MARTHA: Architecture for Control and Emulation of Power Electronics and Smart Grid Systems

Michel A. Kinsy*, Ivan Celanovic†, Omer Khan‡ and Srinivas Devadas*
*Department of Electrical Engineering and Computer Science
†Institute for Soldier Nanotechnology
Massachusetts Institute of Technology
‡Department of Electrical and Computer Engineering
University of Connecticut

***Abstract*—This paper presents a novel Multicore Architecture for Real-Time Hybrid Applications (MARTHA) with time-predictable execution, low computational latency, and high performance that meets the requirements for control, emulation and estimation of next-generation power electronics and smart grid systems. Generic general-purpose architectures running real-time operating systems (RTOS) or quality of service (QoS) schedulers have not been able to meet the hard real-time constraints required by these applications. We present a framework based on switched hybrid automata for modeling power electronics applications. Our approach allows a large class of power electronics circuits to be expressed as switched hybrid models which can be executed on a single hardware platform.**

I. Introduction

A. Power Electronics and Applications

One of the key underlying physical layers of the smart grid is power electronics. Power electronics can broadly be defined as solid-state energy conversion technology [1] that enables efficient and fully controllable conversion of electrical power. To understand how ubiquitous power electronics has become, consider a few power converter examples: (1) Power electronics enables power flow interface of solar photovoltaics with the grid; (2) it provides an efficient interface between variable speed wind turbines and the grid that enables maximum wind power conversion; (3) it allows for power flow control between an electric vehicle motor and battery; and (4) it enables power grid dynamic stabilization. Power electronics could potentially reduce overall electricity consumption by more than 30% [2].

B. Electronic Design Automation (EDA) of Power Electronics

The level of automation in the design, prototyping, verification and testing of power electronics systems is, in many aspects, in the early stages of development. This is especially true if we compare EDA for power electronics with EDA for integrated digital circuit design. The lack of general tools impedes faster development cycles and system integration, increases the cost and slows down deployment.

The design process usually starts with writing a functional specification of the converter. Based on expert knowledge, experience and analytical calculation, a designer arrives at a conceptual design of a converter. This step leads to preliminary circuit design, that includes switching topology, selection of passive elements and decisions on the appropriate controller platform and control strategy. Once this stage is complete, design tasks are divided into a power processing part and a control part which are often modeled in separate simulation platforms (circuit and control domain) and simulated to verify functional, and then performance requirements. Power processing and controller subsystems are separately prototyped, tested and verified. This is often done using low-voltage emulators that provide some level of confidence towards final integration and complete system testing. Power processing components and controls need to work seamlessly, not only for nominal operating points, but also under faulty conditions, various system disturbances, and over the lifespan of the system. This integration process can lead to large uncertainties in the parameter space.

The design process as described, exhibits severe limitations that can be summarized as follows: (1) non-unified design platforms for power and control parts, (2) use of time-consuming off-line simulations (time-domain) that partially cover the parameter and design space, (3) lack of well-defined design process interfaces, (4) lack of rapid prototyping tools for both hardware and control subsystems, (5) inadequate real-time control system verification and testing, and poor test coverage against faults, parameter variations, etc., and (6) lack of rapid prototyping tools for both hardware and control. In order to address some of these limitations, advancements are needed in the area of EDA tools. Design automation could significantly reduce development cycles, improve reliability, and accelerate the development of more complex systems. In particular, we believe that an integrated EDA framework would immensely benefit the area of automated control testing and rapid design prototyping.

C. Requirements for EDA of Power Electronics

The key elements of power electronic circuits are: electronic (solid-state) switches and energy storage elements on one side, and a control subsystem on the other side. Power electronic systems can be well modeled as switched hybrid systems described with a set of discrete states, each with its own continuous dynamics [3]. The challenge in modeling these circuits stems from the fact that digital computers can only interact with or compute continuous phenomena at discrete frequencies or at discrete intervals of time with integration over these discrete time intervals. An adequate design, testing, and validation of power systems must incorporate high-fidelity,

978-3-9815370-0-0/DATE13

real-time emulation of power electronics in Hardware-in-the-Loop (HiL) configurations. HiL permits proper validation of operational scenarios by interfacing the platform with other physical systems needed to model the environment. The validation includes the ability to check system responses and interfaces, to identify failure modes, and to implement recovery states or redundancies in critical circuits.

