Latency Insertion Method Based Real-Time Simulation of Power Electronic Systems

Matthew Milton and Andrea Benigni, *Member, IEEE*

***Abstract*—In this paper, we demonstrate how latency insertion method (LIM) can be used for real-time simulation of high switching frequency power electronics systems and how this approach can be implemented for scalable Field Programmable Gate Array (FPGA) execution. We first present a summary of the LIM and how this method can be developed into a solver for high-performance FPGA execution. We then present how common power electronics topologies—buck and boost—can be modeled using the LIM approach, followed by how the LIM model of a three-phase dc/ac converter is created. Afterward, we demonstrate the accuracy of the developed FPGA solver through a set of power electronic system examples, with each example compared with near ideal results of same example provided by a traditional electromagnetic transient type solver. We complete the paper analyzing the scalability of the proposed approach on FPGA devices, both in terms of achievable time step as well as of resource usage.**

***Index Terms*—Circuit simulation, parallel algorithms, power electronics, power system simulation, real-time systems.**

I. Introduction

REAL-TIME simulation along with Hardware-In-the-Loop (HIL) and Power Hardware-In-the-Loop (PHIL) techniques have been widely used in recent years for the testing of single power electronics converters as well as system of converters. The main advantage of real-time simulation and HIL techniques over traditional simulation is that they allow designers to perform experiments where part of the system is emulated on computational units (simulated) while part of the system is physically installed in the laboratory, thereby closing the gap between laboratory and field tests. In [1]–[5], examples of how HIL and PHIL have been used to support a heterogeneous set of research questions in the power electronics area are presented.

Real-time simulation of power electronics systems is particularly critical due to the nonlinear and discontinuous characteristics of the system under test. For this reason several methodologies have been developed to speedup simulation of power electronic systems to accurately capture these characteristics. In [6], companion models of switching devices are created so as to obtain a linear time invariant conductance matrix. Within [7]–[9] the system of interest is partitioned so as to allow parallel execution and to limit the increase of computational effort as size of system grows. All approaches presented in [7]–[9] take also advantage of multirate execution to further reduce the computational cost.

With the recent development of silicon carbide (SiC) power modules, we expect the development of power electronics converters with switching frequency reaching 100–200 kHz. This high switching frequency challenges the existing methodologies and commercial tools for real-time simulation. To allow real-time simulation with significantly small time step to model converters at these switching frequencies, there has been a change in recent years in the type of processors used in this application. For instance, mixed solution implementations based on digital signal processors (DSPs)/CPUs and FPGAs are now common and standalone FPGAs are used both as interface and for computation in this regard [10]–[14].

To enable the real-time simulation of large power electronic systems at the ever increasing switching frequencies, the use of highly parallelizable and scalable simulation methods is required. One such method is latency insertion method (LIM), traditionally used for the simulation of large passive transmission networks in semiconductor devices. In this paper, we demonstrate a novel way to adapt LIM for the real-time simulation of switching power electronics systems and how this approach can be implemented on Field Programmable Gate Array (FPGA) devices. Due to FPGA execution and the method's high parallelizability a very small time step of 40 ns is achieved for real-time simulation of systems of significant size.

We start the paper by providing an overview of the LIM and describing how it can be implemented on an FPGA platform. Then, we proceed in showing how common power electronics topologies—buck and boost—can be modeled using the LIM approach and how a three-phase dc/ac converter model can be created as well. We continue through demonstrating the accuracy of the developed solver through a set of examples and through comparison with a traditional electromagnetic transient (EMTP) type solver acting as an approximate analytical solution of each example. We complete the paper analyzing the scalability—both in terms of achievable time step in real time as well as of resource usage—of the proposed approach on an FPGA device.

II. LIM on FPGA

LIM is a finite difference method for transient circuit simulation, first defined in [15]. We proceed here with a brief

Manuscript received April 26, 2017; revised August 3, 2017; accepted September 19, 2017. Date of publication September 26, 2017; date of current version April 20, 2018. This work was supported in part by the Office of Naval Research under Grant NOOO14-16-1-3042. Recommended for publication by Associate Editor G. Oriti. *(Corresponding author: Andrea Benigni.)*

The authors are with the Department of Electrical Engineering, University of South Carolina, Columbia, SC 29078 USA (e-mail: mmilton@email.sc.edu; benignia@cec.sc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TPEL.2017.2757449

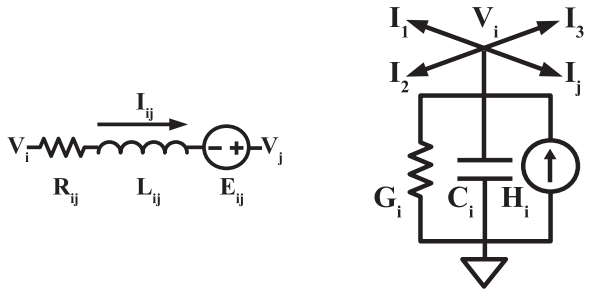

Fig. 1. (a) Generic LIM branch. (b) Generic LIM node.

description of the LIM and its existing stability analysis. This summary is followed by discussion of encapsulation of node and branch models into FPGA logical entities for FPGA execution, and how an FPGA solver is created with these entities. Finally, considerations are presented for implementing the solver to choose between achievable time step and resource usage.

A. LIM Review

A network is solvable using LIM if each branch of the network contains an inductance and each node of the network provides a capacitive path to ground; if these values are not naturally present or if the present values are small and, thus, present latency much smaller than the time features of interest, additional capacitance or inductance can be added to the network to increase latency (and hence the allowable time step). We avoid this situation in this paper since it implies a modification of the system to be simulated and may raise doubts on the accuracy of the method.

In Fig. 1, the required topologies for LIM-compatible branch and node are depicted. As shown in Fig. 1(a), any generic branch is composed of a series combination of a resistance, an inductance, and a voltage source; applying Kirchoff's Voltage Law (KVL) to the circuit of Fig. 1(a), we can write the characteristic branch equation as

$$V_i^{n+\frac{1}{2}} - V_j^{n+\frac{1}{2}} = L_{\mathrm{ij}} \left(\frac{I_{\mathrm{ij}}^{n+1} - I_{\mathrm{ij}}^{n}}{\Delta t} \right) + R_{\mathrm{ij}} I_{\mathrm{ij}}^{n} - E_{\mathrm{ij}}^{n+\frac{1}{2}}. \quad (1)$$

From (1) it is possible to calculate the unknown branch current

$$I_{\mathrm{ij}}^{n+1} = I_{\mathrm{ij}}^{n} + \frac{\Delta t}{L_{\mathrm{ij}}} \left(V_i^{n+\frac{1}{2}} - V_j^{n+\frac{1}{2}} - R_{\mathrm{ij}} I_{\mathrm{ij}}^{n} + E_{\mathrm{ij}}^{n+\frac{1}{2}} \right). \quad (2)$$

Equation (2) must be computed for all the branches of the network at each time step.

As shown in Fig. 1(b), each generic node is connected to ground via a parallel combination of a conductance, a capacitance, and a current source; applying Kirchoff's Current Law (KCL) to the circuit of Fig. 1(b) we can write the characteristic node equation as shown in the following equation:

$$C_i \left(\frac{V_i^{n+\frac{1}{2}} - V_i^{n-\frac{1}{2}}}{\Delta t} \right) + G_i V_i^{n+\frac{1}{2}} - H_i^{n} = -\sum_{j=1}^{M_i} I_{\mathrm{ik}}^{n} \quad (3)$$

where M_i is the number of branches connected to the node i.

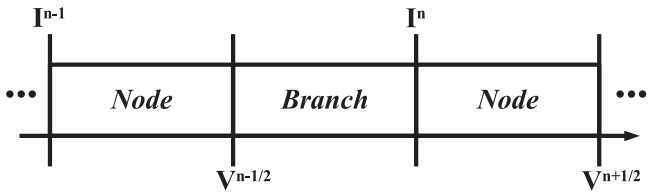

Fig. 2. LIM time base.

From (3) it is possible to calculate the unknown node voltage

$$V_i^{n+\frac{1}{2}} = \frac{\frac{C_i V_i^{n-\frac{1}{2}}}{\Delta t} + H_i^{n} - \sum_{j=1}^{M_i} I_{\mathrm{ik}}^{n}}{\frac{C_i}{\Delta t} + G_i}. \quad (4)$$

Equation (4) is computed for all the nodes of the network at each time step. Using a leap-frog approach, the current through each branch and the voltage at each node can be updated alternately. The time is discretized and the branch current and node voltage quantity updates are allocated in half time steps (see Fig. 2).

