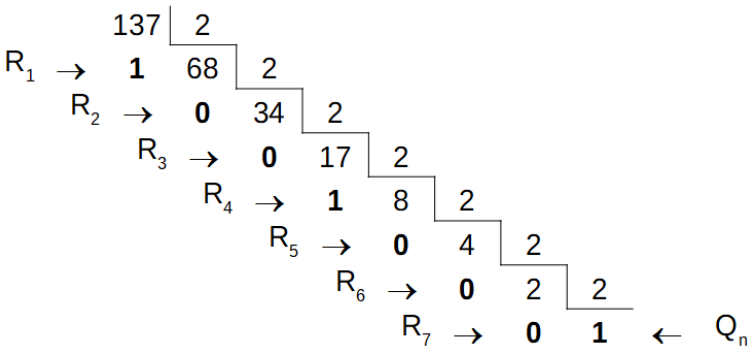


Figura 24 - Exemplo 1 de divisão sucessiva.

Neste exemplo o número inteiro decimal 137_{10} é convertido para a base 2. O valor binário resultado é 10001001_2. Para comprovar este resultado pode-se fazer a soma de cada um dos dígitos multiplicado pelo peso da posição onde ele se encontra, veja a Figura 25.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	0	0	0	1	0	0	1

Figura 25 - Verificação do resultado da conversão.

Observando a informação em formato binário e os respectivos pesos obtém-se a seguinte expressão, onde o resultado pode ser comprovado.

$$10001001_2 = 1 * 2^7 + 0 * 2^6 + 0 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0 = 137_{10}$$

Para um segundo exemplo faremos a conversão do número 9418_{10} para a base 16 (Hexadecimal). Veja a Figura 26.

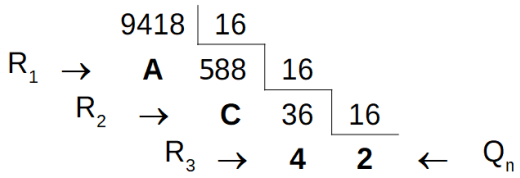


Figura 26 - Exemplo 2 de divisão sucessiva.

O número inteiro decimal 9418_{10} é convertido para a base 16 fica $24CA_{16}$.

Para comprovar este resultado pode-se fazer a soma de cada um dos dígitos multiplicado pelo peso da posição onde ele se encontra, veja a Figura 27.

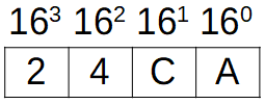


Figura 27 - Verificação do resultado da conversão.

Observando a informação em formato hexadecimal e os respectivos pesos obtém-se a seguinte expressão, onde o resultado pode ser comprovado.

$$24CA_{16} = 2 * 16^3 + 4 * 16^2 + C * 16^1 + A * 16^0 = 9418_{10}$$

É importante salientar que este método se aplica a números inteiros, em caso de números com parte fracionária, outros métodos devem ser empregados.

2.3. Números negativos

Os sistemas digitais também devem ser capazes de operar com números negativos, assim é necessária uma forma de representar digitalmente estes números.

A seguir serão apresentados três métodos de representação de números negativos na forma de bits.

2.3.1. Sinal e magnitude

A primeira forma de representar digitalmente números negativos é através do sinal e da magnitude, também conhecida como sinal e módulo. Nesta representação, um bit adicional é adicionado a esquerda do número, em sua forma binária tradicional, para representar o sinal. Assim, se este bit adicional for 0 o número é positivo, e se for 1 o número é negativo.

A Figura 28 apresenta os números 11_{10}, -11_{10}, 9_{10} e -9_{10} no formato de sinal e magnitude. Observe que o primeiro bit representa o sinal. Quando este bit é 1 o número é negativo.

		Sinal ↓	Magnitude			
11_{10}	=	**0**	1	0	1	1
-11_{10}	=	**1**	1	0	1	1
9_{10}	=	**0**	1	0	0	1
-9_{10}	=	**1**	1	0	0	1

Figura 28 - Representação por sinal e magnitude.

A notação de sinal e magnitude é a forma mais simples de representação de números negativos, porém esta representação não favorece a realização de operações aritméticas com estes números. Esta notação também possui duas representações para o zero o 0_2 e o -0_2.

2.3.2. Complemento de um

Complemento de um é outra forma de representar números negativos na forma binária. Nesta representação a combinação de bits que representa o valor negativo é obtida aplicando-se o complemento (inversão) bit a bit no valor positivo representado em sinal e magnitude. A Figura 29 apresenta os números 11_{10}, -11_{10}, 9_{10} e -9_{10} no formato de complemento de um. Observe que para números negativos todos os bits são invertidos, inclusive o sinal.

		Sinal ↓	Magnitude			
11_{10}	=	**0**	1	0	1	1
-11_{10}	=	**1**	0	1	0	0
9_{10}	=	**0**	1	0	0	1
-9_{10}	=	**1**	0	1	1	0

Figura 29 - Representação em complemento de um.

A representação em complemento de um facilita a construção de circuitos aritméticos digitais, porém ainda tem o inconveniente de duas representações para o valor zero, o 0_2 e o -0_2.

2.3.3. Complemento de dois

A representação de números negativos em formato binário mais comum nos sistemas digitais atuais é o complemento de dois. Nesta notação o valor negativo é obtido aplicando-se o complemento (inversão) bit a bit no valor positivo representado em sinal e magnitude, em seguida soma-se 1 a este valor. A Figura 30 apresenta os números 11_{10}, -11_{10}, 9_{10} e -9_{10} no formato de complemento de dois.

		Sinal ↓	Magnitude			
11_{10}	=	**0**	1	0	1	1
-11_{10}	=	**1**	0	1	0	1
9_{10}	=	**0**	1	0	0	1
-9_{10}	=	**1**	0	1	1	1

Figura 30 - Representação em complemento de dois.

Observando o exemplo pode-se notar que o número 11_{10} tem sua representação em binário igual a 1011_2, para encontrar o valor binário em complemento de dois para o número -11_2 inicialmente adiciona-se um bit de sinal a representação do valor com sinal positivo, assim, chega-se a 01011_2. O próximo passo é inverter todos os bits, o resultado é 10100_2. A última etapa é somar 1 a este valor, o resultado é **10101_2**.

A representação em complemento de dois é amplamente utilizar por apresentar algumas vantagens relevantes. Os circuitos digitais para fazer a adição e a subtração são muito simples podendo inclusive ser unificados.

Outra vantagem desta notação é que o número zero apresenta apenas uma forma de representação.

A título de exemplo a Figura 31 apresenta os números de -8_{10} a 7_{10} na forma binária sem sinal e em complemento de dois.

Decimal	Binário (Sinal + 3 bits)	
	Sem sinal	Complemento de 2
-8	-	1000
-7	-	1001
-6	-	1010
-5	-	1011
-4	-	1100
-3	-	1101
-2	-	1110
-1	-	1111
0	000	0000
1	001	0001
2	010	0010
3	011	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111

Figura 31 - Números negativos em complemento de dois.

2.4. Ponto flutuante

As técnicas de representação de números em sistemas digitais que foram estudadas até aqui são úteis para números inteiros, porém em muitas circunstâncias é necessário o uso de números reais. Para representar números reais, os sistemas digitais atuais utilizam um padrão chamado IEEE 754. Nas seções a seguir será apresentado o básico deste padrão

2.4.1. O padrão IEEE 745

Este padrão apresenta duas opções para a codificação dos números reais. A primeira se chama **ponto flutuante de precisão simples**, e possui 32 bits de tamanho. O primeiro bit (mais significativo) é dedicado a representação do sinal (S). Os 8 bits seguintes são dedicados ao expoente (E). Os 23 bits restantes são dedicados a representação da fração. A Figura 32 apresenta esta

estrutura. Observe que o número de bits é sempre 32, independente do valor que se esteja representando.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	…	2^{-22}	2^{-23}
S	Expoente (E)								Parte fracionária (F)								
1 bit	8 bits								23 bits								

Figura 32 - Ponto flutuante de precisão simples.

Para a representação de números neste formado é considerada a representação em notação científica normalizada, ou seja, com exatamente um dígito diferente de zero antes do ponto binário.

Para que se possa obter o valor numérico correspondente a representação em ponto flutuante de precisão simples deve-se fazer a seguinte operação.

$$Valor = -1^s \cdot (1 + F) \cdot 2^{E-127}$$

$$\text{Onde } 1 \leq E \leq 254$$

Existe também no padrão IEEE 754 uma representação de ponto flutuante de 64 bits. Esta representação é chamada de **ponto flutuante de precisão dupla**. A Figura 33 apresenta esta estrutura.

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	…	2^{-51}	2^{-52}
S	Expoente (E)										Parte fracionária (F)									
1 bit	11 bits										52 bits									

Figura 33 - Ponto flutuante de precisão dupla

Nesta notação, para que se possa obter o valor numérico correspondente deve-se fazer a seguinte operação.

$$Valor = -1^s \cdot (1 + F) \cdot 2^{E-1023}$$

$$\text{Onde } 1 \leq E \leq 2046$$

O termo E sofre um deslocamento tanto na representação de precisão simples como na representação de precisão dupla. Este deslocamento permite expressar expoentes positivos ou negativos, o que por sua vez permite representar números muito grandes ou muito pequenos.

Quando se trabalha com pontos flutuantes normalizados no padrão IEEE 754 deve levar em conta os seguintes fatores:

- No termo (1 + F) o “1” não aparece no valor binário, pois se assume que o número está em notação científica com exatamente um dígito diferente de zero antes do ponto binário.
- As equações não permitem a representação do número zero, assim, o padrão é preencher os campos E e F com zeros para representar o número zero.
- É possível representar o infinito preenchendo o campo E com 1’s e o campo F com 0’s, com o sinal apropriado.

- Quando o campo E é preenchido com 1's e o campo F é diferente de zero tem-se uma situação de número inválido, indicado por NaN (***N**ot **a** **N**umber*). Esta notação é útil para representar resultados inválidos como zero dividido por zero por exemplo.
- Representações com E igual a zero e F diferente de zero, indicam números não normalizados e devem ser evitadas.

A Figura 34 apresenta uma tabela que resume as possíveis representações no padrão IEEE 754. O valor "max" é 255 para números de precisão simples e 2047 para números de precisão dupla.

Sinal (S)	Expoente (E)	Parte Fracionária (F)	Valor
0 / 1	0	0	+0 / -0
0 / 1	max	0	$+\infty$ / $-\infty$
0 / 1	max	$\neq 0$	NaN
0 / 1	0	$\neq 0$	Não normalizado
0 / 1	De 1 a max -1	Qualquer	Normalizado

Figura 34 - Possíveis representações no padrão IEEE 754

Vejamos como exemplo a representação do número $-12{,}25_{10}$. A Figura 35 apresenta a representação deste número em ponto flutuante de precisão simples.

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	…	2^{-22}	2^{-23}
S	Expoente (E)								Parte fracionária (F)									
1	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	…	0	0

Figura 35 - Exemplo de ponto flutuante de precisão simples.

Observado as informações binárias e seus respectivos pesos temos:

$$S = 1$$

$$E = 1 \cdot 2^7 + 1 \cdot 2^1 = 130$$

$$F = 1 \cdot 2^{-1} + 1 \cdot 2^{-5} = 0{,}53125$$

Aplicando estes valores na fórmula tem-se:

$$Valor = -1^1 \cdot (1 + 0{,}53125) \cdot 2^{130-127} = -12{,}25$$

2.5. Código alfanumérico

Sistemas digitais também necessitam operar com informações na forma de textos, assim é necessário codificar na forma binária também letras. A codificação deste tipo de informações é chamada de codificação alfanumérica.

Os códigos alfanuméricos, também chamados de códigos de caracteres (letras), são códigos binários usados para representar dados como letras do alfabeto, números (digitos), símbolos matemáticos e sinais de pontuação, de uma forma que é compreensível e processável por um computador.

Este tipo de codificação é utilizada por dispositivos de entrada e saída, como teclados, monitores, impressoras bem como para a transmissão de informações pelas redes de computadores por exemplo.

Existem várias codificações em uso atualmente, como ASCII, Unicode, UTF8 etc. Um dos mais simples e mais utilizado é o ASCII, que será abordado a seguir.

2.5.1. Código ASCII

O Código ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) é um código muito popular usado em todos os sistemas digitais, ele utiliza 7 bits para representar 128 caracteres.

Os primeiros 32 (0 a 31) caracteres são ditos caracteres não imprimíveis, e servem como comando de controle para os periféricos. Os caracteres de 32 a 127 representam os símbolos letras e dígitos utilizados normalmente em nosso dia a dia. A Figura 36 apresenta a tabela ASCII, que contém todos estes 128 caracteres.

Decimal	Binário	Símbolo	Decimal	Binário	Símbolo	Decimal	Binário	Símbolo	Decimal	Binário	Símbolo
0	0	NUL	32	100000	espaço	64	1000000	@	96	1100000	`
1	1	SOH	33	100001	!	65	1000001	A	97	1100001	a
2	10	STX	34	100010	"	66	1000010	B	98	1100010	b
3	11	ETX	35	100011	#	67	1000011	C	99	1100011	c
4	100	EOT	36	100100	$	68	1000100	D	100	1100100	d
5	101	ENQ	37	100101	%	69	1000101	E	101	1100101	e
6	110	ACK	38	100110	&	70	1000110	F	102	1100110	f
7	111	BEL	39	100111	'	71	1000111	G	103	1100111	g
8	1000	BS	40	101000	(	72	1001000	H	104	1101000	h
9	1001	HT	41	101001	)	73	1001001	I	105	1101001	i
10	1010	LF	42	101010	*	74	1001010	J	106	1101010	j
11	1011	VT	43	101011	+	75	1001011	K	107	1101011	k
12	1100	FF	44	101100	,	76	1001100	L	108	1101100	l
13	1101	CR	45	101101	-	77	1001101	M	109	1101101	m
14	1110	SO	46	101110	.	78	1001110	N	110	1101110	n
15	1111	SI	47	101111	/	79	1001111	O	111	1101111	o
16	10000	DLE	48	110000	0	80	1010000	P	112	1110000	p
17	10001	DC1	49	110001	1	81	1010001	Q	113	1110001	q
18	10010	DC2	50	110010	2	82	1010010	R	114	1110010	r
19	10011	DC3	51	110011	3	83	1010011	S	115	1110011	s
20	10100	DC4	52	110100	4	84	1010100	T	116	1110100	t
21	10101	NAK	53	110101	5	85	1010101	U	117	1110101	u
22	10110	SYN	54	110110	6	86	1010110	V	118	1110110	v
23	10111	ETB	55	110111	7	87	1010111	W	119	1110111	w
24	11000	CAN	56	111000	8	88	1011000	X	120	1111000	x
25	11001	EM	57	111001	9	89	1011001	Y	121	1111001	y
26	11010	SUB	58	111010	:	90	1011010	Z	122	1111010	z
27	11011	ESC	59	111011	;	91	1011011	[	123	1111011	{
28	11100	FS	60	111100	<	92	1011100	\	124	1111100	|
29	11101	GS	61	111101	=	93	1011101	]	125	1111101	}
30	11110	RS	62	111110	>	94	1011110	^	126	1111110	~
31	11111	US	63	111111	?	95	1011111	_	127	1111111	Delete

Figura 36 - A tabela ASCII.

Além dos 128 caracteres da tabela ASCII padrão existem mais 128 caracteres que são uma versão expandida da tabela ASCII. Esta versão expandida contempla os caracteres de 128 a 255. Existem diversas versões desta expansão, estas versões são dependentes do idioma do sistema por exemplo. A Figura 37 apresenta os caracteres de 128 a 255 da tabela ASCII utilizada em um computador rodando Windows 10 em português do Brasil.

Decimal	Binário	Símbolo	Decimal	Binário	Símbolo	Decimal	Binário	Símbolo	Decimal	Binário	Símbolo
128	10000000	Ç	160	10100000	á	192	11000000	└	224	11100000	Ó
129	10000001	ü	161	10100001	í	193	11000001	┴	225	11100001	ß
130	10000010	é	162	10100010	ó	194	11000010	┬	226	11100010	Ô
131	10000011	â	163	10100011	ú	195	11000011	├	227	11100011	Ò
132	10000100	ä	164	10100100	ñ	196	11000100	─	228	11100100	õ
133	10000101	à	165	10100101	Ñ	197	11000101	┼	229	11100101	Õ
134	10000110	å	166	10100110	ª	198	11000110	ã	230	11100110	µ
135	10000111	ç	167	10100111	º	199	11000111	Ã	231	11100111	þ
136	10001000	ê	168	10101000	¿	200	11001000	╚	232	11101000	Þ
137	10001001	ë	169	10101001	®	201	11001001	╔	233	11101001	Ú
138	10001010	è	170	10101010	¬	202	11001010	╩	234	11101010	Û
139	10001011	ï	171	10101011	½	203	11001011	╦	235	11101011	Ù
140	10001100	î	172	10101100	¼	204	11001100	╠	236	11101100	ý
141	10001101	ì	173	10101101	¡	205	11001101	═	237	11101101	Ý
142	10001110	Ä	174	10101110	«	206	11001110	╬	238	11101110	¯
143	10001111	Å	175	10101111	»	207	11001111	¤	239	11101111	´
144	10010000	É	176	10110000	░	208	11010000	ð	240	11110000	
145	10010001	æ	177	10110001	▒	209	11010001	Ð	241	11110001	±
146	10010010	Æ	178	10110010	▓	210	11010010	Ê	242	11110010	‗
147	10010011	ô	179	10110011	│	211	11010011	Ë	243	11110011	¾
148	10010100	ö	180	10110100	┤	212	11010100	È	244	11110100	¶
149	10010101	ò	181	10110101	Á	213	11010101	ı	245	11110101	§
150	10010110	û	182	10110110	Â	214	11010110	Í	246	11110110	÷
151	10010111	ù	183	10110111	À	215	11010111	Î	247	11110111	¸
152	10011000	ÿ	184	10111000	©	216	11011000	Ï	248	11111000	°
153	10011001	Ö	185	10111001	╣	217	11011001	┘	249	11111001	¨
154	10011010	Ü	186	10111010	║	218	11011010	┌	250	11111010	·
155	10011011	ø	187	10111011	╗	219	11011011	█	251	11111011	¹
156	10011100	£	188	10111100	╝	220	11011100	▄	252	11111100	³
157	10011101	Ø	189	10111101	¢	221	11011101	¦	253	11111101	²
158	10011110	×	190	10111110	¥	222	11011110	Ì	254	11111110	■
159	10011111	ƒ	191	10111111	┐	223	11011111	▀	255	11111111	

Figura 37 - Tabela ASCII expandida.

Para aplicações mais complexas, como por exemplo, navegadores de internet ou editores de texto, codificações mais complexas são utilizadas para representar digitalmente os caracteres. Estas codificações permitem por exemplo a codificação de caracteres dos alfabetos orientais.

2.6. Exercícios

1) Preencha os valores que estão faltando na tabela conforma a respectiva base.

Decimal (base 10)	Binário (Base 2)	Octal (Base 8)	Hexadecimal (Base 16)
127_{10}			
	1011001_2		
		7621_8	
			$AB5_{16}$
321_{10}			
	10101100_2		
		352_8	
			$FF3_{16}$

2) Encontre a representação binária dos seguintes números negativos, utilize 5 bits.

Número	Sinal e Magnitude	Complemento de 1	Complemento de 2
-10_{10}			
-8_{10}			
-15_{10}			
-1_{10}			
-7_{10}			

3) Encontre o valor equivalente a seguinte sequência de bits que está em ponto flutuante.

   a) $01000010011000001000000000000000_2$
   b) $11000100010110110010000000000000_2$

Aula 3 - Álgebra booleana

Nesta aula serão abordados assuntos relacionados as regras matemáticas referentes as funções binárias. Serão apresentados teoremas matemáticos e regras de simplificação de expressões booleanas, bem como suas principais formas de representação.

3.1. Introdução

A álgebra booleana tem este nome em homenagem ao matemático e filósofo britânico **George Boole**. É utilizada para fazer uma análise formal dos circuitos digitais através de um conjunto de regras matemáticas.

3.2. Funções booleanas

Para compor um sistema matemático de dois valores (binário), representado por zeros e uns, utiliza-se funções booleanas. Estas funções são compostas por variáveis binárias e por funções matemáticas binarias, a adição lógica (+) também chamada de função **OU** (*OR*), a multiplicação lógica (·) também chamada de função **E** (*AND*) e inversão lógica (¯). Veja alguns exemplos.

$$y = \bar{a} \cdot b$$

$$y = \overline{(a+b)} \cdot c$$

Nas funções booleanas as letras representam variáveis que podem assumir valores 0 ou 1.

3.3. Propriedades fundamentais da álgebra booleana

Não é objetivo deste material demostrar os teoremas relacionados com a álgebra booleana, serão apenas demostradas as propriedades fundamentais necessárias aos estudos dos sistemas digitais.

A Tabela 2 apresenta as propriedades envolvendo a função OU.

Tabela 2 - Propriedades da função OU.

$a + 0 = a$
$a + 1 = 1$
$a + a = a$
$a + \bar{a} = 1$
$a + b = b + a$ (comutativa)
$(a + b) + c = a + (b + c) = a + b + c$ (associativa)

A função OU (+) pode ser imaginada da seguinte forma, se pelo menos um dos operandos for verdadeiro o resultado é também verdadeiro.

Assim como a função OU, a função E também tem um conjunto de propriedades, demostrado na Tabela 3.

Tabela 3 - propriedades da função E.

$a \cdot 1 = a$
$a \cdot 0 = 0$
$a \cdot a = a$
$a \cdot \bar{a} = 0$
$a \cdot b = b \cdot a$ (comutativa)
$(a \cdot b) \cdot c = a \cdot (b \cdot c)$ (associativa)
$a \cdot (b + c) = a \cdot b + a \cdot c$ (distributiva)

3.3.1. Teorema de DeMorgan

Dentre os vários teoremas que compõe a álgebra booleana vale a pena citar os teoremas de DeMorgan. Estes teoremas são extremamente úteis na simplificação de expressões em que um produto ou soma de variáveis é invertido. Estes teoremas podem ser resumidos da seguinte forma.

$$\left(\overline{a + b}\right) = \bar{a} \cdot \bar{b}$$

$$\left(\overline{a \cdot b}\right) = \bar{a} + \bar{b}$$

Segundo estes teoremas quando a operação OU entre duas variáveis é invertida, é equivalente a inverter cada uma das variáveis individualmente e então, fazer a operação E entre elas. Da mesma forma, quando a operação E entre duas variáveis é invertida, é equivalente a inverter cada variável individualmente e então fazer a operação OU. De forma generalizada, estes teoremas são válidos para qualquer número de variáveis, veja a seguir.

$$\left(\overline{a + b + c + \cdots}\right) = \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{\cdots}$$

$$\left(\overline{a \cdot b \cdot c \cdot \cdots}\right) = \bar{a} + \bar{b} + \bar{c} + \bar{\cdots}$$

Os termos a, b ou c podem ser também expressões booleanas complexas, não faz diferença.

3.3.2. Exemplo de aplicação de DeMorgan

Os teoremas de DeMorgan podem ser utilizados para simplificar funções booleanas, veja o seguinte exemplo. Para simplificar a função booleana $z = \overline{(\bar{a} + c) \cdot (b + \bar{d})}$, podemos considerar $(\bar{a} + c)$ como X e $(b + \bar{d})$ como Y, assim teremos $z = \overline{X \cdot Y} = \bar{X} + \bar{Y}$ ou de forma equivalente $z = \overline{(\bar{a} + c)} + \overline{(b + \bar{d})}$.

Da mesma forma podemos aplicar o teorema de DeMorgan a cada um dos termos.

$$\overline{(\bar{a} + c)} = (a \cdot \bar{c})$$

$$\overline{(b + \bar{d})} = (\bar{b} \cdot d)$$

Assim, a expressão final simplificada é $(a \cdot \bar{c}) + \left(\bar{b} \cdot d\right)$ ou ainda $a \cdot \bar{c} + \bar{b} \cdot d$ pois a operação E tem precedência sobre a operação OU.

3.3.3. Algumas identidades auxiliares

Existem ainda algumas identidades que são úteis na hora de simplificar funções booleana, são elas.

$$a + a \cdot b = a$$

$$a + \bar{a} \cdot b = a + b$$

$$(a + b) \cdot (a + c) = a + b \cdot c$$

É importante destacar que em algumas literaturas as operações E podem ter seu sinal (·) suprimido, assim $a \cdot b \cdot c = abc$, porém isso só funciona quando o nome das variáveis tem apenas uma letra, com nomes mais longos pode haver confusão.

3.4. Tabelas verdade

Tabelas verdade é uma tabela que apresenta todas as possíveis entradas e as saídas correspondentes de uma função booleana. Para uma função com N variáveis, a tabela verdade deve possuir 2^N Linhas. Considere por exemplo a função $y = a \cdot b + c$, a seguir é apresentada sua tabela verdade.

Tabela 4 - Tabela verdade.

Entradas			**Saída**
a	**b**	**c**	**y**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Nesta tabela é possível observar todas as possíveis entradas e as saídas correspondentes para a função.

3.5. Simplificação de funções booleanas

O objetivo da simplificação de funções booleanas é encontrar uma outra expressão que seja equivalente, porém com um menor número de operações lógicas. Para isso deve-se utilizar as propriedades e teoremas já apresentados. Vejamos alguns exemplos.

1) Simplifique a expressão: $y = (a + b + c) \cdot (a + b + c)$.

Podemos chamar $(a + b + c)$ de x. Aplicando a propriedade $x \cdot x = x$ obtém-se.

$$y = (a + b + c)$$

Para verificar se a simplificação está correta vamos construir a tabela verdade.

Tabela 5 - Exemplo 1 de simplificação.

Entradas				y
a	**b**	**c**	$(a+b+c)$	$(a+b+c)\cdot(a+b+c)$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Observando a Tabela 5 pode-se notar que $(a+b+c)\cdot(a+b+c)=(a+b+c)$.

2) Simplifique a expressão: $y=a\cdot b+a\cdot(b+c)+b\cdot(b+c)$

Aplicando a propriedade distributiva em $a\cdot(b+c)$ e $b\cdot(b+c)$ obtemos.

$$y=a\cdot b+a\cdot b+a\cdot c+b\cdot b+b\cdot c$$

Aplicando agora a propriedade $a\cdot a=a$ para $b\cdot b$ e $a\cdot b+a\cdot b$ obtemos.

$$y=a\cdot b+a\cdot c+b+b\cdot c$$

Aplicando a expressão auxiliar $a+a\cdot b=a$ em $b+b\cdot c$ obtemos.

$$y=a\cdot b+a\cdot c+b$$

Rearranjando os termos temos.

$$y=b+b\cdot a+a\cdot c$$

Aplicando a expressão auxiliar $a+a\cdot b=a$ em $b+b\cdot a$ obtemos.

$$y=b+a\cdot c$$

Este é o resultado final. Para verificá-lo observe a Tabela 6.

Tabela 6 - Exemplo 2a de simplificação.

Entradas							
a	**b**	**c**	$a\cdot b$	$(b+c)$	$a\cdot(b+c)$	$b\cdot(b+c)$	**y**
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	0	1	0	1	1
0	1	1	0	1	0	1	1
1	0	0	0	0	0	0	0
1	0	1	0	1	1	0	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1

Agora compare o resultado da Tabela 6 com os resultados da Tabela 7.

Tabela 7 - Exemplo 2b de simplificação.

Entradas				
a	**b**	**c**	$a \cdot c$	**y**
0	0	0	0	0
0	0	1	0	0
0	1	0	0	1
0	1	1	0	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1

É possível observar que as duas expressões apresentam o mesmo resultado.

3) Simplifique a expressão: $y = a \cdot b \cdot c + a \cdot \bar{c} + a \cdot \bar{b}$

Aplicando a propriedade distributiva, isolando o a obtemos.

$$y = a \cdot (b \cdot c + \bar{c} + \bar{b})$$

Aplicando DeMorgan em $\bar{c} + \bar{b}$ obtemos.

$$y = a \cdot (b \cdot c + \overline{b \cdot c})$$

Agora aplicando a propriedade $a + \bar{a} = 1$ em $b \cdot c + \overline{b \cdot c}$ obtemos.

$$y = a \cdot 1 = a$$

O resultado pode ser verificado através da tabela verdade como nos casos anteriores.

4) Simplifique a expressão: $y = (a + \bar{b}) \cdot (a + c)$

Aplicando a propriedade distributiva obtemos.

$$y = a \cdot a + a \cdot c + \bar{b} \cdot a + \bar{b} \cdot c$$

Agora aplicando a propriedade $a \cdot a = a$ obtemos.

$$y = a + a \cdot c + \bar{b} \cdot a + \bar{b} \cdot c$$

Aplicando a propriedade associativa podemos escrever.

$$y = (a + a \cdot c + \bar{b} \cdot a) + \bar{b} \cdot c$$

Agora aplicando a propriedade distributiva obtemos.

$$y = a \cdot (1 + c + \bar{b}) + \bar{b} \cdot c$$

Como qualquer coisa OU 1 é 1 obtemos

$$y = a + \bar{b} \cdot c$$

Este resultado também pode ser comprovado via tabelas verdade.

3.6. Formato padrão para funções booleanas

Existem algumas formas padronizadas para a expressão de funções booleanas. Dentre elas a mais utilizada é a soma de produtos (SOP – ***S**um **O**f **P**roducts*). No formato de soma de produtos, a função é representada pela soma de termos chamados **minterms.** Se considerarmos uma função booleana de N variáveis, um minterms é o produto destas N variáveis ou de seu complemento. São exemplos de minterms para três variáveis: $a \cdot \bar{b} \cdot c$, $a \cdot b \cdot c$ e $\bar{a} \cdot b \cdot c$, etc.

A representação das funções booleanas na forma de soma de produtos é facilmente obtida da tabela verdade, veja um exemplo a seguir.

Tabela 8 - Obtenção da soma de produtos

Entradas	Minterms	Saída
a b c		
0 0 0	$\bar{a} \cdot \bar{b} \cdot \bar{c}$	0
0 0 1	$\bar{a} \cdot \bar{b} \cdot c$	1
0 1 0	$\bar{a} \cdot b \cdot \bar{c}$	0
0 1 1	$\bar{a} \cdot b \cdot c$	0
1 0 0	$a \cdot \bar{b} \cdot \bar{c}$	1
1 0 1	$a \cdot \bar{b} \cdot c$	0
1 1 0	$a \cdot b \cdot \bar{c}$	1
1 1 1	$a \cdot b \cdot c$	0

Observando a Tabela 8 podemos notar que cada minterm representa um conjunto de entradas, onde quando a entrada tem valor 1 a variável correspondente aparece diretamente, e quando a entrada é 0 a variável correspondente aparece invertida (complemento). Por exemplo, para a entrada $a = 0, b = 1\, e\, c = 1$ o minterm correspondente é $\bar{a} \cdot b \cdot c$. Para obter a representação em soma de produtos da função que representa a Tabela 8 basta realizarmos a soma dos minterms das linhas da tabela onde a saída é 1, ignorando os minterms das linhas onde a saída é 0. Assim, para a Tabela 8 a função seria a seguinte.

$$y = \bar{a} \cdot \bar{b} \cdot c + a \cdot \bar{b} \cdot \bar{c} + \ a \cdot b \cdot \bar{c}$$

Esta função apresenta exatamente o comportamento da Tabela 8. A representação em soma de produtos não representa a função minimizada, ou seja, é possível obter uma função equivalente menor.

Existem ainda outras representações para funções booleanas, como por exemplo o produto de somas, mas não é o objetivo deste material abordar tais representações.

3.7. Mapas de Karnaugh

Observando as seções anteriores nota-se que a minimização de funções booleanas é muito importante, e não é uma tarefa trivial. Neste sentido, o matemático e cientista da computação Edward W. Veitch criou um sistema, que depois foi aperfeiçoado pelo engenheiro de telecomunicações Maurice Karnaugh, que possibilita simplificar uma equação booleana ou

converter uma tabela verdade em sua equação booleana minimizada. Este método ficou conhecido como mapa de Karnaugh.

Os mapas de Karnaugh são úteis para minimizar circuitos ou equações booleanas de até 6 variáveis, para mais do que isso este método se torna complicado.

Um mapa de Karnaugh pode ser entendido como uma outra forma de representar uma tabela verdade, onde a finalidade é facilitar sua simplificação.

Para uma função de N variáveis, um mapa de Karnaugh é uma tabela de 2^N quadrados denominados celas. A Figura 38 - Modelo de mapa de Karnaugh para quatro variáveis Figura 38 apresenta um mapa de Karnaugh para quatro variáveis, a, b, c e d.

a b \ c d	0 0	0 1	1 1	1 0
0 0	0000	0001	0011	0010
0 1	0100	0101	0111	0110
1 1	1100	1101	1111	1110
1 0	1000	1001	1011	1010

Figura 38 - Modelo de mapa de Karnaugh para quatro variáveis

Na Figura 38 - Modelo de mapa de Karnaugh para quatro variáveis Figura 38 pode-se observar que um mapa de Karnaugh é composto por celas que possuem um valor binário relacionado as quatro variáveis envolvidas. Na esquerda estão representados os valores das variáveis a e b. É importante notar que a distribuição destes valores não segue a distribuição padrão, e sim uma distribuição em que cada elemento se diferencia de seus vizinhos em apenas 1 bit. Assim para as variáveis a e b a sequência de distribuição é 00, 01, 11 e 10. Seguindo a mesma lógica, na parte superior estão representadas as variáveis c e d, estas as variáveis seguem a mesma sequência de distribuição é 00, 01, 11 e 10.

A título de exemplo, no interior das celas foi colocado o valor binário para as variáveis a, b, c e d, seguindo esta ordem. As Figura 39 (a) e (b) apresentam respectivamente exemplos de formatação de mapas de Karnaugh respectivamente para 2 e 3 variáveis.

a \ b	0	1
0	00	01
1	01	11

(a)

a \ b c	0 0	0 1	1 1	1 0
0	000	001	011	010
1	100	101	111	110

(b)

Figura 39 - Mapas de Karnaugh para 2 e 3 variáveis

3.7.1. Preenchimento do mapa de Karnaugh

O preenchimento do mapa de Karnaugh é o processo de observar cada uma das linhas da tabela verdade, identificar a cela correspondente e colocar nesta cela 1 ou 0 correspondente ao resultado daquela linha na tabela.

Como exemplo vamos considerar a tabela verdade da Tabela 9.

Tabela 9 - Tabela verdade com três variáveis.

Entradas a b c	Saída
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	1
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	0

Esta tabela verdade é composta por três variáveis, a, b e c. Para esta tabela o mapa de Karnaugh é apresentado na Figura 40.

a \ b c	00	01	11	10
0	0	1	1	0
1	1	0	0	1

Figura 40 - Mapa de Karnaugh para a tabela.

Observe por exemplo que a linha destacada em vermelho na Tabela 9 corresponde as entradas $a = 0, b = 1, c = 1$ apresenta uma saída igual a 1. Esta linha corresponde ao número 1 destacado em vermelho no mapa de Karnaugh da Figura 40. O mesmo procedimento é adotado para mapas com diferentes números de variáveis.

3.7.2. Grupos de celas adjacentes

A próxima etapa na montagem do mapa de Karnaugh é o agrupamento das celas adjacentes cujo valor é 1. Podem ser criados grupos de 1, 2, 4, 8 e assim por diante de celas, sempre potências de 2. A Figura 41 apresenta alguns agrupamentos de celas válidos.

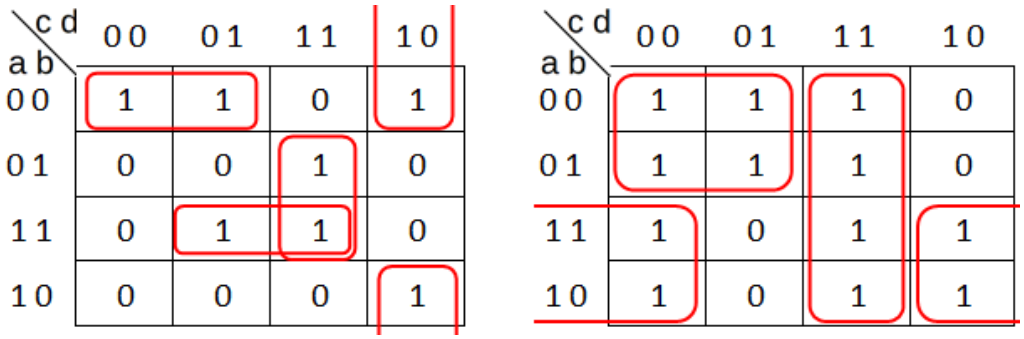


Figura 41 - Exemplos de grupos válidos de celas.

Observe que um mesmo número 1 pode participar de mais de um grupo. Os agrupamentos de celas podem extrapolar as bordas do mapa, por exemplo saindo pela esquerda e entrando pela direita, ou saindo por baixo e entrando por cima.

É muito importante que se faça sempre grupos com o maior número de "uns" possível, pois só assim o processo de simplificação irá funcionar.

3.7.3. Resolvendo o mapa de Karnaugh

A última etapa na simplificação por mapa de Karnaugh é encontrar a expressão que representa cada um dos grupos de celas demarcado na etapa anterior. Para isso faz-se a análise de cada grupo individualmente. Veja os grupos demarcados na Figura 42.

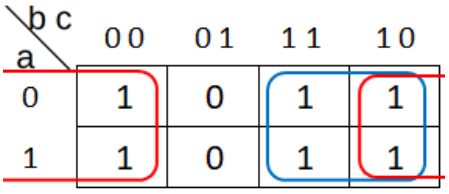


Figura 42 - Resolução de mapa de Karnaugh.

Nesta figura temos dois grupos, o vermelho com quatro celas e o azul também com quatro celas. Para cada grupo deve-se fazer a seguinte pergunta. Quais são os valores que cada uma das variáveis assume neste grupo?

Para o primeiro grupo (vermelho) temos:

$$a = 0/1, b = 0/1 \text{ e } c = 0 .$$

Neste caso, as variáveis a e b pode assumir 0 e 1, e a variável c só pode assumir o valor 0. Isso quer dizer que, as variáveis a e b são irrelevantes para este grupo e a variável c devem ser negada (invertida). Assim temos que $\cancel{a} \cdot \cancel{b} \cdot \bar{c}$ desta forma a expressão que representa este grupo é $\bar{c}$.

Aplicando a mesma lógica ao segundo grupo (azul), temos:

$$a = 0/1, b = 1 \text{ e } c = 0/1 .$$

Assim temos que $\cancel{a} \cdot b \cdot \cancel{c}$ desta forma a expressão que representa este grupo é simplesmente b.

A resposta final é a soma das expressões que representam cada um dos grupos. Para este exemplo temos:

$$y = \bar{c} + b$$

Observe que a entrada a não influencia na saída.

3.7.4. Exemplos de mapa de Karnaugh

Para deixar mais clara a utilização dos mapas de Karnaugh na minimização de funções serão apresentados alguns exemplos com diferentes números de variáveis, desmontando diferentes senários.

1) Utilize o mapa de Karnaugh para determinar a função booleana mínima para os dados da Tabela 10.

Tabela 10 - Exemplo de mapa de Karnaugh com duas variáveis.

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

Observando a Tabela 10 é possível montar o seguinte mapa de Karnaugh.

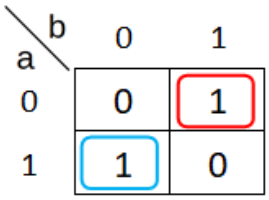


Figura 43 - Mapa de Karnaugh do exemplo.

Note que não é possível fazer grupos com mais de 1 cela, assim temos para este mapa dois grupos. No primeiro grupo (azul) as variáveis a e b assumem os seguintes valores $a = 1\ e\ b = 0$, assim para este grupo temos a expressão $a \cdot \bar{b}$. Já para o segundo grupo (vermelho), as variáveis a e b assumem os seguintes valores $a = 0\ e\ b = 1$, assim para este grupo temos a expressão $\bar{a} \cdot b$. A expressão resultante é a soma das duas anteriores, o que resulta em:

$$y = a \cdot \bar{b} + \bar{a} \cdot b$$

2) Utilize o mapa de Karnaugh para determinar a função booleana mínima para os dados da Tabela 11.

Tabela 11 - Exemplo de mapa de Karnaugh com duas variáveis.

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

Observando a Tabela 11 é possível montar o seguinte mapa de Karnaugh.

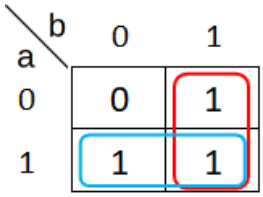


Figura 44 - Mapa de Karnaugh do exemplo.

Para este mapa temos novamente dois grupos, porém grupos com 2 celas. No primeiro grupo (azul) as variáveis a e b assumem os seguintes valores $a = 1$ e ~~$b = 0/1$~~, assim para este grupo temos a expressão a. Já para o segundo grupo (vermelho), as variáveis a e b assumem os seguintes valores ~~$a = 0/1$~~ e $b = 1$, assim para este grupo temos a expressão b.

A expressão resultante é a soma das duas anteriores, o que resulta em:

$$y = a + b$$

3) Neste exemplo o objetivo é minimizar a expressão que está na forma de soma de produtos $y = \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c + a \cdot b \cdot \bar{c} + a \cdot b \cdot c$.

É possível montar o mapa de Karnaugh diretamente da equação, porém, fica mais fácil montar primeiro a tabela verdade. Assim, a Tabela 12 apresenta as entradas e saídas para esta equação booleana.

Tabela 12 - Tabela verdade para este exemplo.

a b c	y
0 0 0	0
0 0 1	1
0 1 0	1
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	1
1 1 1	1

Observando a Tabela 12 é possível montar o seguinte mapa de Karnaugh.

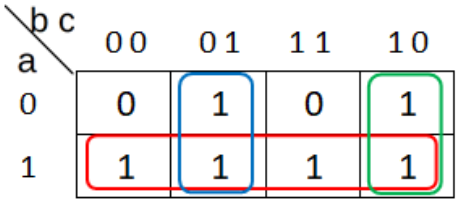


Figura 45 - Mapa de Karnaugh do exemplo.

Para este mapa temos três grupos, dois deles com 2 celas e um com 4 celas. No primeiro grupo (azul) as variáveis a, b e c assumem os seguintes valores ~~$a = 0/1$~~, $b = 0$ e $c = 1$, assim para este grupo temos a expressão $\bar{b} \cdot c$. Já para o segundo grupo (verde), as variáveis a, b e c assumem

os seguintes valores ~~$a = 0/1$~~, $b = 1$ e $c = 0$, assim para este grupo temos a expressão $b \cdot \bar{c}$. No terceiro grupo (vermelho), as variáveis a, b e c assumem os seguintes valores $a = 1$, ~~$b = 0/1$~~ e ~~$c = 0/1$~~, assim para este grupo temos a expressão a.

A expressão resultante é a soma das três anteriores, o que resulta em:

$$y = \bar{b} \cdot c + b \cdot \bar{c} + a$$

4) O quarto exemplo é uma tabela verdade com 4 variáveis, para o qual se deseja encontrar a função minimizada. Veja a Tabela 13.

Tabela 13 - Tabela verdade do exemplo.

a b c d	y
0 0 0 0	1
0 0 0 1	1
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	1
1 0 0 0	1
1 0 0 1	0
1 0 1 0	1
1 0 1 1	1
1 1 0 0	1
1 1 0 1	0
1 1 1 0	1
1 1 1 1	1

Observando a Tabela 13 é possível montar o seguinte mapa de Karnaugh.

c d / a b	0 0	0 1	1 1	1 0
0 0	1	1	1	0
0 1	1	1	1	0
1 1	1	0	1	1
1 0	1	0	1	1

Figura 46 - Mapa de Karnaugh do exemplo.

Para este mapa temos três grupos, todos com 4 celas. No primeiro grupo (azul) as variáveis a, b, c e d assumem os seguintes valores $a = 1$, ~~$b = 0/1$~~, ~~$c = 0/1$~~ e $d = 0$, assim para este grupo temos a expressão $a \cdot \bar{d}$. Já para o segundo grupo (verde), as variáveis a, b, c e d assumem os seguintes valores ~~$a = 0/1$~~, ~~$b = 0/1$~~, $c = 1$ e $d = 1$, assim para este grupo temos a expressão $c \cdot d$.

No terceiro grupo (vermelho), as variáveis a, b, c e d assumem os seguintes valores $a = 0, \sout{b = 0/1}, c = 0$ e $\sout{d = 0/1}$, assim para este grupo temos a expressão $\bar{a} \cdot \bar{c}$.

