ASP-DAC 2024
29th Asia and South Pacific Design Automation Conference

LSTP: A Logic Synthesis Timing Predictor

Haisheng Zheng ♠ Zhuolun He ♠♡ Fangzhou Liu ♠♡ Zehua Pei ♠♡ Bei Yu ♡

♠ Shanghai Artificial Intelligence Laboratory
♡ The Chinese University of Hong Kong

Jan. 25, 2024

Outline

Introduction

Background

Logic Synthesis is Critical:

- Architecture exploration relies on the acquisition of metrics reported by logic synthesis [1].
- Logic synthesis quality determines the best possible design space of subsequent procedure [2].

[1] **Chen Bai et al. (2021). "BOOM-Explorer: RISC-V BOOM Microarchitecture Design Space Exploration Framework".** In: *Proc. ICCAD*.

[2] **Ceyu Xu et al. (2022). "SNS's Not a Synthesizer: A Deep-Learning-Based Synthesis Predictor".** In: *Proc. ISCA*.

Background

Logic Synthesis is Critical:

- Architecture exploration relies on the acquisition of metrics reported by logic synthesis [1].
- Logic synthesis quality determines the best possible design space of subsequent procedure [2].

Can we efficiently predict the desired metrics without actually running expensive logic synthesis?

[1] **Chen Bai et al. (2021). "BOOM-Explorer: RISC-V BOOM Microarchitecture Design Space Exploration Framework".** In: *Proc. ICCAD*.

[2] **Ceyu Xu et al. (2022). "SNS's Not a Synthesizer: A Deep-Learning-Based Synthesis Predictor".** In: *Proc. ISCA*.

Obtain the Performance of a Circuit: Previous Work

Work	Target	Algorithm
D-SAGE [3]	Timing	Graph Neural Network
Yu *et al.* [4]	Timing, Area	Long Short Term Memory
PowerNet [5]	Dynamic IR Drop	Convolutional Neural Network
GRANNITE [6]	Power	Graph Neural Network
Deep H-GCN [7]	Analog Circuit Degradation (i.e., Aging)	Graph Neural Network
De *et al.* [8]	Timing	Machine Learning Methods
SNS [9]	Timing, Area, Power	Transformer

[2] **Ceyu Xu et al. (2022). "SNS's Not a Synthesizer: A Deep-Learning-Based Synthesis Predictor".** In: *Proc. ISCA*.

[3] **Ecenur Ustun et al. (2020). "Accurate Operation Delay Prediction for FPGA HLS Using Graph Neural Networks".** In: *Proc. ICCAD*.

[4] **Cunxi Yu et al. (2020). "Decision Making in Synthesis Cross Technologies Using LSTMs and Transfer Learning".** In: *Proc. MLCAD*.

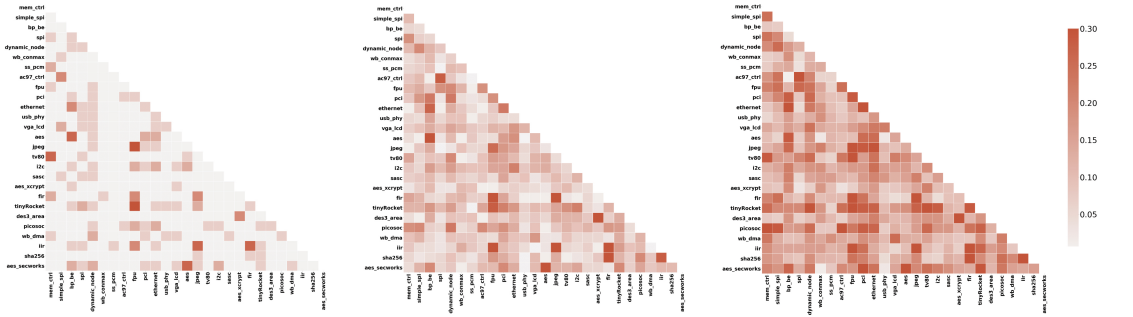
[5] **Zhiyao Xie et al. (2020). "PowerNet: Transferable Dynamic IR Drop Estimation via Maximum Convolutional Neural Network".** In: *Proc. ASPDAC*.

[6] **Yanqing Zhang et al. (2020). "GRANNITE: Graph Neural Network Inference for Transferable Power Estimation".** In: *Proc. DAC*.

[7] **Tinghuan Chen et al. (2021). "Deep H-GCN: Fast Analog IC Aging-Induced Degradation Estimation".** In: *IEEE TCAD*.

