PCS 3115 (PCS2215)

Sistemas Digitais I

Módulo 07 – Síntese de Circuitos Combinatórios

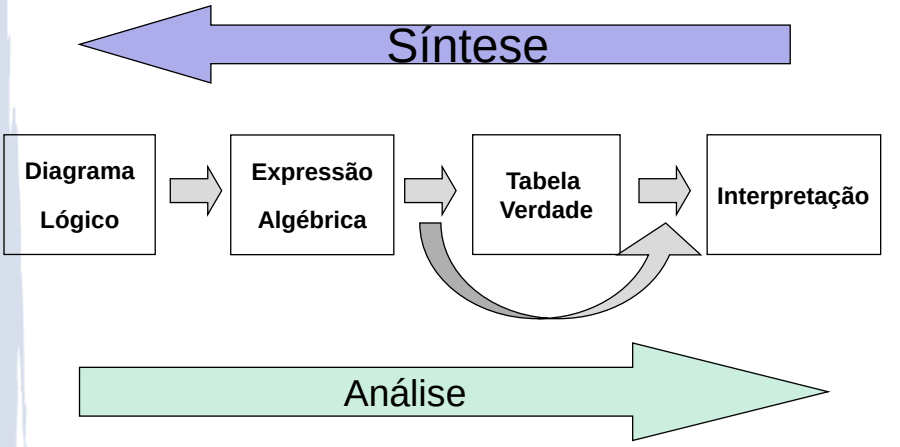
Prof. Dr. Marcos A. Simplicio Jr.

versão: 3.0 (Jan/2017)

Objetivos da aula

- Obter o circuito combinatório (diagrama) a partir de uma função lógica.
- Mapas de Karnaugh: Conceito, construção e utilização de mapas até 4 variáveis.
- Referência: seções 4.3.1 a 4.3.5 (início)

Síntese e Análise

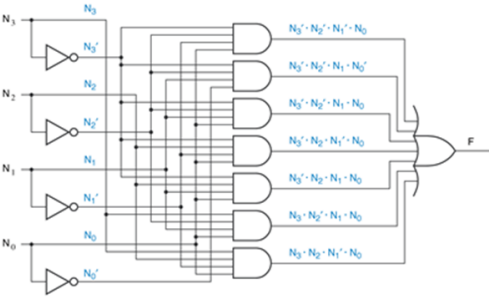


Síntese de Circuitos Combinatórios

- 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 = $\sum_{N_3N_2N_1N_0}(1,2,3,5,7,11,13)$



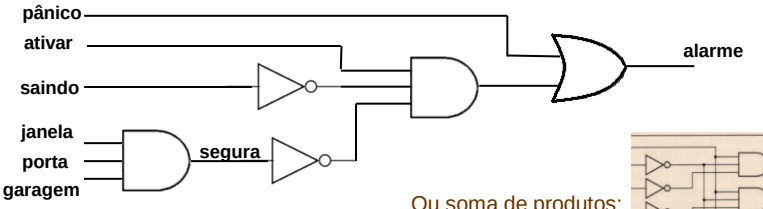
Ex: sistema de alarme

- "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"

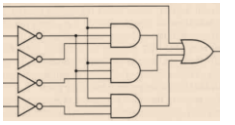
Ex: sistema de alarme

- "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)'

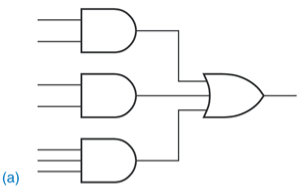


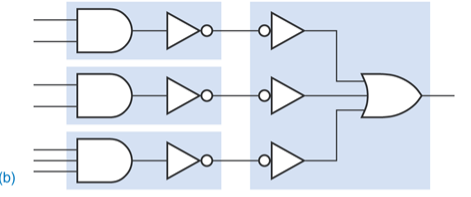
Ou soma de produtos:



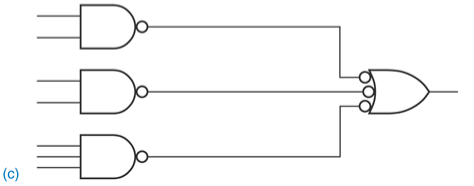
Soma de produtos e NANDs

- Conversão de circuito com ANDs/ORs para NANDs: nega-se saída da camada AND e entrada da camada OR. **Exemplo 1:**



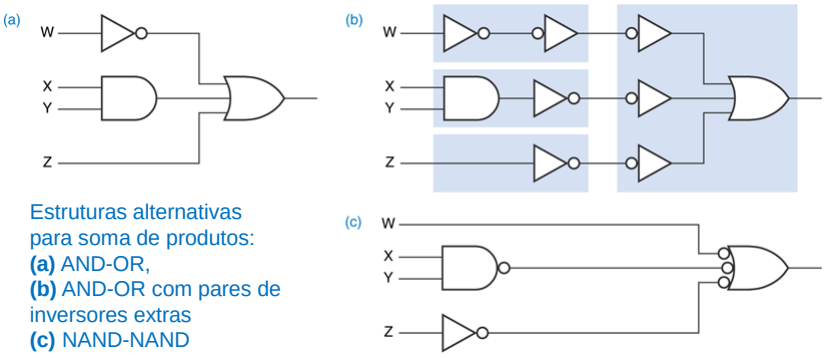


Estruturas alternativas para soma de produtos:
(a) AND-OR,
(b) AND-OR com pares de inversores extras
(c) NAND-NAND



