

# Sistemas Digitais

ET46B

---

Prof. Eduardo Vinicius Kuhn

*[kuhn@utfpr.edu.br](mailto:kuhn@utfpr.edu.br)*

Curso de Engenharia Eletrônica

Universidade Tecnológica Federal do Paraná



# Capítulo 4

## Circuitos lógicos combinacionais

# Conteúdo

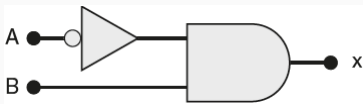
- 4.1 Forma de soma-de-produtos
- 4.2 Simplificação de circuitos lógicos
- 4.3 Simplificação algébrica
- 4.4 Projetando circuitos lógicos combinacionais
- 4.5 Método do Mapa de Karnaugh
- 4.6 Circuitos *exclusive*-OR (XOR) e *exclusive*-NOR (XNOR)
- 4.8 Circuitos de habilitação/desabilitação

# Objetivos

- Implementar circuitos lógicos simples **sem o auxílio da tabela-verdade**.
- Determinar expressões lógicas nas formas (canônicas) de **soma-de-produtos** ou produto-de-somas.
- Implementar circuitos lógicos **a partir de expressões na forma de soma-de-produtos** ou produto-de-somas.
- Usar a **álgebra booleana ou o método do Mapa de Karnaugh** na simplificação de expressões lógicas.
- Introduzir os **circuitos exclusive-OR** (i.e., XOR) e **exclusive-NOR** (i.e., XNOR).
- Descrever o funcionamento de **circuitos de habilitação**.

# Introdução

- Como **circuitos combinacionais não possuem memória**, suas saídas dependem apenas das entradas atuais.
- **O nível lógico da saída pode ser determinado**, em qualquer instante, **em função** dos níveis lógicos das entradas.
- Uma **forma conveniente** de representar a saída para todas as condições de entrada se dá por meio de uma **tabela-verdade**.



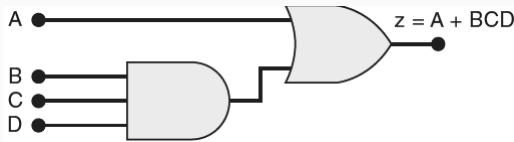
A	B	x
0	0	0
0	1	1
1	0	0
1	1	0

**Exemplo:** Um conversor A/D está monitorando a tensão CC de uma bateria de 12 V. A saída do conversor é um número binário de 4 bits (i.e.,  $ABCD$ ), que corresponde à tensão da bateria em degraus de 1 V, sendo a variável  $A$  o MSB. As saídas do conversor servem como entradas de um circuito que gera uma saída em nível lógico 1, caso a tensão seja maior que  $0110_2 = 6_{10}$  V. Diante disso, projete o circuito lógico.

# Introdução

**Exemplo:** Um conversor A/D está monitorando a tensão CC de uma bateria de 12 V. A saída do conversor é um número binário de 4 bits (i.e.,  $ABCD$ ), que corresponde à tensão da bateria em degraus de 1 V, sendo a variável  $A$  o MSB. As saídas do conversor servem como entradas de um circuito que gera uma saída em nível lógico 1, caso a tensão seja maior que  $0110_2 = 6_{10}$  V. Diante disso, projete o circuito lógico.

**R:**



E, como devemos proceder caso o problema seja mais complexo (tenha mais requisitos)?



# Procedimento para projeto de circuitos combinacionais

- 1) Interprete o problema e construa uma tabela-verdade que descreva o comportamento desejado.
- 2) Determine
  - cada termo AND (produto) em que a saída seja 1; **ou**
  - cada termo OR (soma) em que a saída seja 0.
- 3) Escreva a expressão para a saída na forma de
  - soma-de-produtos; **ou**
  - produto-de-somas.
- 4) **Simplifique a expressão de saída (quando possível).**
- 5) Implemente o circuito para a expressão final (simplificada).

Dessa forma, torna-se possível projetar circuitos combinacionais que satisfarão um determinado conjunto de requisitos.

# Formas canônicas de expressões booleanas

Existem duas **formas canônicas** para representar expressões booleanas, a saber:

- **soma-de-produtos** (soma de minitermos):

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

- **produto-de-somas** (produto de maxitermos):

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

- Todas as variáveis devem estar presentes em cada termo.
- Operações de “inversão” não cobrem mais do que uma variável.

# Formas canônicas de expressões booleanas

- Forma de **soma-de-produtos** (soma de minitermos):
  - Cada **minitermo** resulta em **1** para apenas uma combinação de valores das entradas.

