PPA-Relevant Clustering-Driven Placement for Large-Scale VLSI Designs

Andrew B. Kahng  
UC San Diego  
abk@ucsd.edu

Seokhyeong Kang  
POSTECH  
shkang@postech.ac.kr

Sayak Kundu  
UC San Diego  
sakundu@ucsd.edu

Kyungjun Min  
POSTECH  
kj.min@postech.ac.kr

Seonghyeon Park  
POSTECH  
seonghyeon98@postech.ac.kr

Bodhisatta Pramanik  
UC San Diego  
bopramanik@ucsd.edu

ABSTRACT

Today’s place-and-route (P&R) flows are increasingly challenged by complexity and scale of modern designs. Often, heuristics must trade off between turnaround time and quality of PPA outcomes. This paper presents a clustered placement methodology that improves both turnaround time and final-routed solution quality. Our PPA-aware clustering considers timing, power and logical hierarchy during netlist clustering, effectively reducing problem size and accelerating global placement runtime while improving post-route PPA metrics. Additionally, our machine learning (ML)-accelerated virtualized P&R methodology predicts the best cluster shapes (i.e., aspect ratios and utilizations) to use in P&R of the clustered netlist. With the open-source OpenROAD tool, our methods achieve up to 47% (average: 36%) global placement runtime improvement with similar half-perimeter wirelength (HPWL) and 90% (29%) improvement in post-route total negative slack (TNS). With the commercial Cadence Innovus tool, our methods achieve up to 3.92% (1%) improvement in power and 99% (49%) improvement in TNS.

ACM Reference Format:

1 INTRODUCTION

The placement phase of physical design is central to optimization of performance, power and area (PPA) outcomes, as well as to design space exploration at floorplan/RTL levels and above. Complexity and scale have rapidly increased, such that millions of instances must be efficiently placed within stringent runtime limits. Thus, placement tools rely on heuristics that are often challenged by problem scale, and by the tension between improvement of turnaround time (TAT) and improvement of quality of results (QOR). Clustering has long been seen as a solution to these challenges [20], [11], [9] However, traditional clustering heuristics [12], [6] only optimize a cutsize criterion and do not consider design information (logical hierarchy, timing, switching activity, etc.) that strongly affects PPA outcomes. Recent works, such as [9] and [14], revisit the use of netlist clustering to guide and improve placement flows. These works demonstrate that clustering can either reduce runtime but with PPA degradation [9], or improve PPA but with runtime degradation [14]. By contrast, our present work develops a PPA-aware clustering methodology, and an improved clustered placement approach based on machine learning (ML)-accelerated virtualized place-and-route (V-P&R), to improve both runtime and PPA relative to baseline (academic and commercial) flat placement methods. Following are the key contributions made by our work.

• **PPA-aware clustering.** We consider additional netlist information – logical hierarchy, timing criticality of paths, and switching activity of nets – to achieve PPA-aware clustering. In doing so, we (i) apply a dendrogram-based approach to extract clusters from the netlist logical hierarchy, and (ii) enhance the multilevel clustering framework of [29] [5] to handle logical hierarchy and switching activity of nets. We experimentally demonstrate that our clustering approach achieves noteworthy PPA benefits, and that it outperforms traditional clustering methods when applied in OpenROAD and Cadence Innovus flows (Sections 3.1 and 4).

• **ML-accelerated virtualized P&R.** In the seeded placement approach, a seed placement of clusters is used to induce seed locations of instances, from which the flat P&R flow is continued. Obtaining a high-quality seed placement of clusters requires two elements: how to form the clusters, and how to feed the clusters into a placer. To this end, we use our PPA-aware clusters, and a novel V-P&R framework to determine cluster shapes (utilizations and aspect ratios) to use in the cluster placement. We accelerate the V-P&R framework using a graph neural network (GNN)-based ML model that achieves mean absolute error (MAE) of 0.131 (for label values in the range [0.564, 2.96]) and R2 score of 0.638, (Sections 3.2 and 4.4).

