# Automatic Electronic Circuit Topology Generation Based on Monte Carlo Tree Search

* Shaoze Fan[1*], Ningyuan Cao[1,3*], Shun Zhang[2*], Jing Li[1**], Xiaoxiao Guo[2], Xin Zhang[2**]

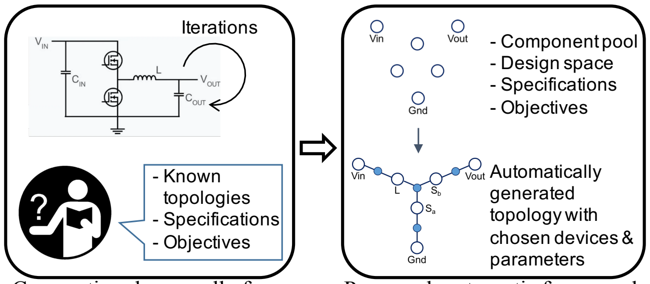
[1] New Jersey Institute of Technology, [2] IBM T. J. Watson Research Center, [3] University of Notre Dame

## Abstract

The tidal waves of modern electronic/electrical devices have led to increasing demands for ubiquitous application-specific electronic circuit designs. A conventional manual design procedure of electronic circuits is computation-and-labor-intensive, which involves selecting and connecting component devices, tuning component-wise parameters and control schemes, and iteratively evaluating and optimizing the design. To automate and speed up the design process, we propose a framework that is capable of creating electronic circuit topologies automatically from the design specifications. Specifically, the framework embraces upper-confidence-bound-tree-based (UCT-based) search to automate topology space exploration with circuit design specification-encoded reward signals. Moreover, our UCT-based approach can exploit small offline data via the specially designed default policy to accelerate topology space exploration. Further, it utilizes a hybrid circuit evaluation strategy to substantially reduces design evaluation costs. Empirically, we took power converter circuits as an example task and demonstrated that our framework could generate energy-efficient circuit topologies for various target voltage conversion ratios. Compared to existing automatic topology optimization strategies, the proposed method is much more computationally efficient — it can generate topologies with the same quality while being up to 67% faster. Moreover, using the proposed framework, new circuit topology that is different from existing topologies can be generated, which leads to potentially innovative circuits that have not been discovered by human designers before.

## Introduction

Electronic circuits are ubiquitous in electronic/electrical devices. With the proliferation of customized electrical systems (Nikolic, Alon, and Asanovic 2018), such as electric vehicles, self-powered IoT, wearable/implantable biosensors, the need for custom electronic circuits is rapidly increasing. Take the power converter circuits as an example, the design specifications, such as voltage conversion ratio, power efficiency, and output ripple, differ significantly from application to application. To meet these specifications, conventional manual circuit optimization is time-consuming and relies heavily on existing circuit topologies, as illustrated in Figure 1. The expensive design process has dramatically hindered the development of novel power converters for fast-paced and innovative custom designs. Hence, there is a pressing need for an automatic circuit design framework that can efficiently search and generate high-quality power converter topologies from the design specifications.



Conventional: manually from known topologies to an optimized design

Proposed: automatic framework from specs to topologies with chosen devices & parameters

Figure 1: Given a custom power converter design task, the conventional manual approach relies heavily on known topologies and is labor-intensive, computationally expensive, and time-consuming. In contrast, our automatic power converter design framework can explore the design space more effectively, thereby immensely decreasing the development time and cost without compromising the performance.

However, how to automate the power converter topology design remains a challenging task. Firstly, topology generation for electronic circuits or integrated circuits (IC) lacks thorough investigation. The state-of-art analog/mixed-signal (AMS) IC design automation mainly addresses device sizing or parameter optimization for a fixed circuit structure (Kim

[*]The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0001210 and in part by the U.S. National Science Foundation, under Grant Number CNS–1948457. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

[*]Equal contribution. [**]Corresponding authors: `jingli@njit.edu`, `xzhang@us.ibm.com`.

et al. 2004; Singh et al. 2012; Wang, Orshansky, and Caramanis 2014; Lyu et al. 2018; Yang et al. 2020; Lee and Oliehoek 2020; Wang et al. 2020; Zhou et al. 2020). People have also investigated automation methods to accelerate the physical implementation of AMS ICs when schematic/topology design is already done (Hakhamaneshi et al. 2019b; Chang et al. 2018; Liu et al. 2020; Han et al. 2020; Chen et al. 2020). More recently, researchers have started looking into circuit synthesis (Zhang, He, and Katabi 2019; Zhao and Zhang 2020; Rojec, Burmen, and Fajfar 2019). But some of them require substantial domain knowledge, and others explore the enormous topology space via exhaustive search, metaheuristic search, or gradient descent, which may not be as efficacious.

Fundamentally, automated topology generation is difficult, as it faces challenges due to the immense search space and severe data discontinuity. Further, metaheuristic search strategies may get stuck in a restricted set of topologies and thus output sub-optimal results unless the number of random samples becomes large enough. Moreover, unlike device parameters such as transistor sizes, a small shift in the component connections of one topology will very likely lead to significant changes in the circuit performance. As such, search algorithms or optimization methods that rely on continuity between “similar topologies”, such as genetic search or gradient descent, may become less effective in reducing search efforts.

Finally, it is time-consuming to evaluate the performance of generated topologies properly. Power converters are often nonlinear and dynamically controlled switching circuits, which require long simulations to reach steady states. The conventional Spice simulation (Nenzi and Vogt 2011) is able to provide high-fidelity evaluation results, but this comes with the cost of long simulation time. The evaluation cost per topology can be as high as minutes, making the topology exploration process prohibitively time-consuming.

