Time-Predictable Computer Architecture for Cyber-Physical Systems: Digital Emulation of Power Electronics Systems

Michel Kinsy*, Omer Khan*†, Ivan Celanovic*, Dusan Majstorovic‡, Nikola Celanovic‡, Srinivas Devadas*

*Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology, Cambridge, MA, USA 02139
†Department of Electrical and Computer Engineering
University of Massachusetts, Lowell, MA, USA 01854
‡Faculty of Technical Sciences
Novi Sad, Serbia

Abstract—**The smart grid concept is a good example of a complex cyber-physical system (CPS) that exhibits intricate interplay between control, sensing, and communication infrastructure on one side, and power processing and actuation on the other side. The more extensive use of computation, sensing, and communication, tightly coupled with power processing, calls for a fundamental reassessment of some of the prevailing paradigms in the real-time control and communication abstractions. Today these abstractions are mostly thought of as embedded systems, and the overall framework needs to be reformed in order to fully realize the potential of the emerging field of cyber-physical systems.**

This paper details the design and application of a new ultra-high speed real-time emulation platform for Hardware-in-the-Loop (HiL) testing and design of high-power power electronics systems. Our real-time hardware emulation for HiL systems is based on a reconfigurable, heterogeneous, multicore processor architecture that emulates power electronics, and includes a circuit compiler that translates graphic system models into processor executable machine code. We present the hardware architecture, and describe the process of power electronic circuit compilation. This approach yields real-time execution on the order of $1\mu s$ simulation time step (including input/output latency) for a broad class of power electronics converters. To the best of our knowledge, no current academic or industrial HiL system has such a fast emulation response time. We present HiL experimental results for three representative systems: a variable speed induction motor drive, a utility grid connected photovoltaic converter system, and a hybrid electric vehicle motor drive.

Keywords-**Cyber-Physical Systems, Power Electronics, Multi-cores, Heterogeneous Architectures, FPGA, Network-on-Chip, Switched Hybrid Automaton, Hardware-in-the-loop (HiL).**

I. Introduction

Cyber-physical systems (CPS) refer to the integration of computational and physical resources [1], and unlike traditional embedded systems, CPS are more distributed and interactive in nature. Applications of CPS range from small-scale, safety critical, e.g., a pace maker controller, to large-scale, distributed, like a smart grid. While these system have great potential, they demand a fundamental reassessment of some of the prevailing paradigms in computation and communication abstractions.

The need to understand the dynamics and instabilities of the early electrical power grid motivated Vannevar Bush to develop the world's first, mechanical, real-time simulator, in the 1920's. This is one of the earliest examples of an advanced, reconfigurable, real-time scientific computational machine that later came to be known as MIT's Differential Analyzer [2]. Today, we are witnessing a new revolution that is poised to dramatically change the way our society generates, distributes and uses electrical energy, and energy in general. The push for green and sustainable energy future is fueling the drive for a new smart grid, larger penetration of renewable energy generation, hybrid and electric vehicles, more efficient energy utilization, and better solutions for large-scale energy storage. One of the key underlying physical layers of the smart grid is power electronics.

Power electronics is a class of cyber-physical systems. It comprises a real-time control layer, and power processing layer. The real-time control part is sensing physical variables (e.g., voltages, currents, speed, torque, temperature) based on which it calculates control signals (e.g., switch gate drive signals) for the power processing layer. The power processing layer consists of semiconductor switches (e.g., mosfets, IGBT's) and energy storage elements (e.g., inductors, capacitors) that jointly provide the function of modulating electric power parameters (e.g., voltage, current, frequency). Power electronics can broadly be defined as solid state energy conversion technology [3] 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 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% [4]. In order for power electronics to reach this point, advancements are needed in the area of *Electronic*
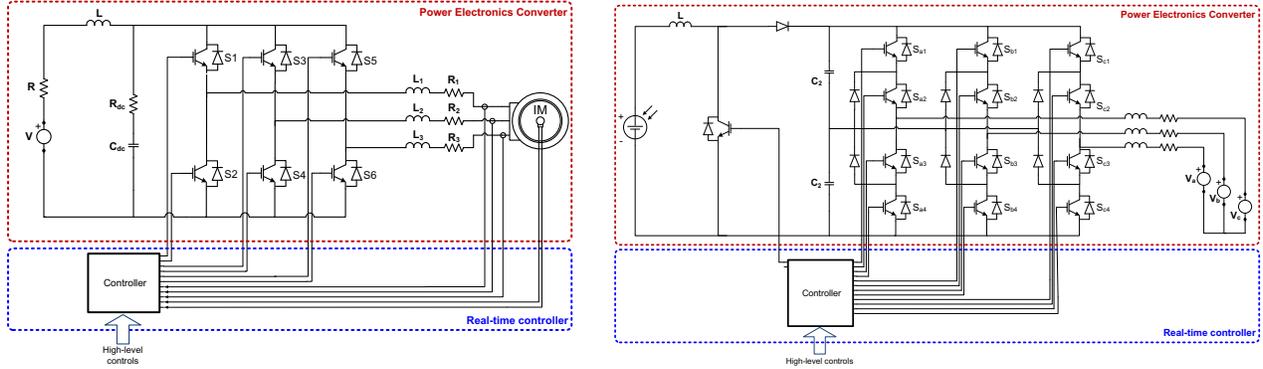
(a) Schematic diagram of a three-phase inverter driving induction machine, commonly used in electric and hybrid vehicles.

(b) Schematic diagram of a PV panel with boost converter and three-level three phase neutral-point clamped (NPC) inverter connected to utility grid.

Figure 1.   Power Electronics Conversion Applications.

Design Automation (EDA) tools, among others. 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 tool would immensely benefit the area of automated control testing and rapid design prototyping.

To illustrate power electronics conversion, consider two representative applications: induction machine drive for electric and hybrid car applications, and power electronics converter for photovoltaic applications. Figure 1(a) shows a schematic diagram of an electric car drive-train that comprises a power electronics converter and a tightly integrated real-time controller. The power electronics converter consists of six semiconductor switches (IGBT's with anti-parallel diodes), a DC-link capacitor, and a filter that controls position, speed and torque of the induction machine. The second example is a power electronics converter for photovoltaic applications that provides a dynamic power flow interface between the photovoltaic panel and the utility grid. This system is shown in Figure 1(b) and it comprises a power electronics converter and a real-time controller. Although the power converter has twice as many semiconductor switches, it has a similar overall structure as the induction machine drive.