In this work, we present a new, advanced, computational framework based on switched hybrid automata that uses new abstractions and modeling approaches and allows various power electronics applications to be expressed under one general hybrid formulation. Our framework includes a software tool for electrical circuit modeling, analysis, partitioning and mapping onto a novel parallel computing engine. The computing engine architecture, specifically tailored for high-performance and deterministic real-time computation, enables high-efficiency execution of switched hybrid models of a large class of power electronics systems. Particular emphasis, in terms of application space and experimental verification, is placed on ultra-low latency, deterministic, and high-throughput computation for power electronics and smart grid systems. Figure 1 depicts our design flow process where circuit models are expressed as switched hybrid models which can be executed on a single hardware platform. The key contributions of this work are as follows:

- Generalized modeling of dynamic hybrid systems, especially, power electronics circuits, as switched hybrid automata;
- A software tool for switched hybrid automata synthesis and heuristics for partitioning power electronics circuits across parallel computing engines, while taking into account electrical component characteristics; and
- A high-performance multicore architecture with cycle-level deterministic computation and routing for switched hybrid automata.

The paper is organized as follows. Section II briefly discusses related work. Section III presents the switched hybrid automaton modeling of power electronics applications. Section V describes the software tool support for modeling and HiL testing and validation. Section IV presents the detailed description of the multicore system architecture. Section VI compares the fidelity of the proposed architecture with the physical system being emulated in the benchmarks. Section VII concludes the paper.

II. Related Work

Power electronics simulation algorithms, for off-line simulations, can broadly be divided into two main categories [4]: (1) nodal analysis based (e.g., SPICE), and (2) state-space equation based (e.g., Matlab Simulink). Most modern power electronics simulations tools are based on the second approach, namely, the state space model formulation. State space modeling approach for power electronics circuits can be further divided into two subcategories [4]: fixed matrix, and variable matrix modeling approach. In [5], Graf et al. present a comprehensive review of current techniques used for real-time simulation of power electronics systems. They indicate that from the simulation point of view synchronous oversampling is desirable but requires $< 5\mu$s sampling time for systems with carrier frequency in the order of 20 kHz. They comment that such a low latency is out of reach of today's real-time processors. There are multiple examples of grid-level digital simulator tools based on off-the-shelf high-performance processors and other software and hardware components [6], [7]. These systems can simulate in real time the power grid's transient and sub-transient behavior, fault conditions, issues concerning the islanding behavior of the parts of the grid, and more. However, many of them are constrained to modeling systems with low switching frequency and therefore slow dynamics. A majority of power converters operate well into the kHz domain, therefore these emulators are unable to accurately model the fast switching behavior and non-linear dynamics of these power systems. Other platforms are designed for a single fixed topology [8]. As a consequence, they are impractical to use as a design tool for rapidly prototyping new power electronics systems.

In power electronics circuits emulation, like most hard real-time applications, tightly bound high-performance systems-on-chip (SoCs) with hard execution guarantees are often preferred [9]. Heterogeneity in these architectures, although making them harder to program, can provide improved real-time behavior by reducing conflicts among processing elements and tasks, as well as improving programmability and flexibility [10]. Unfortunately, current high-performance processors, employ a number of techniques, namely, out-of-order execution, speculation, simultaneous multithreading, dynamic queues and memory hierarchies, that hinder strong quality-of-service (QoS) guarantees and worst-case execution time (WCET) estimations. While there are ongoing research efforts with the goal of providing some amount of predictability in multiprocessor systems, the level of predictability offered is not enough for real-time emulation of power electronics applications.

