AULA 3 – PORTAS LÓGICAS E INTRODUÇÃO À ÁLGEBRA BOOLEANA

Iremos utilizar a Álgebra Booleana que é uma ferramenta matemática relativamente simples que nos permite descrever relações entre as saídas dos circuitos lógicos e suas entradas.

Estudaremos as portas lógicas, os circuitos lógicos mais básicos, outros um pouco mais complexos contruídos a partir da combinação de portas lógicas, podem ser descritos e analisados a partir da Álgebra Booleana.

Enfim a Álgebra Booleana é uma ferramenta usada na descrição, análise, no projeto e implementação de circuitos digitais.

CONSTANTES E VARIÁVEIS BOOLEANAS:

A principal diferença entre a álgebra convencional e a álgebra boolenana é que a última trabalha somente com duas variáveis (“0” ou “1”) que representam estados lógicos.

O nível booleano “0” representa a tensão por exemplo nos terminais de saída de um circuitos lógico que medida com um multímetro pode estar entre 0 a 0,8 Volts, enquanto o nível “1” representa uma saída entre 2 a 5 Volts.

Então as variáveis não representam números, mas sim estados lógicos, podemos utilizar uma letra para representar o estado lógico de uma entrada ou saída em questão.

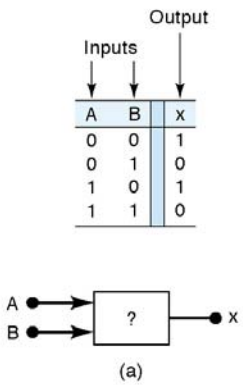
A = 0 ou A = 1

A facilidade da álgebra booleana em relação a álgebra convencional não reside apenas nos dois únicos estados, como também no fato de não existir frações, decimais, números negativos, raízes quadradas, raízes cúbicas, logaritmos, números imaginários e assim por diante, a álgebra booleana tem fundamentalmente três operações básicas OR (ou), AND (E) e NOT (NÃO), chamadas operações lógicas.

Estas operações lógicas, na prática são portas lógicas construídas de diodos, transistores e resistores interligados.

TABELA VERDADE:

A tabela verdade nos mostra o estado que a saída assumirá (coluna da direita) em função das combinações dos estados que as entradas (colunas da esquerda) tiverem, ou seja, descreve o comportamento do bloco lógico analisado.



(a)

A	B	C	x
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(b)

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

(c)

(a) Tabela verdade de um circuito lógico com duas entradas;
(b) Tabela verdade de um circuito lógico com três entradas;
(c) Tabela verdade de um circuito lógico com quatro entradas;

A OPERAÇÃO OR (OU) E A PORTA OR (OU):

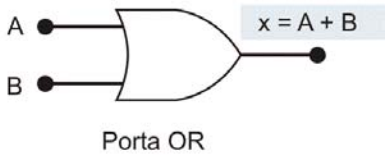
Esta é a primeira operação lógica que iremos estudar, nesta operação basta que uma única entrada esteja em nível lógico “1” para que a saída assuma nível alto, somente quando todas as entradas estiverem em nível lógico baixo é que a saída terá nível “0”.

Abaixo segue a tabela verdade, bem como o símbolo da porta OU de duas entradas:

OR

A	B	x = A + B
0	0	0
0	1	1
1	0	1
1	1	1

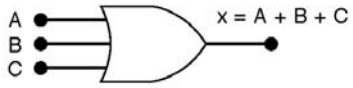
(a)



Porta OR

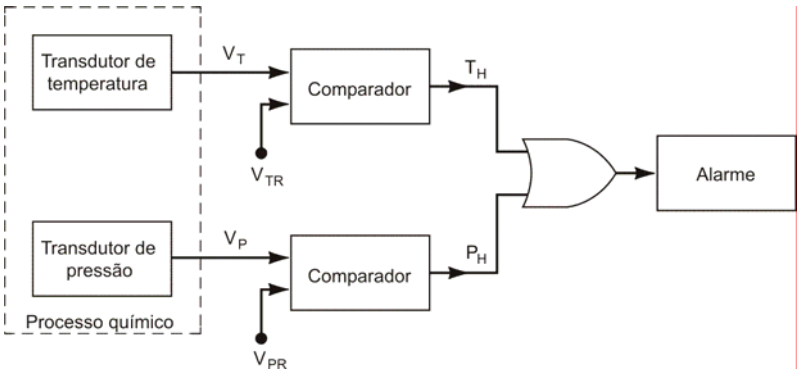
(b)

