CTRL-B: Back-End-Of-Line Configuration Pathfinding using Cross-Technology Transferable Reinforcement Learning

Sung-Yun Lee, Kyungjun Min and Seokhyeong Kang
Department of EE, Pohang University of Science and Technology
Pohang, Republic of Korea
{syun.lee, kj.min, shkang}@postech.ac.kr

***Abstract*—In advanced technology nodes, the impact of the back-end-of-line (BEOL) on chip performance and power consumption becomes progressively significant. This paper presents a BEOL configuration pathfinding framework using the proposed cross-technology transferable reinforcement learning (CTRL) model. We optimize the BEOL technology parameters, including the metal stack, geometry, and design rules, to enhance the power efficiency and performance resulting from the physical design process. First, we extract various design and technology features and embed them into metal-type-wise deep neural networks. Our multi-policy model selects a BEOL parameter configuration that is estimated to improve chip power efficiency and performance. We employ a policy gradient algorithm complemented by various training strategies, such as data normalization, action-reward buffer, and network optimization, to expedite training convergence. Furthermore, we transfer the pathfinding knowledge from the trained model in the old node to the new model for efficient BEOL configuration pathfinding in the advanced node, referred to as cross-technology transfer learning. The proposed framework achieved 19% reduced total power consumption, 68% improved worst negative slack, and 87% improved total negative slack with a high reward efficiency, on average, in several designs. Further, we demonstrate that the reward efficiency of our proposed CTRL model outperforms that of the conventional fine-tuning method in transferring knowledge by 24%.**

***Index Terms*—Parameter optimization, metal stack, metal aspect ratio, metal duty cycle, design rule, progressive neural network**

I. Introduction

Despite the continual downscaling of technology nodes, the impact on the power and performance of integrated circuits has not been as substantial as the corresponding reductions in their areas [1]. This is because the back-end-of-line (BEOL) becomes critical in sub-14-*nm* nodes [2], [3]. In the copper damascene process, as dimensions are scaled down, the BEOL interconnects become progressively narrower, whereas the barrier thickness remains constant [4]. This dimensional reduction increases the parasitic resistance and capacitance (RC) of the interconnect wires, thereby increasing the wire delay [5]. Consequently, BEOL delay increasingly dominates the overall circuit delay in advanced nodes [6], [7]. Because of its involvement in clock and data signal interconnects, power delivery networks, and chip reliability, the BEOL technology exerts a significant influence on chip performance and power consumption. Therefore, the design quality in terms of power consumption, performance, area, and cost (PPAC) can be effectively improved by optimizing the BEOL technology.

Several BEOL optimization methods have been actively investigated over the past decade. Ciofi *et al.* [8] estimated the interconnect delay by modeling the RC regarding the wire geometry, including the width, thickness, and barrier. However, focusing only on minimizing the delay of a single wire segment may not yield corresponding improvements in the overall physical design results of the circuits; thus, design technology co-optimization (DTCO) for BEOL is required [9], [10]. Chan *et al.* [11] predicted routability based on the number of metal layers, whereas Cheng *et al.* [12] extended the prediction of intrinsic routability to include various BEOL technology parameters, such as the number of metal layers, metal pitch, and design rule. However, these studies were limited to considerations of chip area and cost without power and performance optimizations. Some studies [13], [14] aimed to optimize chip PPAC; however, they concentrated on standard cell optimization, despite the increasing importance of BEOL in advanced nodes. Han *et al.* [15] analyzed the impact of wire geometry on chip power and performance; however, they failed to consider the metal stack or design rules while utilizing an inefficient exploration, such as a grid search method.

In this paper, we overcome these critical limitations by presenting *CTRL-B*, an efficient BEOL configuration pathfinding framework. Our framework addresses a comprehensive set of BEOL technology parameters, including the wire geometry, metal stack, and design rule. We define a feasible BEOL parameter space while considering process cost and reliability. We utilize deep reinforcement learning (RL) to enhance the efficiency of exploring the expansive parameter space. However, in the era of rapid scaling, with the development of the new technology node, extensive design iterations are newly required for the BEOL configuration pathfinding. Our RL-based pathfinding is aimed to be a universal framework (that is, generalize to any technology node). Therefore, we propose a cross-technology transferable RL (CTRL) model based on a progressive neural network [16], a sophisticated transfer learning technique, to reduce tedious repetition in new technologies.

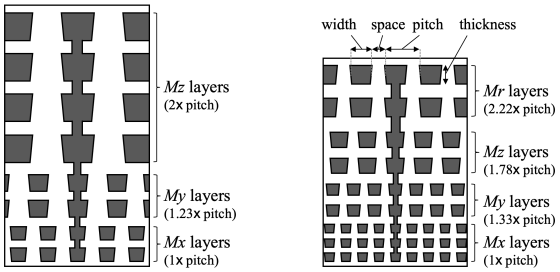
In our *CTRL-B* framework, BEOL technology and design features are extracted and embedded into a multi-policy deep neural network. The network is trained using a gradient descent-based RL algorithm to select a BEOL configuration that is expected to improve the power and performance of chips. To enhance the efficiency of the RL training process, we employ several RL techniques, such as data normalization, action-reward buffer, and network optimization. Furthermore, we implement a customized RL environment that encompasses design enablement and physical design processes. To demonstrate the effectiveness of DTCO for BEOL through our *CTRL-B* framework, we specifically focus on BEOL parameters while maintaining device or standard cell parameters constant. The main contributions of this paper are as follows:

- We propose a *CTRL-B*, a BEOL configuration pathfinding framework that focuses on improving the power and performance of chips through DTCO. Furthre, we consider the area, cost, and reliability of the chips.
- We present the first RL-based DTCO flow for efficient exploration of the expansive BEOL parameter space. We employ a gradient descent algorithm and incorporate various training strategies to improve both the efficiency of the RL training process and the balance between exploration and exploitation.
- Our framework adaptively explores the BEOL parameters based on individual design characteristics, thus achieving optimal PPAC results specific to each circuit.
- Our *CTRL-B* framework is generalizable across different technology nodes by facilitating cross-technology transfer learning. A pre-trained RL agent on an older node (14-*nm*) can significantly enhance the efficiency of pathfinding on an advanced node (7-*nm*) with only a few training iterations.