III. Modeling of Power Electronics Circuits as Switched Hybrid Automaton

In this section we present the generalized modeling of dynamic hybrid systems, in particular, power electronics circuits, as switched hybrid automata. By their very nature, power electronics systems are hybrid systems since they consist of a combination of discrete and continuous features. As such, they are best formalized as hybrid automata (HA) [11], [12]. The general behavior of a hybrid automaton consists of discrete state transitions and continuous evaluation. HA provide a natural modeling formalism for power electronics. This modeling approach allows certain properties of these systems to be verified or for these systems to be automatically synthesized. There are several classes of HA [13]; in our power electronics framework, we use the Switched Hybrid Automaton (SHA) model, where discrete dynamics do not modify the continuous states. In other words, the continuous state of the system does not change during a transition. Hybrid

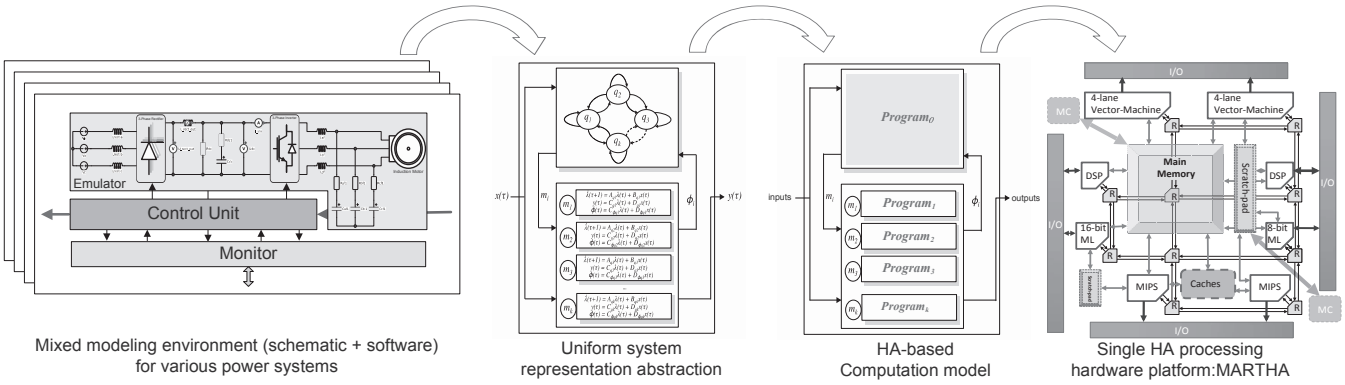


Fig. 1. Our design flow for power electronics systems.

automata are a generalization of timed automata, where the changing rate of variables in each state is expressed as a set of differential equations.

Definition 1: We define a switched hybrid system to be a 6-tuple $H = (Q, \Lambda, f, E, I, \Phi)$ where: $Q = \{q_1, \ \ldots, \ q_k\}$ is set of discrete states, $\Lambda \subseteq R^k$ is the continuous state space; $f : Q \mapsto (\Lambda \mapsto R^k)$ assigns to every discrete state a continuous vector field on Λ; $E \subseteq Q \times Q$ is the set of discrete transitions; $I : E \mapsto 2^{\Lambda}$ assigns each transition $e = (q_i, q_j) \in E$ a guard $\phi_e \in \Phi$.

The switched hybrid system model is given in state space form as:

$$\forall q \in Q \qquad \dot{\lambda}(t) = A_q \lambda(t) + B_q x(t) \tag{1}$$

where $\lambda(t) \in \Lambda \subseteq R^k$ is the continuous state space vector, $A_q \in R^{k \times k}$ and $B_q \in R^{k \times n}$ are *operation matrices*; and $x \in R^n$ is the input vector to the system at time t. Any discrete state of the system belongs to a finite set $Q = \{q_1, \ldots, \ q_k\}$ and further defines the given state space representation. Every discrete state q_i therefore has a unique dynamic behavior associated with it that defines the operation of the system.

Definition 2: In this framework we also define a *system-mode*, denoted m_q, where $q \in Q$, to be the operation of the system defined by given state space $\dot{\lambda}(t) = A_q \lambda(t) + B_q x(t)$ and a given q.