Experimental confirmations. We evaluate our PPA-aware clustering methodology and ML-accelerated V-P&R framework using both OpenROAD and Innovus flows, along with open testcases from the TILOS MacroPlacement [28] and OpenROAD-flow-scripts [27] GitHub repositories. Our methods achieve (maximum, average) percentage improvements of (47, 36) in global placement runtime with similar half-perimeter wirelength (HPWL), compared to the default (i.e., without any clustering or V-P&R) OpenROAD flow. We also achieve (maximum, average) percentage improvements of (4, 1) and (99, 49) in power and post-route total negative slack (TNS), respectively, compared to a standard Innovus flow. To the best of our knowledge, we are the first work to improve both global placement runtime and post-route final PPA simultaneously (Section 4).
2 RELATED WORK

Clustering has been widely used in various stages of VLSI physical design, such as partitioning [5], placement [17] and clock tree synthesis [18]. Popular hypergraph clustering heuristics include First Choice (FC) [12], Best Choice (BC) [1] and cut-overlay [6]. FC and BC find clusters of vertices with stronger intra-cluster connectivity compared to inter-cluster connectivity. The cut-overlay method combines multiple clustering solutions to generate better clusters. These methods predominantly rely on local criteria when finding candidate vertices to cluster. By contrast, community detection algorithms such as Louvain [4] and Leiden [19] adopt a more global perspective, identifying clusters that maximize a modularity score.


With the exception of [14], the above-mentioned works focus on improving global placement runtime. Other gaps are apparent in the literature, e.g., some of the clustering methods used (e.g., BC and Louvain/Leiden) do not scale to large design sizes. Moreover, previous clustering criteria based on cutsize and/or modularity are not well-correlated with PPA outcomes. While [14] proposes a clustering framework that considers additional netlist information for PPA optimization, their approach requires a placed netlist as an input and is runtime-intensive. Our present work addresses these gaps by incorporating PPA-aware clustering and ML-accelerated V-P&R within a clustering-based placement framework that improves both turnaround time and PPA outcomes over strong baseline methods.

3 OUR APPROACH

Our clustering-based placement approach is detailed in Algorithm 1. Figure 1 illustrates the two main components: (i) PPA-aware clustering and (ii) ML-accelerated virtualized P&R. The input is a netlist file (.v, .lib, .lef, .def, .sdc) and a choice of implementation tool (in this work, either OpenROAD [25] or Cadence Innovus [23]).

**Lines 2-7**: We use OpenDB [25] to parse the input netlist, extract the logical hierarchy and construct a hierarchy tree T that captures the hierarchical relationship of instances in the netlist. We then use T and Algorithm 2 to find a hierarchy-based clustering of the instances. These clusters are used to induce grouping constraints [5]. We also extract the top P most critical timing paths, along with vectorless switching activity of nets, using OpenSTA [26].

**Lines 9-10**: Enhanced multilevel clustering adds PPA-awareness to the open-source FC implementation of [29]. Hierarchy-based grouping constraints and path timing criticality (slacks) are applied similarly to [5]. We also introduce hyperedge switching costs to distinguish nets with high switching activity.

**Lines 12-13**: Given the clustered netlist, we estimate the best choices of individual cluster shapes (aspect ratios and utilizations) using an ML-accelerated V-P&R framework (Section 3.2). We run ML-accelerated V-P&R only for clusters that contain more than 200 instances. Cluster shapes are then updated in the cluster .lef file.

**Lines 15-25**: The clustered netlist is placed to obtain a seed placement, according to the choice of Tool. The coordinates of the seed placement are then used to induce a flat seed placement. When Tool is Innovus, we generate the seed placement through a three-step process (Lines 16-20): (i) placing all instances in a cluster at the cluster center; (ii) setting region constraints for clusters whose
shapes were estimated using ML-accelerated V-P&R; and (iii) running incremental placement. When Tool is OpenROAD (Lines 22-25), we first scale IO net weights by $4 [9]$, then generate the seeded placement using the above steps (i) and (iii) only, since OpenROAD cannot handle region constraints.

Lines 27-30: To assess quality of the seeded placement, we collect post-place wirelength (HPWL), then execute CTS and routing to obtain post-route PPA metrics: routed wirelength (rWL), worst negative slack (WNS), total negative slack (TNS) and total power.

