1.Portas Lógicas

1.1 - PORTAS E OPERAÇÕES LÓGICAS

Uma *porta logica* (gate) é um circuito eletrônico, portanto uma peça de hardware, que se constitui no elemento básico e mais elementar de um sistema de computação. Grande parte do hardware do sistema é fabricado através da adequada combinação de milhões desses elementos, como a UCP, memórias principal e cache, interfaces de E/S e outros..

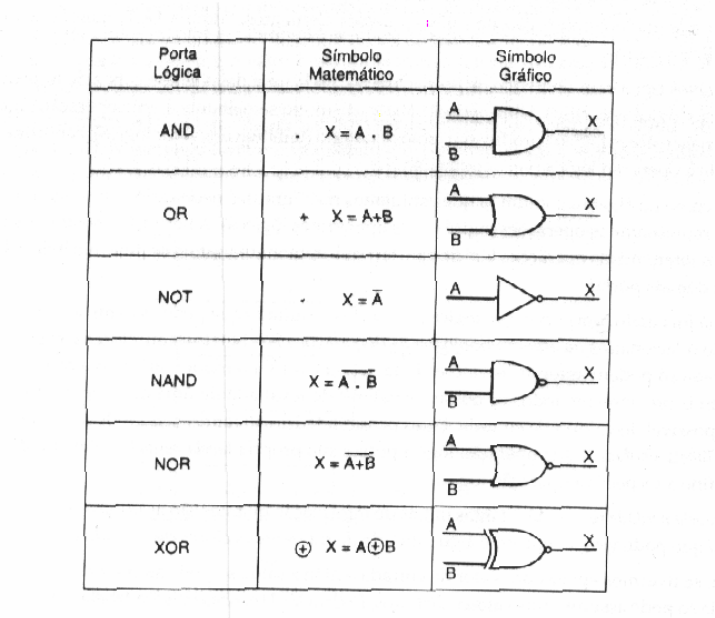
Há diversos tipos bem definidos de portas lógicas, cada uma delas sendo capaz de implementar uma operação ou função lógica específica. Uma *operação lógica* (de modo semelhante a uma operação algébrica) realizada sobre um ou mais valores lógicos produz um certo resultado (também um valor lógico), conforme a regra definida para a específica operação lógica.

Assim como na álgebra comum, é necessário definir símbolos matemáticos e gráficos para representar as operações lógicas (e os operadores lógicos). A figura abaixo mostra os símbolos matemáticos e gráficos referentes às operações lógicas (portas) que iremos analisar neste item.

Como já foi citado, uma operação lógica produz um resultado que pode assumir somente dois valores, 0 ou l, os quais são relacionados na álgebra booleana às declarações FALSO (F=0) ou VERDADEIRO (V=1). Se as variáveis de entrada só podem assumir os valores F (falso) ou 0 ou V (verdade) ou 1 e se o resultado também, então podemos definir previamente todos os possíveis valores de resultado de uma dada operação lógica, conforme a combinação possível de valores de entrada. Essas possibilidades são representadas de forma tabular e chama-se o conjunto de *Tabela Verdade.* Cada operação lógica possui sua própria *tabela verdade,* estabelecida de acordo com a regra que define a respectiva operação lógica.

Uma *tabela verdade* tem, então, tantas linhas de informação quantas são as possíveis combinações de valores de entrada, o que pode variar conforme a quantidade de diferentes valores de entrada que se tenha.

Assim, se tivermos apenas um valor de entrada, então a saída só pode assumir dois valores (já que a variável de entrada só pode assumir dois valores distintos, (F=0 ou V=1) e, nesse caso, a tabela verdade teria duas linhas, uma para a entrada igual a 0 e outra para entrada igual a 1. Se, por outro lado, fossem definidas duas entradas, então haveria quatro possíveis combinações dos valores de entrada (00, 01, l0 e 11), pois $2^2 = 4$ e a tabela verdade possuiria quatro linhas e assim por diante.



Porta Lógica	Símbolo Matemático	Símbolo Gráfico
AND	. X = A . B	A, B, X
OR	+ X = A+B	A, B, X
NOT	- $X = \overline{A}$	A, X
NAND	$X = \overline{A \cdot B}$	A, B, X
NOR	$X = \overline{A+B}$	A, B, X
XOR	⊕ X = A⊕B	A, B, X

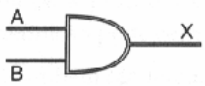
Figura 1

De um modo geral, a tabela verdade de uma dada operação Lógica possui 2^n linhas ou combinações de valores de entrada, sendo o igual à quantidade de elementos de entrada.

1.1.1 - **Operação Lógica ou** Porta **AND (E)**

A porta AND é definida como sendo o elemento (ou operação lógica) que produz um resultado verdade (=1) na saída, *se* e *somente se todas as entradas forem verdade.* Essa definição pode ser expressa pela tabela verdade e simbolos mostrados na figura 2.

