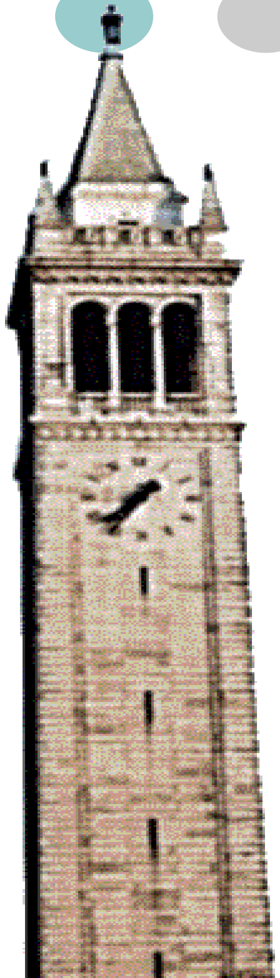


Fundamental Algorithms for System Modeling, Analysis, and Optimization

Edward A. Lee, Jaijeet Roychowdhury, Sanjit A. Seshia
UC Berkeley
EECS 244
Fall 2016
Lecturer: Yu-Yun Dai

Timing Analysis

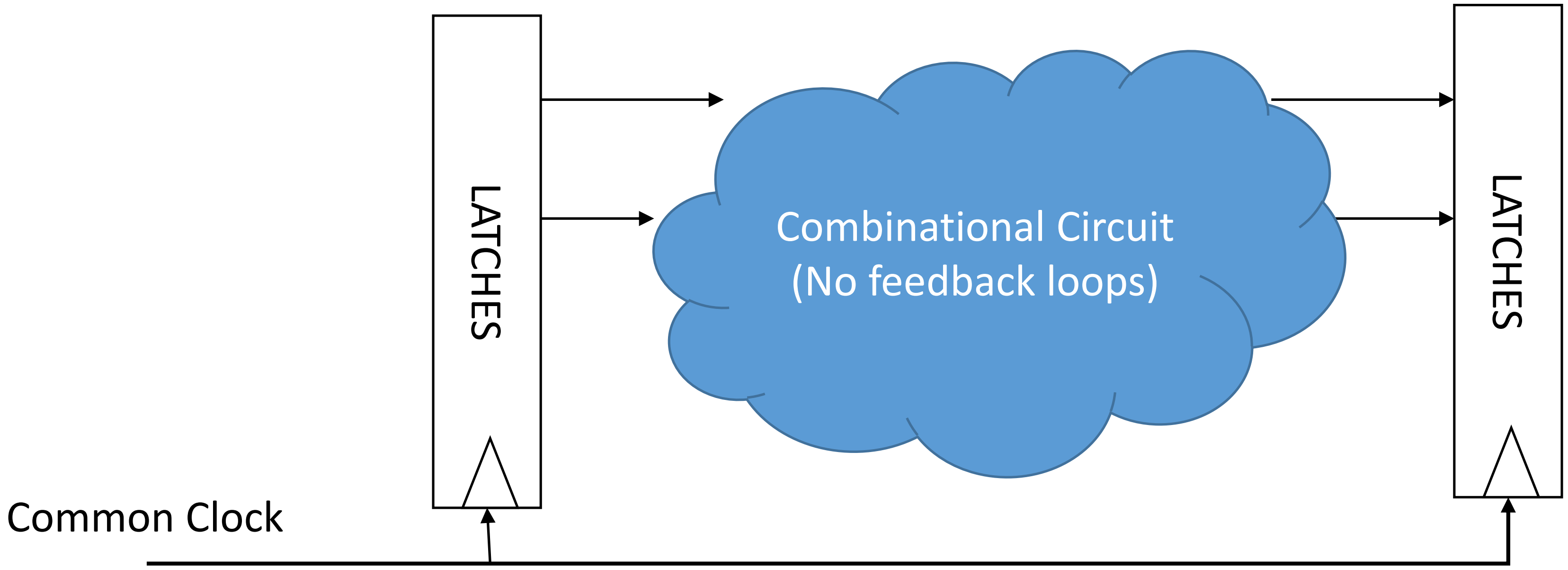
Thanks to Kurt Keutzer for several slides

Why Does Timing Analysis Matter?

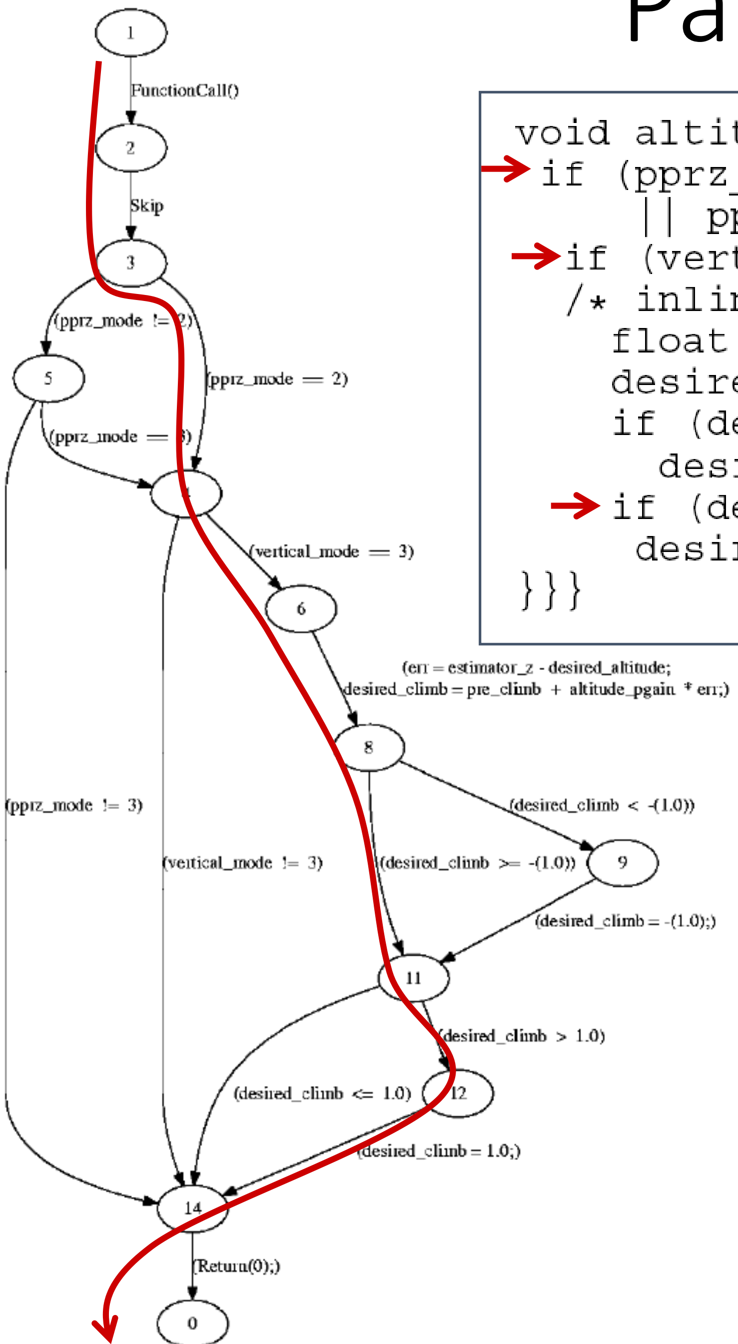
- (Clock) Speed is one of the major performance metrics for digital circuits

Timing Analysis = the process of verifying that a chip meets its speed requirement

- Determine fastest permissible clock speed (e.g. 1 GHz) by determining delay of longest path from register to register (e.g. 1ns.)



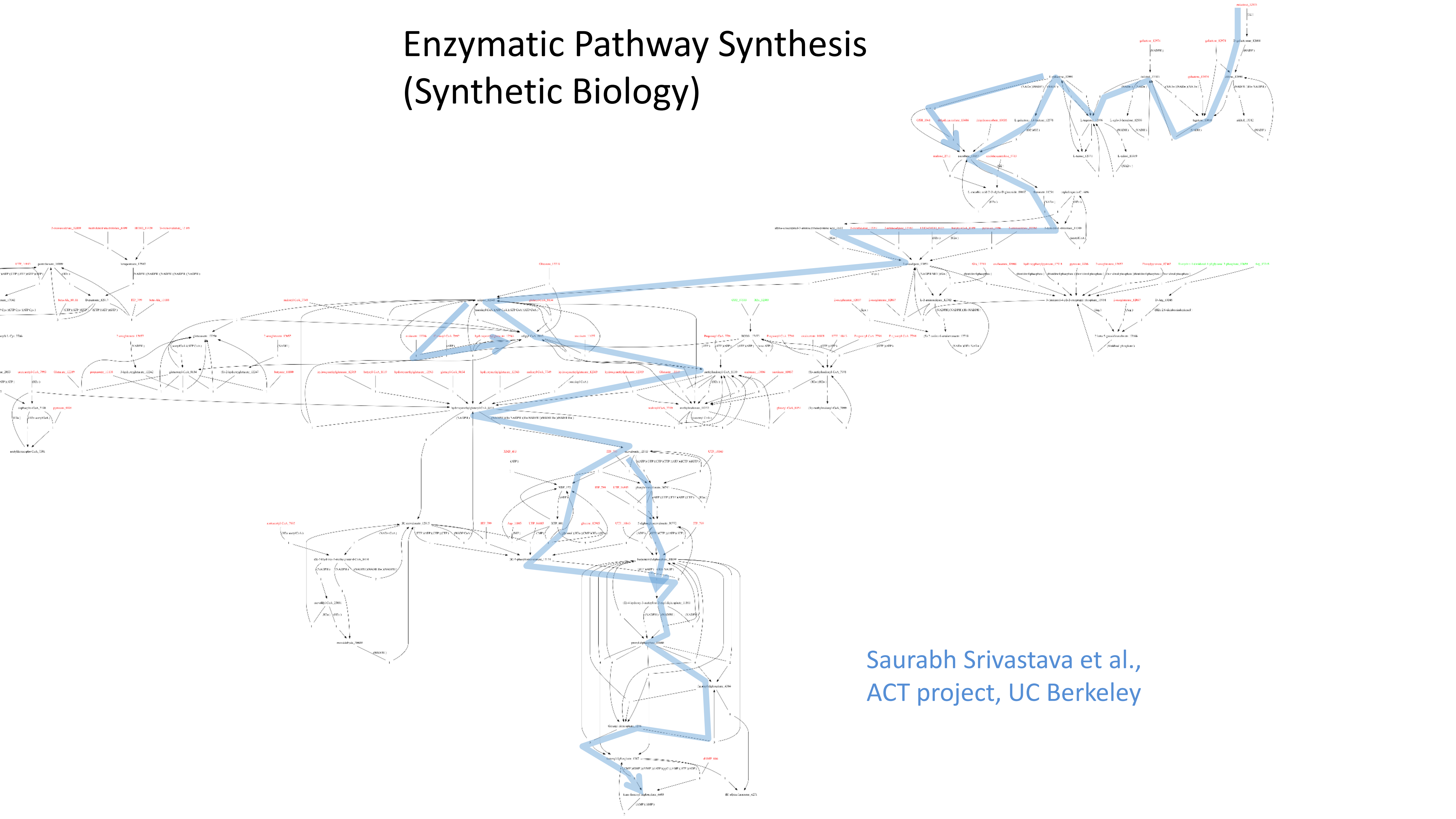
Path Space in Program




```
void altitude_control_task(void) {
 if (pprz_mode == PPRZ_MODE_AUTO2
     || pprz_mode == PPRZ_MODE_HOME) {
  if (vertical_mode == VERTICAL_MODE_AUTO_ALT) {
  /* inlined below: function  altitude_pid_run(); */
    float err = estimator_z - desired_altitude;
    desired_climb = pre_climb + altitude_pgain * err;
    if (desired_climb < -CLIMB_MAX)
      desired_climb = -CLIMB_MAX;
    if (desired_climb > CLIMB_MAX)
     desired_climb = CLIMB_MAX;
}}}
```

