

4646B-04

FUNDAMENTOS DE
SISTEMAS DIGITAIS

Apresentação da Disciplina

Prof. Dr. Fernando Gehm Moraes

Turma 12



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, subtraidores, 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: Introdução à Organização e Arquitetura de Processadores

1. Introdução ao conceito de arquitetura de computadores; diferenciação entre organizações von Neumann e Harvard; RISC x CISC.
2. Conceitos fundamentais de arquitetura de processadores relacionados a conjuntos de instruções e linguagem de montagem.
3. Organização bloco de dados e bloco de controle para suporte a um conjunto de instruções em um processador simples
4. Execução de programas em linguagem de montagem no processador proposto a partir do uso de uma linguagem de descrição de hardware e sua integração com software

PROCEDIMENTOS E RECURSOS:

- ✓ Esta Unidade final tem por objetivo unir os blocos combinacionais anteriores (multiplexadores, ULA, somadores etc.) e sequenciais (registradores e máquina de estados etc.) de tal forma a produzir um processador simples, com suporte a um conjunto reduzido de instruções. Os conceitos introduzidos nesta unidade, tais como arquitetura load-store, formato regular de instruções e execução multi-ciclo, representam a conexão desta disciplina com os conteúdos da disciplina seguinte, Organização e Arquitetura de Processadores (OAP). Dado o tempo reservado para esta unidade conceitos como modos de endereçamento, modelo de acesso à memória, pseudo-instruções e macro-instruções, e uso de sub-rotinas são tópicos a serem abordados apenas em OAP.

AVALIAÇÃO

$$G1 = 0,2 * IA1 + 0,2 * IA2 + 0,3 * IA3 + 0,3 * IA4$$

- ✓ Datas das avaliação: no Moodle
- ✓ Trabalhos fora do prazo de entrega **não terão valor**. Ou seja, se não entregar no prazo estipulado, a nota do aluno é **zero**.
- ✓ **PLÁGIO E FRAUDE NÃO SERÃO TOLERADOS!**
- ✓ **Grau G1** : Para aprovação em G1 é necessário média *maior ou igual a 7*
- ✓ **Grau G2** : Os alunos que não obtiveram aprovação em G1 devem realizar a prova de G2, desde que: *tenham média de G1 maior ou igual a 4*
- ✓ A **média de G2** é calculada pela fórmula abaixo, onde o aluno que obtiver *média de G2 maior ou igual a 5* *será aprovado*.

$$G2 = \frac{(G1 + \text{nota da prova de G2})}{2}$$

INFORMAÇÕES GERAIS

- ✓ Material disponível somente no Moodle
- ✓ Trabalhos serão entregues via Moodle
- ✓ Atendimento (dúvidas, provas, trabalhos):
 - Por email: fernando.moraes@pucrs.br
 - Favor, IDENTIFICAR A DISCIPLINA colocando antes do assunto: **[FSD] Assunto**
 - Fórum Moodle da disciplina
- ✓ Metodologia de ensino e andamento das aulas:
 - Conteúdo e cronograma (Apresentação da teoria)
 - Aulas práticas visando a fixação do conteúdo

BIBLIOGRAFIA

BÁSICA:

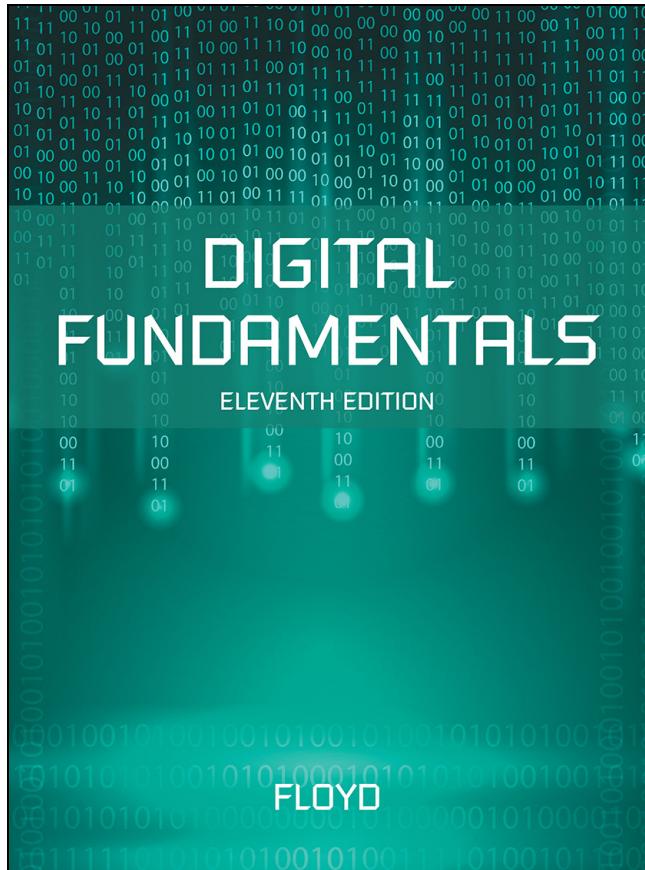
1. FLOYD, T. L. **Sistemas Digitais: Fundamentos e Aplicações**. 9^a edição. 2007.
2. D'AMORE, R. **VHDL: Descrição e Síntese de Circuitos Digitais**. 2^a edição. 2012.
3. TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **Sistemas Digitais: princípios e aplicações**. 11^a edição. São Paulo: Pearson Prentice Hall. 2011.

COMPLEMENTAR:

1. VAHID, F.; LYSECKY, R. **VHDL for Digital Design**. Hoboken, NJ: Wiley. 2007.
2. PEDRONI, V. A. **Eletrônica Digital Moderna e VHDL**. Rio de Janeiro: Elsevier. 2010.
3. BROWN, S. D.; VRANESIC, Z. **Fundamentals of Digital Logic with VHDL Design**. 3^a edição. New York, NY: McGraw-Hill Education. 2008.
4. RUSHTON, A. **VHDL for Logic Synthesis**. 3^a edição. Chichester: John Wiley & Sons. 2011.
5. FLOYD, T. L. **Digital Fundamentals with VHDL**. Boston, MA: Prentice Hall. 2002.

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.

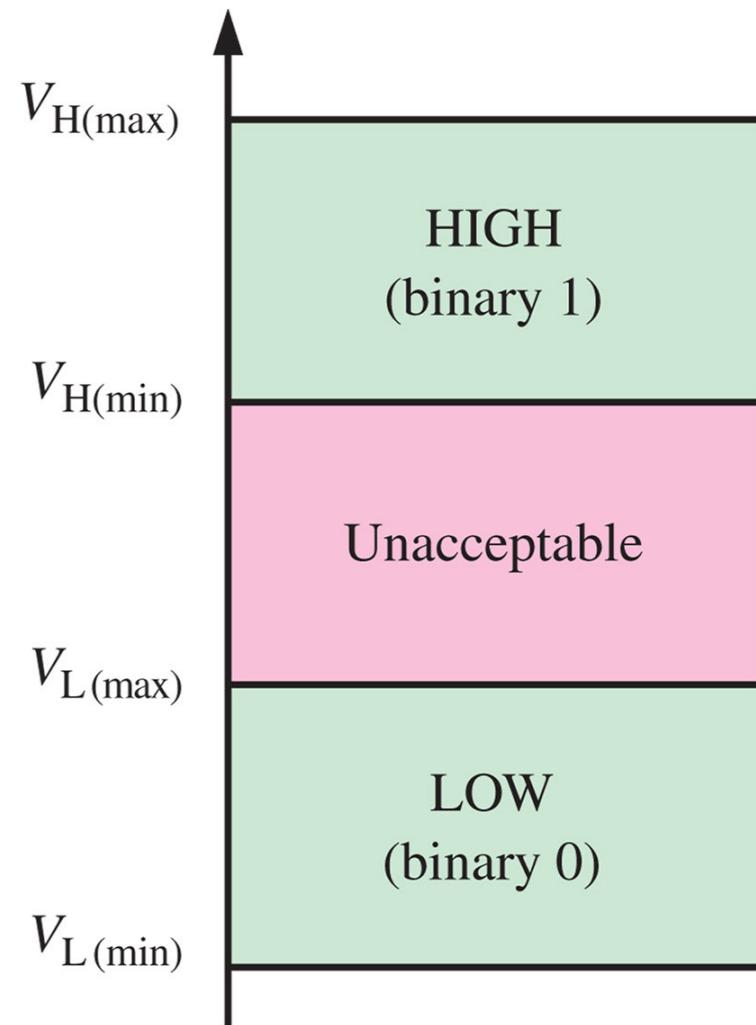


FIGURE 1-6 Logic level ranges of voltage for a digital circuit. (FLOYD)

