A High Performance Architecture for Real Time Power System Simulators Based on FPGA Hardware Acceleration

Julio C G Pimentel, IEEE Senior Member
Laval University
Dept. of Electrical and Computer Engineering
Pavillon Adrien-Pouliot, Ste-Foy, QC G1K7P4
Canada
pimentel@ieee.org

ABSTRACT
Previous generation of FPGA devices were neither big enough nor fast enough to allow them to compete with specialized microprocessors. This situation changed in the last five years with FPGA vendors offering devices with more than 5 million gates and running at clock frequencies higher than 400 MHz. These high density FPGAs can now be used to implement custom digital systems (System-On-Chip – SoC), including systems with single and multiple processors (Multiprocessor-Systems-On-Chip - MpSoC) or a combination of both. Taking advantage of the fast development of FPGA devices, academic researchers and industrial R&D groups started using them as a mean to speed up real time simulation of power systems. Here we present a digital real time power system simulator ("DRTPSS") which is solely based on FPGA devices. Though the methodology was initially developed for power system simulation, it has broader application as we demonstrated in [7].

KEY WORDS
Power System Simulation; Real-Time Simulator; Hardware Emulation; Reconfigurable Computing

1. Introduction

DRTPSSs have gained acceptance as an important tool to study complex electrical transients on power systems. However, despite all the research done in the last ten years on new parallel processing architectures and algorithms, the minimum time-step has decreased from 50 μs to around the 20 μs achieved by a current state-of-the-art DRTPSSs such as the one presented in [3]. The processor-memory hardware paradigm (Read-Modify-Write) and the inter-processor communication overhead experienced with the present technology create a bottleneck making it very difficult to further reduce the minimum timestep.

It was only recently that the new advances in the IC technology allowed the development of bigger and faster devices reducing the gap between FPGAs and high performance microprocessors. Presently, FPGA vendors offer devices with more than 5 million gates and running at clock frequencies higher than 400 MHz [13]. Also recently, there has been a lot of interest by research institutions as well as some industrial companies in studying how FPGAs can be used to develop faster DRTPSSs [1][5]. These works have used FPGAs as coprocessors but most of the arithmetic processing is still done by microprocessors interconnected in a network.

This paper, on the other hand, presents an architecture entirely based on high density FPGA devices which can implement very high performance DRTPSSs capable of simulating small and medium size power networks with very high natural frequency. To be able to achieve such high performance the simulation algorithm is mapped directly into hardware following a hardware acceleration approach as described here. The FPGA simulator can also be integrated to a parallel processor based on a cluster of PCs following a hardware and software co-design approach [4]. This way, part of the simulation algorithm runs in software in the parallel processor and part of the algorithm runs directly in hardware. By combining these two approaches, the hybrid real time simulator can simulate high frequency networks as well as very large networks.

2. Overview of the Subject

Worst case execution time (WCET) is a key metric in evaluating hard real-time systems. It is well established the need for an efficient allocation technique for data memory in order to minimize a task's WCET. Also, any data exchanged between the processor and main memory uses the memory bus, sharing it with data exchanged between I/O devices and main memory. If the processor and a device try to transfer data at the same time, an impact can be seen on the processor as well as on the device. As a result, the execution time of an application on the processor may increase due to the memory-bus load generated by I/O devices [10]. Furthermore, the throughput and latency of the communication interface is at least as important as the first two. Real time power system simulation does not require extremely high bandwidth as some others applications. However, it is

very sensitive to high latency. Table I summarizes the throughput of some of the most popular high performance communication interfaces used in modern cluster and grid computers. As we can see in Table I the smallest measured latency is around 5 µs what sets a minimum for the time step [10] [11] [12] of any simulator that relies on such communication interfaces.

