Improve CTS QoR by H-tree-only (Non-Mesh) Regular MSCTS for complex floorplans and notches design

Presenter: Luan Pham - Ngoc Le

Author: Luan Pham – Ngoc Le – Phuong Le

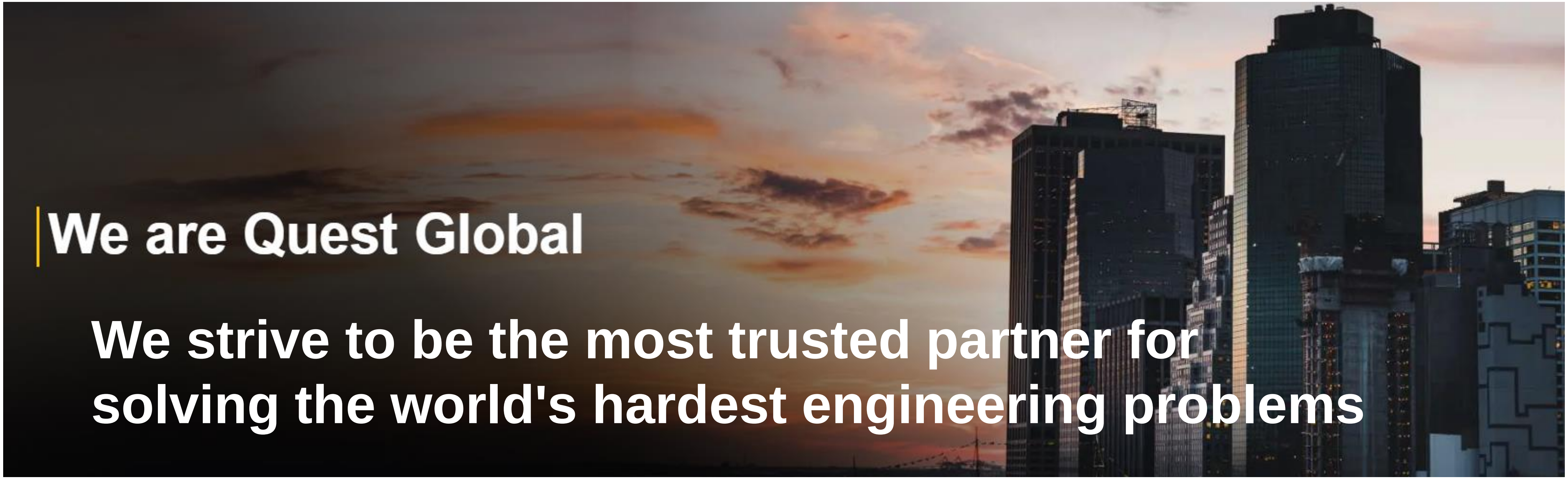
Quest Global Vietnam

Agenda

1. About Quest Global
2. Abstract
3. Problem statement
4. Algorithm htree_sections of tool FC version U 2022
5. Result
6. Conclusions
7. Reference

About Quest Global

About Quest Global



We are Quest Global

We strive to be the most trusted partner for solving the world's hardest engineering problems

Who we serve

Aerospace and Defense

Automotive

Communications

Energy

Hi-Tech

MedTech and Healthcare

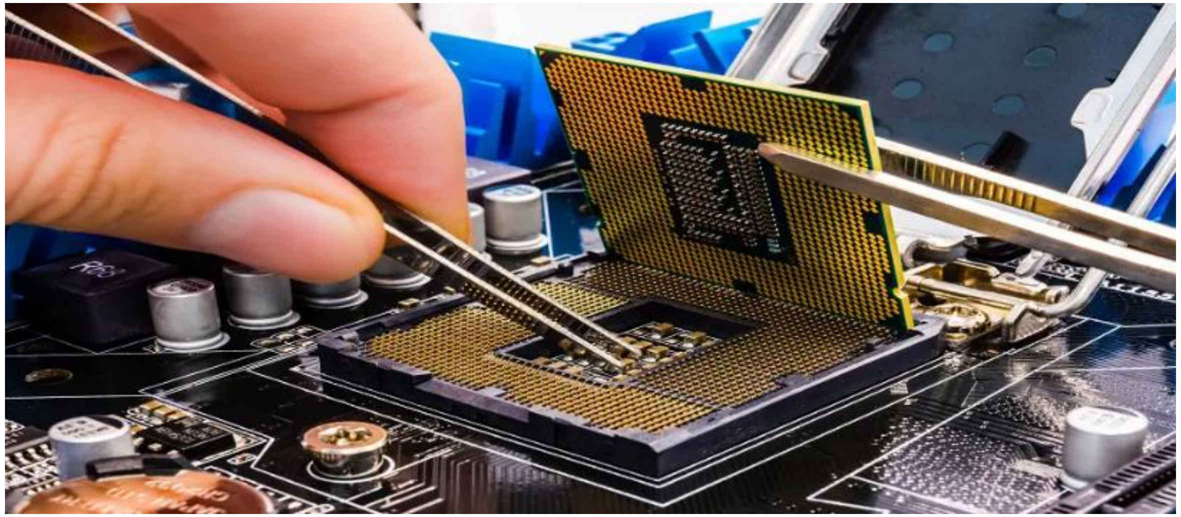
Rail

Semiconductors

Semiconductors

Quest global

snug



Silicon Engineering and Platform Engineering

Highlights

Engagements with 7 of Top-10 Semiconductor companies	**300+ Tape-outs from 2016 onwards 65nm to 3nm**	**>60% of Semiconductor Engineers work on 7nm or later**	**25+ Tape-outs in 5/4/3nm**

