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

Speaker: Xihan Li (xihan.li@cs.ucl.ac.uk)

IWLS 2024

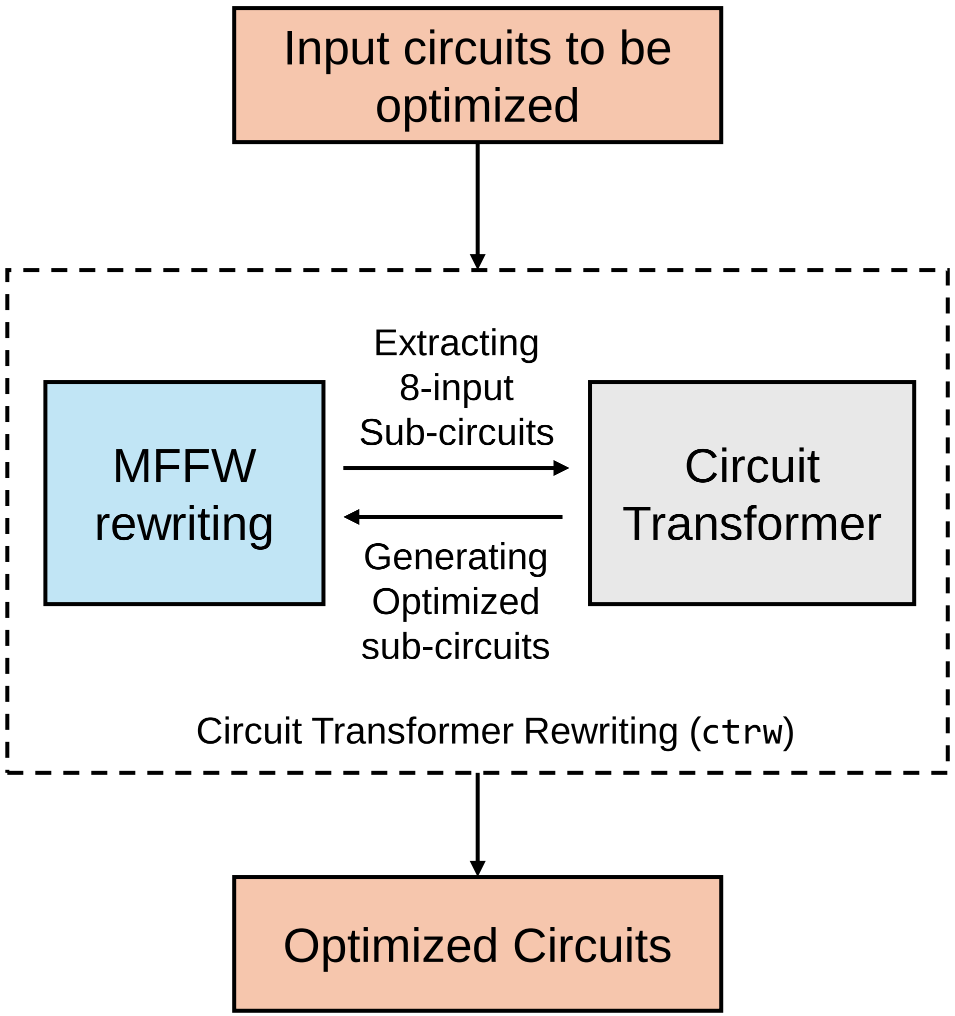
Introduction

A logic synthesis rewriter "`ctrw`" (Circuit Transformer Rewriting) powered by generative neural networks.

Highlight:

- `ctrw` guarantees preciseness (in contrast to ChatGPT that make mistakes occasionally).
- `ctrw` can improve itself (similar to self-play in AlphaGo, without human knowledge).
- `Ctrw` is effective (average improvement of ~30% while drw in abc is ~15%).

Note that this is a preliminary work. Currently `ctrw` runs slow.

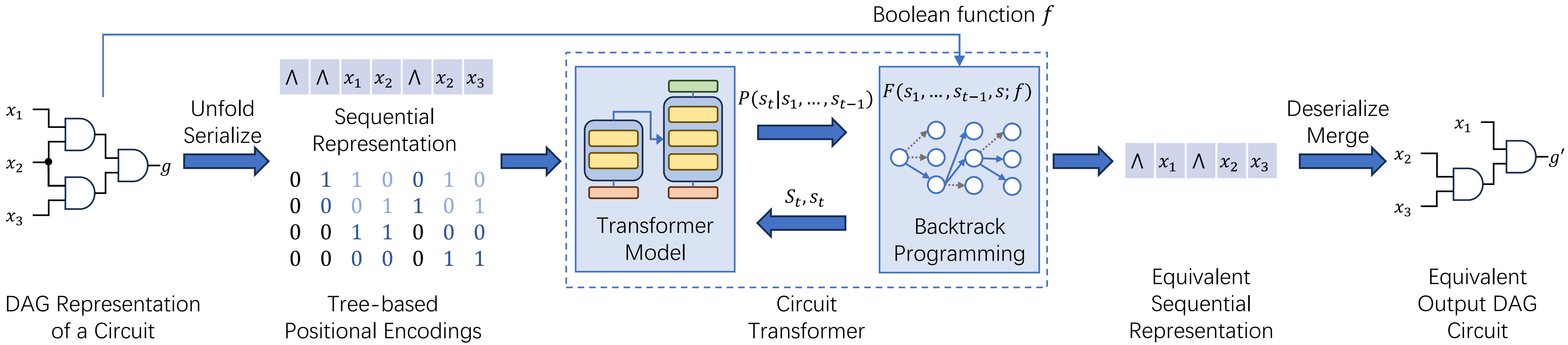


Circuit Transformer: Sequential Generation of Circuits with Equivalence Preserved

