A Novel FPGA-Based Circuit Simulator for Accelerating Reinforcement Learning-Based Design of Power Converters

1st Zhenyu Xu, 2nd Miaoxiang Yu, 3rd Jillian Cai, 4th Qing Yang,
5th Yeonho Jeong, and 6th Tao Wei
Electrical, Computer and Biomedical Engineering
University of Rhode Island
Kingston, US
{zhenyu_xu, miaoxiang_yu, jillian_cai, qyang, yjeong, tao_wei}@uri.edu

***Abstract*—High-efficiency energy conversion systems have become increasingly important due to their wide use in all electronic systems such as data centers, smart mobile devices, E-vehicles, medical instruments, and so forth. Complex and interdependent parameters make optimal designs of power converters challenging to get. Recent research has shown that machine learning (ML) algorithms, such as reinforcement learning (RL), show great promise in design of such converter circuits. A trained RL agent can search for optimal design parameters for power conversion circuit topologies under targeted application requirements. Training an RL agent requires numerous circuit simulations. It requires significantly more training iterations when the tolerance of circuit components due to manufacturing inconsistency, aging, and temperature variation is considered. As a result, they may take days to complete, primarily because of the slow time-domain circuit simulation.**

This paper proposes a new FPGA architecture that accelerates the circuit simulation and hence substantially speeds up the RL-based design method for power converters. Our new architecture supports all power electronic circuit converters and their variations. It substantially improves the training speed of RL-based design methods. High-level synthesis (HLS) was used to build the accelerator on Amazon Web Service (AWS) F1 instance. An AWS virtual PC hosts the training algorithm. The host interacts with the FPGA accelerator by updating the circuit parameters, initiating simulation, and collecting the simulation results during training iterations. A script was created on the host side to facilitate this design method to convert a netlist containing circuit topology and parameters into core matrices in the FPGA accelerator. Experimental results showed 60× overall speedup of our RL-based design method in comparison with using a popular commercial simulator, PowerSim.

***Index Terms*—Power electronics, Simulation, Reinforcement learning, FPGA accelerator**

I. Introduction

High-efficiency power conversion systems are desired in light of the current energy crisis and the global trend to shift to clean energy. Power converters are widely used in industry due to the considerations of power conversion efficiency [1], power density [2], cost [3], reliability [4], etc. However, the design process of these circuits is intricate. Taking resonant power converter as an example, the design parameters of power converters, such as component values, choice of transistors, and the switching frequency, are often interdependent [5], [6]. The design becomes significantly more challenging when the tolerance of circuit components due to manufacturing inconsistency, aging, and temperature variation is under consideration. Over the years, a series of design methods have been proposed. For example, previous works have focused on finding accurate voltage gain equations based on the Fourier series, discussed in [7]–[9]. Time domain-based studies are also considered in [8], [9]. Unfortunately, the high complexity of these design methods makes them non-practical. Nowadays, the fundamental harmonic approximation (FHA) method is widely used in industry due to its simplicity [1], [10]. Still, it is not accurate. Consequently, a large margin is required to ensure that designs following the FHA method meet the design specifications and requirements. This prevents the FHA method from reaching optimal designs with the highest possible power conversion efficiency.

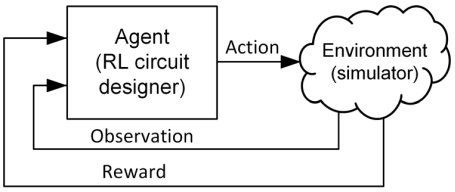


Fig. 1. Generic reinforcement learning (RL) block diagram

Efforts have been made to explore new design methods [4], [11]–[15]. Recently, machine learning (ML)-based design has shown that it can find optimal design parameters with the highest efficiency in a given topology. Prior works [14], [15] show that reinforcement learning (RL) is capable of substantially reducing analog circuit optimization time to meet all design goals (targeted design specifications) in comparison with human experts. In this RL structure, shown in Fig. 1, an action tunes the design parameters for the next iteration; observations are the design performance observed

DOI 10.1109/ASAP57973.2023.00013

from circuit simulation; the reward is calculated based on the observations, as well as the provided design goals. To obtain the observations (design performance), time-domain circuit simulation is required. Time-domain circuit simulation is time-consuming on a general-purpose computer. In power electronics, the simulation timestep can be as small as $10ns$ over a duration of several tens of milliseconds or more, which requires simulation iterations over 1 million timesteps. This may easily bring the simulation time of one RL iteration to a few seconds or more on a general-purpose computer. As a result, offline training easily consumes days of computing time. Wang et al. [11] reported that the simulation takes over 99% percent of the time in one RL training. When considering the tolerance of circuit components, the RL training typically takes much longer to converge. Long training time presents a major bottleneck of the RL-based design method. In practice, different applications and different vendors have drastically different design requirements, requiring us to train many RL circuit designers even for the same power converter topology.

Efforts have been made to accelerate the circuit simulation in the past. Kapre introduced an FPGA accelerator and a GPU accelerator for SPICE simulation [16]. Due to the non-linear nature of many active circuit components, Newton-Raphson iteration is required to search for the solution at each simulation timestep. While in power electronics, the semiconductor switches and diodes are often treated as linear devices, removing the necessity of Newton-Raphson iteration. In addition, the strong data dependency between consecutive simulation timesteps makes GPU acceleration inefficient, as reported in [17]. On the other hand, in the power electronics area, FPGA-based real-time hardware-in-the-loop (HIL) power electronics simulators have been created based on linear circuit models [18], including a few commercial products [19]. In these simulators, the simulation time consumed per simulation timestep is smaller than the simulation timestep, allowing them to predict the behavior of the circuit. Unfortunately, HIL simulators are unsuitable for the RL-based design method since the topology or component values cannot be modified on the fly. A unique bitstream is dedicated to one specific design, and changing design parameters requires regenerating bitstreams.

