Logic Synthesis with Generative Deep Neural Networks

Xihan Li[1], Xing Li[2], Lei Chen[2], Xing Zhang[2], Mingxuan Yuan[2] and Jun Wang[1]

1. Department of Computer Science, University College London
2. Huawei Noah's Ark Lab, Hong Kong, China

Abstract

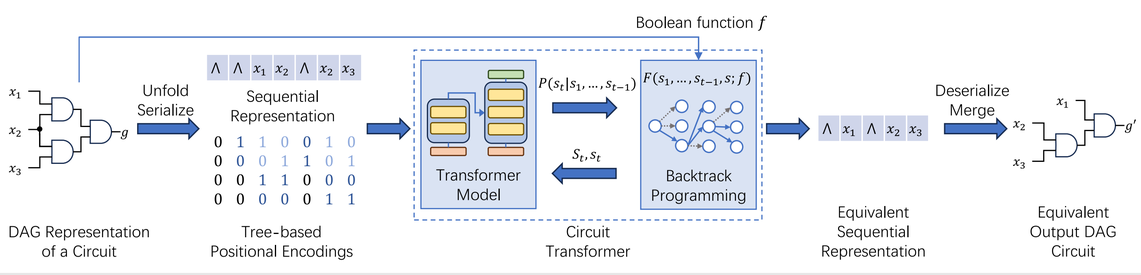
In this work, we introduced the first logic synthesis operator powered by generative deep neural networks. More specifically, a rewriting operator is developed based on the Circuit Transformer model, named `ctrw` (Circuit Transformer Rewriting), which incorporates the following techniques: (1) a two-stage training scheme for the Circuit Transformer tailored for logic synthesis, with iterative improvement of optimality through self-improvement training; (2) integration of the Circuit Transformer with state-of-the-art rewriting techniques to address scalability issues, allowing for guided DAG-aware rewriting. Experimental results on the IWLS 2023 contest benchmark demonstrate the effectiveness of our proposed rewriting methods.

1. Circuit Transformer: Sequential Generation of Circuits with Equivalence Preserved

This work is based on Circuit Transformer [1], a generative deep neural network which has two features:

1. It allows sequential generation of circuits with next token prediction, just like ChatGPT to natural languages.
2. The generated circuit is precisely equivalent to an existing input circuit.

[1] https://arxiv.org/abs/2403.13838



2. Train a Circuit Transformer for Small-sized Logic Synthesis

End-to-end, supervised learning with:

- **Input:** randomly generated circuits with unique canonicalizations (realistic circuits do NOT work well)
- **Output:** optimized circuits with existing optimizers (resyn2)

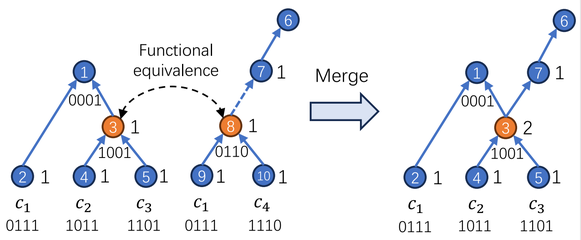
3. Circuit Size Minimization as a Markov Decision Process

We can minimize the number of AND gates of the generated circuit by attaching an immediate reward function $R(s_1, \dots, s_t, s)$ to the generation of token s at step t, so that the generation process can be modelled as an MDP.

- State at step t: $g_1, \dots, g_t$ Action: g
- Immediate reward: $R(g_1, \dots, g_t, g) = \Delta + \begin{cases} -1, g = \wedge \text{ or } g = \overline{\wedge} \\ 0, \text{otherwise} \end{cases}$

(In which Δ reflects the refinement of equivalent node merging)

