



Scan Timing Closure Via Clock Tree Aware Reordering

Luis F. Retana, Graphics CAD Team
Intel Corp

Agenda



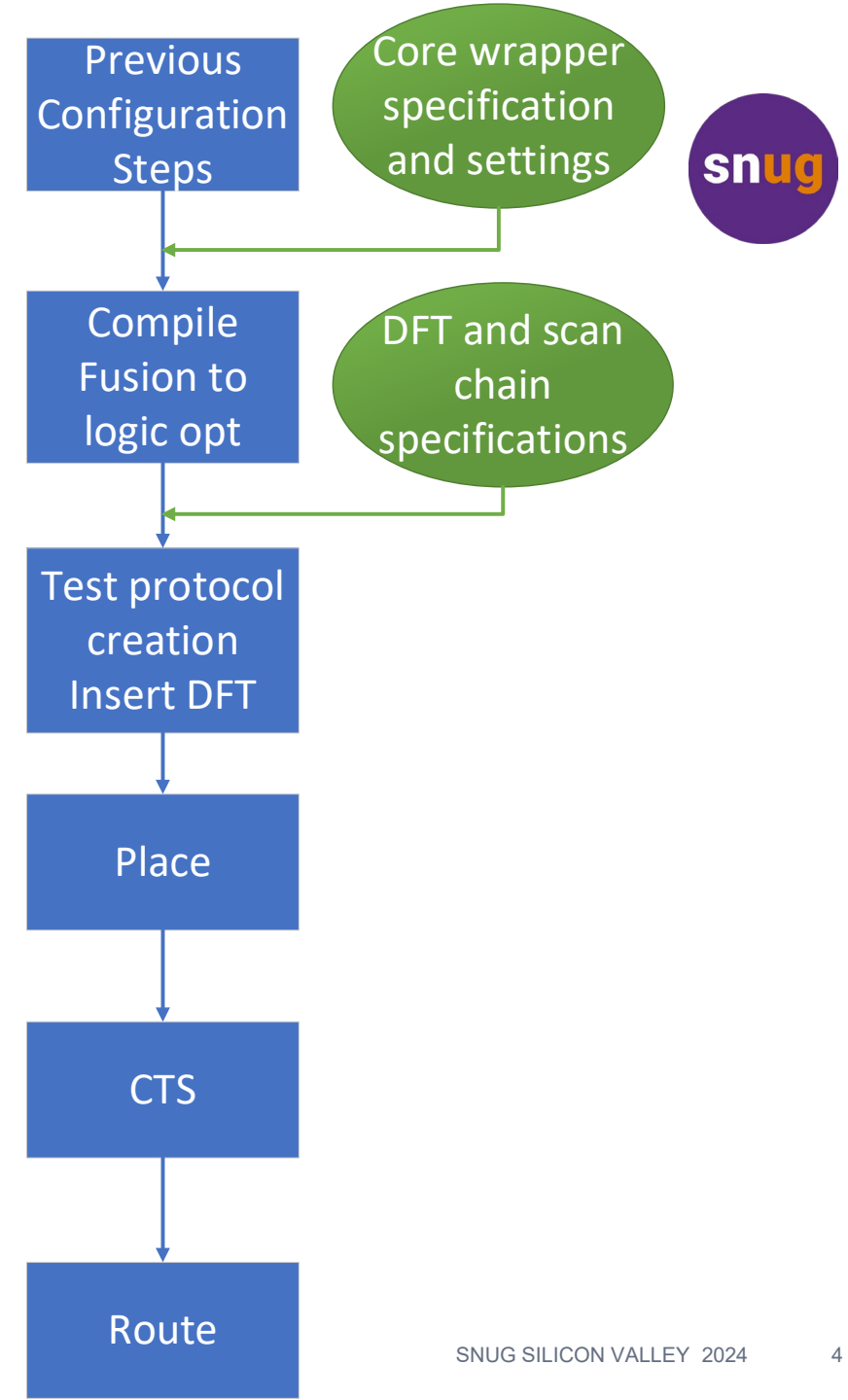
- Background
- Problem Statement
- Intended Solution: Clock Tree Aware Reorder
- Clock Tree Aware Reorder: Implementations
 - Custom Scripted Solution
 - Native Synopsys Fusion Compiler™ Clock Tree Aware Reorder
- Results
- Conclusions and Future work



Background: Our Scan Flow

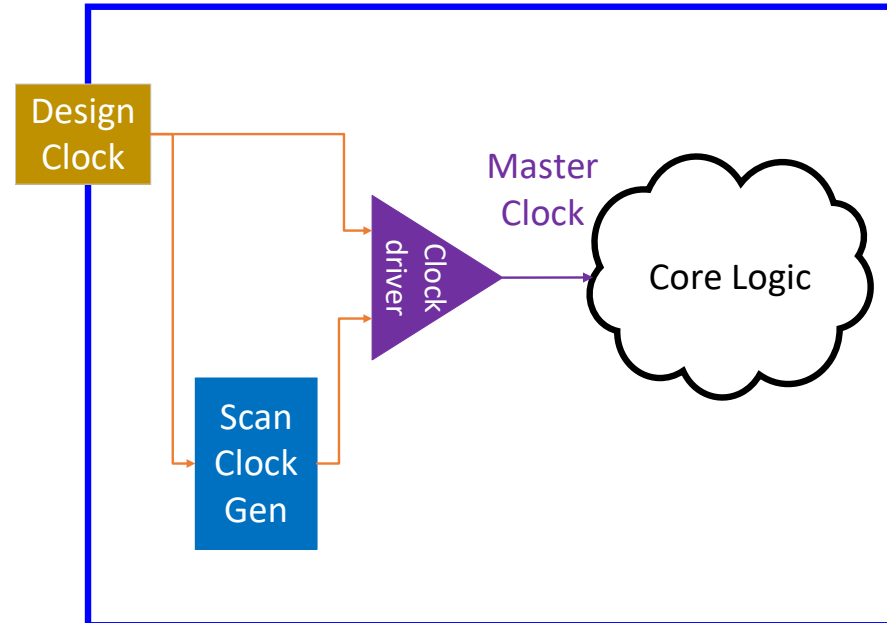
Scan synthesis flow

- Scan configuration happens before and after logic opt
- Scan reordering and repartition is done across different steps of the design flow
- No other scan optimization afterwards
 - In our implementation flow

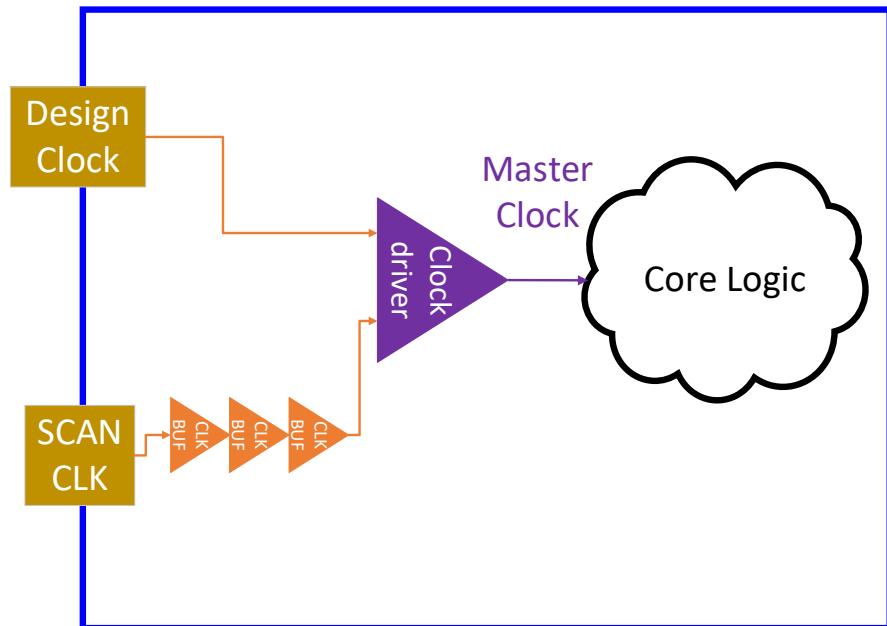


Background: Motivation

- Industry standard Streaming Scan Network implementation
- Different clock network techniques
 - Regular CTS and Structured CTS
- Scan clock skew difference due Regular CTS



