

UNIDADE 2 – FUNCIONAMENTO DOS CIRCUITOS COMBINACIONAIS E SEQUENCIAIS

MÓDULO 1 – SISTEMAS DIGITAIS

01

INTRODUÇÃO

Durante o estudo deste módulo você terá oportunidade de entender o que seja sistema para ficar mais simples a compreensão do que sejam sistemas digitais. Os sistemas digitais modernos possuem diversos níveis de complexidade. Estes componentes podem ser uma simples chave de liga e desliga até sistemas computacionais completos. Estudaremos também a importância do cientista inglês George Boole e do matemático americano Shannon na utilização da álgebra booleana para demonstrar as propriedades dos circuitos elétricos.

1. CONCEITO DE SISTEMA

Para compreendermos melhor o que são **circuitos combinacionais e sequenciais** é interessante começarmos pelo entendimento do que venha a ser **sistema**. Trata-se de um conceito aparentemente simples, porém, é um dos mais abrangentes e complexos de compreender. Mas não se assuste, vamos trabalhar juntos no entendimento deste conceito. Vamos lá?

Você provavelmente já vivenciou a seguinte situação: uma pessoa verifica que seu aparelho celular está tendo problemas de “falta de sinal” e não consegue efetuar ligações. A pessoa entra em contato com a operadora de telefonia celular e após determinado tempo recebe como resposta que “foi verificado problema no sistema e assim que o sistema estiver em funcionamento o sinal será restabelecido”. Você também já deve ter ouvido a seguinte expressão: “a culpa de tudo isso é este “sistema de transporte” ultrapassado e que não funciona...” Enfim, o que não falta são exemplos do uso da palavra sistema.

A origem da palavra sistema é do grego “sietemiun”, que significa formar conjunto. Um sistema pode ser definido como um conjunto de elementos que se relacionam e interagem com o objetivo de desempenhar determinada função.

02

No caso do mundo da computação, os componentes de um sistema computacional estão interconectados, de modo a formar um todo organizado, sendo assim, cada elemento desempenha determinada função e deve atingir uma meta geral.

Um sistema deve ter sua função bem definida e esta definição é que irá identificar as funcionalidades de seus componentes. Uma das muitas funções de um computador é transformar as informações que você acessa na Internet e torná-las possíveis de serem entendidas, pois por trás das imagens há uma

infinidade de bits e bytes, códigos, programas etc. Sendo assim, é possível identificar dois aspectos básicos em qualquer sistema:

- sua estrutura e
- seu comportamento.

A estrutura é reflexo dos componentes e a maneira como estão interconectados, já o comportamento é reflexo da funcionalidade do sistema.

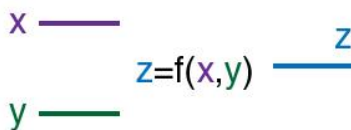
Nos sistemas em que há vários componentes, uma das características será a complexidade das inter-relações, então é por isso que são chamados de **sistemas complexos**. Em projetos que adotam os chamados sistemas complexos, a identificação da ordem ou regularidade é de suma importância, o que exige uma abordagem sistemática e estruturada. Em virtude de qual aspecto está sendo identificado, a representação que deve ser utilizada terá um nível de abstração adequada. As representações mais comuns utilizadas são de três tipos:

- a comportamental,
- a estruturada e
- a física.

03

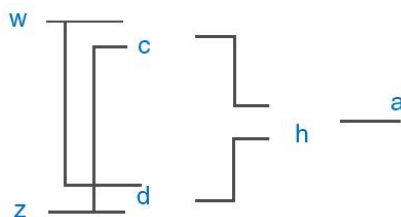
No caso da **representação comportamental**, a identificação do sistema é feita como uma “caixa preta” (difícil de saber o que contém seu interior) e se concentra na especificação do comportamento por meio de uma função com valores de entrada e de tempo (ver figura 01 abaixo).

Fazendo um resumo, uma representação comportamental mostra como é o funcionamento, mas não mostra como é feita a implementação de determinado sistema (caixa preta), definindo apenas as saídas da “caixa preta” para qualquer tipo de combinação de valores de entrada, mas não descreve como é a construção ou projeto do sistema conforme certos componentes.



04

Já no caso da **representação estrutural** há uma definição da “caixa preta” como o conjunto de seus componentes e suas respectivas interconexões. Oposto da representação comportamental, a representação estrutural se atém às especificações e à implementação do sistema sem fazer referência à sua funcionalidade. Você pode observar isso na figura 02.

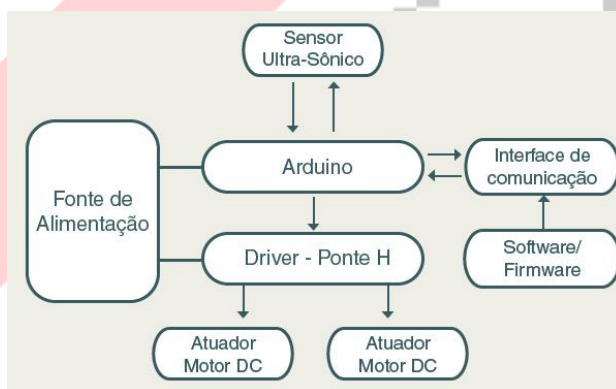


Fica explícito que na maioria das vezes a funcionalidade pode ser obtida a partir dos componentes interconectados. Contudo, derivar a funcionalidade de um sistema desta forma é muito complicado, principalmente se o sistema possui um número grande de componentes.

05

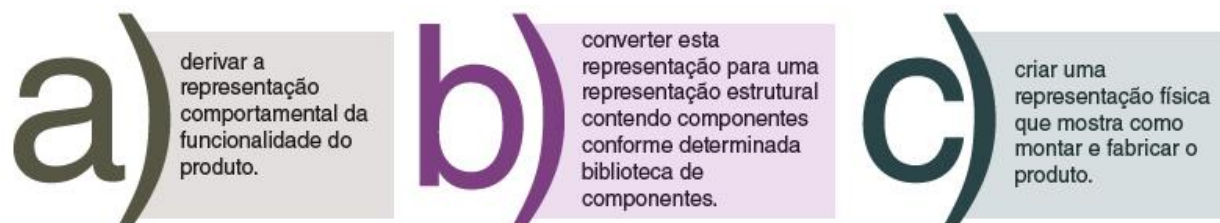
A **representação física** de uma “caixa preta” especifica suas características, mostrando suas dimensões e a localização de cada componente e a conexão existente na descrição estrutural do sistema. No entanto a representação estrutural oferece a conectividade do sistema, somente a representação física faz a descrição precisa das relações espaciais dos vários componentes. Sendo assim, a representação física é utilizada para fazer a descrição do sistema logo após sua fabricação, especificando quais serão suas dimensões, peso, consumo de energia elétrica e as posições de cada dispositivo de entrada ou saída.

Uma maneira simples e natural para representação da estrutura de um sistema é utilizando-se de diagramas de bloco. No diagrama de bloco os componentes podem ser representados por retângulos, chamados blocos, e as conexões que são representadas por linhas que interligam os blocos. A figura 03 mostra um exemplo de diagrama do Arduino (hardware de prototipagem eletrônica).



06

A maneira como é feito o projeto de sistemas, principalmente relacionado a sistemas digitais e em particular a produtos eletrônicos em geral, tem por base sempre ao menos três fases, cada uma tendo como norte uma das representações de projeto:



Ao pararmos para pensar sobre o que vimos até agora quanto a sistemas e representações, entendemos que em qualquer projeto é possível montar ou criar algo quando seguimos estes passos (projeto) que utilizam diferentes tipos e níveis de abstração. Em determinado nível de abstração são representados apenas determinados detalhes. Geralmente, os detalhes identificados em uma dada fase do projeto dependem do quanto é complexo o sistema. Podemos usar o seguinte exemplo: é muito difícil ou quase impossível fazer um projeto de um microprocessador fazendo uso apenas das portas lógicas básicas.

Geralmente, o projeto começa utilizando blocos básicos no nível lógico. Logo depois, estes blocos serão interconectados para comporem um sistema mais complexo. Esta seria uma maneira viável para projetar um microprocessador.

07

2- SISTEMAS DIGITAIS

Vamos agora começar nosso estudo sobre sistemas digitais. Nosso objetivo é fazer uma pequena introdução sobre o que é um sistema digital. Daremos uma especial atenção aos componentes que são básicos para este sistema.

Podemos definir um sistema digital como aquele que é formado por um conjunto de componentes interconectados que processam informações em forma digital ou discreta.

Na maioria dos sistemas digitais, os componentes básicos utilizados são dispositivos eletrônicos chamados circuitos integrados (CIs). As conexões entre estes componentes eletrônicos são ligações físicas por meio das quais as informações digitais são transmitidas.

Os sistemas digitais modernos abrangem diversos níveis de complexidade. Os componentes utilizados para construir um sistema digital são compostos desde o componente do tipo chave “liga/desliga” e vai até um sistema computacional completo.

A quantidade de componentes que compõe um sistema digital pode variar de apenas um até vários componentes. Você já deve ter chegado à conclusão que quanto maior a quantidade de componentes necessários à implementação de um sistema digital, maior será sua complexidade, consequentemente, mais difícil de compreender como é seu funcionamento e de como fazer seu projeto. Tudo isso evidencia a importância do uso de níveis de abstração no processo de projeto de sistemas digitais.

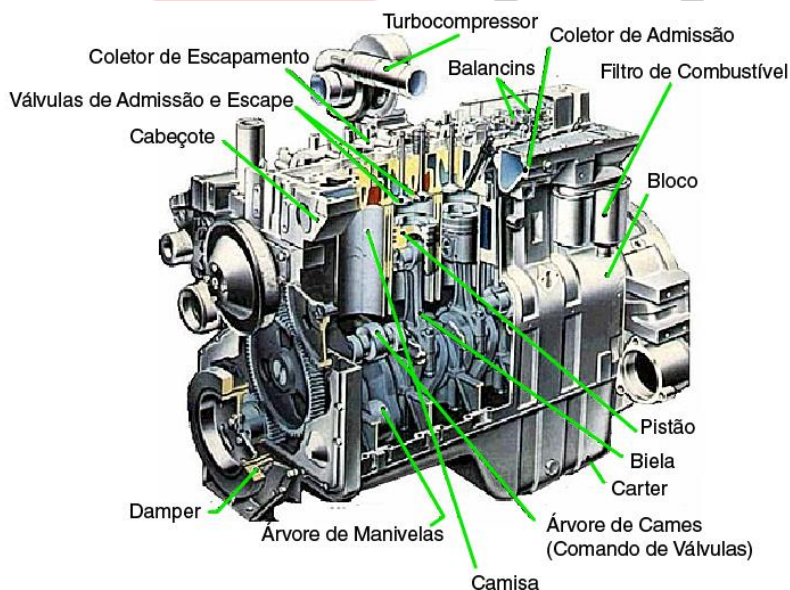
Você provavelmente deve estar perguntando: o que são **níveis de abstração**? Vamos juntos entender este conceito.

Começemos pelo conceito da palavra **abstração**. Há vários conceitos e definições para abstração, no dicionário, abstração significa: “ideias desvinculadas da realidade” ou “isolar mentalmente para considerar à parte um elemento de representação que não é dado separadamente na realidade”.

Abstração tem sua origem do latim *abstractio*, que significa uma forma de mentalmente isolar em um conceito determinado elemento à exclusão de outros do todo.

Um engenheiro, ao fazer a planta para construção de uma casa, utiliza certos símbolos, desenhos e traçados que representam paredes, quartos, janelas, canos, pilares, vigas e etc. Esta planta da obra é uma abstração do que será a casa ao final da construção.

Você também pode pesquisar na Internet e encontrar sites que mostram os componentes de um motor automobilístico e como é a interação de seus componentes, com base nestas informações podemos imaginar (abstrair) como seriam estes componentes interagindo durante o funcionamento real do motor.



Um nível de abstração, ou de granularidade, é caracterizado pelo tipo de objetos utilizados na representação. Geralmente, é possível identificar 4 diferentes tipos de objetos em um produto eletrônico:

- registradores,
- processadores,
- portas e
- transistores.

Na tabela abaixo é possível observar os níveis de abstração.

NÍVEL	COMPORTAMENTO	ESTRUTURA	FÍSICO
Transistor	Equações diferenciais, diagramas corrente- voltagem	Transistores, resistores, capacitores	Células analógicas e digitais
Portas	Equações Booleanas, máquinas de estado finitas (FSM)	Portas lógicas, Flip-flops	Módulos, unidades
Registrador	Algoritmos, <i>flowcharts</i> , conjunto de Instruções, generalizações de FSMs	Somadores, comparadores, contadores registradores	Microcircuitos
Processador	Especificação executável, programas	Processadores, controladores ASICs	Placas de circuito impresso, módulos multicircuitos

Conforme podemos observar na tabela acima, os principais componentes no nível de transistores são transistores, resistores e capacitores, que, ao serem combinados, formam os circuitos analógicos e digitais para realizar e exercer determinada funcionalidade. Esta funcionalidade é geralmente representada por um conjunto de equações diferenciais ou por determinado tipo de relacionamento entre corrente e tensão. Representar fisicamente estes circuitos, chamados **células**, consiste da composição ao **nível de transistores** e das conexões que os conectam.

10

Os componentes principais do **nível de portas** são os flip-flops e as portas lógicas.

NÍVEL	COMPORTAMENTO	ESTRUTURA	FÍSICO
Portas	Equações Booleanas, máquinas de estado finitas (FSM)	Portas lógicas, Flip-flops	Módulos, unidades

Portas lógicas são circuitos especializados que fazem uso de operações booleanas, exemplo: OR e AND.

Um flip-flop é um elemento básico de memória que tem a capacidade de armazenamento de 1 bit de informação, consequentemente ele pode assumir um valor falso ou desligado (0) ou verdadeiro ou ligado (1).

As portas e flip-flops são células digitais que, ao serem grupadas, formam módulos ou unidades aritméticas e de memória.

Os módulos são utilizados na composição básica ao nível de registradores. Utilizando-se de equações booleanas e Finite State Machines - FSMs (diagramas de máquinas de estados finitas) é possível descrever o comportamento de cada módulo.

11

Os **registradores** têm por seus principais componentes as unidades de memória e aritméticas, tais como: multiplicadores, comparadores, somadores, contadores, filas, registradores, bancos de registradores e etc. Cada um destes componentes é um módulo, que possui dimensões padronizadas ou fixas, tendo um atraso de propagação e um conjunto fixo de posições para as saídas e entradas do módulo.

NÍVEL	COMPORTAMENTO	ESTRUTURA	FÍSICO
Registrador	Algoritmos, <i>flowcharts</i> , conjunto de Instruções, generalizações de FSMs	Somadores, comparadores, contadores registradores	Microcircuitos

Estes componentes do nível de registradores são montados e interconectados em microcircuitos, que geralmente são usados como componentes básicos no nível de abstração descrito na tabela. De maneira geral, os microcircuitos mencionados são exemplificados por diagramas de FSMs ou tabelas de estados, fluxogramas e conjuntos de instruções.

12

O **nível de processador** pode ser considerado como o nível mais alto de abstração apresentado na tabela 01. Os componentes básicos são: memórias, processadores, controladores e interfaces, e os chamados: Application Specific Integrated Circuits - ASICs (Circuitos de Aplicação Específica).

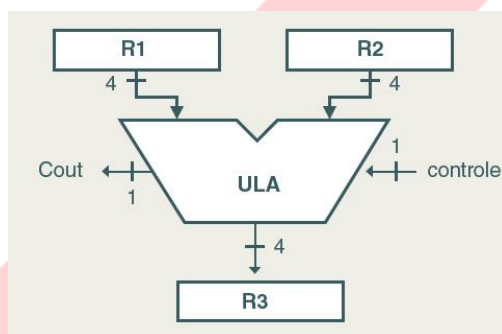
NÍVEL	COMPORTAMENTO	ESTRUTURA	FÍSICO
Processador	Especificação executável, programas	Processadores, controladores ASICs	Placas de circuito impresso, módulos multicircuitos

De maneira geral, os componentes expostos acima são montados em uma placa de circuito impresso e conectados com fios que são impressos na placa.

A maneira de como será o comportamento de sistemas compostos destes componentes do nível de processadores é normalmente feita por meio de linguagem natural, uma especificação executável utilizando-se de Hardware Description Language – HDL – linguagem de descrição de hardware, ou um programa ou algoritmo feito em uma linguagem de programação, como, por exemplo: assembly (notação legível feito por seres humanos para o código de máquina).

