ART-3D: Analytical 3D Placement with Reinforced Parameter Tuning for Monolithic 3D ICs

Gauthaman Murali[1], Sandra Maria Shaji[1], Anthony Agnesina[1], Guojie Luo[2], and Sung Kyu Lim[1]
[1]School of ECE, Georgia Institute of Technology, Atlanta, GA, USA
[2]Peking University, Beijing, China
gauthaman@gatech.edu,limsk@ece.gatech.edu

ABSTRACT

In this paper, we show that true 3D placement approaches, enhanced with reinforcement learning, can offer further PPA improvements over pseudo-3D approaches. To accomplish this goal, we integrate an academic true 3D placement engine into a commercial-grade 3D physical design flow, creating ART-3D flow (Analytical 3D Placement with Reinforced Parameter Tuning-based 3D flow). We use a reinforcement learning (RL) framework to find optimized placement parameter setting of the true 3D placement engine for a given netlist and perform high-quality 3D placement. We then use an efficient 3D optimization and routing engine based on a commercial place and route (P&R) tool to maintain or improve the benefits reaped from true 3D placement till design signoff. We evaluate our 3D flow by designing several gate-only and processor benchmarks on a commercial 28nm technology node. Our proposed 3D flow involving true 3D placement offers the best PPA results compared to existing 3D P&R flows and reduces power consumption by up to 31%, improves effective frequency by up to 25%, and therefore reduces power-delay product by up to 43% compared with commercial 2D IC design flow. These improvements predominantly come from RL-based parameter tuning, as it improves the performance of the 3D placer by up to 12%.

CCS CONCEPTS

• **Computing methodologies** → **Reinforcement learning**; • **Hardware** → **Physical design (EDA)**; **Methodologies for EDA**; *Placement*.

KEYWORDS

Physical Design for 3D ICs, True 3D Placement, RL for Physical Design

ACM Reference Format:
Gauthaman Murali[1], Sandra Maria Shaji[1], Anthony Agnesina[1], Guojie Luo[2], and Sung Kyu Lim[1]. 2022. ART-3D: Analytical 3D Placement with Reinforced Parameter Tuning for Monolithic 3D ICs. In *Proceedings of the 2022 International Symposium on Physical Design (ISPD '22), March 27–30, 2022, Banff, Alberta, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3505170.3506725

1 INTRODUCTION

Monolithic 3D (M3D) integrated circuits (ICs) that leverage Monolithic Inter-tier Vias (MIVs) for inter-die connections have emerged as a promising way to build commercial-quality, industrial-scale designs in 3D fashion. The 3D IC physical design methodology has undergone several critical innovations in the past few decades in terms of floorplanning, placement, routing, and optimization. State-of-the-art 3D design flows rely heavily on 2D physical design tools and partitioning algorithms to build commercial-quality 3D ICs. Specifically, given a netlist, these "Pseudo-3D flows" [1] first utilize commercial 2D physical design tools to generate an initial 2D placement. Then, they leverage bin-based min-cut partitioning algorithm followed by tier-by-tier routing to transform the design into 3D.

Though these pseudo-3D flows [1] do not build the 3D designs in a true 3D manner, they "seem to be" of better quality compared with other existing academic "true" 3D design flows mainly due to their extensive use of commercial tools that provide high-quality optimization. However, based on our study, we find that it is not necessarily true that pseudo-3D flows always yield better power, performance, and area (PPA) metrics. Even though they leverage commercial tools to perform critical optimizations, the pseudo-3D approaches are not aware of the 3D degradation caused by the partitioning algorithm, which results in sub-optimal PPA metrics.

To reap the benefits of 3D ICs in their entirety, we need physical design methodologies involving 3D algorithms, starting from placement to design signoff, integrated into a single tool flow. In this work, we present a 3D place and route (P&R) methodology called ART-3D, Analytical 3D placement with Reinforced parameter Tuning, which lays a foundation for such a physical design methodology by replacing 2D to 3D transformation-based placement engine in the proven pseudo-3D approaches with a placer that is entirely 3D in nature. We also show that such a flow improves the power-delay product of 3D ICs by up to 10% compared to those designed using state-of-the-art 3D P&R flows and by up to 43% compared to 2D ICs.

2 RELATED WORK AND DIFFERENTIATION

Several 3D placers, such as Force-3D [4], ePlace-3D [5], and Non-Linear 3D (NL-3D) [6] compare their wirelength estimation and PPA values post-placement using benchmarks that cannot be designed using a commercial process design kit (PDK). In IC designs, the complete picture of PPA metrics is not known until the entire design is routed and optimized for timing and power. Transformation based pseudo-3D flows [1] are capable of performing the entire P&R flow, thereby providing a standard means to analyze the quality of 3D ICs designed using them. When these academic 3D

Table 1: Qualitative comparison of state-of-the-art Monolithic 3D P&R flows and this work.

