

Sistemas Digitais

ET46B

Prof. Eduardo Vinicius Kuhn

kuhn@utfpr.edu.br

Curso de Engenharia Eletrônica

Universidade Tecnológica Federal do Paraná



Capítulo 3

Descrição de circuitos lógicos

Conteúdo

- 3.1 Constantes e variáveis booleanas
- 3.2 Tabelas-verdade
- 3.3 Operação e porta OR
- 3.4 Operação e porta AND
- 3.5 Operação e porta NOT
- 3.6 Descrevendo circuitos lógicos algebricamente
- 3.7 Avaliando as saídas dos circuitos lógicos
- 3.8 Implementando circuitos a partir de expressões booleanas
- 3.9 Portas NOR e NAND
- 3.10 Teoremas booleanos
- 3.11 Teoremas de DeMorgan
- 3.12 Universalidade das portas NOR e NAND
- 3.15 Atraso de propagação

Objetivos

- Introduzir as **três operações lógicas básicas** (OR, AND e NOT).
- Descrever a operação e **construir tabelas-verdade** para as portas OR, NOR, AND, NAND e NOT.
- **Obter expressões booleanas descrevendo circuitos lógicos.**
- **Implementar circuitos usando portas lógicas** a partir de expressões booleanas.
- Mostrar como **os teoremas da álgebra booleana** podem ser usados para simplificar expressões lógicas.
- **Demonstrar a universalidade das portas NOR e NAND.**
- **Discutir sobre o tempo de atraso de propagação.**

- **Álgebra convencional não é adequada** para lidar com circuitos digitais, uma vez que as variáveis podem assumir valores reais.
- **Na álgebra booleana**, as variáveis só podem assumir apenas dois “valores” distintos (i.e., 0 ou 1).
 - Proposta por George Boole (1854) e **levada para o mundo dos circuitos digitais por Claude Shannon (1930)**.
 - ...introduz **símbolos e operadores “próprios”**.
 - Torna possível descrever, manipular e simplificar expressões lógicas.
- Portanto, é uma ferramenta poderosa de análise, síntese e documentação para circuitos lógicos, tal como tabelas-verdade, símbolos, diagramas esquemáticos e diagramas de tempo.

Constantes e variáveis booleanas

- **Constantes e variáveis booleanas** podem assumir **apenas** dois “valores”, i.e., **0 ou 1**.
- As variáveis booleanas 0 e 1 **representam efetivamente um estado/nível lógico**.
- **Outros termos são, comumente, usados como sinônimos** desses níveis lógicos.

Lógico 0	Lógico 1
Falso	Verdadeiro
Baixo	Alto
Desligado	Ligado
Não	Sim
Aberto	Fechado

O valor booleano 0 pode representar qualquer tensão na faixa de 0–0,8 V, enquanto 1 pode representar qualquer tensão na faixa de 2–5 V.

Constantes e variáveis booleanas

- As **entradas** (e.g., A, B, C, \dots) e **saídas** (e.g., x, y, z, \dots) de um circuito lógico são consideradas variáveis lógicas.
- Os níveis lógicos da(s) entrada(s) determinam, a qualquer momento, os níveis lógicos da(s) saída(s).
- A álgebra booleana permite expressar relações entre entradas e saídas (variáveis e constantes) de um circuito.

$$x = (A + 0)\overline{CD} + (\overline{A} + C)(B + \overline{D}) + B(\overline{C} + 1)D$$

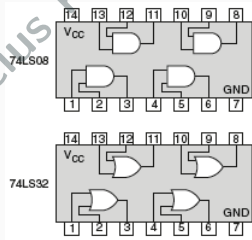
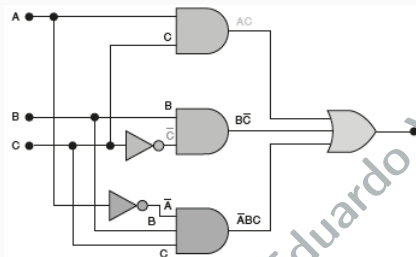
- Variáveis booleanas, em momentos diferentes, podem assumir níveis lógicos 0 ou 1 (se não for um nível lógico, será o outro).
- Constantes booleanas são pontos no circuito onde os níveis lógicos não se alteram ao longo do tempo (são sempre 0 ou 1).

A álgebra booleana tem, de fato, apenas três operações básicas (operações lógicas):

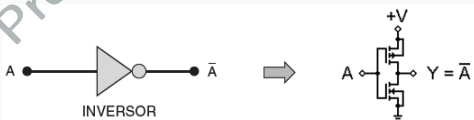
OR, AND e NOT

Constantes e variáveis booleanas

Em nosso mundo digital, essas operações são realizadas por portas lógicas construídas com transistores.

