ECE-256d

Malgorzata Marek-Sadowska
Electrical and Computer Engineering Department
Engineering I, room 4111
mms@ece.ucsb.edu.

CAD for semi-custom ASICs

- ASIC = application specific integrated circuit
- Semi-Custom = try to design reusing some already designed parts
- CAD = flow through a sequence of design steps and software tools.

Spectrum of design approaches

Fully custom means everything Done by hand, mostly at the transistor and layout level.

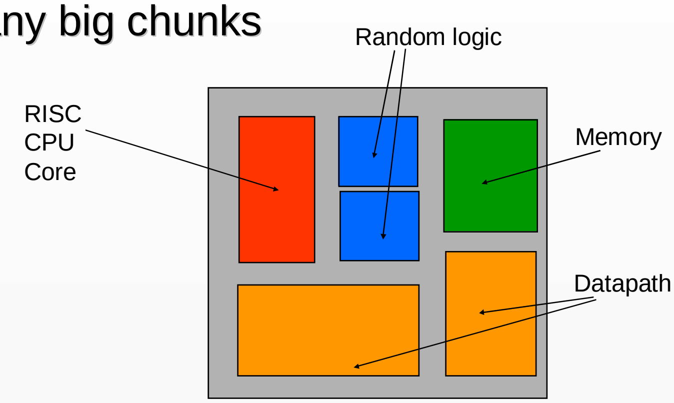
Example : microprocessors.

Semi-custom means try to design using existing parts.

Example: ethernet chip,hard disk controller.

Example of modern system-on-a-chip IC

- Many big chunks



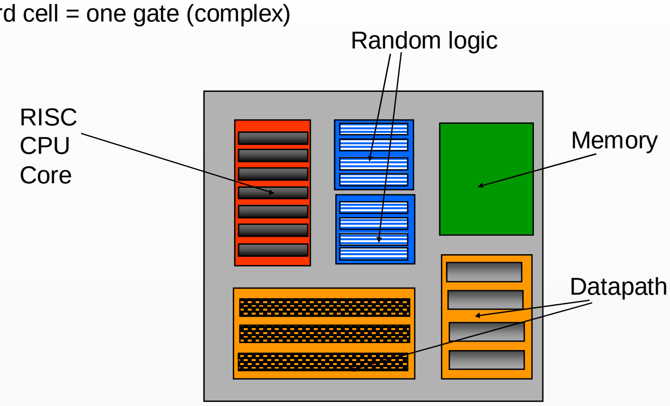
Useful Components in Semi-Custom

- Logic gates
  - Maximally useful components you can reuse
  - Can design without knowing exactly what gates (type, speed, power, size) you have : technology independent design.
  - Later, can map technology independent design onto specific gate library (technology) : technology mapping problem.
- Memories
  - Module generator transforms specs on size (bits, words, speeds) into final layout.
  - Very structured designs.
- Datapaths
  - Well structured (adders, multipliers)
  - Often designed at gate and transistor level
  - Produced by module generators.

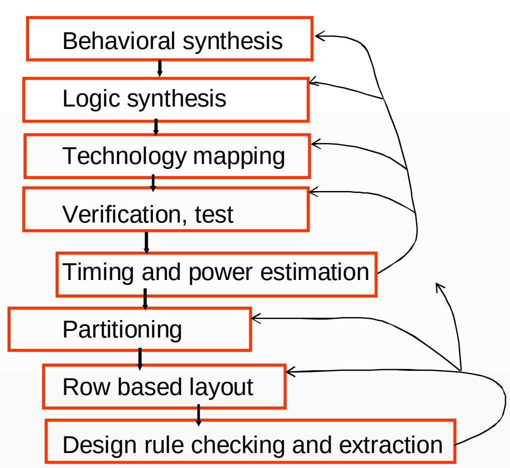
Semi-custom ASIC

- Made out of standard cells

Standard cell = one gate (complex)



ASIC CAD Tool Flow

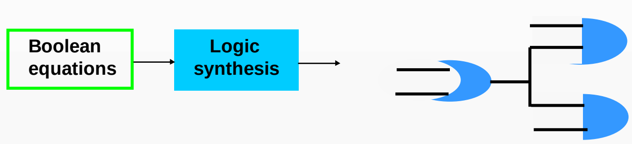


High level (behavioral synthesis)

- Input :
  - High level description of desired system function, usually as a program in a hardware description language (Verilog, VHDL).
- Output:
  - Register transfer level structure: FSMs, logic, ALUs, memory, busses.

