

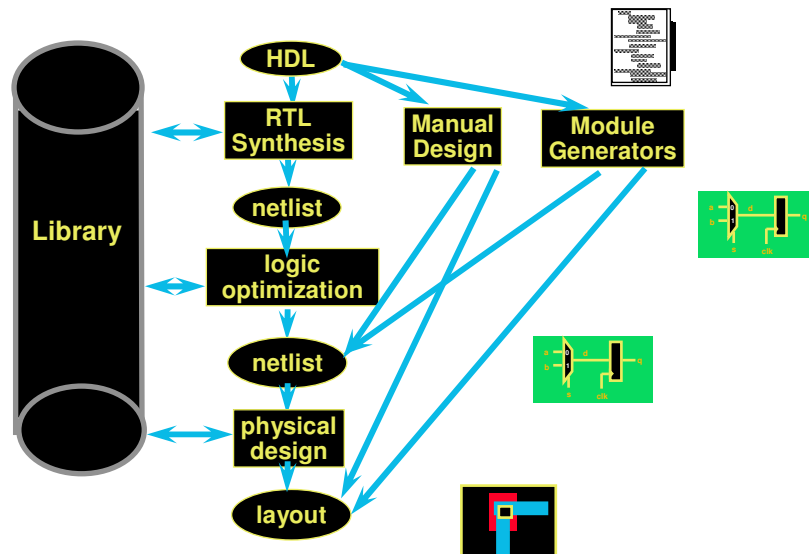
Technology Dependent Logic Optimization

Prof. Kurt Keutzer
EECS
University of California
Berkeley, CA

Thanks to S. Devadas

1

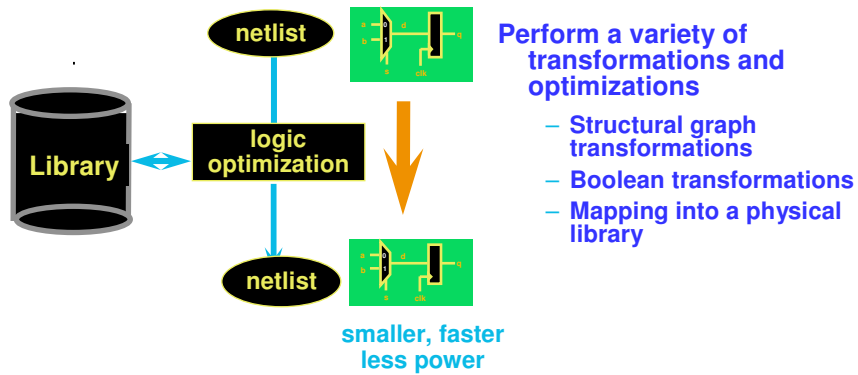
RTL Design Flow



Kurt Keutzer

2

Logic Optimization



Kurt Keutzer

3

Combinational Logic Optimization

Input:

- Initial Boolean network
- Timing characterization for the module
 - - input arrival times and drive factors
 - - output loading factors
- Optimization goals
 - - output required times
- Target library description

Output:

- Minimum-area net-list of library gates which meets timing constraints

A very difficult optimization problem !

Kurt Keutzer

4

Modern Approach to Logic Optimization

Divide logic optimization into two subproblems:

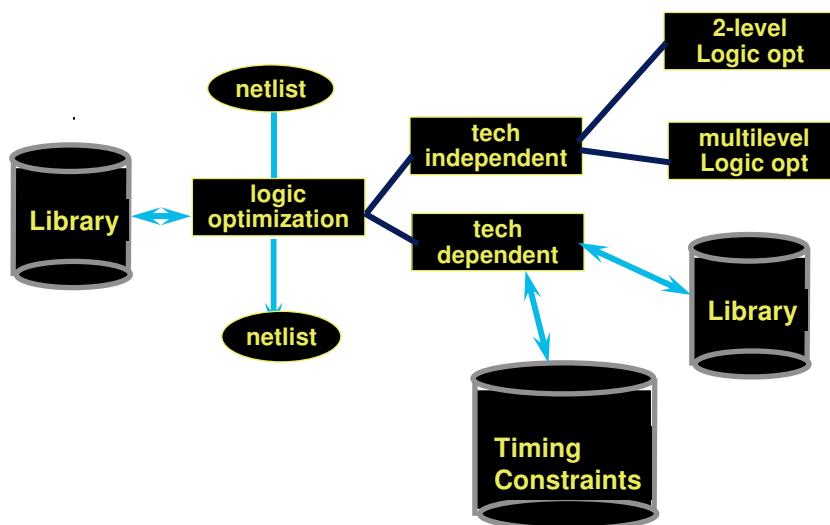
- Technology-independent optimization
 - - determine overall logic structure
 - - estimate costs (mostly) independent of technology
 - - simplified cost modeling
- Technology-dependent **optimization** (technology mapping)
 - - binding onto the gates in the library
 - - detailed technology-specific cost model

Orchestration of various optimization/transformation techniques for each subproblem

Kurt Keutzer

5

Logic Optimization



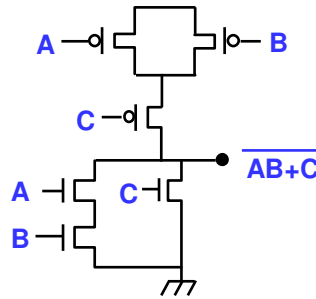
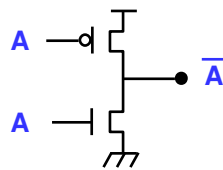
Kurt Keutzer

6

“Closed Book” Technology Library

A standard cell technology or library may contain many hundreds of cells

Typical cells are NAND, NOR, NOT, AOI (AND-or-Invert), OAI (Or-And-Invert) etc.



Kurt Keutzer

7

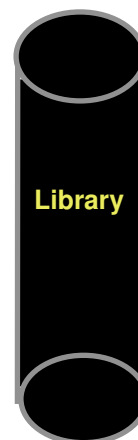
Library

Contains for each cell:

- Functional information: cell = $a * b * c$
- Timing information: function of
 - input slew
 - intrinsic delay
 - output capacitance
- non-linear models used in tabular approach
- Physical footprint (area)
- Power characteristics

Wire-load models - function of

- Block size
- Wiring



Kurt Keutzer

8

Elements of a library - 1

Element/Area Cost

INVERTER 2 

NAND2 3 

NAND3 4 

NAND4 5 

Kurt Keutzer

9

Elements of a library - 2

Element/Area Cost

AOI21 4 

AOI22 5 

Kurt Keutzer

10

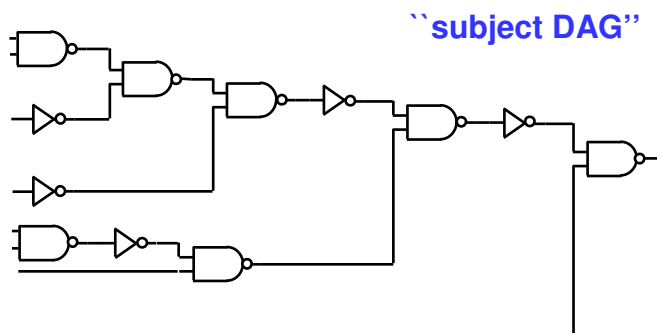
Reasonable Library

Inverter, Buffer
ND2-ND4; NOR2-NOR4; AND2- AND4;
AOI21 - AOI333; OAI21 - OAI333
XOR, XNOR
MUX, Full Adder
Neg-Edge Triggered D-Flip-Flop
Pos-Edge Triggered D-FF
J-K FF
Above with various clears, enables
Scan versions of each of the above
Most of the above in 6 different power sizes:
– 1x, 2x, 4x, 6x, 8x, 16x

Kurt Keutzer

11

Input Circuit Netlist

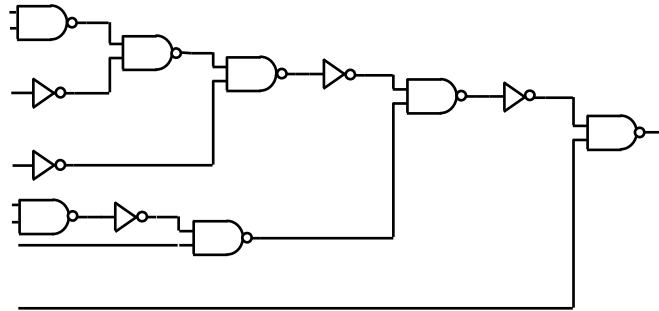


Kurt Keutzer

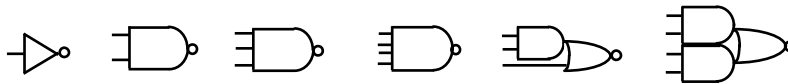
12

Problem statement

Find an "optimal" (in area, delay, power) mapping of a circuit



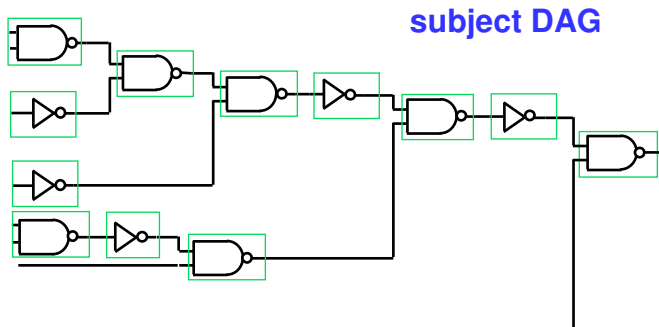
into the technology library (simple example below):



Kurt Keutzer

13

Is there a problem? Trivial Covering #1



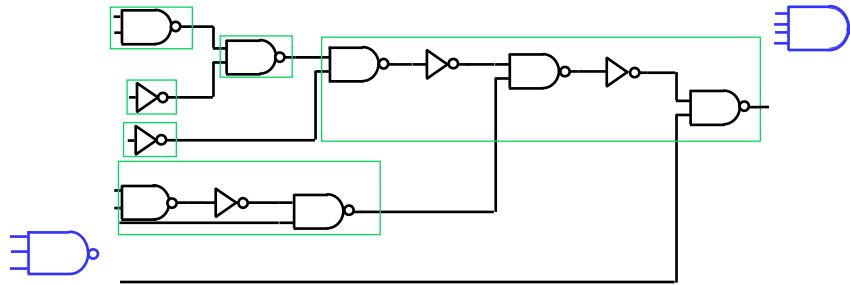
subject DAG

7	NAND2 (3) = 21
5	INV (2) = 10
	Area cost 31

Kurt Keutzer

14

Covering #2

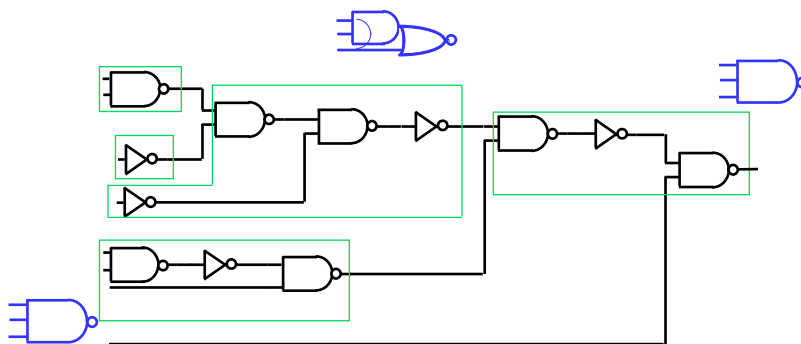


2 INV	= 4
2 NAND2	= 6
1 NAND3	= 4
1 NAND4	= 5
<hr/>	
Area cost 19	

Kurt Keutzer

15

Covering #3



Costs:
31, 19, 17
Yes, there's a problem!

1 INV	= 2
1 NAND2	= 3
2 NAND3	= 8
1 AOI21	= 4
<hr/>	
Area Cost 17	

Kurt Keutzer

16

History of the Problem - 1

Technology mapping in 1986 was a big problem

- **Almost every design group (e.g. AT&T) had their own library**
 - ASIC – 400 cells
 - Microprocessor/DSP – 200 base cells
 - Government – 200+ cells
- **Every group had their own approach to mapping**
 - “Do what you have to do!” – handcrafted mappers tied to particular libraries and optimization tools
 - “Rule-based” systems – e.g. GE Socrates – very slow
 - “expert systems” that made no guarantee on final quality of result

Kurt Keutzer

17

History of the Problem - 2

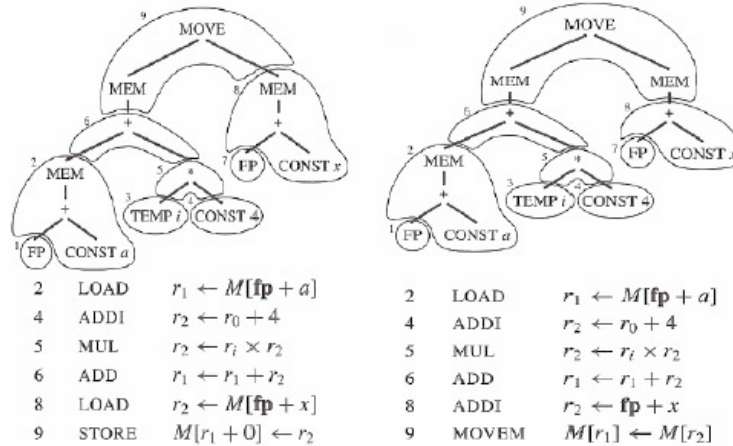
Yes, there are two problems:

- Technology mapping can significantly affect the area, speed, and power dissipation of a circuit
- There are over 200 different semiconductors each with multiple internal libraries – how to create a tool that can utilize a diverse set of libraries??