A expressão resultante é a soma das três anteriores, o que resulta em:

$$y = a \cdot \bar{d} + c \cdot d + \bar{a} \cdot \bar{c}$$

5) Para o quinto exemplo reservamos um caso diferente dos anteriores com relação a montagem dos grupos no mapa de Karnaugh. Para este exemplo, deseja-se minimizar a função de quatro variáveis $y = \bar{a} \cdot \bar{b} \cdot \bar{c} \cdot \bar{d} + a \cdot \bar{b} \cdot \bar{c} \cdot \bar{d} + a \cdot \bar{b} \cdot c \cdot \bar{d} + \bar{a} \cdot \bar{b} \cdot c \cdot \bar{d}$.

Para esta função booleana temos o mapa de Karnaugh da Figura 47.

c d / a b	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

Figura 47 - Mapa de Karnaugh para este exemplo.

O mapa de Karnaugh para esta função permite criar um grupo com as quatro celas dos cantos, como mostrado na Figura 47. As variáveis a, b, c e d assumem os seguintes valores para o grupo $\sout{a = 0/1}, b = 0, \sout{c = 0/1}$ e $d = 0$, assim a expressão minimizada resultante é:

$$y = \bar{b} \cdot \bar{d}$$

3.7.5. Mapas com mais de quatro variáveis

A mesma técnica pode ser utilizada para minimizar equações booleanas com mais de 4 variáveis. Porém, nestes casos os mapas ficam mais complicados, permitindo a formação de grupos mais complexo.

Este material não irá abordar estes casos, mas a literatura apresenta várias soluções para este tipo de problemas.

3.7.6. Funções incompletas

Para muitos problemas do mundo real é possível que não se tenha a tabela verdade completa, ou ainda, algumas das combinações das entradas não irão acontecer devido a restrições físicas do sistema digital. Para estas situações, a saída dos sistemas para determinadas entradas não importa (em inglês é normal dizer *Don't-Care*). Veja um exemplo na Tabela 14.

Tabela 14 - Exemplo de tabela verdade incompleta.

a b c	y
0 0 0	1
0 0 1	x
0 1 0	0

0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	x
1 1 1	1

Nestes casos, para as linhas da tabela onde a saída não é importante o valor desta saída é simbolizado por "x". Da mesma forma, na hora de montar o mapa de Karnaugh, as celas correspondentes a estas linhas recebem "x". Para a Tabela 15 podemos compor o mapa de Karnaugh da Figura 47.

Tabela 15 - Mapa de Karnaugh para o exemplo.

a \ b c	0 0	0 1	1 1	1 0
0	1	x	1	0
1	0	1	1	x

Na hora de resolver o mapa de Karnaugh e obter as expressões booleanas resultantes, os valores desconhecidos "x" podem ser interpretados como 0 ou 1, o que for resultar no maior grupo de celas. Assim será possível obter a menor expressão booleana que atenda a tabela verdade, sem se preocupar com as linhas incompletas ou não relevantes.

Dito isso, para o mapa de Karnaugh da Figura 47, temos dois grupos, um com 2 celas e um com 4 celas. No primeiro grupo (azul) as variáveis a, b e c assumem os seguintes valores ~~$a = 0/1$~~, ~~$b = 0/1$~~ $e\ c = 1$, assim para este grupo temos a expressão c. Já para o segundo grupo (vermelho), as variáveis a, b e c assumem os seguintes valores $a = 0, b = 0\ e$ ~~$c = 0/1$~~, assim para este grupo temos a expressão $\bar{a} \cdot \bar{b}$.

A expressão resultante é a soma das duas anteriores, o que resulta em:

$$y = c + \bar{a} \cdot \bar{b}$$

3.8. Exercícios de mapa de Karnaugh

1) Utilizando mapa de Karnaugh encontre a função booleana minimizada para a seguinte tabela verdade.

Tabela 16 - Tabela verdade do exercício 1 de mapas de Karnaugh.

a	**b**	**y**
0	0	1
0	1	1
1	0	1
1	1	0

2) Utilizando mapa de Karnaugh encontre a função booleana minimizada para a seguinte soma de produtos $y = \bar{a} \cdot \bar{b} \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot c + a \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c + a \cdot b \cdot \bar{c}$.

3) Utilizando mapa de Karnaugh encontre a função booleana minimizada para a seguinte tabela verdade.

Tabela 17 - Tabela verdade do exercício 3 de mapas de Karnaugh.

a b c d	**y**
0 0 0 0	1
0 0 0 1	0
0 0 1 0	1
0 0 1 1	0
0 1 0 0	0
0 1 0 1	0
0 1 1 0	1
0 1 1 1	0
1 0 0 0	0
1 0 0 1	1
1 0 1 0	1
1 0 1 1	1
1 1 0 0	0
1 1 0 1	0
1 1 1 0	1
1 1 1 1	1

4) Utilizando mapa de Karnaugh encontre a função booleana minimizada para a seguinte tabela verdade.

Tabela 18 - Tabela verdade do exercício 4 de mapas de Karnaugh.

a b c	**y**
0 0 0	1
0 0 1	x
0 1 0	x
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	x
1 1 1	1

Aula 4 - Portas lógicas

O objetivo desta aula é introduzir as portas lógicas, que são os componentes eletrônicos fundamentais nos sistemas digitais.

4.1. Introdução

Para que se posso construir sistemas digitais é necessário que se utilize circuitos eletrônicos capazes de executar as funções booleanas básicas. Os circuitos desenvolvidos para este fim são chamados de portas lógicas. A título de exemplo, a Figura 48 apresenta a imagem de um circuito integrado que contém 6 portas lógicas inversoras.

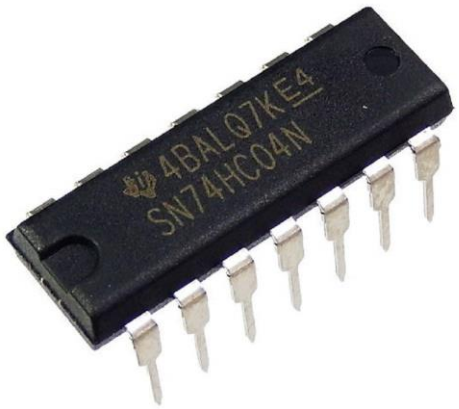


Figura 48 – Circuito integrado da porta lógica inversora.

A seguir serão apresentadas as principais portas lógicas

4.2. Porta lógica inversora

A mais simples das portas lógicas é a porta inversora (*NOT* em inglês). Esta porta possui uma entrada e uma saída. Quando a entrada recebe nível lógico 0, a saída assume o nível lógico 1, e quando a entrada recebe nível lógico 1, a saída assume o nível lógico 0.

A Tabela 19 apresenta a tabela verdade para a porta inversora.

Tabela 19 - Tabela verdade da porta inversora.

Entrada	Saída
a	**y**
0	1
1	0

A expressão booleana para a porta inversora é:

$$y = \bar{a}$$

A porta inversora é representada nos diagramas de circuitos digitais ou sistemas digitais pelos símbolos apresentados na Figura 49.

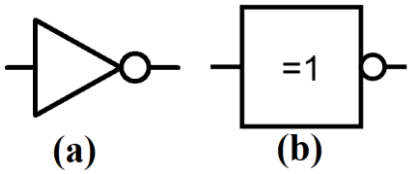


Figura 49 - Simbologia da porta inversora.

A Figura 49 (a) apresenta a simbologia tradicional e mais comum para a porta inversora, esta simbologia segue a norma ANSI (*American National Standards Institute*). Existe também a simbologia normatizada pela IEC (*International Electrotechnical Commission*), apresentada na Figura 49 (b).

A Figura 50 apresenta a forma de onde para a estrada e a saída de uma porta inversora.

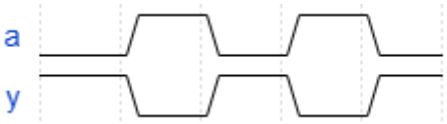


Figura 50 - Forma de onda da porta inversora.

Os fabricantes de circuitos integrados costumam empacotar as portas lógicas em pacotes contendo várias unidades. A Figura 51 apresenta duas imagens de um circuito integrado chamado 7404, que contém 6 portas inversoras.

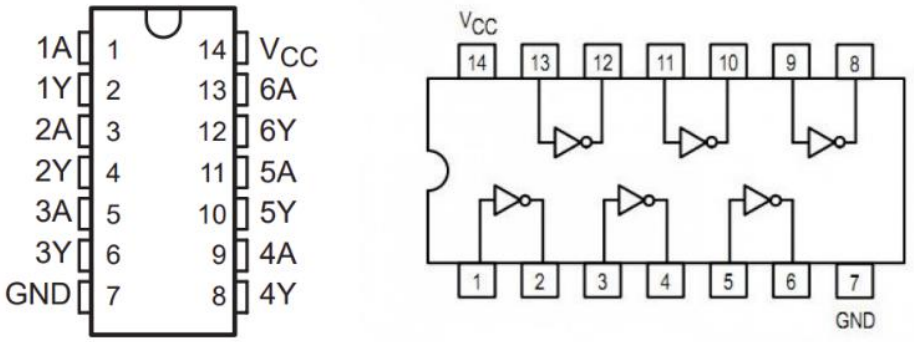


Figura 51 - Circuito integrado 7404.

4.3. Atraso de propagação.

A propagação das entradas para a saída das portas lógicas não acontece instantaneamente, existe um atraso. Este atraso é resultado das capacitâncias internas do circuito. Nos sistemas digitais costuma-se chamar estes atrasos de atrasos de propagação (em inglês *delay*). Apesar de muito pequeno, em algumas situações até desprezível, este atraso pode em algumas situações produzir resultados indesejados nos circuitos.

Na maioria dos gráficos de formas de onda digital estes atrasos são desconsiderados, mas em algumas situações é importante que eles sejam considerados. Assim a Figura 52 apresenta a

mesma forma de onda da porta inversora apresentada na Figura 51, porém agora considerando o atraso de propagação.

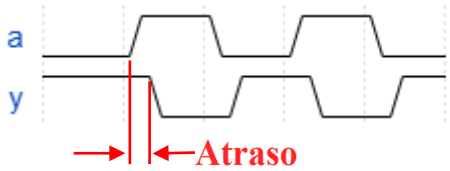


Figura 52 - Forma de onda inversora com atraso.

É importante ressaltar que este atraso de propagação é consequência da tecnologia utilizada na fabricação dos circuitos integrados, assim, tecnologias diferentes de fabricantes diferentes possuem também diferentes atrasos de propagação.

4.4. Porta "OU"

A segunda porta lógica que vamos estudar é a porta lógica OU (em inglês *OR*). Esta porta lógica, diferentemente da porta inversora possui duas entradas. Para esta porta a saída assume 1 se qualquer uma das entradas, ou as duas entradas forem iguais a 1. A saída assume 0 apenas se as duas entradas forem iguais a 0.

A Tabela 20 apresenta a tabela verdade da porta OU.

Tabela 20 - Tabela verdade da porta OU.

Entradas		**Saída**
a	**b**	**y**
0	0	0
0	1	1
1	0	1
1	1	1

A expressão booleana para a porta OU é:

$$y = a + b$$

A porta OU é representada nos diagramas de circuitos digitais pelos símbolos apresentados na Figura 53.

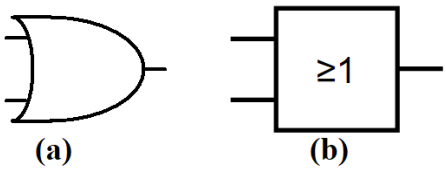


Figura 53 - Simbologia da porta OU.

A Figura 53 (a) apresenta a simbologia tradicional e mais comum para a porta OU, esta simbologia segue a norma ANSI (*American National Standards Institute*). Existe também a

simbologia normatizada pela IEC (*International Electrotechnical Commission*), apresentada na Figura 53 (b).

A Figura 54 apresenta a forma de onda para as estradas e a saída de uma porta OU.

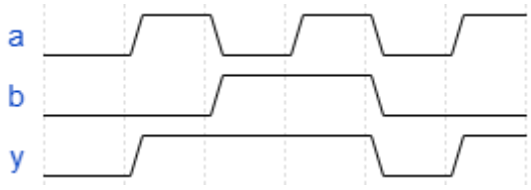


Figura 54 - Forma de onda da porta OU.

O circuito integrado 7432 é um dos mais comuns a implementar as portas OU nos circuitos digitais. A Figura 55 apresenta uma imagem deste circuito, que contém 4 portas OU.

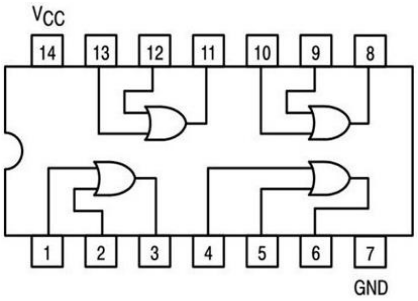


Figura 55 - Circuito integrado 7432

4.5. Porta “E”

A porta lógica E (em inglês *AND*), assim como a porta OU, possui duas entradas. Para esta porta a saída assume 1 somente quando as duas entradas são 1. Se uma ou mais entradas forem 0 a saída assume valor 0.

A Tabela 21 apresenta a tabela verdade da porta E.

Tabela 21 - Tabela verdade da porta E.

Entradas		**Saída**
a	**b**	**y**
0	0	0
0	1	0
1	0	0
1	1	1

A expressão booleana para a porta E é:

$$y = a \cdot b$$

A porta E é representada nos diagramas de circuitos digitais pelos símbolos apresentados na Figura 56.

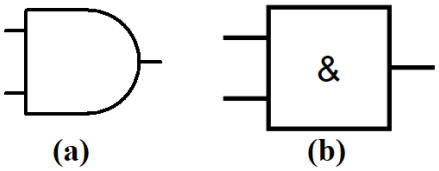


Figura 56 - Simbologia da porta E.

A Figura 56 (a) apresenta a simbologia tradicional e mais comum para a porta E, esta simbologia segue a norma ANSI (*American National Standards Institute*). Existe também a simbologia normatizada pela IEC (*International Electrotechnical Commission*), apresentada na Figura 56 (b).

A Figura 57 apresenta a forma de onda para as estradas e a saída de uma porta E.

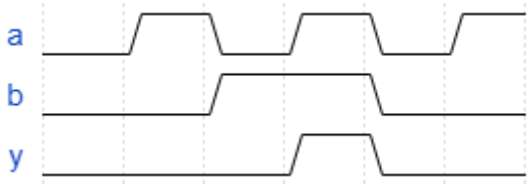


Figura 57 - Forma de onda da porta E.

O circuito integrado 7408 é um dos mais comuns a implementar as portas E nos circuitos digitais. A Figura 58 apresenta uma imagem deste circuito, que contém 4 portas E.

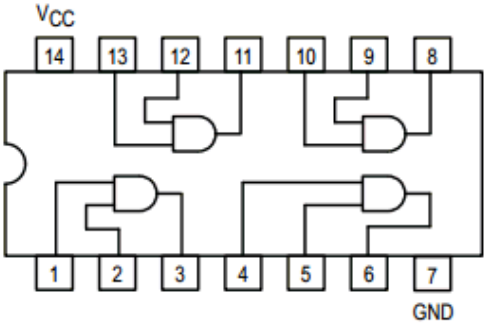


Figura 58 - Circuito integrado 7408.

4.6. Porta “Não OU”

Além das três portas lógicas que representam as três operações fundamentais da álgebra booleana existem ainda outras. A porta lógica Não OU (em inglês *NOR*) é uma delas. Assim como as portas anteriores, esta possui duas entradas. Para esta porta a saída assume 1 somente quando as duas entradas são 0. Se uma ou mais entradas forem 1 a saída assume valor 0.

É possível notar que esta porta apresenta exatamente o mesmo comportamento que a porta OU, só que com a saída invertida.

A Tabela 22 apresenta a tabela verdade da porta Não OU.

Tabela 22 - Tabela verdade da porta Não OU.

Entradas		Saída
a	b	y
0	0	1
0	1	0
1	0	0
1	1	0

A expressão booleana para a porta Não OU é:

$$y = \overline{a + b}$$

A porta Não OU é representada nos diagramas de circuitos digitais pelos símbolos apresentados na Figura 59.

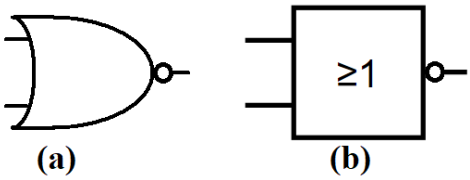


Figura 59 - Simbologia da porta Não OU.

A Figura 59 (a) apresenta a simbologia tradicional e mais comum para a porta Não OU, esta simbologia segue a norma ANSI (*American National Standards Institute*). Existe também a simbologia normatizada pela IEC (*International Electrotechnical Commission*), apresentada na Figura 59 (b). A Figura 60 apresenta a forma de onda para as estradas e a saída de uma porta Não OU.

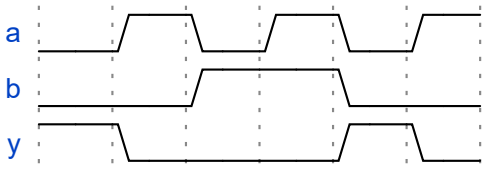


Figura 60 - Forma de onda da porta não OU.

O circuito integrado 7402 é um dos mais comuns a implementar as portas Não OU nos circuitos digitais. A Figura 61 apresenta uma imagem deste circuito, que contém 4 portas Não OU.

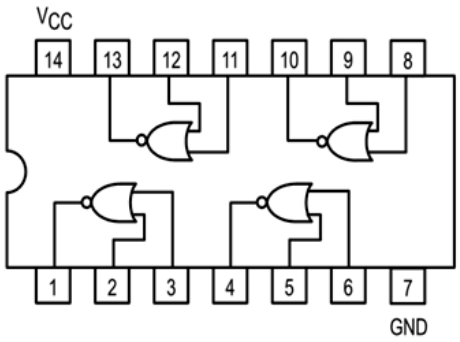


Figura 61 - Circuito integrado 7402.

4.7. Porta “Não E”

A porta lógica Não E (em inglês *NAND*), assim como as portas anteriores, possui duas entradas. Para esta porta a saída assume 0 somente quando as duas entradas são 1. Se uma ou mais entradas forem 0 a saída assume valor 1.

É possível notar que esta porta apresenta exatamente o mesmo comportamento que a porta E, só que com a saída invertida.

A Tabela 23 apresenta a tabela verdade da porta Não E.

Tabela 23 - Tabela verdade para a porta Não E.

Entradas		**Saída**
a	**b**	**y**
0	0	1
0	1	1
1	0	1
1	1	0

A expressão booleana para a porta Não E é:

$$y = \overline{a \cdot b}$$

A porta Não E é representada nos diagramas de circuitos digitais pelos símbolos apresentados na Figura 62.

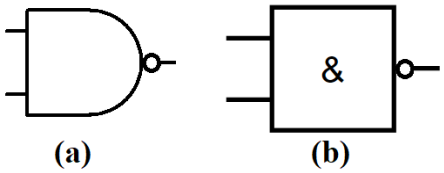


Figura 62 - Simbologia da porta Não E.

A Figura 62 (a) apresenta a simbologia tradicional e mais comum para a porta Não E, esta simbologia segue a norma ANSI (*American National Standards Institute*). Existe também a simbologia normatizada pela IEC (*International Electrotechnical Commission*), apresentada na Figura 62 (b).

A Figura 63 apresenta a forma de onda para as estradas e a saída de uma porta Não E.

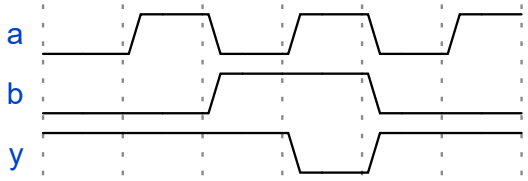


Figura 63 - Forma de onda da porta Não E.

O circuito integrado 7400 é um dos mais comuns a implementar as portas Não E nos circuitos digitais. A Figura 64 apresenta uma imagem deste circuito, que contém 4 portas Não E.

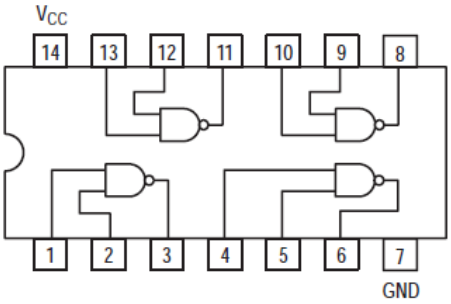


Figura 64 - Circuito integrado 7400.

4.8. Porta “OU Exclusivo”

A porta lógica OU Exclusivo (em inglês *XOR*), assim como as portas anteriores, possui duas entradas. Para esta porta a saída assume 0 quando as duas entradas são iguais e assume 1 se as duas entradas são diferentes.

A Tabela 24 apresenta a tabela verdade da porta OU Exclusivo.

Tabela 24 - Tabela verdade da porta OU Exclusivo.

Entradas		**Saída**
a	**b**	**y**
0	0	0
0	1	1
1	0	1
1	1	0

Para esta função é utilizado um operador especial, com o símbolo $\oplus$. Assim, a expressão booleana para a porta OU Exclusivo é:

$$y = a \oplus b = \bar{a} \cdot b + a \cdot \bar{b}$$

A porta OU Exclusivo é representada nos diagramas de circuitos digitais pelos símbolos apresentados na Figura 65.

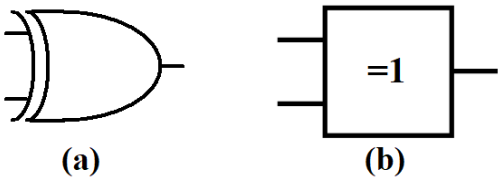


Figura 65 - Simbologia da porta OU Exclusivo.

A Figura 65 (a) apresenta a simbologia tradicional e mais comum para a porta OU Exclusivo, esta simbologia segue a norma ANSI (*American National Standards Institute*). Existe também a simbologia normatizada pela IEC (*International Electrotechnical Commission*), apresentada na Figura 65 (b).

A Figura 66 apresenta a forma de onda para as estradas e a saída de uma porta OU Exclusivo.

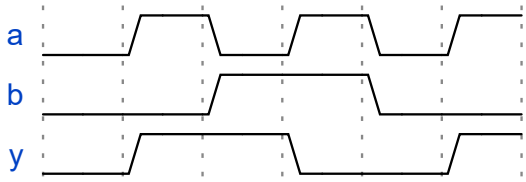


Figura 66 - Forma de onda da porta OU exclusivo.

O circuito integrado 7486 é um dos mais comuns a implementar as portas OU Exclusivo nos circuitos digitais. A Figura 67 apresenta uma imagem deste circuito, que contém 4 portas OU Exclusivo.

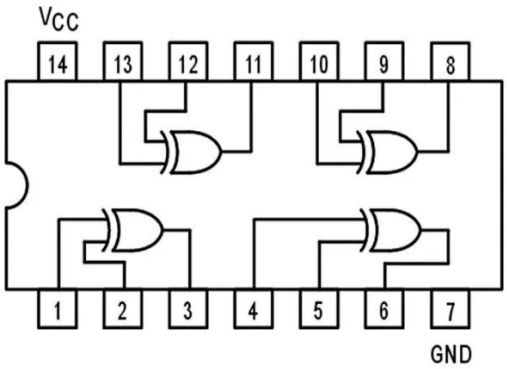


Figura 67 - Circuito integrado 7486.

4.9. Porta “Não OU Exclusivo”

A porta lógica Não OU Exclusivo (em inglês *XNOR*), assim como as portas anteriores, possui duas entradas. Para esta porta a saída assume 1 quando as duas entradas são iguais e assume 0 se as duas entradas são diferentes.

A Tabela 25 apresenta a tabela verdade da porta OU Exclusivo.

Tabela 25 - Tabela verdade da porta Não OU Exclusivo.

Entradas		**Saída**
a	**b**	**y**
0	0	1
0	1	0
1	0	0
1	1	1

A expressão booleana para a porta Não OU Exclusivo é:

$$y = \overline{a \oplus b}$$

A porta Não OU Exclusivo é representada nos diagramas de circuitos digitais pelos símbolos apresentados na Figura 68.

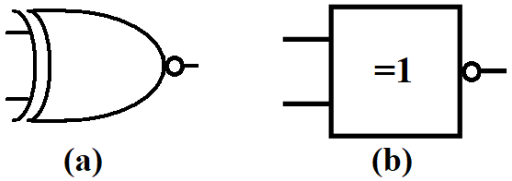


Figura 68 - Porta Não OU Exclusivo.

A Figura 68Figura 65 (a) apresenta a simbologia tradicional e mais comum para a porta Não OU Exclusivo, esta simbologia segue a norma ANSI (*American National Standards Institute*). Existe também a simbologia normatizada pela IEC (*International Electrotechnical Commission*), apresentada na Figura 68 (b).

A Figura 69 apresenta a forma de onda para as estradas e a saída de uma porta Não OU Exclusivo.

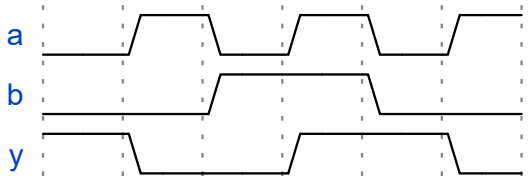


Figura 69 - Forma de onda porta Não OU Exclusivo.

O circuito integrado 74266 implementa as portas Não OU Exclusivo nos circuitos digitais. A Figura 70 apresenta uma imagem deste circuito, que contém 4 portas Não OU Exclusivo.

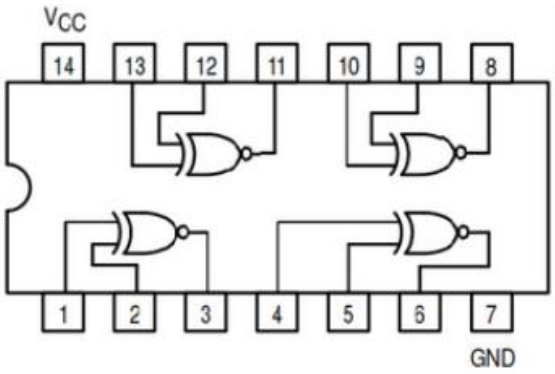


Figura 70 - Pinagem Integrado 74266.

4.10. Portas lógicas com mais entradas.

Até aqui foram apresentadas portas lógicas com duas entradas, porém é comum necessitarmos de portas lógicas com mais entradas.

Para estas situações tem-se duas soluções possíveis, a associação de várias portas para aumentar o número de entradas ou a utilização de circuitos integrados que implementam portas com mais entradas.

A Figura 71 apresenta um exemplo de associação de portas lógicas para a obtenção de mais entradas.

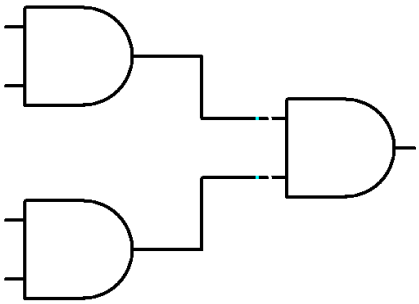


Figura 71 - Associação de portas E.

Neste exemplo são associadas 3 portas E de duas entradas para a obtenção de um circuito digital equivalente a uma porta E de 4 entradas.

A segunda alternativa é a utilização de portas lógicas com mais entradas. São disponibilizadas diversas portas com diversos números de entradas. A Figura 72, apresenta como exemplo, circuitos integrados com portas lógicas de três entradas.

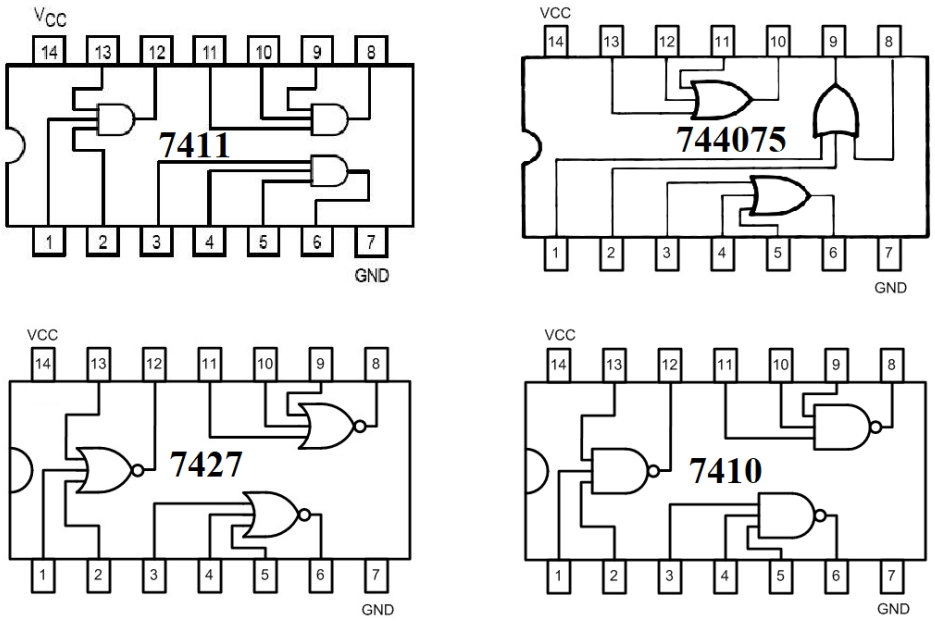


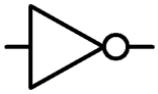
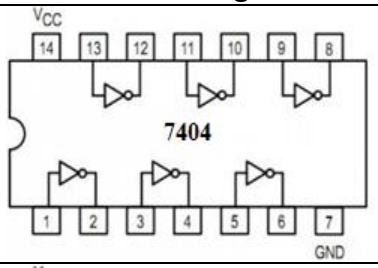
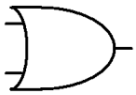
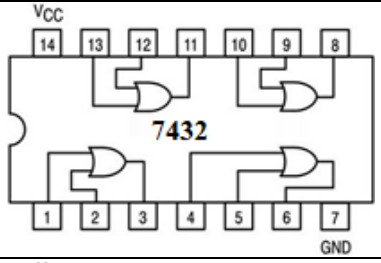
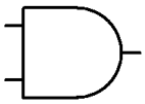
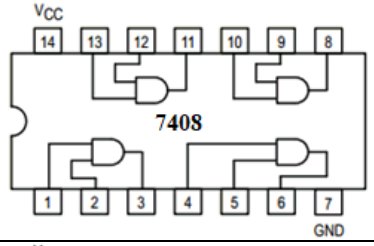
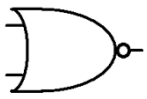
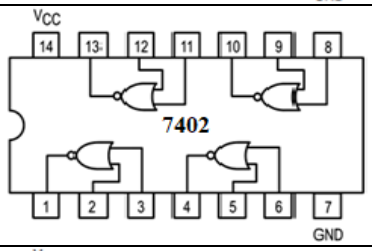
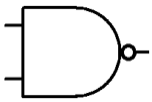
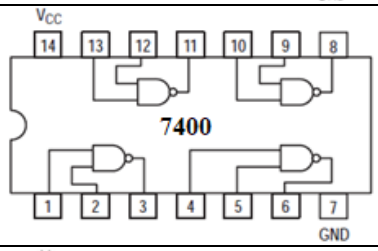
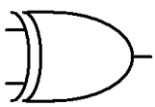
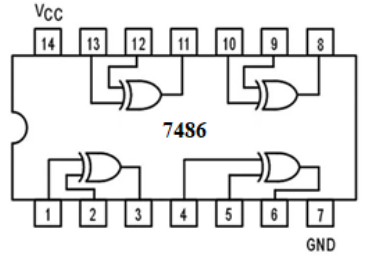
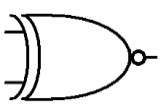
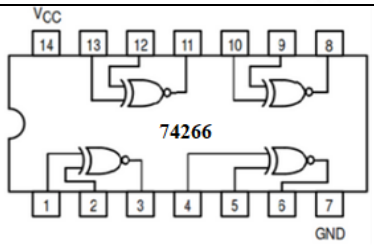
Figura 72 - Portas lógicas com 3 entradas.

4.11. Resumo das portas lógicas

Para facilitar o acesso as informações sobre as portas lógicas, a Tabela 26 apresenta um resumo de tudo o que foi visto com relação as portas lógicas.

É importante destacar que nesta tabela são apresentadas apenas as portas com duas entradas, porém existem inúmeras outras configurações, com números de entradas variados disponíveis no mercado.

Tabela 26 - Resumo das portas lógicas.

Porta	Símbolo	Função	Tabela Verdade	Circuito Integrado
Inversora (*NOT*)		$y = \bar{a}$	<table><tr><th>Entrada</th><th>Saída</th></tr><tr><th>a</th><th>y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	
OU (*OR*)		$y = a + b$	<table><tr><th>a</th><th>b</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	
E (*AND*)		$y = a \cdot b$	<table><tr><th>a</th><th>b</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	
Não OU (*NOR*)		$y = \overline{a + b}$	<table><tr><th>a</th><th>b</th><th>y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	
Não E (*NAND*)		$y = \overline{a \cdot b}$	<table><tr><th>a</th><th>b</th><th>y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	
OU Exclusivo (XOR)		$y = a \oplus b$	<table><tr><th>a</th><th>b</th><th>y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	
Não OU Exclusivo (XNOR)		$y = \overline{a \oplus b}$	<table><tr><th>a</th><th>b</th><th>y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	

4.12. Implementação de funções booleanas com portas lógicas.

Em aplicações práticas é comum que se deseje implementar na forma de um sistema digital físico as funções booleanas necessária ao funcionamento do nosso sistema. A implementação das funções booleanas na forma de circuitos eletrônicos digitais é possível através da utilização de portas lógicas.

Como exemplo vamos implementar a seguinte função booleana na forma de um circuito digital utilizando portas lógicas.

$$y = \overline{\bar{a} \cdot b} + \overline{(b + c)} + a \cdot b \cdot c$$

O circuito digital da Figura 73 implementa esta função.

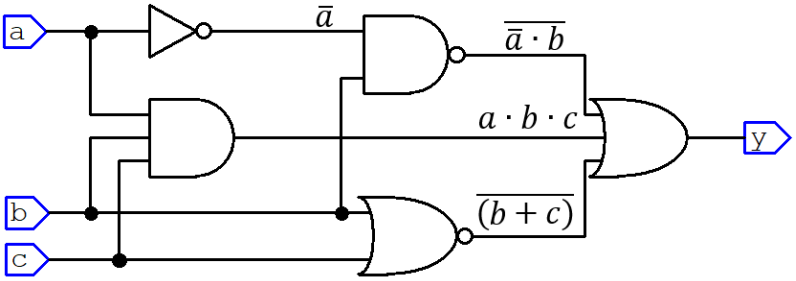


Figura 73 - Implementação da função booleana com portas lógicas.

Com base neste diagrama poderíamos montar então o circuito físico para executar a função desejada. A Figura 74 apresenta a montagem deste circuito em uma matriz de contatos (Protoboard).

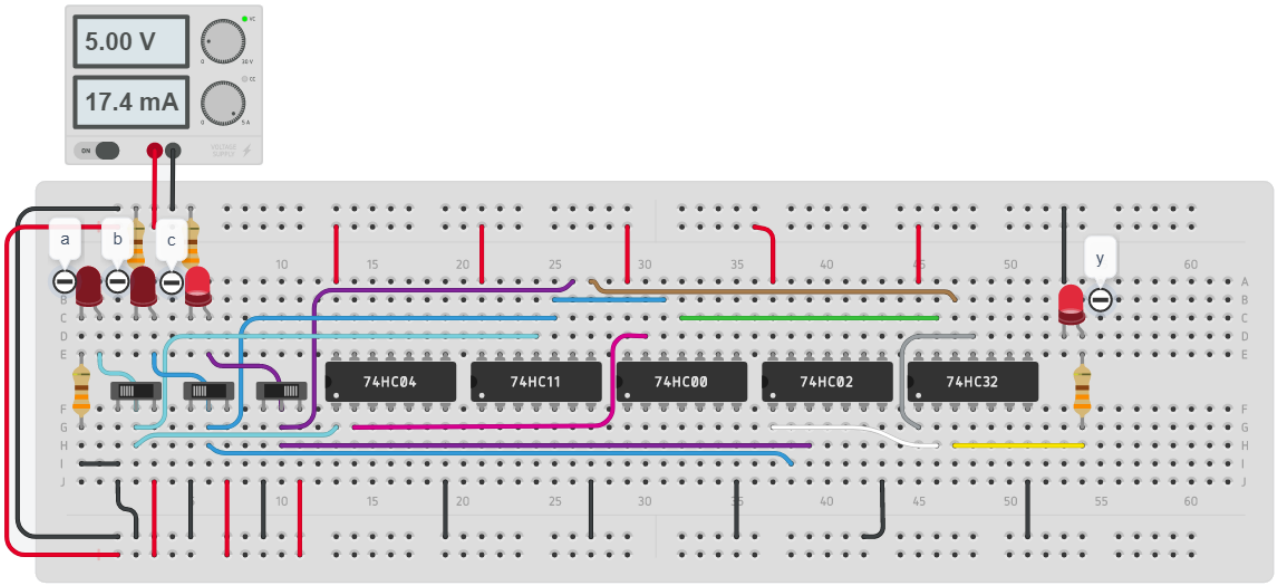


Figura 74 - Montagem do circuito no protoboard.

Este tipo de montagem será realizado nas aulas experimentais da disciplina.

4.13. Cuidados com as portas lógicas

Existem algumas precauções necessárias quando se trabalha com portas lógicas.

O primeiro cuidado é a quantidade de entradas que uma saída digital pode alimentar. Existe um limite de capacidade inerente a cada tecnologia de fabricação, o manual de cada componente fornece estas informações.

Outro cuidado que se deve ter é nunca interligar as saídas de dois circuitos digitais ou portas lógicas, pois quando uma assumir nível lógico 1 e a outra assumir nível lógico 0 teremos um sinal de 5V conectado a um sinal de 0V, ou seja um curto-circuito. Esta situação é normalmente chamada de curto lógico, e deve ser evitada pois danifica os componentes de forma irreversível.

4.14. Exercícios

Para a seguinte expressão algébrica booleana determine a tabela verdade e construa um circuito com portas lógicas que implementa esta função.

$$y = \bar{a} \cdot \bar{b} + a \cdot \bar{b}$$

Aula 5 - Circuitos lógicos combinacionais

O objetivo desta aula é apresentar aos alunos os principais circuitos lógicos combinacionais, enfatizando sua construção e sua aplicação. Serão abordados circuitos codificadores e decodificadores, circuitos multiplexadores e demultiplexadores, circuitos comparadores entre outros.

5.1. Introdução

Existem basicamente dois tipos de circuitos lógicos, os circuitos combinacionais e os circuitos sequenciais. Os circuitos combinacionais são aqueles que são construídos com portas lógicas de forma a determinar o valor das saídas apenas em função dos valores atuais das entradas. Os circuitos apresentados nas aulas anteriores são todos circuitos combinacionais. Um sistema digital é dito combinacional se ele não possui nenhum tipo de realimentação ou memória. Pode-se afirmar que os circuitos combinacionais podem ser representados por um conjunto de equações booleanas, uma equação para cada saída. A Figura 75 apresenta uma representação genérica de um circuito lógico combinacional.

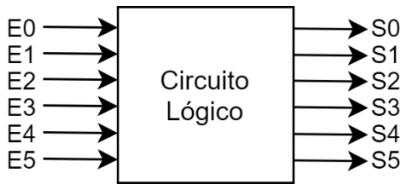


Figura 75 - Circuito lógico combinacional.

Os circuitos sequenciais por sua vez utilizam elementos de armazenamento de informação ou algum tipo de realimentação, além das portas lógicas tradicionais. Desta forma as saídas não dependem apenas das entradas do circuito no momento. Nos circuitos sequenciais, as saídas dependem das entradas atuais e das entradas anteriores do circuito, ou seja, dos estados dos elementos de memória. Assim, pode-se afirmar que as saídas de um circuito sequencial dependem do histórico das entradas deste circuito. A Figura 76 apresenta uma representação genérica de um circuito lógico sequencial.

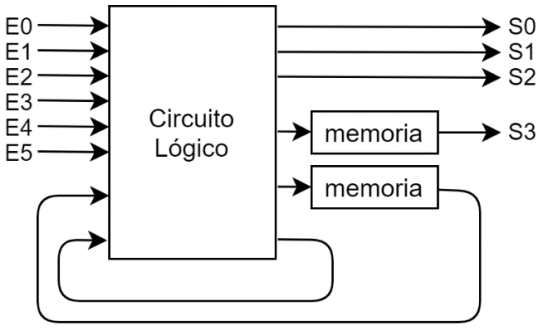


Figura 76 - Circuito lógico sequencial.

Os circuitos lógicos combinacionais já foram vistos nas aulas anteriores, e serão estudados de forma mais aprofundada nesta aula. Já os circuitos lógicos sequenciais serão estudados nas aulas seguintes.

5.2. Codificadores e decodificadores

Os decodificadores são uma categoria de circuitos lógicos sequenciais que permitem selecionar uma dentre várias saídas em função de uma combinação de suas entradas. Os codificadores realizam a função inversa.

5.2.1. Decodificadores

Decodificadores são circuitos lógicos sequenciais com *n* entradas e até 2^n saídas, onde apenas uma das saídas é selecionada de cada vez. A saída selecionada é determinada pela combinação dos valores das entradas. A Figura 77 apresenta um decodificador de duas entradas e quatro saídas.

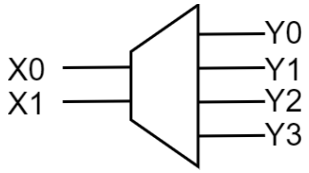


Figura 77 - Decodificador de 2 para 4.

Para este decodificador apenas uma das saídas *Y* pode assumir o valor 1 enquanto as outras três saídas devem permanecer em 0. A saída selecionada para receber 1 é determinada pela combinação dos valores das entradas *X*. A Tabela 27 apresenta a tabela verdade para este circuito.

Tabela 27 - Tabela verdade para o decodificador de 2 para 4.

Entradas		Saídas			
X1	X0	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Como podemos observar cada combinação da entrada seleciona uma e apenas uma das saídas. Como os decodificadores são circuitos lógicos combinacionais eles podem ser construídos utilizando portas lógicas. A Figura 78 apresenta o circuito digital correspondente ao decodificador de 2 para 4.

Em algumas situações é necessário desligar todas as saídas. Para estes casos existem decodificadores que possuem uma entrada adicional que habilita (em inglês *enable*) ou desabilita todas as saídas.

A Figura 79 apresenta um decodificador de duas entradas para quatro saídas com uma entrada de *enable*. Observe que esta entrada é ativa em nível lógico 1.

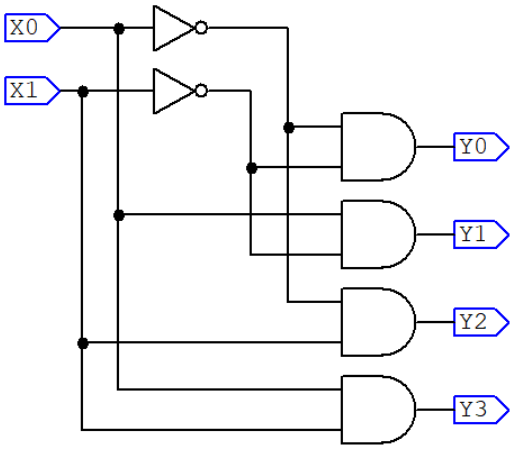


Figura 78 - Circuito do decodificador de 2 para 4.

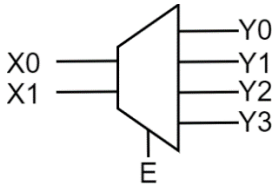


Figura 79 - Decodificador de 2 para 4 com *enable*.

A Tabela 28 apresenta a tabela verdade para este decodificador.

Tabela 28 - Tabela verdade do decodificador de 2 para 4 com *enable*.

Entradas			Saídas			
E	X1	X0	Y0	Y1	Y2	Y3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Para a construção do circuito do decodificador de 2 para 4 com entrada de *enable* também se utiliza portas lógicas, como pode ser observado na Figura 80.