**Contribution.** We present a design automation framework for power converter circuit to address the above challenges. The main contributions are as follows:

- We propose the first automatic power converter design framework that intelligently explores the power converter topology space and generates high-quality candidate circuits based on custom design specifications. As shown in Figure 1, our framework can efficiently locate well-performing topologies with appropriate control schemes and also has the potential to generate novel topologies under specific design constraints.
- For the first time, our framework applies the upper-confidence-bound-tree (UCT) to circuit topology generation. We construct the UCT structure to sufficiently capture the semantic of topologies to explore the topology space more efficiently. Moreover, this UCT structure is able to exploit offline knowledge, which is obtained from a few suitable topologies with smaller sizes and encapsulated into our specially designed default rollout policy, and further accelerate the topology space exploration.
- As the long-running circuit evaluation is the bottleneck of fast topology exploration, we detect isomorphic topologies and adopt a hybrid circuit evaluation approach. A State-Space Averaging method is used during the exploration, which reduces the time cost of circuit evaluation by orders of magnitude, and a high-fidelity Spice transient simulation is used to validate circuit candidates generated by the exploration and filter out the over-optimistic ones.
- We conduct extensive experiments on 5-component (13-port) power converter design tasks. Evaluation results demonstrate that our proposed automatic framework can produce energy-efficient circuits for varying voltage conversion ratios. Furthermore, compared to baseline strategies (i.e., genetic search and random search algorithms) adapted from other circuit design tasks, our framework can generate constraint-satisfied and highly efficient topologies while needing fewer queries to circuit evaluation. Hence, it is up to 70%, 56%, and 50% faster than the baseline strategies for the experimented buck, boost, and buck-boost converter design tasks, respectively.

## Background

Traditional manual design routines are inherently time-consuming and rely heavily on domain expertise. To reduce the cost and improve the design quality, mainstream research about circuit design automation are three folded: (1) automating the device parameter optimization for known circuit topologies; (2) automating the physical implementation for known circuit topologies and parameters; (3) automating the circuit synthesis that directly generates topologies.

**Parameter Optimization.** This is the oprimization for predetermined topologies. For example, (Kim et al. 2004) proposed geometric-programming-based optimization, (Singh et al. 2012; Wang, Orshansky, and Caramanis 2014) used regression and convex/polynomial optimization, and (Lyu et al. 2018) applied a Bayesian optimizer, (Yang et al. 2020; Lee and Oliehoek 2020) both adopted model-based reinforcement learning to find the optimal device parameter combinations. Additionally, (Wang et al. 2020) encoded circuits using graph convolutional neural networks to transfer the parameter optimization knowledge learned between two topologies or between technology nodes of the same circuit.

**Physical Implementation.** Analog generators were proposed in (Hakhamaneshi et al. 2019b; Chang et al. 2018), which directly build analog circuit layouts. (Liu et al. 2020) applied a data-driven approach to check layout symmetry. A feed-forward equalization transmitter layout generator was introduced in (Han et al. 2020), which significantly reduced layout time. (Chen et al. 2020) presented a novel detailed routing framework to address the sensitive net coupling issues.

**Topology Optimization.** Recent works have started investigating circuit topology optimization. (Zhang, He, and Katabi 2019) proposed a graph neural network (GNN) model that learns to simulate the electromagnetic properties of distributed circuits. However, an “edge” in a circuit topology is determined by the physical distance between two nodes and its impact on the circuit is continuous and decomposable. In comparison, the edge in a power converter topology is determined by whether the two components are connected in the circuit, and removing one edge may utterly change the performance of the circuit (e.g., from valid to invalid). Hence, such gradient back-propagation approach cannot be directly applied to the power converter design task. For topology synthesis for large analog integrated circuits, (Zhao and Zhang 2020) presented a graph-grammar-based circuit topology representation. This work focused on designing meaningful decomposition rules and circuit formation rules that domain experts manually add while the generation is performed using an *exhaustive search* within the representation space. For searching in the circuit topology space, (Rojec, Burmen, and Fajfar 2019) developed a *genetic search* algorithm, where the device types of components in the topology is essentially fixed. We extend it to allow changing component types and compare it with our proposed framework.

**AI-assisted Power Electronic Design.** The advancements of artificial intelligence (AI) and IC design automation have introduced marvelous opportunities for power electronic circuit design automation (Zhao, Blaabjerg, and Wang 2021). Existing works have been looking into methods that greatly reduce circuit evaluation time with lower computational efforts (Zhang, He, and Katabi 2019; Zhang et al. 2020; Wu et al. 2019; Lu et al. 2020). Researchers have also applied AI techniques and shown some successes in modeling and optimizing other aspects of power electronic systems, such as component model, system parameters, and post-layout performance (Wang et al. 2020; Zhou et al. 2020; Hakhamaneshi et al. 2019a; Settaluri et al. 2020).

## Problem Statement

We investigate the automatic power converter design problem with topology generation, device type selection, and control parameter tuning. This section first describes the custom design specifications of power converters considered in this work, followed by their topological representation and parameters. Next, we formulate the problem as an optimization problem with encoded custom design specifications.

**Custom Design Specifications.** We consider two primary design metrics of power converters, namely, *voltage conversion ratio* $\gamma$ and *power conversion efficiency* $\eta$, as shown in Figure 2. The voltage conversion ratio is the ratio of the output voltage to the input voltage, i.e., $\gamma = V_{\text{out}}/V_{\text{in}}$, and is the main constraint for the generated power converter. Generating power converter designs with higher power conversion efficiencies is the optimization goal of the custom design task. Other constraints include the number of components in the converter topology and the types of available devices. The design task evaluated in this work only considers devices including capacitors `C`, inductors `L`, phase-I switches $\mathtt{S}_a$, and phase-II switches $\mathtt{S}_b$, but it can be easily extended to other device types.

