LÓGICA PROPOSICIONAL E ÁLGEBRA BOOLEANA

Thiago Felski Pereira

Adaptado do livro Sistemas Digitais (Tocci)

CONSTANTES E VARIÁVEIS BOOLEANAS

- A álgebra booleana permite apenas dois valores: 0 e 1.
  - **Lógica 0** pode ser: *falso*, *desligado*,*baixo*, *não*, *interruptor aberto*.
  - **Lógica 1** pode ser: *verdadeira*, *ligado*, *alto*, *sim*, *interruptor fechado*.

- Três operações básicas:
  - OR, AND e NOT.

CONSTANTES E VARIÁVEIS BOOLEANAS

Lógico 0	Lógico 1
Falso	Verdadeiro
Desligado	Ligado
BAIXO	ALTO
Não	Sim
Aberto	Fechado

TABELAS-VERDADE

- A tabela-verdade descreve a relação entre as entradas e as saídas de um circuito lógico.

- O número de colunas corresponde ao número de entradas.

Uma tabela de duas entradas teria 2^2 = quatro linhas.

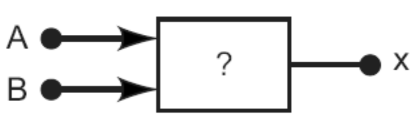
Uma tabela de três entradas teria 2^3 = oito linhas.

TABELAS-VERDADE

Exemplos de tabela-verdade com duas, três e quatro entradas.

Entradas | Saída

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



(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)

OPERAÇÕES OR ("OU") COM PORTAS OR

- A expressão booleana para a operação **OR** é:

$X = A + B$ — Leia "*X* equivale a *A* ou *B*"

O sinal + não se aplica para soma, mas sim para operações OR.

- A operação **OR** é semelhante à adição, e quando A = 1 e B = 1, produz:

1 + 1 = 1 *não* 1 + 1 = 2

Na expressão booleana x = 1 + 1 + 1 = 1…
X é verdade (1) quando *A é verdadeiro* (1) OU *B é verdadeiro* (1) OU C *é verdadeiro* (1).

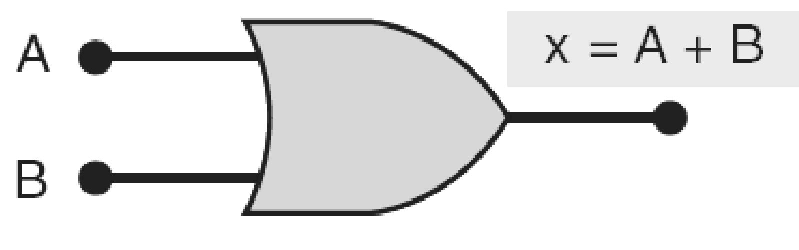
OPERAÇÃO OR COM PORTA OR

Uma porta OR é um circuito com uma ou mais entradas, cuja saída é igual à combinação OR das entradas.

Tabela-verdade símbolo de circuito para duas entradas da porta OR.

OR

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

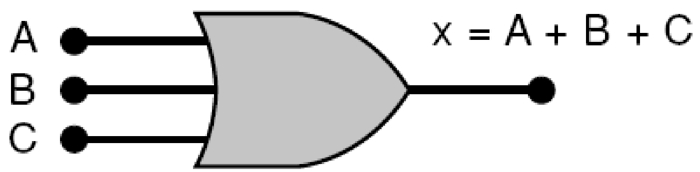


Porta OR

OPERAÇÃO OR COM PORTA OR

A porta **OR** é um circuito com duas ou mais entradas, cuja saída é igual a combinação OR das entradas.

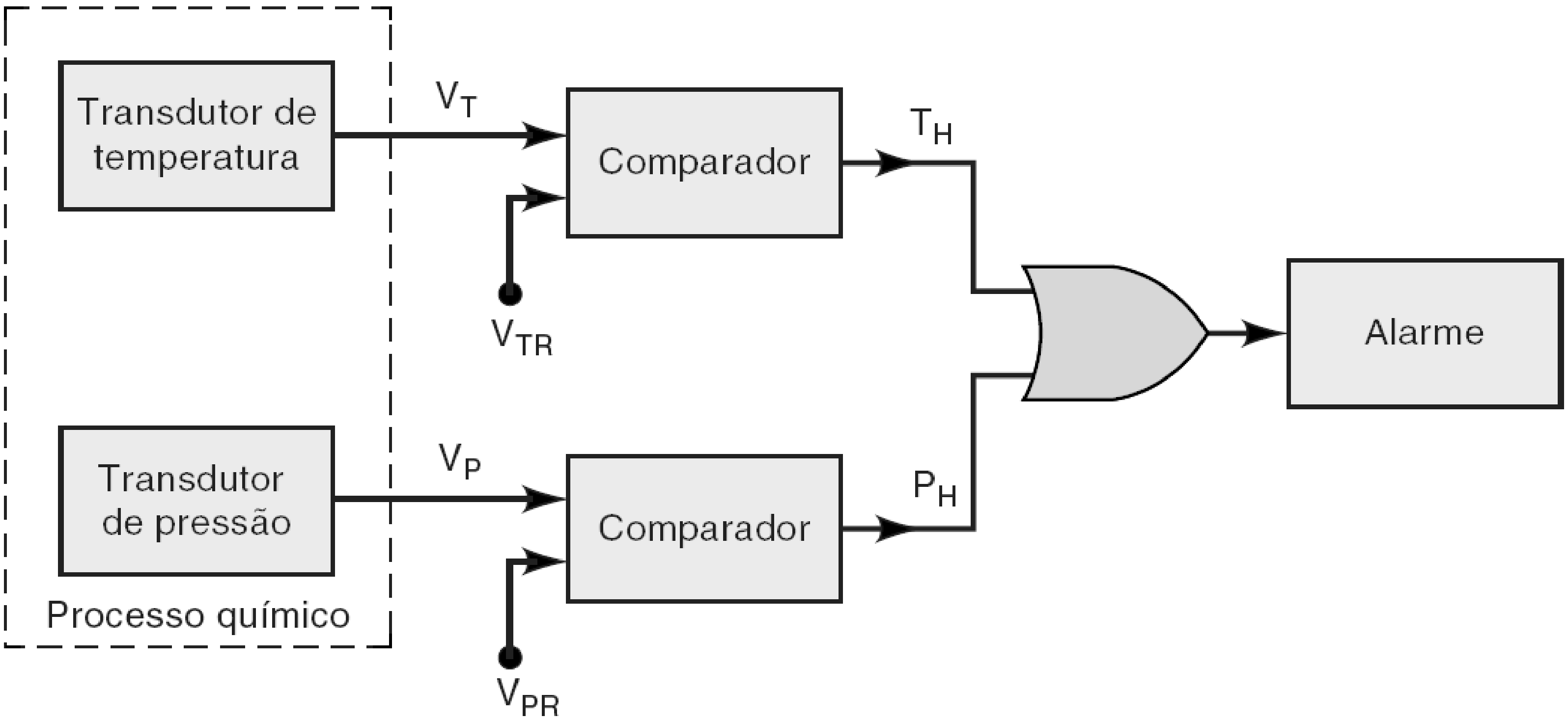
Tabela-verdade símbolo de circuito para três entradas da porta OR.



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

OPERAÇÃO OR COM PORTA OR

Exemplo do uso de uma porta OR em um sistema de alarme.



OPERAÇÃO AND (“E”) COM PORTAS AND

- A operação AND é similar a multiplicação convencional.

$X = A \bullet B \bullet C$ — Leia“X é igual a A e B e C”.

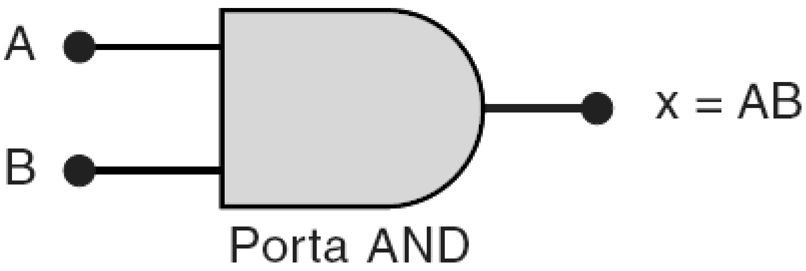
O sinal • + não se aplica para soma, mas sim para operações AND.

X é verdadeiro (1) quando *A* e *B* e *C são verdadeiros* (1).

AND

A	B	$x = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

Tabela-Verdade

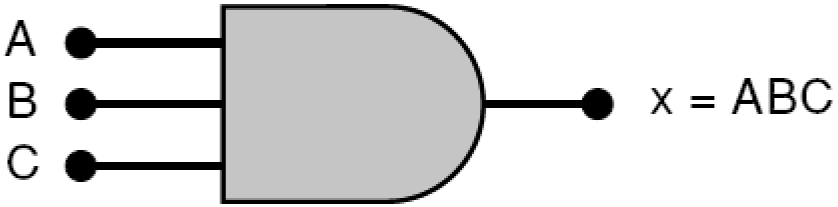


Simbolo da Porta

OPERAÇÃO AND COM PORTA AND

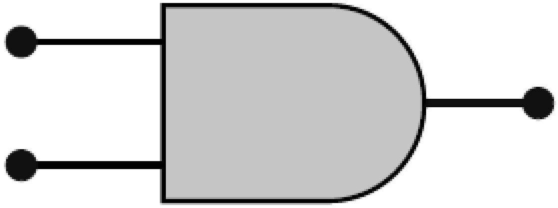
Tabela-verdade símbolo de circuito para três entradas e porta AND.

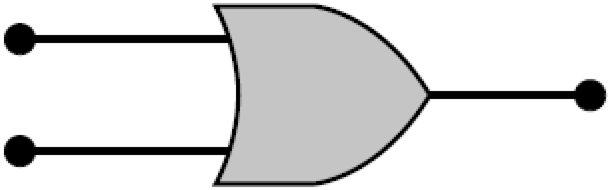
A	B	C	x = ABC
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



AND OR

O símbolo AND em um diagrama de circuito lógico diz que a saída será ALTO apenas quando todas as entradas forem altas.





O símbolo OR será alto quando alguma entrada for alta.

OPERAÇÃO NOT

- A expressão booleana para a operação NOT:

$$X = \overline{A}$$

— Leia:

A barra superior representa a operação NOT.

"X equivale a **NOT** A".

"X equivale ao inverso de A".

"X equivale ao complemento de A".

$$A' = \overline{A}$$

Outro indicador de inversão é o símbolo principal (').

