Design of a Real-Time Digital Simulator for a D-STATCOM System

Venkata Dinavahi, *Member, IEEE*, Reza Iravani, *Fellow, IEEE*, and Richard Bonert, *Member, IEEE*

***Abstract*—Real-time digital simulation of power electronic systems requires significant computational resources due to increasingly complex system configurations, control algorithms, and higher switching frequency. Consequently, it is prudent to exploit various computer resources for optimizing the design of simulators/controllers for such systems. This paper presents the design and implementation details of a real-time digital simulator for a Voltage-Source-Converter-based Distribution STATic COMpensator (D-STATCOM) power system. The design process adopts a modular approach utilizing distributed digital signal processor/field-programmable gate array resources of a digital processing platform. The design has been validated by using an experimental setup of a 5-kVA D-STATCOM system.**

***Index Terms*—Digital control, digital signal processor (DSP), field-programmable gate array (FPGA), power electronics, real-time simulation.**

I. Introduction

POWER SYSTEMS are constantly evolving to include new technologies for controlling the flow of power using power electronics and for improving the reliability of networks using advanced protection strategies. Real-time simulation provides a solid framework to test the new control/protection concepts so as to detect, analyze, and correct any potential problems before commissioning. Recently, there has been significant research effort in this area [1]–[6]. The objective of the real-time simulator design presented in this paper is twofold: first, to prove the viability of accurate and efficient algorithms for the real-time simulation of power electronic systems, and second, to verify digital control systems designed for such systems before they can be applied to physical systems.

Real-time digital simulation of power electronic systems is a heavily computer intensive operation owing to their size, modeling complexity, and higher frequency of switching. Implementation of complex simulation algorithms, control algorithms, signal processing such as filtering and data conversion, communication with the external system as well as with the user interface, diagnostic and protective functions require a vast amount of both concentrated as well as distributed computational resources. Among a host of available digital processors for designing highly efficient, low-cost, stand-alone real-time digital simulators [7]–[10] and controllers, two processors that are gaining popularity are digital signal processors (DSPs) and field-programmable gate arrays (FPGAs).

Since the early 1990s there has been rapid progress in DSP technology with regard to their processing power, internal memory, communication capabilities, power management, and software programming tools. Currently, there are several manufacturers that offer diverse DSP architectures to help developers choose the device that best suits their applications. The main advantage of using DSPs lies in their software implementation which makes the design flexible, extensible, and easy to use.

FPGAs were first introduced in the mid 1980s. Since then there has been a steady advance in their performance and density and their cost has progressively gone down, enabling them to compete with other digital processors on the market. Most FPGAs are organized as: an array of basic logic elements and programmable interconnections between the logic elements, memory, and I/O pins. The configuration of an FPGA is carried out by downloading a bit stream into a Static RAM (SRAM) memory inside the FPGA which defines the functions of each logic element and the interconnections between them. Due to the inherent parallelism a FPGA provides superior performance compared to a microprocessor or a DSP. However, a design involving FPGAs usually lacks the flexibility that comes with a software implementation.

This paper describes the design and implementation of a real-time digital simulator based on an integrated DSP/FPGA platform. The study system is a Voltage-Source-Converter (VSC)-based Distribution STATic COMpensator (D-STATCOM) power system. The same digital processing platform was utilized to develop a real-time controller for an experimental 5-kVA D-STATCOM system.

The paper is organized as follows. Section II gives an overview of the main hardware and software features of the digital signal processing platform. Section III describes the design and implementation details of a real-time simulator and controller. Section IV describes the experimental setup used to validate the real-time simulation. Results and discussions are given in Section V followed by the conclusion in Section VI.

II. DSP/FPGA Digital Processing Platform Features

The structure of the digital signal processing platform which is referred to as UHP40 is shown in Fig. 1. The main hardware components of the platform include a 32-bit floating point TI C40 DSP, a FLEX8000 FPGA from ALTERA, Communication ports, and memory. The FPGA shares the global address and

Manuscript received August 29, 2003; revised March 4, 2004. Abstract published on the Internet July 15, 2004. This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

V. Dinavahi is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada (e-mail: dinavahi@ece.ualberta.ca).

R. Iravani and R. Bonert are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: iravani@ecf.utoronto.ca; bonert@ecf.utoronto.ca).

Digital Object Identifier 10.1109/TIE.2004.834954

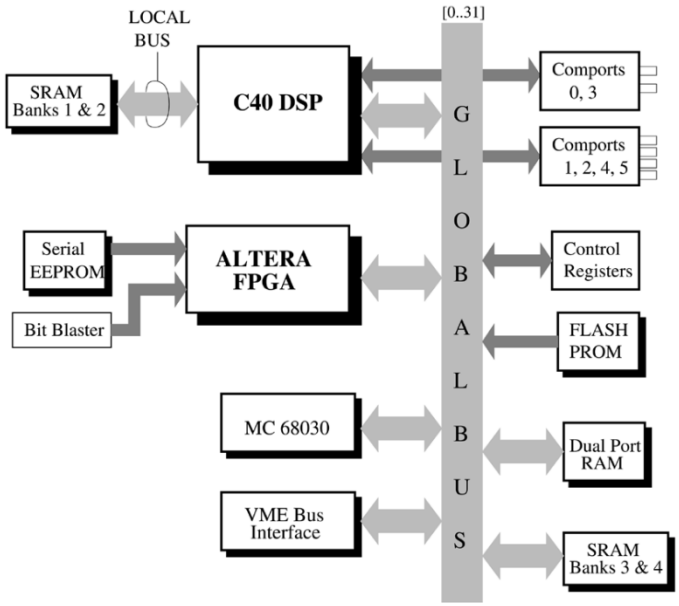

Fig. 1. Functional block diagram of the UHP40 digital processing platform.

data space of the DSP. A global bus interface has to be programmed in the FPGA for communication with the DSP. The platform also consists of a Motorola CPU MC68030 and a VME bus interface, however, this feature was not used in the design of the real-time simulator.