If the circuit of interest has the characteristics required by LIM, it is possible to solve the networks through a set of algebraic steps. As a consequence of LIM having linear computational complexity, the method strongly reduces the computational effort required for the simulation of the network. Moreover, each algebraic equation can be potentially solved in parallel, ensuring perfect scalability.

As can be noticed in (2) and (4), the LIM applies a semi-implicit leap-frog integration method, as originally used in the LIM when first proposed [15]. As a consequence, the stability of the method is a concern as the choice of time step for stable operation is conditional to the inductance and capacitance found in a given model under LIM. The stability of basic LIM and of its variations has been analyzed in many papers [15]–[21]. In [15], the stability of LIM using leap-frog integration is determined using Telegrapher's equations for a per-cell basis (one branch and one node connected together in a model), the stability condition being

$$\Delta t < \sqrt{L_k C_k} \quad (5)$$

where k is the index of the cell with the smallest latency derived from L_{k} and C_{k} in a given model. In [16] and [17], further detailed stability analysis of LIM was explored for RLC and GLC topologies using Lyapunov's stability theorem, though the general case of RLCG topologies was not studied. Stability analysis of the use of explicit, semi-implicit, and implicit integration methods in LIM was presented in [18]. The stability of LIM was analyzed in [19] using block-matrix Formulation, determining the stability of a system for a given time step using the eigenvalues of a computed amplification matrix for said system. A general stability criteria is given in [20] for partitioned LIM, which highlights stability for models where branches and nodes contain dependent sources through which components are interconnected. In [21], an approach is proposed and analyzed for improving stability of LIM to be unconditional to latency or time step through introducing a single formulation to update

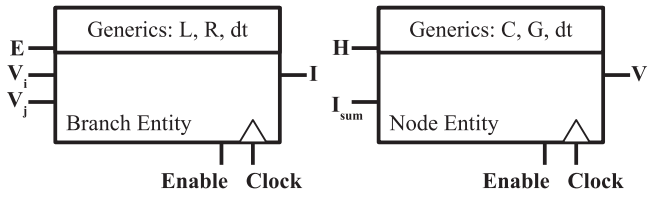

Fig. 3. FPGA RTL entities for LIM models.

branches and nodes, performed through implicitly substituting branch currents and node voltages into said formulation.

Despite the conditional stability of leap-frog integration in LIM, this integration structure provides benefits to modeling high frequency switching power electronic converters. This semi-implicit integration structure separates branch current and node voltage updates into computationally cheap explicit difference half time step terms (similar to Euler forward) which can be exploited to model converter switching behavior in a single time step without the need to use a fully implicit approach on entire converter model; while still maintaining second-order numerical integration accuracy provided by leap-frog integration. By having the current and voltage terms integrated separately, this structure allows representing ideal switching phenomena by linking nodes and branches through their respective ideal voltage and current sources (*E*, *H*), without introducing any additional delay in the integration; more details on the linkage found in Section III. Moreover, this linking maintains full compatibility with the LIM formulation and stability analysis presented in [20], where voltage and current sources can depend on other quantities in the system. Similar advantages for modeling power electronic converters in LIM can be found in applying fully explicit integration methods (Euler forward, Runge–Kutta, and others) instead of leap-frog method, keeping branch and node updates explicit and separate, with the potential to update branch currents and node voltages simultaneously in single time step without sequentializing these updates in leap-frog manner. However, the use of other integration methods will have different tradeoffs in stability and computational cost, depending on the method applied.

B. LIM FPGA Encapsulation

LIM maps well to FPGA architecture due to the high parallelizability of branch and node models, and to the natural expression of the model equations as difference equations which align with the discrete hardware of FPGA devices. To implement LIM-modeled networks in an FPGA design, a logical entity is created for both the branch model and the node model (see Fig. 3). For the branch model, its entity takes as input signals node voltages $V_i^{n+1/2}$ and $V_j^{n+1/2}$, and dependent voltage source $E_{\mathrm{ij}}^{n+\ 1/2}$, and outputs the branch current I_{ij}^{n+1}. The entity for the node model takes as input the branch current sum $\sum_{j=1}^{M_i} I_{\mathrm{ik}}^n$ (as single signal) and dependent current source H_i^n, and outputs node voltage $V_i^{n+1/2}$. Parameterizing these entities for particular circuit branches and nodes (setting *L, R, C*, and G), the parameters of the entities can be set through generics which configure them during FPGA synthesis of the design of simulated model. In each LIM entity design, their respective model equations (2) and (4) are implemented as signed fixed-point combinational arithmetic logic synthesized from the difference forms of said equations. These equations are updated once every full time step in a dataflow manner, using registers to retain past time step states (I_{ij}^n, $V_i^{n-1/2}$) and entity output for next half time step entity input (I_{ij}^{n+1}, $V_i^{n+1/2}$).

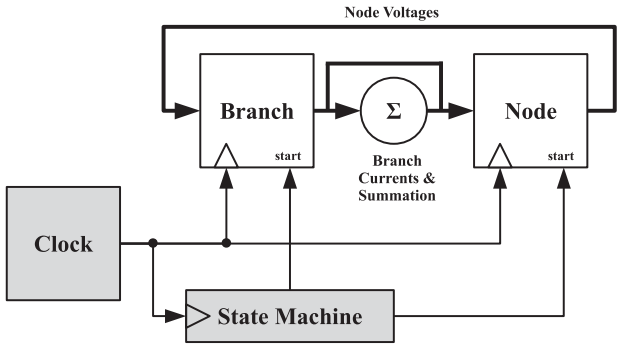

Fig. 4. LIM simulation engine FPGA design.

C. Simulation Engine Composition

To compose a simulation solver engine FPGA design that will simulate an LIM-modeled network, branch entity output current signals are connected to input signal ports of the node entities, and output voltage signals are connected to input ports of the branch entities, corresponding to topology of modeled circuit in question. If multiple branch currents are to feed into a node, they are summed together and the result is given to the corresponding node. Driving the updates of the LIM model entities, a digital clock is fed into all entities which clocks the internal state and output registers. Since a time step is divided into two halves, one for branches and the other for nodes, the time step clock has a period half that of the desired time step length (twice as fast) to have one clock period per half time step. So that branches and nodes are updated in leap-frog fashion, a simple finite state machine of two states is used to schedule when branches and nodes can update through Start (Enable) signals for each. A scheme of the simulation engine composition is reported in Fig. 4.

The above structuring of the simulation engine is designed for leap-frog integration method. This method allows for second-order numerical stability and simple computations to increase execution speed, but still requires two clock periods to compute a total system solution to update branch and node components in leap-frog manner. To achieve system solution computation in a single clock period, potentially lowering simulation time step by half, Euler forward (or any other explicit method for that matter) could be used as an alternative integration method, at the potential cost of numerical instability or inaccuracy. Using Euler forward method, the branch and node component difference equations (2) and (4) would not change in implementation but both types of components will be updated simultaneously rather than in leap-frog approach, using solutions of opposite

components from past full time step rather than past half time step.

D. Simulation Engine Execution Scheduling and Resource Usage

How FPGA resources (logic slices, DSPs, Look Up Tables (LUTs)) are assigned to the arithmetic logic of branch and node FPGA entities, and how the usage of said resources for the entities are scheduled, can dramatically influence the smallest achievable time step and largest size of model to be real-time simulated on a given FPGA, often with tradeoff between resources and time step. For smallest computational delay to reduce time steps, each branch and node entity in the simulation engine of a system model can be assigned their own dedicated FPGA resources so that entities of same type can be scheduled to execute in parallel to one another. However, using dedicated resources per entity sets an upper boundary on the size of a system modeled in LIM, as larger models will consume more resources as the number of entities increase and FPGAs of any size will have finite resources to use. In the case where an LIM model cannot fit on a given FPGA, resources can be shared between LIM entities, where each entity to share is scheduled to have access to a shared pool of resources via a finite state machine similar to that mentioned for the simulation engine composition in Section II-C. The downside to sharing resources is they cannot be practically shared within same clock cycle, so multiple clock cycles are required where each sharing entity is scheduled a clock cycle to access the resources. This scheduling causes the computational delay to increase from sequentializing arithmetic operations, ultimately setting an upper boundary on achievable time step in real-time execution that can be used to simulate high-frequency dynamics in system models as more entities share resources. Ideally, if a number N of LIM FPGA entities are sharing resources, the resource usage goes down by $1/N$ the original amount for these entities but the amount of computational time (or clock cycles) will increase by N; though actual reduction in resource usage will be smaller due to logic overhead of switching access of resources between entities. Generally, tradeoffs can be made between shared resource usage and computational delay to allow larger LIM FPGA models of systems to fit on a given FPGA while still maintaining acceptable time steps. However, the ideal design for an LIM simulation engine is to have the engine implemented to maximize execution parallelism through dedicated resource assignment for entities to reduce time steps, should a given FPGA provide sufficient resources to do so.