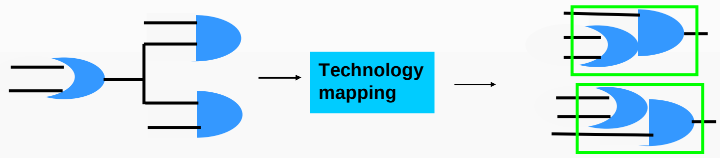
Logic synthesis

- Input:
  - Boolean equations, state diagrams, etc.
- Output:
  - Gates and connections, called netlist, a structural design.



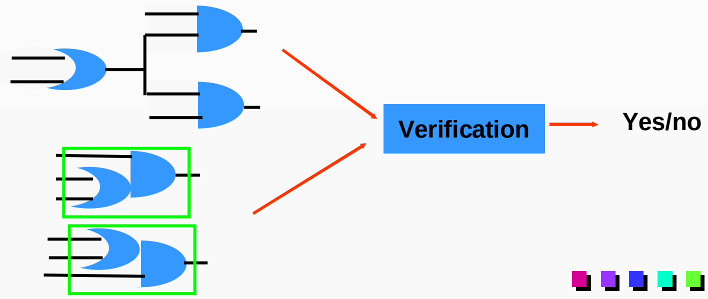
Technology mapping

- Input:
  - Technology independent gate level design (un-commited design)
- Output:
  - Gate level design using specific technology library.



Formal verification

- Input:
  - A specification for a design (Boolean eqns) and an implementation
- Output:
  - Decision yes/no: is specification == implementation

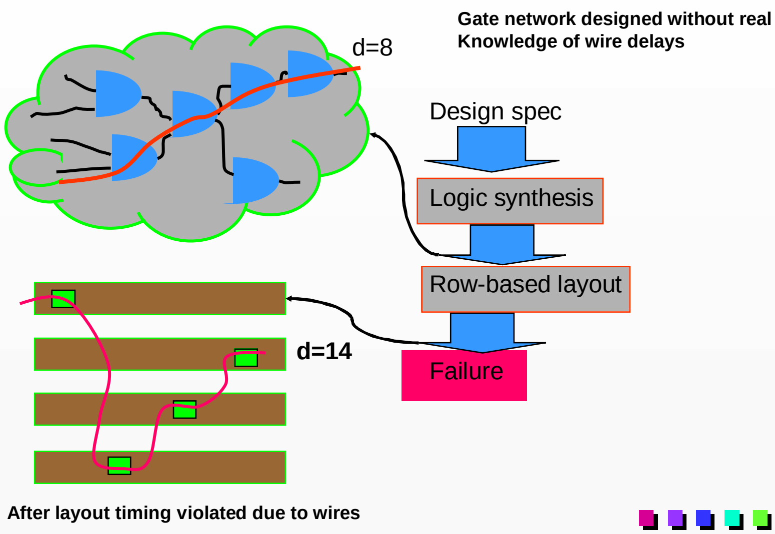


Timing estimation

- Input:
  - A gate level design, timing info about gates and wires
- Output:
  - Delay estimate – critical path length

256d 11

Convergence problems between synthesis and layout



256d 12

Incompletely specified functions

For incompletely specified function ff we build 3 completely specified functions: $ff_{on}, ff_{dc}, ff_{off}$.

on | off | dc   ff

f   on | off ff_{on} On-set the same as On-set of ff

r   off | on | off ff_{off} On-set the same as Off-set of ff

d   off | on ff_{dc} On-set the same as DC-set of ff

$f \cup d \cup r$ is a tautology

Motivation

- Commercial success - used almost everywhere VLSI is done
- More general treatment of discrete functions of discrete value variables.
- Body of useful and general techniques - can be applied to other areas.
- Foundation for:
  - combinational and sequential synthesis
  - testing
  - timing and false paths
  - formal verification
  - optimal clocking schemes
  - power estimation
  - general combinatorics.

Outline of the class

- Introduction
- 2-level combinational circuits
- Binary decision diagrams
- Synthesis of multi-level circuits
- Technology mapping
- Delay in multi-level circuits
- Testability of multi-level circuits
- Boolean matching
- Automatic test pattern generation techniques in logic synthesis

256d 15

Grading

- Homework assignments : 20%
- Final project : 70%
- Class presentation of the project : 10%

- You need to do one project for both 256b and 256d.