### 3.1 PPA-aware Clustering

Our PPA-aware clustering considers logical hierarchy, timing and power information in addition to physical connectivity. Logical hierarchy is obtained using OpenSTA, while timing and power information is obtained using OpenROAD.

**Logical hierarchy.** Netlist clustering based on logical hierarchy is natural to consider, since functionally similar or related instances are often in spatial proximity in the final placement. However, clustering based only on the logical hierarchy can lead to suboptimalities, since other factors such as timing or connectivity also affect the placement and hence the final PPA. Thus, we use the hierarchy-based clusters as guiding clusters (or, grouping constraints), as in [5]. Algorithm 2 formally describes our hierarchy-based clustering; see also Figure 2. Additional details are as follows.

**Lines 2-5:** We interpret the logical hierarchy tree $T$ as the output of hierarchical clustering and construct a dendrogram, $T_{den}$, to visualize the hierarchical relationships derived from $T$ (see Figure 2).

**Lines 7-12:** We levelize $T_{den}$ such that all leaf nodes in $T_{den}$ are at the same level (i.e., having the same path distance from the root). The levelization process replicates all leaf nodes that have levels less than $level_{max}$, the largest level of any leaf node of $T_{den}$. For example, node $x_1$ in Figure 2 is replicated once.

**Lines 14-24:** We evaluate $level_{max}$ − 1 clusterings of the netlist, respectively corresponding to the $level_{max}$−1 levels of $T_{den}$. Evaluation is according to a weighted average Rent exponent criterion [8], defined by

$$R_{ci} = \frac{\ln(E(c_i))/(Int(c_i) + Ext(c_i))}{\ln(|c_i|)} + 1; \quad R_{avg} = \frac{\sum_{c_i \in C}(R_{ci} \times |c_i|)}{|V|}$$

(1)

Here, $R_{ci}$ is the Rent exponent for cluster $c_i$ [8]; $E(c_i)$ is total external hyperedges (i.e., that connect to vertices in other clusters); $Int(c_i)$ is total pins in $c_i$ that connect to external hyperedges; $Int(c_i)$ is total pins that connect to internal hyperedges (i.e., that only connect vertices within $c_i$); and $|c_i|$ is the number of vertices in $c_i$. A “good” cluster has a lower value of $R_{ci}$. We pick the clustering solution with minimum $R_{avg}$ over all $level_{max}$ − 1 clustering solutions.

**Timing and power.** We extract timing information (top $|P|$ timing-critical paths and net slacks) and net switching activity using the OpenSTA tool. We use the findPathEnds function from Search.hh available at [26], with group count ($|P|') = 100000$, endpoint count = 1, unique pins = true, and sort by slack = true. Net switching power is obtained from vectorless power analysis with default tool settings. In particular, we use the findClkedActivity function from Sta.hh available at [26]. We leverage the timing information in our PPA-aware clustering by calculating (i) timing cost $t_p$ for critical path $p$ and (ii) timing cost $t_e$ of hyperedge $e$, as in [5]. We consider power and switching activity by defining a switching cost of hyperedge $e$:

$$s_e = (1 + \frac{\theta_e}{\sum_{e \in E} \theta_e})^\mu$$

(2)

where $\theta_e$ is the switching activity of a hyperedge $e$ and $\mu$ (default=2) is a scaling factor. The heavy-edge rating function of [5] is then extended as:

$$r_{overall}(u,v) = \sum_{e \in I(v) \cup O(u)} (\alpha \cdot w_e + \beta t_e + \gamma s_e) / |e| - 1$$

(3)

where $u$ and $v$ denote the pair of cluster candidates while $I(v)$ denotes the incident hyperedges of $v$. The parameters $\alpha$, $\beta$ and $\gamma$ are scaling factors.

### 3.2 V-P&R and ML-based Acceleration

Cluster shapes (utilization and aspect ratio) significantly impact seed placement and PPA outcomes, as documented in Section 4.4 below. We therefore introduce a virtualized P&R (V-P&R) framework to determine the best shape for each cluster. We further apply GNN-based ML modeling to accelerate the V-P&R framework.