This work describes an FPGA-based SPICE circuit simulator to accelerate the RL-based power converter design. The main contributions of this work include an FPGA-based power converter circuit simulation accelerator that can change the design parameters of a circuit model on the fly without the need for regenerating bitstreams, its integration with RL algorithms in the host machine, and interface design for fast communication between the accelerator and the host machine. The proposed FPGA simulation accelerator is packaged in an OpenAI Gym environment. The Gym environment provides standard programming APIs for RL algorithms. The proposed simulation accelerator is deployed on Amazon Web Services (AWS) F1 instance as a service, so that power electronic engineers worldwide can enjoy high-quality and fast training and design by simply providing the reward function for the RL algorithm that best fits their application needs. To demonstrate the effectiveness of the proposed design method, we employed a popular RL algorithm to perform offline training (training an RL circuit designer) proposed by Settaluri et al. [20] on two examples. The first example is a half-bridge LLC dc-dc converter for data center power supply applications. The second is a full-bridge LLC dc-dc converter for electric vehicle onboard battery charger (OBC). The same RL algorithm was applied to the popular PowerSim simulator for comparison purposes. Experimental results show that the proposed FPGA-enabled method delivers a 60× speedup in offline training.

II. Background

This section briefly describes how power electronic circuit components are modeled in modified nodal analysis (MNA) and how time-domain simulation is conducted using MNA [21]. In power electronics, passive components like resistors (R), inductors (L), and capacitors (C) are considered time-invariant and linear; active components like semiconductor switches (SW) and diodes (D) are considered linear but time-variant (discontinuous). In the time-domain simulation, MNA only supports R (or conductance G), ideal voltage sources (v), ideal current sources (j), and voltage (current) controlled voltage (current) sources in the network. Therefore, other components (L, C, SW, and D) need to be modeled with MNA-supported components.

A. Circuit models of passive components

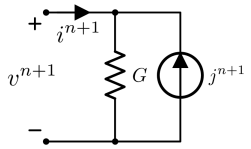


Fig. 2. Circuit models for L, C, SW and D

L and C can be modeled as conductance G in parallel with a current source, as shown in Fig. 2. $n+1$ stands for the next timestep of timestep n, respectively. i^{n+1} is the current that flows into the element at the next timestep $(n+1)$ and j^{n+1} is the current source of the next timestep $(n+1)$. The values of G and j^{n+1} are determined via a numerical method from differential equations. Taking the capacitor as an example, the numeric solution using the backward Euler algorithm is shown in Eq. (1), where T is the timestep used in the simulation. Hence, $G_c = C/T$, and $j^{n+1} = G_c v^n$. Similarly, the model of inductors can also be derived as $G_l = T/L$ and $j_l^{n+1} = -i_L^n$. Notice that G_c and G_l are constants once timestep T is decided, while j_c and j_l depend on the circuit state of the previous timestep.

$$
\begin{aligned}
v_c^{n+1} &= v_c^n + \frac{1}{C}\int_{t^n}^{t^{n+1}} i_c(t)dt \\
v_c^{n+1} &\approx v_c^n + \frac{T}{C} i_c^{n+1} \\
i_c^{n+1} &\approx \frac{C}{T} v_c^{n+1} - \frac{C}{T} v_c^n = G_c \cdot v_c^{n+1} - j_c^{n+1}
\end{aligned} \tag{1}
$$

A transformer is modeled as two coupled inductors, i.e., the current source value at the next timestep $(n+1)$, j^{n+1}, is contributed by currents of both inductors at the current timestep n.

B. Circuit models of active components

In power electronics, switches (SW) and diodes (D) have two states, 'on' (conducting) and 'off' (non-conducting). There are multiple ways to model the switches and diodes. A traditional method models the switch (diode) as a small resistor when it is 'on' and a huge resistor when it is 'off'. However, this method requires an independent model evaluation for all timesteps as the resistance in the netlist could be changed during iteration, which slows down the simulation.

To speed up the model evaluation process, the switch is modeled as a small capacitor (C_{off}) when it is 'off' and modeled as a small inductor (L_{on}) when it is 'on'. According to the circuit model of L and C in Eq. (2), the conductance (G) for the models of L_{on} and C_{off} is identical with a proper timestep T so that the 'on'/'off' state change can be achieved by simply changing the value of the j^{n+1} [22]. Eq. (2) also implies that time step T is related to the C_{off} and L_{on}. Since all real-world devices have their inevitable parasitic capacitance and inductance, the error caused by modeling switches and diodes with capacitors and inductors can be totally avoided by using real parasitic capacitance as C_{off} and using real parasitic inductance as L_{on}. An external control signal determines the state ('on' or 'off') of a switch. A diode is modeled the same way as a switch, except that the state ('on' or 'off') of a diode is controlled by the circuit status.

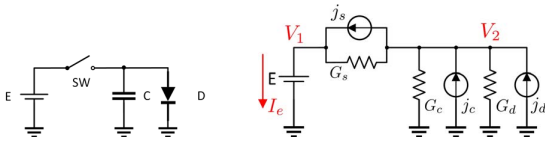
$$
\begin{aligned}
G_l &= \frac{T}{L_{on}} = \frac{C_{off}}{T} = G_c \\
T &= \sqrt{C_{off} L_{on}}
\end{aligned} \tag{2}
$$

C. Time-domain iteration via modified nodal analysis (MNA)

MNA is a classical approach to building an iterative algorithm for circuit simulation in SPICE. A simple circuit example, shown in Fig. 3, describes this method. We can express the system through nodal analysis as Eq. (3).

$$
\underbrace{\left[\begin{array}{ccc:c} G_s & -G_s & & 1 \\ -G_s & G_c+G_s+G_d & & 0 \\ \hdashline 1 & 0 & & 0 \end{array}\right]}_{(\mathbf{A})}
\underbrace{\left[\begin{array}{c} V_1 \\ V_2 \\ \hdashline I_e \end{array}\right]}_{(\mathbf{x}^{n+1})}
=
\underbrace{\left[\begin{array}{c} j_s \\ j_c - j_s + j_d \\ \hdashline E \end{array}\right]}_{(\mathbf{z}^n)} \tag{3}
$$

