A General Framework for FPGA-Based Real-Time Emulation of Electrical Machines for HIL Applications

Nariman Roshandel Tavana, *Student Member, IEEE*, and Venkata Dinavahi, *Senior Member, IEEE*

***Abstract*—Hardware-in-the-loop (HIL) technology is increasingly becoming the preferred, reliable, and cost-effective alternative in a virtual scenario for tedious, time-consuming, and expensive tests on real devices. This paper presents a digital hardware emulation of commonly used electrical machines for HIL simulation on the field-programmable gate arrays (FPGAs) in a general framework. This paper provides a useful and comprehensive comparison between floating- and fixed-point arithmetic for hardware implementation, and addresses the differences of deeply pipelined and highly paralleled realization schemes, and the contribution of schematic and textual programming language methods for design configuration of electrical machine models. The hardware implementation by these approaches is evaluated in terms of real-time step size, accuracy, and hardware resource consumption. Finally, an experimentally measured electrical machine behavior is employed to demonstrate the effectiveness of the emulated electrical machine.**

***Index Terms*—Electrical machines, field-programmable gate arrays (FPGAs), hardware-in-the-loop (HIL) simulation, real-time systems.**

Nomenclature

s, r, f, k	Stator, rotor, field, and damper indexes.
d, q	Reference frame index.
l, m	Leakage and magnetizing indexes.
V, I, λ	Voltage, current, and flux linkages.
T_e, T_{mech}	Electromagnetic torque and mechanical torque.
T_{damp}	Damping torque.
r, L	Resistance and inductance.
ω, θ	Angular speed and rotor position.
J, P	Rotor inertia and the number of pole pairs.
a, b, c	Phase-domain indexes.
$\mathbf{A}_{h\times h}, \mathbf{B}_{h\times p}$	State matrix and input matrix.
$\mathbf{C}_{q\times h}, \mathbf{D}_{q\times p}$	Output matrix and feedforward matrix.

Manuscript received April 1, 2014; revised July 14, 2014 and August 18, 2014; accepted September 6, 2014. Date of publication October 3, 2014; date of current version March 6, 2015. This work was supported in part by the Natural Science and Engineering Research Council of Canada.

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

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2014.2361314

I. Introduction

OFFLINE transient simulation of electrical machines using software such as MATLAB/Simulink or electromagnetic simulation tools such as JMAG or ANSYS has been very successful for many years. Although execution time is still a matter of concern in such offline tools, it is not as critical as in a real-time digital simulator, which has to interact with external devices in a hardware-in-the-loop (HIL) scenario. The main application of real-time emulation of electrical machines is to evaluate the behavior of newly designed machines, drive systems, controllers, and protective devices in a HIL configuration in an effective and economic approach before applying them in a real system [1]–[6]. Such testing allows the system components to be subjected to extreme conditions in a nondestructive environment and in an expedited manner. To reproduce electrical machine transients with high fidelity, an accurate modeling of the machines with a small simulation time step is a necessity for the real-time simulator. To meet stringent real-time step-size constraints, a compromise is usually made between the accuracy and complexity of the system model [5].

Owing to the rapid developments and dramatic advances in digital hardware technology, the field-programmable gate array (FPGA) is becoming the fastest, most reliable, and preferred computational engine for digital hardware realization of complex systems without sacrificing the accuracy [7]–[15]. Today, the FPGA has gained a crucial role in the HIL simulation and rapid control prototyping of electrical machines and drive systems employed in the industrial applications. According to their paralleled hardwired architecture, reconfigurability, large amount of logic resources, full-custom digital-signal-processing (DSP) units, and storage elements, currently available FPGA devices are able to satisfy the accuracy demands of machine models adequately by providing a nanosecond computational clock cycle within the simulation time step in real time.

A number of studies have been conducted in this area with different objectives [16]–[21]. Most methodologies adopted in the literature for the real-time simulation of machines on FPGAs are based on fixed-point calculations [16]–[20], and a 32-bit floating-point hardware emulation of a synchronous generator used in the nodal analysis for power-system transient simulation is presented in [21]. Since the need for a comprehensive comparison between fixed- and floating-point implementation, deeply pipelined and paralleled architecture, and schematic and textual programming language (TPL) methods have not so far been met, in this paper, the focus is to evaluate

FPGA-based real-time emulation of the machine models in a general framework. The remainder of this paper is organized as follows. Section II gives an overview of FPGA design methodologies. Section III explains the mathematical modeling and numerical techniques for digital realization of electrical machines. Implementation of the machine model by different approaches are presented in detail in Section IV. Section V evaluates the various designed architectures in terms of real-time simulation time-step sizes and hardware resource consumption and accuracy. The effectiveness and usefulness of FPGA-based real-time emulated machine models are assessed and highlighted by providing a number of case studies (for induction motors, synchronous generators, line-start permanent-magnet synchronous motors (LSPMSM), and dc motors) in Section VI. Section VII gives the main conclusions of this paper.

II. Overview of FPGA Design Approaches for Electrical Machine Realization

The techniques to emulate a system on FPGAs can be broadly classified into two groups: 1) TPLs, such as a hardware description language (HDL); and 2) a schematic method from vendor-specific block sets. The textual programming method by means of HDL such as Verilog HDL (VHDL) is a powerful method to develop a digital hardware design without any restrictions. However, it can be very complex and cumbersome to debug. Even experts in machine modeling and simulation may find programming in the HDL a daunting task. The schematic or model-based method relies on a library of basic combinatorial and sequential building blocks offered in the Altera DSP Builder, Xilinx System Generator (XSG) block sets, etc., within the MATLAB/Simulink environment. This method allows users to go from system simulation using the industry-standard Mathworks simulation tools to hardware implementation in a short time. This method is user-friendly, easier to troubleshoot, easily understandable, and specifically useful for novice users of embedded digital systems. Expert users will of course be able to save development time by adopting intellectual property (IP) core blocks already available in such block sets, and no HDL hard-coding is required. However, this approach is still limited to applications in which no complex sequencers and deeply pipelined structures are required [22].

Furthermore, the FPGA is a space-oriented logic device that enables full hardwired parallelism to be achieved to the extent permitted by the implemented user model and algorithm. A large number of customized parallel processing units can be easily configured. Thus, a highly parallel implementation is the best realization according to the FPGA architecture (one data per hardware module per time step) for the algorithms where a lower amount of hardware resources for computational processing is required, such as fixed-point arithmetic-based designs. The integrated massive memory blocks can be partitioned into many independent types such as RAM, read-only memory, firt-in–first-out, single-port, or dual-port user memory units through which multiple data can be accessed simultaneously. Although multiple data can be processed in parallel on FPGAs, due to resource limitation, it is difficult to achieve the ideal parallelism for large systems and is not area-optimized for the algorithms that utilize a massive amount of hardware resources such as floating-point-calculation-based designs. In such cases, the pipelining technique has to be used. In a pipelined scheme, a function is divided into several stages by inserting registers between stages, allowing multiple data to be processed at different stages at once and resulting in a high computational throughput (multiple data per hardware module per time step).

As a consequence of the above reasons, the seamless and user-friendly schematic method is the best technique for a highly parallel implementation of a real-time machine model based on fixed-point operations, whereas on the other hand, the TPL method is the most appropriate approach for the pipeline realization of a floating-point-calculation-based real-time system, resulting in an area-optimized hardware architecture acceptable in the industrial applications.

In this paper, floating-point number calculations in the deeply pipelined scheme are employed for an FPGA-based real-time emulation of commonly used electrical machines to support a wide range of machine specifications and characteristics. Moreover, a fixed-point algorithm of the models is implemented for a complete comparison. The implementation is carried out by a state-space approach to provide a unified framework. Therefore, the real-time emulation of electrical machines can be realized by this approach, and the presented methodology can be used for the implementation of other systems such as mechatronics, aerospace, and control systems that can be expressed in terms of state-space equations.

III. State-Space Representation of Machine Models

The governing equations describing magnetically coupled stator and rotor circuits in the electrical machines, whose windings are identical and symmetrically placed and whose parameters and specifications are constant, can be written as follows:

$$\begin{bmatrix} \mathbf{V}_{abc_s} \\ \mathbf{V}_{abc_r} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_s & \mathbf{0} \\ \mathbf{0} & \mathbf{r}_r \end{bmatrix} \begin{bmatrix} \mathbf{I}_{abc_s} \\ \mathbf{I}_{abc_r} \end{bmatrix} + p \begin{bmatrix} \boldsymbol{\lambda}_{abc_s} \\ \boldsymbol{\lambda}_{abc_r} \end{bmatrix}$$
$$\begin{bmatrix} \boldsymbol{\lambda}_{abc_s} \\ \boldsymbol{\lambda}_{abc_r} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{abc_{ss}} & \mathbf{L}_{abc_{sr}} \\ \mathbf{L}_{abc_{rs}} & \mathbf{L}_{abc_{rr}} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{abc_s} \\ \mathbf{I}_{abc_r} \end{bmatrix}. \quad (1)$$

The equations representing the dynamic behavior of machines consist of an inductance matrix being a function of rotor position. Thus, a change of frame is used to reduce the complexity of this matrix by referring the machine variables to a reference frame that rotates at an arbitrary velocity given by [23]

$$\mathbf{f}_{qd0} = \mathbf{k}(\theta)\, \mathbf{f}_{abc}$$
$$\mathbf{k}(\theta) = \frac{2}{3} \begin{bmatrix} \cos\theta & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta + \frac{2\pi}{3}\right) \\ \sin\theta & \sin\left(\theta - \frac{2\pi}{3}\right) & \sin\left(\theta + \frac{2\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad (2)$$

where $\mathbf{f}$ represent voltage, current, or flux linkage vector of the stator and rotor circuits.

Using the state-space approach in the orthogonal *dq*-axis model, the simulation of the electrical side of various machines can be expressed as

$$\begin{cases} \frac{d}{dt}\mathbf{x}(t) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{cases} \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^h$ is the state vector, $\mathbf{u} \in \mathbb{R}^p$ is the input vector, and $\mathbf{y} \in \mathbb{R}^q$ is the output vector. Flux linkages $\boldsymbol{\lambda}_{qdo}$, supply voltages $\mathbf{V}_{qdo}$, and output currents $\mathbf{I}_{qdo}$ are selected as the state, input, and output variables, respectively. The matrices and vectors in (3) are defined in Appendix A for different types of machines.

