PCS 3115

Sistemas Digitais I

Mapas de Karnaugh

Prof. Dr. Marcos A. Simplicio Jr.

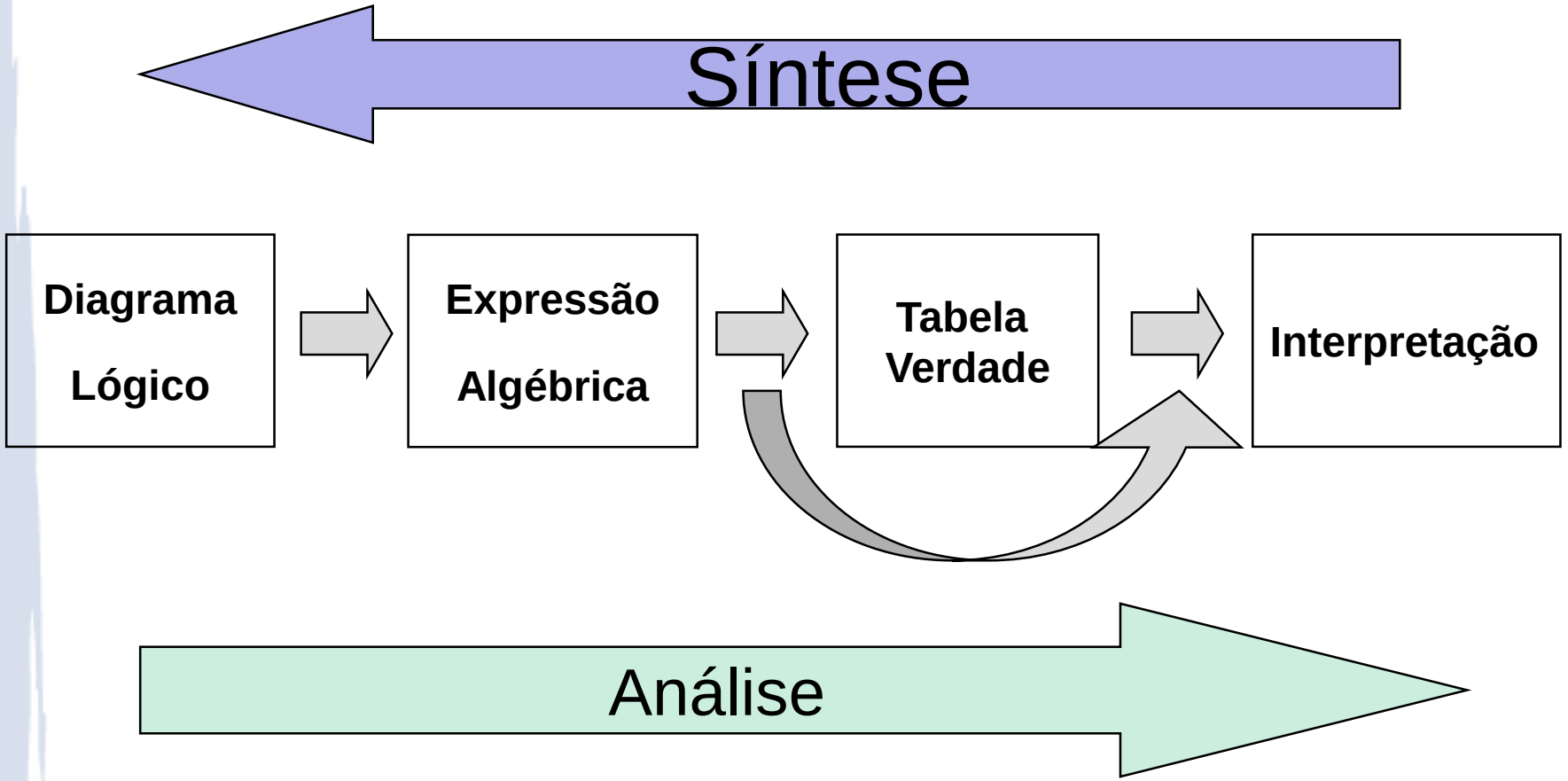
Adaptado por Glauber De Bona (2018)

Objetivos da aula

- Minimização de circuitos digitais combinatórios.
- Mapas de Karnaugh: Conceito, construção e utilização de mapas até 4 variáveis.

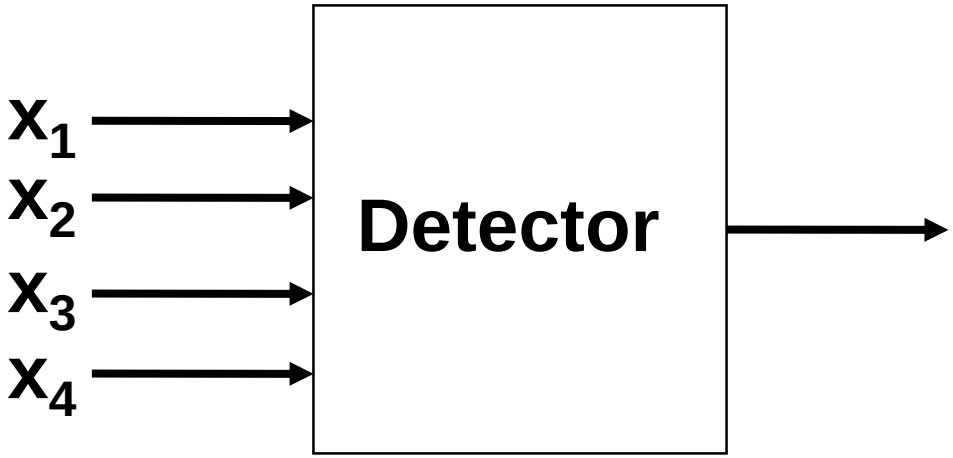
- Referência: Wakerly, Seção 4.3.5

Síntese e Análise



Síntese partindo da tabela verdade

- Encontramos uma expressão Boolena (soma canônica ou produto canônico) e projetamos o diagrama lógico correspondente.
- Exemplo: Sintetizar um circuito de chaveamento para detectar dígitos (de 0 a 9) ímpares codificados em binário

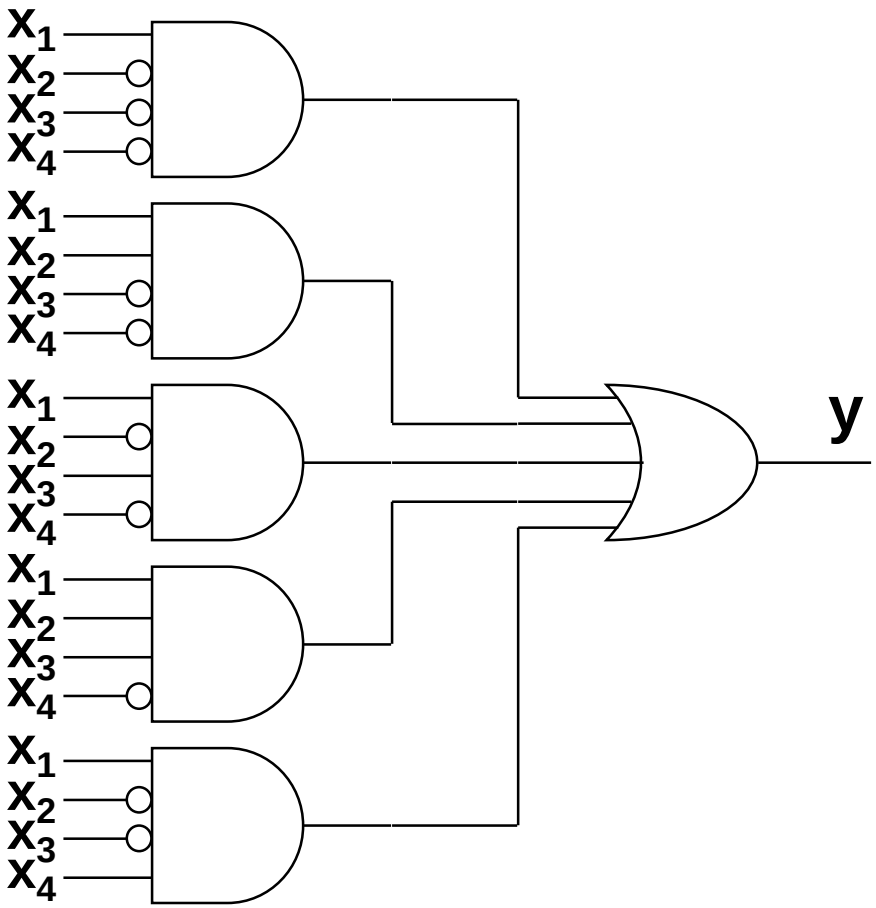


Ex. Detector ímpares
Tabela Verdade

x4 x3 x2 x1	y	Min-termos	Max-termos
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	

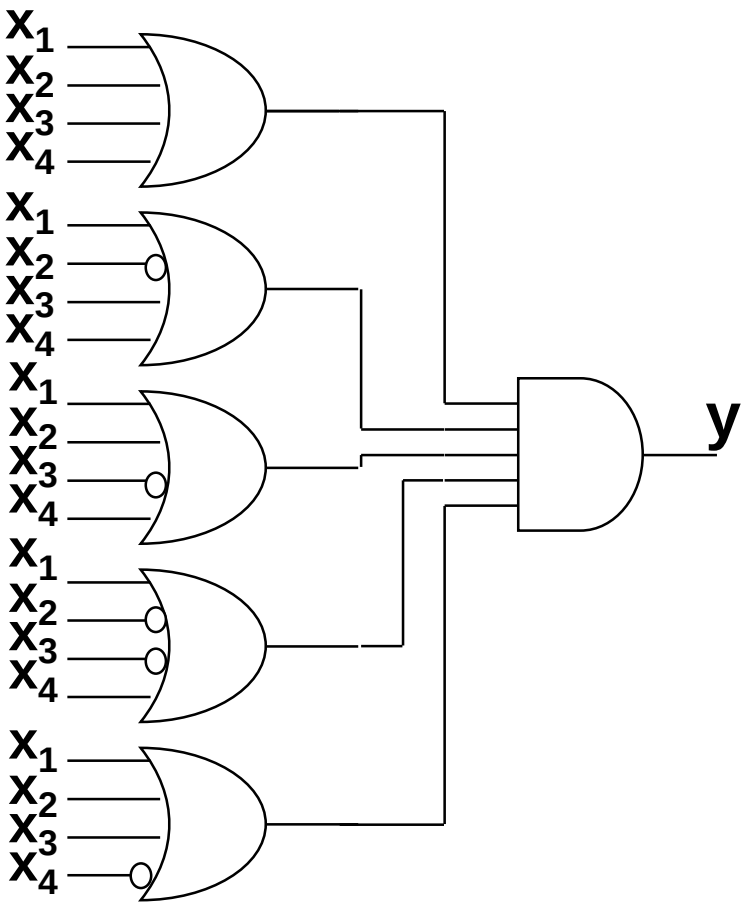
Ex. Detector ímpares
Soma Canônica

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



Ex. Detector ímpares
Produto Canônico

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



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

Minimização de circuitos combinatórios

- Primeira expressão Booleana encontrada nem sempre corresponde a um circuito mínimo.
- Soma e produto canônicos usualmente requerem um número exponencial portas, em relação ao número de entradas.
- Vários critérios possíveis para minimizar:
  - Minimização do número de literais da função de chaveamento.
  - Minimização do número total de portas lógicas.
  - Minimização do número de portas em cascata.
  - Minimização do número de entradas em cada porta.