The state space representation of hybrid automaton modes, as defined in Equation 1, can be discretized based on a time-step τ. We use the exact discretization method via state-transition matrix. The discretized state space system matrices, for a given mode are given as:

$$\dot{\lambda}(\tau + 1) = A_{q_i} \lambda(\tau) + B_{q_i} x(\tau) \tag{2}$$

$$y(\tau) = C_{q_i} \lambda(k) + D_{q_i} x(\tau) \tag{3}$$

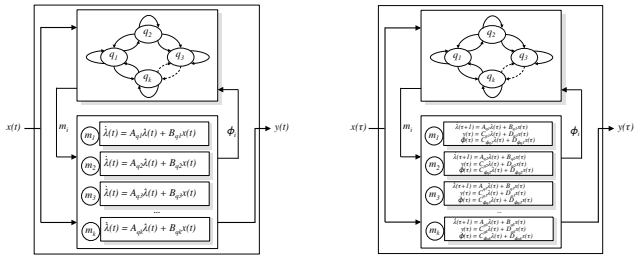
$$\phi(\tau) = C_{\phi q_i} \lambda(\tau) + D_{\phi q_i} x(\tau) \tag{4}$$

where y is the output vector, and ϕ is the guard vector. In the discretized form the set of *operation matrices* $\{A_{q_i}, B_{q_i}, C_{q_i}, D_{q_i}, C_{\phi q_i}, D_{\phi q_i}\}$ defines the dynamic behavior of the system. The switched hybrid model described above can be represented as a block diagram, as shown in Figure 2(a). The equivalent block diagram description of the discretized Switched Hybrid Automaton is given in Figure 2(b).

IV. Multicore Architecture for Real-Time Hybrid Applications (MARTHA)

The design of a computer system to provide the functionalities needed for Switched Hybrid Automaton (SHA)-based application modeling poses a number of technical challenges different from those encountered in general-purpose computers. SHAs are multiple-input, multiple-output (MIMO) systems with complex interactions between software and hardware, hardware and I/O interfaces, analog and digital signals, real-time controls and digital monitoring. A viable architecture model must provide support for the decomposition of models and parallel processing of sub-blocks that alleviates the computation complexity of models.

The MARTHA multicore processor efficiently integrates support for predictable execution-time while providing high-performance and programmability for power electronics applications, smart grid systems, and potentially other hybrid real-time applications. The MARTHA architecture includes SIMD vector-machine-style core(s), used to model linear dynamics of SHAs, with fast, parallel, matrix manipulation operations; general-purpose MIPS-core(s), used for mode transitions, controls, monitoring, and data collection; DSP core(s), used for I/O conditioning, analog/digital communication, and for non-linear machine models; and programmable micro-core(s), employed to model certain signal sources, and to implement on-chip control functions. An interleaved memory structure is used to increase the data transfer rate between the main memory and computational units. The cores communicate via a scratchpad shared memory space and through a mesh network of bufferless routers, where routes are pre-computed and paths configured statically for each system that is being emulated. This heterogeneity in the architecture is dictated by the complexity and the phase variations seen in SHAs.



(a) (b)

Fig. 2. Hybrid modeling of power electronics systems as SHAs.

A. Main hardware modules: Vector Execution Unit

State dynamics calculations in SHAs exhibit the sort of heavy computational and data loads that perform well on single instruction multiple data (SIMD) architectures. For that reason, we explore an architecture under which the SIMD-style execution can be efficiently performed to take advantage of the inherent data parallelism in the state space equations, while providing a flexible programmable computation model. In our current implementation, we adopt a vector-processing core that has a RISC-style architecture with vector operands. The instruction set architecture is based on the MIPS scalar ISA extended to support vector operations. The SIMD format provides excellent code density and allows good usage of the *Instruction Buffer* module. Instructions are prefetched and streamed into the buffer. During static analysis of the program, branch instructions are checked to ensure that branch target instructions are in the buffer. If the program has a branch condition that may violate this requirement, then the buffer is disabled, and instruction fetch goes directly to memory. This enforces the design goal of deterministic execution time.

