IEEE Open Journal of
Power Electronics

Received 15 October 2020; revised 14 November 2020; accepted 16 November 2020. Date of publication 18 November 2020; date of current version 2 December 2020. The review of this article was arranged by Associate Editor Zhixiang Zou.

Digital Object Identifier 10.1109/OJPEL.2020.3039117

Machine Learning Building Blocks for Real-Time Emulation of Advanced Transport Power Systems

SONGYANG ZHANG (Student Member, IEEE), TIAN LIANG (Member, IEEE), AND VENKATA DINAVAHI (Fellow, IEEE)

Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta T6G 2V4, Canada

CORRESPONDING AUTHOR: TIAN LIANG (e-mail: tliang5@ualberta.ca)

This work was supported by the Natural Science and Engineering Research Council of Canada (NSERC).

ABSTRACT The revolution of artificial intelligence (AI) is transforming major industries worldwide. With accurate inferencing, AI has caught the attention of many engineers and scientists. Promisingly, hardware-in-the-loop (HIL) emulation can adopt this type of modeling method as one of the alternatives after comprehensive investigation. This paper proposes an approach for emulating power electronic motor drive transients for advanced transportation applications (ATAs) using machine learning building blocks (MLBBs) without any traditional circuit-oriented transient solver. The more electric aircraft (MEA) power system is chosen as a case study to validate the real-time emulation performance of MLBBs. Inside MLBBs, neural networks (NNs) have been applied to build component-level, device-level, and system-level models for various equipment. These models are well trained in a cluster and transplanted into the field-programmable gate array (FPGA) based hardware platform. Finally, MLBB emulation results are compared with PSCAD/EMTDC for system-level and SaberRD for device-level, which showed high consistency for model accuracy and high speed-up for hardware execution.

INDEX TERMS Artificial intelligence (AI), field -programmable gate arrays (FPGAs), gated recurrent units (GRU), hardware -in-the-loop (HIL), insulated -gate bipolar transistor (IGBT), long short-term memory (LSTM), machine learning (ML), more electric aircraft (MEA), power electronics, real-time systems, recurrent neural network (RNN), silicon carbide (SiC).

LIST OF ABBREVIATIONS

AES	All-Electric Ship
ATA	Advanced Transportation Application
ATRU	Auto-Transformer Rectifier Unit
CRNN	Classical Recurrent Neural Network
ESS	Energy Storage System
GRU	Gated Recurrent Unit
IP	Intellectual Property
LR	Learning Rate
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MEA	More Electric Aircraft
MLBB	Machine Learning Building Block
MSE	Mean Squared Error
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent

I. INTRODUCTION

Increasing adoption of a renovated power electronic drive system has been witnessed in the advanced transportation application (ATA) of more electric aircraft (MEA) [1], [2], all-electric ship (AES)[3], [4], traction [5], etc. The reason behind these ATAs is highly related to their lower-cost ownership and substantially increased system reliability. Innovative power electronics is the key enabling technology in the reduction of physical weight and fuel usage in ATAs. Thus, there is an urgent need to develop the hardware-in-the-loop (HIL)

emulation of these compact power electronic systems under harsh operating environments.

The biggest challenge of the current HIL emulation technique is the limitation of the computation power based on the traditional electromagnetic transient (EMT) algorithms. To be more specific, the increasingly integrated power electronic system introduces excessive system nodes into the circuit network, which results in heavy execution delay for getting the final solution. Recently, the development of artificial intelligence (AI) and its application-specific integrated circuit (ASIC) [6] give a new possibility to represent the next-generation circuit solver. These newly developed AI neural network (NN) models are forecasting methods based on nonlinear mathematical equivalent, which has been applied in the areas of face verification [8], image resolution processing [7], human action recognition [9], and natural language processing [10]–[12]. The ideally suited hardware for the NNs inherent massive parallel network structure that can achieve high-execution efficiency is the field-programmable gate array (FPGA) which is a configurable logic block (CLB) matrix with programmable interconnects.

This paper proposes a novel machine learning building blocks (MLBBs) concept to emulate the power system transients of ATA on high bandwidth memory (HBM) integrated high-performance Xilinx Virtex UltraScale+ FPGA hardware platform. The MEA system is selected as the case study to validate the model accuracy and execution efficiency of MLBBs. The paper is organized as follows: Section II introduces the background of machine learning (ML). Section III describes the modeling method in the ATA system for component-level, device-level, system-level, and hybrid models. Section IV studies the parameter design, training skills, and hardware implementation of MLBBs on the Xilinx VCU128 hardware platform. Section V shows the results of MLBBs emulation and comparisons between this method and commercial software, and finally, Section VI gives the conclusion.

II. BACKGROUND ON MACHINE LEARNING

In this section, general ML methods for the modeling equipment in power conversion systems are discussed, including traditional artificial neural network (ANN) [13], [14] and recurrent neural network (RNN). As for RNN, there are three common types: classical recurrent neural network (CRNN) [15], long short-term memory (LSTM) [16], and gated recurrent units (GRU) [17]. Although all these methods are fundamental NNs today, they have different performances for modeling ATAs. Furthermore, the algorithms and structures of NNs are illustrated in this section.

A. TYPES OF NEURAL NETWORK CELLS

NNs are designed by imitating the human brain's neuronal construction, and the fundamental unit is called a neuron, as shown in Fig. 1(a). Several independent neurons can be combined into a traditional ANN layer. Each NN has at least three layers: input layer, hidden layer, and output layer. Today, a widespread network called convolutional neural network

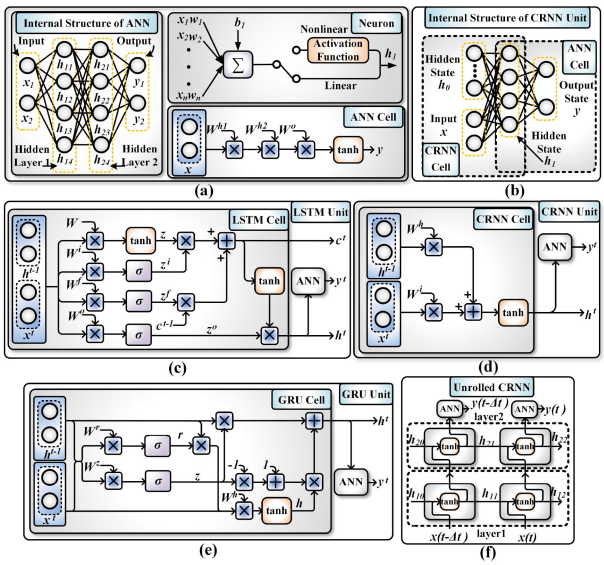

FIGURE 1. Neural network structure: (a) ANN; (b) internal structure of CRNN unit; (c) LSTM; (d) CRNN; (e) GRU; and (f) unrolled CRNN.

(CNN) has become a hot spot of research in many applications. Compared with CNN, RNNs can deal with information related to the past and the present. They can learn from the historical data in a time series and then output their prediction.

In Fig. 1(d), a typical CRNN application unit has a CRNN cell and sometimes includes an ANN cell counter-intuitively. This is because the hidden state h transmitted by CRNN cells has a larger dimension (more information is contained), but it is not expected in the output. In order to change the dimension and obtain the desired predicted value y, an ANN cell is necessary. Then, Fig. 1(b) clearly shows how the internal structure of the CRNN application combines a CRNN cell and an ANN cell. Obviously, CRNN can be seen as multiple copies of the same NN cell, and each NN cell passes the message to the next cell. Therefore, if this loop is unrolled, it can be displayed as a list of NNs as shown in Fig. 1(f).