Pode-se encontrar porta lógica OU com mais de duas entradas como por exemplo com três entradas:



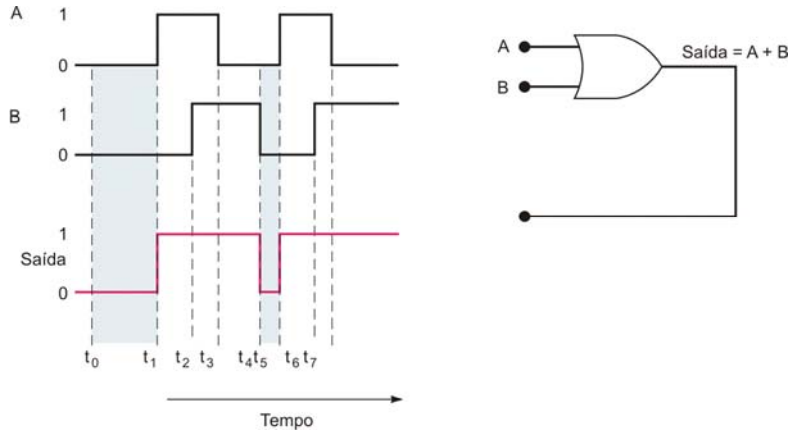
A	B	C	x = A + B + C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Em situações práticas podemos considerar por exemplo um controle industrial onde nem a temperatura, nem a pressão em um sistema podem ultrapassar um valor pré-ajustado (valor referência), em situações como estas podemos utilizar a lógica OU:

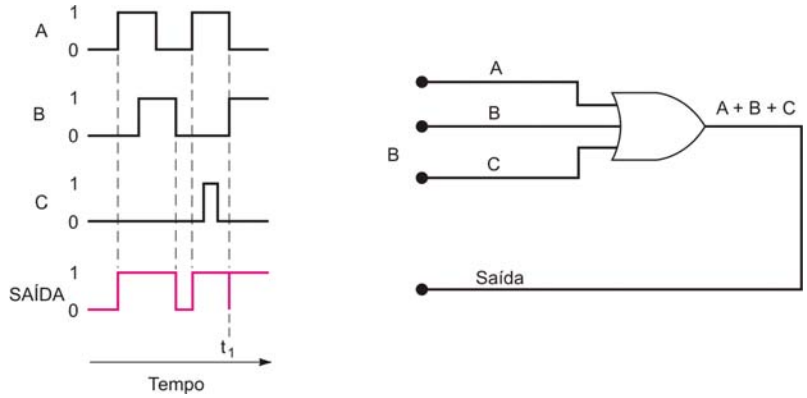


Existem dois transdutores que convertem a temperatura e pressão para sinais elétricos que serão posteriormente comparados com valores de referência (valores limite), se estes valores forem maiores teremos nas saídas TH ou PH níveis lógicos “1”, portanto basta que ou a temperatura ou a pressão suba a níveis indesejados que o alarme irá ser acionado em qualquer dos casos.

Analisando os níveis lógicos podemos verificar pela figura na seqüência que só teremos nível lógico baixo na saída quando todas as entradas estiverem em nível baixo, basta que uma única esteja em nível alto para que a saída seja levada a nível “1”.



