AlphaSyn: Logic Synthesis Optimization with Efficient Monte Carlo Tree Search

Zehua Pei[1], Fangzhou Liu[2], Zhuolun He[1], Guojin Chen[1], Haisheng Zheng[2], Keren Zhu[1], Bei Yu[1]

[1]The Chinese University of Hong Kong
[2]Shanghai Artificial Intelligence Laboratory

Introduction

- Recipe Sequence Generation is essential for logic synthesis optimization, enabling better Quality of Results.
- Previous works typically have low efficiency and are stuck at local optima.
- We propose a logic synthesis optimization framework, AlphaSyn, that incorporates a domain-specific Monte Carlo tree search (MCTS) algorithm.

Keywords: Logic synthesis, MCTS, GNN

Summary of previous works

Classification & Prediction

- Classify [1] or predict [2] the QoR of synthesis sequences.
- A large dataset is required for training and evaluation.
- The accuracy is limited and uncertain.

Sequence Generation

- Generate the sequence with specific optimization objectives by RL [3,4], BO [5] or with heuristics [6].
- RL and BO based methods are lack of enough exploration by performing in a "forward" process, where the sequence is generated as trajectory and evaluated as a whole.
- The methods with heuristics always explore in the reduced search space, which results in local optimum.

Motivation Analysis

Table 1. Results for greedy and modified greedy algorithms with design `bfly`.

Greedy	`rw`	`rfz`	`rwz`	`rf`	`rsk6`	`rwz`	#Final
#TNodes	2083	671	319	250	172	91	25324
Modified_Greedy	`rw`	`b`	`rf`	`rsk6`	`rwz`	`rf`	#Final
#TNodes	2083	**130**	963	265	174	140	25155

Earlier Transformation matters

- The earlier transformations work effectively during the optimization and dominate the synthesis sequence's performance.
- In AlphaSyn, the "accumulated statistics" of the search tree is closely related to this observation.

Earlier Transformation matters

- Restructuring logic, allowing more node reduction possibilities, which is considered to be the long-term effect.
- Taking both immediate and long-term effects into account is necessary to select each synthesis transformation and optimize them sequence.

Reference

[1] C. Yu, H. Xiao, and G. De Micheli, "Developing synthesis flows without human knowledge," in ACM/IEEE Design Automation Conference (DAC), 2018.
[2] A. B. Chowdhury, B. Tan, R. Carey, T. Jain, R. Karri, and S. Garg, "Bulls-Eye: Active few-shot learning guided logic synthesis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2022.
[3] A. Hosny, S. Hashemi, M. Shalan, and S. Reda, "DRiLLS: Deep reinforcement learning for logic synthesis," in IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC), 2020.
[4] K. Zhu, M. Liu, H. Chen, Z. Zhao, and D. Z. Pan, "Exploring logic optimizations with reinforcement learning and graph convolutional net- work," in ACM/IEEE Workshop on Machine Learning CAD (MLCAD), 2020.
[5] A. Grosnit, C. Malherbe, R. Tutunov, X. Wan, J. Wang, and H.B. Ammar, "Boils: Bayesian optimisation for logic synthesis," in IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE), 2022.
[6] W. L. Neto, Y. Li, P.-E. Gaillardon, and C. Yu, "FlowTune: End- to-end automatic logic optimization exploration via domain-specific multi-armed bandit," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2022.
[7] G. Zhou and J. H. Anderson, "Area-driven FPGA logic synthesis using reinforcement learning," in IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC), 2023.

Framework overview

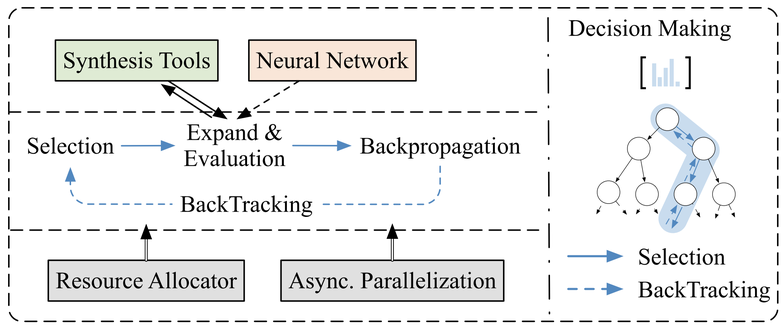

Figure 1. The overview of AlphaSyn.

- **Customized MCTS for Logic Synthesis Optimization.** The customized algorithm SynUCT is developed to take the immediate-long term effect into consideration combined with the exploration-exploitation trade-off.
- **Stable Learning Strategies.** Several learning strategies are proposed to enhance the stability of MCTS with a neural network (PQnet), where the past observations is used in present sampling.
- **Acceleration for MCTS.** A resource allocator and an asynchronous parallel algorithm are developed, then the runtime is reduced and the performance is maintained.