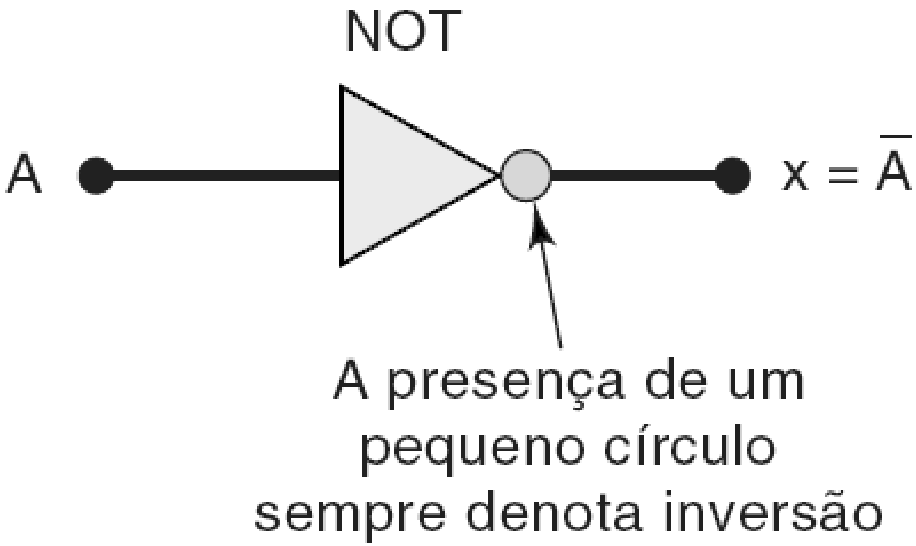
NOT

A	$x = \overline{A}$
0	1
1	0

Tabela-verdade NOT

OPERAÇÃO NOT

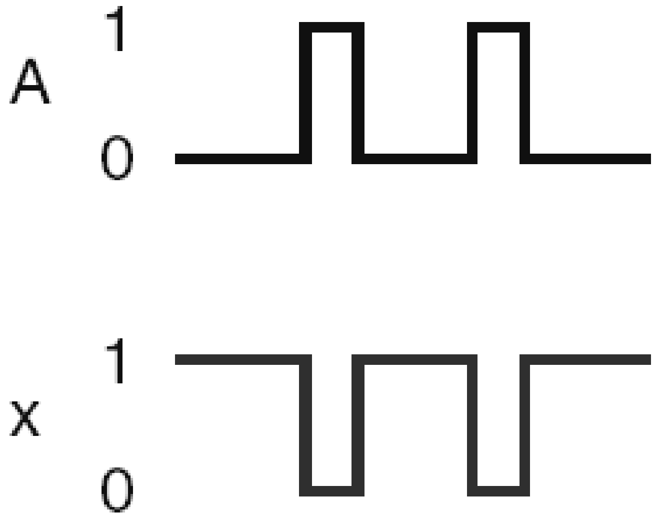
Um circuito NOT é comumente chamado de inversor.



Esses circuitos sempre têm uma única entrada, e a lógica da saída é sempre oposta ao nível da lógica da entrada.

OPERAÇÃO NOT

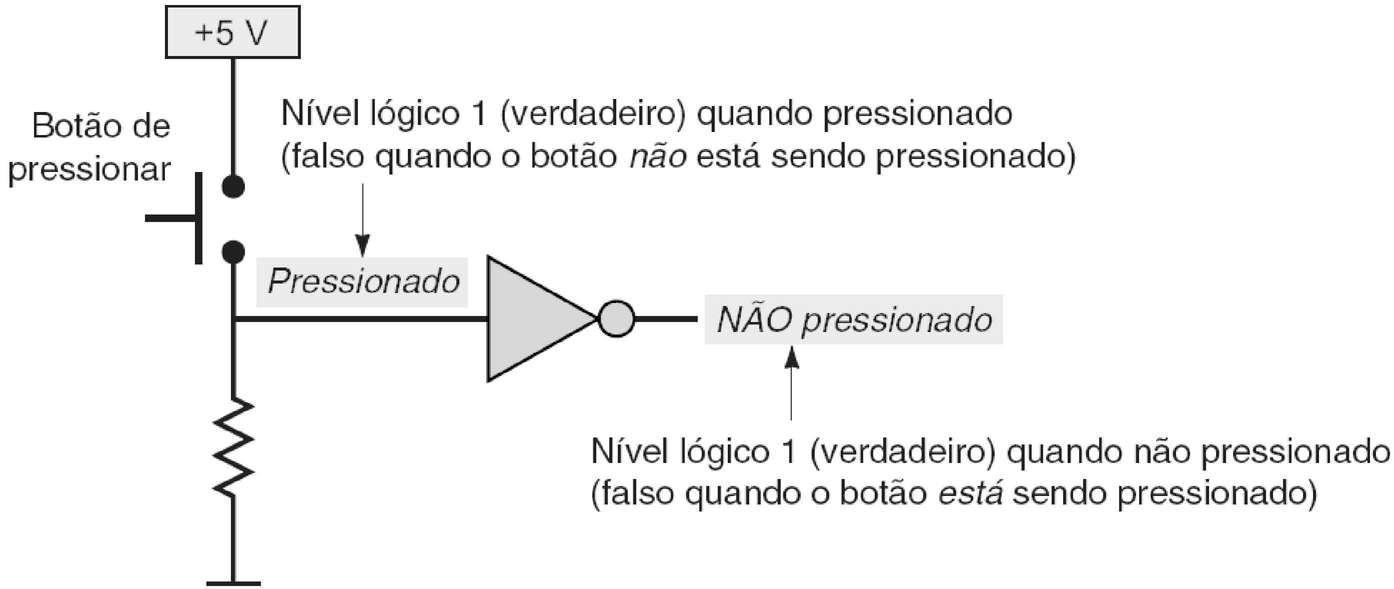
O INVERSOR inverte (complementa) o sinal da entrada, em todos os pontos, na forma de onda.



Sempre que a entrada = 0 a saída = 1 e vice-versa.

OPERAÇÃO NOT

Aplicação típica da porta **NOT**



O circuito fornece uma expressão que é verdadeira quando o botão não está pressionado.

OPERAÇÕES BOOLEANAS

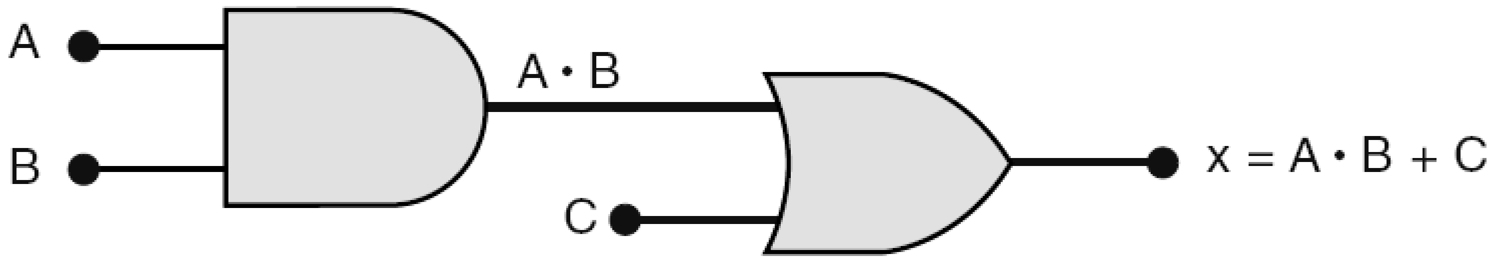
Regras resumidas para **OR**, **AND** e **NOT**

OR	***AND***	***NOT***
$0 + 0 = 0$	$0 \cdot 0 = 0$	$\overline{0} = 1$
$0 + 1 = 1$	$0 \cdot 1 = 0$	$\overline{1} = 0$
$1 + 0 = 1$	$1 \cdot 0 = 0$	
$1 + 1 = 1$	$1 \cdot 1 = 1$	

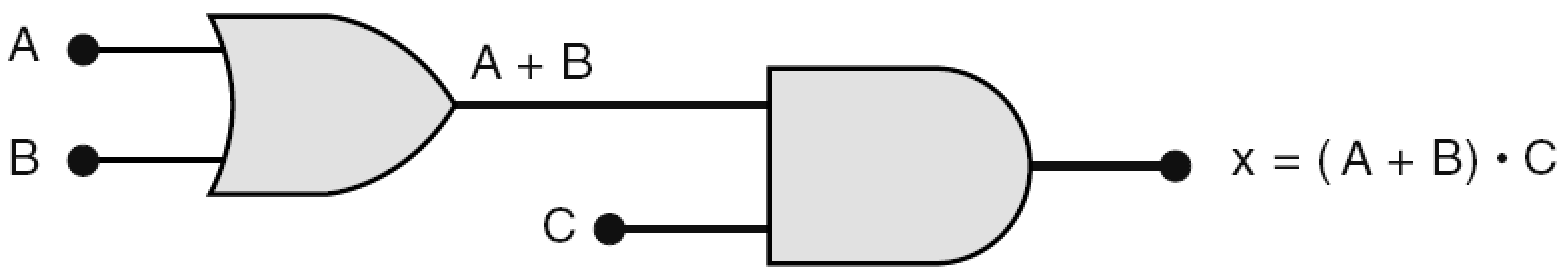
Essas três operações booleanas básicas podem descrever qualquer circuito lógico.

DESCREVENDO CIRCUITOS LÓGICOS ALGEBRICAMENTE

- Se uma expressão contém ambas as portas – **AND** e **OR** – a operação **AND** irá acontecer anteriormente.



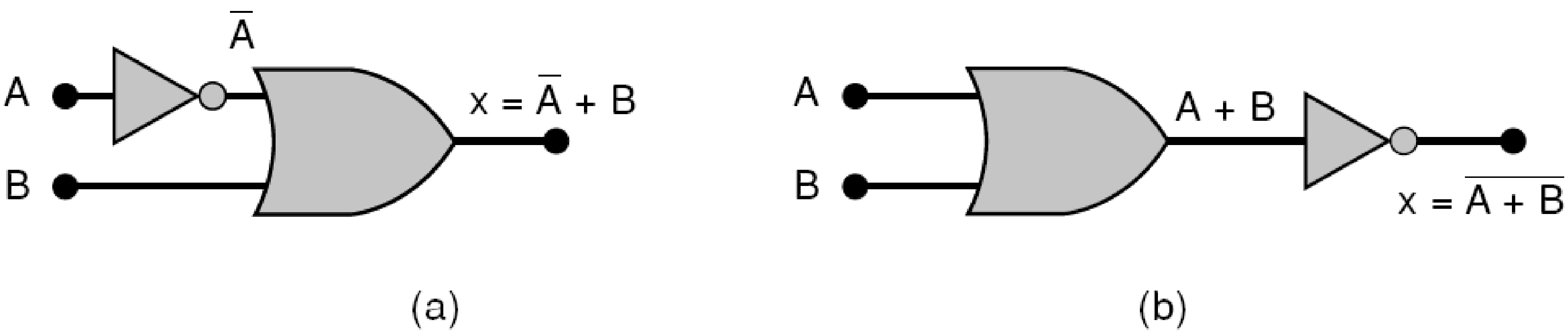
- A menos que existam parêntesis na expressão.



