Improving the DriverClk [Physical] Frequency of System Level Emulation Models

Billa, Vikas
Kokonda, Nagesh
Kumar, Garlapati Anil
Mohd, Ishaq

Moninuola, Olufola A
Umair, Ali
Goveas, Melvyn

Intel India Pvt Ltd

Problem Statement

❑ DriverClk frequency

❖The DriverClk frequency is the clock rate at which the emulation hardware operates. It is a critical parameter set by the compiler software to ensure that the hardware emulation operates at the correct speed.

❑ Bigger the model slower the DriverClk

❖The size and complexity of the model being emulated can inversely affect the DriverClk frequency, with larger models typically requiring a slower clock to ensure accurate and reliable emulation.

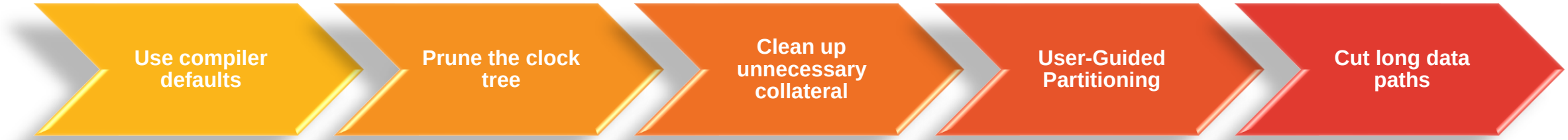
❑ Most model’s DriverClk get slower over their lifetime as functionality and Transactors are added to the model.

❖This leads to bottleneck in overall product validation as it impacts the emulation model speed

Solution

❖ Various techniques are discussed here to improve the overall emulation model speed.

1. Create clustering constraint
2. Break long combo paths (zGates)
3. Black box modules
   Enable minimal transactors as per requirement
4. Tie off signals