[8] **Sayandip De et al. (2022). "Delay Prediction for ASIC HLS: Comparing Graph-Based and Non-Graph-Based Learning Models".** In: *IEEE TCAD*.

Logic Synthesis Recipes are NOT One-Size-Fits-All



(a) Top 1% (b) Top 5% (c) Top 10%

Comparison of Different Top Percentages.

- OpenABC-D [10] has pointed out quantitatively that the similarity between the best synthesis recipes for a set of benchmark circuits is less than 30%.

[10] **Animesh Basak Chowdhury et al. (2021). "OpenABC-D: A Large-Scale Dataset for Machine Learning Guided Integrated Circuit Synthesis".** In: *arXiv preprint*.

Optimization Sequence Quality Improvement

Previous Works:

- Yu *et al.* [11] propose to train a Convolutional Neural Network (CNN) to predict the quality of an optimization sequence.
- Reinforcement Learning (RL) is leveraged [12][13] to generate fixed-length optimization sequences.

[11] **Cunxi Yu et al. (2018). "Developing Synthesis Flows without Human Knowledge".** In: *Proc. DAC*.

[12] **Winston Haaswijk et al. (2018). "Deep Learning for Logic Optimization Algorithms".** In: *Proc. ISCAS*.

[13] **Keren Zhu et al. (2020). "Exploring Logic Optimizations with Reinforcement Learning and Graph Convolutional Network".** In: *Proc. MLCAD*.

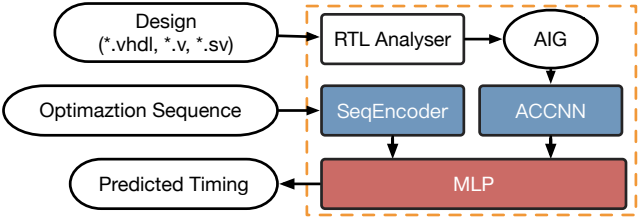
Problem Formulation

Logic Synthesis Timing Prediction:

- *Given a gate-level netlist as an And-Inverter Graph (AIG) representing a set of Boolean functions and a sequence of subgraph optimization procedures for the AIG graph, design a novel learning methodology that automatically predicts the final timing after applying the optimization procedures to the AIG.*

Algorithm

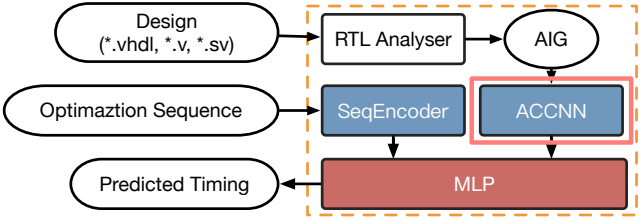
Overall Flow



Overall Flow of *LSTP*.

- **RTL-Analyzer** compiles the input design and transforms it into an AIG representation.

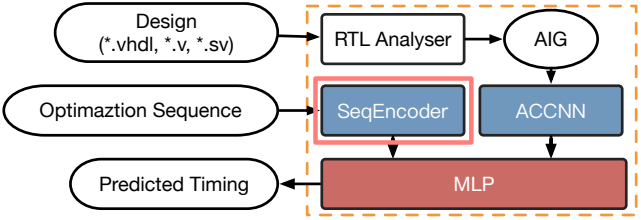
Overall Flow



Overall Flow of *LSTP*.

- **RTL-Analyzer** compiles the input design and transforms it into an AIG representation.
- **ACCNN** is a Customized GNN for node sampling and feature extraction of the AIG circuit.

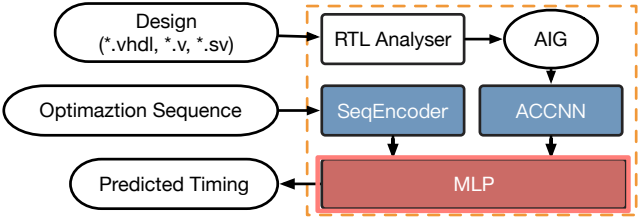
Overall Flow



Overall Flow of *LSTP*.

- **RTL-Analyzer** compiles the input design and transforms it into an AIG representation.
- **ACCNN** is a Customized GNN for node sampling and feature extraction of the AIG circuit.
- **SeqEncoder** is a Transformer Encoder for optimization sequence features extraction.

Overall Flow



Overall Flow of *LSTP*.

- **RTL-Analyzer** compiles the input design and transforms it into an AIG representation.
- **ACCNN** is a Customized GNN for node sampling and feature extraction of the AIG circuit.
- **SeqEncoder** is a Transformer Encoder for optimization sequence features extraction.
- **MLP** aggregates both the optimization sequence features and the circuit diagram features to predict the timing of the input design.

