



SHAPING THE NEXT GENERATION OF ELECTRONICS

JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA





JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

An Effective Hierarchical STA solution for closing large SoC design

Shourya Shukla, Marvell Technology

Sushant Hajare, Marvell Technology

Sainarayanan Karatholuvu Suryanarayanan, Marvell Technology

Prasanjeet Das, Cadence Design Systems

Harshit Jaiswal, Cadence Design Systems

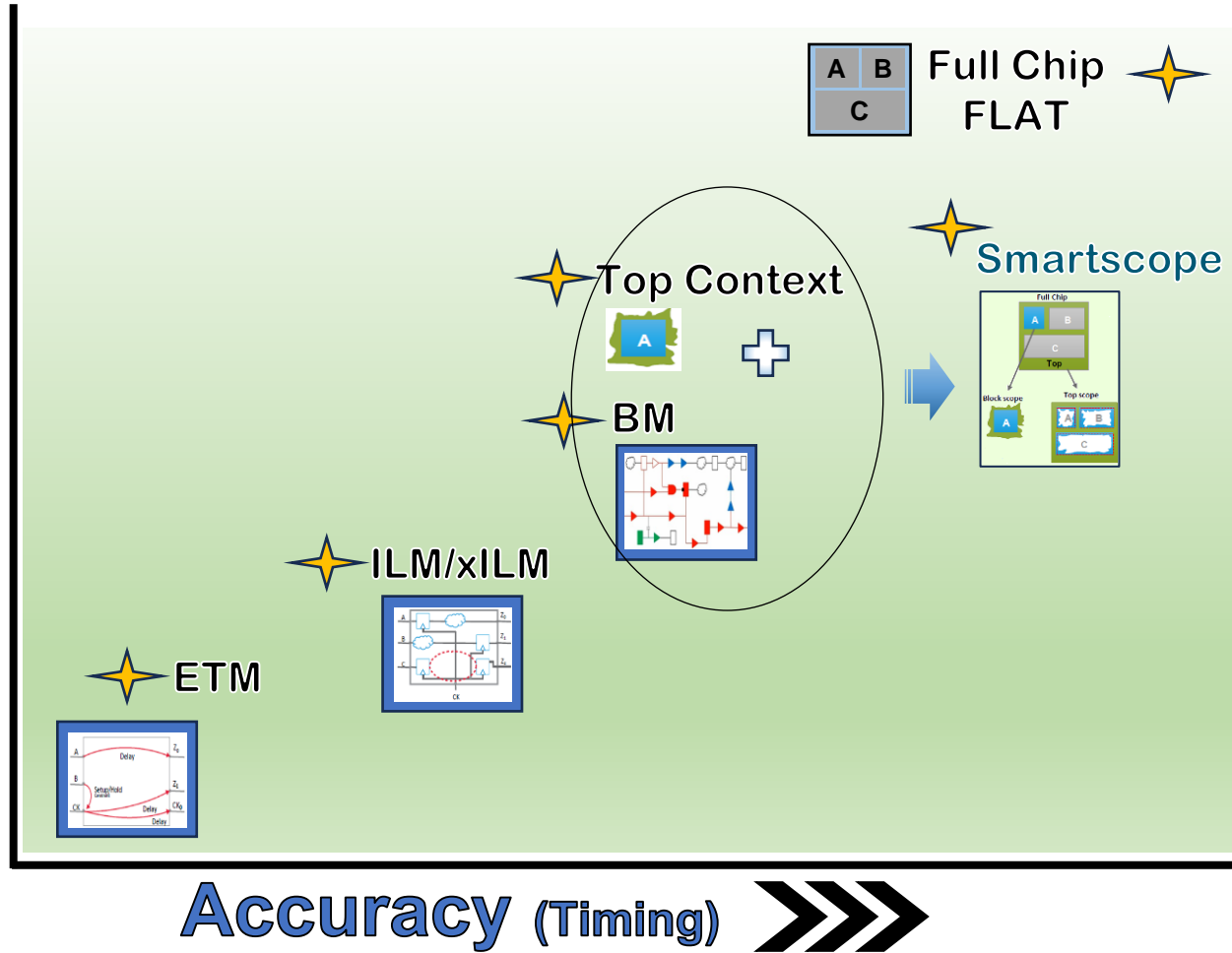
Sharath AC, Cadence Design Systems

Nitin Jain, Cadence Design Systems



Motivation

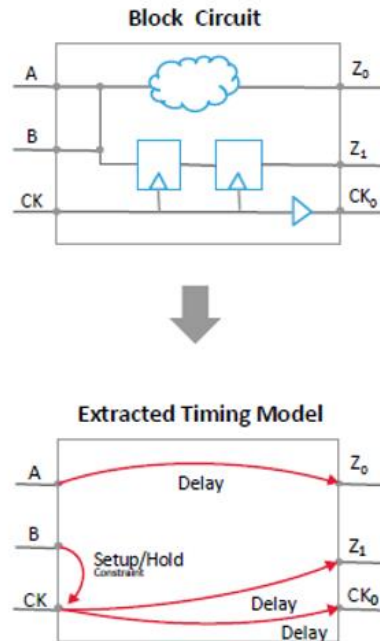

**Compute
Resources**
(TAT & Memory)



- ❖ Choice for Static Timing Analysis (STA), Flat or Hierarchical depends on design size & resources
- ❖ Extracted Timing Models (ETM): Least accurate
- ❖ Interface Logic Models (ILM)/extended Interface Logic Models (xILM): Slightly more accurate than ETM
- ❖ Flat: Most accurate
- ❖ With accuracy comes cost for Turn-around Time (TAT) and memory
- ❖ Adoption of an optimal hierarchical solution with:
 - Accuracy as that of Flat
 - Resource usage as that of ETM

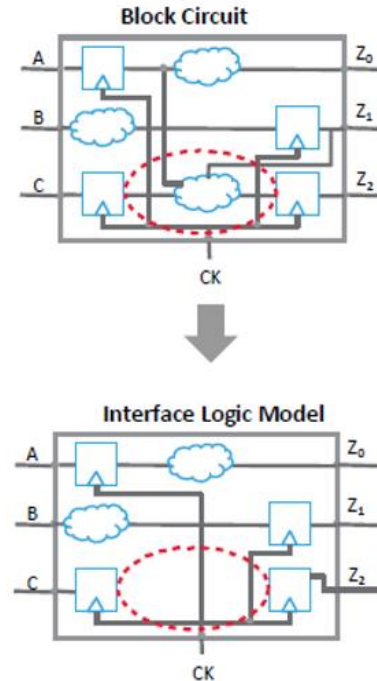
Overview of Tempus Block Abstraction Techniques