TABLE I
THROUGHPUT AND LATENCY OF COMMUNICATION I/F

Interface Type	Throughput (Mbps)	Latency (µs)	Packet Size (bytes)
IEEE1394-400	280	> 17	--
1 GigE	300	> 50	1024
10 GigE	7000	> 10	1024
Myrinet-2000 (*)	4000	> 10	--
Myri-10G (*)	20000	> 5	--
Infiniband 1X	3800	> 5	1024
Infiniband 4X	6500	> 10	1024

(*) Theoretical performance

Recently, academic researchers and industrial R&D groups have proposed the use of FPGAs as a mean to alleviate these bottlenecks [1][4][5]. FPGAs are programmable devices containing distributed resources (memory, data processing and logic) interconnected by a huge switch matrix capable of transferring data at very high speed and at extremely low latency. Therefore, they are not subjected to the same limitations as parallel processor based DRTPSSs. Table II summarizes the amount of resources for 4 examples of Xilinx Virtex FPGA devices [13].

The works presented in [1], [4] and [5] show that hybrid DRTPSSs using microprocessors and FPGAs can achieve higher performance than microprocessor-only ones. The reported results demonstrate that these hybrid simulators can achieve a time step of 2 µs which is one order of magnitude smaller than the 20 µs attained with the traditional parallel-processor based ones. However, these works still use FPGAs as coprocessors but most of the arithmetic processing is still done by microprocessors interconnected in a network. In [2], the authors present an effort to develop a fully FPGA based DRTPSS. The proposed method is based on the generalized algorithm developed by Dommel for time domain simulation of transients in networks. Separate units are implemented on the FPGA chip, each one of these units is dedicated for the evaluation of a single equation representing the equivalent resistances and current sources standing for the previous states of the element. Though the reported results show the simulator can achieve very high performance on a sample single-phase test network, it is unclear how the method can be extended to more complex networks. Herewith, we present a fully FPGA-based DRTPSS that can simulate a wide class of networks and can achieve a time step smaller than 0.3 µs.

TABLE II
INTERNAL RESOURCES OF VIRTEX II AND VIRTEX 4 FPGAS

	Virtex II Pro		Virtex 4	
Ressource	2VP30[a]	2VP100[b]	4VLX40[a]	4VLX200[b]
# of logic cells	30816	99216	41472	200448
4-input LUTs	27392	88192	36864	126336
mult 18x18	136	444	N/A	N/A
DSP48	N/A	N/A	64	96
block RAM (kb)	2448	7992	1728	6048
Power PC	2	2	N/A	N/A
Max freq (MHz)	400	400	500	500

[a]Medium size of a Virtex family device [b]Largest FPGA of the family

3. High Level Architecture

3.1 Task Partitioning

The power network is decoupled in one linear sub-circuit including all linear elements and many nonlinear sub-circuits being one per nonlinear element as shown in figure 1. In cases where the linear sub-circuit contains very different natural frequencies it can be decoupled even further in smaller networks. These decoupled sub-circuits are concurrently simulated using a global timestep or using multiple timesteps depending on their natural frequencies.

Additional pairs of voltage-current sources are introduced at every linear-nonlinear interface to link the decoupled networks as shown in figure 2. Then, the linear and nonlinear sub-circuits can be concurrently simulated in hardware (or the linear sub-circuit can be simulated in software as presented in [4]) using its own simulation time step. Obviously, each sub-circuit is missing information about the state of the others sub-circuits so that their simulation results should be linked in some way. At the end of each timestep, the decoupled networks exchange information through the voltage-current sources. Some others solutions exist for the partitioning of power networks for real time simulation but space limitations do not allow us to address them here [2][3][6]. The simulator processes linear and nonlinear sub-circuits in different way as explained below.