Podemos verificar o mesmo comportamento para a porta lógica OU com três entradas:



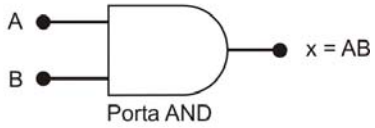
A OPERAÇÃO AND (E) E A PORTA AND (E):

Na segunda operação booleana básica E somente quando todas as entradas estiverem em nível alto a saída ficará em nível “1”, bastando apenas uma entrada estar em nível baixo para que a saída vá a “0”, abaixo podemos ver a simbologia e a tabela verdade para uma porta lógica E de duas entradas:

AND

A	B	x = A · B
0	0	0
0	1	0
1	0	0
1	1	1

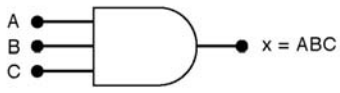
(a)



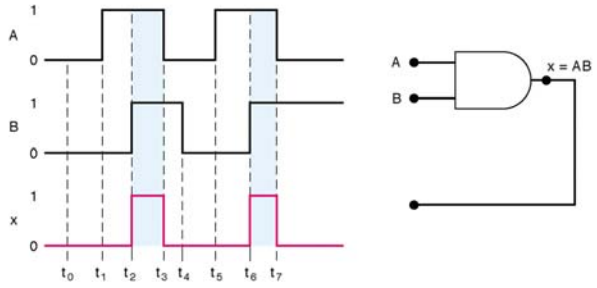
(b)

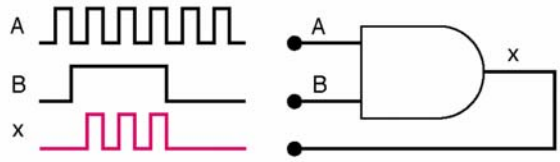
Poderíamos imaginar chaves conectadas em série, sendo que somente quando todas as chaves estiverem fechadas (nível “1”) é que a saída terá tensão (nível alto), bastando apenas 1 delas estar aberta (nível “0”) para que não haja tensão na saída (nível baixo). Veja abaixo a porta E com três entradas:

A	B	C	x = ABC
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



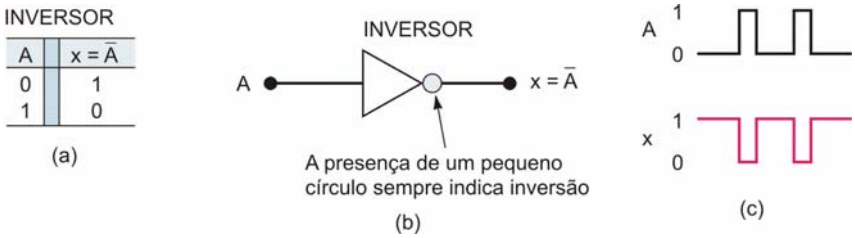
Similar ao funcionamento da porta OU, só que aqui só haverá nível alto se todas as entradas estiverem em nível alto:





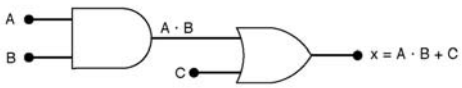
A OPERAÇÃO NOT (NÃO) E A PORTA NOT (NÃO):

Esta operação lógica é diferente das anteriores porque pode ser realizada somente com uma única variável, o que ela faz é justamente inverter o nível lógico na saída do que recebeu em sua entrada, portanto se uma porta lógica NÃO receber na entrada nível lógico “0” em sua saída teremos “1” e vice-versa. Veja abaixo:



REPRESENTANDO OS CIRCUITOS LÓGICOS NA FORMA ALGÉBRICA:

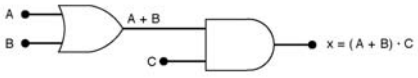
Qualquer circuito lógico pode ser representado pelas três operações lógicas básicas, OU, E e NÃO, assim como podemos fazer a representação literal de qualquer circuito lógico, veja o exemplo abaixo:



A saída X é o resultado da operação lógica OU entre a entrada C e o resultado da operação lógica E entre as entradas A e B:

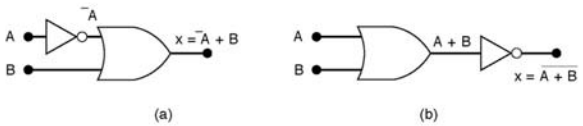
X = A.B+C

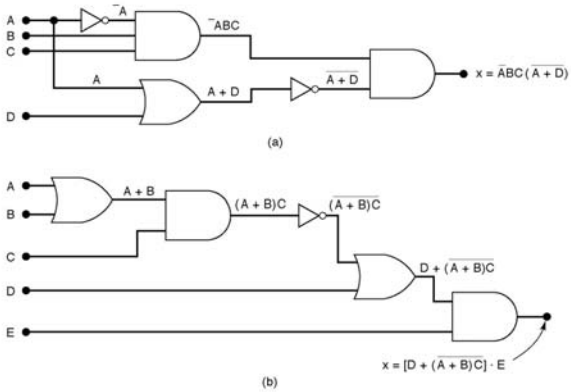
Sempre que tivermos na expressão uma operação AND e OR, a operação AND deverá ser realizada antes, salvo a representação de parênteses, como na álgebra convencional, se o circuito fosse como descrito abaixo, nesta situação deve-se realizar a operação OR primeiro:



X = (A+B).C

Além das operações AND e OR podemos utilizar o inversor (NOT) também, sempre que uma variável é conectada a uma porta NOT representa-se uma barra ou aspa simples sobre ela, isto denota a inversão:





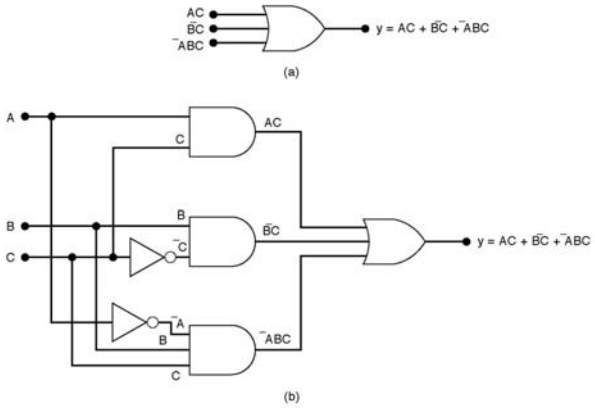
Assim como na álgebra convencional podemos determinar o valor de X em função de determinados valores na entrada, imagine que no circuito (a) da figura da direita tivéssemos as seguintes entradas:
A = 0, B = 1, C = 1, D = 1, a saída poderia ser calculada da seguinte forma:

X=A'.B.C.(A+D)' => X=(0)'.1.1.(0+1)' => X=1.1.1.(1)' => X=1.1.1.0 => X=0

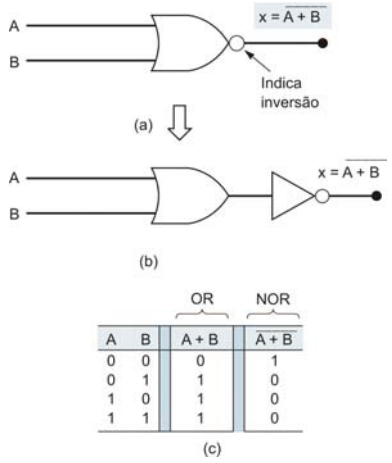
RESUMO:
1º Realize todas as inversões de termos simples;
2º Realize todas as operações dentro de parênteses;
3º Realize as operações AND antes das operações OR;
4º Se uma expressão tiver barra sobre ela, encontre o resultado da expressão para somente depois inverte-la.

Podemos também a partir de uma expressão booleana implementar o circuito lógico a partir desta expressão:

Y = A.C+B.C'+A'.B.C

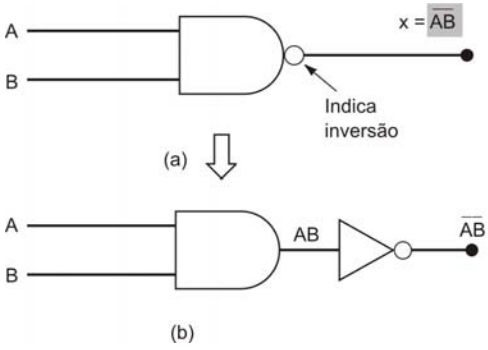


Existe no mercado também portas resultantes de associações de portas lógicas básicas, mas que são encontradas prontas em circuitos integrados como o caso das portas NOR (OR com NOT) e NAND (AND com NOT):



		OR	NOR
A	B	A + B	$\overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

(c)



		AND	NAND
A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

(c)

Se tivermos a situação de dois inversores ligados na seqüência, o inverso da inversão é a própria expressão sem inversão: