ML-based AIG Timing Prediction to Enhance Logic Optimization

Wenjing Jiang
University of Minnesota
Minneapolis, MN, USA

Jin Yan
Google DeepMind
Mountain View, CA, USA

Sachin S. Sapatnekar
University of Minnesota
Minneapolis, MN, USA

***Abstract*—Traditional logic optimization relies on proxy metrics to approximate post-mapping performance and area, which may not correlate well with post-mapping delay and area. This paper explore a ground-truth-based optimization flow that directly incorporates the post-mapping delay and area during optimization using decision tree-based machine learning models. Results show high prediction accuracy and generalization to unseen designs.**

***Index Terms*—Logic synthesis, PPA, machine learning.**

I. Introduction

As technology scales to smaller nodes, efficient logic optimization becomes increasingly critical to minimize delay, area, and power consumption in circuit designs. In recent years, machine learning (ML) techniques have been applied to predict design metrics during synthesis [1], [2]. A major limitation of these prior approaches is that they utilize proxy metrics, e.g., taking the delay to be proportional to the logical depth of the circuit, and incorporate them into objective functions for optimization. However, proxy delay metrics may not be well correlated with the post-synthesis delay after technology mapping, and can be ineffective in optimizing the design.

Our study finds that integrating the post-mapping delay in the cost function can lead to a design of better quality than the optimization based on proxy metrics, but the runtime can be 20× larger due to the high cost of the mapping step, the procedure is not scalable to large designs. Therefore, we propose an ML-enhanced timing-aware optimization approach that makes the flow 88% faster than the flow integrating post-mapping while maintaining the quality of solution, by incorporating ML inference in the cost calculation.

II. ML-based AIG Timing Prediction

As mentioned above, the correlation between number of AIG levels and the corresponding maximum delay after mapping is imperfect. Fig. 1(a) plots the post-technology-mapping maximum delay against the number of levels, for a number of AIGs associated with a multiplier design. From this experiment, the correlation between maximum delay and the number of AIG levels is only 0.74. Moreover, as seen in the figure, the best delay after mapping does not correspond to an AIG with the smallest number of levels. These observations indicate that proxy metrics may not accurately guide the AIG optimization to achieve the "true" optimal design. These inaccuracies expose another major limitation in the use of AIG-based proxy metrics: two AIGs with same node count and levels may have significantly different delay and area at the post-mapping stage

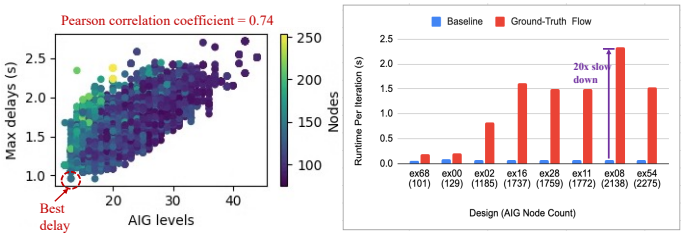

Fig. 1. (a) Scatter plot: post-mapping circuit delay vs. the number of AIG levels. (b) Runtime comparison for one iteration in the original logic optimization flow and the ground-truth-based logic optimization flow. The x-axis shows the name of each design, with its AIG node count in parentheses.

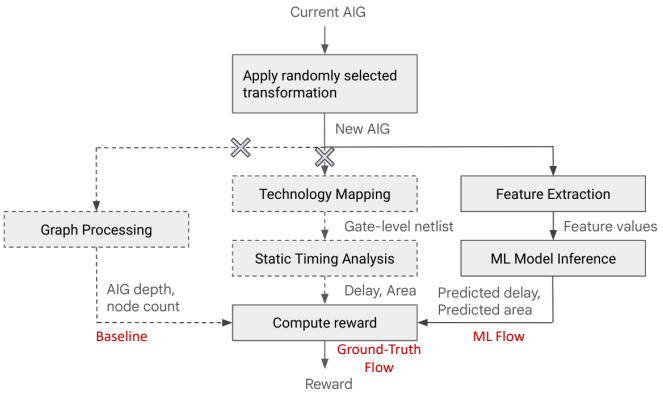

Fig. 2. Three flows for AIG optimization.

but an optimizer using the number of levels as a proxy for delay would treat them as the same. This may cause the search to lose the opportunity to fully explore the design space.

