# Circuitos Lógicos e Organização de Computadores

## Capítulo 4 – Implementações Otimizadas de Funções Lógicas

Ricardo Pannain

pannain@puc campinas.edu.br

http://docentes.puc campinas.edu.br/ceatec/pannain/

1

## Mapa de Karnaugh

Seja a função $f = ? m(0, 2, 4, 5, 6)$, olhando para a tabela verdade,
É difícila verificar que f=1 quando x3 = 0 ou x1 = 1 e x2 = 0.
Outra maneira de se representar uma função ✍ Mapa de Karnaugh

| **Número da linha** | **x1** | **x2** | **x3** | **f** |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | 1 |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 0 |

2

## Mapa de Karnaugh

É uma aplicação sistemática das propriedades 14a e 14b

14a) $x \cdot y + x \cdot \overline{y} = x$ 14b) $(x + y) \cdot (x + \overline{y}) = x$

Exemplo da tabela do slide anterior $f = (m_0, m_2, m_4, m_5, m_6)$

$f = \overline{x1}\ \overline{x2}\ \overline{x3} + \overline{x1}\ x2\ \overline{x3} + x1\ \overline{x2}\ \overline{x3} + x1\ \overline{x2}\ x3 + x1\ x2\ \overline{x3}$

Aplicando 14a com $(m_0, m_2)$ e $(m_4, m_6)$, temos:

1) $\overline{x1}(\overline{x2}\ \overline{x3} + x2\ \overline{x3}) = \overline{x1}\ \overline{x3}$ e 2) $x1(\overline{x2}\ \overline{x3} + x2\ \overline{x3}) = x1\ \overline{x3}$

Aplicando novamente em 1) e 2) ✍ $\overline{x1}\ \overline{x3} + x1\ \overline{x3} = \overline{x3}$

$m_0, m_2, m_4$ e $m_6$ foram trocados por $\overline{x3}$ ✍ os mintermos estão incluídos em $\overline{x3}$

$m_5$ pode ser combinado com $m_4$ ✍ $x1(\overline{x2}\ \overline{x3} + \overline{x2}\ x3) = x1\ \overline{x2}$

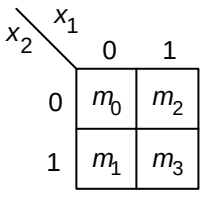
✍ $f = \overline{x3} + x1\ \overline{x2}$ ✍ (expressão de custo mínimo)



## Mapa de Karnaugh – 2 variáveis

| $x_1$ | $x_2$ | |
|---|---|---|
| 0 | 0 | $m_0$ |
| 0 | 1 | $m_1$ |
| 1 | 0 | $m_2$ |
| 1 | 1 | $m_3$ |

(a) Tabela Verdade




(b) Mapa de Karnaugh

## Mapa de Karnaugh

Estratégia de minimização ✍ encontrar sempre que possível, os maiores grupos de 1s - SOP (ou 0s - POS), que cubram todos os casos onde o valor de f = 1 ($\bar{f}$ = 1)

TERMINOLOGIA

LITERAL ✍ $x_i$ ou $\bar{x}_i$
IMPLICANTE ✍ termo produto onde f = 1 (SOP)
PRIMO IMPLICANTE ✍ um implicante que não pode se combinado com outro, i. é, nenhum literal deste implicante pode ser suprimido.
COBERTURA ✍ conjunto de implicantes que determina o valor 1 para a função
CUSTO ✍ ? no. de portas + ? no. de entradas (assumir que entradas e entradas barradas estão disponíveis)
MENOR CUSTO ✍ quando a cobertura de uma função consiste de primos implicantes (essenciais ou não)
Se um primo implicante é um mintermo, que não está incluído em outro primo implicante, então ele deve ser incluído na cobertura e é chamado de primo implicante essencial

5

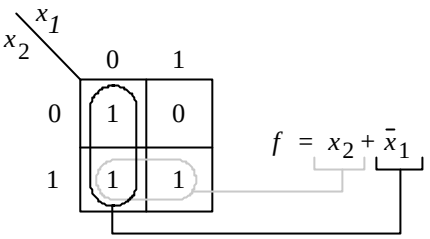
## Mapa de Karnaugh

Processo para encontrar circuito de menor custo:

- Gerar todos os primos implicantes de uma dada função f.
- Encontrar o conjunto de primos implicantes essenciais.
- Se o conjunto cobrir todas as possibilidades para f = 1, temos o circuito de menor custo. Caso contrário, determinar a(s) primo(s) implicante(s) não essenciais, que seriam adicionados para completar a cobertura de custo mínimo. (esta escolha grealmnete não é obvia ✍ usar heurística para escolher e melhor solução.