The remainder of this paper is organized as follows. Section II presents our defined BEOL configuration and parameter space. Sec-

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT). (No.2021-0-00754, S/W Systems for AI Semiconductor Design; No.RS-2023-00222085, Development of memory module/compiler for non-volatile PIM) and Samsung Electronics Co., Ltd. The EDA Tools were supported by the IC Design Educator Center.

979-8-3503-4859-0/DATE24/© 2024 EDAA



(a) *8M_2Mx_2My_4Mz* (b) *9M_3Mx_2My_2Mz_2Mr*

Fig. 1. Examples of the metal stack (MS) with a cross-sectional view. (a) *8M_2Mx_2My_4Mz* comprises eight metal layers, including two *Mx*, two *My*, and four *Mz* layers in the *N14*. (b) *9M_3Mx_2My_2Mz_2Mr* comprises nine layers, including three *Mx*, two *My*, two *Mz*, and two *Mr* layers in the *N7*.

TABLE I
METAL STACK PARAMETER SPACE FOR EACH NODE (BASELINE IN **BOLD**)

N14	*N7*
9M_2Mx_1My_6Mz	***9M_3Mx_2My_2Mz_2Mr***
9M_3Mx_1My_5Mz	*9M_3Mx_3My_2Mz_1Mr*
9M_2Mx_2My_5Mz	*9M_2Mx_2My_3Mz_2Mr*
8M_2Mx_2My_4Mz	*8M_2Mx_2My_2Mz_2Mr*
8M_2Mx_1My_5Mz	*8M_2Mx_3My_2Mz_1Mr*
8M_3Mx_1My_4Mz	*8M_3Mx_3My_1Mz_1Mr*
7M_2Mx_1My_4Mz	*7M_2Mx_2My_2Mz_1Mr*
7M_3Mx_1My_3Mz	*7M_3Mx_2My_1Mz_1Mr*
6M_2Mx_1My_3Mz	*6M_2Mx_2My_1Mz_1Mr*
6M_3Mx_1My_2Mz	*6M_3Mx_1My_1Mz_1Mr*

tion III describes the details of the proposed *CTRL-B* framework. Section IV discusses the experimental results, and the conclusion is presented in Section V.

II. BEOL PARAMETER CONFIGURATION

In this section, we define the comprehensive BEOL technology parameter space. We consider four types of BEOL parameters: metal stack, metal aspect ratio, metal duty cycle, and design rule. The metal aspect ratio and duty cycle are related to wire geometry.

A. Metal Stack (MS)

MS denotes the total number of metals, including the individual counts for each metal type. We define the metal types based on the baseline technologies for a 14-*nm* node (*N14*) [17] and a 7-*nm* node (*N7*) [18]. This paper utilizes three metal types in the *N14* (*Mx*, *My*, and *Mz*), and four types are used in the *N7* (*Mx*, *My*, *Mz*, and *Mr*) (Figure 1). The metal types are listed in ascending order based on their pitch sizes. For example, the *Mx* type has the smallest pitch (1× pitch) and is assumed to utilize extreme ultraviolet (EUV) lithography. In determining the MS parameter space, we consider the high cost of EUV lithography and accordingly limit the number of *Mx* layers to at most three, based on the international roadmap for devices and systems (IRDS) [26]. Moreover, Dong *et al.* [19] reported that five and eight metal layers were sufficient for designs with 5M cells and 50M cells, respectively. Thus, we configure the total number of metal layers between six and nine. Table I presents the MS parameter space in our framework for each node, offering 10 configured options of different metal layer combinations.

B. Metal Aspect Ratio (MAR)

$$MAR = \frac{thickness}{pitch/2} \tag{1}$$

MAR denotes a ratio of the thickness to the half-pitch of a metal wire (Equation 1). Higher MAR values indicate thicker and narrower metal structures, whereas lower MAR values indicate thinner and wider metal structures. MAR influences the resistance and capacitance of the BEOL interconnects, which in turn influences the overall power consumption, performance, and signal integrity of chips. However, a high MAR renders trench filling or electroplating challenging during the manufacturing process, whereas a low MAR is vulnerable to electromigration and physical stress [20]. We set the MAR parameter space to range from 1.0 to 2.5 with 0.1 units for each metal type, based on industrial processes [26]. BEOL configuration pathfinding is performed at a given technology node; thus, we fix the metal pitch to the baseline technology value. For example, suppose the MAR for the *Mx* type is 2.5. In that case, the metal thickness becomes 45*nm* because the pitch size is 36*nm* in the *N7*. For reference, MAR for both baseline nodes is near-2.0 for all metal types, based on the international technology roadmap for semiconductors (ITRS) [27].

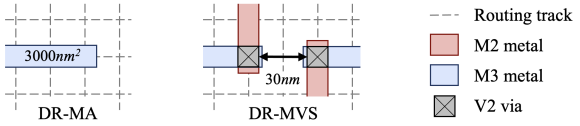


Fig. 2. Example of design rules for the minimum area rule (DR-MA) and minimum via spacing rule (DR-MVS). In the *N7*, the DR-Norm set has DR-MA=2664nm^2 and DR-MVS=18*nm*; while the DR-Hard set has DR-MA=3312nm^2 and DR-MVS=54*nm*. The design rules are met for the DR-Norm set but violated for the DR-Hard set in this example.