The second set of equations is for the mechanical side of machines for which electromagnetic torque T_e and rotor speed ω_r are the input and state vectors, which are represented by

$$T_e = \left(\frac{3}{2}\right)\left(\frac{P}{2}\right)(\lambda_{ds} i_{qs} - \lambda_{qs} i_{ds}) \tag{4}$$

$$\dot{\omega}_r = \frac{P}{2J} T_{\text{effective}} = \frac{P}{2J}(T_e - T_{\text{mech}} - T_{\text{damp}}). \tag{5}$$

For the digital hardware realization, the differential equations of the machine model should be discretized. The implicit techniques for discretization are more expensive due to a root finding procedure at any given time step. The important observation regarding explicit methods is that the unknown quantities at each time step are given in terms of history parameters. Thus, the discretized equations with an explicit technique can be computed within the shorter elapsed time compared with the implicit one with the same order, resulting in a reduced local truncation error at every time step, a reduced global error, and higher simulation accuracy. However, the drawback arises from the limitation on the time-step size in the explicit method to ensure numerical stability. Since the FPGA can provide nanosecond computation clock cycles, a very small simulation time step, which is much smaller than electrical and evidently mechanical time constants of machines, can be achieved to capture all transients of machine behavior. Thus, the numerical stability is of no concern for the FPGA-based real-time emulation of machines.

The explicit Adams–Bashforth (A-B) method can be employed as a good choice to discretize the machine's equations based on $s = (z - 1/T_s)$ for the first-order A-B transformation or $s = (2/T_s)(z^2 - 1/3z - 1)$ for the second-order A-B transformation [24]. In this paper, the first-order A-B method (forward Euler) is chosen for the discretization of state-space equations, and several implementations in the literature have also used the forward Euler method [25], [26] as follows:

$$\begin{cases} \mathbf{x}(t) = \mathbf{x}(t - T_s) + T_s \times [\mathbf{A}(t - T_s)\mathbf{x}(t - T_s) \\ \qquad\qquad + \mathbf{B}(t - T_s)\mathbf{u}(t - T_s)] \\ \mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t). \end{cases} \tag{6}$$

IV. System Configuration on FPGA

A. Number Representation

Choosing either a fixed- or floating-point number representation is the first step for any hardware design. To provide a comparison for hardware resource utilization and achieve accuracy, a 32-bit single precision floating-point format (*IEEE* Standard 754) by the TPL method (VHDL) and a 32-bit fixed-point format. (The binary point is located where the integer part of the maximum or minimum value of flowing data is effectively fitted in the underlying integer part of the fixed-point number for the highest accuracy) by the schematic method are used for real-time emulation in this paper.

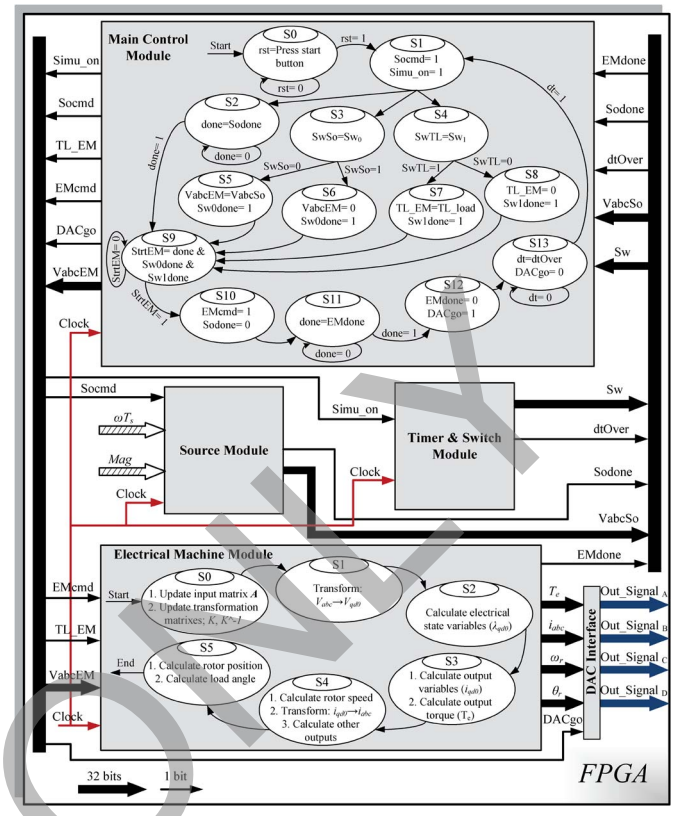


Fig. 1. FSM for paralleled and deeply pipelined real-time algorithm for FPGA implementation of electrical machines.

B. Realization of Machine Models

The FPGA design procedure basically involves the design entry step using the TPL or schematic method to configure the system model and the implementation step to generate the downloadable bitstream.

In this paper, the realization approach can be applied to any type of FPGA devices such as Altera, Xilinx, Lattice, etc. In the developed machine blocks for real-time emulation, the TPL (HDL coding) and schematic methods for different hardware platforms remain the same, and the only difference is between the IP cores of one FPGA type to another designed just for the implementation of basic arithmetic operations.

1) Floating-Point Implementation by VHDL: The hardware modules that assist the real-time emulation to be executed include: `Main Control Module`, `Source Module`, `Timer & Switch Module`, and `Electrical Machine Module`.

Fig. 1 shows the overall procedure in a simulation time step in the proposed hardware. First, `Source Module` generates input voltages for the three-phase machine terminals, whereas `Switch Module` checks the switch states to

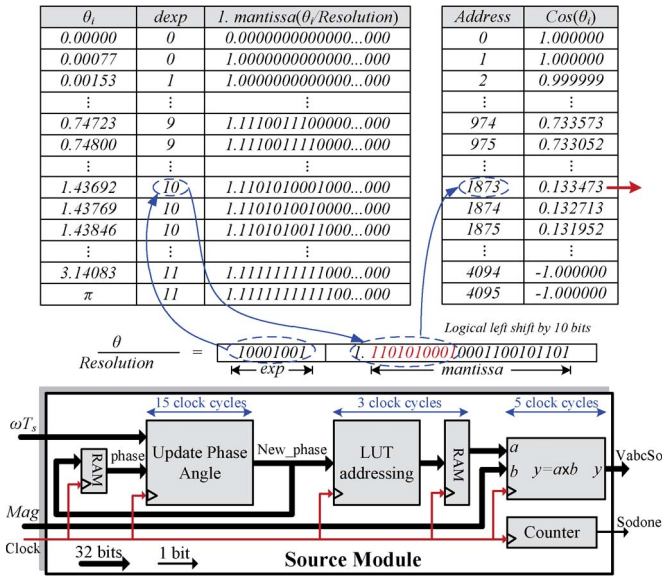


Fig. 2. Pipelined configuration for calculating `Source Module`.

apply or remove faulty conditions and load torque. Then, `Electrical Machine Module` starts to solve machine's equations. It is obvious that the parallel processing exists in each stage, while preserving the necessary sequential stages in the overall simulation algorithm.

The `Main Control Module` coordinates the operation of the whole emulator to carry out the algorithm. It sends out control signals (`Simu_on`, `Socmd`, `EMcmd`) to each module to perform the required functions. Meanwhile, it receives the acknowledged signals (`Sodone`, `EMdone`, `dtOver`, `Sw`) to judge if the functions are done or the switches are on or off.

Using the `rst` function on the FPGA board, the digital hardware emulator starts executing real-time simulation. At first, `Main Control` sends out a command signal `Socmd` to the `Source Module` for generating supply voltages while it checks the switch status signals (`Sw0`, `Sw1`, `Sw2`, ...) to apply or remove voltage source to electrical machine terminals or mechanical torque to the machine shaft. Once acknowledge, signals are received from `Source` and `SwitchModules`, the main control sends out a command signal (`EMcmd`) to the `Electrical Machine Module` to compute the state variables of the machine. Then, it waits to be given acknowledge signal `EMdone` from the machine module and transmits real-time signals to the output ports of the emulator. Finally, `Main Control` checks whether a real-time procedure is performed within the simulation time step `dtOver` and decides to go to the next simulation time step or send the error signal to output ports and terminate the real-time emulation procedure.

The voltage source in the `Source Module`, as shown in Fig. 2, are represented using sinusoidal function cos function. The lookup table (LUT) is the most commonly used method to evaluate this nonlinear function. Since cos is a periodic function, only a half-cycle of the cos function values need to be stored in the LUT in order to save the memory space of the LUT. The accuracy of the cos value is determined by the length of the LUT. In this design, 4096 (2^{12}) cos values for half-cycle are stored in the LUT; thus, the resolution of the LUT is

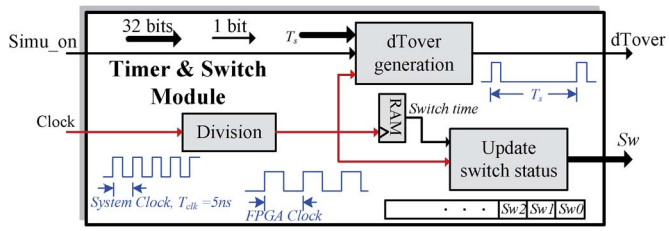
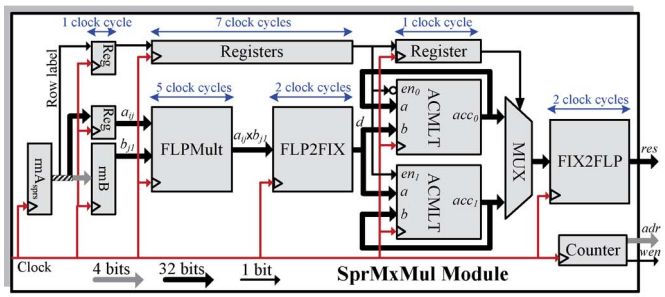


Fig. 3. Functions realized in the `Timer` & `Switch Module`.