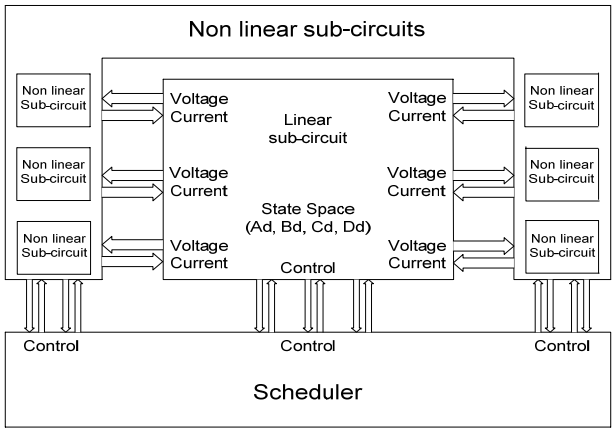

Figure 1. FPGA DRTPSS High Level Architecture

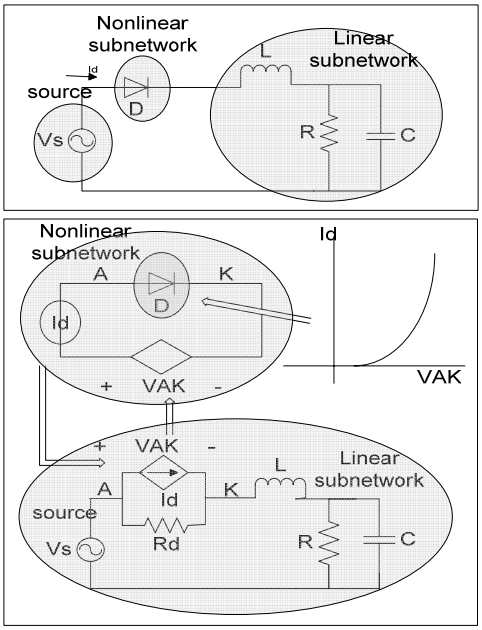

Figure 2. Partition into linear and nonlinear sub networks

3.1.1 The linear sub network

The RLC part of the electrical network is represented by its continuous-time state space model as:

$$\dot{X}(t) = A \bullet X(t) + B \bullet U(t), \quad (1)$$

$$Y(t) = C \bullet X(t) + D \bullet U(t), \quad (2)$$

where: X and $\dot{X}(t)$ are the state and its derivative
U is the input stimulus vector
Y is the output signal vector
and A, B, C and D are the state space matrices

The bilinear transform is then used to map the continuous-time state space into its discrete form below which is used for simulating the linear sub network, where n is the time step number and Ad, Bd, Cd and Dd are the discrete state space matrices.

$$X(n+1) = Ad \bullet X(n) + Bd \bullet U(n), \quad (3)$$

$$Y(n+1) = Cd \bullet X(n) + D \bullet Ud(n), \quad (4)$$

The next state and the outputs of the linear sub-circuit are computed by a state space solver VHDL module that exploits the characteristics of the state space formulation to allow an efficient mapping to the FPGA resources and its high level architecture is shown in figure 3. It includes a MAC unit, a blockRAM memory bank configured as 512 entries x 32 bits, multiplexers and a state machine. The MAC units can run at a minimum clock frequency of 150 MHz with a latency of 6 clock cycles

3.1.2 The nonlinear sub networks

The nonlinear sub-circuits are simulated by using specialized processors specific for each device type. These processors are designed as concurrent periodic EFSMs ("Extended Finite State Machine") and coded in synthesizable VHDL. They are stored in a parameter driven library which currently includes models for diodes, thyristors, MOSFETs, many different type of voltage and current sources (step, ramp, sinus and cosines, etc.). In general, these models contain a list of parameters to allow them to be configured to the specific device behaviour, a data path responsible for the mathematical calculations, predictors to extrapolate sampled values at the inputs, digital filters to output instantaneous values as well as averaged values, etc. An EFSM controls how these specialized processors move from the current state to the next one. Each transition rule is defined as $S_{cur} \xrightarrow{a[guard]} S_{next}$ where *guard* is a transition condition such that if its value is true at state S_{cur} and event *a* is executed, then the EFSM moves to state S_{next}. As an example, figure 4 shows the simplified representation of a thyristor model [8] [9].