C. Metal Duty Cycle (MDC)

$$MDC = \frac{width}{pitch} = \frac{pitch - space}{pitch} \tag{2}$$

MDC represents a ratio of the width to the pitch of a metal wire (Equation 2). Provided the pitch is constant, a higher MDC indicates that the width increases and the spacing between them decreases, which decreases the resistance but increases the capacitance of the wires. Conversely, a lower MDC results in reduced metal width and increased spacing, leading to decreased capacitance and increased resistance. A high MDC increases crosstalk and complicates patterning and etching with high-density metals; a low MDC can also complicate patterning due to the narrow wires [20]. We set the MDC parameter space ranging from 0.4 to 0.6 with 0.1 units for each metal type. For example, suppose the MDC for *Mx* type is 0.6. In that case, the width becomes 22*nm* and the space becomes 14*nm* because the pitch is 36*nm* and the manufacturing grid is 1*nm* in the *N7*. For reference, the MDC values for both baseline nodes are 0.5 for all metal types.

D. Design Rule (DR)

We opt for two BEOL-related DRs that influence design rule violations (DRVs) and chip PPAC: minimum area rule (*DR-MA*) and minimum via spacing rule (*DR-MVS*) [29]. DR-MA specifies the area of a metal segment must exceed a specified constraint (Figure 2). DR-MVS prescribes that the minimum distance between vias must exceed a constraint. In our framework, we define two DR parameter sets: *DR-Norm* and *DR-Hard*. In the DR-Norm set, the constraint thresholds for DR-MA and DR-MVS are the same as those established in the baseline nodes. In the DR-Hard set, we tighten the existing constraints from the baseline; the value for DR-MA in each metal layer is increased by the $width \times pitch$ value; for DR-MVS in each cut layer, we add the $pitch$ value of the underlying metal layer to the constraint.

III. PROPOSED CTRL-B FRAMEWORK

This section presents our *CTRL-B* framework, a BEOL configuration pathfinding using a cross-technology transferable RL (CTRL) model. We introduce the details of our RL model, including the Markov decision process, network, and training flow. We then describe a cross-technology transfer learning from the *N14* to the *N7*.

A. Overall Flow

Our proposed *CTRL-B* framework comprises an RL agent and an environment, including a technology generator and EDA tools for the physical design flow (Figure 3). The main objective is to maximize the improvements in power consumption, worst negative slack

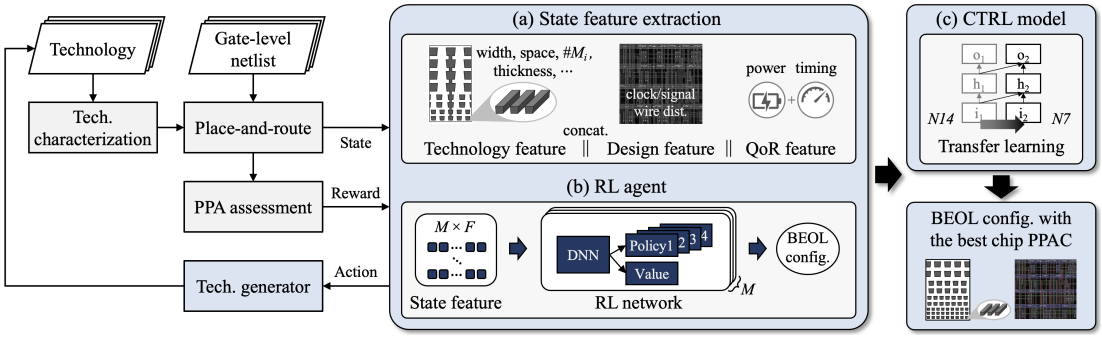


Fig. 3. Overall flow of our proposed *CTRL-B* framework. (a) After physical design using the selected BEOL configuration, features related to technology, design, and quality of result (QoR) are extracted for each metal type. (b) Subsequently, these features are embedded in our multi-policy network to select the BEOL configuration expected to maximize the reward. (c) Knowledge of pathfinding in the *N14* is transferred to our CTRL model in the *N7*.

(WNS), and total negative slack (TNS) by optimizing the BEOL parameters within small design iterations. Given a gate-level netlist and technology libraries, we conduct a technology characterization and place-and-route (P&R), and subsequently evaluate the chip PPA. The features related to technology, design, and quality of result (QoR) are extracted from the design results, and the reward and state are updated (Figure 3a). The extracted features are categorized by metal type and embedded in parallel into our RL model, which comprises multi-policy and value networks (Figure 3b). The multi-policy network creates probability vectors to select each BEOL parameter through categorical sampling, and the value network provides an estimated reward. By utilizing our technology generator, the technology files are created and characterized based on the selected BEOL configuration, which is the process of design enablement. Subsequently, the P&R flow is conducted using the updated technology files. The entire process is repeated until it reaches a user-specified stopping condition. Finally, the trained RL model in the *N14* is transferred to the CTRL model in the *N7* utilizing the progressive neural network architecture [16] (Figure 3c).

Specifically, our implemented technology generator takes the baseline technology files and selected BEOL parameters as inputs. Then, it rebuilds the interconnect technology file (ITF) [28] and Synopsys technology file (TF) [29]. To demonstrate the effectiveness of BEOL on the power and performance of chips, we adjust the BEOL parameters for the layers above M2. This is because cell pins in the advanced nodes are located in layers below M3, where routing is constrained. In ITF construction, the number of layers, thickness, and minimum width and spacing of each layer are specified based on the MS, MAR, and MDC parameters. When constructing TF, the number of layers, design rule, and minimum width and spacing of each layer are specified by utilizing the MS, MDC, and DR parameters. The manufacturing grid is considered in our technology generator, and all relevant values are rounded according to the grid unit. In constructing ITF and TF, we adopt the default values of the baseline technologies for the other variables, including sheet resistance, dielectric constant, and the other design rules. We generate parasitic RC models from the constructed ITF utilizing a technology characterization tool [28]. Subsequently, we perform P&R utilizing these RC models and the TF. We limit the routing phase to global routing during training to enhance the efficiency and practicality of our *CTRL-B* framework. Consequently, we can effectively expedite the training steps, because the trends of chip performance and power remain consistent before and after detailed routing [21].

