PCS 3115

Sistemas Digitais I

Síntese de Circuitos Combinatórios

Prof. Dr. Marcos A. Simplicio Jr.

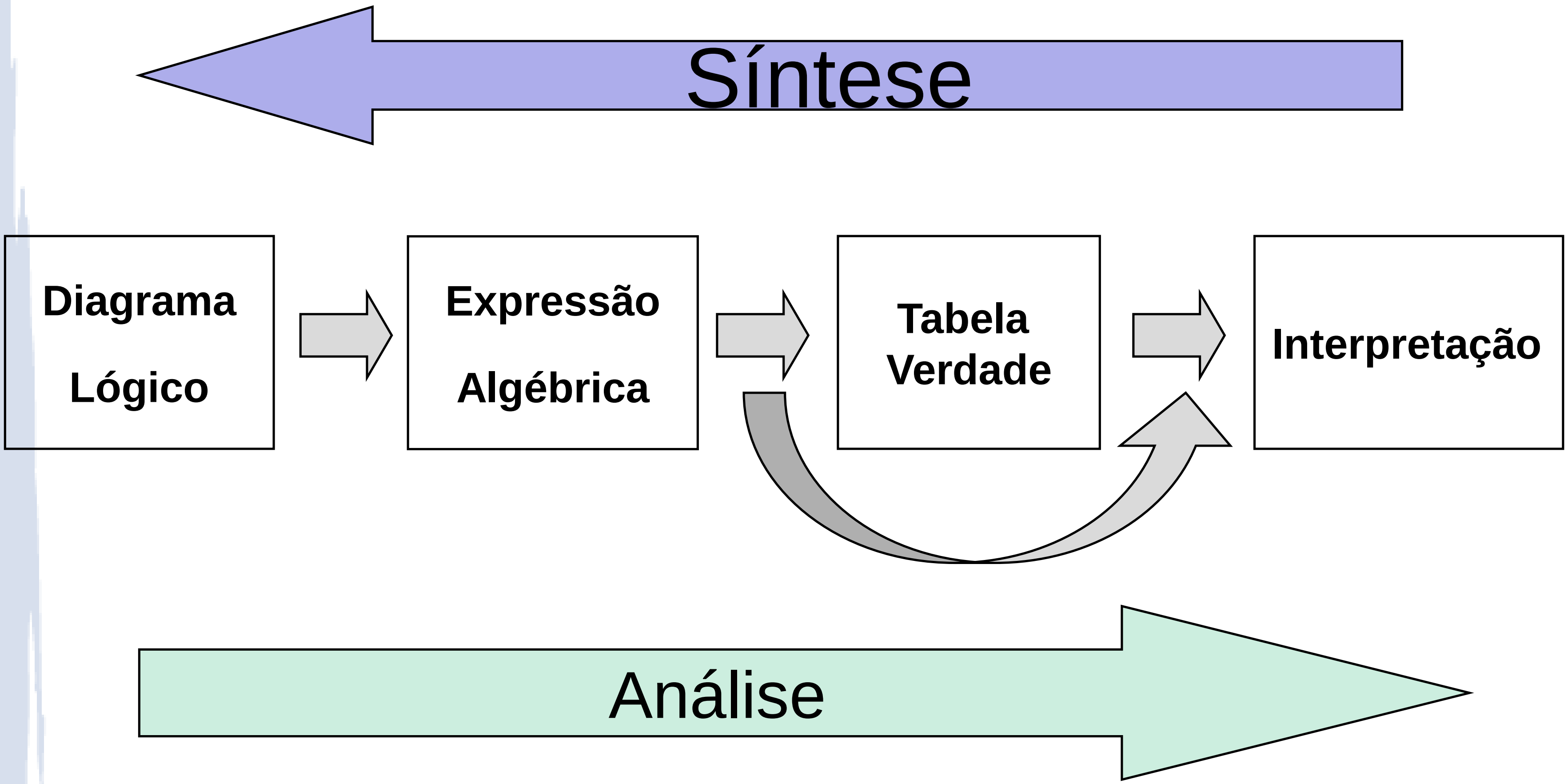
versão: 3.2 (Jan/2020)

Objetivos da aula

- Obter o circuito combinatório (diagrama) a partir de uma função lógica.
- Minimização com mapas de Karnaugh: Conceito, construção e utilização de mapas até 4 variáveis.
- Exemplos de síntese com VHDL

- Referência: seções 4.3.1 a 4.3.5 (início)

Síntese e Análise



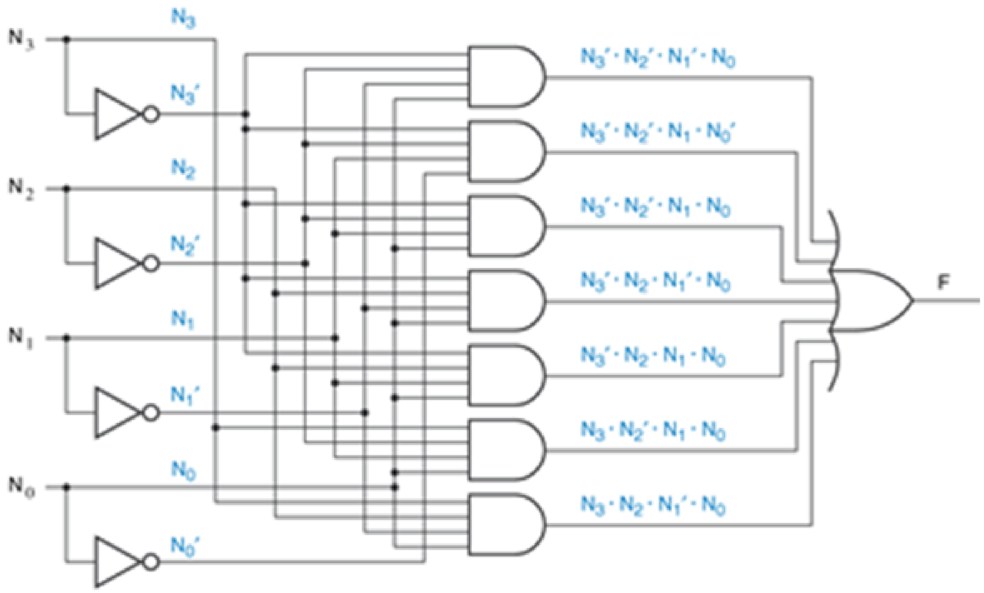
Síntese de Circuitos Combinatórios

- Dada a descrição do problema, interpretar, fazer a descrição lógica (VHDL, Tabela Verdade, expressão algébrica e simplificações) até obter o diagrama lógico para implementação.

- Métodos vistos até o momento são a base para o processo.... Vamos ver isso em um exemplo: detector de números primos de 4 bits
  - Quais as entradas que levam a uma saída 1?
  - Descreva esse circuito na forma de soma de mintermos
  - Desenhe esse circuito com portas lógicas

Ex: detector de nº primo (4 bits)

- Dada uma entrada de 4 bits $N = N_3N_2N_1N_0$, produza saída 1 para N = {1,2,3,5,7,11,13}, e 0 caso contrário.
  - Nota: 1 não é de fato um primo, mas vamos considerar que ele é porque isso leva a um design mais interessante
- **Soma de mintermos**: $F = \sum_{N_3N_2N_1N_0}(1,2,3,5,7,11,13)$