6

Minimização de uma função lógica de 2 variáveis usando Mapa de Karnaugh




7

Mapa de Karnaugh – 3 variáveis

| $x_1$ | $x_2$ | $x_3$ | |
|---|---|---|---|
| 0 | 0 | 0 | $m_0$ |
| 0 | 0 | 1 | $m_1$ |
| 0 | 1 | 0 | $m_2$ |
| 0 | 1 | 1 | $m_3$ |
| 1 | 0 | 0 | $m_4$ |
| 1 | 0 | 1 | $m_5$ |
| 1 | 1 | 0 | $m_6$ |
| 1 | 1 | 1 | $m_7$ |

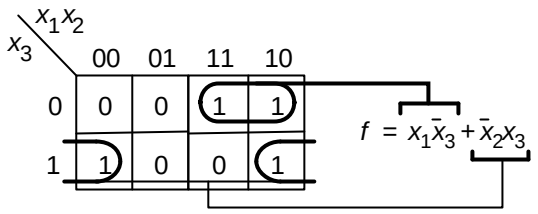
(a) Tabela Verdade

| $x_3$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | $m_0$ | $m_2$ | $m_6$ | $m_4$ |
| 1 | $m_1$ | $m_3$ | $m_7$ | $m_5$ |

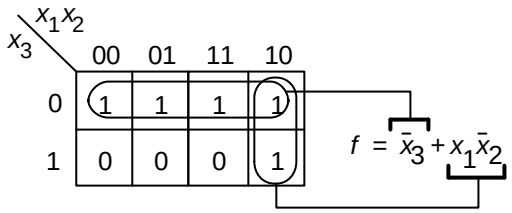
(b) Mapa de Karnaugh

8

## Minimização de uma função lógica de 3 variáveis usando Mapa de Karnaugh



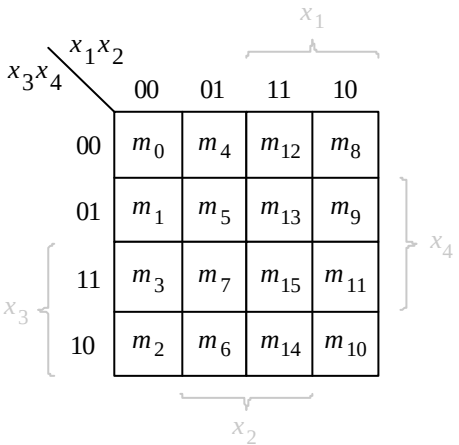

(a) Exemplo de simplificação




(b) Função do slide 2

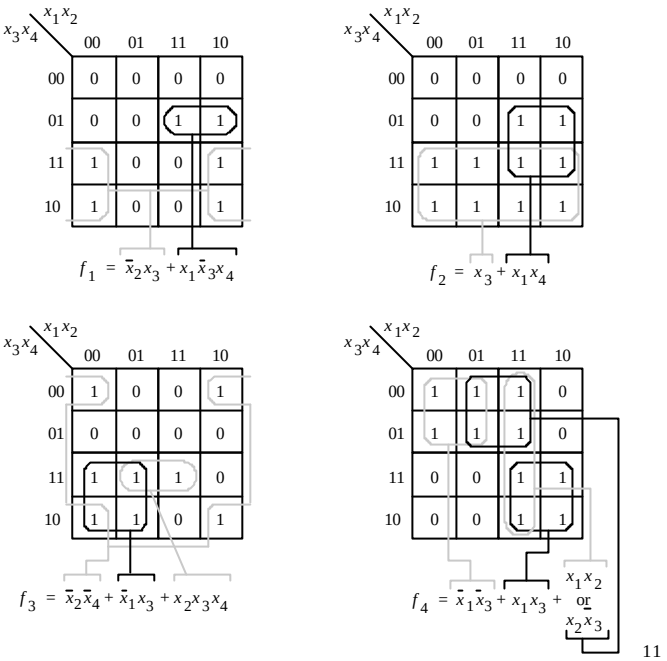
9

## Mapa de Karnaugh – 4 variáveis



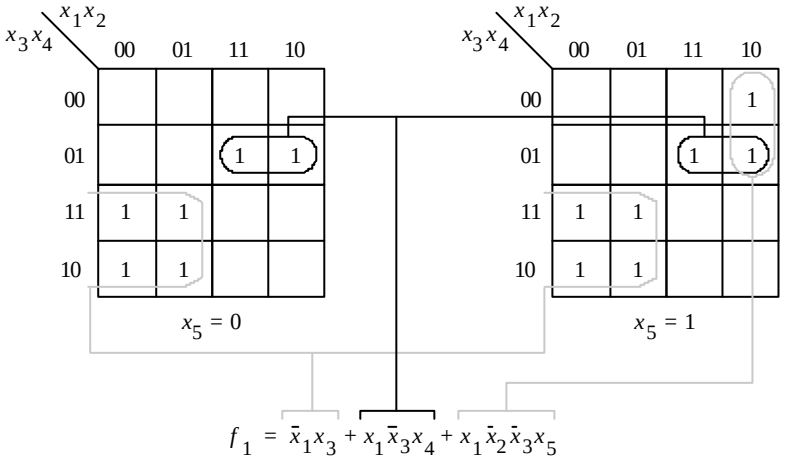

10

Mínimo 4 maiúsculas para variáveis e - s




11

## Simplificação de uma função lógica de 5 variáveis usando Mapa de Karnaugh




12

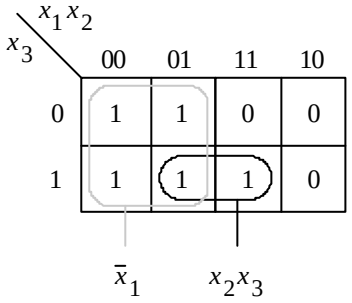
## Mapa de Karnaugh- Exercícios

*1)* $f = ?\ m(0, 1, 2, 3, 7)$
*2)* $f = ?\ m(2, 3, 5, 6, 7, 10, 11, 13, 14)$
*3)* $f = ?\ m(0, 4, 8, 10, 11, 12, 13, 15)$
*4)* $f = ?\ m(0, 2, 4, 5, 10, 11, 13, 15)$