LSTM is a designed RNN to solve gradient disappearance and gradient explosion during long sequence training [18]. In Fig. 1(d), the calculated $\mathbf{Z}_f$ (f represents forget) is used as the forget gate to control the value left or forgotten in $\mathbf{C}^{t-1}$(also known as cell state) of the previous state. Then, the selected gating signal is controlled by $\mathbf{Z}_i$ (i stands for information). It mainly selects and memorizes the input $\mathbf{X}^t$ and $\mathbf{H}^{t-1}$, which records the important information and drops others. Finally, the output gate, controlled by $\mathbf{Z}_o$, scales $\mathbf{C}^t$ (changes through a $tanh$ activation function) obtained in the previous stage.

As shown in Fig. 1(e), GRU, another popular kind of RNN, was proposed in [17]. It is a substitute for LSTM with long-term predictive capability, which can be applied to time-series prediction (e.g., traffic flow prediction [19]). The forget gate $\mathbf{Z}^f$ and the input gate $\mathbf{Z}^i$ in LSTM are combined into a single update gate $\mathbf{Z}$ in GRU, which makes GRU simpler than LSTM. The reset matrix $\mathbf{R}$ is obtained from input $\mathbf{X}^t$ and previous state $\mathbf{H}^{t-1}$, then it resets information in $[\mathbf{X}^t, \mathbf{H}^{t-1}]$ as

TABLE 1. Comparison of ML Models With Time-Series Signals

Feature	ANN	CRNN	LSTM	GRU
Complexity	+	++	++++	+++
Execution Time	+	++	++++	+++
Resource Consumption	++++	+	+++	++
Accuracy	+	++	++++	+++
Long-Term Prediction	No	No	Yes	Yes

$\mathbf{H}'$. Similar to the reset stage, the update matrix $\mathbf{Z}$ is calculated by input $\mathbf{X}^t$ and previous state $\mathbf{H}^{t-1}$. After that, the present hidden state $\mathbf{H}^t$ is updated by $\mathbf{Z}$, $\mathbf{H}^{t-1}$ and $\mathbf{H}'$. The comparison of these NNs with time-series inputs is shown in Table 1. Among these NNs, ANN is the simplest NN with minimum execution time and accuracy but cost maximum resource consumption. As for RNNs (CRNN, LSTM, and GRU), the more accurate it is, the more complex the structure, the longer the execution time, and the more resources consumption it needs. Furthermore, LSTM and GRU have the capability of long-term prediction.

B. COMPLEXITY OF NEURAL NETWORK CELLS

From Fig. 1, it is clear that the calculation burden of different kinds of RNNs vary with their structure. Generally, the complexity for NNs algorithms is studied and displayed as $O()$ (e.g., $O(n^3)$ is the complexity of the linear ANN with one hidden layer and the matrix multiplication). However, the complexity function $O()$ can not demonstrate the precise execution times because $O()$ only focuses on the function or the orders of magnitude (e.g. $O(n)$, $O(n^3)$, $O(log(n))$, etc.). The general structure RNNs and their complexity are classified in [20]. Their complexity measurement is divided into three parts: recurrent depth, feedforward depth, and recurrent skip coefficient [20]. In this paper, the recurrent skip coefficient is trivial since the number of layers is 1, which will be further discussed in Section IV, and the accurate execution time for each kind of RNN cells mainly depends on the feedforward depth. The execution time expressed as T is given as:

$$\begin{aligned} T_{ANN} &= i\cdot h\cdot(2h-1)+o\cdot h\cdot(2h-1)+o\cdot T_{\tanh} \\ &\approx 2i\cdot h^2+2o\cdot h^2+o\cdot T_{\tanh}, \end{aligned} \tag{1}$$

$$\begin{aligned} T_{CRNN} &= i\cdot h\cdot(2h-1)+h^2\cdot(2h-1)+h^2+h\cdot T_{\tanh} \\ &\approx 2i\cdot h^2+2h^3+h\cdot T_{\tanh}, \end{aligned} \tag{2}$$

$$\begin{aligned} T_{LSTM} &= 4(i+h)\cdot h\cdot(2h-1)+3h^2\cdot(2h-1) \\ &\quad +h^2+2h\cdot T_{\tanh}+3h\cdot T_{sigm} \\ &\approx 8i\cdot h^2+14h^3+2h\cdot T_{\tanh}+3h\cdot T_{sigm}, \end{aligned} \tag{3}$$

$$\begin{aligned} T_{GRU} &= 3(i+h)\cdot h\cdot(2h-1)+4h^2\cdot(2h-1)+2h^2 \\ &\quad +i\cdot h\cdot(2h-1)+h\cdot T_{\tanh}+2h\cdot T_{sigm} \\ &\approx 6i\cdot h^2+14h^3+h\cdot T_{\tanh}+2h\cdot T_{sigm}, \end{aligned} \tag{4}$$

where i, h and o mean the input size, the hidden size and the output size of NN, respectively. Then, $T_{\tanh}$ and T_{sigm} are the execution time of activation function $tanh()$ and $sigmoid()$.

For $n\times m$ matrix N and $m\times p$ matrix P, their multiplication has $n\times m\times p$ times multiplications and $n\times(m-1)\times p$ times additions. Therefore, from Fig. 1(a), (d), (e) and (f), the execution times of NN cells can be analyzed by matrix multiplication.

III. MACHINE LEARNING MODELING FOR ATA

Among ATAs, the MEA is a promising system because of its straightforward and energy-saving structure, which has been widely studied [21]–[23]. Hence, the ATA topology similar to that of the Boeing-787 MEA microgrid [24] is taken as the case study in this paper. In Fig. 2(a), the whole system consists of a synchronous generator, three auto-transformer rectifier units (ATRU), two permanent magnet synchronous motor (PMSM) drive systems, and an energy storage system (ESS). All kinds of equipment can also be classified into three categories of MLBBs: component-level, device-level, and system-level. The classification is based on the complexity and the function of these equipment. Then, to model these equipment, a certain kind of NN technology can be applied, or one equipment can be modeled as a hybrid model. The whole system is built in PSCAD/EMTDC to obtain the dataset for system-level ML models' training. Then, the particular part, the ESS, is built in SaberRD for device-level dataset.

