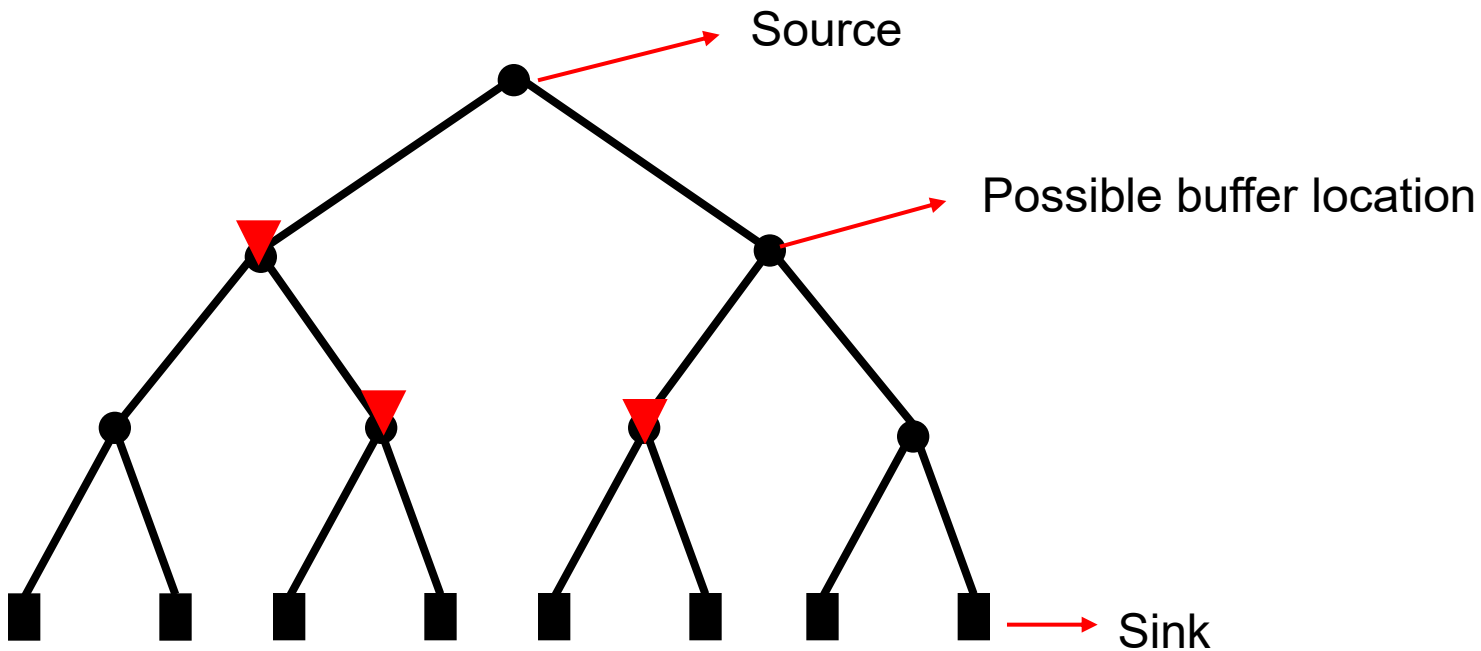
Buffered Tree Construction

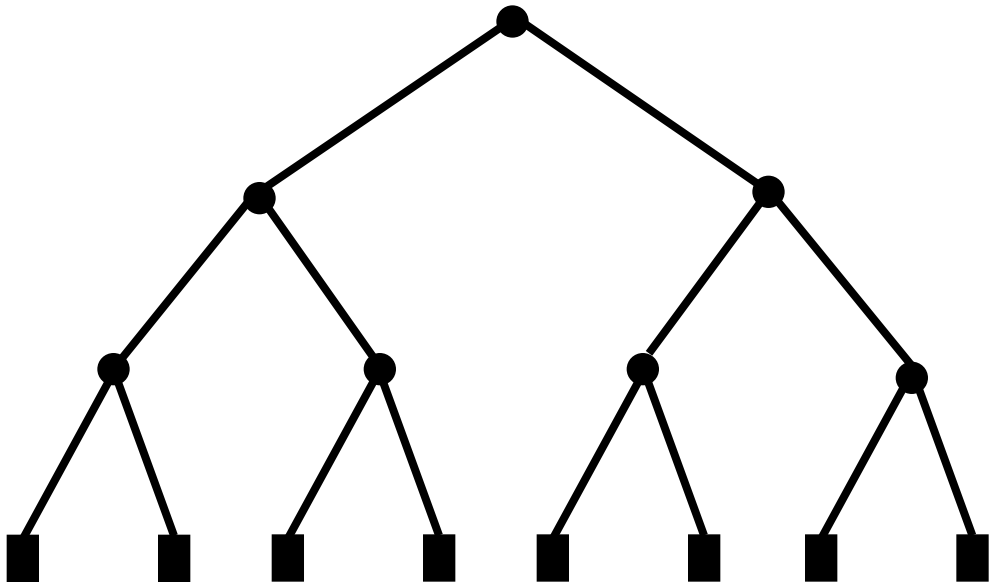
- Two-stage approach
  - Constructing a timing-driven tree
  - Considering buffer insertions at candidate locations