Before

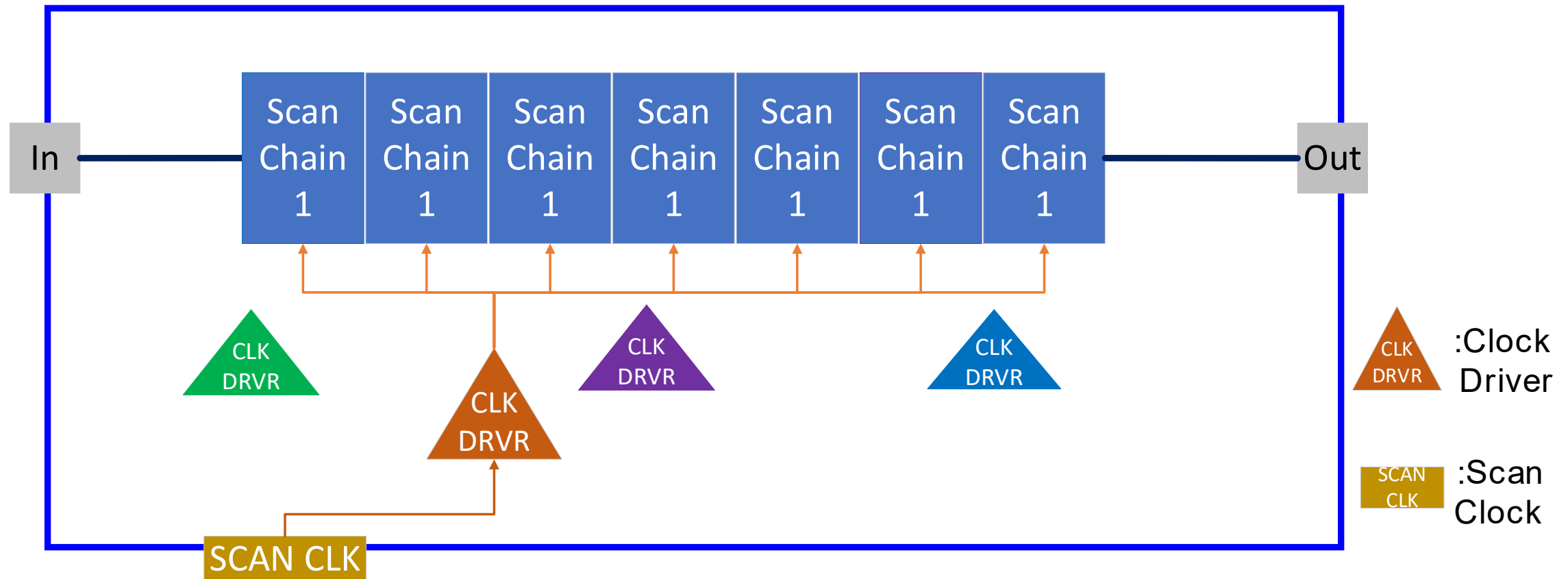


After



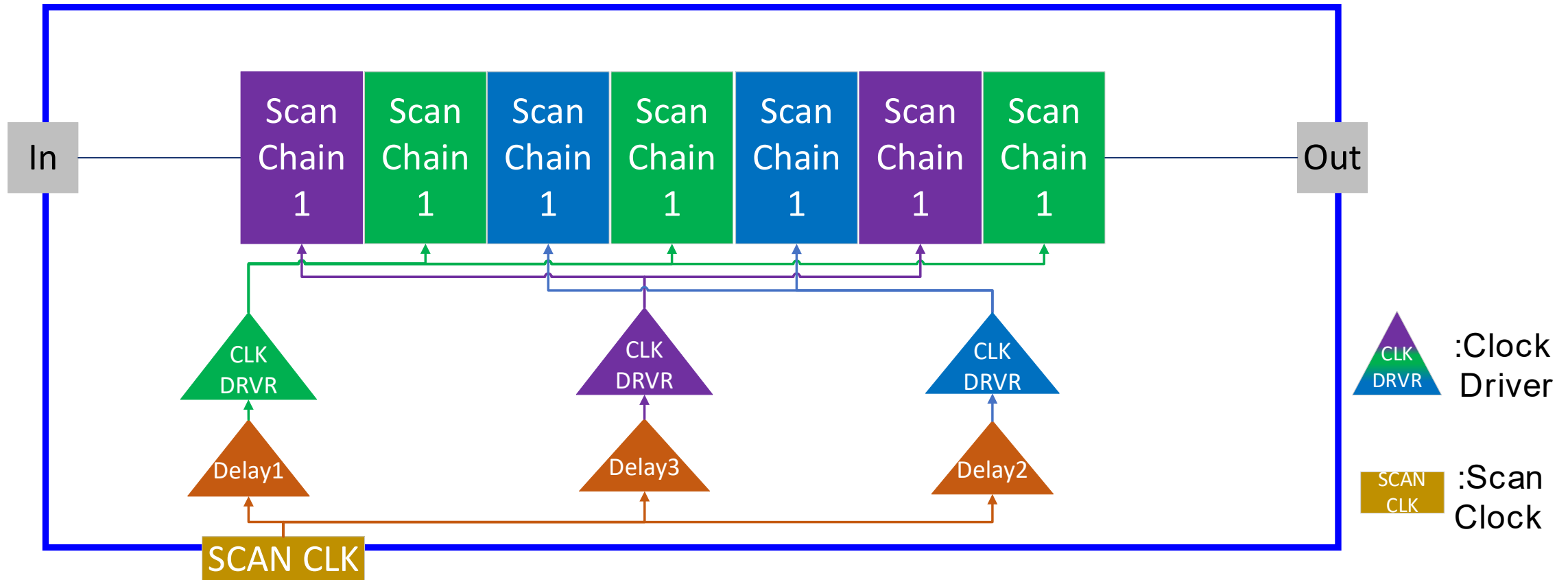
Background: Conditions for Scan Insertion Pre-CTS

- Scan clock driver is shared among flops during scan insertion



Background: Conditions for Scan Optimization Post-CTS

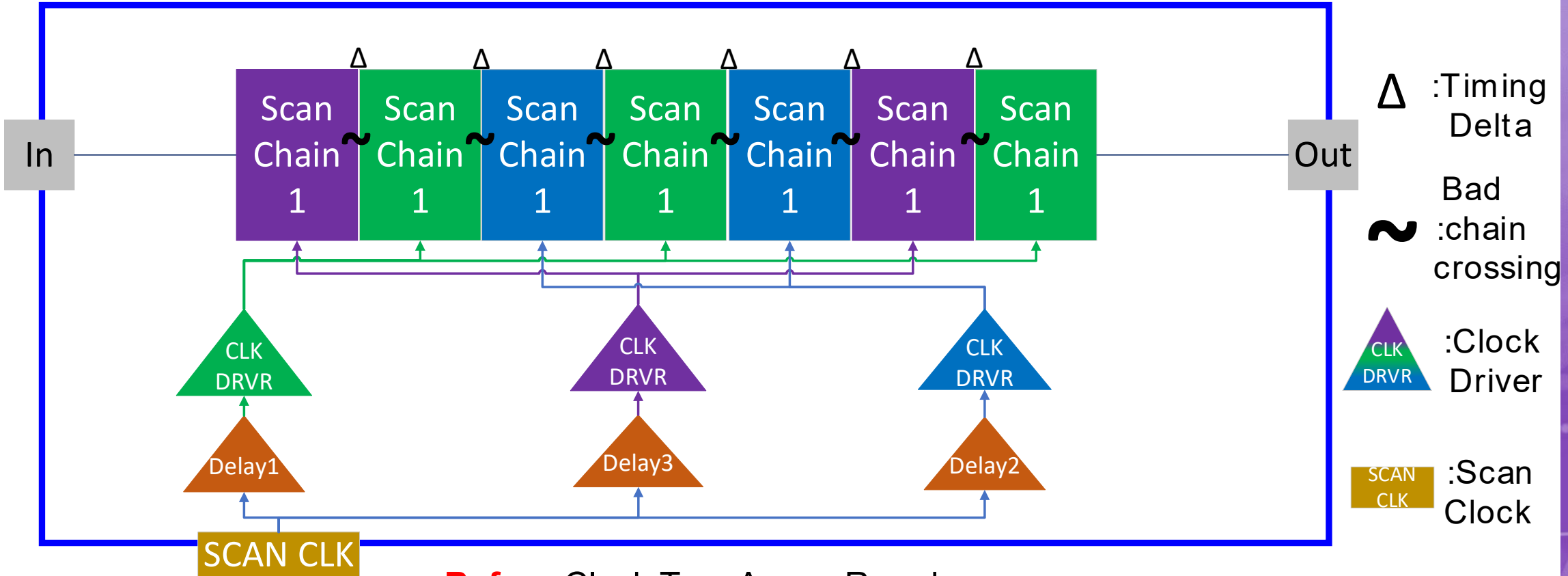
- Multiple clock drivers after CTS completes increasing the scan timing overhead



Problem Statement

Skew difference across clock drivers

- More bad chain crossings = more timing closure overhead!
 - Bad Chain Crossing = different clock driver connection



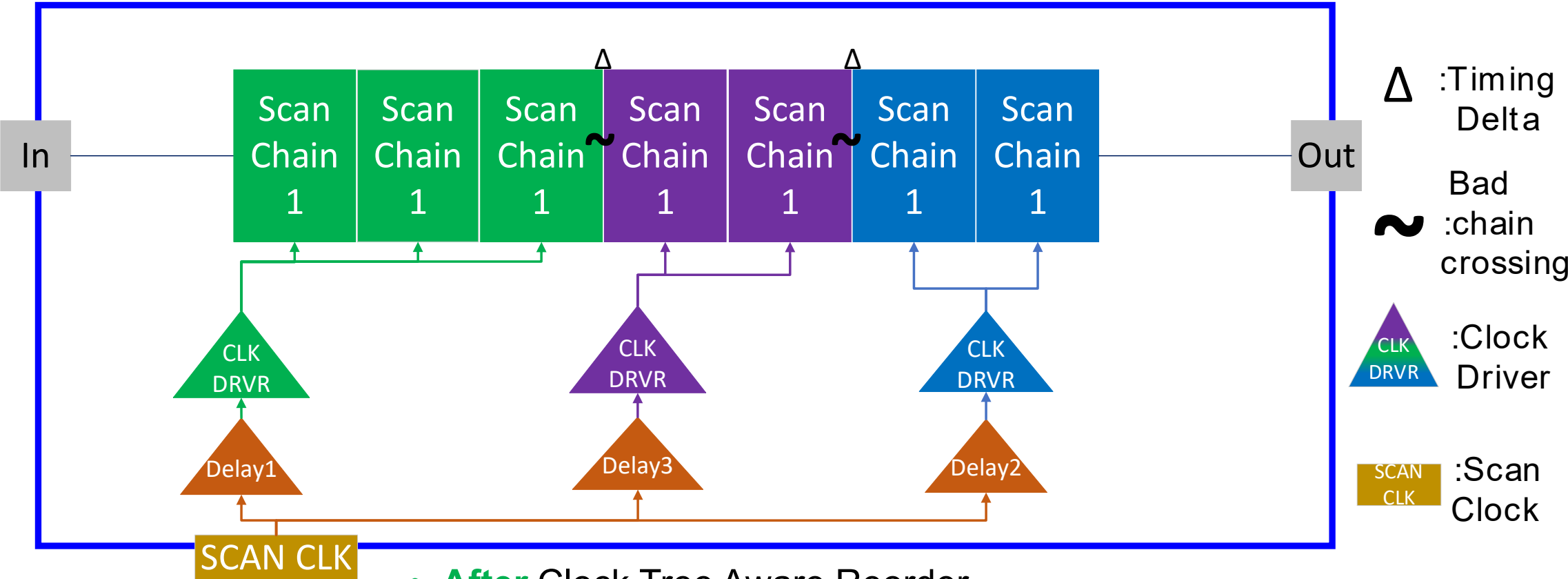
• **Before** Clock Tree Aware Reorder



Intended Solution

Clock Tree Aware Reorder

- Reduced timing closure overhead for scan paths



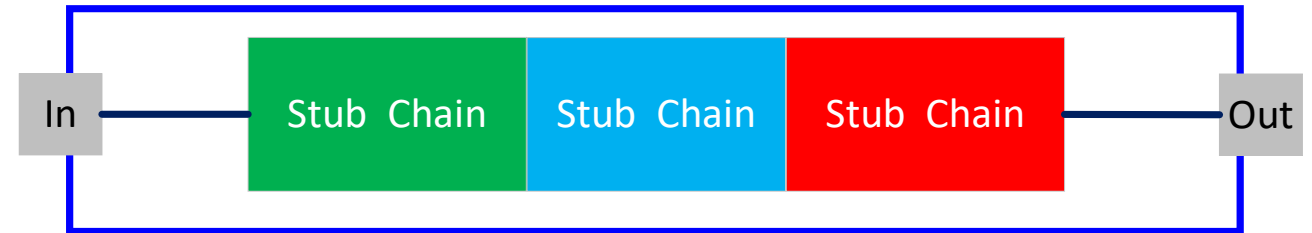
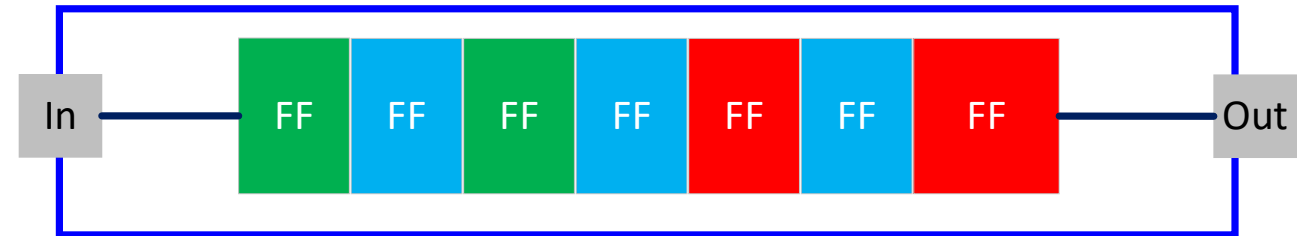
• **After** Clock Tree Aware Reorder



Custom Scripted Solution

High-Level Algorithm

- Reorders the scan chains based on clock driver
 - Creates multiple stub chain for a single scan path
- Runs `optimize_dft` to improve timing on each stub chain