	Shrunk-2D [1]	Pin-3D [2]	Snap-3D [3]	ART-3D
Key idea	cell and wire shrinking	placement compaction & pin projection	tier-row snapping	**true 3D placement, cell width shrinking**
Die stacking	separate dies	3D metal stack	3D metal stack	3D metal stack
Strength	first pseudo-3D flow	better buffering/sizing	better power	**best performance, PDP, & EDP**
Weakness	shrinking causes DRC issues	die-by-die legalization & optimization	modified pin locations during optimization	-
Placement	commercial 2D + tier partitioning	commercial 2D + tier partitioning	tier partitioning + commercial 2D	**true 3D placer tuned with RL**
Legalization	die-by-die	die-by-die	both tiers together	both tiers together
Signal routing	die-by-die	3D	3D	3D
Clock tree design	commercial 2D + tier partitioning	commercial 2D + tier partitioning	commercial 2D	commercial 2D
Post routing optimization	not supported	enhanced die-by-die	not performed	enhanced die-by-die

placers are integrated into a full-fledged 3D P&R flow, the results seen are usually worse than pseudo-3D designs as shown in [1], due to non-optimized placer parameters. Additionally, the commercial tool integration approach used for pseudo-3D placement does not work well with true 3D placement. 3D flows involving 3D placers currently possess these drawbacks, which prevent them from reaching their true potential. In this paper:

- We propose a novel 3D P&R flow integrating an academic true 3D placer.
- We improve the placement quality of the academic placer using reinforcement learning (RL) based parameter autotuning. RL tuning has been applied for 2D placement in previous works. But, to the best of our knowledge, this is the first work using RL for 3D placement optimization.
- We introduce a commercial-grade 3D router and optimizer that works well with true 3D placers.

The proposed 3D P&R flow methodology significantly improves the performance, power, and therefore the power delay product (PDP) and energy-delay product (EDP) of 3D ICs compared to 2D and state-of-the-art pseudo-3D flow. The key differences among our proposed flow and various state-of-the-art 3D P&R flows are presented in Table 1.

3 OVERVIEW OF OUR 3D FLOW

For the first time, we integrate an academic true 3D placement algorithm with a commercial P&R tool to create a 3D physical design flow that operates on the entire three-dimensional space of 3D ICs. We use NL-3D [6] as our true 3D placement engine[1]. We enhance the quality of results (QoR) of the placer by tuning its parameters using an RL-based framework. Finally, we perform optimization, routing, and design signoff using a commercial P&R tool. Our overall 3D P&R flow is shown in Fig. 1. We perform physical design of different circuits using the described 3D P&R methodology. The PPA results are presented and compared against the state-of-the-art 3D flows and the standard 2D flow in Section 7.

[1]We have acquired the source code of this placer for modification and integration.

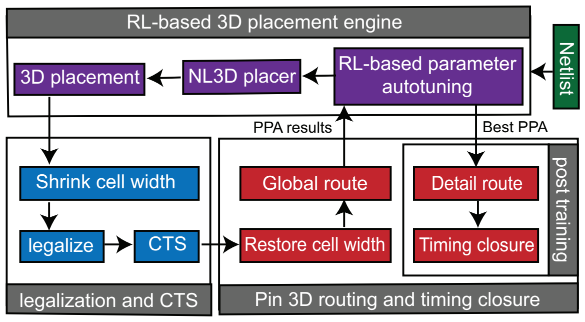


Figure 1: Flow chart depicting our ART-3D P&R flow.

4 NON-LINEAR 3D PLACER INTEGRATION

NL-3D placer [6] is an analytical 3D placement framework. It uses Huber-based local smoothing and Helmholtz-based global smoothing techniques to handle the inter-tier and intra-tier non-overlapping constraints, respectively.

The objective of 3D placement is given as:

$$\min \mathrm{OBJ}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{e \in E} (1 + \gamma_e)(\mathrm{WL}(e) + \alpha_{MIV} \cdot \mathrm{MIV}(e)), \quad (1)$$

where the placement variables $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ are the vectors of 2D (x, y) and tier z locations of the movable cells, E is the set of nets, γ_e and α_{MIV} are respectively the tunable wire and via weights, and $\mathrm{WL}(e)$ and $\mathrm{MIV}(e)$ are respectively the half-perimeter wirelength (HPWL) and the number of vias on net $e \in E$ depending on the placement variables. The objective function is subjected to non-overlapping constraints, made differentiable using the log-sum-exp function and density smoothing techniques, and is solved using a non-linear programming (NLP) solver, as described in [6]. Hence, we call this placement engine as the NL-3D placer. By analytically solving this problem using NLP, we obtain a high-quality 3D placement solution.

The NL-3D placement engine is designed to work on netlists in bookshelf format [7]. However, standard netlists are synthesized using Verilog format. We build an interface to convert synthesized Verilog netlists to bookshelf format using design exchange format

Table 2: Parameters tuned in non-linear 3D placer [6]. The solution space (ρ) is infinite.

Parameter	Description	Type	Value
Grids	Number of placement bins in each x and y dimensions	Integer	[50, 150]
Clustering Depth	Levels of clusters during initial placement	Integer	[1, 5]
Cluster Ratio	Ratio of clusters at current and previous clustering level	Float	[0.1, 0.3]
Wire Weight (γ_e)	Weight of HPWL in placer objective function	Float	[1, 100]
Via Weight (α_{MIV})	Weight of #MIV in placer objective function	Float	[1, 10]
Detail Place Overlap Ratio	Maximum cell overlap percentage allowed to terminate detail placement	Float	[0.1, 0.25]
Target Density	Percentage of maximum density allowed in each placement bin	Float	[0.5, 1.0]
ε	Helmholtz bin density smoothing parameter	Float	[0.5, 2.5]
Υ	Parameter to accurately depict WL using LogSum expression	Float	[0.5, 2.5]
Congestion Driven Placement	Enable routing congestion driven placement	Boolean	[True, False]
Dummy Cells	Add dummy blocks to prevent routing congestion	Boolean	[True, False]

(DEF) and library exchange format (LEF) files, and then perform two-tier 3D placement using NL-3D placer.

