MapTune: Advancing ASIC Technology Mapping via Reinforcement Learning Guided Library Tuning

Mingju Liu
University of Maryland, College Park
Maryland, USA
mliu9867@umd.edu

Daniel Robinson
Massachusetts Institute of Technology
Massachusetts, USA
daniel_r@mit.edu

Yingjie Li
University of Maryland, College Park
Maryland, USA
yingjiel@umd.edu

Cunxi Yu
University of Maryland, College Park
Maryland, USA
cunxiyu@umd.edu

Abstract

Technology mapping involves mapping logical circuits to a library of cells. Traditionally, the full technology library is used, leading to a large search space and potential overhead. Motivated by randomly sampled technology mapping case studies, we propose MapTune framework that addresses this challenge by utilizing reinforcement learning to make design-specific choices during cell selection. By learning from the environment, MapTune refines the cell selection process, resulting in a reduced search space and potentially improved mapping quality.

The effectiveness of MapTune is evaluated on a wide range of benchmarks, different technology libraries and technology mappers. The experimental results demonstrate that MapTune achieves higher mapping accuracy and reducing delay/area across diverse circuit designs, technology libraries and mappers. The paper also discusses the Pareto-Optimal exploration and confirms the perpetual delay-area trade-off. Conducted on benchmark suites ISCAS 85/89, ITC/ISCAS 99, VTR8.0 and EPFL benchmarks, the post-technology mapping and post-sizing quality-of-results (QoR) have been significantly improved, with average Area-Delay Product (ADP) improvement of 22.54% among all different exploration settings in MapTune. The improvements are consistently remained for four different technologies (7nm, 45nm, 130nm, and 180 nm) and two different mappers.

1 Introduction

Targeted specialization of functionality in hardware has become arguably the best means for enabling improved compute performance and energy efficiency. However, as the complexity of modern hardware systems explodes, fast and effective hardware explorations are hard to achieve due to the lack of guarantee in the existing in electronic design automation (EDA) toolflow. Several major limitations prevent practical hardware explorations [1–3]. First, as the hardware design and technology advance, the design space of modern EDA tools has increased dramatically. Besides, evaluating a given design point is extremely time-consuming, such that only a very small sub-space of the large design space can be explored. Last but not least, while the initialization of design space exploration is important for the final convergence, it is difficult to initialize the search for unseen designs effectively.

Recent years have seen increasing employment of decision intelligence in EDA, which aims to reduce the manual efforts and boost the design closure process in modern toolflows [1, 2, 4–12]. For example, various of machine learning (ML) techniques have been used to automatically configure the tool configurations of industrial FPGA toolflow [1, 4, 5, 13–15] and ASIC toolflow [2, 7**?** –9]. These works focus on end-to-end tool parameter space exploration, which are guided by ML models trained based on either offline [2] or online datasets [1, 4]. Moreover, exploring the sequence of synthesis transformations (also called synthesis flow) in EDA has been studied in an iterative training-exploration fashion through Convolutional Neural Networks (CNNs) [8] and reinforcement learning [9]. While the design quality is very sensitive to the sequence of transformations [8], these approaches are able to learn a sequential decision making strategy to achieve better quality-of-results [8, 9]. Moreover, [15, 16] demonstrate the effectiveness of lightweight Multi-Arm Bandit (MAB) models in identifying the optimal synthesis flow and it achieves a balance between exploring and exploiting arms through multiple trials to maximize overall payoffs. In addition, neural network based image classification and image construction techniques have been leveraged in placement and route (PnR), in order to accelerate design closure in the physical design stage [17–23]. As the design of digital circuits continues to grow in complexity, technology mapping process faces an increasingly large search space due to the vast number of cells contained in modern technology libraries. Utilizing the entire technology library as an input for technology mapping can result in excessive runtime and sub-optimal results, which has been demonstrated with our comprehensive case studies using 7nm ASAP library [24–26].

To address this issue, we argue that carefully calibrated partial technology libraries that contain a subset of cells have been proposed to mitigate the search space and reduce runtime. However, the selection of an optimal subset of cells requires significant expertise and experience to carefully consider the design goals, target technology, and characteristics of each cell. This process is known as cell selection and represents a significant challenge for EDA

ICCAD '24, October 27–31, 2024, Newark, NJ, USA

ACM ISBN 979-8-4007-1077-3/24/10
https://doi.org/10.1145/3676536.3676762

research. In addition, the optimal selection of cells process could result in a new area-delay trade-off. Thus, identifying whether there exists new performance Pareto frontier from the technology mapping during the exploration stage is of great interest.

Thus, this paper presents a novel Reinforcement Learning guided sampling framework to explore the design space of partially selected technology library in optimizing and exploring the technology mapping performance, without changing the mapping algorithms, namely MapTune. The main contributions of this work are summarized as follows:

- We present a comprehensive case study utilizing the 7nm ASAP library [24] to showcase the performance implications of various partially random-sampled libraries. Our study reveals substantial variations in delay (up to 40%) and area (up to 60%) across selected designs.
- In this paper, we introduce a novel cell selection framework, MapTune, based on Multi-Armed Bandit (MAB) and Q-Learning, seamlessly integrated within the ABC framework [27]. This framework facilitates effective library tuning for technology mapping, wherein timing evaluations are performed during the post-sizing stage using Static Timing Analysis (STA) techniques.
- We evaluate MapTune framework using designs from five distinct benchmark suites: ISCAS 85/89 [28], ITC/ISCAS 99 [29], VTR8.0 [30], and EPFL benchmarks [31] mapped on four different libraries: 7nm ASAP library [24], FreePDK45 45nm libary [32], SKYWATER 130nm library [33], GlobalFoundries 180nm MCU library [34]. The results demonstrate average Area-Dealy Product (ADP) improvements of 22.54% by solely tuning the libraries.
- MapTune will be released as open-source project in the integration of ABC [27] framework at https://github.com/Yu-Maryland/MapTune.

2 Background

2.1 Technology Mapping and Library

Technology mapping is a critical phase in the logic synthesis process, converting high-level circuit descriptions, such as those at the Register Transfer Level (RTL), into technology-specific gate-level netlists, particularly for Application-Specific Integrated Circuits (ASIC) designs. It involves selecting appropriate cells from an EDA library to realize a circuit in a chosen technology, effectively bridging high-level design with physical implementation. This step not only follows logic optimization but is also essential for optimizing Power, Performance, and Area (PPA), significantly influencing the cost, performance, and manufacturability of a circuit by determining the optimal gates and their interconnections.

Various algorithms have been developed to address the technology mapping problem, crucial in logic synthesis for ASIC design. These include tree-based approaches [35, 36], which focus on mapping trees or sub-trees to specific gates, and Directed Acyclic Graph (DAG)-based methods [37–39] that consider the entire circuit topology for enhanced optimization. Additionally, genetic algorithms [40], inspired by natural selection, use a population of solutions evolved over time to find optimal or near-optimal configurations for technology mapping. Recently, ML approaches have also been involved in improving the technology mapping process, either by correlating the technology-independent representation to technology-dependent PPAs (prediction models) [25, 41], or by directly optimizing the technology mapping algorithms [26]. These techniques aim to optimize PPA by minimizing gate count and interconnections, reducing power consumption, and meeting timing constraints. Thus, technology mapping is essential for producing efficient, cost-effective, and high-performance circuits in modern electronic systems.

Technology libraries are critical for technology mapping, as they provide predefined gates and components optimized for specific fabrication processes. Consequently, significant efforts have been made to optimize or generate standard cell libraries to improve the PPAs in the design flow [42]. However, there has been no effort to analyze the impact of these libraries on the technology mapping procedure. Specifically, in this work, we observe and demonstrate a counter-intuitive finding: *a partially selected set of cells from the full technology library might significantly improve the technology mapping PPAs*. The main intuition behind this is that the complexity and algorithmic space of technology mapping algorithms increase as the number of cells in the libraries increase, while most practical technology mappers in academic and industrial toolflows are heavily heuristic-based. Our comprehensive case studies in Section 3 will first discuss the impact of specific cell selection within a full library on the technology mapping PPA results using the ABC framework.

2.2 Learning-based techniques in EDA

Learning-based techniques have found widespread application across various aspects of the EDA area, including synthesis, placement and routing, and design space exploration for different design stages. Numerous studies have achieved substantial success in addressing placement problems using reinforcement learning [20, 43–45]. Additionally, various studies such as [46], [47], [48], have explored different learning techniques, including Convolutional Neural Networks (CNN) and Generative Adversarial Networks (GAN), during the routing phase. Another notable application is Design Space Exploration (DSE) for logic synthesis. The optimization of quality-of-results (QoR) in logic synthesis often requires extensive tuning runtime, making efficient DSE a challenging task due to the exponential number of potential design permutations. In response to this challenge, [44, 49] have applied GAN and CNN techniques to automate design space exploration and synthesis design flow.

3 MapTune Case Studies

Technology mapping plays a crucial role in the logic synthesis process within the domain of EDA. An excessively large technology mapping library can impose significant pressure on the exploration of design space and make the search for optimal gate selection challenging. Therefore, reducing the library size in a reasonable manner presents opportunities for optimizing technology mapping. Thus, to explore the impact of library sampling size on technology mapping performance, we conduct a comprehensive case study that involves random sampling from the 7nm ASAP library [24].

In particular, we use the complete 7nm ASAP technology library with 161 cells as the baseline. This baseline approach represents the conventional method that is widely employed in the industry. Then,

we randomly sample the technology library space using three different sampling ranges: 1) 75 – 100 cells, 2) 100 – 125 cells, and 3) 125 – 150 cells, denoted as Sampling 1, Sampling 2, Sampling 3 in Figure 1, respectively. Note that all tests with different random-sampled partial libraries are conducted on synthesis framework ABC [50] with mapping, gate sizing, and STA timing analysis commands[1]. The case study results are presented in Figure 1.

