Invited: Post-Placement Buffering and Sizing Contest

Andrew B. Kahng
UC San Diego
La Jolla, CA, USA
abk@ucsd.edu

Seokhyeong Kang
POSTECH
Pohang, South Korea
shkang@postech.ac.kr

Sayak Kundu
UC San Diego
La Jolla, CA, USA
sakundu@ucsd.edu

Yiting Liu
UC San Diego
La Jolla, CA, USA
yil375@ucsd.edu

Davit Markarian
UC San Diego
La Jolla, CA, USA
dmarkarian@ucsd.edu

Seonghyeon Park
POSTECH
Pohang, South Korea
seonghyeon98@postech.ac.kr

Zhiang Wang
Fudan University
Shanghai, China
zhiangwang@fudan.edu.cn

Abstract

The ISPD 2026 Contest [22] challenges participants to develop post-detailed placement buffering and sizing tools that optimize timing and fix electrical rule check (ERC) violations under real-world constraints. Unlike prior contests, this contest emphasizes practical physical design challenges including fixed macros and I/Os, power delivery network (PDN) blockages, soft placement blockages, and fixed routing resources. The contest provides eight public benchmarks and four hidden benchmarks, with a range from 15K to 1.4M instances, in the ASAP7 7nm technology node [4] with multi-threshold voltage cell libraries. Evaluation is performed using the open-source OpenROAD infrastructure, with scoring based on timing (total negative slack), power (dynamic and leakage) and penalties for ERC violations, displacement, routing congestion and runtime. This paper describes the contest problem formulation, benchmarks, evaluation methodology, a review of related contests and a two-year roadmap for continuation in the ISPD 2027 Contest.

CCS Concepts

• **Hardware → Physical design (EDA)**; • **Computing methodologies → Machine learning**.

Keywords

VLSI physical design, buffering, sizing, timing optimization, detailed placement

ACM Reference Format:
Andrew B. Kahng, Seokhyeong Kang, Sayak Kundu, Yiting Liu, Davit Markarian, Seonghyeon Park, and Zhiang Wang. 2026. Invited: Post-Placement Buffering and Sizing Contest. In *Proceedings of the 2026 International Symposium on Physical Design (ISPD '26), March 15–18, 2026, Bonn, Germany.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3764386.3779616

1 Introduction

Post-placement timing closure is increasingly limited by interconnect delay, congestion and electrical rule check (ERC) violations in modern technology nodes. Two key levers for timing optimization after global placement are repeater insertion (buffers and inverters) and gate sizing. The ISPD 2026 Contest challenges contestants to develop a post-*detailed* placement buffering and sizing tool that improves power, performance and area (PPA) while fixing ERC violations under a set of real-world physical 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, which include:

- **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, as in real-world designs, the I/Os are fixed in our testcases.
- **Power delivery network (PDN).** The PDN uses significant routing resources, exacerbating congestion and making DRC-free routability more challenging. As a result, fixing timing and ERC violations in congested regions becomes more difficult.
- **Soft placement blockages.** These blockages represent reserved regions and allow only specific cells (e.g., physical cells, buffers, and inverters) to be placed, which further constrains the available areas in the placement canvas and increases legalization difficulty.
- **Fixed routing resource.** The routing track supply is fixed, so the global router may be forced to detour nets in congested regions to stay within routing resource limits, which in turn makes it difficult to resolve timing and ERC violations.

Together, these real-world constraints significantly complicate post-placement timing optimization by limiting available white-space, increasing local utilization, and raising the difficulty of ensuring legal and congestion-free cell placement.

Contest Scope. The ISPD 2026 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.** Post-detailed placement, interconnect delays can be estimated more accurately than after earlier 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 rules constraints (maximum slew, maximum capacitance, and maximum fanout) must be addressed at the post-placement stage to improve timing analysis accuracy and reduce iterations at the signoff stage.
- **Legal and congestion-aware insertion.** Solutions must satisfy all legality constraints, including with respect to PDN and placement blockages, fixed macros, and fixed I/Os. Inserted repeaters

and resized gates must be placed in legal areas such that global routing finishes within time limits.

The contest is built on OpenROAD [1] [24] infrastructure, yielding 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 [14], which have shown strong potential to address large-scale optimization problems in physical design [5][7][8][9][10][16]. The contest's evaluation framework, along with sponsor-provided compute resources made available to contest teams that pass certain contest stages, will include GPUs. Nonetheless, the primary objective of this contest remains to achieve better PPA results under real-world constraints.

In the following, Section 2 presents the contest problem formulation, and Section 3 describes benchmark details. Section 4 discusses the evaluation environment and metrics. Section 5 reviews related contests and the development of the ISPD 2026 Contest, and presents a two-year roadmap. We conclude in Section 6.

2 Contest Problem Formulation

We now present the contest problem definition, followed by the optimization operations, design constraints, and workflow.

