Prototipação de Sistemas Digitais

Síntese Lógica

Cristiano Araújo

Síntese Lógica

- Síntese lógica é o nível de Síntese que serve de ponto entre o que chamamos síntese de Alto-Nível/RTL e a Síntese de Layout(projeto Físico).
- Síntese da estrutura RTL ou equações Booleanas para nível lógica de gates (associada a uma biblioteca de células)
  - Características
    - Reduz significativamente o processo de projeto lógico. Permite otimização tão boa ou melhor que as feitas manualmente pelo projetista.
    - Podemos considerar em geral que a etapa de síntese lógica é é "*independe da tecnologia de implementação do circuito*". Permite mudar tecnologia sem a necessidade de recomeçar todo o processo.
    - Algoritmos de síntese são formulados como independentes de tecnologias específicas (NMOS, CMOS), e estilos de projetos(full-custom, standard-cell). No entanto, estratégias diferentes devem ser combinadas para considerar o máximo de vantagem dos vários tipos de tecnologias e estilos de projeto.

Síntese Lógica

- **Síntese de alto-nível**
- **Síntese RTL**

HDL (VHDL)
Tabelas verdade
Equações lógicas

Independe da Tecnologia

- **Bibliotecas de células**
- **Temporização**
- **Restrições de área**

Síntese Lógica

- **Mapeamento tecnológico(PLA, PLD)**
- **Lógica baseada em células**

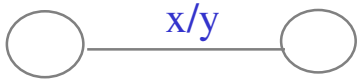
Síntese Lógica

- Síntese do caminho de dados (otimização)
- Síntese do controlador
  - Sistemas de estados finitos
    - Função de transição de estado

      $S(t+\delta t) = f_{\text{próximo estado}}\,(S(t), x(t))$
    - Função de saída

      $y(t) = f_{\text{saída}}\,(S(t), x(t))$
    - Diagrama de estado

      S_i S_j



Síntese Lógica

Diagrama de estados

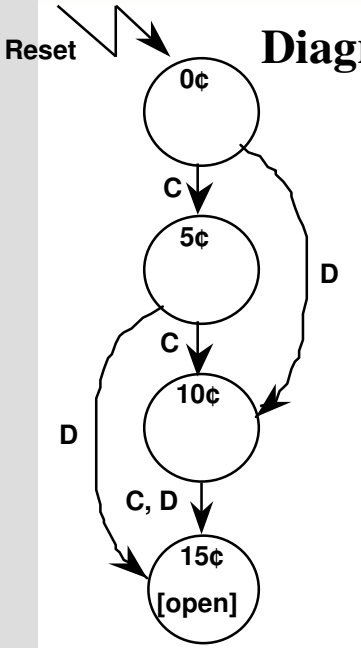
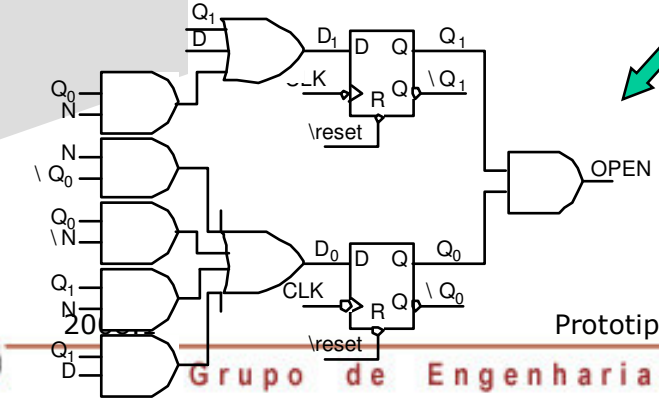


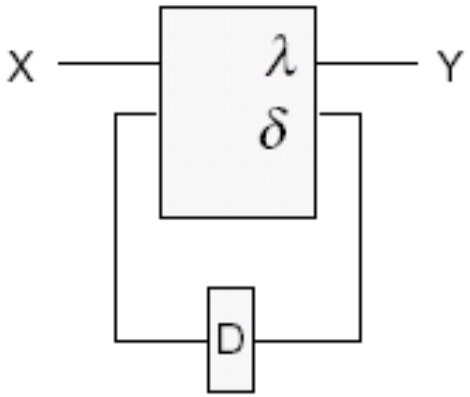
Tabela de estados

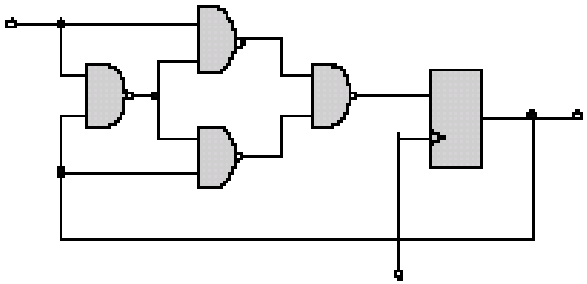
Present State	INPUT D	INPUT C	Next State	Output Open
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	X	X
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	0
	1	1	X	X
10¢	0	0	10¢	0
	0	1	15¢	0
	1	0	15¢	0
	1	1	X	X
15¢	X	X	15¢	1

Diagrama esquemático da FSM

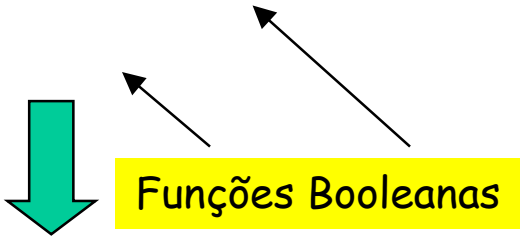


Síntese Lógica





- Dada uma FSM F(X,Y,Z,λ,δ), onde:
  - X - sinais de entrada
  - Y - sinais de saída
  - Z - estados internos da FSM
  - λ - função de próximo estado (XxZ)
  - δ - função saída (XxZ)



- Gerar um circuito C(G,W), onde
  - G : conjunto de componentes (portas lógicas, flip-flops, etc)
  - W: conjunto de fios conectando componentes e G.

Síntese Lógica

- Objetivos
  - **Otimização lógica**
    - Minimização de área do projeto
    - Respeitar temporização do circuito
- Estratégias de síntese
  - Minimização em dois níveis - visa implementação da lógica em dispositivos como PLAs (Programmable Gate Arrays), ou PLDs (Programmable Logic Devices).
  - Minimização multi-nível - visa implementação em tecnologia baseada em células como standard cells, gate array, ou FPGA (Field Programmable Gate Array).

Síntese Lógica

- **Álgebra de Boole como ferramenta matemática para resolução de circuitos combinacionais**
- **Leis e Teoremas da Álgebra Booleana:**
  - *Operações com 0 and 1:*

    1. $X + 0 = X$ **1D. $X \cdot 1 = X$**

    2. $X + 1 = 1$ **2D. $X \cdot 0 = 0$**

  - *Lei da Indepotência:*

    3. $X + X = X$ **3D. $X \cdot X = X$**

  - *Lei da Involução:*

    4. $(\overline{X}) = X$

  - *Lei de Complementação:*

    5. $X + \overline{X} = 1$ **5D. $X \cdot \overline{X} = 0$**

  - *Lei comutativa:*

    6. $X + Y = Y + X$ **6D. $X \cdot Y = Y \cdot X$**

Síntese Lógica

- **Leis Associativas**

7. $(X + Y) + Z = X + (Y + Z) = X + Y + Z$	7D. $(X \cdot Y) \cdot Z = X \ (Y \cdot Z) = X \cdot Y \cdot Z$
8. $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$	8D. $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$
9. $X \cdot Y + X \cdot \overline{Y} = X$	9D. $(X + Y) \cdot (X + \overline{Y}) = X$
10. $X + X \cdot Y = X$	10D. $X \cdot (X + Y) = X$
11. $(X + \overline{Y}) \cdot Y = X \cdot Y$	11D. $(X \cdot \overline{Y}) + Y = X + Y$