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

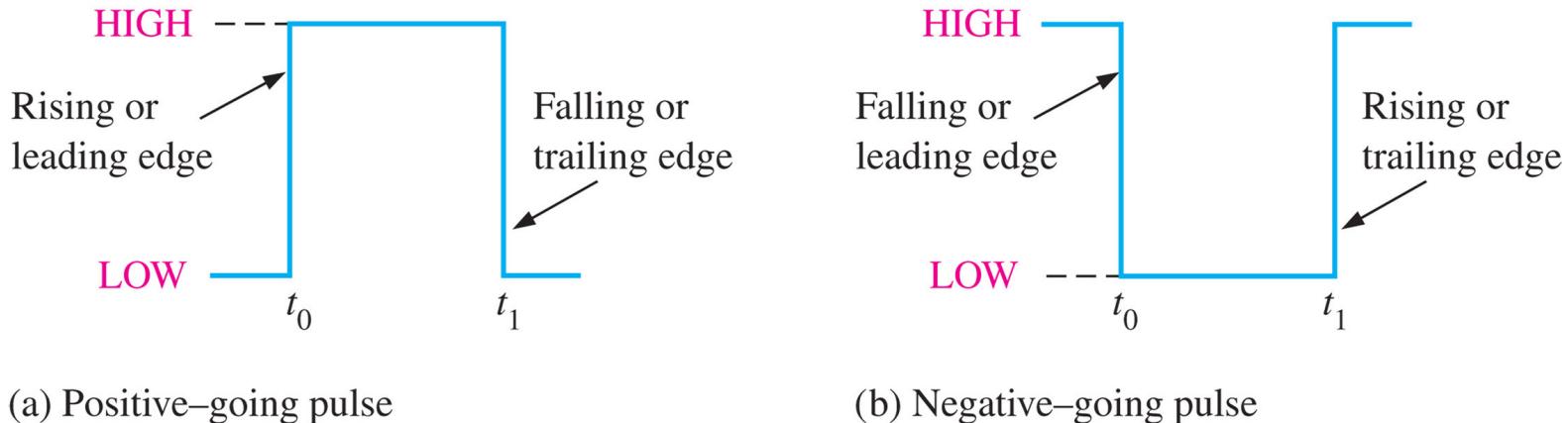
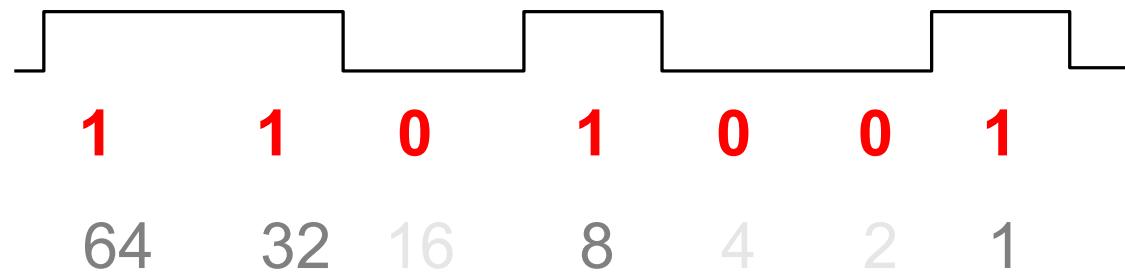


FIGURE 1-7 Ideal pulses.



(105)₁₀ ou (69)₁₆

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

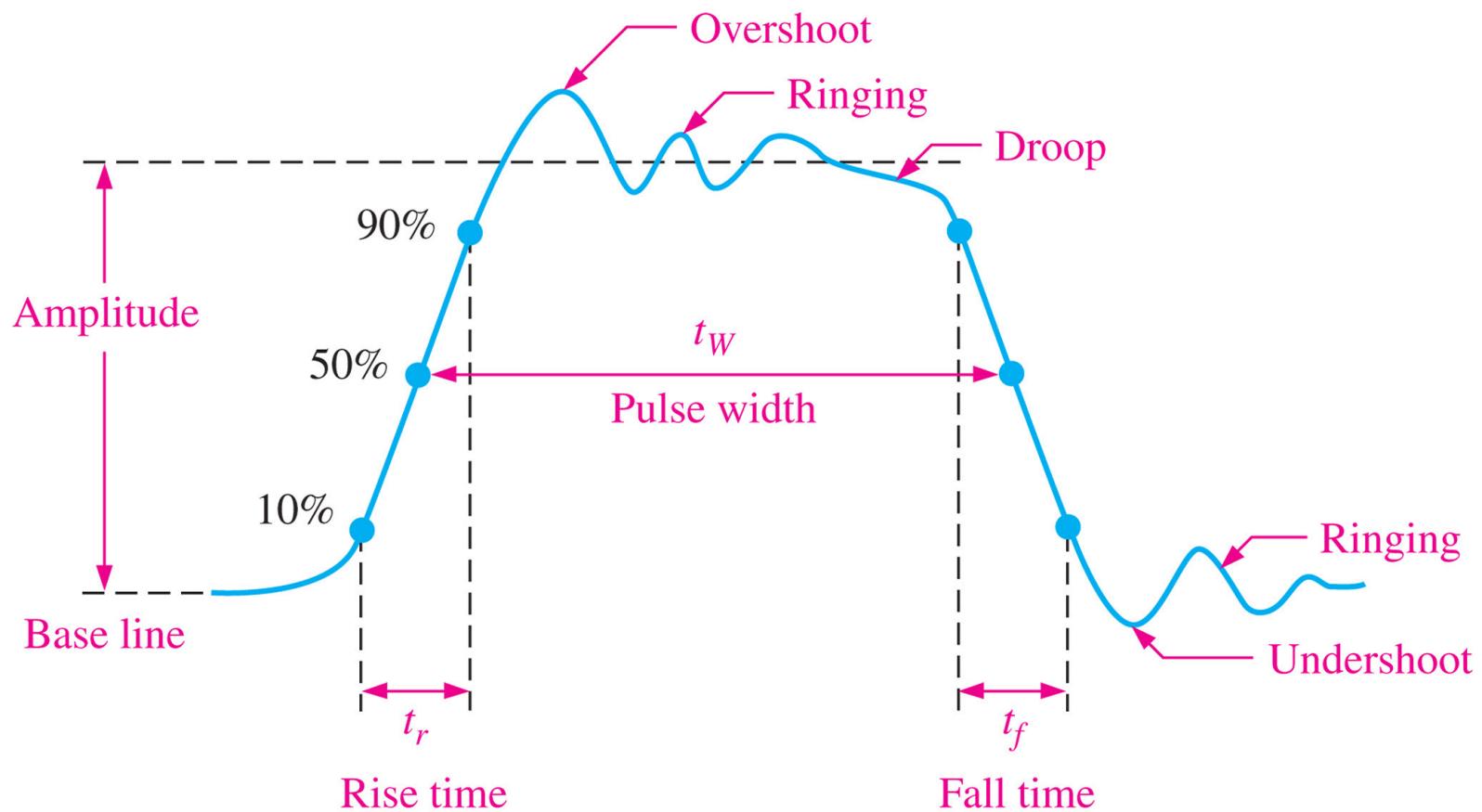
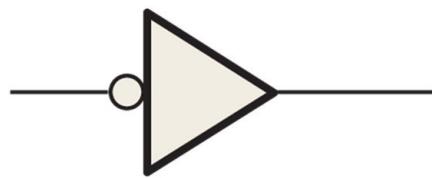
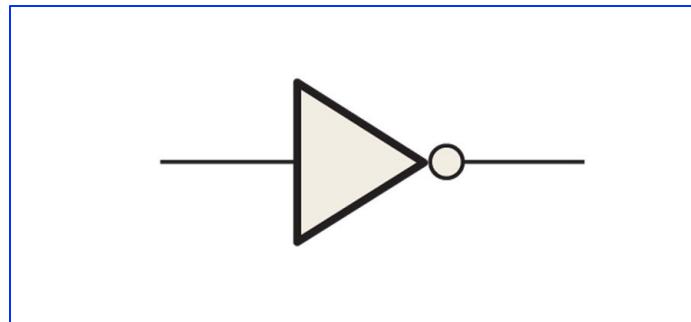
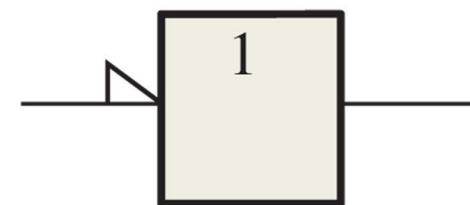
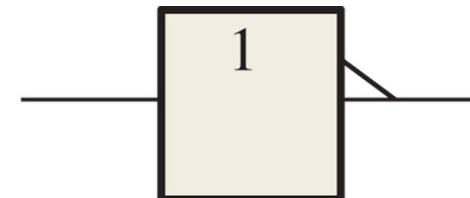


FIGURE 1-8 Nonideal pulse characteristics.

