ESE535: Electronic Design Automation

Day 19: April 6, 2015
Two-Level Logic-Synthesis

Today

• Two-Level Logic Optimization
  – Problem
  – Definitions
  – Basic Algorithm: Quine-McCluskey
  – Improvements

Problem

• **Given**: Expression in combinational logic
• **Find**: Minimum (cost) sum-of-products expression
• Ex.
  – $Y = a*b*c + a*b'*c + a'*b*c$
  – $Y = a*b + a*c$

EDA Use

• Minimum size PLA, PAL, …
  – Programmable Logic Array
  – Programmable Array Logic
• Minimum number of gates for two-level implementation
• Starting point for multi-level optimization

Programmable Logic Arrays (PLAs)

Pla

• Directly implement flat (two-level) logic
  – $O = a*b*c*d + a*b'*d + b'*c*d$
• Exploit substrate properties allow wired-OR
**Wired-or**

- Connect series of inputs to wire
- Any of the inputs can drive the wire high

**Wired-or**

- Obvious Implementation with Transistors

**Programmable Wired-or**

- Use some memory function to programmable connect (disconnect) wires to OR
- Fuse:

**Programmable Wired-or**

- Memory configurable PLA model

**Diagram Wired-or**

- Build into array
  - Compute many different or functions from set of inputs
Combined or-arrays to PLA

- Combine two or (nor) arrays to produce PLA (or-and / and-or array)

PLA

- Can implement each and on single line in first array
- Can implement each or on single line in second array

Strictly speaking: or in first term and in second, but with both polarities of inputs, can invert so is and-or.

Nanowire PLA

PLA and PAL

PAL = Programmable Array Logic
PAL has fixed OR plane.

EDA Use for 2-level Logic Min.

- Minimum size PAL, PLA, …
  - Programmable Logic Array
  - Programmable Array Logic
- Minimum number of gates for two-level implementation
- Starting point for multi-level optimization

...back to optimization...
Complexity

- Set covering problem
  - NP-hard

Terminology (1)

- Literals -- a, /a, b, /b, ....
  - Qualified, single inputs
- Minterms --
  - full set of literals covering one input case
  - in y=a*b+a*c
    - a*b*c
    - a*/b*c

Product Term

- Each product = row in PLA is known as a **Product Term** or **PTERM**
- With fixed number of inputs and outputs, area is determined by the number of PTERMs needed

Cost

- PLA/PAL – to first order costs is:
  - number of product terms
- Abstract (mis, sis)
  - (multilevel,sequential) interactive synthesis
  - number of literals
    - $\text{cost}(y=a*b+a*c )=4$
- General (simple, multi-level)
  - $\sum \text{cost(product-term)}$
    - e.g. nand2=4, nand3=5, nand4=6...

Terminology (2)

- Cube:
  - product covering one or more minterms
  - $Y=a*b+a*c$
  - cubes:
    - $a*b*c$ abc
    - $a*b$ ab
    - $a*c$ ac
Terminology (3)

- Cover:
  - set of cubes
  - sum products
  - \{abc, a/bc, ab/c\}
  - \{ab, ac\}

Truth Table

- Also represent function

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

Specify on-set only

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

Cube/Logic Specification

- Canonical order for variables
- Use \{0,1,-\} to indicate input appearance in cube
  - 0 = inverted \ abc \ 111
  - 1 = not inverted \ a/bc \ 101
  - - = not present \ ac \ 1-1

In General

- Three sets:
  - on-set (must be set to one by cover)
  - off-set (must be set to zero by cover)
  - don’t care set (can be zero or one)
- Don’t Cares
  - allow freedom in covering (reduce cost)
  - arise from cases where value doesn’t matter
  - e.g. outputs in non-existent FSM state
  - data bus value when not driving bus

Multiple Outputs

<table>
<thead>
<tr>
<th>Truth Table:</th>
<th>Convert to single-output problem</th>
<th>On-set for result</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Multiple Outputs

- Can reduce to single output case
  - write equations on inputs and each output
    - with onset for relation being true
    - after cover
    - remove literals associated with outputs
Multiple Outputs

- Could Optimize separately
- By optimizing together
  - Maximize sharing of cubes/product-terms

Consider:
- $X = \overline{a}b + ab + ac$
- $Y = bc$

Now read off cover:
- $Y = \overline{bc}$
- $A = \overline{a}b/c + \overline{bc} + ab$
  
  = $\overline{a}/b + \overline{bc} + ab$

Only need 3 product terms (versus 4 w/ no sharing)

Prime Implicants

- Implicant -- cube in on-set
  - (not entirely in don’t-care set)
- Prime Implicant -- implicant, not contained in any other cube
  - for $y = a*b + a*c$
    - $a*b$ is a prime implicant
    - $a*b*c$ is not a prime implicant (contained in $ab$, $ac$)
  - i.e. largest cube still in on-set (on+dc-sets)

