2023 IEEE 41st International Conference on Computer Design (ICCD) | 979-8-3503-4291-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCD58817.2023.00059

AiMap: Learning to Improve Technology Mapping for ASICs via Delay Prediction

Junfeng Liu*, Liwei Ni†, Xingquan Li†, Min Zhou‡, Lei Chen‡, Xing Li‡, Qinghua Zhao*, Shuai Ma*
*SKLSDE Lab, Beihang University, Beijing, China
†Peng Cheng National Laboratory, Shenzhen, China
‡Huawei Noah's Ark Lab, Shenzhen, China
{liujunfeng,zhaoqh,mashuai}@buaa.edu.cn; {nlwmode, fzulxq}@gmail.com; {zhoumin27, lc.leichen, li.xing2}@huawei.com

***Abstract*—Technology mapping is an essential process in the EDA flow which aims to find an optimal implementation of a logic network from a technology library. In ASIC designs, the estimated cell delay *w.r.t.* the cut has a significant impact on both area and delay of the mapped network. In this work, we first propose formulating cell delay estimation as a regression learning task by incorporating multiple perspective features, such as the structure of logic networks and non-linear cell delays, to guide the mapper search. We design a learning model that incorporates a customized attention mechanism to be aware of the pin delay and jointly learns the hierarchy between the logic network and library, with the help of proposed parameterizable strategies to generate learning labels. Experimental results show that our proposed method noticeably improves area by 12% and delay by 1%, compared with ABC.**

***Index Terms*—Technology Mapping, ASIC Design, Timing Analysis, Machine Learning**

I. Introduction

Technology mapping is an essential process in the Electronic Design Automation (EDA) flow for digital systems, *e.g.,* ASICs, and FPGAs. It transforms a technology-independent logical network into a circuit implemented by primitive gates chosen from a technology library, with its objective to achieve better performance, power, and area (PPA) [1].

The approaches for technology mapping in both ASICs and FPGAs have witnessed a shift from tree-based mapping [2] to cut-based mapping [1], [3]–[5], as cut-based methods enable better exploration of different logic structures and optimization with the scalability to large-scale designs.

One important distinction between ASIC and FPGA technology mapping lies in the fact that the former maps the logic network to a predefined set of standard cells, while the latter maps it to lookup-tables. Thereby, taking into account the properties of standard cells, works for ASIC technology mapping mainly focus on the two main categories: (1) supergate generation to mitigate the structural bias [6], [7], and (2) accurate delay estimation of the cells (supergates) as their delays are unknown before mapping [1], [8], [9]. For the first category, studies by [6], [7] construct the supergate library by combining gates from the standard-cell library which enables matching with different logic graph structures. For the second category, studies by [1], [9] present load-independent timing models that employ simple gain-based delay models to approximate the delay by assuming all gates with the same input slew and output load. The work proposed by [8] introduces a load-dependent timing model which approximates the loads by utilizing fanout estimation to account for the impacts of loads on supergate delay.

However, the estimation delay of the supergate *w.r.t.* the cut has a significant impact on both area and delay (Quality-of-Result, QoR) of the mapped network. All existing cut-based methods are performed following the dynamic programming (DP) paradigm, where the estimated supergate delay *w.r.t.* the cut directly guides the search direction of DP. Unfortunately, all these approaches to estimating the supergate delay *w.r.t.* the cut involve too many approximate simplifications, which are far from the accurate delay. Thereby, this imposes limitations on achieving better QoR.

In fact, the node-cut-supergate tuples of the logic network have already partially provided the graph structure of the mapped network (*e.g.,* node/cut fanouts, cut structure) and the non-linear delay behavior of the supergates. These structural features of the logic network and the physical features of supergates have partially revealed the input slew and output capacitance, thereby contributing to the delay estimation.