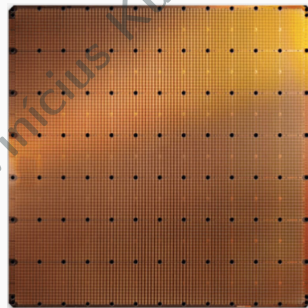


Exemplo:



Curiosidade

Porta/Função	Número de Transistores
<u>NOT</u>	2
<u>Buffer</u>	4
<u>NAND 2-input</u>	4
<u>NOR 2-input</u>	4
<u>AND 2-input</u>	6
<u>OR 2-input</u>	6
<u>NAND 3-input</u>	6
<u>NOR 3-input</u>	6
<u>MUX 2-input with TG</u>	6
<u>MUX 4-input with TG</u>	18
<u>MUX 4-input</u>	24
<u>1-bit Adder full</u>	28
<u>1-bit Adder-subtractor</u>	48
<u>8-bit multiplier</u>	3,000
<u>16-bit multiplier</u>	9,000
<u>32-bit multiplier</u>	21,000



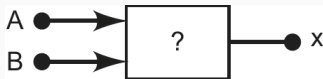
Cerebras WSE-2
46,225mm² Silicon
2.6 Trillion transistors



Largest GPU
826mm² Silicon
54.2 Billion transistors

Tabelas-verdade

Uma **tabela-verdade** descreve como a saída x de um circuito lógico depende dos níveis lógicos das entradas (e.g., A e B).



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

- A **lista das combinações possíveis é usualmente uma sequência de contagem binária**; logo, é fácil preencher uma tabela-verdade sem esquecer nenhuma combinação.
- Caso o circuito lógico tenha mais entradas, basta adicionar mais colunas (e.g., C , D , E ...).
- O número de linhas é 2^N , sendo N o número de entradas.

Operação e porta OR

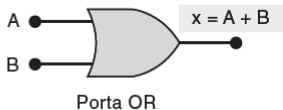
Na álgebra booleana, a operação OR é representada por '+', i.e.,

$$x = A + B$$

a qual é lida como ' x é igual a A OU B'.

Com respeito a **tabela-verdade e símbolo**,

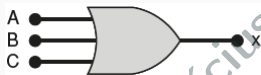
OR		
A	B	$x = A + B$
0	0	0
0	1	1
1	0	1
1	1	1



Note que x assume nível lógico 1 caso ao menos uma entrada seja 1;
caso contrário, x assume nível lógico 0.

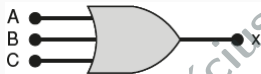
Operação e porta OR

Exemplo: Determine a saída x para uma porta OR de 3 entradas (A , B e C).



Operação e porta OR

Exemplo: Determine a saída x para uma porta OR de 3 entradas (A , B e C).

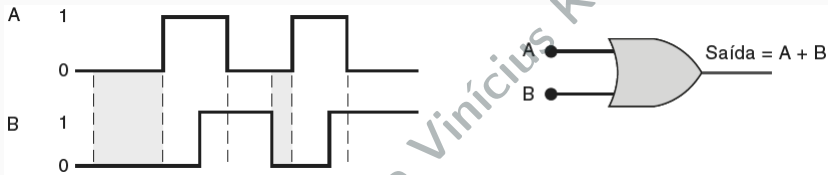


R:

A	B	C	x
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

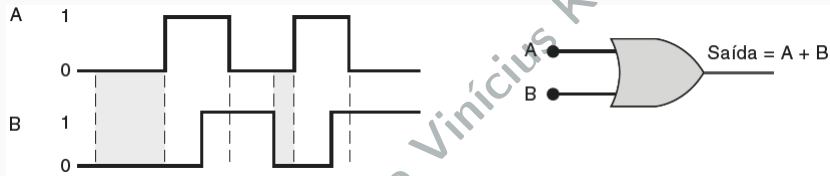
Operação e porta OR

Exemplo: Determine a saída da porta OR nos diferentes instantes de tempo indicados.

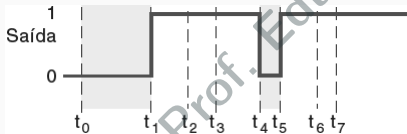


Operação e porta OR

Exemplo: Determine a saída da porta OR nos diferentes instantes de tempo indicados.



R:



Entre t_0-t_1 e t_3-t_4 , a saída assume nível lógico 0; em contraste, assume nível lógico 1 em todos os outros intervalos.

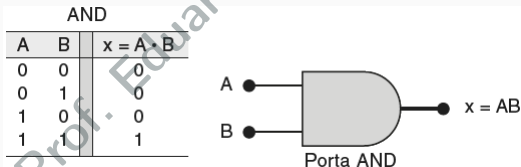
Operação e porta AND

Na álgebra booleana, a operação **AND** é representada por **'.'** (comumente omitido por simplicidade), i.e.,

$$x = A \cdot B$$

a qual é lida como ' **x é igual a A E B** '.

Com respeito a **tabela-verdade e símbolo**,



Note que x assume nível lógico 1 caso todas as entradas sejam 1;

caso contrário, assume nível lógico 0.

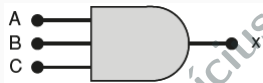
Operação e porta AND

Exemplo: Determine a saída x para uma porta AND de 3 entradas (A , B e C).