Feature	scalar Module	Vector Module
$registers$	16 scalar registers numbered from 16 to 31	16 registers, numbered from 0 to 15, per lane
lw \$17, 24(\$22)	scalar load operation	no vector operation involved
lw \$4, 8(\$2)	no scalar operation	load register 4 based on address called only involving lane 0
lw \$6, 12(\$25)	use scalar unit for address calculation	load register 6 based on address calculated for all lanes
lv \$6, 12(\$25)	use scalar unit for address calculation based on register 25 content	load register 6 for each lanes based on the starting address calculated
sv \$2, 4(\$21)	use scalar unit for address calculation based on register 21 content	store register 2 for each lanes based on the starting address calculated
add \$17, \$19, \$23	scalar add operation	no vector operation involved
$multv$ \$4, \$2, \$3	no scalar operation	vector multiply
$mult$ \$16, \$5, \$7	take scalar register 16 content	multiple scalar register by register 5 for each lane

Fig. 3. Vector core register and instruction formats.

The instruction format used is MIPS-compliant, with a few vector operation extensions. In the *Decode* stage, the registers involved in an execution determine whether the instruction is a scalar or a vector operation. Figure 3 shows how the register space is partitioned and how instruction decoding is done. The vector register is a 16-entry register file with 128-bit registers, divided into four 32-bit parts for the four lanes. The scalar register is a 16-entry, 32-bit unit used for scalar or scalar-vector operations. There is a vector length register (VLR) responsible for holding the number of elements in a vector operation. This allows for one instruction to execute an arbitrary number of vector operations.

B. Main hardware modules: Memory Subsystem

The main memory in *MARTHA* features a bank-based or interleaved flat memory system, where the on-chip and off-chip memory organization is fully exposed (and logically programmed before the start of the application). There are 8 independent memory banks, which allow for up to 8 concurrent memory accesses, and increase the overall system memory bandwidth by 8-fold. This approach provides better parallelism and performance. Each bank has separate data and address lines, and the implementation of the bank-based memory is also relatively simple. The low-order bits of the address are used to select the appropriate bank, and the higher-order bits are used to address the location within the memory bank. In the current implementation, each bank is $8MB$. Part of the application compilation process is to make bank accesses contention-free through judicious scheduling of tasks onto processing units. One key aspect of the *MARTHA* architecture, is the interaction of interleaved memory structure and the *load unit* of vector-processing core. The low-order bits of the address contain information regarding starting bank index, stride, and single or multiple bank accesses (3-bit, 2-bit and 1-bit respectively).

The MIPS-core has both instruction and data caches. The vector-processing core has an instruction buffer (i.e., instruction cache). These cache structures can be disabled to enforce tighter data access determinism. The decision is made at the end of the application compilation, based on instruction code size and the working set. If instruction code can be streamed into the caches, with no misses even in presence of branching, then prefetching is done to fill the cache or streaming is set up before the start of the application. Data caching is only done in the MIPS-core and enabled if the application can tolerate a miss or if the working set is fully cacheable. For SHA-based application modeling, the vector-processing core is meant to work on a large data set. For this reason, a cacheless data memory organization is adopted at the vector-processing core. This cacheless memory structure also removes the need for hardware logic for cache line replacement and coherence. Furthermore, it makes the data access time deterministic and latency predictable.

C. Hardware prototype on a Virtex-5 FPGA

A version of the proposed *MARTHA* architecture, shown in Figure 1, is prototyped on FPGA using the *Heracles* RTL-based multicore design platform [14], and synthesized using Xilinx ISE Design Suite 11.5 on a Virtex-5 LX330T platform board. On the board, the MIPS core runs at 162.5 MHz, and uses 1.3% of the available registers and 2.6% of the available lookup tables. The virtual-channel router synthesis requires 1.3% and 1% FPGA resources respectively for the number of registers and lookup tables. The vector core uses about 12% of the available FPGA resources. For the DSP block we use the Virtex-5 DSP48E slices (each slice can provide support for a 25×18 bit multiplier). The DSP block uses 12% of the board DSP48E resources.