- ***Lei de DeMorgan:***
  1. $\overline{(X + Y + Z + \ldots)} = \overline{X} \cdot \overline{Y} \cdot \overline{Z} \cdot \ldots$ 1D. $\overline{(X \cdot Y \cdot Z \cdot \ldots)} = \overline{X} + \overline{Y} + \overline{Z} + \ldots$
  2. $\{F(X1,X2,\ldots,Xn,0,1,+,\cdot)\}' = \{F(\overline{X1},\overline{X2},\ldots,\overline{Xn},1,0,\cdot,+)\}$
-

Síntese Lógica

- ***Lei de DeMorgan:***

  1. $\overline{(X + Y + Z + \ldots)} = \bar{X} \cdot \bar{Y} \cdot \bar{Z} \cdot \ldots$ 1D. $\overline{(X \cdot Y \cdot Z \cdot \ldots)} = \bar{X} + \bar{Y} + \bar{Z} + \ldots$
  2. $\{F(X1,X2,\ldots,Xn,0,1,+,\cdot)\}' = \{F(\bar{X}1,\bar{X}2,\ldots,\bar{X}n,1,0,\cdot,+)\}$

- ***Dualidade:***

  1. $(X + Y + Z + \ldots)^D = X \cdot Y \cdot Z \cdot \ldots$ 1D. $(X \cdot FY \cdot Z \cdot \ldots)^D = X + Y + Z + \ldots$
  2. $\{F(X1,X2,\ldots,Xn,0,1,+,\cdot)\}^D = \{F(X1,X2,\ldots,Xn,1,0,\cdot,+)\}$

- ***Teoremas para Multiplicação e Fatoração:***

  1. $(X + Y) \cdot (\bar{X} + Z) = X \cdot Z + \bar{X} \cdot Y$ 1D. $X \cdot Y + \bar{X} \cdot Z = (X + Z) \cdot (\bar{X} + Y)$

- ***Teorema do Consenso:***

  1. $(X \cdot Y) + (Y \cdot Z) + (\bar{X} \cdot Z) = X \cdot Y + \bar{X} \cdot Z$

  1D. $(X + Y) \cdot (Y + Z) \cdot (\bar{X} + Z) = (X + Y) \cdot (X + Z)$

Síntese Lógica

- **Estratégia**
  - **Minimização em Dois níveis**
  - **Minimização Multi-nível**
- **Definições**
  - Função Booleana completamente Especificada
    - f: $B^n \rightarrow B^m$ com B:={0,1} e n, m ε N

      No caso especial em que m=1, saída simples f: $B^n \rightarrow B$ e é dito ser uma função de saída simples.

    - ON-SET
      - $F_k := \{ x \varepsilon \{0,1\}^n \mid f_k(x)=1\}$. Contém todas as combinações de entrada para a qual a saída da função é definida como 1 ε B.
    - OFF-SET
      - $F'_k := \{ x \varepsilon \{0,1\}^n \mid f_k(x)=0\}$. Contém todas as combinações de entrada para a qual a saída da função é definida como 0 ε B.

- **Função completamente especificada**
  - As seguintes declarações são válidas para on(1) e off(0) para funções completamente especificadas:
    - $F_k \cup F'_k = B^n \quad k= 1, \ldots, m$
    - $F_k \cap F'_k = \phi \quad k= 1, \ldots, m$

Síntese Lógica

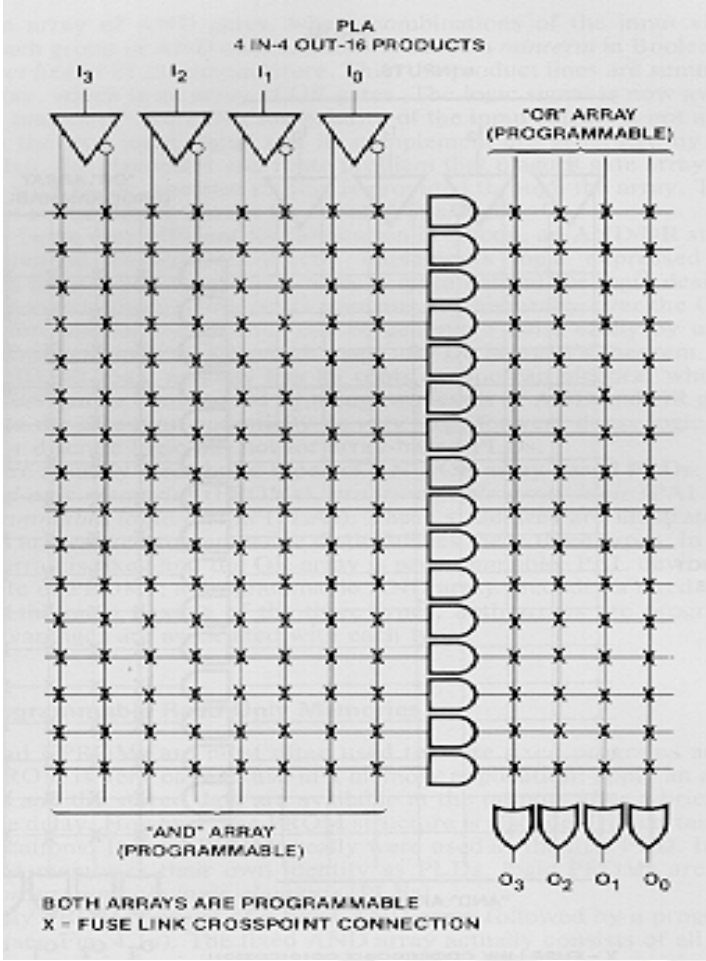
- **Função não completamente especificada**
  - Uma função f_k, k = 1, ……,m que mapea pontos x ε {0,1} sobre os valores $f_k(x)$ ε {0,1}, mas cujo valor é indefinido por algum argumento é dita ser uma "*Função booleana não completamente especificada*".
  - Os valores indefinidos, " Don't-Care: D_k "são indicados por um "x" ou "-"e pode ser interpretado como valores correspondente a entradas que nunca aparecerão na aplicação.
  - Durante o processo de minimização estes valores podem ser arbitrariamente adicionados ao conjunto dos '1s' ou dos '0s' com o intuito de otimizar a função.
  - Funções booleanas mapeam do conjunto $\{0,1\}^n$ sobre o conjunto $\{0,1,-\}^m$
  - $F_k \cup F'_k \cup D_k = \{0,1\}^n$ e
  - $F_k \cap F'_k = \phi$, $F_k \cap D_k = \phi$, $F'_k \cap D_k = \phi$

Minimização para Lógica de Dois Níveis

- ***Objetivo***
  - ***Minimização da área para uma lógica de soma de produto***

- ***Minimização em dois níveis:*** $f_1(x_1,x_2,x_3) = x_1x_3 + x_2x_3$

  $f_2(x_1,x_2,x_3) = x_2x_3 + x_1x_2$



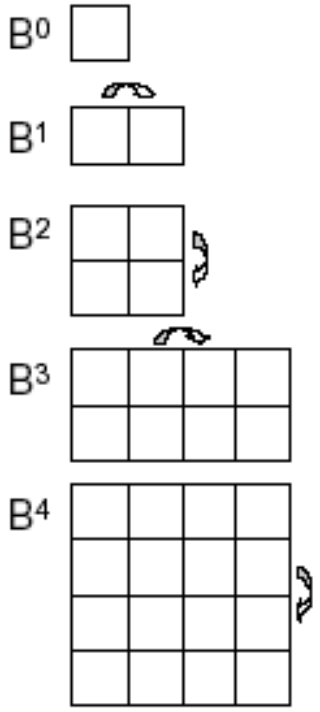
Síntese Lógica (Dois níveis)