B. Markov Decision Process

A BEOL configuration pathfinding problem is a sequential decision of the parameters to achieve the maximized design quality. This can be represented as a Markov decision process (MDP), and our RL setup is defined as follows, detailing the state, action, and reward.

TABLE II
SELECTED FEATURES FOR EACH METAL TYPE

Type	Feature
Tech.	Total number of metal layers
	Number of $^*M_i$ type layers // $^*M_i \in \{M_x, M_y, M_z, M_r\}$.
	Minimum width of metal in M_i type layers
	Minimum space between metals in M_i type layers
	Thickness of metal in M_i type layers
	Boolean indicator for M_x type
	Boolean indicator for M_y type
Design	Total number of datapath nets
	Total number of clock-path nets
	Number of wire segments of datapath in M_i type layers
	Number of wire segments of clock-path in M_i type layers
	Wirelength of datapath in M_i type layers
	Wirelength of clock-path in M_i type layers
QoR	Reward value of the current training step

- **State** s: A state is an $M \times F$ array of features, where M is the number of metal types ($M = 3$ for the *N14* and $M = 4$ for the *N7*) and F is the number of features for each metal type. To address different technologies with different numbers of metal types, we extract features for each metal type. For each metal type, the features comprise three groups: (i) technology, (ii) design, and (iii) QoR (Table II). Technology features encompass the total number of metal layers, the specific number of layers for the corresponding metal type, minimum width and spacing of metals, thickness, and two Boolean indicators for the *Mx* and *My* types. The design features include the total number of nets across all metal types as well as the number of wire segments and wirelength (WL) for the corresponding metal type for both datapath and clock-path. QoR feature indicates the reward value of the current training step. The efficiency of RL training can be increased by embedding the reward into the state by providing the agent immediate feedback on its actions, thereby expediting the value estimation and training convergence.
- **Action** a: An action is defined as a decision of the BEOL parameter configuration. The MAR and MDC parameters are determined for each metal type, whereas the MS and DR parameters are collectively determined. For example, 10 BEOL parameters are selected for the *N7*: four MARs, four MDCs, one MS, and one DR.
- **Reward** $r(s, a)$: We employ a parametric scalarization technique to address the multi-objective problem of optimizing total power consumption (P_T), WNS, and TNS. The reward is constructed to maximize the improvement over the baseline configuration (Equation 3).

$$c_p \cdot \frac{P_{T,ref} - P_T}{P_{T,ref}} + c_w \cdot \frac{WNS_{ref} - WNS}{WNS_{ref}} + c_t \cdot \frac{TNS_{ref} - TNS}{TNS_{ref}}, \quad (3)$$

where the terms subscripted with *ref* are their baseline results, and c_p, c_w, and c_t are the coefficients of total power, WNS, and TNS, respectively, which are described in Section IV-B.

C. Network Architecture

Our RL network is constructed to manage various technologies that have different numbers of metal types. The state features for each

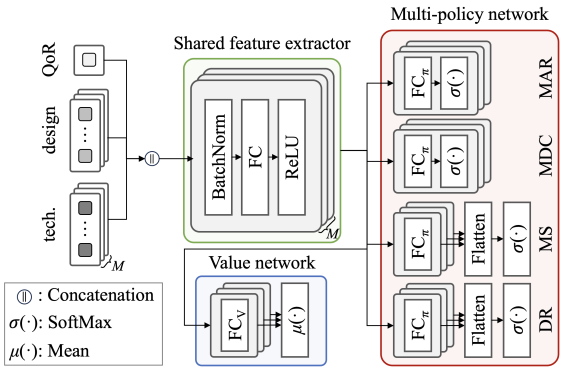


Fig. 4. Architecture of our RL network comprising multi-policy and value networks, where M denotes the number of metal types in a given technology.

metal type are embedded in parallel into our network, which comprises feature extractor, multi-policy, and value networks (Figure 4). For example, given a 3×14 state in the *N14*, three 1×14 feature vectors for each metal type are inputted in parallel to the networks that share the network parameters. The policy and value networks share the feature extractor, which comprises a normalization layer, a fully connected (FC) layer, and a ReLU activation layer. In the multi-policy network, four separate policy networks are constructed for four types of BEOL parameters: MAR, MDC, MS, and DR. For the MAR and MDC parameters, which are determined individually for each metal type, their respective policy networks include a sequential pair of an FC layer and a softmax layer. The softmax layer generates a probability vector from which the MAR or MDC parameter is selected through categorical sampling. In contrast, the MS and DR parameters are collectively determined for all metal types. The corresponding policy networks flatten the outputs from the FC layers across each metal type, and each consolidated output is processed through a softmax layer to generate a probability vector. Subsequently, the MS and DR parameters are selected through categorical sampling and the modulo operation from these probability vectors. Meanwhile, the value network comprises an FC layer followed by an arithmetic mean layer, which is designed to provide the estimated return.