13

## Mapa de Karnaugh- Exercícios

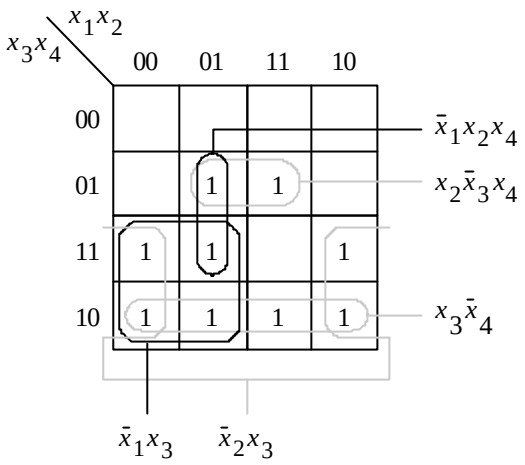
1) $f = ?\ m(0, 1, 2, 3, 7)$




14

## Mapa de Karnaugh- Exercícios

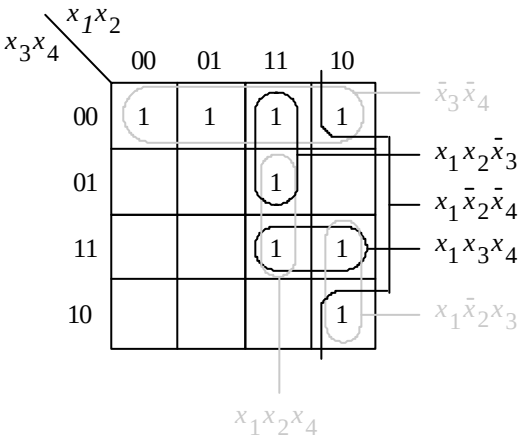
2) $f = ?\ m(2, 3, 5, 6, 7, 10, 11, 13, 14)$




15

## Mapa de Karnaugh- Exercícios

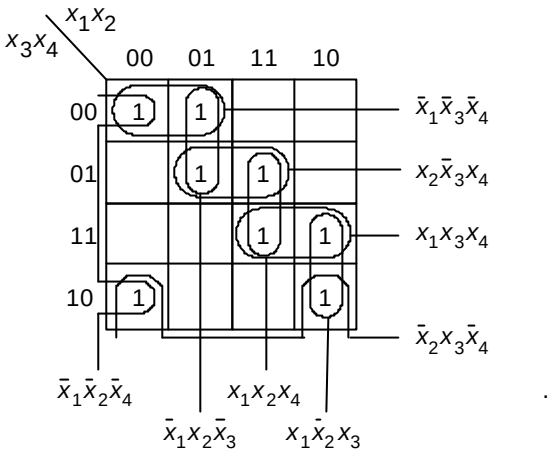
3) $f = ?\ m(0, 4, 8, 10, 11, 12, 13, 15)$




16

## Mapa de Karnaugh- Exercícios

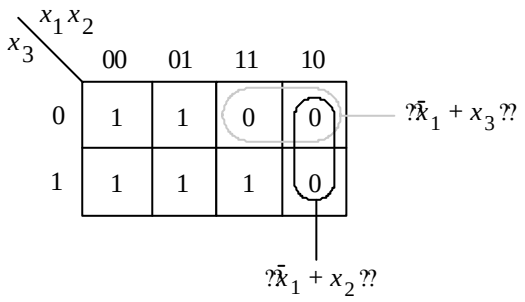
4) $f = ?\ m(0, 2, 4, 5, 10, 11, 13, 15)$




17

## Mapa de Karnaugh – Minimização usando POS

Função $f = ?\ M(4, 5, 6)$




OBS. -

Para se obter a implementação POS de custo mínimo, a partir do SOP ✍ fazer f = $\overline{f}$

18

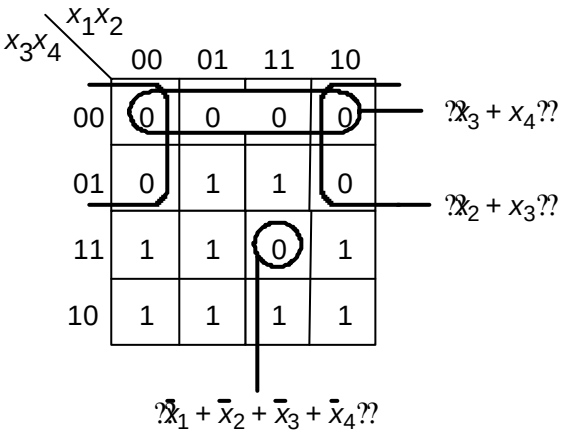
## Mapa de Karnaugh- Exercícios

Minimizar a função $f = ? M(0, 1, 4, 8, 9, 12, 15)$ e obter a implementação POS de custo mínimo, a partir do SOP

19

## Mapa de Karnaugh- Exercícios

$f = ? M(0, 1, 4, 8, 9, 12, 15)$




20

## Método de Quine McKluskey

**O método de Quine-McCluskey para encontrar primos implicantes de uma função booleana, usa um procedimento sistemático para tabulá-los, iniciando com o mintermos e usando a expressão AB + AB´ repetidamente. Os passos básicos para este método são:**