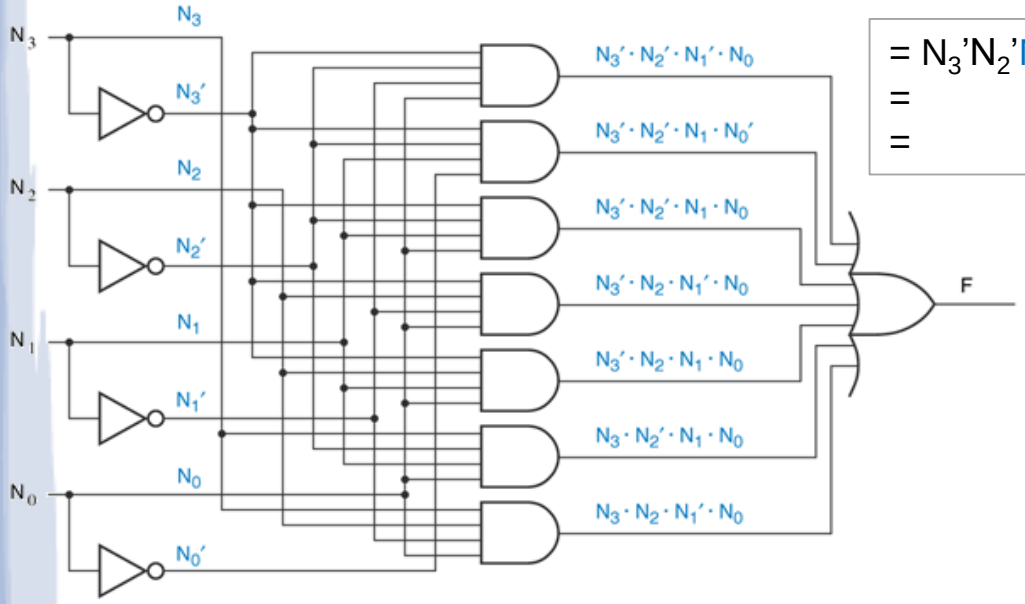
Minimização de circuitos combinatórios

- Impossível minimizar todos os critérios.
- Os métodos que veremos projetam circuitos em dois níveis (AND-OR ou OR-AND) minimizando:
  - Número de portas no primeiro nível
  - Número de entradas em cada porta no primeiro nível
  - Número de entradas em cada porta no segundo nível
- Portas inversoras ocorrem apenas antes do primeiro nível, mas não são minimizadas.
- Na álgebra Booleana: Minimizamos uma soma de produto (de literais) ou um produto da soma (de literais).

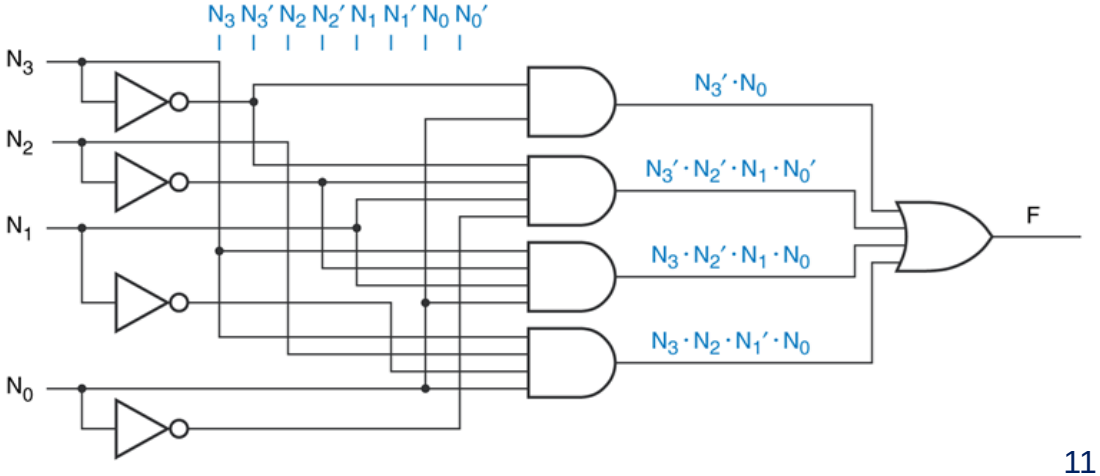
Soma e Produto Minimais

- **Soma minimal** é uma soma de produtos (SdP) tal que:
  - Não há SdPs com menos produtos
  - SdPs com o mesmo número de produtos não possuem menos literais.
- **Produto minimal** é um produto de somas (PdS) tal que:
  - Não há PdSs com menos somas
  - PdSs com o mesmo número de somas não possuem menos literais.
- Podemos usar os teoremas de combinação
  - $X \cdot Y + X \cdot Y' = X$ $\quad\quad$ $(X+Y) \cdot (X+Y') = X$
- Focaremos na **soma minimal**, dada a dualidade

Ex.: 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)

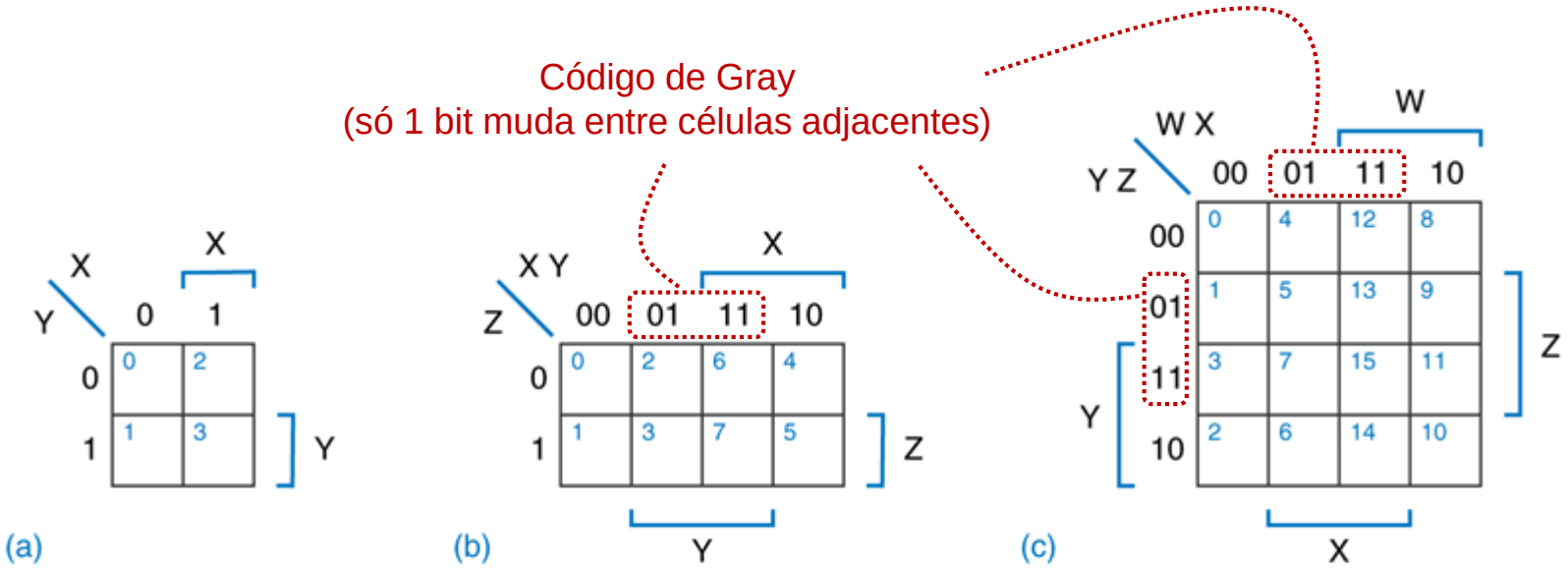


Figure 4-26

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 correspondente àquela célula
  - Nota: números nas células normalmente não são desenhados, mas apenas inferidos pelos bits das entradas (e.g., WXYZ=1101 ➔ **13**)

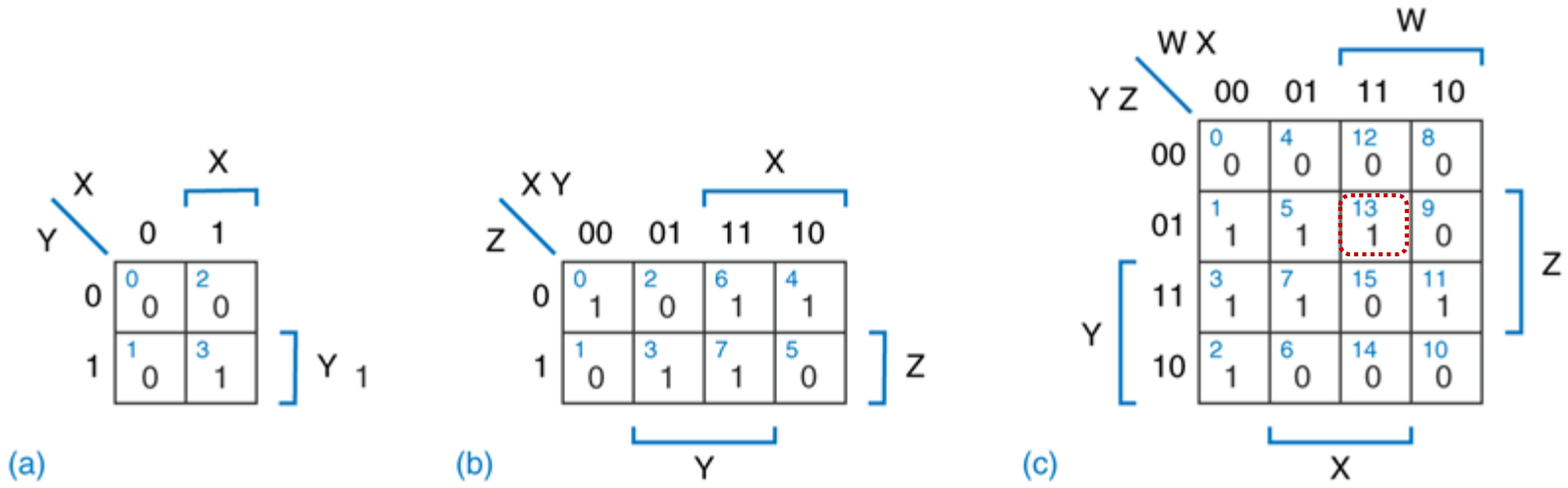
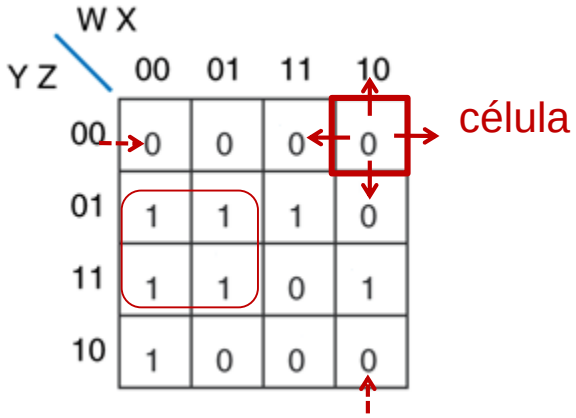


Figure 4-27

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**: linha da tabela verdade, mintermo, 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 ≥ 0

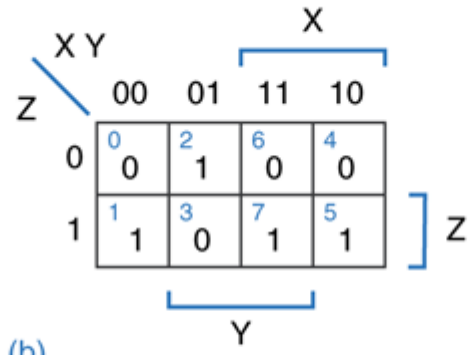


Mapa de Karnaugh e Mintermos

- Grupamentos de 1's (mintermos): aplicação gráfica do teorema da combinação
  - Cada grupamento refere-se ao conjunto de variáveis que não mudam. Ex.: grupamento(7, 5) = X•Y•Z + X•Y'•Z = X•Z
    - Quanto maior o grupamento, menor o número de variáveis
  - Função lógica equivale a OU lógico entre todos os grupamentos
    - Soma minimal: Mínimo de grupamentos que cobrem todos 1's, com os grupamentos de maior tamanho.

(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



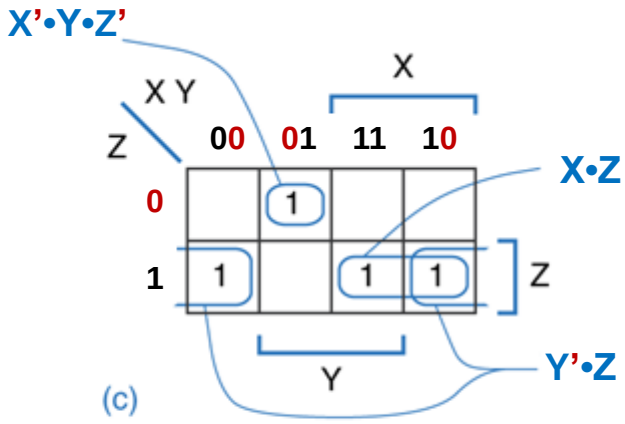


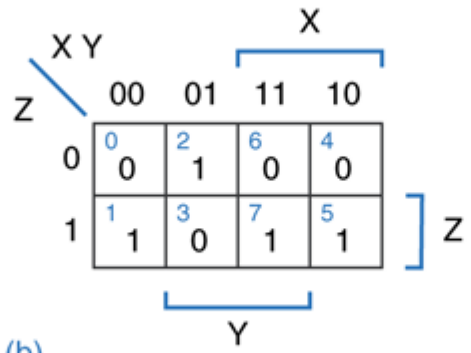
Figure 4-28

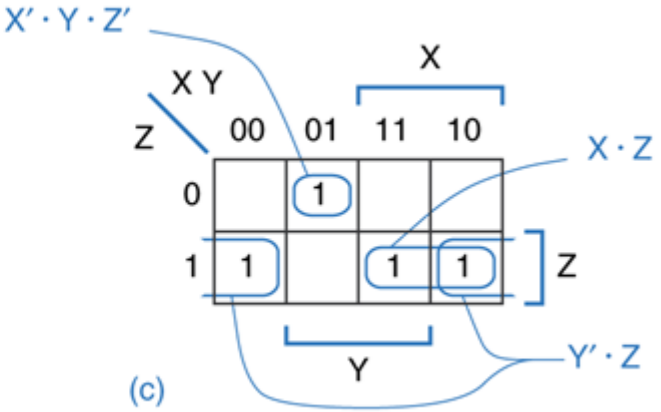
F = $\Sigma_{X,Y,Z}$(1,2,5,7): (a) truth table; (b) Karnaugh map; (c) combining adjacent 1-cells.