13

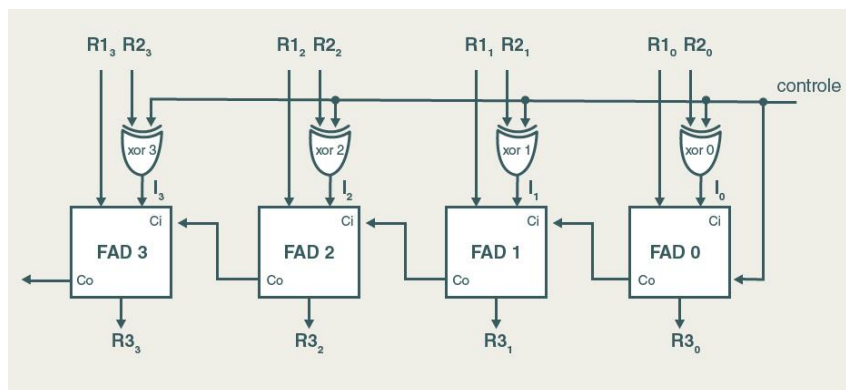
A representação estrutural comumente utilizada para sistemas digitais é o diagrama de blocos. Dependendo do nível de abstração, cada bloco irá representar os objetos correspondentes ao nível de uma porta lógica quando relacionado ao nível de abstração de portas; registradores, somadores quando relacionado ao nível de registradores; controladores e ASICs quando relacionado ao nível de processador. As interconexões são representações da comunicação entre os objetos.



A figura 04 é um exemplo de um diagrama de blocos, que representa uma ULA (Unidade Lógica e Aritmética) e também os registradores para a entrada e saída. Eles são componentes que constituem partes essenciais de qualquer tipo de sistema computacional.

14

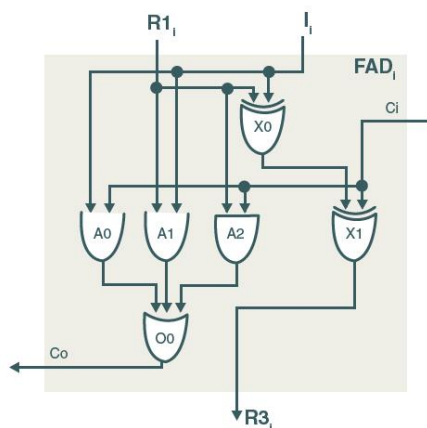
Você deve ter observado que, em um diagrama de blocos, não há preocupação quanto ao detalhamento dos componentes, mas apenas uma ideia generalizada da estrutura como um todo. Cada bloco poderá até ser mostrado em um nível maior de detalhamento por meio de outro diagrama de blocos ou em um diagrama esquemático. Isto demonstra a natureza hierárquica que há em qualquer tipo de projeto de sistema.



Para exemplificar o que falamos no parágrafo acima, podemos observar na figura 05, uma representação de um diagrama de blocos com um nível de detalhamento maior da ULA (Unidade Lógica e Aritmética). Você já havia observado na Figura 04 um diagrama de blocos que representa a ULA de uma maneira menos detalhada.

15

Na figura abaixo, figura 06, continuamos a detalhar ainda mais, vendo com detalhes como é um somador completo, que é um dos blocos que representa um dos componentes da ULA.



Contudo, para a representação do comportamento de sistemas digitais no nível de portas lógicas, torna-se necessária a utilização de **tabelas verdades** e **equações booleanas**. No nível de registradores, utiliza-se de determinados tipos de linguagem para a descrição de como ocorrem as transferências de dados entre os registradores. Com o objetivo de possibilitar uma simulação do sistema no nível de processador, geralmente utilizamos as chamadas **linguagens procedurais**. Estas linguagens fazem uma descrição dos algoritmos na forma de programas executáveis.

16

Vamos agora usar uma descrição algorítmica. O algoritmo abaixo foi criado tendo por base o diagrama de blocos que representa a estrutura da ULA conforme a figura 4:

```
se (controle=0)
    R3_R1+R2;
senão
    R3_R1-R2;
```

Este pequeno algoritmo mostra que o valor de (controle) irá determinar se a ULA realizará uma soma ou uma subtração.

17

3. TIPOS DE DADOS E REPRESENTAÇÃO DE DADOS

Anteriormente, falamos sobre notação posicional em que estudamos os sistemas de numeração e conversão de bases. Quando falamos em tipos de dados, o objetivo é entender que nos sistemas digitais a codificação mais utilizada é a binária.

Os dados encontrados nos atuais computadores, que são sistemas digitais, são classificados em uma das seguintes categorias:

- Números utilizados em cálculos matemáticos;
- Letras do alfabeto, utilizadas para processar dados;
- Os chamados símbolos discretos, que são utilizados para vários propósitos.

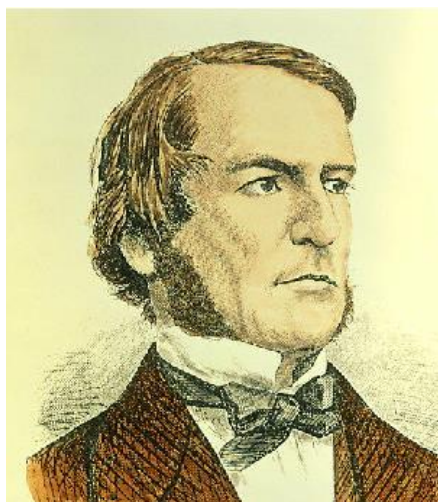
Você estudou anteriormente que nos sistemas computacionais os dados são representados em formato binário porque este formato facilita a criação dos projetos de circuitos eletrônicos, pois há apenas duas condições possíveis, as quais foram convencionadas para serem interpretadas como os valores 0 e 1 de um dígito binário (bit). Tais circuitos eletrônicos são projetados para realizar um repertório de operações necessárias que são disponibilizadas nos computadores.

18

4. ÁLGEBRA BOOLEANA E CIRCUITOS LÓGICOS

Você observou nos parágrafos anteriores que falamos sobre operações e lógica booleana. Para entendermos este conceito vamos começar falando sobre o conceito conhecido como: álgebra booleana.

Podemos definir como álgebra booleana o conjunto de operadores e um conjunto de axiomas (verdades inquestionáveis universalmente válidas), que são assumidos verdadeiros sem precisar de prova.



George Boole (1815-1864)

No ano de 1854, o matemático inglês George Boole introduziu uma maneira sistemática e formal que até hoje é usada para o tratamento sistemático da lógica, conhecida como álgebra de Boole ou álgebra booleana.

Com o passar dos anos, quase um século depois, outro cientista e matemático, o norte-americano Claude Elwood Shannon, teve a ideia de usar o sistema matemático de George Boole para analisar e projetar circuitos. Shannon utilizou a álgebra booleana para fazer a demonstração das propriedades dos circuitos elétricos de chaveamento com dois valores.

19

Oposto ao que conhecemos na matemática de nosso dia a dia, também conhecida como a álgebra ordinária dos reais, as variáveis podem assumir valores no intervalo $(-8;+8)$ – infinito ao + infinito, entretanto, as variáveis booleanas só podem assumir um **número finito** de valores. Em especial, na álgebra booleana de dois valores, cada variável pode assumir um dentre dois valores possíveis, os quais podem ser denotados por falso ou verdadeiro (F,V), ligado ou desligado ou zero/um (0,1). Neste nosso estudo iremos utilizar a notação de zero e um $[0,1]$, que utilizaremos também no estudo da eletrônica digital ou circuitos digitais.

Em virtude do pequeno valor (finito) que cada variável pode assumir, a opção de estados que uma função booleana assume também será limitado (finito), conseqüentemente se torna fácil descrever completamente as funções booleanas com o uso de tabelas. Em virtude deste detalhe, uma tabela que descreva uma função booleana é chamada de tabela verdade.

Ao estudarmos a álgebra booleana é importante entender que existem três funções básicas ou operações. São elas:

- operação “OU”,
- operação “E” e
- complementação.

Todas as funções booleanas podem ser representadas por meio destas operações básicas.

Tabela verdade

É na tabela verdade que podemos listar todas as combinações de valores que as variáveis de entrada podem vir a assumir e os correspondentes valores da função, que serão as saídas.

20

4.1 - OPERAÇÃO OU (ADIÇÃO LÓGICA)

Uma definição para a operação “OU”, que também é chamada de adição lógica, é:

A operação “OU” resulta “1” se pelo menos uma das variáveis de entrada valer “1”.

Você viu nos parágrafos anteriores que uma variável booleana ou vale “1” ou vale “0”, e como o resultado de uma operação qualquer pode ser visto como (ou atribuído a) uma variável booleana, basta que definamos quando a operação vale 1. Automaticamente, a operação resultará 0 no restante dos casos. Sendo assim, pode-se concluir que a operação “OU” resulta “0” apenas quando todas as variáveis de entrada tiverem o valor “0”.

Para representar a operação “OU” podemos usar o símbolo da adição algébrica dos números reais, o símbolo “+”. Entretanto, em virtude de estarmos trabalhando com variáveis booleanas, sabemos que não é uma adição algébrica, mas na verdade uma adição lógica.

Observe a figura abaixo, que lista as possibilidades de combinações entre dois valores booleanos (0/1) e os correspondentes resultados para a operação “OU”.

Preste muita atenção: a figura abaixo é a representação de uma operação booleana de adição lógica, pois na álgebra matemática (número reais) $1+1=2$

$$\begin{array}{l} 0 + 0 = 0 \\ 0 + 1 = 1 \\ 1 + 0 = 1 \\ 1 + 1 = 1 \end{array}$$



Observe que a operação OU só pode ser descrita caso haja, ao menos, duas variáveis envolvidas. Sendo assim, não será possível realizar uma operação sobre apenas uma variável. Em virtude disso, o operador “+” (OU) é dito **binário**.

21

Nas equações, geralmente não escrevemos todas as possibilidades de valores. Apenas convencionamos que uma letra (ou uma letra com um índice) irá designar uma variável booleana. Sendo assim, já sabemos que aquela variável poderá assumir ou o valor “0” (zero) ou o valor “1” (um).

Consequentemente, supondo que vamos demonstrar o comportamento da equação “A OU B” (entende-se como: $A + B$), poderíamos reproduzir o resultado por meio da tabela verdade abaixo:

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Vamos agora acrescentar mais uma variável. Não se preocupe, pois da mesma maneira, podemos simular o comportamento da equação “A OU B OU C” por meio de uma tabela verdade. Outra maneira de representar a mesma equação seria: $A+B+C$ (entende-se como: A **OU** B **OU** C). Como na equação há apenas o símbolo “+”, trata-se de uma operação “OU” com três variáveis. Consequentemente aplica-se diretamente a definição da operação **OU**. Vamos relembrar esta definição? Então vamos: o resultado será “1” (um) se pelo menos uma das variáveis de entrada valer “1” (um). Observe a tabela verdade abaixo:

A	B	C	A + B + C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

22

Torna-se importante frisar que, em virtude do fato de haver somente um operador na equação, pode-se fazer também uma avaliação da equação dividindo-a em pares. Para melhorar o entendimento vamos supor que primeiramente vamos achar o resultado de $A+B$, para então operar os valores resultantes com os respectivos valores de C. Esta propriedade é conhecida como associativa. A ordem das variáveis não é importante em virtude da propriedade comutativa, sendo assim a ordem em que são avaliadas as variáveis A, B e C é irrelevante. Estas propriedades são visualizadas pela tabela verdade abaixo. Nesta tabela, os parêntesis servem para indicar as subexpressões que já foram avaliadas na coluna imediatamente à esquerda.

Você deve ter observado que os valores das colunas referentes às expressões $A+B+C$, $(A+B)+C$ e $(B+C)+A$ são os mesmos (na mesma ordem).

A	B	C	A + B + C	A + B	(A + B) + C	B + C	(B + C) + A
0	0	0	0	0	0	0	0
0	0	1	1	0	1	1	1
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1

Propriedade associativa

Quer dizer que, ao associarmos algumas parcelas, o total fica inalterado. No conjunto dos números naturais, a adição é associativa, isto é, é possível associar as parcelas de quaisquer modos, ou seja, com três números naturais, somando o primeiro com o segundo e ao resultado obtido somarmos um terceiro, obteremos um resultado que é igual à soma do primeiro com a soma do segundo e do terceiro.

Exemplo:

$$(2 + 3) + 4 = 2 + (3 + 4) = 9$$

Propriedade comutativa

Quer dizer que a ordem das parcelas não altera a soma. No conjunto dos números naturais, a adição é comutativa, pois a ordem das parcelas não altera a soma, ou seja, somando a primeira parcela com a segunda parcela, teremos o mesmo resultado que somando a segunda parcela com a primeira parcela.

Exemplo:

$$2 + 6 = 6 + 2 = 8$$

23

4.2 - OPERAÇÕES “E” (MULTIPLICAÇÃO LÓGICA)

Seguiremos o mesmo raciocínio que utilizamos para a operação lógica OU. Veremos que na multiplicação lógica o operador é a letra “E”.

A operação “E” é definida como operação que resulta “0” (zero) se pelo menos uma das variáveis de entrada tiver o valor de “0” (zero).

Podemos deduzir pela definição que o resultado da operação E será 1 (um) se, e somente se todas as entradas tiverem o valor de 1 (um).

O símbolo usualmente utilizado na operação E é “.”, porém outra notação que alguns autores utilizam é “^”. Vamos, na tabela abaixo, listar as possibilidades de combinações entre dois valores booleanos e os respectivos resultados da operação E:

$$\begin{array}{l} 0 . 0 = 0 \\ 0 . 1 = 0 \\ 1 . 0 = 0 \\ 1 . 1 = 1 \end{array}$$

24

Também como a operação **OU**, a operação **E** só pode ser definida ao menos por duas variáveis. Consequentemente, o operador “.” (E) é também binário. No intuito de mostrar o comportamento da equação $A . B$ (pode-se ler como: A e B), observe a tabela verdade abaixo:

A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

Conforme fizemos anteriormente, vamos verificar qual seria o resultado da equação $A . B . C$ (pode-se ler como: A e B e C) utilizando de forma direta a definição da operação “E” que diz: o resultado será **0** (zero) se pelo menos uma das variáveis de entrada valer **0** (zero).

A	B	C	A . B . C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

25

Você deve estar lembrado que falamos sobre as propriedades associativa e comutativa. Pois é, na operação E também valem estas propriedades. Sendo assim, a equação $A \cdot B \cdot C$ pode ser analisada tomando-se as variáveis aos pares, sem necessidade de ordem.

Observe a tabela verdade abaixo e verifique os resultados:

A	B	C	$A \cdot B \cdot C$	$A \cdot B$	$(A \cdot B) \cdot C$	$B \cdot C$	$A \cdot (B \cdot C)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	0	0	0	1	0
1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	1
1	1	0	0	1	0	0	0
1	1	1	1	1	1	1	1

26

4.3 - COMPLEMENTAÇÃO (OU NEGAÇÃO, OU INVERSÃO)

Dando continuidade ao estudo das operações booleanas, vamos agora estudar a operação chamada de Complementação.

A operação de complementação consiste em que seu resultado é simplesmente o valor complementar ao que a variável apresenta.

Em virtude desta condição, uma variável booleana pode assumir um entre somente dois valores, o valor complementar será “1” (um) se a variável valer “0” (zero) e será “0” (zero) se a variável tiver o valor de “1” (um).

Você irá observar que os símbolos utilizados para a representação da operação complementação sobre uma variável booleana, exemplo a variável “A”, pode ser: A , $\sim A$ e A' (lê-se A negado). Para nosso estudo, iremos utilizar: \overline{A} . Veja a tabela abaixo para entender o resultado da operação complementação:

$$\begin{array}{l} \overline{0} = 1 \\ \overline{1} = 0 \end{array}$$

Você deve ter notado que diferentemente das operações “OU” e “E”, a operação de complementação só pode ser definida sobre uma variável, ou sobre o resultado de uma expressão. Neste caso, o operador

da operação de complementação é chamado de: unário. Veja a tabela abaixo para compreender melhor o que falamos:

A	\bar{A}
0	1
1	0

27

5 - AVALIAÇÕES DE EXPRESSÕES BOOLEANAS

Conforme observamos uma equação qualquer, que descreve uma função booleana, entendemos que o objetivo é compreender com detalhes como esta função se comporta para qualquer combinação de entrada das variáveis. O comportamento de uma função pode ser observado em sua tabela verdade e isto é conhecido como expressão, que descreve a função considerada ou avaliação da função. Resumindo, deseja-se achar a tabela verdade para a função booleana.