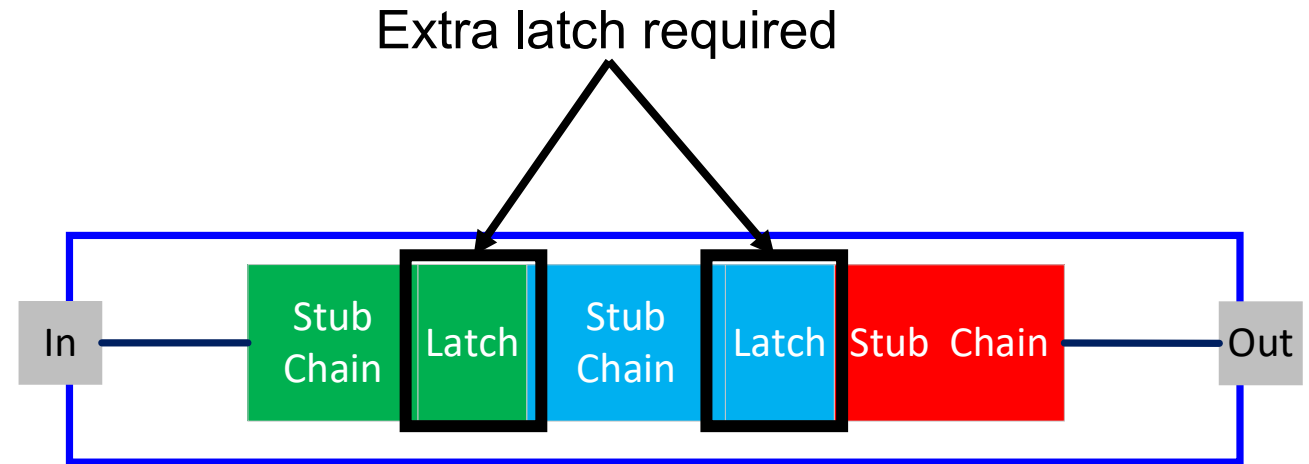


Red, green and blue represent different clock driver elements

Custom Scripted Solution

Limitations

- Does not look across chains
- Stub chain limit the optimization within the same scan path
- Extra min protection latches required



Red, green and blue represent different clock driver elements



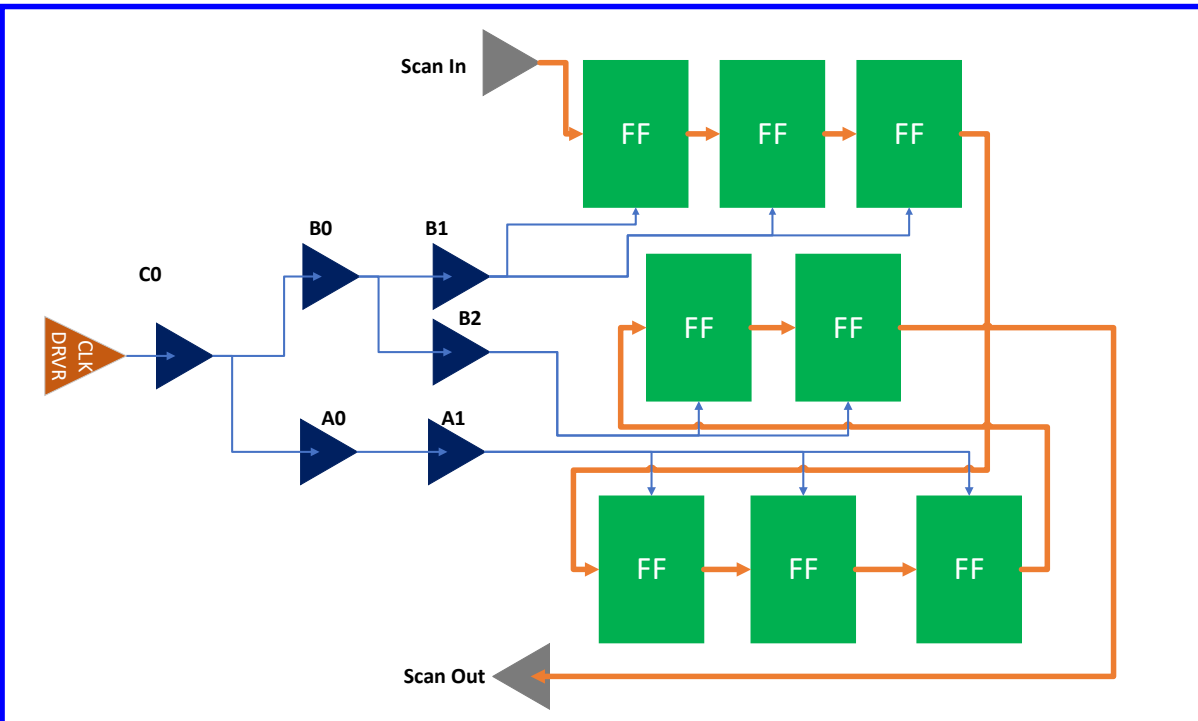
Native Clock Tree Aware Reorder

Native Optimization Technique

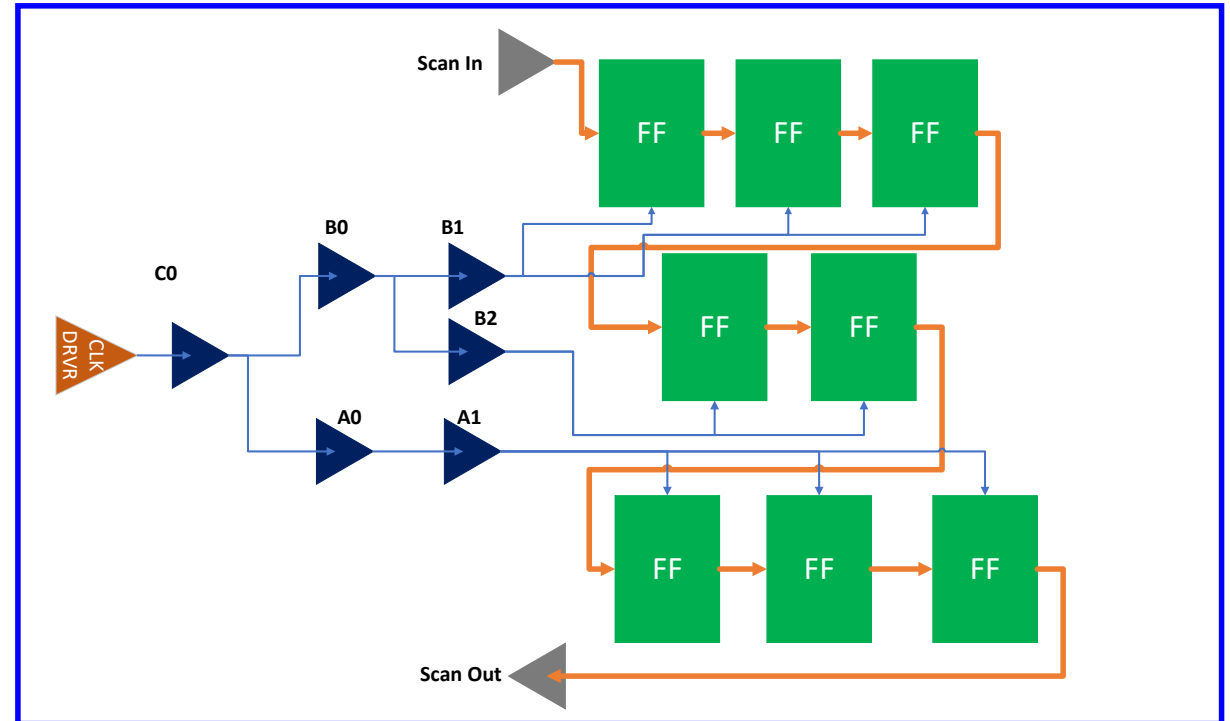
- Clock Tree Aware Reorder looks at full clock tree hierarchy
 - Optimizes the scan path by looking at different levels of the clock tree



Before



After



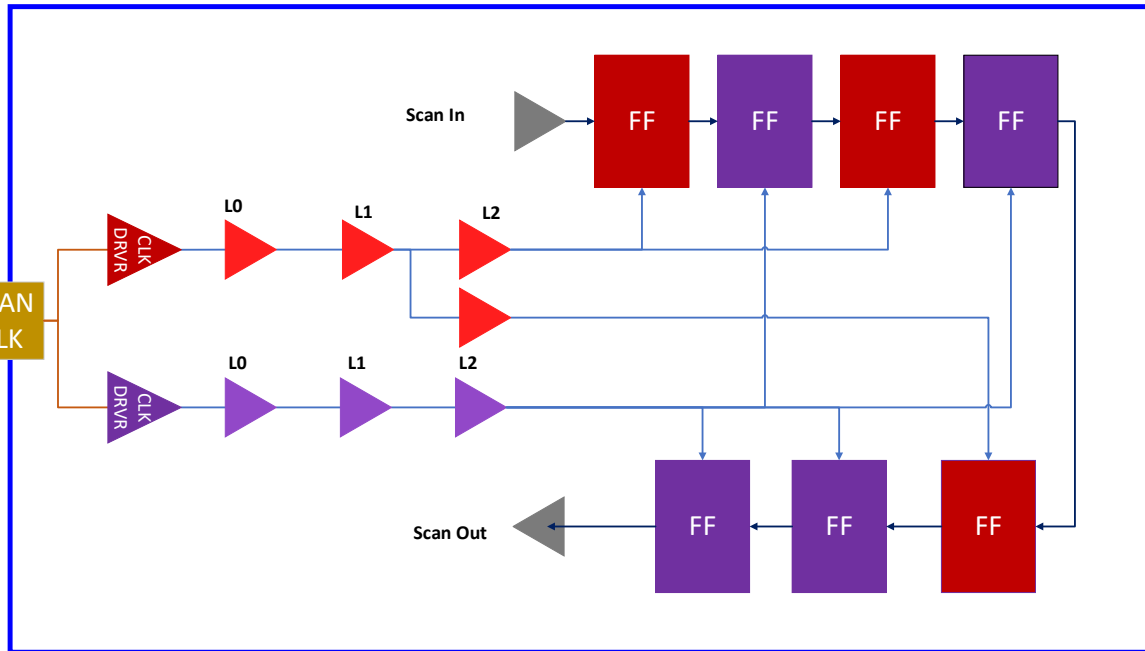
Native Clock Tree Aware Reorder

Native Optimization Technique

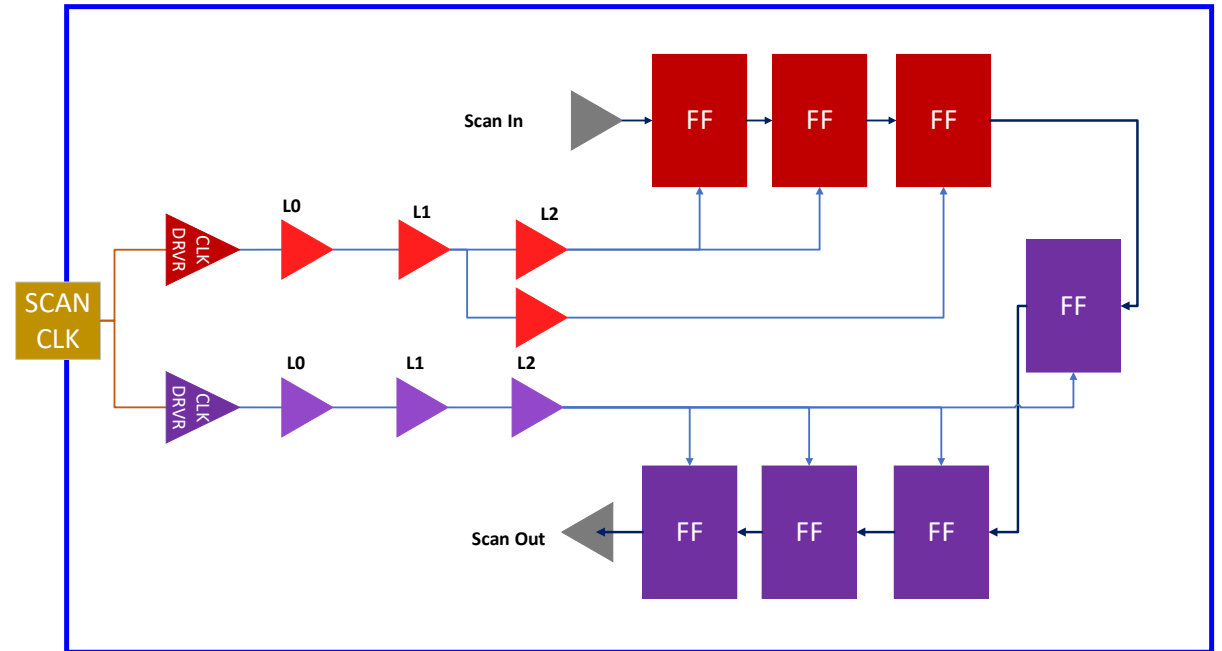
- Optimization across clock drivers
 - Reduced clock driver crossings and improved timing



