Dynamic Simulation of Electric Machines on FPGA Boards

Hao Chen, Song Sun, Dionysios C. Aliprantis, and Joseph Zambreno
Department of Electrical and Computer Engineering Iowa State University, Ames, IA 50011 USA

***Abstract*—This paper presents the implementation of an induction machine dynamic simulation on a field-programmable gate array (FPGA) board. Using FPGAs as computational engines can lead to significant simulation speed gains when compared to a typical PC computer, especially when operations can be efficiently parallelized on the board. The textbook example of a free acceleration followed by a step load change is used to outline the basic steps of designing an explicit Runge–Kutta numerical ordinary differential equation (ODE) solver on the FPGA platform. The FPGA simulation results and speed improvement are validated versus a Matlab/Simulink simulation.**

I. Introduction

A field-programmable gate array (FPGA) is a reconfigurable digital logic platform, which allows for the parallel execution of millions of bit-level operations in a spatially programmed environment. Research has been under way on the modeling and real-time simulation of various electrical power components using FPGAs as computational [1]–[6] and non-computational [7], [8] devices. Herein, the goal is to implement an entire dynamic simulation of an induction machine on a single FPGA board, as fast as possible (i.e., without being constrained by the requirement of real-time simulation). Even though an induction machine has been selected, a similar process can be used to simulate other types of electric machinery. The textbook example of a free acceleration followed by a step load change is used to outline the basic steps of implementing an explicit fourth-order Runge–Kutta (RK4) numerical ordinary differential equation (ODE) solver on the FPGA platform.

The individual mathematical operations required by numerical integration algorithms are generally simple in terms of required logic (additions and multiplications). Hence, hardware implementations can be used to increase efficiency by reducing the overhead introduced by software, thus leading to simulation speed gains of two orders of magnitude when compared to PCs. Moreover, complex systems requiring the simultaneous solution of numerous differential equations for simulation are inherently conducive to a parallel mapping to physical computational resources. Therefore, an FPGA becomes an attractive choice for simulating complex electrical power and energy systems. This paper presents initial work towards an "ultimate" goal of a fully functional FPGA-based simulation platform, as shown in Fig. 1. Herein, an induction machine model is designed using Very High Speed Integrated Circuit Hardware Description Language (VHDL), synthesized and verified using Xilinx Integrated Software Environment (ISE), and implemented on the FPGA development board using Xilinx Embedded Development Kit (EDK).

This project was financially supported by the "CODELESS: COnfigurable DEvices for Large-scale Energy System Simulation" project, funded by the Electrical Power Research Center (EPRC) at Iowa State University.

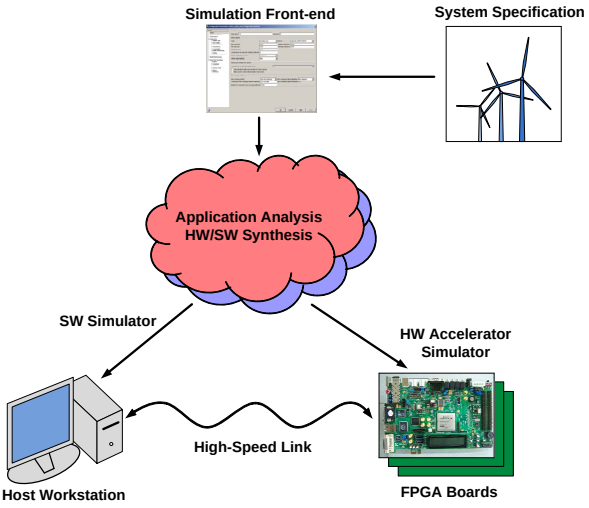

Fig. 1. Conceptual approach for FPGA-based simulation

II. Induction Machine Model

The equations of a fifth-order squirrel-cage induction machine model in the stationary reference frame are given by [9]:

$$v_{qs}^s = R_s i_{qs}^s + p(L_s i_{qs}^s + L_m i_{qr}'^s) \quad (1)$$

$$v_{ds}^s = R_s i_{ds}^s + p(L_s i_{ds}^s + L_m i_{dr}'^s) \quad (2)$$

$$0 = R_r' i_{qr}'^s - \omega_r(L_r' i_{dr}'^s + L_m i_{ds}^s) + p(L_r' i_{qr}'^s + L_m i_{qs}^s) \quad (3)$$

