



Enhancing Timing Signoff with Timing Constraint Manager: Case Study for SerDes and PCIe IPs

Synopsys Tutorial

Pawankumar Yendigeri

Srini Gaddam

Synopsys India Pvt Ltd, India

Priyanka Goel

Apoorv Srivastava

Outline

- Timing closure and its challenges with current flows
- Synopsys E2E Constraints Flow
- Case-Study in SerDes MIPI MPHY
- Discussion
- Key Takeaways
- TCM Promotion flow

Timing closure and its challenges with current flows

Overview

Front-End QA:

- Uses Quality of Results (QoR) metrics.
- Involves linting and synthesis constraints.

Back-End Challenges:

- Complexity increases in Clock Tree Synthesis (CTS) and Place and Route (PnR).
- Critical for achieving performance, power, and area (PPA) targets.

Gate-Level STA:

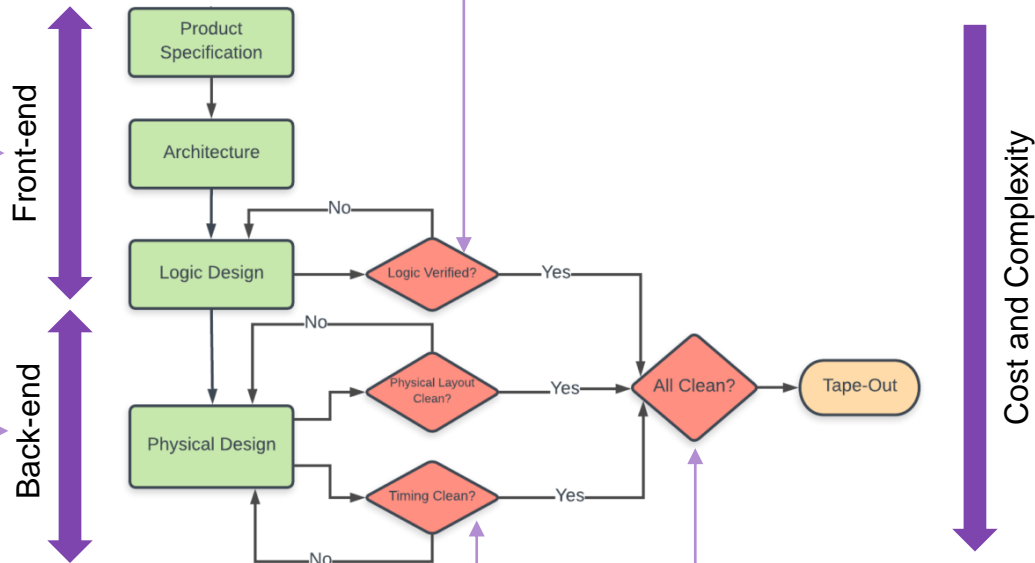
- Major timing issues often found here.
- Penultimate ASIC flow stage with detailed design analysis.

Gate-Level Simulations (GLS):

- Identifies and addresses timing issues.
- Limited modification scope; debugging is costly.
- Propagated issues cause delays and higher costs.

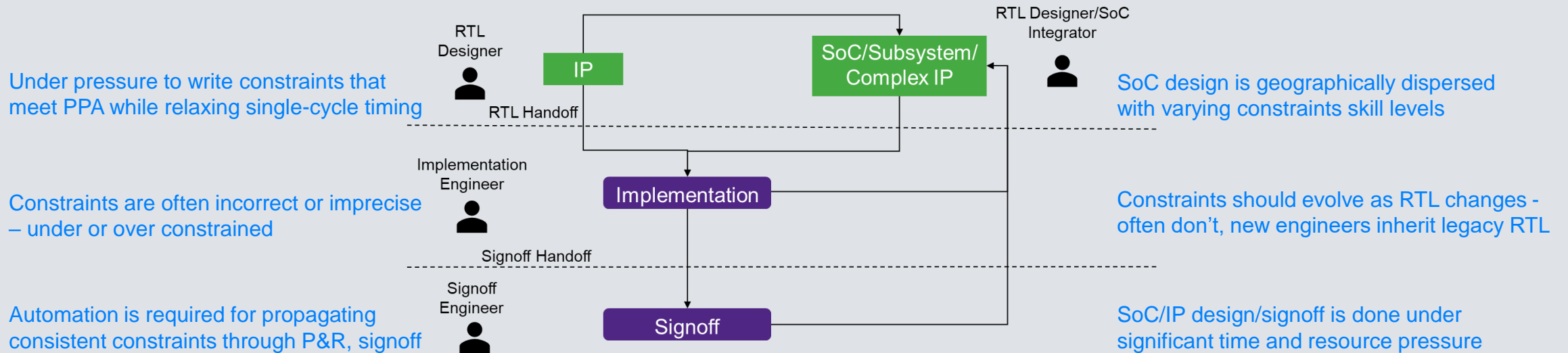
Timing Signoff:

- Critical for verifying backend ASIC design flows.
- Ensures design meets timing requirements.



