System-Level, FPGA-Based, Real-Time Simulation of Ship Power Systems

Matthew Milton, Andrea Benigni, *Member, IEEE*, and Jason Bakos, *Member, IEEE*

***Abstract*—In this paper, we present a scalable approach for real-time simulation of ship power systems with high-frequency power electronics converters (100–200 kHz). The proposed approach is based on the latency-based linear multistep compound method and relies on field-programmable gate array (FPGA) execution. Several examples of increasing dimension and complexity are used to evaluate the scalability—both of in terms of computational delay and of resources usage—of the proposed approach. Real-time execution with a 50 ns time step is achieved for all the examples considered.**

***Index Terms*—Field-programmable gate arrays, parallel algorithms, power system simulation, power electronics, real time systems.**

I. Introduction

REAL time simulation, Hardware In the Loop (HIL) and Power Hardware In the Loop (PHIL) techniques have been widely used in the last twenty years to support design and analysis of ship power systems. By filling the gap existing between field test and traditional simulation, HIL and PHIL approaches significantly de-risk the development of new designs. In [1] and [2], a high power PHIL set-up is used for the evaluation of propulsion motor and motor drives. A HIL set-up in [3] is also used for the testing of propulsion systems control. In [4], a HIL set-up is used to evaluate a stabilizing control for MVDC ship power systems. Still in relation to MVDC ship systems, a high power PHIL set-up is used in [5] to evaluate the impact of fast power transfer between dynamic loads and in [6] to test fault management approaches. In [7] and [8], simulation methods for execution of real-time simulation with very small time step have been evaluated in the context of ship system simulation.

To be able to perform effective and accurate HIL and PHIL tests, it is clear that real-time simulation execution at a proper time step is necessary. The selection of the time step depends on the dynamics of interest for the system simulated and for the device under test. The needs of using a very small time step and to simulate large systems have always been the main challenge of real time simulation. Over the last several decades, a significant amount of research and effort has been focused on parallelizing electromagnetic transient stability simulations, mainly for real-time applications. Several approaches have been developed based on latency effects (e.g., Bergeron line model) and on tearing approaches (e.g., Diakoptics). In the area of terrestrial power systems, there has also been an increasing interest – specifically with the US Department of Energy – in the parallelization of transient stability simulation solvers. Historically, one of the main interests for real-time simulation in the electrical engineering field was the testing of relays for terrestrial power systems [9], [10]. Starting in the same time, but with a significant growth of interest in recent years, real-time simulation and HIL methods for power electronic systems have attracted the interest of both academia and industry. The strong nonlinear behavior of the systems and the small time step size required for the simulation of power converters are the main challenges of real-time simulation of power electronics systems. In the last few years, there has been an increasing use of new computational units to address these challenges: mixed solutions based on DSP/CPU and FPGA devices are increasingly common. FPGAs are used both as interface [11] and for computation: in [12]–[14] an AC machine, a power converter and a nonlinear power transformer are directly simulated on an FPGA. In [15], a Modular Multilevel Converter (MMC) converter is simulated using an FPGA in combination with a CPU. In [16], a MMC is simulated in real time using an FPGA with the goal of performing hardware-in-the-loop testing. In [17], the authors propose the use of an FPGA implemented state space solver for the simulation of power electronics converters. In [18], a multi-FPGA platform is used for the simulation for the real-time electromagnetic transient simulation of very large power systems. In [19], a test bench for the HIL testing of electric vehicles is developed using FPGA based real time simulation capability. In [20], a test bench for the HIL testing of multiple-output power converters is realized using an FPGA platform. In [21], a tearing approach is proposed for the parallel execution of the simulation of power electronics converter using and FPGA based platform. In [22], a mixed solution based on CPU and FPGA is used to simulate a two-terminal MMC-HVDC system. In [23], the real time simulation of an MMC is executed on an FPGA platform including device level details.In [24], an FPGA is used for the simulation of an induction machine. In [25], the model of a permanent magnet machines is executed on a FPGA platform for HIL testing.

Manuscript received June 24, 2016; revised December 22, 2016; accepted April 3, 2017. Date of publication April 7, 2017; date of current version May 18, 2017. This work was supported in part by the Office of Naval Research (ONR) under Grant NOOO14-16-1-3042. Paper no. TEC-00542-2016. *(Corresponding author: Andrea Benigni.)*

M. Milton and A. Benigni are with the Department of Electrical Engineering, University of South Carolina, Columbia, SC 29208 USA (e-mail: mmilton@email.sc.edu; benignia@cec.sc.edu).

J. Bakos is with the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208 USA (e-mail: jbakos@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/TEC.2017.2692525

Ship systems are expected to make large use of high switching frequency – 100-200 kHz – power electronic converters (e.g., SiC based). Commercial tools, however, do not yet systematically support scalable real-time simulation with very small time steps – less than 100 ns – as required for simulation of high switching frequency power electronics systems.

In [26], we developed and implemented a simulation method that we named Latency Based Linear Multi-step Compound Method (LB-LMC). The main idea of this method is to exploit the small time step required for the simulation of high switching frequency converter to decouple the solutions of non-linear components from that of the rest of the system. In [26], we showed several examples executed in real time on DSP and CPU with a time step around $20\,\mu s$.

In this paper, we introduce a modified version of the approach proposed in [26] to execute on FPGAs and take full advantage of the characteristic of these devices. Execution on FPGA devices allows exploiting greater levels of parallelization than that on previous platforms which create too much overhead in this regard. Moreover, FPGA execution eliminates any system latency typical of CPU based systems and offers a very high scalability. Significant part of the paper is dedicated to explain how the LB-LMC method has been implemented and on the challenge associated with it. The developed method is tested on ship system examples of different sizes to highlight the high scalability obtained.

II. Latency Based Linear Multi-Step Compound Method

The Latency Based Linear Multi-step Compound Method (LB-LMC) is a highly parallelizable simulation method designed for real-time simulation of dynamic electrical systems. In this section, we provide a summary description of this method which is detailed in [26].

