GPU Accelerated Open Source Timer and Logic Simulation Efforts

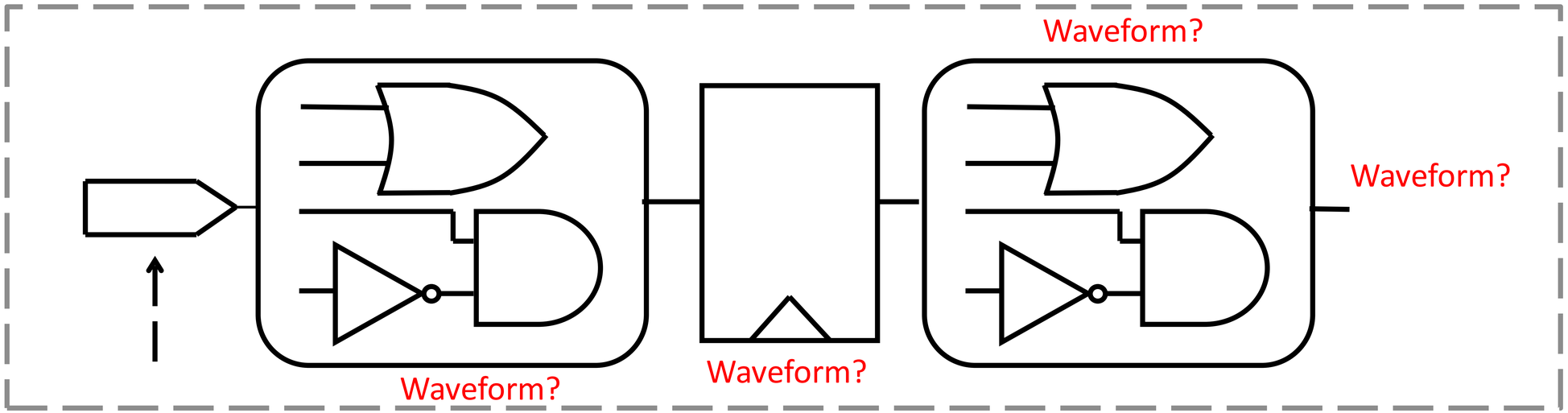
Yanqing Zhang and Yi-Chen Lu, NVIDIA

June 2025

What is Logic Simulation?

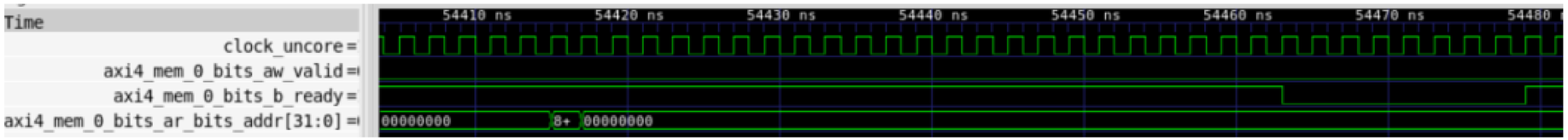
(Perhaps) Needs No Introduction For This Audience

- “I have some hardware representation” + “ I have some form of stimuli” → “I need to know what is the circuit state after being driven by the stimuli”
- Needs accurate characterization through a window of time
- Has many purposes in VLSI design process
- SystemC sim., RTL-level sim., gate-level sim, ALL of which GPUs have contributed acceleration to!



SystemC,
RTL,
Gate-level Verilog

Stimuli



NVIDIA.

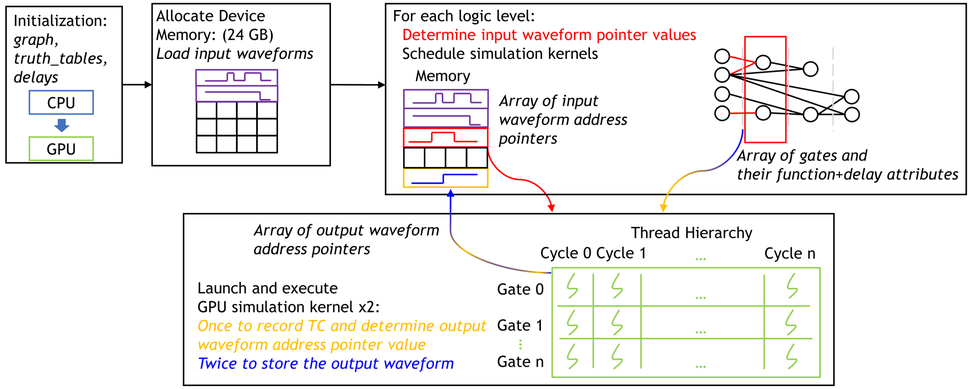
Our Solutions

Multiple levels of simulation abstractions. All Open Source.