The 3×3 matrix in Eq. 3 that contains conductance (G) of all components is a constant matrix, referred to as $\mathbf{A}$. All



(a) Circuit (b) Circuit model

Fig. 3. A toy circuit to exemplify MNA method

nodal voltages (V_1, V_2) and the currents flowing into all voltage sources (I_e) at the next timestep $(n+1)$ are placed in a 3×1 vector, named as $\mathbf{x}^{n+1}$. On the right side of the equation, vector $\mathbf{z}^n$ is formed by current sources coming from the discrete-time component model (j_s, j_c, j_d), all voltage sources (E) and real-world current sources (I). Therefore, all nodal voltages and a part of component currents at timestep $n+1$ can be calculated by solving the linear equation $\mathbf{A} \cdot \mathbf{x}^{n+1} = \mathbf{z}^n$ via matrix-vector multiplication (MVM), $\mathbf{x}^{n+1} = \mathbf{A}^{-1} \cdot \mathbf{z}^n$. Then, $\mathbf{z}^{n+1}$ is calculated from $\mathbf{x}^{n+1}$, and one iteration is completed.

III. Accelerator Design Principles

Benefiting from linear models in power electronics simulation, where L_{on},C_{off} model are used for switches and diodes, the Newton-Raphson iteration used in generic SPICE simulator is eliminated, and model evaluation is only required at the beginning of the simulation. This suggests that the matrix $\mathbf{A}$ can be reused over the entire simulation, and it can be stored in the on-chip memory (OCM) at the beginning of the time-domain simulation iteration (in our implementation, the inverse of $\mathbf{A}$ is stored in the OCM). However, there are a few existing challenges that hinder fast time-domain simulation iteration. This section discusses how the MNA algorithm is tailored to tackle the challenges of FPGA circuit simulation acceleration.

A. Auxiliary components

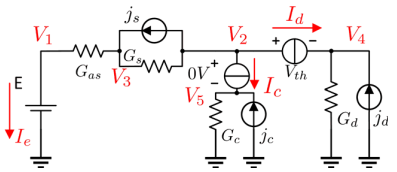


Fig. 4. Discrete-time model for the example circuit with auxiliary components

One significant challenge is that vector $\mathbf{x}$ in MNA only contains all nodal voltages and the currents flowing through the voltage sources. Depending on the circuit schematic, the current flows into the inductors, switches, or diodes cannot be derived from $\mathbf{x}$ directly. As a result, critically, calculating $\mathbf{z}^{n+1}$ could be impossible without knowing the values of both currents at the previous timestep ($\mathbf{x}^n$ and $\mathbf{x}^{n-1}$). Thus, overhead including extra memory, latency, and computational resources are required. To tackle the first challenge, we introduce an auxiliary voltage source (V_a), or an auxiliary conductance (G_a), to be in series with every component except for R and real-world voltage/current sources. The example circuit

after adding the auxiliary components is shown in Fig. 4. The auxiliary voltage source mostly equals 0. For diodes, V_a equals the threshold voltage of the diode, which takes the voltage drop across the diode into consideration. If V_a is added, the current flowing through the device equals the current that flows into V_a; if G_a is added, the current that flows through the device equals G_a times the voltage across G_a. Now, since vector $\mathbf{x}^n$ contains all nodal voltages and currents flowing into voltage sources, the value of all current sources from the discrete-time component model j^{n+1} can be directly calculated using $\mathbf{x}^n$.

B. Partitioning vector z

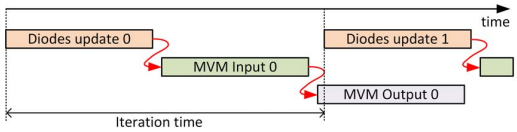
Since the circuit is time variant (the status of switches and diodes are changing), matrix $\mathbf{A}$ is also time variant. To avoid generating $\mathbf{A}$ for all possible combinations of switches and diodes status, the $\mathbf{z}^n$ in Eq. 3 is firstly partitioned into 4 categories, including vector $\mathbf{z}_0^n$ which contains real-world voltage or current sources, vector $\mathbf{z}_{re}^n$ which contains current sources of all reactive components (j_l^{n+1} and j_c^{n+1}), vector $\mathbf{z}_s^n$ which contains current sources (j_s^{n+1}) of all switches, and vector $\mathbf{z}_d^n$ which contains current sources (j_d^{n+1}) of all diodes. This partitioning method is valid as the linear circuit satisfies the superposition property, stating that the overall circuit response with multiple sources equals the sum of the response of each source. Notably, with the auxiliary voltage source or conductance added, $\mathbf{z}_{re}^n$ only depends upon the value of $\mathbf{x}^n$. Therefore, a constant conversion matrix $\mathbf{H}_{re}$ can be pre-determined to convert $\mathbf{x}^n$ into $\mathbf{z}_{re}^n$. However, each switch is controlled differently by an external signal. Assuming there are N_s switches in the circuit, the kth switch has a dedicated conversion matrix ($\mathbf{H}_{s_k}$), and $\mathbf{H}_{s_k}$ contains two options, corresponding to 'on' and 'off' states. Similarly, assuming there are N_d diodes, the kth diode has a dedicated conversion matrix ($\mathbf{H}_{d_k}$), which also contains two options, determined by $\mathbf{x}^n$. Thus, a more general simulation iteration equation is written in Eq. 4.