By undertaking this case study, we aimed to investigate optimization opportunities through various library sampling sizes. Our analysis primarily focused on design-specific scenarios, aiming at providing valuable insights into the selection of an appropriate sampling size for efficient technology mapping. Based on the experimental results summarized in Figure 1, three critical observations have been summarized as follows:

Observation 1 – The sampled partial libraries has significant impacts on QoR in the post-sizing stage. It is evident that the results of the baseline, indicated by the red star, fall within the distribution range of QoR obtained from the sampled partial libraries. Consider design `s13207` illustrated in Figure 1a as an example. In terms of area, ~30% of the results from the sampled libraries tend to outperform the baseline, while in terms of delay, ~60% exhibit better performance. By reducing the number of available components, technology mapping can focus on a refined library of cells that aligns precisely with the specific design requirements. This approach has demonstrated promising results, facilitating fine-grained optimizations and improved performance.

Observation 2 – QoR distributions vary significantly with different sampling sizes. We discover that larger sampling sizes tend to generate more clustered results, while smaller sampling sizes reveal greater potential for optimization. Consider design `c2670` illustrated in Figure 1b. The results obtained from the three different sampling sizes exhibits various improvements in both delay and area optimization compared to the baseline. Notably, the smallest sampling range (75 - 100 cells), represented by the blue dots, reveals a wider distribution of QoR than the two larger sampling sizes. This disparity suggests that certain components may be underutilized when larger sampling sizes are employed, thereby missing out on the opportunity to achieve better quality-of-results. This observation provides us with insight into the importance of reaching a balance between exploitation and exploration, which is a key consideration in developing our framework.

Observation 3 – Technology mapping on partially sampled libraries likely outperforms the baseline but not for all designs. From the two aforementioned observations, we can draw an initial conclusion that partially sampled libraries for technology mapping are likely to achieve better results compared to using the full library. However, it is important to note that there are exceptions that contradict this conclusion. For example, when examining design `b20` in Figure 1c, we observe that the baseline point falls within the lower left corner of the range of results obtained from the three sampled libraries. This suggests that narrowing down the library size for this specific design does not entirely benefit technology mapping. Additionally, we must consider another potential drawback of smaller sampling sizes: while smaller sizes offer greater potential for improved results, there is a risk of failed mapping due to the reduced number of available components, which limits the ability to find a suitable mapping solution for certain designs. Therefore, it is crucial to carefully evaluate the trade-off between optimization objectives and the probability of successful mapping when determining the appropriate sampling size.

Through this case study, we have highlighted the impact of partially sampled library on the the potential for achieving better results, and the trade-off between optimization and successful mapping. Our observations underscore the importance of tuning the technology mapping library to meet design-specific requirements. Furthermore, this study motivates us to explore a new framework to address this specific problem iteratively by leveraging learning-based techniques to hold promise for advancing technology mapping in design-specific applications.

[1] `read library.lib;map;topo;upsize;dnsize;stime`

4 Approach

Table 1: Notations of MapTune formulation.

Notation	Description
$\mathcal{L}$	Set of all cell variants in the Library file
$\mathcal{N}$	Number of cells variants in the Library file
$\mathcal{A}$	Action space with a discrete and finite set of actions
a^i	Action that select cell variant i
$\mathcal{S}$	A state indicates which actions have been taken/which cell variants have been chosen
$ADP_{\mathcal{S}}$	Normalized Area-Delay Product under current state $\mathcal{S}$
p_{a^i}	Probability of cell variant i minimizing Area-Delay Product

4.1 Formulation of MapTune

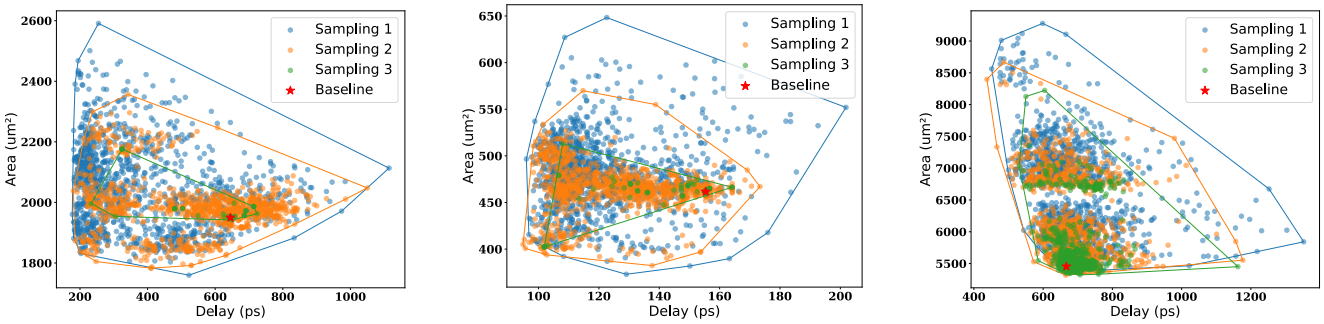
An overview of the MapTune framework is shown in Figure 2. MapTune is dedicated to addressing a cell selection problem with a pool of $\mathcal{N}$ candidates, denoted as $\mathcal{L}$ (LibSet). Each candidate in $\mathcal{L}$ is associated with two performance metrics: *Delay* and *Area*. In this work, we use the Area-Delay Product (*ADP*) as a single metric to assess overall circuit efficiency. By definition, $ADP = Delay \times Area$. In each decision iteration, a subset of n candidates is selected from $\mathcal{L}$, forming a potential solution. Each candidate $i \in \mathcal{L}$ corresponds to a binary decision variable $S_i \in \{0, 1\}$. A solution is then represented by a multi-hot encoded vector $\mathcal{S} \in \{0, 1\}^{\mathcal{N}}$, where $\mathcal{S} = [S_0, S_1, \ldots, S_{\mathcal{N}-1}]$ and $\sum_{i=0}^{\mathcal{N}-1} S_i = n$. Each distinct $\mathcal{S}$ represents a unique combination of candidates, each associated with a specific reward.

Here are the essential model formulation configurations:

Action Space: The action space $\mathcal{A}$ consists of a discrete and finite set of actions, where each action corresponds to selecting a specific cell to form a subset of cells from the technology library, subsequently utilized for the technology mapping of a design. The cardinality of this action space is equal to $\mathcal{N}$, denoting the total number of unique cells in the initial technology library. Here, we define each action as $a^i : S_i = 1, i \in [0, \mathcal{N} - 1], a^i \in \mathcal{A}$, where taking action a^i means selecting cell i from the original library.

State: During each iteration of forming a subset of cells from the original library with all candidate cells, the state $\mathcal{S}$ represents the current data collection condition, where $\mathcal{S} = [S_0, S_1, \ldots, S_{\mathcal{N}-1}]$. And when $\sum_{i=0}^{\mathcal{N}-1} S_i = n$, where n is the required subset size, it refers to a complete state.

Reward: The reward function corresponds to the cell selections made by the agent. It is defined as $\mathcal{R}_{\mathcal{S}} = -ADP_{\mathcal{S}} = -\left(\frac{D_{\mathcal{S}}}{D_{Base}} \cdot \frac{A_{\mathcal{S}}}{A_{Base}}\right)$,



(a) s13207 (b) c2670 (c) b20

Figure 1: Technology mapping results of selected designs: Baseline: All 161 cells of ASAP7 library; Sampling 1: Randomly sampling 75 - 100 cells; Sampling 2: Randomly sampling 100 - 125 cells; Sampling 3: Randomly sampling 125 - 150 cells.

where $D_{\mathcal{S}}$ and $A_{\mathcal{S}}$ are the metrics derived from the technology mapping of the design using the selected subset of cells indicated at state $\mathcal{S}$. The terms D_{Base} and A_{Base} represent the baseline metrics, established using all the cells in the original library for technology mapping. In this case, $ADP_{\mathcal{S}}$ is a product of normalized $D_{\mathcal{S}}$ and $A_{\mathcal{S}}$. The negative function is employed to invert the metric, encouraging maximizing the reward by minimizing ADP, thus optimizing both metrics concurrently.

Formally, we define the probability vector as $\boldsymbol{p} = [p_{a^0}, p_{a^1}, \ldots, p_{a^{N-1}}]$. **The probability p_{a^i} is defined as the likelihood of selecting cell i for the sampled library can maximize the reward.** These vectors are updated at each decision epoch based on the observed performance metric, i.e., ADP. The objective is to iteratively refine $\boldsymbol{p}$ such that the probability of selecting candidates that lead to minimizing ADP is maximized.

4.2 Implementation

In the realm of Reinforcement Learning, we choose two directions to formulate the MapTune Framework: Multi-Armed Bandit (MapTune-MAB) and Q-Learning (MapTune-Q). More specifically, our prominent MapTune-MAB algorithms are with ϵ-greedy strategy [51] and Upper Confidence Bound strategy [52], denoted as MapTune-ϵ and MapTune-UCB, respectively. For MapTune-Q methods, we implement and compare both the Deep Q-Network (DQN) [53] and the Double Deep Q-Network (DDQN) [54].

4.2.1 MapTune-MAB As for the bandit problem settings, we refer to each cell $i \in \mathcal{L}$ in the library as an arm i, during each iteration, if an action a^i is taken, this means arm i is selected for this action.

