

LSTP : A Logic Synthesis Timing Predictor

Haisheng Zheng¹, Zhuolun He^{1,2}, Fangzhou Liu^{1,2},
Zehua Pei^{1,2}, Bei Yu²

¹Shanghai AI Laboratory, Shanghai, China

²The Chinese University of Hong Kong

Jan. 25, 2024



① Introduction

② Algorithm

③ Experiments

Introduction

Logic Synthesis is critical:

- Architecture exploration relies on the acquisition of metrics reported by logic synthesis¹
- Logic synthesis quality determines the best possible design space of subsequent procedure²

¹Chen Bai et al. (2021). “BOOM-Explorer: RISC-V BOOM microarchitecture design space exploration framework”. In: *Proc. ICCAD*.

²Ceyu Xu et al. (2022). “SNS’s Not a Synthesizer: A Deep-Learning-Based Synthesis Predictor”. In: *Proc. ISCA*.

Logic Synthesis is critical:

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

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

Previous Works

Work	Estimation	Algorithm
D-SAGE ³	Timing	GNN
Yu <i>et al.</i> ⁴	Timing, Area	LSTM
PowerNet ⁵	Dynamic IR Drop	CNN
GRANNITE ⁶	Power	GNN
Deep H-GCN ⁷	Analog Circuit Degradation (i.e., aging)	GNN
De <i>et al.</i> ⁸	Timing	ML methods
SNS	Timing, Area, Power	Transformer

³Ecenur Ustun et al. (2020). “Accurate operation delay prediction for FPGA HLS using graph neural networks”. In: *Proc. ICCAD*.

⁴Cunxi Yu et al. (2020). “Decision making in synthesis cross technologies using LSTMs and transfer learning”. In: *Proc. MLCAD*.

⁵Zhiyao Xie et al. (2020). “PowerNet: Transferable dynamic IR drop estimation via maximum convolutional neural network”. In: *Proc. ASPDAC*.

⁶Yanqing Zhang et al. (2020). “GRANNITE: Graph neural network inference for transferable power estimation”. In: *Proc. DAC*.

⁷Tinghuan Chen et al. (2021). “Deep H-GCN: Fast analog IC aging-induced degradation estimation”. In: *IEEE TCAD*.

⁸Sayandip De et al. (2022). “Delay Prediction for ASIC HLS: Comparing Graph-based and Non-Graph-based Learning Models”. In: *IEEE TCAD*.

Previous Works

- Yu *et al.*¹⁰ propose to train a CNN to predict the quality of an optimization sequence.
- Reinforcement Learning (RL) is leveraged^{11,12} to generate fixed-length optimization sequences.

¹⁰Cunxi Yu et al. (2018). “Developing synthesis flows without human knowledge”. In: *Proc. DAC*.

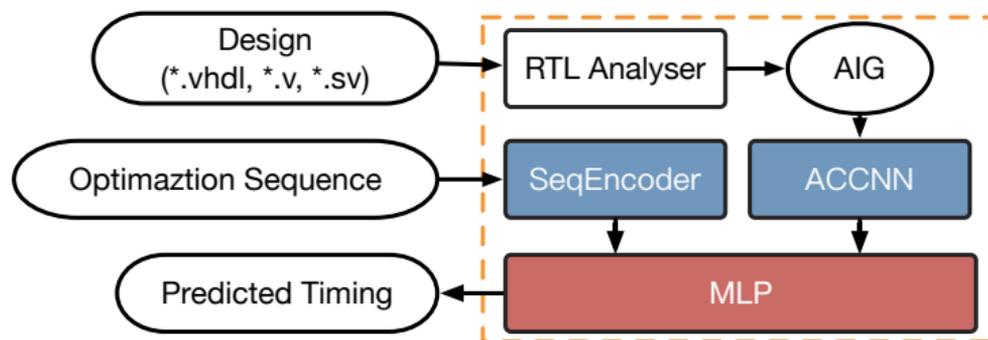
¹¹Winston Haaswijk et al. (2018). “Deep learning for logic optimization algorithms”. In: *Proc. ISCAS*.

¹²Keren Zhu et al. (2020). “Exploring logic optimizations with reinforcement learning and graph convolutional network”. In: *Proc. MLCAD*.

