A Versatile Cluster-Based Real-Time Digital Simulator for Power Engineering Research

Lok-Fu Pak, *Student Member, IEEE*, M. Omar Faruque, *Student Member, IEEE*, Xin Nie, *Student Member, IEEE*, and Venkata Dinavahi, *Member, IEEE*

***Abstract*—This paper presents the development of a unique parallel and distributed real-time digital simulator for power engineering research at the University of Alberta. The simulator is built entirely from commodity-off-the-shelf (COTS) hardware and software components, making it very flexible and scalable. In parallel multi-tasking operations, a combination of real-time Linux operating system and an ultra-fast network forms the backbone for the internal communication between the 16 powerful processors of the Xeon-Cluster. Field programmable gate array (FPGA)-based multi-channel digital and analog I/O ports are interfaced to communicate with the external hardware components. The model development software utilized for power applications is based on the highly customizable industry-standard MATLAB/SIMULINK environment. A detailed case study of the real-time simulation of a three-level 12-pulse vector-controlled ac drive is presented to illustrate the precision capabilities of the simulator. Multiple integration algorithms and multirate computation have been applied for the simulation of the system, with slow (machine) and fast (converter) dynamic components. Real-time simulation of the entire system has been achieved with a maximum computation time of 5.35 μs on a step-size of 10 μs for the first time. Results obtained from the real-time simulation have been validated with an offline simulation using PSCAD/EMTDC.**

***Index Terms*—Field programmable gate arrays (FPGA), parallel processing, power engineering, real-time systems.**

I. Introduction

SIMULATION has always been a mainstay at all stages of the development and operation of power systems—planning, design, prototyping, and testing [1]. Prior to the advent of digital computers, analog simulators, based on scaled-down models, were relied upon to emulate and predict the behavior of actual power systems. Due to their cost, complexity, and inherent inaccuracies, such simulators are now being supplanted by digital simulators. Currently, several offline digital simulation software tools are available, with varying degrees of modeling and simulation capabilities, such as PSCAD/EMTDC, MATLAB/SIMULINK, EMTP-RV, PSS/E, SPICE, and many others. However, one of their main drawbacks is that they often lack the capability of interfacing with actual hardware, such as a digital controller or a protective relay. A real-time digital simulator with adequate computational bandwidth can overcome this obstacle. The main requirement of a real-time digital simulator is to ensure that the calculation for a time-step is completed within the chosen step-size. Systems with complex circuitry and fast operating switches, such as multi-level multi-pulse power electronic converters, present the greatest challenge. The key to achieving real-time simulation in such cases is a combination of fast computer hardware with efficient software for parallel computation. In addition, real-time simulation should facilitate human-machine interface and real hardware interaction.

Several real-time simulators have been reported in the literature, based on DSP [2]–[8], RISC [9]–[11], CISC, and VLSI technology [12]–[14]; however, the main advantages of using general purpose processors [15] over specialized ones lie in their lower cost and faster development cycle. Field programmable gate arrays (FPGAs) are yet another kind of digital processor that is gaining popularity in real-time simulators, although not as core computational engines but as high-speed peripheral I/O devices. With further development in intellectual property (IP) cores and software protocol, it will not be long before they will be used as the main computing elements. A growing trend in affordable high-performance computing is to use PC clusters. A PC cluster is a group of off-the-shelf personal computers connected using a fast communication network. Depending on the underlying operating system, the cluster can be made to execute tasks in offline or real-time.

The objective of this paper is to present the development of a cluster-based, parallel real-time digital simulator for power engineering research. The simulator (see Fig. 1) is an integral component of the **R**eal-**T**ime e**X**perimental **Lab**oratory (RTX-Lab) in the Power Engineering Group at the University of Alberta. The motivation behind this development was to create a real-time simulation platform that would be fully extensible and affordable; that can support commercially available hardware and software components; and that has sufficient computational capacity to satisfy the practical needs of a variety of research areas in power engineering. The main features of the simulator include the following:

- built entirely from commodity-off-the-shelf (COTS) components;
- fully flexible, scalable, and allows parallel-in-space as well as parallel-in-time processing on multiple CPUs. The former approach is based on partitioning the original system into smaller subsystems and distributing their computation among several processors, while the latter approach is based on simultaneous multiple time-step

Manuscript received July 12, 2005; revised August 23, 2005. This work was supported in part by the Canada Foundation for Innovation (CFI), in part by the Natural Science and Engineering Research Council of Canada (NSERC), and in part by the University of Alberta. Paper no. TPWRS-00414-2005.

The authors are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada (e-mail: lokpak@ece.ualberta.ca; faruque@ece.ualberta.ca; xnie@ece.ualberta.ca; dinavahi@ece.ualberta.ca).

Digital Object Identifier 10.1109/TPWRS.2006.873414

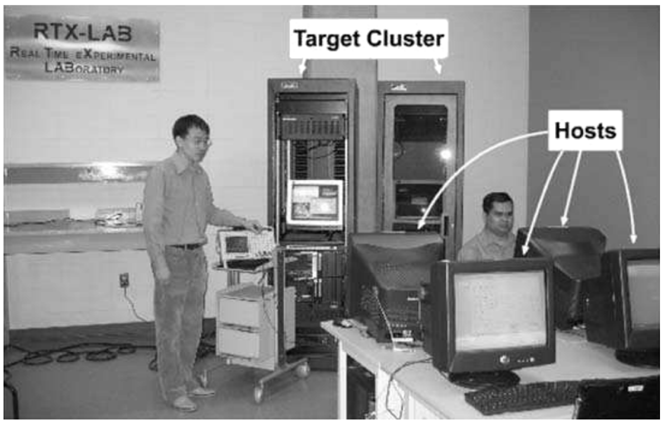


Fig. 1. RTX-Lab real-time simulator.

solution of differential algebraic equations, as in transient stability simulations [16].

- high communication speed with 10 Gigabit per second (Gbps) bandwidth between computational nodes;
- true multirate operation based on a reliable Linux real-time operating system (RTOS);
- precise accounting of switching events for power electronic applications;
- fast FPGA-based analog and digital I/Os;
- capability to create stand-alone embedded control systems;
- variety of synchronization options for various applications: software, hardware, and eXtreme High Performance (XHP) synchronization;
- use of industry standard mathematical modeling software MATLAB/SIMULINK;
- ability to integrate custom models and solvers written in S-function using high-level programming languages, such as C/C++ and FORTRAN;
- enables development of advanced user interfacing applications using commercial GUI software such as LABVIEW or PYTHON.

The power engineering applications of this simulator include, but are not limited to, the following areas:

- large-scale electro-magnetic transient simulation;
- network planning and feasibility studies;
- parallel load-flow and short-circuit studies;
- harmonics and power quality studies;
- FACTS and HVDC control studies;
- hardware-in-the-loop simulation and rapid controller prototyping;
- large-scale transient stability simulations;
- multirate digital simulation and control;
- coordination and testing of protective relays;
- design of variable frequency industrial drives and controllers;
- design and implementation of renewable energy sources such as wind and photo-voltaic systems.

This paper is organized as follows: Sections II and III provide details on the hardware and software architecture of the simulator, respectively. Synchronization issues are discussed in Section IV. A detailed case study of the real-time digital simulation

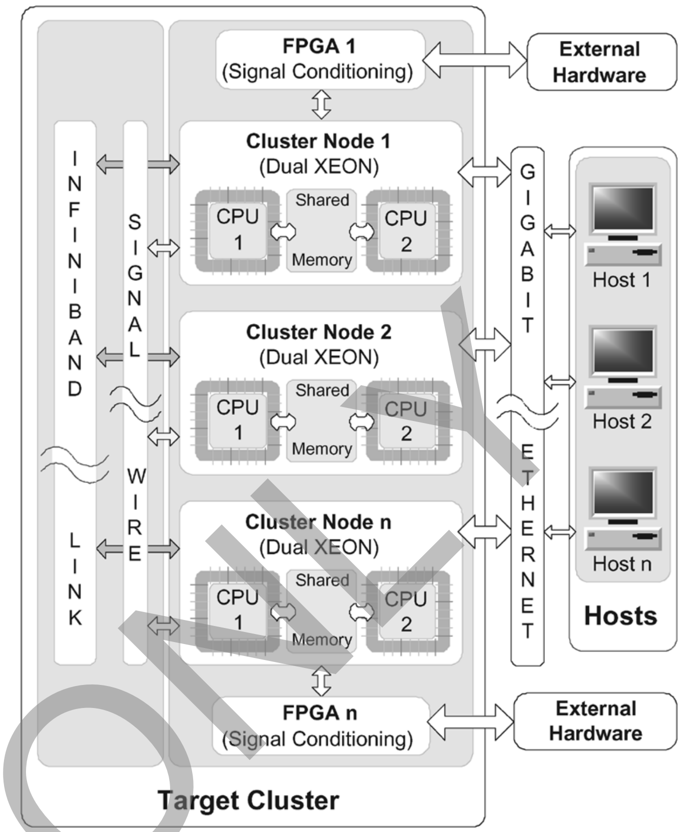


Fig. 2. Hardware architecture of the RTX-Lab simulator.

of a three-level 12-pulse vector-controlled ac drive is presented in Section V, followed by conclusions in Section VI.

II. Hardware Architecture

The hardware architecture of the real-time simulator is shown in Fig. 2. It mainly consists of two groups of computers known as *target cluster* or *targets* and *hosts* or *command stations*. In addition, there are FPGA-based digital and analogs I/Os and high-speed communication links between *targets*, *hosts*, and the external hardware.

A. Target Cluster Configuration

The computers inside the *target cluster* are high-speed, high-performance general purpose PCs that work as the computation engine for the real-time simulator. During the simulation, one of the *cluster nodes* works as the *master* and the others work as *slaves*. In addition to its computation task, the *master* manages the communications between the *hosts* and the *target cluster* and among all the other assigned *cluster nodes*. *Slaves* are used for computation purpose only. The user has complete control in distributing the *master* and *slaves* among the available *cluster nodes*.

In the RTX-Lab simulator, eight *cluster nodes* are connected in parallel through two different communication links. The *target cluster* is highly flexible, and the computation power can be scaled by adding more nodes. Each node has two 3.0-GHz Intel Xeon processors that can be run interactively or independently. The two processors communicate with each other through the shared memory. When the XHP mode is activated,

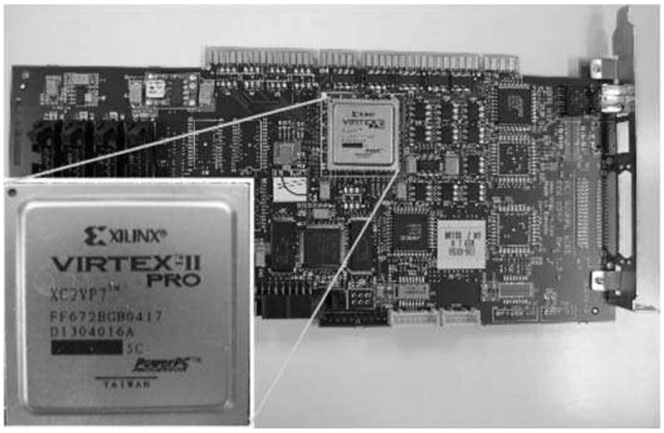


Fig. 3. FPGA-based signal conditioning module.

one CPU could be dedicated entirely to the simulation while the other CPU is running the RTOS.

B. Host Configuration

The group of computers used for the model development, compilation, and loading the compiled programs to the *cluster nodes* is known as *hosts* or *command stations*. Through the *hosts*, the user can obtain full access and utilization of the *target clusters*. When the FPGA-controlled hardware I/O is not activated, the *hosts* could replace the external hardware for obtaining simulation results and feeding control values to the *target clusters* in real-time. The number of *hosts* that can be connected to the *target clusters* is unlimited; however, the RTX-lab simulator currently uses six *hosts* networked with the *target clusters* through a Gigabit Ethernet LAN. One of the *hosts* is dedicated as the central licensing unit. Each *host* has a 3.0-GHz Intel Pentium IV CPU with hyper-threading (HT) to offer fast loading and compilation of the developed models.

C. FPGA-Based I/Os and Signal Conditioning

Real hardware I/O signals of various types (analog, digital, PWM, timers, encoders, etc.) can be interfaced with the *target clusters* through the FPGA-based multi-channel I/O modules. Each module transmits data back and forth through the 100-MHz Xilinx Virtex-II Pro FPGA (see Fig. 3). Similar to the *cluster nodes*, the I/O capacity could be scaled by simply adding more modules. The 100-MHz operation speed gives the I/O channels a resolution down to 10 ns. This means that multiple events could be captured or generated within a simulation time-step to incorporate very precise timing into the model, such as the PWM-controlled IGBT switching. An effective precision better than 1 μs can be obtained when combining the FPGA event detection with specialized real-time interpolation algorithms [17].

D. Communication

Several state-of-the-art computer networking technologies have been utilized to achieve the best communication throughput. The four types of communication involved in the RTX-Lab simulator are as follows: 1) the inter-CPU communication; 2) data transfer between the *cluster nodes*; 3) interfacing the external hardware to the simulator; and 4) executable loading and result acquisition between the *hosts* and the *cluster nodes*. All these communication needs have been addressed by one or several corresponding communication technologies as described below.

1) Shared Memory: On a single motherboard, the communication across the symmetric Xeon processors is achieved through the shared memory. With a bus-speed of 2.67 Gbps, this architecture allows the CPUs to take advantage of the raw internal data communication speed.

2) InfiniBand Link: Emerging as an industrial standard, the channel-based high-bandwidth InfiniBand link is engineered for real-time data transfer among the *cluster nodes*. The three building blocks for the InfiniBand link consist of the host channel adapters (HCA), the InfiniBand switch, and the optical fiber connecting them. Through the standard PCI-X bus, the HCA adapter provides each *cluster nodes* with the theoretical maximum throughput of 30 Gbps. The 480-Gbps InfiniBand Switch equipped with the RTX-Lab simulator has 24 10-Gbps dual-ports in a 1U chassis providing high performance and low latency communication.

3) SignalWire Link: The SignalWire link is another high-speed real-time networking interface that connects the *target* processors together to scale up the computation power. This real-time link exploits the benefits of the fast DMA burst transfer between the aforementioned FPGA modules. The 1.2-Gbps full-duplex transfer rate and the 200 ns latency of the SignalWire link ensures that the cycle time of a distributed simulation could be reduced below 10 μs without data overruns.

4) Gigabit Ethernet Link: The Gigabit Ethernet link has been built on top of the conventional Ethernet protocol, allowing data speed up to 1 Gbps. Similar to InfiniBand, this link has its corresponding network adapters, network switch, and cables (CAT5) connecting them. This link has been established mainly to facilitate the communication and data transfer between the *hosts* and the *targets*.

For any specific application, the user has full control in combining the above communication links to maximize the performance.

III. Software Architecture

The software architecture of the RTX-Lab simulator is depicted in Fig. 4. *Hosts* and *targets* can be running different operating systems (OS); however, their communication is established through different standardized protocols with the specific physical communication links described in Section II.

A. Target Operating System

The *target cluster* runs on RedHawk Linux, a state-of-the-art real-time version of the open-source Linux, modified to support the functionality and the performance required by complex real-time applications. The main features of the RTOS are listed below.

- It offers a single programming environment and direct control of all system operations by using a single kernel design.

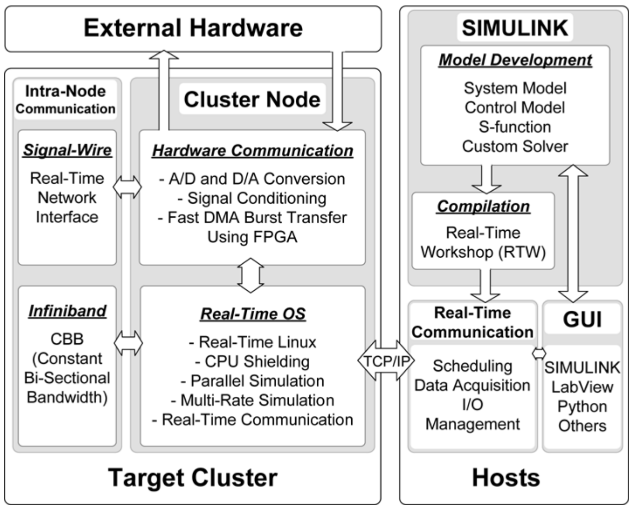


Fig. 4. Functional block diagram of software used in the RTX-Lab simulator.

- It allows deterministic program execution and response to interrupt as well as maintaining high throughput for other tasks such as writing to the file, networking, and graphics I/O operations.
- The highly optimized support for symmetrical processing features XHP mode—CPU shielding. The shielded CPU is dedicated to tasks that are deterministic in nature, which excludes interrupt processing, kernel daemons, and other RTOS tasks.

The above features of the RTOS make the simulator a powerful and flexible multi-CPU operator. When the XHP-mode is enabled, the shielded CPU is completely taken over to perform the deterministic computations, and the unshielded CPU is responsible for running the RTOS. The following functions are performed by the *target*:

- real-time execution of the model;
- real-time communication between *target* nodes and I/Os;
- data acquisition and system initialization;
- implementation of parameter changes while the simulation is running;
- recording data and sending them to the *host* or any external monitoring devices;
- enabling I/O connection and signal conditioning.

B. Host Operating System

The *hosts* could be running either Windows or Linux OS. Currently, the RTX-Lab *hosts* use Windows XP as the resident operating system. A real-time interfacing software RT-Lab provided by OPAL-RT Technologies Inc. [14] has been set up to coordinate all the hardware engaged for the simulation. The *hosts* could also run other applications or user interface programs developed for the simulation. In summary, a *host* is mainly used

- to prepare, edit, and customize the power system and control model;
- to manipulate model parameters and acquire simulation results;
- to perform the offline verification simulation of the model in the MATLAB/SIMULINK environment;
- to divide large and complex systems into subsystems;
- to compile the model and generate the executable code ready for real-time simulation;
- to control the simulator's task sequence.

C. Model Development Software

As shown in Fig. 4, MATLAB/SIMULINK environment is employed for the development of power and control system models. Most physical systems and their controllers can be modeled with the built-in MATLAB/SIMULINK toolboxes; however, user-defined models and solver algorithms written in MATLAB or other high-level languages, such as C/C++ or FORTRAN, can also be included through the MATLAB/SIMULINK S-function interface. Large and complex systems model can be divided into several subsystems and distributed over a number of parallel operating *cluster nodes*. In the RTX-Lab simulator, additional model enhancement tools, such as RT-EVENTS [14], are good examples of the MATLAB/SIMULINK environment expansion.

D. User Interface—Command and Control

The MATLAB/SIMULINK environment provides the user with interfacing tools to monitor and control the input and outputs of real-time simulation and control. In addition, LABView, PYTHON, as well as specialized application programming interfaces (APIs) can also be incorporated for developing customized interface.

IV. Synchronization

Synchronization of various processes is paramount in real-time simulation. Two methods of synchronization are available in the RTX-Lab simulator—software synchronization and hardware synchronization. The former mode is synchronized with an RTOS timer that has a limited resolution of 500 μs. However, such resolution is insufficient for the simulations involving power electronic devices. With the application of XHP mode, the simulation time-step is now synchronized with the physical CPU clock, whose resolution is directly related to the CPU clock speed. For a 3.0-GHz Intel Xeon processor, the resolution could be reduced to much smaller than 1 μs. When analog or digital I/O is involved, hardware synchronization is a must. In spite of the application of the XHP mode, the synchronization is always triggered by the hardware timer on an FPGA card with sub-μs resolution.

A. Time-Step Anatomy

For parallel-in-space execution, the complete system model needs to be broken into subsystems. The computational burden could then be distributed among the subsystems assigned across multiple CPUs to accomplish real-time simulation. Fig. 5(a) and (b) illustrate the synchronization for the single subsystem model and the model with two subsystems, respectively. For the model with one subsystem, four sequential stages are executed within each time-step.

1) At the beginning of the time-step, all analog and digital I/O subroutines that need to be synchronized are called to ensure all samples are spaced in time.

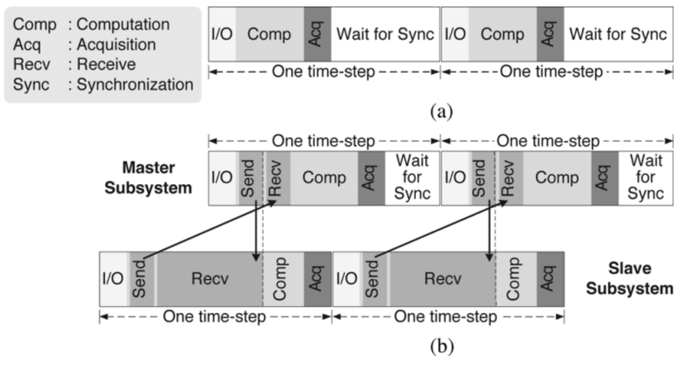


Fig. 5. Time-step anatomy. (a) Single-subsystem case. (b) Two-subsystem cases.

2) Computation (Comp) for the subsystem model is carried out in one of the *cluster nodes*.
3) Data acquisition (Acq) and model parameters/signals are updated to and from *host* computers.
4) The *cluster node* then waits for the synchronization signal while other background tasks such as writing data to storage disk are performed.

In the multiple subsystem case, Fig. 5(b), two extra sending and receiving stages are required to transfer data between the master and the slave subsystems. After the I/O stage, the master subsystem needs to

1) send the data that have been computed in the previous time-step to the slave subsystem;
2) receive the data that have been generated by the slave subsystem in the previous time-step.

Since only the master subsystem performs synchronization, the slave subsystem will be blocked until it receives release from the master. After the initial I/O stage, the sequence of execution for the slave subsystem would include

1) sending the data that have been computed in the previous time-step to the master subsystem;
2) blocking and waiting for the master subsystem to send the data that have been computed in the previous time-step;
3) performing computation for the model within the slave subsystem;
4) acquiring data and model parameter/signal updates from the *host* computers.

B. Latency Issues

Both software and hardware will contribute to the latencies in real-time simulation. Software latencies mainly result from OS interruptions such as context switches, cache invalidation, and nonpreemptive function calls. Such latencies create harmful jitters during computation, which may cause the actual time taken for computation to exceed the given simulation step-size. This is undesirable since in real-time simulation, the actual computation time must fall within the specified step-size at all times. This problem is resolved by enabling the XHP mode on the *targets*.

Hardware latencies are mostly created by the physical communication links. The communication threshold could be minimized by the application of high-speed and low-latency implementation architecture. The selection of InfiniBand, SignalWire, and Gigabit Ethernet provides different degrees of hardware latency compensation.

C. Multirate Systems

When a system contains models requiring different simulation resolutions, the multirate simulation approach could be applied. Since the slow transients in the system do not need to be computed at each simulation time-step, the computation for the fast and slow transients could be separated into different subsystems with different sampling rates; however, the various sampling rates should be integer multiples of the smallest rate for synchronization. This practice allows for the overall reduction in simulation time-step due to a reduction in computational stage. By distributing multi-threads on to multiple CPUs, the real-time simulator is capable of performing parallel execution for further reduction in simulation time-step. However, except for large system models, inter-node communication and hardware latencies would diminish the advantage of further system division for multirate simulation.

V. Case Study: Real-Time Simulation of a Three-Level 12-Pulse Vector Controlled AC Drive

A rigorous performance evaluation of the real-time simulator has been carried out using a three-level 12-pulse vector-controlled variable-speed ac drive system as a case study. The complete system consists of an squirrel cage induction motor (SCIM), the drive, and the digital controller. Decoupled vector-control technique [19] has been employed to realize the V/Hz motor speed manipulation.

A. System Model

As shown in Fig. 6, the electrical system consists of two three-phase diode rectifier bridges that are connected to the three-phase ac supply through a three-winding transformer. After LC filters, the dc sides of the rectifiers are connected to a three-level 12-pulse IGBT converter that produces the controlled voltages for the operation of the SCIM. The objective of choosing such a system is to incorporate a fair amount of complexity through the introduction of multiple power electronic devices undergoing high-frequency switching. All parts of the electrical system have first been modeled using built-in models in the SimPowerSystems blockset from MATLAB/SIMULINK. When the controller design and system configuration were verified, the inverter bridge was then replaced using a time-stamped RT-EVENTS block optimized for real-time simulation; a time-stamp basically involves capturing and labeling [7] the incoming gating signals, from the digital controller, with respect to the simulation time grid. The SCIM model is the fifth-order Park's machine model with rotor reference frame.

B. Simulation Methodology

In the SimPowerSystems, a power system is generally modeled in two parts: a state–space (SS) model for the linear circuit and a current injection feedback model for the nonlinear

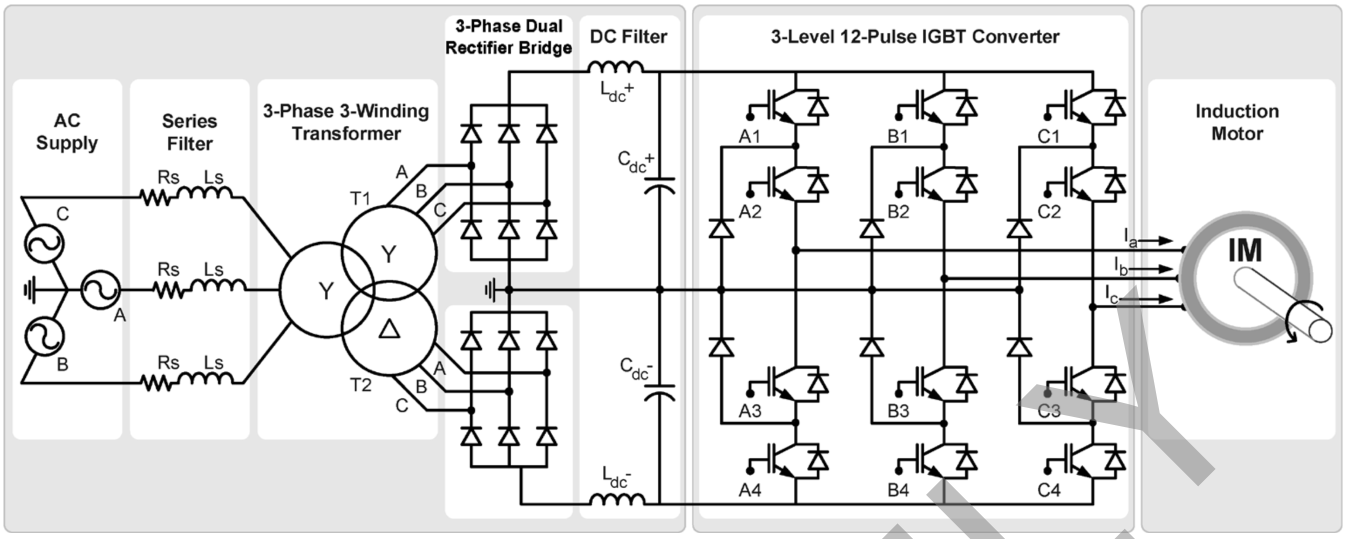


Fig. 6. Electrical system of vector-controlled variable-speed ac drive.

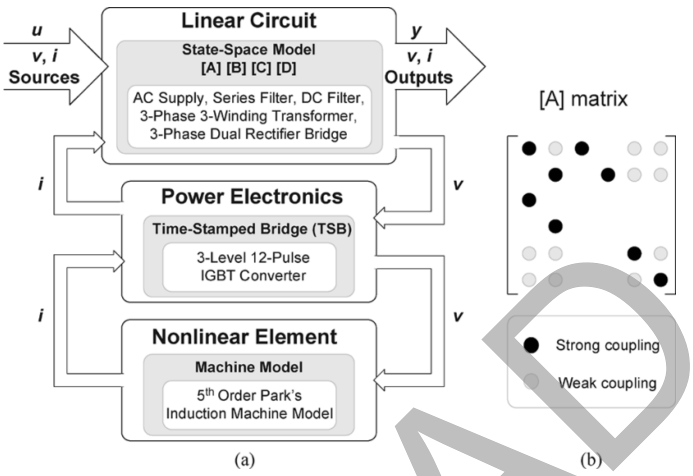

Linear Circuit
State-Space Model
[A] [B] [C] [D]
AC Supply, Series Filter, DC Filter, 3-Phase 3-Winding Transformer, 3-Phase Dual Rectifier Bridge
u
v, i
Sources
y
v, i
Outputs
Power Electronics
Time-Stamped Bridge (TSB)
3-Level 12-Pulse IGBT Converter
Nonlinear Element
Machine Model
5th Order Park's Induction Machine Model
i
v
[A] matrix
Strong coupling
Weak coupling
(a)
(b)

Fig. 7. (a) Electrical system model. (b) Sparsity of $[\mathbf{A}]$ matrix.

elements [18]. In this case, as shown in Fig. 7(a), the electrical model was divided into three parts. All the linear circuit elements, such as the ac supply, the series filter, the three-winding transformer, the three-phase dual rectifier bridge, and the dc filter, were modeled by SS equations. The IGBT converter was modeled using a first-level feedback interfacing with the main SS model through the input voltages and output currents. Taking the output voltages of the converter as the input stator voltages, the custom SCIM model would then generate the stator line currents and feed them back to the converter model.

The SS equation for the linear part of the system can be expressed as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{1}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^6$ is the network state vector, $\mathbf{u} \in \mathbb{R}^{17}$ is the input vector, and $\mathbf{y} \in \mathbb{R}^{14}$ is the output vector. The six elements of the state vector are i_{Ldc+}, i_{Ldc-}, v_{Cdc+}, v_{Cdc-}, i_{LT1}, and i_{LT2}. Fig. 7(b) illustrates the sparsity of the $\mathbf{A}$ matrix where strongly coupled elements are differentiated from the weakly coupled elements.

C. Model Optimization for Real-Time Simulation

The objective of this optimization is to be able to run the model in real-time with a minimum time-step. Simulation with a small time-step not only provides maximum resolution but also compensates for any inherited weakness in the model or in the numerical solver. The following tasks were undertaken to optimize the system model.

1) For the linear circuit model, a reduction in the number of states reduced the size of $[\mathbf{A}]$. Because the capacitor voltages and inductor currents were translated into state variables in MATLAB/SIMULINK, the logical first step was to combine the inductances of the series filter with the primary winding inductance of the three-winding transformer. After transferring the transformer's second and third winding inductances to the primary, a total of four states was reduced from the original SS model. To limit the states associated with the capacitor, the snubber circuits were eliminated from the dual rectifier bridges without significant degradation in simulation results.
2) Since the nonlinear elements were grouped into two feedback paths with only the converter model interfacing with the main SS model, the reduction of the input and the output vector minimized the $[\mathbf{B}]$, $[\mathbf{C}]$, and $[\mathbf{D}]$ of the main SS model. To increase the speed of simulation, the time-stamped converter bridge utilized a switching function model of the converter. A limitation of this model, however, is that it does not represent rectifier action of the three-level converter, should the electrical system contain a synchronous machine and should the gating signals for the converter be interrupted.
3) SS solution enabled the use of multiple integration techniques for the solution of different parts of the system with different dynamics. The induction machine being a slow dynamic component was solved using the forward Euler method, while the rest of the system was solved using the fourth-order Runge–Kutta method. The custom machine model is highly efficient without the need for calling and calculating initial values from an external

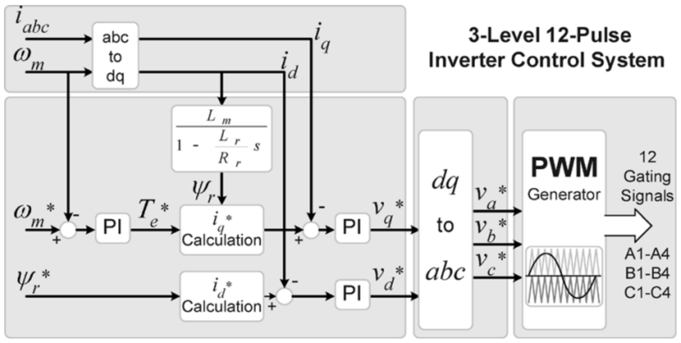


Fig. 8. Control scheme for the variable-speed ac drive.

MATLAB file. Furthermore, the use of the forward Euler method resolved the risk of creating algebraic loops for the model in the MATLAB/SIMULINK environment [18]. As the model was simulated with a small time-step, the inherent inaccuracy of the forward Euler method solution was compensated.

D. Control Algorithm

The block diagram of digital controller is shown in Fig. 8. The objective of the controller is to affix the SCIM speed (ω_m) to the given reference regardless of the disturbances from the changing mechanical torque (T_m^*). The independent control of (ω_m) is achieved in the synchronously rotating dq frame with the d-axis oriented along the rotating magnetic-flux vector. For the speed control, the measurements of ω_m and stator currents $(i_{s_{abc}})$ are required. In order to obtain the orthogonal current values in the desired dq frame, sequential manipulations associated with the Park's and Clarke's transformation are applied to $i_{s_{abc}}$. The electrical torque and rotor flux reference signals are obtained using the following equations:

$$T_e^* = i_q^* \psi_r \frac{3}{2}\frac{P}{2}\frac{L_m}{L_r} \tag{3}$$

$$\psi_r^* = L_m i_d^* \tag{4}$$

where L_m, L_r, and P are the mutual inductance, the rotor inductance, and the number of pole pairs, respectively.

A proportional-integral (PI) controller was implemented for the speed control loop to produce the reference torque (T_e^*). Equation (3) shows the realization of T_e^* with the quadrature-axis current (i_q^*). For the V/Hz speed control, the motor flux is held constant by the direct-axis current reference (i_d^*), given by (4). After the second set of PI controllers, the reference voltages v_d^* and v_q^* are transformed from the dq frame back to the stator-side abc frame. Finally, the comparison of the three-phase reference signals, V_a^*, V_b^*, and V_c^*, with the positive and negative carrier sequence produced the 12-PWM gating signals for the three-level IGBT converter.

E. Subsystem Configuration for Real-Time Simulation

In the parallel-in-space simulation approach, a large system is split into subsystems so that tasks can be divided among the *cluster nodes*. To maximize parallel computation, output data between subsystems must be exchanged in such a way that nodes can run their main calculation without having to wait for outputs from other nodes at any time-step. As there is a communication latency between the targets, it is not always possible that splitting a model into several subsystems will ensure a lower time-step. The important point in using many subsystems is the confirmation of parallelism in computation. If parallelism is ensured, the computation time will be reduced; however, due to communication overhead, the total time-step may not be greatly reduced for small systems. For large and complex systems, the benefit is obvious, and splitting a model into subsystems is a must to achieve real-time simulation at a lower time-step.

The three-level 12-pulse ac drive and its control system has also been simulated using multiple subsystems; a *master* subsystem for the linear part of the electrical system, a *slave* subsystem for the machine, 12-pulse converter, and the controller and a *console* subsystem for command and monitoring of the various inputs and outputs. Out of these three subsystems, only the *master* and the *slave* are allowed to run in real-time using two *target* nodes. The *console*, whose purpose is to communicate with the targets for inputs and outputs, runs on any of the *hosts*. The electrical outputs can be monitored either through the console or through external monitoring devices such as an oscilloscope connected to the FPGA-based I/Os.

F. Results and Discussion

Results obtained from the real-time simulation have been validated against an offline simulation using PSCAD/EMTDC. Attention has been given to maintain exactness in choosing the parameters and their values in both simulation environments (real-time and offline). Any differences in the results would thus be solely caused due to the inherent modeling and solution techniques involved. Reference parameters were changed dynamically to observe the transient response of the system while the real-time simulation was running. Other performance measures such as execution time, minimum time-step achieved, and simulation using parallel subsystems have also been observed. A time-step of 10 μs was assigned for the simulation of three-level 12-pulse vector-controlled ac drive system and its controller using the real-time simulator. A carrier frequency of 2.5 kHz was used for the PWM. The simulation results have been recorded for the two operating conditions: steady-state and transient situations.

1) Steady-State: Prior to recording the steady-state results, the simulation was allowed to run for 10 s so that all transients subsided. The initial settings for reference torque (T_m^*) and mechanical speed (ω_m^*) were 140 $N \cdot m$ and 160 rad/s, respectively. In the real-time simulator, line-to-line voltages and line currents at the output of the inverter were recorded through an oscilloscope connected to the I/O ports of the simulator. Fig. 9 shows 40.0 ms (starting from 10 s) of the waveforms V_{ab} and I_a produced by the real-time simulator. The rms and peak-to-peak voltage of V_{ab} have been found to be 436 V and 1.61 kV, respectively. Phase-a current (I_a) shows an rms of 42.30 A and a peak-to-peak value of 132 A.

On the other hand, results obtained from PSCAD/EMTDC have been shown in Fig. 10. A time-step of 10 μs and the same carrier signal as of real-time simulator (2.5 kHz) are used to simulate the system in PSCAD/EMTDC. Results obtained through the real-time simulator even with a larger time-step

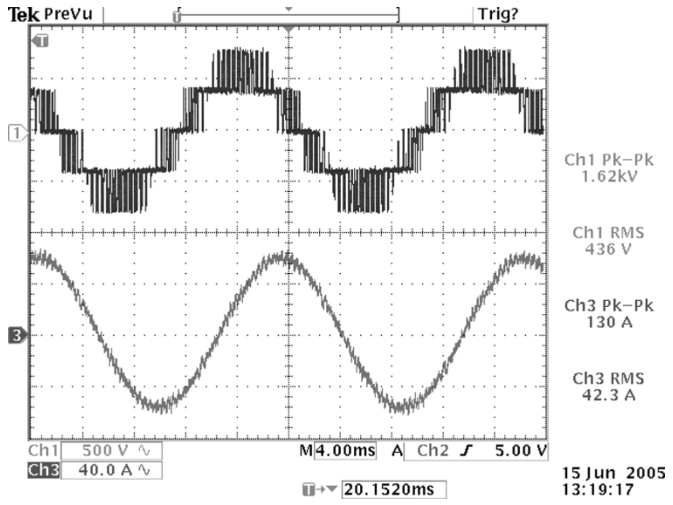


Fig. 9. Oscilloscope trace of voltage V_{ab} and I_a (real-time simulation).

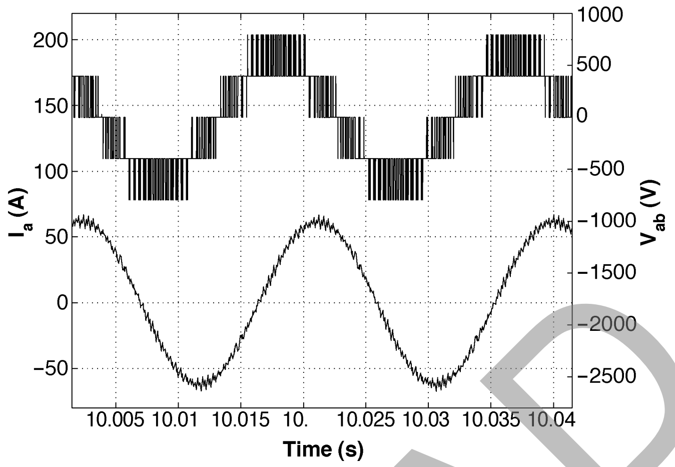


Fig. 10. Voltage V_{ab} and current I_a (offline simulation).

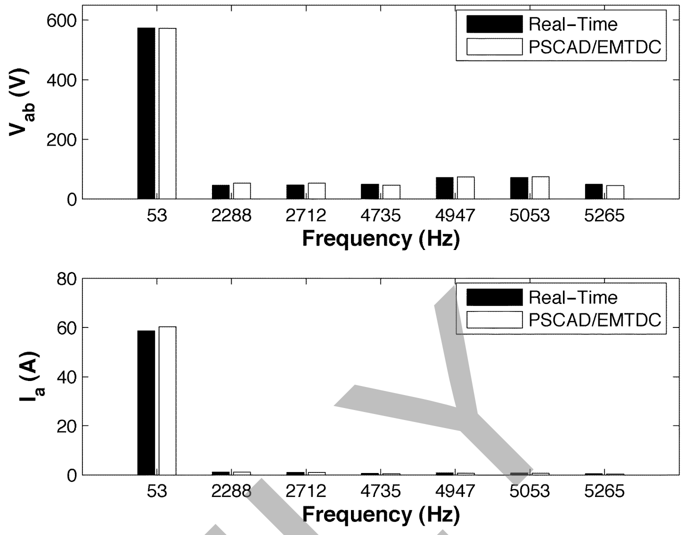


Fig. 11. Harmonic spectrum of voltage V_{ab} and current I_a.

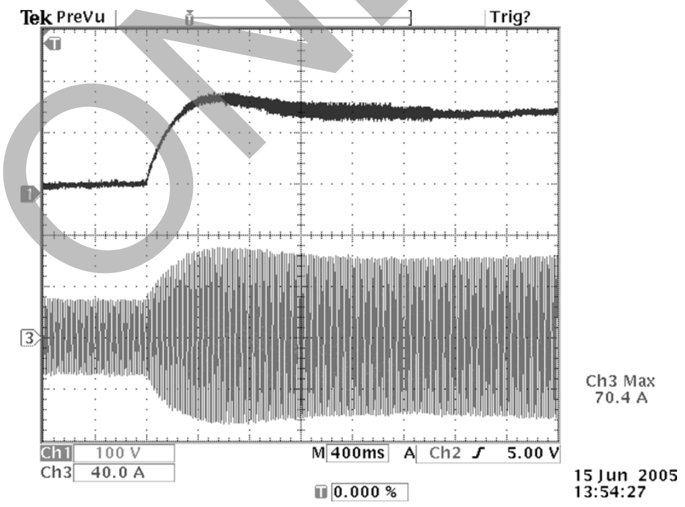


Fig. 12. Oscilloscope trace of transient response of T_e and I_a (real-time simulation).

were found to be in good agreement with the results obtained through PSCAD/EMTDC with smaller time-step. The rms and peak-to-peak voltage of V_{ab} have been found to be 439 V and 1.59 kV respectively. Similarly, phase-a current of 43.95 A (rms) with a peak-peak value of 136 A has been found in PSCAD/EMTDC simulation.

The harmonic spectrum of V_{ab} and I_a obtained through real-time simulation and PSCAD/EMTDC simulation are compared in Fig. 11, where major harmonics are shown. The spectrum indicates that the V/Hz control system produced a three-phase ac voltage at the output of the converter whose fundamental frequency is 53 Hz at a particular operating condition. PSCAD/EMTDC simulation with a 10 μs time-step corroborated this result. Similarly, the fundamental frequency of the current has been found to be the same as that of voltage as expected.

The harmonic spectrum of V_{ab} for both cases indicates that a few harmonics of very high order are present in the waveforms. Of them, the significant harmonics, plotted in Fig. 11, are at the following frequencies (Hz): 2288, 2712, 4736, 4947, 5053, and 5265. Considering m_f as the ratio of carrier frequency (2.5 kHz) to the fundamental frequency (53 Hz), the above harmonics are present at $m_f \pm 4$, $2m_f \pm 1$, and $2m_f \pm 5$. These results are in agreement with the principle of PWM. Differences in magnitude between the harmonics generated by real-time and PSCAD/EMTDC simulation are negligible.

2) Transients: The transient response of the system has been simulated through dynamic changes of the following two parameters: the reference torque (T_m^*) and the reference speed (ω_m^*).

Case 1) At $t = 10$ s, T_m^* has been changed from 0 to 140 $N \cdot m$, while keeping the speed unchanged at $\omega_m^* = 160$ rad/s. Due to the increase of the applied torque, a change in current has been observed. Fig. 12 shows the response of the torque and current obtained from the real-time simulator. The transition of torque to its reference value has been found to be quite smooth. While the torque settles to its new value (140 $N \cdot m$), the amplitude of the current increases from 33.5 to 70.4 A. Fig. 13 shows the similar response obtained from PSCAD/EMTDC simulation. For the same transient situation, the current amplitude jumps from 33.8 to 69.5 A.

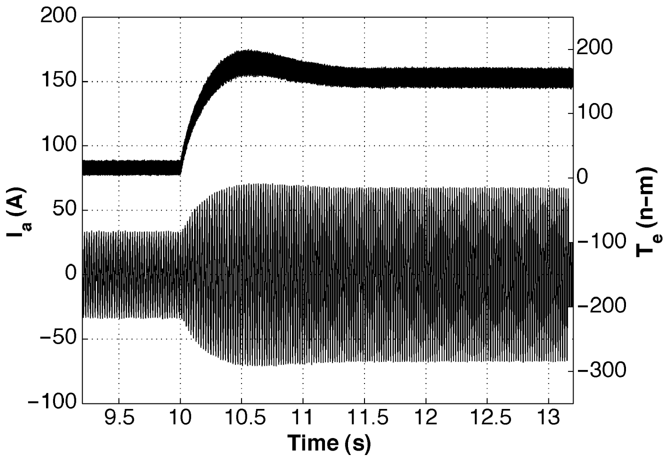


Fig. 13. Transient response of T_e and I_a (offline simulation).

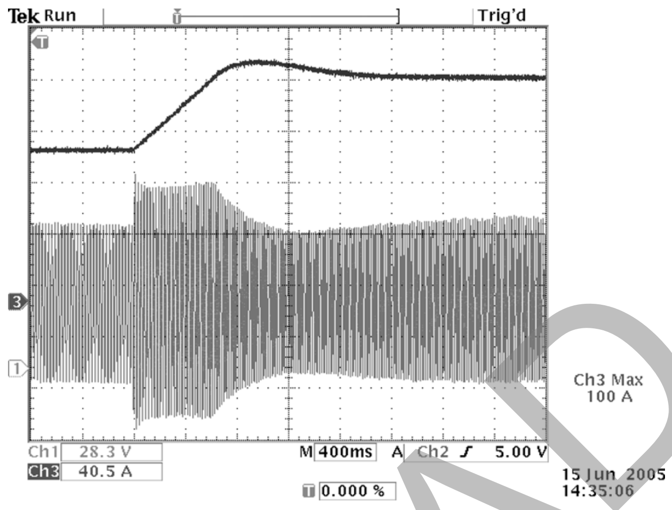


Fig. 14. Oscilloscope trace of transient response of ω_m and I_a (real-time simulation).

Case 2) In this case, the reference torque has been maintained at $T_m^* = 140\ N \cdot m$, while the reference speed ω_m^* has been changed from 120 to 160 rad/s at $t = 10$ s. Fig. 14 shows the transient response obtained through real-time simulation. A smooth transition of speed to the desired level associated with a slight disturbance in the current has been observed. An almost identical result has been experienced in the simulation using PSCAD/EMTDC as shown in Fig. 15. Table I shows the comparison between the results obtained from the real-time simulator and PSCAD/EMTDC.

In all of the cases, PSCAD/EMTDC simulation results are in agreement with the results of real-time simulation validating the accuracy of the real-time simulator. Negligible discrepancies were observed, which can be attributed to differences in modeling and the numerical solvers. Nevertheless, the results obtained through real-time simulator are very close to that of PSCAD/EMTDC.

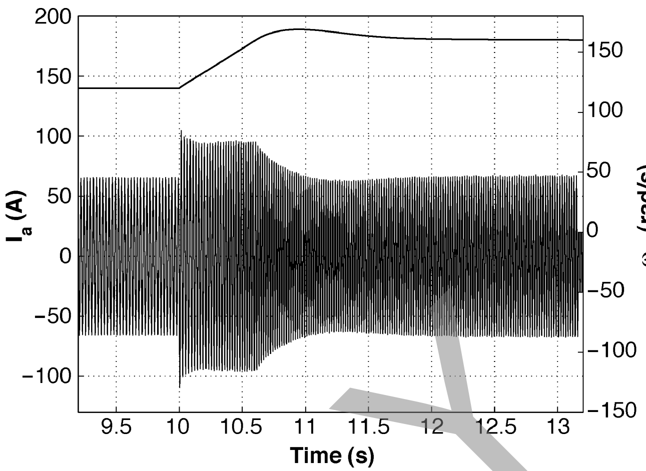


Fig. 15. Transient response of ω_m and I_a (offline simulation).

TABLE I
COMPARISON OF REAL-TIME SIMULATION AND OFFLINE (PSCAD/EMTDC) SIMULATION RESULTS WITH A 10 μs TIME-STEP

Steady-State Results		
Parameters	**Real-time**	**Off-line**
RMS voltage (V_{ab})	$436V$	$439V$
Pk-Pk voltage	$1.61kV$	$1.59kV$
RMS Current (I_a)	$42.30A$	$43.95A$
Pk-Pk Current	$132A$	$136A$
V_{ab} fundamental	$573.80V$	$573.24V$
I_a fundamental	$58.71A$	$60.30A$
%THD (V_{ab})	43.58	44.58
%THD (I_a)	21.09	22.89
Fund. Frequency (Hz)	53	53
V/Hz Ratio	8.226	8.283
Transient Results		
$\Delta\omega_m$	$40rad/s$	$40rad/s$
Max. I_a (due to $\Delta\omega_m$)	$100.0A$	$101.3A$
ΔT_m	$140N.m.$	$140N.m.$
ΔI_a (due to ΔT_e)	$35.4A$	$36.6A$

G. Real-Time Execution

Continuous effort has been delivered to optimize the model so that step-size of real-time simulation can be brought down as small as possible for the simulation of the entire system. The number of inputs and outputs has been maintained as minimal as possible so that CPU interruption is minimal. A time-step dissection revealed that using a single target node, real-time simulation has been achieved with a step-size of 10 μs, without causing any overruns, of which the maximum computation time is only 5.35 μs. A detailed breakdown of the execution time requirement for a complete time-step has been shown in Table II. Apparently, an idle time of 2.49 μs indicates the possibility of further reduction in the step-size. However, it was not feasible as during dynamic changes of input/output parameters, at some time-steps, the computation time exceeded the assigned time-step and caused overruns. Moreover, increase in the number of hardware I/Os would require higher execution time.

Further performance optimization has been attempted through parallel execution of the model running two subsystems in two different target nodes. This reduced the maximum computation time further to 2.46 μs for the *master* and 3.51 μs for the *slave*. As they were running in parallel, the net execution time was smaller than the single subsystem case; however, the required communication time between the two subsystems

TABLE II
EXECUTION TIMES FOR INDIVIDUAL TASKS WITHIN ONE TIME-STEP

Task	**Execution Time (μs)**
Computation Time	5.347666667
Idle Time	2.492333333
Data Acquisition	1.000000000
Status Update	0.232666667
Target Request Handling	0.067666667
Host Request Handling	0.040000000
Synchronization Handling	0.032333333
Others	0.787333333
Total Step-Size	10.00000000

TABLE III
SYSTEM PARAMETERS USED FOR CASE STUDY

Parameters	**Values**
AC voltage Source	$780V_{ll}$ rms @ $60Hz$
Source impedances	R_s=0.0312Ω, L_s=5.635mH
Transformer winding's voltages	$1000V$, $390V$, $390V$
Transformer kVA rating	100
Transformer leakage reactance	$0.005pu$
DC link voltage	± $390V$
DC link capacitance	$30000\mu F$
Induction motor rating	4-pole, $50HP$, $460V_{ll}$

overwrote that benefit and the assigned step-size became 12 μs. The communication overhead was approximately 4 μs. Further improvement of the performance was observed when hyper-threading of the CPUs was activated in both targets. The maximum computation times for the *master* and *slave* were found as 1.91 and 3.47 μs, respectively.

VI. CONCLUSION

Real-time digital simulation and control is rapidly assuming a mainstream position in power engineering by presenting a new and exciting research opportunity with a wide range of applications that were previously unknown or thought impracticable. The main research goal and continuing quest is to design new architectures and algorithms for simulating complex and realistic power systems for various types of studies, using state-of-the-art hardware and software components. This paper represents an important milestone in that quest by presenting the ongoing research efforts in real-time simulation and control at the University of Alberta. The contributions of the paper can be summarized as follows.

- A detailed view of the hardware and software architecture of a cluster-based real-time simulator built entirely using COTS components, and its communication strategy among multiple CPUs has been presented. Synchronization issues for different modes of simulation have also been elaborated. The fully flexible and scalable simulator is capable of simulating a large and complex models through parallel-in-space and parallel-in-time approaches and provides realistic interaction with external hardware.
- A detailed case study of real-time simulation of a three-level 12-pulse vector-controlled ac drive has been presented to demonstrate the performance of the simulator.
- Various techniques to optimize the model for running in real-time at a very small time-step has been described. The outcome is that for the first time, the entire system was simulated in real-time with a time-step of 10 μs and a maximum computation time of 5.35 μs.
- Results from the real-time simulator have been validated using PSCAD/EMTDC simulation results and they were found to be in excellent agreement.

APPENDIX

The parameters of the three-level 12-pulse ac drive used for the case study are given in Table III.

REFERENCES

[1] D. Jakominich, R. Krebs, D. Retzmann, and A. Kumar, "Real time digital power system simulator design considerations and relay performance evaluation," *IEEE Trans. Power Del.*, vol. 14, no. 3, pp. 773–781, Jul. 1999.
[2] X. Wang and R. M. Mathur, "Real-time digital simulator of the electromagnetic transients of transmission lines with frequency dependence," *IEEE Trans. Power Del.*, vol. 4, no. 4, pp. 2249–2255, Oct. 1989.
[3] P. G. McLaren, R. Kuffel, R. Wierckx, J. Giesbrecht, and L. Arendt, "A real time digital simulator for testing relays," *IEEE Trans. Power Del.*, vol. 7, no. 1, pp. 207–213, Jan. 1992.
[4] P. J. Throckmorton and L. Wozniak, "A generic DSP-based real-time simulator with application to hydrogenerator speed controller development," *IEEE Trans. Energy Convers.*, vol. 9, no. 2, pp. 238–242, Jun. 1994.
[5] X. Wang, D. A. Woodford, R. Kuffel, and R. Wierckx, "A real-time transmission line model for a digital TNA," *IEEE Trans. Power Del.*, vol. 11, no. 2, pp. 1092–1097, Apr. 1996.
[6] C. Dufour, L.-H. Hoang, J. C. Soumagne, and A. El Hakimi, "Real-time simulation of power transmission lines using Marti model with optimal fitting on dual-DSP card," *IEEE Trans. Power Del.*, vol. 11, no. 1, pp. 412–419, Jan. 1996.
[7] V. Dinavahi, M. R. Iravani, and R. Bonert, "Design of a real-time digital simulator for a D-STATCOM system," *IEEE Trans. Ind. Electron.*, vol. 51, no. 5, pp. 1001–1008, Oct. 2004.
[8] Y. Li, D. M. Vilathgamuwa, and P. C. Loh, "Design, analysis, and real-time testing of a controller for multibus microgrid system," *IEEE Trans. Power Electron.*, vol. 19, no. 5, pp. 1195–1204, Sep. 2004.
[9] M. Foley, Y. Chen, and A. Bose, "A real time power system simulation laboratory environment," *IEEE Trans. Power Syst.*, vol. 5, no. 4, pp. 1400–1406, Nov. 1990.
[10] J. R. Marti and L. R. Linares, "Real-time EMTP-based transients simulation," *IEEE Trans. Power Syst.*, vol. 9, no. 3, pp. 1309–1317, Aug. 1994.
[11] O. Devaux, L. Lavacher, and O. Huet, "An advanced and powerful real-time digital transient network analyzer," *IEEE Trans. Power Del.*, vol. 13, no. 2, pp. 421–426, Apr. 1998.
[12] H. Taoka, I. Iyoda, H. Noguchi, N. Sato, and T. Nakazawa, "Real-time digital simulator for power system analysis on a hypercube computer," *IEEE Trans. Power Syst.*, vol. 7, no. 1, pp. 1–10, Feb. 1992.
[13] J. A. Hollman and J. R. Marti, "Real time network simulation with PC-cluster," *IEEE Trans. Power Syst.*, vol. 18, no. 2, pp. 563–569, May 2003.
[14] C. Dufour and J. Belanger, "Real-time PC-based simulator of electric systems and drives," in *Proc. Int. Conf. Parallel Computing Electrical Engineering*, vol. 1, Sep. 7–10, 2004, pp. 105–113.
[15] M. Mir and M. A. Al-Saleh, "A digital simulator for determining the performance limits of computer relays," *IEEE Trans. Power Del.*, vol. 17, no. 1, pp. 60–67, Jan. 2002.
[16] M. La Scala, G. Sblendorio, and R. Sbrizzai, "Parallel-in-time implementation of transient stability simulations on a transputer network," *IEEE Trans. Power Syst.*, vol. 9, no. 2, pp. 1117–1125, May 1994.
[17] M. O. Faruque, V. Dinavahi, and W. Xu, "Algorithms for the accounting of multiple switching events in digital simulation of power electronic systems," *IEEE Trans. Power Del.*, vol. 20, no. 2, pp. 1157–1167, Apr. 2005.
[18] G. Sybille, P. Brunelle, H. Le-Huy, L. A. Dessaint, and K. Al-Haddad, "Theory and applications of power system blockset, a MATLAB/Simulink-based simulation tool for power systems," in *Proc. IEEE Power Eng. Soc. Winter Meeting*, vol. 1, Jan. 23–27, 2000, pp. 771–779.

[19] B. K. Bose, *Modern Power Electronics and AC Drives*. Upper Saddle River, NJ: Prentice-Hall, 2000.

Lok-Fu Pak (S'03) received the B.Sc. degree in electrical engineering from the University of Alberta, Edmonton, AB, Canada, in 2003. Currently, he is working toward the M.Sc. degree at the University of Alberta.

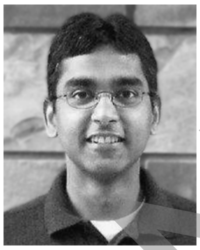
His research interests include renewable energy, wind turbine generation, electromagnetic transient simulation of FACTS, power electronics, and real-time simulation of power systems.

M. Omar Faruque (S'03) received the B.Sc.Engg degree in 1992 from Chittagong University of Engineering and Technology (CUET), Chittagong, Bangladesh, and the M.Eng.Sc degree in 1999 from the University of Malaya, Kuala Lumpur, Malaysia. He is working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada.

He worked in both industry and academia prior to starting his Ph.D. program, and his research interests include FACTS, HVDC and real-time digital simulation, and control of power electronics and power systems.

Xin Nie (S'03) received the B.Eng. degree in electrical engineering from Northeastern University, Shenyang, China, in 1997. Currently, he is working toward the M.Sc. degree at the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada.

Then, he worked as a Transmission Line Design Engineer in China. His research interests include electromagnetic transient simulation and real-time simulation of power systems.



Venkata Dinavahi (S'94–M'00) received the B.Eng. degree in electrical engineering from VRCE Nagpur University, Nagpur, India, in 1993, the M.Tech. degree from the Indian Institute of Technology, Kanpur, India, in 1996, and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2000.

Presently, he is an Associate Professor at the University of Alberta, Edmonton, AB, Canada. His research interests include electromagnetic transients, power electronics, and real-time digital simulation and control.

Dr. Dinavahi is a Professional Engineer in the Province of Alberta.