Must find:
Longest path in the control-flow graph (CFG)

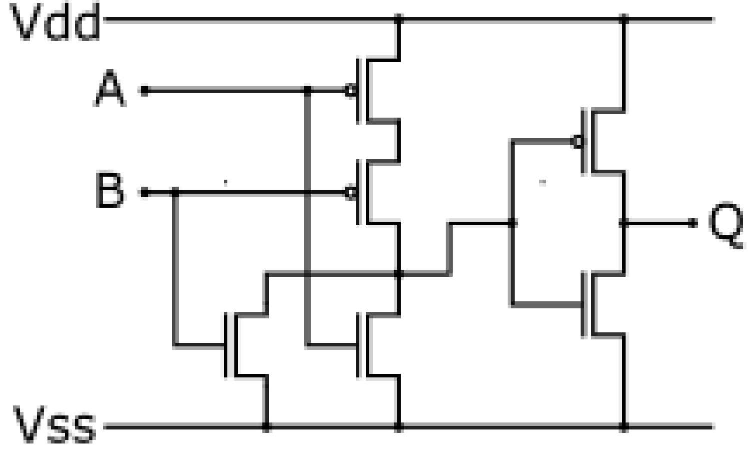
Enzymatic Pathway Synthesis
(Synthetic Biology)



Saurabh Srivastava et al.,
ACT project, UC Berkeley

Timing Analysis for Circuits

- Consider a signal in a clocked design:
  - The value varies between one (high-voltage) and zero (low-voltage)
  - Changes can occur at different times in each cycle
    - Time required for change depends on input patterns
    - May not change at all in some cycles
    - May make multiple changes before settling to a final value



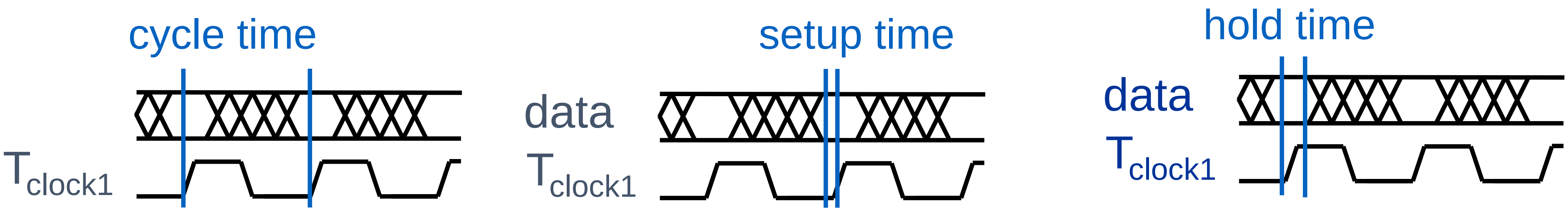
Static Timing Analysis

- “Static” means we are not doing simulation (dynamic)
- Consider the worst case
  - Assume that signal becomes *stable* at latest possible time
  - Assume signal becomes *unstable* at the earliest possible time

Real circuits are made from gates made out of transistors…….

Timing Analysis: Basic Model

- Set up/Cycle time: Does data always ***reach*** a stable value at all latch inputs in time for the clock to capture it?
  - Look at **late mode** timing, or **longest path**
- Hold time: Does data always ***stay*** stable at all latch inputs long enough after the clock to get stored?
  - Determine this by looking at **early mode** timing, or **shortest path**



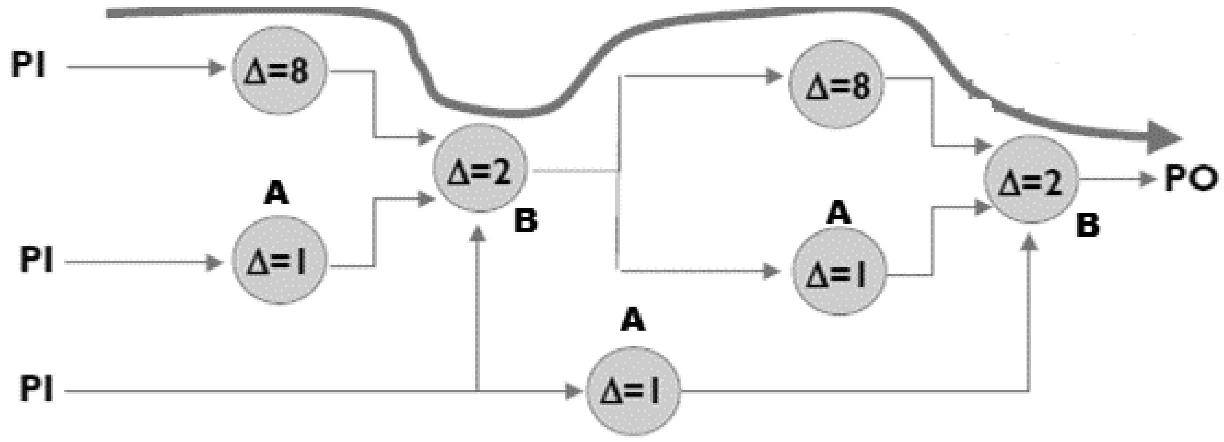
$T_{max} \leq T$

$T_{max} + T_{setup} \leq T$

$T_{min} \geq T_{hold} + T_{skew}$

Timing Analysis: Topological and Functional

- Do we worry about “gate function”?
  - Logical timing analysis: **YES**, We care “false path”
  - Topological timing analysis: **NO**, we only worry about the delay through the paths => overestimate

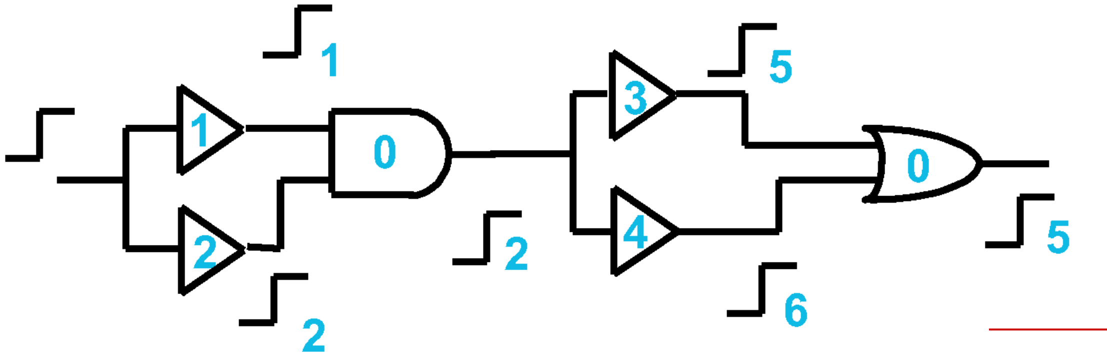


We will Learn

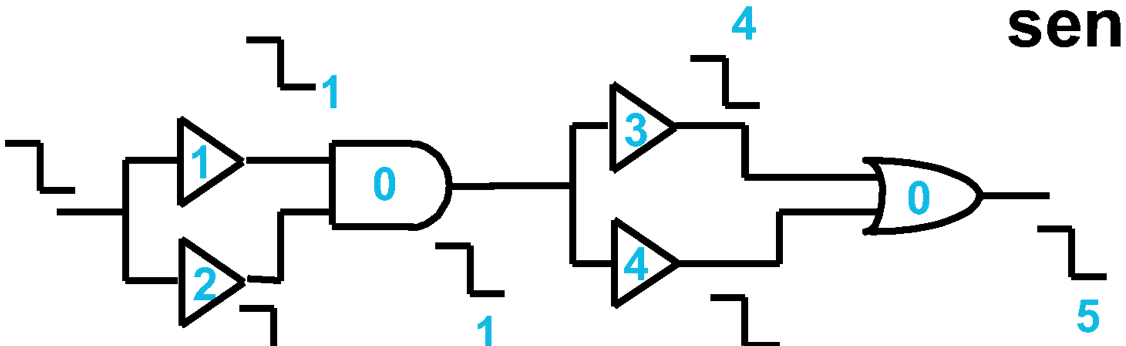
- False path v.s. True path
  - Static Sensitization
  - Static Co-sensitization
- Algorithms to find the longest path in a DAG
  - Incremental longest path in a DAG
  - Top k longest paths in a DAG
- Sequential synthesis: Retiming*

False Paths (consider Transition Mode)

