Funções Lógicas e Portas Lógicas

Uma introdução ao sistema matemático e análise de circuitos lógicos, conhecido como Álgebra de Boole;

Serão vistos os blocos básicos e suas equivalências.

Histórico

- Em meados do século XIX o matemático inglês George Boole desenvolveu um sistema matemático de análise lógica;
- Em meados do século XX, o americano Claude Elwood Shannon sugeriu que a Álgebra Booleana poderia ser usada para análise e projeto de circuitos de comutação;
- Nos primórdios da eletrônica, todos os problemas eram solucionados por meio de sistemas analógicos;
- Com o avanço da tecnologia, os problemas passaram a ser solucionados pela eletrônica digital;
- Na eletrônica digital, os sistemas (computadores, processadores de dados, sistemas de controle, codificadores, decodificadores, etc) empregam um pequeno grupo de circuitos lógicos básicos, que são conhecidos como portas **e, ou, não** e **flip-flop**;
- Com a utilização adequadas dessas portas é possível implementar todas as expressões geradas pela álgebra de Boole.

Álgebra Booleana

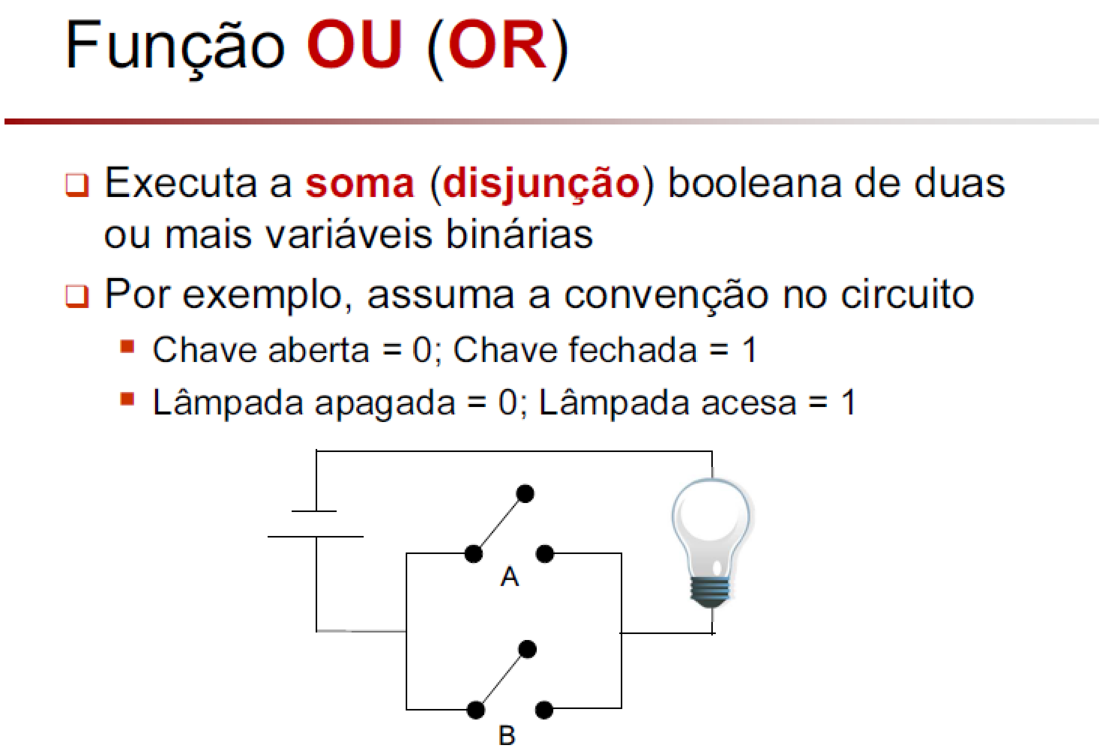
Na álgebra de Boole, há somente dois estados (valores ou símbolos) permitidos:
Estado 0 (zero)
Estado 1 (um)