**Topological Representation and Parameters.** A candidate power converter design consists of a *topological representation* $s$ and a *switching control parameter* $d$. Specifically, the

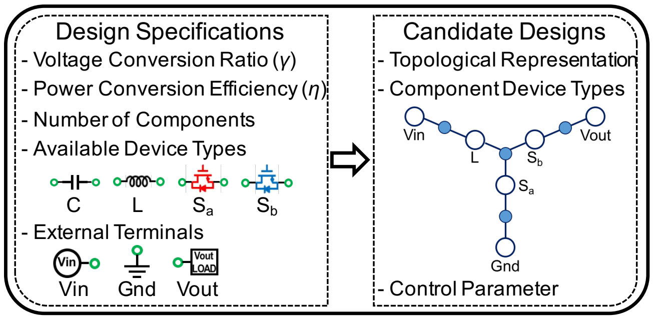


Figure 2: Custom power converter design task from design specifications to candidate designs.

topological representation contains a set of components with ports and edges connecting the ports. Each component has a device type (from the set of available devices) with device parameters (e.g., inductance, capacitance, and transistor dimensions) and two ports (i.e., left and right ports). Note that each component is nondirectional — switching all the connections of its two ports results in the same circuit, despite needing different indexes to distinguish the two ports in the topological representation. Additionally, there are three *external terminal ports*: the input voltage port `Vin`, the output voltage port `Vout`, and the ground port `Gnd`. The edges in the topological representation specify the connections between the components' ports and the terminal ports. The switching control parameter specifies the duty cycle of a candidate design, which often affects the output voltage. In this work, the design task involves designing the device types of components, the edges connecting ports, and the control parameter, while the device parameters for each device type are predefined. Thus, this work focuses mainly on the challenging topology design problem and leaves the integration with existing device parameter optimization methods as future work.

**Circuit Evaluation.** Given a generated power converter circuit with topological representation $s$ and control parameter $d$, the Spice-based transient simulation is conducted to obtain the voltage conversion ratio $\gamma_{s,d}$ and power efficiency $\eta_{s,d}$.

**Circuit Generation Objective.** Given the custom design task with a target voltage conversion ratio $\gamma_0$, the goal of our framework is to automatically generate the topological representation $s$ (with chosen component types and edges) and configure the control parameter $d$ of the power converter circuit. Specifically, the objective can be described as:

$$s^*, d^* = \arg\max_{s \in S, d \in D} [U_{\gamma_0}(\gamma_{s,d}, \eta_{s,d})], \tag{1}$$

where $S$ and $D$ denote the set of topological representations and the set of control parameter configurations. $U_{\gamma_0}$ is a utility function that evaluates a circuit design's conversion ratio and efficiency for the custom design task with target conversion ratio $\gamma_0$. Specifically, the utility function is formulated as:

$$U_{\gamma_0}(\gamma_{s,d}, \eta_{s,d}) = \eta_{s,d} \cdot \delta(\gamma_{s,d}, \gamma_0). \tag{2}$$

In our formulation, $\delta$ measures how close the obtained conversion ratio $\gamma_{s,d}$ and the target conversion ratio $\gamma_0$ are. In our experiments, we use $\delta(\gamma, \gamma_0) = 1.1^{-\left(\frac{15(\gamma-\gamma_0)}{|\gamma_0|}\right)^2}$. The utility function equals 0 when the topology is invalid or incomplete.

## Framework

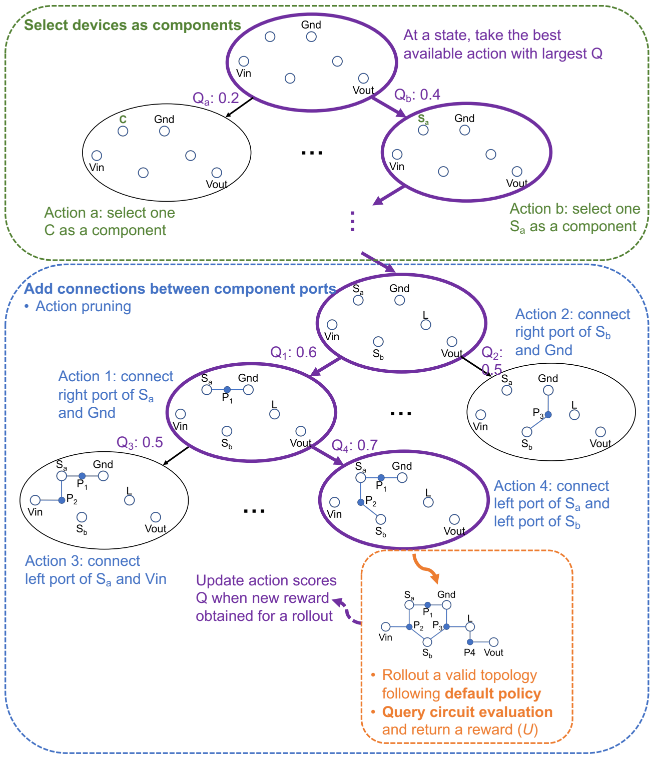
The proposed framework is able to efficiently explore the topological representation space and locate candidate designs with high utility scores. We first formulate the circuit generation task as a sequential decision-making problem. Instead of synthesizing the entire topology all at once, the component device types and connections between ports are added step by step, as illustrated in Figure 3. This multi-step formulation allows physics-aware connection pruning and removes many isomorphic topologies by construction. We further improve the topology generation by incorporating offline knowledge from the pre-collected dataset through a default policy. Finally, to address the issue of costly circuit evaluation, we use a State-Space Averaging technique to evaluate different parameter configurations of the same circuit and validate the output candidates via Spice simulations.

### Sequential Circuit Topology Generation

We formulate topology generation as a sequential decision task. In particular, we model the topology generation as a Markov Decision Process, namely a 4-tuple of $\langle S, A, T, R\rangle$, representing the state set $S$, action set $A$, state transition function $T$, and reward function $R$. As shown in Figure 3, the topology generation always starts with an empty topology of the power converter and has two phases with multiple steps: the device type selection phase and connection selection phase.