$$0 = R_r' i_{dr}'^s + \omega_r(L_r' i_{qr}'^s + L_m i_{qs}^s) + p(L_r' i_{dr}'^s + L_m i_{ds}^s) \quad (4)$$

$$p\omega_r = \tfrac{P}{2J}(T_e - T_L) \quad (5)$$

$$T_e = 0.75PL_m(i_{qs}^s i_{dr}'^s - i_{ds}^s i_{qr}'^s) \quad (6)$$

where $p = \frac{d}{dt}$ is the differentiation operator; R_s and R_r' are the stator and rotor resistances; L_s and L_r' are the stator and rotor inductances; L_m is the magnetizing inductance; v_{qs}^s and v_{ds}^s are the qd-axes stator voltages; i_{qs}^s and i_{ds}^s are the qd-axes stator currents; $i_{qr}'^s$ and $i_{dr}'^s$ are the qd-axes rotor currents; T_e is the electromagnetic torque; T_L is the load torque; ω_r is the rotor angular electrical speed; P is the number of poles; and J is the total rotor inertia.

These equations are used to simulate an induction machine

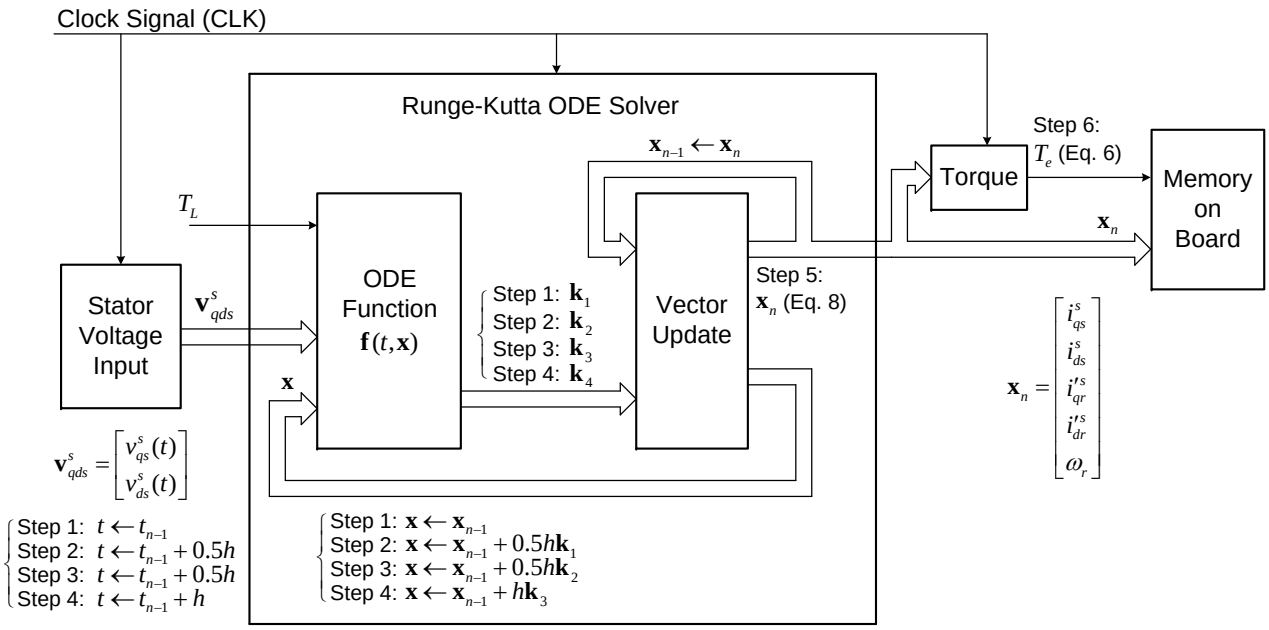

Fig. 2. FPGA implementation of induction machine simulation

free acceleration (where $T_L = 0$) followed by a step load change (where T_L is stepped to its rated value). The machine is excited by a balanced sinusoidal three-phase voltage set, which is expressed in *qd*-axes variables as

$$\begin{aligned} \begin{bmatrix} v^s_{qs} \\ v^s_{ds} \\ v_{0s} \end{bmatrix} &= \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} V_s \cos(\omega_e t) \\ V_s \cos(\omega_e t - \frac{2\pi}{3}) \\ V_s \cos(\omega_e t + \frac{2\pi}{3}) \end{bmatrix} \\ &= \begin{bmatrix} V_s \cos(\omega_e t) \\ -V_s \sin(\omega_e t) \\ 0 \end{bmatrix} . \end{aligned} \tag{7}$$

Note that the zero-axis voltage is zero, and can be neglected in this case.