- **Listar todos os mintermos e agrupá-los pelo número de 1s que eles contenham.**
- **Formar pares de mintermos que diferem por uma variável e criar um novo termo com uma variável a menos (a variável que difere).**
- **Repetir o passo 2 até não existir um novo termo a ser formado. O resultado é um conjunto de primos implicantes da função.**
- **Formar uma tabela onde os mintermos originais definem as colunas e os primos implicantes definem as linhas. A relação entre cada mintermo e um dado primo implicante é indicado por um *X* no cruzamento da linha e coluna, respectivamente, referentes a ambos.**
- **Usando a tabela, determinar o primo implicante essencial a um conjunto adicional de primos implicantes que cobrem toda a função.**

21

## Método de Quine McKluskey

**Exemplo: f = ? m(0,2,3,7,8,10,11,13)**
**Encontrando primos implicantes**

| Mintermos | Termo | Termo |
|---|---|---|
| 0 0000 † | 0,2 00_0 † | 0,2,8,10 _0_0 * |
| --- | 0,8 _000 † | --- |
| 2 0010 † | --- | 2,3,10,11 _01_ * |
| 8 1000 † | 2,3 001_ † | |
| --- | 2,10 _010 † | |
| 3 0011 † | 8,10 10_0 † | |
| 10 1010 † | --- | |
| --- | 3,7 0_11 * | |
| 7 0111 † | 3,11 _011 † | |
| 11 1011 † | 3,10 101_ † | |
| 13 1101 * | --- | |
| --- | | |

22

## Método de Quine McKluskey

| Mintermos | 0 | 2 | 3 | 7 | 8 | 10 | 11 | 13 |
|---|---|---|---|---|---|---|---|---|
| Primos Implicantes | | | | | | | | |
| 13 * | | | | | | | | X |
| 0,8 | X | | | | X | | | |
| 2,3 | | X | X | | | | | |
| 2,10 | | X | | | | X | | |
| 3,7* | | | X | X | | | | |
| 3,10 | | | X | | | X | | |
| 3,11 | | | X | | | | X | |
| 10,11 | | | | | | X | X | |
| 0,2,8,10 * | X | X | | | X | X | | |
| 2,3,10,11 * | | X | X | | | X | X | |

$$f = (13) + (3,7) + (2,3,10,11) + (0,2,8,10) = x1\,x2\,\overline{x3}\,x4 + x1\,\overline{x3}\,x4 + \overline{x2}\,x3 + \overline{x2}\,\overline{x4}$$

23

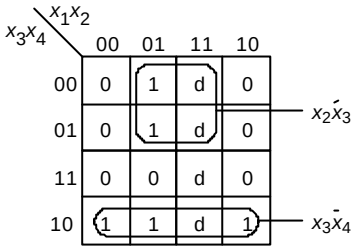
## Funções especificadas não completamente

Em sistemas digitais, freqüêntemnte temos certas entradas que nunca ocorrem. Pro exemplo: 2 chaves x1 e x2 interligadas que nunca podem ser fechadas ao mesmo tempo ✍ (x1,x2) = (0,0) , (0,1) ou (1, 0). A combinação (1,1) nunca vai ocorrer. Chamamos esta combinação de don't care conditions.

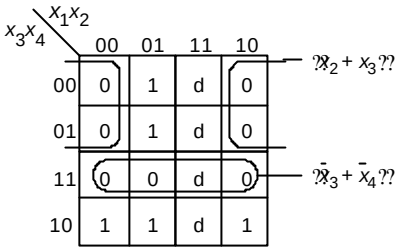
24

## Funções especificadas não completamente

$f = ?\ m(2, 4, 5, 6, 10) + D(12, 13, 14, 15)$




(a) SOP implementation

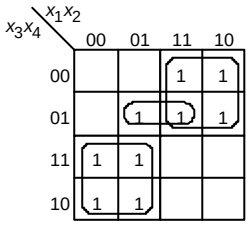



(b) POS implementation

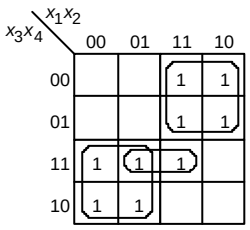
OBS. – Escrever a função sem levar em conta o don't care e comparar