**Virtualized P&R.** The basic idea of V-P&R is that by running place-and-route on the sub-netlist induced by a cluster, we can gain insight into how to model that cluster during seed placement. Figure 3 shows our V-P&R framework. For each given cluster, we first induce the sub-netlist over the instances in the cluster. For each inter-cluster net that is incident to the given cluster, we create input (output) ports in the sub-netlist, corresponding to any sinks (drivers) in the cluster. This sub-netlist is passed to the V-P&R framework, along with 20 different combinations of 5 aspect ratios and 4 utilizations. For each combination of aspect ratio and utilization,
we initialize the floorplan of a “virtual die”, then run placement and global routing on the sub-netlist using the default OpenROAD flow script [27]. We then record HPWL and routing congestion. To identify a combination of aspect ratio and utilization that achieves both good HPWL and low congestion, we define HPWL Cost as

$$\text{Cost}_{\text{HPWL}} = \frac{\text{HPWL}_{\text{avg}}}{\text{Width}_{\text{core}} + \text{Height}_{\text{core}}}$$  \hspace{1cm} (4)

where $\text{HPWL}_{\text{avg}}$ is the average HPWL of nets in the sub-netlist, and $\text{Width}_{\text{core}}$ and $\text{Height}_{\text{core}}$ are respectively the width and height of the virtual die. We also define Congestion Cost as

$$\text{Cost}_{\text{Congestion}} = \frac{\sum \text{TopX\% GCells} \times \text{Congestion}}{\text{TopX\% GCells}}$$  \hspace{1cm} (5)

where $X$ is a hyperparameter (default = 10). Following [13], we define overall Total Costs: $\text{Total Cost} = \text{Cost}_{\text{HPWL}} + \delta \times \text{Cost}_{\text{Congestion}}$, where $\delta$ is a normalization factor (default = 0.01). The aspect ratio, utilization combination that achieves best Total Cost is used to create the cluster’s .lef model during seed placement.

**ML Modeling to Accelerate V-P&R.** As described above, V-P&R determines each cluster’s ideal shape by running OpenROAD 20 times through the end of global routing (each run can require as much as 3 seconds). This effort grows linearly with design size, and can reach undesirable levels. To address this, we implement an ML-based strategy that in practice accelerates our V-P&R framework by approximately 30×. Specifically, we use a GNN-based model to predict Total Cost, replacing execution of OpenROAD in Figure 3.

Our ML model training uses a diversity of clusters generated by perturbing seed and coarsening hyperparameters [29] in our PPA-aware clustering. The training, validation and testing datasets respectively consist of 22700, 5600 and 3200 clusters. Following [9], we sweep aspect ratio in the range [0.75, 1.75] with step size 0.25,5 and sweep utilization in the range [0.75, 0.90] with step size 0.05. This results in 20 distinct (aspect ratio, utilization) combinations, i.e., 20 candidates for the cluster shape. For each candidate, we run (i) V-P&R, (ii) calculate Total Cost, and (iii) use Total Cost as a label.

Figure 4 shows our GNN-based model architecture. We convert a cluster’s sub-netlist to an undirected graph (“Cluster Graph”) using standard clique expansion with edge weight $1/|\{e\} - 1$ for each hyperedge $e$ [16]. Each node in the graph has 28 features, extending [15] with two new features italicized below. These features are:

- **Design parameters**: floorplan utilization and aspect ratio.
- **Cluster-level features**: #cells, #nets, #pins, #nets w/ fanout 5 - 10, #nets w/ fanout > 10, #internal nets, #border nets, total cell area, average cell degree, average net degree, average clustering

4 EXPERIMENTAL EVALUATION

Our PPA-aware clustering framework is written in C++ and built on the OpenROAD infrastructure. Our ML model is implemented using PyTorch Geometric. We make available all codes and scripts at our GitHub repository [22]. We run all experiments on a server with four 2.4 GHz Intel Xeon(R) Gold 6148 processors and 376 GB RAM. For evaluation we use testcases that are publicly available in the MacroPlacement [28] and OpenROAD [27] GitHub repositories. We use six designs (aes, ariane, BlackParrot, jpeg, Megaboom and MemPool Group) and the NanGate45 [24] open benchmark in our experiments. Table 1 lists the main statistics of these benchmarks. We evaluate our PPA-aware clustering and ML-accelerated V-P&R with OpenROAD and Innovus v.21.1.6 For clarity, we divide our validation efforts into two categories: (i) validation of runtime and PPA with OpenROAD (Section 4.1) and (ii) validation of PPA with Innovus (Section 4.2). To show the benefits of our PPA-aware clustering and ML-accelerated V-P&R methods, we present ablation studies in Sections 4.3 and 4.4. Finally, we explore the tuning of hyperparameters in Section 4.5.

