LIÇÃO 2

A ÁLGEBRA DE BOOLE

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

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

2.1 - A álgebra de Boole

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

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

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

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

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

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

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

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

2.2 - Os níveis lógicos

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

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

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

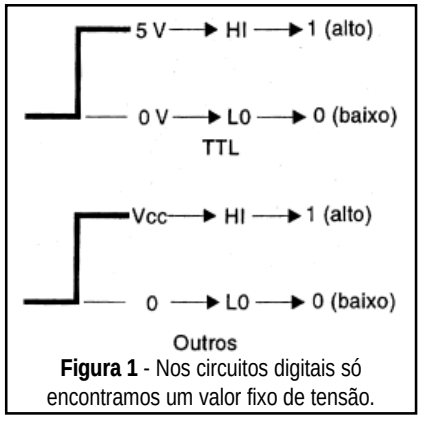

Figura 1 - Nos circuitos digitais só encontramos um valor fixo de tensão.

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

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

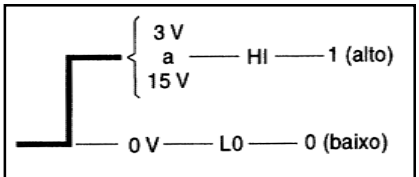

Figura 2 - A tensão encontrada nos circuitos CMOS terá um valor fixo entre 3 e 15 V.

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

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

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

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

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

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

0 V
Falso
Desligado
Nível baixo ou LO

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

3.1 - Operações Lógicas

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

Assim, podemos representar uma soma como:

$$Y = A + B$$

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

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

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

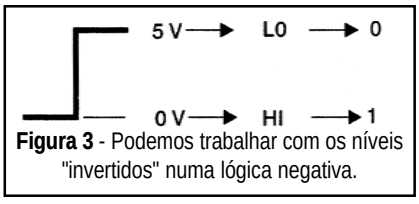

Figura 3 - Podemos trabalhar com os níveis "invertidos" numa lógica negativa.

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

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

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

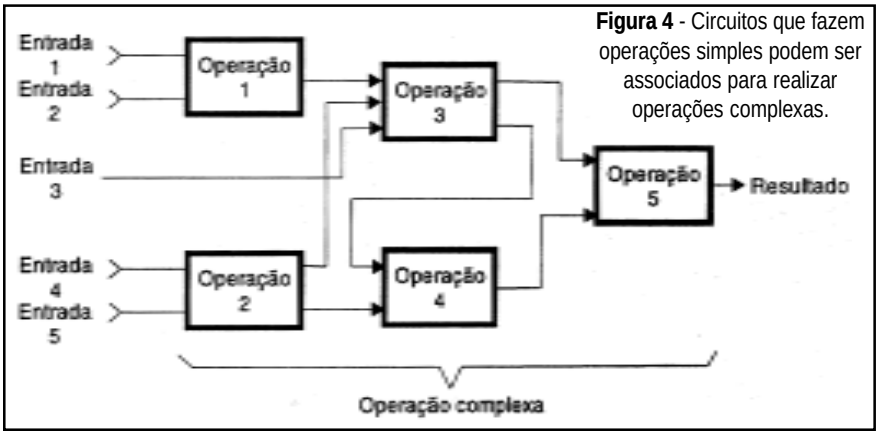

Figura 4 - Circuitos que fazem operações simples podem ser associados para realizar operações complexas.

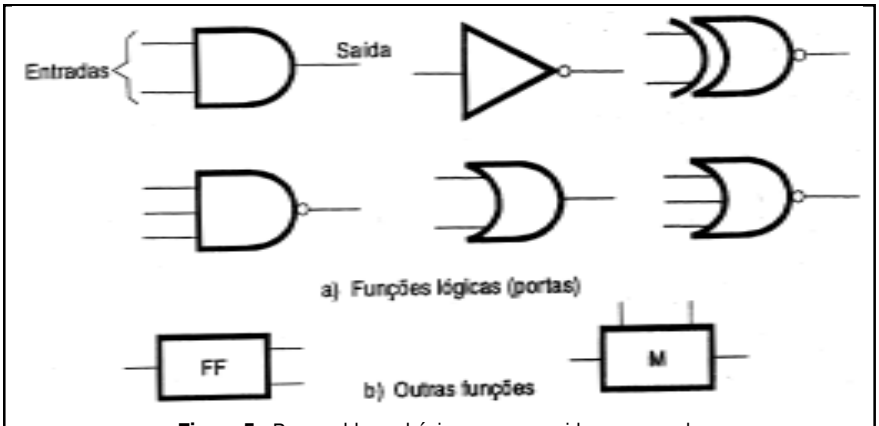

Figura 5 - Poucos blocos básicos, mas reunidos em grande quantidae podem realizar operações muito complexas.