The LB-LMC method is derived from the Resistive Companion (RC) method, solving dynamic systems as a set of linear equations $Gx = b$ every simulation time step, where G is the conductances of the system, b is the current contributions of components, and x is the node voltages of the system. In this paper, we often use the term Resistive Companion to indicate a generic method/solver similar to the ElectroMagnetic Transients Program (EMTP). Unlike traditional RC method, the LB-LMC method models all nonlinear components in a linear network system as functional voltage sources with series resistance, as seen in Fig. 1(a); or as current sources with parallel conductance, as shown in Fig. 1(b). These series resistances or parallel conductances are held fixed and are inserted into the G conductance matrix to stay with standard form of RC components. The nonlinear behavior of the nonlinear components are then reflected in the voltage or current source that is updated every simulation step through an internal step that computes the state equation of the component to update said source. The nonlinear component state equations are expressed as:

$$\frac{di_i^n}{dt} = f(v, i, x_i^n, u_i^n, t) \tag{1}$$

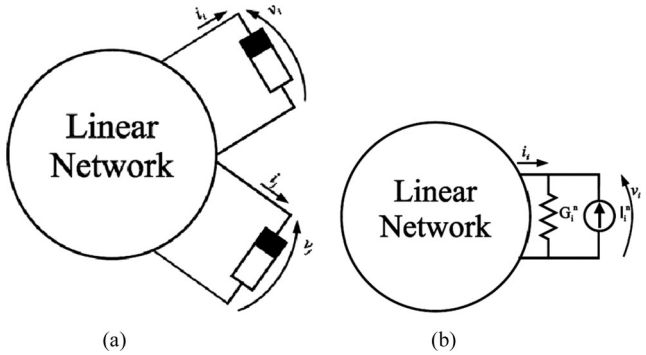

Fig. 1. Linear networks with nonlinear components. (a) With two nonlinear components. (b) With one current-type nonlinear component.

$$\frac{dv_j^n}{dt} = f(v, i, x_j^n, u_j^n, t) \tag{2}$$

where v is the vector of the network node voltages, i is the vector of the network branch currents, x_i^n is the vector of the state variable internal to the i-th nonlinear component, and u_i is the vector of the input internal to the i-th nonlinear component. Components with multiple terminals can be described by a mix of these current and voltage sources. These equations are explicitly discretized to obtain:

$$I_i^n(k+1) = f(v(k), i(k), x_i^n(k), u_i^n(k), k) \tag{3}$$

$$V_j^n(k+1) = f(v(k), i(k), x_j^n(k), u_j^n(k), k) \tag{4}$$

Since the state equations for I_i^n and V_j^n are explicitly discretized and only depend on the solutions from previous time step, and the equations are independent from one another, each nonlinear component can perform its internal step in parallel to other components. From these state equations, the source contribution vector b can be updated and the system solution each time step can be found with:

$$Gx(k+1) = b(v(k), i(k), I^n(k), V^n(k), k) \tag{5}$$

From having the conductance matrix G held constant due to consisting of only fixed conductances, LU factorization for the LB-LMC method system solver can be performed offline, and only forward and backward substitution to solve the system is performed each time step.

Fig. 2 shows the solution flow for LB-LMC. In this flow, G, x, and b are built from initial conditions and the LU factorization of the conductance matrix is performed. Once all non-linear components are initialized, the simulation loop begins. Each iteration consists of each component performing its own internal step in parallel, then the source vector b is updated. From the updated b vector, the system solution x is computed via forward and backward substitution and saved for the next step. The simulation loop continues until final simulation time is reached.

The most important characteristic of the proposed approach is the use of explicit integration for the non-linear components, while the linear part of the network is integrated using an implicit numerical method. While the use of a Linear Multi-step Compound method offers always better accuracy and stability property than the worst of the integration methods used, at the

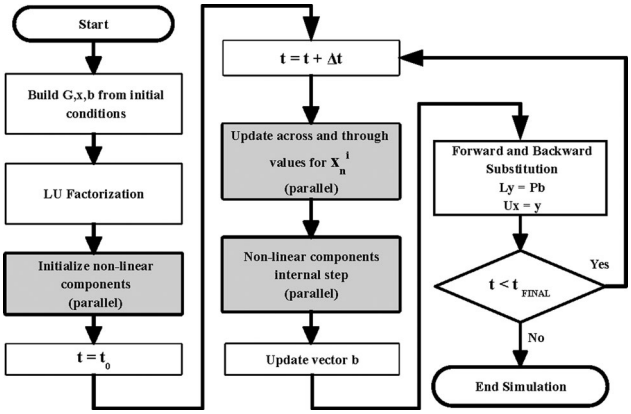

Fig. 2. LB-LMC solution flow.

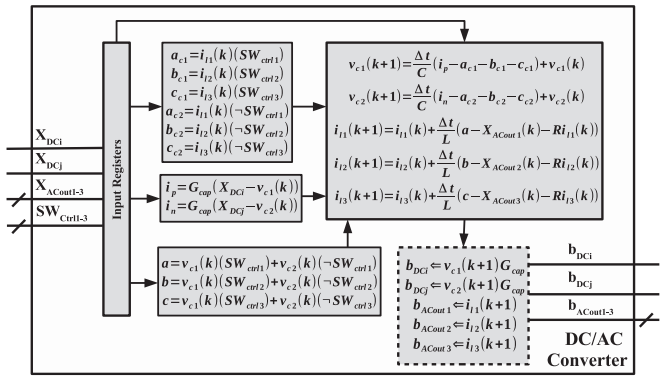

Fig. 3. Example of DC/AC converter component entities.

same time the use of an explicit integration algorithm always implies some concern related to stability and accuracy. In [26] — where we first presented the method — we performed a complete stability analysis of the proposed method. We showed several examples related to power electronics systems and multi-physic applications and most important we showed how for power electronics system the selection of the time step is driven by the switching frequency of the converters and not by the stability and accuracy limit of the proposed LB-LMC approach.

III. FPGA Encapsulation

In this section, the encapsulation of the LB-LMC method elements for FPGA implementation is explained.

A. Component Entities

For each nonlinear component type used to model a system, a FPGA entity is developed. As input, these component entities take the system solution computed in a previous time step. Along with system solution, component entities can also take other input signals to control behavior of the entity, such as switch controller signals for a DC/AC converter component. At the beginning of each time step, the component entities sample and register their inputs. From these inputs and past internal states, the components perform their internal step for (3) and/or (4) and compute their source contributions.

The component entities perform computational operations for their internal step in a non-pipelined, dataflow (data-driven) manner. In this manner, all internal step operations are immediately executed in response to any changes in the component entity inputs, or in the results passed between operations. Moreover, these operations are all performed in parallel to one another, no matter the dependency between operations. Due to this execution flow, internal step operations never wait on prerequisite operations to finish to begin their own execution, the operations converging to correct results as prerequisite ones complete. All internal step operations of a component are expected to complete with correct results within a single pass before new inputs are registered.

An example component entity for a DC/AC converter (see Fig. 8) is depicted in Fig. 3. The DC/AC converter entity takes five inputs that are the DC bus and AC phase voltages on the terminals of the converter, and three switch control inputs to control the output phase modulation. Each time step, the component will register its past states and inputs from step k then use these to execute its internal step. The internal step for the converter involves handling the switching action of the converter through toggling bus capacitor voltages and filter inductor currents ($a, b, c, a_{c1}, b_{c1}, c_{c1}, etc.$) and computing the said capacitor and inductors states for the current time step $k+1$. The source contribution computational step (dashed block) computes the source currents for the bus capacitors and feeds these currents and the inductor currents out as the contribution output. Using dataflow execution in the DC/AC converter entity, all of the internal step operations, as depicted in Fig. 3, are executed in parallel, propagating results to dependent operations without wait until the source contributions of the component converge to correct results in a single pass.