Before



After





Experiment setup

Conditions

- Synopsys Fusion Compiler version:
 - T-2022.03-SP5-T-20230324
- 5 designs:
 - Single clock
 - Single power
 - 2 high utilization and macro dominated blocks
- Two optimization flows:
 - Native Clock Tree Aware Reorder
 - Custom Scripted Solution

Summary of results



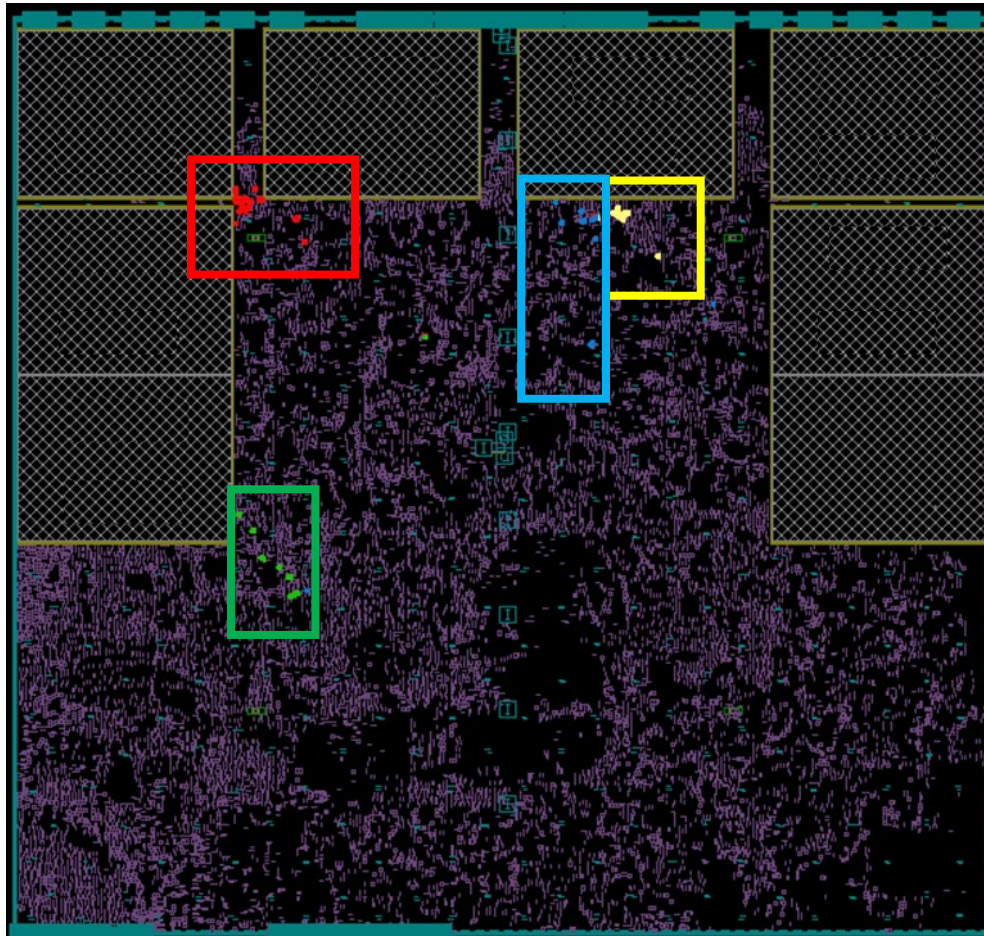
- Native Clock Tree Aware Reorder solution shows a tradeoff between wirelength and timing optimization
- Native Clock Tree Aware Reorder appears to show improved timing optimization for scan paths
- Native Clock Tree Aware Reorder seems to yield less clock driver cross connections

Scan Chain Optimization

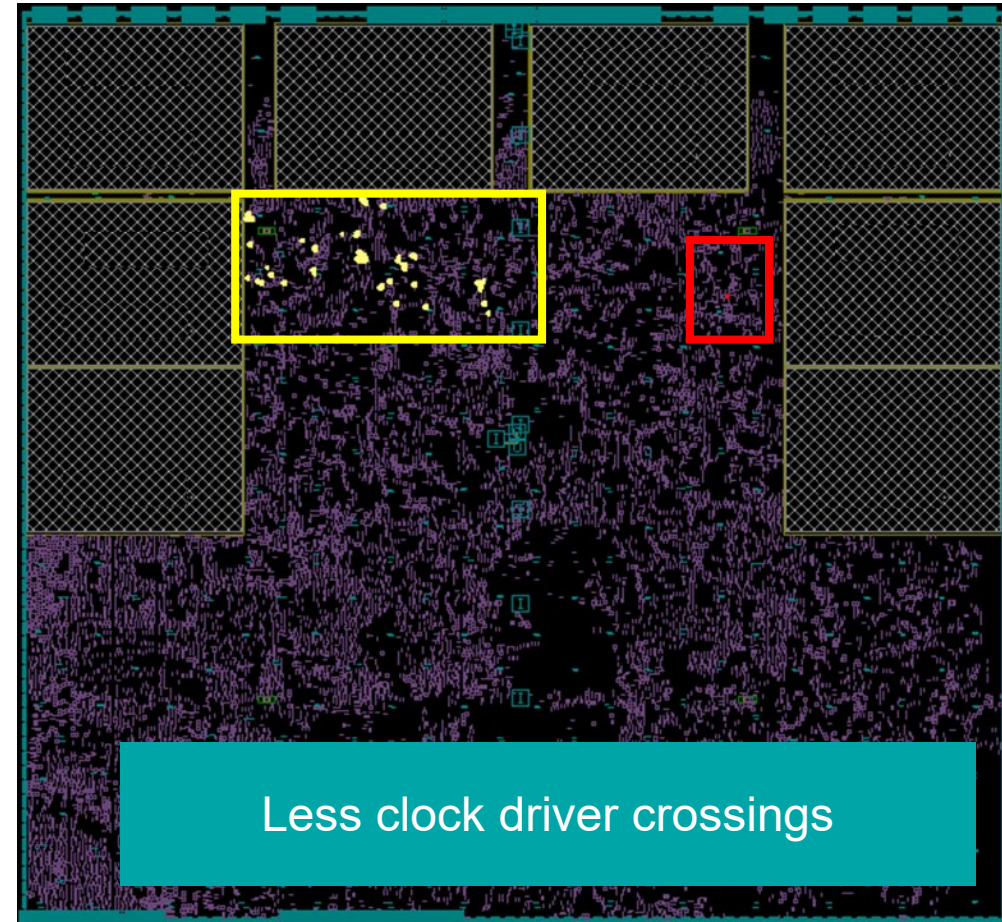
Clock Driver Crossing



Custom Scripted Solution

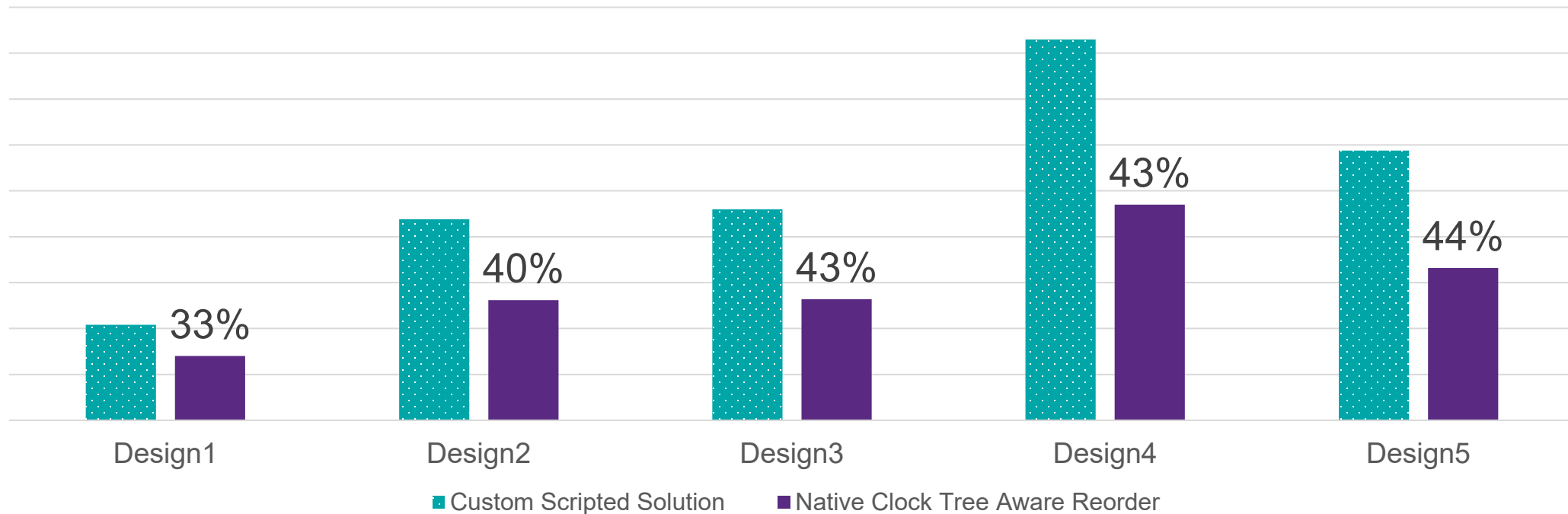


Native Clock Tree Aware Reorder



Timing Overhead Results: Clock Driver Crossings

Scan chain clock driver crossings after implementing clock tree aware reorder per design



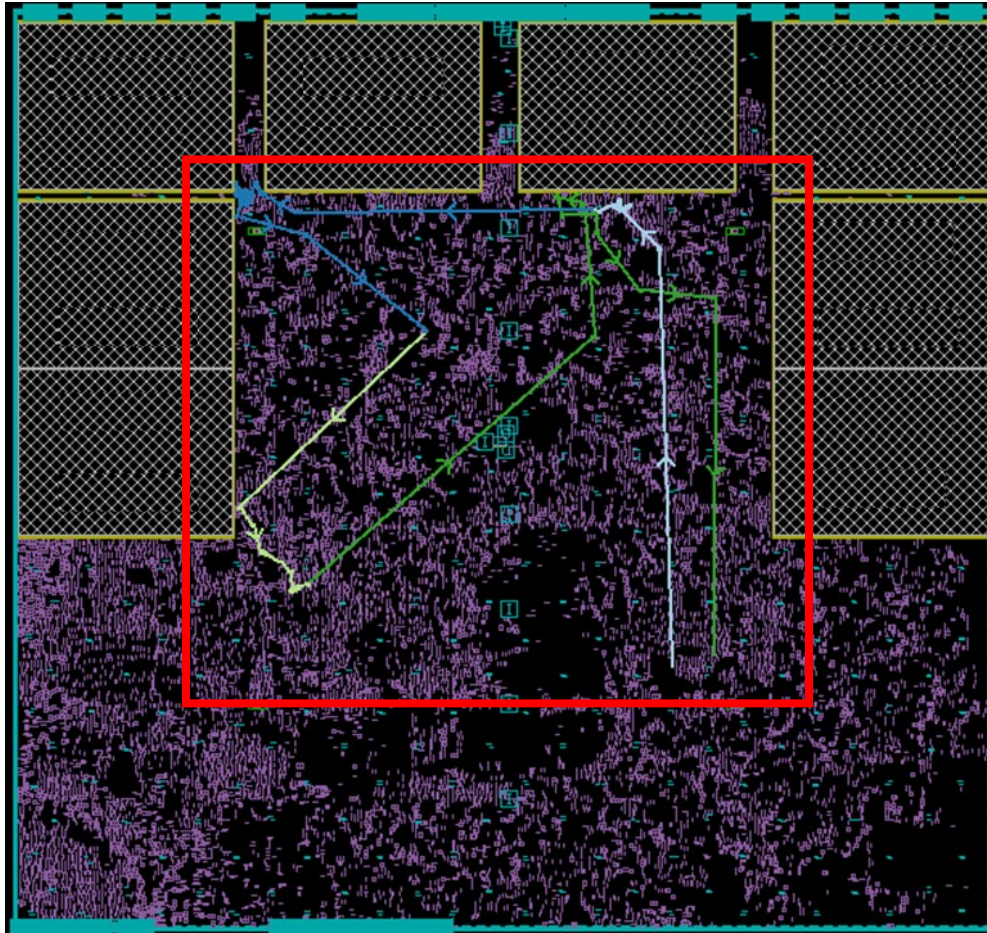
- Chain crossing reduction by 40% on average

