A PPA Study of Reinforced Placement Parameter Autotuning: Pseudo-3D vs. True-3D Placers

GAUTHAMAN MURALI, ANTHONY AGNESINA, and SUNG KYU LIM, Georgia Institute of Technology, USA

3D Place and Route (P&R) flows either involve true-3D placement algorithms or use commercial 2D tools to transform a 2D design into a 3D design. Irrespective of the nature of the placers, several placement parameters in these tools affect the quality of the final 3D designs. Different parameter settings work well with different circuits, and it is impossible to manually tune them for a particular circuit. Automated approaches involving reinforcement learning have been shown to adapt and learn the parameter settings and create trained models. However, their effectiveness depends on the input dataset quality. Using a set of 10 netlists and 10–21 handpicked placement parameters in P&R flows involving pseudo-3D or true-3D placement, the dataset quality is analyzed. The datasets are the design metrics obtained through different P&R stages, such as placement optimization, clock tree synthesis, or 3D partitioning and global routing. The training runtime and the quality of the final design metrics are compared. On a pseudo-3D flow, the training takes around 126–290 hours, whereas, on a true-3D placer-based flow, it takes around 305–410 hours. It is observed that the datasets obtained from different stages lead to drastically different final design results. With the RL-based training processes, the quality of results in 3D designs improves by up to 23.7% compared to their corresponding untrained P&R flows.

CCS Concepts: • **Computing methodologies → Machine learning approaches**; • **Hardware → Physical design (EDA)**; **3D integrated circuits;**

Additional Key Words and Phrases: Physical design for 3D ICs, pseudo 3D placement, true 3D placement, RL for 3D physical design

ACM Reference format:
Gauthaman Murali, Anthony Agnesina, and Sung Kyu Lim. 2023. A PPA Study of Reinforced Placement Parameter Autotuning: Pseudo-3D vs. True-3D Placers. *ACM Trans. Des. Autom. Electron. Syst.* 28, 5, Article 75 (September 2023), 22 pages.
https://doi.org/10.1145/3582007

1 INTRODUCTION

Monolithic 3D (M3D) integrated circuits (ICs) that leverage **Monolithic Inter-tier Vias (MIVs)** for inter-die connections are emerging as a promising way to build commercial-quality, industrial-scale designs in 3D fashion. The 3D IC physical design methodology has undergone

This research is funded by the Semiconductor Research Corporation under GRC Task 2929, and the National Science Foundation and the industry members of the CAEML I/UCRC.
Authors' address: G. Murali, A. Agnesina, and S. K. Lim, Georgia Institute of Technology, Atlanta, Georgia, USA, 30308; emails: gauthaman@gatech.edu, agnesina@gatech.edu, limsk@ece.gatech.edu.

1084-4309/2023/09-ART75 $15.00
https://doi.org/10.1145/3582007

Table 1. Variation in Design Metrics of 28 nm Rocketcore on Using Different Placement Parameter Settings on Compact-2D [9] + Pin-3D [16] Flow

Metrics	Run1	Run2	Run3	Run4	Run5	Run6	Run7
Worst negative slack (ns)	−0.142	−0.117	−0.082	−0.239	−0.243	−0.307	−0.324
Max. frequency (GHz)	1.06	1.09	1.13	0.96	0.96	0.90	0.89
Power (mW)	177.6	174.5	168.2	193.9	193.6	195.2	199.5
PDP (pJ)	167.3	160.0	148.4	201.5	201.9	216.1	224.2

several critical innovations in the past few decades with respect to 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" [15] first utilize commercial 2D physical design tools to generate an initial 2D placement. For example, the pseudo-3D flows, such as Shrunk-2D [14], Compact-2D [9] + Pin-3D [16], Snap-3D [20], and Cascade-2D [4], do not perform true 3D placement. They leverage bin-based min-cut partitioning algorithm [6] followed by tier-by-tier routing to transform a 2D design into 3D.

However, academia has been actively involved in developing true-3D placement algorithms, such as ePlace-3D [10], Force-3D [8], **Non-linear 3D (NL-3D)** [11], and TSV aware-3D [7]. The true-3D placers, unlike the pseudo-3D placers, perform three-dimensional placement and do not involve any transformations. The objective functions of these placers consider both 2D and 3D nets while optimizing the **half perimeter wirelength (HPWL)**.

Regardless of the nature of the placer involved in 3D place and route (P&R) flows, hundreds of placement-related parameters influence the design quality. As the technology scales down, the slightest increase in wire and pin capacitance leads to significant degradation in power and performance. This change is observed even as we scale from 28 nm to 16 nm technology node as demonstrated in Reference [13]. This trend continues as the technology node scales further down. The placement parameters are crucial in affecting these capacitance values and, thereby, the number and power of timing buffers required to optimize the design. An unoptimized parameter setting can lead to significant degradation in wirelength, timing, and power.

Therefore, it is necessary to build automated approaches into the physical design **electronic design automation (EDA)** tools to identify the best parameter settings for a given circuit. EDA tools have started integrating reinforcement learning models into them to improve the quality of 2D designs. We need similar approaches to build better 3D physical design tools as well.

2 BACKGROUND AND MOTIVATION

3D P&R flows involving pseudo-3D [15] or true-3D [8, 10, 11] contain several hundred parameters in them. Each parameter affects the P&R results to a different extent. To understand the importance of parameter tuning, we chose a set of 13 placement-related parameters in Compact-2D [9] + Pin-3D [16] flow and studied their impact on 28 nm rocketcore, a RISC-V processor benchmark. We manually set the chosen parameters to different values and performed seven 3D P&R runs on rocketcore. Table 1 shows the design metrics obtained using different parameter settings. We observed that their **Power-Delay Product (PDP)** varies by 35%, approximately.

The placement parameters that work for one circuit do not always work for other circuits. Among the million parameter combinations, it is hard to pick the right set of parameters that work best with the circuit being designed. Manually tuning each parameter and identifying the best parameters is an impossible task; rather, an automated approach to finding suitable parameters can make the search easier. RL-based approaches [1, 2, 21, 24] have often been used to automate

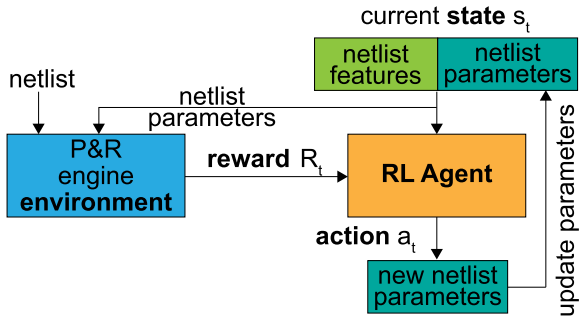

Fig. 1. Reinforcement learning framework used for placement parameter autotuning (adopted from Reference [1]).

parameter tuning processes and create a trained database. This trained database can then be used to pick the best parameters for any given scenario.

However, the trained database is as good as the training dataset.

In the case of 3D circuit designs, the training dataset includes a set of netlists and corresponding **quality of results (QoR)** metrics. The QoR metrics involve the **power, performance, and area (PPA)** of the 3D designs. The quality of this dataset depends on how accurate these metrics are compared to the final design metrics.

For example, the QoR metrics obtained at the end of the placement stage may differ entirely from those obtained after the routing stage, which may vary from the final design signoff QoR. Further, in pseudo-3D approaches, the QoR before and after partitioning the design may be entirely different. Therefore, it is essential to understand the quality of the datasets obtained at different stages of the P&R process and how they affect the quality of the trained RL model.

In this work, we pick two 3D P&R flows, one involving a state-of-the-art pseudo-3D placer and the other with a state-of-the-art true-3D placer, and perform the following analyses:

- We study the effect of different P&R stages on the quality of the parameter autotuning process.
- We analyze the tradeoff between the training runtime and the quality of the final result obtained.
- We compare the behavior of RL-based parameter tuning on pseudo-3D and true-3D placer-based 3D P&R flows.