Soma de produtos e NANDs

- Conversão de circuito com ANDs/ORs para NANDs: nega-se saída da camada AND e entrada da camada OR. **Exemplo 2:**

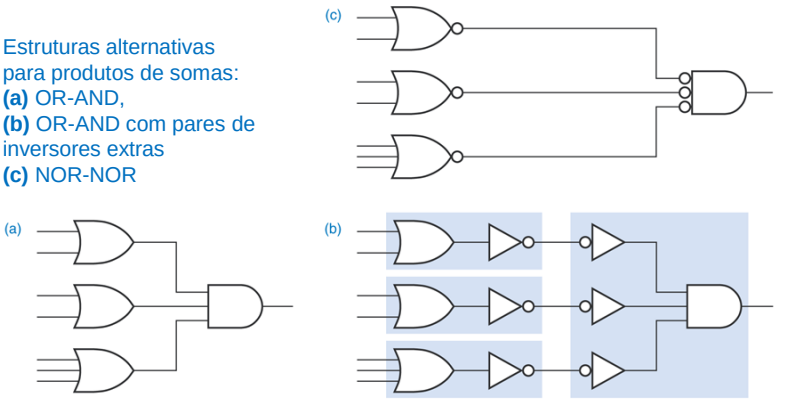


Estruturas alternativas para soma de produtos:
(a) AND-OR,
(b) AND-OR com pares de inversores extras
(c) NAND-NAND

Produto de somas e NORs

- Conversão de circuito com ANDs/ORs para NORs: nega-se saída da camada OR e entrada da camada AND. **Exemplo 1:**

Estruturas alternativas para produtos de somas:
(a) OR-AND,
(b) OR-AND com pares de inversores extras
(c) NOR-NOR



Outras manipulações

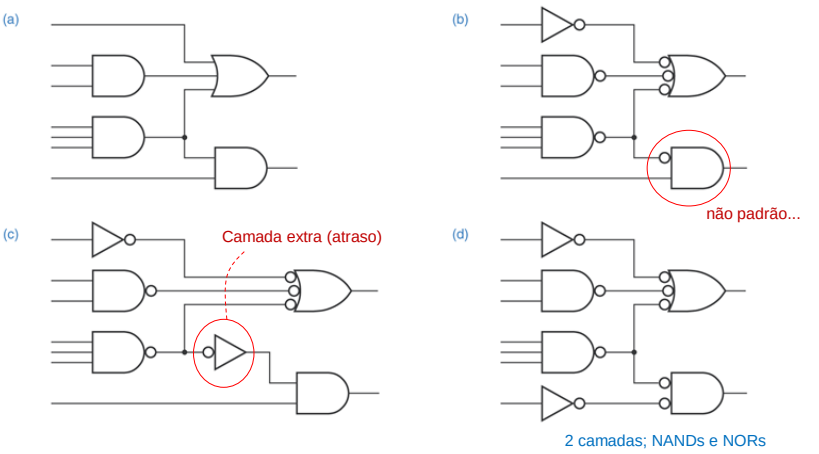
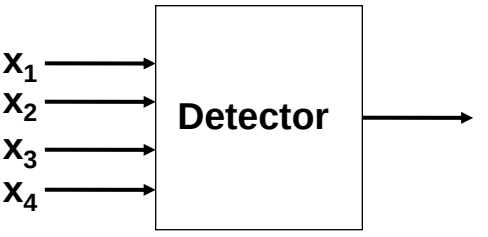


Figure 4-24

Logic-symbol manipulations: (a) original circuit;(b) transformation with a nonstandard gate; (c) inverter used to eliminate nonstandard gate; (d) preferred inverter placement.

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 os códigos BCD correspondentes aos números ímpares.

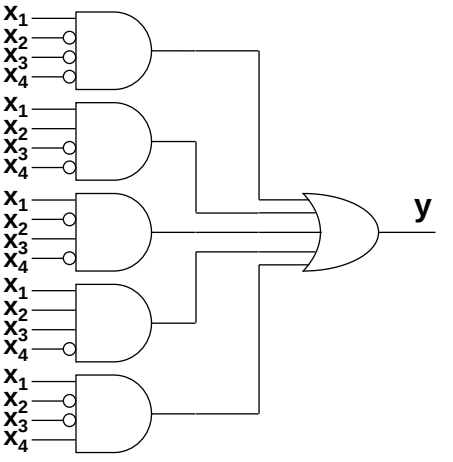


Ex. Detector ímpares
Tabela Verdade

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	

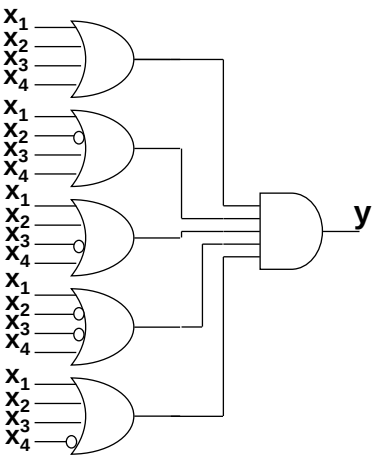
Ex. Detector ímpares
Soma de Produtos

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



Ex. Detector ímpares
Produto de Somas

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

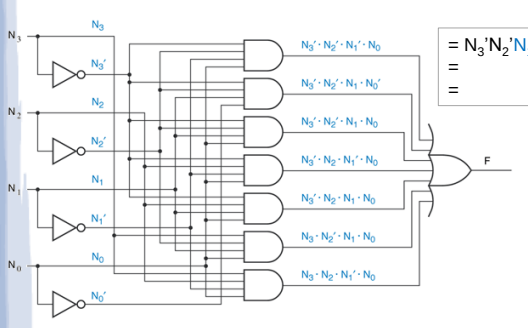


Er... Mas não era mais fácil fazer y = x1...?