B. System Solver Entity

A dedicated system solver FPGA entity is created to compute the system solution. This solver entity takes as input the component source contributions and accumulates these contributions together to create the whole source vector b used to compute the system solution. The entity provides the system solution vector x as output which are fed back to component entities as input for the next time step execution.

Unlike the original LB-LMC method, the system solver entity does not use forward-backward substitution for system solution computation. Instead, this entity uses an inverted conductance matrix precomputed offline and multiple algebraic sum of product (SOP) expressions to find the system solution. In this approach, the system solution is found by solving (5) for the vector x like in (6), where A is the inverted G conductance matrix ($A = G^{-1}$).

$$b = f(v(k), i(k), I^n(k), V^n(k), k)$$
$$x(k+1) = Ab \tag{6}$$

This solution is computed by expanding the multiplication between A and b matrices into SOP expressions, like seen in (7), which are to be each computed individually from one another. Since the inverted conductance matrix is fixed, the A terms in the

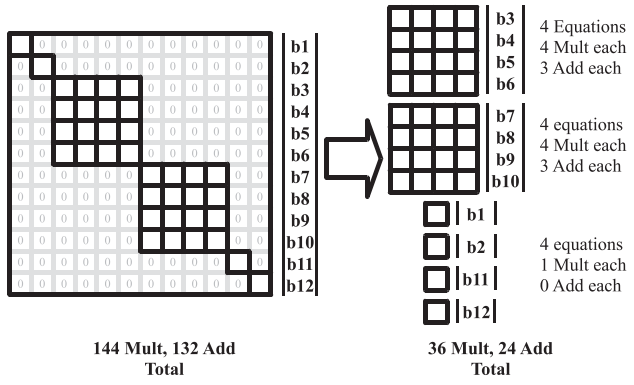

Fig. 4. Separation of subsystems.

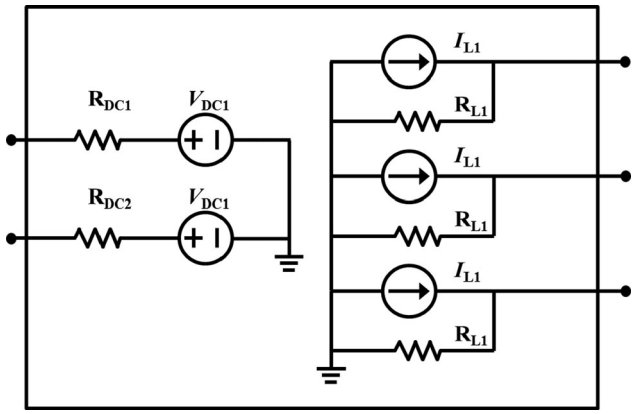

Fig. 5. DC/AC converter system solver structure.

SOP expressions can be defined as constants in said expressions.

$$x = Ab \Rightarrow \begin{matrix} A_{11}b_1 + A_{12}b_2 + \cdots + A_{1n}b_n \\ A_{21}b_1 + A_{22}b_2 + \cdots + A_{2n}b_n \\ \vdots \\ A_{n1}b_1 + A_{n2}b_2 + \cdots + A_{nn}b_n \end{matrix} \quad (7)$$

One main benefit of using this approach over forward-backward substitution is division operations are not required in calculations which tend to be computationally more expensive timewise and use more FPGA resources compared to addition and multiplication operations. Moreover, this approach has only SOP expressions for the system that can be solved for system solution elements in parallel. A disadvantage to using this approach is that since the A matrix is precomputed offline and the SOP expressions are dependent solely on the system being modeled, the system solver entity and its expressions will have to be recreated or modified for each new system that is to be simulated.

IV. System Solver Realization

In this section, we detail how the system solver can be designed to realize desired FPGA resource usage and computational latency.

A. Subsystem Decomposition

Due to how the nonlinear behavior of components is moved to the source contribution computations from the conductance matrix in LB-LMC, it is possible to have multi-terminal components modeled as separate elements whose conductances are independent from one another. Then, the elements' behavior is coupled together via the component's internal step to properly model the whole component. For example, if we consider the three phase DC/AC converter discussed in Section III, from a system solver point of view the converter reduces to a set of voltage and current sources, as seen in Fig. 5. A component like this may lead to a global conductance matrix that is block diagonal with up to six different blocks, even if this is not typically the case because other components may create links between the different subsystems; at the least, fewer diagonal blocks are obtained. It is important to underline that in any case the user is never involved in this process. To help the reader understand the practicality of this approach in real system models, we indicate in Section VIII how many subsystems are obtained for each of the models considered. From exploiting this possible separation of elements, the overall system model is expressed to contain independent subsystems which appear as independent diagonal blocks on the conductance matrix. A subsystem solver can be created from each diagonal block matrix and operated separately to compute a sub-vector of the solution. These subsystem solvers can be encapsulated into the top level system solver. The impact of using subsystem solvers is that the number of terms per system solution equation can be reduced substantially, lowering amount of FPGA hardware resources required.

An example of this subsystem separation for a 12-node system is shown in Fig. 4. In this example, the system has two 4-node subsystem blocks and four 1-node blocks. If this system was solved without subsystem decomposition, 144 multiplications and 132 additions would be needed. However, with the decomposition, the operations are reduced to 36 multiplication and 24 additions, significantly reducing resources needed for the system solver. The shipboard power system models we present in this paper are expressed with a similar structure as this example.

B. System Solver Architecture

The system solver is implementable using two types of architecture: dataflow execution that solves solution equations in parallel, data-driven manner within one pass, and multi-cycle execution which solves solution equations sequentially in multiple iterations within single time step. These architecture designs are explained below.

1) Dataflow Execution: In the dataflow implementation, the system solver solves all of its SOP solution equations entirely in parallel using a data-driven approach. All solution equations are each given dedicated computational units, composed of combinational multiplier and adder units on the host FPGA, which operate independently from each other. As component source contributions are provided to the system solver, the computational units will compute the system solutions immediately without the need to wait on any control or clock signal to initiate the computations; only the input contributions are

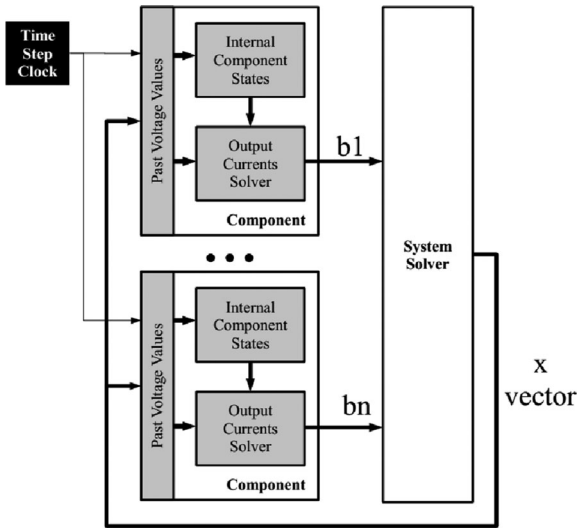

Fig. 6. Simulation engine.

