Application of Specific Delay Window Routing for Timing Optimization in FPGA Designs

Evan Wegley, Qinhai Zhang
Lattice Semiconductor Corporation

23rd ACM / SIGDA International Symposium on FPGAs

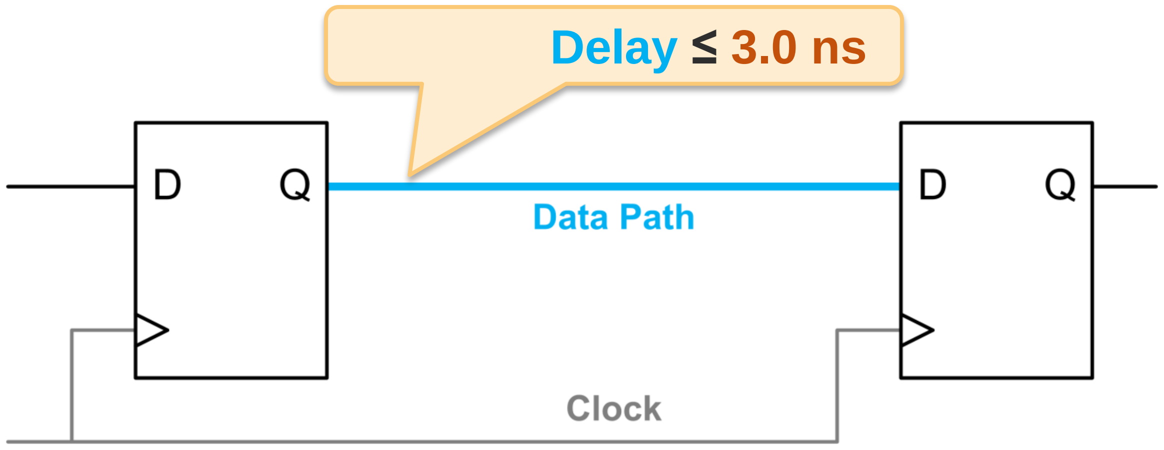
FPGA Routing

- FPGA routers must optimize for competing goals
  - Routability
  - Timing performance

- Timing constraints come in many forms
  - Setup timing
  - Hold timing
  - Other timing constraints
    - Maximum skew, for example

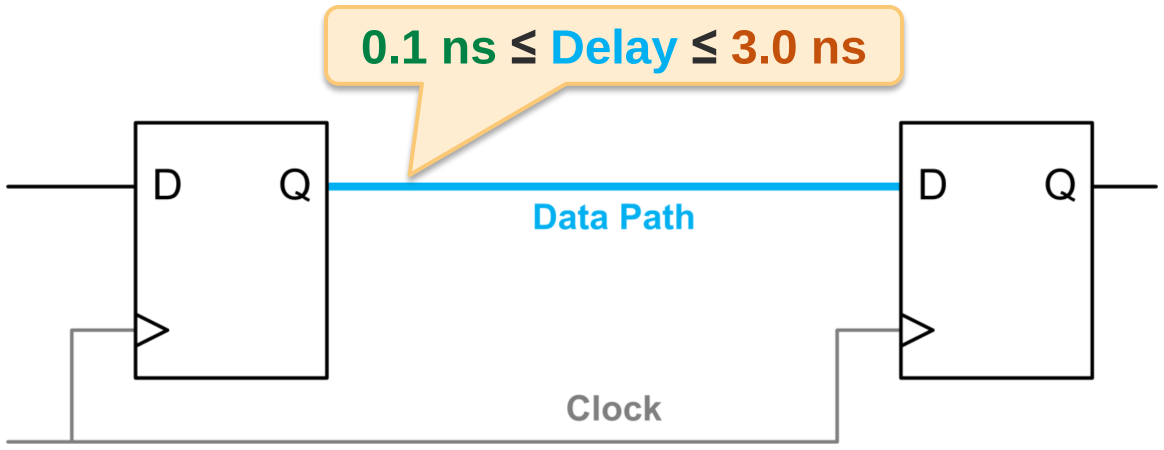
Setup Timing

- Constrains data paths between registers to arrive before the next clock cycle starts
  - Produces an upper bound on delay of the data path



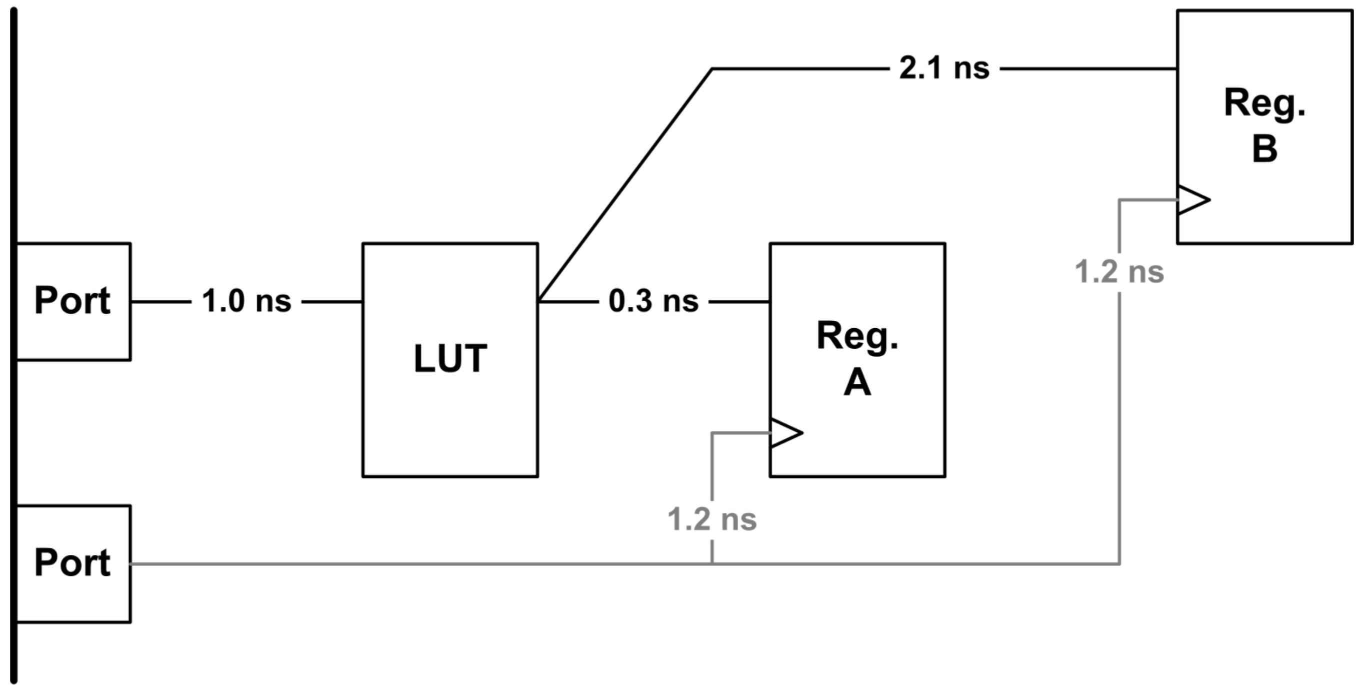
Hold Timing

- Constrains data paths between registers to arrive after the previous cycle has been captured
  - Produces a lower bound on delay of the data path
  - At routing phase, we can correct hold timing violations by rerouting the data path to have additional delay



More Complex Hold Timing

- Example: constrain the data paths to 2.0 ns for maximum setup time and 0.4 ns for minimum hold time with respect to clock

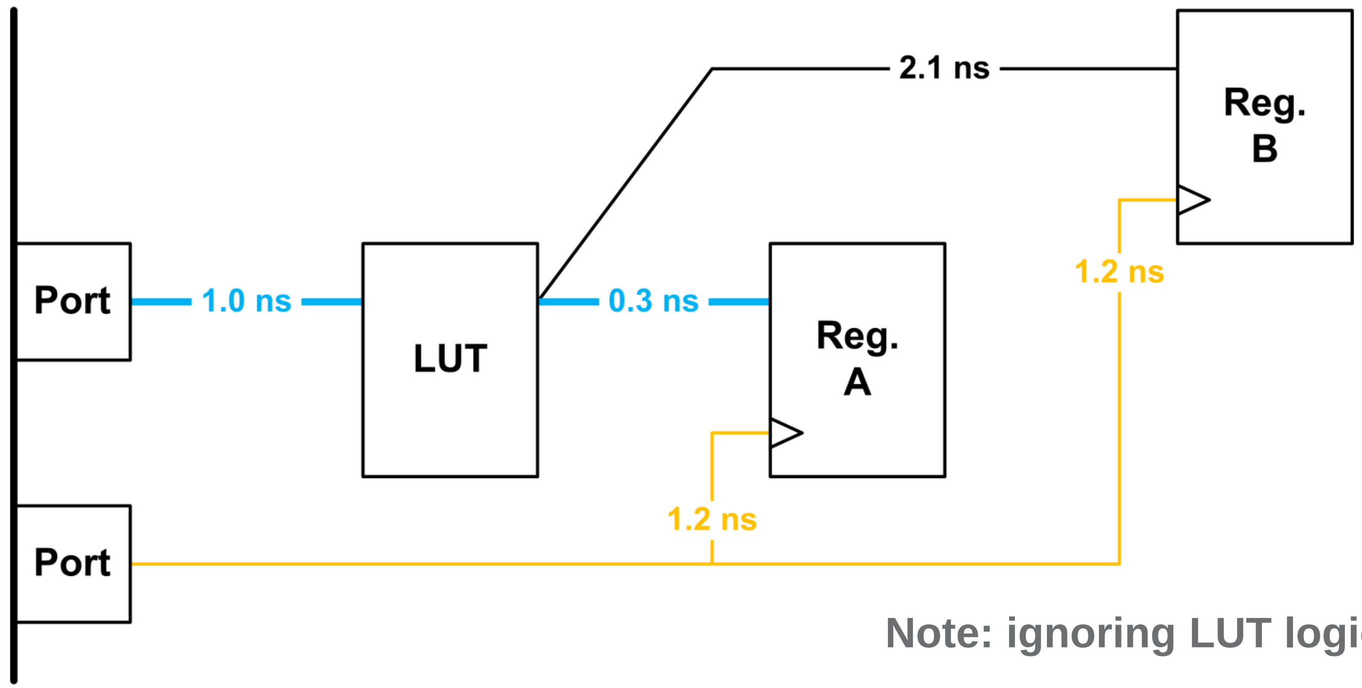


More Complex Hold Timing

- Example: constrain the data paths to 2.0 ns for maximum setup time and 0.4 ns for minimum hold time with respect to clock

$\text{Slack}_{A,\ \text{Hold}} = (1.0 + 0.3 - 1.2) - 0.4$
$= -0.3$ ns

Hold violation!



Note: ignoring LUT logic delays

More Complex Hold Timing

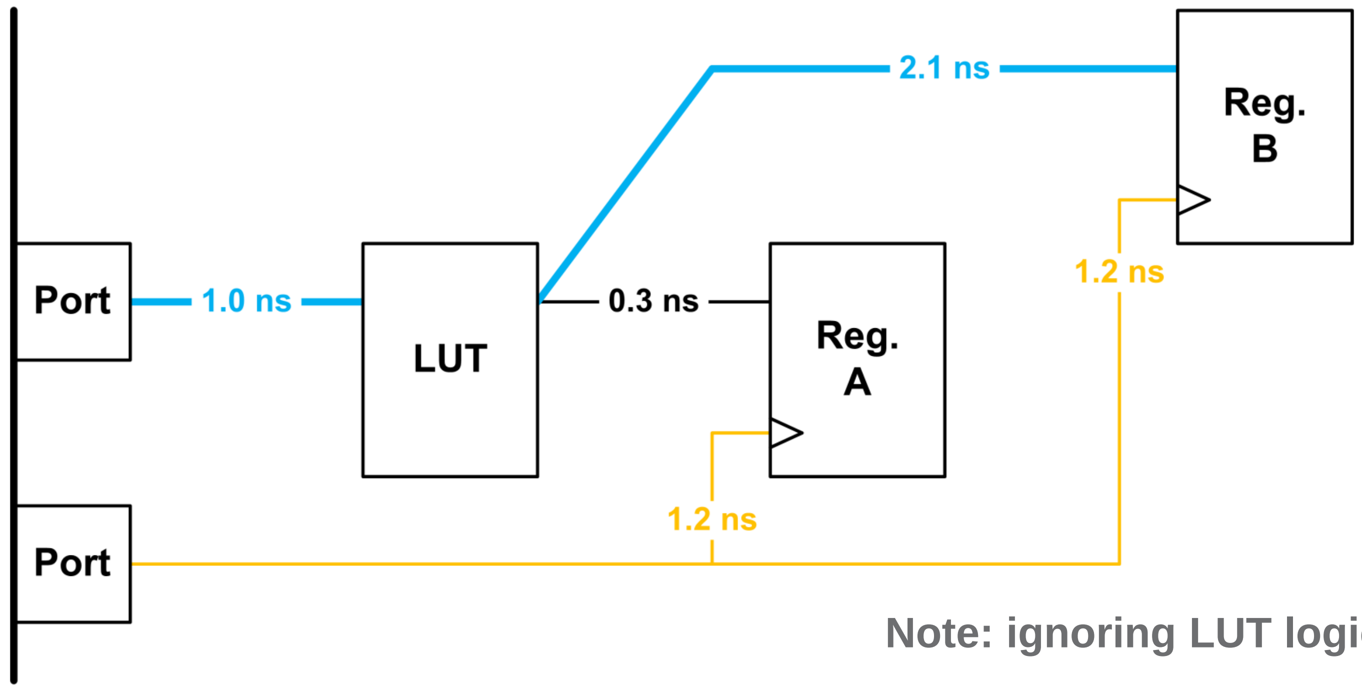
- Example: constrain the data paths to 2.0 ns for maximum setup time and 0.4 ns for minimum hold time with respect to clock

$\text{Slack}_{A,\ \text{Hold}} = (1.0 + 0.3 - 1.2) - 0.4$
$= -0.3$ ns

Hold violation!