$d\theta = \pi/4096$. The calculation of the source values begins with the updating of the `phase` angle. The `new_phase` is obtained by adding the previous `phase` by ωT_s. Since the LUT has only a half-cycle of cos function values, the calculated `phase` needs to be checked with π; if greater than π, it is subtracted by π, and the sign of the result is inverted. Then, the `new_phase` is converted to the address of the LUT. Finally, the retrieved cos value is multiplied by the magnitude `Mag`. The exponent and mantissa of the input floating-point number are used directly to access the LUT when the step length is always a power of two. Assume the floating-point input is θ, the LUT `addressing` unit in Fig. 2 outputs the addresses of the point θ_i, making $\theta - \theta_i <$ resolution, where resolution is the interval of the LUT. This is done by left shifting the leading "1" and mantissa of θ/resolution by *dexp* bits, where the *dexp* is the exponent of θ/resolution (without bias). An example is shown in Fig. 2. In this example, input θ is 1.437, whose *dexp* is 10 (without bias). Left shifting the leading "1" and mantissa by 10 bits gives 1873, which is the address of $\theta_i = 1.43692$ and $\cos(\theta_i) = 0.133473$ [11].

A hardware module of `Timer` & `Switch` is designed to simulate switches. Fig. 3 shows the details of this module and its input/output signals. Since the switches are time controlled, the core of this module is a real-time clock generator. The best achievable clock frequency is generated by the input system clock frequency (200 MHz). This clock signal is counted and compared with the switch operation times saved in a RAM device. Once the switch times are reached, the corresponding switch state bit in `Sw` register (`Sw0`, `Sw1`, `Sw2`, ...) is inverted using "0"/"1" for switch open/closed. Another important function of the `Switch Module` is to generate the `dtOver` signal, which indicates the end of the real simulation time step T_s. When a simulation step is finished, the T_s-over signal is checked. If it is not "1," the simulation step is finished within T_s; thus, the real-time simulation is achieved. Otherwise, the simulation step takes a longer time than T_s, and the real-time constraint is not met [27].

The finite-state machine (FSM) diagram for the hardware realization of machine models is depicted in Fig. 1. The procedure starts with the update of input matrix $\mathbf{A}$ using the calculated speed from the previous time step in parallel with the computation of transformation matrices $\mathbf{K}$ and $\mathbf{K}^{-1}$ using the sinusoidal function LUT based on information about rotor and reference frame positions from the passed time step in state S_0. Then, in state S_1, $\mathbf{V}_{abc}$ is transformed to an arbitrary reference frame qdo to provide an input vector for state-space equations. Once $\mathbf{V}_{qdo}$ is available, the electrical state variables $\boldsymbol{\lambda}_{qdo}$ are calculated in state S_2. The output state variables $\mathbf{I}_{qdo}$ and output torque T_e are computed concurrently in state S_3. $\mathbf{I}_{qdo}$



(a)

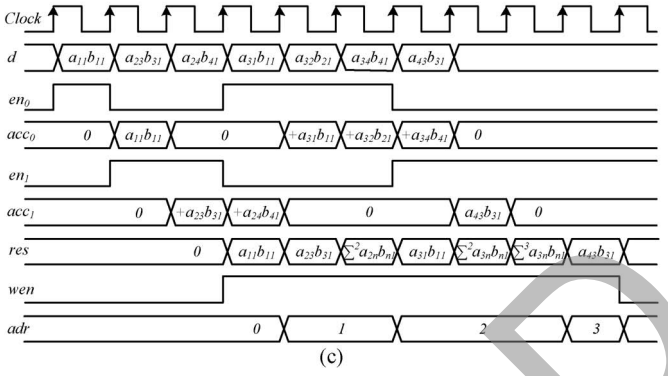
$$A = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & 0 & a_{23} & a_{24} \\ a_{31} & a_{32} & 0 & a_{34} \\ 0 & 0 & a_{43} & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{41} \end{bmatrix}$$

rmB	
RAM address	RAM value (31:0)
0	b_{11}
1	b_{21}
2	b_{31}
3	b_{41}

rmA_{sprs}			
RAM address	RAM value		
	Row label (36)	A Column (35:32)	A elements (31:0)
0	0	1	a_{11}
1	1	3	a_{23}
2	1	4	a_{24}
3	0	1	a_{31}
4	0	2	a_{32}
5	0	4	a_{34}
6	1	3	a_{43}

(b)



(c)

Fig. 4. SprMxMul unit. (a) Hardware design. (b) RAM initialization. (c) Timing diagram.

is transformed back to the abc frame in state S_4; meanwhile, rotor speed ω_r and other required outputs are obtained. Finally, the rotor position and load angle are calculated simultaneously in state S_5. Obviously, since the implementation involves six sequential steps, it is time-consuming. To improve the hardware computational efficiency of the deeply pipelined modules, all possible parallel processing paths have been taken into account, although the overall procedure is sequential.

The hardware realization of electrical machines is designed and performed by arrangement and manipulation of sparse matrix multiplication (SprMxMul) and floating-point multiply-add/subtract (FLPMAS) units. These submodules can be chained to realize many functions and computations, such as simple addition/subtraction, multiplication, conversion between floating- and fixed-point numbers, $a \times b \times c$, $a \times b \pm c \times d \pm e \times f$, $(a \pm b) \times c$, $\mathbf{A}_{n\times1} \pm c \times \mathbf{B}_{n\times1}$, $\mathbf{A}_{n\times n} \times \mathbf{B}_{n\times1} \pm c \times \mathbf{D}_{n\times1}$, etc.

A fast sparse matrix multiplication submodule where a compact sparse matrix storage format (see Fig. 4), which uses only one vector is defined for the realization of $\mathbf{A}_{n\times n} \times \mathbf{B}_{n\times1}$. Each entry in this format has the following: 1) a 32-bit value to store the subsequent nonzero value of matrix $\mathbf{A}_{n\times n}$ in row order; 2) a 4-bit column number to identify the column index of this nonzero value; and 3) a 1-bit row index to label all nonzero values in the same row with "0" or "1". Fig. 4(b) shows an

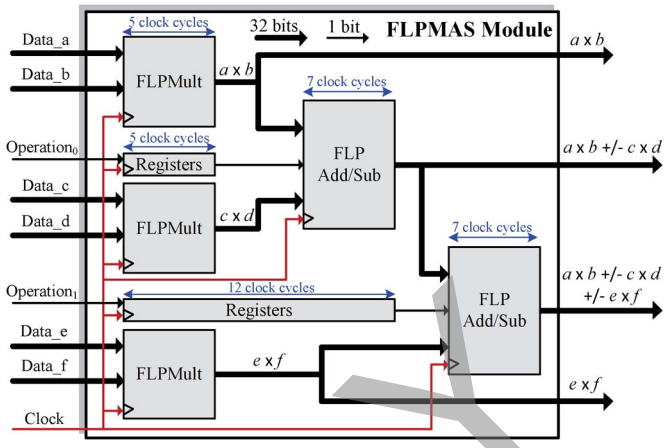


Fig. 5. Pipelined realization of FLPMAS Module.

example sparse matrix and its storage format. In the SprMxMul submodule, the accumulation is done in the fixed-point format. The reason is that floating-point accumulator has much longer latency and requires much more logic resources to implement. The fixed-point accumulator needs only one adder with one clock cycle latency. The used fixed-point number format is 40.100, which has 40 integer bits and 100 fraction bits to guarantee both the range and precision. As shown in Fig. 4(a), the SprMxMul submodule contains one floating-point multiplier, one floating-to-fixed-point converter, two fixed-point adders, and one fixed-point converter. The elements of sparse matrix $\mathbf{A}_{n\times n}$ are retrieved from RAM A_{sprs}, whereas 4-bit column indexes are used to access the $\mathbf{B}_{n\times1}$ matrix stored in RAM B. The registers are inserted for synchronization in the computation. The realized matrix–vector multiplication is fully pipelined and fast because there is no stall between two consecutive matrix row–vector multiplications. This is achieved by the two parallel fixed-point adders (accumulators) acc_0 and acc_1 with opposite enable inputs en_0 and en_1 controlled by the row label information. Fig. 4(c) shows the logic timing diagram for the SprMxMul unit based on the example in this figure. As can be seen, the accumulation of the first matrix row–vector multiplication is processed in the acc_0, whereas the acc_1 is reset to zero, which makes it ready for the accumulation of the next matrix row–vector multiplication [10].

Moreover, the basic floating-point functions, including addition/subtraction and multiplication, are combined to build a basic arithmetic unit. Fig. 5 presents the data flow of various outputs of this submodule. The latency of the longest path is 19 clock cycles. It can be used for floating-point arithmetic operations of both scalar and vector quantities, e.g., the realization of (4)–(6). The FLPMAS unit is pipelined to achieve high data throughput.

Fixed-Point Implementation by the Schematic Method: The schematic method is performed under the MATLAB/Simulink software environment. The procedure is started by the development of a functional model of a system using basic Simulink continuous-time blocks or M-file coding. The behavior of the developed model is verified by the performance of the machine model from the SimPowerSystem toolbox, and the fixed-point formats are then defined for all coefficients, variables, and data paths inside the model. In addition, the number

of required clock cycles and suitable real-time simulation time-step size are determined to compute all governing equations of the system model.

At this stage, the emulation is realized by the development of a digital fixed-point model by using XSG. The final step is linked with FPGA-based implementation. The HDL code of the design architecture is automatically generated. The main control and interface files, as well as pin assignments suitable for the FPGA platforms or HIL test setups, are written and organized. The modifications and adjustments are performed in this step to get the balance in terms of area/time performances. Synthesis, map, place, and route processes and analysis of the static timing performances are done in the Xilinx ISE software. The binary files produced are then transferred to an FPGA via a serial interface (JTAG) for its reconfiguration.

The generated FPGA digital and analog input/output signals, which transmit low-level voltages and currents, can be interfaced with an amplifier that generates and absorbs high-level power where an actual real-time emulated machine works in the HIL configuration.

A generic digital machine model implementation is shown in Fig. 6. It mainly consists of five functional steps. The algorithm starts from the calculation of voltage source in the three-phase domain. The phase angle is incremented over each time step to feed the Xilinx DDS Compiler block that implements a high-performance optimized phase to sinusoidal circuits. The core sources sinusoidal waveforms and consists of a SIN/COS LUT. Moreover, the appropriate voltage magnitude can be achieved by a multiplication unit at the output port of this module [28].