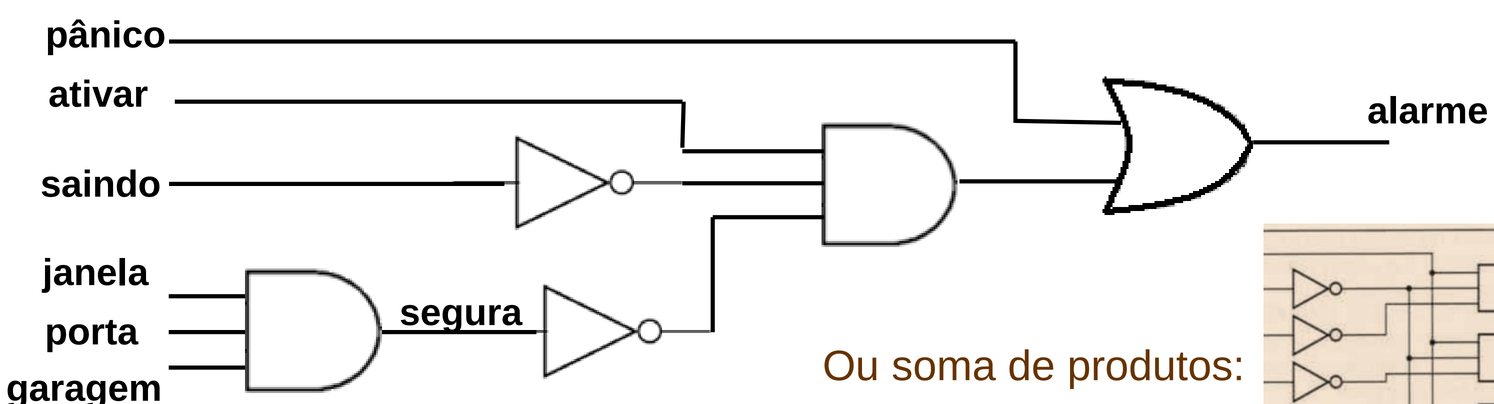


Ex: sistema de alarme

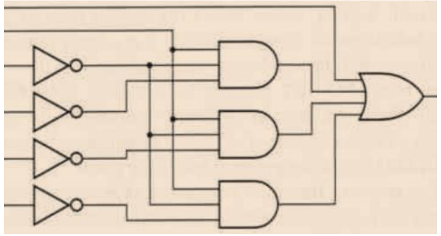
- “O **alarme** é ativado (saída = 1) se a entrada de **pânico** for 1, ou se a entrada **ativar** for 1 e **saindo** for 0 e se a casa não estiver **segura.** A casa está segura se as entradas **janela**, **porta** e **garagem** forem todas 1”

Ex: sistema de alarme

- “O **alarme** é ativado (saída = 1) se a entrada de **pânico** for 1, ou se a entrada **ativar** for 1 e **saindo** for 0 e se a casa não estiver **segura.** A casa está segura se as entradas **janela**, **porta** e **garagem** forem todas 1”
  - alarme = panico + ativar • saindo’ • segura’
  - segura = janela • porta • garagem
  - ➔ alarme = panico + ativar • saindo’ • (janela • porta • garagem)’

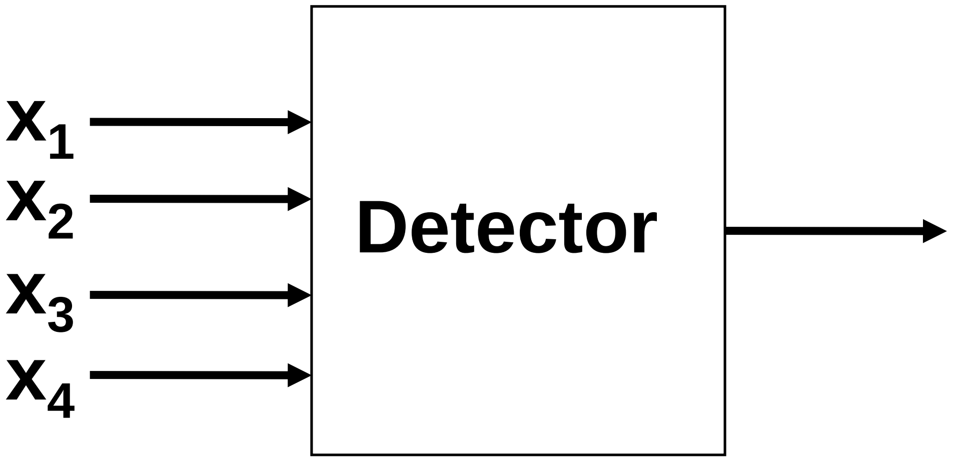


Ou soma de produtos:



Minimização de circuitos combinatórios

- Primeira estrutura encontrada nem sem é ótima
  - Em especial, **somas de produtos** e **produtos de somas** são bastante caros: complexidade exponencial com #entradas
- Exemplo: Sintetizar um circuito de chaveamento para detectar números ímpares de 0 a 9, representados com 4 bits.

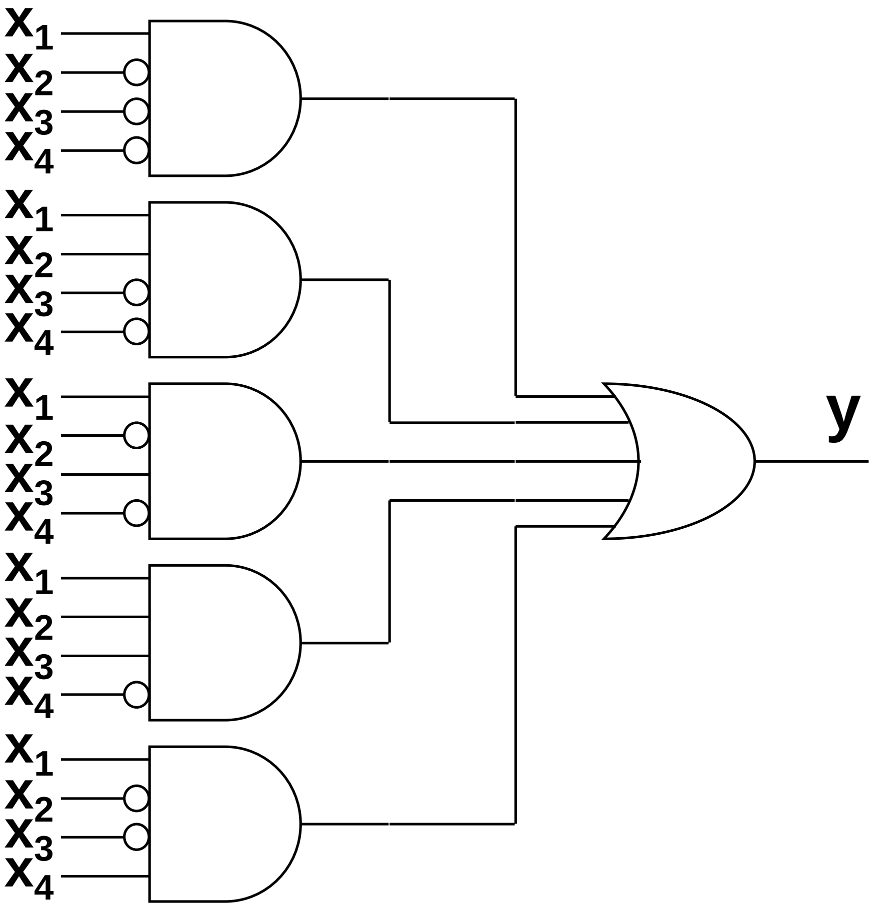


Ex. Detector ímpares
Tabela Verdade

x4 x3 x2 x1	y	Soma Produtos	Produto de Somas
0 0 0 0	0		x4 + x3 + x2 + x1
0 0 0 1	1	x4’ • x3’ • x2’ • x1	
0 0 1 0	0		x4 + x3 + x2’ + x1
0 0 1 1	1	x4’ • x3’ • x2 • x1	
0 1 0 0	0		x4 + x3’ + x2 + x1
0 1 0 1	1	x4’ • x3 • x2’ • x1	
0 1 1 0	0		x4 + x3’ + x2’ + x1
0 1 1 1	1	x4’ • x3 • x2 • x1	
1 0 0 0	0		x4’ + x3 + x2 + x1
1 0 0 1	1	x4 • x3’ • x2’ • x1	

