PUCRS ESCOLA POLITÉCNICA

4646B-04

FUNDAMENTOS DE SISTEMAS DIGITAIS

Apresentação da Disciplina

Prof. Dr. Fernando Gehm Moraes

Turma 10

MARISTA

EMENTA

- ✓ Estudo e otimização de representações Booleanas: portas lógicas, formas algébricas, mapas de Karnaugh.
- ✓ Estudo de circuitos combinacionais e circuitos sequenciais.
  - Domínio de conceitos básicos do nível de modelagem de sistemas digitais no nível de transferência entre registradores e representações hierárquicas de circuitos digitais.
- ✓ Introdução a arquitetura e organização de computadores

CONTEÚDO

UNIDADE 1: Álgebra Booleana e Otimização Lógica

1. Operadores lógicos
2. Tabela verdade
3. Expressões e equações Booleanas
4. Soma de produtos e produto de somas
5. Teoremas da álgebra Booleana
6. Relações de equivalência
7. Mapas de Karnaugh

PROCEDIMENTOS E RECURSOS:

- ✓ Esta Unidade deve revisar a lógica Booleana incluindo operações de minimização lógica. Deve-se enfatizar a importância da minimização lógica para projetos de hardware. Nesta Unidade, os alunos devem ser orientados para o fato de terem atividades extraclasse para fixação do conteúdo. Neste sentido, deve-se reservar algumas horas-aula para solução de dúvidas.

CONTEÚDO

UNIDADE 2: Circuitos Digitais Combinacionais

1. Introdução a circuitos digitais combinacionais
2. Estudo detalhado de componentes de circuitos combinacionais, como: codificadores, decodificadores, geradores de paridade, comparadores, multiplexadores, somadores, subtratores, multiplicadores e ULAs
3. Representação de circuitos combinacionais em diferentes níveis de abstração
4. Demonstração da modelagem de componentes de circuitos combinacionais em uma linguagem de descrição de hardware

PROCEDIMENTOS E RECURSOS:

- ✓ Esta Unidade deve apresentar o fluxo de projeto de sistemas digitais e as formas de representação dos circuitos combinacionais, incluindo representações em tabelas verdade, formas de onda, representações de circuitos com esquemáticos e exemplos de modelagem em uma linguagem de descrição de hardware. Trabalhos práticos são recomendados, a fim de que o aluno possa se apropriar dos diferentes níveis de representação e abstração de circuitos combinacionais. Nesta Unidade, os alunos devem ser orientados para o fato de terem atividades extraclasse para fixação do conteúdo. Neste sentido, deve-se reservar algumas horas-aula para solução de dúvidas.

CONTEÚDO

UNIDADE 3: Circuitos Digitais Sequenciais

1. Introdução a circuitos digitais sequenciais
2. Comparação entre circuitos sequenciais e combinacionais
3. Estudo de elementos de memória unitária
4. Conceitos e aplicações de outros elementos de memória, como: contadores, temporizadores e registradores de deslocamento
5. Estudo sobre máquinas de estado finitas: Moore e Mealy
6. Demonstração da modelagem de componentes de circuitos sequenciais síncronos em uma linguagem de descrição de hardware
7. Modelagem de um sistema digital utilizando máquina de estados

PROCEDIMENTOS E RECURSOS:

- ✓ Esta Unidade inicia apresentando o ganho fundamental obtido pelo uso de circuitos sequenciais, qual seja, o armazenamento dinâmico de informações no sistema. Deve-se fazer uma discussão entre as diferenças de circuitos sequenciais e combinacionais. A seguir, deve-se aprofundar o estudo no conjunto de componentes sequenciais usados na construção de sistemas digitais, tais como: memórias de diversas naturezas, contadores e máquinas de estados. Por fim, recomenda-se propor um trabalho prático onde os alunos possam modelar um sistema digital cobrindo circuitos sequenciais e combinacionais.

CONTEÚDO

UNIDADE 4: Modelagem de um Sistema Digital

1. Especificação de um sistema digital
2. Definição dos recursos sequenciais e combinacionais para a execução do sistema digital proposto .
3. Definição do controle dos recursos através de uma máquina de estado finitas
4. Modelagem e validação do sistema digital proposto
5. Introdução ao conceito de arquitetura e organização de computadores

PROCEDIMENTOS E RECURSOS:

- ✓ Esta Unidade final tem por objetivo unir os blocos combinacionais (multiplexadores, ULA, somadores) e sequenciais (registradores e máquina de estados) através de um projeto integrador. Os recursos sequenciais e combinacionais necessários ao sistema digital proposto devem ser acompanhados do conceito de bloco de dados (data path), utilizado em processadores. Os recursos de controle (máquina de estado finitas) devem ser acompanhados do conceito de bloco de controle, utilizado em processadores. Ao final desta unidade deve-se apresentar conceitos básicos de arquitetura e organização de computadores, a serem explorados na disciplina seguinte, Organização e Arquitetura de Processadores (OAP)..

https://csed.acm.org/wp-content/uploads/2023/03/Version-Beta-v2.pdf

Computer Science Curricula 2023

Version Beta

March 2023

The Joint Task Force on Computing Curricula

Association for Computing Machinery

(ACM)

IEEE-Computer Society

(IEEE-CS)

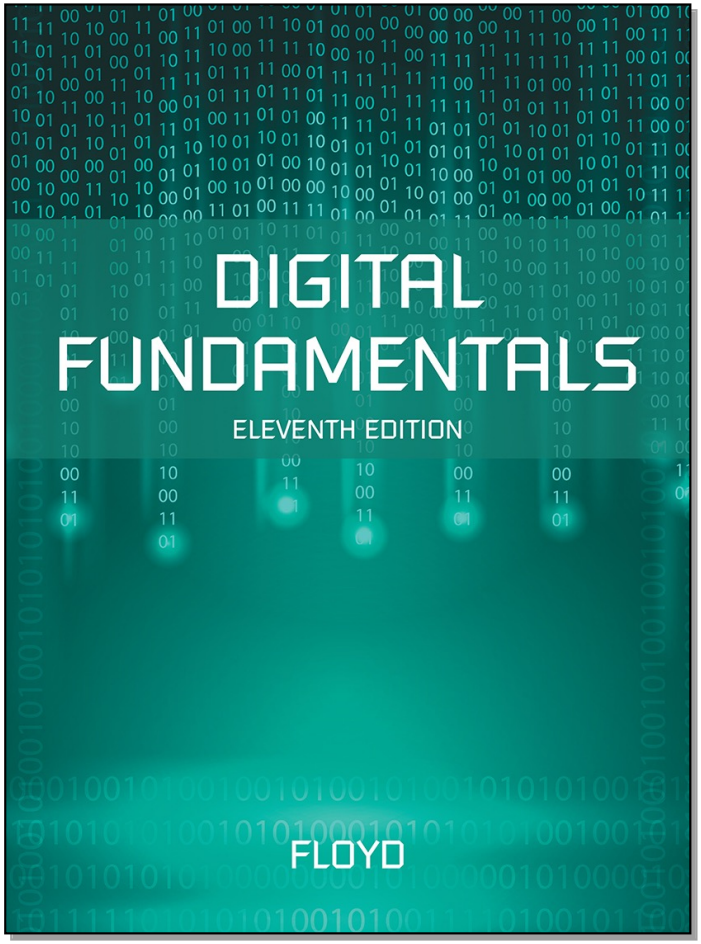
Association for Advancement of Artificial Intelligence

(AAAI)

Os profissionais de computação gastam um tempo considerável escrevendo códigos eficientes para resolver um problema específico em um domínio de aplicação. Com o fim iminente das leis de escala de Moore e Dennard, o **paralelismo no nível do sistema de hardware tem sido cada vez mais utilizado para atender aos requisitos de desempenho de quase todos os sistemas, incluindo a maioria dos hardwares comuns**. **Este afastamento do processamento sequencial exige uma compreensão mais profunda das arquiteturas de computador subjacentes. A arquitetura não pode mais ser tratada como uma caixa onde os princípios de uma podem ser aplicados a outros. Em vez disso, os programadores devem olhar dentro da caixa preta e usar componentes específicos para melhorar o desempenho do sistema e a eficiência energética.**

Digital Fundamentals

ELEVENTH EDITION



CHAPTER 3

Logic Gates

A eletrônica digital usa circuitos que têm dois estados, que são representados por dois níveis de tensão diferentes chamados **ALTO** e **BAIXO**. As tensões representam números no sistema binário.

Em binário, um único número é chamado de ***bit*** (*binary digit*).

Um ***bit*** pode ter o valor de um 1 ou de um 0, dependendo se a tensão for ALTA ou BAIXA.