Uma porta Lógica AND pode ter várias utilidades na fabricação de um sistema digital, algumas das quais



X = A . B ou X = AB

Entrada		Saída
A	B	X = AB
0	0	0
0	1	0
1	0	0
1	1	1

Figura 2

Serão mostradas em oportunidades. Entre elas, uma das mais importantes pode ser a de ativação de uma linha de dados para movimentar bits de um registrador (ou células) para outro.

A figura 3 mostra um exemplo de aplicação do circuito lógico (ou porta) AND como elemento de controle em transferências de dados. Para cada bit do registrador, um sinal da unidade de controle (UC) serve de entrada, juntamente com o sinal correspondente ao bit do registrador origem (registrador A na figura 3); quando o sinal da unidade de controle for igual a 1, a combinação dos sinais de entrada produz na saída um valor, sempre igual ao do bit do registrador de entrada, o qual será armazenado no registrador de destino (registrador B na figura 3). Com isso, obteve-se a transferência dos bits do registrador origem para o registrador de destino durante o período em que a linha da UC esteve com o bit 1 ativo. No item 1.3 serão abordadas outras aplicações de utilização de portas lógicas.

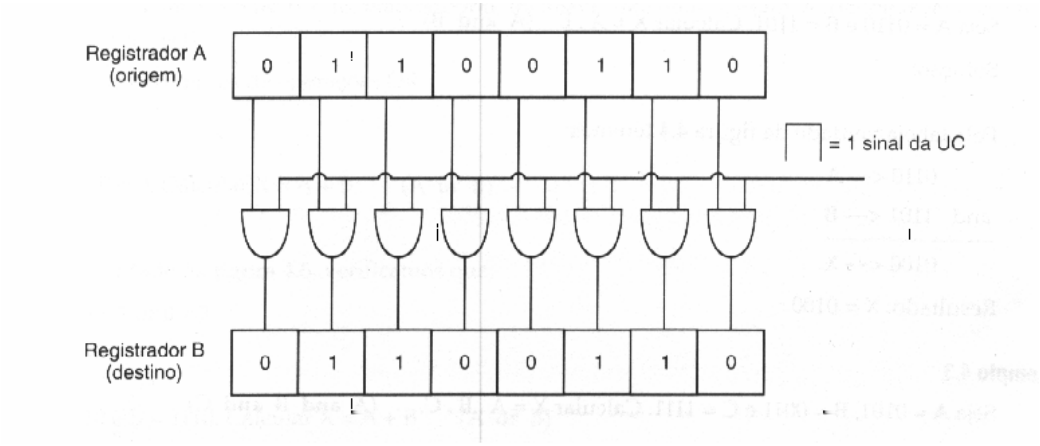


Figura 3

A combinação dos dois valores de entrada que estão exemplificados na figura 4 e que produziram resultados de acordo com a tabela verdade apresentada na figura 2 constitui, na realidade, uma operação lógica AND porque foi realizada com o operador lógico AND.

Operações lógicas AND podem ser realizadas para satisfazer um determinado requisito de hardware, como veremos mais pra frente - Expressões Lógicas - Aplicações de Portas, ou para atender a uma especificação de um programador em um programa. Para tanto, a maioria dos computadores possui uma instrução de máquina AND em seu conjunto de instruções, bem como muitas linguagens de programação de alto nível implementam essa função para, como já mencionamos, atender a determinadas condições de programa.

Por exemplo, a lógica de um determinado programa pode estabelecer:

• Ler X, Y e Z

• T=X+Y

•R =Z+X

• SE (T> 6 E(AND) R < 10)

• ENTÃO IMPRIMIR T

• SENÃO IMPRIMIR R

O trecho de programa exemplificado é ridículo mas exprime um exemplo do uso de AND na composição uma condição a ser satisfeita (ser VERDADE) para que uma determinada ação seja realizada ou não.

No exemplo verificamos que o valor de T somente será impresso se ambas as condições forem verdadein T> 6 e R < 10. O operador AND uniu ambas as afirmações.

Operações lógicas AND também podem ser realizadas com valores constituídos de vários algarismos UAL - Unidade Aritmética e Lógica realiza tal tipo de operação).

Exemplo 1.1

Seja A=l e B=0. Calcular X=A.B (A and **B)**

Solução:

Analisando a tabela verdade da figura 2 verificamos que:

X = 0, pois: **1 and** 0 = 0

Exemplo 1.2

Seja A=0ll0 e B=1101.CalcularX=A.B (A **and B)**

Solução:

Pela tabela verdade da figura 2 teremos:

0110 ← A

and 1101 ← B

0100 ← x

Resultado: X = 0100

Exemplo 1.3

Seja A=0101, B=00ll e C=1111.Calcular X=A.B.C. (A **and B and C)**

Solução:

O resultado é obtido através da realização das operações em duas etapas. Na primEira parte, calcula-se A.B e, em seguida, o resultado parcial obtido é combinado com C em outra operação lógica AND, sempre utilizando as combinações de entrada e os resultados definidos na tabela verdade da figura 2

.

	0101 ← A	0001 ← T
and	0011 ← B	1111 ← C
	0001 ← T	0001 ← X

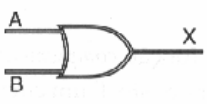
Resultado: X = 0001

1.1.2 - **Operação Lógica ou** Porta **OR (OU)**

A porta OR é definida para produzir um resultado verdade (=1) na sua saída, se pelo menos uma das entradas for verdade. Esta definição pode ser expressa pela tabela verdade e símbolos mostrados na figura 4.