D. RL Training Flow

We utilize the proximal policy optimization (PPO) algorithm [22] that updates the policy function to maximize the expected reward utilizing the policy gradient method. For our BEOL configuration pathfinding, we train our network using a non-episodic PPO algorithm (Algorithm 1). First, we construct the network comprising a θ-parameterized multi-policy network and a ϕ-parameterized value network (Line 1). Then, the state features are initialized based on the baseline technology and its corresponding physical design outcomes, and the reward is initialized to zero (Line 2). The action-reward buffer is structured as a map, with actions as keys and corresponding rewards as values (Line 3). The training continues until the maximum step (Line 5). A set of trajectories {state, action, reward, next state} is collected with a batch size by interacting with the environment using the recent policy (Lines 6-15). In step t, the action a_t is performed by the multi-policy network; thereby, the BEOL parameter configuration is selected (Line 7). To obtain the reward r_t, we first check whether the action is already processed in the action-reward buffer (Lines 8-12). If so, we retrieve the reward for the corresponding action from the buffer. Otherwise, we generate technology files and conduct the physical design; then, the reward is calculated, and the action-reward pair is stored in the buffer. Subsequently, the state is updated based on the action and reward, and the trajectory is collected for mini-batch training (Lines 13-14). We iteratively update the network parameters over the collected batches for a specified number of training epochs

Algorithm 1: RL training of policy and value networks

Inputs: Max. training step (T), batch size (B), number of epochs (E)
1 Initialize policy and value parameters θ, ϕ
2 Initialize state and reward: $s_0 \leftarrow$ baseline, $r_0 \leftarrow 0$
3 Initialize action-reward buffer $\Gamma[\,] \leftarrow \emptyset$
4 $t \leftarrow 0$
5 **for** $step = 1, \cdots, T$ **do**
6 **for** $batch = 1, \cdots, B$ **do**
7 $a_t \leftarrow get\_BEOL\_config(s_t)$
8 **if** $a_t \in \Gamma$ **then**
9 $r_t \leftarrow \Gamma[a_t]$; // Retrieve the reward for a_t from Γ
10 **else**
11 $r_t \leftarrow get\_reward\_from\_P\&R(a_t)$
12 $\Gamma[a_t] \leftarrow r_t$; // Store action-reward pair in Γ
13 $s_{t+1} \leftarrow state\_update(a_t, r_t)$
14 $S, A, R, S' \leftarrow collect\_trajectory(s_t, a_t, r_t, s_{t+1})$
15 $t \leftarrow t + 1$
16 **for** $epoch = 1, \cdots, E$ **do**
17 $L \leftarrow get\_loss(S, A, R, S')$
18 $\Delta\theta, \Delta\phi \leftarrow backpropagation(L)$
19 $\theta, \pi \leftarrow \theta + \Delta\theta, \phi + \Delta\phi$

(Lines 16-19). The PPO loss is computed, and the network is updated to minimize the loss through backpropagation.

1) Loss function: We construct the PPO loss function, including (i) policy loss, (ii) value loss, and (iii) entropy bonus, as follows:

$$L_{PPO} = L_{policy} + \beta \cdot L_{value}, -\eta \cdot L_{entropy}, \tag{4}$$

where β and η are the coefficients for value loss and entropy bonus. (i) The policy loss updates the policy network with the aim of maximizing expected future rewards. To ensure modest policy updates, this loss is constructed as Equation 5, which is the ϵ-clipped surrogate objective function. We utilize the generalized advantage estimation (GAE) [23], denoted by A_t, to quantify the effectiveness of the selected action compared with the expected effectiveness of other actions in the same state. Additionally, $r_t(\theta)$ denotes the ratio of the probability of taking action a_t under the new policy (θ) to that under the old policy (θ'). Because four policies exist for four types of BEOL parameters, the ratios of these policies are multiplied together.

$$\begin{aligned} L_{policy}(\theta) &= \mathbb{E}_t\Big[min\big(r_t(\theta)\cdot A_t,\ clip(r_t(\theta), 1{-}\epsilon, 1{+}\epsilon)\cdot A_t\big)\Big] \\ r_t(\theta) &= \frac{\pi_1(a_t|s_t;\theta)}{\pi_1(a_t|s_t;\theta')} \times \frac{\pi_2(a_t|s_t;\theta)}{\pi_2(a_t|s_t;\theta')} \times \frac{\pi_3(a_t|s_t;\theta)}{\pi_3(a_t|s_t;\theta')} \times \frac{\pi_4(a_t|s_t;\theta)}{\pi_4(a_t|s_t;\theta')} \end{aligned} \tag{5}$$

(ii) The value loss updates the value network to estimate the expected return from each state. We employ the Huber loss function, which measures the error between the predicted and actual rewards and is less sensitive to outliers compared with the mean squared error function. In Equation 6, $V(s_t)$ is an estimated value, and V_t^{target} is the actual discounted return observed at step t.

$$L_{value}(\phi) = Huber\big((V(s_t;\phi), V_t^{target}\big) \tag{6}$$

(iii) The entropy bonus promotes exploration by updating the policy network to be more stochastic (Equation 7).

$$L_{entropy}(\theta) = \sum \pi(a_t|s_t;\theta) \cdot log\pi(a_t|s_t;\theta) \tag{7}$$

2) Training strategies: We employ several strategies for efficient training of our CTRL model.

• Data normalization is a crucial process because it ensures training stability and enhances generalization across diverse environments. First, we apply batch normalization to the state features to address the varying scales of each feature type and reduce the sensitivity of our

TABLE III
CIRCUIT SPECIFICATIONS IMPLEMENTED IN THE BASELINE *N14* AND *N7*

Node	Circuit	Clock (*ps*)	# Nets	# Cells	WL (*mm*)
N14	*AES*	500	14,816	14,254	105.52
	LDPC	1,000	41,233	39,180	796.93
	NOVA	2,000	172,832	167,806	1,806.22
N7	*AES*	500	13,219	12,955	49.65
	LDPC	1,000	53,731	51,768	511.56
	NOVA	2,000	153,986	151,785	948.86