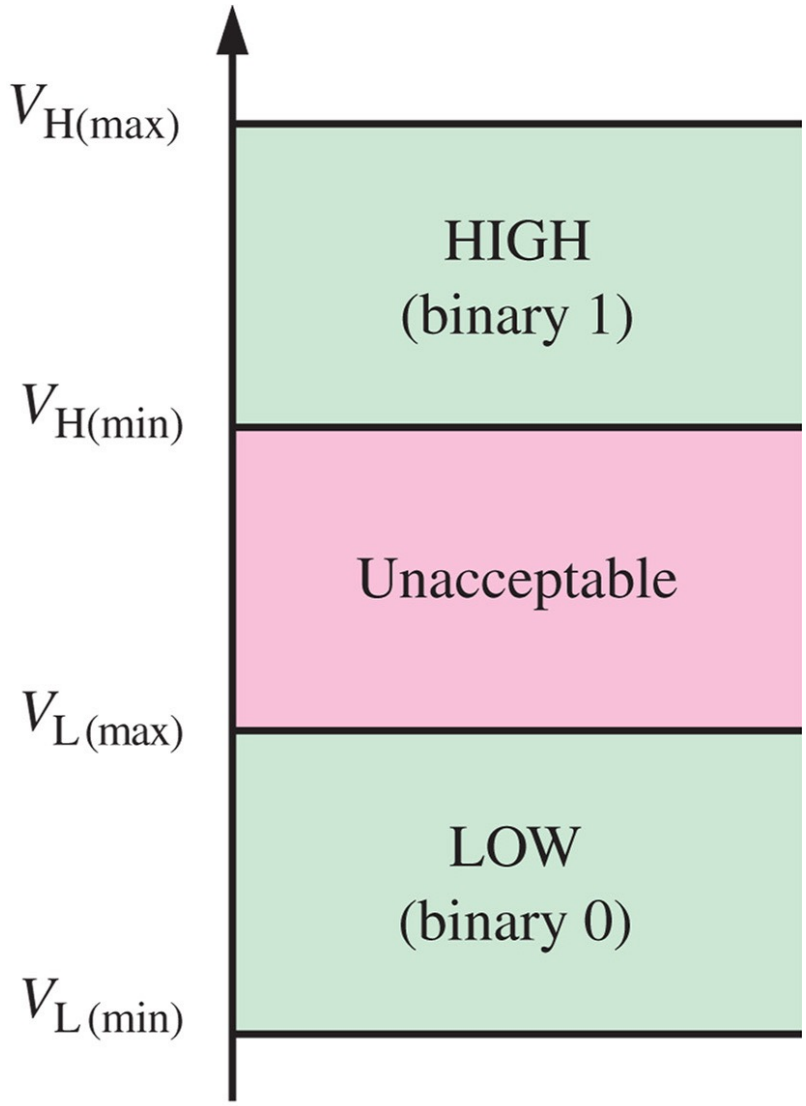


FIGURE 1-6 Logic level ranges of voltage for a digital circuit. (FLOYD)

- As **formas de onda** mudam entre os níveis BAIXO e ALTO
- As formas de onda digitais são formadas por uma série de pulsos

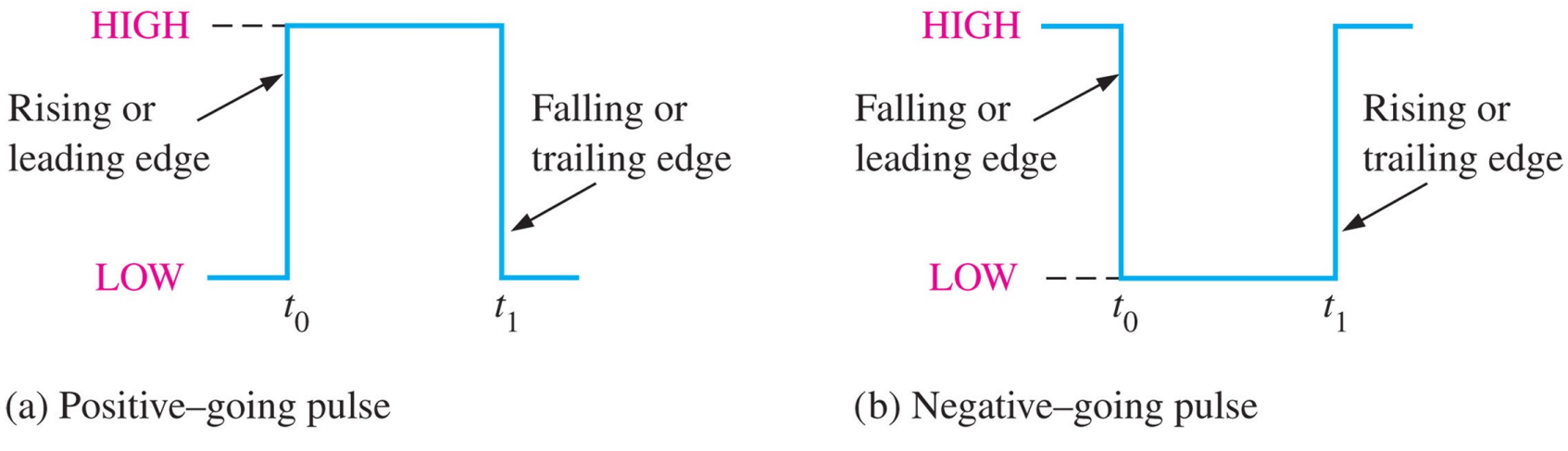
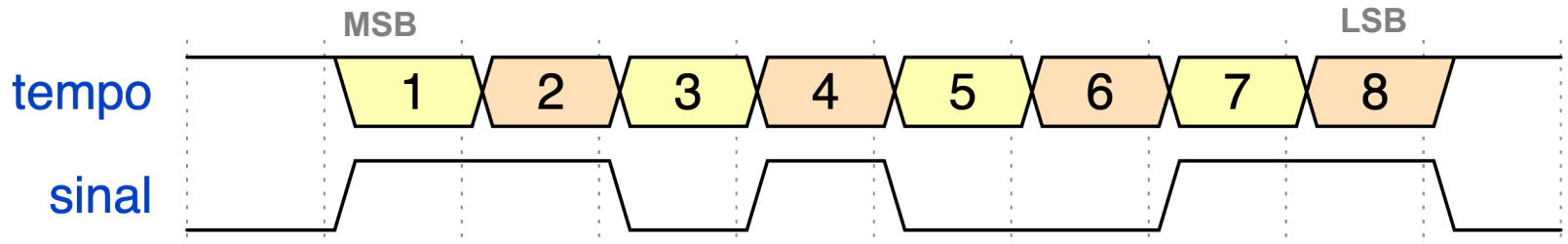


FIGURE 1-7 Ideal pulses.

- Exemplo de forma de onda



1 1 0 1 0 0 1 1

128 64 32 16 8 4 2 1

$(1101\ 0011)_2$ ou $(211)_{10}$ ou $(D3)_{16}$

MSB (Most significant bit): bit mais significativo
LSB (Least significant bit): bit menos significativo

Os pulsos reais não são ideais, mas são descritos pelo tempo de subida, tempo de queda, amplitude e outras características

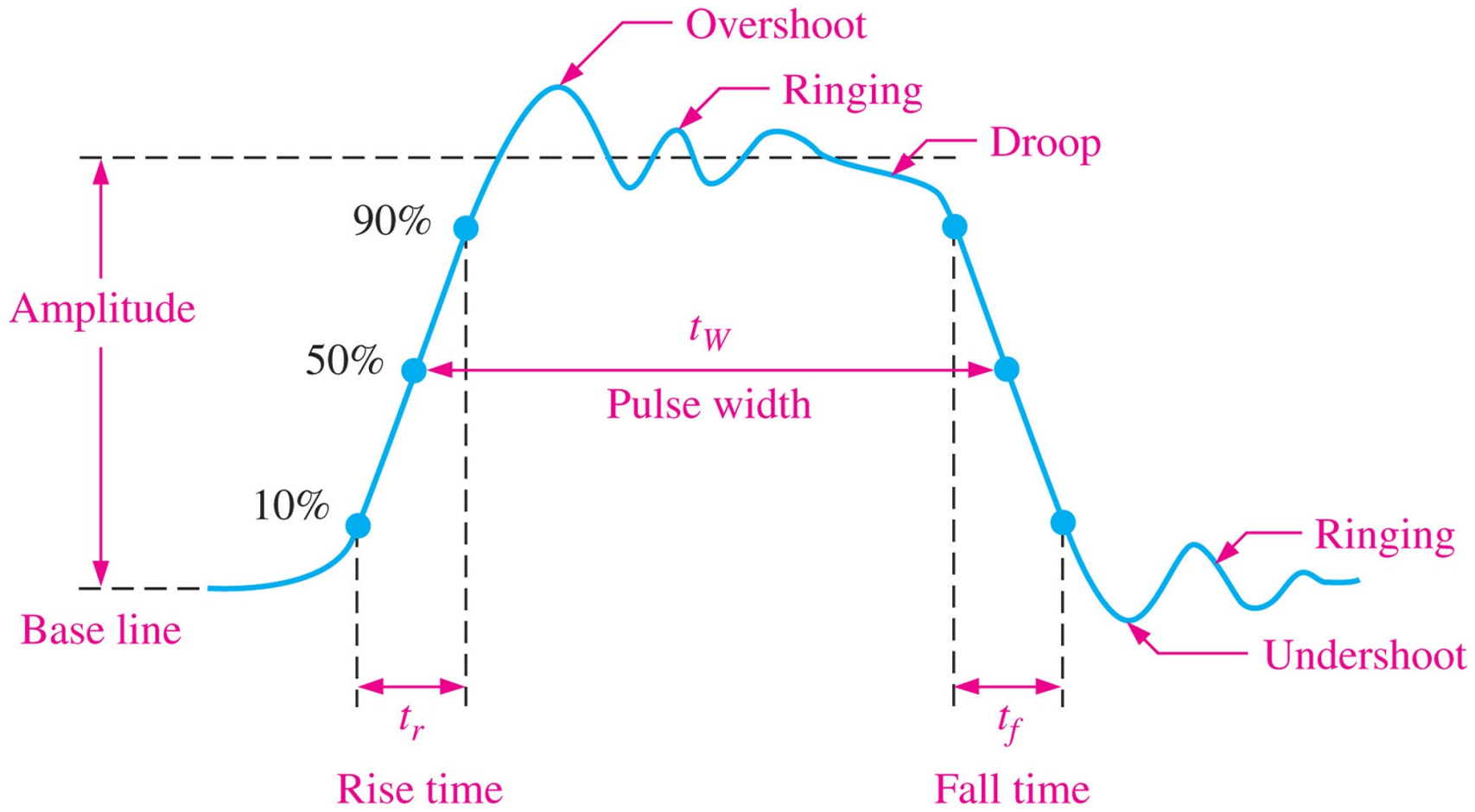
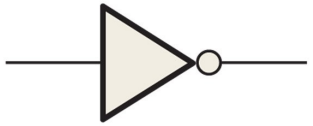
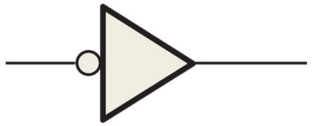
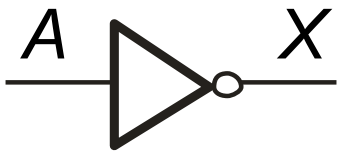