Kurt Keutzer

18

A similar problem – code generation



Example of code generation in compilers using tree-covering

- Handles complex instruction sets → Handles complex libraries
- Easily portable to other instruction sets → Easily portable to

Kurt Keutzer

19

Problem Formulation: DAG Covering

Represent input netlist in normal form

⇒ subject DAG

Represent each library gate with normal forms for the logic function

⇒ primitive DAGs

Each primitive DAG has a cost

Goal: Find a minimum cost covering of the subject DAG by the primitive DAGs

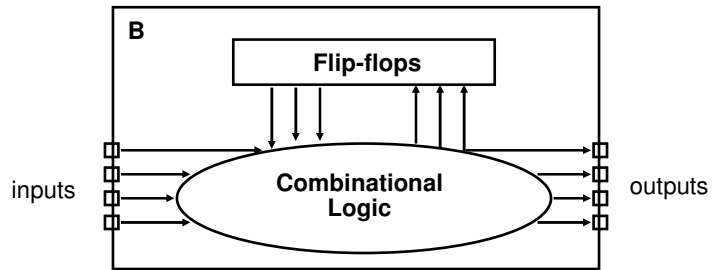
Normal form: 2-input NAND gates and inverters

K. Keutzer, *DAGON: Technology Binding and Local Optimization by DAG Matching*, in Proceedings of the 24th Design Automation Conference, 1987 and 25 Years of Design Automation

Kurt Keutzer

20

Step 1: Extract Combinational Logic

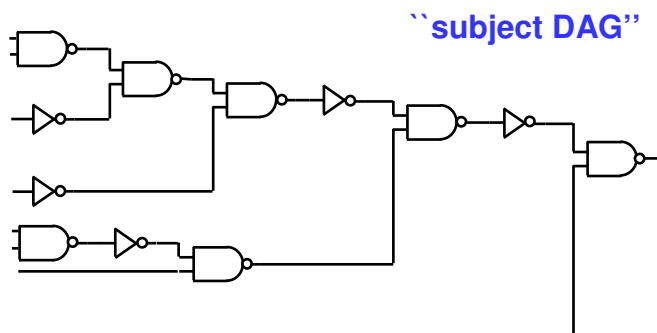


Since FF's don't need to be optimized with surrounding combinational logic we can partition them out

Kurt Keutzer

21

Step 2: Normalize Circuit Netlist






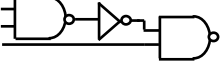

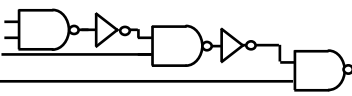
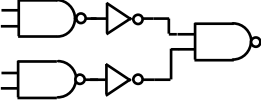


Reduce the netlist into ND2 gates

Kurt Keutzer

22


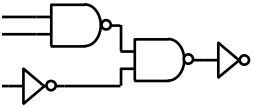
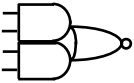
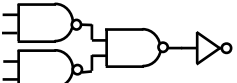
Step 3a: Normalize library

	Element/Area Cost	Tree Representation (normal form)
INVERTER	2 	
NAND2	3 	
NAND3	4 	
NAND4	5 	 

Kurt Keutzer

23

Step 3b: Normalize library

	Element/Area Cost	Tree Representation (normal form)
AOI21	4 	
AOI22	5 	

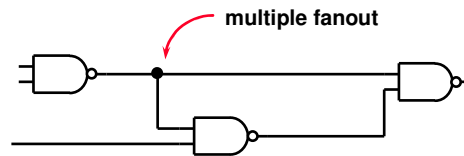
Kurt Keutzer

24

Step 4: DAG Covering

Sound Algorithmic approach
NP-hard optimization problem

K. Keutzer, D. Richards, *Computation Complexity of Logic Synthesis and Optimization*, in Proceedings of the International Workshop on Logic Synthesis, 1989

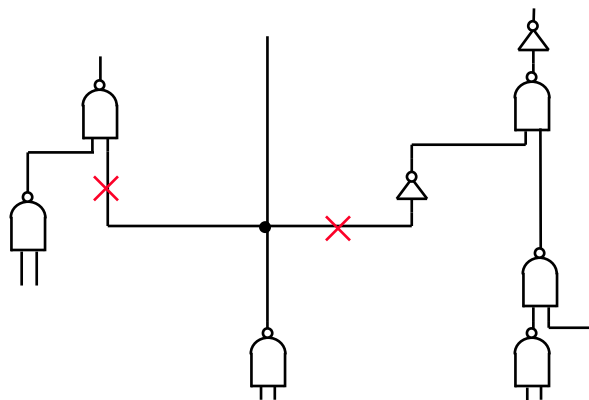


Tree covering heuristic: If subject and primitive DAGs are trees, efficient algorithm can find optimum cover \Rightarrow dynamic programming formulation

