

# Design-Closure of Multi-Million Gate Chip Using Full Flat Optimization Technology of Olympus-SoC

Pooja Mehra\* Johann Meleard\*\* , STMicroelectronics \*India \*\*France

{ [pooja.arora@st.com](mailto:pooja.arora@st.com) , [johann.meleard@st.com](mailto:johann.meleard@st.com) }

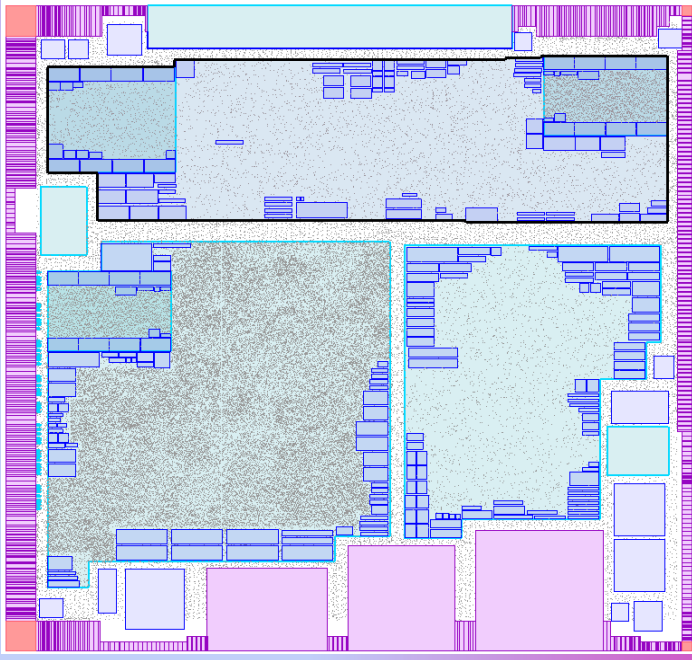
## Introduction

- With ever increasing complexity & more -n- more features being squeezed into single chip, die sizes are increasing phenomenally
- Due to huge design size and with increase in number of modes & number of corners for timing closure, raises the possibility of cross mode setup-hold conflict
- Multiple modes and corner are contributing to large number of ECO cycle required for timing closure
- Cycle times are dramatically increasing, with increase in eco loops , which is impacting time-to-market schedule

So Needed solution which can reduce design closure cycle time

## Device In Discussion

- Set-top box decoder that integrate an MPEG-4 decoder
- SOC designed - 65nm process node & shrunk to 55nm
- Contains 4.5 million instances
- 3 partition units
- 311 macros/memories
- Has 3 Cores
- 640 IO's (360 functional pads )
- 125 clock domains
- 4 operating modes & 3 process corners



## Challenges

- Inaccurate timing models used in existing hierarchical timing closure methodology, in which the chip was partitioned, the partitions implemented, and then instantiated at the chip level as abstraction models for top-level timing signoff
- Need to rely on pessimistic timing budgets allocated during chip budgeting, which often caused I/O & inter-partition timing violations
- Due to the inaccurate timing models, 'flat' chip-level clock tree synthesis couldn't be performed to meet very precise skew and latency targets. This required iteration during timing closure phase
- Handling MCMM timing in our flow was also a challenge. The block implementations were done for best/worst case timing in 2 modes, but signoff timing had to met for several other modes and corners. Therefore, we usually found some 'cross corner' or 'cross mode' violations during signoff that were not seen during implementation phase. Analysis and fixing these through ECO loops was possible but can take several iterations and engineers, also it is expensive from a schedule standpoint
- Tool's capacity to handle 18 million gate design 'flat' is one of the limitation which is forcing us to opt for hierarchical timing closure methodology