The Constraints Challenge at Customers

SoC Design with External / Internal IP



Undetected Incorrect Constraints



Silicon Failure

Noisy Unusable Reports



Weeks of Unproductive Review Efforts

Over-Constrained Design

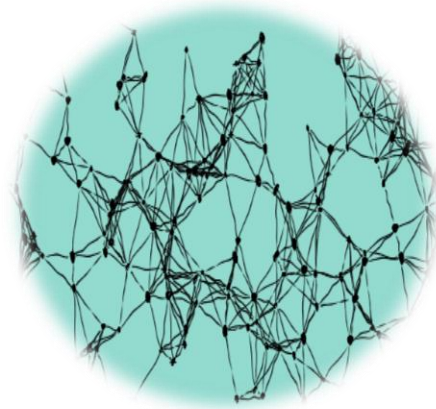


Long TAT / Sub-optimal PPA

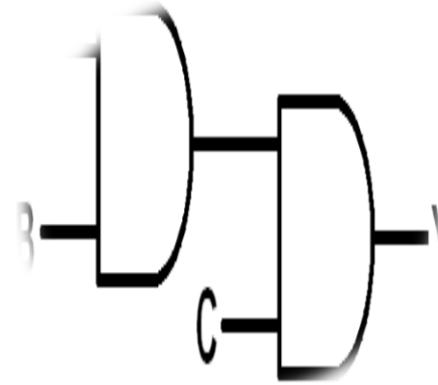
Issues with Timing Closure



Detection
at
penultimate
stages



Complex
clock tree
networks



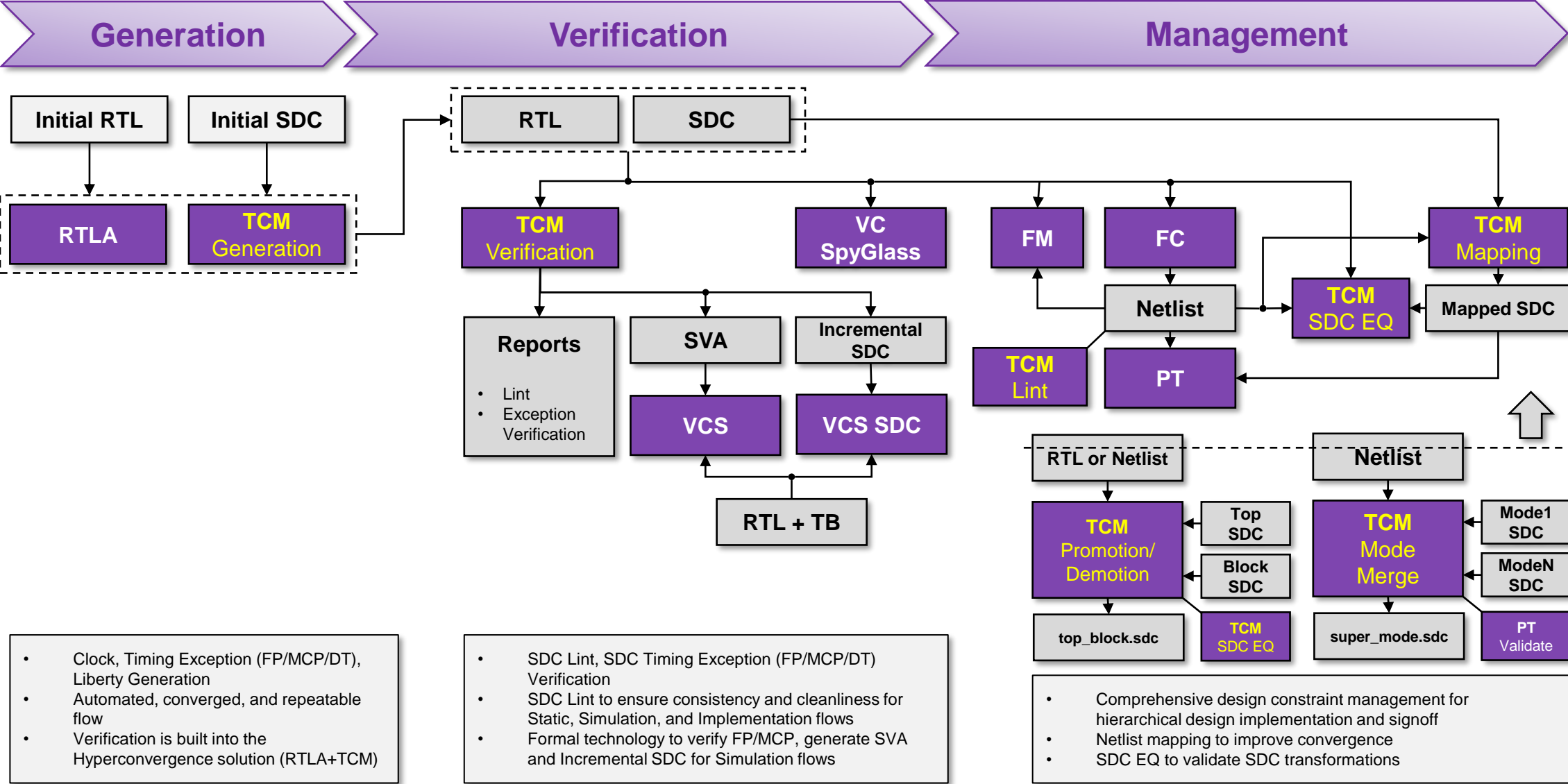
Post CTS
Netlist based
path checks



High
computation
Power TAT

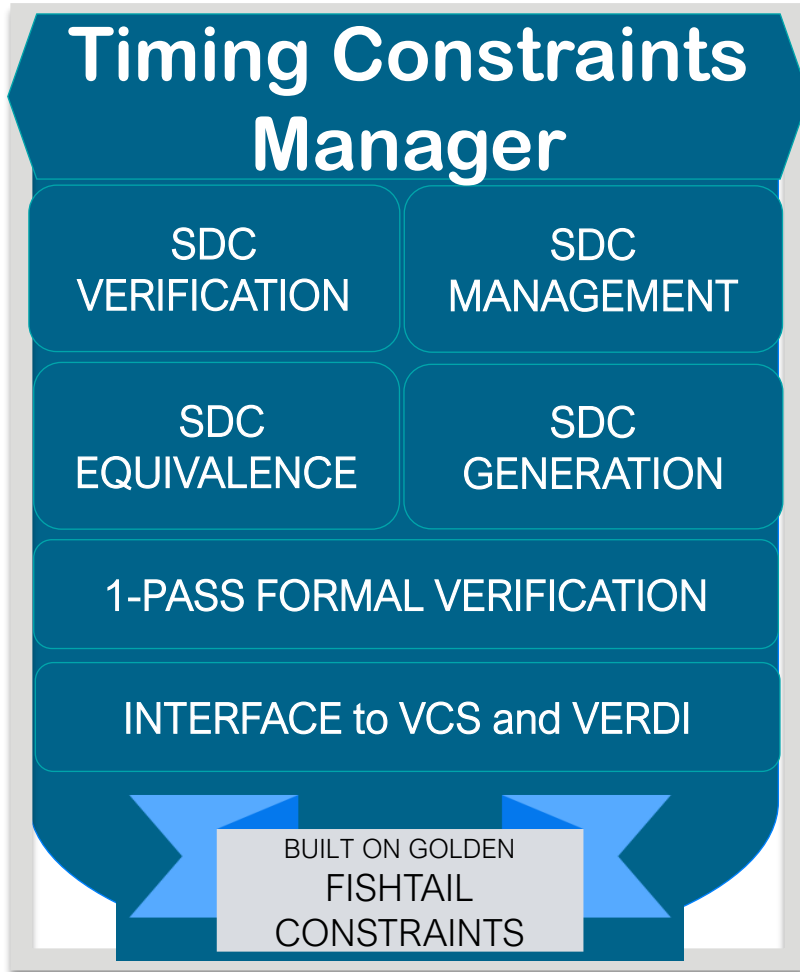
Synopsys E2E Constraints Flow

Synopsys E2E Constraints Flow



Synopsys Timing Constraints Manager

Built on FishTail Best-in-Class SDC Constraints Solution



- Comprehensive SDC Timing Constraints Generation, Verification and Management
- Multi-Cycle/False Path Exception Verification with No Noise
RTL Designers are provided precise feedback on their SDC bugs
- Comprehensive SDC Management solution
Tape-out proven promotion, demotion, mapping solution
- Automated SDC Generation from RTL
Saves weeks of designer effort and development schedule

<p>90% </p> <p>Reduction in Designer SDC Review Effort</p>	<p>Auto SDC Gen/Mgmt. </p> <p>shortens SoC Execution by 6-8 Weeks</p>	<p>Eliminate  silicon failure from SDC bugs</p>
--	---	--


Deployed at Top Tier Semi Companies with Production-Proven Tape-outs

Case-Study SerDes applications

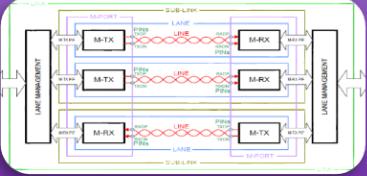
Case Study



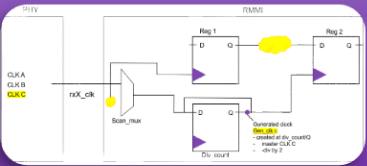
Use of TCM can be extensively tested on applications such as SerDes, where higher number of clock generation and verifications is involved



Majorly high-speed interface IPs like MIPI MPHY, involves high number of clocks and propagated paths that can be majorly benefitted through TCM flow

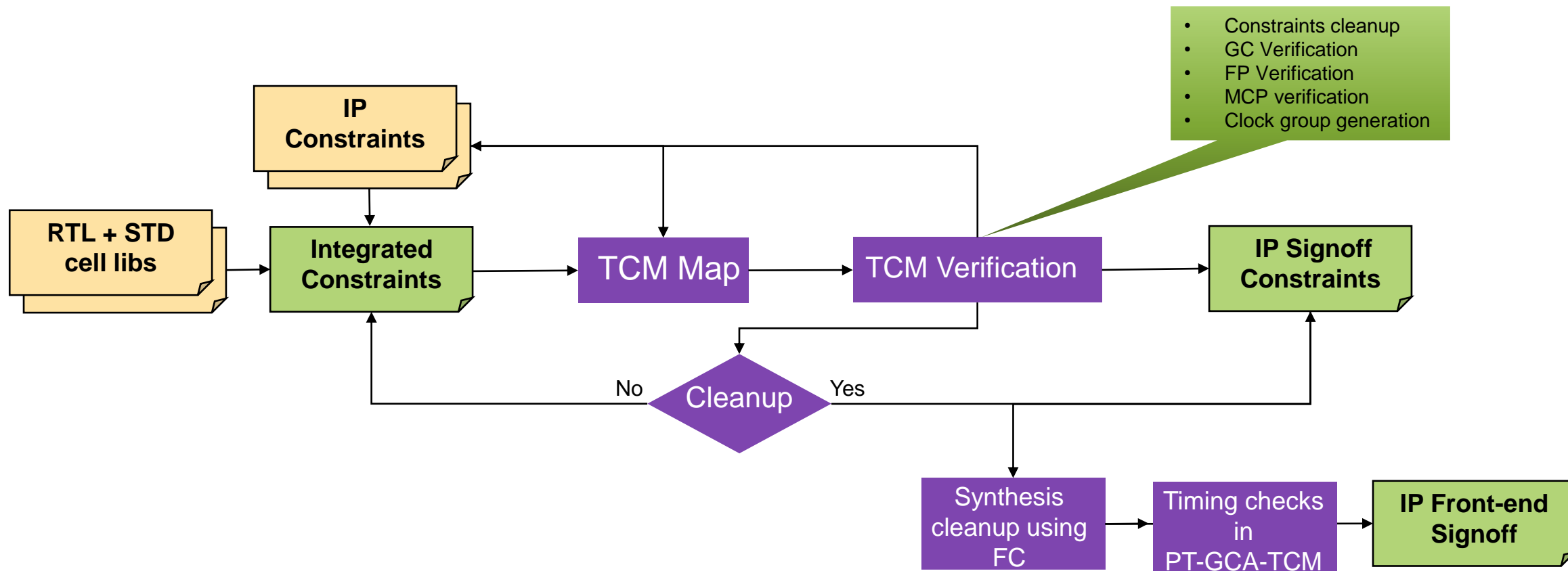


The MPHY (generic PHY+RMMI) clocking architecture involves more than 50 clock generations based on mode, gears(1 to 5), high speed(PLL) and low speed(PWM) etc.



The exceptions reported are crucial as it contains many constraints that cater to be used for specific configurations in

TCM Deployment Methodology



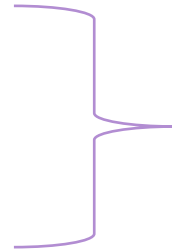
Discussions

The background of the slide features a teal gradient on the left. On the right, there are two large overlapping circles: a yellow one in front of a purple one. A white dashed circle is positioned above the yellow circle. In the bottom right corner, there is a faint white network diagram with nodes and connecting lines.

Clock Verification checks - Example

Design Information (dwc_mipi_mphy_type1_g5_2tx_2rx_ns)

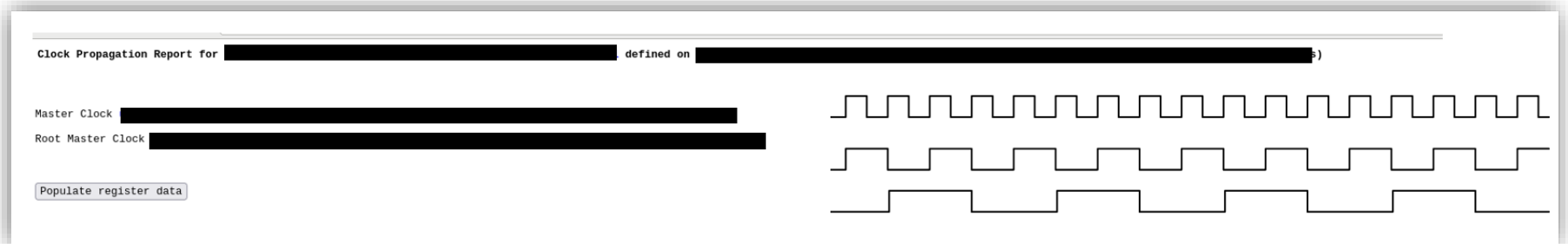
- Clocks
- Clock Distribution
- I/O Delays
- Constant Propagation For Case Analysis
- Legal Values
- Metastable Endpoints
- Modules