$$\begin{aligned}\mathbf{x}^{n+1} &= \mathbf{A}^{-1}\mathbf{z}_0^n + \mathbf{A}^{-1}\mathbf{z}_{re}^n + \mathbf{A}^{-1}\mathbf{z}_s^n + \mathbf{A}^{-1}\mathbf{z}_d^n \\ &= \mathbf{A}^{-1}\mathbf{z}_0^n + \left(\mathbf{A}^{-1}\mathbf{H}_{re}\right)\mathbf{x}^n \\ &+ \left(\sum_{k=1}^{N_s}\mathbf{A}^{-1}\mathbf{H}_{s_k}\right)\mathbf{x}^n + \left(\sum_{k=1}^{N_d}\mathbf{A}^{-1}\mathbf{H}_{d_k}\right)\mathbf{x}^n\end{aligned} \tag{4}$$

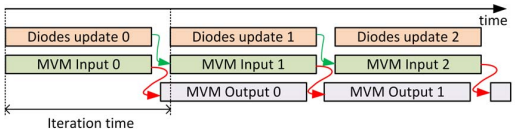
Though the partitioning method seems to create a bunch of matrices ($2+2\times(N_s+N_d)$), it doesn't cause much overhead to the FPGA implementation as each switch (or diode) only has one current source, which can only be connected to at most two nodes in the netlist (it could be one if one node is connected to the ground), the corresponding vector $\mathbf{z}_s$ or $\mathbf{z}_d$ cannot have more than two non-zero components. Hence, $\mathbf{A}^{-1}\mathbf{H}_{s_k}$ or $\mathbf{A}^{-1}\mathbf{H}_{d_k}$ cannot have more than two non-zero columns. Therefore, except for $\mathbf{A}^{-1}$ and $\mathbf{A}^{-1}\mathbf{H}_{re}$, all other matrices are structured sparse matrices with non-zero values at known positions. Storing these sparse matrices does not consume much OCM.

C. Delayed update for diodes

Another significant challenge stems from the fact that the state ('on' or 'off') of each switch and diode has to be determined ahead of time so that $\mathbf{z}^{n+1}$ can be calculated. This is especially painful for diodes since the choice of the state depends on $\mathbf{x}^n$. It introduces a bubble in the pipeline since the whole system has to halt and wait for the $\mathbf{z}^{n+1}$ to be determined before it can proceed with MVM (solving the linear equation), as is shown in Fig. 5(a). To tackle the second challenge, it must be assured that the MVM can start without knowing the state of all the switches/diodes.



(a) Timing diagram of scheduling with unmodified diode update module



(b) Timing diagram of scheduling with modified diode update module

Fig. 5. Illustration of the benefit of delayed diode updates

To tackle this problem, at the beginning of a simulation, all diode states are assumed to be 'off', i.e., $s_d^{-1} = 0$. When the simulation starts, $\mathbf{x}^0$ is sent to the MVM module and the diode update module. After that, $\mathbf{x}^1$ and the actual diode state s_d^0 are provided. Then, the MVM module can start the next iteration without any halt, as shown in Fig. 5(b). With this scheme, the iteration time can be shortened significantly. The drawback is that the diode's state update is always delayed by one timestep. Strictly speaking, this is not correct in physics. However, since the timestep picked in simulation is much smaller than the switching period, the influence of this delay is negligible.

IV. Implementation

This section discusses the implementation of the proposed FPGA simulation accelerator and its integration with RL algorithms in a host machine. Firstly, the RL-based design method's overall architecture (top design) will be introduced. Secondly, hardware implementation details, including dense MVMs for $\mathbf{A}^{-1}\mathbf{z}_0$ and $\mathbf{A}^{-1}\mathbf{H}_{re}\mathbf{x}^n$, sparse MVMs for all $\mathbf{A}^{-1}\mathbf{H}_{s_k}\mathbf{x}^n$ and $\mathbf{A}^{-1}\mathbf{H}_{d_k}\mathbf{x}^n$, and diode update module. Lastly, the interface program between the FPGA accelerator and the RL-based algorithm on the host machine.

A. Overall architecture

As shown in Fig. 6, the model evaluation module is packaged into a python class. FPGA kernel I/O driver functions are also packaged into a python class, which conceptually includes the hardware accelerator on the FPGA, using Xilinx PYNQ

on XRT platforms. We created an OpenAI-Gym environment based on these two classes to ensure compatibility with popular RL programming APIs. OpenAI-Gym is a popular open-source library that provides standard RL programming APIs.

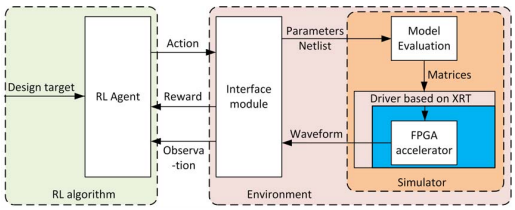


Fig. 6. Architecture of the reinforcement learning-based design method.

The overall architecture of the proposed RL-based design method is shown in Fig. 6. This architecture is based on a recently developed RL algorithm for circuit design, named AutoCkt and proposed by Settaluri et al. [20]. AutoCkt uses ray[rllib], an open-source RL algorithm developer, and the OpenAI-Gym environment to implement a Proximal Policy Optimization (PPO) algorithm to train an RL agent (RL circuit designer). AutoCKT is designed as pure software where ngspice (a free spice simulator) was used as the circuit simulator. In AutoCkt, the agent and environment communicate via the operating system. In our design, the ngspice is replaced by the proposed FPGA accelerator. Efforts are made to ensure that the software and hardware can work together flawlessly with high efficiency. RL algorithm is built in the host machine (a virtual PC in the cloud). The RL agent receives the design specifications (target) from the user. An interface module is responsible for the communication between the RL agent and the circuit simulation accelerator. This module converts an "action" (design parameters should increase, decrease, or remain the same) from the RL algorithm into a netlist and passes it to the model evaluation module. The model evaluation module, considered part of the simulation accelerator, is also created in the host machine (software). It catches the netlist and generates all core matrices needed for the FPGA simulation accelerator to operate. It is worth noting that more than one simulation is needed in one training step in certain applications. An example will be given later. The interface module is also responsible for making observations from the output of the simulation accelerator and computing the reward. Once core matrices are generated in the interface module, they are sent to the FPGA accelerator via Xilinx Runtime Library (XRT). After that, the FPGA simulation accelerator starts running and returns the waveform to the interface module. Benefiting from the Xilinx PYNQ project, the interface module is all implemented with python so that the communication overhead is reduced compared with AutoCkt.