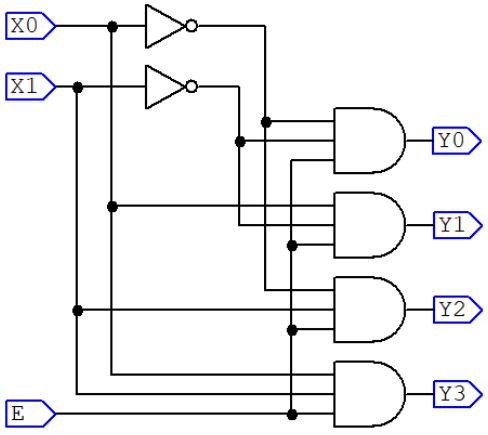


Figura 80 - Circuito do decodificador de 2 para 4 com *enable*.

É importante destacar que existem decodificadores onde as saídas ou a entrada de *enable* são ativas em nível lógico 0, isso depende das especificações do fabricante.

Em muitas aplicações é normal necessitarmos de decodificadores com mais entradas e saídas. A Figura 81 apresenta três decodificadores comuns com mais entradas.

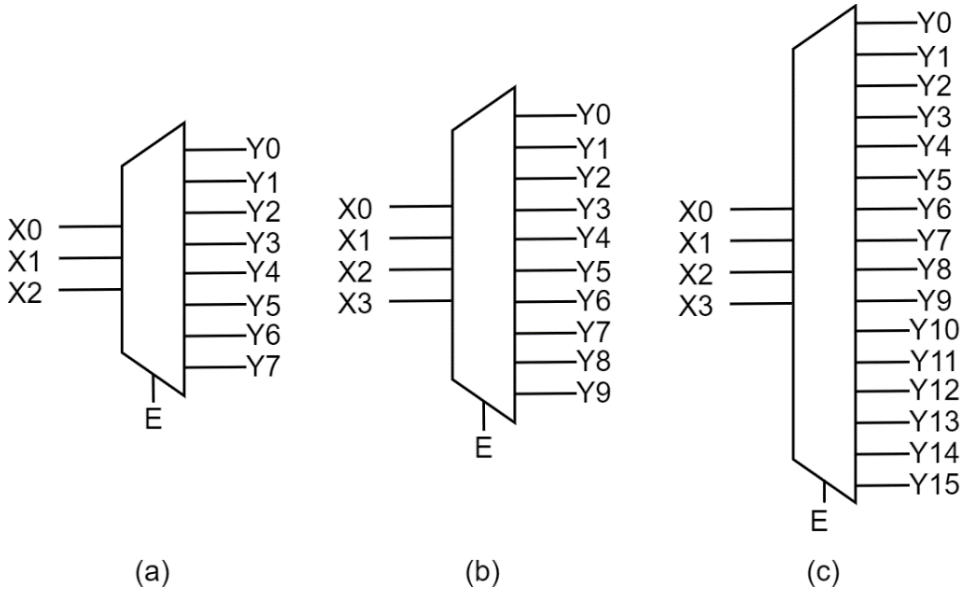


Figura 81 - Decodificadores com mais entradas.

Na Figura 81 (a) é possível observar um decodificador de 3 entradas e 8 saídas. A Tabela 29 apresenta a tabela verdade para este componente.

Tabela 29 - Tabela verdade do decodificador de 3 para 8.

Entradas				Saídas							
E	X2	X1	X0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

Este decodificador assim como os anteriores ativa apenas uma de suas saídas *Y* de cada vez, conforme a combinação dos bits da entrada *X.* A entrada de *enable* também deve receber nível 1 para que a saída seja ativada.

Já na Figura 81 (b) é possível observar um decodificador de 4 entradas e 10 saídas. Para este codificador temos uma situação um pouco diferente, nem todas as combinações da entrada são utilizadas para ativar saídas, A Tabela 30 presenta a tabela verdade para este componente onde esta situação dica mais evidente.

Tabela 30 - Tabela verdade do decodificador de 4 para 10.

Entradas					Saídas									
E	X3	X2	X1	X0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9
0	X	X	X	X	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	1	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	1	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	0	0	1	0	0	0	0	0	0	0	0	0	1
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

E finalmente na Figura 81 (c) é possível observar um decodificador de 4 entradas e 16 saídas. A Tabela 31 presenta a tabela verdade para este componente.

Tabela 31 - Tabela verdade do decodificador de 4 para 16.

	Entradas				Saídas															
E	X3	X2	X1	X0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10	Y11	Y12	Y13	Y14	Y15
0	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Assim como as portas lógicas, os decodificadores também são fabricados na forma de circuitos integrado. A Tabela 32 apresenta alguns circuitos integrados bastante utilizados que implementam decodificadores.

Tabela 32 - Circuitos integrados decodificadores.

Circuito integrado	Entradas / saídas	Comentário
74139	2 / 4	Contém 2 decodificadores de 2 entradas e 4 saídas, as saídas e o *enable* são ativas em 0.
74138	3 / 8	Contém 1 decodificadores de 3 entradas e 8 saídas, as saídas são ativas em 0, possui 3 *enables*.
7442	4 / 10	Contém 1 decodificadores de 4 entradas e 10 saídas, as saídas são ativas em 0, não possui *enable*.
74154	4 / 16	Contém 1 decodificadores de 4 entradas e 16 saídas, as saídas são ativas em 0, possui 2 *enables* ativos em 0.

Outra situação comum envolvendo decodificadores é a necessidade de aumentarmos o número de entradas e saídas dos decodificadores disponíveis. Nestes casos é possível realizar a associação de decodificadores.

A Figura 82 apresenta a associação de dois decodificadores de 4 entradas e 16 saídas para formar um decodificador de 5 entradas e 32 saídas.

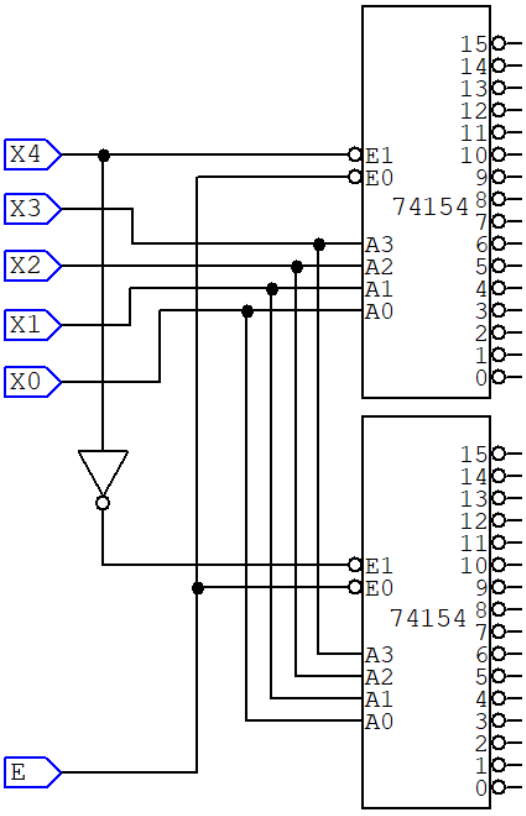


Figura 82 - Associação de decodificadores.

Observando a figura podemos notar que para realizar a associação foi necessário criar um novo sinal de entrada, neste caso chamado de *X4*. Para isso foram utilizadas as entradas de *enable* e uma porta inversora. Assim, quando a entrada *X4* está em 0 o circuito integrado de cima funciona, gerando as primeiras 16 saídas, e quando *X4* está em 1 o circuito integrado de baixo funciona, gerando as 16 saídas restantes. A porta inversora garante que apenas um dos circuitos integrados trabalhe por vez.

5.2.2. Codificadores

Os codificadores são circuitos lógicos combinacionais que realizam a operação inversa dos decodificadores. Nestes circuitos tem-se um conjunto de entradas onde apenas uma delas pode ser verdadeira por vez. As saídas codificam de forma binária a representação de qual das entradas está ativa no momento. A Figura 83 apresenta um codificador típico de 4 entradas e 2 saídas.

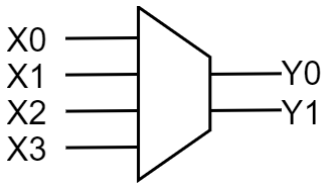


Figura 83 - Codificador de 4 para 2.

A tabela verdade para este circuito pode ser visualizada na Tabela 33.

Tabela 33 - Tabela verdade do codificador de 4 para 2.

Entradas				Saídas	
X0	X1	X2	X3	Y1	Y0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

É possível observar que apenas uma das entradas deve estar ativa por vez, e as saídas representam de forma binária o número da entrada selecionada. A Figura 84 apresenta um circuito combinacional elaborado com portas lógicas que implementa um codificador de 4 entradas e 2 saídas.

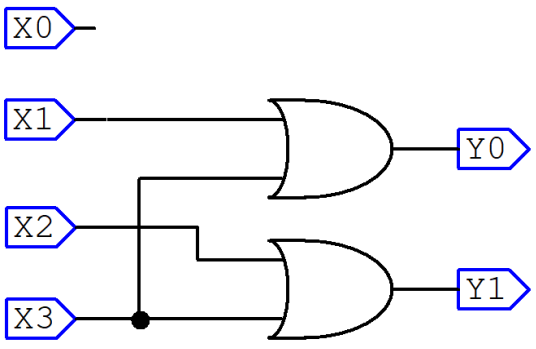


Figura 84 - Circuito do codificador de 4 para 2.

Observando este circuito podemos notar que a entrada *X0* não é conectada a nenhuma lógica, isso acontece, pois, as saídas permanecem em nível lógico 0 independentemente do valor da entrada *X0*.

Existem ainda codificadores com mais entradas, porém devido a construção destes circuitos, quando mais de uma entrada é ativada, a saída assume valores inválidos. Para minimizar este problema existem os codificadores de prioridade, que serão apresentados na seção a seguir.

5.2.3. Codificadores de prioridade

Os codificadores de prioridade funcionam de forma semelhante aos codificadores normais, porém neste tipo de componente as entradas de maior número têm prioridade sobre as entradas de menor número. Assim, se mais de uma entrada for acionada ao mesmo tempo, a entrada de maior prioridade será codificada na saída.

Neste tipo de codificador é comum também encontrarmos mais uma saída, que serve para indicar se o resultado do codificador é válido ou não. A Figura 85 apresenta um exemplo de codificador de prioridade de 4 entradas e 2 saídas, com mais uma saída de resultado válido.

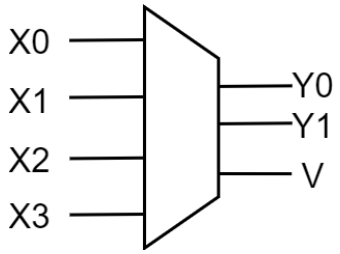


Figura 85 - Codificador de prioridade de 4 para 2.

A tabela verdade apresentada na Tabela 34 permite verificar o comportamento deste componente.

Tabela 34 - Tabela verdade do codificador de prioridade de 4 para 2.

Entradas				Saídas		
X0	X1	X2	X3	Y1	Y0	V
0	0	0	0	0	0	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Observado a Tabela 34 podemos notar que a saída de resultado válido *V* assume 0 se nenhuma entrada está ativa, e assume 1 se alguma entrada estiver ativa. Pode-se também observar que se uma entrada de maior prioridade está ativa, as entradas de menor prioridade são ignoradas.

Este tipo de codificador é mais utilizado pois permite um maior controle sobre o comportamento das entradas, informando inclusive quando as saídas estão válidas ou não.

A Figura 86 apresenta o circuito construído com portas lógicas para um codificador de prioridade de 4 entradas e 2 saídas.

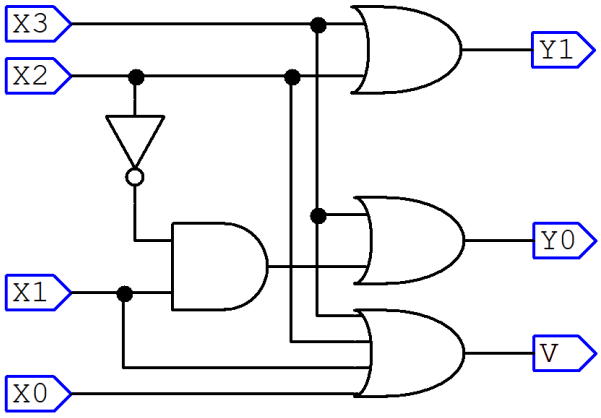


Figura 86 - Circuito do codificador de prioridade de 4 para 2.

Existem diversos tipos e tamanhos de codificadores de prioridade, a Tabela 35 apresenta dois dos mais comumente utilizados.

Tabela 35 - Alguns codificadores de prioridade comerciais.

Circuito integrado	Entradas / saídas	Comentário
74147	9 / 4	Apesar de possuir apenas 9 entradas é considerado um codificador de prioridade decimal com 4 saídas. As entradas e as saídas são ativas em nível lógico 0.
74148	8 / 3	É um codificador de 8 entradas e 3 saídas, tanto as entradas como as saídas são ativas em 0. Ele também possui uma entrada de *enable* ativa em 0 e duas saídas auxiliares. A saída *OE* que é ativa em 0 e indica que todas as entradas estão inativas em 1. E a saída *GS* que também é ativa em 0 e indica que pelo menos uma entrada está ativa em 0.

Mais informações sobre estes componentes, bem como suas tabelas verdade podem ser encontradas em seus *datasheets*.

5.3. Decodificador para *display* de 7 segmentos

Uma categoria de decodificadores bastante útil nos sistemas digitais é a dos decodificadores para *displays* de 7 segmentos. Para entender seu funcionamento é necessário entender primeiro o funcionamento dos *displays* de 7 segmentos.

5.3.1. *Displays* de 7 segmentos

Os *displays* de 7 segmentos, são mostradores de baixo custo, construídos com 7 LEDs na forma de 7 segmentos, que permitem visualizar valores numéricos decimais e hexadecimais. Estes componentes possuem ainda um LED que permite indicar o ponto decimal.

A Figura 87 apresenta um *display* de sete segmentos típico.

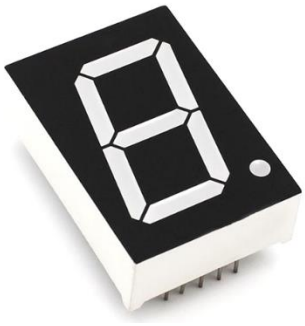


Figura 87 - Display de 7 segmentos.

Para apresentar os números este tipo de *display* liga apenas os LEDs apropriados e mantém ou outros desligados, como pode ser observado na Figura 88.

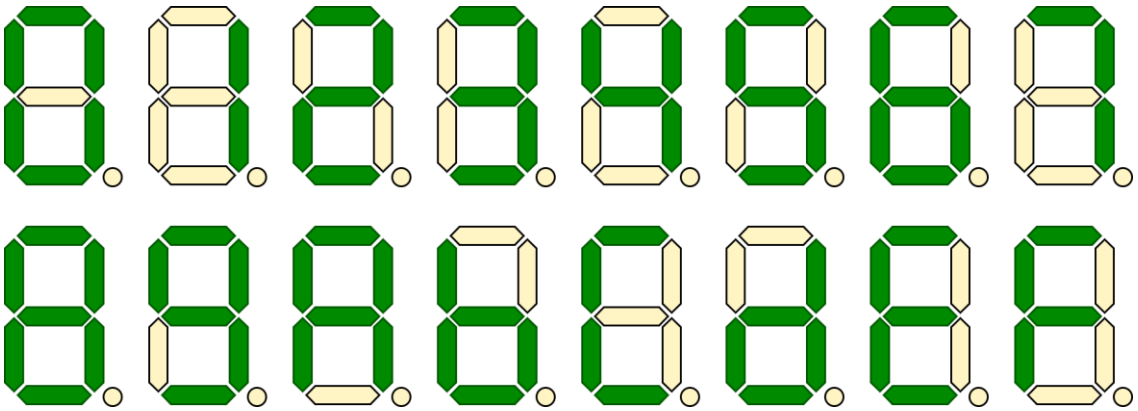


Figura 88 - Representação dos números no *display*.

Para que os sistemas digitais possam utilizar este tipo de *display* para apresentar as informações é necessário que os sinais binários sejam convertidos de forma a energizar os LEDs coretos no display. Para isso o primeiro passo é identificar cada um dos LEDs que compõe o *display*, como pode ser observado na Figura 89.

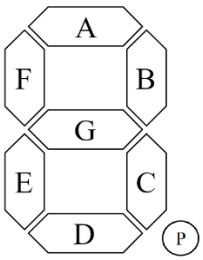


Figura 89 - Identificação dos segmentos do *display*.

Com o *display* devidamente identificado pode utilizar um decodificador para *displays* de 7 segmentos, como veremos a seguir.

5.3.2. Decodificador para *displays* de 7 segmentos

Os decodificadores para displays de 7 segmentos são componentes lógicos digitais que recebem como entrada 4 sinais binários que representam o número que se deseja apresentar no

display e fornecem na saída os 7 sinais correspondentes aos 7 segmentos do display. A Figura 90 apresenta um diagrama genérico deste tipo de componente.

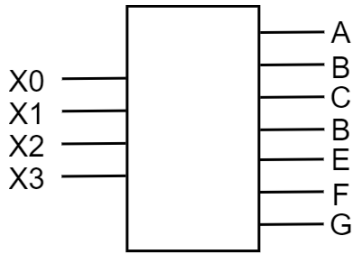


Figura 90 - Decodificador para *display* de 7 segmentos.

A Tabela 36 apresenta a tabela verdade para este componente.

Tabela 36 - Tabela verdade do decodificados para *display* de 7 segmentos.

	Entradas				Saídas						
Número	X3	X2	X1	X0	A	B	C	D	E	F	G
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
A	1	0	1	0	1	1	1	0	1	1	1
B	1	0	1	1	0	0	1	1	1	1	1
C	1	1	0	0	1	0	0	1	1	1	0
D	1	1	0	1	0	1	1	1	1	0	1
E	1	1	1	0	1	0	0	1	1	1	1
F	1	1	1	1	1	0	0	0	1	1	1

Existem várias características elétricas que devem ser consideradas na utilização de displays de 7 segmentos, como a polaridade dos LEDs e os resistores para limitação de corrente deles. Não é objetivo deste material tratar destes detalhes.

Diversos circuitos integrados que implementam a função de decodificador para *display* de 7 segmentos, um dos mais utilizados é o CD4511. O circuito integrado CD4511 é um decodificador para *display* de 7 segmentos decimal ou seja, apresenta os números de 0 a 9. Ele possui saídas ativas em 1 e possui além das 4 entradas de dados também 3 entradas de controle. A entrada *LT* é ativa em 0 e serve para testar os LEDs do *display*, ligando todos ao mesmo tempo. A entrada *BI* também é ativa em 0 e serve para apagar todos os LEDs. E finalmente a entrada *LE* que é ativa em 1 serve para travar o número apresentado no *display*, ignorando as entradas.

Para a operação normal como *driver*, ou decodificador para *display* de 7 segmentos as entradas de controle do circuito integrado CD4511 devem receber os seguintes valores: *LT* = 1, *BI* = 1 e *LE* = 0.

A Figura 91 apresenta a pinagem do circuito integrado CD4511.

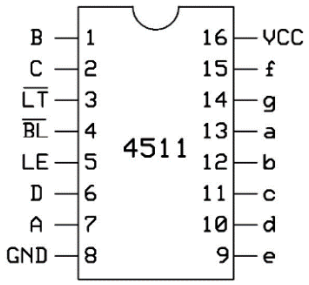


Figura 91 - Pinagem do circuito integrado CD4511.

5.4. Multiplexadores

O termo multiplexação é utilizado para descrever o processo de enviar mais de um sinal através de uma única linha de transmissão. Nos sistemas digitais, os multiplexadores são circuitos lógicos combinacionais projetados para selecionar uma de várias linhas de entrada e direcionar seus sinais para uma única linha de saída.

Assim, um multiplexador, também conhecido como "MUX" é um circuito digital construído com portas lógicas, que possui *n* sinais de controle utilizados para selecionar uma dentre 2^n entradas e transmitir seus sinais para uma saída única. A Figura 92 apresenta um multiplexador com 4 entradas *A*, *B*, *C* e *D*, duas entradas de controle *S0* e *S1* e uma saída *f.* As entradas S0 e S1 são utilizadas para selecionar qual das entradas de sinal vai ser conectadas a saída.

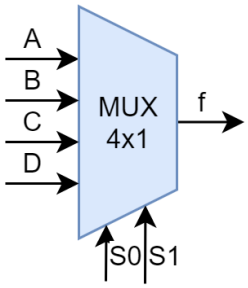


Figura 92 - Multiplexador de 4 para 1.

A Tabela 37 apresenta a tabela verdade para este multiplexador. Nesta tabela é possível observar que as entradas que não estão selecionadas não têm influência na saída.

Tabela 37 - Tabela verdade de um multiplexador de 4 entradas e 1 saída.

Entradas						Saída
D	C	B	A	S1	S0	f
X	X	X	0	0	0	0
X	X	X	1	0	0	1
X	X	0	X	0	1	0
X	X	1	X	0	1	1
X	0	X	X	1	0	0
X	1	X	X	1	0	1
0	X	X	X	1	1	0
1	X	X	X	1	1	1

A Figura 93 apresenta um exemplo de circuito multiplexador construído com portas lógicas.

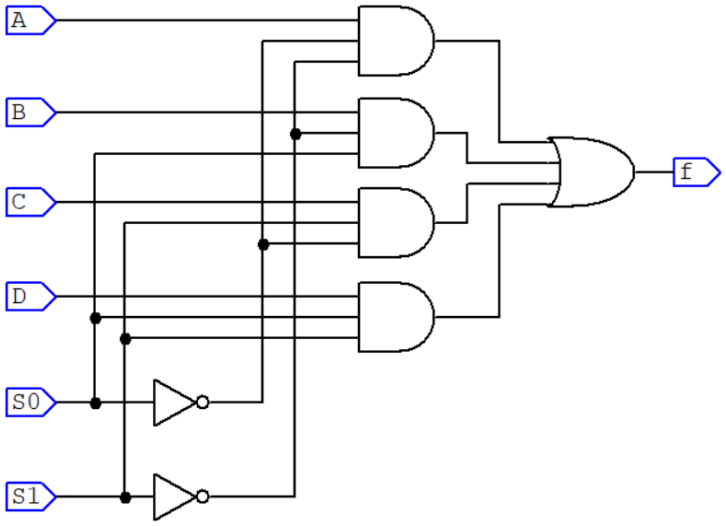


Figura 93 - Circuito multiplexador com portas lógicas.

Existem diversos circuitos integrados comerciais que desempenham a função de multiplexador, dentre eles podemos destacar o circuito 74151 que é um multiplexador de 8 entradas e 1 saída e o 74153 que contém 2 multiplexadores de 4 entradas e 1 saída. A Figura 94 apresenta a pinagem destes circuitos integrados.

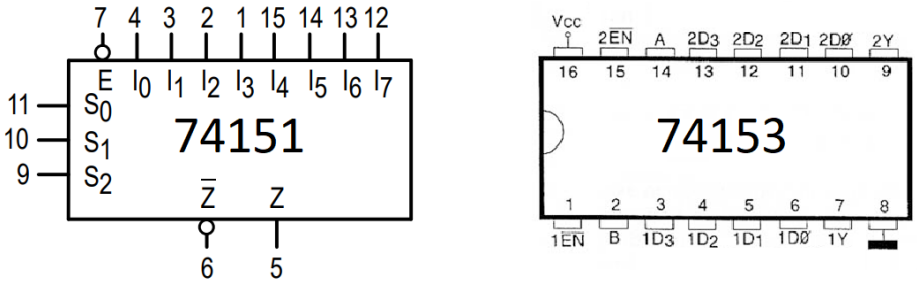


Figura 94 - Circuitos integrados multiplexadores.

5.5. Demultiplexadores

Os demultiplexadores realizam a função contrária dos multiplexadores, ou seja, eles recebem um sinal de entrada e selecionam uma de várias saídas para receber este sinal. Os Demultiplexadores são também chamados de “DEMUX”. A Figura 95 apresenta um Demultiplexador de 1 entrada e 4 saídas.

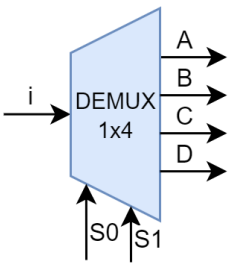


Figura 95 - Demultiplexador de 1 entrada e 4 saídas.

Neste circuito as entradas *S0* e *S1* são utilizadas para selecionar qual das saídas, *A*, *B*, *C* ou *D* vai receber o sinal da entrada *i*.

De maneira geral os demultiplexadores possuem *n* entradas de seleção que ativam uma das 2^n saídas de forma a transmitir o sinal recebido em sua entrada.

A Tabela 38 apresenta a tabela verdade para o Demultiplexador do exemplo. Nesta tabela é possível observar que as saídas que não estão selecionadas permanecem em nível lógico 0.

Tabela 38 - Tabela verdade de um demultiplexador de 1 entrada para 4 saídas.

Entradas			Saídas			
i	S1	S0	A	B	C	D
0	0	0	0	0	0	0
1	0	0	1	0	0	0
0	0	1	0	0	0	0
1	0	1	0	1	0	0
0	1	0	0	0	0	0
1	1	0	0	0	1	0
0	1	1	0	0	0	0
1	1	1	0	0	0	1

A Figura 96 apresenta um circuito digital composto por portas lógicas que desempenha a função de Demultiplexador de 1 entrada para 4 saídas.

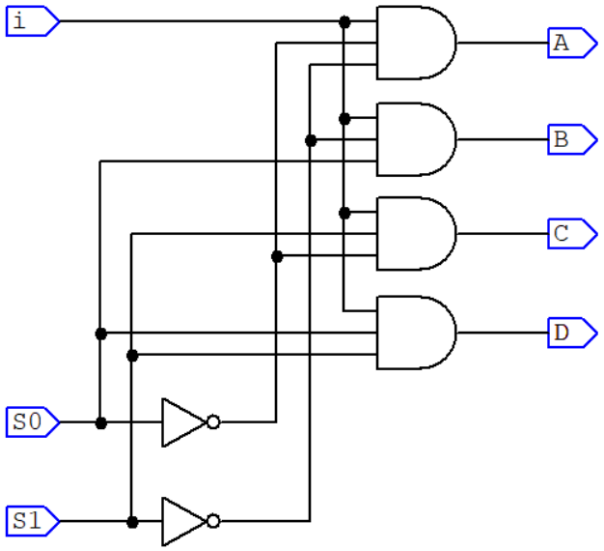


Figura 96 - Circuito de um demultiplexador de 1 entrada e 4 saídas.

Com relação aos circuitos integrados demultiplexadores comerciais existem diversas opções. Os decodificadores apresentados na Tabela 32 podem ser utilizados como demultiplexadores, basta que se use a entrada de "*enable*" como entrada de dados. Pode-se também utilizar circuitos integrados específicos como o 74155 apresentado na Figura 97.

Este circuito integrado comercial possui dois demultiplexadores de 1 entra e 4 saídas. Para selecionar a saída ele possui duas entradas de controle. Para ativar e desativar cada um dos demultiplexadores este circuito integrado possui duas entradas chamadas "*strobe*".

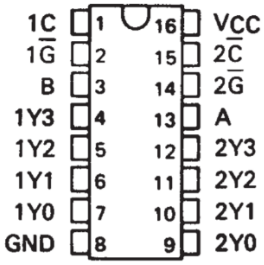


Figura 97 - Circuito integrado demultiplexador.

5.6. Comparadores

Comparadores ou comparadores de magnitude, são circuitos digitais combinacionais que comparam dois números binários para determinar se um destes números é menor, igual ou maior que o outro. Estes circuitos possuem duas entradas A e B de *n* bits a serem comparadas, e possui três saídas, uma para a condição A < B, um para a condição A = B e um para a condição A > B. A Figura 98 apresenta um comparador deste tipo. As entradas A e B podem ter 1 ou mais bits, enquanto que apenas uma das saídas pode assumir o valor verdadeiro de cada vez.

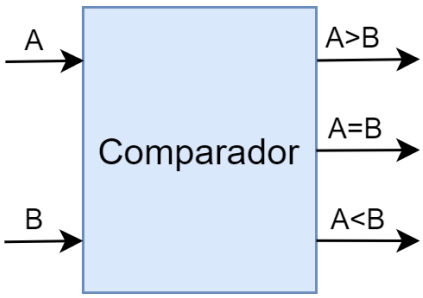


Figura 98 - Comparador de magnitude.

Circuitos comparadores possuem diversas aplicações nos sistemas digitais. Os sistemas computacionais, por exemplo, possuem unidades de processamento que por sua vez possuem comparadores. Estes comparadores são amplamente utilizados na execução das instruções dos programas, permitindo a estes tomar decisões e desta forma realizar suas tarefas.

5.6.1. Comparador de 1 bit

O mais simples dos comparadores compara sinais A e B de apenas 1 bit. Para este comparador as saídas devem respeitar a tabela verdade apresentada na Tabela 39.

Tabela 39 - Tabela verdade do comparador de 1 bit.

Entradas		Saídas		
A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Um circuito combinacional que implementa esta tabela verdade é apresentado na Figura 99.

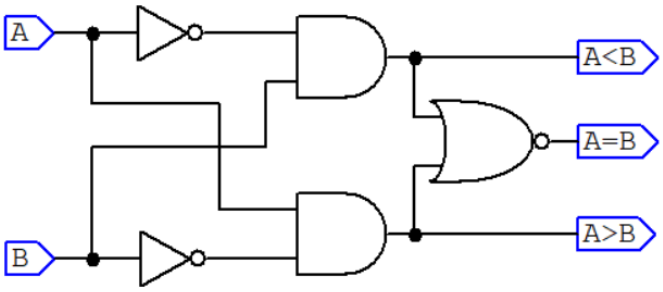


Figura 99 - Circuito comparador de 1 bit.

5.6.2. Comparador de 2 bits

Em um comparador de 2 bits as entradas A e B possuem 2 bits cada. As saídas continuam sendo as mesmas, porém a interpretação das entradas é um tanto mais complexa. Neste caso é necessário levarmos em consideração o peso de cada um dos bis nas entradas, de forma a representar grandezas numéricas binárias. Veja a Figura 100.

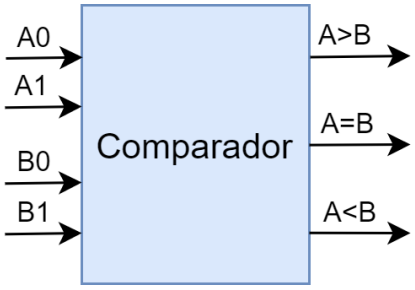


Figura 100 - Comparador de 2 bits.

A implementação deste comparador deve seguir a tabela verdade apresentada na Tabela 40.

Tabela 40 - Tabela verdade do comparador de 2 bits.

Entradas				Saída		
A1	A0	B1	B0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Um circuito que implemente esta tabela verdade pode ser obtido fazendo-se o mapa de Karnaugh para cada uma das saídas e otimizando os circuitos.

A Figura 101 apresenta uma implementação utilizando portas lógicas de um comparador de 2 bits.

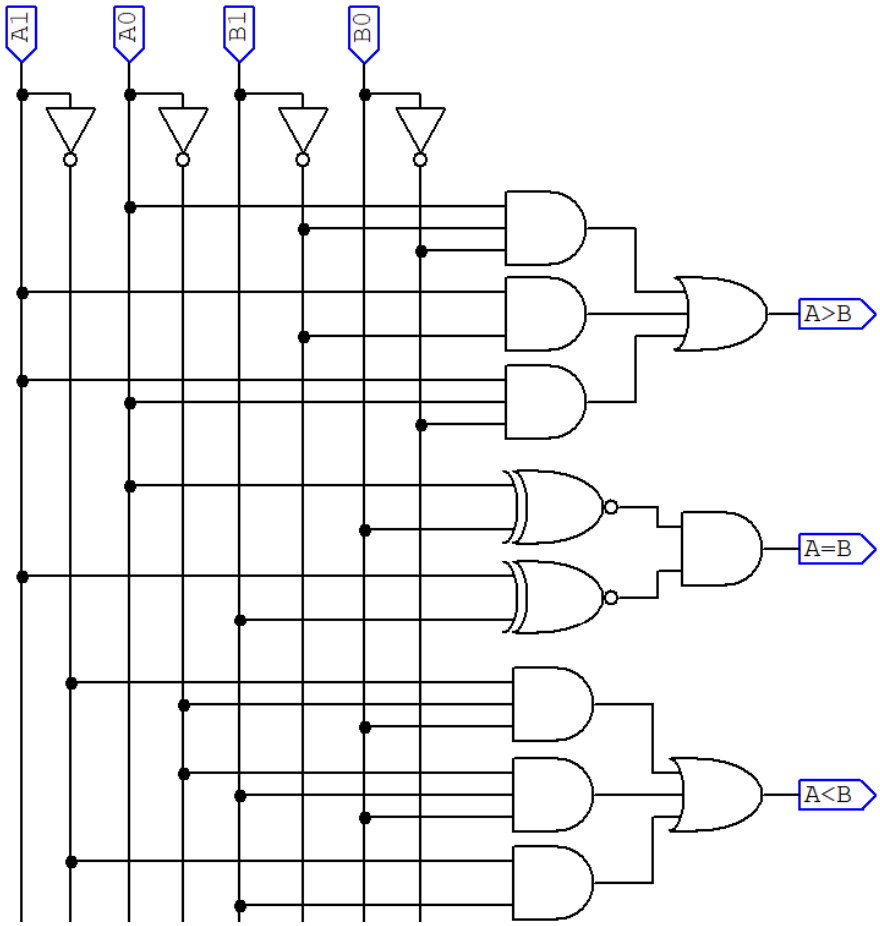


Figura 101 - Circuito comparador de 2 bits.

5.6.3. Comparadores de mais bits

Observando os comparadores anteriores nota-se que conforme o número de bits das palavras de entrada aumenta a complexidade do circuito também aumenta. Para comparadores com entradas de vários bits é conveniente utilizar circuitos comparadores integrados. Um circuito integrado comparador bastante utilizado é o 7485. O 7485 é um comparador para palavras de entrada de 4 bits, que pode ser cascateado para comparar palavras ainda maiores. A Figura 102 apresenta sua pinagem.

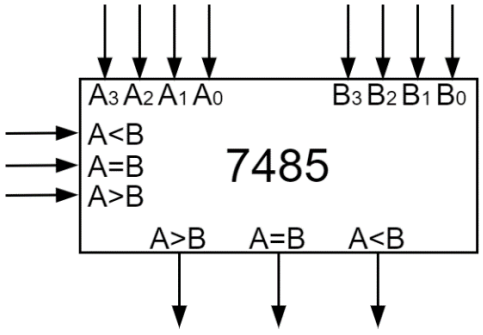


Figura 102 - Circuito integrado comparador 7485.

Observando a figura é possível notar que o circuito integrado 7485 possui duas entradas de dados de 4 bits, ou seja, é um comparador de palavras de 4 bits. Ele possui também as três saídas A<B, A=B e A>B, como esperado. Além disso, ele possui outras três entradas, A<B, A=B e A>B que servem para cascatear vários comparadores 7485. A Figura 103 mostra um arranjo de dois circuitos integrados 7485 para formar um comparador de 8 bits.

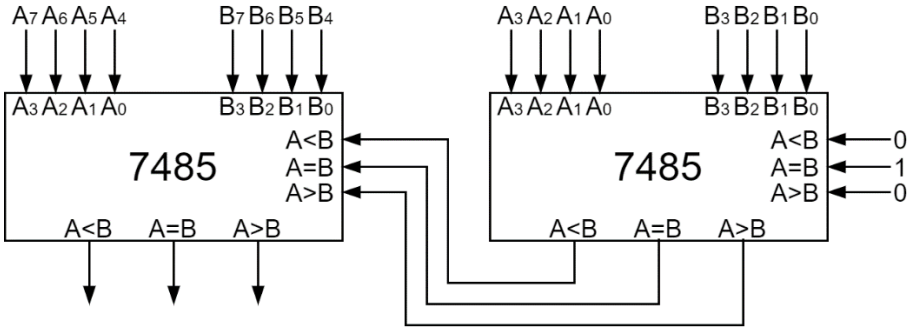


Figura 103 - Associação de comparadores 7485.

É importante destacar que tanto com um comparador 7485 simples quanto em uma associação as entradas auxiliares não utilizadas devem receber os seguintes sinais. A<B = 0, A=B = 1 e A>B = 0, conforme demostrado na figura.

5.7. Exercícios

1) Construa utilizando portas lógicas um decodificador de 3 entradas e 8 saídas. O decodificador deve ter uma entrada de *enable*. As entradas e saídas devem ser ativam em nível lógico 1.
2) Altere o decodificado do exercício anterior para que a entrada de *enable* e as saídas sejam ativas em nível lógico 0.
3) Faça a associação de dois decodificadores 74138 para formar um decodificador de 4 entradas e 16 saídas.
4) Construa utilizando portas lógicas um codificador de 8 entradas e 3 saídas.
5) Implemente um circuito utilizando um display de 7 segmentos e um decodificador para display de 7 segmentos. Utilize chaves nas entradas para gerar os sinais de teste.
6) Construa um circuito contendo um decodificador de 4 entradas e 16 saídas, um codificador de 16 entradas e 4 saídas, um decodificador para *display* de 7 segmentos e um *display* de 7 segmentos. O circuito deve possuir também 4 chaves de entrada, que enviam o sinal para o decodificador, que por sua vez envias os 16 sinais para o codificador que envia os sinais para o decodificador do *display* que finalmente mostra o número no *display*.
7) Construa no simulador, utilizando portas lógicas um multiplexador de 8 entradas e 1 saída.
8) Construa no simulador, utilizando portas lógicas um demultiplexador de 1 entrada e 8 saídas.
9) Construa um circuito no simulador utilizando dois circuitos integrados 7485 que compara uma entrada de 8 bits (pode vir de 8 chaves) com o número 100_{10} também representado em 8 bits. Sé o valor numérico da entrada for maior que 100_{10} um LED deve acender.

Aula 6 - Circuitos lógicos sequenciais

O objetivo desta aula é apresentar aos alunos os principais circuitos lógicos sequenciais, enfatizando sua construção e sua aplicação. Serão abordados circuitos latches, flip-flops, registradores, contadores entre outros.

6.1. Introdução

Como explicado na aula anterior, existem basicamente dois tipos de circuitos lógicos, os circuitos combinacionais e os circuitos sequenciais. Os circuitos combinacionais são aqueles que são construídos com portas lógicas de forma a determinar o valor das saídas apenas em função dos valores atuais das entradas, como na Figura 104.

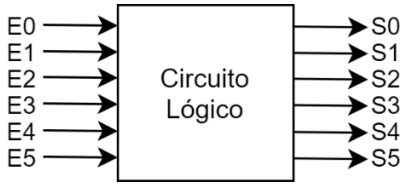


Figura 104 - Circuito lógico combinacional.

Os circuitos sequenciais, que são objeto de estudo desta aula, utilizam elementos de armazenamento de informação, ou como é mais comum falar “memória”, ou ainda algum tipo de realimentação, além das portas lógicas tradicionais, como na Figura 105.

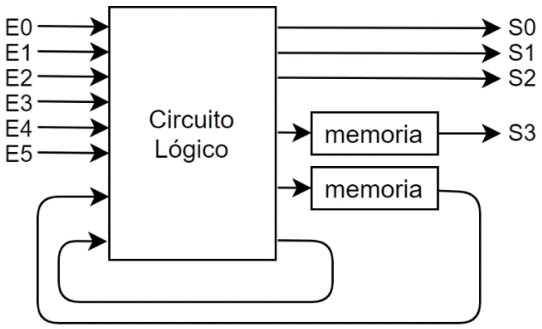


Figura 105 - Circuito lógico sequencial.

Sendo assim, nos circuitos sequenciais, as saídas não dependem apenas das entradas do circuito no momento, nesta categoria de circuitos digitais, as saídas dependem das entradas atuais e das entradas anteriores do circuito, ou seja, dos estados armazenados nos elementos de memória. Assim, pode-se afirmar que as saídas de um circuito sequencial dependem das condições iniciais e do histórico das entradas deste circuito.

A seguir serão apresentados os elementos fundamentais dos circuitos digitais sequenciais, iniciando pelos latches.

6.2. Latches

Os latches são os dispositivos lógicos sequenciais mais básicos, utilizam o princípio da realimentação para armazenar 1 bit. Existem diversos tipos de latches, os principais serão vistos a seguir.

6.2.1. Latch RS

O latch RS é composto por duas portas lógicas "Não OU", conforma Figura 106, e utiliza realimentação para reter a informação.

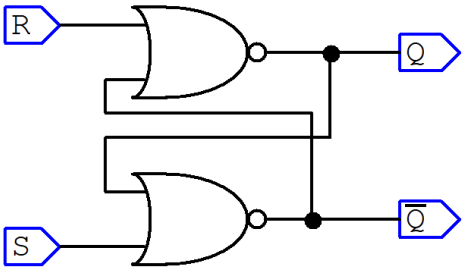


Figura 106 - Circuito do latch RS.

Observando este circuito notamos que as saídas do circuito são utilizadas nas entradas das portas lógicas, a este arranjo chamamos realimentação.

Neste circuito a saída recebe o nome de "Q", enquanto que "$\bar{Q}$" é o complemento da saída, ou a saída invertida. É comum utilizarmos o termo "Setada" para indicar que a saída está em nível lógico 1 e "Resetada" para indicar que a saída está em nível lógico 0. Estes termos foram adaptados das palavras inglesas *set* e *reset*. Da mesma forma, as letras S e R das entradas são derivadas das palavras inglesas *set* e *reset*. Isto se deve ao comportamento destas entradas no circuito. Um sinal 1 na entrada R desliga a saída (reseta), enquanto um sinal 1 na entrada S liga a saída (seta). Observe a tabela verdade deste circuito apresentada na Tabela 41.

Tabela 41 - Tabela verdade do latch RS

	Entradas		Saídas	
	R	S	Q	$\bar{Q}$
	0	0	Não muda	
	0	1	1	0
	1	0	0	1
Evitar ->	1	1	0	0

A diferença do comportamento dos circuitos sequenciais e dos circuitos combinacionais pode ser observada nesta tabela. Quando as entradas R e S estão em zero a saída permanece com o estado anterior, ou seja, não muda. Isso significa que para uma entrada R = 0 e S = 0 a saída Q pode permanecer em 1 ou em 0, dependendo de seu estado anterior, isso não acontecia em circuitos combinacionais.

Como mencionado, colocar a entrada R ou S em um serve para desligar ou ligar a saída Q. Se tentarmos ligar quando a saída já está ligada, ou desligar quando a saída já está desligada, nada acontece.

Acionar as entradas R e S ao mesmo tempo deve ser evitado, pois pode causar resultados indefinidos e oscilações na saída do circuito. Veja as formas de onda da Figura 107.

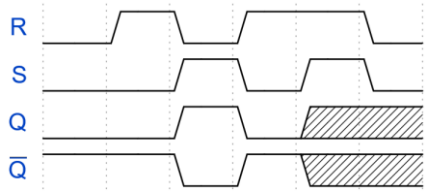


Figura 107 - Formas de onda do latch RS.

Nos circuitos digitais utiliza-se o símbolo apresentado na Figura 108 para representar os latches RS.

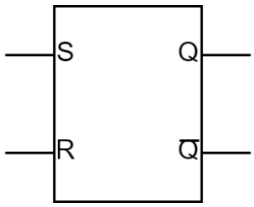


Figura 108 - Símbolo do latch RS.

Como exemplo de circuito integrado comercial que implementa a função de latch RS podemos citar o CD4043, apresentado na Figura 109.

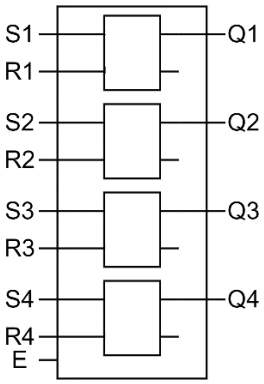


Figura 109 - Circuito integrado latch RS, CD4043.

Trata-se de um circuito integrado que contém 4 latches RS, com uma entrada de "*enable*" que permite ligar ou desligar as saídas Q.

6.2.2. Latch RS Síncrono

O latch RS síncrono, assim como o latch RS é composto por duas portas lógicas "Não OU", porém, ele possui uma entrada de *enable*, que permite o controle dos sinais de entrada, veja a Figura 110.

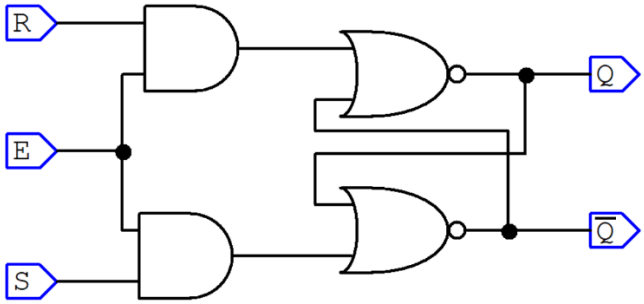


Figura 110 - Circuito do latch RS síncrono.