In the $t$-th step, the state $s_t \in S$ is a partial or complete topology of the power converter circuit. Inspired by the simple fact that circuit topology is a graph, each state $s_t$ maintains a component set, a port set, and an adjacency matrix specifying the connections between each pair of ports. The action set $A_t$ depends on the current state $s_t$. For a state in the device type selection phase, an action $a_t \in A_t$ decides whether a device type is selected for a component. For the connection selection phase, an action either decides to skip adding more connections or decides which port is connected to the port under consideration. Given our state and action formulation, the state transition is a deterministic function $s_{t+1} = T(s_t, a_t)$, which maps the current partial topology and the action choice to the next topology. The reward function $R_{\gamma_0}$ encodes the custom design objective $U_{\gamma_0}$ in Equation 2 such that $R_{\gamma_0}(s_t, a_t) = \max_d U_{\gamma_0}(\gamma_{s_{t+1},d}, \eta_{s_{t+1},d}) - \max_{d'} U_{\gamma_0}(\gamma_{s_t,d'}, \eta_{s_t,d'})$. The optimization objective is to find an action sequence that maximizes the final topology's power efficiency:

$$a^*_{0:T} = \arg\max_{a_{0:T}} \sum_{t=0}^{T} R_{\gamma_0}(s_t, a_t) \tag{3}$$



Select devices as components
At a state, take the best available action with largest Q
Gnd
Vin
Vout
$Q_a$: 0.2
$Q_b$: 0.4
C
$S_a$
Action a: select one C as a component
Action b: select one $S_a$ as a component
Add connections between component ports
• Action pruning
L
$S_b$
$Q_1$: 0.6
$Q_2$: 0.5
Action 1: connect right port of $S_a$ and Gnd
Action 2: connect right port of $S_b$ and Gnd
$P_1$
$P_2$
$P_3$
P4
$Q_3$: 0.5
$Q_4$: 0.7
Action 3: connect left port of $S_a$ and Vin
Action 4: connect left port of $S_a$ and left port of $S_b$
Update action scores Q when new reward obtained for a rollout
• Rollout a valid topology following **default policy**
• **Query circuit evaluation** and return a reward ($U$)

Figure 3: UCT-based Topology Generation Overview: it starts with a topology having empty components and 3 terminals. It first makes device selections (shown in green) and then connects ports of components and terminals (shown in blue), where the actions are specially designed to reduce isomorphism and invalid circuits. The look-ahead tree construction and action score calculation (shown in purple) follow the UCT algorithm, which utilizes upper confidence bound in exploitation and exploration trade-offs. To improve the sample efficiency and rollout speed, we designed a default policy and adopted a fast evaluation technique (shown in orange).

Note that the above objective requires computing the maximum utility over the different control parameters. The effective way to compute this utility will be presented in Section .

### UCT-based Topology Generation

A UCT-based method is used to optimize the action sequence. UCT is a popular Monte Carlo Tree Search algorithm for large state-space sequential decision-making optimization problems, which suits the large topological representation space in our problem formulation. At a high level, UCT builds a look-ahead tree and greedily selects an action based on the actions' estimated scores at each step. In our proposed framework shown in figure 3, each tree-node of UCT corresponds to a partial or complete topology (i.e., a state) and each tree-edge corresponds to a device type or connection selection (i.e., an action).

To accurately estimate the action scores $Q$, UCT performs multiple *look-ahead rollouts* (also called simulations in RL literature), which sample the remaining action sequence following a *default policy* and evaluate the quality of the samples in terms of reward. UCT is an anytime optimization algorithm with three parameters, the number of look-ahead rollouts, the maximum depth (uniform for each rollout), and an exploration parameter. In general, the larger the number of rollouts and the depth parameter are, the slower UCT is, but the better it is. Compared to other Monte Carlo Tree Search algorithms, UCT utilizes the Upper Confidence Bound in the tree building process to provide improved estimations of action scores. UCT computes a score for each action $a_t$ at a state $s_t$ as the sum of the exploitation and exploration terms, as follows:

$$Q^{\mathrm{UCT}}(s_t, a_t) = Q^{\mathrm{MC}}(s_t, a_t) + \sqrt{\frac{\log(n(s_t))}{n(s_t, a_t)}} \qquad (4)$$

Here, $Q^{\mathrm{MC}}$ is the Monte Carlo average of the sum of rewards obtained from the look-ahead rollouts, i.e., $Q^{\mathrm{MC}}(s_t, a_t) = \frac{1}{M}\sum_{m=1}^{M}\sum_{t'=t}^{T} R(s_{t'}^m, a_{t'}^m)$, where $m$ identifies a look-ahead rollout in the total $M$ rollouts. The exploration term $\sqrt{\log(n(s_t))/n(s_t, a_t)}$ is the Upper Confidence Bound, where $n(s_t)$ is the number of visits for the state node $s_t$ and $n(s_t, a_t)$ the number of visits of the action $a_t$ at state $s_t$. UCT selects the action to rollout greedily with respect to this summed score using the look-ahead tree. Once the input-parameter number of rollouts are produced each to the maximum depth, UCT returns the exploitation term for each action at the root node.

## Combining Knowledge into UCT via Action Pruning

As discussed above, UCT estimates a state-action pair's score $Q^{\mathrm{MC}}(s_t, a_t)$ more accurately when more rollouts are performed and the total number of rollouts is fixed according to the affordable computation cost. Hence, pruning the action space can improve UCT's search efficiency.

**Reduce Invalid Topologies.** Based on the knowledge of electronic circuits, we pose a set of constraints when adding connections to avoid generating invalid power converter circuits. Specifically, we do not include an action $a$ of connecting two ports into the candidate action set $A$ if adding this connection leads to one of the following situations: (1) a shortcut; (2) a direct connection between terminal ports `VIN`, `VOUT`, or `GND`; (3) a prohibited path; (4) a disconnected circuit. Here, the prohibited paths are the paths that violate basic circuit principles. For example, if we connect `VIN` and `GND` only with an inductor, it is equivalent to a power shortcut. A circuit is considered disconnected if there is at least one port not connected to any of the terminal ports via any paths of connections after the generation process. Together with the action pruning rules described below for connection selection, we can identify such disconnected topologies early in the generation process. Specifically, if there is no more allowed connection that can be added to a port and there is no path connecting this port to any of the terminal ports, then this partial topology will eventually become a disconnected circuit even if additional connections are selected between other ports. Thus, we mark the corresponding state as a terminal state with a reward of 0, so no more actions can be taken from this state.

**Reduce Isomorphic Topologies.** Combining the states representing isomorphic topologies can also improve UCT's search efficiency, since the rewards of rollouts from all these states can be collectively used to estimate the score more accurately. Hence, we propose the following methods to reduce the generated isomorphic topologies by construction.

First, we split the device type selection phase into multiple rounds, where each round has an ordered set of available device types. The set in the first round includes all device types. In each step of a round, each available device type is considered for selection. If a device type has been skipped in the current round, it will be removed from the set of available device types for the next round. The device type selection phase ends when the number of selected devices is equal to the number of components. In this way, every state in this phase is unique , while all combinations of device selections can be generated.