Gate Level Re-simulation:

GATSPI: GPU Accelerated Gate-level Simulation for Power Improvement

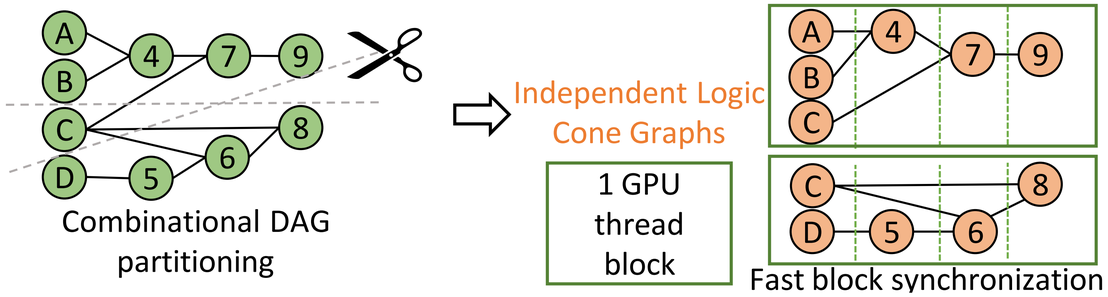
(https://github.com/NVlabs/GL0AM)



Gate Level Simulation:

GL0AM: GPU Logic Simulation Using 0-Delay and Re-simulation Acceleration Method

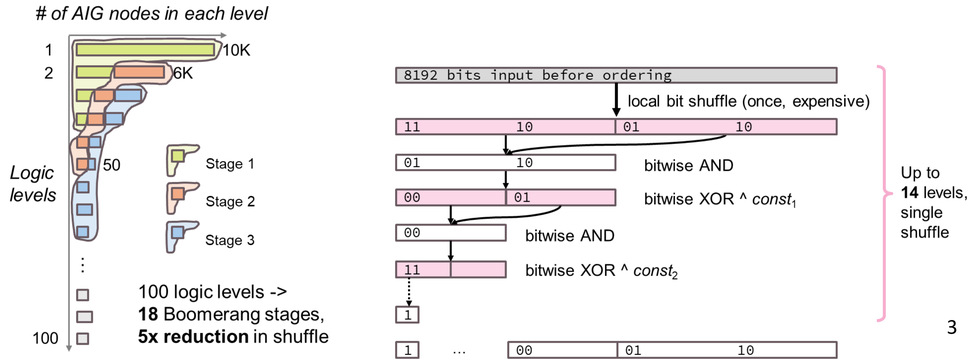
(https://github.com/NVlabs/GL0AM)



RTL Level Simulation:

GEM: GPU-Accelerated Emulator-Inspired RTL Simulation

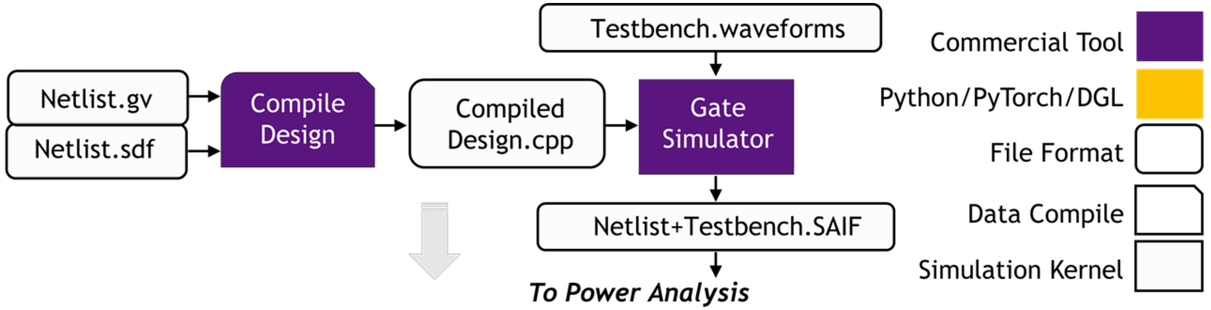
(https://github.com/NVlabs/GEM)