B. Accelerator implementation

1) Matrix-vector multiplication (MVM): The core function in MNA iteration is multiplying a $M \times M$ matrix with a $M \times 1$ vector. We categorize MVMs into two groups. The first group is dense MVM, including MVMs for $\mathbf{A}^{-1}\mathbf{z}_0$ and $\mathbf{A}^{-1}\mathbf{H}_{re}\mathbf{x}^n$, and the second group is structured sparse MVM, including MVMs for all $\mathbf{A}^{-1}\mathbf{H}_{s_k}\mathbf{x}^n$ and $\mathbf{A}^{-1}\mathbf{H}_{d_k}\mathbf{x}^n$. Dense MVM operation is achieved by systolic arrays (SA). An SA is created by cascading M systolic processing elements (PEs) via the AXI stream interface to form a serial-in-serial-out MVM module, as shown in Fig. 7. Each PE handles the multiplication between one row of the $M \times M$ matrix (a $1 \times M$ vector) and the $M \times 1$ vector.

Since the matrices are readily available before $\mathbf{x}$ is generated, the PEs are always ready to accept new data. Therefore, the 'ready' signal is not drawn in Fig. 7(a). The 'user' signal is defined as the first data of $\mathbf{x}$, and the 'last' signal indicates the last data of vector $\mathbf{x}$. Once a new valid $\mathbf{x}$ comes ('valid' is logic '1'), the PE grabs one data from the FIFO (rd_en is set to '1'), and times it with input data. If the 'user' is set as '1' (the first data), the multiplication result is saved to a set of registers (D flip-flops) directly; if 'user' is '0', it is added with the previous data saved in the registers, which performs as an accumulator. Then, all the AXI stream signals are registered and forwarded to the next PE. Once the last data comes ('last' is '1'). The PE gives out a 'valid' signal to the output. This forms a pipelined structure with II=1, and the SA can receive data even when the previous MVM is still running. As is shown in Fig. 7(b), the $\mathbf{x}^n$ (marked in green) comes before its previous timestep $\mathbf{x}^{n-1}$ (marked in blue) completely leaves the SA, allowing for high iteration time-efficiency and fully utilize the benefit of delayed diode update scheme.

As discussed earlier, all matrices for switches and diodes are sparse matrices with no more than two non-zero columns, implying that only two values in vector $\mathbf{x}$ are required. Therefore, only two multiplications and one add operation are required and it can be easily pipelined.

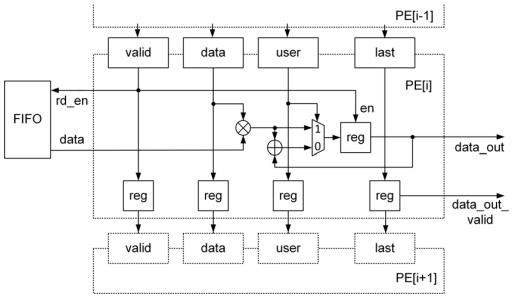
2) Diode update module: When a diode's current state is 'on', the next state depends on the electrical current flowing through it at the current state; when a diode's current state is 'off', the next state depends on the voltage applied across it at the current state.

$$s_d^{n+1} = (s_d^n \ \& \ (i_d > 0)) | (\overline{s_d^n} \ \& \ (v_d > 0)) \tag{5}$$

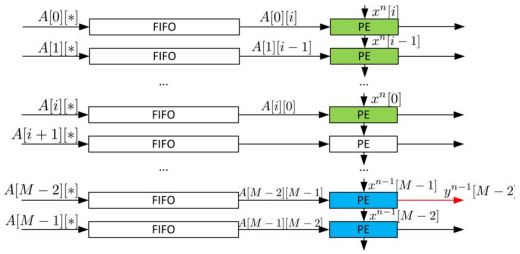
A combinational logic function can be created to achieve it, shown in Eq. 5, where s_d is logic '1' if the diode is 'on' and logic '0' if it is 'off' [22]. Notice that v_d is always compared with '0' since the threshold voltage has been considered in the auxiliary voltage source.

The current i_d^n and the voltage v_d^n are extracted by multiplying a row vector with $\mathbf{x}^n$. For example, the $\mathbf{x}^n$ vector for the example circuit shown in Fig. 4 is $[V_1, V_2, V_3, V_4, V_5, I_e, I_c, I_d]^T$. To pick out the voltage across the diode, we can multiply $[0, 0, 0, 1, 0, 0, 0, 0]$ with $\mathbf{x}^n$; to pick out the current, multiply $[0, 0, 0, 0, 0, 0, 0, 1]$ with $\mathbf{x}^n$.

3) Simulator top design and loop control: The top design of the proposed FPGA circuit simulation accelerator is shown in Fig. 8. A controller is responsible for managing the simulation iterations. The controller receives the $\mathbf{z}_0$ vector from the host



(a) Systolic processing element; reg: D flip-flop type registers; rd_en: read enable of the FIFO