- **Representação de funções Booleanas**
  - Tabela Verdade
  - Soma de Produtos
  - Produtos de Soma
  - Matrizes
  - Mapa de Karnaugh
  - Representação usando n-cubos (n-cubes)
  - BBD (Binary Decision Diagram)
  - Redes Booleanas
  - Quine-MacMcCluskey
  - Espresso

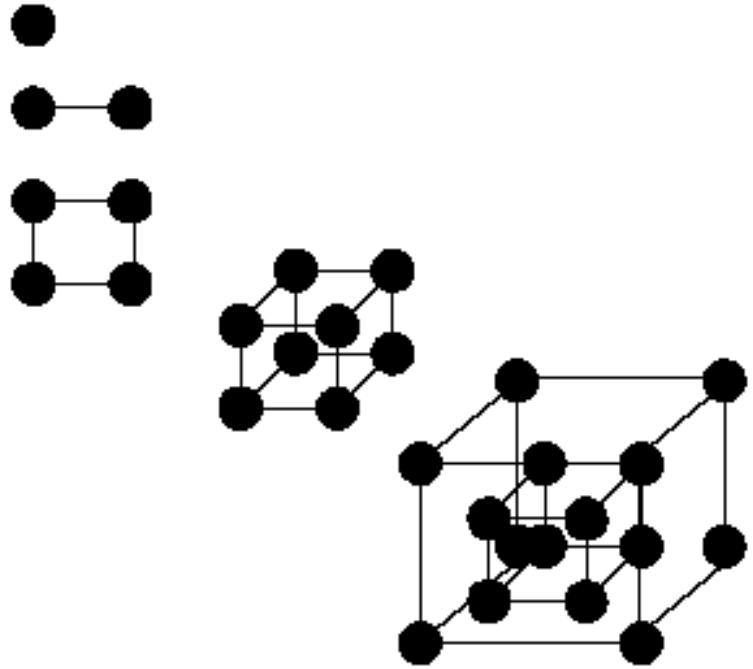
Espaço Booleano

- $B = \{0,1\}$
- $B^2 = \{0,1\} \times \{0,1\} = \{00, 01, 10, 11\}$

Karnaugh Maps:

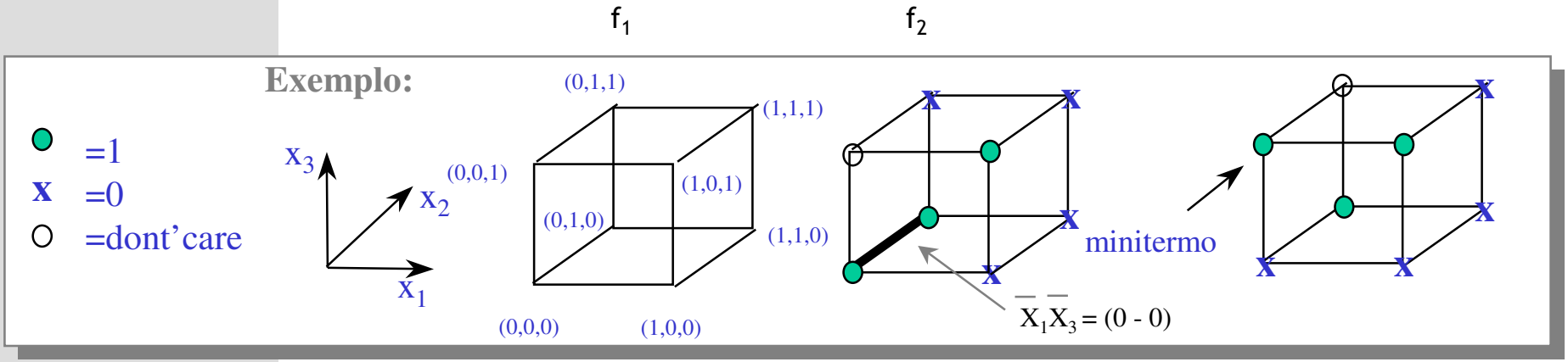


Boolean Cubes:



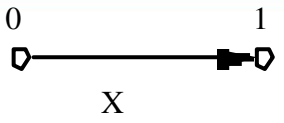
Síntese Lógica (Minimização usando cubos)

- **Representação de funções Booleanas**
  - Uma função booleana de n variáveis pode ser visualizada como um conjunto de vértices de um cubo n-dimensional (n-cubo).
  - Cada vértice representa uma das 2^n possíveis entradas combinacionais de B^n e é marcada (representação geométrica) como pertencente ao conjunto dos '1s', dos '0s'ou dont's care.

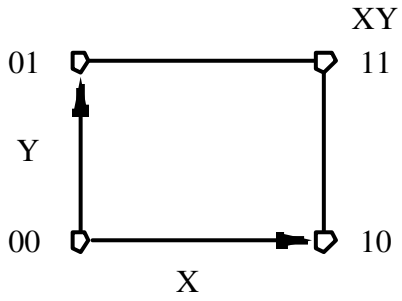


  - No exemplo acima considere a representação de duas funções f_1 e f_2 de três entradas (x_1, x_2, x_3) não completamente especificadas.
  - O termo produto x_1x_3 cobre os dois minitermos $x_1x_2x_3$ e $x_1x_2x_3$

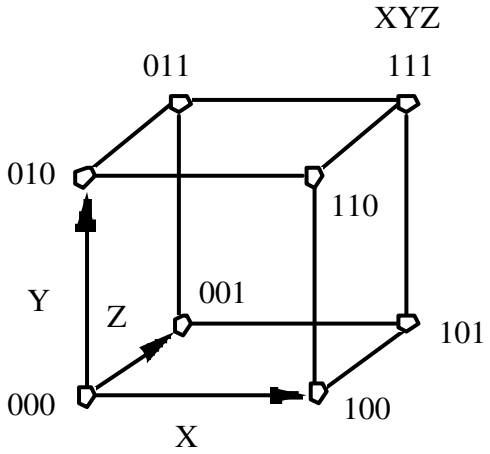
Síntese Lógica (Minimização usando cubos)



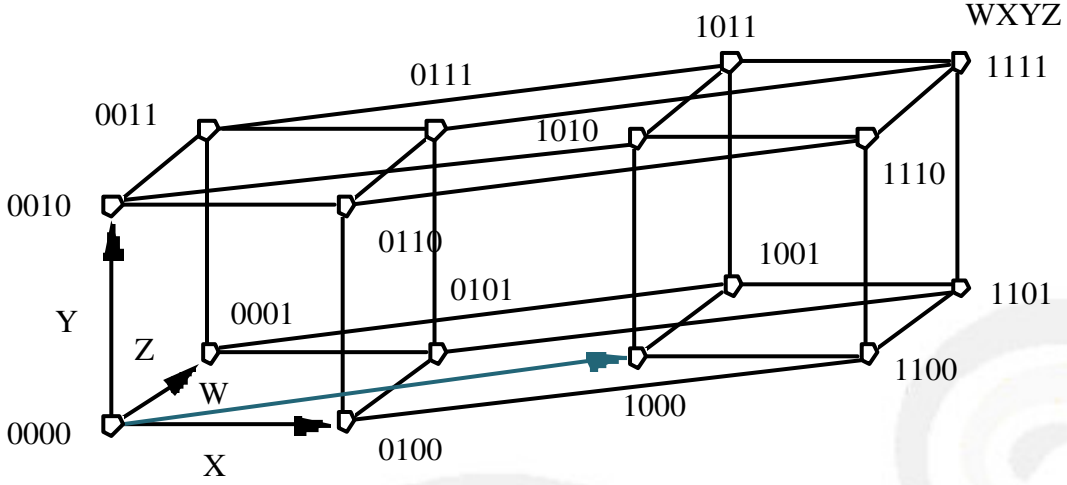
1-cube



2-cube

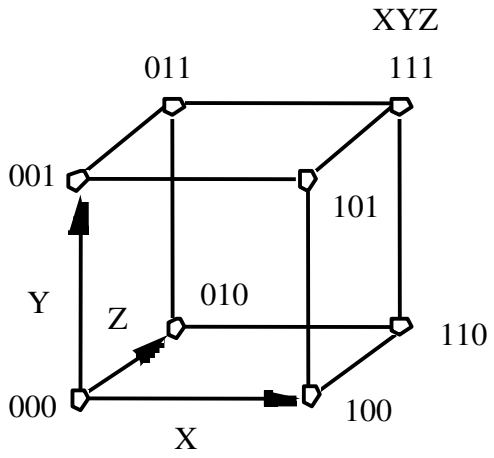


3-cube



Exemplo (Cubo Triplo - três variáveis)