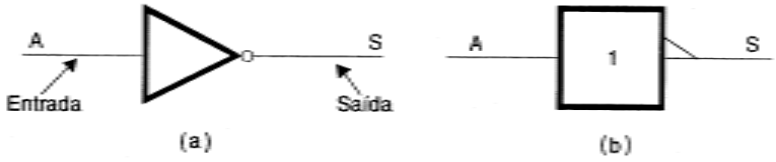

Figura 6 - Em (a) o simbolo mais comum e em (b) o simbolo IEEE usado em muitas publicações técnicas mais modernas dos Estados Unidos e Europa.

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

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

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

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

2.4 - Função Lógica NÃO ou Inversora

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

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

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

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

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

Entrada	Saída
0	1
1	0

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

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

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

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

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

Basta então lembrar que:

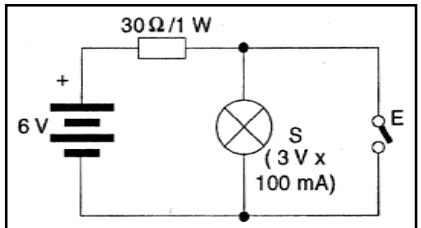

Figura 7 - Circuito simples para simular a função NÃO (NOT) ou inversor.

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

2.5 - Função Lógica E

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

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

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

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

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

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

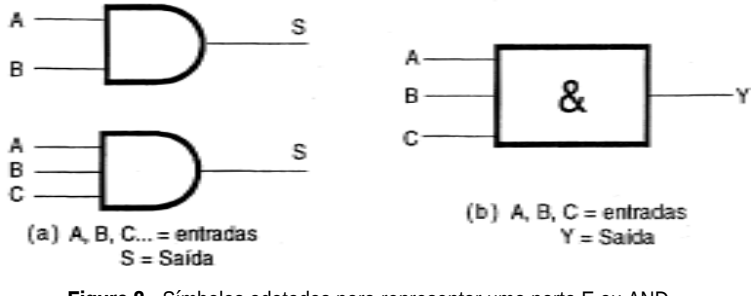

Figura 8 - Símbolos adotados para representar uma porta E ou AND.

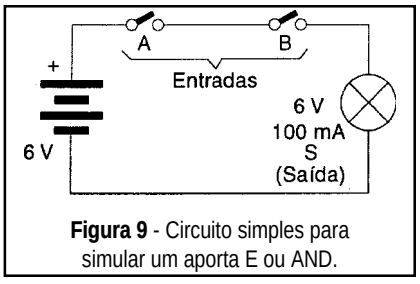

Figura 9 - Circuito simples para simular um aporta E ou AND.

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

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

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

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

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

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

2.6 - Função lógica OU

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

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

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

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

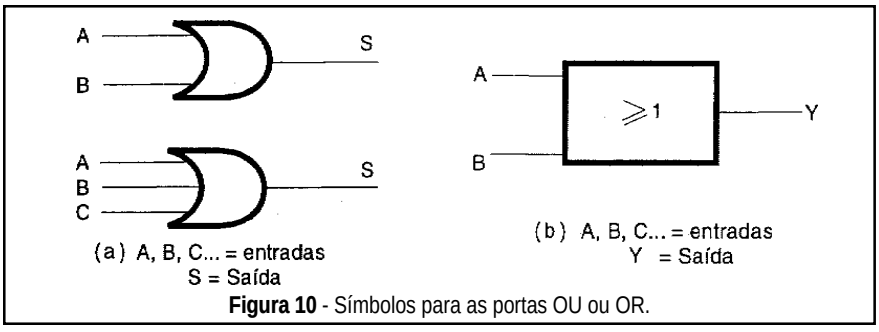

Figura 10 - Símbolos para as portas OU ou OR.

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

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

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

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

2.7 - Função NÃO-E

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

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

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

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

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

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

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

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

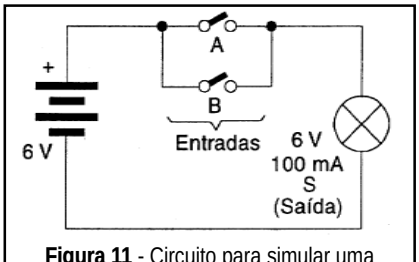

Figura 11 - Circuito para simular uma porta OU ou OR de duas entradas.

