

ESE535: Electronic Design Automation

Day 7: February 11, 2008
Static Timing Analysis
and Multi-Level Speedup



Penn ESE535 Spring 2008 -- DeHon

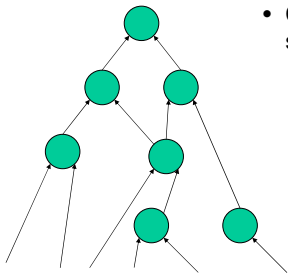
Today

- Topological Worst Case
 - not adequate (too conservative)
- Sensitization Conditions
- Timed Calculus
- Delay-justified paths
 - Timed-PODEM
- Speedup

Penn ESE535 Spring 2008 -- DeHon

2

Topological Worst-Case Delay

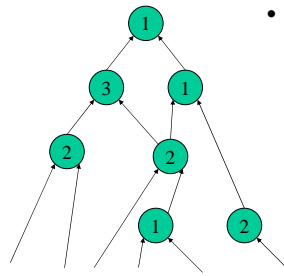


- Compute ASAP schedule
 - Take max of arrival times
 - Apply node Delay

Penn ESE535 Spring 2008 -- DeHon

3

Topological Worst-Case Delay

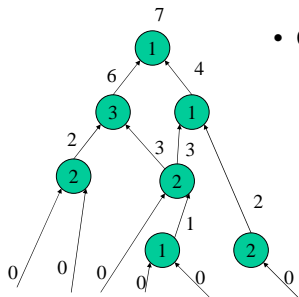


- Node Delays

Penn ESE535 Spring 2008 -- DeHon

4

Topological Worst-Case Delay



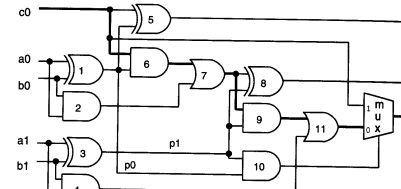
- Compute Delays

Penn ESE535 Spring 2008 -- DeHon

5

Conservative

- Topological Worst-Case Delay can be conservative

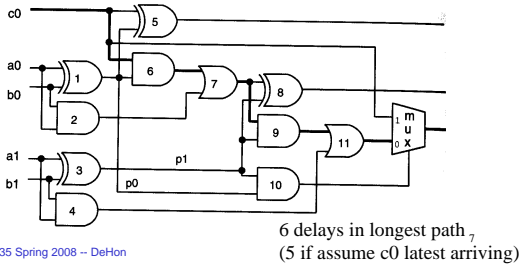


[Fig/Examples from Logic Synthesis
by Devadas, Gosh, Keutzer 1994]

Penn ESE535 Spring 2008 -- DeHon

Example

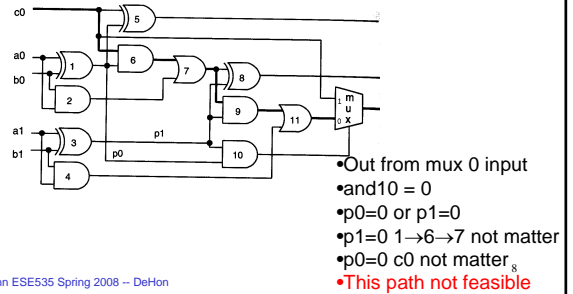
- Assume each gate 1:



Penn ESE535 Spring 2008 -- DeHon

Example

- Is this path possible?



Penn ESE535 Spring 2008 -- DeHon

False Paths

- Once consider logic for nodes
 - There are logical constraints on data values
- There are paths which cannot logically occur
 - Call them **false paths**

Penn ESE535 Spring 2008 -- DeHon

9

What can we do?

- Need to assess what paths are real
- Brute force
 - for every pair of inputs
 - compute delay in outputs from in1→in2 input transition
 - take worst case
- Expensive:
 - 2^{2n} delay traces

Penn ESE535 Spring 2008 -- DeHon

10

Alternately

- Look at single vector and determine what controls delay of circuit
 - I.e. look at values on path and determine path *sensitized* to change with input

Penn ESE535 Spring 2008 -- DeHon

11

Controlled Inputs

- Controlled input to a gate:
 - input whose value will determine gate output
 - e.g.
 - 0 on a AND gate
 - 1 on a OR gate

Penn ESE535 Spring 2008 -- DeHon

12

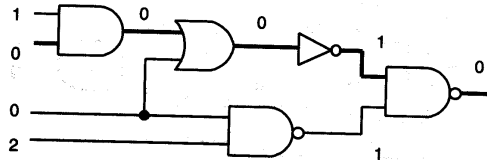
Static Sensitization

- A path is statically sensitized
 - if all the side (non-path) inputs are non-controlling
 - I.e. this path value flips with the input