MapTune-ϵ Agent. The MapTune-ϵ agent utilizes ϵ-greedy to balance exploration and exploitation via a parameter $\epsilon \in [0, 1]$. This parameter dictates the probability of random action selection (exploration) vs. choosing the action with the highest probability leads to higher reward based on historical data (exploitation). Formally, the agent selects action $a^i \in \mathcal{A}$ according to the following rule:

$$a^i = \begin{cases} \arg\max_{a^i \in \mathcal{A}} p_{a^i} & \text{with parameter } 1-\epsilon \\ \text{a random selection } a^i \in \mathcal{A} & \text{with parameter } \epsilon \end{cases}$$

MapTune-UCB Agent. The MapTune-UCB agent integrates a confidence interval around the reward estimates based on historical trial data to tackle the similar exploration-exploitation problem effectively. Action selection is governed by the following formula:

$$a^i = \arg\max_{a^i \in \mathcal{A}} (p_{a^i} + c\sqrt{\frac{\log(t)}{n_{a^i}}}) \tag{1}$$

where t is the current iteration, n_{a^i} is the number of times that action a^i is taken during t iterations , c is the coefficient that modulates the extent of exploration.

Note that, for both MapTune-ϵ and MapTune-UCB agents, the probability vector $\boldsymbol{p}$ are updated as following:

$$p_{a^i}(t+1) = \frac{p_{a^i}(t) n_{a^i}(t) + \mathcal{R}_{\mathcal{S}}(t)}{n_a(t)} \tag{2}$$

where $p_a(t)$ is the probability of action a^i at iteration t, $n_{a^i}(t)$ is the number of times that action a^i is taken during t iterations. In this context, the probability of taking action a^i leads to minimizing ADP is an average of the obtained reward during the data trial process. Note that, we ensure every time when updating p_{a^i} for the next iteration $(t+1)$ at state $\mathcal{S}(t)$, a required number of arms has been selected, so that the reward $\mathcal{R}_{\mathcal{S}}$ is influencing a subset of arms during each iteration.

4.2.2 MapTune-Q Following the same action space, state and reward setup, we can implement MapTune-Q Agent similarly. Instead of influencing the probability of action decision directly from the observed reward, our prominent MapTune-Q methods facilitate the DQN to approximate the optimal state-action value function with the associated reward. Both MapTune-DQN Agent and MapTune-DDQN Agent are implemented as following by adapting the same environment settings:

MapTune-DQN Agent. Given a state vector $\mathcal{S}$ and action a_i as the input of the DQN, the model will predict a Q-value $Q(\mathcal{S}, a^i)$. To keep the consistency, we use the same probability $p_{a^i} = Q(\mathcal{S}, a^i)$ to refer to the predicted Q-value of the taken action a^i. The model also derives a target Q-value, denoted as $p_{a^i}^{tar}$ through Bellman equation as follows:

$$p_{a^i}^{tar} = \mathcal{R} + \gamma \max_{a^i} p_{a^i} \tag{3}$$

where $\mathcal{R}$ is the current reward introduced by taking action a^i, and γ is the discount factor to emphasize the significance of the future

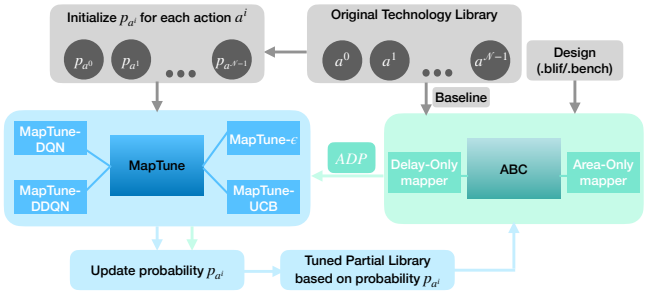


Figure 2: MapTune Framework Overview. Termination is based on the user-defined number of iterations for p update.

rewards. Note that, only after a certain number of actions have been taken reaches the required selection size, the $\mathcal{R}$ will be reflected as the actual $-ADP$ as aforementioned.

During each iteration, we use the DQN by parameter θ to calculate the Q-function and choose the action with the highest probability (i.e., p_{a^i}) leads to maximizing the reward. For the trainable parameters of DQN, we use Mean Squared Error (MSE) as the loss function to calculate between the predicted Q-value (p_{a^i}) and target Q-value ($p^{tar}_{a^i}$), then through backward propagation, the network parameters are updated hence affecting further probabilities of chosen actions. It can defined as follows:

$$L_\theta = MSE(p_{a^i}, p^{tar}_{a^i}) \tag{4}$$

MapTune-DDQN Agent. To mitigate the overestimation bias in the DQN due to the maximization step in the Bellman Equation 3, we use DDQN to optimize this process. DDQN incorporates additional network, the target DQN (Q_{target}), which mirrors the online DQN (Q_{online}) in terms of number and configurations of layers but with only periodically updated weights. When taking a state $\mathcal{S}$ and action a^i, both Q_{target} and Q_{online} will produce a Q-value denoted as, $p_{a^i} = Q_{online}(\mathcal{S}, a^i)$ and $p'_{a^i} = Q_{target}(\mathcal{S}, \arg\max_{a^i} p_{a^i})$, respectively. Consequently, we calculate the target Q-value $p^{tar}_{a^i}$ as following:

$$p^{tar}_{a^i} = \mathcal{R} + \gamma p'_{a^i} \tag{5}$$

While Q_{online} update the parameters using the same settings as in MapTune-DQN Agent, the weight update of Q_{target} is under a soft update rule as follows:

$$\theta_{target} \leftarrow \tau\theta_{online} + (1-\tau)\theta_{target} \tag{6}$$

where τ is a small coefficient that controls the rate of the update hence stabilizing the learning process.

5 Results

We demonstrate the proposed approach on designs from five benchmark suites: ISCAS 85, ISCAS 89, ITC/ISCAS 99, VTR8.0 and EPFL benchmarks, to evaluate the performance of MapTune after technology mapping and gate sizing are performed. Specifically, MapTune explores the technology libraries and evaluates it by mapping on the library using ABC with the same command as in Section 3. Note that, by default, the ABC built-in `map` command is a Delay-driven mapper. To provide more robustness to this work, we also use `map -a` command in ABC which is an Area-driven mapper by default for experiments. All experiments are conducted with an Intel®Xeon®Gold 6418H CPU and NVIDIA RTX™A6000 GPU. We evaluate the mapped results with different sampling sizes in MapTune, which searches for the best achievable results. Through the observed results, we evaluate the design space exploration using MapTune, which aims to search for Pareto Frontier in area-delay trade-offs. All experiments are conducted with the 7nm ASAP library [24], FreePDK45 45nm libary [32], SKYWATER 130nm library [33], GlobalFoundries 180nm MCU library [34]. For simplicity, we will refer to them as `ASAP7`, `NAN45`, `SKY130`, and `GF180` respectively.

All results presented in this section are conducted with MapTune framework given a one-hour timeout constraint. All MapTune-MAB and MapTune-Q methods are implemented with a batch size of 10. Following the approach used in the motivating case studies, MapTune is evaluated by fixing its sampling size range over the same technology library among different optimization methods. To ensure a fair comparison, sampling sizes are set as follows: 1) 45 - 135 cells; 2) 35 - 75 cells; 3) 220 - 310 cells; and 4) 40 - 130 cells for `ASAP7`, `NAN45`, `SKY130`, and `GF180` library, respectively, with a step size of 10 within the sampling range.

5.1 Technology Mapping Results

Due to the variations of the proposed approaches, we structure the experimental results to answer the three following research questions (RQ):

RQ1: How effective is MapTune in optimizing ADP?

MapTune shows a stable convergence rate regardless of methods/designs. In Section 4, we employ Normalized *ADP* as a single metric to guide our MapTune framework. Here, we compare the normalized *ADP* optimization trends across selected designs and methods. Due to page limits, we choose nine representative designs mapped on `ASAP7` library with Delay-driven mapper, as depicted in Figure 3. We focus on the first 1200 seconds time span to emphasize the rapid convergence rates of the different methods within MapTune.

As illustrated in Figure 3, across nine selected designs, various MapTune methods are able to converge to a lower achivable *ADP* within the given optimization time span. Take design `bar` in Figure 3b as an example, all four MapTune methods can achieve at least ~15% ADP reduction within 300 seconds, while MapTune-UCB can obtain an over 20% ADP reduction in the first 15 seconds which is significantly rapid.

Despite the rapid convergence feature among various methods and their variational settings, we do notice MapTune-MAB methods show a slight superiority than MapTune-Q methods in terms of general convergence rates and lowest achievable *ADP*. For instance, for design `c880` as shown in Figure 3e, MapTune-ϵ converges to the lowest *ADP* within 160 seconds, while MapTune-DDQN eventually achieves a ~5% higher *ADP* despite a similar convergence time. MapTune-DQN converges to a similar *ADP* as of MapTune-ϵ however suffers more than 5× convergence time.

RQ2: Are MapTune adaptive to different technologies?