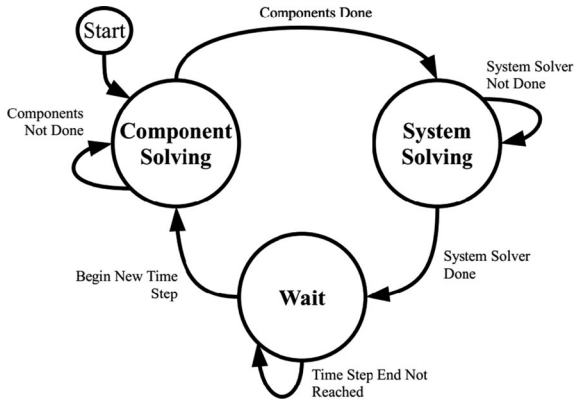

Fig. 7. Finite state machine for multi-cycle simulation engine.

required to start computation. This approach allows solutions to be produced without delays induced from performing clocked operations sequentially.

2) Multiple Cycle Execution: Another approach to implementing the system solver is to have it compute the system solution within multiple clock cycles per time step. In this approach, the solution equations are broken up into like operations. These like operations are then executed sequentially, with a set number of operations executed per clock cycle. After all operations of the solution equations are executed over multiple clock cycles, the results of each operation are compiled or accumulated to reach the complete system solution. The equation operations are performed by computational units which are reused every clock cycle as the operations are expected to be identical but with different inputs. The reuse of the same computational units every iteration allows reduction of FPGA resource usage for larger system models though at the expense of additional computational latency per time step from executing operations sequentially.

An effective usage of multi-cycle execution is to iterate the solving of each subsystem block in a model. With such a setup, each subsystem block is solved each cycle of the system solver. If a model has sizable but few subsystem blocks, then using same subsystem solver and iterating it per subsystem can noticeably reduce resource usage while maintaining low enough clock latency for nanosecond-range time steps.

V. Simulation Engine Composition

This section provides explanation of how the entity encapsulations of the components and system solver are linked together on FPGA hardware to perform simulations.

To perform simulation of a system with the FPGA-adapted LB-LMC method, a simulation engine like seen in Fig. 6 is composed, consisting of multiple component entities and one system solver entity tailored to the system simulated. In the engine, a component entity for each nonlinear component of the system is instanced and their source contribution outputs are linked to the appropriate inputs of the instanced system solver entity. The system solution output of the system solver is fed back to the component entities' inputs, the components taking solution elements that corresponds to their model terminals. If component entities require input from peripherals such as a switch controller, the appropriate FPGA elements are added to the design and linked to the requiring component entities.

The execution scheduling of the simulation engine depends on whether the dataflow or multi-cycle system solver is used:

A. Single Pass With Dataflow System Solver

In use of the dataflow execution system solver, the simulation engine execution is performed in one pass, bounded to a system clock whose period is equal to the simulation time step. On the start of the time step, the component entities sample their inputs for the system solution from past time step and any peripheral inputs. Then, the components perform their computations. As source contributions' values are computed, the dataflow system solver will immediately compute the current time step system solution without wait. The choice of time step clock period is selected to be greater than the computational time needed by the simulation engine for stable operation.

B. Multiple Passes With Multi-Cycle System Solver

For the simulation engine using the multi-cycle solver, the composition of the engine is similar to the single pass design, but multiple clock cycles are required to compute a system solution per time step period. The component entities will need only a single clock cycle to compute their solutions as they are designed to compute the results in dataflow manner within a single pass. Moreover, these component entities need to compute their solutions before the solver can begin. As such, a finite state machine is required to synchronize the execution of the component and system solver entities to one another and to the simulation time step. This finite state machine is created to have the component entities solve their contributions first, and then allow the system solver to compute the system solution. Once the system solution is computed, the state machine has the engine wait until the beginning of the next time step period. This state machine driven operation is shown in Fig. 7. The

bounding of the entities' solution computation to each engine state is done through use of start input signals of each entity which is triggered by the state machine during each state.

VI. FPGA Implementation

We discuss in this section the implementation of the LB-LMC simulation engines in regard to how computation execution is scheduled for parallelism and how numerical quantities are stored and processed.

For high scalability of performance of the LB-LMC method on FPGAs, the parallelism of FPGA hardware is exploited to accelerate computations. To utilize this high parallelism, all equation computations in component entities are expressed to be executed independently where possible, allocated to dedicated arithmetic units for each equation so that they can be solved in parallel. Furthermore, to avoid serial data paths in component entity computations, solution equations are expressed to avoid dependencies between one another where allowed by the component's model and solution integration method. Furthermore, all component entities are instanced with independent hardware.

Parallelism is also exploited in the system solver. For the dataflow solver, all system solution equations like seen in (7) are expressed to have dedicated arithmetic hardware provided to each one so they can be scheduled to run simultaneously. Moreover, the equations are implemented in dataflow manner, as discussed before, in the form of pure combinational logic composed of Lookup Table (LUT) and DSP slices which compute new solutions as soon as source contribution results change. This execution manner allows solutions to be computed as soon as possible without having to wait for all source contributions to be computed by the component entities. In the multi-cycle system solver, the solution equations, though terms are looped, are also all implemented with separate hardware as well. Due to the repeated use of the arithmetic hardware in the multi-cycle solver for each solution equation during each time step, this hardware is pipelined to reduce number of cycles needed to reach a solution to be equal to number of terms per equation plus any cycles needed to fill the pipelines.

So that computational delays for the component entities and system solver is reduced and mostly dependent on the low propagation delays of the FPGA primitives, fixed-point arithmetic logic is used instead of floating-point logic for all calculations performed within. Common floating-point arithmetic implementations, such as IEEE 754, typically require complex, high-latency, pipelined operations to handle their sophisticated formats. Fixed-point arithmetic logic, on the other hand, can be easily created with simpler combinational logic for integer arithmetic which does not require pipelining. Due to not needing to be clocked or pipelined to produce an output, fixed-point computational delay can depend almost solely on propagation delay of the comprised logic primitives. Since fixed point arithmetic hardware is much simpler than floating point hardware, these propagation delays can be kept low. Moreover, many FPGA platforms have built-in integer DSP slices or blocks which can be applied to accelerate operations and reduce delays of integer and fixed-point arithmetic. The main downside to using fixed-point arithmetic is limited numerical precision compared to floating point, which can adversely affect numerical stability and accuracy. However, this limitation can be alleviated with careful selection of integral and fractional bit widths for fixed point signals within a simulation, giving numerical accuracy comparable to use of floating point arithmetic.