<i>A</i>	<i>B</i>	<i>C</i>	<i>x</i>	<b>minitermo</b>
0	0	0	1	$\rightarrow \overline{A} \cdot \overline{B} \cdot \overline{C}$
0	0	1	0	-
0	1	0	1	$\rightarrow \overline{A} \cdot B \cdot \overline{C}$
0	1	1	1	$\rightarrow \overline{A} \cdot B \cdot C$
1	0	0	0	-
1	0	1	0	-
1	1	0	0	-
1	1	1	1	$\rightarrow A \cdot B \cdot C$

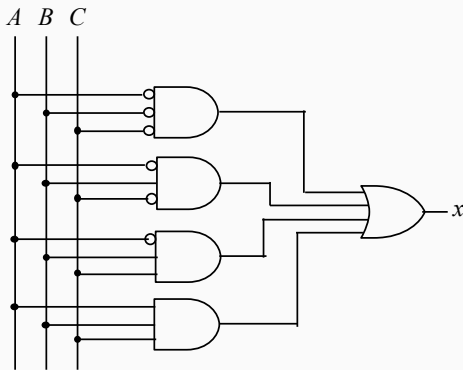
Logo, a expressão de saída é obtida **“somando”** minitermos,

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

# Formas canônicas de expressões booleanas

- Forma de **soma-de-produtos** (soma de minitermos):

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



Usar formas canônicas assegura um atraso de propagação similar para todos os sinais, facilitando a determinação da velocidade operacional máxima do sistema.

# Formas canônicas de expressões booleanas

- Forma de **produto-de-somas** (produto de maxitermos):
  - Cada **maxitermo** resulta em **0** para apenas uma combinação de valores das entradas.

<i>A</i>	<i>B</i>	<i>C</i>	<i>x</i>	maxitermo
0	0	0	1	-
0	0	1	0	$\rightarrow A + B + \overline{C}$
0	1	0	1	-
0	1	1	1	-
1	0	0	0	$\rightarrow \overline{A} + B + C$
1	0	1	0	$\rightarrow \overline{A} + B + \overline{C}$
1	1	0	0	$\rightarrow \overline{A} + \overline{B} + C$
1	1	1	1	-

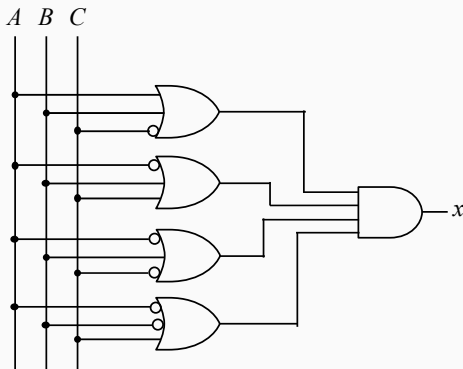
Logo, a expressão de saída é obtida **“multiplicando”** maxitermos,

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

# Formas canônicas de expressões booleanas

- Forma de **produto-de-somas** (produto de maxitermos):

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



Usar formas canônicas assegura um atraso de propagação similar para todos os sinais, facilitando a determinação da velocidade operacional máxima do sistema.

# Formas canônicas de expressões booleanas

**Exemplo:** Determine as expressões booleanas na forma de

- a) soma-de-produtos; e
- b) produto-de-somas

que caracterizam o circuito lógico com a seguinte tabela-verdade:

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



# Formas canônicas de expressões booleanas

**Exemplo:** Determine as expressões booleanas na forma de

- a) soma-de-produtos; e
- b) produto-de-somas

que caracterizam o circuito lógico com a seguinte tabela-verdade:

<i>A</i>	<i>B</i>	<i>C</i>	<i>x</i>
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

**R:**

a)

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

b)

$$\begin{aligned} x = & (A + B + C) \cdot (A + B + \overline{C}) \cdot \\ & (\overline{A} + B + C) \cdot (\overline{A} + B + \overline{C}) \cdot \\ & (\overline{A} + \overline{B} + C) \end{aligned}$$

# Simplificação de expressões/circuitos lógicos

- Busca-se obter uma expressão lógica mais simples, contendo assim o menor número de termos/variáveis.
- Essa expressão simplificada se traduz então em um **circuito equivalente** (mesma lógica) com menos portas e conexões.
- Um **circuito mais simples**, com menos portas e conexões, é
  - **mais barato** (menos elementos);
  - **mais confiável** (por eliminar causas potenciais de defeitos); e
  - **mais veloz** (por reduzir o atraso de propagação).
- Nesse processo de simplificação, pode-se considerar
  - o uso dos **teoremas da álgebra booleana**; ou
  - uma técnica de mapeamento introduzida por **Karnaugh**.

# Simplificação de expressões/circuitos lógicos

Independentemente da forma canônica adotada, i.e.,

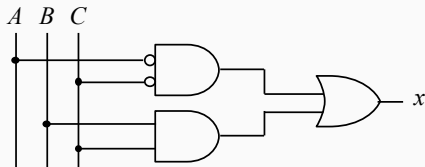
- **soma-de-produtos** (soma de minitermos):

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

- **produto-de-somas** (produto de maxitermos):

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

**simplificações podem ser realizadas**, e.g.,



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

# Simplificação de expressões/circuitos lógicos

**Exemplo:** Projete um circuito lógico com 3 entradas ( $A$ ,  $B$  e  $C$ ), cuja saída  $x$  assuma nível lógico 1 quando a maioria das entradas estiver em nível lógico 1.