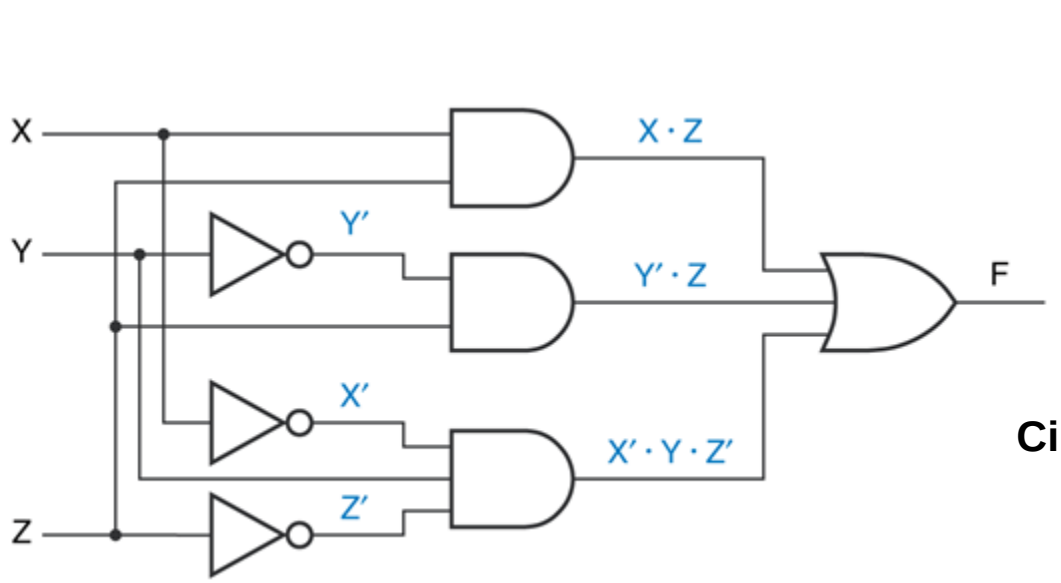
Mapa de Karnaugh e Mintermos

(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



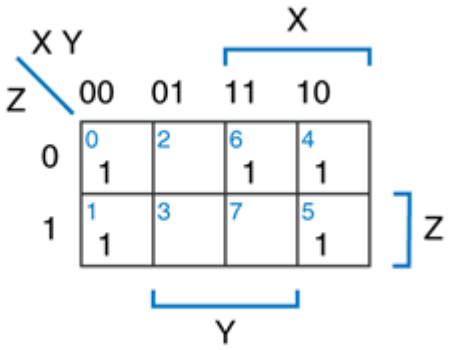


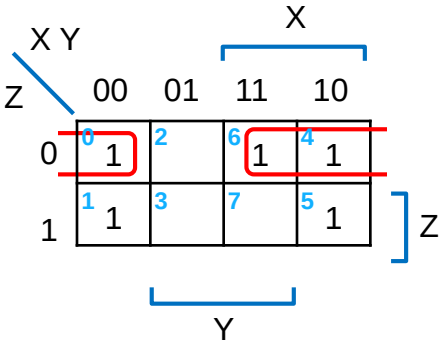


Circuito AND-OR mínimo

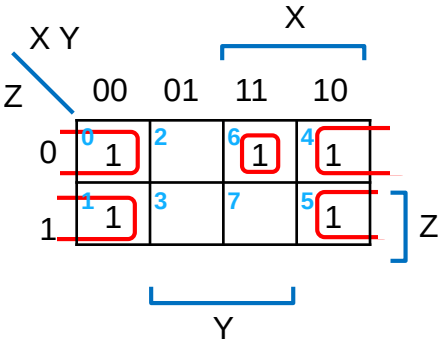
Outro exemplo

(a)



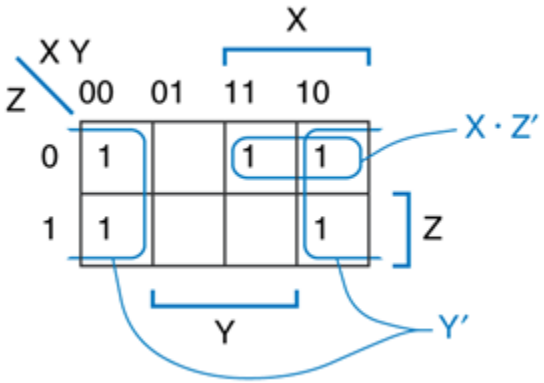


Grupamento inválido (não tem 2^n células)

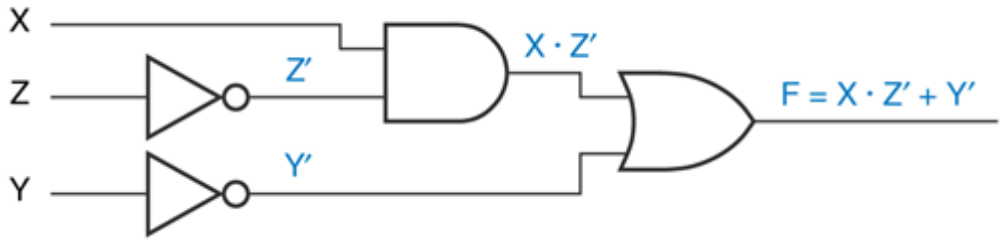


Grupamentos sub-ótimos

(b)



(c)



Exemplo: detector de números primos

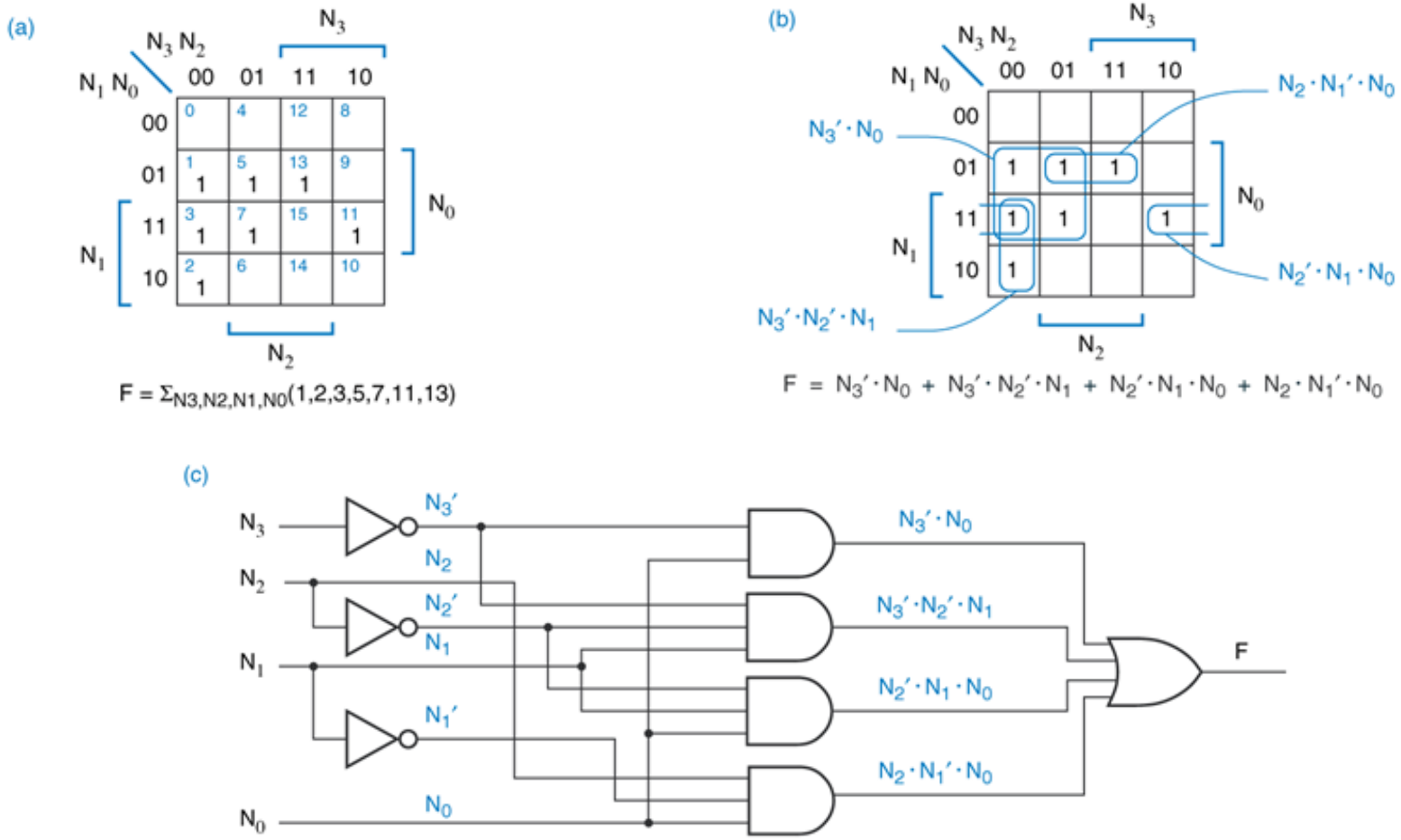


Figure 4-31

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

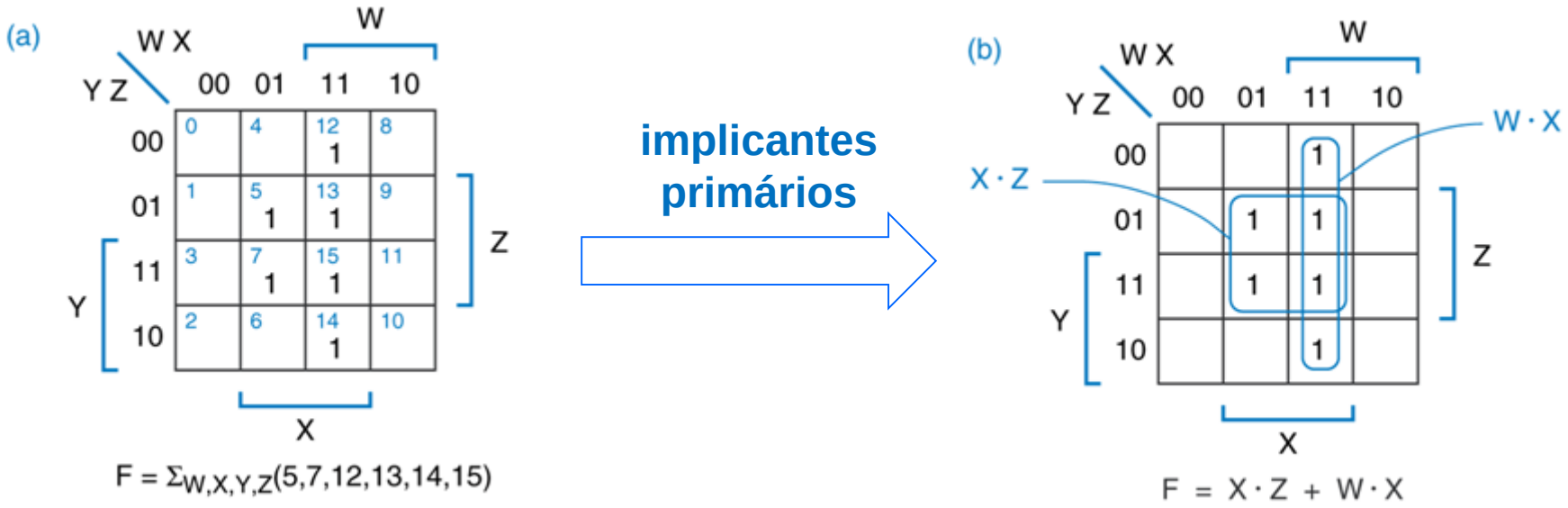
- Determinar o menor conjunto de adjacências que cubra (contenha) todos os mintermos, e escrever a soma de produtos correspondente