(b) Block diagram of systolic array

Fig. 7. Implementation of the systolic array (SA) for dense matrix-vector multiplication (MVM)

machine (a virtual PC) and forwards it to the corresponding SA MVM module on the rightmost side of Fig. 8 to perform dense MVM for $\mathbf{A}^{-1}\mathbf{z}_0$. At the beginning of the simulation iteration, the controller sends the initial state vector $\mathbf{x}^0$ to the loop for all MVMs. After that, at each simulation timestep, it continuously collects $\mathbf{x}^n$, feeds the $\mathbf{x}^n$ vector at each timestep back into the loop, and sends a copy of $\mathbf{x}^n$ to the host machine. At the last timestep, it sends the result only to the host machine so that the loop becomes idle. The delayed diode update scheme is realized via a small loop between the 'Diode_Controller' and the 'Sparse_MVM_Diode'. At the beginning of the simulation iteration, the 'Sparse_MVM_Diode' sends the diode states, which are all 'off' to the 'Diode_Controller', and the 'Diode_Controller' sends the updated diode states back after the $\mathbf{x}$ at the first timestep is received. Following that, the 'Sparse_MVM_Diode' pushes the received diode states back to the 'Diode_Controller' at every simulation time step until the required number of iterations is reached. It is worth noting that this scheme causes a side effect: in the final iteration, the 'Diode_Controller' still gives a new set of diode states to the sparse MVM module. The 'Sparse_MVM_Diode' must read it and abandon it. Otherwise, the FIFO connecting the two modules will overflow after the simulation times reaches the depth of the FIFO.

As discussed in session III-B, the $\mathbf{z}$ vector is partitioned into four of MVMs. A summation module is used to add all the output vectors.

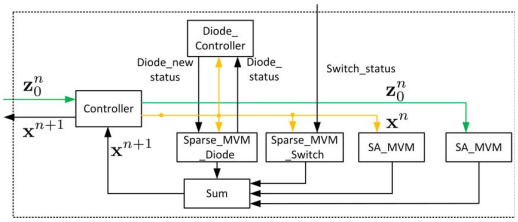


Fig. 8. Top design of the FPGA circuit simulation accelerator

V. Experiments

This section describes the evaluation experiments for the proposed design method and summarizes the experimental results. Multiple aspects of the proposed FPGA circuit simulation accelerator were evaluated, including the accuracy of simulation results (waveform), the speed of the FPGA simulation accelerator, the compatibility with popular RL framework and algorithms, and finally, offline-training speed in comparison with the current state-of-the-art (with the use of computer simulation software). The evaluation experiments were conducted on two widely used power converter topologies. The first is a half-bridge LLC resonant dc-dc converter for data center power supply applications. The second is a full-bridge LLC resonant dc-dc converter for electric vehicle OBC. For comparison purposes, a popular commercial circuit simulation software, PowerSim, optimized for power electronics, was employed. The software simulator was running on a PC.

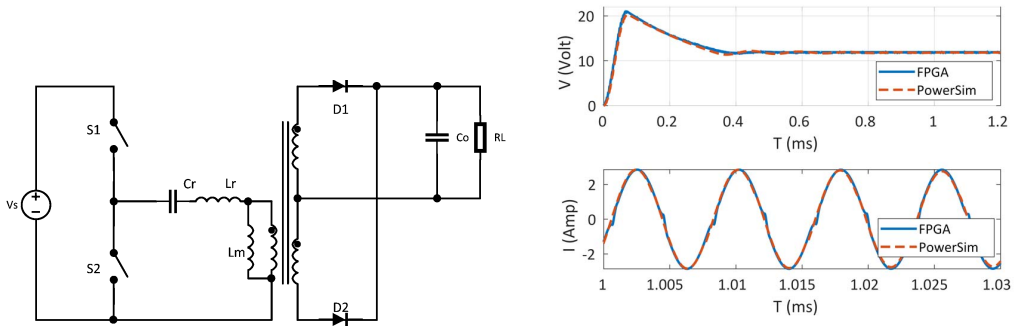
A. Simulation accuracy evaluation

A half-bridge LLC converter circuit was simulated to evaluate the simulation accuracy of the proposed FPGA circuit accelerator, shown in Fig. 9(a). The circuit was simulated using the proposed FPGA accelerator and PowerSim with identical design parameters. The simulation results are shown in Fig. 9(b). The proposed FPGA simulation accelerator produces a similar waveform compared with the PowerSim result. A waveform comparison experiment was also conducted for a full-bridge LLC converter, and the two simulation results matched very well.

B. Training tests

Two design examples were used to demonstrate and evaluate the performance of the proposed RL-based design method.

1) Half-bridge LLC resonant dc-dc converter for data center power supply applications: The first example was to train an RL agent to design half-bridge LLC resonant dc-dc converters with constant output (12 V) for different loads. This design is usually used as the power supply of server machines in data centers. The RL agent (RL circuit designer, essentially a DNN) is trained to learn to design this voltage converter with a dc input of 300V - 400 V at different loads (from 100 W to 300 W). The design parameter space $\mathbf{p}$ contains three sets of values for L_r, C_r, and L_m, forming a high-dimensional (3-D) grid. Each design parameter has 30 choices as the applicable value of the components are limited in the real world due to the



(a) Half-bridge LLC resonant dc-dc converter for testing.

(b) Test result for half-bridge LLC converter

Fig. 9. Circuit simulation accuracy evaluation, $V_s = 400\ V$, $Cr = 24\ nF$, $Lr = 60\ \mu H$, $Lm = 280\ \mu H$, $n_1 : n_2 = 1 : 17$, $Co = 1000\ \mu F$, $R = 0.48\ \Omega$ ($300\ W$), and $f_{sw} = 130\ kHz$

control system, and 30 is enough to cover the range. An action **a** moves the design parameters within 1 grid cell in this design parameter space. An action commands each design parameter to increase/decrease by one grid or remain the same.

According to AutoCkt, the reward function depends on if hard constraints are satisfied. In this case, the hard constraint is if the converter can supply 12V even under the lowest input voltage and highest load. If the hard constraint is not satisfied, the reward is set as

$$r = -5 + \frac{0.6^2 - (V_{out} - 12)^2}{0.6^2 + (V_{out} - 12)^2}$$

. When the hard constraint is satisfied, the reward is set as

$$r = \frac{1}{1 - \eta^2} - 5.263$$

, where η is the efficiency of the circuit calculated by analyzing the root-mean-square (RMS) values and peak values of currents that flows through the circuit [23], [24]. The reward function assures that the reward when hard constraints are satisfied is always higher than the rewards when the hard constraints are not satisfied. When the hard constraints are satisfied, the reward increases as the efficiency increases and becomes positive once the required efficiency (90%) is achieved. This reward function guides the agent to give the maximum efficiency design to get the maximum reward.