Extracted Timing Model



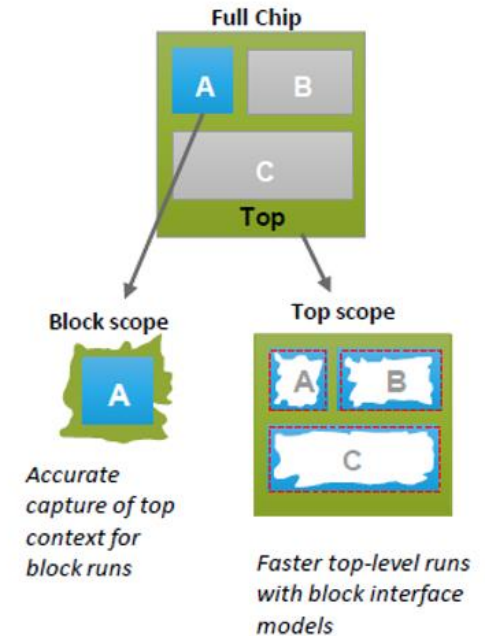
- Arc-based .lib format

Interface Logic Model



- Netlist-based interface model
- Internal logic is abstracted away

SmartScope Model

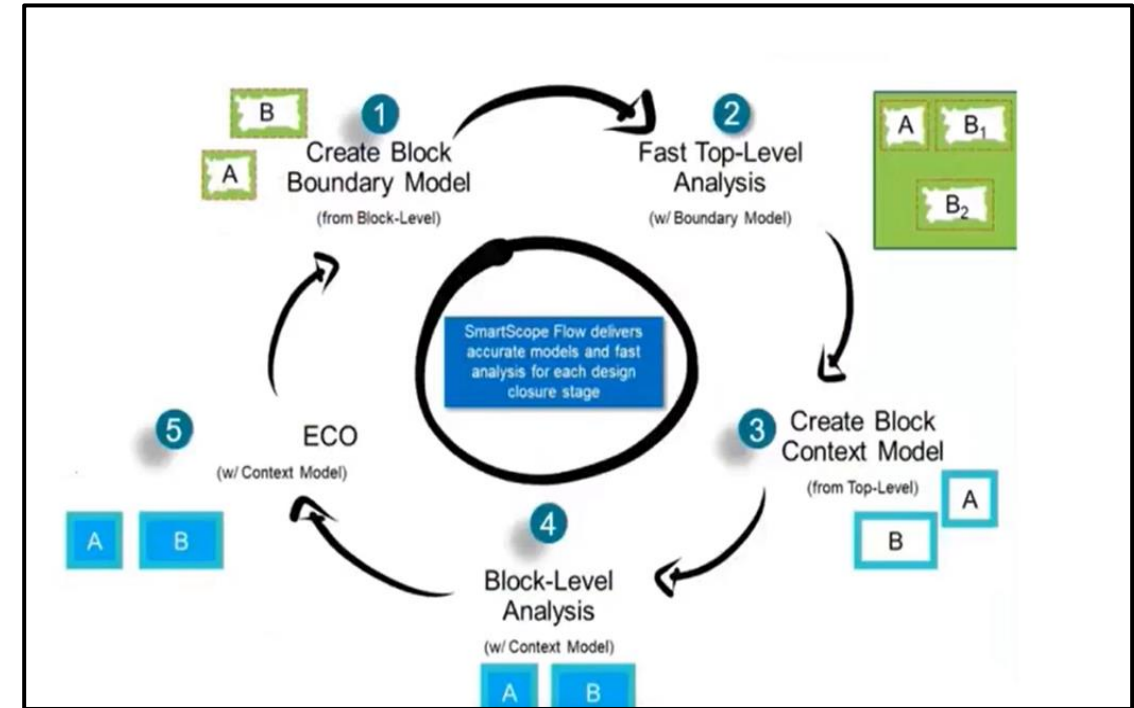


- Netlist-based model
- Include all logic & timing context relevant to block or top scope

Main Idea

Smartscope Flow Implementation

- ❖ Generate block boundary model (BM)
- ❖ Read block boundary model in top hierarchical design and perform STA analysis at top level
- ❖ Generate block timing context data for block from top hierarchical design where block's boundary model is read
- ❖ Read block context timing data and perform STA analysis and perform block register to register timing path (reg2reg) closure using context data
- ❖ Run block level ECO's for block level reg2reg path closure and generate boundary model for the block
- ❖ Perform top level Hierarchical STA (HSTA) analysis by reading block's boundary model from previous step