2.1 Problem Definition

Given a post-detailed placement design with fixed macros, I/Os, PDN and placement blockages, the objective is to optimize timing and power while fixing ERC violations through gate sizing, VT-swapping and buffer insertion.

Given:

- A gate-level netlist $N = (G, E)$, where G is the set of gates and E is the set of nets;
- A legalized placement P with cell locations (x_i, y_i) for each gate $g_i \in G$;
- A standard cell library L with multiple drive strengths and threshold voltage variants for each logic function; and
- Design constraints including clock period, input/output delays and ERC limits (maximum slew, maximum capacitance and maximum fanout).

Find:

- For each gate g_i: a library cell $l_i \in L$ with appropriate drive strength and threshold voltage;
- A set of buffers and inverters B to insert, with locations and connectivity; and
- An updated placement P' that remains legal.

Minimize: A weighted combination of:

- Total negative slack (TNS) for timing optimization;
- Dynamic and leakage power consumption;
- ERC violations (slew, capacitance and fanout);
- Runtime of the developed buffering and sizing tool as well as the overall flow;
- Cell displacement; and
- Global routing overflow.

Subject to:

- Placement legality;
- Logical equivalence (netlist function preserved); and
- Fixed elements (macros, I/Os and physical cells unchanged).

2.2 Optimization Operations

The contest permits three types of netlist transformations.

Gate Sizing. Gate sizing replaces a logic gate with a functionally equivalent cell of different drive strength. Larger drive strengths reduce output slew and delay at the cost of increased area, input capacitance and power. Smaller drive strengths reduce power and capacitive load on upstream drivers but may increase delay and output slew.

VT-Swapping. VT-swapping replaces a cell with a functionally equivalent cell having different threshold voltage. The ASAP7 library provides three variants:

- **SLVT (Super-Low VT):** Fastest switching but highest leakage;
- **LVT (Low VT):** Fast switching with high leakage; and
- **RVT (Regular VT):** Slower switching but lowest leakage.

Critical paths benefit from SLVT/LVT cells for speed, while non-critical paths use RVT cells to minimize leakage power.

Buffer and Inverter Insertion. Repeater insertion adds buffer or inverter cells to:

- Fix slew violations by breaking long interconnects;
- Fix capacitance violations by isolating high-fanout loads; and
- Improve timing by reducing interconnect delay on critical paths.

Inverters may be inserted in pairs to maintain signal polarity. The number of inverters on any source-to-sink path must be even to preserve logical equivalence.

2.3 Design Constraints

The following constraints must be satisfied by all solutions.

- Macros, I/Os and physical cells are fixed and cannot be moved.
- PDN grid defines blocked routing layers; cells with pin shapes on blocked layers cannot be placed in those regions.
- Soft placement blockages permit only physical cells, along with buffers and inverters, to be placed within their boundaries.
- Routing track supply is fixed.
- Netlist restructuring is limited to repeater insertion, sizing and VT swapping.
- The output netlist must remain functionally equivalent to the input netlist (as verified via logic equivalence checking).
- Timing analysis uses setup (late-mode) checks with a single PVT corner and ideal clock.
- Logic gates (except repeaters) must not move beyond a specified displacement threshold; excess movement is penalized.

2.4 Workflow

Figure 1 shows the contest workflow. The inputs to contestants include (i) a debuffered gate-level netlist (.v file) that is input to OpenROAD global placement, (ii) a post-detailed placement, legalized placement (.def file) and (iii) design constraints (.sdc file) including clock period and I/O delays. Contestants' tools perform buffering and gate sizing, and must output the modified .v, .def and changelist files. The evaluation flow then performs legality checks (`checkPlacement`) and equivalence checks, followed by global routing, parasitic extraction and OpenSTA-based timing and power analysis to report final QoR metrics.

Input Formats. Each benchmark includes 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 containing the logic description, look-up tables for delay, slew and power with the area of each cell;
- **.sdc file:** Specifies input port delay and transition, output port delay and capacitance, and specified clock period;
- **.v file:** Debuffered gate-level netlist that is input to OpenROAD global placement; this netlist is debuffered with the exception of the I/O buffering added through the OpenROAD-flow-scripts (ORFS) [25] flow; and
- **.def file:** Post-detailed placement, legalized placement with fixed I/Os, macros, PDNs, and placement and routing blockages.

Output Formats. The tool must return a legalized placement consisting of:

- **Modified Verilog Netlist (.v):** Includes only repeater insertions and cell sizing modifications; and
- **Updated DEF File (.def):** Reflects the placement solution (must be legalized).