Kurt Keutzer

25

Solution formulation

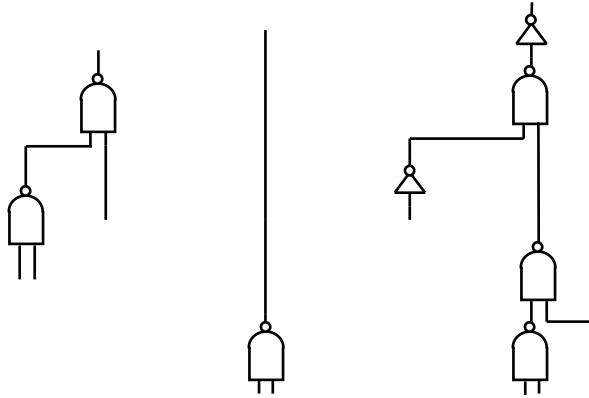


Kurt Keutzer

26

Resulting Trees

Break at multiple fanout points

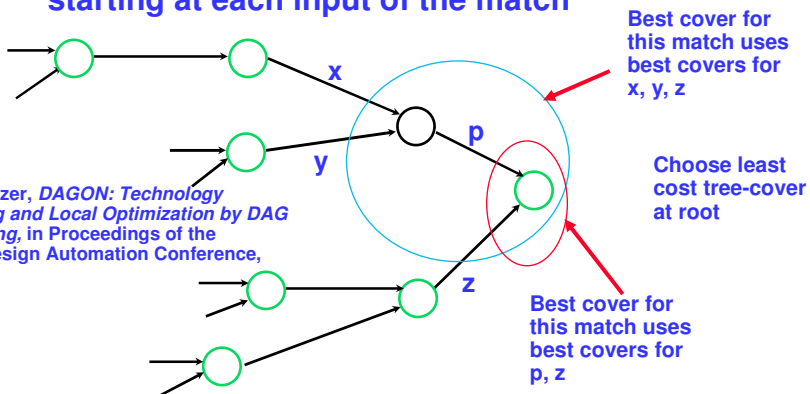


Kurt Keutzer

27

For each tree - Dynamic Programming

Principle of optimality: Optimal cover for a tree consists of a best match at the root of the tree plus the optimal cover for the sub-trees starting at each input of the match

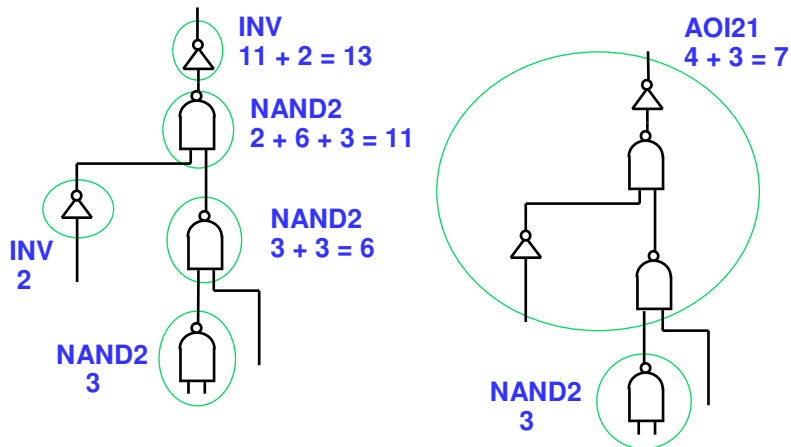


K. Keutzer, *DAGON: Technology Binding and Local Optimization by DAG Matching*, in Proceedings of the 24th Design Automation Conference, 1987

Kurt Keutzer

28

Example of Optimal Tree Covering



Kurt Keutzer

29

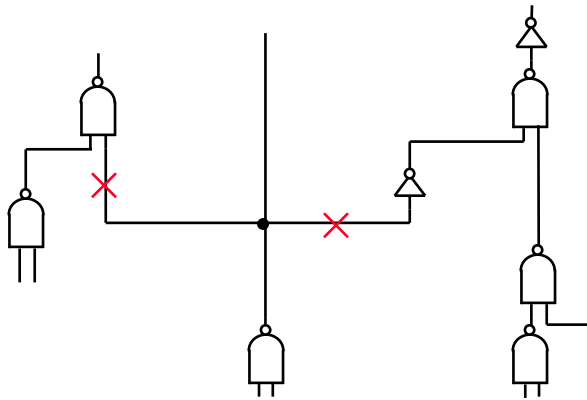
DAG covering in detail

- 1) partition DAG into a forest of trees
- 2) normalize netlist
- 3) optimally cover each tree
 - a) generate all candidate matches
 - b) find the optimal match using dynamic programming

Kurt Keutzer

30

Partition DAG into Forest of trees



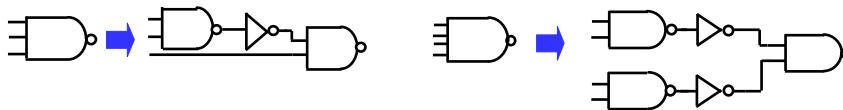
Each gate with fanout > 1 becomes root of a new tree

Kurt Keutzer

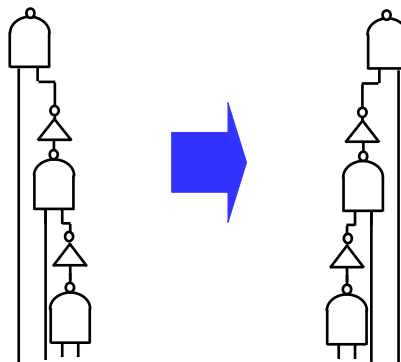
31

Normalize netlist

Re-express netlist into 2-input Nand gates and Inverters



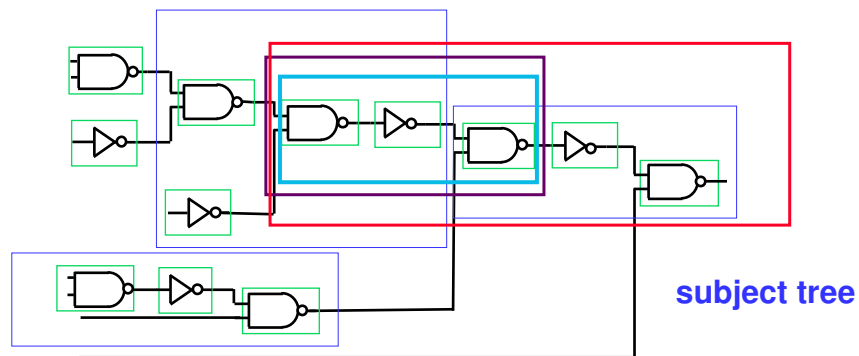
Make each tree left-oriented



Kurt Keutzer

32

Generate candidate matches - 1



At the end of this segment each gate in the subject tree is annotated with every possible library cell that could be rooted at that gate

What are some ways we can generate matches?

Kurt Keutzer

33

Generating candidate matches -2

Naïve approach -

try to match each cell in the library with each node of the tree (libraries can be large! - beware of large constants!!)