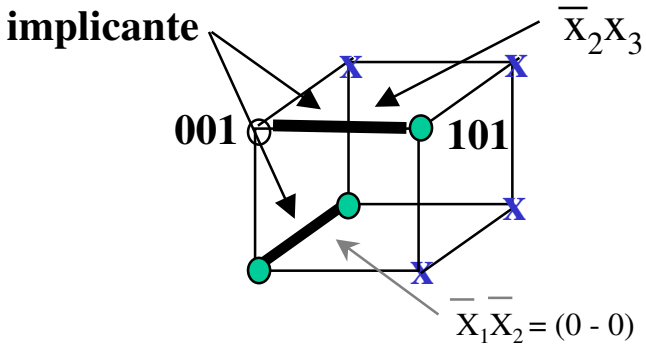
- Esta forma de representar funções booleanas pode ser muito mais compacta que a soma de minitermos. Nesta metodologia cada termo cobre exatamente um vértice.
- Assim, a idéia é selecionar um conjunto mínimo de sub-cubos para cobrir o conjunto dos '1s', e possivelmente parte do conjunto dos don't cares. Isto é exatamente a intenção de otimização lógica de dois níveis.



3-cube

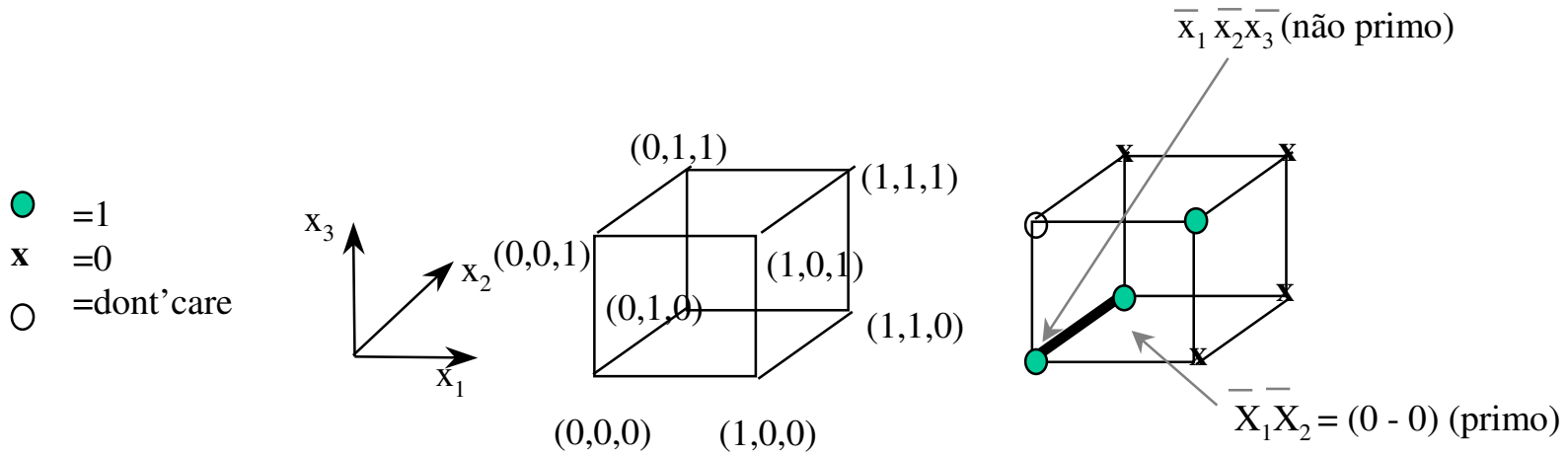
Síntese Lógica (Minimização usando cubos)

- **Definições:**
  - *Implicante* - **é o cubo *c* de uma função f_k que está contido em F_k U D_k. Exemplo *c* = (101) e (001) em f_1.**
  - *Cubo ou implicante primo* **é um implicante que não pode ser combinado com outro para eliminar um literal. Isto significa dizer que *c* não está contido em qualquer outro implicante de f_k, ou, um implicante que não pode ser combinado com outro para eliminar um literal.**



No exemplo ao lado o produto $x_1\bar{x}_2x_3$ não é primo, desde que o minitermo está contido no implicante $\bar{x}_2x_3$. No entanto, o termo $\bar{x}_2x_3$ é primo .

Síntese Lógica



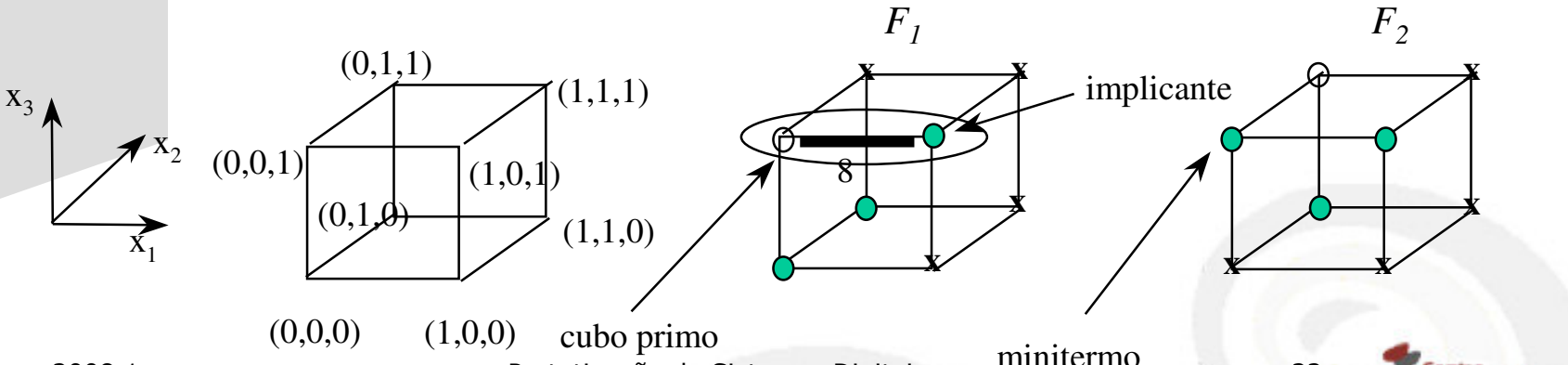
• Dizemos que um conjunto C = {c_1, c_2... , c_k} de cubos cobrem uma função *f* se todos os cubos contém todos os vértices correspondentes '1' ('1 ' e don't cares) e nenhum vértice correspondente aos '0'.