## New Design Flow

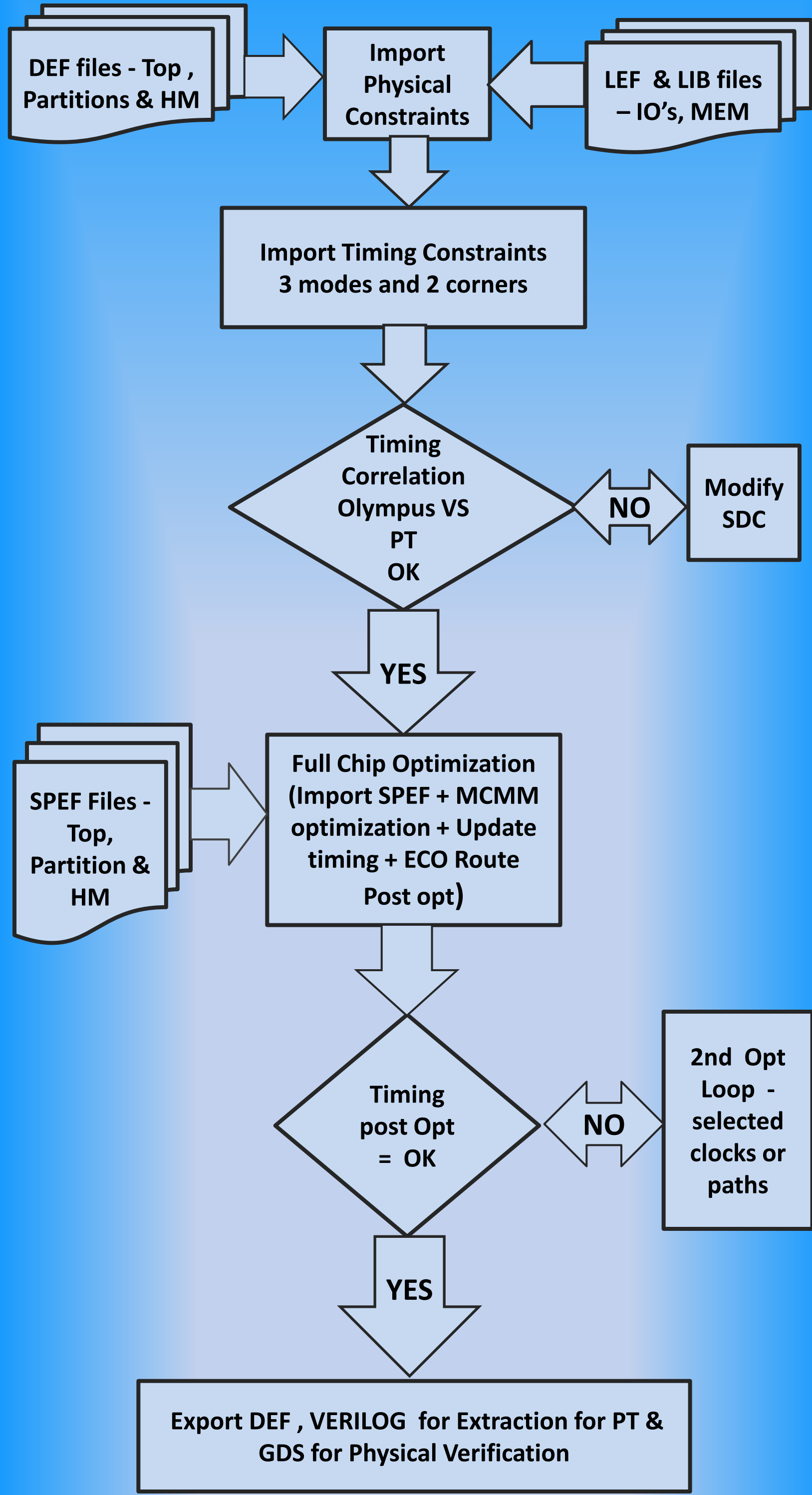
We needed a solution to improve the MCMM closure, allow a full-chip optimization platform without changing logical partitioning, can integrate seamlessly into existing methodology without much perturbing of existing flow, and do the chip assembly and finishing with fewer sign-off iterations and fewer resources.