$\text{Slack}_{B,\ \text{Setup}} = 2.0 - (1.0 + 2.1 - 1.2)$
$= 0.1$ ns

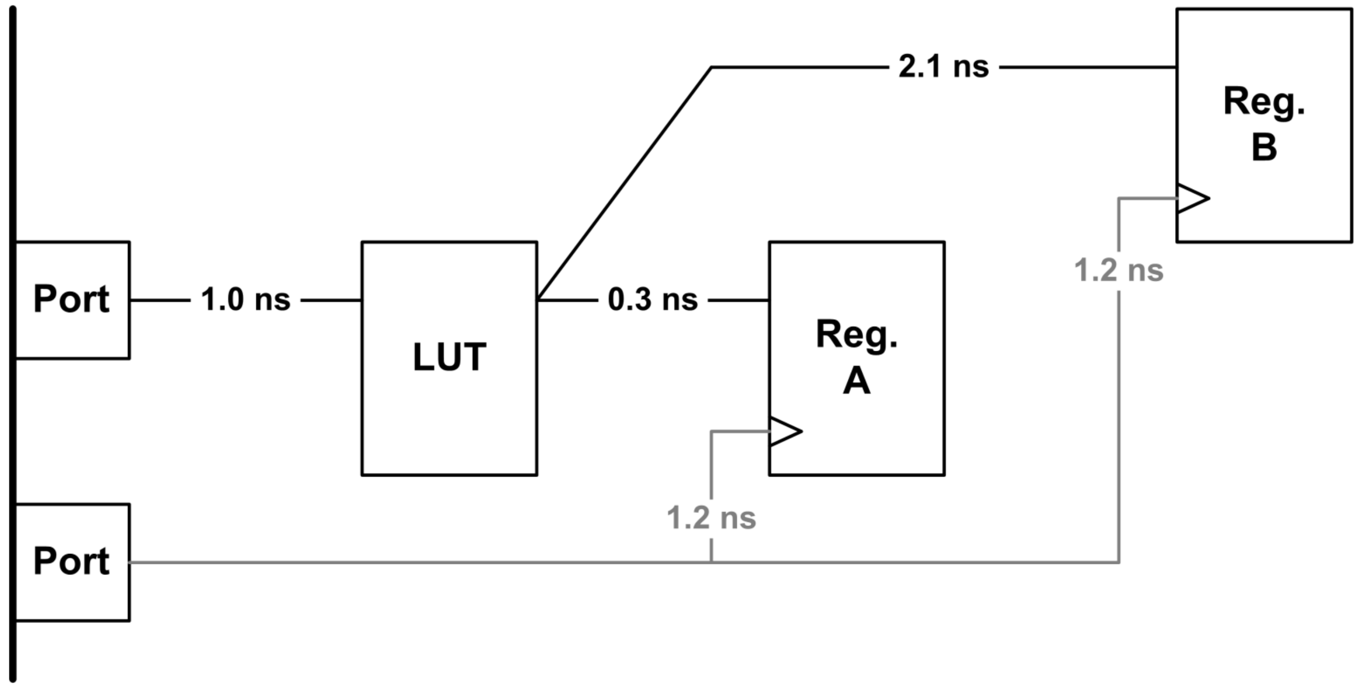
Close to setup threshold



Note: ignoring LUT logic delays

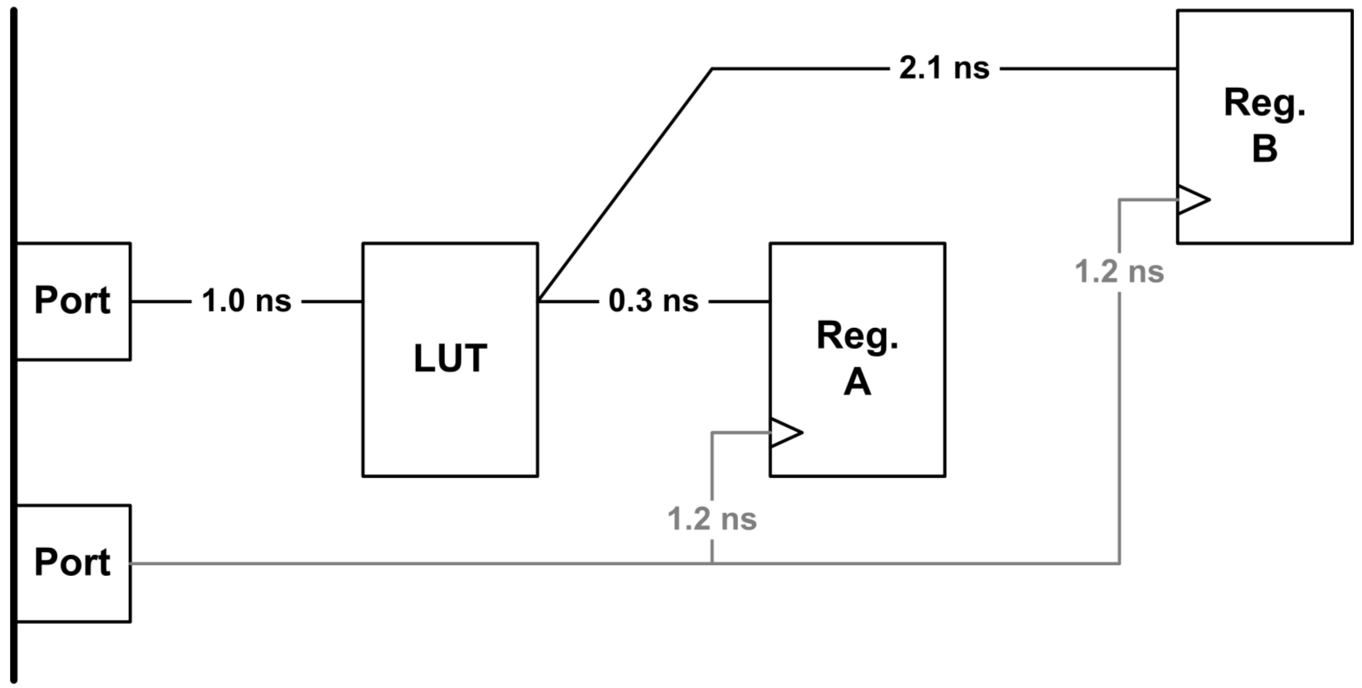
More Complex Hold Timing

- Need awareness of both setup and hold requirements for each connection
- We can do so by setting lower and upper bounds on delay for each connection: slack allocation



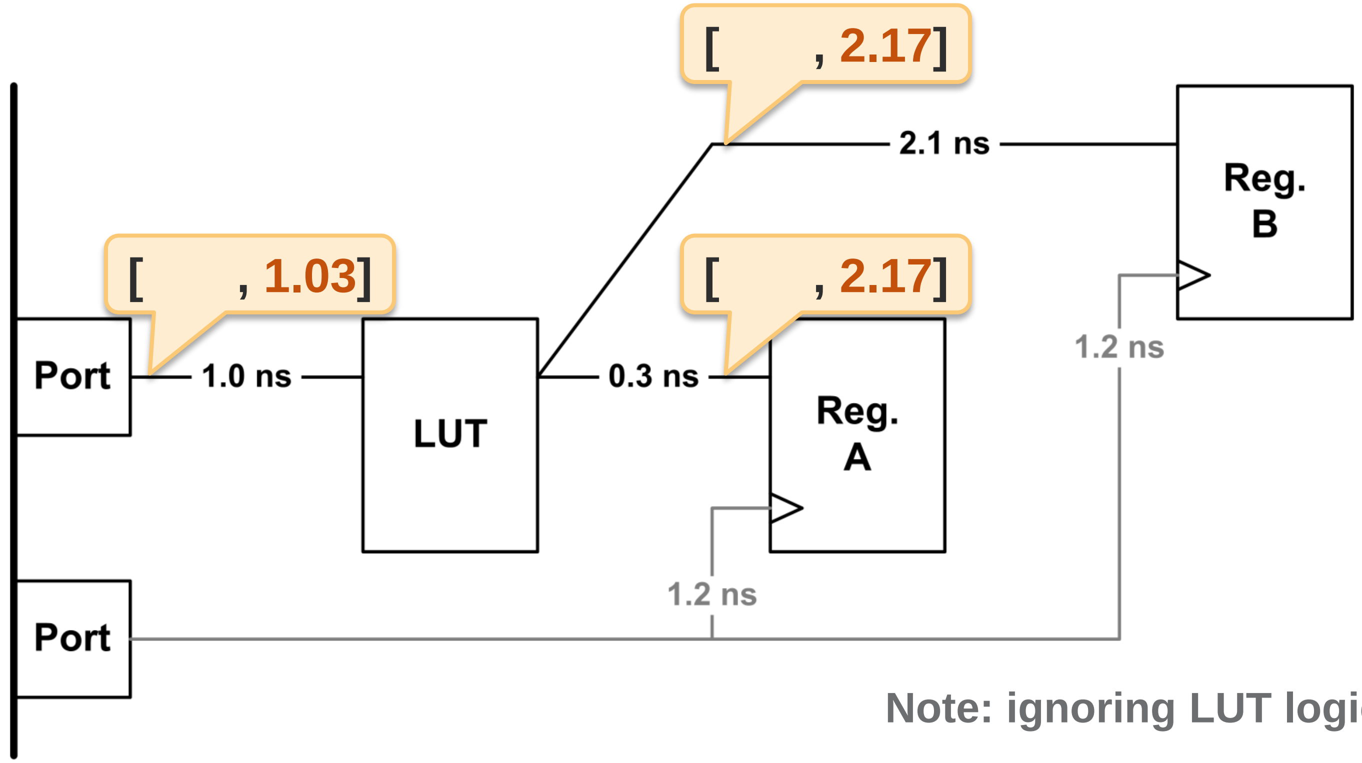
Slack Allocation

- **Process of distributing slack to all connections of a circuit**
  (Youssef 1990, Frankle 1992, Fung 2008)



Slack Allocation

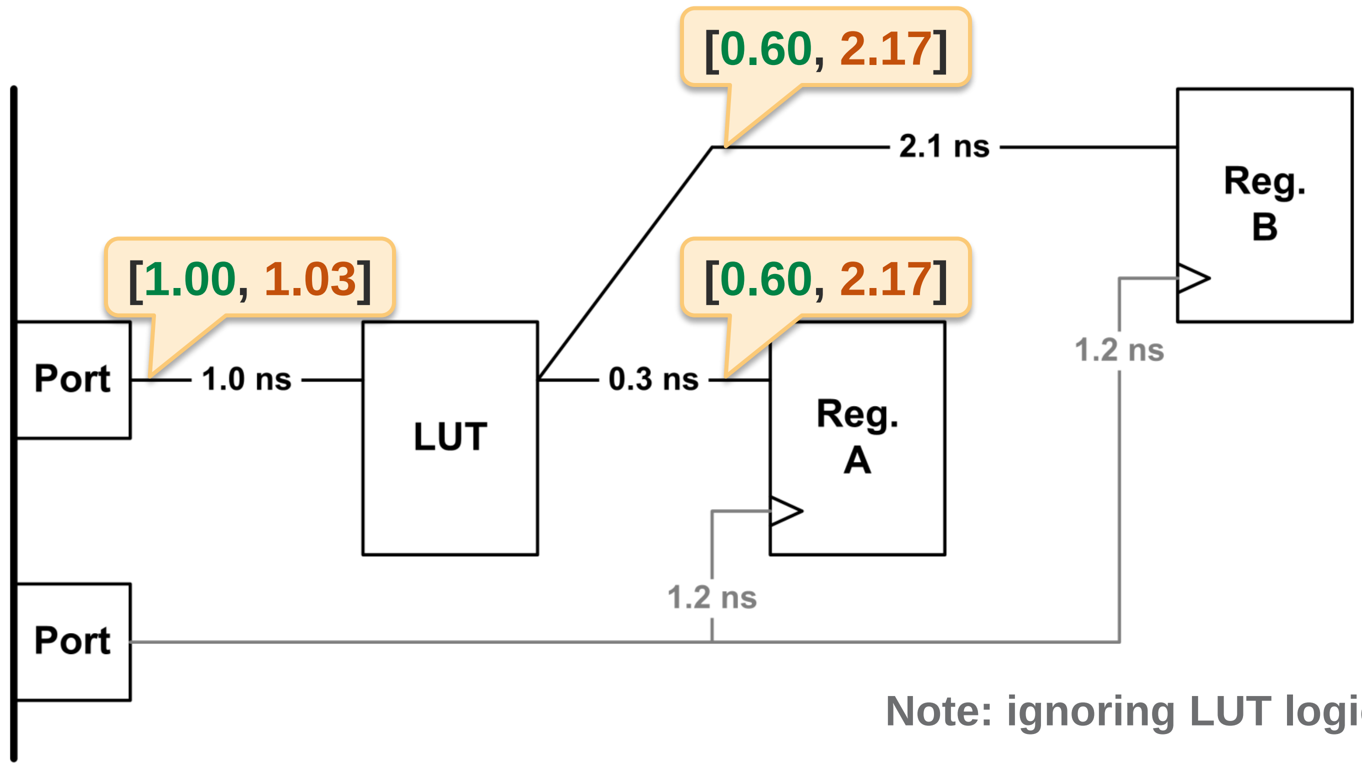
- Process of distributing slack to all connections of a circuit (Youssef 1990, Frankle 1992, Fung 2008)
  - Allocate slack on setup timing to get upper bound



