Powerful Hack SDF Flow

for Function and SDC validation

Jianhau Huang
MediaTek

Agenda

- Introduction
- Motivation
- New Methodology
- Conclusion
- Future work

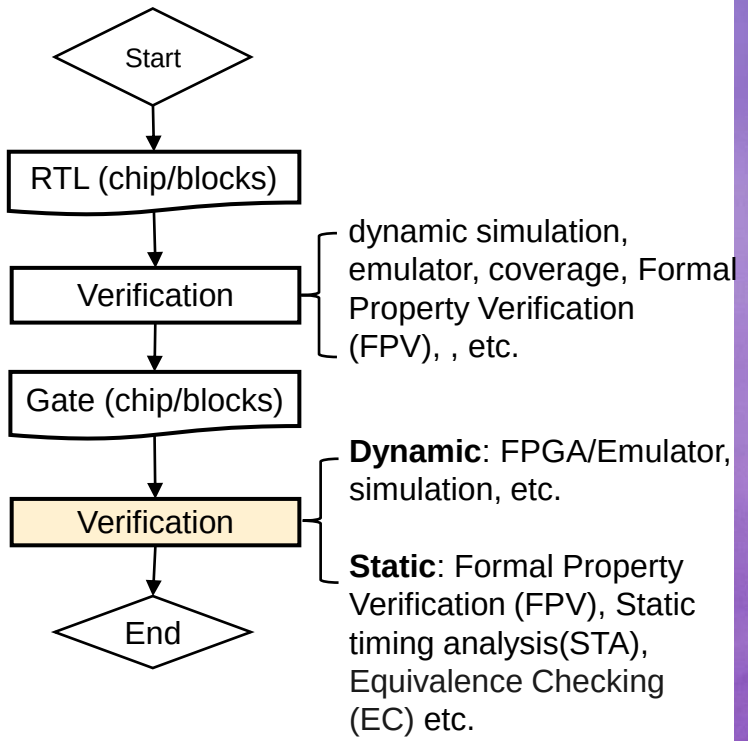
Introduction

Introduction

Motivation for running Gate Level Simulations

- Static verification tool can boost verification performance and reliability but still has its limitation. Clock Domain Crossing(CDC), bugs due to X (unknown) signal propagation or glitch, timing constraint false and multi-cycle paths (MCP) inconsistent, those issue can be catch in gate level simulation w/ delay .

- Reasons for running GLS:
  - **Function**: verify power up & reset operation, DFT structure, power-aware simulation
  - **Timing**: catch false and multi-cycle paths constraint issue, asynchronous designs that are skipped by STA, check if design works at the desired frequency with actual delays
  - **Power**: power pattern generation for signoff
  - **Production**: DFT ATPG pattern generation for tester

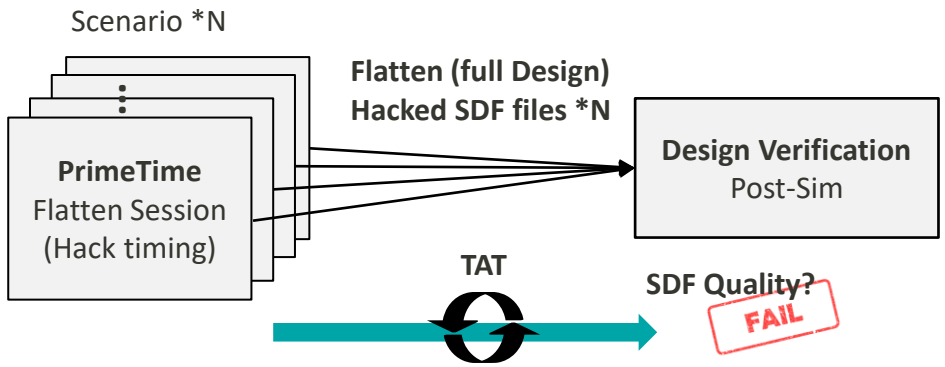


Motivation

Motivation

Running Gate Level Simulations at early stage

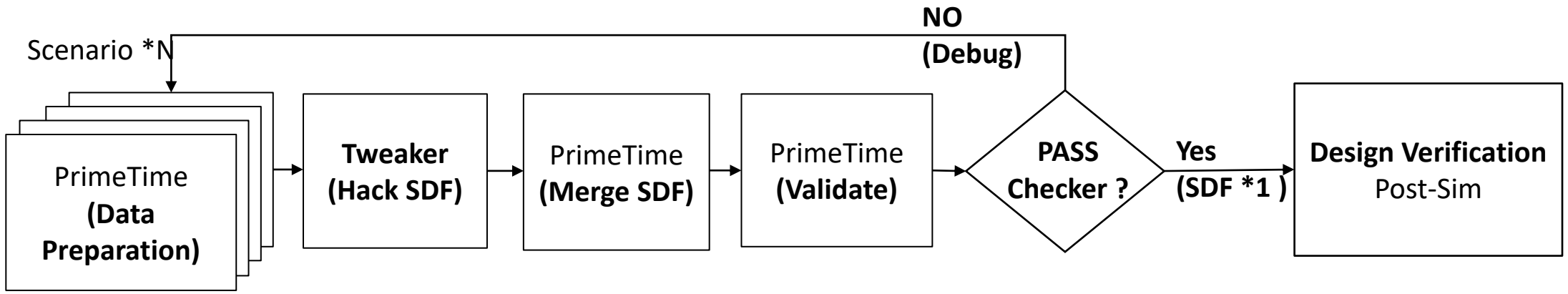
- Need timing clean timing (both setup and hold) SDFs for early stage in clean up flow and finding certain issues before final full speed SDFs ready → **SDF timing hacking**
- Challenges of legacy flow for SDF timing hacking:
    - **Long Turnaround Time**: Design size (>100M) and complexity increase bring growing runtime and highly machine memory usage.
    - **Low Efficiency**: N*Scenario →N* SDF file. Multiple scenario need to generate clean SDFs respectively.
    - **Insufficient Output Quality**: Current flow was not able to clean all pins violations, it only hack the slack value of Endpoint. It's prone to result in hold violation and post-sim fail.