III. Modeling of Switching Power Converters Using LIM

In general, most circuits with sufficient reactive latency can be simulated with LIM after proper manipulation. This fact holds true for switching power converters which commonly contain capacitive and inductive elements. In such circuits, the arrangements of the latency components can align with LIM branch and node models, with the nonlinear switching elements handled by exploiting the E and H terms of the LIM models.

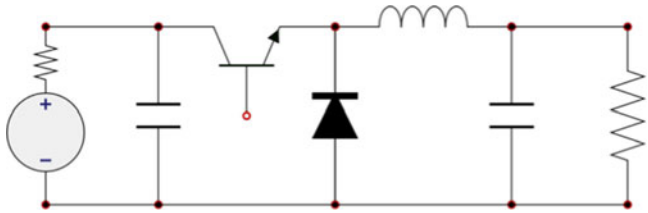

Fig. 5. Buck converter.

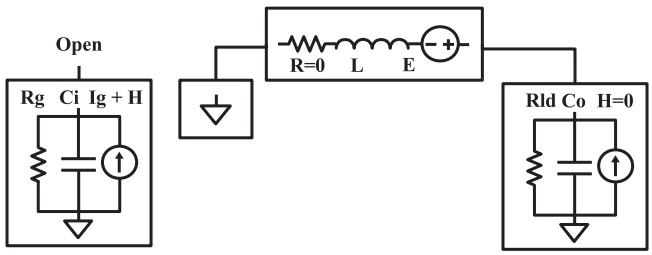

Fig. 6. Buck converter LIM model.

Application of LIM alignment to different model switching converters is demonstrated in the following examples for a buck topology, a boost topology, and a three-phase inverter. These simple converter examples, despite similarities, highlight how the LIM topology and switching behavior handling under LIM changes for each converter. For each topology, traditional switching power electronics analysis is used to determine how the switching modes are to be realized in LIM, with all switching elements (transistor, diode) modeled as ideal switches.

A. Buck Topology

Take for instance the ideal buck converter shown in Fig. 5. In the buck converter, the output filtering capacitor and load resistance are mappable to an LIM node, the inductor is mappable to an LIM branch, and the voltage source input and input capacitor are mappable to another LIM node after the voltage source has been transformed into a current source with Norton's transformation. The nonlinear switching elements of the converter are left out of the LIM topology. This mapping of the converter to LIM components is shown in Fig. 6.

The question arises on how to handle the nonlinear switching action of the buck converter with LIM without incorporating the switching elements in the LIM topology. As seen in previous discussion on LIM, branch and node models contain, respectively, a voltage source E and a current source H which can be arbitrarily altered during simulation. Using these model sources, the switching action can be handled by altering the values of H and E in LIM component models in sync with the switching states of this simulated converter, while keeping the LIM topology fixed regardless of the switching state; this is unlike typical state equation modeling of converters where the state model of the converter is changed within a state machine for each switching topology the converter is in. Let us first consider the continuous conduction mode using traditional power electronic system analysis. As shown in Fig. 7(a) and (b) during switch-on

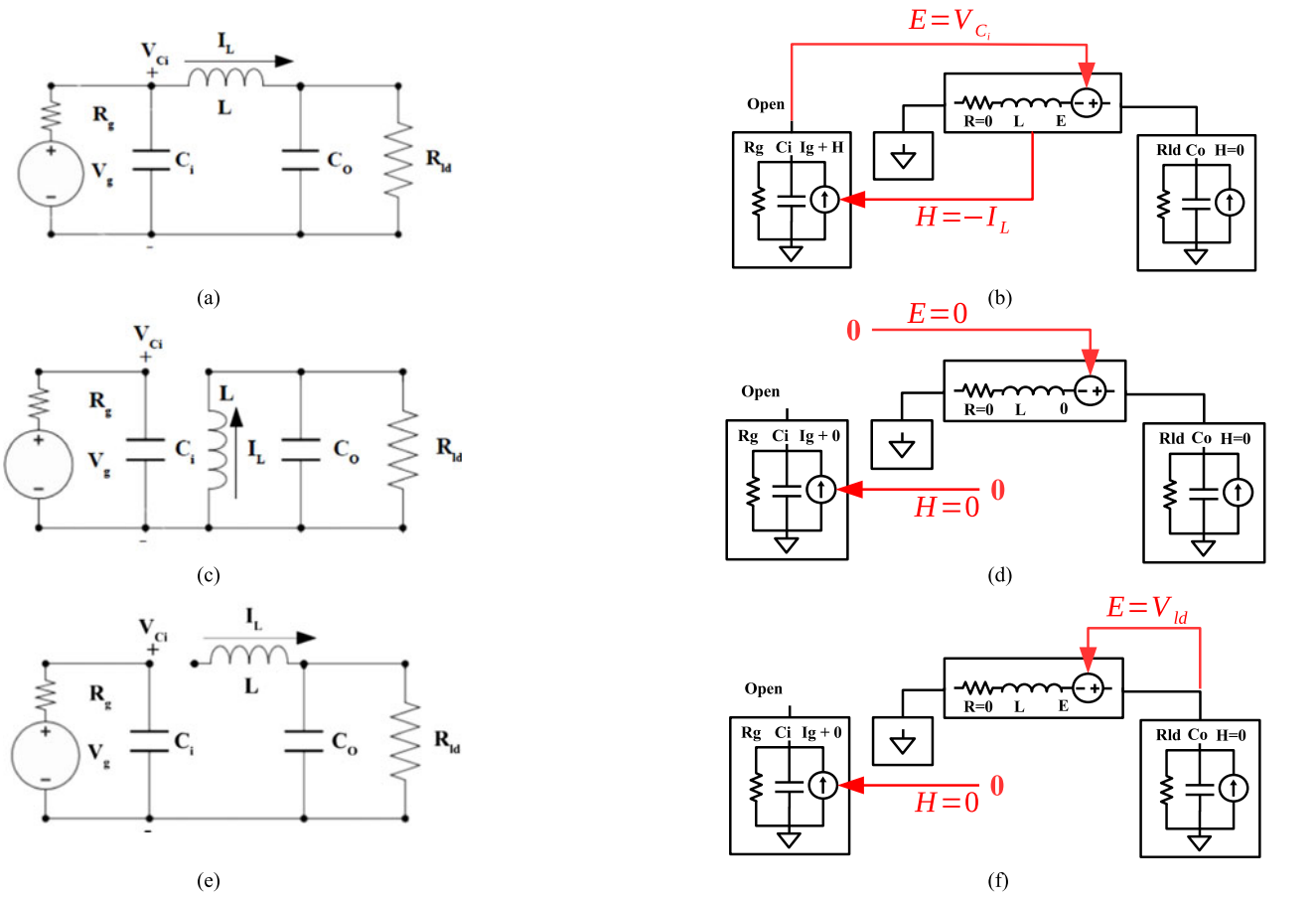

Fig. 7. Buck converter switching action with LIM Model. (a) and (b) the IGBT is conducting and the diode is in the open state. (c) and (d) the IGBT is in the open state and the diode is conducting. (e) and (f) the IGBT and the diode are in the open state.

state, the E term of the inductor branch is equated to the input voltage across the input capacitor node, and the H term of the input capacitor node is equated to the inductor branch current. During the off state of this converter, both H and E are set to zero, as shown in Fig. 7(c) and (d). By altering H and E terms according to a switching control signal in this fashion, the switching action of the converter using LIM is simulated. To also consider discontinuous conduction mode, the model monitors the current through the inductance during the OFF state; if the current crosses zero, the E term of the inductor branch equates to the output voltage, displayed in Fig. 7(f).