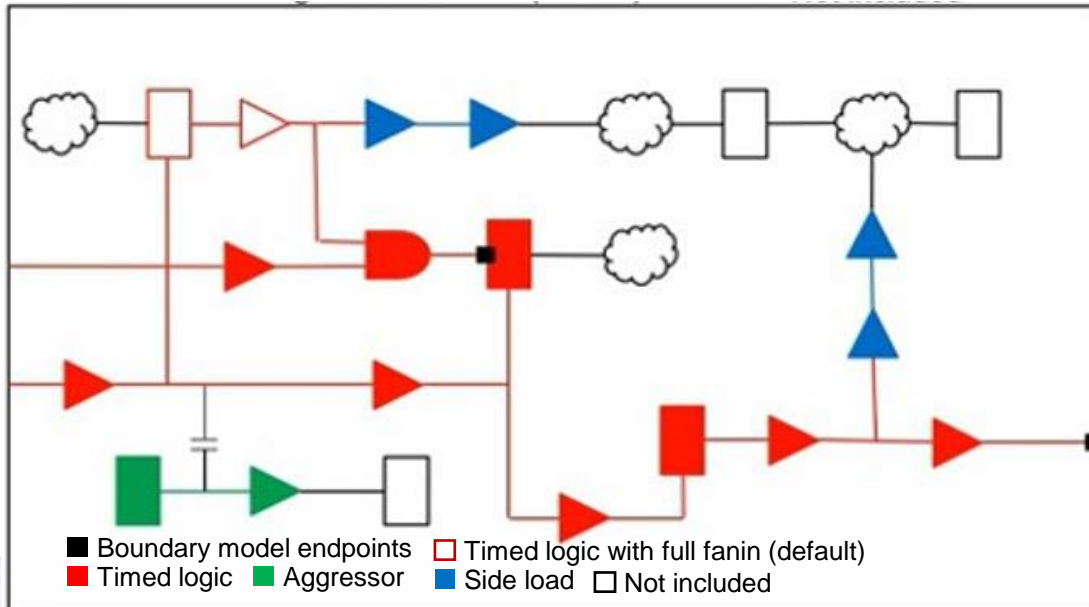


Step by step representation of the Smartscope flow

Smartscope Hierarchical Flow

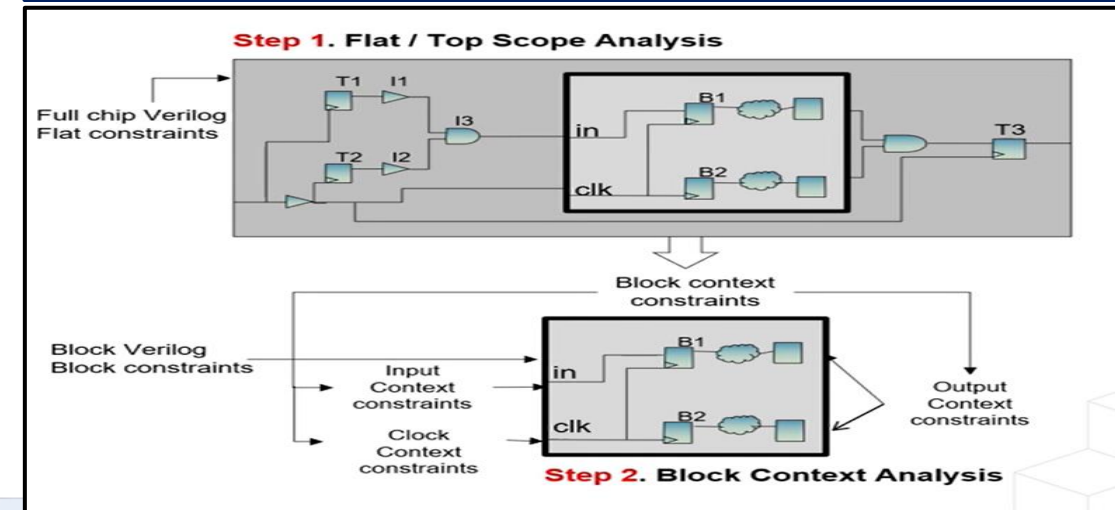
Boundary Model Flow

- ❖ STA with block abstraction
- ❖ BM netlist of the model in binary form:
 - Interface paths
 - Extended fanin
 - Internal attackers
 - Side load receivers
- ❖ Parasitics in the Standard Parasitic Exchange Format (SPEF)
- ❖ BM model does not include block internal timing paths
- ❖ Timing context information to improve accuracy



Hierarchical Context Analysis

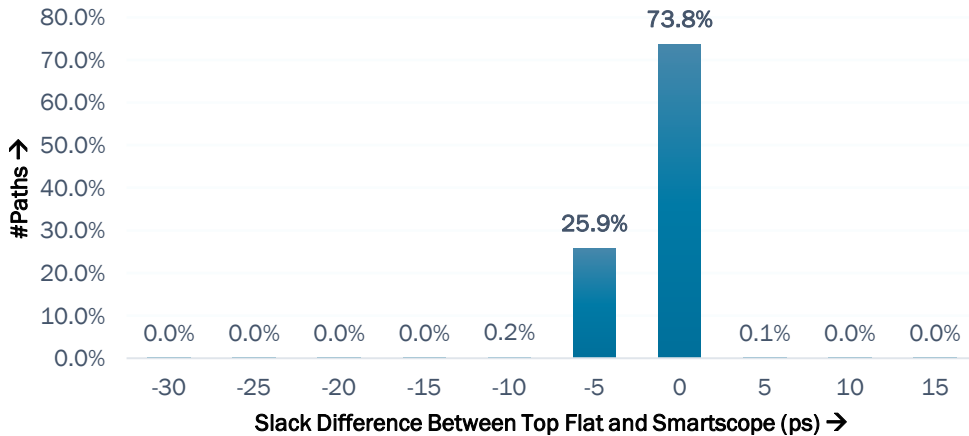
- ❖ Model accurate block boundary constraints seen from its top STA environment
- ❖ Block Timing Constraints contains:
 - Arrival time of data signal entering input port of block
 - Required time of data signal leaving output port of block
 - Clock latencies of clocks entering block clock ports
- ❖ Closure for block reg2reg paths
- ❖ Two step flow:
 - Create context model of block from top level
 - Read context model in block STA run



Results

Design Info:

- ❖ Technology Node: 5nm
- ❖ Instance Count: ~91M
- ❖ Total Child Blocks: 15
- ❖ 10 Single Instantiated Module (SIM)
- ❖ 5 Multiple Instantiated Module (MIM)
- ❖ Setup Corner: SSGNP_0p765v_m40c_cworstCCwTn40c