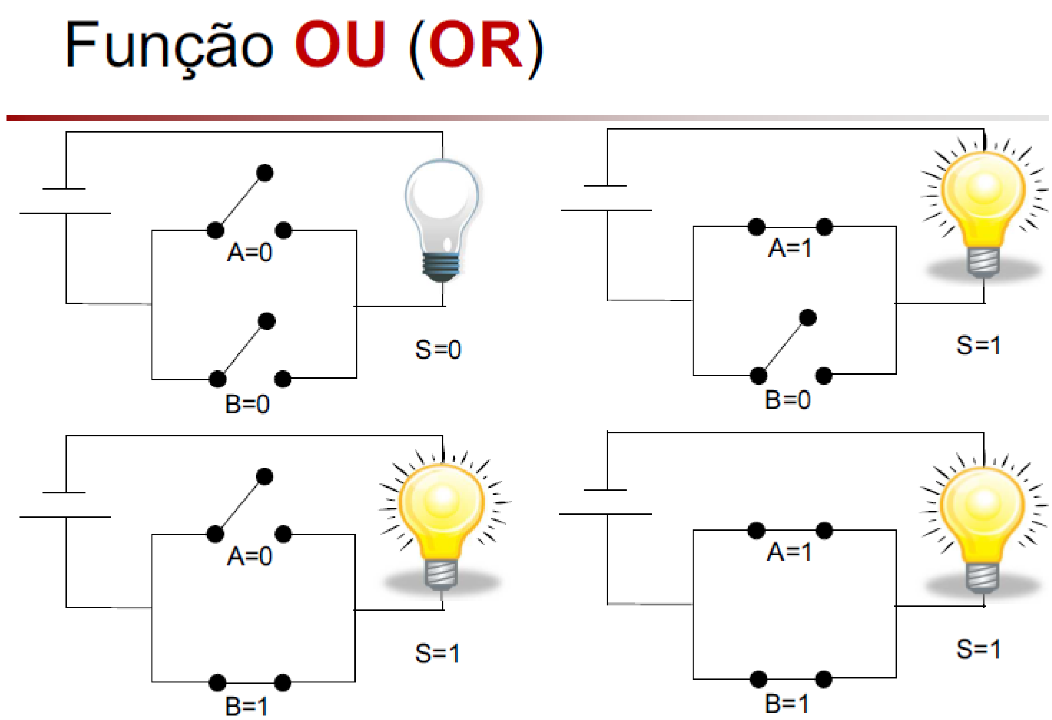
Em geral:
O estado zero representa não, falso;
O estado um representa sim, verdadeiro.

Assim, na álgebra booleana, se representarmos por 0 uma situação, a situação contrária é representada por 1.
Portanto, em qualquer bloco (porta ou função) lógica somente esses dois estados (0 ou 1) são permitidos em suas entradas e saídas.
Uma variável booleana também só assume um dos dois estados permitidos (0 ou 1).

Portas Lógicas e Álgebra Booleana

A álgebra booleana é a ferramenta fundamental para descrever a relação entre as saídas de um circuito lógico e suas entradas através de uma equação (expressão booleana). Existem três operações básicas: OR (OU), AND (E) e NOT (NÃO).

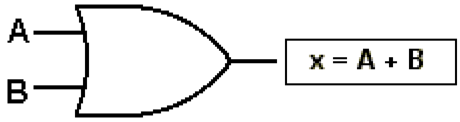




A figura 1.3 ilustra a tabela verdade, juntamente com o desenho que representa a operação lógica OR.

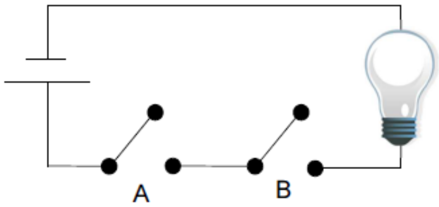
Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.1: Porta OR (OU)