The 3D placement algorithm used in [6] predominantly targets through-silicon vias (TSVs). As state-of-the-art M3D ICs use MIVs, which are significantly smaller than TSVs, we modify the algorithm to reduce the penalty of using inter-tier vias to achieve better wirelength results with an optimum number of MIVs. Further, NL-3D placer has several parameters that affect the placement quality. We tune these intelligently using an RL-based approach. Details of this RL framework are explained in Section 5. We run 3D placement on different benchmarks using this upgraded NL-3D placer.

5 RL FRAMEWORK

5.1 Motivation

The NL-3D placer [6] provides several configurable parameters, whose settings can impact the placement result and quality significantly. However, parameter settings that work well for a given netlist do not necessarily apply to other netlists. For example, a parameter configuration that improves the placement quality of a net-dominant netlist can degrade that of a cell-dominant netlist. Due to the infinite number of parameter combinations and the significant run times of each tuning iteration, it is practically infeasible to tune individual parameters manually and observe their effect on the overall placement quality and PPA of different types of netlists. Hence, we resort to an RL-based automated approach to find optimized placement parameter settings suiting the specificities of different netlists. We pick the parameters tabulated in Table 2, as they have a significant impact on placement quality.

5.2 Overview of the Framework

The key idea of our RL framework based on [8] is shown in Fig. 2. The RL agent in our framework solves the problem of identifying optimal placement parameter settings using the following four key RL elements.

State (s): For any given netlist, the set of 11 parameters listed in Table 2 forms a state. The entire state space consists of a single netlist and all possible parameter combinations (ρ).

Tool environment (T): The environment is the complete P&R tool flow methodology performing the entire 3D design. It inputs the current state (parameter settings and netlist) and performs 3D

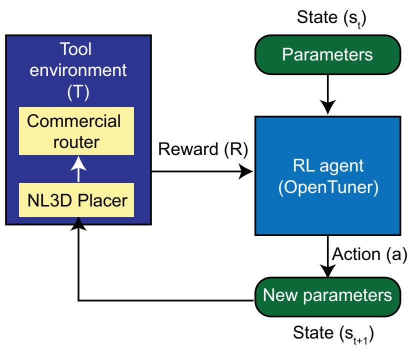


Figure 2: Reinforcement learning framework used to optimize placement quality.

placement using the NL-3D placer. It then performs placement optimization, clock tree synthesis, and global routing using a commercial P&R tool and computes the PPA results of the given state. The parameters used for the commercial tool are set to known best values and are not tuned.

Action (a): Based on a reward, the RL agent repeatedly acts on a given state s_t by tuning all the placement parameters and creates a new state s_{t+1} to start the next optimization iteration until a specific reward threshold is reached.

Reward (R): The reward of a high-quality IC design is the confluence of low power, better timing, and small wirelength. Each action performed by the RL agent is evaluated based on the reward obtained. In order to encapsulate these multiple objectives into a single numerical value, we use a weighted sum in our reward:

$$R = \frac{1}{4}\tanh\left(\frac{WL}{10^7}\right) + \tanh\left(\left|\frac{WNS}{1000}\right|\right) + \tanh\left(\frac{P}{1000}\right) \quad (2)$$

where WL is the wirelength (μm), WNS is the design's worst negative slack (ns), and P is the total design power (mW). To maximize the reward, an action taken by the RL agent should reduce these three metrics. To render these differently scaled values comparable, we first squash the metrics using the tanh function to the range $[0, 1]$, as demonstrated in [9], and then consider 25% of normalized wirelength, 100% of normalized WNS, and 100% of normalized total power towards the reward computation. We give higher weights

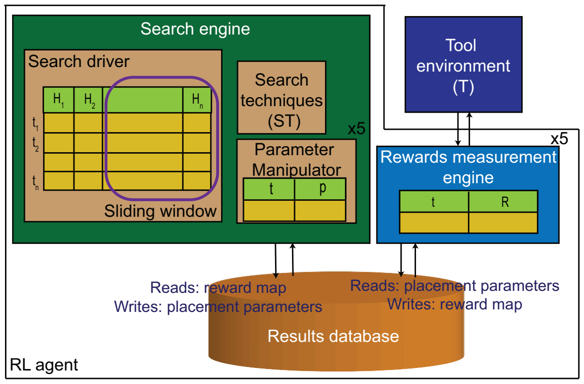


Figure 3: Our RL agent used for placement parameter tuning based on OpenTuner [10].

to the scores of WNS and power P as improving the effective frequency and total power are the primary goals of our 3D flow.

These four key elements help define the **goal of our RL framework**: For a given netlist, find a parameter setting $\boldsymbol{p} \in \boldsymbol{\rho}$ s.t. reward **R** is maximized, where $\boldsymbol{\rho}$ is the set of all parameter combinations.

5.3 RL Agent for Parameter Autotuning

OpenTuner [10] offers a generic framework for autotuning, with a preset of complex search spaces and a set of complex search techniques that help find an optimal solution. In addition, OpenTuner defines ways to combine complex search techniques to handle various solution spaces. We use the RL model called multi-armed bandit (MAB). [11, 12] show how the MAB scheme and OpenTuner framework have been successfully applied to parameter tuning in VLSI. So, we prefer OpenTuner over other RL frameworks.