A path is false if it cannot be responsible for the delay of a circuit



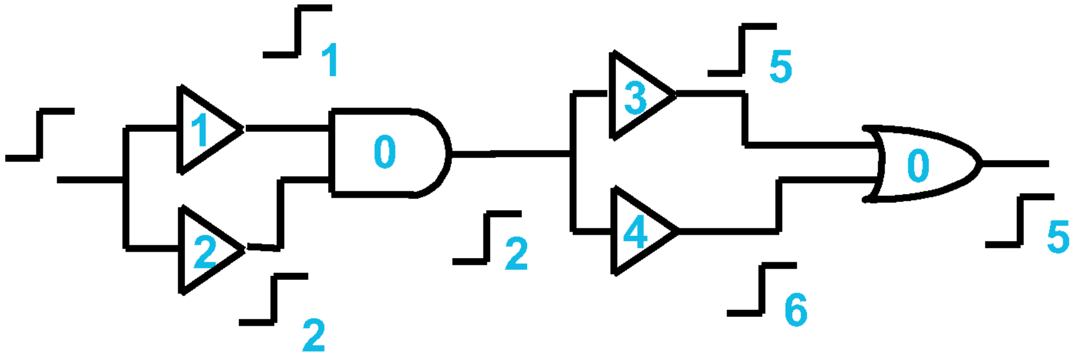
Length of
sensitized path = 5

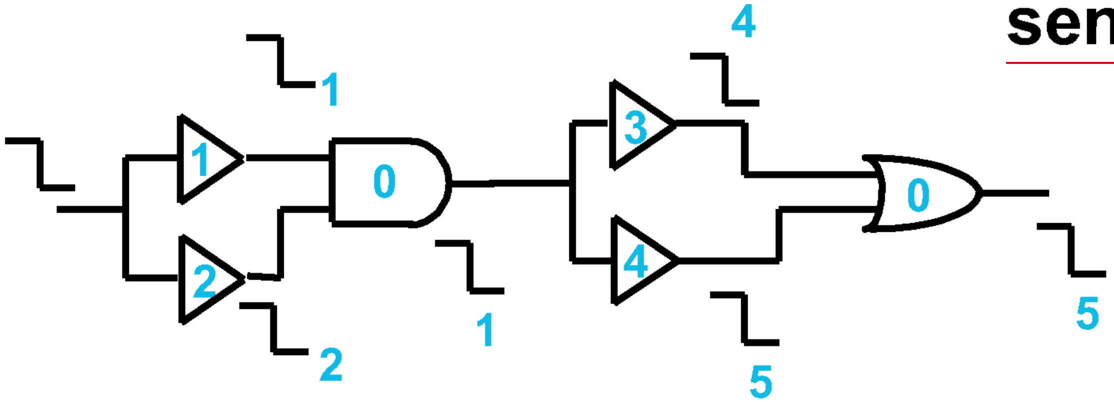


Graph model implies path of length 6

False Paths

A path is false if it cannot be responsible for the delay of a circuit





Length of sensitized path = 5

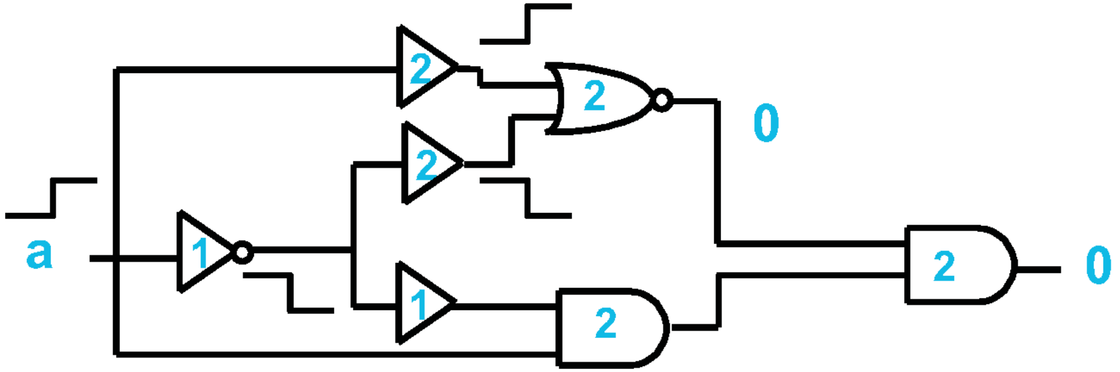
Graph model implies path of length 6

False vs. True Paths

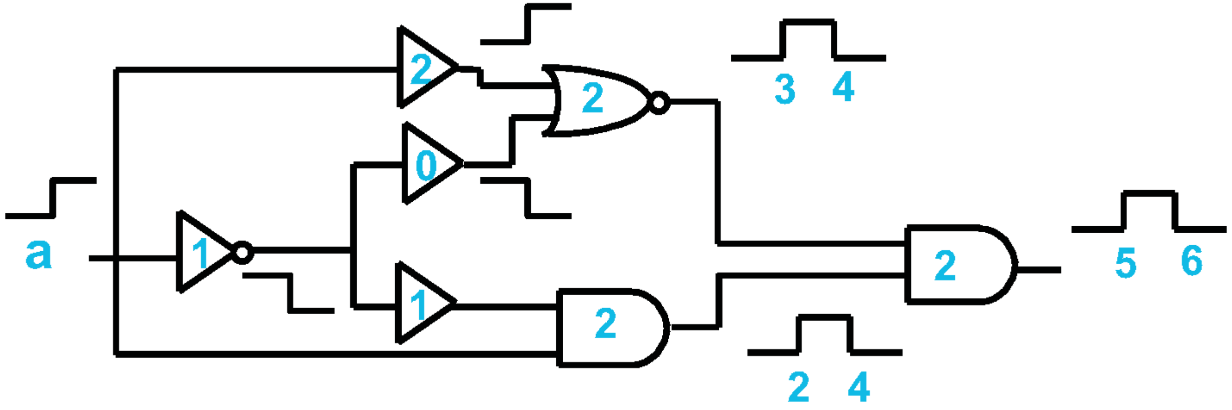
- TRUE path = one that can be responsible for the delay of a circuit

- Need techniques to find whether a path is TRUE or FALSE

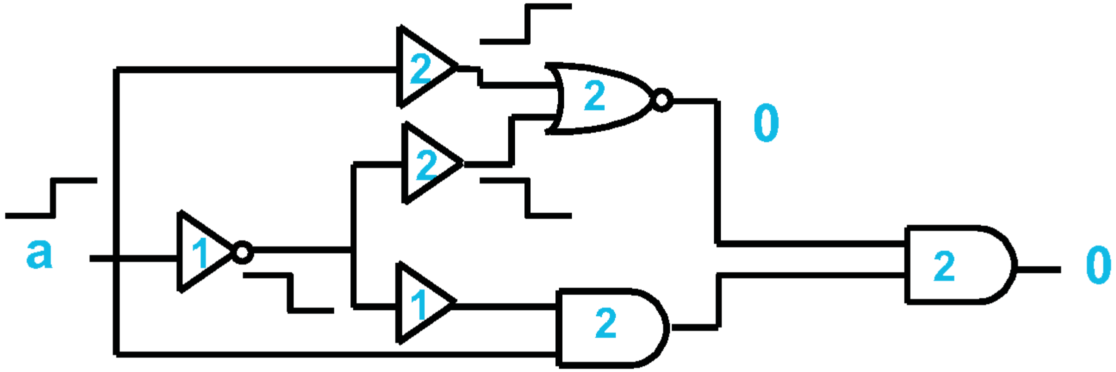
The Fixed Delay Model: Constant Delay for Each Gate (or Wire)



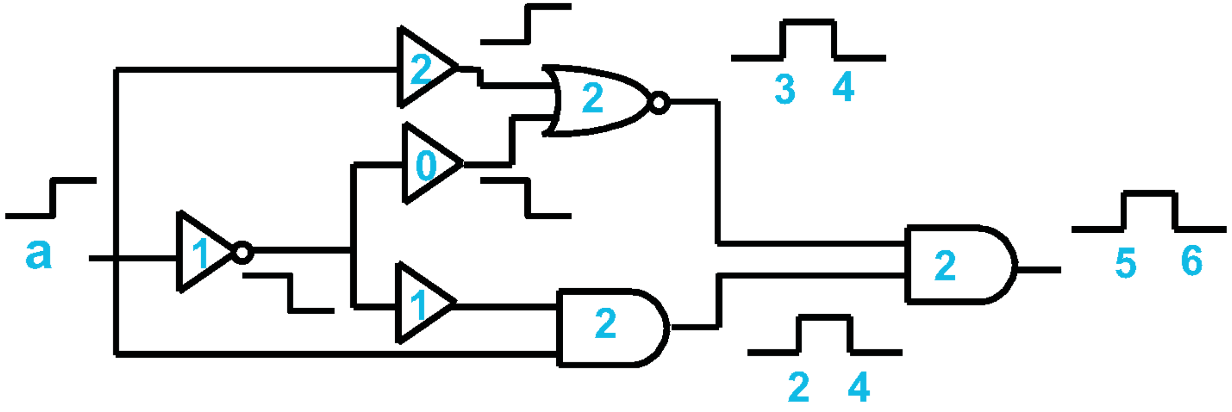
Transition delay is 0, for both input transitions.
Consider the "faster" circuit



Paradoxical Behavior with the Transition Model?



Transition delay is 0, for both input transitions.
Consider the "faster" circuit



Problems with Fixed Delay + Transition Model

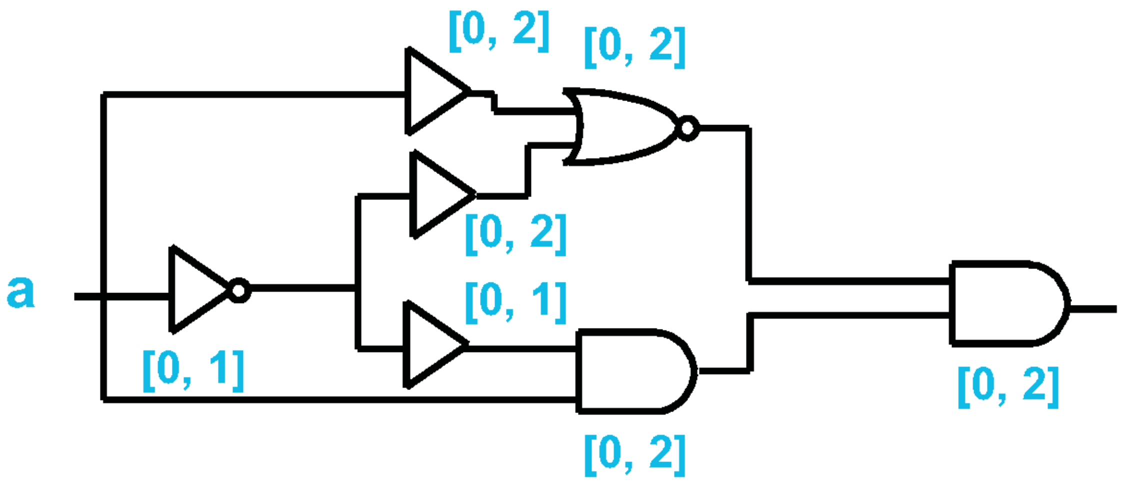
1. Transition model can be tricky to reason about
2. Fixed gate delays are unrealistic, due to manufacturing process variations

- More realistic delay model: *Lower and upper bounds*

- Perform timing analysis for a whole family of circuits that share the same lower/upper bounds

Fixed Delays → Bounded Delays

Want algorithms that report the **critical path delay** of the **slowest circuit** in the circuit family



Delay of 6 for the above circuit for transition model (longest path that can propagate a transition)

Floating-Mode Delay Model

Input transition → Single input vector condition

Pessimistic, but easier to compute

Floating-Mode Delay Model

Assume an input pair <v_1, v_2> has been applied, but we only look at v_2 -- i.e. node values are unknown until set by v_2

(pessimistic because we assume any v_1 can be adversarially selected, to reason about long paths)

Assume the 1 at the input of the AND arrives before the 0 (even if in reality it arrives later and the gate output stays at 0 throughout, and no path is sensitized).