Routing Tree

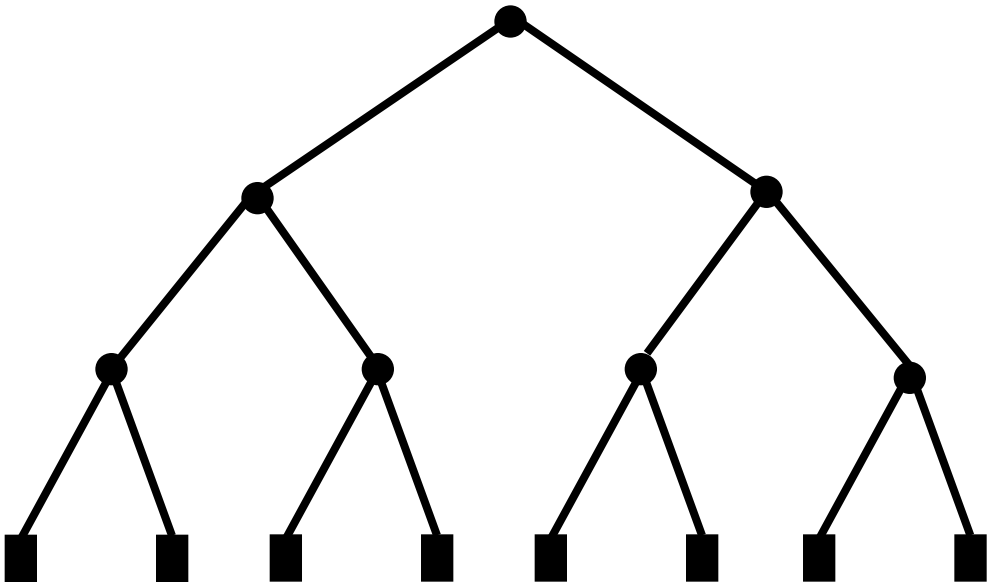
Classic Algorithm

Proposed by van Ginneken, ISCAS 1990



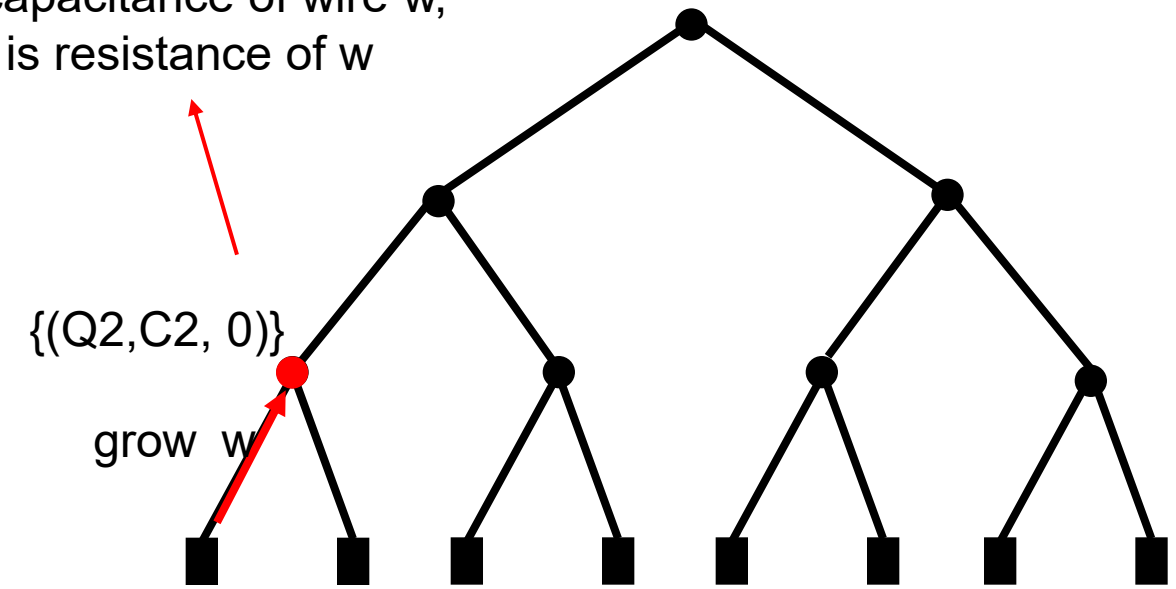
Routing Tree

Classic Algorithm

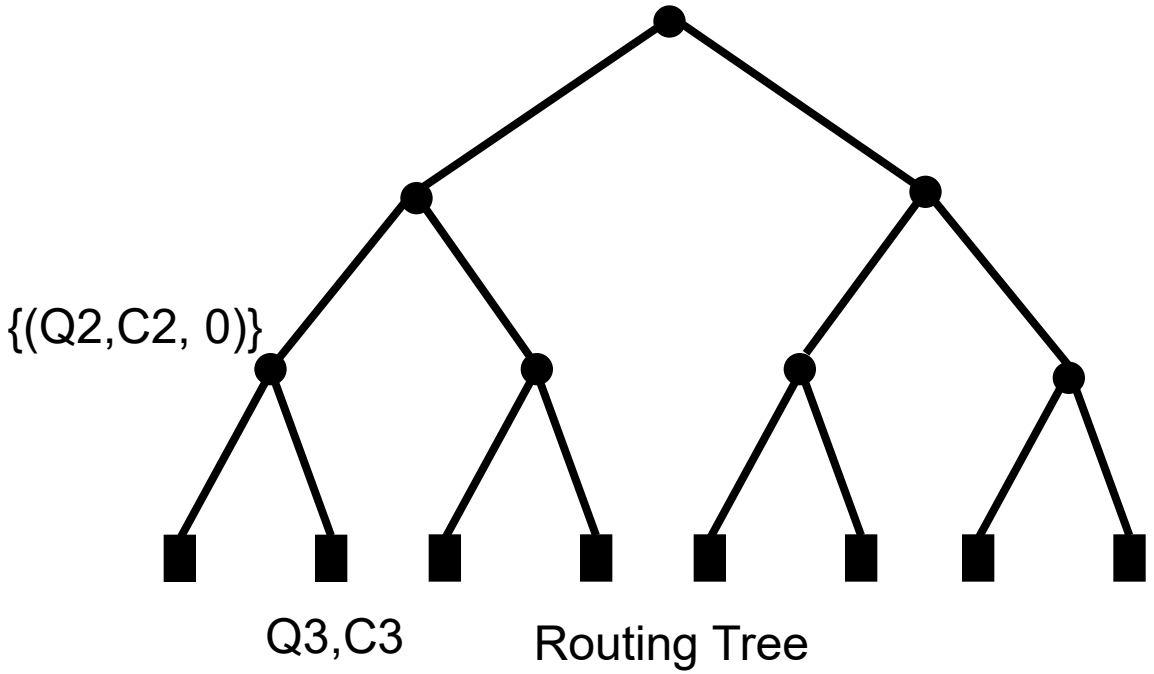


Classic Algorithm

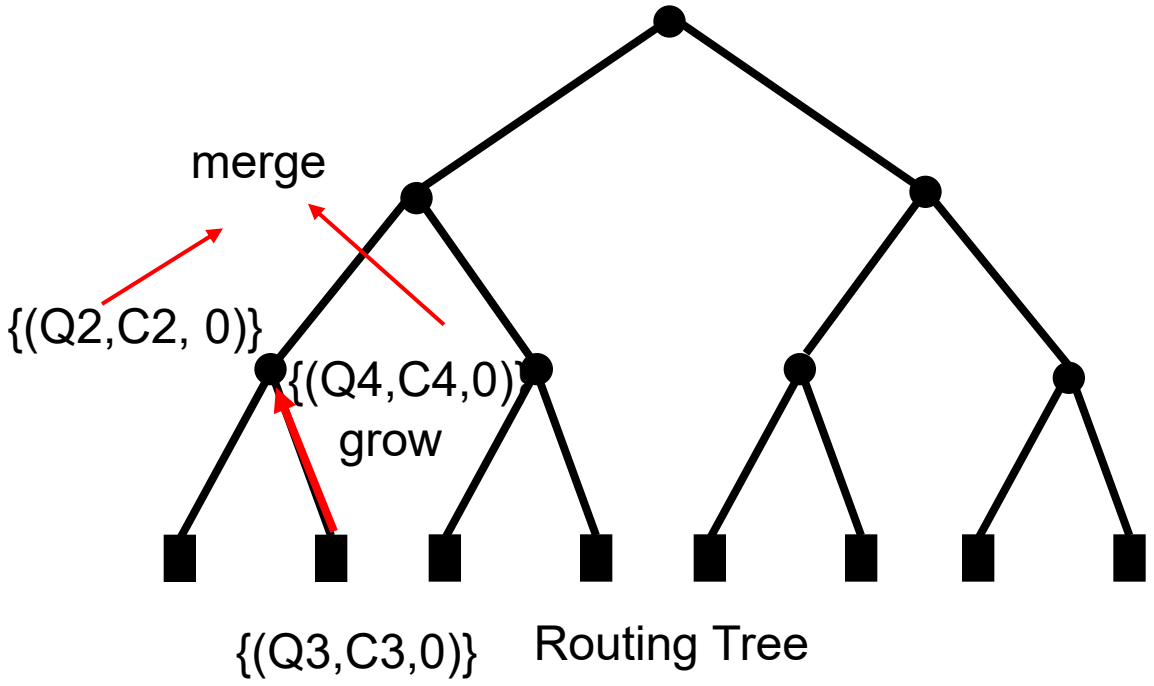
Q2= Q1- R (C/2 + C1)
C2= C+C1
C is capacitance of wire w,
R is resistance of w