- Cumulative reward: negative number of AND nodes in the generated circuits

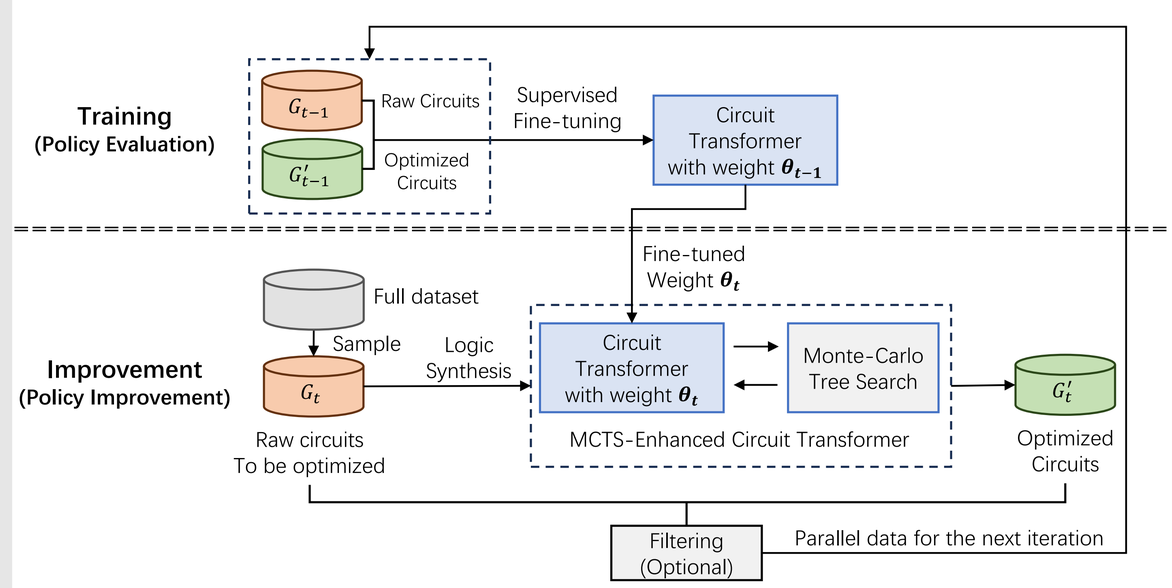


Step (Node ID)	1	2	3	4	5	6	7	8	9	10
Token	∧	c_1	∧	c_2	c_3	∧	∧	$\overline{\wedge}$	c_1	c_4
Immediate Reward	-1	0	-1	0	0	-1	-1	-1	0	**1**
Cumulative Reward	-1	-1	-2	-2	-2	-3	-4	-5	-5	**-4**

4. Iterative Self-Improvement Training

Iteratively fine-tune the model to generate more compact circuits with Monte-Carlo tree search (MCTS) based self-improving.

- **Training stage**: fine-tune the model with parallel data
- **Improvement stage**: generate new parallel data with fine-tuned model and MCTS



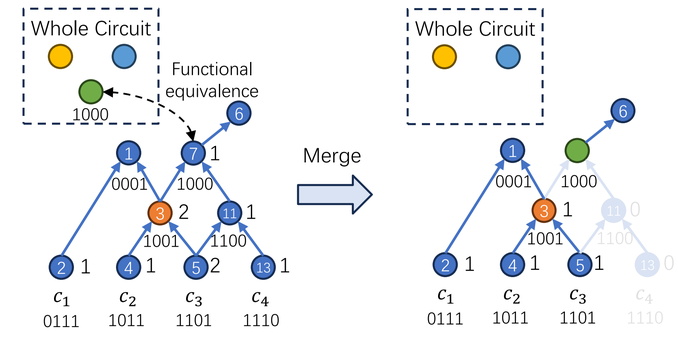
5. Cooperate Circuit Transformer with Fanout-Free Window Rewriting

We leverage the fanout-free window rewriting framework in [2], and apply our trained Circuit Transformer with self-improvement to replace each fanout-free window by a more compact one. MCTS can also be applied during the rewriting. Such replacement enables additional flexibility that the replaced circuit does not require to be equivalent to the sub-circuit it replaces.

[2] https://ieeexplore.ieee.org/document/10247727

6. Guided DAG-aware Rewriting

We refined the immediate reward to reflect the node merging in the rewriting process. Then MCTS is guided by the reward function to minimize the size of final rewritten circuit after node merging.



Step (Node ID)	1	2	3	4	5	6	7	8	9	10	11	12	13
Token	∧	c_1	∧	c_2	c_3	∧	∧	$\overline{\wedge}$	c_1	c_4	∧	c_3	c_4
Immediate Reward	-1	0	-1	0	0	-1	-1	-1	0	**1**	-1	0	**2**
Cumulative Reward	-1	-1	-2	-2	-2	-3	-4	-5	-5	**-4**	-5	-5	**-3**

7. Experiments

Methods	Avg. Improv.	Time cost
Drw Rewriting (ABC)	15.42%	<0.01s
MFFW Rewriting (in Python)	21.16%	1-300s
Ctrw (w/o self-improvement)	18.55%	1-250s
Ctrw	23.23%	1-285s
Ctrw with MCTS	26.02%	300-31000s
Ctrw with MCTS and guided DAG-aware Rewriting	30.19%	240-35000s

Our proposed approach successfully generated strictly feasible circuits (checked via cec), and demonstrated significant effectiveness in reducing circuit size. (However, the efficiency still needs to be improved, especially for MCTS)

Scan the QR code for the full paper, poster and future updates. Or visit:
https://snowkylin.github.io/publications

Correspondence email: xihan.li@cs.ucl.ac.uk