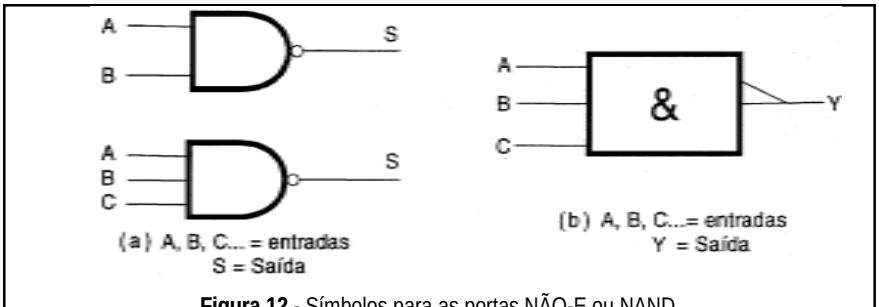

Figura 12 - Símbolos para as portas NÃO-E ou NAND.

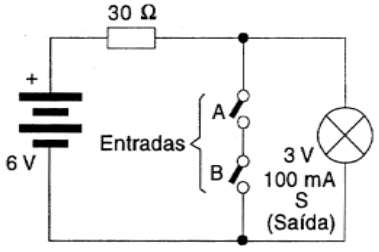

Figura 13 - Circuito que simula uma porta NAND ou NÃO-E de duas entradas.

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

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

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

2.8 - Função NÃO-OU

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

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

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

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

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

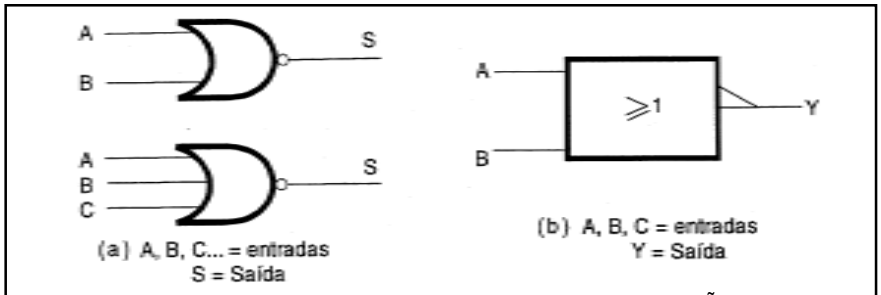

Figura 14 - Símbolo usados para representar a função NOR ou NÃO-E

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

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

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

2.9 - Função OU-exclusivo

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

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

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

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

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

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

2.10 - Função NÃO-OU exclusivo ou coincidência

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

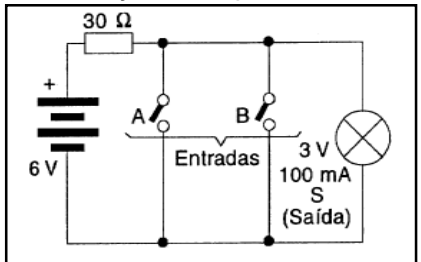

Figura 15 - Circuito usado para simular uma porta NOR de duas entradas.

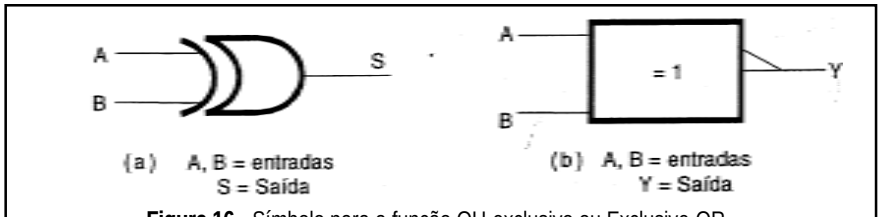

Figura 16 - Símbolo para a função OU-exclusivo ou Exclusive-OR.

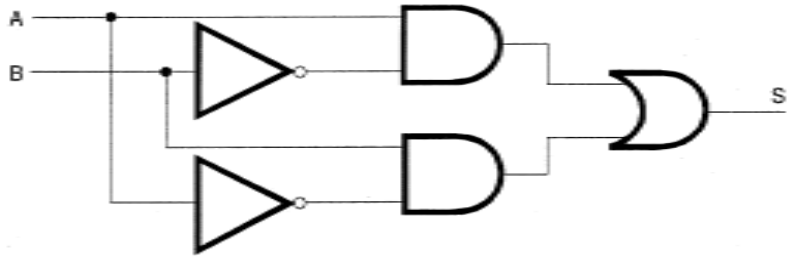

Figura 17 - Elaboração da função OU-exclusivo com inversores, portas AND e uma porta OR.