Operação e porta AND

Exemplo: Determine a saída x para uma porta AND de 3 entradas (A , B e C).

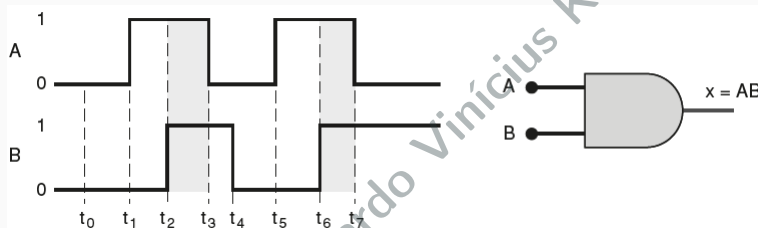


R:

A	B	C	x
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

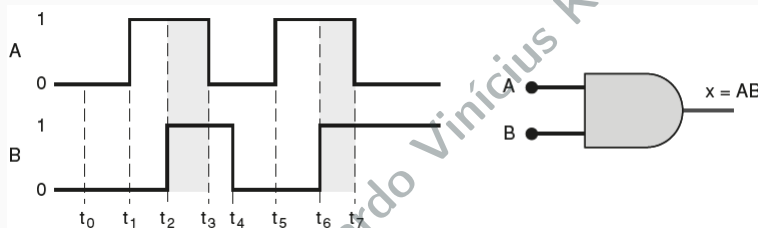
Operação e porta AND

Exemplo: Determine a saída da porta AND nos diferentes instantes de tempo indicados.

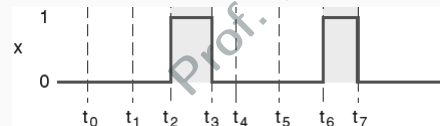


Operação e porta AND

Exemplo: Determine a saída da porta AND nos diferentes instantes de tempo indicados.



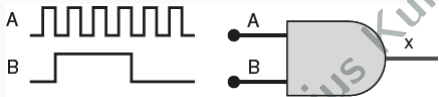
R:



A saída assume nível lógico 1 apenas entre t_2 – t_3 e t_6 – t_7 .

Operação e porta AND

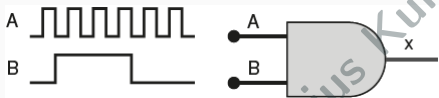
Exemplo: Considerando



- a) Qual a forma de onda da saída da porta AND?
- b) E, caso a entrada B seja mantida em nível lógico 0?

Operação e porta AND

Exemplo: Considerando



a) Qual a forma de onda da saída da porta AND?

R:



b) E, caso a entrada B seja mantida em nível lógico 0?

R: Nesse caso, a porta AND é usada como circuito inibidor, já que $B = 0$ implica $x = 0$; por outro lado, quando $B = 1$, tem-se a condição de habilitação, fazendo $x = A$.

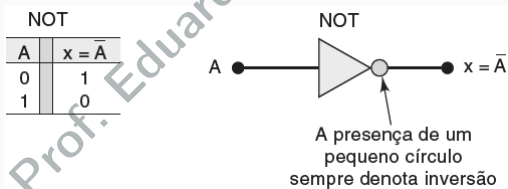
Operação e porta NOT

Na álgebra booleana, a operação NOT é representada por ‘ \neg ’ (ou, eventualmente, por ‘ $'$ ’), i.e.,

$$x = \overline{A} \quad \text{ou} \quad x = A'$$

a qual é lida como ‘ x é igual a A negado’ (complemento de A).

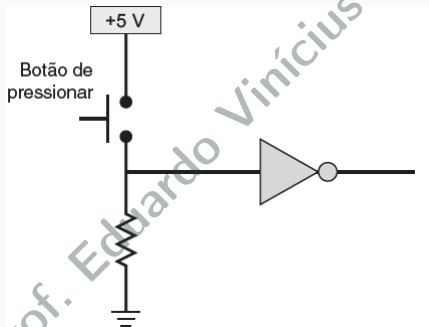
Com respeito a **tabela-verdade e símbolo**,



Note que x assume o nível lógico oposto ao aplicado na entrada A .

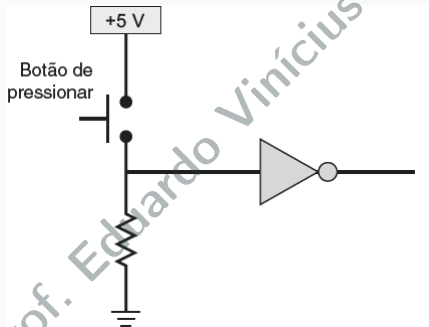
Operação e porta NOT

Exemplo: Explique o funcionamento da porta NOT na aplicação (típica) ilustrada na figura.



Operação e porta NOT

Exemplo: Explique o funcionamento da porta NOT na aplicação (típica) ilustrada na figura.

