ISPD26 Contest: Post-Placement Buffering and Sizing

Organizing Team: UCSD, Fudan University, POSTECH

Co-Chairs

Dr. Yiting Liu, UCSD ABKGroup [yil375@ucsd.edu]

Prof. Zhiang Wang, Fudan University [zhiangwang@fudan.edu.cn]

Table of Contents

==========

1. Contest Introduction

Two of the most important approaches for timing optimization after global placement are **repeater insertion** (buffers and inverters) and **gate sizing**. The ISPD26 Contest challenges contestants to develop a **post-detailed placement buffering and sizing tool** under a set of **real-world constraints.**

Real-World Motivation

This open, academic research contest bridges the gap between research methods and real industry flows. Unlike previous contest settings, this contest tries to emphasize physical and electrical constraints seen in practice. Specifically, this contest includes:

- Fixed macros and I/Os
  - Macros occupy large, immovable regions of the layout, often fragmenting available placement space and constraining repeater insertion and gate sizing.
  - Also, I/Os are fixed in our testcases.
- Power delivery network (PDN) blockages
  - The power delivery network (PDN) uses significant routing resources, exacerbating congestion and making DRC-free routability challenging. This makes fixing timing and ERC violations in congested regions more difficult.
- Placement blockages
  - Representing reserved regions, these blockages take away from the usable areas in the placement canvas, and increase legalization difficulty.
- Fixed routing resource
  - The routing track supply is fixed, so the fixing of timing and ERC violations in congested regions, as estimated by the global router, should fit within the available routing track supply.

Together, these constraints significantly complicate post-placement timing optimization. They limit available whitespace, increase local utilization, and raise the difficulty of ensuring legal and congestion-free cell placement.

Scope of the Challenge

The ISPD26 Contest challenges contestants to develop buffering and sizing tools that operate after detailed placement. Their tools are expected to offer the following benefits:

- Physically-aware timing optimization
  - With detailed placement completed, interconnect delays can be estimated more accurately compared to previous physical design stages (e.g., synthesis, floorplanning and global placement). This enables the tool to make effective buffering and sizing decisions for timing optimization.
- ERC violation fixes
  - Electrical rule check (ERC) constraints, such as maximum slew, maximum capacitance, and maximum fanout, need to be addressed at the post-placement stage to ensure that timing analysis is more accurate and requires fewer iterations to fix ERC violations at the signoff stages.
- Legal and congestion-aware insertion
  - Solutions must satisfy all legality constraints, including placement blockages, PDN-induced pin blockage, fixed macros and fixed I/Os. Inserted repeaters and resized gates must be placed in legal, congestion-friendly areas so that global routing finishes within time limits.

This contest is built on top of the OpenROAD [1] infrastructure and provides a controlled, open-source environment for benchmarking and evaluation. To encourage scalability and innovation, contestants may (but are not required to) leverage **GPU-accelerated** solutions or **modern ML** infrastructure such as PyTorch [2], which have shown strong potential in solving large scale optimization problems in physical design [3][4][5][6][7][8]. The contest's evaluation framework, along with sponsor-provided compute resources that we plan to make available to contest teams who pass certain contest stages, will include GPUs. Nonetheless, the primary objective of this contest remains achieving **better PPA results under real-world constraints.**

2. Contest Objectives

- The contest focuses on post-global placement optimization for better PPA results via:
  - **Repeater (buffers/inverters) insertion**
    - Insert repeaters (buffers/inverters) to fix ERC violations, and to improve timing and power.
  - **Gate sizing**
    - Select appropriate cell variants from the given cell library (differing in drive strength and VT flavor) to balance timing and power.
- Constraints
  - Macros and I/Os are fixed and can not be moved.
  - **PDN and placement blockages must be honored.**
  - Routing track supply is fixed.
  - No netlist restructuring beyond buffer/inverter insertion and sizing.
  - The output netlist must remain functionally equivalent (equivalence will be checked).
  - Setup (late-mode) timing check, single mode with ideal clock.
  - Displacement: logic gates except for repeaters must not be moved too far from their initial legal position. Movement beyond the specified threshold will be penalized.
- The workflow before and after integrating the contestant's tool is shown in Figure 1.
  - ISPD26 Contest workflow steps
    - **Inputs to contestants**:
      - .v file: Debuffered gate-level netlist that is input to OpenROAD global placement
        - Note: The netlist is debuffered with the exception of the I/O buffering added through the ORFS flow. We also run global_placement with timing-driven placement disabled so that no buffers or inverters are added.
      - .def file: post-detailed placement, legalized placement
        - Note: After global placement we run OpenROAD *detailed_placement* and *improve_placement* commands to generate this .def file.
      - .sdc file: Design constraints include clock period, I/O delay etc.
    - **Contestant optimization**: Contestants' tools perform buffering and gate sizing. Their tools must output the modified .v, .def and changelist files (for more details please see output formats).