Scan Chain Optimization

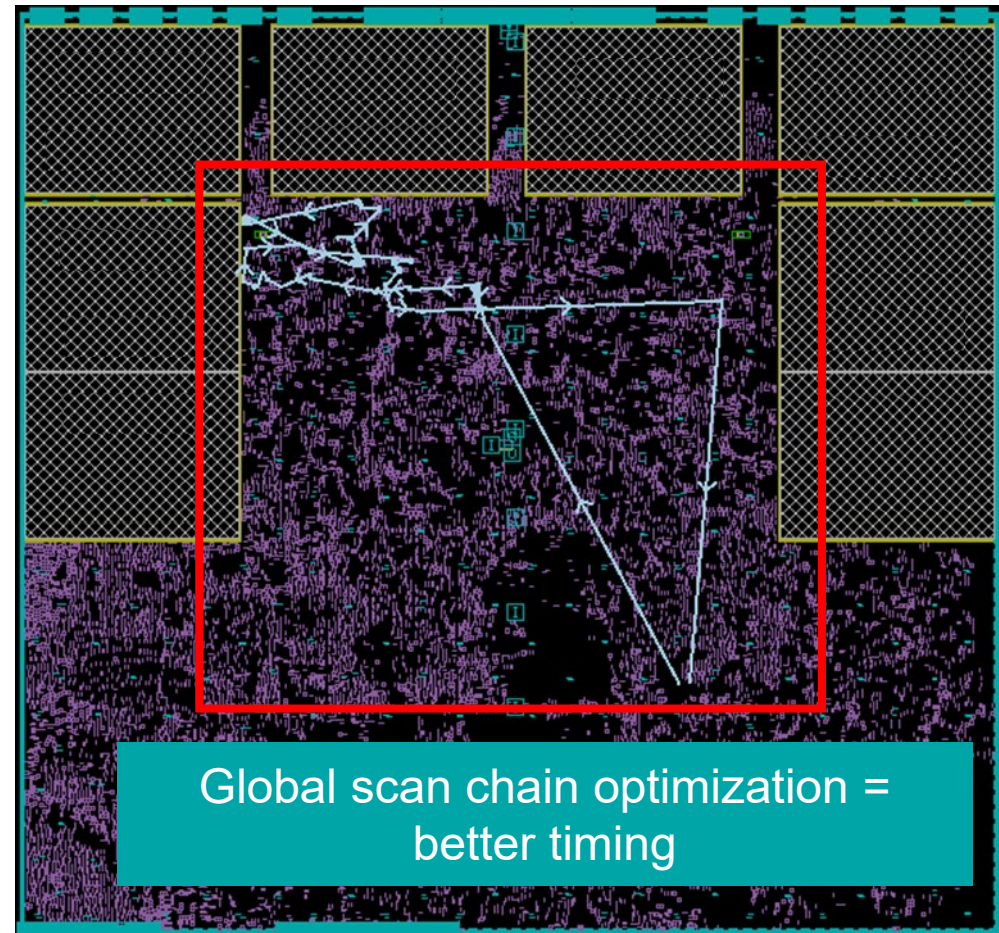


Min Timing

Custom Scripted Solution



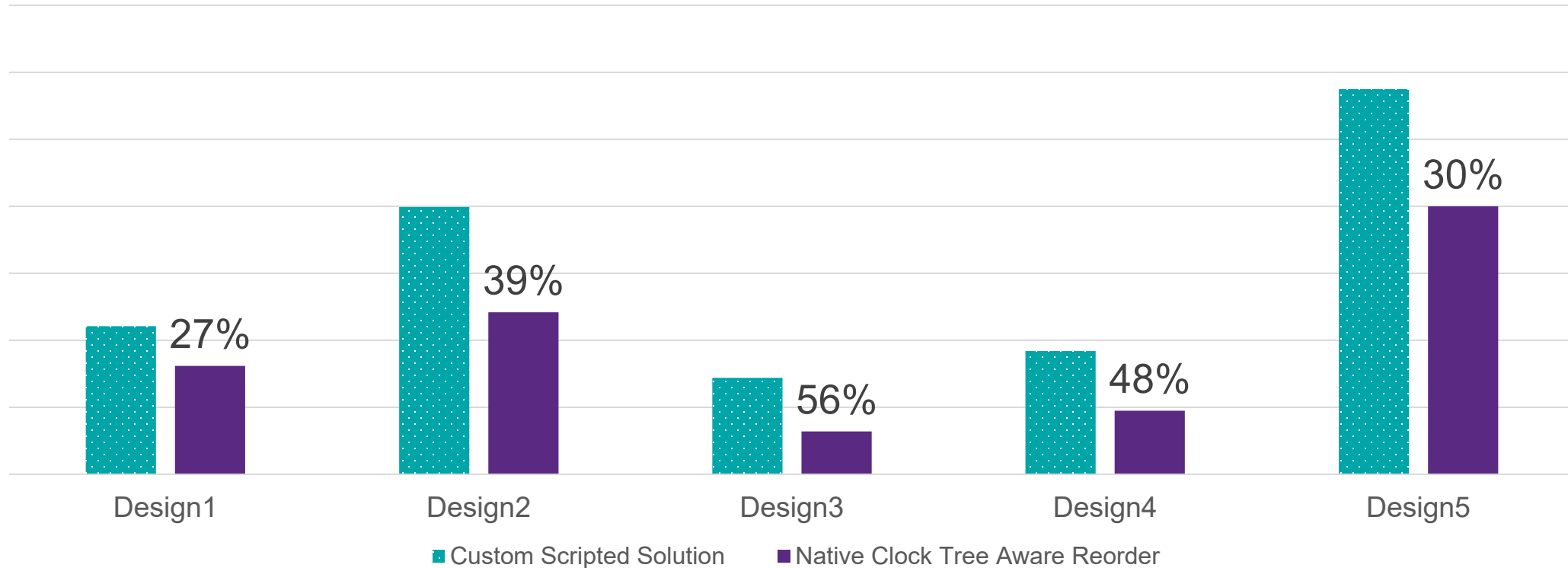
Native Clock Tree Aware Reorder



Global scan chain optimization =
better timing

Timing Overhead Results: Min Scan Paths

Scan path min delay violations per design



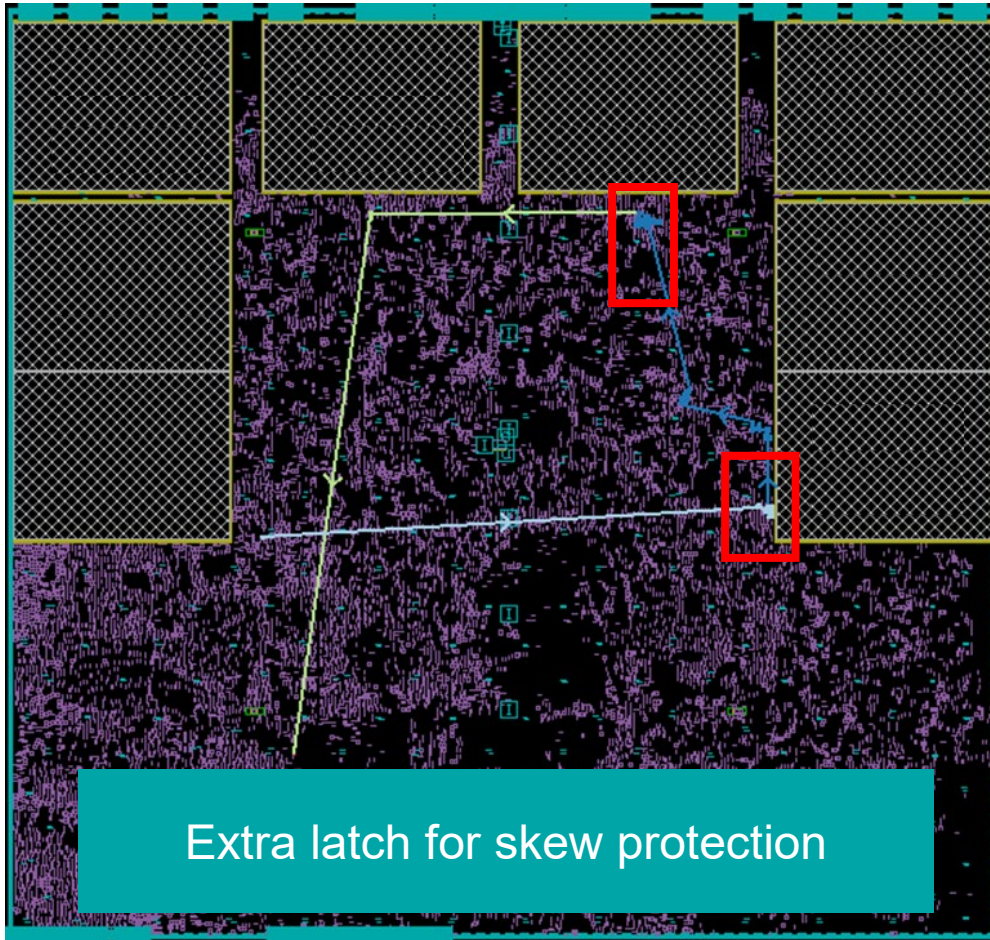
- Reduction of min scan paths up to 56%

Scan Chain Optimization

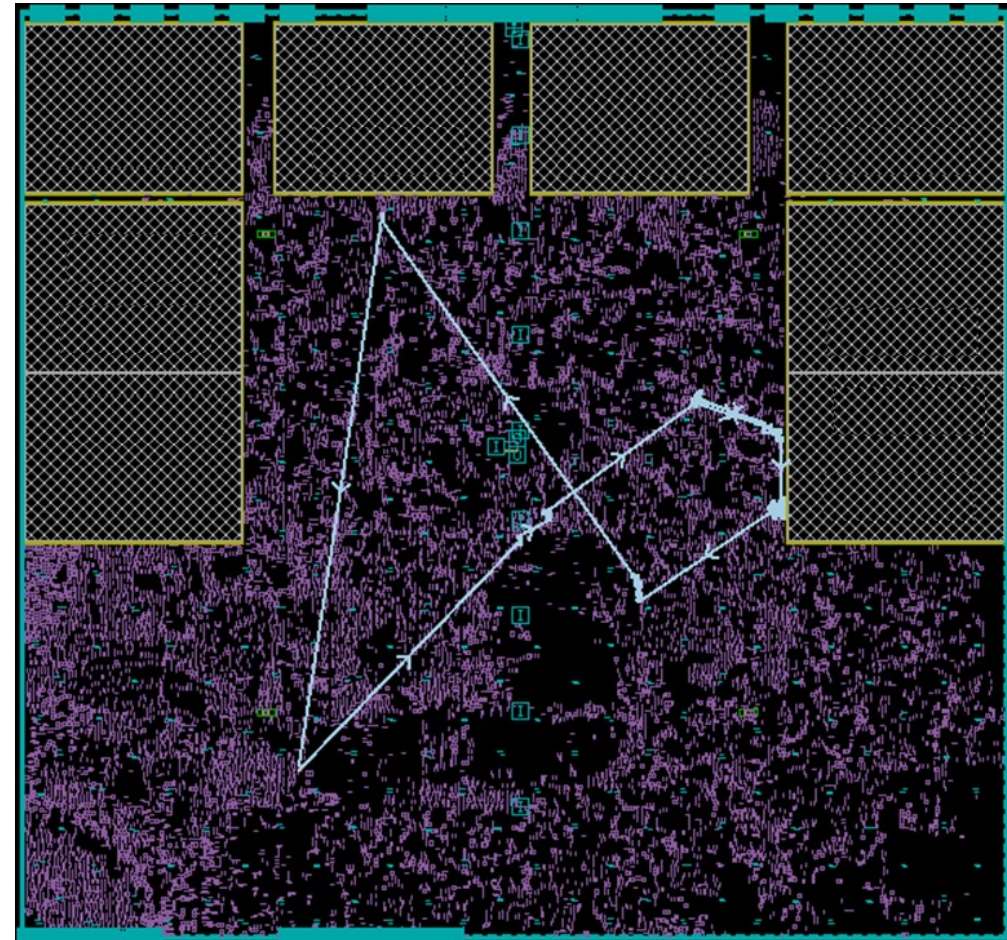


Cell Count

Custom Scripted Solution

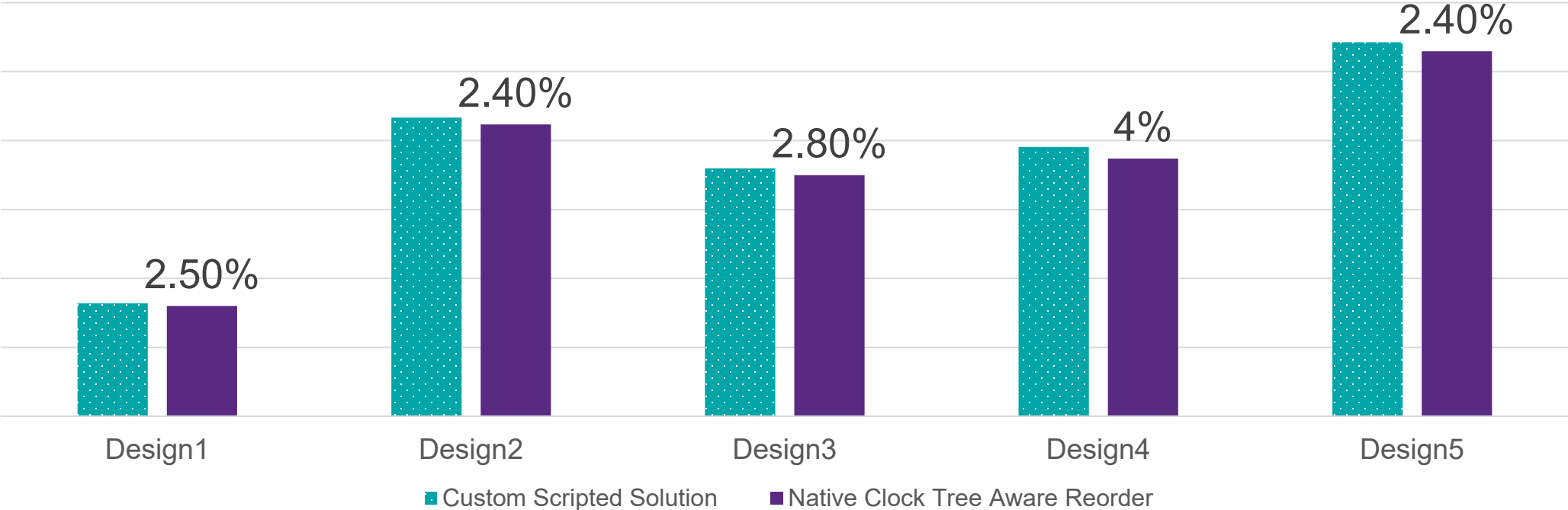


Native Clock Tree Aware Reorder



Design Metrics: Sequential Count