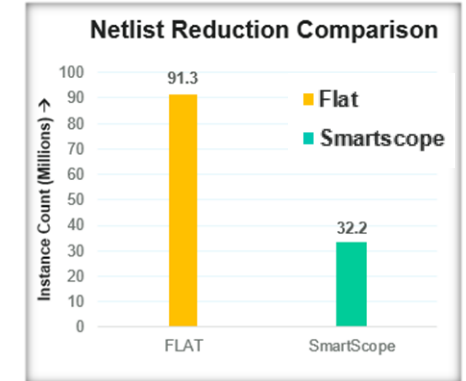


Timing Correlation: Smartscope shows good correlation with top flat in terms of timing slack for ~91M design

Overall, **99.7%** of paths lie within -0.005ns -> +0.005ns slack range for setup corner

Netlist Reduction →

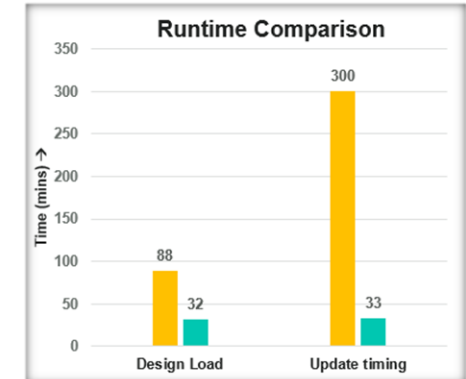
Total amount of netlist reduction with Smartscope when block BM netlist for all blocks used is **~65%** when compared to top flat



Runtime Reduction →

Overall, **~83%** runtime reduction with Smartscope when compared to top flat STA

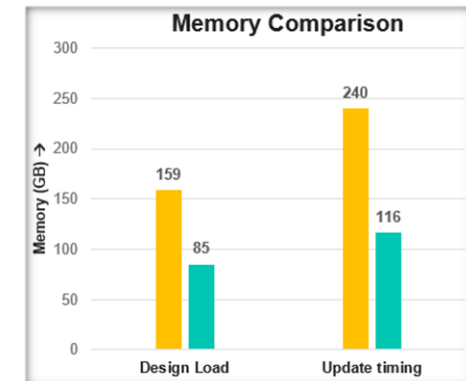
Bar graph represent major runtime benefit across main stages