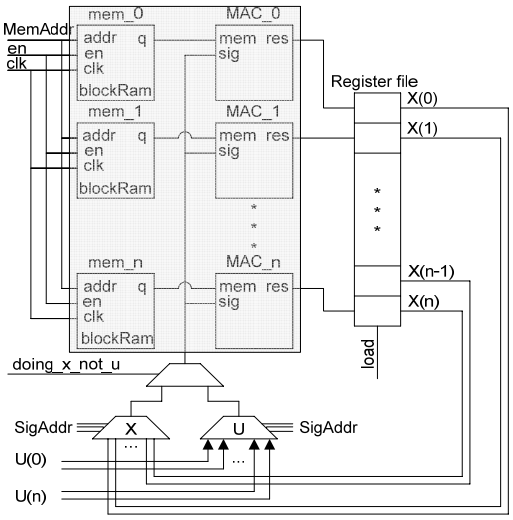

Figure 3. State Space Solver Macro Architecture

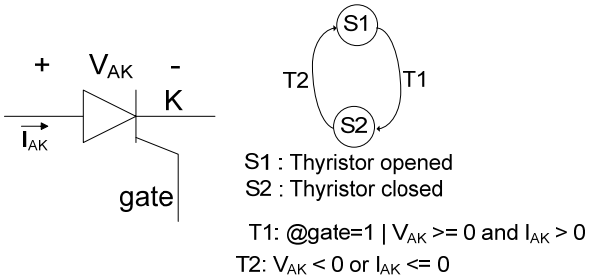

Figure 4. Simplified internal structure of a thyristor model

3.2 Decoupling Link Model

We should notice that one of the most important features of the proposed architecture is its ability to simulate each sub network using a different timestep. As an example, in a power network including power electronic devices the maximum natural frequency of the linear sub network is much smaller than the maximum natural frequency of the power electronics circuit. Decoupling circuits with very

different natural frequencies and simulating them independently allow each circuit continuous-time model to be discretized using the most appropriate time step which reduces numerical errors and most important optimizes the use of the available hardware. Otherwise, the whole network would have to be simulated with one unique timestep increasing the stiffness of the resulting discrete model as well as forcing the whole model to be evaluated in larger timesteps.

In general, the linear sub network and the nonlinear sub networks run at different timesteps and, therefore, the information processed by each part can not be immediately exchanged. Because the linear cub-circuits can be simulated in software and because FPGAs are very efficient in implementing DSP functions, we have chosen to let the nonlinear device models be responsible for doing an up sampling and down sampling of the inputs and outputs signals respectively in order to retime them to their appropriate clock domain. The linear part assumes the signals at their inputs and outputs are in the appropriated clock domain

3.2.1 Averaging Output Signals Link Model

The output of the diode model of Figure 2 is a current signal. Depending on the device type a voltage signal can also be used. The approach used to down sample the output signals consists in deriving the average current simulated at each timestep of the fast clock model during one timestep of the slow clock. Both the instantaneous and the averaged values are available at the output of the fast clock model. The approach assumes the slow clock period is an integer multiple n of the fast clock period. The average current between times Ta and Tb is the defined integral (5):

$$\overline{i(t)} = \frac{1}{T_b - T_a} \int_{Ta}^{Tb} i(t)dt \,, \tag{5}$$

The value of the output current is known at the end of each timestep of the fast clock. So we can write:

$$\overline{i(t)} = \frac{1}{T_b - T_a} \sum_{k=1}^{n} \int_{Ta+(k-1)}^{Ta+k} i(t)dt \,, \tag{6}$$

Assuming Δt is the fast clock time step we can use an integration rule to estimate the integral on the left hand side of (6). Two of the most popular integration rules are backward Euler and the trapezoidal. The expressions for the average current using these two integration rules are shown in (7) and (8) respectively. Because of the assumption the current can not abruptly change the two methods should produce the same result for all practical circuits. We have chosen to use backward Euler because it utilises a little less FPGA resources. If the smoothness of current signal assumption doesn't hold, a more accurate integration rule should be used or the ratio n should be kept large enough to reduce numerical errors.

$$\overline{i(t)} = \frac{1}{T_b - T_a} \sum_{k=1}^{n} i(T_{a+(k-1)}) \cdot \Delta t \,, \tag{7}$$

$$\overline{i(t)} = \frac{1}{T_b - T_a} \sum_{k=1}^{n} \frac{i(T_{a+(k-1)}) + i(T_{a+k})}{2} \cdot \Delta t \,, \tag{8}$$

3.2.2 Extrapolation of Input Sampled Nodes Link Model

The input signals of the fast clock models are sampled at the beginning of each slow clock timestep. The first possible approach to up sample the input signals could be hold its value constant during the whole slow clock timestep. Because the maximum bandwidth of the input signals can be comparable to the frequency of the slow clock this approach can incur in non negligible simulation error. A more accurate approach would be to use a predictor strategy [6]. A predictor extrapolates their value at the beginning of each fast clock timestep between two slow clock timesteps, based on the past history of the slow clock sampled values, and update them with the new sampled value at the beginning of the next slow clock timestep. Notice that in order to obtain reasonable results the ratio parameter n and the input signal bandwidth has to be appropriately chosen. We have constrained our analyses to lower order predictors for two reasons: higher order methods have smaller region of stability; lower order methods consumes lesser FPGA resources. We studied the extrapolation rules listed in table III and experimental results show the error of Adams-Bashforth-2 is half the error of Mid-Point or Euler so that we chose to implement the Adams-Bashforth-2 extrapolation rule.