- **Evaluation flow**: The evaluation flow requires that the submitted placement is legalized and that the netlist is logically equivalent to the input netlist:
  - Legality checks (checkPlacement) and equivalence checks
    - Also: stitch the component placements into the given floorplan .def (including PDN)
  - **If legalized and equivalent**: Directly perform global routing → OpenSTA timing analysis → **report metrics**
  - **If not legalized or not equivalent**:
    - Mark as evaluation failure

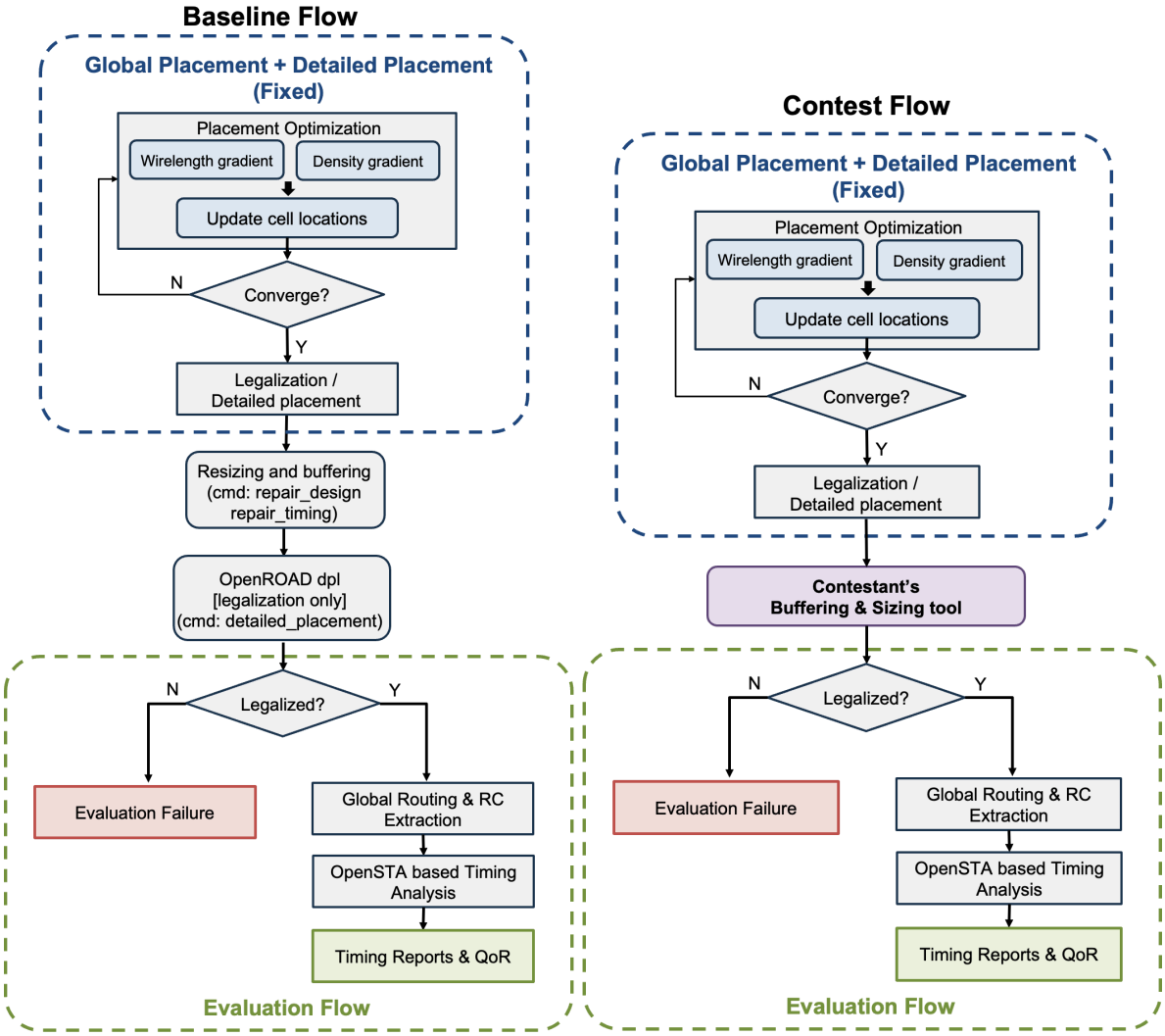

Figure 1. **Left:** Baseline OpenROAD flow. **Right:** ISPD26 Contest flow that integrates the contestant's buffering/sizing tool. Fixed macros, PDN, blockages, and I/O constraints are respected in both flows. The outputs of global and detailed placement are fixed. The evaluation pipeline includes placement legality and equivalence check, global routing, and OpenSTA-based timing and power analysis to report final QoR metrics. **Only the contest integration block (purple) is editable**.

3. Problem Formulation

3.1 Input Formats

Each benchmark will include standard design files in the **ASAP7** technology node. The provided files include:

- **.lef file**: Specifies the physical definitions of the different cells, metal layers, and layout of the design.
- **.lib file**: Library file that consists of the logic description, look-up-tables for delay, slew, and power computation with the area of each cell.
- **.sdc file**: Specifies input port delay and transition, output port delay and capacitance, and specified clock period.
- **.v files**: Debuffered gate-level netlist that is input to OpenROAD global placement.
    - Note: The netlist is debuffered with the exception of the IO buffering added through the ORFS flow.
- **.def file**: post-detailed placement, legalized placement with fixed IOs, macros, PDNs, and placement and routing blockages.

3.2 Output Formats

Your tool must return a legalized placement

- **Modified Verilog Netlist (.v)**: Includes only repeater insertions and cell sizing modifications
- **Updated DEF File (.def)**: Reflects your placement solution (must be legalized)
- **.changelist file**: Indicate what are the sizing and buffering transforms being done to the original netlist
    - Sizing:
        - **`size_cell <cellName> <libCellName> <x, y>`**
        - <cellName> must exist in the provided gate-level verilog netlist.
        - <libCellName> must exist in the provided library (.lib and .lef).