FIGURE 1-8 Nonideal pulse characteristics.

Inversor







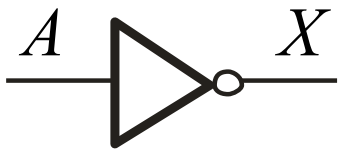
O inversor realiza a operação booleana ***NOT***

Input	Output
A	X
LOW (0)	HIGH (1)
HIGH (1)	LOW(0)

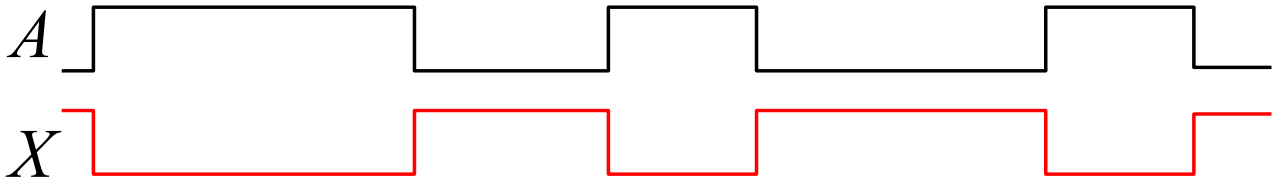
A operação ***NOT*** (complemento) é mostrada com uma barra superior

Assim, a expressão booleana para um inversor é $\boldsymbol{X = \overline{A}}$

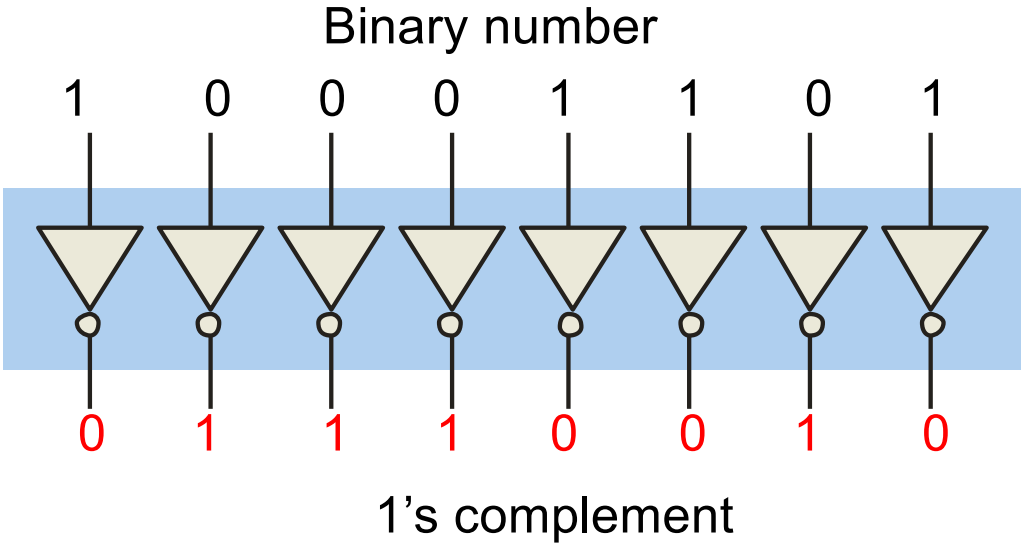
Inversor



Exemplos de formas de onda:



Um grupo de inversores pode ser usado para formar o complemento de 1 (1's) de um número binário:



1's + 1 → negativo

0 0 1 1 → **3**

1 1 0 0 → 1's comp

\+ 1

1 1 0 1 → **-3** (-8+4+1)

AND

- A porta **AND** produz uma saída **'1'** quando todas as entradas são **'1'**; caso contrário, a saída é '0'
- Para uma porta AND de 2 entradas, a tabela verdade é:

Inputs		Output
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

- A operação **AND** geralmente é mostrada com um ponto entre as variáveis, mas pode estar implícita (sem ponto). Assim, a operação AND é escrita como ***X = A . B*** ou ***X = AB***

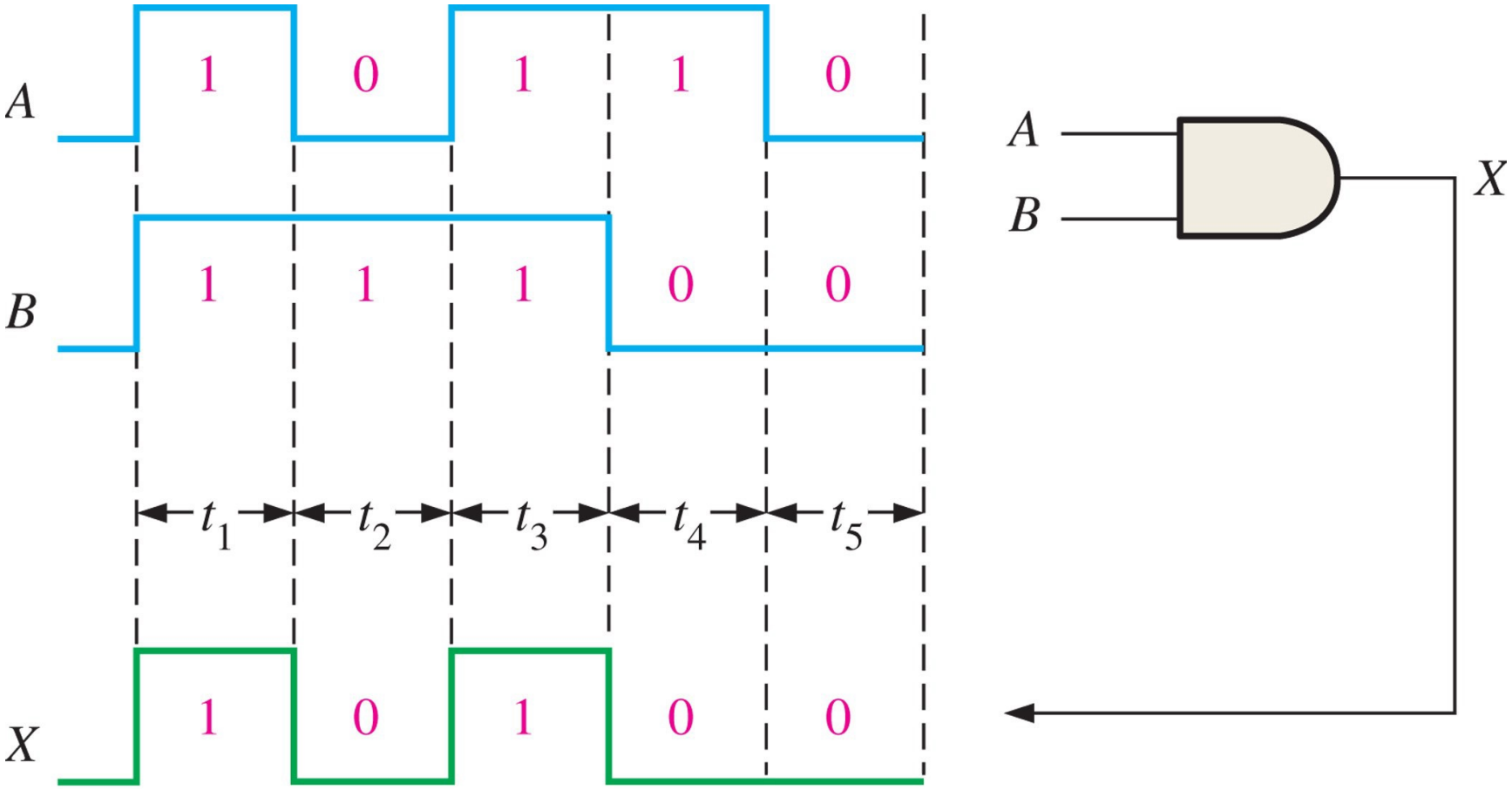


FIGURE 3-10 Example of AND gate operation with a timing diagram showing input and output relationships.