Uma tabela verdade, como o próprio nome informa, é formada basicamente por um conjunto de colunas, nas quais são listadas todas as combinações possíveis entre as variáveis de entrada (à esquerda) e o resultado da função (à direita).

É possível, também, criar colunas intermediárias, onde serão listados os resultados de subexpressões contidas na expressão principal. Tudo isso tem por objetivo facilitar a avaliação, principalmente no caso de equações muito complexas, que possuem muitas variáveis.



Quando notamos na mesma equação booleana que há operações **E** e **OU**, torna-se necessário observar a **ordem de precedência**. Da mesma maneira que seguimos regras na álgebra dos reais, a multiplicação (lógica) possui precedência sobre a adição (lógica).

Seguindo o mesmo raciocínio, expressões que estão entre parêntesis têm precedência sobre operadores **E** e **OU** que estejam no mesmo patamar.

No caso da operação de **complementação**, esta deve ser avaliada tão logo seja possível. Caso a complementação seja aplicada sobre uma subexpressão inteira, é necessário que se avalie primeiramente a subexpressão para, para apenas depois, inverter o seu resultado.

28

Você pode calcular o número de combinações que as variáveis de entrada podem assumir por meio do seguinte cálculo 2^n . Nesse cálculo, o “n” representa o número de variáveis de entrada. Para criar uma tabela verdade tendo por base uma equação booleana é preciso observar 3 passos:



29

Vamos agora usar como exemplo a seguinte expressão $W = X + (Y \cdot \sim Z)$.

Nesta expressão, a variável W representa a função booleana. Essa variável depende das variáveis que estão à direita do sinal = (igual), sendo assim, irá depender de X, Y e Z. Então, serão três as variáveis de entrada.

Podemos então deduzir que o total de combinações entre 3 variáveis será $2^3=8$. Por consequência, a tabela verdade para W deverá ter 3 colunas à esquerda e 8 linhas. Seguindo o que mencionamos acima, vamos criar uma coluna, e nela listar os valores para Z. Logo depois, começamos a avaliação propriamente dita, partindo do nível mais interno de parêntesis. Em virtude de não haver parêntesis na expressão, iremos começar resolvendo as subexpressões que envolvem a operação “E”. Na fórmula que estamos utilizando há somente uma subexpressão: $X \cdot Y$. Em virtude disso, vamos criar uma coluna para $X \cdot Y$, e nela colocaremos os resultados para este produto.

Ao final, utilizam-se os resultados de $X \cdot Y$, listados na coluna anterior, para operar o **OU** com a variável X. Observe os passos descritos na tabela verdade abaixo e veja que os parêntesis em torno do produto $X \cdot Y$ indicam apenas que este termo já passou por avaliação e que no passo referente a esta coluna, tomaram-se apenas os valores previamente encontrados.

X	Y	Z	$\sim Z$	$(Y \cdot \sim Z)$	$W = X + (Y \cdot \sim Z)$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	0	0	1

RESUMO

Durante o estudo deste módulo tivemos oportunidade de estudar e entender que sistema pode ser definido como um conjunto de elementos que se relacionam e interagem com objetivo de desempenhar determinada função. Vimos que um sistema digital é formado por um conjunto de componentes interconectados que processam informações em forma digital ou discreta. Entendemos que em oposição ao que conhecemos na matemática de nosso dia a dia, também conhecida como a álgebra ordinária dos reais, as variáveis podem assumir valores no intervalo $(-\infty; +\infty)$, entretanto as variáveis booleanas só podem assumir um número finito de valores, podendo ser 0 ou 1. Ao final vimos que na álgebra booleana há três funções básicas ou operações, elas são: operação “OU”, operação “E” e Complementação. Todas as funções booleanas podem ser representadas por meio destas operações básicas.

UNIDADE 2 – FUNCIONAMENTO DOS CIRCUITOS COMBINACIONAIS E SEQUENCIAIS

MÓDULO 2 – CIRCUITOS LÓGICOS

1 - PORTAS LÓGICAS

Neste módulo iremos compreender que uma função booleana pode ser representada de forma gráfica, onde cada operador está associado a um símbolo específico (portas lógicas), tornando possível o imediato reconhecimento visual. Você terá oportunidade de entender que um circuito lógico tem por característica ser composto pelas portas lógicas relacionadas às operações realizadas sobre as variáveis de entrada. Então, os resultados das operações são conduzidos por meio de fios que representamos com a utilização de linhas simples. Faremos uma passagem pelos conceitos, leis e propriedades da álgebra booleana para podermos fundamentar nosso conhecimento. Ao final, veremos como, a partir de uma expressão booleana, podemos criar circuitos lógicos. Então vamos começar?

Durante nosso estudo sobre função booleana entendemos que ela pode ser representada por uma equação ou também pode ser detalhada pela tabela verdade.

Todavia uma função booleana também pode ter sua representação feita de forma gráfica, onde cada operador está associado a determinado símbolo específico, tornando possível o imediato reconhecimento visual. Estes símbolos são conhecidos como portas lógicas.

Na verdade, muito além do que apenas símbolos de operadores lógicos, as portas lógicas representam

recursos físicos, sendo assim, circuitos eletrônicos, com capacidade de realizar as operações lógicas. Na eletrônica trabalhamos apenas com dois estados, a qual é denominada eletrônica digital, o nível lógico 0 (zero) está associado normalmente à ausência de tensão (0 volt) enquanto o nível lógico 1 está associado à presença de tensão (de maneira geral é de 5 volts). Durante nosso estudo abordaremos alguns pontos do vasto mundo da álgebra booleana, compreendendo que as portas lógicas representam também circuitos eletrônicos que, de alguma maneira, realizam as funções booleanas simbolizadas

Consequentemente, chamaremos de **circuito lógico** ao conjunto de portas lógicas e respectivas conexões que simbolizam uma equação booleana.

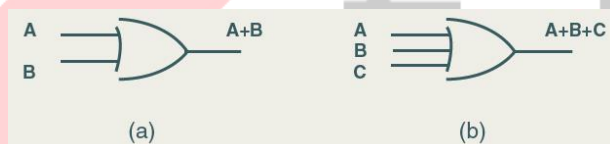
02

1.1 - Porta OU

O símbolo da porta **OU** pode ser visualizado na figura abaixo. Da mesma forma que na porta **E**, as entradas são colocadas à esquerda e a saída, à direita.

Note, ao observar a figura abaixo, que há duas e três entradas e apenas uma saída. Observe também que deverá haver no mínimo duas entradas, contudo haverá somente uma saída.

Você deve estar lembrado do operador lógico **E**, então, o funcionamento da porta **E** segue esta mesma definição operação.



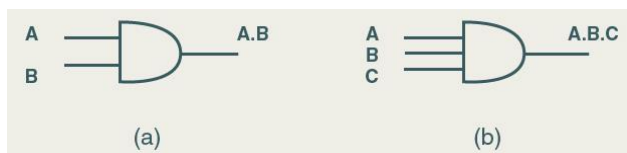
03

1.2 - PORTA E

O símbolo da porta **E** é mostrado na figura abaixo. À esquerda estão dispostas as entradas (conforme já falamos, no mínimo duas e à direita, apenas uma saída).

Podemos dizer que as linhas que conduzem as variáveis de entrada e saída simbolizam os fios que transportam os sinais elétricos associados às variáveis.

O comportamento da porta **E** segue estritamente o que estudamos anteriormente sobre a tabela verdade.



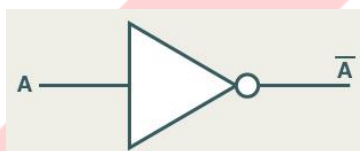
04

1.3 - INVERSOR (NEGADOR OU PORTA INVERSORA)

Relembrando o que estudamos, a porta que simboliza a operação complementação é chamada de **inversor** (negador ou porta inversora).

Em virtude de a operação complementação ser apenas possível de realização sobre uma variável por vez ou sobre o resultado de uma subexpressão, consequentemente, o inversor só possui uma entrada e, claro, apenas uma saída. Então, se houver necessidade de complementar uma expressão, o primeiro passo será obter-se o seu resultado, para apenas então aplicar a complementação.

O símbolo do inversor é mostrado na figura abaixo.



05

2 - EXEMPLO DE CIRCUITO LÓGICO

Nosso objetivo até o momento é obtermos conhecimento suficiente para que, ao observar uma equação booleana qualquer, tenhamos capacidade de desenhar a implementação do circuito lógico. Vamos lá?

Um circuito lógico tem por característica ser composto pelas portas lógicas relacionadas às operações realizadas sobre as variáveis de entrada. Então, os resultados das operações são conduzidos por meio de fios que representamos com a utilização de linhas simples.

Os passos que devemos seguir para fazermos um desenho do circuito lógico tendo por base uma equação são similares aos usados na avaliação da expressão. Vamos usar como exemplo a equação ($W = X + Y \cdot \bar{Z}$). O primeiro passo é identificar quais são as variáveis independentes, que neste exemplo são X, Y e Z. Para cada variável iremos traçar uma linha (observando que será da esquerda para a direita), estas linhas irão representar os fios que conduzem os sinais elétricos (valores). Logo depois vamos desenhar as portas necessárias para representar cada uma das subexpressões. Observe que há regras para a ordem tomada para a avaliação, que é:

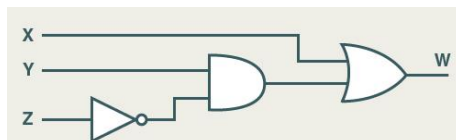
1º parêntesis (dos mais internos para os mais externos);

2º operações **E**;

3º operações **OU**.

Na figura abaixo podemos ver o desenho do circuito lógico para a equação:

$$W = X + Y \cdot \bar{Z}$$



06

3 - LEIS FUNDAMENTAIS E PROPRIEDADES DA ÁLGEBRA BOOLEANA

Você já deve estar acostumado á seguinte expressão: “para tudo há regras ou leis”. Pois é, no mundo da álgebra booleana não é diferente, as leis da álgebra booleana tratam dos valores que uma variável pode assumir (espaço booleano) e as operações elementares deste espaço. As propriedades são deduzidas tendo por base as definições das operações.

Vamos tomar como exemplo duas variáveis booleanas. Chamaremos de A e B. Vamos definir o espaço booleano tendo por base a figura abaixo:

se $A \neq 0$, então $A = 1$;
se $A \neq 1$, então $A = 0$.

As operações elementares deste espaço são operação **OU**, operação **E** e complementação.

As propriedades da álgebra booleana são as seguintes:

<p>Da adição lógica:</p> <p>(1) $A + 0 = A$ (2) $A + 1 = 1$ (3) $A + A = A$ (4) $A + \bar{A} = 1$</p> <p>Da multiplicação lógica:</p> <p>(5) $A \cdot 0 = 0$ (6) $A \cdot 1 = A$ (7) $A \cdot A = A$ (8) $A \cdot \bar{A} = 0$</p> <p>Da complementação:</p> <p>(9) $\bar{\bar{A}} = A$</p>	<p>Comutatividade:</p> <p>(10) $A + B = B + A$ (11) $A \cdot B = B \cdot A$</p> <p>Associatividade:</p> <p>(12) $A + (B + C) = (A + B) + C = (A + C) + B$ (13) $A \cdot (B \cdot C) = (A \cdot B) \cdot C = (A \cdot C) \cdot B$</p> <p>Distributiva (da multiplicação em relação à adição):</p> <p>(14) $A \cdot (B + C) = A \cdot B + A \cdot C$</p>
---	--

3.1 - TEOREMAS DE DE MORGAN

Veremos agora dois teoremas do lógico e matemático britânico Augustus De Morgan que viveu no século XIX. Pedimos que você não fique assustado com as proposições dos teoremas, pois aparentemente eles são complexos, mas são base para o entendimento dos circuitos lógicos.

Veremos que Augustus De Morgan diz em seu primeiro teorema que a complementação de um produto (lógico) equivale à soma (lógica) das negações de cada variável do referido produto. Tendo por base a Figura 01 vista anteriormente, observe como seria em forma de equação:

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

No segundo teorema de De Morgan observamos que é chamado de dual (espelho ou contrário) do primeiro, sendo assim, ele diz que a complementação de uma soma (lógica) equivale ao produto das negações individuais das variáveis. Tendo por base a Figura 02 vista anteriormente, veja como seria em forma de equação:

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

Particularizando os teoremas de De Morgan para duas variáveis. Tendo por base a Figura 03 e 04 vistas anteriormente, veja como seria em forma de equação:

$$\begin{aligned}\overline{A \cdot B} &= \overline{A} + \overline{B} \\ \overline{A + B} &= \overline{A} \cdot \overline{B}\end{aligned}$$

3.2 - DERIVAÇÃO DE EXPRESSÕES BOOLEANAS

Tendo por base uma função booleana, conforme sua tabela verdade, derivar uma expressão booleana para esta função é encontrar descrição por meio de uma equação. Então, entendemos que a derivação de expressões booleanas é o problema inverso da avaliação de uma expressão booleana, conforme vimos anteriormente.

Existem basicamente duas maneiras para descrever uma função booleana:



Toda e qualquer função booleana pode ser descrita tendo por base a soma de produtos ou por meio de produto de somas. Em virtude das funções booleanas só poderem assumir dois valores (0 ou 1), basta utilizar um dos dois métodos para encontrarmos uma equação para uma função.

09

3.3 - DERIVAÇÃO DE EXPRESSÕES USANDO SOMA DE PRODUTOS (SDP)

Tendo por base uma função booleana que possui “n” variáveis ou “n” entradas, consequentemente poderá haver 2^n (dois elevado a n) possibilidades de combinações de valores. Chamamos essas possibilidades de valores que as variáveis podem assumir, juntamente com os respectivos valores da função, de “espaço da função”. A cada combinação de entradas podemos associar um determinado termo produto, no qual todas as variáveis da função estarão presentes, e que é composto da seguinte maneira:

- caso a variável correspondente tiver o valor 0 (zero), ela irá aparecer “negada”;
- se a variável tiver o valor 1 (um), ela deve aparecer como “não negada”.

Veja a tabela abaixo, na qual é possível observar os termos de produto associados a cada combinação de entradas para uma função booleana de três variáveis (A, B, C).

A	B	C	mintermos
0	0	0	$\bar{A} \cdot \bar{B} \cdot \bar{C}$
0	0	1	$\bar{A} \cdot \bar{B} \cdot C$
0	1	0	$\bar{A} \cdot B \cdot \bar{C}$
0	1	1	$\bar{A} \cdot B \cdot C$
1	0	0	$A \cdot \bar{B} \cdot \bar{C}$
1	0	1	$A \cdot \bar{B} \cdot C$
1	1	0	$A \cdot B \cdot \bar{C}$
1	1	1	$A \cdot B \cdot C$

Cada termo produto construído conforme a regra falada anteriormente é chamado de mintermo (também pode ser chamado de minitermo).

Observe que, para determinado dado mintermo, caso substituirmos os valores das variáveis associadas, obteremos o valor 1 (um). Entretanto, se substituirmos nesse mesmo mintermo por qualquer outra combinação de valores, iremos obter o valor 0 (zero).

Dessa maneira, caso queiramos encontrar a equação para uma função tendo por base sua tabela verdade, basta utilizar um “OU” entre os mintermos associados aos números um da função (conhecidos como: mintermos 1).

10

Tendo por base a tabela verdade abaixo, vamos encontrar a equação em **soma de produtos** (SdP) para a função:

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

Temos que F é uma função das variáveis A, B e C. Observamos que os valores para A, B e C quando F=1 são (0,1,0), (0,1,1), (1,0,1) e (1,1,0). Agora os “mintermos” (mintermos 1) associados a essas condições são respectivamente:

$$\overline{A} \cdot B \cdot \overline{C}, \overline{A} \cdot B \cdot C, A \cdot \overline{B} \cdot C \text{ e } A \cdot B \cdot \overline{C}$$

Então, a equação em soma de produtos para F será o **OU** entre estes produtos. Observe a figura abaixo:

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

Tendo por objetivo simplificar a notação, o símbolo da operação **E** poderá ficar oculto. Desta maneira, a equação vista anteriormente ficaria mais enxuta e ser escrita da seguinte forma:

$$F = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + AB\overline{C}$$

11

3.4 - DERIVAÇÃO DE EXPRESSÕES USANDO PRODUTO DE SOMAS (PdS)

Agora vamos ver que o método de derivação usando produto de somas é o dual (oposto) do método de derivação em soma de produtos. Sendo que em cada combinação das variáveis de entrada de uma função é possível fazer associação a um termo soma, no qual todas as variáveis da função estão presentes e que é formado da seguinte maneira:

- caso uma variável correspondente tenha o valor de 1 (um), ela deve aparecer como: “negada”;
- caso a variável tenha o valor 0 (zero), ela deve aparecer como: “não negada”.