Atenção à quantidade de '1s' e '0s' na tabela-verdade antes de escolher a forma canônica a ser utilizada/adotada.

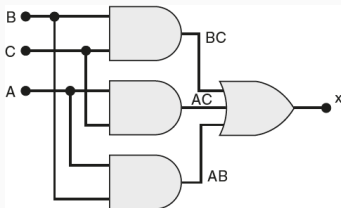
# Simplificação de expressões/circuitos lógicos

**Exemplo:** Projete um circuito lógico com 3 entradas ( $A$ ,  $B$  e  $C$ ), cuja saída  $x$  assuma nível lógico 1 quando a maioria das entradas estiver em nível lógico 1.

**R:**

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

$$\begin{aligned}x &= \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC \\ &= AB + AC + BC\end{aligned}$$



Atenção à quantidade de '1s' e '0s' na tabela-verdade antes de escolher a forma canônica a ser utilizada/adotada.

# Simplificação de expressões/circuitos lógicos

Geralmente, **simplificações algébricas** são um processo de **tentativa e erro**, os quais envolvem:

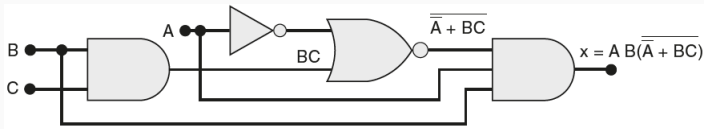
- Colocar a expressão original na **forma de soma-de-produtos**.
- Realizar a **fatoração dos termos produto** que têm fatores comuns.
- “Com sorte,” a fatoração resultará na eliminação de um ou mais termos.

*Infelizmente, nem sempre é óbvio quais teoremas e/ou em que ordem eles devem ser usados.*

Os métodos de simplificação usados aqui são baseados na forma de soma-de-produtos; todavia, a forma de produto-de-somas aparece em alguns casos particulares.

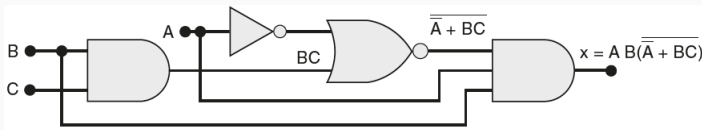
# Simplificação de expressões/circuitos lógicos

**Exemplo:** Realize a simplificação do circuito lógico ilustrado a seguir e ilustre o diagrama esquemático da implementação.



# Simplificação de expressões/circuitos lógicos

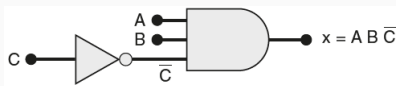
**Exemplo:** Realize a simplificação do circuito lógico ilustrado a seguir e ilustre o diagrama esquemático da implementação.



**R:** A partir dos Teoremas booleanos, é possível mostrar que

$$x = ABC\overline{C}$$

o que resulta em





# Simplificação algébrica

**Exemplo:** Simplifique, usando álgebra booleana, as seguintes expressões lógicas:

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

b)  $z = (\overline{A} + B)(A + B + D) \cdot \overline{D}$

c)  $y = \overline{A} \cdot C \cdot \overline{(\overline{A} \cdot B \cdot D)} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot C$

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

# Simplificação algébrica

**Exemplo:** Simplifique, usando álgebra booleana, as seguintes expressões lógicas:

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

**R:**  $z = A(\overline{B} + C)$

b)  $z = (\overline{A} + B)(A + B + D) \cdot \overline{D}$

**R:**  $z = B \cdot \overline{D}$

c)  $y = \overline{A} \cdot C \cdot \overline{(\overline{A} \cdot B \cdot D)} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot C$

**R:**  $y = \overline{B} \cdot C + \overline{A} \cdot \overline{D} \cdot (B + C) \rightarrow y = \overline{A} \cdot B \cdot \overline{D} + \overline{B} \cdot C$  (?)

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

**R:**  $w = \overline{A} \cdot \overline{B} + A \cdot B$  (\*)

(\*)O processo de simplificação produz um circuito equivalente;  
todavia, nem sempre é mais simples.

Não é fácil dizer se uma expressão está em sua forma mais simples ou se ainda pode ser simplificada.

# Método do Mapa de Karnaugh

- O Mapa de Karnaugh, tal como a tabela-verdade, **é um meio de mostrar a relação entre as entradas e a(s) saída(s).**
- O Mapa de Karnaugh é um método gráfico **usado para obter uma expressão lógica já simplificada.**
- O método pode ser usado para obter expressões
  - tanto na forma de soma-de-produtos (agrupando '1s');
  - quanto produto-de-somas (agrupando '0s').
- Embora possa ser usado em problemas com mais entradas, **sua utilidade prática está limitada a 5 ou 6 variáveis.**
- A discussão aqui se restringe a problemas com até 4 entradas; em casos com mais entradas, outros algoritmos são utilizados.

