

ESE535: Electronic Design Automation

Day 4: February 25, 2009
Two-Level Logic-Synthesis



Penn ESE535 Spring 2009-- DeHon

Today

- Two-Level Logic Optimization
 - Problem
 - Definitions
 - Basic Algorithm: Quine-McClusky
 - Improvements

2

Penn ESE535 Spring 2009-- DeHon

Problem

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

3

Penn ESE535 Spring 2009-- DeHon

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

4

Penn ESE535 Spring 2009-- DeHon

Programmable Array Logic (PLAs)

5

Penn ESE535 Spring 2009-- DeHon

PLA

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

6

Penn ESE535 Spring 2009-- DeHon

Wired-or

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

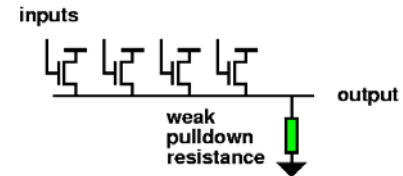


Penn ESE535 Spring 2009-- DeHon

7

Wired-or

- Obvious Implementation with Transistors

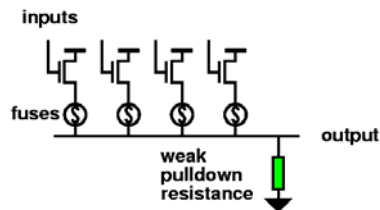


Penn ESE535 Spring 2009-- DeHon

8

Programmable Wired-or

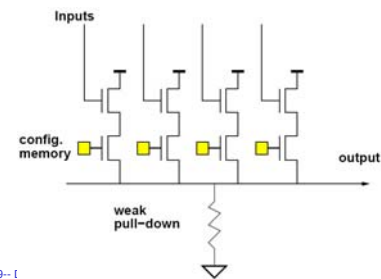
- Use some memory function to programmable connect (disconnect) wires to OR
- Fuse:



Penn ESE535 Spring 2009-- DeHon

Programmable Wired-or

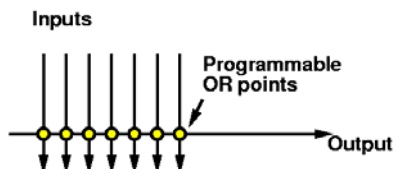
- Memory configurable PLA model



Penn ESE535 Spring 2009-- I

10

Diagram Wired-or

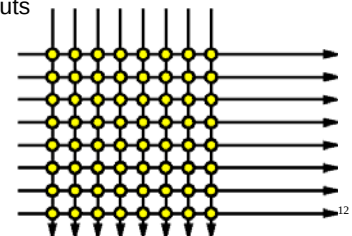


Penn ESE535 Spring 2009-- DeHon

11

Wired-or array

- Build into array
 - Compute many different or functions from set of inputs



Penn ESE535 Spring 2009-- DeHon

Combined **or**-arrays to PLA

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

The diagram illustrates a combined or-and array structure. It consists of two 8x8 grids of yellow circular nodes. The left grid has vertical lines extending upwards and downwards from each node, and horizontal lines extending to the right from each node. The right grid has vertical lines extending upwards and downwards from each node, and horizontal lines extending to the left from each node. The two grids are connected by a central vertical line. At the bottom of the diagram, there are two rows of green circular nodes, each with a vertical line extending downwards from it. These green nodes are connected to the horizontal lines of the right grid.

13

Penn ESE535 Spring 201

- Penn ESE535 Spring 200

13

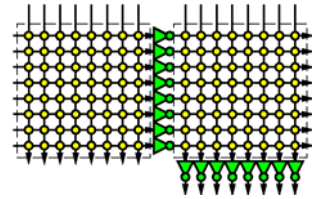
PLA

- 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**.

The diagram illustrates a PLA structure. It features two vertical columns of logic gates. The left column contains AND gates, and the right column contains OR gates. Each gate is represented by a circle with a dot (for AND) or a cross (for OR). The gates are connected to a common set of inputs at the top, which are represented by a horizontal line with vertical connections to each gate. The outputs of the gates are shown at the bottom, with lines connecting to a common output bus.

- Penn ESE535 Spring 2009-- DeHon

[illegible]

Penn ESE535 Spring 2009-- DeHon

15

PLA and PAL

The image contains two schematic diagrams of programmable logic devices. The left diagram represents a Programmable Array Logic (PLA) device. It features a fixed AND plane (represented by a grid of blue AND gates) and a programmable OR plane (represented by a grid of blue OR gates). The inputs are labeled i3, i2, i1, and i0. The outputs are labeled o3, o2, o1, and o0. The right diagram represents a Programmable Array Logic (PAL) device. It features a programmable AND plane (represented by a grid of blue AND gates) and a fixed OR plane (represented by a grid of blue OR gates). The inputs are labeled i3, i2, i1, and i0. The outputs are labeled o3, o2, o1, and o0.

PLA = Programmable Array Logic

PAL has **fixed** AND plane.

Penn ESE535 Spring 2009-- DeHon

Penn ESE535 Spring 2009-- DeHon

...back to optimization...

