ASAP: Accurate Synthesis Analysis and Prediction with Multi-task Learning

Yikang Ouyang[1] Sicheng Li[2] Dongsheng Zuo[1]
Hanwei Fan[1] Yuzhe Ma[1]

December 21, 2024

[1]The Hong Kong University of Science and Technology (Guangzhou)

[2]Alibaba DAMO Academy

- The complexity and size of modern VLSI keeps soaring.
- It's costly to obtain feedback from running EDA flows, which hinders the expedition of hardware designs.

[1] Silicon Compilers -Version 2.0

- The complexity and size of modern VLSI keeps soaring.
- It's costly to obtain feedback from running EDA flows, which hinders the expedition of hardware designs.

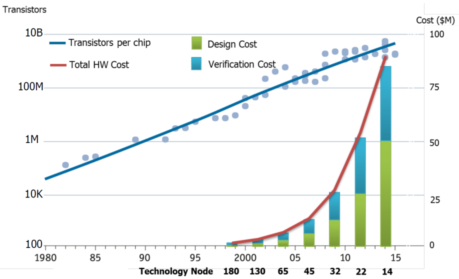

Figure 1: The scaling of VLSI development costs.[1]

[1] Silicon Compilers -Version 2.0

- To facilitate hardware design, many works have been raised to predict the metrics of circuits without launching EDA flows.

M. Nemani and F. N. Najm, “Delay estimation vlsi circuits from a high-level view,” in Proc. DAC, 1998, pp. 591–594.

S. Li et al., “Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures,” in Proc. MICRO, 2009, pp. 469–480.

D. S. Lopera et al., “Early rtl delay prediction using neural networks,” Microprocessors and Microsystems, vol. 94, p. 104671, 2022, S. Roy et al., “A learning bridge from architectural synthesis to physical design for exploring power efficient high-performance adders,” in Proc. ISLPED, 2017, pp. 1–6.

N. Wu et al., “Lostin: Logic optimization via spatio-temporal information with hybrid graph models,” in Proc. ASAP, 2022, pp. 11–18.

- To facilitate hardware design, many works have been raised to predict the metrics of circuits without launching EDA flows.
- These works can be classified into two types:
  - Analytical: Boolean complexity Analysis, design functionality analysis

M. Nemani and F. N. Najm, "Delay estimation vlsi circuits from a high-level view," in Proc. DAC, 1998, pp. 591–594.

S. Li et al., "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in Proc. MICRO, 2009, pp. 469–480.

D. S. Lopera et al., "Early rtl delay prediction using neural networks," Microprocessors and Microsystems, vol. 94, p. 104671, 2022, S. Roy et al., "A learning bridge from architectural synthesis to physical design for exploring power efficient high-performance adders," in Proc. ISLPED, 2017, pp. 1–6.

N. Wu et al., "Lostin: Logic optimization via spatio-temporal information with hybrid graph models," in Proc. ASAP, 2022, pp. 11–18.

- To facilitate hardware design, many works have been raised to predict the metrics of circuits without launching EDA flows.
- These works can be classified into two types:
  - Analytical: Boolean complexity Analysis, design functionality analysis
  - Machine Learning-based: Artificial neural networks, support vector regression, graph neural networks

M. Nemani and F. N. Najm, "Delay estimation vlsi circuits from a high-level view," in Proc. DAC, 1998, pp. 591–594.

S. Li et al., "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in Proc. MICRO, 2009, pp. 469–480.

D. S. Lopera et al., "Early rtl delay prediction using neural networks," Microprocessors and Microsystems, vol. 94, p. 104671, 2022, S. Roy et al., "A learning bridge from architectural synthesis to physical design for exploring power efficient high-performance adders," in Proc. ISLPED, 2017, pp. 1–6.

N. Wu et al., "Lostin: Logic optimization via spatio-temporal information with hybrid graph models," in Proc. ASAP, 2022, pp. 11–18.

- Logic synthesis gives a first look at circuit metrics like delay and area.

L.-T. Wang et al., Electronic design automation: synthesis, verification, and test. Morgan Kaufmann, 2009.

Background and Motivation

- Logic synthesis gives a first look at circuit metrics like delay and area.
- During logic synthesis certain metrics are correlated.