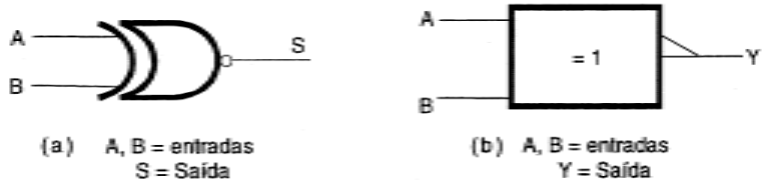

Figura 18 - Símbolos da função Não-OU-Exclusive ou Exclusive NOR também chamada função coincidência.

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

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

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

Uma tabela verdade para esta função é a seguite:

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

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

2.11 - Propriedades das operações lógicas

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

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

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

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

Representações

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

a) Operação E

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

A . B = S

b) Operação OU

Esta operação é representada pelo sinal (+).

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

A + B = S

c) Operação NÃO

Esta operação é indicada por uma barra da seguinte forma:

A\ = S

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

1. Propriedade comutativa das operações E e OU:

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

2. Propriedade associativa das operações E e OU:

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

3. Teorema da Involução:

(A negação da negação é a própria afirmação)

A\\ = A

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

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

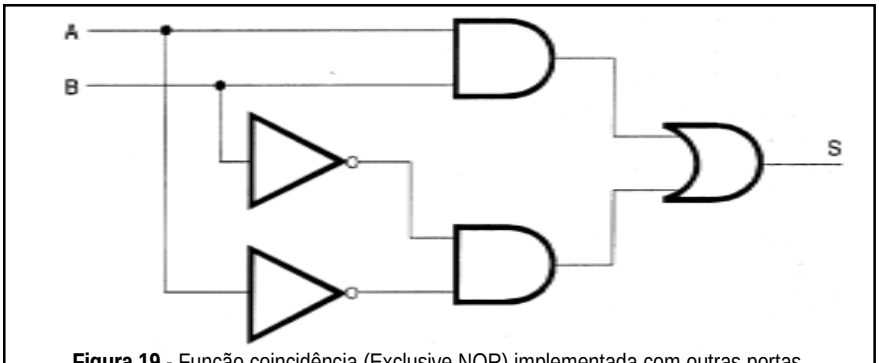

Figura 19 - Função coincidência (Exclusive NOR) implementada com outras portas.

5. Propriedades diversas:

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

6. Teoremas de De Morgan:

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

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

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

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

2.12 - Fazendo tudo com portas NAND

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

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

Inversor

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

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

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

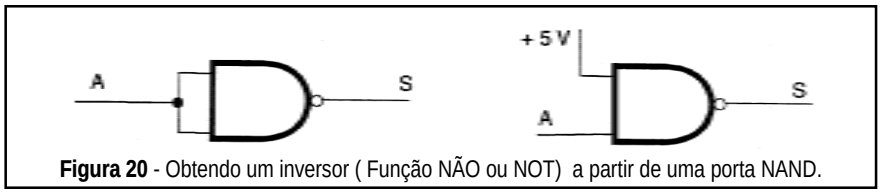

Figura 20 - Obtendo um inversor (Função NÃO ou NOT) a partir de uma porta NAND.

2.13 - Conclusão

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

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

QUESTIONÁRIO

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

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

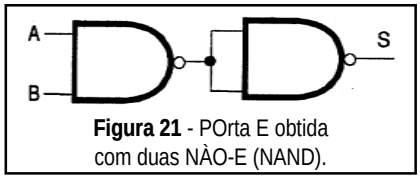

Figura 21 - POrta E obtida com duas NÀO-E (NAND).

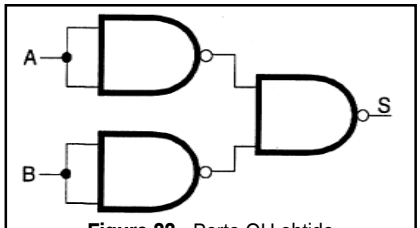

Figura 22 - Porta OU obtida com duas NÃO-E (NAND).

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

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

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

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

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

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