In this paper, we propose a new software/hardware co-design *EDA* platform and a time-predictable, heterogeneous multicore architecture for comprehensive testing and validation of power electronics controller hardware, firmware, and software performance by means of high-fidelity, real-time emulation of power electronics in *Hardware-in-the-Loop (HiL)* configuration. In HiL testing configuration, the power electronics converter is replaced by our ultra-fast real-time emulator that interacts with the controller via high-speed physical input/output interfaces. From the controller point of view, the emulation should be seamless, i.e., there should be no distinction between the real-time control of the physical plant and the hardware emulator.

The paper is organized in five sections. Section II frames the challenges facing the design automation and testing of power electronics. Sections III, IV, and V describe the proposed framework and architecture for real-time hardware emulation. Section VI compares the fidelity of the proposed real-time emulator with the real power electronics system, and presents two more design examples. Section VII summarizes the paper.

II.   DESIGN REQUIREMENTS FOR POWER ELECTRONICS SYSTEMS

Today, power electronics engineers mostly rely on off-line simulations in the early design stage, and on hardware prototypes–low voltage simulators–in the final stages of the design and testing. However, off-line simulations are generally slow and only provide a certain level of functional faithfulness, especially when modeling an embedded controller and its interactions. Hardware emulation in *HiL* configuration enables more realistic testing while reducing the complexity and cost. The key benefits of real-time emulation for HiL testing are: (1) accelerated testing and validation; (2) reduced testing time needed in the laboratory; (3) simulation of all operating points and scenarios that are difficult or impossible to recreate with a real system (4) fault injection capability; and (5) real-time access to all signals that are difficult to measure and quantify in a real system.

The automotive industry has been using HiL simulation as an ubiquitous tool for testing the engine control unit (ECU). Connecting an ECU to a real-time simulator of a car enables exhaustive testing of the ECU. In computer engineering, hardware emulation, or in-circuit emulation, is used in processor design by using another system (hardware emulator) that imitates the behavior of the system under test. In power electronics, replacing the power converter, grid, and electromechanical components with a real-time digital emulation takes high-power hardware out of the testing environment with minimal loss of fidelity (from the controller

point of view). However, high-fidelity testing of power electronics requires a real-time emulation system with: (1) ultra-low latency and high-speed simulation ($\sim 1\mu s$); (2) ability to simulate switched and nonlinear systems; (3) flexible modeling; (4) ease of use, and (5) variable levels of modeling abstractions. Most HiL simulators, based on off-the-shelf processors, exhibit large latencies on the order of $50\mu s$. Yet, power converters operate at switching frequencies of ~ 10kHz, thus limiting the fidelity and applicability of HiL platforms based on the standard general-purpose processors.

III. PROPOSED MODELING APPROACH

Power electronics system design and verification require tightly coupled software and hardware environment. This arises from the fact that these systems can be complex, hybrid, and include analog hardware, synchronous and asynchronous discrete hardware, custom software, and computational intensive algorithms over a large amount of data. The real-time inputs and outputs required by these systems, for in-system testing critical to verification, impose further constraints on the software/hardware partition of the design flow, the mapping of circuit topology onto the hardware, and runtime computation and communication of the power electronics blocks. Mismatch at the software/hardware interface can severely reduce system flexibility, and can lead to degradation of performance in both throughput and latency. This interface also has a direct effect on the design flow process, for example, on circuit model specification. In other words, schematic editing of the power electronics systems is done offline, and falls under the software partition of the platform, which can be executed on a general-purpose computer. This also means that if the circuit topology changes or the characteristics of a component changes, e.g., inductance value for an inductor, then a new compilation and hardware mapping process must be initiated.

Power electronics simulation algorithms can broadly be divided into two main categories [5], [6]: (1) nodal analysis based (e.g., SPICE), and (2) state-space equation based (e.g., Matlab Simulink). Most modern power electronics simulation tools are based on the state space model formulation. State space modeling approach for power electronics circuits can be further divided into two subcategories [5]: fixed matrix, and variable matrix modeling approach. In the fixed matrix approach, all the non-linear elements (e.g., switching elements such as mosfets, diodes etc.) are modeled via nonlinear current-voltage elements which warrant uniform system description, i.e., fixed state-space matrix size. For example, when the switch changes the conduction state from *"off"* state to *"on"* state, the equivalent switch impedance changes the value from high impedance to low impedance during the switching time. On the other hand, in the variable matrix approach, switches are treated as ideal switches with instantaneous transition from one state to another (most commonly with piece-wise linear charac-

teristic). Hence, every combination of the switches in the circuit is described as a unique state-space representation. This leads to a system description with an N state space formulation where upper bound on the number of state-space representations is $N = 2^m$; m is the number of switches. This is the upper-bound on the number of state-space formulations since the circuit constraints will deem a number of circuit configurations not feasible due to inherent topological constraints. Our approach to high-speed low-latency simulation of power electronics circuits is based on a switched hybrid system framework, which on one side is naturally suited for the description of power electronics circuits, and on the other side naturally lends itself to digital architecture implementation [7], [8].

A. Switched Hybrid Automaton

The proposed switched hybrid system approach to power electronics converter modeling relies on piece-wise linear passive elements, piece-wise linear switches, current and voltage sources (both controlled and idependent). The switched hybrid system model is given in the state space form, as:

$$\dot{x}(t) = A_q x(t) + B_q u(t) \tag{1}$$

where $x(t) \in X \subset R^n$ is the continuous state space vector, $A_q \in R^{n \times n}$; $B_q \in R^{n \times m}$; and $u \in R^m$ is the input vector. Any discrete state of the circuit belongs to a finite set $Q = \{q_1..q_n\}$ 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 circuit. In this framework one can also define a mode, denoted m_q, where $q \in Q$, is the operation of the system defined by given state space $\dot{x}(t) = A_q x(t) + B_q u(t)$ and a given q.

Definition 1 (Switched Hybrid Automaton) A switched hybrid system is a collection $H = (Q, X, f, E, G)$ where: $Q = \{q_1..q_n\}$ is set of discrete states, $X \subseteq R^n$ is the continuous state space; $f : Q \mapsto (X \mapsto R^n)$ assigns to every discrete state a continuous vector field on X; $E \subseteq Q \times Q$ is a collection of discrete transitions; $G : E \mapsto 2^X$ assigns each $e = (q, q') \in E$ a guard.

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

$$\dot{x}(k+1) = A_{d(q)} x(k) + B_{d(q)} u(k) \tag{2}$$

$$y(k) = C_{d(q)} x(k) + D_{d(q)} u(k) \tag{3}$$

$$g(k) = C_{gd(q)} x(k) + D_{gd(q)} u(k) \tag{4}$$

where k is the discrete time, y is the output vector, g is the guard vector (it is treated as the output vector). In the discretized form, the set of matrices

(a) Schematic diagram of boost converter.