VII. Scalability

This section discusses the scalability of the FPGA implementation of the LB-LMC solver as model size increases, in terms of achievable time step (computation delay), clock cycle latency, and FPGA resource usage.

A. Components

The number of operations required to compute the internal states and source contributions of a component is largely dependent on the component model and integration method used. However, the total number of operations required for a collection of components of same model and type will scale linearly as more components of same type are instanced in a simulation engine. This linear scaling of operations also applies to resource usage as each operation of same type uses similar amount of resources. Though resource usage will increase linearly with number of components, the computational delay for all components of same type to perform their operations will stay constant due to the parallel operation of said components.

B. Dataflow System Solver

As the size of a modeled, independent system or subsystem grows to n solutions, the number of operations required for the solver grows by an order of 2, with number of multiplications needed being n^2, and additions being $n(n-1)$. If each operation type (multiplication or addition) is mapped to unchanging FPGA resources without any FPGA synthesis optimizations, the amount of resources needed for the dataflow will also grow by an order of 2 as well. Due to this growth of resources, the system solver can act as a bottleneck that determines how large of a model and its simulation engine can fit on a given FPGA device. To reduce number of operations and FPGA resources in the dataflow system solver, the modeled system is broken up into subsystems where possible and each subsystem is given its own solver with reduced size n.

The computation delay of the dataflow system solver will grow sublinearly as a model size increases due to the multiplication and addition operations performed in parallel, dataflow manner on FPGA hardware. This scaling is unlike a traditional CPU or DSP whose computational time or delay for the solving of these system equations will grow with an order of 2 as the number of solutions increases, due to performing all operations sequentially.

C. Multi-Cycle System Solver

The number of operations implemented in hardware of the multi-cycle system solver is inversely proportional to the number of iterations selected for the solver to compute a solution.

Resource usage will scale similarly, though extra resources are required to enable multi-iteration computation and pipelining. Computational time of the system solver is a function of cycles needed for the solver to reach solution, where the time is a product of the number of cycles, including extra cycles for pipeline priming, and the clock period used.

D. Simulation Engine Time Step and Computation Delay

The time step usable for the simulation engine is dependent on the computational delay and latency of the components and system solver. With the dataflow system solver, the time step must be greater than the sum of computational delay required for the slowest component entity type and the delay needed for the system solver to have all solutions computed and stabilized; this sum being the total computational delay of the simulation engine:

$$\Delta t > t_{solver} + t_{comp\_delay} \tag{8}$$

For larger system models, it is expected that the simulation engine computational delay will be dominated by the system solver delay as component model entities' delays do not grow with system size and expected to typically be small in computational complexity. To greatly reduce system solver delay, and reduce time step, subsystem decomposition can be used within the system solver as noted before.

In the case of using a multi-cycle system solver, the system solver will again greatly influence the time step for the simulation engine due the solver's need for multiple cycle latency needed to reach the system solution each time step. The computation time of the simulation engine will be the number of cycles needed for system solver to reach solution times the clock period used to clock the solver, plus the delay needed for the slowest type of component entities to perform their operations. From this relation, the time step will have to be:

$$\Delta t > n_{sol\_cycles} t_{clk} + t_{comp\_delay} \tag{9}$$

Reduction of multi-cycle system solver latency, and in turn the time step, can be achieved through reducing the number of cycles needed to compute the solution through performing more system solution equation operations per cycle, or to an lesser effect, reduce the clock period. In either case, the tradeoff is higher usage of FPGA resources.

VIII. Test Models

In this section, the power electronic system models used to evaluate the LB-LMC FPGA simulation engine is discussed. Each model is of increasing size and complexity.

A. Three-Phase DC/AC Converter

A three-phase DC/AC converter, depicted in Fig. 8, is modeled in LB-LMC method using parameters seen in Table I. The converter operates with 12 kV DC input. Switching frequency for the converter is 100kHz. The switching devices are modeled using a switching function approach similarly to what is described in [27]. The component entity of the converter model

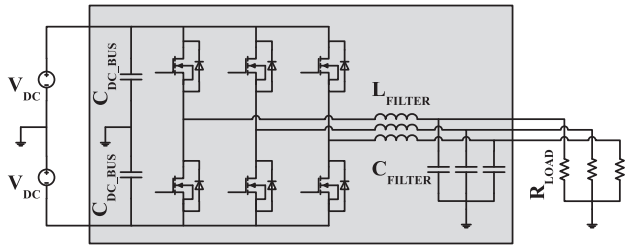

Fig. 8. Three phase DC/AC converter.

TABLE I
DC/AC Converter Model Parameters

V_{DC}	$C_{\mathrm{DC\_Bus}}$	L_{Filter}	C_{Filter}	R_{Load}
12000	0.001	0.0001	1.0e-6	7.0

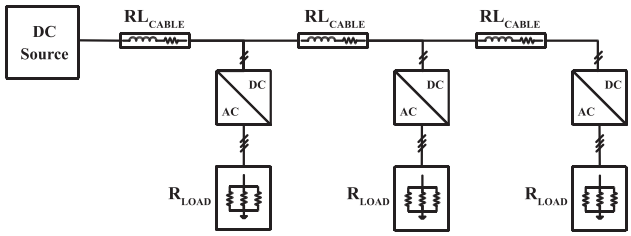

Fig. 9. Single bus shipboard power system.

separates its internal elements into independent subsystems to allow subdividing the system solver into smaller block solvers, though the elements are coupled analytically through the internal step equations. Overall system has five node voltage solutions to solve, each associated with a 1-node subsystem block. During the Component Solving state in Fig. 7, the internal equations of the three phase DC/AC converter are updated together with the voltage and current across and through the other dynamic components of the circuit. In the System Solving state, the node voltages are obtained using the approach described in Section III-B and using the equivalent circuit of Fig. 5 for the AC/DC converter.

B. Single Bus Shipboard Power System

A single-bus power system found on ships, shown in Fig. 9, is modeled using same converter model and parameters as the three-phase converter system, with other parameters chosen to have total system operate with 40 MW load. This system contains three converters and uses a straight DC input source of 12 kV. The overall system has 23 node voltage solutions to solve, and consists of two 7-node subsystem and nine 1-node subsystem blocks.