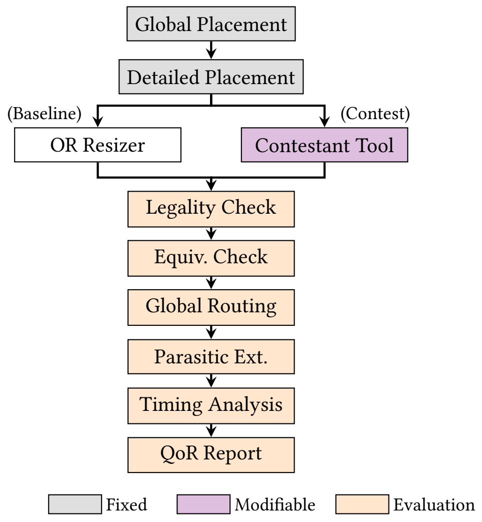


Figure 1: ISPD 2026 Contest flow. The contestant's buffering and sizing tool replaces the OpenROAD Resizer step. Fixed placement stages (gray) provide inputs; the evaluation flow (orange) performs equivalence checking, global routing, parasitic extraction and timing analysis to measure QoR.

3 Benchmarks

We evaluate contestants' solutions using a diverse benchmark suite that includes eight public designs and four hidden designs. These designs are drawn from the OpenROAD-flow-scripts [25] and MacroPlacement [30] [2] [3] repositories. They are implemented in the ASAP7 [4] technology node using a 7.5T multi-threshold voltage (multi-VT) cell library. The synthesized design sizes range from 15K to 1.4M instances and include both macro-free designs (four public designs) and macro-containing designs (eight designs: four public and four hidden).

Technology Platform. All benchmarks use the ASAP7 technology [4], which is a predictive 7nm academic process design kit. The cell library includes multiple threshold voltage variants (SLVT, LVT and RVT) and multiple drive strengths, enabling contestants to explore timing-power tradeoffs through gate sizing. The library includes standard cells for combinational logic (AND-OR, OR-AND, simple gates), sequential elements (flip-flops) and buffers/inverters for repeater insertion.

Designs. Table 1 summarizes the public benchmark characteristics. The benchmarks include designs with varying target clock periods (TCPs) ranging from 200ps to 1300ps and placement utilization ranging from 30% to 70%. Each design has two variants. The base variant (without suffix) uses commercial synthesis with manual macro placement. The *v2* variant uses Yosys [31] for synthesis and OpenROAD for macro placement, and includes soft placement blockages that further constrain the available whitespace for buffer insertion and gate sizing. Both variants use ORFS for physical cell (e.g., tapcell) insertion, power delivery network (PDN) insertion, global placement and detailed placement. Four hidden benchmarks are derived from AR37 and BSG_CHIP with different TCPs, placement utilizations and blockages.

Table 1: Benchmark characteristics. TCP denotes target clock period in picoseconds. Util denotes placement utilization. #blocks denotes the number of soft placement blockages.

Design	TCP	Util	#std cells	#macros	#nets	#pins	#blocks
AES	250	0.40	15K	0	14K	389	0
JPEG	350	0.70	50K	0	44K	48	0
AR37	900	0.30	0.1M	37	147K	496	0
AES_v2	200	0.40	15K	0	15K	390	8
JPEG_v2	450	0.65	50K	0	61K	49	26
AR37_v2	950	0.45	0.1M	37	183K	497	41
BSG_CHIP	1200	0.30	0.9M	220	1.1M	136	83
BSG_CHIP_v2	1300	0.50	1.4M	220	1.3M	137	90

Figure 2 shows the placement layouts of the four *v2* benchmarks, illustrating the placement blockages (shown in light magenta) that contestants must navigate when inserting buffers or moving cells. The yellow triangles indicate I/O pin locations on the chip boundary.

4 Evaluation

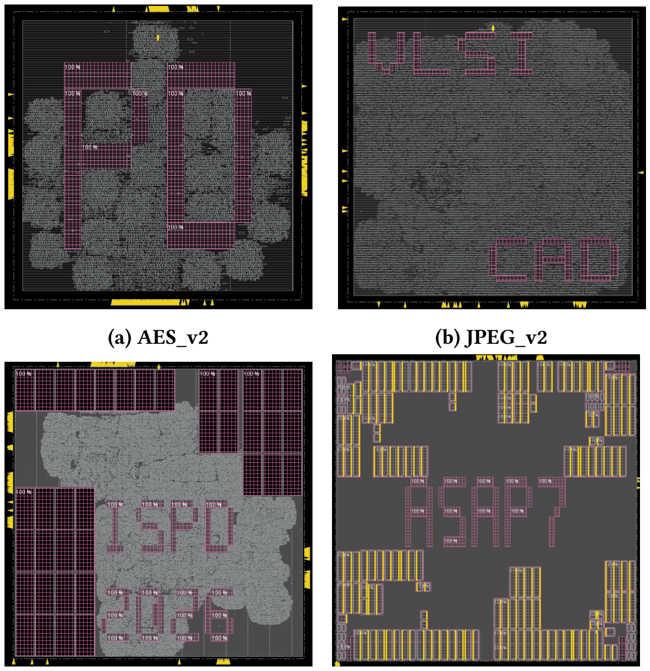
This section describes the evaluation framework. We present the evaluation environment, followed by the metrics used to score contestants' solutions.