MapTune is effective regardless of various technology libraries. In Figure 4, we showcase the final converged *ADP* for eight selected designs on four libraries tuned by MapTune. Note that, in this figure, we choose the one with the best final converged

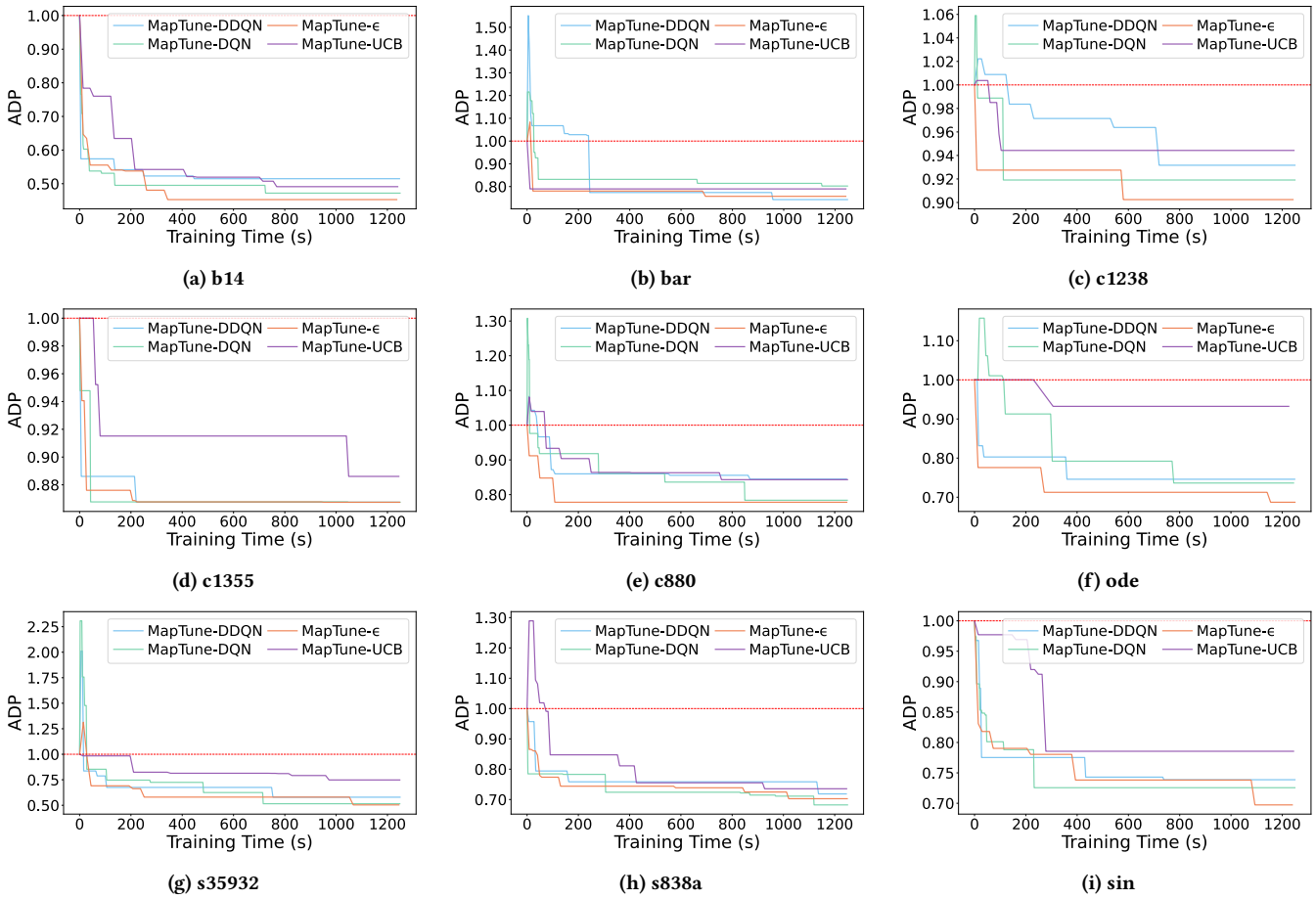


Figure 3: Comparison of ADP convergence rates of nine selected designs mapped on ASAP7 library tuned by various MapTune-MAB and MapTune-Q methods with ABC Delay-driven mapper. Baselines (constant one) are collected with the original technology library. *The lower the better.

ADP within MapTune-MAB and MapTune-Q methods respectively, denoted as Best MAB and Best QL.

As shown in Figure 4, for the same design mapped on different technologies, MapTune achieves at least a 4% *ADP* reduction (as shown in Figure 4a, design `b14` mapped on `GF180` tuned by MapTune-MAB with Delay-driven mapper) over the baseline (i.e., all cells retained in the original library, depicted by the red dashed line). More specifically, for design `s838a` mapped on various technology libraries tuned by MapTune, an average of 36% *ADP* reduction can be obtained. This highlights effectiveness of MapTune on diverse technology libraries.

RQ3: How effective is MapTune regarding different technology mappers?

MapTune is effective regardless of different technology mappers. Besides various technology libraries are compared in Figure 4, both ABC built-in Delay-driven and Area-driven mappers are also evaluated for MapTune framework. For the eight selected designs with both mappers on different technology libraries tuned by MapTune, the final converged ADP are all brought to a lower level comparing to the baseline. For example, for design `s35932` shown in Figure 4f, MapTune achives an average of 40.12% ADP reduction with Delay-driven mapper while an average of 40.00% ADP reduction with Area-driven mapper highlighting that MapTune is effective regardless of technology mappers.

5.2 Pareto-Optimal Exploration

While we have confirmed that MapTune is able to identify design point that significantly improves the quality-of-results (QoR), we want to see whether MapTune is able to identify Pareto frontier of technology mapping. One of the key aspects in this domain is the trade-off between delay and area, two primary metrics that dictate the efficiency and compactness of a given solution. With the results at hand, the focus lies on discussing how the algorithm has managed to identify a new frontier that is superior to the baseline and confirming the perpetual trade-off between delay and area.

We present the exact Delay and Area results in Table 2 and Table 3 from the Delay-driven mapper, and Area-driven mapper, respectively. For space constraints, we showcase results from 20 selected designs across the entire benchmark suites. The final mapping results for each method were obtained at the one-hour timeout,

Table 2: Detailed delay/area comparison results between ABC (baseline) and MapTune in Delay-driven mapping. Our results shows more than 20.34% delay improvements with slightly area increase (<2%) or simultaneous area reduction, on average.