B. Boost Topology

Following a very similar approach to the one used for the buck converter, it is possible to create an LIM-compliant model for a boost topology converter. Let us take for instance the ideal boost converter shown in Fig. 8. In the boost converter, the output filtering capacitor and load resistance are mappable to an LIM node, the inductor is mappable to an LIM branch, and the voltage source input and input capacitor are mappable to another LIM node after the voltage source has been transformed into a current source with Norton's transformation. This mapping to LIM components is shown in Fig. 9, with the switching elements not inserted into the LIM topology.

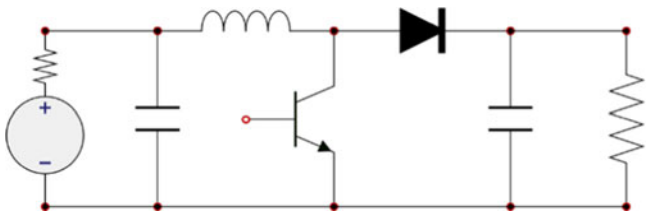

Fig. 8. Boost converter.

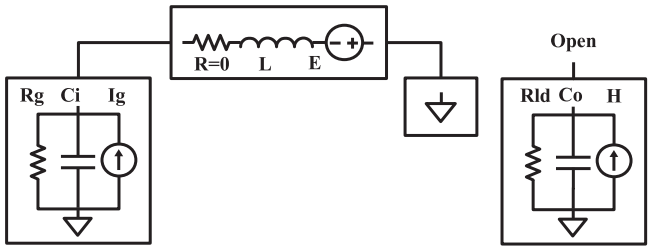

Fig. 9. Boost converter LIM model.

Following the same approach we have seen for the buck converter, we first consider the continuous conduction mode. During the switch-on state shown in Fig. 10(a), the E term of the inductor branch is equated to the output voltage changed of sign, the

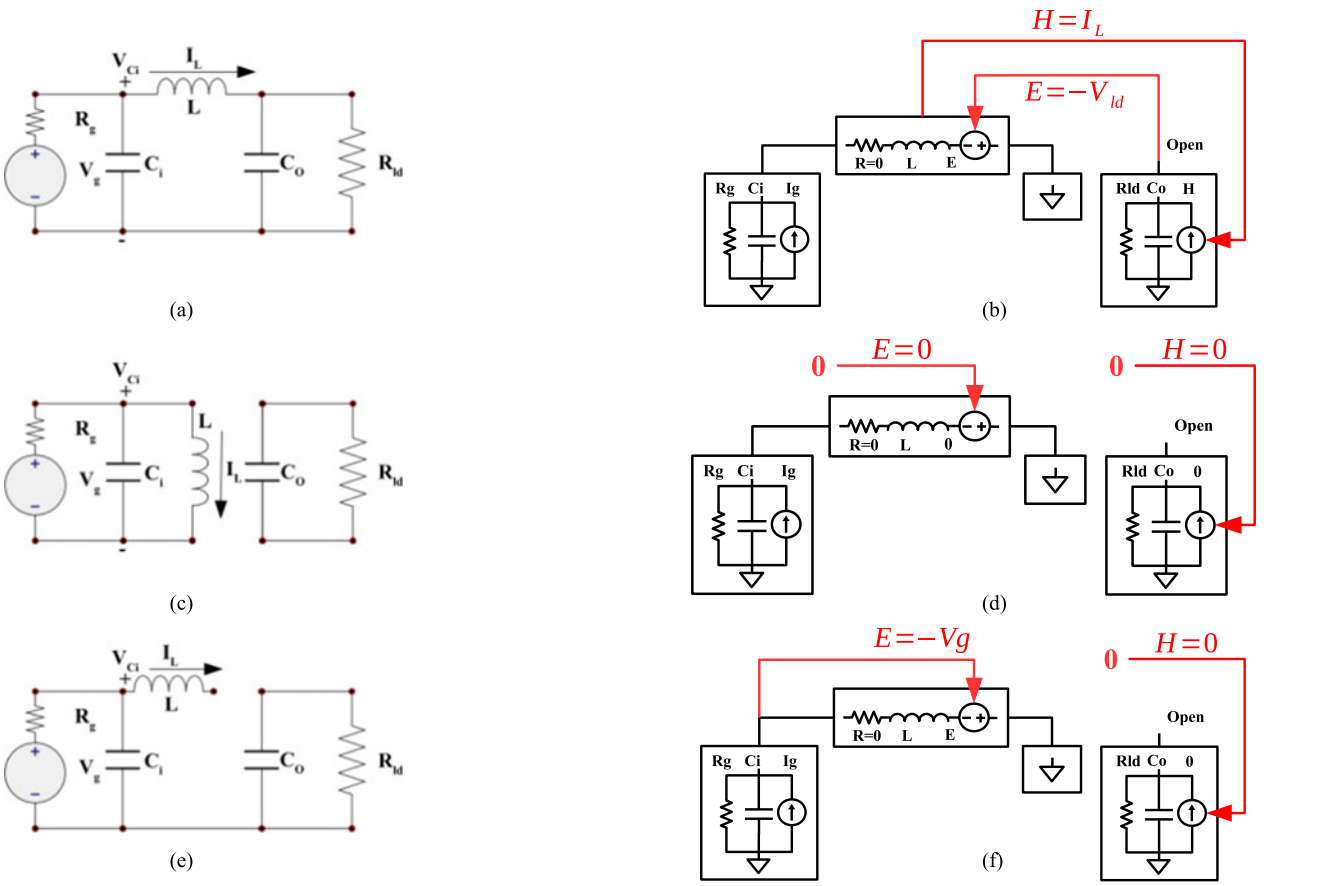

Fig. 10. Boost converter switching action with LIM Model. (a) and (b) the IGBT is conducting and the diode is in the open state. (c) and (d) the IGBT is in the open state and the diode is conducting. (e) and (f) the IGBT and the diode are in the open state.

H term of the output capacitor node is equated to the inductor current; depicted in Fig 10(b). During the off state of the converter depicted in Fig. 10(c), both *H* and *E* are set to zero in the LIM model shown in Fig. 10(d). To also consider discontinuous conduction mode, the model monitors the current through the inductance during the off state; if the current crosses zero the *E* term of the inductor branch is equated to the input voltage changed of sign, as shown in Fig. 10(f).

C. Three-Phase DC/AC Converter

A three-phase dc/ac converter can be modeled following an approach very similar to the one adopted for the buck converter. As can be seen from the converter's topology in Fig. 11(a), the inductors can be each mapped to LIM branch models, the dc bus capacitance can be modeled as LIM nodes, along with the output capacitor filter and resistive load. This mapping of the dc/ac converter to LIM is seen in Fig. 11(b). In this case, since the input voltage sources with series resistance is held as constant and has no capacitive or inductive elements, these sources can be treated as purely resistive branches without latency, modeled by

$$I_g^{n+1} = \frac{1}{R_g}\left(V_g - V_j^{n+\frac{1}{2}}\right) \tag{6}$$

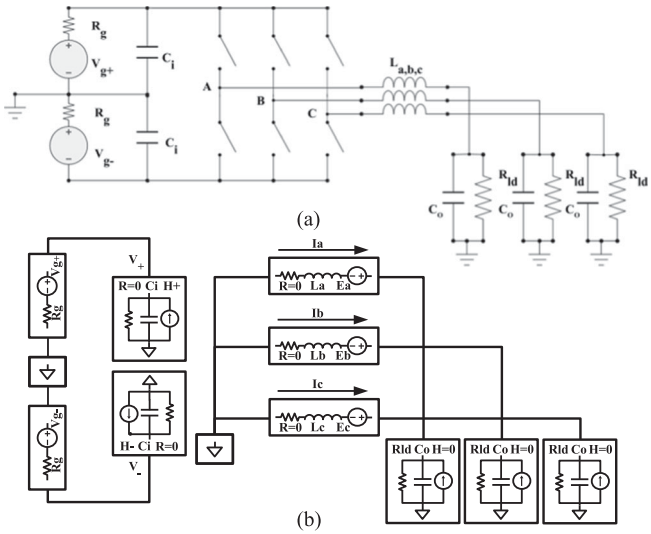

Fig. 11. (a) Three-phase dc/ac converter. (b) Three-phase dc/ac converter LIM model

which can be used as LIM-compatible branch current quantity. All switching elements are excluded from the LIM topology.

