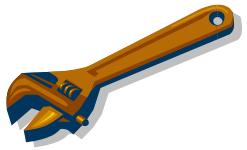


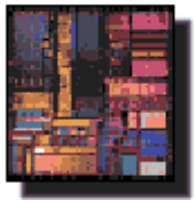




ECE 595Z
Digital VLSI Design Automation

Module 6 (Lectures 21-24): Timing Analysis and Optimization

Lecture 22



Anand Raghunathan

MSEE 318

raghunathan@purdue.edu

Lecture 21 - Recap

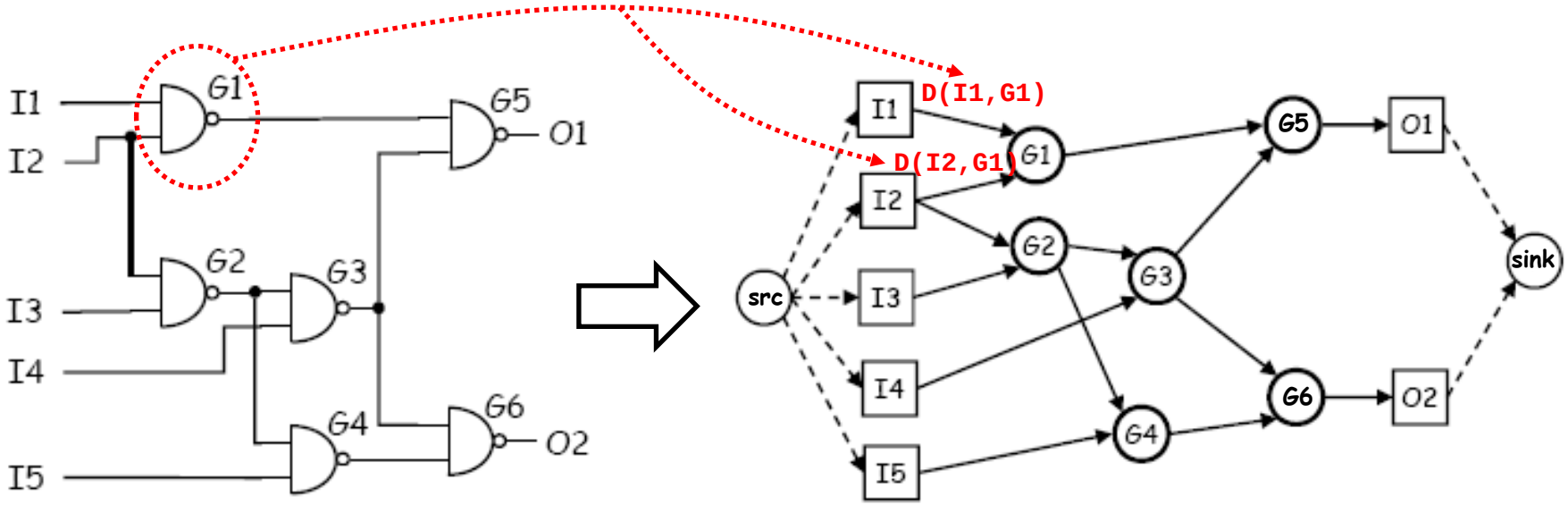
- Timing constraints on combinational logic are imposed by requirements for correct operation in sequential circuits
  - Setup and hold condition
- Various delay models for gates / cells
  - Unit, constant, pin-to-pin, load/slew-rate dependent, state dependent, PVT-corners, statistical
- Given a delay model, how to analyze combinational logic for timing?