V. Software Support Toolchain

In this section we describe a software environment for: specifying power electronics systems, synthesizing them into switched hybrid automata (SHA), and mapping them onto the hardware platform. The automated compilation environment has three layers: the power electronics application modeling layer, the SHA formulation, synthesis, and analysis layer, and finally the mapping and execution layers.

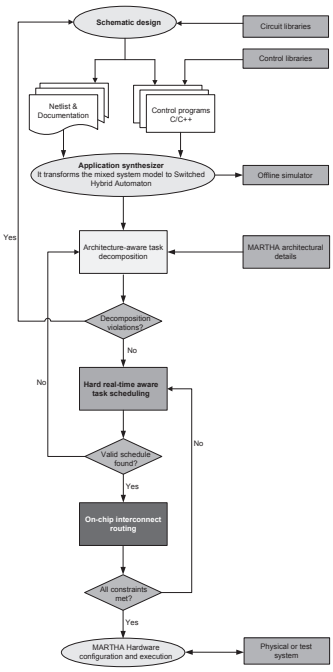


Fig. 4. Automated SHA compilation flow for MARTHA.

The first layer consists of the schematic editor and model netlist abstractor. It is a mixed-model design environment where schematic and software components can be combined. The *Schematic Editor* gives designers the ability to convert their power electronics systems specification into model representations, where they can define component values, connectivity, and characteristics. These components range from linear circuit elements, like resistors, inductors, and capacitors, to machine models, including semiconductor devices, like diode and IGBTs, and voltage and current, both independent and controlled, sources. The library of components has predefined power electronics switching blocks, such as a three-phase diode rectifier block, and control circuit blocks, such as a three-phase carrier-based pulse width modulator (PWM). The component library can also contain user-defined modules, blocks, or control/monitoring software programs. Given a power electronics circuit and a set of control code snippets, written in C++, the *Netlist Extractor* generates a unified system representation with components, their values or code, and connectivity.

The hybrid model formulation is done through the *Application Synthesizer*, which generates the set of matrices representing the discrete states of the power electronics systems, transition equations, control functions, and output signals. The synthesizer output files can be fed to an offline simulator for soft simulation of the system, or can be used to further decompose the application into tasks with execution constraints for hardware mapping. We develop several task scheduling and on-chip routing algorithms for mapping of the SHA system onto the hardware, depending on switching blocks and component blocks signal communications. This stage consists primarily of finding the proper task-to-core mapping, loading state-space equation matrices and transition data to memory, loading execution bit files onto cores, and configuring the on-chip network. We use the C++ programming language for data manipulations and Python for scripting and graphic user interfaces. Figure 4 shows automated SHA compilation flow for the MARTHA architecture.

The switched hybrid automaton approach to modeling of power electronics circuits exhibits deterministic and bounded-time execution. In addition, due to minimal circuit representation it has the potential to be efficiently implemented on an application-specific digital architecture. One of the drawbacks of this approach is the exponential growth of the number of discrete modes, as a function of switching elements. Indeed, the number of modes is 2^n where n is the number of switches. This poses a serious challenge for using this approach for complex systems due to the exponential increase in memory resources on one side, and the exponential complexity increase of the finite state machine on the other. In order to alleviate the exponential growth of the problem, in the *Application Synthesizer*, a larger complex circuit is partitioned into subcircuits that communicate via slowly changing state space variables (e.g., capacitor voltage). Although we have devised a partitioning algorithm, the detailed presentation of this algorithm and the mapping algorithms are beyond the scope of this paper.

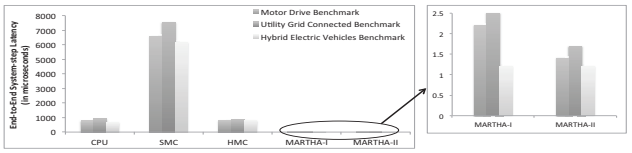
VI. Evaluation of proposed design methodology

In this section we present the results of our experiments to validate our design framework. We focus on four key performance areas: (1) computation power, which affects system-step latency and allowable system dynamic behavior complexity; (2) memory, and its effects on system performance and data access time predictability; (3) on-chip communication, which influences system-step latency and the deterministic aspect of the system; and (4) I/O signal processing, which affects system-step latency. Three representative power electronics dynamic system models are used as benchmarks for evaluation: (1) variable speed induction motor drive; (2) utility grid connected photovoltaic converter system; and (3) hybrid electric vehicle motor drive.