Note: ignoring LUT logic delays

Slack Allocation

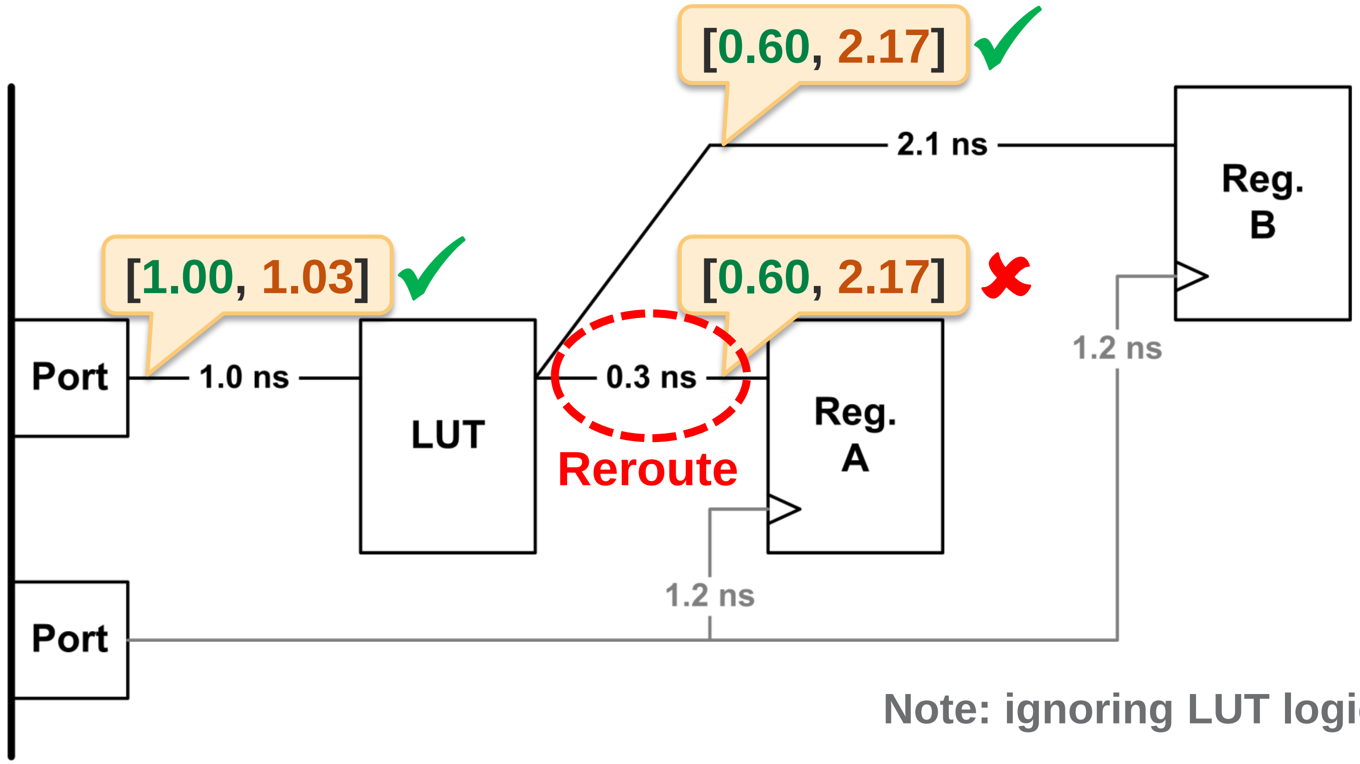
- Process of distributing slack to all connections of a circuit (Youssef 1990, Frankle 1992, Fung 2008)
  - Allocate slack on setup timing to get upper bound
  - Allocate slack on hold timing to get lower bound



Note: ignoring LUT logic delays

Slack Allocation

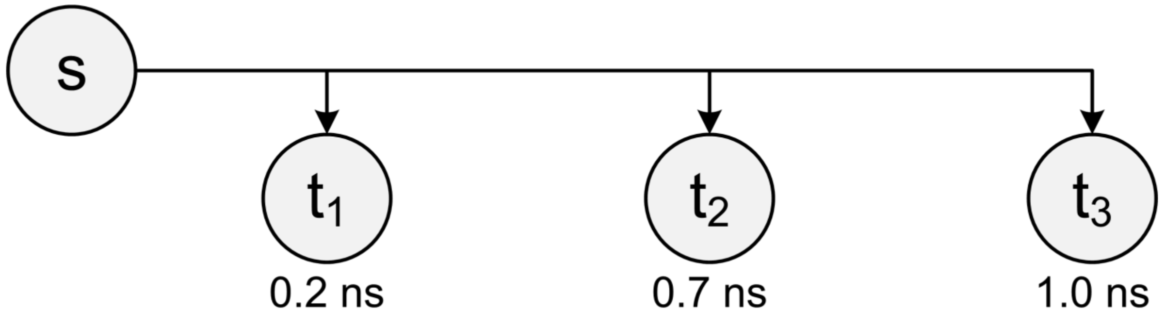
- Process of distributing slack to all connections of a circuit (Youssef 1990, Frankle 1992, Fung 2008)
  - Allocate slack on setup timing to get upper bound
  - Allocate slack on hold timing to get lower bound



Note: ignoring LUT logic delays

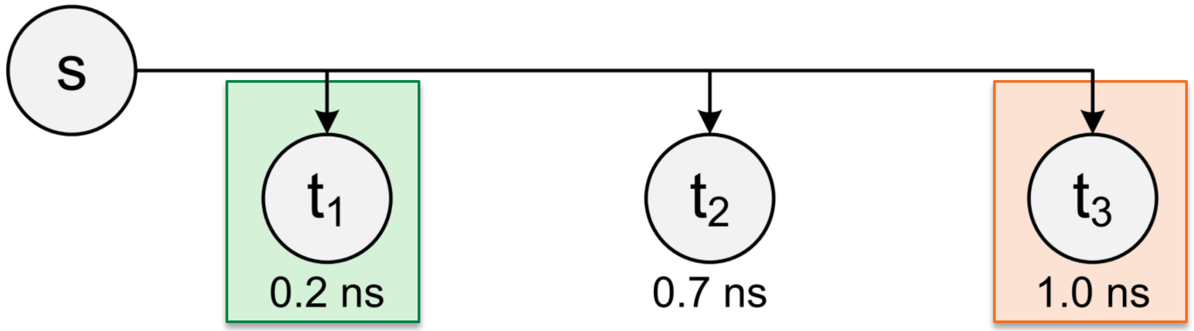
Maximum Skew

- Constrains the range of delays on the loads of some net(s) or buses
- Example: constrain the net to have a maximum skew of 0.5 ns



Maximum Skew

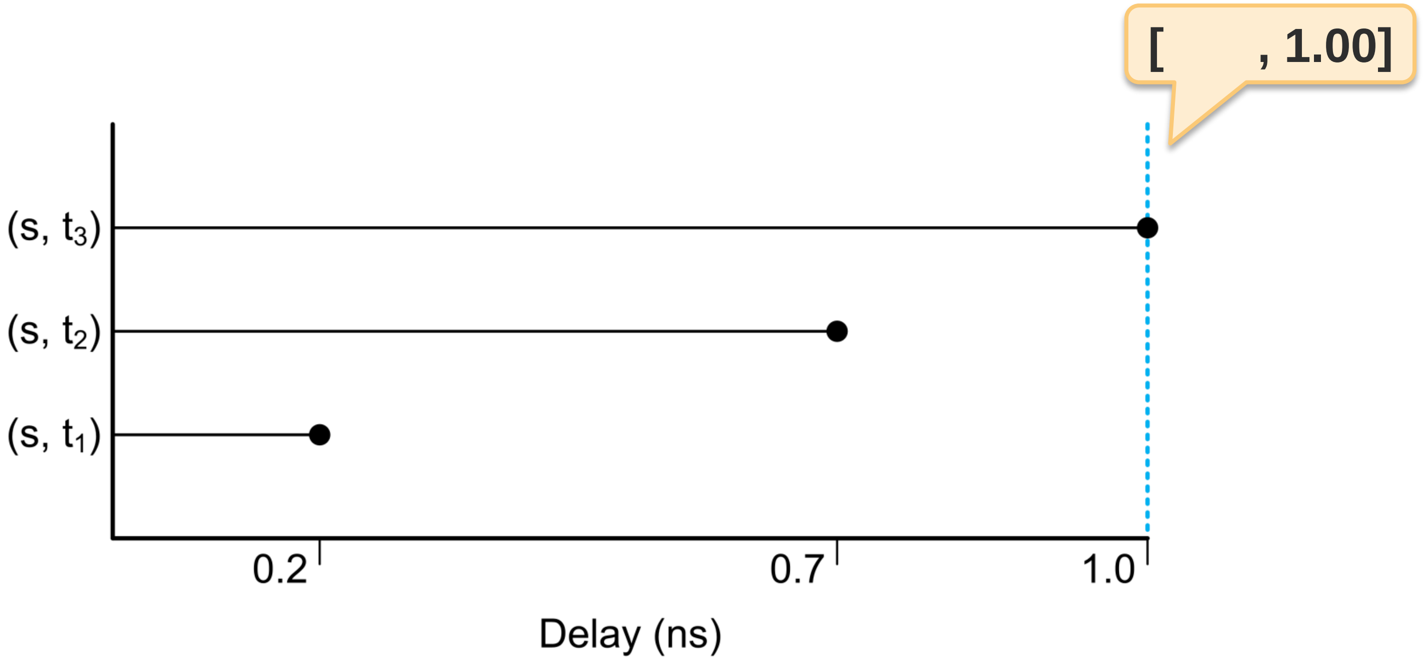
- Constrains the range of delays on the loads of some net(s) or buses
- Example: constrain the net to have a maximum skew of 0.5 ns



- Initial skew is 1.0 – 0.2 = 0.8 ns
- Violates constraint of 0.5 ns maximum skew by 0.3 ns

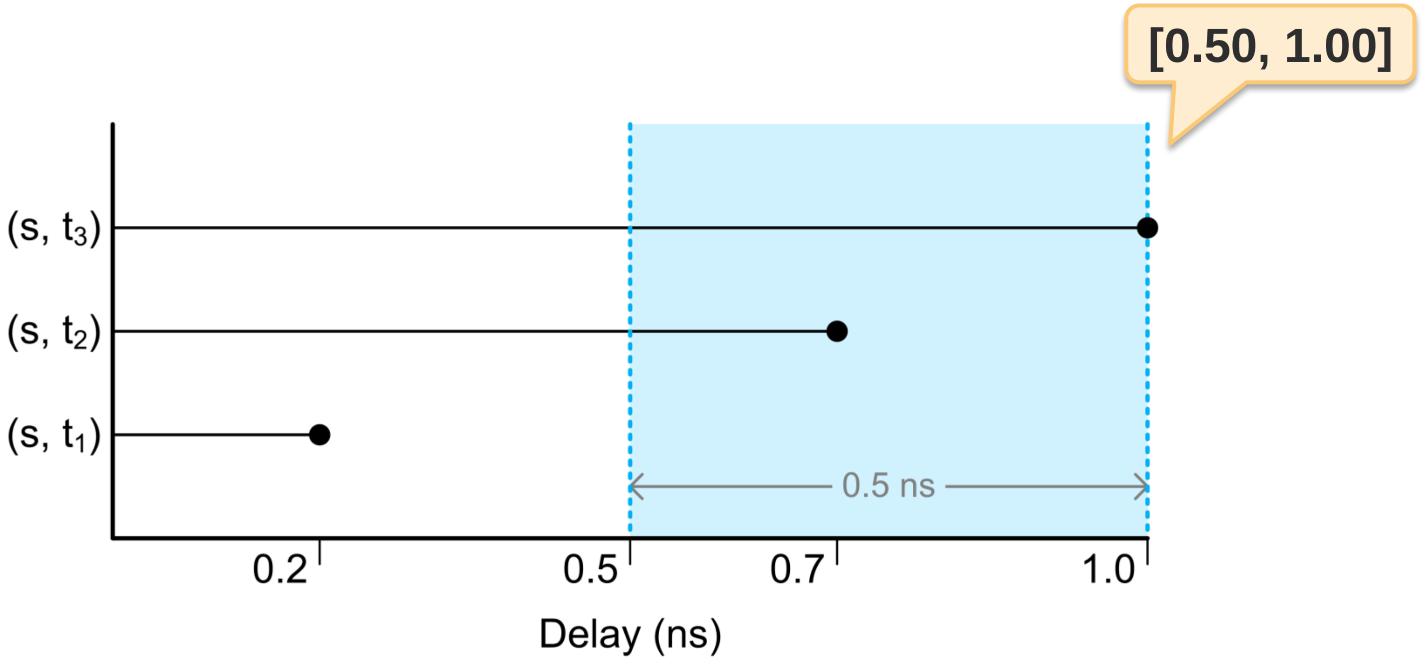
Maximum Skew

- **Setting delay bounds on each connection**
  - **Step 1: Use the largest delay value as the upper bound**



