2023 Design, Automation & Test in Europe Conference (DATE 2023)

DTOC: integrating Deep-learning driven Timing Optimization into the state-of-the-art Commercial EDA tool

Kyungjoon Chang*, Jaehoon Ahn* , Heechun Park†, Kyu-Myung Choi‡ and Taewhan Kim*
*‡Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea
†School of Electrical Engineering, Kookmin University, Seoul, Korea
*{kyungjoon0, jhahn97, tkim}@snucad.snu.ac.kr, †phc@kookmin.ac.kr , ‡kmchoi@snu.ac.kr

Abstract—**Recently, deep-learning (DL) models have paid a considerable attention to timing prediction in the placement and routing (P&R) flow. As yet, the DL-based prior works are confined to timing prediction at the time-consuming global routing stage, and very few have addressed the timing prediction problem at the placement, i.e., at the pre-route stage. This is because it is not easy to "accurately" predict various timing parameters at the pre-route stage. Moreover, no work has addressed a seamless link of timing prediction at the pre-route stage to the final timing optimization through making use of commercial P&R tools. In this work, we propose a framework called DTOC, to be used at the pre-route stage for this end. Precisely, the framework is composed of two models: (1) *a DL-driven arc delay and arc output slew prediction model*, performing in two levels: (*level-1*) predicting net resistance (R), net capacitance (C), and arc length (Len), followed by (*level-2*) predicting arc delay and arc output slew from the R/C/Len prediction obtained in (*level-1*); (2) *a timing optimization model*, which uses the inference outcomes in our DL-driven prediction model to enable the commercial P&R tools to calculate the full path delays, setting update timing margins on paths, so that the P&R tools should use more accurate margins on timing optimization. Experimental results show that, by using our DTOC framework during timing optimization in P&R, we improve the pre-route prediction accuracy on arc delay and arc output slew by 20∼26% on average, and improve the WNS, TNS, and the number of timing violation paths by 50∼63% on average.**

I. Introduction

At the pre-route stage, several design optimizations such as gate sizing, buffer insertion, and gate displacement are carried out before performing clock tree synthesis (CTS) and time-consuming net routing process to make a better timing closure and design convergence. One problem in the design optimizations at the pre-route stage is that the net timing data is not available as yet, thus relying on timing prediction. Since an incorrect pre-route timing prediction makes the resulting implementation far away from an optimal one, a considerable effort is made on long design iterations to converge to an optimal implementation.

Recently, deep-learning (DL) models have been proposed to improve the prediction accuracy of timing in P&R flow. The work in [1]–[3] predicted the sign-off timing from the post-route circuit. It predicted sign-off wire delay and slew by taking the relevant data extracted from the design tool [1], sign-off timing tool [2], and signal integrity induced delay increments [3].

On the other hand, a couple of DL approaches have been proposed to predict timing at pre-route stage. The work in [4] divided nets into multiple zones based on net length and fanout, and derived, by using linear regression, an R/C scaling factor that correlated pre- and post-route R and C for each zone. Then, the scaling values were imported to a commercial P&R tool in a look-up table format to perform pre-route design optimizations with updated R/C values, resulting in timing improvement. However, its contribution was *limited to improving R/C correlation* and required an additional DRV-clean-level first run (i.e., reference run) at the beginning to collect the dataset for regression. The work in [5] predicted arc delay and arc output slew from net features extracted at the placement. It also predicted crosstalk [6], i.e., predicting whether the crosstalk effect was active or not for each net. Its inherent limitation is that the model *did not fully incorporate arc characteristics as input features*. For example, it captured potential routing congestion with merely the median and standard deviation of pin locations. In addition, net R and net C values were not included in the prediction consideration. Its path delay calculation method based on PERT traversal [7] is also inaccurate. More importantly, the work *provided no solution to the problem of exploiting the timing prediction outcomes* to any commercial timing optimization tool.

To overcome the limitations of the prior works, we introduce a framework for Deep-learning driven Timing Optimization with Commercial tool, which we named as DTOC. In detail, we devise DL-driven accurate pre-route timing parameter prediction model, and a plug-in timing optimizer that is able to not only fully utilize the prediction outcomes but also to be applicable in the state-of-the-art commercial electronic design automation (EDA) tools.