Penn ESE535 Spring 2009-- DeHon

17

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

Penn ESE535 Spring 2009-- DeHon 18

- Penn ESE535 Spring 2009-- DeHon

18

Complexity

- Set covering problem
 - NP-hard

Penn ESE535 Spring 2009-- DeHon

19

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. $\text{nand}2=4, \text{nand}3=5, \text{nand}4=6\dots$

Penn ESE535 Spring 2009-- DeHon

20

Terminology (1)

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

Penn ESE535 Spring 2009-- DeHon

21

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

Penn ESE535 Spring 2009-- DeHon

22

Terminology (3)

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

Penn ESE535 Spring 2009-- DeHon

23

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					

Penn ESE535 Spring 2009-- DeHon

24

Cube/Logic Specification

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

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 -

Penn ESE535 Spring 2009-- DeHon

25

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

Penn ESE535 Spring 2009-- DeHon

26

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 1	00-1
	0 1 0 - 1	010 -
	0 1 - 0 1	01-0
	1 0 0 - 1	100 -
	1 0 - 0 1	10-0
	1 1 0 - 1	110 -
	1 1 - 1 1	11-1

Penn ESE535 Spring 2009-- DeHon

27

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

Penn ESE535 Spring 2009-- DeHon

28

Multiple Outputs

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

Penn ESE535 Spring 2009-- DeHon

29

Multiple Outputs

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

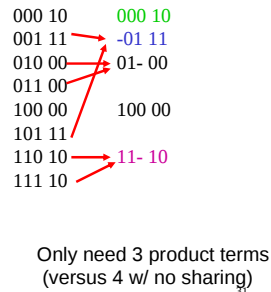
000 10	000 10
001 11	001 11
010 00	010 00
011 00	011 00
100 00	100 00
101 11	101 11
110 10	110 10
111 10	111 10

Penn ESE535 Spring 2009-- DeHon

30

Multiple Outputs

- Consider:
 - $X = a/b + ab + ac$
 - $Y = bc$
- Now read off cover:
 - $Y = bc$
 - $A = a/b/c + bc + ab$
 - $= a/b + bc + ab$



Penn ESE535 Spring 2009-- DeHon

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)

Penn ESE535 Spring 2009-- DeHon

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

Penn ESE535 Spring 2009-- DeHon

Restate Goal

- Goal in terms of PIs
 - Find minimum size set of PIs which cover the on-set.

Penn ESE535 Spring 2009-- DeHon

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)

Penn ESE535 Spring 2009-- DeHon

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

Penn ESE535 Spring 2009-- DeHon

Compute Prime Example

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

Penn ESE535 Spring 2009-- DeHon

37

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, 9 100-
8,10 10-0
9 1001
9,11 10-1
10,11 101-
10,14 1-10
11,15 1-11
14,15 111-

Penn ESE535 Spring 2009-- DeHon

38

Compute Prime Example

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

8, 9,10,11 10--
10,11,14,15 1-1-

Penn ESE535 Spring 2009-- DeHon

39

Covering Matrix

- Minterms $\times$ Prime Implicants

Goal:
minimum
cover

	/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	X

Penn ESE535 Spring 2009-- DeHon

40

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

Penn ESE535 Spring 2009-- DeHon

41

Essential Reduction

- This case:
 - Cover determined by essentials
- General case:
 - Reduces size of problem
 - These are easy...

Penn ESE535 Spring 2009-- DeHon

42

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			

C dominates B
G dominates H

Penn ESE535 Spring 2009-- DeHon

43

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			

C D E F G

Penn ESE535 Spring 2009-- DeHon

44

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
1110 X X
1111 X X
E dominates D and F
Cover = {C,E,G}

Penn ESE535 Spring 2009-- DeHon

45

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

Penn ESE535 Spring 2009-- DeHon

46

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?

Penn ESE535 Spring 2009-- DeHon

47

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			

Penn ESE535 Spring 2009-- DeHon

48

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

Penn ESE535 Spring 2009-- DeHon

49

Example

	A	B	C	D	E	F	G	H	
0000	X							X	A in Cover:
0001	X	X							B C D E F G H
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

Penn ESE535 Spring 2009-- DeHon

50

Example

	A	B	C	D	E	F	G	H	
0000	X							X	A in Cover:
0001	X	X							B C D E F G H
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

C dominates B
G dominates H

Penn ESE535 Spring 2009-- DeHon

51

Example

	A	B	C	D	E	F	G	H	
0000	X							X	A not in Cover:
0001	X	X							B C D E F G H
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

Penn ESE535 Spring 2009-- DeHon

52

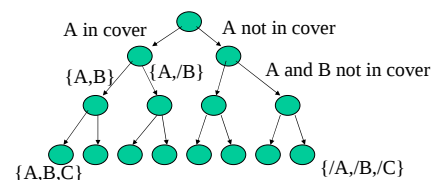
Basic Two-Level Minimization (espresso-exact)

- 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

Penn ESE535 Spring 2009-- DeHon

53

Branching Search

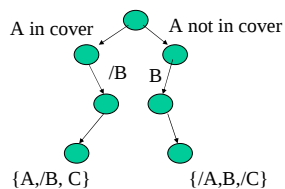


For N primes, how large?

Penn ESE535 Spring 2009-- DeHon

54

Branching Search w/ Implications



Implications Prune Tree

Only exponential in decision
where must branch

Penn ESE535 Spring 2009-- DeHon

55

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)

Penn ESE535 Spring 2009-- DeHon

56

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

Penn ESE535 Spring 2009-- DeHon

57

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

Penn ESE535 Spring 2009-- DeHon

58

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

Penn ESE535 Spring 2009-- DeHon

59

Admin

Reading for Monday online

Penn ESE535 Spring 2009-- DeHon

60

Big Ideas

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