Delay (*ps*)												
Design	ASAP7			NAN45			SKY130			GF180		
	Baseline	Best MAB	Best RL	Baseline	Best MAB	Best RL	Baseline	Best MAB	Best RL	Baseline	Best MAB	Best RL
b10	55.03	**42.94 (-21.97%)**	45.69 (-16.97%)	241.39	175.88 (-27.14%)	**173.1 (-28.29%)**	1145.98	**799.06 (-30.27%)**	821.04 (-28.35%)	4603.08	**3560.60 (-22.65%)**	3589.27 (-22.02%)
b12	116.45	89.63 (-23.03%)	**81.86 (-29.70%)**	424.46	284.98 (-32.86%)	**273.76 (-35.50%)**	1549.49	**1212.27 (-21.76%)**	1287.23 (-16.93%)	8173.18	**6058.64 (-25.87%)**	6120.06 (-25.12%)
b14	1013.26	**472.98 (-53.32%)**	492.91 (-51.35%)	1429.50	**1118.92 (-21.73%)**	1203.23 (-15.83%)	6885.36	**5707.95 (-17.10%)**	6102.48 (-11.37%)	27208.01	25863.77 (-4.94%)	**25577.95 (-5.99%)**
b20_1	656.76	**481.22 (-26.73%)**	503.69 (-23.31%)	1573.15	**1369.98 (-12.91%)**	1394.64 (-11.35%)	6660.12	**5800.71 (-12.90%)**	6038.11 (-9.34%)	37376.29	**27323.8 (-26.90%)**	29971.08 (-19.81%)
bar	109.36	96.03 (-12.19%)	**83.63 (-23.53%)**	459.82	**259.00 (-43.67%)**	261.78 (-43.07%)	1716.51	**1095.46 (-36.18%)**	1166.22 (-32.06%)	10544.08	**6152.74 (-41.65%)**	6947.68 (-34.11%)
c1238	94.61	**80.32 (-15.10%)**	84.3 (-10.90%)	388.76	**308.88 (-20.55%)**	326.04 (-16.13%)	1764.11	**1295.82 (-26.55%)**	1349.81 (-23.48%)	8039.22	**6302.22 (-21.61%)**	6990.11 (-13.05%)
c1355	111.23	103.75 (-6.72%)	103.75 (-6.72%)	579.92	**392.28 (-32.36%)**	406.47 (-29.91%)	1976.14	**1815.32 (-8.14%)**	1832.60 (-7.26%)	11295.16	**8955.87 (-20.71%)**	8955.90 (-20.71%)
c5315	198.75	**168.02 (-15.46%)**	174.85 (-12.03%)	884.01	**667.62 (-24.48%)**	687.15 (-22.27%)	3262.46	**2927.58 (-10.26%)**	3019.47 (-7.45%)	18189.21	14338.69 (-21.17%)	**13708.21 (-24.64%)**
c880	118.43	**107.99 (-8.82%)**	112.94 (-4.64%)	362.81	342.43 (-5.62%)	342.43 (-5.62%)	1825.23	**1652.25 (-9.48%)**	1688.04 (-7.52%)	8526.98	**6953.58 (-18.45%)**	7051.45 (-17.30%)
multiplier	1598.54	**1260.14 (-21.17%)**	1267.93 (-20.68%)	4856.00	**4216.85 (-13.16%)**	4429.79 (-8.78%)	34431.54	**20674.34 (-39.96%)**	21196.68 (-38.44%)	99312.62	96962.83 (-2.37%)	**92458.75 (-6.90%)**
ode	906.43	**590.64 (-34.84%)**	626.32 (-30.90%)	2015.62	**1763.65 (-12.50%)**	1864.10 (-7.52%)	9918.08	**8571.75 (-13.57%)**	8591.19 (-13.38%)	43618.88	38601.39 (-11.50%)	**38556.65 (-11.61%)**
priority	1260.12	1055.66 (-16.23%)	**1050.30 (-16.65%)**	3659.98	3477.99 (-4.97%)	3477.99 (-4.97%)	22218.13	15721.45 (-29.24%)	**15606.88 (-29.76%)**	120522.85	76260.92 (-36.72%)	**74356.55 (-38.31%)**
s1488	50.46	47.15 (-6.56%)	**46.71 (-7.43%)**	230.23	**163.86 (-28.83%)**	166.30 (-27.77%)	978.64	**800.39 (-18.21%)**	814.95 (-16.73%)	4391.10	3533.09 (-19.54%)	**3426.87 (-21.96%)**
s1494	50.36	45.99 (-8.68%)	**45.71 (-9.23%)**	253.81	**164.9 (-35.03%)**	172.67 (-31.97%)	939.60	**805.15 (-14.31%)**	810.01 (-13.79%)	5039.91	3408.24 (-32.37%)	**3372.47 (-33.08%)**
s35932	87.11	**43.75 (-49.78%)**	44.65 (-48.74%)	265.45	**166.4 (-37.31%)**	178.06 (-32.92%)	1414.57	**785.21 (-44.49%)**	788.74 (-44.24%)	6968.70	3535.62 (-49.26%)	**2956.22 (-57.58%)**
s838a	106.92	**63.02 (-41.06%)**	75.51 (-29.38%)	447.68	**249.14 (-44.35%)**	253.98 (-43.27%)	1695.11	**995.8 (-41.25%)**	1071.75 (-36.77%)	7176.47	5116.82 (-28.70%)	**4780.47 (-33.39%)**
s9234	154.28	**128.61 (-16.64%)**	133.07 (-13.75%)	560.33	**458.73 (-18.13%)**	477.27 (-14.82%)	2242.39	**1902.39 (-15.16%)**	1915.98 (-14.56%)	12882.61	**10049.83 (-21.99%)**	10356.36 (-19.61%)
sin	1639.39	**1068.69 (-34.81%)**	1181.55 (-27.93%)	4170.50	3772.51 (-9.54%)	**3627.14 (-13.03%)**	23617.97	**17100.18 (-27.60%)**	18111.63 (-23.31%)	87953.53	78433.59 (-10.82%)	**77112.08 (-12.33%)**
sqrt	81597.74	**61400.04 (-24.75%)**	74079.97 (-9.21%)	147455.03	**114225.2 (-22.54%)**	116682.71 (-20.87%)	643198.38	**482223.94 (-25.03%)**	487509.94 (-24.21%)	3517744.75	**2803110.25 (-20.32%)**	2818643.00 (-19.87%)
voter	564.57	422.69 (-25.13%)	**408.10 (-27.71%)**	1625.70	**1731.55 (6.51%)**	1757.63 (8.12%)	6910.82	**6307.93 (-8.72%)**	6362.95 (-7.93%)	37259.50	**31563.56 (-15.29%)**	35214.18 (-5.49%)
Avg. Delay Change		**-23.15%**	-21.04%		-22.06%	**-20.29%**		**-22.51%**	-20.34%		**-22.64%**	-22.14%
Area (μm^2)												
b10	118.74	122.01 (2.75%)	**117.34 (-1.18%)**	136.19	146.03 (7.23%)	**144.70 (6.25%)**	648.12	624.35 (-3.67%)	**615.59 (-5.02%)**	2179.83	2283.01 (4.73%)	**2228.13 (2.22%)**
b12	629.39	**577.06 (-8.31%)**	633.82 (0.70%)	692.40	**734.43 (6.07%)**	754.38 (8.95%)	3400.76	3671.02 (7.95%)	**3538.39 (4.05%)**	11208.69	**11676.27 (4.17%)**	11733.34 (4.68%)
b14	2951.46	**2861.88 (-3.04%)**	2950.76 (-0.02%)	3643.40	4030.17 (10.62%)	**3895.3 (6.91%)**	18531.51	**18950.68 (2.26%)**	18964.44 (2.34%)	58690.87	**59384.55 (1.18%)**	59619.44 (1.58%)
b20_1	5034.18	5584.96 (10.94%)	**5248.57 (4.26%)**	6641.49	7037.83 (5.97%)	**7017.88 (5.67%)**	33651.02	34520.61 (2.58%)	**34470.56 (2.44%)**	103310.50	113142.8 (9.52%)	**107670.17 (4.22%)**
bar	1881.64	**1622.46 (-13.77%)**	1827.52 (-2.88%)	2075.33	1999.79 (-3.64%)	**1988.08 (-4.20%)**	15033.17	11781.30 (-21.63%)	**11319.61 (-24.70%)**	36782.77	**27742.94 (-24.58%)**	27960.26 (-23.99%)
c1238	265.94	**275.5 (3.59%)**	278.07 (4.56%)	367.88	**386.23 (4.99%)**	386.50 (5.06%)	1823.00	**1816.74 (-0.34%)**	1833.01 (0.55%)	5927.04	6499.99 (9.67%)	**6039.00 (1.89%)**
c1355	343.15	319.13 (-7.00%)	319.13 (-7.00%)	267.86	302.71 (13.01%)	**298.98 (11.62%)**	1919.34	**1482.67 (-22.75%)**	1512.70 (-21.19%)	4688.95	4757.00 (1.45%)	**4717.48 (0.61%)**
c5315	904.43	857.07 (-5.24%)	**840.27 (-7.09%)**	994.04	1055.49 (6.18%)	**1053.63 (5.99%)**	5277.56	5425.20 (2.80%)	**5310.09 (0.62%)**	15454.21	**16593.52 (7.37%)**	17673.55 (14.36%)
c880	224.65	**190.82 (-15.06%)**	193.16 (-14.02%)	240.73	**233.28 (-3.09%)**	234.08 (-2.76%)	1169.87	1146.10 (-2.03%)	**1143.60 (-2.25%)**	4030.39	3854.77 (-4.36%)	**3826.23 (-5.07%)**
multiplier	16728.51	16864.04 (0.81%)	**16386.52 (-2.04%)**	16237.97	17010.17 (4.76%)	**16438.00 (1.23%)**	78690.47	**81863.52 (4.03%)**	82953.30 (5.42%)	274606.34	**272070.88 (-0.92%)**	280667.28 (2.21%)
ode	9341.46	**9678.79 (3.61%)**	10084.69 (7.96%)	10211.47	10535.99 (3.18%)	**10341.81 (1.28%)**	49906.61	**50335.77 (0.86%)**	50922.59 (2.04%)	162034.30	**166284.20 (2.62%)**	167006.42 (3.07%)
priority	322.86	**315.39 (-2.31%)**	320.29 (-0.80%)	680.43	**646.38 (-5.00%)**	649.31 (-4.57%)	3663.51	**2936.57 (-19.84%)**	3016.64 (-17.66%)	10183.53	10791.60 (5.97%)	**10776.24 (5.82%)**
s1488	246.34	**223.72 (-9.18%)**	228.85 (-7.10%)	428.53	456.46 (6.52%)	**445.55 (3.97%)**	1951.87	1963.13 (0.58%)	**1949.37 (-0.13%)**	6258.52	**6921.47 (10.59%)**	7015.86 (12.10%)
s1494	246.11	**228.15 (-7.30%)**	237.01 (-3.70%)	408.04	474.54 (16.30%)	**461.24 (13.04%)**	2004.42	**1979.40 (-1.25%)**	2023.19 (0.94%)	6528.52	**7272.70 (11.40%)**	7310.02 (11.97%)
s35932	5688.3	5704.86 (0.29%)	**5658.44 (-0.52%)**	5885.78	6979.84 (18.59%)	**6663.3 (13.21%)**	32073.26	34531.87 (7.67%)	**34396.74 (7.24%)**	94316.77	112157.16 (18.92%)	**106001.81 (12.39%)**
s838a	225.58	266.41 (18.10%)	**228.61 (1.34%)**	231.95	**254.83 (9.86%)**	255.63 (10.21%)	1116.07	**1154.86 (3.48%)**	1178.63 (5.61%)	3758.18	**3705.50 (-1.40%)**	3938.19 (4.79%)
s9234	1080.55	**1121.84 (3.82%)**	1097.58 (1.58%)	1302.34	**1318.56 (1.25%)**	1331.86 (2.27%)	6632.61	**6546.28 (-1.30%)**	6816.54 (2.77%)	20105.84	20964.16 (4.27%)	**20476.83 (1.85%)**
sin	3567.78	3816.46 (6.97%)	**3578.28 (0.29%)**	4252.81	**3996.38 (-6.03%)**	4228.87 (-0.56%)	19977.91	**20264.44 (1.43%)**	20339.51 (1.81%)	63149.32	64347.90 (1.90%)	**63814.46 (1.05%)**
sqrt	4686.83	2913.67 (-37.83%)	**2631.17 (-43.86%)**	24080.18	**17959.52 (-25.42%)**	20780.19 (-13.70%)	98564.53	85310.57 (-13.45%)	**84581.12 (-14.19%)**	366113.25	**252371.17 (-31.07%)**	314102.38 (-14.21%)
voter	11340.67	**10181.97 (-10.22%)**	10391.69 (-8.37%)	16298.35	**10402.2 (-36.18%)**	10666.33 (-34.56%)	50180.62	**49650.12 (-1.06%)**	49891.60 (-0.58%)	195868.91	181044.73 (-7.57%)	**170245.55 (-13.08%)**
Avg. Area Change		-3.42%	**-3.89%**		1.76%	1.76%		**-2.68%**	-2.49%		**1.19%**	1.42%

Table 3: Detailed delay/area comparison results between ABC (baseline) and MapTune in Area-driven mapping. Our results shows more than 20.85% delay improvements with slightly area increase or simultaneous area reduction, on average.