(b) Boost converter switching states representation.

(c) Block diagram of boost converter Discrete Hybrid Automaton.
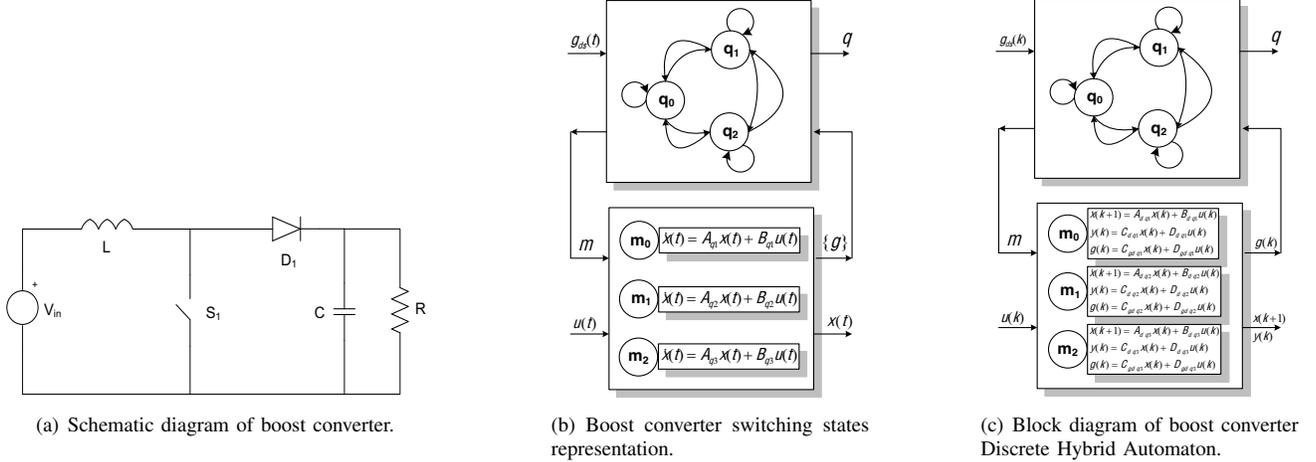
Figure 2. Hybrid Modeling of the boost converter.

$\{A_{d(q)}, B_{d(q)}, C_{d(q)}, D_{d(q)}, C_{gd(q)}, D_{gd(q)}\}$ define the dynamic behavior of the system.

B. Example Switched Hybrid Model of a Boost Converter

A standard boost converter is given in Figure 2(a), comprising of an inductor, capacitor, load resistor, DC voltage source, ideal diode and an ideal switch. Although there are four possible combinations for the switching states in practical systems, we will only consider the three as shown in Figure 2(b). The equivalent block diagram description of the Discrete Hybrid Automaton for the boost converter is given in Figure 2(c). The three combinations of the switches define a set of discrete states $Q = \{q_0, q_1, q_2\}$, which define three modes $\{m_0, m_1, m_2\}$. The modes can be defined in the following way:

Mode m_0

$$\dot{u}_c(t) = -\frac{1}{RC} u_c(t) \tag{5}$$

$$E = \{e_{00}, e_{01}, e_{02}\} \tag{6}$$

$$g_{00} : g_{ds} = 0 \wedge V_{in} - u_c > 0; \tag{7}$$

$$g_{01} : g_{ds} = 0 \wedge V_{in} - u_c > 0; \tag{8}$$

$$g_{02} : g_{ds} = 1 \wedge V_{in} - u_c > 0; \tag{9}$$

where u_c is the capacitor voltage, V_{in} is input voltage source, g_{ds} is the digital input gate drive signal controlling the switch S_1, $E = \{e_{00}, e_{01}, e_{02}\}$ is the collection of discrete transitions for the given mode m_0 and $G = \{g_{00}, g_{01}, g_{02}\}$ is the set of guards associated with a corresponding discrete transition from E.

C. Computational Complexity

The discrete Hybrid Automaton approach to modeling of power electronics circuits exhibit deterministic and time bounded 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 DHA 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.

In order to alleviate exponential growth of the problem, we propose an approach to partition the circuit into subcircuits that communicate via slower state space variables. Although we have devised a partitioning algorithm, the detail presentation of the algorithm is beyond the scope of this paper. For example, if a circuit is partitioned into two subcircuits with n and m switches respectively, the total number of modes will be $2^n + 2^m$ instead of 2^{n+m}. Once a circuit is partitioned, every partition is modeled as an independent *Discrete Hybrid Automaton*, while independent DHA's communicate via input-output variables.

IV. PROPOSED SOFTWARE ARCHITECTURE

The overall system architecture has a unifying layered design framework. Figure 3(a) illustrates this layered approach and shows the structure of our application decomposed into sets of subtasks where each set represents a particular level of abstraction. The software environment has three layers, the power electronics modeling layer, formulation and analysis layer, and finally the synthesis and simulation layer. Since the software platform is both general-purpose and embedded in nature, we focus on three key aspects, namely, abstraction,

(a) System Architecture Layers

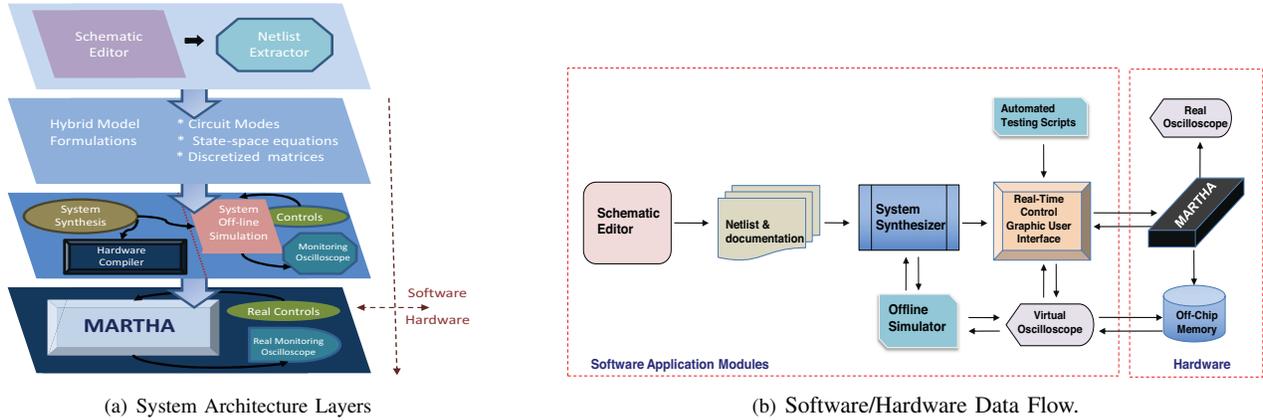(b) Software/Hardware Data Flow.