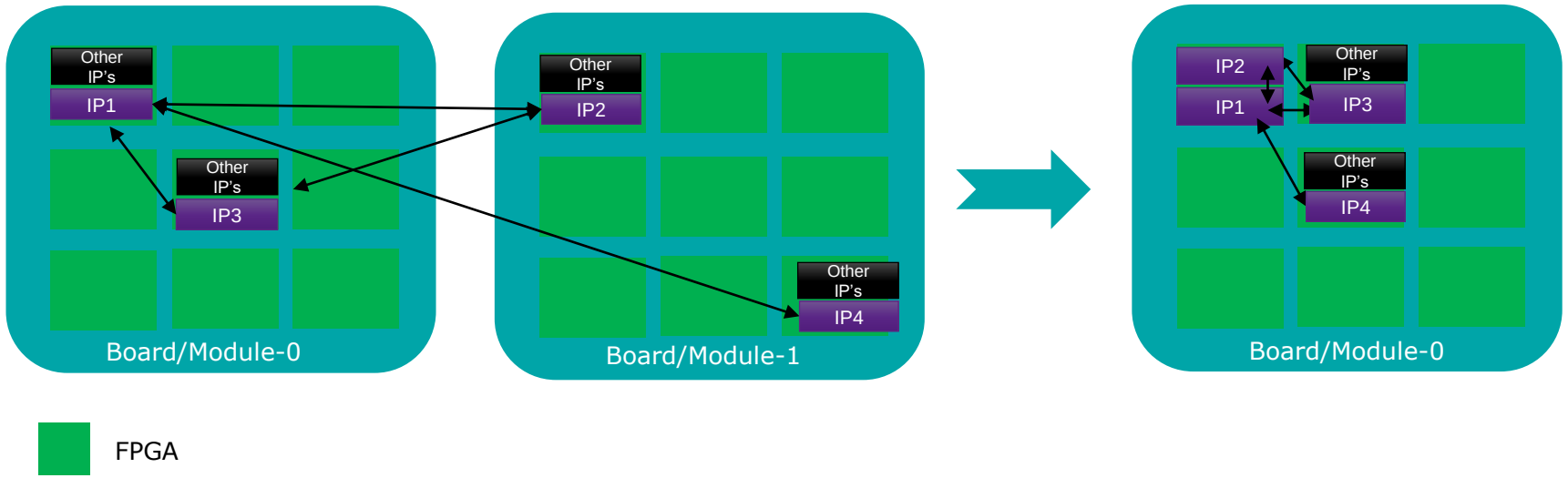


- Fill rates options, etc.
- Achieve full clock localization, reduce internal FPGA delay
- Black-boxes, tieoffs, zgates
- Xtors, trackers, forces, FWCs, $monitors, …
- Prevent IPs/sub-systems from being split across FPGAs / boards / units
- Typically for DFX models, using zgates

Grouping of Frequent Interactive IP’s

Grouping of Frequent Interactive IP's

- IP's that were interacting frequently was spread across different boards thus adding the respective delays in the critical path which impacts the overall Driverclk frequency of the model.
- Grouping of such logic to a single Board/FPGA will reduce the overall path delay.

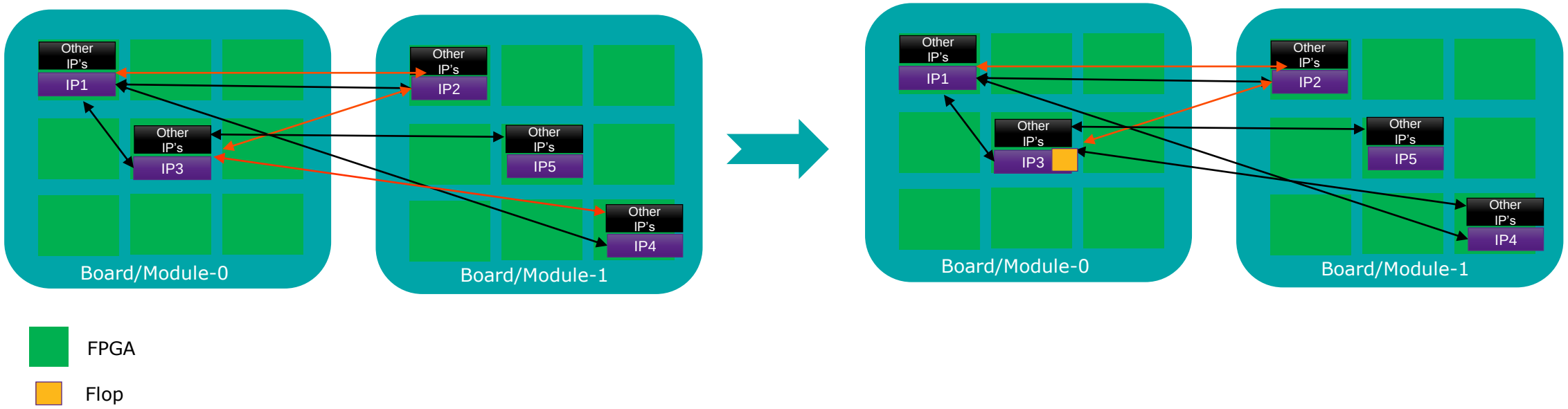



```
cluster_constraint -command ADD_GROUP -group_name AtomC0_arr -instance "soc_tb.soc.IP1"
cluster_constraint -command ADD_GROUP -group_name AtomC1_arr -instance "soc_tb.soc.IP2"
```