Better approach

build tables such that only potential candidate matches are checked

Best approach

fancy string matching - pp. 862-869

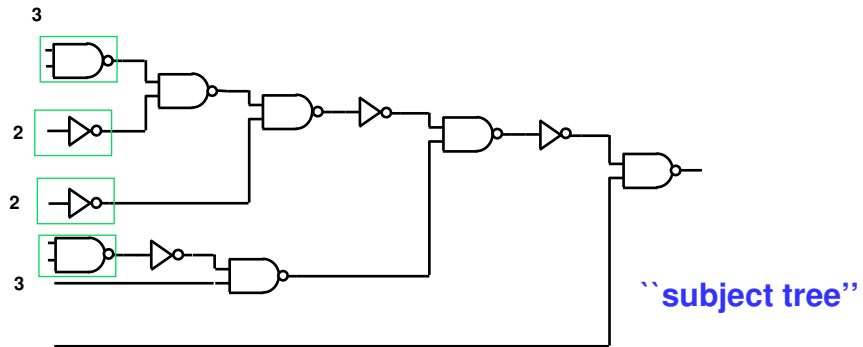
Introduction to Algorithms, T. Cormen, C. Lesierson, R. Rivest, The MIT Press, Second Printing, 1996. - pp. 862-869

What's the complexity of each approach?

Kurt Keutzer

34

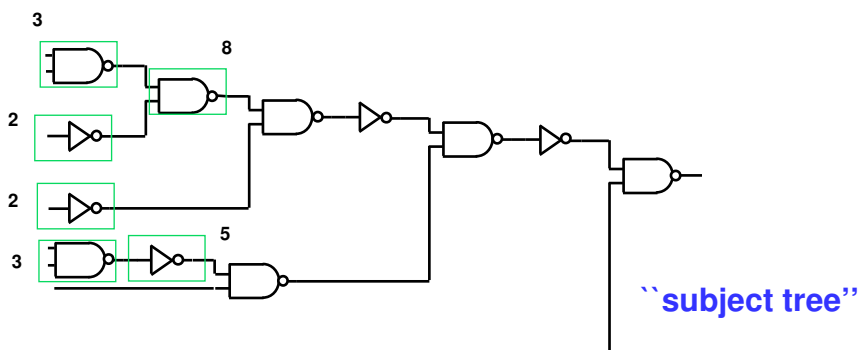
Optimal tree covering - 1



Kurt Keutzer

35

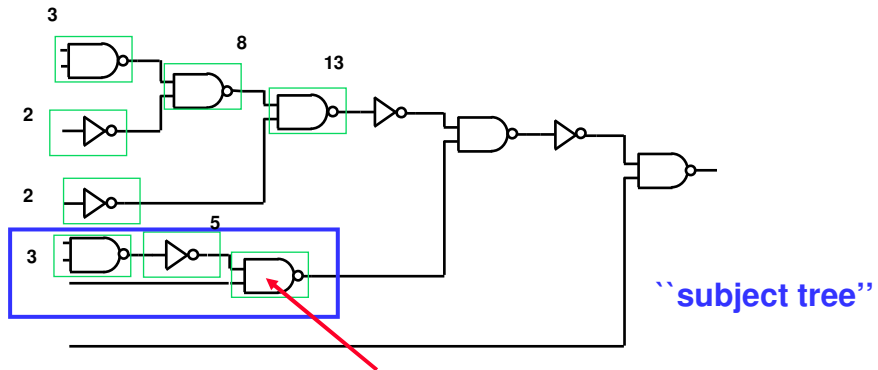
Optimal tree covering - 2



Kurt Keutzer

36

Optimal tree covering - 3



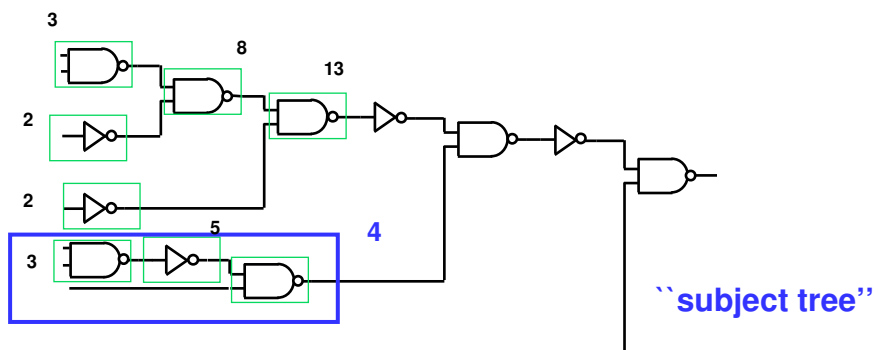
Cover with ND2 or ND3 ?