Figure 3.   EDA Software/Hardware System Architecture.

structure, and modularity, with the goal of an adaptable software system for power electronics design, simulation, and real-time emulation.

The first layer is made out of the schematic editor and the circuit netlist abstractor. The *Schematic Editor* gives designers the ability to convert their power electronics systems specification into model representations, where they can capture 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, 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 or blocks. Given a power electronics circuit, the *Netlist Extractor* generates the associated list of components used, their values, and connectivity. It also performs a node annotation of the circuit, which is essential for model analysis. The netlist extractor also provides circuit metadata, such as the hierarchical composition and degree of idealization of components. The hybrid model formulation is done through the *System 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 either through the offline simulator, or through the real-time controller to be mapped onto the the hardware. The *Offline Simulator* simply takes the state space equations, transition equations, and control functions (both preset and actively user controlled) and emits via the *Virtual Oscilloscope* the corresponding transient output signals.

The *Real-Time Control Graphic User Interface* performs the mapping of the synthesis of the power electronics system onto the hardware, depending on switching blocks and component blocks signal communications. This mapping consists primarily of loading state-space equation matrices and transition data to memory, emulation execution bit files onto cores, and on-chip network configuration bits. This software control unit allows for both automated scripts and the user to control in real-time the hardware emulation of the power electronics systems under test. Figure 3(b) illustrates the data flow in the software, and intermodule communications. (We use the C++ programming language for data manipulations and Python for scripting and graphic user interfaces.)

V. PROPOSED HARDWARE MULTICORE ARCHITECTURE

For power electronics, traditional approaches using off-the-shelf single core or multicore general-purpose processor or FPGA's (Field Programmable Gate Array), are unlikely to meet such demands. Indeed, generic general-purpose architectures use deep pipelines, complex memory hierarchy, and non-deterministic operating systems which often fail to meet the hard real-time constraints. On the other hand, FPGA's lack coarse-grained design abstractions and force designers to become experts in digital hardware design, and to spend significant design time building and verifying the hardware platform on which to emulate or verify their power electronics design.

As an alternative, we propose and develop *MARTHA* (*Multicore Architecture for Real-Time Hybrid Applications*), a heterogeneous, reconfigurable multicore architecture, designed to meet the high computation throughput, deterministic, and low-latency requirements for a spectrum of cyber-physical systems (e.g., power, electronics, biomolecular networks, mechanical systems, financial systems). We implement *MARTHA* on an FPGA providing an easier programming abstraction to the user.

MARTHA has the following processing structures for computation:

(a) Datapath of the Vector Processing Core.
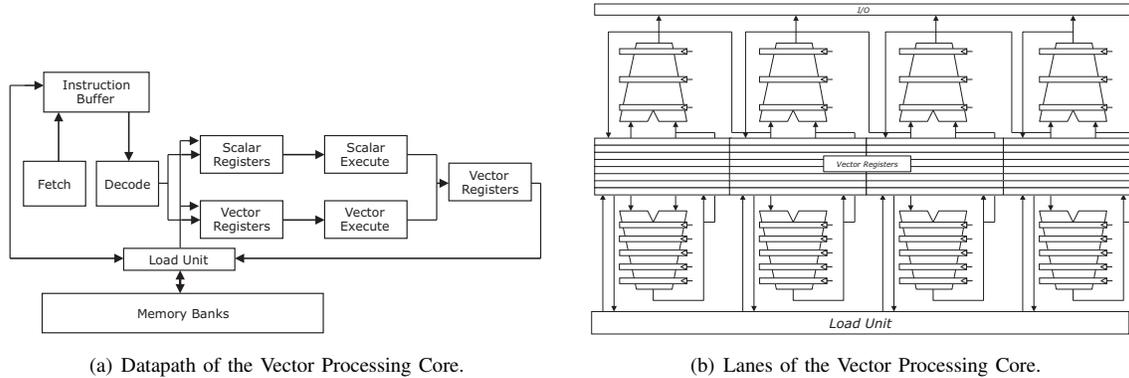

(b) Lanes of the Vector Processing Core.

Figure 4. *MARTHA's* Vector Core.

- Vector machine style core(s): used to model linear dynamics of power electronics, with fast, parallel, matrix manipulation operations.
- General-purpose MIPS-core(s): used for mode transitions, controls, monitoring, and data collection.
- DSP cores: used for I/O conditioning, analog/digital communication, and for non-linear machine models.
- Programmable core: employed to model certain signal sources, and to implement on-chip control functions.

Interleaved memory structure is used to increase the data transfer rate between the main memory and computational resources. These cores communicate via shared memory and a mesh network of dedicated registers, where routes are pre-computed, configured, and are static for each system under test. This interconnect network architecture is tailored for deterministic and ultra-low latency. Using these processing structures, the proposed architecture may be realized with varying level of parallelism and connectivity, depending on the computational complexity and distribution of the class of CPS one wants to support. This heterogeneity in the architecture is completely dictated by the complex interactions between software and hardware, hardware and I/O interfaces, analog and digital signals, and real-time controls and digital monitoring, as seen in power electronics. Our current implementation is optimized for real-time emulation, prototyping, estimation of power electronics systems and controllers. It is flexible enough to cover a wide range of power electronics systems, from a simple buck converter to a multi-level converter.

A. Vector Core Architecture

Real-time embedded systems require a significant amount of fast computation. One of the most common, but computationally expensive operations in these systems is matrix vector multiplication (MVM). From a hardware design view, matrix operations exhibit the sort of fine-grained data parallelism and computational predictability that works well with the computer architecture of the Single Instruction, Multiple

Data (SIMD) class, in particular, vector processor machines. The advantages of such an architecture are well documented in [9], [10].

Vector processors are characterized by vector registers, vector instructions, hardwired control, and interleaved (bank) memory system. The vector processing cores in *MARTHA* are RISC style with vector operands, and the instruction set architecture (ISA) is based on the industry standard MIPS scalar ISA [11] extended to support vector operations. For most power electronics systems design and testing, full-precision floating-point is not needed. Therefore our vector processing units are optimized for fixed-point computations.