New Methodology

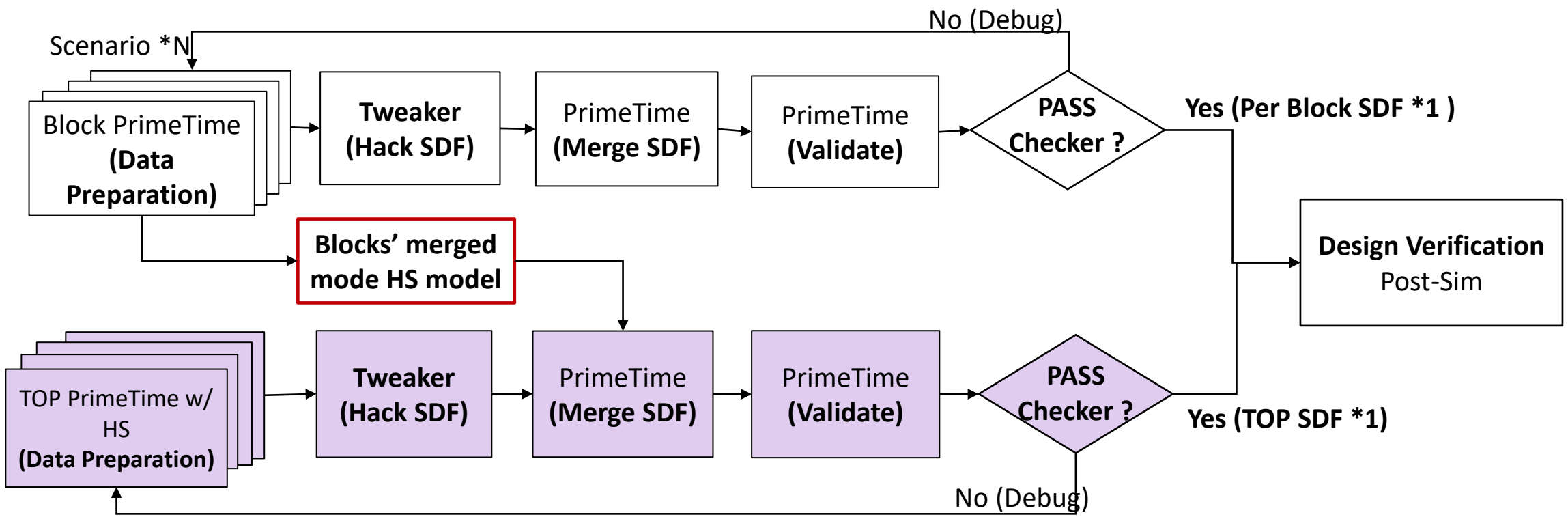
Main Idea

Merge multiple scenarios into one → **Complexity reduction**



Main Idea

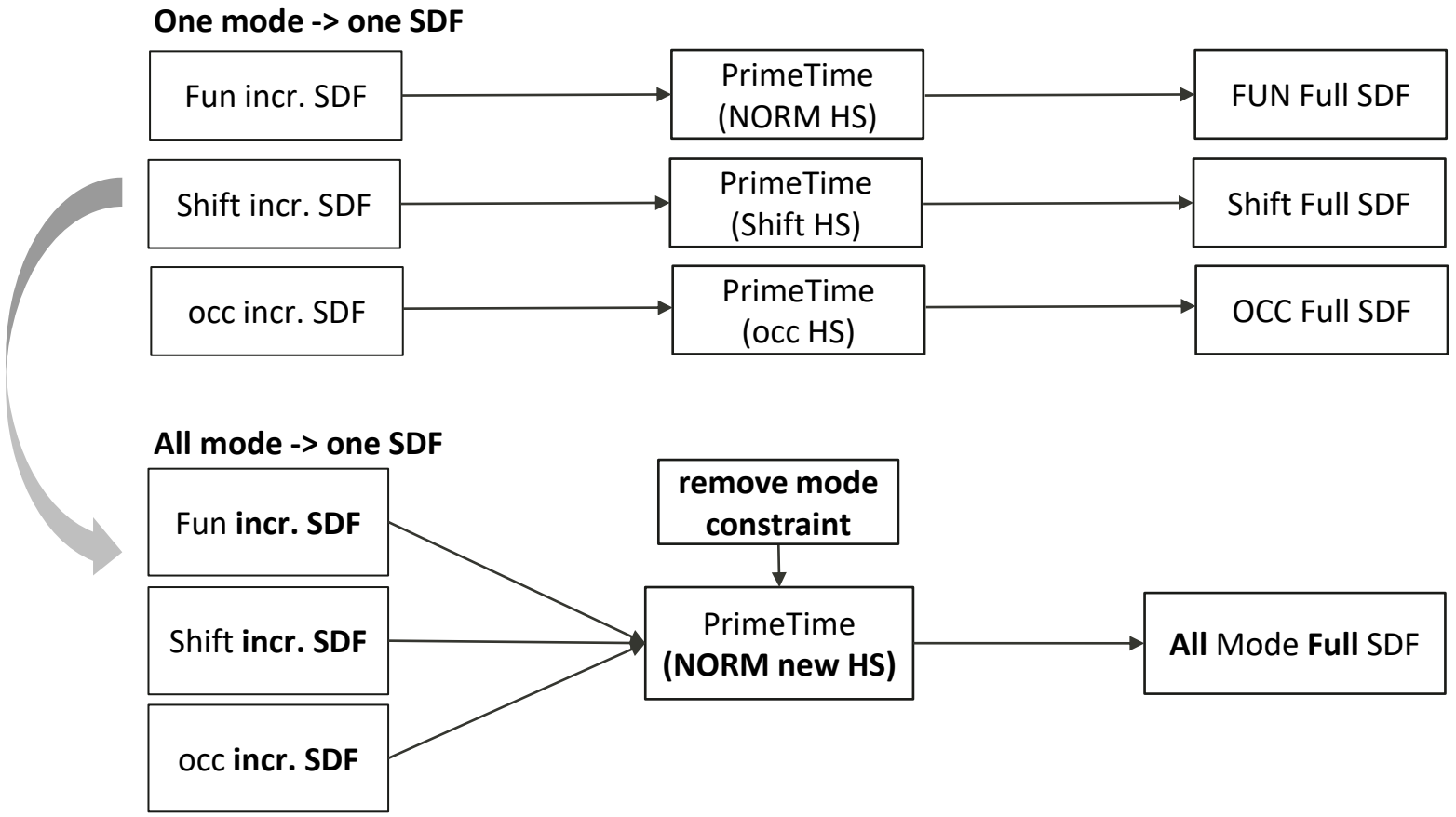
Support Hierarchical → Design scale wear-down



Merge SDF (PrimeTime)

Purpose: VCS can't support multiple-func switch by diff. SDF

- one SDF to support all mode



Solution Novelty & Competitive Differentiation

Competitive Differentiation

Comparison	Existing solution	xSDF flow
Runtime	1 weeks	24~48hrs
~~Memory~~	~~4x~~	~~ba1x~~
Tool Chain	PT*N + VCS*N	PT*N + ECO*1+ VCS*N
Ease Of Use	Very complex for Flow Level Operation	Fully tool chain integration
Quality	Bad	good
Scalability	Difficult	Design & Scenarios
Special technique	-	• Hierarchical mode support • Distribute parallel run • Merge sdf??

100M instance count, >100 clock domain, >1.5GMhz

Benchmark

Benchmark

xSDF (Hold R2R) Result

Case Information			AS-IS				TO-BE				Note
