Two-Level Logic Optimization: Fundamentals

Priyank Kalla

Associate Professor
Electrical and Computer Engineering, University of Utah
kalla@ece.utah.edu
http://www.ece.utah.edu/~kalla

Two-Level Logic Simplification

- Why minimize two-level logic?
  - Simpler implementation, also helps with multi-level logic optimization
- Representations: SOP, POS, tabular forms
- Implementations: PLAs
- Two level minimization improves: Area, Delay and Testability
- Limitation: Circuits can become large and slow

Algorithmic Issues

- Recall: Variable, literal, minterms and cubes (ON-set), implicants
- Without loss of generality, we will deal with implicants
- Cover of a function: a list of implicants that covers the function
- Hence the term: "minimum(al) cover of a function"
- We want large cubes: prime implicants
- We want fewest prime implicants that cover the function – cost issues
- In general: Cost of a two level implementation can be counted as the total number of SOP literals

What do we mean by Logic “Simplification”?

- Ideally, we want a minimum cover with minimum cost

What do we mean by Logic "Simplification"?

- Ideally, we want a minimum cover with minimum cost
- Minimal Cover: Irredundant cover, not contained in any other cover
    - No implicant contained in any subset of implicants of the cover

What do we mean by Logic "Simplification"?

- Ideally, we want a minimum cover with minimum cost
- Minimal Cover: Irredundant cover, not contained in any other cover
    - No implicant contained in any subset of implicants of the cover
- Minimal cover w.r.t. single implicant containment
    - Weaker property, where no single cube contained in any other cube
    - Also called single cube containment (SCC)
    - Ex: $SCC(a + ab) = a$

What do we mean by Logic "Simplification"?

- Ideally, we want a minimum cover with minimum cost
- Minimal Cover: Irredundant cover, not contained in any other cover
  - No implicant contained in any subset of implicants of the cover
- Minimal cover w.r.t. single implicant containment
  - Weaker property, where no single cube contained in any other cube
  - Also called single cube containment (SCC)
  - Ex: $SCC(a + ab) = a$
- Minimum cover: a cover of minimum cardinality (of implicants)

What do we mean by Logic "Simplification"?

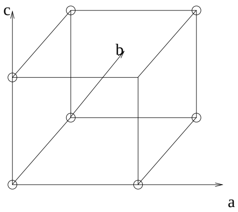
- Ideally, we want a minimum cover with minimum cost
- Minimal Cover: Irredundant cover, not contained in any other cover
  - No implicant contained in any subset of implicants of the cover
- Minimal cover w.r.t. single implicant containment
  - Weaker property, where no single cube contained in any other cube
  - Also called single cube containment (SCC)
  - Ex: $SCC(a + ab) = a$
- Minimum cover: a cover of minimum cardinality (of implicants)
- Minimum covers also with minimum cost (strongest condition!)

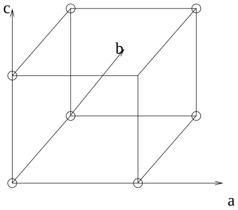
What do we mean by Logic “Simplification”?

- Ideally, we want a minimum cover with minimum cost
- Minimal Cover: Irredundant cover, not contained in any other cover
  - No implicant contained in any subset of implicants of the cover
- Minimal cover w.r.t. single implicant containment
  - Weaker property, where no single cube contained in any other cube
  - Also called single cube containment (SCC)
  - Ex: $SCC(a + ab) = a$
- Minimum cover: a cover of minimum cardinality (of implicants)
- Minimum covers also with minimum cost (strongest condition!)
- Prime implicants play an important role: **There exists a minimum cover consisting of only prime implicants.**

Example

- $f_1 = a'b'c' + a'b'c + ab'c + abc + abc'$ and $f_2 = ab'c + a'b'c$
- Single output cover f_1:
- Multi-Output Cover: F:





Primality and Irredundancy

- Making a cover prime and irredundant is good, but may not have minimum cost
- However, primality and irredundancy is still very important
  - For testability of two-level logic
  - Also applicable to heuristic minimization – i.e. when very large problems cannot be exactly minimized
- See example given in class on the effect of primality and irredundancy
- Prime and irredundant = 2-level logic fully testable! No "redundancy"
- Also see these examples:
  - $f(x_1, \ldots, x_4) = \sum m(0, 4, 8, 10, 11, 12, 13, 15)$
  - $f = x_3'x_4' + x_1, x_2, x_3' + x_1x_3x_4 + x_1x_2'x_3$ and $f = x_3'x_4' + x_1x_2x_4 + x_1x_2'x_3$
  - Are both implementations prime & irredundant?