Next, for the connection selection phase, we number all the ports where the terminal ports have the smallest indexes. We consider each port for adding connections with other ports one by one. Although a connection in a converter topology is not directed, we only allow a port with a smaller index to be connected to a port with a larger index. Thus, this connection can only be added once. In addition, since a device is nondirectional, we always have its left port be connected before its right port when both ports have no connection. Following the above rules, many actions that lead to states representing isomorphic topologies are pruned.

## Combining Knowledge into UCT via Default Policy

Due to the large space of topologies, the computation cost for finding a good topology can be high even after action pruning. Hence, we further improve the effectiveness of UCT by replacing the random default policy with a data-driven one. In particular, we randomly generate some topologies with fewer components and evaluate their efficiencies and conversion ratios. For example, we collect a small data set with 3-component topologies for the design task of 5-component topologies. Among them, we find the *good topologies* with higher rewards and collect their device selections and all paths of connections between the terminal ports. Exploiting the information obtained from good topologies with smaller sizes, we develop the following two default policies for the two topology generation phases.

**Node Selection.** We collect the number of times a device selection combination $C$ (e.g., $\{\mathtt{S}_a, \mathtt{S}_a, \mathtt{L}\}$) occurring in the good topologies, denoted by $n_d(C)$. This information is encoded into the default policy, such that its probability of a device selection combination $C'$ is approximately proportional to the collected device selection distribution. Specifically, we calculate the weight of a device selection combination $C'$ as $w(C') = \sum_{C \subseteq C'} n_d(C)$. The total weight is defined as $W_d = \sum_{C''} w(C'')$. Then the probability of the device selection combination $C'$ generated by the default policy is $\mathbb{P}_d(C') = \frac{w(C')/W_d + \epsilon_d}{\sum_{C''}(w(C'')/W_d + \epsilon_d)}$, where $\epsilon_d$ is a small constant that decides by how much the default policy follows the collected distribution. In our experiments, $\epsilon_d$ is set to 0.01.

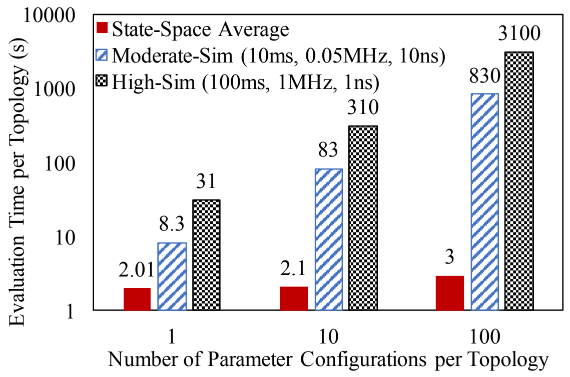


Figure 4: Evaluation time per topology for different numbers of parameter configurations per topology.

**Edge Selection.** We also collect the number of occurrences of a *good path* of connections between any two of the terminal ports (`VIN`, `VOUT`, and `GND`) in the good topology data set, denoted as $n_d(p)$. When considering a port $e$ for adding a connection onto the current partial topology $s_t$, we first construct a set of all possible good paths $P(e, s_t)$. A path in $P(e, s_t)$ must contains port $e$ and adding this path to $s_t$ cannot lead to invalid topologies (e.g., short-cuts). Next, for any connection $(e, e')$ allowed to be added to $s_t$, we calculate the weight of taking this action as $w(e, e') = \sum_{p \in P(e,s_t) \& (e,e') \subseteq p} n_d(p)$. Similar to node selection, we can calculate the total weight of all allowed actions as $W_p = \sum_{(e,e'')} w(e, e'')$. Then the probability of the connection selection action $(e, e')$ generated by the default policy is $\mathbb{P}_p(e, e') = \frac{w(e,e')/W_p + \epsilon_p}{\sum_{(e,e'')}(w(e,e'')/W_p + \epsilon_p)}$, where $\epsilon_p$ is a small constant and is set to 0.4 in the evaluation. Note that although the weights are calculated using the good paths in the data set with fewer components, longer paths with more components can be generated with the help of $\epsilon_p$.

With the default policies, we can bias UCT towards searching the topology space that may contain high-reward ones.

### Efficient Evaluation via State-Space Averaging

The reward function used by UCT is a computational bottleneck as it requires evaluating different control parameters for different topologies sampled by all the rollouts. Simulation software, such as NGSpice (Nenzi and Vogt 2011), often takes a long time to evaluate one topology with fixed parameters. To speed up the circuit evaluation, we instead adopt the *State-Space Averaging* approach (Polivka, Chetty, and Middlebrook 1980). This approach shares the major computation among the circuits with the same topology but different parameters, which makes it a faster surrogate model in estimating a large number of parameters.

Specifically, State-Space Averaging characterizes the transfer properties of switching stages of a power converter circuit. By dividing the circuit into phases-I and phase-II sub-circuits, deriving state-space equations, and averaging state changes with the corresponding duty-cycles (i.e., $d$ and $1 - d$), it derives the estimated output voltage and power efficiency.

We compared the computation times of State-Space Averaging versus NGSpice. Figure 4 shows that the evaluation time per topology under State-Space Averaging is significantly lower compared to moderate-fidelity and high-fidelity simulations. The advantage of State-Space Averaging is even higher when more parameter configurations of one topology (e.g., the same topology with different duty cycles) need to be evaluated. This is because deriving state-space equations is the most expensive step of State-Space Averaging, but these equations remain the same as long as the topology remains the same. Hence, increasing the number of parameter configurations per topology does not increase the evaluation time much under State-Space Averaging, while simulations must be performed for each parameter configuration of the same topology.

Despite its speed advantage, State-Space Averaging has a disadvantage — it is only accurate if the circuit is linear during operations. Nonlinear effects, such as discontinuity in conduction or enforced switch dead-zone, will introduce errors. As such, the state-space method usually overestimates the efficiencies of topologies. To address this issue, we take a hybrid circuit evaluation approach. During topology search, a topology is evaluated by State-Space Averaging. With the optimistic nature of State-Space Averaging, it covers all qualified topologies. Once the top topology candidates are generated, NGSpice simulation will be invoked to offer ground-truth evaluation for the final topology and control scheme selection.