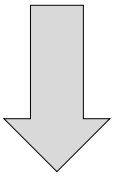
Minimização de circuitos combinatórios

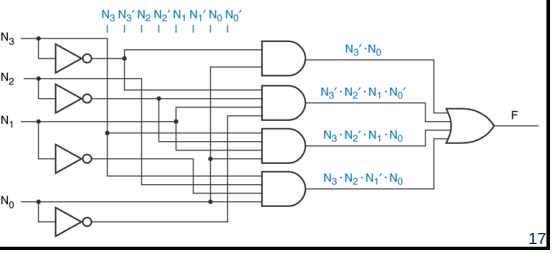
- Objetivo: Obter solução mais econômica!
- Critérios possíveis:
  - Minimização do número de literais da função de chaveamento.
  - Minimização do número de interconexões entre as portas.
  - Minimização do número de pinos do circuito integrado a ser eventualmente construído.
- Maioria das estratégias baseada em T10 (combinação)
  - (T10) X•Y+X•Y'=X (T10') (X+Y)•(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)

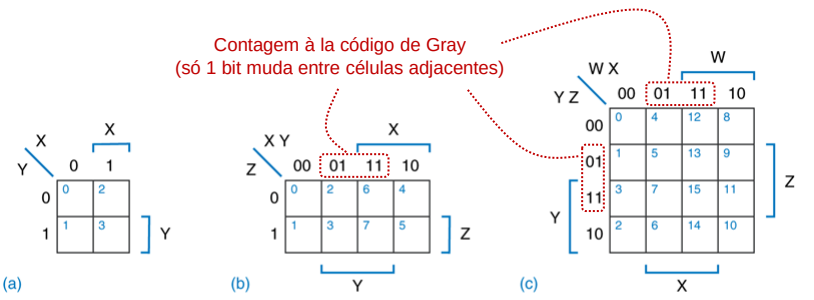


Figure 4-26
Karnaugh maps: (a) 2-variable; (b) 3-variable; (c) 4-variable.

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**)

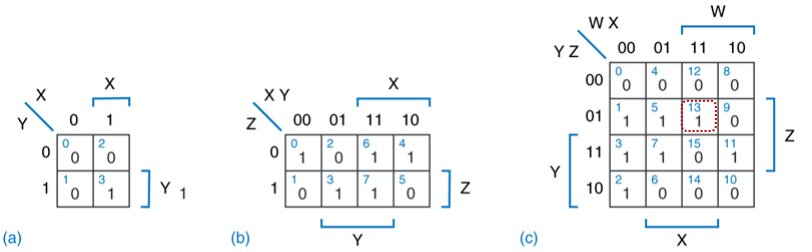
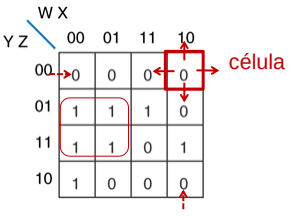


Figure 4-27

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 ≥ 0



Mapa de Karnaugh e Mintermos

- Grupamentos de 1's (mintermos): aplicação gráfica de T10!
  - Cada grupamento refere-se o 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

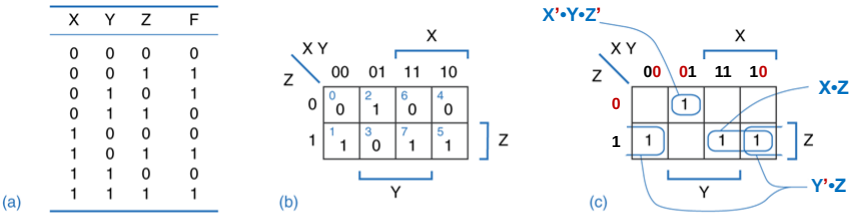
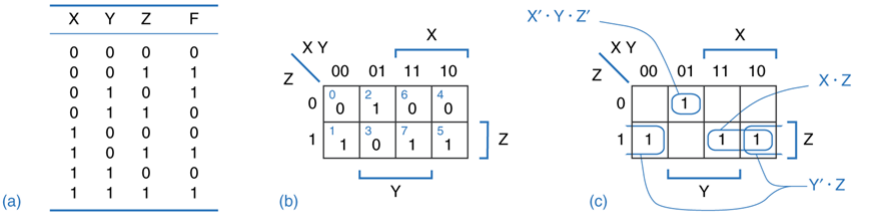
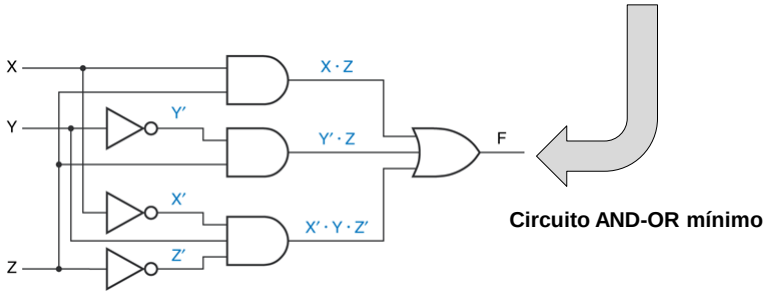