TABLE III
EXTRAPOLATION RULES

Rule	Expression	Order
Euler	y(n+1) = y(n) + h . f(x(n), y(n))	1
Milne's method	k1 = 2 . f(x(n-2), y(n-2)) - f(x(n-1), y(n-1))	4
	K2 = 2 . f(x(n), y(n))	
	y(n+1) = y(n-3) + (4h/3) . (k1 + k2)	
Runge-Kutta	k1 = h . f(x(n), y(n))	2
	k2 = h . f((x(n)+h), (y(n) +k1))	
	y(n+1) = y(n) + 0.5 . (k1 + k2)	
Midpoint method	y(n+1) = y(n-1) + 2 . h . f(x(n), y(n))	2
Adams-Bashforth-2	k1 = 3 . f(x(n), y(n)) - f(x(n-1), y(n-1))	
	y(n+1) = y(n) + 0.5 . h . k1	2

3.3 The Macro Architecture Implemented in the FPGA

The resources available in modern FPGAs, listed in table II, make them a convenient platform to implement systems including a large number of parallel small specialized processors; distributed memory access algorithms; high speed control logic and finite state

machines as well as very high throughput DSP functions. Besides the FPGA switch matrix allow the exchange of a high amount of data between modules at very high clock frequency and with very small latency. These features solve the bottleneck issues present in the parallel processor based architectures. On the other hand, these features don't come without a price. There is a limited amount of resources inside the FPGA so they should only be used to perform tasks for which the classical architecture has limitations.

The proposed architecture shown in Figure 1 uses a data flow processing model where the sub-circuits are interconnected through their input and output ports so that the inputs of a sub-circuit can only change at the end of a time step. The outputs of the sub-circuits only depend on their inputs and their internal states. At the end of each time step the sub-circuit transfers its calculated voltages and currents to the next sub-circuit. There is a scheduler that synchronizes the function of all modules and is responsible for coordinating the following tasks: simulation initialization; generation of the multiple timesteps reference clocks; synchronization of the simulation and exchange data between linear and nonlinear. We should notice that the stability of the simulation depends a lot on the initialization strategy adopted. The scheduler coordinates the initialization following these steps:

a) The main voltage and current sources are initialized;
b) The linear sub network is initialized with the state space initial condition vector;
c) Afterwards, the slow clock linear circuits are initialized using the main sources and state space output values (ex.: digital filters, PI controllers, etc.). Though these circuits are linear they are implemented in the FPGA together with the nonlinear sub networks;
d) Finally, the fast clock nonlinear sub networks are initialized using the output values of the previous circuits.

All modules of the system, including the state space solver, follow the same standard interface specification shown in Figure 5 which is based on this processing model: sample input signals, process information, and hold computed data at outputs. The voltage, current and logical signals are part of the data path. The remaining signals are part of the control path and are intended to send information to or receive information from the scheduler. Based on the processing model described above, the scheduler shown in Figure 5 was defined. It uses the following control signals:

TS1 -> TSn – in the current version the system supports only two clock domains: the slow and the fast clock;
EN – for debug purpose, the scheduler can assert EN control to suspend the functioning of all modules. The normal functioning resumes when EN is negated;
STC_1 -> STC_n – the scheduler can selectively start each class of modules. The input signals are sampled when a STC for the module is detected;
REG_1 -> REG_n – at the end of each timestep this signals are asserted to transfer the newly computed result to the module output registers and hold them during the next timestep;
EOC_1 -> EOC_n – These signals tell the scheduler all modules completed their processing before the end of the timestep. Otherwise an error is generated meaning the real time constraint was violated.

