ECE 274 - Digital Logic
Lecture 20

Lecture 20 - Optimization

Two-Level Minimization
  - Exact and Heuristic Algorithms

Automating Two-Level Logic Size Minimization

Minimizing by hand
  - Is hard for functions with 5 or more variables
  - May not yield minimum cover depending on order we choose
  - Is error prone
Minimization thus typically done by automated tools
  - Exact algorithm finds optimal solution
  - Heuristic finds good solution, but not necessarily optimal

Basic Concepts Underlying Automated Two-Level Logic Minimization

Definitions
  - On-set: All minterms that define when \( F = 1 \)
  - Off-set: All minterms that define when \( F = 0 \)
  - Implicant: Any product term (minterm or other) that when 1 causes \( F = 1 \)
  - On K-map, any legal (but not necessarily largest) circle
  - Cover: Implicant \( x'y'z \) covers minterms \( x'y'z \) and \( x'y'z' \)
  - Expanding a term: removing a variable (like larger K-map circle)
    - \( xyz \rightarrow xy \) is an expansion of \( xyz \)

Definitions (cont)
  - Essential prime implicant: The only prime implicant that covers a particular minterm in a function's on-set
  - Importance: We must include all essential PIs in a function's cover
  - In contrast, some, but not all, non-essential PIs will be included

Automated Two-Level Logic Minimization Method

1. Determine all prime implicants
2. Add essential PIs to cover
   - Italicized 1s are thus already covered
   - Only one uncovered 1 remains
3. Cover remaining minterms with non-essential PIs
   - Pick among the two possible PIs

Example of Automated Two-Level Minimization

Steps 1 and 2 are exact
Problem with Methods that Enumerate all Minterms or Compute all Prime Implicants

- Too many minterms for functions with many variables
  - Function with 32 variables: \(2^{32} = 4\) billion possible minterms.
  - Too much compute time/memory
- Too many computations to generate all prime implicants
  - Comparing every minterm with every other minterm, for 32 variables, is \((4 \text{ billion})^2 = 1\) quadrillion computations
  - Functions with many variables could require days, months, years, or more of computation – unreasonable

Solution to Computation Problem

- Solution
  - Don’t generate all minterms or prime implicants
  - Instead, just take input equation, and try to “iteratively” improve it

Ex: \(F = \text{abcdefg} + \text{abcdefh} + \text{jklmnop}\)

- Note: 15 variables, may have thousands of minterms
  - But can minimize just by combining first two terms:
    \(F = \text{abcdefg} + \text{abcdefh} + \text{jklmnop} = \text{abcdef} + \text{jklmnop}\)

Two-Level Minimization using Iterative Method

- Method: Randomly apply “expand” operations, see if helps
  - Expand: remove a variable from a term
    - Like expanding circle size on K-map
      - e.g., Expanding \(x'z\) to \(2\) legal, but expanding \(x'z\) to \(2'\) not legal, in shown function
      - After expand, remove other terms covered by newly expanded term
    - Keep trying (iterate) until doesn’t help
  - Ex:
    \(F = \text{abcdefg} + \text{abcdefh} + \text{jklmnop}\)
    \(F = \text{abcdefg} + \text{abcdefh} + \text{jklmnop}\)
    \(F = \text{abcdefg} + \text{jklmnop}\)

Multi-Level Logic Optimization – Performance/Size Tradeoffs

- We don’t always need the speed of two level logic
  - Multiple levels may yield fewer gates
    - Example
      \(F_1 = ab + acd + ace\) \(\rightarrow\) \(F_2 = ab + ac(d + e) = ab + (c(d + e))\)
    - General technique: Factor out literals – \(xy + xz = x(y+z)\)

- Ex:
  - \(F_1 = abcd + abcef\)
    - Has fewer gate inputs, thus fewer transistors
      \(\rightarrow\) \(F_2 = abcd + abcef\)

Multi-Level Example

- Q: Use multiple levels to reduce number of transistors for
  \(F_1 = abcd + abcef\)
- A: \(ab cd + abce f = ab(c(d + ef))\)
  - Has fewer gate inputs, thus fewer transistors

- Ex:
  - \(F_1 = abcd + abcef\)
    - \(\rightarrow\) \(F_2 = abcd + abcef\)

Multi-Level Example: Non-Critical Path

- Critical path: longest delay path to output
- Optimization: reduce size of logic on non-critical paths by using multiple levels
Automated Multi-Level Methods

- Main techniques use heuristic iterative methods
  - Define various operations
    - "Factor out": \( xy + xz = x(y+z) \)
  - Expand, and others
  - Randomly apply, see if improves
    - May even accept changes that worsen, in hopes eventually leads to even better equation
  - Keep trying until can't find further improvement
  - Not guaranteed to find best circuit, but rather a good one

Design Challenge

- For the function \( F(a,b,c,d) = ab'c' + abc'd + abcd + a'bcd' \), determine all prime implicants, and all essential prime implicants.
- Use a heuristic method to obtain a two-level size optimized function expressed in sum-of-products form.

Due:
- Next Lecture (Monday, November 21)
- Extra Credit / Homework
  - 2 points