For our timing prediction model, we are interested in estimating two essential timing parameters, namely *arc delay* (i.e., pin-to-pin delay) and *arc output slew*, from which any commercial timing tool can derive an estimation of full path timing. Precisely, our DL based timing prediction model consists of two levels of prediction: (*level-1*) predicting net resistance (R), net capacitance (C), and arc length (Len) and (*level-2*) predicting arc delay and arc output slew from the R/C/Len prediction obtained in *level-1*. Then, we use the trained DL model to predict arc delay and arc output slew values in the target design, and tune the related design

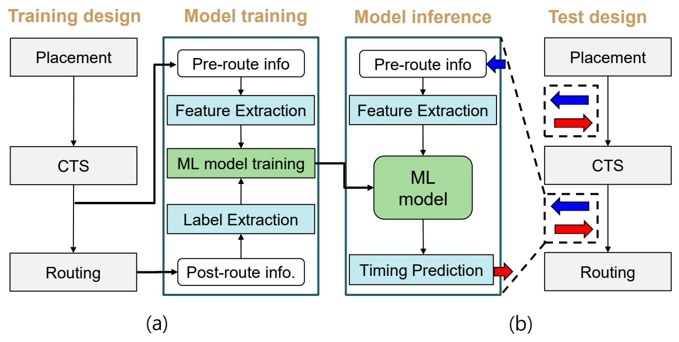


Fig. 1. Flowchart of our DTOC framework. (a) Flow of training our proposed timing parameter prediction model. (b) Flow of our model inferencing and linking to a commercial EDA tool.

parameters in the commercial P&R tools to operate its timing optimization engines with more accurate timing information. Through experiments with benchmark circuits, it is shown that our proposed 2-level timing prediction model outperforms the conventional prediction tools, achieving on average 20∼26% improved accuracy in predicting arc delay and output slew. Furthermore, our timing optimization flow is compatible to any commercial P&R tool, and provides an optimized design layout with better timing closure, achieving on average 62/63/50% improvements on WNS, TNS, and the number of timing violation paths, respectively, which are otherwise unachievable by commercial tools.

Our DTOC framework is charted in Fig. 1, in which Fig. 1(a) shows the training flow of DL driven timing parameter prediction model, while Fig. 1(b) shows the flow of inferencing parameter values using input features (specified by blue arrows) and linking to pre-route stages (specified by red arrows) in a commercial EDA tool. In the following, we discuss, in Sec.II, what timing parameters are commonly used in commercial EDA tools and how they are related to each other. Then, we propose our DL-driven timing parameter prediction model in Sec.III, followed by describing, in Sec.IV, our timing knob for controlling the commercial timing optimization tools. Sec.V shows the experimental results, and Sec.VI concludes this work.

II. Anatomy of Timing Parameters used in Commercial EDA Tools

• **Evaluation dependency among timing parameters**: Fig. 2 shows the evaluation dependency map among the timing parameters commonly used in commercial EDA tools. The ultimate parameter we want to estimate at the pre-route stage is the *path delay*, specified as the pink box in Fig. 2.

Path delay is evaluated from the values of *gate delays* and *arc delays* on the path. The gate delays are taken from the Liberty (*.lib*) file, which is a simple look-up table of the combination of gate input slew and load capacitance values in a non-linear delay model (NLDM) or in a more accurate composite current source (CCS) model. Here, the values of gate input slew correspond to the values of arc output slew, which are predicted using net R, net C, arc input slew, and

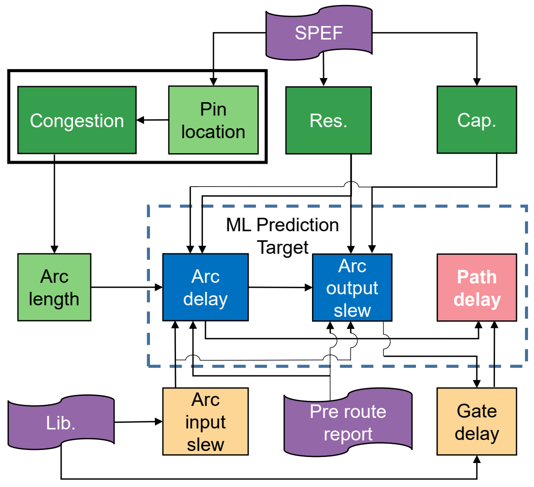


Fig. 2. Evaluation dependency among pre-route timing parameters.

arc delay. On the other hand, the arc delays are evaluated using net R, net C, *pin-to-pin length*, and *arc input slew*. Consequently, to accurately predict the path delays, it is essential to accurately predict arc delays and arc output slews, which in turn requires accurately predicting the values of net R, net C, arc length, and arc input slew, as shown in the evaluation dependency among timing parameters in Fig. 2.