No latch RS não se tem o controle das entradas, com a adição de um sinal de *enable* é possível controlar quando as entradas R e S acessam o latch.

Enquanto a entrada *enable* permanecer em zero as entradas R e S ficam bloqueadas. Quando a entrada de enable está em nível lógico 1 os sinais de entrada R e S são diretamente transmitidos as portas "Não OU".

A tabela verdade para o latch RS síncrono é apresentada na Tabela 42.

Tabela 42 - Tabela verdade do latch RS síncrono

	Entradas			Saídas	
	Enable	R	S	Q	$\bar{Q}$
	0	X	X	Não muda	
	1	0	0	Não muda	
	1	0	1	1	0
	1	1	0	0	1
Evitar ->	1	1	1	0	0

Assim como no latch RS normal, quando a entrada de *enable* estiver em um, colocar a entrada R ou S em um serve para desligar ou ligar a saída Q. Se tentarmos ligar quando a saída já está ligada, ou desligar quando a saída já está desligada, nada acontece.

Acionar as entradas R e S ao mesmo tempo deve ser evitado, pois pode causar resultados indefinidos e oscilações na saída do circuito. Veja as formas de onda da Figura 111.

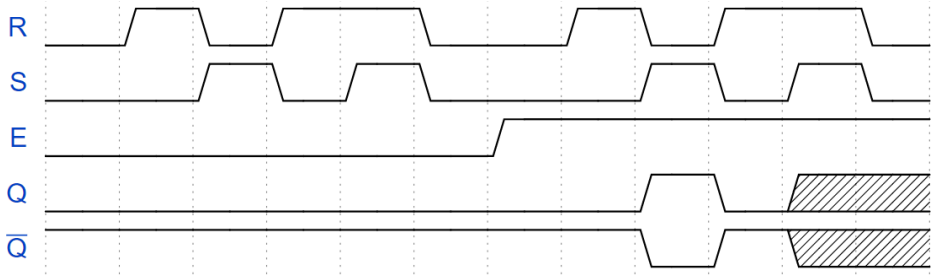


Figura 111 - Formas de onda do latch RS síncrono.

Nos circuitos digitais utiliza-se o símbolo apresentado na Figura 112 para representar os latches RS síncronos.

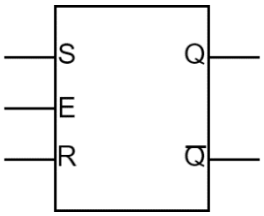


Figura 112 - Símbolo do latch RS síncrono.

6.2.3. Latch RS com entradas diretas

O latch RS com entradas diretas é uma combinação das versões anteriores, ele possui as entradas R e S, controladas pela entrada de *enable*, mas possui também uma entrada chamada "set" que liga a saída independentemente do *enable* e uma entrada chamada "*reset*" que desliga a saída independentemente do *enable*, veja a Figura 113.

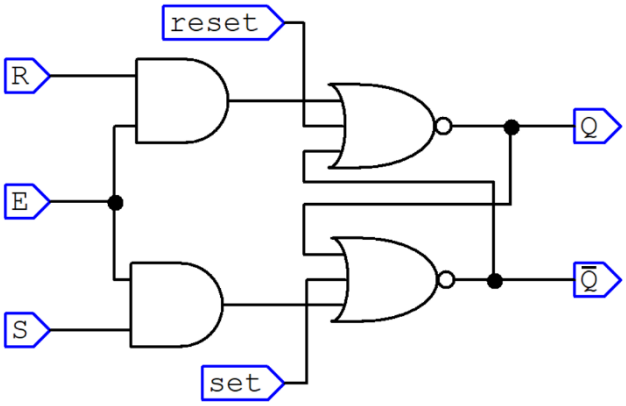


Figura 113 - Circuito do latch RS com entradas diretas.

No latch com entradas diretas é possível escolher se desejamos utilizar o sinal de *enable* ou não, ou ainda, utilizar as duas opções ao mesmo tempo.

Nos circuitos digitais utiliza-se o símbolo apresentado na Figura 114 para representar os latches RS síncronos.

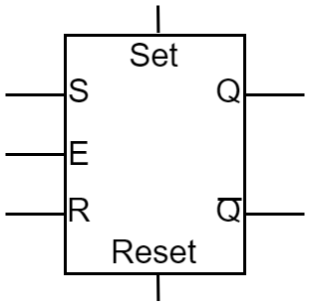


Figura 114 - Símbolo do latch RS com entradas diretas.

6.2.4. Latch D

Como apresentado anteriormente o latch RS e suas variações possui o problema de produzir uma saída indeterminada se as entradas R e S estiverem ativas ao mesmo tempo. Uma alternativa

ao latch RS que evita este tipo de problema é o latch D. No latch D é disponibilizada apenas uma entrada de sinal chamada entrada D, vela o circuito da Figura 115.

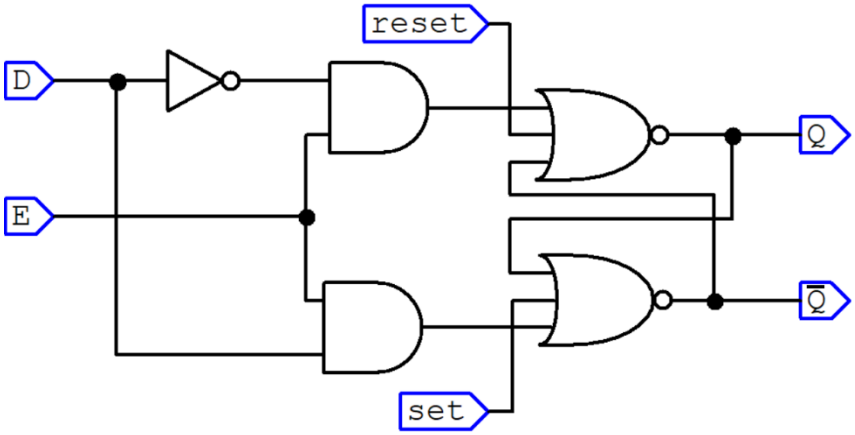


Figura 115 - Circuito do latch D.

Observando o circuito da figura é possível notar que as entradas R e S são combinadas utilizando uma porta inversora e formam a entrada D. Assim, quando a entrada D estiver em 1 e o sinal de *enable* for ativado a saída assume o valor 1, e se a entrada D estiver em 0 e o sinal de *enable* for ativado a saída assume o valor 0. Podemos dizer que se a entrada de *enable* estiver em 1 o sinal da entrada D é armazenado no latch, apresentando o comportamento de uma memória de 1 bit.

A tabela verdade para o latch D é apresentada na Tabela 43.

Tabela 43 - Tabela verdade do latch D

Entradas		Saídas	
Enable	D	Q	$\overline{Q}$
0	X	Não muda	
1	0	0	1
1	1	1	0

Com esta configuração o latch fica mais fácil de ser aplicado nos circuitos, pois a entrada pode facilmente gravada no latch e seu valor fica armazenado até que a entrada de enable seja novamente ativada. Veja a forma de ondas da Figura 116.

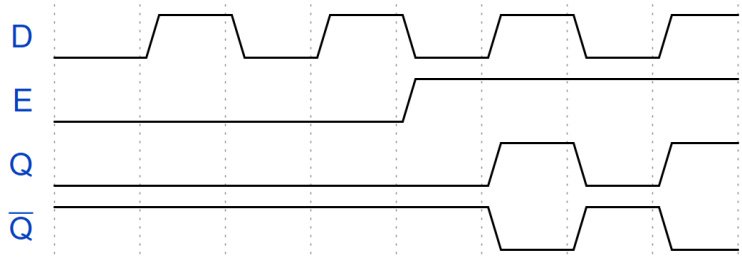


Figura 116 - Formas de onda do latch D.

Nos circuitos digitais utiliza-se o símbolo apresentado na Figura 117 para representar os latches D.

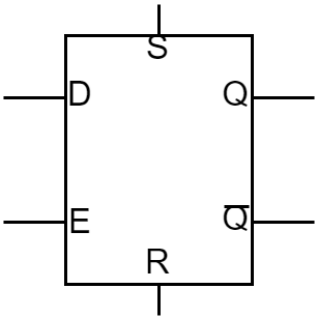


Figura 117 - Símbolo do latch D.

Podemos ainda destacar que o latch D possui entradas diretas S e R para ligar ou desligar a saída independentemente do sinal de *enable*.

Existem vários circuitos integrados que implementam latches D, a Figura 118 mostra o 7475, que possui 4 latches tipo D.

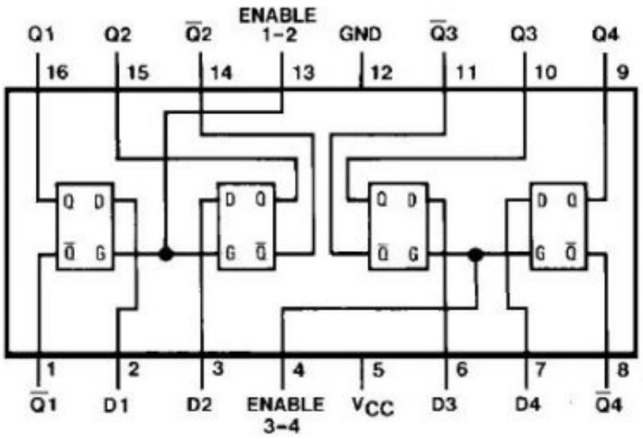


Figura 118 - Circuito integrado latch D 7475.

Outro circuito integrado latch D muito utilizado é o 74573, apresentado na Figura 119.

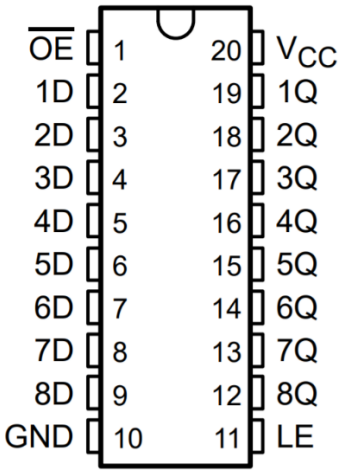


Figura 119 - Circuito integrado latch D 74573.

O circuito integrado 74573 é composto por 8 latches tipo D, com apenas uma entrada de *enable* (LE) comum para todos eles.

Este circuito integrado possui também um sinal de *output enable* (OE), ativo em 0, que serve para controlar as saídas do circuito. Quando o sinal OE está desativado as saídas ficam desligadas. Nesta situação as saídas não assumem nem 0 nem 1, ficando em um chamado terceiro estado, que na prática é como se as saídas não estivessem conectadas a nada (um circuito aberto).

6.3. Flip-Flops

Os flip-flops são uma evolução dos latches, diferenciando-se deles em função de seu sinal de ativação.

6.3.1. Flip-flop D

Os latches apresentados anteriormente possuem uma entrada de *enable* que é ativa por nível lógico, 0 ou 1. Em algumas situações este comportamento não é apropriado. Para estes casos foi desenvolvida uma nova categoria de circuitos com sinal de *enable* ativo apenas na borda de transição, ou seja, na passagem de 0 para 1 ou de 1 para 0. Este tipo de sinal é chamado de ***clock***. A Figura 120 apresenta sinais típicos de *clock* destacando a borda de subida ou descida onde o sinal é ativo.

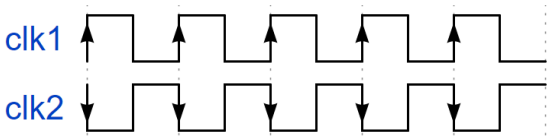


Figura 120 - Sinais típicos de *clock*.

Neste contexto, o primeiro componente digital que utiliza sinal de *clock* que iremos estudar é o flip-flop D. Trata-se de um latch D em que a entrada de *enable* é substituída por um sinal de *clock*. A Figura 121mostra a implementação de um flip-flop D com portas lógicas.

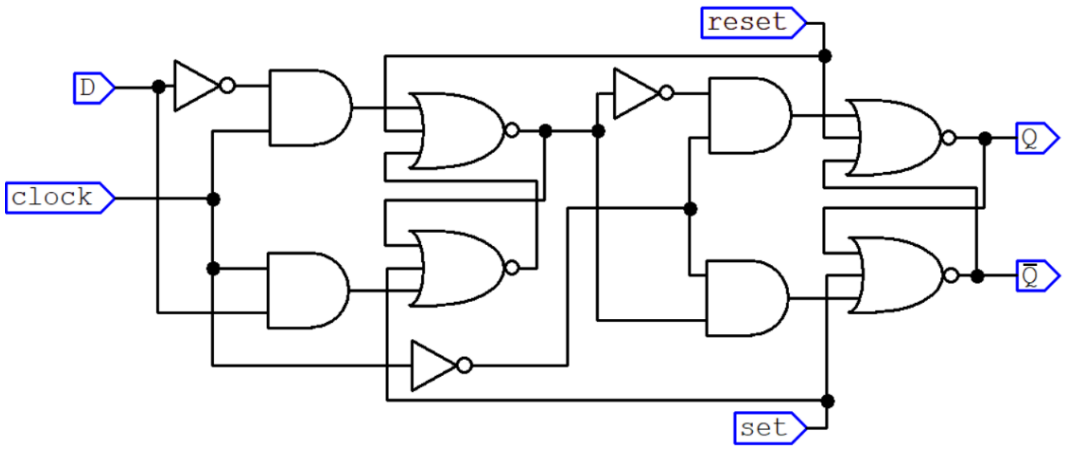


Figura 121 - Circuito de um flip-flop tipo D.

Observando este circuito podemos notar que o flip-flop D é composto por 2 latches D ligados em série, com os sinais de *enable* interligados através de uma porta inversora.

O funcionamento deste circuito é simples, quando o sinal de *clock* passa de 1 para 0, ou seja, na borda de descida, o sinal da entrada D é transferido para a saída Q e assim armazenado no flip-

flop D. As entradas *set* e *reset* servem respectivamente para ligar e desligar a saída Q independentemente do *clock*.

Assim podemos dizer que a entrada D é uma entrada síncrona, enquanto as entradas *set* e *reset* são assíncronas.

A tabela verdade para o flip-flop D é apresentada na Tabela 44.

Tabela 44 - Tabela verdade do flip-flop D

Entradas		Saídas	
clock	D	Q	$\bar{Q}$
0	X	Não muda	
1	X	Não muda	
↓	0	0	1
↓	1	1	0

Observe que não existe mudança da saída para níveis de *clock* iguais a 0 ou 1, ou ainda para a borda de subida deste sinal. A mudança na saída acontece apenas na borda de descida do *clock*.

A forma de ondas para o flip-flop D é apresentada na Figura 122.

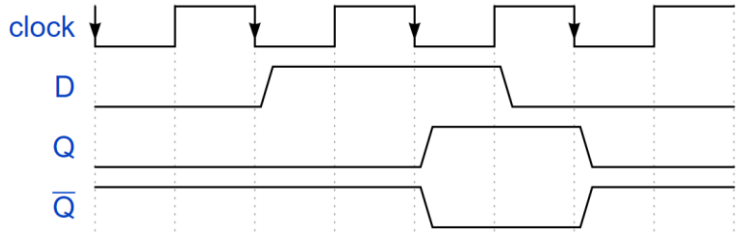


Figura 122 - Formas de onda do flip-flop D.

Nos circuitos digitais utiliza-se o símbolo apresentado na Figura 123 para representar os flip-flops D.

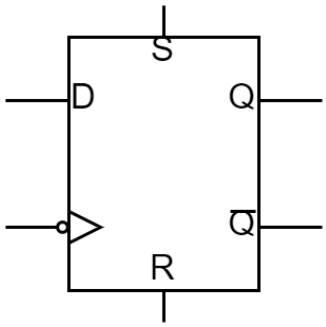


Figura 123 - Símbolo do flip-flop D.

É importante destacar que as entradas de *clock* possuem uma simbologia diferente nos componentes. É adicionado um pequeno triangulo a este tipo de entrada.

A Figura 124 (a) mostra uma entrada de *clock* ativa na borda de subida, enquanto a Figura 124 (b) mostra uma entrada de *clock* ativa na borda de descida.

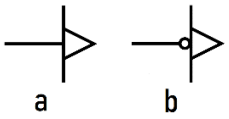


Figura 124 - Simbologia utilizada nos sinais de *clock*.

Existem vários circuitos integrados que implementam flip-flop D, a Figura 125 Figura 118mostra o 7474, que possui 2 flip-flops tipo D.

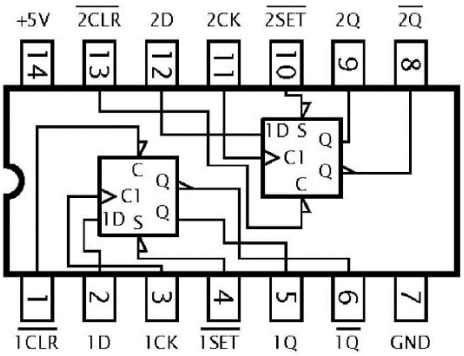


Figura 125 - Circuito integrado flip-flop D 7474.

Neste circuito integrado o *reset* é chamado de *clear* (CLR).

6.3.2. Flip-flop JK

Outro componente digital que utiliza sinal de *clock* para sincronizar suas operações é o flip-flop JK. Trata-se de um flip-flop de duas entradas chamadas de J e K. A Figura 126 mostra a implementação de um flip-flop JK com portas lógicas.

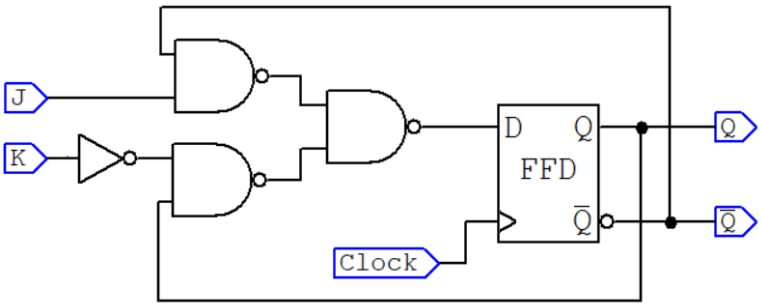


Figura 126 - Circuito de um flip-flop tipo JK.

A tabela verdade para o flip-flop JK é apresentada na Tabela 45

Tabela 45 - Tabela verdade do flip-flop JK

Entradas			Saídas	
clock	J_n	K_n	Q_{n+1}	$\bar{Q}_{n+1}$
↓	0	0	Q_n	$\bar{Q}_n$
↓	0	1	0	1
↓	1	0	1	0
↓	1	1	$\bar{Q}_n$	Q_n

Observando este circuito e sua tabela verdade podemos notar que o flip-flop JK permite através de suas entradas J e K que se tenha um controle completo da saída. É possível manter a saída estática, com J=0 e K=0, é possível desligar a saída, com J=0 e K=1, é possível ligar a saída, com J=1 e K=0 e é possível inverter a saída, com J=1 e K=1. Esta versatilidade torna o flip-flop JK útil para uma grande variedade de sistemas digitais. É importante salientar que todas estas mudanças acontecem apenas na boda do sinal de *clock*.

É possível entender melhor seu funcionamento na forma de ondas apresentada na Figura 127.

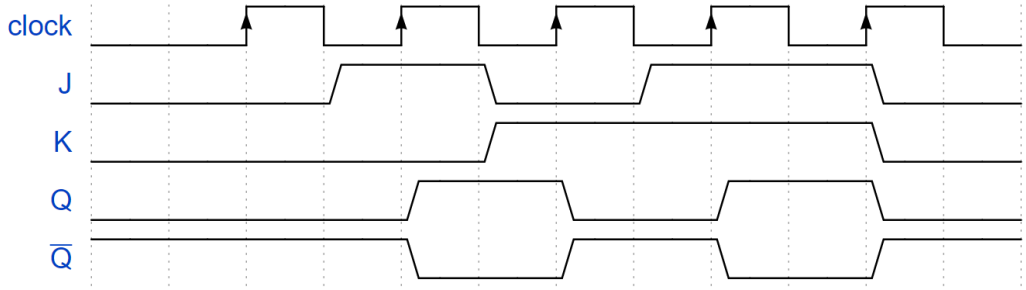


Figura 127 - Formas de onda do flip-flop JK.

Nos circuitos digitais utiliza-se o símbolo apresentado na Figura 128 para representar os flip-flops JK.

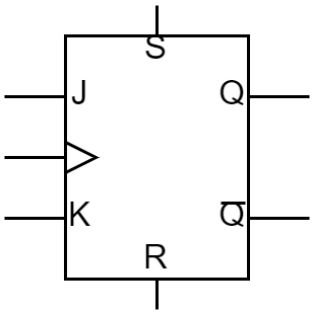


Figura 128 - Símbolo do flip-flop JK.

Existem vários circuitos integrados que implementam flip-flop JK, a Figura 129 mostra o 7476, que possui 2 flip-flops tipo JK.

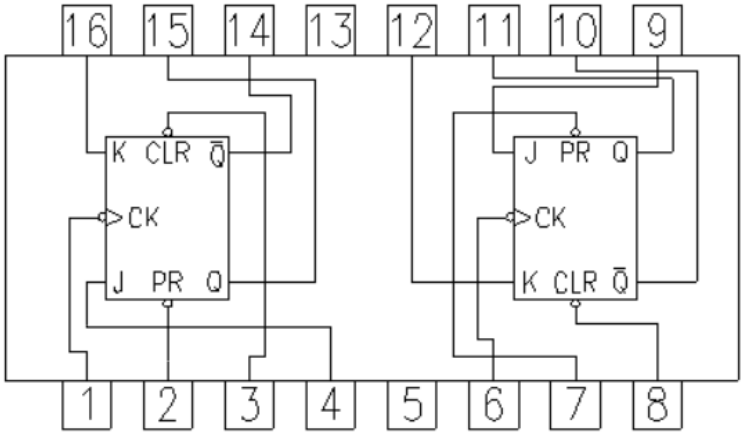


Figura 129 - Circuito integrado flip-flop JK 7476.

Neste circuito integrado o *set* é chamado de *preset* (PR) e o *reset* é chamado de *clear* (CLR), ambos ativos em 0.

6.4. Registradores

Os circuitos sequenciais vistos até aqui permitem armazenar 1 bit de informação. Porém em muitas situações é necessário armazenar quantidades maiores de informação. Para este fim foram criados os registradores. Nos registradores o armazenamento das informações é feito através de um arranjo conveniente de flip-flops. Conforme este arranjo, os flip-flops podem armazenar todos os dados de entrada em um único pulso de *clock*, neste caso o registrador é chamado de registrador paralelo. Em um registrador serial, por sua vez, é necessário um pulso de clock para cada bit de informação que desejamos armazenar. A seguir veremos estes dois tipos de registradores.

6.4.1. Registrador paralelo

Em um registrador paralelo, as informações são armazenadas todas de uma única vez, com um único pulso de clock. O diagrama da Figura 130 apresenta um registrador paralelo de 4 bits.

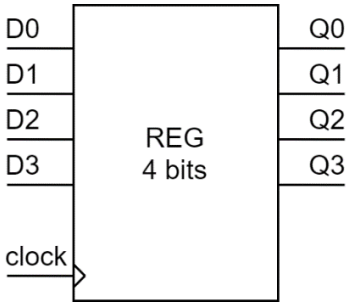


Figura 130 - Registrador paralelo de 4 bits.

O circuito interno de um registrador paralelo de 4 bits é apresentado na Figura 131.

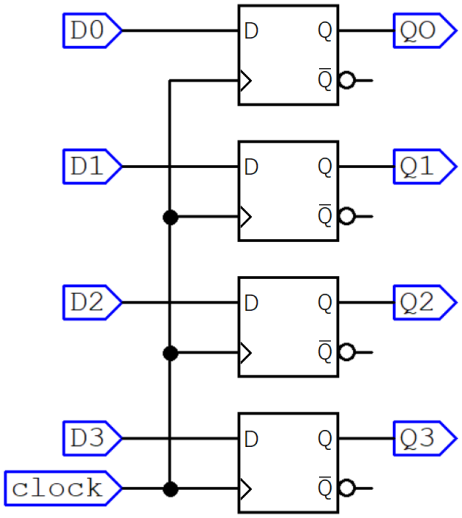


Figura 131 - Circuito interno de um registrador paralelo.

Observando o circuito é fácil entender o funcionamento de um registrador paralelo, na borda do sinal de *clock* todos os flip-flops armazenas os sinais das entradas. Estes sinais são então apresentados nas saídas.

Existem diversos circuitos integrados que implementam registradores paralelos, com diferentes configurações e diferentes números de bits. Um dos principais é o 74273, que implementa um registrador paralelo de 8 bits, com um sinal de reset. A Figura 132 apresenta este circuito integrado.

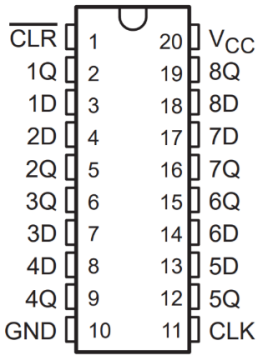


Figura 132 - Circuito Integrado registrador paralelo de 8 bits, 74273.

O circuito interno deste circuito integrado é apresentado na Figura 133.

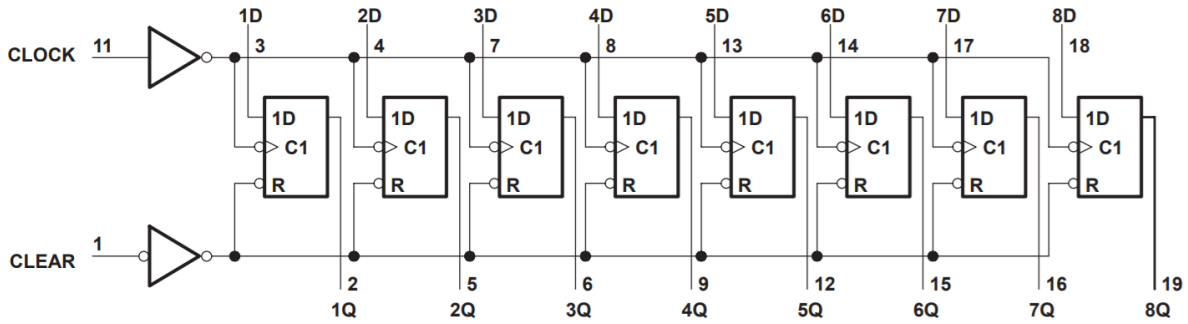


Figura 133 - Circuito interno do 74273.

Outro circuito integrado que implementa um registrador paralelo é o 74173, a Figura 134 apresenta este circuito.

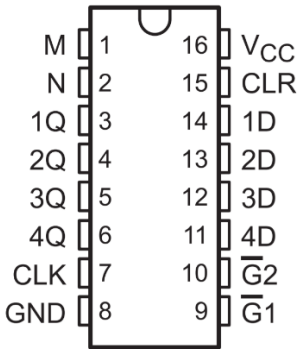


Figura 134 - Circuito Integrado registrador paralelo de 4 bits, 74173.

Este circuito integrado implementa um registrador paralelo de 4 bits, com um sinal de *reset* (CLR).

Também são disponibilizados 2 sinais de *enable* ($\overline{G1}$ *e* $\overline{G2}$), ativos em 0, que permitem controlar a entrada de dados nos flip-flops do registrador.

Os sinais $\overline{M}$ e $\overline{N}$ permitem controlar as saídas dos flip-flops, quando ambos estão em 0 as saídas apresentam os dados armazenados, qualquer outra combinação deixa a saída desconectada (terceiro estado).

A Figura 135 apresenta o circuito interno do 74173.

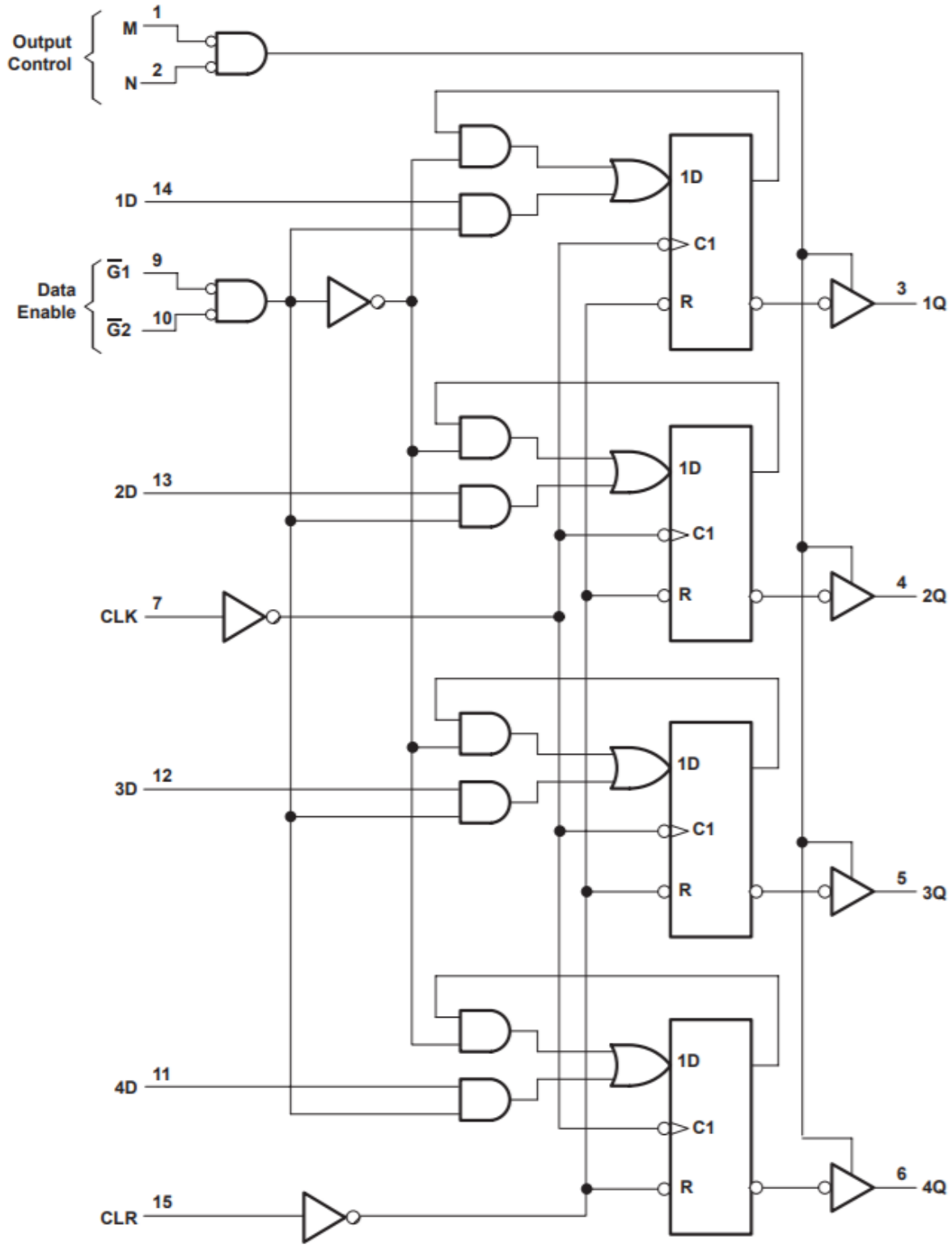


Figura 135 - Circuito interno do 74173.

6.4.2. Registrador série

Em um registrador série os flip-flops são arranjados de forma diferente, permitindo que a informação seja armazenada um bit por vez. A Figura 136 apresenta um registrador série de 4 bits.

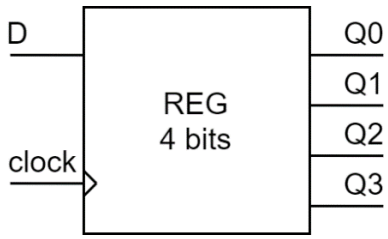


Figura 136 - Registrador série de 4 bits.

Em um registrador série existe apenas um pino de entrada, assim a informação é gravada um bit por vez, em série, por isso o nome registrador série. Este registrador é também conhecido como *shift-register*, pois conforma cada bit é armazenado os demais bits são deslocados. A Figura 137 mostra o circuito interno de um registrador série.

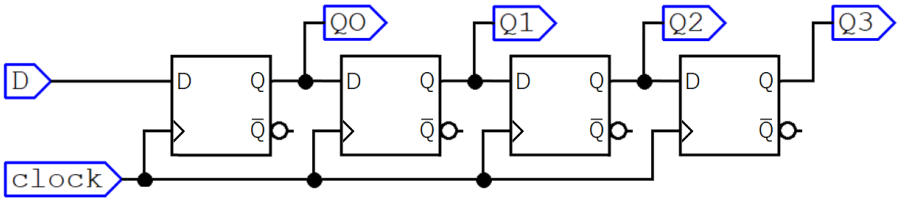


Figura 137 - Circuito interno de um registrador série.

Observando a figura notamos que a cada pulso de *clock* os dados armazenados são deslocados de um flip-flop para outro, e um novo dado da entrada D é armazenado. Ou seja, o dado armazenado na saída Q3 é perdido, e Q3 recebe o valor de Q2. A saída Q2, recebe o valor da saída Q1. A saída Q1 recebe o valor da saída Q0. E finalmente a saída Q0 recebe o valor da entrada D.

É importante salientar que todas estas transições acontecem ao mesmo tempo, de forma síncrona.

Existem diversos circuitos integrados que implementam registradores série, um dos principais é o 74164, que implementa um registrador série de 8 bits, com um sinal de reset ($\overline{CLR}$). A Figura 138 apresenta este circuito integrado.

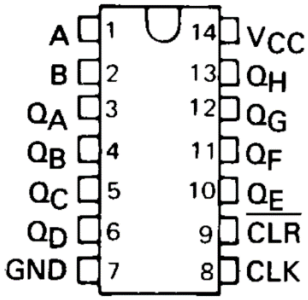


Figura 138 - Circuito integrado registrador série 74164.

Este circuito possui duas entradas, A e B, que executam a função “E” nos dados de entrada a serem armazenados, ou seja, para armazenar o bit 1 no registrador A e B devem ser 1.

6.4.3. Registrador série e paralelo

Existem registradores que combinam as duas funções, série e paralelo, permitindo selecionar qual o comportamento desejado através de uma entrada adicional. A Figura 139 mostra o circuito integrado 7495 como um exemplo.

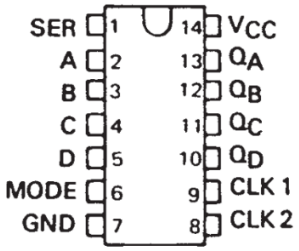


Figura 139 - Circuito integrado 7495.

A Figura 140 mostra o diagrama interno deste registrador.

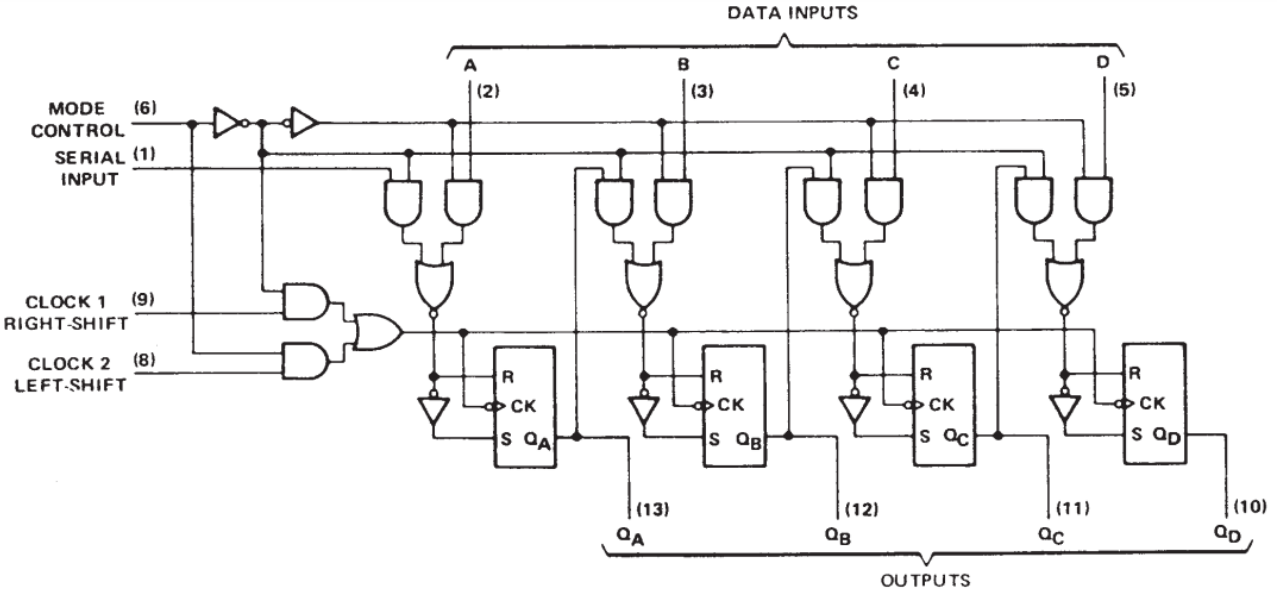


Figura 140 - Circuito interno do 7495.

Neste diagrama é possível notar que o circuito integrado possui uma entrada paralela (pinos A, B, C e D) e uma entrada serial. Possui também dois sinais de *clock* e um sinal de modo de operação.

Este circuito pode ser operado como um registrados série ou paralelo, dependendo da configuração escolhida, podendo inclusive, com o auxílio de ligações externas, deslocar os bits em ambas as direções.

Para aumentar o número de bits armazenados no registrador é possível fazer a associação de vários deste componente.

Veja o *datasheet* do componente para mais detalhes.

Outro circuito integrado interessante quando falamos de registradores série é o 74165, que pode ser visto na Figura 141.

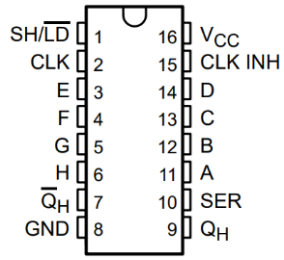


Figura 141 - Circuito integrado 74165.

Este circuito integrado possui o diagrama interno apresentado na Figura 142.

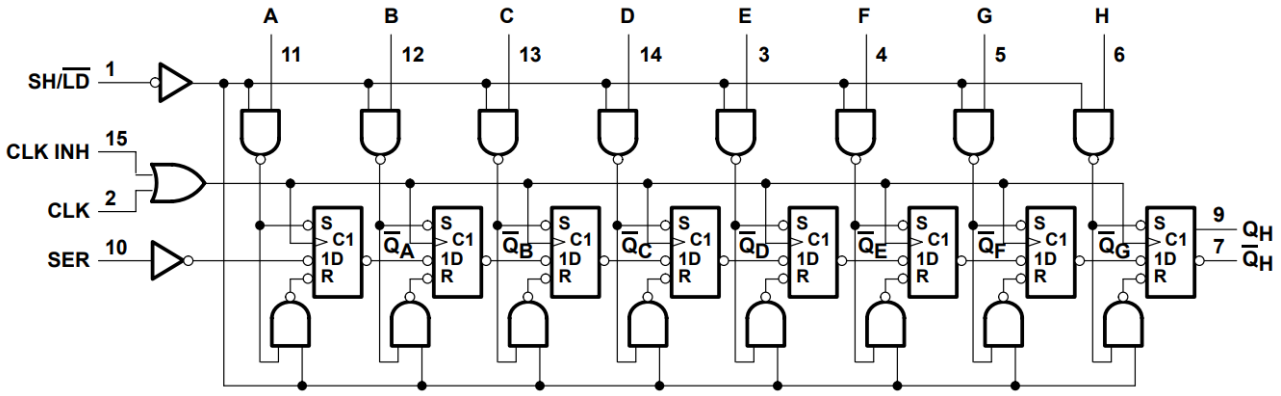


Figura 142 - Circuito interno do 75165.

Analisando o circuito interno do 75165 podemos notar que este circuito interno possui uma entrada paralela de 8 bits e uma saída série de apenas 1 bit. Existem também sinais de controle que permitem escolher entre a função de carregamento paralelo ou saída serial do componente.

6.5. Contadores

É comum utilizarmos a ideia de contagem nos mais variados sistemas, por exemplo, quando contamos o número de peças produzidas por uma máquina ou quando contamos o número de caixas transportadas por uma esteira. Os sistemas digitais são muitas vezes utilizados para realizar este tipo de contagem. Os circuitos digitais responsáveis por esta tarefa são chamados de contadores. Os principais tipos de contadores serão estudados nesta seção.

Assim como os registradores, os contadores são construídos com flip-flops, e o arranjo destes flip-flops define o tipo do contador.

Os contadores possuem uma entrada de contagem, que normalmente é a entrada de *clock*, e “n” saídas, cujo valor binário representa a saída atual do contador, veja a Figura 143 onde é apresentado um contador de 4 bits.

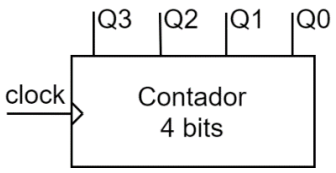


Figura 143 - Contador de 4 bits.

Algumas definições são necessárias para o estudo dos contadores, veja a seguir.

- **Estado:** é a combinação atual na saída de um contador.
- **Módulo:** Número de estados diferentes que a saída pode assumir.
- **Contador assíncrono:** contador onde os sinais de *clock* dos flip-flops internos são diferentes uns dos outros.
- **Contador síncrono:** contador onde existe apenas um sinal de *clock* para todos os flip-flops.
- **Sequência de contagem:** pode ser crescente (0, 1, 2, 3 ...), decrescente (... 3, 2, 1, 0) ou qualquer.

Outra característica importante dos contadores é a relação entre o número de bits (saídas) e o número de estados. Um contador de *n* bits pode ter até 2^n estados. Por exemplo, um contador de 3 bits pode ter até $2^3 = 8$ estados, contando assim de 0 a 7. Dependendo do contador nem todos os estados possíveis são permitidos ou utilizados.

6.5.1. Contador assíncrono crescente

Contadores assíncronos crescentes são o tipo mais simples de contador, onde um conjunto de flip-flops JK são arranjados de forma que a saída de um sirva de *clock* para o próximo. As entradas J e K são fixas em nível lógico 1, veja a Figura 144 para um exemplo de 4 bits.

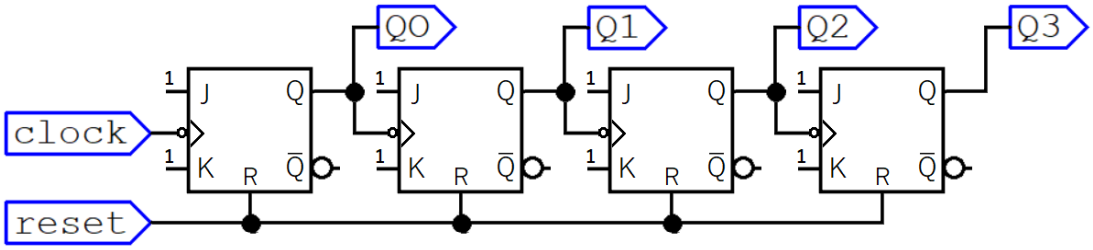


Figura 144 - Circuito de um contador assíncrono crescente.

É importante destacar que a saída Q0 é o bit menos significativo e a saída Q3 é o bit mais significativo. O sinal de *reset* serve para colocar o contador em seu estado inicial, neste caso, 0.

A Figura 145 apresenta as formas de onda para um contador crescente de 4 bits, com estado inicial 0.

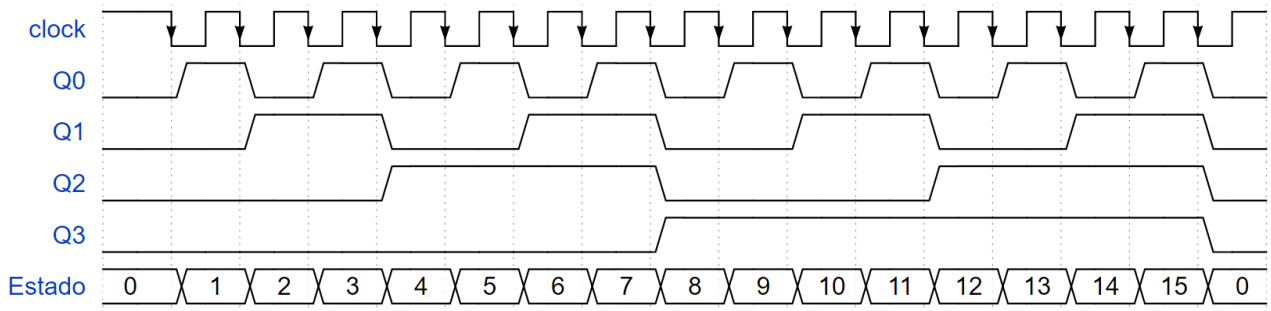


Figura 145 - Formas de onda de um contador crescente.