A tabela abaixo faz uma lista dos termos soma associados a cada combinação de entradas para uma função booleana quando utilizamos 3 variáveis (A, B, C).

A	B	C	maxtermos
0	0	0	$A \cdot B \cdot C$
0	0	1	$A \cdot B \cdot \bar{C}$
0	1	0	$A \cdot \bar{B} \cdot C$
0	1	1	$A \cdot \bar{B} \cdot \bar{C}$
1	0	0	$\bar{A} \cdot B \cdot C$
1	0	1	$\bar{A} \cdot B \cdot \bar{C}$
1	1	0	$\bar{A} \cdot \bar{B} \cdot C$
1	1	1	$\bar{A} \cdot \bar{B} \cdot \bar{C}$

Cada termo soma efetuado tendo por base a regra anteriormente falada é chamado de: maxtermo(maxitermo).

Observe que, para um dado maxtermo, caso façamos a subtração dos valores das variáveis associadas, iremos obter o valor “0” (zero). Todavia, caso façamos a substituição nesse mesmo maxtermo com qualquer tipo de combinação de valores, iremos obter o valor 1 (um).

Dessa maneira, caso queiramos encontrar a equação para uma função tendo por base a respectiva tabela verdade, basta utilizar um “E” entre os maxtermos associados aos valores zero da função que também podemos chamar de: maxtermos 0 (zero).

Por meio da tabela verdade abaixo iremos encontrar a equação em produto de somas (PdS) para a função F:

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

Vamos utilizar a mesma função do exemplo anterior, para que possamos fazer comparações entre os dois métodos de derivação. Os valores das variáveis de entrada A,B,C para os quais F=0 são (0,0,0), (0,0,1), (1,0,0) e (1,1,1). Os maxtermos (maxtermos zero) associados a essas condições são respectivamente:

12

$$A + B + C, A + B + \overline{C}, \overline{A} + B + C \text{ e } \overline{A} + \overline{B} + \overline{C}$$

Por consequência, a equação em produto de somas para F será o “E” entre estas somas:

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



Observe que há uma ordem de precedência, na expressão acima segue a ordem de que em produto de somas é “primeiro cada soma deve ser avaliada, para só então avaliar-se o produto”. Sendo assim, devemos entender que é obrigatório os parêntesis em torno de cada termo. Preste atenção também que os símbolos referentes à operação “E”, entre os termos soma, podem ser ocultos.

13

3.5 - FORMAS CANÔNICAS, PADRÃO E NÃO-PADRÃO

Você deve ter observado que as representações em soma de produtos e em produto de somas seguem um padrão, por isso elas são chamadas de: **formas padrão**.

A soma de produtos e o produto de somas que estudamos nos parágrafos anteriores apresentam ainda uma característica peculiar: **todas as variáveis da função estão presentes em cada termo soma e em cada termo produto**. Em virtude desta característica, essas formas são chamadas de: **canônicas**.

Muito mais do que foi visto nas representações mostradas nos parágrafos anteriores, há representações alternativas e também mais enxutas para as expressões canônicas. Caso façamos associação para cada combinação das variáveis de entrada e ao seu equivalente em decimal, cada mintermo pode ser representado por m_i , onde i é o decimal associado. De forma parecida, cada maxtermo pode ser representado por M_i , onde i é o decimal associado. Na tabela abaixo temos uma listagem de todos os mintermos e maxtermos tendo por base uma função com três variáveis: A, B, C).

A	B	C	mintermo	maxtermo
0	0	0	m_0	M_0
0	0	1	m_1	M_1
0	1	0	m_2	M_2
0	1	1	m_3	M_3
1	0	0	m_4	M_4
1	0	1	m_5	M_5
1	1	0	m_6	M_6
1	1	1	m_7	M_7

14

Retornando à função F que utilizamos como exemplo nos parágrafos anteriores, podemos refazer a expressão em soma de produtos, na forma canônica, como o seguinte:

$$F = m_2 + m_3 + m_5 + m_6$$

Podemos deixar esta expressão mais enxuta usando o símbolo de somatório (Σ); vamos lá:

$$F = \Sigma(2,3,5,6)$$

Uma expressão em produto de somas, na forma canônica, podemos reescrever da seguinte forma:

$$F = M_0 \cdot M_1 \cdot M_4 \cdot M_7$$

Podemos, também, deixar esta expressão mais enxuta utilizando o símbolo de produto (\prod); vamos lá:

$$F = \prod(0,1,4,7)$$

Você deve ter notado que mesmo as representações canônicas sendo práticas, elas são pouco úteis quando o assunto é a implementação de circuitos digitais. O número de portas lógicas e conexões de um circuito lógico depende diretamente do número de operações booleanas (E, inversão, OU) contidas na expressão associada. Sendo assim, é mais prático a redução do número de operações contidas numa função, de tal maneira poder implementar utilizando circuitos lógicos mais simples, e por consequência, de custo mais reduzido.

A diminuição da quantidade de operações é alcançada por meio da eliminação de literais da expressão, utilizando as propriedades da álgebra booleana descritas nos parágrafos anteriores. Podemos dizer que um “literal” pode ser uma “variável negada” ou uma “variável não negada”. Chamamos de **simplificação** o processo de redução de literais ou de redução de operações, equivalentemente.

15

Vamos agora usar exemplos para entendermos os passos básicos para a simplificação algébrica (literal) de expressões booleanas. Vamos utilizar uma expressão canônica, em soma de produtos, para uma dada função F:

$$F = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + ABC\overline{C}$$

A primeira observação que precisamos fazer é a identificação dos pares de “mintermos” que se diferenciam por apenas um literal, a fim de aplicar a propriedade (14). Os mintermos $A\overline{C}B$ e ABC , por exemplo, possuem os mesmos literais, exceto pela variável C: no primeiro, o literal é \overline{C} , enquanto no segundo, o literal é C. Então, conforme a expressão abaixo que exemplifica que a propriedade Distributiva diz “da multiplicação em relação à adição”:

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

Tendo por base esta propriedade podemos fatorar esses dois mintermos, tendo como resultado a expressão abaixo:

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

Então pela propriedade Distributiva temos que $C + \overline{C} = 1$. Então, substituindo na expressão acima, obtemos:

$$F = \overline{A}B \cdot 1 + A\overline{B}C + ABC\overline{C}$$

Pela propriedade da multiplicação lógica onde $A \cdot 1 = A$ podemos substituir e obtemos:

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

Então, por meio da manipulação algébrica, iremos obter uma expressão em soma de produtos mais enxuta em comparação à mesma expressão em soma de produtos na forma canônica, em virtude da redução do número de operações e também de literais.

Os símbolos utilizados para representar a operação complementação sobre uma variável Booleana A é o “-” na parte superior da letra, exemplo: \bar{A} .

Observe a equação:

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + ABC$$

Diferente do que é possível ver na equação acima, poderá haver situações que um mintermo ser $\bar{A}CB$. Neste caso ele poderia ser agrupado com o mintermo ABC, pois os dois possuem os mesmos literais, exceto pela variável A (A no primeiro e A no segundo). É muito natural que os passos que foram seguidos seriam os mesmos falados anteriormente. Sendo assim, a equação resultante seria um pouco diferente, mas com a mesma quantidade de operações, sendo, por conseguinte, da mesma complexidade. Realmente, a melhor maneira seria se fosse possível agrupar o mintermo A C B com o mintermo ABC e ao mesmo tempo com o mintermo $\bar{A}CB$. Por sorte, a propriedade “Da adição lógica” da álgebra booleana diz que ($A + A = A$); então o “OU” entre duas ou mais variáveis booleanas iguais é igual a própria variável booleana. Saiba +

Fazendo uma ampliação desta propriedade, podemos falar que o “OU” entre duas ou mais funções (também os produtos) booleanas iguais é equivalente à própria função booleana. Desta maneira, podemos expandir o mintermo $\bar{A}CB$ para:

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

Esta manipulação algébrica é decorrente da propriedade “adição lógica” da álgebra booleana que afirma que ($A + A = A$).

Saiba +

Os símbolos utilizados para representar a operação de complementação sobre uma variável Booleana é o “-” na parte superior da letra, por exemplo: E. Na explicação desta equação é possível entender que ABC e ACB são diferentes, pois possuem os mesmos literais que é a letra “A” (um “literal” pode ser uma “variável negada” ou uma “variável não negada”). O traço na parte superior é o indicativo.

16

Utilizando novamente a expressão:

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

Também utilizando a expressão:

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

Observando as duas expressões acima e por meio da propriedade “adição lógica” da álgebra booleana que afirma que $(A + A = A)$, Consequentemente, notamos que as expressões acima são equivalentes, embora o mintermo $\overline{A}B\overline{C}$ apareça duplicado. E pelo fato de aparecer duas vezes, pode-se usar uma cópia de $\overline{A}B\overline{C}$ para simplificar com $A\overline{B}\overline{C}$ e outra para simplificar com $\overline{A}B\overline{C}$.

Para obter a simplificação vamos utilizar a propriedade Distributiva (da multiplicação em relação à adição) onde “ $A \cdot (B + C) = A \cdot B + A \cdot C$ ” então:

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

Vamos simplificar utilizando a propriedade da multiplicação lógica onde “ $A \cdot 1 = A$ ”

$$F = \overline{A}B \cdot 1 + A\overline{B}\overline{C} + 1 \cdot B\overline{C}$$

Agora ao final vamos utilizar a propriedade da “adição lógica” da álgebra booleana que afirma que $(A + A = A)$:

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

Observe que o mintermo $\overline{A}B\overline{C}$ não pôde ser agrupado com outro mintermo. Veja também que utilizamos das simplificações possíveis, uma vez que foram agrupados e simplificados todos os pares de mintermos que se diferenciam de somente uma variável. Consequentemente, a última expressão acima é uma mostra da máxima simplificação possível sob a forma de soma de produtos. É em virtude disso que esta expressão é chamada de equação mínima em soma de produtos da função **F**.

Ao voltarmos a falar sobre a expressão que foi simplificada:

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

17

Dizemos que esta equação foi apenas simplificada, isto difere e ao mesmo tempo evidencia que toda equação mínima é simplificada, contudo, nem toda equação que foi simplificada é necessariamente mínima.

18

4. CIRCUITOS LÓGICOS PARA FORMAS PADRÃO E NÃO-PADRÃO

Durante o desenvolvimento do estudo deste módulo, vimos as regras gerais para se realizar o desenho de circuitos lógicos. Neste parágrafo iremos ver as seguintes regras que devem ser observadas, com o intuito de facilitar a compreensão do desenho:

- variáveis de entrada precisam ser identificadas de modo preferencial à esquerda, ao lado aos respectivos fios;
- na equação os inversores devem ser inseridos para as variáveis que aparecem negadas;
- conforme a ordem de avaliação os operadores as portas que implementam as operações booleanas devem aparecer na equação posicionados da esquerda para a direita.

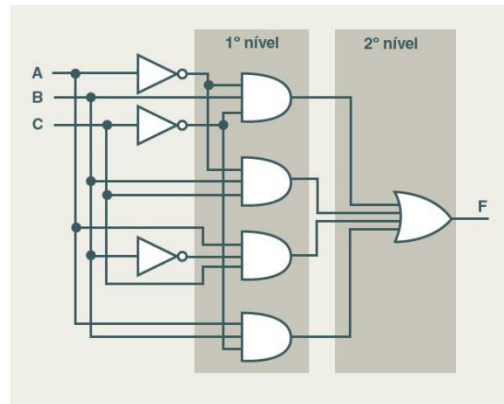
No caso de equações canônica ou simplificada, ou chamadas de soma de produtos, há um primeiro nível (desprezando possíveis inversores), provido somente por portas “E”, sendo cada porta “E” uma implementação dos produtos da equação. Existe ainda um segundo nível, composto por uma porta “OU”, responsável pela “soma” lógica dos produtos.

19

Tendo por base a equação abaixo, vamos criar um possível circuito lógico.

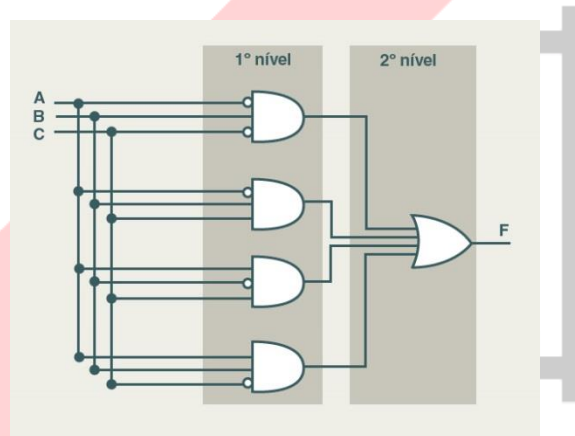
$$F = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + ABC$$

Vamos agora ver como seria o circuito lógico tendo por base a equação acima, veja que em todas as interseções de fios em que há ligação física, deve também haver um ponto (com tamanho razoável), como se fosse uma “solda”. Então, quando não existe o referido ponto na interseção de fios, significa que estes fios estão “eletricamente isolados”. Vamos ver:



Circuito Lógico 01

O circuito pode também ser desenhado por meio de uma notação simplificada para os inversores das entradas. Para isso, ao invés de fazer um desenho de um inversor para cada variável que aparece negada na equação, é colocado um círculo junto a cada entrada de cada porta na qual há uma variável negada. Podemos ver que a aplicação desse procedimento para o circuito da figura acima terá como resultado o seguinte desenho:



Circuito Lógico 02

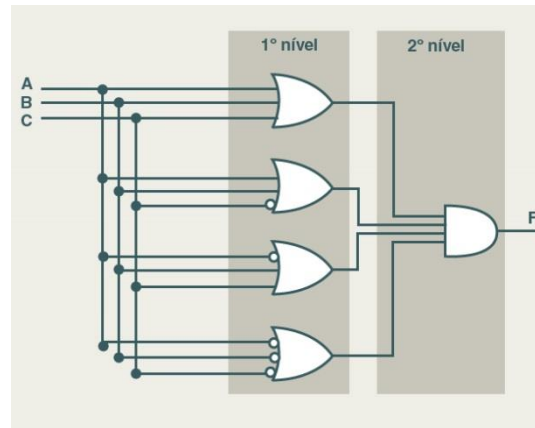
No caso das equações que possuem a forma produto de somas, também conhecida como canônica ou simplificada, vemos que o primeiro nível será constituído por portas **OU**, sendo que cada porta responsável por uma das “somas” lógicas da equação. Continuando e chegando ao segundo nível, vemos que é constituído por uma porta **E**, que irá realizar o produto lógico das parcelas.

20

Tomemos por base a expressão lógica abaixo:

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

Esta equação irá gerar o seguinte circuito lógico:



Circuito Lógico 03

Em virtude da constatação de apresentarem apenas dois níveis de portas ou dois níveis lógicos, circuitos para equações que são representadas nas formas padrão, canônicas ou simplificadas, são chamados de: **circuitos em dois níveis** ou **lógica a dois níveis**.



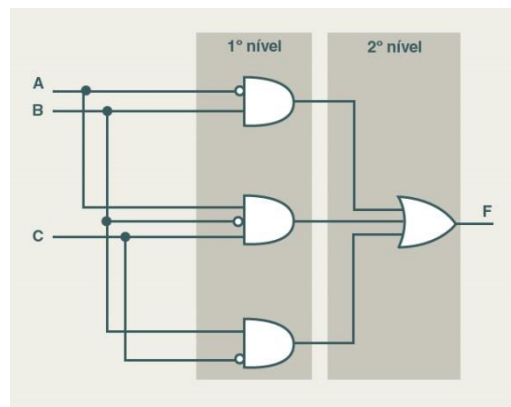
Havendo uma equação canônica de uma função qualquer, o circuito para uma equação simplificada tendo por base uma equação canônica possui uma quantidade menor de portas e/ou portas com menor complexidade. É preciso entender que a complexidade relativa de uma porta não é medida pelo número de entradas que ela apresenta.