25

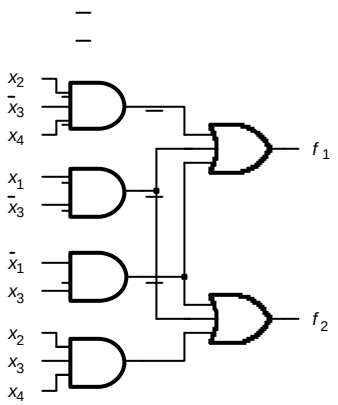
## Circuitos de múltiplas saídas




(a) Função $f_1$



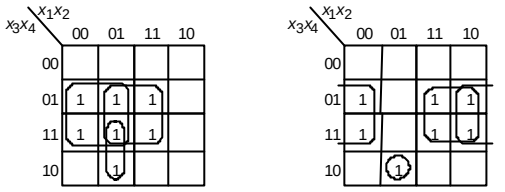

(b) Função $f_2$




(c) Circuito combinando $f_1$ and $f_2$

26

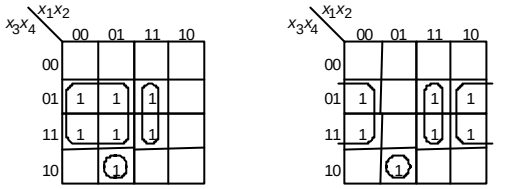
## Circuitos de múltiplas saídas




(a) Funçao otimizada $f_3$




(b) Função otimizada $f_4$



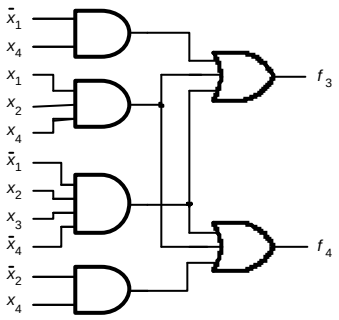




(c) Otimização usando $f_3$ e $f_4$ juntas

27

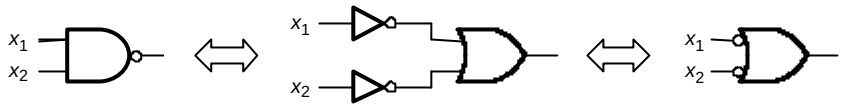
## Circuitos de múltiplas saídas



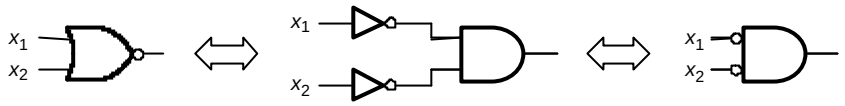

(d) Circuito combinado $f_3$ e $f_4$

28

## Teorema de DeMorgan ✍ circuitos com NANDs e NORs



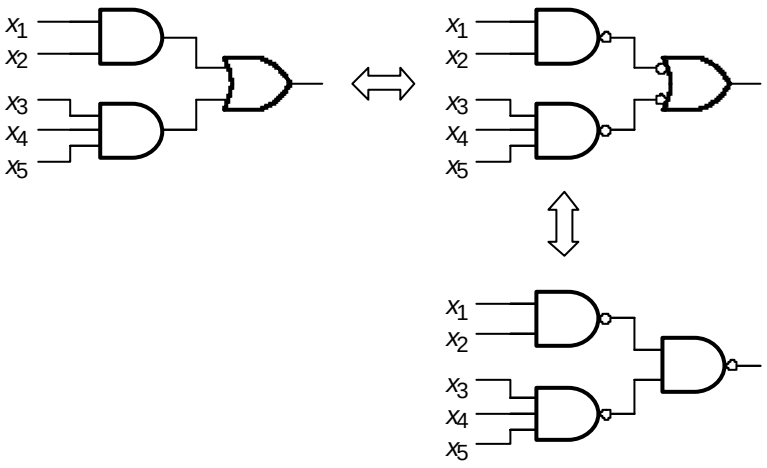

(a) $\overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2$




(b) $\overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2$

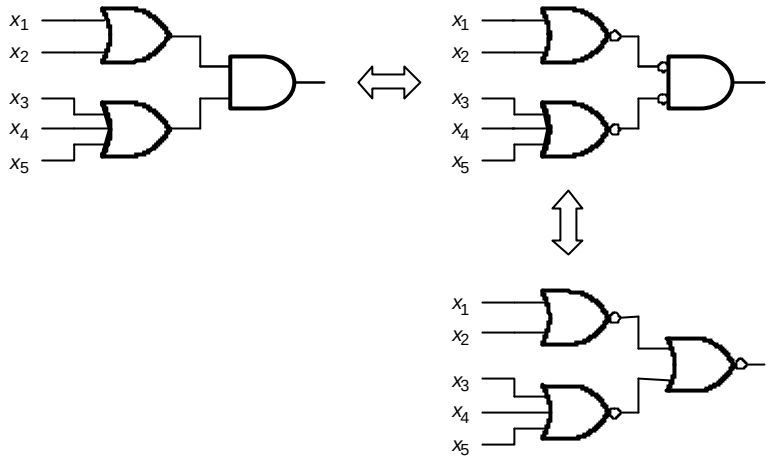
29

## Circuitos com NANDs




30

## Circuitos com NORs




31

## Decomposição Funcional – Circuitos multiníveis

A complexidade de um circuito lógico pode ser reduzido por decomposição de um circuito de 2 níveis em sub-circuitos, onde um ou mais sub-circuitos implementem funções comuns no circuito final.

Exemplo

$f = \overline{x1}\ x2\ x3 + \overline{x1}\ x2\ x3 + x1\ x2\ x4 + \overline{x1}\ \overline{x2}\ x4$

custo = 4 ANDs + 2 NOT + 1 OR + 18 entradas

$f = (\overline{x1}\ x2 + x1\ \overline{x2})x3 + (x1\ x2 + \overline{x1}\ \overline{x2})x4$

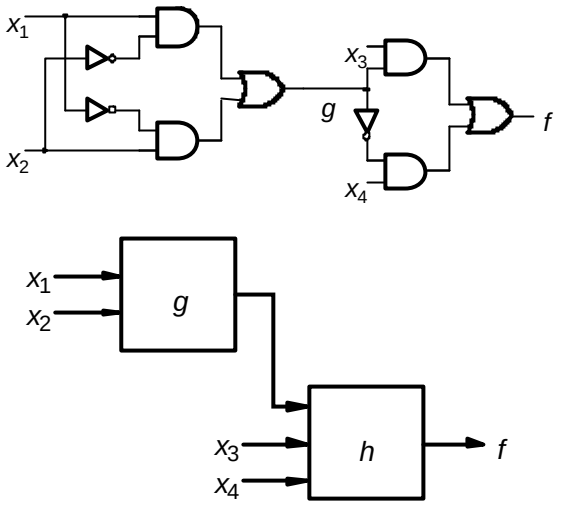
Supor $g(x1,x2) = (\overline{x1}\ x2 + x2\ \overline{x2})$ ✍ $\overline{g} = x1\ x2 + \overline{x1}\ \overline{x2}$