4.1 Evaluation Environment

The contest provides an evaluation environment via Docker and Apptainer (formerly Singularity) containers. All participants must build on top of this environment. If additional packages are needed, participants provide a `setup.sh` script that completes in under two hours to set up the environment. More details about the submission requirements are described in the ISPD 2026 Contest GitHub repository [22].

Compute Resources. We provide an evaluation environment via the Purdue Anvil supercomputer [28], with a specified number of CPU and GPU hours in each team's allocation, after successful alpha submission. Purdue Anvil offers 128-core/128-thread AMD EPYC-7763 CPUs and NVIDIA A100 GPUs, with CUDA 12.6 and Apptainer. The evaluation environment has the following limits:

- One NVIDIA A100 GPU (40 GB)



(a) AES_v2 **(b) JPEG_v2**

(c) AR37_v2 **(d) BSG_CHIP_v2**

Figure 2: Placement layouts of the four *v2* benchmarks. Light magenta regions indicate placement blockages. Yellow triangles indicate I/O pins.

- 16 physical CPU cores/threads
- 64 GB memory
- 200 GB disk

Software Infrastructure. The evaluation uses a modified OpenROAD [24] (based on commit hash: 7559f96) and OpenROAD-Flow-Scripts [25] (commit hash: 66e441c). The evaluation flow performs the following steps: (i) read technology LEF files and standard-cell libraries; (ii) read placement DEF files and Verilog netlist; (iii) check placement legality and logical equivalence; (iv) perform global routing with congestion reporting; (v) extract parasitics from global routing; and (vi) run timing analysis and power analysis to report violations and metrics. The evaluation flow is contained within a hybrid Docker and Apptainer container, with support for CUDA and an included copy of Miniconda 3 to enable the use of GPU-accelerated ML frameworks.

4.2 Evaluation Metrics

Contestants' solutions are evaluated based on global routing quality, runtime efficiency and legality, using OpenROAD's timing and power analysis infrastructure. The scoring metric compares normalized improvement versus a baseline [19] (OpenROAD Resizer with `repair_design` and `repair_timing` commands). Table 2 shows the weights for each score component. These weights may be adjusted and finalized after the alpha submission, and will be incorporated in working code in the contest GitHub repository [22].

PPA Metrics. The primary PPA (power, performance, area) metrics are calculated as

$$S_{PPA} = w_{\text{tns}} \times \Delta_{\text{TNS}} + w_{\text{dpower}} \times \Delta_{\text{dpower}} + w_{\text{lpower}} \times \Delta_{\text{lpower}}, \quad (1)$$

where Δ_{TNS} is the normalized TNS improvement

$$\Delta_{\text{TNS}} = \frac{\text{TNS} - \text{TNS}_{\text{baseline}}}{|\text{TNS}_{\text{baseline}}| + \epsilon}, \quad (2)$$

Δ_{dpower} is the normalized dynamic power (DPower) improvement

$$\Delta_{\text{dpower}} = \frac{\text{DPower}_{\text{baseline}} - \text{DPower}}{\text{DPower}_{\text{baseline}}}, \quad (3)$$

Δ_{lpower} is the normalized leakage power (LPower) improvement

$$\Delta_{\text{lpower}} = \frac{\text{LPower}_{\text{baseline}} - \text{LPower}}{\text{LPower}_{\text{baseline}}}, \quad (4)$$

w_{tns}, w_{dpower} and w_{lpower} are weights for each PPA component. All timing and power results are reported by OpenSTA [27].

ERC Violation Penalty. ERC violations are penalized based on the sum of violation values after global routing

$$P_{\text{ERC}} = w_{\text{slew}} \cdot \Delta_{\text{slew}} + w_{\text{cap}} \cdot \Delta_{\text{cap}} + w_{\text{fanout}} \cdot \Delta_{\text{fanout}}, \quad (5)$$

where Δ_{slew} is the normalized slew violation reduction

$$\Delta_{\text{slew}} = \frac{\text{SlewViolation} - \text{SlewViolation}_{\text{baseline}}}{|\text{SlewViolation}_{\text{baseline}}| + \epsilon}, \quad (6)$$

Δ_{cap} is the normalized capacitance violation reduction

$$\Delta_{\text{cap}} = \frac{\text{CapViolation} - \text{CapViolation}_{\text{baseline}}}{|\text{CapViolation}_{\text{baseline}}| + \epsilon}, \quad (7)$$

Δ_{fanout} is the normalized fanout violation reduction

$$\Delta_{\text{fanout}} = \frac{\text{FanoutViolation} - \text{FanoutViolation}_{\text{baseline}}}{|\text{FanoutViolation}_{\text{baseline}}| + \epsilon}, \quad (8)$$

and w_{slew}, w_{cap} and w_{fanout} are corresponding weights.

