Achieving Scalability in Transistor-level STA

Synopsys NanoTime for DRAM Design

Hyungjung Seo, Joohan Kim, Taehwan Kim, Younsik Park, and Jung Yun Choi
Memory Business Division, Samsung Electronics Co., Ltd.

Agenda

- Introduction
- Motivation
- Distributed Transistor-level STA
- Experimental Results
- Summary

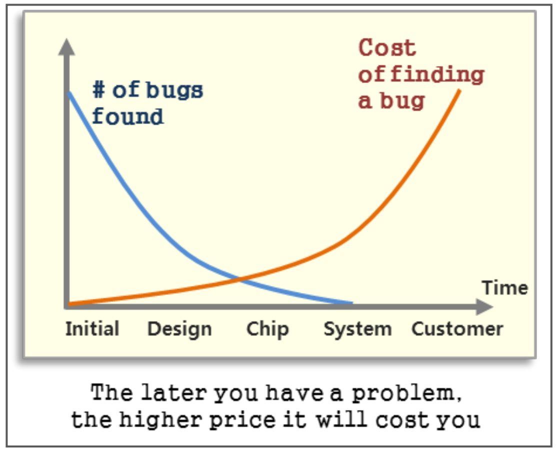
Introduction & Motivation

Signoff Challenges for DRAM Design

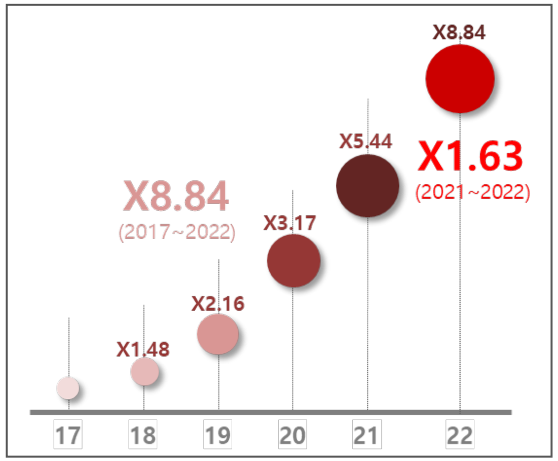
Introduction (1/2)

Verification Challenges in DRAM Design

- Advancements in Design and Verification
  - **Cost and Complexity Surge**: Increase in both design complexity and verification overhead, with a heightened impact of bugs
  - **Memory Product Outlook**: The evolving trajectory of complexity for DRAM within the EDA sector
  - **Design Hurdles**: Challenges stemming from process-design co-optimization, cell-based architecture, and the pursuit of area-efficient design



[Bug Cost]



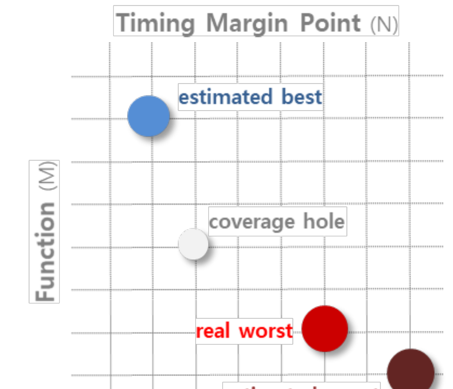
[Memory Complexity Index]

[1] Methodology for Critical Path Analysis of DRAM Design with Transistor-level Static Timing Analysis, SNUG SV 2023

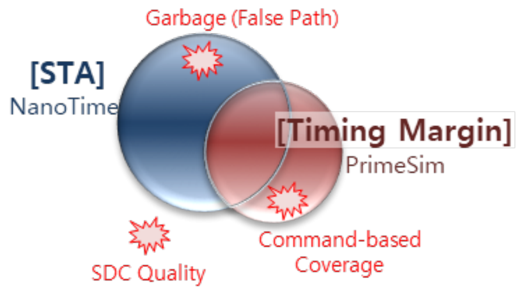
Introduction (2/2)

Verification Challenges in DRAM Design

- Verification Innovations
  - Emphasizing innovative verification techniques tailored to these evolving challenges



※ All Case : N*M
Dynamic Scenario : (N*M).estimated
Static Scenario : (N*M).all



※ Garbage ?
: Non-critical path (Option/Test mode)
→ False Path Reduction
→ SDC Quality dependent
※ 2nd Signoff

[Necessity of Innovations: Paradigm shift like dynamic to static analysis is necessary]

[1] Methodology for Critical Path Analysis of DRAM Design with Transistor-level Static Timing Analysis, SNUG SV 2023

Motivation

Tackling NanoTime's Limitations

- NanoTime Challenges [1]
  - **Speed vs. Coverage**: Extensive coverage but slower compared to optimized SPICE
  - **Efficient Processing Needs**: Limited parallel and distributed processing capabilities in key commands
  - **STA Workflow Hurdles**: Iterative constraint optimization is intensive, especially in DRAM designs

- Divide and Conquer Benefits in STA
  - **Segmentation for Scalability**: Dividing designs by critical paths aligns with full-chip results, improving scalability
  - **Speed Boost with Parallel Processing**: Implementing parallel processing in design segments accelerates STA significantly

[1] Methodology for Critical Path Analysis of DRAM Design with Transistor-level Static Timing Analysis, SNUG SV 2023

Distributed Transistor-level STA

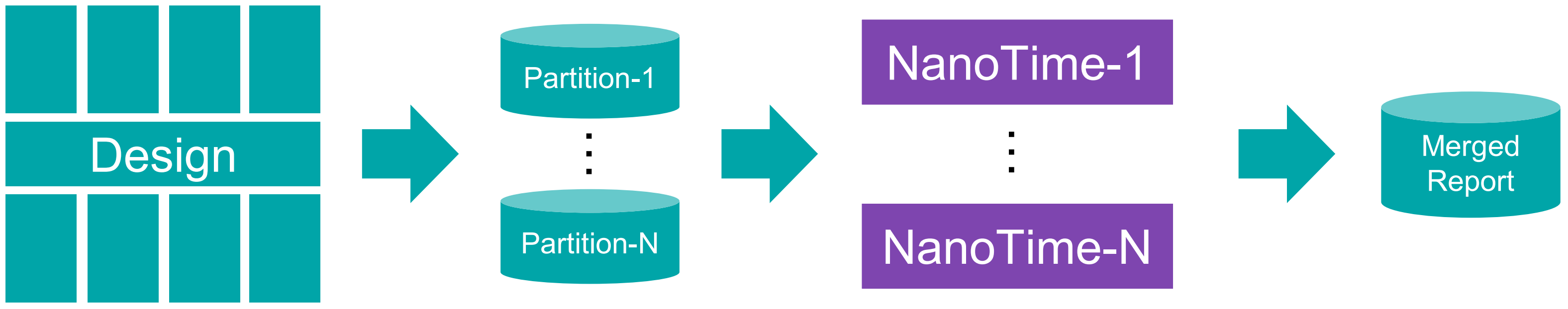
Proposed Method

Distributed Transistor-level STA

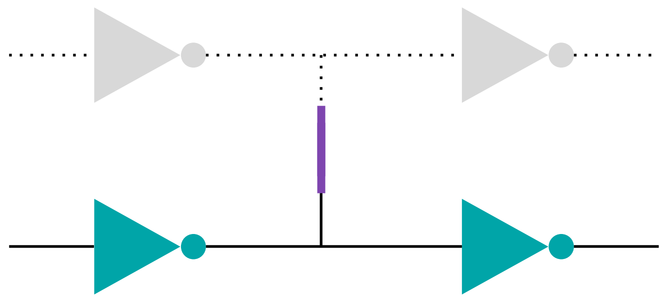
Objectives

- Objectives: Distributed and Accurate NanoTime Runs
  - Implementing distributed NanoTime runs without accuracy loss

- Key Challenges
  - **Precision in Circuit Partitioning**: Success in distributed STA relies heavily on accurate partition
  - **Maintaining Coupling Aggressors**: Crucial to accurately preserve coupling aggressors in partitioned circuits to ensure STA integrity



[Concept of Circuit Partitioning & Distributed STA]



[Coupling Aggressor Node from Non-Critical Paths]

Approaches to Circuit Partitioning (1/2)

User-Defined and Automated Methods

- User-defined Partitioning
  - Effective division of circuits based on critical paths, assuming user capability
  - Risk of accuracy loss due to potential errors in user-driven partitioning
  - Simple and effect way for partitioning

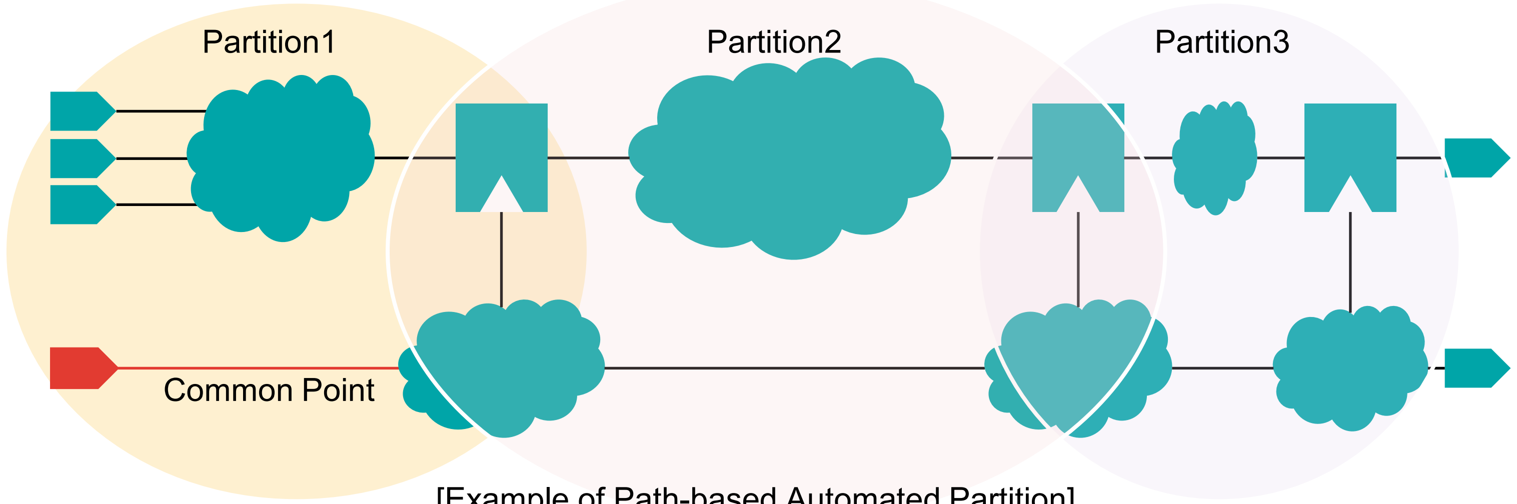
```
suppress_design_except xdesign_0
suppress_design_except xdesign_1
suppress_design_except xdesign_2
suppress_design_except xdesign_3
…
suppress_design_except xdesign_N
```

[Example of User-defined Partition]

Approaches to Circuit Partitioning (2/2)

User-Defined and Automated Methods

- Automated Partitioning Overview
  - Path Tracer technology [2] for automated critical path segmentation and data extraction
  - Algorithmic approach to identify minimum partitioned sets, akin to greedy algorithm solutions in set-cover problems [3]
  - A precise, complex alternative, not the primary focus of this presentation



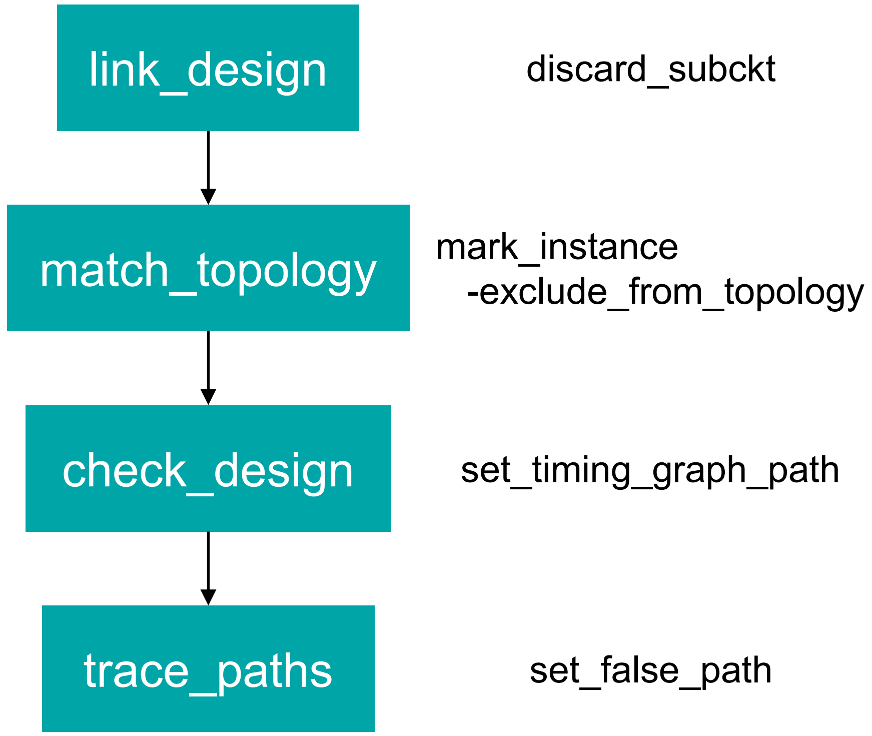
[Example of Path-based Automated Partition]

[2] Path Tracer for Tr.-level STA Setup using Synopsys NanoTime, Engineering track poster in 2023 Design Automation Conference (DAC)
[3] Set cover problem - Wikipedia, https://en.wikipedia.org/wiki/Set_cover_problem

Commands for Tackling Runtime (1/2)

Detailed Commands

- Key Phases in NanoTime Execution
  - match_topology, check_design, and trace_paths significantly influence NanoTime's runtime
- New Commands for Runtime Reduction
  - link_design: link_control_options { discard_subckt } / discard_subckt_contents
  - match_topology: mark_instance -exclude_from_topology
  - check_design: set_timing_graph_path (V-2023.12)
  - trace_paths: set_false_paths / set_find_paths



[RunTime Reduction Commands matched to NT phase]

Commands for Tackling Runtime (2/2)

Detailed Commands

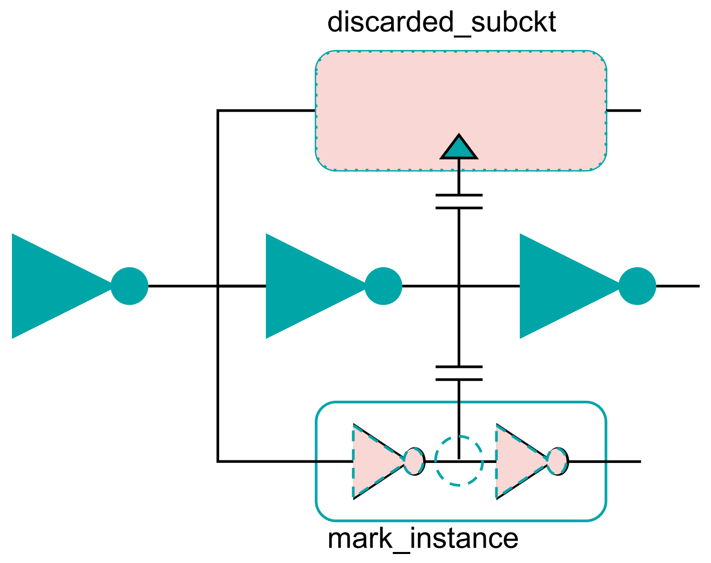
- Addressing Read Parasitics Runtime
  - Reducing the runtime of read_parasitics is crucial. The upcoming "Selective Back Annotation" feature, expected in 2024, is designed to address this
  - As a workaround solution, modifying the Detailed Standard Parasitic Format (DSPF) can help achieve faster runtimes. [1]

[1] Methodology for Critical Path Analysis of DRAM Design with Transistor-level Static Timing Analysis, SNUG SV 2023

NanoTime Command Strategies (1/2)

discard_subckt vs. mark_instance

- discard_subckt Commands
  - Skips specific subcircuits during analysis, leading to significant runtime improvement
  - However, it limits the use of powerful nt_shell functions like get_* commands
    → making high-level partitioning challenging
  - Has risk of loss of coupling aggressors
    - discard_subckt_contents resolves the problem but it generate ground cap



NanoTime Command Strategies (2/2)

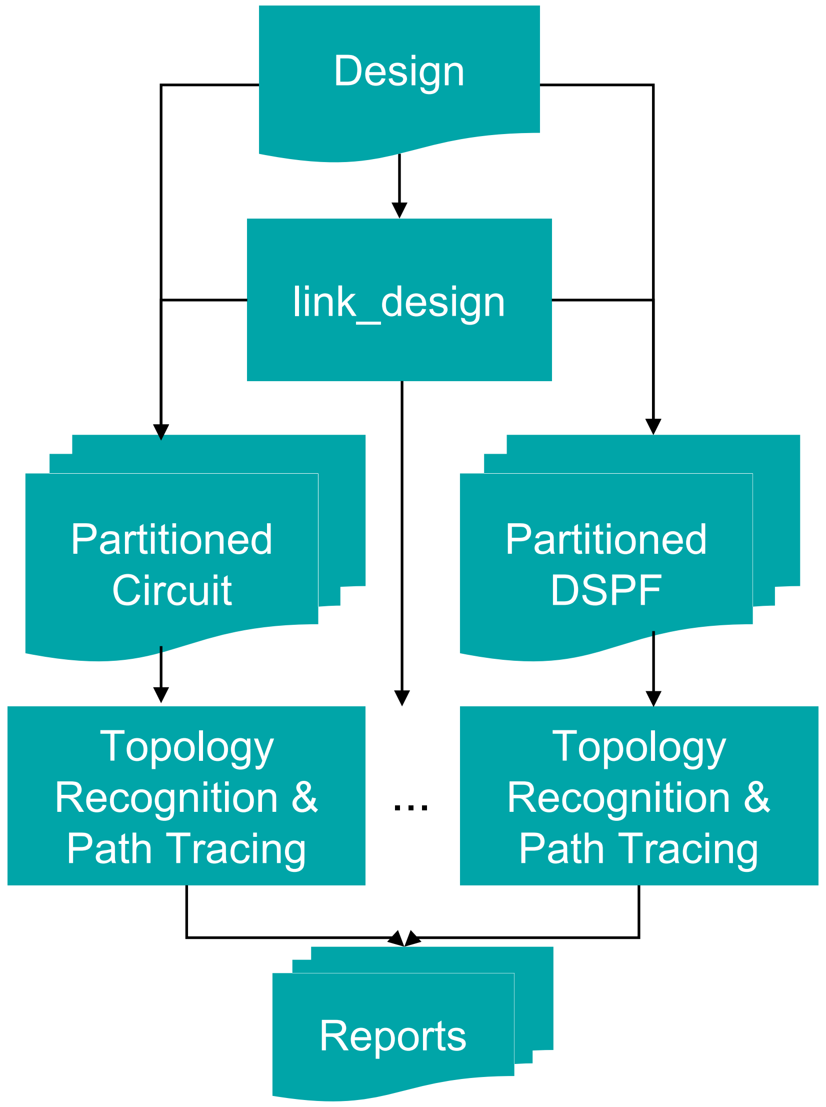
discard_subckt vs. mark_instance

- Proposed Alternative: mark_instance -exclude_from_topology
  - Executed after link_design, this command allows for more manageable circuit deletions while preserving critical path components
  - The runtime for link_design remains reasonable even for 6-7M CMOS counts, typically taking 1-2 hours
  - Speeds up all phases after match_topology and maintains coupling aggressors, ensuring accuracy

Proposed Methodology

Implementing Efficient Flow

- Initial Phase with Link_Design
  - Commence with link_design, applying minimal discard subckt commands for initial circuit reduction
- Partitioning and Circuit Reduction
  - Utilize user-defined or automated partitioning for further reduction using mark_instance -exclude_from_topology
- Selective Parasitic Data Handling
  - Apply a workaround script to extract DSPF only from the reduced circuit, followed by read_parasitics
- Completing the Analysis
  - Proceed with all subsequent analysis phases on the streamlined circuit design


Design
link_design
Partitioned Circuit
Partitioned DSPF
Topology Recognition & Path Tracing
…
Topology Recognition & Path Tracing
Reports

Experimental Results

Single vs. Distributed Run

Experimental Results

DRAM Design Case Study

- Design Overview
  - Complex block for data parity calculation
  - Over 8M CMOS
  - Custom and transistor-level design

Item	# Elements
CMOS elements	8,283,792
resistors	607,504
elements	8,891,296
nodes	4,307,383
subckt	1,573
top-level instances	2

- Approach to Circuit Partitioning
  - User-defined partitioning based on the hierarchical structure of the DRAM design
  - Non-critical paths and remaining blocks from partitioned sections were excluded using the mark_instance -exclude_from_topology command

Experimental Results

Single vs. Distributed

- Distributed run demonstrated over 100x faster performance compared to single run
  - While link_design phase maintained similar runtime, subsequent phases experienced drastic reductions
  - Current results based on experiments with a single design; plans to expand testing to multiple designs, and future challenges and potential issues to be continuously monitored and addressed in partnership with Synopsys

STAGE	SINGLE	DISTRIBUTED (1 Partition)
LINK_DESIGN	00h:24m:54s	00h:23m:08s
MATCH_TOPOLOGY	05h:47m:22s	00h:11m:46s
CHECK_TOPOLOGY	00h:53m:08s	00h:03m:32s
READ_PARASITICS & CHECK_DESIGN	08h:11m:15s	00h:32m:28s
TRACE_PATHS	07d:16h:59m:15s	00h:16m:37s
NT is run on RHEL 7.9 machine with 16 cores of 3.6GHz Intel Xeon and 8 partition can cover target critical paths within 1~2 hours of runtime		

Summary

Distributed Tr.-STA for Efficient Verification

- NanoTime's Essential Role and Challenges
  - Acknowledged the criticality of NanoTime in verification innovation while highlighting the runtime challenges encountered in its application
- Proposing a Distributed Methodology
  - Introduced a distributed methodology to address these challenges, leveraging native NanoTime commands to develop an effective solution
- Significant Runtime Reduction Achieved
  - Demonstrated that the application of this methodology resulted in a substantial reduction in runtime, achieving over 10x efficiency improvements

SAMSUNG snug

THANK YOU

YOUR
INNOVATION
YOUR
COMMUNITY