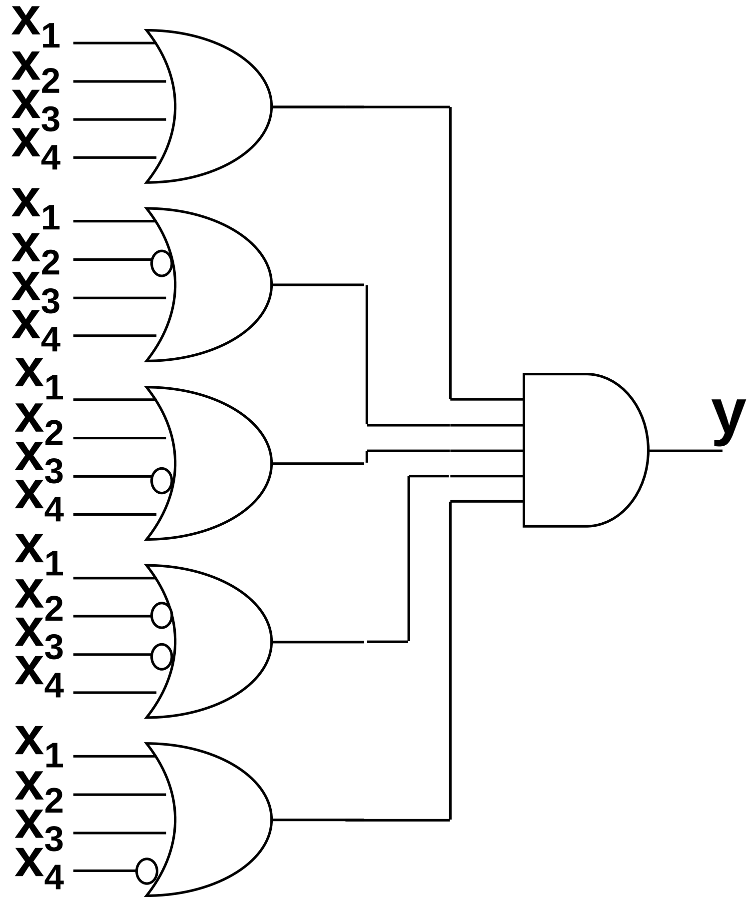
Ex. Detector ímpares
Soma de Produtos

$$y = x_4'\bullet x_3'\bullet x_2'\bullet x_1 +$$
$$x_4'\bullet x_3'\bullet x_2\bullet x_1 +$$
$$x_4'\bullet x_3\bullet x_2'\bullet x_1 +$$
$$x_4'\bullet x_3\bullet x_2\bullet x_1 +$$
$$x_4\bullet x_3'\bullet x_2'\bullet x_1$$



Ex. Detector ímpares
Produto de Somas

$$y = (x_4+x_3+x_2+x_1) \bullet (x_4+x_3+x_2'+x_1) \bullet (x_4+x_3'+x_2+x_1) \bullet (x_4+x_3'+x_2'+x_1) \bullet (x_4'+x_3+x_2+x_1)$$

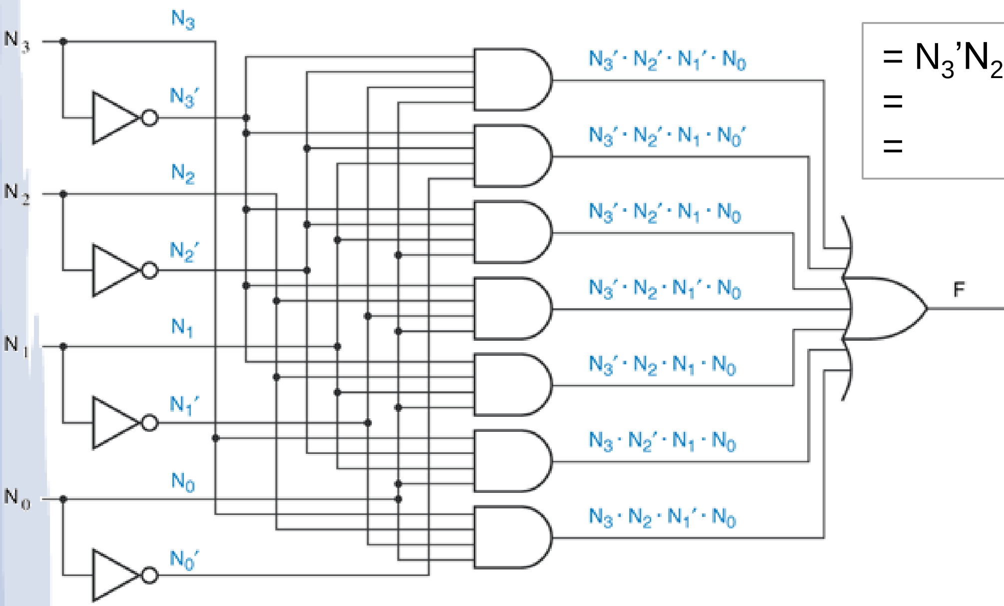


Er... Mas não era mais fácil fazer y = x1...?

Minimização de circuitos combinatórios

- Objetivo: Obter solução mais econômica!
- Critérios possíveis:
  - Minimização do número de literais da função de chaveamento.
  - Minimização do número de interconexões entre as portas.
  - Minimização do número de pinos do circuito integrado a ser eventualmente construído.
- Maioria das estratégias baseada em T10 (combinação)
  - (T10) $X \bullet Y + X \bullet Y' = X$ (T10') $(X+Y) \bullet (X+Y') = X$
  - Ex.: T10 no circuito detector de primos

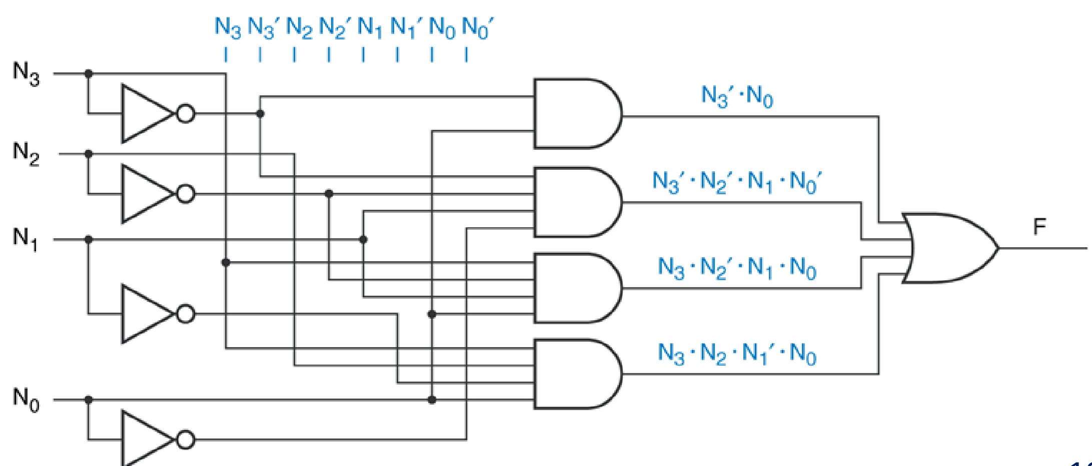
Ex.: T10 no circuito detector de primos



$= N_3'N_2'N_1'N_0 + N_3'N_2'N_1N_0 + N_3'N_2N_1'N_0 + N_3'N_2N_1N_0 + \ldots$

$= N_3'N_2'N_0 + N_3'N_2N_0 + \ldots$

$= N_3'N_0 + \ldots$



Vamos discutir um método mais amigável para minimização

Mapas de Karnaugh

- Representação gráfica da Tabela Verdade de função lógica
  - Nota: funciona para 5+ variáveis, mas método fica pouco prático nesse caso (não será discutido aqui)