Runtime. Runtime efficiency is evaluated using both the buffering and sizing tool runtime and the total flow runtime. The tool runtime penalty is defined as

$$R_{\text{tool}} = \frac{t_{\text{tool}} - t_{\text{baselineTool}}}{t_{\text{baselineTool}}}, \quad (9)$$

where t_{tool} is the runtime of the contestant-developed tool, and $t_{\text{baselineTool}}$ is the runtime of the OpenROAD Resizer.

The total flow runtime penalty is defined as

$$R_{\text{flow}} = \frac{t_{\text{flow}} - t_{\text{baselineFlow}}}{t_{\text{baselineFlow}}}, \quad (10)$$

where t_{flow} is the total runtime of the contestant's flow (including buffering, sizing, and the fixed evaluation flow), and $t_{\text{baselineFlow}}$ is the runtime of the baseline flow.

The overall runtime penalty is computed as

$$R = w_{\text{toolRuntime}} \cdot R_{\text{tool}} + w_{\text{flowRuntime}} \cdot R_{\text{flow}}, \quad (11)$$

where $w_{\text{toolRuntime}}$ and $w_{\text{flowRuntime}}$ are the runtime weights. A maximum runtime limit is enforced for each testcase, with final limits determined after the alpha submission.

Average Displacement Penalty. Placement legality is evaluated using the average Manhattan displacement of movable cells from their original positions

$$P_{\text{dis}} = w_{\text{dis}} \cdot \frac{D_{\text{avg}} - D_{\text{avg,baseline}}}{D_{\text{avg,baseline}}}, \quad (12)$$

where D_{avg} is the average displacement of the contestant's solution and $D_{\text{avg,baseline}}$ is the displacement of the baseline solution.

Global Routing Overflow Penalty. Poor global routing quality is further penalized using routing overflow metrics. The maximum overflow penalty is defined as

$$P_{\text{max}} = \frac{O_{\text{max}} - O_{\text{max,thr}}}{O_{\text{max,thr}}}, \tag{13}$$

and the total overflow penalty is defined as

$$P_{\text{total}} = \frac{O_{\text{total}} - O_{\text{total,thr}}}{O_{\text{total,thr}}}, \tag{14}$$

where O_{max} and O_{total} are the maximum and total global routing overflow values after routing, and $O_{\text{max,thr}}$ and $O_{\text{total,thr}}$ are the corresponding thresholds.

The overall overflow penalty is computed as

$$P_{\text{overflow}} = w_{\text{maxOverflow}} \cdot P_{\text{max}} + w_{\text{totalOverflow}} \cdot P_{\text{total}}, \tag{15}$$

where $w_{\text{maxOverflow}}$ and $w_{\text{totalOverflow}}$ are the overflow weights.

Table 2: Scoring weights for each metric component. TBD values will be determined after the alpha submission.

Metric	Weight	Description
w_{tns}	80	Total negative slack
w_{dpower}	40	Dynamic power
w_{lpower}	40	Leakage power
w_{slew}	0.001	Slew violation
w_{cap}	10	Capacitance violation
w_{fanout}	1	Fanout violation
$w_{\text{toolRuntime}}$	TBD	Tool runtime
$w_{\text{flowRuntime}}$	1	Flow runtime
w_{dis}	TBD	Displacement
$w_{\text{maxOverflow}}$	1	Max GR overflow
$w_{\text{totalOverflow}}$	1	Total GR overflow

Hard Constraints. The following hard constraints must be satisfied by all submitted solutions.

- **Placement legality:** Final placement must pass legality checks performed by the OpenROAD `checkPlacement` command. The following checks are enforced.
  - *Placed check:* All cells must have valid placement status.
  - *In rows check:* Cells must be properly placed within defined rows, and cell orientation must match row/site orientation.
  - *Site alignment check:* Cell x-coordinate must be aligned to site width, and y-coordinate must be on a valid row coordinate.
  - *Overlap check:* No cell overlaps are permitted.
  - *Blocked layers check:* Cells with pin shapes on blocked layers must not overlap blocked routing layers.
  - *Soft blockage check:* Only buffers, inverters and physical cells (tapcells, fillers, endcaps, tie cells) may be placed in soft blockage regions.
- **Logical equivalence:** The netlist must be logically equivalent. Several checks are performed.
  - *Sizing equivalence:* When sizing an instance, the new library cell must be functionally equivalent to the original master cell (same logic function, but different drive strength or threshold voltage). Equivalence is determined using a predefined equivalent cell list provided with the contest benchmarks.
  - *Instance insertion:* Only buffers and inverters may be inserted. No other cell types can be added to the netlist, and no existing cells (except those being resized) can be removed.
  - *Pin connectivity:* Original driver-to-sink pin connections must be preserved; pin swapping on gates is not permitted.
  - *Buffer tree parity:* For each source-to-sink path in the original netlist, the buffered tree must contain an even number of inverters to preserve logical equivalence. This is verified efficiently in $O(n)$ time (where n is the number of cells in the buffer tree) using Algorithm 1.