# Formato do Mapa de Karnaugh

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

$A \backslash B$	0	1
0	1	0
1	0	1

- O Mapa de Karnaugh contém as mesmas informações da tabela-verdade.
- Cada linha na tabela-verdade corresponde a um “quadrado” no Mapa de Karnaugh.
- Portanto, é apenas uma técnica de mapeamento que evidencia as possíveis simplificações.

# Formato do Mapa de Karnaugh

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

$A \backslash B$	0	1
0	1	0
1	0	1

- Com o **Mapa de Karnaugh preenchido**, expressões para a saída  $x$  podem ser obtidas na forma de
  - soma-de-produtos** (quadrados que contêm 1s):

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

- produto-de-somas** (quadrados que contêm 0s):

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

# Mapas de Karnaugh com mais variáveis

A \ B	0	1
0		
1		

A \ BC	00	01	11	10
0				
1				

AB \ C	0	1
00		
01		
11		
10		

AB \ CD	00	01	11	10
00				
01				
11				
10				

- **Quadrados adjacentes diferem de apenas uma variável.**
- Numeração ordenada conforme o código Gray.
- Menor distância de Hamming entre duas sequências binárias.

## Mapas de Karnaugh com mais variáveis

**Exemplo:** Construa o Mapa de Karnaugh a partir da tabela-verdade e determine a expressão booleana do circuito lógico.

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

Focando em expressões na forma de soma-de-produtos, é útil representar as variáveis do Mapa de Karnaugh em suas formas negada e não-negada.



# Mapas de Karnaugh com mais variáveis

**Exemplo:** Construa o Mapa de Karnaugh a partir da tabela-verdade e determine a expressão booleana do circuito lógico.

A	B	C	x
0	0	0	1
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	0

**R:** Com respeito ao Mapa de Karnaugh,

	$\overline{B}\overline{C}$	$\overline{B}C$	$BC$	$B\overline{C}$
$\overline{A}$	1	1	0	1
A	0	0	0	1

Portanto, sem realizar agrupamentos,

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

Focando em expressões na forma de soma-de-produtos, é útil representar as variáveis do Mapa de Karnaugh em suas formas negada e não-negada.

# Mapas de Karnaugh com mais variáveis

**Exemplo:** Construa o Mapa de Karnaugh a partir da tabela-verdade e determine a expressão booleana do circuito lógico.

A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

# Mapas de Karnaugh com mais variáveis

**Exemplo:** Construa o Mapa de Karnaugh a partir da tabela-verdade e determine a expressão booleana do circuito lógico.

A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

**R:** Com respeito ao Mapa de Karnaugh,

	$\overline{C}\overline{D}$	$\overline{C}D$	$CD$	$C\overline{D}$
$\overline{A}\overline{B}$	0	1	0	0
$\overline{A}B$	0	1	0	0
$AB$	0	1	1	0
$A\overline{B}$	0	0	0	0

Portanto, sem realizar agrupamentos,

$$x = \overline{A}\overline{B}\overline{C}D + \overline{A}B\overline{C}D + AB\overline{C}D + ABCD$$

A expressão de saída  $x$  pode ser simplificada agrupando '1s' (ou '0s') adjacentes no Mapa de Karnaugh.

## Agrupamento de pares de '1s'

$AB \backslash C$	0	1
00	0	0
01	1	0
11	1	0
10	0	0

$S = B\bar{C}$

$AB \backslash C$	0	1
00	0	0
01	1	1
11	0	0
10	0	0

$S = \bar{A}B$

$AB \backslash C$	0	1
00	1	0
01	0	0
11	0	0
10	1	0

$S = \bar{B}\bar{C}$

$AB \backslash CD$	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

$S = \bar{A}\bar{B}C + A\bar{B}\bar{D}$

- Pares de '1s' adjacentes podem ser agrupados, resultando assim na eliminação de uma variável.
- As linhas superior e inferior, bem como as colunas mais à esquerda e mais à direita, são adjacentes entre si.

## Agrupamento de quartetos de '1s'

$AB \setminus C$	0	1
00	0	1
01	0	1
11	0	1
10	0	1

$S = C$

$AB \setminus CD$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	0	0

$S = AB$

$AB \setminus CD$	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

$S = \overline{B}\overline{D}$

$AB \setminus CD$	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

$S = BD$

$AB \setminus CD$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	0	0	1
10	1	0	0	1

$S = A\overline{D}$

- Quartetos de '1s' adjacentes podem ser agrupados, resultando na eliminação de duas variáveis.
- Os cantos, assim como as colunas mais à esquerda e mais à direita, são adjacentes entre si.