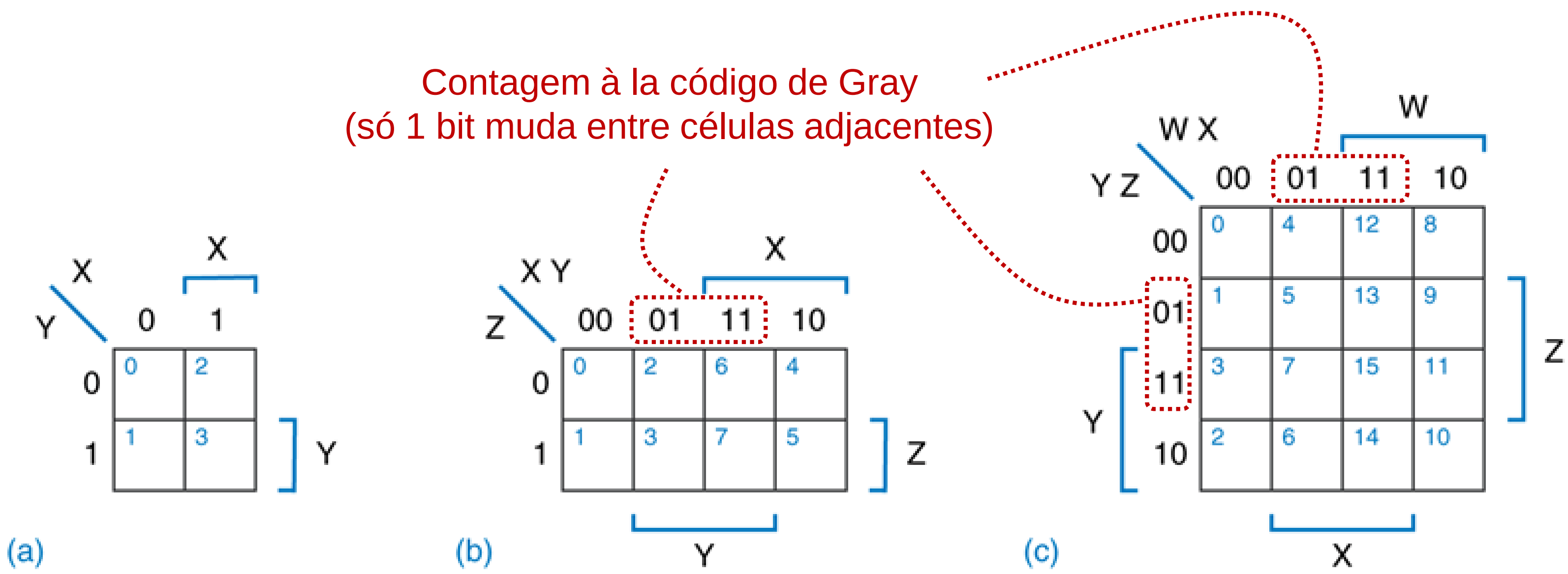


Figure 4-26

Karnaugh maps: (a) 2-variable; (b) 3-variable; (c) 4-variable.

Mapas de Karnaugh

- Para representar função lógica, células são preenchidas com 0 ou 1: saída para a entrada (mintermo) vinculada àquela célula
  - Nota: números nas células normalmente não são desenhados, mas apenas inferidos pelos bits das entradas (e.g., WXYZ=1101 ➔ **13**)

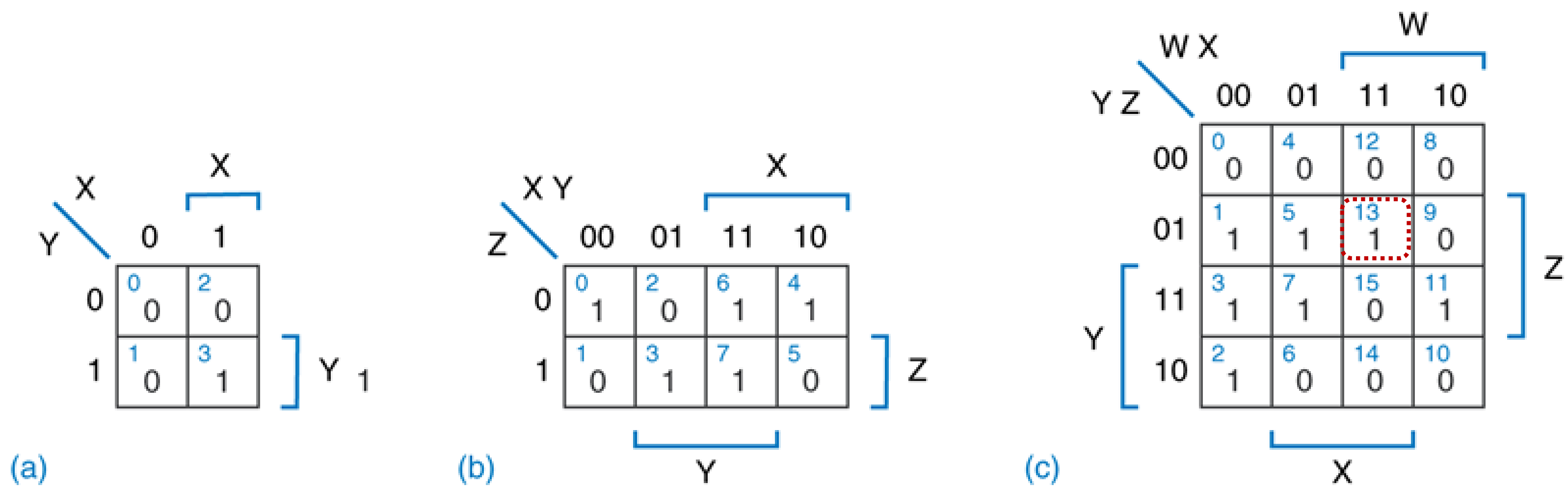
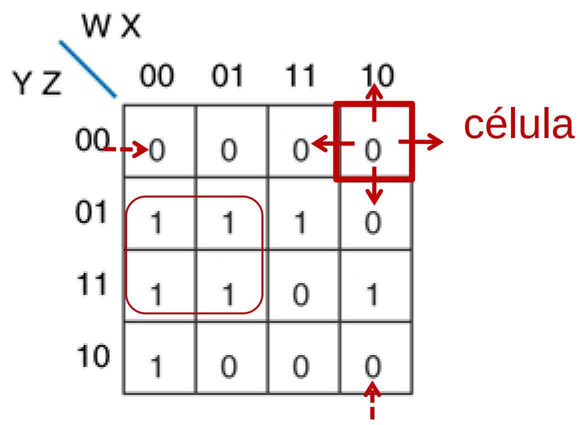


Figure 4-27

Karnaugh map for logic functions: (a) $F = \Sigma_{X,Y}(3)$;
(b) $F = \Sigma_{X,Y,Z}(0,3,4,6,7)$; (c) $F = \Sigma_{W,X,Y,Z}(1,2,3,5,7,11,13)$.

Mapas de Karnaugh: definições

- **Célula**: é um mintermo ou um maxtermo.
- **Células adjacentes**: Duas células são adjacentes quando diferem apenas no valor de uma variável.
  - Mapa “circular”: adjacências entre bordas também!
- **Adjacências**: grupamentos (retangulares ou quadrados) de 2^n células adjacentes.
  - Também denominados “Cubos-n”, para $n \geq 0$



Mapa de Karnaugh e Mintermos

- Grupamentos de 1’s (mintermos): aplicação gráfica de T10!
  - Cada grupamento refere-se ao conjunto de variáveis que não mudam. Ex.: grupamento(7, 5) = X•Y•Z + X•Y’•Z = X•Z ← T10
    - Quanto maior o grupamento, menor o número de variáveis
  - Função lógica equivale a OU lógico entre todos os grupamentos

(a)

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

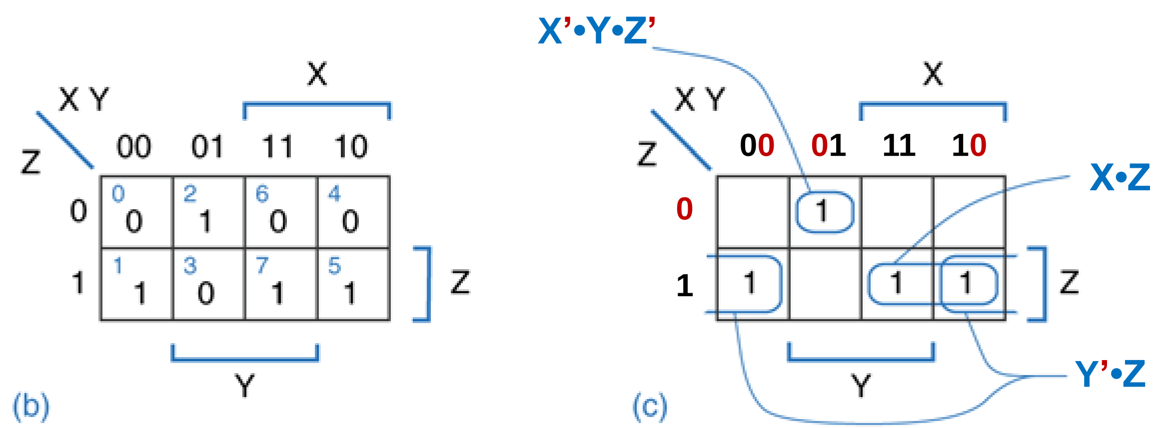


Figure 4-28

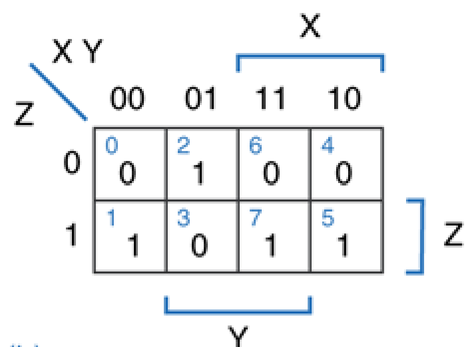
$F = \Sigma_{X,Y,Z}(1,2,5,7)$: (a) truth table; (b) Karnaugh map; (c) combining adjacent 1-cells.