Figure 4-28

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

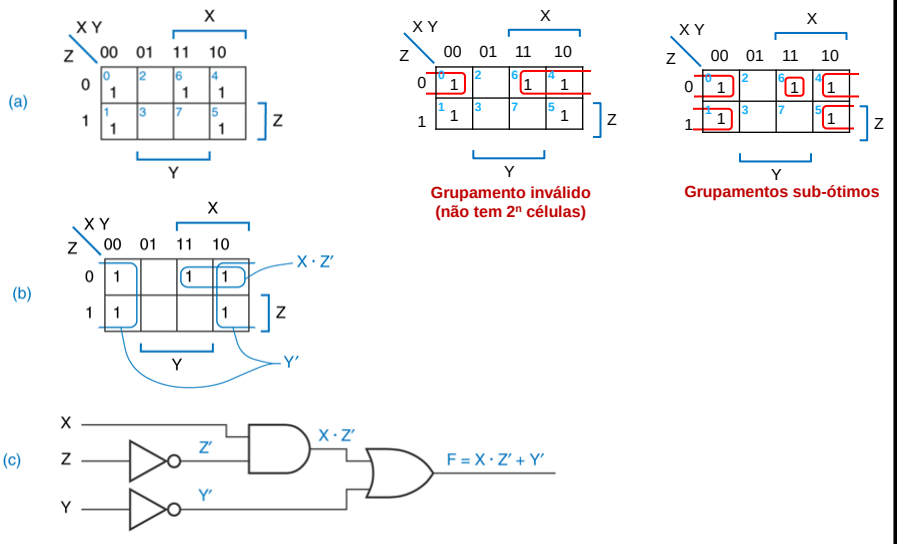
Mapa de Karnaugh e Mintermos



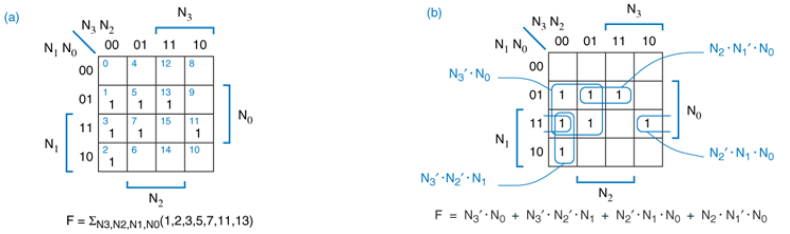


Circuito AND-OR mínimo

Outro exemplo



Exemplo: detector de números primos



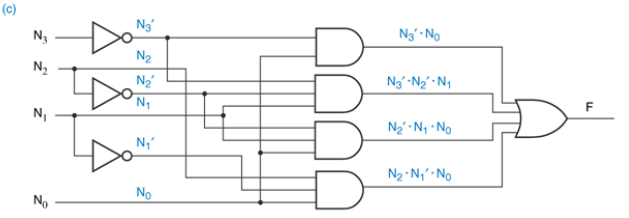


Figure 4-31
Prime-number detector: (a) initial Karnaugh map; (b) circled product terms; (c) minimized circuit.

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

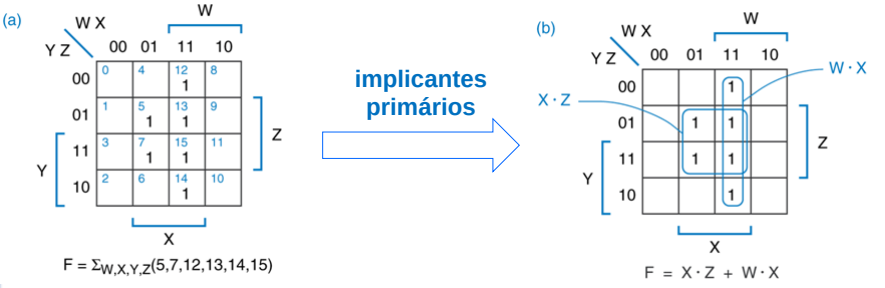
- Determinar o menor conjunto de adjacências que cubra (contenha) todos os mintermos, e escrever a soma de produtos correspondente