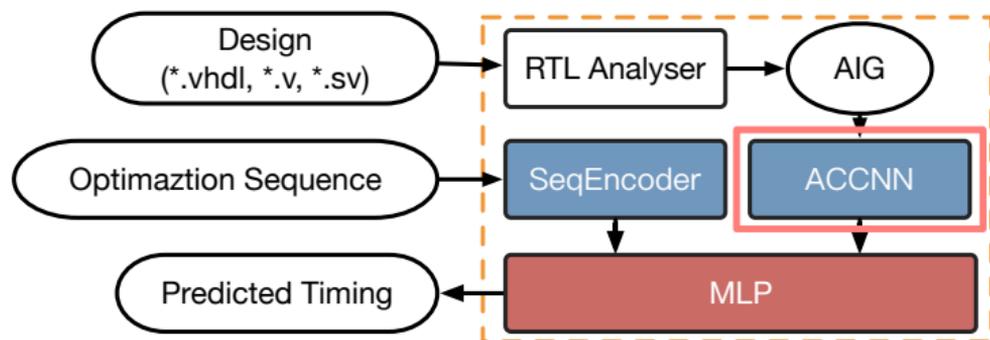
Logic Synthesis Timing Prediction

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.

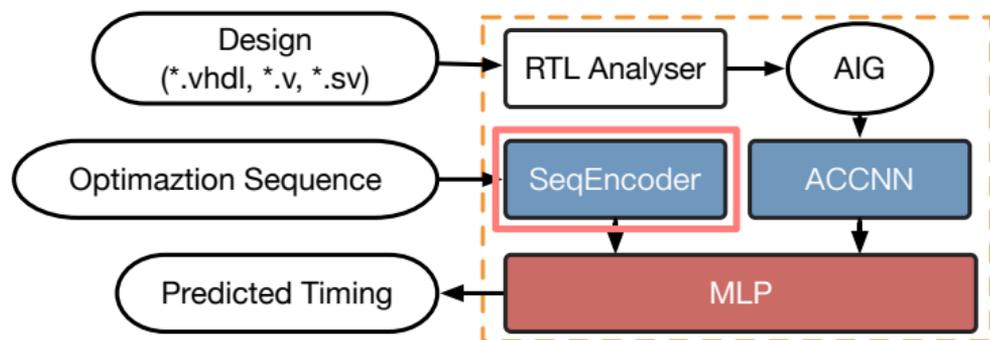
Algorithm



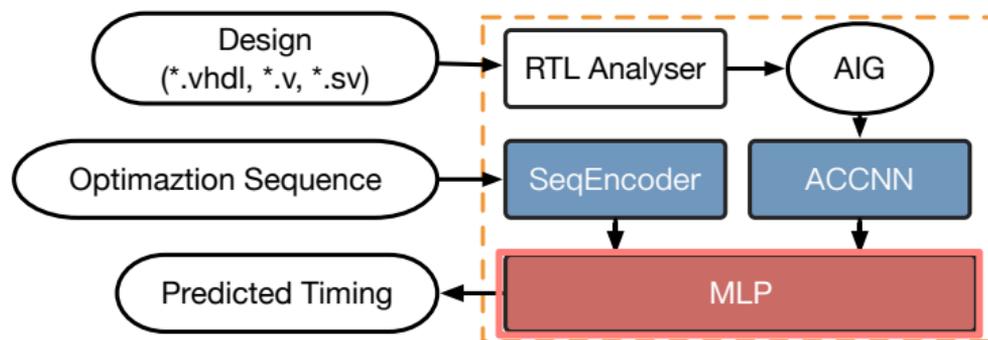
- **RTL-Analyzer** compiles the input design and transforms it into an And-Inverter-Graph (AIG) representation.



- **RTL-Analyzer** compiles the input design and transforms it into an And-Inverter-Graph (AIG) representation.
- **ACCNN** is a trained graph neural network (GNN) for node sampling and feature extraction of the AIG circuit.



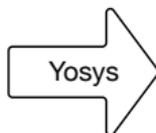
- **RTL-Analyzer** compiles the input design and transforms it into an And-Inverter-Graph (AIG) representation.
- **ACCNN** is a trained graph neural network (GNN) for node sampling and feature extraction of the AIG circuit.
- **SeqEncoder** is a trained Transformer encoder for optimization sequence features extraction.