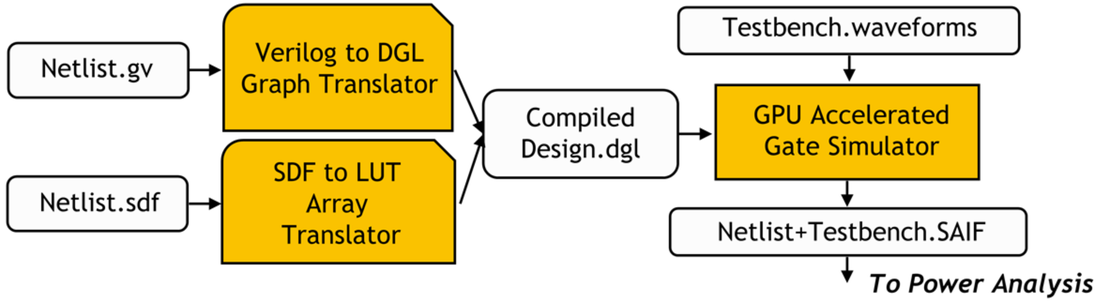


Workflows Mimic Real Tool Usage

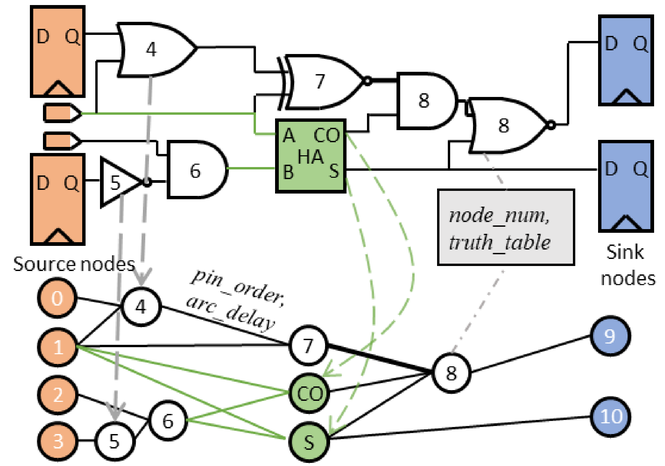
Try To Adhere to the Common Use Case Scenarios, Waveform Dumping IS Supported!

- Compile Verilog to simulation graph.
- Toolflow tries to mimic commercial tool common use-case scenarios
- Non-interactive testbenches…for now…?