✍ $f = g\ x3 + \overline{g}\ x4$ ✍ menos 3 entradas, mais 1 OR e mais 1 NOT

$f = h\ (\ g\ (x1,x2),\ x3,\ x4)$

32

## Decomposição Funcional – Circuitos multiníveis




33

## Exemplo de decomposição de uma função

Seja a função f definida por:

| $x_3x_4$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | | | |
| 01 | | 1 | 1 | 1 |
| 11 | 1 | | | |
| 10 | | 1 | 1 | 1 |

$x_5 = 0$

| $x_3x_4$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | | |
| 10 | 1 | 1 | 1 | 1 |

$x_5 = 1$

(a) Mapa de Karnaugh para a função *f*

$f = x1\overline{x3}x4 + x1x3\overline{x4} + x2\overline{x3}x4 + x2x3\overline{x4} + \overline{x3}x4x5 + x3\overline{x4}x5 + \overline{x1}\,\overline{x2}x3\overline{x4}\,\overline{x5} + \overline{x1}\,\overline{x2}x3x4\overline{x5}$

Usaremos a decomposição para chegar a um circuito de menor custo

34

## Exemplo de decomposição de uma função

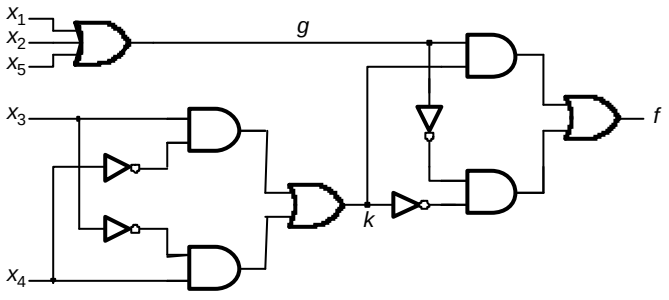
Seja g( x1, x2,x5) = x1 + x2 + x5 ✍ definida pelas linhas azuis

| $x_5$ \ $x_1x_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Linhas onde g ocorre ✍ k = $\overline{x3}$x4 + x3$\overline{x4}$ ✍ k.g representa a parte de f que é definida pelas linhas 2 e 4 dos mapas do slide anterior
Para as linhas 1 e 3 ✍ x1 = x2 = x5 = 0 ✍ $\overline{g}$ ✍ ($\overline{x3}$$\overline{x4}$ + x3x4) $\overline{g}$
✍ f(x1, x2, x3, x4,x5) = $\overline{k}$ $\overline{g}$ + k g

35

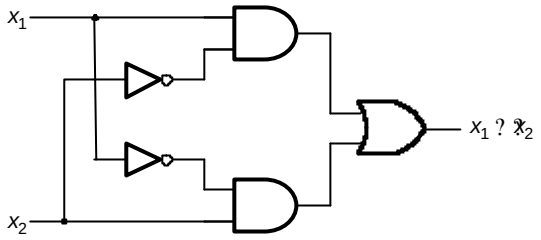
## Exemplo de decomposição de uma função



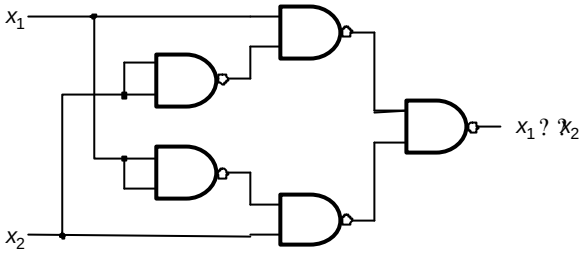

(b) Circuito obtido usando decomposição

36

Implementação de um XOR




(a) Implementação SOP

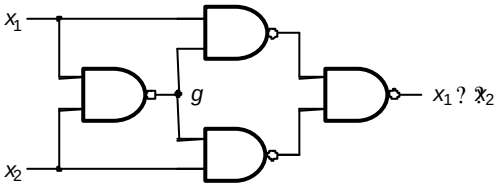



(b) Impemantação com NAND

37

Implementação de um XOR

$$f = x_1 ? x_2 = x_1\bar{x}_2 + \bar{x}_1x_2 = x_1(\bar{x}_1 + \bar{x}_2) + x_2(\bar{x}_1 + \bar{x}_2)$$