Adding Flops to Cutdown the Critical Path

Adding Flops to Cutdown the Critical Path

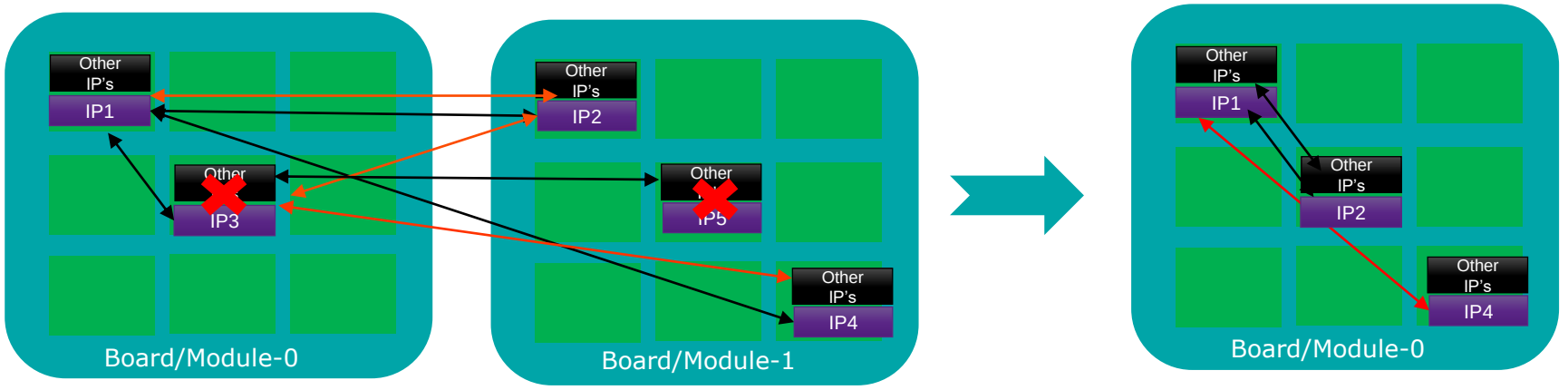
- A critical path is the longest path in the circuit that determines the minimum clock period or the maximum operating frequency of the design.
- Improving the critical path is a continuous and iterative process that requires understanding of the design. One such technique is to cutdown the critical path by adding a flop in-between.
- Utmost care should be taken such that the functionality doesn't break.



Removing unused IP’s or Transactors

Removing Unused IP's or Transactors

- Removing of unused IP's or Transactors will cut down the emulator board utilization [saving $ emulator costs] also improving the overall critical path.
- Utmost care should be taken such that the functionality doesn't break. Review and evaluate the results and seek feedback from design team while removing the IP's.

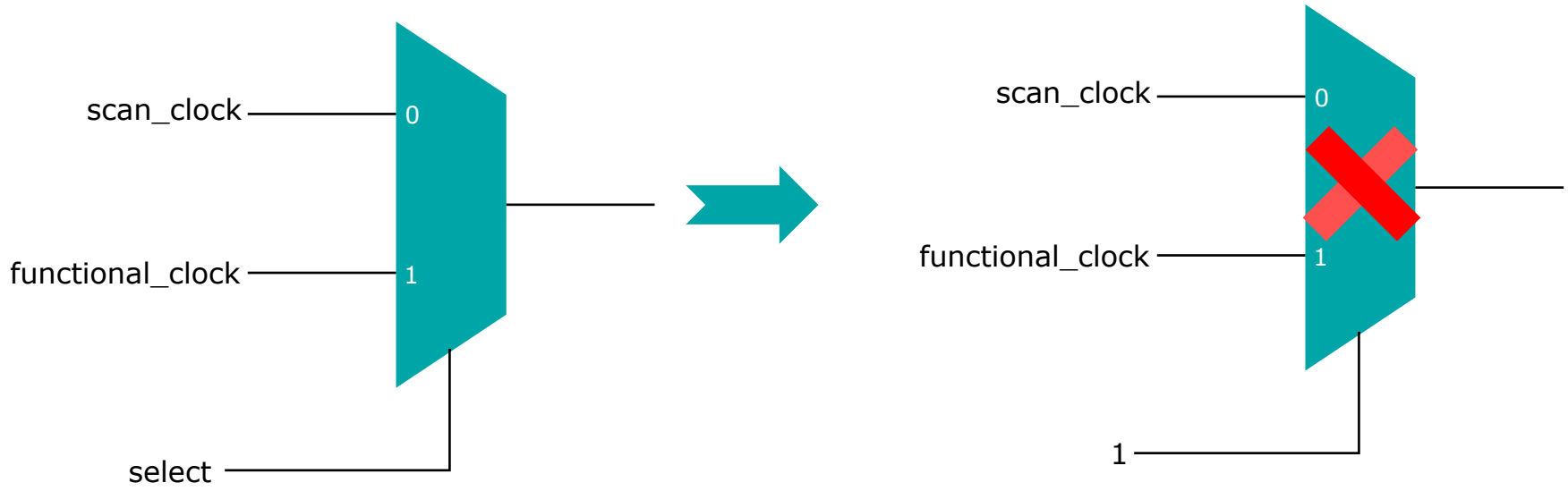



```
synthesis -full_blackbox {IP1}
synthesis -full_blackbox {IP2}
synthesis -full_blackbox {BFM}
```