DESCREVENDO CIRCUITOS LÓGICOS ALGEBRICAMENTE

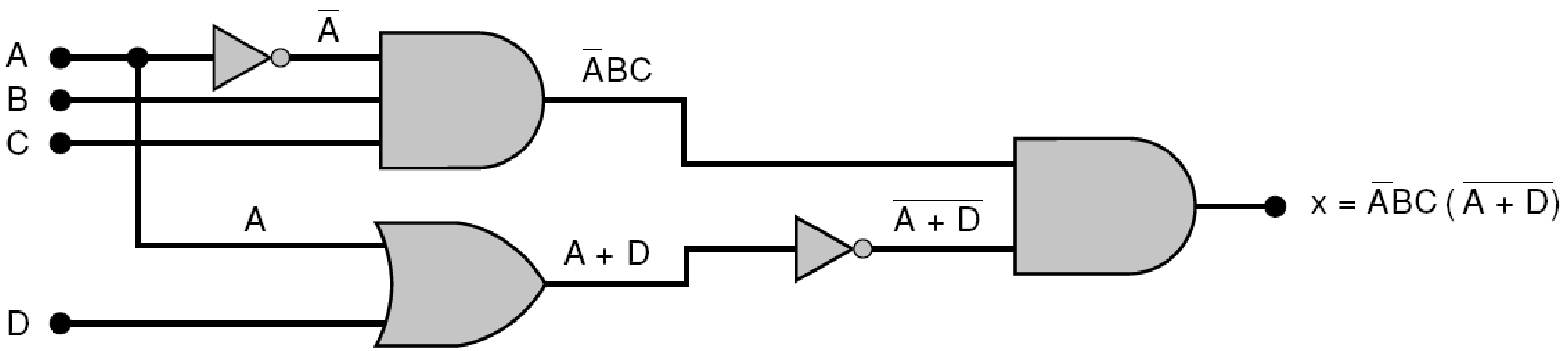
- Sempre que um INVERSOR estiver presente, a saída é equivalente a entrada, com uma barra sobre ele.

  Entrada *A* através de um inversor é igual a *A*.



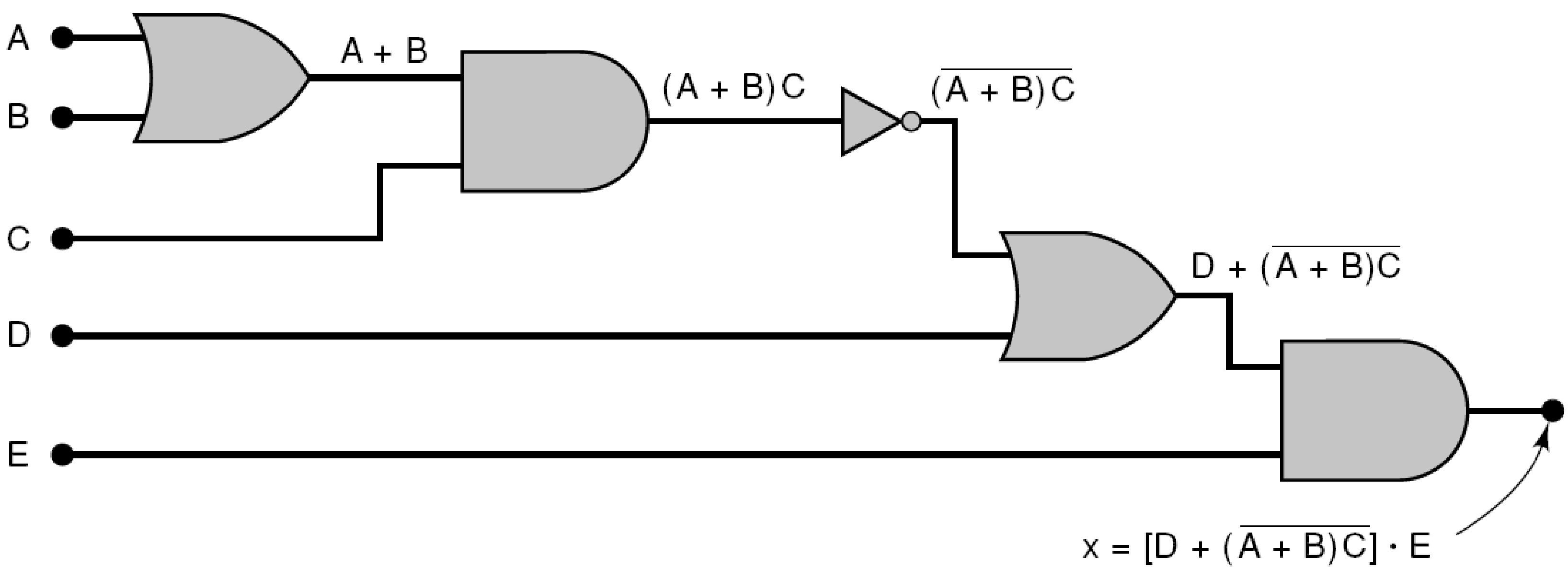
DESCREVENDO CIRCUITOS LÓGICOS ALGEBRICAMENTE

- Outros exemplos…



DESCREVENDO CIRCUITOS LÓGICOS ALGEBRICAMENTE

- Outros exemplos…



AVALIANDO AS SAÍDAS DOS CIRCUITOS LÓGICOS

Regras para avaliação de uma expressão booleana:

- Executar todas as inversões de termos individuais.
- Realizar todas as operações dentro de parêntesis.
- Realizar a operação **AND** antes de uma operação **OR**, a menos que os parêntesis indiquem o contrário.
- Sempre que uma expressão tiver uma barra sobre ela, realizar as operações no interior da expressão e depois inverter o resultado.

AVALIANDO AS SAÍDAS DOS CIRCUITOS LÓGICOS

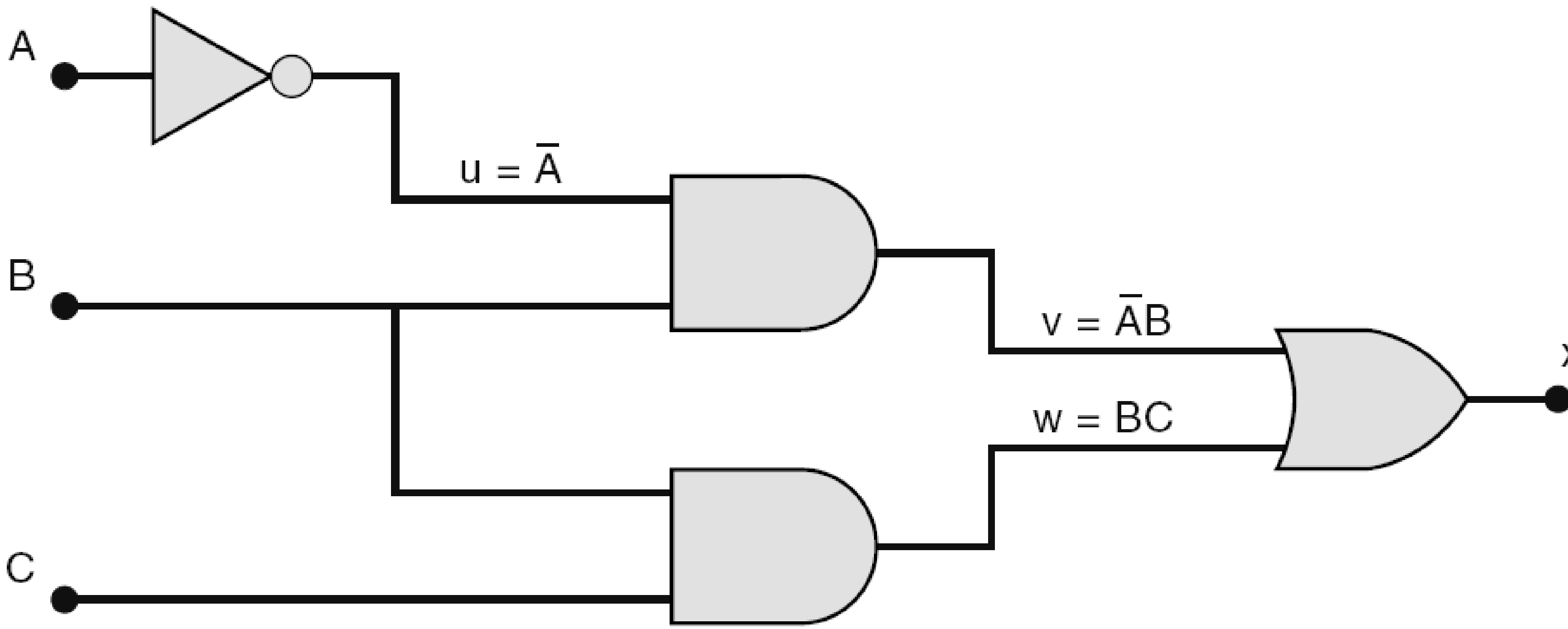
- A melhor maneira de analisar um circuito composto por várias portas lógicas é usar uma tabela-verdade.

Ela permite analisar uma porta ou uma combinação lógica de uma só vez.

Ela também permite verificar novamente seu trabalho.

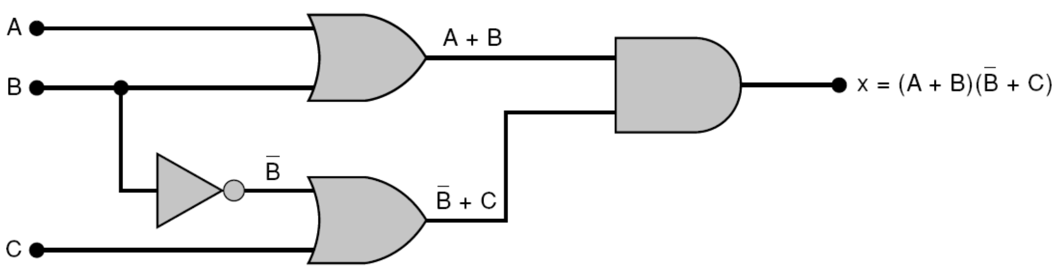
Ao terminar, você tem um quadro de enorme benefício para solucionar o circuito lógico.

Avaliando as Saídas dos Circuitos Lógicos



AVALIANDO AS SAÍDAS DOS CIRCUITOS LÓGICOS

O primeiro passo, após listar todas as combinações de entradas, é criar uma coluna na tabela-verdade para cada sinal intermediário (nó).