The Inverter



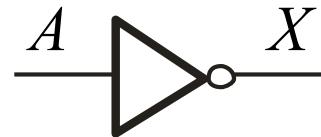
(a) Distinctive shape symbols
with negation indicators



(b) Rectangular outline symbols
with polarity indicators

FIGURE 3-1 Standard logic symbols for the inverter (ANSI/IEEE Std. 91-1984/Std. 91a-1991).

The Inverter

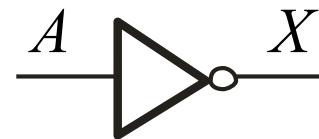


The inverter performs the Boolean **NOT** operation. When the input is LOW, the output is HIGH; when the input is HIGH, the output is LOW

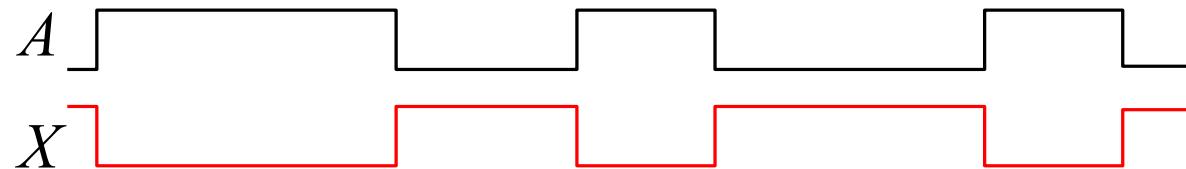
Input	Output
A	X
LOW (0)	HIGH (1)
HIGH (1)	LOW(0)

The **NOT** operation (complement) is shown with an overbar. Thus, the Boolean expression for an inverter is $X = \overline{A}$

The Inverter



Example waveforms:



A group of inverters can be used to form the 1's complement of a binary number:

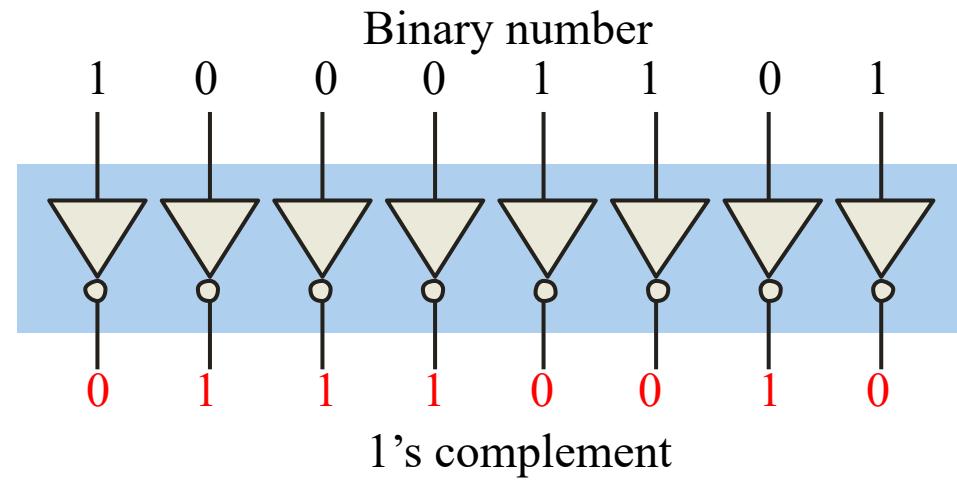
1's + 1 → negative

$$0011 \rightarrow 3$$

$$1100 \rightarrow 1's \text{ compl}$$

$$+ 1$$

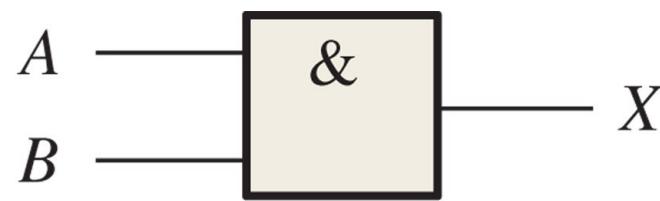
$$\hline 1101 \rightarrow -3 (-8+4+1)$$



The AND gate



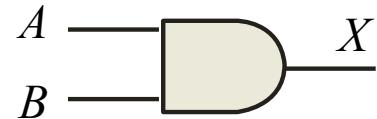
(a) Distinctive shape



(b) Rectangular outline with the
AND ($\&$) qualifying symbol

FIGURE 3-8 Standard logic symbols for the AND gate showing two inputs (ANSI/IEEE Std. 91-1984/Std. 91a-1991).

The AND gate



The **AND gate** produces a HIGH output when all inputs are HIGH; otherwise, the output is LOW. For a 2-input gate, the truth table is

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

The **AND** operation is usually shown with a dot between the variables but it may be implied (no dot). Thus, the AND operation is written as $X = A \cdot B$ or $X = AB$

The AND gate

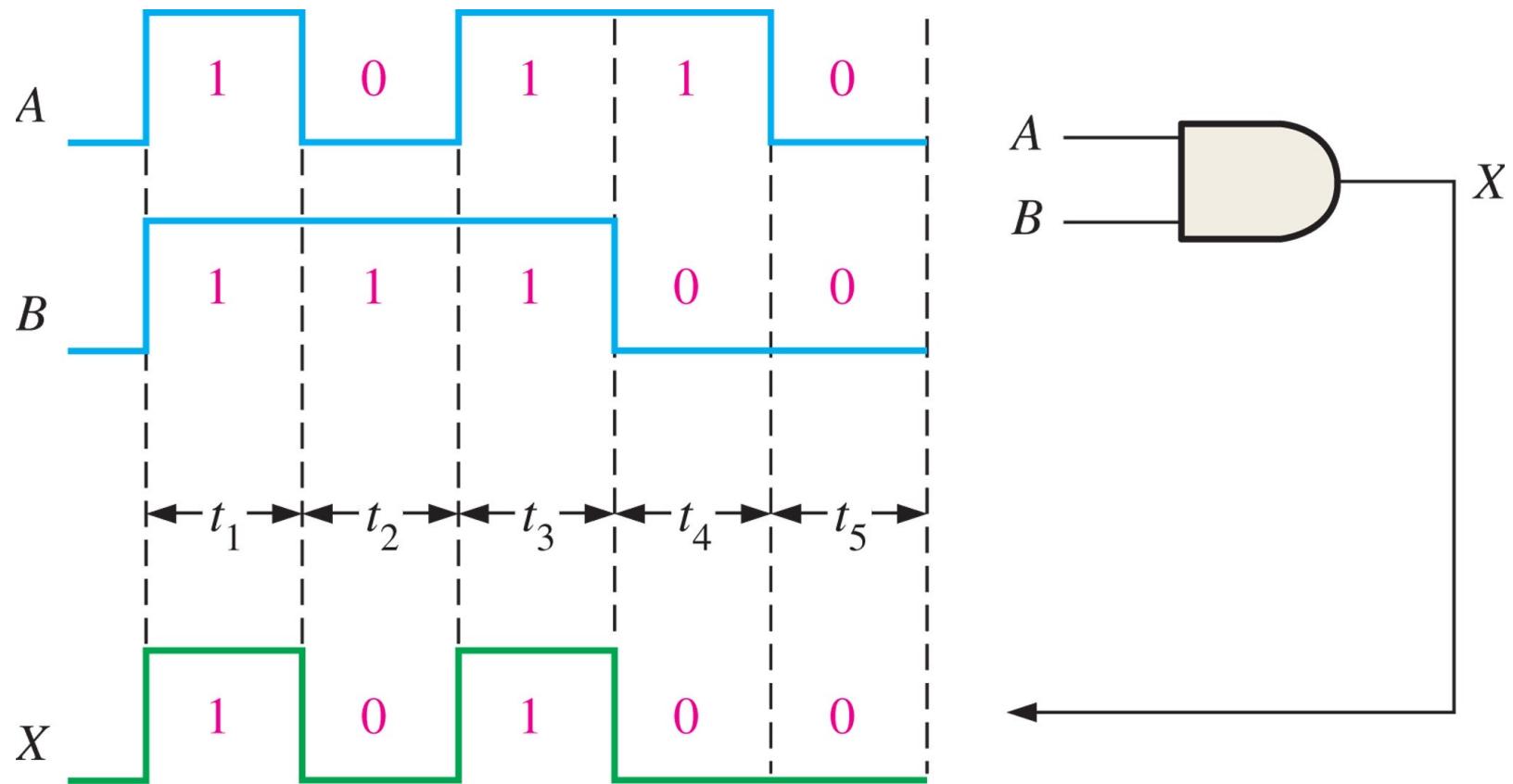
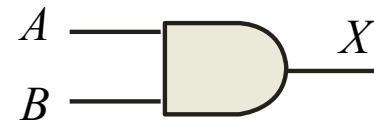
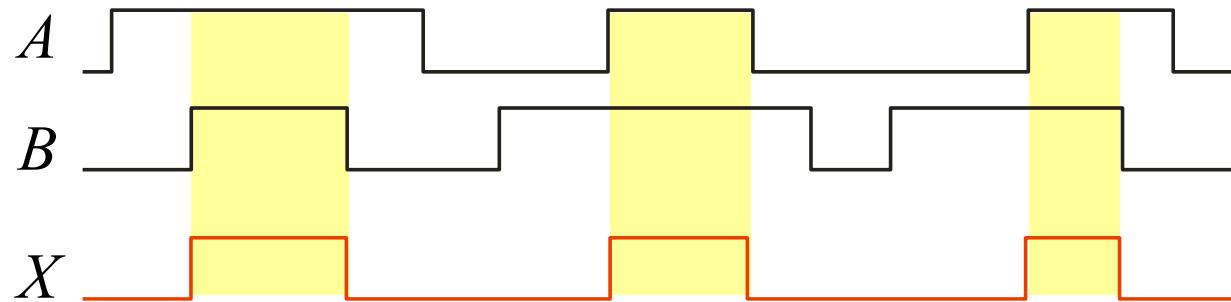