Domain-Specific MCTS

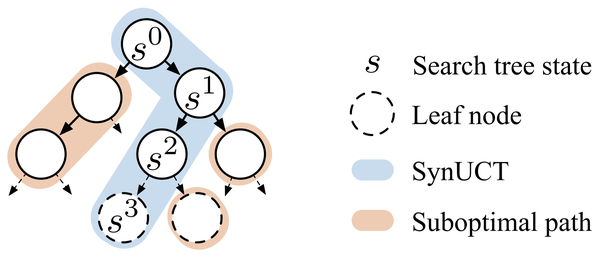

Figure 2. The selection phase in MCTS. From the root s^0 of the search tree, transformations are selected iteratively based on SynUCT.

Selection
SynUCT:

$$a^t = \arg\max_{a\in\mathcal{A}}(Q^{t,a} + R^{t,a} + U^{t,a}). \tag{1}$$

$Q^{t,a}$: long-term return. $R^{t,a}$: immediate reward.
$U^{t,a}$: balance mechanism:

$$U^{t,a} = c_{puct}\cdot P^{t,a}\cdot\frac{\sqrt{N^t}}{N^{t,a}+1}. \tag{2}$$

Expansion & Evaluation
Define the reward $R^{\mathcal{T}}$:

$$R^{\mathcal{T}} = \mathrm{sgn}(O^{\mathcal{T}-1} - O^{\mathcal{T}})\cdot\sqrt{\frac{|O^{\mathcal{T}-1} - O^{\mathcal{T}}|}{baseline}}. \tag{3}$$

Multi-objecitves reward:

$$R^{\mathcal{T}} = (1-\beta)\cdot R_1^{\mathcal{T}} + \beta\cdot R_2^{\mathcal{T}}. \tag{4}$$

Then new nodes are **expanded** with initial statistics:

$$\{N^{\mathcal{T},a} = 0, Q^{\mathcal{T},a} = 0, P^{\mathcal{T},a} = p^{\mathcal{T},a}\}. \tag{5}$$

Backpropagation
Update the node visit count:

$$N^t \longleftarrow N^t + 1. \tag{6}$$

Update the long-term return Q^t, all $t < \mathcal{T}$:

$$Q^t \longleftarrow \lambda\cdot\max_{a\in\mathcal{A}}(Q^{t,a} + R^{t,a}). \tag{7}$$

Decision Making
Take action T_i depends on the statistics:

$$T_i = \arg\max_{a\in\mathcal{A}}(Q^{0,a} + R^{0,a}). \tag{8}$$

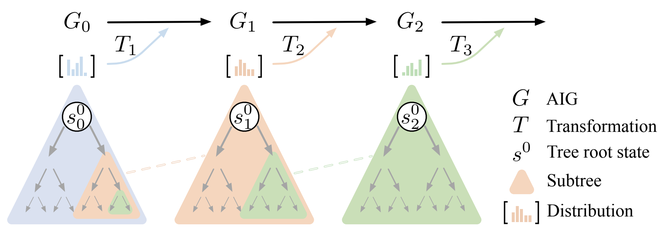

Figure 3. Decision making in AlphaSyn.

Learning Strategies

Collecting data via *self-syn*, the accumulated Q and exponential distribution of (Q+R) are stored:

$$\pi^{0,a} = \frac{(Q^{0,a} + R^{0,a})^{1/\kappa}}{\sum_{b\in\mathcal{A}}(Q^{0,b} + R^{0,b})^{1/\kappa}}. \tag{9}$$

Input and Output of the model PQnet (f_θ):

$$(\vec{p}^{\mathcal{T}}, Q_{es}^{\mathcal{T}}) = f_\theta(s_i^{\mathcal{T}},\ a_i^{\mathcal{T}-1},\ (\mathcal{T}-1)+i). \tag{10}$$

Loss function:

$$L_{\text{total}} = L_{\text{CE}}(\vec{p}^0, \vec{\pi}^0) + L_{\text{MSE}}(Q_{es}^0, Q^0). \tag{11}$$

We design the PQnet based on SAGEConv and the self-attention pooling SAGPool.