        - The new <libCellName> must have the same function as the original.
        - <x, y> is the location of the resized instance
    - Buffering:
        - **`insert_repeater <library repeater name> <new repeater cell name> <new buffered net name> <driver pin name> <list of load pin names>  <x,y>`**
        - <library repeater name> must exist in the provided library
        - <new repeater cell name> <new buffered net name> must match in the updated .def file
        - <driver pin name> is the name of the driver pin that drives the new repeater
        - <list of load pin names> is the list of load pin names that are driven by the new repeater
        - <x, y> is the location of the new repeater

Note: I/Os can not move. Net names must be preserved. New nets and instances must be named according to the existing OpenROAD convention; see Resizer::makeUniqueNetName [link] and Resizer::makeUniqueInstName [link] [documentation / examples will also be prepared for use in the contest]. All modifications must preserve netlist functionality (contestants' solutions must pass LEC) and satisfy all hard (non-overlap, legal sites, PDNs, blockages, etc.) design constraints.

3.3 Evaluation: Metrics, Platforms

3.3.1 Metrics*

The scoring metric is subject to change, based on the needs of the contest and the quality of the solutions received.
For example, we may include the normalized area in the objective. Also, timing violations may be squared (instead of just using TNS) to emphasize setup timing closure and to discourage excessive WNS violations.
Contestants will be notified before changes are made to the scoring metrics.

Contestants' solutions will be evaluated based on global routing quality, runtime efficiency, and legality, using OpenROAD's timing and power analysis infrastructure. Evaluation metrics include

- Post-global route PPA: Compare normalized improvement vs. baseline on setup TNS, dynamic power (internal power + switching power), and leakage power. All results are reported by OpenSTA.
  - Normalized TNS improvement
    - $\Delta_{TNS} = \frac{TNS - TNS_{baseline}}{|TNS_{baseline} + \epsilon|}$
  - Normalized dynamic power improvement
    - $\Delta_{dpower} = \frac{DPower_{baseline} - DPower}{DPower_{baseline}}$
  - Normalized leakage power improvement
    - $\Delta_{lpower} = \frac{LPower_{baseline} - LPower}{LPower_{baseline}}$
  - Combine with weight w_{tns}, w_{dpower} and w_{lpower}
    - $S_{PPA} = w_{tns} * \Delta_{TNS} + w_{dpower} * \Delta_{dpower} + w_{lpower} * \Delta_{lpower}$