1 NAND2	3	1 NAND3	= 4
+ subtree	5		
Area cost 8			

Kurt Keutzer

37

Optimal tree covering – 3b

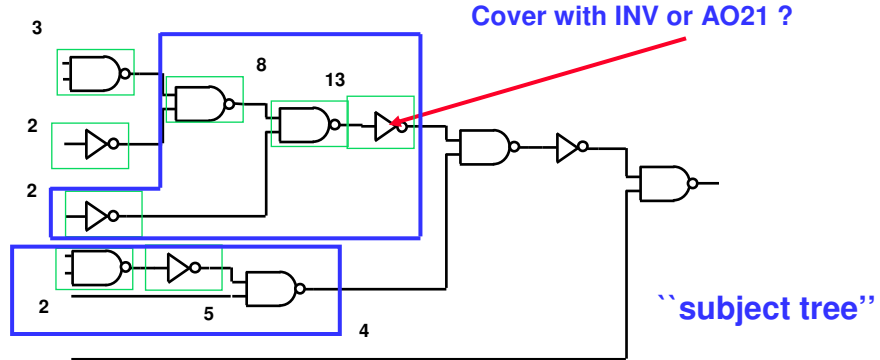


Label the root of the sub-tree with optimal match and cost

Kurt Keutzer

38

Optimal tree covering – 4a



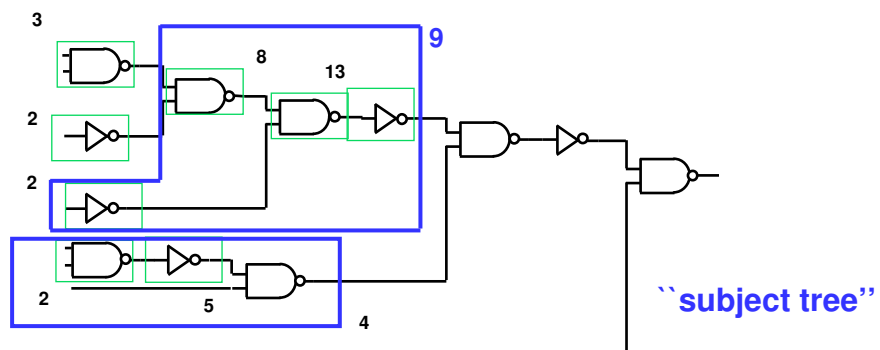
1 Inverter
+ subtree 2
 13
Area cost 15

1 AO21 4
+ subtree 1 3
+ subtree 2 2
Area cost 9

Kurt Keutzer

39

Optimal tree covering – 4b

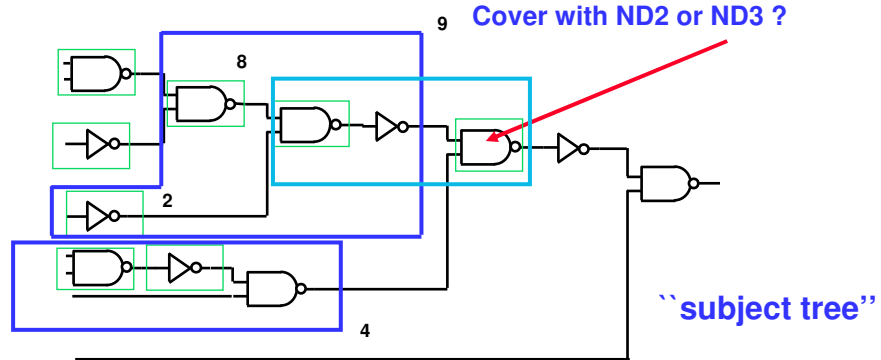


Label the root of the sub-tree with optimal match and cost

Kurt Keutzer

40

Optimal tree covering - 5

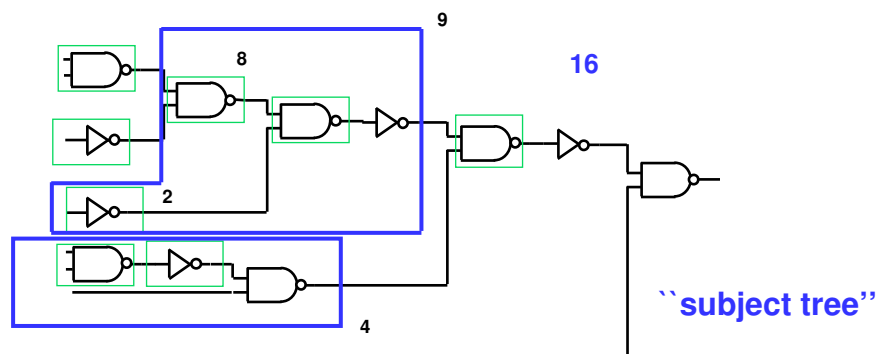


NAND2	subtree 1	9	subtree 1	8	NAND3
	subtree 2	4	subtree 2	2	
	1 NAND2	<u>3</u>	subtree 3	4	
	Area cost 16		1 NAND3	<u>4</u>	
			Area cost 18		

Kurt Keutzer

41

Optimal tree covering - 5b

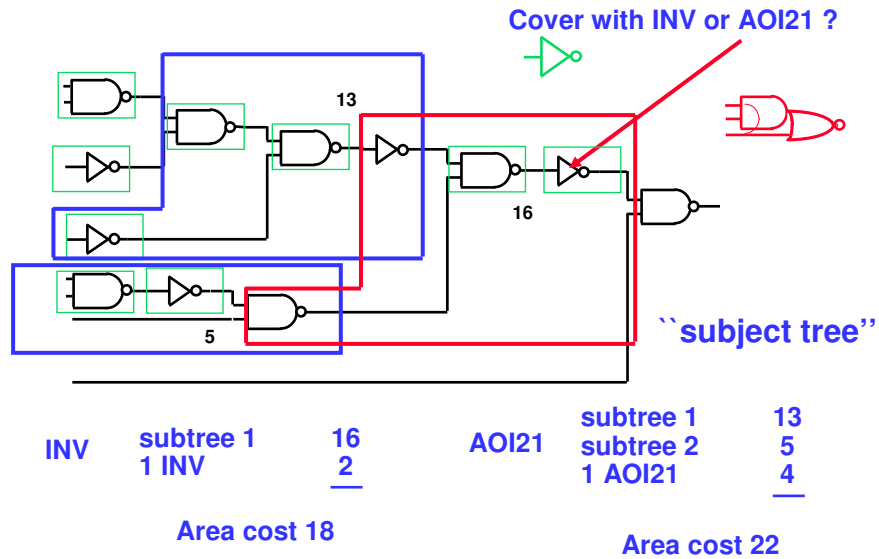


Label the root of the sub-tree with optimal match and cost

Kurt Keutzer

42

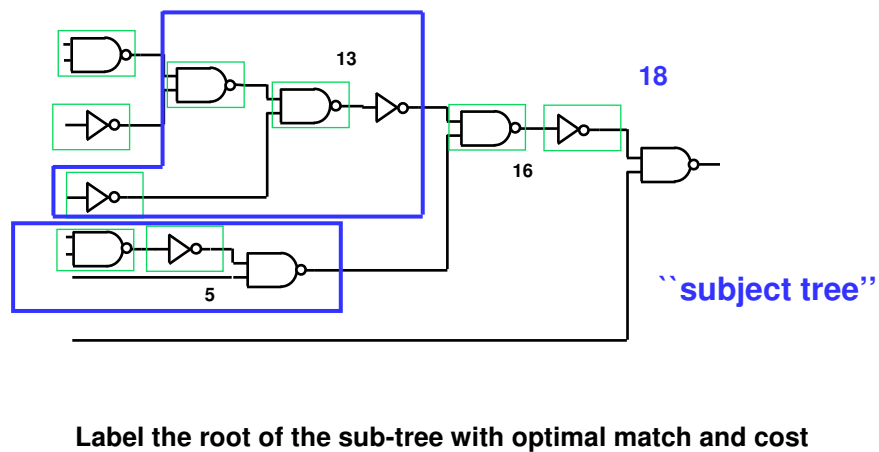
Optimal tree covering - 6



Kurt Keutzer

43

Optimal tree covering – 6b

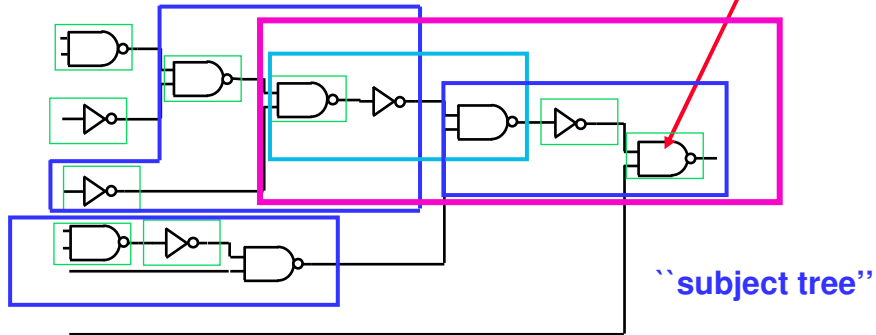


Kurt Keutzer

44

Optimal tree covering - 7

Cover with ND2 or ND3 or ND4 ?