A. COMPONENT-LEVEL MODELS

Component-level ML models for ATA are built for only two components: inductor and capacitor. As the structure of these components is simple, component-level models are the simplest ML models. In Fig. 3(a), the traditional inductance model can be described as (5); for the discrete-time solution, the difference equation can be obtained as (6) [25], given as:

$$v_m - v_n = L\frac{di_{mn}}{dt}, \tag{5}$$

$$\begin{aligned} &\frac{v_m(t)-v_n(t)+v_m(t-\Delta t)-v_n(t-\Delta t)}{2} \\ &\quad = L\frac{i_{mn}(t)-i_{mn}(t-\Delta t)}{\Delta t}. \end{aligned} \tag{6}$$

To calculate the present value, a history message, marked as $hist(t)$ in (7), can be utilized from the solution at the previous time-step. Finally, the iterative equations are established as (8) and (9):

$$\begin{aligned} hist_{mn}(t-\Delta t) &= i_{mn}(t-\Delta t) \\ &\quad +\frac{\Delta t}{2L}\left\{v_m(t-\Delta t)-v_n(t-\Delta t)\right\}, \end{aligned} \tag{7}$$

$$i_{mn}(t) = hist_{mn}(t-\Delta t)+\frac{\Delta t}{2L}\left\{v_m(t)-v_n(t)\right\}, \tag{8}$$

$$hist_{mn}(t) = hist_{mn}(t-\Delta t)+\frac{\Delta t}{L}\left\{v_m(t)-v_n(t)\right\}. \tag{9}$$

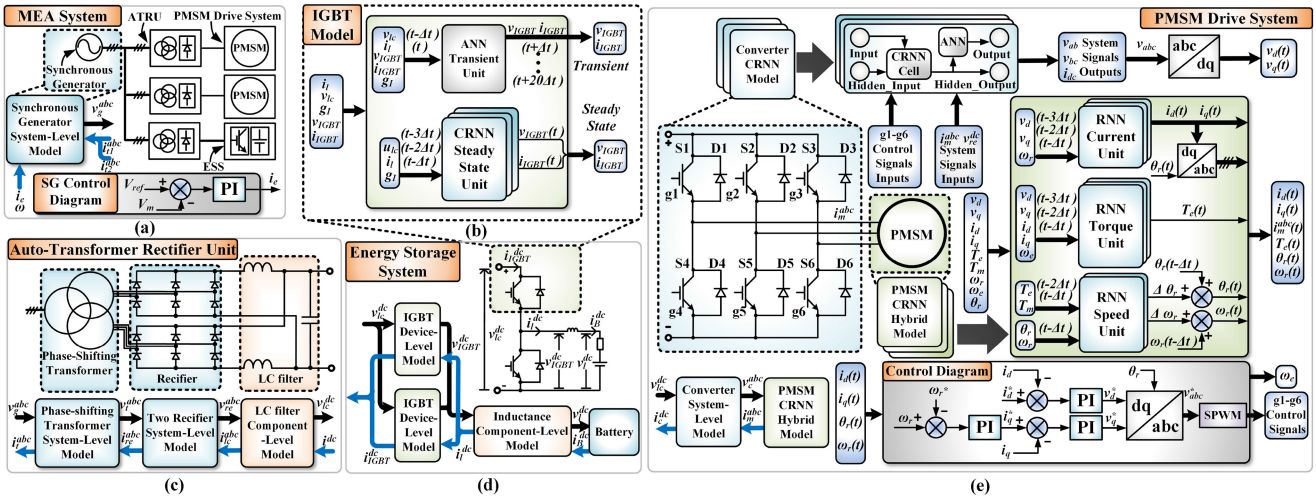

FIGURE 2. MEA power system: (a) overall system topology; (b) SiC IGBT-based device-level hybrid model; (c) ATRU; (d) ESS; and (e) PMSM drive system.

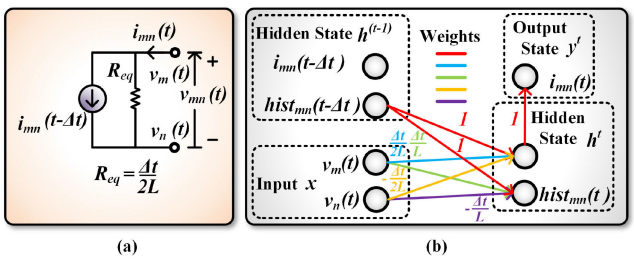

FIGURE 3. Component-level models: (a) traditional modeling schematic. (b) component-level ML modeling schematic.

Equations (8) and (9) are applied in the traditional transient simulation, and they can also be transferred into a CRNN model, as shown in Fig. 3(b). When the value L is known, the weights and bias of CRNN can be determined from (8) and (9) without any training process. Arrows of each colour in Fig. 3(b) represent different parameters in traditional equations (8) and (9), which vividly shows how the traditional model is transformed into the ML model.

B. DEVICE-LEVEL MODELS

Device-level models have the same NN structure as the system-level models, but two differences should be highlighted:

1) Nanosecond-level time-step. The interval is 50 *ns* to describe the device-level transient processes. This means increased massive datasets, challenging training, and complex structures of the device-level models.

2) Enlarged dimension training dataset. The dataset prepared for the detailed models is more demanding to obtain. For nonlinear silicon carbide (SiC) insulated-gate bipolar transistor (IGBT) device-level models, the comprehensive current and voltage waveform during turn-on and turn-off transients must be included to retrain the models.

In this work, device-level models are applied in the ESS (shown in Fig. 2(b) (d)), and here it is built as a hybrid model instead of a single ANN or RNN model. Hence, it is elaborated in the Section III *D*.

C. SYSTEM-LEVEL MODELS

The system-level ATA transients can be conducted at a larger time-step, and this is significant to reduce hardware resource while device-level requires much resource demand. Then, as device-level models focus on details and update with small time-step, system-level models can respond faster with larger time-step and less execution. Therefore, system-level models (e.g., converters, rectifiers, transformers, synchronous generators, and PMSMs) are trained by system-level data and built with fewer hidden sizes and layers. These models can be simplified to a certain extent through this strategy, diminishing the consumption of hardware resources and reducing training difficulty. They may not output the model's detailed transient with a small interval as device-level models do but have high accuracy in their system-level application. The interval for system-level models in this paper is 1 μs for ATA.

Almost all the equipment can be modeled as system-level models by their electrical connections, displayed in Fig. 2(a), (c), and (e). For example, the inputs of the rectifier model is i_{lc}^{abc} (connected to LC filter) and v_t^{abc} (connected to transformer); its output signals are i_{re}^{abc} (connected to transformer) and v_{re}^{dc} (connected to LC filter). There are two exceptions: the converter and the synchronous generator. The converter needs six control signals as inputs from the motor control module, and the details of the converter ML model are shown in Fig. 2(e) PMSM drive system. As for the synchronous generator, the excitation current i_e and rotor speed ω are also necessary for its inputs, as shown in Fig. 2(a).

D. HYBRID MODELS

Hybrid models are a broader term which means a piece of equipment does not consist solely of a single type of NN but includes multiple NNs of the same or different kinds, or even

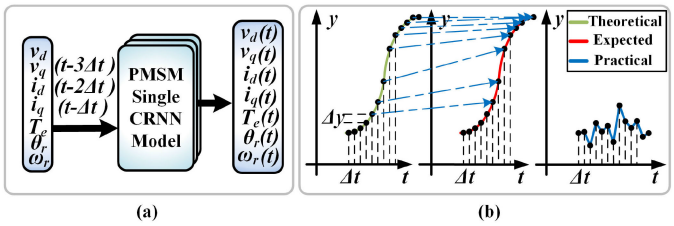

FIGURE 4. PMSM single NN model: (a) schematic; (b) performance comparison.

