Today

- Two-Level Logic Optimization
  - Problem
  - Definitions
  - Basic Algorithm: Quine-McClusky
  - 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 Array Logic (PLAs)

- 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

Programmable Wired-or

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

Diagram Wired-or

Wired-or array

- 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 AND 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

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}(\text{product-term})
  • e.g. nand2=4, nand3=5,nand4=6...

Terminology (1)

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

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

| a | b | c | y   | Specify on-set only
|---|---|---|-----|------------------
| 0 | 0 | 0 | 0   | a | b | c | y
| 0 | 0 | 1 | 0   | 1 | 0 | 1 | 1
| 0 | 1 | 0 | 0   | 1 | 1 | 0 | 1
| 0 | 1 | 1 | 0   | 1 | 1 | 1 | 1
| 1 | 0 | 0 | 1   | 1 | 1 | 1 | 1
| 1 | 0 | 1 | 1   | 1 | 1 | 1 | 1
| 1 | 1 | 0 | 1   | 1 | 1 | 1 | 1
| 1 | 1 | 1 | 1   | 1 | 1 | 1 | 1
Cube/Logic Specification

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

<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>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>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

| 10 | 0 | 1 | 1 | 1 |
| 10 | 1 | 0 | 1 | 1 |
| 11 | 0 | 1 | 1 | 1 |
| 11 | 1 | 1 | 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 b y x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Multiple Outputs

- Consider:
  - \ X=/a/b+ab+ac 
  - \ Y=/bc 
- Trivial solution has 4 product terms
Multiple Outputs

- Consider:
  - X = a/b + ab + ac
  - Y = bc
- Now read off cover:
  - Y = /bc
  - A = /a/b/c + /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
  - less products if cover more
  - less 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

Restate Goal

- Goal in terms of PIs
  - Find minimum size set of PIs which 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
  - 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

0 0000
5 0101
7 0111
8 1000
9 1001
10 1010
11 1011
14 1110
15 1111

0, 8  -000
5, 7  01-1
7, 15 -111
8, 10 -100
8, 10, 10 10-0
10, 11 -10-0
10, 11, 14, 15 10-1
14, 15 -111

Covering Matrix

• Minterms \( \times \) Prime Implicants

<table>
<thead>
<tr>
<th>/b/c/d</th>
<th>/abd</th>
<th>bcd</th>
<th>a/b</th>
<th>ac</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Goal: minimum cover

Essential Reduction

• Must pick essential PI
  – pick and eliminate row and column

<table>
<thead>
<tr>
<th>/b/c/d</th>
<th>/abd</th>
<th>bcd</th>
<th>a/b</th>
<th>ac</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• This case:
  – Cover determined by essentials

• General case:
  – Reduces size of problem
  – These are easy…
D dominators: Column

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

<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>0111</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>X</td>
<td>X</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>
</tr>
<tr>
<td>1110</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C dominates B
G dominates H

New Essentials

• Dominance reduction may yield new Essential PIs

<table>
<thead>
<tr>
<th>B</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>1000</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C, G now essential

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>0000</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>X</td>
<td>X</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>
</tr>
<tr>
<td>0111</td>
<td>X</td>
<td>X</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>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>

1000 dominates 1010
remove 1010
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  B  C  D  E   F  G   H
0000   X                               X
0001   X  X
0101        X  X
0111             X  X
1000                                 X  X
1010                            X  X
1110                       X  X
1111                  X  X
```

Example

```
A in Cover:

A  B  C  D  E   F  G   H
0000   X                               X
0001   X  X
0101        X  X
0111             X  X
1000                                 X  X
1010                            X  X
1110                       X  X
1111                  X  X
```

```
B  C  D  E  F  G  H
0101    X  X
0111         X  X
1000                           X  X
1010                      X  X
1110                  X  X
1111             X  X
```

C dominates B
G dominates H

Example

```
A not in Cover:

B  C  D  E   F  G   H
0000                                     X
0001        X
0101        X  X
0111             X  X
1000                                 X  X
1010                            X  X
1110                       X  X
1111                  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

For N primes, how large?
Branching Search w/ Implications

- A in cover
- A not in cover
- \{A, B, C\} \rightarrow \{A, B, /C\}

Implications Prune Tree

Only exponential in decision where must branch

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

- 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

Admin

Reading for Monday online
Big Ideas

• Canonical Form
  – eliminate bias of input specification

• Technique:
  – branch-and-bound
  – dominators
  – use structure of problem to derive
    reduction between branching selection