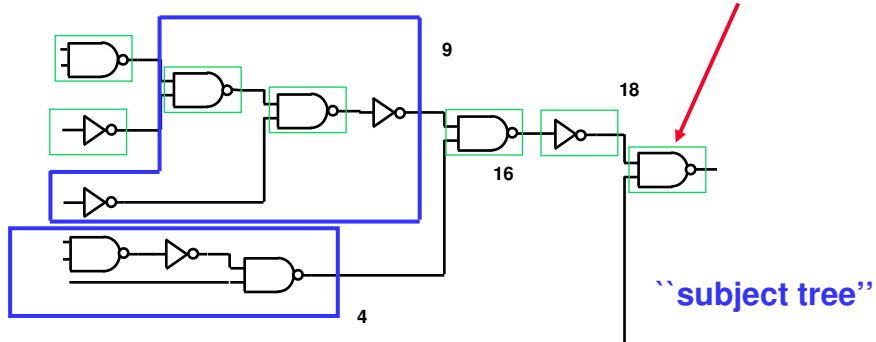


Kurt Keutzer

45

Cover 1 - NAND2

Cover with ND2 ?



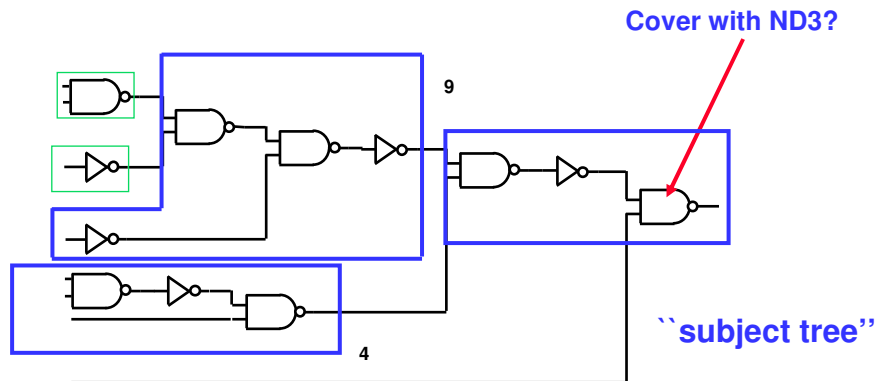
subtree 1	18
subtree 2	0
1 NAND2	<u>3</u>

Area cost 21

Kurt Keutzer

46

Cover 2 - NAND3



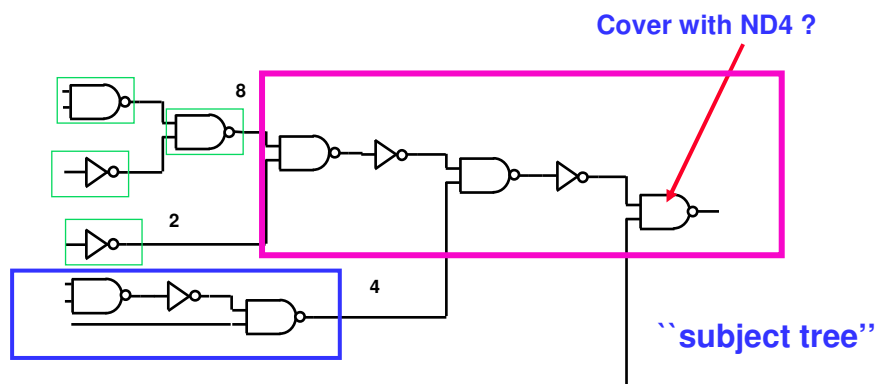
subtree 1	9
subtree 2	4
subtree 3	0
1 NAND3	<u>4</u>

Area cost 17

Kurt Keutzer

47

Cover - 3



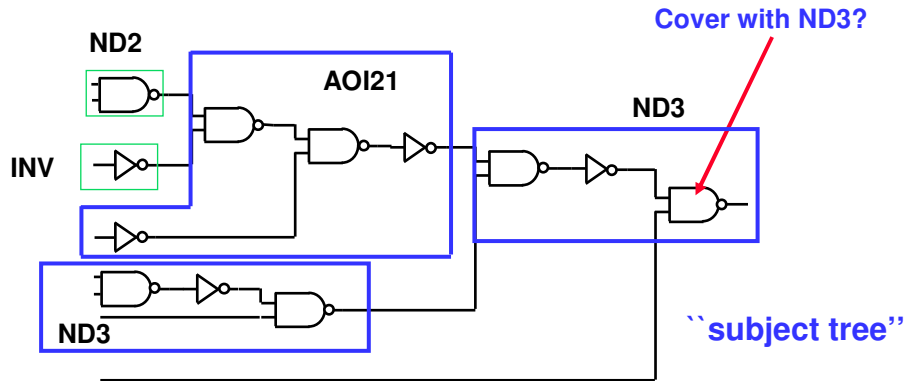
subtree 1	8
subtree 2	2
subtree 3	4
subtree 4	0
1 NAND4	<u>5</u>

Area cost 19

Kurt Keutzer

48

Optimal Cover was Cover 2



Clear that greedy doesn't work well
What's the complexity?