FIGURE 3-10 Example of AND gate operation with a timing diagram showing input and output relationships.

The AND gate



Exemplo de formas de onda



A operação AND pode se usar em programação para **mascaramento** seletivo de bits.

- ‘1’ para manter o valor
- ‘0’ para zerar um determinado bit

Exemplo:

$$\begin{array}{r} 1010\ 0011 \\ \text{AND } 0000\ 1111 \\ \hline 0000\ 0011 \end{array}$$

The AND gate

Waveform for X ?

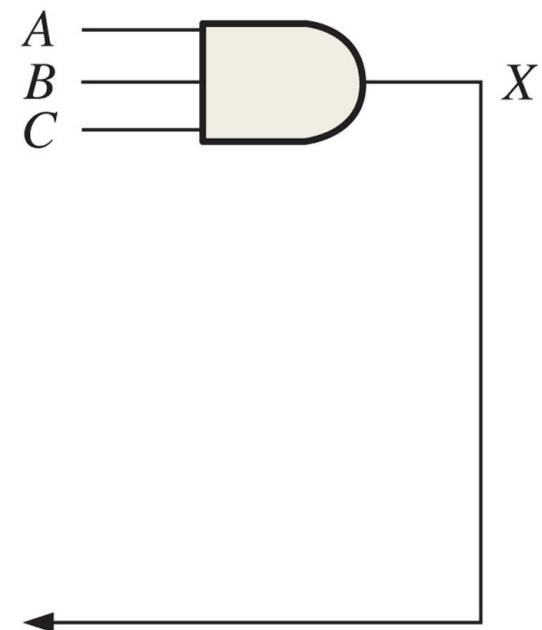
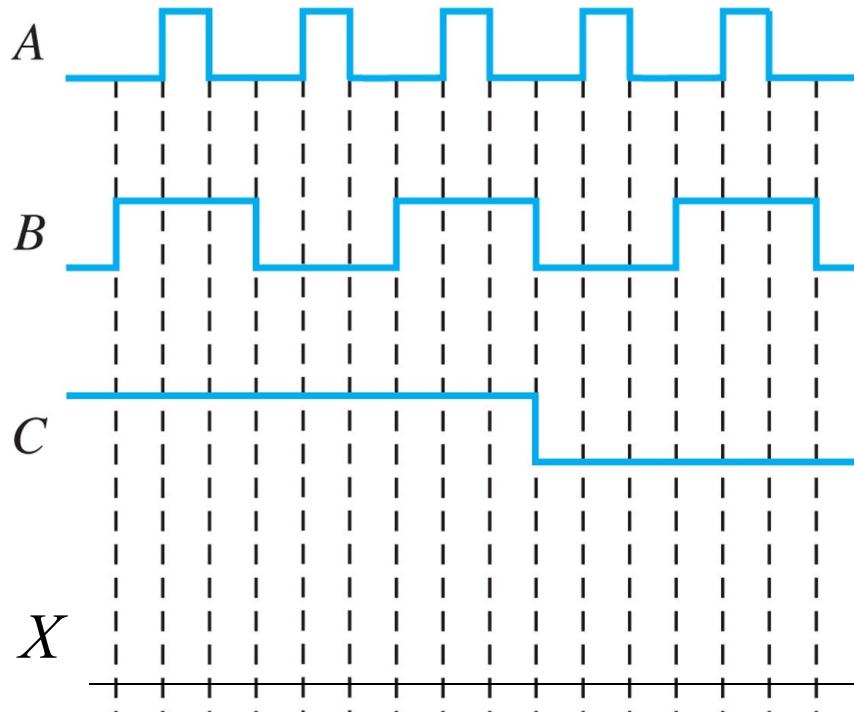
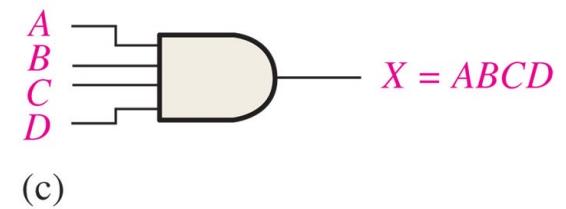
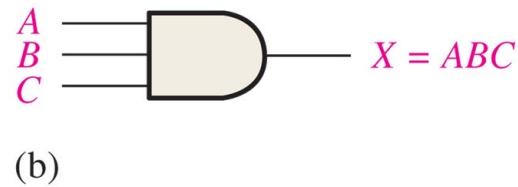
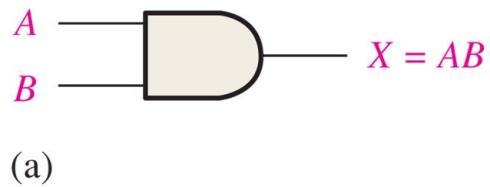


FIGURE 3-13

The AND gate



(a)

(b)

(c)

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

The OR gate

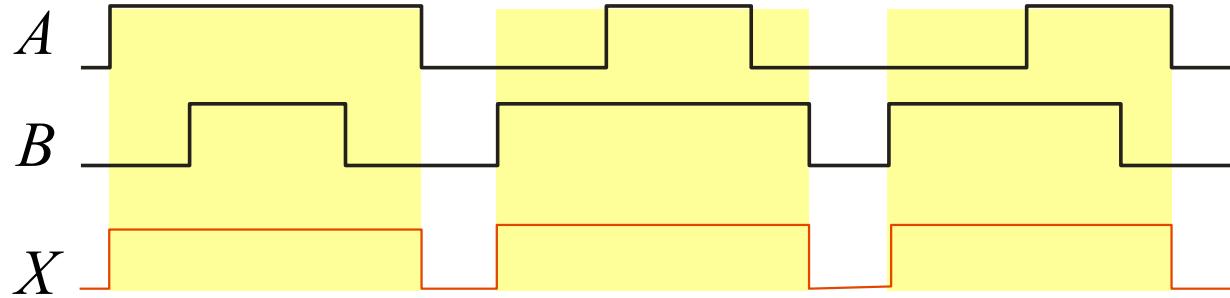
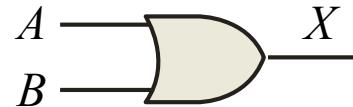


The **OR gate** produces a HIGH output if any input is HIGH; if all inputs are LOW, the output is LOW. For a 2-input gate, the truth table is

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

The **OR** operation is shown with a plus sign (+) between the variables. Thus, the OR operation is written as $X = A + B$.

The OR gate



The OR operation can be used in computer programming to set certain bits of a binary number to 1.

ASCII letters have a **1** in the **bit 5 position** for lower case letters and a **0** in this position for capitals. (Bit positions are numbered from right to left starting with 0.) What will be the result if you OR an ASCII letter with the 8-bit mask 00100000?

The resulting letter will be lower case.