Minimum cover will be made up of primes
- fewer products if cover more
- fewer literals in prime than contained cubes

Necessary but not sufficient that minimum cover contain only primes
- $y = ab + ac + bc$
- $y = ac + bc$

Number of PI’s can be exponential in input size
- more than minterms, even!
- Not all PI’s will be in optimum cover

Restate Goal

- Goal in terms of PIs
  - Find minimum size set of PIs that cover the on-set.
Essential Prime Implicants

- Prime Implicant which contains a minterm not covered by any other PI
  - Essential PI must occur in any cover
  - $y = ab + ac + b/c$
  - $ab \ 11\ - \ 110 \ 111$
  - $ac \ 1-1 \ 101 \ 111$  
  * essential (only 101)
  - $b/c -10 \ 110 \ 010$
  * essential (only 010)

Computing Primes

- Start with minterms
  - for on-set and dc-set
- merge pairs (distance one apart)
- for each pair merged,
  - mark source cubes as covered
- repeat merging for resulting cube set
  - until no more merging possible
- retain all unmarked cubes which aren’t entirely in dc-set

Compute Prime Example

<table>
<thead>
<tr>
<th></th>
<th>0000</th>
<th>0101</th>
<th>0111</th>
<th>1000</th>
<th>1001</th>
<th>1010</th>
<th>1101</th>
<th>1110</th>
<th>1111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

(in-class assignments, back of preclass sheet; record solutions on board.)

Covering Matrix

<table>
<thead>
<tr>
<th>Minterms $\times$ Prime Implicants</th>
</tr>
</thead>
</table>

Goal: minimum cover

<table>
<thead>
<tr>
<th></th>
<th>0000</th>
<th>0101</th>
<th>0111</th>
<th>1000</th>
<th>1001</th>
<th>1010</th>
<th>1101</th>
<th>1110</th>
<th>1111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Essential Reduction

- Must pick essential PI
  - pick and eliminate row and column

<table>
<thead>
<tr>
<th></th>
<th>a/b/c/d</th>
<th>a/b/d</th>
<th>a/b/c</th>
<th>a/c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Which essential?

- This case:
  - Cover determined by essentials
  - Preclass 3: ac+a/b+/abd+/b/c/d

- General case:
  - Reduces size of problem

Dominators: Column

- If a column (PI) covers the same or strictly more than another column
  - can remove dominated column

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C dominates B

Any others?

G dominates H

Dominators: Row

- If a row has the same (or strictly more) PIs than another row, the larger row dominates
  - we can remove the dominating row
  - (NOTE OPPOSITE OF COLUMN CASE)

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C,D now essential

What now?

C,G now essential

New Essentials

- Dominance reduction may yield new Essential PIs
Cyclic Core

- After applying reductions
  - essential
  - column dominators
  - row dominators
- May still have a non-trivial covering matrix

  How do we move forward from here?

Example

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Example

A in Cover: What now?

C dominates B

G dominates H

Example

A not in Cover: What now?

B dominates C

D dominates E

F dominates G

H dominates I

9
Basic Two-Level Minimization (espresso-exact)

• Generate Prime Implicants
• Reduce (essential, dominators)
• If not done,
  – pick a cube
  – branch (back to reduce) on selected/not
    • *i.e.* search tree … branch and bound
• Save smallest

Branching Search

Branching Search w/ Implications

Covering Technique?

Optimization

Heuristic Variant

Implications Prune Tree

(like BCP in SAT)

Only exponential in decision where must branch

• Possibly useful for dataflow subgraph selection? (Day 17)
  – Treat application components as rows (minterms)
  – Treat patterns as columns (PIs)
• But, more general (complicated) cost model

• Summarize Minterms (signature cubes)
  – rows represent collection of minterms with same primes
• Avoid generating full set of PIs
  – pre-combining dominators during generation
• Branch-and-bound pruning
  – get lower bound on remaining cost of a cover by computing independent set of primes
    • (not necessarily maximal, that would be NP-hard)

• Don’t backtrack when select prime for inclusion/exclusion
  – pick cover large set of minterms/signatures
  – weight to select “hard” to cover signatures
• Generate reduced set of PIs
• Iterative improvement
**Canonical Form**

- Can start with *any* form of logical expression
- Get unique truth-table/minterms
- Problem not sensitive to input statement
  - compare covering (decomposition)
  - compare sequential programming languages
- **Cost:** potentially exponential explosion in minterms/PIs

**Summary**

- Formulate as covering problem
- Solution space restricted to PIs
- Essentials must be in solution
- Use dominators to further reduce space
- Then branching/pruning to explore rest of PIs
- Ways to reduce work
  - group minterms/PIs together early
  - mostly fall into this general scheme

**Big Ideas**

- **Canonical Form**
  - eliminate bias of input specification
- **Technique:**
  - branch-and-bound
  - pruning search – exploit structure
  - Dominators

**Admin**

- Reading for Wednesday online (web)