III. FPGA Implementation

A. Simulation Architecture

The transient response of the electric machine is obtained by the RK4 numerical integration algorithm [10]. This is a fixed-step explicit integration algorithm, which is based on simple numerical calculations (additions and multiplications), and is thus straightforward to implement on the FPGA. The RK4 method for the initial value problem $(p\mathbf{x} = \mathbf{f}(t,\mathbf{x}), \mathbf{x}(t_0) = \mathbf{x}_0)$ is described by:

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \tfrac{h}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right) \tag{8}$$

$$t_n = t_{n-1} + h \tag{9}$$

where $\mathbf{x}_n$ is the RK4 approximation of $\mathbf{x}(t_n)$ (i.e., the exact solution), h is the time step, and

$$\mathbf{k}_1 = \mathbf{f}\left(t_{n-1}, \mathbf{x}_{n-1}\right) \tag{10}$$

$$\mathbf{k}_2 = \mathbf{f}\left(t_{n-1} + 0.5h, \mathbf{x}_{n-1} + 0.5h\mathbf{k}_1\right) \tag{11}$$

$$\mathbf{k}_3 = \mathbf{f}\left(t_{n-1} + 0.5h, \mathbf{x}_{n-1} + 0.5h\mathbf{k}_2\right) \tag{12}$$

$$\mathbf{k}_4 = \mathbf{f}\left(t_{n-1} + h, \mathbf{x}_{n-1} + h\mathbf{k}_3\right) . \tag{13}$$

From (1)–(6), the system of ODEs can be expressed in the form $p\mathbf{x} = \mathbf{f}(t,\mathbf{x})$ by

$$p\mathbf{i} = \mathbf{A}\mathbf{i} + \mathbf{B}\mathbf{v}^s_{qds} \tag{14}$$

$$p\omega_r = \tfrac{0.375P^2L_m}{J}(i^s_{qs}i'^s_{dr} - i^s_{ds}i'^s_{qr}) - \tfrac{P}{2J}T_L \tag{15}$$

where

$$\mathbf{i} = \begin{bmatrix} i^s_{qs} & i^s_{ds} & i'^s_{qr} & i'^s_{dr} \end{bmatrix}^T ,$$

$$\mathbf{v}^s_{qds} = \begin{bmatrix} v^s_{qs} & v^s_{ds} \end{bmatrix}^T ,$$

$$\mathbf{A} = \begin{bmatrix} -\frac{R_s}{\sigma L_s} & -\frac{L_m^2}{\sigma L_s L'_r}\omega_r & \frac{L_m R'_r}{\sigma L_s L'_r} & -\frac{L_m}{\sigma L_s}\omega_r \\ \frac{L_m^2}{\sigma L_s L'_r}\omega_r & -\frac{R_s}{\sigma L_s} & \frac{L_m}{\sigma L_s}\omega_r & \frac{L_m R'_r}{\sigma L_s L'_r} \\ \frac{L_m R_s}{\sigma L_s L'_r} & \frac{L_m}{\sigma L'_r}\omega_r & -\frac{R'_r}{\sigma L'_r} & \frac{1}{\sigma}\omega_r \\ -\frac{L_m}{\sigma L'_r}\omega_r & \frac{L_m R_s}{\sigma L_s L'_r} & -\frac{1}{\sigma}\omega_r & -\frac{R'_r}{\sigma L'_r} \end{bmatrix} ,$$

$$\mathbf{B} = \begin{bmatrix} \frac{1}{\sigma L_s} & 0 \\ 0 & \frac{1}{\sigma L_s} \\ -\frac{L_m}{\sigma L_s L'_r} & 0 \\ 0 & -\frac{L_m}{\sigma L_s L'_r} \end{bmatrix} ,$$

and $\sigma = 1 - L_m^2/L_s L'_r$. The state variables are i^s_{qs}, i^s_{ds}, i'^s_{qr}, i'^s_{dr}, and ω_r. The input variables are v^s_{qs}, v^s_{ds}, and T_L. The output variable is T_e given by (6). Herein, this output is only selected as an example to highlight the computational step of calculating outputs that depend on the model states.

