PCS 3115

Sistemas Digitais I

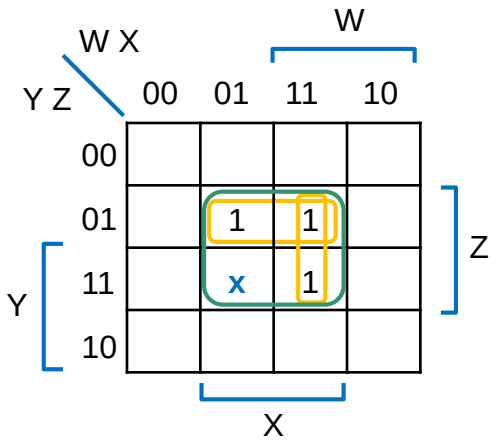
Síntese de Circuitos Combinatórios

Prof. Dr. Marcos A. Simplicio Jr.

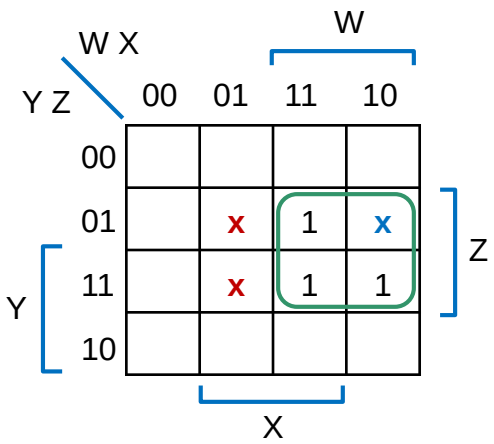
versão: 3.2 (Jan/2020)

Minimização com “don’t care”

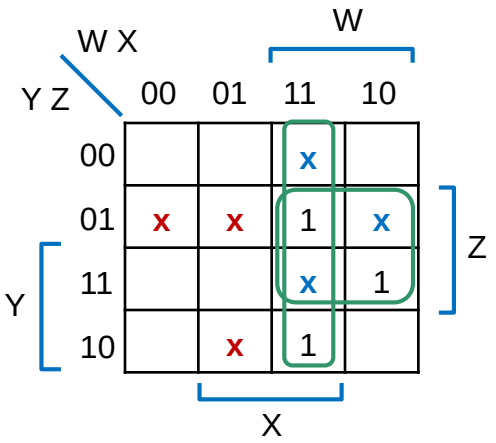
- Em alguns casos, a saída para certas combinações de entradas não importa (saída = “x”): pode ser 0 ou 1
  - Ex.: combinação de entradas não acontece na prática
- Minimização: escolher x=0 ou x=1 para reduzir número de IPs e/ou aumente a cobertura dos IPs



F = X•Z < X•W•Z + X•Y’•Z



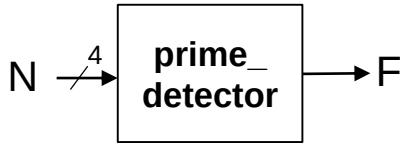
F = W•Z



F = W•Z + WX

Minimização com VHDL

- Várias otimizações feitas internamente por sintetizador
  - Mas bom projeto ajuda: veremos isso mais adiante
- Ex.: Detector de primos

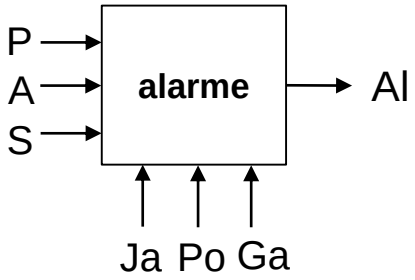



```
-- Detector de primos de 4 bits
library IEEE;
use IEEE.std_logic_1164.all;

entity prime_detector is
port( N: in std_logic (3 downto 0);
      F: out std_logic);
end prime_detector;

architecture arch of prime_detector is
begin              -- Obs.: usa bloco multiplexador (Tema da aula 9)
   with N select -- Primos-Wakerly: {1,2,3,5,7,11,13}
       F <= '1' when "0001"|"0010"|"0011"|"0101"|"0111"|"1011"|"1101",
            '0' when others; -- "catch all"
end arch;
```

Minimização com VHDL

- Ex.: “O alarme ativado (Al = 1) se entrada de pânico P=1, ou se entrada ativar A=1 e saindo S=0 e se a casa não estiver segura. A casa está segura (Se = 1) se entradas janela Ja , porta Po e garagem Ga forem 1”

```
-- Sistema de alarme
library IEEE;
use IEEE.std_logic_1164.all;

entity alarme is
port( P,A,S,Ja,Po,Ga: in std_logic;
      Al: out std_logic);
end alarme;

architecture arch of alarme is
   signal Se: std_logic;
begin -- Obs.: possivelmente usará portas lógicas
   Se <= Ja and Po and Ga;
   Al <= P or (A and (not S) and (not Se));
end arch;
```



Minimização com VHDL

- Ex.: Detector de ímpares

```
-- Detector de ímpares de 4 bits
library IEEE;
use IEEE.std_logic_1164.all;

entity impar is
port( X: in std_logic (3 downto 0);
      Y: out std_logic);
end impar;

architecture arch of impar is
begin -- Obs.: provavelmente usará 1 fio...
   Y <= X(0);
end arch;
```

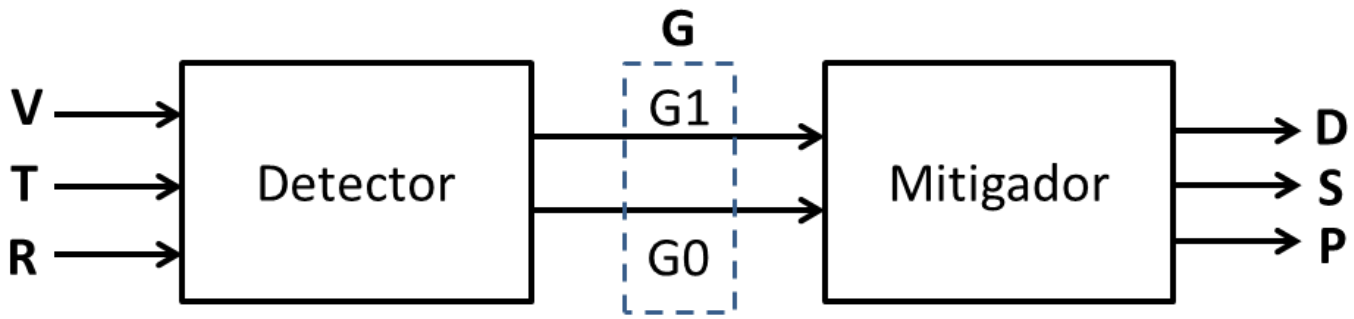
Tarefas

- 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.

Exercícios

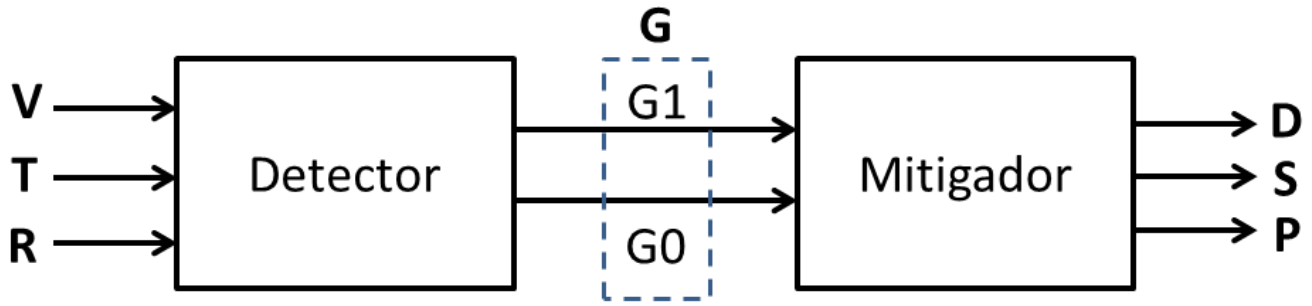
Exercício (P2-2017)

- **Cenário**: projeto de sistema de proteção p/ centrífugas de uma usina nuclear, com 2 módulos
  - **Detector**: monitora velocidade (V), temperatura (T) e radiação (R). Atribui gravidade de eventuais problemas na forma de sinal G de 2 bits (G1 e G0: respectivamente os bits mais e menos significativos de G);
  - **Mitigador**: usa entrada G para eventualmente ativar luz em um painel (P), soar sirene (S) e/ou forçar o desligamento do sistema (D).