In this work, we formulate the estimation of supergate delay *w.r.t.* the cut as a regression learning task to guide the mapper search. The main contributions are listed as follows:
(1) We first convert it as a regression learning task and present a novel data-driven framework by incorporating multiple features related to the supergate delay, to guide the mapper search.
(2) We design the learning model to be aware of the pin delay and jointly learn cut-node and cut-supergate features, based on our parameterizable strategies for learning label generation.
(3) Experimental results from a wide range of benchmarks show that AiMap noticeably improves area by 12% and delay by 1%, and improve area by 14% with a 9% delay penalty compared with ABC [1] and SLAP [5], respectively.

II. Learning Label Generation

In order to generate supergate delays under the better circuit delay as learning labels to facilitate the capability of learning

This work is supported in part by NSF of China under Grant 61925203, U22B2021, and Major Key Project of PCL (No. PCL2023AS2-3). For any correspondence, please refer to Shuai Ma.

2576-6996/23/$31.00 ©2023 IEEE
DOI 10.1109/ICCD58817.2023.00059

models, we propose three parameterizable strategies from the perspectives of load-independent, load-dependent supergate delay estimation, and cut sample.

Strategy 1: Load-Independent Supergate Delay Estimation. We extend the load-independent supergate delay estimation by considering the impact of input slews. ABC provides a load-independent linear estimation by providing a factor to measure the gain of pre-unit load, referred as to $\mathrm{d}(g)$. However, it fails to model the non-linear behavior of the input slew.

Hence, we model a linear delay estimation $\mathrm{d}_s(g)$ when input slew changes, similar to $\mathrm{d}(g)$:

$$\begin{aligned} \mathrm{d}_s(g) &= \mathtt{LD}_s \times \mathtt{Ga}_s + \mathtt{P}_s \\ \bar{\mathrm{d}}(g) &= \alpha \cdot \mathrm{d}(g) + (1-\alpha) \cdot \mathrm{d}_s(g) \end{aligned} \tag{1}$$

where $\mathtt{P}_s$ is the load-independent parasitic delay from the view of slew, $\mathtt{LD}_s$ is the induced delay per unit slew, and $\mathtt{Ga}_s$ is a pre-defined gain factor for slew. Further, we assign weights α and $1-\alpha$ to $\mathrm{d}(g)$ and $\mathrm{d}_s(g)$, respectively, to accurately access the delay sensitivity of various supergates towards slews and loads. Finally, our load-independent supergate delay estimation $\bar{\mathrm{d}}(g)$ is obtained.

Strategy 2: Load-dependent Supergate Delay Estimation. In order to enhance the accuracy of delay estimation by incorporating the impact of load, we propose a parameterizable strategy that takes into account the number of supergate fanouts *w.r.t.* the cut.

Intuitively, the load of a supergate is expected to be positively correlated with the number of its fanouts. However, during the dynamic programming from PIs to POs, we are unable to calculate the fanout of the current matching supergate in advance. We approximate the fanout of the cut's root node as the fanout of the supergate since a larger fanout of the cut root node usually implies a higher fanout of the supergate. Thus, the delay $\bar{\mathrm{d}}(g,l)$ of supergate g *w.r.t.* the cut under output capacitive load l is defined as follows:

$$\bar{\mathrm{d}}(g,l) = \bar{\mathrm{d}}(g) \cdot (\beta \cdot \gamma \cdot \log \mathtt{RF} + (1-\beta) \cdot \tau \cdot \log \mathtt{LF}) \tag{2}$$

where $\bar{\mathrm{d}}(g)$ is our load-independent delay estimation, $\mathtt{RF}$ and $\mathtt{LF}$ are the fanout of the root nodes and leaf nodes of the cut, respectively. The parameters γ and τ scales $\log \mathtt{RF}$ and $\log \mathtt{LF}$ to the same scale, and β, $(1-\beta)$ weight the influence of the root nodes and leaf nodes on the output load. That is the term $(1-\beta) \cdot \tau \cdot \log \mathtt{LF}$ in Eq. (2) can be interpreted as a penalty term associated with the area. Since the mapped network often duplicates leaf nodes when their quantity is more than 1, this duplication contributes to an increase in the overall area of the circuit. Hence, our load-dependent delay estimation formulated incorporates the consideration of area impact, and balances the area and delay of the mapped circuit.