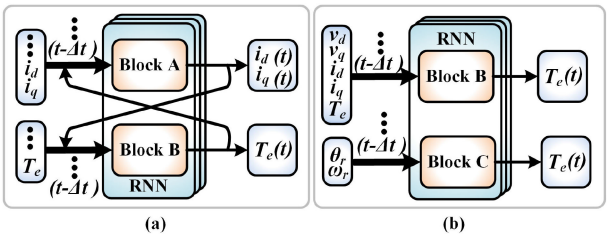

FIGURE 5. Causes of pitfalls: (a) internal interlock; (b) logical topsy-turvydom.

a mixture of NNs and traditional models. There are two main reasons for hybrid models instead of a single NN model.

1) High-efficiency execution: A SiC IGBT device-level hybrid model is shown in Fig. 2(b). It consists of two parts: a device-level ANN model (deals with the transient processes with five input signals) and a system-level RNN model (deals with the steady-state processes with three input signals in time-series). Undeniably, SiC IGBT can be modeled as only one NN (an ANN or an RNN with long time-series input signals). However, it leads to a complex structure and causes a heavy computational burden. What is worse, higher execution delay and increased hardware resources are expected in training processes. In contrast, when the SiC IGBT model is separated into two parts, each part can be modeled by a suitable NN. Each part only focuses on specific applications and takes advantage of their different structures. Compared with the single ML model, the hybrid ML model can be divided into smaller ML parts with empirical knowledge, which is a powerful example of the interdisciplinary application of power electronics and ML.

2) Accessibility of training and implementation: Although one single NN for one piece of equipment is intuitive, however, this is not easy to achieve sometimes. A PMSM system-level hybrid model is shown in Fig. 2(e), and PMSM can be built as a single CRNN model, shown in Fig. 4(a). In order to discuss the differences between these two models, the analytical equations of PMSM are stated first:

$$v_q = Ri_q + \omega_e \lambda_d + \frac{d\lambda_q}{dt}, \tag{10}$$

$$v_d = Ri_d - \omega_e \lambda_q + \frac{d\lambda_d}{dt}, \tag{11}$$

$$\lambda_q = L_q i_q, \tag{12}$$

$$\lambda_d = L_d i_d + \lambda_{pm}, \tag{13}$$

where v_d, v_q, i_d and i_q represent the dq axis voltage and current of PMSM; L_d, L_q and R mean its dq axis inductance and resistance; λ_{pm}, λ_d and λ_q are the flux linkage of permanent magnets and dq axis flux linkage in the stator; and ω_e is the electrical supply frequency. Then, to obtain i_q and i_d, new expression can be derived from (10)–(13) or simply expressed as the following nonlinear model:

$$\{i_q, i_d\} = f\left(v_q, v_d, \omega_e\right). \tag{14}$$

Even no parameters (e.g., L_d, L_q, R, λ_{pm}, etc.) are known, (14) can be learned and established by an RNN current unit, which is a CRNN model and a part of the PMSM hybrid model.

As for the electric torque T_e, it can be calculated as:

$$T_e = \frac{3}{2}\frac{p}{2}\left[\lambda_{pm} i_q + \left(L_d - L_q\right) i_d i_q\right], \tag{15}$$

and T_e can be also obtained by an RNN torque unit, which is a CRNN model and a part of the PMSM hybrid model.

When it comes to the mechanical section in PMSM, the relationship between its speed and torque is given as:

$$J\frac{d\omega_r}{dt} = T_e - T_m - B\omega_r, \tag{16}$$

$$\frac{d\theta_m}{dt} = \omega_r, \tag{17}$$

where ω_m, T_e, T_m, J, B, θ_m are the mechanical rotor speed, electrical torque, mechanical torque, moment of inertia, friction factor, and rotor position angle, respectively. From (16) and (17), ω_m and θ_m can be learned and calculated by an RNN speed unit, which is a CRNN model and a part of the PMSM hybrid model.

In Fig. 4(b), the left curve is the theoretical one that can be observed from the real PMSM and the middle curve is the PMSM single CRNN model prediction when it is trained by the observed data. Although this well-trained PMSM single NN model predicts accurate waveform when the inputs are correct, it may not work well (as shown in Fig. 4(b) the right curve) in a closed-loop system. As the interval of model's time-step is made small, the output $y(t)$ are so close to $y(t-1)$ that the ML model just takes $y(t-1)$ as $y(t)$, and it could not learn the Δy ($\Delta y = y(t) - y(t-1)$). Δy is less than 0.1% y and the model will consider this difference as tolerant. These outputs of a PMSM single NN model strongly depend on its inputs. If correct inputs are given, expected outputs will be obtained. However, when the inputs with small errors are given, the PMSM single NN model will not be stable, and it is hard to output the trend like the result of iteration in traditional processing.

Even if the time interval is large enough, there are still traps for the PMSM single NN model: 1) Internal interlock in Fig. 5(a). Block A is the mathematical logic of i_d and i_q generators by T_e in the CRNN, while Block B is the opposite. i_d and i_q are strongly connected with T_e, which means they work as a positive feedback system. This results in a model

that is oscillating rather than stable in a closed-loop system. 2) Logical topsy-turvydom in Fig. 5(b). Block B is the mathematical logic of a T_e generator by i_d and i_q in the CRNN, while Block C is the anti-causal mathematical logic of a T_e generator by ω_r and θ_m in the CRNN. T_e should be decided by the voltage and current; then T_e influences ω_r and θ_m. However, since T_e has relation with ω_r and θ_m the single ML model may calculate T_e by the input ω_r and θ_m as anti-causal Block C in the single NN model. This is a causal problem that the effect decides the cause, because NNs can only learn correlation but not causation.

All these pitfalls of the PMSM single NN model can be avoided when the model is divided into several parts, and each part, which is built by the ML model, works like the traditional equations in the process. Moreover, as each part only focuses on fewer variables, it is easier to be trained than a PMSM single NN model.

IV. MLBB IMPLEMENTATION FOR REAL-TIME ATA SYSTEM HIL EMULATION

The MLBBs approach can exploit FPGA technology for parallel HIL emulation. In this section, dataset processing and the parameter design for NNs modeling are introduced; then, the training skills, hardware platform as well as the comparison between RNNs are also discussed; lastly, the hardware resource consumption is analyzed.

A. DATASETS

Datasets and normalization are among the most important parts of ML training because they significantly affect the results. To guarantee models' generality, various working conditions of the equipment need to be included in the datasets. The data collection for SiC IGBT model is taken as an example: the topology of ESS in SaberRD is the same as that on FPGA, but the working voltage and current provided by the two-quadrant buck converter are broad to train the versatile SiC IGBT ML model. Some of these working conditions are similar to that on FPGA, but others are different. When the SiC IGBT ML model is built, it can be applied to other kinds of power converters. It is worth mentioning that the dataset does not require sampling of dense and continuous data for training. The training dataset can sample from the original dataset with a relatively large interval. Then, all the data should be normalized to $(-1,1)$ by min-max normalization or z-score normalization, which is a linear transformation used for improving ML performance.