Roadmap for rest of lecture

- Consider conditions under which paths are TRUE or FALSE under the *floating-mode delay model with fixed delays*
- + under floating-mode model, fixed and bounded delays yield same worst-case circuit delay (for same upper bounds)
- + worst-case delay under floating-mode model is upper bound on that under the transition model

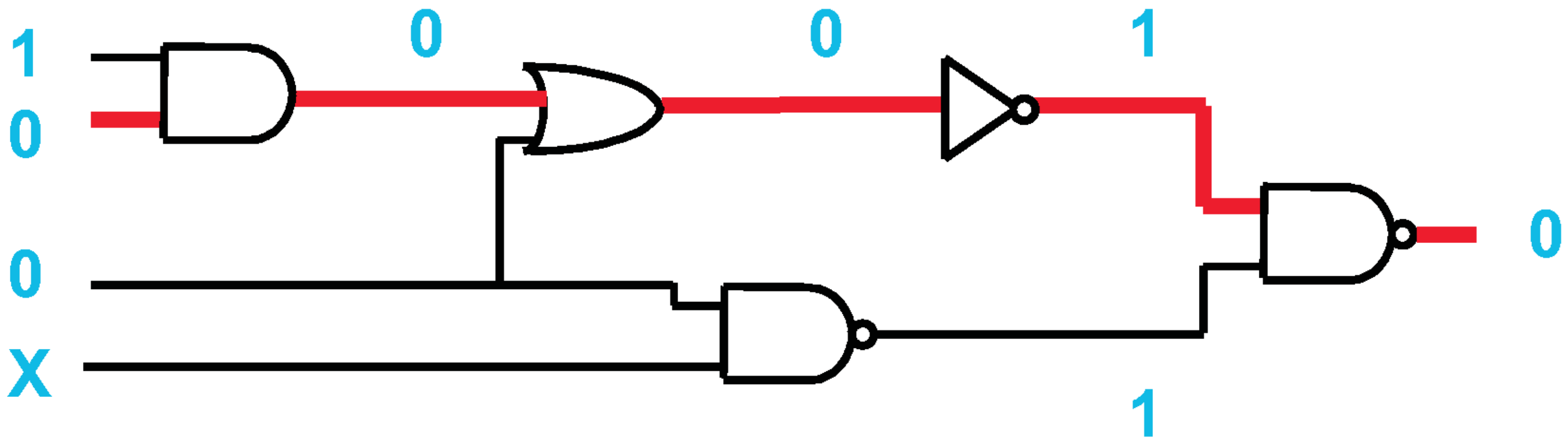
Controlling and Non-Controlling Values

- A **controlling value** at a gate input is the value that determines the output value of that gate irrespective of the other input value.
- (the output value is called a **controlled** value)

- A **controlling** value for an AND gate is 0 and for an OR gate is 1. (The controlled values are 0 and 1 resp.)
- A **non-controlling** value for an AND gate is 1 and for an OR gate is 0.
- What about NAND and NOR gates?

Static Sensitization

Definition: A path is statically sensitized by a vector V, if along each gate on the path, *if* the gate output is a controlled value, the input corresponding to the path is the *only* input with a controlling value

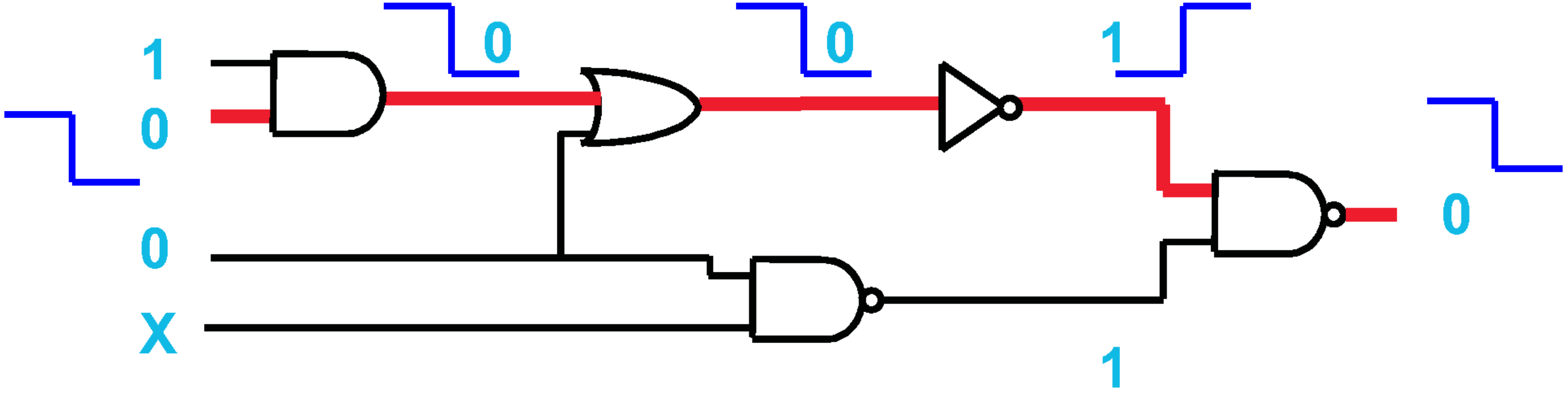


Input vector 100X statically sensitizes red path

Static Sensitization

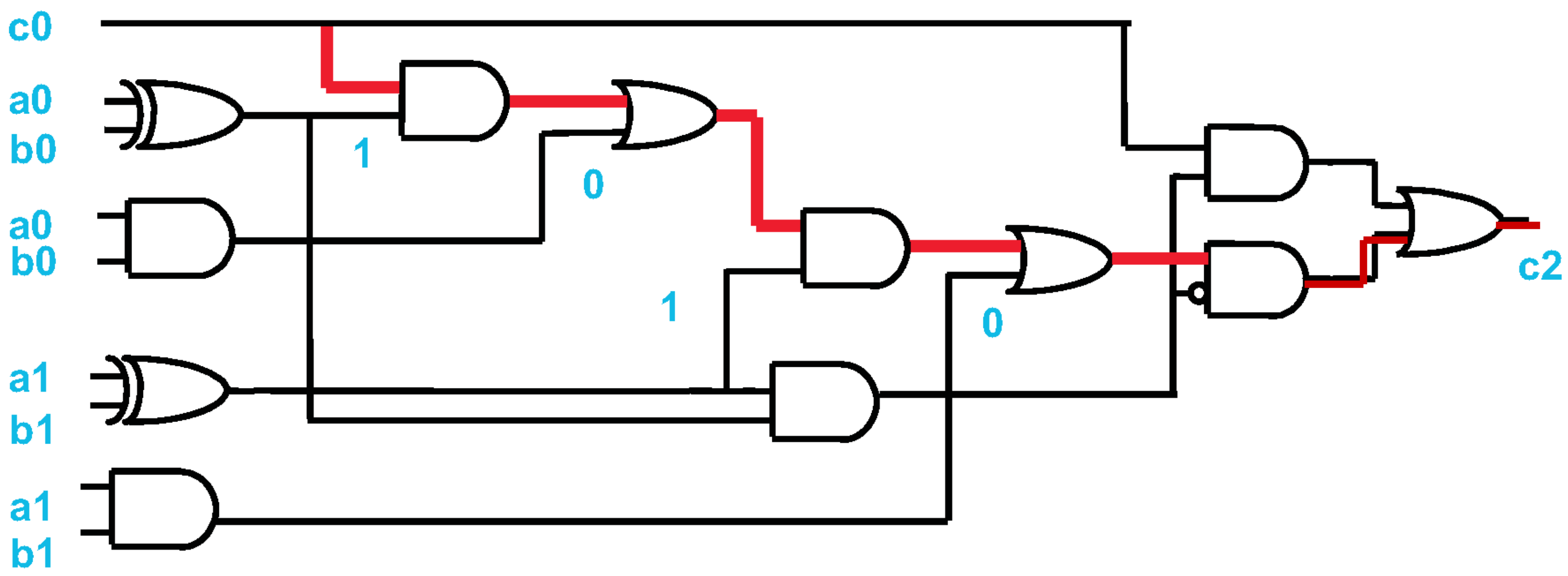
Static sensitization is **sufficient** for a path to be responsible for the delay of a circuit

WHY?