Benefiting from the load-dependent delay estimation, assuming the arrival time of all cut leaves is already given, we can update the arrival time $\mathrm{Arr}(v)$ of the node v:

$$\mathrm{Arr}(v) = \max_{\forall g \in \mathrm{CutLeaves}(v)} \{\mathrm{Arr}(g) + \bar{\mathrm{d}}(g,l)\} \tag{3}$$

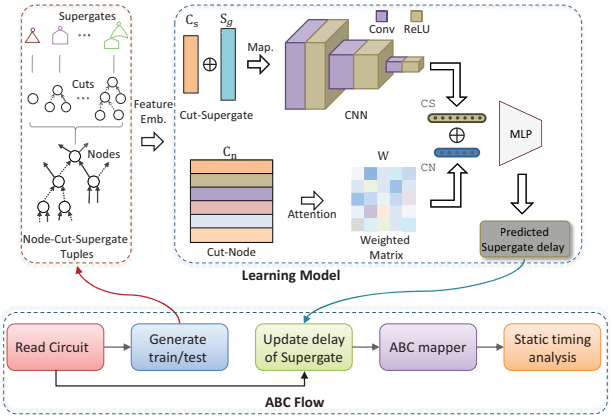

Fig. 1. Framework overview of our proposed AiMap. The red arrow stands for the data flow in the training and testing phases and the green arrow only denotes that in the testing. "Feature Emb." refers to "Feature Embedder".

Strategy 3: Cut Sample. Due to the sophisticated estimation of supergate delay, increasing the number of cuts in a node can not directly lead to improvements in the area or delay optimization, and often results in settling for a suboptimal solution [5]. This insight inspires we can enhance the sensitivity of the quality of results to the cut size by filtering the cuts of each node. Specifically, we can randomly sample a relatively smaller number of cuts (with parameter r) as priority cuts for each node in this strategy. In practice, for the purpose of reproducibility of the best QoR, we indeed use five ranking criteria (with parameter t) to sort the candidate cuts to replace the random shuffle manner.

Finally, we combine three strategies to generate supergate delays as learning labels under the best circuit delay.

III. Approach–AiMap

AiMap framework overview is presented in Fig. 1, where the mapping process of ABC is guided by the predicted supergate delay derived from the learning model.

A. Feature Embedder

Given an AND-Inverter-Graph structurally represented as a DAG, the candidate cuts for each node of AIG, and the candidate supergates matched with each cut, we consider how to explore and encode the fine-grained features of the node-cut-supergate tuples that contribute to predicting the supergates delay *w.r.t.* the cut including node/cut fanouts, cut structure and non-linear delay behavior of supergates.

Node Embedding. The node feature extraction in AiMap mostly follows the prior work [5] that captures the node of AIG structural information from the node itself and from its two children. From the considerations of model efficiency, we opted not to employ graph representation techniques, *e.g.,* GNNs [10], and instead concentrated on the inherent features related to the supergate delay of the AIG itself. Specifically, the node u itself contains 4 features, *i.e.,* level of u, fanout of u, whether an outgoing edge of u has an inverter or not, and

the relative level of u. The features of two children u_1 and u_2 only include the first three features of u.

As mentioned, fanout and level of the node are relatively important as they potentially reveal the characteristics of output load. Thus, we encode the fanout and level of u, u_1, and u_2 into learnable embeddings (with 4 dimensions in practice) based on their values. That is, after normalizing the remaining 4 features in node features, the embedding $\mathbf{h}_u \in \mathbb{R}^{1\times 28}$ of node u is obtained by appending these together.

