Two-Level Logic-Synthesis

Problem

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

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
  – \( O = a \cdot b \cdot c \cdot d + !a \cdot b \cdot !d + b \cdot !c \cdot d \)
• Exploit substrate properties allow wired-OR
Wired-or

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

Programmable Wired-or

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

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

Cost

• PLA/PAL – to first order costs is:
  – number of product terms
• Abstract (mis, sis)
  – {multilevel,sequential} interactive synthesis
  – number of literals
    • cost(y=a*b+a*/c )=4
• General (simple, multi-level)
  – Σcost(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
  Specify on-set only
  
  | a | b | c | y | Specify on-set only
  | a | b | c | y |
  0 0 0 0
  0 0 1 0
  0 1 0 0
  0 1 1 0
  1 0 0 0
  1 0 1 1
  1 1 0 1
  1 1 1 1
Cube/Logic Specification

- Canonical order for variables
- Use \{0,1,-\} to indicate input appearance in cube
  - 0 = inverted
  - 1 = not inverted
  - - = not present

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

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

- 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

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

Multiple Outputs

- Consider:
  - \(X = ab + ab + ac\)
  - \(Y = bc\)
- Trivial solution
  - has 4 product terms

Truth Table:

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

Convert to single-output problem

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

On-set for result

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

Penn ESE535 Spring 2013 -- DeHon
Multiple Outputs

• Consider:
  – \( X = a/b + ab + ac \)
  – \( Y = bc \)

• Now read off cover:
  – \( Y = bc \)
  – \( A = a/b + bc + ab \)

  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)

Prime Implicants

• 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
  – \( y = ab + ac + b/c \)
  – \( y = ac + b/c \)

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

Essential Prime Implicants

• Prime Implicant which contains a minterm not covered by any other PI
  – Essential PI must occur in any cover
  – \( y = ab + ac + b/c \)
  – \( ab = 1111 \)
  – \( ac = 110111 \)
  – \( b/c = -1011010 \)

  * essential (only 101)
  * essential (only 010)

Restate Goal

• Goal in terms of PIs
  – Find minimum size set of PIs that cover the on-set.

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

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

Note this is preclass 3.

<table>
<thead>
<tr>
<th>Value</th>
<th>Minterm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
</tr>
<tr>
<td>11</td>
<td>1011</td>
</tr>
<tr>
<td>14</td>
<td>1110</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
</tr>
</tbody>
</table>

Compute Prime Example

\[
\begin{array}{cc}
0, 8 & \text{-000} \\
5, 7 & \text{01-1} \\
7, 15 & \text{-111} \\
8, 9, 10, 11 & \text{10--} \\
8, 10, 14 & \text{10-0} \\
9, 11 & \text{10-1} \\
10, 11 & \text{101-} \\
10, 14 & \text{110-} \\
10, 15 & \text{111-} \\
11, 15 & \text{111-} \\
14, 15 & \text{111-} \\
\end{array}
\]

Covering Matrix

- **Minterms x Prime Implicants**
- **Goal:** minimum cover

<table>
<thead>
<tr>
<th>Minterm</th>
<th>Prime Implicants</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>X</td>
</tr>
<tr>
<td>0101</td>
<td>X</td>
</tr>
<tr>
<td>0111</td>
<td>X</td>
</tr>
<tr>
<td>1000</td>
<td>X</td>
</tr>
<tr>
<td>1010</td>
<td>X</td>
</tr>
<tr>
<td>1011</td>
<td>X</td>
</tr>
<tr>
<td>1100</td>
<td>X</td>
</tr>
<tr>
<td>1101</td>
<td>X</td>
</tr>
<tr>
<td>1111</td>
<td>X</td>
</tr>
</tbody>
</table>

Essential Reduction

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

<table>
<thead>
<tr>
<th>Prime Implicant</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>b/c/d</td>
<td>X</td>
</tr>
<tr>
<td>/abd</td>
<td>X</td>
</tr>
<tr>
<td>bcd</td>
<td>X</td>
</tr>
<tr>
<td>a/b</td>
<td>X</td>
</tr>
<tr>
<td>ac</td>
<td>X</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>X</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>1010</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C dominates B

<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>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Any others?

G dominates H

New Essentials

- Dominance reduction may yield new Essential PIs

<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></td>
<td></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>1110</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C,G now essential

What’s essential?

<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></td>
<td></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>1110</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What now?

E dominates D and F

Cover = {C,E,G}

Dominators: Row

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

<table>
<thead>
<tr>
<th></th>
<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>0000</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td>X</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>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>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0111 dominates 0101 remove 0111

Others?

1010 dominates 1000 remove 1010

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
Cyclic Core

- Cannot select (e.g. essential) or exclude (e.g. dominated) a PI definitively.
- Make a guess
  - A in cover
  - A not in cover
- Proceed from there

Example

A in Cover: What now?

```plaintext
A  B  C  D  E  F  G  H
0000   X                               X
0001   X  X
0011   X  X
0100   X  X
0101                     X  X
0110   X  X
0111   X  X
```

Example

A not in Cover

```plaintext
B  C  D  E  F  G  H
0000                                     X
0001        X
0011        X  X
0100                           X  X
0101                      X  X
0110                  X  X
0111             X  X
```

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

A in cover

A not in cover

\{A,B\}, \{A,B\}

A and B not in cover

\{A,B,C\}, \{/A,B,C\}
Branching Search w/ Implications

Branching Search w/ Implications

A in cover

A not in cover

\{A,B, C\}

\{A, B, C\}

Implications Prune Tree

(like BCP in SAT)

Only exponential in decision where must branch

Covering Technique?

- Possibly useful for dataflow subgraph selection? (Day 15)
  - Treat application components as rows (minterms)
  - Treat patterns as columns (PIs)
- But, more general (complicated) cost model

Optimization

- 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)

Heuristic Variant

- 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

• Assign5b due today
• Reading for Wednesday online (web)
• Withdraw date is Friday
  – Probably won’t get anything further graded
• Assign 6 & 7 out today
• No office hours on Tuesday
  – Makeup office hours Wed. 5pm
  – …or send email, make appointment