The OpenTuner RL agent customized for our work is shown in Fig. 3. It orchestrates a set of search techniques, ranking them using the area under the curve (AUC) credit assignment method. We run several search iterations to train our RL model, where each iteration consists of five P&R runs. A P&R run involves physical design stages from placement to global routing (as described in section 6), and takes a few minutes to several hours, depending on the complexity of the netlist. For this reason, we choose to speed up the entire search process by parallelizing it (5 runs for each search iteration), thereby reducing the latency involved in obtaining optimized placement parameters.

Before building accurate AUC scores for a given search technique, the MAB explores all the different search techniques in its ensemble ST, which includes random Nelder-Mead, uniform Nelder-Mead, and random differential evolution. After sufficient search history is collected—the MAB tries different techniques even in a single iteration—it can further identify the best technique and exploit it to optimize the placement parameters. Our RL agent assigns a search technique t to each P&R run based on a trade-off between exploitation and exploration. At each iteration, the RL agent chooses search techniques using the following equation:

$$t = \underset{t' \in ST}{\arg\max}\, \mathrm{AUC}(t') + C\sqrt{\frac{|H|}{H(t')}}, \tag{3}$$

where $|H|$ is the length of the sliding history window, $\mathrm{AUC}(t)$ is the credit assignment term quantifying the performance of technique t in the sliding window, and $H(t)$ is the number of times that technique has been used in the sliding window. The first term in Eqn. 3 indicates how good a given technique is in finding the best placement parameters (exploitation). The second term estimates a confidence bound to allow least used techniques to be picked by the MAB (exploration). The more a technique is used to find placement parameters, the smaller the exploration value becomes for that technique. C is a constant that determines the trade-off between exploration and exploitation.

Based on the techniques chosen by the search driver, a parameter manipulator updates the 11 placement parameters p for each of the five P&R runs in each training iteration within a few milliseconds. A reward measurement engine uses the tool environment T to run the P&R flow based on the updated parameters and measures the reward R given by Eqn. 2. It associates each search technique t with the obtained reward R and thereby assists the search driver in computing the credit $\mathrm{AUC}(t)$.

Table 3: Trained placement parameters for different 28nm benchmarks.

Parameters	AES_128	LDPC	RocketCore
Grids	144	76	99
Clustering depth	1	4	2
Cluster ratio	0.29	0.15	0.25
γ_e	96.77	97.17	80.71
α_{MIV}	4.75	4.55	6.47
Overlap ratio	0.2	0.1	0.25
Target density	0.55	0.63	0.97
ε	1.93	2.29	1.64
Υ	1.5	1.78	0.83
Congestion-driven	0	1	0
Dummy cells	1	1	0

5.4 Training and Testing

We use cell-dominant benchmarks to train our RL framework, such as AES_128 & Netcard, net-dominant benchmarks, LDPC & ECG, and macro-based benchmarks, RocketCore & Cortex A7. We perform on each of these benchmarks around 20 training iterations (equivalent to 100 P&R runs) to create three trained models, one each for cell-dominant, net-dominant, and macro-based benchmarks. Examples of the best parameters found during training on our benchmarks at 28nm node are tabulated in Table 3.

The parameters differ significantly among the benchmarks, indicative of the capability of the RL agent to adapt its tuning to the particularities of each netlist. For example, in AES_128, which is cell-dominant, there is only a single level of clustering, the target density of each placement bin is low to avoid placement congestion, and dummy cells are used to avoid routing congestion. On the other hand, in net-dominant LDPC, there are four levels of clustering, the placement is congestion-driven to avoid routing congestion among several nets, and there is a smaller overlap ratio, as well as a higher bin density compared with AES_128. For RocketCore, containing large memory macros, the bin density and overlap ratio are higher,

Table 4: Training runtime vs PPA trade-off : PPA comparison of LDPC trained with QoR metrics from the end of the flow, post placement optimization, and post global routing.

Metric	Full P&R	Partial P&R (preCTS)	Partial P&R (global route)
Runtime (mins)	2405	829	1432
WNS (ps)	-30	-78	-45
Power (mW)	182.1	171.8	176.6
PDP (pJ)	126.8	127.8	125.2

and the congestion-related parameters are turned off. As the bigger macros with many nets are fixed in the design, the parameters are optimized for the remaining logic cells only.

The RL tuner adapts and tunes the parameters based on the nature of the circuit. Therefore, we do not reuse these trained parameters directly for designing similar benchmarks. Instead, after training the models on various circuits, we use the appropriate trained model, among the three models, to generate optimized parameters for any new circuits and then perform P&R using the newly generated parameters.

5.5 RL Runtime Improvements

Initially, our RL framework was performing multiple iterations of the complete flow (NL3D placement and entire P&R flow using commercial P&R tools), with five parallel runs per design. Depending on the complexity of the netlist, this flow runtime can lengthen the training phase, thus rendering the proposed flow unrealistic, even if the obtained results are much better. To speed up RL training, we compute the reward given by Eqn. 2 after global routing using the commercial tool for each run.

The runtime vs. PPA trade-off on training LDPC (target freq @ 1.5 GHz) computed using the QoRs obtained from different stages of physical design flow are shown in Table 4. Even though using QoR obtained from the preCTS stage (post-placement optimization) reduces the training time tremendously, the final reward obtained after 20 training iterations is much worse than the reward obtained by using the full P&R flow training. However, by using the global routing results, we save close to 40% of training time while still producing the best PPA results.