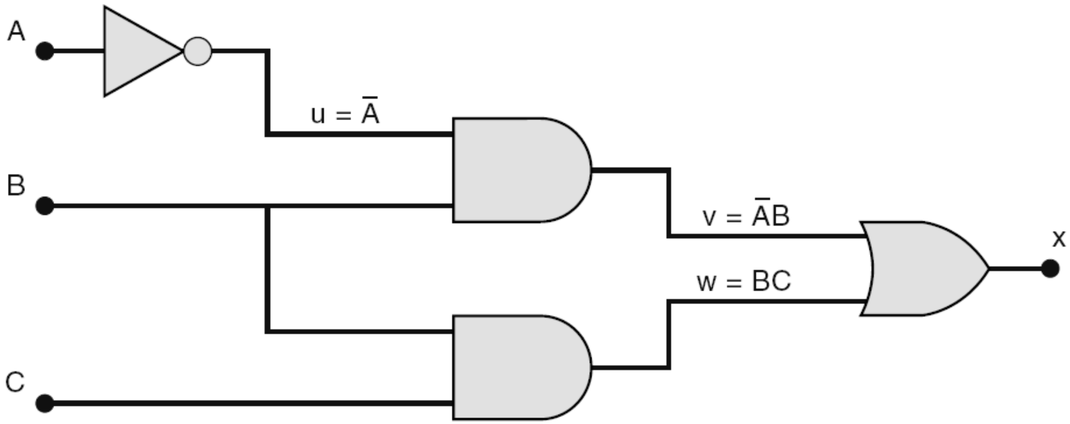


A	B	C	u= $\overline{A}$	v= $\overline{A}B$	w= BC	x= v+w
0	0	0	1			
0	0	1	1			
0	1	0	1			
0	1	1	1			
1	0	0	0			
1	0	1	0			
1	1	0	0			
1	1	1	0			

O nó *U* foi preenchido como complemento de *A*.

AVALIANDO AS SAÍDAS DOS CIRCUITOS LÓGICOS

- O próximo passo é preencher os valores para a coluna v.

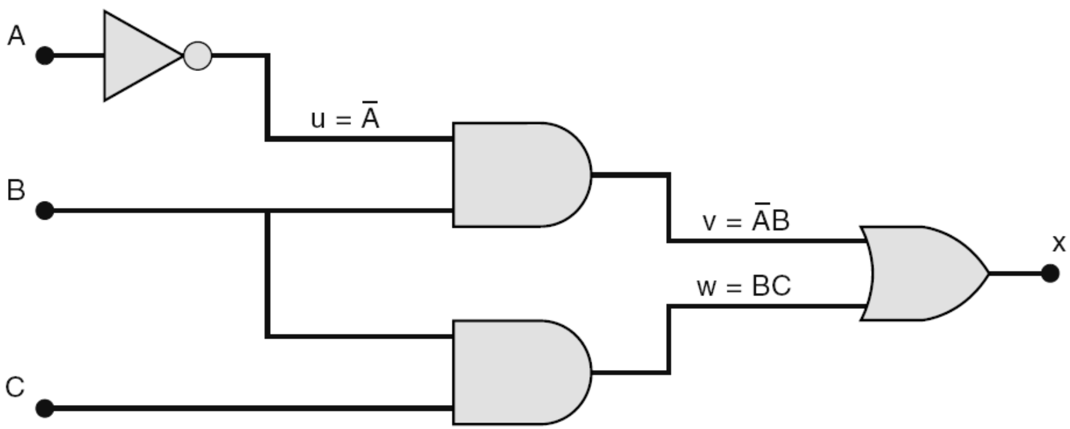


A	B	C	u= $\overline{A}$	v= $\overline{A}B$	w= BC	x= v+w
0	0	0	1	0		
0	0	1	1	0		
0	1	0	1	1		
0	1	1	1	1		
1	0	0	0	0		
1	0	1	0	0		
1	1	0	0	0		
1	1	1	0	0		

v =*AB* — O nó *v* deve ser ALTO
quando *A* (nó *u*) é ALTO e *B* é ALTO.

AVALIANDO AS SAÍDAS DOS CIRCUITOS LÓGICOS

- O terceiro passo é estimar os valores do nó *w*, o produto lógico de BC.

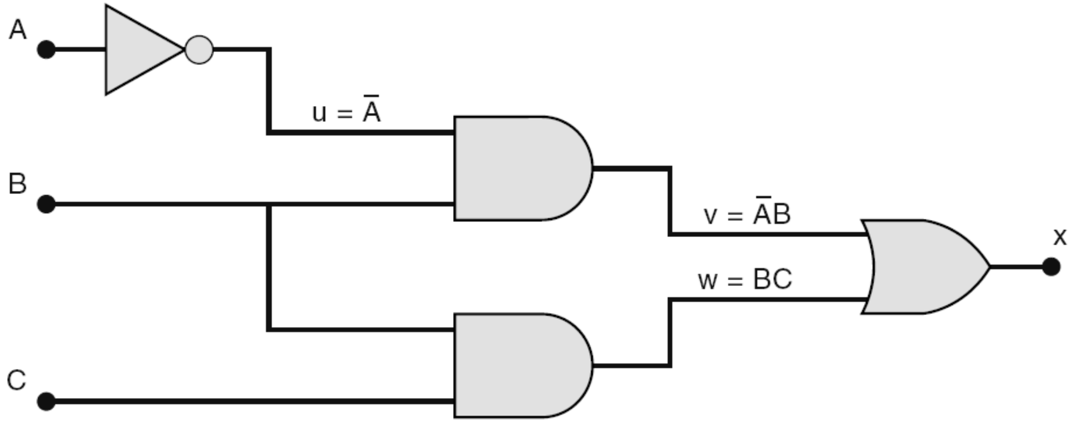


A	B	C	$u=\overline{A}$	$v=\overline{A}B$	$w=BC$	$x=v+w$
0	0	0	1	0	0	
0	0	1	1	0	0	
0	1	0	1	1	0	
0	1	1	1	1	1	
1	0	0	0	0	0	
1	0	1	0	0	0	
1	1	0	0	0	0	
1	1	1	0	0	1	

A coluna é ALTO sempre que *B* é ALTO e *C* é ALTO.

AVALIANDO AS SAÍDAS DOS CIRCUITOS LÓGICOS

- Logicamente, a etapa final é a combinação das colunas V e W para prever a *saída* x.



A	B	C	u= $\bar{A}$	v= $\bar{A}$B	w= BC	x= v+w
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	1	0	1	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	1	1

Desde que $x = v + w$, a saída *x* será ALTO quando *v* OU *w* for ALTO.

AVALIANDO AS SAÍDAS DOS CIRCUITOS LÓGICOS

- Níveis lógicos de saída podem ser determinados diretamente a partir de um diagrama de circuito.
  As saídas de cada porta são percebidas até que a saída final seja encontrada.

- Os técnicos usam esse método com frequência.

AVALIANDO AS SAÍDAS DE CIRCUITOS LÓGICOS

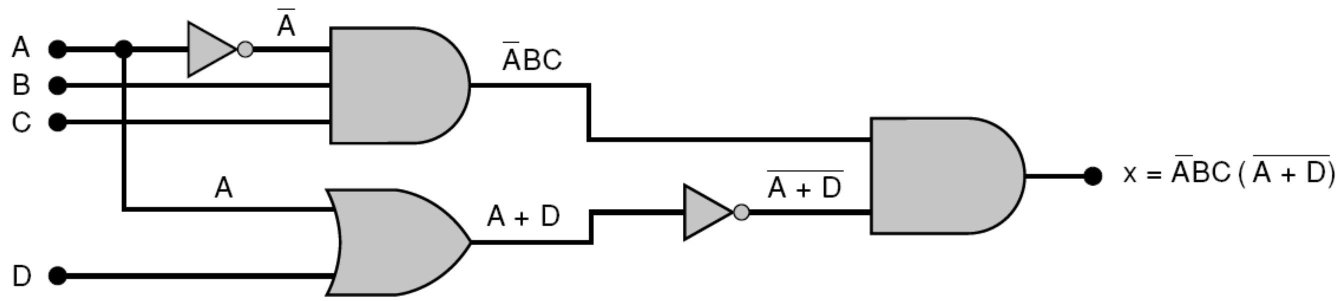


Tabela de estado lógico em cada nó do circuito mostrado

A	B	C	D	$t = \overline{A}BC$	$u = A + D$	$v = \overline{A + D}$	$x = tv$
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	1	0	0	0	1	0
0	0	1	1	0	1	0	0
0	1	0	0	0	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	1
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	0
1	0	0	1	0	1	0	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	0	0
1	1	0	0	0	1	0	0
1	1	0	1	0	1	0	0
1	1	1	0	0	1	0	0
1	1	1	1	0	1	0	0

IMPLEMENTANDO CIRCUITOS A PARTIR DE EXPRESSÕES BOOLEANAS

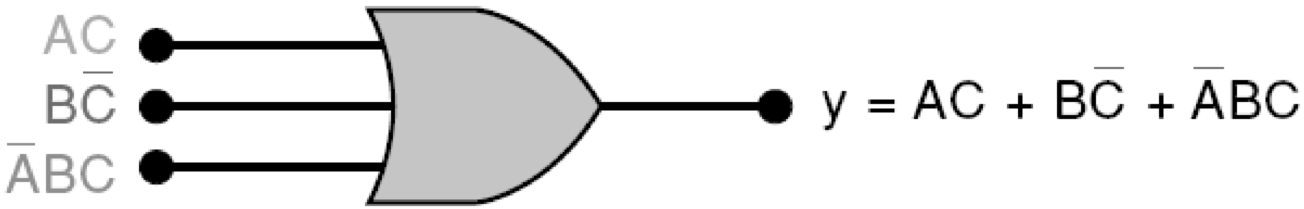
- É importante saber desenhar um circuito lógico de uma expressão booleana.

A expressão X = A . B . C poderia ser desenhada como três entradas de uma porta AND.

Um circuito definido por X = A + B usaria duas entradas de uma porta OR com um INVERSOR em uma das entradas.

IMPLEMENTANDO CIRCUITOS A PARTIR DE EXPRESSÕES BOOLEANAS

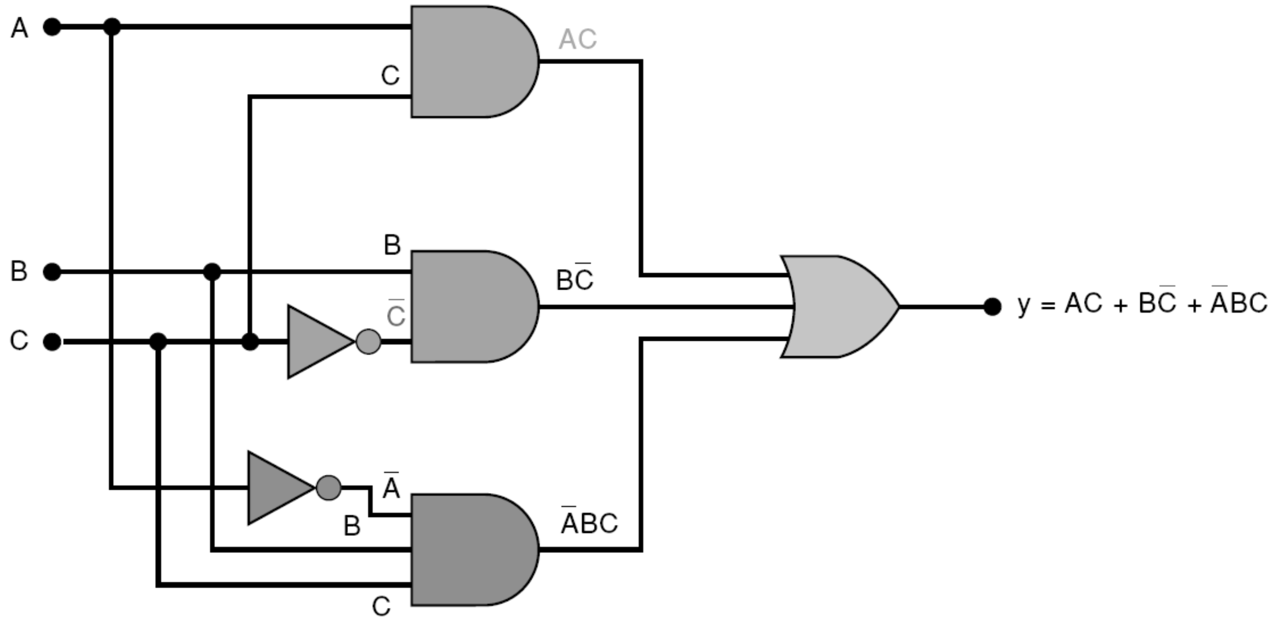
Um circuito com saída $y = AC + B\overline{C} + \overline{A}BC$ contém três termos sobre os quais é aplicada a operação OR…



…e requer uma porta OR de três entradas.