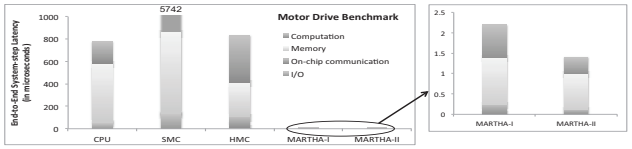
A. Hardware-in-the-loop (HiL) deployment

The modular design of *MARTHA* allows for different instances of the architecture to be realized with a varying level of parallelism, hardware complexity, heterogeneity, and scale. In addition to the four FPGA prototypes, namely, a single MIPS core (SMC), a homogeneous mesh network of 8 MIPS cores (HMC), a MARTHA-I architecture (1 MIPS core and 1 vector core), and a MARTHA-II architecture, shown in Figure 1, we also present the *software-only* implementation (CPU) on Intel Xeon 6-core CPU to put the results in context. Figure 5(a) shows the end-to-end system-step latency for the different platforms. Figures 5(b)-5(d) show the breakdown of the latency for each benchmark.

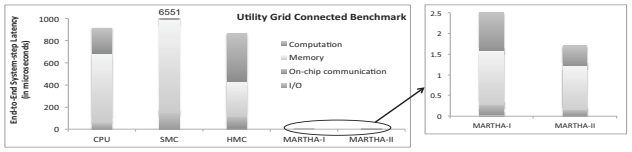
To validate our proposed architecture for power electronics applications, we deploy the prototype in an HiL testing configuration for the motor drive model. Figure 6 shows the experimental setup, where the same controller is used for both the real system and the computation engine emulating



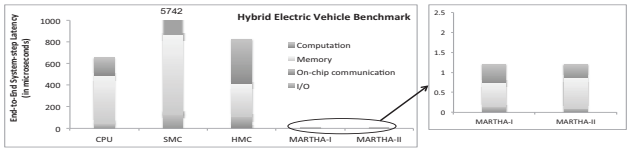
(a) End-to-end latency comparison on all five systems for the three benchmarks.



(b) End-to-end latency comparison on all five systems for the motor drive benchmark.

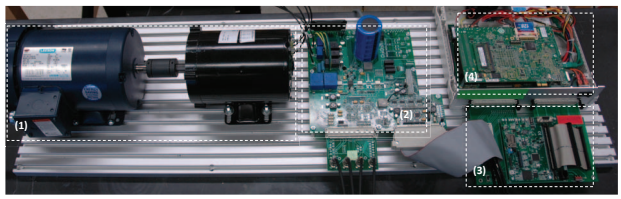


(c) End-to-end latency comparison on all five systems for the utility grid connected benchmark.



(d) End-to-end latency comparison on all five systems for the hybrid electric vehicle benchmark.

Fig. 5. Performance results of the various benchmarks on the different hardware platforms.



(3) Variable speed converter connected to a three-phase induction machine. (2) Inverter. MARTHA: (3) Controller (4) Real-time emulator

Fig. 6. Motor-drive experimental setup for parallel physical system and real-time digital emulator testing.

the model. Measured waveforms on both the real system and emulator on *MARTHA* have almost one-to-one matching. Figure 7 shows the side by side reading of the emulator and the physical system. Emulator response latency is measured to be less than (1.3μs).

B. Comparing MARTHA *to commercial emulators*

Table I compares the simulation time step, which includes input/output latency, computation time, and A-to-D and/or D-to-A conversion, of three real-time hardware digital emulators with our proposed platform. Note that OPAL's eDriveSim is electric car domain specific, whereas *MARTHA* is shown to perform well across all benchmarks.

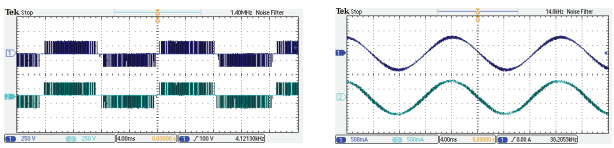


Fig. 7. Line-to-line inverter voltage and the motor phase current, emulator measurements shown on channel 1 and physical system on channel 2.