256d 16

Texts

- Suggested books:
  - R.K.Brayton, G.D.Hachtel, C.T.McMullen and A.Sangiovanni-Vincentelli, “Logic Minimization Algorithms for VLSI Synthesis”, Kluwer Academic Publishers, Boston, MA, 1984.
  - G.D. Hachtel and F.Somenzi, “Logic Synthesis and Verification Algorithms”, Kluwer Academic Publishers, Boston/Dordrecht/London, 1998.

Logic Synthesis

- Goal:
  - Map a high level functional description of logic function into a set of primitives in a given technology.
- Automation:
  - Predominantly for random logic
- Automatic logic synthesis
  - Functional design (functional specification of the system, transformed into a logic description in terms of Boolean variables)
  - Logic design (manipulation of the logic representation without modification of functionality).

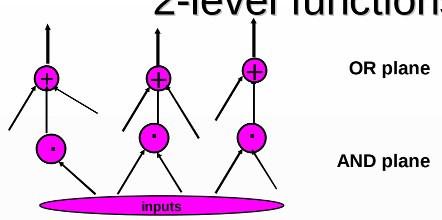
Physical design

- Custom (macro-cells)
  - high performance, highly optimized designs
- Standard cells
- Gate arrays
- Field programmable gate arrays

Do not support highly optimized designs

- Between macro cells and standard cell: algorithmically generated macros produced by module generators.
  - PLA: effective for designing combinational circuits
  - ROM: look-up table (large Si area)

2-level functions



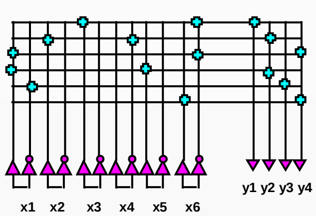
- PLA are the most popular structures for implementation of 2-level logic functions.

Input matrix

X1	x2	x3	x4	x5	x6
*	*	1	*	*	0
*	1	*	0	*	*
1	*	*	*	*	0
1	*	*	*	1	*
0	*	*	*	*	*

Output matrix

y1	y2	y3	y4
1	0	0	0
0	1	0	0
0	0	0	1
0	1	0	0
0	0	0	1



y1=x3x6'

y2=x2x4'+x1x5

y3=x1'

y4=x1x6'+x6

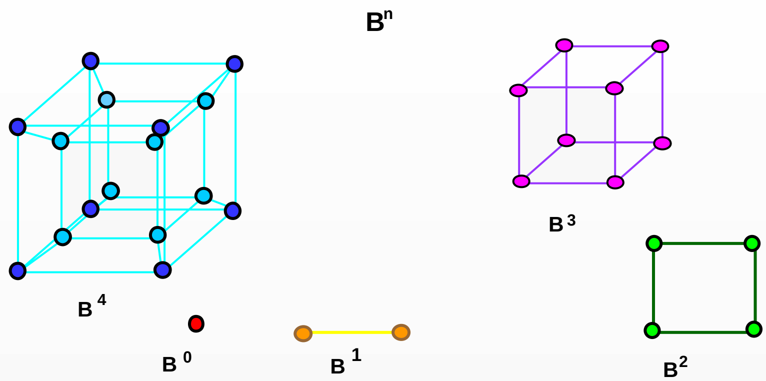
Optimization steps for PLA

- Logic optimization: reduction of the number of product terms needed to implement the given function.
- Topological: elimination of unused space; folding and partitioning.
- Layout and circuit optimization: optimal sizing and placement of drivers, devices and lines.
- Up to the definitions of the device and interconnect location, PLA is independent of implementation technology.
- Advantages:
  - regular structure, easy to automate
  - minimization is well understood
- Disadvantages:
  - no shape control
  - little control of speed
  - little control of I/O placement

PLA (2-level) vs Multi-level

PLA (2-level)	Multi-level
Well developed	Relatively undeveloped
Technology independent	?
Multi-valued	?
Control logic	All (control and data flow)
Constrained layout	Flexible, no layout style
Automatic layout	+- Automatic layout
Mature	Not so

The Boolean n-cube



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

Basic definitions

- B = { 0,1}, Y = {0,1,2} , a logic function ff.: B^n -> Y^m
  x $\in$ B^n is an input, y $\in$ Y^m is an output.
  2 - don't care value
  ff - incompletely specified function
  f - a completely specified function
  ff = (ff_1,ff_2,,,,ff_m)
  $\forall i : 1 \le i \le m :$
- On-set: $X_i^{on} \subseteq B^n$: such x that ff_i(x)=1
- Off-set: $X_i^{off} \subseteq B^n$: such x that ff_i(x)=0
- Don't care set: $X_i^{dc} \subseteq B^n$: such x that ff_i(x)=2
  m=1 : single output function
  m>1 : multiple output function

Example

- Tabular representation

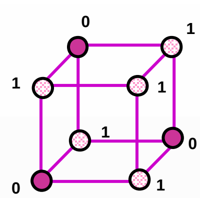
X1	X2	X3	Y1	Y2
0	0	0	1	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	2
1	1	0	1	1
1	1	1	2	1

$X1^{on}$ = {[0,0,0],[0,0,1],[1,0,0],[1,0,1],
[1,1,0]}

$X1^{off}$= {[0,1,0],[0,1,1]}

$X1^{DC}$ = {[1,1,1]}

Boolean functions



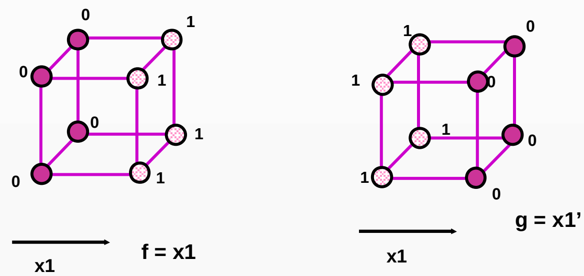
f(x) : B^n -> B

B={0,1}, x={x1,x2,…xn}

- Each vertex of B^n is mapped to 0 or 1.
- The **onset** of f is {x | f(x)=1} = $f^1 = f^{-1}(1)$
- the **offset** of f is { x | f(x) =0} = $f^0 = f^{-1}(0)$
- if $f^1 = B^n$, f is the **tautology**.
- If $f^0 = B^n$, f is not **satisfiable.**
- If f(x) = g(x) for all x in B^n, then f and g are **equivalent.**
- x1, x2, … are **variables**
- x1, x1', x2, x2' …are **literals**

Literals

- A literal is a variable or its negation : y, y'.
  It represents a **logic function**
  Literal x1 represents the logic function f, where f^1 = { x | x1 = 1}
  Literal x1' represents the logic function g, where g = { x | x1 = 0}



256d

Boolean formulas

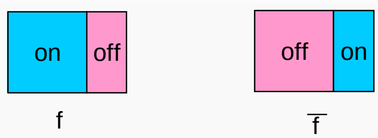
- Boolean functions can be represented by formulas defined as catenations of
  - parentheses - (,)
  - literals - x, y, z, x', y', z'.
  - Boolean operations - + (or), * (and)
  - complementations (x+y)'
- Examples:
  - f = x1 * x2' + x1' * x2 = (x1+x2) * (x1' + x2')
  - h= a + b*c = (a' * (b' + c'))'
  - We will usually replace * by catenation, e.g. a*b -> ab.

256d

Operations on Boolean functions

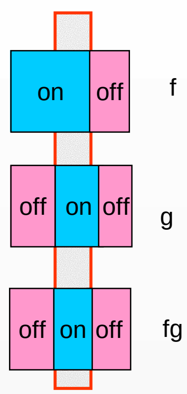
Multiple output functions: the usual Boolean operations are performed component-wise on the outputs.

A complement of $f : B^n \to B^m$ is a function $\overline{f} : B^n \to B^m$ such that $\overline{f}_1, \overline{f}_2, \cdots, \overline{f}_m$ have their on-sets equal to the off-sets of f.



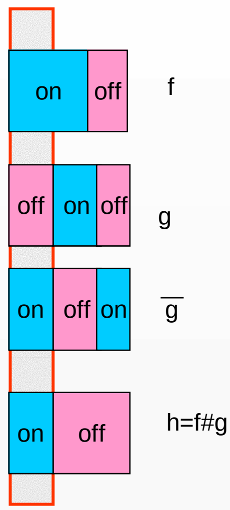
256d 29

The intersection: $h = f \cdot g \; (f \cap g) : h_i$ has an on-set equal to the intersection of the on-sets of $f_i and \; g_i$.