Cut Embedding. The features related to supergate delay of *k-feasible* cuts are also extracted from the fine-grained, where k is typically equal to 5 in the mapper of ABC. We collect 19 features for each cut from the cut-node and cut-structure aspects. In terms of cut-node, it includes 6 nodes as the features, *i.e.,* the root node of the cut and its 5 leaves nodes. These features are stacked to create a representation $\mathbf{C}_n \in \mathbb{R}^{6\times 28}$ by assembling each node embeddings $\mathbf{h}_u \in \mathbb{R}^{1\times 28}$. In terms of cut structure, it contains 13 features, *i.e.,* (i) the root fanouts, (ii) the cut leaf number and the cut volume for the cut itself and its two parents, (iii) the max, min, and gap of the cut levels and fanouts, respectively. Similar to node embedding, the feature of root fanout in the cut-structure view is also encoded into a 4-dimensional learnable embedding. The remaining 12 features of the cut-structure view are further normalized and the representation is denoted as: $\mathbf{C}_s \in \mathbb{R}^{1\times 16}$. In conclusion, the cut embedding is a combination of the cut-node $\mathbf{C}_n \in \mathbb{R}^{6\times 28}$ and cut-structure $\mathbf{C}_s \in \mathbb{R}^{1\times 16}$.

Supergate Embedding. We extract 60 features associated with the mapped supergate delay for each supergate from an open-source ASAP 7nm PDK standard cell library [11] and ABC. Below, we provide a brief introduction to these 60 features. They contain (i) the basic descriptions of the standard cell, *e.g.,* the area, leakage power, and the number of inputs/outputs, (ii) features related to delay on each pin, *e.g.,* the estimation of load-independent delay under different gain factors, and (iii) the overall features of the supergate, *e.g.,* max and sum delay on all pins. Thereby, the supergate embedding $\mathbf{S}_g \in \mathbb{R}^{1\times 60}$ is obtained followed by the normalization.

B. Learning Supergate Delay

Taking advantage of the embedding of node-cut-supergate tuples and the generated supergate delay labels under the better circuit delay, we design the regression model to capture two inherent characteristics of supergates: (1) the varying contributions of different pins to the estimation of supergate delay and (2) the hierarchy of node-cut-supergate tuples.

Pin-delay-aware Attention based on Cut-Node. Due to various factors such as fanout, and load capacitance of the supergate, different pins should be received different attention to estimating the supergate delay. Essentially, the six-dimension features of cut-node embedding $\mathbf{C}_n \in \mathbb{R}^{6\times 28}$ corresponds to the representation of one output and five input pins, such as fanout and other graph structure features. Thus, we propose the following attention mechanism to learn different weights of pins to be adhering the varying contributions of different pins on the supergate delay, which is guided by a better overall circuit delay, as shown in Fig. 1.

First, a global context embedding $\mathbf{K} \in \mathbb{R}^{1\times 28}$ is computed, with an average of cut-node embeddings $\mathbf{C}_n$ followed by a nonlinear transformation:

$$\begin{aligned} \mathbf{K} &= \tanh\big(\text{mean}(\mathbf{C}_n)\mathbf{W}\big) \\ \texttt{CN} &= \sigma\big(\mathbf{K}\mathbf{C}_n^\top \mathbf{C}_n\big) \end{aligned} \tag{4}$$

where $\mathbf{W} \in \mathbb{R}^{28\times 28}$ is a learnable weight matrix. Second, based on the context $\mathbf{K}$, the pin-delay-aware supergate representation $\texttt{CN} \in \mathbb{R}^{1\times 6}$ is obtained by performing weighted summation of the cut-node embeddings based on the aggregation coefficients $\mathbf{K}\mathbf{C}_n^\top$, where σ is the sigmoid function. Eq. (4) implies that the pin similar to the global supergate context should receive higher attention weights.

Cut-Supergate Fusion. The cut-structure view embedding $\mathbf{C}_s$ provides the structural features potentially related to the supergate delay, *e.g.,* root fanout of the cut, while the supergate embedding $\mathbf{S}_g$ provides the physical characteristic of the standard cell directly related to the supergate delay. Thus, we utilize a Convolutional Neural Network (CNN) to learn and exploit the strengths of structural and physical features' correlation for supergate delay prediction, as shown in Fig. 1.