Scan sequential count after implementing clock tree aware reorder per design



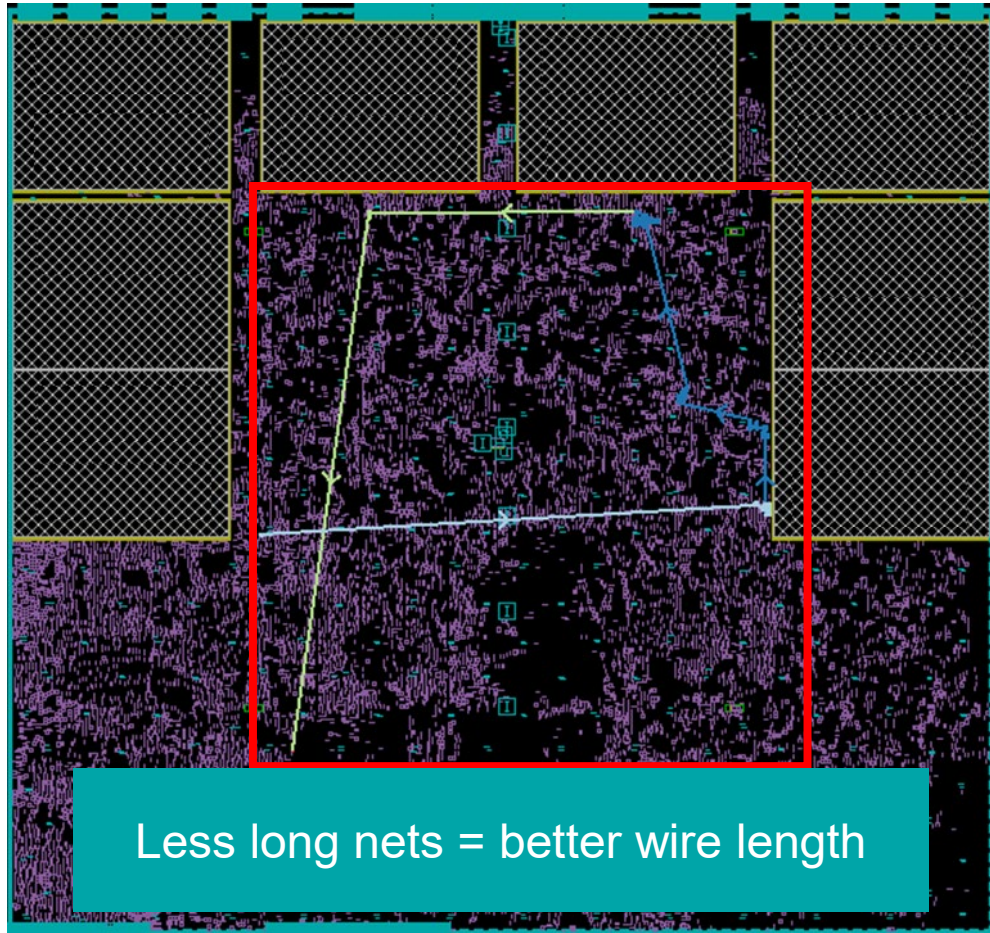
- Custom solution requires extra sequential cells to control the skew

Scan Chain Optimization

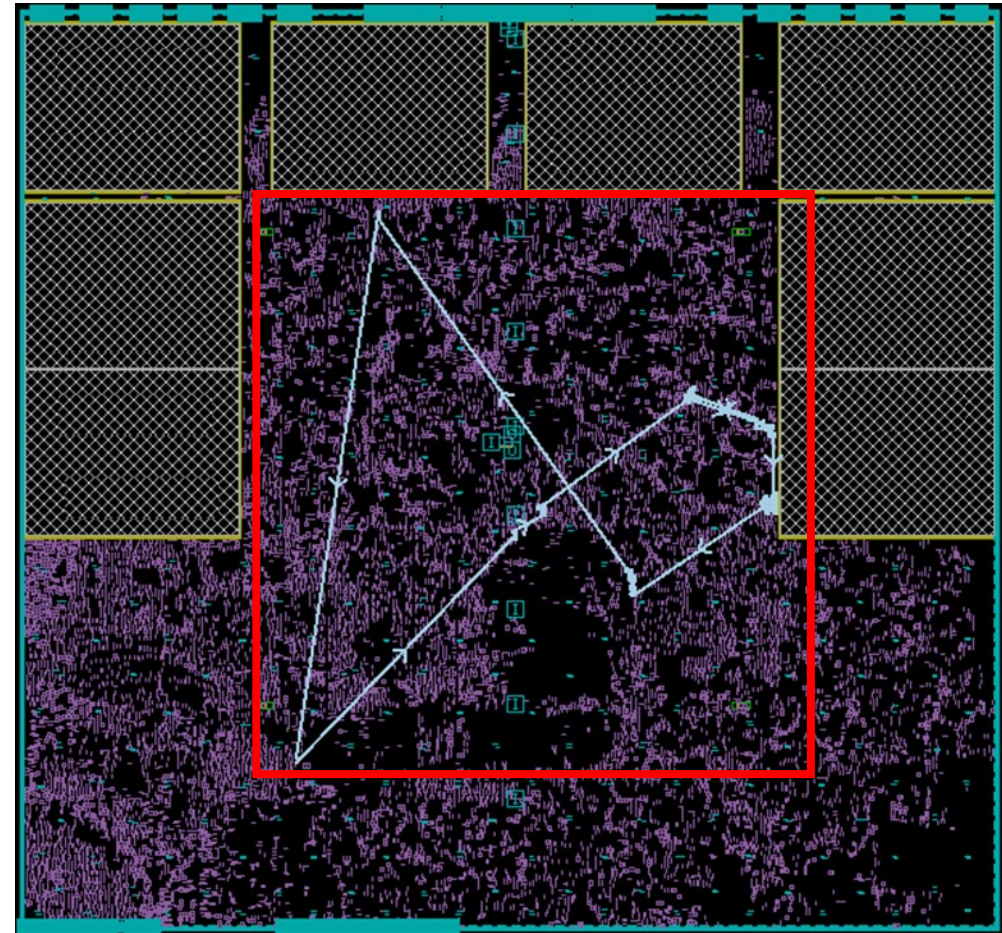


Wirelength

Custom Scripted Solution

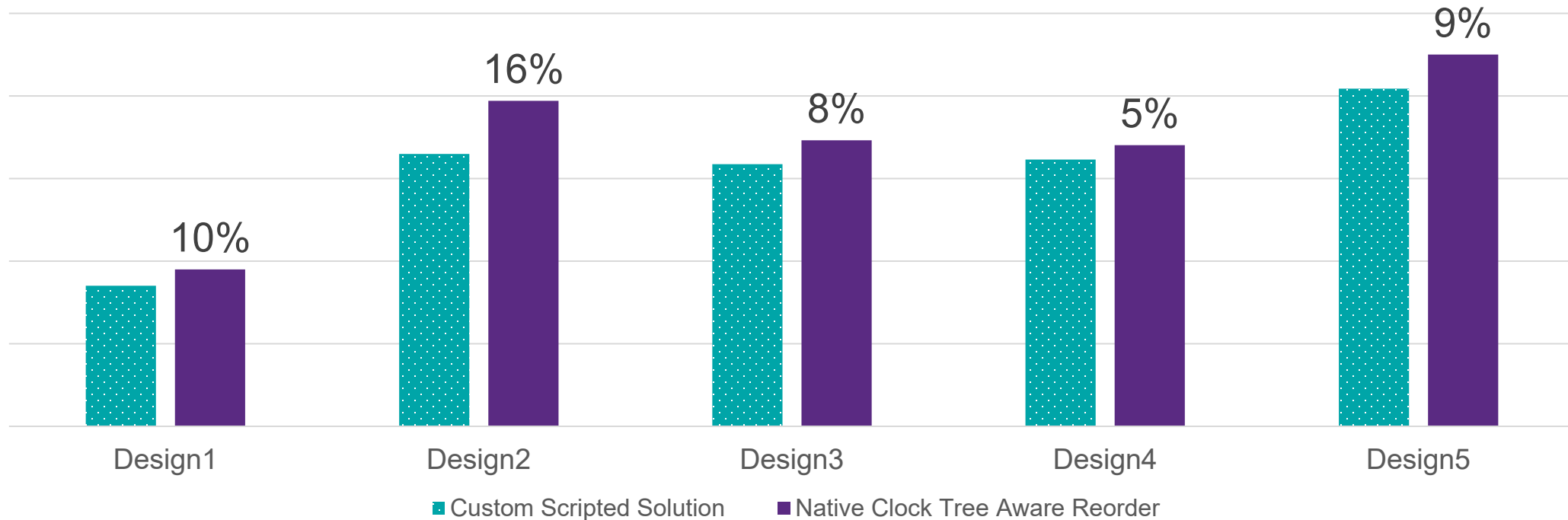


Native Clock Tree Aware Reorder



Design Metrics: Scan Chain Wirelength

Scan chain wirelength after implementing clock tree aware reorder per design



- Slight wire length degradation using native approach
 - More noticeably in high utilization blocks (Design1 and Design2)



Clock Tree Aware Scan Reorder

How to Implement It?