• **Pre-route timing parameter update using commercial tools**: Fig. 3 illustrates how we can re-evaluate a path delay with more accurate prediction on arc delay and arc output slew, and annotate it into timing optimization constraint used in commercial tools.[1] Starting from a path delay shown in Fig. 3(a), which consists of three gate delays (D_1, E_1, F_1) and four arc delays (d_1, e_1, f_1, g_1), the tool's internal delay annotation command (e.g., `set_annotated_delay`) replaces arc delays to the predicted values from the prediction model (i.e., (d_1, e_1, f_1, g_1) $\rightarrow$ (d_2, e_2, f_2, g_2)). Also, internal arc output slew annotation command (e.g., `set_annotated_transition`) changes slew values for all input pins for three gates (i.e., $H_1/p_1 \rightarrow H_1/p_4$, $H_2/p_2 \rightarrow H_2/p_5$, ...), each of which changes the input for the internal gate delay calculation function (e.g., LUT in NLDM), thereby updates the corresponding gate delay (i.e., (D_1, E_1, F_1) $\rightarrow$ (D_2, E_2, F_2)).

Then, with the internal timing update command (e.g., `update_timing` in Fig. 3(b)) , the P&R tool takes all the changes from our annotations into account and updates all path delays with accurate calculations, from which we can extract the updated worst slack value, t_{slk}, on each path. We use an empirically devised margin ratio function $\rho(t_{slk})$,[2] to obtain the new *path margin*, γ', and annotate it into the endpoint flipflop via a path margin annotation command (e.g., `set_path_margin` in Fig. 3(b)). Then, a commercial design optimization tool is ready to re-optimize pre-route

[1]Here, we use timing annotation commands used in Synopsys ICC2. However, the commands can be replaced with compatible ones used in other tools such as Cadence Innovus.

[2]The function $\rho(t_{slk})$ will be described in Sec.IV.

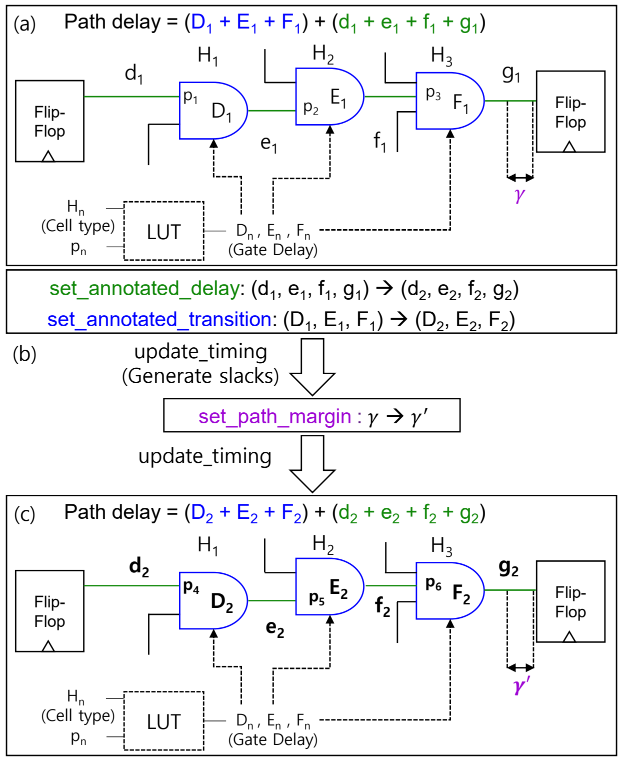

(a)
Path delay = ($D_1 + E_1 + F_1$) + ($d_1 + e_1 + f_1 + g_1$)
Flip-Flop
LUT
H_n (Cell type) p_n
D_n, E_n, F_n (Gate Delay)
γ
set_annotated_delay: (d_1, e_1, f_1, g_1) → (d_2, e_2, f_2, g_2)
set_annotated_transition: (D_1, E_1, F_1) → (D_2, E_2, F_2)
(b)
update_timing (Generate slacks)
set_path_margin : $\gamma \rightarrow \gamma'$
update_timing
(c)
Path delay = ($D_2 + E_2 + F_2$) + ($d_2 + e_2 + f_2 + g_2$)
γ'

Fig. 3. Illustrating the pre-route timing parameter updating process through EDA tool's annotation commands.

timing by referring to the accurately annotated timing values, as shown in Fig. 3(c).

III. The Proposed Timing Prediction Model in Placement

Fig. 4 shows the structure of our proposed DL-based timing prediction model, which is performed in two levels: predicting net R, net C, and arc length (measured in terms of congestion induced arc length increments) by three sub-models in *level-1* (left part), followed by predicting arc delay and output slew by two sub-models in *level-2* (right part). The predicted values of net R, net C, and arc length from three *level-1* sub-models are concatenated to be transformed into input features to predict arc delay and arc output slew by the two *level-2* sub-models, in which the output slew prediction sub-model uses the outcomes of the arc delay prediction sub-model, together with the outcomes of *level-1* sub-models, to predict arc output slew.