C. Dual Bus Shipboard Power System

A dual-bus shipboard power system, displayed in Fig. 10, is similar to the single-bus system, but is composed of six DC/AC converters and two DC/DC converters. Parameters for this system is set for 40MW load and the DC/DC converters are set to output 12kV DC voltage onto bus lines. The overall system has

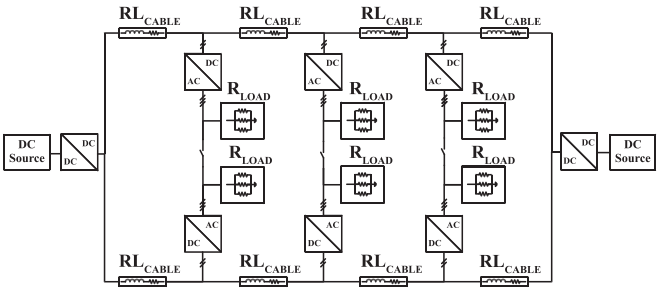

Fig. 10. Dual bus shipboard power system.

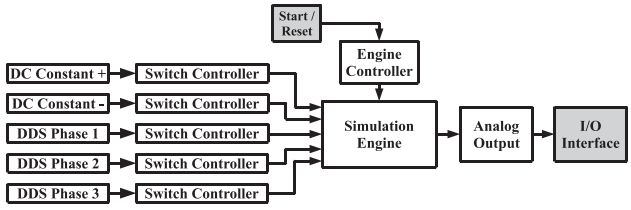

Fig. 11. Top level design for simulation platform.

54 nodes, and consists of two 16-node subsystem and twenty-two 1-node subsystem blocks. Similar to what was indicated for the DC/AC converter case, the power converter internal equations and current/voltage across/through operations are updated in the Component Solving state depicted in Fig. 7.

IX. Implementation Results

In this section, we reveal results taken from separate LB-LMC FPGA simulation engines modeling in real-time the three power electronic systems discussed in Section VIII. All models were run at 50 ns time step, using the dataflow system solver. Resource usage and clock cycle latency of the dual-bus power system simulation engine using the multi-cycle system solver is also presented.

A. Setup

For all three models, the same top-level FPGA design was used, shown in Fig. 11. The simulation engine was developed in C++ under Xilinx Vivado HLS 2015.4, and the complete top-level design was composed in standard Vivado using VHDL for the Xilinx Virtex-7 VC707 FPGA evaluation board. The FPGA was provided a 200 MHz (5 ns) clock source to serve as primary clock for all internal logic. This clock source was divided down to a 50 ns clock within the top-level design to drive the simulation engine. All numerical operations in the simulation engine were performed with fixed point logic defined with HLS ap_fixed library, using 72-bit width with 43-bit fractional precision. The engine controller seen in Fig. 11 handles the start and reset of the simulation engine, as well as the wait state of the simulation engine's finite state machine when using a multi-cycle system solver. All models were run with open-loop switching control to minimize impact of correcting control action on simulation results.

TABLE II
Model Error

	Three-Phase Inverter	Single-Bus Shipboard System	Dual-Bus Shipboard System
C++ LB-LMC (%)	85.97e-06	0.0087	0.0141
Traditional RC (%)	1.0034	0.6545	0.5221

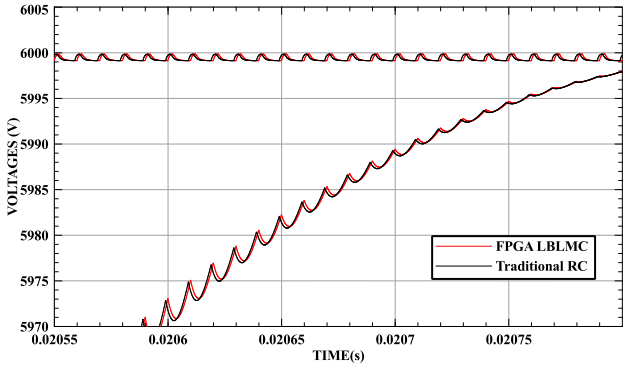

Fig. 12. Error comparison for three phase inverter.

B. Simulation Accuracy and Error

To validate the accuracy of the results for each model, all system solution results, logged from the RTL-simulation of each model simulation engine design, is compared for error to a pure C++ implementation of the LB-LMC solver running at same time step length, using double precision floating point data type. Moreover, error comparison is made to a traditional resistive companion-based simulator running with 500 ps time step. The error, shown in Table II was computed using two-norm (Euclidean) error equation, expressed here:

$$error\% = \frac{\|\hat{x} - x\|_2}{\|x\|_2} 100\% \tag{10}$$

where $\hat{x}$ is a matrix of all solutions taken over a 50 ms simulation time period from the simulation engine and x is the matrix of all solutions from the reference solver in same time frame. As can be seen from the table, going to fixed point from double floating point data type has minimal impact on the the accuracy of the solver implementation, with error around 0.1 percent and below. Compared to the traditional RC solver (EMTP), some accuracy is lost from applying LB-LMC solver. However, accuracy between solvers is still reasonably similar, with percentages of around one percent and less. To better appreciate the impact of the LB-LMC solver on simulation accuracy, we compare the results obtained using the LB-LMC with the one obtained using a traditional RC solver (EMTP) in Fig. 12. For this comparison, we used the DC bus and phase A voltages on the second converter of the single bus shipboard power system example.

C. Scalability

The FPGA resource usage and per-time-step computation delay of the simulation engine of each test model was captured from reports given after full implementation from Vivado,

TABLE III
RESOURCE USAGE AND COMPUTE DELAYS

	Three-Phase Inverter	Single-Bus Shipboard System	Dual-Bus Shipboard System
Time Step (ns)	50.0	50.0	50.0
Compute Delay (ns)	42.8	48.3	48.7
DSP	170 (6.1%)	712 (25%)	1884 (67%)
LUT	3943 (1.3%)	32558 (11%)	87525 (29%)
FF	893 (0.1%)	3637 (0.6%)	8932 (1.5%)

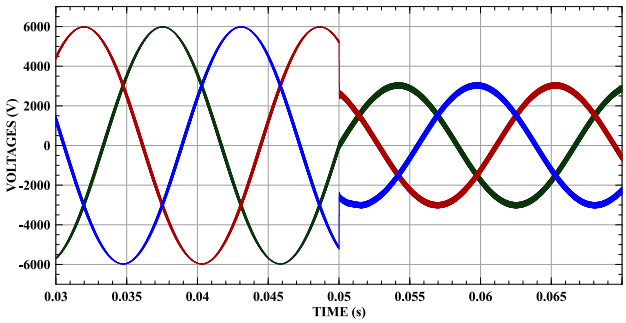

Fig. 13. Single-bus power system analog output.

results shown in Table III, with percentage of Virtex-7 FGPA resources used in parentheses. Results from the engine only are shown, not of the complete top-level design with peripheral hardware. As can be seen from the results, the computational delays stayed low enough to allow the small time step to be used for each model in real-time, despite large growth in model size. Total resource usage scaled approximately linearly. Much of the resource usage increase between models is from the increase in number of component entities whose resources scale almost linearly with amount of components of each type. Applying the subsystem decomposition for the system solver allows the resource usage there to increase more linearly in the case of these models, compared to increasing by an order of two without subsystem break-up.