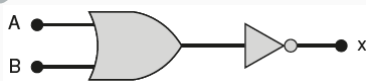
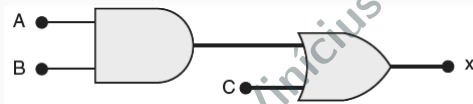


R: Nessa aplicação, quando o botão é pressionado, tem-se nível lógico 0 na saída da porta NOT (inversora).

Qualquer circuito lógico pode ser descrito usando as 3 operações booleanas básicas (OR, AND e NOT).

Descrevendo circuitos lógicos algebricamente

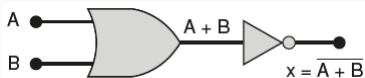
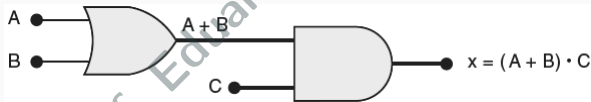
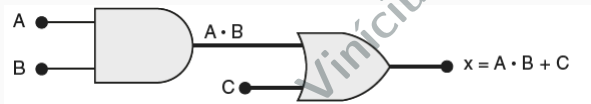
Exemplo: Determine as expressões booleanas que descrevem a relação entre as entradas e a saída dos seguintes circuitos lógicos:



Descrevendo circuitos lógicos algebricamente

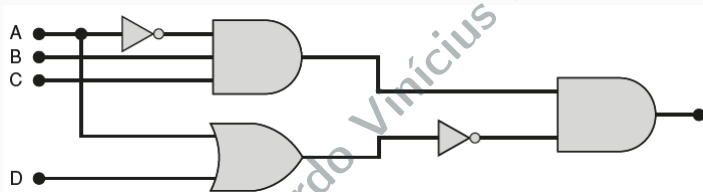
Exemplo: Determine as expressões booleanas que descrevem a relação entre as entradas e a saída dos seguintes circuitos lógicos:

R:



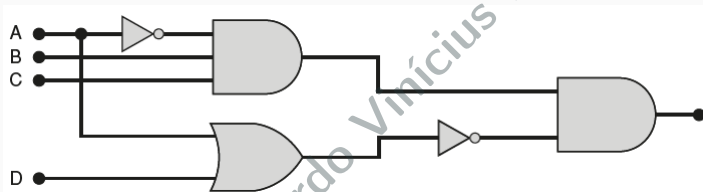
Descrevendo circuitos lógicos algebricamente

Exemplo: Determine a expressão booleana que descreve o seguinte circuito lógico:



Descrevendo circuitos lógicos algebricamente

Exemplo: Determine a expressão booleana que descreve o seguinte circuito lógico:

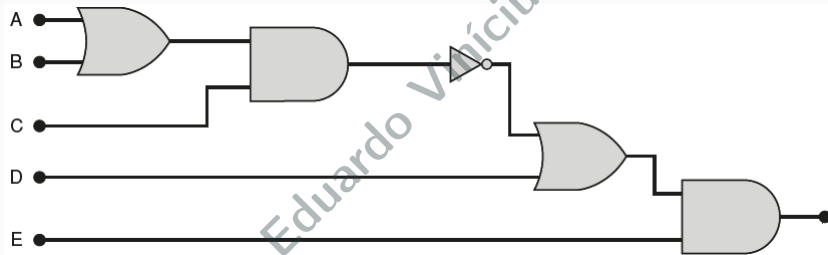


R:

$$x = \overline{A}BC\overline{(A + D)}$$

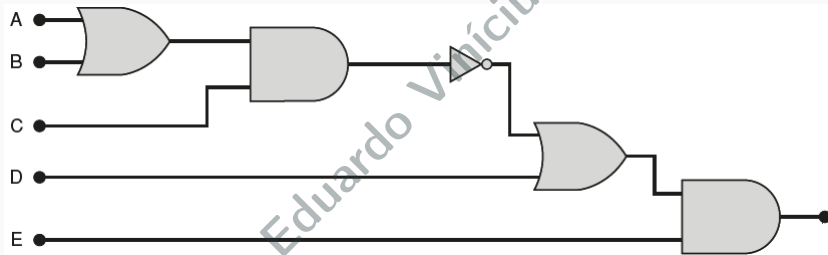
Descrevendo circuitos lógicos algebricamente

Exemplo: Determine a expressão booleana que descreve o seguinte circuito lógico:



Descrevendo circuitos lógicos algebricamente

Exemplo: Determine a expressão booleana que descreve o seguinte circuito lógico:



R:

$$x = [D + \overline{(A + B)C}]E$$

Avaliando as saídas dos circuitos lógicos

A partir da expressão booleana que descreve um circuito, pode-se **determinar o nível lógico da saída** para uma dada entrada. Para tal, deve-se:

- atribuir o nível lógico às entradas**; e, então,
- avaliar a expressão** usando a álgebra booleana.

Eventualmente, podem surgir dúvidas quanto a **qual operação deve ser avaliada primeiro**. Nesse contexto, lembre-se:

Precedência de operadores

Operador	Símbolo	Descrição
NOT	- ou '	Mais alta
AND	.	Média
OR	+	Mais baixa

Avalie primeiro as expressões contidas entre parênteses.