3 RL TUNING ENVIRONMENT

3.1 Overview of the Framework

Several researches [22, 23] have been carried out on parameter autotuning in EDA tools using machine learning approaches. Agnesina et al. propose a state-of-the-art deep RL-based VLSI placement parameter optimization methodology involving actor-critic framework [21, 24] in their recent work [1]. The proposed method focuses on improving the wirelength of a 2D commercial placement engine. It offers up to 11% wirelength improvement in unseen 2D netlists over manual parameter tuning and around 2.5% wirelength improvement over state-of-the-art autotuners involving MAB. We modify the tool environment and the reward system in the RL framework described in Reference [1] for our study as follows:

The key idea of our RL framework based on Reference [1] is shown in Figure 1. The RL agent in our framework solves the problem of identifying optimal placement parameter settings using the following four key RL elements:

Table 2. Our 20 Handcrafted Netlist Features

Metadata (10)		Topological (10)	
Name	**Type**	**Name**	**Type**
# cells	integer	average degree	float
# nets	integer	average fanout	float
# cell pins	integer	largest strongly connected component	integer
# IO	integer	max. clique	integer
# nets w. fanout $\in]5, 10[$	integer	chromatic nb.	integer
# nets w. fanout ≥ 10	integer	max. logic level	integer
#Sequential elements	integer	$\frac{\text{High degree coefficient}}{\text{clustering coefficient}}$	float
total cell area (μm^2)	integer		float
# hardmacros	integer	Fiedler value	float
macro area (μm^2)	integer	spectral radius	float

State (s): For any given netlist, a set of placement parameters forms a state. The entire state space consists of 10 different netlists and all possible parameter combinations ($\boldsymbol{\rho}$).

Tool environment (T): The environment is the complete P&R tool flow methodology performing the entire 3D design. Depending on the depth of parameter tuning, the P&R process is stopped at a certain stage, and a reward is obtained. It receives the current state (parameter settings and netlist) as the input and performs pseudo/true 3D P&R.

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.

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 $\mathbf{R}$ is maximized, where $\boldsymbol{\rho}$ is the set of all parameter combinations.

We define our state as the joint values of different placement parameters from the commercial P&R tool and information metrics on the netlist being placed. The netlist information consists of a mixture of metadata knowledge (e.g., number of cells, floorplan area) with topological graph features (Table 2) and unsupervised features extracted using a graph neural network. These are adapted from Reference [1]. Netlist characteristics are essential to transfer knowledge across very different netlists so our agent generalizes its tuning process to unseen netlists. We represent our states using one-hot encoded categorical (Booleans or enumerates), integer, and floating point parameters along with integer and floating point netlist features.

We carefully selected the placement parameters among the 60 available ones in the software. We pruned the ones not relevant to our study; we do not, for example, consider structured data paths, fillers, scan chains, shifters, and IR drops.

3.2 Our Actions

We choose deterministic actions to change the setting of a subset of parameters as suggested in Reference [1]. They render the state Markovian, i.e., given state-action pair (s_t, a_t), the resulting state s_{t+1} is unique. An advantage of fully observed determinism is that it allows *planning*. Starting from state s_0 and following a satisfactory policy π, the trajectory

$$s_0 \xrightarrow{\pi(s_0)} s_1 \xrightarrow{\pi(s_1)} \cdots \xrightarrow{\pi(s_{n-1})} s_n \tag{1}$$

leads to a parameter set s_n of good quality. If π has been learned, then s_n can be computed directly in $O(1)$ time without performing any placement.

Table 3. Our 15 Actions

1. FLIP Boolean parameters
2. UP Integer parameters
3. DOWN Integer parameters
4. UP Efforts
5. DOWN Efforts
6. UP Detailed placement parameters
7. DOWN Detailed placement parameters
8. UP Global placement parameters (does not touch the bool)
9. DOWN Global placement parameters (does not touch the bool)
10. INVERT-MIX timing vs. congestion vs. WL efforts
11. UP Partitioner parameters
12. DOWN Partitioner parameters
13. UP True-3D placer parameters
14. DOWN True-3D placer parameters
15. DO NOTHING

11 actions are common to pseudo-3D and true-3D placer. Actions 11 and 12 are specific to pseudo-3D flow. Actions 13 and 14 are specific to true-3D placer.

Based on the suggestions in Reference [1], we create 15 different actions $\mathcal{A}$, as presented in Table 3. Our action space is designed to be as simple as possible to help neural network training but also expressive enough so such transformations can reach any parameter settings.

3.3 Our Reward Structure

To learn with a single RL agent across various netlists with different QoR, we adopt a normalized reward function that renders the magnitude of the value approximations similar among netlists of the form

$$R_t := \frac{\alpha \times \mathrm{WL}_{\text{Human Baseline}} - \mathrm{WL}_t}{\mathrm{WL}_{\text{Human Baseline}}} + \frac{\beta \times (|\mathrm{WNS}|_{\text{Human Baseline}} - |\mathrm{WNS}|_t)}{|\mathrm{WNS}|_{\text{Human Baseline}}} + \frac{\gamma \times (\mathrm{P}_{\text{Human Baseline}} - \mathrm{P}_t)}{\mathrm{P}_{\text{Human Baseline}}}, \tag{2}$$

where WL is the wirelength (μm), WNS is the design's worst **negative slack (ns)**, and P are 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 normalize each result with respect to a human baseline value. The human baseline value refers to the wirelength, WNS, and power values obtained using the default/manual (without RL tuning) P&R run. As the wirelength, WNS, and power have different magnitudes, normalizing them using a human baseline value makes the corresponding rewards comparable and makes it easier to add the rewards together to generate an overall reward value. In other words, the reward function helps obtain the overall percentage improvement brought about by the parameter tuning iteration. Therefore, the greater the percentage improvement, the greater the reward of the corresponding parameters. Even though defining rewards in this manner necessitates knowing human baseline values, it only requires one P&R flow to be completed by the designer.

The choice of the parameters: α, β, and γ dictates which QoR metric is the most optimized. In this work, we set α to 1, and β & γ to 2. Through experimentation, we observed that the best wirelength does not always offer the best frequency and power. Therefore, our reward system offers lower significance to the wirelength improvement and slightly higher but equal importance

Table 4. Netlists Used in This Work

Netlist	#Cells	#Nets	#Hardmacros	Area (mm^2)	Default runtime (hr)	Type
aes	114k	112k	0	0.18	4.1	gate-dominant
b19	33k	34k	0	0.04	2.2	net-dominant
des	47k	48k	0	0.06	1.8	net-dominant
ecg	97k	97k	0	0.11	2	neutral
enc_dec	76k	81k	0	0.11	8.7	net-dominant
fpu	44k	43k	0	0.06	3.5	gate-dominant
ldpc	41k	43k	0	0.05	1.9	net-dominant
rocketcore	95k	94k	6	0.26	2.9	gate-dominant/processor
tate	154k	154k	0	0.19	5.8	neutral
vga	33k	33k	0	0.06	4.2	neutral

Default runtime is calculated based on default Compact-2D [9] + Pin-3D [16] run.

to WNS and power improvements. The effects of other combinations of these parameters are still open for experimentation and are beyond the scope of this work.

WNS values closer to 0 indicate more scope for frequency improvement. Therefore, for the tuning process to perform better and offer significant frequency improvements, we choose design frequencies such that the human baseline designs have a significant WNS, irrespective of the design size. This helps the tuning process to optimize the placement parameters for achieving better design frequencies.

If the designs have power-limitation, then the frequency cannot be increased beyond a certain stage, and in this case the WNS values would be small or even 0 in some cases. In such instances, the scope of improvement is limited to the wirelength and power values, and hence the parameter β can be set to 0 in the reward function shown in Equation (2). However, in this work, we do not consider designs that are power-limited.

3.4 Technology Details

The 3D designs in this work are performed at a 28 nm technology node and involve 2-tiers. The 2D technology files, such as **library exchange format (LEF)**, **liberty (LIB)**, and parasitics files, are extended to create 3D technology files. The 3D design uses a 12-metal layer routing stack, with 6 metal layers (M1–M6) on each tier. The two tiers are integrated in a **face-to-back (F2B)** fashion using 100 nm MIVs. The MIVs connect the M6 layer of the bottom tier with the M1 layer of the top tier through the **front end of the line (FEOL)** of the top tier. The 3D technology LEF file reflects this 3D routing stack. Therefore, our 3D designs consider the area and parasitics overhead introduced by the inter-tier vias while performing area, timing, and power optimization.

3.5 Training Setup

We train the P&R flows and create trained models using 10 different netlists, shown in Table 4. These include a mix of gate-dominant, net-dominant, and processor benchmarks to expose the RL framework to different kinds of netlists.

The main issues with integrating RL in EDA are the latency of tool runs (it takes minutes to hours to perform one P&R) and the sparsity of data (there is no database of millions of netlists placed designs or layouts). We implement a parallel version of the actor-critic framework to solve both issues. In this implementation, an agent learns from the experiences of multiple *Actors* interacting in parallel with their copy of the environment. This configuration increases the throughput of acting and learning and helps de-correlate samples during training for data efficiency [5].