4.1 PPA and Runtime Validation (OpenROAD)

We evaluate our PPA-aware clustering and ML-accelerated V-P&R methods using OpenROAD. We compare our post-place HPWL and

---

5Based on aspect ratio and utilization, a floorplan .lef is created for P&R of the sub-netlist. The IO ports are placed with the OpenROAD pin placer; in the NanGate45 enablement these use the metal2 (vertical) and metal3 (horizontal) layers.

6We do not perform any benchmarking of commercial EDA tools. Further, to avoid inadvertent benchmarking, we mask the target clock period values TCP$_{\text{min}}$ in Table 1.
4.3 Comparison of PPA-awareness

We now assess the PPA-relevance of our clustering method, relative to Leiden clustering and TritonPart’s default clustering method (multilevel FC, denoted as MFC in Table 5). We use post-route PPA metrics obtained from using Leiden and multilevel FC in our overall flow. Table 5 presents evaluation with OpenROAD (more results are available in [22]). Routed wirelength is normalized to the value obtained with the default OpenROAD flow. We observe that compared to Leiden, our clustering approach leads to better PPA outcomes – up to 5% improvement in rWL, WNS and TNS and up to 2% improvement in Power. Compared to the multilevel FC, our clustering achieves better percentage PPA improvements (up to 6, 13, 10 and 2, respectively) on the same metrics. These results indicate that consideration of additional netlist information (logical hierarchy, timing path slacks, and switching activity of nets) during clustering helps to improve the final PPA. Thus, the Table 5 data confirm PPA-relevance of our proposed clustering methodology.

4.4 V&P&R Model Evaluation

We evaluate the performance of our GNN-based model using two metrics: (i) mean absolute error (MAE), which evaluates the average magnitude of absolute prediction errors, and (ii) $R^2$ score, which quantifies the amount of variance in the predicted values. The values of the (Total Cost) labels lie in the range [0.564, 2.96] and have a mean of 1.703 with standard deviation 0.727. Our results show that we achieve an MAE of 0.105, 0.113, and 0.131 for the training, validation, and test datasets, respectively. Our $R^2$ score is 0.788, 0.753, and 0.638 for the three datasets. These results confirm the model prediction accuracy of our GNN-based architecture.

Table 2: Evaluation of post-place results with OpenROAD.

<table>
<thead>
<tr>
<th>Design</th>
<th>[9]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>aes</td>
<td>1.007</td>
<td>1.567</td>
</tr>
<tr>
<td>jpeg</td>
<td>0.91</td>
<td>0.52</td>
</tr>
<tr>
<td>ariane</td>
<td>0.946</td>
<td>0.604</td>
</tr>
<tr>
<td>BlackParrot</td>
<td>0.998</td>
<td>0.664</td>
</tr>
<tr>
<td>MegaBoom</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>MemPool Group</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Table 3: Evaluation of post-route results with OpenROAD.

<table>
<thead>
<tr>
<th>Design</th>
<th>Flow</th>
<th>Post-route PPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>aes</td>
<td>Default</td>
<td>1.00 -220</td>
</tr>
<tr>
<td>jpeg</td>
<td>Default</td>
<td>1.00 -410</td>
</tr>
<tr>
<td>ariane</td>
<td>Default</td>
<td>1.00 -200</td>
</tr>
<tr>
<td>BP</td>
<td>Default</td>
<td>1.00 -410</td>
</tr>
</tbody>
</table>

The unit of WNS is $\mu$s, the unit of TNS is ns, and the unit of Power is W.

Table 4: Evaluation of post-route results with Innovus.