To handle the switching action of the three-phase converter, the *H* terms for the input node models can be set as functions

of the three inductor branch currents, such as in the following equations:

$$H_+ = -I_a s_a - I_b s_b - I_c s_c \tag{7}$$

$$H_- = -I_a \bar{s}_a - I_b \bar{s}_b - I_c \bar{s}_c \tag{8}$$

where s_a, s_b, s_c, and their inversions are the three-phase converter's modulating switch control signals per phases *a–c*, either of value zero or one. Then, for the *E* terms of the branch models, their functions can be declared as in the following equations:

$$E_a = V_+ s_a + V_- \bar{s}_a \tag{9}$$

$$E_b = V_+ s_b + V_- \bar{s}_b \tag{10}$$

$$E_c = V_+ s_c + V_- \bar{s}_c. \tag{11}$$

In these equations, V_+ and V_- are the voltages of the dc bus capacitance nodes, respectively. To model switching action with deadband interval, extra functions are applied for *H* and *E* when both switches are OFF (zero). In this case, the functions for the converter leg associated with phase a are reported in the following equations:

$$H'_+ = \begin{cases} 0, & I_a > 0 \\ +I_a, & I_a \leq 0 \end{cases} \tag{12}$$

$$H'_- = \begin{cases} 0, & I_a \leq 0 \\ -I_a, & I_a > 0 \end{cases} \tag{13}$$

$$E'_a = \begin{cases} V_-, & I_a > 0 \\ V_+, & I_a < 0 \\ V_a, & I_a = 0. \end{cases} \tag{14}$$

In these equations, V_a is the output phase voltage. The H'_+ from each converter leg are added to get H_+ for complete converter during deadband interval; the same for H'_-. These equations assume that the converter switches have ideal antiparallel diodes across them which conduct appropriately during deadband interval.

D. Converter Switching Behavior Handling on FPGA

Since *H* and *E* terms of branches and nodes of switching circuits are generally functions of branch currents and node voltages, switching behavior can be implemented by feeding results of functions of these current or voltage signals into a multiplexer whose output feeds into respective *H* or *E* input of an LIM entity, as shown in Fig. 12(a). Based on the switch state of the converter driven by a switch control signal, whether continuous or discontinuous mode, appropriate function can be selected via the multiplexer. Seen in Fig. 12(b), another way to handle switching behavior is to have branch current and node voltage signals switched ON or OFF as input to the *H* and *E* functions with multiplexers, based on the switching signal. In any case, the *H* and *E* functions are implemented as simple dataflow computational expressions which are expected to produce stable output within the half time step of entity that is recipient of said functions' results. Selection of the handling method is largely dependent on the converter model to be simulated with LIM on an FPGA device.

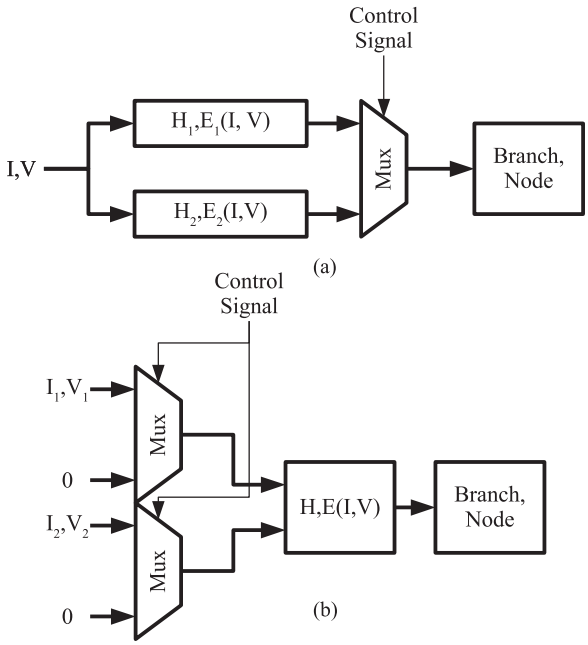

Fig. 12. Handling switching action in LIM simulation engine.

E. Converter Topology Encapsulation for Reusable Component Library

In general, any converter topology that is to be modeled in LIM must be manually analyzed through power electronic analysis to determine the LIM structure and *H*/*E* switching update equations of a given topology; and then be implemented in an FPGA design using the FPGA entities noted in Sections II-B and C for every instance that is within a power electronic system model. To avoid this tedious process for every instance of a given topology, the FPGA LIM entities and *H/E* switch update logic of a converter topology can be encapsulated into reusable component entities collectible into libraries. To allow these components to fit within an LIM simulation engine as described in Section II-C, the inputs and outputs of each branch and node FPGA entity of a component can be passed through the ports of the converter component entity, along with the control signals for the switch update logic. The use of LIM FPGA components for complete converters moves the task of developing LIM models of said converters from the simulation engine user to the model developer.

F. Limitations in Modeling Converters in Traditional LIM

As the LIM was originally developed for the simulation of semiconductor passive transmissions lines which consist of inductance with series resistance and capacitance with parallel conductance, limitations exist in adapting the method for power electronic systems. Elements such as branch capacitors or node inductors which are often found in some power electronic systems cannot be directly modeled using traditional LIM without deviating from the ideal modeling of such elements, often requiring said components to be modeled as their integration companion model equivalents that incorporate inserted fictitious latency

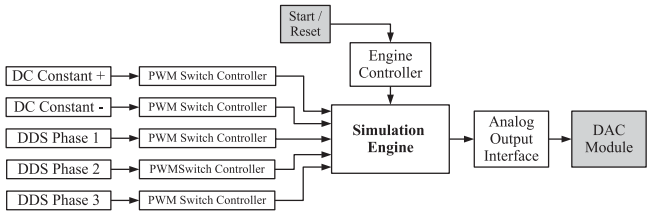

Fig. 13. Top-level design for simulation platform.

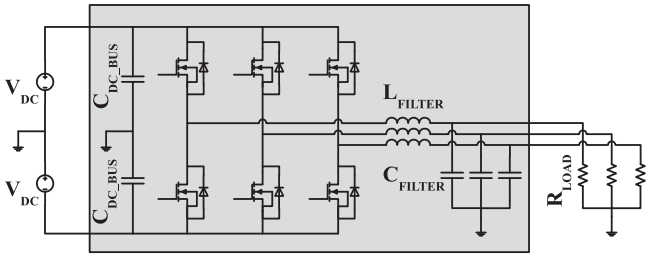

Fig. 14. Three-phase dc/ac converter.

to fit LIM structure [15]. Moreover, some topologies, such as wye or delta arrangement of inductors seen in three-phase converter systems, cannot be modeled in LIM without inserting fictitious, small capacitances to ground at each node connection of the inductor branches. If a converter topology contains floating branch or node elements, techniques beyond traditional LIM would be needed to model mutual capacitance or inductance between such elements, such as seen in [22]. Along with topology restrictions, the LIM expects sufficient latency to exist in all branch and node elements of a given network topology while still achieving feasible and stable time steps; should a system incorporate small latency elements, additional latency is to be inserted, which is undesirable as doing so will deviate the model from the original system. Despite these limitations, sufficient number of common power electronic systems can be effectively modeled in LIM without deviating significantly, or at all, from the overall characteristics of the system in question. In this paper, we demonstrated how this modeling can be done for both boost and buck two-level converters; additional topologies will be analyzed in future papers.

IV. Simulation Results

To demonstrate the FPGA execution of LIM for the real-time simulation of power electronic based systems, three test models have been selected and implemented with the LIM FPGA simulation engine, using the top-level design of Fig. 13. A Xilinx VC707 Virtex-7 FPGA evaluation board was used in the setup, with a Texas Instruments DAC34H84EVM four-channel digital-to-analog converter (DAC) evaluation board handling the analog output. The simulation engines and top-level design were, respectively, developed in C++ and VHDL with Xilinx Vivado HLx 2015.4 suite, and each engine used 64-bit signed fixed-point data types with 35 fractional bits. All designs were loaded on the FPGA board and ran in real-time at 40 ns time step, using a 20 ns (50 MHz) clock source.