The learning updates may be applied synchronously or asynchronously in parallel training setups. We use a synchronous version, i.e., a deterministic implementation that waits for each

Actor to finish its segment of experience (according to the current policy provided by the step model) before performing a single batch update to the network weights. One advantage is that it allows for larger batch sizes, which are more effectively used by computing resources. The parallel training setup does not modify the equations presented before. Instead, the gradients are just accumulated among all the environments' batches.

Each training iteration involves 10 parallel runs. This helps the RL agent collect sufficient data from different netlists and parameter configurations to make a better parameter selection in future iterations. Each run does not involve the entire P&R flow. Different stages of the P&R flow are used during the training process, depending on the 3D flow being trained. The number of training iterations also varies based on the 3D flow, depending on their runtime. These flow-specific details are explained in the subsequent sections.

4 TUNING PSEUDO-3D PLACEMENT PARAMETERS

The well-known pseudo-3D flows [15] use commercial 2D P&R tools to perform 3D IC designs. These flows transform 2D designs built using 2D P&R tools into 3D designs. The quality of the 2D placement obtained from these tools plays a significant role in determining the **quality of the result (QoR)** of the final 3D designs. Commercial tools have numerous parameters that affect the placement quality. The placement parameters that work well during the 2D stage of the pseudo-3D flows may not produce the best results after 3D transformation. Therefore, tuning the placement parameters of 2D tools for 3D designs is essential to achieving high-quality 3D IC designs.

Among the several pseudo-3D flows [15], we choose the state-of-the-art, compact-2D [9] + pin-3D [16] to understand the benefits of RL-based parameter autotuning on pseudo-3D flows. In ART-3D [13], we compared the state-of-the-art pseudo-3D flows, such as shrunk-2D [14], snap-3D [20], and compact-2D [9] + pin-3D [16], and observed that compact-2D + pin-3D performs better on most circuits and offers the most performance benefits among all pseudo-3D flows. Hence, we use the best-performing pseudo-3D flow to improve it further using RL tuning methodology proposed in this work. As all the other pseudo-3D flows also use a similar commercial tool environment, the results shown in this work are applicable to other pseudo-3D flows as well. The following section describes the Compact-2D + Pin-3D pseudo-3D flow.

4.1 Compact-2D + Pin-3D 3D P&R Flow

In compact2D, a 2D design is first implemented using a commercial 2D P&R tool. The 2D design is performed using a Verilog netlist, 2D standard cells & technology files, and timing constraints, as used in a typical 2D IC design flow. But the RC parasitics of the design interconnects are scaled by a factor of $\sqrt{2}$, as it would be in the 3D design. The 2D design has twice the area of the final 3D design intended. Using the modified interconnect parasitics, placement, post-placement optimization (preCTS), **clock tree synthesis (CTS)**, clock optimization, and signal routing are performed. Then the routed design is partitioned into multiple tiers using **Fiduccia–Mattheyses (FM)** [6] min-cut algorithm. In this stage, the 2D design is transformed into a 3D design and hence the name pseudo-3D. During transformation, the 2D design is contracted to the desired footprint for the 3D IC.

After partitioning, the 3D design is optimized using the Pin-3D router & optimizer flow. The whole 3D design is loaded into the commercial tool. Except for the tier being optimized, other tiers are made transparent to avoid placement density issues from loading a whole 3D design onto a 2D tool. Initially, a refine placement is performed to remove any overlaps after partitioning and the placement is legalized. Then the design is routed and post-routing optimization is performed tier-by-tier. Even though the design is optimized tier-by-tier, the tool is aware of both 2D

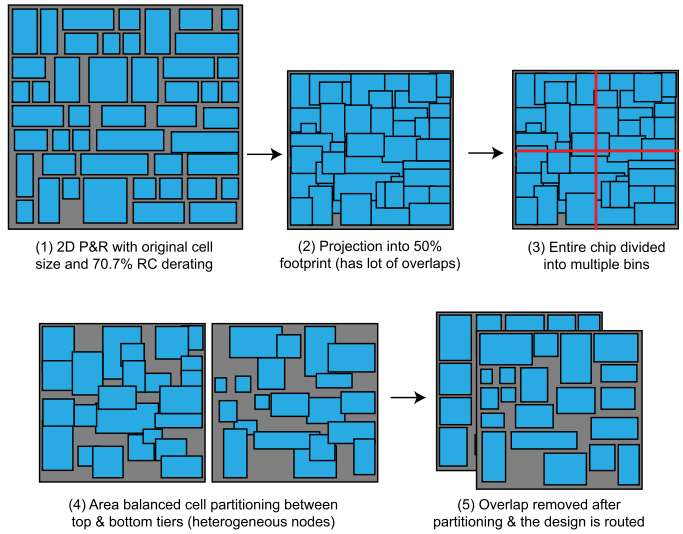

Fig. 2. Illustration of Compact-2D [9] + Pin-3D [16] flow.

Table 5. Commercial Tool Placement Parameters Tuned in Compact-2D/Pin-3D P&R Flow

Parameter	Description	Type	Value
Legalization inst gap	Max. legalization instance gap (μm)	Integer	[0,10]
Global max density	Max. density in any part of the design	Integer	[60, 90]
ECO max distance	Max. ECO instance distance (μm)	Integer	[0, 10]
Wirelength effort	Effort to optimize total wirelength	Enum	[none, medium, high]
Congestion effort	Effort to reduce design congestion	Enum	[low, medium, high]
Timing effort	Effort to optimize overall design timing	Enum	[low, medium, high]
Clock power effort	Effort to optimize the clock power	Enum	[low, medium, high]
Power effort	Effort to optimize overall design power	Enum	[low, medium, high]
Uniform density	Enable even cell distribution	Boolean	[True, False]
Clock gate aware placement	Enable clock gating aware placement	Boolean	[True, False]

The solution space (ρ) is 3.6×10^6.

and 3D interconnects in the design as the whole 3D design is loaded. The Compact-2D + Pin-3D is illustrated in Figure 2.

4.2 Placement Parameters Tuned

The commercial 2D P&R tool and the FM min-cut algorithm have numerous parameters that affect the QoR of the 3D design. Among those, we handpicked a set of 10 parameters in the P&R tool environment and 3 parameters in the partitioner that affect the placement quality and thereby the QoR of the 3D design. The placement parameters tuned in the commercial tool and the permissible values for them are shown in Table 5. Among the 10 parameters, 3 are integers, 2 are Boolean, and 5 are enumeration parameters, leading to possible combinations of 3.6×10^6. The parameters tuned in the partitioner and their permissible value ranges are shown in Table 6. The overall solution space combining all the 13 parameters is 4.96×10^{10}. The parameters and the value ranges are picked based on the suggestions from the authors of Compact-2D [9] & Pin-3D [16] flow. We tune these parameters using the RL agent described in Section 3 through different approaches, as discussed below.

Table 6. FM Min-cut Partitioner Parameters Tuned in Compact-2D/Pin-3D P&R Flow

Parameter	Description	Type	Value
Bin size	Partitioning bin dimension(μm)	Integer	[10, 50]
Target bin density	Maximum bin density allowed in %	Integer	[75, 95]
Target bin overlap	Maximum cell overlap (%) allowed in each bin	Integer	[10, 25]

The solution space (ρ) is 13,776.

4.3 RL-based Parameter Autotuning

While performing RL-based parameter autotuning on this flow, the quality of the reward R (shown in Equation (2)) is heavily influenced by how deep the tuning process extends into the flow. We modify the P&R engine environment in the RL framework (see Figure 1) to include different stages of compact-2D + pin-3D flow.

We train three different RL models with the tool environment in the RL framework extending through various stages of the P&R flow. The three models presented in this work are as follows:

- Model1: The P&R engine environment includes the place and placement optimization (preCTS) stages of Compact-2D flow. The rest of the Compact-2D and Pin-3D flows are not included. The model is trained based on the reward obtained at the end of placement optimization. The tool environment in this model uses a synthesized netlist as the input.
- Model2: Model 1 + CTS stages of Compact-2D flow. The model is trained based on the reward obtained at the end of CTS. The tool environment in this model uses a synthesized netlist as the input.
- Model3: The P&R engine environment includes the FM min-cut partitioner, placement legalization, and global routing stages of Pin-3D flow. The model is trained based on the reward obtained at the end of Pin-3D's global routing. The tool environment in this model uses the output of default Compact-2D flow as the input. By default Compact-2D, we refer to the Compact-2D flow using default manual parameter settings, as suggested by the creators of the flow.