Mapa de Karnaugh e Mintermos

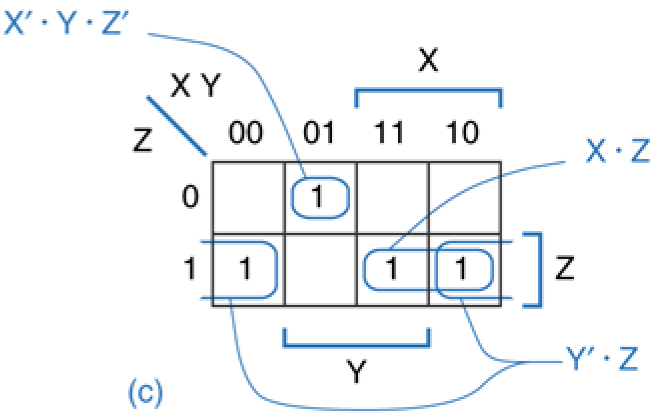
(a)

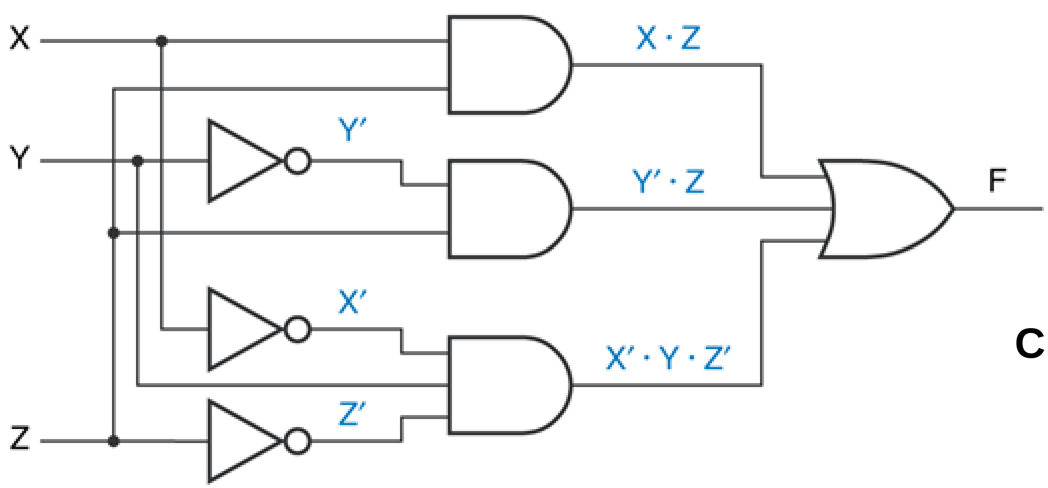
X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

(b)



(c)

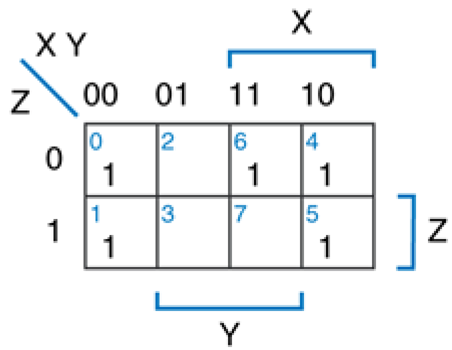


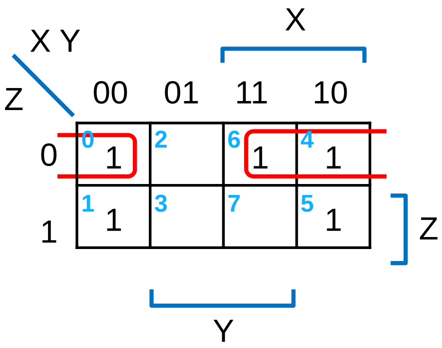


Circuito AND-OR mínimo

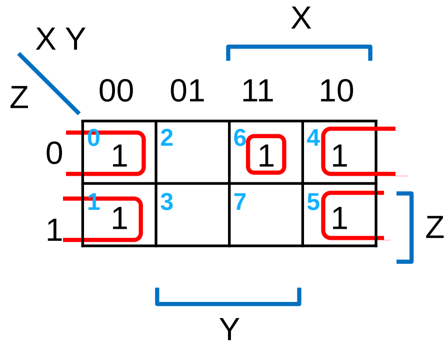
Outro exemplo

(a)



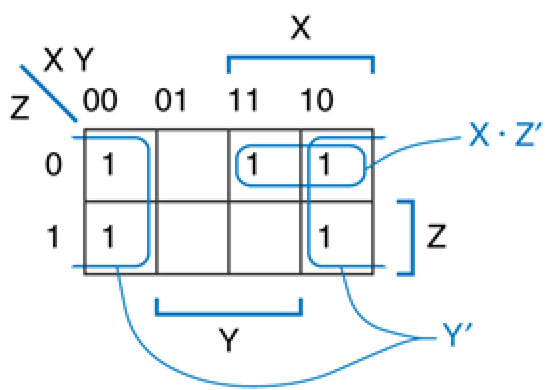


Grupamento inválido (não tem 2^n células)

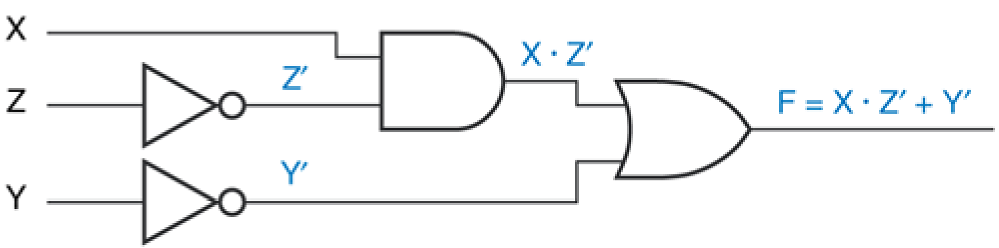


Grupamentos sub-ótimos

(b)



(c)