Exercício (P2-2017)

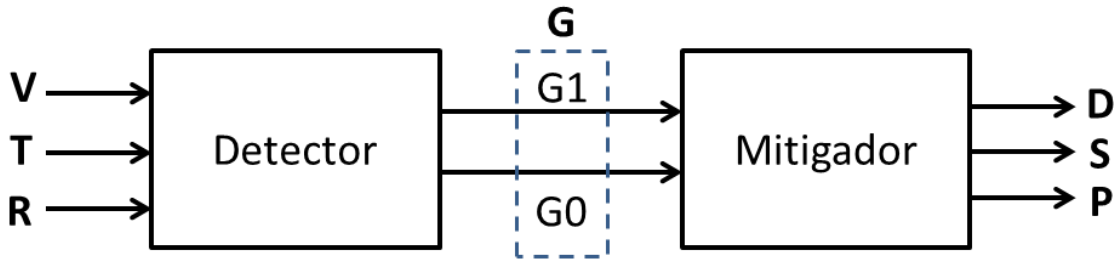
- **Cenário**: projeto de sistema de proteção p/ centrífugas de uma usina nuclear, com 2 módulos



a) Escreva equação algébrica referente às saídas G1 e G0 do módulo **Detector**. Assuma que no projeto:

- G = mínimo(3 , V+T+2*R) (nesta expressão os símbolos + e * referem-se à soma e à multiplicação aritmética)
  - Ou seja: G = soma ponderada da entrada, radiação tendo R peso 2 e saturando o valor em 3 (o máximo representável com 2 bits).
  - **Nenhuma minimização é necessária.**

Exercício (P2-2017) -- RESPOSTA



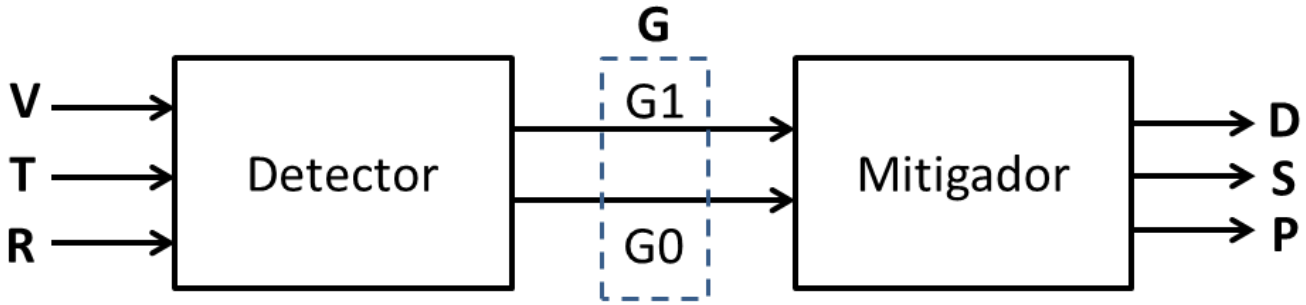
a) Escreva equação algébrica referente às saídas G1 e G0 do módulo **Detector**. Assuma que no projeto:

- G = mínimo(3 , V+T+2*R) (nesta expressão os símbolos + e * referem-se à soma e à multiplicação aritmética)

R	T	V	G1	G0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

G1 = R'•T•V + R•T'•V' + R•T'•V + R•T•V' + R•T•V
= T•V + R
G0 = R'•T'•V + R'•T•V' + R•T'•V + R•T•V' + R•T•V
= T'•V + T•V' + {R•T e/ou RV}
= T⊕V + {R•T e/ou RV}

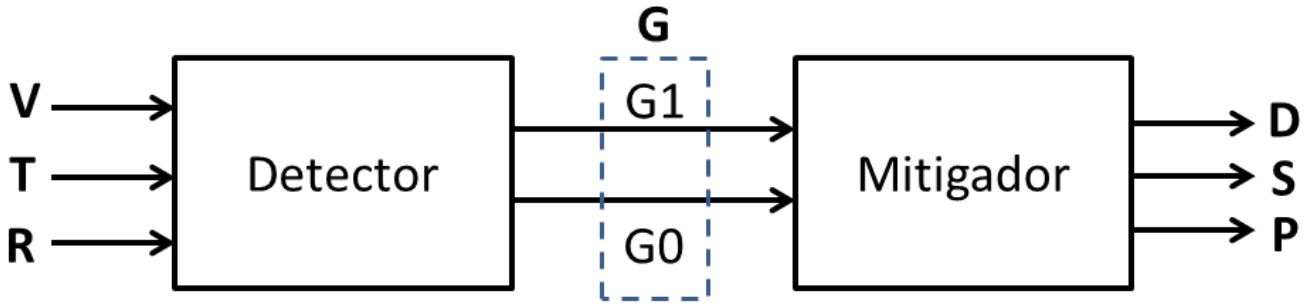
Exercício (P2-2017)



b) Escreva a equação algébrica para as saídas D, S e P do módulo **Mitigador**, em função de G1 e G0. Para isso, assuma que as saídas são assim ativadas (valor lógico 1):

- Painel (P): acende para G ≥ 1;
- Sirene (S): toca para G ≥ 2;
- Sinal de desligamento (D): ativado se G ≥ 3.
- **Nenhuma minimização é necessária.**

Exercício (P2-2017) -- RESPOSTA



b) Painel (P): acende para G ≥ 1; Sirene (S): toca para G ≥ 2; Sinal de desligamento (D): ativado se G ≥ 3.

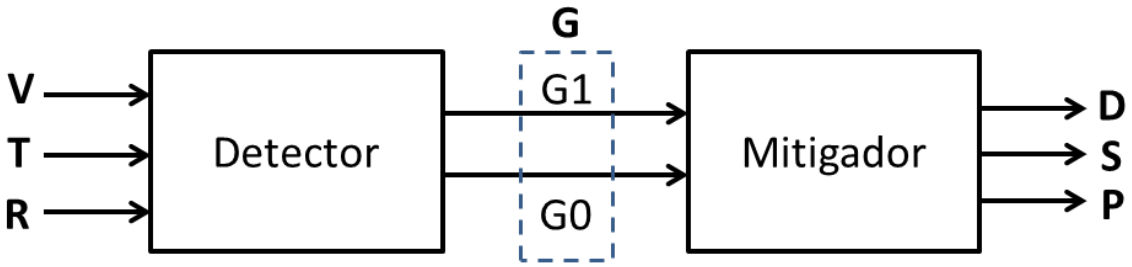
G1	G0	D	S	P
0	0	0	0	0
0	1	0	0	1
1	0	0	1	1
1	1	1	1	1

D = G1 • G0

S = G1 • G0 + G1 • G0’
= G1

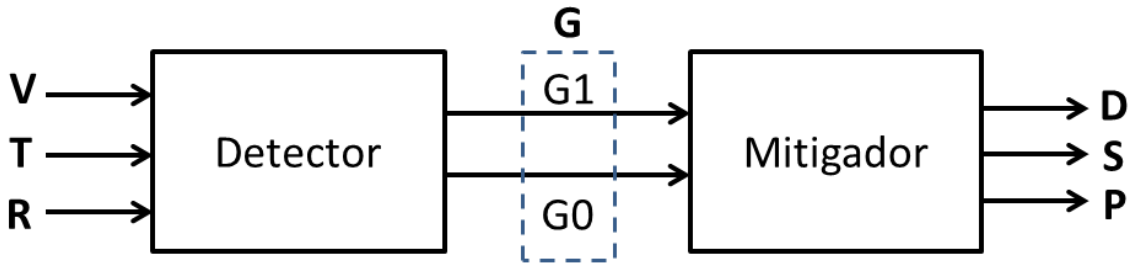
P = G1 • G0 + G1 • G0’ + G1’ • G0
= G1 + G0

Exercício (P2-2017)



c) Se Detector e Mitigador forem implementados como somas de mintermos, qual usará mais portas AND? Explique, assumindo por simplicidade que AND de 2 entradas tem o mesmo custo de AND de 3+ entradas.

Exercício (P2-2017) -- RESPOSTA



c) Se Detector e Mitigador forem implementados como somas de mintermos, qual usará mais portas AND?

R	T	V	G1	G0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

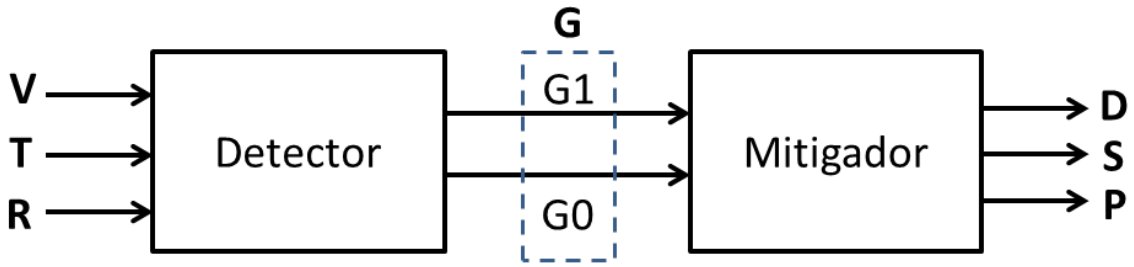
G1	G0	D	S	P
0	0	0	0	0
0	1	0	0	1
1	0	0	1	1
1	1	1	1	1

Detector é mais caro.

EXPL. 1: Detector tem **mais 1s** (são 10) que Mitigador (são 6) ➔ mais mintermos = mais portas AND.