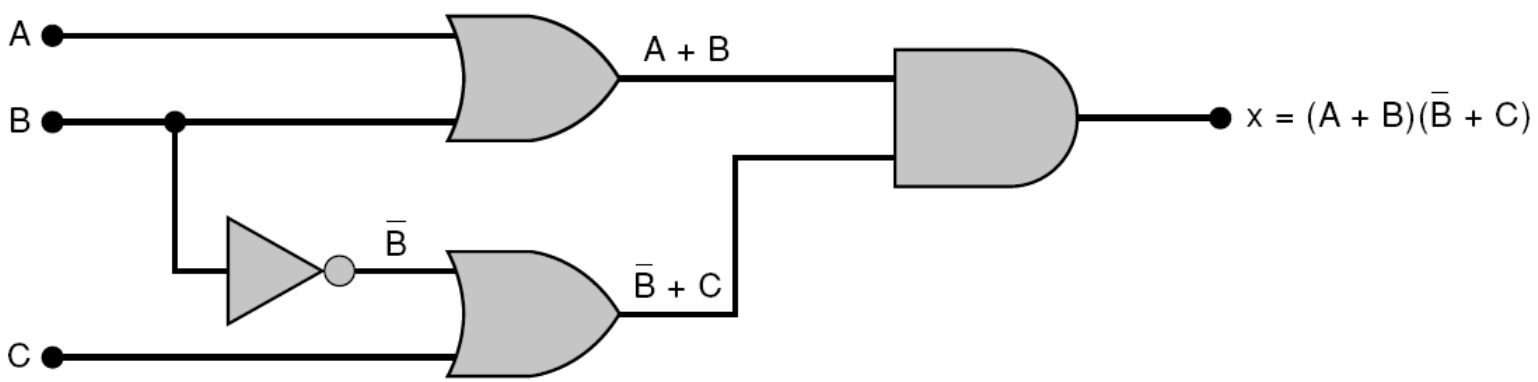
IMPLEMENTANDO CIRCUITOS A PARTIR DE EXPRESSÕES BOOLEANAS

- Cada entrada da porta OR é um termo do produto AND.
- Uma porta AND com entradas adequadas pode ser usada para gerar cada um desses termos.



IMPLEMENTANDO CIRCUITOS A PARTIR DE EXPRESSÕES BOOLEANAS

Diagrama de circuito para implementar $x = (A + B)\,(\overline{B} + C)$.



PORTAS NOR ("NÃO-OU") E PORTAS NAND

- Combine operações básicas **AND**, **OR e NOT** simplificando a escrita de expressões booleanas.

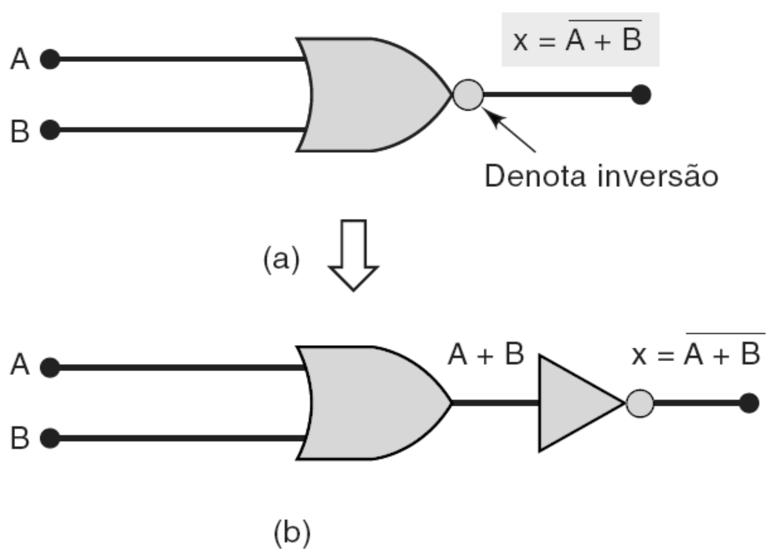
- As saídas das portas **NAND** e **NOR** podem ser encontradas ao determinar a saída de uma porta **AND** ou **OR** e invertê-la**.**

  As tabelas-verdade para portas **NOR** e **NAND** mostram o complemento das tabelas-verdade para portas **OR** e **AND**.

PORTAS NOR E PORTAS NAND

- A porta **NOR** é uma porta **OR** invertida.

Um “ bubble ” de inversão é colocado na saída da porta **OR**, tornando a saída da expressão booleana $\boldsymbol{x} = \overline{\boldsymbol{A} + \boldsymbol{B}}$

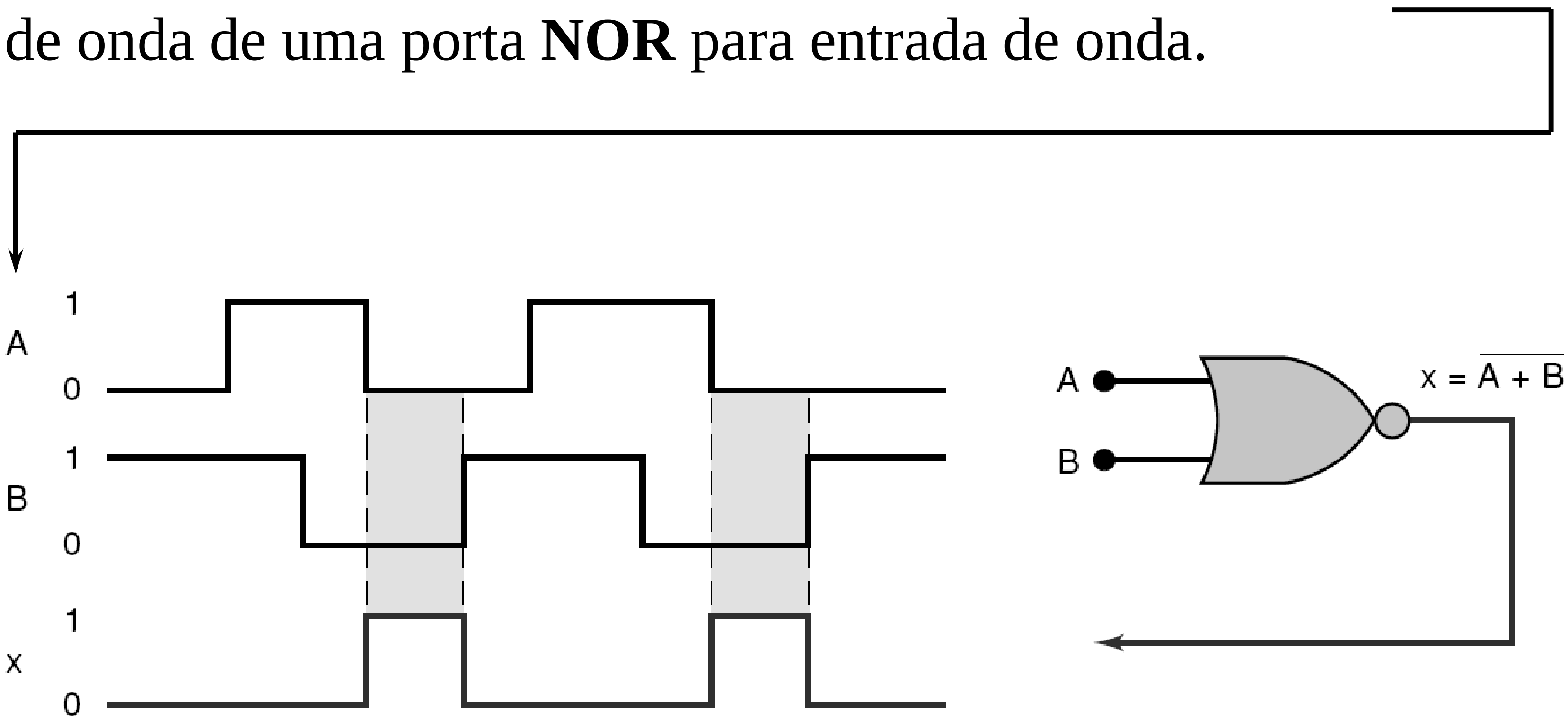


		OR	NOR
A	B	$A + B$	$\overline{A + B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

(c)

PORTAS NOR E PORTAS NAND

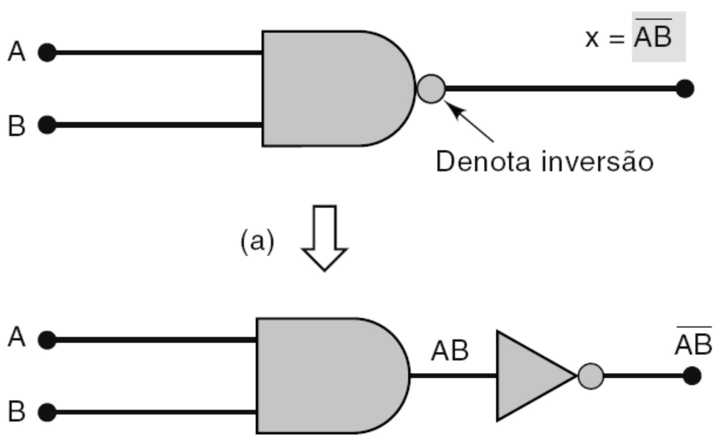
Saída de onda de uma porta **NOR** para entrada de onda.



Portas NOR e Portas NAND

- A porta **NAND** é uma porta **AND** invertida.