21

Vamos observar a seguinte equação, que é a fórmula mínima para o circuito lógico abaixo:

$$F = \bar{A}B + A\bar{B}C + B\bar{C}$$

Esta equação servirá para criar o seguinte circuito lógico:



Circuito Lógico 04

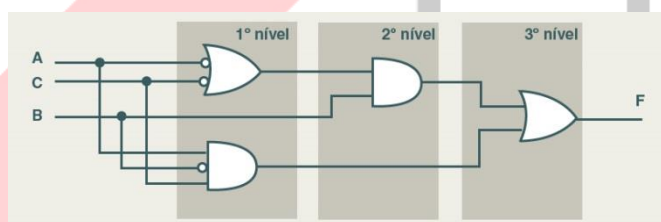
Observe que este circuito é de menor complexidade que os circuitos que vimos anteriormente. A complexidade relativa de um circuito lógico é calculada ao somarmos o número de entradas das portas.

Nos circuitos que vimos anteriormente notamos a presença de quatro portas de três entradas e uma porta de quatro entradas. Consequentemente, a complexidade relativa será: $4 \times 3 + 1 \times 4 = 16$. No circuito da figura acima visualizamos duas portas de duas entradas e duas portas de três entradas. Então podemos dizer que a complexidade relativa será $2 \times 2 + 2 \times 3 = 10$. Fica evidente que o circuito da figura acima possui menor complexidade em relação aos circuitos vistos anteriormente.

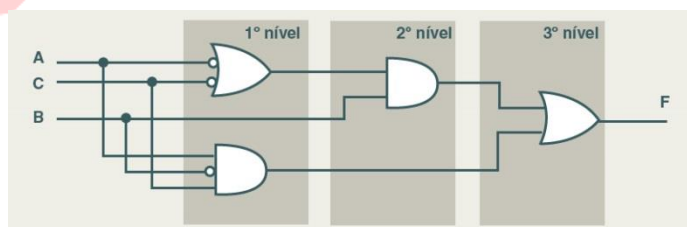
22

Os circuitos para formas fatoradas são vistos como o caso mais generalista. Geralmente, as formas fatoradas nos levam a circuitos em que o número de níveis lógicos é maior do que 2. Em virtude disso, os circuitos lógicos para formas fatoradas são chamados de **circuitos multinível** ou também lógica multinível.

Estudamos, anteriormente, que às vezes uma forma fatorada apresenta um menor número de operações em relação à respectiva forma padrão. Quando isso ocorre, o circuito associado à forma fatorada também terá menor complexidade relativa. Contudo, caso não haja redução no número de operações, mesmo assim é possível que o circuito para a forma fatorada tenha menor complexidade relativa, pois o conceito de complexidade relativa inclui também o número de entradas de cada porta. Sendo assim, a forma mais segura e objetiva de saber se o circuito associado à forma fatorada possui menor complexidade ou não é fazer um desenho e somar o número de entradas.



Circuito Lógico 05



Circuito Lógico 06

A figura acima mostra o circuito para a seguinte equação:

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

Observe que o número de operações booleanas desta equação é quatro. Entretanto, a complexidade do circuito da forma fatorada é $3 \times 2 + 1 \times 3 = 9$, conseqüentemente é menor do que a complexidade do circuito 2.5.

23

RESUMO

Neste módulo tivemos a oportunidade de verificar a importância das portas lógicas e sua utilidade. Conhecemos as Leis Fundamentais e Propriedades da Álgebra booleana e com este conhecimento entendemos a importância da teoria para a fundamentação do conhecimento. Tivemos a oportunidade de, por meio de equações booleanas, criarmos um Circuito Lógico e vermos a complexidade da interação dos componentes de um sistema computacional.

UNIDADE 2 – FUNCIONAMENTO DOS CIRCUITOS COMBINACIONAIS E SEQUENCIAIS

MÓDULO 3 – CIRCUITO COMBINACIONAL

01

1- TIPOS DE CIRCUITOS

Neste módulo teremos a oportunidade de estudar o que são os circuitos sequencial e combinacional. Você compreenderá que a partir de um circuito combinacional podemos criar uma tabela de condição e notação. Ao final falaremos sobre circuitos combinacionais de interconexão que têm a responsabilidade pelas operações lógicas e aritméticas em um sistema digital (computador).

Como já falamos nos módulos anteriores, nos sistemas digitais, os circuitos lógicos podem ser de dois tipos:

- circuitos sequenciais e
- circuitos combinacionais.

Os componentes de um circuito sequencial são:

- circuito combinacional e
- elementos de memória.

Estão conectadas somente ao circuito combinacional as entradas e as saídas do circuito sequencial.

Os elementos de memória são circuitos com capacidade de armazenamento de informação codificada em binário.

Interessante observar que algumas das saídas do circuito combinacional são entradas para os elementos de memória, recebendo o nome de: variáveis do próximo estado.

Consequentemente, as saídas dos elementos de memória constituem parte das entradas para o circuito combinacional e recebem o nome de: variáveis do estado atual.

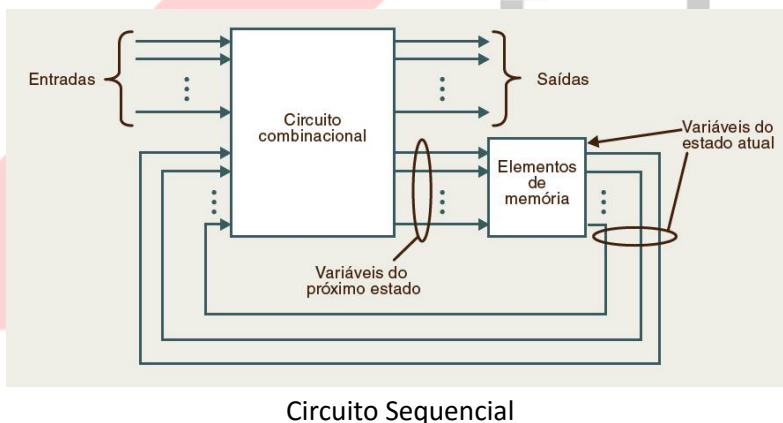
As ligações ou conexões entre o circuito combinacional e os elementos de memória são geralmente chamados de laço de realimentação, em virtude da saída de um bloco ser a entrada para o outro e vice-versa.

02

A informação guardada nos elementos de memória em um determinado instante evidencia em que estado está o circuito sequencial. O circuito sequencial recebe informação de forma binária das entradas que, juntamente com a informação do estado atual, vão determinar os valores das saídas e os valores do próximo estado. Sendo assim, fica explícito que as saídas de um circuito sequencial não dependem apenas das entradas, dependem também do estado atual, guardado nos elementos de memória.

Podemos dizer o mesmo sobre as variáveis de próximo estado. Em função deste comportamento sequencial, um circuito sequencial é especificado pela sequência temporária de entradas, saídas e estados internos.

Observe a figura abaixo que mostra um circuito sequencial.

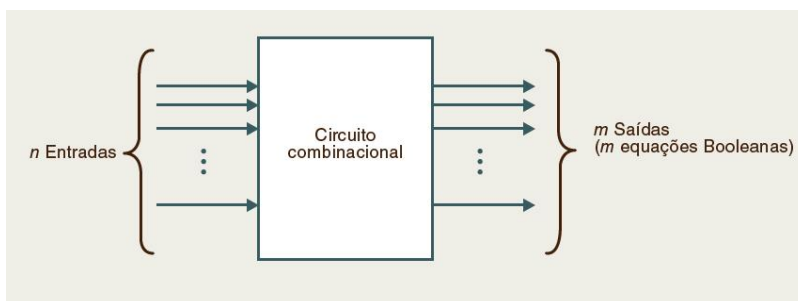


03

Um **circuito combinacional** constitui-se por um conjunto de portas lógicas, as quais determinam os valores das saídas diretamente tendo por base os valores atuais das entradas.

Podemos dizer que no circuito combinacional é realizada uma operação de **processamento da informação**, que poderá ser detalhada utilizando-se um conjunto de equações booleanas (0 ou 1 / V ou F / Sim ou Não / Y ou N). No caso, cada maneira de combinar os valores de entrada poderá ser vista como um resultado diferente e cada grupo de valores de saída representam o resultado da operação.

Podemos observar, na figura abaixo, um diagrama de blocos (ilustrativo) de um circuito combinacional.



04

Para fazermos uma **análise de um circuito combinacional** é preciso entendê-lo e determinar qual será o seu comportamento. Consequentemente, tendo por base um diagrama qualquer de um determinado circuito, procura-se encontrar equações que podem descrever suas saídas. Conseguindo encontrá-las (equações), poderemos conseguir uma “tabela verdade”, caso venha ter necessidade. Será de suma importância checar se o circuito é combinacional e não sequencial.

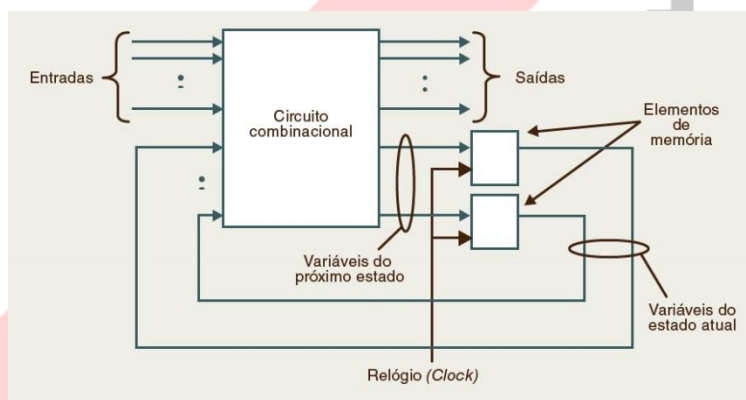


Diagrama genérico de um circuito sequencial segundo o modelo de Mealy

05

Uma maneira eficiente para conseguir determinar as equações que descrevem as saídas de um circuito combinacional será possível observar abaixo:

1. Nomear as variáveis associadas a cada saída de cada porta do circuito, exceto aquelas saídas que já possuem um nome (exemplo: as saídas do circuito);
2. Tendo como orientação a esquerda, e seguindo uma ordem de precedência determinada pelas ligações, escolher as equações associadas a cada variável, até que as equações de todas as saídas tenham sido encontradas.

Conseguindo determinar as equações das saídas, a montagem da tabela verdade será direta, havendo uma coluna para cada saída.

Vamos reforçar o conceito do que é um circuito combinacional?

Então podemos dizer que um circuito combinacional é todo circuito cuja saída tem dependência direta das várias combinações das variáveis de entrada.

Por meio do estudo desses circuitos, iremos conseguir entender o funcionamento de circuitos somadores, somadores completos, subtratores, codificadores, decodificadores, circuitos que executam prioridades, dentre outros circuitos que são utilizados na arquitetura (construção) de computadores ou sistemas digitais.

Para construir um circuito, como já falamos anteriormente, é necessário conhecer sua expressão característica.

Uma maneira eficiente de construir a expressão de um problema consiste em construir a tabela verdade para cada situação do problema para, em seguida, obter a expressão.

Observe a figura abaixo e veja uma sequência para construção de uma expressão.

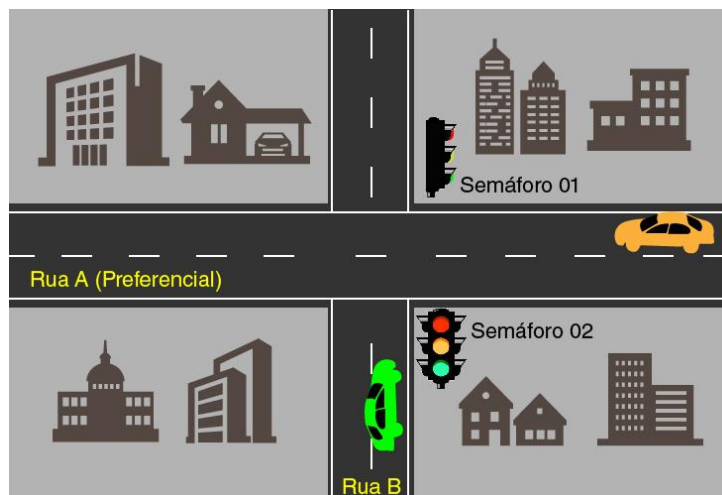


Variável

Em um programa de computador, uma variável é um objeto (uma posição, geralmente localizada na memória) capaz de armazenar e representar um determinado valor ou uma expressão.

06

O que você acha de fazermos um exemplo utilizando duas variáveis para melhor compreender como funciona um circuito? Então vamos observar a figura abaixo para iniciarmos:



Iremos utilizar a figura acima, que representa um cruzamento de ruas em uma cidade, para melhor compreender o funcionamento de um circuito combinacional. Nesta cidade, há a necessidade de instalação de semáforos automatizados para controle do fluxo dos carros nas ruas A e B. Na figura acima vemos a representação do cruzamento das ruas A e B, cada rua com o seu respectivo semáforo. A rua “B” tem o semáforo 2 e a rua A, que é preferencial, tem o semáforo 1. As seguintes características devem ser observadas para automatização dos semáforos:

- 1- Quando houver carros trafegando somente na rua B, o semáforo 2 deverá permanecer **“verde”** para os carros trafegarem livremente.
- 2- Da mesma maneira, quando houver carros trafegando somente na rua A, o semáforo 1 deverá permanecer **“verde”**.
- 3- Quando houver carros trafegando em ambas as ruas, o semáforo da rua A deve ficar **“verde”**, pois é a rua preferencial.

07

2- TABELA CONDIÇÃO/NOTAÇÃO


Tendo por base as características apresentadas para automatizar os semáforos, então iremos criar um circuito lógico para solucionar esta situação, mas antes de mais nada, iremos criar uma expressão. Contudo, vamos visualizar a figura abaixo (Tabela Condição/Notação) para entendermos as condições e então estabelecermos uma notação.

Condição	Notação
Existência de carro na rua A	$A = 1$
Não existência de carro na rua A	$A = 0$ (ou $\bar{A} = 1$)
Existência de carro na rua B	$B = 1$
Não existência de carro na rua B	$B = 0$ (ou $\bar{B} = 1$)
Verde do sinal 1 aceso	$G1 = 1$
Verde do sinal 2 aceso	$G2 = 1$
Se $G1 = 1$ então Vermelho do sinal 1 apagado Verde do sinal 2 apagado Vermelho do sinal 2 aceso	$R1 = 0$ $G2 = 0$ $R2 = 1$
Se $G2 = 1$ então Vermelho do sinal 1 aceso Verde do sinal 1 apagado Vermelho do sinal 2 apagado	$R1 = 1$ $G1 = 0$ $R2 = 0$

Observe na figura anterior (Tabela Condição/Notação), na qual para cada condição há uma notação. Veja que a lógica booleana está sendo usada. Exemplo: existência de carro na rua A, então a variável A receberá o valor 1, $A = 1$. Inexistindo carro na rua A, então a variável A receberá $= 0$. Relembrando: 0 e 1, podemos utilizar o numeral “1” para indicar que está ligado, e numeral “0” para indicar que está desligado.


08

Tendo por referência o que estudamos até agora, vamos montar uma tabela verdade com cada situação, podemos assim analisar cada situação. Veja na figura abaixo que começamos com a situação 0 (zero). Vamos lá.



Situação	A	B	G1	R1	G2	R2
0	0	0	Ø	Ø	Ø	Ø
1	0	1				
2	1	0				
3	1	1				

Observando a figura acima, vamos iniciar com a situação 0. Conforme falamos no início, esta situação representa que não há veículos nas ruas ($A=0$ e $B=0$). Desta forma, não é importante qual sinal permaneça aceso. Nestas situações, também chamadas de irrelevantes, utiliza-se o símbolo Ø para indicar que as variáveis podem assumir tanto o valor “0” ou o valor “1”.



Situação	A	B	G1	R1	G2	R2
0	0	0	Ø	Ø	Ø	Ø
1	0	1			1	
2	1	0				
3	1	1				

Observando a situação 1, notamos que a variável B recebeu o valor 1 (B=1). Esta situação representa a presença de veículos na rua B e que não há veículos na Rua A. Então, conforme a regra estabelecida, é preciso acender o sinal **verde** do semáforo 2 para a rua B. Note, também, que R1 está ligado (1) indicando que o sinal **vermelho** do semáforo 2 (rua A) está aceso (ligado) na para quem trafega na Rua A. Lembre-se de também observar a figura (Tabela Condição/Notação) para relembrar a dinâmica do tráfego.

09

Veja a situação 2.

Situação	A	B	G1	R1	G2	R2
0	0	0	Ø	Ø	Ø	Ø
1	0	1	0	1	1	0
2	1	0				
3	1	1				

Se G2 = 1 então
Vermelho do sinal 1 aceso
Verde do sinal 1 apagado
Vermelho do sinal 2 apagado

R1 = 1
G1 = 0
R2 = 0

Note a representação da presença de veículos na rua A (A=1) e a inexistência de veículos na Rua B (B=0). Então, torna-se necessário acender o sinal verde na rua A e o sinal vermelho no semáforo 2 (Rua B).