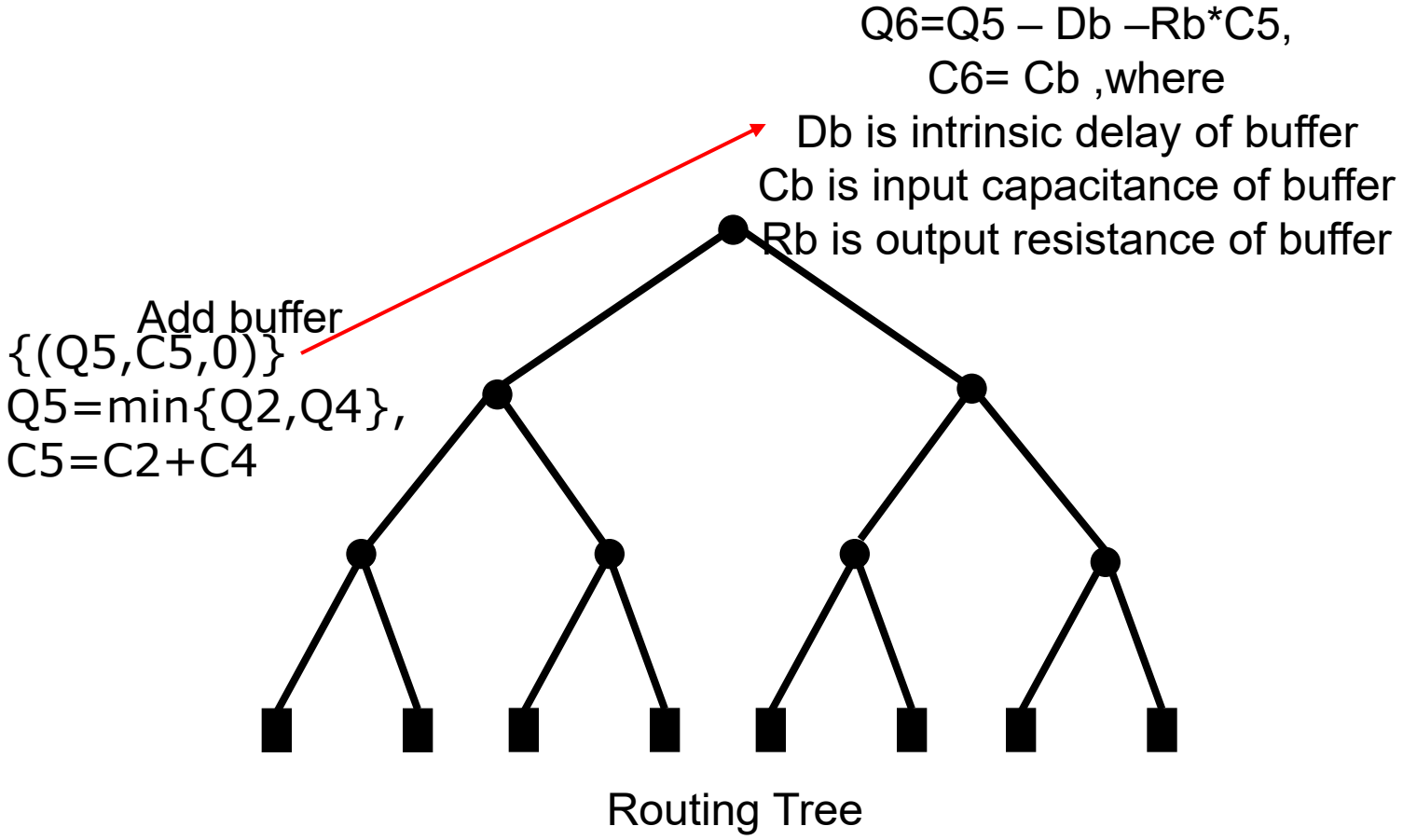
Classic Algorithm



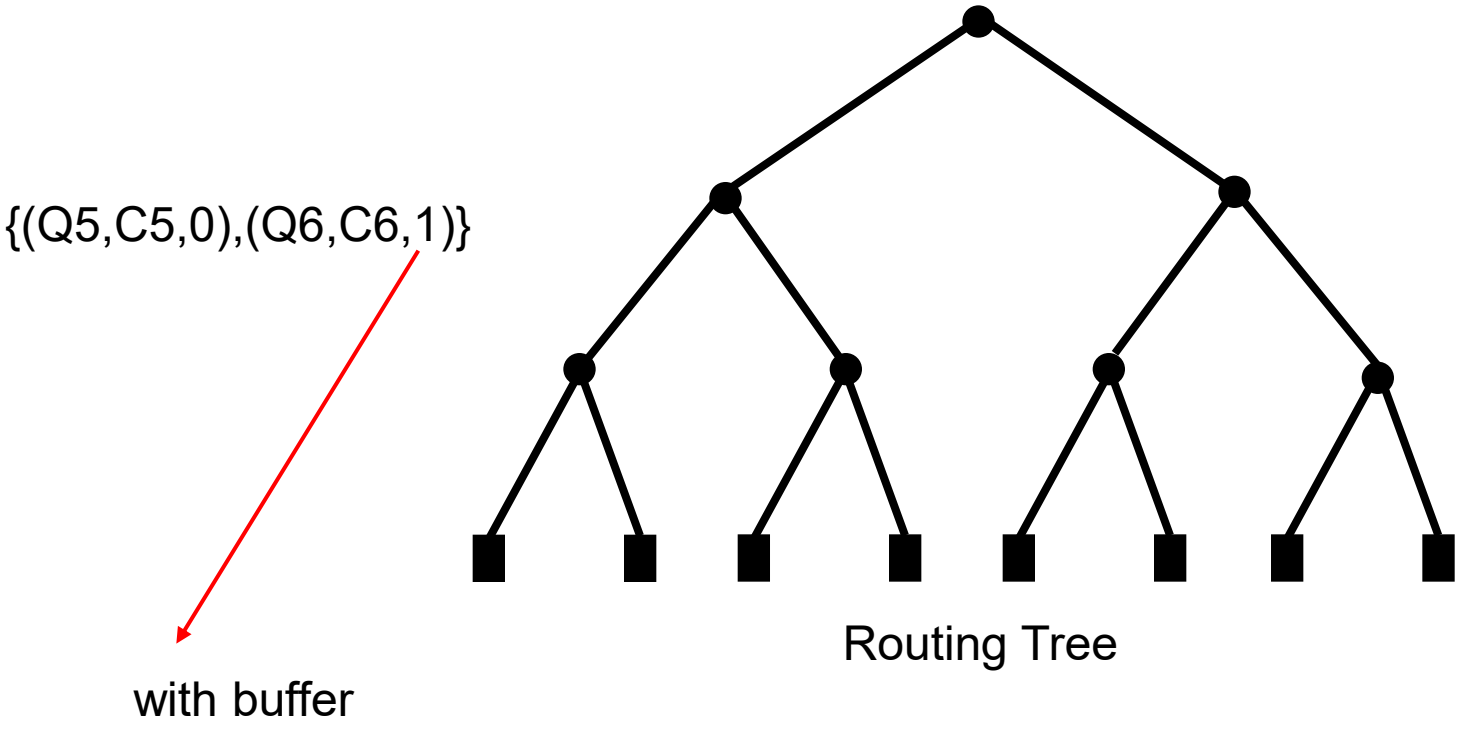
Classic Algorithm



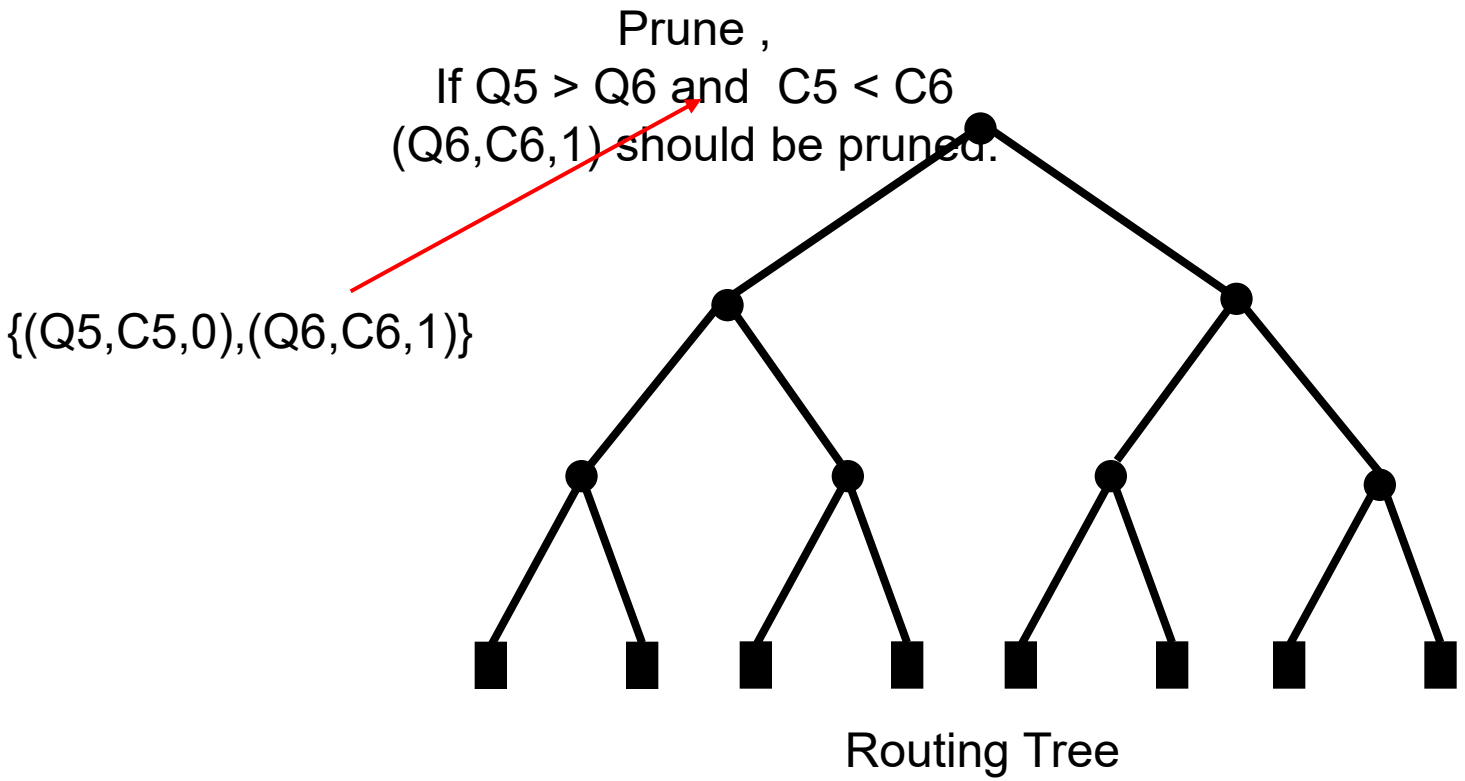
Classic Algorithm



Classic Algorithm



Classic Algorithm

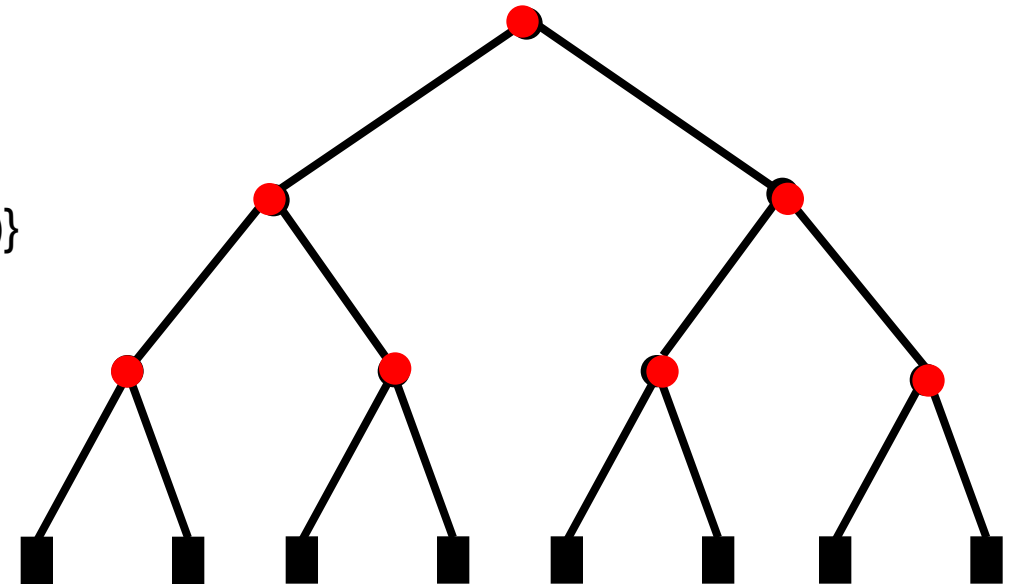