Approaches to Timing Analysis

- Simulation
  - Very accurate (can use complex delay models)
  - Very slow (need to try all input vectors unless we are given the "worst case" input)
- Static Timing Analysis
  - Topological
    - Fast (single traversal of circuit)
    - Conservative (can lead to pessimistic estimate)
    - Best suited for repeated use in synthesis
  - Functional
    - Reduce pessimism of topological analysis
      - Account for false paths
    - Slower (involves solution to a search problem)
- Combination of both approaches may be used in practice

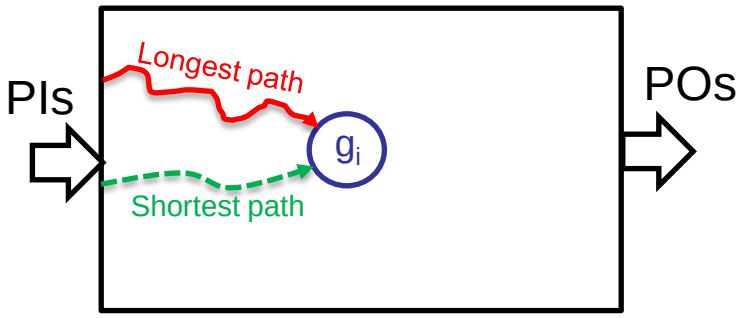
Topological Timing Analysis

- Timing graph
  - Representation of combinational logic circuit as a directed acyclic graph (DAG)
  - Add dummy source and sink vertices
  - Do not care about the function of each gate, only it's delay
  - Annotate delays on edges of timing graph
  - Flexible: Pin-to-pin delays, multiple delays for different transitions, delays based on load and slew rate



Topological Timing Analysis

- Latest arrival time: Latest time a which a gate's output becomes *stable*
  - Longest path from primary inputs to the gate output
- Earliest arrival time: Earliest time at which a gate's output can become *unstable*
  - Shortest path from primary inputs to the gate output



Timing analysis = Finding longest/shortest paths through the timing graph

Topological Timing Analysis

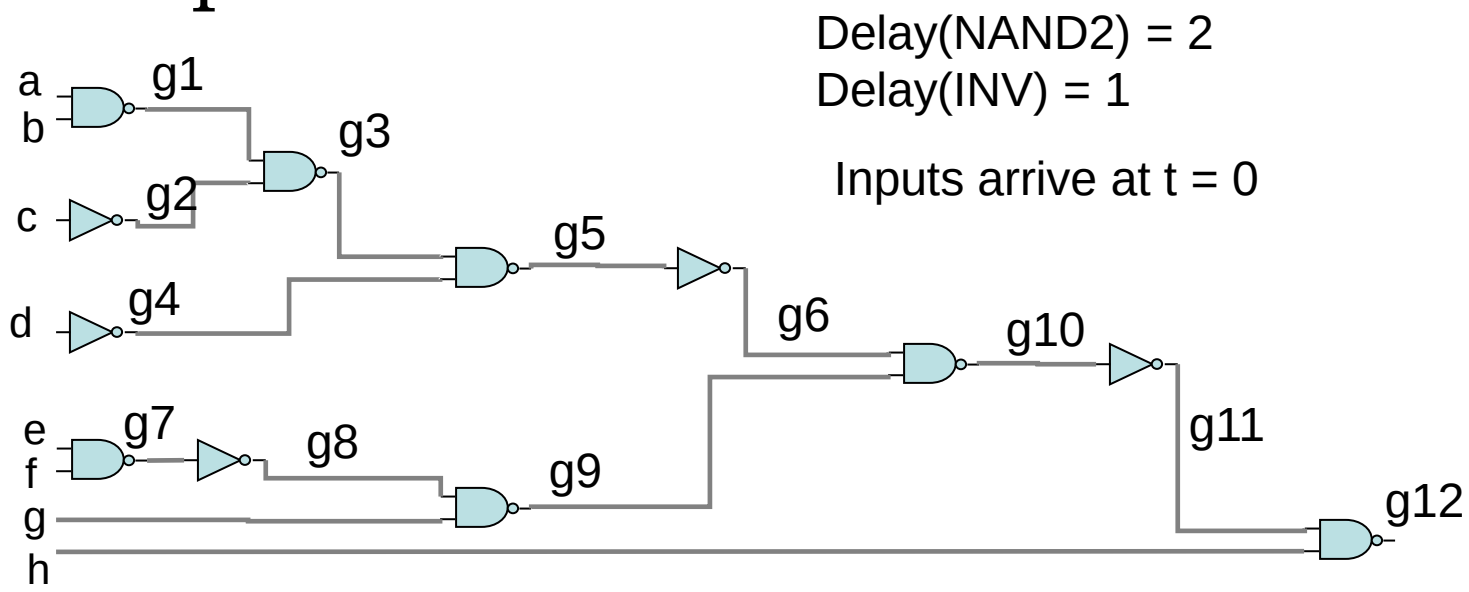
- Procedure for computing the (longest) topological delay of a circuit

