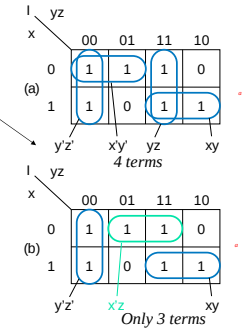
ECE 274 - Digital Logic

Lecture 20

- Lecture 20 – Optimization
  - Two-Level Minimization
    - Exact and Heuristic Algorithms

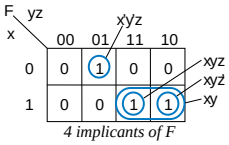
Automating Two-Level Logic Size Minimization

- 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 xy ***covers*** minterms xyz and xyz'
  - ***Expanding*** a term: removing a variable (like larger K-map circle)
    - xyz → xy is an expansion of xyz



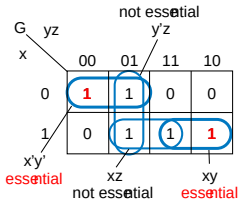
4 implicants of F

*Note: We use K-maps here just for intuitive illustration of concepts; automated tools do **not** use K-maps.*

- ***Prime implicant***: Maximally expanded implicant – any expansion would cover 1s not in on-set
  - x'y'z, and xy, above
  - But not xyz or xyz' – they can be expanded

Basic Concepts Underlying Automated Two-Level Logic Minimization

- 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



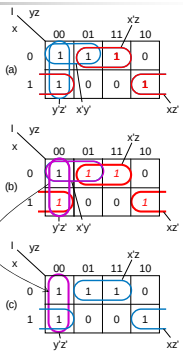
Automated Two-Level Logic Minimization Method

Step	Description
1 *Determine prime implicants*	For every minterm in the function's on-set, maximally expand the term (meaning eliminate literals from the term) such that the term still only covers minterms in the function's on-set (like drawing the biggest circle possible around each 1 in a K-map). Repeat for each minterm. If don't cares exist, use them to maximally expand minterms into prime implicants (like using X's to create the biggest circles possible for a given 1 in a K-map).
2 *Add essential prime implicants to the function's cover*	Find any minterms covered by only one prime implicant (i.e., by an essential prime implicant). Add those prime implicants to the cover, and mark the minterms covered by those implicants as already covered.
3 *Cover remaining minterms with nonessential prime implicants*	Cover the remaining minterms using the minimal number of remaining prime implicants.

- Steps 1 and 2 are exact
- Step 3: Hard. Checking all possibilities: exact, but computationally expensive. Checking some but not all: heuristic.

Example of Automated Two-Level Minimization

- 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



Problem with Methods that Enumerate all Minterms or Compute all Prime Implicants

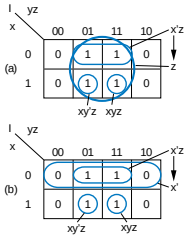
- Too many minterms for functions with many variables
  - Function with 32 variables:
    - 2^{32} = 4 billion possible minterms.
    - Too much compute time/memory
- Too many computations to generate all prime implicants
  - Comparing every minterm with every other minterm, for 32 variables, is (4 billion)^2 = 1 quadrillion computations
  - Functions with many variables could requires 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 = abcdefgh + abcdefgh'+ jklmnop
    - Note: 15 variables, may have thousands of minterms
    - But can minimize just by combining first two terms:
      - F = abcdefg(h+h') + jklmnop = abcdefg + jklmnop

Two-Level Minimization using Iterative Method

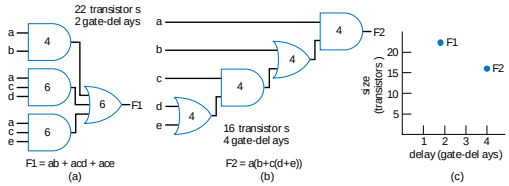
- Method: Randomly apply "expand" operations, see if helps
  - Expand: remove a variable from a term
    - Like expanding circle size on K-map
      - e.g., Expanding x'z to z legal, but expanding x'z to z' not legal, in shown function
      - After expand, remove other terms covered by newly expanded term
  - Keep trying (iterate) until doesn't help



Ex:
F = abcdefgh + abcdefgh'+ jklmnop
F = abcdefg + abcdefgh' + jklmnop
F = abcdefg + jklmnop

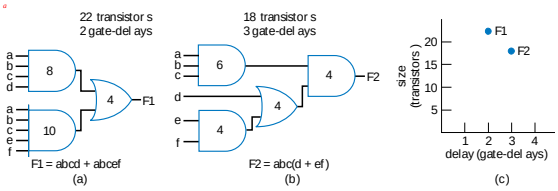
Multi-Level Logic Optimization – Performance/Size Tradeoffs

- We don't always need the speed of two level logic
  - Multiple levels may yield fewer gates
  - Example
    - F1 = ab + acd + ace → F2 = ab + ac(d + e) = a(b + c(d + e))
    - General technique: Factor out literals – xy + xz = x(y+z)



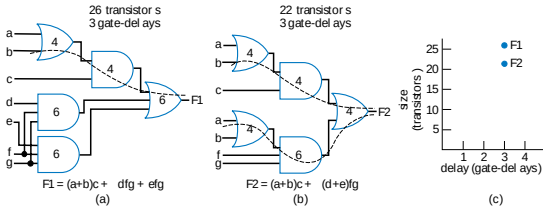
Multi-Level Example

- Q: Use multiple levels to reduce number of transistors for
  - F1 = abcd + abcef
- A: abcd + abcef = abc(d + ef)
  - Has fewer gate inputs, thus fewer transistors



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

- Design Challenge
  - For the function *F(a,b,c,d)=ab'c' + abc'd + abcd + a'bcd + 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