This is where Mentor Olympus-SoC came in

### Full Chip Optimization Methodology

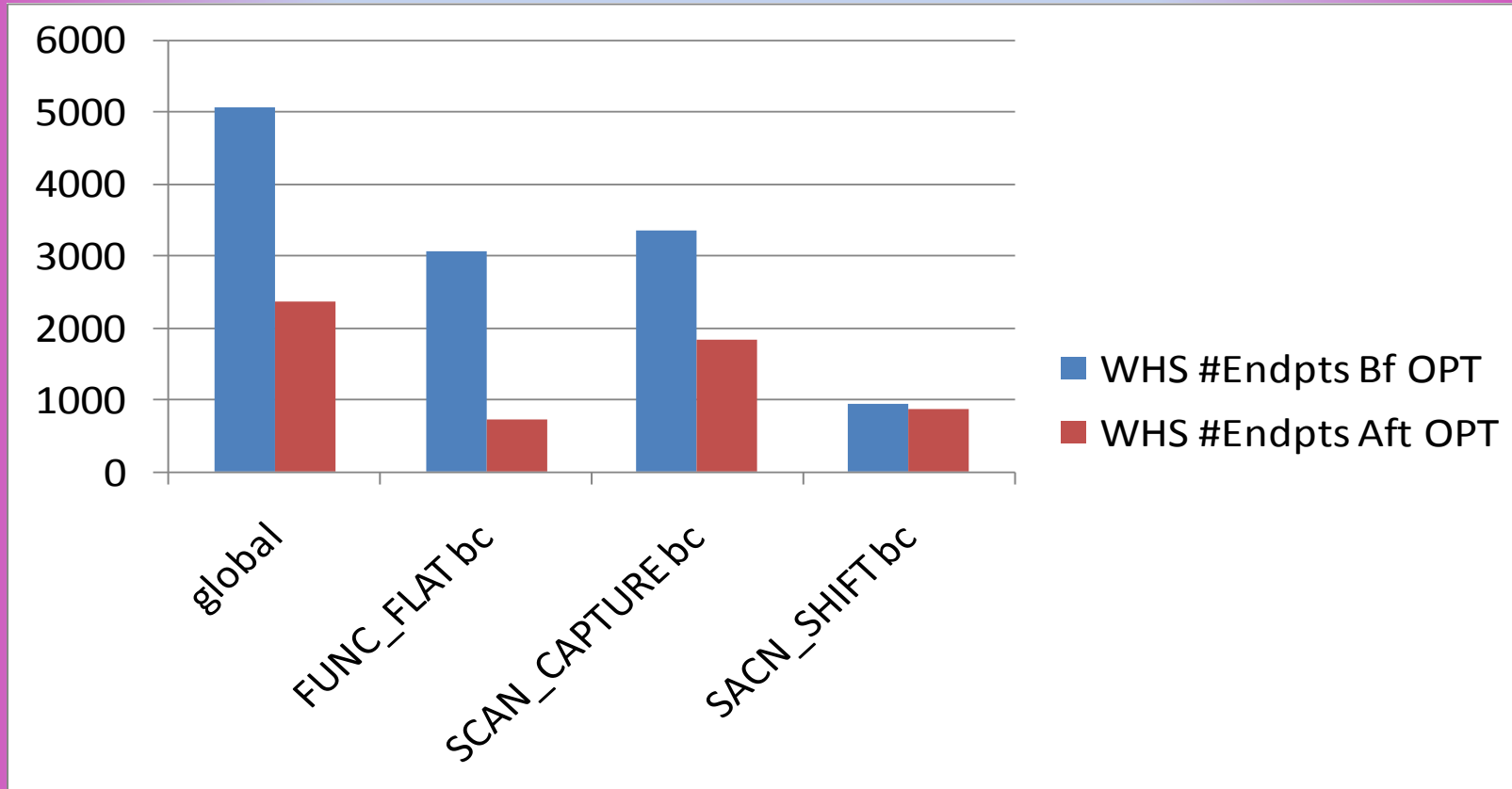
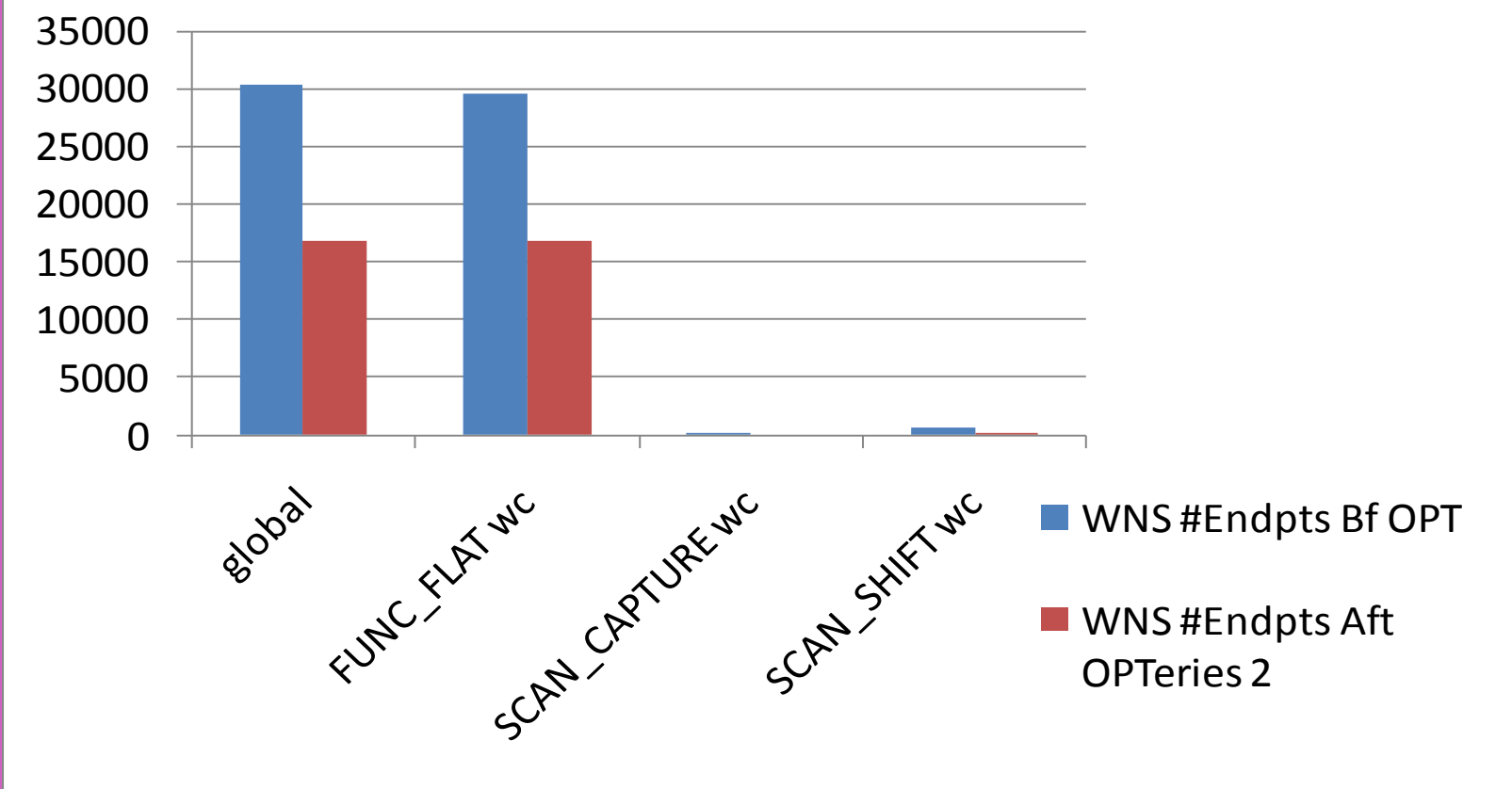
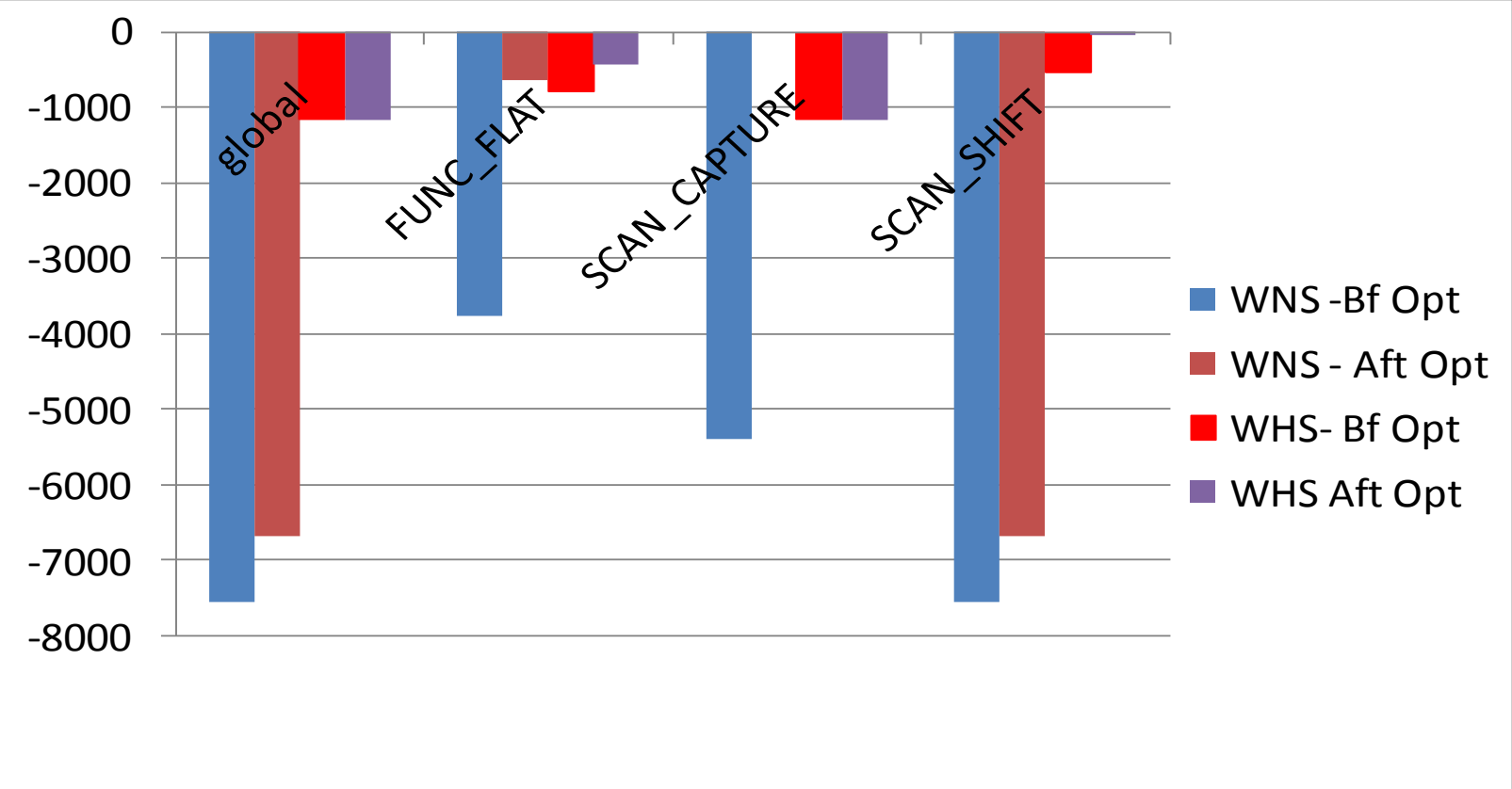
- Top and Partitions were placed & routed in SoC-Encounter and then ported to Olympus-SoC for full-chip MCMM optimization. In SoC-Encounter functional and scan-shift mode were only optimized for timing
- Only 3 modes and 2 corners were defined for optimization in Olympus-SoC
- Optimized design at top level including IO's , Top and block level concurrently
- Maintained design logical hierarchy and partition boundary while doing full flat timing analysis
- DEF, Verilog and SPEF were used for some of the don't touch HM for optimization, to have full picture of clock source point
- Freeze Clock Network & HM for which DEF , Verilog were defined to avoid any modification while optimization
- Setup, Hold, transition, capacitance all were optimized concurrently
- First optimization was white space driven, where placed cells movement was not allowed
- Dummy Filler cells were defined for block halos as Olympus doesn't understand def blockages defined for block halos