Algorithm 1: Buffer Tree Parity Verification

```
Input: Original netlist $N$, post-optimization netlist $N'$
Output: true if parity check passes, false otherwise
1  foreach net $n$ in $N$ do
2      $src \leftarrow$ driver pin of $n$;
3      $origSinks \leftarrow$ sink pins of $n$;
4      $validSinks \leftarrow \emptyset$;
5      $Q \leftarrow$ queue with $(src, 0)$;        // (pin, inverter count)
6      while $Q$ is not empty do
7          $(pin, cnt) \leftarrow Q$.dequeue();
8          // $cnt$ is #inv found in $N'$ between original
               source-sink pairs
9          foreach sink $s$ connected to pin via net edge in $N'$ do
10             $cell \leftarrow$ instance containing $s$;
11             if $cell$ is newly inserted then
12                 if $cell$ is inverter then
13                     $Q$.enqueue(output of $cell$, $cnt + 1$);
14                 else
15                     $Q$.enqueue(output of $cell$, $cnt$);   // buffer
16             else
17                 // Original cell: check parity
18                 if $cnt$ mod 2 == 0 then
19                     $validSinks \leftarrow validSinks \cup \{s\}$;
20     if $origSinks != validSinks$ then
21         return false;
22 return true;
```

The algorithm performs a Breadth-First Search (BFS) traversal from each driver pin to verify inverter parity.

- Lines 1–4: For each net, extract the driver pin (src), collect original sink pins ($origSinks$), and initialize an empty valid sinks set.
- Line 5: Initialize BFS queue with the source pin and inverter count of zero.
- Lines 6–7: Process each queued pin by dequeuing and examining its connected sinks in the optimized netlist.
- Lines 8–15: For newly inserted cells, inverters increment the count ($cnt + 1$) while buffers preserve it (cnt), and the output pin is enqueued for further traversal.
- Lines 16–19: For original cells (sinks), mark the sink as valid only if the inverter count is even (cnt mod 2 == 0), ensuring signal polarity is preserved.
- Lines 20–21: After processing all paths, return false if any original sink is unreachable with even parity.
- Line 22: Return true if all nets pass the parity check.

- **No floorplan perturbation:** The following elements must remain unchanged from the input.
  - *Fixed locations:* Macro cells and I/O pins must not be moved.
  - *Physical cells:* Standard cells that are not being resized must remain at their original locations.
  - *Layout elements:* Blockages and PDN must be preserved.

Final Score. The final score is computed as

$$S_{\text{final}} = C_{\text{HC}} \cdot (S_{\text{PPA}} - P_{\text{ERC}} - R - P_{\text{dis}} - P_{\text{overflow}}), \tag{16}$$

where $C_{\text{HC}} = 1$ if all hard constraints are satisfied and $C_{\text{HC}} = -\infty$ otherwise.

5 Discussion

This section reviews relevant previous contests, the development of this ISPD 2026 Contest, and the two-year roadmap leading to the ISPD 2027 Contest.

Table 3: Comparison of contest features across prior related contests and the ISPD 2026 Contest. DPL = Detailed Placement; RC-GRT = RC extraction from global routing; STA-PT = PrimeTime-based static timing analysis.

Features	ISPD		ICCAD		MLCAD	Ours
	'12 [12]	**'13** [13]	**'24** [15]	**'25** [11]	**'25** [6]	
Physical Design Awareness						
DPL-Aware	×	×	✓	✓	✓	✓
Macro	×	×	✓	✓	✓	✓
PDN	×	×	✓	✓	✓	✓
Blockage	×	×	×	×	×	✓
PDK-Routable	×	×	✓	✓	✓	✓
Parasitic Model						
C-Lumped	✓	×	×	×	×	×
RC-Distributed	×	✓	✓	✓	✓	✓
RC-Placement	×	×	×	✓	×	×
RC-GRT	×	×	✓	×	✓	✓
Timing Analysis						
STA-PT	✓	✓	×	×	×	×
STA-OpenSTA	×	×	✓	✓	✓	✓
Optimization Capabilities						
Buffering	×	×	×	✓	✓	✓
VT-Swapping	✓	✓	×	✓	✓	✓
Sequential	×	×	×	✓	×	✓
Restructuring	×	×	×	×	✓	×
Scale & Compute						
950K+ #Inst	✓	✓	×	×	×	✓
GPU	×	×	✓	✓	✓	✓

5.1 Review of Related Contests

Several prior contests have explored aspects of gate sizing, buffering and timing optimization. However, these contests typically simplified key elements of real-world physical design flows. Table 3 summarizes key feature differences.