RTL Analyser

RTL Code

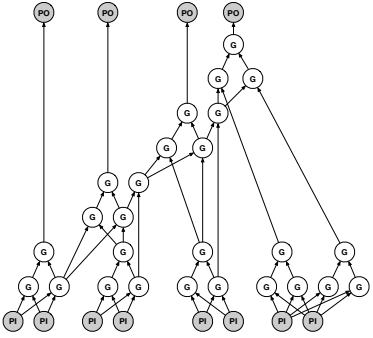
```
module Subtractor
(
  input   [3:0] in0,
  input   [3:0] in1,
  output  [3:0] out
);

assign out = in0 - in1;

endmodule
```

Yosys

And-Inverter-Graph (AIG)



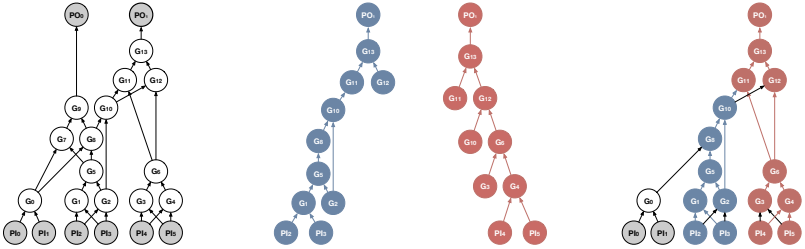
Visualization of *RTL Analyser*.

ACCNN: Cascaded Cones

Motivation:

- The delay of a circuit depends on the number of hops on the longest path from the primary inputs (PIs) to the primary outputs (POs).
- We wish to design an algorithm to effectively exploit the characteristics of graph representation for the longest path in AIG.

ACCNN: Cascaded Cones



A Visual Illustration of *Sampling Cascaded Cones*.

A Random Walk-Based Approach to Sample Cascaded Cones Within the Circuit:

- Each 'path' originate from primary input (PI) and end at primary output (PO).
- The output of flip-flop $\rightarrow$ PI.
- The input of flip-flop $\rightarrow$ PO.

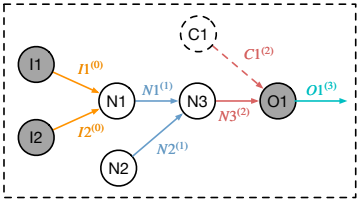
ACCNN: Netlist Feature Extraction

Motivation:

- We aim for a model that similar to logic simulation, efficiently propagating information step-by-step along the sampled paths.
- ABGNN [14] serves this purpose well.

[14] **Zhuolun He et al. (2021). "Graph Learning-Based Arithmetic Block Identification".** In: *Proc. ICCAD*.

ACCNN: Netlist Feature Extraction



	I1	I2	N1	N2	N3	C1	O1
T = 0	✓	✓					
T = 1			✓	✓			
T = 2					✓	✓	
T = 3							✓

A Visual Illustration of *ACCNN*.

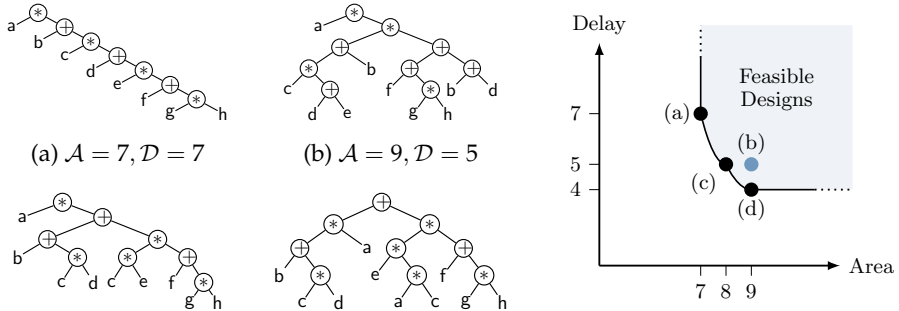
- **Feature Aggregation:**

$$a^{(k)}_{\{i:\mathcal{D}(i,v)=\Delta-k\}} = \text{AGGREGATE}(\{h^{(k-1)}u : u \in N(i)\}). \tag{1}$$

- **Update Function:**

$$h^{(k)}_{\{i:\mathcal{D}(i,v)=\Delta-k\}} = \text{COMBINE}(a_i^{(k)}, h_i^{(0)}). \tag{2}$$