Síntese Lógica

- **Representação usando n-cubes**

Existem 2^n termos produto diferentes (minitermos) com n variáveis correspondendo a 2^n vértices de um n-cubo

- $F_1(x_1,x_2,x_3) = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2x_3$ $D_1 = \bar{x}_1\bar{x}_2x_3$
- $F_2(x_1,x_2,x_3) = x_1\bar{x}_2x_3 + x_1x_2x_3 + \bar{x}_1x_2x_3$ $D_1 = \bar{x}_1x_2x_3$
- *Onde :*
  - *f(x) = 1*
  - x *f(x) = 0*
  - *f(x) = Don't Care*



Síntese Lógica

- Um conjunto C = {c_1, c_2, c_3,....., c_n} de cubos é dito ser uma cobertura de uma função booleana f, se os cubos contêm todos os vértices correspondentes a 1's e nenhum vértice correpondente a 0's da função booleana f. Ou seja, a cobertura representa a união dos 1's de f e possivelmente alguns pontos de don't care de f.
- **Estratégia**
  - **Minimização lógica de dois níveis.**
    - O objetivo principal é obter uma área mínima de implementação da função booleana na forma de uma PLA. A função Booleana neste caso é representada por uma soma de produtos, implementáveis numa PLA.
    - É suficiente realizar a cobertura de f (função) onde o conjunto *D* de don't cares possam ser usados para minimização.
      - PLAs - Programmable Logic Arrays
      - PLDs - Programmable Logic Devices

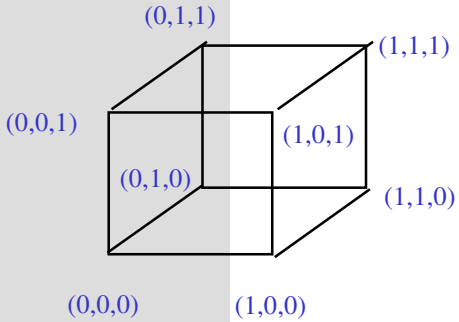
Síntese Lógica

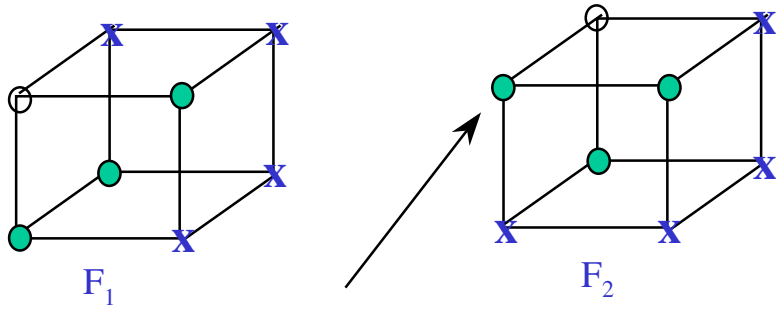
- **Minimização de dois níveis usando a forma n-cubos**
  - Minimizar o número de minitermos
  - **Mecanismos:**
    - Detectar cubos primos ou cubos primos múltiplos , considerando os cubos do conjunto don't cares e deletar cubos pequenos já cobertos.
    - Remover cubos reduntantes

Síntese Lógica

- **Exemplo:**

- Implementar as funções F_1 e F_2 conforme figura abaixo em uma PLA após minimização





conjunto dos termos '1's

x_1	x_2	x_3	f_1	f_2
0	0	0	1	0
0	0	1	0	1
0	1	0	1	1
1	0	1	1	1

conjunto dos don't cares

x_1	x_2	x_3	f_1	f_2
0	0	1	1	0
0	1	1	0	1

Na soma de produtos as saídas das duas funções são inicialmente:

$F_1(x_1,x_2,x_3) = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2x_3$ $\quad d_1 = \bar{x}_1\bar{x}_2x_3$

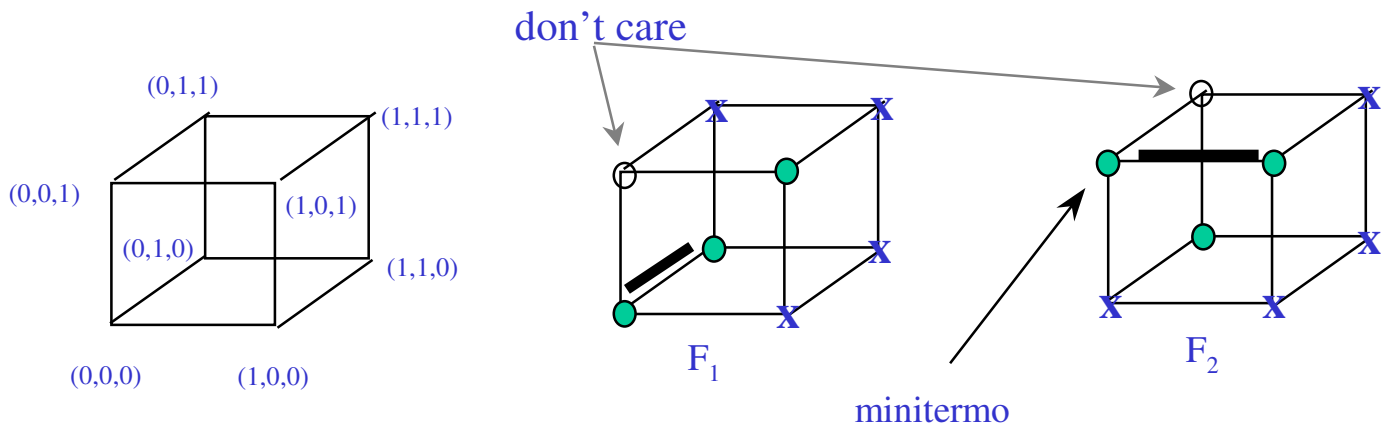
$F_2(x_1,x_2,x_3) = \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2x_3$ $\quad d_2 = \bar{x}_1x_2x_3$

Síntese Lógica

1. Após simplificação, pesquisando por cubos primos e sem considerar don't cares temos:

$F_1(x_1,x_2,x_3) = \bar{x}_1\bar{x}_3 + x_1\bar{x}_2\bar{x}_3$ $\quad d_1 = \bar{x}_1x_2x_3$

$F_2(x_1,x_2,x_3) = \bar{x}_2x_3 + x_1\bar{x}_2\bar{x}_3$ $\quad d_2 = \bar{x}_1x_2x_3$