Classic Algorithm

$\{(Q_{r1},C_{r1},B_{r1}),\ldots,(Q_{rm},C_{rm},B_{rm})\}$.

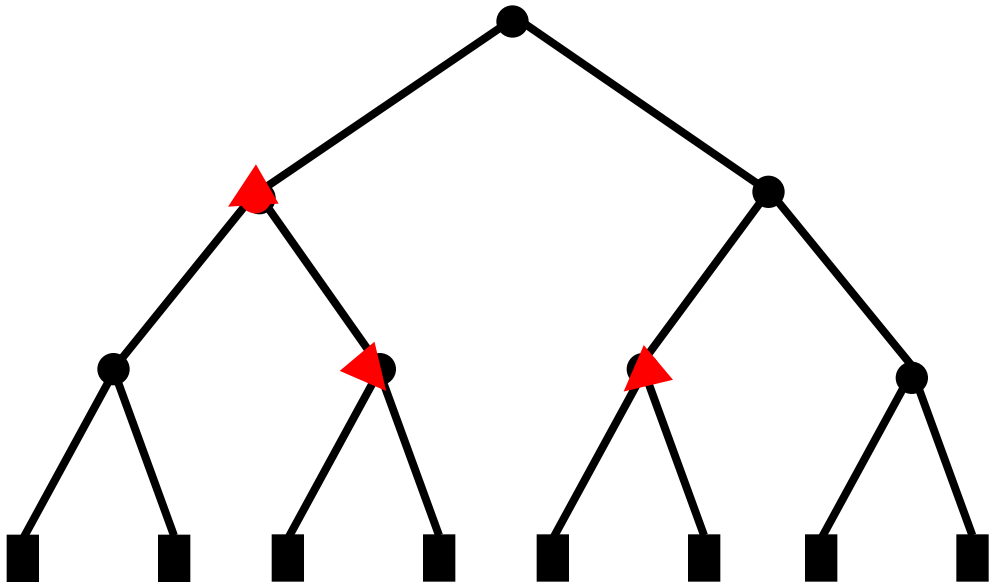
One with max required time will be picked up

{(Q5,C5,0),(Q6,C6,1)}



Routing Tree

Classic Algorithm