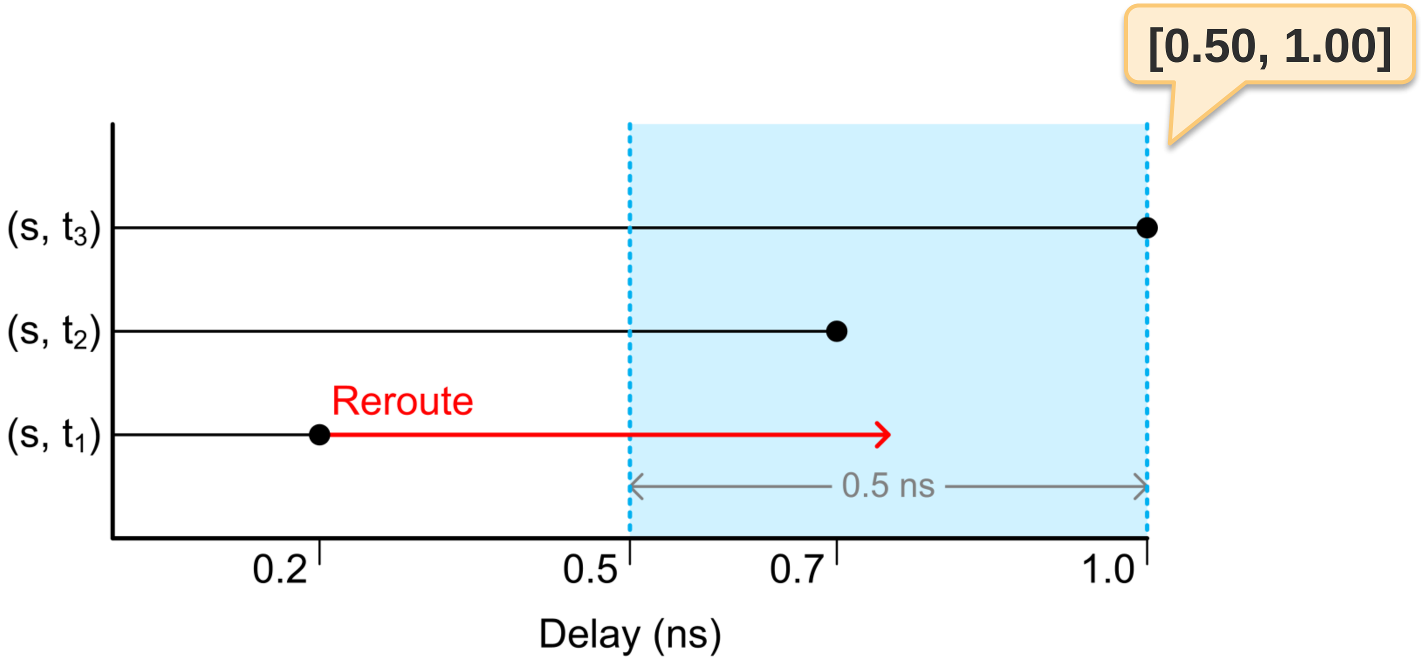
Maximum Skew

- Setting delay bounds on each connection
  - Step 1: Use the largest delay value as the upper bound
  - Step 2: Find the lower bound by subtracting the constraint value



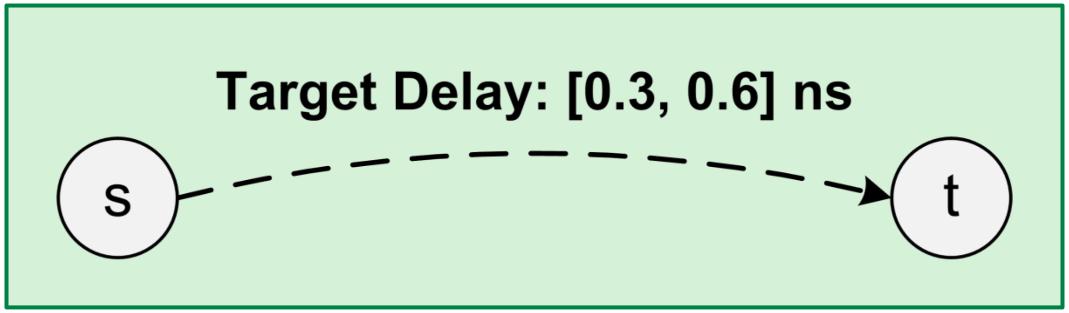
Maximum Skew

- **Setting delay bounds on each connection**
  - **Step 1: Use the largest delay value as the upper bound**
  - **Step 2: Find the lower bound by subtracting the constraint value**
  - **Step 3: Reroute any connections falling outside the bounds**



Specific Delay Window Routing

- Definition: routing a connection with a target delay between some lower and upper bounds
  - Lower and upper delay bounds form a “window”



- Allows us to optimize for various timing constraints constituting both lower and upper bounds on delay

Current FPGA Routing Technology

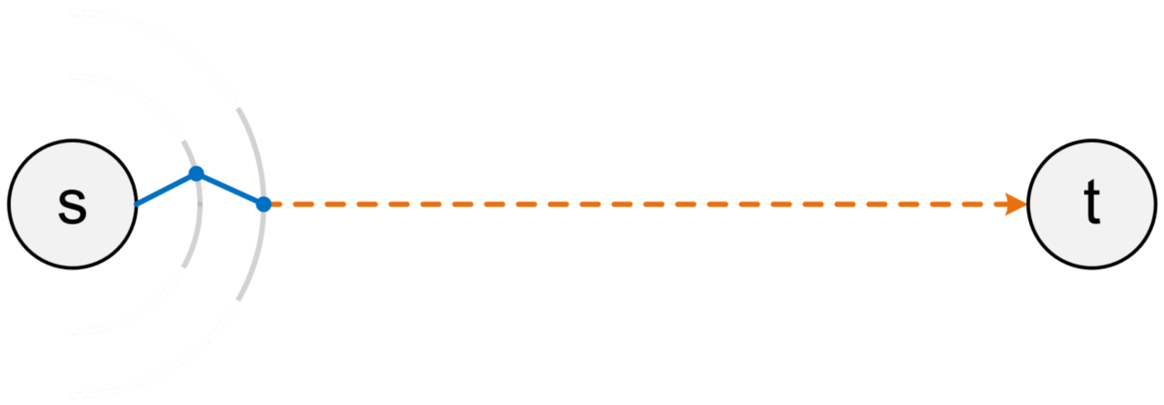
- PathFinder based (McMurchie 1995)
  - Basis of VPR router (Betz 1997) and many other academic and commercial FPGA routers
  - Effective at balancing routability and timing performance

$$\text{Cost}_n = \overbrace{A_{ij}}^{\text{Criticality Factor}}\ \underbrace{d_n}_{\text{Delay Cost}} + (1 - A_{ij})\ \underbrace{c_n}_{\text{Congestion Cost}}$$

  - Delay cost is the total delay of the connection
  - Traditional single-wave search: total delay contains estimation

Single-Wave Expansion

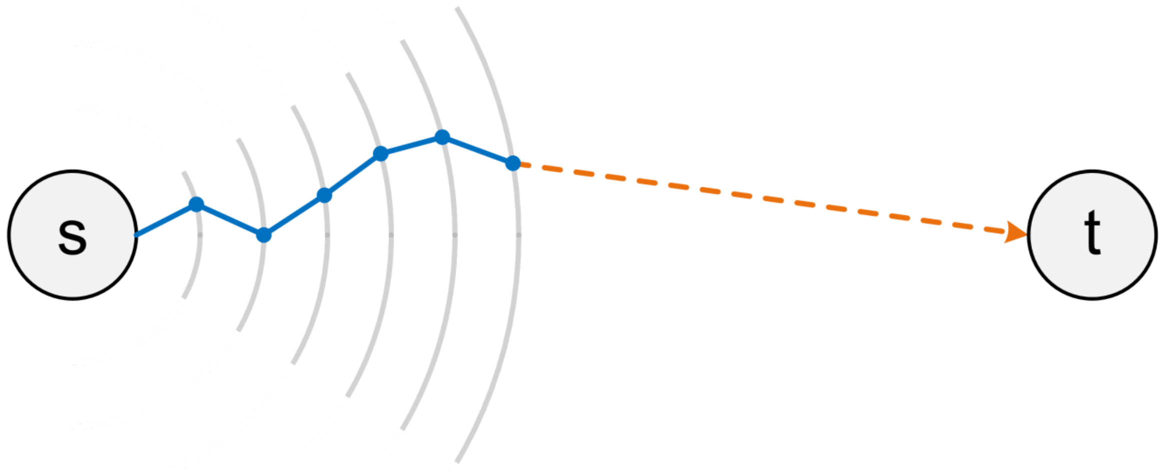
- One approach to performing Specific Delay Window Routing
  - Single-wave expansion using delay estimation to direct search towards the target delay window



Total Delay = Known Delay + Estimated Delay

Single-Wave Expansion

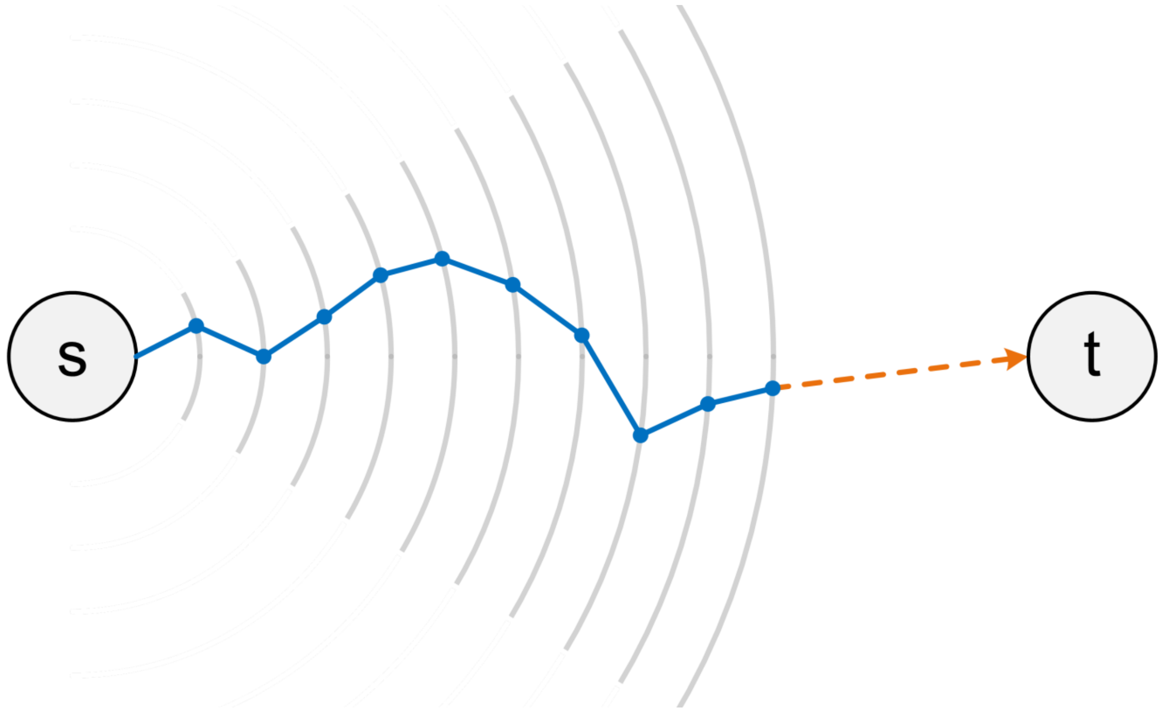
- One approach to performing Specific Delay Window Routing
  - Single-wave expansion using delay estimation to direct search towards the target delay window



Total Delay = Known Delay + Estimated Delay

Single-Wave Expansion

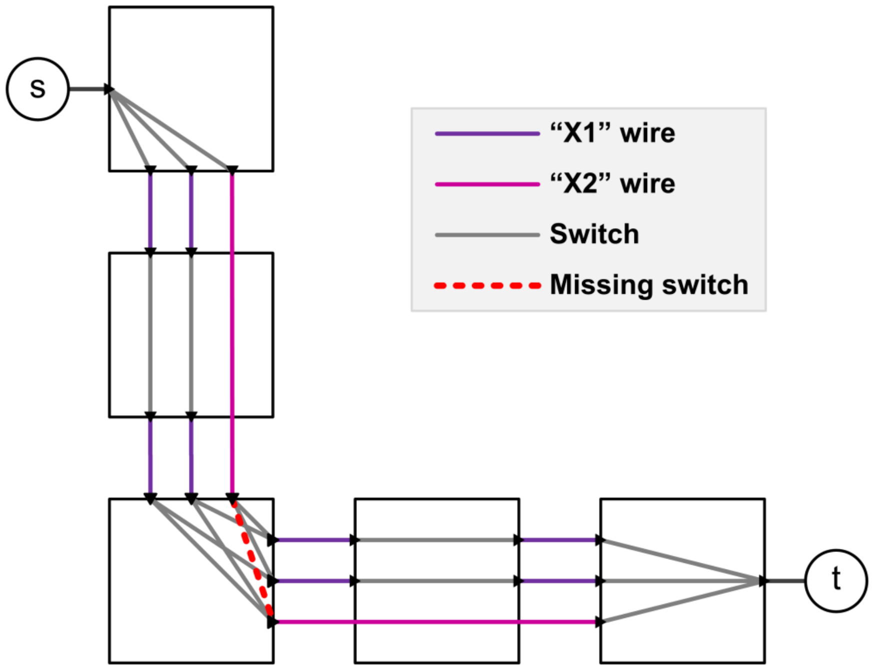
- One approach to performing Specific Delay Window Routing
  - Single-wave expansion using delay estimation to direct search towards the target delay window