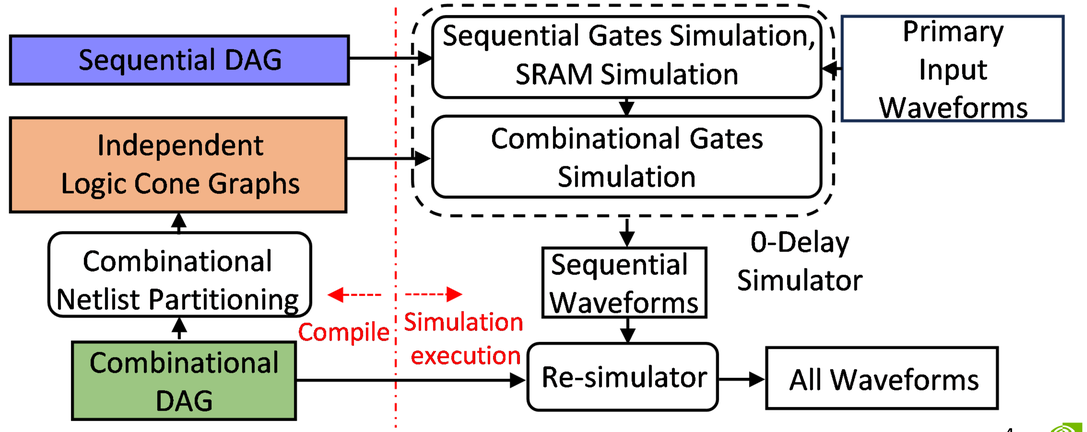




GATSPI



Compile Verilog to simulation graphs



GL0AM

Benchmark Experiments and Results

Activity Factor and Design Size Affect Acceleration Potential

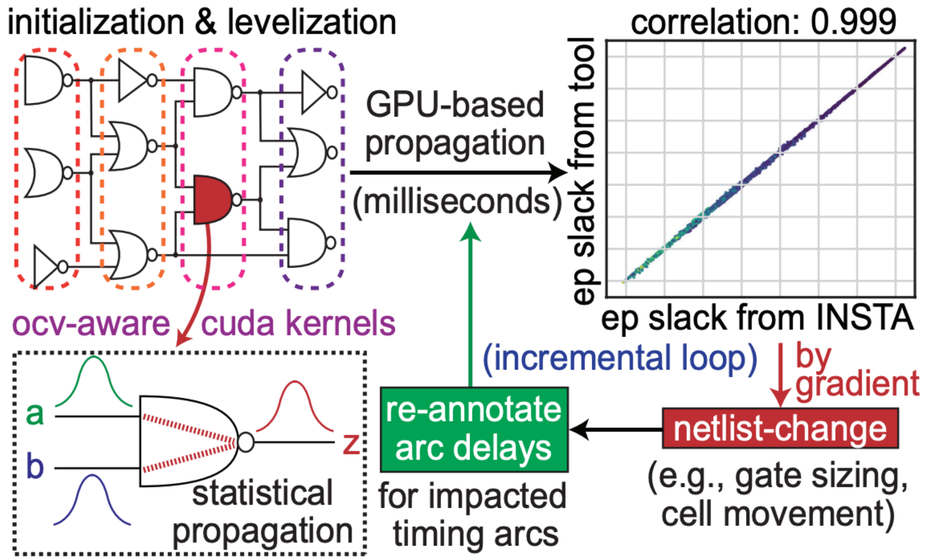
- Benchmarks chosen over wide range of activity factor and design size
- NVDLA (https://github.com/nvdla/hw) , Rocket (https://chipyard.readthedocs.io/en/stable/Generators/Rocket-Chip.html) , OpenPiton (https://github.com/PrincetonUniversity/openpiton)
- Up to 2 million gate, 50k cycle benchmarks
- Open source data set too (includes directed test stimuli)
- Up to 14K X speedups for GATSPI re-simulation in 8 GPU environment

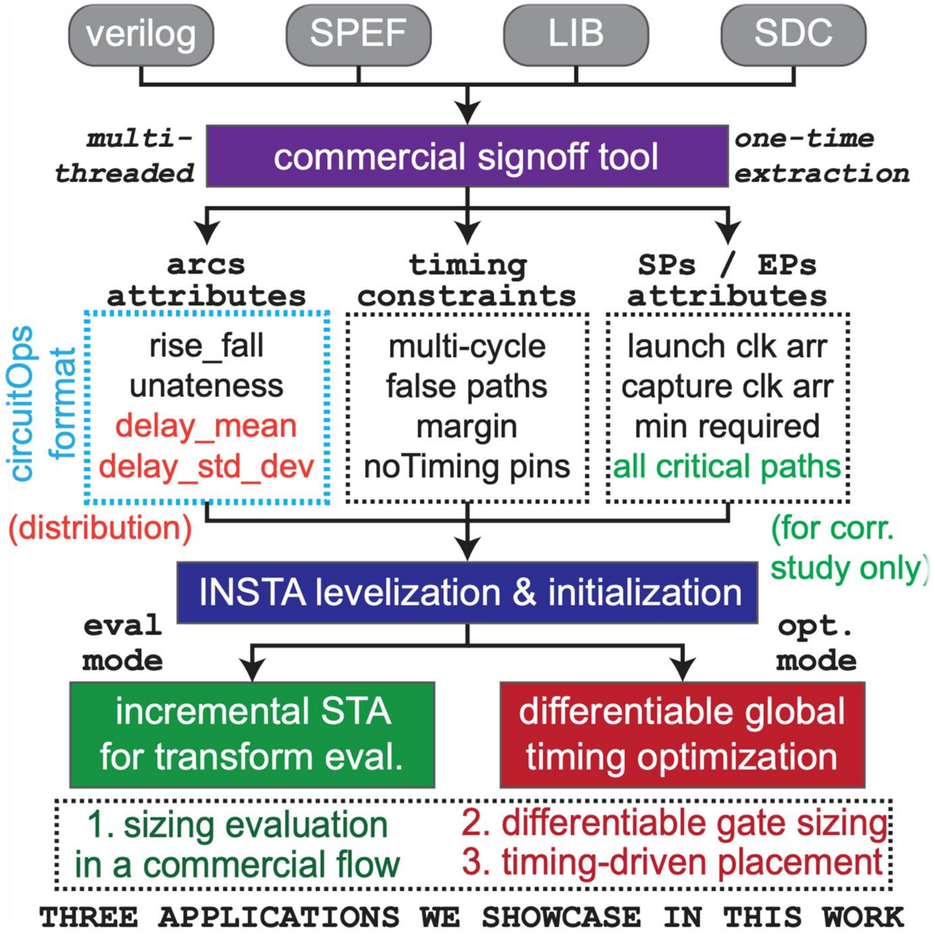
Simulator	Gates	Speedup (X)
GATSPI	1k-2M	28-1198
GL0AM	61k-2M	19-537
GEM	400k-4M	1-40