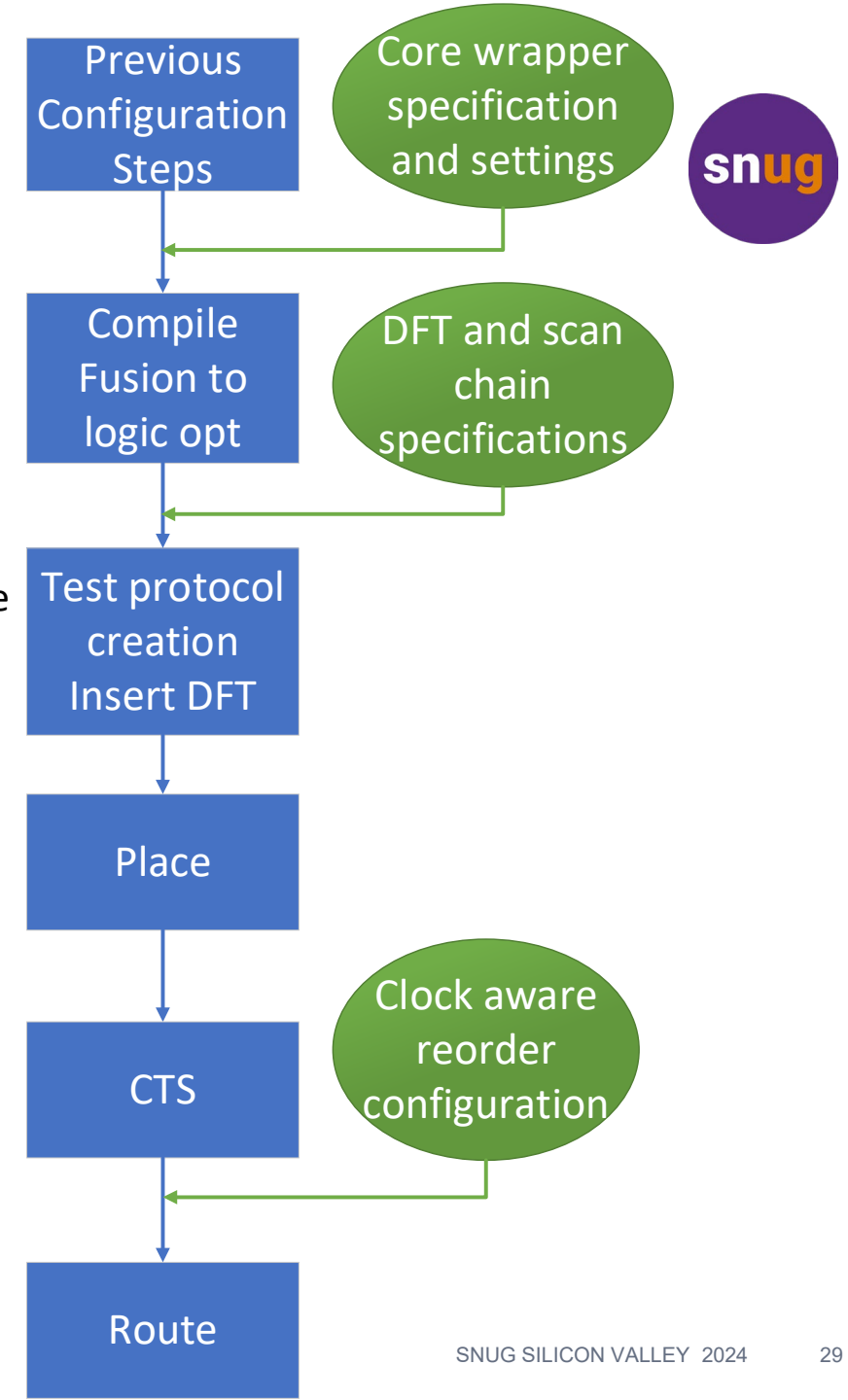
- App options:

- `set_app_options -name opt.dft.use_ng_engines -value true`
- `set_app_options -name opt.dft.clock_aware_skip_hold_check -value true`
- `set_app_options -name opt.dft.clock_aware_scan_repartition_reorder -value true`
- `set_app_option -name opt.dft.ng_clock_tree_aware -value true`
- `set_app_options -name opt.dft.repart_ng_integrated -value true`

- Clock_opt will call `optimize_dft` to optimize the scan chains.

- Feature is available for FC T-2020.03-SP5 versions and onwards

- Note: For newer T-SP5 and U branch FC versions, the behavior of the app option `opt.dft.ng_clock_tree_aware` changed to accept integers only (1 to 3), where 1 is leaf buffer-based reordering, 2 is clock tree aware reorder and 3 is a custom clock driver aware reorder.







Conclusions

- Native solution shows a **slight tradeoff** between timing convergence and wirelength
- Native solution **appears to improve** convergence over the custom scripted solution
- Can be **applied to any design** to reduce the scan timing closure overhead

Future Work



- Reduction of long scan paths 
- Further reduction of clock driver crossings 



THANK YOU

***YOUR
INNOVATION
YOUR
COMMUNITY***

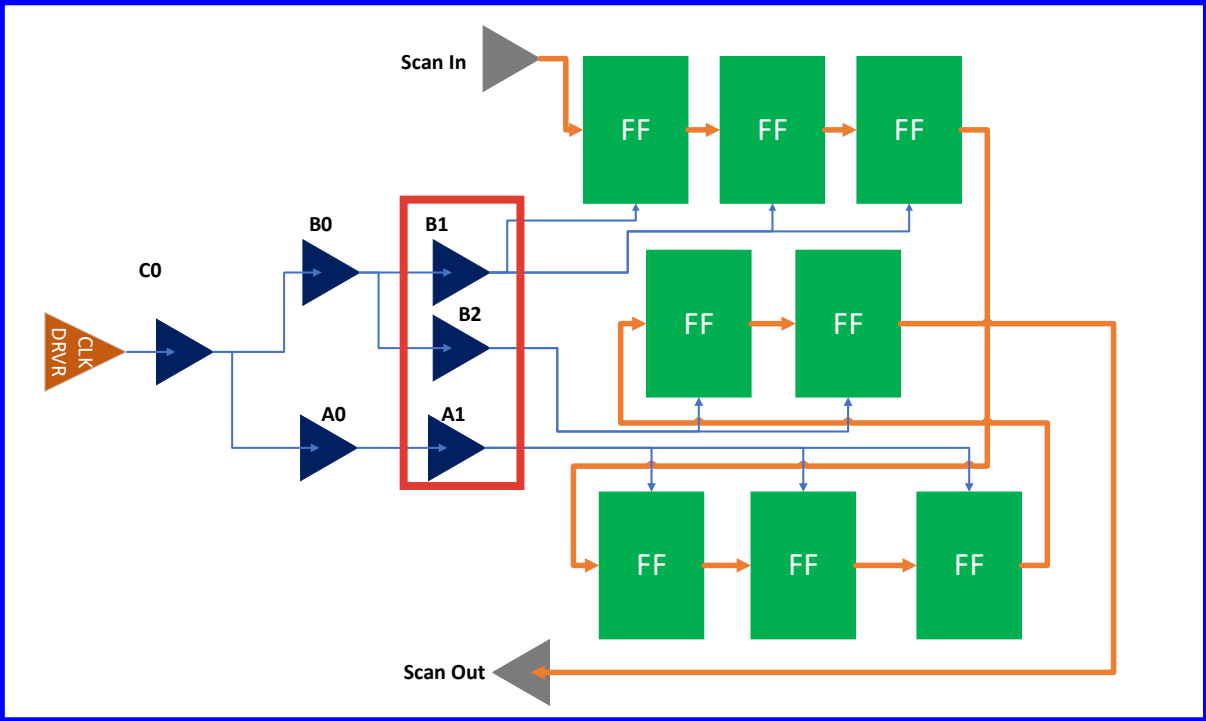


Native Clock Tree Aware Reorder

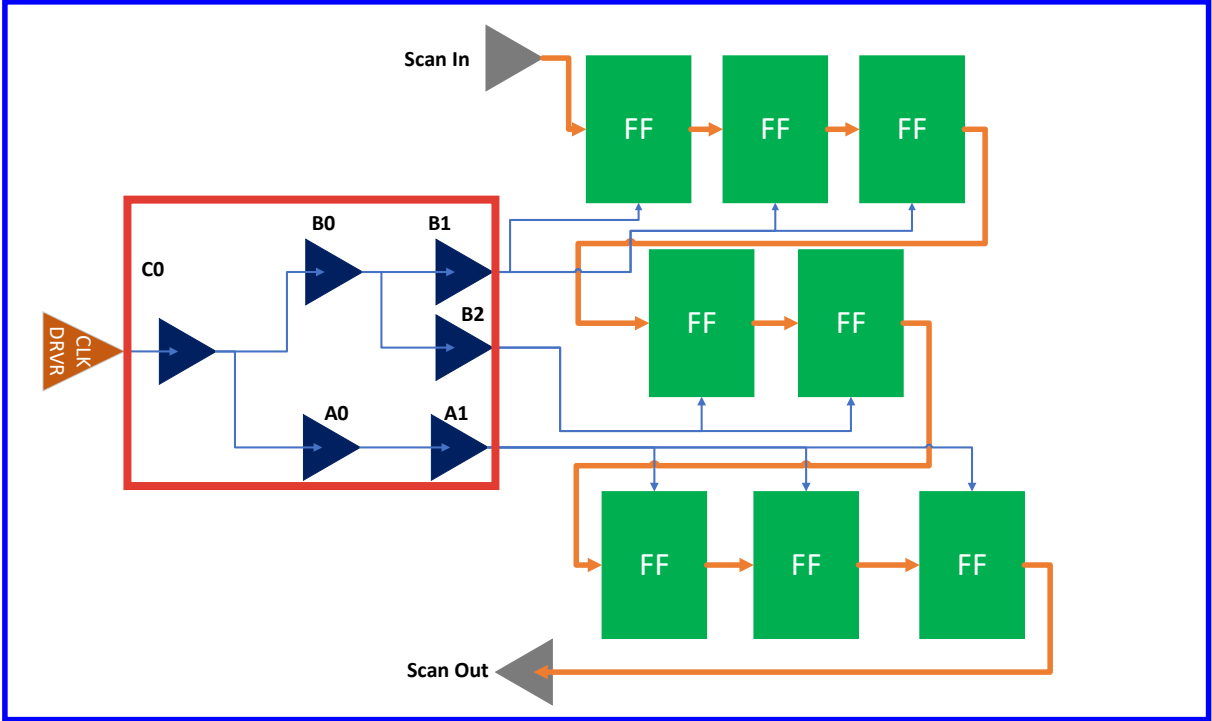
Difference between previous and native clock tree aware techniques