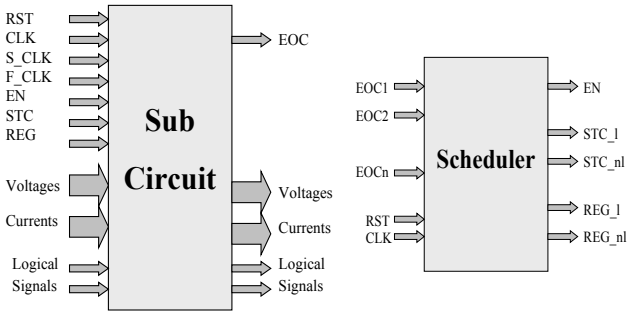

Figure 5. Sub-Circuit and Scheduler signal interface

4. Experimental Results

The test environment built to evaluate this approach consists of an AMD AthlonTM XP 2400+ microcomputer and a Digilent Inc. XUP Virtex II Pro Development FPGA Card with a 2VP30-7-FF896 Virtex II Pro FPGA. The proposed simulator results were obtained with Mentor Graphics ModelsimTM VHDL simulator and the Xilinx ISE™ Development Environment. The test circuit, shown in Figure 6, is a three-phase cicloconverter. It takes a three-phase 50Hz voltage source and converts it to a 60 Hz power bus. The first stage uses a diode rectifier to convert the 50Hz voltages in a DC voltage across the system capacitor. The second stage is a DC-AC PWM converter which takes the energy stored in the capacitor and converts it to 60Hz by using PWM modulation ("PulseWidth Modulation"). The gates of the MOSFET power switches are controlled by a sinusoidal PWM modulator which had its carrier frequency set to 1080 Hz at a modulation index of 0.8. The PWM was configured to generate a 60 Hz sinus at the output when the high frequency carrier is filtered out. The results for the SimPowerSystem simulation and the proposed simulator are shown in Figures 7 and 8 respectively. As we can see, the results show similar accuracy to those obtained with SymPowerSystems from Mathworks, which is a commercial power system simulator tool largely used by electrical utilities and research centers.

5. Conclusion

This paper presented a methodology and an architecture for real time power system simulation based on FPGA devices. The simulator can be entirely implemented in a FPGA device to simulate a small to medium size electrical network with very high natural frequencies and fast power electronic devices. It can also be integrated to

a parallel processor based simulator to simulate large and complex power systems. The main advantages of this methodology are: it can achieve higher performance than traditional RTSS architectures by relaxing the constraints imposed by the memory-processor-interface bottlenecks; it is based on a HW/SW co-design methodology that allows RTSS designers to explore new alternatives in the design space to optimize the tradeoffs between cost, accuracy and performance.

We must also notice there is an underway research on using the methodology proposed here to simulate physiological systems in real time. The basics of this work were published in [7] and more detailed results are expected to be published in the near future.