## Flow Chart



## MCMM Variability Report

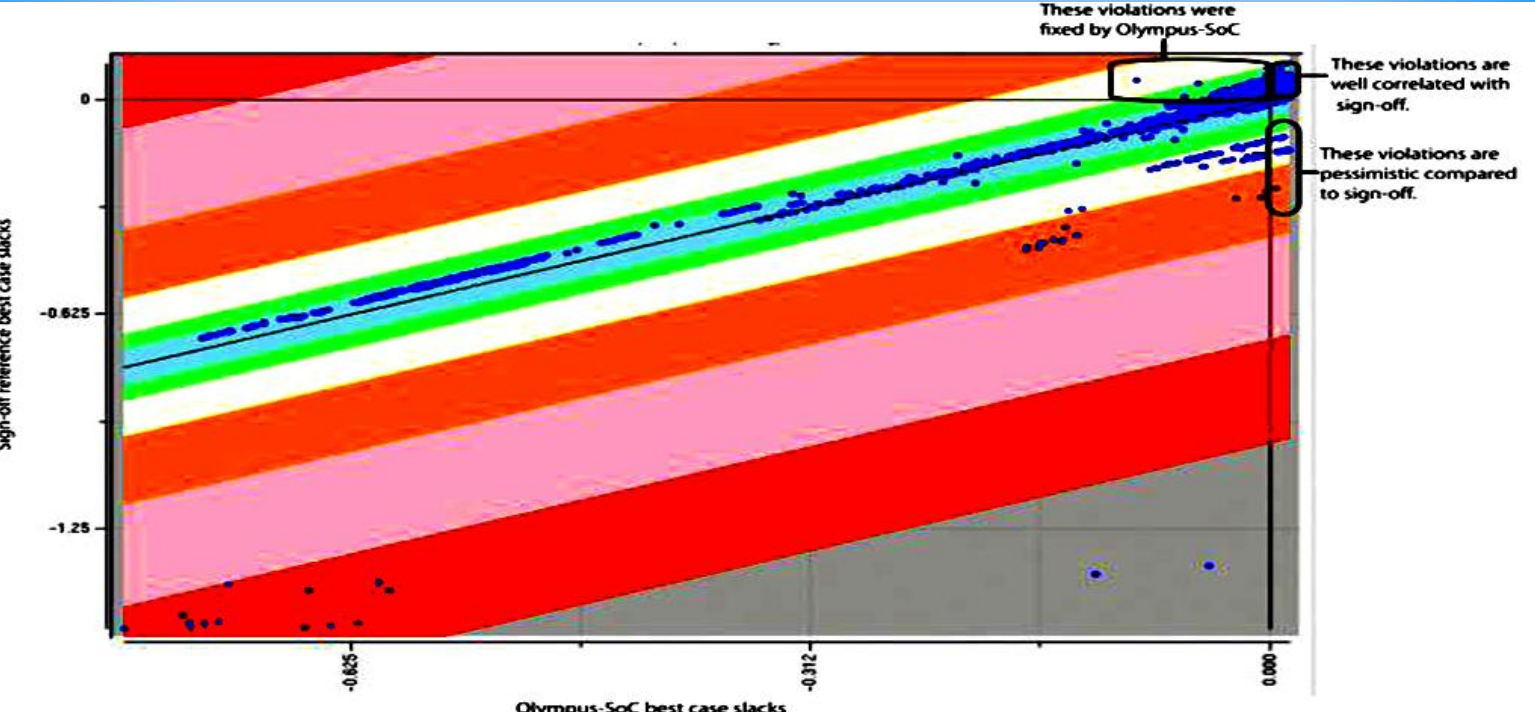
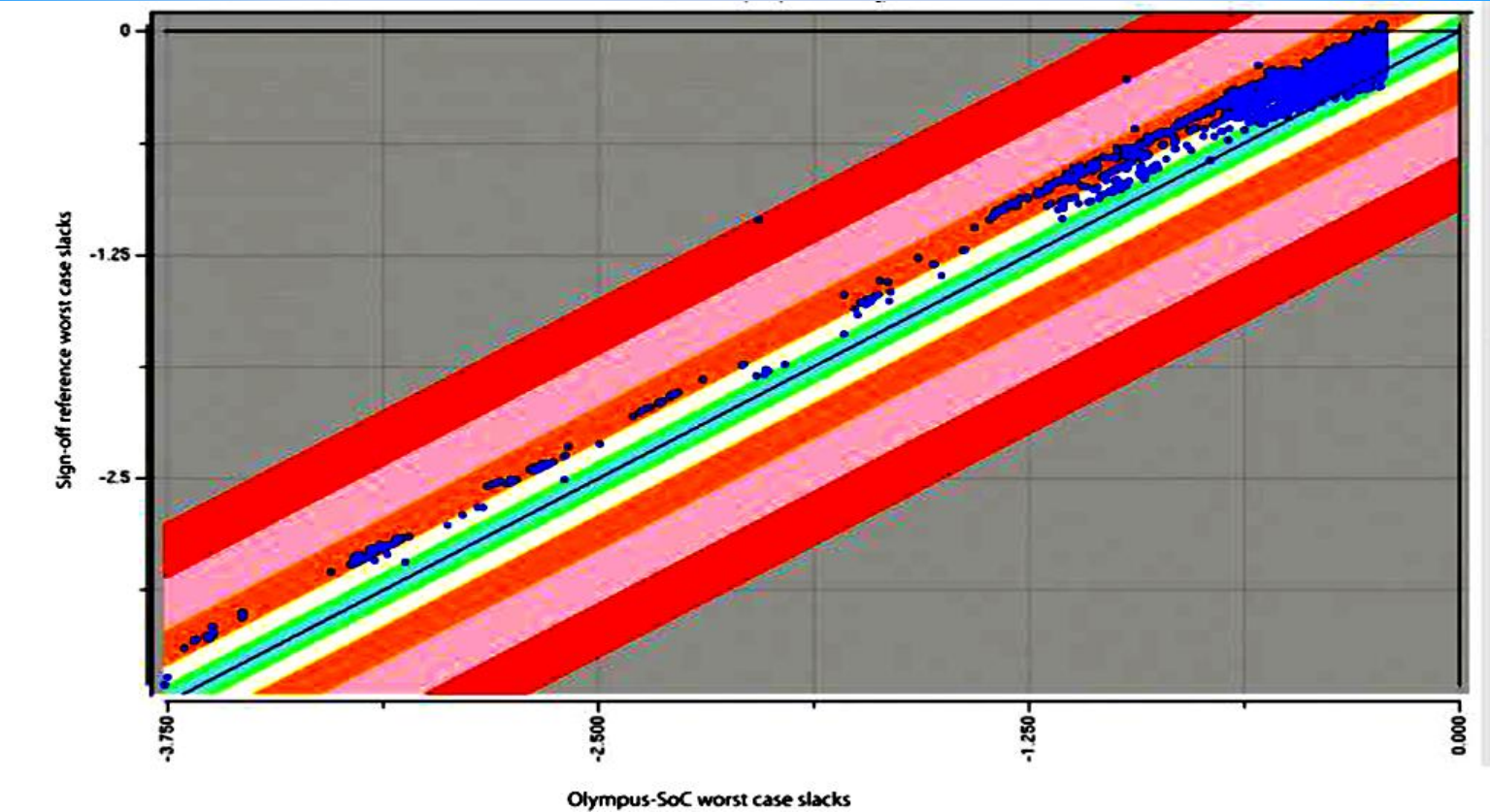
For the MCMM optimization, we set up the mode-corner combinations information upfront, and the tool quite smoothly and automatically optimized for all the relevant scenarios. At the beginning of the full flat optimization flow, we generated a variability report from within Olympus-SoC, which showed the initial timing situation for blocks in the full chip context. The following table shows the initial vs final variability report:



## Tool Setup & Timing Correlation

We performed timing correlation between our sign-off tool and Olympus-SoC. The results showed a very close correlation with Olympus within 10% of the sign-off reference slack calculations.

Range	Color
0.000 - 0.050 ns	Light Green
0.000 - 0.100 ns	Light Green
0.000 - 0.200 ns	Yellow
0.000 - 0.400 ns	Orange
0.000 - 0.700 ns	Red
0.000 - 1.000 ns	Dark Red



Signoff vs. Olympus-SoC timing correlation scatter diagram for worst case (top) and for best case (bottom). Signoff reference slacks are shown along the Y axis, Olympus-SoC slacks are on the X axis.

## Full Chip Path Highlighted

Olympus-SoC was able to load and process our full 18 million gate design full flat, while maintaining the designs logical hierarchy.

We liked the fact that timing modeling of partitions/blocks was not a requirement for chip assembly. We needed the freedom to choose which blocks to abstract, depending on the top-level paths and the level of accuracy needed. For example, we might abstract a block that is cloned heavily, while a block that is not cloned can remain un-abstracted, giving us greater accuracy in chip-level timing measurements. This is particularly important when optimizing inter-block paths because the tool has the ability to see the entire path end-to-end.

The following Figure (a) shows the chip with an inter-block path highlighted.