<table>
<thead>
<tr>
<th>Design</th>
<th>Flow</th>
<th>Post-route PPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>aes</td>
<td>Default</td>
<td>1.000 -37</td>
</tr>
<tr>
<td>jpeg</td>
<td>Default</td>
<td>1.000 -41</td>
</tr>
<tr>
<td>ariane</td>
<td>Default</td>
<td>1.000 -97</td>
</tr>
<tr>
<td>BP</td>
<td>Default</td>
<td>1.000 -134</td>
</tr>
</tbody>
</table>

The unit of Power is W.

Table 5: Evaluation of our PPA-aware clustering framework.

<table>
<thead>
<tr>
<th>Design</th>
<th>Method</th>
<th>Post-route PPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>aes</td>
<td>Leiden</td>
<td>0.991 -211</td>
</tr>
<tr>
<td>jpeg</td>
<td>MFC</td>
<td>1.029 -219</td>
</tr>
<tr>
<td>ariane</td>
<td>MFC</td>
<td>1.011 -542</td>
</tr>
<tr>
<td>BP</td>
<td>MFC</td>
<td>1.001 -134</td>
</tr>
</tbody>
</table>

The unit of WNS is $\mu$s, the unit of TNS is ns, and the unit of Power is W.

4.2 PPA Validation (Cadence Innovus)

In this section, we validate our PPA-aware clustering and ML-accelerated V&P&R methods with Cadence Innovus v.21.1. Table 4 compares post-route PPA metrics to those obtained with the standard Innovus flow [28]. For all designs, our methods significantly improve most PPA metrics. We achieve maximum (average) percentage improvements of 1.9 (0.2), 98 (35), 99 (49) and 4 (1) in rWL, WNS, TNS and Power, respectively. We observe similar runtime compared to the standard Innovus flow.
Table 6: Evaluation of our ML-based V-P&R framework.

| Design | Shape | Post-route PPA
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PWL</td>
</tr>
<tr>
<td>ariane</td>
<td>Random</td>
<td>0.992</td>
</tr>
<tr>
<td></td>
<td>Uniform</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td>V-P&amp;RML</td>
<td>0.977</td>
</tr>
<tr>
<td>jpg</td>
<td>Random</td>
<td>1.010</td>
</tr>
<tr>
<td></td>
<td>Uniform</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td>V-P&amp;RML</td>
<td>0.996</td>
</tr>
<tr>
<td>MB</td>
<td>Random</td>
<td>0.999</td>
</tr>
<tr>
<td></td>
<td>Uniform</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td>V-P&amp;RML</td>
<td>0.961</td>
</tr>
</tbody>
</table>

The unit of WNS is pm, the unit of TNS is ns, and the unit of Power is W.

5 CONCLUSION

We have developed new PPA-aware clustering and ML-accelerated virtualized P&R methods that improve seed placements for large-scale global placement. For PPA-aware clustering, we adapt the multilevel clustering framework of [29] to consider logical hierarchy, timing paths and switching activities. Our ML-accelerated V-P&R framework efficiently predicts beneficial aspect ratios andutilizations to apply with clusters produced by the PPA-aware clustering. As noted above, our ML-model accelerates the V-P&R framework by 30× with a one-time training cost. Together, these elements enable generation of a high-quality seed placement that leads to final placements with improved post-route PPA metrics. Experimental results confirm both PPA and runtime benefits of our methods. When integrated with the open-source OpenROAD tool, our methods can improve both PPA and runtime, with 29% average TNS improvement and 36% runtime speedup. With the commercial Cadence Innovus tool, we achieve better PPA, with 49% average TNS improvement. Our ongoing research pursues confirmation of the benefits from our methods on additional testcases, design enablements and P&R tools. We are also studying the effects of different cluster shapes (L-shaped, diamond, circle, etc.) on placement, and enhancing power-awareness of our clustering methodology to further improve the post-route power metric. Last, we plan to study the benefits of our PPA-aware clustering and ML-accelerated V-P&R framework in the context of 3D placement.

ACKNOWLEDGMENTS

We thank the authors of [9] for sharing scripts and codes to run their flows. We also thank the authors of [15] for sharing their feature extraction codes and ML model. This work is supported in part by DARPA HR0011-18-2-0032 (OpenROAD), the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2021-0-00754, Software Systems for AI Semiconductor Design) and by the Samsung Advanced Institute of Technology.

REFERENCES