INV	2
ND2	3
2 ND3	8
AOI21	4
	<hr/>
	Area cost 17

Kurt Keutzer

49

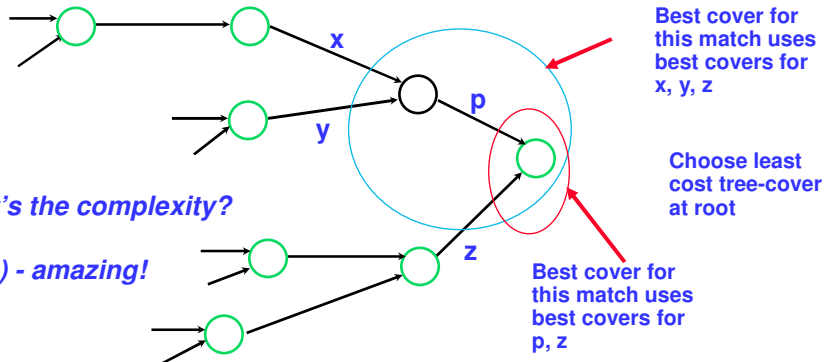
Computational Complexity

To determine the optimal cover for a tree we only need to consider a best cost match at the root of the tree

This is constant-time in the number of matched cells

Plus the optimal cover for the sub-trees starting at each input of the match

This is constant-time in the indegree/fan-in of each match



What's the complexity?

$O(n)$ - amazing!

Kurt Keutzer

50

Enhancements to DAG covering

Many enhancements incorporated over the last decade

- Timing optimization incorporating load-dependent delays
 - Rudell - UCB
- Optimization for low power
- Application to FPGAs –
 - J. Rose - Chortle
 - J. Cong - Flowmap
- Optimal direct DAG covering without tree covering approximation (didn't net much)

Kurt Keutzer

51

Summary of Technology Mapping

DAG covering formulation

- Separated library issues from mapping algorithm

Heuristics based on tree covering for area and delay

- surprisingly efficient final result - for technology/library dependent reasons

Very efficient

- linear time

Very flexible approach

- applicable to wide range of libraries (standard cell, gate array) and technologies (FPGAS)

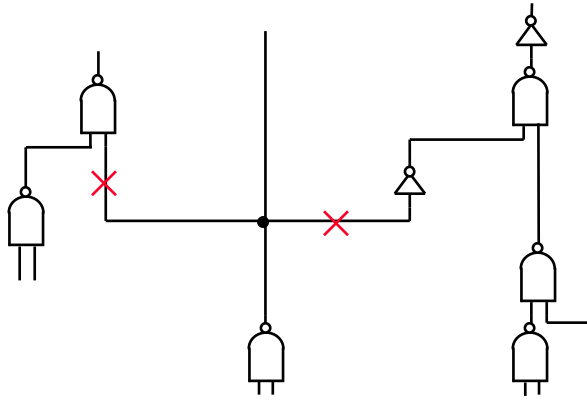
Best enhancement is integration of technology decomposition

Also requires "follow up" rule based approaches for best final circuit efficiency

Kurt Keutzer

52

Why does this approximation work well?

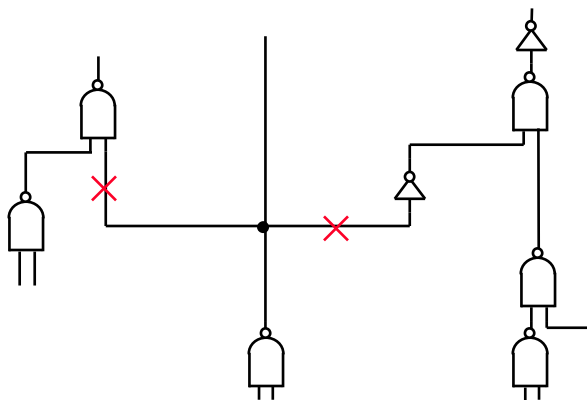


Each gate with fanout >1 becomes root of a new tree

Kurt Keutzer

53

Why does this approximation work well?

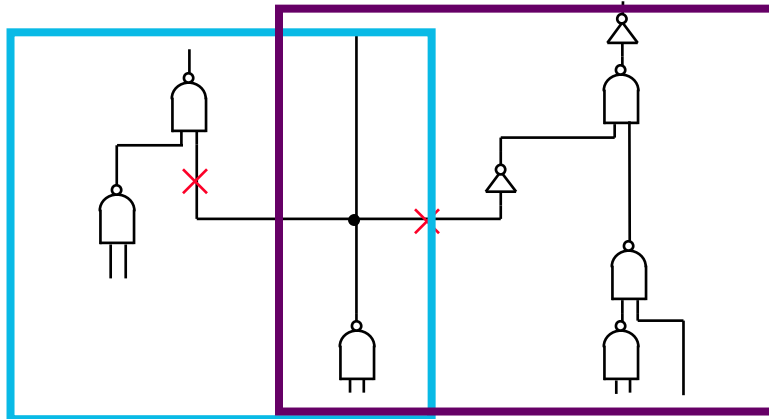


Few non-tree cells – XOR, MUX – one-level deep

Kurt Keutzer

54

Why does this approximation work well?



Non-tree matching usually requires duplication – rarely a benefit for area

Kurt Keutzer

55

Kurt Keutzer

56

Retrospective

DAG covering by tree-covering is effective for four reasons

- separates library definition and characterization from mapping algorithm
- Duplication of logic not a win in terms of area optimization. Advantage of duplication of logic for timing is very (physical) context dependent
- provided an efficient mapping in what appears to be a relatively flat solution space
- Very computationally efficient so suitable to VLSI scale (millions of gates) netlist

Principal weaknesses

- Problems handling multiplexor-trees, full-adders, other DAG patterns
- Problems in performing performance optimization tricks in tight pipelined logic

Kurt Keutzer

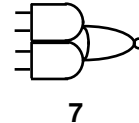
57

Extra Slides

Kurt Keutzer

58

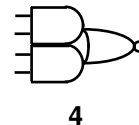
Typical library costs



Kurt Keutzer

59

But what if?



Kurt Keutzer

60

Strong (or Boolean) Division

Given a function f to be strong divided by g

- Add an extra input to f corresponding to g , namely G and obtain function h as follows

$$h_{DC} = G\bar{g} + \bar{G}g$$

$$h_{ON} = f_{ON} - h_{DC}$$

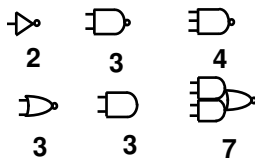
$$h_{OFF} = \overline{f_{ON} + h_{DC}}$$

Minimize h using two-level minimizer

Kurt Keutzer

61

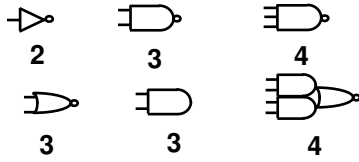
Typical library costs



Kurt Keutzer

62

But what if?



Kurt Keutzer

63