The software requirements for the UHP40 include: 1) a user program; 2) DSP Kernel program; 3) DSP Monitor program. The user program and the DSP Kernel program run on the DSP while the Monitor program runs on the PC. The user program consists of two C functions, `user_function ()` and `user_init ()`. The `user_function ()` contains the main simulation/control code that the user wants to run in real time. During initialization the Kernel program calls the `user_init ()` function which contains the initialization variables and a pointer to the `user_function ()`. The Monitor program running on the PC emulates a real-time oscilloscope. With its help the user cannot only display the signals internal to the DSP but also change the parameters of the system online while the real-time simulation/control is in progress.

III. Design of the Real-Time Digital Simulator

The real-time digital simulator has been designed to simulate a six-pulse pulsewidth-modulated (PWM) VSC-based D-STATCOM power system utilizing the UHP40 platform as the principal computational engine. In this section, an overview of the entire design is first presented. Thereafter, the hardware and software features of the design are systematically described.

A. Overall Design

The design of a real-time digital simulator for the D-STATCOM system comprises of four major modules: 1) System Simulator module; 2) Digital Controller module; 3) PWM module; and 4) Switching Event Capture (SEC) module. The System Simulator module is where the mathematical equations describing the system are solved in real time at a fixed time step Δt. The Digital Controller implements the control algorithm at a fixed sampling period T_s. The PWM module generates the firing signals for the power electronic converter and the SEC module captures the timing of those signals and sends them to the simulator.

The design, shown in Fig. 2, is comprised of one UHP40 platform monitored by one User Interface. The Simulator and the Controller modules are configured in the DSP while the PWM and SEC modules are assigned to the FPGA. There are two main data transfer interfaces: one internal to the DSP, which transfers the system state samples to the Controller, and the other internal to the FPGA, which transfers firing signals from the PWM module to the SEC module. The bidirectional data exchange between the DSP and the FPGA is managed by the Global Bus Interface in the FPGA.

B. Digital Hardware Design for the FPGA

The design proceeds in a hierarchy with the main design file consisting of the three subdesigns for: 1) Global Bus Interface; 2) Pulsewidth Modulator (PWM); and 3) SEC.

1) Global Bus Interface: The primary function of this module (Fig. 3) is to arbitrate bidirectional data transfer between the DSP and the FPGA. In addition to the communication signals it provides Δt and T_s signals to the PWM and SEC modules to synchronize their respective operations.

For data transfer from the Controller module in the DSP to the PWM module in the FPGA, the control values are first clocked into a set of 16-bit registers whenever the DSP finishes its calculation cycle. A second set of registers takes the control values from the first set at the beginning of each sampling period T_s. This operation involves a delay of one T_s which constitutes the *controller dead time* and ensures proper transfer of the control values.

The data transfer from the SEC module in the FPGA to the Simulator module in the DSP utilizes a tri-state buffer where the gating signals and their timing are stored before being transferred to the DSP at the beginning of each time step Δt. A `watchdog` macro function makes sure that the control values coming in from the DSP are refreshed every T_s.

2) Pulsewidth Modulator: Sinusoidal PWM has been implemented in this subdesign (Fig. 4). A modified triangular carrier waveform synchronized with the controller sampling period T_s is compared with the control value sent from the Controller module in the DSP. The carrier wave is represented by a 16-bit counter value.

At the beginning of every T_s the counter is preloaded with a positive/negative limit. The counter starts counting down/up till the opposite limit is reached when it holds the count for 4 μs. The hold operation creates a flat top on the carrier wave and determines the minimum pulse width of the resulting gating signal. At the next sampling instant T_s an opposite limit is preloaded and the counter counts in the other direction till the limit is reached when it again holds for 4 μs. The counter limit u is initialized in the DSP program and is given as follows:

$$u = T_s \star f_{\text{clock}}/4 \tag{1}$$

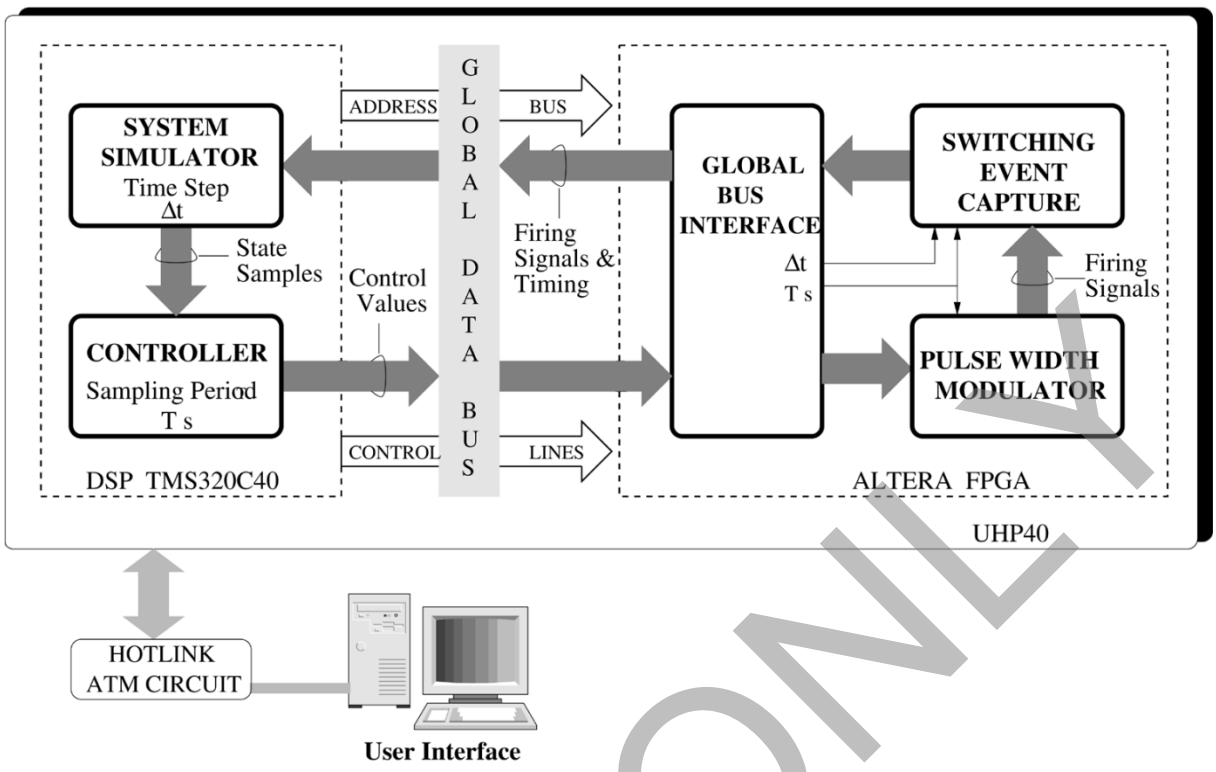

Fig. 2. Real-time digital simulator and controller setup for the D-STATCOM system.

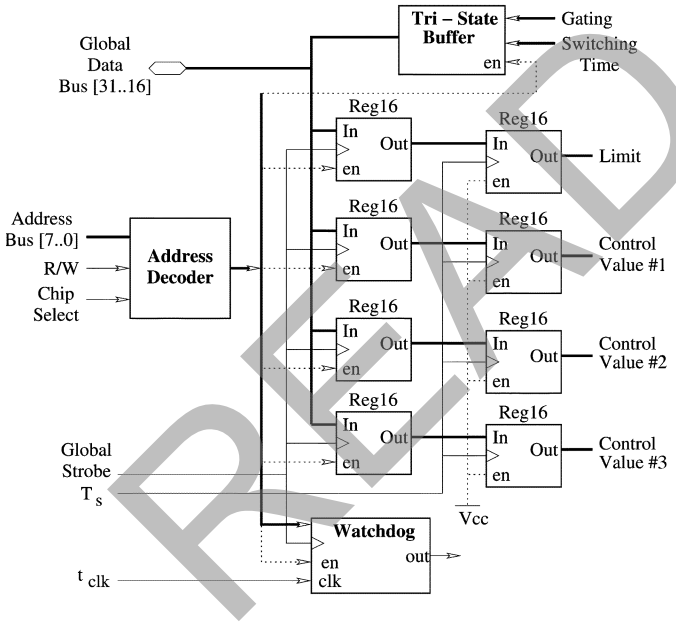

Fig. 3. Global bus interface.

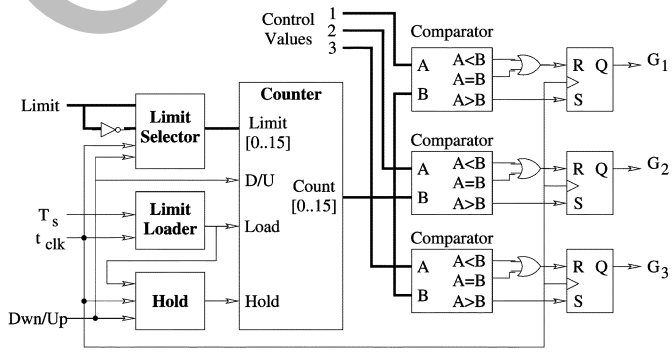

Fig. 4. Digital PWM.

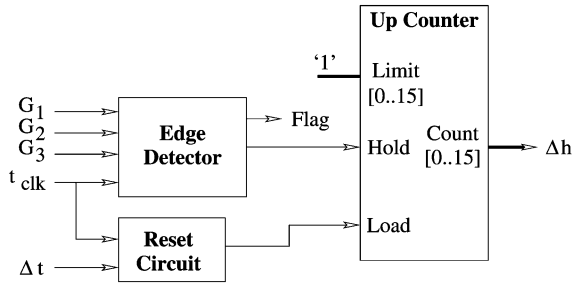

Fig. 5. Switching event capture.

where $T_s = 1/f_{\text{sw}}$ and $f_{\text{clock}} = 20$ MHz. The gating pulses are stored in an output buffer in the Global Bus Interface and transferred to the Simulator module at the beginning of every Δt. The output pins of the FPGA are accessible for the user to examine the firing signals using an oscilloscope.

3) SEC: The purpose of this module is to put a time stamp on the firing pulses coming from the Modulator. It registers the location of the firing pulses with respect to the beginning of every sampling instant T_s. The capture mechanism simply consists of an edge detection circuit and a 16-bit counter as shown in Fig. 5. The resolution of the counter is 50 ns/count. Every rising edge of the Δt signal the counter is reset and starts counting up. When a switching event is detected by the edge detector, the `flag` is set and a `hold` signal is sent to the counter. The counter value and the `flag` are held in an output buffer in the Global Bus Interface until the end of the time period Δt. They are sent to the Simulator module in the DSP at the beginning

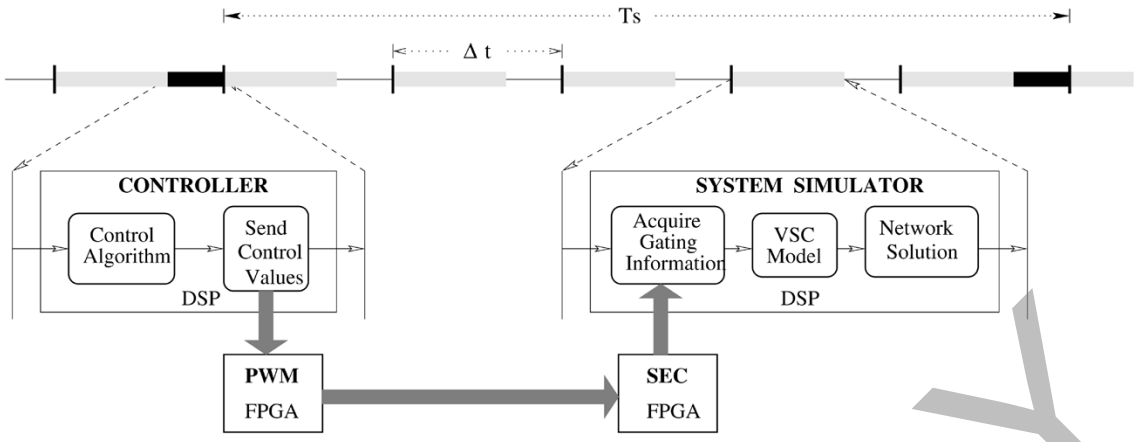

Fig. 6. Real-time program structure.

of the next simulation time step. Based on the state of the firing pulses from the PWM module, the counter value and the `flag`, the simulator can precisely determine the occurrence and location of any event in the previous time step.

C. Software Design for the C40 DSP

The software program for the real-time simulation and control of the D-STATCOM system consists of two functions `user_function` and `user_init` written in C. During compilation this program is linked with the Kernel program to produce an executable code which is then downloaded to the DSP. The `user_function` consists of two main parts: 1) System Simulator module and 2) Controller module. The real-time program structure is shown in Fig. 6. The System Simulator code is executed as an interrupt service routine on the DSP at every time step Δt while the Controller code is executed once every sampling period T_s. The simulation can run indefinitely on the DSP once it is initialized. The user has full control of all the parameters of the simulation and can view the simulation variables using the Monitor program on the PC. The `user_init` function is used to initialize the real-time simulation on the DSP. It defines the initial values of all the system and control variables, fixed parameters of simulation, time step Δt, sampling period T_s, and PWM information. It also defines FPGA address space, SRAM address where the control values are sent to an address from where the gating information can be acquired. Computations involving fixed parameters of the simulation, for example, the admittance matrix of a linear network, may also be defined in `user_init` to optimize the `user_function`. In addition, one period of a fundamental sinusoidal function is calculated and stored in a lookup table which may be used as an ac source in the simulation.

1) System Simulator: As shown in Fig. 6 at the beginning of every time step Δt the code for this module performs the following three functions in the given sequence: (A) acquires PWM gating information from the FPGA, (B) decides the status of the six switches in the VSC model, and (C) solves the network equations. The gating information communicated to this module from the FPGA via the Global Bus Interface comprises of the gates $G_k(k = 1, 2, 3)$, the `flag` and the time Δh of any gate transition that occurred in the previous time step Δt. The

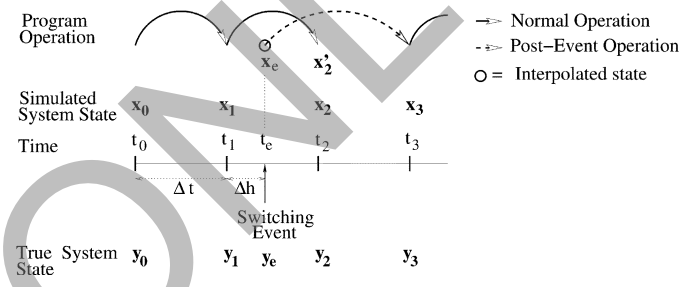

Fig. 7. FICS real-time simulation algorithm.

insulated gate bipolar transistor (IGBT) switches in the VSC are modeled as ideal switches. The VSC model is based on discrete switching functions $S_k(t)$.

$$v_{kN}(t) = S_k(t) * v_{\rm dc}(t), \qquad k = a, b, c \tag{2}$$

where $S_k(t) = 1$ if $G_k = \text{high}$ and $S_k(t) = 0$ if $G_k = \text{low}, G_k$ are the gating signals. The *R–L* network on the ac side of the converter is represented by three differential equations

$$L\frac{di_k}{dt} + Ri_k(t) = v_k(t) - e_k(t), \qquad k = a, b, c. \tag{3}$$

The dc-side equations are

$$-C\frac{dv_{\rm dc}}{dt} = i_{\rm dc}(t) + i_l(t) \tag{4}$$

$$i_{\rm dc}(t) = \sum_k S_k(t) * i_k(t), \qquad k = a, b, c. \tag{5}$$

The network components are assumed to be linear and the network solution utilizes the Fixed step size with Interpolation and Clock Synchronization (FICS) algorithm [11] shown in Fig. 7 to correct the system state depending on whether a switching event has been detected in the previous calculation step. The FICS algorithm uses linear interpolation for the state correction procedure. Based on the status of the `flag`, at the beginning of every time step the simulator performs either the Normal Operation (if `flag` is low) or the Post-Event Operation (if `flag` is high). The trapezoidal numerical method and nodal analysis are used to produce a set of linear algebraic equations which are then solved for the system state at each time step.

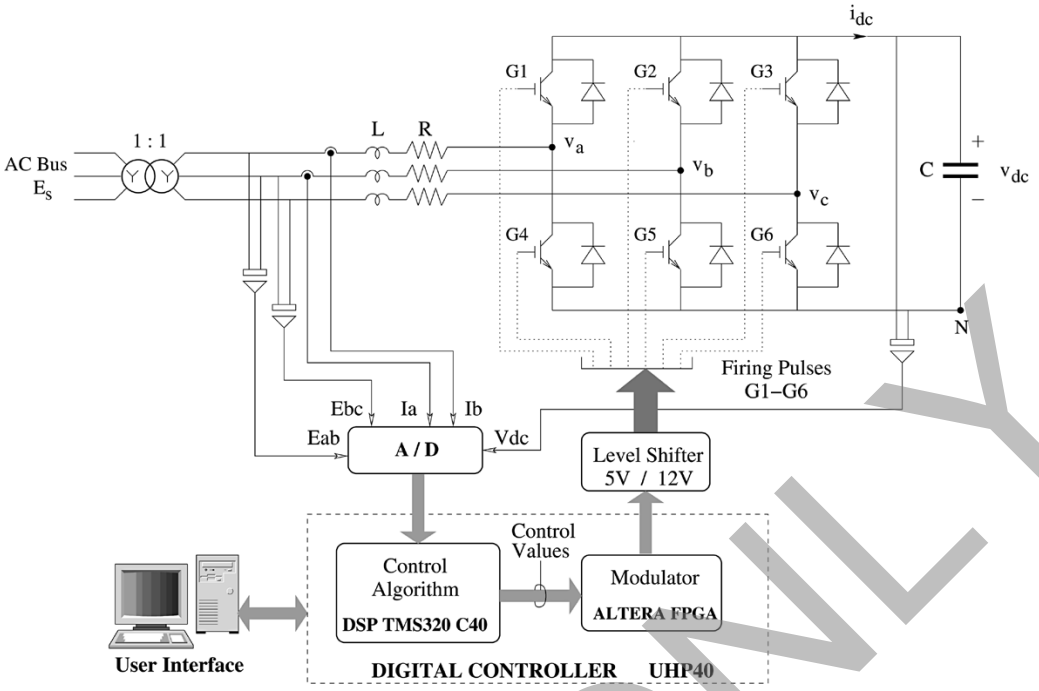

Fig. 8. Experimental setup for the D-STATCOM system.

Once the System Simulator module finishes its calculation the system signals are transferred to the Controller module at the beginning of every T_s.

2) Controller: On receiving the system signals the Controller Module executes the digital control algorithm and transfers sinusoidally modulated control values to the PWM Module in the FPGA. The control algorithm proceeds by transforming the system signals into a synchronous d–q reference frame and then using PI compensator's to individually regulate the converter ac-side currents and the dc-link voltage. The system model in the d–q frame can be expressed as

$$\frac{di_d}{dt} = -\frac{R}{L}i_d + \omega i_q + \frac{1}{L}(v_d - e_d) \tag{6}$$

$$\frac{di_q}{dt} = -\frac{R}{L}i_q - \omega i_d + (v_q - e_q) \tag{7}$$

$$\frac{dv_{\rm dc}}{dt} = -\frac{3}{2}\frac{(v_d i_d + v_q i_q)}{C v_{\rm dc}} - i_l \tag{8}$$

where e_d and e_q are the d and q components of the ac-bus voltages $\overrightarrow{e}$ ($e_d = |\overrightarrow{e}|$ and $e_q = 0$) and v_d and v_q are the components of the converter output voltages expressed as

$$v_d = 0.5 m_a v_{\rm dc} \cos\delta \tag{9}$$

$$v_q = 0.5 m_a v_{\rm dc} \sin\delta \tag{10}$$

where $m_a = \sqrt{(e_d^2 + e_q^2)}/(0.5 * v_{\rm dc})$ is the Modulation Index and $\delta = \arctan(e_q/e_d)$ is the phase angle between the fundamental component of the converter output voltage and the ac-bus voltage. A decoupled control [12] of i_d and i_q is performed using the proportional plus integral (PI) compensators of the form $G_i(s) = (K_i(1 + sT_i))/(sT_i)$ for each of the current loops. The dc capacitor voltage is regulated using a PI controller in an outer feedback loop. The reactive current reference may be derived from the reactive power requirement of the converter while the real current reference is obtained from the output of the outer feedback control loop that regulates $v_{\rm dc}$. Based on $m_a, \delta, v_{\rm dc}$ and the reference ac voltage vector $\overrightarrow{e}$ three sinusoidally modulated control values are generated and are sent to the PWM module in the FPGA.

IV. Experimental Setup for the D-STATCOM System

A laboratory prototype of a 5-kVA D-STATCOM system was built to verify the real-time simulation results. Fig. 8 shows the experimental setup. The system parameters are given as follows:

- three-phase supply $E_s = 110$ V$_{ll}$, 60 Hz;
- three-phase transformer Y/Y, 10 kVA, 115 V, 60 Hz, $X_T = 3.06\%, R_T = 2.71\%$;
- $R = 0.5\ \Omega, L = 3.0$ mH, $C = 4900\ \mu$F;
- VSC composed of a six-pack 1200-V 50-A IGBT module.

The UHP40 platform described in Section II was used to implement the experimental digital controller. The control algorithm was executed on the C40 DSP while the PWM was implemented on the FPGA. Data acquisition of five system signals (three voltages and two currents) is done by dedicated A/D boards connected to the Comports on the UHP40. The user has full control of the system using the Monitor program running on the PC. The FPGA hardware design used for the experimental setup consists of three subdesigns: 1) Global Bus Interface; 2) Pulsewidth Modulator; and 3) Dead Time Insertion Module.

The Global Bus Interface is the same as the one in Fig. 3 with the exception that now the data transfer is the unidirectional flow of control values from the DSP to the FPGA, resulting in a simpler design. The PWM is identical to the one used for the real-time simulator. The Dead Time Insertion module inserts a 2-μs dead time T_d between the gating signals of the upper and lower switches of each converter leg in order to prevent a short circuit of the dc bus. During the dead time both switches are in

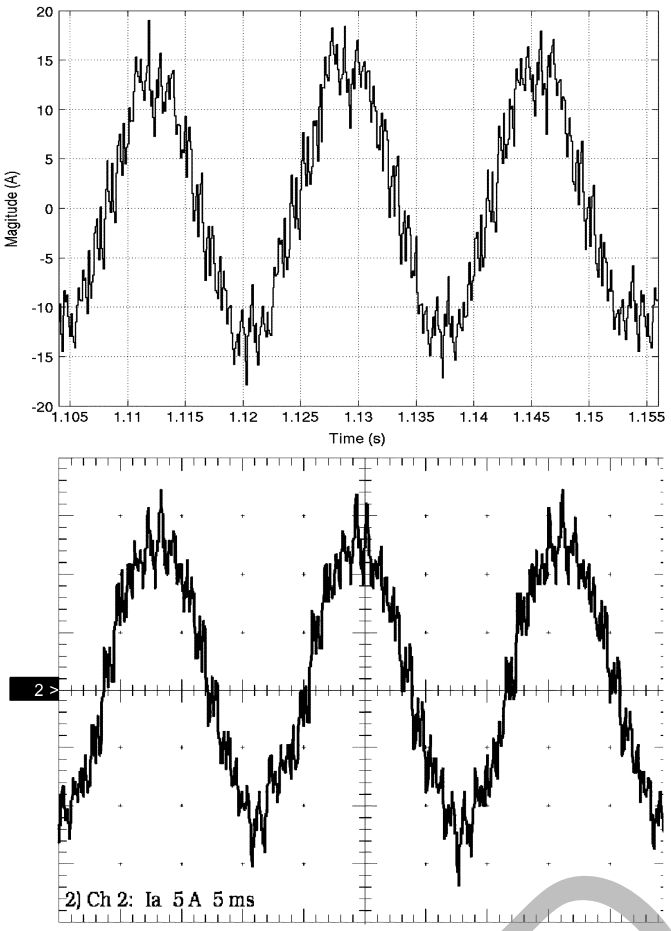

Fig. 9. Steady-state current i_a from real-time simulation (top), and experiment (bottom).

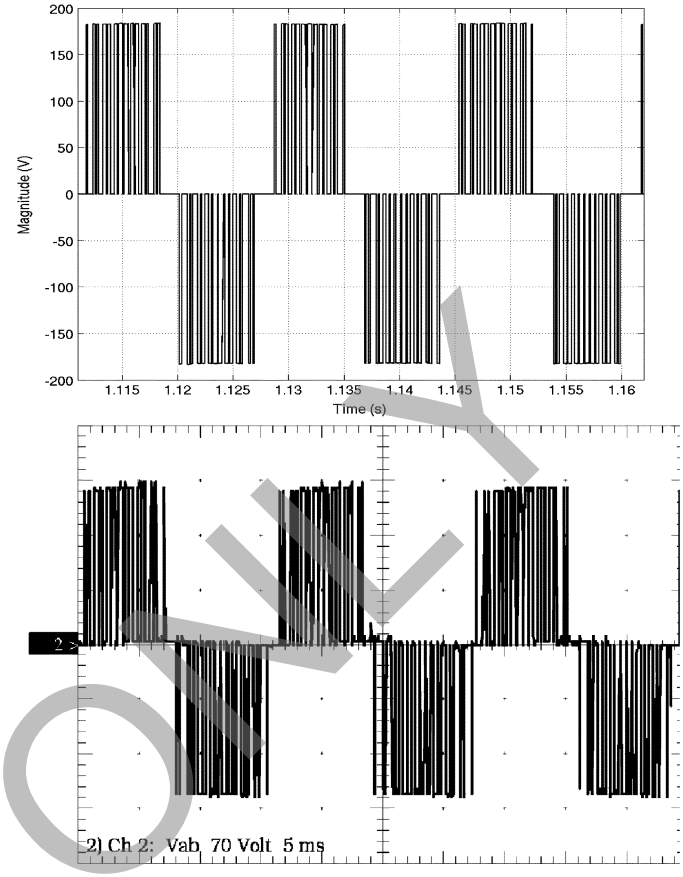

Fig. 10. Steady-state line-to-line voltage v_{ab} from real-time simulation (top), and experiment (bottom).

the off position. The counter limit u for the Modulator is altered to include the dead time T_d and is initialized in the DSP program as

$$u = \frac{(T_s - 2 * T_d) * f_{\text{clock}}}{4}. \tag{11}$$

The software program executed on the DSP to control the D-STATCOM system implements the control principle described in Section III-C.2. The program includes some extra code for signal acquisition using the A/D converters and for digital low-pass filtering. The PWM carrier frequency $f_{\text{sw}} = 1/T_{\text{sw}}$ is 1 kHz and the controller sampling period $T_s(= 2 * T_{\text{sw}})$ is 500 μs. The gating signals coming out from the FPGA are shifted to a higher voltage level to make them compatible with the gate drive circuit of the IGBT module. The FPGA design has a built-in protection feature to disable the gating pattern generation in case of any maloperation. The Monitor program on the PC allows a full two-way communication with the system.

V. Results and Discussion

This section presents results obtained from the experimental (physical) setup and the real-time simulator setup of the D-STATCOM system. Open-loop and closed-loop control tests were conducted under similar operating conditions for both setups. The parameters of the system used in the simulation were measured from the experimental setup. A 100-μs time step was chosen for the real-time simulation. This choice was based on the complexities of the simulation algorithm, the system equations, and the performance of the C40 DSP.

Figs. 9–11 show the open-loop ac line current, line-to-line voltage, and line-to-neutral voltage, respectively, of the VSC under steady-state operation. These figures illustrate the close agreement of the results obtained from the real-time simulation and the experimental setup.

Under closed-loop control conditions step response of the i_q controller under real-time simulation [Fig. 12(a) and (b)] and experimental conditions [Fig. 12(c) and (d)] was obtained. The performance of the closed loop controller was first tested by offline simulation. The tuned PI controller parameters were then employed in the real-time simulation and the experimental setup. The closed-loop results demonstrate adequate tracking and stability in both sets of results. Finally, an offline time domain simulations using the fixed step-size method with a small time step (10 μs) were conducted and the results were found to be in good agreement with the above results providing a high level of confidence in the real-time simulation. The factors that account for differences in the measured and simulated results are as follows.

1) Converter switches in the offline simulation program as well as the real-time simulation were treated as ideal switches and switches in the same converter leg were

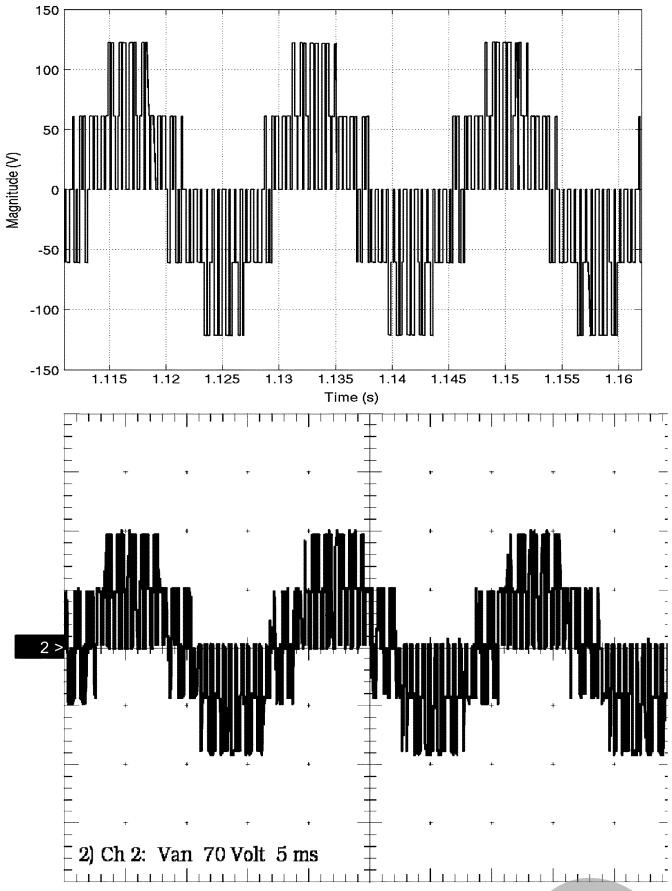

Fig. 11. Steady-state line-to-neutral voltage v_{an} from real-time simulation (top), and experiment (bottom).

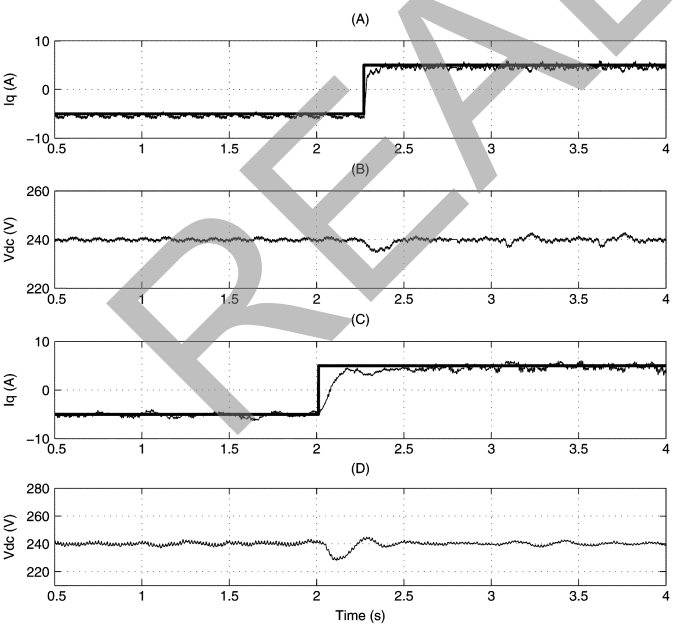

Fig. 12. i_q transient under closed-loop control. (a), (b) Real-time simulation. (c), (d) Experiment.

turned on/off simultaneously ignoring the PWM dead time. However, in the experimental setup a 2-μs dead time was programmed in the FPGA not to short circuit the dc bus. Inclusion of this effect in the real-time simulation would have necessitated a far smaller time step than 100 μs, thus preventing real-time operation.

2) Asynchronous PWM, i.e., $m_f = f_{\rm sw}/f_1$ is a noninteger, is used for the experimental setup. To implement synchronous PWM the switching frequency $f_{\rm sw}$ must be made variable as the line frequency f_1 changes to keep the ratio m_f an exact integer. Since $f_{\rm sw}$ is synchronized to the controller sampling period T_s that would mean changing T_s online. However, changing T_s was not feasible due to a fixed sampling period which needs to be specified during the DSP initialization. Although the PWM implemented for the real-time simulator was also asynchronous the sources were considered ideal and frequency was fixed at 60 Hz, whereas in the experimental setup the line frequency had constant variations of small amounts which have effects on the results.
3) Unbalance in the system voltages, parameters, and stray inductances and capacitances also contribute to a little offset in the experimental results.
4) In the experiment *R*'s and *L*'s are frequency dependent due to skin effect and iron losses in inductors. However, these effects were not included in the simulation.

A. Execution Time

Two functions defined in the Kernel program enable the control of status of a specific bit in the Output Register located in Memory Bank 1. The execution time of any module code defined in the `user_interrupt` function can be measured by setting the bit at the beginning of the code and resetting it at the end of the code. The time taken by the C40 DSP to execute each of the modules described in Section III.A was measured in this manner. Out of a time step of 100 μs, 84 μs was used for simulating the D-STATCOM system and 16 μs overhead was used for tasks such as initializing the DSP kernel (4 μs) and managing data acquisition and display of up to three signals (12 μs) on the PC. The simulation time comprises of 36 μs used by the Simulator module and 48 μs used by the Controller module. In the Simulator module the Normal mode of operation (fixed step size) took 16 μs to execute while the Post-Event mode took 26 μs. The FPGA operated at a clock frequency of 20 MHz. The PWM and the SEC in the FPGA are carried out in parallel with the simulation in the DSP. In comparison to the simulation time on the DSP, the FPGA execution times were negligibly small.

VI. Conclusion

This paper presented the design and implementation details of a real-time digital simulator and controller for a PWM VSC-based D-STATCOM power system. The design follows a modular approach based on an integrated DSP-FPGA platform, thereby combining the flexibility and processing power of both the software and hardware features of the platform. Experimental tests on a 5-kVA D-STATCOM power system validate the real-time simulator results.

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 Delivery*, vol. 14, pp. 773–781, July 1999.

[2] O. Devaux, L. Levacher, and O. Huet, "An advanced and powerful real-time digital transient network analyzer," *IEEE Trans. Power Delivery*, vol. 13, pp. 421–426, Apr. 1998.

[3] A. G. Jack, D. J. Atkinson, and H. J. Slater, "Real-time emulation for power equipment development. I. Real-time simulation," *Proc. IEE—Elect. Power Applicat.*, vol. 145, no. 2, pp. 92–97, Mar. 1998.

[4] H. Le-Huy, G. Sybille, P. Giroux, J. C. Soumagne, and F. Guay, "Digital real-time simulation of a four-quadrant DC drive for static transfer switch testing," in *Proc. IEEE Power Engineering Soc. Winter Meeting*, vol. 1, Feb. 1999, pp. 761–765.

[5] J. Parle, E. Acha, and C. R. Fuerte-Esquivel, "Real-time digital simulation of electromagnetic transient phenomena in power transmission lines," in *Proc. Int. Conf. Advances in Power System Control, Operation, and Management (APSCOM)*, vol. 2, Nov. 1997, pp. 563–568.

[6] J. R. Marti and L. R. Linares, "Real-time EMTP-based transients simulation," *IEEE Trans. Power Syst.*, vol. 9, pp. 1309–1317, Aug. 1994.

[7] R. C. Durie and C. Pottle, "An extensible real-time digital transient network analyzer," *IEEE Trans. Power Syst.*, vol. 8, pp. 84–89, Feb. 1993.

[8] P. J. Throckmorton and L. Wozniak, "A generic DSP-based real-time simulator with application to hydrogenerator speed controller development," *IEEE Trans. Energy Conversion*, vol. 9, pp. 238–242, June 1994.

[9] Y. Guo, H. C. Lee, and B. T. Ooi, "Extensible digital-signal-processing modules for real-time control and simulation," in *Proc. IEEE IECON'93*, vol. 3, Nov. 1993, pp. 2229–2234.

[10] P. G. McLaren, R. Kuffel, R. Wierckx, J. Giesbrecht, and L. Arendt, "A real time digital simulator for testing relays," *IEEE Trans. Power Delivery*, vol. 7, pp. 207–213, Jan. 1992.

[11] V. R. Dinavahi, M. R. Iravani, and R. Bonert, "Real-time digital simulation of power electronic apparatus interfaced with digital controllers," *IEEE Trans. Power Delivery*, vol. 16, pp. 775–781, Oct. 2001.

[12] C. Schauder and H. Mehta, "Vector analysis and control of advanced static VAR compensators," *Proc. Inst. Elect. Eng.*, pt. C, vol. 140, no. 4, pp. 299–306, July 1993.

Venkata Dinavahi (S'94–M'00) received the B.Eng. (1993) in electrical engineering from 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 Assistant Professor at the University of Alberta, Edmonton, AB, Canada. His research interests include electromagnetic transient analysis, power electronics, real-time simulation, and control.

Dr. Dinavahi is a Member of CIGRÈ and a Professional Engineer in the Province of Alberta, Canada.

Reza Iravani (M'85–SM'00–F'03) received the B.Sc. degree from Tehran Polytechnic University, Tehran, Iran, in 1976, and the M.Sc. and Ph.D. degrees from the University of Manitoba, Winnipeg, MB, Canada, in 1981 and 1985, respectively, all in electrical engineering.

Following a career as a Consulting Engineer, he is currently a Professor at the University of Toronto, Toronto, ON, Canada. His research interests include power electronics and power system dynamics and control.

Richard Bonert (M'81) received the Dipl.-Ing. and Doctorate degrees in electrical engineering from the University of Karlsruhe, Karlsruhe, Germany, in 1969 and 1977, respectively.

He joined Brown Boveri Germany in 1969 as a Project Engineer. He was also with the Elektrotechnisches Institut of the University of Karlsruhe as a Research Associate and Chief Engineer. In 1980, he joined the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, where he is currently a Professor. His main interests are in the fields of electrical drives, automatic feedback control systems for electric drives, and power electronic circuits.