This work is based on Circuit Transformer [1], a generative neural model with 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



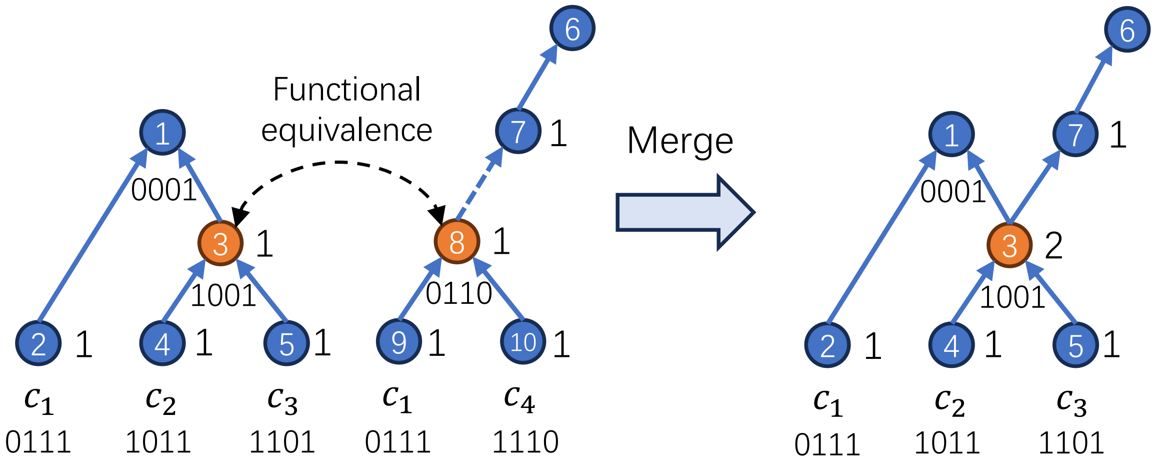
Circuit Size Minimization as a Markov Decision Process

We can minimize the number of AND gates of the generated circuit by attaching an immediate reward function $R(g_1, \ldots, g_t, g)$ to the generation of token s at step t

$$R(g_1, \ldots, g_t, g) = \Delta + \begin{cases} -1, g = \wedge \text{ or } g = \overline{\wedge} \\ 0, \text{otherwise} \end{cases}$$

Δ reflects the refinement of equivalent node merging.

We refine the generative neural model to maximize the cumulative reward (i.e., minimize the number of AND nodes)



Step (Node ID)	1	2	3	4	5	6	7	8	9	10
Token	$\wedge$	c_1	$\wedge$	c_2	c_3	$\wedge$	$\wedge$	$\overline{\wedge}$	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