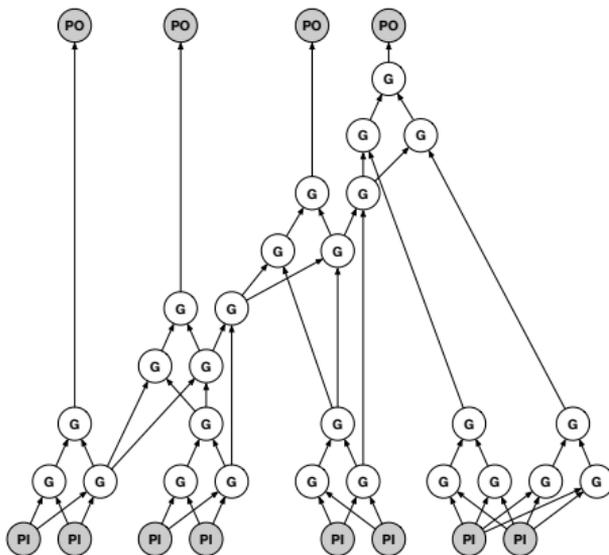
- **RTL-Analyzer** compiles the input design and transforms it into an And-Inverter-Graph (AIG) representation.
- **ACCNN** is a trained graph neural network (GNN) for node sampling and feature extraction of the AIG circuit.
- **SeqEncoder** is a trained Transformer encoder for optimization sequence features extraction.
- **MLP** aggregates both the optimization sequence features and the circuit diagram features to predict the timing of the input design.

HDL Code

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

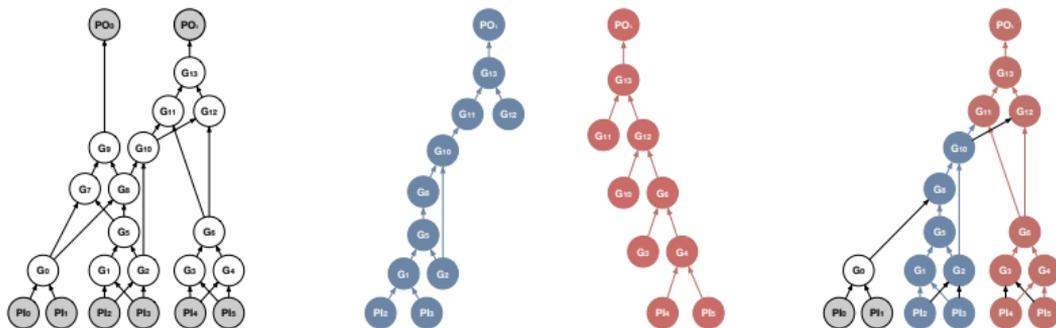


And-Inverter-Graph (AIG)



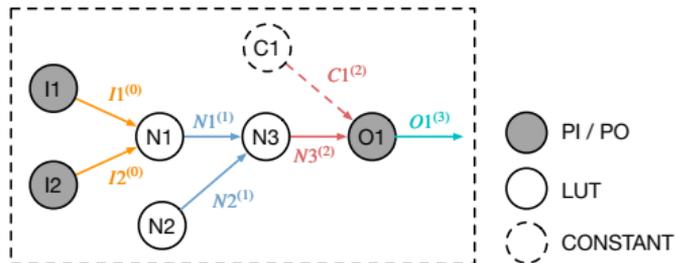
- The delay of a circuit depends on the number of hops on the longest path from the primary inputs (PIs) to the primary outputs (POs).
- We wish to design an algorithm to effectively exploit the characteristics of graph representation for the longest path in AIG.

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

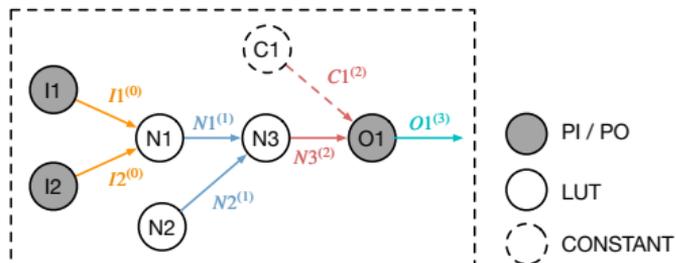


	I1	I2	N1	N2	N3	C1	O1
T = 0	✓	✓					
T = 1			✓	✓			
T = 2					✓	✓	
T = 3							✓

A visual illustration of ACCNN.