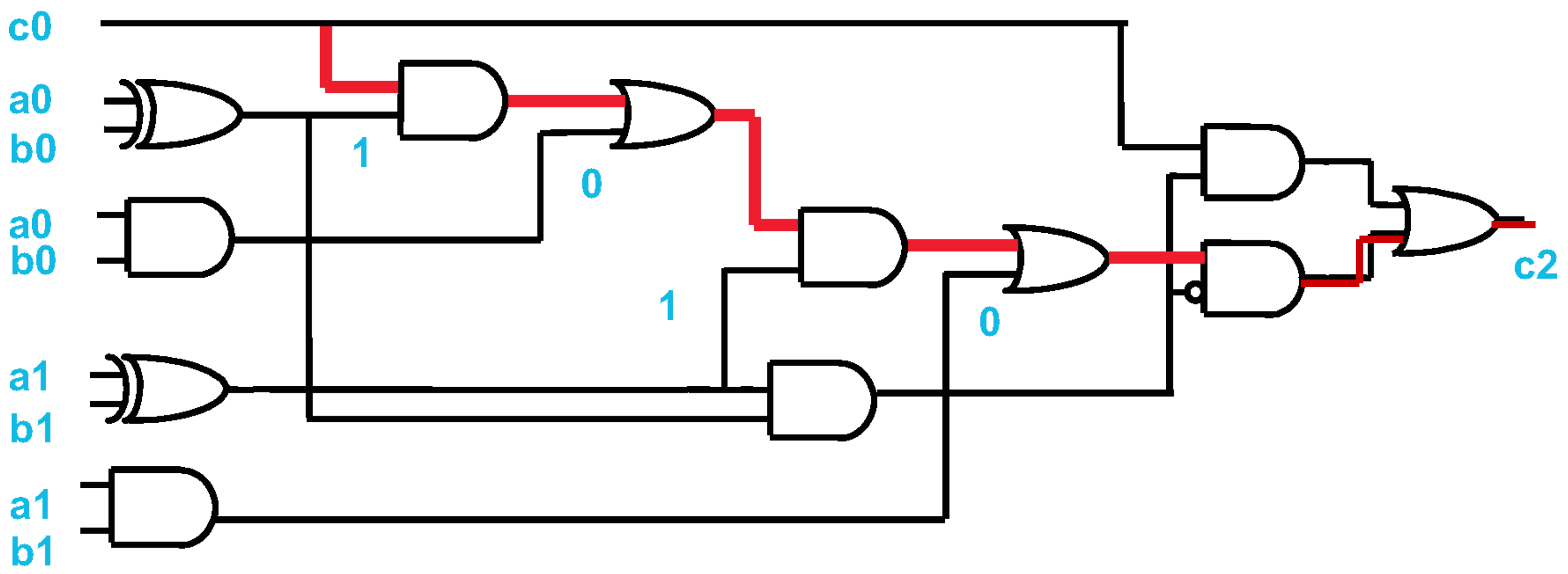
Input vector 100X statically sensitizes red path

Is this path statically sensitizable?



Definition: A path is statically sensitized by a vector V, if along each gate on the path, if the gate output is a controlling value, the input corresponding to the path is the only input with a controlling value

Is this path statically sensitizable?

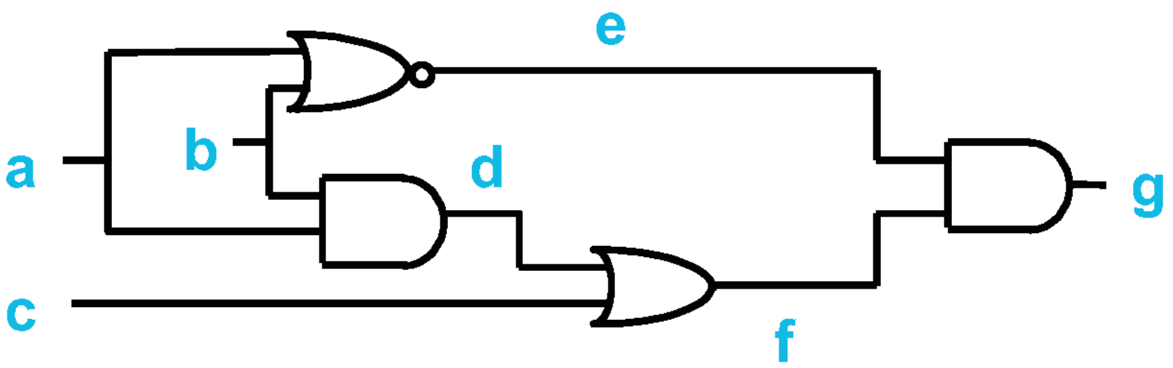


No, red path is NOT statically sensitizable (work this out)

More on Static Sensitization

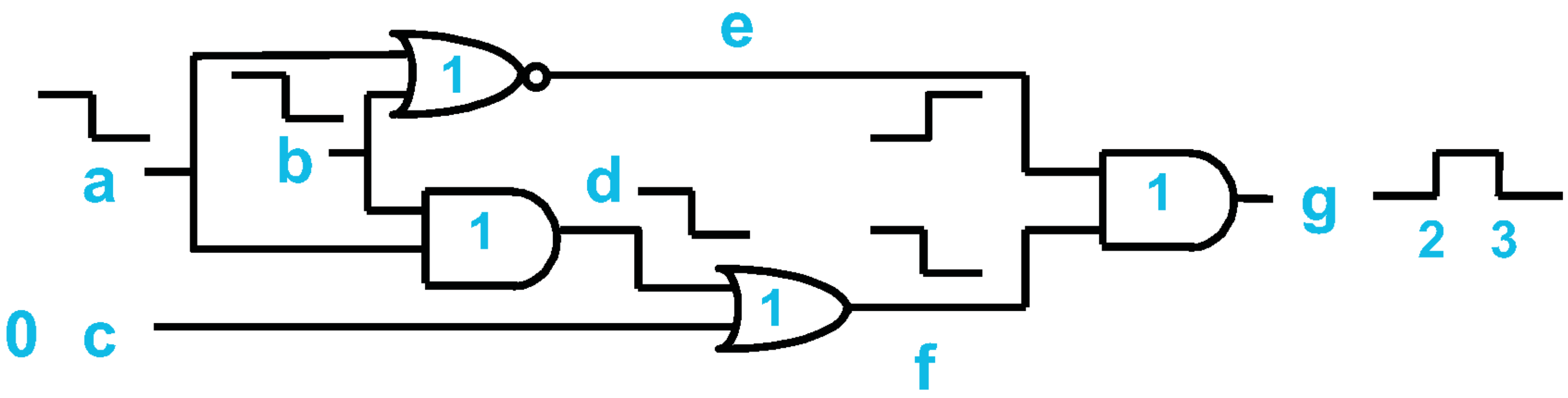
Are paths a,d,f,g and b,d,f,g statically sensitizable?

Are they true paths?



Static Sensitization is too strong

A true path (one that is responsible for delay of a circuit) need not be statically sensitizable



Paths a,d,f,g and b,d,f,g are NOT statically sensitizable.
But they are TRUE paths.

Static Co-sensitization

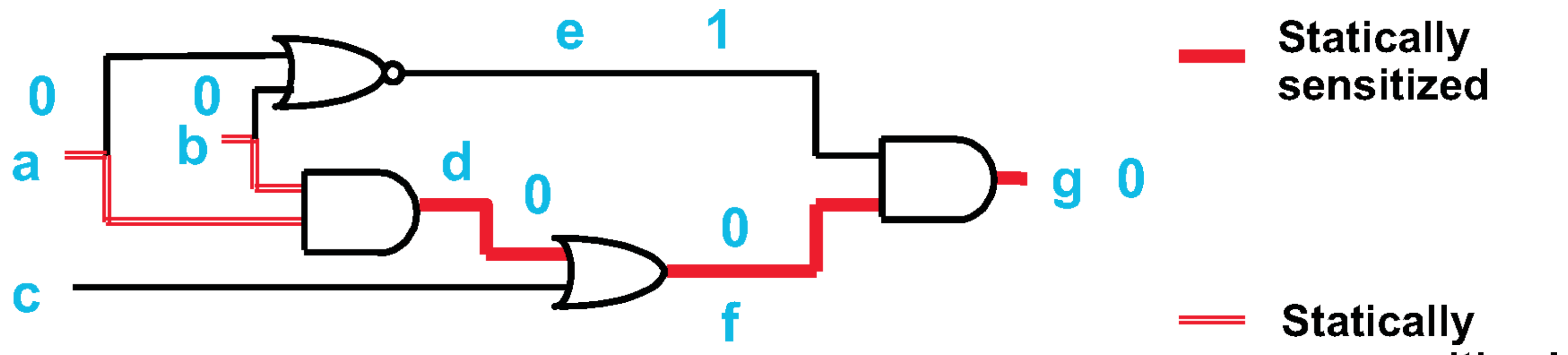
Definition: A path is statically co-sensitized by a vector V, if the input corresponding to the path presents a controlling value at each gate along the path whose output is a controlled value.

Not necessarily the ONLY controlling value

Static Co-sensitization

<u>Definition:</u> A path is statically co-sensitized by a vector V, if the input corresponding to the path presents a controlling value at each gate along the path whose output is a controlling value.

Not necessarily the ONLY controlling value

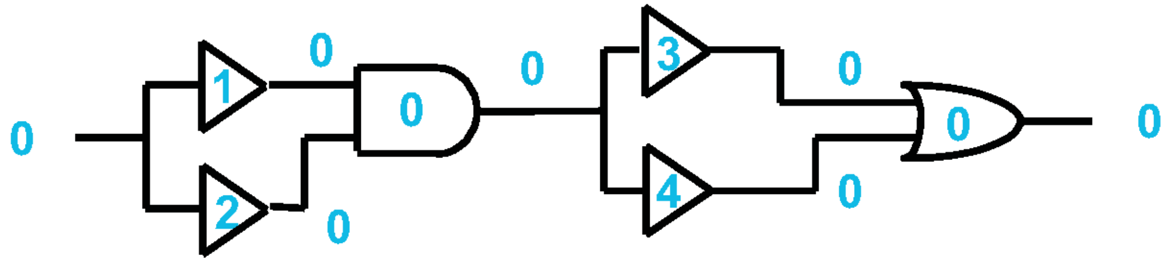


Paths a,d,f,g and b,d,f,g are statically co-sensitizable

Static Co-sensitization and Delay

Static co-sensitization is **necessary** for a path to be responsible for the delay of a circuit. (WHY?)

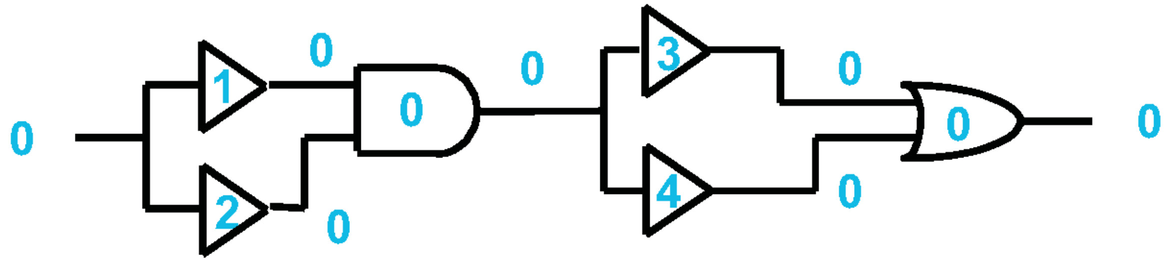
Is it sufficient?