L.-T. Wang et al., Electronic design automation: synthesis, verification, and test. Morgan Kaufmann, 2009.

- Logic synthesis gives a first look at circuit metrics like delay and area.
- During logic synthesis certain metrics are correlated.
  - Collapsing Eq.(1) is a logic optimization technique that reduces delay but increases area, while substitution works in the opposite way.
  - Gate-sizing and buffer-insertion are strategies that trade off between delay and area during technology mapping.

$$
\begin{aligned}
&\mathrm{F} = \mathrm{G} \cdot \mathrm{a} + \neg \mathrm{G} \cdot \mathrm{b} \text{ and } \mathrm{G} = \mathrm{c} + \mathrm{d} \\
&\text{Collapsing G into F results in} \\
&\mathrm{F} = \mathrm{a} \cdot \mathrm{c} + \mathrm{a} \cdot \mathrm{d} + \mathrm{b} \cdot \neg \mathrm{c} \cdot \neg \mathrm{d}
\end{aligned}
\tag{1}
$$

L.-T. Wang et al., Electronic design automation: synthesis, verification, and test. Morgan Kaufmann, 2009.

- Logic synthesis gives a first look at circuit metrics like delay and area.
- During logic synthesis certain metrics are correlated.
  - Collapsing Eq.(1) is a logic optimization technique that reduces delay but increases area, while substitution works in the opposite way.
  - Gate-sizing and buffer-insertion are strategies that trade off between delay and area during technology mapping.

$$
\begin{aligned}
&\mathrm{F} = \mathrm{G} \cdot \mathrm{a} + \neg \mathrm{G} \cdot \mathrm{b} \text{ and } \mathrm{G} = \mathrm{c} + \mathrm{d} \\
&\text{Collapsing G into F results in} \\
&\mathrm{F} = \mathrm{a} \cdot \mathrm{c} + \mathrm{a} \cdot \mathrm{d} + \mathrm{b} \cdot \neg \mathrm{c} \cdot \neg \mathrm{d}
\end{aligned}
\tag{1}
$$

L.-T. Wang et al., Electronic design automation: synthesis, verification, and test. Morgan Kaufmann, 2009.

- Multi-task Learning (MTL) solves related tasks simultaneously.

R. Caruana, "Multitask learning," Machine learning, vol. 28, pp. 41–75, 1997.

I. Misra et al., "Cross-stitch networks for multi-task learning," in Proc. CVPR, 2016, pp. 3994–4003.

X. Xu et al., "Mtformer: Multi-task learning via transformer and cross-task reasoning," in Proc. ECCV, Springer, 2022, pp. 304–321.

S. Ruder, "An overview of multi-task learning in deep neural networks," arXiv preprint arXiv:1706.05098, 2017.

- Multi-task Learning (MTL) solves related tasks simultaneously.
- Advanced mechanisms are exploited to share features, like:
  - Linear combination of activations,
  - Attention-based feature sharing.

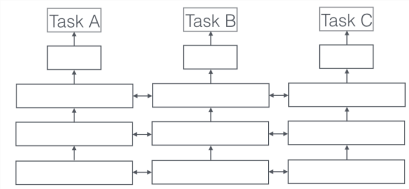

Figure 2: An example of feature sharing in MTL.

R. Caruana, "Multitask learning," Machine learning, vol. 28, pp. 41–75, 1997.

I. Misra et al., "Cross-stitch networks for multi-task learning," in Proc. CVPR, 2016, pp. 3994–4003.

X. Xu et al., "Mtformer: Multi-task learning via transformer and cross-task reasoning," in Proc. ECCV, Springer, 2022, pp. 304–321.

S. Ruder, "An overview of multi-task learning in deep neural networks," arXiv preprint arXiv:1706.05098, 2017.

We propose ASAP, a multi-task learning model that predicts delay and area of RTL design after logic synthesis simultaneously.

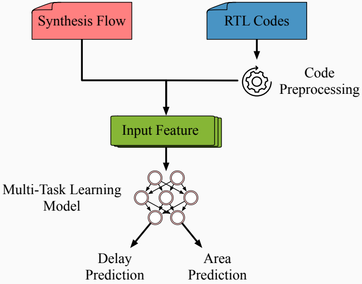

Figure 3: Our multi-task learning model.