Buffered routing tree with maximal required time

Extensions of Classic Algorithm

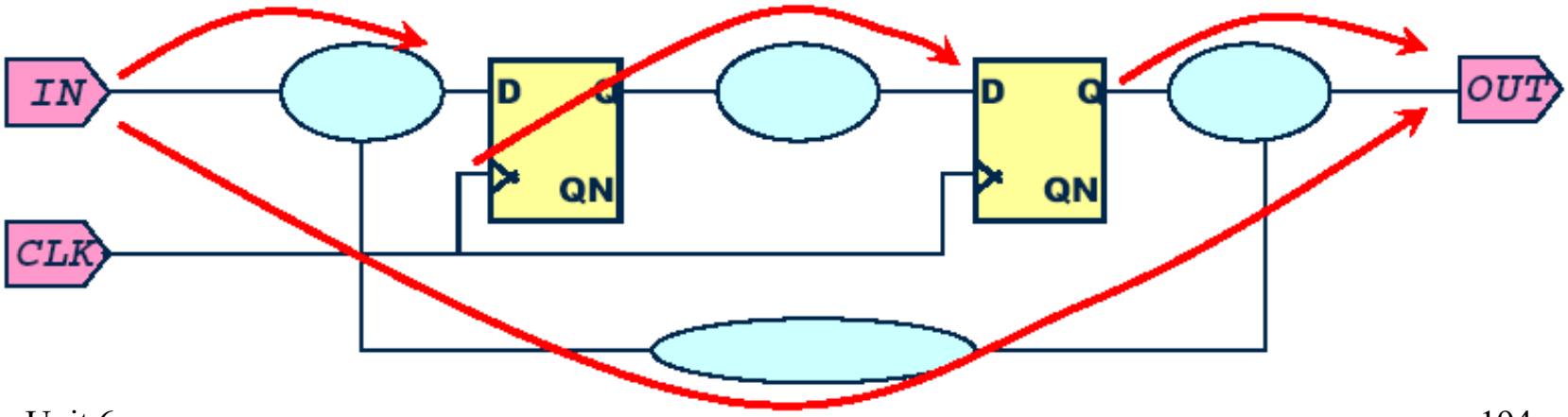
- Multiple buffer types
- Inverters
- Polarity constraints
- Controlling buffer resources
- Capacitance constraints
- Wire sizing
- Blockage recognition