Total Delay = Known Delay + Estimated Delay

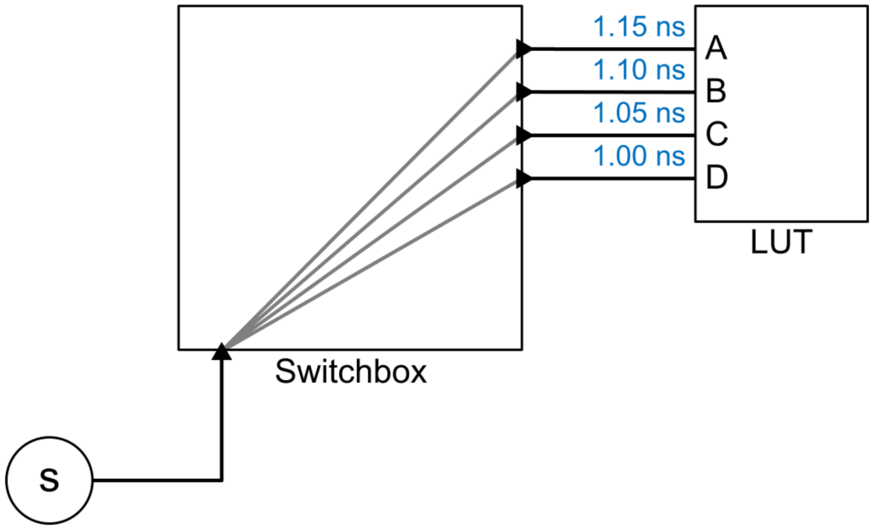
Accuracy Issues

- Estimation can be inaccurate
  - Sparse crossbar
    - Manhattan distance: 4
    - Estimate: 2 “X2” wires
    - Switch between “X2” wires is missing!



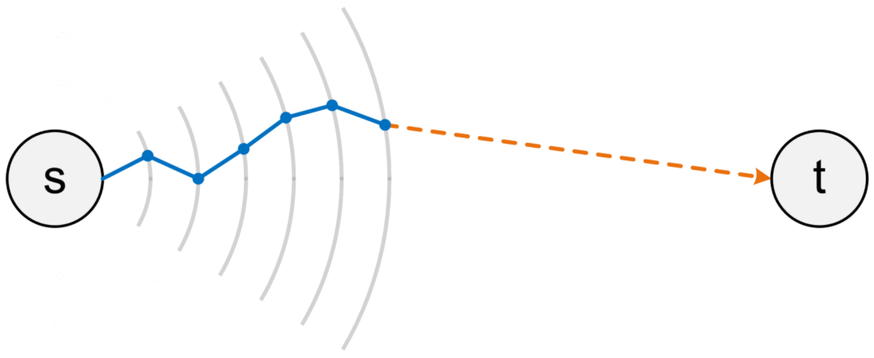
Accuracy Issues

- **Estimation can be inaccurate**
  - **Sparse crossbar**
  - **Swappable pins**
    - **LUT input pins have different delays**
    - **Pins are logically equivalent**
    - **Actual target pin not known during estimation**



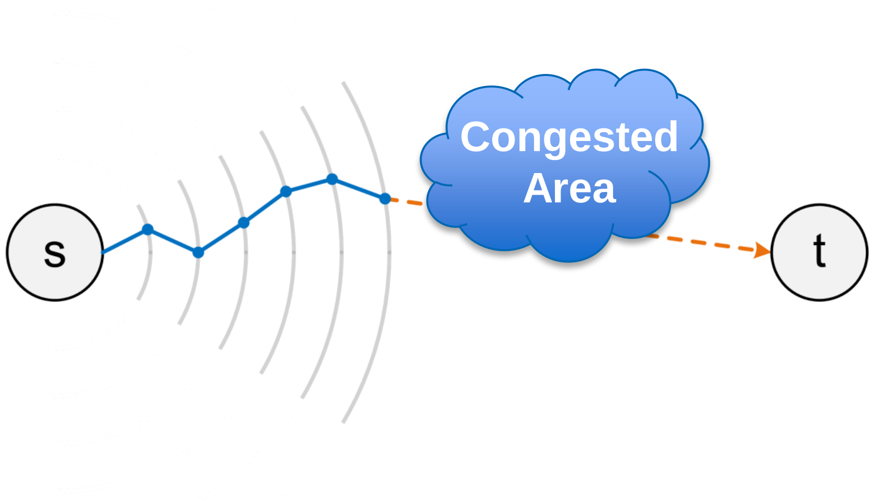
Accuracy Issues

- Estimation can be inaccurate
  - Sparse crossbar
  - Swappable pins
  - Congestion adds variability



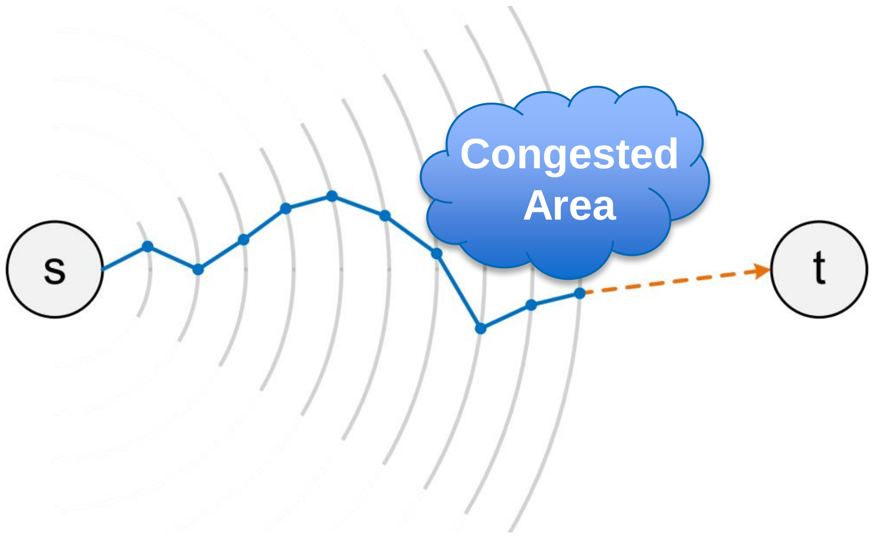
Accuracy Issues

- **Estimation can be inaccurate**
  - **Sparse crossbar**
  - **Swappable pins**
  - **Congestion adds variability**



Accuracy Issues

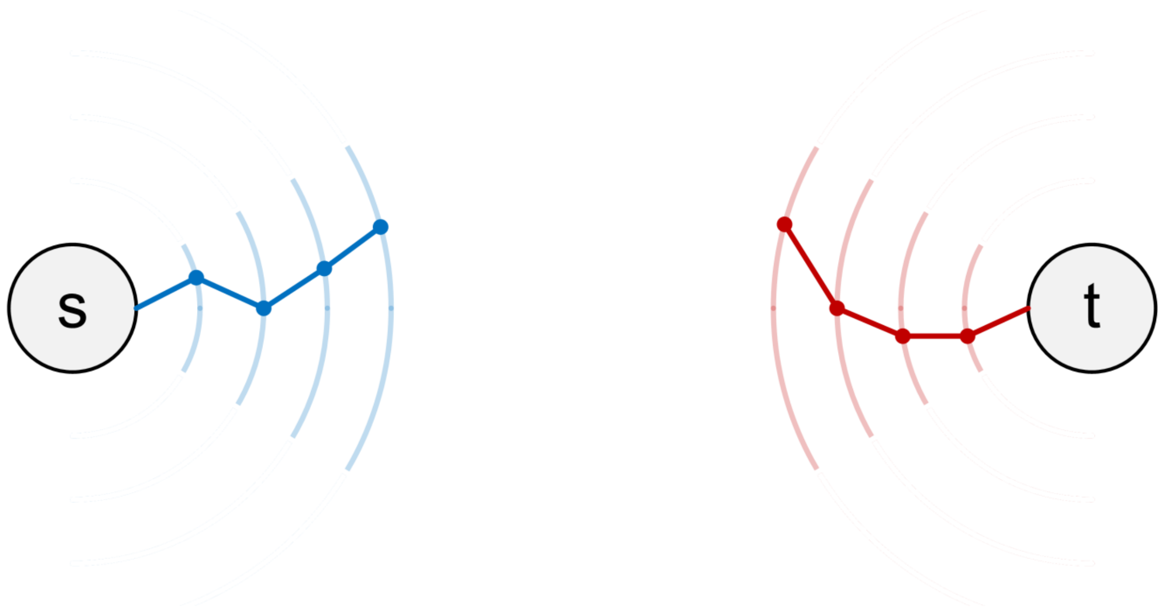
- **Estimation can be inaccurate**
  - **Sparse crossbar**
  - **Swappable pins**
  - **Congestion adds variability**



Dual-Wave Expansion

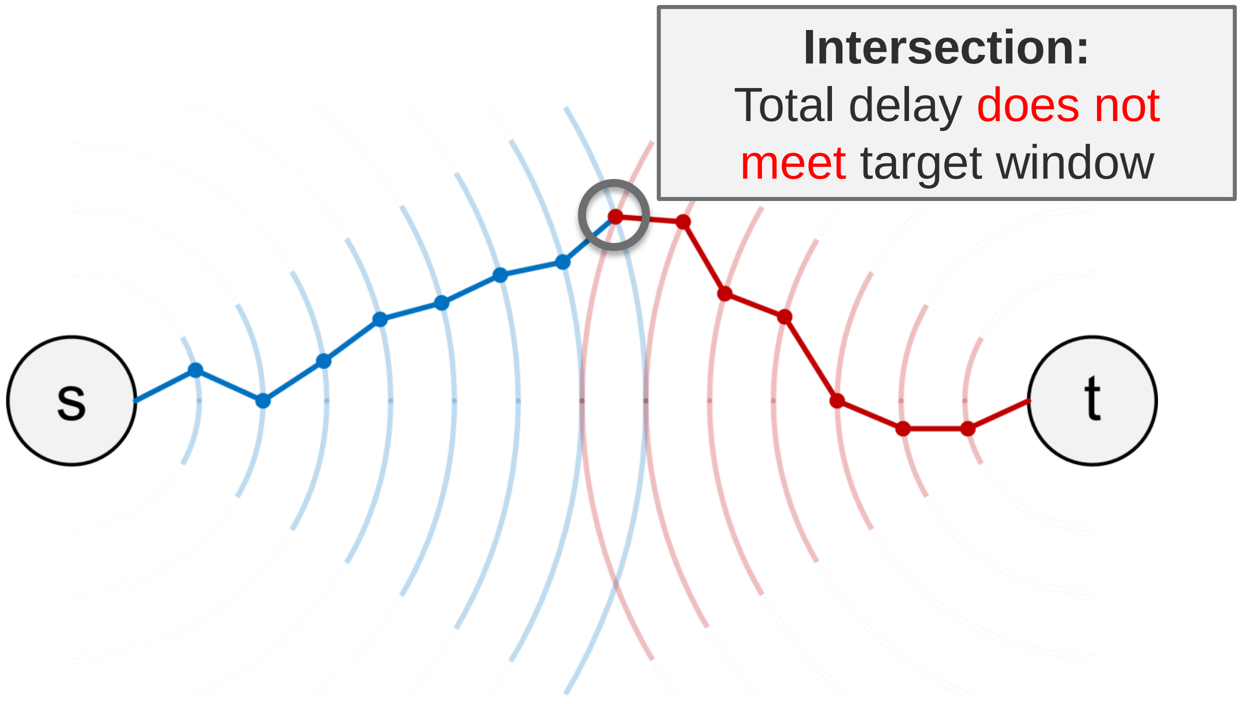
- To address these accuracy issues, we propose to use dual-wave search
  - Instead of directing one search towards the target, we can expand from both the source and target
  - Each time the waves intersect, we check if the resulting path meets the target delay window
  - This eliminates estimation from the selection process