INSTA: A GPU-Accelerated, Differentiable STA Engine

DAC'25 Best Paper Award Nomination

- Static Timing Analysis (STA) at scale is critical
  - Every netlist transformation require timing verification
- GPU-STA tools have been developed. Why not seen in industry?
  - Accuracy gaps vs. commercial CPU tools
    - Inaccurate delay models (often proprietary)
    - Constraints handling
- Hence, we developed **INSTA**, which is the first to deliver:
  - **Signoff-quality timing evaluation in advanced nodes**
    - 0.9999+ endpoint slack correlation on 15M-pin partition
  - **Differentiable timing optimization at scale**
    - Introduce **"timing gradients"** for PD optimization





Everything Differentiable in INSTA

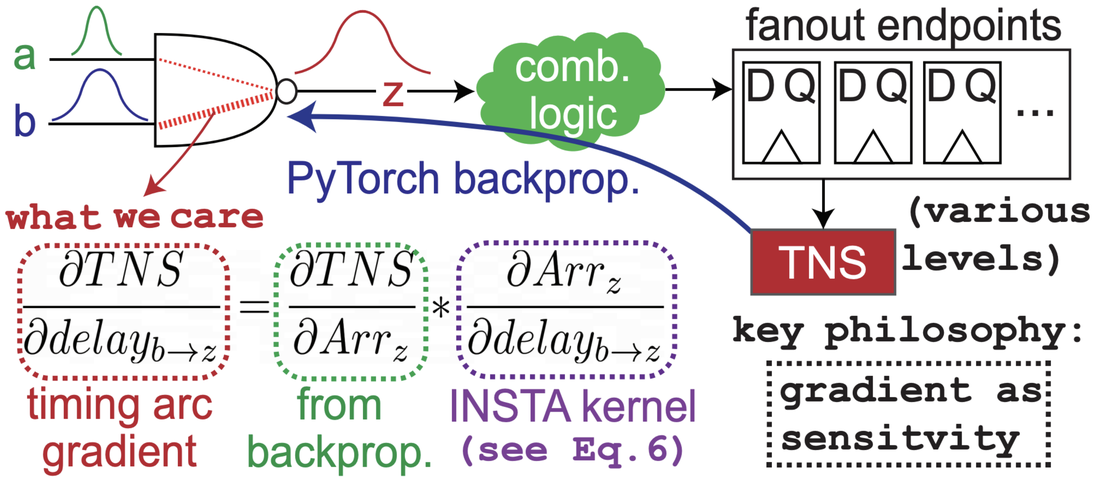
- For the first time, TNS/WNS is differentiable w.r.t. any leaf timing variables
  - Traditional CPU-based STA engine cannot achieve this
  - It is hard to express gradients in explicit forms

- This is enabled by the backpropagation feature in modern ML infrastructure
  - We write custom CUDA kernels and compile with PyTorch

- Use Log-Sum-Exponential (LSE) instead of Max for smooth update

$$\mathrm{LSE}(\{\mathcal{A}_i\}_{i=1}^n) = M + \tau \cdot \log\left(\sum_{i=1}^{n} \exp\left(\frac{\mathcal{A}_i - M}{\tau}\right)\right)$$

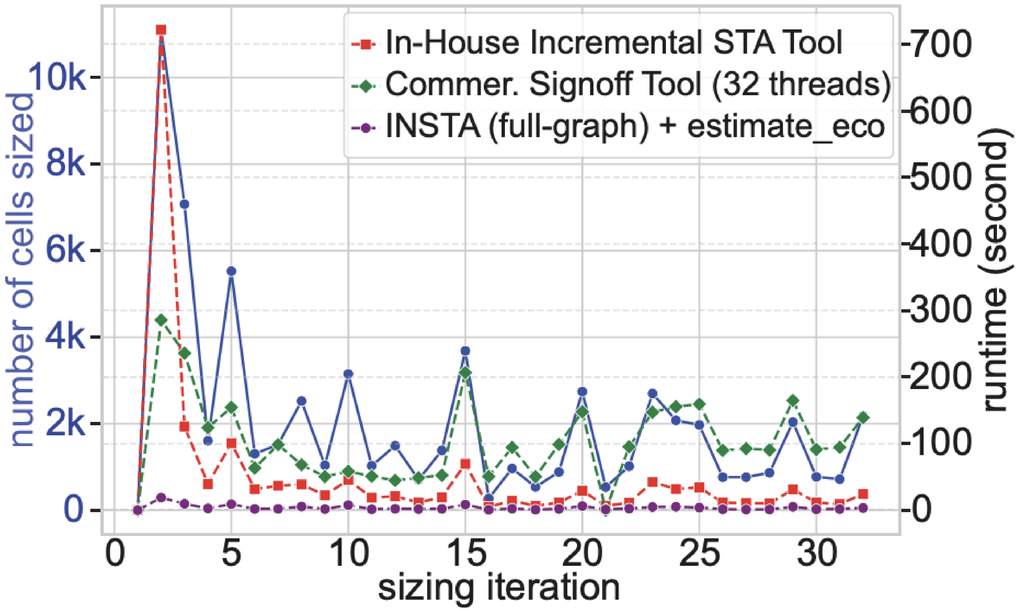
$$\frac{\partial \mathrm{LSE}(\{\mathcal{A}_i\}_{i=1}^n)}{\partial \mathcal{A}_i} = \frac{\exp\left(\frac{\mathcal{A}_i - M}{\tau}\right)}{\sum_{j=1}^{n} \exp\left(\frac{\mathcal{A}_j - M}{\tau}\right)}$$