As shown in Fig. 2, four functional modules are used to establish the induction machine model. The 'Stator Voltage Input'

module is responsible for the generation of v_{qs}^s and v_{ds}^s. The 'ODE Function' and 'Vector Update' modules constitute the RK4 solver. The 'Torque' module implements the calculation of the 'output' (6). These modules have been developed using VHDL in ModelSim [11], which is a verification and simulation tool for VHDL designs. All variables and parameters are represented as signed fixed-point numbers with 24 bits representing the integral part, and 32 bits representing the fractional part. This provides a numerical range that can accommodate every variable involved in the simulation, with a resolution of 2^{-32}.

Every RK4 iteration shown in Fig. 3 consists of six steps. The 'ODE Function' module executes the evaluation of (14) and (15). The 'Vector Update' module is responsible for the alteration of $\mathbf{x}$ in $\mathbf{f}(t, \mathbf{x})$ during step 2, 3 and 4, as well as the calculation of (8) in step 5. Since v_{qs}^s and v_{ds}^s in (14) are dependent on the time t, the 'Stator Voltage Input' module should generate the appropriate v_{qs}^s and v_{ds}^s for the 'ODE Function' module. Specifically, $v_{qs}^s(t_{n-1} + 0.5h)$ and $v_{ds}^s(t_{n-1}+0.5h)$ are generated during step 1 and stored for the usage of the 'ODE Function' module in step 2 and step 3, while $v_{qs}^s(t_{n-1} + h)$ and $v_{ds}^s(t_{n-1} + h)$ are generated during step 3 and stored for the usage of the 'ODE Function' module in step 4 and step 1 of the next iteration. Note that the 'Stator Voltage Input' module and the 'ODE Function' module are executed in parallel in step 1. A similar parallel execution is also performed in step 3. On the other hand, the 'ODE Function' module and the 'Vector Update' module have to be executed in serial pattern because the inputs of one strictly depend on the outputs of the other.

To design a sinusoidal function involved in the 'Stator Voltage Input' module, a look-up table approach is followed. The value of $\sin(x)$ is obtained from a 1000-element long look-up table, while $\cos(x)$ is obtained from the same table using an offset of $\pi/2$, which corresponds to a searching index offset of 250 in the look-up table. As shown in Fig. 4, the START signal (active high) starts the sin–cos function block, and the DONE signal (active high) signifies the end of operations. The operations are allocated in six clock cycles, as follows:

1. calculate $x\frac{1}{2\pi}$ (a multiplication operation);
2. obtain the fractional part of $|x\frac{1}{2\pi}|$: x_f;
3. calculate Nx_f, where $N = 1000$;
4. obtain $\sin(|x|)$ from the look-up table, where the searching index is the integer part of Nx_f: x_i;
5. obtain $\cos(x)$ from the look-up table, where the searching index is x_i+250; meanwhile, determine the value of $\sin(x)$ according to $\sin(|x|)$ and the sign of x;
6. write the values of $\sin(x)$ and $\cos(x)$ to the output ports.

The 'ODE Function' module shown in Fig. 5 includes three blocks. The 'Matrix Calculation' block which yields the matrix $\mathbf{A}$ $(=[a_{ij}]_{4\times4})$ in (14) is implemented using one multiplier and one subtracter. The elements (a_{21}, a_{23}, a_{32}, a_{34}) in $\mathbf{A}$, which involve multiplications with ω_r, are obtained from the multiplier. The other four elements (a_{12}, a_{14}, a_{41}, a_{43}) also associated with ω_r are easily obtained from the subtracter since each one is the opposite of a former multiplication result. The remaining elements are stored in advance because they are constants. The evaluation of (14) is executed by the 'Equation Group' block, which is implemented using a 5-stage tree-shaped pipelined multiplier and adder structure, shown in Fig. 6 [12], [13]. The evaluation of (15) is executed by the '5$^{\text{th}}$