Static Co-sensitization and Delay

Static co-sensitization is **necessary** for a path to be responsible for the delay of a circuit

But NOT sufficient



Path of length 6 is statically co-sensitized

Delay of circuit is 5 (as observed earlier)

Summary

Static sensitization (SS) sufficient for true path, but not necessary

Static co-sensitization (SC) necessary for true path, but not sufficient

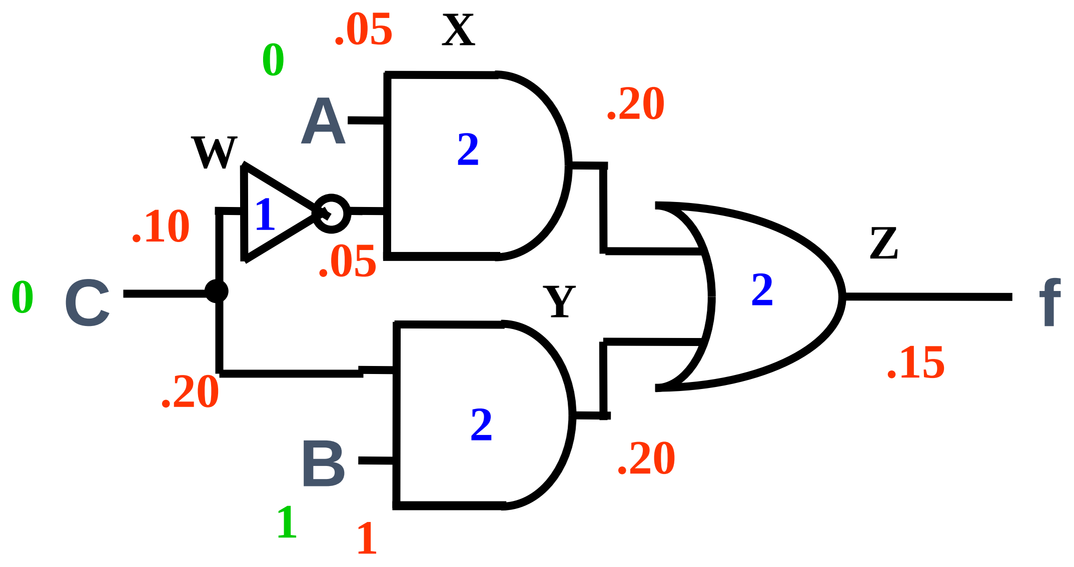
Determining whether a path is SS/SC can be formulated as a SAT problem

Modeling Timing in a Combinational Circuit

- Arrival time in green

Interconnect delay in red

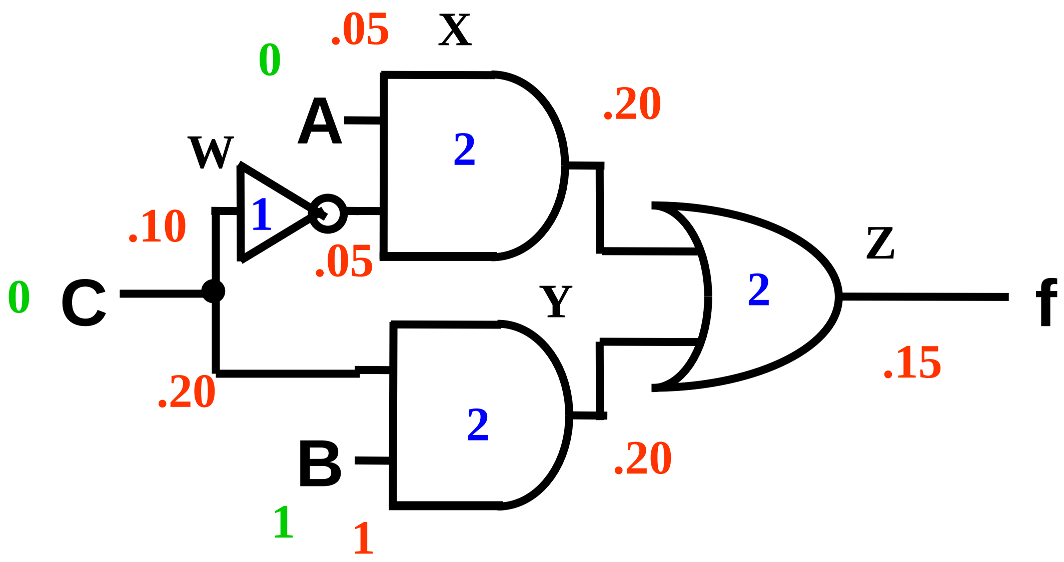
Gate delay in blue

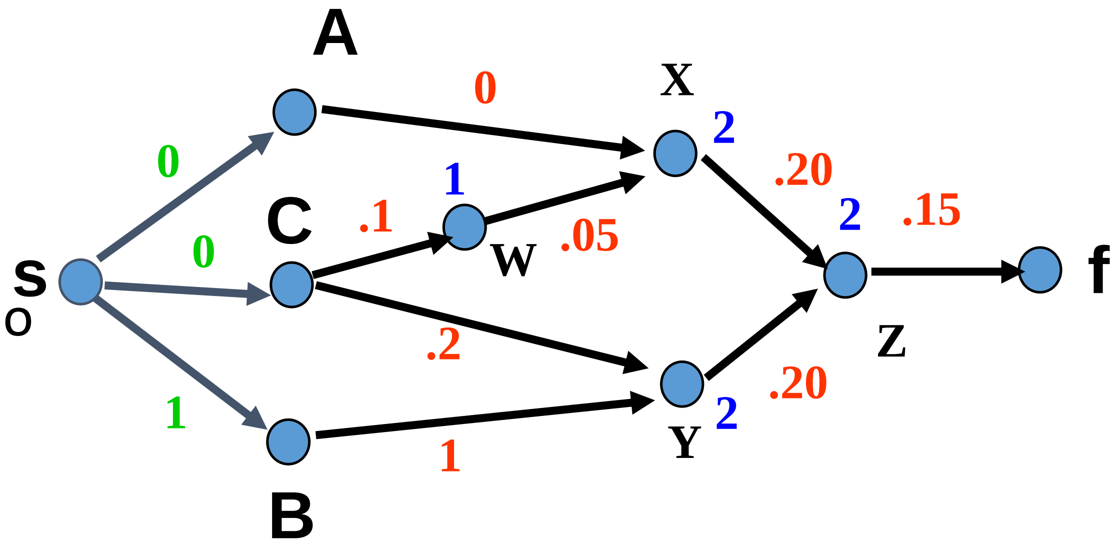


What's the right mathematical object to use to represent this physical object?

Modeling - 1

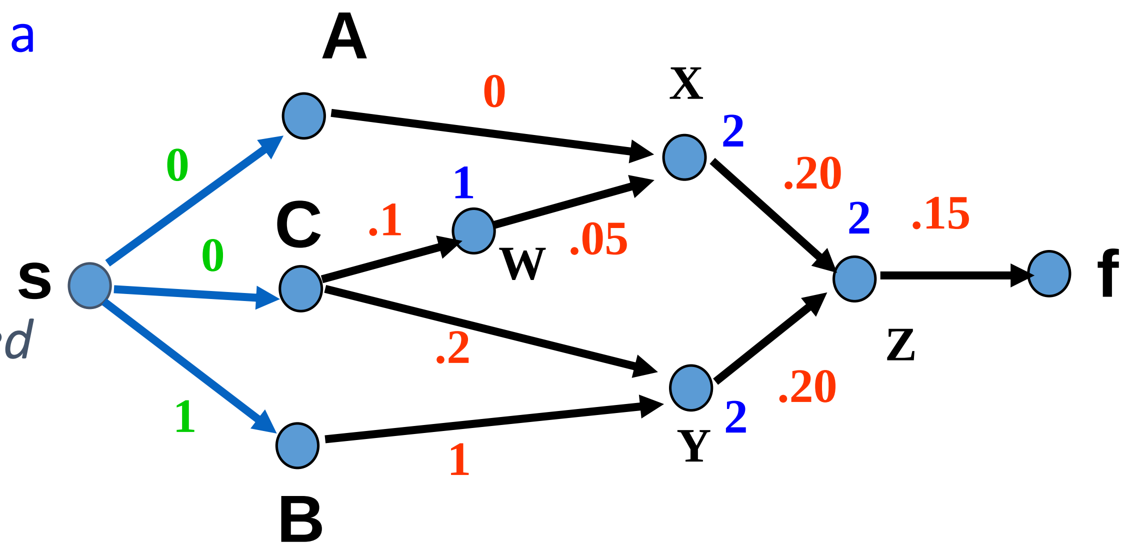
- Use a labeled *directed* graph
- *G = <V,E>*
- *Vertices* represent gates, primary inputs and primary outputs
- *Edges* represent wires
- *Labels* represent delays
- Now what do we do with this?