Optimization Sequence Feature Extraction: Motivation

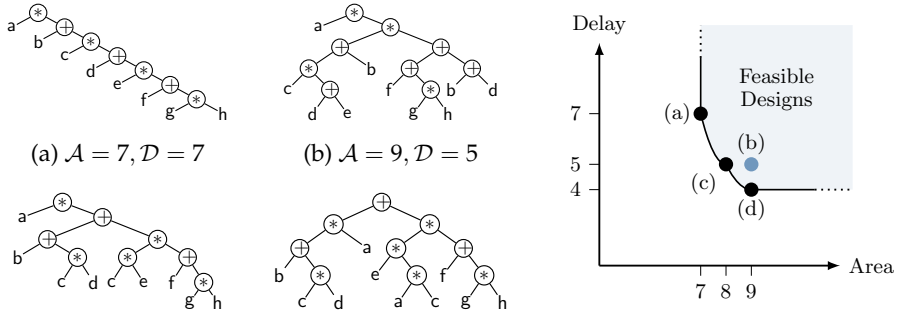


(a) $\mathcal{A} = 7, \mathcal{D} = 7$ (b) $\mathcal{A} = 9, \mathcal{D} = 5$ (c) $\mathcal{A} = 8, \mathcal{D} = 5$ (d) $\mathcal{A} = 9, \mathcal{D} = 4$ (e)

(a)–(d) Equivalent Factored Forms; (e) Area/Delay Trade-Off for the Trees.

- Boolean Expression: $ab + acd + acef + acegh$.
- Assuming zero arrival time for all PIs, unit area ($\mathcal{A} = 1$), unit delay ($\mathcal{D} = 1$).

Optimization Sequence Feature Extraction: Motivation

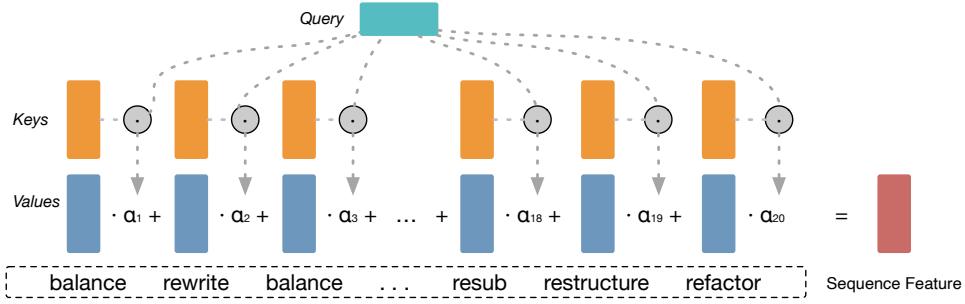


(a) $\mathcal{A} = 7, \mathcal{D} = 7$ (b) $\mathcal{A} = 9, \mathcal{D} = 5$ (c) $\mathcal{A} = 8, \mathcal{D} = 5$ (d) $\mathcal{A} = 9, \mathcal{D} = 4$ (e)

(a)–(d) Equivalent Factored Forms; (e) Area/Delay Trade-Off for the Trees.

- It is hardly possible for designers to determine the effect of optimization sequences for different designs.
- We need a model that takes into account optimization sequence ordering and position.

SeqEncoder: Optimization Sequence Feature Extraction



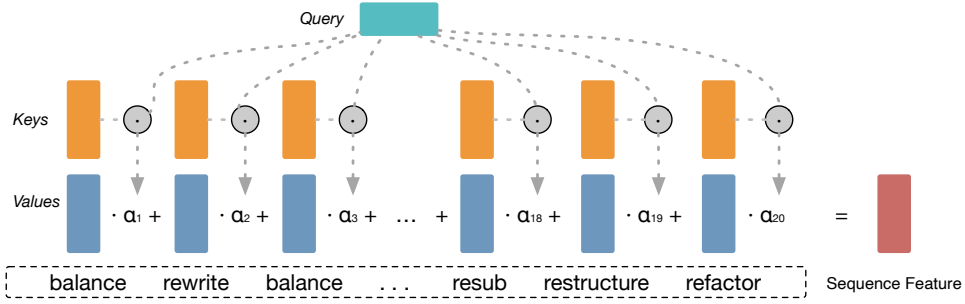
A Visual Illustration of *SeqEncoder*.

Transformer [15] is one of such models:

$$\text{Attention}(\vec{Q}, \vec{K}, \vec{V}) = \text{softmax}(\frac{\vec{Q}\vec{K}^\top}{\sqrt{d_k}})\vec{V} \tag{3}$$

[15] **Ashish Vaswani et al. (2017). "Attention is All You Need".** In: *Proc. NeurIPS*.