An important practical consideration is the tolerance of circuit components. The electronic component's tolerance defines how much its actual value can differ from its nominal value. Additionally, tolerance may come from aging and temperature variation. During offline training, 10% percent tolerance (common in power electronics design) is added. In each simulation, we randomly pick a component value between in $\pm$ 10% of the nominal value to ensure a robust design. Each episode has 25 environment steps within the RL algorithm, which is sufficient for the RL agent to explore the design parameter space to find the optimal design from the initial state. All design parameters are set at the midpoint of their range at the initial state. The offline training stops after 1000 episodes and it takes 4.5 hours, meaning each episode takes 16.2 sec on average and each environment step takes 0.648 sec. A typical training curve (blue) is shown in 10.

A similar design method was built around PowerSim, running on a PC with Intel E3-1245 V5 CPU. The offline training takes 270 hours (over 11 days). The proposed RL-based design method with FPGA simulation accelerator delivers a 60$\times$ speedup to the offline training, significantly improving design productivity.

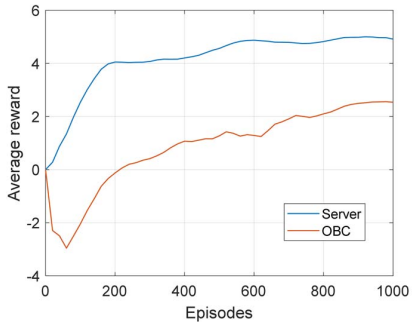


Fig. 10. Average reward for each environment step over time in the training process

2) Full-bridge LLC resonant dc-dc converter for EV charging applications: The second example was to train an RL agent to design full-bridge LLC resonant dc-dc converters for EV battery charger applications. An EV charger typically has three parts, which are an AC-DC rectifier, a power factor corrector, and finally the OBC, which is mostly a full-bridge LLC converter. The first two stages are comparatively easy to be designed with high efficiency as fewer components are required. Therefore, the bottleneck of increasing efficiency is typically the final stage, which is our design target. Since charging a battery requires both constant current mode and constant voltage mode, the hard constraint and reward function differ from the first example. In this example, the hard constraint is that the output voltage can reach the required

maximum voltage to charge the battery when the charging current is at its maximum value. The efficiency is only considered under the full load, where the power loss is at its highest. A training curve (red) is shown in Fig. 10.

C. Resource usage and discussion

According to the report, the simulation iteration loop has an II of 1 and latency of 16 at a clock frequency of 250MHz, which means the equivalent II is $(M+16)/M$, where M is the node size of the FPGA simulation accelerator. The speedup of the simulation acceleration increases as the circuit size increases.

Three FPGA simulation accelerators (bitstreams) were created with different node sizes M (32, 48, 64) and a different number of switches/diodes (2, 4, 8). These three simulation accelerators can be viewed as three containers. A "container" can simulate a circuit with a smaller node size and fewer switches/diodes via zero padding. We can choose the most suitable container based on the size and complexity of the circuit. It is worth noting that more accelerators ("containers") with different configurations (M and a different number of switches/diodes) can be generated in advance in the future.

According to the Xilinx Vitis report, the resource utilization of three different accelerators ($M = 32, 48, 64$) is shown in table I. The largest design ($M = 64$) is sufficient to contain the most complex power converter topology in the field. This design only consumes a small fraction of the total computational resources on the Xilinx vu9p platform. It also suggests the possibility of implementing more than one independent accelerator on a single FPGA.

On the other hand, FPGA cloud service providers, such as AWS F1 instance, allow customers to use multiple FPGAs in parallel. Running simulations in parallel holds the promise to further shorten the training time. For instance, Ray[rllib] supports multi-worker training and remote RL environment workers. We deployed the RL structure with 1, 2, and 4 FPGA workers (Each kernel worker has 1 independent simulation accelerator). Compared with 1 FPGA worker case, the overall training time is reduced by 46% with 2 FPGA kernels and 74% with 4 FPGA kernels.

TABLE I
UTILIZATION UNDER DIFFERENT SETUP

Node Size	$M = 32$	$M = 48$	$M = 64$
Switches	$N_s = 2$	$N_s = 4$	$N_s = 8$
Diodes	$N_d = 2$	$N_d = 4$	$N_d = 8$
LUT	258635 (21.9%)	273988 (23.2%)	284141 (24.1%)
FF	373587 (15.8%)	390139 (16.5%)	407888 (17.3 %)
DSP	330 (4.8%)	522 (7.6%)	778 (11.4%)
BRAM	529 (24.5%)	531(24.6%)	535 (24.8%)

The proposed scheme cannot be easily applied to CPU-based simulators. CPU can benefit from sparse matrix multiplication while $\mathbf{A}^{-1}$ and $\mathbf{A}_{re}^{-1}$ are typically not sparse due to many reactive components in the circuit. Even though the CPU can finish other MVM faster, the CPU has to wait until the two non-sparse MVM are finished. In FPGA, the non-sparse MVM and sparse MVM run at the same speed, so this bottleneck problem doesn't show up.

VI. Conclusion

RL-based design method for power converters is very slow in training. This paper proposes a new FPGA architecture that speeds up an RL-based design method for power converters alleviating this bottleneck. This design method supports all power electronic circuit converters and their variations. HLS was used to build the FPGA accelerator on AWS F1 instance. An AWS virtual PC hosts the training algorithm, interface module, and model evaluation module. The software and hardware are seamlessly integrated via Xilinx PYNQ using the XRT platform. Experimental results showed that the overall speedup of the RL-based design method in comparison with using a popular commercial simulator, PowerSim, is around $60\times$. Our new architecture enables power converter designers to achieve high power conversion efficiency under given constraints and components' tolerance. This design method drastically shortens the proof-of-concept time, reduces time-to-market, boosts productivity, and accelerates the pace of energy innovation.