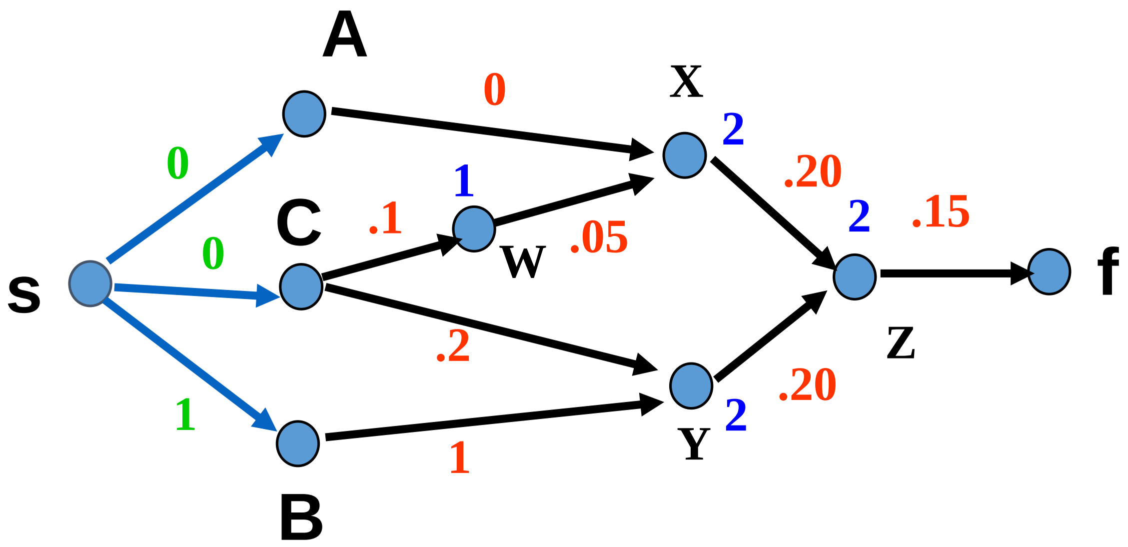


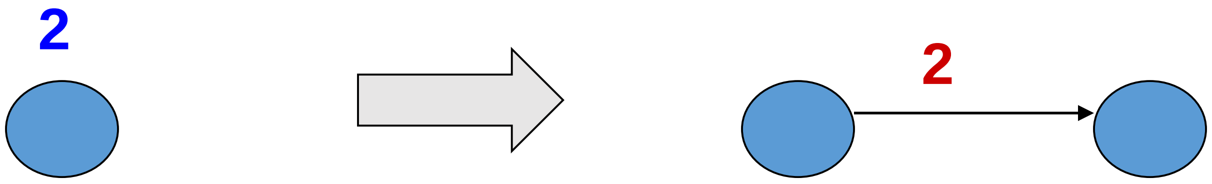
Modeling - 2

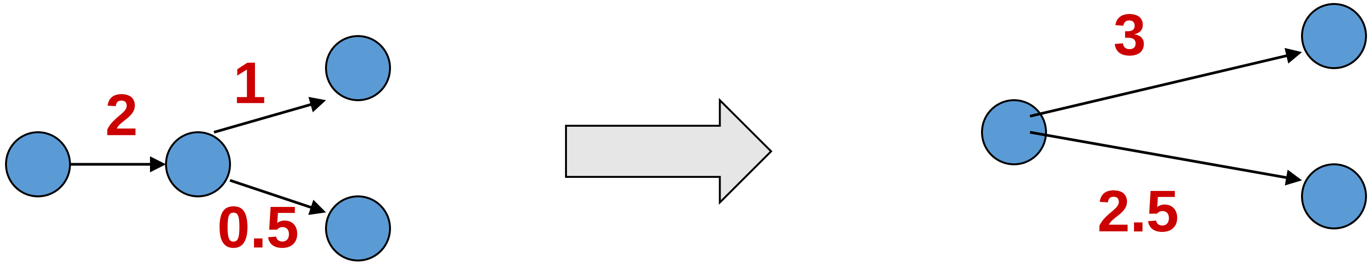
- Find longest path in a *directed* graph *G = <V,E>*
- *What sort of directed graph do we have?*
- *Is this in the standard form for a longest/shortest path problem?*



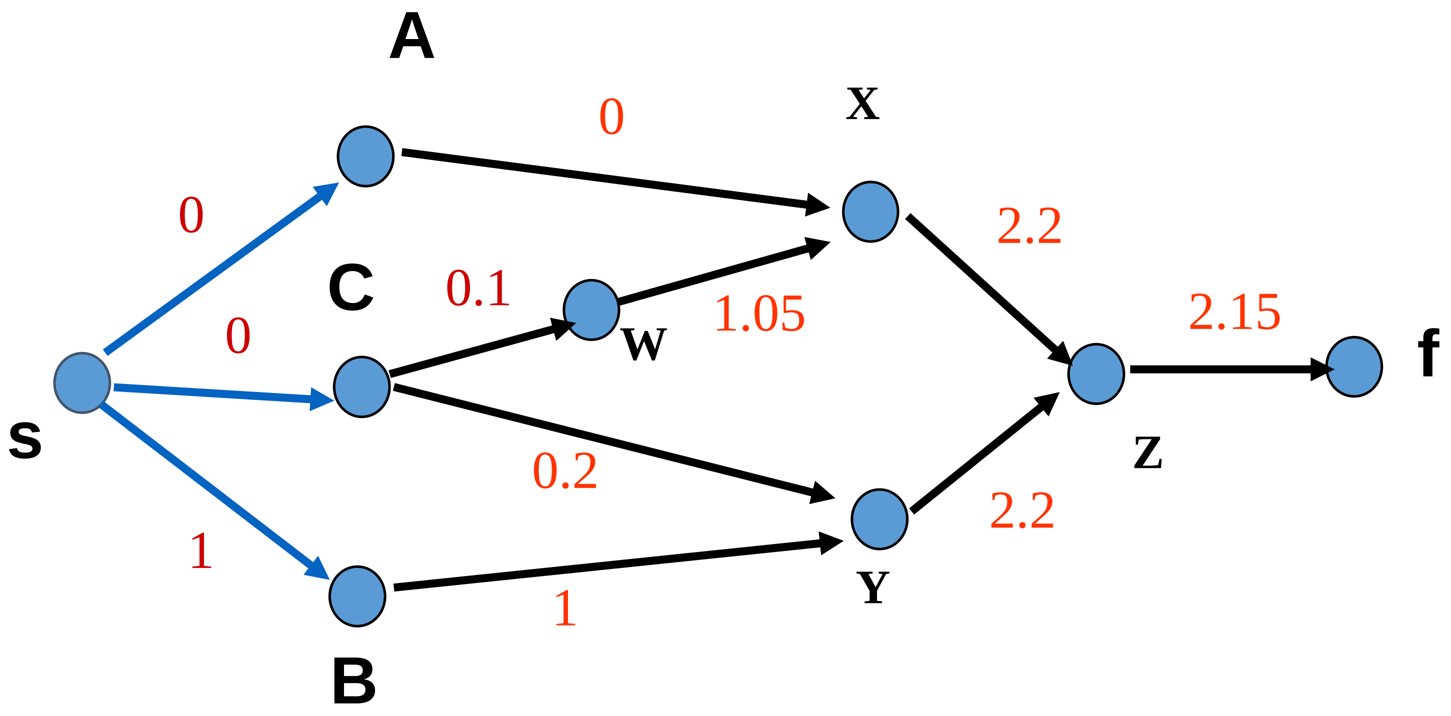
Split Nodes into Edges







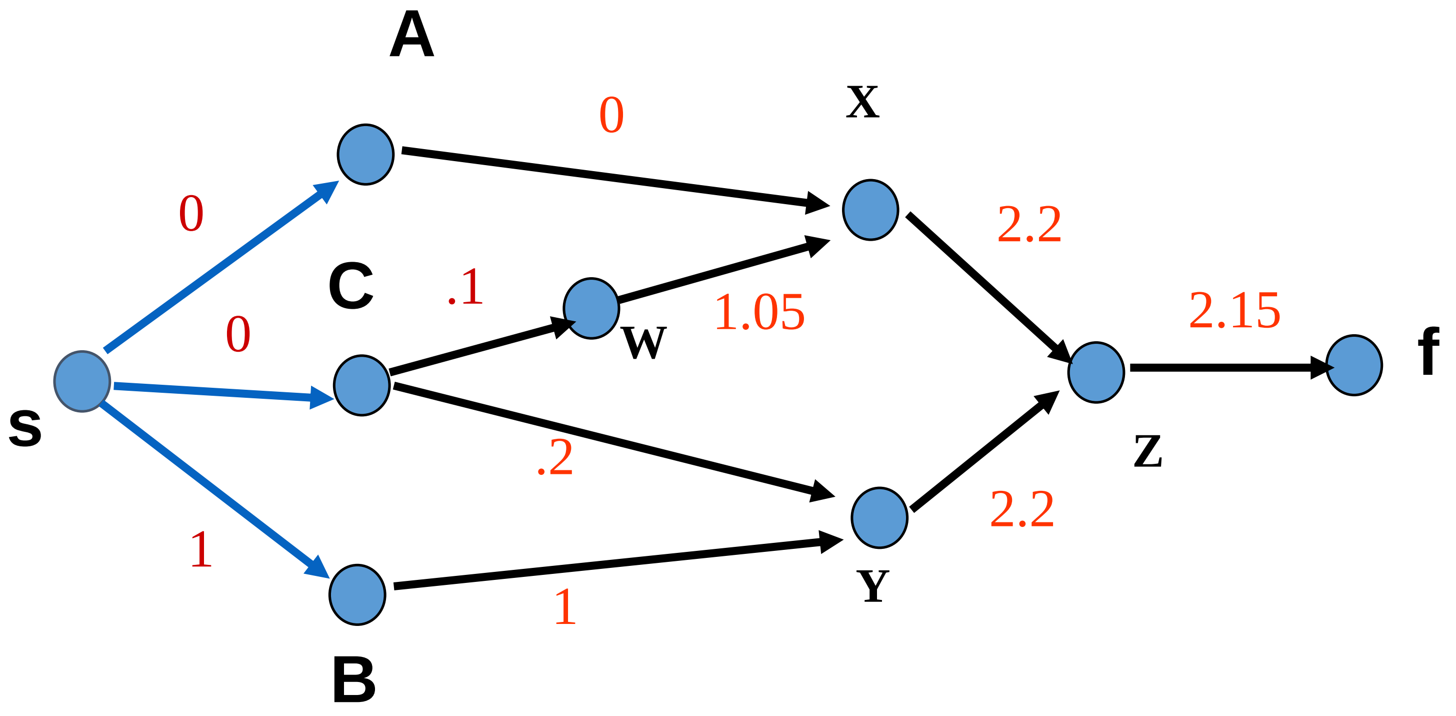
DAG with Weighted Edges



Problem: Find the longest (critical) path from source s to sink f.

Naïve Approach: Enumerate Paths

How many paths in this example?
In the worst case?



Problem:
Find the longest path from source s to sink f.