Situação	A	B	G1	R1	G2	R2
0	0	0	Ø	Ø	Ø	Ø
1	0	1	0	1	1	0
2	1	0	1	0	0	1
3	1	1				

Se G1 = 1 então
Vermelho do sinal 1 apagado
Verde do sinal 2 apagado
Vermelho do sinal 2 aceso

R1 = 0
G2 = 0
R2 = 1

Observando a situação 3, verificamos a representação da presença de veículos em ambas as ruas. Nesta situação, o sinal verde para a rua A deve permanecer aceso, pois ela é preferencial. Consequentemente, na rua B o semáforo estará vermelho (sinal vermelho).

10


Observe a situação 0.

Situação	A	B	G1	R1	G2	R2
0	0	0	Ø	Ø	Ø	Ø
1	0	1	0	1	1	0
2	1	0	1	0	0	1
3	1	1	1	0	0	1

Se G1 = 1 então
Vermelho do sinal 1 apagado
Verde do sinal 2 apagado
Vermelho do sinal 2 aceso

R1 = 0
G2 = 0
R2 = 1

Há saídas que convencionamos chamar de “irrelevantes”, uma vez que não importa qual sinal permanece aceso, então, pode-se omitir o que estiver com o símbolo Ø, e deixar a tabela apenas com os 0 e 1. Então, é possível entender o porquê do sinal 2 permanecer aceso no verde. Veja a tabela verdade abaixo com os valores preenchidos.



Situação	A	B	G1	R1	G2	R2
0	0	0			1	
1	0	1	0	1	1	0
2	1	0	1	0	0	1
3	1	1	1	0	0	1

Durante a construção das tabelas verdade, tomemos por base a situação 0, com saídas irrelevantes, em que é indiferente qual sinal permanece aceso. Consequentemente, é possível adotar que o verde do sinal 2 (semáforo 2 da rua B) permaneça aceso.

Então, vamos escrever uma tabela verdade com novos valores preenchidos para a situação 0, lembrando que cada saída, G1, R1, G2, R2 terá um circuito independente. Veja, a seguir, como fica.

11

Iniciando pela escrita da expressão de G1, em quais situações G1 acende?

Situação	A	B	G1	R1	G2	R2
0	0	0	0	1	1	0
1	0	1	0	1	1	0
2	1	0	1	0	0	1
3	1	1	1	0	0	1

Se G2 = 1 então
 Vermelho do sinal 1 aceso
 Verde do sinal 1 apagado
 Vermelho do sinal 2 apagado

R1 = 1
 G1 = 0
 R2 = 0

Antes de iniciarmos, é importante consultar a tabela verdade.

Iniciando pela escrita da expressão de G1, em quais situações G1 acende?

Resposta: nas situações 2 OU 3

Veja resolução da situação 2

Veja resolução da situação 3

Como se tem G1=1 na Situação 2 OU (OR) Situação 3, uma porta OU contendo as expressões tanto da situação 2 quanto da situação 3 resultará no valor 1 nesses casos, que representa a situação referente ao verde aceso do semáforo 1, consequentemente a expressão seria a seguinte:

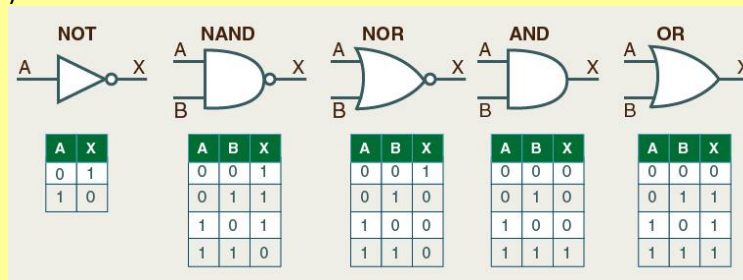
$$G1 = A.\overline{B} + A.B$$

Situação	A	B	G1	R1	G2	R2
0	0	0	0	1	1	0
1	0	1	0	1	1	0
2	1	0	1	0	0	1
3	1	1	1	0	0	1

Tabela verdade

Lembre-se de que, em qualquer bloco (porta ou função) lógico somente dois estados (0 ou 1) são permitidos em suas entradas e saídas. Vale a pena rever aqui algumas formas de representação de blocos:

E (AND)
 OU (OR)
 NÃO (NOT)
 NÃO E (NAND)
 NÃO OU (NOR)
 OU EXCLUSIVO (XOR)



Resolução da situação 2

Na situação 2 o preenchimento da equação é:

Quando $G1=1$ quando $A = 1$ e $B = 0$, ou seja, $A = 1$ E $\bar{B} = 1$

Usando uma porta E, é possível escrever $G1=1$ quando $A \cdot \bar{B}=1$

Resolução da situação 3

Na situação 3 seria:

Quando $G1=1$ (verde do sinal 1 aceso) quando $A = 1$ e $B = 1$

Portanto, $G1=1$ quando $A \cdot B = 1$

Agora, em quais situações R1 (vermelho do sinal 1 aceso) acende?

Resposta: nas situações 0 OU 1

Veja resolução da situação 0

Veja resolução da situação 1

Como se tem $R1=1$ na situação 0 OU situação 1, uma porta contendo as expressões tanto da situação 0 quanto da situação 1 resultará no valor 1 nesses casos, que representa a situação referente ao vermelho aceso do semáforo 1

$$R1 = \bar{A} \cdot \bar{B} + \bar{A} \cdot B$$

Escreva as expressões quando

- $G2 = 1$
- $R2 = 1$

$G2 = 1$ nas situações 0 OU 1

- Situação 0: $= 1$
- Situação 1: $= 1$

Portanto:

$$G2 = \bar{A} \cdot \bar{B} + \bar{A} \cdot B$$

$R2 = 1$ nas situações 2 OU 3

- Situação 2: $= 1$
- Situação 3: $= 1$

Portanto:

$$R2 = A \cdot \bar{B} + A \cdot B$$

Situação	A	B	G1	R1	G2	R2
0	0	0	0	1	1	0
1	0	1	0	1	1	0
2	1	0	1	0	0	1
3	1	1	1	0	0	1

Situação	A	B	G1	R1	G2	R2
0	0	0	0	1	1	0
1	0	1	0	1	1	0
2	1	0	1	0	0	1
3	1	1	1	0	0	1

Veja resolução da situação 0

Situação 0:

$R1=1$ quando $A = 0$ e $B = 0$, ou seja, $\bar{A} = 1$ e $\bar{B} = 1$

Usando uma porta E, é possível escrever $R1=1$

quando $\bar{A} \cdot \bar{B} = 1$

Resolução da situação 1

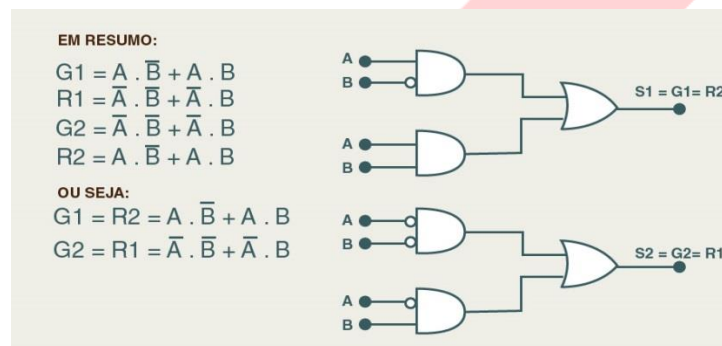
Situação 1:

R1=1 quando A = 0 e B = 1

Portanto, R1=1 quando $\bar{A}.B = 1$

13

Com o intuito de ilustração, veja abaixo uma representação das equações e do circuito lógico.



14

3- Circuito combinacional

Dando continuidade ao nosso estudo sobre arquitetura de organização de computadores, veremos o que é um circuito combinacional. Cabe ressaltar que o autor Willian Stallings chama circuito combinacional de **circuitos combinatórios**. Também teremos a oportunidade de estudar os tipos de circuitos combinacionais e o que seja o projeto de um multiplicador e de um decodificador. Vamos começar nosso estudo.

No livro Arquitetura e Organização de Computadores, o autor Willian Stallings define que:

Um **circuito combinatório** consiste em uma interconexão de portas lógicas (ou circuitos lógicos, dispositivos que operam um ou mais sinais lógicos de entrada para produzir uma e somente uma saída, dependente da função implementada no circuito).

Portas lógicas são geralmente usadas em circuitos eletrônicos, em virtude das situações que os sinais deste tipo de circuito apresentem para presença de sinal o valor "1"; e a ausência de sinal "0". As

situações chamadas de "Verdade" e/ou "Falso" são estudadas na lógica de Boole (lógica matemática), que é a origem do nome destas portas.

A maneira de como é o comportamento das portas lógicas é conhecido pela tabela verdade, que apresenta os estados lógicos das entradas e das saídas. Desta forma, como em uma porta simples, a alteração de sinais de entrada é quase imediatamente seguida pela alteração correspondente no sinal de saída, apenas com retardo devido à transmissão de sinais por meio das portas do circuito.

Portas lógicas

As portas lógicas são circuitos eletrônicos que operam com um ou mais sinais de entrada de forma a produzir um sinal de saída. Uma vez que computador digital processa informações como um conjunto de sinais elétricos que podem ser considerados para representar valores binários onde um determinado valor pode ser interpretado como zero ou um.

15

Em termos gerais, um circuito combinatório consiste de n entradas binárias em m saídas binárias. Assim como uma porta, um circuito combinatório pode ser definido de três formas:

- **Tabela verdade** → Para cada uma das possíveis combinações dos n sinais de entrada, é listado o valor binário de cada um dos m sinais de saída.
- **Símbolos gráficos** → Esquema de conexão das portas lógicas
- **Equações booleanas** → Cada sinal de saída é expresso como uma função booleana dos sinais de entrada.

No início da eletrônica, para ter uma solução para os problemas, utilizavam-se os sistemas analógicos. Com a evolução da tecnologia, os problemas passaram a ser resolvidos pela eletrônica digital.

Na eletrônica digital, os sistemas (computadores pessoais, processadores, codificadores, decodificadores etc.) empregam um grupo pequeno de circuitos lógicos básicos, que são conhecidos como **portas (e, ou, não e flip-flop)**. Com a utilização adequada dessas portas tem-se a possibilidade de implementação das expressões geradas pela **álgebra booleana**.

Sistemas analógicos

Dispositivos que manipulam quantidades físicas sob a forma analógica. Nestes sistemas, as quantidades variam continuamente dentro de uma faixa de valores. Exemplo de equipamentos analógico: videocassete.

16

Vale lembrar que, na álgebra de Boole ou Booleana, existem apenas dois estados (valores ou símbolos) que são permitidos, sendo o estado 0 (zero) e o estado 1 (um).

0

Geralmente o estado zero (0) representa **falso, desligado, não, chave elétrica desligada, ausência de tensão, ausência de sinal** e etc.

1

Já o estado 1 (um) representa **verdadeiro, aparelho ligado, sim, chave ligada, presença de sinal** e etc.

Sendo assim, na álgebra de Boole ou booleana, se representarmos por 0 uma situação, a situação contrária é representada por 1 e assim por diante.

Consequentemente, em qualquer bloco (porta ou função) lógico somente esses dois estados (0 ou 1) são permitidos em suas entradas e saídas. Vale a pena rever aqui algumas formas de representação de blocos:

E (AND)
OU (OR)
NÃO (NOT)
NÃO E (NAND)
NÃO OU (NOR)
OU EXCLUSIVO (XOR)

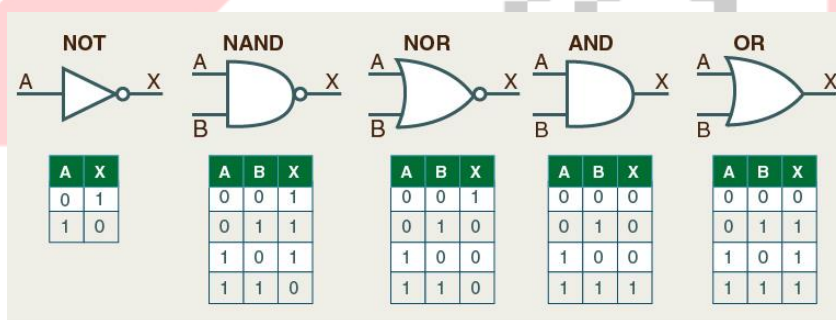


Tabela Verdade

17

Para fixar bem esse assunto e compreender melhor o conceito de portas lógicas e tabela verdade, assista ao vídeo a seguir.

<https://www.youtube.com/watch?v=17JHVZuK36A>

4- CIRCUITOS COMBINACIONAIS DE INTERCONEXÃO

Os circuitos combinacionais têm a responsabilidade pelas operações lógicas e aritméticas em um sistema digital (computador). Estes circuitos fazem muito mais que apenas operações lógicas e aritméticas, há ainda outras funções necessárias para a realização de ligações entre os diversos operadores. Dentre essas funções há:

- a multiplexação e
- a decodificação.

Os responsáveis por realizar essas operações são denominados multiplexadores e decodificadores, respectivamente, e são também circuitos combinacionais.

4.1 - DECODIFICADORES

Um decodificador é um circuito combinacional usado para ativar ou habilitar um (e somente um) dentre m componentes. Assume-se que cada componente tem um índice entre 0 e m-1, representado por um endereço em binário. Um decodificador $n : m$ (lê-se n por m) possui n entradas e m saídas, com $m \leq 2^n$.

No caso de um decodificador **3:8 (n=3 : m=8)**, serão 8 saídas, onde cada saída pode ser vista como um diferente endereço. Para ativar uma dentre 8 saídas, são necessárias 3 variáveis de entrada (por isso 3:8). Cada arranjo das variáveis de entrada seleciona uma e apenas uma dentre as 8 saídas, de tal maneira que cada saída apenas será selecionada por um dos 8 arranjos. Desta maneira, é presumido que se associe a cada saída um índice decimal que represente o arranjo de entradas responsável pela sua ativação.

Tendo por base a ativação em lógica direta, sendo assim, uma saída está ativada quando seu valor é 1, então uma tabela verdade para um decodificador 3:8 poderia ser conforme a figura abaixo:

Endereço	Entradas (sinais de controle)			Saídas							
	A ₂	A ₁	A ₀	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅	D ₆	D ₇
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

Observe que uma saída terá o valor **1** para um determinado arranjo das variáveis de entrada. Para cada combinação de entrada só é ativada uma dentre todas as 8 saídas. O circuito de um decodificador 3:8 terá, portanto, 8 saídas, sendo cada saída um dentre os 8 termos possíveis para uma função Booleana de 3 variáveis.

20

Um decodificador pode vir a possuir uma entrada de *enable* (habilitação). Esta entrada tem a função de habilitar ou desabilitar seu funcionamento. Assim, se esta entrada tiver o valor 0, nenhuma saída estará ativada, independente dos valores das demais entradas. Caso a entrada de habilitação tiver o valor 1, o decodificador estará ativando uma das saídas.

Na figura abaixo há uma tabela verdade de um decodificador 2:4:

Endereço	Entradas (sinais de controle)			Saídas			
	E	A ₁	A ₀	D ₀	D ₁	D ₂	D ₃
-	0	0	0	0	0	0	0
-	0	0	1	0	0	0	0
-	0	1	0	0	0	0	0
-	0	1	1	0	0	0	0
0	1	0	0	1	0	0	0
1	1	0	1	0	1	0	0
2	1	1	0	0	0	1	0
3	1	1	1	0	0	0	1

saídas
desabilitadas

Como pode ser visto nas primeiras 4 linhas, o sinal de habilitação (E) vale zero, o que desativa as saídas, independentemente dos valores das demais entradas (A₁ e A₀).