Dual-Wave Expansion



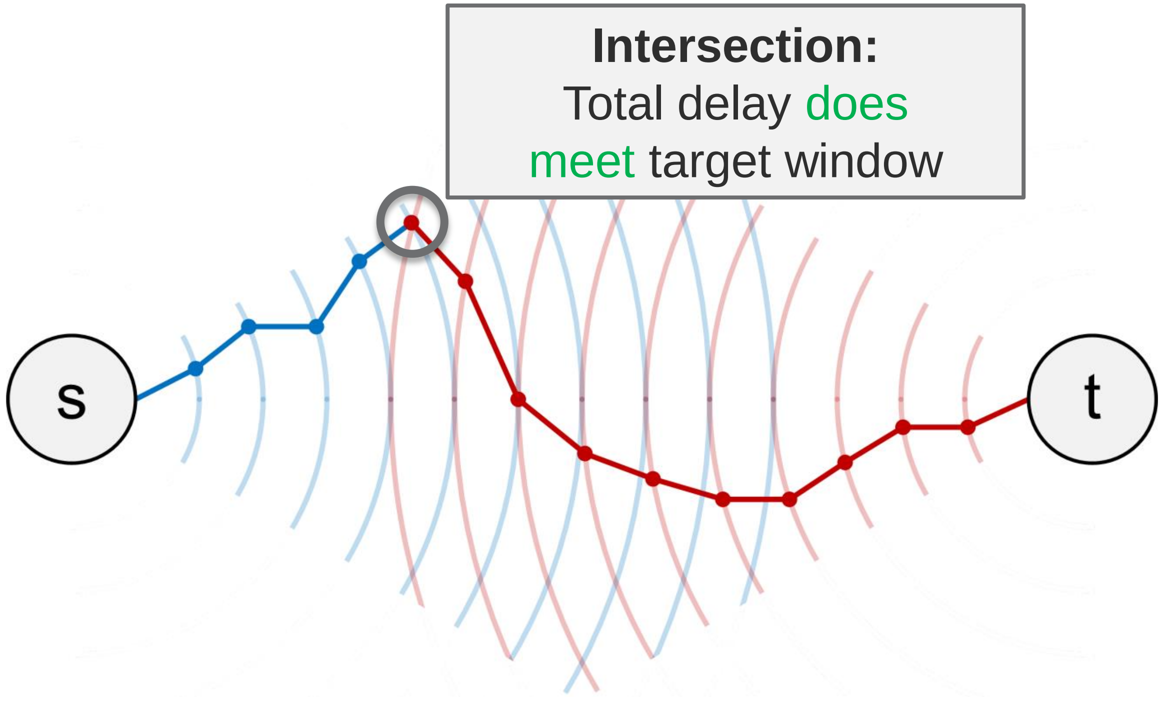
Wave from both source and target

Dual-Wave Expansion



Total Delay = Known Delay + Known Delay

Dual-Wave Expansion



Total Delay = Known Delay + Known Delay

Routing Flow

Normal Routing

- Global Routing
  - Clock routing, other architecture-specific routing
- Detail Routing
  - PathFinder-based

Specific Delay Window Routing

- Skew Optimization
  - Calculate delay windows for constrained connections
  - Perform specific delay window routing on connections in violation of skew constraint
- Hold Timing Optimization
  - Calculate delay windows using slack allocation
  - Perform specific delay window routing on connections in violation of hold timing

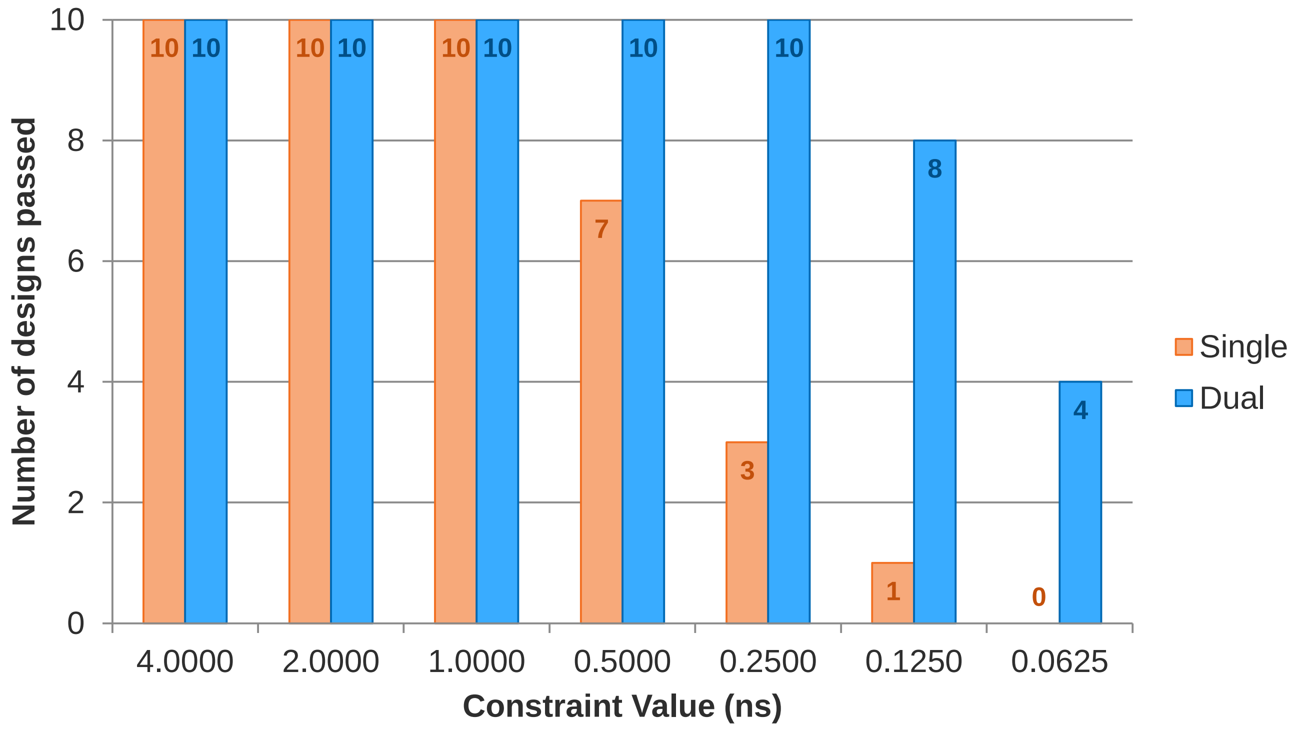
Experimental Methodology

- Compared two generations of commercial tools
  - Older one uses single-wave expansion with delay estimation
  - Newer one uses dual-wave expansion without delay estimation
- Ten designs each tested for hold timing and skew correction
  - Customer designs with known violations
  - Designs with strict skew constraints

	Device				Device		
Design	Family	LUT4s	Utilization	**Design**	Family	LUT4s	Utilization
N0	MachXO2	4K	70%	**H0**	LatticeEC	33K	9%
N1	LatticeECP3	150K	11%	**H1**	LatticeEC	33K	9%
N2	LatticeECP2	20K	45%	**H2**	MachXO	2K	85%
N3	LatticeECP3	150K	66%	**H3**	MachXO	2K	85%
N4	ECP5	85K	67%	**H4**	LatticeECP3	150K	83%
B0	MachXO2	2K	67%	**H5**	LatticeECP3	150K	14%
B1	LatticeECP2	20K	74%	**H6**	LatticeECP3	150K	82%
B2	LatticeECP3	70K	41%	**H7**	LatticeECP3	150K	87%
B3	LatticeECP3	150K	28%	**H8**	LatticeECP3	150K	76%
B4	ECP5	45K	23%	**H9**	LatticeECP3	150K	84%

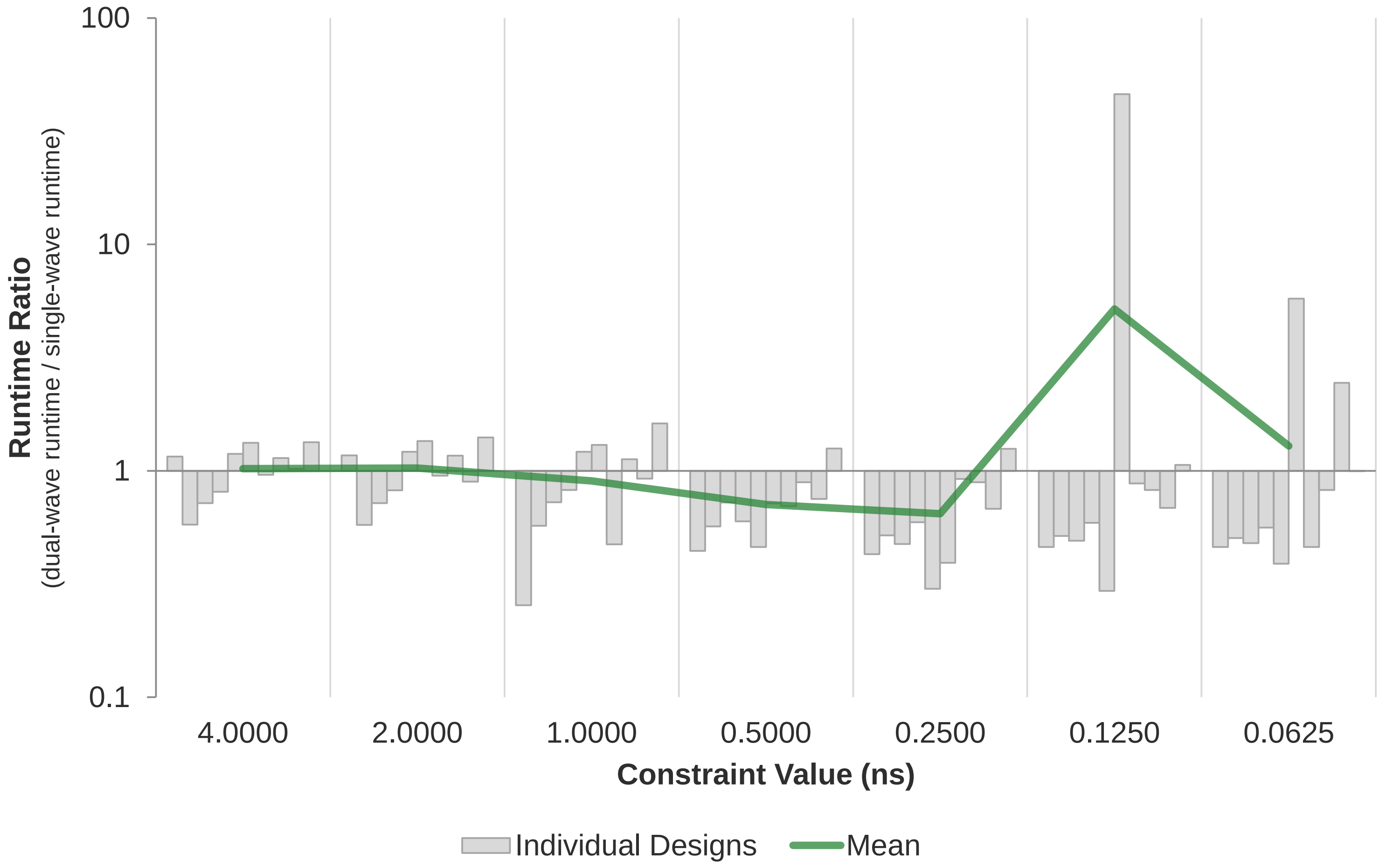
Experimental Results: Skew Correction

- Varied the skew constraint values from 4.0 ns (loose) to 0.0625 ns (very strict) and compared ability to find a solution