Algorithm 1: Longest path in a DAG

Critical Path Method **[Kirkpatrick 1966, IBM JRD]**

Let w(u,v) denote weight of edge from u to v

Steps:

1. Topologically sort vertices

   order: $v_1, v_2, \ldots, v_n$ $\quad v_1 = s, \; v_n = ?$

2. For each vertex v, compute

   d(v) = length of longest path from source s to v

   $d(v_1) = 0$

   For i = 2..n

   $d(v_i) = \max_{\text{all incoming edges } (u,\, v_i)} d(u) + w(u,v_i)$

Algorithm 1: Longest path in a DAG

Critical Path Method **[Kirkpatrick 1966, IBM JRD]**

Let $w(u,v)$ denote weight of edge from u to v

Steps:

1. Topologically sort vertices

   order: $v_1, v_2, \ldots, v_n$ $v_1 = s$, $v_n = f$

2. For each vertex v, compute

   $d(v)$ = length of longest path from source s to v

   $d(v_1) = 0$

   For $i = 2..n$

   $d(v_i) = \max_{\text{all incoming edges } (u, v_i)} d(u) + w(u,v_i)$

Time Complexity?

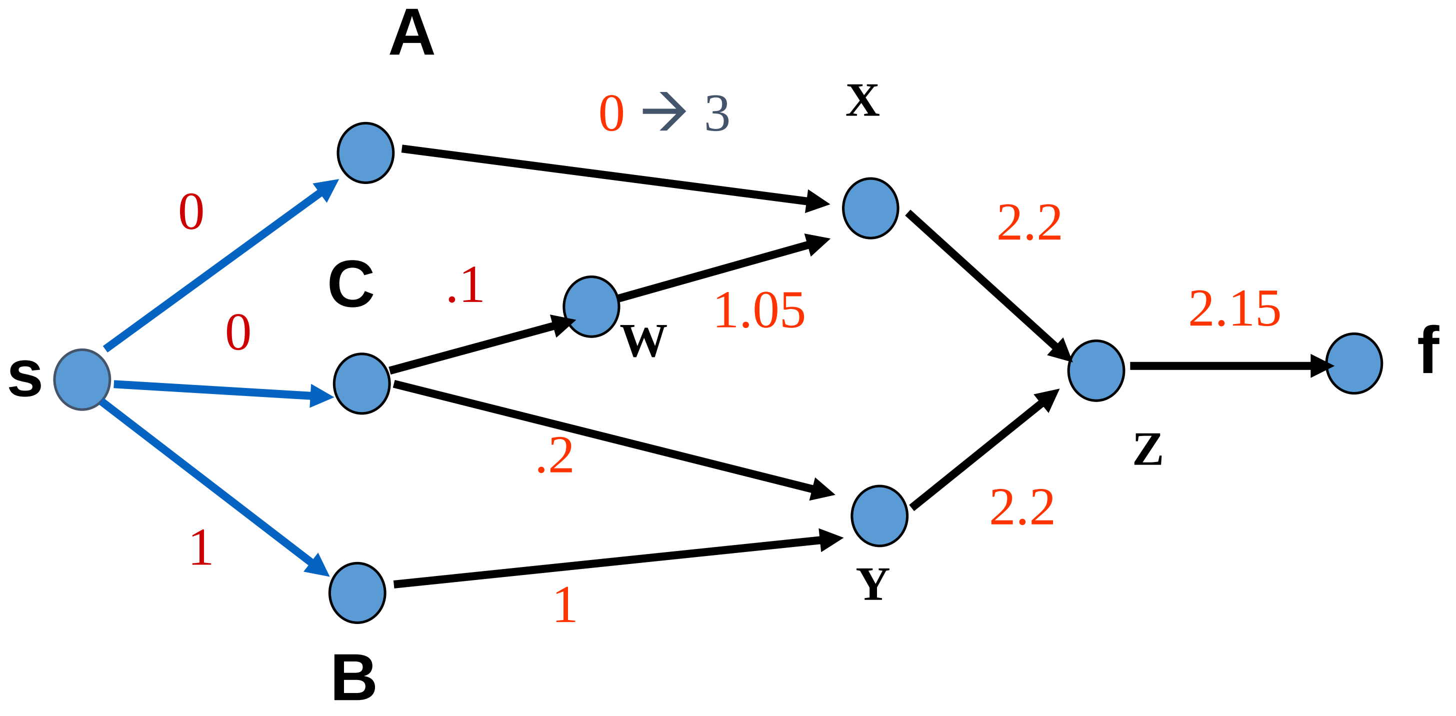
$O(m+n)$

Run the CPM on our example

Algorithm 2: Incremental longest path in a DAG

Suppose only a few weights/nodes/edges change.

How do we recompute the longest path efficiently?



Exercise: READ HANDOUT

Algorithm 3: Top k longest paths in a DAG

Often, we don’t want just the longest path

Want to find the *top* k *longest paths*

How to do this efficiently? (i.e., polynomial in n, m, k)

Key insight/idea:

- The 2nd longest path shares a prefix with the longest path.
- From each node along longest path, keep track of the “next longest” route to sink f.

Algorithm 3: Top k longest paths in a DAG

Pre-compute phase:

1. $\delta(v)$ = length of longest path from vertex v to sink f.

   How to compute? Complexity?

2. At each vertex v: order successor vertices $u_1, u_2, \ldots, u_k$ by decreasing $cost(u_i) = w(v, u_i) + \delta(u_i)$

3. Compute 'branch' slacks at v: $bs_i(v) = cost(u_i) - cost(u_{i+1})$

   $bs_k(v) = cost(u_k)$

Algorithm 3: Top k longest paths in a DAG

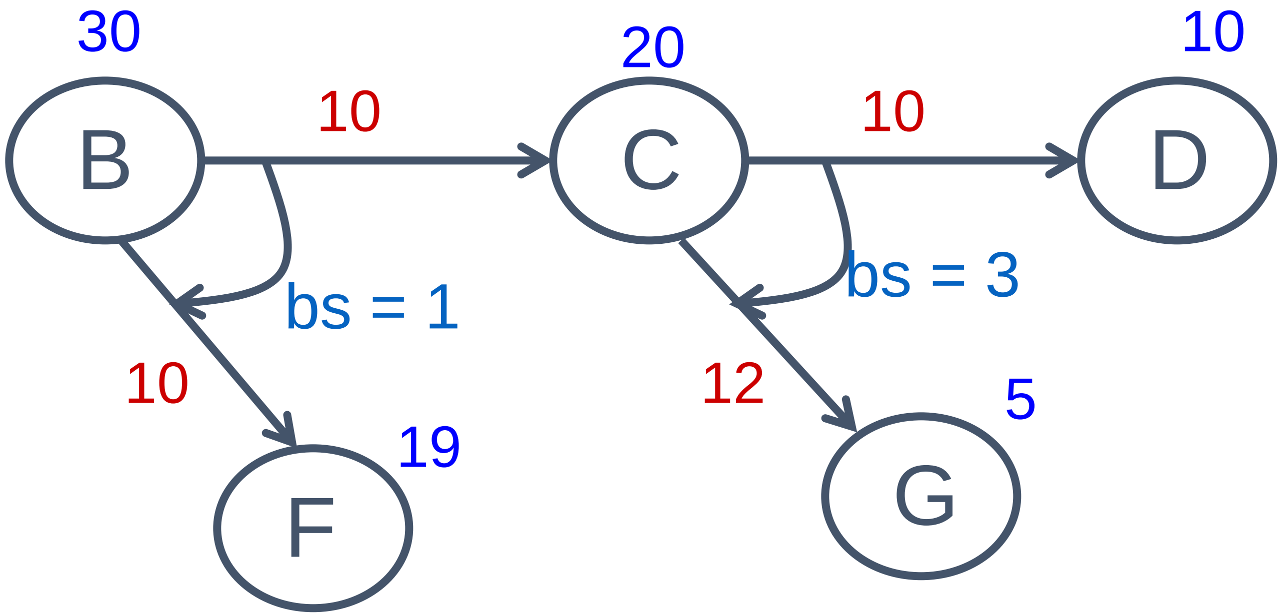
Pre-compute phase:

1. $\delta(v)$ = length of longest path from vertex v to sink f.
   How to compute? Complexity?
2. At each vertex v: order successor vertices $u_1, u_2, \ldots, u_k$ by decreasing $cost(u_i) = w(v, u_i) + \delta(u_i)$
3. Compute 'branch' slacks at v: $bs_i(v) = cost(u_i) - cost(u_{i+1})$
   $bs_k(v) = cost(u_k)$

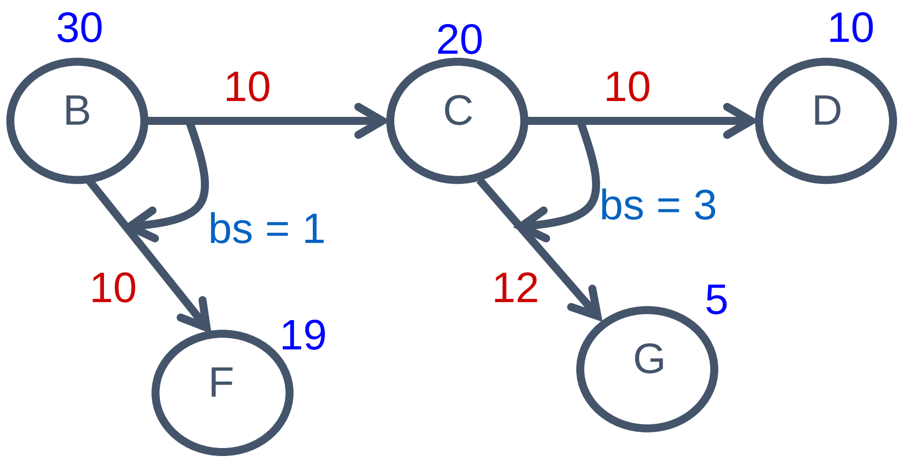
Main phase:

1. Let longest path $p = s, v_1, v_2, \ldots, v_r, f$
2. For 2nd (next) longest, order nodes according to branch slacks: $bs1(s), bs1(v_1), \ldots bs1(v_r)$, and pick the smallest. The corresponding successor indicates the next longest path.
3. For 3rd longest, add nodes along 2nd longest to the ordered node list, maintaining order. Go back to step 2 (check 'next' branch slack). (see handout for details)

Example: Top k longest paths in a DAG



Example: Top k longest paths in a DAG



Bibliography

- S. Devadas, K. Keutzer, S. Malik: “Computation of Floating Mode Delay in Combinational Circuits: Theory and Algorithms”, IEEE TCAD, December 1993.
- Sachin Sapatnekar , “Timing”, Chapter 5-”Timing Analysis for Combinational Circuits”, Springer-Verlag New York, Inc. 2004
- E. A. Lee and S. A. Seshia , Chapter 15 of “Introduction to Embedded Systems” , http://leeseshia.org