We also performed two other experiments, one where the training process was stopped right after compact-2D placement and the other where the parameter autotuning involved the FM partitioner and the pin-3D legalizer. However, the observed results were worse than the default P&R flow; therefore, we exclude those results from this article. The modified RL-based training flows are shown in Figure 3. We train these models using the 10 netlists described in Section 3.5 and perform 100 training iterations. Each iteration performs 10 runs on different netlists and parameter settings in parallel.

4.4 Training Results

After training the three different RL models on 10 netlists for 100 iterations, we picked the parameters that provided the best rewards for each netlist. We run the rest of the P&R flow using these parameters on each circuit. The results of these runs are shown in Table 7.

Among the three models, model 2 and model 3 show better results on most circuits over the default flow. RL model 1, in which the training is performed based on placement optimization results, offers a slight PDP improvement of <2% on aes, b19, ecg, and tate over the default flow. In the case of rocketcore, enc_dec, and vga, the results are worse than the default flow. RL model 1 offers a significant improvement of 5%–10% on des & fpu and 17.5% on ldpc. On training for 247 hours, RL model 1 offers a significant improvement on only two circuits.

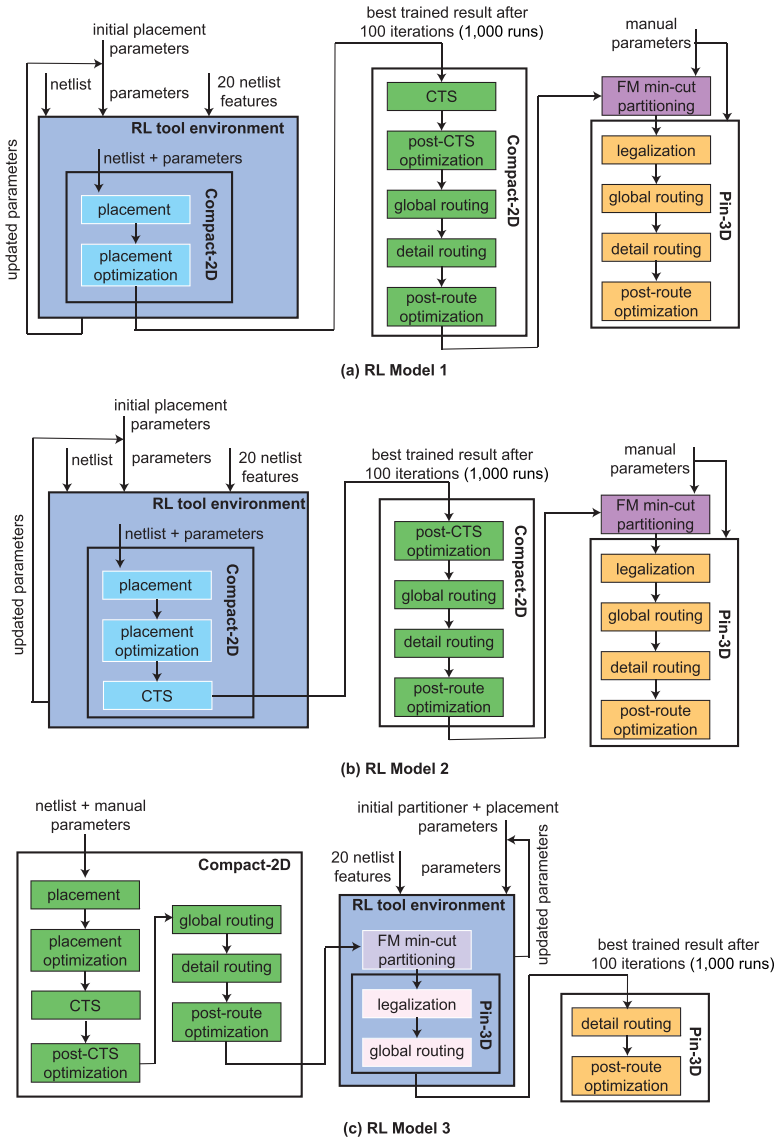

Fig. 3. Different tool environments used in RL-based parameter tuning of Compact-2D + Pin-3D flow.

RL model 2, trained for 290 hours using the results obtained from the CTS stage, offers a slight PDP improvement of <5% on aes, b19, des, and fpu. It offers a PDP improvement of 5%–10% on ecg, tate, rocketcore, and enc_dec. It offers a significant improvement of >10% on ldpc and vga.

However, RL model 3, trained for just 126 hours, offers a significant PDP improvement of 11%–23% on ecg, ldpc, rocketcore, and vga. It also offers a PDP improvement of 5%–10% on aes, des, fpu, and enc_dec, and <5% improvement on b19 and tate.

On comparing the training results of the three RL models, RL model 3 offers a whopping improvement of up to 23% in a training time of almost 50% of that of the other two models. Table 8 compares the default parameters and the parameters chosen by the three RL models on

Table 7. QoR Comparison of Default vs. the Three RL-trained Models of Compact-2D [9] + Pin-3D [16] Flow

Metrics	aes	b19	des	ecg	fpu	ldpc	tate	rocketcore	enc_dec	vga	Runtime
DEFAULT											
WL (m)	1.29	0.36	0.54	0.91	0.57	1.43	1.75	1.60	6.51	1.26	
WNS (ps)	−94	−52	−61	−82	−180	−67	−133	−175	−81	−127	
Freq. (GHz)	3.40	1.17	2.17	1.72	1.47	1.36	2.61	1.03	1.13	3.05	N/A
Power (mW)	455.5	38.5	167.4	158.1	102.7	181.2	769.3	184	330.9	142.6	
PDP (pJ)	133.9	32.8	77.2	92.0	69.8	133.2	294.6	179.4	291.5	46.6	
RL MODEL 1											
WL (m)	1.35	0.36	0.56	0.90	0.58	1.26	1.80	2.04	6.91	1.31	
WNS (ps)	−92	−41	−42	−75	−141	−101	−136	−243	−188	−131	
Freq. (GHz)	3.42	1.19	2.26	1.74	1.56	1.30	2.59	0.96	1.01	3.02	247 hrs
Power (mW)	453.1	38.7	163.6	158.6	100.7	143.1	757.4	183.6	344.8	145.6	
PDP (pJ)	132.3	32.5	72.1	91.2	64.5	109.9	292.3	191.5	340.7	48.2	
Δ_{PDP}	−1.2%	−0.9%	−6.6%	−0.9%	−7.5%	−17.5%	−0.8%	+6.7%	+16.8%	+3.4%	
RL MODEL 2											
WL (m)	1.31	0.35	0.54	0.88	0.56	1.27	1.70	1.89	6.07	1.05	
WNS (ps)	−95	−38	−52	−40	−166	−86	−110	−169	−96	−124	
Freq. (GHz)	3.39	1.19	2.21	1.85	1.50	1.33	2.77	1.03	1.12	3.08	290 hrs
Power (mW)	454	38.3	163.3	154.2	101.2	143.8	761.7	173.9	305.1	128.6	
PDP (pJ)	133.93	32.1	73.8	83.3	67.4	108.3	276.9	168.5	273.4	41.7	
Δ_{PDP}	0%	−2.1%	−4.4%	−9.5%	−3.4%	−18.7%	−6%	−6.1%	−6.2%	−10.5%	
RL MODEL 3											
WL (m)	1.25	0.35	0.54	0.89	0.57	1.42	1.94	1.56	6.51	1.04	
WNS (ps)	−75	−30	−0.40	−30	−122	−52	−111	−86	−65	−114	
Freq. (GHz)	3.68	1.20	2.27	1.89	1.61	1.39	2.77	1.13	1.16	3.18	126 hrs
Power (mW)	440.7	37.9	163.1	152.8	102.6	141.4	776.6	166.5	305.7	127.1	
PDP (pJ)	121.2	31.5	71.8	81.0	63.8	101.6	280.4	147.5	264.4	39.9	
Δ_{PDP}	−9.5%	−4%	−7.0%	−11.9%	−8.5%	−23.7%	−4.9%	−17.8%	−9.3%	−14.4%	

The netlists in this table were used for training the models. The models were trained for 100 iterations, with each iteration consisting of 10 custom P&R runs, depending on the RL model. This table refers to the baseline, manually tuned flow as DEFAULT. $\Delta_{PDP} = \frac{PDP_x - PDP_{DEFAULT}}{PDP_{DEFAULT}} \times 100$. Test results of model 3 are shown in Table 12.

rocketcore. The parameter settings chosen by different RL models are significantly different. It is also observed from Table 8 that RL models 1 and 2 share 7 out of 13 parameters with the default run. The parameters: bin size, target bin density, and overlap remain the same, as they are related to the partitioner, and we do not tune them in RL models 1 and 2. The parameters legalization instance gap, power effort, uniform density, and clock-aware placement do not have much impact on the pre-partitioning stages of the rocketcore circuit. Therefore, the pre-partitioning models (RL models 1 and 2) have the same parameter values as the default flow.