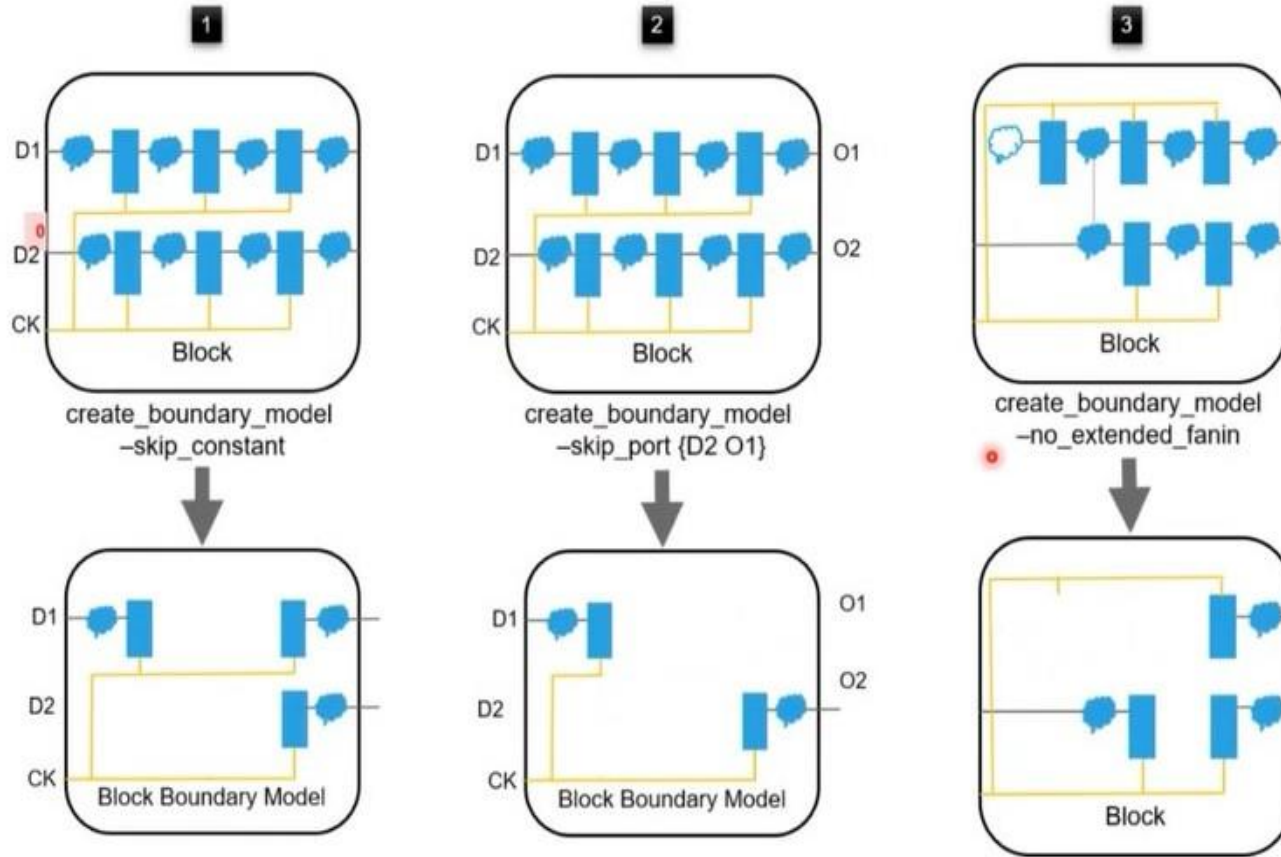
Memory Reduction →

Overall, **~50%** memory reduction with Smartscope when compared to top flat STA

Bar graph represent major memory benefit across main stages



Boundary Model Netlist Optimization

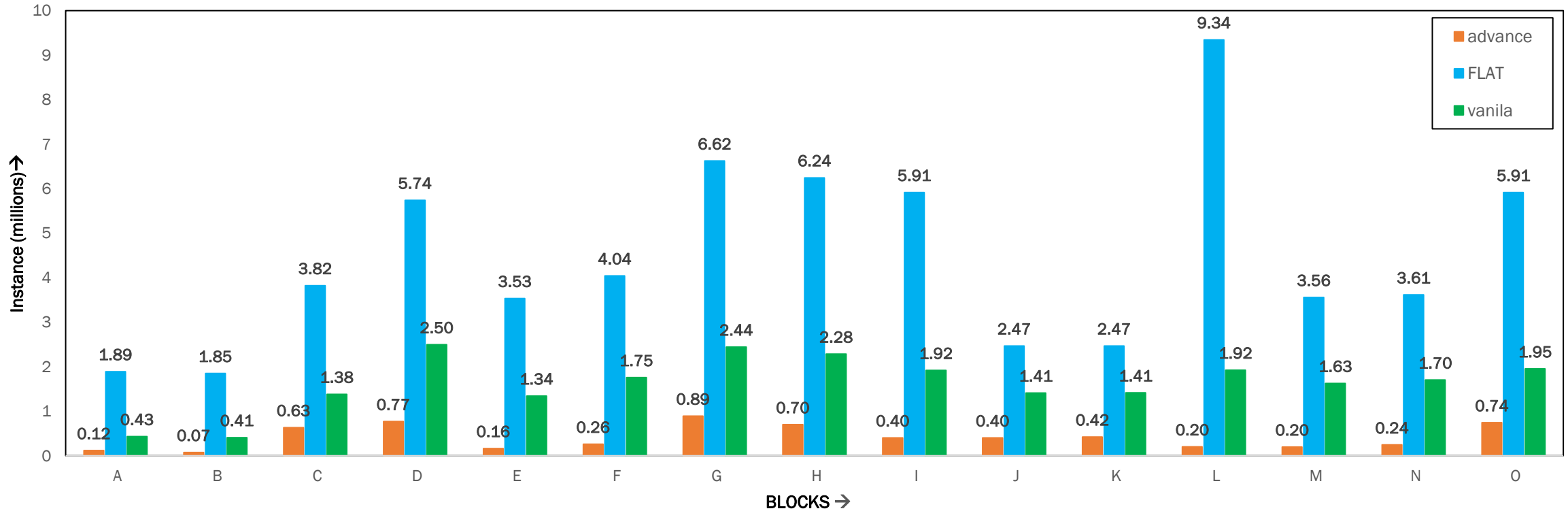


Options to optimize size of BM netlist

- Skip BM generation for ports which have case specified (skip_constant)
- Skip BM generation for user specified ports (skip_port)
- Skip preserving extended fanin (no_extended_fanin)