TABLE IV
COMPARISON OF PPA RESULTS AND REWARD EFFICIENCY FOR LOW-POWER AND HIGH-PERFORMANCE SCENARIO IN THE *N14*

Scenario	Circuit	Baseline				Grid search					Bayesian optimization					Ours				
		P_T	*WNS*	*TNS*	*Area*	P_T	*WNS*	*TNS*	*Area*	R_{eff}	P_T	*WNS*	*TNS*	*Area*	R_{eff}	P_T	*WNS*	*TNS*	*Area*	R_{eff}
Low power	*AES*	3.85	-55.2	-4.00	5,911	3.45	-45.0	-2.99	6,181	0.59	3.12	-40.3	-1.45	5,820	1.27	2.95	-2.5	-1.24	5,960	1.48
	LDPC	25.62	-53.4	-39.60	21,656	20.09	-7.8	-0.61	21,008	1.07	19.53	-0.8	-0.06	20,919	1.55	18.14	-0.9	-0.02	20,526	1.98
	NOVA	3.73	-63.7	-23.79	74,400	3.62	-54.0	-6.91	75,103	0.32	3.39	-53.8	-9.68	78,217	0.58	3.31	-28.8	-4.01	74,185	0.99
	Norm avg.	1.00	1.00	1.00	1.00	0.88	0.60	0.35	1.01	1.00	0.83	0.53	0.26	1.00	1.81	**0.79**	**0.35**	**0.16**	**0.98**	**3.09**
High performance	*AES*	3.85	-55.2	-4.00	5,911	3.72	-41.1	-2.00	6,181	0.31	3.23	-35.0	-1.42	6,054	0.43	2.98	-30.6	-1.00	6,012	0.55
	LDPC	25.62	-53.4	-39.60	21,656	20.29	-3.5	-0.02	20,974	0.55	19.43	-0.9	-0.00	20,885	0.76	19.58	0.2	0.00	20,985	0.88
	NOVA	3.73	-63.7	-23.79	74,400	3.74	-43.9	-6.00	74,660	0.33	3.47	-25.8	-2.27	75,484	0.65	3.51	-21.4	-1.12	76,503	0.71
	Norm avg.	1.00	1.00	1.00	1.00	0.92	0.50	1.25	1.01	1.00	0.84	0.35	0.15	**1.00**	1.97	**0.83**	**0.30**	**0.10**	**1.00**	**2.15**

The best results are highlighted in **bold**, and the units of P_T, *WNS*, *TNS*, and *Area* are *mW*, *ps*, *ns*, and μm^2, respectively.

network to specific distributions of the features. Second, we normalize the reward by using the improvement ratio of power, WNS, and TNS to address the disparate units and variability of the metrics. However, this ratio can also fluctuate for different BEOL configurations; thus, we constrain the reward range to between -2 and 2.

• Because the processes of DTCO and physical design are time-consuming, we employ a multifaceted approach in our framework. First, we restrict the routing phase to global routing, as mentioned earlier. Despite this restriction, our environment remains computationally demanding owing to the intrinsic complexities of the physical design process. To expedite the training process, we utilize an action-reward buffer assuming our environment is stationary, which indicates that the same action yields a consistent reward. We store and retrieve rewards for already processed actions, thereby reducing redundant computations. However, because the actual physical design flows are subject to tool noise, we run the entire experiment multiple times to demonstrate the robustness of our results in Section IV. Furthermore, we asynchronously parallelize the P&R tasks within the same batch of trajectories to further minimize the time required for the training process. These optimizations collectively serve to make our *CTRL-B* efficient and practical.

• We optimize the dimension of our network and hyperparameters using 50 iterations of Bayesian optimization [25] to optimize training performance. First, we optimize the network dimension D of the FC layers ranging from 64 to 512. Subsequently, we optimize the hyperparameters, including learning rate α, discount factor γ, GAE lambda λ, clip parameter ϵ, batch size B, number of epochs E, value loss coefficient β, and entropy bonus coefficient η. Finally, we adopt the best configuration: we set D to 256, α to 0.0005, γ to 0.95, λ to 0.95, ϵ to 0.1, B to 20, E to 10, β to 0.5, and η to 0.05.

E. Cross-Technology Transfer Learning

To generalize our *CTRL-B* framework across various technology nodes, we construct our network to process different metal types in parallel. Additionally, we propose a cross-technology transferable RL (CTRL) model to facilitate efficient pathfinding in new nodes. The knowledge acquired from the BEOL configuration pathfinding in the old node (*N14*) is transferred to the pathfinding in the advanced node (*N7*). We employ the progressive neural network (PNN) architecture [16] for our CTRL model to overcome the inherent limitations of conventional fine-tuning methods (e.g., catastrophic forgetting and susceptibility to local optima). In our PNN-based CTRL model, lateral connections are established from the transferred network to the newly introduced task-specific network. The weights of the transferred network are frozen during the training process in the *N7*, ensuring that the knowledge of feature representations acquired in the *N14* is preserved. Concurrently, the new network is trainable (not frozen) to adapt to the conditions of the *N7*. This hybrid architecture of frozen and trainable networks provides a robust framework for effective knowledge transfer while adapting to the unique characteristics of different nodes. Moreover, when new nodes are introduced, the existing network can be progressively expanded with new networks to perform efficient pathfinding in the new nodes. Consequently, our CTRL model ensures scalability and compatibility across multiple technology nodes.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

Our experiments are conducted on a CentOS Linux 7.9 with an AMD EPYC 7402P 24-core CPU at 1.5 *GHz* with 200 *GB* RAM and an Nvidia Titan RTX GPU. We implemented the technology generator in *Python* and the RL agent using the *PyTorch* library and an Adam optimizer [24]. We utilized *Synopsys* EDA tools, including *Grdgenxo* [28] for technology characterization and *IC Compiler II* [29] for P&R. We assessed the efficacy and generalizability of our *CTRL-B* framework through evaluations on three OpenCores circuits [30]. The specifications of these circuits as implemented in the baseline nodes are listed in Table III. We set the clock periods to yield negative timing slacks, thereby clearly demonstrating improvements in timing performance. In the physical design phase, we set design utilization to 50% and floorplan aspect ratio to 1.0.