The transformation module is in charge to change *abc* to an arbitrary reference frame. As can be seen in the associated unit for *abc* to stationary transformation, to obtain V_q, two sets of calculation (production of 2/3 and V_a0 and production of 1/3 with the summation of V_b and V_c) must be performed in parallel mode with the same latency before the final addition happens. This is why a register is inserted in the first path to synchronize the data flow with the second parallel path. Consequently, the final addition is only done when the operations at the two input ports are completed at the same time. A similar strategy is applied to compute V_d and also can be used for the transformation of other reference frames.

The state-space variable calculator, depicted in Fig. 6, includes a memory controller that schedules the reads and writes to memory, and a highly parallel processing unit that computes state variables. To show how to implement a state-space system without losing the generality of the problem, the first row of the induction motor state equations is expanded and realized. All potential parallelism and data synchronism by registers are considered in the data flow graph of the following:

$$\lambda_{qs}\left((n)T_s\right) = \lambda_{qs}\left((n-1)T_s\right) + T_s \times \left[A_{11}\cdot\lambda_{qs}\left((n-1)T_s\right) + A_{14} \cdot\lambda_{qr}\left((n-1)T_s\right) + V_{qs}\left((n-1)T_s\right)\right]. \quad (7)$$

In this step, the dual-port RAM devices are employed to store the history terms of the state variables in the $(n-1)$th time step and call them for the integration procedure in the (n)th time step.

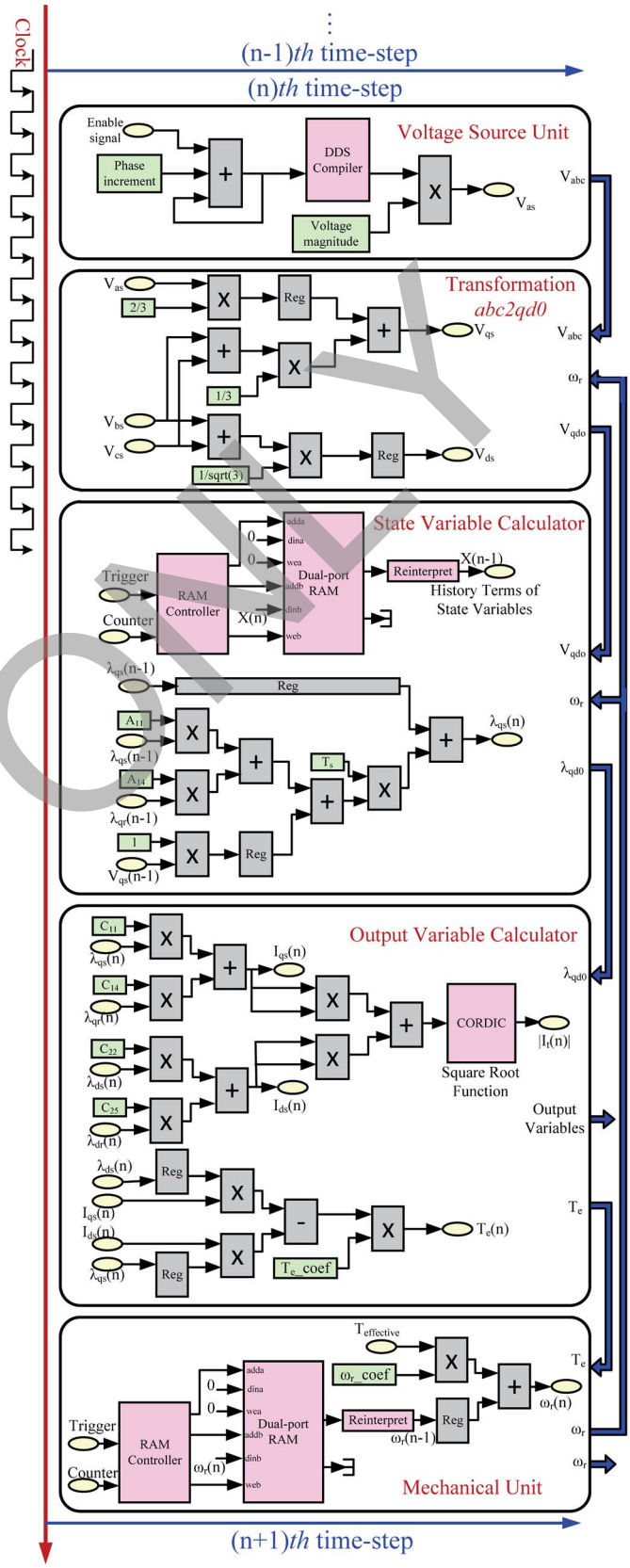


Fig. 6. Design configuration for one emulation step of induction motor by the schematic method.

Once the parallel computation of the state variables are performed, the required output variables are calculated. Most of the output variables such as stator and rotor currents and

TABLE I
LATENCIES AND TIME-STEP SIZES OF DIGITAL HARDWARE REALIZATION

Real-Time Emulator	Fixed-Point		Floating-Point	
	Latency	Time-Step	Latency	Time-Step
Induction Motor	46	$230ns$	234	$1.755\mu s$
Synchronous Generator	55	$275ns$	256	$1.920\mu s$
Line Start-PMSM	52	$260ns$	240	$1.800\mu s$
DC Motor	29	$145ns$	126	$0.945\mu s$

electromagnetic torque can be obtained by using elementary operators, including adders, multipliers, multiplexers, and registers. (The calculation of stator current and electromagnetic torque is provided in the corresponding module in Fig. 6.) However, to obtain some output variables, e.g., the magnitude of a space vector, a square-root function is required, for which the generalized CORDIC function has been used.

Additionally, the structure of blocks arranged to calculate rotor speed is realized based on the A-B integration technique. The RAM controller is responsible to manage the data flow of the history information of rotor speed synchronized by the incoming current speed difference for an accumulation over each time step. The same pattern is applied to calculate rotor position based on currently computed rotor speed information.

V. EVALUATION OF DESIGNED ARCHITECTURES

A. Real-Time Emulation Time-Step Size and Accuracy Assessment

Emulation time step is an important criteria from a practical point of view. In reality, currently available FPGA-based HIL test setups are not able to accommodate real-time models with a time step of more than a few microseconds due to practical limitations to communicate and transfer data to amplifiers and actual devices under real-time tests.

It is worth mentioning that, although in the offline simulation, an increase in the order of the discretization method or in the precision of numbers and operations undoubtedly results in an increase in the accuracy of final results; in the real-time emulation, it does not necessarily lead to an increase in accuracy.

As will be explained later about all case studies, with the same discretization method, single-precision (SP) floating-point implementation of machine models expends more time, as listed in Table I, to finish all calculations within the time step compared with a fixed point. It means that the executed real-time emulation of floating-point realization is carried out with longer time-step size than that of the fixed point. Therefore, although floating-point calculations mathematically offer a higher level of accuracy in comparison with the fixed point, an increase in the associated simulation time-step size of digital hardware implementation increases the truncation error at each time step. Consequently, it may reduce the accuracy of the results in the time-marching or time-stepping digital real-time simulation. That is why, in real-time emulation, an increase in discretization order and precision of calculations may not guarantee more accurate results, whereas in the offline simulation, they can.

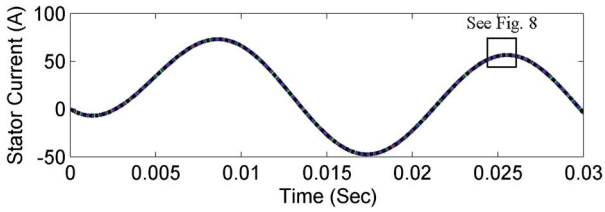


Fig. 7. Stator current of the induction machine using different methods presented in the caption of Fig. 8.

It should be noted that the real-time hardware emulation of machine models in this paper is realized by 5 and 7.5 ns of FPGA clock frequency for fixed- and floating-point implementation, respectively, resulting in smaller real-time emulation time-step sizes compared with [21] and [20] (confirms better real-time implementation).

B. Offline and Experimental Validation

The first validation step is an offline simulation of design architecture according to the best case achievable frequency obtained at the end of the place and route report. This step can be performed by using ISim, ModelSim, or MATLAB software tools. The good functionality and accuracy of the design algorithm written by the TPL or schematic method can be verified in this step.

In the following transient offline study, the first two oscillations of stator current of a Baldor induction machine, whose specifications are listed in Appendix B, by the various approaches are plotted in Fig. 7. The induction machine is started from stall. The reference solution is obtained using the *dq* induction machine model of the SimPowerSystem toolbox and solved with the Runge–Kutta fourth-order (RK4) method and double-precision (DP) operations using a small time step of 100 ns. The simulation results obtained by XSG, MATHLAB M-file, and ISim software are overlaid with the reference solution. As can be observed in Fig. 7, the transient responses produced by all methods coincide and converge to the reference solution. This clearly demonstrates that all approaches predict the machine behavior with the acceptable accuracy. A magnified fragment of Fig. 7 is also shown in Fig. 8 for better comparison. As shown in Fig. 8, the behavior of deeply pipelined digital hardware implementation of the machine model by VHDL coding and SP floating-point operations obtained by ISim (red solid line) with a time step equal to 1.755 μs gives the most deviation from the reference (green solid line), whereas a minor improvement can be achieved by DP calculations with the same algorithm and time step (blue dashed line) obtained by MATLAB M-file coding. The result produced by a digital fixed-point (FiP) hardware implementation (32 bits) of induction motor with a 230-ns time step (magenta dotted line) by the schematic method (XSG) provides more accurate results compared with the previous ones. The most noticeable difference of this waveform from others is its stair step shape. The reason is attributed to the errors that occur when a value lies outside the representable range and when the number of

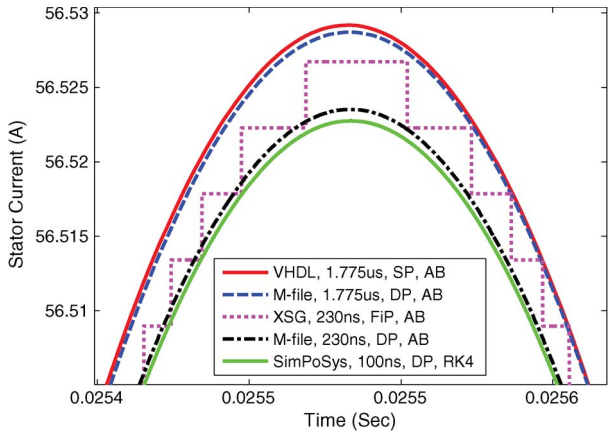


Fig. 8. Zoomed in view of Fig. 7 identified by a legend (realization technique, time-step size, number representation, discretization method). Red solid line: (digital hardware realization by VHDL in ISim, 1.755 μs, SP, A-B), blue dashed line: (offline simulation by M-file, 1.755 μs, DP, A-B), magenta dotted line: (digital hardware realization by XSG, 230 ns, FiP (32 bits), A-B), black dash-dot line: (offline simulation by M-file, 230 ns, DP, A-B), and green solid line: (offline simulation by SimPowerSystem tools, 100 ns, DP, RK4).