1. Construct timing graph
2. Sort nodes in topological order
3. For each node in sorted order
   Compute (latest) arrival time $AT(v_i)$ as $\max_{v_j \in inputs(v_i)} \{AT(v_j) + D(v_j, v_i)\}$

NOTE: Above pseudo-code assumes pin-to-pin delay model.
If using load and slew rate dependent model, also need to propagate slew rates and use them along with loads to compute delay of each gate.
For transition-dependent delay model, propagate separate rising and falling arrival times

Topological Timing Analysis

- Example:



Topological order:

a, b, c, d, e, f, g, h, g1, g2, g4, g7, g3, g8, g5, g9, g6, g10, g11, g12

Latest Arrival Times:

a: 0, b: 0, c: 0, d: 0, e: 0, f: 0, g: 0, h: 0
g1: 2, g2: 1, g4: 1, g7: 2
g3: 4, g8: 3
g5: 6, g9: 5, g6: 7
g10: 9, g11: 10, g12: 12

Computing Earliest and Latest Arrival Times Using min-max Delays

- A single traversal of the timing graph can be used to compute both earliest and latest arrival times
  - Similar algorithm as before, except
    - Gate delays are specified as min-max ranges
    - Propagate earliest and latest arrival times

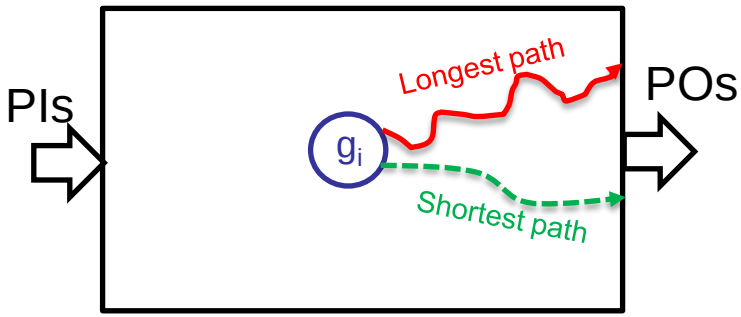
1. Construct timing graph
2. Sort nodes in topological order
3. For each node in sorted order
   a. Compute latest arrival time $AT_L(v_i)$ as $\max_{v_j \in inputs(v_i)} \{AT_L(v_j) + D_{max}(v_j, v_i)\}$
   a. Compute earliest arrival time $AT_E(v_i)$ as $\min_{v_j \in inputs(v_i)} \{AT_E(v_j) + D_{min}(v_j, v_i)\}$

Usage of Timing Analysis

- **Answer the question**: Will a circuit operate correctly at the intended clock frequency
  - Yes/No
  - Check whether earliest and latest arrival times at the primary outputs satisfy the setup/hold conditions
- **Help in answering the question**: What can I do to improve the timing of the circuit?
  - Diagnostic information (why not?)
  - Identify targets for improvement (where should I focus re-design effort?)
  - Guide synthesis towards better implementations

Required Times

- Given a circuit and a timing constraint on it's primary outputs
  - **Latest required time**: Latest time at which the output of a gate may become stable without causing violation of a timing constraint
    - Longest path from the gate output to the primary outputs
  - **Earliest required time**: Earliest time at which the output of a gate may become unstable without causing violation of a timing constraint
    - Shortest path from the gate output to the primary outputs