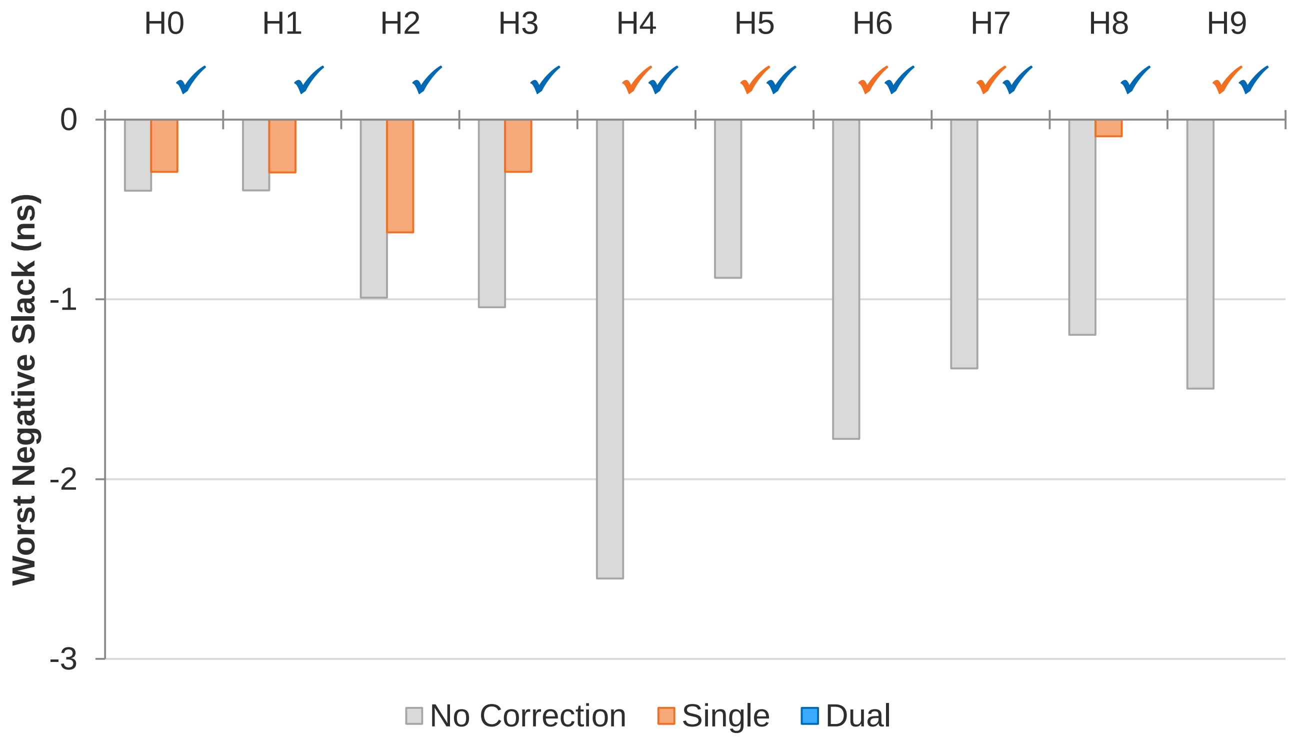
Experimental Results: Skew Correction

- **Runtime comparison between dual-wave and single-wave approaches**



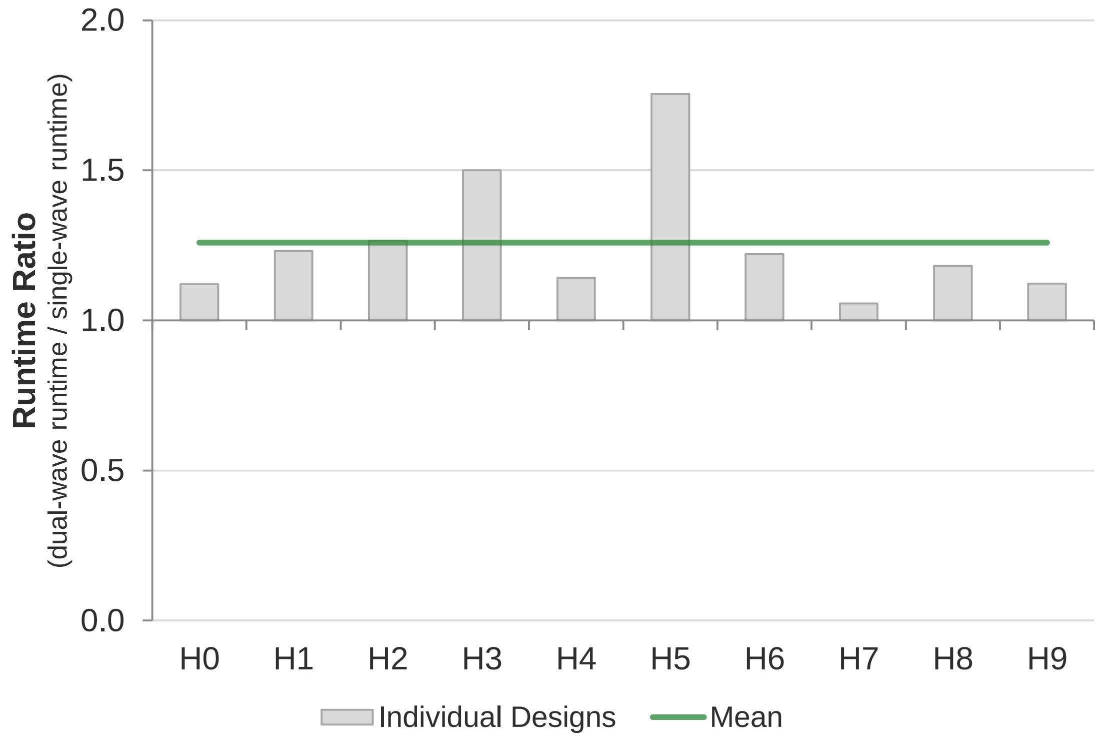
Experimental Results: Hold Correction

- **Compared ability to solve hold timing violations in 10 designs**
  - **Single-wave approach corrected only 5 of 10**
  - **Dual-wave approach corrected all 10 of 10**



Experimental Results: Hold Correction

- **Runtime not significantly increased using dual-wave approach**
  - **Older tool generation used heuristic method for setup-awareness**
  - **Newer tool generation uses slack allocation, which uses more time**



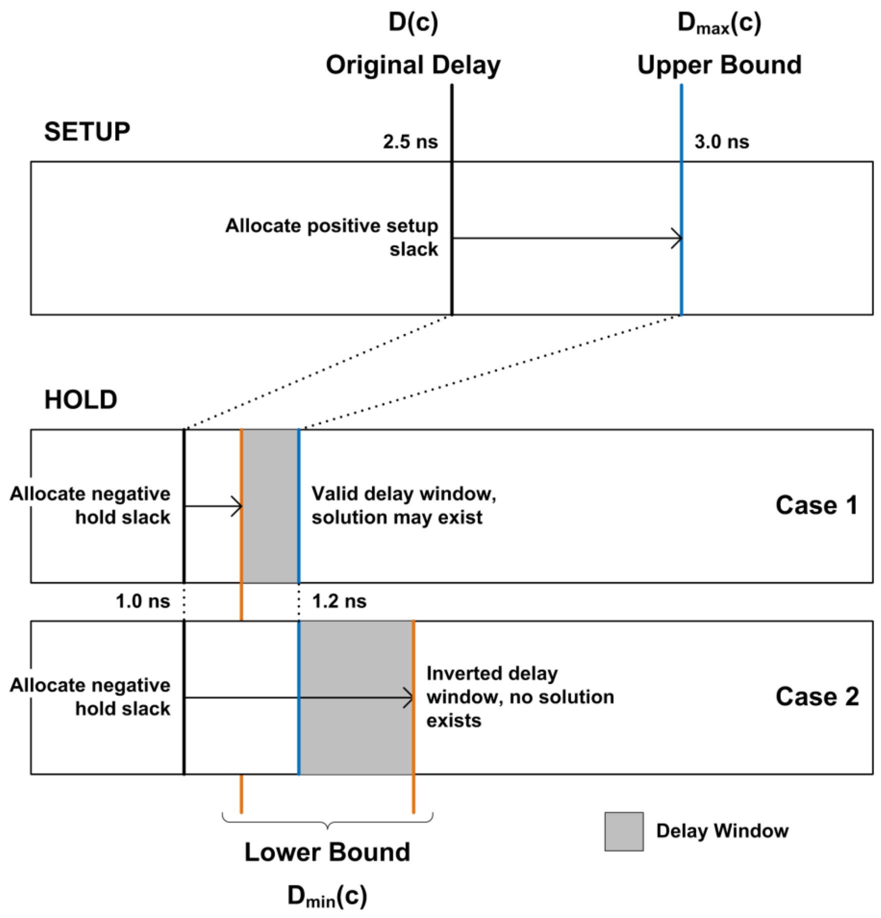
Conclusion

- Presented specific delay window routing as a generic framework to optimize for constraints constituting lower and upper bounds on delay
- Proposed dual-wave expansion to address accuracy issues with single-wave expansion when targeting a specific delay window
- Future work
  - Applying specific delay window routing to more constraints
    - Clock-to-output
    - Cycle stealing

Multiple Speed Grades

- We consider multiple speed grades for slack allocation
  - For setup timing, we use the design speed grade
  - For hold timing, we use the fastest speed grade

- Inverted delay windows
  - If the lower bound exceeds the upper bound, then we cannot satisfy the setup and hold requirements for the connection



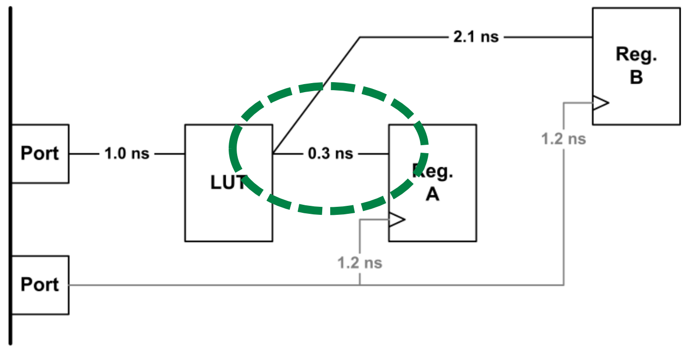
D(c)
Original Delay
$D_{max}(c)$
Upper Bound
SETUP
2.5 ns
3.0 ns
Allocate positive setup slack
HOLD
Allocate negative hold slack
Valid delay window, solution may exist
Case 1
1.0 ns
1.2 ns
Allocate negative hold slack
Inverted delay window, no solution exists
Case 2
Lower Bound
$D_{min}(c)$
Delay Window

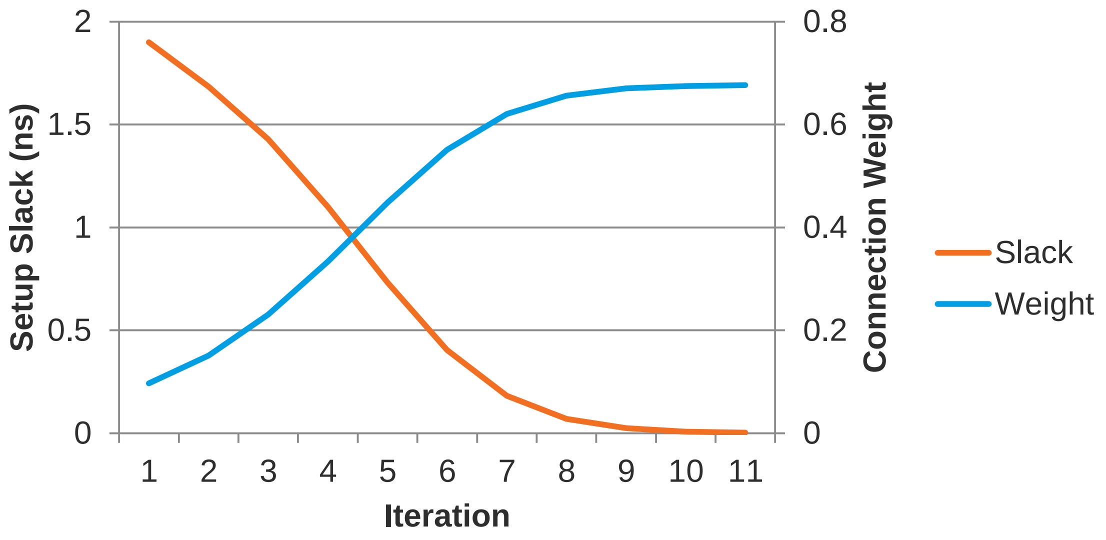
Slack Allocation

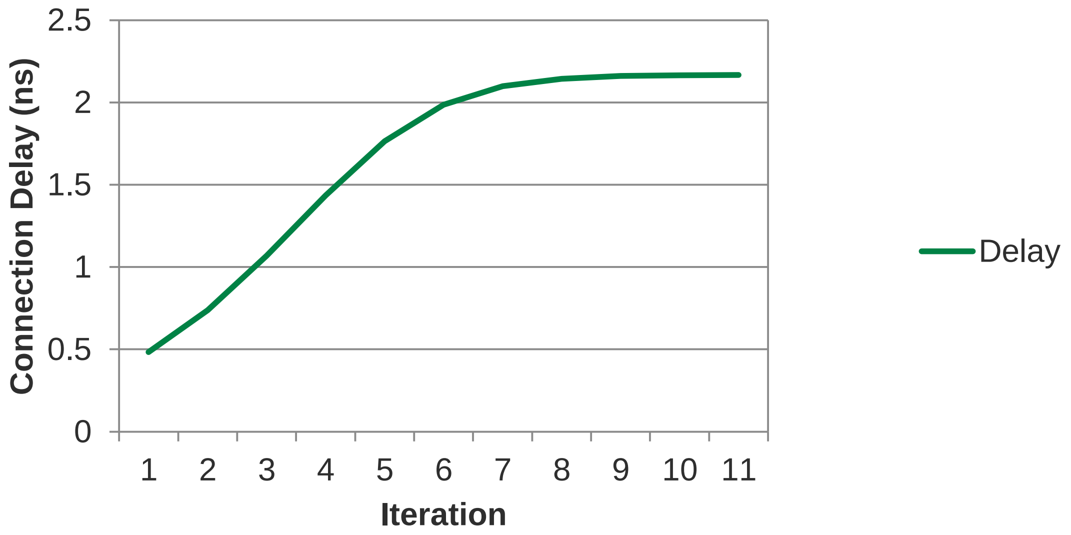
```
LOOP {
   1. Update timing
   2. Compute slack on each connection
         slack(c) = min(slack(p))
            where p is any path containing c
   Stop if the cumulative slack is near zero
   3. Compute the weight for each connection
         weight(c) = delay(c) / max(delay(p))
            where p is any path containing c
   4. Allocate slack for each connection
         delay(c) = delay(c) + weight(c) * slack(c)
}
```