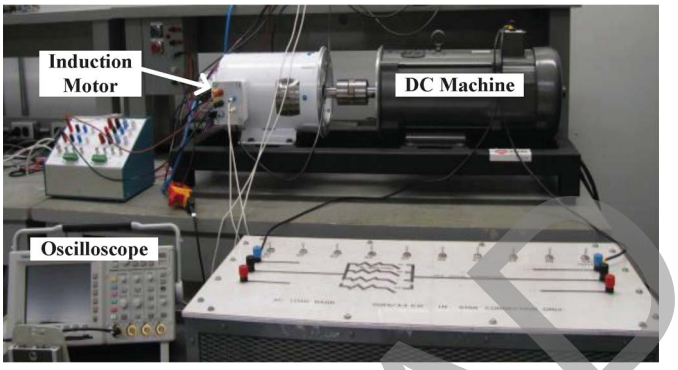


Fig. 9. Experimental setup: induction machine.

fractional bits is insufficient to represent the fractional portion of a value in fixed-point architecture. Moreover, DP calculated stator current by M-file coding (black dash–dot line) with the same condition of fixed-point simulation is also provided for further comparison.

The second validation step is done via a comparison between the FPGA-based real-time emulated machine model and an actual induction machine to make sure that the FPGA-based model can truly duplicate the behavior of the machine in the virtual environment of HIL tests.

A 3-hp Baldor induction machine that is mechanically coupled to a dc generator, shown in Fig. 9, is employed for experimental test. The experiment is carried out to capture the stator current and rotor speed when the test motor is started directly from three-phase power supply.

To plot the experimental and real-time emulated results in the same figure with high precision, the real-time data of output signals generated from the FPGA-based emulator are exported by the ChipScope analyzer and laid over the real values captured by using a current probe and an encoder mounted on an induction machine shaft.

As can be seen from Fig. 10, there is a good agreement between FPGA-based emulated and experimentally measured

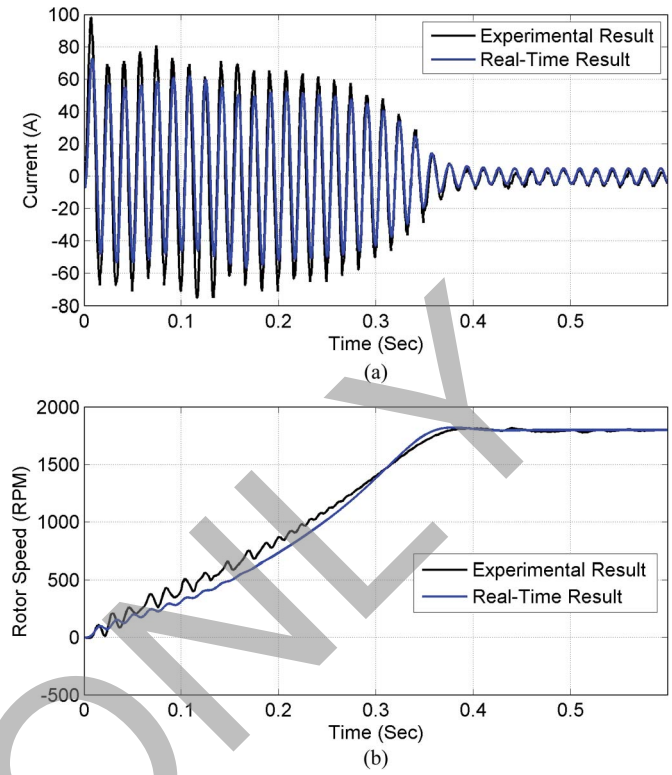


Fig. 10. FPGA-based real-time emulated and experimentally measured results.

induction motor current and rotor speed. Both currents decay to a steady state over almost 0.4 s. Furthermore, it is found that the simulated speed follows the measured one closely. The difference between the current amplitude and speed fluctuation, particularly in the lower speed region or the transient period, is mainly due to the backlash between the induction machine shaft and the dc machine, which magnifies the effect of torque pulsations on the experimentally measured current and speed. Another reason for the discrepancies is due to the lumped *qd* method not being able to model all distributed and spatial effects inside the actual machine. Still, the *qd* model remains the commonly used approach for modeling of electrical machines in the industry and its accuracy has been verified by MATLAB/Simulink in Figs. 7 and 8.

C. Hardware Resource Utilization

The electrical machine hardware designs were targeted to Xilinx Virtex-7 XC7VX485T FPGA. This FPGA is mainly composed by configurable logic blocks that contain a pair of logic slices that are configured by six-input LUTs and storage elements, configurable input/output blocks, and programmable interconnections. It has the following features: 1955k logic cells, 68-Mb block RAM, 2800 DSP48E1, and 1200 I/O pins.

Table II shows the FPGA hardware resource utilization of various types of machines. It can be observed that, although fixed-point operators are thrifty, massively parallel fixed-point implementation of machine models consumes more hardware resources in terms of registers and LUTs compared with

TABLE II
FPGA RESOURCE UTILIZATION OF REAL-TIME MACHINE EMULATOR

Real-Time Emulator	Fixed-Point by Schematic method			Floating-Point by TPL		
	Number of Slice Registers (607,200 available)	Number of Slice LUTs (303,600 available)	Number of DSPs (2,800 available)	Number of Slice Registers (607,200 available)	Number of Slice LUTs (303,600 available)	Number of DSPs (2,800 available)
Induction Motor	25,127	36,567	26	6,689	29,115	38
Synchronous Generator	42,287	61,024	43	7,670	30,493	67
Line Start-PMSM	36,915	53,298	28	7,253	29,540	59
DC Motor	9,886	14,551	9	13,102	6,585	27

deeply pipelined floating-point realization containing extravagant floating-point calculations.

VI. REAL-TIME EMULATION CASE STUDIES

This section presents the real-time performances from the developed hardware machine models. The results were captured on a 500-MHz four-channel oscilloscope that was connected to the data acquisition card on the Xilinx Virtex-7 VC707 FPGA development board. The results show the details of the electrical machine transients under normal and disturbed conditions. To demonstrate the usefulness of the proposed hardware designs for real-time emulated electrical machines in the HIL environment, the models of an induction motor, a synchronous generator with field windings, an LSPMSM, and a shunt-connected dc motor are tested and evaluated. The parameters and specifications of the machines are listed in Appendix B. Note that all the real-time results have been validated by offline simulation of machine models from the SimPowerSystems toolbox in MATLAB/Simulink; however, for the sake of brevity, only the offline results of the induction machine were provided in the previous section.

A. Case I: Induction Motor Transients

The dynamic performance of the induction motor is shown in Fig. 10. The oscilloscope traces of the real-time emulator corresponding to induction motor transients during free acceleration from stall are captured in this case. Moreover, the corresponding offline and experimental results are presented and discussed in Section V. Good agreement between offline, real-time, and experimental results confirms the effectiveness of proposed approach for real-time emulation of the induction machine.

B. Case II: Synchronous Generator Transients

This case focuses on the real-time emulation of dynamic performance of the synchronous generator during a three-phase fault at the machine terminals. The stability of synchronous machines in a power system following a fault is of importance to determine line loading limits. The real-time oscilloscope traces shown in Figs. 11 and 12(a) illustrate the dynamic behavior of the synchronous generator during and following a three-phase fault. The machine is initially connected to an infinite bus delivering rated apparent power at a nominal power factor. The input torque and field voltage are held constant. With the machine operating in a steady state, a three-phase fault occurs at the machine terminals at $t = 0.25$ s. During the fault, the terminal voltage is zero, and the machine is unable to transmit power to system. Hence, all of the input torque, with the exception of the ohmic losses, accelerates the rotor. The fault is cleared at $t = 0.5$ s, and the machine returns to its original operating condition after experiencing a transient condition. Moreover, Fig. 12(b) shows the impact of this disturbance on torque-rotor angle characteristics by a real-time oscilloscope Lissajous curve.

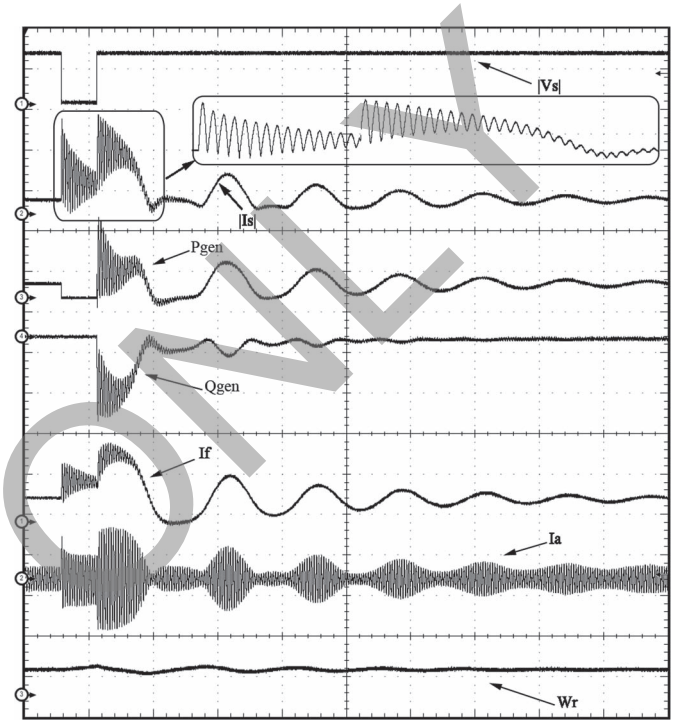


Fig. 11. Real-time traces of synchronous generator dynamic behavior under normal and faulted conditions. [Time: 450 ms/div, v_s (stator voltage phasor): 11.3 kV/div, i_s (stator current phasor): 104.4 kA/div, P_{gen}: 2660 MW/div, Q_{gen}: 2660 MVar/div, I_f: 98.5 kA/div, I_a: 166.5 kA/div, ω_r: 590 rps/div].