- ERC violation penalty
  - Sum of ERC violation values after global routing: slew violation, capacitance violation, and fanout violation
    - $P_{ERC} = w_{slew} * \frac{SlewViolation - SlewViolation_{baseline}}{SlewViolation_{baseline} + \epsilon}$

$$+ w_{cap} * \frac{CapViolation - CapViolation_{baseline}}{CapViolation_{baseline} + \epsilon}$$

$$+ w_{fanout} * \frac{FanoutViolation - FanoutViolation_{baseline}}{FanoutViolation_{baseline} + \epsilon}$$

- Runtime
  - The runtime of the buffering & sizing tool itself
    - $R_{tool} = \frac{t_{tool} - t_{baselineTool}}{t_{baselineTool}}$
      - Where t_{tool} is the runtime of contestants' developed tools, and $t_{baselineTool}$ is the runtime of OR Resizer
  - Total flow runtime (buffering & sizing tool + fixed evaluation flow)
    - $R_{flow} = \frac{t_{flow} - t_{baselineFlow}}{t_{baselineFlow}}$
      - Where t_{flow} is the contestants' total flow runtime, and $t_{baselineFlow}$ is the default (Figure 1, left) total flow runtime
  - Combine with weight $w_{toolRuntime}$ and $w_{flowRuntime}$
    - $R = w_{toolRuntime} R_{tool} + w_{flowRuntime} R_{flow}$
  - Note: There will be a maximum runtime limit subject to each testcase. The final runtime limits will be determined after the alpha submission.

- Average displacement penalty
  - Average Manhattan displacement per movable cell from its original position
    - $P_{dis} = w_{dis} \frac{D_{avg} - D_{baselineAvg}}{D_{baselineAvg} + \epsilon}$
- Global routing overflow penalty
  - **Maximum** O_{max} global routing overflow values after routing (thresholds $O_{max\,Thr}$)
    - $P_{max} = \frac{O_{max} - O_{maxThr}}{O_{maxThr} + \epsilon}$
  - **Total** O_{total} global routing overflow values after routing (thresholds $O_{total\,Thr}$)
    - $P_{total} = \frac{O_{total} - O_{totalThr}}{O_{totalThr} + \epsilon}$
  - Combine with weight $w_{maxOverflow}$ and $w_{totalOverflow}$
    - $P_{overflow} = w_{maxOverflow} P_{max} + w_{totalOverflow} P_{total}$
- Hard constraints
  - Placement legality
    - Final placement must pass legality checks performed by checkPlacement: no cell overlaps, correct on-site placement and orientation, fixed I/O pins, and compliance with placement blockages and PDN constraints.

- Logical equivalence
  - The netlist must be logically equivalent; a logic equivalence check (LEC) will be performed.
- No perturbation of the given floorplan is allowed
  - Blockages, PDN, Macro and I/Os must be the same as the input.
  - The component placement must be stitched into the provided floorplan .def (including PDN)

$$C_{HC} = \begin{cases} 1, \ if \ follow \ all \ hard \ constraints \\ 0, \ otherwise \end{cases}$$

- Final Score
  - $S_{final} = C_{HC} * (S_{PPA} - P_{ERC} - R - P_{dis} - P_{overflow})$

Note: The **baseline** results are generated using the Baseline OpenROAD Flow (baseline results on public testcases are shown here). Global routing iteration count will be fixed (capped). Netlist equivalence will be checked. Only submissions that follow the hard constraints will be considered for scoring. Note that we may update the scoring function till beta submission.

3.3.2 Platforms

- We will provide an evaluation environment for you via the Purdue Anvil supercomputer, with a specified number of CPU and GPU hours in your allocation, after successful Alpha submission. We expect participants to otherwise use their own resources for main development, and to use this provided resource for tuning and practice evaluation.
  - Purdue Anvil provides 128c/256t AMD EPYC-7763 CPUs with NVIDIA A100 GPUs with CUDA 12.6 and Singularity.
- The final evaluation environment will have stricter limits as defined below and will be run using the Singularity environment:
  - 1 NVIDIA A100 GPUs (40 GB)
  - 8 CPU cores (16 threads)
  - 64 GB memory
  - 200 GB disk
- Availability
  - Contestants **with university email addresses** can access these resources.
  - Teams that submit a working alpha can obtain resources, and those that submit a working beta can obtain more resources.
  - **Caveats:** (1) The planned resource availability is subject to change (e.g., with changes to U.S. government policy), and (2) the amount of resource per team may depend on the number of teams registering.
  - **We will provide a README on how to use this resource.**
- The runtime limits will be determined after the alpha submission.