Delay (*ps*)												
Design	ASAP7			NAN45			SKY130			GF180		
	Baseline	Best MAB	Best RL	Baseline	Best MAB	Best RL	Baseline	Best MAB	Best RL	Baseline	Best MAB	Best RL
b10	78.00	**56.59 (-27.45%)**	60.44 (-22.51%)	249.83	193.37 (-22.60%)	**179.48 (-28.16%)**	1076.14	**755.82 (-29.77%)**	815.00 (-24.27%)	5037.13	**3932.94 (-21.92%)**	4220.97 (-16.20%)
b12	133.02	107.70 (-19.03%)	**105.85 (-20.43%)**	392.13	**312.97 (-20.19%)**	316.54 (-19.28%)	1896.75	1311.05 (-30.88%)	**1279.23 (-32.56%)**	9132.96	**6493.24 (-28.90%)**	6806.19 (-25.48%)
b14	718.17	569.61 (-20.69%)	**527.29 (-26.58%)**	1558.66	1272.08 (-18.39%)	**1249.09 (-19.86%)**	7521.55	5973.44 (-20.58%)	**5952.85 (-20.86%)**	31771.41	29077.95 (-8.48%)	**28262.66 (-11.04%)**
b20_1	821.18	648.92 (-20.98%)	**643.02 (-21.70%)**	2129.22	**1465.56 (-31.17%)**	1515.98 (-28.80%)	8077.29	**5949.87 (-26.34%)**	6441.54 (-20.25%)	41607.07	**30930.43 (-25.66%)**	31001.52 (-25.49%)
bar	128.94	105.15 (-18.45%)	**102.49 (-20.51%)**	470.66	**254.97 (-45.83%)**	256.95 (-45.41%)	1541.36	**1319.64 (-14.38%)**	1364.77 (-11.46%)	10048.86	**6095.23 (-39.34%)**	6163.29 (-38.67%)
c1238	124.14	96.49 (-22.27%)	**92.86 (-25.20%)**	391.93	**336.43 (-14.16%)**	356.84 (-8.95%)	1871.51	**1423.62 (-23.93%)**	1426.17 (-23.80%)	9359.37	**7142.81 (-23.68%)**	7250.06 (-22.54%)
c1355	138.47	112.23 (-18.95%)	112.23 (-18.95%)	555.56	447.75 (-19.41%)	447.75 (-19.41%)	2039.63	1878.48 (-7.90%)	**1822.67 (-10.64%)**	12310.56	9675.28 (-21.41%)	9675.28 (-21.41%)
c5315	229.13	**184.71 (-19.39%)**	186.02 (-18.81%)	892.24	**688.88 (-22.79%)**	699.90 (-21.56%)	3228.29	3011.14 (-6.73%)	**2912.83 (-9.77%)**	19885.61	**13224.56 (-33.50%)**	13645.25 (-31.38%)
c880	152.27	119.03 (-21.83%)	**112.72 (-25.97%)**	438.36	360.19 (-17.83%)	**357.32 (-18.49%)**	2093.78	**1462.71 (-30.14%)**	1472.74 (-29.66%)	10432.21	7795.75 (-25.27%)	**7553.17 (-27.60%)**
multiplier	1730.96	1613.41 (-6.79%)	**1472.71 (-14.92%)**	5065.54	**4735.58 (-6.51%)**	4814.37 (-4.96%)	27388.50	**19885.95 (-27.39%)**	19934.38 (-27.22%)	128414.73	108910.55 (-15.19%)	**98087.97 (-23.62%)**
ode	805.92	706.92 (-12.28%)	**699.14 (-13.25%)**	2235.29	**1883.13 (-15.75%)**	1986.79 (-11.12%)	9189.29	**8276.70 (-9.93%)**	8448.82 (-8.06%)	44018.07	40538.12 (-7.91%)	**39287.02 (-10.75%)**
priority	1168.48	1178.5 (0.86%)	**1143.62 (-2.13%)**	4570.09	**3492.89 (-23.57%)**	3577.46 (-21.72%)	15662.87	13501.69 (-13.80%)	**13467.05 (-14.02%)**	119129.84	**76262.34 (-35.98%)**	76684.87 (-35.63%)
s1488	74.48	**51.32 (-31.10%)**	51.40 (-30.99%)	229.02	**170.08 (-25.74%)**	171.43 (-25.15%)	979.55	**751.57 (-23.27%)**	787.24 (-19.63%)	4348.27	3673.70 (-15.51%)	**3659.24 (-15.85%)**
s1494	58.68	49.39 (-15.83%)	**48.85 (-16.75%)**	254.51	**174.14 (-31.58%)**	181.60 (-28.65%)	1035.66	**764.72 (-26.16%)**	768.71 (-25.78%)	5321.30	3817.89 (-28.25%)	**3800.43 (-28.58%)**
s35932	201.3	**48.59 (-75.86%)**	49.00 (-75.66%)	242.91	194.31 (-20.01%)	**178.06 (-26.70%)**	1178.92	**796.76 (-32.42%)**	806.94 (-31.55%)	7884.46	4682.06 (-40.62%)	**4681.56 (-40.62%)**
s838a	174.36	79.58 (-54.36%)	**76.84 (-55.93%)**	448.87	316.57 (-29.47%)	**310.30 (-30.87%)**	2649.33	1230.48 (-53.56%)	**1230.23 (-53.56%)**	7944.43	**5308.84 (-33.18%)**	5801.50 (-26.97%)
s9234	190.33	**154.52 (-18.81%)**	156.32 (-17.87%)	622.95	**498.10 (-20.04%)**	524.48 (-15.81%)	2758.56	**2060.54 (-25.30%)**	2175.33 (-21.14%)	15761.50	**10964.62 (-30.43%)**	11589.27 (-26.47%)
sin	1428.39	**1152.76 (-19.30%)**	1197.45 (-16.17%)	4334.68	**3931.06 (-9.31%)**	3932.31 (-9.28%)	20230.89	16712.39 (-17.39%)	**16284.0 (-19.51%)**	86733.62	80945.42 (-6.67%)	**80799.16 (-6.84%)**
sqrt	79262.41	**65561.55 (-17.29%)**	68122.86 (-14.05%)	144198.16	109393.30 (-24.14%)	**106396.55 (-26.22%)**	702093.06	**443822.59 (-36.79%)**	451261.25 (-35.73%)	2651079.50	2684096.50 (1.25%)	**2602480.00 (-1.83%)**
voter	554.09	**522.52 (-5.70%)**	550.56 (-0.64%)	1652.92	**1456.98 (-11.85%)**	1543.13 (-6.64%)	6949.40	6328.38 (-8.94%)	**6282.78 (-9.59%)**	34664.91	32527.51 (-6.17%)	**28732.47 (-17.11%)**
Avg. Delay Change		-22.27%	**-22.95%**		**-21.52%**	-20.85%		**-23.28%**	-22.45%		-22.34%	**-22.70%**
Area (μm^2)												
b10	102.88	107.54 (4.53%)	**97.98 (-4.76%)**	152.15	**138.59 (-8.91%)**	153.75 (1.05%)	673.15	705.68 (4.83%)	**648.12 (-3.72%)**	2322.52	2243.49 (-3.40%)	**2157.88 (-7.09%)**
b12	546.34	**543.31 (-0.55%)**	567.80 (3.93%)	680.96	729.64 (7.15%)	**714.48 (4.92%)**	3271.89	**3429.54 (4.82%)**	3608.46 (10.29%)	10820.14	11559.92 (6.84%)	**11169.18 (3.23%)**
b14	2644.46	**2647.03 (0.10%)**	2802.86 (5.99%)	3583.82	**3779.59 (5.46%)**	3840.77 (7.17%)	17915.93	**18217.47 (1.68%)**	18296.30 (2.12%)	55988.57	**56379.32 (0.70%)**	57413.26 (2.54%)
b20_1	4720.89	4664.43 (-1.20%)	**4654.64 (-1.40%)**	6335.06	6750.02 (6.55%)	**6656.65 (5.08%)**	32595.01	35129.94 (7.78%)	**33732.35 (3.49%)**	98757.66	106605.49 (7.95%)	**106533.05 (7.87%)**
bar	1637.86	**1437.0 (-12.26%)**	1453.33 (-11.27%)	1854.55	2013.89 (8.59%)	**2010.96 (8.43%)**	11581.11	**9203.83 (-20.53%)**	9551.66 (-17.52%)	30502.30	27859.28 (-8.66%)	**27637.57 (-9.39%)**
c1238	270.14	**265.24 (-1.81%)**	273.87 (1.38%)	421.34	389.42 (-7.58%)	**376.66 (-10.60%)**	1762.94	**1775.45 (0.71%)**	1834.26 (4.05%)	5602.15	6260.71 (11.76%)	**6229.98 (11.21%)**
c1355	318.89	289.27 (-9.29%)	289.27 (-9.29%)	245.52	245.25 (-0.11%)	245.25 (-0.11%)	1291.24	**1238.69 (-4.07%)**	1292.49 (0.10%)	4034.78	4197.22 (4.03%)	4197.22 (4.03%)
c5315	852.64	823.71 (-3.39%)	**808.78 (-5.14%)**	990.05	**1042.45 (5.29%)**	1064.53 (7.52%)	5166.20	**5144.93 (-0.41%)**	5367.65 (3.90%)	15234.69	18696.52 (22.72%)	**18678.96 (22.61%)**
c880	202.72	**183.12 (-9.67%)**	196.66 (-2.99%)	242.06	**238.17 (-1.61%)**	247.65 (2.31%)	1269.97	**1196.15 (-5.81%)**	1203.65 (-5.22%)	3802.09	**3723.06 (-2.08%)**	3815.26 (0.35%)
multiplier	16125.25	**14903.09 (-7.58%)**	16071.59 (-0.33%)	15953.88	15417.09 (-3.36%)	**15387.57 (-3.55%)**	79746.48	80374.59 (0.79%)	**79396.15 (-0.44%)**	247087.31	**245930.45 (-0.47%)**	271576.97 (9.91%)
ode	8505.39	**8722.81 (2.56%)**	8935.32 (5.05%)	9867.80	10173.70 (3.10%)	**9969.68 (1.03%)**	48400.17	48915.66 (1.07%)	**48889.39 (1.01%)**	154263.28	**158592.22 (2.81%)**	164299.73 (6.51%)
priority	386.54	337.56 (-12.67%)	**313.76 (-18.83%)**	724.05	**650.37 (-10.18%)**	653.56 (-9.74%)	3065.44	**3349.46 (9.27%)**	3447.06 (12.45%)	10155.00	**10916.73 (7.50%)**	11162.59 (9.92%)
s1488	206.92	**218.12 (5.41%)**	220.22 (6.43%)	426.93	460.18 (7.79%)	**453.80 (6.29%)**	1951.87	**2001.92 (2.56%)**	2004.42 (2.69%)	6383.64	**6892.93 (7.98%)**	6996.10 (9.59%)
s1494	219.52	**222.78 (1.49%)**	227.91 (3.82%)	418.68	**459.65 (9.79%)**	461.51 (10.23%)	1940.61	**2044.46 (5.35%)**	2066.98 (6.51%)	6271.69	6903.90 (10.08%)	**6798.53 (8.40%)**
s35932	5626.95	5605.95 (-0.37%)	**5596.85 (-0.53%)**	5874.88	**6054.43 (3.06%)**	6646.28 (13.13%)	30548.05	32513.68 (6.43%)	**31446.41 (2.94%)**	92244.50	97231.99 (5.41%)	**96740.27 (4.87%)**
s838a	228.85	**239.81 (4.79%)**	256.37 (12.03%)	234.88	**255.36 (8.72%)**	260.95 (11.10%)	1117.32	**1177.38 (5.38%)**	1232.43 (10.30%)	3931.60	4133.56 (5.14%)	**4105.02 (4.41%)**
s9234	1031.33	**1031.56 (0.02%)**	1037.86 (0.63%)	1260.84	1320.42 (4.73%)	**1301.27 (3.21%)**	6373.61	**6476.21 (1.61%)**	6537.52 (2.57%)	19647.04	21256.12 (8.19%)	**20044.37 (2.02%)**
sin	3154.88	3668.09 (16.27%)	**3537.46 (12.13%)**	3733.84	**3597.92 (-3.64%)**	3626.11 (-2.89%)	18058.57	**17909.68 (-0.82%)**	18355.10 (1.64%)	59511.87	58047.67 (-2.46%)	**57966.45 (-2.60%)**
sqrt	4808.83	4522.13 (-5.96%)	**4379.83 (-8.92%)**	22275.64	**16751.08 (-24.80%)**	17262.07 (-22.51%)	107377.98	**83705.28 (-22.05%)**	79645.13 (-25.83%)	295388.31	**251383.33 (-14.90%)**	253400.72 (-14.21%)
voter	8316.90	7143.73 (-14.11%)	**6764.19 (-18.67%)**	10472.95	9996.81 (-4.55%)	**9875.52 (-5.70%)**	52048.67	**45740.12 (-12.12%)**	45150.80 (-13.25%)	168534.28	**154517.94 (-8.32%)**	173170.55 (2.75%)
Avg. Area Change		**-2.19%**	-1.54%		**0.27%**	1.32%		**-0.68%**	-0.10%		**3.04%**	3.85%