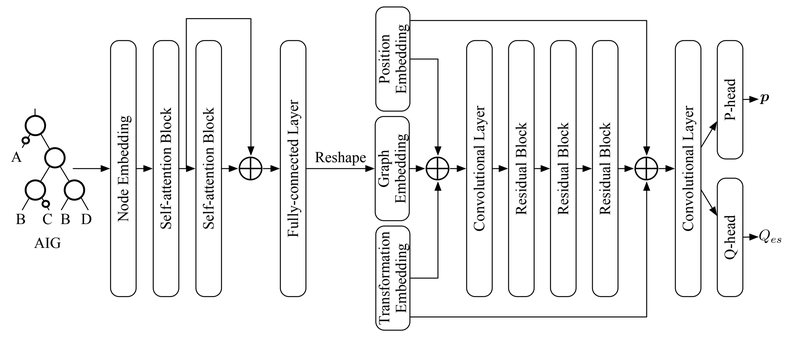

Figure 4. The overview of PQnet.

Acceleration Techniques for AlphaSyn

Resource Allocator.
Search times are gradually decreased as the index increase:

$$n_{sche}^i = n_{init} - D\cdot(i-1). \tag{12}$$

Asynchronous Parallelization.
Conditional **path-blocking-releasing** Asynchronous Parallelization, **Par-SynUCT**:

$$a_{par}^t = \arg\max_{a\in\mathcal{A}}(Q^{t,a} + R^{t,a} + U^{t,a} + B^{t,a}). \tag{13}$$

Experienments

Table 2. Comparison with FlowTune [6] for technology mapping on Nangate 45nm library.

Design	Flowtune [6] Area (μm^2)	Flowtune [6] rt (s)	AlphaSyn w/o nn $\hat{\text{Area}}$ (μm^2)	AlphaSyn w/o nn $\bar{\text{rt}}$ (s)	AlphaSyn w/ nn $\hat{\text{Area}}$ (μm^2)	AlphaSyn w/ nn $\bar{\text{Area}}$ (μm^2)	AlphaSyn w/ nn $\bar{\text{rt}}$ (s)
bfly	16881.8	969.95	16089.0	**450.78**	**15493**	**15959.2**	830.20
dscg	16393.2	913.20	16142.2	**411.80**	**15743**	**16097.1**	958.01
fir	16323.5	922.73	15876.9	**431.70**	**15336**	**15724.4**	896.26
ode	9302.5	479.77	9138.4	**272.59**	**9101**	**9151.2**	710.97
or1200	6731.4	228.24	6756.6	**153.92**	**6689.6**	**6729.1**	531.53
syn2	17246.4	880.14	16410.6	**490.37**	**16051.5**	**16078.1**	943.32
Average	13813.1	732.34	13402.3	**368.53**	**13069.0**	**13289.9**	811.72
Ratio	1.000	1.000	0.970	**0.503**	**0.946**	**0.962**	1.108

Table 3. Comparison with DRiLLS [3] and ASPDAC'23 [7] for FPGA mapping.

Design	DRiLLS [3] LUTs	ASPDAC'23 [7] LUTs	AlphaSyn w/o nn $\hat{\text{LUTs}}$	AlphaSyn w/o nn $\bar{\text{rt}}$ (s)	AlphaSyn w/ nn $\hat{\text{LUTs}}$	AlphaSyn w/ nn $\bar{\text{LUTs}}$	AlphaSyn w/ nn $\bar{\text{rt}}$ (s)
max	694	687.8	680.5	74.59	**674**	**680**	342.56
adder	**244**	**244**	**244**	62.74	**244**	**244**	368.74
cavlc	112.2	111.3	106.8	53.14	**106**	**106**	321.12
ctrl	**28**	**28**	**28**	38.81	**28**	**28**	341
int2float	42.6	42.3	39.2	56.62	**39**	**39**	332.82
router	70.1	69.5	65.6	24.59	**65**	**65**	320.43
priority	133.4	142.9	135.6	59.15	**131**	135	350.11
i2c	292.1	289.32	280.6	47.78	**272**	**280**	373.46
sin	1441.5	1438	1439.7	91.02	**1435**	**1438**	406.19
square	3889.4	3889	3877	166.27	**3875**	**3877**	523.26
sqrt	4708	4685.3	**4415**	269.59	**4415**	**4415**	589.93
log2	7583.6	7580.1	**7580**	365.77	**7580**	**7580**	706.96
multiplier	5678	5672	5687.5	245.75	**5670.5**	**5672**	620.53
voter	1834.7	1678.1	1538.8	111.44	**1534**	**1537.4**	470.64
div	7944.4	7807.1	6685.3	244.04	**5088.4**	**6650.1**	712.57
mem_ctrl	10527.6	10309.7	9567.7	71.63	**9211.5**	**9513.2**	659.67
Average	2826.5	2792.2	2648.2	123.93	**2523.0**	**2641.2**	464.99
Ratio	1.000	0.988	0.937	-	**0.893**	**0.934**	-