6 BACK-END DESIGN WITH COMMERCIAL TOOLS

After performing 3D placement using our parameter tuned NL-3D placer, we perform the rest of the physical design using a commercial P&R tool. We import the 3D placement obtained from NL-3D into the commercial tool and perform the following stages.

6.1 3D Placement Optimization

To retain the 3D nature of our proposed P&R flow, we perform commercial placement and placement optimization on a 3D design involving two tiers of standard cells and a 3D metal layer stack in a commercial 2D P&R tool environment. The LEF (physical information file) and LIB (timing file) files are modified appropriately for the 2D tool to perform 3D design. However, placing all the 3D

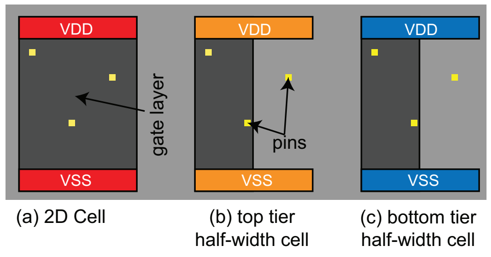


Figure 4: Half width standard cells used in ART-3D flow. Only the width of gate layer is halved, the pins are retained at the original location.

cells in a 2D placement tool leads to placement density violation, as the overall density exceeds 100%. To avoid this issue, we halve the width of the standard cells but retain the pin locations of all the cells at their original position (for example, retaining pins of top-tier standard cells at top-tier metal layers), as shown in Fig. 4.

In the case of hard macro blocks, we retain the original pin locations of the top tier block (if any) and reduce the hard macro dimensions to the site size (minimum possible size) of the PDK used. However, hard macros in bottom tier are retained at their original dimensions. Thus, we tweak a 2D placement tool to function as a 3D placer. However, the 2D P&R tool is incapable of differentiating the cells of two different tiers and swaps cells of one tier with that of the other during optimization leading to area imbalance between the two tiers. This issue is overcome by creating two classes of cell footprints (top/bottom) and forcing the tool to perform cell resizing within the same class, similar to snap3D flow [3]. The look-up table restricts the tool from swapping a logic cell of one tier with that of the other tier. Combining all these techniques, we use the commercial placement algorithm to further improve the placement quality of the NL-3D placer in our work.

6.2 3D Clock Routing and Optimization

After placement optimization, we perform 3D clock tree synthesis and optimization with half-width cells. Similar to placement optimization, we restrict the tool from swapping the cells of one tier with the other with the help of a user-defined 3D cell resizing look-up table. This way, we perform 3D clock routing using a commercial 2D P&R tool. The tool is aware of both 2D and 3D timing paths in the design, and optimizes them simultaneously, leading to optimized clock buffering and optimizing placement density and clock power.

One of the significant merits of our 3D flow is performing clock and signal routing/optimization after partitioning the circuit. A major advantage of a 3D clock and signal routing is that the timing information of all the nets (both 2D and 3D) is available to the router. This makes it easier to estimate each path's timing slack and borrow slack from a path with substantial positive slack and use it to improve the timing of a failing path. The slack borrowing concept is integrated into commercial routing tools and can be effectively used with our 3D flow to achieve better timing performance. Pseudo-3D flows do not benefit from slack borrowing, as clock routing and

Table 5: Comparison of ART-3D against published state-of-the-art 3D flows: Pin-3D (P3D) [2] and Snap-3D (S3D) [3].

28nm Benchmarks (Training)						
	LDPC[2]			RocketCore[3]		
	2D	P3D	A3D	2D	S3D	A3D
Target freq. (GHz)		1.5			1.0	
Footprint (mm^2)	0.11	0.05	0.05	0.31	0.15	0.15
Total WL (m)	2.29	1.43	1.45	1.87	1.46.	1.50
Eff. freq. (GHz)	1.43	1.36	1.41	0.95	0.93	0.97
Total power (mW)	256.4	181.2	176.6	151.4	142.55	142.3
PDP (pJ)	179.5	133.2	125.2	158.9	154.0	146.6
Runtime (min)			1332			2345

optimization is performed in them before partitioning, and can lead to worsening of the placement quality post partitioning.

6.3 3D Routing and Timing Closure

After clock routing and optimization, we restore the original width of the standard cells and original dimensions of the shrunk top-tier hard macros. We then use Pin-3D router and optimizer [2], which is based on commercial P&R tool to perform global and detailed routing, and timing closure on the entire 3D design. During RL training process, we stop each iteration at the global routing stage. We perform detailed routing using Pin-3D router on the best result obtained. We then perform post-routing optimization, using Pin-3D optimizer, in a 3D fashion by placing the cells of both tiers simultaneously. To overcome the placement density issue, Pin-3D fixes one tier of cells and makes them transparent to the P&R tool. As this restricts the movement of cells in one tier while optimizing the other, we do not use this tool flow methodology for pre-routing optimization to achieve better optimization.

During this stage, we allow buffer-resizing to use cells that improve timing and reduce the overall power dissipation. We also perform slack borrowing to further reduce the worst negative slack in the design. Performing enhanced die-by-die optimization using Pin-3D flow is remarkably better than optimizing each tier individually. In enhanced die-by-die optimization, even though individual tiers are optimized, the tool is aware of the entire 3D structure, all parasitics and the timing and power information of cells in both top and bottom tiers. This way, we improve timing, power, and therefore overall EDP of 3D ICs.