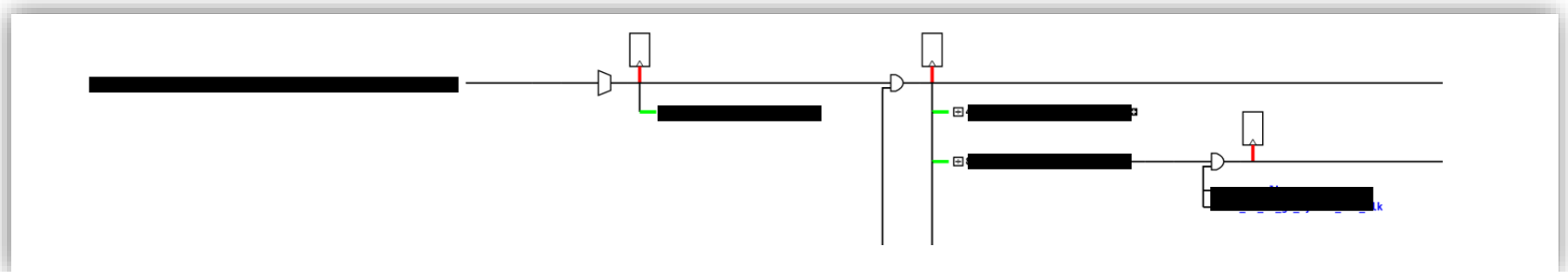
Basic constraint dashboard with all RTL mapped constraints along with module information

```
Consider below GC constraint-
create_generated_clock
-master <clk a>
-source <pll/out>
-divided_by 2
get_pins *mux/out
-combinational
```

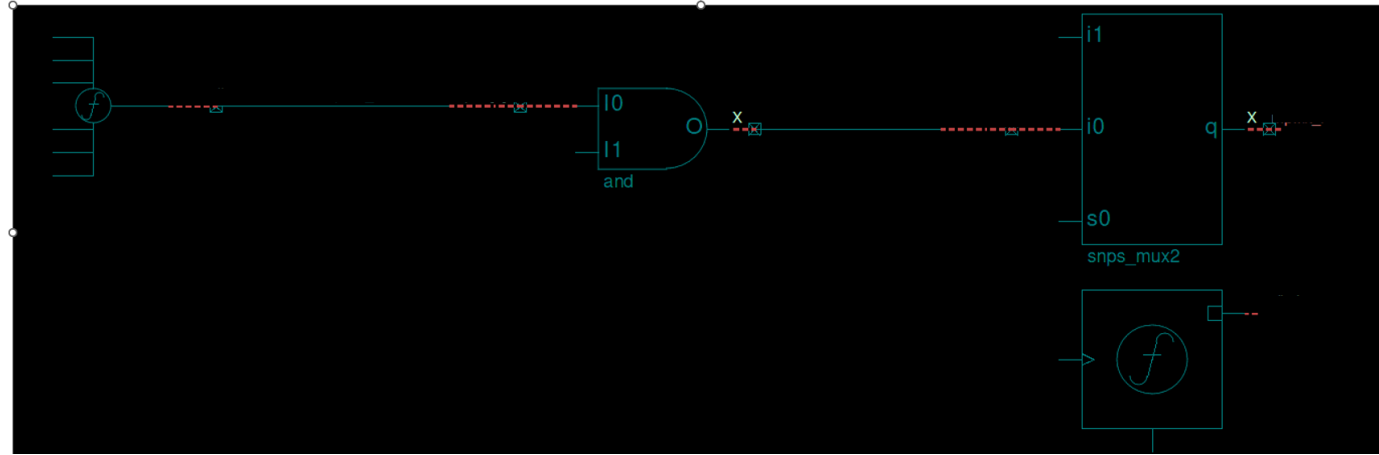
Clock Propagation report with root master information with the clocked registers



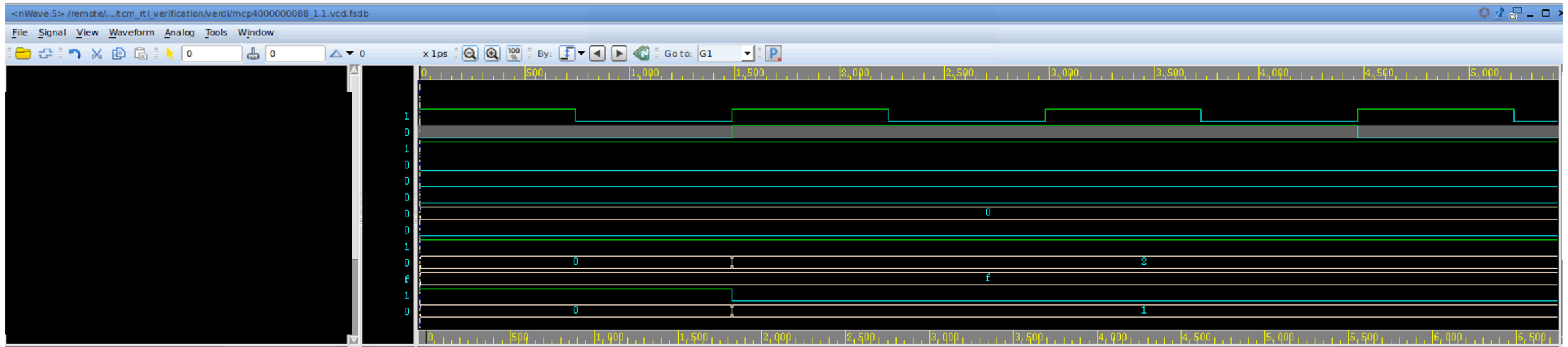
GUI based schematic for representing Clock distribution



Clock Verification checks - Example



GUI Schematic depicting clock propagation condition

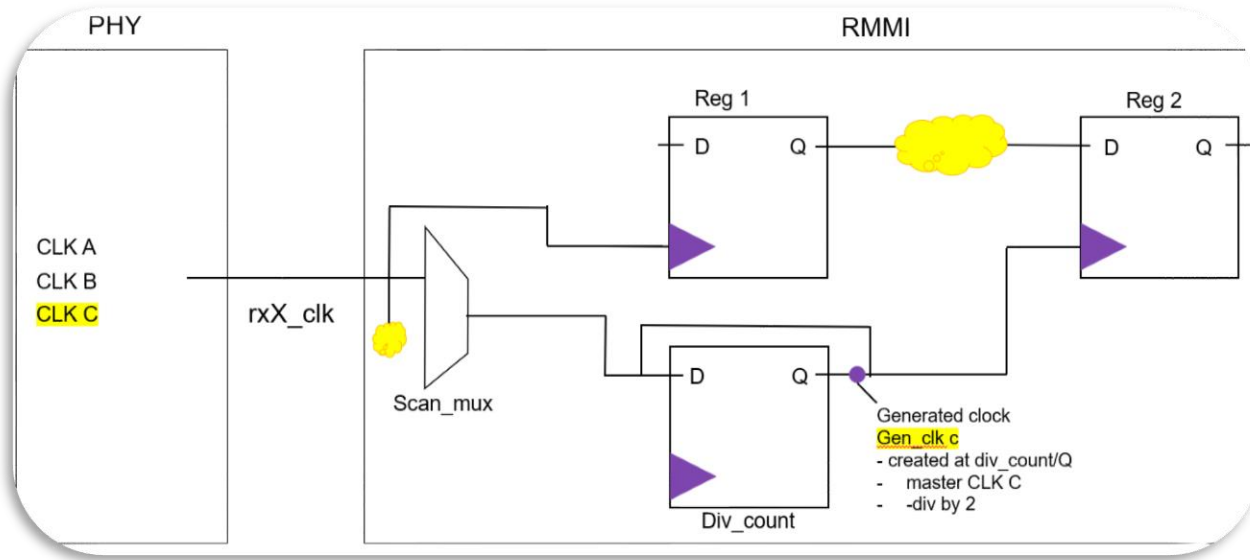


Assertion dumped for verifications of high/low pulse checks for generated clocks

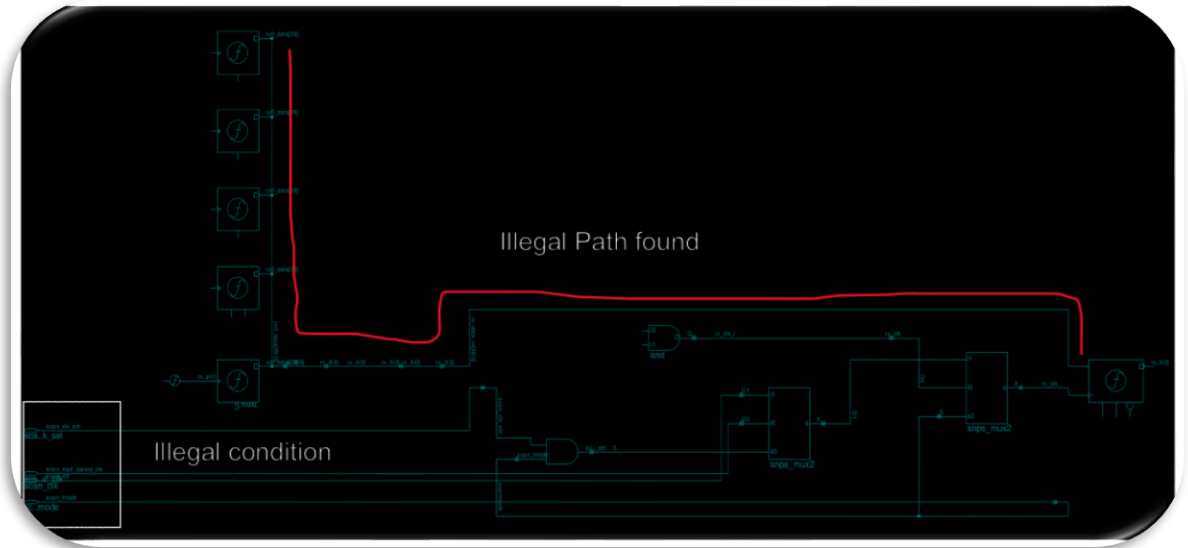
False Path Verification checks - Example

False path exception mapped by TCM

```
set_false_path -from [get_clocks clk a] -through [get_pins -hier "port a port b ..."]
```