C. Case III: LSPMSM Transients

The LSPMSM is a very high efficient alternative to replace induction motor in the constant speed operations with load variations. The dynamic performance of real-time emulated LSPMSM is depicted in Fig. 13 by oscilloscope traces for applied three-phase supply voltages. The motor accelerates from stall. Once the steady-state operation is established, the load torque is suddenly stepped to 80% of nominal torque at $t = 1$ s. Motor increases load angle δ to maintain a steady-state operation, and then a three-phase fault occurs at the motor terminal after $t = 1.5$ s, whereupon the motor loses its synchronism and reestablishes the steady-state condition when

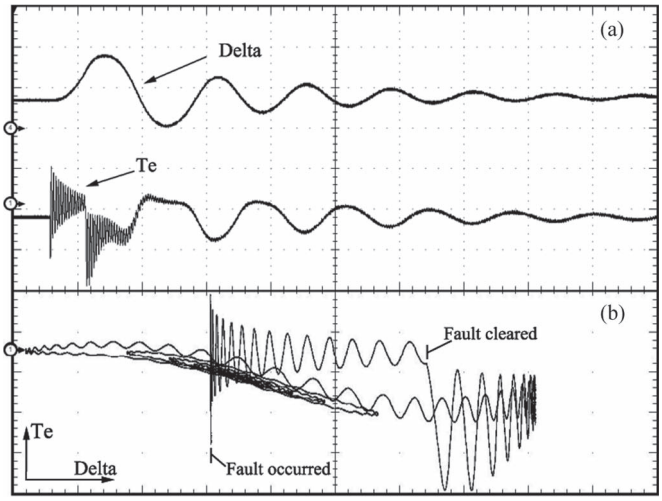


Fig. 12. (a) Real-time traces of torque and rotor angle of the synchronous generator versus time. [Time: 450 ms/div, δ: 1.4 rad/div, T_e: 12.8 MN · m/div]. (b) Real-time X-Y mode trace of torque versus rotor angle of synchronous generator. [T_e: 7.9 MN · m/div], δ: 0.33rad/div].

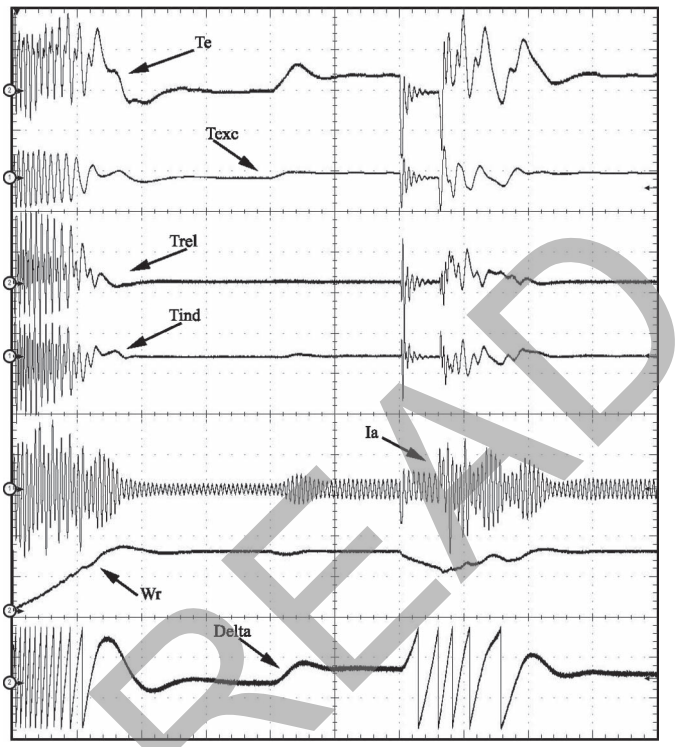


Fig. 13. Real-time traces of LSPMSM performance. [Time: 250 ms/div, T_e: 18.2 N · m/div/div, $T_{\rm exc}$: 39.5 N · m/div/div, , $T_{\rm rel}$: 91.7 N · m/div,], I_a: 46.1 A/div, ω_r: 269.3 rps/div, δ: 2.6 rad/div].

the fault is cleared at $t = 1.65$ s. An induction component $T_{\rm ind}$ of electromagnetic torque T_e significantly contributes to the acceleration of the rotor during starting and faulty conditions, whereas excitation component $T_{\rm exc}$ plays a considerable role in the steady-state operation where reluctance torque $T_{\rm rel}$ has a small effect.

D. Case IV: DC Motor Transients

The results from the real-time emulator of the dc machine are assessed in this case study. A shunt-connected dc motor is selected for evaluation due to its desirable features and

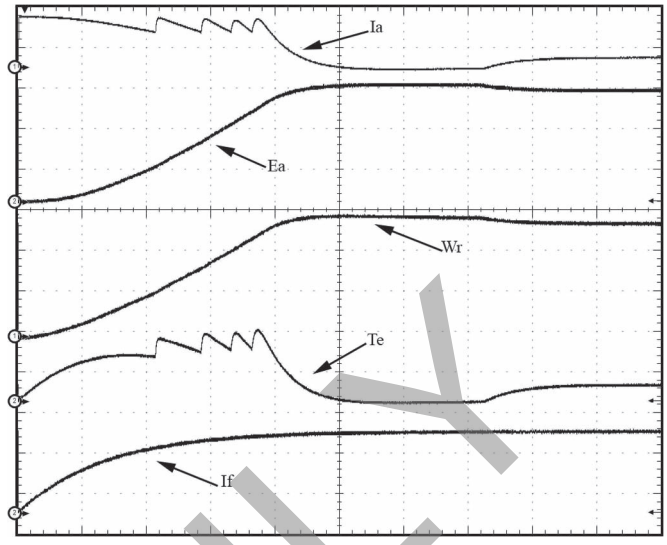


Fig. 14. Real-time trace of dc motor behavior. [Time: 375 ms/div, I_a: 62.5 A/div, E_a: 86.1 V/div, ω_r: 45.5 rps/div, δ: 2.6 rad/div, T_e: 72 N · m/div/div, I_f: 0.5 A/div].

characteristics in the industrial and adjustable speed drive applications. The starting transients of the motor are represented in Fig. 14, when it is started with a resistance starter switched at a fixed armature current level to keep it within the safe limit. The armature and field winding currents I_a and I_f, internal electromagnetic force E_a, rotor speed ω_r, and torque T_e are captured in the results. During no-load starting, the switching of the four resistor segments is being triggered by the crossing of I_a below the threshold value. After a transient condition, steady-state operation is achieved at $t = 2$ s approximately. Then, the rated load torque is suddenly applied at $t = 2.7$ s. There is an increase in the armature current and a decrease in speed for this increase in load torque.

VII. Conclusion

In this paper, a unified framework for FPGA-based digital hardware emulation of electrical machines by different approaches has been presented, and a comprehensive comparison has been provided. Hardware designs are developed for an induction motor, a synchronous generator, an LSPMSM, and a dc machine. The close agreement between the real-time emulated performances, offline simulation, and experimental measurements confirms the effectiveness of the proposed approaches. Such a real-time emulation of the electrical machine can be used in HIL simulations to test new controller and drive systems against a virtual model of the machine. Furthermore, the general framework for digital hardware implementation of the state-space model presented in this paper not only can be applied to machine equations but also can be used for real-time emulation of any system such as power converters, controls, mechatronic systems, etc., that can be described in terms of state-space equations. Moreover, in this paper, the results have demonstrated that, although floating-point arithmetic utilizes more hardware resources compared with fixed-point calculations, deeply pipelined implementation by floating-point number representation can consume less hardware resources

than massively parallel realization by fixed-point operations. In addition, the presented results show that, in spite of the fact that floating-point calculation leads to more accurate results in offline simulation compared with fixed point, in real-time simulation, floating-point computation may not guarantee such a basis. Future work would include modeling of the spatial and nonlinear phenomena inside the machine structure to emulate the machine performances with higher accuracy.

APPENDIX A

Based on the following matrix notation, the state-space definitions for various machines are given as follows:

$$\mathbf{A}=\begin{bmatrix}\mathbf{A}_{11} & \mathbf{A}_{12}\\ \mathbf{A}_{21} & \mathbf{A}_{22}\end{bmatrix},\mathbf{B}=\begin{bmatrix}\mathbf{B}_{11} & \mathbf{B}_{12}\\ \mathbf{B}_{21} & \mathbf{B}_{22}\end{bmatrix},\mathbf{C}=\begin{bmatrix}\mathbf{C}_{11} & \mathbf{C}_{12}\\ \mathbf{C}_{21} & \mathbf{C}_{22}\end{bmatrix}.$$