Figure 4(a) illustrates the datapath of the core. Each vector register per core is 8-entry, 128-bit, divided into four 32-bit parts for the four lanes (Figure 4(b)). There is also an 8-entry 32-bit scalar register unit used for scalar or scalar-vector operations. The vector length register (VLR) holds the number of elements in a vector operation. In the absence of a cache, the instruction buffer is used to stream in multiple instructions at the same time from memory. The *Load Unit* serves as the main control logic to the memory. It takes care of the addressing of the memory, using a base and a stride, to automatically load vector instructions, and to initiate N loads or stores over multiple memory banks.

The most important operation in this functional unit is the address generation. The vector stride is simply the number of memory locations separating elements that are merged to form a single vector operation. Our vector core also supports chaining, which allows for the result of one operation to be made immediately available to the next operation without passing through the vector registers. This is particularly useful for fast *multiply-and-accumulate* operations in matrix multiplication. For conditional operation on the elements of a vector, we use vector masks. The vector mask is loaded with a vector test instruction and the vector operation executed only on the elements whose the corresponding entry in the vector mask is set to 1.

Each lane has a 6-stage multiplier chained to a 3-stage

accumulator. For sparse matrices, the general vector machine operation of *scatter-gather* is used. The scatter operation consists of storing the vector elements across multiple banks using the index vector. The gather operation consists of taking an index vector and getting the vector elements at the addresses obtained by adding the base-address and the index vector offsets.

B. General Purpose MIPS-core Architecture

The MIPS (Microprocessor without Interlocked Pipeline Stages) processing unit in MARTHA consists of an integer-based 7-stage Core. MIPS is a register-based RISC architecture widely used in commercial products and for teaching purposes [12]. Our implementation is very standard and configured to run a large class of embedded software. It is a fully-bypassed, single-issue, in-order pipelined architecture, with no branch predictor or branch delay slot, running MIPS-III instruction set architecture (ISA) without floating point.

C. Digital Signal Processing (DSP) Core Architecture

The Digital Signal Processing (DSP) unit is used for signal domain conversions. Our DSP core can handle an array of signal types: audio and speech signal processing, sonar and radar signal processing, geological and biomedical signal processing, etc. It is standard and built around a finite impulse response (FIR) filter. Although similar to the MIPS unit in microarchitecture, it is less generic and designed with signal processing in mind. As such, it has extra multipliers and pipelines to allow for faster signal processing.

D. Memory System Organization

The emphasis in traditional general-purpose, multilevel memory systems is to minimize the average data access time, to improve overall system performance. In contrast, power electronic emulation, like most CPS, require predictable and repeatable execution time.

In our initial design, we propose instruction and data caches, and instruction fetch and data memory access take two cycles. Instruction address is issued at *I-Fetch 1* stage and on a cache hit, the actual instruction appears in the *I-Fetch 2* stage. Stall and bypass signals are modified to support the extended pipeline. Data memory (cache) is implemented over *D-Memory 1* and *D-Memory 2* stages. For a read, the memory address is presented to the cache in the *D-Memory 1* stage and the data on a cache hit appears in the *D-Memory 2* stage. On a memory write, we also check in the *D-Memory 2* stage that there is a cache hit before continuing execution. Instructions are issued and executed in-order, and the data memory accesses are also in-order. Although having caches does improve the overall average data access, it does decrease system predictability. Furthermore, when the system switches discrete state, matrices associated to the continuous mode change, caches need to be flushed, which makes the system cache-hit rate close to 70%.

For the proposed *MARTHA* architecture, we adopt a cacheless memory system with multiple concurrent accesses to hide the fixed data access time. *MARTHA* features bank-based or interleaved flat memory system, where the on-chip and off-chip memory organization is fully exposed. The uniform access time of each memory data makes the performance highly predictable and reasonably fast for these classes of CPS. *MARTHA* has a wide main memory, with independent memory banks. There are 8 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 resilience to bank failure. 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 our current implementation, each bank is $8MB$.

There are two modes of communication between processing units and the main memory in architecture, namely, direct bus-based communications and indirect network-based communications, both with fixed latencies. Memory traffic consists of packets, where a packet is composed of an address, a data block (of one or more words), and some control bits. The control bits in *MARTHA* consist of the encoding of the source and destination, number of words in the packet, and error checking bits.

This cacheless memory structure also allows us to completely bypass the expensive and complex hardware logic involved in providing cache-line replacement and coherence policies. Furthermore, the data access time is deterministic for all data communication in the system. For a given static traffic route, the data access time is simply a function of the memory in/out time; network core sending time; path length (number of routers traversed); routing time (in the current architecture is 1 cycle), and network core receiving time.

E. Interconnect Network

Traditionally, systems-on-chip (SoCs) use buses to establish communications between the different on-chip components, but due to the lack of scalability of wired connections between components, network-on-chip (NoC) architectures have been introduced as an effective data communication infrastructure [13], [14]. Therefore to provide scalability, *MARTHA* uses NoC architecture for its data communication.

Although virtual channels are generally used at the router, they also introduce complex buffer allocation logic and arbitration among buffers to access physical links, which lead to non-deterministic data transfer delays. For power electronics applications multicore architectures, where non-deterministic communication delays are not desired, a bufferless routing scheme seems more appropriate, and our system results reflect that. The router for *MARTHA* is bufferless with a very simple logic consisting of 4-

to-1 multiplexers with, per application, pre-configuration selectors.

One important aspect of NoC design is the interface through which the processing units are connected to the network. This aspect of the design deals with the conversion of data traffic coming from the processors, or microprocessors, into packets that can be routed inside the Network-on-chip, and the reconstruction of packets into data traffic at the opposite side when exiting the NoC. Since the focus of this design is on power electronics with real-time constraints and fast communication, we adopt a very simple register-based interface.

The interface and communication protocol rely on the compiler to map out routes, and cores use dedicated registers to communicate. The proposed design has eight registers − four outgoing registers and four incoming registers, one pair per neighbor. This interface logic is found in all the processing structures (e.g., General purpose MIPS core(s), Vector core(s), DSP core(s)).