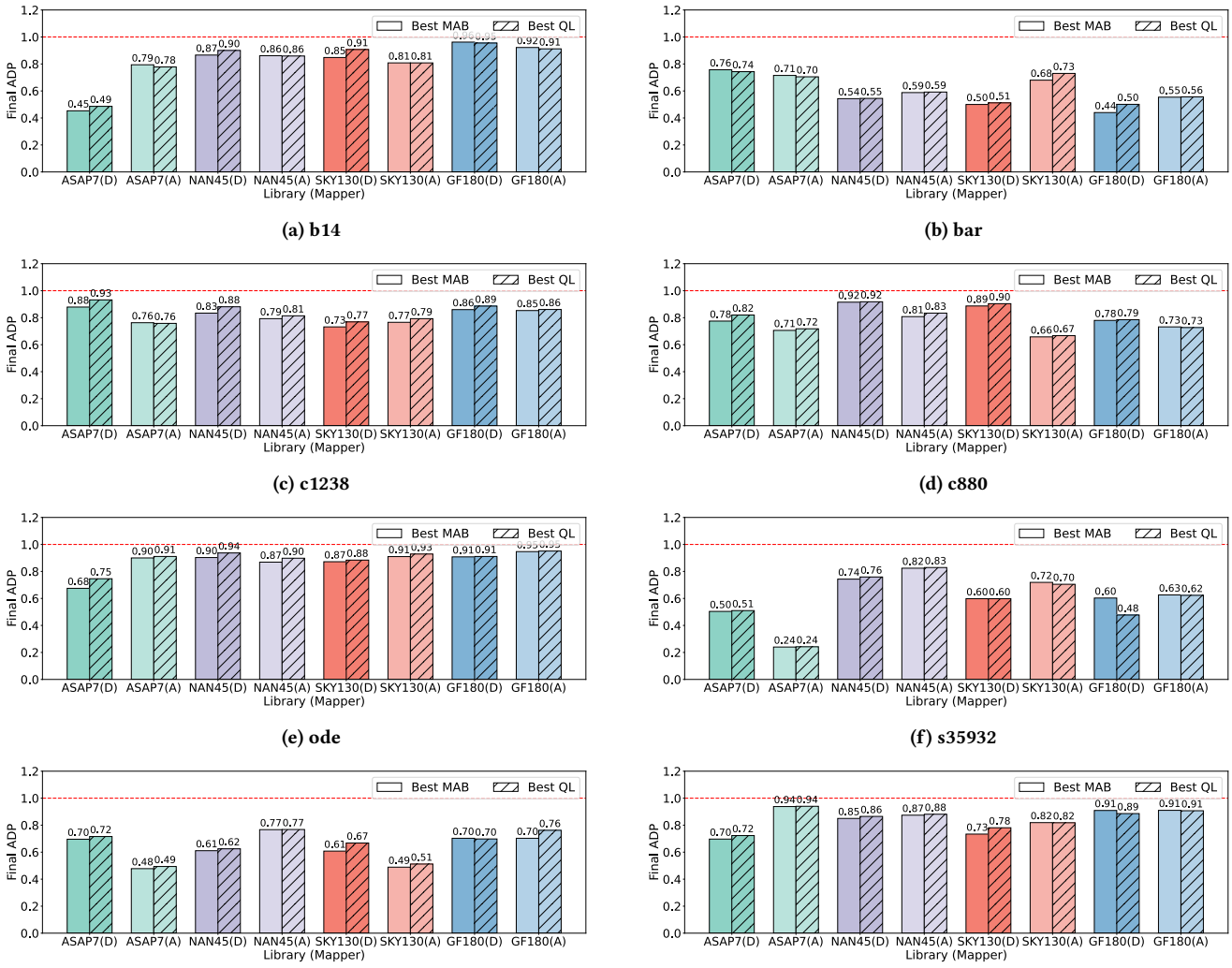


Figure 4: Comparison of final converged ADP of eight selected designs mapped on various technology libraries tuned by MapTune-MAB and MapTune-Q methods with ABC Delay-driven (D) and Area-driven (A) mappers. Baselines (constant one) are collected with the original technology library. *The lower the better.

corresponding to the best achieved ADP for each MapTune-MAB and MapTune-Q method.

Analyzing the tables reveals that MapTune emphasize on delay optimization with both ABC technology mappers. Across designs, libraries, and methods, we observe an average delay reduction of 21.77%, while area reductions average only 0.79% for Delay-driven mapper. Similarly, for Area-driven mapper, the average delay reduction is 22.30%, but there exists an area penalty of 0.5% on average. In fact, delay tends to benefit more from the tuned technology libraries optimized by MapTune with modest area trade-offs. This is evident in cases such as the design `multiplier` mapped with `SKY130` library tuned by MapTune-MAB which achieves a substantial 39.96% delay improvement with a 4.03% area penalty as shown in Table 2. There still exists opposite cases that indicating trade-off delay for area optimization. As for design `sqrt` mapped on `GF180` library tuned by MapTune-MAB with Area-driven Mapper shown in Table 3, MapTune introduces 1.25% delay penalty resulting a 14.90% area reduction ultimately. Such Pareto-Optimal trade-offs are often desirable, as significant delay/area reductions can potentially benefits more throughout the design flow than the introduced modest area/delay penalty.

6 Conclusions

This paper explores the use of partially sampled technology libraries to reduce the search space for better QoR in technology mapping. Our case study empirically demonstrates the importance of this process, given its potential impact on the trade-off between area and delay, as well as its capability to reveal new performance Pareto

frontiers. In response to this challenge, we introduce MapTune, a novel sampling framework based on Reinforcement Learning by leveraging both MAB and Q-Learning and seamlessly integrated within the ABC framework. Extensive evaluations using five distinct benchmark suites confirmed the effectiveness of MapTune framework for technology library tuning. By solely focusing on library optimization, MapTune is able to achieve an average ADP improvement of 22.54% and identify pareto-optimal results. Future work will concentrate on cross-design library exploration and integration with automatic library generation tools.

Acknowledgement

This work is funded by the National Science Foundation (NSF) under awards NSF 2229562, 2349670, 2349461, 2403134, and University of Maryland.

References

[1] Nachiket Kapre, Harnhua Ng, Kirvy Teo, and Jaco Naude. InTime: A Machine Learning Approach for Efficient Selection of FPGA CAD Tool Parameters. February 2015.

[2] Matthew M. Ziegler, Ramon Bertran Monfort, Alper Buyuktosunoglu, and Pradip Bose. Machine Learning Techniques for Taming the Complexity of Modern Hardware Design. *IBM Journal of Research and Development*, 61(4):13, 2017.

[3] Cunxi Yu, Chau-Chin Huang, Gi-Joon Nam, Mihir Choudhury, Victor N Kravets, Andrew Sullivan, Maciej Ciesielski, and Giovanni De Micheli. End-to-end industrial study of retiming. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 203–208. IEEE, 2018.