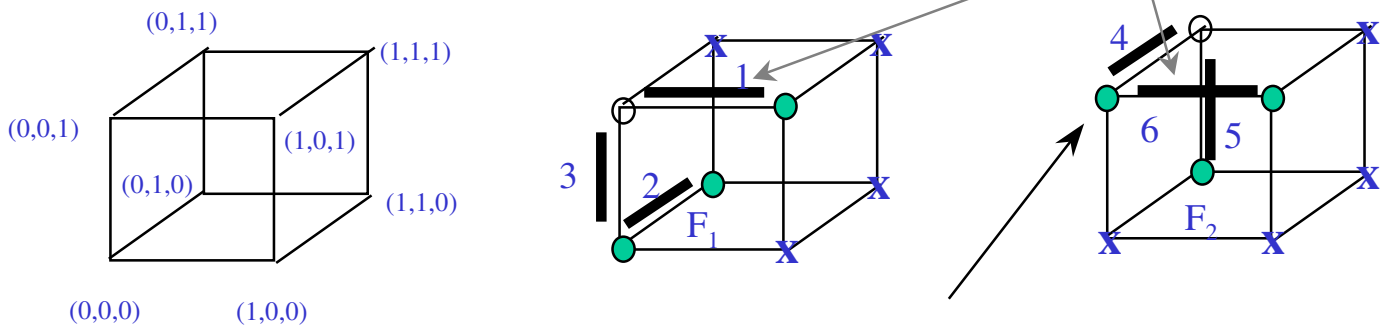


Síntese Lógica

2. Minimização com don't cares para expandir os cubos primos implicantes:

$F_1(x_1,x_2,x_3) = \bar{x}_1\bar{x}_3 + \bar{x}_2x_3 + \bar{x}_1\bar{x}_2$

$F_2(x_1,x_2,x_3) = \bar{x}_2x_3 + \bar{x}_1x_2 + \bar{x}_1x_3$



$F_1(x_1,x_2,x_3) = \bar{x}_1\bar{x}_3 + \mathbf{x_2x_3} + \bar{x}_1\bar{x}_2$

$F_2(x_1,x_2,x_3) = \mathbf{x_2x_3} + \bar{x}_1x_2 + \bar{x}_1x_3$

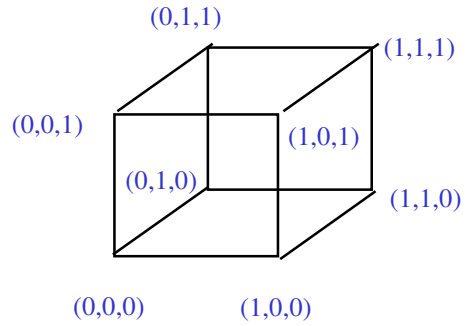
primos múltiplos implicantes

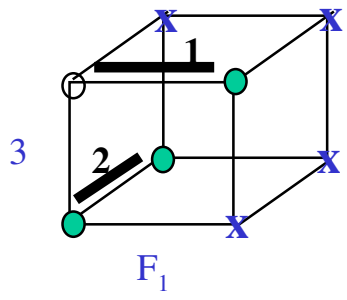
A detecção de múltiplos primos implicantes permite o uso de portas AND do primeiro estágio e de várias portas OR dos segundo estágio de uma PLA

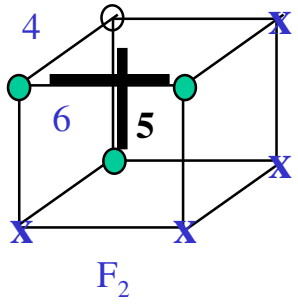
Síntese Lógica

3. A última fase é a retirada dos cubos redundantes:

Na figura abaixo podemos ver que com apenas três cubos podemos cubrir toda a função. São eles os cubos 1, 2 e 5. Portanto, os dois cubos $\bar{x}_1\bar{x}_2$ (00-) de f_1 e $\bar{x}_1x_3$ (0-1) de f_2 podem ser removidos de f.







Resultado final a ser implementado na PLA:

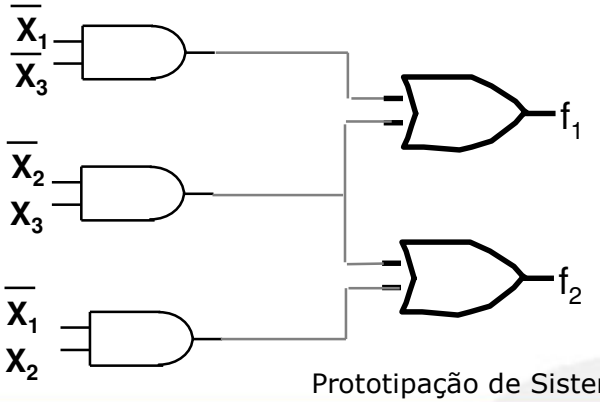
- $F_1(x_1,x_2,x_3) = \bar{x}_1\bar{x}_3 + \bar{x}_2x_3$
- $F_2(x_1,x_2,x_3) = \bar{x}_2x_3 + \bar{x}_1x_2$

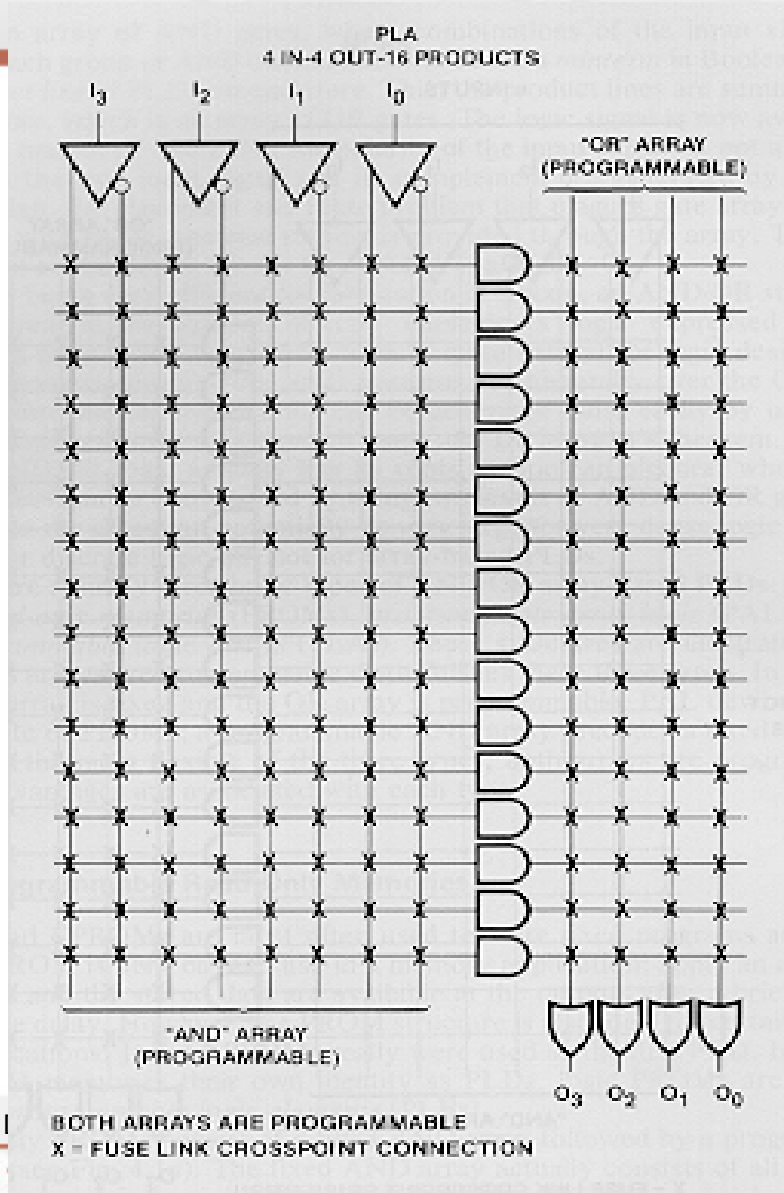
PLA - Programmable Logic Array

- ***Este tipo de arquitetura oferece maior flexibilidade entre os PLDs.***
- ***Minimização em dois níveis:***

$f_1(x_1,x_2,x_3) = \overline{x}_1\overline{x}_3 + \overline{x}_2 x_3$

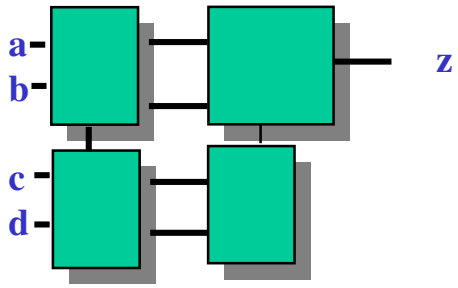
$f_2(x_1,x_2,x_3) = \overline{x}_2 x_3 + \overline{x}_1 x_2$

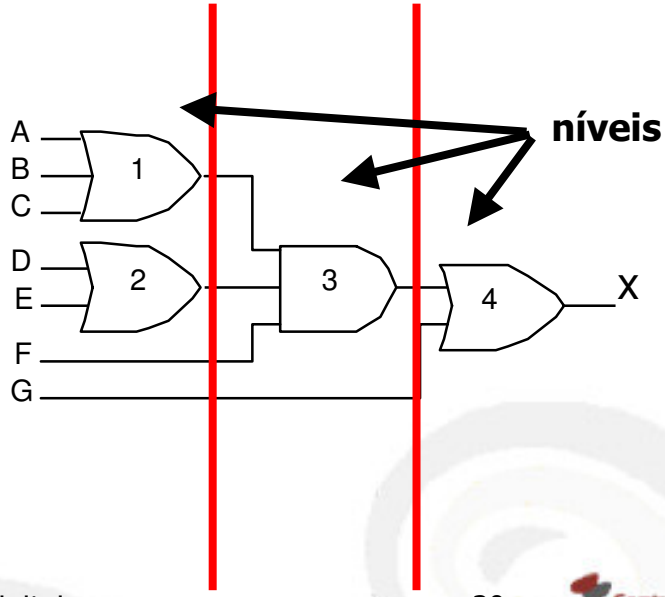




Síntese Lógica

- Otimização Multi-Nível
  - Inicialmente toda a estrutura do circuito é manipulada para otimizar o projeto em termos de área. Este passo independe da tecnologia e o projeto é expresso como equações Booleanas.
  - Na segunda fase o circuito é mapeado em células de uma determinada biblioteca estabelecida pelo usuário. (mapeamento tecnológico). Assim o projeto é finalmente convertido em células que devem satisfazer certos parâmetros previamente estabelecidos pelo projetista.