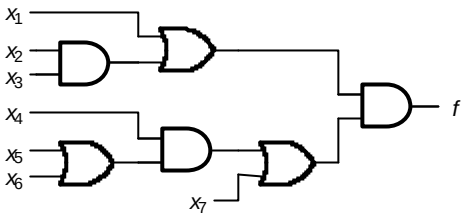



(c) Implementação otimizada com NAND

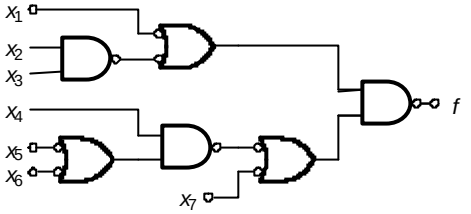
Figure 4.29 b Implementation of XOR

38

## Conversão para um circuito com NANDs



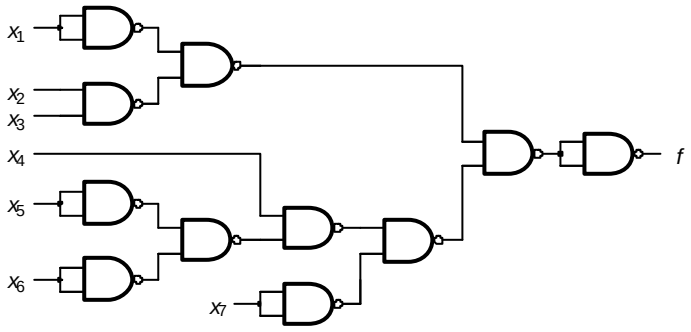

(a) Circuito com portas AND e OR




(b) Inversões necessárias para converter em NANDs

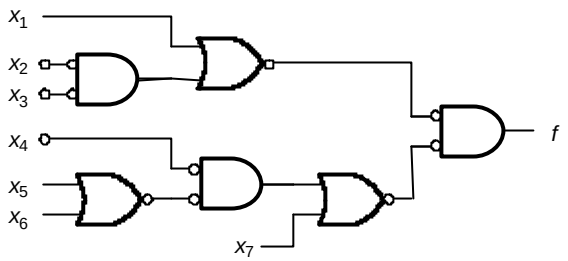
39

## Conversão para um circuito com NANDs




40

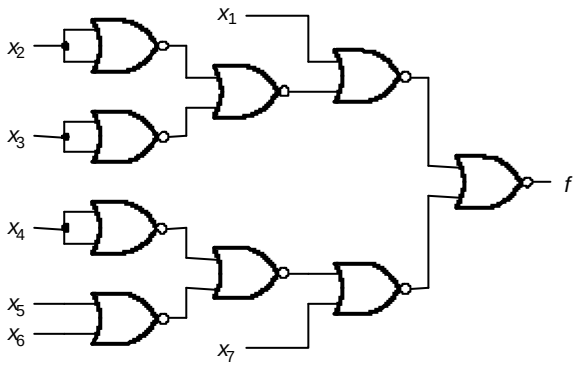
Conversão para um circuito com NORs




(a) Inversões necessárias para converter em NORs

41

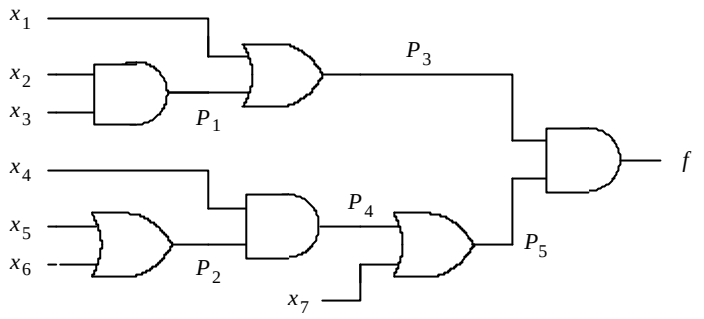
Conversão para um circuito com NORs




42

## Análise de circuitos multi-níveis

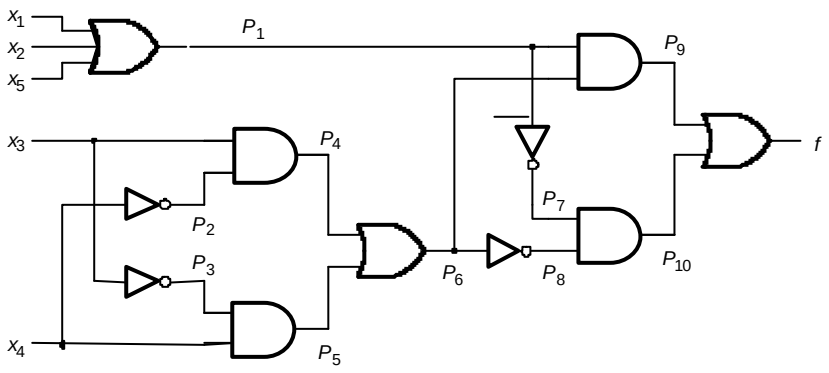
Exemplo: Escrever a função f




43

## Análise de circuitos multi-níveis

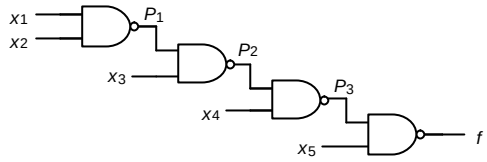
Exemplo: Escrever a função f



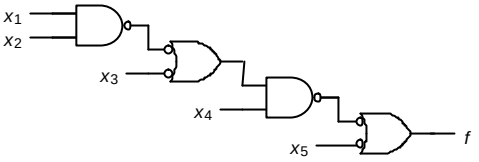

44

## Análise de circuitos multi-níveis

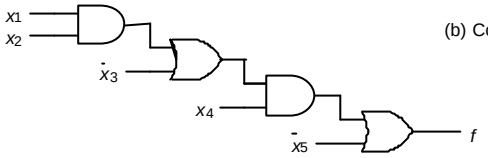
Exemplo: Converter para ANDs e ORs e escrever a função f