  Um "bubble" de inversão é colocado no output de porta **AND**, tornando o output da expressão booleana $\boldsymbol{x = \overline{AB}}$

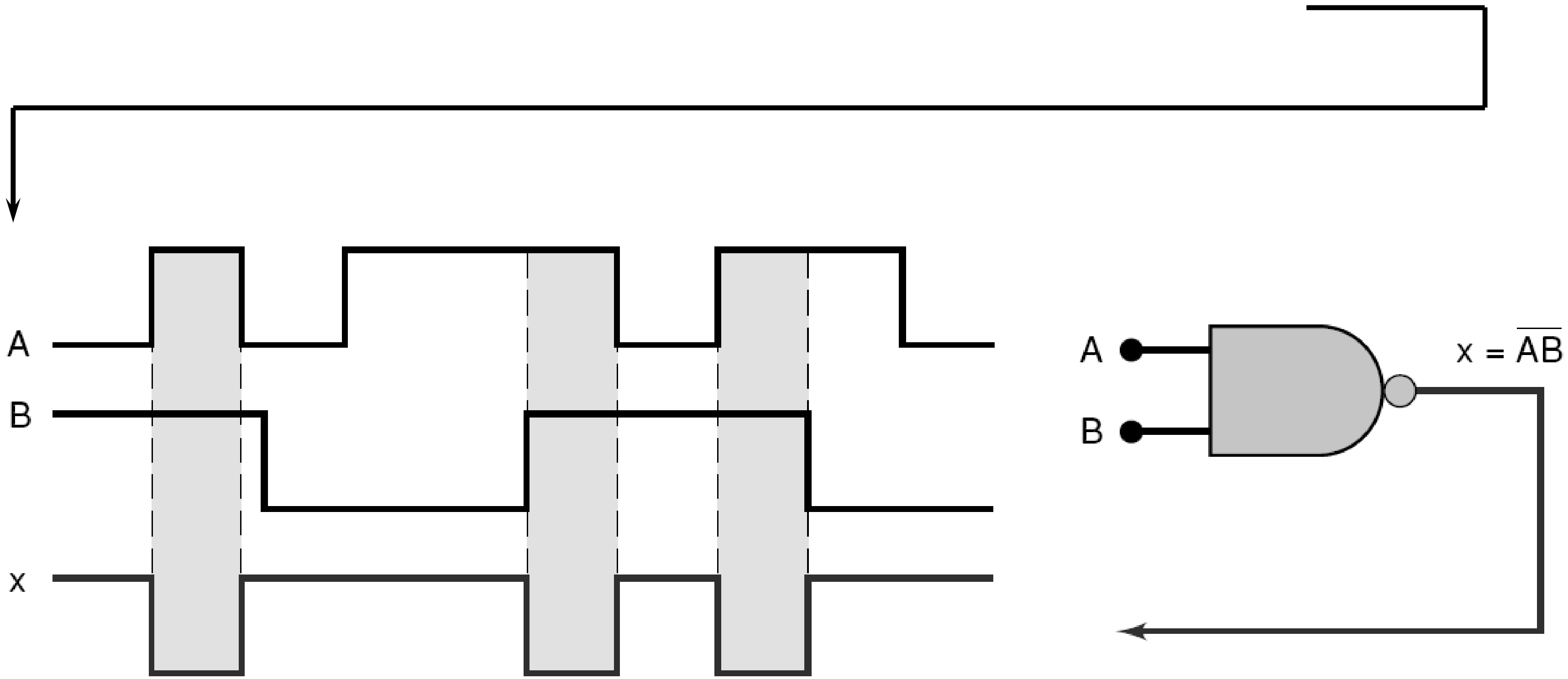


		AND	NAND
A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

(c)

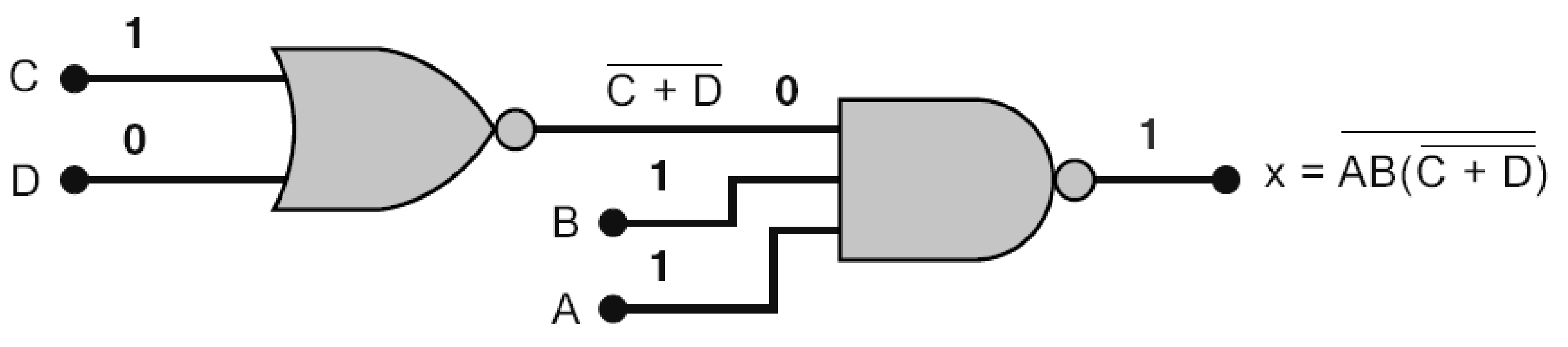
PORTAS NOR E PORTAS NAND

Saída de onda de uma porta **NAND** para entrada de onda.



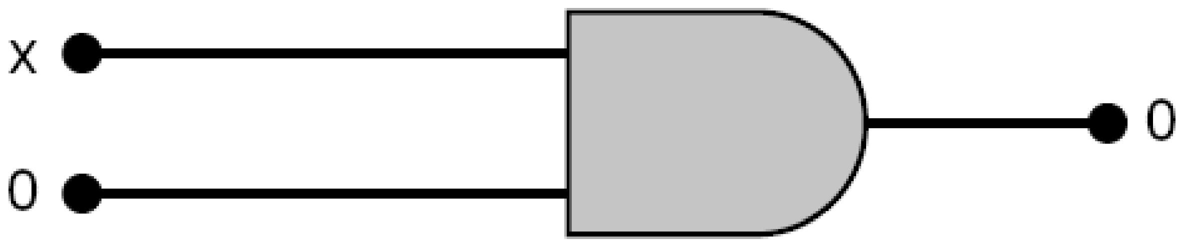
Portas NOR e Portas NAND

Circuito lógico com a expressão $\boldsymbol{x = \overline{AB \cdot (\overline{C + D})}}$ usando apenas **NOR** e **NAND**.



TEOREMAS BOOLEANOS

Os seguintes teoremas e leis podem representar uma expressão que contém mais de uma variável.

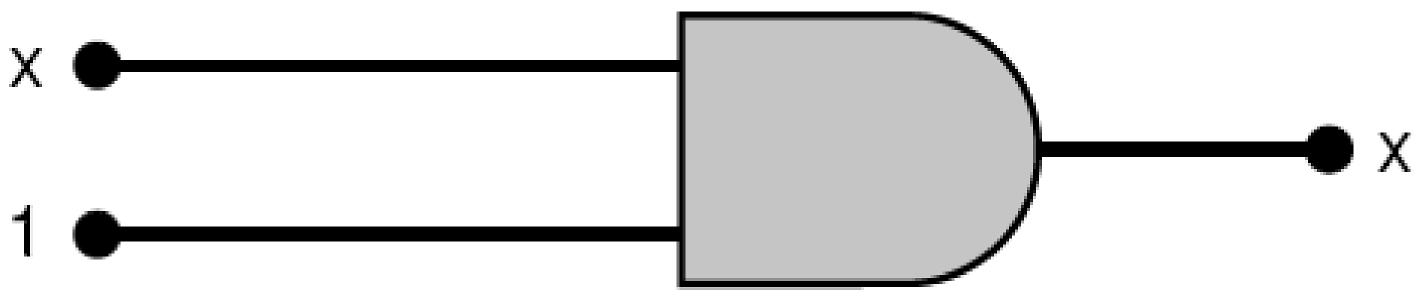


(1) $x \cdot 0 = 0$

O teorema (1) afirma que, se qualquer variável é combinada com 0 usando a operação AND, o resultado deve ser 0.

TEOREMAS BOOLEANOS

O teorema (2) também fica evidente quando comparado com a multiplicação ordinária.

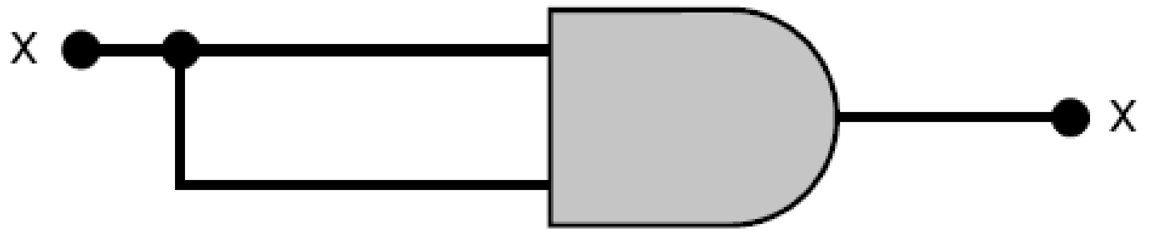


(2) $x \cdot 1 = x$

TEOREMAS BOOLEANOS

Comprove o teorema (3) tentando caso a caso

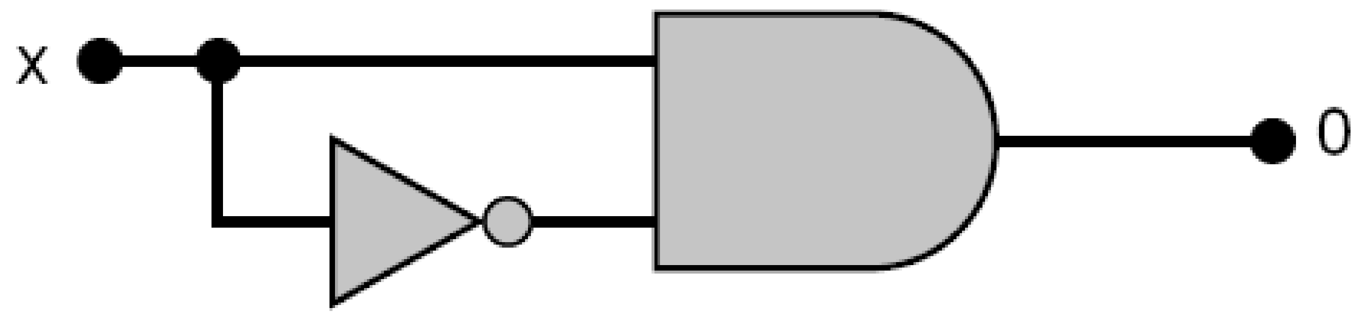
SE x = 0, então 0 • 0 = 0.
Se x = 1, então 1 • 1 = 1.
Logo, x • x = x.