Observe que após o décimo quinto pulso na entrada o contador retorna ao estado inicial e recomeça a contagem.

Existem diversos circuitos integrados que implementam a função de contador assíncrono crescente. Um dos mais utilizados é o 7493, apresentado na Figura 146.

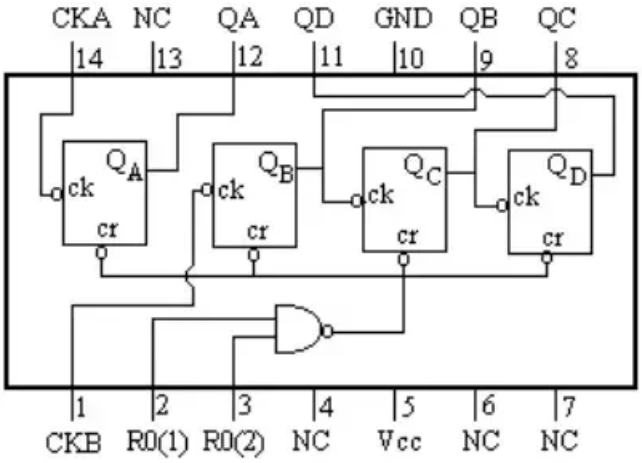


Figura 146 - Circuito integrado 7493.

É importante destacar que neste componente a saída do primeiro flip-flop não está conectada a entrada do segundo flip-flop. Assim para que se possa utilizá-lo como contador de 4 bits esta ligação deve ser realizada externamente.

6.5.2. Contador assíncrono crescente decimal

É muito comum nos projetos de sistemas decimais que se deseje contar de forma decimal, ou seja, de 0 a 9. Assim é necessário fazer uma alteração em um contador de 4 bits para que ele conte de 0 a 9 e não de 0 a 15. Veja o circuito da Figura 147.

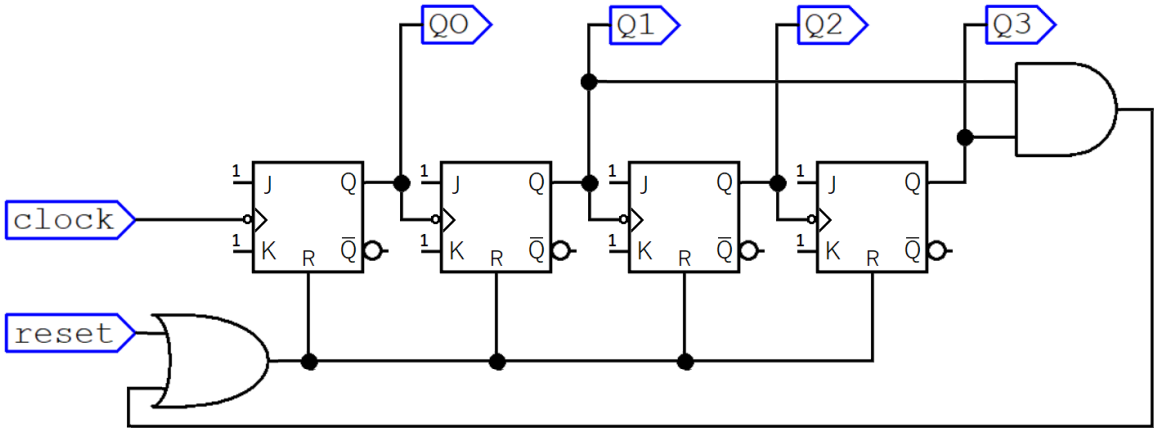


Figura 147 - Circuito de um contador decimal.

Neste circuito quando a contagem ultrapassa o número 9 o contador é resetado voltando a 0. A porta E executa esta operação.

Quando o contador passa do número 9 para o 10, as saídas Q1 e Q3 assumem valor 1, assim a porta E envia nível 1 as entradas de reset dos flip-flops, zerando todas as saídas. Desta forma, no primeiro pulso de *clock* após chagar ao número 9 o contador volta instantaneamente a 0.

Veja as formas de onda da Figura 148.

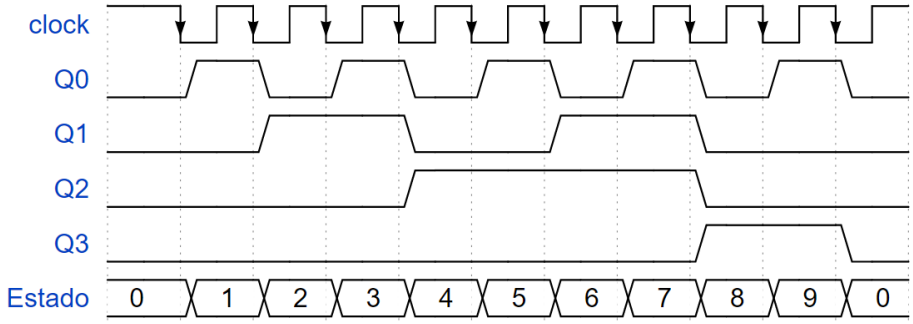


Figura 148 - Formas de onda de um contador decimal.

O circuito integrado 7490 é um dos contadores decimais mais utilizados, a Figura 149 mostra este componente.

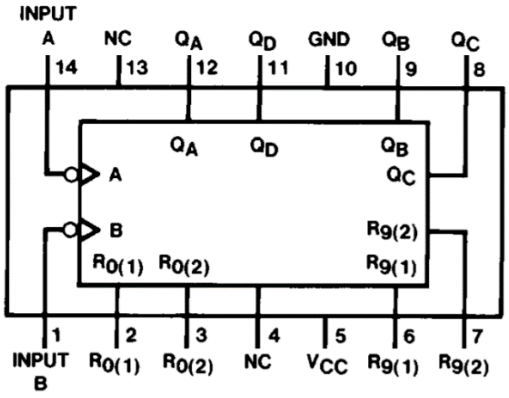


Figura 149 - Circuito integrado 7490.

Este componente possui duas entradas de clock, para operar como contador decimal a saída QA deve ser conectada a entrada de *clock* B. O *datasheet* explica melhor seu funcionamento.

6.5.3. Contador assíncrono decrescente

Contadores assíncronos podem também ser decrescentes, basta mudar o arranjo de flip-flops JK de forma que a saída invertida de um sirva de *clock* para o próximo. As entradas J e K continuam fixas em nível lógico 1, veja a Figura 150 para um exemplo de 4 bits.

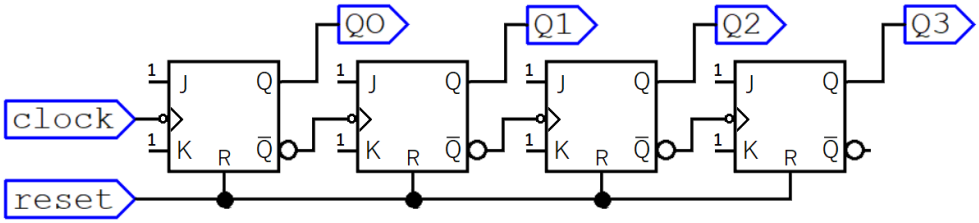


Figura 150 - Circuito de um contador assíncrono decrescente.

A saída Q0 continua sendo o bit menos significativo e a saída Q3 é o bit mais significativo. O sinal de *reset* serve para colocar o contador em seu estado inicial, neste caso também 0.

A Figura 151 apresenta as formas de onda para um contador crescente de 4 bits, com estado inicial 0.

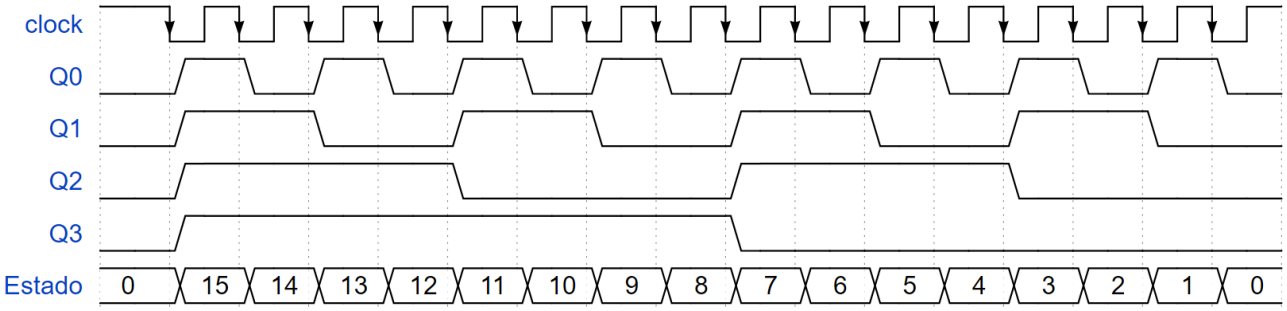


Figura 151 - Formas de onda de um contador decrescente.

Observe que após o décimo quinto pulso na entrada o contador retorna ao estado inicial e recomeça a contagem.

6.5.4. Problemas com contadores assíncronos

Os contadores assíncronos possuem uma deficiência intrínseca a sua construção, que impede sua utilização em alguns tipos de aplicações. Esta característica está relacionada ao atraso de propagação dos sinais nos flip-flops. A Figura 152 apresenta as formas de onda de um contador assíncrono considerando os atrasos de propagação dos sinais nos flip-flops. Observe que entre os sinais de saída desejados (números 0, 1, 2, 3, 4, 5, 6...) surgem temporariamente estados indesejados.

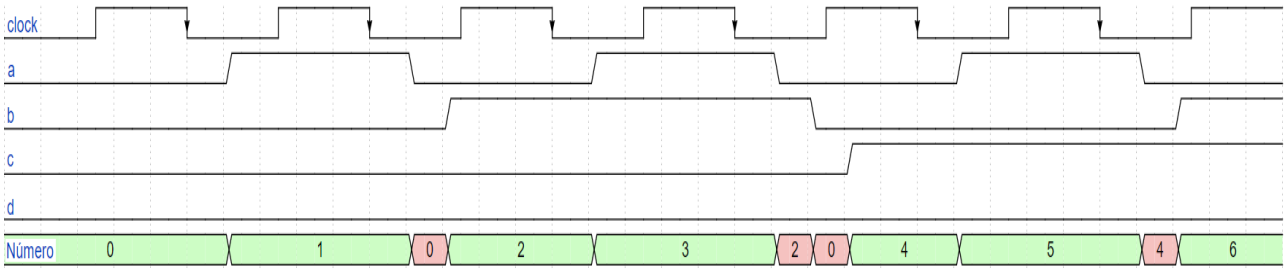


Figura 152 - Formas de onda contador assíncrono com atraso.

Entre os números 1 e 2 por exemplo, surge rapidamente o número 0, entre os números 3 e 4, surgem rapidamente os números 2 e 0, e assim por diante. Para muitas aplicações estes sinais indesejados são tão rápidos que não chegam a interferir no funcionamento do sistema.

Porém, em algumas aplicações estes sinais podem comprometer o funcionamento do sistema, e nestes casos é necessário utilizar um outro tipo de contador, o contador síncrono.

6.5.5. Contador síncrono crescente

Os contadores síncronos são projetados de forma a não produzir saídas indesejadas nas transições entre os números, como acontece nos contadores assíncronos. A Figura 153 apresenta o circuito de um contador síncrono de 4 bits. Observe que o sinal de *clock* neste circuito é conectado a todos os flip-flops. Desta forma, todos os flip-flops são ativados no mesmo instante e todas as saídas são atualizadas também no mesmo instante.

Como todas as saídas são atualizadas no mesmo instante podemos dizer que as saídas estão sincronizadas, por isso este contador é considerado um contador síncrono.

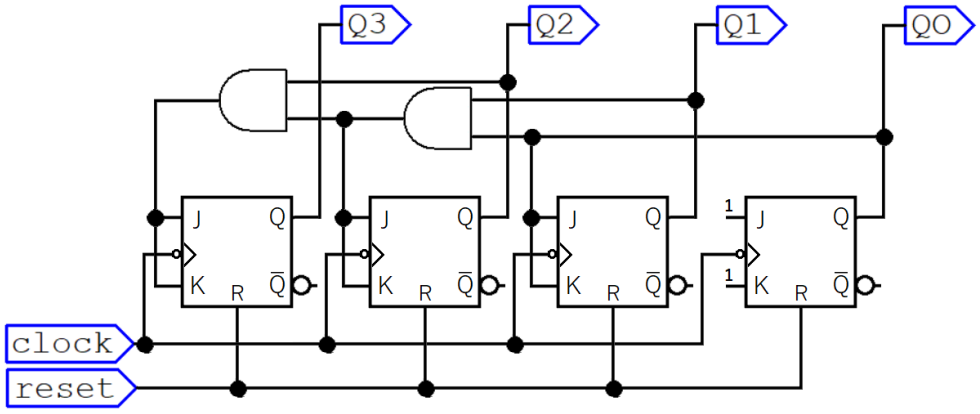


Figura 153 - Circuito de um contador síncrono.

A Figura 154 apresenta as formas de onda para o contador do exemplo. Mesmo que seja levado em consideração os atrasos de propagação dos circuitos, não aparecem estados indesejados na saída.

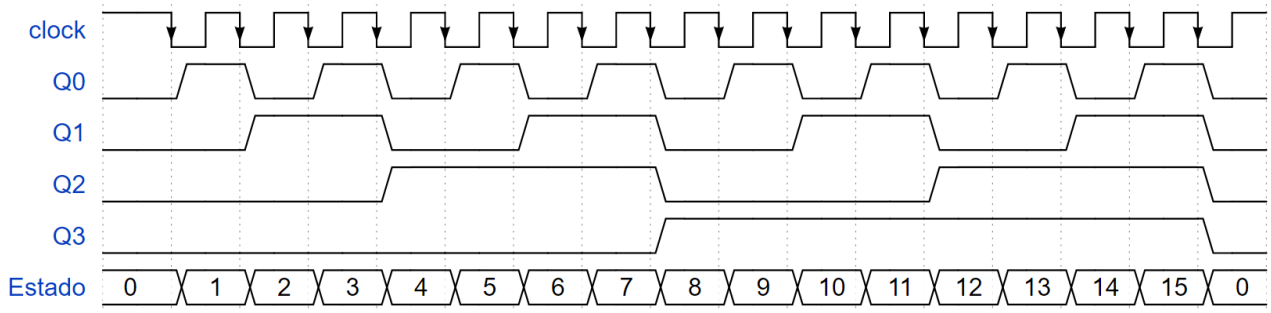


Figura 154 - Formas de onda para um contador síncrono de 4 bits.

A lógica de contagem é determinada pelas conexões estabelecidas nas entradas J e K dos flip-flops. No exemplo anterior, duas portas E são utilizadas para implementar uma lógica de contagem crescente. Diferentes circuitos podem definir diferentes lógicas, como uma contagem decrescente por exemplo.

Existem vários componentes comerciais que implementam contadores síncrono, podemos destacar dois deles, o 74168 e o 74169. Trata-se de contadores síncronos de 4 bits. O 74168 é um contador decimal contando de 0 a 9, enquanto o 74169 conta de 0 a 15. Ambos possuem a possibilidade de carregar um valor inicial de forma paralela, e permitem contar de forma crescente ou decrescente. A Figura 155 mostra a embalagem o a pinagem destes circuitos integrados.

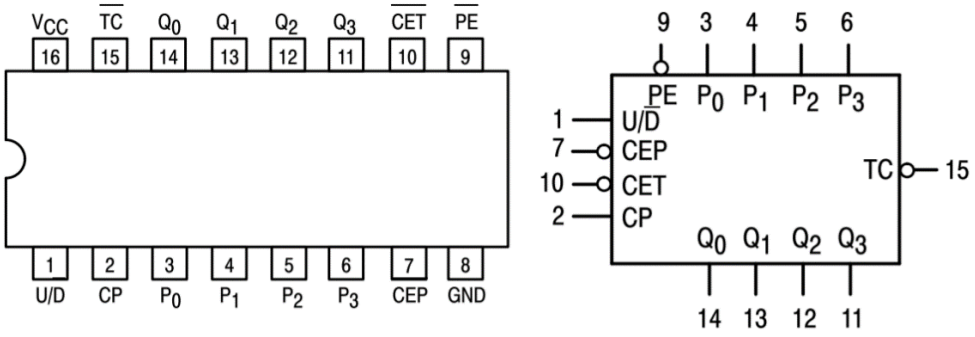


Figura 155 - Circuito integrado contador síncrono 74168 e 74169.

6.6. O circuito integrado LM555

Existem diversos circuitos integrados que implementam funções digitais sequenciais. Um componente importante neste contexto é o LM555. O LM555 é um circuito integrado projetado para gerar oscilações ou atrasos precisos, mas pode desempenhar diversas outras funções interessantes se configurado corretamente. No modo de operação de retardo de tempo (atraso), o tempo é controlado com precisão por um resistor e um capacitor externos. Para operação como oscilador, a frequência de funcionamento e razão cíclica são controlados com precisão por dois resistores externos e um capacitor. O dispositivo pode operar de 4,5 a 16 V, com sua saída podendo fornecer ou drenar até 200 mA.

A Figura 156 apresenta o diagrama funcional do LM555.

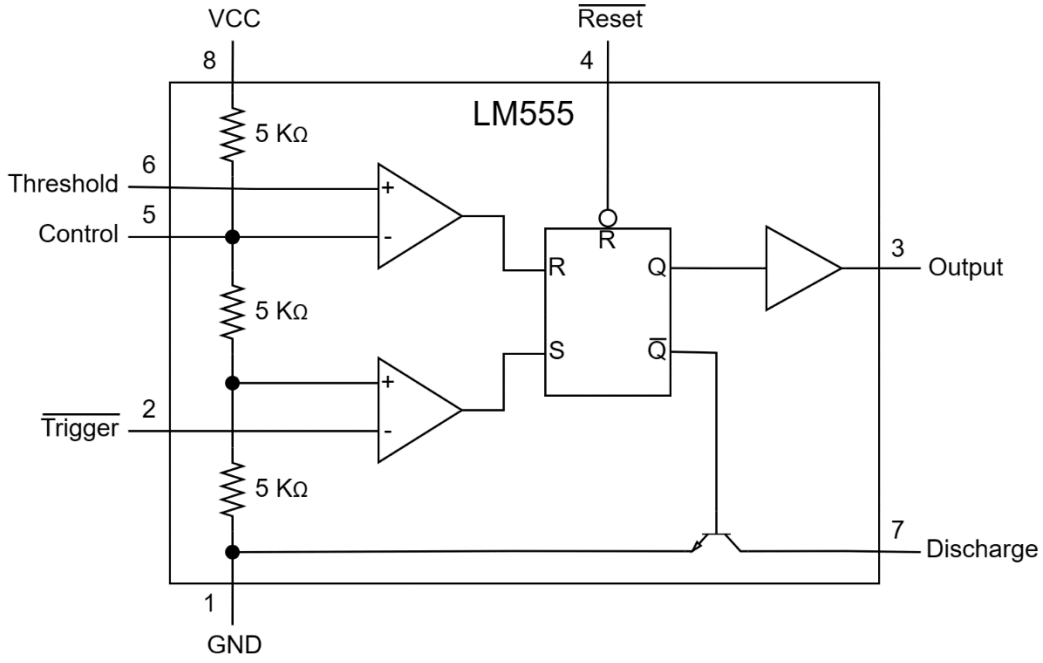


Figura 156 – Diagrama interno do LM555.

O LM555 é fornecido em diversas embalagens, com a mais comum apresentada Figura 157.

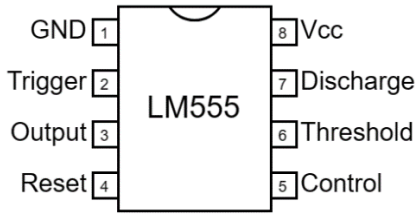


Figura 157 – O circuito integrado LM555.

6.6.1. Modos de operação do LM555

O LM555 pode operar de várias formas, mas as principais são como oscilador, chamado de astável e como retardo de tempo ou atraso, chamado de monoestável, dependendo dos requisitos da aplicação.

Modo astável

O temporizador LM555 pode operar como um oscilador e emite um fluxo contínuo de pulsos retangulares com uma frequência especificada. A frequência do fluxo de pulso depende dos valores de RA, RB e C.

A Figura 158 apresenta o circuito na configuração astável.

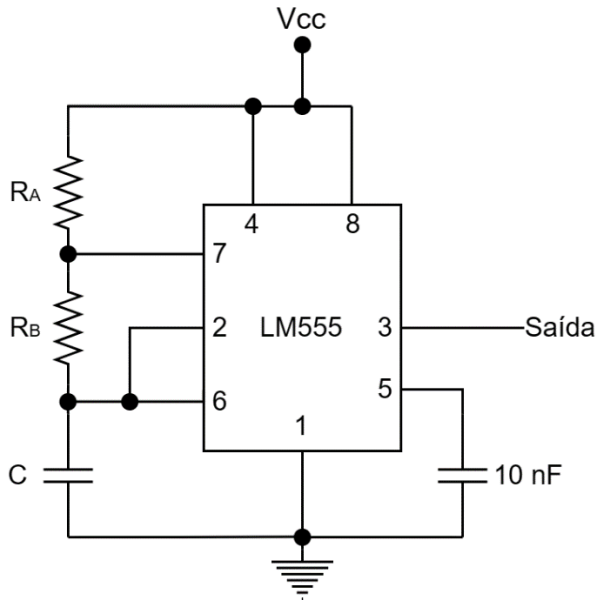


Figura 158 - LM555 na configuração astável.

As expressões a seguir permitem determinar o período, a frequência e a razão cíclica do sinal gerado.

Período do sinal: $T = 0{,}693(R_A + 2R_B)C$

Frequência do sinal: $f = \frac{1{,}44}{(R_A+2R_B)C}$

Razão cíclica (duty cucle): $D = \frac{R_B}{R_A+2R_B}$

A Tabela 46 apresenta alguns exemplos de valores para os componentes.

Tabela 46 - Exemplos de configuração do LM555.

R_A	R_B	C	Frequência	Razão cíclica
10 KΩ	68 KΩ	10 uF	1 Hz (0.988)	53.4%
8,2 KΩ	68 KΩ	1 uF	10 Hz (10,007)	52.8%
10 KΩ	68 KΩ	10 nF	1 KHz (997)	53.4%

Modo monoestável:

O temporizador LM555 pode atuar como um gerador de pulsos “one-shot”. O pulso ocorre quando o temporizador LM555 recebe um sinal na entrada do gatilho (pino 2) que cai abaixo de 1/3 da tensão de alimentação. A largura do pulso de saída é determinada pela constante de tempo de

uma rede RC. O pulso de saída termina quando a tensão no capacitor é igual a 2/3 da tensão de alimentação. A largura do pulso de saída pode ser estendida ou encurtada dependendo da aplicação, ajustando os valores R e C.

A Figura 159 apresenta o circuito na configuração monoestável.

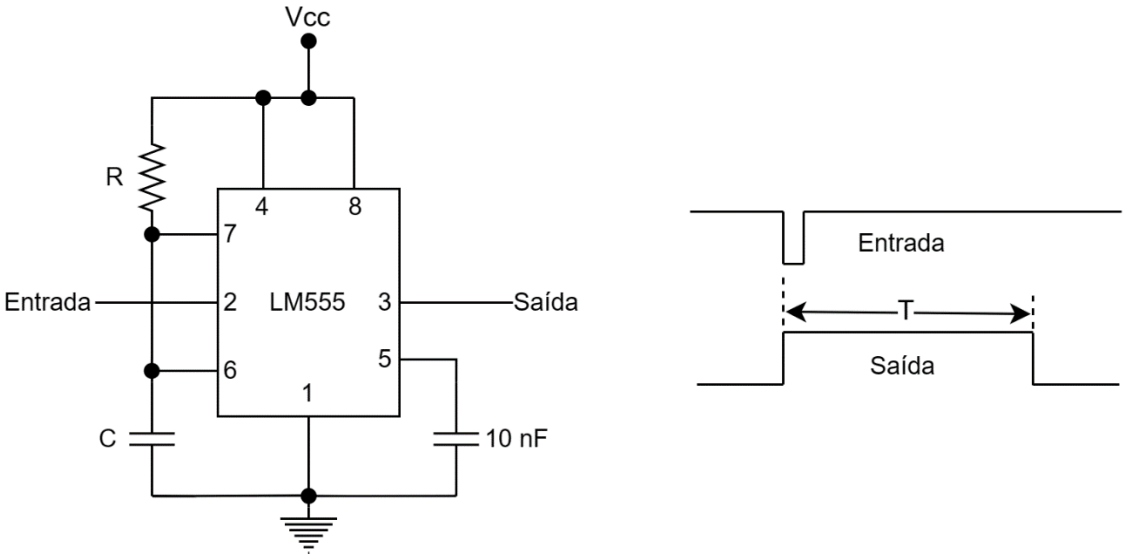


Figura 159 - LM555 na configuração monoestável.

A expressão a seguir permite determinar o período da saída em segundos: $T = 1{,}1 \cdot R \cdot C$.

Por exemplo, para gerar um pulso com período (T) de 1 segundo podemos utilizar um capacitor (C) de 10 uF e um resistor (R) de 91 KΩ.

6.7. Exercícios

1) Complete as linhas Q e $\bar{Q}$ nas formas de onda da Figura 160 para um latch RS.

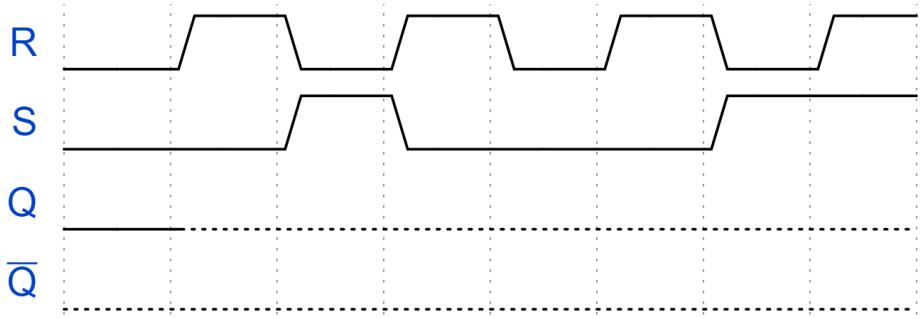


Figura 160 - Exercício latch RS.

2) Complete as linhas Q e $\bar{Q}$ nas formas de onda da Figura 161 para um latch RS síncrono.

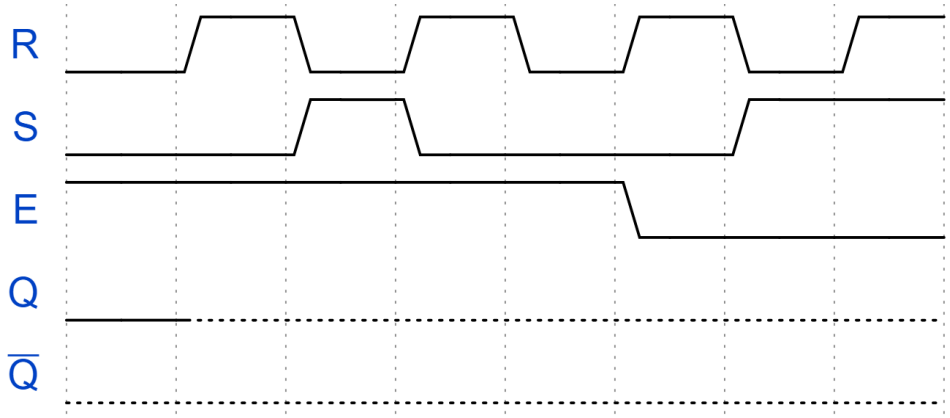


Figura 161 - Exercício latch RS síncrono.

3) Verifique no simulador o funcionamento do circuito integrado 74573 que implementa um latch D. Utilize chaves nas entradas e LEDs nas saídas, dando especial atenção a função do pino OE.
4) Para o circuito da Figura 162 determine a forma de onda da saída. Observe que o *clock* é ativo na borda de subida.

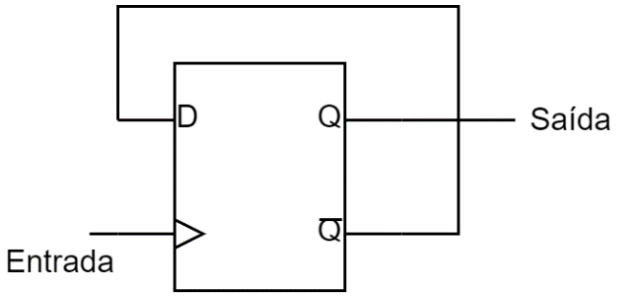
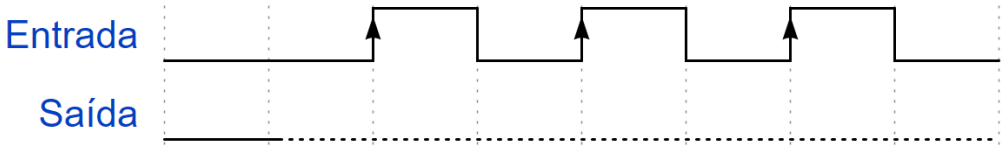


Figura 162 - Exercício flip-flop D.



5) Implemente o circuito da Figura 163 no simulador, verifique seu funcionamento e complete as formas de onda para as saídas a, b, c e d.

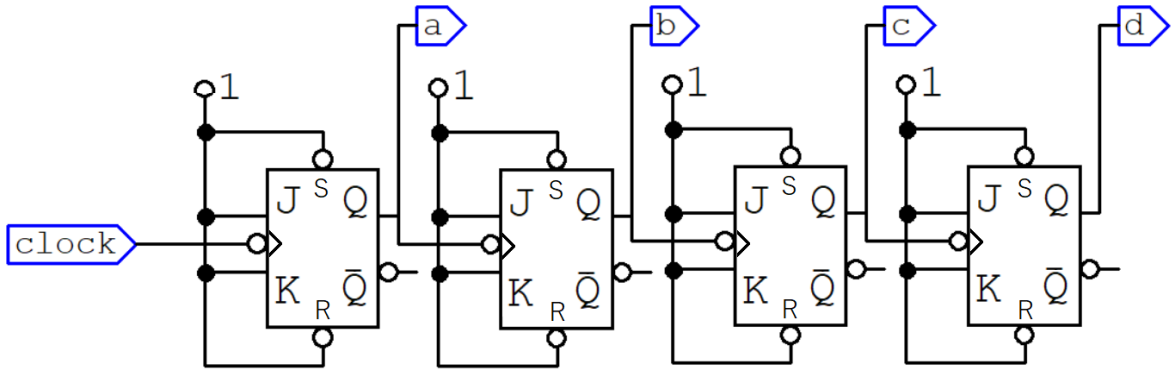
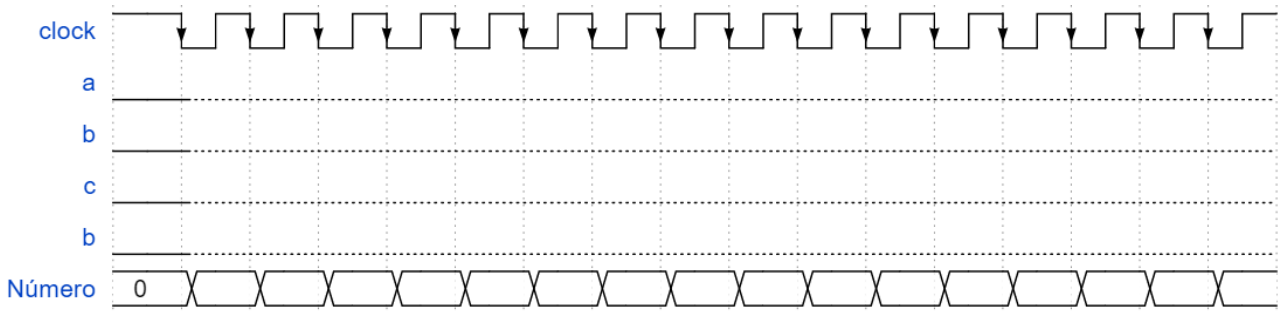


Figura 163 - Exercício sobre flip-flop JK.



6) Implemente no simulador, utilizando um circuito integrado 74164 e um circuito integrado 74165 o circuito do diagrama de blocos da Figura 164. O circuito deve transmitir os 8 bits da entrada para os 8 LEDs da saída através de um sinal de dados e um de *clock*. Implemente os circuitos auxiliares necessários e não apresentados no diagrama de blocos.

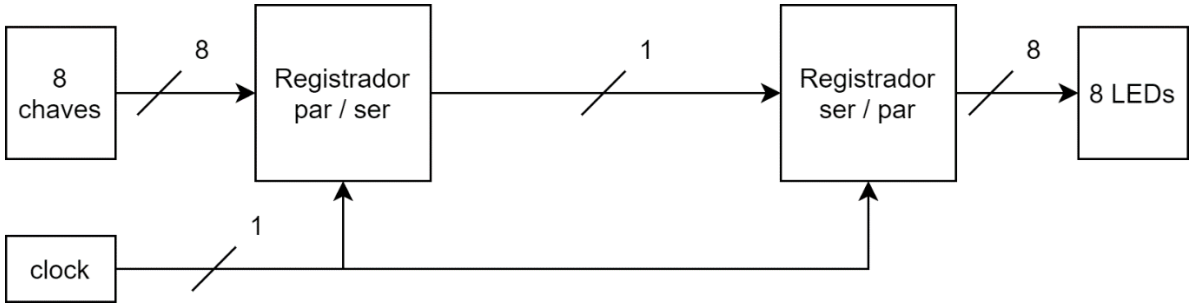


Figura 164 - Exercício sobre registradores.

7) Construa no simulador, utilizando contadores 7490, um contador de dois dígitos, que conta de 0 a 99. Os números devem aparecer em 2 *displays* de 7 segmentos.
8) Construa utilizando flip-flops um contador síncrono decrescente de 4 bits.
9) Utilizando circuitos integrados contadores síncronos monte no simulador um contador de quatro dígitos. O contador deve contar segundos e minutos, 2 dígitos para os segundos e dois dígitos para os minutos. Os números devem aparecer em displays de 7 segmentos. Lembre-se que segundos e minutos são contados de 0 a 59.

Aula 7 - Circuitos lógicos aritméticos

O objetivo desta aula é apresentar aos alunos os principais circuitos lógicos aritméticos, enfatizando sua construção e sua aplicação. Serão abordados circuitos somadores e subtratores, bem como operações em complemento de 2. Também faremos uma breve introdução às unidades lógicas e aritméticas.

7.1. Introdução

Circuitos aritméticos são aqueles que realizam operações aritméticas, como soma subtração multiplicação e divisão. Estes circuitos podem operar sobre sinais digitais em vários formatos, como números no sistema binários, números em complemento de 2, números em ponto flutuante etc.

Neste material trataremos apenas de soma e subtração de números no sistema decimal e em complemento de 2.

Serão abordadas também as unidades lógicas aritméticas, que são circuitos integrados capazes de desempenhar várias funções sobre um conjunto de bits de entradas, inclusive funções aritméticas.

7.2. Soma

A operação de soma entre valores binários segue a mesma lógica da operação de soma de números decimais. No sistema decimal os valores possíveis para um dígito vão de 0 a 9, assim se somarmos dois números e o resultado estiver dentro deste intervalo teremos um resultado também de um dígito. Porém, se somarmos dois valores e o resultado for maior que 9 teremos um resultado com dois dígitos. Sim temos o chamado transporte, o popular “vai um”, transportando um valor do primeiro dígito para o segundo. Por exemplo, 9 + 8 = 7 e transporta 1 para o próximo dígito.

Com valores binários o raciocínio é parecido, porém cada “dígito” é um bit e pode assumir os valores 0 ou 1. Assim, se o resultado da soma for maior que 1 existe um transporte para o próximo bit. A Tabela 47 apresenta a soma de dois valores de 1 bit (1 dígito binário).

Tabela 47 - Soma de um bit.

Num. 1		Num. 2		Resultado	Transporte
0	+	0	=	0	0
0	+	1	=	1	0
1	+	0	=	1	0
1	+	1	=	0	1

Este conceito pode ser utilizado para a soma de valores binários de vários bits. Neste caso, a partir do segundo bit tem-se a soma dos dois valores deste dígito mais o valor do transporte do dígito anterior.

Neste caso, temos uma nova tabela de valores para a soma, apresentada na Tabela 48.

Tabela 48 - Soma completa de um bit.

Transporte dígito anterior	Num. 1		Num. 2		Resultado	Transporte próximo dígito
0	0	+	0	=	0	0
0	0	+	1	=	1	0
0	1	+	0	=	1	0
0	1	+	1	=	0	1
1	0	+	0	=	1	0
1	0	+	1	=	0	1
1	1	+	0	=	0	1
1	1	+	1	=	1	1

A Figura 165 apresenta um exemplo onde é realizada a soma dos números 12_{10} e 6_{10}, ambos representados por 4 bits. Observe que neste caso o resultado possui 5 bits.

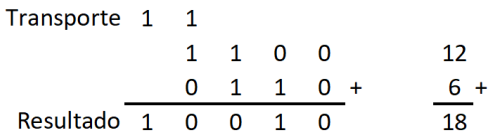


Figura 165 - Exemplo 1 de soma binária.

Outro exemplo é apresentado na Figura 166, onde é realizada a soma dos números 31_{10} e 10_{10}, ambos representados por 5 bits, o resultado agora possui 6 bits.

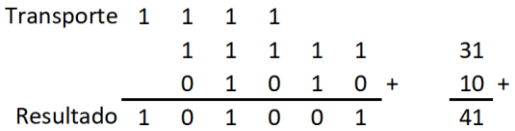


Figura 166 - Exemplo 2 de soma binária.

A implementação desta lógica nos sistemas digitais é realizada através de circuitos ditos somadores. Para o primeiro bit, que não recebe transporte, temos um circuito chamado somador incompleto. Para os outros bits, que necessitam de uma entrada de transporte temos um circuito chamado de somador completo. Estes dois circuitos serão apresentados a seguir.

7.2.1. Somador Incompleto

O circuito somador incompleto, em inglês *Half adder* possui duas entradas para os valores a ser somados A e B. Este circuito possui também duas saídas, o resultado da soma chamado S e o transporte, em inglês *carry*, chamado C. A Figura 167 apresenta a implementação deste circuito com portas lógicas.

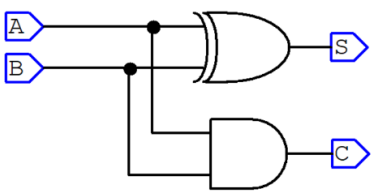


Figura 167 - Circuito somador incompleto.

A tabela verdade para este circuito é semelhante a tabela apresentada anteriormente, e pode ser visualizada na Tabela 49.

Tabela 49 - Tabela verdade para o somador incompleto.

Entradas		Saídas	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

7.2.2. Somador completo

O circuito somador completo, em inglês *Full adder* possui duas entradas para os valores a ser somados A e B, e uma entrada para o transporte do dígito anterior C_{in}. Este circuito possui também duas saídas, o resultado da soma chamado S e o transporte, em inglês *carry*, chamado C_{out}. A Figura 168 Figura 167apresenta a implementação deste circuito com portas lógicas.

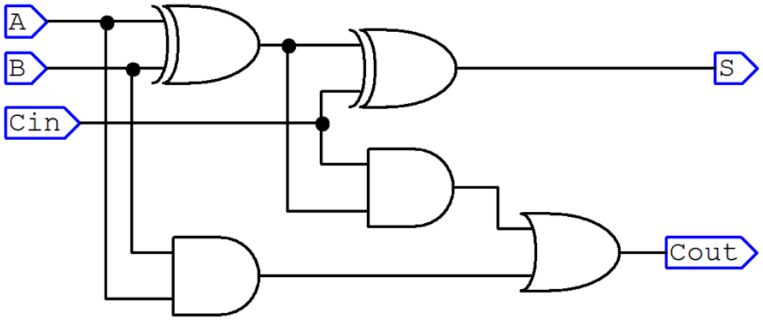


Figura 168 - Circuito para um somador completo.

A tabela verdade para este circuito é semelhante a tabela apresentada anteriormente, e pode ser visualizada na Tabela 50

Tabela 50 - Tabela verdade para o somador completo.

Entradas			Saídas	
C_{in}	A	B	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

É importante salientar que o somador incompleto é apropriado para o primeiro bit do conjunto a ser somado, enquanto o somador completo é apropriado para os demais bits.

7.2.3. Somador paralelo

Quando se deseja somar palavras com vários bits é necessário fazer uma composição chamada de somador paralelo. A Figura 169 mostra um somador paralelo de 4 bits.

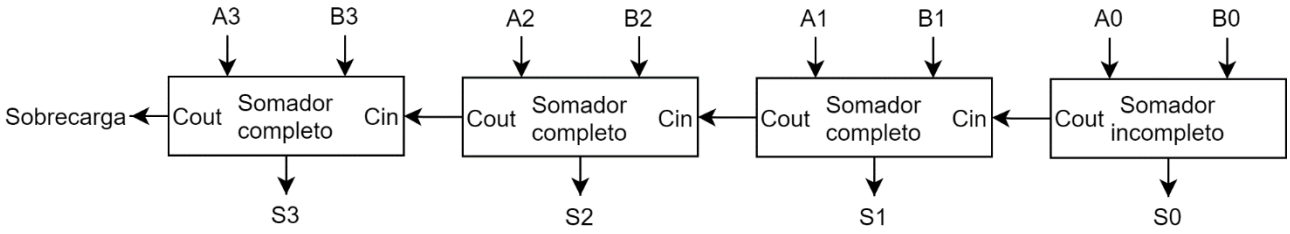


Figura 169 - Somador paralelo.

Neste exemplo o primeiro somador é um somador incompleto, pois não é necessária uma entrada de transporte. Os demais somadores são somadores completos.

A última saída de transporte é utilizada para indicar uma sobrecarga, ou seja, o resultado da soma não pode ser armazenado em 4 bits.

Um circuito integrado muito utilizado na função de somador paralelo de quatro bits é o 7483. A Figura 170 mostra este circuito.

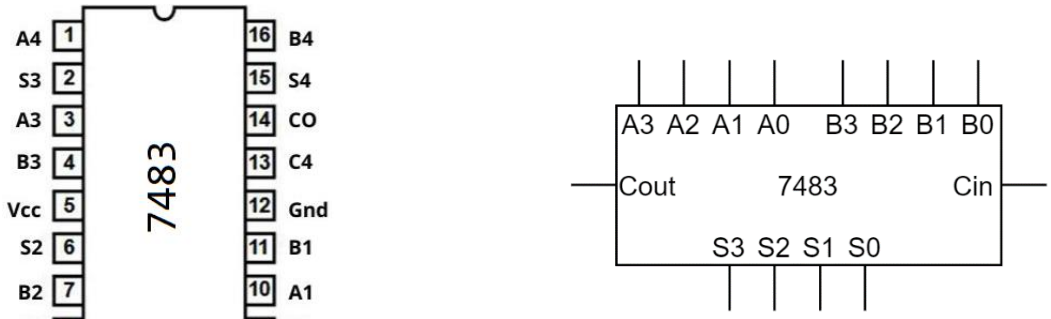


Figura 170 - Circuito integrado somador paralelo 7483.

Este circuito integrado possui entradas e saídas de transporte auxiliares, assim é fácil interligar vários deles para formar somadores com mais bits.

7.3. Subtração

A subtração binária segue uma lógica parecida com a subtração decimal. A Tabela 51 apresenta uma tabela com os possíveis resultados para uma operação de subtração entre dois bits.

Tabela 51 - Subtração binária.

Num. 1		Num. 2		Resultado	Empresta
0	-	0	=	0	0
0	-	1	=	1	1
1	-	0	=	1	0
1	-	1	=	0	0

Observe na tabela que existe um sinal chamado empresta, em inglês chamado *borrow*, cuja função é parecida com o empresta na subtração decimal. Quando trabalhamos com subtração de vários bits, o empresta subtrai um do bit a esquerda.

Veja o exemplo da Figura 171.

Empresta	-1	-1				
	1	1	0	0		12
	0	1	1	0	-	6 -
Resultado	0	1	1	0		6

Figura 171 - Exemplo de subtração binária.