Exemplo de código

```
#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
```

The OR gate

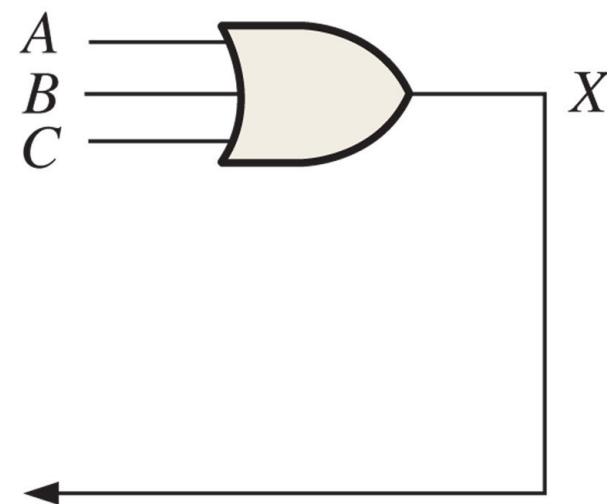
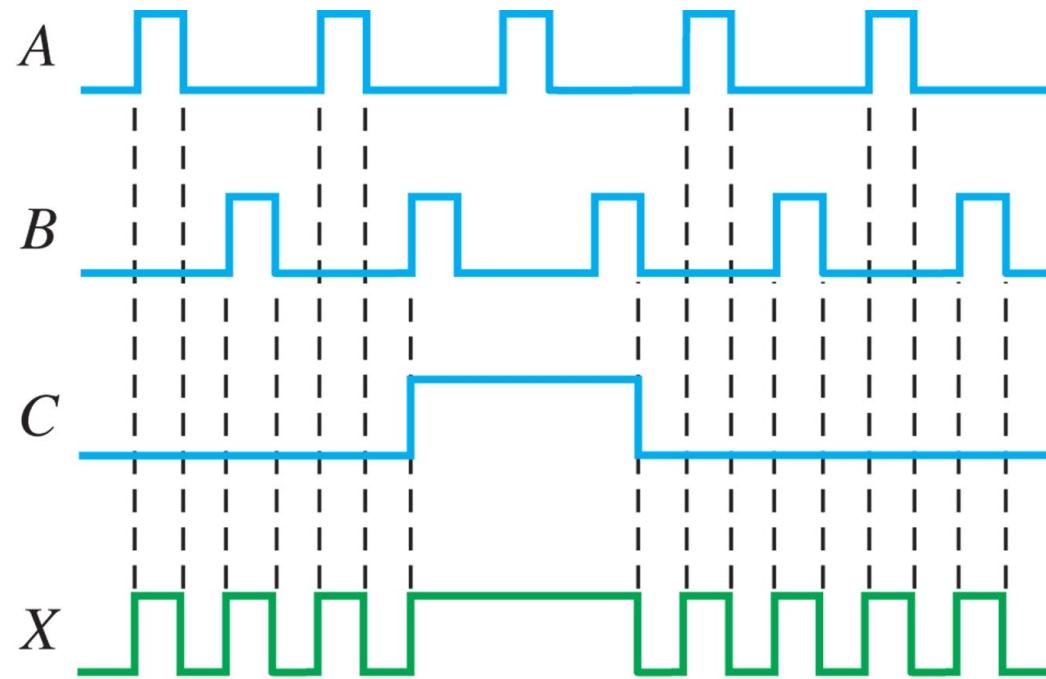


FIGURE 3-23

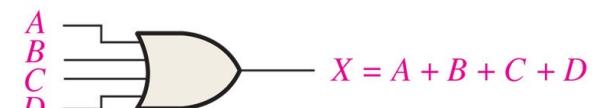
The OR gate



(a)



(b)



(c)

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

The NAND gate

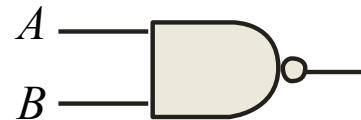


The **NAND gate** produces a LOW output when all inputs are HIGH; otherwise, the output is HIGH. For a 2-input gate, the truth table is

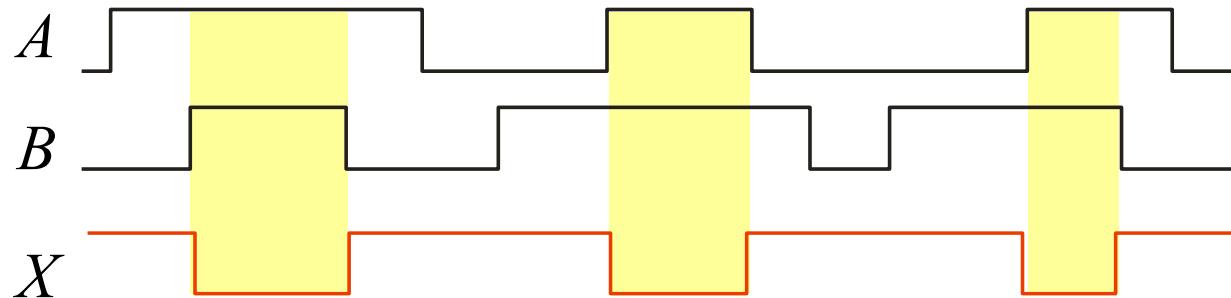
Inputs		Output
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

The **NAND** operation is shown with a dot between the variables and an overbar covering them. Thus, the NAND operation is written as $X = \overline{A \cdot B}$ (alternatively, $X = \overline{AB}$)

The NAND gate

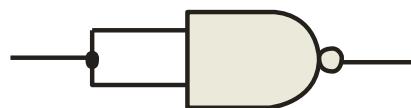


Example waveforms:



The NAND gate is particularly useful because it is a “universal” gate – all other basic gates can be constructed from NAND gates.

A 2-input NAND gate with both inputs connected together is equivalent to which gate ?



The NAND gate

The output waveform X is LOW only when **all three input waveforms are HIGH** as shown in the timing diagram.

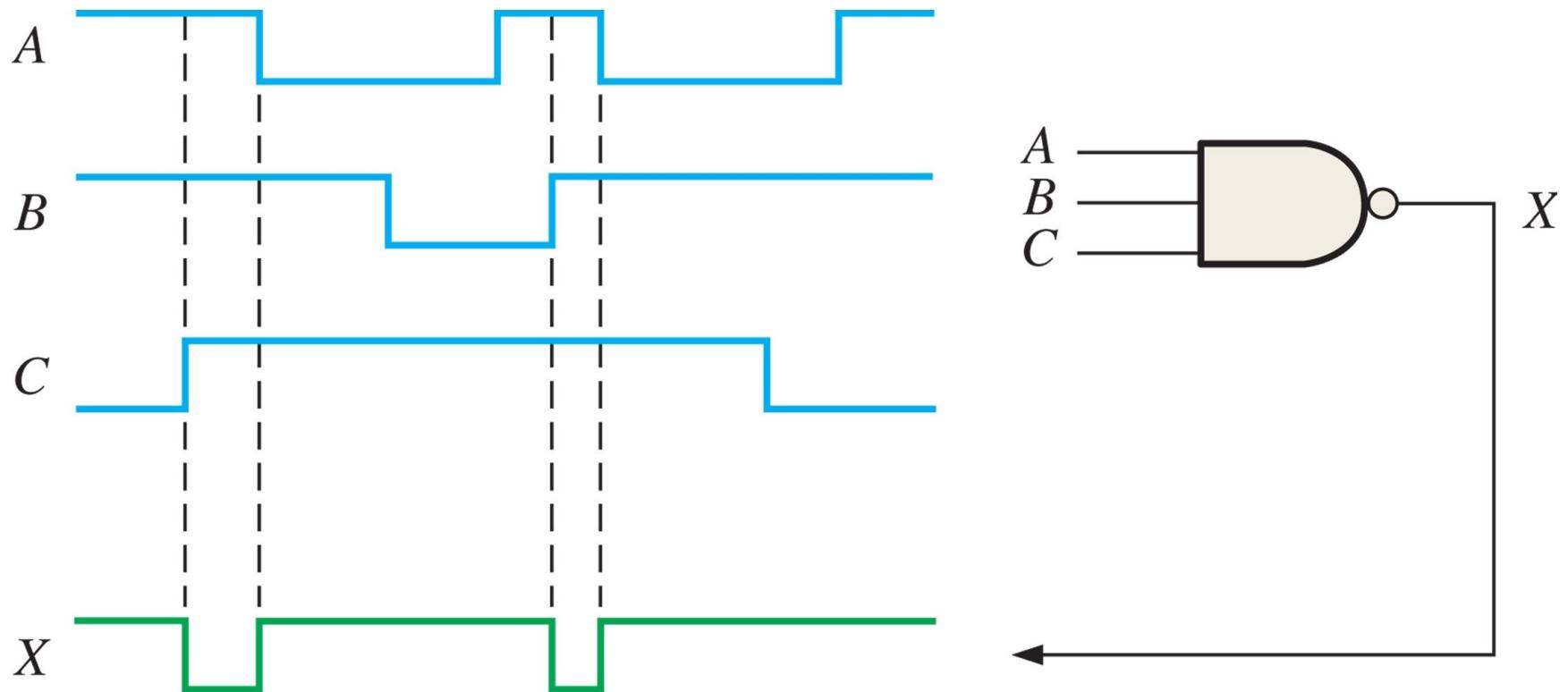
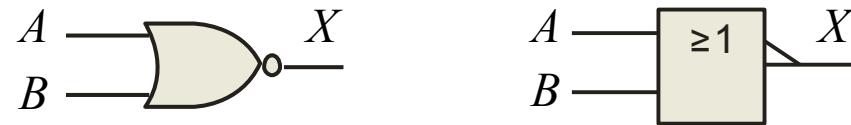