Avaliando as saídas dos circuitos lógicos

Especificamente, **as seguintes regras devem ser obedecidas:**

- 1) Avalie as expressões dentro de parênteses.
- 2) Realize as operações NOT “simples” (uma variável).
 - Caso uma operação NOT envolva mais de uma variável, resolva a expressão e, em seguida, “inverta” o resultado.
- 3) Execute as operações AND e, então, OR.

A ordem das operações é a mesma da álgebra convencional.

Avaliando as saídas dos circuitos lógicos

Exemplo: Determine a saída $x = [D + \overline{(A + B)C}]E$ quando

a) $A = B = 0$ e $C = D = E = 1$

b) $A = B = E = 0$ e $C = D = 1$

Lembre-se da precedência de operadores, i.e., $() \rightarrow \text{NOT} \rightarrow \text{AND} \rightarrow \text{OR}$.

kuhn@utfpr.edu.br | youtube.com/@eduardokuhn87

Avaliando as saídas dos circuitos lógicos

Exemplo: Determine a saída $x = [D + \overline{(A + B)C}]E$ quando

a) $A = B = 0$ e $C = D = E = 1$

b) $A = B = E = 0$ e $C = D = 1$

R:

a)

$$\begin{aligned}x &= [1 + \overline{(0 + 0)1}]1 \\&= [1 + \overline{0}]1 \\&= 1.\end{aligned}$$

b)

$$\begin{aligned}x &= [1 + \overline{(0 + 0)1}]0 \\&= 0.\end{aligned}$$

Lembre-se da precedência de operadores, i.e., $() \rightarrow \text{NOT} \rightarrow \text{AND} \rightarrow \text{OR}$.

kuhn@utfpr.edu.br | youtube.com/@eduardokuhn87

E, se quiséssemos avaliar/verificar o funcionamento de um dado circuito lógico frente a todos os casos possíveis?

Avaliando as saídas dos circuitos lógicos

A melhor maneira de verificar **como um circuito lógico funciona se dá através de sua tabela-verdade**, a qual lista **todas** as possíveis combinações de entrada.

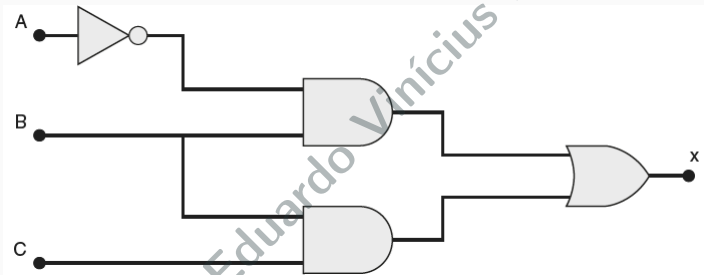
Dessa forma, torna-se possível:

- Analisar uma porta ou combinação lógica de cada vez.
- Conferir se a implementação está correta.
- Identificar pontos de erros do circuito lógico.

Para cada possível combinação de entrada, determina-se o estado lógico em cada ponto (nó) do circuito lógico, inclusive a saída.

Avaliando as saídas dos circuitos lógicos

Exemplo: Determine a tabela-verdade do circuito combinacional ilustrado na figura.



Avaliando as saídas dos circuitos lógicos

R: Considerando que a saída do circuito lógico é dada por

$$x = \overline{A}B + BC$$

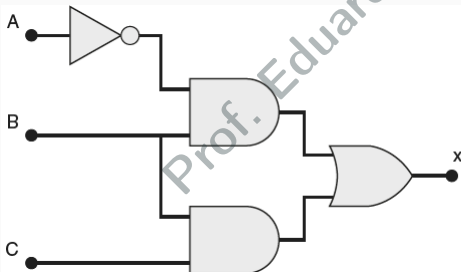
é possível mostrar que

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

Avaliando as saídas dos circuitos lógicos

A partir de uma tabela-verdade, pode-se então **testar o circuito para todas as combinações de entradas**, e.g.,

- Se uma combinação de entrada produzir uma **saída incorreta**, basta verificar o nível lógico de cada nó intermediário.
- Se o nível lógico de um **nó intermediário está correto**, o problema está à direita; caso contrário, à esquerda desse nó.



A	B	C	\overline{A}	$\overline{A}B$	BC	$x = \overline{A}B + BC$
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

O diagrama de um circuito lógico pode ser obtido diretamente a partir de uma dada expressão booleana.