MPHY Rx Clocking architecture
Uses CLK C to create Gen_clk c



Verdi based schematic view for assertion generated for false path constraint

Implicit timing updates- Example

Use of below queries in the constraints causes Synthesis/STA tool to trigger implicit timing update-

- ❖ get_attribute [*] query
- ❖ set_case_analysis
- ❖ remove_case_analysis
- ❖ all_registers/all_fanin/all_fanout

Consider below MPHY example-

```
set_input_delay [expr {[get_clk_period CLK a]} [get_ports port a] -add_delay
```

where,

```
proc get_clk_period {clk} {
  set Period [get_attribute [get_clocks $clk] period]
  return $Period }
```

This resulted in TCM warning (SDC_068) in setup as below as it will not query the attribute-

```
Error: Execution error in SDC on line 'xx' of file 'mphy_constraints. (SDC-068)
>> can't use empty string as operand of "/"
>>   while executing
>> "expr {[get_clk_period CLK a]}"
>>   invoked from within
>> set_input_delay [expr {[get_clk_period CLK a]} [get_ports port a] -add_delay..."
```



TCM Checks

TCM FT Dashboard

Message	Severity	Warnings	Waived
CLK-006	⚠	13	0
CLK-009	⚠	47	0
CLK-010	⚠	11	0
CLK-016	⚠	4	0
CLK-018		22	0
CLK-027		20	0
CLK-028		313	0
CLK-037	⚠	3	0

Clock warnings

Message Type	Warnings	Waived
Clock Issues	624	0
I/O Issues	703	0
Case Analysis Issues	1	1
Exception Issues	0	0

Constraint warnings

Clock Verification Warnings	0
Clock Issues	433
I/O Issues	3421
Case Analysis Issues	1
Exception Issues	0

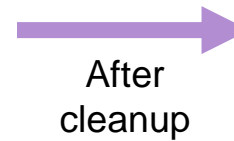
Generated Clock Verification	100%
Generated Clocks	110
Passing	19
Failing	0
Skipped	82
Waived	0

Classification	Exception Count	Path Count		
		Pass	Fail	Total
PASS	0	0	0	0
PARTIAL FAIL	2	108	448	556
FAIL	4	0	4008	4008
NO PATHS	29	0	0	0
SKIPPED (Async Clocks)	0	0	0	0
SKIPPED (I/O Paths)	0	0	0	0
SKIPPED (Internal)	0	0	0	0
SKIPPED (Hold FP)	0	0	0	0
SKIPPED (Duplicate)	0	0	0	0
SKIPPED (User)	0	0	0	0
WAIVED	0	0	0	0
WAIVED BY TOOL	0	0	0	0
Total	35	108	4456	4564

False path checks

Classification	Exception Count	Path Count		
		Pass	Fail	Total
PASS	0	0	0	0
PARTIAL FAIL	0	0	0	0
FAIL (Incorrect Shift)	0	0	0	0
FAIL	0	0	0	0
NO PATHS	0	0	0	0
SKIPPED (Async Clocks)	0	0	0	0
SKIPPED (I/O Paths)	1	0	0	0
SKIPPED (Internal)	0	0	0	0
PASS (Hold < Setup)	1	0	0	0
SKIPPED (No Relaxation)	0	0	0	0
SKIPPED (Duplicate)	0	0	0	0
SKIPPED (User)	0	0	0	0
WAIVED	0	0	0	0
WAIVED BY TOOL	0	0	0	0
Total	2	0	0	0

Multi-cycle path checks



False Path Verification	100%
False Paths	35
Passing	0
Failing	0
No Paths	30
Skipped	5
Waived	0

Multicycle Path Verification	100%
Multicycle Paths	2
Passing	1
Failing	0
No Paths	0
Skipped	1
Waived	0

Key Takeaways

Early-stage detection

- As TCM requires only RTL as a part of design input, it proves to be beneficial in tracking and debugging constraint issues at initial stages of ASIC flow

Rigorous constraints checks

- The rules involved in TCM are categorical in addressing issues related to clock domain crossing, clock relationships, io delays for design ports etc.

Timing Optimisation

- TCM helps getting rid of constraints that cause implicit timing updates that result in long source time for constraints during Back-end flows

Runtime

- As TCM does not involve any requirement of synthesis-based netlist setup, the QA time is much lower than any other constraint management tool

Promotion

- Issues related to adaptation of constraints in the upper layer in IP integration can be tracked in early stages using TCM promotion flow

Less noise in GLS setup

- Cleanup of GLS will be quicker, since the simulation-based setup in TCM mimics the issues that can be seen in the GLS setup

Guidelines and Best Practices for Subsystem Constraints Development with TCM Promotion Tool

Apoorv Srivastava, Priyanka Goel, Akanksha Jat,
Naveen Battu
Synopsys

Agenda

- TCM Constraints Promotion Flow
 - The Value Proposition
 - The Flow
 - Steps (Unix Commands for TCM Constraints development flow)
 - Why Top SDC if we are promoting?
- Insights from Case Study of PCIe6 Subsystem
 - Overview of Pilot Project
 - Tool Behavior
 - Guidelines
- Conclusion: Pros/Cons

TCM Constraints Promotion Flow

The Value Proposition

SOC involves multiple sub-systems (SS) and blocks with complex timing constraints

TCM tool promotes block-level constraints to the SS level

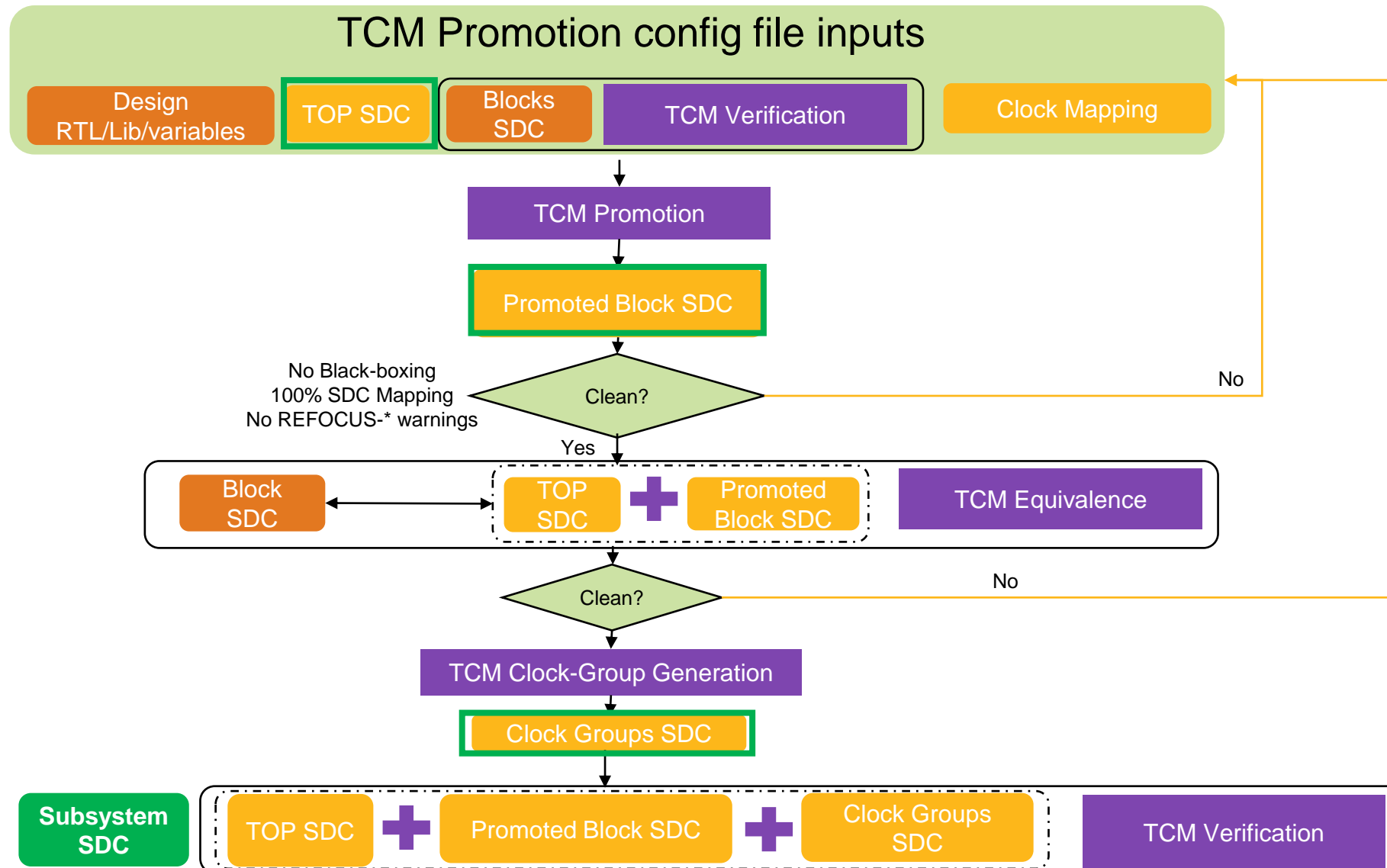
Writing SS Constraints requires understanding of block-level constraints

=> Longer lead times & is error prone

With TCM, deep understanding of block-level constraints is not needed

In this presentation, we will share our experience of using TCM Constraints promotion for a PCIe6 Subsystem consisting of Synopsys PCIe6 Controller and Synopsys PCIe6 PHY

The Flow



Steps (Unix Commands for TCM Constraints development flow)

- –Unix %> <source> tcm
- –Unix %> create_fishtail_scripts-config rtl_constraint_promotion.cfg
- –Unix %> cd <fishtail_rtl_constraint_promotion>/constraint_promotion
- –Unix %> ./promote_constraints
- –Unix %> output : promoted.sdc
- –Unix %> view_fishtail_result-verdi
- –Unix %> cd ../equivalence_checking
- –Unix %> ./check_equivalence
- –Unix %> view_fishtail_result-verdi
- –Unix %> cd ../clock_group_generation
- –Unix %> ./generate_clock_groups
- –Unix %> output : clock_groups.sdc

Steps (Unix Commands for TCM Verification)

- –Unix %> `create_fishtail_scripts -config rtl_verification.cfg`
- –Unix %> `cd ./<tcm_verification>/setup`
- –Unix %> `./map_sdc_and_check_setup`
- –Unix %> `view_fishtail_result–Verdi`

Why Top SDC if we are promoting ?