FIGURE 3-29

The NOR gate

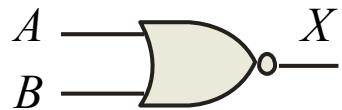


The **NOR gate** produces a LOW output if any input is HIGH; if all inputs are LOW, the output is HIGH. For a 2-input gate, the truth table is

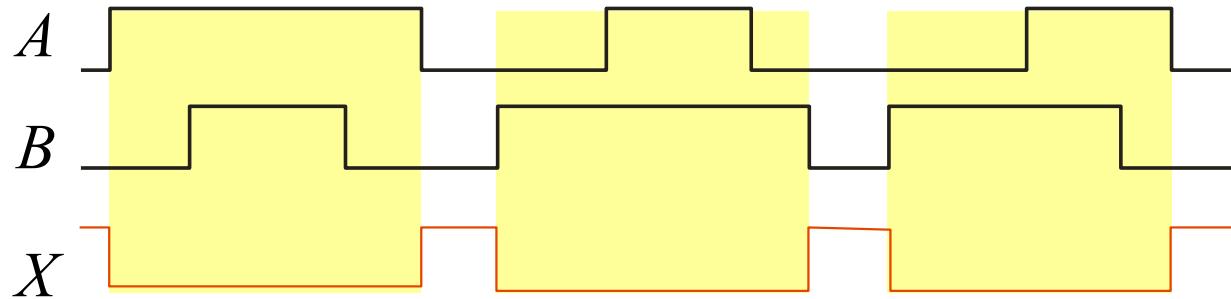
Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

The **NOR** operation is shown with a plus sign (+) between the variables and an overbar covering them. Thus, the NOR operation is written as $X = \overline{A + B}$.

The NOR gate



Example waveforms:



The NOR operation will produce a LOW if any input is HIGH.

The NOR gate

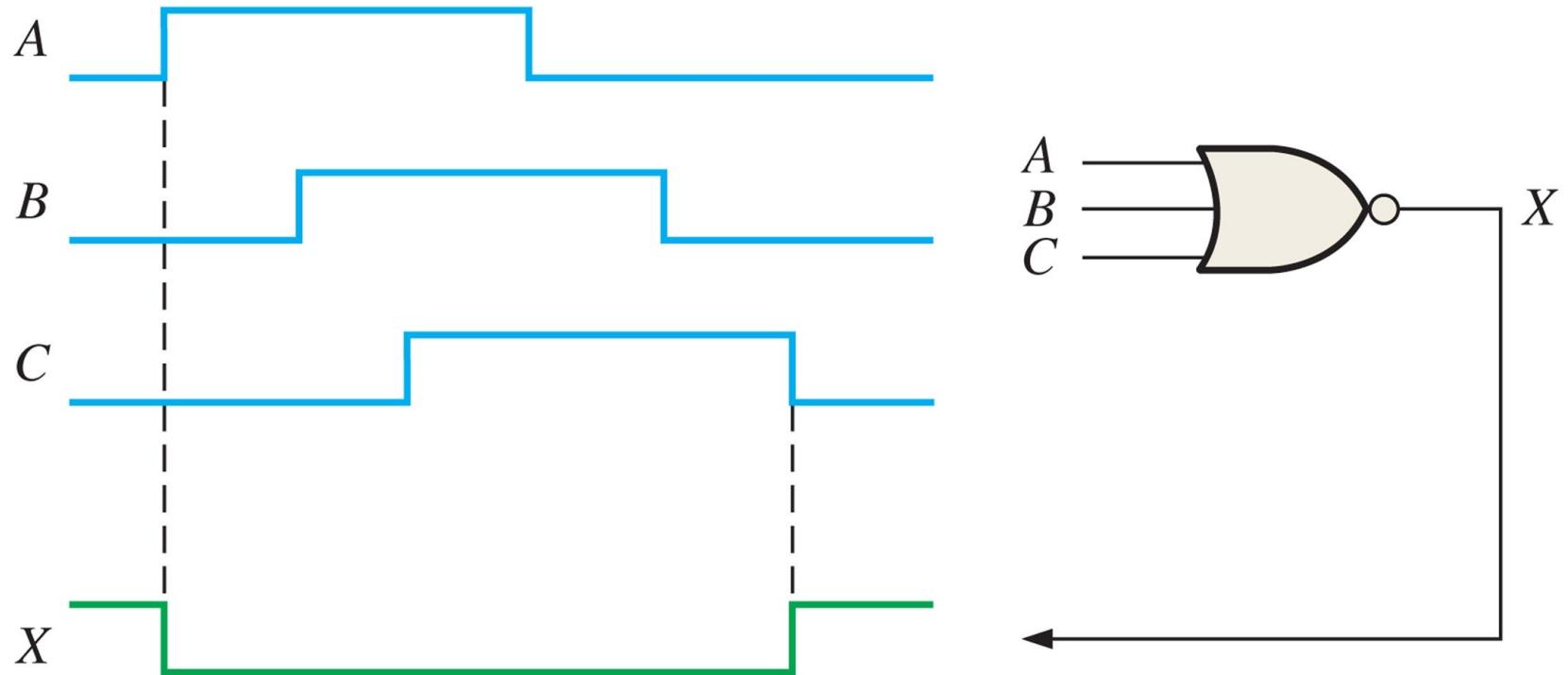
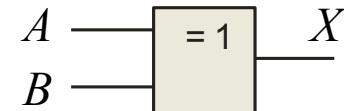
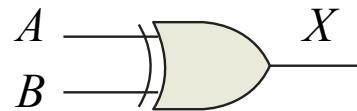


FIGURE 3-37

The XOR gate

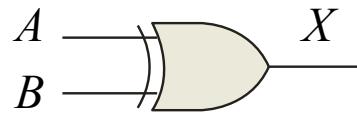


The **XOR gate** produces a HIGH output only when both inputs are at opposite logic levels. The truth table is

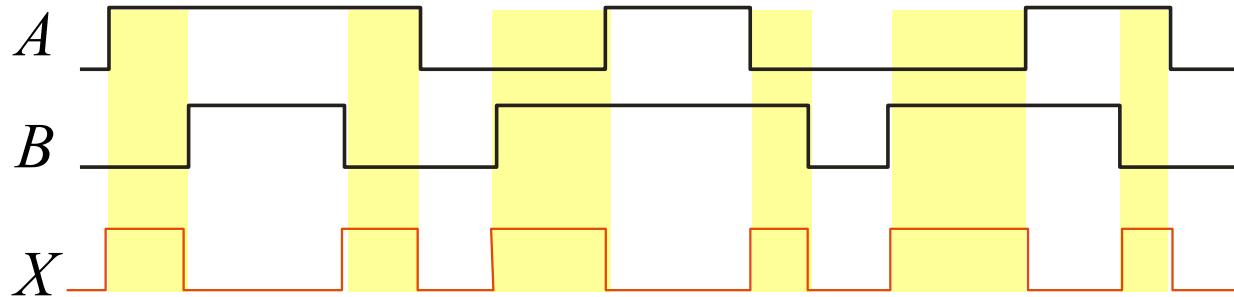
Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

The **XOR** operation is written as $X = \overline{AB} + \overline{A}\overline{B}$. Alternatively, it can be written with a circled plus sign between the variables as $X = A \oplus B$.

The XOR gate



Example waveforms:



Notice that the XOR gate will produce a HIGH only when exactly one input is HIGH.

If the A and B waveforms are both inverted for the above waveforms, how is the output affected?

There is no change in the output.

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 **ímpar**
- Corresponde à **soma** dos bits

The XNOR Gate



The **XNOR gate** produces a HIGH output only when both inputs are at the same logic level. The truth table is

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

The **XNOR** operation shown as $X = \overline{A}\overline{B} + AB$. Alternatively, the XNOR operation can be shown with a circled dot between the variables. Thus, it can be shown as $X = A \odot B$.