Partners

NVIDIA | Preferred Partner

arm APPROVED DESIGN PARTNER

NXP Registered Partner

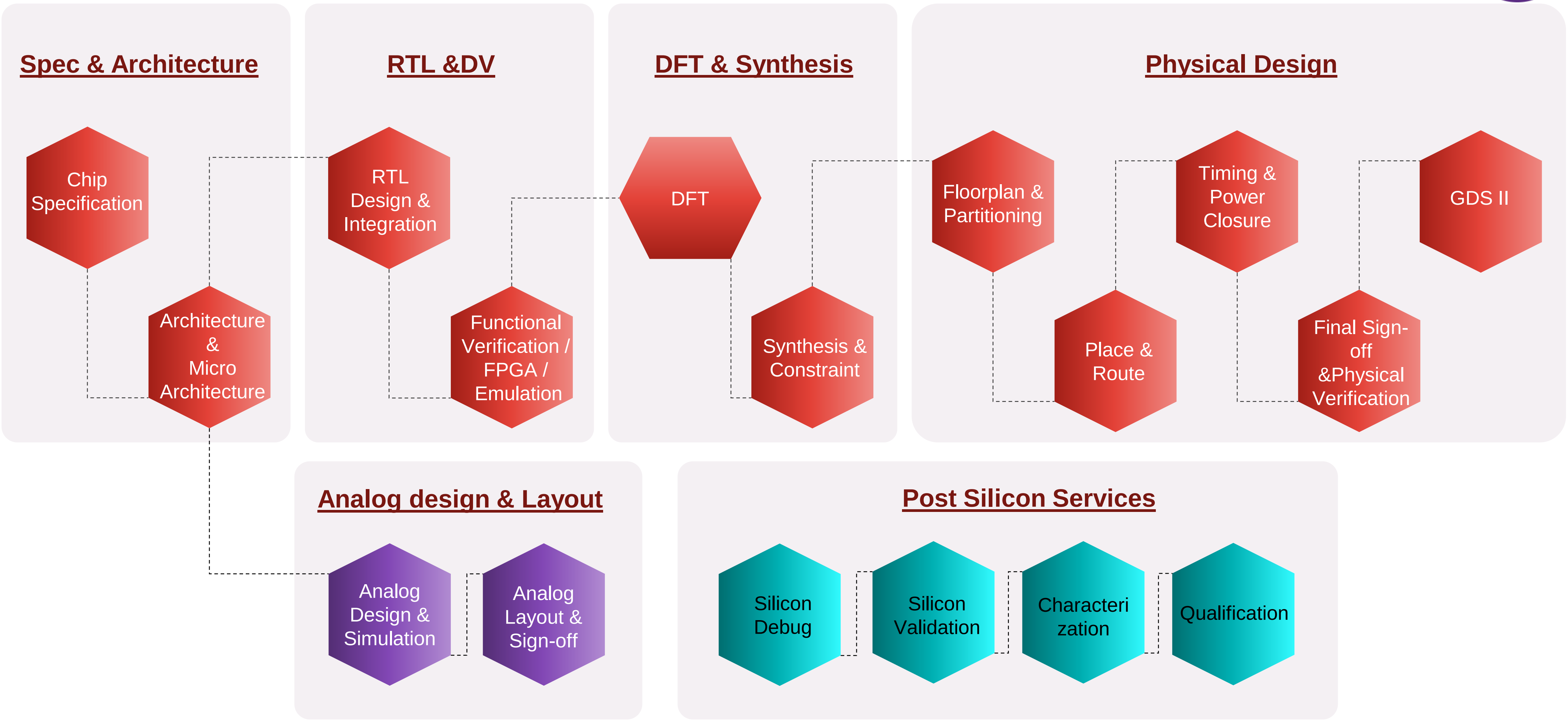
RENESAS R-Car Consortium

tsmc ®

IP Ecosystem Partr

XILINX. ALLIANCE PROGRAM MEMBER

End-to-end Semiconductor Capabilities



Spec & Architecture

Chip Specification

Architecture & Micro Architecture

RTL &DV

RTL Design & Integration

Functional Verification / FPGA / Emulation

DFT & Synthesis

DFT

Synthesis & Constraint

Physical Design

Floorplan & Partitioning

Place & Route

Timing & Power Closure

Final Sign-off &Physical Verification

GDS II

Analog design & Layout

Analog Design & Simulation

Analog Layout & Sign-off

Post Silicon Services

Silicon Debug

Silicon Validation

Characteri zation

Qualification

Abstract

Section Subtitle

Abstract

- In VLSI (Very Large Scale Integration) design, a clock plays a crucial role in synchronizing the operations of various components within a digital circuit. And various clock structures or methodologies are used to implement the clock scheme efficiently.
- This paper introduces H-tree-only Regular Multisource Clock Trees (MSCTS) with htree_sessions, which is new feature of Fusion Compiler (FC) aimed at simplifying the implementation. H-tree-only MSCTS offer a streamlined approach to clock distribution, particularly in complex floorplans and notches, where traditional methods may encounter challenges. Beside, the **htree_sessions feature**, which is available from version **U-2022.12-SP3**, provides a framework for automating the generation of H-tree structures and multiple clock sources tailored to diverse design requirements. This feature ensures efficient clock signal distribution while minimizing timing skew and increase CRPR, especially in high complexity floorplan. Through testing and validation, this paper demonstrates the effectiveness and ease of implementation of H-tree-only Regular MSCTS using the htree_sessions feature, offering a robust solution for clock distribution in a CPUs high-performance computing, complex floorplan and high density operating at 1.5GHz.

Problem statement

Section Subtitle

Challenging Points

1

Many designs have notch complex floorplan,
So, Designers face difficulty in improving the Quality of Results (QoR) at CTS