[4] Ecenur Ustun, Shaojie Xiang, Jinny Gui, Cunxi Yu, and Zhiru Zhang. LAMDA: Learning-Assisted Multi-stage Autotuning for FPGA Design Closure. In *FCCM'19*.

[5] Hung-Yi Liu and Luca P. Carloni. On Learning-based Methods for Design-space Exploration with High-level Synthesis. June 2013.

[6] Ghasem Pasandi, Shahin Nazarian, and Massoud Pedram. Approximate logic synthesis: A reinforcement learning-based technology mapping approach. In *20th International Symposium on Quality Electronic Design (ISQED)*, pages 26–32. IEEE, 2019.

[7] Dandan Li, Shuzhen Yao, Yu-Hang Liu, Senzhang Wang, and Xian-He Sun. Efficient Design Space Exploration via Statistical Sampling and AdaBoost Learning. June 2016.

[8] Cunxi Yu, Houping Xiao, and Giovanni De Micheli. Developing synthesis flows without human knowledge. In *Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, June 24-29, 2018*, pages 50:1–50:6, 2018.

[9] Abdelrahman Hosny, Soheil Hashemi, Mohamed Shalan, and Sherief Reda. Drills: Deep reinforcement learning for logic synthesis. *arXiv preprint arXiv:1911.04021*, 2019.

[10] Yuzhe Ma, Haoxing Ren, Brucek Khailany, Harbinder Sikka, Lijuan Luo, Karthikeyan Natarajan, and Bei Yu. High performance graph convolutional networks with applications in testability analysis. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[11] Cunxi Yu and Wang Zhou. Decision making in synthesis cross technologies using lstms and transfer learning. In *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, pages 55–60, 2020.

[12] Cunxi Yu. Flowtune: Practical multi-armed bandits in boolean optimization. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–9, 2020.

[13] Benjamin Carrion Schafer and Zi Wang. High-level synthesis design space exploration: Past, present and future. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

[14] Shuangnan Liu, Francis CM Lau, and Benjamin Carrion Schafer. Accelerating fpga prototyping through predictive model-based hls design space exploration. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2019.

[15] Walter Lau Neto, Yingjie Li, Pierre-Emmanuel Gaillardon, and Cunxi Yu. Flowtune: End-to-end automatic logic optimization exploration via domain-specific multi-armed bandit. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[16] Fangzhou Liu, Zehua Pei, Ziyang Yu, Haisheng Zheng, Zhuolun He, Tinghuan Chen, and Bei Yu. Cbtune: Contextual bandit tuning for logic synthesis. 2024.

[17] Zhiyao Xie, Yu-Hung Huang, Guan-Qi Fang, Haoxing Ren, Shao-Yun Fang, Yiran Chen, et al. Routenet: routability prediction for mixed-size designs using convolutional neural network. In *ICCAD*, page 80. ACM, 2018.

[18] Haoyu Yang, Shuhe Li, Yuzhe Ma, Bei Yu, and Evangeline FY Young. Gan-opc: Mask optimization with lithography-guided generative adversarial nets. In *DAC*, 2018.

[19] Biying Xu, Yibo Lin, Xiyuan Tang, Shaolan Li, Linxiao Shen, Nan Sun, and David Z Pan. WellGAN: Generative-Adversarial-Network-Guided Well Generation for Analog/Mixed-Signal Circuit Layout. In *Proceedings of the 56th Annual Design Automation Conference 2019*, page 66. ACM, 2019.

[20] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

[21] Cunxi Yu and Zhiru Zhang. Painting on placement: Forecasting routing congestion using conditional generative adversarial nets. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[22] Haoyu Yang, Piyush Pathak, Frank Gennari, Ya-Chieh Lai, and Bei Yu. Deepattern: Layout pattern generation with transforming convolutional auto-encoder. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[23] Wei Zhong, Shuxiang Hu, Yuzhe Ma, Haoyu Yang, Xiuyuan Ma, and Bei Yu. Deep learning-driven simultaneous layout decomposition and mask optimization. 2020.

[24] Lawrence T Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. Asap7: A 7-nm finfet predictive process design kit. *Microelectronics Journal*, 53:105–115, 2016.

[25] Walter Lau Neto, Matheus Trevisan Moreira, Luca Amaru, Cunxi Yu, and Pierre-Emmanuel Gaillardon. Read your circuit: leveraging word embedding to guide logic optimization. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, pages 530–535, 2021.

[26] Walter Lau Neto, Matheus T Moreira, Yingjie Li, Luca Amarù, Cunxi Yu, and Pierre-Emmanuel Gaillardon. Slap: A supervised learning approach for priority cuts technology mapping. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 859–864. IEEE, 2021.

[27] Robert Brayton and Alan Mishchenko. ABC: An Academic Industrial-strength Verification tool. In *Computer Aided Verification*, pages 24–40. Springer, 2010.

[28] Franc Brglez, David Bryan, and Krzysztof Kozminski. Combinational profiles of sequential benchmark circuits. In *1989 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1929–1934. IEEE, 1989.

[29] Luis Basto. First results of itc'99 benchmark circuits. *IEEE Design & Test of Computers*, 17(3):54–59, 2000.

[30] Jason Luu, Jeffrey Goeders, Michael Wainberg, Andrew Somerville, Thien Yu, Konstantin Nasartschuk, Miad Nasr, Sen Wang, Tim Liu, Nooruddin Ahmed, et al. VTR 8.0: Next generation architecture and CAD system for FPGAs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 7(2):6, 2014.

[31] Mathias Soeken, Heinz Riener, Winston Haaswijk, Eleonora Testa, Bruno Schmitt, Giulia Meuli, Fereshte Mozafari, and Giovanni De Micheli. The epfl logic synthesis libraries. *arXiv preprint arXiv:1805.05121*, 2018.

[32] NC State FreePDK45's Documentation. https://eda.ncsu.edu/freepdk/freepdk45/.

[33] SkyWater SKY130 PDK's Documentation. https://skywater-pdk.readthedocs.io/en/main/.

[34] GlobalFoundries 0.18UM 3.3V/(5V)6V MCU PDK's Documentation. https://gf180mcu-pdk.readthedocs.io/en/latest/.

[35] Peter Marwedel. Tree-based mapping of algorithms to predefined structures. In *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, pages 586–593. IEEE, 1993.

[36] Jason Cong and Yuzheng Ding. Flowmap: An optimal technology mapping algorithm for delay optimization in lookup-table based fpga designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(1):1–12, 1994.

[37] Alan Mishchenko, Satrajit Chatterjee, and Robert Brayton. Improvements to technology mapping for lut-based fpgas. In *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, pages 41–49, 2006.

[38] Alan Mishchenko, Satrajit Chatterjee, Robert Brayton, Xinning Wang, and Timothy Kam. Technology Mapping with Boolean Matching, Supergates and Choices. 2005.

[39] Alan Mishchenko, Satrajit Chatterjee, and Robert K. Brayton. Dag-aware AIG rewriting a fresh look at combinational logic synthesis. In *Proceedings of the 43rd Design Automation Conference, DAC 2006, San Francisco, CA, USA, July 24-28, 2006*, pages 532–535, 2006.

[40] Venkataramana Kommu and Irith Pomeranz. Gafpga: Genetic algorithm for fpga technology mapping. In *Proceedings of EURO-DAC 93 and EURO-VHDL 93-European Design Automation Conference*, pages 300–305. IEEE, 1993.

[41] Yingjie Li, Anthony Agnesina, Yanqing Zhang, Haoxing Ren, and Cunxi Yu. Boolgebra: Attributed graph-learning for boolean algebraic manipulation. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–2. IEEE, 2024.

[42] Haoxing Ren and Matthew Fojtik. Nvcell: Standard cell layout in advanced technology nodes with reinforcement learning. In *2021 58th ACM/IEEE Design*

Automation Conference (DAC), pages 1291–1294. IEEE, 2021.

[43] Mohamed A Elgammal, Kevin E Murray, and Vaughn Betz. Rlplace: Using reinforcement learning and smart perturbations to optimize fpga placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(8):2532–2545, 2021.

[44] Cunxi Yu, Houping Xiao, and Giovanni De Micheli. Developing synthesis flows without human knowledge. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018.

[45] Haoxing Ren and Jiang Hu. *Machine Learning Applications in Electronic Design Automation*. Springer Nature, 2023.

[46] Rongjian Liang, Hua Xiang, Diwesh Pandey, Lakshmi Reddy, Shyam Ramji, Gi-Joon Nam, and Jiang Hu. Drc hotspot prediction at sub-10nm process nodes using customized convolutional network. In *Proceedings of the 2020 International Symposium on Physical Design*, pages 135–142, 2020.

[47] Yi-Chen Lu, Jeehyun Lee, Anthony Agnesina, Kambiz Samadi, and Sung Kyu Lim. Gan-cts: A generative adversarial framework for clock tree prediction and optimization. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019.

[48] Mohamed Baker Alawieh, Wuxi Li, Yibo Lin, Love Singhal, Mahesh A Iyer, and David Z Pan. High-definition routing congestion prediction for large-scale fpgas. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 26–31. IEEE, 2020.

[49] Winston Haaswijk, Edo Collins, Benoit Seguin, Mathias Soeken, Frédéric Kaplan, Sabine Süsstrunk, and Giovanni De Micheli. Deep learning for logic optimization algorithms. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2018.

[50] Alan Mishchenko et al. Abc: A system for sequential synthesis and verification. *http://www. eecs. berkeley. edu/alanmi/abc*, 17, 2007.

[51] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

[52] Wassim Jouini, Damien Ernst, Christophe Moy, and Jacques Palicot. Upper confidence bound based decision making strategies and dynamic spectrum access. In *2010 IEEE International Conference on Communications*, pages 1–5. IEEE, 2010.

[53] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[54] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.