(3) $x \cdot x = x$

TEOREMAS BOOLEANOS

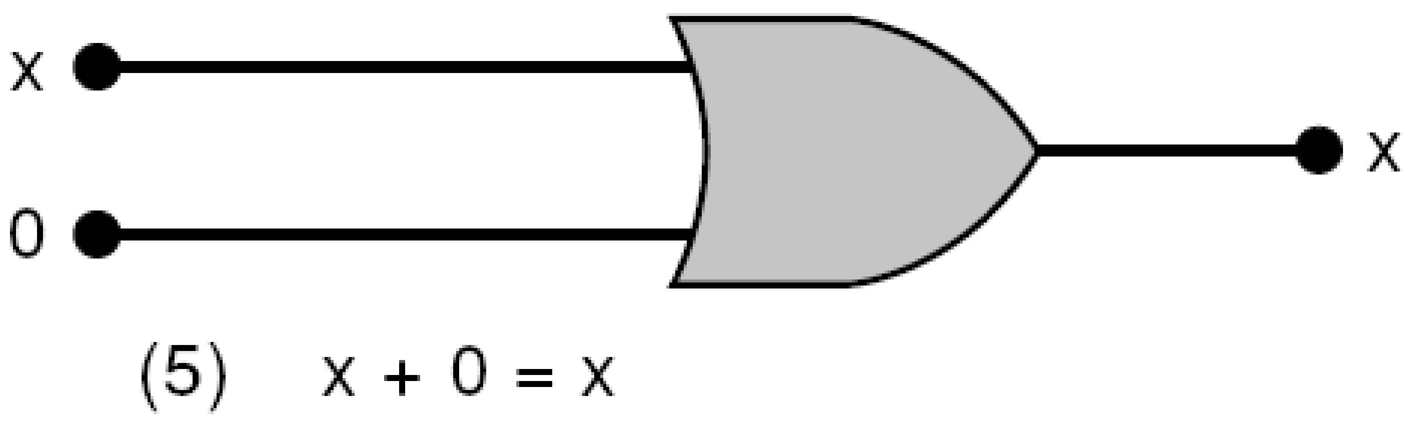
O teorema (4) pode ser comprovado da mesma maneira.



(4) $x \cdot \overline{x} = 0$

TEOREMAS BOOLEANOS

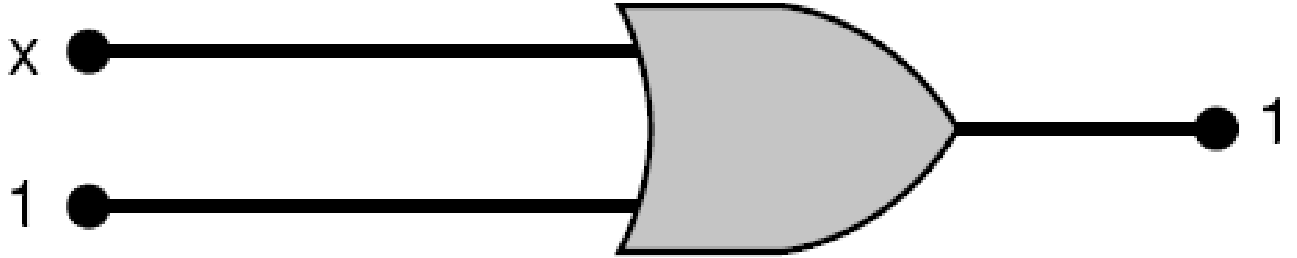
O teorema (5) é simples, pois 0 acrescentado a *alguma coisa* não afeta seu valor, tanto na adição regular quanto na operação OR.



TEOREMAS BOOLEANOS

O teorema (6) afirma que, se uma variável é combinada com 1 usando-se a operação OR, o resultado é sempre 1.

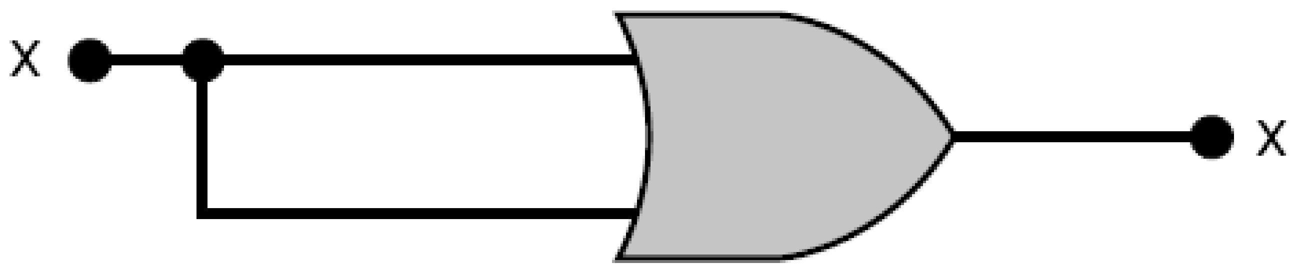
Valores de conferência: 0 + 1 = 1 e 1 + 1 = 1.



(6) $x + 1 = 1$

TEOREMAS BOOLEANOS

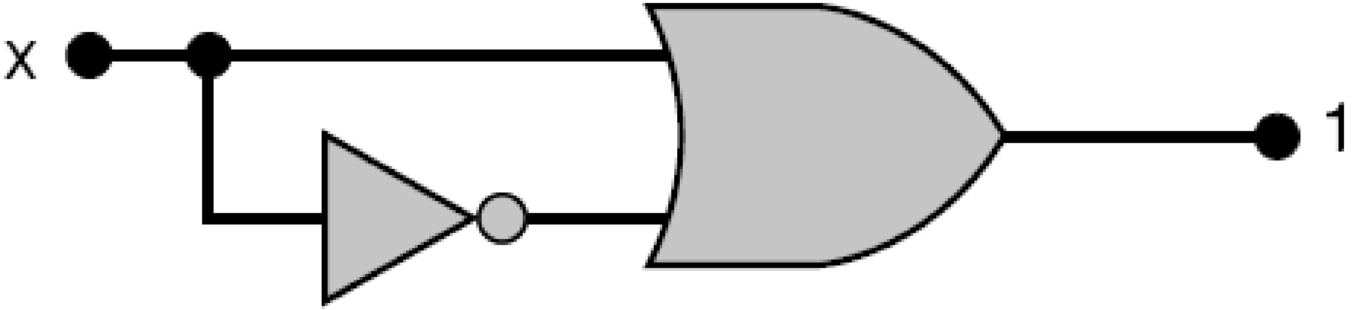
O teorema (7) pode ser comprovado através da verificação para ambos os valores de x: 0 + 0 = 0 e 1 + 1 = 1.



(7) $x + x = x$

TEOREMAS BOOLEANOS

O teorema 8 pode ser provado similarmente.



(8) $x + \overline{x} = 1$

TEOREMAS BOOLEANOS

Teoremas multivariáveis

Leis comutativas

(9) $x + y = y + x$

(10) $x \cdot y = y \cdot x$

Leis associativas

(11) $x + (y + z) = (x + y) + z = x + y + z$

(12) $x(yz) = (xy)z = xyz$

Leis distributivas

(13*a*) $x(y + z) = xy + xz$

(13*b*) $(w + x)(y + z) = wy + xy + wz + xz$

TEOREMAS BOOLEANOS

Teoremas multivariáveis

Os teoremas (14) e (15) não possuem equivalentes na álgebra comum. Cada um deles pode ser provado ao tentar todos os casos possíveis para *x* e *y*.

$$(14)\quad x + xy = x$$
$$(15a)\quad x + \bar{x}y = x + y$$
$$(15b)\quad \bar{x} + xy = \bar{x} + y$$

Tabela de análise e fatoração para teorema (14)

$$\begin{aligned} x + xy &= x(1+y) \\ &= x \cdot 1 && [\textit{usando o teorema } (6)] \\ &= x && [\textit{usando o teorema } (2)] \end{aligned}$$

x	y	xy	x + xy
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

TEOREMAS DE DEMORGAN

- **Teoremas de DeMorgan** são extremamente úteis na simplificação de expressões em que um produto ou a soma das variáveis é invertida.

$$(16)\quad (\overline{x + y}) = \overline{x} \cdot \overline{y}$$

O teorema (16) diz que INVERSOR a soma OR de duas variáveis é o mesmo que INVERSOR cada variável individualmente. Com isso, operar com AND as variáveis invertidas.

$$(17)\quad (\overline{x \cdot y}) = \overline{x} + \overline{y}$$

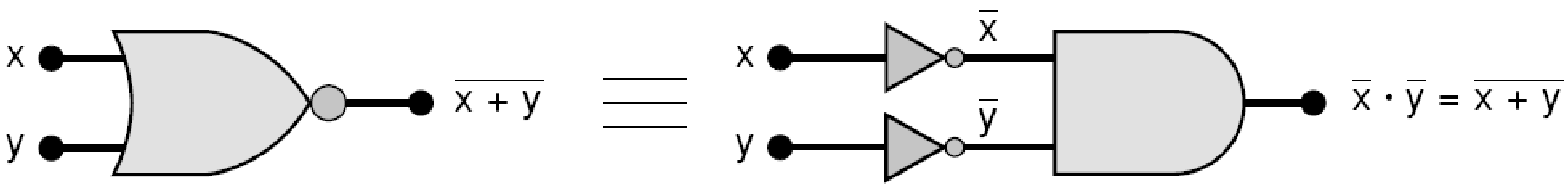
O teorema (17) diz que INVERSOR o produto E de duas variáveis é o mesmo que INVERSOR cada variável individualmente e, em seguida, operar com OR.

Cada um dos teoremas de DeMorgan pode ser facilmente comprovado por meio da verificação de todas as combinações possíveis de *x* e *y*.

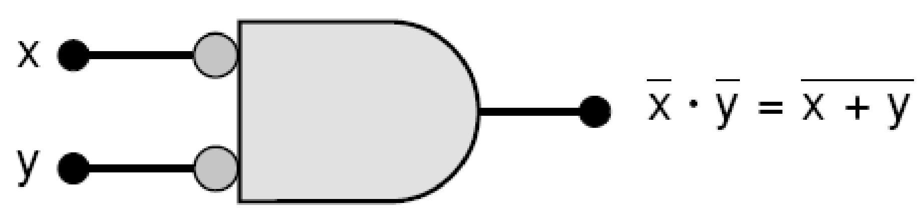
TEOREMAS DE DEMORGAN

Circuitos equivalentes decorrentes do teorema (16)

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



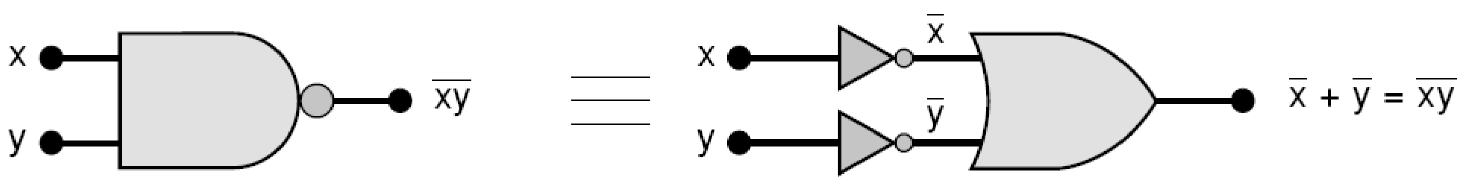
Símbolo alternativo para a função NOR.