We will provide a Docker script that generates the Docker image, as well as a script to convert the Docker image to a Singularity image. Along with that, we provide Docker and Singularity images. We will use this Singularity image for evaluation. **Contestants must build on top of this Docker or Singularity environment.** For more details see the submission guideline.

4. Benchmarks

We will release a diverse set of 8 public benchmarks and 4 hidden benchmarks in the ASAP7 technology node with 7.5T multi-VT cell library. The benchmarks will vary in size from 15K - 1000K instances, including

- Macro-free designs **(4 public benchmarks)**
- Macro-containing designs **(4 public benchmarks + 4 hidden benchmarks)**

All public benchmarks are as follows

(Name, TCP, Util)	**Size**	**Macros (y,n)**	**PDN (y,n)**	**Blockages (y,n)**
AES, 250, 0.4	15K	N	Y	N
JPEG, 350, 0.7	50K	N	Y	N
AR37, 900, 0.3	0.1M	Y	Y	Y
AES_v2, 200, 0.4	15K	N	Y	**Y**
JPEG_v2, 450, 0.65	50K	N	Y	**Y**
AR37_v2, 950, 0.45	0.1M	Y	Y	**Y**
BSG_CHIP, 1200, 0.3	0.9M	Y	Y	**Y**
BSG_CHIP_v2, 1300, 0.5	1.4M	Y	Y	**Y**

5. Calibration/baseline Results

5.1 Post-Global Route Results

The following tables show the post-global routing results reported by the contest evaluation flow for three placement results

- **PreOpt**: placement results (.def_dp_comm) (.v1) **without** buffering&sizing (this is the output of the fixed global and detailed placement flow shown in Figure 1 and serves as input to the buffering/sizing tool).
- **Comm**: Placement after buffering&sizing (.v2_comm) using a **commercial tool** (for calibration purposes only).
- **OR (Baseline)**: Placement after buffering&sizing (.v2_or) using **OpenROAD Resizer (repair_design + repair_timing)**.

- Following are details of the different types of netlists that we will use in the contest:
  - .v1: debuffered netlist generated by an unnamed commercial or open-source synthesis tool
  - .v2_comm: buffered netlist generated by an unnamed commercial tool
    - This netlist is made available only for contestants' calibration purposes
  - .v2_or: buffered netlist generated by OpenROAD Resizer (repair_design + repair_timing)
    - Command: *repair_design*
      repair_timing -setup -skip_gate_cloning -skip_pin_swap

[We will post the calibration results soon]

5.2 Scores

The following tables show the weights of each score component and final scores of the three placement results on each testcases

- For each team, the total score is the sum of the scores for each testcase. We may use different weights for the hidden and open benchmarks. However we will release and fix all weights after the alpha submission deadline.
- Note: The weights of each score component will be further adjusted as more testcases are released.
- The following scores do not include the averaged displacement penalty and tool runtime. The final contestant scores will be based on all metrics listed in Section 3.3.1 [link].

Weights of each score component (preliminary, subject to change as noted)								
w_tns	w_dpower	w_lpower	w_slew	w_cap	w_fanout	w_flowRuntime	w_maxOverflow	w_totalOverflow
80	40	40	0.001	10	1	1	1	1

[We will post PreOpt, Comm and OR (baseline) scores for all public benchmarks soon]

6. Submission Guidelines

Environment setup. All participants must build on the Docker or Singularity environment we provide. If additional packages are needed, include a **setup.sh** script that completes in under two hours to set up the environment. During the contest period, we will update the Docker/Singularity environment with any major missing packages so participants do not need to install them.

Submission files. Each participant must submit a **ZIP file** containing setup.sh, run.sh and any additional data or code not already retrievable in setup.sh.

The run.sh must invoke your developed tool to read the input .lib, .lef, .v, .def and .sdc files and produce three outputs – modified .def, modified .v, and .changelist (see output formats) – in the directory specified by <output_dir>. The execution format should be:

<developed_tool> <design_name> <tech_dir> <design_dir> <output_dir>

The input files are as follows:
1. Technology LEF file: <tech_dir>/lef/asap7_tech_1x_201209.lef
2. Standard-cell LEF files: <tech_dir>/lef/asap7sc7p5t_28_*_1x_220121a.lef
3. Macro LEF files: <tech_dir>/lef/sram_asap7_*.lef
4. LIB files: <tech_dir>/lib/*.lib
5. Design files:
    DEF: <design_dir>/contest.def
    Verilog: <design_dir>/contest.v
    SDC: <design_dir>/contest.sdc

The output files should follow the following format:

<output_dir>/<design_name>.def, <output_dir>/<design_name>.v and <output_dir>/<design_name>.changelist

All three files must be present in the <output_dir> and consistent with each other; the changes listed in the changelist must be reflected in the Verilog and DEF files.

7. Timeline

- Registration Open: Oct 1, 2025
- Release the first set of testcases, evaluation scripts and Dockerfile: Oct. 1, 2025
- Release the send set of testcases with blockages: Nov. 6, 2025
- Registration Close: Nov 30, 2025
- Release all public testcases: Dec. 3
- Alpha Submission Deadline: Jan 12, 2026
- Beta Submission Deadline: Feb, 2, 2026
- Final Submission Deadline: Mar, 7, 2026 (Anywhere on earth, and it is a hard deadline)
- Results Announcement: March 18, 2026

8. Contact

Email: ispd26contest@gmail.com

9. Registration

- ~~Please fill in this online registration form~~
- Registration window: Oct.1, 2025 - Nov 30, 2025

10. Contest Prizes

- First, second and third place winning teams will receive prizes consisting of cash and/or NVIDIA GPUs, with a total value of prizes at least USD $5000. Thanks to NVIDIA for their sponsorship of the ISPD26 contest prizes!

11. Organizers

- Andrew B. Kahng, Sayak Kundu, Yiting Liu, and Davit Markarian from UCSD
- Zhiang Wang from Fudan University
- Seonghyeon Park from POSTECH

12. Sponsors

- Purdue University and the NSF Chipshub: compute resources for teams and submission evaluation
- NVIDIA: prizes for winning teams

References

[1] OpenROAD-Flow-Scripts. https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts

[2] I. Sagar, K. B. Prakash, and G. R. Kanagachidambaresan, “PyTorch.” *Programming with TensorFlow: solution for edge computing applications* (2021), pp. 87-104.

[3] A. B. Kahng, Y. Liu, and Z. Wang, “Recursive Learning-Based Virtual Buffering for Analytical Global Placement”, *MLCAD*, 2025.

[4] Y-C. Lu, Z. Guo, K. Kunal, R. Liang, and H. Ren, “INSTA: An Ultra-Fast, Differentiable, Statistical Static Timing Analysis Engine for Industrial Physical Design Applications”, Proc. ICCAD, 2025.

[5] Y. Du, Z. Guo, Y. Lin, R. Wang and R. Huang, “Fusion of Global Placement and Gate Sizing with Differentiable Optimization”, *Proc. ICCAD*, 2024.

[6] H. Wu, Z. Huang, X. Li and W. Zhu, “AiTO: Simultaneous gate sizing and buffer insertion for timing optimization with GNNs and RL”, *Integration* 98 (2024), pp. 102211.

[7] R. Liang, S. Nath, A. Rajaram, J. Hu, and H. Ren, “BufFormer: A Generative ML Framework for Scalable Buffering”, *Proc. ASP-DAC*, 2023.

[8] Z. Guo, and Y. Lin, “Differentiable-timing-driven global placement”, *Proc. DAC,* 2022.

[9] B-Y. Wu, R. Liang, G. Pradipta, A. Agnesina, H. Ren and V. A. Chhabria, “Invited Paper: 2024 ICCAD CAD Contest Problem C: Scalable Logic Gate Sizing using ML Techniques and GPU Acceleration”, *Proc. ICCAD*, 2024.

[10] M. M. Ozdal, C. Amin, A. Ayupov, S. M. Burns, G. R. Wilke, and C. Zhuo, “An improved benchmark suite for the ISPD-2013 discrete cell sizing contest”, *Proc. ISPD*, 2013.

[11] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke, and C. Zhuo, “The ISPD-2012 discrete cell sizing contest and benchmark suite”, *Proc. ISPD*, 2012.