Generate primes, and find minimum (cost) covers
$f = \sum m(0,3,4,5,7,9,11) + D(8,12,13,14)$

- One solution:
  $p_1 = (0,4,8,12), p_2 = (4,5,12,13), p_3 = (3,7), p_4 = (9,11)$. Cost = 4 primes, 10 literals
- Another solution:
  $p_1 = (0,4,8,12), p_2 = (5,7), p_3 = (3,11), p_4 = (9,11)$. Cost = 4 primes, 11 literals

So, the strongest problem formulation is: Find a minimum cost cover from among the prime implicants that contains a minimum number of primes!

Prime Implicant Generation

- For a n variable Boolean function, can have $3^n/n$ primes, and 2^n minterms in the worst case
- Generation of primes is a challenge
- One approach: Prime implicant table generation, see any undergrad digital logic textbook, also shown in class
- A more interesting way to generate primes using Shannon's expansion: $f = xf_x + x'f'_x$
- Prime of f can be:
  - A prime of xf_x;
  - Or, a prime of $x'f_{x'}$;
  - Or, the "consensus of two implicants", one in xf_x and one in $x'f_{x'}$
- What is the "consensus of two implicants"?

Consensus of two Implicants

- Given two implicants α, β, $\mathcal{CONSENSUS}(\alpha, \beta)$ is the single largest cube contained in their union, but not contained in either of them.
- If Hamming distance between $\alpha, \beta \geq 2$, $\mathcal{CONSENSUS}(\alpha, \beta)$ is void.
- Given a variable x and two cubes A, B, $\mathcal{CONSENSUS}(xA, x'B) = AB$

Recursive Approach to Prime Computation

$$P(f) = \mathcal{SCC}[x \cdot P(f_x) \cup x' \cdot P(f_{x'}) \cup \mathcal{CONSENSUS}(x \cdot P(f_x),\ x' \cdot P(f_{x'}))]$$

- Notation: $P(f)$ denotes the set of primes of f
- Pick a variable x for branching, x should be the highest binate variable
- Compute primes in $x \cdot P(f_x), x' \cdot P(f_{x'})$ and then in their consensus.
- $\mathcal{SCC}$ operator is needed because primes in $x \cdot P(f_x)$ or $x' \cdot P(f_{x'})$ may be contained in their consensus
- When does recursion bottom out?
  - When a (cofactor) cover f is a single implicant, $P(f) = f$
  - When a (cofactor) cover f is **strongly unate in all variables**, $P(f) = \mathcal{SCC}(f)$ (Can you prove it?)
- Original cover of f may not be unate, but cofactors generally tend to be unate, so exploit unateness.

Unate Covers versus Unate Functions

- Recall, $f = a'bc + ab'c + abc' + abc$ is a unate function (majority function) w.r.t. all variables
- When f is minimized, $f = ab + ac + bc$, this **cover is unate**
- Unate covers imply unate functions
- Unate functions do not always have unate covers
- Checking for unate covers is easier: just check the polarity of each variable for (+ve or -ve) unateness
  - Unate cover: Every variable x appears either in only positive polarity, or only in negative polarity
- Checking for unate functions: $f_x \supseteq f_{x'}$ or vice-versa; this containment check is harder
- So, for unate recursive paradigm, we operate mostly on unate covers
- In general, to avoid confusion, use f for function and F for its cover
  - $f = ab + ac + ac, \quad F = \{ab, ac, bc\}$

Now read Chapter 4 from the textbook... for solving the table covering problem