We repeated the experiments K times to demonstrate the robustness of our *CTRL-B* framework under the EDA tool noise and the inherent randomness of RL in exploration and exploitation. In addition, we defined the expected value of the reward efficiency, R_{eff}, as follows:

$$R_{eff} = \frac{1}{K} \cdot \sum_{k=1}^{K} \Big(\frac{1}{J} \cdot \sum_{j=1}^{J} r_j^{max} \Big), \tag{8}$$

where J is the user-specified maximum number of steps for actual physical design, and r_j^{max} denotes the maximum reward up to the j-th step in the k-th experiment. R_{eff} evaluates how efficiently our RL agent can maximize the rewards within a given timeframe. In our experiments, we set K to 10 and limited J to 500.

B. PPAC Metrics Evaluation

We evaluated the chip PPAC and the R_{eff} to demonstrate the efficacy of our RL-based BEOL configuration pathfinding in the *N14*. The final PPAC results were reported after completing the detailed routing, unlike during the training process, where we calculated the reward after the global routing. If the number of DRVs in the best PPAC result exceeded the [1]threshold, we reported the next-best PPAC outcome. Additionally, to demonstrate the robustness of our *CTRL-B* framework, we established two design scenarios with different major objectives: low-power (LP) and high-performance (HP). The LP scenario primarily aimed to reduce total power consumption, with a secondary focus on minimizing timing slacks. Conversely, the HP scenario prioritized the minimization of timing slacks while considering power efficiency. Each scenario has a different reward function based on Equation 3: for the LP scenario, we set c_p, c_w, and c_t to 5, 0.5, and 0.5, respectively; for the HP scenario, they were set to 0.1, 0.5, and 0.5, respectively. The coefficients were determined considering the scale of the baseline results for power and slacks.

When the maximum training step was set to 2,000, the training for all circuits converged to the maximum reward within 1,300

[1]In our experiments, a high DRV threshold of 600 was set because standard cells in the academic nodes are not as optimized as those in industrial nodes.

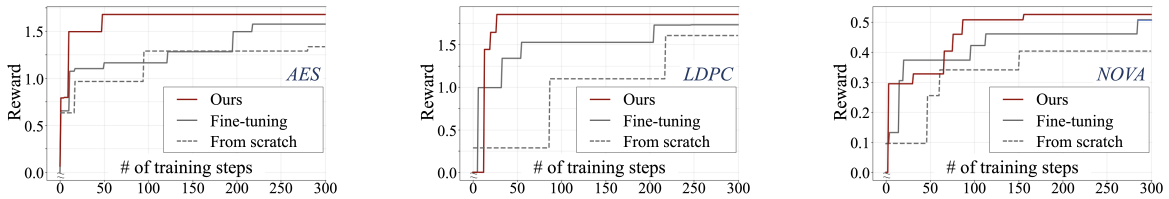


Fig. 5. Comparison of the maximum reward per step in the CTRL using our PNN-based model, fine-tuning technique, and learning from scratch.

training steps. For example, in the *NOVA*, the training converged within 1,200 steps, where approximately 180 actual physical design processes were conducted. For each scenario, we reported the PPA result, including total power consumption (P_T), WNS, TNS, and cell area (*Area*) (Table IV). After conducting the experiments K times, we reported the median of the best PPA result obtained from each run. We compared the PPA result achieved by our *CTRL-B* framework to the PPA result using the baseline *N14*. Further, we compared the PPA and R_{eff} results of our framework to those obtained using the grid search (GS) method [15] and Bayesian optimization (BO), which we implemented using [25]. In the GS method, we constructed 500 BEOL configurations by evenly dividing the parameter spaces of eight parameters in the *N14*. The R_{eff} of the GS method was computed as an average of the resulting rewards achieved by all the configurations. For a fair comparison, the GS and BO methods were also repeated K times, and the median results were reported.

In the LP scenario, we achieved average improvements of 21%, 65%, and 84% over the baseline in P_T, WNS, and TNS, respectively. We reduced the P_T by up to $7.5mW$ without degrading timing or area. In addition, we decreased the cell area by an average of 2% by reducing the insertion of buffer and inverter during the P&R step. Furthermore, in terms of R_{eff}, our RL-based method outperformed the GS and BO methods by 3.09× and 1.71×, respectively. In the HP scenario, average improvements of 17%, 70%, and 90% were achieved for P_T, WNS, and TNS, respectively, and the overhead of cell area was negligible. Furthermore, our method outperformed the GS and BO methods, achieving 2.15× and 1.09× higher R_{eff}, respectively. By embedding the physical design outcomes, which are affected by the BEOL configuration, into the state features, our RL-based method facilitates more efficient pathfinding than the other methods. In this experiment, the computational overhead for RL-related processes, including forward and backpropagation, is negligible, requiring only a few minutes (within 15 *min* during the entire pathfinding process). This is a minor overhead, considering the physical design processes take several hours during the entire pathfinding process (e.g., 6 *hrs* in *AES*, and 42 *hrs* in *NOVA*) and traditional DTCO processes require weeks or months.

C. Cross-Technology Transfer Learning

We evaluated the efficiency of our PNN-based CTRL model in the *N7* by comparing it with a fine-tuning (FT) technique and a model trained from scratch (FS). In this experiment, the maximum training step was limited to 300, and the maximum reward per training step was plotted in the LP scenario (Figure 5). Our CTRL model outperformed the FT and FS methods in all three circuits by achieving the highest rewards within 100 steps. On average, our CTRL model achieved an R_{eff} of 1.27, constituting an improvement of 24% and 48% over the FT and FS methods, respectively. The FT method started with a higher reward in the initial step; however, it was prone to getting stuck in local optima. Conversely, our CTRL model significantly increased the reward at the start of training by referring to the pre-trained model in the *N14*, with a modest memory overhead of less than 5 *MB* compared to the FT method.