- Top-level clocks, timing exceptions and IO constraints are required to be defined

```
#Top level SDC

#TOP level clocks
create_clock \
  -period 9.0 \
  { phy0_pmd_jtag_tck } \
  -name phy0_pmd_jtag_tck_i \
  -add

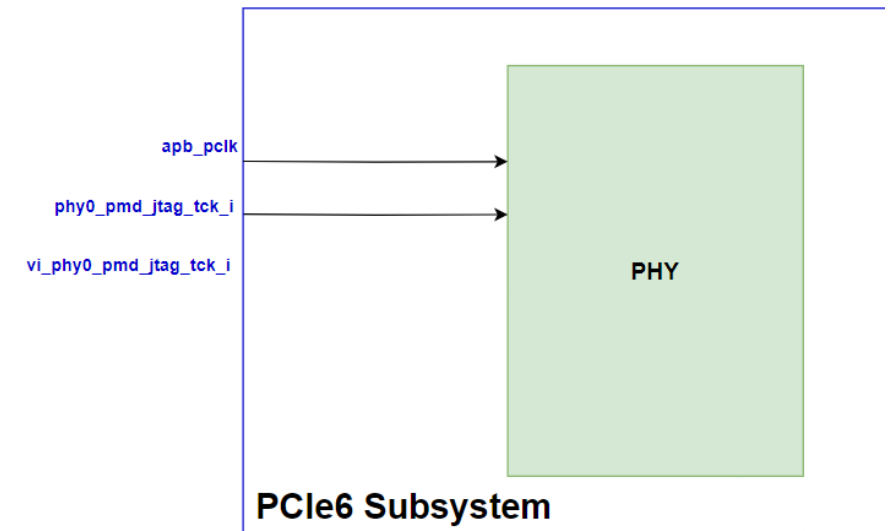
create_clock \
  -period 9.0 \
  -name vi_phy0_pmd_jtag_tck_i

create_clock \
  -waveform { 0 2.0 } \
  -period 4.000 \
  { apb_pclk } \
  -name apb_pclk

#Top level timing exceptions
set_clock_groups \
  -asynchronous \
  -name all_phy_functional_clks_from_different_sources_top_clks \
  -group { apb_pclk } \
  -group { phy0_pmd_jtag_tck_i vi_phy0_pmd_jtag_tck_i }

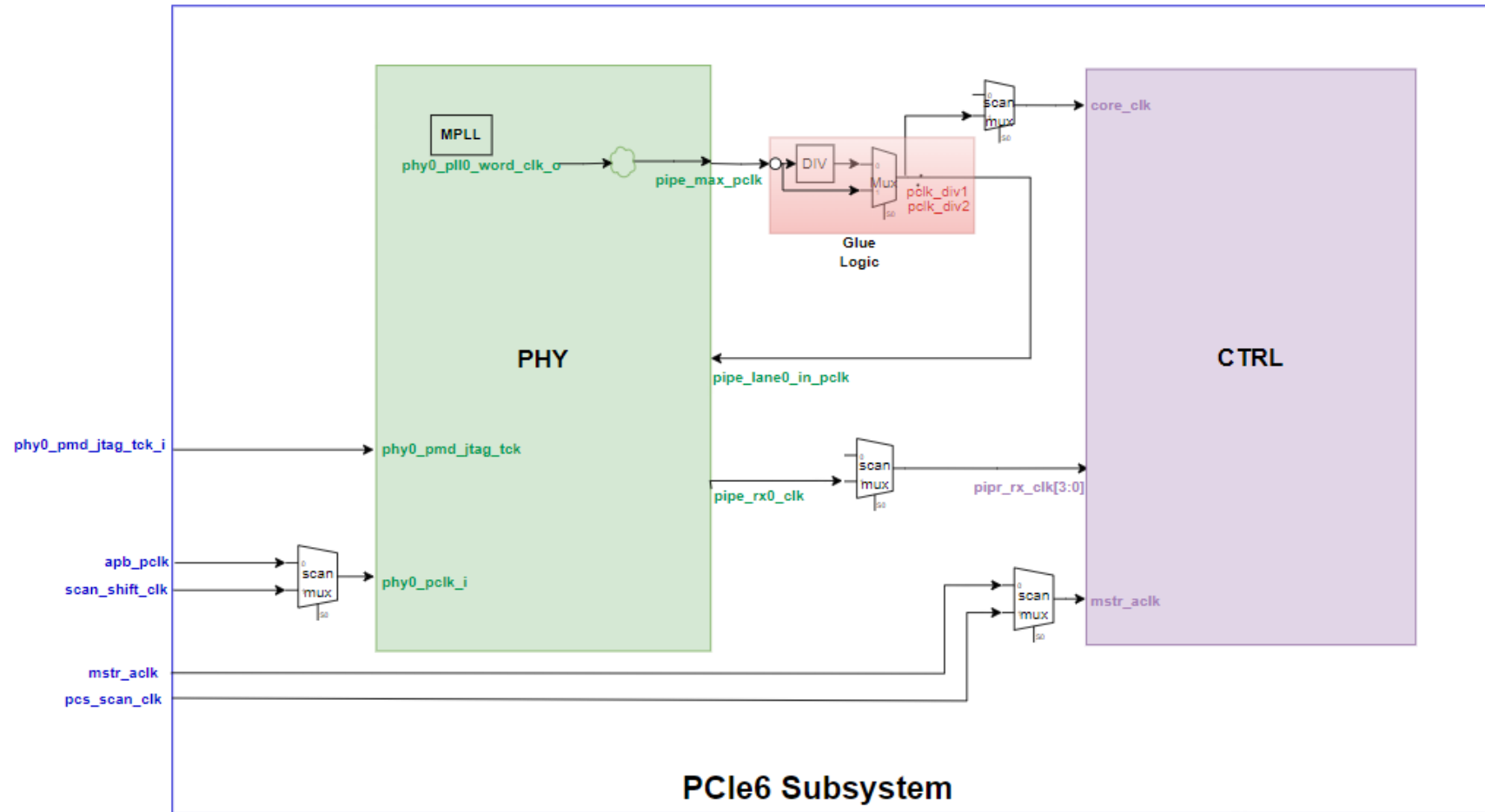
set_input_delay \
  2.2 \
  -clock { vi_phy0_pmd_jtag_tck_i } \
  { phy0_pmd_jtag_tms } \
  -max

set_false_path \
  -through { u_pcie_ctrl_top_0/u_pcie_core/u_DWC_pcie_native_core/u_DWC_pcie_core/u_cdm/cfg_tc_vc_map[*] }
```



Insights from Case Study of PCIe6 Subsystem

Overview of Pilot Project



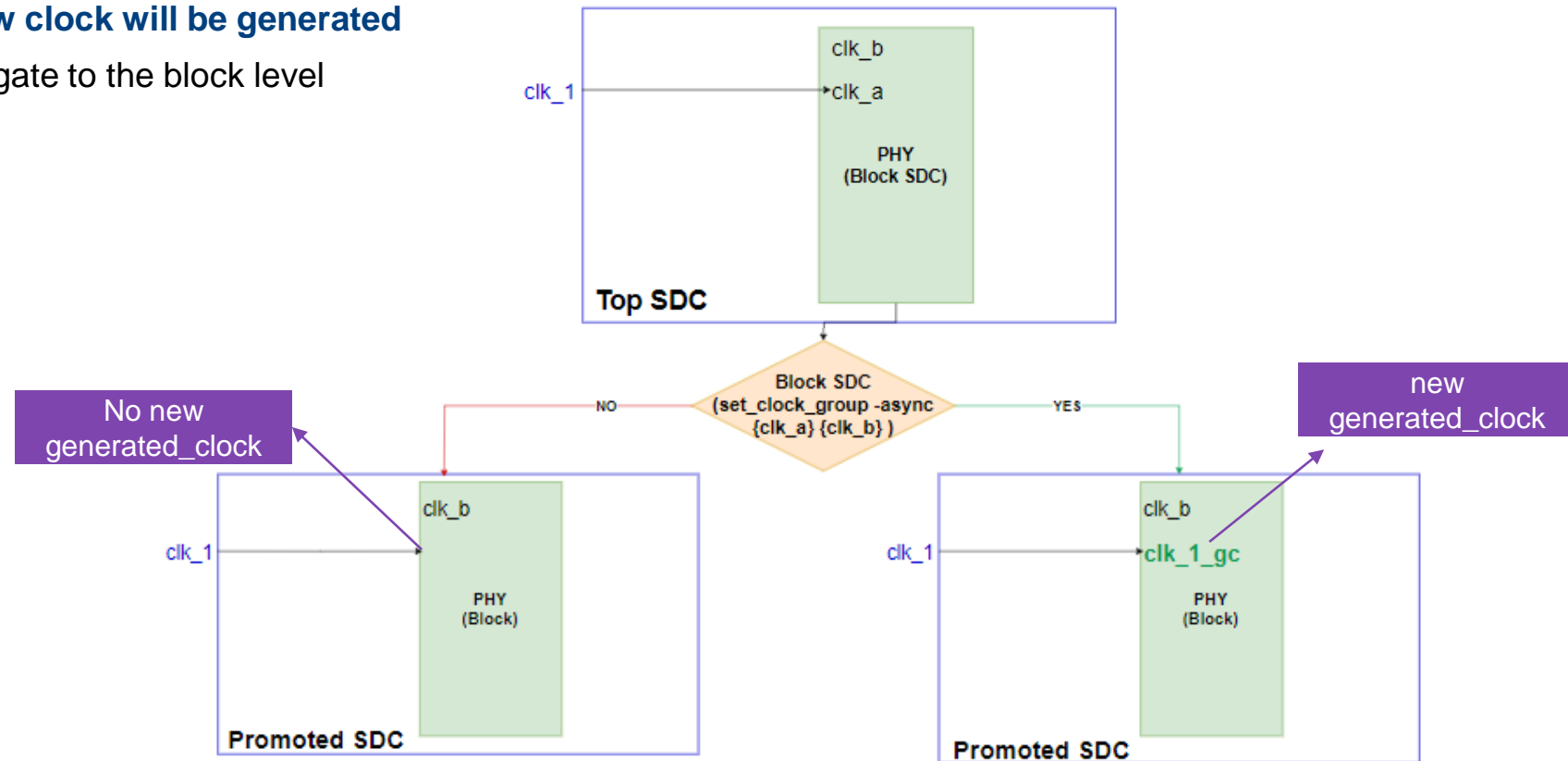
Block Diagram of PCIe6 Subsystem (with clocking)