Penn ESE535 Spring 2008 -- DeHon

13

Statically Sensitized Path

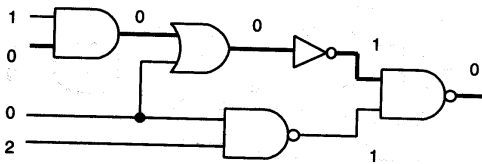


Penn ESE535 Spring 2008 -- DeHon

14

Sufficiency

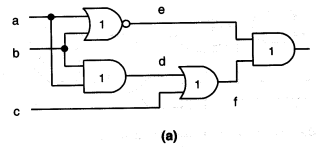
- Static Sensitization is **sufficient** for a path to be a **true** path in circuit



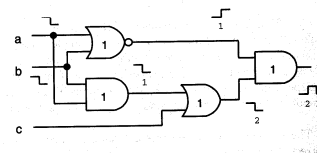
Penn ESE535 Spring 2008 -- DeHon

15

...but not necessary



Paths of length 3 not statically sensitizable.



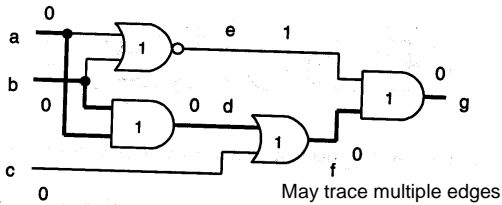
But there is a true path of delay 3.

Penn ESE535 Spring 2008 -- DeHon

16

Static Co-sensitization

- Each output with a controlled value
 - has a controlling value as input on path
 - (and vice-versa for non-controlled)

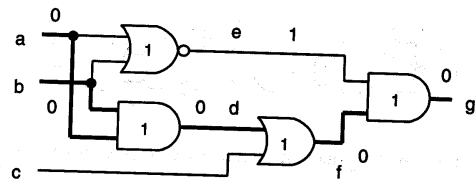


Penn ESE535 Spring 2008 -- DeHon

17

Necessary

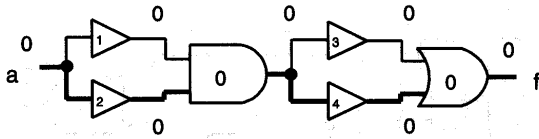
- Static Co-sensitization is a **necessary** condition for a path to be true



Penn ESE535 Spring 2008 -- DeHon

18

...but not sufficient

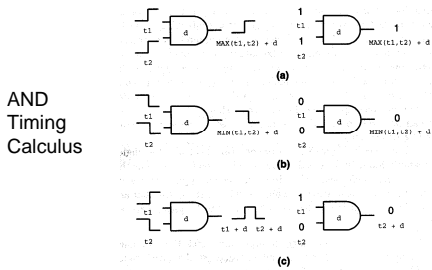


Cosensitize path of length 6.
Real delay is 5.

Combining

- Combine these ideas into a timed-calculus for computing delays for an input vector

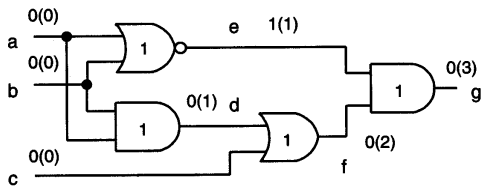
Computing Delays



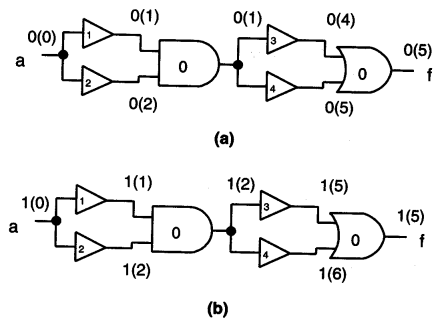
Rules

- If gate output is at a controlling value, pick the minimum input and add gate delay
- If gate output is at a non-controlling value, pick the maximum input and add gate delay

Example (1)



Example (2)



Now...