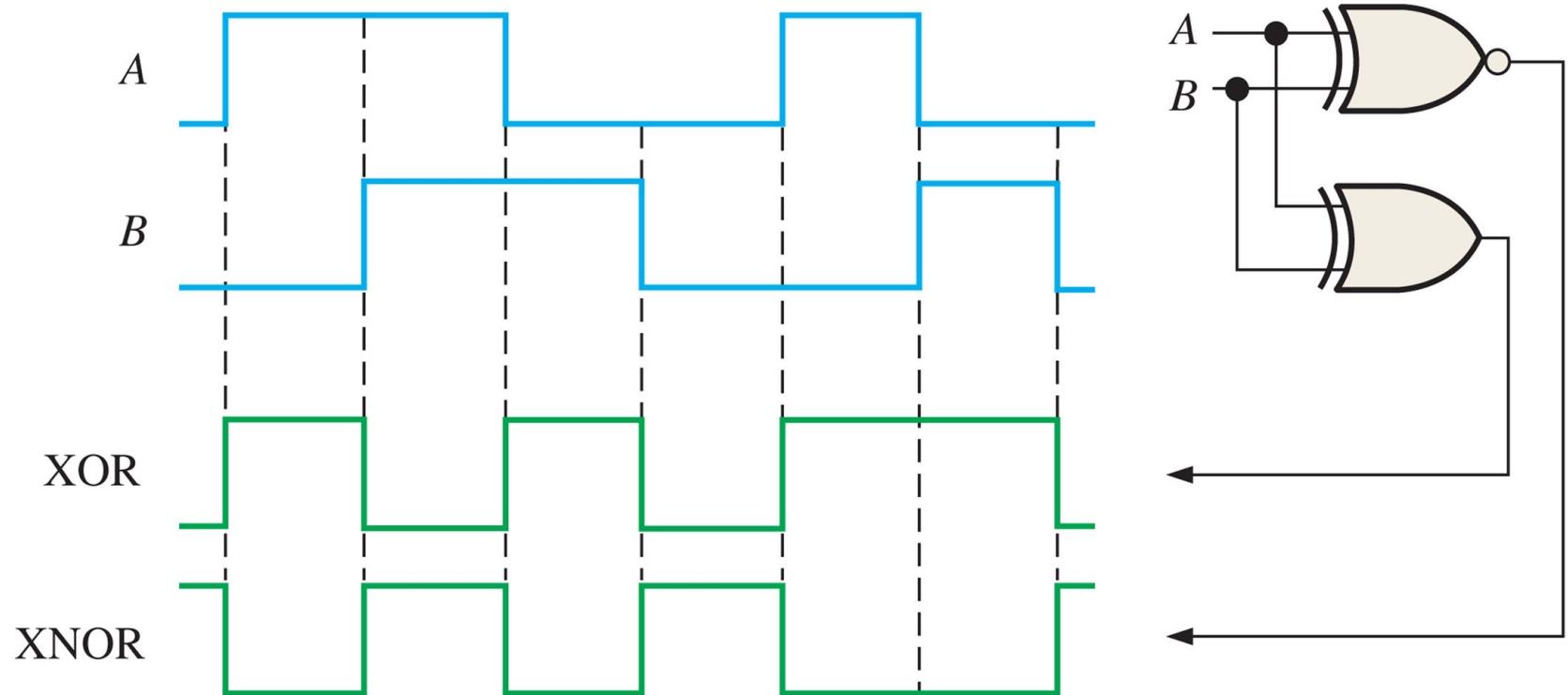
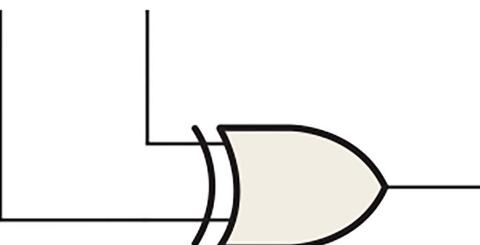


FIGURE 3-48

TABLE 3-13

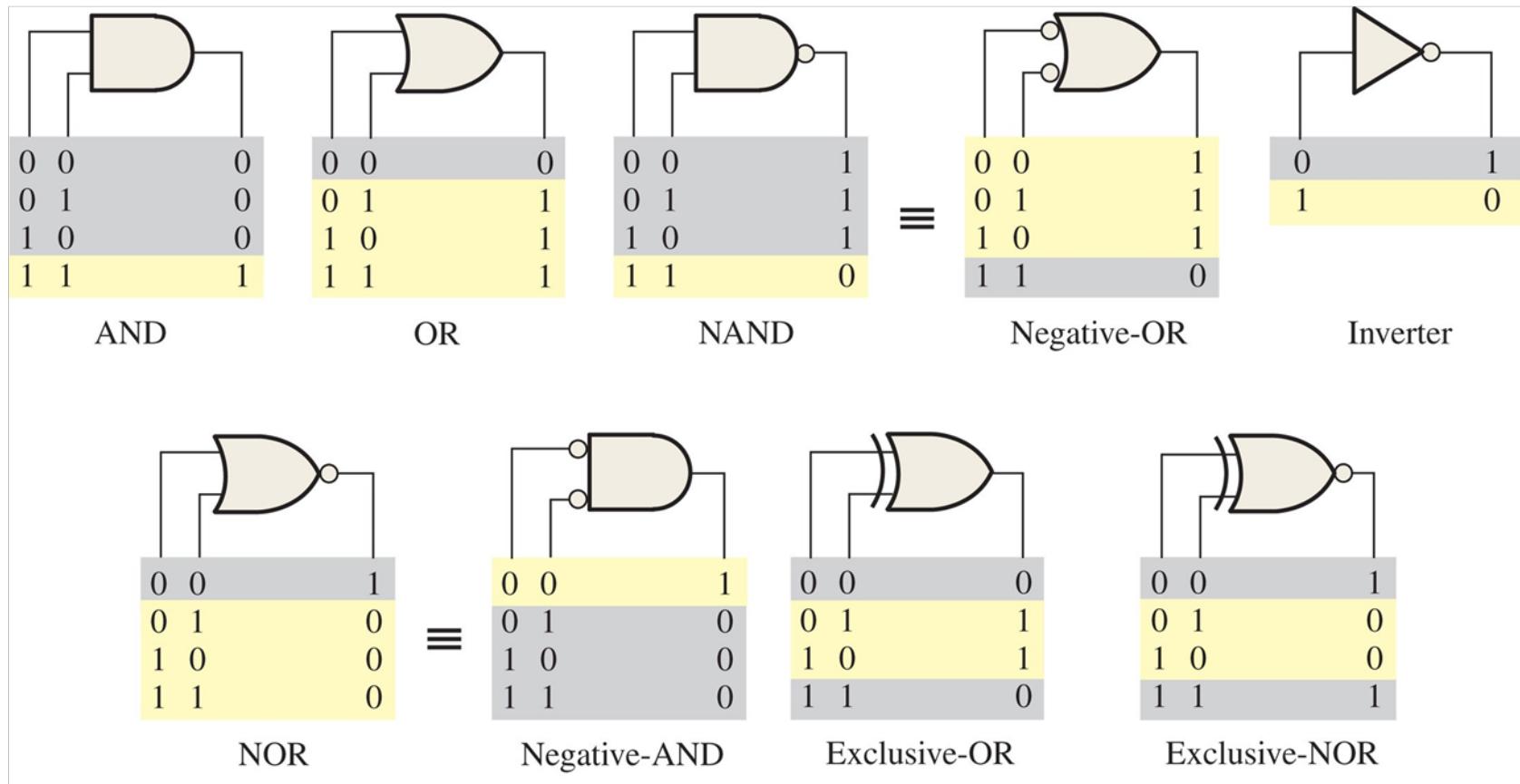
An XOR gate used to add two bits.