Our model consists of the following parts:

1. A primary module that extracts synthesis-related features from RTL design and synthesis sequence.
2. Attention-based feature-extraction and feature-sharing modules extract specific features for delay and area and share them.
3. Ensemble prediction modules give the final prediction for delay and area.

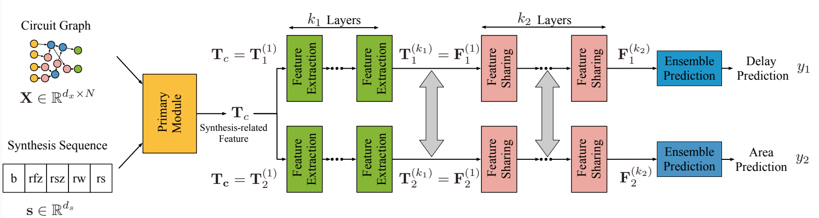

Figure 4: Overview of our model.

- The node features in GIN include node functionality (and or not), logic level, and number of fan-in and fan-out.
- The LSTM yield embedding for synthesis sequence.
- The GIN updates node embedding e_v by aggregating the neighbors', Eq.(2).

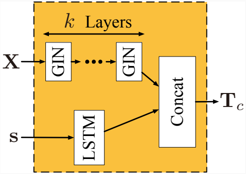

Figure 5: Primary module.

K. Xu et al., "How powerful are graph neural networks?" arXiv preprint arXiv:1810.00826, 2018.

A. M. R. Brayton, "Scalable logic synthesis using a simple circuit structure," in Proc. IWLS, vol. 6, 2006, pp. 15–22.

Synthesis-related Primary Module

- The node features in GIN include node functionality (and or not), logic level, and number of fan-in and fan-out.
- The LSTM yield embedding for synthesis sequence.
- The GIN updates node embedding $\mathrm{e_v}$ by aggregating the neighbors', Eq.(2).

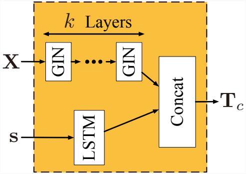

Figure 5: Primary module.

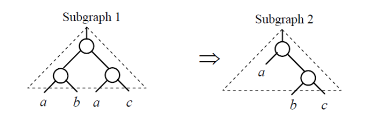

Figure 6: A cut being optimized.

$$\mathrm{e_v^{(l+1)}} = \mathrm{MLP^{(l)}}\left(\left(1+\epsilon^{(l)}\right)\cdot \mathrm{e_v^{(l)}} + \sum_{\mathrm{u}\in\mathcal{N}(\mathrm{v})} \mathrm{e_u^{(l)}}\right) \quad (2)$$

K. Xu et al., "How powerful are graph neural networks?" arXiv preprint arXiv:1810.00826, 2018.

A. M. R. Brayton, "Scalable logic synthesis using a simple circuit structure," in Proc. IWLS, vol. 6, 2006, pp. 15–22.

We use multi-head attention (MHA) mechanism to further extract specific features for each task.

A. Vaswani et al., "Attention is all you need," Proc. NIPS, vol. 30, 2017.

We use multi-head attention (MHA) mechanism to further extract specific features for each task.

- The Q,K,V are the same so each branch extracts specific features for the corresponding task.
- Multiple layers are stacked to get more specific representation.

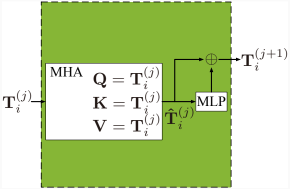

Figure 7: Feature-extraction.

A. Vaswani et al., "Attention is all you need," Proc. NIPS, vol. 30, 2017.

We use multi-head attention (MHA) mechanism to further extract specific features for each task.

- The Q,K,V are the same so each branch extracts specific features for the corresponding task.
- Multiple layers are stacked to get more specific representation.