7 EXPERIMENTAL RESULTS

7.1 Experimental Settings

We trained our model using various benchmarks as listed in Section 5.4 and tested it on VGA_LCD and Cortex A53. Due to limited space, we present the results of 5 benchmarks only: LDPC, RocketCore, Cortex A7, VGA_LCD, and Cortex A53[2]. We design our benchmarks at commercial 16 and 28nm technology nodes to show that our 3D tool performs better regardless of the technology node. Our 2D designs use 6 metal layers, and 3D designs use 2 tiers and 12 metal layers. MIVs in 3D designs have a pitch of 70nm in 28nm tech node, and 40nm in 16nm tech node.

We compare our proposed flow against 2D and a 3D flow involving manually tuning of NL3D placer parameters to show the benefits of RL tuner in our ART-3D flow in Table 6. The runtimes reported for ART-3D include the time taken for parameter manipulation and RL training as well. The runtime of ART-3D on the tested benchmarks (VGA_LCD & ARM Cortex A53) are comparable to that of other flows.

7.2 ART-3D vs State-of-the-art 3D flows

To show the improvement offered by ART-3D over the state-of-the-art 3D flows, we compare the PPA results of 28nm LDPC and RocketCore against the published Pin3D [2] and Snap-3D [3] results respectively in Table 5. We used the same design and corner settings for a fair comparison. ART-3D provides 6% improvement in PDP on LDPC over Pin-3D and 5% improvement in PDP on RocketCore over Snap-3D. We do not directly compare the results of 28nm Cortex A7 and A53 with the published results as the results are normalized. But approximately, we see a PDP improvement of 5-10% in these benchmarks over the published pin3D[2] and snap3D[3] results.

7.3 PPA Comparisons: 28nm Benchmarks

In this subsection, we compare ART-3D results against results of 2D designs and 3D designs involving manual parameter tuning of NL-3D placer. This comparison in shown in Table 6 Cortex A7, an industry-standard benchmark from ARM, shows 3% improvement in frequency, 10% improvement in total power, thereby offering 12.5% improvement in PDP over the 2D design. This is approximately 8% improvement in PDP over the 3D design involving manual tuning of placement parameters in NL3D placer. The PPA comparison of two of the other 28nm benchmarks used for training: LDPC and RocketCore, are presented in Table 5. In LDPC, ART-3D improves the power consumption by 31% leading to a PDP improvement of 30% over 2D design. ART-3D RocketCore design offers 2% improvement in effective frequency and 6% improvement in power, leading to 8% improvement in PDP over the corresponding 2D design.

Using the trained model, we directly implement another industry-standard processor benchmark, Cortex A53. In Cortex A53, we see a 25% improvement in frequency and 29% improvement in total power, leading to an improvement of 43% in PDP over the 2D design. This is 10% improvement in PDP over the 3D design performed with manual placement parameter tuning.

7.4 PPA Comparisons: 16nm Benchmarks

We compare our results against 2D for 16nm benchmarks as there are no published pin3D and snap3D results on 16nm circuits. LDPC designed using ART-3D does not show much frequency improvement, however, the power improves by 21%, leading to 21% improvement in PDP over the 2D design. When compared with the 3D design involving manual parameter tuning of NL3D placer, our ART-3D placer offers 10% improvement in PDP. As our RL-based training only involves stages from placement through global routing, the training time is significantly shorter, given that we perform around 20 training iterations involving 80 P&R runs.

[2]Due to limited space, we have not included the results of AES_128, Netcard, ECG, 16nm RocketCore and 16nm Cortex A7

Table 6: PPA comparison summary of 2D, Non-linear 3D with manual parameter tuning (NL3D) and ART-3D (A3D) : LDPC, RocketCore and Cortex A7 are some of the circuits used for training the RL model, and VGA_LCD and Cortex A53 are used for inferencing. Results are normalized w.r.t 2D for comparison.

	16nm Benchmarks									28nm Benchmarks					
	Training			Inferencing						Training			Inferencing		
	LDPC			VGA_LCD			Cortex A53			Cortex A7			Cortex A53		
	2D	NL3D	A3D	2D	NL3D	A3D	2D	NL3D	A3D	2D	NL3D	A3D	2D	NL3D	A3D
Target Freq.	3 GHz			8 GHz			1 (no units)			1 (no units)			1 (no units)		
Footprint	1.00	0.50	0.50	1.00	0.50	0.50	1.00	0.50	0.50	1.00	0.50	0.50	1.00	0.50	0.50
Total WL	1.00	0.82	0.77	1.00	0.89	0.85	1.00	0.85	0.81	1.00	1.03	0.84	1.00	0.84	0.78
Eff. Freq.	1.00	0.95	1.00	1.00	0.92	1.00	1.00	0.95	1.05	1.00	0.96	1.03	1.00	1.13	1.25
Total Power	1.00	0.83	0.79	1.00	0.97	0.94	1.00	0.95	0.94	1.00	0.91	0.90	1.00	0.73	0.72
PDP (pJ)	1.00	0.87	0.79	1.00	1.05	0.95	1.00	1.11	0.90	1.00	0.95	0.87	1.00	0.63	0.57
Runtime	1.00	1.19	4.89	1.00	2.54	2.00	1.00	3.12	2.90	1.00	1.02	12.69	1.00	0.59	0.53