Induction Machine:

$$\mathbf{x}=[\lambda_{qs}\quad \lambda_{ds}\quad \lambda_{qr}\quad \lambda_{dr}]^T \tag{A1}$$

$$\mathbf{u}=[v_{qs}\quad v_{ds}\quad v_{qr}\quad v_{dr}]^T \tag{A2}$$

$$\mathbf{y}=[i_{qs}\quad i_{ds}\quad i_{qr}\quad i_{dr}]^T \tag{A3}$$

$$\mathbf{A}_{11}=\begin{bmatrix}\frac{r_s}{L_{ls}}\left(\frac{L_{aq}}{L_{ls}}-1\right) & -\omega\\ \omega & \frac{r_s}{L_{ls}}\left(\frac{L_{ad}}{L_{ls}}-1\right)\end{bmatrix}$$

$$\mathbf{A}_{21}=\begin{bmatrix}\frac{r_r}{L_{lr}}\frac{L_{aq}}{L_{ls}} & 0\\ 0 & \frac{r_r}{L_{lr}}\frac{L_{ad}}{L_{ls}}\end{bmatrix},\mathbf{A}_{12}=\begin{bmatrix}\frac{r_s}{L_{ls}}\frac{L_{aq}}{L_{lr}} & 0\\ 0 & \frac{r_s}{L_{ls}}\frac{L_{ad}}{L_{lr}}\end{bmatrix}$$

$$\mathbf{A}_{22}=\begin{bmatrix}\frac{r_r}{L_{lr}}\left(\frac{L_{aq}}{L_{lr}}-1\right) & \omega_r-\omega\\ \omega-\omega_r & \frac{r_r}{L_{lr}}\left(\frac{L_{ad}}{L_{lr}}-1\right)\end{bmatrix} \tag{A4}$$

$$\mathbf{B}=eye(4,4) \tag{A5}$$

$$\mathbf{C}=\frac{1}{D}\begin{bmatrix}L_{rr} & 0 & -L_M & 0\\ 0 & L_{rr} & 0 & -L_M\\ -L_M & 0 & L_{ss} & 0\\ 0 & -L_M & 0 & L_{ss}\end{bmatrix} \tag{A6}$$

$$\mathbf{D}=zeros(4,4). \tag{A7}$$

where

$$L_{ss}=L_{ls}+L_m, L_{rr}=L_{lr}+L_m, D=L_{ss}L_{rr}+L_m^2 \tag{A8}$$

$$L_{ad}^{-1}=L_{aq}^{-1}=1/L_m+1/L_{ls}+1/L_{lr}. \tag{A9}$$

Synchronous Machine:

$$\mathbf{x}=[\lambda_{qs}\quad \lambda_{ds}\quad \lambda_{kq1}\quad \lambda_{kq2}\quad \lambda_{fd}\quad \lambda_{kd}]^T \tag{A10}$$

$$\mathbf{u}=[v_{qs}\quad v_{ds}\quad v_{kq1}\quad v_{kq2}\quad e_{xfd}\quad v_{kd}]^T \tag{A11}$$

$$\mathbf{y}=[i_{qs}\quad i_{ds}\quad i_{kq1}\quad i_{kq2}\quad i_{fd}\quad i_{kd}]^T \tag{A12}$$

$$\mathbf{A}_{11}=\begin{bmatrix}\frac{r_s}{L_{ls}}\left(\frac{L_{aq}}{L_{ls}}-1\right) & -\omega_r & \frac{r_s}{L_{ls}}\frac{L_{aq}}{L_{lkq1}}\\ \omega_r & \frac{r_s}{L_{ls}}\left(\frac{L_{ad}}{L_{ls}}-1\right) & 0\\ \frac{r_{kq1}}{L_{lkq1}}\frac{L_{aq}}{L_{ls}} & 0 & \frac{r_{kq1}}{L_{lkq1}}\left(\frac{L_{aq}}{L_{lkq1}}-1\right)\end{bmatrix}$$

$$\mathbf{A}_{21}=\begin{bmatrix}\frac{r_{kq2}}{L_{lkq2}}\frac{L_{aq}}{L_{ls}} & 0 & \frac{r_{kq2}}{L_{lkq2}}\frac{L_{aq}}{L_{lkq1}}\\ 0 & \frac{r_{fd}}{L_{lfd}}\frac{L_{ad}}{L_{ls}} & 0\\ 0 & \frac{r_{kd}}{L_{lkd}}\frac{L_{ad}}{L_{ls}} & 0\end{bmatrix}$$

$$\mathbf{A}_{12}=\begin{bmatrix}\frac{r_s}{L_{ls}}\frac{L_{aq}}{L_{lkq2}} & 0 & 0\\ 0 & \frac{r_s}{L_{ls}}\frac{L_{ad}}{L_{lfd}} & \frac{r_s}{L_{ls}}\frac{L_{ad}}{L_{lkd}}\\ \frac{r_{kq1}}{L_{lkq1}}\frac{L_{aq}}{L_{lkq2}} & 0 & 0\end{bmatrix}$$

$$\mathbf{A}_{22}=\begin{bmatrix}\frac{r_{kq2}}{L_{lkq2}}\left(\frac{L_{aq}}{L_{lkq2}}-1\right) & 0 & 0\\ 0 & \frac{r_{fd}}{L_{lfd}}\left(\frac{L_{ad}}{L_{lfd}}-1\right) & \frac{r_{fd}}{L_{lfd}}\frac{L_{ad}}{L_{lkd}}\\ 0 & \frac{r_{kd}}{L_{lkd}}\frac{L_{ad}}{L_{lfd}} & \frac{r_{kd}}{L_{lkd}}\left(\frac{L_{ad}}{L_{lkd}}-1\right)\end{bmatrix} \tag{A13}$$

$$\mathbf{B}_{11}=eye(3,3),\mathbf{B}_{12}=\mathbf{B}_{21}=zeros(3,3),\mathbf{B}_{22}=\begin{bmatrix}1 & 0 & 0\\ 0 & \frac{r_{fd}}{L_{md}} & 0\\ 0 & 0 & 1\end{bmatrix} \tag{A14}$$

$$\mathbf{C}_{11}=\begin{bmatrix}\frac{(L_{kq1}L_{kq2}-L_{mq}^2)}{Dq} & 0 & \frac{(-L_{mq}L_{kq2}+L_{mq}^2)}{Dq}\\ 0 & \frac{(L_{fd}L_{kd}-L_{md}^2)}{Dd} & 0\\ \frac{(L_{mq}L_{kq2}-L_{mq}^2)}{Dq} & 0 & \frac{(-L_{mq}L_{kq2}+L_{mq}^2)}{Dq}\end{bmatrix}$$

$$\mathbf{C}_{21}=\begin{bmatrix}\frac{(L_{mq}L_{kq1}-L_{mq}^2)}{Dq} & 0 & \frac{(L_qL_{mq}+L_{mq}^2)}{Dq}\\ 0 & \frac{(L_{md}L_{kd}-L_{md}^2)}{Dd} & 0\\ 0 & \frac{(L_{md}L_{fd}-L_{md}^2)}{Dd} & 0\end{bmatrix}$$

$$\mathbf{C}_{12}=\begin{bmatrix}\frac{(-L_{mq}L_{kq1}+L_{mq}^2)}{Dq} & 0 & 0\\ 0 & \frac{(-L_{md}L_{kd}+L_{md}^2)}{Dd} & \frac{(-L_{md}L_{fd}+L_{md}^2)}{Dd}\\ \frac{(L_qL_{mq}+L_{mq}^2)}{Dq} & 0 & 0\end{bmatrix}$$

$$\mathbf{C}_{22}=\begin{bmatrix}\frac{(-L_qL_{kq1}+L_{mq}^2)}{Dq} & 0 & 0\\ 0 & \frac{(-L_dL_{kd}+L_{md}^2)}{Dd} & \frac{(L_dL_{md}-L_{md}^2)}{Dd}\\ 0 & \frac{(L_dL_{md}-L_{md}^2)}{Dd} & \frac{(-L_dL_{fd}-L_{md}^2)}{Dd}\end{bmatrix} \tag{A15}$$

$$\mathbf{D}=zeros(6,6) \tag{A16}$$

where

$$L_q=L_{ls}+L_{mq}, L_d=L_{ls}+L_{md}, L_{kq1}=L_{lkq1}+L_{mq} \tag{A17}$$

$$L_{kq2}=L_{lkq2}+L_{mq}, L_{fd}=L_{lfd}+L_{md}, L_{kd}=L_{lkd}+L_{md} \tag{A18}$$

$$D_q=L_{mq}^2(L_q-2L_{mq}+L_{kq1}+L_{kq2})-L_qL_{kq1}L_{kq2} \tag{A19}$$

$$D_d=L_{md}^2(L_d-2L_{md}+L_{fd}+L_{kd})-L_dL_{fd}L_{kd} \tag{A20}$$

$$L_{aq}^{-1}=1/L_{mq}+1/L_{ls}+1/L_{lkq1}+1/L_{lkq2} \tag{A21}$$

$$L_{ad}^{-1}=1/L_{md}+1/L_{ls}+1/L_{lfd}+1/L_{lkd}. \tag{A22}$$

LSPMSM:

$$\mathbf{x} = [\lambda_q \quad \lambda_d \quad \lambda_{kq} \quad \lambda_{kd}]^T \tag{A23}$$

$$\mathbf{u} = \left[v_q \quad v_d + r_s \frac{L_{md}}{L_{ls}} i_m \quad v_{kq} \quad v_{kd} + r_{kd} \frac{L_{md}}{L_{lkd}} i_m\right]^T \tag{A24}$$

$$\mathbf{y} = [i_q \quad i_d \quad i_{kq} \quad i_{kd}]^T \tag{A25}$$

$$\mathbf{A}_{11} = \begin{bmatrix} \frac{r_s}{L_{ls}}\left(\frac{L_{aq}}{L_{ls}} - 1\right) & -\omega_r \\ \omega_r & \frac{r_s}{L_{ls}}\left(\frac{L_{ad}}{L_{ls}} - 1\right) \end{bmatrix}$$

$$\mathbf{A}_{21} = \begin{bmatrix} \frac{r_{kq}}{L_{lkq}}\frac{L_{aq}}{L_{ls}} & 0 \\ 0 & \frac{r_r}{L_{lr}}\frac{L_{ad}}{L_{ls}} \end{bmatrix}$$

$$\mathbf{A}_{12} = \begin{bmatrix} \frac{r_s}{L_{ls}}\frac{L_{aq}}{L_{lkq}} & 0 \\ 0 & \frac{r_s}{L_{ls}}\frac{L_{ad}}{L_{lkd}} \end{bmatrix}$$

$$\mathbf{A}_{22} = \begin{bmatrix} \frac{r_{kq}}{L_{lkq}}\left(\frac{L_{aq}}{L_{lkq}} - 1\right) & 0 \\ 0 & \frac{r_{kd}}{L_{lkd}}\left(\frac{L_{ad}}{L_{lkd}} - 1\right) \end{bmatrix} \tag{A26}$$

$$\mathbf{B} = eye(4,4) \tag{A27}$$

$$\mathbf{C} = \begin{bmatrix} -L_{kq}/D_q & 0 & L_{mq}/D_q & 0 \\ 0 & -L_{kd}/D_d & 0 & -L_{md}/D_d \\ L_{mq}/D_q & 0 & -L_q/D_q & 0 \\ 0 & L_{md}/D_d & 0 & -L_d/D_d \end{bmatrix} \tag{A28}$$

$$\mathbf{D} = zeros(4,4). \tag{A29}$$

where

$$L_q = L_{ls} + L_{mq}, D_q = L_{mq}^2 - L_{kq}L_q \tag{A30}$$

$$L_d = L_{ls} + L_{md}, D_d = L_{md}^2 - L_{kd}L_d \tag{A31}$$

$$L_{aq}^{-1} = 1/L_{mq} + 1/L_{ls} + 1/L_{lkq}, L_{ad}^{-1} = 1/L_{md} + 1/L_{ls} + 1/L_{lkd}. \tag{A32}$$