AND

Forma de onda para *X* ?

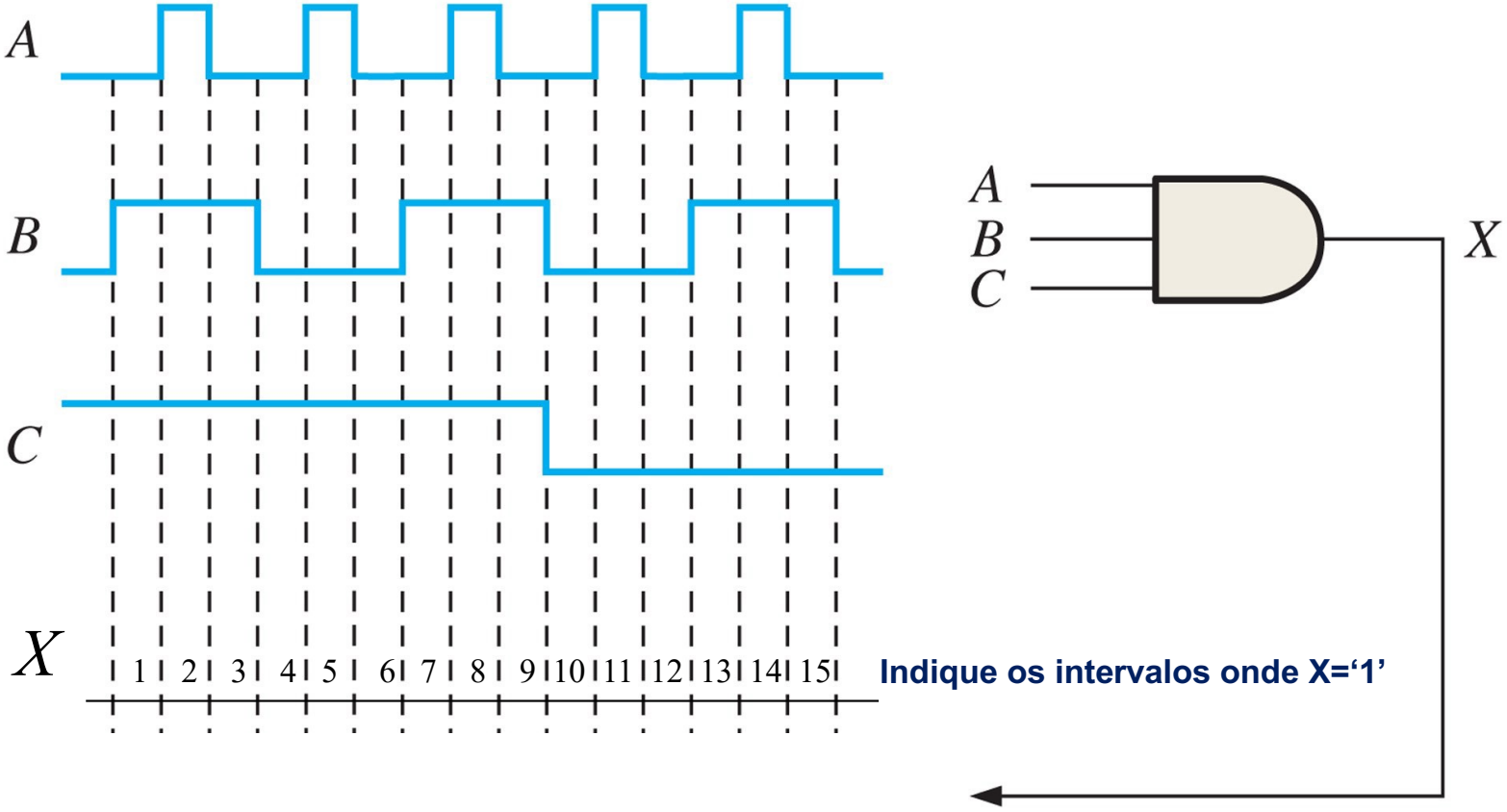


FIGURE 3-13

AND

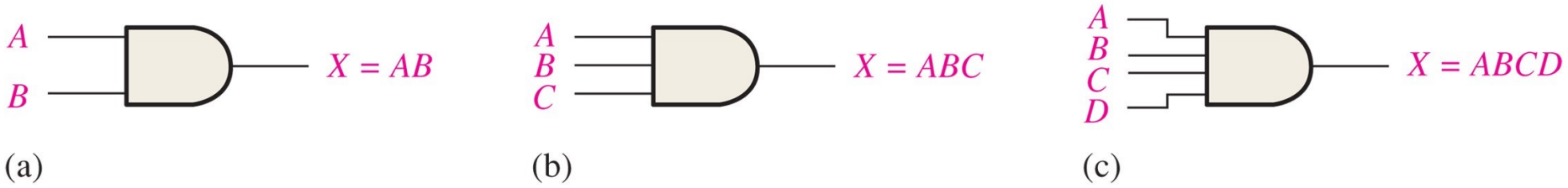
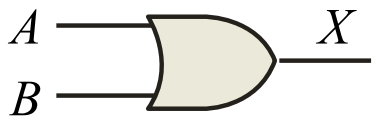


FIGURE 3-15 Boolean expressions for AND gates with two, three, and four inputs.

OR



- A porta **OR** produz uma saída **'1'** se qualquer entrada for '1'; se todas as entradas forem '0', a saída será '0'
- Para uma porta OR de 2 entradas, a tabela verdade é:

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

- A operação **OR** é mostrada com um sinal de mais (+) entre as variáveis. Assim, a operação OU é escrita como ***X = A + B***

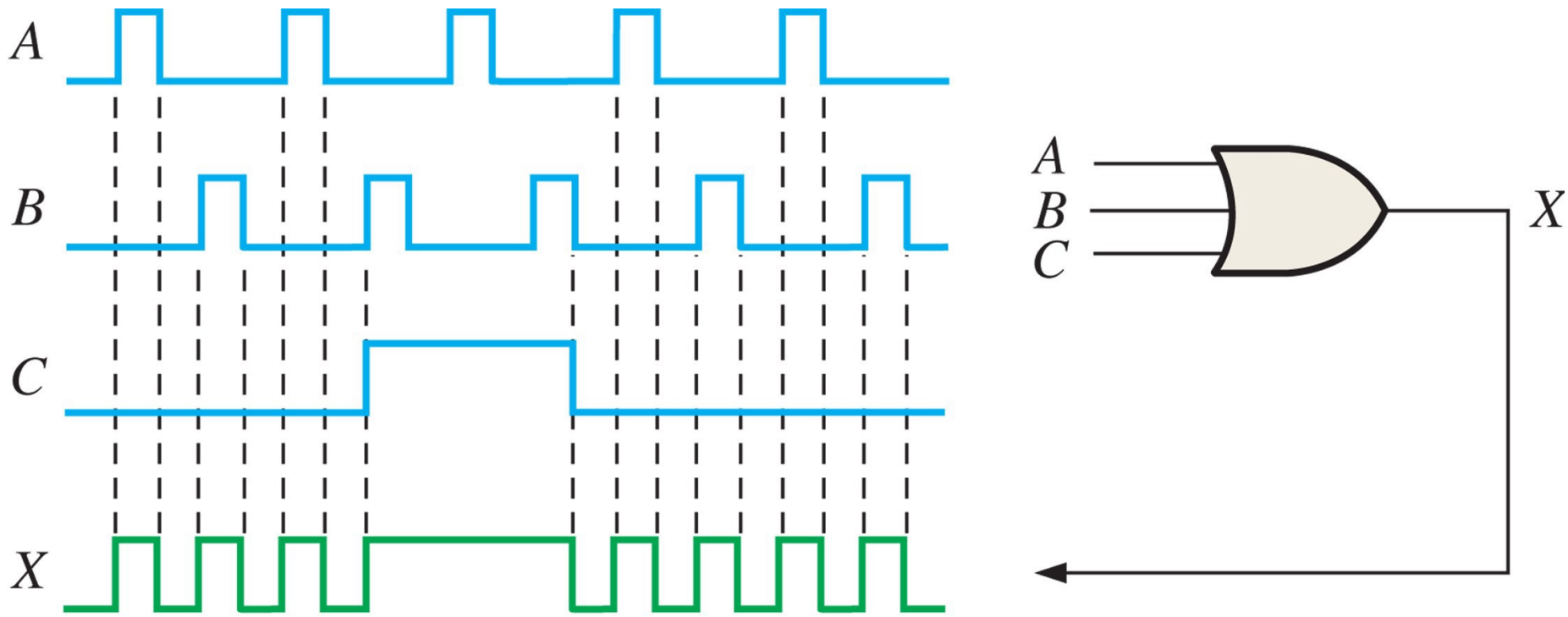


FIGURE 3-23

OR

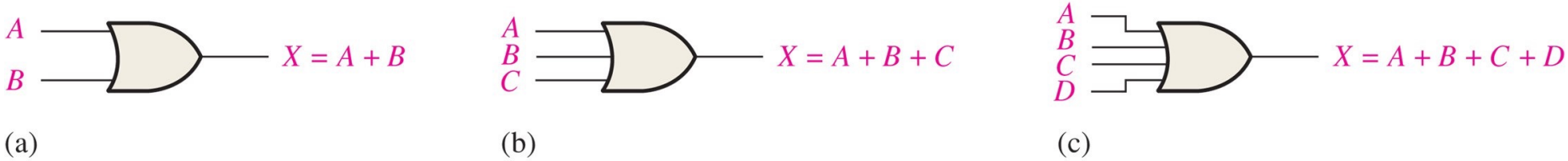


FIGURE 3-24 Boolean expressions for OR gates with two, three, and four inputs.