Designing clock networks manually in complex floorplans can be quite challenging due to various layout constraints such as routing congestion, timing closure, and signal integrity issues.

2

Improve QoR at CTS needs experienced engineer

Improving QoR CTS (Optimal Routing Clock Tree Synthesis) indeed requires a deep understanding of clock tree synthesis techniques, timing closure, and physical design challenges

3

Manual implementation of clock building consumes significant amount of time

Building a clock manually can indeed be a time-consuming task, especially if you're crafting it from scratch with intricate details. From designing the mechanism to assembling the components, it requires precision and patience

Using H-tree-only (Non-Mesh) Regular MSCTS

1

This approach can improve the Quality of Results (QoR) in CTS

With this feature, user can build multiple global trees at different parts of the floorplan and achieve better clock QoR especially latency and skew.
With this feature, FC tools to generate initial clock tree structures based on the floorplan constraints. These tools can help in achieving timing closure and optimizing the clock distribution network.

2

This feature can save time for engineers and also saves human resources

This feature saves time and effort for engineers and helps ensure the optimal performance of the clock distribution network.

Can save human resources through its design simplicity and potential for automation

3

This technique can be automated to build clocks efficiently

The feature automatic insertion of tap drivers in H-tree-only Regular MSCTS streamlines the clock tree synthesis process by efficiently balancing timing, power, and area considerations while adhering to design constraints.

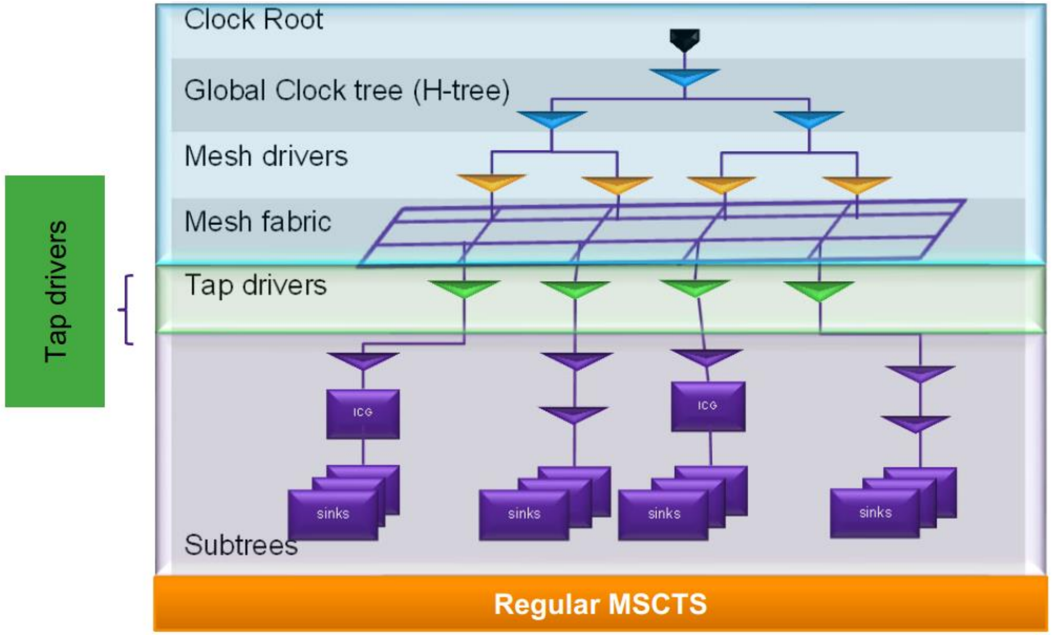
Algorithm htree_sections of tool U 2022

Section Subtitle

Introduction to Multisource Clock Tree Synthesis (MSCTS)

Multisource CTS, or Multisource Clock Tree Synthesis, indeed presents an innovative approach to clock distribution technology. Traditionally, clock distribution in integrated circuits has relied on a single source for propagating the clock signal throughout the chip. However, as chip designs become more complex and demand higher performance, conventional clock distribution methods encounter challenges such as clock skew, jitter, and power consumption.