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

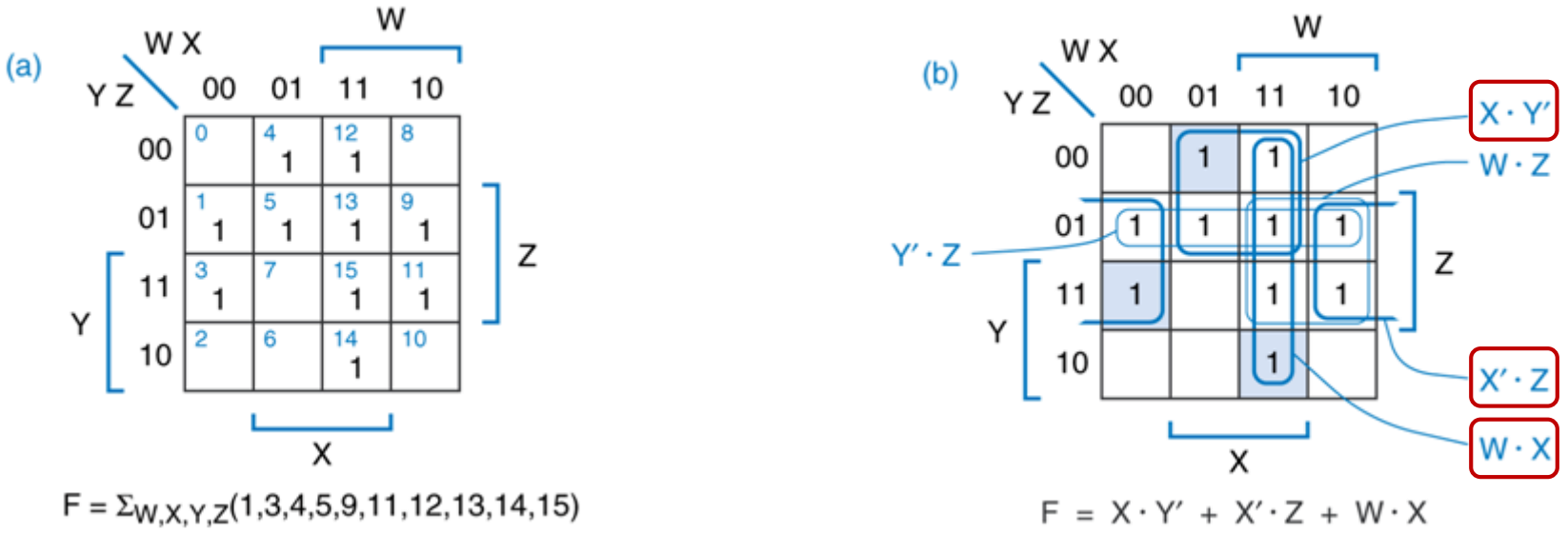
Mais minimização

- **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
- **Toda soma minimal** é **<u>uma</u> soma de IPs**



Mais minimização

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

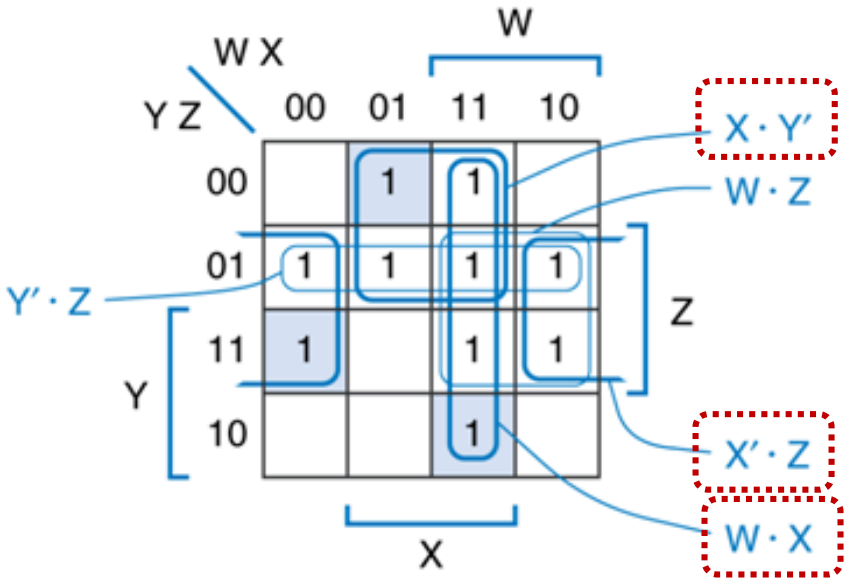


- 5 IPs, mas soma mínima requer apenas 3 deles: X•Y’ + X’•Z + W•X já cobrem todos os 1s

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

Mais minimização

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

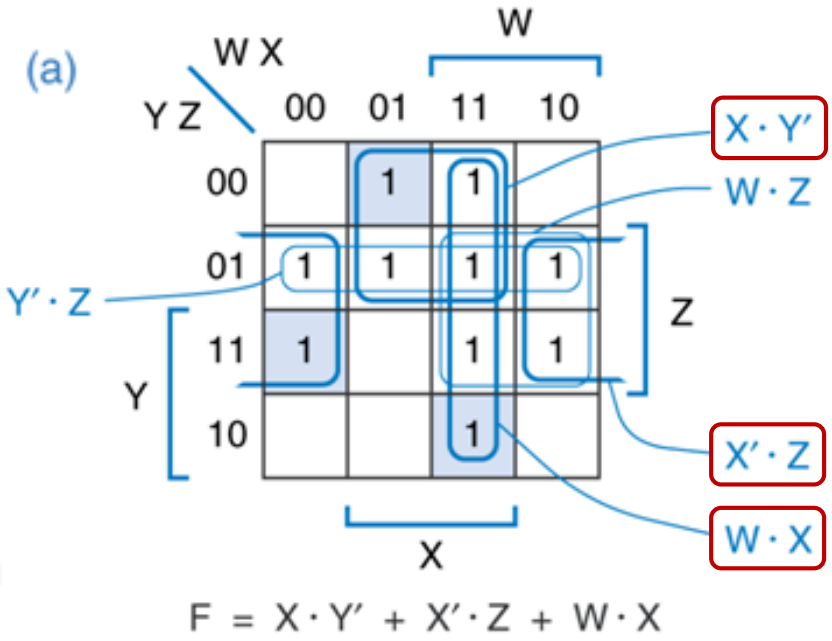


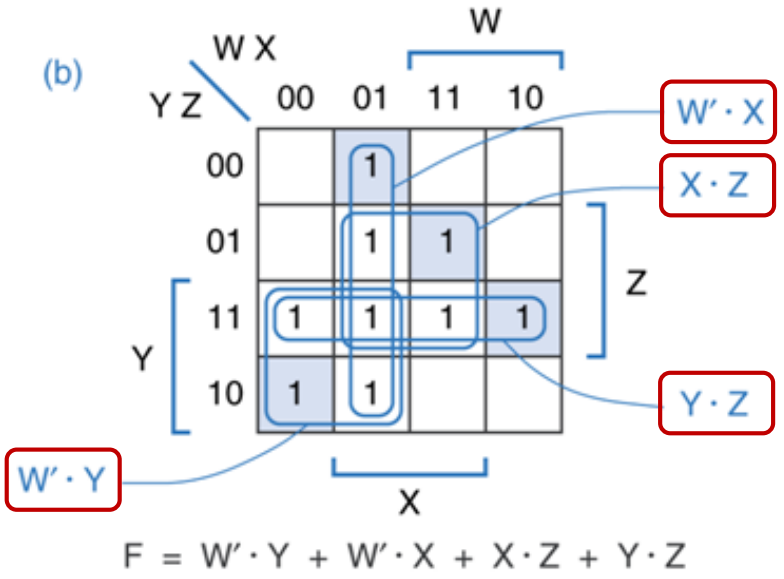
$$F = X \cdot Y' + X' \cdot Z + W \cdot X$$

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

Mais minimização

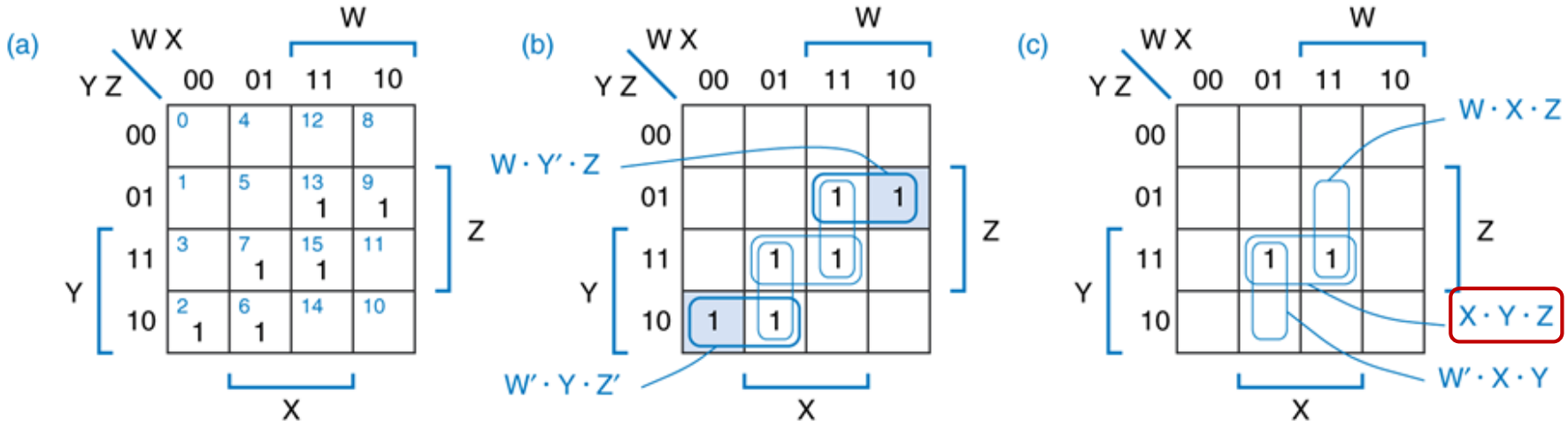
- Se IPs essenciais já **cobrem todos os 1s**: nenhum outro IP precisa ser incluso no circuito mínimo
  - Exemplos:





Mais minimização

- 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:

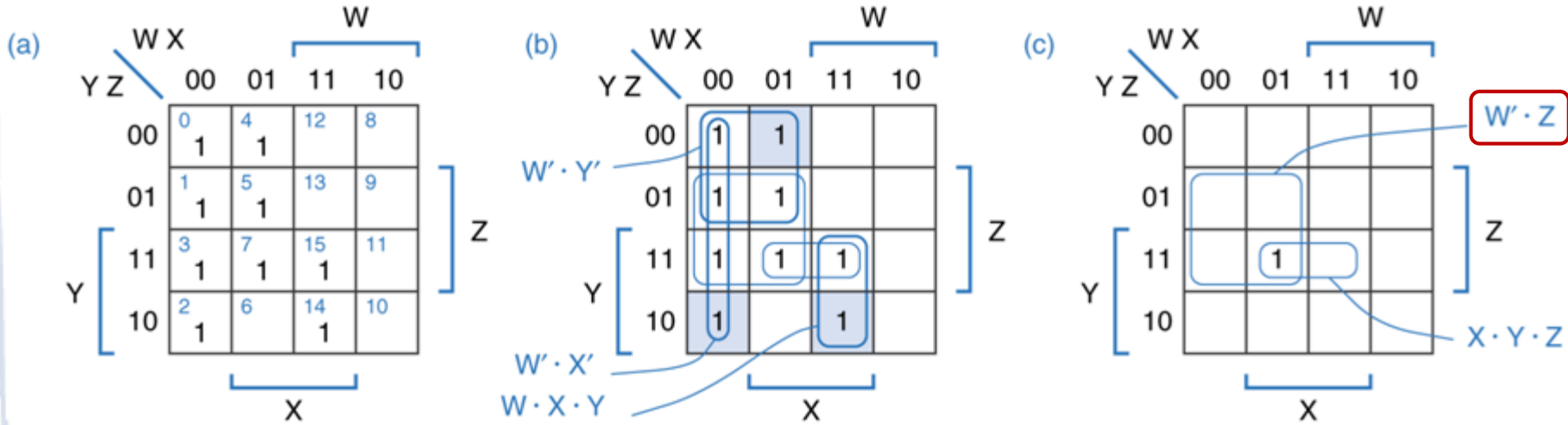


$F = \Sigma_{W,X,Y,Z}(2,6,7,9,13,15)$

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

Mais minimização

- 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:

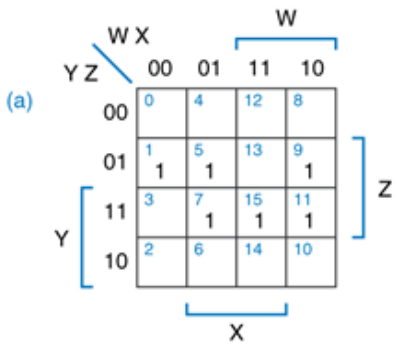


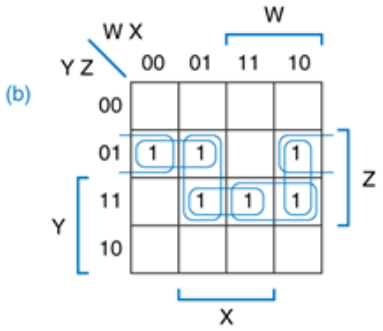
$F = \Sigma_{W,X,Y,Z}(0,1,2,3,4,5,7,14,15)$

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

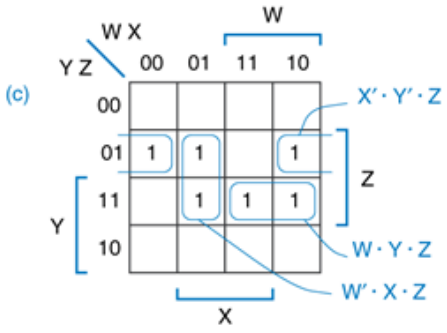
Mais minimização

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

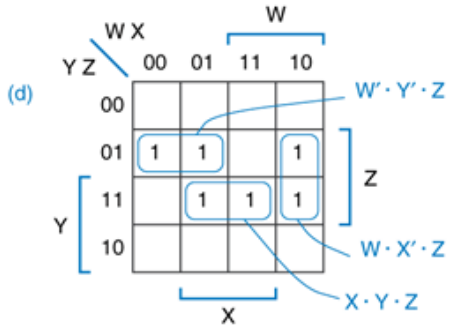




Sem IPs essenciais



Duas possibilidades

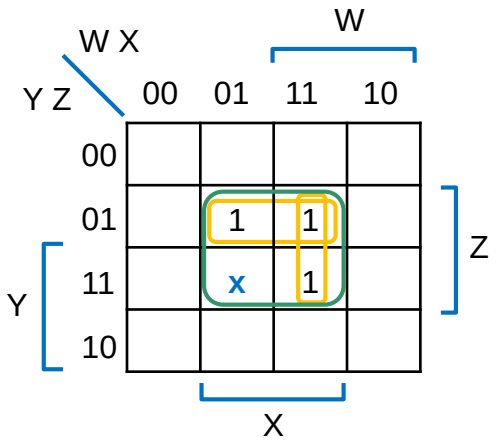


$F = W' \cdot X \cdot Z + W \cdot Y \cdot Z + X' \cdot Y' \cdot Z$

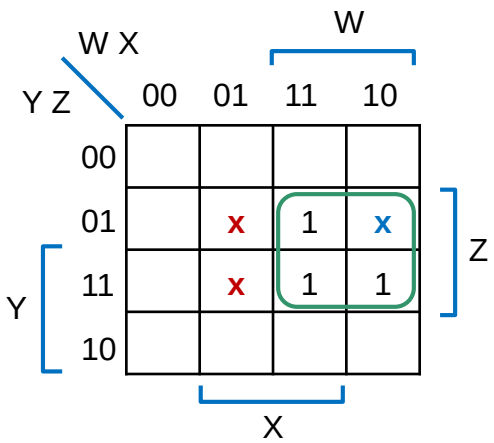
$F = X \cdot Y \cdot Z + W \cdot X' \cdot Z + W' \cdot Y' \cdot Z$

Minimização com “don’t care”

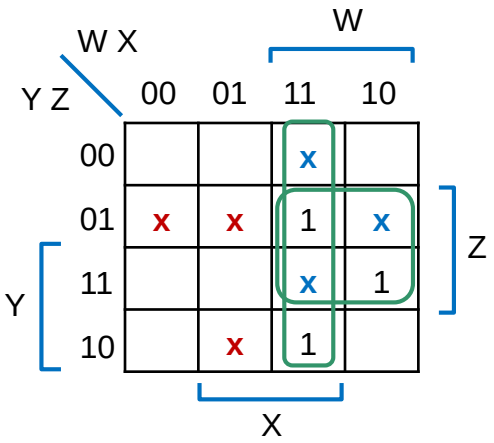
- Em alguns casos, a saída para certas combinações de entradas não importa (saída = “x”): pode ser 0 ou 1
  - Ex.: combinação de entradas não acontece na prática
- Minimização: escolher x=0 ou x=1 para reduzir número de IPs e/ou aumente a cobertura dos IPs



F = X•Z < X•W•Z + X•Y’•Z



F = W•Z



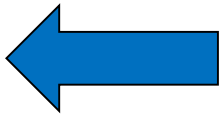
F = W•Z + WX

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

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

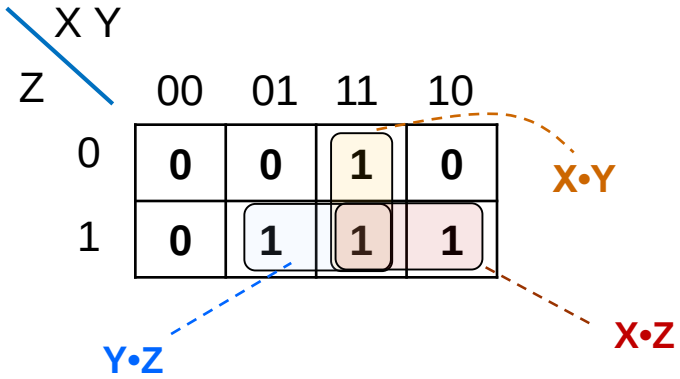


Minimização via manipulação algébrica dos mintermos

Exercício

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

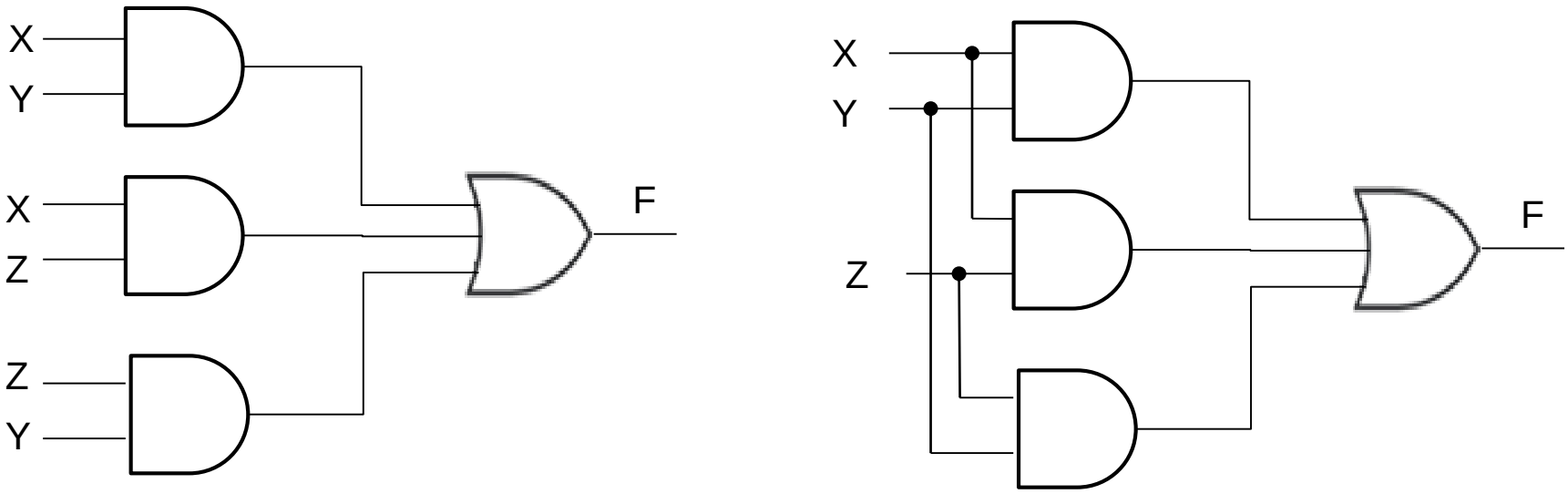
X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Minimização via Mapa de Karnaugh

Exercício

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



Dois diagramas (equivalentes) de portas lógicas

Minimização: Método Tabular

- Também denominado “Algoritmo de Quine-McCluskey”
- É um procedimento de extração dos Implicantes Primários (IPs).
  - Método programável: pode-se construir algoritmo iterativo para implementá-lo de forma automatizada!!!

Procedimento

- **Passo 1:**
  - Listam-se os mintermos de **f**, (“Cubos-0”) com a notação:
    - Ex: $m_6 = x_4'x_3x_2'x_1 = 0110$ e $m_4 = x_4'x_3x_2'x_1' = 0100$
    - Ex: $m_7 = x_4'x_3x_2x_1 = 0111$ e $m_5 = x_4'x_3x_2'x_1 = 0101$
  - Agrupam-se os Cubos-0 pelo número de 1’s
  - Identificam-se pares de Cubos-0 compatíveis (apenas1 bit diferente); isto é, t.q. exista um Cubo-1 que os contenha.
  - Define-se uma operação entre Cubos-0 compatíveis para gerar o Cubo-1 que os contém. Os Cubos-1 gerados colocados em outra tabela para o passo 2.
    - Ex: 0110 combinado com 0100 gera 01-0
    - Ex: 0111 combinado com 0101 gera 01-1
    - Ex: 0100 combinado com 0101 gera 010-
    - Ex: 0111 combinado com 0110 gera 011-

Procedimento

- **Passo 2**
  - Os Cubos-1 gerados pelo Passo 1 são agrupados de acordo com o número de 1’s.
  - Combinamos Cubos-1 para gerar Cubos-2:
    - Ex: 01-1 combinado com 01-0 gera 01--
    - Ex: 011- combinado com 010- gera 01--
- **Passo n+1**
  - O procedimento é análogo aos anteriores.
  - Continua-se essa forma até que não sejam gerados cubos de ordem superior.
- **Ao Fim**
  - Os Cubos que não foram mais combinados são os IPs!

Exemplo

$$f = \sum(0,2,4,5,7,8,10,12,15)$$

$x_2\,x_1$ \ $x_4\,x_3$	00	01	11	10
00	(0)	(4)	(12)	(8)
01	(1)	(5)	(13)	(9)
11	(3)	(7)	(15)	(11)
10	(2)	(6)	(14)	(10)

$x_2\,x_1$ \ $x_4\,x_3$	00	01	11	10
00	**1**	**1**	**1**	**1**
01		**1**		
11		**1**	**1**	
10	**1**			**1**

Exemplo: Passo 1

	x_4	x_3	x_2	x_1
(0)	0	0	0	0
(2)	0	0	1	0
(4)	0	1	0	0
(8)	1	0	0	0
(5)	0	1	0	1
(10)	1	0	1	0
(12)	1	1	0	0
(7)	0	1	1	1
(15)	1	1	1	1

Exemplo: Passos 2 e 3

	x_4	x_3	x_2	x_1
(0)	0	0	0	0
(2)	0	0	1	0
(4)	0	1	0	0
(8)	1	0	0	0
(5)	0	1	0	1
(10)	1	0	1	0
(12)	1	1	0	0
(7)	0	1	1	1
(15)	1	1	1	1

Passo 2
(um só bit de diferença)

Passo 3

	X_4	x_3	x_2	x_1
(0,2)	**0**	**0**	**–**	**0**
(0,4)	**0**	**–**	**0**	**0**
(0,8)	–	0	0	0
(2,10)	–	0	1	0
(4,5)	0	1	0	–
(4,12)	–	1	0	0
(8,10)	**1**	**0**	**–**	**0**
(8,12)	**1**	**–**	**0**	**0**
(5,7)	0	1	–	1
(7,15)	–	1	1	1
(0,2,8,10)	**–**	**0**	**–**	**0**
(0,4,8,12)	**–**	**–**	**0**	**0**

Exemplo: Marcando IPs

	x_4	x_3	x_2	x_1
(0)	0	0	0	0
(2)	0	0	1	0
(4)	0	1	0	0
(8)	1	0	0	0
(5)	0	1	0	1
(10)	1	0	1	0
(12)	1	1	0	0
(7)	0	1	1	1
(15)	1	1	1	1

IPs do passo 2

IPs do passo 3

	X_4	x_3	x_2	x_1
(0,2)	0	0	–	0
(0,4)	0	–	0	0
(0,8)	–	0	0	0
(2,10)	–	0	1	0
(4,5)	**0**	**1**	**0**	**–**
(4,12)	–	1	0	0
(8,10)	1	0	–	0
(8,12)	1	–	0	0
(5,7)	**0**	**1**	**–**	**1**
(7,15)	**–**	**1**	**1**	**1**
(0,2,8,10)	**–**	**0**	**–**	**0**
(0,4,8,12)	**–**	**–**	**0**	**0**

Exemplo: Escolhendo os IPs

	0	2	4	5	7	8	10	12	15
(4,5)			X	X					
(5,7)				X	X				
(7,15)					X				X
(0,2,8,10)	X	X				X	X		
(0,4,8,12)	X		X			X		X	

Escolhemos IPs que cobrem todos mintermos

(7,15) = -111 = $x_3 \cdot x_2 \cdot x_1$: IP essencial devido a 15

(0,2,8,10) = -0-0 = $x_3' \cdot x_1'$: IP essencial devido a 2 e 10

(0,4,8,12) = --00 = $x_2' \cdot x_1'$: IP essencial devido a 12

Por fim, escolhemos (4,5) ou (5,7) para cobrir o 5.