DC Machine:

$$\mathbf{x} = [\lambda_f \quad \lambda_a]^T, \mathbf{u} = [v_f \quad v_a]^T, \mathbf{y} = [i_f \quad i_a]^T \tag{A33}$$

$$\mathbf{A} = \begin{bmatrix} -\frac{r_f}{L_{ff}} & 0 \\ -\omega_r \frac{r_{af}}{L_{ff}} & -\frac{r_a}{L_{aa}} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{A34}$$

$$\mathbf{C} = \begin{bmatrix} 1/L_{ff} & 0 \\ 0 & 1/L_{aa} \end{bmatrix}, \mathbf{D} = zeros(2,2). \tag{A35}$$

Appendix B

The parameters of various machines are given in the following.

i: 3 hp, 230 V, $r_s = 0.5\ \Omega$, $r_r = 0.51\ \Omega$, $l_{ls} = 4$ mH, $l_{lr} = 4$ mH, $l_m = 89.4$ mH.

ii: 26 kV, 802.5 MVA, $r_s = 0.0048\ \Omega$, $r_{fd} = 0.58043\ \Omega$, $r_{kd} = 0.0203\ \Omega$, $r_{kq1} = 0.0727\ \Omega$, $L_{ls} = 0.57031$ mH, $L_{md} = 4.2$ mH, $L_{mq} = 3.8$ mH, $L_{fd} = 0.40759$ mH, $L_{kd} = 0.27852$ mH, $L_{kq1} = 0.16605$ mH.

iii: 4 hp, 230 V, $r_s = 0.017\ \Omega$, $r_{kq} = 0.108\ \Omega$, $r_{kd} = 0.054\ \Omega$, $l_{ls} = 0.172$ mH, $l_{lkq} = 0.350$ mH, $l_{lkd} = 0.350$ mH, $l_{mq} = 2.7$ mH, $l_{md} = 1.3$ mH, $i_m = 1.6203$ A.

iv: 240 V, $r_a = 0.6\ \Omega$, $r_f = 240\ \Omega$, $L_{ff} = 120$ mH, $L_{af} = 1.8$ H, $L_{aa} = 0.012$ H.

References

[1] O. Vodyakho, M. Steurer, C. S. Edrington, and F. Fleming, "An induction machine emulator for high-power applications utilizing advanced simulation tools with graphical user interfaces," *IEEE Trans. Energy Convers.*, vol. 27, no. 1, pp. 160–172, Mar. 2012.

[2] M. Steurer, C. S. Edrington, M. Sloderbeck, W. Ren, and J. Langston, "A megawatt-scale power hardware-in-the-loop simulation setup for motor drives," *IEEE Trans. Ind. Electron.*, vol. 57, no. 4, pp. 1254–1260, Apr. 2010.

[3] S. C. Oh, "Evaluation of motor characteristics for hybrid electric vehicles using the hardware-in-the-loop concept," *IEEE Trans. Veh. Technol.*, vol. 54, no. 3, pp. 817–824, May. 2005.

[4] S. Abourida, C. Dufour, J. Be'langer, T. Yamada, and T. Arasawa, "Hardware-in-the-loop simulation of finite-element based motor drives with RT-LAB and JMAG," in *Proc. IEEE Int. Symp. Ind. Electron.*, Jul. 9–13, 2006, vol. 3, pp. 2462–2466.

[5] M. O. Faruque and V. Dinavahi, "Hardware-in-the-loop simulation of power electronic systems using adaptive discretization," *IEEE Trans. Ind. Electron.*, vol. 57, no. 4, pp. 1146–1158, Apr. 2010.

[6] I. Munteanu, A. I. Bratcu, S. Bacha, D. Roye, and J. Guiraud, "Hardware-in-the-loop-based simulator for a class of variable-speed wind energy conversion systems: design and performance assessment," *IEEE Trans. Energy Convers.*, vol. 25, no. 2, pp. 564–576, Jun. 2010.

[7] H. F. Blanchette, T. Ould-Bachir, and -P. David, "A state-space modeling approach for the FPGA-based real-time simulation of high switching frequency power converters," *IEEE Trans. Ind. Electron.*, vol. 59, no. 12, pp. 4555–4567, Dec. 2012.

[8] M. W. Naouar, E. Monmasson, A. A. Naassani, and I. Slama-Belkhodja, "FPGA-based dynamic reconfiguration of sliding mode current controllers for synchronous machines," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1262–1271, Aug. 2013.

[9] H. Berriri, W. Naouar, I. Bahri, I. Slama-Belkhodja, and E. Monmasson, "Field programmable gate array-based fault-tolerant hysteresis current control for AC machine drives," *IET Electric Power Appl.*, vol. 6, no. 3, pp. 181–189, Mar. 2012.

[10] Y. Chen and V. Dinavahi, "An iterative real-time nonlinear electromagnetic transient solver on FPGA," *IEEE Trans. Ind. Electron.*, vol. 58, no. 6, pp. 2547–2555, Jun. 2011.

[11] Y. Chen and V. Dinavahi, "Hardware emulation building blocks for real-time simulation of large-scale power grids," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 373–381, Feb. 2014.

[12] Y. Wang and V. Dinavahi, "Low-latency distance protective relay on FPGA," *IEEE Trans. Smart Grid*, vol. 5, no. 2, pp. 896–905, Mar. 2014.

[13] J. Liu and V. Dinavahi, "A real-time nonlinear hysteretic power transformer transient model on FPGA," *IEEE Trans. Ind. Electron.*, vol. 61, no. 7, pp. 3587–3597, Jul. 2014.

[14] N. K. Quang, N. T. Hieu, and Q. P. Ha, "FPGA-based sensorless PMSM speed control using reduced-order extended kalman filters," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 4574–4582, Dec. 2014.

[15] Y. Chen and V. Dinavahi, "Multi-FPGA digital hardware design for detailed large-scale real-time electromagnetic transient simulation of power systems," *IET Gen. Transmiss. Distrib.*, vol. 7, no. 5, pp. 451–463, May 2013.

[16] P. A. Rajne and V. Ramanarayanan, "Programming an FPGA to emulate the dynamics of DC machines," in *Proc. IEEE IICPE*, Dec. 2006, pp. 120–124.

[17] G. G. Parma and V. Dinavahi, "Real-time digital hardware simulation of power electronics and drives," *IEEE Trans. Power Del.*, vol. 22, no. 2, pp. 235–1246, Apr. 2007.

[18] C. Dufour, J. Be'langer, and V. Lapointe, "FPGA-based ultra-low latency HIL fault testing of a permanent magnet motor drive using RT-LAB-XSG," in *Proc. IEEE POWERCON*, Oct. 2008, pp. 12–15.

[19] C. Hao, S. Song, D. C. Aliprantis, and J. Zambreno, "Dynamic simulation of electric machines on FPGA boards," in *Proc. IEEE IEMDC*, May 2009, pp. 1523–1528.

[20] M. Matar and R. Iravani, "Massively parallel implementation of AC machine models for FPGA-based real-time simulation of electromagnetic transients," *IEEE Trans. Power Del.*, vol. 26, no. 2, pp. 830–840, Apr. 2011.
[21] Y. Chen and V. Dinavahi, "Digital hardware emulation of universal machine and universal line models for real-time electromagnetic transient simulation," *IEEE Trans. Ind. Electron.*, vol. 59, no. 2, pp. 1300–1309, Feb. 2012.
[22] E. Monmasson *et al.*, "FPGAs in industrial control applications," *IEEE Trans. Ind. Informat.*, vol. 7, no. 2, pp. 224–243, May 2011.
[23] P. C. Krause, O. Wasynczuk, and S. Sudhoff, *Analysis of Electric Machinery and Drive Systems*. New York, USA: IEEE Press, 1994.
[24] G. A. Korn and T. M. Korn, *Mathematical Handbook for Scientists and Engineers: Definitions, Theorems, and Formulas for Reference and Review*. New York, USA: McGraw-Hill, 1961.
[25] I. Bahri, L. Idkhajine, E. Monmasson, and M. E. A. Benkhelifa, "Hardware/software codesign guidelines for system on chip FPGA-based sensorless AC drive applications," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2165–2176, Nov. 2013.
[26] L. Idkhajine, E. Monmasson, and A. Maalouf, "Fully FPGA-based sensorless control for synchronous AC drive using an extended kalman filter," *IEEE Trans. Ind. Electron.*, vol. 59, no. 10, pp. 3908–3918, Oct. 2012.
[27] Y. Chen, "Large-scale real-time electromagnetic transient simulation of power systems using hardware emulation on FPGAs," Ph.D. dissertation, Dept. Elect. Comp. Eng., Univ. Alberta, Edmonton, AB, Canada, 2012.
[28] *User Guide of Xilinx System Generator for DSP*, Xilinx Inc., San Jose, CA, USA, Oct. 2012.

Nariman Roshandel Tavana (S'08) was born in Rasht, Iran, in 1983. He received the B.Sc. degree in electrical engineering from the University of Guilan, Guilan, Iran, in 2006 and the M.Sc. degree (with honors) in electrical power engineering from Iran University of Science and Technology (IUST), Tehran, Iran, in 2009. He is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada.

His research interests include field-programmable-gate-array-based real-time emulation, hardware-in-the-loop simulation, design and modeling of electrical machines and drive systems, and finite-element analysis of electromagnetic devices.

Venkata Dinavahi (SM'08) received the Ph.D. degree from the University of Toronto, Toronto, ON, Canada, in 2000.

He is currently a Professor of electrical and computer engineering with the University of Alberta, Edmonton, AB, Canada. His research interests include real-time simulation of electrical machines, power electronics and power systems, large-scale system simulation, and parallel and distributed computing.