## Evaluation

We evaluate our framework across a spectrum of custom design tasks with different conversion ratios and compare its performance with baseline algorithms. We also conduct an ablative study on the effectiveness of the proposed data-driven default policy. Finally, we discuss the nonconventional power converter topologies discovered by our framework.

### Experiment Setup

We conduct the evaluation on the power converter design task with five components, each with two ports. Together with the three external ports (`Vin`, `Vout`, and `Gnd`), the design space contains topologies with a total of thirteen ports. The component device types have fixed device parameters and include capacitors `C` ($10\mu F$), inductors `L` ($100\mu H$), phase-I switches $\texttt{S}_a$, and phase-II switches $\texttt{S}_b$ For external ports, we consider an input resistor of $0.1\Omega$ for `Vin` and an output resistor of $100\Omega$ and an output capacitor of $10\mu F$ for `Vout`. The candidate control parameter (i.e., duty cycle) ranges from 0.1 to 0.9, with a step size of 0.1. We configure the frequency as $1MHz$ for both analytic evaluation and simulation. We set the input voltage to $100V$, and the target voltage outputs are chosen from the range of -300$V$ to 300$V$. Additionally, a topology with a conversion ratio smaller than -5 or larger than 5 will be considered invalid. In NGSpice simulation, the transient simulation time is set as $60s$.

## Implementation Details and Competitive Algorithms

In this section, we describe the implementation details of our UCT-based framework and the baseline algorithms.

**Hash Table.** Both the UCT-based and baseline algorithms may query the circuit evaluation about a topology more than once. To avoid unnecessary circuit evaluation costs, we implemented a hash table to cache the efficiencies and conversion ratios of the topologies that have been queried during each experiment. Therefore, in our experiments for both the proposed algorithm and baselines, all the queries to the circuit evaluation are about unique topologies.

**Implementation Details of Proposed Algorithm.** There are two important implementation details of our UCT-based topology generation algorithm. First, we use the number of rollouts to decide the number of explorations each run of the UCT-based topology generation. In each rollout, we expand a UCT tree-node and execute a rollout to get a complete topology. However, the number of rollouts can be larger than the number of queries, since we only query the circuit simulation when a rollout leads to a valid topology that is not in the hash table. Second, different from the common UCT design that completely reconstructs a new tree for each step, we inherit the sub-tree corresponding to the selected action from the tree of the previous step to make full use of the collected information. This operation makes our UCT-DP algorithm converge faster.

**Random Search (RS).** Random Search is a strategy that starts with an empty topology, randomly selects device types, and then randomly connects ports until reaching a complete topology. Note that, if a port already has connections to other ports, then it is not required to connect to yet another port. Thus, in this case, RS may skip adding more connections to this port with a probability of 0.8. RS uses the same reward function as UCT, along with the same prohibitive paths that prevent generating invalid topologies. After searching a prefixed number of complete circuit topologies, RS outputs the one that has the highest reward.

**Genetic Search (GS).** We implemented another popular heuristic-based search algorithm, namely Genetic Search, for the power converter design task (McConaghy et al. 2011). GS starts with 15 random topologies. In each round, GS selects 3 topologies with the highest rewards from the current generation and uses them as parents to generate offspring by mutation. We implemented the following mutation types: (1) A crossover randomly selects a component that exists in two parents and exchanges the connections of this component (Rojec, Burmen, and Fajfar 2019); (2) A component change randomly changes the device type of a random component in a parent; (3) A connection insertion randomly selects and connects two unconnected ports in a parent; (4) A connection removal removes a random connection of a parent; (5) A connection switch selects a random connection of a parent and changes one of the connection's ports to another random port; In each round, the above mutation types are chosen with probabilities of $\{20\%, 10\%, 30\%, 20\%, 20\%\}$, respectively.

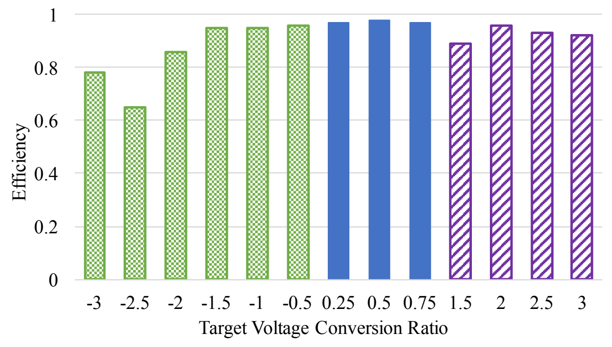


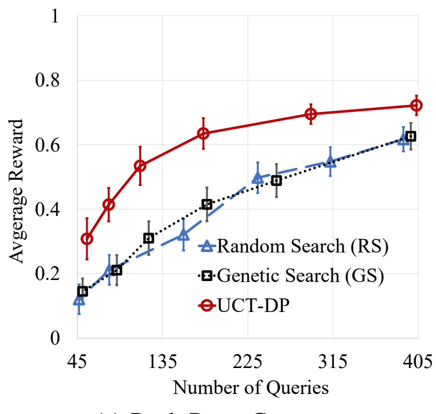
Figure 5: Average power efficiencies of power converters generated by the proposed framework for custom design tasks with increasing voltage conversion ratios. From left to right in different colors and shades, the target converter types are buck-boost, buck, and boost.

## Experiment Results

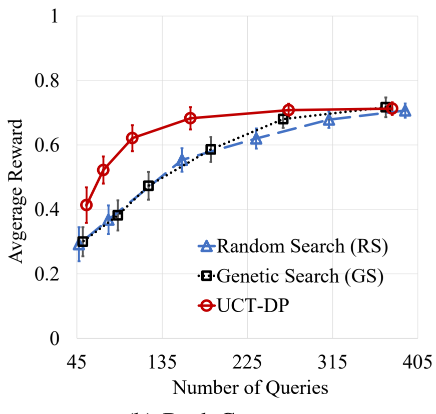
**Custom Design Tasks.** We first evaluated our UCT-based framework on different custom design tasks for power converters. For each target voltage conversion ratio, we run our framework five times and compute the average efficiency of the generated topologies. Figure 5 shows the average power efficiencies under our framework for different voltage conversion ratios. Results show that our framework can successfully find high-performing topologies for most settings. The average efficiencies for smaller conversion ratios tend to be lower, mainly because these buck-boost converters are more sensitive to their duty cycles.