F. On-Chip Data Routing

The routing algorithm in *MARTHA*, static and offline, determines routes considering the cyber-physical application's communication characteristics, in this case, the power electronics topology. Our main challenge is a fair and an effective tradeoff between load balancing in the network, to avoid congestion and dropping of packets, since routers are bufferless, and communication latency minimization, since these applications are real-time and demand a bounded time-step. To that end, *MARTHA*'s oblivious routing scheme uses a function corresponding to minimizing the maximum channel load (MCL) across all network links, while providing a mechanism for controlling the average path length. The core premise of this approach is to efficiently balance network load, while having the shortest possible path lengths and keeping the router architecture simple and fast. Our algorithm, based on [15], exploits knowledge of estimated bandwidths for all or a subset of data transfers in a given power electronics topology, and focuses on optimizing satisfaction of bandwidth demand and latency of individual data transfers. This scheme is livelock and deadlock free, producing routes that can be minimal or non-minimal in an effort to globally optimize the real-time application throughput fidelity. Once routes are determined, they are pre-configured by setting the proper selector bits at each router.

This routing table is set to allow multiple concurrent data transfers through a router as long as there is no sharing of physical links. Figure 5 shows an example of possible routes allowed. Note that two types of communications are supported in the *MARTHA* architecture, one-to-one and one-to-many. We disallow many-to-one due to potential conflicts and the fact that there is no arbitration or buffer at the routers.
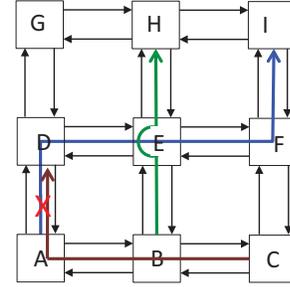

Figure 5. Network Possible Routes Configurations.

VI. Results and Evaluation

To demonstrate the fidelity and the versatility of our full electronic systems design and automation approach, and our integrated software/hardware system solution, we present Hardware-in-the-Loop simulations of three representative systems: namely, a variable speed induction motor drive, a utility grid connected photovoltaic converter system, and a hybrid electric vehicle motor drive.

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 described below, we also model a *software-only* baseline architecture (CPU).

A. Hardware Models and Methods

With the computation units described in Section V, we construct four variations of the proposed architecture. In the first system configuration, we have a *Single MIPS Core* (SMC) to execute the application and a DSP unit for I/O communication with the board simulating the environmental variables. The second configuration has an *Homogenous mesh network of eight MIPS Cores* (HMC) with two DSPs. The third system configuration, called *MARTHA-I*, shown in Figure 6, comprises:
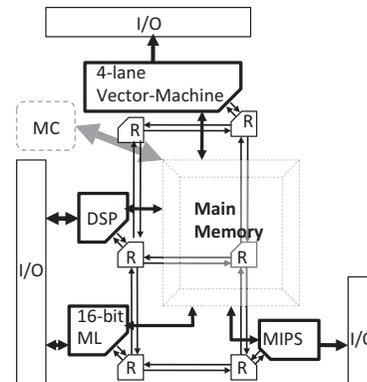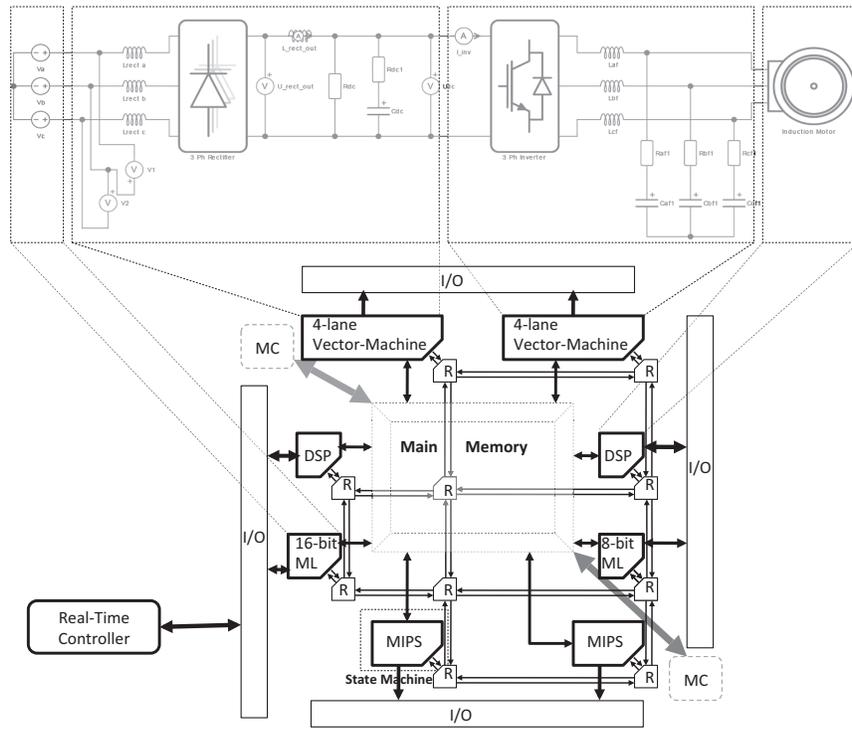

Figure 6. *MARTHA-I* Architecture.

Figure 7. *MARTHA-II* with Motor Drive Model Mapping.

- one RISC MIPS core,
- one four-lane chained vector core for fast and parallel matrix computation,
- one digital signal processor (DSP),
- a 16-bit programmable microprocessor,
- one memory controller for off-chip communication, and
- bank style main memory.

The fourth configuration, called *MARTHA-II*, as shown in Figure 7, comprises:

- two RISC MIPS cores,
- two four-lane chained vector cores for fast and parallel matrix computation,
- two digital signal processors (DSP),
- 16-bit programmable microprocessor,
- 8-bit programmable microprocessor,
- two memory controllers for off-chip communication, and
- bank style main memory.

MARTHA-II is intended to show the scalability of the *MARTHA* architecture, and how subsystem-parallelism can be supported. In addition to the computation and communications structures, Figure 7 shows the partitioning and mapping of the variable speed motor drive benchmark application onto *MARTHA-II*.

The proposed *MARTHA* and baseline (SMC and HMC) architectures are designed using verilog register transfer language (RTL) and synthesized using Xilinx ISE Design Suite 11.5, with Virtex-5 LX330T as the platform board. On the board, the 7-stage MIPS core, for example, runs at 162.5 MHz, and uses 1.3% 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 x 18 bit multiplier). The DSP block uses 12% of the board DSP48E resources.

We compare the end-to-end execution time for our power electronics applications on all five system configurations. Figure 8 shows the end-to-end *system-step* latency for the different platforms. For the hardware-in-the-loop testing, we use the *MARTHA-II* configuration.

