

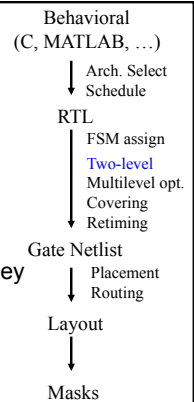
ESE535: Electronic Design Automation

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

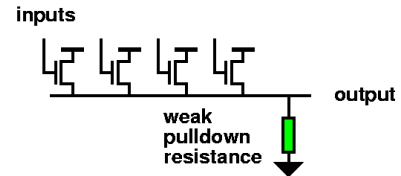


Penn ESE535 Spring 2015 -- DeHon

7

Wired-or

- Obvious Implementation with Transistors

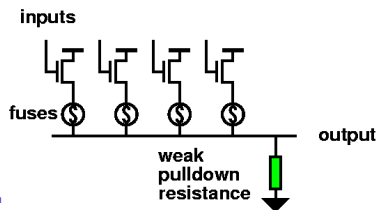


Penn ESE535 Spring 2015 -- DeHon

8

Programmable Wired-or

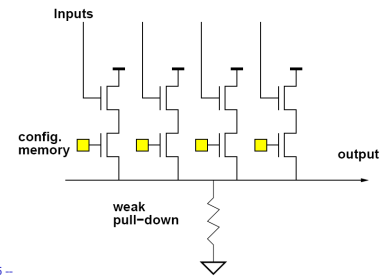
- Use some memory function to programmable connect (disconnect) wires to OR
- Fuse:



Penn ESE535 Spring 2015 -- DeHon

Programmable Wired-or

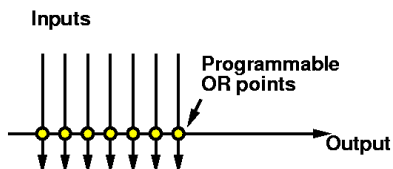
- Memory configurable PLA model



Penn ESE535 Spring 2015 --

10

Diagram Wired-or

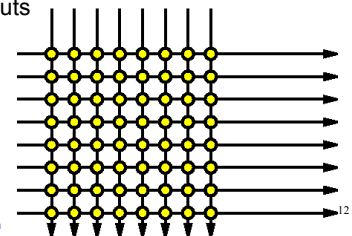


Penn ESE535 Spring 2015 -- DeHon

11

Wired-or array

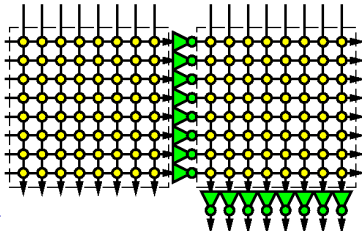
- Build into array
 - Compute many different or functions from set of inputs



Penn ESE535 Spring 2015 -- DeHon

Combined or-arrays to PLA

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



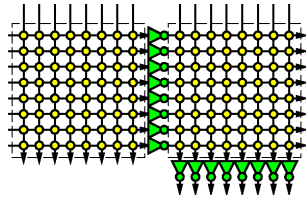
Penn ESE535 Spring 20

13

PLA

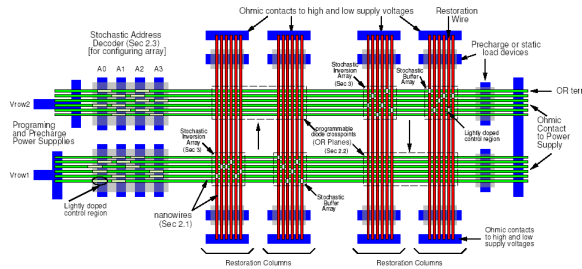
- Can implement each **and** on single line in first array
- Can implement each **or** on single line in second array

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



Penn ESE535 Spring 2015 -- DeHon

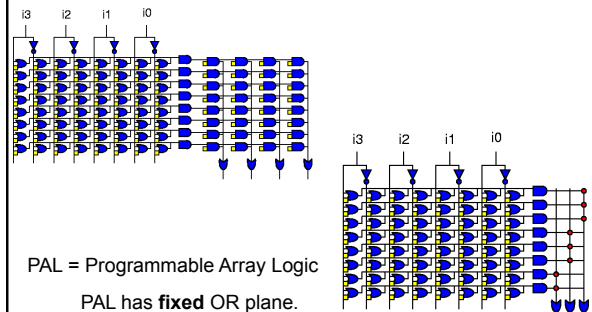
Nanowire PLA



Penn ESE535 Spring 2015 -- DeHon

15

PLA and PAL



PAL = Programmable Array Logic
 PAL has **fixed** OR plane.

Penn ESE535 Spring 2015 -- DeHon

...back to optimization...