Tying off clock muxes

Tying off Clock Muxes

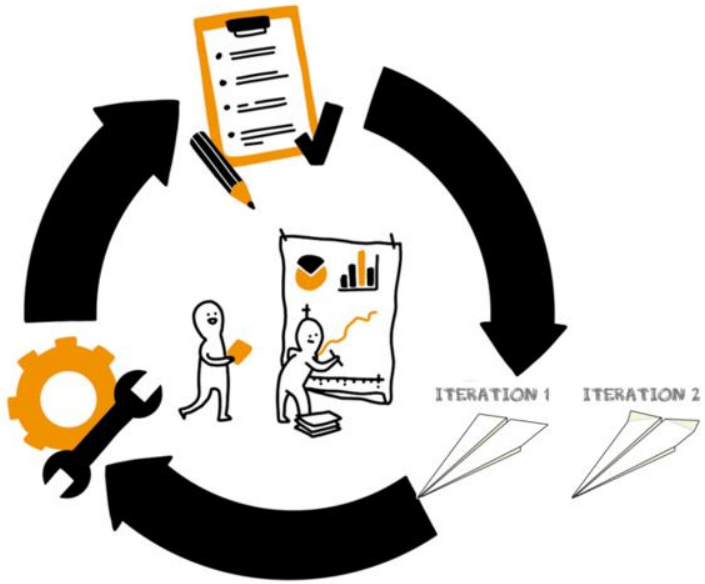
- A chain of multiplexers is used to select between functional and scan clocks. For non-DFX models, the select signal of the multiplexers can be tied off to a constant value such that functional clock is always picked.
- With this tie-off, the compiler is able eliminate the clock mux and further simplify the clock tree.



Challenges

Challenges

- Design Knowledge is required for doing perf optimizations else will be landing in doing more build iterations.



Overall Improvement

Overall Improvement

- All improvement at a glance
  - ❖ Grouping of frequent interactive IP's helped in cutting down the path delays.
  - ❖ Adding Flops to Cutdown the Critical helped in improving the critical path.
  - ❖ Removing Unused IP's or Transactors helped in cutting down the board size and saving $ emulator cost.
  - ❖ Tying off Clock Muxes helped in simplify the clock tree.

- Combination of above techniques resulted in impressive improvement in DriverClk.

Model [ZSE5]	Original DriverClk [KHz]	Improved DriverClk with above Optimizations [KHz]	Improvement
Model 1	700	2222	~3X
Model 2	600	1480	~2.5X

Key takeaways

Key takeaways

- Improved Driverclk of the model helps in doing multiple debug test runs.

  The time to run the same test now takes lesser time with this Driverclk techniques.

- Saving $ emulator costs, helps in the project budgeting.

  This Driverclk techniques helps to reduce the usage of emulator time(it means running the design at high speed) with same design size, thus saving the dollars for the organization.


THANK YOU
YOUR
INNOVATION
YOUR
COMMUNITY