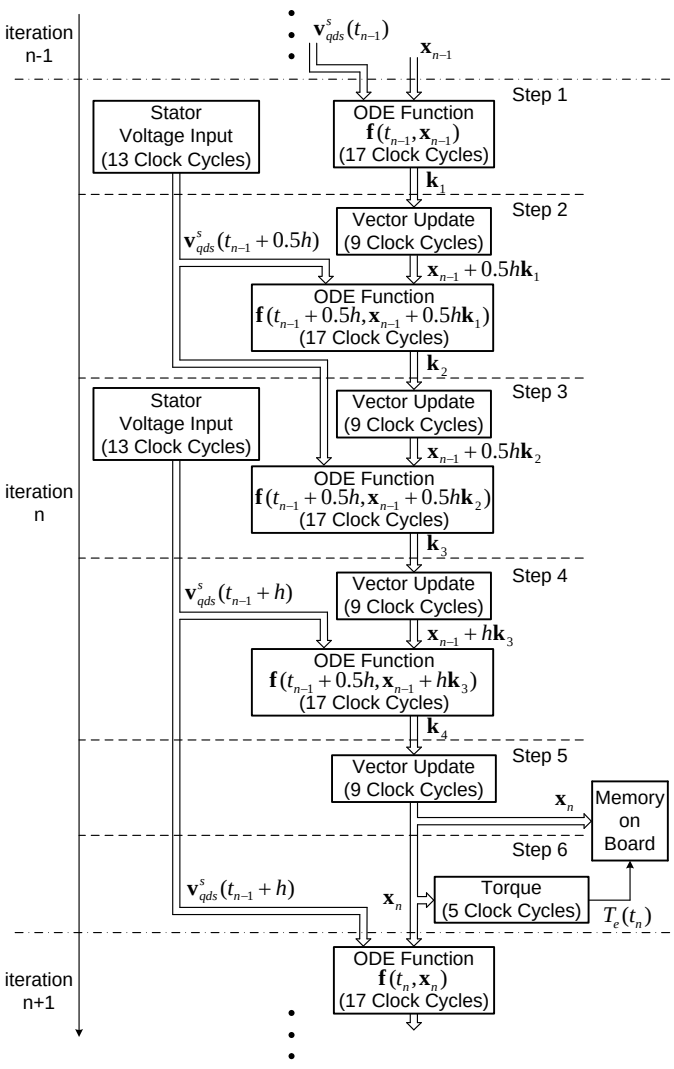

Fig. 3. RK4 iteration process

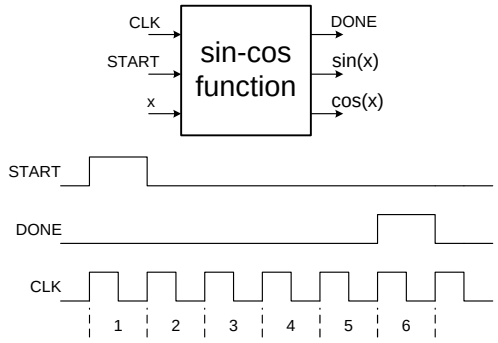

Fig. 4. sin–cos function block

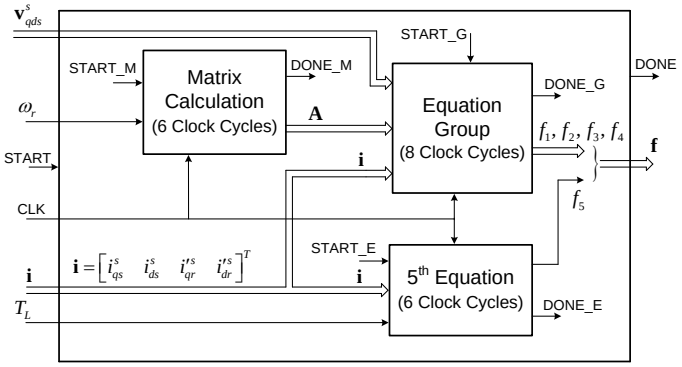

Fig. 5. ODE function module

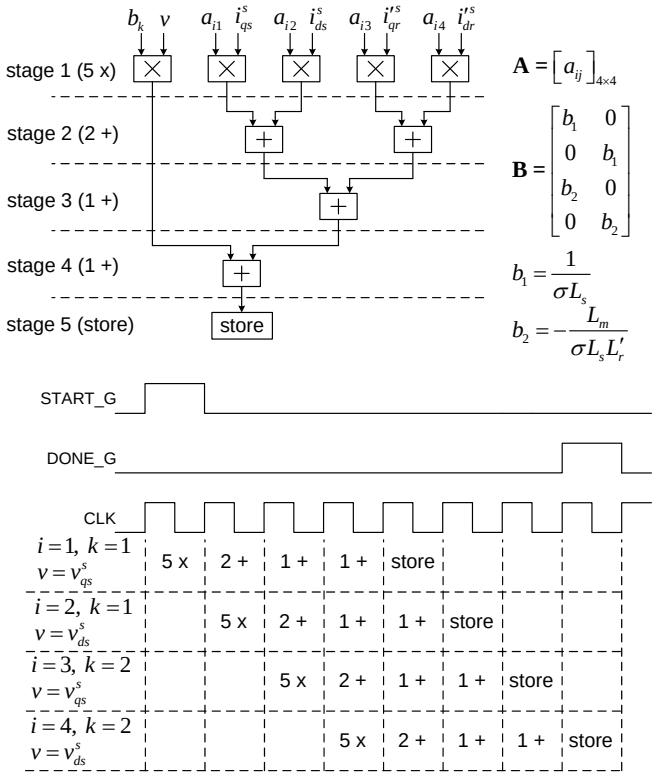

Fig. 6. 5-stage pipeline structure and operation

Equation' block, which is implemented using one multiplier and one subtracter.

A finite state machine (FSM) [14], shown in Fig. 7, is designed to coordinate the behaviors of these three blocks. The START and DONE signals (also shown in Fig. 5) correspond to the main 'ODE Function' module. Each internal block has its own operation-start signal (START_x) and operation-end signal (DONE_x), where 'x' can be either 'M' ('Matrix Calculation' block), 'G' ('Equation Group' block), or 'E' ('5th Equation' block). For example, START_M is activated (START_M = 1) as soon as START is active. After completing the matrix calculation, the 'Equation Group' block and the '5th Equation' block are executed in a parallel pattern (START_G and START_E are activated at the same time) because these two blocks are

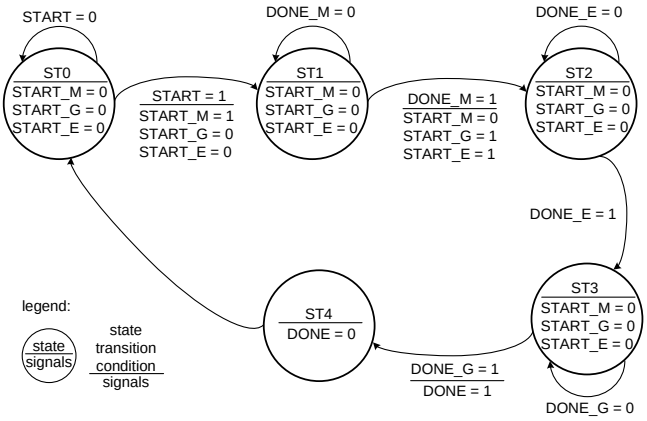

Fig. 7. State diagram of FSM for ODE function module

TABLE I
XILINX VIRTEX–5 XC5VLX110T RESOURCES USAGE SUMMARY

Logic Utilization	Used	Available
Number of Slice Registers	7722	69120
Number of Slice LUTs (Look Up Tables)	27503	69120
Number of LUT-FF (Flip Flop) pairs	28927	69120
Number of DSP48Es	55	64

completely independent. Note that the number of clock cycles required for the 'ODE Function' module is not $14 = 6 + 8$ (as implied by Fig. 5) but $17 = 6 + 8 + 3$ (as shown in Fig. 3), because the FSM introduces 3 extra clock cycles. A similar FSM is also used to coordinate the behaviors of the four modules in Fig. 2, and adds an overhead of 12 clock cycles per iteration, so that the total number of clock cycles per iteration is $121 = 17 + 9 + \cdots + 9 + 5 + 12$ (see Fig. 3).

B. Synthesis and Implementation

After the functionality and results of all modules designed using VHDL were validated in the ModelSim environment, the Xilinx ISE was used to develop, synthesize and verify the substantial top-level wrapper module together with the machine model. The target FPGA device was Xilinx Virtex-5 XC5VLX110T [15]. The post-place and route report presented the FPGA hardware resources usage as shown in Table I, and that the maximum frequency of the clock signal that can be applied is 89.735 MHz. Generally, the consumption of FPGA hardware resources increases with model complexity. Note that the entire design for the machine model must fit within the resource limitation of the target FPGA device. Otherwise, an FPGA device with more hardware resources should be chosen or the machine model should be redesigned in order to meet the requirement of the FPGA device.