B. Variable Speed Induction Motor Drive

The variable speed drive consists of a three-phase voltage source, a three-phase diode bridge rectifier, a DC-link, a three-phase two-level voltage-source inverter, an output filter and an induction machine. To illustrate the fidelity of our real-time digital emulation platform, we run in parallel an industrial grade open-loop variable speed drive, ABB ACS-150, and our real-time hardware emulator, both driven from the same controller. Figure 9 shows the experimental setup,
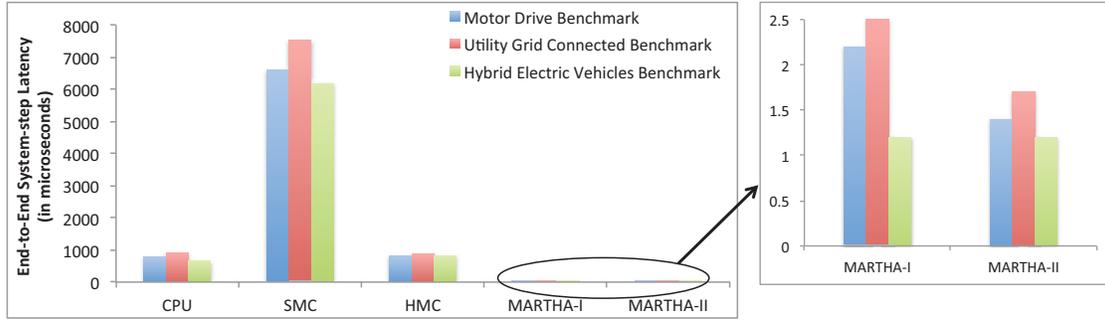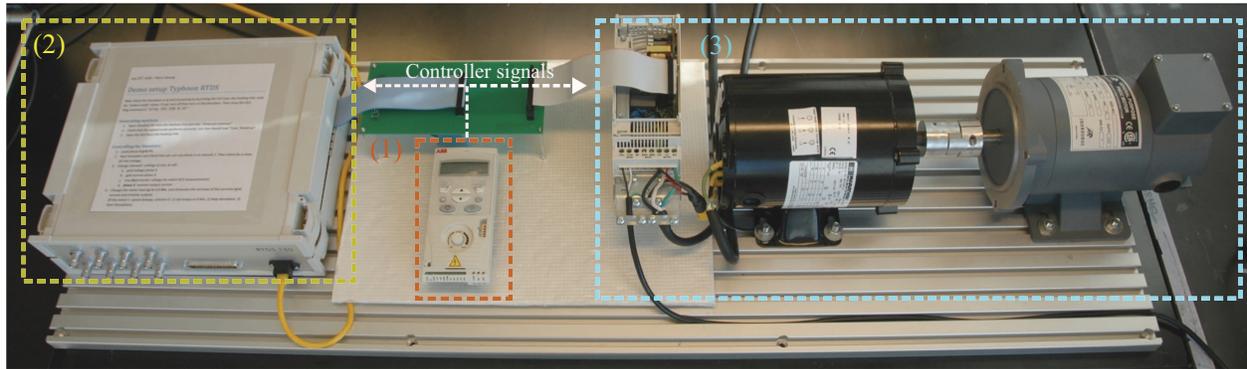
Figure 8.   End-to-End Latency Comparison on all five systems for the three power electronics applications.

(2) HIL platform simulating variable speed drive connected to three-phase

(1) ACS-150 variable speed drive controller. Device-under-test (DUT)

(3) Variable speed drive converter connected to three-phase induction machine.

Figure 9.   Motor-drive experimental setup for parallel physical system and real-time digital emulator testing.

where the same controller is used for both the real system and the real-time emulated system.

Figure 10 shows the system comprising the three-phase diode rectifier and three-phase two-level IGBT inverter. Measured waveforms on both the real system and the emulator are shown in Figure 11(a) with almost one-to-one matching. Emulator response latency is measured to be less than $1.3\mu s$, as shown in Figure 11(b). The ACS-150 is a 0.5 horsepower, three-phase, 240 volt inverter, with a switching frequency up to 16kHz.

C. Utility Grid Connected Photovoltaic Converter System

To illustrate, that beyond the real-time high-fidelity, our EDA environment also provides support for flexible modeling, design, prototyping, and HiL testing for a wide range of applications, we present another example system: a grid connected photovoltaic converter system. Figure 12(a) shows a two-stage utility grid connected photovoltaic converter model, that consists of a photovoltaic panel module, boost converter, a three-level NPC inverter, an input filter, and a grid. Figure 12(b) shows voltages V_{ab} and V_{bc} on channels 1 and 2, respectively.

D. Hybrid Electric Vehicle Motor Drive

In recent years, we have witnessed a significant increase in the development efforts focused on hybrid and electric vehicle drivetrains. Being a safety critical application, these system require ever increasing attention to testing and verification of power electronics designs. Standard vehicle HiL simulators can provide good fidelity for mechanical systems while lacking speed and latency to simulate power electronics subsystems. In this example, we present HiL simulation, based on the proposed platform, of a typical electric vehicle drivetrain with an induction machine, two-level voltage source inverter, and battery as shown in Figure 13(a). High-fidelity HiL simulation enables engineers to observe all the fine details, e.g., DC-link and battery current, induction machine currents, and even common mode voltages.

Figure 13 shows a simplified voltage-source inverter model of the system, with switching frequency of 4 kHz. Figure 8 shows the end-to-end *system-step* latency for the different platforms. Results presented for these three applications are for the *MARTHA-II* configuration.

314
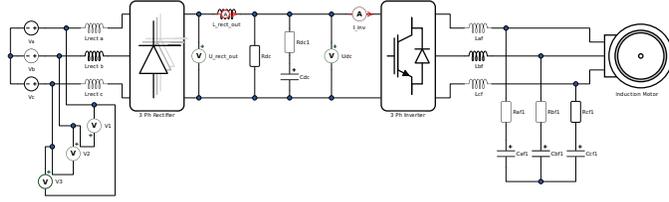
(a) Schematic diagram of the motor drive model of a three-phase rectifier, two-level voltage source inverter, filter and induction machine.
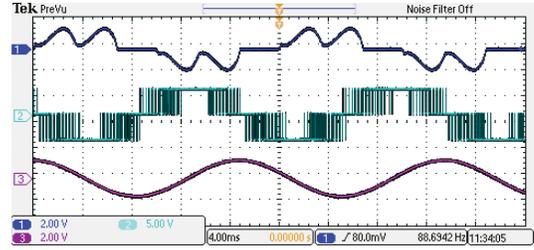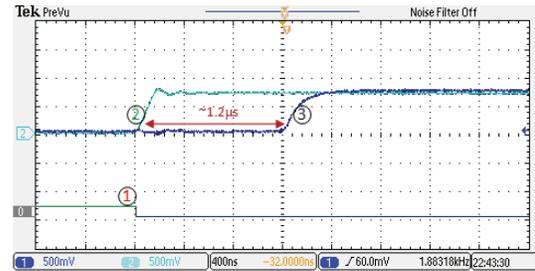

(b) Experimental measurements of the real-time emulation system rectifier input current, line to line inverter voltage, and stator current (channels 1, 2, and 3).

