DESIGN, AUTOMATION & TEST IN EUROPE
25 - 27 March 2024 · Valencia, Spain
The European Event for Electronic System Design & Test

CBTune: Contextual Bandit Tuning for Logic Synthesis

Fangzhou Liu [♠] Zehua Pei [♠] Ziyang Yu [♠] Haisheng Zheng [♡]
Zhuolun He [♠♡] Tinghuan Chen [♣] Bei Yu [♠]

[♠] The Chinese University of Hong Kong
[♡] Shanghai Artificial Intelligence Laboratory
[♣] The Chinese University of Hong Kong, Shenzhen

Mar. 25, 2024

上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

Outline

1. Introduction
2. CBTune
3. Evaluation

Introduction

Introduction

Synthesis Flow is a set of synthesis transformations that apply iteratively to the AIG.

- Use typical `ABC` commands like *balance(b)*, *rewrite(rw)*, *refactor(rf)*, *resubstitute(rs)*, ...

Introduction

Synthesis Flow is a set of synthesis transformations that apply iteratively to the AIG.

- Use typical `ABC` commands like *balance(b)*, *rewrite(rw)*, *refactor(rf)*, *resubstitute(rs)*, ...

Problems

- Exponential solution space.
- Same synthesis flow works differently across designs.

Synthesis Flow is a set of synthesis transformations that apply iteratively to the AIG.

- Use typical `ABC` commands like *balance(b)*, *rewrite(rw)*, *refactor(rf)*, *resubstitute(rs)*, ...

Problems

- Exponential solution space.
- Same synthesis flow works differently across designs.

Limitations of Recent Works

- Timing-intensive and resource-demanding.
- Lack of transferability; Training is hard to converge.

Bandit Model Definition

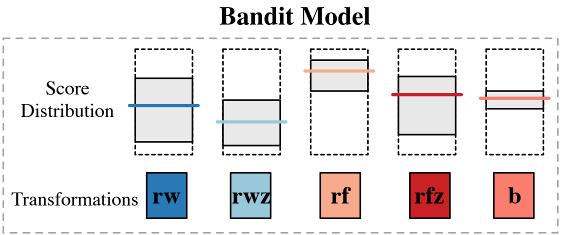

Figure: Bandit Model for Synthesis Flow Generation.

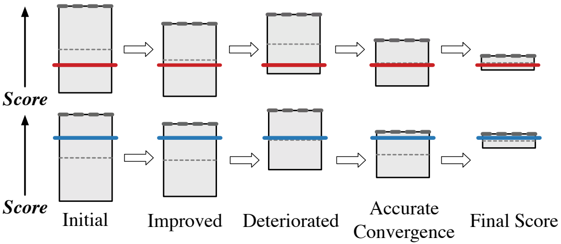

Figure: Score Iterations for Each Arm in Bandit Model.

- **Bandit Model**
  A Decision-Making Framework focused on maximizing rewards by balancing exploration and exploitation.
- **Upper Confidence Bound**

$$\mathrm{UCB}_i(t) = \hat{x}_i(t) + \sqrt{\frac{2\ln(t)}{T_{i,t}}} \quad (1)$$

CBTune

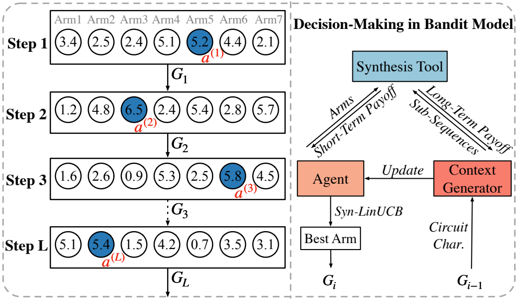

Figure: CBTune Framework Overview.

- **Action Space**
  $\mathcal{A}$ = {*resub*, *resub -z* , *rewrite* , *rewrite -z* , *refactor*, *balance* }, denoted as $\mathcal{A} = \{a_1, a_2, a_3, ..., a_6\}$.
- **Reward**
  The short-term payoff (a single-step) of each arm: The number of AIG nodes or 6-LUTs.
- The highest-scoring one $a^{(i)}$ is selected at each step, and the final synthesis flow has the ordered sequence of transformations $a^{(1)}, a^{(2)}, ..., a^{(L)}$.