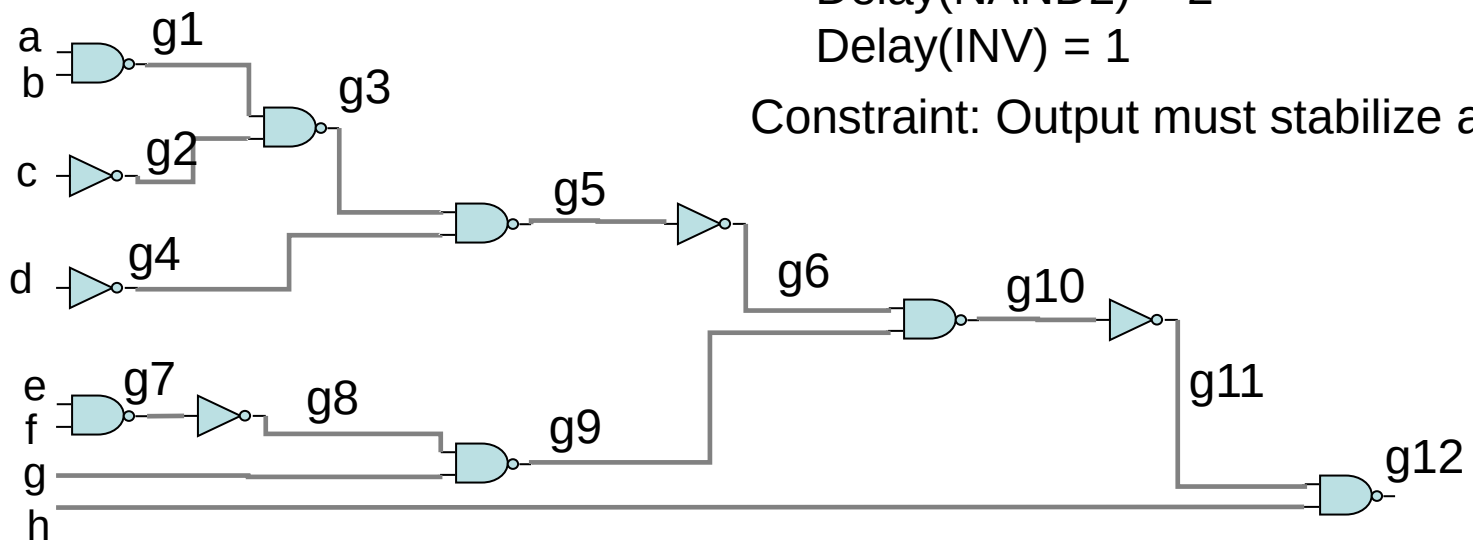
Computing Required Times

- Similar to ATs, except propagate backwards from the primary outputs

1. Construct timing graph
2. Sort nodes in topological order
3. Set RTs at primary outputs
4. For each node in **reverse** sorted order
   a. Compute latest required time $RT_L(v_i)$ as $\min_{v_j \in fanouts(v_i)} \{RT_L(v_j) - D_{max}(v_i,v_j)\}$
   b. Compute earliest required time $RT_E(v_i)$ as $\max_{v_j \in fanouts(v_i)} \{RT_E(v_j) - D_{min}(v_i,v_j)\}$

Required Times: Example

- Example:



Delay(NAND2) = 2
Delay(INV) = 1

Constraint: Output must stabilize at t=10

Topological order:

a, b, c, d, e, f, g, h, g1, g2, g4, g7, g3, g8, g5, g9, g6, g10, g11, g12

Latest Required Times:

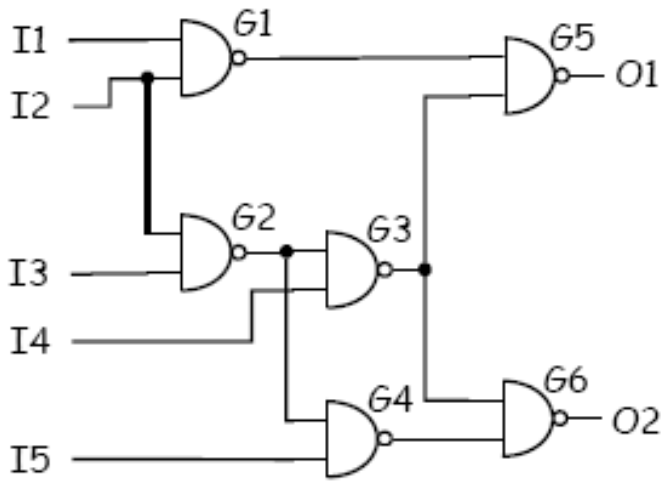
g12: 10
g11: 8, g10: 7
g6: 5, g9: 5, g5: 4,
g8: 3, g3: 2, g7: 2, g4: 2, g2: 0, g1: 0, h: 8, g: 3, f: 0, e: 0, d: 1, c: -1, a: -2, b: -2

