

EDA (CS286.5b)

Day 15
Logic Synthesis:
Two-Level

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 PAL, PLA, ...
- Minimum number of gates for two-level implementation
- Starting point for multi-level optimization

Complexity

- Set covering problem
 - NP-hard

Cost

- PLA/PAL - first order
 - number of product terms
- Abstract (mis)
 - number of literals
 - $\text{cost}(y = a*b + a*/c) = 4$
- General (simple, multi-level)
 - $\text{Scost}(\text{product-term})$
 - e.g. $\text{nand2} = 4, \text{nand3} = 5, \text{nand4} = 6 \dots$

Terminology (1)

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

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	a b c y
0 0 0	0	1 0 1 1
0 0 1	0	1 1 0 1
0 1 0	0	1 1 1 1
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 abc 111
 - 1 - not inverted a/bc 101
 - - - not present ac 1-1

a b c	y			
1 0 1	1	1 0 1	1	
1 1 0	1	1 1 0	1	1 - 1 1 1 - 1
1 1 1	1	1 1 1	1	1 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

Truth Table:		Convert to single-output problem	On-set for result
a	b	y	x
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	1

a	b	y	x	o	
0	0	1	-	1	001-
0	0	-	1	-	00-1
0	1	-	1	1	010-
0	1	0	-	1	01-0
1	0	-	0	1	100-
1	0	0	-	1	10-0
1	0	-	0	1	110-
1	1	0	-	1	11-1
1	1	-	1	1	11-1

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

Prime Implicants

- Implicant -- cube in on-set
 - (not entirely in don't-case 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 be primes
 - $y = ab + ac + b/c$
 - $y = ac + b/c$
- Number of PI's can be exponential in input size
 - more than minterms, even!

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

```

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

Compute Prime Example

```

0 0000      0, 8 -000
5 0101      5, 7 01-1
7 0111      7,15 -111
8 1000      8, 9 100-
9 1001      8,10 10-0
10 1010     9,11 10-1
11 1011     10,11 101-
14 1110     10,14 1-10
15 1111     11,15 1-11
              14,15 111-
    
```

Compute Prime Example

```

0 0000      0, 8 -000      8, 9,10,11 10--
5 0101      5, 7 01-1     10,11,14,15 1-1-
7 0111      7,15 -111
8 1000      8, 9 100- *
9 1001      8,10 10-0 *
10 1010     9,11 10-1 *
11 1011     10,11 101- *
14 1110     10,14 1-10 *
15 1111     11,15 1-11 *
              14,15 111- *
              /b/c/d
              /abd
              bcd
              a/b
              ac
    
```

Covering Matrix

- Minterms x Prime Implicants

	/b/c/d	/abd	bcd	a/b	ac
0000	X				
0101		X			
0111		X	X		
1000	X			X	
1001				X	
1010				X	X
1011				X	X
1110					X
1111			X		X

Essential Reduction

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

	/b/c/d	/abd	bcd	a/b	ac
0000	X				
0101		X			
0111		X	X		
1000	X			X	
1001				X	
1010				X	X
1011				X	X
1110					X
1111			X		X

Dominators: Column

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

	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				

Dominators: Column

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

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

C dominates B
G dominates H

Dominators: Column

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

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

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

New Essentials

- Dominance reduction may yield new Essential PIs

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

C,G now essential

	C	D	E	F	G
1110		X	X		
1111	X	X			

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 dominates
 - we can remove the **dominating** row

(NOTE OPPOSITE OF COLUMN CASE)

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

0111 dominates 0101
remove 0111

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

	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				

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	A in Cover:
0000	X						X	
0001	X	X						
0101	X	X						0101 X X
0111		X	X					0111 X X
1000					X	X		1000 X X
1010				X	X			1010 X X
1110			X	X				1110 X X
1111		X	X					1111 X X

Example

A	B	C	D	E	F	G	H	A not in Cover:
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
0000						X
0001	X					
0101	X	X				
0111		X	X			
1000				X	X	X
1010				X	X	
1110			X	X		
1111		X	X			

Basic Two-Level Minimization

- Generate Prime Implicants
- Reduce (essential, dominators)
- If not done,
 - pick a cube
 - branch (back to reduce) on selected/not
- Save smallest

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

Today's Big Ideas

- Canonical Form
 - eliminate bias of input specification
- Technique:
 - branch-and-bound
 - dominators
 - use structure of problem to derive reduction between branching selection