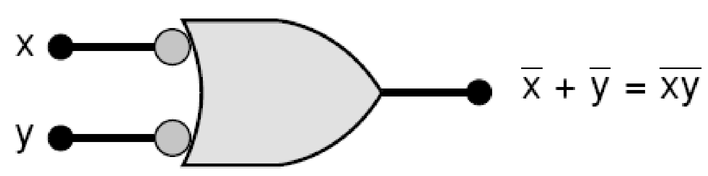
TEOREMAS DE DEMORGAN

Circuitos equivalentes decorrentes do teorema (17)

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



Símbolo alternativo para a função NAND.



UNIVERSALIDADE DAS PORTAS NAND E NOR

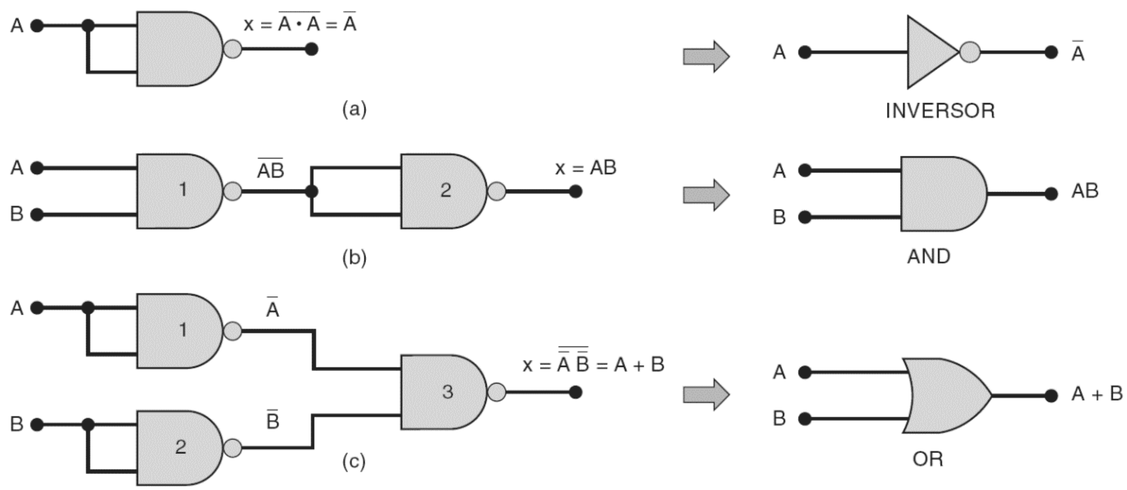
- Portas NAND ou NOR podem ser usadas para criar as três expressões lógicas básicas:

  OR, AND e NOT.

- Proporciona flexibilidade e é muito útil no projeto de circuito lógico.

UNIVERSALIDADE DAS PORTAS NAND E NOR

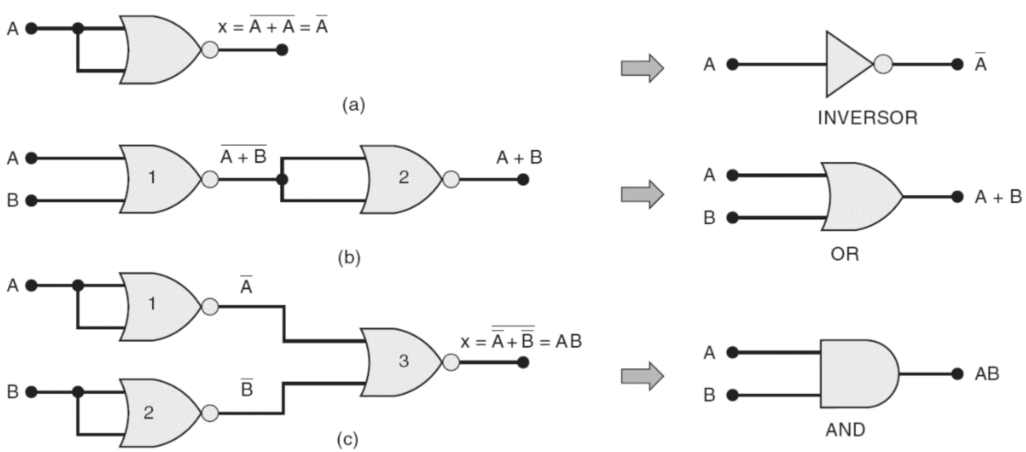
Como as combinações de NANDs ou NORs são usadas para criar as três funções lógicas.



No entanto, é possível implementar qualquer expressão lógica usando apenas portas NAND e nenhum outro tipo de porta, como mostrado.

UNIVERSALIDADE DAS PORTAS NAND e NOR

Como combinações de NANDs ou NORs são usadas para criar as três funções lógicas.



Portas NOR podem ser organizadas para implementar cada uma das operações booleanas, como mostrado.

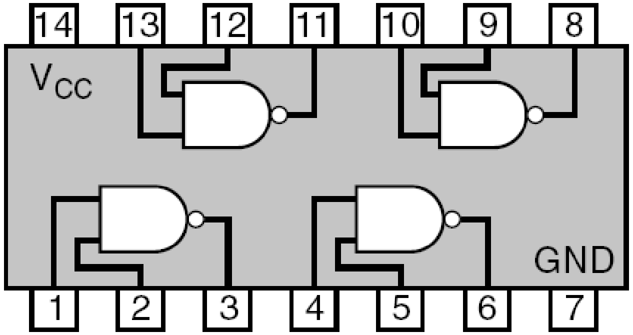
UNIVERSALIDADE DAS PORTAS NAND e NOR

Um circuito lógico gera um sinal *x*, que será ALTO sempre que as condições *A* e *B* existirem simultaneamente, ou sempre que as condições *C* e *D* existirem simultaneamente.

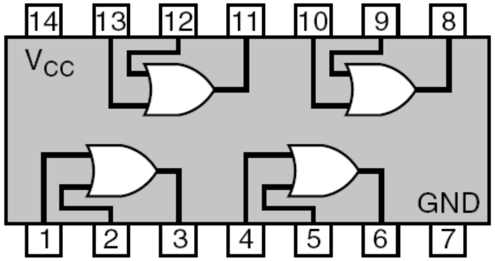
A expressão lógica será *x* = *AB* + *CD*.

Cada um dos ICs TTL mostrados aqui vai cumprir a função. Cada IC é um quad, com quatro portas idênticas em um único chip

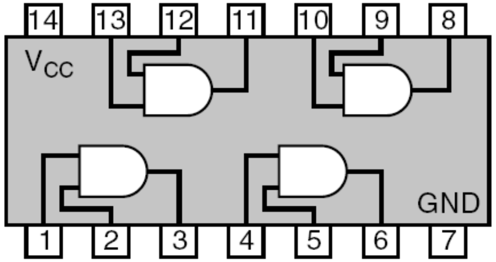
74LS00



74LS32

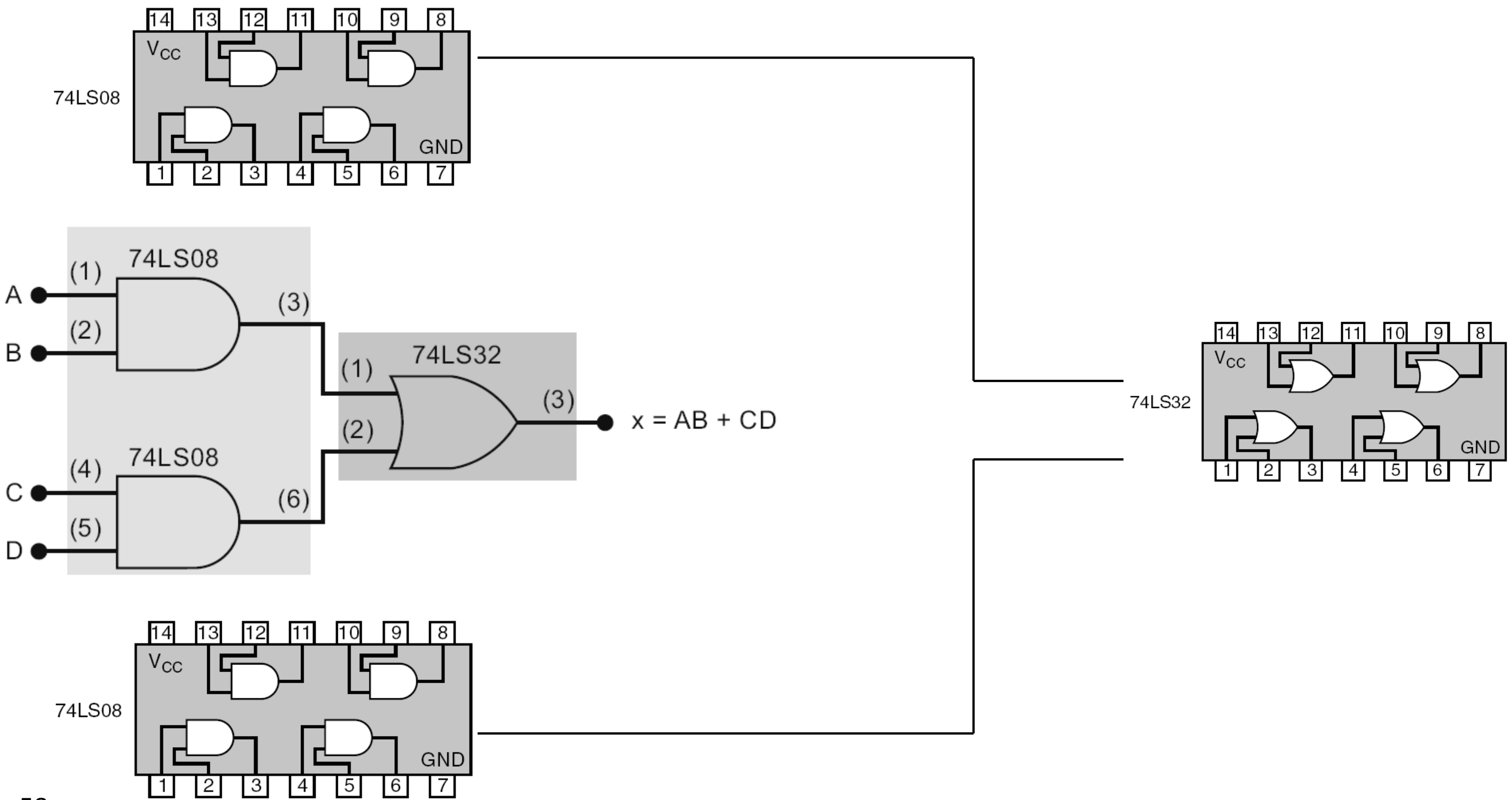


74LS08



UNIVERSALIDADE DAS PORTAS NAND e NOR

Possíveis implementações # 1:



UNIVERSALIDADE DAS PORTAS NAND e NOR

Possíveis implementações # 2: