Logic Optimization

Heuristic Based

Virendra Singh
Professor
Department of Electrical Engineering &
Dept. of Computer Science & Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@{ee, cse}.iitb.ac.in

EE-677: Foundations of VLSI CAD

Logic Minimization

Exact Methods:

Redundant Faults

2-Leve:

Heuristic based 2-Level Logic Minimization

Unateness

- Function $f\ (x_1, x_2, \ldots., x_i, \ldots., x_n)$
- *Positive unate* in x_i when:
  - $f_{xi} \geq f_{xi'}$
- *Negative unate* in x_i when:
  - $f_{xi} \leq f_{xi'}$

Binate

- A function is positive/negative unate when positive/negative unate in all its variables

Operators

- Function $f\ (x_1, x_2, \ldots., x_i, \ldots., x_n)$
- *Boolean difference* of f w.r.t. variable x_i:
  - $\partial f/\partial x_i \equiv f_{xi} \oplus f_{xi'}$
- *Consensus* of f w.r.t. variable x_i:
  - $C_{xi} \equiv f_{xi} \cdot f_{xi'}$ intersection
- *Smoothing* of f w.r.t. variable x_i:
  - $S_{xi} \equiv f_{xi} + f_{xi'}$ union.

Generalized Expansion

- Given:
  - A Boolean function f.
  - Orthonormal set of functions:

    ϕ_i, i = 1, 2, … , k
- Then:
  - $f = \sum_i^k \phi_i \cdot f_{\phi_i}$
  - Where f_{ϕ_i} is a *generalized cofactor*.
- The generalized cofactor is not unique, but satisfies:
  - $f \cdot \phi_i \subseteq f_{\phi_i} \subseteq f + \phi_i'$

$f = x_i f_{x_i} + \bar{x}_i f_{\bar{x}_i}$

$= x_i f_{x_i} \oplus \bar{x}_i f_{\bar{x}_i}$

$\phi_i = x_i$

$= \phi_1 f_{\phi_1} + \phi_2 f_{\phi_2} + \cdots$

$\phi_1 = x_1$

$\phi_2 = \bar{x}_1$

Example

$a \cdot b(\bar{a}+\bar{b}) = 0$

$x_i \cdot \bar{x}_i = 0$

- Function: $f = ab + bc + ac$
- Basis: $\phi_1 = ab$ and $\phi_2 = a' + b'$.
- Bounds:
  - $ab \subseteq f_{\phi_1} \subseteq 1$
  - $a'bc + ab'c \subseteq f_{\phi_2} \subseteq ab + bc + ac$
- Cofactors: $f_{\phi_1} = 1$ and $f_{\phi_2} = a'bc + ab'c$.

$$\sum_{i=1}^{2} \phi_i \cdot f_{\phi_i}$$

$f_{\phi_1} = f_{ab} = 1$

$\phi_2 = \bar{a}+\bar{b}$

$f_{\phi_2} = \bar{a} \cdot bc + \bar{b} \cdot ac$

$ab(\bar{a}+b) = 0 + ab$

$$f = \phi_1 f_{\phi_1} + \phi_2 f_{\phi_2}$$

$$= ab1 + (a' + b')(a'bc + ab'c)$$

$$= ab + bc + ac$$

Generalized expansion theorem

- Given:
  - Two function f and g.
  - Orthonormal set of functions: ϕ_i , i=1,2,…,k
  - Boolean operator $\odot$
- Then:
  - $f \odot g = \sum_i^k \phi_i \cdot (f_{\phi_i} \odot g_{\phi_i})$

$= x_i \, (f_{x_i} \cdot \overset{\odot}{op} \cdot g_{x_i}) + x_i' (f_{x_i'} \, op \cdot g_{x_i'})$

- Corollary:
  - $f \odot g = x_i \cdot (f_{x_i} \odot g_{x_i}) + x_i' \cdot (f_{x_i'} \odot g_{x_i'})$

ROBDD

Matrix representation of logic covers

- Representations used by logic minimizers
- Different formats
  - Usually one row per implicant
- Symbols:
  - 0, 1, *, …
- Encoding:

a b c, E.

$\boxed{ab + bc + cd.}$

ab
bc
ca

3 values

∅	00
0	10
1	01
*	11

void

positional representation

2 bits

Advantages of positional cube notation

- Use binary values:
  - Two bits per symbols
  - More efficient than a byte (char)
- Binary operations are applicable
  - Intersection – bitwise AND
  - Supercube – bitwise OR
- Binary operations are very fast and can be parallelized

Example

- $f = a'd' + a'b + ab' + ac'd$

ā * * d
10 11 11 10

a	b	c	d
10	11	11	10
10	01	11	11
01	10	11	11
01	11	10	01

var ×2

4×8

Implicant

bit wise operation

10 11 11 10
01 11 11 11
→ 00 11 11 10

void

Cofactor computation

- Cofactor of α w.r.t β
  - Void when α does not intersect β
  - $a_1 + b_1'$ $a_2 + b_2'$ … $a_n + b_n'$

- Cofactor of a set $C = \{\gamma_i\}$ w.r.t β:
  - Set of cofactors of γ_i w.r.t β

Example $f = a'b' + ab$

v_1 v_2

a* wrt $\underline{\underline{a}}$

- Cofactor w.r.t 01 11
  - First row – void
  - Second row – 11 01

10	10
01	01

1
0

10	10	
01	11	
00	10	→ void

$f = \bar{a}\bar{b} + ab$

$f_a = \underline{\underline{b}}$

Example $f = a'b' + ab$

$a*$ wrt @

- Cofactor w.r.t 01 11
  - First row – void
  - Second row – 11 01
- Cofactor $f_a = b$

$f = \bar{a}\bar{b} + ab$

$f_a = b$

10	10
01	01
01	01
01	11
01	01
10	00
11	01

Multiple-valued-input functions

- Input variables can take many values
- Representations:
  - Literals: set of valid values
  - Function = sum of products of literals
- Positional cube notation can be easily extended to mvi
- Key fact
  - Multiple-output binary-valued functions represented as mvi single-output functions

Example

- 2-input, 3-output function:
  - $f_1 = a'b' + ab$
  - $f_2 = ab$
  - $f_3 = ab' + a'b$

$f_1(a,b)$

$f_2(a,b)$

$f_3(a,b)$

a

01 11

- Mvi representation:

$\bar{a}\bar{b}$
ab
$a\bar{b}$
$\bar{a}b$

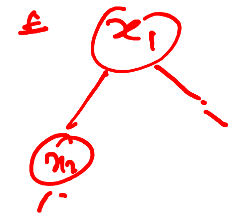
			f1 f2 f3
$\bar{a}\bar{b} \rightarrow$	10	10	100
$\bar{a}b$	10	01	001
	01	10	001
	01	01	110

Fundamental Operation

- Objective
  - Operations on logic covers
  - Application of the recursive paradigm
  - Fundamental mechanisms used inside minimizers

Operations on logic covers

- Recursive paradigm
  - Expand about a mv-variable
  - Apply operation to co-factors
  - Merge results
- Unate heuristics
  - Operations on unate functions are simpler
  - Select variables so that cofactors become unate functions
- Recursive paradigm is general and applicable to different data structures
  - Matrices and binary decision diagrams



Tautology

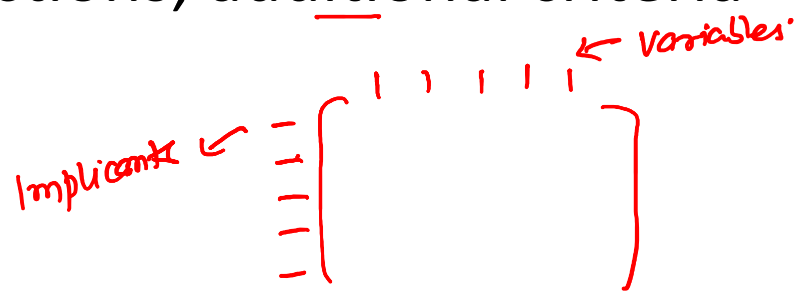
- Check if a function is always TRUE
- Recursive paradigm:
  - ➢ Expend about a mvi variable
  - ➢ If all cofactors are TRUE, then the function is a tautology
- Unate heuristics
  - ➢ If cofactors are unate functions, additional criteria to determine tautology
  - ➢ Faster decision

f

$x_i \cdot f_{x_i} + \bar{x}_i \cdot f_{\bar{x}_i}$

1 $\quad$ 1

$x_i + \bar{x}_i = 1$



a b

$$\begin{bmatrix} 1\ 1 & 1\ 1 \end{bmatrix}$$

1

one Altr implicant 1

f = ( ) + 1

f = 1

v

$$\begin{bmatrix}1\,0\\1\,1\\1\,0\\1\,0\\1\,1\end{bmatrix}$$

$$\begin{bmatrix} 0\ 1 \\ 0\ 1 \\ 0\ 1 \\ 0\ 1 \\ 0\ 1 \end{bmatrix}$$

v

00

$$\begin{bmatrix} 1\,0 \\ 1\,0 \\ 1\,0 \\ 1\,0 \\ 1\,0 \end{bmatrix}$$

$\bar{v}$

01 → 1

10 → 0

11 → * x

ab + bc

b(a+c)

No tautology

Recursive tautology

- TAUTOLOGY:
  - The cover matrix has a row of all 1s. (Tautology cube)
- NO TAUTOLOGY:
  - The cover has a column of 0s. (A variable never takes a value)
- TAUTOLOGY:
  - The cover depends on one variable, and there is no column of 0s in that field
- Decomposition rule:
  - When a cover is the union of two subcovers that depend on disjoint sets of variables, then check tautology in both subcovers

$b+\bar{b}=1$

01
10

01
01 x

Example
f = ab + ac + ab'c' + a'

$f_{\bar{c}} = 1$

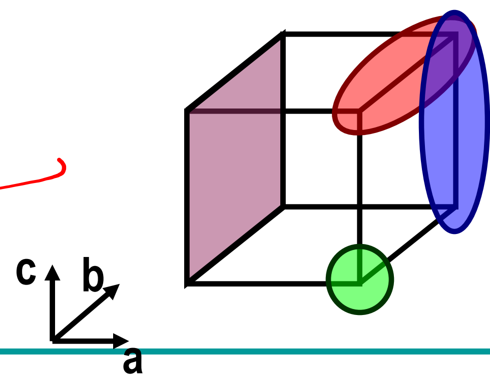
- Select variable a
- Cofactor w.r.t. to a' is

$f_{\bar{a}}$ 11 11 11 – Tautology.

	a	b	c
ab	01	01	11
ac	01	11	01
$a\bar{b}\bar{c}$	01	10	10
$\bar{a}$	10	11	11
	10	11	11
	00	01	11
	00	11	01
	00	10	10
	10	11	11
	01	00	00
	11	11	11

=

f_a



Example

f = ab + ac + ab'c' + a'

$f_{c'} = 1$

- Select variable a
- Cofactor w.r.t. to a' is

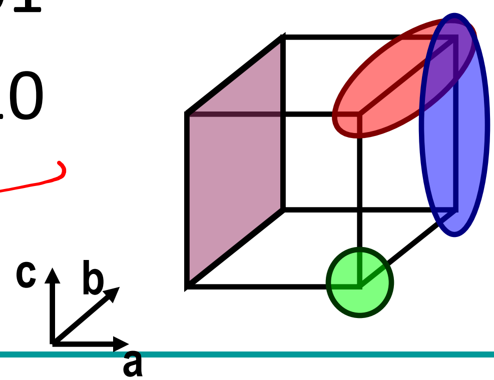
$f_{a'}$ 11 11 11 – Tautology.

- Cofactor w.r.t. to a is:

11	01	11
11	11	01
11	10	10

f_a

	a	b	c
ab	01	01	11
ac	01	11	01
ab'c'	01	10	10
a'	10	11	11
	01	11	11
	01	01	11
	01	11	01
	01	10	10
	00	11	11
	10	00	00
	11	01	11
	11	11	01
	11	10	10



Example (2)

11	01	11
11	11	01
11	10	10

- Select variable b.
- Cofactor w.r.t b' is

a b c

11	11	01
11	11	10

- No column of 0 - Tautology

11	01	11
11	11	01
11	10	10
11	10	11
11	00	11
11	10	01
11	10	10
00	01	00
11	11	01
11	11	10

$f_{\bar{c}} = 1$

$f_{ab} = 1$

$f_{a\bar{b}} = 1$

Example (2)

11	01	11
11	11	01
11	10	10

- Select variable b.
- Cofactor w.r.t b' is

a b c

11	11	01
11	11	10

- No column of 0 - Tautology
- Cofactor w.r.t b is:

11 11 11

- Function is a *TAUTOLOGY.*

11	01	11
11	11	01
11	10	10
11	01	11
11	01	11
11	00	01
11	00	10
00	10	00
11	11	11

$f_{\bar{c}} = 1$

$f_{a b} = 1$

$f_{a \bar{b}} = 1$

Thank You