A. Input Features

The input features are summarized below. Some features will be elaborated for their corresponding sub-models.

- **Pre-route arc delay, arc output slew:** Their values are taken from the pre-route timing report generated by an existing P&R tool.
- **Arc input slew:** It corresponds to the slew value at cell outputs.
- **Pin-to-pin distance:** It amounts to the Manhattan distance between the two pins.
- **Arc R, τ(RC):** They correspond to an accumulated resistance and Elmore RC delay from arc source to sink.
- **Pre-route net R, net C, net length:** Their values are taken from the pre-route parasitic report generated by an existing placement and routing tool.
- **Fanout:** It indicates the number of net sinks.
- **Net RUDY (Rectangular Uniform wire DensitY [8]), long-range net RUDY [9]:** They correspond to estimated congestion values in a net bounding box.
- **G-cell RUDY map:** It indicates an estimated congestion value on a G-cell.
- **Net density map:** It indicates an estimated track usage value on a G-cell.
- **Pin capacitance location map:** It indicates an accumulated pin capacitance value on a G-cell.
- **Source and sink location map:** It indicates the G-cell locations of arc source and sink.

B. Prediction Models for net R and net C

Each of net R and net C prediction models is constructed with three FC (fully connected) ANN layers, as shown in *level-1* of Fig. 4. The models use four common input features, which are net length, fanout, net RUDY, and long-range net RUDY. Then, the net R prediction model uses the feature of pre-route R and the net C prediction model uses the feature of pre-route C. While net length and fanout values are straightforward, the Net RUDY is computed by:

$$net\_RUDY(i) = \frac{Vol(i)+\sum_{j\in N(i)}[Vol(j)\times OL(i,j)/Area(j)]}{Area(i)}, \quad (1)$$

where $Area(i)$, $Vol(i)$, $N(i)$, and $OL(i,j)$ indicate the bounding box area of net i, wire volume of net i, the overlapped net of net i, and the overlapped area of nets i and j, respectively. Net RUDY includes a measure of potential routing coverage in the bounding box of the target net with its neighboring nets that can also be routed across the box. Long-range net RUDY follows the same computation process, but it includes only the nets longer than a threshold (25um in our setup).

The five input features travel through three FC networks (5→64→16→1) toward the 1-bit output, which corresponds to the label of the post-route net R or net C estimation. We independently train the net R and net C prediction sub-models using the input features to set the weight values in their fully connected neural networks. The trained net R and net C sub-models are then used to train the sub-models in *level-2*.

C. Prediction Model for Congestion

Our congestion prediction model is used to predict the arc length increments incurred by detouring wire in a congested area. We adopt a convolutional neural network (CNN), as shown in Fig. 4, in that routing congestion in a region is highly correlated with the status of neighboring regions.

To extract input features (i.e., input channels), we divide the design layout into G-cells and extract a $5\times W\times H$ sized 3D array, in which W and H represent the width and height of the net bounding box in terms of covered G-cell count. Each of

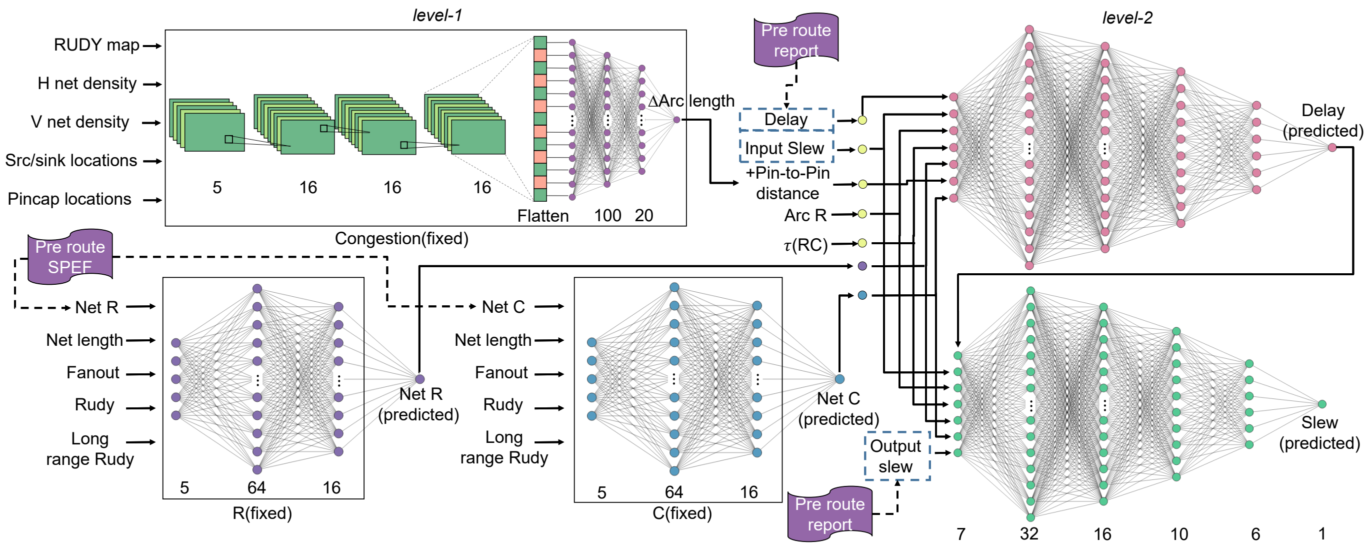
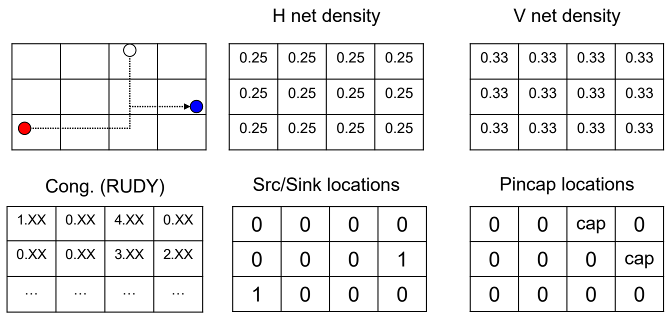