7.3.1. Representação de números negativos e a subtração

Já vimos que existem algumas formas de representa números negativos na forma binária, sinal e magnitude, complemento de 1 e complemento de 2, por exemplo. A representação de números negativos está diretamente relacionada com a operação de subtração, pois se desejarmos por exemplo subtrair dois números positivos basta inverter o sinal do segundo e então realizar a operação de soma. Este raciocínio também é válido para números binários com representação em complemento de 2. Vejamos um exemplo.

Na Figura 172 temos o exemplo da operação 5 – 10 = -5, onde a subtração é substituída pela soma e o número 10 é substituído por -10 em complemento de 2 (10110_2). Observe que o resultado é -5, também representado em complemento de 2.

Transporte		1					
	0	0	1	0	1		5
	1	0	1	1	0	+	(-10)+
Resultado	1	1	0	1	1		(-5)

Figura 172 - Subtração através da soma em complemento de 2.

7.3.2. Subtrator paralelo

Uma vez que se pode utilizar a representação em complemento de 2 e a operação de soma para realizar a subtração, os circuitos apresentados anteriormente podem ser utilizados. Assim para realizar a subtração basta fazer o complemento de 2 do valor a ser subtraído e realizar a soma normalmente. É importante lembrar que o resultado também vai estar em complemento de 2.

Um exemplo de circuito que realiza a soma ou a subtração, conforme determinado por um sinal de entrada é apresentado na Figura 173.

Neste circuito são utilizadas portas OU Exclusivo para fazer o complemento de 2 do operando B. Se lembrarmos que o complemento de 2 de um número é seu inverso acrescido de 1, então quando o sinal M for 1, as portas OU Exclusivo fazem a inversão de todos os bits do operando B. O sinal C_{in} do primeiro circuito 7483 também recebe 1 do sinal M, assim é somado 1 ao operando B, realizando sua conversão para complemento de 2.

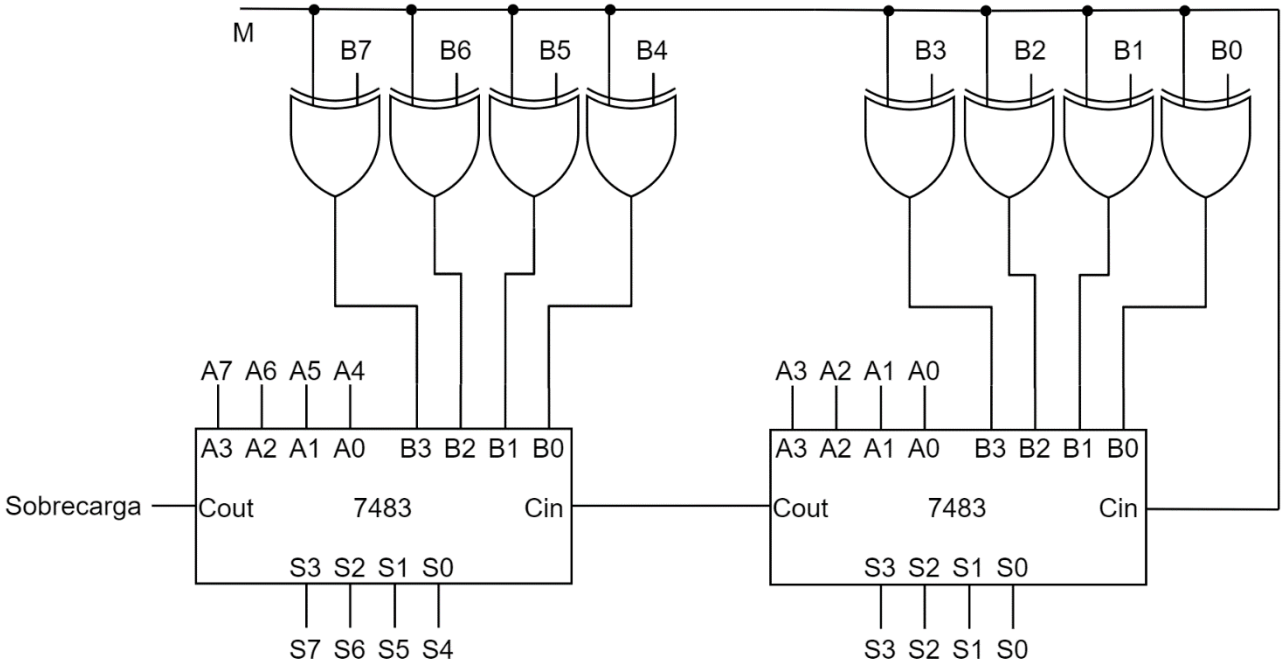


Figura 173 - Somador / Subtrator paralelo de 8 bits.

Desta forma, este circuito se comporta como um somador se M = 0 e como um Subtrator se M = 1.

7.4. Unidade Lógica e Aritmética

Para facilitar a implementação de circuitos que executam operações lógicas ou aritméticas entre sinais de vários bits foram criados circuitos integrados chamados “Unidade Lógica e Aritmética - ULA” ou em inglês “Arithmetic Logic Unit - ALU”. Estes circuitos integrados permitem executar várias operações diferentes sobre um conjunto de bits, através da um conjunto de sinais de controle.

Um exemplo de ULA é o circuito integrado presentado na Figura 174, onde podemos observar sua embalagem com 24 pinos seu símbolo, utilizado nos diagramas eletrônicos digitais.

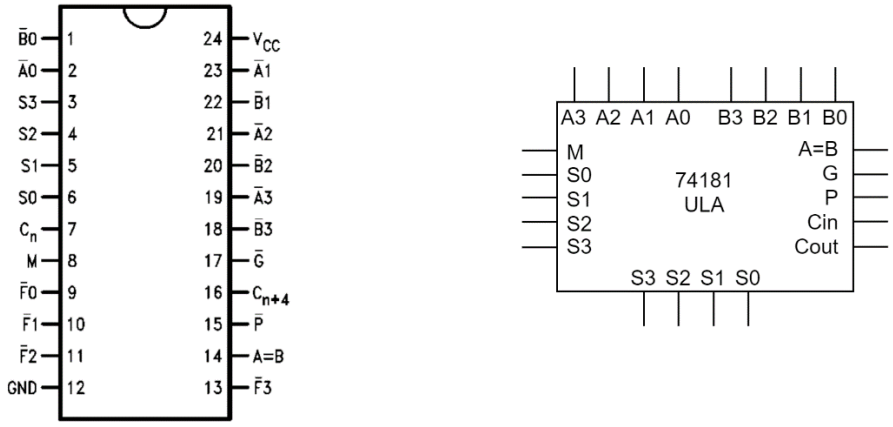


Figura 174 - Circuito integrado ULA 74181.

As funções dos pinos são as seguintes:

- $\overline{A_0} - \overline{A_3}$ Operando A (lógica invertida).
- $\overline{B_0} - \overline{B_3}$ Operando B (lógica invertida).
- $S_0 - S_3$ Entradas para seleção d operação desejada.
- M Controle do modo de operação.
- C_n ou C_{in} Entrada de *carry* (empresta ou transporte)
- $\overline{F_0} - \overline{F_3}$ Saídas (lógica invertida).
- $A = B$ Saída do comparador.
- $\bar{G}$ Usado para associação com o integrado 74182 para aumentar a velocidade.
- $\bar{P}$ Usado para associação com o integrado 74182 para aumentar a velocidade.
- C_{n+4} ou C_{out} Saída de *carry* (empresta ou transporte)

Este circuito integrado executa as operações apresentadas na Figura 175.

Mode Select Inputs				Active LOW Operands & F_n Outputs		Active HIGH Operands & F_n Outputs	
				Logic	Arithmetic (Note 2)	Logic	Arithmetic (Note 2)
S3	S2	S1	S0	(M = H)	(M = L) (C_n = L)	(M = H)	(M = L) (C_n = H)
L	L	L	L	$\overline{A}$	A minus 1	$\overline{A}$	A
L	L	L	H	$\overline{AB}$	AB minus 1	$\overline{A} + \overline{B}$	A + B
L	L	H	L	$\overline{A} + \overline{B}$	$A\overline{B}$ minus 1	$\overline{A}$ B	$A + \overline{B}$
L	L	H	H	Logic 1	minus 1	Logic 0	minus 1
L	H	L	L	$\overline{A} + \overline{B}$	A plus (A + $\overline{B}$)	$\overline{AB}$	A plus $A\overline{B}$
L	H	L	H	$\overline{B}$	AB plus (A + $\overline{B}$)	$\overline{B}$	(A + B) plus $A\overline{B}$
L	H	H	L	$\overline{A} \oplus \overline{B}$	A minus B minus 1	A ⊕ B	A minus B minus 1
L	H	H	H	$A + \overline{B}$	$A + \overline{B}$	$A\overline{B}$	AB minus 1
H	L	L	L	$\overline{A}$ B	A plus (A + B)	$\overline{A} + B$	A plus AB
H	L	L	H	A ⊕ B	A plus B	$\overline{A} \oplus \overline{B}$	A plus B
H	L	H	L	B	$A\overline{B}$ plus (A + B)	B	(A + $\overline{B}$) plus AB
H	L	H	H	A + B	A + B	AB	AB minus 1
H	H	L	L	Logic 0	A plus A (Note 1)	Logic 1	A plus A (Note 1)
H	H	L	H	$A\overline{B}$	AB plus A	$A + \overline{B}$	(A + B) plus A
H	H	H	L	AB	$A\overline{B}$ minus A	A + B	(A + $\overline{B}$) plus A
H	H	H	H	A	A	A	A minus 1

Note 1: Each bit is shifted to the next most significant position.
Note 2: Arithmetic operations expressed in 2s complement notation.

Figura 175 - Funções executadas pela ULA 74181.

Este circuito integrado permite que seja feito uma associação de várias unidades de forma a compor uma ULA com mais bits de entrada e saída. Mais informações estão disponíveis no *datasheet* do componente.

7.5. Exercícios

1) Construa no simulador um circuito somador para palavras de 3 bits utilizando portas lógicas.
2) Construa no simulador o circuito apresentado na Figura 173 e verifique seu funcionamento adicionando displays de 7 segmentos e chaves nas entradas e displays de 7 segmentos na saída. Teste o circuito com números positivos e negativos em complemento de 2.
3) No simulador, construa um circuito utilizando o 74181 e conectando chaves e displays de 7 segmentos a suas entradas e LEDs e displays de 7 segmentos em suas saídas. Teste algumas de suas funções e verifique seu funcionamento.

Aula 8 - Famílias lógicas e circuitos MSI

O objetivo desta aula é apresentar aos alunos detalhes sobre o funcionamento e as tecnologias empregadas na fabricação de circuitos integrados digitais. Também serão apresentadas as tecnologias mais modernas e sua evolução em relação aos sistemas digitais.

8.1. Introdução

Sistemas digitais modernos são majoritariamente compostos por circuitos integrados. Muitos destes circuitos já foram apresentados nas aulas anteriores, porém existem diversas tecnologias para a fabricação de circuitos integrados, algumas já ultrapassadas e outras sendo empregadas a pouco tempo. As seções a seguir apresentam as principais tecnologias envolvidas na fabricação de circuitos integrados, iniciando por uma discussão dos principais termos técnicos relacionados com o assunto.

Para isso, o assunto será dividido em duas partes, uma falando sobre as famílias lógicas, ou seja, as famílias de circuitos integrados utilizados em sistemas digitais e outra falando sobre as escalas de integração e sua classificação.

8.2. Famílias lógicas

Para que possamos entender as diferenças entre as famílias de circuitos digitais é necessário que compreendamos as características elétricas envolvidas nos sinais digitais e consequentemente nas entradas e saídas das portas lógicas e dos circuitos integrados digitais. Neste sentido iniciaremos discutindo os parâmetros de tensão e corrente envolvidos.

8.2.1. Parâmetros de tensão e corrente

Nos sistemas digitais os sinais digitais 1s e 0s são representados por sinais elétricos. Estes sinais possuem níveis de tensão e corrente pré-definidos e que devem ser respeitados por todos os componentes do sistema.

Quando uma porta lógica envia nível lógico 1 em sua saída ela deve manter sua tensão de saída acima de um mínimo, de forma que os demais componentes possam receber este nível 1. Da mesma forma, quando uma porta lógica envia nível lógico 0 em sua saída ela deve manter sua tensão de saída abaixo de um máximo, de forma que os demais componentes possam receber este nível 0.

De forma semelhante, uma porta deve ser capaz de fornecer um determinado valor de corrente em sua saída quando em nível 1 e drenar um determinado valor de corrente quando em nível 0.

Estes níveis de tensão e corrente de entrada e saída dos circuitos digitais são característicos de cada família lógica e são definidos pelos fabricantes segundo uma nomenclatura padrão.

A Figura 176 (a) apresenta uma porta lógica transmitindo nível lógico 1 para outra porta lógica, enquanto a Figura 176 (b) apresenta uma porta lógica transmitindo nível lógico 0 para outra porta lógica.

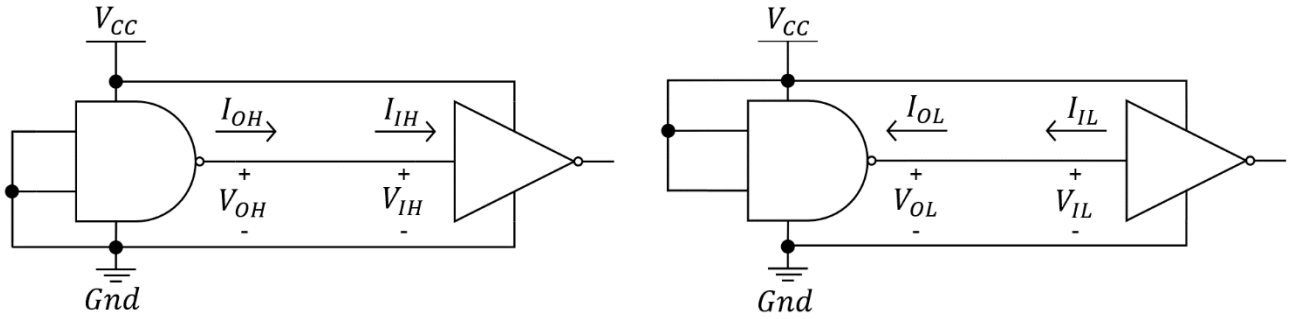


Figura 176 - Sinais de tensão e corrente.

Observando a figura nota-se que as tenções e correntes de entrada e saída de cada uma das portas possui um nome específico. A seguir estão relacionados estes nomes.

- V_{IH} (*High-Level Imput Voltage*) Tensão mínima de entrada para nível lógico alto.
- V_{IL} (*Low-Level Imput Voltage*) Tensão máxima de entrada para nível lógico baixo.
- V_{OH} (*High-Level Output Voltage*) Tensão mínima de saída para nível lógico alto.
- V_{OL} (*Low-Level Output Voltage*) Tensão máxima de saída para nível lógico baixo.
- I_{IH} (*High-Level Imput Current*) Corrente que entra pela entrada de uma porta para nível lógico alto.
- I_{IL} (*Low-Level Imput Current*) Corrente que sai pela entrada de uma porta para nível lógico baixo.
- I_{OH} (*High-Level Output Current*) Corrente que sai na saída de uma porta para nível lógico alto.
- I_{OL} (*Low-Level Output Current*) Corrente que entra pela saída de uma porta para nível lógico baixo.
- $V_{CC}\ ou\ V_{DD}$Tensão de alimentação nominal positiva.
- $Gnd\ ou\ V_{SS}$Tensão de alimentação nominal negativa.

Margem de ruído

Um parâmetro dos componentes digitais é a chamada margem de ruído, veja a Figura 177.

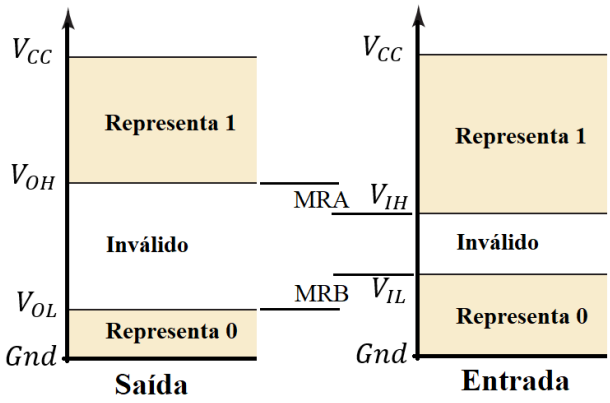


Figura 177 - Margem de ruído.

A margem de ruído é a diferença de tensão entre a saída e a entrada de uma determinada família de componentes digitais. Esta margem permite que pequenas variações elétricas nos sinais não interfiram no funcionamento dos circuitos.

Para o nível 1 temos a Margem de Ruido Alta (MRA) que é a diferença $V_{OH} - V_{IH}$ e para o nível lógico 0 temos a Margem de Ruido Baixa (MRB) que é a diferença $V_{IL} - V_{OL}$.

8.2.2. Atraso de propagação

Também conhecido como *delay* o atraso de propagação é o atraso que todo sinal sofre ao atravessar um circuito digital. Costuma-se utilizar dois tipos de atraso de propagação.

- t_{PLH} Tempo para passar de baixo para alto 0 → 1.
- t_{PHL} Tempo para passar de alto para baixo 1 → 0.

A Figura 178 apresenta o atraso de propagação para uma porta inversora.

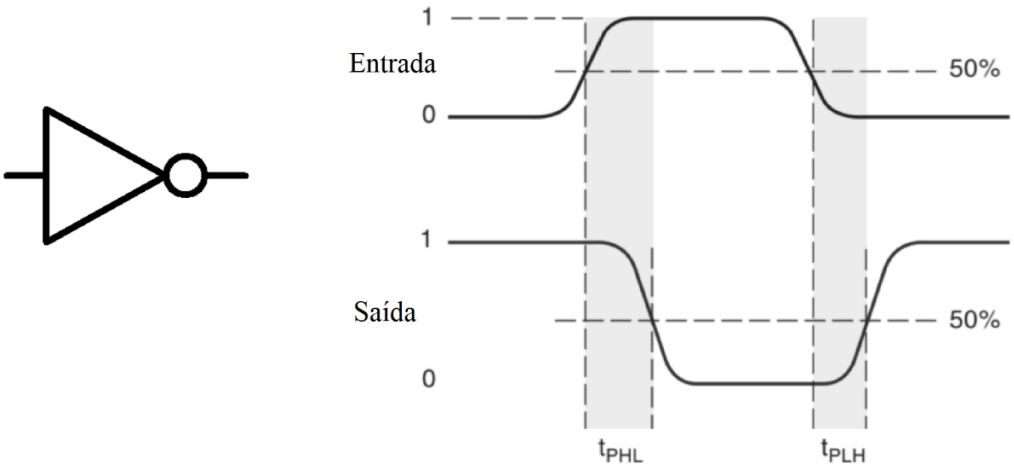


Figura 178 - Atraso de propagação.

É importante salientar que em muitos casos o atraso na subida e na descida do sinal não são necessariamente iguais. Este atraso é importante pois ajuda a definir a máxima frequência de operação de um circuito.

8.2.3. Fan-Out

Fan-Out é um termo utilizado para definir o fator de carga de uma saída digital. Uma saída de um circuito digital não é capaz de fornecer alimentar um número infinito de entradas digitais, existe um limite para a corrente que uma saída consegue fornecer ou drenar. Assim, o Fan-Out é o número máximo de entradas que uma saída pode alimentar. Por exemplo, quando falamos que o Fan-Out de uma porta lógica é 20, significa que ela pode comandar até 20 entradas digitais.

O Fan-Out está relacionado com as capacidades de corrente explicadas anteriormente. Assim, o Fan-Out para o nível baixo pode ser calculado pela seguinte expressão.

$$Fan\text{–}Out_{nível\ baixo} = \frac{I_{OL}}{I_{IL}}$$

Da mesma forma para o nível alto temos.

$$Fan\text{–}Out_{nível\ alto} = \frac{I_{OH}}{I_{IH}}$$

8.3. Famílias lógicas baseadas em transistor bipolar

Um conjunto de circuitos digitais que compartilha a mesma estrutura básica e a mesma tecnologia de fabricação, bem como as mesmas especificações elétricas é chamada de uma família lógica.

Os primeiros circuitos integrados digitais e consequentemente as primeiras famílias lógicas foram construídas com transistores bipolares de junção (BJTs). Com esta tecnologia foram criadas três famílias lógicas, a DTL, a TTL e a ECL.

8.3.1. Famílias lógica Diodo-Transistor - DTL

O nome DTL vem do inglês *Diode-Transistor Logic,* esta tecnologia é baseada em transistores bipolares e diodos. Esta tecnologia surgiu na década de 50 e não é muto apropriada para a construção de circuitos integrados pois exige grandes resistores e consome grandes quantidades de energia. A Figura 179 mostra o circuito de uma porta Não E construída com esta tecnologia.

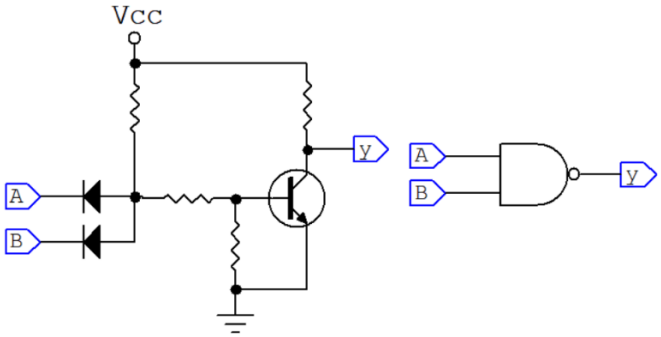


Figura 179 - Porta lógica da família DTL.

8.3.2. Família Transistor-Transistor - TTL

A família TTL (*Transistor-Transistor Logic*) surgiu na década de 60 e obteve enorme sucesso pois permitiu a fabricação em grande escala de circuitos integrados lógicos.

Como exemplo a Figura 180 apresenta uma porta Não E construída com esta tecnologia.

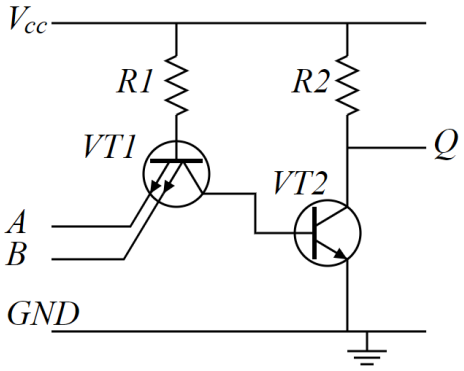


Figura 180 - Porta lógica da família TTL.

A família TTL é de grande importância pois deu origem a série 74 de circuitos integrados, já utilizados, mas aulas anteriores. Nesta família os circuitos integrados têm sua nomenclatura padronizada, iniciando pelo número 74 e complementada por um número que corresponde a função lógica implementada. Existe ainda uma variação que inicia com o número 54 e possui uma margem de temperatura maior de operação, destinada a aplicações militares.

A família TTL original, apesar de funcional possuía baixa velocidade de operação e alto consumo de energia. Com o propósito de melhorar estas características e com o avanço da tecnologia, com o passar do tempo surgiram evoluções desta tecnologia.

Desta forma a família TTL passou a ser composta por várias categorias de circuitos integrados, relacionadas a seguir. Para identificar cada categoria, foram adicionadas letras entre o número 74 (ou 54) e o número que identifica a função lógica.

- 74: TTL comum.
- 74S: TTL Schottky.
- 74AS: TTL Schottky avançado.
- 74LS: TTL Schottky de baixa potência.
- 74ALS: TTL Schottky de baixa potência avançado.
- 74F: TTL rápido.

A utilização de diodos Schottky junto aos transistores permitiu um ganho de velocidade e uma redução do consumo de potência.

Cada categoria de circuito integrado da família TTL possui características únicas de velocidade e consumo de energia, bem como de Fan-Out. Assim, para mais informações sobre um modelo específico o manual do componente deve ser consultado.

Apesar das características das diferentes tecnologias, a família TTL apresenta tensão de alimentação, entrada e saída padronizadas, como mostrado a seguir.

- V_{IH} Tensão mínima de entrada para nível lógico alto = **2,0V**.
- V_{IL} Tensão máxima de entrada para nível lógico baixo = **0,8V**.
- V_{OH} Tensão mínima de saída para nível lógico alto = **2,4V**.
- V_{OL} Tensão máxima de saída para nível lógico baixo = **0,4V**.
- $V_{CC}\ ou\ V_{DD}$Tensão de alimentação nominal positiva = **5,0V**.

Observando estas informações é possível determinar que a margem de ruído para a família TTL é de **0,4V**.

8.3.3. Lógica acoplada pelo emissor ECL

A família ECL (*Emiter-Coupled Logic*) é uma família desenvolvida especialmente para altas velocidades. Apesar de operar em altíssimas velocidades esta tecnologia é empregada apenas em aplicações muito específica, pois apresenta grande consumo de energia, e devido a suas características construtivas, custo elevado.

Outra característica que dificulta a utilização desta tecnologia são as tenções empregadas em sua alimentação e em seus sinais lógicos. A alimentação, por exemplo, é V_{ee} = -5,2V e o circuito

necessita de uma tensão de referência de -1,3V. Os níveis lógicos também empregam tensões incomuns, -1,7V para nível lógico 0 e -0,9V para nível lógico 1. A Figura 181 apresenta um exemplo desta tecnologia.

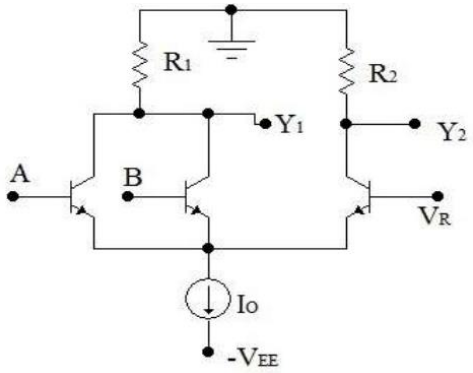


Figura 181 - Exemplo de circuito da família ECL.

As famílias lógicas baseadas e BJTs exigiam uma grande área de silício e possuíam um grande consumo de energia, assim, de forma a melhorar estas características, foi criada uma tecnologia com base em transistores MOSFET, apresentada a seguir.

8.4. Famílias lógicas baseadas em transistor MOSFET

Os transistores MOSFET são transistores mais modernos que ocupam menos espaço nos circuitos integrados e consomem menos energia que os transistores bipolares. Além disso a tecnologia MOSFET continua evoluindo, com o tamanho dos transistores diminuindo a cada ano.

Outra vantagem desta tecnologia é que as entradas de sinal necessitam de uma corrente muito baixa, aumentando muito a capacidade de Fan-out.

Existem diversas tecnologias que empregam transistores MOSFET em sua construção, a seguir apresentaremos algumas delas.

8.4.1. Lógica CMOS

A primeira série de componentes lógicos a empregar a tecnologia MOSFET, mais especificamente a tecnologia CMOS possui uma nomenclatura que começa com 40. Esta série de componentes apresenta vários circuitos integrados que implementam diversas funções lógicas. OS códigos numéricos utilizados não são compatíveis com os códigos utilizados na família TTL. A Figura 182 apresenta como exemplo uma porta inversora CMOS.

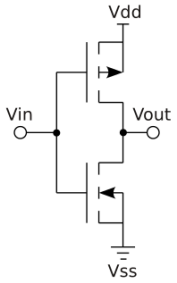


Figura 182 - Exemplo de circuito CMOS.

Uma característica interessante dos circuitos integrados CMOS é a tensão de operação, os circuitos podem ser alimentados de 3V a 15V, não exigindo uma tensão fixa de 5V.

Como exemplo desta tecnologia podemos citar o latch RS CD4043, já mencionado anteriormente e apresentado novamente na Figura 183.

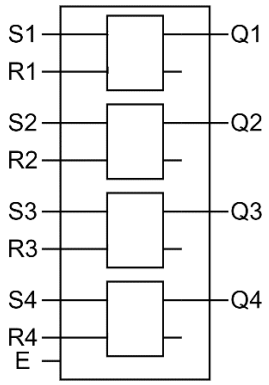


Figura 183 - Exemplo de circuito integrado da família CMOS 40XX.

Circuitos CMOS 74

Para aumentar a compatibilidade entre circuitos integrados CMOS e circuitos integrados TTL foram criadas categorias de CMOS com compatibilidade de função e pinagem com os circuitos da família 74.

A série 74CXX possui desempenho similar a série 40XX mas é compatível em função e pinagem com a série TTL 74XX.

As séries 74HCXX e HCTXX foram desenvolvidas para operar em 5V (embora possa funcionar de 2V a 6V) e são um aperfeiçoamento da série 74CXX. A linha HC não apresenta compatibilidade com a família TTL, mas a linha HCT foi projetada para ser compatível com a família TTL, permitindo misturar componentes dos dois tipos. Se comparados com componentes TTL, estes componentes possuem quase a mesma velocidade, mas com um consumo de energia significativamente mais baixo.

As séries 74ACXX e 74ACTXX foram desenvolvidas para apresentar maior velocidade e maior corrente de saída. A linha AC pode operar de 3V a 5,5V enquanto a linha ACT pode operar de 4,5V a 5,5V

Existem ainda outras séries mais modernas e mais rápidas, cada uma com características únicas, principalmente com relação ao nível de tensão de alimentação e de sinal.

CMOS de baixa tensão

Para melhorar a performance, principalmente com relação ao consumo de energia, a velocidade e a dissipação de calor, os fabricantes vêm desenvolvendo novas tecnologias com tenções de operação cada vez mais baixa.

Inicialmente a tensão de operação dos sistemas digitais foi reduzida de 5V para 3,3V, com muitos sistemas atuais operando nesta tensão. Para sistemas mais modernos, principalmente computadores, telefones celulares e outros sistemas mais complexos a tensão de operação é ainda

menor. Tensões padronizadas para este tipo de sistema digital são: 2,5V, 1,8V, 1,5V, 1,2V e 1V. A seguir são apresentados os nomes de alguns destes padrões.

- LVTTL 3,3V
- LVCMOS 3,3V
- LVCMOS 2,5V
- LVCMOS 1,8V
- LVCMOS 1,5V
- LVCMOS 1,2V
- LVCMOS 1,0V

Cada um destes padrões possui características específicas de tensão de entrada e saída, margem de ruído e Fan-Out, que devem ser respeitadas nos projetos. Existem ainda outros padrões modernos, e a cada dia surgem novas tecnologias que melhoram as características dos sistemas digitais, mas não é objetivo deste material se aprofundar neste assunto.

8.5. Circuitos lógicos MSI

Quando se fala em circuitos integrados estamos tratando de circuitos compostos por vários componentes e construídos em uma única embalagem. Os circuitos integrados estudados nas aulas anteriores pertencem a uma classe de circuitos integrados conhecida pela sigla MSI (*Medium Scale Integration*) ou seja, circuitos integrados construídos com um número de portas lógicas entre 13 e 99 portas por componente.

Foram estudados codificadores e decodificadores, multiplexadores e demultiplexadores, comparadores, contadores, somadores entre outros componentes digitais, todos pertencentes a classe MSI.

A classificação completa dos circuitos integrados digitais é a seguinte.

- SSI (*Small Scale Integration*) até 12 portas lógicas.
- MSI (*Medium Scale Integration*) de 13 a 99 portas lógicas.
- LSI (*Large Scale Integration*) de 100 a 9.999 portas lógicas.
- VLSI (*Very Large Scale Integration*) de 10.000 a 99.999 portas lógicas.
- ULSI (*Ultra Large Scale Integration*) de 100.000 ou mais portas lógicas.

As escalas maiores de integração normalmente são utilizadas em circuitos integrados com funções complexas específicas, memórias ou microcontroladores e microprocessadores.

Uma categoria interessante são os dispositivos lógicos programáveis ou reconfiguráveis, que possuem em seu interior um número elevado de unidades lógicas que podem ser conectadas conforme o projeto do usuário. Estes componentes serão estudados em uma aula específica.

8.6. Embalagens dos circuitos integrados

Como os circuitos integrados podem possuir vários componentes em seu interior é comum que necessitem de embalagens com vários terminais. Os circuitos MSI estudados anteriormente

possuem até algumas dezenas de terminais, mas outros componentes podem possuir até milhares de terminais. Assim os fabricantes disponibilizam uma enorme variedade de embalagens.

As embalagens são classificadas em dois tipos, com relação a forma que elas são soldadas nas placas de circuito impresso. Os componentes podem ser do tipo PTH (*Pin Through Hole*), onde os terminais são inseridos em furos na placa para ser soldados, como na Figura 184.

Figura 184 - Componentes PTH.

Quando os terminais dos componentes são projetados para ser soldados na superfície da placa, sem necessitar de um furo, estes componentes são do tipo SMD (*Surface Monted Device*), como na Figura 185.

Figura 185 - Componentes SMD.

Os componentes apresentados anteriormente neste material possuem em sua maioria uma embalagem do tipo PTH, chamada DIP (*Dual Inline Package*), apresentada na Figura 186. Nesta embalagem os pinos são contados no sentido anti-horários a partir da marcação.

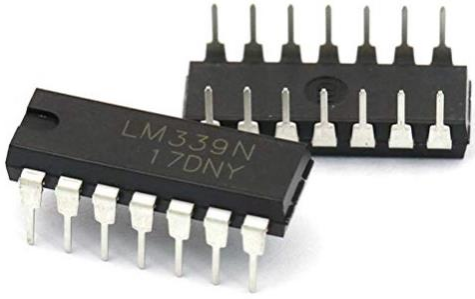


Figura 186 - Embalagem DIP.

Porém com hoje em dia praticamente todas as placas de circuito são montadas de forma automatizada, existe uma preferência por componentes SMD, isso porque este tipo de componente é mais facilmente posicionado na placa por máquinas automatizadas. Desta forma, a maioria dos componentes digitais possui embalagens do tipo SMD, principalmente para componentes com muitos terminais.

A Figura 187 apresenta algumas das embalagens SMD mais utilizadas em componentes digitais.

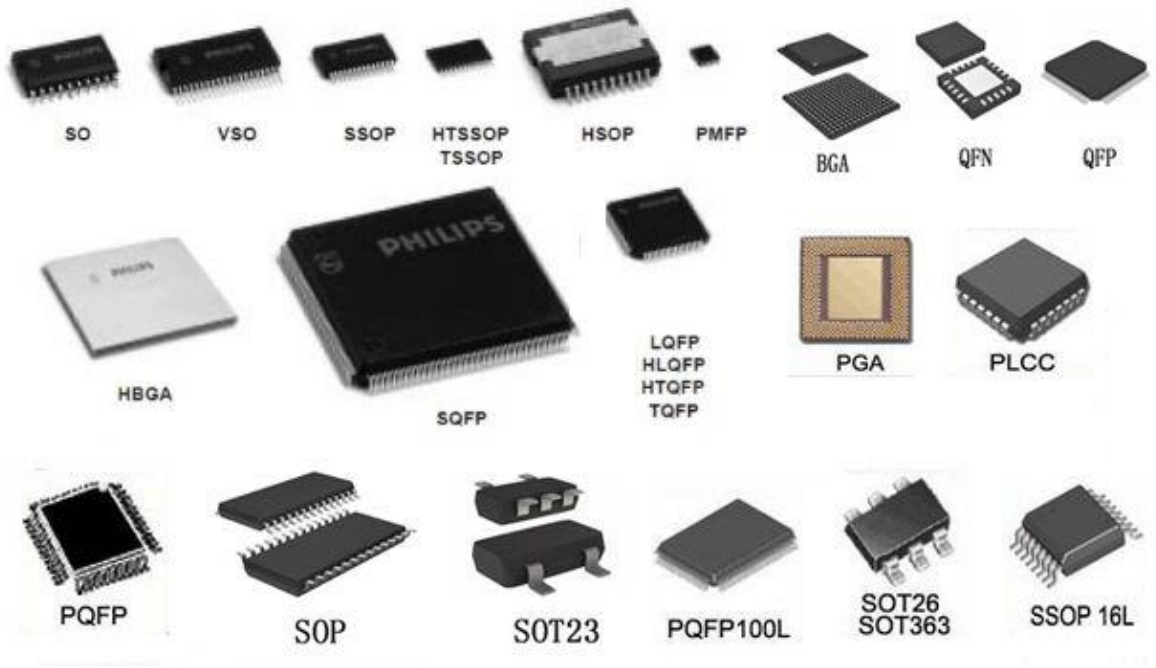


Figura 187 - Embalagens SMD para componentes digitais.

8.7. Exercícios

1) Assista os seguintes vídeos para conhecer o processo de fabricação de circuitos integrados: https://www.youtube.com/watch?v=bor0qLifjz4 https://www.youtube.com/watch?v=Fxv3JoS1uY8
2) Faça uma pesquisa na internet para conhecer mais sobre as embalagens utilizadas nos circuitos integrados.
3) Assista o seguinte vídeo para conhecer o processo de soldagem dos circuitos integrados nas placas de circuito impresso: https://www.youtube.com/watch?v=24ehoo6RX8w

Aula 9 - Memórias

O objetivo desta aula é apresentar aos alunos os dispositivos de armazenamento de informação digital, voláteis e não voláteis, bem como explicar seu funcionamento e sua utilização nos sistemas digitais.

9.1. Introdução

Nas aulas anteriores foras estudados dispositivos capazes de armazenar pequenas quantidade de informação digital, como latches, flip-flops e registradores. Porém, para muitas aplicações o armazenamento de grandes quantidades de informação é necessário e estas tecnologias não atendem a esta demanda. Para este tipo de aplicações existem dispositivos eletrônicos destinados ao armazenamento de grandes quantidades de informação, as memórias. Existem diversos tipos de memórias, divididas em duas categorias, memórias voláteis, que perdem as informações quando a energia é retirada e memórias não voláteis que preservam as informações quando a energia é removida.

9.1.1. Algumas terminologias

Para facilitar o entendimento das memórias é importe conhecer alguns termos utilizados.

- **Entradas de endereço** (*address*) são as entradas dos circuitos integrados que recebem sinais digitais para identificar uma certa posição da memória.
- **Entradas de dados** (*data input*) são as entradas dos circuitos integrados que recebem sinais digitais contendo as informações a serem armazenadas em uma certa posição da memória. As entradas de dados podem compartilhar os pinos com as saídas de dados.
- **Saída de dados** (*data output*) são as saídas dos circuitos integrados que disponibilizam os sinais digitais contendo as informações armazenadas em uma certa posição da memória. As saídas de dados podem compartilhar os pinos com as entradas de dados.
- **Escrever** (*write*) operação de armazenar as informações da entrada na posição de memória selecionada.
- **Ler** (*read*) operação de disponibilizar nas saídas as informações armazenadas na posição de memória selecionada.
- **Dados** (*data*) conjunto de informações.

Outros termos que pertencem a algum tipo específico de memória serão definidos no momento de sua utilização.

9.1.2. O que são memórias

As memórias são dispositivos digitais com a capacidade de armazenar M palavras de N bits cada. É comum encontrarmos memórias de 4, 8, 16 ou mais bits por palavra. Cada palavra possui um endereço único de armazenamento na memória e pode ser acessada (lida ou gravada) de forma paralela. O número M de palavras é definido pelo número de endereços (L) disponíveis. Para selecionar um endereço, seu valor binário é fornecido através das entradas de endereço. A relação entre o número de endereços disponível e o número de entradas de endereço é a seguinte:

$$Número\ de\ endereços\ (M) = 2^{número\ de\ entradas\ de\ endereço\ (L)}$$

Assim para uma memória com 10 entradas de endereço teremos $2^{10} = 1024$ endereços, numerados de 0 a 1023. Ou seja, esta memória pode armazenar 1024 palavras de dados, cada uma com N bits.

A capacidade da memória em bits é dada pelo produto do número de palavras pelo número de bits em cada palavra que a memória é capaz de armazenar.

9.1.3. Um exemplo de memória

Como exemplo de memória a Figura 188 apresenta uma memória RAM (*Random Access Memory*), ou seja, uma memória de acesso aleatório genérica, com 10 entradas de endereço (A0-A9) e 8 entradas/saídas de dados (IO0-IO9). Esta memória também possui duas entradas de controle o CS (*chip Select*) que em nível 1 desabilita a operação da memória e em nível 0 habilita sua operação, o WE (*Write Enable*) que quando em nível lógico 0 habilita a gravação de dados na memória e quanto em nível lógico 1 habilita a leitura. É importante destacar que quando CS estiver em 1 as saídas de dados assumem terceiro estado (ficam desconectadas).

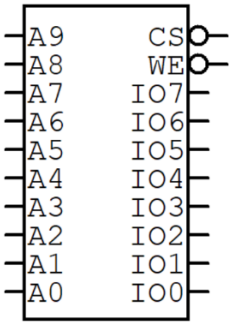


Figura 188 - Exemplo de memória.

Esta memória pode armazenar $1024 \cdot 8 = 8192$ bits de dados, em 1024 palavras de 8 bits cada.

9.2. Memórias voláteis

Memórias voláteis são aquelas que perdem suas informações assim que a energia do circuito é retirada. São normalmente rápidas e baratas, por isso muito utilizadas nos sistemas digitais.

Existem diversos tipos de memória volátil disponíveis no mercado, os principais serão apresentados a seguir.

9.2.1. SRAM

As memórias SRAM (*Static Random Access Memory*) são memórias estáticas de acesso aleatório. São memórias construídas com transistores MOSFET e armazenam as informações nos estados elétricos destes transistores. São rápidas e não necessitam de refrescamento (*refresh*), pois as informações não são perdidas, ao menos não enquanto a energia estiver presente. A desvantagem deste tipo de memória é o espaço necessário no semicondutor e consequentemente seu custo. A Figura 189 mostra a arquitetura interna de uma memória SRAM de 8192 palavras de 8 bits.

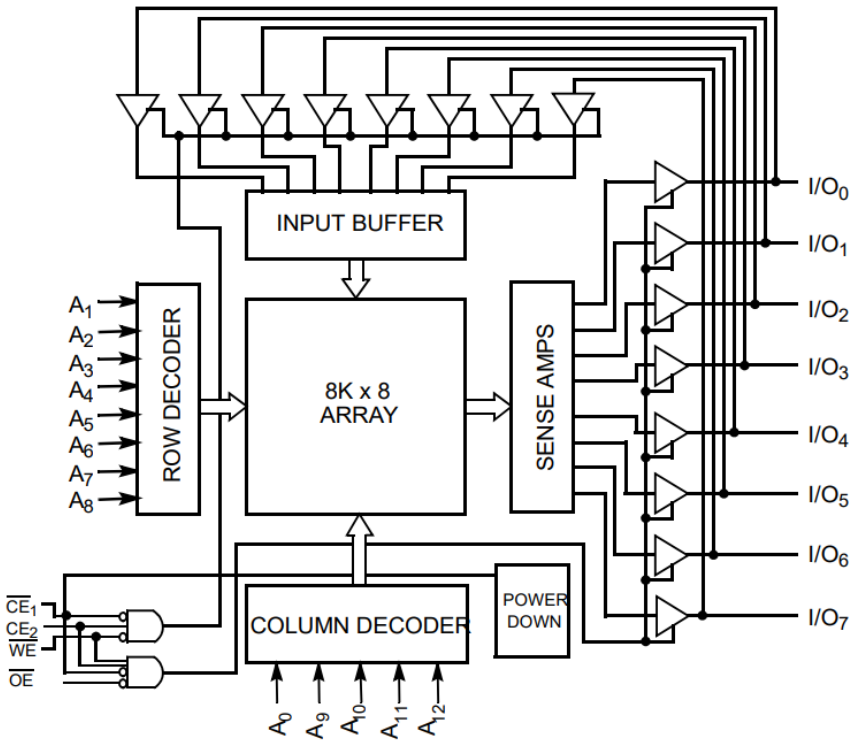


Figura 189 - Exemplo de SRAM.

Existe uma grande diversidade de memórias SRAM comerciais, com diferentes capacidades, velocidades e tecnologias de fabricação. A título de exemplo a Figura 190 apresenta a memória CY6264, uma memória SRAM pequena de 8K bytes.

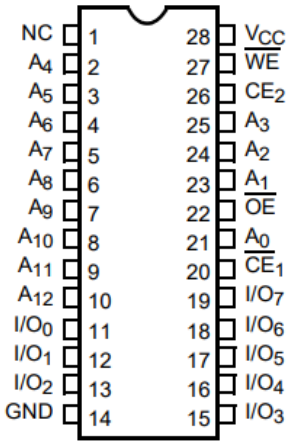


Figura 190 - Memória SRAM CY6264.