D. Real-Time Performance

The FPGA implementation is capable of simulating the presented power systems with a time step of 50 ns in real-time as seen in Table III.

E. Demonstration

The simulation engine designs of the two shipboard systems are loaded onto the VC707 FGPA board and analog output of each model was captured, via an oscilloscope, from their respective engine; results seen in Figs. 13 and 15. For the single-bus system model results, three AC output phases from one of the DC/AC converters is shown. The dual-bus system results display two of the output phases and the positive and negative DC bus line voltages. The results for the single-bus system were captured while switch control for the DC/AC converters was set to reduce phase output voltage by half suddenly. In Fig. 14, we report a zoom of the associated transient. Similarly, the

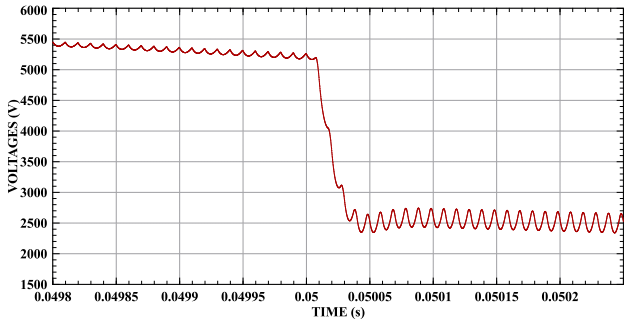

Fig. 14. Single-bus power system analog output, phase 1 zoom.

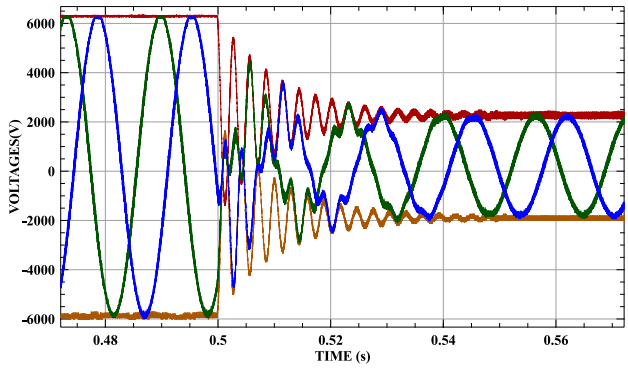

Fig. 15. Dual-bus power system analog output.

dual-bus system results were captured while the switch control of the DC/DC converters powering the system was set to reduce bus voltage to simulate sudden drop in DC/DC converter voltages. Ringing in the dual-bus system voltages is consistent with traditional RC version of said system, and is expected due to operating without closed-loop control to correct for the oscillations.

F. Multi-Cycle System Solver Resource Usage

To evaluate impact on resource usage from using a multi-cycle system solver with subsystem iteration, the system solver for the dual-bus shipboard system was implemented in Vivado with the dataflow design and the multi-cycle design for a 50 ns clock cycle, where the dataflow is expected to compute its solution before 50 ns while the multi-cycle design is clocked every 50 ns. Each version of the solver was implemented separate from the top-level design so that the resource usage reports shown the system solvers' usage only. The multi-cycle version was designed to use same subsystem solver unit for the two subsystems in the shipboard system and compute all solutions and be prepared to receive new source contribution inputs within two cycles; effectively doubling the feasible time step. Both versions solved the 1-node subsystems all in parallel to the subsystem computations. The resource usage of the two system solver architectures and their usage percentage on the Virtex-7 FPGA is shown in Table IV. As can be seen from the results, using the multi-cycle design reduced DSP and LUT usage of the total system solver by approximately 33-36% compared to the dataflow design while still allowing the simulation engine

TABLE IV
RESOURCE USAGE FOR MULTI-CYCLE SYSTEM SOLVER

	Dataflow	Multi-Cycle
Cycles	0	2
DSP	724 (26%)	466 (17%)
LUT	54172 (18%)	36349 (12%)
FF	0 (0%)	3830 (0.6%)

to perform with a reasonable 100 ns time step. Though not an one-to-one tradeoff between latency and resource usage, this resource reduction is significant enough to highlight that this multi-cycle approach can enable simulation engines of large models to potentially fit on a given FPGA where resource usage of a dataflow solver may not allow. Flip-flop usage went up from needing to maintain memory for the iterations of the multi-cycle architecture, but usage percentage on the Virtex-7 is insignificant at below one percent.

X. CONCLUSION

This paper presents the FPGA implementation of the LB-LMC method for scalable, real-time simulation of ship power systems. Using the parallelism and low-latency of FPGA devices, the adapted LB-LMC method is able to simulate power systems of dramatically increasing size while still maintaining effective scaling of computational delays to realize constant time step of 50 ns. Despite maintaining scalable time steps, FPGA resource usage is still a concern for larger system models which can be compensated for through use of subsystem decomposition allowed by the LB-LMC method. As is seen in the results, the use of a multi-cycle solver can also reduce hardware usage at cost of increased computational time and time step, though challenges exist to implement the multi-cycle architecture to have one-to-one or better trade-off between resource usage and computational time. While the LB-LMC FPGA implementation performs well, modeling accuracy is not sacrificed as simulation results deviated little from that of traditional resistive companion method solvers.

REFERENCES

[1] M. Steurer *et al.*, "Hardware-in-the-loop investigation of rotor heating in a 5 MW HTS propulsion motor," *IEEE Trans. Appl. Supercond.*, vol. 17, no. 2, pp. 1595–1598, Feb. 2007.

[2] M. Steurer, C. S. Edrington, M. Sloderbeck, W. Ren, and J. Langston, "A megawatt-scale power hardware-in-the-loop simulation setup for motor drives," *IEEE Trans. Ind. Electron.*, vol. 57, no. 4, pp. 1254–1260, Apr. 2010.

[3] Z. Lin, H. Chen, H. Gao, and K. Zhan, "Hardware-in-the-loop simulation of marine electric propulsion system," in *Proc. IEEE Elect. Ship Technol. Symp.*, 2015, pp. 104–108.

[4] M. Cupelli, M. de Paz Carro, and A. Monti, "Hardware in the loop implementation of linearizing state feedback on MVDC ship systems and the significance of longitudinal parameters," in *Proc. Int. Conf. Elect. Syst. Aircr., Railw., Ship Propulsion Road Vehicles*, 2015, pp. 1–5.

[5] M. Bosworth, D. Soto, M. Sloderbeck, J. Hauer, and M. Steurer, "MW-scale power hardware-in-the-loop experiments of rapid power transfers in MVDC naval shipboard power systems," in *Proc. IEEE Elect. Ship Technol. Symp.*, 2015, pp. 459–463.