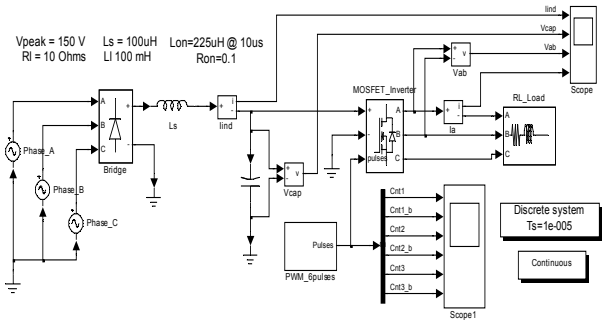

Figure 6. Schematic Diagram of the 50Hz–60Hz icloconverter

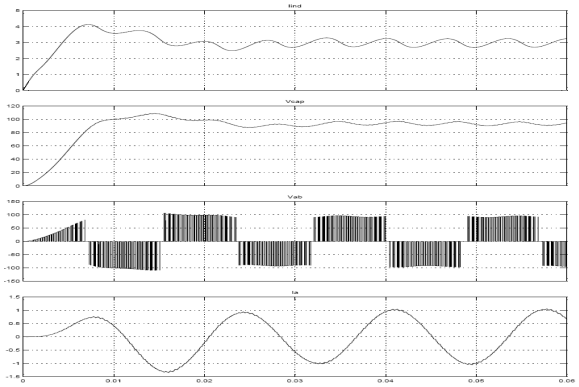

Figure 7. SimPowerSystems simulation results

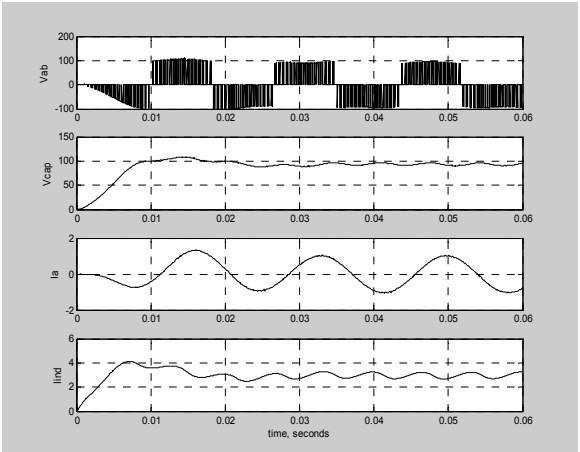

Figure 8. FPGA real time simulator simulation results

Acknowledgements

J. C. G. Pimentel thanks Xilinx Inc. for its support providing the FPGA development kit used for the simulation and experimental results, and Dr. Yosef Tirat-Gefen at Castel Research Inc., Dr. Guilherme DeSouza at Univ. of Missouri-Columbia and Dr. Antonio Mesquita at COPPE/UFRJ for valuable feedback during the preparation of this paper.

References

[1] C. Dufour and J. Belanger, “Real-time Simulation of a 48-Pulse GTO STATCOM Compensated Power System on a Dual-Xeon PC using RT-LAB,” IPST’2005, June 19-23, 2005, Montreal, CA.

[2] M. Matar, M. Rahman and A. Soliman, “FPGA-Based Real-Time Digital Simulation,” IPST’2005, June 19-23, 2005, Montreal, CA.

[3] C. Gombert, “Real Time Simulation of Power Electronics Devices Applied to Electrical Power Systems," INPG – Institute Nationale Polytechnique de Grenoble, Doctoral Dissertation, 2005 (in French).

[4] J.C.G. Pimentel, "Hardware Emulation for Real-Time Power System Simulation," IEEE ISIE’2006, Montreal, QC, CA, July 9-13, 2006, accepted for publication.

[5] T. Maguire and J. Giesbrecht, “Small Time-step (< 2μSec) VSC Model for the Real Time Digital Simulator,“ IPST’2005, June 19-23, 2005, Montreal, CA.

[6] L.O. Chua and P.Y. Lin, “Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques,” Prentice Hall, 1975.

[7] J.C.G. Pimentel and Y.G. Tirat-Gefen, "Real Time Simulation of Physiological Systems," IEEE EMBC’2006, Easton, PA, USA, April 1-2, 2006.

[8] K. D. Nguyen, Z. Sun, P. S. Thiagarajan and W. F. Wong, “Model-Driven SoC Design via Executable UML to SystemC,” IEEE RTSS’2005, Dec. 5-8, 2004, Lisbon, PT, pp. 459-468.

[9] T. Kitani, Y. Takamoto, K. Yasumoto, A. Nakata and T. Higashino, “A Flexible and High-Reliable HW/SW Co-Design Method for Real-Time Embedded Systems,” IEEE RTSS’2004, Dec. 5-8, 2004, Lisbon, PT, pp. 437-446.

[10] S. Shonberg, “Impact of PCI-Bus Load on Applications in a PC Architecture,” IEEE RTSS’2003, Dec. 3-5, 2003, Cancun, MX, pp. 430-439.

[11] A. Cohen, “A Performance Analysis of 4X InfiniBand Data Transfer Operations,” IPDPS, Apr. 22-26, Nice, FR.

[12] W. Feng, B. Balaji, C. Baron, L. D. K. PandaN. Bhuyan and , “Performance Characterization of a 10-Gigabit Ethernet TOE,” Proc. of the 13th Symposium on High Performance Interconnects, Aug. 17-19, 2005.

[13] Xilinx Inc.: www.xilinx.com

[14] Mathworks Inc.: www.mathworks.com