Implementando circuitos a partir de expressões booleanas

Exemplo: Desenhe o diagrama do circuito que implementa a seguinte expressão lógica:

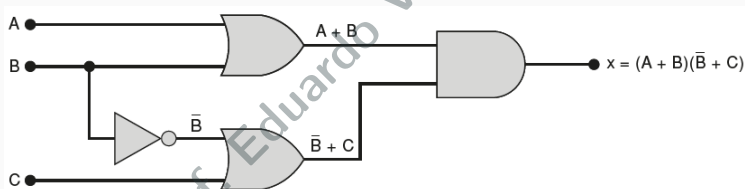
$$x = (A + B)(\overline{B} + C).$$

Implementando circuitos a partir de expressões booleanas

Exemplo: Desenhe o diagrama do circuito que implementa a seguinte expressão lógica:

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

R:



Implementando circuitos a partir de expressões booleanas

Exemplo: Desenhe o diagrama do circuito que implementa a seguinte expressão lógica:

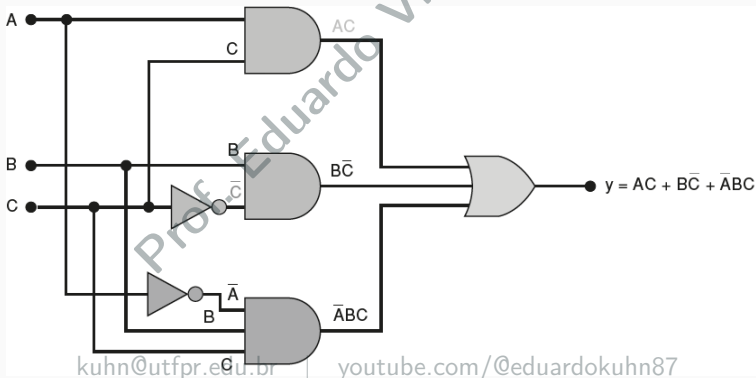
$$y = AC + B\overline{C} + \overline{A}BC.$$

Implementando circuitos a partir de expressões booleanas

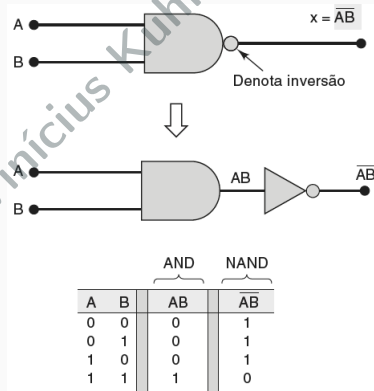
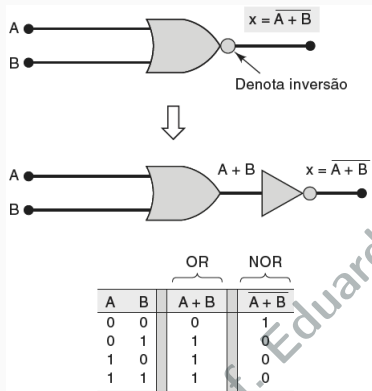
Exemplo: Desenhe o diagrama do circuito que implementa a seguinte expressão lógica:

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

R:



Portas NOR e NAND



Exceto pelo pequeno círculo na saída, que representa a operação de "inversão", os símbolos das portas NOR e NAND são iguais às portas OR e AND, respectivamente.

Exemplo: Implemente o circuito lógico cuja expressão booleanada é dada por

$$x = \overline{AB(C + D)}$$

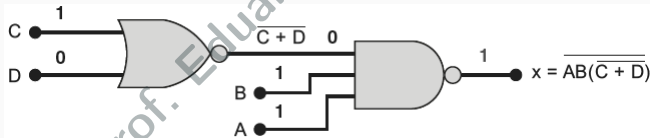
usando apenas portas NOR e NAND. E, em seguida, determine o nível lógico da saída quando $A = B = C = 1$ e $D = 0$.

Exemplo: Implemente o circuito lógico cuja expressão booleanada é dada por

$$x = \overline{\overline{AB(C + D)}}$$

usando apenas portas NOR e NAND. E, em seguida, determine o nível lógico da saída quando $A = B = C = 1$ e $D = 0$.

R:



Lembre-se da precedência de operadores, i.e., $() \rightarrow \text{NOT} \rightarrow \text{AND} \rightarrow \text{OR}$.

kuhn@utfpr.edu.br | youtube.com/@eduardokuhn87

Em álgebra, um teorema é uma proposição
que foi provada verdadeira por uma cadeia
de raciocínio...

Teoremas booleanos (de uma variável)

1) Elementos de identidade:

$$\begin{cases} A + 0 = A \\ A \cdot 1 = A \end{cases}$$

2) Elementos nulos:

$$\begin{cases} A + 1 = 1 \\ A \cdot 0 = 0 \end{cases}$$

3) Lei da involução:

$$\{\bar{\bar{A}} = A$$

4) Lei da idempotência:

$$\begin{cases} A \cdot A = A \\ A + A = A \end{cases}$$

5) Complemento:

$$\begin{cases} A \cdot \bar{A} = 0 \\ A + \bar{A} = 1 \end{cases}$$

6) Princípio da dualidade:

Cada postulado/teorema tem seu par dual obtido intercambiando-se as operações OR “+” e AND “·”, bem como os elementos 0 e 1.