ISPD 2012–2013 Discrete Gate Sizing Contests. The ISPD 2012 [12] and ISPD 2013 [13] Contests focused on timing-driven gate sizing using discrete standard-cell libraries, with the objective of meeting setup timing constraints while minimizing leakage power. These contests operated purely at the logic level, assuming fixed netlists without placement or physical awareness. Consequently, they did not consider congestion, PDN, legalization, or other physical feasibility constraints.

ICCAD 2024 Contest Problem C: Scalable Logic Gate Sizing. The ICCAD 2024 Contest Problem C [15] focused on scalable gate sizing for timing and power optimization, encouraging the use of machine learning techniques and GPU acceleration. This contest disallowed buffer insertion, netlist edits, or cell relocation.

ICCAD 2025 Contest Problem C: Incremental Placement Optimization Beyond Detailed Placement. The ICCAD 2025 Contest [11] addressed post-detailed placement optimization, allowing simultaneous gate sizing, buffer insertion, and limited cell relocation within a legalized design context. While this formulation is similar to that of ISPD 2026 Contest, the testcases were simplified and did not capture real-world constraints such as PDN-induced placement gaps and complex blockages. In addition, the contest focused on post-placement PPA metrics without consideration of routing quality.

MLCAD 2025 Contest: ReSynthAI: Physical-Aware Logic Resynthesis for Timing Optimization Using AI. The MLCAD 2025 Contest [6] explored AI-driven logic resynthesis for timing optimization, introducing the CircuitOps [17] abstraction to support graph-based learning and structure-aware transformations. The contest allowed cloning, resizing, and logic rewiring operations. However, the problem formulation remained abstracted from actual implementation flows, omitting physical blockages and routing considerations.

5.2 ISPD 2026 Contest Development

The ISPD 2026 Contest differs from prior contests by incorporating key real-world constraints, including fixed macros, user-specified soft placement blockages, PDN–induced gaps in the placement site map and fixed I/O locations. These constraints reduce the available placement area and increase legalization and congestion complexity, thereby significantly raising the difficulty of post-placement buffering and gate sizing.

The contest infrastructure and benchmarks were developed in close collaboration with the OpenROAD developers. They provided valuable feedback during benchmark construction and evaluation flow design, helping identify scalability limitations and corner cases. In particular, the evaluation flow has been refined to support large-scale designs by excluding extremely large fanout nets (e.g., clock net) from global routing [20], since the contest focuses on post-placement buffering and sizing for data paths under ideal clock assumptions. In addition, the OpenROAD placement verification flow has been extended to correctly honor soft placement blockages, ensuring consistent and physically meaningful legality checking [29].

To support a deployable and reproducible evaluation environment, the contest provides compute resources to contestants after successful alpha submissions. Specifically, the contest provides evaluation environment on the Purdue Anvil supercomputer [28], with a specified allocation of CPU and GPU hours. Contestants are expected to use their own computational resources for primary development and to leverage the provided environment for tuning and practice evaluation.

5.3 Two-year Roadmap

Building on the foundation established in ISPD 2026 Contest, our plan is for the ISPD 2027 Contest to further advance the research context by introducing more challenging and realistic conditions. The goal is to stress the robustness, adaptability and scalability of buffering and sizing strategies under tighter physical and timing constraints. Enhancements under consideration for the ISPD 2027 Contest include the following.

- **Multi-Corner Multi-Mode (MCMM) timing constraints.** Timing and power must be optimized across multiple process-voltage-temperature corners and operating modes. This introduces significant complexity for buffering and sizing decisions, which must meet timing constraints across all mode-corner combinations while balancing power consumption.
- **Incremental infrastructure improvements.** As OpenROAD capabilities advance, the evaluation infrastructure will incorporate higher cell utilization targets, updated baseline flows, and refined legality checks.
- **Post-CTS sizing and buffering.** Extending optimization to post-clock tree synthesis will enable contestants to leverage useful clock skew for timing improvement through coordinated data path and clock path optimization.
- **Hold timing constraints.** Including hold timing checks will reflect real-world signoff constraints, and will require sizing and buffering decisions to balance setup and hold margins simultaneously.
- **Iterative ECO-style optimization.** In ECO-style workflows, tools will iteratively generate sizing and buffering solutions, evaluate results, and refine decisions to progressively improve PPA across multiple optimization passes.
- **Multi-stage evaluation.** Evaluation of sizing and buffering quality at multiple design stages, including post-placement, post-CTS and post-route, will enable assessment of tool robustness and adaptability across the physical design flow.
- **Community and contestant feedback.** Importantly, the ISPD 2027 Contest will also be shaped by feedback from ISPD 2026 contestants and the broader EDA research community. We welcome suggestions on problem formulations, evaluation metrics and new challenges that reflect emerging industry needs.

6 Conclusion

This paper has presented the ISPD 2026 Post-Placement Buffering and Sizing Contest, describing the problem formulation, benchmark suite, and evaluation methodology. Building on insights from prior contests, this year's contest is designed to reflect realistic industrial challenges while remaining accessible to the research community. A two-year roadmap will support continued development and community engagement for the ISPD 2027 Contest. We hope that this work will spur new research innovations and provide a robust platform for evaluating future advances in physical optimization.