Dynamic Timing Analysis

- So-called Simulation
  - Input vectors are applied during the simulation time
  - Simulator calculates the logic value and delay
- Disadvantages of DTA
  - Virtually impossible to do exhaustive analysis
    - Vector creation takes too long
    - Incomplete timing coverage
  - Hard to discern the cause of failure because the function and timing are analyzed at the same time
  - Requires more memory and CPU resources over STA
    - Long simulation time
    - Capacity limited

Static Timing Analysis

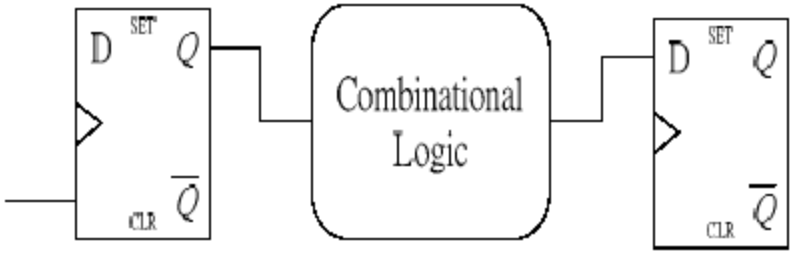
- A method for determining if a circuit meets timing constraints without having to simulate clock cycles
- Three main steps
  - Break the design into sets of timing paths
  - Calculate the delay of each path (create timing graph)
  - Check all path delays to see if the given timing constraints are met



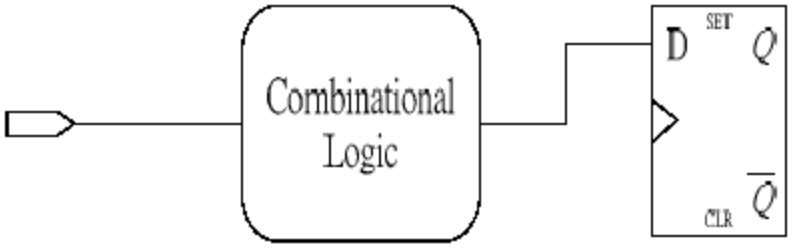
Advantages of STA

- Exhaustive timing coverage
- Does not require input vectors
- More efficient than DTA in memory and CPU resources
  - Faster operation
  - Capacity for millions of gates

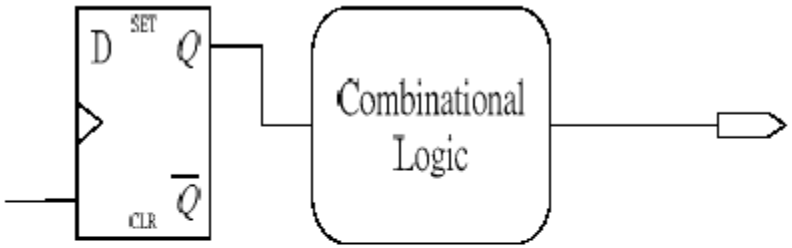
Four Types of Timing Paths



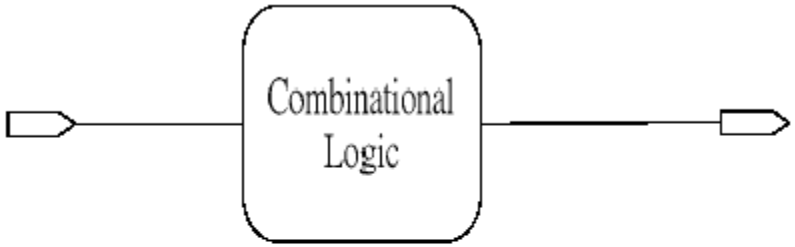
Start Point:
clock pin of sequential device
End Point:
data input pin of sequential devices