256d 30

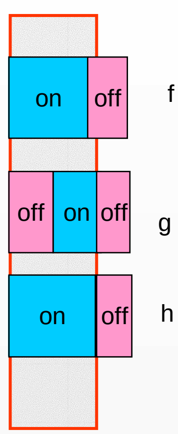
The difference: $h = f - g\ (f \# g) = f \cap \overline{g}$

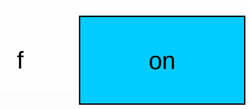


256d

31

The union: $h = f + g \ \ (f \cup g)$





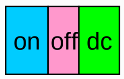
The tautology: off set is empty.

256d

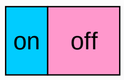
32

Incompletely specified functions

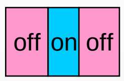
For incompletely specified function ff we build 3 completely specified functions: $ff_{on}, ff_{dc}, ff_{off}$.



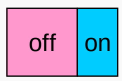
ff

f


ff_{on} On-set the same as On-set of ff

d


ff_{off} On-set the same as Off-set of ff

r


ff_{dc} On-set the same as DC-set of ff

$f \cup d \cup r$ is a tautology

Algebraic representation

$ff = (ff_1, ff_2, \cdots, ff_m)$. f_i is an algebraic representation of ff_i if it is a Boolean expression that evaluates to 1 for all inputs in X_i^{ON}, to 0 for all inputs of X_i^{OFF}, and either to 0 or 1 for all inputs in X_i^{DC}.

Algebraic representation of ff is denoted by **f**, **f**(ff).

Example

x1	x2	x3	y1	y2
0	0	0	1	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	2
1	1	0	1	1
1	1	1	2	1

Can be simplified

$$f_1 = \bar{x}_2 + x_1\bar{x}_3$$

$$f_2 = x_2 + \bar{x}_1\bar{x}_3$$

$$f_1 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2 x_3 + x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2 x_3 + x_1 x_2\bar{x}_3$$

$$f_2 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1 x_2\bar{x}_3 + \bar{x}_1 x_2 x_3 + x_1 x_2\bar{x}_3 + x_1 x_2 x_3$$

(Sum of products form)

256d 35

$$f_1 = \bar{x}_2 + x_1\bar{x}_3$$

$$f_2 = x_2 + \bar{x}_1\bar{x}_3$$

Each product term in the sum of products algebraic representation of f determines a logic function.

$\bar{x}_2$

x1	x2	x3	
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	2

2-D cube

$x_1\bar{x}_3$

x1	x2	x3	
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	2

1-D cube

256d 36

Cubes

P - a product term in an algebraic sum of products expression of a logic function of n inputs and m outputs

A cube p is specified by $c = [c_1, c_2, \cdots, c_{n+m}]$

$$c_i = \begin{cases} 0 & \text{if } x_i \text{ appears complemented in p} \\ 1 & \text{if } x_i \text{ appears not complemented in p} \\ 2 & \text{if } x_i \text{ does not appear in p} \end{cases} \quad i=1,2,\ldots,n$$

$$\begin{cases} 3 & \text{if p is not present in algebraic representation of } f_{i-n} \\ 4 & \text{if p is present in the algebraic representation of } f_{i-n} \end{cases} \quad i=n+1\ldots n+m$$

Example: $f_1 = \overline{x}_2 + x_1\overline{x}_3$

$f_2 = x_2 + \overline{x}_1\overline{x}_3$

P= $\overline{x}_2$ $\quad$ $C=[\underbrace{2\quad 0\quad 2}_{\text{Input cube}}\ \overbrace{4\quad 3}^{\text{output cube}}]$

Input cube = compact form of the coordinates of the vertices of the cube corresponding to the product term.

Example: I(c) = [2 0 2] represents (1,0,1), (0,0,0), (1,0,0) and (0,0,1).

O(c) = [4 3] identifies the space where the cube belongs.

$C = \{c^1, c^2, \cdots, c^k\}$ is a cover of ff with n inputs and m outputs, if for

j=1,2,….m, the set of input parts of the cubes that have a 4 in the j-th position contain all the vertices corresponding to the on-set of ff_j and none of the off-set of ff_j, i.e. a cover represents a union of the on-set and some arbitrary position of the don't cares.