Table 7: Capacitance analysis of 28nm & 16nm RocketCore

	28nm		16nm	
Cap. metrics (pF)	2D	A3D	2D	A3D
Gate Capacitance	239	230	173	142
Wire Capacitance	265	221	152	141

VGA_LCD, implemented using the trained RL model, has a total power that is 5.7% lower than the corresponding 2D design. The PDP reduces by 5% than the 2D design. Our ART-3D also offers 9% improvement in PDP over the 3D design involving manual parameter tuning. It is also observed that the runtimes of 16nm designs are higher than the corresponding 28nm designs due to a large number of DRC checks involved in 16nm technology. The DRC rules check time increases further in 3D 16nm designs as there are 2x metal layers in them than the 2D design.

Using the trained model, we also implement Cortex A53. In Cortex A53, we see a 5% improvement in frequency and 6% improvement in total power, leading to an improvement of 10% in PDP over the 2D design. This is 12% improvement in PDP over the 3D design performed with manual placement parameter tuning.

It is observed that the 3D IC PPA improvement over 2D seen in 16nm node is slightly less than 28nm. This is due to higher gate capacitance in FinFET technologies. Until 28nm, wire capacitance dominated gate capacitance. Therefore, the wire capacitance reduction in 3D design offered significant PPA improvements. But in 16nm, due to multiple fins in standard cells, the gate capacitance dominates the wire capacitance. So, the effect of wire capacitance reduction in 16nm 3D designs on PPA improvement is diminished.

Table 7 shows this capacitance comparison of 28nm and 16nm 2D and 3D designs of RocketCore. 28nm 2D RocketCore has dominant wire capacitance, whereas 16nm 2D RocketCore has dominant gate capacitance. Hence, the reduction of wire capacitance in 3D 28nm RocketCore lowers the total design capacitance by 11.2%. Whereas, the wire capacitance reduction in 16nm 3D RocketCore lower the total design capacitance by 8.9% only. This trend is more prominent in cell dominant benchmarks, and hence we see less PPA improvement in 16nm 3D VGA_LCD. However, the latest FinFET technologies at 5 and 3nm nodes have lesser number of 3D fins per standard cell, making gate capacitance less dominant. Therefore, we expect that 3D designs offer better PPA at advanced nodes.

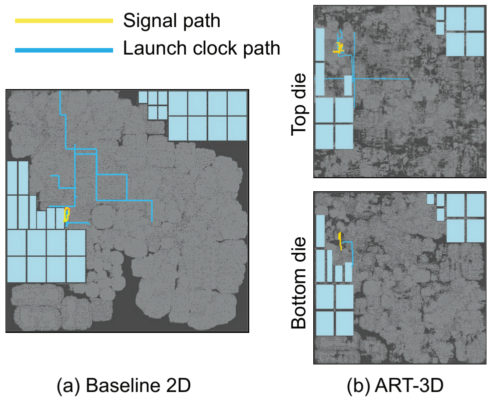


Figure 5: Critical paths in 28nm Cortex A53.

7.5 Detailed Timing Analysis

Critical path metrics of 28nm Cortex A53 designed using our 3D flow are presented in Table 8. As 2D generally offers better timing performance compared to other state-of-the-art flows, we compare the critical path metrics of our ART-3D flow against the 2D metrics. The critical paths in 2D and 3D designs of 28nm Cortex A53 are shown in Fig. 5. It is observed that the ART-3D designs have smaller critical path delays and larger capture clock delays. These metrics help in achieving better timing in ART-3D based designs. However, the launch clock delay is worse than 2D. Lowering this delay can reduce the slack and improve the effective frequency further.

We perform clock metric analysis using 28nm Cortex A53 as it is dominated by sequential logic. The clock metric analysis is presented in Table 9. The reduction in WNS is due to extensive buffering. This results in higher clock wirelength and buffer count.

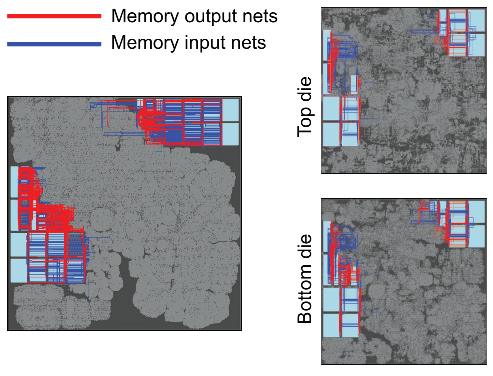
Another important benefit of 3D ICs is that they shorten the memory latency if properly partitioned. We perform memory net analysis on memory-dominant Cortex A53. The results of comparison are presented in Table 10. The memory nets of 2D and ART-3D designs of 28nm Cortex A53 are shown in Fig. 6. It is observed that our 3D flow improves memory net metrics by a significant amount over 2D flow. A massive 25% reduction in maximum memory net delays plays a significant role in improving the overall effective frequency of industry standard processor designs.

Table 8: Critical path analysis of 28nm Cortex A53. Results are normalized w.r.t 2D.