B. PARAMETERS FOR MODELS

Before the discussion about the parameters, a criterion that measures the performance is introduced. The mean squared error (MSE) is a recommended criterion for evaluation in ML, and the mean absolute error (MAE) is another criterion to measure errors. Their results in these models are similar, but MAE has a more stable value over the whole dataset. Without loss of generality, MAE is chosen in this work, given as:

$$MAE = \sum_{i=1}^{n} \frac{\left|y_i^{pre} - y_i\right|}{n}. \tag{18}$$

The MAE is applied to describe error between expectation and prediction in the AI domain, which is the metric to evaluate model effectiveness. y^{pre} is the predictive output of NNs, y is the expected output, and n is the number of the outputs. Then, the training process is iterative and based on stochastic gradient descent (SGD) optimization algorithm [26] to obtain the best weights and minimum error. Adam algorithm [27], which is the most popular SGD optimization algorithm, is utilized to minimize the model's MAE in this work. An essential parameter for NNs is the layer size. The more layers NNs have, the more accurate they will be. However, increasing layers also lead to significantly higher hardware resource, latency, and execution time. When the layer size is 1, the models can meet application requirements by adjustable hidden size and sequence length. Hence, the default layer size for all models is 1. A PMSM single CRNN model is studied so as to evaluate the training results for different pairs of hidden-size coefficient and sequence length when the layer size is 1, as shown in Fig. 6(a).

In Fig. 6(a), the hidden-size coefficient means the multiple of the number of neurons in the hidden layer relative to that in the input layer, and the sequence length is how many RNN calls it has in one layer. From the result in Fig. 6(a), the suitable parameters for modeling power system's equipment is in the red circle. In this paper, the parameters (hidden size is approximately 4 times the input size, and the sequence length is 3) are the general parameters for all RNNs. Nevertheless, they change with the complexity of the models.

C. TRAINING AND COMPARISON BETWEEN RNNS

When the hidden-size coefficient is 4, and the sequence length is 3 in a single PMSM model, the MAE of different kinds of RNN is shown in Fig. 6(b). As can be seen, LSTM has the best performance among these three RNNs; GRU's MAE is close to LSTM's; CRNN has a similar result when the sequence length is small. However, the sequence length is usually less than 4, and CRNN causes much less computational burden, making CRNN the best one for the present applications.

The training for NNs is time-consuming. There are various methods to train better: 1) Data shuffling: to avoid overfitting, the data should be shuffled before sending it to the training program; 2) Varying learning rate (LR): during the training, LR should decrease gradually to obtain better performance with less epoch of training. Both data shuffling and varying LR are used in this work. Fig. 7(a) shows results from the model trained with shuffling data, while Fig. 7(b) is the one without data-shuffling training. This model, trained without data shuffling, may focus on a small part and quickly become overfitted. Fig. 7(c) and (d) show how varying LR works: with the varying LR, the models can get closer to the best-weight point, while those that have constant LR may swing their

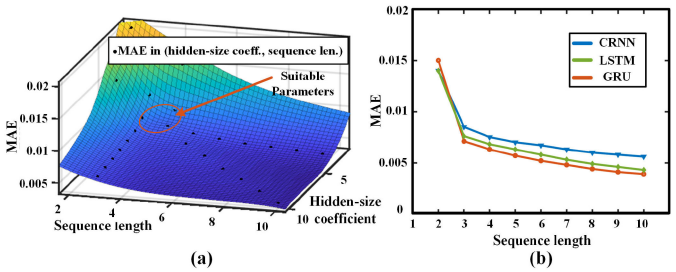

FIGURE 6. MAE comparisons: (a) error at different sequence length and hidden-size coefficient; (b) CRNN vs GRU vs LSTM.

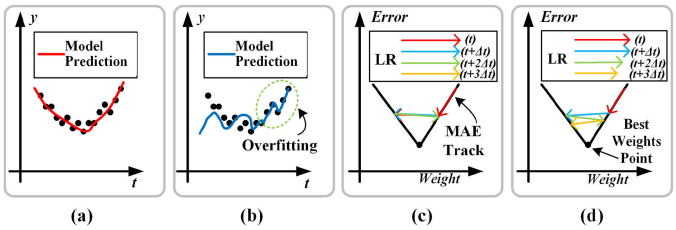

FIGURE 7. Training results and processes. (a) results with data-shuffling training; (b) results without data-shuffling training; (c) training with constant learning rates; (d) training with varying learning rates.

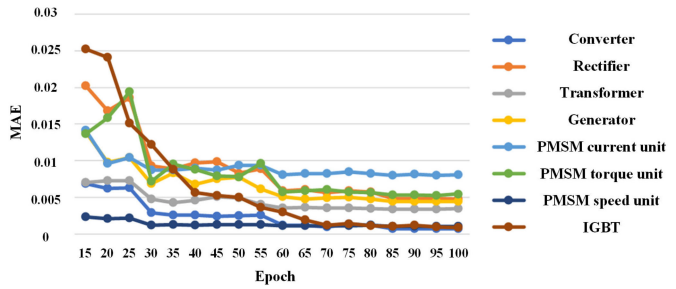

FIGURE 8. Models' MAE during training processes.

weights around the best-weight point. The LR in traditional SGD is constant, but the Adam algorithm has varying LR, and the initial LR can be adjusted to narrow the range of LR in the Adam algorithm. Without these methods, the trained models may be unavailable, or they may have low accuracy.

After the parameter design, determination of NN types, and choice of training optimization strategy, models were trained in a cluster with 196 nodes. Each node is comprised of two Intel Silver 4216 Cascade Lake central processing units (CPUs), four Nvidia V100 Volta graphics processing units (GPUs), and 187 *GB* memory. A maximum of 8 cluster nodes were used for training MLBB models. While a single model may cost 12–24 *h* on personal computers (PCs) for training, it only takes 6–12 *h* on a cluster node. Furthermore, since each node can train one or two models, numerous training processes can run simultaneously in the cluster, whereas less than four training processes can run in parallel on the PC. Eventually, the training results are obtained from the cluster, including weights, biases, as well as errors during training. In Fig. 8, MEAs of different CRNN models are displayed during the training process. These MAEs are calculated by the

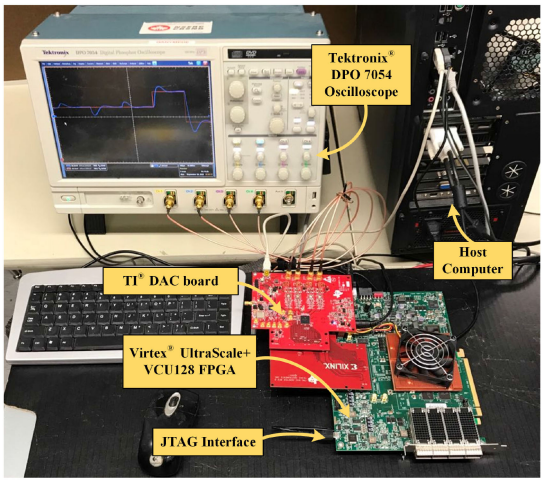

FIGURE 9. Hardware connection of the real-time ATA emulation system.

TABLE 2. Hardware Resource Consumption for ML-Models in MEA

Device	BRAM	DSP	FF	LUT	Latency
Each IGBT	4.27%	4.43%	0.33%	2.63%	370 ns
Generator	4.27%	2.16%	0.21%	1.50%	520 ns
Each Transf.	4.27%	4.75%	0.28%	2.29%	550 ns
Each Rectifier	4.27%	3.58%	0.31%	2.66%	640 ns
Each Converter	4.27%	5.95%	0.34%	3.02%	640 ns
Each PMSM	4.27%	7.19%	0.36%	3.19%	810 ns
Total Util.	55.51%	68.24%	2.76%	37.05%	820 ns
Available	4032	9024	2607360	1303680	

test datasets that differ from the training datasets. Although the NN parameters design, complexity of modeling objects, epoch number, the training strategy, and other factors can influence the MAEs during training, all the CRNN models in this paper are built within 1% MAE after 100 epochs. In order to test their generalization ability, these models are placed in the traditional simulation system to run various evaluation conditions. Only if the models pass all these tests, will they be applied to build a system block by block on FPGA.