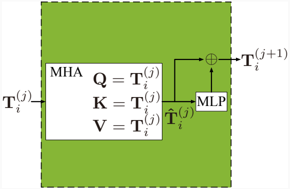

Figure 7: Feature-extraction.

$$\mathrm{MHA(Q, K, V)} = \mathrm{Concat}\left(\mathrm{H_1}, \ldots, \mathrm{H_G}\right) \mathrm{W}^{(\mathrm{O})}, \tag{3}$$

$$\begin{aligned} \mathrm{H_g} &= \mathrm{Attention}\left(\mathrm{QW_g^{(Q)}}, \mathrm{KW_g^{(K)}}, \mathrm{VW_g^{(V)}}\right) \\ &= \mathrm{softmax}\left(\frac{(\mathrm{QW_g^{(Q)}})(\mathrm{KW_g^{(K)}})^{\mathrm{T}}}{\sqrt{\mathrm{d_f}}}\right) \mathrm{VW_g^{(V)}}, \end{aligned} \tag{4}$$

A. Vaswani et al., "Attention is all you need," Proc. NIPS, vol. 30, 2017.

Attention-based Feature Sharing

After specific features for each task are extracted, we further share them across tasks.

After specific features for each task are extracted, we further share them across tasks.

- The cross-task attentions are computed for task i w.r.t all tasks.
- Then they are merged to get the new representation of features.
- Multiple layers are stacked to improve sharing.

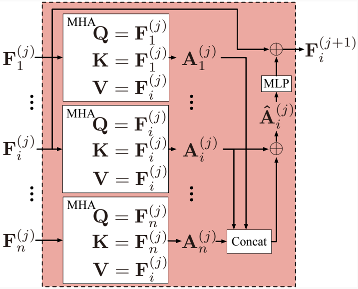

Figure 8: Feature sharing module for task i at layer j.

After specific features for each task are extracted, we further share them across tasks.

- The cross-task attentions are computed for task i w.r.t all tasks.
- Then they are merged to get the new representation of features.
- Multiple layers are stacked to improve sharing.

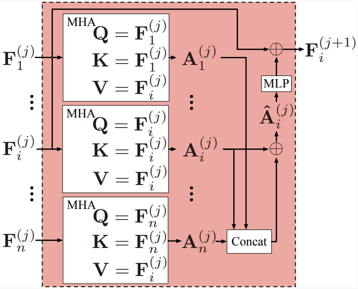

Figure 8: Feature sharing module for task i at layer j.

- Usually, regression models like MLP are used to obtain the final prediction from the features.
- Since the values of our data span across 3 orders of magnitudes, we use a classifier and a set of MLPS to give predictions.

- Usually, regression models like MLP are used to obtain the final prediction from the features.
- Since the values of our data span across 3 orders of magnitudes, we use a classifier and a set of MLPS to give predictions.
- The output of the classifier will decide which MLP will be used for the final prediction.

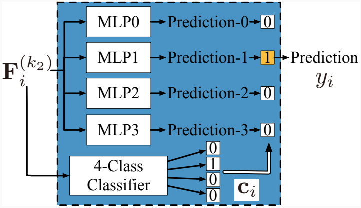

Figure 9: Ensemble prediction for task i.

- The designs are collected from EPFL15, ISCAS85 benchmark, prefix adders, and compressor-tree multipliers.
- Number of layers of GIN and LSTM are 2.
- Numbers of layers of feature-extraction and feature sharing are both 2, with 4 heads in multi-head attention.

L. Amarú et al., "The epfl combinational benchmark suite," in Proc. IWLS, 2015.

M. C. Hansen et al., "Unveiling the iscas-85 benchmarks: A case study in reverse engineering," IEEE Design & Test of Computers, vol. 16, no. 3, pp. 72–80, 1999.

L. Flea, Iamflea/addercircuitgenerator: This script generates and analyzes prefix tree adders. 2021. [Online]. Available: https://github.com/IamFlea/AdderCircuitGenerator.

Experiment Settings

- The designs are collected from EPFL15, ISCAS85 benchmark, prefix adders, and compressor-tree multipliers.
- Number of layers of GIN and LSTM are 2.
- Numbers of layers of feature-extraction and feature sharing are both 2, with 4 heads in multi-head attention.