A. Test Model Description

Three test models have been used. The test models used in this paper are the same used in [14] to facilitate a comparison between the presented approaches. The first test model used is a three-phase dc/ac converter, depicted in Fig. 14. The converter has been modeled using LIM as described earlier in this paper and the model parameters are reported in Table I. The converter operates with 12 kV dc input. Switching frequency for the converter is 100 kHz. The second test model is a single-bus power

TABLE I
Three-Phase DC/AC Converter Parameters

C_{FILTER}	$C_{\mathrm{DC\_BUS}}$	L_{FILTER}	L_{FILTER}
1 μF	1 mF	0.1 mH	7 Ω

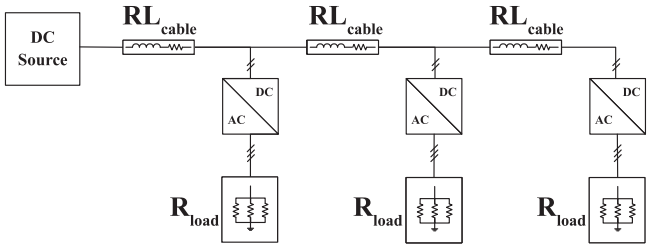

Fig. 15. Signal-bus shipboard power system.

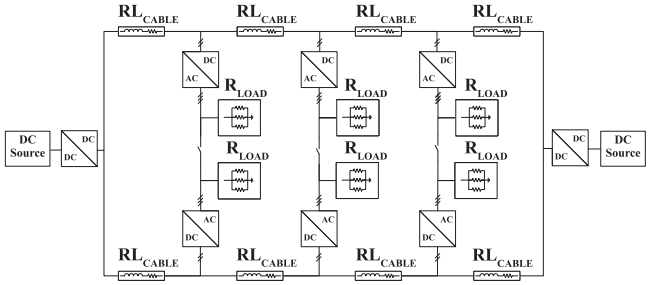

Fig. 16. Dual-bus shipboard power system.

system as can be found on ships or in a microgrid; the system is depicted in Fig. 15. The second test model is implemented using the same converter model and parameters of the three-phase converter system; the load resistance has been modified so that total load of the system is equal to 40 MW. This system contains three converters and uses a straight dc input source of 12 kV. As third and last example we used a dual-bus shipboard power system, displayed in Fig. 16. This system is similar to the single-bus system, but is composed of six dc/ac converters and two dc/dc converters.

B. Model Control Scheme

In all three test models, all of the converters were merely controlled in open loop using simple pulse width modulation which compares references (dc constant for dc/dc converters,

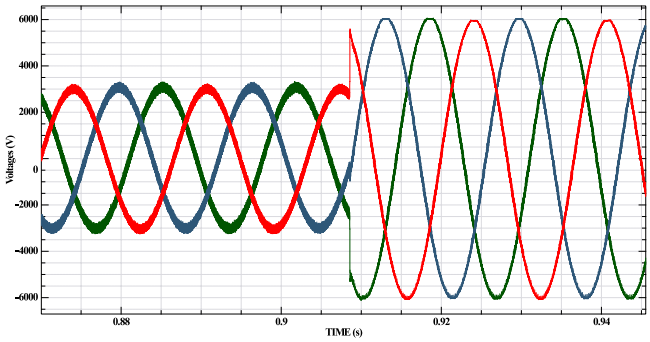

Fig. 17. Analog output for single-bus power system under LIM engine.

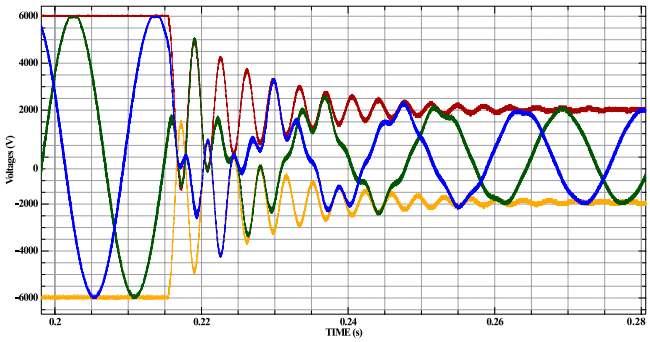

Fig. 18. Analog output for dual-bus power system under LIM engine.

direct digital synthesized sine waveform for dc/ac converters) to a ramp counter for generating switch signals for the converters; control scheme depicted in top-level design seen in Fig. 13. To show dynamic control behavior during real-time simulation for demonstration, the reference signal was set to change when an external control signal outside of the FPGA containing the simulation engine was toggled. Closed-loop control was not considered as such control could mask the simulation solution error of the LIM FPGA implementation and development of full control scheme is outside the scope of this current work.

C. Demonstration

The leap-frog engine designs for the shipboard systems were loaded onto the Xilinx VC707 FPGA board and analog output from the board was logged, as seen in Figs. 17 and 18. The output phases of one of the converters for the single-bus system were captured while the switch control suddenly changed output voltage to higher level. For the dual-bus system, the dc bus voltages and two of the output phases from one of the converters were taken during condition that switch control induced sudden change in dc bus voltages.

To demonstrate deadband interval in the switching action, a tuned, three-phase dc/ac converter simulation engine with induction load in the converter model was created, using a deadband time of 260 ns for 100 kHz switching frequency. One of the phases for the converter, showing visible distortion from deadband interval, is shown in Fig. 19.

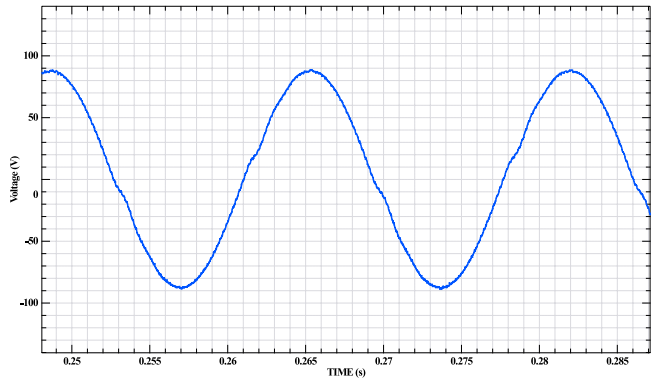

Fig. 19. Analog output, phase voltage for dc/ac converter (only one phase).

D. Simulation Accuracy

To validate the accuracy of the LIM FPGA fixed-point simulation engine for each of the test models, the node voltage solutions were logged from the register-transfer level (RTL) simulation of each model engine FPGA design. Then, the solutions were compared for error to offline C++ implementation of the same LIM solver running at same time step with double precision floating-point data type to determine accuracy loss from using fixed-point numerical representation which typically has lower precision than floating-point types. Afterward, the FPGA engine results are compared to the results obtained using a traditional, EMTP solver, using implicit integration and executed with a 400 ps time step. The results of the EMTP solver at this small time step represent the approximate analytical results of each model as time step approaches zero, which was used instead of true analytical results as analyses of these models are impractical to perform as model size increases. Under both solvers, the models under analysis used open-loop switching control and no external inputs to ensure detected error is contributed only from the LIM implementation.

The error between solvers was computed using two-norm (Euclidean) percent error equation as

$$\text{error}\% = \frac{\|\hat{x} - x\|_2}{\|x\|_2} 100\%. \tag{15}$$

In this equation, $\hat{x}$ is a matrix of all node voltage solutions taken over a 50 ms simulation time period during simulation from the LIM FPGA simulation engine and x is the matrix of all node voltage solutions from the reference solver (floating-point LIM, EMTP) in same time frame. Along with total two-norm error of complete system, the two-norm error of each node voltage solution of the test model (per-element error) was also computed, where, $\hat{x}$ and x of (15) are individual vectors of each node voltage solution extracted from same 50 ms simulation time period for the simulation engine and reference solver, respectively. The greatest per-element error results of the LIM FPGA implementation compared to stated solvers are shown in Table II, with overall error for each in parentheses. Despite going to fixed-point data type which typically has lower precision than floating-point representation, the FPGA implementation of LIM had low error well below one percent compared to the C++

TABLE II
GREATEST MODEL PER-ELEMENT TWO-NORM RELATIVE PERCENT ERROR FOR LIM FPGA SOLVER COMPARED TO OTHER SOLVERS (OVERALL SYSTEM ERROR IN PARENTHESES)

Reference Solver	Three-Phase Inverter	Ship-Bus Shipboard System	Dual-Bus Shipboard System
LIM Floating Point C++ (%)	0.0011 (676.07e–6)	0.00196 (0.0011)	0.001189 (452.72e–6)
EMTP solver (%)	0.02643 (0.0162)	0.02645 (0.01402)	0.02635 (0.01077)

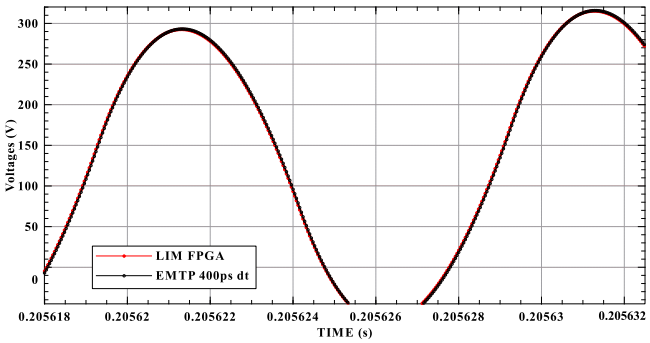

Fig. 20. Comparison of LIM FPGA and EMTP Solver for converter phase ripple of dual-bus shipboard system.