Start Point:
primary input port
End Point:
data input pin of sequential devices



Start Point:
clock pin of sequential device
End Point:
primary output port

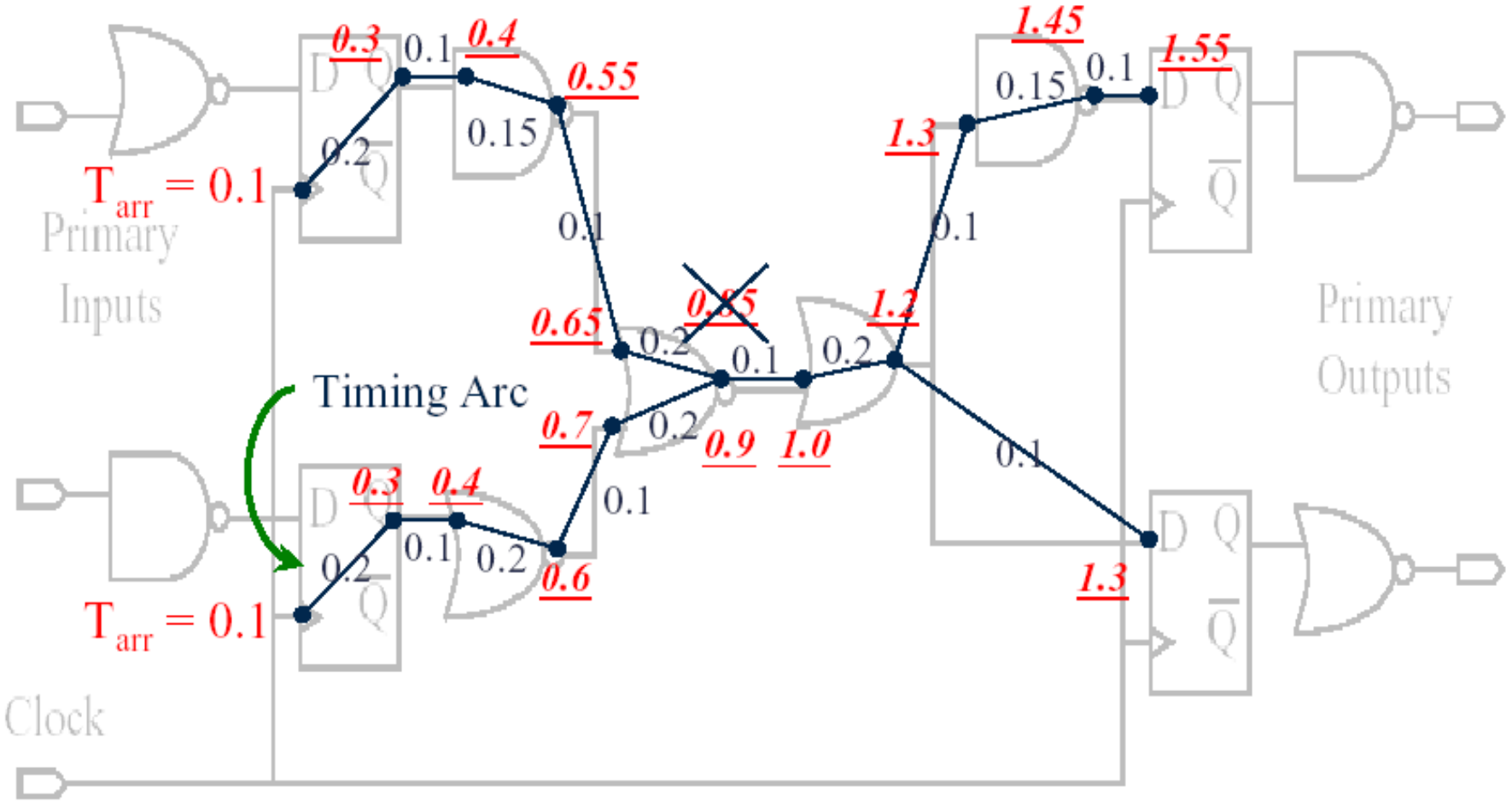


Start Point:
primary input port
End Point:
primary output port

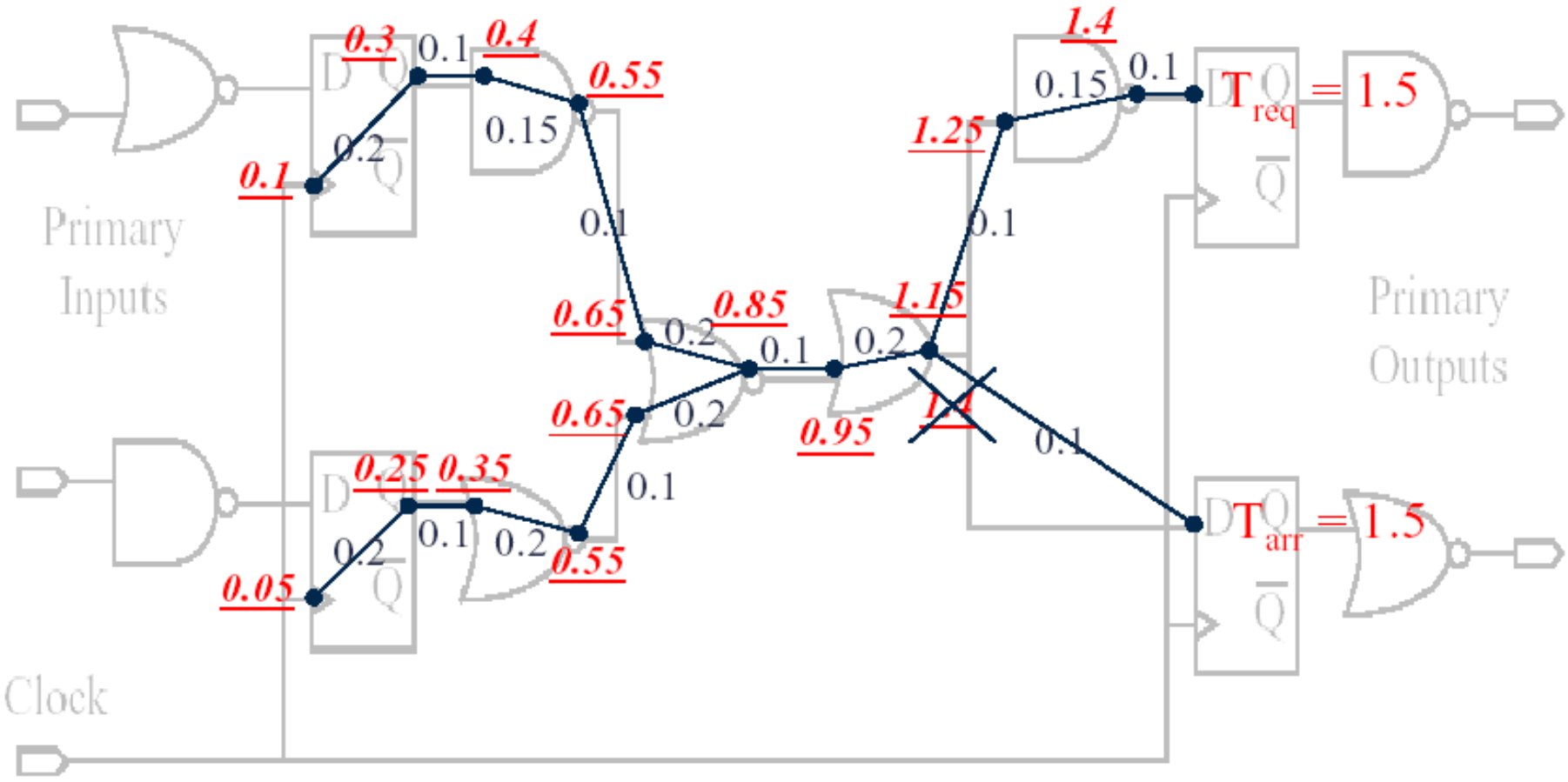
Create Timing Graph

- Netlist is represented as directed acyclic graph (DAG)
- Delay values associated with links (cells & nets) are calculated
- Create the timing graph of arrival time (AT)
- Create the timing graph of required time (RT)
- Create the slack graph
  - Timing is met when slack is greater than or equal to zero (RT should always be greater than or equal to AT)