D. HARDWARE PLATFORM

The MLBB models are rebuilt by C language in Xilinx HLS, which transfers the C functions into intellectual property (IP) cores for parallel execution. Then, IP cores are applied in Xilinx Vivado, and the bitstream file is generated and downloaded into the HBM FPGA based Xilinx VCU128 board. The hardware connection is shown in Fig. 9. The XCVU37P FPGA runs at 100 MHz, and has the following resources: 4,032 K block random-access memories (BRAMs), 9,024 digital signal processors (DSPs), 2,607,360 flip flops (FFs), and 1,303,680 lookup tables (LUTs). Furthermore, the utilization of total ML models is 2,238 BRAMs (56%), 6,158 DSPs (68%), 71,963 FFs (3%), and 483,013 LUTs (37%). The hardware resource consumption is shown in Table 2. Particularly, the reason for all the models utilizing 4.27% BRAM is that their unrolled factors for $tanh()$ look-up tables are designed

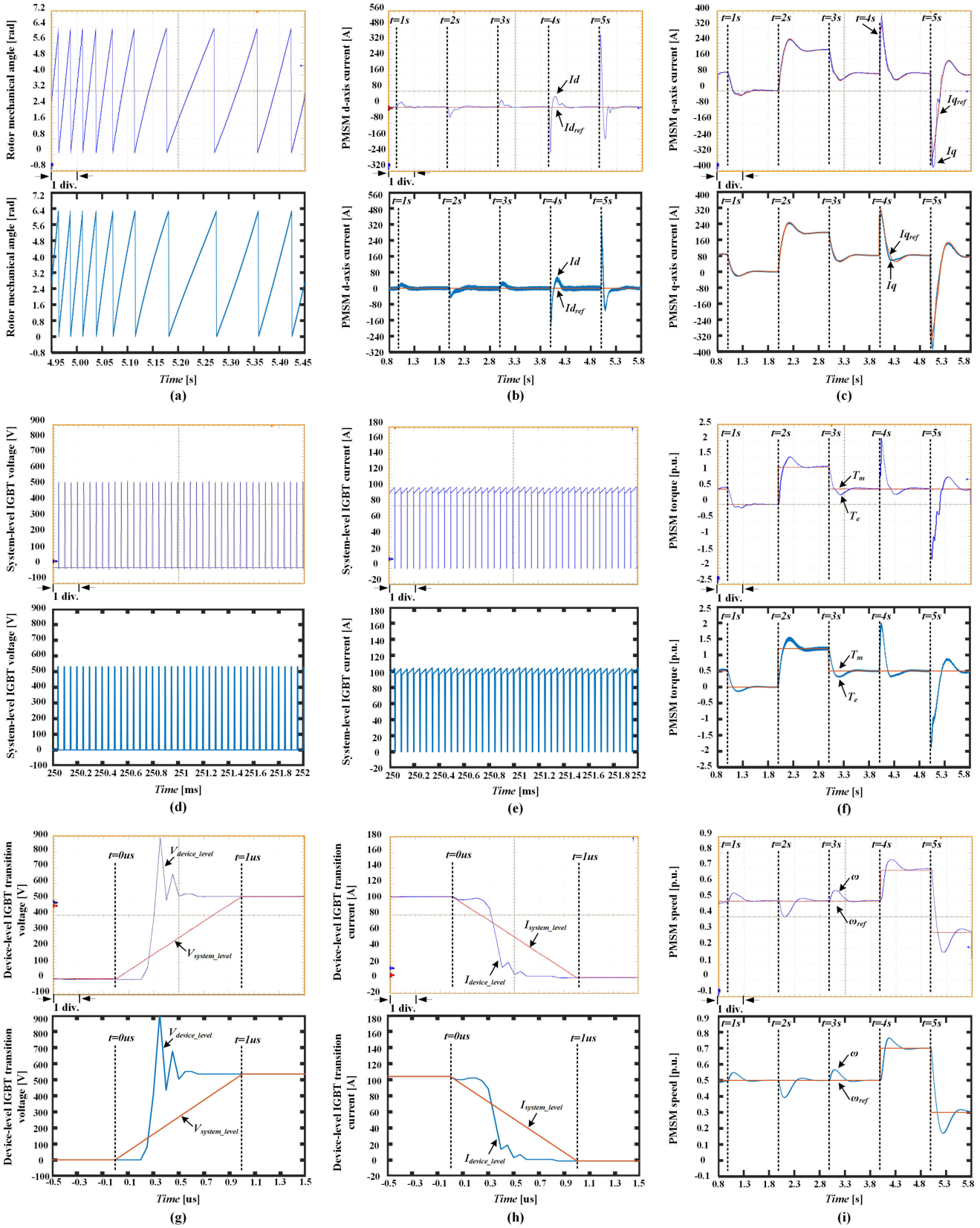

FIGURE 10. System-level and device-level hybrid models' results for the MEA system from real-time emulation (top oscilloscope sub-figure) and off-line simulation by PSCAD/EMTDC or SaberRD software (bottom sub-figure) for: (a) PMSM rotor mechanical angle; (b) PMSM *d*-axis current; (c) PMSM *q*-axis current; (d) system-level IGBT ML model output voltage; (e) system-level IGBT ML model output current; (f) PMSM torque; (g) device-level SiC IGBT ML model output voltage; (h) device-level SiC IGBT ML model output current; and (i) PMSM rotor speed. Scale: (a) x-axis: 50 *ms*/div. (b) (c) (f) (i) x-axis: 500 *ms*/div. (d) (e) x-axis: 200 μ*s*/div. (g) (h) x-axis: 200 *ns*/div.