21

Poderemos escrever novamente esta tabela de uma maneira mais enxuta, escrevendo em uma linha que, quando $E=0$, os valores das entradas A_1 e A_0 não interessam (=don't cares de entrada):

Endereço	Entradas (sinais de controle)			Saídas			
	E	A_1	A_0	D_0	D_1	D_2	D_3
-	0	X	X	0	0	0	0
0	1	0	0	1	0	0	0
1	1	0	1	0	1	0	0
2	1	1	0	0	0	1	0
3	1	1	1	0	0	0	1

saídas
desabilitadas

Figura A

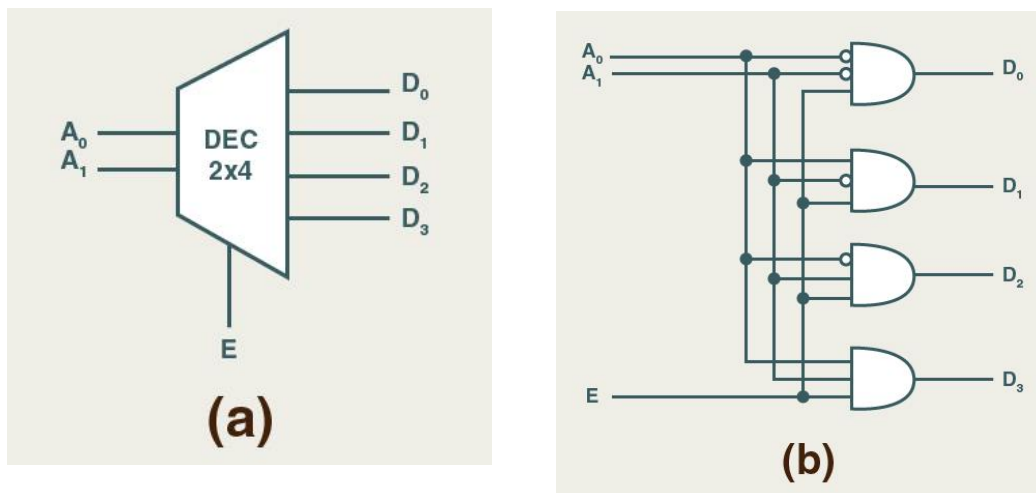


Figura B

Nas figuras acima, observamos na figura A uma tabela para esse decodificador e na figura B visualiza-se uma possível implementação (circuito lógico).

22

Resumo

Neste módulo tivemos a oportunidade de estudarmos juntos a importância do entendimento do que seja um circuito sequencial e um circuito combinacional. Vimos que em especial o circuito combinacional é constituído por um conjunto de portas lógicas, as quais determinam os valores das saídas diretamente tendo por base os valores atuais das entradas. Você compreendeu, ao relembrar da importância da lógica booleana, que é a partir de um circuito combinacional que podemos criar uma tabela de condição e de notação.

UNIDADE 2 – FUNCIONAMENTO DOS CIRCUITOS COMBINACIONAIS E SEQUENCIAIS

MÓDULO 3 – CIRCUITO SEQUENCIAL

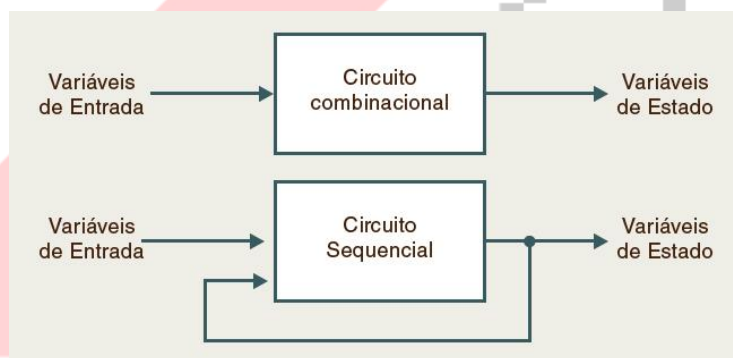
01

1 - CIRCUITO SEQUENCIAL

Dando continuidade ao nosso estudo sobre arquitetura de organização de computadores, neste módulo vamos entender o que é um circuito sequencial, conhecer os principais tipos de circuitos sequenciais e o circuito equivalente a partir de uma expressão lógica. Também durante o estudo deste módulo iremos definir o que seja circuito sequencial, quais os tipos de *flip-flops* e entendermos o que é uma memória.

Para começarmos a definir um circuito sequencial, precisamos relembrar que os circuitos combinatórios implementam as funções básicas de um computador digital. Contudo, exceto no caso especial da ROM, eles não disponibilizam informação de estado ou memória, elementos também essenciais para a operação de um computador digital. Para esse fim, utiliza-se uma forma mais complexa de circuito lógico digital, conhecido por **circuito sequencial**.

A saída corrente de um circuito sequencial tem dependência não somente da entrada corrente, também tem de valores anteriores da entrada. Outra maneira, e normalmente mais útil, de visualizar esses circuitos é que a saída corrente de um circuito sequencial depende da entrada corrente e do estado do circuito.



Isso quer dizer que, para se obter o próximo estado do sistema (circuito sequencial), é necessário lembrar-se do estado atual (em outros termos, o caminho que me leva à sala depende do lugar na casa onde estou). Significa que o sistema necessita possuir memorizar o estado atual para alcançar o estado seguinte. A lição disso é que o circuito sequencial necessita de elementos de memória.

ROM

Read-Only Memory (ROM), que em português quer dizer “Memória de Apenas Leitura”. Basicamente, a função da memória ROM é oferecer dados apenas para leitura. Normalmente, a ROM é utilizada para armazenar firmwares, pequenos softwares que funcionam apenas no hardware para

o qual foram desenvolvidos e que controlam as funções mais básicas do dispositivo. Na ROM de uma calculadora, por exemplo, podemos encontrar as rotinas matemáticas básicas. No caso de celulares, o normal é que as ROMs carreguem o sistema operacional e os softwares básicos do dispositivo.

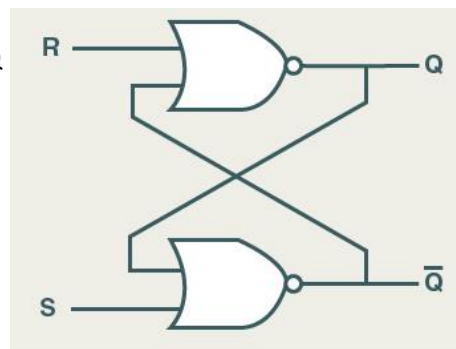
02

Dentre os tipos mais conhecidos de circuito sequencial, iniciaremos pela forma mais básica, que é o *flip-flop*. Há uma variedade de *flip-flops*, todos compartilhando duas características:

- 1- O *flip-flop* é um dispositivo biestável. Existe em 1 de 2 estados estáveis e, na falta de sinal de entrada, permanece nesse estado. Portanto, o *flip-flop* funciona similarmente a uma memória de 1 bit.
- 2- O *flip-flop* tem duas saídas, que geralmente possuem valor complementar uma da outra. Essas saídas são normalmente rotuladas de: Q e \overline{Q} .

No intuito de exemplificar, é possível visualizar na figura uma configuração comumente reconhecida como flip-flop S-R ou latch S-R. Este circuito possui duas entradas, S (**Set** -) e R (**Reset**), e duas saídas, Q e \overline{Q} , e consiste de duas portas **NOR** (NÃO OR), conectadas de forma que a saída retorna como entrada para a outra.

Inicialmente, observe por que esse circuito é biestável, ou seja, tem dois estados. Imaginemos que tanto **R** quanto **S** tenha valor zero (0) e que **Q** também tenha o valor zero (0).



As entradas da porta **NOR** inferior são $Q=0$ e $S=0$. Portanto, a saída $\overline{Q}=1$ isto tem o significado que as entradas para a porta **NOR** superior são $\overline{Q}=1$ e $R=0$, o que terá como saída $Q=0$. Desta maneira, o estado do circuito é coerente internamente, e permanece estável desde que $S=R=0$. Um raciocínio análogo mostra que o estado $Q=1$, $\overline{Q}=0$ também é estável, para $R=S=0$.

Flip-flop

Um flip-flop é um circuito básico que armazena um bit de informação.

03

Agora vamos abordar a **memória**. Antes de qualquer coisa é preciso entender que a maioria dos dispositivos ou circuitos possuem dois estados estáveis, chamados de biestável. Uma lâmpada pode estar ligada ou desligada, dependendo da posição do interruptor. Determinado circuito poderá fazer uso de um componente que armazene (memória) sobre o estado que se encontra (ligado ou desligado), permanecendo em um estado definido até que alguém ou algo mude este estado. Quando um sinal de entrada é aplicado num dispositivo, a saída muda em resposta à entrada. Quando o sinal de entrada é retirado, a saída volta ao seu estado original. Este dispositivo não mostra a propriedade de memória, em virtude de sua saída voltar ao estado anterior.

Há dispositivos e circuitos digitais que possuem memória. Esta característica faz que, ao receberem determinado sinal de entrada, esta informação fique guardada na memória. A saída desta informação da memória poderá mudar seu estado ou não, podendo inclusive permanecer o valor da entrada mesmo após a entrada ter sido retirada.

Sendo assim, chama-se de **memória** a propriedade de reter sua resposta a uma entrada momentânea. Consequentemente, memória é todo dispositivo que mantém indefinidamente uma informação ao longo do tempo.

Ao mencionar dispositivos com capacidade de memória, lembramos geralmente da mente humana e dos processadores digitais, entretanto, há vários tipos de memória. Pode parecer estranho e provavelmente você já sabe, mas os livros, fotos, CDs e DVDs de música são também dispositivos de memória. Até coisas simplórias podem ter a função de memória. Alguns historiadores dizem que o famoso Einstein usava uma caneta no bolso do lado esquerdo ou direito para poder lembrar se já havia ou não almoçado.

04

Nos sistemas digitais que têm por base a lógica combinacional, os estados de suas saídas são dependentes **apenas dos estados presentes** (agora) das entradas, logo, estes sistemas não conseguem lidar com a variável **tempo** e perceber sequências de eventos, portanto, não têm capacidade de resolver qualquer problema que envolva a noção de tempo.

Usaremos como exemplo um controle automático de enchimento de uma caixa de descarga (caixa acoplada ao vaso sanitário). O objetivo é que o sistema controle a válvula de entrada V a partir de dois sensores de nível de água A e B, como no esquema a seguir.

M	A	B	V
0	0	0	1 (liga válvula, caixa acabou de esvaziar)
0	0	1	0 (caixa esvaziando)
0	1	0	X (impossível)
0	1	1	0 (válvula foi recentemente desligada)
1	0	0	1 (válvula foi recentemente ligada)
1	0	1	1 (caixa enchendo)
1	1	0	X (impossível)
1	1	1	0 (desliga válvula, caixa acabou de encher)

Um circuito de lógica sequencial possui elementos de armazenamento e as saídas não são dependentes apenas das entradas presentes no momento. Elas dependem também de valores armazenados. Sendo assim, pode ser visto como uma parte puramente combinatória juntamente com uma memória.

Ao conjunto formado pelo bloco combinacional e também com mais um dispositivo de memória é o que chamaremos de **lógica sequencial**.

Nesta lógica, os estados presentes das saídas não são dependentes apenas dos estados das entradas, mas são também dependentes dos estados anteriores do próprio sistema (memória).

Uma solução para o problema da caixa de descarga é usarmos um dispositivo de memória com a capacidade de guardar um bit, que é a definição funcional de **flip-flop**.

05

2- SISTEMAS DIGITAIS

Os **sistemas digitais** estão divididos em duas classes:

- sistemas combinacionais e
- sistemas sequenciais.

Como já estudamos anteriormente, nos sistemas combinacionais, uma saída no tempo **t** depende somente da entrada no tempo **t**. Sendo assim, o sistema **não possui memória em virtude da saída não depender de entradas anteriores**. Consequentemente, a saída depende unicamente das variáveis de entrada.

Para melhorar o entendimento, usaremos como exemplo um cadeado que utiliza números (códigos), geralmente utilizado para prender bicicletas no bicicletário. Este tipo de cadeado poderá ser aberto num dado tempo **X** quando a sequência numérica (código) do cadeado for colocado nas entradas em **X**, sem considerar a história das entradas. Caso o código utilizado seja 802, por exemplo, o cadeado somente será aberto quando a referida combinação for colocada nas entradas, independentemente da ordem adotada na colocação dos dígitos do código.

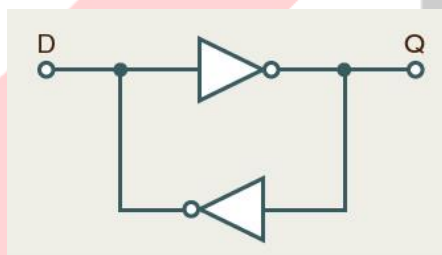
Nos sistemas sequenciais, uma saída no tempo X depende da entrada no tempo X e será bem possível que também haja dependência da entrada no tempo anterior a X. **A saída tem dependência das variáveis de entrada e/ou de seus estados anteriores guardados.**

06

Outro exemplo interessante é o sistema de discagem telefônica. Um número utilizado por um usuário quando for discado será feito em um determinado momento X, caso sejam atendidas as seguintes condições:

- a) os números discados antes do instante X devem seguir uma sequência conforme o número do usuário;
- b) o número discado no instante X, isto é, o último a ser discado, corresponde ao último dígito do número do usuário;
- c) todos os números devem estar na memória e estarem disponíveis na mesma ordem da discagem no instante X.

Ao obter o efeito memória (buffer realimentado) precisamos entender que construtivamente, um *flip-flop* pode ser descrito como um inversor realimentado por outro inversor, conforme pode ser observado na figura abaixo.

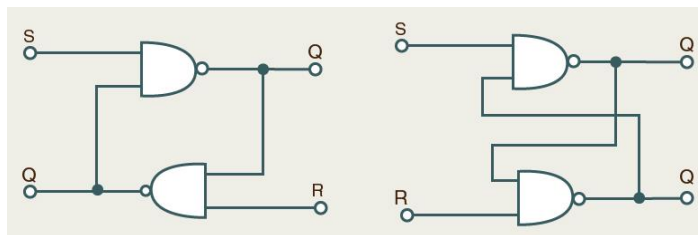


Ao olharmos para a figura acima, podemos perceber que uma vez determinado um dos estados lógicos à entrada D, o estado da saída Q se manterá de forma sem definição.

Como poderíamos mudar o estado de Q sem provocar uma contradição com o estado de Q? Vejamos a seguir.

07

Para mudar o estado de Q sem que provoque uma contradição com o estado de Q, a solução é adicionar terminais de entrada, trocando os inversores por portas lógicas **Não-E**.



Sendo assim, podemos levar ($Q = 1$) impondo 0 em S (set) e levar ($Q = 0$) impondo 0 em R (reset), armazenando o estado que desejamos no *flip-flop*. Entretanto, o *flip-flop* necessita de aperfeiçoamento para atender à definição lógica de **flip-flop**, que é um dispositivo com duas saídas complementares Q e \bar{Q} , com duas entradas S e R que operam de acordo com a tabela abaixo.

S	R	Q	\bar{Q}
0	0	não permitido	
0	1	1	0
1	0	0	1
1	1	Q_A	\bar{Q}_A

08

3- CIRCUITOS ELETRÔNICOS

Numa breve explicação sobre circuito eletrônico podemos dizer que uma de suas características é ser biestável em virtude de possuir dois estados estáveis, isto é, sua saída é 0 (nível lógico 0) ou 1 (nível lógico 1). Assim, este dispositivo pode ser usado para armazenar um dígito binário (bit).

Para melhorar o entendimento dos conceitos que iremos estudar, vamos inicialmente falar sobre Latch, que é a forma mais básica de se implementar um circuito lógico de memória. Latch significa, em português, trinco, ferrolho.

O Latch é um circuito sequencial biestável assíncrono (muda de estado sem necessidade de sincronismo), sendo assim, é um circuito constituído por portas lógicas, podendo armazenar um bit de informação, onde as saídas em determinado momento dependem dos valores de entrada.

Latches controlados D e RS são controlados ou ativados pelo nível lógico do sinal de controle. O significado é que enquanto houver o sinal de controle ativando o latch, eventuais mudanças nas entradas D ou R e S serão detectadas pelo latch e por consequência irá mudar de estado. Essa característica não é bem-vinda na construção de circuitos sequenciais síncronos, isto é devido ao fato de, nestes circuitos, qualquer mudança de estado deve acontecer de forma sincronizada conforme o sinal do clock (relógio).

Os flip-flops são circuitos que têm por referência os latches, porém sua ativação é feita pela transição do sinal de controle. Esta característica faz que um flip-flop se mantenha ativado apenas durante um pequeno intervalo de tempo, após a ocorrência de uma transição do sinal de controle. Sendo assim, uma provável troca de estado só deve ocorrer durante esse pequeno intervalo de tempo em que o flip-flop está ativado. O flip-flop mantém o último estado adquirido entre duas transições sucessivas do mesmo tipo (ou subida ou descida) do sinal de controle.



Dependendo de como foi a construção de um flip-flop ele pode ser disparado pela transição de subida ou pela transição de descida do sinal de controle. Neste caso os flip-flops são disparados pela borda (ascendente ou descendente, conforme for o caso), no entanto os latches são sensíveis ao nível lógico (alto ou baixo, conforme for o caso).