Results: BM Netlist Optimization

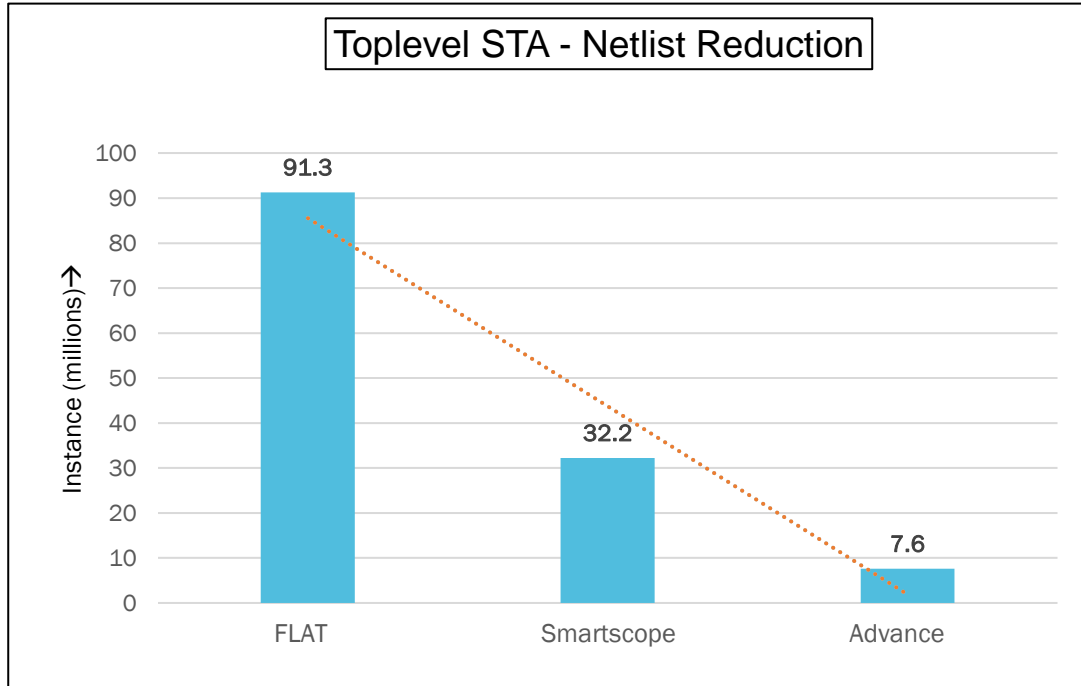
Block Level Netlist Reduction



With advance settings:

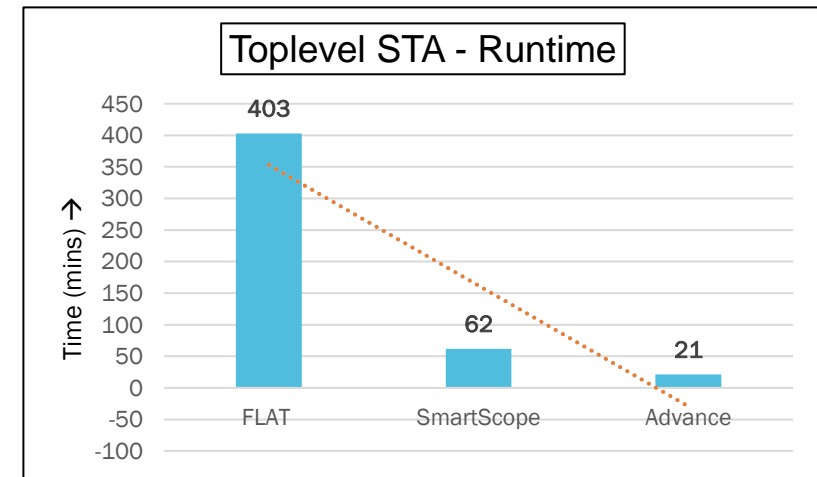
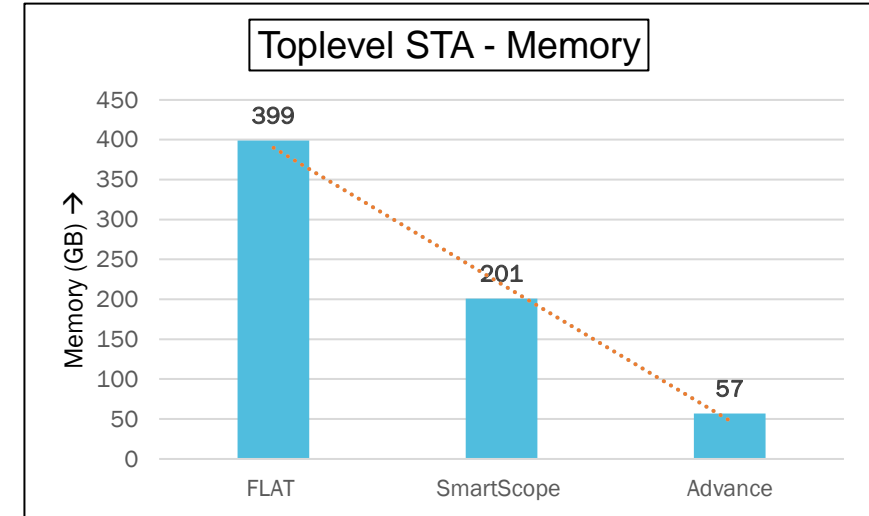
1. Average reduction in block level netlist ~92% of total flat netlist
2. ~25% more reduction in netlist than conventional Smartscope HSTA