EXPL. 2: Detector requer **7 mintermos** para G1 e G0 (todos menos R'T'V'); Mitigador envolve **3 mintermos** para D, S e P (todos menos G1'G0')

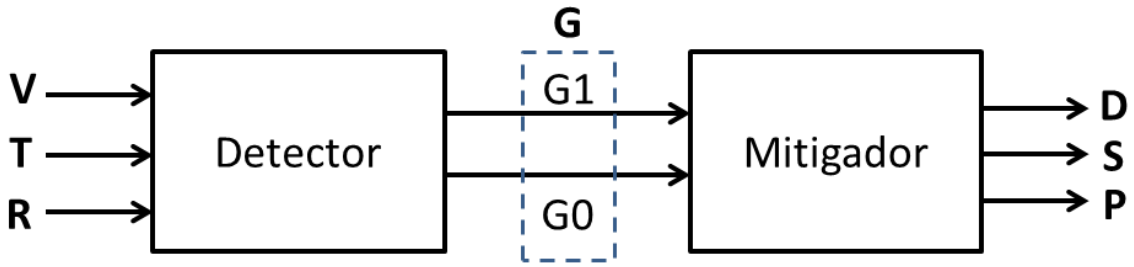
Exercício (P2-2017)



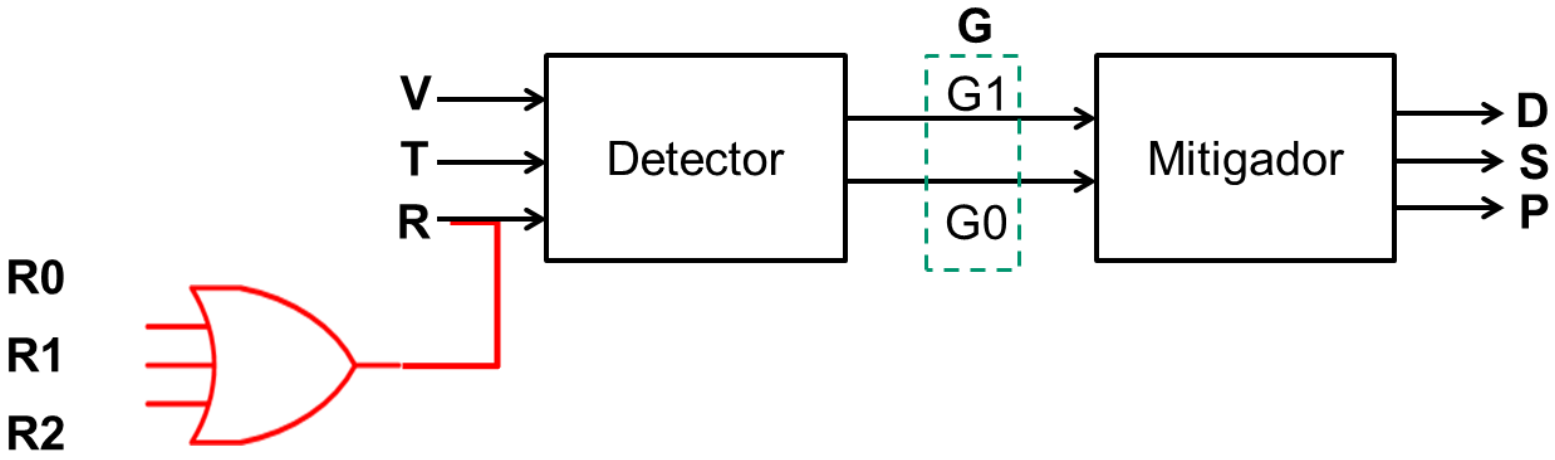
d) A sala monitorada é grande, então é preciso usar não 1, mas 3 sensores de radiação: a entrada R do circuito deve ser ativada quando qualquer dos sensores indicar a presença de radiação.

- Chamando de R0, R1 e R2 a saída desses sensores, seria possível integrá-los no sistema de proteção já projetado sem alterar sua estrutura interna? Explique.

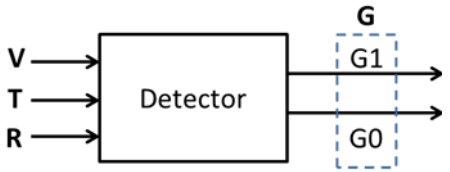
Exercício (P2-2017) -- RESPOSTA



d) Chamando de R0, R1 e R2 a saída desses sensores, seria possível integrá-los no sistema de proteção já projetado sem alterar sua estrutura interna? Explique.



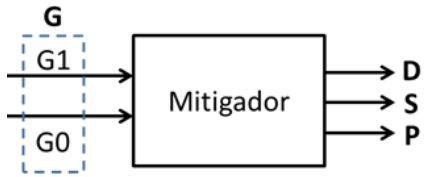
Exercício (P2-2017) – Bônus VHDL




```
library IEEE;
use IEEE.std_logic_1164.all;

entity Detector is
port (V, T, R      : in  std_logic;
      G            : out std_logic_vector(1 downto 0));
end Detector;

architecture archD_v1 of Detector is -- Com portas lógicas
begin
    G <= "00"  when  (R ='0')  and (T ='0') and (V='0')       else
         "01" when ((R = '0') and ((T xor V) = '1'))          else
         "10" when ((R ='0')  and (T ='1') and (V ='1')) or
                   ((R ='1')  and (T ='0') and (V ='0'))      else
         "11" when (R ='1')   and ((T or V) ='1')             else
         "ZZ"; -- "catch all"
end archD_v1;

architecture archD_v2 of Detector is -- Com with-select (multiplexador)
    signal entrada : std_logic_vector (2 downto 0);
begin
    entrada <= (R & T & V); -- monta vetor de bits
    with entrada select
         G <= "00" when "000" ,
              "01" when "001" | "010" ,
              "10" when "011" | "100" ,
              "11" when "101" | "110" | "111",
              "ZZ" when others; -- "catch all"
end archD_v2;
```

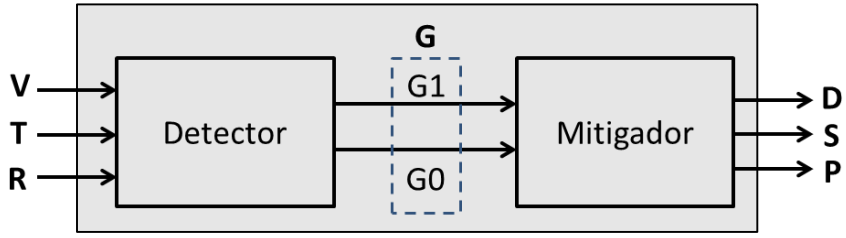
Exercício (P2-2017) – Bônus VHDL

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all; -- suporte a manipulação de inteiros
```




```vhdl
entity Mitigador is
port (G       : in  std_logic_vector(1 downto 0);
      D, S, P : out std_logic);
end Mitigador;

architecture archM_v1 of Mitigador is -- com portas lógicas
Begin
    D <= G(1) and G(0);
    S <= G(1);
    P <= G(1) or G(0);
end archM_v1;

architecture archM_v2 of Mitigador is -- Com comparador (opção mais cara... Aula 13)
begin
    P <= '1' when unsigned(G) >= 1 else '0'; --
    S <= '1' when unsigned(G) >= 2 else '0';
    D <= '1' when unsigned(G) >= 3 else '0';
end archM_v2;
```

Exercício (P2-2017) – Bônus VHDL

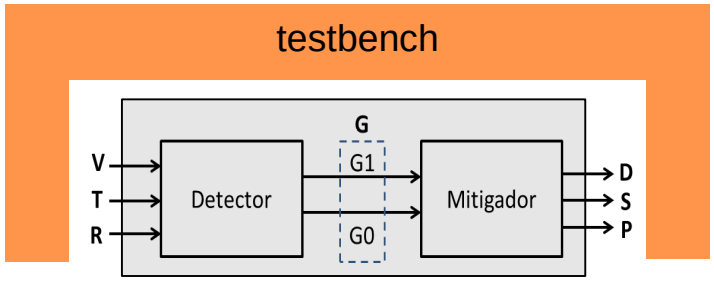


Usando componentes

```
library IEEE;
use IEEE.std_logic_1164.all;

entity SistemaCompleto is
port (V, T, R      : in  std_logic;
      D, S, P : out std_logic);
end SistemaCompleto;

architecture archS of SistemaCompleto is
    component Detector is
         port ( V, T, R      : in  std_logic;
                G            : out std_logic_vector(1 downto 0));
    end component;

    component Mitigador is
         port ( G       : in  std_logic_vector(1 downto 0);
                D, S, P : out std_logic);
    end component;

    signal G: std_logic_vector(1 downto 0); -- sinal interno de ligação

begin
    modulo1: Detector port map (V, T, R, G); -- conecta Detector
    modulo2: Mitigador port map (G, D, S, P); -- conecta Mitigador
end archS;
```

Exercício (P2-2017) – Bônus VHDL

```
-- Testbench for Mitigation+Detection circuit
library IEEE;
use IEEE.std_logic_1164.all;

entity testbench is
   -- empty
end testbench;

architecture tb of testbench is
-- DUT component
component SistemaCompleto is
port (V, T, R      : in  std_logic;
      D, S, P : out std_logic);
end component;

signal VTR_in : std_logic_vector(2 downto 0);
signal DSP_out: std_logic_vector(2 downto 0);

begin
  -- Connect DUT
  DUT: SistemaCompleto port map ( VTR_in(2), VTR_in(1), VTR_in(0),
                                  DSP_out(2), DSP_out(1), DSP_out(0));
  process
  begin
    assert false report "Test init." severity note;