Tool Behavior

- TB1: If, timing exception is not defined on a clock in block SDC

- After promotion, **no new clock will be generated**
- the top clock will propagate to the block level

e.g.:



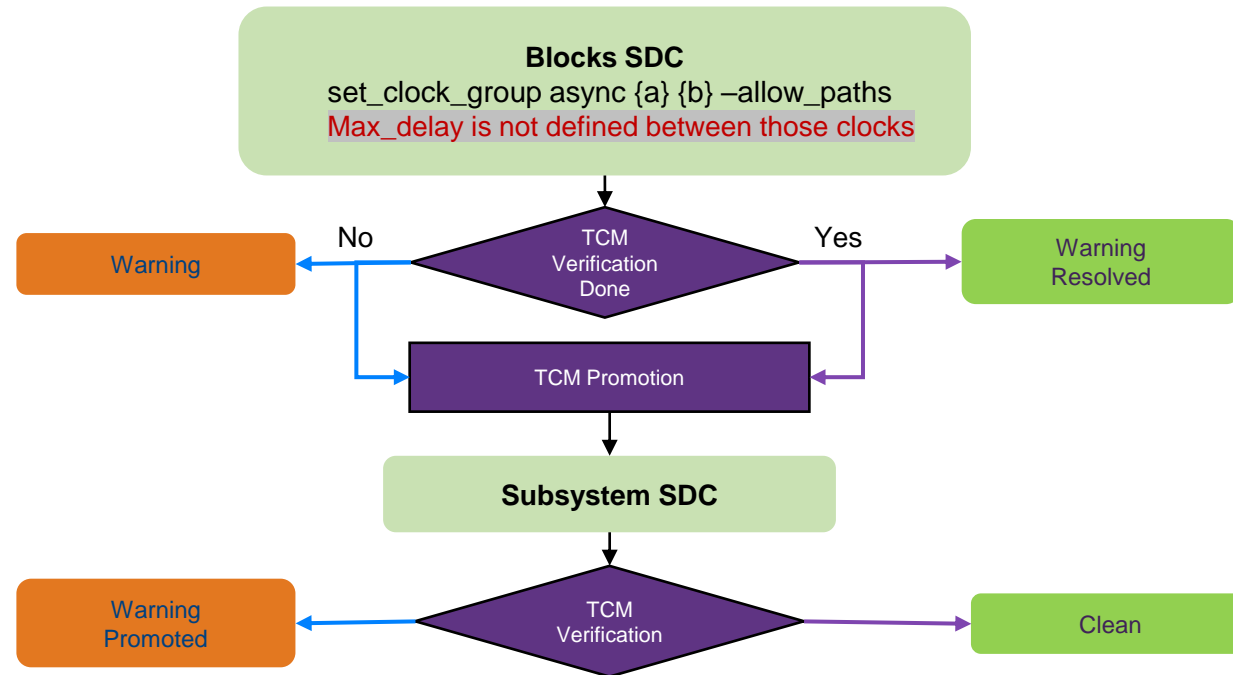
- TB2: Output delay at the block level will be promoted w.r.t. top clocks if there is a **direct connection between the mapped block clock to the top clock**

Guidelines:1/8

- G1: Block SDC should be passed through a **constraint's quality check** using the TCM verification tool
 - Otherwise, any weakness at block level SDC will get promoted to Subsystem SDC

e.g.

Warning: There is a timed asynchronous clock crossing but one or more paths for this clock crossing do not have a max delay constraint. (CLK-038)



Guidelines:2/8

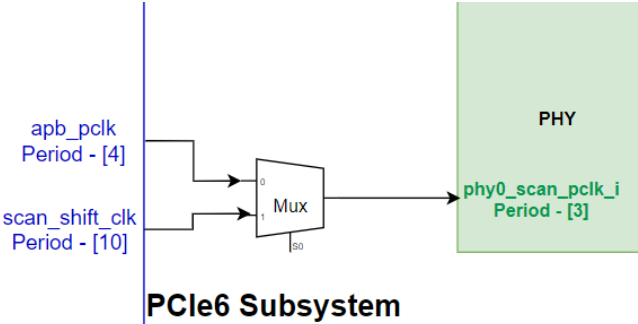
- G2: Top-level clocks propagating to the block-level clock ports **should match their period/name for tool default clock mapping**

- Otherwise, the tool will do a random mapping with a single clock and report a warning

Warning: The period of the top-level clock is different from the block-level clock that it is mapped to. (REFOCUS-070)

- The mapping can be overridden by user-defined clock mapping

e.g.:



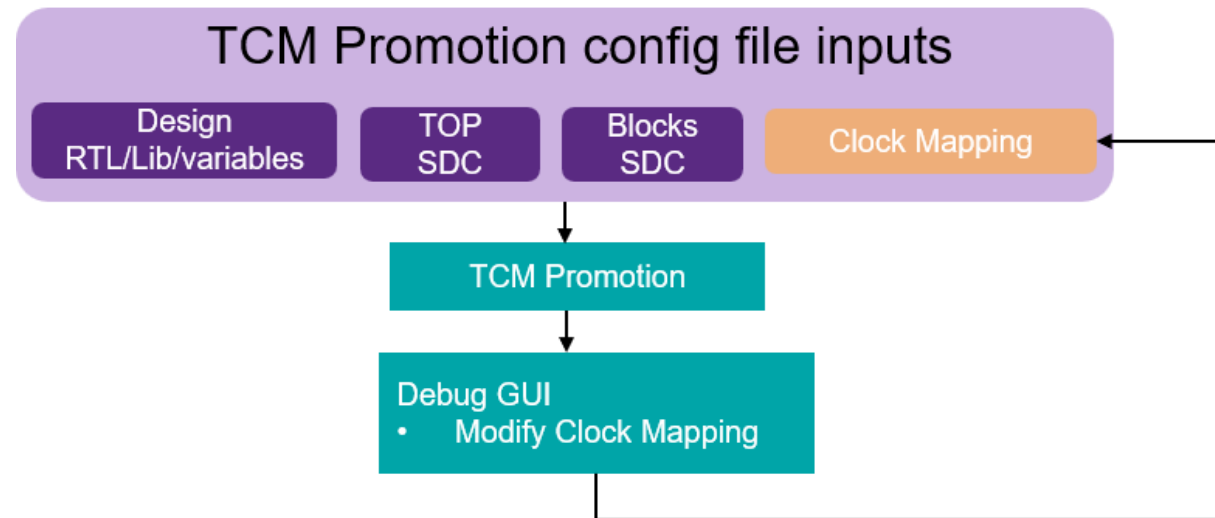
	CLOCK NAME	PERIOD
Top Clock	apb_pclk	4
	scan_shift_clk	10
PHY Clock	phy0_scan_pclk_i	3

- Tool Clock Mapping from TCM GUI looks like

Instance	Block Clock	Definition Pin	Propagated Top-Level Clocks	Mapped Top-level Clock	User Defined Clock Mapping
u_pcie_phy_top/pcie_pipe	phy0_scan_pclk_i	phy0_pclk_i	apb_pclk scan_shift_clk	apb_pclk	<input type="checkbox"/> scan_shift_clk <input type="checkbox"/> phy0_scan_pclk_i <input type="checkbox"/> apb_pclk

Guidelines:3/8

- G3: If multiple clocks need to map to block-level clock, **user-defined clock mapping is required**
 - Otherwise, the tool will do a random mapping with a **single clock** and report a warning
Warning: Multiple top-level clocks propagate to a block-level clock. All but one of them are dropped. (REFOCUS-064)
- After the first iteration of TCM Promotion, this file is dumped from the tool, modified, and provided as input to the promotion flow



Guidelines:3/8

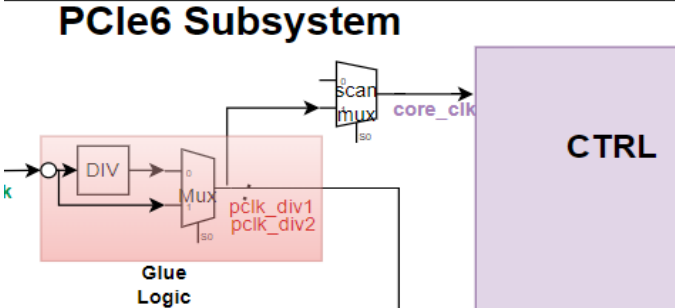
Clock Mapping

- Tool Clock Mapping from TCM GUI looks like

Block Clock	Definition Pin	Propagated Top-Level Clocks	Mapped Top-Level Clock	User Defined Clock Mapping
core_clk	core_clk	lane0_pcs_scan_pclk lane0_pcs_scan_shift_pcs_clk pclk_div1 pclk_div2	lane0_pcs_scan_pclk	<input type="checkbox"/> lane0_pcs_scan_shift_pcs_clk <input type="checkbox"/> pclk_div1 <input type="checkbox"/> pclk_div2 <input type="checkbox"/> core_clk <input type="checkbox"/> lane0_pcs_scan_pclk