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

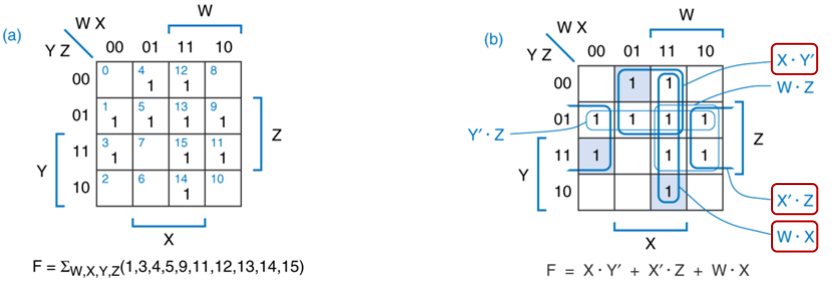
Mais minimização: Tabela de Cobertura

- **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 **uma soma de IPs**



Mais minimização: Tabela de Cobertura

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

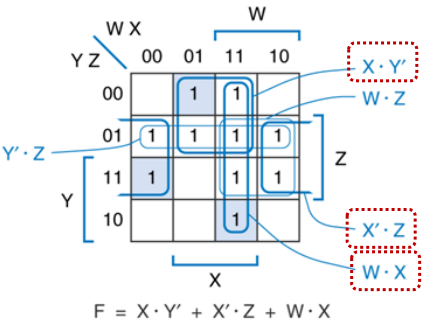


- 5 IPs, mas soma mínima requer apenas 3 deles: X•Y' + X'•Z + W•X já cobrem todos os 1s

➔ 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

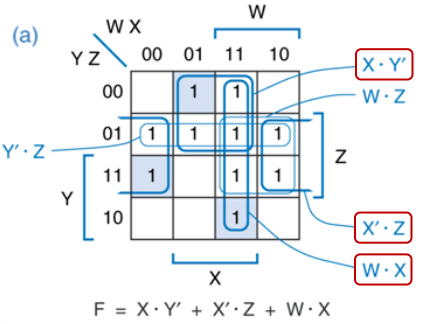


$F = X \cdot Y' + X' \cdot Z + W \cdot X$

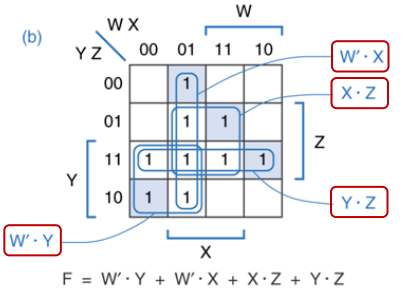
$F = \Sigma_{W,X,Y,Z}(1,3,4,5,9,11,12,13,14,15)$: (a) Karnaugh map; (b) prime implicants and distinguished 1-cells.

Mais minimização: Tabela de Cobertura

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



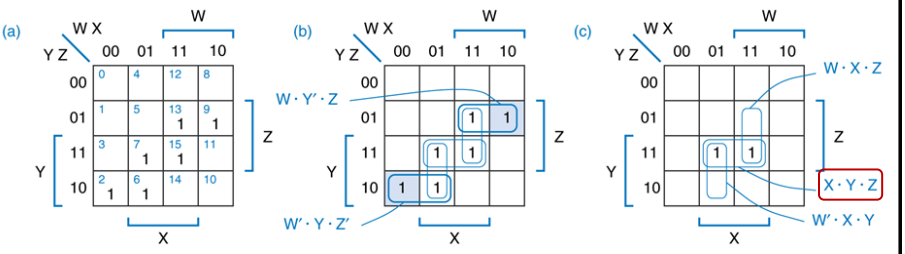
$F = X \cdot Y' + X' \cdot Z + W \cdot X$



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

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:

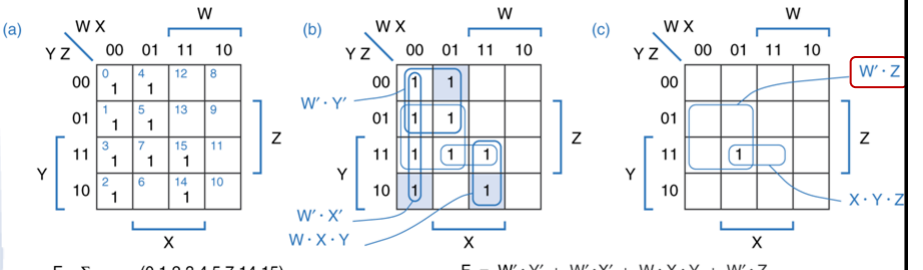


$F = \Sigma_{W,X,Y,Z}(2,6,7,9,13,15)$

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

Mais minimização: 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:

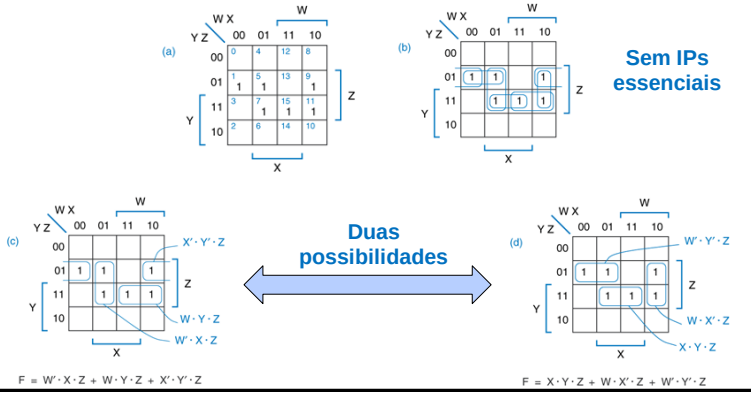


$F = \Sigma_{W,X,Y,Z}(0,1,2,3,4,5,7,14,15)$

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

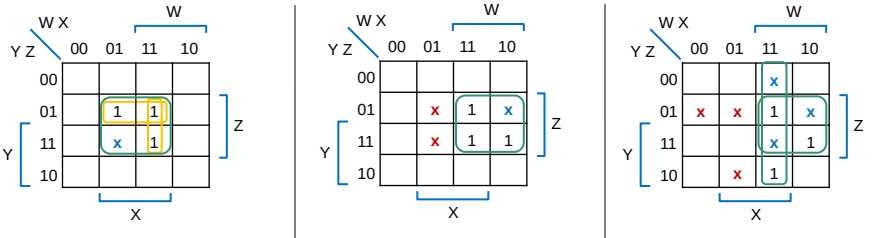
Mais minimização: Tabela de Cobertura

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



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



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 (1/2)