Fig. 4. Structure of our proposed levelized timing prediction model in placement, predicting net R, net C, and arc length by three sub-models in *level-1* (left part), followed by predicting arc delay and output slew by two sub-models in *level-2* (right part).


H net density

0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25

V net density

0.33	0.33	0.33	0.33
0.33	0.33	0.33	0.33
0.33	0.33	0.33	0.33

Cong. (RUDY)

1.XX	0.XX	4.XX	0.XX
0.XX	0.XX	3.XX	2.XX
...	...	...	...

Src/Sink locations

0	0	0	0
0	0	0	1
1	0	0	0

Pincap locations

0	0	cap	0
0	0	0	cap
0	0	0	0

Fig. 5. Example of input feature extraction for a target net to be used in our congestion prediction model.

the 5 input channels represents G-cell RUDY map, horizontal and vertical net density map [10], source/sink location, and pin capacitance location, respectively. Fig. 5 illustrates an example of input feature extraction for a target net.

G-cell RUDY map contains RUDY values for $W \times H$ G-cells. Net density statistically estimates how many nets are routed across a G-cell. Note that net density of a G-cell is the accumulation of the probabilities of all nets crossing the target G-cell. Net density map is decomposed into two different channels (horizontal and vertical) since the horizontal and vertical wires are routed on different metal layers. Source/sink location map indicates whether arc source/sink pins exist in a G-cell. This channel is to differentiate arc characteristics since a net with fanout size of n entails n different arcs. For example, the net in Fig. 5 has two fanout pins (white and blue) from a source pin (red). In this case, the input features of the two arcs (i.e., red→white, red→blue) will have the same values for all other channels except the source/sink location channel that has value of 1 in different sink grids (i.e., (0,2) for white pin and (1,3) for blue pin). Lastly, the pin capacitance location map contains pin capacitance values in the G-cells that the sink pins are located.

Our congestion prediction model consists of three CNN layers (5→16→16→16 channels) and three FC layers (flatten→100→20→1), which follows a conventional CNN-based prediction model structure (e.g., [11]). To enable flattening between CNN - FC layers and speedup the training process, we apply **adaptive average pooling** to the input feature to use a fixed feature size ($5 \times W \times H \rightarrow 5 \times 7 \times 7$ in our setup) and allow parallel batch-wise computation. We achieved 30x speedup in both training and inference with adaptive average pooling with no loss in prediction accuracy. For training, we label each input feature with the *difference* between post-route and pre-route arc length. The trained congestion sub-model is then used to train the arc delay prediction sub-model in *level-2* by adding the predicted value to the original arc length (i.e., pin-to-pin distance).

D. Prediction Models for Arc Delay and Arc Output Slew

While both of our arc delay and arc output slew prediction models share the same structure of five FC networks (7→32→16→10→6→1), their input features are assigned separately according to the effect of each feature on the prediction target.

The arc delay prediction sub-model uses the pre-route arc delay and arc input slew values taken from the pre-route report as a prediction reference. Arc length is updated from pin-to-pin distance by adding predicted increments from the congestion prediction sub-model. Arc R is calculated by tracing the net tree (in SPEF) from source to sink and accumulating the resistances of the segments. τ(RC) is an Elmore delay value produced by converting every wire segment into a π-model and computing the RC delay from source to sink. Net R and net C are provided by pre-trained net R and net C prediction sub-models.