- Providing User defined mapping- by selecting desired clocks

Block Clock	Definition Pin	Propagated Top-Level Clocks	Mapped Top-Level Clock	User Defined Clock Mapping
core_clk	core_clk	lane0_pcs_scan_pclk lane0_pcs_scan_shift_pcs_clk pclk_div1 pclk_div2	lane0_pcs_scan_pclk	<input checked="" type="checkbox"/> lane0_pcs_scan_shift_pcs_clk <input checked="" type="checkbox"/> pclk_div1 <input checked="" type="checkbox"/> pclk_div2 <input type="checkbox"/> core_clk <input checked="" type="checkbox"/> lane0_pcs_scan_pclk



- Dumped File – clock_mapping.tcl

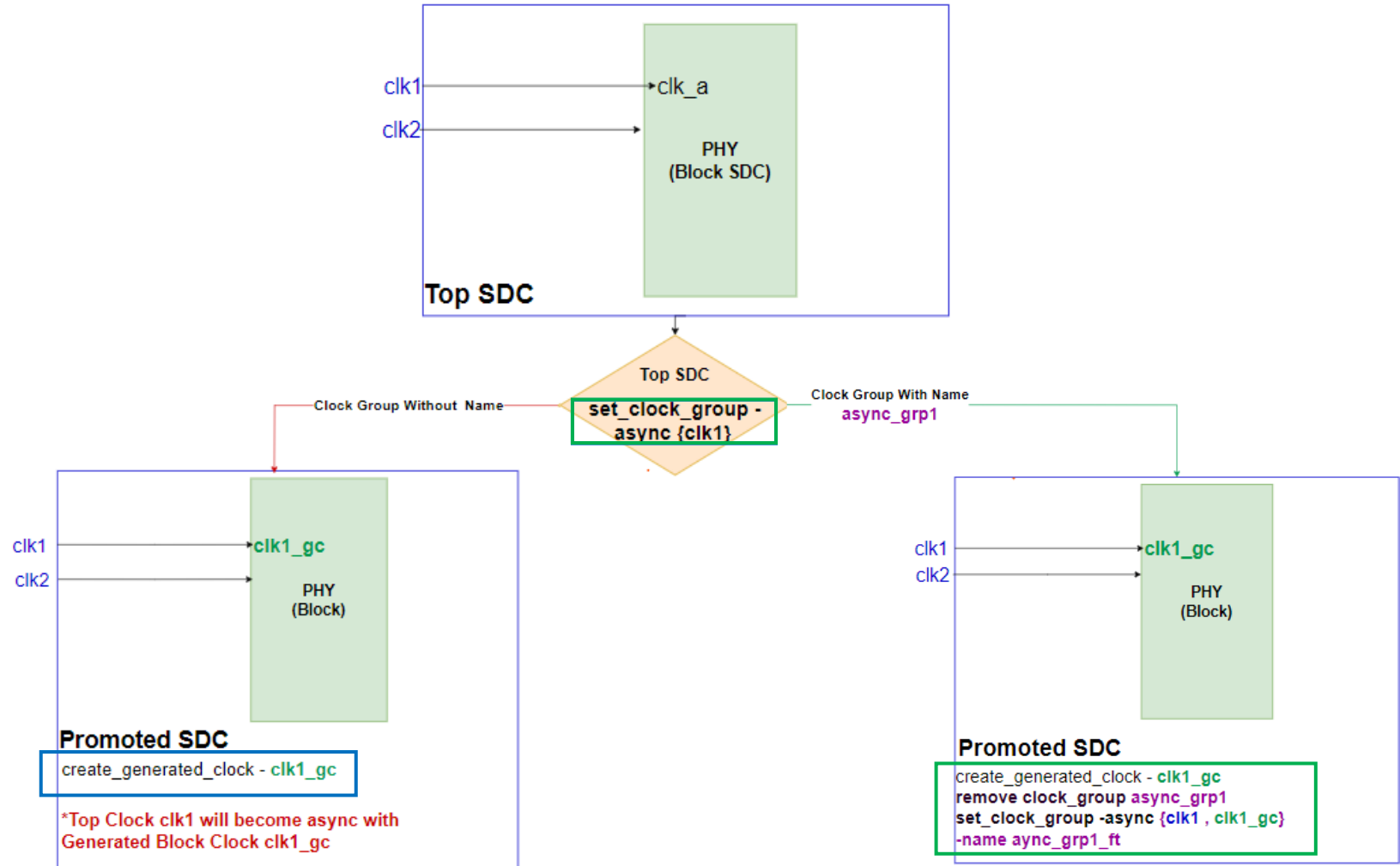
```
map_top_clock_to_block_clock -block_clock core_clk_ug -top_clock { lane0_pcs_scan_pclk lane0_pcs_scan_shift_pcs_clk pclk_div1 pclk_div2 } -pin u_pcie_ctrl_top_0/u_pcie_core/core_clk_ug
```

- User-defined Clock Mapping

Block Clock	Definition Pin	Propagated Top-Level Clocks	Mapped Top-Level Clock	User Defined Clock Mapping
core_clk	core_clk	lane0_pcs_scan_pclk lane0_pcs_scan_shift_pcs_clk pclk_div1 pclk_div2	lane0_pcs_scan_pclk lane0_pcs_scan_shift_pcs_clk pclk_div1 pclk_div2	<input type="checkbox"/> core_clk <input checked="" type="checkbox"/> lane0_pcs_scan_pclk <input checked="" type="checkbox"/> lane0_pcs_scan_shift_pcs_clk <input checked="" type="checkbox"/> pclk_div1 <input checked="" type="checkbox"/> pclk_div2

Guidelines:4/8

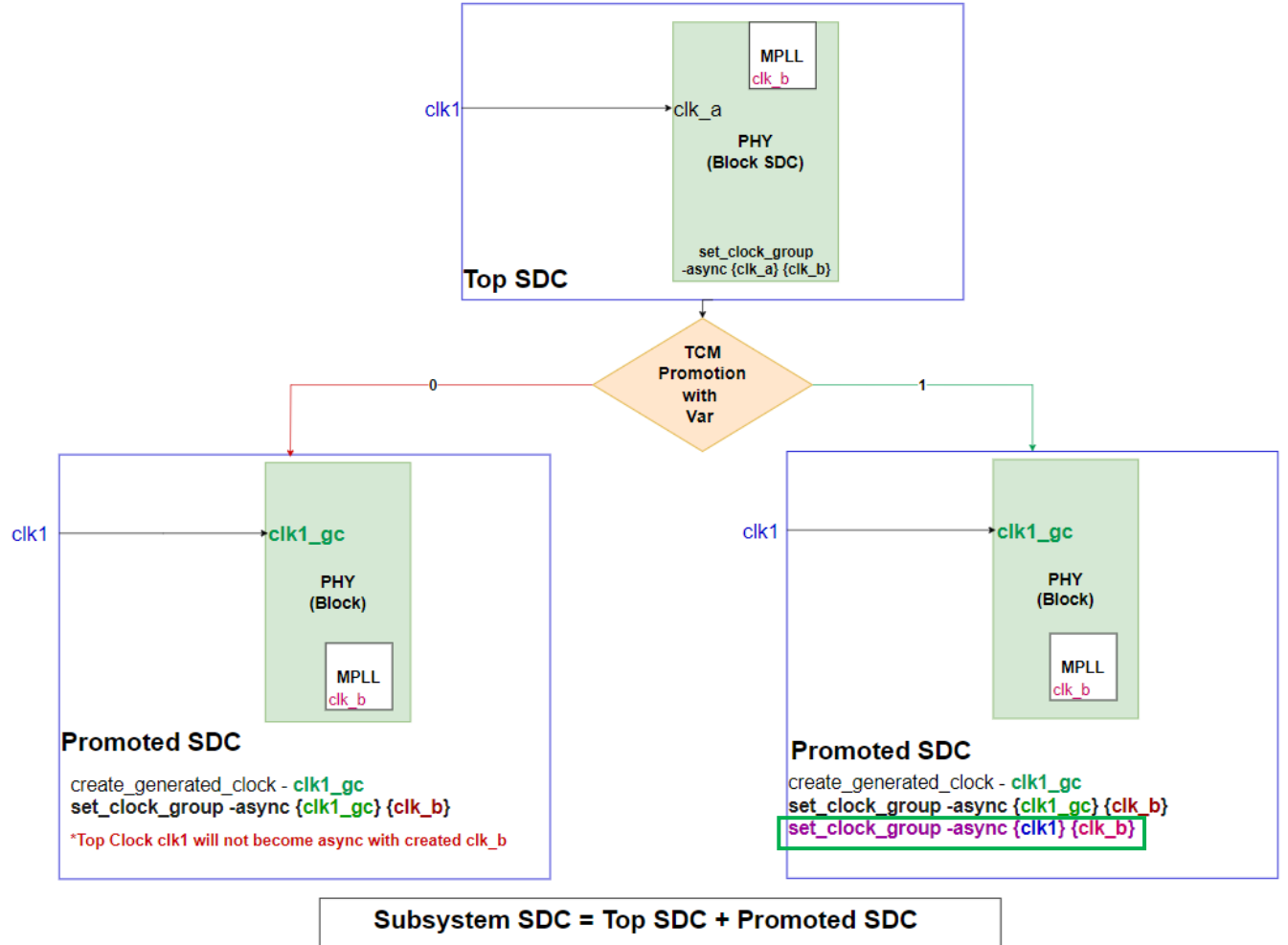
- G4: In Top SDC, **clock_group** should be defined with **clock_group_name**
 - The tool can remove the top clock group after promotion, if required and create a new clock group with updated clock relationship
e.g. a clock async with all other clocks at the top level
 - After promotion it should be synchronous with its generated clock and asynchronous with all other clocks.