Figure 10. Variable Speed Induction Motor Drive Application Experimental Results.
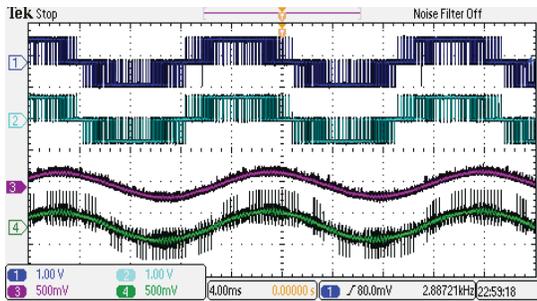
(a) Experimental measurements of the line-to-line inverter voltage and the motor phase current of the physical system are shown on channels 2 and 4, and the matching real-time hardware emulator waveforms are on channels 1 and 3.

① Digital gate drive signal
② Physical system (ACS150) line-to-line voltage response
③ Real-time digital emulator line-to-line voltage response

(b) Three-phase inverter line-to-line voltage response on the physical system, and real time emulation system to a gate drive signal.

Figure 11. High-Fidelity Emulation of Variable Speed Induction Motor Drive Application.

VII. CONCLUSION

In this paper, we show an example of how using a combination of old and new computer architecture techniques, one can design a new class of heterogeneous computation platforms to support the emerging field of cyber-physical systems. Our main goals in this presentation are to highlight the modeling and the computation needs of power electronics systems, a class of CPS, to show how these systems are intrinsically parallel and distributed in nature, and argue that they require heterogeneous architectures for fast emulation.

In addition, the proposed platform brings a true emulator performance to the control design and test engineer's desk. It enables a high-fidelity (with $1\mu s$ 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] E. A. Lee, "Cyber physical systems: Design challenges," in *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, May 2008, invited Paper. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/427.html

[2] L. Owens, *Vannevar Bush and the differential analyzer: the text and context of an early computer*. San Diego, CA, USA: Academic Press Professional, Inc., 1991. [Online]. Available: http://portal.acm.org/citation.cfm?id=132180.132183

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

[4] F. C. Lee and et. al, "Power electronics system integration– a cpes perspective," *14th IEEE Power Electronics Society Newsletter*, vol. 20, 2008.

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

[6] N. Mohan, W. Robbins, T. Undeland, R. Nilssen, and O. Mo, "Simulation of power electronic and motion control systems-
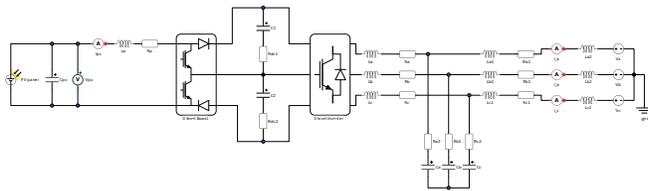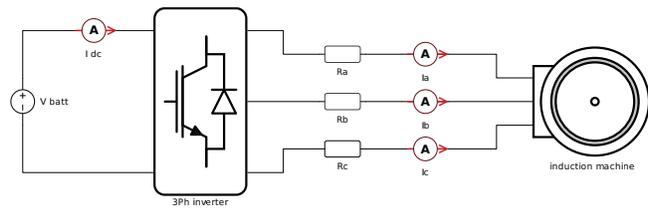
(a) Schematic diagram of the PV panel, boost converter, three-level voltage source inverter, filter and utility grid.
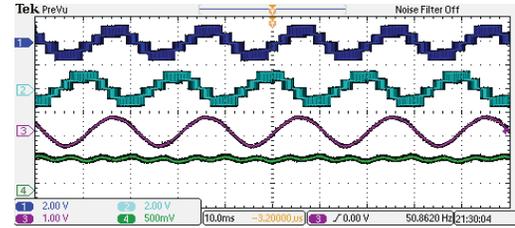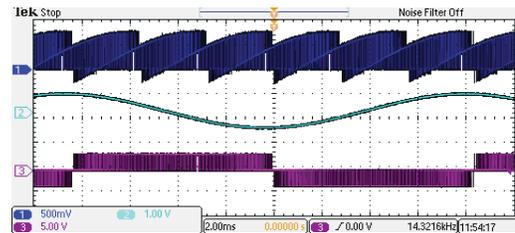

(b) Experimental measurements of the real time emulator variables, line to line inverter voltages, output current, and upper DC-link voltage.

Figure 12.   Utility Grid Connected Photovoltaic Converter System Application.

(a) Schematic diagram of the hybrid electric vehicle motor drive with battery model, two-level voltage source inverter, and induction machine.

(b) Experimental measurements of the real time emulation system waveforms, DC link current, rotor current, and inverter output line-to-line voltage.

Figure 13.   Hybrid Electric Vehicle Motor Drive System Application.

an overview," *Proceedings of the IEEE*, vol. 82, no. 8, pp. 1287 –1302, Aug. 1994.

[7] 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.

[8] 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.

[9] M. Dowd, "An example risc vector machine architecture," *SIGARCH Comput. Archit. News*, vol. 16, pp. 91–99, March 1988. [Online]. Available: http://doi.acm.org/10.1145/44571.44579

[10] K. Asanovic, "Vector microprocessors," Ph.D. dissertation, University of California, Berkeley, 1998.

[11] G. Kane and J. Heinrich, *MIPS RISC architectures*.   Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[12] D. Patterson and J. Hennessy, *Computer Organization and Design: The Hardware/software Interface*.   Morgan Kaufmann, 2005.

[13] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," in *Proc. of the 38th Design Automation Conference (DAC)*, Jun. 2001. [Online]. Available: http://citeseer.ist.psu.edu/dally01route.html

[14] A. Ivanov and G. D. Micheli, "The Network-on-Chip Paradigm in Practice and Research," *Design & Test of Computers*, vol. 22, no. 5, pp. 399–403, 2005.

[15] M. A. Kinsy, M. H. Cho, T. Wen, E. Suh, M. van Dijk, and S. Devadas, "Application-aware deadlock-free oblivious routing," in *Proceedings of the 36th annual international symposium on Computer architecture*, ser. ISCA '09.   New York, NY, USA: ACM, 2009, pp. 208–219.