We also performed another experiment by extending the RL tuning process to the post-CTS stage. However, the PDP improvement observed was less than 2% for a 25.9% runtime increase.

Based on the training results, it is evident that for pseudo-3D flows, RL tuning after the pseudo-3D stage, i.e., during and after partitioning, is highly effective than RL tuning before partitioning. If training runtimes can be relaxed further, then combining pre- and post-partition stages during the training can offer even more significant improvements.

Using the parameter settings obtained from RL Model 2 during the compact-2D stage and the settings from Model 3 during the partitioning and pin-3D stages, we observed a 20.4% improvement in the PDP of rocketcore circuit instead of 6.1% and 17.8% improvement obtained using Model 2 and 3, individually.

Table 8. Parameters Used in the Default Flow vs. Parameters Chosen by the Three RL-based Flows of Compact-2D [9] + Pin-3D [16] Flow on Rocketcore

rocketcore				
Parameters	Default	RL 1	RL 2	RL 3
Bin size	41	41	41	28
Target bin density	95	95	95	88
Target bin overlap	20	20	20	21
Legalization inst gap	0	0	0	2
Global max density	N/A	90	88	90
ECO max distance	0	20	0	0
Wirelength effort	medium	low	medium	high
Congestion effort	auto	low	medium	high
Timing effort	medium	high	medium	high
Clock power effort	low	low	medium	high
Power effort	low	low	low	high
Uniform density	false	false	false	false
Clock gate aware placement	true	true	true	false

Except for the bin size, every parameter setting is the same in the default flow on all netlists. The bin size is calculated as $\frac{\text{max\_chip\_dimension}}{10}$ in the default flow.

5 TUNING TRUE-3D PLACEMENT PARAMETERS

There are several state-of-the-art academic 3D placers, such as Force-3D [8], ePlace-3D [10], and **Non-Linear 3D (NL-3D)** [11]. These placers, unlike pseudo-3D flows, do not pose 3D placement as a 2D placement problem. They rather perform 3D placement in a true 3D space, and hence we call them true-3D placers. Similar to the pseudo-3D flows, the P&R flows involving true-3D placers also contain several placement parameters that affect the QoR of the final design. We chose NL-3D [11] placer to analyze the performance of RL tuning on true-3D placer. Among the true-3D flows, we were able to attain access to the source codes of NL-3D [11] and ePlace-3D [10] flows. While both codes offered similar performance improvements, we picked NL-3D for its easier hard macro integration.

The following section describes the NL-3D placer-based 3D flow.

5.1 NL-3D Placer-based 3D P&R Flow

5.1.1 Non-Linear 3D Placer. NL-3D placer [11] 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:

$$\min \text{OBJ}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{e \in E} (1 + \gamma_e)(\text{WL}(e) + \alpha_{MIV} \cdot \text{MIV}(e)), \tag{3}$$

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 $\text{WL}(e)$ and $\text{MIV}(e)$ 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 **non-linear programming (NLP)** solver, as described in Reference [11]. 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.

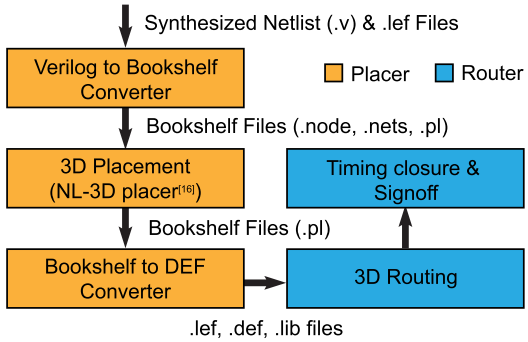

Fig. 4. 3D P&R flow involving true-3D placement.

5.1.2 Overview of 3D Flow. The overall 3D flow involving true-3D placement is shown in Figure 4. The generated placement results are processed and imported into commercial routing tools like Cadence Innovus to perform clock routing, signal routing, timing, and power optimizations.

5.1.3 RTL Adoption and File Format Conversion. The academic placement engines are designed to work on netlists based on bookshelf [3] format. However, standard netlists are synthesized using the Verilog format. So, we use a script to convert Verilog netlists to bookshelf format. First, the Verilog netlist is converted to **design exchange format (DEF)**. The size of standard cells in the netlist are contained in the **library exchange format (LEF)** files of the corresponding PDK used. Both DEF and LEF files are used to translate a synthesized netlist from Verilog to bookshelf format. Once the bookshelf files are generated, we use one of the placer algorithms to generate 3D placement.

5.1.4 Modification Needed for True-3D Placers. The NL-3D algorithm is several years old and is predominantly based on **Through Silicon Vias (TSVs)**. As the 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.

5.1.5 Back-end Design with Commercial Tools. Academic 3D placers, such as Force-3D [8], ePlace-3D [10], and **Non-Linear 3D (NL-3D)** [11], compare their wirelength estimation and PPA values after 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, such as Shrunk2D [14], Compact2D [9], and others [15], are capable of performing the entire P&R flow, thereby providing a standard means to analyze the quality of 3D ICs designed using them. Therefore, after performing NL-3D placement, we perform the rest of the physical design using a commercial P&R tool to create a full-fledged 3D flow. We import the 3D placement obtained from the NL-3D placer into the commercial tool and perform the following stages.

5.1.6 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 a 3D design. However, placing all the 3D 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

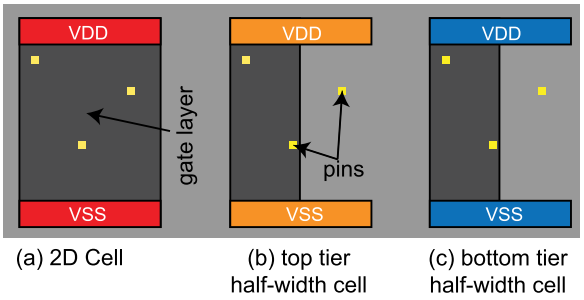

Fig. 5. Half width standard cells used in our 3D flow. Only the width of the gate layer is halved, but the pins are retained at the original location.

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 Figure 5.

In the case of hardmacro 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 the 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 [20]. 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.

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

5.1.8 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 the Pin-3D router and optimizer [16], which is based on a commercial P&R tool to perform global and detailed routing and timing closure on the entire 3D design. During the RL training process, we stop each iteration at the global routing stage. We perform detailed routing using the 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

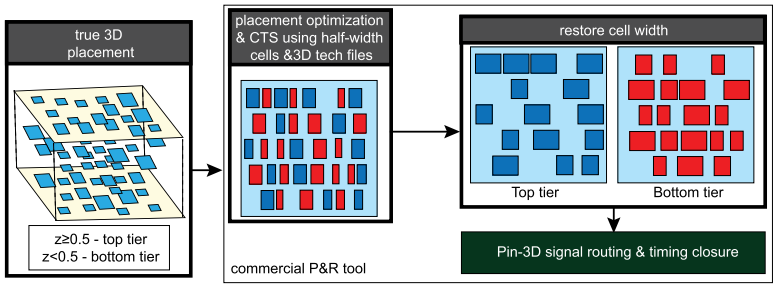

Fig. 6. Pictorial representation of 3D P&R flow with true-3D placement.

Table 9. Parameters Tuned in Non-linear 3D Placer [11]

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	Maximum cell overlap % 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	Enable routing congestion driven placement	Boolean	[True, False]
Dummy Cells	Add dummy blocks to prevent routing congestion	Boolean	[True, False]

The solution space ($\boldsymbol{\rho}$) is infinite.

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 parasitic, 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.

The overall 3D P&R flow is pictorially shown in Figure 6.