- We aim for a model that similar to logic simulation, efficiently propagating information step-by-step along the sampled paths
- ABGNN¹³ serves this purpose well

¹³Zhuolun He et al. (2021). "Graph Learning-Based Arithmetic Block Identification". In: *Proc. ICCAD*.

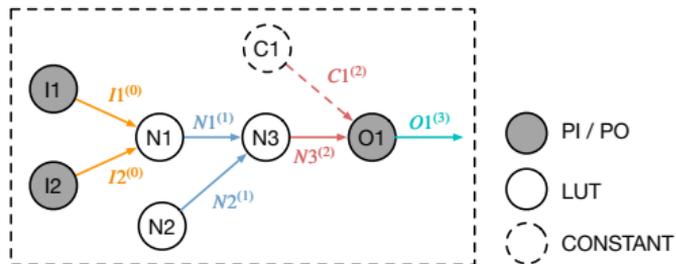


	I1	I2	N1	N2	N3	C1	O1
T = 0	✓	✓					
T = 1			✓	✓			
T = 2					✓	✓	
T = 3							✓

A visual illustration of ACCNN.

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

$$h_{\{i:\mathcal{D}(i,v)=\Delta-k\}}^{(k)} = \text{COMBINE}(a_i^{(k)}, h_i^{(0)}) \quad (2)$$

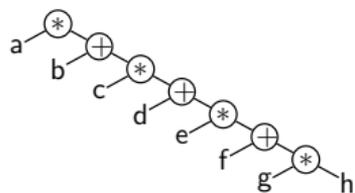


	I1	I2	N1	N2	N3	C1	O1
T = 0	✓	✓					
T = 1			✓	✓			
T = 2					✓	✓	
T = 3							✓

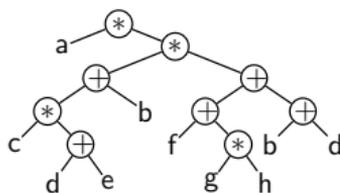
A visual illustration of ACCNN.

Node Type

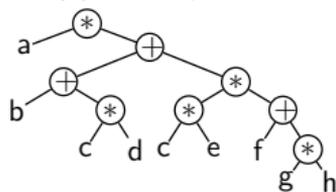
PI PO CONSTANT LUT 0x1 LUT 0x2 LUT 0x8



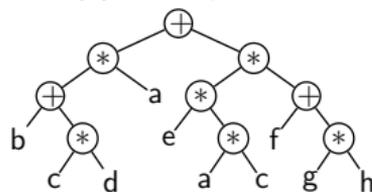
(a) $\mathcal{A} = 7, \mathcal{D} = 7$



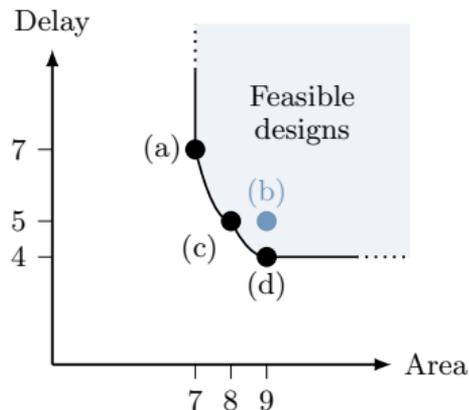
(b) $\mathcal{A} = 9, \mathcal{D} = 5$



(c) $\mathcal{A} = 8, \mathcal{D} = 5$



(d) $\mathcal{A} = 9, \mathcal{D} = 4$

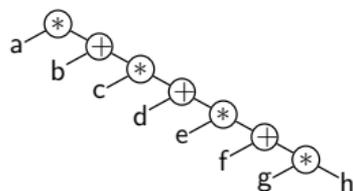


(e)

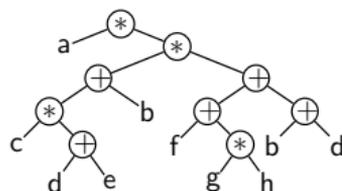
(a)–(d) Equivalent factored forms; (e) Area/delay trade-off for the trees.¹⁴