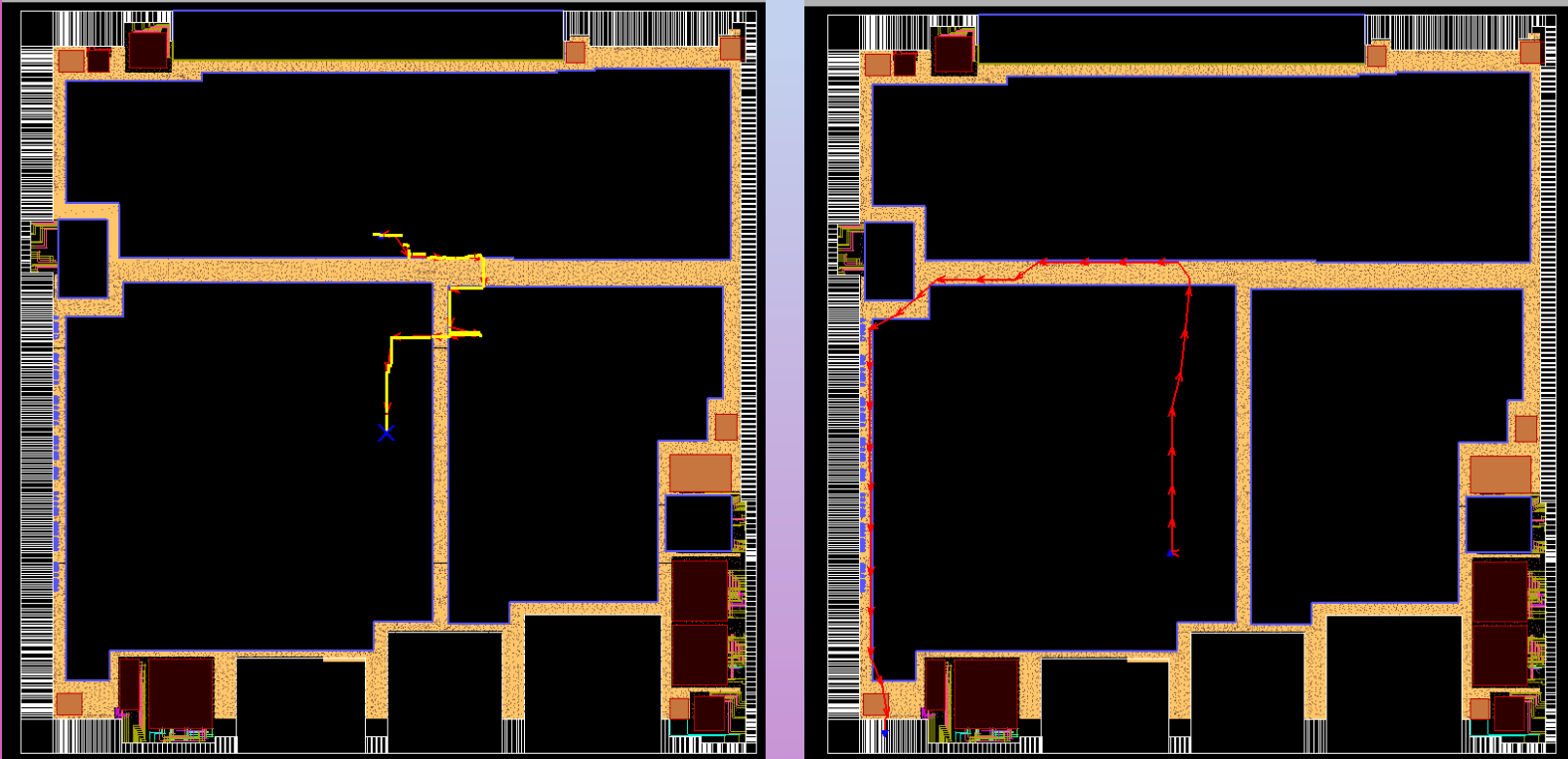


Figure (a)

Figure (b)

Olympus-SoC also allowed us to optimize the design at the top level (including IO), and the block level at the same time, full flat. The above Figure (b) shows Olympus capability to optimize partition to I/O paths.

As the block-level boundaries continue to be maintained, any optimization-logic changes made to the blocks during flat optimization were directly updated inside the blocks. This type of dynamic updating in-situ is only possible with a tool that has the capacity to support a flat full-chip view while respecting logical and physical hierarchy.

## Benefits

- MCMM full chip optimization flow of Olympus-SoC helped reducing time required for timing closure while respecting logical hierarchy for timing and physical hierarchy for routing.
- Shielding of clock nets is possible on full chip routed database
- Incremental DEF allows updating changes related to pre-route/Custom routing (e.g. analog routing) or pads modification which is done after moving to Olympus from SoC-Encounter tool.
- Manual ECO implementation is easy to execute using change file.
- MCMM TA convergence in 3 days using multi-cpu (previously two weeks)
- Its capability of handling extremely large design with small memory footprint enables integration of this tool seamlessly into the existing design flow.
- Compute Statistics are reasonable

STEP	TOTAL MEMORY	CPU TIME
Import Physical Const.	23 GB	1.30 hr
Reading Timing Const.	26 GB	½ hr
Full Chip Optimization	44 GB	40 hr
Step Included in Optimization (This opt. is with 3 modes and 2 corners+ setup, hold, transition and capacitance fixing concurrently)		
Import SPEF (Partition, Top, HM)		3hr

## Potential Tool and Flow Improvement

We derived good benefits by using Olympus-SoC in our flow. However, introduction of a new tool into any flow always comes with some challenges. For example:

- We were not able to close timing for two tricky clock domains. This was on account of a floor planning limitation; however the tool could have given some pointers or metrics on the level of difficulty it was facing
- We also found that some pad delays differed between the sign-off tool and Olympus-SoC. This issue is now fixed. For some of the pads Olympus-SoC timing calculations were slightly more optimistic than signoff tool and for other IO paths, Olympus-SoC timing were pessimistic so we masked all IO paths/violations for some of the modes, which were later fixed during an ECO loop
- During manual last-mile DRC/XTALK fixing, we think that manual wire editing could be more users friendly and be more dynamically aware of violations caused during editing. This manual fixing consumed lot of time in the end
- In the final GDSII stream-out, the tool had some problems with unification and understanding the empty GDS of hard macros. This problem was fixed before final GDSII stream out
- Routing over a particular Hard Macro where metal blockage was defined. This caused "short" which was discovered and debugged at costly signoff LVS stage
- Tool crashes unpredictably while doing manual DRC fixing in GUI mode

## Result and Conclusion

Using Olympus-SoC for full-chip optimization and chip finishing gave good improvement over our previous timing closure approach. Therefore, we may consider continued use of Olympus-SoC. It was easy to setup and the correlation with sign-off tool was good throughout the entire flow. On account of its capacity, relatively small memory footprint , MCMM timing closure capability, and the accuracy, we achieved the previously 2 weeks TA convergence step in less than 3 days using multi-cpu machine. Notably, we saved time in signoff cycle by being in a position to run full chip flat , MCMM optimization that significantly reduced the number of ECO iterations required for closure