CBTune Framework

Contextual Generator

- Observe features of $a \in \mathcal{A} : \boldsymbol{x}_{t,a} = [\boldsymbol{x}_a^c, \boldsymbol{x}_{t,a}^l] \in \mathbb{R}^d$;

Feature	Description	Example
Circuit Characteristics ($\boldsymbol{x}^c$)	AIG information extracted by applying $a_j^{(i)}$ ($j \leq n$) to G_{i-1} using yosys and ccirc.	#Number of wires/cells/nots, #Maximum delay, #Number of combinational nodes, #Number of high degree comb, #Fanout distribution ......
Long-term Payoff of the Arm ($\boldsymbol{x}^l$)	Random DSE result: Employ "arm" $a^{(i)}$ as the first transformation to produce m random subsequences of length l, and then utilize these subsequences to obtain synthesis results.	Arm: *rewrite* (*rw*); $l = 5$; $m = 3$; {**rw**,rw,rfz,rf,rs} $\rightarrow$ Nodes: 28010, Level: 66 Arm: *refactor* (*rf*); $l = 4$; $m = 2$; {**rf**,b,rf,rw} $\rightarrow$ Nodes: 28324, Level: 67

Agent (Syn-LinUCB) [1]

- Update hyperparameter α by $\alpha = 1.0 + \sqrt{\frac{\log(2.0/\delta)}{s_a}}$ (2)
- Update the decision parameter by $\boldsymbol{\theta}_a = \boldsymbol{A}_a^{-1}\boldsymbol{b}_a$ (3)
- Calculate the weighted context $\boldsymbol{x}_{t,a}^w = \boldsymbol{x}_{t,a}\boldsymbol{w}$ (4)
- Update score by $p_{t,a} = \boldsymbol{\theta}_a^\top \boldsymbol{x}_{t,a}^w + \alpha\sqrt{\boldsymbol{x}_{t,a}^{w\ \top} \boldsymbol{A}_a^{-1} \boldsymbol{x}_{t,a}^w}$ (5)
- Update the parameters $\boldsymbol{A}_{a_t}$ and $\boldsymbol{b}_{a_t}$ of the chosen arm a_t by $\boldsymbol{A}_{a_t} = \boldsymbol{A}_{a_t} + \boldsymbol{x}_{t,a_t}\boldsymbol{x}_{t,a_t}^\top$, $\boldsymbol{b}_{a_t} = \boldsymbol{b}_{a_t} + r_{t,a}\boldsymbol{x}_{t,a_t}$ (6)

[1] **Lihong Li et al. (2010). "A Contextual-bandit Approach to Personalized News Article Recommendation".** In: *The Web Conference*.

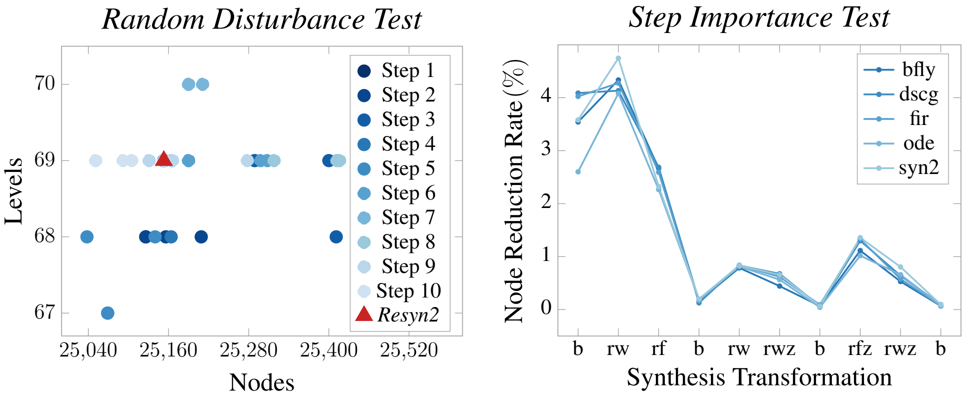

Figure: Analysis of Transformation Effectiveness in the Synthesis Flow.

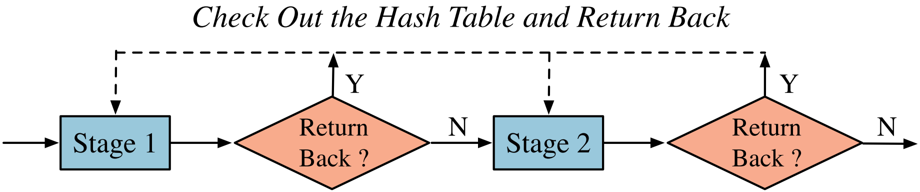

Figure: The Return-Back Mechanism in CBTune.