    VTR_in <= "000";
    wait for 1 ns;
    assert (DSP_out="000") report "Fail 000: " severity error;
                          -- ... testes ... --
```

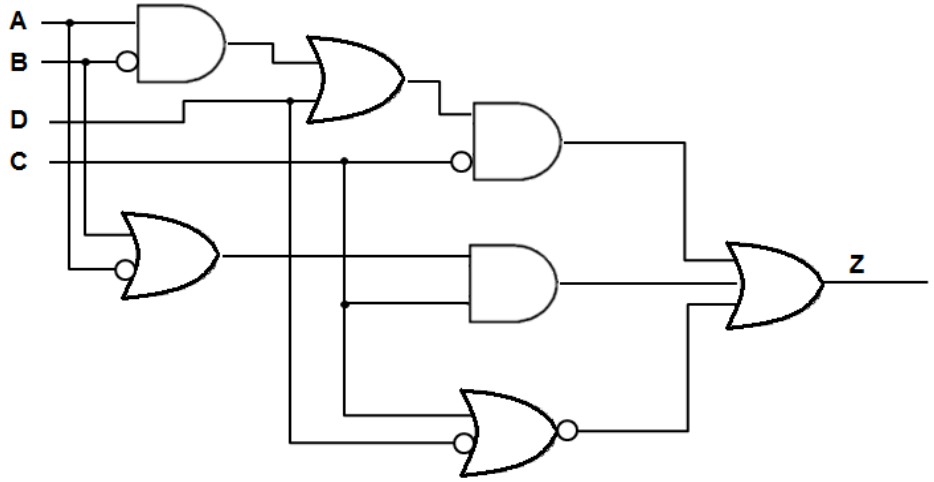


Exercício (P2-2017) – Bônus VHDL

```
process
  begin
                                -- ... testes ... --
    VTR_in <= "001";
    wait for 1 ns;
    assert (DSP_out="011") report "Fail 001: " severity error;

    VTR_in <= "010";
    wait for 1 ns;
    assert (DSP_out="001") report "Fail 010: " severity error;

    VTR_in <= "011";
    wait for 1 ns;
    assert (DSP_out="111") report "Fail 011: " severity error;

    VTR_in <= "101";
    wait for 1 ns;
    assert (DSP_out="111") report "Fail 101: " severity error;

    VTR_in <= "111";
    wait for 1 ns;
    assert (DSP_out="111") report "Fail 111: " severity error;