- **Passo 1:**
  - Listam-se os mintermos de **f**, com a notação correspondente no formato “Cubo-0” (cobertura de $2^0 = 1$ célula)
    - Ex: $m_7=x_4'x_3x_2x_1$
  - Agrupam-se os Cubos-0 de modo que:
    - No primeiro grupo todos possuam zero 1’s.
    - No segundo grupo todos possuam um 1, etc.
  - Identificam-se pares de Cubos-0 compatíveis, isto é, para os quais 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-0 são marcados com “√” e os Cubos-1 gerados colocados em outra tabela para o passo 2.

Procedimento (1/2)

- **Passo 2**
  - Os Cubos-1 gerados pelo Passo 1 são agrupados, de modo que:
    - Os elementos do primeiro grupo possuem zero 1's,
    - Os elementos do segundo grupo possuem um 1, etc.
  - É utilizado o mesmo procedimento de operação entre Cubos-1 para gerar Cubos-2 para o passo 3.
- **Passo 3**
  - O procedimento é análogo aos anteriores.
  - Continua-se essa forma até que não sejam gerados cubos de ordem superior.

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: Resultado

	X_4	x_3	x_2	x_1	
(0,8)	–	0	0	0	*fora*
(2,10)	–	0	1	0	*fora*
(*4*,**5**)	0	1	0	–	? ------> IP_3: $x_4'x_3x_2'$
(4,12)	–	1	0	0	*fora*
(**5**,*7*)	0	1	–	1	? ------> IP_4: $x_4'x_3x_1$
(7,15)	–	1	1	1	●——→ IPE_1: $x_3x_2x_1$
(0,2,8,10)	–	0	–	0	●——→ IP_1: $x_3'x_1'$
(0,4,8,12)	–	–	0	0	●——→ IP_2: $x_2'x_1'$

IPE_1: essencial devido a 15
IP_1 e IP_2: maiores grupamentos
IP_3 e IP_4: apenas um deles é necessário devido a 5

Métodos de minimização programáveis

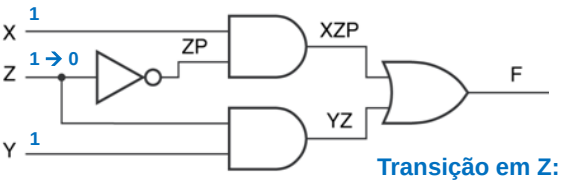
- Algoritmos clássicos:
  - Quine-McCluskey
  - Iterative consensus
- Melhorias computacionais:
  - baseado nos algoritmos clássicos, utilizam boas estruturas de dados e/ou alteram ordem dos passos.
- Métodos Heurísticos: saída não exata
  - Espresso-II
- Looking at things differently:
  - Espresso MV: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

Ref: DDPPonline – section Pmin

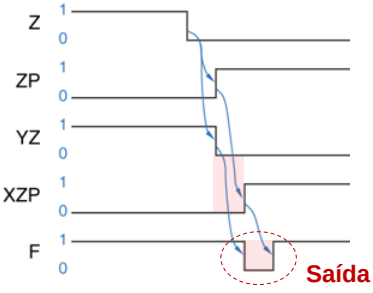
Timing Hazards

- Também denominados perigos/dificuldades de tempo.
- Métodos apresentados/estudados até agora consideram sinais de entrada estáveis (*steady-state*).
  - 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": quando se espera uma saída *n* constante, mas durante transições observam-se momentos em que a saída assume o valor *n'*