1. Tuning Iteration Count; Early Stop
2. Return-back Mechanism

Evaluation

AIG Node Optimization

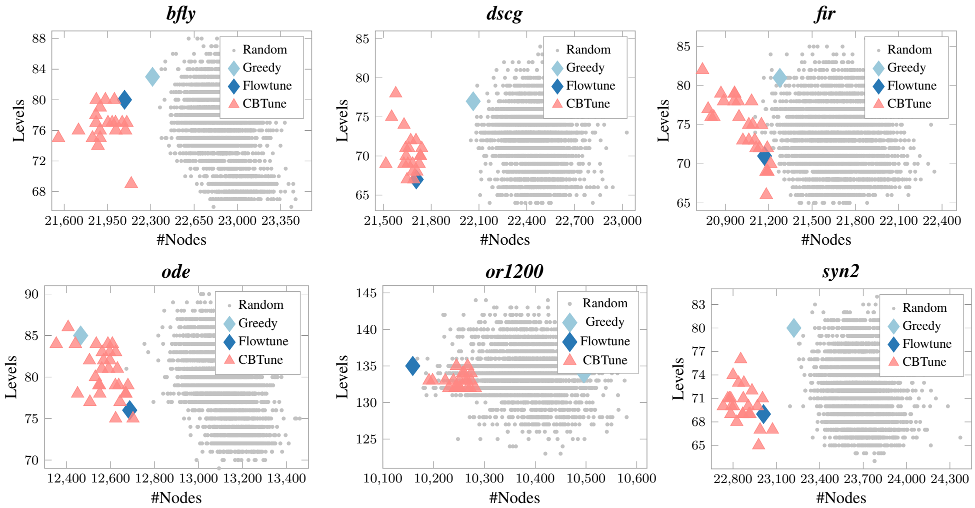

Figure: Performance Comparison between *CBTune* & FlowTune.

6-LUTs Optimization

Benchmark	Initial	Greedy	FlowTune[2]		CBTune		
	#LUTs	#LUTs	#LUTs	$\tau(m)$	$\#\mathrm{L\hat{U}Ts}$	$\#\mathrm{L\bar{U}Ts}$	$\tau(m)$
bfly	9019	8269	8216	76.47	**7962**	8086.03	**29.63**
dscg	8534	8313	8302	77.15	**7981**	8119.84	**30.44**
fir	8646	8385	8094	74.23	**7820**	7977.38	**27.6**
ode	5244	5316	5096	34.83	**4920**	5046.71	**17.32**
or1200	2776	2748	2747	20.08	**2731**	2754.07	**15.62**
syn2	8777	8669	8603	81.33	**8234**	8360.53	**31.67**
GEOMEAN	6631.20	6464.69	6364.89	54.04	**6166.39**	6271.82	**24.48**
Ratio Avg.	1.000	0.975	0.960	1.000	**0.930**	0.946	**0.453**

Table: Performance Comparison Between *CBTune* & FlowTune.

[2] **Walter Lau Neto et al. (2022). "FlowTune: End-to-End Automatic Logic Optimization Exploration via Domain-specific Multi-armed Bandit". In: *IEEE TCAD*.**

6-LUTs Optimization

Benchmark	Initial	Greedy	DRiLLS [3]		RL4LS*		CBTune	
	#LUTs	#LUTs	#LUTs	$\tau(m)$	#LUTs	$\tau(m)$	#LUTs	$\tau(m)$
max	721	697	694	32.58	687.8	54.34	**684.25**	**6.01**
adder	249	**244**	**244**	24.05	**244**	10.05	**244**	**5.97**
cavlc	116	115	112.2	26.02	111.3	3.22	**111**	**2.37**
ctrl	29	**28**	**28**	24.25	**28**	2.85	**28**	**0.59**
int2float	47	46	42.6	21.7	42.3	2.81	**40**	**2.76**
router	73	67	70.1	22.01	69.5	3.07	**68.11**	**2.32**
priority	264	146	**133.4**	23.32	142.9	5.9	138.86	**3.41**
i2c	353	291	292.1	25.17	289.32	7.55	**283.11**	**3.61**
sin	1444	1451	1441.5	51.15	**1438**	20.1	1441.67	**9.71**
square	3994	3898	3889.4	130	3889	72.88	**3882.11**	**25.99**
sqrt	8084	4807	4708	147.64	4685.3	196.15	**4607**	**36.51**
log2	7584	7660	7583.6	198.6	7580.1	125.28	**7580**	**41.27**
multiplier	5678	5688	5678	180.84	**5672**	187.81	5679.75	**29.08**
voter	2744	1904	1834.7	84.43	**1678.1**	330.48	1682.25	**11.46**
div	23864	4205	7944.4	259.75	7807.1	482	**4180.91**	**25.58**
mem_ctrl	11631	**10144**	10527.6	229.33	10309.7	1985.84	10242.57	**45.81**
GEOMEAN	926.59	732.69	753.49	59.48	748.34	34.54	**712.83**	**8.37**
Ratio Avg.	1.000	0.791	0.813	1.000	0.808	0.581	**0.769**	**0.141**

Table: Performance Comparison Between *CBTune* & NN-Enhanced RL. * *Last10 in RL-PPO-Pruned* [4].

[3] **Abdelrahman Hosny et al. (2020). "DRiLLS: Deep Reinforcement Learning for Logic Synthesis".** In: *Proc. ASPDAC*.

[4] **Guanglei Zhou and Jason H Anderson (2023). "Area-Driven FPGA Logic Synthesis Using Reinforcement Learning".** In: *Proc. ASPDAC*.

THANK YOU!