    assert false report "Test done." severity note;
    wait;
  end process;
end tb;
```

Exercício (P2-2016)

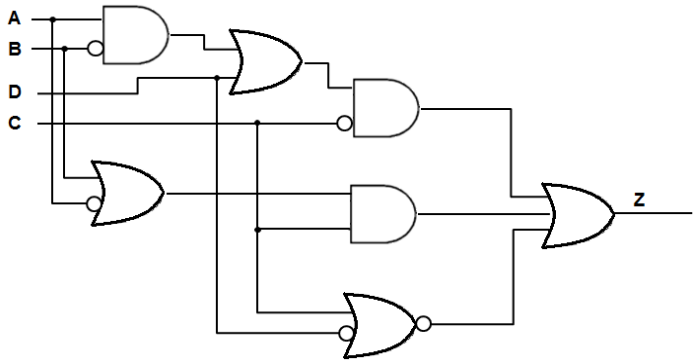
- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



- 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)

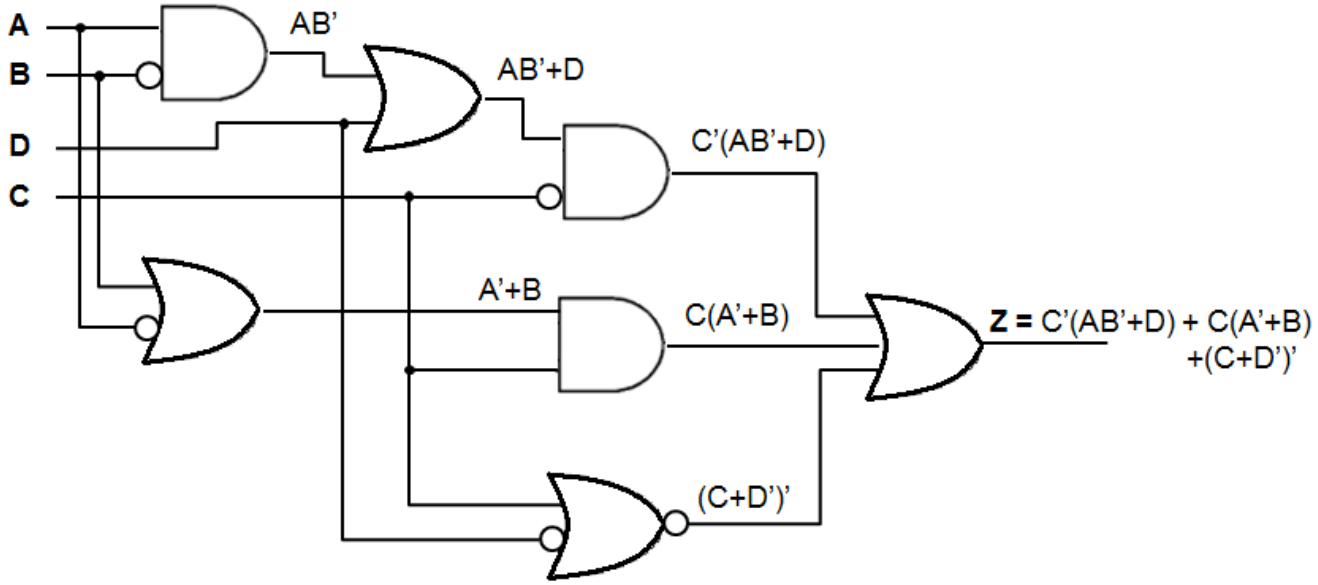
- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



  - 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)

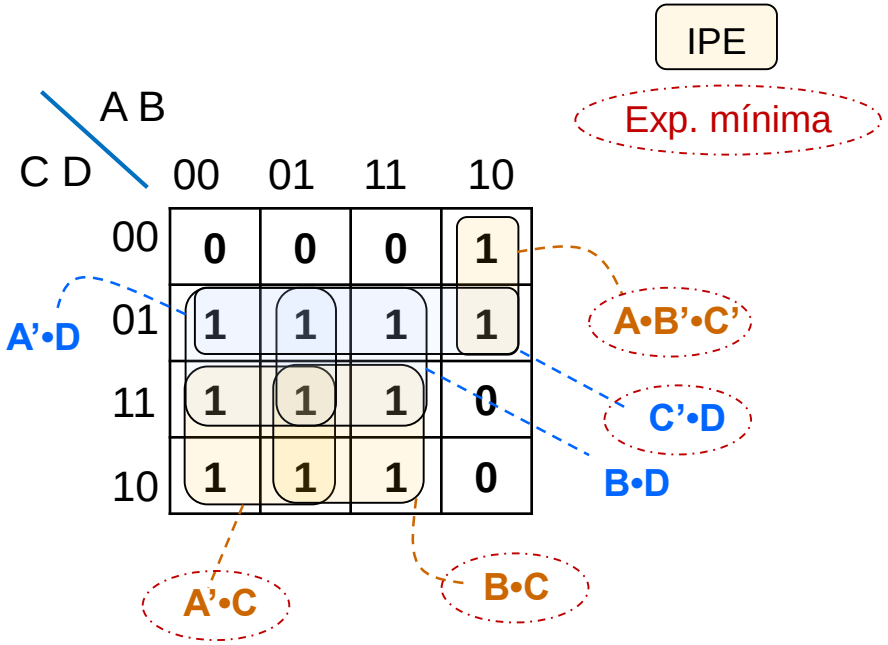
- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



- 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:

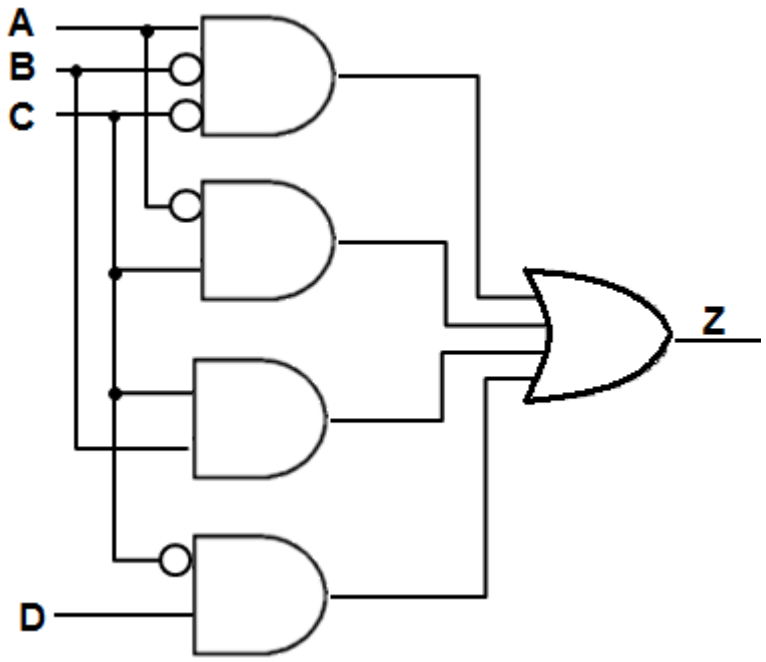


$Z = A'•C + A•B'•C' + B•C + C'•D$

- 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)

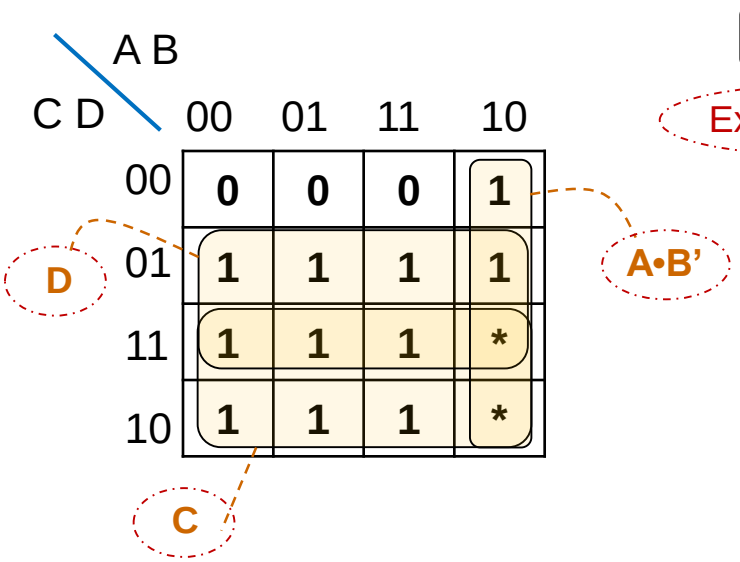
- Você precisa fazer engenharia reversa de um circuito digital, para propor melhorias ao mesmo. Analisando a documentação, você encontra o seguinte diagrama de portas lógicas:



- 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:



IPE

Exp. mínima

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**

Apêndice

Minimização Sistemática:

Método Tabular

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

Apêndice

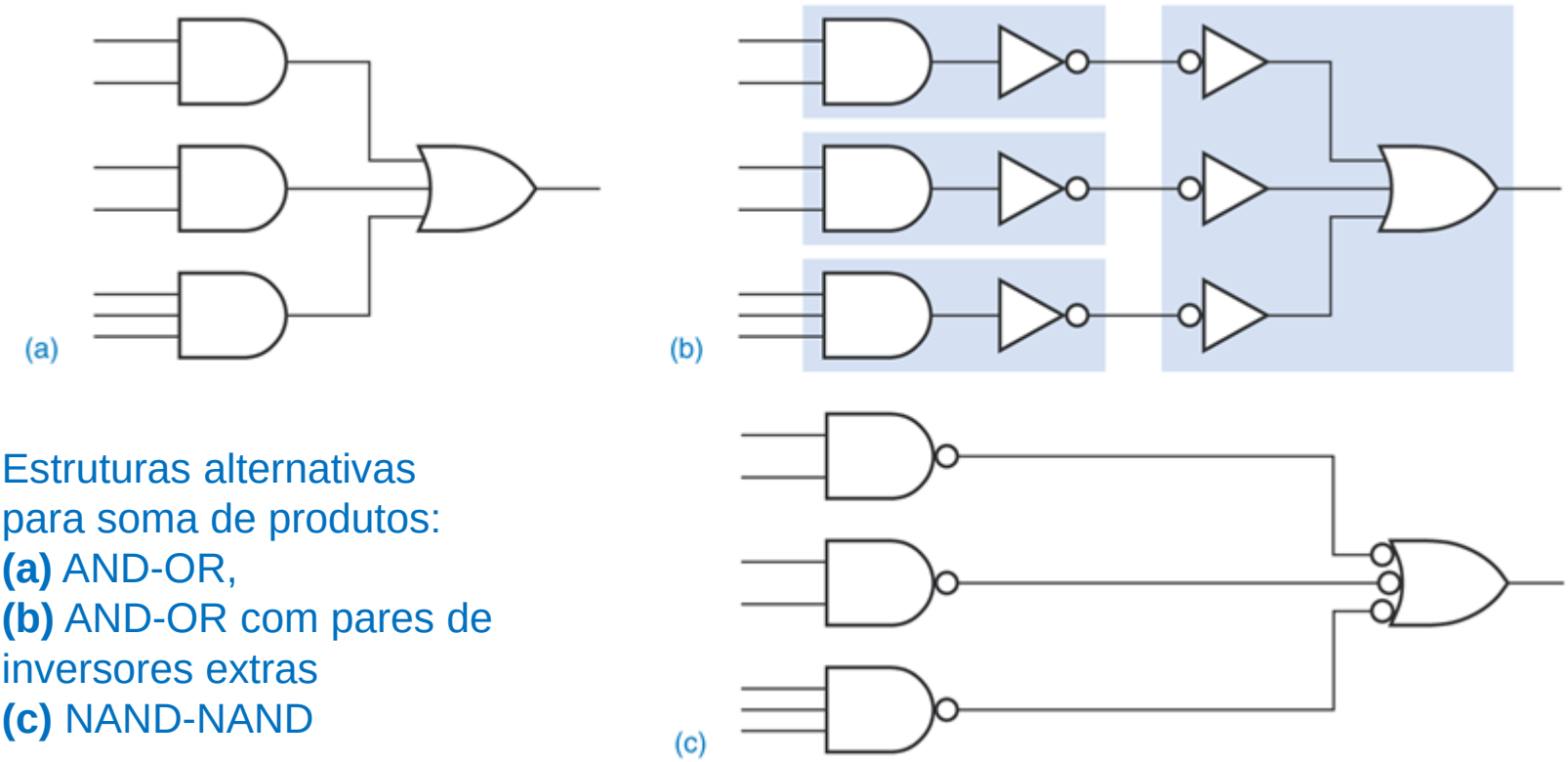
Manipulações em circuitos canônicos

Soma de Produtos

Produto de Somas

Soma de produtos e NANDs

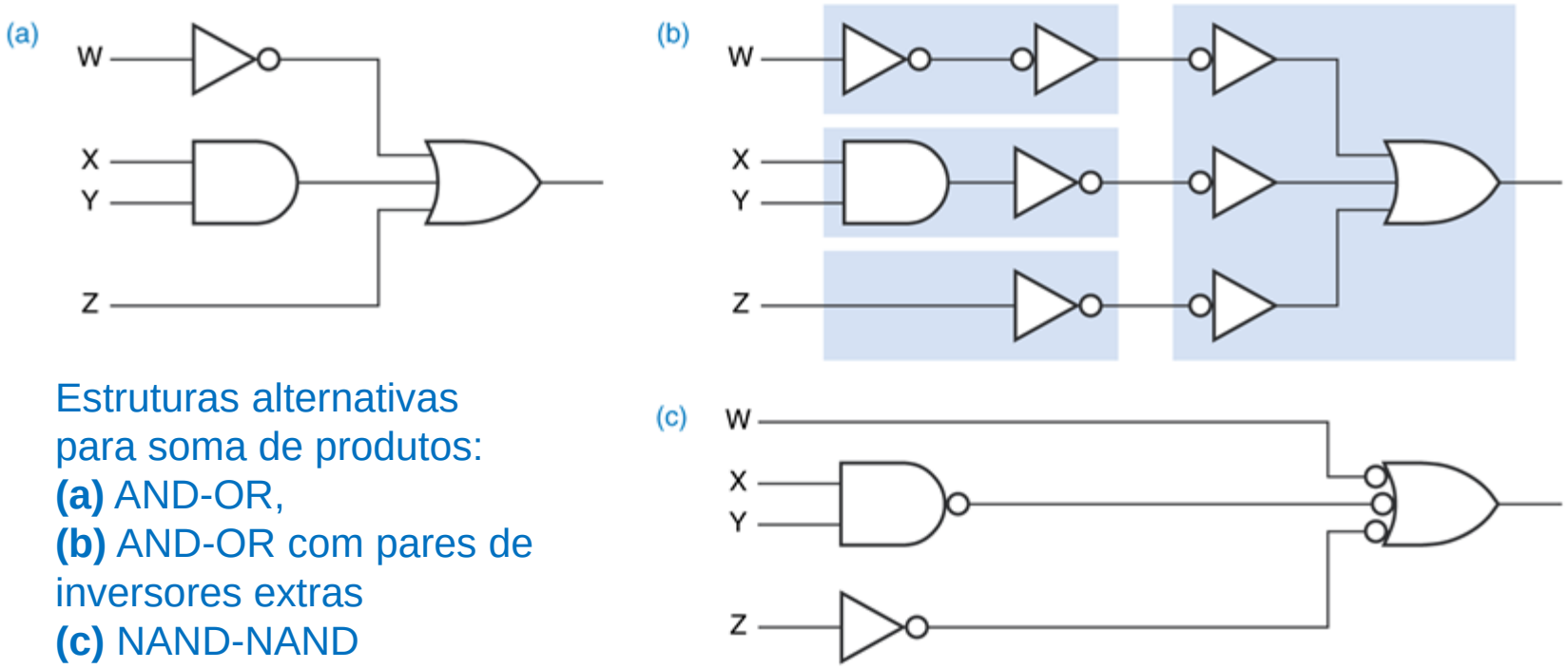
- Conversão de circuito com ANDs/ORs para NANDs: nega-se saída da camada AND e entrada da camada OR. **Exemplo 1:**



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:**

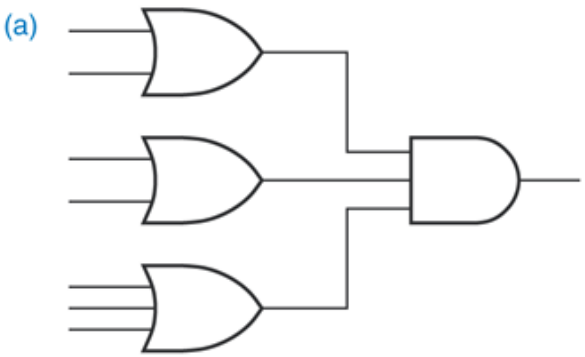


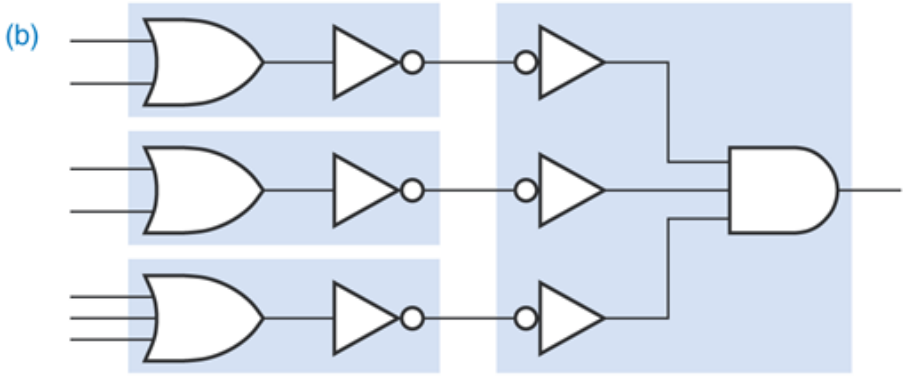
Estruturas alternativas para soma de produtos:
(a) AND-OR,
(b) AND-OR com pares de inversores extras
(c) NAND-NAND

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

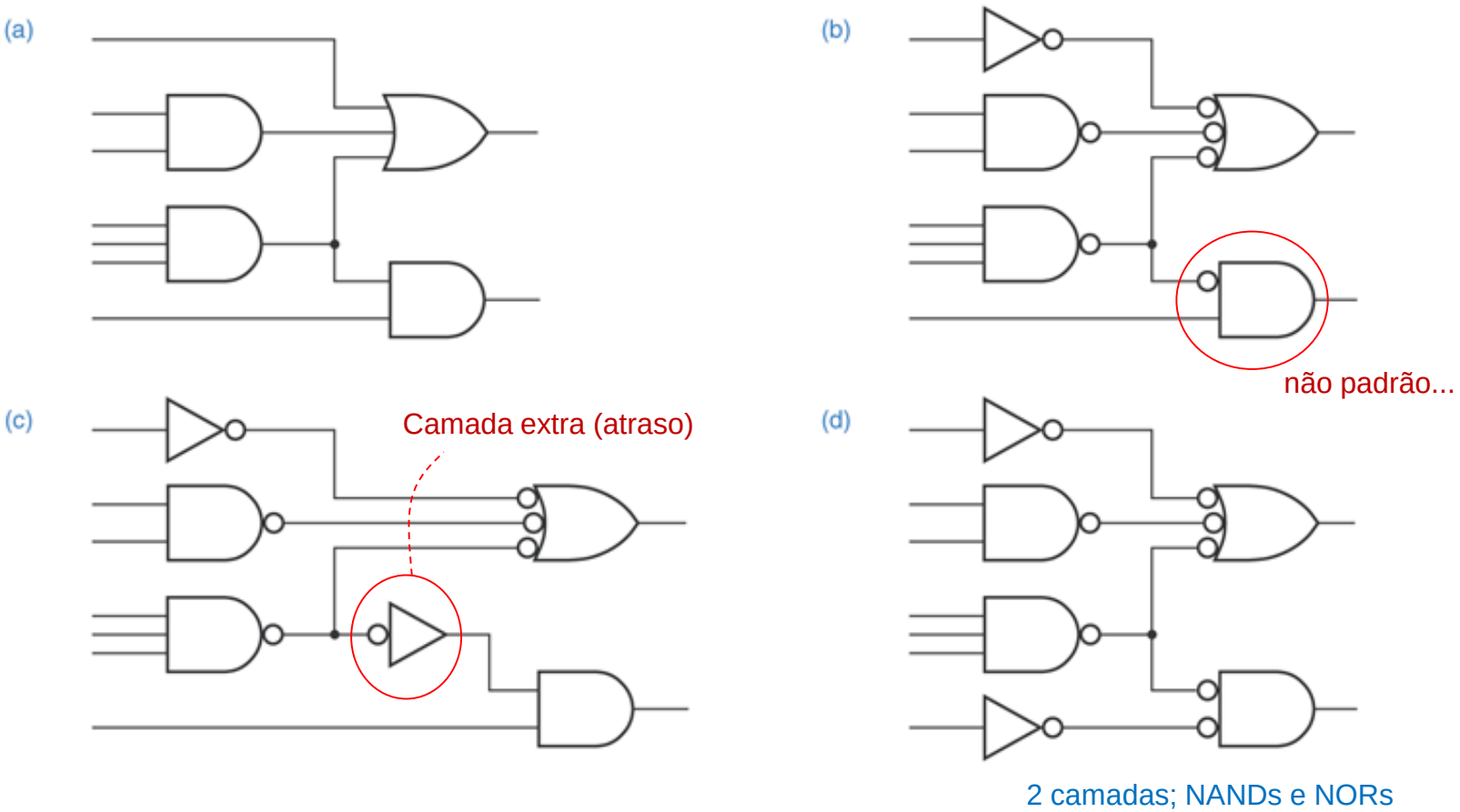


Figure 4-24

