Today's material might at first appear difficult
Perhaps even a bit dry
. . . but follow closely
Trust me, if you really get it, there is great depth and beauty here

Goal: two-level logic realizations with fewest gates and fewest number of gate inputs
Algebraic
Karnaugh map
Quine–McCluskey
Espresso heuristic

For difficult and large functions, solve by heuristic search
Multi-level logic minimization is also best solved by search
The general search problem can be introduced via two-level minimization
* Examine simplified version of the algorithms in Espresso

If \( g \) and \( h \) are two Boolean functions s.t. the on-set of \( g \) is a subset of the on-set of \( h \) then
* \( h \) covers \( g \) or . . .
* . . . \( g \subseteq h \)
Espresso pseudocode

Procedure Espresso(F, D, R)
1. /* F is ON set, D is don't care, R OFF */
2. R = Comp(F + D); /* Compute complement */
3. F = Expand(F, R); /* Initial expansion */
4. F = Irredundant(F, D); /* Initial irredundant cover */
5. E = Essentials(F, D); /* Extracting essential primes */
6. F = F - E; /* Remove essential primes from F */
7. D = D + E; /* Add essential primes to D */
8. while Cost(F) keeps decreasing do
9. F = Reduce(F, D); /* Perform reduction, heuristic which cubes */
10. F = Expand(F, R); /* Perform expansion, heuristic which cubes */
11. F = Irredundant(F, D); /* Perform irredundant cover */
12. end while
13. F = F + E;
14. return F;

Espresso moves

- Add a literal to a cube (reduce)
- Remove a literal from a cube (expansion)
- Remove redundant cubes (irredundant cover)

Irredundant functions need not be minimal

- Sometimes necessary to increase cost to escape local minima
- Add a literal to a cube (reduce)
- To later allow expansion in another dimension

Espresso algorithm

Repeat the following

1. **Reduce** sometimes necessary to contain cubes within others
   - Another cover with fewer terms or fewer literals might exist
   - Shrink prime implicants to allow expansion in another variable
2. **Irredundant Cover** is extracted from the expanded primes
   - Similar goals to the Quine-McCluskey prime implicant chart
   - Good performance requires a few tricks
3. **Expand** implicants to their maximum size
   - Implicants covered by an expanded implicant are removed from further consideration
   - Quality of result depends on order of implicant expansion
   - Heuristic methods used to determine order

Espresso example

Irredundant but not minimal Reduce Expand Irredundant Cover to find alternative prime implicants
Keep doing this as long as new covers improve on last solution
A number of optimizations are tried, e.g., identify and remove essential primes early in the process

Redundancy in Boolean space

- If a formula contains \( AB \) and \( \overline{B} \), \( AB \subseteq \overline{B} \Rightarrow AB \) is redundant
- Sometimes redundancy is difficult to observe
  - If \( F = BC + AB + AC \), then \( AB \) is redundant
- Redundancy in Boolean space

Espresso example

Irredundant but not minimal Reduce Expand Irredundant Cover to find alternative prime implicants
Keep doing this as long as new covers improve on last solution
A number of optimizations are tried, e.g., identify and remove essential primes early in the process

Homework

Definitions
Tautology checking
Review of logic minimization
Irredundant Cover
Sometimes necessary to contain cubes within others
Sometimes necessary to increase cost to escape local minima
Sometimes redundant is difficult to observe
Add a literal to a cube (reduce)
Remove a literal from a cube (expansion)
Remove redundant cubes (irredundant cover)
Irredundant functions need not be minimal
Sometimes necessary to increase cost to escape local minima
Add a literal to a cube (reduce)
To later allow expansion in another dimension
Irredundant but not minimal Reduce Expand Irredundant Cover to find alternative prime implicants
Keep doing this as long as new covers improve on last solution
A number of optimizations are tried, e.g., identify and remove essential primes early in the process

Irredundant but not minimal Reduce Expand Irredundant Cover to find alternative prime implicants
Keep doing this as long as new covers improve on last solution
A number of optimizations are tried, e.g., identify and remove essential primes early in the process

Homework

Definitions
Tautology checking
Review of logic minimization
Irredundant Cover
Sometimes necessary to contain cubes within others
Sometimes necessary to increase cost to escape local minima
Sometimes redundant is difficult to observe
Add a literal to a cube (reduce)
Remove a literal from a cube (expansion)
Remove redundant cubes (irredundant cover)
Irredundant functions need not be minimal
Sometimes necessary to increase cost to escape local minima
Add a literal to a cube (reduce)
To later allow expansion in another dimension
Irredundant but not minimal Reduce Expand Irredundant Cover to find alternative prime implicants
Keep doing this as long as new covers improve on last solution
A number of optimizations are tried, e.g., identify and remove essential primes early in the process

Homework

Definitions
Tautology checking
Review of logic minimization
Irredundant Cover
Sometimes necessary to contain cubes within others
Sometimes necessary to increase cost to escape local minima
Sometimes redundant is difficult to observe
Add a literal to a cube (reduce)
Remove a literal from a cube (expansion)
Remove redundant cubes (irredundant cover)
Irredundant functions need not be minimal
Sometimes necessary to increase cost to escape local minima
Add a literal to a cube (reduce)
To later allow expansion in another dimension
Irredundant but not minimal Reduce Expand Irredundant Cover to find alternative prime implicants
Keep doing this as long as new covers improve on last solution
A number of optimizations are tried, e.g., identify and remove essential primes early in the process
Tautology check for relatively essential cubes

- $c$ is a 1-cube
- Check to see whether the union of 1-cubes and don't-care cubes minus $c$, cofactored by $c$, is a tautology
- Let $A$ be the set of 1-cubes
- Let $D$ be the set of don't-care cubes
- $(A \cup D) - c \neq 1$ if $c$ is relatively essential
- That's it! You can use tautology checking to determine whether a cube is relatively essential
- Of course, an example would make it clearer
Detecting relatively essential cubes