Timing Graph – Arrival Time

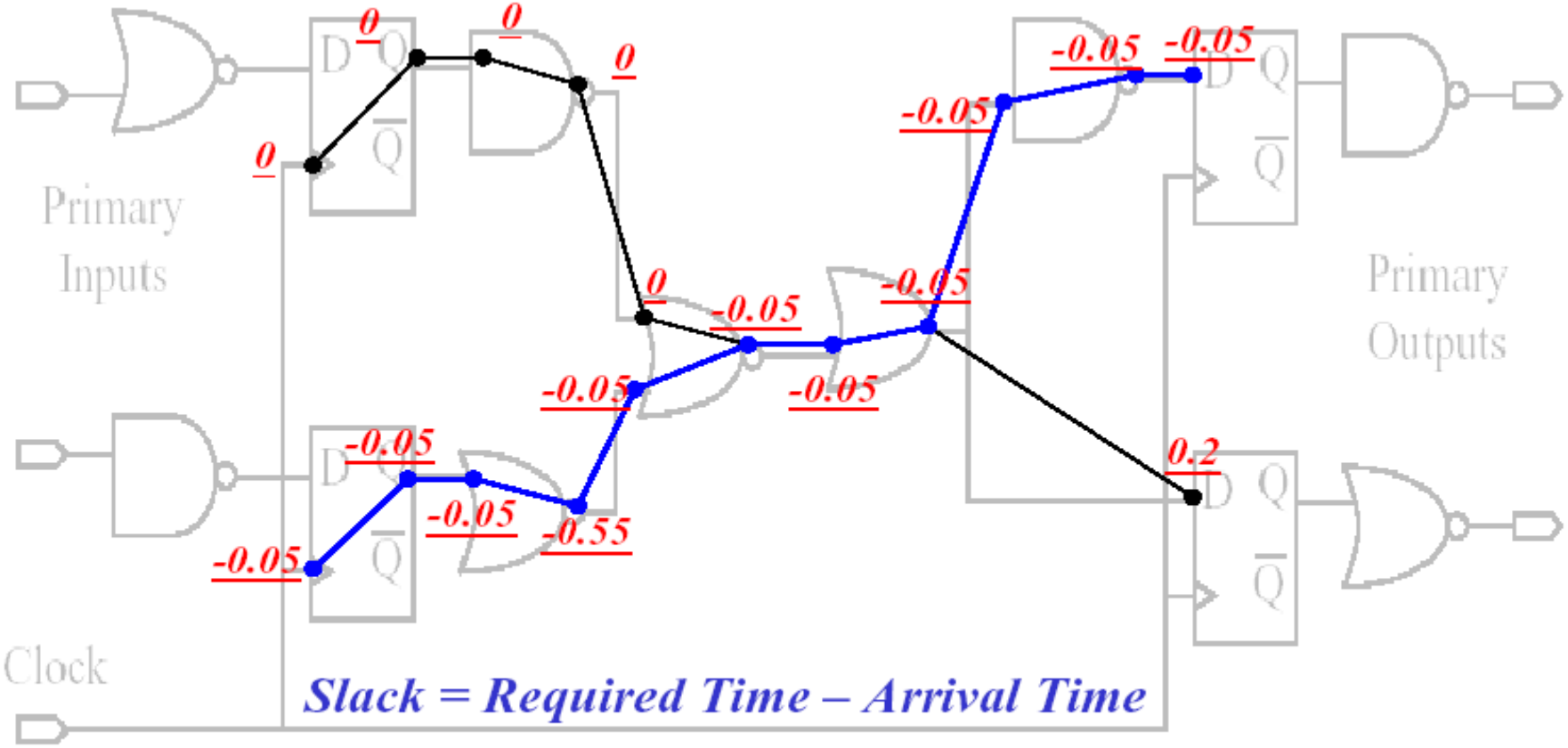


Timing Graph – Required Time



setup time case

Slack Graph

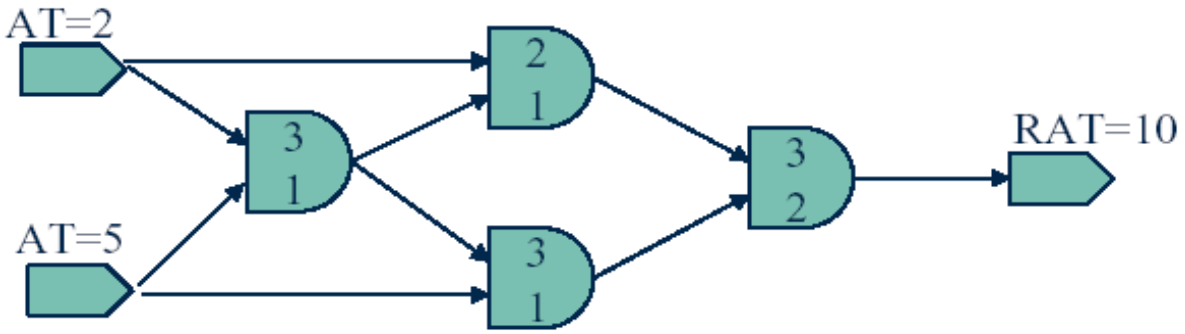


setup time case

Block-based vs. Path-based STA (1/2)

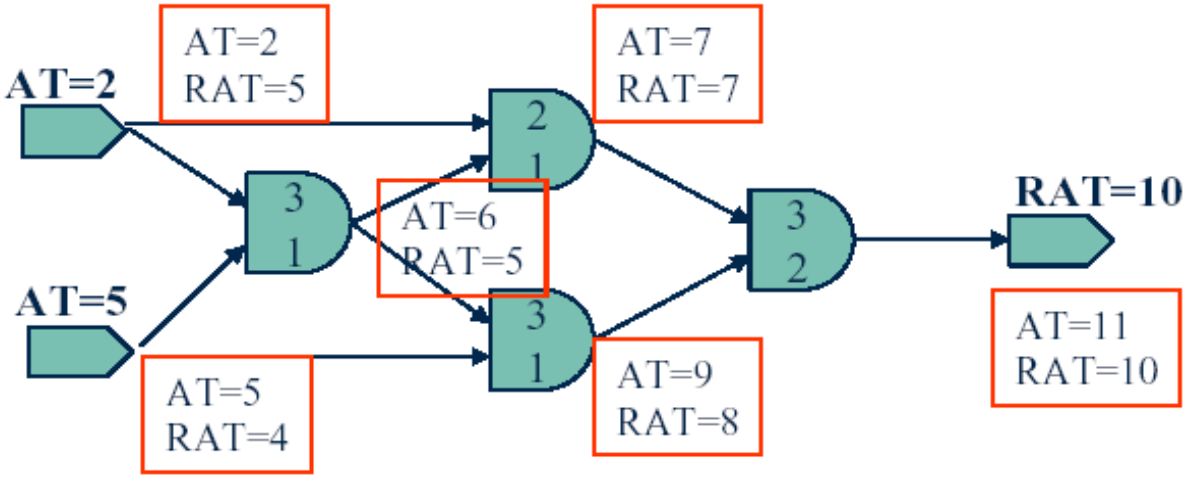
- Block-based: timing information is associated with discrete design elements (ports, pins, gates)
  - Slack is calculated on every design element
- Path-based: timing information is associated with topological paths (collections of design elements)

Block-based vs. Path-based STA (2/2)



Path-based:

2+2+3 = 7 (OK)
2+3+1+3 = 9 (OK)
2+3+3+2 = 10 (OK)
5+1+1+3 = 10 (OK)
5+1+3+2 = 11 (Fail)
5+1+2 = 8 (OK)



Block-based:

Critical path is determined as collection of gates with the same, negative slack: In our case, we see one critical path with slack = -1