[6] M. Andrus, H. Ravindra, J. Hauer, M. Steurer, M. Bosworth, and R. Soman, "PHIL implementation of a MVDC fault management test bed for ship power systems based on megawatt-scale modular multilevel converters," in *Proc. IEEE Elect. Ship Technol. Symp.*, 2015, pp. 337–342.

[7] R. Crosbie, J. Zenor, D. Word, R. Bednar, and N. G. Hingorani, "A low-cost high-speed real-time simulator for ships power systems," in *Proc. IEEE Elect. Ship Technol. Symp.*, 2011, pp. 102–105.

[8] R. Crosbie, J. Zenor, D. Word, R. Bednar, and N. G. Hingorani, "Advances in high-speed real-time multi-rate simulation techniques for ship power systems," in *Proc. IEEE Elect. Ship Technol. Symp.*, 2009, pp. 165–169.

[9] P. McLaren, R. Kuffel, R. Wierckx, J. Giesbrecht, and L. Arendt, "A real time digital simulator for testing relays," *IEEE Trans. Power Del.*, vol. 7, no. 1, pp. 207–213, Jan. 1992.

[10] R. Kuffel, J. Giesbrecht, T. Maguire, R. P. Wierckx, and P. McLaren , "Advances in high-speed real-time multi-rate simulation techniques for ship power systems," in *Proc. Int. Conf. Digit. Power Syst. Simulators*, 1995, pp. 19–24.

[11] H. Figueroa, A. Monti, and X. Wu, "An interface for switching signals and a new real-time testing platform for accurate hardware-in-the-loop simulation," in *Proc. IEEE Int. Symp. Ind. Electron.*, 2004, pp. 883–887.

[12] M. Matar and R. R. Iravani, "FPGA implementation of the power electronic converter model for real-time simulation of electromagnetic transients," *IEEE Trans. Power Del.*, vol. 25, no. 2, pp. 852–860, Feb. 2010.

[13] M. Matar and R. R. Iravani, "Massively parallel implementation of AC machine models for FPGA-based real-time simulation of electromagnetic transients," *IEEE Trans. Power Del.*, vol. 26, no. 2, pp. 830–840, Feb. 2011.

[14] J. Liu and V. Dinavahi, "A real-time nonlinear hysteretic power transformer transient model on FPGA," *IEEE Trans. Ind. Electron.*, vol. 61, no. 7, pp. 1254–1260, Jul. 2014.

[15] 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.

[16] M. Matar, D. Paradis, and R. Iravani, "Real-time simulation of modular multilevel converters for controller hardware-in-the-loop testing," *IET J. Power Electron.*, vol. 9, pp. 42–50, Jan. 2016.

[17] H. F. Blanchette, T. Ould-Bachir, and J. P. David, "A state-space modeling approach for the FPGA-based real-time simulation of high switching frequency power converters," *IEEE Trans. Ind. Electron.*, vol. 59, no. 12, pp. 4555–4567, Dec. 2012.

[18] V. D. Y. Chen, "Multi-FPGA digital hardware design for detailed large-scale real-time electromagnetic transient simulation of power systems," *IET J. Gener., Transm. Distrib.*, vol. 7, pp. 451–463, May 2013.

[19] L. Herrera, C. Li, X. Yao, and J. Wang, "FPGA based detailed real-time simulation of power converters and electric machines for EV HIL applications," *IEEE Trans. Ind. Appl.*, vol. 51, no. 2, pp. 1702–1712, 2015.

[20] O. Lucia, I. Urriza, L. A. Barragan, D. Navarro, O. Jimenez, and J. M. Burdio, "Real-time FPGA-based hardware-in-the-loop simulation test bench applied to multiple-output power converters," *IEEE Trans. Ind. Appl.*, vol. 47, no. 2, pp. 853–860, Feb. 2011.

[21] T. Ould-Bachir, H. F. Blanchette, and K. Al-Haddad, "A network tearing technique for FPGA-based real-time simulation of power converters," *IEEE Trans. Ind. Electron.*, vol. 62, no. 6, pp. 3409–3418, Jun. 2015.

[22] T. Ould-Bachir, H. Saad, S. Dennetiere, and J. Mahseredjian, "CPU/FPGA-based real-time simulation of a two-terminal MMC-HVDC system," *IEEE Trans. Power Del.*, vol. 32, no. 2, pp. 645–655, 2017.

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

[24] N. R. Tavana and V. Dinavahi, "Real-time nonlinear magnetic equivalent circuit model of induction machine on FPGA for hardware-in-the-loop simulation," *IEEE Trans. Energy Convers.*, vol. 31, no. 2, pp. 520–530, Feb. 2016.

[25] N. R. Tavana and V. Dinavahi, "Real-time FPGA-based analytical space harmonic model of permanent magnet machines for hardware-in-the-loop simulation," *IEEE Trans. Magn.*, vol. 51, no. 8, pp. 1–9, Aug. 2015.

[26] 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.

[27] L. Salazar and G. Joos, "PSPICE simulation of three-phase inverters by means of switching functions," *IEEE Trans. Power Electron.*, vol. 9, no. 1, pp. 35–42, Jan. 1994.

Matthew Milton received the B.Sc. and M.Sc. degrees in electrical engineering in 2015 and 2016, respectively, from the University of South Carolina, Columbia, SC, USA, where he is currently an Research Assistant and Software Development Consultant with the Department of Electrical Engineering.

Andrea Benigni (S'09–M'14) received the B.Sc. and M.Sc. degrees from Politecnico di Milano, Milano, Italy, in 2005 and 2008, respectively, and the Ph.D. degree 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 with the Department of Electrical Engineering, University of South Carolina, Columbia, SC, USA.

Jason Bakos (S'96–M'05) received the B.S. degree in computer science from Youngstown State University, Youngstown, OH, USA, in 1999, and the Ph.D. degree in computer science from the University of Pittsburgh, Pittsburgh, PA, USA, in 2005. He is currently a Professor of computer science and engineering and leads the Heterogeneous and Reconfigurable Computing Group at the University of South Carolina. He holds 2 U.S. patents and has published approximately 50 refereed publications in computer architecture and high performance computing as well as a textbook on embedded systems. His research interest include high-performance and energy-efficient computing with emerging processing technologies such as reconfigurable, massively parallel, and processor-in-memory architectures. He received the U.S. National Science Foundation (NSF) CAREER Award in 2009 and won DAC Design Contests in 2002 and 2004. His work is currently funded by NSF, ONR, and Texas Instruments Corporation. He is currently serving as an Associate Editor for the *ACM Transactions on Reconfigurable Technology and Systems* and the General Chair of the IEEE International Symposium on Field Programmable Custom Computing Machines. He is a member of the IEEE, Computer Society, and ACM.