Multisource CTS addresses these challenges by introducing multiple sources for clock distribution strategically placed across the chip. By distributing the clock signal from multiple points, Multisource CTS mitigates issues related to skew and jitter, resulting in improved timing, reduced power consumption, and enhanced performance scalability.



Flow of H-tree only (Non-Mesh) Regular MSCTS

1

Tap Drivers

These are specialized buffers typically inserted at the root nodes of each H-tree, connecting directly to the clock source.

Their primary function is to isolate the H-tree from variations in the clock source itself.

2

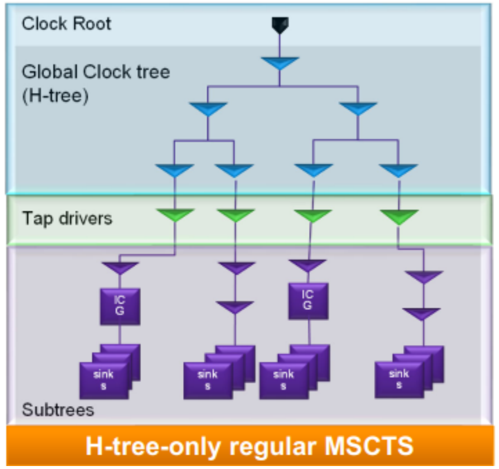
Build global clock tree (H-tree)

Once the buffers and taps are inserted, the next step is to route the clock nets from the sources to the Tap Driver

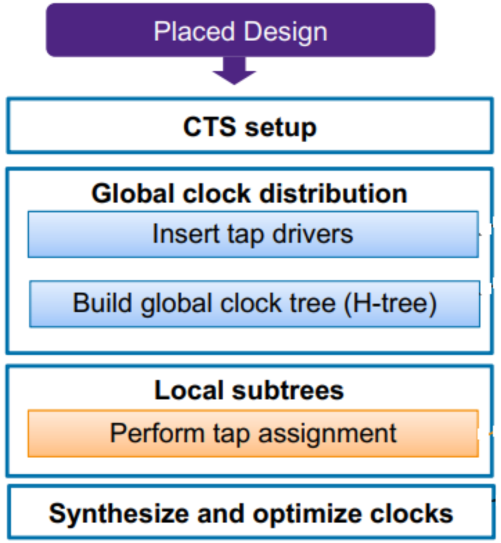
3

Perform tap assignment

Utilize an algorithm to assign tap drivers to the selected candidate locations. The tap assignment algorithm aims to optimize clock skew, minimize insertion delay, and balance the load across the clock network.



H-tree-only regular MSCTS



Placed Design

CTS setup

Global clock distribution

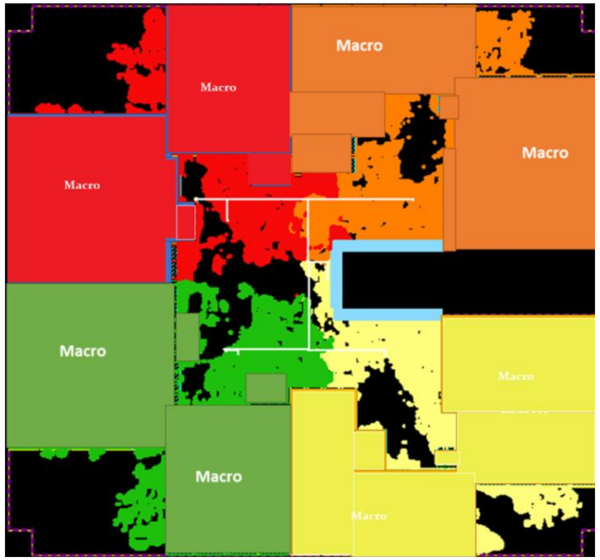
Insert tap drivers

Build global clock tree (H-tree)

Local subtrees

Perform tap assignment

Synthesize and optimize clocks



Algorithm htree_session of FC tool version U 2022

1

set_regular_multisource_clock_tree_options –htree_sections

User can provide “tap configs along with section boundary” or tap locations directly. The H-tree structure will be divided into segments or subsections, possibly to optimize the clock tree for specific regions of the chip or to address routing constraints more effectively.