Teoremas booleanos (com mais de uma variável)

7) **Propriedade comutativa:**

$$\begin{cases} A + B = B + A \\ A \cdot B = B \cdot A \end{cases}$$

8) **Propriedade associativa:**

$$\begin{cases} (A + B) + C = A + B + C \\ (A \cdot B) \cdot C = ABC \end{cases}$$

9) **Propriedade distributiva:**

$$\begin{cases} A(B + C) = AB + AC \\ A + BC = (A + B)(A + C) \end{cases}$$

10) **Lei da absorção:**

$$\begin{cases} A + AB = A \\ A(A + B) = A \end{cases}$$

11) **Teorema '(?)':**

$$\begin{cases} A + \overline{A}B = A + B \\ \overline{A} + AB = \overline{A} + B \end{cases}$$

12) **Teoremas de De Morgan:***

$$\begin{cases} \overline{(A + B)} = \overline{A} \cdot \overline{B} \\ \overline{(A \cdot B)} = \overline{A} + \overline{B} \end{cases}$$

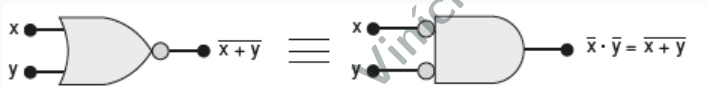
*Os dois teoremas mais importantes da álgebra booleana foram contribuições

Implicações dos Teoremas de De Morgan

Dado que

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

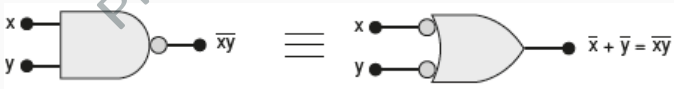
verifica-se que



Analogamente, de

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

observa-se que



Os teoremas booleanos são úteis na simplificação de expressões lógicas...

Simplificação de expressões usando teoremas booleanos

Exemplo: Simplifique as seguintes expressões booleanas:

a) $x = A \cdot \overline{B} \cdot D + A \cdot \overline{B} \cdot \overline{D}$

b) $y = (\overline{A} + B)(A + B)$

c) $z = ACD + \overline{A}BCD$

d) $w = \overline{(\overline{A} + C)(B + \overline{D})}$

Simplificação de expressões usando teoremas booleanos

Exemplo: Simplifique as seguintes expressões booleanas:

a) $x = A \cdot \overline{B} \cdot D + A \cdot \overline{B} \cdot \overline{D}$

R: $y = A\overline{B}$

b) $y = (\overline{A} + B)(A + B)$

R: $z = B$

c) $z = ACD + \overline{A}BCD$

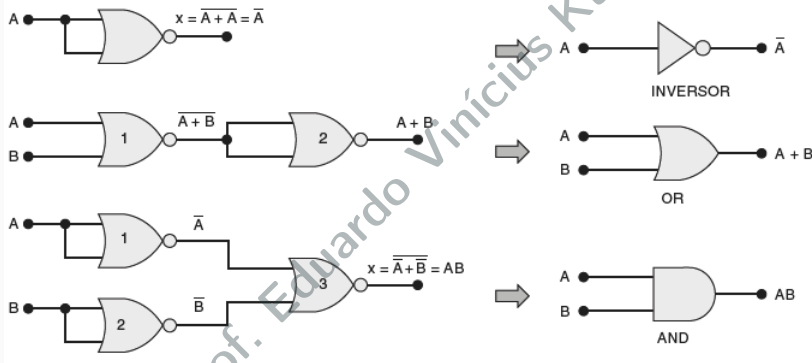
R: $z = (A + B)CD$

d) $w = \overline{(\overline{A} + C)(B + \overline{D})}$

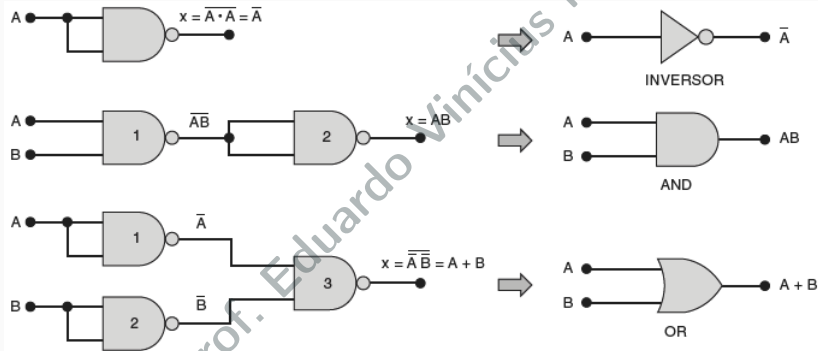
R: $w = A\overline{C} + \overline{B}D$

Qualquer expressão pode ser implementada
usando apenas portas NOR ou NAND.

Universalidade das portas NOR e NAND



Universalidade das portas NOR e NAND



Universalidade das portas NOR e NAND

Exemplo: Implemente a expressão booleana

$$x = AB + CD$$

usando apenas portas NAND.