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



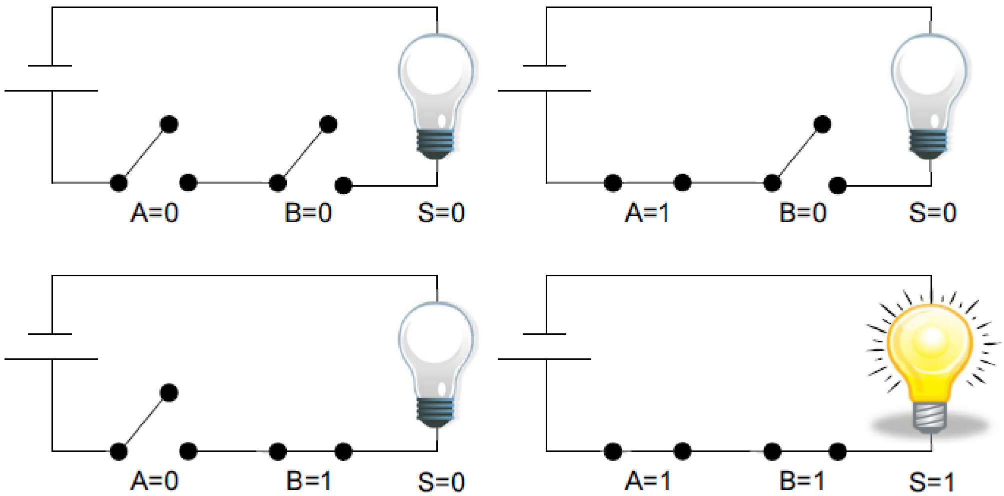
Função E (AND)

- Executa a **multiplicação** (**conjunção**) booleana de duas ou mais variáveis binárias
- Por exemplo, assuma a convenção no circuito
  - Chave aberta = 0; Chave fechada = 1
  - Lâmpada apagada = 0; Lâmpada acesa = 1



Função E (AND)

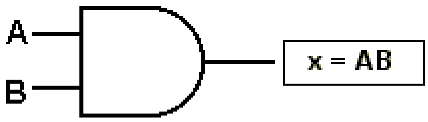
- Situações possíveis:

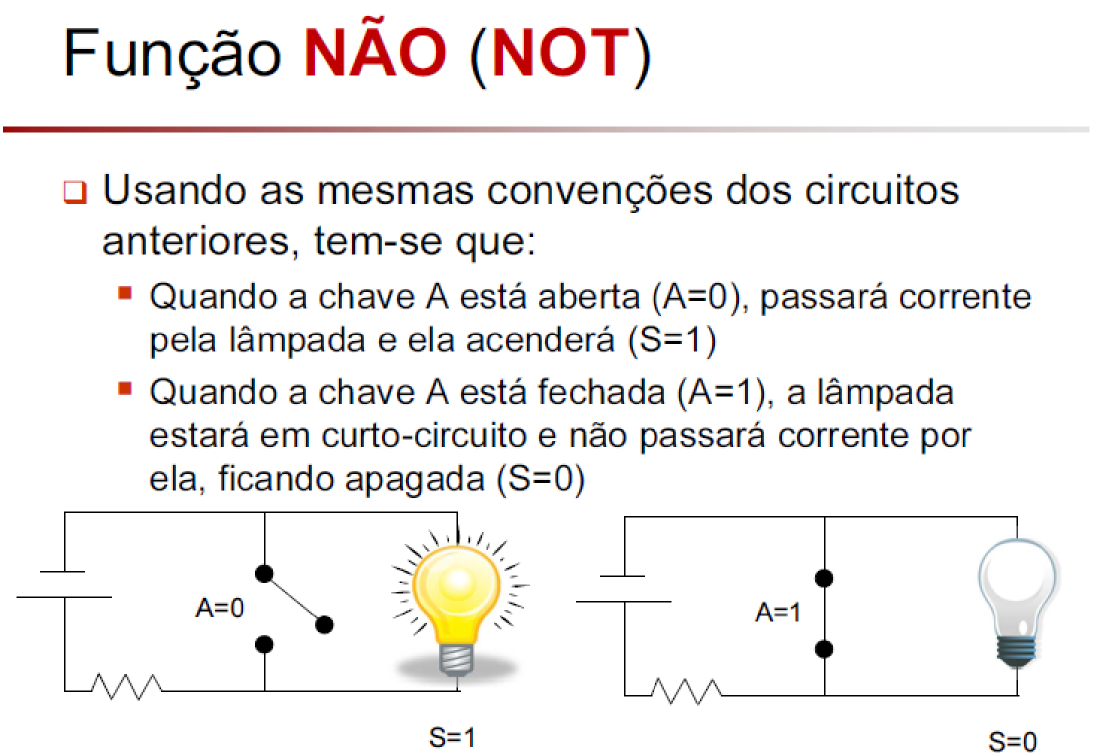


A figura 1.4 ilustra a tabela verdade, juntamente com o desenho que representa a operação lógica AND.

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.2: Porta AND (E)

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

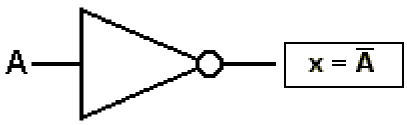




A figura 1.5 ilustra a tabela verdade, juntamente com o desenho que representa a operação lógica NOT.

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.3: Porta NOT (INVERSOR)

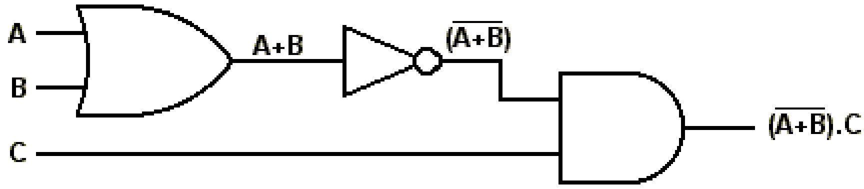
A	x
0	1
1	0



1.1.1. Descrevendo Circuitos Lógicos Algebricamente

Qualquer circuito lógico pode ser descrito usando as portas AND, OR e NOT. Essas três portas são os blocos básicos na construção de qualquer sistema digital, visto pelo exemplo na figura 1.6.

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.4: Circuito Lógico e sua Expressão Lógica



A) EXERCÍCIOS:

Treinar e entender a montagem das expressões lógicas dada no quadro.

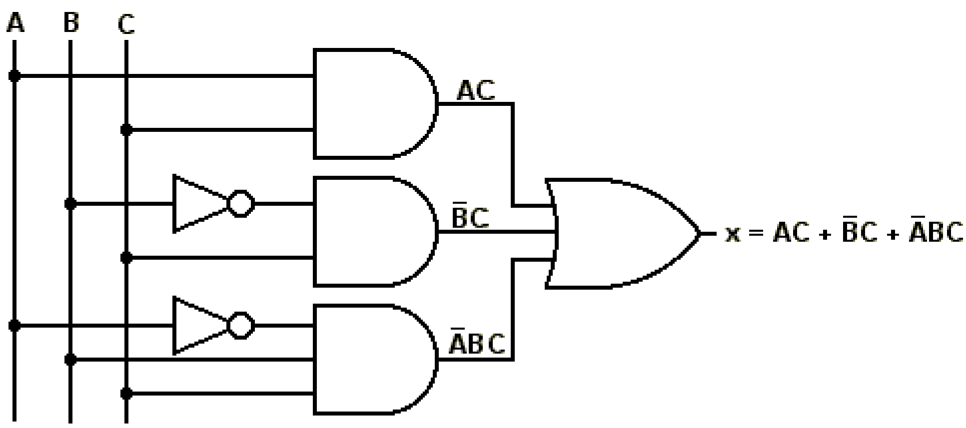
****

1.1.2. Implementação de Circuitos Lógicos a partir de Expressões Booleanas

Pode-se usar a expressão booleana para gerar o circuito lógico. Um exemplo é ilustrado na figura 1.7.

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.5**:** Expressão Lógica e seu Circuito Lógico

$$\boxed{x = AC + \bar{B}C + \bar{A}BC}$$



B) EXERCÍCIOS:

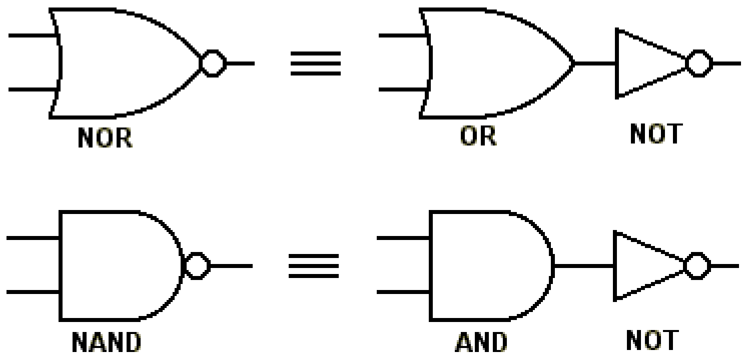
Treinar e entender a montagem das expressões lógicas dada no quadro.

****

1.1.3. Portas NOR e NAND

Outros tipos de portas lógicas existentes são as portas NOR e NAND, que na verdade são combinações das portas OR, AND e NOT, visto na figura 1.8.

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.6: Portas NOR e NAND



1.2. Teoremas da Álgebra de Boole

Esses teoremas, aplicados na prática, visam simplificar as expressões booleanas e consequentemente os circuitos gerados por estas expressões.

1.2.1. Teoremas Booleanos

1) $x \cdot 0 = 0$
2) $x \cdot 1 = x$
3) $x \cdot x = x$
4) $x \cdot \overline{x} = 0$
5) $x + 0 = x$
6) $x + 1 = 1$
7) $x + x = x$
8) $x + \overline{x} = 1$
9) $x + y = y + x$
10) $x \cdot y = y \cdot x$
11) $x + (y + z) = (x + y) + z = x + y + z$
12) $x(yz) = (xy)z = xyz$
13) $x(y + z) = xy + xz$
14) $x + xy = x$
15) $x + \overline{x}y = x + y$

1.2.2. Teoremas de DeMorgan

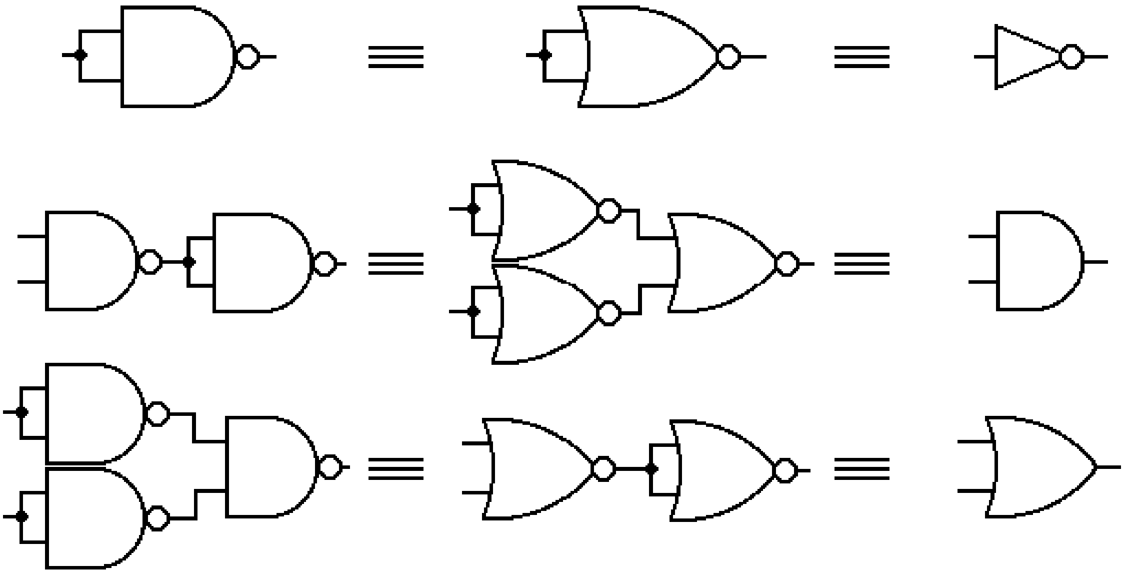
$$(\overline{x + y}) = \overline{x} \cdot \overline{y}$$

$$(\overline{x \cdot y}) = \overline{x} + \overline{y}$$

1.3. Universalidade das Portas NAND e NOR

Qualquer expressão lógica pode ser implementada usando apenas portas NAND ou portas NOR. Isso porque pode-se representar portas OR, AND ou NOT usando apenas portas NAND ou NOR, como visto na figura 1.9.

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.7: Uso de PORTAS NAND para implementar outras funções booleanas



1.4. Simplificação de Circuitos Lógicos

Depois de encontrada a expressão de um circuito lógico, pode-se reduzi-la para uma forma mais simples. A intenção é diminuir o número de variáveis nessa expressão, o que significa diminuir o número de portas lógicas e conexões em um circuito lógico.