We explore three logic optimization flows illustrated in Fig. 2 to address the problem. From left to right in the figure:
(1) The **baseline flow** uses proxy metrics from a graph processing step. An industry flow that we are familiar with uses 103 combinations of the basic transformations [3] available in ABC [4], from which one combination is selected in each iteration and applied to the the current AIG to get a new AIG.
(2) The **ground-truth-based flow** performs technology mapping, followed by STA, for each new AIG to obtain the post-mapping delay. As shown in Fig. 1(b), such a flow is up to 20× slower, on eight IWLS benchmark designs [5], due to the high cost of mapping and STA, and is impractical for large designs.
(3) Our **proposed ML-based flow** leverages ML models to predict post-mapping delay based on AIG features, avoiding the large runtime cost of frequent STA and mapping.

978-3-9826741-0-0/DATE25/© 2025 EDAA

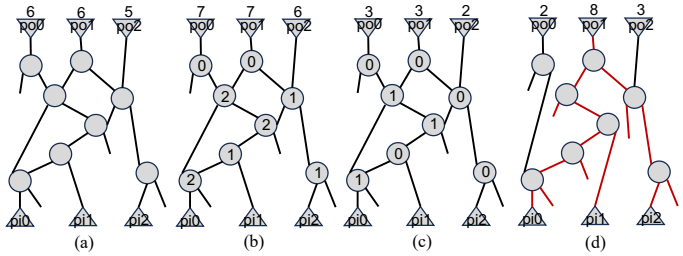

Fig. 3. Feature extraction in an example AIG. (a) The maximum depth is annotated for each PO. (b) The fanout of each node is annotated as the weight and the updated maximum depth is annotated for each PO. (c) The binary-encoded fanout of each node is annotated as the weight and the updated maximum depth is annotated for each PO. (d) The subgraph of PO1 is highlighed in red, and the number of paths is annotated for each PO.

To extract the features used in our approach to drive a decision tree ML model, two primary sources of timing miscorrelation between AIG depth and the maximum delay of the mapped netlist are considered in this work: (a) the path depth change in the mapped netlist compared to the AIG structure, which impacts the number of stages in the critical path, thus affecting the overall path delay; and (b) the fanout change after mapping, in which several nodes in AIG are merged into a large cell, resulting in changes to the fanout of the node driving these cells, which affects the gate delay and contributes to the miscorrelation. The features are extracted from AIG and can be classified into three categories: those associated with the critical path; those related to the fanout distribution; and those related to the structural complexity of the subgraph corresponding to each primary output (PO), reflecting how the topology may affect timing after mapping. Fig. 3 demonstrates how the features are extracted using an example AIG with paths between three primary inputs (PIs) and three POs. Different methods for depth calculation are shown in Fig. 3(a)-(c), which take the nodes between PO and PI, including the node for PI and excluding the PO node (which is a gate output). Fig. 3(d) shows an example of subgraph extraction for PO.

III. Experimental Setup and Results

Our experiments are conducted on eight designs from the IWLS benchmark suite, each from a different functional category. Table I summarizes the number of PIs and POs, the median number of AIG nodes across 40K generated AIGs for each design. Four designs are used for model training, and four for testing to evaluate the ability of the model to be generalized to unseen designs. The experiments are conducted using the simulated annealing (SA) paradigm [6]. Our models can also be integrated into other conventional approaches besides SA.

Table I summarizes the metrics for the ML model and its prediction accuracy. The results show that across all designs, the average prediction error is 4.03% and the average standard deviation is 3.27%, thus demonstrating good overall accuracy.