Subsystem SDC = Top SDC + Promoted SDC

Guidelines:5/8

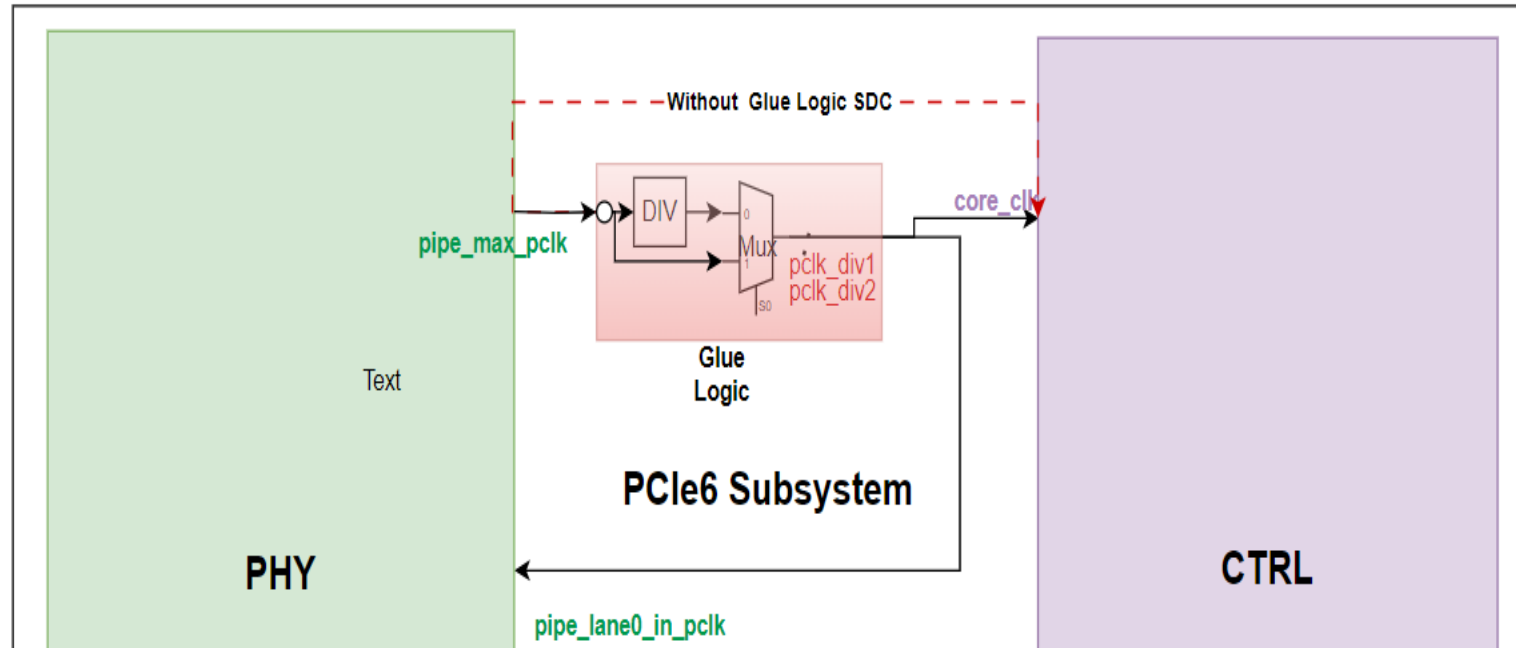
- G5: Set TCM variable `create_async_clock_group 1` in TCM config file
 - It creates an async clock group that specifies
 - That clocks **created** in different blocks being promoted are async to each other and top-level clocks
 - If `create_async_clock_group` is `0`, the tool will report a warning in TCM verif for missing clock group
 - Warning: There is an asynchronous clock group between clock 'clk_a' and clock 'clk_b' but their master clocks 'clk1' and 'clk_b' are missing this constraint. (CLK-026)



Guidelines:6/8

- G6: The **SDC should be provided** for a block in Subsystem glue logic that has a **clock-shaping circuit (divider)**
e.g.:

In our PCIe6 Subsystem, the CLKRST block has a divider that connects the PHY output max_pclk to the CTRL input core_clk
Glue Logic (CLKRST) SDC is provided along with PHY and CTRL for constraint promotion



Guidelines:7/8

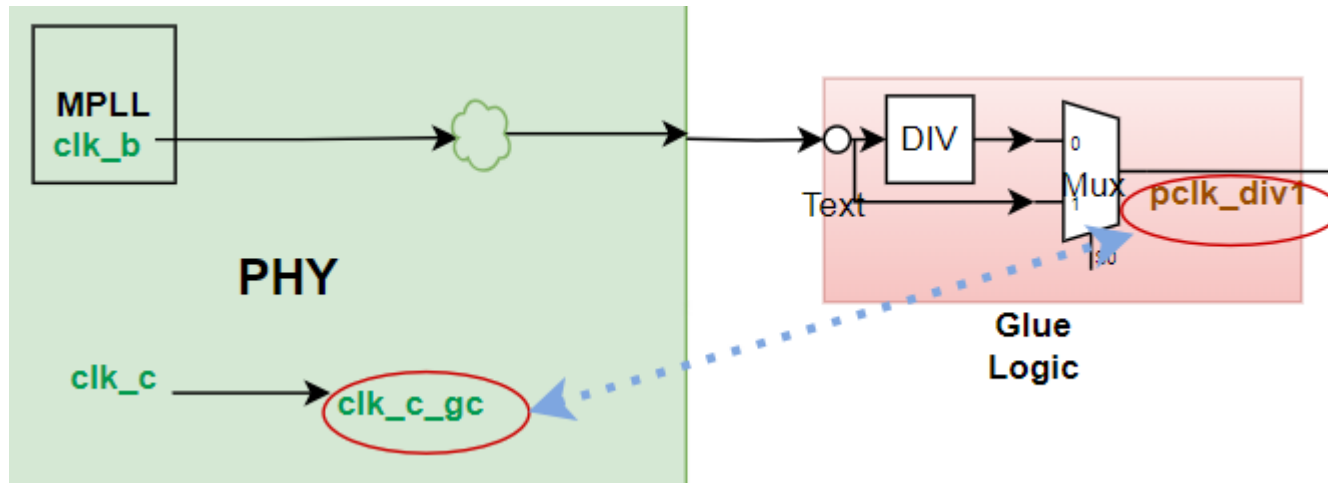
- G7: Run **Clock Group Generation** after TCM promotion

- Some of clock grouping is already handled in promotion flow
- Few clock grouping were missing which will be added by clock group generation step
- In absence of it, TCM verif will report below warning

Warning: There is an asynchronous clock group between master clocks **clk_b** and **clk_c** but generated clocks **plck_div1'** and **clk_c_gc'** are missing this constraint. (CLK-023)

- **With Clock_Group_Generation:**

- New async clock_groups created by clock_group_generation between the clocks present in different blocks

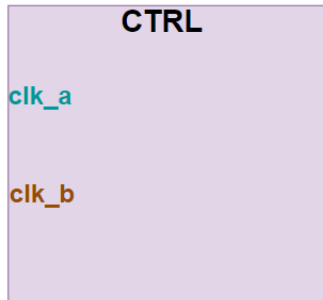


Guidelines:8/8

Run TCM verification after Promotion at Subsystem SDC- it can highlight important issues

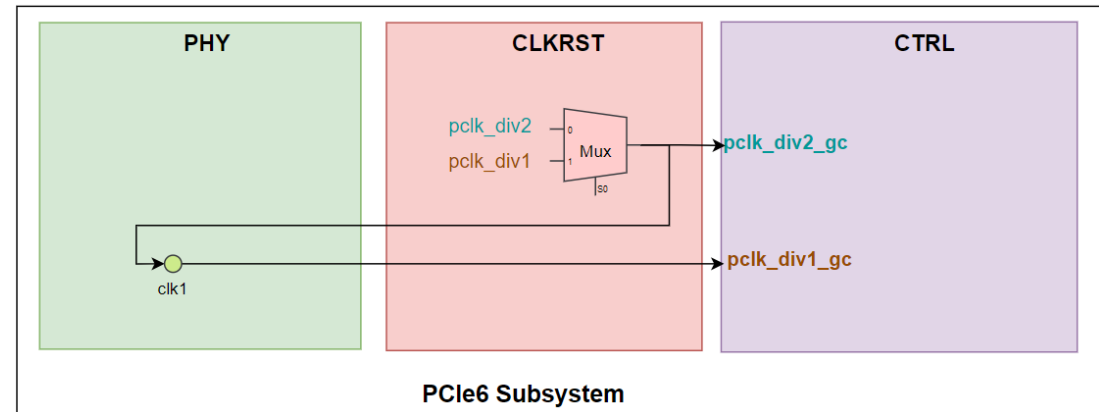
- G8: If at the block level, some clock_group is defined between clocks e.g. asynchronous
 - After promotion those promoted clocks become logically_exclusive by design
 - Then the promotion tool will not change clock group from async to logically_exclusive
 - TCM verification tool report the warning
Warning: Clocks are logically exclusive but they have not been specified as such. (CLK-022)
- e.g.
 - CTRL Block Level

async {clk_a} {clk_b}



After Promotion

- Logical exclusive {pclk_div2} {pclk_div1}
- Async {pclk_div2_gc} {pclk_div1_gc}



- Manual modification (Logical Exclusive {pclk_div2_gc} {pclk_div1_gc} in promoted constraints is required to take care of such a scenario based on design

How are the promoted constraints checked ?

TCM Equivalence is run to ensure promoted SDC is aligned with block SDC



TCM Verification is run on Subsystem SDC (Top SDC + Promoted SDC + Clock Group SDC) to check the quality of constraints

Conclusion

Conclusion : Pros/Cons

TRADITIONAL WAY	VS	TCM PROMOTION
Deep understanding of block SDC required		
Yes		No (understanding of top level clocking required)
Time to create Subsystem constraints		
More		Less
Efforts to create Subsystem SDC		
High		Less
Error probability		
More		Less
Manual modification is required to incorporate SS glue logic connectivity		
Yes		No
Manual modification of clock_groups to incorporate additional clocks		
Yes		Minor Nodification may need



Q&A

Our
Technology,
Your
Innovation™



THANK YOU

Other collaborators-

Utkarsh Pandharkar

Prabhat Yadav

Naveen Battu

Our
Technology,
Your
Innovation™