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

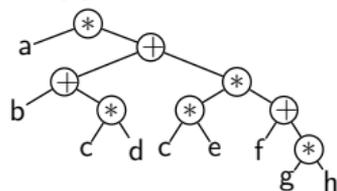
¹⁴Kanwar Jit Singh et al. (1988). "Timing optimization of combinational logic.". In: *Proc. ICCAD*. 18/28



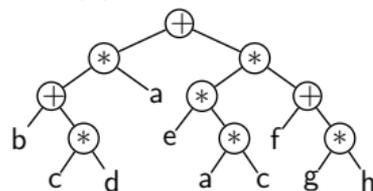
(a) $\mathcal{A} = 7, \mathcal{D} = 7$



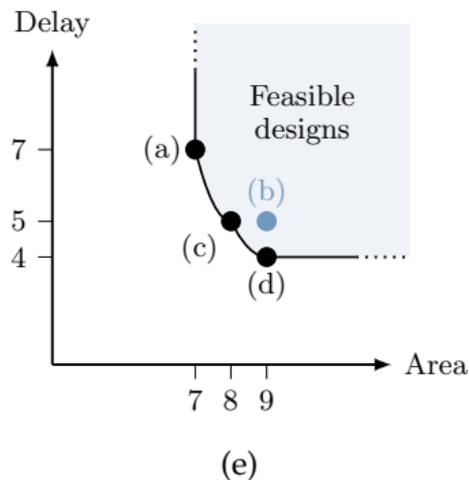
(b) $\mathcal{A} = 9, \mathcal{D} = 5$



(c) $\mathcal{A} = 8, \mathcal{D} = 5$

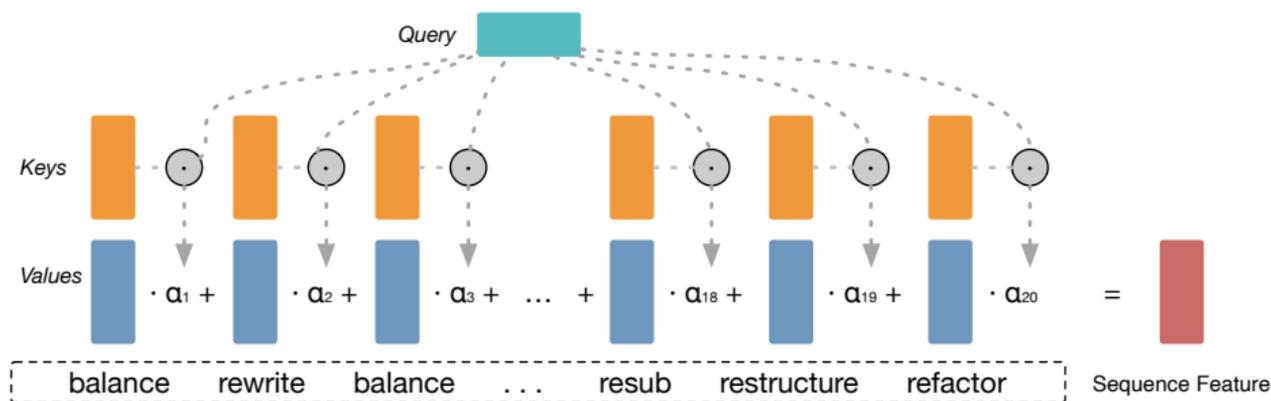


(d) $\mathcal{A} = 9, \mathcal{D} = 4$



(a)–(d) Equivalent factored forms; (e) Area/delay trade-off for the trees.