- How to determine whether a cube is fully covered by other 1 and don't-care cubes?
- Could decompose everything to minterm canonical form
- Recall that there may be 2^n minterms, given n variables
- Decomposition is a bad idea
  - Exponential

**Unate functions**

- $f(x_1, x_2, \ldots, x_n)$ is monotonically increasing in $x_1$ if and only if $\forall x_2, \ldots, x_n: f(0, x_2, \ldots, x_n) \leq f(1, x_2, \ldots, x_n)$
- $f(x_1, x_2, \ldots, x_n)$ is monotonically decreasing in $x_1$ if and only if $\forall x_2, \ldots, x_n: f(0, x_2, \ldots, x_n) \geq f(1, x_2, \ldots, x_n)$
- A function that is neither monotonically increasing or monotonically decreasing in $x_1$ is non-monotonic in $x_1$
- A function that is monotonically increasing or monotonically decreasing in $x_1$ is unate in $x_1$
- A function that is unate in all its variables is unate

Recursive pivoting?

- Could also recursively pivot on variables if inclusion fails
  
  \[
  \begin{align*}
  000 & | 1 \\
  001 & | 01X \\
  010 & | 011 \end{align*}
  \]

  - Let's us terminate recursion as soon as cube is covered by single other cube, e.g., 01X
  - However, even with pruning, this is still slow in practice
  - Worst-case time complexity?

Definition: Cofactor by variable

$$f_{x_1} = f(1, x_2, \ldots, x_n)$$

$$f_{x_2} = f(0, x_2, \ldots, x_n)$$

Note that it's commutative,

$$(f_{x_1})_y = (f_{x_2})_y$$

Problem conversion

**Thus, we have taken the problem**

Determine whether a cube, $c$, is covered by a set of 1-cubes, $A$, or don't-care, $D$, cubes.

and converted it to

Determine whether a set of 1-cubes, $A$, and don't-care cubes, $D$, cofactored by cube $c$ is a tautology.

Conversion benefits

- Cofactoring eliminates variables, speeding analysis
- Tautology is a straight-forward and well-understood problem
- However, tautology checking is not easy
  - Could pivot on all variables... but this is too slow
  - Example?

Unate functions

$0$ $a$ $b$ $1$ $c$

Unate

0 $b$ 1

a

0 $c$ 1
Unate covers

- Unate functions are difficult to identify
- A cover is unate as long as the complemented and uncomplemented literals for the same variable do not both appear
- Identifying unate covers is easy

Identifying unate covers is easy

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

- Scan the columns for the presence of a 0 and 1
- Note, some unate functions can have non-unate covers
  - Unate covers always express a unate function

Fast tautology checking using unate covers

Given that $C$ is a cover containing cubes composed of variables $x_1, x_2, \ldots, x_n$

Let $T(A)$ is tautology if and only if it contains a 1, i.e., $XX:1$

<table>
<thead>
<tr>
<th>x_1</th>
<th>x_2</th>
<th>\ldots</th>
<th>x_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>\ldots</td>
<td>c</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>\ldots</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>\ldots</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>\ldots</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>\ldots</td>
<td>1</td>
</tr>
</tbody>
</table>

- Assume $C = x_1 \cdot F_1(x_2, \ldots, x_n) + F_2(x_2, \ldots, x_n)$
- Then the $C_a$ cofactor is $F_1(x_2, \ldots, x_n)$
- The $C_\tau$ cofactor is $F_2(x_2, \ldots, x_n)$

Example of unate cofactoring

$\bar{c} + bc + \bar{c}\bar{\tau}\bar{c}$

Cover unate only in $a$

$(\bar{c} + bc + \bar{c}\bar{\tau}\bar{c})_a = bc$

$(\bar{c} + bc + \bar{c}\bar{\tau}\bar{c})_a = c + bc + \bar{c}\bar{\tau}\bar{c}$

Notice anything nice about this?
Fast tautology checking using unate covers

\[ C_1 = F_1(x_2, \ldots, x_n) + F_2(x_2, \ldots, x_n) \]
\[ C_\pi = F_2(x_2, \ldots, x_n) \]

Clearly,
\[ C_\pi \subseteq C_1 \]

Therefore we need only consider \( C_\pi \) for tautology checking, significantly simplifying the problem, i.e., if \( C_\pi \) is a tautology, then \( C_1 \) is obviously also a tautology.

Summary: Fast tautology checking

- Identify unate covers
  - No columns with 1s and zeros
- If unate, scan for an XXX|1 row
- If not unate, cofactor on (preferable) unate variable
- Only need to consider uncomplemented or uncomplemented cofactor
  - Why? \( F_2 \leq F_1 + F_2 \)

Two-level heuristic minimization summary

- Generating all prime implicants can be too expensive
- Make incremental changes: \textsc{Expand, Reduce, and Irredundant Cover}
- Determining whether incremental change represents same function is too expensive
- Use cofactoring to convert it to a tautology check
- Use unateness to make the tautology check fast in most cases

CAD

Computer–Aided Design (of Integrated Circuits and Systems)
Also called Electronics Design Automation (EDA)
Without it, computers wouldn’t work

Questions

- What is the unate covering problem?
- Where have we seen it used?
- What can tautology checking be used for?
- How do we make it fast?
Next lecture: Implementation technologies

Reading assignment

- Chapter 4
- VLSI Programmable Logic Devices, document 1

- PALs, PLAs
- MUX, DEMUX review
- Steering logic

Reading assignment

- Chapter 4
- VLSI Programmable Logic Devices, document 1

- PALs, PLAs
- MUX, DEMUX review
- Steering logic