5.2 Placement Parameters Tuned

NL-3D placer has around 11 critical parameters, which affect the wirelength of the 3D design. These parameters are tabulated in Table 9. Among the 11 parameters, 2 are integers, 2 are Boolean, and 6 are floating point parameters. The presence of the floating point parameters makes the solution space infinite. In addition to the NL-3D placement parameters, we also tune the 10 commercial tool placement parameters (shown in Table 5) similar to the Compact-2D + Pin-3D flow. These parameters are used during the placement optimization stage in the back-end design flow.

The NL-3D parameters and the value ranges in Table 9 are picked based on the suggestions from the authors of NL-3D [11] flow. We tune these parameters using the RL agent described in Section 3 through different approaches, as discussed below. For the default NL-3D flow, we manually tune and set these parameters.

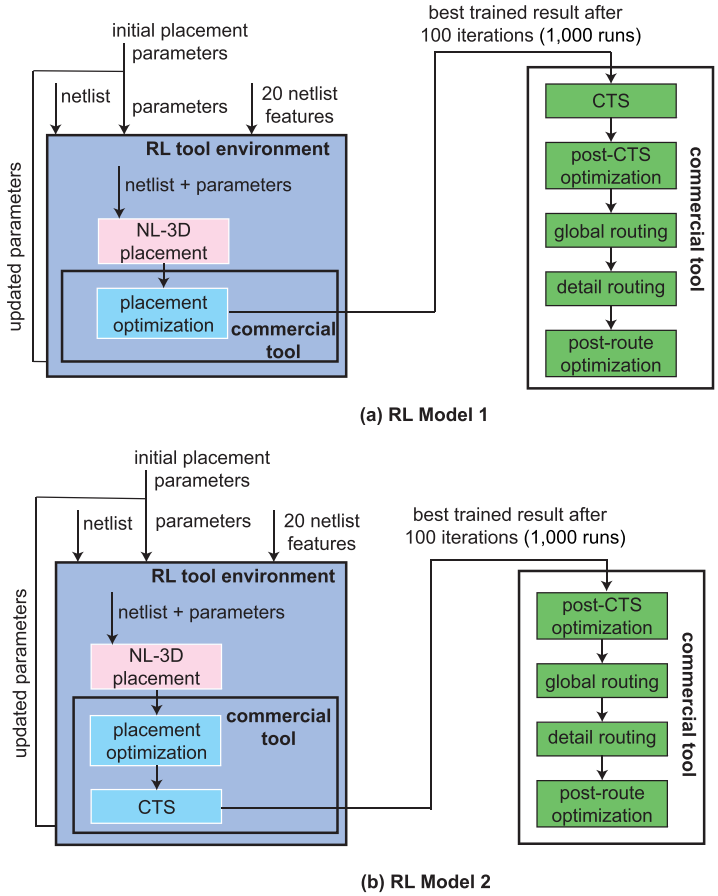

Fig. 7. Different tool environments used in RL-based parameter tuning of NL-3D based 3D P&R flow.

5.3 RL-based Parameter Autotuning

Similar to the training performed in the pseudo-3D flow, we modify the P&R engine environment in the RL framework to include different stages of NL-3D flow. We train two different RL models, as listed below:

- Model1: The P&R engine environment includes the NL-3D placement and back-end placement optimization (preCTS) stage. The rest of the NL-3D flow is not included. The model is trained based on the reward obtained at the end of placement optimization. The tool environment in this model uses a synthesized netlist as the input.
- Model2: Model 1 + CTS stages. The model is trained based on the reward obtained at the end of the CTS stage. The tool environment in this model uses a synthesized netlist as the input.

The modified RL-based training flows are shown in Figure 7.

5.4 Training Results

After training the two different RL models on 10 netlists for 100 iterations, we picked the parameters that provided the best rewards for each netlist. We run the rest of the P&R flow using these

Table 10. QoR Comparison of Default vs. the Two RL-trained Models of NL-3D [11]-based 3D P&R Flow

Metrics	aes	b19	des	ecg	fpu	ldpc	tate	rocketcore	enc_dec	vga	Runtime
DEFAULT											
WL (m)	1.36	0.35	0.54	1.10	0.61	1.46	1.92	1.82	6.65	0.90	
WNS (ps)	−115	−173	−60	−100	−213	−93	−197	−200	−94	−262	
Freq. (GHz)	3.17	1.03	2.17	1.67	1.40	1.32	2.24	1.00	1.12	2.16	N/A
Power (mW)	385.9	38.8	167.5	160.1	105.4	208.2	774.5	198.9	328.7	116.5	
PDP (pJ)	121.6	37.8	77.1	96.1	75.2	158.2	346.2	198.9	293.9	53.8	
RL MODEL 1											
WL (m)	1.24	0.33	0.52	1.14	0.60	1.72	1.90	1.71	6.82	0.92	
WNS (ps)	−64	−397	−36	−74	−198	−141	−179	−100	−113	−255	
Freq. (GHz)	3.79	0.84	2.29	1.74	1.43	1.24	2.33	1.11	1.10	2.20	305 hrs
Power (mW)	401.6	38.28	169.3	161.4	103.8	231.7	781.2	188.7	335.4	115.5	
PDP (pJ)	106	45.8	73.8	92.6	72.2	187.1	335.1	169.8	306.2	52.6	
Δ_{PDP}	−12.8%	+21.2%	−4.3%	−3.6%	−4.0%	+18.3%	−3.2%	−14.6%	+4.0%	−2.2%	
RL MODEL 2											
WL (m)	1.24	0.32	0.51	1.15	0.58	1.43	1.91	1.46	6.59	0.89	
WNS (ps)	−75	−166	−20	−34	−191	−142	−192	−72	−79	−220	
Freq. (GHz)	3.63	1.04	2.38	1.87	1.45	1.24	2.26	1.15	1.14	2.38	410 hrs
Power (mW)	398.7	37.1	164.8	162.8	103.5	207.2	780.6	177.6	325.1	116.9	
PDP (pJ)	109.6	35.8	69.2	86.9	71.5	167.6	345.0	154.9	285.7	49.1	
Δ_{PDP}	−9.9%	−5.3%	−10.2%	−9.6%	−4.9%	+5.9%	−0.3%	−22.1%	−3.1%	−8.7%	

The netlists in this table were used for training the models. The models were trained for 100 iterations, with each iteration consisting of 10 custom P&R runs, depending on the RL model. This table refers to the baseline, manually tuned flow as DEFAULT. $\Delta_{PDP} = \frac{PDP_x - PDP_{DEFAULT}}{PDP_{DEFAULT}} \times 100$. Test results of model 2 are shown in Table 12.

parameters on each circuit. The results of some of these runs are shown in Table 10. It is seen that the NL-3D-based flow performs worse than the pseudo-3D flow.

RL model 1 offers a slight 2%–5% PDP improvement on des, ecg, fpu, tate, and vga compared to the default flow. It offers a significant PDP improvement of >10% on aes and rocketcore over the default flow. However, the PDP on b19, enc_dec, and ldpc are much worse than the default flow.

In the case of RL model 2, we observe a <5% PDP improvement on fpu, tate, and enc_dec, a 5%–10% PDP improvement on aes, b19, ecg, and vga, and >10% PDP improvement on des and rocketcore over the default flow. However, ldpc offers a worse PDP than the default flow.

Table 11 compares the default parameters and the parameters chosen by the two RL models on rocketcore. The parameters chosen by RL-Model 1 are on the lower end of the permissible value range. In contrast, those selected by RL-Model 2 are on the higher end, leading to more performance improvement over the other two flows.

In the case of NL-3D placer-based 3D flow, RL training based on CTS data offers better results than the training performed with placement optimization results. However, despite the large training times of 305–410 hours in these models, the results observed are still worse than those seen in the default and RL-trained flows of the pseudo-3D flow. The reason behind this is explained in Section 6.