Timing metrics	2D	A3D
Slack	-1.00	-0.13
Critical path delay	1.00	0.90
(+) Launch clock delay	1.00	1.30
Signal arrival time	1.00	0.98
Capture clock delay	1.00	1.62
(+) Clock period	1.00	1.00
(-) Setup time	1.00	0.26
Required arrival time	1.00	1.17

Table 9: Clock metrics comparison for 28nm Cortex A53 . Results are normalized w.r.t 2D

Clock Metrics	2D	A3D
Frequency	1.0	
Clock latency	1.0	0.87
Clock skew	1.0	1.06
Clock WL	1.0	1.04
#Clock buffers	1.0	2.39



(a) Baseline 2D (b) ART-3D

Figure 6: Memory nets in 28nm Cortex A53

Table 10: Memory metrics comparison for 28nm Cortex A53. Results are normalized w.r.t 2D.

Memory Metrics	2D	A3D
Max. mem. latency	1.00	0.75
Avg. mem. latency	1.00	0.53
Max. mem. path length	1.00	0.70
Avg. mem. path length	1.00	0.66
Mem. access power	1.00	0.67

7.6 Why ART-3D Outperforms Pseudo-3D?

Pseudo-3D placers focus on minimizing the MIV usage post an initial 2D placement to achieve a shorter total wirelength. The min-cut based approach can result in few longer critical nets in the process of minimizing the overall wirelength. However, efficient true 3D placement optimizes each net in the design to reduce overall wirelength. Though this approach does not result in the shortest wirelength, optimizing individual nets results in shorter critical nets. Even though the average path length is higher, the maximum path length is smaller, leading to smaller critical net delays and higher effective frequencies. In Table 5, we observe a huge improvement in effective frequency of ART-3D designs over other flows.^{3}

8 CONCLUSION

In this paper, we present ART-3D, a high quality 3D P&R flow, involving RL-enhanced true 3D placement. To the best of our knowledge this is the first work applying RL to 3D placement optimization. Our work improves the true 3D placer performance by up to 12%, making ART-3D outperform 2D and state-of-the-art 3D flows. Our 3D flow offers efficient methodology to design high-frequency circuits. The frequency and power improvement offered by ART-3D can help build high performance industry-standard 3D designs.

ACKNOWLEDGEMENTS

This research is funded by the DARPA ERI 3DSOC Program under Award HR001118C0096, the Semiconductor Research Corporation under GRC Task 2929, and the National Research Foundation of Korea under NRF-2020M3F3A2A02082445.

REFERENCES

[1] H. Park, B. W. Ku, K. Chang, D. E. Shim, and S. K. Lim, "Pseudo-3D Approaches for Commercial-Grade RTL-to-GDS Tool Flow Targeting Monolithic 3D ICs," in ACM International Symposium on Physical Design, ser. ISPD '20, 2020, p. 47–54.

[2] S. Surya, K. Pentapati, K. Chang, V. Gerousis, R. Sengupta, and S. Lim, "Pin-3D: A Physical Synthesis and Post-Layout Optimization Flow for Heterogeneous Monolithic 3D ICs," in 2020 IEEE/ACM International Conference On Computer Aided Design, 2020.

[3] P. Vanna-Iampikul, C. Shao, Y.-C. Lu, S. Pentapati, and S. K. Lim, "Snap-3D: A Constrained Placement-Driven Physical Design Methodology for Face-to-Face-Bonded 3D ICs," in ACM International Symposium on Physical Design, ser. ISPD '21, 2021, p. 39–46.

[4] D. H. Kim, K. Athikulwongse, and S. K. Lim, "Study of Through-Silicon-Via Impact on the 3-D Stacked IC Layout," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 5, pp. 862–874, 2013.

[5] J. Lu, H. Zhuang, I. Kang, P. Chen, and C.-K. Cheng, "EPlace-3D: Electrostatics Based Placement for 3D-ICs," in Proceedings of the 2016 on International Symposium on Physical Design, ser. ISPD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 11–18.

[6] G. Luo, Y. Shi, and J. Cong, "An Analytical Placement Framework for 3-D ICs and Its Extension on Thermal Awareness," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 32, no. 4, pp. 510–523, 2013.

[7] A. E. Caldwell and I. L. Markov, "Toward CAD-IP Reuse: A Web Bookshelf of Fundamental Algorithms," IEEE Design & Test, vol. 19, no. 3, p. 72–81, 2002.

[8] A. Agnesina, K. Chang, and S. K. Lim, "VLSI Placement Parameter Optimization using Deep Reinforcement Learning," in 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2020, pp. 1–9.

[9] A. Agnesina, S. S. K. Pentapati, and S. K. Lim, "A General Framework For VLSI Tool Parameter Optimization with Deep Reinforcement Learning," in NeurIPS 2020 Workshop on Machine Learning for Systems, 2020.

[10] J. Ansel et al., "OpenTuner: An extensible framework for program autotuning," in 2014 23rd International Conference on Parallel Architecture and Compilation Techniques (PACT), 2014, pp. 303–315.

[11] A. Stefanidis, D. Mangiras, C. Nicopoulos, and G. Dimitrakopoulos, "Multi-Armed Bandits for Autonomous Timing-driven Design Optimization," in 2019 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2019, pp. 17–22.

[12] P. Bruel, A. Goldman, S. R. Chalamalasetti, and D. Milojicic, "Autotuning high-level synthesis for FPGAs using OpenTuner and LegUp," in 2017 International Conference on ReConFigurable Computing and FPGAs, 2017, pp. 1–6.

^{3}We use published data for comparison, which prevented us from using other 28 nm and 16nm benchmarks.