2

synthesize_regular_multisource_clock_trees

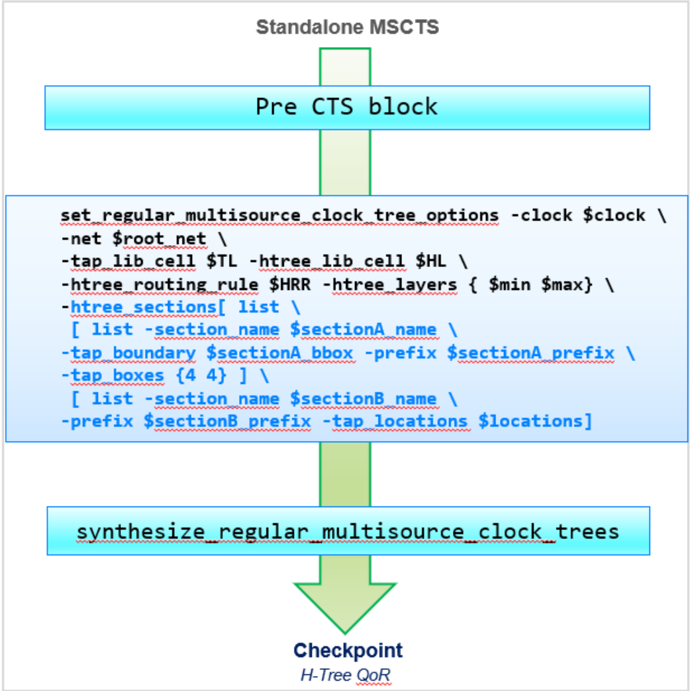
– Derives boundary based on sink distribution and insert the tap drivers based on inputs

– Explores different tap driver locations for H-tree compatibility and inserts them

– Builds symmetric H-tree

Checkpoints :

- Htree QoR after global tree



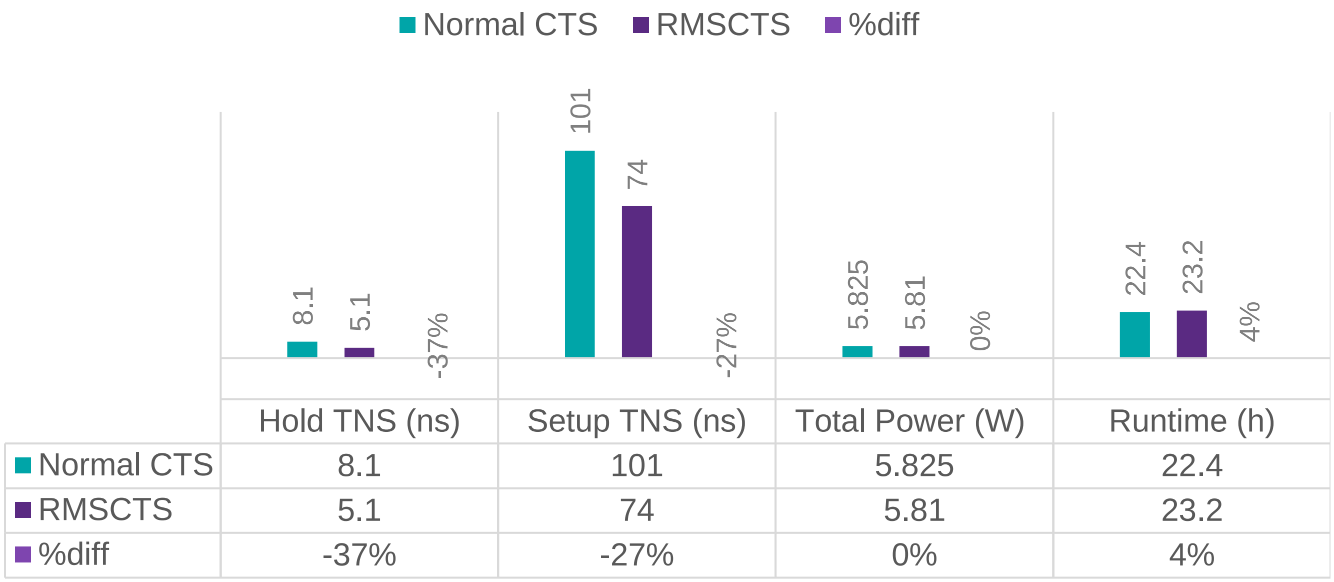
```
set_regular_multisource_clock_tree_options -clock $clock \
-net $root_net \
-tap_lib_cell $TL -htree_lib_cell $HL \
-htree_routing_rule $HRR -htree_layers { $min $max} \
-htree_sections[ list \
 [ list -section_name $sectionA_name \
-tap_boundary $sectionA_bbox -prefix $sectionA_prefix \
-tap_boxes {4 4} ] \
 [ list -section_name $sectionB_name \
-prefix $sectionB_prefix -tap_locations $locations]
```