Specifically, the cut-structure embedding $\mathbf{C}_s$ and the supergate embedding $\mathbf{S}_g$ are first concatenated followed by a linear feature mapping transformation. Further, the transformed features are then fed to our stack CNN model and pass through the flatten layer, yielding fused cut-supergate embedding $\texttt{CS} \in \mathbb{R}^{1\times 162}$.

Delay Prediction. Finally, the embedding for the supergate delay estimation is obtained by concatenating the cut-node CN and cut-supergate CS embeddings. A multi-layer perceptron (MLP) regression model is employed to gradually reduce the concatenated embedding for predicting the supergate delay *w.r.t.* the cut. The training process is guided by minimizing the Mean Squared Error (MSE) loss.

IV. Experiments

A. Experimental Settings

We evaluate the EPFL benchmark [12] and ISCAS'85 benchmark [13] using 15 circuits, as shown in Table I.

The learning labels come from 4 arithmetic circuits (*i.e.,* adder, bar, max, and sin) and 3 control circuits (*i.e.,* i2c, priority, and router) with in total of almost $9,000$ node-cut-supergate tuples, as only the mapped supergates are selected. Each circuit label has undergone 900 iterations of parameter search based on the three strategies we proposed, averagely achieving delay improvements of 20.5%, 26.3%, and 26.6%. Note that, 80% of the tuples are used as the training set, while the remaining 20% constitutes the testing set.

We implement AiMap based on the open source logic synthesis tool ABC [1] and PyTorch.

TABLE I
RESULTS COMPARISONS OF ABC, SLAP, AIMAP. WE HIGHLIGHT OUR METHOD IF IT BEATS THE COUNTERPARTS *w.r.t.* THE AREA AND DELAY.

Circuits	ABC		SLAP		AiMap		AiMap/ABC		AiMap/SLAP	
	Area(μm^2)	Delay(ps)	Area(μm^2)	Delay(ps)	Area(μm^2)	Delay(ps)	Area	Delay	Area	Delay
adder	898.13	3,770.65	1,031.33	3,268.67	1,060.96	3,486.11	1.18	0.92	1.03	1.07
bar	2,680.39	1,114.90	3,083.23	923.82	**2,059.40**	1,058.98	0.77	0.95	0.67	1.15
log2	26,561.26	6,797.77	–	–	**23,330.10**	6,855.81	0.88	1.01	–	–
multiplier	25,458.31	4,649.10	–	–	**20,106.40**	**4,512.38**	0.79	0.97	–	–
sin	5,207.04	3,955.57	5,087.60	3,584.79	**4,503.00**	3,599.18	0.86	0.91	0.89	1.00
sqrt	20,252.20	180,518.05	–	–	**19,918.61**	185,280.97	0.98	1.03	–	–
C6288	2,991.82	1,248.55	3,023.54	1,236.59	**2,479.53**	1,385.40	0.83	1.11	0.82	1.12
C7552	1,978.45	797.54	2,002.01	800.09	**1,758.93**	913.78	0.89	1.15	0.88	1.14
mul32-booth	9,889.44	3,229.54	–	–	**8,041.16**	3,410.47	0.81	1.06	–	–
mul64-booth	39,736.45	6,715.12	–	–	**31,087.36**	7,058.33	0.78	1.05	–	–
64b_mult	52,318.64	7,922.55	–	–	**37,275.11**	9,343.50	0.71	1.18	–	–
aes	18,190.01	656.60	16,489.63	594.64	**15,792.12**	625.55	0.87	0.95	0.96	1.05
cavlc	471.23	294.12	–	–	480.32	**259.77**	1.02	0.88	–	–
int2float	159.56	160.04	–	–	162.13	163.32	1.02	1.02	–	–
ctrl	107.54	134.04	–	–	**107.31**	**101.96**	1.00	0.76	–	–
Geomean	4,290.43	2,098.17	–	–	3,797.73	2,079.21	0.885	0.991	0.865	1.087

B. The Results of Training Model

During the training, we select the training parameters that avoid overfitting as the final inference parameters, (*i.e.,* after 3 epoch iterations), ensuring that both the training loss and the testing loss decrease. The MSE on our training and testing sets are reduced to 7.2 and 9.3, respectively.