Uso de AND e OR em programação

A operação AND pode se usada em programação para mascaramento seletivo de bits

'1' para manter o valor

'0' para zerar um determinado bit

Exemplo:
```
  1010 0011
AND 0000 1111
-------------
  0000 0011
```

A operação OU pode ser usada em programação para definir certos bits de um número binário como 1.

As letras ASCII têm um **1** na posição do bit **5** para letras **minúsculas** e um **0** nesta posição para **maiúsculas**. (As posições dos bits são numeradas da direita para a esquerda começando com 0.)

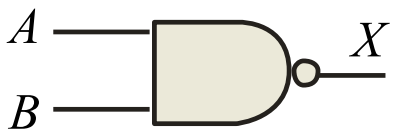
Qual será o resultado se você **OU** uma letra ASCII com a **máscara** de 8 bits **00100000**?

The resulting letter will be lower case.

```
#include<stdio.h>
#include<string.h>

int main(int argc, char **argv)
{
 char str[20];

 strcpy(str, argv[1]);

 for(int  i=0;i<=strlen(str);i++)
      if(str[i]>='A' && str [i]<='Z')
                str[i] = str[i] | 0x20;

 printf("\nInput     String: %s",argv[1]);
 printf("\nLowercase String: %s",str);

 for(int  i=0;i<=strlen(str);i++)
     if(str[i]>='a' &&  str[i]<='z')
                str[i] = str[i] & 0xDF;

 printf("\nUppercase String: %s\n",str);

 return(0);
}
```

```
 gcc -Wall -o x lower_case.c

./x ABCacc123XY

Input     String: ABCacc123XY
Lowercase String: abcacc123xy
Uppercase String: ABCACC123XY
```

NAND

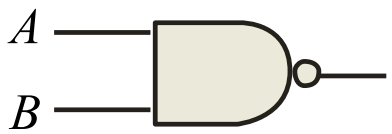


- A porta **NAND** produz uma saída **'0'** quando todas as entradas são '1'; caso contrário, a saída é '1'
- Para uma porta NAND de 2 entradas, a tabela verdade é:

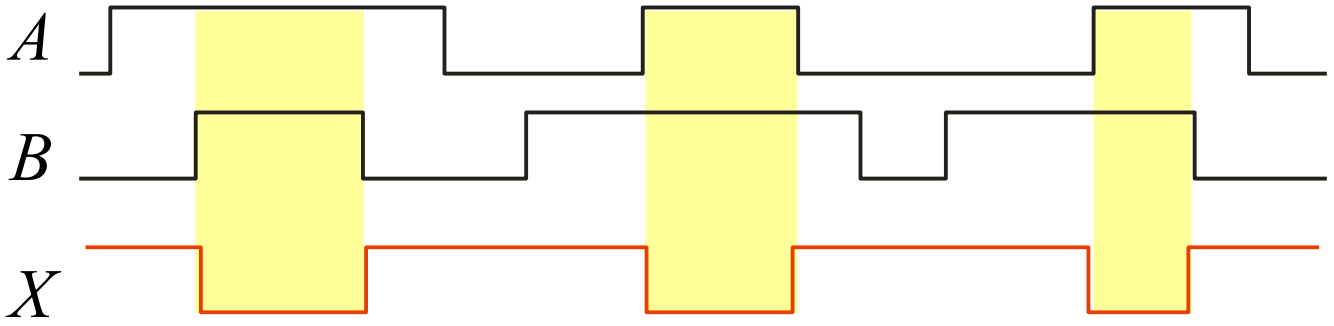
Inputs		Output
A	*B*	*X*
0	0	1
0	1	1
1	0	1
1	1	0

- A operação NAND é mostrada com um ponto entre as variáveis e uma barra cobrindo-as. Assim, a operação NAND é escrita como $\boldsymbol{X = \overline{A \,.\, B}}$ (ou $\boldsymbol{X = \overline{A\, B}}$)

NAND

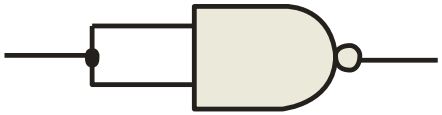


Exemplos de formas de onda:



A porta NAND é particularmente útil porque é uma porta “universal” – todas as outras portas básicas podem ser construídas a partir de portas NAND

Uma porta NAND de 2 entradas com ambas as entradas conectadas é equivalente a qual porta?



NAND

A forma de onda de saída X é ‘0’ somente quando todas as três entradas forem ‘1.

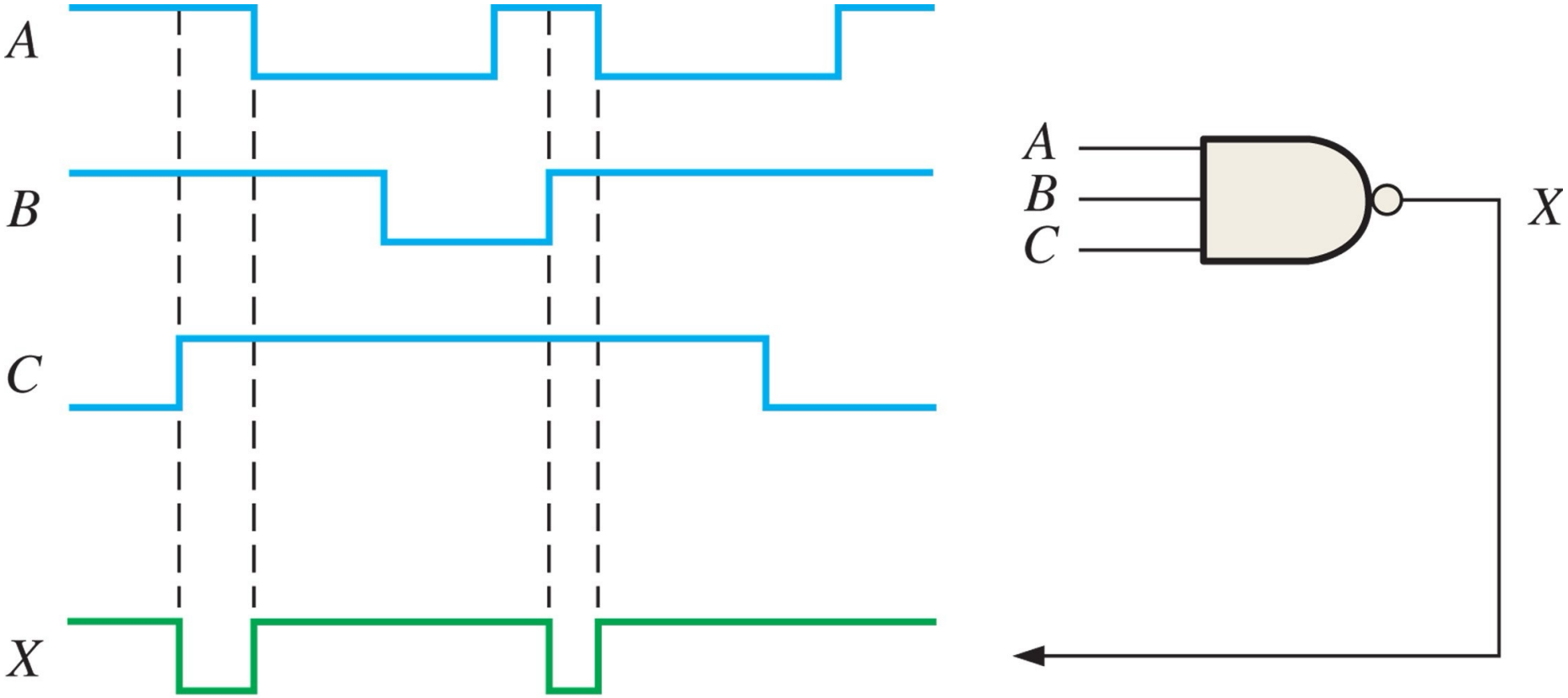
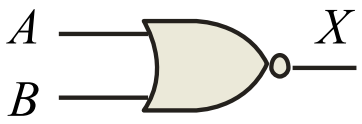


FIGURE 3-29

NOR

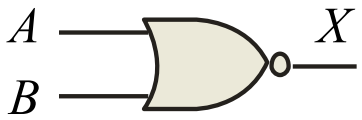


- A porta **NOR** tem na saída o valor ‘0’ quando qualquer entrada for ‘1’; se todas as entradas forem ‘0’, a saída será ‘1’
- Para uma porta NOR de 2 entradas, a tabela verdade é:

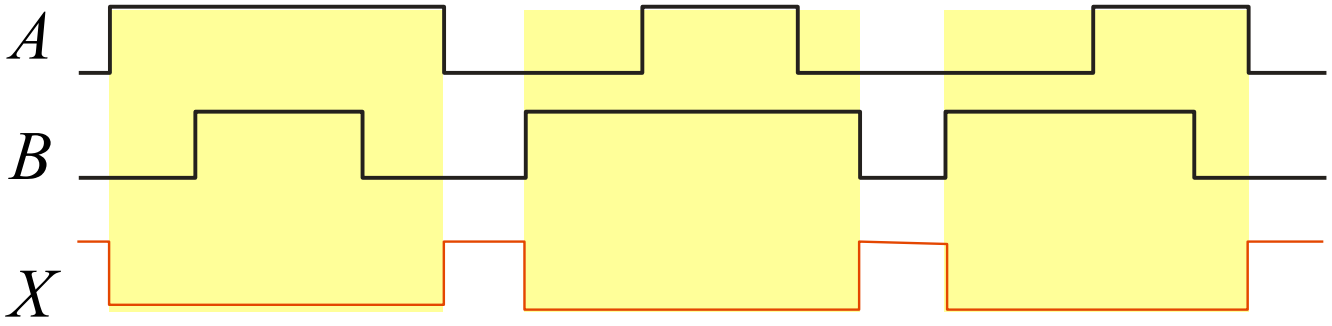
Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	0

- A operação **NOR** é mostrada com um sinal de mais (+) entre as variáveis e uma barra cobrindo-as. Assim, a operação NOR é escrita como $\boldsymbol{X = \overline{A + B}}$

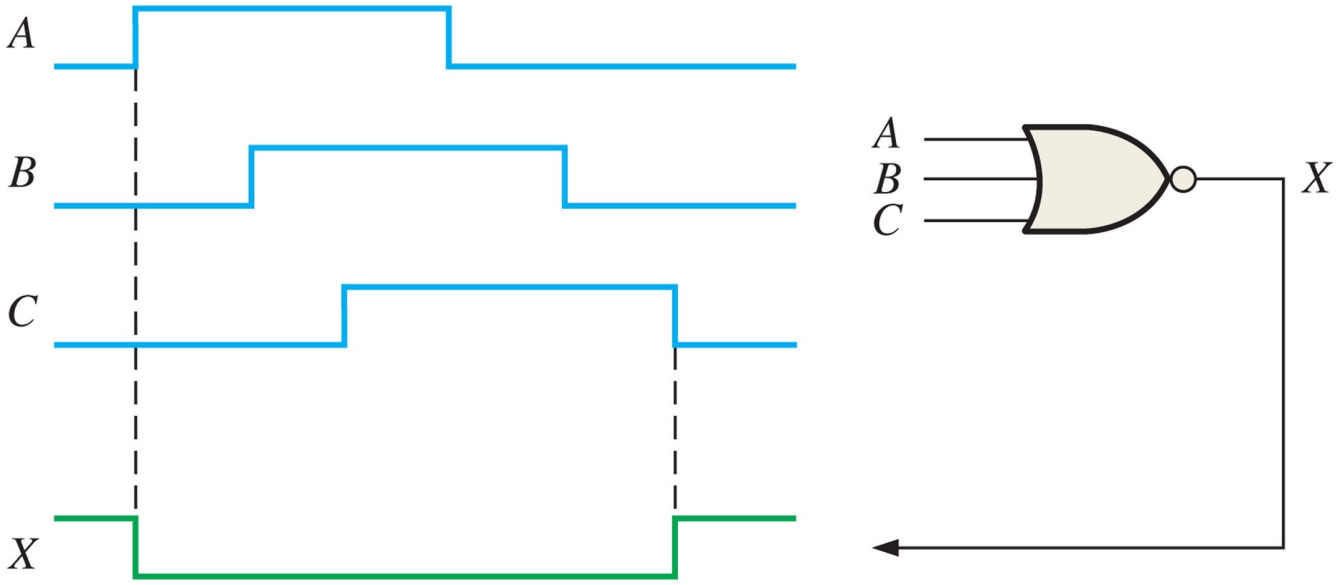
NOR



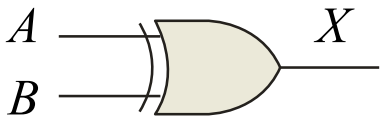
Exemplos de formas de onda:



A NOR gera na saída o valor ‘0’ se qualquer entrada for ‘1’



XOR



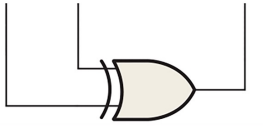
- A porta **XOR** produz uma saída ‘1’ quando ambas as entradas estão em níveis lógicos opostos
- A tabela verdade é:

Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

TABLE 3–13

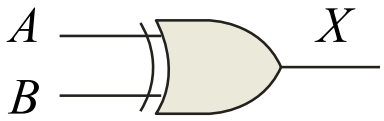
An XOR gate used to add two bits.

Input Bits		Output (Sum)	
A	B	Σ	
0	0	0	
0	1	1	
1	0	1	
1	1	0	(without the 1 carry bit)

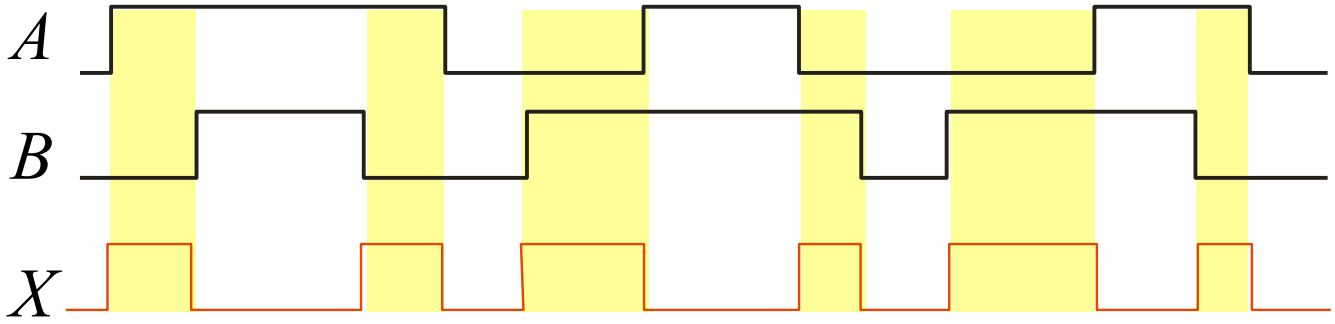


- A operação **XOR** é dada por: $X = \overline{A}.B + A.\overline{B}$
- A operação **XOR** é escrita com um sinal de mais dentro de um círculo entre as variáveis: $X = A \oplus B$

XOR



Exemplos de formas de onda:



Interpretar X como sendo a **soma aritmética** de A + B

Se as formas de onda A e B forem ambas invertidas, como a saída é afetada?

Não há alteração na saída.

$$X = \overline{\overline{A}}.\overline{B} + \overline{A}.\overline{\overline{B}} = A.\overline{B} + \overline{A}.B$$

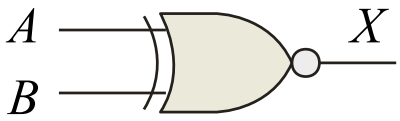
The XOR gate

a	b	c	d	XOR
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Xor de *n* entradas:

- ‘1’ quando número de bits é **impar**
- Corresponde à **soma** dos bits

XNOR



- A porta **XNOR** produz ‘1’ quando ambas as entradas estão no mesmo nível lógico. A tabela verdade é:

Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	1

- A operação **XNOR** é dada por: $\boldsymbol{X = \overline{A}.\overline{B} + A.B}$
- A operação **XNOR** é escrita com um sinal de ‘.’ dentro de um círculo entre as variáveis: $\boldsymbol{X = A \odot B}$