9.2.2. DRAM

As memórias DRAM (*Dynamic Random Access Memory*) são memórias de acesso aleatório dinâmicas, que assim como as SRAMs perdem seus dados quando não recebem mais energia. Porém, nas DRAMs o armazenamento das informações é feito em pequenos capacitores, o que torna a memória mais simples e mais barata. A desvantagem é que estes pequenos capacitores perdem sua carga muito rapidamente, exigindo ciclos de recarga ou refrescamento (*refresh*) dos dados. A Figura 191 apresenta o diagrama interno de uma memória DRAM de 256K palavras de 16 bits cada.

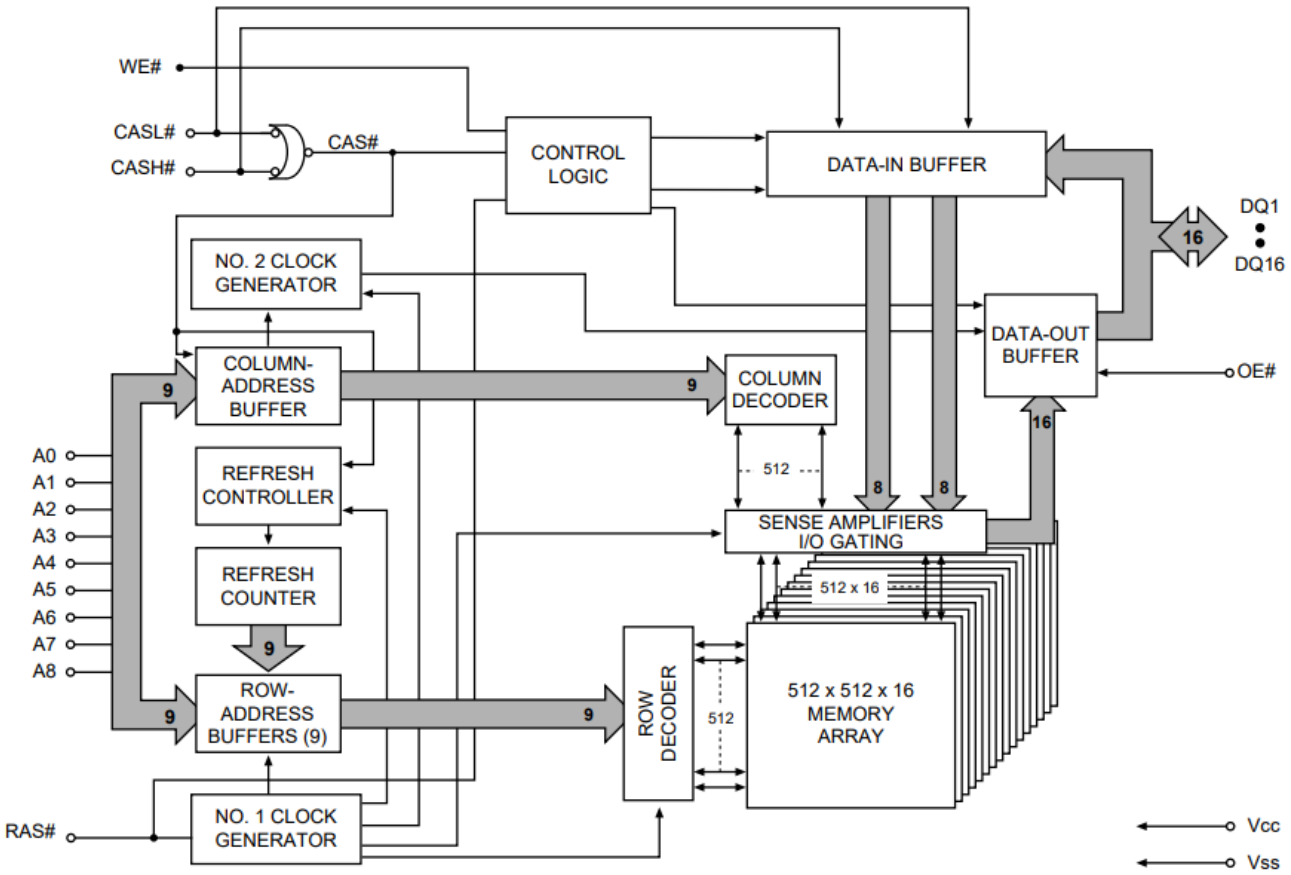


Figura 191 - Diagrama de uma memória DRAM.

Como as memórias DRAM consumam armazenar grandes quantidades de dados e devido ao processo de *refresh* são utilizados alguns sinais adicionais no sistema de endereçamento. Estes sinais são: **RAS** (*Row Address Strobe*), um sinal de clock que controla as entradas de endereçamento das linhas e **CAS** (*Column Address Strobe*), um sinal de clock que controla as entradas de endereçamento das colunas.

A principal diferença entre a SRAM e a DRAM é o processo de *refresh* necessário nesta última. Cada modelo possui sua própria lógica de *refresh,* normalmente envolvendo uma sequência de pulsos nos pinos de controle da memória. O intervalo de *refresh* é normalmente de alguns milésimos de segundo.

Normalmente este processo é executado por um sistema controlador de memória separado, o que torna o sistema digital mais caro e complexo.

Como exemplo de memória DRAM podemos citar o circuito integrado MT4C16270. Trata-se de uma memória DRAM de 256K palavras, cada uma com 16 bits, totalizando 4.194.303 bits.

A Figura 192 apresenta a embalagem desta memória.

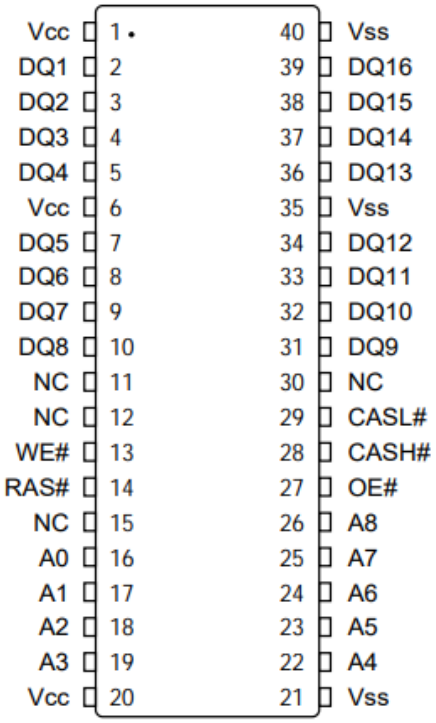


Figura 192 - Memória DRAM MT4C16270.

9.2.3. SDRAM

As memórias SDRAM (*Synchronous DRAM*) são memórias DRAM com diversas melhorias incorporadas. Os dados continuam sendo armazenados em capacitores, porém seu funcionamento é diferente.

A principal vantagem da memória SDRAM sobre a DRAM é que o sistema de controle de *refresh* está embutido na própria memória, o que diminui o custo e a complexidade do sistema digital como um todo. Esta melhoria também possibilita que a memória SDRAM opere a velocidades mais altas que a DRAM.

Esta memória é chamada de síncrona porque trabalha com um sinal de clock que estabelece um sincronismo de funcionamento entre a memória e o restante do sistema.

Por sua grande capacidade e grande velocidade de operação, entre outras vantagens, as memórias SDRAM são amplamente utilizadas nos sistemas digitais mais modernos, como computadores e telefones celulares.

Existem diversas tecnologias de memórias SDRAM em uso atualmente, podemos sitar por exemplo as memórias DDR1, DDR2, DDR3 e DDR4. A sigla DDR significa *Double Data Rate* ou seja taxa de dados dupla em tradução livre. Estas memórias conseguem transferir dados na borda de subida e na borda de descida do sinal de clock, o que lhe confere uma velocidade de escrita e leitura muito elevada.

A título de exemplo a Figura 193 apresenta o diagrama interno de uma memória SDRAM de 16M palavras com 4 bits cada.

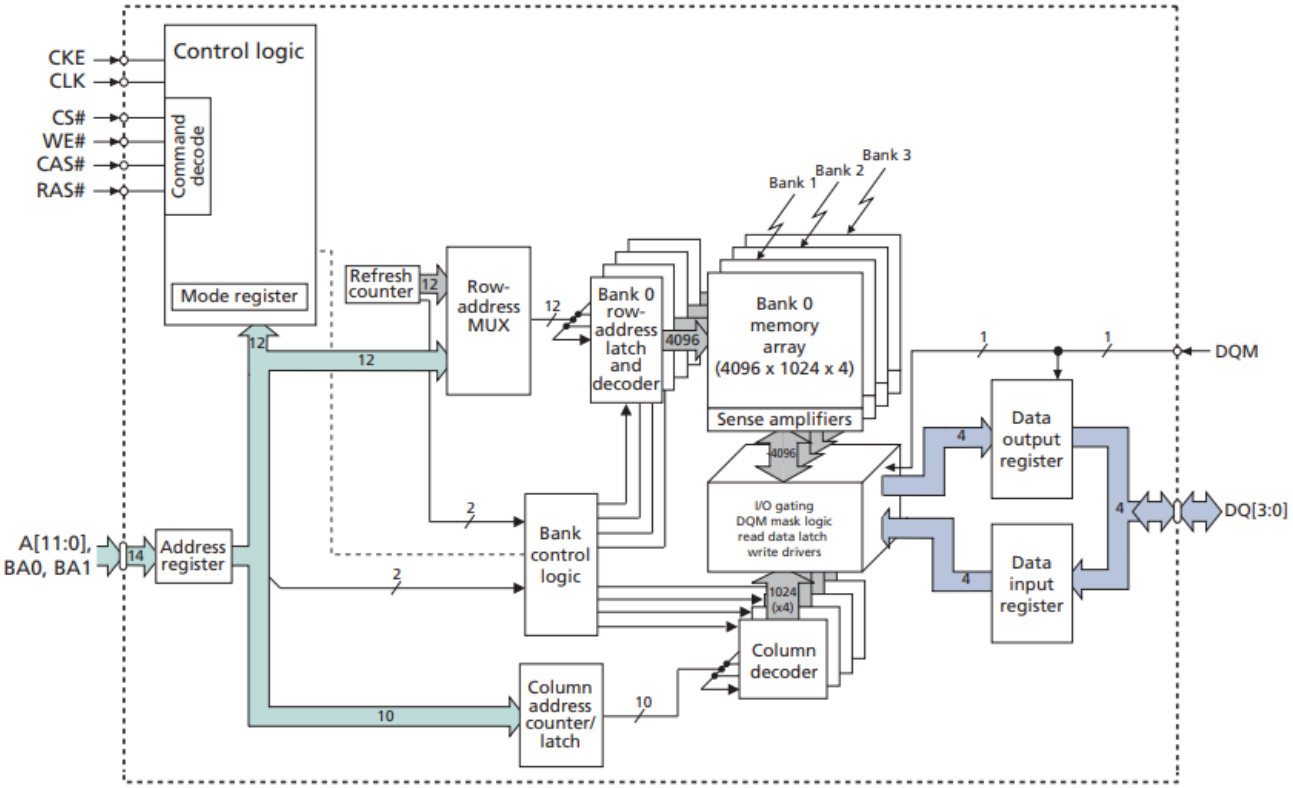


Figura 193 - Diagrama de blocos de uma memória SDRAM.

Como exemplo de memória SDRAM podemos sitar o circuito integrado, que é uma memória SDRAM de 16M palavras com 4 bits cada, cuja pinagem é apresentada na Figura 194.

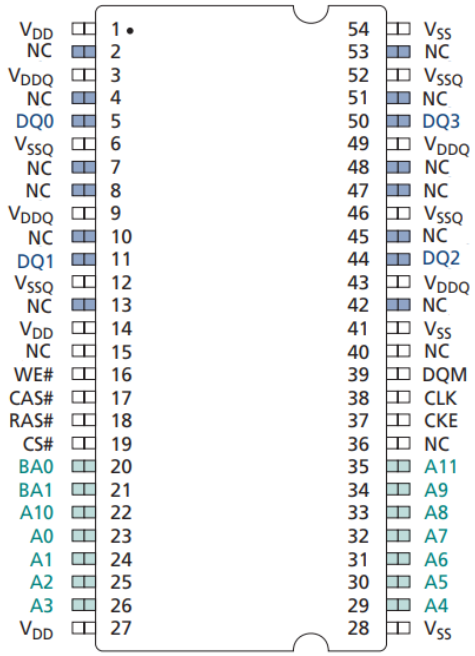


Figura 194 - Memória SDRAM MT48LC16M4A2.

Normalmente este tipo de memória é comercializado na forma de módulos, como na Figura 195

Figura 195 - Exemplo de módulo de memória SDRAM.

9.3. Memórias não voláteis

As memórias estudas até aqui são todas do tipo volátil, ou seja, apesar de rápidas e baratas perdem as informações nelas armazenadas quando a energia do sistema é retirada. Assim, sempre que necessitarmos manter as informações quando o sistema não estiver energizado é necessário utilizar outra categoria de memórias, as memórias não voláteis.

Existem diversas técnicas para armazenar informações de forma não volátil, como discos e fitas magnéticas, discos ópticos, memórias de estado sólido etc. Este material tratará apenas dos dispositivos eletrônicos de memórias (de estado sólido), não abordando outros meios físicos de armazenamento de dados.

Existem diversas técnicas de fabricação de circuitos integrados de memória não volátil, de forma genérica estas memórias são chamadas de ROM (*Read-Only Memory*) ou memória de somente leitura. Este termo é um tanto confuso, pois a maiorias das memórias não volátil pode ser gravada de alguma forma, mas o processo de gravação normalmente é complexo e demorado.

A seguir serão apresentadas as principais tecnologias de fabricação de memórias não voláteis.

9.3.1. MP-ROM

As memórias MP-ROM (*Mask-Programmed Read-Only Memory)* são um tipo de memória somente leitura (ROM) cujo conteúdo é programado pelo fabricante do circuito integrado, não podendo ser alterado pelo usuário. O termo *Mask-Programmed* vem do processo de fabricação de circuitos integrados, onde áreas específicas do chip são mascaradas durante o processo de fotolitografia, determinado se aquela área representa nível lógico 1 ou 0.

A principal vantagem deste tipo de memória é o custo. Além disso, a área ocupada no semicondutor é muito pequena. Como o custo de um circuito integrado depende fortemente de seu tamanho, este tipo de memória é significativamente mais barato do que qualquer outro tipo de memória semicondutora.

Contudo, esta vantagem só é relevante se o número de componentes produzidos com o mesmo conteúdo gravado (mesmos dados) for muito grande, pois o custo do desenvolvimento da máscara é elevado, não justificando pequenas produções.

9.3.2. OTP-ROM

Outro tipo de memória não volátil de somente leitura é a OTP-ROM (*One-Time-Programmable Read-Only Memory*). Este tipo de memória pode ser programado uma única vez, normalmente com um equipamento específico, e então não perde mais as informações e nem pode ter suas informações alteradas.

O princípio de funcionamento destes dispositivos é o seguinte. Normalmente cada bit de memória é constituído de um micro fusível, estes fusíveis podem ser destruídos (queimados) pela passagem de uma corrente elétrica elevada. Assim, o gravador queima seletivamente alguns destes fusíveis deixando outros intactos, gravando assim os 1s e 0s dos dados. Como não é mais possível restaurar um fusível queimado, esta memória não pode ser apagada.

A Figura 196 apresenta o diagrama de blocos de uma memória deste tipo.

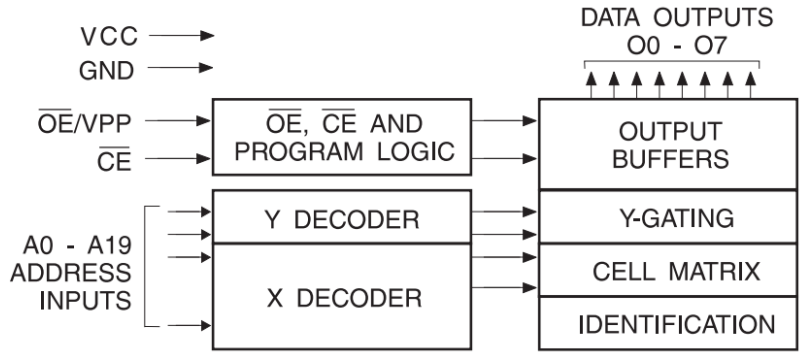


Figura 196 - Diagrama de blocos de uma memória OTP-ROM.

Um exemplo de memória OTP-ROM é a memória AT27C080, apresentada na Figura 197.

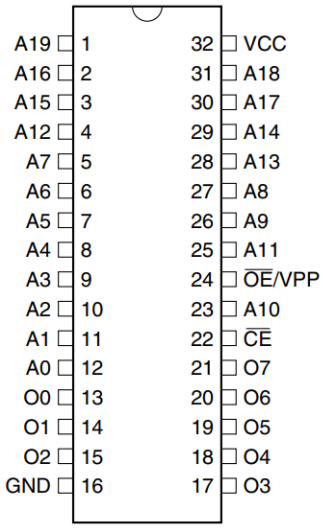


Figura 197 - Memória OTP-ROM AT27C080.

Esta memória armazena 1M palavras de 8bits por palavra.

9.3.3. EPROM

As memórias EPROM (*Erasable Programmable Read-Only Memory ou Electrically Programmable Read-Only Memory* para alguns autores) é uma memória somente leitura que pode ser gravada eletricamente e pode também ser apagada, podendo ser gravada novamente.

Neste tipo de memória, a informação é armazenada em transistores de porta flutuante. Estes transistores podem ser programados individualmente por um dispositivo eletrônico que fornece tensões mais altas do que as normalmente usadas em circuitos digitais. Uma vez programada, um EPROM pode ser apagado expondo-o a uma forte fonte de luz ultravioleta.

É fácil reconhecer uma memória EPROM, pois ela possui uma janela de quartzo fundido transparente na parte superior da embalagem, veja a Figura 198.

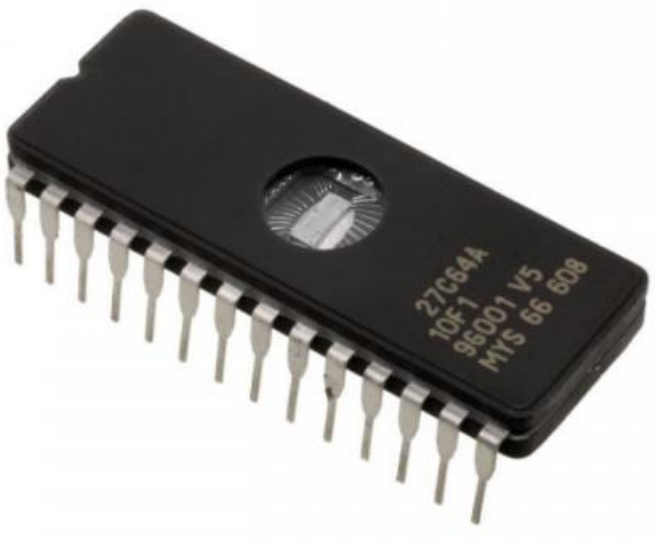


Figura 198 - Embalagem de uma memória EPROM.

Apesar de ser apagável e regravável esta memória é somente leitura, pois o processo de gravação é complexo e exige equipamentos específicos. A Figura 199 apresenta um gravador e um apagador de memórias EPROM.

Figura 199 - Equipamentos para gravar e apagar memórias EPROM.

Um exemplo de memória EPROM é a memória 27C64, apresentada na Figura 200. Esta memória armazena 8K palavras de 8 bits cada palavra.

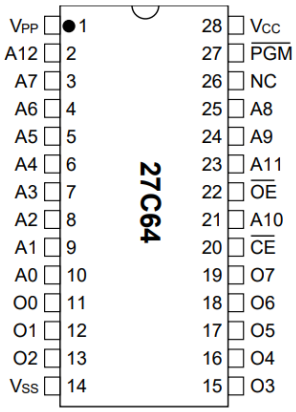


Figura 200 - Memória EPROM 27C64.

Devido à dificuldade de manuseio e ao custo este tipo de memória não é mais utilizado.

9.3.4. EEPROM

A memória EEPROM (*Electrically Erasable Programmable Read-Only Memory*) é uma evolução da EPROM, e com modificações nos transistores de porta flutuante que armazena as informações permite que seu conteúdo seja apagado eletricamente.

A principal vantagem da EEPROM é que ela pode ser programada e apagada no próprio circuito, isso aplicando sinais de programação especiais. Uma EEPROM suporta um número limitado de operações de escrita a apagamento, assim, a vida útil da EEPROM é uma consideração importante do projeto.

Outra vantagem desta tecnologia é que não é necessário apagar todo o componente de uma vez, pois é possível apagar e regravar palavras individualmente, o que torna o componente muito mais versátil.

A desvantagem deste tipo de memória está no tamanho da célula de armazenamento, que para permitir o apagamento e a gravação no circuito acaba ficando maior e com custo elevado. Assim, as EEPROMs são normalmente utilizadas para armazenar pequenas quantidade de informação. A Figura 201 apresenta o diagrama interno de uma EEPROM com capacidade para 32K palavras de 8 bits por palavra.

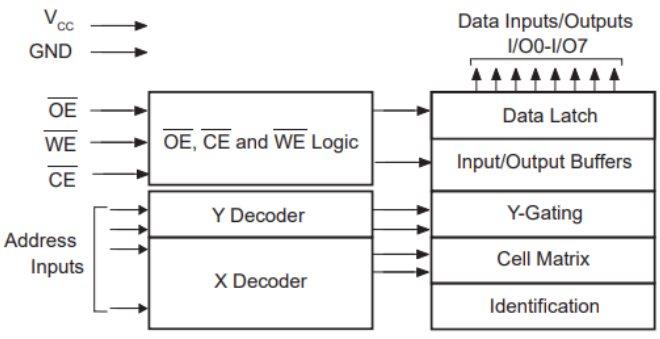


Figura 201 - Diagrama de uma EEPROM típica.

Um exemplo de memória EEPROM comercial é a memória AT28C256, apresentada na Figura 202, ela armazena 32K palavras de 8 bits por palavra.

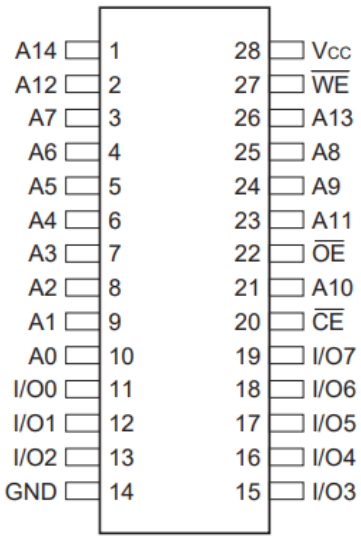


Figura 202 - Memória EEPROM AT28C256.

9.3.5. FLASH

A memória FLASH é uma combinação da memória EPROM com a memória EEPROM, conservando as melhores propriedades de cada uma delas. Na memória FLASH cada célula de armazenamento possui apenas um transistor, como na memória EPROM, mas é possível apagar e regravar seu conteúdo como na EEPROM.

Estas características tornam a memória FLASH uma memória de alta densidade e baixo custo. Desta forma a memória FLASH é a memória não volátil mais utilizada atualmente, estando presente em Microcontroladores Telefones Celulares, Pendrives, SSD e muito mais.

As memórias FLASH estão em constante evolução, com novos modelos menores, mais baratos e mais rápidos surgindo constantemente.

A Figura 203 apresenta o diagrama de blocos de uma memória Flash de 4M palavras de 16 bits cada palavra.

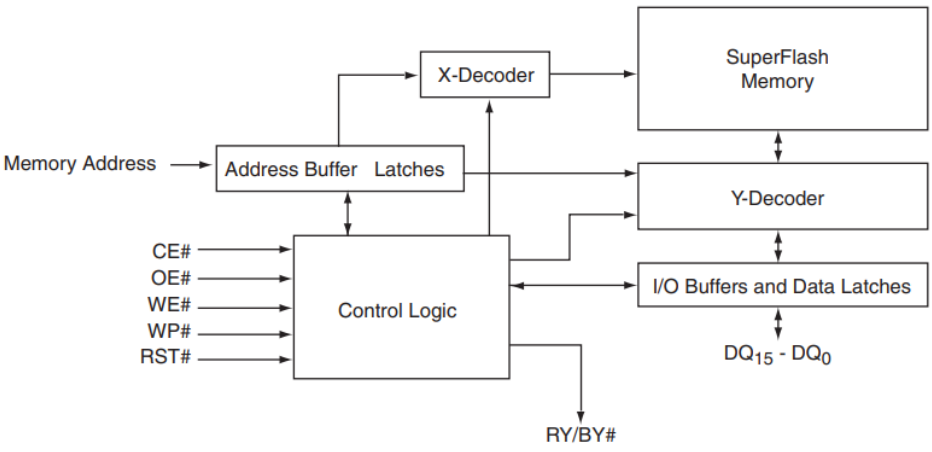


Figura 203 - Diagrama de uma memória Flash.

Um exemplo de memória Flash comercial é a memória SST38VF6401B, de 4M palavras de 16 bits cada palavra, apresentada na Figura 204.

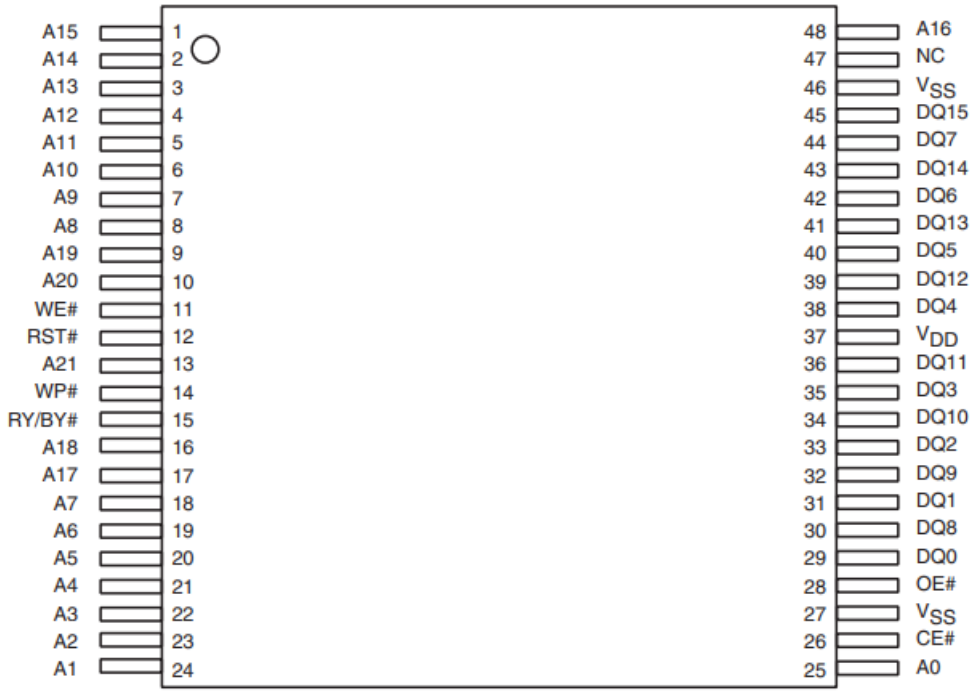


Figura 204 - Memória FLASH SST38VF6401B.

É importante destacar que apesar de ser gravável e apagável, as memórias FLASH não substituem as memórias voláteis utilizadas nos sistemas digitais. As memórias voláteis, como a SDRAM por exemplo, continuam sendo muito mais rápidas na leitura e principalmente na escrita dos dados. Além disso, as memórias FLASH possuem um número limitado de ciclos de apagamento e escrita, o que limita sua vida útil, já as memórias voláteis não têm esta limitação.

9.3.6. Memórias de próxima geração

A área de armazenamento não volátil é uma área onde muitas pesquisas vem sendo realizadas e muitas tecnologias vem se mostrando promissoras. A seguir são apresentadas algumas delas.

FRAM

A FRAM (*Ferroelectric RAM*) ou RAM ferroelétrica (FeRAM, F-RAM ou FRAM) é uma memória de acesso aleatório semelhante em construção a memória DRAM, porém ela possui uma camada de material ferroelétrico no lugar da camada dielétrica. Esta modificação permite que os dados sejam armazenados de forma não volátil.

Em comparação com a memória FLASH a FRAM apresenta menor consumo de energia, maior velocidade de gravação e maior durabilidade.

As desvantagens da FRAM são densidades de armazenamento muito mais baixas do que dispositivos flash, limitações de capacidade de armazenamento e custo mais alto. Além disso, o processo de leitura do FRAM é destrutivo, apagando o dado armazenado, o que exige uma arquitetura de gravação após leitura.

MRAM

A memória MRAM (*Magnetoresistive RAM*) é uma memória de acesso aleatório não volátil que armazena dados em domínios magnéticos. É uma memória com grande potencial para superar as tecnologias concorrentes e tornar a memória dominante ou mesmo universal, porém, as tecnologias de memória em uso atualmente, como flash e SDRAM, têm vantagens práticas que até agora mantiveram a MRAM em um nicho de mercado.

PRAM

A memória PRAM (*Phase-Change RAM*) é outra tecnologia em desenvolvimento que pretende no futuro ser a memória universal superando as memórias flash e SDRAM.

As PRAMs exploram o comportamento exclusivo de um tipo de vidro "*chalcogenide*" onde a passagem de uma corrente elétrica através de um elemento de aquecimento geralmente feito de nitreto de titânio e usado para aquecer o vidro, mudando seu estado cristalino. A informação é então armazenada nos diferentes estados cristalinos deste vidro.

Esta tecnologia ainda não está em estágio comercial.

9.4. Memórias com acesso serial

Nas memórias apresentadas até aqui, as entradas de endereço e as entradas e saídas de dados possuíam terminais independentes, ditos paralelos. Existe um outro tipo de memórias, onde os sinais de endereço e de dados são todos transmitidos pelos mesmos terminais. Esta tecnologia tem a vantagem de produzir circuito com número muito reduzido de terminais, porém a velocidade de gravação e leitura é drasticamente reduzida.

Estas memórias existem em diversos modelos com diversos protocolos de comunicação. Os principais protocolos de comunicação são o I2C e o SPI.

Como exemplo de memória serial I2C podemos citas a memória 24C64, que é uma memória EEPROM de 64K Bits. A Figura 206 apresenta um diagrama interno desta memória.

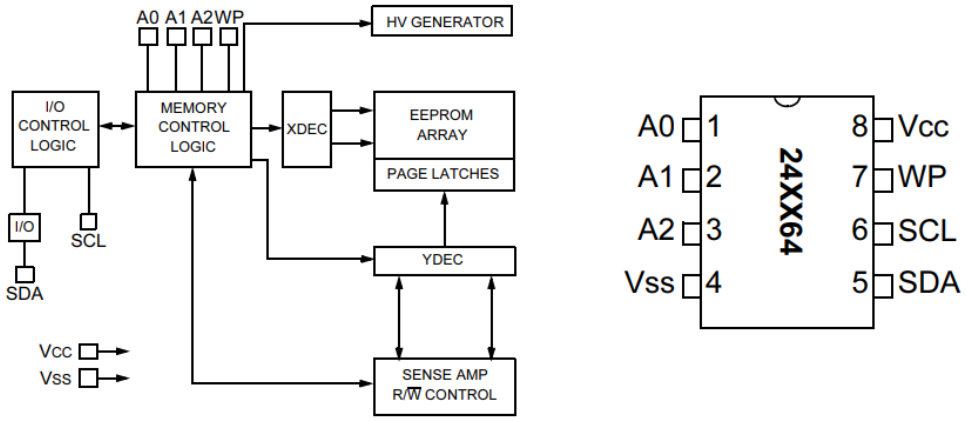


Figura 205 - Memória 24C64.

Observe o reduzido número de terminais no componente.

Toda a comunicação com o circuito integrado é realizada apenas por dois pinos, os terminais DAS e SCL, por isso se diz que esta é uma memória serial. A título de ilustração, a Figura 206 mostra em exemplo de sinal comunicação com esta memória.

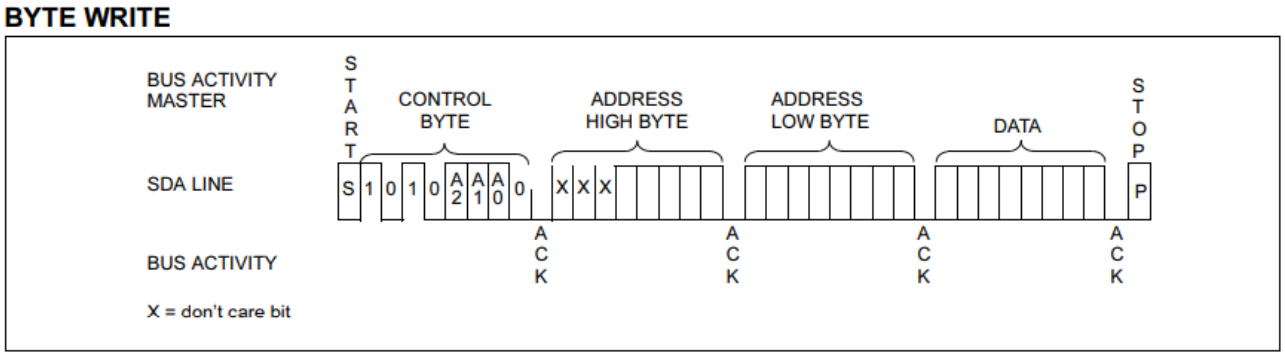


Figura 206 - Comunicação I2C com a memória 24C64.

9.5. Associação de memórias

A associação de memórias é utilizada quando o número de palavras ou o número de bits por palavra disponível em um determinado circuito integrado não é suficiente.

A Figura 207 apresenta a associação de duas memórias de modo a ampliar o número de bits por palavra. É importante observar que o número de entradas de endereço e consequentemente o número de palavras armazenadas na memória continua o mesmo, o que dobrou foi o número de bits por palavra, criando assim as conexões de IO8 a IO15.

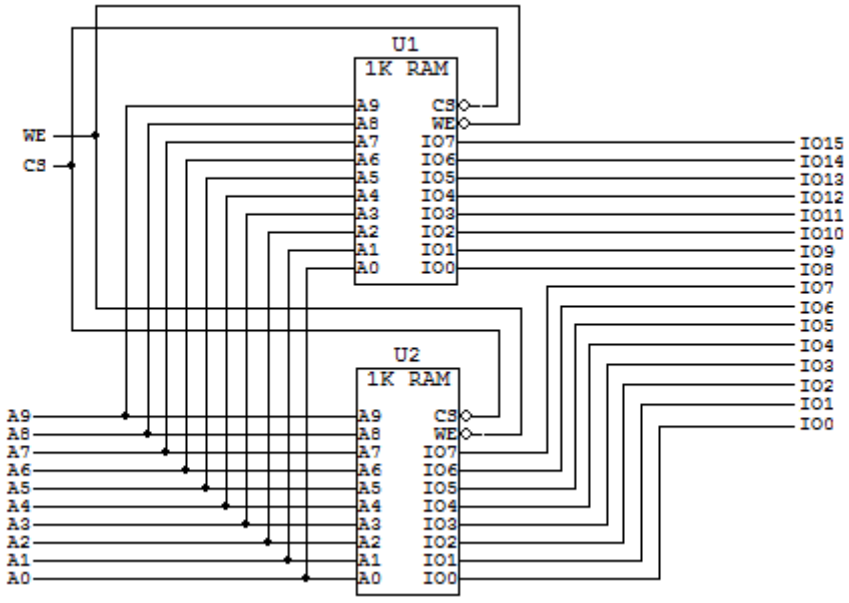


Figura 207 - Associação de memória para aumentar o número de bits.

A seguir temos a Figura 208 que apresenta a associação de dias memórias com o objetivo da aumentar o número de palavras e manter o número de bits de cada palavra. É importante observar que o número de conexões de dados não se alterou, porem surgiu mais uma entrada de endereço, chamada A10. Assim o conjunto passa agora a conter o dobro de palavras.

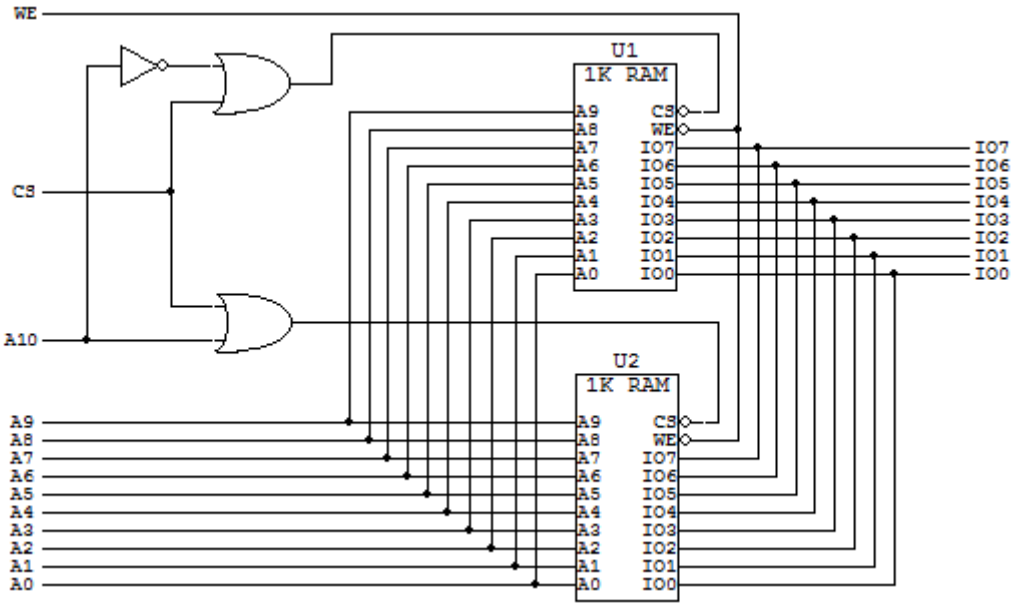


Figura 208 - Associação de memória para aumentar o número de palavras.

9.6. Exercícios

1) Construa no simulador um circuito que utilize uma memória RAM para armazenar informações. O circuito deve permitir armazenar dados na memória e ler os dados armazenados.
2) No simulador, faça uma associação de memórias como na Figura 207, implemente os circuitos necessários para testar seu funcionamento.
3) No simulador, faça uma associação de memórias como na Figura 208, implemente os circuitos necessários para testar seu funcionamento.

Aula 10 - Conversores A/D e D/A

O objetivo desta aula é apresentar aos alunos os conversores Analógico Digitais e Digitais Analógicos, utilizados para converter sinais elétricos analógicos de forma que possam ser processados por sistemas digitais e depois convertidos novamente em sinais elétricos analógicos.

10.1. Introdução

No início do curso foi feita uma breve introdução dobre os conversores analógico digitais (A/D) e digitais analógicos (D/A). Agora vamos aprofundar este assunto detalhando o funcionamento destes importantes componentes eletrônicos. Normalmente as grandezas físicas processadas pelos sistemas digitais são analógicas, assim é necessário convertê-las para sinais digitais para que possam ser processadas. Da mesma forma, as saídas dos sistemas digitais não podem ser diretamente entregues aos sistemas analógicos, necessitando ser convertidos.

Muitos sistemas digitais são utilizados para processar sinais oriundos de sensores analógicos, como por exemplo, microfones, sensores de temperatura, sensores de luminosidade etc. Assim, estes sinais necessitam primeiro ser convertidos de analógicos para digitais para só então serem processados. Os circuitos responsáveis por fazer esta conversão são os conversores analógico digitais (A/D).

O processo de transformar os sinais digitais em sinais analógicos é realizado pelos conversores digitais analógicos (D/A).

A Figura 209 apresenta um diagrama de blocos de um sistema onde um sinal analógico é convertido para digital por um conversor A/D. Este sinal é então processado por um sistema de processamento digital. O sinal digital resultante é então convertido para um sinal analógico por um conversor D/A.

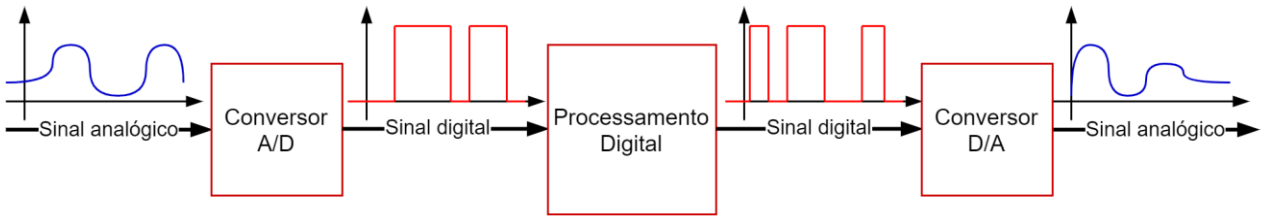


Figura 209 - Processamento digital de sinais

A Figura 210 apresenta um exemplo deste tipo de sistema digital. Trata-se de um controle digital de temperatura para um forno industrial. Neste forno existe um sensor de temperatura que tem seu sinal amplificado por um circuito amplificador. O sinal analógico resultante representa a temperatura do forno. Este sinal é então convertido para digital por um conversor A/D. O sinal digitalizado de temperatura é então entregue a um sistema digital, que referenciado por uma

interface de controle determina a potência necessária para manter a temperatura do forno no valor desejado. O sinal digital de saída do controlador é convertido para um sinal analógico por um conversor D/A e entregue a um circuito de potência. O circuito de potência ajusta a intensidade do aquecedor conforme o sinal recebido.

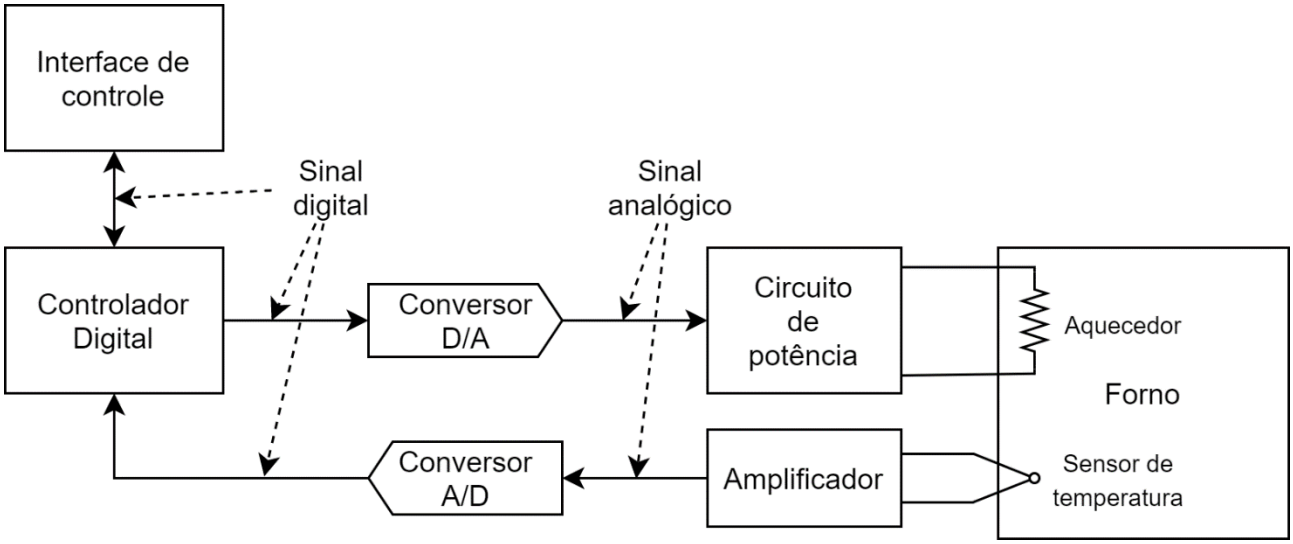


Figura 210 - Sistema digital de controle de temperatura.

Desta forma, a temperatura no interior do forno é controlada pelo sistema controlador digital conforme os ajustes realizados pelo operador na interface de controle.

A seguir serão estudados os conversores A/D e D/A em maiores detalhes.

10.2. Conversores A/D

Um conversor A/D converte um sinal analógico contínuo no tempo em um sinal digital de tempo discreto e amplitude discreta, ou seja, o processo de conversão envolve a quantização da entrada de tempos em tempos. Este processo sempre introduz uma pequena quantidade de erro ou ruído. O processo de conversão não é contínuo, ou seja, a entrada é amostranda periodicamente, o que limita a largura de banda do sinal de entrada. A Figura 214 mostra em exemplo de sinal amostrado por um conversor A/D.