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

Apêndice

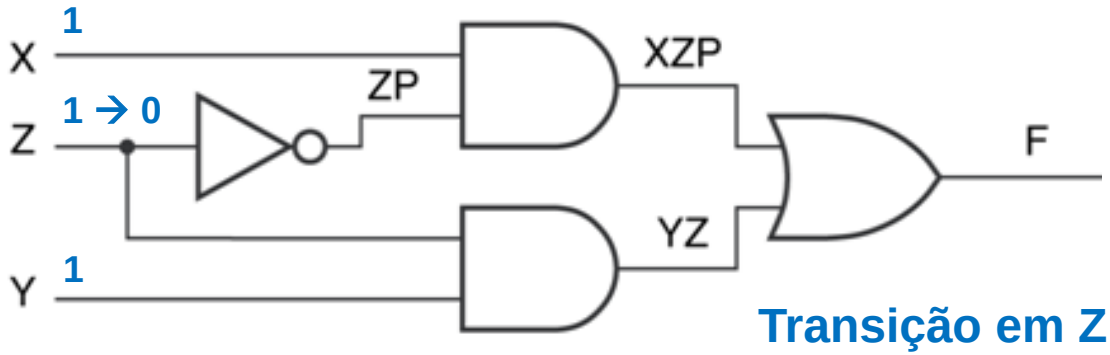
Projeto de circuitos digitais:

Hazards

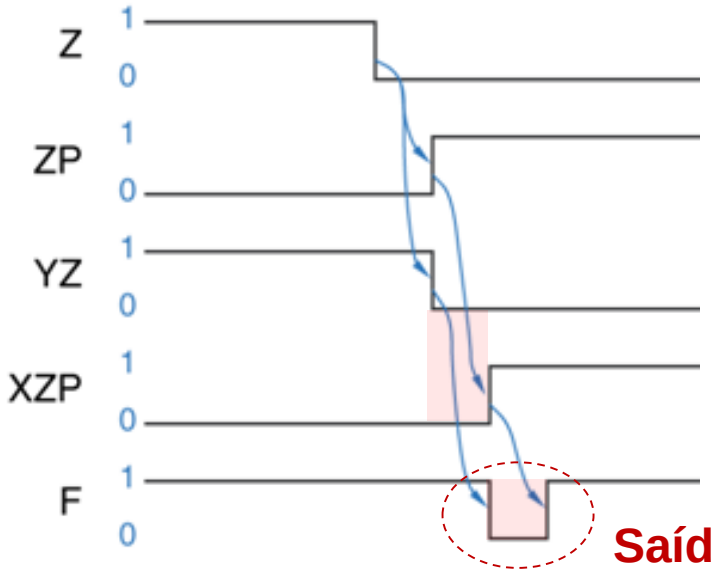
Timing Hazards

- 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

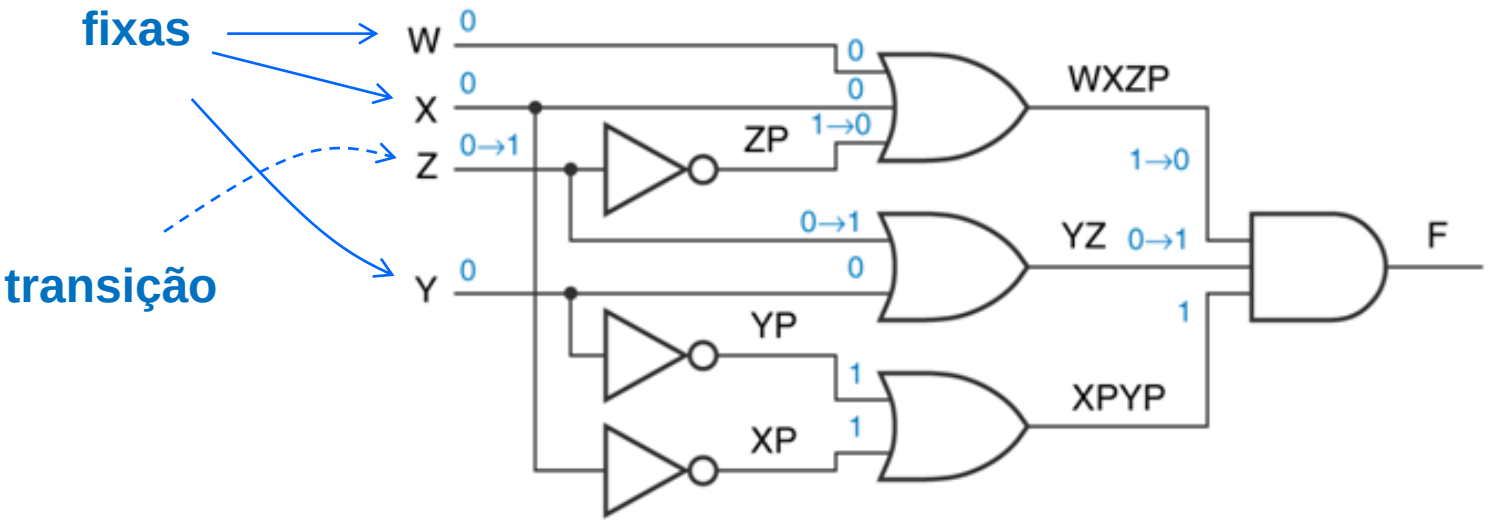


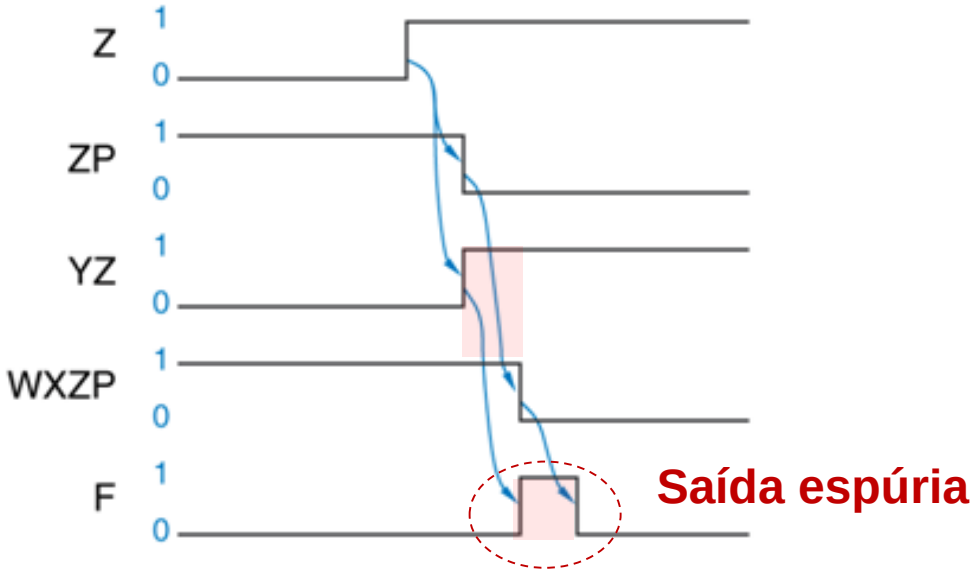
Transição em Z: efeito em YZ mais rápido que em XZP



Saída espúria

Static-0 Hazard

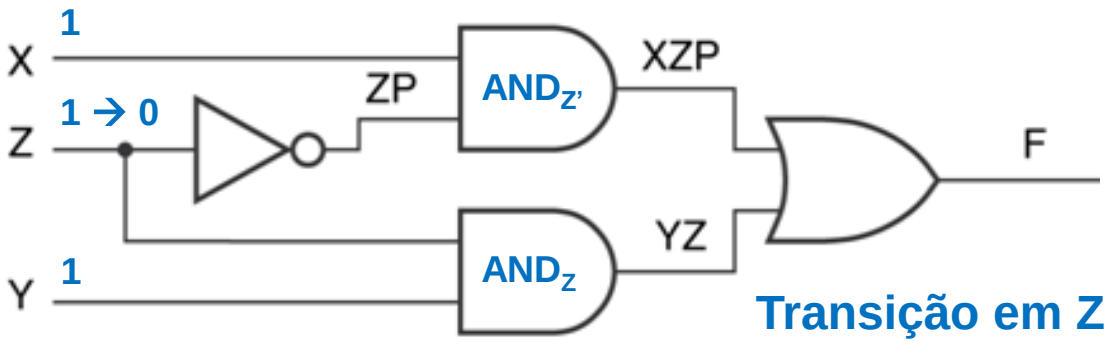




De onde vêm os *hazards*?

- 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



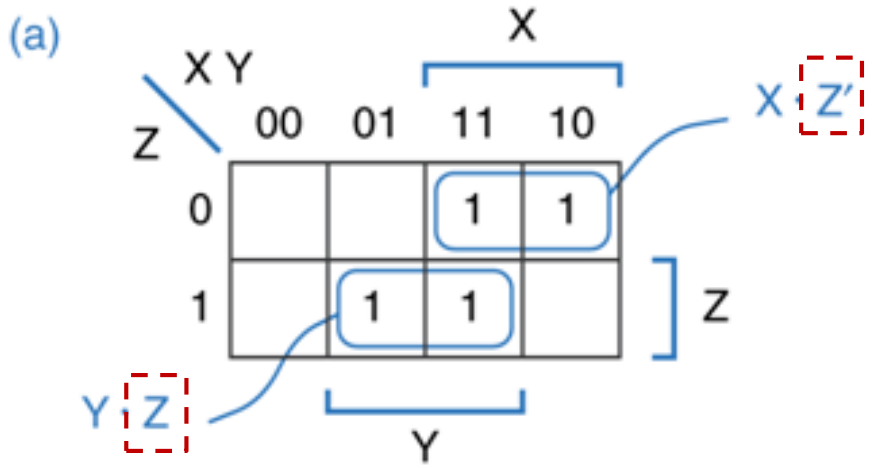
Transição em Z: efeito em YZ mais rápido que em XZP

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 encontrar *hazards* usando mapas?

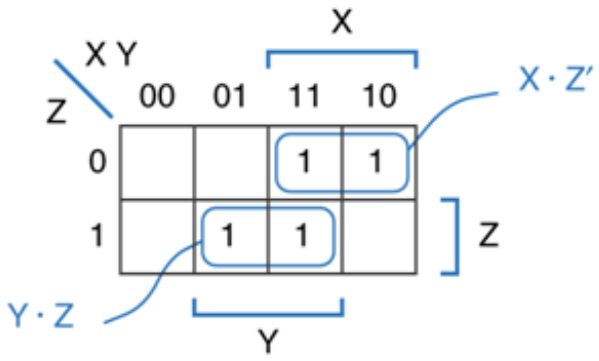
- Observar se IPs incluídos no circuito contêm termo e seu complemento



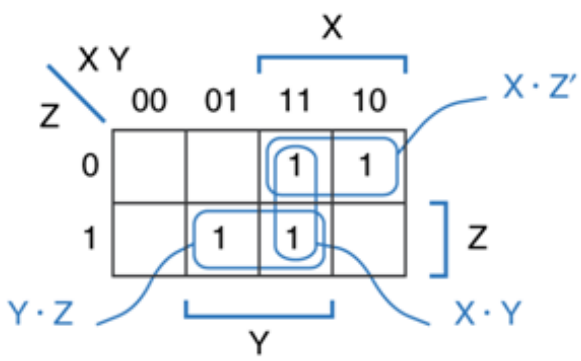
$$F = X \cdot Z' + Y \cdot Z$$