Acknowledgment

This research is supported in part by the Office of Naval Research (ONR) under grant N00014-17-1-2632, and National Science Foundation (NSF) under grants 2027069 and 2106750.

References

[1] Y. Jeong, M. H. Park, and G. W. Moon, "High-Efficiency Zero-Voltage-Switching Totem-Pole Bridgeless Rectifier with Integrated Inrush Current Limiter Circuit," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 9, pp. 7421–7429, sep 2020.

[2] Y. Jeong, G. W. Moon, and J. K. Kim, "Analysis on half-bridge LLC resonant converter by using variable inductance for high efficiency and power density server power supply," *Conference Proceedings - IEEE Applied Power Electronics Conference and Exposition - APEC*, pp. 170–177, may 2017.

[3] Y. Jeong, J. D. Park, R. Rorrer, K. W. Kim, and B. H. Lee, "A Novel Multi-input and Single-output DC/DC Converter for Small Unmanned Aerial Vehicle," *Conference Proceedings - IEEE Applied Power Electronics Conference and Exposition - APEC*, vol. 2020-March, pp. 1302–1308, mar 2020.

[4] T. Dragicevic, P. Wheeler, and F. Blaabjerg, "Artificial Intelligence Aided Automated Design for Reliability of Power Electronic Systems," *IEEE Transactions on Power Electronics*, vol. 34, no. 8, pp. 7161–7171, aug 2019.

[5] T. Instruments, "*llc* resonant half-bridge converter, 300-w evaluation module," *SLUU361 datasheet, Apr*, 2009.

[6] S. Abdel-Rahman, "Resonant llc converter: Operation and design," *Infineon Technologies North America (IFNA) Corp*, vol. 19, 2012.

[7] A. Bhat, "A generalized steady-state analysis of resonant converters using two-port model and fourier-series approach," *IEEE Transactions on Power Electronics*, vol. 13, no. 1, pp. 142–151, 1998.

[8] N. Shafiei, M. A. Saket, and M. Ordonez, "Time domain analysis of llc resonant converters in the boost mode for battery charger applications," in *2017 IEEE Energy Conversion Congress and Exposition (ECCE)*, 2017, pp. 4157–4162.

[9] Y. Wei, T. Pereira, Y. Pan, M. Liserre, F. Blaabjerg, and H. A. Mantooth, "Rms current based automated optimal design tool for llc resonant converters," in *2022 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2022, pp. 2146–2153.

[10] M.-H. Park, Y. Jeong, R. A. L. Rorrer, D. Choi, and G.-W. Moon, "Hold-up time extension method for llc resonant converter by detecting operation region," *IEEE Transactions on Power Electronics*, vol. 35, no. 10, pp. 9949–9952, 2020.
[11] S. Wang, Y. L. Murphey, W. Su, M. Wang, V.-H. Bui, F. Chang, C. Huang, L. Xue, F. Leno, D. Silva, and R. Glatt, "An Intelligent System for Automatic Selection of DC-DC Converter Topology with Optimal Design," nov 2021.
[12] S. Zhao, F. Blaabjerg, and H. Wang, "An overview of artificial intelligence applications for power electronics," *IEEE Transactions on Power Electronics*, vol. 36, no. 4, pp. 4633–4658, apr 2021.
[13] P. Kofinas, S. Doltsinis, A. I. Dounis, and G. A. Vouros, "A reinforcement learning approach for MPPT control method of photovoltaic sources," *Renewable Energy*, vol. 108, pp. 461–473, 2017.
[14] H. Wang, J. Yang, H.-S. Lee, and S. Han, "Learning to Design Circuits," dec 2018. [Online]. Available: http://arxiv.org/abs/1812.02734
[15] Z. Zhao and L. Zhang, "Deep reinforcement learning for analog circuit sizing," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 2020-October, 2020.
[16] N. Kapre, "SPICE^2 - A Spatial Parallel Architecture for Accelerating the SPICE Circuit Simulator," Ph.D. dissertation, 2010. [Online]. Available: https://thesis.library.caltech.edu/6159/
[17] L. Han, X. Zhao, and Z. Feng, "TinySPICE: A parallel SPICE simulator on GPU for massively repeated small circuit simulations," in *Proceedings - Design Automation Conference*, 2013.
[18] L. Herrera and J. Wang, "FPGA based detailed real-time simulation of power converters and electric machines for EV HIL applications," in *2013 IEEE Energy Conversion Congress and Exposition, ECCE 2013*, 2013, pp. 1759–1764.
[19] Y. P. Siwakoti and G. E. Town, "Design of FPGA-controlled power electronics and drives using MATLAB Simulink," in *2013 IEEE ECCE Asia Downunder - 5th IEEE Annual International Energy Conversion Congress and Exhibition, IEEE ECCE Asia 2013*, 2013, pp. 571–577.
[20] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, "AutoCkt: Deep Reinforcement Learning of Analog Circuit Designs," in *Proceedings of the 2020 Design, Automation and Test in Europe Conference and Exhibition, DATE 2020*, 2020, pp. 490–495.
[21] L. M. Wedepohl and L. Jackson, "Modified nodal analysis: An essential addition to electrical circuit theory and analysis," *Engineering Science and Education Journal*, vol. 11, no. 3, pp. 84–92, 2002. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/esej_20020301
[22] P. Pejović and D. Maksimović, "A Method for Fast Time-Domain Simulation of Networks with Switches," *IEEE Transactions on Power Electronics*, vol. 9, no. 4, pp. 449–456, 1994.
[23] S. Maniktala, *Switching Power Supplies A-Z*, 2012.
[24] S. Mulder, "Power ferrite loss formulas for transformer design," *Power-conversion and Intelligent Motion*, vol. 21, no. 7, pp. 22–31, 1995.