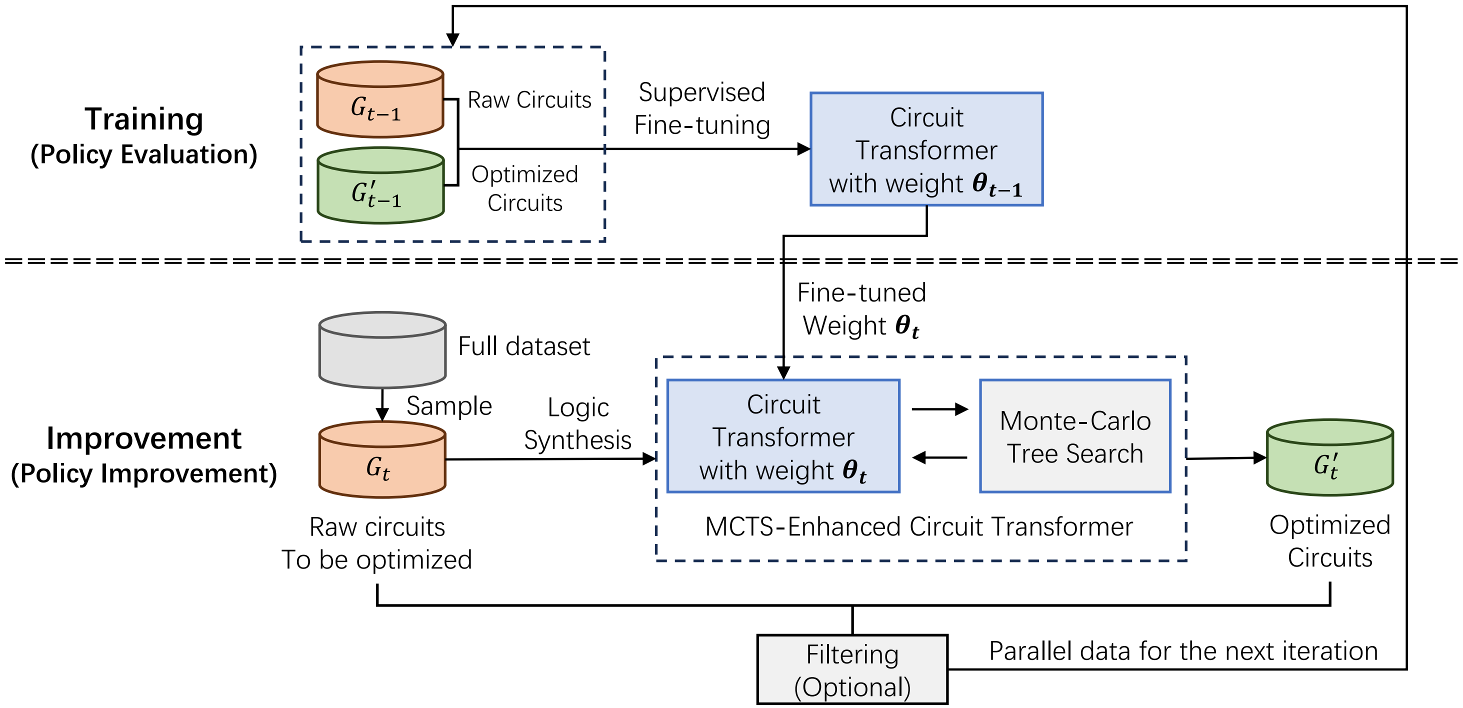
Negative number of AND nodes

(node 1, 3, 6, 7, 8)

(node 1, 3, 6, 7, node 8 is merged)

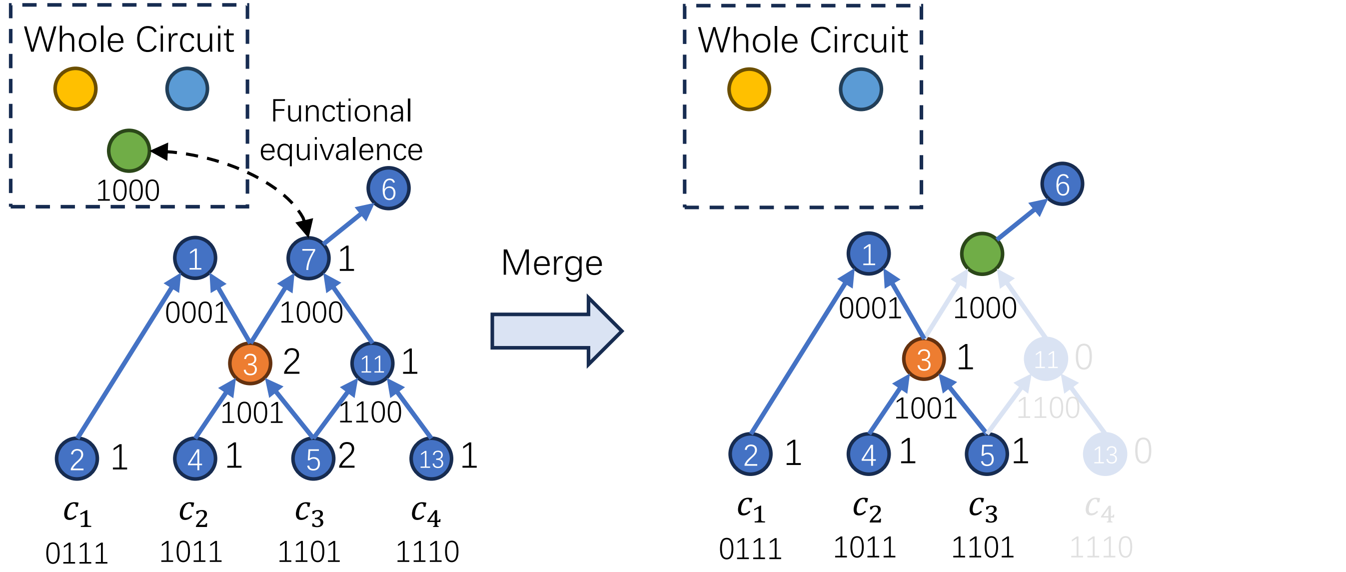
Iterative Self-Improvement Training

- 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 supervised data pairs (circuits before and after optimization)
- **Improvement stage**: generate new supervised data pairs with the fine-tuned model and MCTS



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 (rather than minimizing the sub-circuit).



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

Node 7 is replaced by the green node in the whole circuit. Node 11 is dereferenced after replacement.

Experiments

On 22 small circuits (#(AND) < 100) generated from IWLS 2023 contest benchmark.

Our proposed approach successfully generated strictly feasible circuits (checked via cec), and demonstrated significant effectiveness in reducing circuit size.

However, as a preliminary work, the scalability and efficiency still needs to be improved, especially for MCTS.

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

Thank you!

Xihan Li

Department of Computer Science

University College London

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