Result

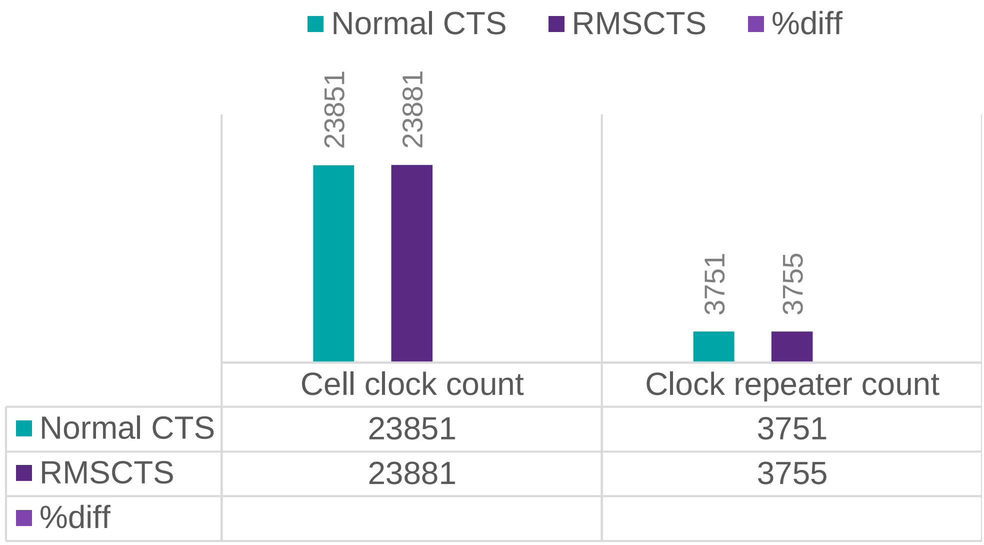
Section Subtitle

Result

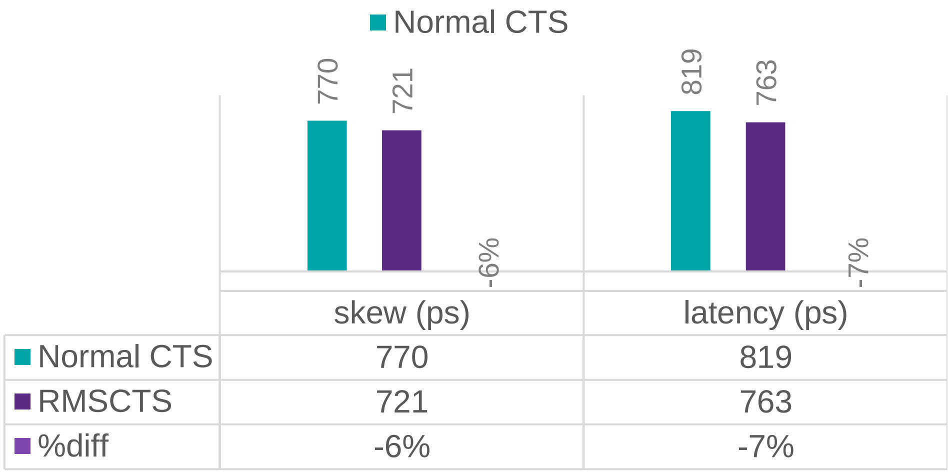
Compare QOR **CTS** RESULT: Normal CTS and H-tree-only RMSCTS use **htree_section**



	Hold TNS (ns)	Setup TNS (ns)	Total Power (W)	Runtime (h)
■ Normal CTS	8.1	101	5.825	22.4
■ RMSCTS	5.1	74	5.81	23.2
■ %diff	-37%	-27%	0%	4%



	Cell clock count	Clock repeater count
■ Normal CTS	23851	3751
■ RMSCTS	23881	3755
■ %diff		



	skew (ps)	latency (ps)
■ Normal CTS	770	819
■ RMSCTS	721	763
■ %diff	-6%	-7%

Comment: Setup and hold timing are improved so much:

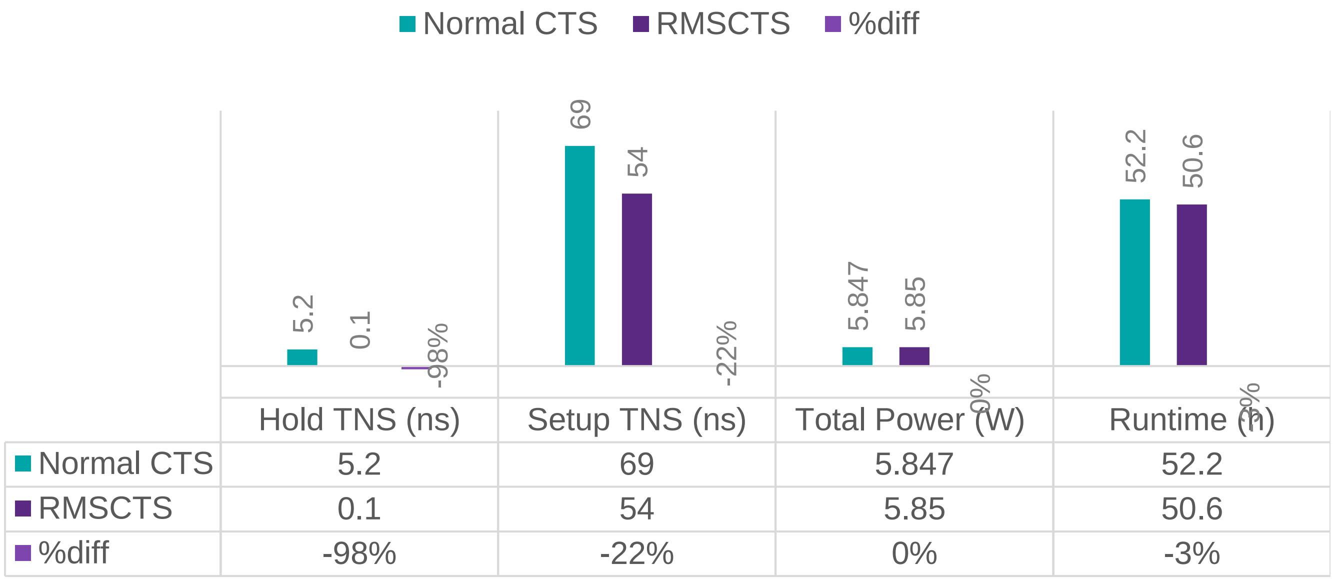
- Setup : -27%

- Hold : -37%

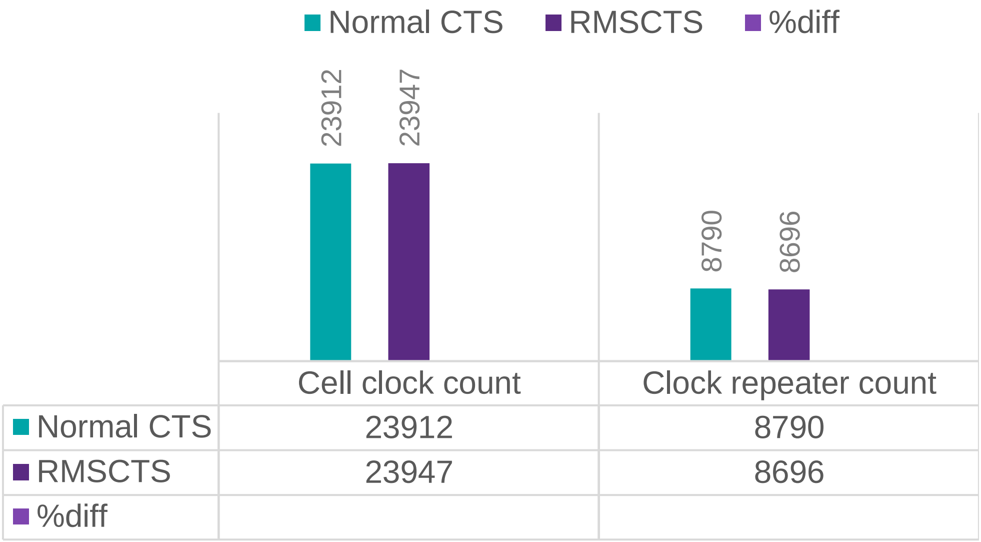
- Skew and latency are reduce about 6%.
- Total power is same.

Result

Compare QOR **ROUTE_OPT** RESULT: Normal CTS and H-tree-only RMSCTS use **htree_section**



	Hold TNS (ns)	Setup TNS (ns)	Total Power (W)	Runtime (h)
■ Normal CTS	5.2	69	5.847	52.2
■ RMSCTS	0.1	54	5.85	50.6
■ %diff	-98%	-22%	0%	-3%



	Cell clock count	Clock repeater count
■ Normal CTS	23912	8790
■ RMSCTS	23947	8696
■ %diff		

Comment: Setup and hold timing are improved so much:

- Setup : -22%

- Hold : -98%

- Skew and latency are reduce about 1%.
- Runtime reduce -3% after PnR
- Total power is same.