- We know how to get the delay of a single input condition
- Could:
 - find critical path
 - search for an input vector to sensitize
 - if fail, find next path
 - ...until find longest true path
- May be $O(2^n)$

Penn ESE535 Spring 2008 -- DeHon

25

Better Approach

- Ask if can justify a delay greater than T
- Search for satisfying vector
 - ...or demonstration that none exists
- Binary search to find tightest delay

Penn ESE535 Spring 2008 -- DeHon

26

Delay Computation

- Modification of a testing routine
 - used to justify an output value for a circuit
- PODEM
 - backtracking search to find a suitable input vector associated with some target output
 - Simply a branching search with implication pruning
 - Heuristic for smart variable ordering

Penn ESE535 Spring 2008 -- DeHon

27

Search1

- Takes in list of nodes to satisfy
- If all satisfied → done
- Backtrace to set next PI
- if inconsistent PI value
 - try inverting this PI call Search2
- else
 - search to set next PI
 - if fail
 - try inverting and Search2

Penn ESE535 Spring 2008 -- DeHon

28

Search2

- ;; same idea, but this one not flip bit
- ;; because already tried inverted value
- If no conflict
 - search to set next PI
- otherwise
 - pass back failure

Penn ESE535 Spring 2008 -- DeHon

29

Backtrace

- Follow back gates w/ unknown values
 - sometimes output dictate input must be
 - (AND needing 1 output; with one input already assigned 1)
 - sometimes have to guess what to follow
 - (OR with 1 output and no inputs set)
 - Uses heuristics to decide what to follow

Penn ESE535 Spring 2008 -- DeHon

30

Example

Try justify $g=1$

(a)

(b)

Penn ESE535 Spring 31

Example

(a)

(b)

Penn ESE535 Spring 2008 -- T 32

For Timed Justification

- Also want to compute delay
 - on incompletely specified values
- Compute bounds on timing
 - upper bound, lower bound
 - Again, use our timed calculus
 - expanded to unknowns

Penn ESE535 Spring 2008 -- DeHon 33

Delay Calculation

AND rules

$i_1 \rightarrow$ $i_2 \downarrow$	0	1	2
0	0 $MIN(l_1, l_2) + d$ $MIN(u_1, u_2) + d$	0 $l_2 + d$ $u_2 + d$	0 $MIN(l_1, l_2) + d$ $u_2 + d$
1	0 $l_1 + d$ $u_1 + d$	1 $MAX(l_1, l_2) + d$ $MAX(u_1, u_2) + d$	2 $l_1 + d$ $MAX(u_1, u_2) + d$
2	0 $MIN(l_1, l_2) + d$ $u_1 + d$	2 $l_2 + d$ $MAX(u_1, u_2) + d$	2 $MIN(l_1, l_2) + d$ $MAX(u_1, u_2) + d$

Penn ESE535 Spring 2008 -- DeHon 34

Timed PODEM

- **Input:** value to justify and delay T
- **Goal:** find input vector which produces value and exceeds delay T
- Algorithm
 - similar
 - implications check timing as well as logic

Penn ESE535 Spring 2008 -- DeHon 35

Example

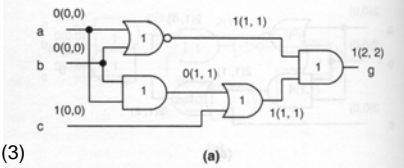
Justify 1(3)

(a)

(b)

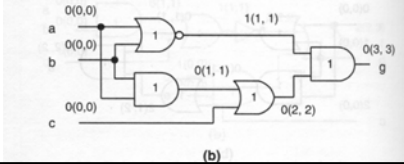
Penn ESE535 Spring 2008

Example



Fail to justify 1(3)

Justify 0(3)



Penn ESE535 Spring 2008 --

Search

- Less than 2^n
 - pruning due to implications
 - here saw a must be 0
 - no need to search 1xx subtree

Penn ESE535 Spring 2008 -- DeHon

38

Questions

- On static timing analysis?

Penn ESE535 Spring 2008 -- DeHon

39

Speed Up

(sketch flavor)

Penn ESE535 Spring 2008 -- DeHon

40

Speed Up

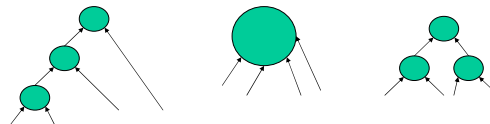
- Start with area optimized network
- Know target arrival times
 - Know delay from static analysis
- Want to reduce delay of node

Penn ESE535 Spring 2008 -- DeHon

41

Basic Idea

- Improve speed by:
 - Collapsing node(s)
 - Refactoring collapsed subgraph to reduce height



Penn ESE535 Spring 2008 -- DeHon

42

Speed Up

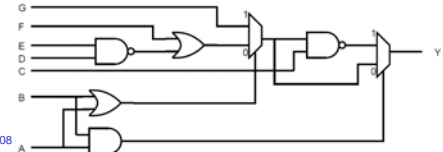
- While (delay decreasing, timing not met)
 - Compute delay (slack)
 - Static timing analysis
 - Generate network close to critical path
 - w/in some delay ϵ , to some distance d
 - Weight nodes in network
 - Less weight = more potential to improve, prefer to cut
 - Compute **mincut** of *nodes* on weighted network
 - For each node in cutset
 - Partial collapse
 - For each node in cutset
 - Timing redecompose