The three flows in Section II are applied to the eight IWLS benchmarks. Fig. 4 shows the results for the baseline optimization flow, the ground-truth-based flow, and the ML-based flow. Each point on the plot corresponds to an optimal AIG obtained from a specific run of SA logic optimization flow with a certain hyperparameter setting. The curves for the ML-based and

TABLE I
Accuracy of XGBoost model for timing prediction

Design		PI/PO	#Node (Range)	Mean of % Error	Max of %Error	Std. of % Error
Training	**EX00**	16/7	69-189	4.24%	29.87%	3.56%
	EX08	18/5	1828-1448	1.90%	14.05%	1.59%
	EX28	17/7	1296-2222	1.53%	14.41%	1.36%
	EX68	14/7	62-140	4.23%	33.64%	3.48%
Test	**EX02**	18/6	848-1522	5.77%	32.52%	4.86%
	EX11	17/7	1253-2290	5.22%	36.96%	3.90%
	EX16	16/5	1237-2236	4.50%	36.74%	3.55%
	EX54	17/7	1469-3080	4.83%	39.85%	3.87%
				Avg. 4.03%	Max 39.85%	Avg. 3.27%

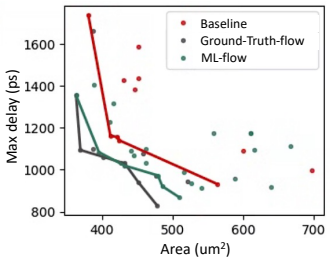

Fig. 4. A comparison of the Pareto-optimal fronts for the delay and area of a test design from the baseline, ground-truth-based, and ML-based flows.

ground-truth-based flows significantly outperform the baseline. However, as shown in Table II, across all testcases, the ML-based flow achieves a significant runtime reduction (80.83% on average, and up to 88.79%) compared to the ground-truth-based flow. This demonstrates that the ML-enhanced logic optimization flow can not only maintain high-quality results but also make substantial improvements on efficiency, which makes it an excellent alternative for larger designs.

Acknowledgment

This work is supported by Google DeepMind. The authors would like to thank X. Xu, D. Ruić, R. B. Apte, V. Nair, G. Rotival, amd T. Sitdikov for helpful discussions and feedback.

References

[1] C. Yu and W. Zhou, "Decision making in synthesis cross technologies using lstms and transfer learning," in *Proc. MLCAD*, pp. 55–60, 2020.
[2] H. Zheng, *et al.*, "LSTP: A logic synthesis timing predictor," in *Proc. ASP-DAC*, pp. 728–733, 2024.
[3] "ABC commands." https://people.eecs.berkeley.edu/~alanmi/abc/abc.rc, 2024.
[4] R. K. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *Proc. CAV*, 2010.
[5] "IWLS 2024 benchmarks." https://github.com/alanminko/iwls2024-ls-contest/tree/main/benchmarks, 2024.
[6] A. Hillier, *et al.*, "Learning to design efficient logic circuits." https://www.iwls.org/iwls2023, 2024.

TABLE II
Runtime for the three flows

Design		Baseline (s)	Ground-Truth-flow Mapping+STA (s)	ML-flow ML Inference (s)
Training	**EX00**	0.0936	0.3028	0.1118 (-63.08%)
	EX08	0.0736	2.3375	0.1967 (-88.79%)
	EX28	0.0712	1.4966	0.1555 (-85.54%)
	EX68	0.0463	0.1961	0.0166 (-74.05%)
Test	**EX02**	0.0670	0.8335	0.1213 (-79.09%)
	EX11	0.0693	1.5044	0.1575 (-85.59%)
	EX16	0.0764	1.6131	0.1617 (-85.91%)
	EX54	0.0766	1.5320	0.1715 (-84.58%)
Avg.				-80.83%
Max				-88.79%