Prof. Eduardo Vinícius Kuhn

Universalidade das portas NOR e NAND

Exemplo: Implemente a expressão booleana

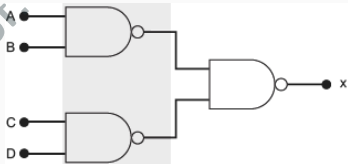
$$x = AB + CD$$

usando apenas portas NAND.

R: A partir da Lei da involução e dos Teoremas de De Morgan,

$$\begin{aligned} x &= \overline{\overline{AB}} + \overline{\overline{CD}} \\ &= \overline{\overline{AB} \cdot \overline{CD}} \end{aligned}$$

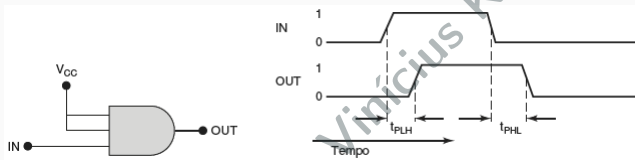
o que resulta em



O atraso de propagação é definido, basicamente, como o tempo que leva para um sistema produzir uma saída apropriada após receber uma entrada.

Atraso de propagação

Visando **mensurar o atraso de propagação** (e.g., porta AND), considere:



Quando a entrada assume nível ALTO, a saída assume um nível ALTO pouco depois (e vice-versa). Logo, observa-se que

- Transições não são verticais (instantâneas)**; por isso, mede-se o ponto de 50% na entrada até 50% na saída.
- O tempo de propagação BAIXO/ALTO t_{PLH} **não é necessariamente o mesmo** de ALTO/BAIXO t_{PHL} .

Atraso de propagação

A velocidade de um circuito lógico está relacionada à característica de atraso de propagação. Portanto, **cabe ao projetista assegurar que o CI é adequado para a aplicação considerada.**

Philips Semiconductors

Product specification

8-input NAND gate

74HC/HCT30

QUICK REFERENCE DATA

GND = 0 V, $T_{amb} = 25^{\circ}\text{C}$, $t_r = t_f = 6\text{ ns}$

SYMBOL	PARAMETER	CONDITIONS	TYPICAL		UNIT
			HC	HCT	
t_{PHL} / t_{PLH}	propagation delay A, B, C, D, E, F, G, H to Y	$C_L = 15\text{ pF}$; $V_{CC} = 5\text{ V}$	12	12	ns
C_i	input capacitance		3.5	3.5	pF
C_{PD}	power dissipation capacitance per gate	notes 1 and 2	15	15	pF



MOTOROLA

QUAD 2-INPUT OR GATE

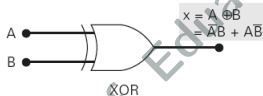
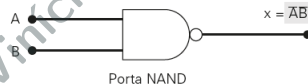
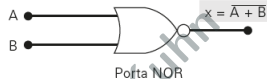
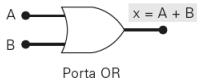
SN54/74LS32

AC CHARACTERISTICS ($T_A = 25^{\circ}\text{C}$)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min	Typ	Max		
t_{PLH}	Turn-Off Delay, Input to Output		14	22	ns	$V_{CC} = 5.0\text{ V}$ $C_L = 15\text{ pF}$
t_{PHL}	Turn-On Delay, Input to Output		14	22	ns	

- As 3 operações básicas foram mostradas, i.e., **OR**, **AND** e **NOT**.
- **Ferramentas úteis de análise, síntese e documentação de circuitos lógicos foram apresentadas**, e.g.,
 - descrição em nossa própria língua;
 - construção de tabelas-verdade;
 - derivação de expressões lógicas;
 - implementação de diagramas esquemáticos; e/ou
 - obtenção de diagramas de tempo.
- Teoremas da álgebra booleana foram introduzidos e **utilizados para simplificar expressões lógicas**.
- A universalidade das portas NOR e NAND foi mostrada, as quais **possibilitam sintetizar quaisquer circuitos lógicos**.
- O conceito de atraso de propagação foi discutido.

Resumo



A	B	OR	NOR	AND	NAND	XOR	XNOR
		$A + B$	$\overline{A + B}$	$A \cdot B$	$\overline{A \cdot B}$	$A \oplus B$	$\overline{A \oplus B}$
0	0	0	1	0	1	0	1
0	1	1	0	0	1	1	0
1	0	1	0	0	1	1	0
1	1	1	0	1	0	0	1

Considerações finais

Exercícios sugeridos:

3.16, 3.19, 3.20, 3.22, 3.24, 3.26-3.33, 3.38-3.40, 3.48 e 3.49

de R.J. Tocci, N.S. Widmer, G.L. Moss, *Sistemas digitais: princípios e aplicações*, 12a ed., São Paulo: Pearson, 2019. → (Capítulo 3)

Para a próxima aula:

R.J. Tocci, N.S. Widmer, G.L. Moss, *Sistemas digitais: princípios e aplicações*, 12a ed., São Paulo: Pearson, 2019. → (Capítulo 4)

Até a próxima aula... =)