Results: BM Netlist Optimization (contd.)



Advance Smartscope compared to Flat STA:

- ❖ Netlist: ~91% ↓
- ❖ Memory: ~86% ↓
- ❖ Runtime: ~94% ↓



Runtime, Memory & Timing Comparison (Advance command)

Block

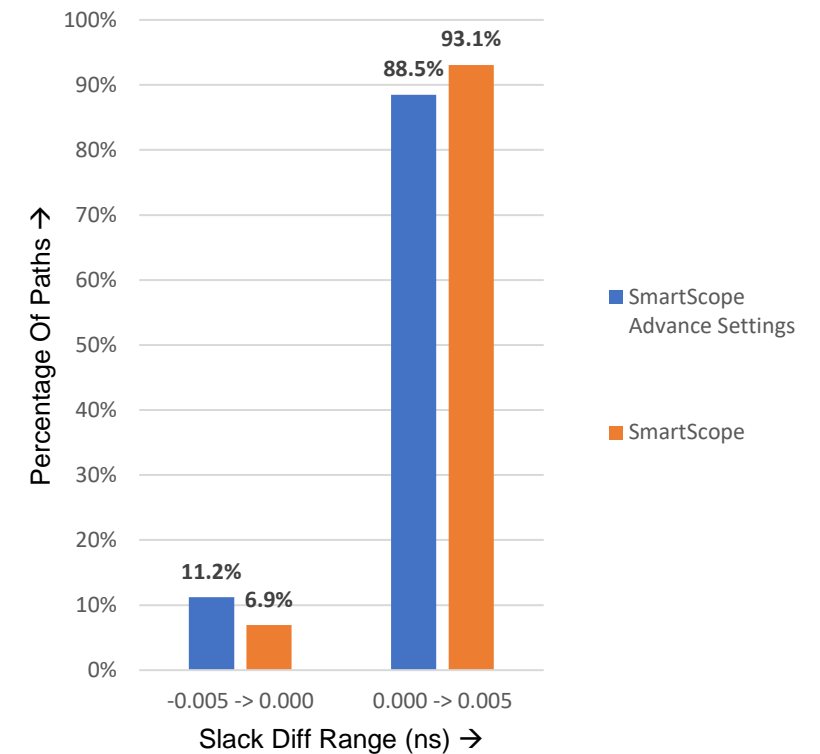
Generate BM	SmartScope	Advance SmartScope	Percentage Change
Runtime (sec)	66	181	36.50% ↑
Memory (MB)	7561.6	7115.7	-6.30% ↓

TOP Design

Design Load	SmartScope	Advance SmartScope	Percentage Change
Runtime (sec)	365	346	-5.2% ↓
Memory (MB)	4550.6	3606.5	-20.7% ↓

Timer Update	SmartScope	Advance SmartScope	Percentage Change
Runtime (sec)	380	218	-42.6% ↓
Memory (MB)	10958.7	7476.5	-31.8% ↓

Timing comparison for SmartScope Flow when using BM-netlist reduction settings



Summary

- ❖ Smartscope flow helps provides realistic constraints to delay calculation engine compared to ETM based HSTA
- ❖ Smartscope BMs retain block interface path information which helps to compute more accurate delays for interface paths
- ❖ Smartscope context timing model provides accurate constraints for reg2reg paths of block which helps in block timing closure
- ❖ Flow shows healthy correlation with top flat with ~99% of the timing paths correlated with the tolerance of 5ps and 2ps for setup and hold respectively
- ❖ This can be used instead of running full chip flat STA as it provide good timing correlation with saving design STA runtime and memory resource

Acknowledgement

- ❖ Tim Helvey, Marvell
- ❖ David Lawson, Marvell
- ❖ Akshay Mankotia, Cadence
- ❖ Keeshik, Cadence

THANK YOU

QnA