- Previous technique, limits the cross-clock driver optimization

- Previous Reorder Technique



- Native Clock Tree Aware Reorder



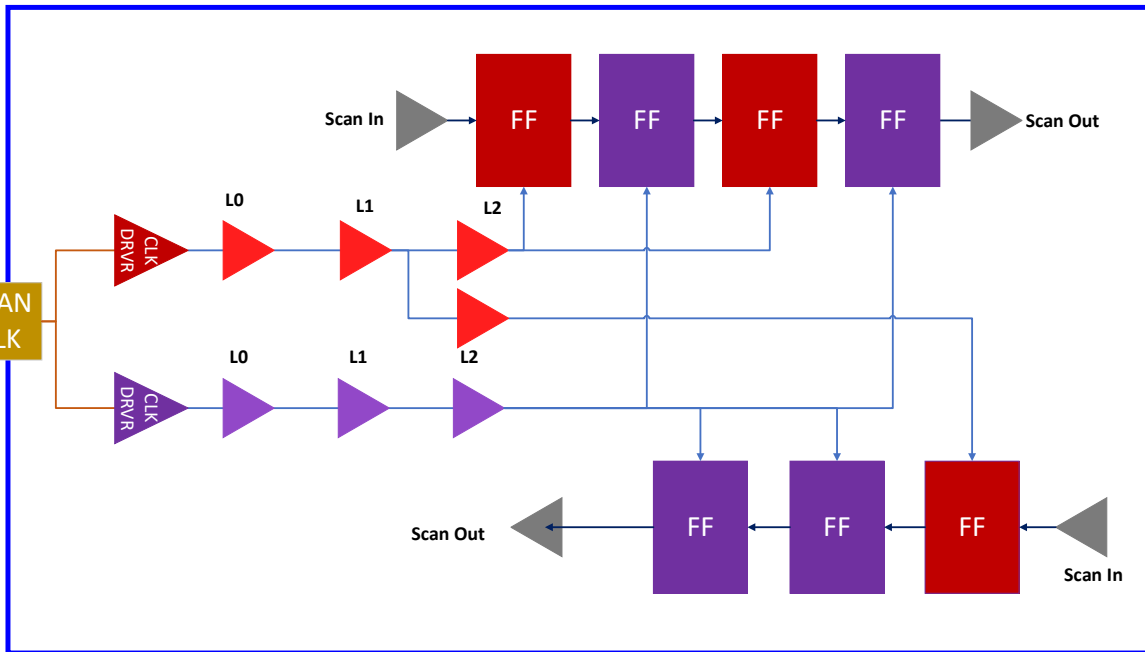
Native Clock Tree Aware Reorder

Native Optimization Technique

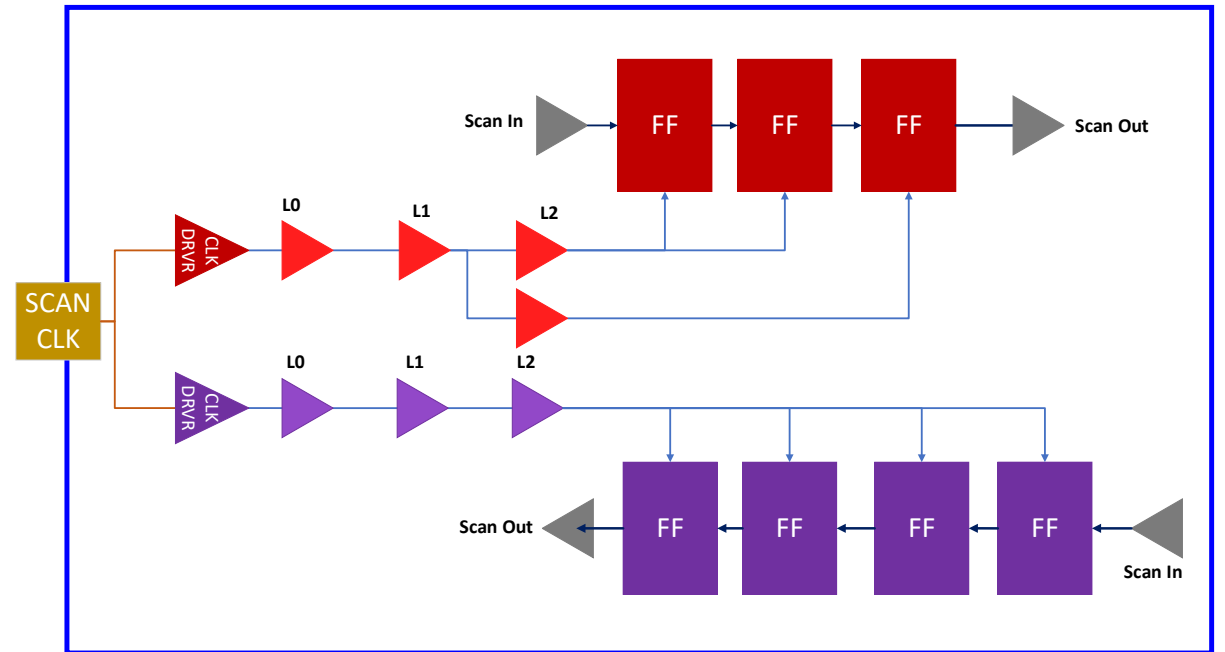
- Optimization across scan chain

 :Clock Driver

Before

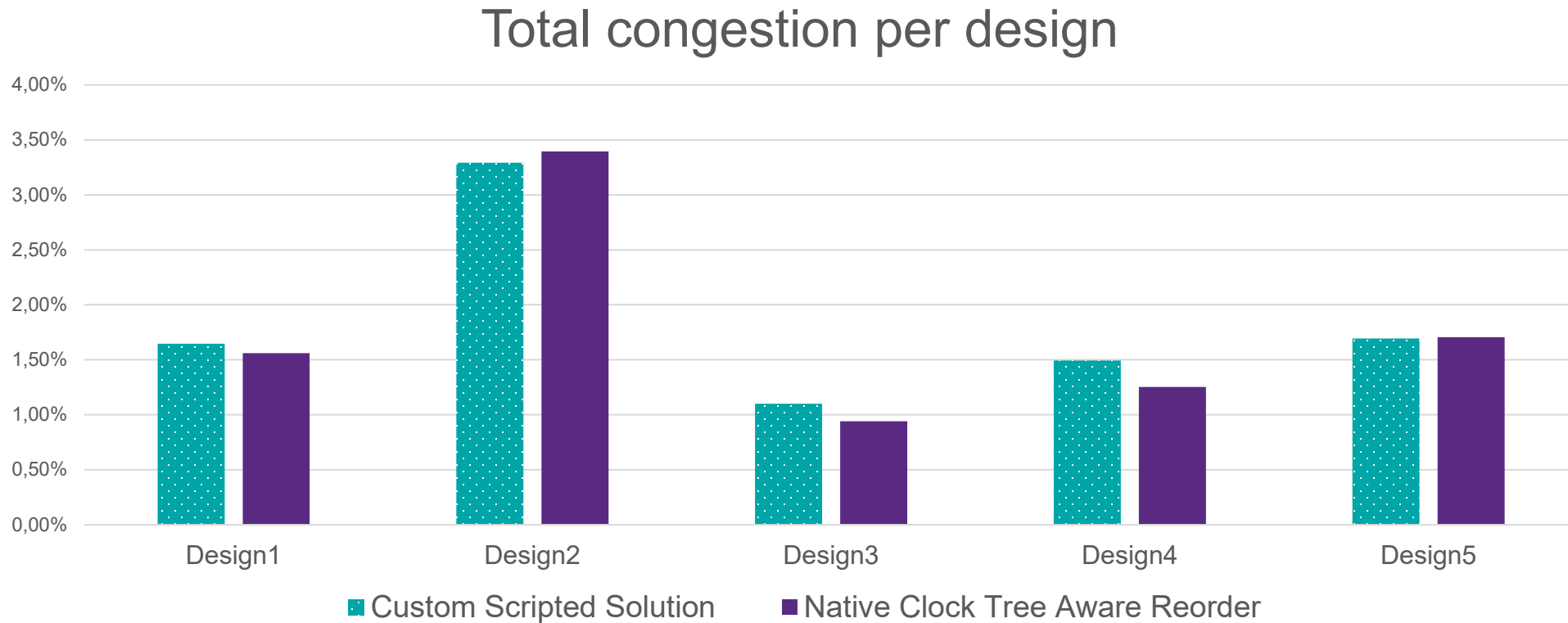


After



Routing Overhead Global Results: Congestion

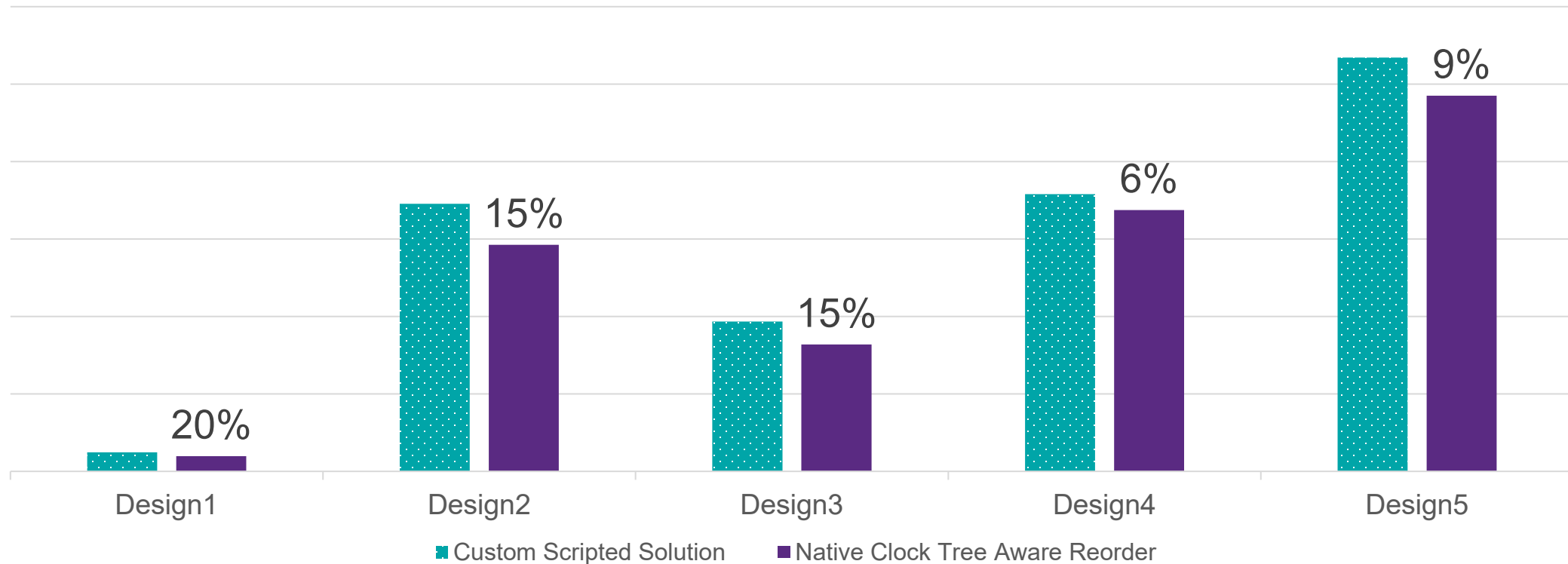
Congestion



- No major route/congestion impact seen

Design metrics: Total Hold Buffering

Total hold buffering per design



- Higher hold buffering with custom solution