Penn ESE535 Spring 2015 -- 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 2015 -- DeHon

18

Complexity

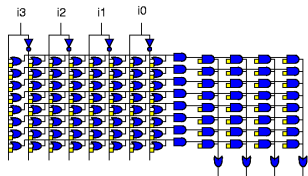
- Set covering problem
 - NP-hard

Terminology (1)

- Literals -- $a, /a, b, /b, \dots$
 - 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

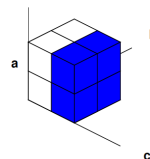
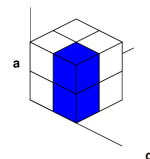
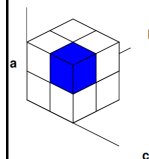
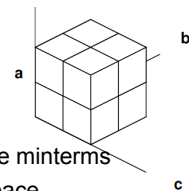
- PLA/PAL – to first order costs is:
 - number of product terms
- Abstract (mis, sis)
 - {multilevel, sequential} interactive synthesis
 - number of literals
 - $\text{cost}(y=a*b+a*/c)=4$
- General (simple, multi-level)
 - $\sum \text{cost}(\text{product-term})$
 - e.g. nand2=4, nand3=5, nand4=6...

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 (2)

- Cube:
 - product covering one or more minterms
 - Think of as n-dimensional space
 - Looking for a $k \leq n$ cube



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				
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

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

Multiple Outputs

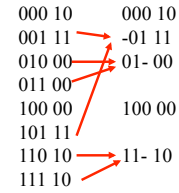
- Can reduce to single output case
 - write equations on inputs and each output
 - with onset for relation being true
 - after cover
 - remove literals associated with outputs

Multiple Outputs

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

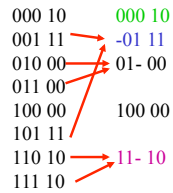
Multiple Outputs

- 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:
 - $Y = bc$
 - $A = a/b/c + bc + ab$
 - $= a/b + bc + ab$



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

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)

Penn ESE535 Spring 2015 -- DeHon

37

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 2015 -- DeHon

38

Compute Prime Example

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

Note this is preclass 3.

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

Penn ESE535 Spring 2015 -- DeHon

39

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-

Penn ESE535 Spring 2015 -- DeHon

40

Compute Prime Example

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

Penn ESE535 Spring 2015 -- DeHon

41

Covering Matrix

- Minterms × Prime Implicants

Goal:
minimum
cover

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

Penn ESE535 Spring 2015 -- DeHon

42

Essential Reduction

- Must pick essential PI Which essential?
– 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 2015 -- DeHon

43

Essential Reduction

- This case:
 - Cover determined by essentials
 - Preclass 3: $ac+a/b+/abd+/b/c/d$
- General case:
 - Reduces size of problem

Penn ESE535 Spring 2015 -- DeHon

44

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

Any others?

Penn ESE535 Spring 2015 -- DeHon

45

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			

Penn ESE535 Spring 2015 -- DeHon

46

New Essentials

- Dominance reduction may yield new Essential PIs What's essential?

	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		

What now?
E dominates D and F

Cover = {C,E,G} 47

Penn ESE535 Spring 2015 -- DeHon

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

Others?

1010 dominates 1000
remove 1010

Penn ESE535 Spring 2015 -- DeHon

48

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
0000	X							X
0001	X	X						
0101		X	X					
0111		X	X					
1000						X	X	
1010						X	X	
1110				X	X			
1111		X	X					

A in Cover: What now?

	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				

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					

A in Cover:

	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	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					

What now?

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					

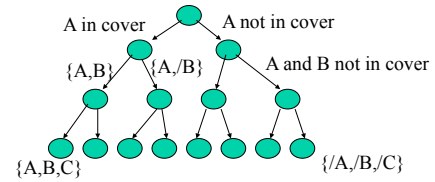
Basic Two-Level Minimization (espresso-exact)

- 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

Penn ESE535 Spring 2015 -- DeHon

55

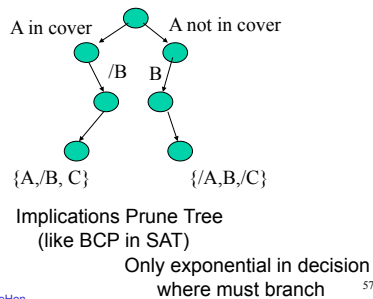
Branching Search



Penn ESE535 Spring 2015 -- DeHon

56

Branching Search w/ Implications



Penn ESE535 Spring 2015 -- DeHon

57

Covering Technique?

- 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

Penn ESE535 Spring 2015 -- DeHon

58

Optimization

- Summarize Minterms (signature cubes)
 - rows represent collection of minterms with same primes
- Avoid generating full set of PI's
 - 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 2015 -- DeHon

59

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 PI's
- Iterative improvement

Penn ESE535 Spring 2015 -- DeHon

60

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)