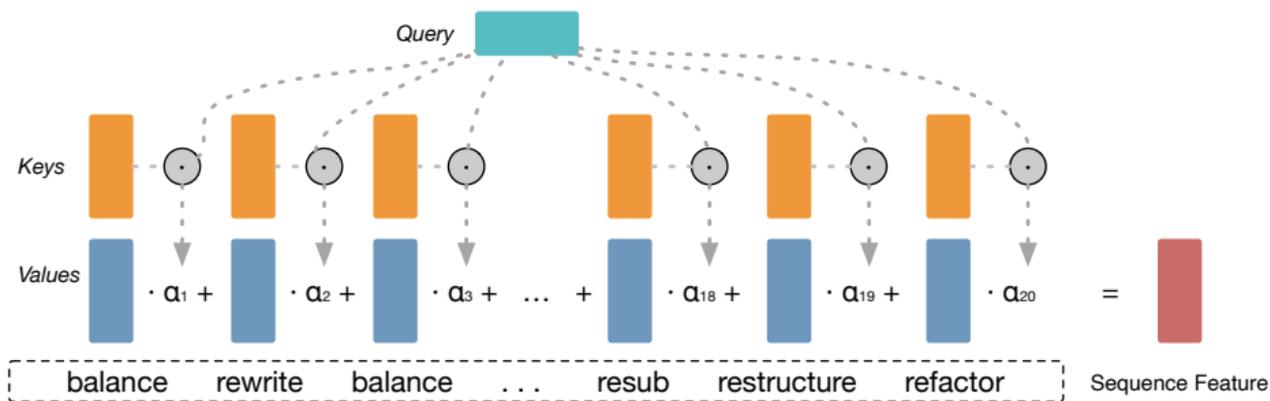
- It is hardly possible for designers to determine the effect of optimization sequences for different designs.
- We need a model that takes into account optimization sequence ordering and position.



- Transformer¹⁵ is one of such models

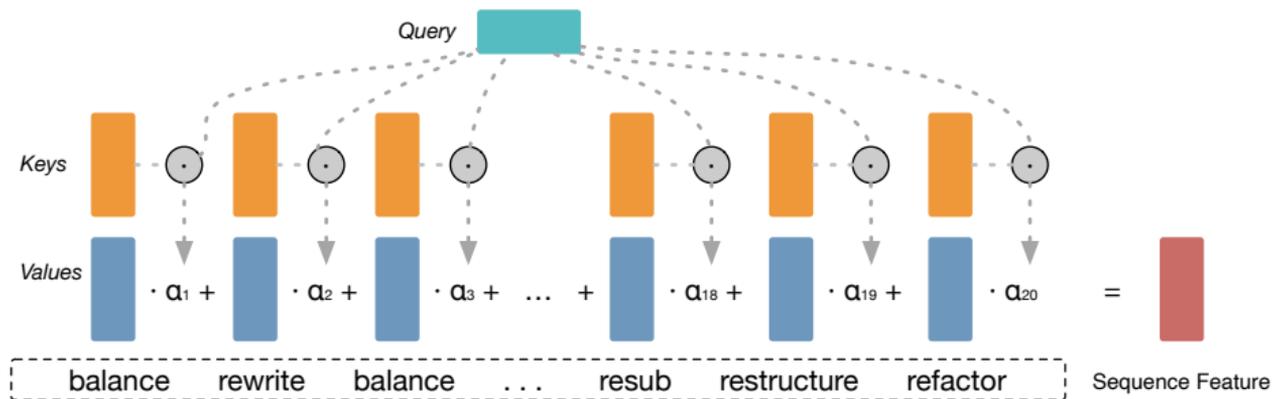
$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \quad (3)$$

¹⁵Ashish Vaswani et al. (2017). "Attention is all you need". In: *Proc. NIPS*.



Optimization Methods

Balancing, Reconfiguration, Replacing and Rewriting.



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, ..., resub, restructure, refactor]

Experiments

- Developed the timing prediction framework in Python
 - Tools: Yosys, ABC
 - Libraries: Pytorch Geometric, PyTorch, networkx
- Dataset: open-source designs

Type	IP	
	Train	Valid/Test
Bus protocol	i2c, spi ethernet, wb_dma simple_spi, pci wb_conmax	usb_phy ss_pcm sasc
Controller	ac97_ctrl, bp_be vga_lcd	mem_ctrl
Crypto	aes_secworks aes_xcrypt sha256	aes des3_area
DSP	fir, iir, jpeg	dft, idft
Processor	dynamic_node picosoc, tv80	fpu, tinyRocket

- Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{\hat{Y}_i - Y_i}{Y_i} \right| \quad (4)$$

Table: Evaluation Accuracy (MAPE)

Name	# PI	# PO	# Node	# Level	SNS	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

- Our proposed method greatly outperforms prior works on all the test cases except for the `idft`.

Table: Comparison of Timing minimums

Name	Initial [ns]	<i>resyn2-2</i> [ns]	Improve [%]	LSTP [ns]	Improve [%]
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

- Various tasks from architectural exploration to physical design DSE have highlighted the demand for fast logic synthesis result prediction
- 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
- We conducted comprehensive experiments on real-world circuit designs to evaluate our methods

THANK YOU!