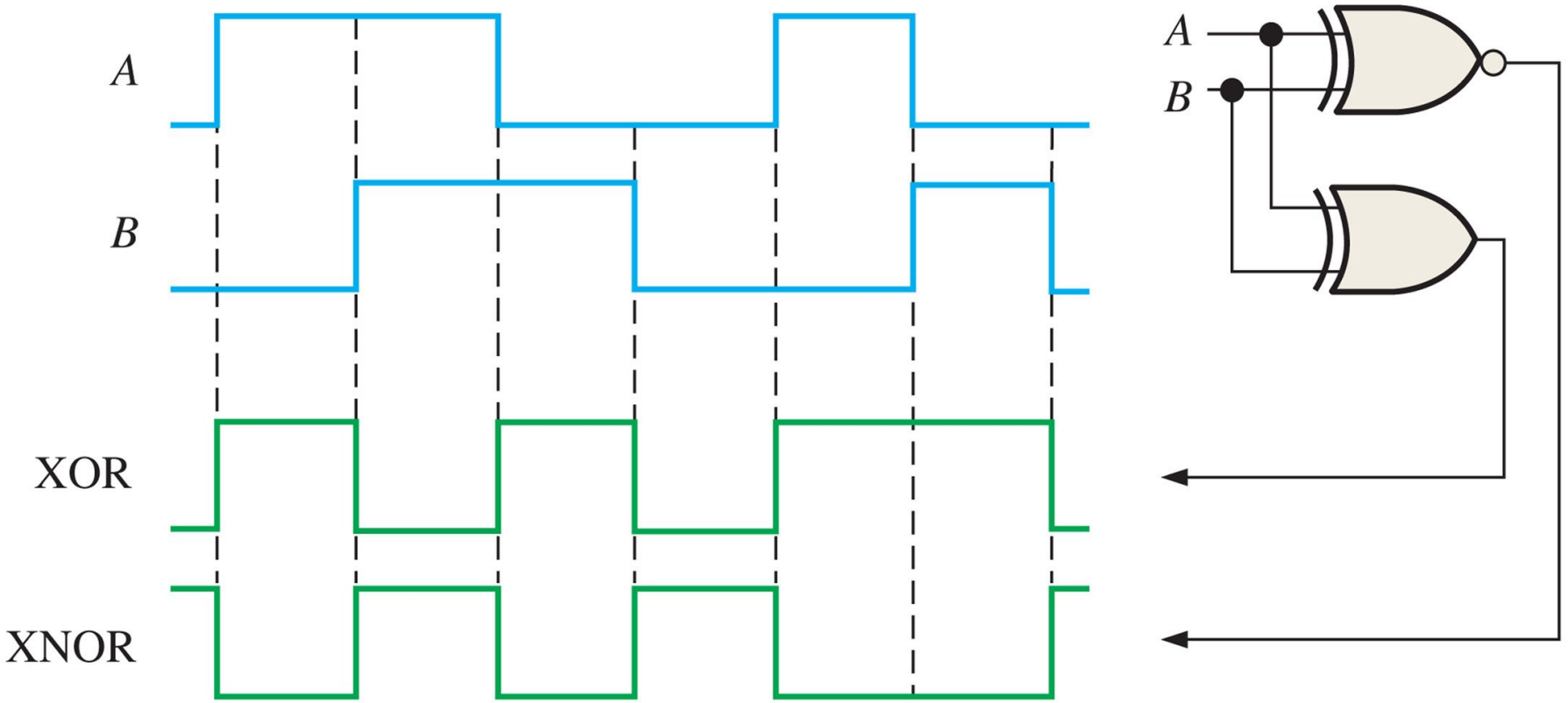
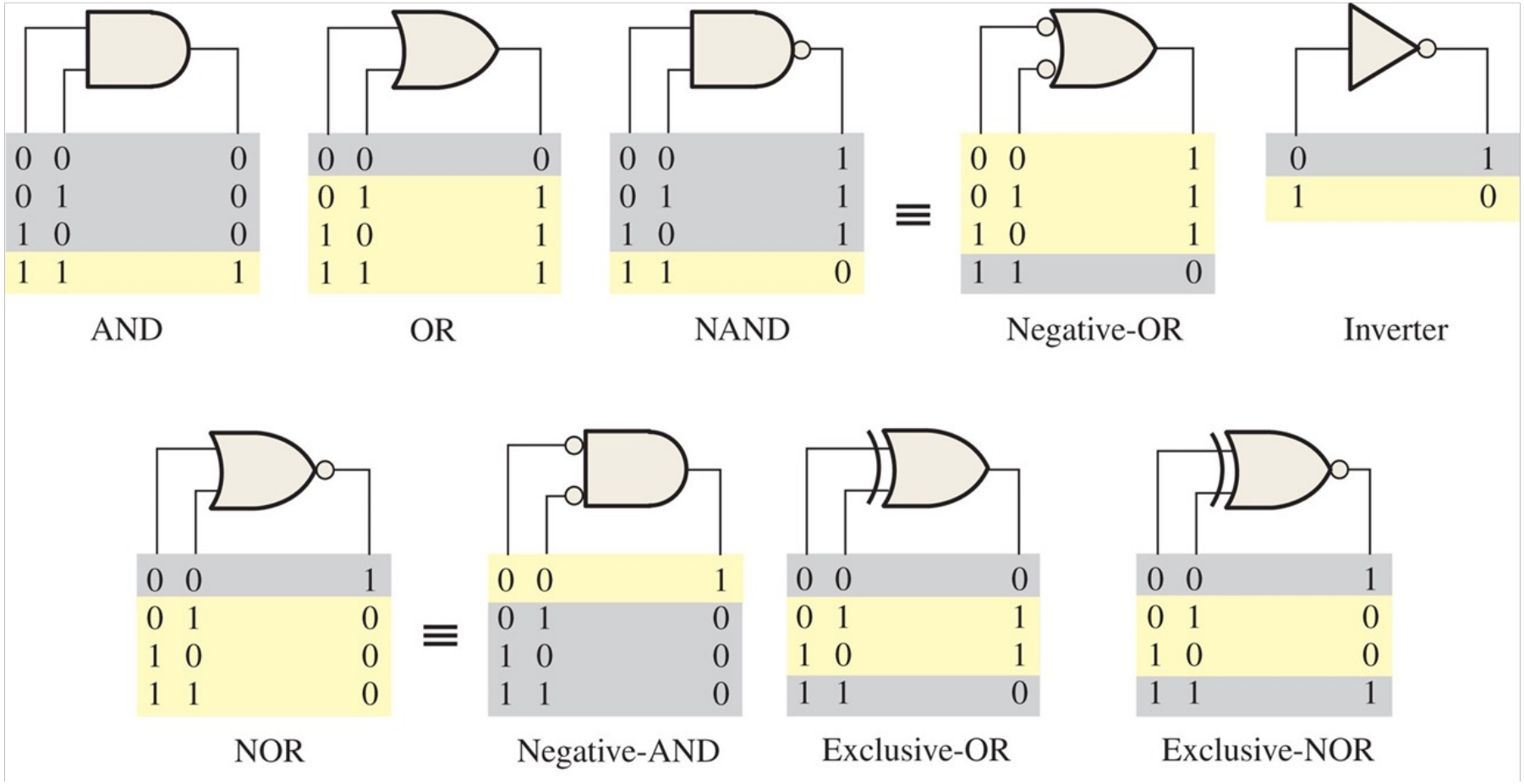


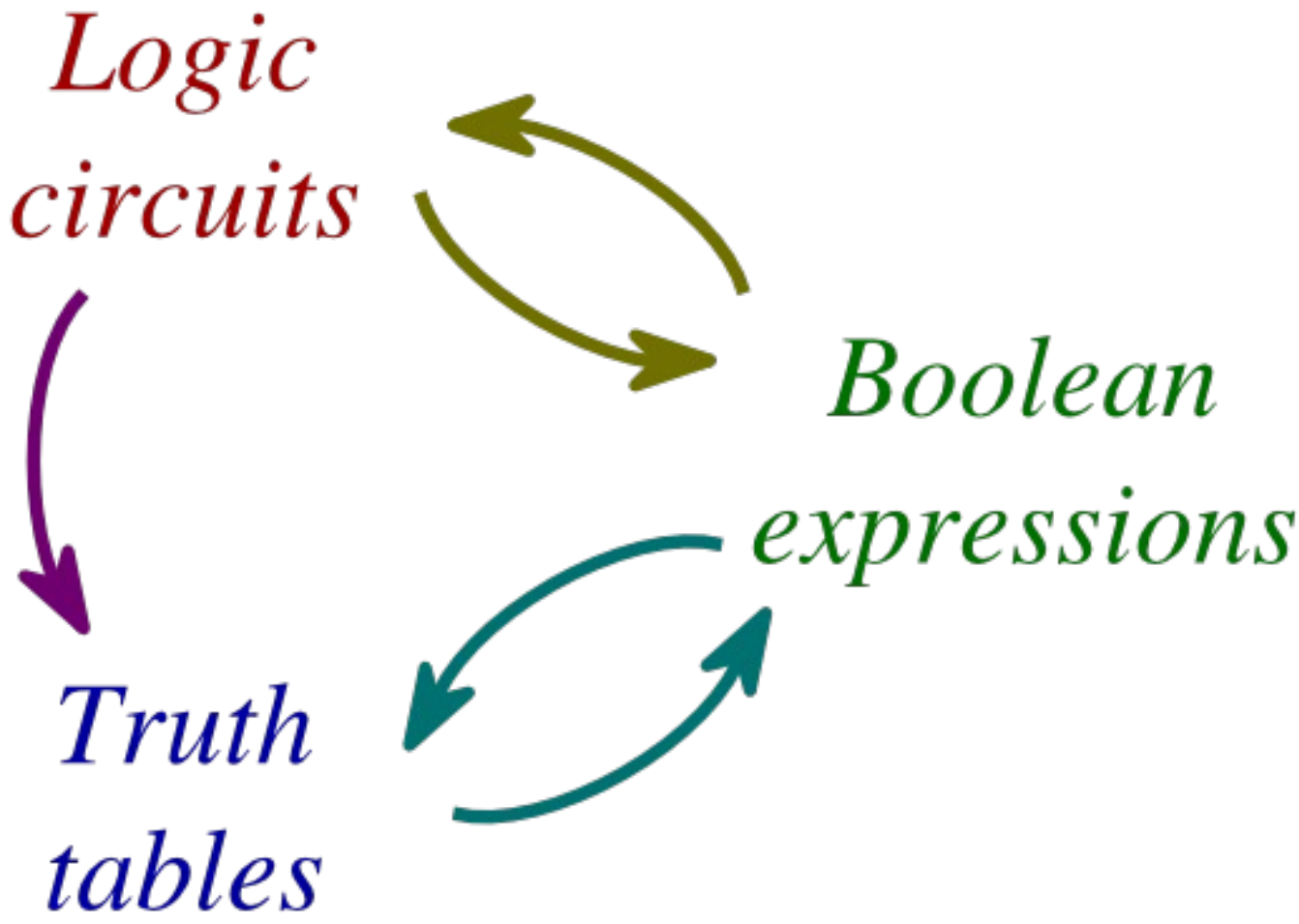
FIGURE 3-48

Resumo das Portas Lógicas

FIGURE 3-75



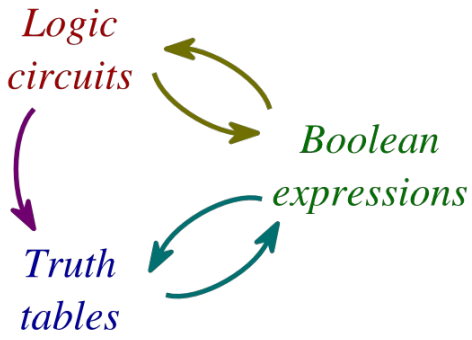
Transformações entre Representações Lógicas