Síntese Lógica (Multi-nível)

- **Otimização Multi-nível**
  - Este tipo de implementação tem se tornado um dos mais importantes métodos de implementação de lógica e apresenta vantagens sobre a lógica de dois níveis:
    - Representação de termos mais compactos, como por exemplo, o uso de EXOR ao invés de um explosão de portas lógicas, em operações em adicionadores, geradores de paridade, etc.
    - Na lógica de dois níves a representação de uma equação Booleana pode ter até 2^{n-1} termos produto enquanto na Lógica Multi-nível pode ser implementada usando menos área e ser muito mais rápida que sua realização em PLA.
  - Projetos baseados em células, tais como:
    - Gate Array
    - Standard Cell
    - FPGA (Field Programmable Gate Array)

Síntese Lógica – Multi-Nível

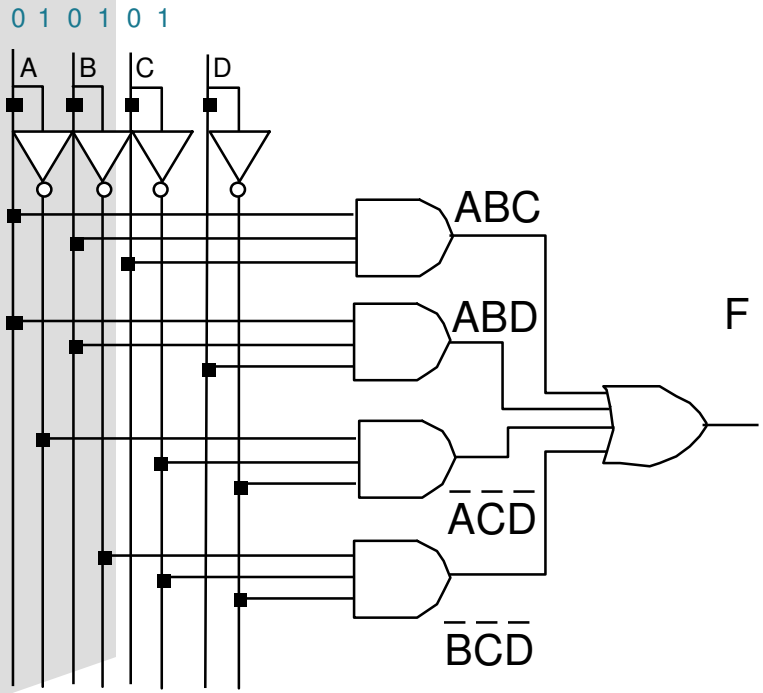
- Operações básicas
  - Decomposição
  - Extração
  - Fatoração
  - Substituição
  - Colapsing

Síntese Lógica - Multi-Nível

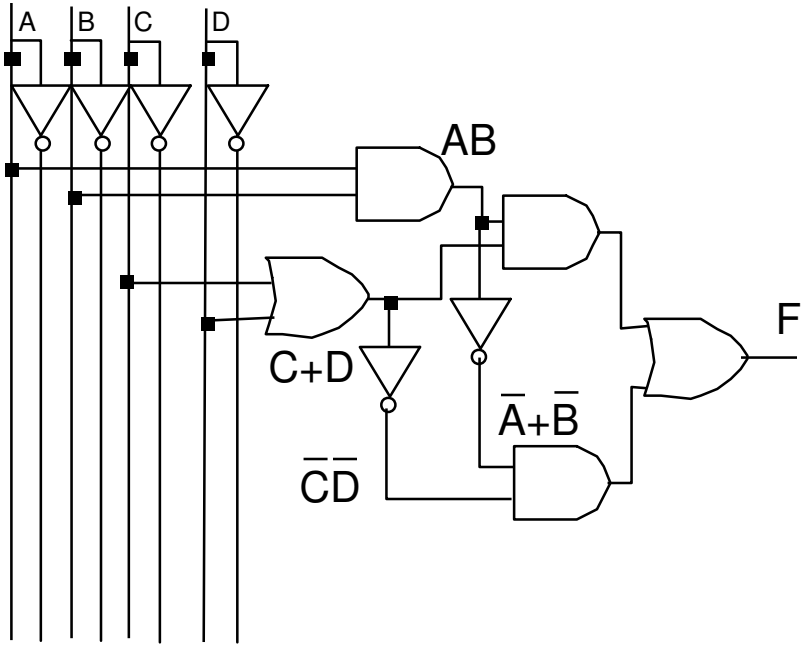
- Decomposição
  - Apanha uma simples expressão Booleana e a decompõe por uma coleção de expressões equivalentes.
  - Exemplo
    - $F=ABC+ABD+\overline{A}CD+\overline{B}\overline{C}\overline{D}$
    - Esta expressão contém 12 literais e requer 9 portas lógicas para ser implementada, contando com inversores (3 entradas e inversores).
  - Decompondo a expressão em duas expressões mais simples teríamos:
    - $F= AB(C + D) + \overline{C}\overline{D}(A + B)$, fazendo $X=AB$ e $Y= C+D$ teremos:
    - $F= XY+\overline{X}\overline{Y}$
    - Esta expressão contém 8 literais e pode ser implementada com 11 porta lógicas (inversores e portas duas entradas)

Decomposição

F=ABC+ABD+ACD+BCD



F= AB(C+D)+CD(A+B)



Síntese Lógica - Multi-Nível

- Extração
  - Nesta técnica podemos usar a regra da decomposição em sistemas com várias funções:
  - A operação identifica sub-expressões comuns à coleção de funções a qual ela é aplicada.
  - Exemplo:
    - F= (A+B)CD+E
    - G=(A+B)E
    - H=CDE
    - Observe que a operação de extração não requer necessariamente que as funções sejam de dois níveis. As expressões acima possuem 11 literais e podem ser implementadas com 8 portas lógicas.
    - Assim identificamos X=(A+B) e Y=CD comuns a F, G e F, H respectivamente.
    - Estas sub-expressões são chamada de *kernel e cubos*.

Síntese Lógica - Multi-Nível

- Extração
- Assim re-escrevendo as expressões, considerando X e Y teríamos:
  - F=XY+E
  - G=XE
  - H=YE
  - As expressões resultantes possuem ainda 11 literais, mas podem ser implementadas com 6 portas lógicas.