Acknowledgments

We thank our sponsors: Purdue University and the NSF Chipshub [18] for providing compute resources for teams and submission evaluation, and NVIDIA for providing prizes for winning teams. We appreciate the support and encouragement of the OpenROAD Initiative [26] and its open-source ecosystem. We thank the ISPD committee for featuring this contest, and in particular Stephan Held and David Chinnery for their interaction and guidance. We also thank Vidya Chhabria, Matt Liberty, Tom Spyrou, Wen-Hao Liu, Yi-Chen Lu, and Mark Ren for their valuable feedback on the contest proposal [23] and description [21].

References

[1] T. Ajayi, V. A. Chhabria, M. Fogaça et al., "Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project", *Proc. DAC*, 2019, pp. 1–4.
[2] C.-K. Cheng, A. B. Kahng, S. Kundu, Y. Wang and Z. Wang, "Assessment of Reinforcement Learning for Macro Placement", *Proc. ISPD*, 2023, pp. 158–166.
[3] C.-K. Cheng, A. B. Kahng, S. Kundu, Y. Wang and Z. Wang, "An Updated Assessment of Reinforcement Learning for Macro Placement", *IEEE Trans. CAD* (2025) (DOI 10.1109/TCAD.2025.3644293).
[4] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy and G. Yeric, "ASAP: A 7-nm FinFET Predictive Process Design Kit", *Microelectronics Journal* 53 (2016), pp. 105–115.
[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] V. Gopalakrishnan, A. Dey, R. Liang, Y. Zhang, H. Ren and V. A. Chhabria, "Invited Paper: MLCAD 2025 Contest on ReSynthAI: Physical-aware Logic Resynthesis using AI", *Proc. MLCAD*, 2025.
[7] Z. Guo and Y. Lin, "Differentiable-Timing-Driven Global Placement", *Proc. DAC*, 2022, pp. 1–6.
[8] A. B. Kahng, Y. Liu and Z. Wang, "Recursive Learning-Based Virtual Buffering for Analytical Global Placement", *Proc. MLCAD*, 2025.
[9] R. Liang, S. Nath, A. Rajaram, J. Hu and H. Ren, "BufFormer: A Generative ML Framework for Scalable Buffering", *Proc. ASP-DAC*, 2023.
[10] 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.
[11] Y.-C. Lu, R. Liang, W.-H. Liu and H. Ren, "Invited Paper: 2025 ICCAD CAD Contest Problem C: Incremental Placement Optimization Beyond Detailed Placement: Simultaneous Gate Sizing, Buffering, and Cell Relocation", *Proc. ICCAD*, 2025.
[12] 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.
[13] 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.
[14] A. Paszke, S. Gross, F. Massa et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library", *Proc. NeurIPS*, 2019, pp. 8024–8035.
[15] 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.
[16] 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.
[17] R. Liang, A. Agnesina, G. Pradipta, V. A. Chhabria and H. Ren, "Invited Paper: CircuitOps: An ML Infrastructure Enabling Generative AI for VLSI Circuit Optimization", *Proc. ICCAD*, 2023.
[18] Chipshub. https://nanohub.org/groups/chipshub/.
[19] Contest baseline script. https://github.com/ABKGroup/ISPD26-Contest/blob/main/scripts/evaluation_baseline.tcl.
[20] GRT: Add option to skip large fanout nets, Pull Request #8822. https://github.com/The-OpenROAD-Project/OpenROAD/pull/8822
[21] ISPD 2026 Contest description. https://github.com/ABKGroup/ISPD26-Contest/blob/main/ISPD26_contest_description.pdf.
[22] ISPD 2026 Contest GitHub. https://github.com/ABKGroup/ISPD26-Contest
[23] ISPD26-27 contest proposal. https://docs.google.com/document/d/1C4sJMvcaS6nxKFl7jdJOxjf9PB2FiT9RiS7whZEipIM.
[24] OpenROAD, *commit hash: 7559f96.* https://github.com/The-OpenROAD-Project/OpenROAD.
[25] OpenROAD-Flow-Scripts, *commit hash: 66e441c.* https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts
[26] OpenROAD Initiative. https://openroadinitiative.org/
[27] OpenSTA. https://github.com/The-OpenROAD-Project/OpenSTA.
[28] Purdue Anvil supercomputer. https://www.rcac.purdue.edu/compute/anvil.
[29] Soft blockage support for detailed placement, Pull Request #9080. https://github.com/The-OpenROAD-Project/OpenROAD/pull/9080
[30] TILOS-MacroPlacement Repository. https://github.com/TILOS-AI-Institute/MacroPlacement.
[31] Yosys Open SYnthesis Suite. https://github.com/YosysHQ/yosys