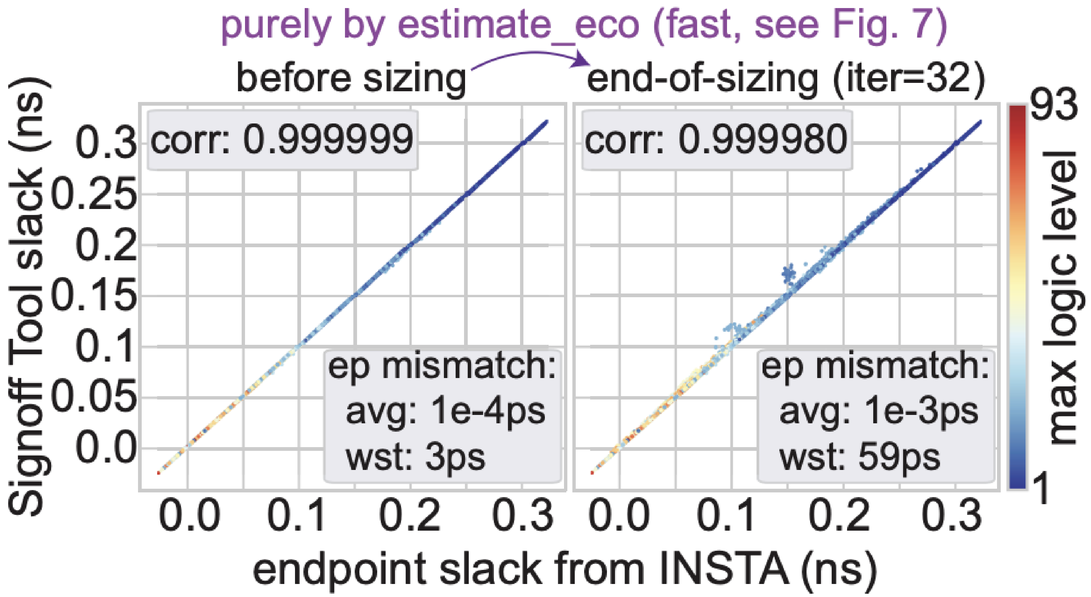


Application-1: INSTA as a Timing Evaluator

In an industrial ECO flow

- We leverage INSTA as incremental timing evaluator
- We re-annotated local delay changes using EstimateECO
  - Note that local delay estimation is extremely fast, and can by highly-threaded on CPU
- Results: >1000x Speed-Up to Well-Established Commercial Tool
- Highly-accurate incremental evaluation results compared with a commercial signoff tool



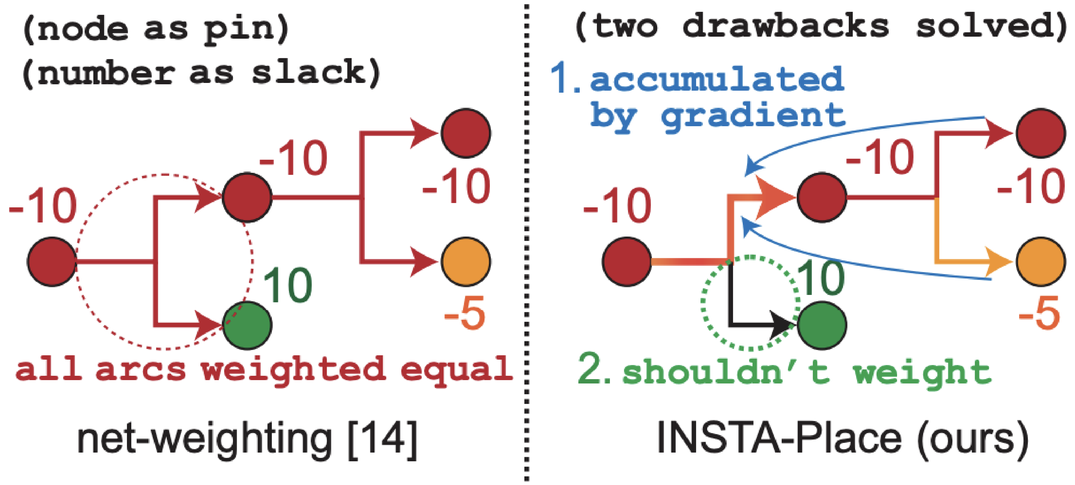


Application-2: INSTA in Timing-Driven Placement

Timing-Driven Placement with Timing Gradients

- Net-weighting-based timing-driven techniques are inherently flawed
  - **Not all arcs are equal on the same net!**
  - Slack does not reflect downstream fanout cone!

- INSTA's propagation kernels elegantly solved the above two drawbacks
  - We perform arc-based weighting
  - We weight by gradients not slack!



$$L_{\text{timing}} = \lambda_{RC} \cdot \sum_{k=1}^{M} \left(|x_{f_k} - x_{t_k}| + |y_{f_k} - y_{t_k}| \right) \cdot g_k$$

$$L = \underbrace{L_{\text{WL}} + \lambda_1 L_{\text{den}}}_{\text{default objective}} + \underbrace{\lambda_2 L_{\text{timing}}}_{\text{INSTA-Place}}, \text{ s.t. } \lambda_2 = \frac{\|\nabla(L_{\text{WL}} + \lambda_1 L_{\text{den}})\|}{\|\nabla L_{\text{timing}}\|}$$

Open-Sourcing INSTA

- INSTA will be available under NVIDIA software license
- INSTA Github: https://github.com/NVlabs/INSTA