Transformações entre Representações Lógicas

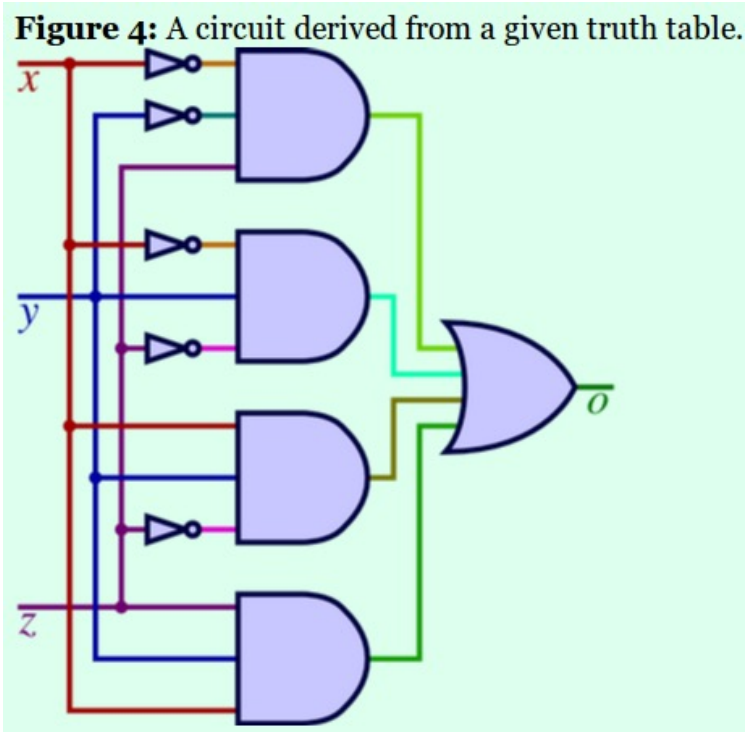
x	y	z	o
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$f = \bar{x}.\bar{y}.z + \bar{x}.y.\bar{z} + x.y.\bar{z} + x.y.z$$



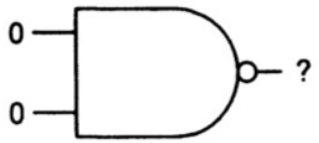
$$f = \sum(1,2,6,7)$$

Somatório de Mintermos

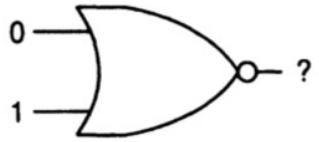


A. What is the value of the output bit in each of the following circuit diagrams?

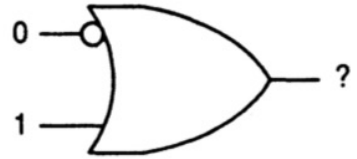
1.



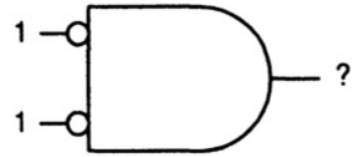
2.



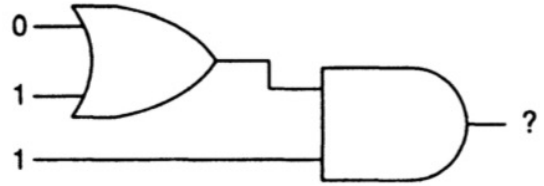
3.



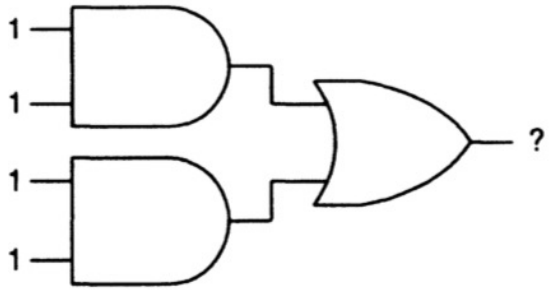
4.



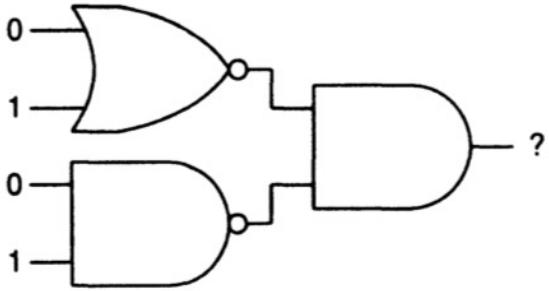
5.



6.



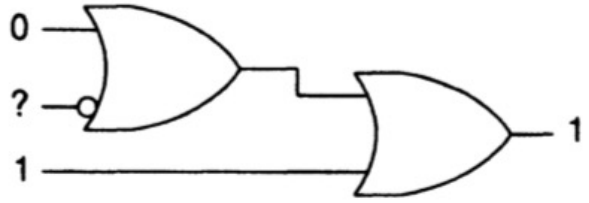
7.



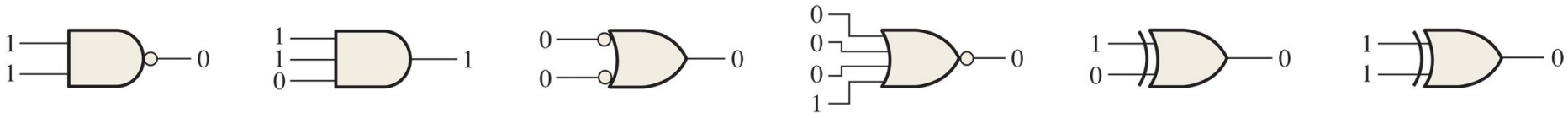
B. If possible, determine the value of the missing input bit in each of the following circuit diagrams:

9.

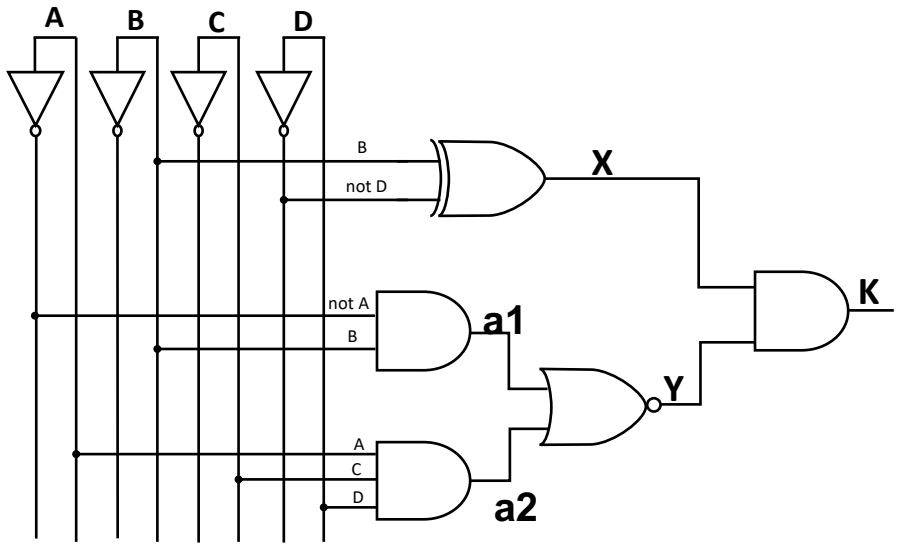
16.



Examine the conditions indicated in Figure 3–92, and identify the faulty gates.



Escrever a equação algébrica do circuito abaixo e preencher tabela verdade



$$K = \overline{(\bar{A}.B + A.C.D)} \ . \ (B \oplus \bar{D})$$

$$K = \sum(0, 2, 8, 10, 13)$$

$$K = \bar{B}.\bar{D} + A.B.\bar{C}.D$$

A	B	C	D	a1	a2	Y	X	K	
0	0	0	0			1	1	1	0
0	0	0	1			1			1
0	0	1	0			1	1	1	2
0	0	1	1			1			3
0	1	0	0	1					4
0	1	0	1	1			1		5
0	1	1	0	1					6
0	1	1	1	1			1		7
1	0	0	0			1	1	1	8
1	0	0	1			1			9
1	0	1	0			1	1	1	10
1	0	1	1		1				11
1	1	0	0			1			12
1	1	0	1			1	1	1	13
1	1	1	0			1			14
1	1	1	1		1		1		15

Simplificação Lógica

Lógica simplificada == hardware mais eficiente

- Menor área de hardware
- Menor o atraso da lógica
- Menor consumo de energia

Próximo conteúdo: simplificação lógica