6 RL FOR PSEUDO-3D VS. TRUE-3D

A striking difference between training Compact-2D + Pin-3D flow and NL-3D based 3D P&R flow is the time required to train the RL models. NL-3D [11], similar to other academic 3D placers, such as ePlace-3D [10] and Force-3D [8], optimizes only the wirelength of the 3D designs. It does not perform any timing or power-driven 3D placement. In many cases, an optimized wirelength does not necessarily mean an optimized design QoR. This is observed in the results presented in this article. The lack of timing and power optimization stage in the 3D placer overloads the commercial

Table 11. Parameters Used in the Default Flow vs. Parameters Chosen by the Two RL-based Flows of NL-3D [11]-based Flow on Rocketcore

rocketcore			
Parameters	Default	RL 1	RL 2
Grids	50	28	79
Clustering Depth	4	1	3
Cluster Ratio	0.25	0.24	0.27
Wire Weight (γ_e)	10.0	1.0	40.6
Via Weight (α_{MIV})	1.0	1.0	4.6
Detail Place Overlap	0.25	0.19	0.22
Target Density	0.75	0.70	0.88
ε	1.0	0.5	2.02
Υ	1.0	0.5	2.02
Congestion Driven	False	True	True
Dummy Cells	False	False	True

tool-based back-end flow to perform these optimizations, increasing the design runtime. For example, 293 out of 410 hours are spent in the placement optimization and CTS stages of RL Model 2. Therefore, the training runtime is also longer compared to the pseudo-3D flow.

Further, the NL-3D placer involves several floating point parameters, unlike the pseudo-3D placer. The RL framework proposed in Reference [1] follows a deterministic approach involving methodical tuning of each parameter. For a given initial state s_0 of netlist features and NL-3D and commercial tool parameters, the actor-critic framework follows an acceptable policy π, leading to a state with a high-quality parameter set s_n. The trajectory of this policy is given in Equation (1), as suggested in [1].

Many samples are required to learn a good policy π and obtain the state with high-quality parameters s_n. According to Sidford et al. [17], to obtain an ϵ-optimal policy with a probability of $1-\delta$, we need a sample size of $O(\frac{|SA|}{(1-\gamma)^3(\epsilon)^2} log\frac{1}{\delta})$. The flow methodology optimized by Agnesina et al. in Reference [1] has both finite solution space and a shorter runtime for each optimization iteration, leading to obtaining the sample size required to find an optimal policy. The actor-critic framework is a more promising technique for parameter autotuning, provided enough data is obtained for training [12]. However, NL-3D-based 3D P&R flow involves infinite solution space and several hours of training iterations.

To handle the infinite solution space, our RL algorithm performs range-based updates to the floating-point parameters. For example, the UP operation on a parameter with a current value in the range of 0.25–0.50 modifies the parameter to be in the range of 0.5–0.75. We perform range-based updates, since small variations in floating-point parameters do not cause major variations in the design metrics. This also helps cover the entire range of floating point parameters.

Further, unlike the pseudo-3D flow, the RL agent has to tune a set of 10 (commercial tool) + 11 (NL-3D placer) = 21 parameters on NL-3D-based P&R flow. This renders the actor-critic framework impractical for autotuning NL-3D, as obtaining an optimum sample takes several days. However, the finite solution space, lesser parameters (10–13), and shorter runtime of different stages in the pseudo-3D flow make it easier for the RL framework to identify an optimized solution.

We also tested the best RL models (RL model 3 of pseudo-3D flow and RL model 2 of NL-3D-based flow) on two unseen netlists: Cortex A7 and A53, which are commercial processor benchmarks (See Table 12). We observe a whopping PDP improvement of 16% and 11% on Cortex A7 and A53, respectively, on using RL model 3 over the default Compact-2D + Pin-3D flow. Whereas, RL model

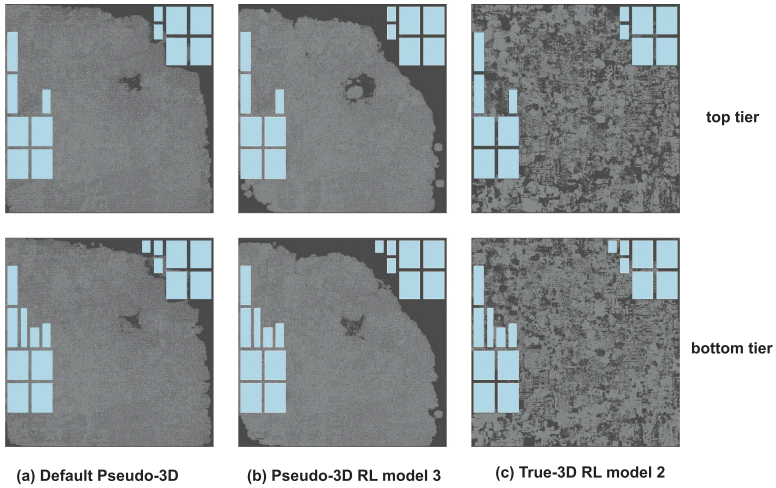

Fig. 8. Cortex A53 final placement layouts.

Table 12. Testing RL Model 3 of Compact-2D + Pin-3D flow (Pseudo-3D) and RL Model 2 of NL-3D-based Flow (True-3D) on **Cortex A7 and A53**

	Default	Pseudo-3D RL Model 3	True-3D RL Model 2
Cortex A7			
WL	1.0	0.97	1.04
WNS	1.0	0.98	1.05
Max. Freq.	1.0	1.02	0.95
Power	1.0	0.86	1.08
PDP	1.0	0.84	1.03
Cortex A53			
WL	1.0	0.90	1.01
WNS	1.0	0.92	0.97
Max. Freq.	1.0	1.08	1.03
Power	1.0	0.97	0.99
PDP	1.0	0.89	0.96

The results are compared with the baseline default Compact-2D + Pin-3D flow and are normalized w.r.t. the default flow.

2 of NL-3D-based flow performs worse than the default Compact-2D + Pin-3D flow on Cortex A7 and offers around 4% improvement on Cortex A53. The final placement snapshots of Cortex A53 designed using these three flows are shown in Figure 8. Therefore, with the state-of-the-art features, pseudo-3D flow shows better optimization with the actor-critic RL framework.

7 EXPERIMENT ANALYSIS

We expose the RL models to different kinds of netlists, such as cell-dominant, net-dominant, and with macros. The netlist sizes are not as big as commercial designs. But our training set involves different netlist characteristics observed in commercial designs. This helps us to accelerate the training while considering different possible scenarios. When we test our models using commercial processors such as ARM Cortex A7 and A53, we observe a significant 11%–16% performance improvement using the trained RL-model 3 pseudo-3D flow. This shows that we can significantly improve large designs by using a wide variety of small/medium-sized netlists.

7.1 Runtime vs. QoR Tradeoff

The training runtimes vary significantly depending on the depth of the tuning process. In the NL-3D-based flow, we see an average PDP improvement of 9.7% between training models 1 and 2. But as the tuning process extends deep into the flow (RL model 1 vs. 2), the training runtime also increases by a significant 34%. Similarly, in the case of the pseudo-3D flow, we observe an average PDP improvement of 7% between RL-Models 1 and 2 for a training runtime increase of 17%. Thus, depending on the designer's needs, a tradeoff is required between the percentage of performance improvement and the training runtime.

8 FUTURE SCOPE

The actor-critic RL framework [2] used in this work performs incremental parameter tuning. We chose this framework, as it performs extensive searches and covers all possible combinations of parameter settings. While it works for pseudo-3D flows, true-3D flows can benefit further using randomized parameter autotuning approaches, such as **multi-armed bandit (MAB)** [18], as shown in ART-3D [13]. However, MAB techniques are often specific to a given netlist and do not facilitate creating a generic trained database of all netlists. Therefore, using contextual MAB techniques, such as MABWiser [19], to perform parameter autotuning of true-3D placer-based flows can be an excellent future scope of our work.

Further, the reward structure (refer to Equation (2)) in our work is based on the insights obtained from our experiments that a better wirelength does not always offer a better PPA. The α, β, and γ values used in this work have led to a minimum variation in the wirelength while offering significant improvements in frequency and power. Experimenting with these parameters further could offer more research insights into parameter autotuning.

9 CONCLUSION

The 3D P&R flows, whether based on true-3D or pseudo-3D placers, are often more complex than the 2D P&R flows, involving a broader range of parameters that affect the QoR. Often, parameter settings that work well during certain stages of the 3D flow may not result in a highly optimized final 3D design. Hence, it is harder to identify efficient parameter settings, either manually or through RL-based training. In this work, we analyzed how the datasets of design metrics obtained from different stages of a 3D P&R flow affect the parameter tuning process. We trained RL models to optimize 3D IC design based on design metrics obtained from different stages of the P&R flow. We analyzed how the depth of the RL tuning process into the P&R flow affects the QoR by extending the training process to different stages of the pseudo-3D and true-3D placer-based flows. We also analyzed the required tradeoff between desired QoR and training runtime for pseudo-3D and true-3D placer-based flows.