Exemplo: detector de números primos

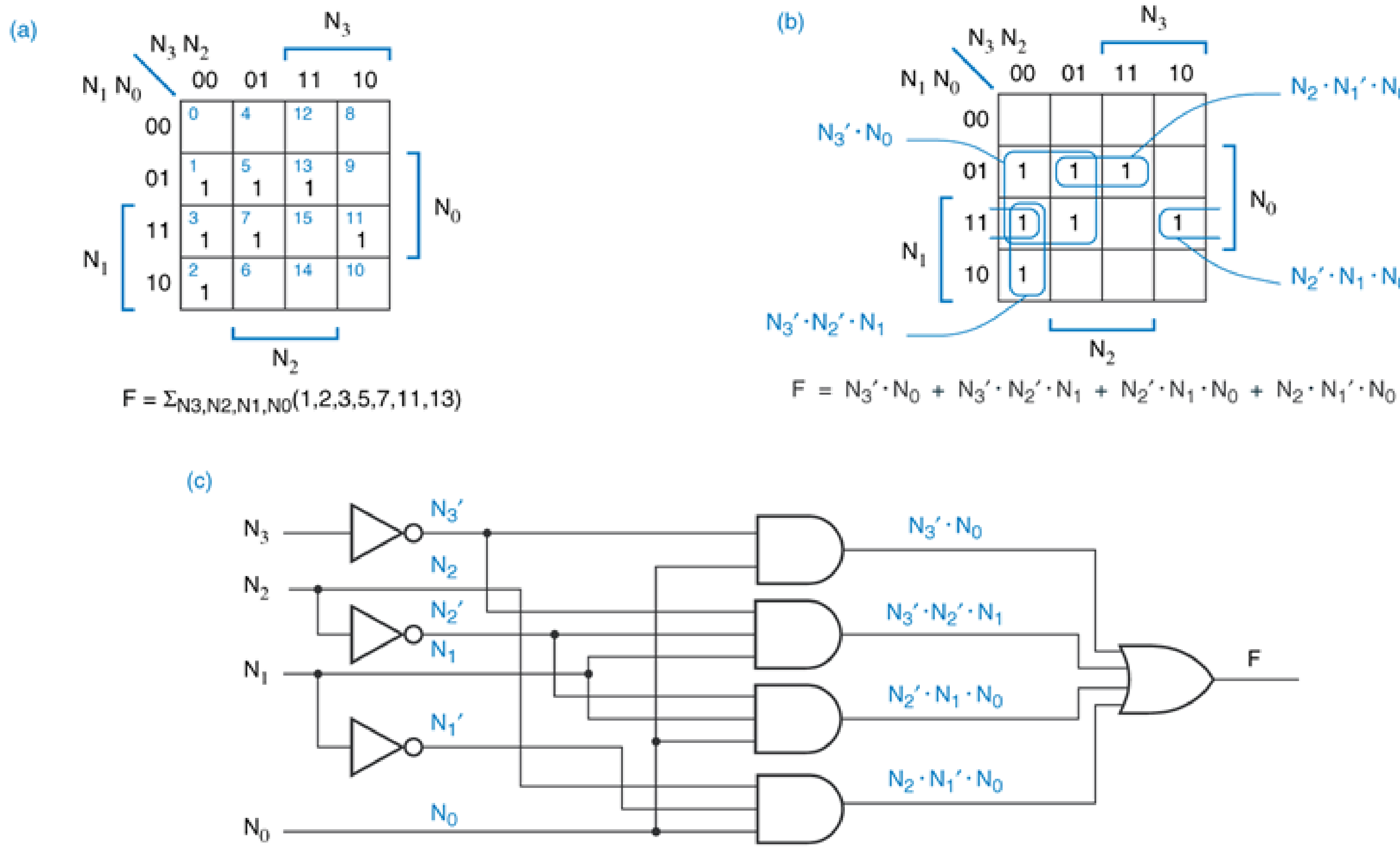


Figure 4-31

Prime-number detector: (a) initial Karnaugh map; (b) circled product terms; (c) minimized circuit.

Exercício 1

- Determinar o menor conjunto de adjacências que cubra (contenha) todos os mintermos, e escrever a soma de produtos correspondente

x_1 \ x_3x_2	00	01	11	10
0	0	1	1	0
1	1	0	0	1

x_1 \ x_3x_2	00	01	11	10
0	0	1	1	1
1	0	1	1	0

x_1 \ x_3x_2	00	01	11	10
0	1	0	0	1
1	1	0	0	1

Exercício 2

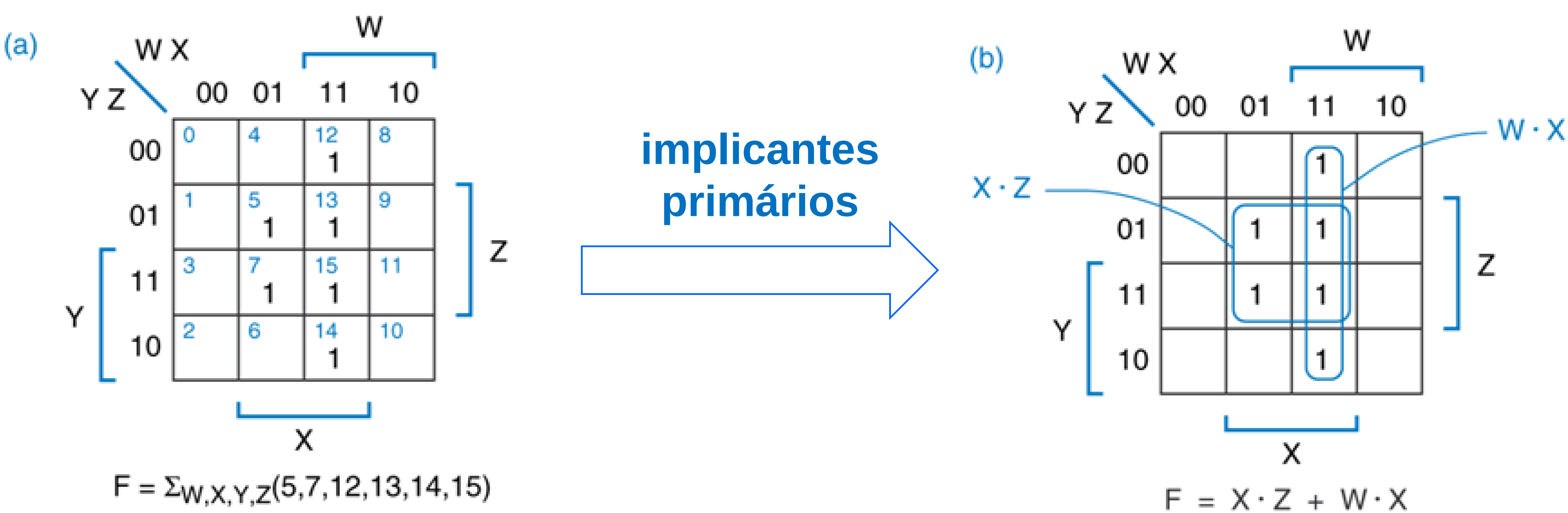
- Determinar o menor conjunto de adjacências que cubra (contenha) todos os mintermos, e escrever a soma de produtos correspondente

x_2x_1 \ x_4x_3	00	01	11	10
00	1	0	0	1
01	1	1	0	1
11	0	1	0	0
10	1	0	0	1

x_2x_1 \ x_4x_3	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	0	1
10	0	0	0	1

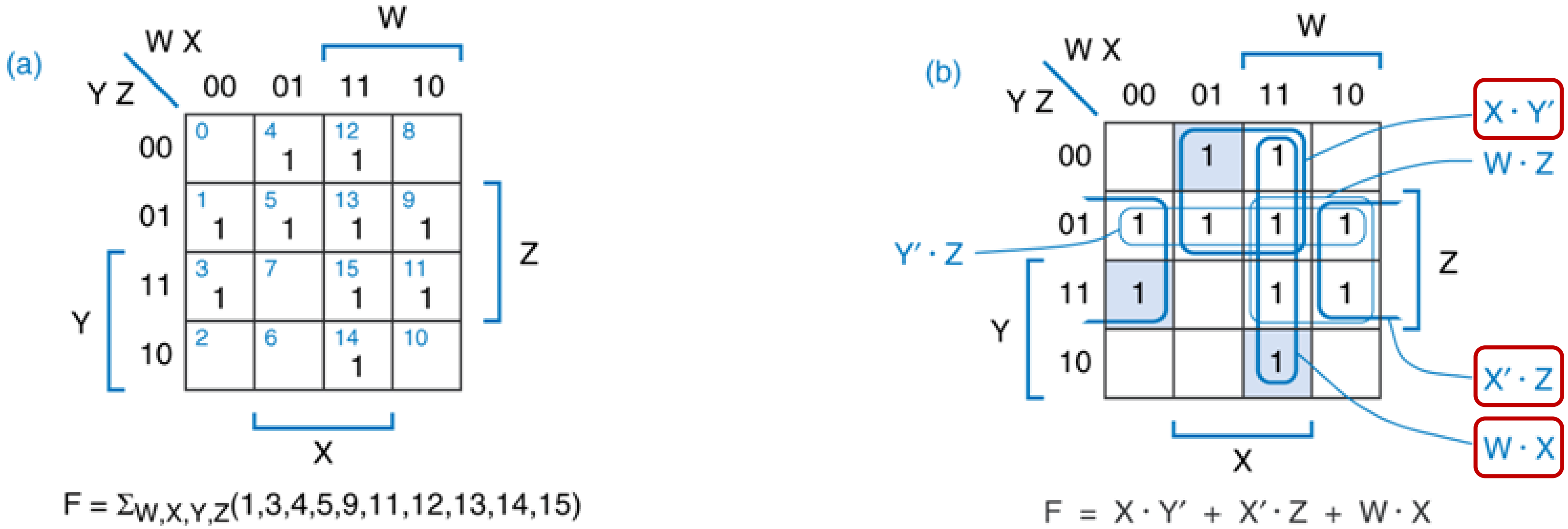
Mais minimização: Tabela de Cobertura

- **Implicante primário** (IP) em um Mapa de Karnaugh: agrupamento de células com tamanho máximo
  - Ou seja, se tentarmos aumentar sua cobertura (dobrar seu tamanho em alguma direção), ele passa a cobrir um ou mais 0s
- **Soma mínima**: é <u>**uma**</u> **soma de IPs**