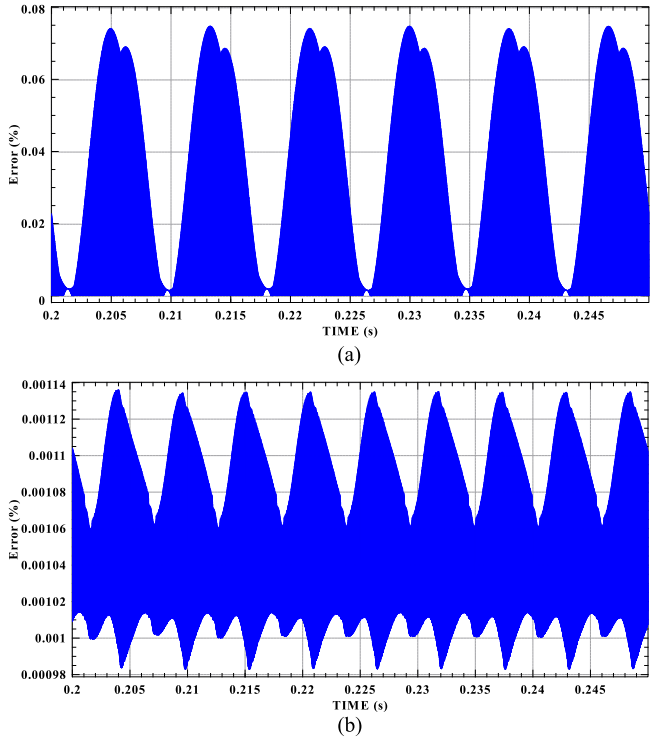

Fig. 21. Error for (a) converter phase output and (b) dc bus of dual-bus shipboard system.

implementation with double precision floating-point data type. In comparison to the traditional EMTP type solver, greatest per-element errors were between 0.02635% and 0.02645%, with overall solution errors being between 0.01077% and 0.01620%. From the small two-norm error, the LIM FPGA simulation engine implementation is shown to be capable of performing accurate ideal modeling of power electronic systems with reasonable loss of error. To visually compare the results of the LIM FPGA and EMTP solvers for the error, the close-up of the phase output ripple of a dc/ac converter in the dual-bus shipboard system taken from both solvers is presented in Fig. 20. The percent error for the same converter phase output over the 50 ms simulation time is shown in Fig. 21(a), with the error for the dc positive bus voltage supplied to the converter shown in Fig. 21(b). The percent error over time for Fig. 21 was computed as the ratio of the absolute difference between the LIM FPGA and EMTP solver results, and the rms value of the EMTP solver results, scaled to a percentage and formulated as

$$\text{error}\% = \frac{|\hat{x} - x|}{\text{rms}\,(x)} 100\% \tag{16}$$

where x is the vector of the EMTP results over time and $\hat{x}$ is the vector of LIM FPGA results of same quantity.

V. SCALABILITY ANALYSIS

One of the important aspects of choosing a simulation method for FPGA implementation is how the method scales on such hardware as the size and complexity of the simulated system grows. One element of scalability is computational delay which affects the size of time step. Should a simulation method require quadratic or even linear scaling of delay as the model size grows, the time step achievable for large systems may become too great to precisely capture high-frequency model behavior in real-time simulation. As such, it is desirable to have computational delay scale sublinearly or stay constant as a model grows in size. Another element of scalability is amount of resources required to realize the computations. To achieve low computational delay in the nanosecond range, it is required to exploit parallelism on FPGAs to maximize number of computations in a given time. This parallelism is achieved by giving each computation of an equation or expression in a system model dedicated resources on the given FPGA. Therefore, as a model grows in size, so does the resources needed to simulate the model. Since all FPGAs have finite amount of resources to give for a simulation engine of a given method, understanding how resources scales with model size is important to ensure larger system models can fit on a chosen FPGA device. In this section, we proceed in analyzing the computation delay and resource usage of the proposed FPGA implementation of LIM.

A. Entities

Due to the fixed nature of the component entities in LIM, the number of operations within a branch and a node component

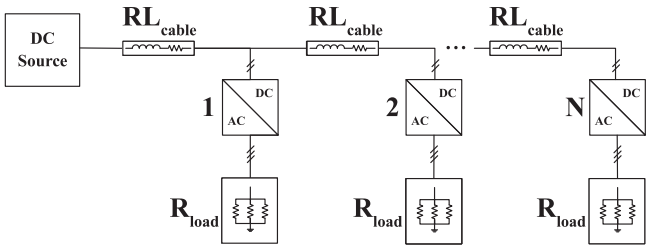

Fig. 22. Scalability test model.

will stay constant, regardless of the system model and its size. Therefore, the total number of operations needed for a given number of branches and nodes in a model grows linearly with the model in question. Since resources realize these operations, they will increase at same scale as said operations. Though resource usage grows linearly, computational delays for both branch and node entities shall stay relatively constant due to entities of same type all updated in parallel.

B. Simulation Engine Time Step and Computational Delay

Due to the leap-frog approach used to handle the simulation flow of the LIM engine, and using the same clock period for branch updates and node updates, the achievable time step for a given system model will always be greater than or equal to twice the said clock period, expressed as

$$\Delta t \geq 2t_{\text{clk}}. \tag{17}$$

This clock period t_{clk} is a function of how long the engine takes to compute the longest half time step update period, whether for branch updates or node updates. The computation time for the update periods is influenced by switching action computations performed under either component update time for converter simulation and by computational time needed to sum branch currents together for a given node component during node update period; on top of computational time for branches and nodes themselves. For branch updates, the compute time will be

$$t_{\text{branch}} = t_{\text{switch\_compute}} + t_{\text{update}} \tag{18}$$

and for the node updates

$$t_{\text{node}} = t_{\text{switch||sum\_compute}} + t_{\text{update}} \tag{19}$$

where $t_{\text{switch||sum\_compute}}$ is the time needed to perform the longest current summation and switching action calculations in parallel. From these computational times, the time step is determined by

$$\Delta t \geq 2\text{MAX}\left(t_{\text{branch}}, t_{\text{node}}\right). \tag{20}$$

C. Setup

As a test case for the scalability evaluation of the LIM FPGA implementation, the single-bus system model shown in Fig. 22 was developed under the method with increasing N number of converters and their corresponding cable segments and loads. For each size of the model, the simulation engine of the model

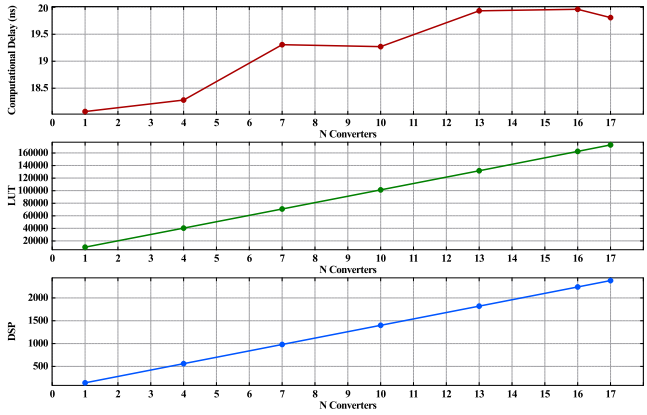

Fig. 23. Plots of LIM scalability results.