The arc output slew prediction sub-model shares five input features with the arc delay prediction sub-model as shown in Fig. 4. The pre-route arc delay is replaced with the predicted

TABLE I
BENCHMARK INFORMATION

Design	Training designs			Test designs		
	Clk.(ns)/Util.	#Gates	#Nets	Clk.(ns)/Util.	#Gates	#Nets
ecg	0.5 / 70%	115,425	116,338	0.4 / 60%	114,454	115,367
ethernet	0.5 / 60%	46,306	46,306	0.5 / 70%	46,091	46,196
jpeg	0.7 / 70%	243,458	265,187	0.7 / 60%	266,319	288,048
ldpc	0.5 / 80%	44,173	47,620	0.6 / 70%	44,453	47,900
nova	1.1 / 60%	159,279	161,858	1.0 / 70%	158,945	161,519
tate	0.6 / 70%	243,360	244,119	0.45 / 65%	244,297	245,057
vga	0.65 / 70%	72,821	72,910	0.65 / 60%	73,120	73,209
wb_conmax	0.4 / 80%	22,449	23,581	0.4 / 55%	23,068	24,200

value from the arc delay sub-model to improve prediction accuracy, and the pre-route arc output slew value is added to the input feature instead of pin-to-pin distance.

For training, we train the arc delay prediction sub-model first, and then train the output slew prediction sub-model by attaching the predicted arc delay values as an input feature.

IV. TIMING KNOB FOR CONTROLLING COMMERCIAL TIMING OPTIMIZER

As stated in Sec. II, we fully utilize the predicted timing values to control the commercial timing optimizer by varying the path margin values annotated to the endpoint flipflops. We devise a *margin ratio function* $\rho(t_{slk})$ which returns the elaborated path margin annotation ratio according to the worst slack value (t_{slk}).

Fig. 6 shows the margin ratio function where A, B, and C are empirically set by an iterative process of circuit timing optimization and evaluation. It avoids large path margin insertions for the flipflops that have high-positive worst slacks which are unnecessary in terms of timing optimization, and also helps to exclude outliers (e.g., extremely wrong prediction results) from being imported into the design, thereby improves the design quality.

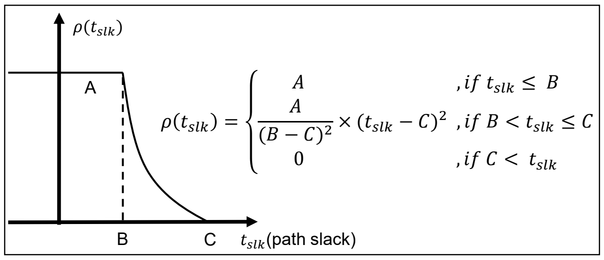


Fig. 6. Our margin ratio function $\rho(t_{slk})$ to be used for setting path margin. The curve indicates that as t_{slk} decreases/increases, a more/less emphasis is placed on optimizing timing on the corresponding path.

V. EXPERIMENTAL RESULTS

• **Experimental setup**: We assess the timing prediction accuracy of our DTOC by comparing the predicted arc delay and arc output slew with the reported pre-route values from a commercial P&R tool (Synopsys ICC2). We implement the open-source benchmark circuits [12] using Nangate 15nm Open Cell Library [13], [14] as PDK and P&R tool to collect training dataset (i.e., input features and corresponding labels of post-route arc delay and arc output slew values) for our pre-route timing prediction model.

To evaluate our model, we construct another set of designs as a test dataset by slightly modifying the major design parameters (clock period and initial chip utilization) and use them to compare prediction accuracy against the pre-route estimations from the tool. Details about our benchmark circuits are listed in Table I. Our DTOC prediction model is implemented in Pytorch, and trained with one NVIDIA GeForce RTX 2080Ti GPU. Likewise, our DTOC timing optimization flow is implemented in C++ and Python, generating TCL script for the P&R tool to annotate predicted delays and output slews after CTS stage (i.e., `clock_opt`), update timing information, and assign path margins. For the margin ratio function $\rho(t_{slk})$, we set A, B, and C to 1.0, 0.1×(clock period), and 0.5×(clock period), respectively.