There is a 1-1 correspondence between a cover and an algebraic representation of a function as a sum-of-products.

A matrix representation of a cover:

M(C) of $c = [c_1, c_2, \cdots, c_{n+m}]$ is a matrix obtained by stacking the row vectors representing each of the cubes of C.

Example: $f_1 = \bar{x}_2 + x_1 \bar{x}_3$
$f_2 = x_2 + \bar{x}_1 \bar{x}_3$

$$M(C) = \begin{bmatrix} 2 & 0 & 2 & 4 & 3 \\ 1 & 2 & 0 & 4 & 3 \\ 2 & 1 & 2 & 3 & 4 \\ 0 & 2 & 0 & 3 & 4 \end{bmatrix}$$

G=**I**(M(C)) input matrix

H=O(M(C)) output matrix

Matrix representation and cover are used interchangeably.
If C is a cover of a single output function, then H=∅

Let $c = [c_1, c_2, \cdots, c_{n+m}]$ and $d = [d_1, d_2, \cdots, d_{n+m}]$

be 2 cubes.

The cube c contains d if:

the cube represented by the input part of c contains all the vertices of d; and must be present in all Boolean spaces where d is present.

A minterm e^i is a cube whose input part does not contain any 2s and whose output part contains (m-1) 3s and one 4 in position I.

The input cube is a vertex and this vertex is present only in one, I-th Boolean n-space. A minterm does not contain any other cube. If a cube contains a minterm e^i we say that e^i is an element of c.

Example: [1,1,1,4,3] is a minterm and an element of [2,2,1,4,4].

Each cube can be decomposed into a set of all minterms that are elements of the cube.

Example

c=[2, 2, 1, 4, 4]

e^1 = [0, 0, 1, 4, 3]
e^2 = [0, 0, 1, 3, 4]
e^3 = [1, 0, 1, 4, 3]
e^4 = [1, 0, 1, 3, 4]
e^5 = [0, 1, 1, 4, 3]
e^6 = [0, 1, 1, 3, 4]
e^7 = [1, 1, 1, 4, 3]
e^8 = [1, 1, 1, 3, 4]

A set of cubes $C = [c^1, c^2, \cdots, c^{n+m}]$ covers a cube c $(c \subseteq C)$ if each of the minterms of c is covered by at least one cube of C.

Special cubes:

u^j - the universe of the j-th Boolean space

$u^j = \dfrac{1,2,\ldots\ldots,j\ldots.n,\ n+1,\ \ldots,\ n+j,\ \ldots,\ n+m}{2,\ldots\ldots\ldots 2\ldots 2,\ 3,\ \ldots.3,\ldots 4,\ \ldots.3\ldots.3}$

$U = \dfrac{1,2,\ldots\ldots,j\ldots.n,\ n+1,\ \ldots,\ n+j,\ \ldots,\ n+m}{2,\ldots\ldots\ldots 2\ldots 2,\ 4,\ \ldots.4,\ldots 4,\ \ldots.4\ldots.4}$

U is the total universe

x^j : the positive half-space of x^j $\dfrac{1,2,\ldots\ldots,j\ldots.n,\ n+1,\ \ldots,\ n+j,\ \ldots,\ n+m}{2,\ldots\ldots\ldots 1\ldots 2,\ 4,\ \ldots.4,\ldots 4,\ \ldots.4\ldots.4}$

De Morgan's law:

$$C = [c^1, c^2, \cdots, c^{n+m}]$$

1. Express the cover in algebraic form

2. Exchange AND and OR

3. Change variables to complements

Example

$$f = x_1\overline{x}_3 + \overline{x}_2 x_4 + x_1 x_2 \overline{x}_4$$

$$\overline{f} = (\overline{x}_1 + x_3)(x_2 + \overline{x}_4)(\overline{x}_1 + \overline{x}_2 + x_4)$$

Multiply out using rules of Boolean algebra:

$$.\ x\overline{x} = 0$$

$$2.\ xx = x$$

$$3.\ x + \overline{x} = 1$$

$$\boxed{\overline{f} = \overline{x}_1 x_2 + \overline{x}_1 \overline{x}_4 + x_2 x_3 x_4 + x_3 \overline{x}_4 (\overline{x}_1 + \overline{x}_2)}$$

Intersection or a product of 2 cubes

$c \cap d$

$1 <= i <= n$

c_i \ d_i	0	1	2
0	0	$\emptyset$	0
1	$\emptyset$	1	1
2	0	1	2

$n < i <= n+m$

c_i \ d_i	3	4
3	3	3
4	3	4

$\emptyset$ is an empty cube

If an output part of a cube has all 3 it is empty.