1.4.1. Simplificação Algébrica

A simplificação algébrica é feita com o uso dos teoremas da álgebra booleana e de DeMorgan. Exemplo:

$$x = A.B.C + A.\bar{B}.(\overline{\bar{A}.\bar{C}})$$

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

$$x = A.B.C + A.\bar{B}.A + A.\bar{B}.C$$

$$x = A.B.C + A.\bar{B} + A.\bar{B}.C$$

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

$$x = A.C + A.\bar{B}$$

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

1.5. Projetando Circuitos Lógicos

Passos para o projeto completo de um circuito lógico:

a) **Montar a tabela-verdade:**

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

b) **Analisar a saída:**

Quando qualquer entrada de uma porta OR for "1" então a saída será "1". Então pode-se deduzir que a saída x é uma operação OR de todos os casos em que a saída x é "1". Cada caso corresponde a uma operação lógica AND com todas as variáveis de entrada.

$$x = \bar{A}.B.C + A.\bar{B}.C + A.B.\bar{C} + A.B.C$$

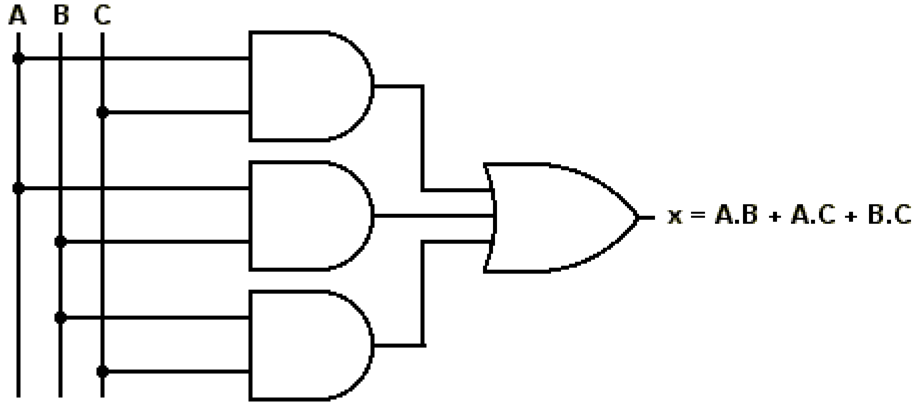
c) **Simplificar a expressão lógica obtida:**

A expressão pode ser reduzida a um número menor de termos se aplicado os teoremas booleanos e de DeMorgan.

$$x = A.B + A.C + B.C$$

d) **Implementar o circuito, visto na figura 1.10, através da expressão lógica:**

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.8: Circuito lógico final



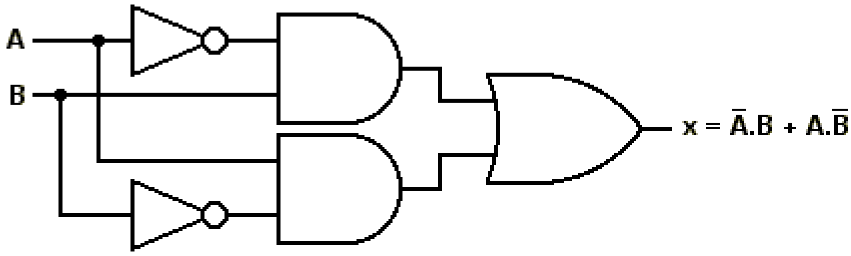
1.6. Outras Portas Lógicas

Outras portas lógicas importantes existentes são as XOR e XNOR.

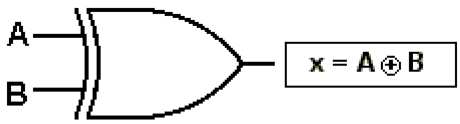
Circuito XOR

A figura 1.11 representa a porta lógia OU exclusivo (XOR).

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.9: Porta XOR (OU-Exclusivo)



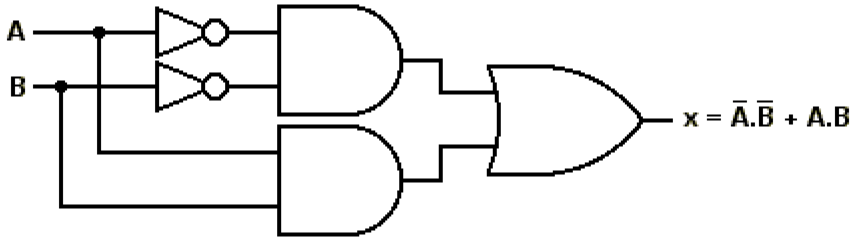
A	B	x
0	0	0
0	1	1
1	0	1
1	1	0



1.6.2. Circuito XNOR

A figura 1.12 representa a porta lógia NOU exclusivo (XNOR).

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.10: Porta XNOR (NOU-Exclusivo)



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