V. Conclusion

We proposed a BEOL configuration pathfinding framework using RL and transfer learning. We aimed to improve the power and performance of circuits while considering their area and cost. Several experiments demonstrated the reliability and generalizability of our *CTRL-B* framework. Furthermore, the experimental results can serve as a roadmap for the development of future technology nodes. In the future, we aim to achieve improved chip PPAC by optimizing extended DTCO parameters, including front-end-of-line, middle-end-of-line, and buried power rail in advanced technology nodes.

References

[1] K. Vanstreels, H. Zahedmanesh and M. Gonzalez, "Impact of Back-end-of-line Architecture on Chip-Package-Interaction in Advanced Interconnects," *Microelectronics Reliability* 112 (2020), pp. 1–9.

[2] C. Hou, "1.1 A Smart Design Paradigm for Smart Chips," *Proc. IEEE International Solid-State Circuits Conference*, 2017, pp. 8–13.

[3] N. A. Lanzillo *et al.*, "Impact of line and via resistance on device performance at the 5nm gate all around node and beyond," *Proc. IEEE International Interconnect Technology Conference*, 2018, pp. 70–72.

[4] E. N. Shauly, "Physical, Electrical, and Reliability Considerations for Copper BEOL Layout Design Rules," *Journal of Low Power Electronics and Applications* 8(2) (2018), pp. 1–33.

[5] S. Saini, *Low Power Interconnect Design*, Berlin, Springer, 2015.

[6] M. Darmi *et al.*, "Integrated Circuit Conception: A Wire Optimization Technic Reducing Interconnection Delay in Advanced Technology Nodes," *Electronics* 6(4), 78 (2017).

[7] V. Huang *et al.*, "From Interconnect Materials and Processes to Chip Level Performance: Modeling and Design for Conventional and Exploratory Concepts," *Proc. IEEE International Electron Devices Meeting*, 2020, pp. 32.6.1–32.6.4.

[8] I. Ciofi *et al.*, "Impact of wire geometry on interconnect RC and circuit delay," *IEEE Trans. on Electron Devices* 63(6) (2016), pp. 2488–2496.

[9] D. E. Shim and A. Naeemi, "Balancing Interconnect Resistance and Capacitance at the Advanced Technology Nodes based on Full Chip Analysis," *Proc. IEEE International Interconnect Technology Conference*, 2022, pp. 64–66.

[10] D. Prasad and A. Naeemi, "Interconnect Design and Technology Optimization for Conventional and Emerging Nanoscale Devices: A Physical Design Perspective," *Proc. IEEE International Electron Devices Meeting*, 2018, pp. 5.1.1–5.1.4.

[11] W.-T. J. Chan *et al.*, "BEOL Stack-Aware Routability Prediction From Placement Using Data Mining Techniques," *Proc. IEEE International Conference on Computer Design*, 2016.

[12] C.-K. Cheng *et al.*, "PROBE2.0: A Systematic Framework for Routability Assessment from Technology to Design in Advanced Nodes," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 41(5) (2022), pp. 1495–1508.

[13] S. Choi *et al.*, "PROBE3.0: A Systematic Framework for Design-Technology Pathfinding with Improved Design Enablement," *arXiv preprint arXiv:2304.13215*.

[14] C.-K. Cheng *et al.*, "Machine Learning Prediction for Design and System Technology Co-Optimization Sensitivity Analysis," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems* 30(8) (2022), pp. 1059–1072.

[15] K. Han *et al.*, "Performance- and Energy-Aware Optimization of BEOL Interconnect Stack Geometry in Advanced Technology Nodes," *Proc. IEEE International Symposium on Quality Electronic Design*, 2017, pp. 104–110.

[16] A. A. Rusu *et al.*, "Progressive Neural Networks," *arXiv preprint arXiv:1606.04671v4* (2022).

[17] V. Melikyan *et al.*, "14nm Educational Design Kit: Capabilities Deployment and Future," *Proc. Small Systems Simulation Symposium*, 2018, pp. 37–41.

[18] L. T. Clark *et al.*, "ASAP7: A 7-nm FinFET Predictive Process Design Kit," *Microelectronics Journal* 53 (2016), pp. 105–115.

[19] X. Dong, J. Zhao and Y. Xie, "Fabrication Cost Analysis and Cost-Aware Design Space Exploration for 3-D ICs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 29(12) (2010), pp. 1959–1972.

[20] J. Lienig *et al.*, *Addressing Reliability in Physical Design. Fundamentals of Layout Design for Electronic Circuits*, Cham, Springer, 2020, pp. 257–302.

[21] V. A. Chhabria *et al.*, "From Global Route to Detailed Route: ML for Fast and Accurate Wire Parasitics and Timing Prediction," *Proc. ACM/IEEE Workshop on Machine Learning for CAD*, 2022, pp. 7–14.

[22] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347v2* (2017).

[23] J. Schulman *et al.*, "High-Dimensional Continuous Control Using Generalized Advantage Estimation," *arXiv preprint arXiv:1506.02438v6* (2018).

[24] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980v9* (2017).

[25] R, Liaw *et al.*, "Tune: A Research Platform for Distributed Model Selection and Training," *arXiv preprint arXiv:1807.05118* (2018).

[26] "IEEE International Roadmap for Devices and Systems (IRDS™) 2022 Edition," *https://irds.ieee.org/editions/2022*.

[27] "International Technology Roadmap for Semiconductors (ITRS2.0)," *http://itrs2.net*.

[28] "Synopsys StarRC User Guide," *http://www.synopsys.com*.

[29] "Synopsys IC Compiler II User Guide," *http://www.synopsys.com*.

[30] "OpenCores: Open Source IP-Cores," *http://www.opencores.org*.