Static-1 Hazard

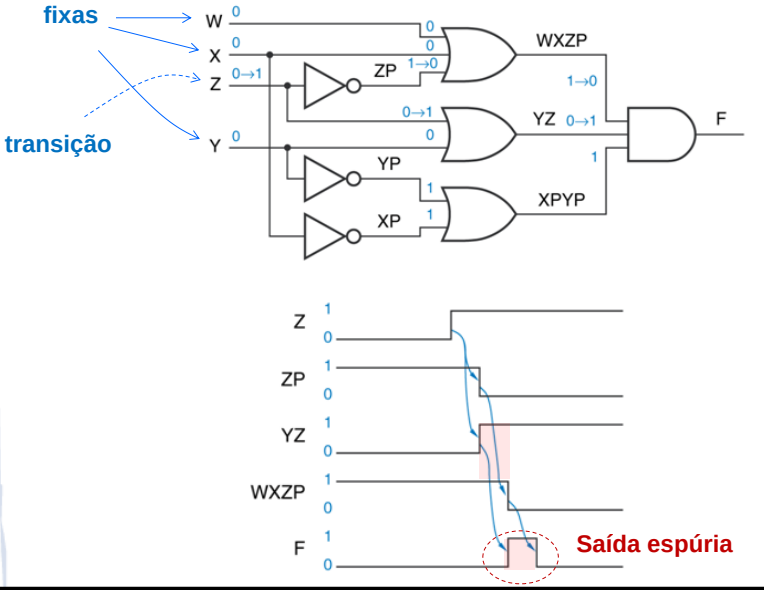


Transição em Z: efeito em YZ mais rápido que em XZP



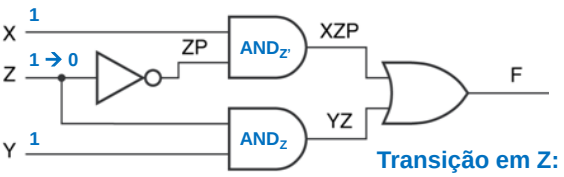
Saída espúria

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)
- Raíz do problema: Z e Z' em portas AND distintas



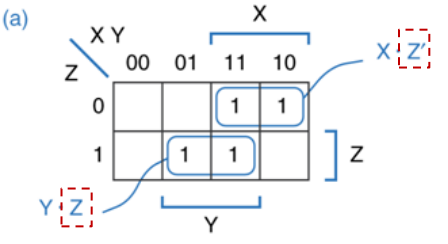
Transição em Z: efeito em YZ mais rápido que em XZP

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 <u>encontrar</u> *hazards* usando mapas?

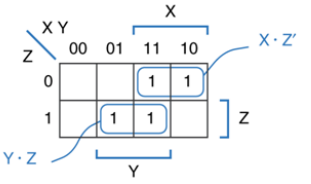
- Observar se IPs incluídos no circuito contêm termo e seu complemento



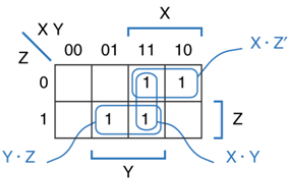
$$F = X \cdot Z' + Y \cdot Z$$

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

- Incluir IPs extras que cubram os IPs problemáticos
  - Isso equivale ao consenso entre os IPs originais: não é afetado pelo atraso!



F = X · Z′ + Y · Z

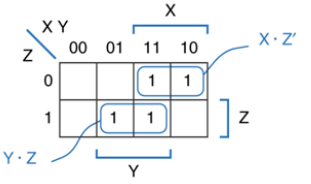


F = X · Z′ + Y · Z + X · Y

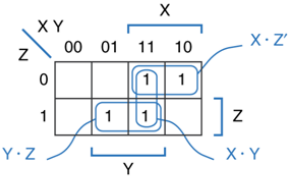
hazard eliminado

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

- Incluir IPs extras que cubram os IPs problemáticos
  - Isso equivale ao consenso entre os IPs originais: não é afetado pelo atraso!

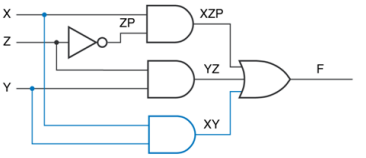


F = X · Z′ + Y · Z



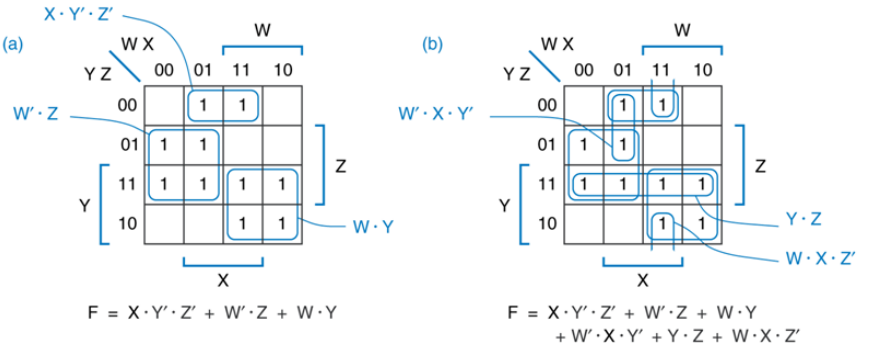
F = X · Z′ + Y · Z + X · Y

Circuito com hazard eliminado:



Outro exemplo de *static-1 hazard*

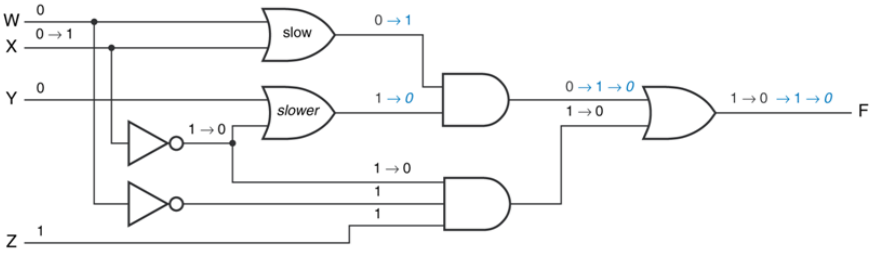
- Incluir IPs extras que cubram os IPs problemáticos
  - Isso equivale ao consenso entre os IPs originais: não é afetado pelo atraso!



(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.