**Comparison with Baseline Algorithms.** We compared our UCT-based approach with RS and GS on buck-boost, buck, and boost converters. Since all these algorithms are anytime algorithms (i.e., the performance monotonically increases as more computation is used), we compared their performances conditioned on their computation costs, measured by the number of queries to circuit evaluations. Figure 6 reports the average reward of topologies generated by each algorithm. For all types of converters, our UCT-based approach, named **UCT-DP**, outperforms both RS and GS, while GS is comparable with or slightly outperforms RS. When the number of queries is around 110, UCT-DP achieves 83%, 35%, and 37% higher rewards compared with GS for buck-boost, buck, and boost converters, respectively. In terms of computation efficiency, results show that, compared to GS, UCT-DP needs up to 63%, 52%, and 67% fewer queries to obtain the same average rewards for buck-boost, buck, and boost converters, respectively. We also observe that all the algorithms perform slightly worse for buck-boost converters, which may be caused by the step size of duty cycles as discussed above. Overall, the results demonstrate the efficacy of our UCT-based approach to discover high-quality topologies using only a small number of queries.

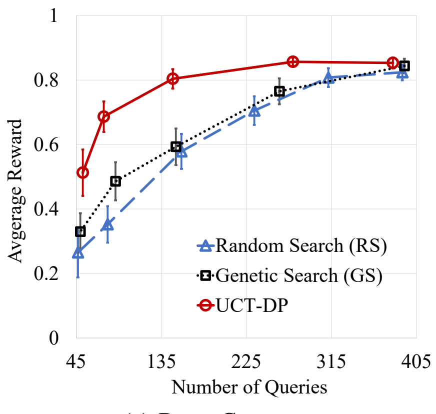
**Ablative studies on default policies.** In Section , we described how we incorporate offline knowledge using default policies. In this experiment, we examine the effectiveness of the default policies by performing an ablation study. Figure 7 presents the average rewards with and without the two



(a) Buck-Boost Converter



(b) Buck Converter



(c) Boost Converter

Figure 6: Average rewards under UCT-DP vs. GS vs. RS with increasing numbers of unique queries to circuit evaluations. The x-axis shows increasing numbers of queries to circuit evaluations for unique topologies. The y-axis is the obtained reward calculated using the power efficiency and voltage conversion ratio of the best candidate circuit generated by an algorithm, averaging from 200 runs.

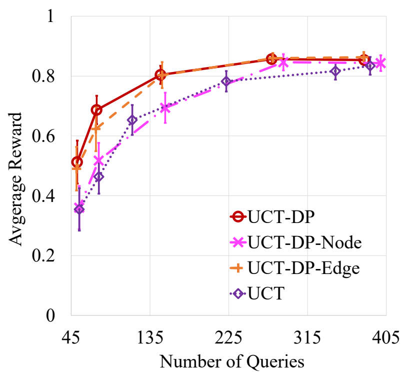
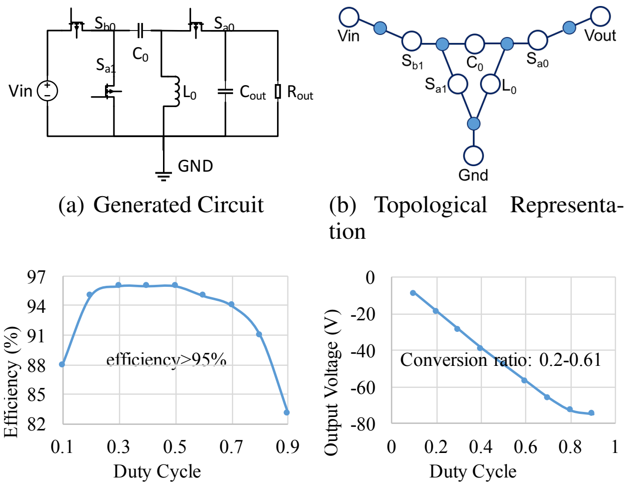


Figure 7: Ablative studies of UCT-based approaches without default policy (UCT), with node selection default policy (UCT-DP-Node), with edge selection default policy (UCT-DP-Edge), and with both node and edge selection default policies (UCT-DP) for boost converters.



(a) Generated Circuit (b) Topological Representation
(c) Circuit Power Efficiency (d) Circuit Output Voltage

Figure 8: An example of unconventional power converter circuit discovered by our framework.

default policies, namely node selection and edge selection, for the boost converter design task. The results for other power converter types are similar. Not surprisingly, UCT-DP with both default policies performs the best. We also observe that UCT with edge selection only (UCT-DP-Edge) performs significantly better than with node selection only (UCT-DP-Node). This is partly because choosing good connections among the exponentially many connections is more challenging. Moreover, since device type selections are performed in the first phase before connection selections, the node selection default policy is only used by the rollouts during the first phase. In contrast, the edge selection default policy is used by all the rollouts, so it has a higher impact on the performance.

## Discussion on New Topologies

Our automatic power converter design framework generates topologies that meet design targets. Besides classical power converter topologies, it is also able to find interesting unconventional topologies that both satisfy specifications and have high power efficiencies, such as the one in Figure 8. These automatically generated topologies have the potential to shed light on fundamental circuit innovations. With close collaboration with human experts, our framework can help to discover innovative circuits that have not been studied.

## Conclusion

In this work, we proposed a UCT-based framework, which is able to explore the design space of power converter topologies automatically. We also incorporated physics-informed constraints and data-driven default policies to reduce the design space and improve the efficiency of our framework. Additionally, we adopted a hybrid circuit evaluation with both the fast State-Space Averaging method and the accurate high-fidelity simulation. Finally, evaluations showed that our framework can generate near-optimal circuit topology for buck-boost, buck, and boost converters. Compared to the alternative approaches, our framework can discover better circuit topologies with reduced computational costs.