To test the ability of our model to generalize, we devise two scenarios:

- Transductive Testing: The model will be tested on unseen synthesis flows with seen designs during training.
- Inductive Testing: The model will be tested on new designs.

L. Amarú et al., "The epfl combinational benchmark suite," in Proc. IWLS, 2015.

M. C. Hansen et al., "Unveiling the iscas-85 benchmarks: A case study in reverse engineering," IEEE Design & Test of Computers, vol. 16, no. 3, pp. 72–80, 1999.

L. Flea, Iamflea/addercircuitgenerator: This script generates and analyzes prefix tree adders. 2021. [Online]. Available: https://github.com/IamFlea/AdderCircuitGenerator.

Experimental Results

- We evaluate the model performance by mean-absolute-percentage-error (MAPE) with Linear Regression, CNN, LSTM, and LOSTIN.
- MTL baselines: 2-Ensemble, Self-attention, and Cross-stitch.

		LR	CNN	LSTM	LOSTIN	2E[a]	SA[b]	CS[c]	Ours
Transductive Testing	Delay Prediction	23.37%	42.21%	80.27%	11.52%	1.32%	0.90%	1.34%	0.79%
	Area Prediction	98.07%	63.58%	16.30%	10.85%	1.23%	1.06%	1.22%	0.83%
Inductive Testing	Delay Prediction	26.33%	38.04%	-	22.51%	7.78%	7.27%	12.35%	5.80%
	Area Prediction	146.05%	74.57%	-	25.78%	11.11%	7.63%	9.34%	5.84%

[a] 2-Ensemble baseline. [b] Self-attention baseline. [c] Cross-stitch baseline.

Table 1: MAPE Results Comparison with Baseline Methods.

C. Yu et al., "Developing synthesis flows without human knowledge," in Proc. DAC, 2018, pp. 1–6.

C. Yu and W. Zhou, "Decision making in synthesis cross technologies using lstms and transfer learning," in Proc. MLCAD, 2020, pp. 55–60.

N. Wu et al., "Lostin: Logic optimization via spatio-temporal information with hybrid graph models," in Proc. ASAP, 2022, pp. 11–18.

R. Caruana, "Multitask learning," Machine learning, vol. 28, pp. 41–75, 1997.

- We further validate our model on a dataset solely consisting of adders and multipliers.
- Adders are of 4,8,16,32,64,128 bit., multipliers are of 4,8,16,32,64 bits.
- We compare the results in terms of normalized hypervolumn.

Table 2: Hypervolume Ratio of Delay and Area Comparison.

Design	ADD32	ADD64	ADD128	MUL16	MUL32	MUL64	Averaged
LOSTIN [5]	0.705	0.704	0.826	0.873	0.693	0.352	0.692
2-Ensemble [7]	0.889	0.831	0.927	0.931	0.593	0.861	0.838
Self-attention [9]	0.814	0.859	0.835	0.802	0.870	0.939	0.853
Cross-stitch [8]	0.724	0.861	0.964	0.558	0.910	0.804	0.803
Ours	1.000	1.000	1.000	1.000	1.000	1.000	1.000

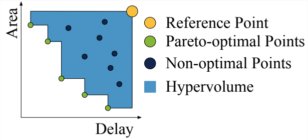

Figure 10: An illustration of hypervolume.

We can also look at the Pareto frontier given by the model on 16 and 32-bit multipliers.

Our model has better coverage than baseline methods.

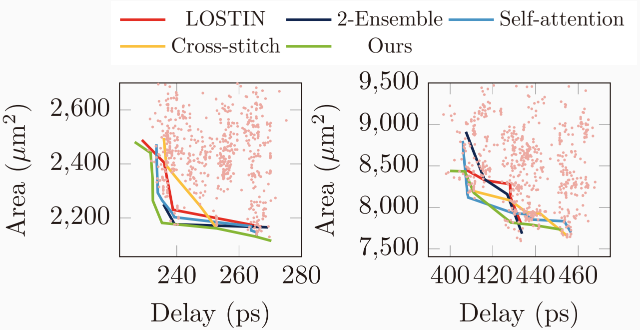

Figure 11: Pareto-frontiers comparison of baseline methods on 16 and 32-bit multipliers.

Thanks for listening