Tarefas

- Leitura das seções 4.3.7 e 4.4.
- Exercícios do Capítulo 4 do livro-texto
  - ao menos *drill problems* 4.18 e 4.19
  - Exercícios 4.47 a 4.64.

Apêndice

Exercícios

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

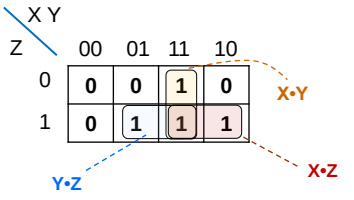
(X’•Y•Z)+(X•Y’•Z)+(X•Y•Z’)+(X•Y•Z)
= X•Y•(Z+Z’)+X’•Y•Z+X•Y’•Z
= X•Y+X’•Y•Z+X•Y’•Z
= X•(Y+Y’•Z)+X’•Y•Z
= X•(Y+Z) + X’•Y•Z
= X•Y+X•Z + X’•Y•Z
= X•Y+Z•(X + X’•Y)
= X•Y+Z•(X+Y)
= X•Y+Z•X+Z•Y

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

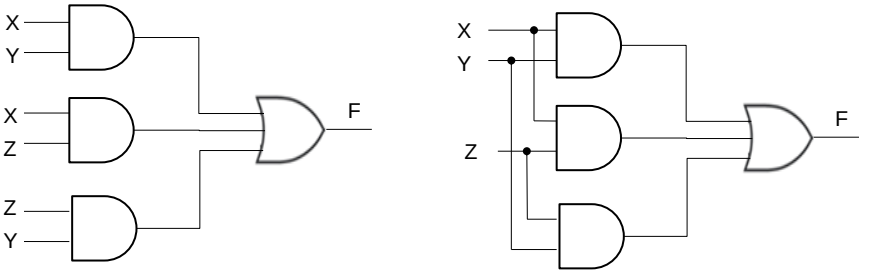
X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Minimização via Mapa de Karnaugh

Exercício

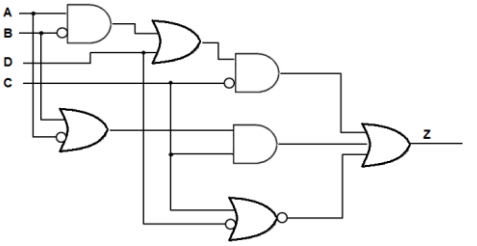
- Sintetize um circuito para a função “F = maioria dentre 3 variáveis X, Y e Z”. Minimize o circuito obtido
  - F = X•Y+Z•X+Z•Y



Dois diagramas (equivalentes) de portas lógicas

Exercício (P2-2016)

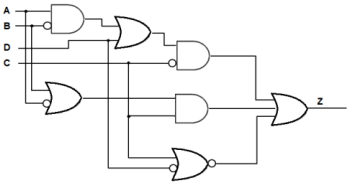
- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



- 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)

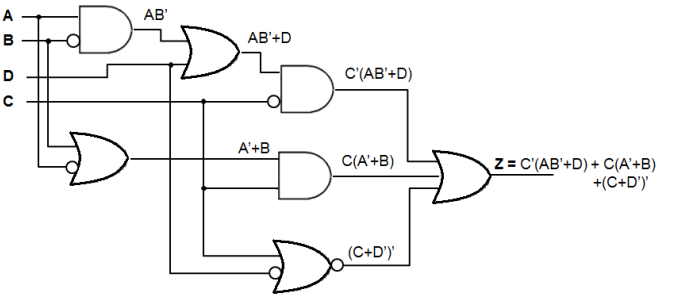
- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



- 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)

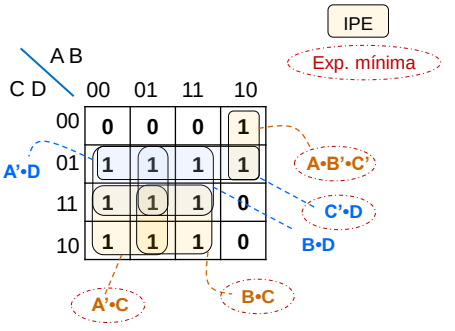
- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



- 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:

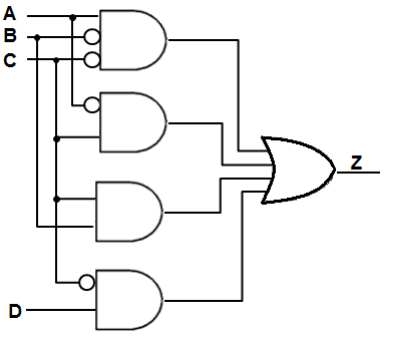


Z = A'•C + A•B'•C' + B•C + C'•D

- 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)

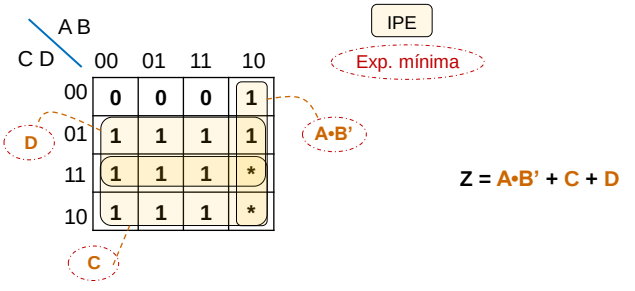
- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



- 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**