## Agrupamento de octetos de '1s'

AB\CD	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

$$S = B$$

AB\CD	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	1	1	0	0
10	1	1	0	0

$$S = \bar{C}$$

AB\CD	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

$$S = \bar{D}$$

AB\CD	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	0	0
10	1	1	1	1

$$S = \bar{B}$$

- Octetos de '1s' adjacentes podem ser agrupados, resultando assim na eliminação de três variáveis.
- As linhas superior e inferior, bem como as colunas mais à esquerda e mais à direita, são adjacentes entre si.

Apenas variáveis que não se alteram dentro de um agrupamento são mantidas.



# Implicações dos agrupamentos

O agrupamento de pares, quartetos e octetos em um Mapa de Karnaugh pode ser usado para obter uma expressão simplificada.

Especificamente, um agrupamento de

- dois '1s' (ou '0s') elimina **uma variável**;
- quatro '1s' (ou '0s') elimina **duas variáveis**;
- oito '1s' (ou '0s') elimina **três variáveis**; e
- dezesseis '1s' (ou '0s') elimina **quatro variáveis**.

Portanto, **a formação do maior agrupamento possível de '1s' deve sempre ser preterida**, visando eliminar o maior número de variáveis de um dado termo.

## Processo completo de simplificação

1. Monte o Mapa de Karnaugh (no formato apropriado).
2. **Insira '1s' nos quadrados correspondentes aos '1s' da tabela-verdade** e '0s' nos demais.
3. **Agrupe octetos de '1s'** (ou '0s').
4. **Agrupe quartetos de '1s'** (ou '0s').
5. **Agrupe pares de '1s'** (ou '0s').
6. **Agrupe '1s' isolados** (ou '0s').
7. Construa uma expressão na **forma de soma-de-produtos** (ou produto-de-somas) envolvendo todos os termos.

**Sobreposições são permitidas nos agrupamentos, i.e., mesmo '1s' já agrupados podem fazer parte de outros agrupamentos.**

## Processo completo de simplificação

**Exemplo:** Determine a expressão simplificada (soma-de-produtos) para o circuito lógico dado pelos seguintes mapas de Karnaugh:

a)

	$\overline{C}\overline{D}$	$\overline{C}D$	$CD$	$C\overline{D}$
$\overline{A}\overline{B}$	0	0	0	1
$\overline{A}B$	0	1	1	0
$AB$	0	1	1	0
$A\overline{B}$	0	0	1	0

b)

	$\overline{C}\overline{D}$	$\overline{C}D$	$CD$	$C\overline{D}$
$\overline{A}\overline{B}$	0	0	1	0
$\overline{A}B$	1	1	1	1
$AB$	1	1	0	0
$A\overline{B}$	0	0	0	0

## Processo completo de simplificação

**Exemplo:** Determine a expressão simplificada (soma-de-produtos) para o circuito lógico dado pelos seguintes mapas de Karnaugh:

a)

	$\overline{C}\overline{D}$	$\overline{C}D$	$CD$	$C\overline{D}$
$\overline{A}\overline{B}$	0	0	0	1
$\overline{A}B$	0	1	1	0
$AB$	0	1	1	0
$A\overline{B}$	0	0	1	0

b)

	$\overline{C}\overline{D}$	$\overline{C}D$	$CD$	$C\overline{D}$
$\overline{A}\overline{B}$	0	0	1	0
$\overline{A}B$	1	1	1	1
$AB$	1	1	0	0
$A\overline{B}$	0	0	0	0

**R:**

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

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

## Processo completo de simplificação

**Exemplo:** Determine duas expressões lógicas (agrupamentos diferentes) no Mapa de Karnaugh fornecido e identifique a melhor?

	$\overline{C}\overline{D}$	$\overline{C}D$	$CD$	$C\overline{D}$
$\overline{A}\overline{B}$	0	1	0	0
$\overline{A}B$	0	1	1	1
$AB$	0	0	0	1
$A\overline{B}$	1	1	0	1

## Processo completo de simplificação

**Exemplo:** Determine duas expressões lógicas (agrupamentos diferentes) no Mapa de Karnaugh fornecido e identifique a melhor?

	$\overline{C}\overline{D}$	$\overline{C}D$	$CD$	$C\overline{D}$
$\overline{A}\overline{B}$	0	1	0	0
$\overline{A}B$	0	1	1	1
$AB$	0	0	0	1
$A\overline{B}$	1	1	0	1

**R:** Do Mapa de Karnaugh, obtém-se

i)  $x = \overline{A}\overline{C}D + \overline{A}BC + A\overline{B}\overline{C} + AC\overline{D}$

ii)  $y = \overline{A}BD + BC\overline{D} + \overline{B}\overline{C}D + A\overline{B}\overline{D}$ .

Todavia, as duas expressões têm a mesma complexidade e, por isso, nenhuma é melhor do que a outra.

# Processo completo de simplificação

**Exemplo:** Use o método do Mapa de Karnaugh para simplificar:

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

Para preencher o Mapa de Karnaugh: i) Escreva a expressão na forma de soma-de-produtos; ii) Insira '1s' em quadrados cuja denominação correspondam a combinação de variáveis de entrada; e iii) Coloque '0s' nos demais.