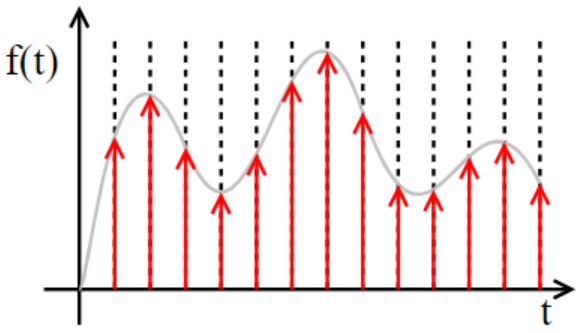


Figura 211 - Sinal amostrado.

As setas vermelhas na figura representam os momentos em que o conversor A/D realizou a medida do sinal.

Após a digitalização de um sinal o que se obtém é um conjunto de números que representa a amplitude do sinal no momento da conversão.

Algumas informações importantes relacionadas aos conversores A/D são listadas a seguir.

- **Resolução**, quantidade de bits do sinal digital resultante da conversão.
- **Taxa de amostragem**, número de medidas realizadas por segundo.
- **Tensão de referência (V_{ref})**, normalmente a tensão onde o conversor A/D apresenta o valor máximo de saída (todos os bits em 1).

Os conversores A/D são circuitos integrados que recebem como entrada um sinal analógico e apresentam em suas saídas digitais um valor binário proporcional a esta tensão de entrada. A entrada normalmente é limitada entre 0 e a tensão de referência V_{ref}, porém existem conversores A/D com referência negativa diferente de 0.

Um conversor A/D possui um determinado número de bits, que definem sua resolução. Este número de bits determina também o número de valores discretos possíveis na saída do conversor. Assim temos que o número de valores possíveis na saída é 2^n, onde n é o número de bits desta saída. A Figura 212 mostra o diagrama de um conversor analógico digital de 8 bits típico.

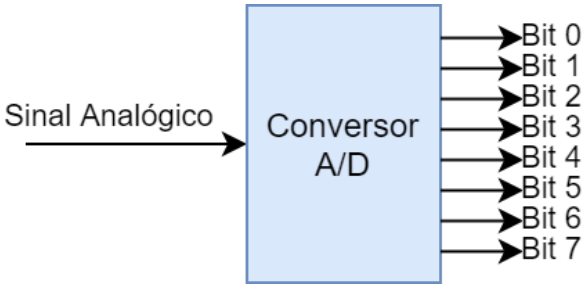


Figura 212 - Conversor Analógico Digital.

A Figura 213 apresenta a relação entre a tensão de entrada e o valor binário de saída para um conversor A/D de 3 bits. A tensão de referência V_{ref} neste caso é 1 V.

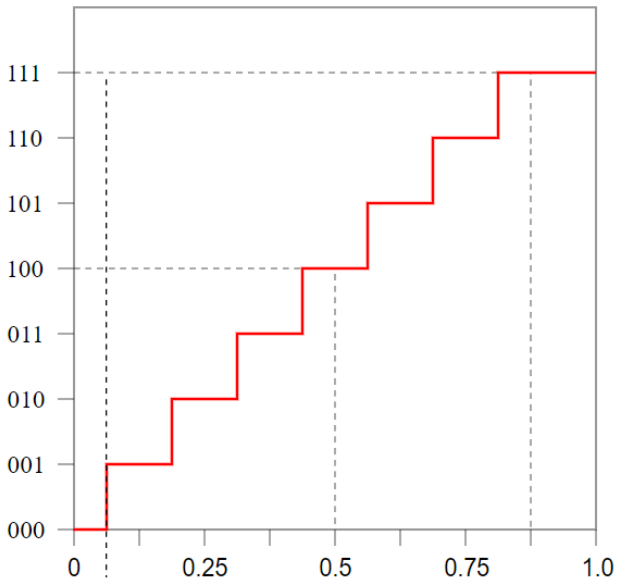


Figura 213 - Saída de um conversor A/D.

Observe que os valores binários do eixo vertical são proporcionais a tensão de entrada do eixo horizontal. Assim, o valor binário apresentado na saída do conversor A/D é proporcional a relação entre a entrada analógica e a tensão de referência, onde o maior valor binário possível representa uma tensão de entrada igual a tensão de referência. A expressão a seguir permite determina o valor binário aproximado na saída de um conversor A/D em função de sua tensão de entrada, da tensão de referência V_{ref} e uma resolução n em bits.

$$Saída = \frac{V_{entrada} \cdot (2^n - 1)}{V_{ref}}$$

Como a saída binária não apresenta casas decimais, o valor deve ser arredondado para um número inteiro.

Observe que para um conversor A/D de 10 bits, por exemplo, temos teremos $2^{10} = 1024$ possíveis valores de saída. Ou seja, uma faixa de saída binária de 0 a 1023. Outra informação importante é a resolução de tensão (Q) de um conversor A/D, ou seja, qual a variação de tensão na entrada que provoca uma variação no valor de sua saída. Este valor Q pode ser determinado pela seguinte expressão.

$$Q = \frac{V_{ref}}{2^n}$$

É possível observar que quanto maior o número de bits (n) menor é o valor Q, ou seja, mais sensível é o conversor A/D. Assim podemos afirmar que para aplicações que necessitam de maior precisão, um conversor A/D com um número maior de bits deve ser utilizado.

10.3. Conversor D/A

Um conversor digital analógico (D/A) realiza a operação contrária de um conversor A/D. Em um conversor D/A a entrada é um valor digital de n bits, e a saída é uma tensão analógica proporcional ao valor binário da entrada e da tensão de referência (V_{ref}). A Figura 214 mostra o diagrama de um conversor digital analógico de 8 bits típico.

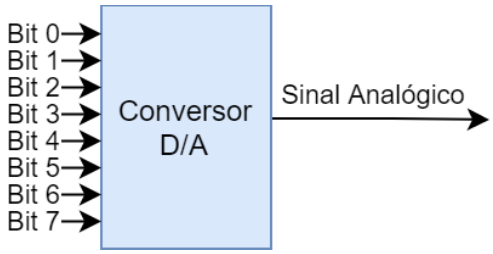


Figura 214 - Conversor Digital Analógico

A maioria dos conceitos apresentados para o conversor A/D podem ser aproveitados também para o conversor D/A. A tensão aproximada na saída ($V_{saída}$) de um conversor D/A pode ser calculada pela seguinte expressão.

$$V_{saída} = \frac{Entrada \cdot V_{ref}}{2^n - 1}$$

10.4. Exercícios

1) Construa no simulador um circuito com um conversor A/D de 8 bits. Em sua entrada conecte uma fonte variável de tensão e em sua saída conecte displays de 7 segmentos. Varie a tensão da entrada e verifique o funcionamento do conversor A/D.
2) Construa no simulador um circuito com um conversor D/A de 8 bits. Em sua entrada conecte 8 chaves e displays de 7 segmentos, de forma a variar os sinais digitais de entrada do conversor. Verifique então o funcionamento do conversor D/A medindo a tensão analógica em sua saída.

Aula 11 - Dispositivos lógicos programáveis

O objetivo desta aula é apresentar aos alunos os dispositivos lógicos programáveis, seus tipos, tecnologias e aplicações. Serão presentadas as características e as formas de programação das principais famílias de dispositivos lógicos programáveis.

11.1. Introdução

Os circuitos integrados normalmente utilizados nos sistemas digitais tem sua função definida pelos seus projetistas e são fabricados para desempenhar esta função, sem possibilidade de modificação ou atualização. Assim para desenvolver novos sistemas digitais é usual utilizar vários destes circuitos integrados em uma placa de circuito impresso. Esta placa fornece as interconexões necessárias ao funcionamento do sistema.

Estes sistemas, assim como os circuitos integrados que os compões não podem ser facilmente alterados ou atualizados, o que é uma desvantagem podendo tornar o sistema obsoleto rapidamente.

Outras desvantagens dos sistemas digitais compostos por vários circuitos integrados são o tamanho do sistema resultante e o consumo de energia envolvido. Sistemas com muitos circuitos integrados são grandes e consomem muita energia. A velocidade de operação também é comprometida, pois as conexões elétricas nas placas de circuito impresso limitam a frequência de operação dos sinais.

Uma alternativa é a utilização de microcontroladores ou microprocessadores. Estes componentes podem executar muitos tipos de operações diferentes, seguindo a lógica descrita em um programa. Esta abordagem resolve o problema de atualização da lógica do sistema, pois basta atualizar o programa em execução.

Porém este tipo de sistema apresenta uma limitação importante, as instruções do programa são executadas sequencialmente, uma a uma, limitando o número de operações que podem ser executadas paralelamente. Para realizar mais operações é necessário um processador mais poderoso, o que aumenta o custo do sistema e a energia consumida.

Os dispositivos lógicos programáveis (*Programmable Logic Device* - PLD) foram desenvolvidos para aproveitar o melhor das duas abordagens anteriores. Trata-se de dispositivos compostos por circuitos digitais e não programas, assim podem executar várias operações paralelamente, contudo, estes dispositivos podem ter sua lógica programada, ou reconfigurada, conforme a necessidade do projetista.

A utilização das PLDs permite o desenvolvimento de sistemas digitais rápidos e com baixo consumo de energia, sem perder a flexibilidade, podendo ser atualizados conforme necessário sem alterações físicas na placa de circuito impresso.

Assim podemos dizer que um dispositivo lógico reconfigurável é um componente eletrônico, na forma de um circuito integrado que permite construir circuitos lógicos reconfiguráveis.

Esta categoria de circuitos digitais surgiu na década de 1970, onde alguns fabricantes de circuitos integrados começaram a fabricar dispositivos programáveis, compostos por várias portas lógicas onde a ligação entre estas portas podia ser definida pelo usuário e ficava gravada no dispositivo através de tecnologia de memória não volátil PROM ou EPROM.

Com o passar dos anos a tecnologia evoluiu e surgiram várias famílias de PLDs, as principais serão apresentadas a seguir.

11.2. SPLD

O termo SPLD vem do inglês "*Simple Programmable Logic Device*" ou dispositivo lógico programável simples. Esta categoria de PLDs engloba os dispositivos mais simples, e que deram início a tecnologia de lógica reconfigurável. Fazem parte desta categoria as PLAs, PALs e as GALs, apresentadas a seguir.

11.2.1. PAL

Uma PAL (*Programmable Array Logic*) é uma matriz lógica programável, ou seja, é um tipo de dispositivo lógico programável usado para implementar circuitos lógicos combinacionais (e não sequenciais). As PALs são compostas por um conjunto de portas E programáveis, que se conectam a um conjunto de planos de portas OU fixas. Este arranjo permite implementar funções digitais do tipo soma de produtos. A Figura 215 apresenta o diagrama interno simplificado de uma PAL. Os pequenos círculos são conexões que podem ser programadas (conectadas ou desconectadas) pelo usuário.

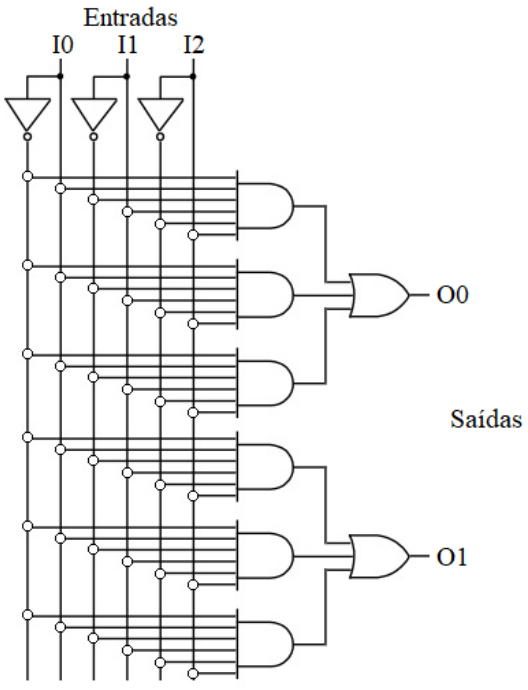


Figura 215 - Diagrama de uma PAL.

A programação de uma PAL é realizada gravando na memória não volátil do dispositivo as informações das conexões de cada uma das entradas, de forma direta ou invertida, nas portas E.

11.2.2. PLA

Uma PLA (*Programmable Logic Array*) é uma matriz lógica programável, ou seja, assim como a PAL é um tipo de dispositivo lógico programável usado para implementar circuitos lógicos combinacionais (e não sequenciais). As PLAs são compostas por um conjunto de portas E programáveis, que se conectam a um conjunto de planos de portas OU, também programáveis, que podem ser interconectadas para produzir uma saída. A Figura 216 apresenta o diagrama interno simplificado de uma PLA. Os pequenos círculos são conexões que podem ser programadas pelo usuário.

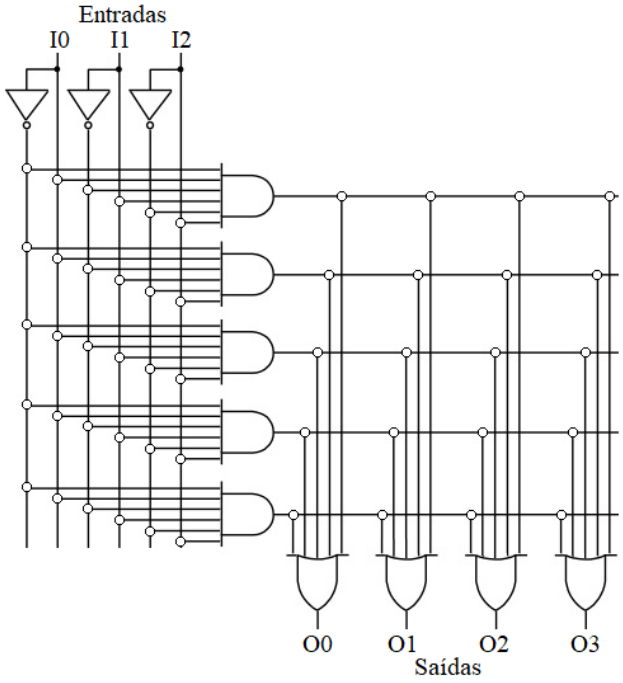


Figura 216 - Diagrama de uma PLA.

Observe que programando as diversas possíveis conexões é possível criar circuitos combinacionais complexos, apenas limitados pelo número de entradas e saídas disponível.

As PLAs diferem das PALs pois permitem programar tanto as entradas das portas E como as entradas das portas OU.

11.2.3. GAL

Uma GAL (Generic Array Logic) ou matriz lógica genérica é uma evolução das PALs onde em cada saída é adicionada um módulo chamado OLMC (*Output Logic MacroCell*). Esta macro-célula é um componente versátil composto por um flip-flop tipo D, portas OU Exclusivo, multiplexadores etc, podendo ser programada internamente e assim desempenhando várias funções.

Por possuir flip-flops, as GAL são capazes de implementar não apenas circuitos combinacionais, mas também circuitos sequenciais, como contadores e registradores, ou seja, uma GAL tem a capacidade de armazenar informações digitais nos flip-flops.

A Figura 217 apresenta parte do diagrama interno de uma GAL, onde é possível visualizar a matriz de interconexões, duas entradas e duas saídas com suas macro-células OLMC.

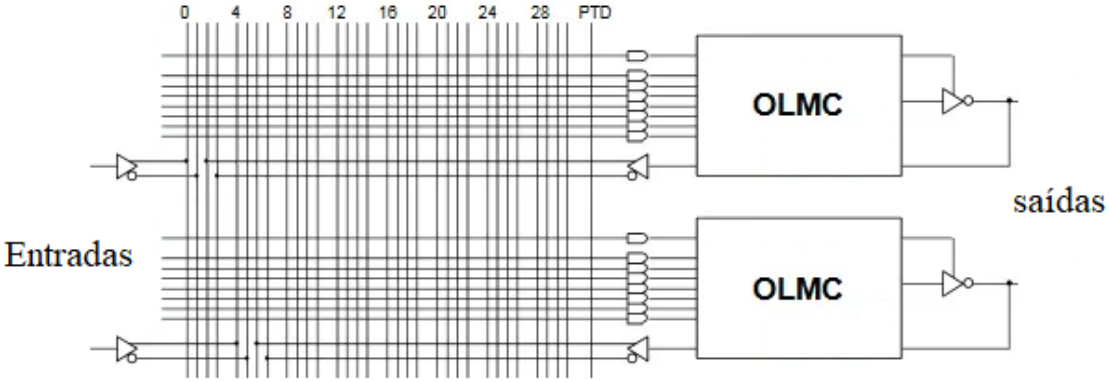


Figura 217 - Diagrama interno parcial de uma GAL.

A Figura 218 mostra o diagrama interno simplificado de uma macro-célula típica.

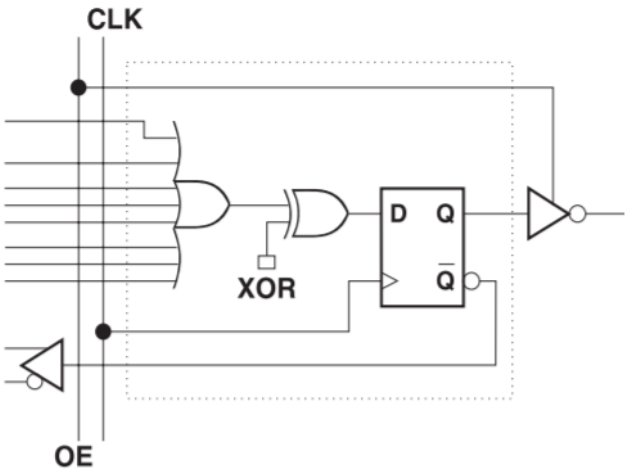


Figura 218 - Exemplo de macro-célula.

Outra vantagem importante das GALs é que elas são dispositivos mais modernos e utilizam memória EEPROM para armazenar as interligações, o que facilita sua gravação e alteração.

Como exemplo de GAL comercial podemos citas a GAL16V8, produzida por diversos fabricantes. A Figura 219 mostra o circuito integrado ATF16V8 que é uma GAL16V8, produzida pela ATMEL.

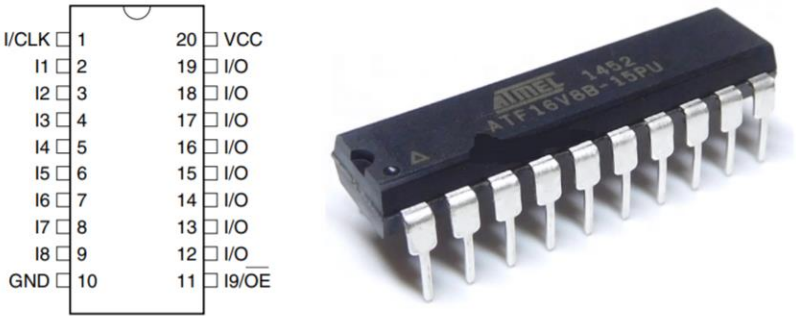


Figura 219 - Circuito integrado GAL ATF16V8.

Um diagrama de blocos deste dispositivo é apresentado na Figura 220.

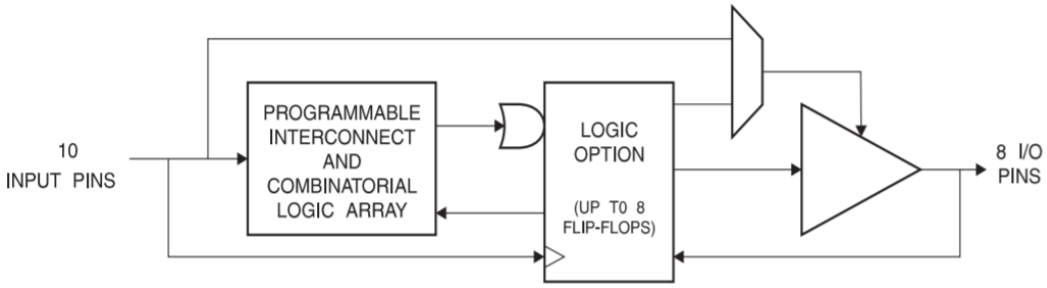


Figura 220 - Diagrama interno do circuito integrado GAL ATF16V8.

As GALs apesar de funcionais e ainda empregadas em alguns sistemas estão ultrapassadas, sendo substituídas nos sistemas mais modernos pelas CPLDs.

11.3. CPLD

Uma CPLD (*Complex Programmable Logic Array*) ou matriz lógica programável complexa é, em uma visão muito simplista, um circuito integrado composto por várias GALs e por barramentos que permitem sua interconexão. As principais vantagens das CPLDs são o maior número de entradas e saídas, o maior número de elementos lógicos integrados e padrões elétricos de entrada e saída mais modernos.

Os blocos construtivos de uma CPLD são as LAB (*Logic Array Blocks*) ou blocos de matriz lógica. Cada uma delas se assemelha a uma GAL, porém com várias melhorias. A Figura 221 a presenta um exemplo da estrutura interna de uma CPLD.

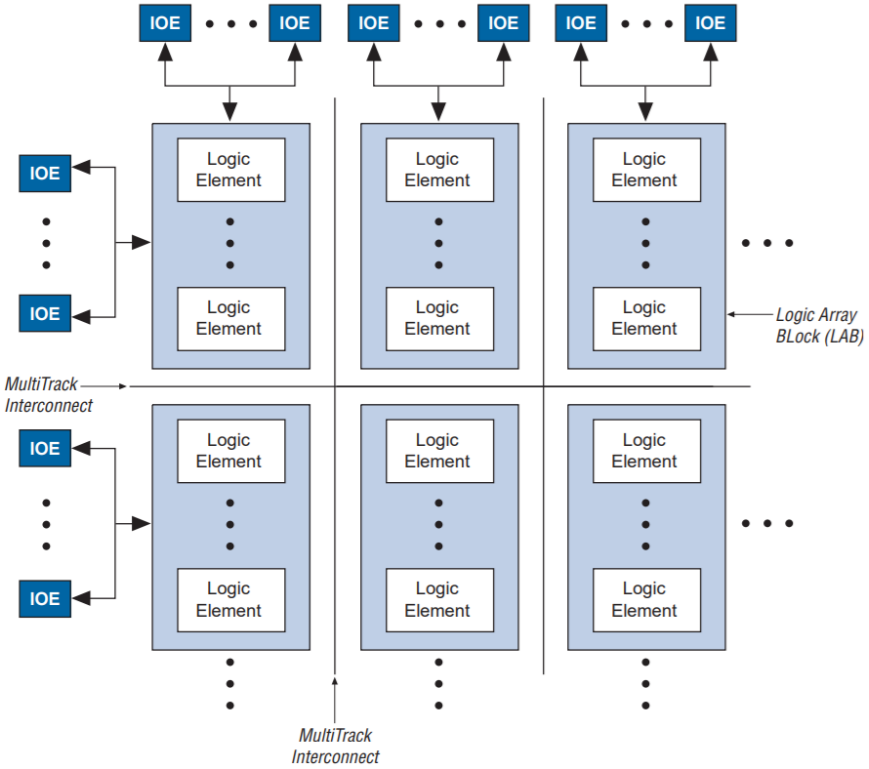


Figura 221 - Exemplo de estrutura interna de uma CPLD.

Cada LAB é composta por um conjunto de elementos lógicos (*Lógic Elements LE*), conjuntos de interligações destes LE, sinais de controle e tabelas de consulta (*look-up table LUT*). São disponibilizadas interconexão locais para transferir sinais entre os LEs de um mesmo LAB e conexões globais para transferir sinais entre as diferentes LABs da CPLD. A Figura 222 mostra um exemplo de LAB.

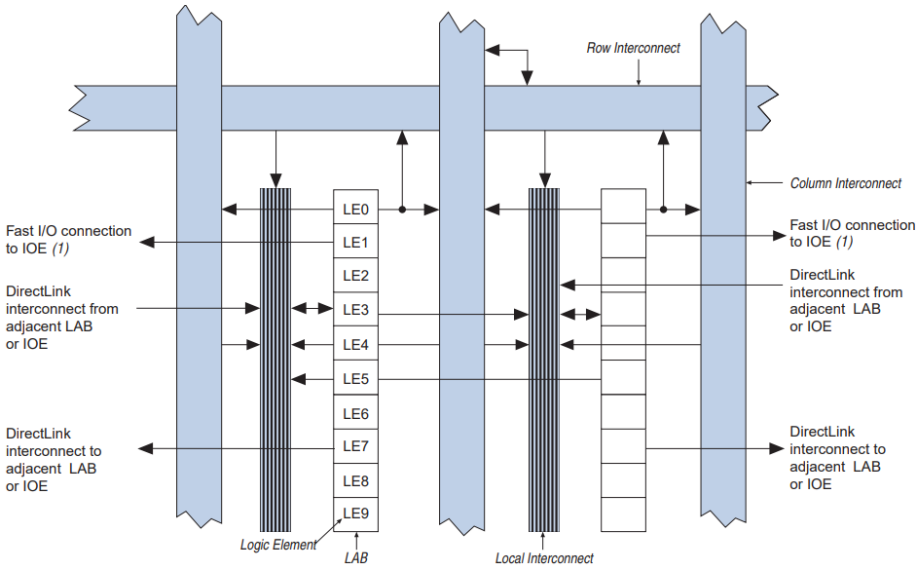


Figura 222 - Um exemplo de LAB.

Os elementos lógicos LE são os principais componentes das LABs e consequentemente das CPLDs, são semelhantes as macro-células utilizadas nas GALs, porém com diversas melhorias. Vela um exemplo na Figura 223.

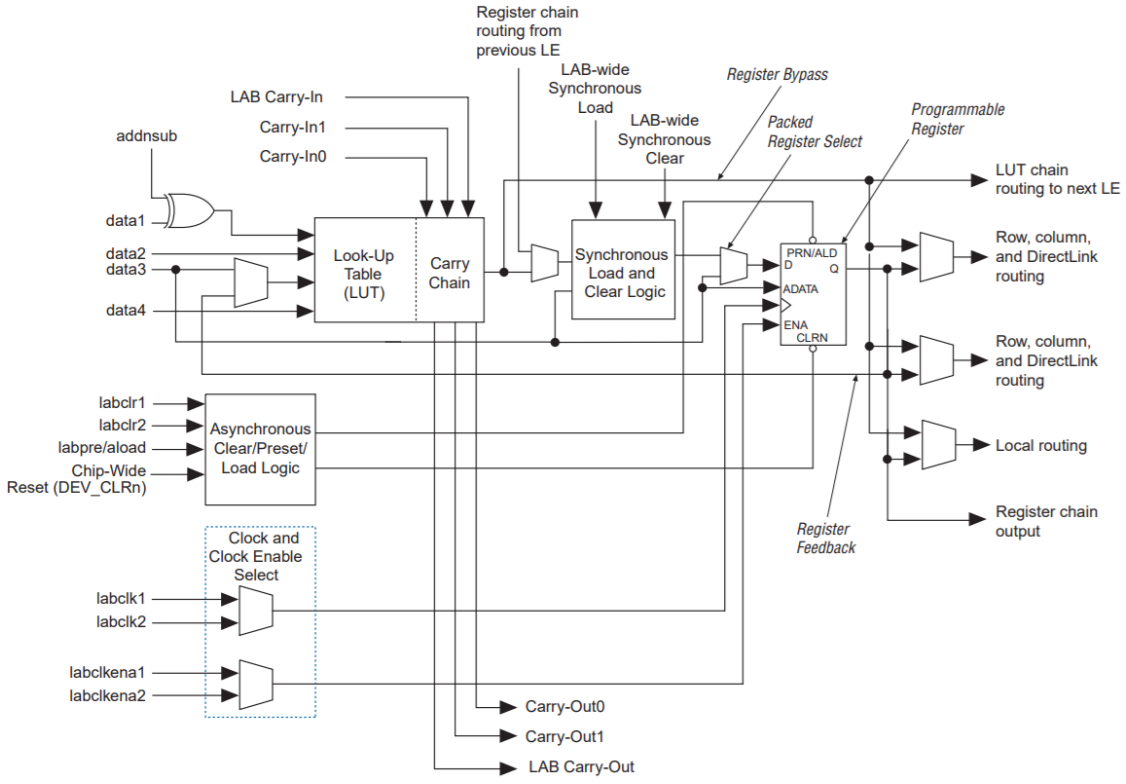


Figura 223 - Exemplo de elemento lógico (LE).

Cada elemento lógico possui diversos componentes, dos quais pode-se destacar a tabela de consulta (*look-up table LUT*) e o registrador. A LUT é capaz entre outras coisas de emular o comportamento de qualquer circuito combinacional de "n" entradas e uma saída, onde "n" depende do modelo da LUT. Este componente é responsável pela parte combinacional do circuito. O registrador por sua vez é capaz de armazenar um bit de informação, e é responsável pela parte sequencial do circuito.

O armazenamento das conexões internas da CPLD continua sendo realizado com a tecnologia EEPROM, o que torna estes dispositivos fáceis de programar e de utilizar.

Assim, as CPLDs são superiores aos dispositivos lógicos programáveis anteriores em vários aspectos, principalmente na capacidade, no número de entradas e saídas, na velocidade e no consumo de energia, além de serem compatíveis com padrões de entrada e saída mais modernos, com níveis de tensão mais baixos. As CPLDs são disponibilizadas com diferentes números de LE, indo der algumas dezenas até milhares destes componentes.

11.3.1. Um exemplo de CPLD

Existem diversos fabricantes cada um com vários modelos de CPLD, como exemplo vamos apresentar uma CPLD chamada EPM240, fabricada pela ALTERA, agora adquirida pela INTEL, e que pertence a família MAX II de CPLDs. Trata-se de uma CPLD com 240 LE, 80 pinos de entrada e saída, 8192 bits de memória não volátil e que pode operar a até 304 MHz. A Figura 224 mostra ente componente e sua pinagem.

Figura 224 - CPLD EPM240 da família MAX II.

11.3.2. Periféricos das CPLDs

As CPLDs são circuitos integrados maiores e mais complexos, e assim alguns fabricantes aproveitam para integrar outros elementos importantes para a construção dos sistemas digitais no mesmo dispositivo. Cada fabricante e cada modelo apresenta um conjunto diferente de dispositivos

periféricos integrados a CPLD, com cada dia surgindo novas opções no mercado. Alguns dos periféricos mais comumente integrados são:

- Blocos de interface de entrada e saída compatíveis com vários níveis de tensão.
- Blocos de memória não volátil para armazenamento de dados.
- Portas de conexões para programação no próprio circuito.

11.4. FPGA

FPGA (*Field Programmable Gate Array*) ou matriz de portas programável em campo é uma categoria de dispositivo lógico programável que é uma evolução das CPLDs. As FPGAs, se diferenciam das CPLDs, no número de LE disponibilizado, no tamanho, no custo e no desempenho. Além disso, a tecnologia empregada em sua fabricação e o número de periféricos embutidos também é diferente.

As FPGAS também são construídas com blocos parecidos com as LABs, porém estes blocos são mais modernos e menores. Assim um número muito maior destes blocos é utilizado na construção do circuito integrado. A Figura 225 mostra um destes blocos, chamado de *Adaptive Logic Module* (ALM), utilizado nas FPGAs da família Cyclone V da INTEL.

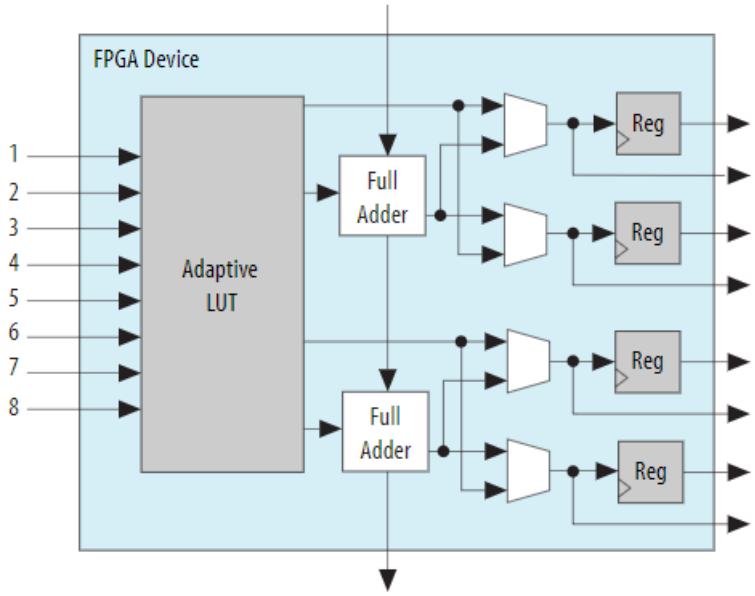


Figura 225 - Exemplo de bloco construtivo de uma FPGA.

Estas ALMs executam a função de vários LE, sendo também compostos por uma tabela de consulta (*look-up table LUT*) e alguns registradores.

As FPGAs construídas com pilhas contendo um número muito grande destas ALMs, com cada fabricante e cada modelo tendo suas próprias variações e características.

Enquanto as CPLD podem chegas a milhares de LEs, as FPGAs podem chegar a ter o equivalente a centenas de milhares ou até milhões de LEs.

Uma característica marcante das FPGAS é que na maioria dos modelos, para reduzir a área ocupada e aumentar a velocidade de operação as interconexões são armazenadas por bits de memória RAM. Ou seja, as FPGAs são construídas com memória volátil. Neste caso, sempre é necessário conectar uma memória EEPROM a FPGA, onde suas conexões ficam armazenadas. Toda vez que o sistema é energizado a FPGA faz automaticamente a leitura da EEPROM e recarrega os dados de comexão.

11.4.1. Periféricos das FPGAs

Uma característica importante de FPGAs são os periféricos embutidos no circuito integrado, veja alguns exemplos a seguir.

- Blocos de interface de entrada e saída compatíveis com vários níveis de tensão e de alta velocidade.
- Blocos de memória RAM de uso geral.
- Blocos DSP (*Digital Signal Processor*), blocos para multiplicação numérica, que aceleram o processamento de sinais digitais.
- Controladores para memória DSRAM externa.
- Portas de conexão PCI Express.
- *Phase-locked loops* (PLLs), sintetizadores de sinais de clock de precisão.
- Processadores de múltiplos núcleos.
- Interfaces USB.
- Interfaces Ethernet.
- Blocos criptográficos, aceleram a criptografia dos dados.

É importante salientar que nem todos os modelos possuem todos os tipos de periféricos, e a cada dia são lançados novos dispositivos com características diferentes.

11.4.2. Exemplos de FPGA

A Figura 226 mostra dois exemplos de FPGA, uma Cyclone IV de baixo custo e uma StratiX 10 de alta performance, ambas fabricadas pela INTEL.

Figura 226 - Exemplos de FPGA.

É possível notar na figura que as FPGAs são dispositivos grandes, com muitos terminais o que torna sua utilização bastante complexa.

Para facilitar o desenvolvimento de aplicações com FPGAs são disponibilizadas placas de desenvolvimento, como as mostradas na Figura 227.

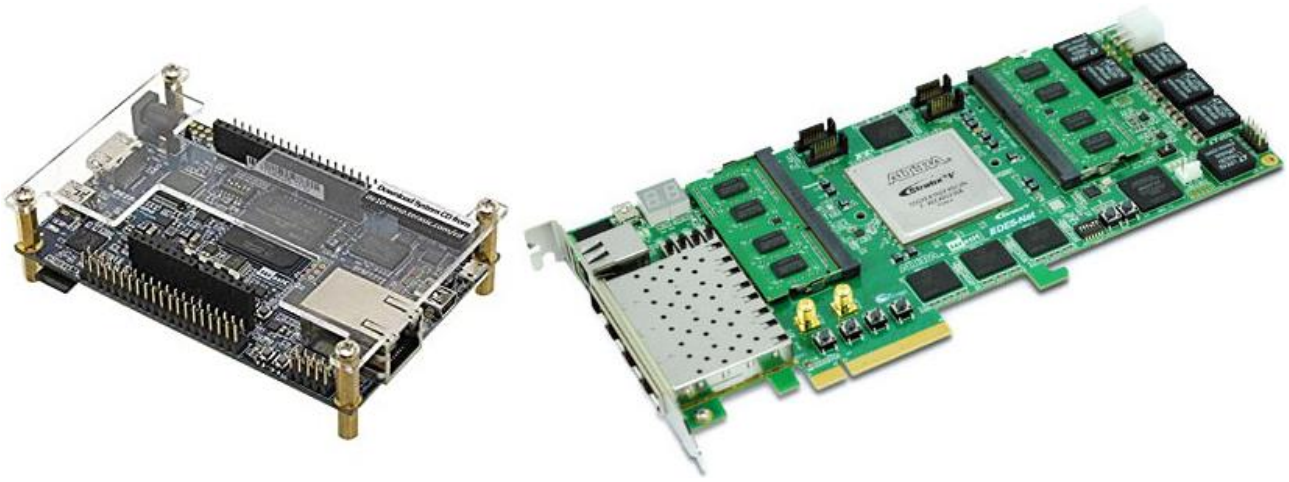


Figura 227 - Placas de desenvolvimento FPGA.

A placa da esquerda possui uma FPGA da família Cyclone V com um processador ARM de 2 núcleos integrado, e serve para desenvolvimento de sistema digitais em geral. Já a placa da direita, possui uma FPGA muito mais poderosa, da família STRATIX V, sendo utilizada principalmente para aplicações de inteligência artificial e visão computacional. Observe que esta placa foi desenvolvida para ser conectada a um computador.

11.5. Programação das PLDs

Como já pode ser observado, para que se possa utilizar os dispositivos lógicos programáveis é necessário realizar a sua programação. A maioria dos fabricantes disponibiliza ferramentas de software e hardware para realizar esta tarefa.

Os dispositivos mais simples são programados nos mesmos gravadores das EEPROMs, apresentado anteriormente. Porém, as CPLD e FPGAS, podem ser programadas na própria placa de circuito. Para isso são utilizados software como este apresentado na Figura 228.

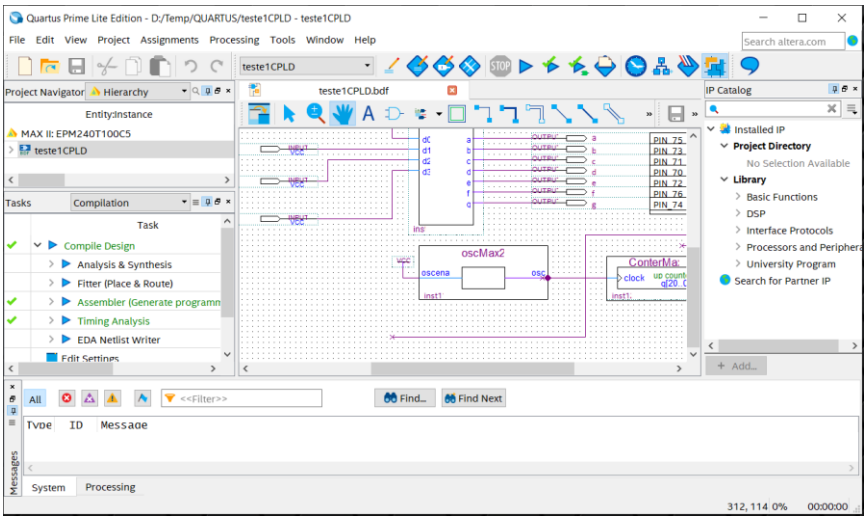


Figura 228 - Software para programação das PLD.

O software apresentado é o QUARTUS, possui versão gratuita e paga e serve para a programação de dispositivos da INTEL.

Para transferir os dados de descrição do circuito do computador para o circuito impresso com o dispositivo, são utilizados programadores como o da Figura 229.

Figura 229 - Programador de PLDs.

Programadores deste tipo são conectados a placa onde o dispositivo a ser programado esta inserido, e ao computador através de um cabo USB, permitindo assim a transferência dos dados de descrição do circuito do computador para o dispositivo.

Existem várias formas de descrever os circuitos que se deseja gravar no dispositivo e assim gerar os dados de descrição do circuito. Os principais são através dos diagramas de circuito e de linguagens de descrição de hardware, como veremos a seguir.

11.5.1. Diagramas de circuitos

Para sistemas digitais simples é possível desenhar o diagrama do circuito que se deseja gravar no dispositivo, e então gravar este circuito na placa. Veja um exemplo na Figura 230.

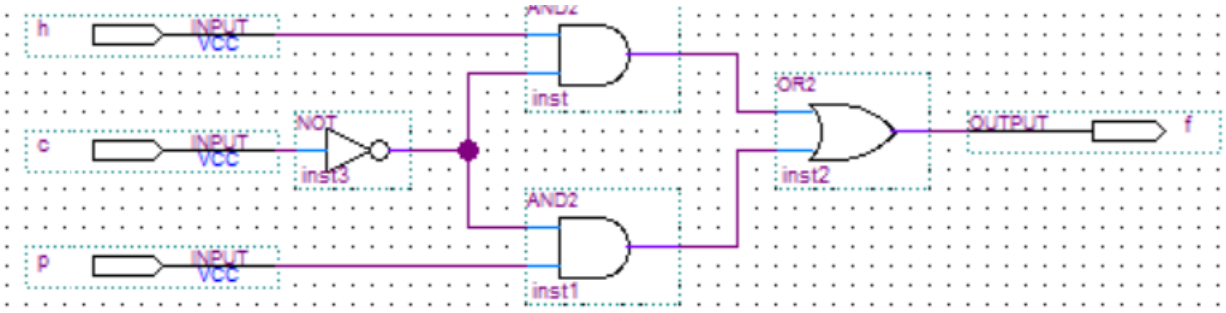


Figura 230 - Programando PLDs através de diagramas.

11.5.2. Linguagem de descrição de hardware

Outra abordagem para descrição dos circuitos a serem gravados nas PLDs é a utilização das chamadas linguagens de descrição de hardware. Trata-se de linguagens parecidas com linguagens de programação utilizadas para descrever o circuito a ser gravado no dispositivo.

Como exemplos podemos de linguagem de descrição de hardware podemos citas a linguagem VHDL e a Linguagem Verilog.

A Figura 231 apresenta um exemplo de descrição de hardware realizado em VHDL. Trata-se de um trecho do código de um driver para display de 7 segmentos.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity Driver7SegMAX2 is
5
6      port
7      (
8          d0 : in std_logic;
9          d1 : in std_logic;
10         d2 : in std_logic;
11         d3 : in std_logic;
12
13         a  : out std_logic;
14         b  : out std_logic;
15         c  : out std_logic;
16         d  : out std_logic;
17         e  : out std_logic;
18         f  : out std_logic;
19         g  : out std_logic
20     );
21
22 end entity;
23
24 architecture rtl of Driver7SegMAX2 is
25 --signal ta, tb, tc, td, te, tf, tg: std_logic;
26 begin
27 PROCESS (d0, d1, d2, d3)
28 begin
29 if (d0 = '0' and d1 = '0' and d2 = '0' and d3 = '0') then -- 0
30     a <= '1';
31     b <= '1';
32     c <= '1';
33     d <= '1';
34     e <= '1';
35     f <= '1';
36     g <= '0';
37 elsif (d0 = '1' and d1 = '0' and d2 = '0' and d3 = '0') then -- 1
38     a <= '0';
39     b <= '1';
40     c <= '1';
```

Figura 231 - Exemplo de descrição de hardware em VHDL.

11.6. Simulações

Uma etapa importante no desenvolvimento de sistemas digitais utilizando PLDs é a verificação do funcionamento do circuito. Como nem sempre é possível testar o sistema físico, os fabricantes disponibilizam softwares específicos para este fim. Veja um exemplo na Figura 232.

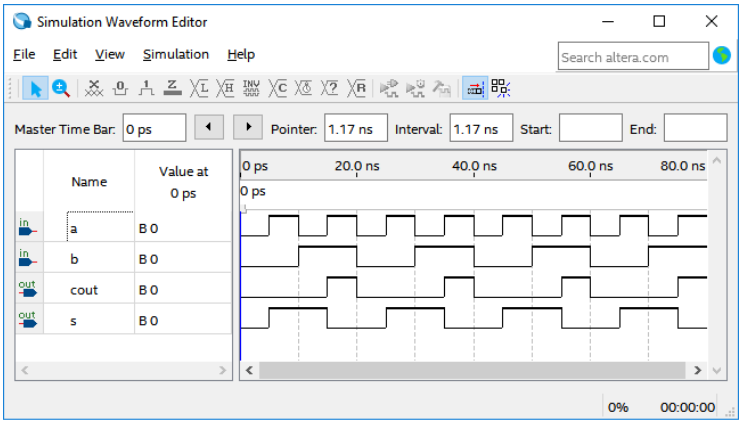


Figura 232 - Simulação de PLDs.

Com estas ferramentas é possível simular as entradas e verificar as saídas dos circuitos projetados para serem gravados nos dispositivos programáveis.