Mais minimização: Tabela de Cobertura

- **Mas:** a soma de **<u>todos</u> os IPs** (chamada soma completa) nem sempre leva a um circuito mínimo

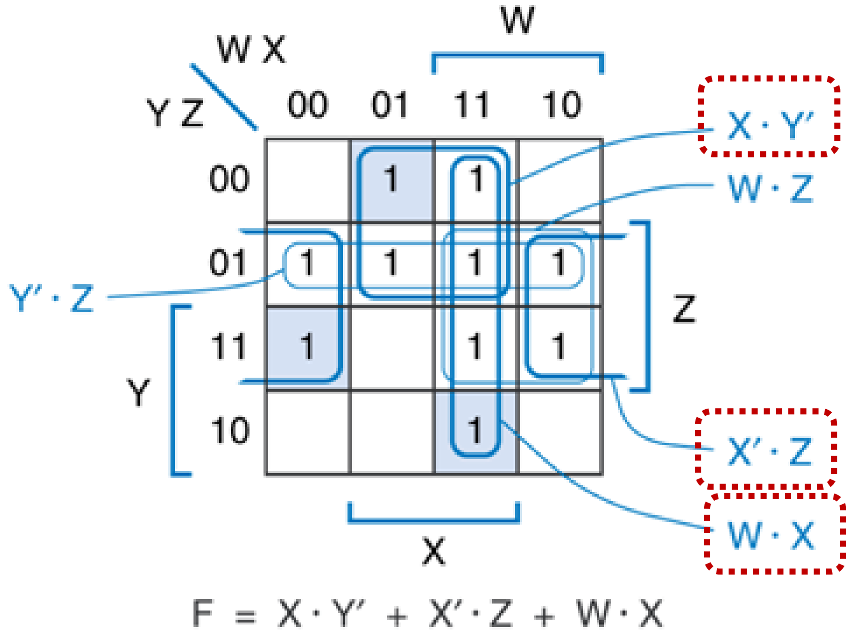


- 5 IPs, mas soma mínima requer apenas 3 deles: X•Y' + X'•Z + W•X já cobrem todos os 1s

➔ Como determinar quais IPs incluir e quais deixar de fora para minimizar circuito?

Mais minimização: Tabela de Cobertura

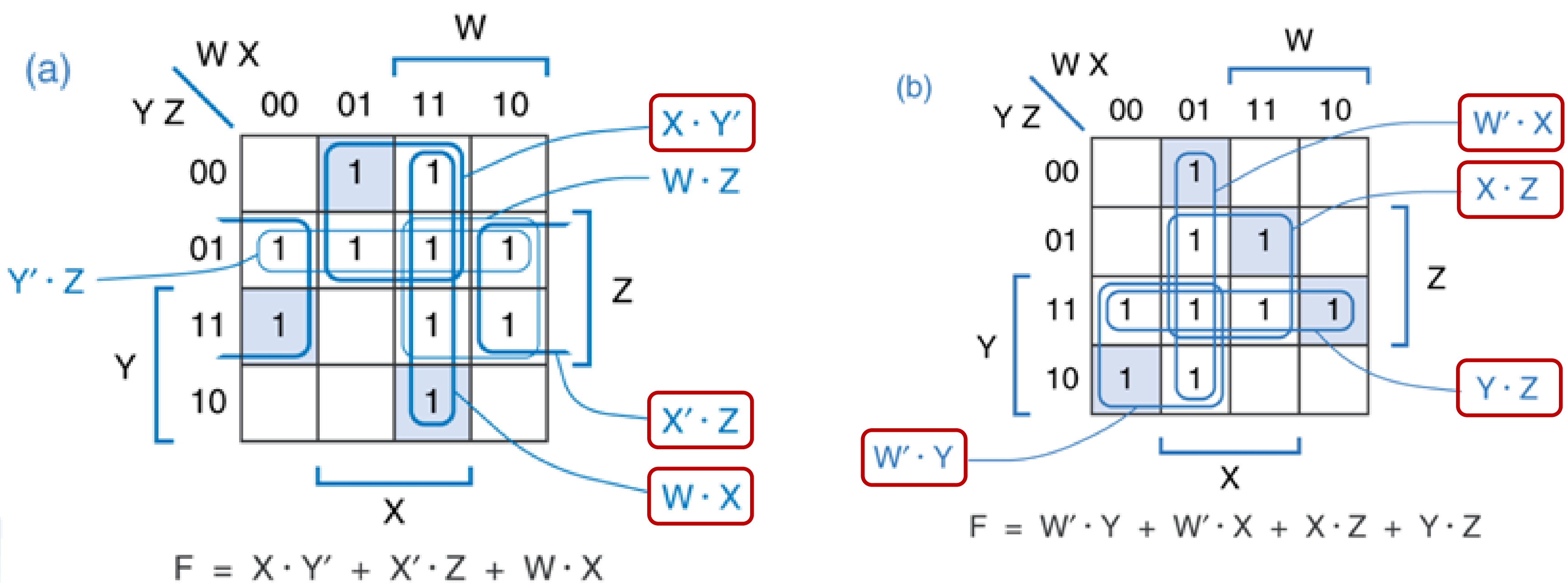
- **1s isolados** (cobertos por apenas um IP) definem os **implicantes primários essenciais**
  - Estes devem necessariamente ser inclusos na equação mínima



F = $\Sigma_{W,X,Y,Z}$(1,3,4,5,9,11,12,13,14,15): (a) Karnaugh map; (b) prime implicants and distinguished 1-cells.

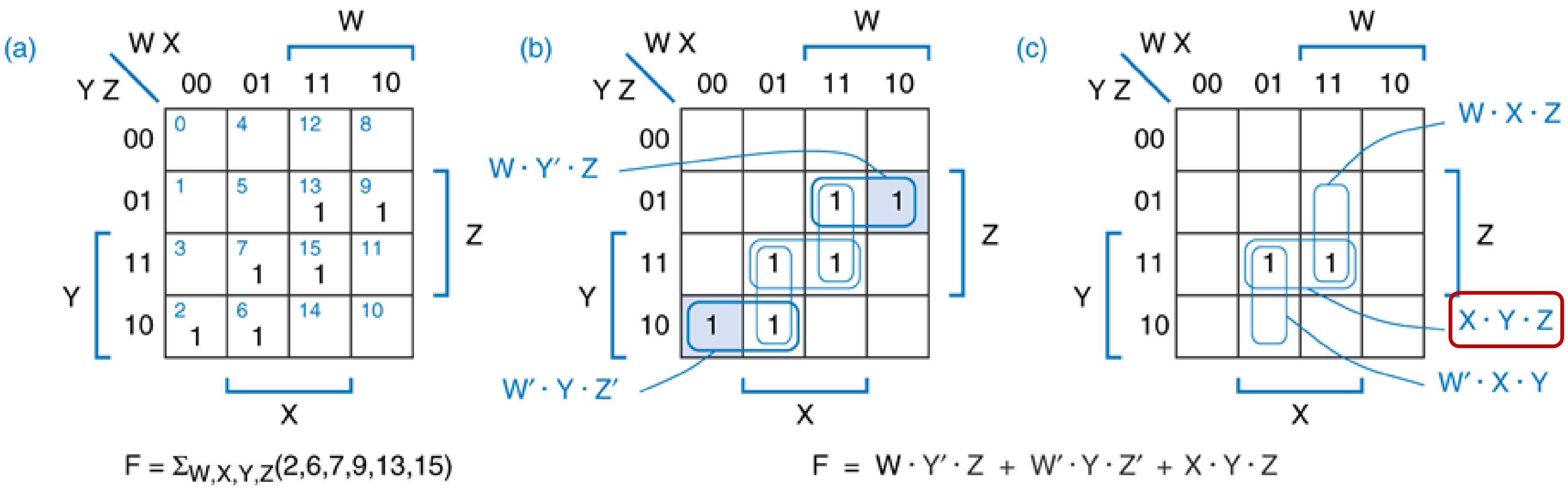
Mais minimização: Tabela de Cobertura

- Se IPs essenciais já **cobrem todos os 1s**: nenhum outro IP precisa ser incluso no circuito mínimo
  - Exemplos:



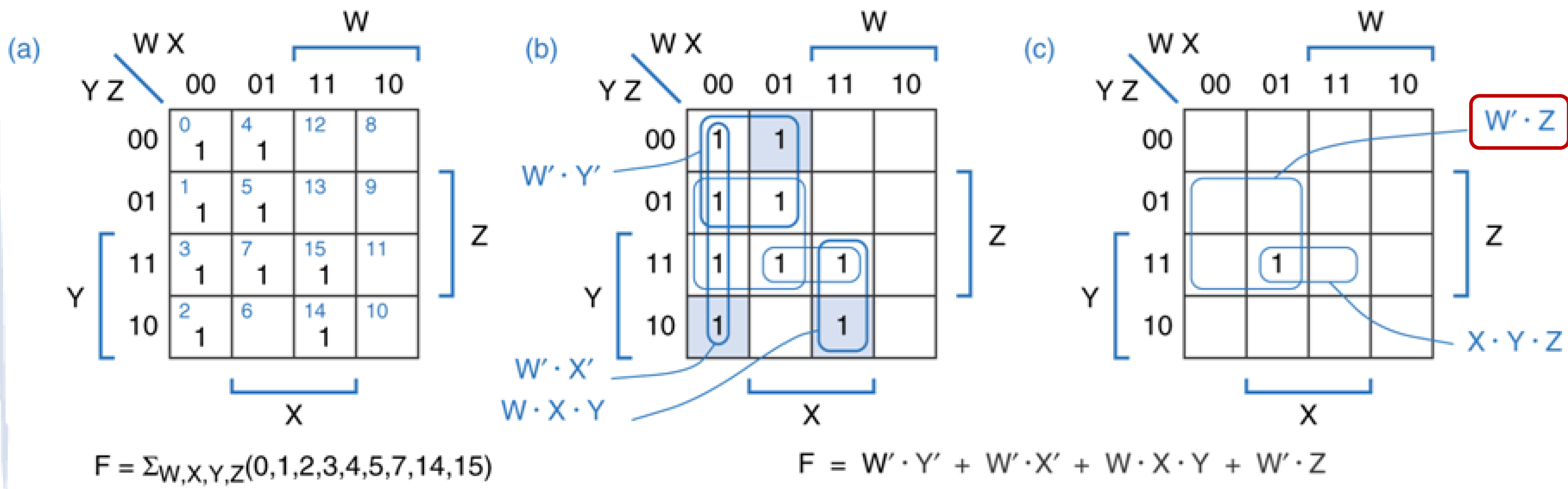
Mais minimização: Tabela de Cobertura

- Se IPs essenciais não **cobrem todos os 1s**:
  - Após remoção dos IPs essenciais: incluir **menor número** de IPs com **maior cobertura** que cubram todos os 1s restantes
  - Exemplo:



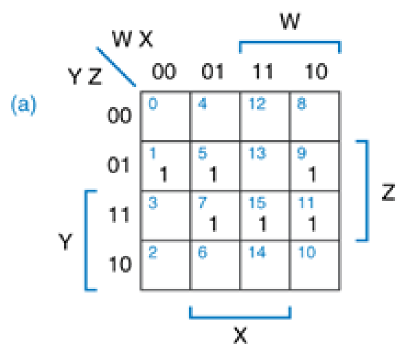
Mais minimização: Tabela de Cobertura

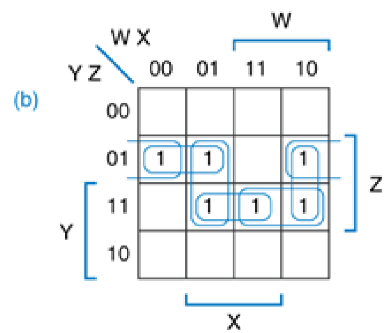
- Se IPs essenciais não **cobrem todos os 1s**:
  - Após remoção dos IPs essenciais: incluir **menor número** de IPs com **maior cobertura** que cubram todos os 1s restantes
  - Exemplo:



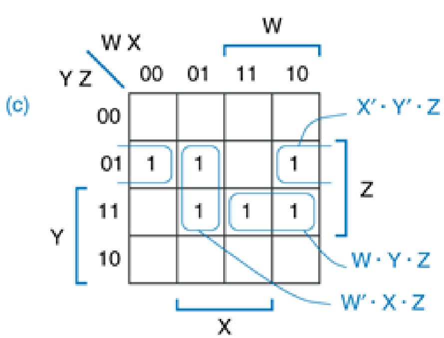
Mais minimização: Tabela de Cobertura

- Há casos mais complexos: sem IPs essenciais
  - Remoção de IPs arbitrariamente escolhidos como "essenciais", até obter cobertura completa (processo de tentativa e erro)
  - Exemplo:



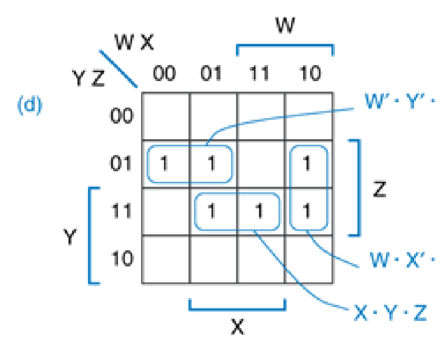


Sem IPs essenciais



$F = W' \cdot X \cdot Z + W \cdot Y \cdot Z + X' \cdot Y' \cdot Z$

Duas possibilidades



$F = X \cdot Y \cdot Z + W \cdot X' \cdot Z + W' \cdot Y' \cdot Z$

Exercícios

Exercício

- Sintetize um circuito para a função “F = maioria dentre 3 variáveis X, Y e Z”. Minimize o circuito obtido

Exercício

- Sintetize um circuito para a função “F = maioria dentre 3 variáveis X, Y e Z”. Minimize o circuito obtido

X	Y	Z	F
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

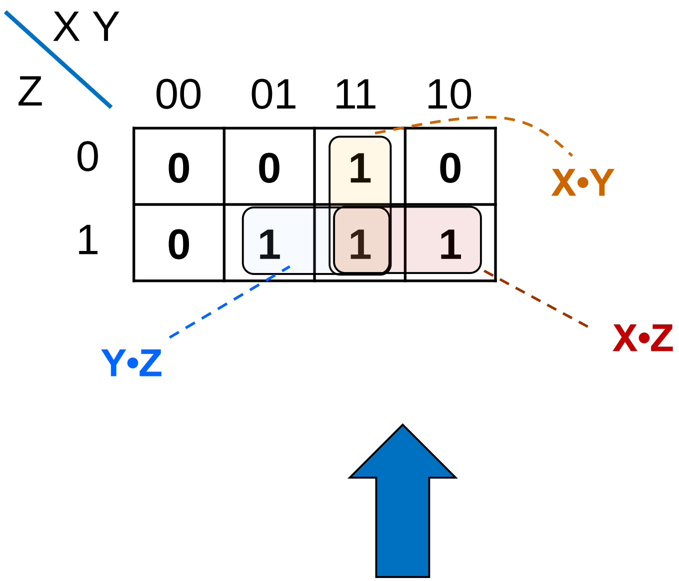
$$
\begin{aligned}
& (X'\cdot Y\cdot Z)+(X\cdot Y'\cdot Z)+(X\cdot Y\cdot Z')+(X\cdot Y\cdot Z) \\
&= X\cdot Y\cdot (Z+Z')+X'\cdot Y\cdot Z+X\cdot Y'\cdot Z \\
&= X\cdot Y+X'\cdot Y\cdot Z+X\cdot Y'\cdot Z \\
&= X\cdot (Y+Y'\cdot Z)+X'\cdot Y\cdot Z \\
&= X\cdot (Y+Z) + X'\cdot Y\cdot Z \\
&= X\cdot Y+X\cdot Z + X'\cdot Y\cdot Z \\
&= X\cdot Y+Z\cdot (X + X'\cdot Y) \\
&= X\cdot Y+Z\cdot (X+Y) \\
&= X\cdot Y+Z\cdot X+Z\cdot Y
\end{aligned}
$$

Minimização via manipulação algébrica dos mintermos

Exercício

- Sintetize um circuito para a função “F = maioria dentre 3 variáveis X, Y e Z”. Minimize o circuito obtido

X	Y	Z	F
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



Minimização via Mapa de Karnaugh