Slack Allocation

Slack allocation on setup timing for the circled connection



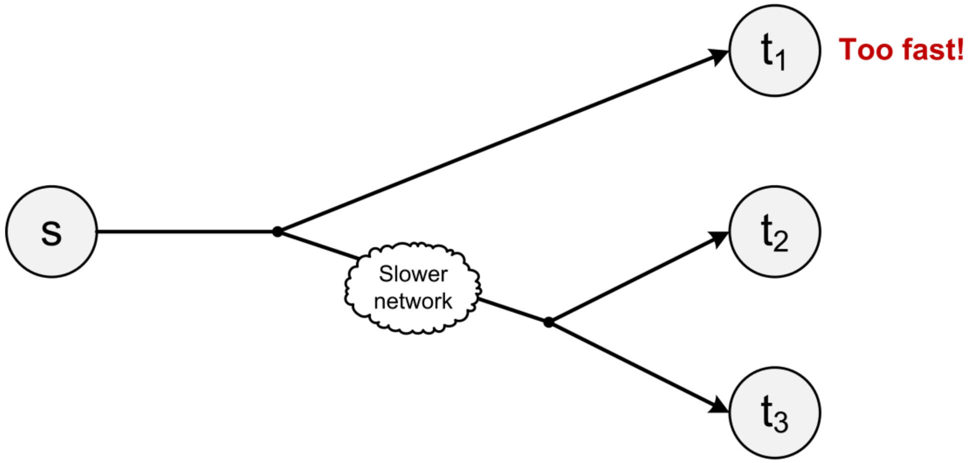




Results Analysis

- Why does design "B0" take so much longer with dual-wave expansion?
  - Both single-wave and dual-wave approaches fail for this design with a 0.125 ns constraint

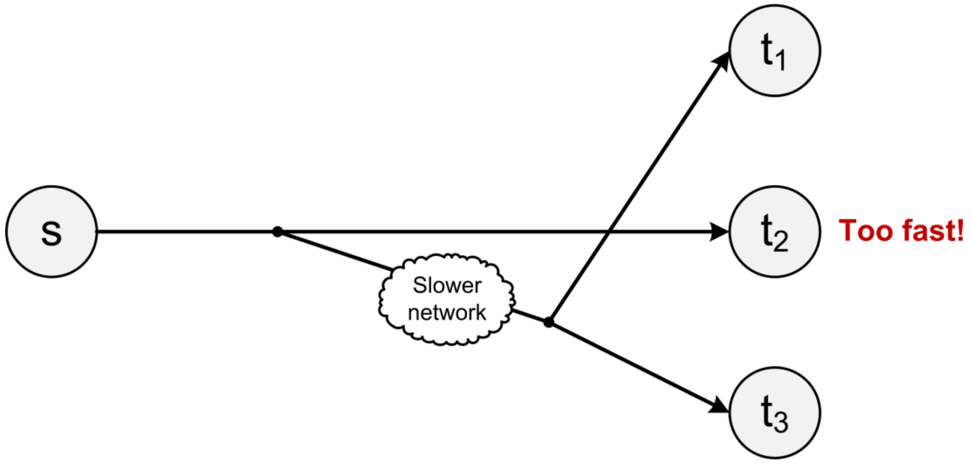
- Architectural corner case involving three connections
  - Stuck in a scenario where 2 have equal skew, but another is faster



Results Analysis

- Why does design "B0" take so much longer with dual-wave expansion?
  - Both single-wave and dual-wave approaches fail for this design with a 0.125 ns constraint

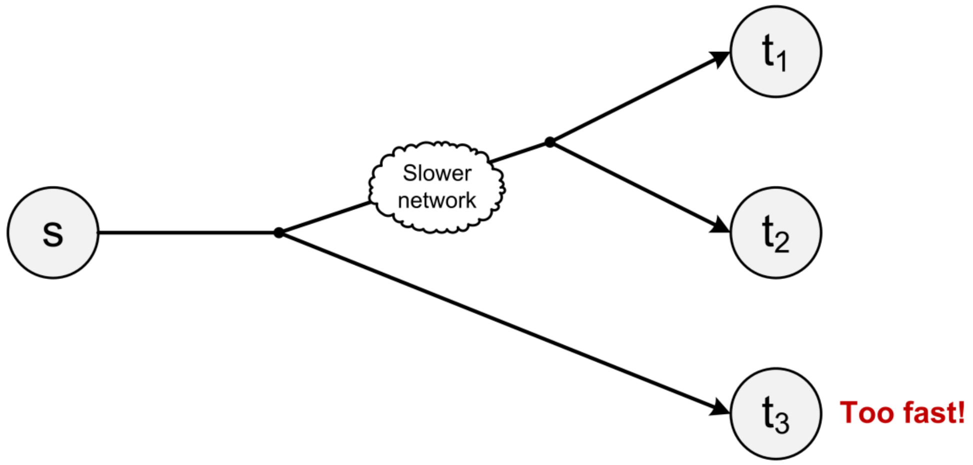
- Architectural corner case involving three connections
  - Stuck in a scenario where 2 have equal skew, but another is faster



Results Analysis

- Why does design “B0” take so much longer with dual-wave expansion?
  - Both single-wave and dual-wave approaches fail for this design with a 0.125 ns constraint

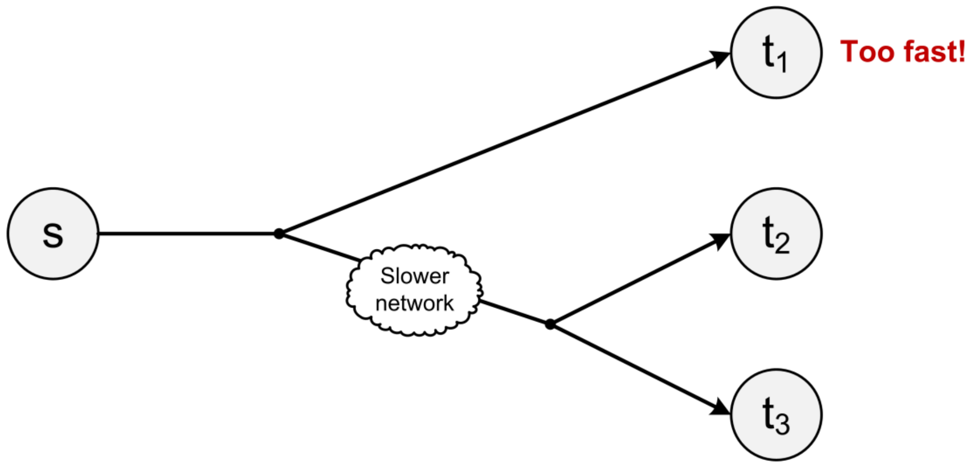
- Architectural corner case involving three connections
  - Stuck in a scenario where 2 have equal skew, but another is faster



Results Analysis

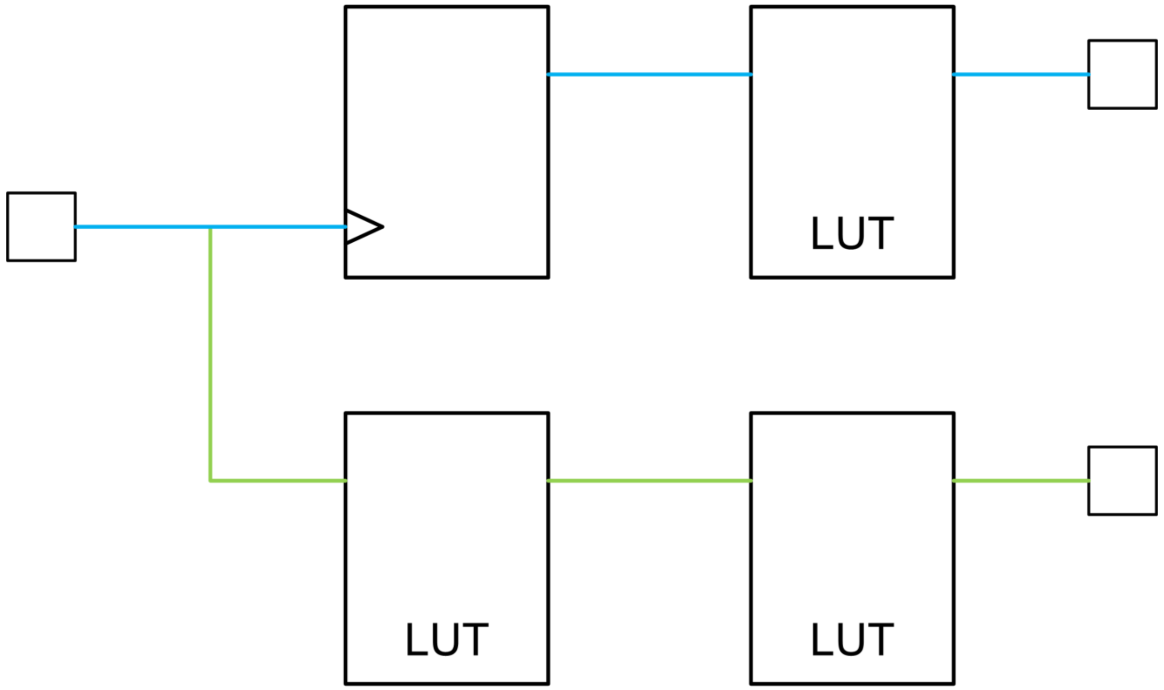
- Why does design "B0" take so much longer with dual-wave expansion?
  - Both single-wave and dual-wave approaches fail for this design with a 0.125 ns constraint

- Architectural corner case involving three connections
  - Stuck in a scenario where 2 have equal skew, but another is faster



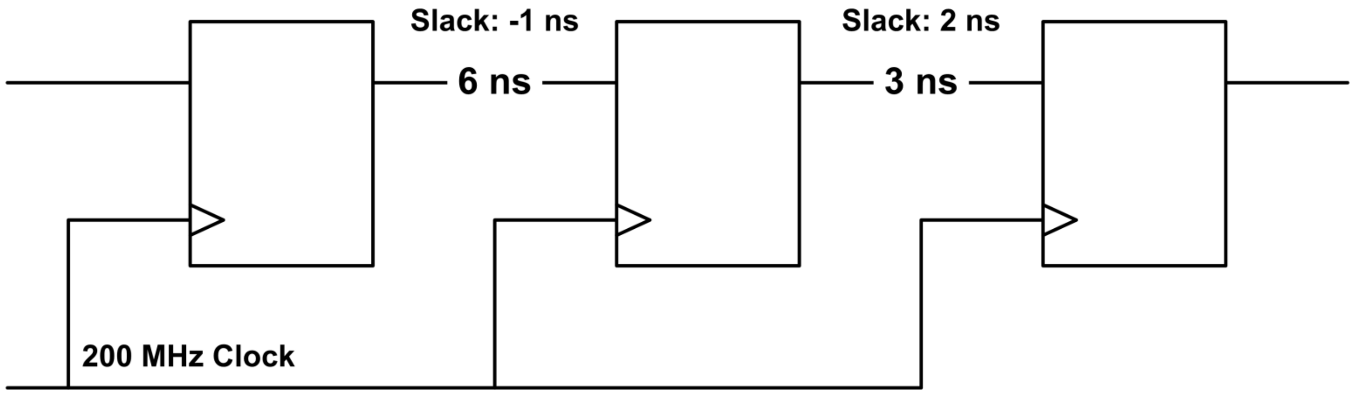
Future Work

- Clock-to-output constraint
  - Relative timing constraint between paths
  - Example: 0.5 ns < clock-to-out path – clock-out path < 4 ns



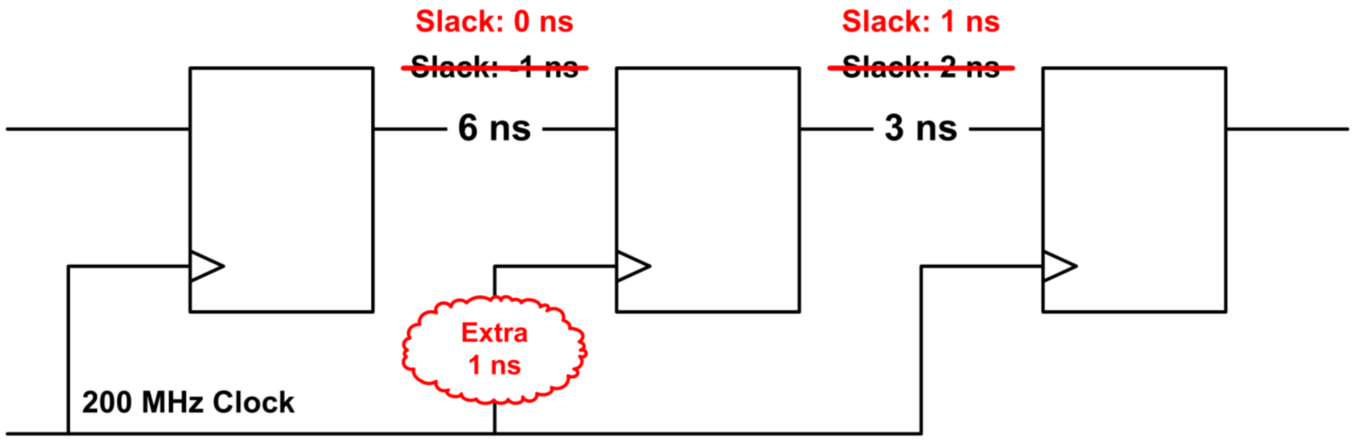
Future Work

- Cycle stealing
  - Add extra delay to clock path to alleviate setup violations



Future Work

- Cycle stealing
  - Add extra delay to clock path to alleviate setup violations


Slack: 0 ns
~~Slack: -1 ns~~
6 ns
Slack: 1 ns
~~Slack: 2 ns~~
3 ns
Extra 1 ns
200 MHz Clock