Penn ESE535 Spring 2008 -- DeHon

43

MinCut of Nodes

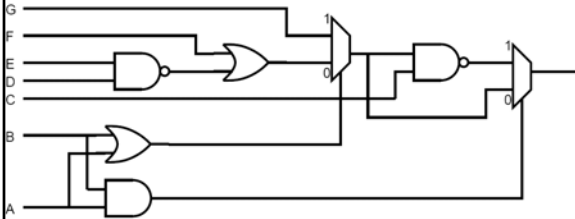
- Cut nodes not edges
 - Typically will need to transform to dual graph
 - All edges become nodes, nodes become edges
 - Then use maxflow/mincut



Penn ESE535 Spring 2008

MinCut of Nodes

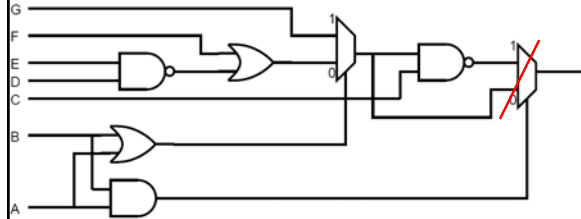
- What are possible cuts?



Penn ESE535 Spring 2008 -- DeHon

45

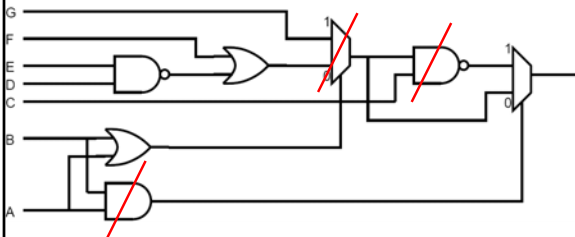
MinCut of Nodes



Penn ESE535 Spring 2008 -- DeHon

46

MinCut of Nodes



Penn ESE535 Spring 2008 -- DeHon

47

Weighted Cut

- $W = W_t + \alpha W_a \rightarrow \alpha$ tuning parameter
- Want to minimize area expansion
 - Things in collapsed network may be duplicated
 - E.g. W_a = literals in duplicated logic
- Want to maximize likely benefit
 - Prefer nodes with varying input times to the “near critical path” network
 - Quantify: large variance in arrival times
 - Prefer nodes with critical path on longer paths

Penn ESE535 Spring 2008 -- DeHon

48

Weighing Benefit

- Want to maximize likely benefit
 - Prefer nodes with varying input times to the "near critical path" network

Penn ESE535 Spring 2008 -- DeHon 49

Weighing Benefit

- Want to maximize likely benefit
 - Prefer nodes with critical path on longer paths

Penn ESE535 Spring 2008 -- DeHon 50

Timing Decomposition

- Extract area saving kernels that do not include critical inputs to node
 - $f = abcd + abce + abef$ (last time)
 - Kernels = $\{cd + ce + ef, e + d, c + f\}$
 - $F = abe(c + f) + abcd, ab(cd + ce + ef), abc(e + d) + abef$
 - What if:
 - Critical input is f? d? a? {a,d}?
- When decompose (e.g. into nand2's) similarly balance with critical inputs closest to output

Penn ESE535 Spring 2008 -- DeHon 51

Example

Penn ESE535 Spring 2008 -- DeHon 52

Example

Penn ESE535 Spring 2008 -- DeHon 53

Example

Penn ESE535 Spring 2008 -- DeHon 54

Speed Up (review)

- While (delay decreasing, timing not met)
 - Compute delay (slack)
 - Static timing analysis
 - Generate network close to critical path
 - w/in some delay ϵ , to some distance d
 - Weight nodes in network
 - Less weight = more potential to improve, prefer to cut
 - Compute **mincut** of *nodes* on weighted network
 - For each node in cutset
 - Partial collapse
 - For each node in cutset
 - timing redecompose

Penn ESE535 Spring 2008 -- DeHon

55

Admin

- Assignment 1 return
- Reading
 - Wed. retiming (handout today)
 - Mon. cover+retime (link on web)
- Assignment 2 due Monday
- Office hours Tuesday 4pm

Penn ESE535 Spring 2008 -- DeHon

56

Big Ideas

- Topological Worst-case delays are conservative
 - Once consider logical constraints
 - may have false paths
- Necessary and sufficient conditions on true paths
- Search for paths by delay
 - or demonstrate non existence
- Search with implications
- Iterative improvement

Penn ESE535 Spring 2008 -- DeHon

57