09

Já falamos anteriormente sobre flip-flops, mas está faltando uma definição formal do que venha a ser.

Um **flip-flop** é um elemento de circuito que pode apresentar em seu funcionamento apenas dois estados estáveis. Aplicando um sinal de entrada poderá haver a mudança de um estado para outro e desta forma pode-se conhecer o respectivo estado do sinal em que se encontrava.

Exemplo: um sinal é aplicado podendo ser 0 ou 1, caso o estado estivesse 0 ao aplicar o sinal 1 haveria mudança de estado e vice-versa. Sendo assim, este circuito é considerado como uma célula básica de memória da lógica sequencial capaz de guardar um bit.

S	R	Q _A	Q _F
0	0	0	0 - estável
0	0	1	1 - estável
0	1	0	0 - estável
0	1	1	0 - instável
1	0	0	1 - instável
1	0	1	1 - estável
1	1	0	1 - instável (não permitido)
1	1	1	1 - instável (não permitido)

} Q_A

} 0

} 1

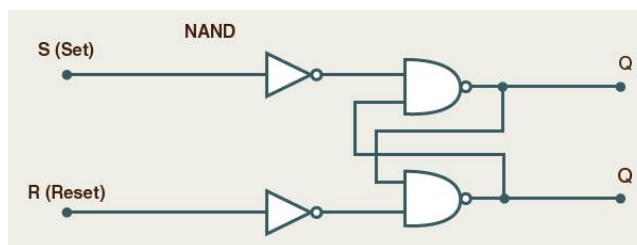
S	R	Q _F
0	0	Q _A
0	1	0
1	0	1
1	1	não permitido

Na tabela acima vemos os possíveis estados do flip-flop. Vale lembrar que ele armazena apenas 1 bit e que este armazenamento obedece a regras de estabilidade, conforme estudamos nos parágrafos anteriores. Veja no quadro abaixo uma demonstração do S (set), R(reset) e o Q que é a saída.

10

Existem vários tipos de flip-flop, tais como: FLIP-FLOP SR BÁSICO, FLIP-FLOP JK, FLIP-FLOP JK Mestre-Escravo, FLIP-FLOP JK Mestre-Escravo com terminais de programa, FLIP-FLOP T e FLIP-FLOP D.

Neste momento iremos falar especificamente do FLIP-FLOP SR BÁSICO. Ele possui duas entradas, definidas como Set e Reset e duas saídas Q e Q. Estas saídas somente podem permanecer com valores lógicos complementares.



11

4- MEMÓRIAS

As memórias são dispositivos semicondutores que guardam informações na forma binária. Estas informações são constituídas de números, letras, caracteres diversos, comandos, operações, endereços e etc. Os bits das informações podem ser acessados no momento de leitura ou gravação/substituição, ou no procedimento de escrita ou armazenamento.

As memórias semicondutoras são utilizadas como memória principal (interna) ou memória de trabalho de uma máquina (computador), pois mantêm comunicação constante com a unidade central de processamento (CPU) à medida que um programa de instruções está em execução.

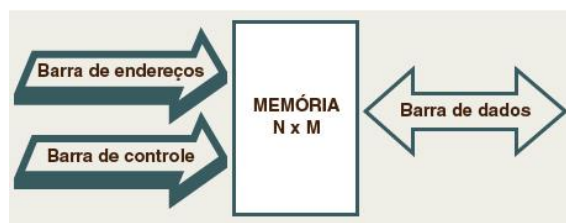
Outra maneira de armazenar no computador é pela memória auxiliar (externa) ou memória de massa ou memória secundária, que trabalha em uma velocidade inferior a memória principal e armazena programas e dados que não estão sendo usados a todo o momento pela CPU ou UCP.

Para evitar troca excessiva de informações e instruções, esta memória transfere as informações para a memória principal apenas quando for necessário ou solicitado no processamento.

12

Uma memória guarda ou acessa as informações digitais em lugares denominados localidades, mediante o uso de um endereço (endereçamento). Para acessar estas localidades, um bloco possui uma série de

terminais de entradas de endereços que são ligados a um conjunto de fios denominado barra de endereços (inglês= *address bus*). Para a entrada e saída dos dados, da mesma forma, o bloco possui uma série de terminais ligados à barra de dados (inglês= *data bus*). O bloco ainda possui terminais de controle ligados à barra de controle (inglês= *control bus*).



A barra de dados é bidirecional, sendo assim, pode ser utilizada tanto como entrada como para saída de dados, onde um dos terminais da barra de controle irá definir o sentido. As memórias são especificadas pela notação $N \times M$, onde **N** indica o número de localidades de memória (palavras) e **M** indica o número de *bits* da informação armazenada por localidade. O número de bits que podem compor uma palavra varia de computador para computador, estando na faixa de 4 bits até 36 bits, geralmente. Uma dada "pastilha" de memória guardará um dado número de palavras de tantos bits por palavra. Tomemos por exemplo uma pastilha de memória popular que a capacidade de armazenar 1024 palavras de 4 bits cada, totalizando 4096 bits (4K), que é a tamanho total de memória.

13

A maneira como é organizada uma pastilha de memória é por meio do estabelecimento de como é constituída por um grupo de registradores, onde cada registrador armazena uma palavra. A "largura" de cada registrador é o número de bits por palavra. O número de registradores é o número de palavras guardadas na memória. Valores comuns para o número de palavras por pastilha são 64, 256, 512, 1024, 2048 e 4096.

A maioria deles são de base dois (potências inteiras de 2). Os valores mais comuns para o tamanho da palavra são 1, 4 e 8. Também há a possibilidade de obtenção de outros tamanhos para as palavras, ao se combinar várias pastilhas de memória.

As informações que estão em cada registrador estarão sujeitas a duas operações que são possíveis: **escrita** e **leitura**.

A **leitura** é o processo de obtenção da palavra que está guardada no registrador e remetê-la para outra localização, onde poderá ser utilizada.

As informações que estão guardadas no registrador não serão modificadas pela operação de leitura.

A **escrita** é o processo de por uma nova palavra em um determinado

registrador.

Fica evidente que a operação de escrita remove a palavra que estava previamente guardada no registrador. Contudo, não são todas as pastilhas de memória que possuem a propriedade de ter o seu conteúdo escrito.

Palavra

Palavra (inglês: word) é a unidade natural de informação usada pelo computadores. É uma sequência de bits com tamanho fixo e que é processado em conjunto em uma determinada máquina. O número de bits em uma palavra (tamanho / comprimento da palavra) é uma característica preponderante da arquitetura de um computador. Esta característica ecoa em vários aspectos de sua estrutura, da sua operação e na indicação da unidade de transferência entre a UCP e MP (memória principal).

14

Com todas as diferenças existentes na implementação de cada um dos tipos de memória e mesmo com toda evolução computacional, ainda permanecem da mesma forma a implementação e os princípios básicos de operação da memória. Particularmente em cada sistema há certo conjunto de tipos diferentes de entrada e saída para a realização das seguintes **funções**:

- Eleger o endereço que está sendo utilizado em determinada operação de leitura ou escrita;
- Escolher a operação que deverá ser realizada, leitura ou escrita;
- Ofertar os dados de entrada para a realização da operação de escrita;
- Estabilizar e manter as informações de saída da memória que são resultado de determinada operação de leitura, durante determinado tempo;
- Desativar (ou Ativar) a memória, de maneira a torná-la (ou não) em condições de responder ao endereço na entrada e ao comando de leitura/escrita.

As memórias podem ser **classificadas** quanto:

- à forma de acesso,
- à volatilidade,
- ao tipo de armazenamento,
- capacidade de armazenamento e
- à tecnologia.

Veremos cada um desses tipos a seguir.

15

4.1– FORMA DE ACESSO.

As maneiras de acessar uma locação de memória podem ser de dois tipos: acesso sequencial ou aleatório.

SEQUENCIAL	ALEATÓRIO
Na forma de acesso sequencial, a locação não deve ser feita diretamente. Sendo assim, um número variado de locações da memória é acessado até a informação desejada. O tempo gasto para acessar irá depender da posição onde está localizada (armazenada) a informação.	No acesso aleatório, como o próprio nome diz, qualquer locação pode ser acessada de maneira aleatória, então, poderá ser lida de forma direta e por consequência não haverá necessidade da leitura das outras locações. Este tipo de acesso possui a vantagem de ter um tempo de acesso reduzido e sem diferenciação para qualquer uma das localizações da memória.

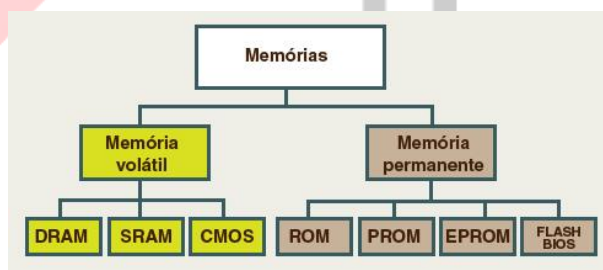
16

4.2- VOLATILIDADE

Quando falamos que é algo é volátil, geralmente, estamos dizendo que é algo que evapora com facilidade. Outras vezes podemos também entender, usando o sentido figurado, que é algo volúvel, ou seja, instável ou inconstante. Quando falamos em volátil uma substância geralmente lembrada é o éter, que evapora com muita facilidade.

As memórias voláteis também possuem esta característica, pois quando é interrompida sua alimentação (corrente elétrica – computador é desligado) as informações armazenadas são perdidas. O exemplo da área da TI são as memórias RAM.

Há as memórias que não são voláteis. Elas possuem a propriedade de mesmo sem alimentação permanecerem e manterem as informações que estão armazenadas. O exemplo utilizado da área da TI são as memórias ROM, PROM, EPROM.



17

4.3- TIPO DE ARMAZENAMENTO

Quanto ao tipo de armazenamento, as memórias podem ser:

• de armazenamento →
ESTÁTICO

Possuem a propriedade de manter-se de forma permanente ao se inserir um determinado dado em uma determinada localização. Uma de suas características é a facilidade de ser utilizada.

• de armazenamento →
DINÂMICO

Que possuem a característica de haver a necessidade de inserção de informação em intervalos repetidos de tempo, com o intuito de evitar que estas informações sejam perdidas.

18

4.4- CAPACIDADE DE ARMAZENAMENTO

Quando falamos em capacidade de armazenamento estamos fazendo referência ao número total de bits ou palavras que a memória pode guardar (armazenar).



Veja o exemplo: quando lemos, em uma especificação de memória, a seguinte nomenclatura: 1.024 x 8, podemos entender que a numeração de 1.014 corresponde à capacidade de armazenamento de 1.024 palavras. O número 8 indica que cada palavra terá 8 bits, tudo isso totaliza 8.192 bits.

19

4.5– Quanto à tecnologia

Quando falamos em tecnologia devemos entender o significado desta palavra.

Tecnologia vem do grego (do grego Τεχνη — "técnica, arte, ofício" e λογία — "estudo", ou seja, "técnica + estudo") e está relacionada a tudo que envolve conhecimento científico e técnico e sua aplicação na transformação de ferramentas, de processos e nos materiais criados e utilizados a partir deste

conhecimento.

As memórias são construídas com tecnologia que pode utilizar:

- semicondutor bipolar,
- semicondutor de óxido metálico (MOS)
- ou semicondutor de óxido complementar (CMOS).

As memórias do tipo bipolares são encontradas nos circuitos TTL padrão, e no Schottky ECL. As memórias que são do tipo MOS, e utilizam o canal N, são as mais utilizadas em virtude de apresentarem grande densidade e serem de custo baixo. Contudo, as memórias CMOS têm por características serem mais lentas em comparação com as NMOS e bipolares, entretanto, possuem a vantagem de economizarem no consumo de energia e serem imunes a ruídos.

Quando falamos em ruído, geralmente, estamos falando de barulho, som alto, poluição sonora ou aquela música que seu vizinho insiste em escutar no último volume. Contudo, quando falamos em ruídos na TI, estamos fazendo referência a todo sinal elétrico que, ao propagar-se por um meio de transmissão, pode sofrer algum tipo de alteração ou degradação. Sendo assim, as distorções de fase ou "ruídos" são perturbações de natureza aleatória, causadas por agentes externos ao sistema.

Ruído

São distorções de fase ou perturbações de natureza aleatória provocadas por agentes externos ao sistema e que podem causar perturbação ou degradação do sinal elétrico, interferindo em sua transmissão.

20

4.6 Memória flash

Você já deve ter ouvido falar em **memória FLASH**.

A memória Flash é um tipo particular de EEPROM (Memória Somente de Leitura Programável e Apagável Eletricamente). Podemos dizer que é um chip de memória de computador que mesmo sem uma fonte de energia mantém as informações armazenadas.

A memória flash é geralmente utilizada em aparelhos eletrônicos portáteis, por exemplo: aparelhos de MP3, *smartphones*, câmeras digitais e *pendrives*.

As memórias flash são assim chamadas em virtude de possuírem **tempos curtos** para serem apagadas e escritas. A maioria destas memórias apaga um bloco em poucos microssegundos. Contudo, também são encontradas atualmente algumas que permitem apagar dados por setor.

Na tabela abaixo é possível verificar as vantagens e desvantagens dentre os vários tipos de memórias semicondutoras não voláteis. À proporção em que há o aumento da flexibilidade para apagar e programar, também há o aumento da complexidade e o aumento proporcional do custo do dispositivo.

Complexidade e custo do dispositivo ↑	MEMÓRIA	CARACTERÍSTICAS
	EEPROM	Pode ser apagada eletricamente, no circuito, byte a byte.
	FLASH	Pode ser apagada eletricamente, no circuito, por setor ou todo de uma vez.
	EPROM	Pode ser apagada toda de uma vez com luz UV e apagada e reprogramada fora do circuito.
	PROM MROM	Não pode ser apagada nem reprogramada.

Microssegundo

Microssegundo (mS) é um múltiplo do segundo, uma unidade de tempo, com o prefixo pelos padrões base multiplicador micro (μ), igual a 0,000001 segundo.

21

No intuito de incentivar a leitura e mostrar o quanto a eletrônica e os circuitos elétricos permeiam nosso dia a dia, visite a página da UNICAMP, onde é possível ler uma aula dos professores Zuben e Attux, os quais, numa aula de Introdução à Computação Natural, falam sobre a associação da biologia sintética com a eletrônica, comentam sobre a semelhança entre a interação de enzimas, fatores de transcrição e DNA, e circuitos elétricos, criando um interdisciplinaridade entre a computação e a biologia.

Estes estudos nos levam também à nova área chamada **nanotecnologia**.

A **nanotecnologia** visa produzir objetos e dispositivos na escala atômica (referida como escala nano) a partir dos átomos.

Fernando Rebouças, em um artigo escrito para o sítio da Infoescola, diz que por meio da organização de átomos, a nanotecnologia é uma ciência que possibilita a capacidade de criar objetos em diferentes “microescalas” em sistema de planejamento em engenharia molecular. Esta área científica será responsável pela nova revolução industrial avançada. A nanotecnologia está presente na medicina, eletrônica, ciência da computação, física, química, biologia e engenharia.

Assista a este vídeo sobre as 10 tecnologias de 2013:

<http://olhardigital.uol.com.br/embed/10-tecnologias-que-devem-crescer-em-2013/33130>

22

RESUMO

Durante o estudo deste módulo compreendemos o que é um circuito sequencial e conhecemos os principais tipos de circuitos sequenciais e que a partir de uma expressão lógica conseguimos chegar a um circuito equivalente. Relembramos que, em um circuito combinacional, a saída depende apenas de uma combinação das entradas, enquanto que em um circuito sequencial, a saída depende, além de uma combinação das entradas, de uma combinação das variáveis de estado do sistema, ou seja, de variáveis que identifiquem o estado em que o sistema se encontrava.

Vimos os tipos básicos de flip-flops e pudemos entender que as memórias são dispositivos semicondutores, que guardam informações na forma binária. Estas informações são constituídas de números, letras, caracteres diversos, comandos, operações, endereços e etc. Os bits das informações podem ser acessados no momento de leitura ou gravação/substituição, ou no procedimento de escrita ou armazenamento.

Vimos que para aprender o que é um circuito sequencial, precisamos relembrar o que são os circuitos combinatórios. A esses circuitos cabe implementarem as funções básicas de um computador digital. Todavia, exceto no caso especial da ROM, eles (circuitos combinatórios) não disponibilizam informação de estado ou memória, elementos essenciais para a operação de um computador digital. Para esse fim, utiliza-se uma forma mais complexa de circuito lógico digital, que vimos que é o circuito sequencial.