# Processo completo de simplificação

**Exemplo:** Use o método do Mapa de Karnaugh para simplificar:

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

Para preencher o Mapa de Karnaugh: i) Escreva a expressão na forma de soma-de-produtos; ii) Insira '1s' em quadrados cuja denominação correspondam a combinação de variáveis de entrada; e iii) Coloque '0s' nos demais.

**R:**

	$\overline{C} \overline{D}$	$\overline{C} D$	$C D$	$C \overline{D}$
$\overline{A} \overline{B}$	1	1	0	1
$\overline{A} B$	1	1	0	1
$A B$	1	1	0	1
$A \overline{B}$	1	1	1	1

Portanto, a expressão simplificada  
(na forma de soma-de-produtos)  
para o circuito lógico é dada por

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



## Processo completo de simplificação

E, por que não considerar o agrupamento de '0s'?

	$\overline{C}\overline{D}$	$\overline{C}D$	$CD$	$C\overline{D}$
$\overline{A}\overline{B}$	1	1	0	1
$\overline{A}B$	1	1	0	1
$AB$	1	1	0	1
$A\overline{B}$	1	1	1	1

A expressão (na forma de produto-de-somas) é dada por

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

Contudo, o tamanho dos agrupamentos é menor; por isso, a expressão contém mais elementos em comparação com

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

## Condições de irrelevância (*don't care*)

- Condições de entrada para as quais a saída não é especificada, denominadas “*don't care*”.
- Usadas em combinações de entrada que não ocorrerão na prática.
- O projetista tem liberdade de definir a saída como 0/1 para otimizar a expressão lógica.
- O Mapa de Karnaugh pode ser utilizado para representar e otimizar a expressão lógica, incluindo condições “*don't care*” (usualmente, representadas por ‘x’).
- Portanto, o desafio é: Como tratar “*don't-cares*” para obter a expressão mais simples?

## Condições de irrelevância (*don't care*)

**Exemplo:** A partir da tabela-verdade apresentada, obtenha a expressão simplificada que descreve a lógica do circuito digital.

<i>A</i>	<i>B</i>	<i>C</i>	<i>z</i>
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	x
1	0	0	x
1	0	1	1
1	1	0	1
1	1	1	1

## Condições de irrelevância (*don't care*)

**Exemplo:** A partir da tabela-verdade apresentada, obtenha a expressão simplificada que descreve a lógica do circuito digital.

**R:** Com respeito ao Mapa de Karnaugh,

$A$	$B$	$C$	$z$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	x
1	0	0	x
1	0	1	1
1	1	0	1
1	1	1	1

	$\overline{C}$	$C$
$\overline{A}\overline{B}$	0	0
$\overline{A}B$	0	$x \rightarrow 0$
$AB$	1	1
$A\overline{B}$	$x \rightarrow 1$	1

Portanto,

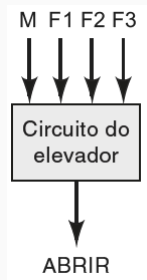
$$z = A.$$

## Condições de irrelevância (*don't care*)

**Exemplo:** Projete um circuito lógico que controla a porta de um elevador em um prédio de 2 pavimentos (térreo, 1º e 2º andar). Para tal, considere um circuito com 4 entradas, em que

- $M$  é um sinal lógico que indica quando o elevador está se movendo ( $M = 1$ ) ou parado ( $M = 0$ ); e
- $F1$ ,  $F2$  e  $F3$  são sinais indicadores (normalmente em nível BAIXO) dos andares, assumindo nível ALTO quando o elevador estiver naquele andar; e
- $M = 0$  e  $F1 = F2 = F3 = 0$  indica que o elevador está parado, mas não adequadamente alinhado com qualquer andar (porta fechada).

Por sua vez, a saída do circuito é o sinal ABRIR que assume nível ALTO quando a porta do elevador precisar ser aberta.



# Condições de irrelevância (*don't care*)

M	F1	F2	F3	ABRIR
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	X
0	1	0	0	1
0	1	0	1	X
0	1	1	0	X
0	1	1	1	X
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	X
1	1	0	0	0
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

	$\overline{F2}\overline{F3}$	$\overline{F2}F3$	$F2\overline{F3}$	$F2F3$
$\overline{M} \overline{F1}$	0	1	X	1
$\overline{M} F1$	1	X	X	X
$M \overline{F1}$	0	X	X	X
$M F1$	0	0	X	0

	$\overline{F2}\overline{F3}$	$\overline{F2}F3$	$F2\overline{F3}$	$F2F3$
$\overline{M} \overline{F1}$	0	1	1	1
$\overline{M} F1$	1	1	1	1
$M \overline{F1}$	0	0	0	0
$M F1$	0	0	0	0

$$ABRIR = \overline{M} (F1 + F2 + F3)$$

Para circuitos com um grande número de entradas (e.g.,  $> 4$ ), técnicas computacionais são mais apropriadas.