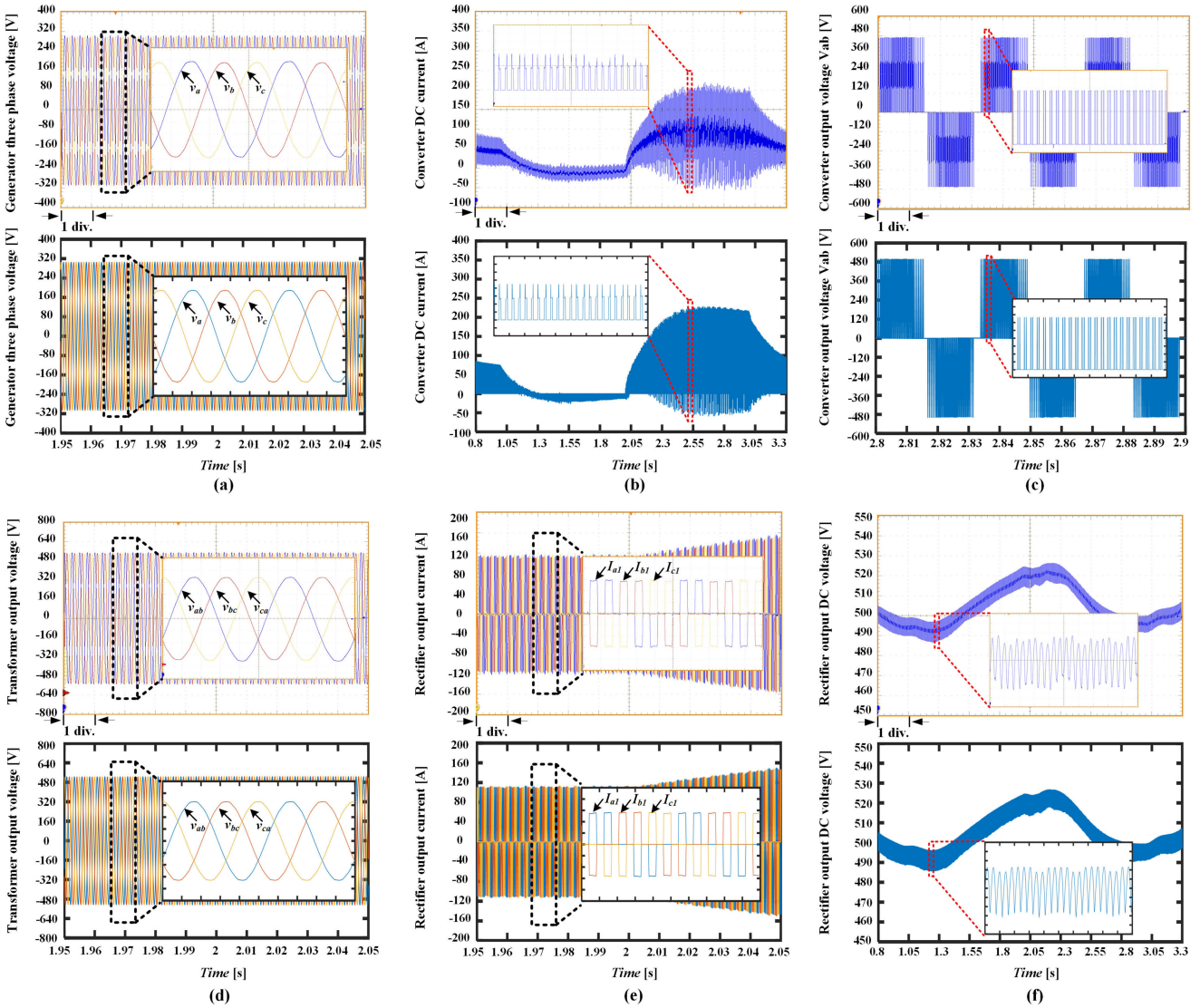

FIGURE 11. System-level models' results for MEA system from real-time emulation (top oscilloscope sub-figure) and off-line simulation by PSCAD/EMTDC software (bottom sub-figure) for: (a) generator output voltage; (b) converter output current; (c) converter output voltage; (d) transformer output voltage; (e) rectifier output current; and (f) rectifier output voltage. Scale: (a) (c) (d) (e) x-axis: 10 *ms*/div. (b) (f) x-axis: 250 *ms*/div.

the same. All the models can be processed within 1 μs. Among these models, the longest latency is 0.81 μs in PMSM ML model, and it also consumes the most resource: 172 K BRAMs (4%), 649 DSPs (7%), 9,498 FFs (0.4%), and 41,652 LUTs (3%). The whole emulation runs in real-time with 1 μs time-step at system-level and 50 *ns* time-step at device-level on the FPGA, while the same system in PSCAD/EMTDC takes about five minutes execution time for each second of MEA simulation on a PC equipment with 16 *GB* RAM and a 4-cores 3.4 GHz CPU.

V. RESULTS AND DISCUSSION

This section compares the results of the proposed MLBB based emulation on FPGA with the traditional transient methods, which utilizes PSCAD/EMTDC for system-level simulation and SaberRD for device-level simulation. All the ML models are working in the same condition with the same control. In this system, the PMSM changes its speed and mechanical load at 1 *s*, 2 *s*, 3 *s*, 4 *s*, and 5 *s*, respectively; other equipment is expected to keep stable outputs related to the PMSM's changes. The results of PMSM hybrid ML models are shown in Fig. 10 where (a), (b), (c), (f), and (i) are the rotor mechanical angle, *d*-axis current, *q*-axis current, torque, and speed of PMSM, respectively. The PMSM mechanical torque, as its load, changes to 0 p.u. at 1 *s*, 0.8 p.u. at 2 *s*, and 0.5 p.u. at 3 *s* while the PMSM's referenced speed raises from 0.5 p.u. to 0.7 p.u. at 4 *s* and drops into 0.3 p.u. at 5 *s*. The MLBB-based MEA system emulates the changing load and stable speed of the PMSM during 1-3 *s* to verify the hybrid ML model's dynamic effect. Clear support for the accuracy of the hybrid ML model is found from the results of the dq-axis current, torque, and speed of the PMSM in Fig. 10(b), (c), (f), and (i). Then, the stable load and changing speed test is given from 4 *s* to 5.8 *s*, which also shows excellent performance of

PMSM hybrid ML models. When the PMSM changes speed, another important variable, the rotor mechanical angle, shows the slow frequency shifting of rotation in Fig. 10(a). And the hybrid ML model's output matches that of the traditional model, which reiterates the accuracy of the hybrid ML model.

Fig. 11 shows the single CRNN models' results for each equipment in the system. The generator's output voltage is 300 V, and its frequency is 400 Hz, shown in Fig. 11(a). Then, the output current and single-phase voltage of the converter is shown in Fig. 11(b) and (c). The current value changes because of the changing speed and load of the PMSM. A similar pattern of results is obtained in Fig. 11(c), which demonstrates that the CRNN converter works well in the same system as the traditional model. The result of the transformer output voltage is displayed in Fig. 11(d) and its line to line output voltage amplitude is about 550 V. The output current and voltage of the rectifier are given in Fig. 11(e) and (f), respectively. In Fig. 11(f), the voltage floats around 520 V during the running period because of the changeable load. All the results show the system-level CRNN models have almost the same performance as those from PSCAD/EMTDC.

When it comes to the difference between system-level and device-level ML models, it is demonstrated in Fig. 10(g) and (h), which are the SiC IGBT's turn off transient in Fig. 10(d) and (e). These results are obtained from the SiC IGBT device-level hybrid model in Fig. 2(b). To make a comparison, the inside steady-state unit and the whole SiC IGBT device-level hybrid model are output as two channels. The intervals are 50 *ns* and 1 μs for the device-level ANN transient model and system-level CRNN steady-state model, respectively. In Fig. 10(g) and (h), the red curves represent the outputs from the system-level model that also works as the steady-state unit in the IGBT hybrid model, while the blue curves are the outputs from the SiC IGBT device-level hybrid model. The system-level output voltage and current jump from two values linearly while the device-level outputs show the nonlinear transient.