F= (A+B)CD+E

G=(A+B)E

H=CDE

-11 literais e podem ser implementadas com 8 portas lógicas

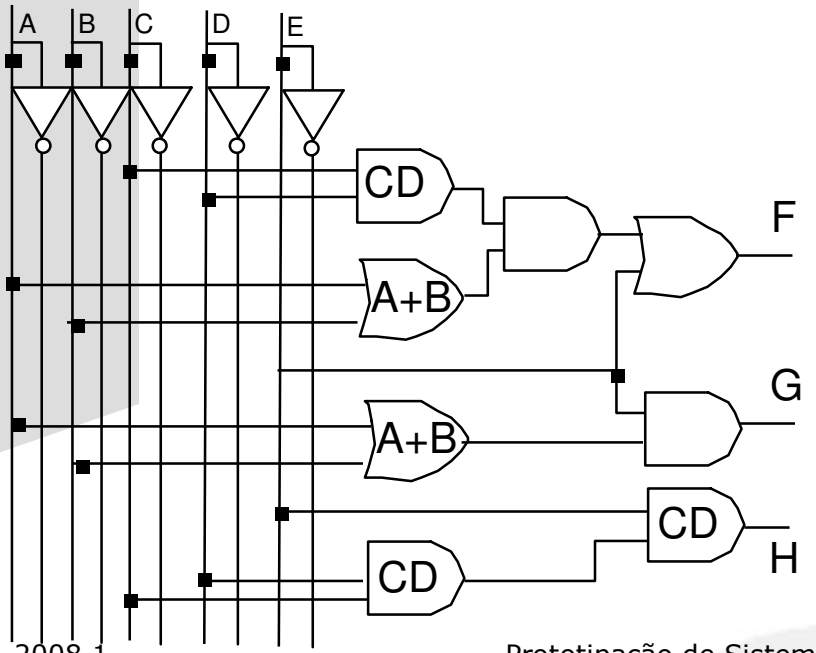
Se fizermos X=(A+B) e Y=CD
Então:

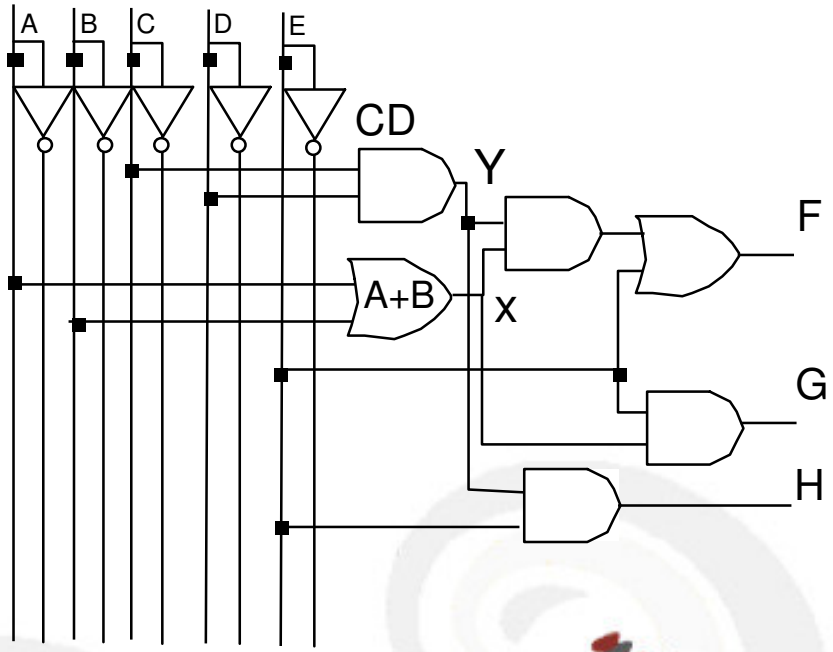
F=XY+E

G=XE

H=YE

- 11 literais, mas podem ser implementadas com 6 portas lógicas.





Síntese Lógica – Multi-Nível

- Fatoração
  - Fatoração assume expressões em dois níveis e as re-escreve como funções multi-nível, sem introduzir nenhuma sub-função intermediária.
  - Exemplo:
    - F=AC+AD+BC+BD+E
    - Esta expressão tem 9 literais e pode ser implementada por 5 portas lógicas.
    - Após a fatoração o número de literais é reduzido para 5 e a expressão pode ser implementada em 4 portas lógicas.
    - F=(A+B)(C+D)+E ou F=XY+E para X=A+B e Y=C+D

Síntese Lógica

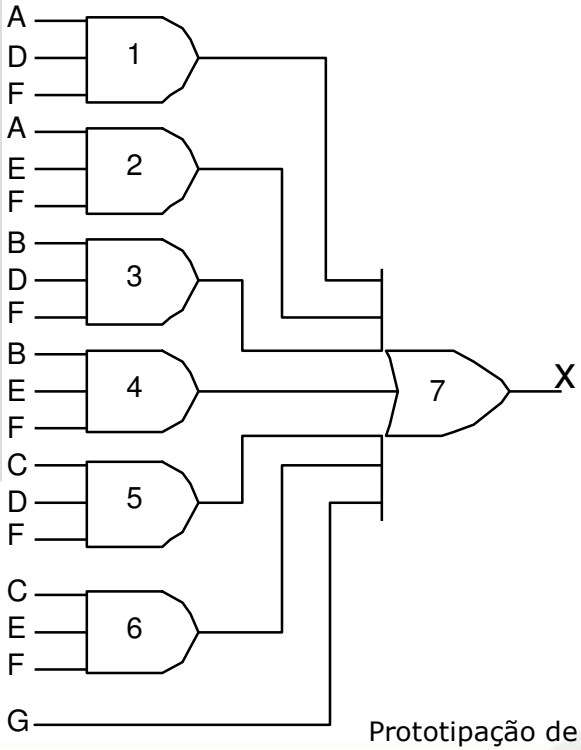
- **Forma fatorada**
  - É especialmente útil em lógica multi-nível, desde que ela representa uma lógica mais próxima da implementação multi-nível. Muitas vezes o número de literais da função fatorada é usada como medida de área durante otimizações multi-nível.
  - Formas fatoradas podem ser recursivamente definidas:
    - Um literal é uma forma fatorada
    - Uma soma de formas fatoradas é uma forma fatorada
    - Um produto de formas fatoradas é uma forma fatorada.
      - Exemplo: F=adf+aef+bdf+bef+cdf+cef+g pode ser representada por: F= (a+b+c)(d+e)f+g, ou seja soma de produtos de profundidade arbitrária.

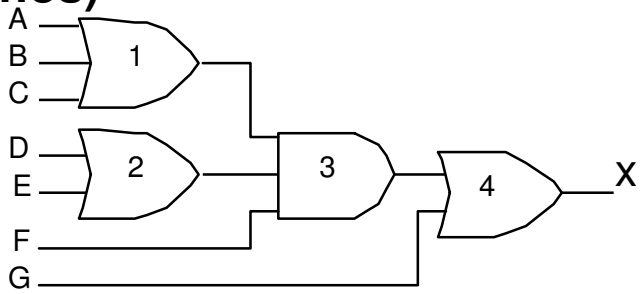
Síntese Lógica

- ***Redução de equação na forma de soma de produtos***

  x = A D F + A E F + B D F + B E F + C D F + C E F + G

  6 x 3-input AND gates + 1 x 7-input OR gate (não existe)
  25 fios (19 literais mais 6 fios internos)





Forma fatorada:

x = (A + B + C) (D + E) F + G

1 x 3-input OR gate, 2 x 2-input OR gates,
1 x 3-input AND gate
10 fios (7 literais mais 3 fios internos)

Síntese Lógica - Multi-Nível

- Substituição
  - Técnica que substitui uma função G dentro de outra função F e re-define F em termos de G.
  - Exemplo:
    - Suponha F=A+BC e G=A+B
    - Então F poderia ser re-escrito como F=G(A+C)
    - Ou seja, em geral, sub-expressões comuns identificadas num conjunto de expressões podem ser usadas como forma de fatoração de expressões.

- Colapsing
  - Esta técnica é a reversa da substituição. Colapsing pode ser usada para reduzir o número de níveis de lógica em função de restrições temporais.
  - Exemplo:
    - Colapsar G de volta para F: Considere G=A+B
    - F=G(A+C) = (A+B)(A+C)=AA+AC+AB+BC=A+BC
    - O número de literais da expressão F foi reduzida de 4 para 3 literais.