Métodos de minimização programáveis

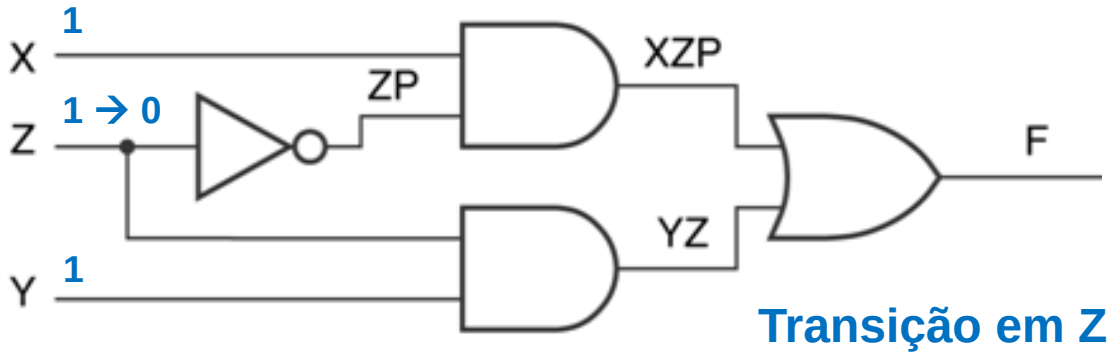
- Algoritmos clássicos:
  - Quine-McCluskey
  - Iterative consensus
- Melhorias computacionais:
  - baseado nos algoritmos clássicos, utilizam boas estruturas de dados e/ou alteram ordem dos passos.
- Métodos Heurísticos: saída não exata
  - Espresso-II
- Minimização com múltiplas saídas:
  - Espresso MV: observa minimização de múltiplas saídas usando lógica multivalorada (não-binária).

Ref: DDPPonline – section Pmin

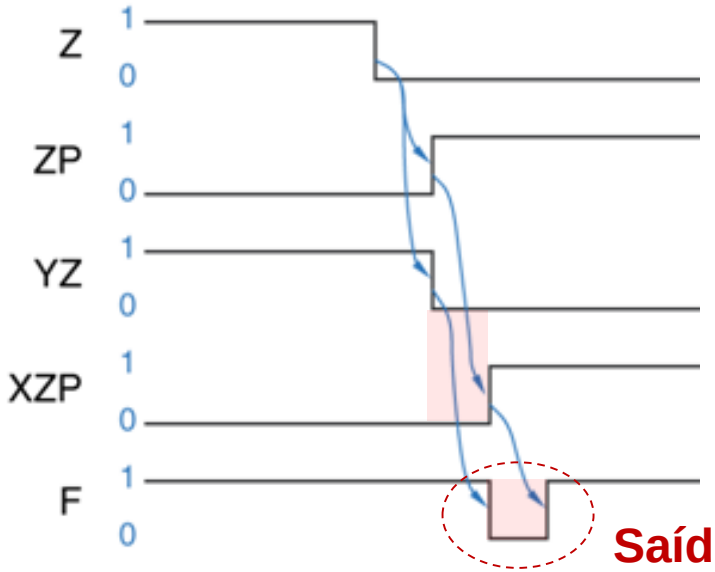
Timing Hazards

- Também denominados perigos/dificuldades de tempo.
- Métodos apresentados/estudados até agora consideram sinais de entrada estáveis.
  - Mas existem atrasos de propagação dos sinais: lembrar das aulas de eletrônica digital e circuitos CMOS!
  - Esses atrasos podem provocar saídas espúrias de curta duração (*glitches*).
- "Static-*n* hazards": mudando apenas um bit na entrada, espera-se uma saída *n* constante, mas podem ocorrer momentos em que a saída assume o valor *n'*

Static-1 Hazard

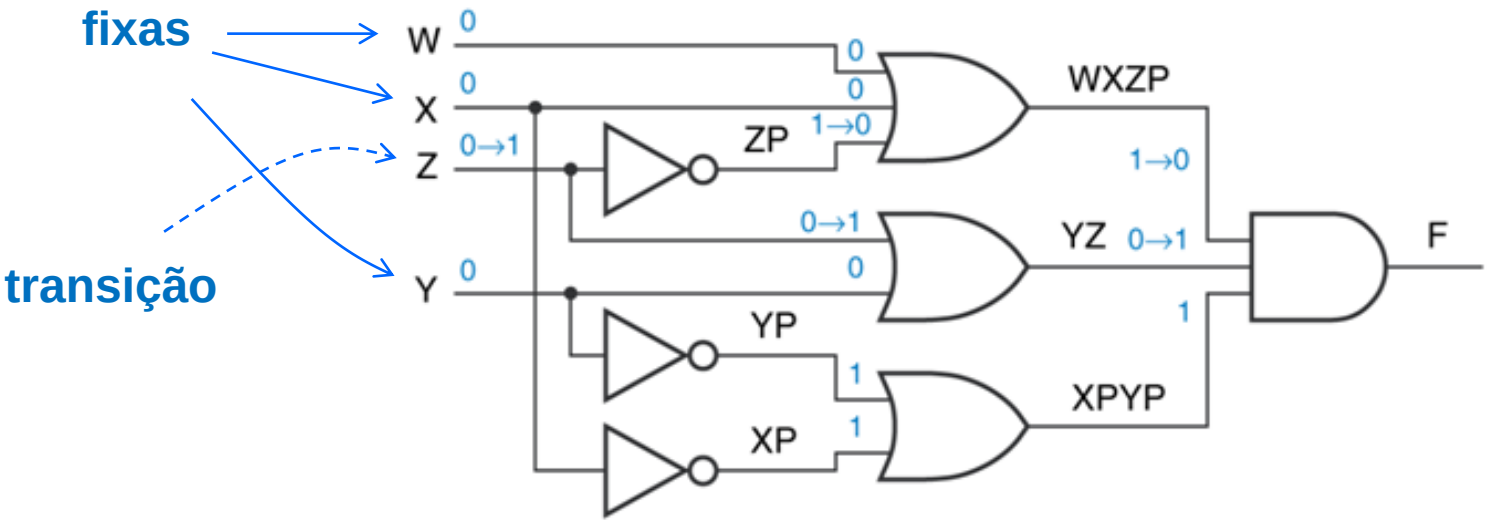


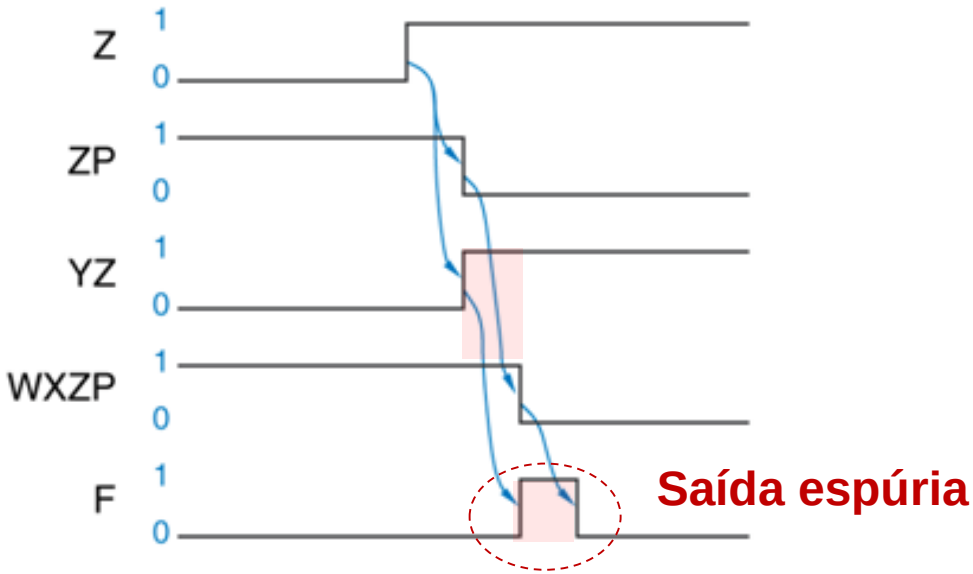
Transição em Z: efeito em YZ mais rápido que em XZP



Saída espúria

Static-0 Hazard

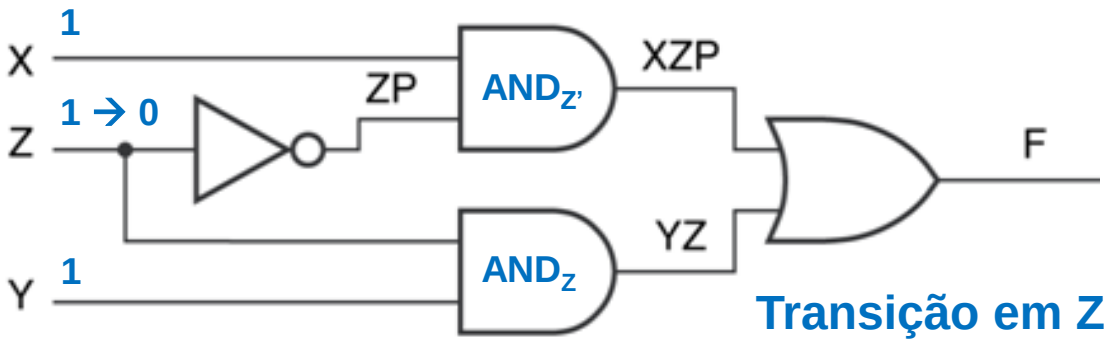




De onde vêm os *hazards*?

- Vamos analisar apenas circuitos na forma de **soma de produtos** com dois níveis
  - Têm sido o foco da atenção até agora
  - Técnicas usadas para eles são aplicáveis também a produtos de somas com dois níveis (princípio da dualidade)

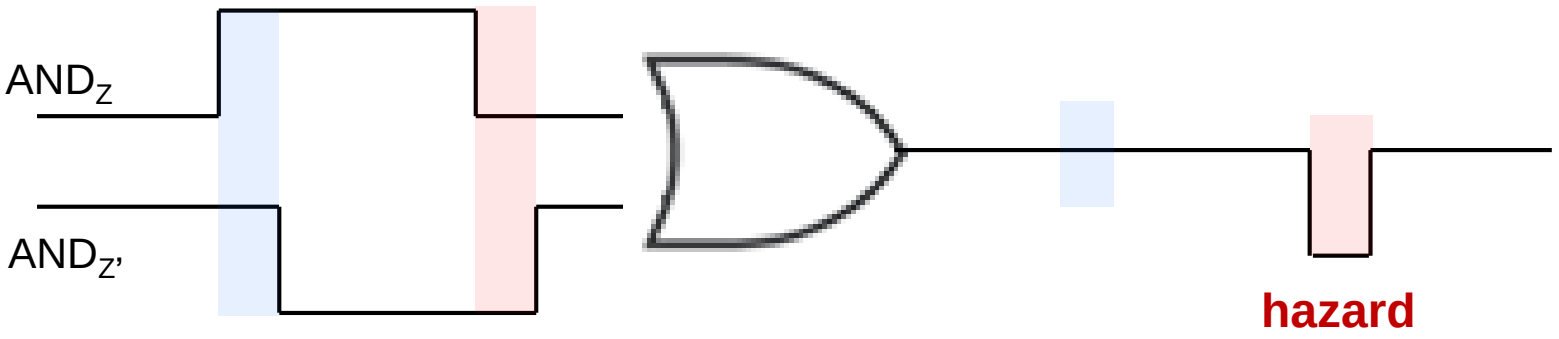
- Raiz do problema: Z e Z' em portas AND distintas



Transição em Z: efeito em YZ mais rápido que em XZP

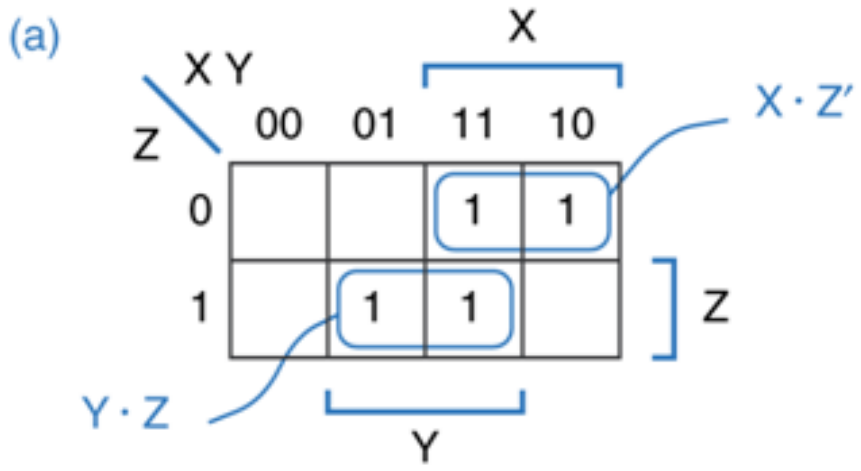
De onde vêm os *hazards*?