## References

Chang, E.; Han, J.; Bae, W.; Wang, Z.; Narevsky, N.; NikoliC, B.; and Alon, E. 2018. BAG2: A process-portable framework for generator-based AMS circuit design. In *2018 IEEE Custom Integrated Circuits Conference (CICC)*, 1–8.

Chen, H.; Zhu, K.; Liu, M.; Tang, X.; Sun, N.; and Pan, D. Z. 2020. Toward Silicon-Proven Detailed Routing for Analog and Mixed-Signal Circuits. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 1–8.

Hakhamaneshi, K.; Werblun, N.; Abbeel, P.; and Stojanović, V. 2019a. Bagnet: Berkeley analog generator with layout optimizer boosted with deep neural networks. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 1–8. IEEE.

Hakhamaneshi, K.; Werblun, N.; Abbeel, P.; and Stojanović, V. 2019b. Late Breaking Results: Analog Circuit Generator based on Deep Neural Network enhanced Combinatorial optimization. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 1–2.

Han, S.; Jeong, S.; Kim, C.; Park, H.-J.; and Kim, B. 2020. GUI-Enhanced Layout Generation of FFE SST TXs for Fast High-Speed Serial Link Design. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 1–6.

Kim, J.; Lee, J.; Vandenberghe, L.; and Yang, C.-K. K. 2004. Techniques for improving the accuracy of geometric-programming based analog circuit design optimization. In *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, 863–870. IEEE.

Lee, W.; and Oliehoek, F. A. 2020. Analog Circuit Design with Dyna-Style Reinforcement Learning. In *NeurIPS 2020 Workshop: Machine Learning for Engineering Modeling, Simulation, and Design*.

Liu, M.; Li, W.; Zhu, K.; Xu, B.; Lin, Y.; Shen, L.; Tang, X.; Sun, N.; and Pan, D. Z. 2020. S3DET: Detecting System Symmetry Constraints for Analog Circuits with Graph Similarity. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*.

Lu, Y.-C.; Pentapati, S. S. K.; Zhu, L.; Samadi, K.; and Lim, S. K. 2020. TP-GNN: a graph neural network framework for tier partitioning in monolithic 3D ICs. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 1–6. IEEE.

Lyu, W.; Yang, F.; Yan, C.; Zhou, D.; and Zeng, X. 2018. Batch Bayesian Optimization via Multi-objective Acquisition Ensemble for Automated Analog Circuit Design. In *International Conference on Machine Learning*, 3306–3314. PMLR.

McConaghy, T.; Palmers, P.; Steyaert, M.; and Gielen, G. G. E. 2011. Trustworthy Genetic Programming-Based Synthesis of Analog Circuit Topologies Using Hierarchical Domain-Specific Building Blocks. *IEEE Transactions on Evolutionary Computation*, 15(4): 557–570.

Nenzi, P.; and Vogt, H. 2011. Ngspice Users Manual Version 23.

Nikolic, B.; Alon, E.; and Asanovic, K. 2018. Generating the Next Wave of Custom Silicon. In *ESSCIRC 2018 - IEEE 44th European Solid State Circuits Conference (ESSCIRC)*, 6–11. ISSN: 1930-8833.

Polivka, W. M.; Chetty, P. R.; and Middlebrook, R. D. 1980. State-Space Average modelling of converters with parasitics and storage-time modulation. In *1980 IEEE Power Electronics Specialists Conference*, 119–143.

Rojec, z.; Burmen, A.; and Fajfar, I. 2019. Analog circuit topology synthesis by means of evolutionary computation. *Engineering Applications of Artificial Intelligence*, 80: 48–65.

Settaluri, K.; Haj-Ali, A.; Huang, Q.; Hakhamaneshi, K.; and Nikolic, B. 2020. Autockt: Deep reinforcement learning of analog circuit designs. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 490–495. IEEE.

Singh, A. K.; Ragab, K.; Lok, M.; Caramanis, C.; and Orshansky, M. 2012. Predictable equation-based analog optimization based on explicit capture of modeling error statistics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(10): 1485–1498.

Wang, H.; Wang, K.; Yang, J.; Shen, L.; Sun, N.; Lee, H.-S.; and Han, S. 2020. GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 1–6.

Wang, Y.; Orshansky, M.; and Caramanis, C. 2014. Enabling efficient analog synthesis by coupling sparse regression and polynomial optimization. In *Proceedings of the 51st Annual Design Automation Conference*, 1–6.

Wu, T.; Wang, Z.; Ozpineci, B.; Chinthavali, M.; and Campbell, S. 2019. Automated Heatsink Optimization for Air-Cooled Power Semiconductor Modules. *IEEE Transactions on Power Electronics*, 34(6): 5027–5031.

Yang, K.-E.; Tsai, C.-Y.; Shen, H.-H.; Chiang, C.-F.; Tsai, F.-M.; Wang, C.-A.; Ting, Y.; Yeh, C.-S.; and Lai, C.-T. 2020. Fast Design Space Adaptation with Deep Reinforcement Learning for Analog Circuit Sizing. *arXiv preprint arXiv:2009.13772*.

Zhang, G.; He, H.; and Katabi, D. 2019. Circuit-GNN: Graph Neural Networks for Distributed Circuit Design. In *International Conference on Machine Learning*, 7364–7373. PMLR.

Zhang, Y.; Wang, Z.; Wang, H.; and Blaabjerg, F. 2020. Artificial Intelligence-Aided Thermal Model Considering Cross-Coupling Effects. *IEEE Transactions on Power Electronics*, 35(10): 9998–10002.

Zhao, S.; Blaabjerg, F.; and Wang, H. 2021. An Overview of Artificial Intelligence Applications for Power Electronics. *IEEE Transactions on Power Electronics*, 36(4): 4633–4658.

Zhao, Z.; and Zhang, L. 2020. An Automated Topology Synthesis Framework for Analog Integrated Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 39(12): 4325–4337.

Zhou, Z.; Belakaria, S.; Deshwal, A.; Hong, W.; Doppa, J. R.; Pande, P. P.; and Heo, D. 2020. Design of Multi-Output Switched-Capacitor Voltage Regulator via Machine Learning. In *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 502–507.