TABLE I
COMPARISON OF SIMULATION TIME STEP FOR SEVERAL STATE-OF-THE-ART HARDWARE EMULATORS FOR POWER ELECTRONICS.

HiL	NI cRIO	dSpace DS1006	OPAL eDriveSim	*MARTHA*
Time Step	1640μs	340μs	15μs	1.4μs

VII. CONCLUSION

Our main goal in this work is to highlight the modeling and the computational needs of power electronics systems, and how to model these systems as switched hybrid automata. We are able to derive from this modeling approach a hardware architecture where performance, programmability, and predictability are all first-class targets. We present a novel heterogeneous computation platform to enable the efficient execution of the control and emulation of next generation power electronics and smart grid systems. A prototype of the proposed architecture on an FPGA is deployed and tested in a physical environment. It enables a high-fidelity (with 1μs latency and emulation time-step), safe, and fully realistic testing and validation of detailed aspects of power electronics systems. To the best of our knowledge, no current academic or industrial HiL system, for hybrid systems, or power electronics systems, has such a fast emulation response time.

REFERENCES

[1] R. W. Erickson and D. Maksimovic, *Fundamentals of Power Electronics*, 2nd ed. Springer, 2001.

[2] F. C. Lee and et. al, "Power electronics system integration–a cpes perspective," *14th IEEE Power Electronics Society Newsletter*, 2008.

[3] V. Rajagopalan, *Computer-aided analysis of power electronic systems*. New York, NY, USA: Dekker, 1987.

[4] J. Kassakian, "Simulating power electronic systems-a new approach," *Proceedings of the IEEE*, vol. 67, no. 10, pp. 1428 – 1439, 1979.

[5] C. Graf, J. Maas, T. Schulte, and J. Weise-Emden, "Real-time hil-simulation of power electronics," in *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*, nov. 2008, pp. 2829 –2834.

[6] J. Belanger, V. Lapointe, C. Dufour, and L. Schoen, "emegasim: an open high-performance distributed real-time power grid simulator. architecture and specification," in *Proceedings of International Conference on Power Systems*, 2007.

[7] S. Abourida and J. Belanger, "Real-time platform for the control prototyping and simulation of power electronics and motor drives," in *Proceedings of 3rd International Conference on Modeling, Simulation, and Applied Optimization*, 2009.

[8] L.-F. Pak, V. Dinavahi, G. Chang, M. Steurer, and P. Ribeiro, "Real-time digital time-varying harmonic modeling and simulation techniques ieee task force on harmonics modeling and simulation," *Power Delivery, IEEE Transactions on*, vol. 22, no. 2, pp. 1218 –1227, april 2007.

[9] S. Stuijk, T. Basten, M. Geilen, and H. Corporaal, "Multiprocessor resource allocation for throughput-constrained synchronous dataflow graphs," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, june 2007, pp. 777 –782.

[10] W. Wolf, A. Jerraya, and G. Martin, "Multiprocessor system-on-chip (mpsoc) technology," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 10, pp. 1701 –1713, 2008.

[11] T. Geyer, F. Torrisi, and M. Morari, "Efficient mode enumeration of compositional hybrid systems," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, O. Maler and A. Pnueli, Eds. Springer Berlin / Heidelberg, 2003, vol. 2623, pp. 216–232.

[12] M. Senesky, G. Eirea, and T. Koo, "Hybrid modelling and control of power electronics," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, O. Maler and A. Pnueli, Eds. Springer Berlin / Heidelberg, 2003, vol. 2623, pp. 450–465.

[13] T. A. Henzinger, "The theory of hybrid automata." IEEE Computer Society Press, 1996, pp. 278–292.

[14] M. Kinsy, M. Pellauer, and S. Devadas, "Heracles: Fully synthesizable parameterized mips-based multicore system," in *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, Sept. 2011, pp. 356 –362.