- Hazards: aparecem quando **uma porta AND depende de Z e outra depende de Z'**
  - AND_Z, é atrasado com relação a AND_Z ➔ se AND_Z for para 0 antes que AND_Z, vá para 1, pode haver um static-1 hazard
    - Saída espúria 0 porque ambos AND_Z e AND_Z, ficam em 0 temporariamente
  - Sem problema se AND_Z for para 1 antes de AND_Z, ir para 0



Como encontrar *hazards* usando mapas?

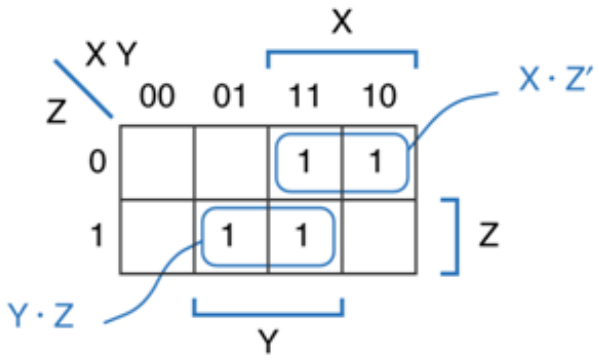
- Observar no mapa os IPs incluídos no circuito.
- Hazards existem se células vizinhas contendo 1's não estão cobertas por um mesmo IP.



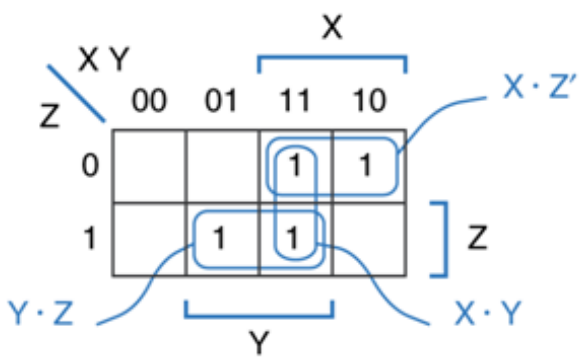
$$F = X \cdot Z' + Y \cdot Z$$

Como <u>resolver</u> *hazards* usando mapas?

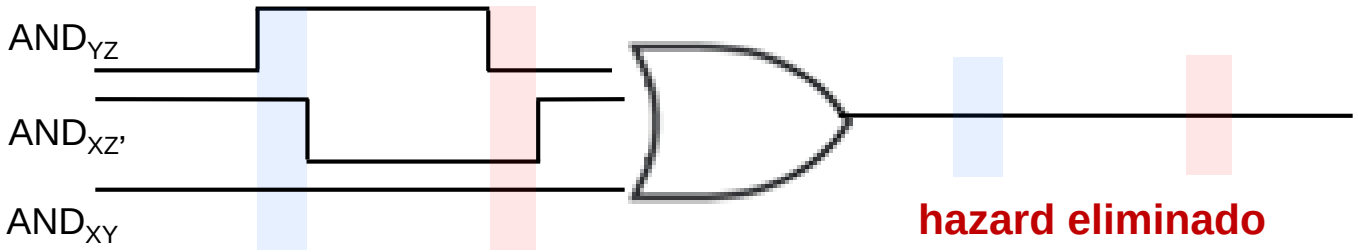
- Incluir IPs extras que cubram os vizinhos problemáticos



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

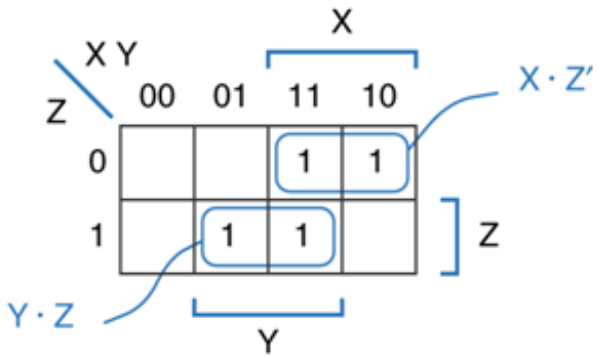


$F = X \cdot Z' + Y \cdot Z + X \cdot Y$

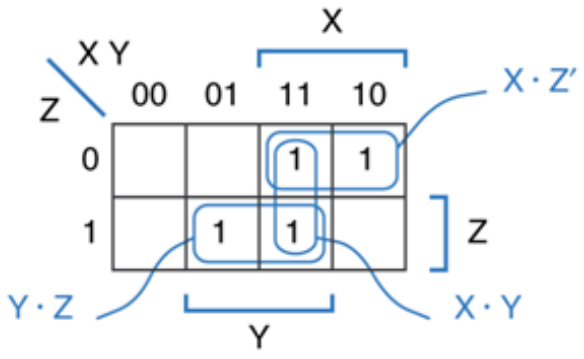


Como <u>resolver</u> *hazards* usando mapas?

- Incluir IPs extras que cubram os vizinhos problemáticos

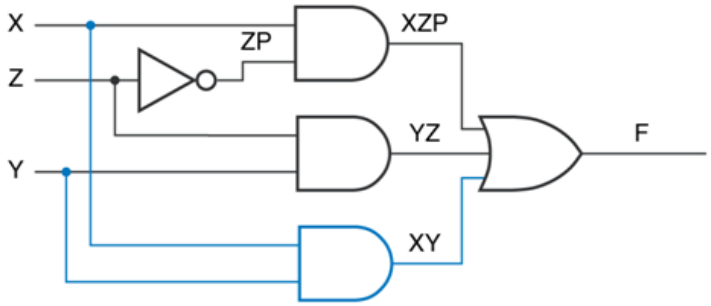


$F = X \cdot Z' + Y \cdot Z$



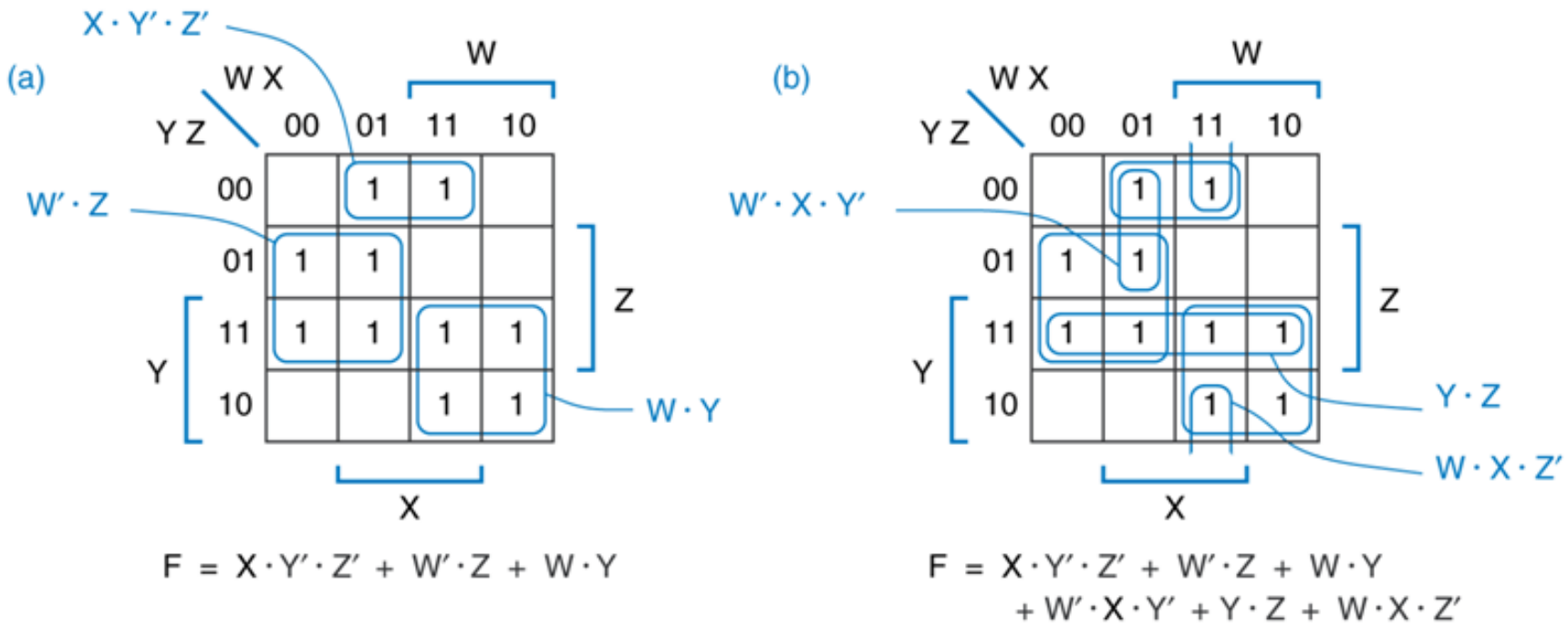
$F = X \cdot Z' + Y \cdot Z + X \cdot Y$

Circuito com hazard eliminado:



Outro exemplo de *static-1 hazard*

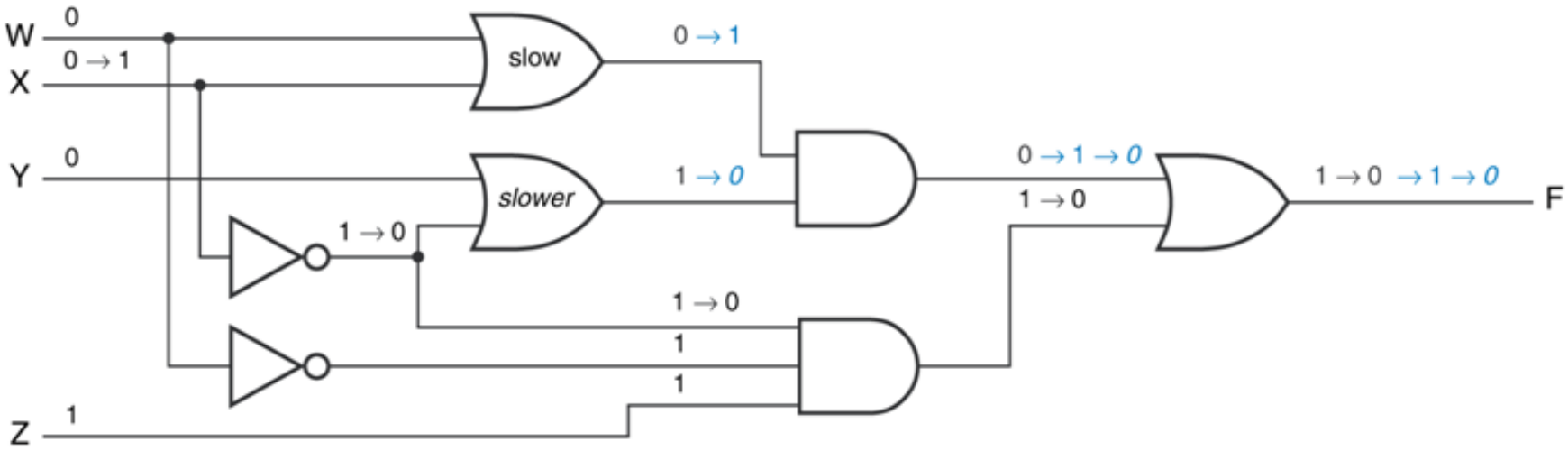
- Incluir IPs extras que cubram os IPs problemáticos



(a) Projeto original; (b) Projeto com produtos extras para eliminar static-1 hazards

Dynamic Hazards

- É a possibilidade de alterações na saída mais de uma vez como resultado de uma única transição de entrada.
- Podem ocorrer devido a caminhos com atrasos diferentes a partir da entrada em que houve mudança.