VI. CONCLUSION

This paper proposed MLBB-based modeling method to emulate the transients of ATA with high accuracy and execution efficiency from component-level (50 *ns* time-step), device-level (50 *ns* time-step), and system-level (1 μs time-step) on the FPGA platform in real-time. Finally, different level models' accuracy are verified and compared with the offline results obtained from PSCAD/EMTDC (system-level) and SaberRD (device-level) tools. The proposed method has these advantages: 1) High-execution efficiency: traditional methods have matrix solver, whereas there is no matrix inversion in MLBBs, which significantly reduces model latency and the computational complexity for each execution step; while compared with traditional iteration algorithm for nonlinear models, ML algorithm causes less execution delay for the nonlinear processes; NNs are perfectly suitable for parallel execution by FPGAs. 2) Flexible modeling: although the devices vary from the system, they can be modeled in a similar ML structure; the emulation system can be split into hierarchical execution units based on the user's specifications; ML models can be built by the external characteristics of running devices, while traditional modeling methods need to stop the devices and test internal characteristics. 3) High-accuracy: the errors between the MLBB outputs and the original datasets for all the models in the ATA are less than 1%. The above benefits make MLBBs significantly versatile, remarkably flexible, and highly executable. Based on the MLBB approach, future research will focus on real-time multi-domain modeling and emulation of ATAs.

TABLE 3. Parameters of PMSM Drive System and ESS on MEA

PMSM	
Nominal apparent power	60 kVA
Nominal voltage	0.3 kV
Rated frequency	60 Hz
Stator winding resistance	0.021 p.u.
Stator leakage reactance	0.064 p.u.
d-axis and q-axis inductance	0.689 p.u.
d-axis, q-axis damper winding resistance	0.055 p.u., 0.183 p.u.
d-axis, q-axis damper reactance	0.62 p.u., 1.175 p.u.
Permanent magnet strength	1.0 p.u.
DC bus	
DC bus voltage	540 V
DC bus capacitor	1 mF
SiC IGBT	
Emitter, collector inductance L_E, L_C	10 nH, 20 nH
Parasitic inductance L_{PCE}	30 nH
Total capacitive charge Q_c	2.5 μC
Peak reverse recovery current I_{rr}	100 A

APPENDIX

The parameters of the PMSM drive system and the ESS on MEA are shown in Table 3.

REFERENCES

[1] P. Wheeler and S. Bozhko, "The more electric aircraft: Technology and challenges," *IEEE Electrific. Mag.*, vol. 2, no. 4, pp. 6–12, Dec. 2014.

[2] B. Sarlioglu and C. T. Morris, "More electric aircraft: Review, challenges, and opportunities for commercial transport aircraft," *IEEE Trans. Transp. Electrific.*, vol. 1, no. 1, pp. 54–64, Jun. 2015.

[3] B. Zahedi and L. E. Norum, "Modeling and simulation of all-electric ships with low-voltage DC hybrid power systems," *IEEE Trans. Power Electron.*, vol. 28, no. 10, pp. 4525–4537, Oct. 2013.

[4] G. Sulligoi, A. Vicenzutti, and R. Menis, "All-electric ship design: From electrical propulsion to integrated electrical and electronic power systems," *IEEE Trans. Transp. Electrif.*, vol. 2, no. 4, pp. 507–521, Dec. 2016.

[5] T. Liang and V. Dinavahi, "Real-time device-level simulation of MMC-based MVDC traction power system on MPSoC," *IEEE Trans. Transp. Electrific.*, vol. 4, no. 2, pp. 626–641, Jun. 2018.

[6] D. Miyashita, S. Kousai, T. Suzuki, and J. Deguchi, "A neuromorphic chip optimized for deep learning and CMOS technology with time-domain analog and digital mixed-signal processing," *IEEE J. Solid-State Circuits*, vol. 52, no. 10, pp. 2679–2689, Oct. 2017.

[7] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[8] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, 2014, pp. 1701–1708.
[9] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.
[10] R. Sarikaya, G. E. Hinton, and A. Deoras, "Application of deep belief networks for natural language understanding," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 4, pp. 778–784, Apr. 2014.
[11] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. IEEE Int.*, Prague, 2011, pp. 5528–5531.
[12] Y. Xu, J. Du, L. Dai, and C. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 1, pp. 7–19, Jan. 2015.
[13] D. F. Specht, "A general regression neural network," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[14] K. Y. Lee, Y. T. Cha, and J. H. Park, "Short-term load forecasting using an artificial neural network," *IEEE Trans. Power Syst.*, vol. 7, no. 1, pp. 124–132, Feb. 1992.
[15] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.
[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[17] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," Dec. 2014, *arXiv:1412.3555* [cs].
[18] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[19] D. Zhang and M. R. Kabuka, "Combining weather condition data to predict traffic flow: A GRU-based deep learning approach," *IET Intell. Transp. Syst.*, vol. 12, no. 7, pp. 578–585, Sep. 2018.
[20] S. Zhang, Y. Wu, T. Che, Z. Lin, and R. Memisevic, "Architectural complexity measures of recurrent neural networks," in *Proc. Adv. Neural Inf. Syst.*, 2016, pp. 1822–1830.
[21] K. Xu, N. Xie, C. Wang, and X. Shi, "Modeling and simulation of variable speed variable frequency electrical power system in more electric aircraft," *Open Elect. Electron. Eng. J.*, vol. 11, pp. 87–98, 2017.
[22] E. Ganev, "Selecting the best electric machines for electrical power generation systems: High-performance solutions for aerospace more electric architectures," *IEEE Electrific. Mag.*, vol. 2, no. 4, pp. 13–22, Dec. 2014.
[23] Z. Huang and V. Dinavahi, "An efficient hierarchical zonal method for large-scale circuit simulation and its real-time application on more electric aircraft microgrid," *IEEE Trans. Ind. Electron.*, vol. 66, no. 7, pp. 5778–5786, Jul. 2019.
[24] M. Sinnett, "787 no-bleed systems: Saving fuel and enhancing operational efficiencies," *Aeromagazine*, vol. 4, pp. 6–11, 2007.
[25] H. W. Dommel, "Digital computer solution of electromagnetic transients in single-and multiphase networks," *IEEE Trans. Power Appar. Syst. vol. PAS-* 88, no. 4, pp. 388–399, Apr. 1969.
[26] S. Ruder, "Overview of gradient descent optimization algorithms," Sep. 2016, *arXiv:1609.04747* [cs].
[27] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–15.

SONGYANG ZHANG (Student Member) received both the B.Eng. and M.Eng. degrees in electrical engineering from the Huazhong University of Science and Technology, Wuhan, Hubei, China, in 2017 and 2019, respectively. He is currently working toward the Ph.D. degree in electrical and computer engineering with the University of Alberta, Edmonton, Alberta, Canada. His research interests include machine learning, real-time simulation, power electronics and field programmable gate arrays.

TIAN LIANG (Member, IEEE) received the B.Eng. degree in electrical engineering from Nanjing Normal University, Nanjing, Jiangsu, China, in 2011, the M.Eng. degree from Tsinghua University, Beijing, China, in 2014, the Ph.D. degree in energy systems from the University of Alberta, Edmonton, AB, Canada, in 2020. His research interests include real-time simulation of power systems, power electronics, artificial intelligence, field-programmable gate arrays, and system on chip.

VENKATA DINAVAHI (Fellow, IEEE) received the B.Eng. degree in electrical engineering from the Visveswaraya National Institute of Technology , Nagpur, India, in 1993, the M.Tech. degree in electrical engineering from the Indian Institute of Technology (IIT) Kanpur, India, in 1996, and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Ontario, Canada, in 2000. Presently, he is a Professor with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. His research interests include real-time simulation of power systems and power electronic systems, electromagnetic transients, device-level modeling, large-scale systems, and parallel and distributed computing.