Como resolver *hazards* usando mapas?

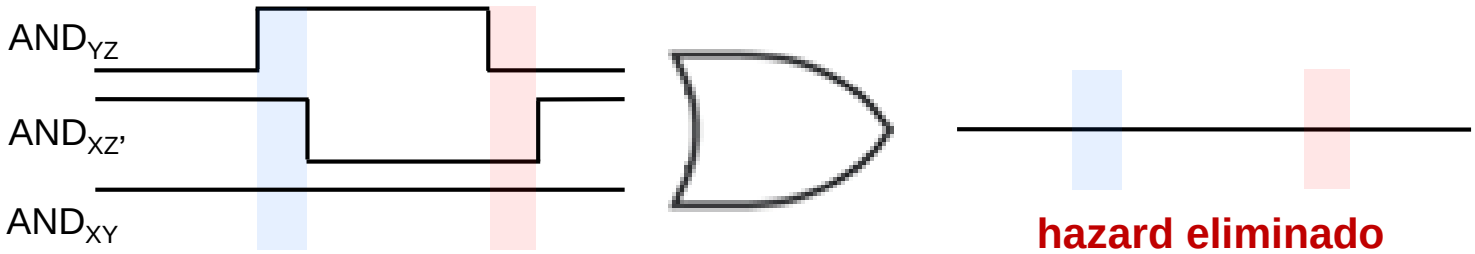
- Incluir IPs extras que cubram os IPs problemáticos
  - Isso equivale ao consenso entre os IPs originais: não é afetado pelo atraso!



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

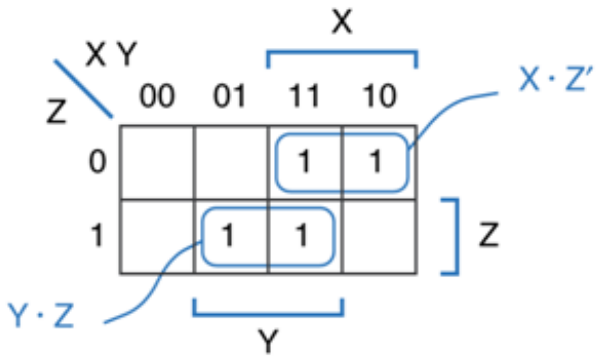


$F = X \cdot Z' + Y \cdot Z + X \cdot Y$

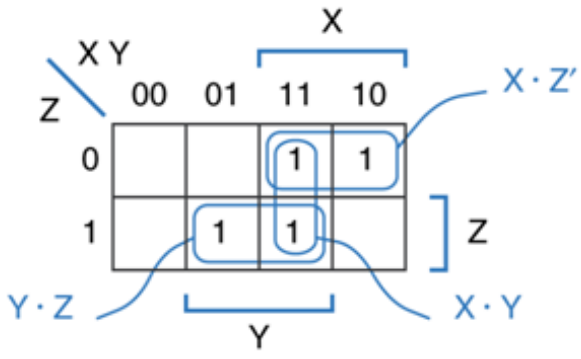


Como resolver *hazards* usando mapas?

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

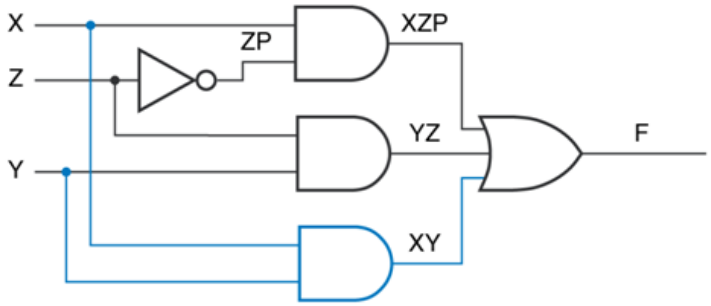


$F = X \cdot Z' + Y \cdot Z$



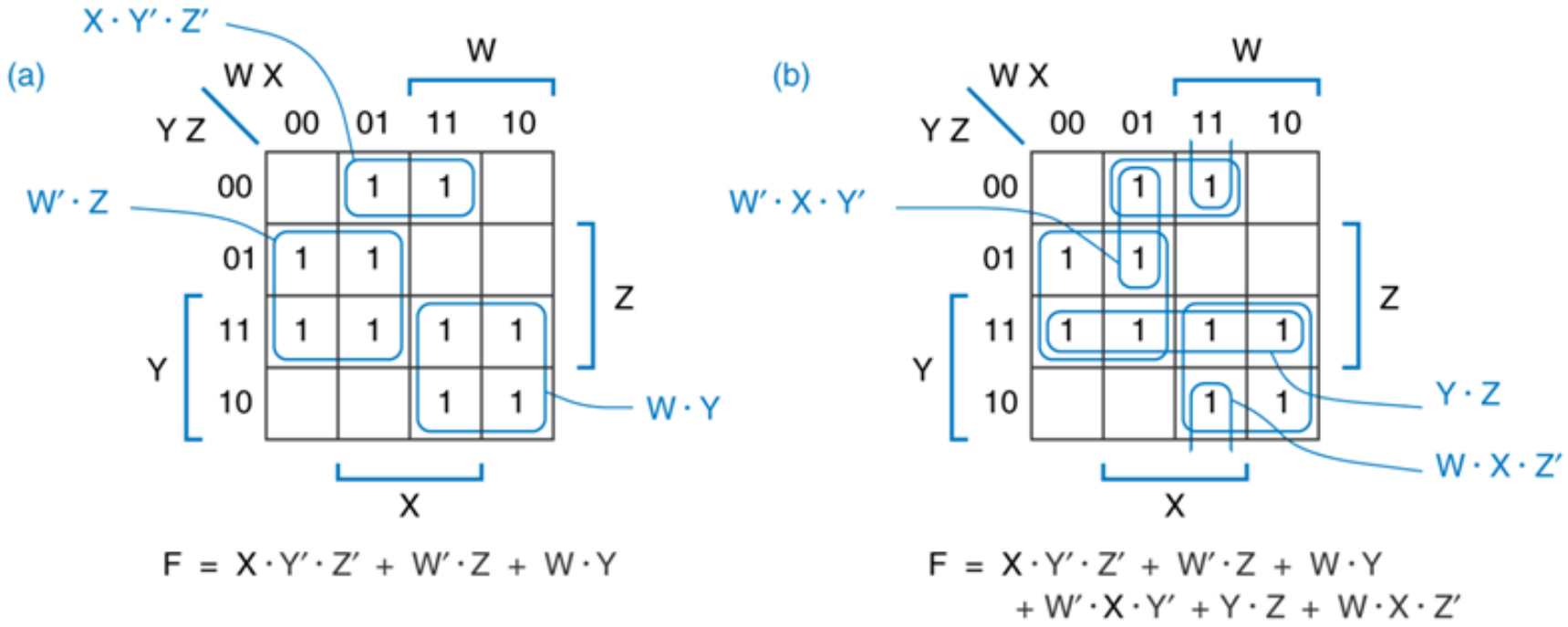
$F = X \cdot Z' + Y \cdot Z + X \cdot Y$

Circuito com hazard eliminado:



Outro exemplo de *static-1 hazard*

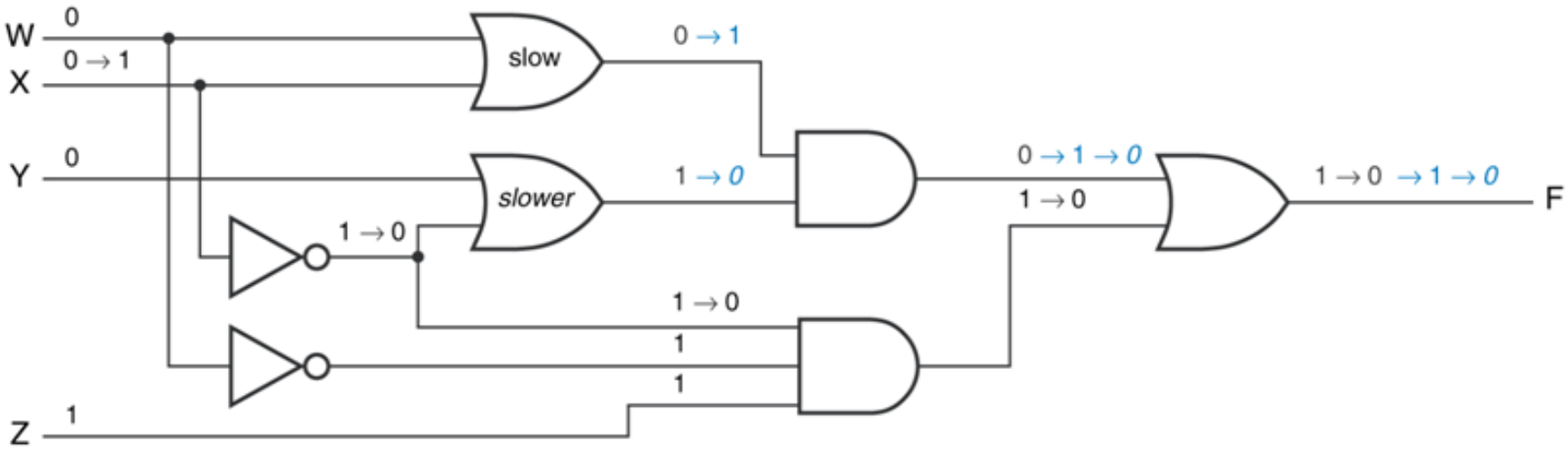
- Incluir IPs extras que cubram os IPs problemáticos
  - Isso equivale ao consenso entre os IPs originais: não é afetado pelo atraso!



(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.