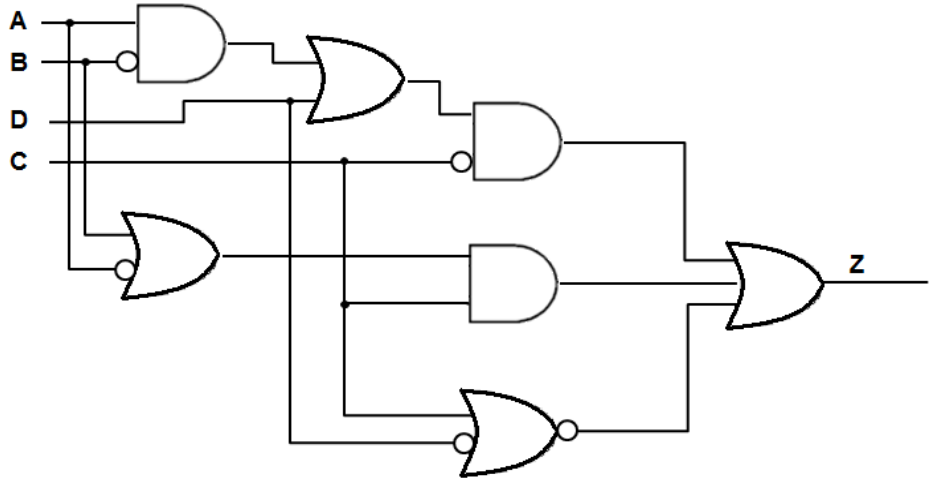


Projeto de circuitos livre de hazards

- Circuitos de dois níveis AND-OR bem projetados não estão sujeitos a hazards static-0 ou dinâmicos.
  - Static-1 hazards podem ser eliminados através do método indicado.
- Métodos gerais indicados nas referências extras no livro-texto.
- Críticos para circuitos assíncronos.

Exercício (P2-2016)

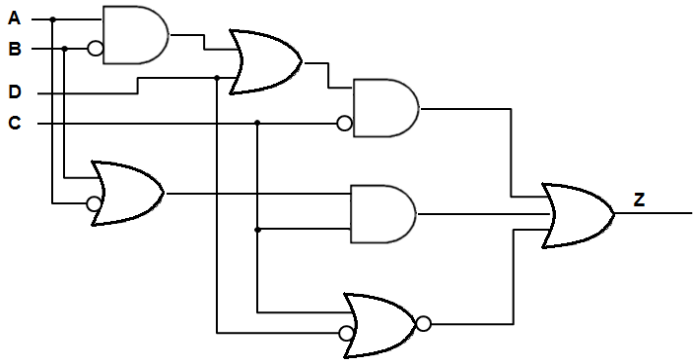
- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



- a) Qual a expressão de álgebra de chaveamento que descreve a saída Z em função das entradas A, B, C e D?
- b) Utilize um Mapa de Karnaugh para minimizar o circuito em questão. Descreva a expressão de álgebra de chaveamento mínima resultante (na forma de soma de produtos) e indique quais são os implicantes primários essenciais (IPEs) na mesma.

Exercício (P2-2016)

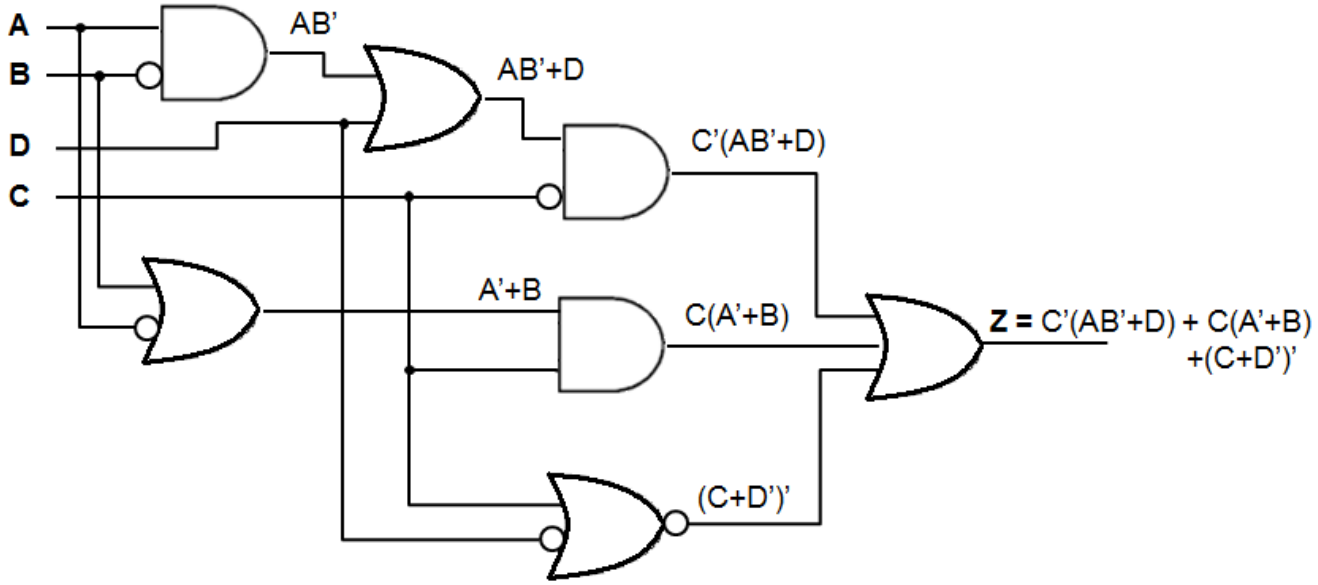
- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



  - c) Desenhe o circuito correspondente à expressão mínima obtida no item (b), na forma de soma de produtos.
  - d) Analisando melhor o problema que o circuito deve resolver, você percebe que, na prática, a saída do circuito não importa quando a combinação de entradas é A = 1, B = 0 e C=1. Portanto, você pode reprojetar o circuito para que todas as saídas correspondentes a essa combinação de entradas sejam do tipo “don’t care”. Isso possibilita alguma otimização extra? Justifique, apresentando o mapa de Karnaugh e a expressão mínima correspondente a esse cenário.

Exercício (P2-2016)

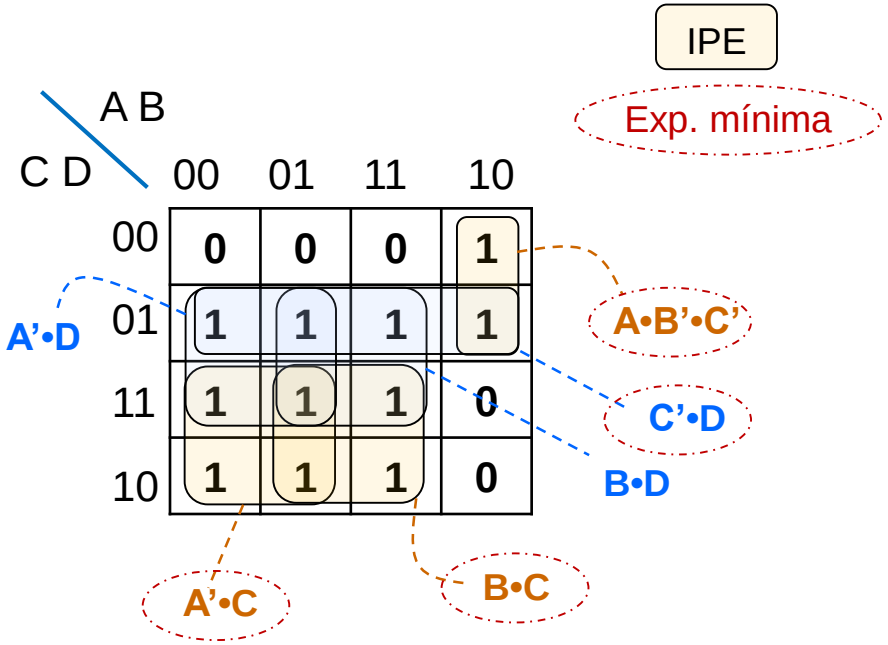
- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



- a) Qual a expressão de álgebra de chaveamento que descreve a saída Z em função das entradas A, B, C e D? **RESPOSTA**

Exercício (P2-2016)

- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:

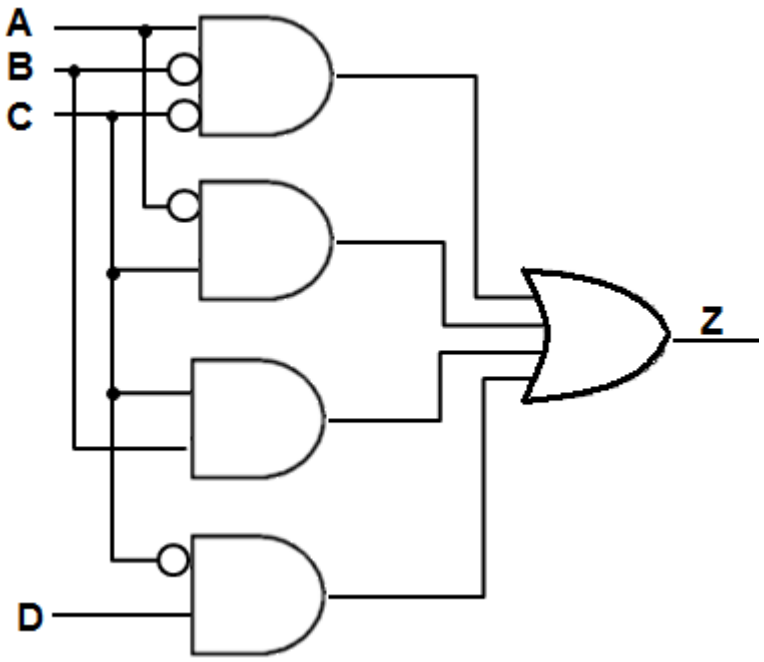


$Z = A'•C + A•B'•C' + B•C + C'•D$

- b) Utilize um Mapa de Karnaugh para minimizar o circuito em questão. Descreva a expressão de álgebra de chaveamento mínima resultante (na forma de soma de produtos) e indique quais são os implicantes primários essenciais (IPEs) na mesma. **RESPOSTA**

Exercício (P2-2016)

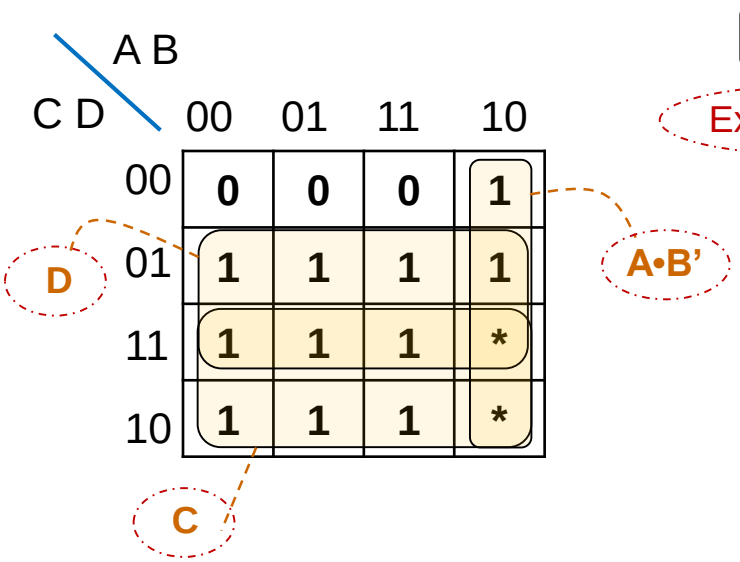
- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



- c) Desenhe o circuito correspondente à expressão mínima obtida no item (b), na forma de soma de produtos. **RESPOSTA**

Exercício (P2-2016)

- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



Z = A•B' + C + D

- d) Analisando melhor o problema que o circuito deve resolver, você percebe que, na prática, a saída do circuito não importa quando a combinação de entradas é A = 1, B = 0 e C=1. Portanto, você pode reprojetar o circuito para que todas as saídas correspondentes a essa combinação de entradas sejam do tipo “don’t care”. Isso possibilita alguma otimização extra? Justifique, apresentando o mapa de Karnaugh e a expressão mínima correspondente a esse cenário. **RESPOSTA**