Operações lógicas OR também são largamente utilizadas em lógica digital ou na definição de condições em comandos de decisão de certas linguagens de programação. No exemplo de trecho de programa mostrado no item anterior, pode-se modificar a condição do SE, tornando-a mais flexível:

- SE (1 > 6 OU (OR) R < 10)
- ENTÃO IMPRIMIR T
- SENÃO IMPRIMIR R



X = A+B

Entrada		Saída
A	B	X = A+B
0	0	0
0	1	1
1	0	1
1	1	1

Figura 4

Nesse caso, para T ser impresso, basta que uma das duas condições seja verdadeira (não *ambas,* como determina a operação AND), **ou** T> 6 ou R < 10. Não importa, inclusive, que ambas sejam verdadeiras, embora baste que apenas uma delas o seja.

Vejamos alguns exemplos de operações OR:

Exemplo 1.4

Seja A=l e B=0.Calcular X=A+B (A **or** B)

Solução:

Pela tabela verdade da figura 4, verificamos que:

X = 1, porque 1 or 0 = 1

Exemplo 1.5

Seja A = 0110 e B 1110. Calcular X A + B

Solução:

0110 ← A
or 1110 ← B
1110 ← X

Resultado: x = 1110

Exemplo 1.6

Seja A=1100, B=llll e C=0001.CalcularX=A+B+C.(A or **B** or **C)**

Solução:

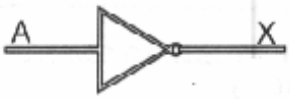
O cálculo também é realizado em duas etapas, utilizando-se a tabela verdade da figura 4.

1100 ← A	1111 ← T
or 1111 ←B	0001 ← C
1111 ← X	1111 ← X

Resultado: x = 1111

1.1.3 - Operação Lógica NOT (Inversor)

A operação lógica NOT é também chamada de *inversor* ou função *complemento*. Ela inverte o valor de um sinal binário colocado em sua entrada, produzindo na saída o valor oposto. É um circuito lógico que requer apenas um valor na entrada (um outro circuito lógico - "buffer" - também requer apenas um valor de entrada, como será mostrado mais a frente). A figura 5 mostra um circuito inversor, bem como os símbolos utilizados e a respectiva tabela verdade.



$X = \overline{A}$

Entrada	Saída
A	$X = \overline{A}$
0	1
1	0

Figura 5

É interessante observar que a conexão de dois circuitos inversores em série produz, na saída, um resultado de valor igual ao da entrada. Ou seja, um duplo complemento restaura o valor original.

É possível também encontrar na literatura o apóstrofo (')para representar um circuito NOT, como exemplificado a seguir:

NOT A = A'

No entanto, adotaremos a barra em cima do valor de entrada

NOT A = $\overline{A}$

também mostrado na figura 5, simbologia definida pela ANSI (American National Standards Institute).

Uma das aplicações mais comuns do circuito inversor ou NOT é justamente em operações aritméticas em ponto fixo, quando se usa aritmética de complemento (complemento a 1 ou complemento a 2).

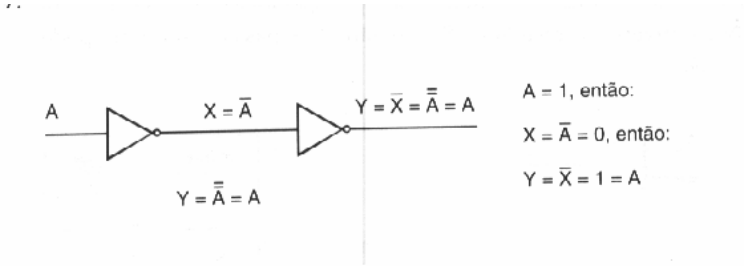


Figura 6

A seguir, vamos apresentar alguns exemplos de operações com circuitos inversores.

Exemplo 1.7

Seja A = 0. Calcular X = A

Solução:

Utilizando a tabela verdade da figura 5, temos: X = 1 porque 0 = $\overline{1}$.

Exemplo 1.8

Seja A = 10011. Calcular X = $\overline{A}$

Solução:

Utilizando a tabela verdade da figura 5, obtemos o valor de X invertendo o valor de cada algarismo.

10011 ← A

01100 ← inverso de A, bit a bit

X = 01100 = A

Exemplo 1.9

Seja A = l00l0 e B = llll0. Calcular X= $\overline{\mathbf{A.B}}$

Solução:

Trata-se da realização de duas operações lógicas em seqüência. Primeiro, a operação lógica AND e, em seguida, obtém-se o inverso do resultado.

Pela tabela verdade da figura 2, temos:

10010 ← A

and 11110 ← B
10010 ← T

Inverte-se T, usando a tabela verdade da figura 5:

10010 ← T → A.B

01101 ← $\overline{T}$ → **A . B**

Resultado: X = $\overline{T}$ = **A. B**