Timing Slack

- Slack at a node is equal to the Required Time – Arrival Time
- Reflects the timing margin available before the node becomes “critical”
- Negative slack means that a timing constraint is not satisfied

Timing Critical Nodes

- A **critical node** is a node in the network that has the lowest slack
  - **Not unique**



Delay(NAND2) = 2

RT(O1) = RT(O2) = 5

Topological order:

Arrival times:

Required times:

Slacks:

Critical nodes:

Timing Critical Paths

- Interesting Property: The set of timing critical nodes forms a connected network consisting of paths from PIs to POs
  - Called **critical paths**

Algorithm to find critical paths

1. Construct timing graph
2. Sort nodes in topological order
3. Compute AT, RT, and Slack at each node
4. $Slack_{min} = min_{all\ nodes\ vi}(Slack(v_i))$
5. Trace path(s) through the network where each node has Slack = $Slack_{min}$

ε-critical Network

- In practice, we may be interested in paths that are critical or "close-to-critical"
  - Close-to-critical paths could become critical when we optimize the critical paths
  - Uncertainty / margin of error in delay models
- ε-critical node: A node v_i is **ε-critical** if $Slack(v_i) \leq Slack_{min} + \varepsilon$
- Interesting Property: The set of ε-critical nodes forms a connected network consisting of paths from PIs to Pos
  - Called the ε-critical network

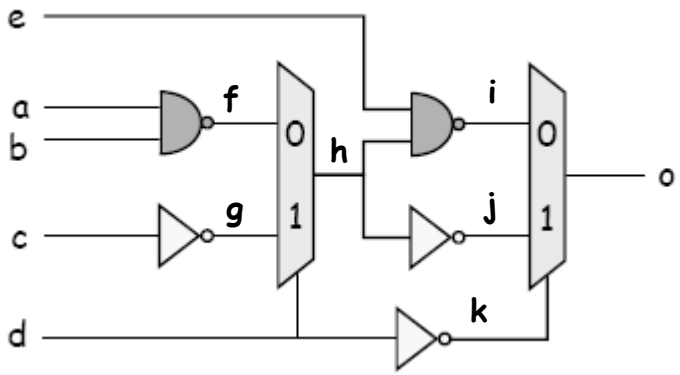
Summary: Topological Timing Analysis

- Computation of Arrival Times and Required Times
- Slack represents the timing margin at a node
- Critical paths: Paths with minimum slack

The False Path Problem

- Problem: Topological timing analysis may be pessimistic!
  - Ignores functionality of the nodes in the circuit
- Some paths can never be responsible for determining the delay of a circuit
  - Called **"false" paths**
- A path is false if no sequence of input vectors can result in an event propagating along it

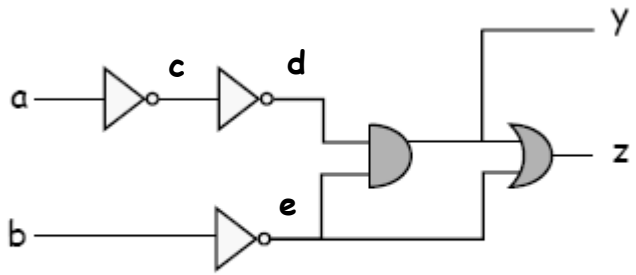
False Path Examples



Delay(NAND2) = 2, Delay(INV)= 1,
Delay(MUX) = 2

Longest Path:

False?



Delay(AND) = 2, Delay(INV)= 1,
Delay(OR) = 2

Longest Path:

False?

False Path: Real Example

- 2-bit carry-bypass adder