Intersection: input part corresponds to the vertices that are common to c and d. Output part specifies that the cube is present in the Boolean n-spaces in which both c and d are present.

If 2 cubes have no common vertices or no common Boolean space:

$c \cap d = \emptyset$, c and d are orthogonal. $f \cap \overline{f} = \emptyset$

The union of 2 cubes: $c \cup d$ (c+d): the set of vertices covered by the input part of either c or d in the Boolean n-space where they are present.

In matrix representation: $c \cup d$ is the matrix formed by 2 rows corresponding to c and d, respectively.

The distance between 2 cubes: (# of conflicts)

$$\delta(c,d) = \delta(I(c), I(d)) + \delta(O(c), O(d))$$

where

$$\delta(I(c), I(d)) = |\{ j \mid c_j \cap d_j = \emptyset \}|$$

$$\delta(O(c), O(d)) = \begin{cases} 0 \text{ if } c_j \cap d_j = 4 \\ 1 \text{ otherwise} \end{cases}$$

Some j>n

The consensus of 2 cubes: $e = c \Theta d$

If $\delta(c,d) \neq 1$ then $e = \begin{cases} c \cap d & if \quad \delta(c,d) = 0 \\ \emptyset & if \ \delta(c,d) \geq 2 \end{cases}$

$If \ \delta(I(c), I(d)) = 1 \wedge \delta(O(c), O(d)) = 0$ then

$$e_l = \begin{cases} c_l \cap d_l & if \quad c_l \cap d_l \neq 0 \\ 2 & otherwise \end{cases}$$

$If \ \delta(I(c), I(d)) = 0 \wedge \delta(O(c), O(d)) = 1$

$$e_l = \begin{cases} c_l \cap d_l \ \ 1 \leq l \leq n \\ 4 \quad if \ c_l \ or \ d_l = 4 \ for \ n < l \leq n+m \\ 3 \quad otherwise \end{cases}$$

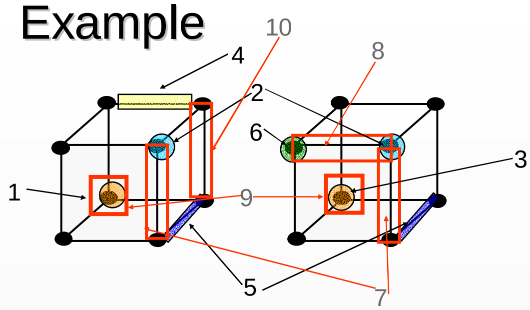
256d 45

Theorem: The consensus of 2 cubes a and b, $p = a \Theta b$ is contained in $a \cup b$. If $a \Theta b \neq \emptyset$, it contains minterms of both a and b. p is the largest cube contained in $a \cup b$.

$$\exists x, \exists y, x, y \in p, x \in a, y \in b.$$

256d 46

Example



cubes
consensus

$$c_7 = c_2 \Theta c_5 \qquad c_2 \Theta c_3 = \emptyset$$
$$c_8 = c_2 \Theta c_6 \qquad c_1 \Theta c_2 = \emptyset$$
$$c_9 = c_1 \Theta c_3 \qquad c_3 \Theta c_6 = \emptyset$$
$$c_{10} = c_5 \Theta c_4 \qquad c_5 \Theta c_6 = \emptyset$$

The complement of a set of cubes C, $\overline{\text{C}}$ covers the complement of logic corresponding to C.

The difference: C-H covers $C \cap \overline{H}$.

A cube is an implicant of ff=(f,d,r) if it has an empty intersection with the cubes of a representation of r.

Example.

$$F=M(C)=\begin{bmatrix} 2 & 0 & 2 & 4 & 3 \\ 1 & 2 & 0 & 4 & 3 \\ 2 & 1 & 2 & 3 & 4 \\ 0 & 2 & 0 & 3 & 4 \end{bmatrix}$$

(1,2,0,4,3) is an implicant of ff. (0,2,1,3,4) is not since it contains (0,0,1) in the Boolean space representing ff2 that is in the off-set of ff2.