SeqEncoder: Optimization Sequence Feature Extraction

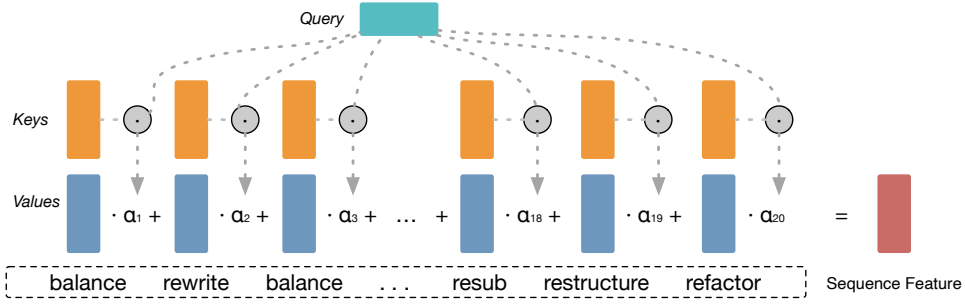


A Visual Illustration of *SeqEncoder*.

Optimization Methods

- Balancing, Reconfiguration, Replacing and Rewriting.

SeqEncoder: Optimization Sequence Feature Extraction



A Visual Illustration of *SeqEncoder*.

SeqEncoder supports extracting features of optimization sequences of length 20 or less, when the length of the optimization sequence is less than 20:

- Zero padding $\rightarrow$ *'empty optimization'*.
- $[empty, rewrite, balance, \ldots, resub, restructure, refactor]$.

Experiments

Experimental Setup

- Developed the timing prediction framework in Python.
  - Tools: Yosys, ABC.
  - Libraries: Pytorch Geometric, PyTorch, NetworkX.
- Dataset: Open-Source Designs.
- Mean Absolute Percentage Error (MAPE):

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{\hat{Y}_i - Y_i}{Y_i} \right|. \tag{3}$$

Performance of LSTP

Table: Evaluation Accuracy *(MAPE)*.

Name	# PI	# PO	# Node	# Level	SNS [15]	Runtime (s)	LSTP	Runtime (s)
aes	683	529	39215	44	50.21%	2.85	**25.44%**	3.38
des3_area	303	64	7766	47	53.84%	2.16	**20.29%**	0.70
dft	37597	37417	488165	83	86.90%	27.18	**33.56%**	55.53
fpu	632	409	55935	1522	26.11%	4.96	**3.35%**	6.97
idft	37603	37419	481184	82	**5.07%**	16.16	8.18%	54.04
mem_ctrl	1187	962	29814	56	23.21%	32.02	**19.22%**	3.71
sasc	135	125	1214	15	21.44%	2.38	**2.48%**	0.12
ss_pcm	104	90	762	13	67.82%	2.30	**6.46%**	0.09
tinyRocket	4561	4181	99775	156	37.31%	81.87	**10.81%**	11.86
usb_phy	132	90	893	16	14.4%	2.65	**7.75%**	0.10
Average					38.63%	17.45	**13.75%**	**13.65**

- **LSTP** outperforms prior works on all the test cases except for the `idft`.

[2] **Ceyu Xu et al. (2022). "SNS's Not a Synthesizer: A Deep-Learning-Based Synthesis Predictor".** In: *Proc. ISCA*.

LSTP for Optimization Sequence Generation

Table: Comparison of Timing Minimums.

Name	Initial (ns)	*resyn2*-2 (ns)	Improvement	LSTP (ns)	Improvement
aes	1.58	1.37	13.29%	1.24	**21.52%**
des3_area	2.66	3.74	-40.60%	3.33	-25.19%
dft	5.82	6.35	-9.11%	4.94	**15.12%**
fpu	51	41.5	18.63%	40.34	**20.90%**
idft	5.82	6.35	-9.11%	5.54	**4.81%**
mem_ctrl	6.74	3.03	55.04%	2.94	**56.38%**
sasc	0.89	0.69	22.47%	0.49	**44.94%**
ss_pcm	0.66	0.58	12.12%	0.48	**27.27%**
tinyRocket	78.1	12	84.64%	10.39	**86.70%**
usb_phy	0.41	0.42	-2.44%	0.32	**21.95%**
Average			14.49%		**27.44%**

- **LSTP** can be used to find a *Better Optimization Sequence*.

Conclusion

In this paper, we proposed:

- A Machine Learning-Driven logic synthesis timing predictor.
- A Specialized GNN to sample and learn the intrinsic features of circuit AIG.
- An Appropriate Neural Model to model the complex interaction between optimization passes and their effects on the input netlist.

THANK YOU!