C. QoR of Technology Mapping.

We present the achieved QoR improvements *w.r.t.* the area (μm^2) and delay (ps) compared to the heuristic technology-mapping available in ABC [1], [6], and the recent work SLAP [5] which uses supervised learning to filter the candidate cuts. Note that, in this test, area recovery is performed by ABC and AiMap to optimize the area with the same iterations.

1) Comparison against ABC: AiMap noticeably improves area by 12% and delay by 1% on the 15 evaluation circuits, on average. As for the area, AiMap improves 11/15 circuits, achieving up to 29% on 64b_mult, and for most multiplier circuits, it can bring about a 20% improvement. Furthermore, our performance on arithmetic circuits is significantly better than on control logic circuits, primarily due to the fact that the majority (more than 70%) of node-cut-supergate tuples in the training dataset arise from arithmetic circuits. As for the delay, AiMap improves 7/15 circuits, achieving up to 24% on the ctrl logic. The relatively small improvement of delay is primarily attributed to the fact that both ABC and AiMap undergo area recovery in this test. However, area recovery can have unpredictable effects on delay for ASIC mappers due to the resulting changes in the graph structure, which in turn impact the delay of supergates.

2) Comparison against SLAP*:* AiMap significantly improves the area by 14% with a 9% delay penalty on 15 evaluation circuits, on average. The results of the circuit shared by SLAP and AiMap are derived from their original article, while we are unable to replicate the results of other circuits, which are denoted as "–". As for the area, AiMap has surpassed SLAP on almost all circuits, achieving up to 33% on the bar circuit, with the exception of the adder (slightly less than 3%). As for the delay, the results of SLAP are superior to ours due to their exclusive focus on delay optimization.

V. CONCLUSION

In this work, we first formulated the cell delay estimation *w.r.t.* the cut as a regression learning task by incorporating multiple perspective features, to guide the mapper search. Then, we designed a framework AiMap by incorporating pin-delay-aware attention and a cut-supergate feature fusion. AiMap noticeably improves area by 12% and delay by 1%, compared with ABC.

REFERENCES

[1] R. Brayton and A. Mishchenko, "Abc: An academic industrial-strength verification tool," in *CAV*, 2010, pp. 24–40.

[2] K. Keutzer, "Dagon: Technology binding and local optimization by dag matching," in *DAC*, 1987, pp. 341–347.

[3] J. Cong, C. Wu, and Y. Ding, "Cut ranking and pruning: Enabling a general and efficient fpga mapping solution," in *FPGA*, 1999, pp. 29–35.

[4] A. Mishchenko, S. Cho, S. Chatterjee, and R. Brayton, "Combinational and sequential mapping with priority cuts," in *ICCAD*, 2007, pp. 354–361.

[5] W. L. Neto, M. T. Moreira, Y. Li, L. Amarù, C. Yu, and P.-E. Gaillardon, "Slap: a supervised learning approach for priority cuts technology mapping," in *DAC*, 2021, pp. 859–864.

[6] S. Chatterjee, A. Mishchenko, R. K. Brayton, X. Wang, and T. Kam, "Reducing structural bias in technology mapping," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2894–2903, 2006.

[7] M.-C. Wu, A. Q. Dao, and M. P.-H. Lin, "A novel technology mapper for complex universal gates," in *ASPDAC*, 2021, pp. 475–480.

[8] S.-C. Huang and J.-H. R. Jiang, "A dynamic accuracy-refinement approach to timing-driven technology mapping," in *ICCD*, 2008, pp. 538–543.

[9] B. Hu, Y. Watanabe, A. Kondratyev, and M. Marek-Sadowska, "Gain-based technology mapping for discrete-size cell libraries," in *DAC*, 2003, pp. 574–579.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016.

[11] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.

[12] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The epfl combinational benchmark suite," in *IWLS*, no. CONF, 2015.

[13] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the iscas-85 benchmarks: A case study in reverse engineering," *IEEE Design & Test of Computers*, vol. 16, no. 3, pp. 72–80, 1999.