TABLE III
SCALABILITY TEST RESULTS

N	Computational Delay (ns)	Time Step (ns)	LUTs	DSPs
1	18.0	40	10149	140
4	18.2	40	40478	560
7	19.3	40	70939	980
10	19.2	40	101341	1400
13	19.9	40	131792	1820
16	19.9	40	162634	2240
17	19.8	40	172903	2380

was synthesized and implemented under Xilinx Vivado HLx 2015.4 suite for the Xilinx VC707 evaluation board and the timing and resource usage report generated by said tool suite was recorded; only the results of the simulation engine and none of peripheral entities (switch control, analog output) were noted. The size of the model was increased by three converters for every run after the first converter, until the simulation engine of the model could no longer fit on the FPGA, either by running out of resources or signal route paths on the FPGA. The LIM engine used 64-bit fixed-point data type as previous tests. For N number of converters included in the model under the LIM implementation, both the branch entity and node entity amount increased by 5N.

D. Results

The LIM simulation engine scalability results are shown in Fig. 23 and Table III. The computational delay is the estimated longest time needed by the LIM engine to update one half time step, either the branch half or the node half. As expected, the results show that LIM is very scalable on an FPGA, having linear increase in resources as converters are added to the model, with 140 DSPs and approximately 10,000 LUTs needed for every converter and corresponding circuit elements; this usage allowing up to 17 three-phase converter models to be instanced on the FPGA.

The computational delay stayed relatively constant for every N of converters included, with delay approximately between 18 and 20 ns, allowing each model size examined to use a 40 ns time

step. It is suspected that the variation in delay is primarily from routing delays between resources on the FPGA, these delays increasing as resource utilization on the FPGA becomes more congested.

VI. Conclusion

In this paper, we demonstrated how LIM can be used for real-time simulation of high switching frequency power electronics systems and how this approach can be implemented for FPGA execution. Through a set of examples we demonstrated how high accuracy can be ensured and that large converter systems can be simulated at significantly small time steps which stay consistent as the systems scale in size. As we have shown, the size of the system that can be simulated at 40 ns is limited only by the resources available on the FPGA. We are now investigate the use of multi-FPGA solutions to solve this limit, using LIM in these solutions as it seems promising in this direction since no central solver is required and the solution is fully parallelizable. Moreover, we are exploring how to expand LIM to model a wider set of components to simulate all possible power electronic systems and to encapsulate various power converter LIM models into reusable components in a library for application users.

References

[1] L. Yunwei, D. M. Vilathgamuwa, and L. Poh Chiang, "Design, analysis, and real-time testing of a controller for multibus microgrid system," *IEEE Trans. Power Electron.*, vol. 19, no. 5, pp. 1195–1204, May 2005.

[2] Z. R. Ivanovic, E. M. Adžić, M. S. Vekić, S. U. Grabić, N. L. Čelanović, and V. A. Katić, "HIL evaluation of power flow control strategies for energy storage connected to smart grid under unbalanced conditions," *IEEE Trans. Power Electron.*, vol. 27, no. 11, pp. 4699–4710, Nov. 2012.

[3] J. Jin-Hong *et al.*, "Development of hardware in-the-loop simulation system for testing operation and control functions of microgrid," *IEEE Trans. Power Electron.*, vol. 25, no. 12, pp. 2919–2929, Dec. 2010.

[4] L. Wai Chung and D. Drury, "Development of a hardware-in-the-loop simulation system for testing cell balancing circuits," *IEEE Trans. Power Electron.*, vol. 28, no. 12, pp. 5949–5959, Apr. 2013.

[5] Z. Li *et al.*, "Power module capacitor voltage balancing method for a ±350-kV/1000-MW modular multilevel converter," *IEEE Trans. Power Electron.*, vol. 31, no. 6, pp. 3977–3984, Jun. 2016.

[6] S. Y. R. Hui and K. K. Fung, "Fast decoupled simulation of large power electronic systems using new two-port companion link models," *IEEE Trans. Power Electron.*, vol. 12, no. 3, pp. 462–473, Mar. 2007.

[7] T. Kato, K. Inoue, T. Fukutani, and Y. Kanda, "Multirate analysis method for a power electronic system by circuit partitioning," *IEEE Trans. Power Electron.*, vol. 24, no. 12, pp. 2791–2802, Dec. 2009.

[8] A. Benigni, A. Monti, and R. Dougal, "Latency based approach to the simulation of large power electronics systems," *IEEE Trans. Power Electron.*, vol. 29, no. 6, pp. 3201–3213, Jun. 2014.

[9] A. Benigni and A. Monti, "A parallel approach to real-time simulation of power electronics systems," *IEEE Trans. Power Electron.*, vol. 30, no. 9, pp. 5192–5206, Sep. 2015.

[10] M. S. Vekić, S. U. Grabić, D. P. Majstorović, I. L. Čelanović, N. L. Čelanović, and V. A. Katić, "Ultralow latency HIL platform for rapid development of complex power electronics systems," *IEEE Trans. Power Electron.*, vol. 27, no. 11, pp. 4436–4444, Nov. 2012.

[11] H. Saad, T. Ould-Bachir, J. Mahseredjian, C. Dufour, S. Dennetiere, and S. Nguefeu, "Real-time simulation of MMCs using CPU and FPGA," *IEEE Trans. Power Electron.*, vol. 30, no. 1, pp. 259–267, Jan. 2015.

[12] Z. Shen and V. Dinavahi, "Real-time device-level transient electrothermal model for modular multilevel converter on FPGA," *IEEE Trans. Power Electron.*, vol. 31, no. 9, pp. 6155–6168, Sep. 2016.

[13] N. Lin and V. Dinavahi, "Detailed device-level electro-thermal modeling of proactive hybrid HVDC breaker for real-time hardware-in-the-loop simulation of HVDC grids," *IEEE Trans. Power Electron.*, Early Access.

[14] M. Milton, A. Benigni, and J. Bakos, "System-level, FPGA-based, real-time simulation of ship power systems," *IEEE Trans. Energy Convers.*, vol. 32, no. 2, pp. 737–747, Apr. 2017.

[15] J. E. Schutt-Aine, "Latency insertion method (LIM) for the fast transient simulation of large networks," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 48, no. 1, pp. 81–89, Jan. 2001.

[16] S. N. Lalgudi, M. Swaminathan, and Y. Kretchmer, "On-chip power-grid simulation using latency insertion method," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 3, pp. 914–931, Apr. 2008.

[17] S. N. Lalgudi and M. Swaminathan, "Analytical stability condition of the latency insertion method for nonuniform GLC circuits," *IEEE Trans. Circuits Syst. II, Express Briefs*, vol. 55, no. 9, pp. 937–941, Sep. 2008.

[18] Z. Deng and J. E. Schutt-Aine, "Stability analysis of latency insertion method (LIM)," in *Proc. IEEE 13th Top. Meeting Elect. Perform. Electron. Packag.*, Oct. 2004, pp. 167–170.

[19] J. E. Schutt-Aine, "Stability analysis of the latency insertion method using a block matrix formulation," in *Proc. Elect. Des. Adv. Packag. Syst. Symp.*, 2008, pp. 155–158.

[20] P. Goh *et al.*, "Partitioned latency insertion method with a generalized stability criteria," *IEEE Trans. Compon., Packag. Manuf. Technol.*, vol. 1, no. 9, pp. 1447–1455, Sep. 2011.

[21] K. H. Tan, P. Goh, and M. F. Ain, "Voltage-in-current formulation for the latency insertion method for improved stability," *IEEE Electron. Lett.*, vol. 52, no. 23, pp. 1904–1906, Nov. 2016.

[22] T. Sekine and H. Asai, "Block latency insertion method (block-LIM) for fast transient simulation of tightly coupled transmission lines," in *Proc. IEEE Int. Symp. Electromagn. Compat.*, 2009, pp. 253–257.

Matthew Milton received the B.S. and M.S. degrees in electrical engineering in 2015 and 2016, respectively, from the University of South Carolina, Columbia, SC, USA, where he is currently working toward the Ph.D. degree in the Department of Electrical Engineering.

He is currently a Research Assistant in the Department of Electrical Engineering, University of South Carolina.

Andrea Benigni (S'09–M'14) received the B.Sc. and M.Sc. degrees in electrical engineering from Politecnico di Milano, Milano, Italy, in 2005 and 2008, respectively, and the Ph.D. degree in electrical engineering from RWTH-Aachen University, Aachen, Germany, in 2013.

From 2009 to 2013, he was a Research Associate in the Institute for Automation of Complex Power System, E.ON Energy Research Center, RWTH-Aachen University. He is currently an Assistant Professor in the Department of Electrical Engineering, University of South Carolina, Columbia, SC, USA.