Dois circuitos lógicos especiais, que aparecem muitas vezes em sistemas digitais, são os circuitos *exclusive-OR* e *exclusive-NOR*, denominados usualmente portas XOR e XNOR.



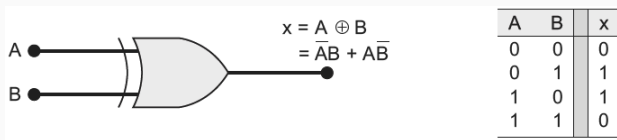
## Circuitos *exclusive*-OR (XOR)

Na álgebra booleana, a operação *exclusive*-OR (XOR) é representada por ' $\oplus$ ', i.e.,

$$\begin{aligned}x &= A \oplus B \\ &= A\overline{B} + \overline{A}B\end{aligned}$$

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

Com respeito a **tabela-verdade e símbolo** da “porta” XOR,



Note que  $x$  assume nível lógico 1 se as duas entradas estiverem em níveis opostos; caso contrário,  $x$  assume nível lógico 0.

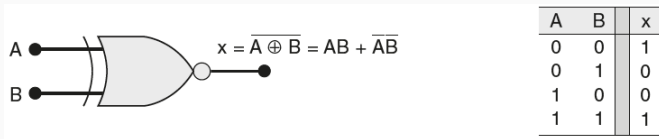
## Circuitos *exclusive*-NOR (XNOR)

Na álgebra booleana, a operação *exclusive*-NOR (XNOR) é representada por ' $\oplus$ ' com saída negada, i.e.,

$$\begin{aligned}x &= \overline{A \oplus B} \\ &= AB + \overline{A} \overline{B}\end{aligned}$$

a qual é lida como ' $x$  é igual a **NÃO**  $A$  **OU**  $B$  **EXCLUSIVO**'.

Com respeito a **tabela-verdade e símbolo** da “porta” XNOR,



Note que  $x$  assume nível lógico 1 se as duas entradas coincidirem; caso contrário,  $x$  assume nível lógico 0.

Não existem “portas” XOR e XNOR  
com mais de duas entradas.

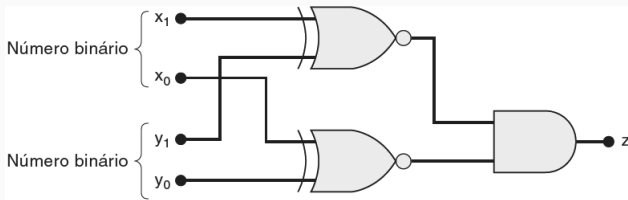
## Circuitos lógicos XOR e XNOR

**Exemplo:** Projete um circuito, usando “portas” XOR e/ou XNOR, para detectar quando dois números binários de dois dígitos, i.e.,  $x_1x_0$  e  $y_1y_0$ , são iguais; nesse caso, produza nível lógico 1 na saída.

# Circuitos lógicos XOR e XNOR

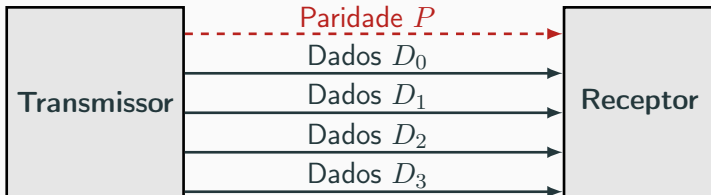
**Exemplo:** Projete um circuito, usando “portas” XOR e/ou XNOR, para detectar quando dois números binários de dois dígitos, i.e.,  $x_1x_0$  e  $y_1y_0$ , são iguais; nesse caso, produza nível lógico 1 na saída.

**R:** Embora as técnicas de projeto estudadas possam ser usadas, a natureza do problema *per se* indica o uso de portas XNOR (coincidência). Dessa forma,



# Circuitos gerador e verificador de paridade

- Um transmissor pode “anexar” um bit de paridade em um conjunto de bits de dados antes de transmiti-lo ao receptor.
- Esse **bit de paridade possibilita ao receptor detectar erros** que possam ter ocorrido na transmissão.
- Esse bit de paridade é transmitido para o receptor juntamente com os bits do dado original, totalizando 5 bits.



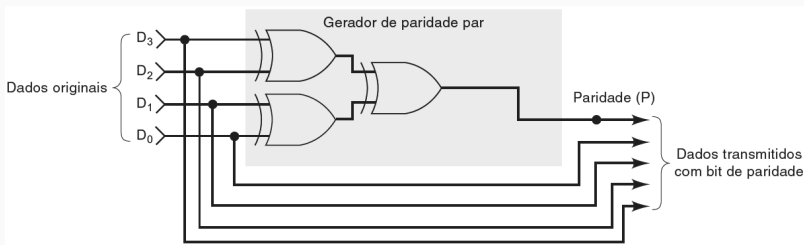
## Circuitos gerador e verificador de paridade

**Exemplo:** Projete um circuito, usando “portas” XOR e/ou XNOR, gerador de paridade “par” para grupos de 4 bits.

# Circuitos gerador e verificador de paridade

**Exemplo:** Projete um circuito, usando “portas” XOR e/ou XNOR, gerador de paridade “par” para grupos de 4 bits.

R:



Os dados “colocados” no barramento são usado como entradas do circuito gerador de paridade, o qual produz um bit de paridade  $P$  na saída.