On average, the RL-based parameter tuning worked well for most of the circuits designed using Compact-2D + Pin-3D flow. Noticeably, the training performed using the FM min-cut partitioner, the Pin-3D placement legalizer, and global router offered up to 23.7% PDP improvement over the corresponding default flow, with a runtime of almost 50% less than the other two training flows. It also offered a PDP improvement of 16% on the unseen netlist of Cortex A7, an industrial processor benchmark. In NL-3D placer-based 3D P&R flow, the tuning process worked well on a few circuits. But for most circuits, the improvements obtained are significantly less, given the runtime involved in achieving them. The commercial tool-based flow delivers tremendous PPA optimizations with the correct RL tuning strategy. If academic placers involve collective wirelength, timing, and power optimization strategies similar to the commercial tools, then RL tuning can provide significant improvements similar to pseudo-3D flows.

REFERENCES

[1] Anthony Agnesina, Kyungwook Chang, and Sung Kyu Lim. 2020. VLSI placement parameter optimization using deep reinforcement learning. In *Proceedings of the 39th International Conference on Computer-aided Design (ICCAD'20)*. Association for Computing Machinery, New York, NY. DOI: https://doi.org/10.1145/3400302.3415690

[2] Anthony Agnesina, Sai Surya Kiran Pentapati, and Sung Kyu Lim. 2020. A general framework for VLSI tool parameter optimization with deep reinforcement learning. In *Proceedings of the NeurIPS 2020 Workshop on Machine Learning for Systems*. Conference on Neural Information Processing Systems.

[3] Andrew E. Caldwell and Igor L. Markov. 2002. Toward CAD-IP reuse: A web bookshelf of fundamental algorithms. *IEEE Des. Test* 19, 3 (2002), 72–81.

[4] Kyungwook Chang et al. 2016. Cascade2D: A design-aware partitioning approach to monolithic 3D IC with 2D commercial tools. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'16)*. IEEE, New York, NY, 1–8. DOI: https://doi.org/10.1145/2966986.2967013

[5] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. 2018. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. DOI: https://doi.org/10.48550/ARXIV.1802.01561

[6] C. M. Fiduccia and R. M. Mattheyses. 1982. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*. IEEE, New York, NY, 175–181. DOI: https://doi.org/10.1109/DAC.1982.1585498

[7] Meng-Kai Hsu, Yao-Wen Chang, and Valeriy Balabanov. 2011. TSV-aware analytical placement for 3D IC designs. In *Proceedings of the 48th Design Automation Conference (DAC'11)*. Association for Computing Machinery, New York, NY, 664–669. DOI: https://doi.org/10.1145/2024724.2024875

[8] D. H. Kim, K. Athikulwongse, and S. K. Lim. 2013. Study of through-silicon-via impact on the 3-D stacked IC layout. *IEEE Trans. Very Large Scale Integ. Syst.* 21, 5 (2013), 862–874. DOI: https://doi.org/10.1109/TVLSI.2012.2201760

[9] B. Ku, K. Chang, and S. Lim. 2020. Compact-2D: A physical design methodology to build two-tier gate-level 3-D ICs. *IEEE Trans. Comput.-aid Des. Integ. Circ. Syst.* 39, 6 (2020), 1151–1164. DOI: https://doi.org/10.1109/TCAD.2019.2952542

[10] Jingwei Lu, Hao Zhuang, Ilgweon Kang, Pengwen Chen, and Chung-Kuan Cheng. 2016. EPlace-3D: Electrostatics based placement for 3D-ICs. In *Proceedings of the International Symposium on Physical Design (ISPD'16)*. Association for Computing Machinery, New York, NY, 11–18. DOI: https://doi.org/10.1145/2872334.2872361

[11] G. Luo, Y. Shi, and J. Cong. 2013. An analytical placement framework for 3-D ICs and its extension on thermal awareness. *IEEE Trans. Comput.-aid Des. Integ. Circ. Syst.* 32, 4 (2013), 510–523. DOI: https://doi.org/10.1109/TCAD.2012.2232708

[12] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML'16)*. JMLR.org, 1928–1937.

[13] Gauthaman Murali, Sandra Maria Shaji, Anthony Agnesina, Guojie Luo, and Sung Kyu Lim. 2022. ART-3D: Analytical 3D placement with reinforced parameter tuning for monolithic 3D ICs. In *Proceedings of the International Symposium on Physical Design (ISPD'22)*. Association for Computing Machinery, New York, NY, 97–104. DOI: https://doi.org/10.1145/3505170.3506725

[14] S. Panth, K. Samadi, Y. Du, and S. K. Lim. 2017. Shrunk-2-D: A physical design methodology to build commercial-quality monolithic 3-D ICs. *IEEE Trans. Comput.-aid Des. Integ. Circ. Syst.* 36, 10 (2017), 1716–1724. DOI: https://doi.org/10.1109/TCAD.2017.2648839

[15] Heechun Park, Bon Woong Ku, Kyungwook Chang, Da Eun Shim, and Sung Kyu Lim. 2020. Pseudo-3D approaches for commercial-grade RTL-to-GDS tool flow targeting monolithic 3D ICs. In *Proceedings of the International Symposium on Physical Design (ISPD'20)*. Association for Computing Machinery, New York, NY, 47–54. DOI: https://doi.org/10.1145/3372780.3375567

[16] Sai Surya Kiran Pentapati, Kyungwook Chang, Vassilios Gerousis, Rwik Sengupta, and Sung Kyu Lim. 2020. Pin-3D: A physical synthesis and post-layout optimization flow for heterogeneous monolithic 3D ICs. In *Proceedings of the 39th International Conference on Computer-Aided Design (ICCAD'20)*. Association for Computing Machinery, New York, NY. DOI: https://doi.org/10.1145/3400302.3415720

[17] Aaron Sidford, Mengdi Wang, Xian Wu, Lin F. Yang, and Yinyu Ye. 2018. Near-optimal time and sample complexities for solving Markov decision processes with a generative model. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. Curran Associates Inc., Red Hook, NY, 5192–5202.

[18] Apostolos Stefanidis, Dimitrios Mangiras, Chrysostomos Nicopoulos, and Giorgos Dimitrakopoulos. 2019. Multi-armed bandits for autonomous timing-driven design optimization. In *Proceedings of the 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS'19)*. IEEE Press, New York, NY, 17–22. DOI: https://doi.org/10.1109/PATMOS.2019.8862056

[19] Emily Strong, Bernard Kleynhans, and Serdar Kadioğlu. 2019. MABWiser: A parallelizable contextual multi-armed bandit library for Python. In *Proceedings of the IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI'19)*. IEEE Press, New York, NY, 909–914. DOI: https://doi.org/10.1109/ICTAI.2019.00129

[20] Pruek Vanna-Iampikul, Chengjia Shao, Yi-Chen Lu, Sai Pentapati, and Sung Kyu Lim. 2021. Snap-3D: A constrained placement-driven physical design methodology for face-to-face-bonded 3D ICs. In *Proceedings of the International Symposium on Physical Design (ISPD'21)*. Association for Computing Machinery, New York, NY, 39–46. DOI: https://doi.org/10.1145/3439706.3447049

[21] Pengyue Wang, Yan Li, Shashi Shekhar, and William F. Northrop. 2019. Actor-critic based deep reinforcement learning framework for energy management of extended range electric delivery vehicles. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM'19)*. IEEE Press, New York, NY, 1379–1384. DOI: https://doi.org/10.1109/AIM.2019.8868667

[22] Chang Xu, Gai Liu, Ritchie Zhao, Stephen Yang, Guojie Luo, and Zhiru Zhang. 2017. A parallel bandit-based approach for autotuning FPGA compilation. In *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'17)*. Association for Computing Machinery, New York, NY, 157–166. DOI: https://doi.org/10.1145/3020078.3021747

[23] Cody Hao Yu, Peng Wei, Max Grossman, Peng Zhang, Vivek Sarker, and Jason Cong. 2018. S2FA: An accelerator automation framework for heterogeneous computing in datacenters. In *Proceedings of the 55th Annual Design Automation Conference (DAC'18)*. Association for Computing Machinery, New York, NY. DOI: https://doi.org/10.1145/3195970.3196109

[24] Chen Zhong, Ziyang Lu, Mustafa Cenk Gursoy, and Senem Velipasalar. 2019. A deep actor-critic reinforcement learning framework for dynamic multichannel access. *CoRR* abs/1908.08401.

Received 14 August 2022; revised 30 November 2022; accepted 14 January 2023