	Process: N3 **Instance**: 140M	**# of Hier**:	**Setup Timing** (ns)		**Hold Timing** (ns)		**Setup Timing** (ns)		**Hold Timing** (ns)		ORG: 1 week
Testing	# of Scenario	Item	R2R	IO	R2R	IO	R2R	IO	R2R	IO	
Case1	3 (Flatten)	WNS	-37.416	0	-38.03	0	-37.302	0	0	0	15.1 h 342 G
		TNS	-88.781	0	-12877.2	0	-68.945	0	0	0	
		NUM	15	0	136020	0	214	0	0	0	
Case2	3 (Flatten) SDC update	WNS	-1.711	-0.577	-0.946	0	-0.418	0	0	0	13.1 h 258 G
		TNS	-421.94	-10.571	-3499.24	0	-7.472	0	0	0	
		NUM	432	21	21571	0	21	0	0	0	
Case3	3 (Hierarchy with HyperScale Model)	WNS	-9.167	-0.363	-1.47	0	-1.125	0	0	0	5.45 h 230 G
		TNS	-62.523	-1.305	-2400.594	0	-8.114	0	0	0	
		NUM	95	4	33811	0	68	0	0	0	

Conclusion

Conclusion

- Generated a SDF which can cover all different working modes without any hold violations.
- Flatten run takes only 24hrs, which is **7X** runtime speedup than existing solution.
- Support Hierarchical flow (HyperScale).
- It has been adopted by ongoing production projects


THANK YOU
Our Technology, Your Innovation™