(a) Circuito com NAND




(b) Convertendo para ANDs and ORs

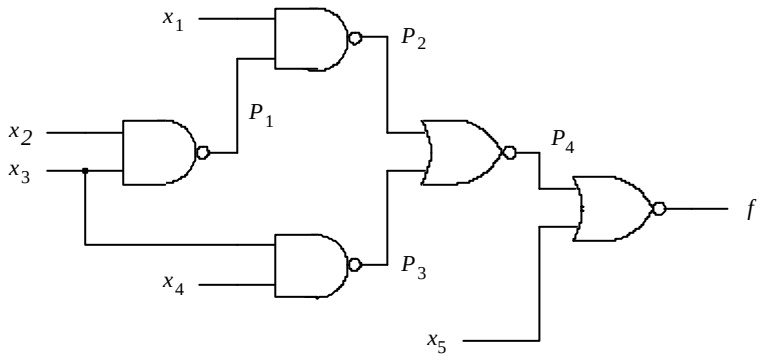



(c) Circuito com ANDs e ORs

45

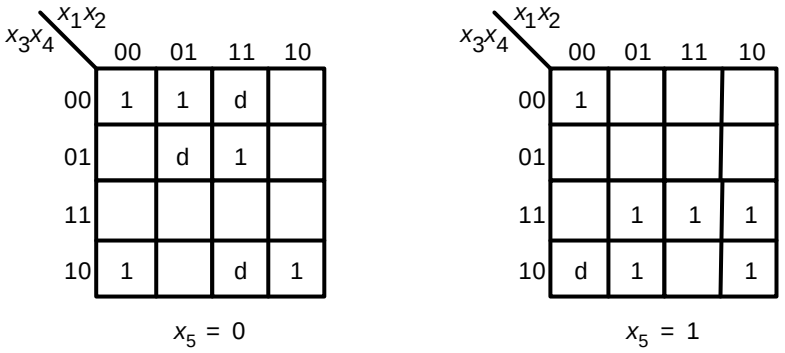
## Análise de circuitos multi-níveis

Exemplo: Escrever a função f




46

## EXERCÍCIO




Escrever a função f

47

Exemplo de código VHDL para a função $f = ? \; m(1, 4, 5, 6)$

```
ENTITY func1 IS
    PORT ( x1, x2, x3   : IN    BIT ;
           f            : OUT   BIT ) ;
END func1 ;

ARCHITECTURE LogicFunc OF func1 IS
BEGIN
    f <= (NOT x1 AND NOT x2 AND x3) OR
         (x1 AND NOT x2 AND NOT x3) OR
         (x1 AND NOT x2 AND x3) OR
         (x1 AND x2 AND NOT x3) ;
END LogicFunc ;
```

48

Código VHDL usando STD_LOGIC

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY func2 IS
     PORT ( x1, x2, x3   : IN     STD_LOGIC ;
                f        : OUT  STD_LOGIC ) ;
END func2 ;

ARCHITECTURE LogicFunc OF func2 IS
BEGIN
     f <= (NOT x1 AND NOT x2 AND x3) OR
          (x1 AND NOT x2 AND NOT x3) OR
          (x1 AND NOT x2 AND x3) OR
          (x1 AND x2 AND NOT x3) ;
END LogicFunc ;
```

49

Exemplo de código VHDL para a função $f = ? \ m(0, 2, 4, 5, 6)$

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY func3 IS
     PORT ( x1, x2, x3   : IN     STD_LOGIC ;
                f        : OUT  STD_LOGIC ) ;
END func3 ;

ARCHITECTURE LogicFunc OF func3 IS
BEGIN
     f <= (NOT x1 AND NOT x2 AND NOT x3) OR
          (NOT x1 AND x2 AND NOT x3) OR
          (x1 AND NOT x2 AND NOT x3) OR
          (x1 AND NOT x2 AND x3) OR
          (x1 AND x2 AND NOT x3) ;
END LogicFunc ;
```

50

Exemplo de código VHDL para a função $f = ? \ m(2, 3, 9, 10, 11, 13)$

```
LIBRARY ieee ;
USE ieee.std logic 1164.all ;

ENTITY func4 IS
     PORT ( x1, x2, x3, x4   : IN     STD LOGIC ;
              f              : OUT  STD LOGIC ) ;
END func4 ;

ARCHITECTURE LogicFunc OF func4 IS
BEGIN
     f <= (NOT x1 AND NOT x2 AND x3 AND NOT x4) OR
          (NOT x1 AND NOT x2 AND x3 AND x4) OR
          (x1 AND NOT x2 AND NOT x3 AND x4) OR
          (x1 AND NOT x2 AND x3 AND NOT x4) OR
          (x1 AND NOT x2 AND x3 AND x4) OR
          (x1 AND x2 AND NOT x3 AND x4) ;
END LogicFunc ;
```

51

Exemplo de código VHDL para uma função com 7 variáveis

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY func5 IS
     PORT ( x1, x2, x3, x4, x5, x6, x7   : IN     STD_LOGIC ;
              f                          : OUT  STD_LOGIC ) ;
END func5 ;

ARCHITECTURE LogicFunc OF func5 IS
BEGIN
     f <= (x1 AND x3 AND NOT x6) OR
          (x1 AND x4 AND x5 AND NOT x6) OR
          (x2 AND x3 AND x7) OR
          (x2 AND x4 AND x5 AND x7) ;
END LogicFunc ;
```

52