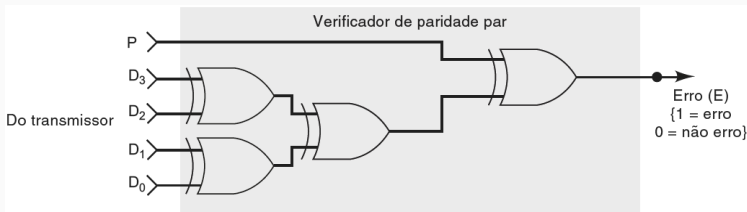
## Circuitos gerador e verificador de paridade

**Exemplo:** Projete um circuito, usando “portas” XOR e/ou XNOR, verificador de paridade “par” para grupos de 5 bits (4 bits de dados e 1 bit de paridade).

# Circuitos gerador e verificador de paridade

**Exemplo:** Projete um circuito, usando “portas” XOR e/ou XNOR, verificador de paridade “par” para grupos de 5 bits (4 bits de dados e 1 bit de paridade).

R:



Esses circuitos empregam “portas” XOR, as quais produzem saída 1 caso o número de entradas ‘1s’ seja ímpar; por outro lado, caso o número de ‘1s’ nas entradas seja par, a saída é 0.

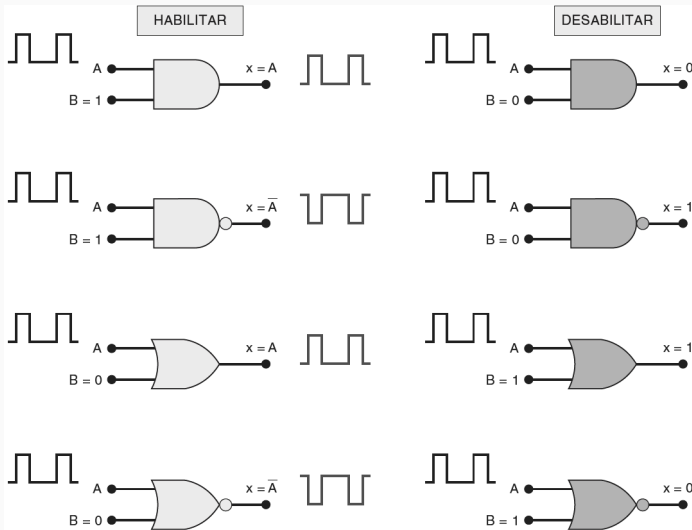
# Circuitos de habilitação

As portas lógicas básicas podem ser usadas para controlar a passagem de um sinal da entrada para a saída, servindo como um circuito habilitador/desabilitador.

- Um sinal lógico é aplicado em uma das entradas da porta, sendo a outra usada para controle.
- A entrada de controle determina se o sinal de entrada está habilitado ou impedido (desabilitado) de alcançar a saída.
- Essa ação de controle é a razão para esses circuitos serem denominados portas.



# Circuitos de habilitação



- As **formas canônicas** de expressões booleanas foram mostradas, i.e., i) **soma-de-produtos**; e ii) **produto-de-somas**.
- Um **procedimento sistemático de projeto** foi apresentado:
  - 1) Construir uma tabela-verdade com o comportamento desejado.
  - 2) Obter uma expressão booleana em uma das formas canônicas.
  - 3) Simplificar a expressão de saída obtida (quando possível), via
    - Álgebra booleana (depende da experiência do projetista); **ou**
    - **Método do Mapa de Karnaugh** (procedimento sistemático).
  - 4) Implementar expressão final (simplificada).
- Alguns **circuitos lógicos especiais** foram discutidos, usando
  - “Portas” XOR:  $x = A \oplus B \rightarrow$  **Saída 1 se  $A$  e  $B$  são opostos.**
  - “Portas” XNOR:  $x = \overline{A \oplus B} \rightarrow$  **Saída 1 se  $A$  e  $B$  são iguais.**
- O conceito de **circuitos de habilitação** usando portas lógicas básicas foi revisitado.

**Sugestão de leitura:** Seções 4.10–4.13,  
as quais tratam sobre a Análise de Defeitos.

# Considerações finais

## Exercícios sugeridos:

4.1(b),(d) e (h), 4.4–4.7, 4.11, 4.12, 4.14, 4.16, 4.20, 4.22 e 4.31

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

## 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 5)

Até a próxima aula... =)