TABLE II
COMPARISON BETWEEN SYNOPSYS ICC2 TIMING REPORTS AND OUR DTOC SUBMODEL OUTPUTS

Design	Net R. MSE.		Net C. MSE.		Arc length. MSE.	
	ICC2	DTOC	ICC2	DTOC	ICC2	DTOC
ecg	0.0265	0.0126 (52.51%)	0.0921	0.0772 (16.14%)	0.7116	0.6248 (12.2%)
ethernet	0.0356	0.0134 (62.27%)	0.1215	0.0850 (30.01%)	2.4489	2.3305 (4.83%)
jpeg	0.0219	0.0118 (46.32%)	0.0657	0.0649 (1.11%)	1.0266	0.9341 (9.01%)
ldpc	0.1646	0.1121 (31.91%)	0.4535	0.3593 (20.78%)	4.3558	4.3149 (0.94%)
nova	0.0493	0.0301 (38.96%)	0.1301	0.1207 (7.23%)	3.7238	3.5731 (4.05%)
tate	0.0671	0.0495 (26.20%)	0.2474	0.2031 (17.90%)	0.7059	0.6508 (7.81%)
vga	0.0812	0.0541 (33.37%)	0.2656	0.2133 (19.67%)	2.7032	2.5874 (4.28%)
wb_conmax	0.0195	0.0060 (69.11%)	0.0525	0.0426 (18.74%)	1.9578	1.8487 (5.57%)
avg		45.08%		16.44%		6.08%

TABLE III
COMPARISON BETWEEN SYNOPSYS ICC2 TIMING REPORTS AND OUR DTOC PREDICTION ON ARC DELAY AND ARC OUTPUT SLEW

Design	**Arc delay**, MSE.			**Arc output slew**, MSE.		
	ICC2	DTOC		ICC2	DTOC	
		Seen	Unseen		Seen	Unseen
ecg	0.9774	0.7086 (27.50%)	0.7918 (18.99%)	7.4999	5.9955 (20.05%)	6.4289 (14.28%)
ethernet	4.1089	2.3632 (42.48%)	2.4300 (40.86%)	32.1749	22.4541 (30.21%)	22.3837 (30.43%)
jpeg	4.6331	2.9530 (36.26%)	3.5958 (22.38%)	36.7542	23.6061 (35.77%)	26.8129 (32.14%)
ldpc	16.5036	11.7751 (28.65%)	11.7275 (28.93%)	131.7677	83.7896 (36.41%)	89.4116 (32.14%)
nova	10.5605	8.9057 (15.67%)	8.9329 (15.41%)	83.8984	66.2062 (21.08%)	70.5714 (15.88%)
tate	3.2182	2.7407 (14.83%)	2.9792 (7.42%)	21.9702	19.1128 (13.00%)	20.7556 (5.52%)
vga	32.0595	30.5049 (4.84%)	31.0291 (3.21%)	248.2665	216.7199 (12.70%)	235.3165 (5.21%)
wb_conmax	3.0293	1.7915 (40.86%)	2.2023 (27.30%)	21.9280	12.106 (44.79%)	14.9558 (31.79%)
Avg.		26.38%	20.56%		26.75%	20.28%

• **Assessing DTOC timing prediction accuracy**: We compare our timing prediction results of arc delay and output slew values with the estimated values produced by the P&R tool, in terms of MSE (mean square error) to the post-route measured values. Smaller MSE means that the prediction is close to the target values, thus much more accurate.

TABLE IV
COMPARISON BETWEEN SYNOPSYS ICC2 TIMING REPORTS AND OUR PREDICTION MODEL TIMING OPTIMIZATION RESULTS

Test Bench	ICC2			DTOC		
	WNS	TNS	#VP	WNS (impr.)	TNS (impr.)	#VP (impr.)
ecg	-29.44	-837.38	66	0.84 (102.85%)	0 (100.00%)	0 (100.00%)
ethernet	-78.29	-1581.14	49	-44.09 (43.68%)	-1091.24 (30.98%)	62 (-26.53%)
jpeg	-72.08	-1206.85	56	-61.93 (14.08%)	-655.23 (45.71%)	36 (35.71%)
ldpc	-128.78	-977.67	19	-80.70 (37.33%)	-885.42 (9.44%)	23 (-21.05%)
nova	-75.27	-6601.72	179	-21.39 (71.58%)	-251.76 (96.19%)	17 (90.50%)
tate	-82.38	-9079.31	349	-70.18 (14.81%)	-6661.54 (26.63%)	241 (30.95%)
vga	-62.64	-2552.08	135	-5.71 (90.88%)	-11.70 (99.54%)	3 (97.78%)
wb_conmax	-13.41	-49.45	5	3.83 (128.56%)	0 (100.00%)	0 (100.00%)
avg				62.97%	63.56%	50.92%

Table II shows a comparison of MSE of the predictions by our three sub-models (i.e., net R, net C, and arc length) in *level-1*. Our sub-models show higher MSE reduction rates for all target values, especially on net R and net C prediction. These predicted values serve as key input features to the *level-2* sub-models for accurate arc delay and arc output slew prediction.