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

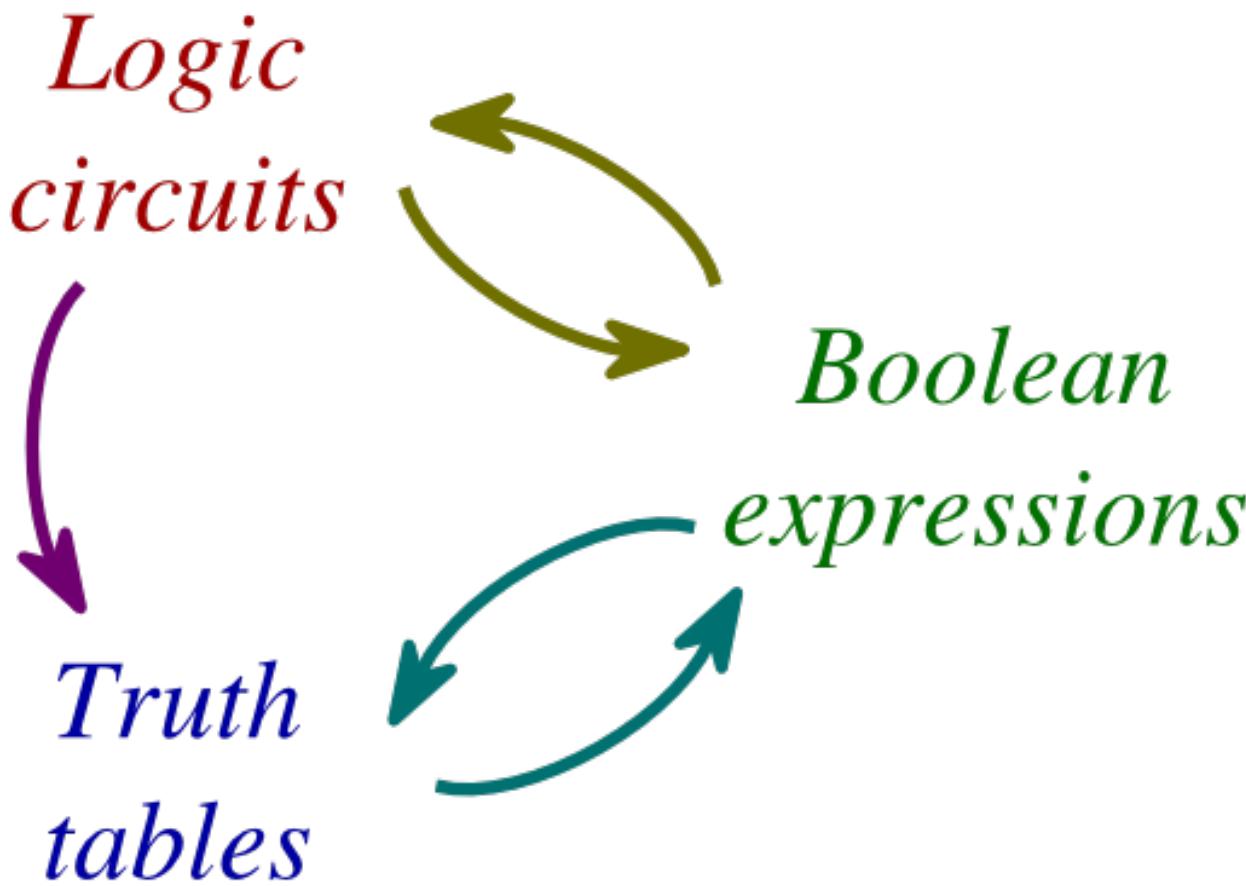


Resumo das Portas Lógicas

FIGURE 3-75

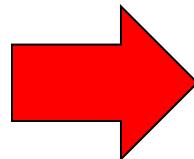


Transformações entre Representações Lógicas



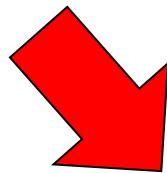
Transformações entre Representações Lógicas

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



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

Logic circuits *Boolean expressions*
 ↘ ↗
Truth tables *Boolean expressions*



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

Somatório de Mintermos

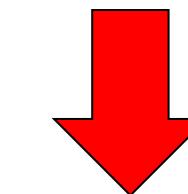
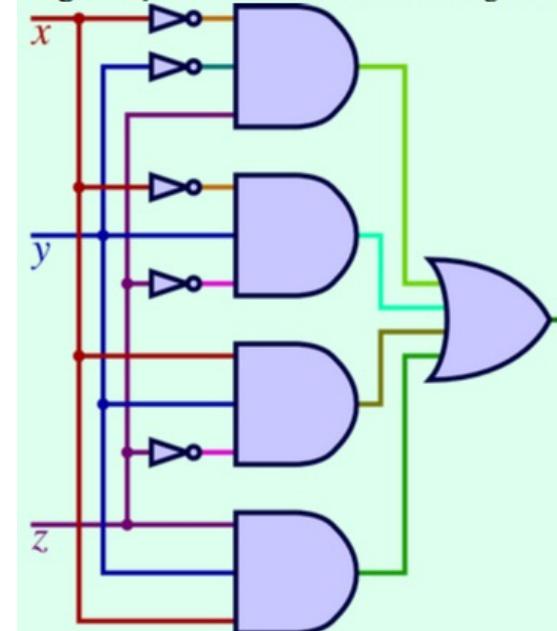
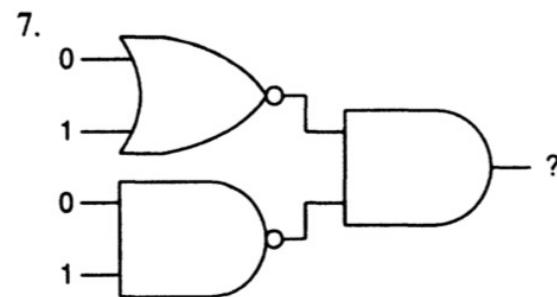
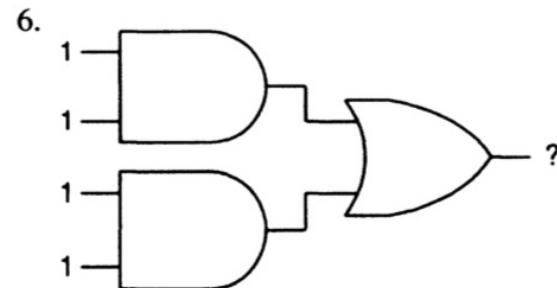
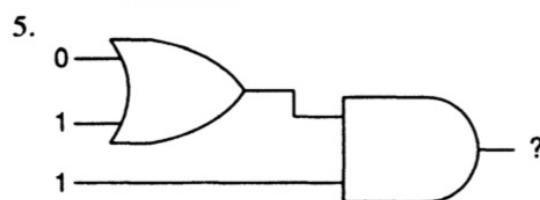
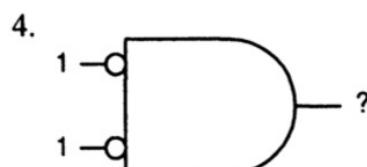
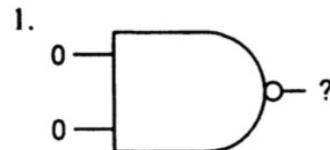


Figure 4: A circuit derived from a given truth table.



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

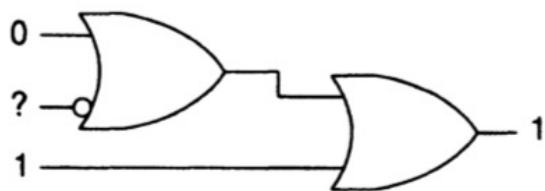


B. If possible, determine the value of the missing input bit in each of the following circuit diagrams:

9.



16.



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



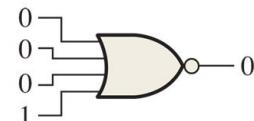
(a)



(b)



(c)



(d)

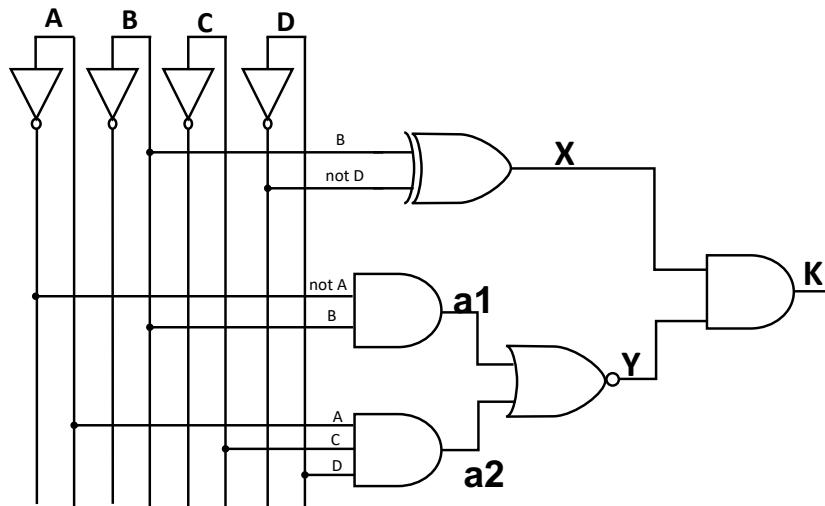


(e)



(f)

Escrever a equação algébrica do circuito abaixo e preencher tabela verdade



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

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

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

A	B	C	D	a1	a2	Y	X	K
0	0	0	0			1	1	1
0	0	0	1			1		
0	0	1	0			1	1	1
0	0	1	1			1		
0	1	0	0	1				
0	1	0	1	1			1	
0	1	1	0	1		1		
0	1	1	1	1			1	
1	0	0	0			1	1	1
1	0	0	1			1		
1	0	1	0			1	1	1
1	0	1	1	1				
1	1	0	0			1		
1	1	0	1			1	1	1
1	1	1	0			1		
1	1	1	1	1		1		

Simplificação Lógica

- Lógica simplificada == hardware mais eficiente
 - Menor área de hardware
 - Menor o atraso da lógica
- Próximo conteúdo: simplificação lógica