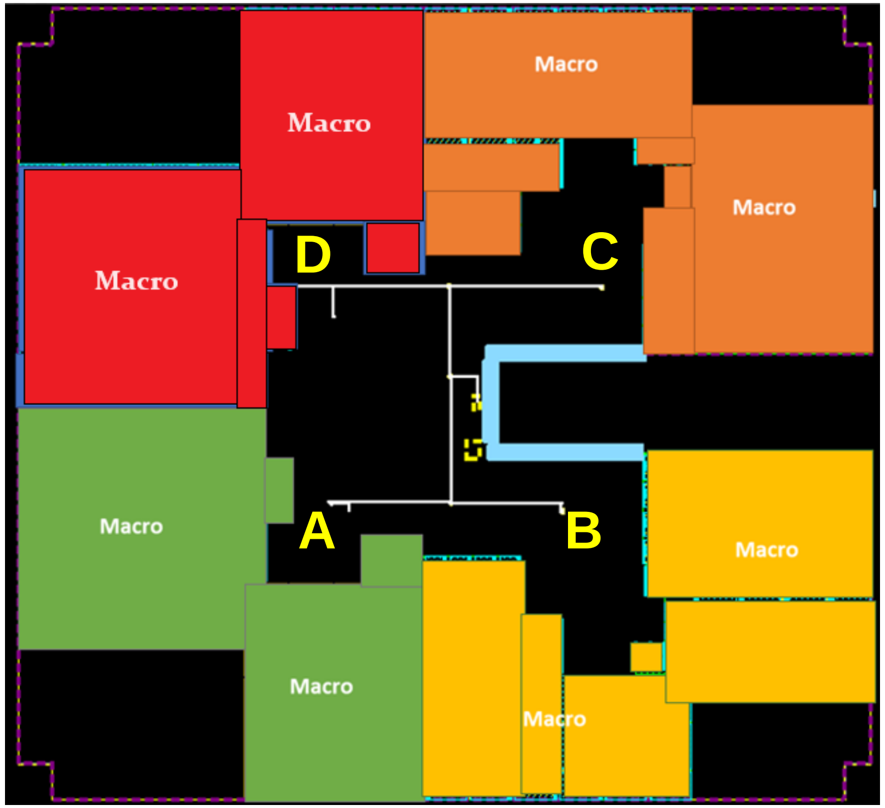
Result

- **Define regular MSCTS settings**

```
set net CKnet_name ; # set net need to build H-tree-only Regular MSCTS
set clk_name clka ; # set main clock to build H-tree-only Regular MSCTS

#Define regular MSCTS settings
set_regular_multisource_clock_tree_options \
                    -clock $clk_name \
                    -topology htree_only
                    -prefix MSCTS
                    -net [get_nets $net] \
                    -tap_lib_cells $CKbuf \
                    -htree_routing_rule $htree_ndr \
                    -htree_lib_cells $CKbuf
                    -htree_layers "M12 M13" \
                    - htree_sections [ list \
                                        [ list_section_name LuanP_A -prefix MSCTS_htree -tap_locations {{525.1 552.5}} \
                                        [ list_section_name LuanP_B -prefix MSCTS_htree -tap_locations {{511.2 964.5}} \
                                        [ list_section_name LuanP_C -prefix MSCTS_htree -tap_locations {{905.3 1035.5}} \
                                        [ list_section_name LuanP_D -prefix MSCTS_htree -tap_locations {{479.2 909.3}} ]

report_regular_multisource_clock_tree_options
```

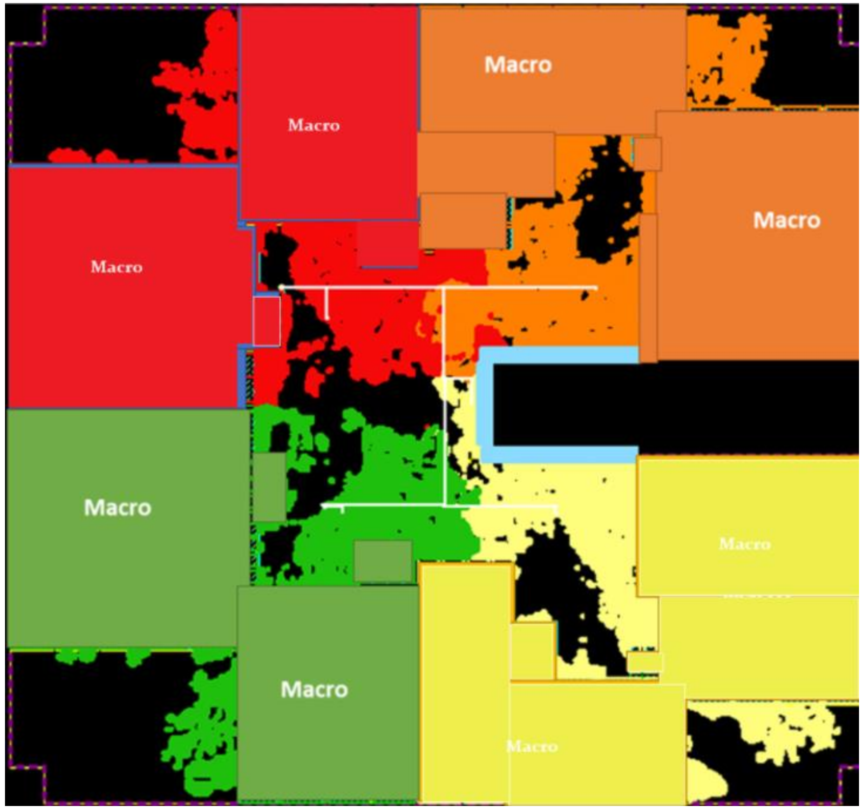


Result

- **The FC tool automatically inserts tap drivers, build H-tree, perform tap assignment.**

```
#Insert tap drivers, build H-tree, perform tap assignment
synthesize_regular_multisource_clock_trees -from tap_synthesis -to htree_synthesis

# Highlight sink distribution from tap assignment
source highlight_multisource_clock_subtrees.tcl
highlight_multisource_clock_subtrees –clock clka
```



Conclusions

Section Subtitle

Conclusions

- Overall, this new feature –htree_section of RMSCTS, which available from ver U-2022 of Fusion Compiler is very useful for us.
- **Advantage**:
  - QoR improved:
    - Latency and skew at CTS step improved 6%
    - Timing violation reduced 37%
    - Run time reduce 3% after PnR
    - Power almost same.
  - This feature will be most effective for complex floorplan because sinks are divided into section.
  - Basically, implement clock tree manually need to take care by experience engineer and it also take time. But with new technique, engineer only need to provide full combo 3 commands and FC can handle it.
- **Disadvantage**:
  - Need several trials to define good boundary section. To overcome it, we share an experience:

    - Each branch of H-tree will have the specific number of End points -> The boundary of corresponding section must cover all of the target End points (FF) area.

Reference

Body Slide Subtitle

[1] IC_Compiler_II_MCTS_overview_2018.06

[2] Fusion Compiler Tool Commands [U-2022]

THANK YOU

YOUR
INNOVATION
YOUR
COMMUNITY