Table III shows an MSE comparison of the predictions by our two sub-models (i.e., arc delay and arc output slew) in *level-2*. We use training designs to train the two sub-models and compute the MSE of each test design listed in Table I. We compare three different predictors: commercial P&R tool (Synopsys ICC2); our prediction model trained with the 'seen' dataset; and our prediction model trained with the 'unseen' dataset. The 'seen' dataset includes arcs in all 8 training designs, while the 'unseen' dataset has arcs from 7 training designs except the target design we want to test. Our model achieves an average of 20$\sim$26% improved accuracy on predicting arc delay and 20$\sim$26% improvements on predicting arc output slew. The table shows that our DTOC prediction model shows high accuracy on the 'unseen' designs as well, which verifies that our model can also be applicable for the designs that are not considered during the training.

The runtime for the full training process of our model is 22 hours, which includes 6.5 hours for the congestion prediction sub-model (in parallel with 0.5 hours for net R and net C prediction sub-models), 7.5 hours for the arc delay prediction sub-model and 8 hours for the output slew prediction sub-model.

• **Assessing the effectiveness of DTOC timing optimization flow**: Lastly, Table IV shows comparisons of WNS, TNS, and timing-violated path count (#VP) for the circuits produced by ICC2 and our DTOC, in which we used the test designs in Table I. In summary, our DTOC achieves an average of 62%, 63%, 50% improvement on WNS, TNS, #VP respectively over those of ICC2, which clearly implies that timing convergence by our DTOC is very effective.

VI. CONCLUSION

In this work, we proposed a timing optimization framework called DTOC, which consists of DL-driven pre-route timing prediction model and a plug-in timing optimization integrated into the state-of-the-art commercial EDA tool. For our timing prediction model, we focused on estimating arc delay and arc output slew, and constructed the model with comprehensive input features and two levels of prediction for high accuracy. Then, we fully utilized the predicted timing values by annotating them into the commercial EDA tool and elaborating the constraints for the timing optimization. Through experiments, it was confirmed that our DTOC prediction model outperformed the commercial P&R tool, achieving 20$\sim$26% improved accuracy in predicting arc delay and arc output slew even for unseen designs. It was also confirmed that our DTOC timing optimization flow was very effective, showing significant improvement in timing metrics, e.g., reducing WNS, TNS, and #VP on average by 62%, 63%, and 50%, respectively.

ACKNOWLEDGMENT

This work was supported in part by Samsung Electronics Company, Ltd. under IO201216-08205-01, IO230102- 04421-01, and IO221227-04376-01, in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MEST) under 2021-R1A2C2008864, 2021-R1I1A1A01049399, in part by the Institute of Information and communications Technology Planning and Evaluation (IITP) grant funded by Korea government (MSIT) under 2021-0-00754, Software Systems for AI Semiconductor Design), and in part by the BK21 Four Program of the Education and Research Program for Future ICT Pioneers, Seoul National University. The EDA tool was supported by the IC Design Education.

REFERENCES

[1] A. B. Kahng *et al.*, "Learning-based approximation of interconnect delay and slew in signoff timing tools," in *SLIP*, 2013.

[2] S.-S. Han *et al.*, "A deep learning methodology to proliferate golden signoff timing," in *DATE*, 2014.

[3] A. B. Kahng, M. Luo, and S. Nath, "Si for free: machine learning of interconnect coupling delay and transition effects," in *SLIP*, 2015.

[4] L. Jianfeng *et al.*, "A machine learning based p&r flow to enhance pre-route vs post-route r/c correlation," in *DAC*, 2020.

[5] E. C. Barboza *et al.*, "Machine learning-based pre-routing timing prediction with reduced pessimism," in *DAC*, 2019.

[6] R. Liang *et al.*, "Routing-free crosstalk prediction," in *ICCAD*, 2020.

[7] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," in *ICCAD*, 2003.

[8] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *DATE*, 2007.

[9] Z. Xie *et al.*, "Routenet: Routability prediction for mixed-size designs using convolutional neural network," in *ICCAD*, 2018.

[10] J. Chen *et al.*, "Pros: A plug-in for routability optimization applied in the state-of-the-art commercial eda tool using deep learning," in *ICCAD*, 2020.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[12] Oliscience, "Opencores," 1999. [Online]. Available: https://opencores.org

[13] "Nangate freepdk15 open cell library." [Online]. Available: http://www.nangate.com/?page id=2328.

[14] K. Bhanushali and W. R. Davis, "Freepdk15: An open-source predictive process design kit for 15nm finfet technology," in *ISPD*, 2015.