The final system was integrated on a XUPV5-LX110T development board [16], which features the XC5VLX110T device.

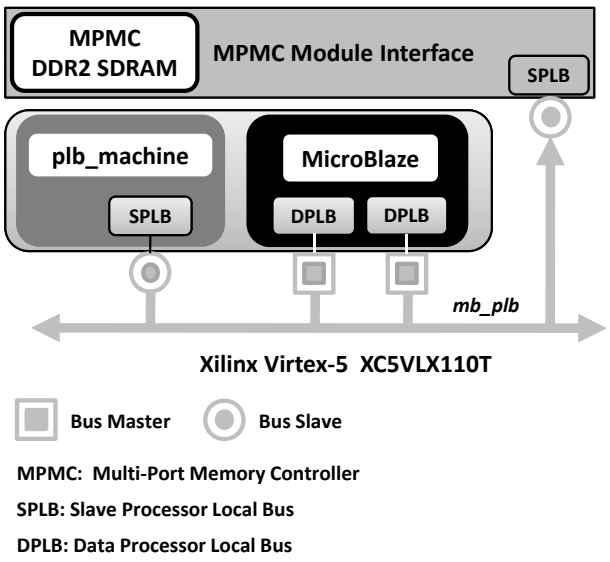

Fig. 8. Implementation architecture

Besides FPGA logic blocks, this device has the embedded MicroBlaze soft processor whose software is written in C. The Xilinx EDK was used to co-design the software and hardware of the device. The architecture of this embedded system is shown in Fig. 8. The machine model was implemented in the FPGA hardware denoted by 'plb_machine.' The software exchanges data with 'plb_machine' through the CoreConnect Processor Local Bus (PLB) denoted by 'mb_plb.' The MicroBlaze is the 'master' device on the 'mb_plb,' while other devices are considered 'slaves.' The MicroBlaze generates the clock signal, and is responsible for coordinating data exchanges between the FPGA hardware and software.

When the system starts up, the MicroBlaze sends an initialization signal and initial state values to the 'plb_machine.' Then it launches the simulation, and begins controlling the execution of iterations, sending/updating the input data (i.e., T_L) to the 'plb_machine' at the beginning of each time step calculation. The simulation output data (i.e., $\mathbf{x}_n$ and T_e) are stored in the registers of 'plb_machine' as soon as each iteration is completed. While the next iteration proceeds in the FPGA, the MicroBlaze reads the data from the registers, and writes it on the DDR2 memory embedded on the development board through 'mb_plb.' In other words, the execution of iteration $n+1$ in 'plb_machine' takes place in parallel with the action of sending the data of iteration n to memory.

IV. Simulation Results

The parameters of the induction machine and voltage source used for simulation are shown in Table II. Based on the options provided by the Xilinx EDK, the Processor-Bus clock frequency was set to 66.67 MHz, a value less than the maximum clock frequency (89.735 MHz) in the post-place and route report of the Xilinx ISE. The simulation time-step h was 10^{-4} s. In this simple implementation, the simulation data were retrieved from the memory at the end of each simulation using the Xilinx Microprocessor Debugger.

TABLE II
Induction Machine and Voltage Source Parameters

R_s	0.087 Ω	P	4
R'_r	0.228 Ω	J	1.662 Kg·m^2
L_s	35.5 mH	ω_e	$2\pi 60$ rad/s
L'_r	35.5 mH	V_s	$460\sqrt{2/3}$ V
L_m	34.7 mH	P_{rated}	50 HP

The exact same machine model was also implemented in Matlab/Simulink. The verification of the results coming from the FPGA implementation was performed versus a Simulink simulation using the ODE23tb solver with a maximum time step of 10^{-5} s. Fig. 9 shows the transient response of all five state variables and the electromagnetic torque during free acceleration followed by a step load change (at $t = 1$ s) from both FPGA board and Simulink. The results from the FPGA are superimposed on the Simulink waveforms, but they are so close that differences cannot be distinguished.

To compare simulation speed, we ran the simulation using the ODE45 and ODE23 integration algorithms of Simulink with maximum step size of 10^{-4} s (typically the two "simplest" available solvers), because they are implementations of the explicit Runge–Kutta algorithm, albeit of a variable-step nature. The simulation speed was further increased by using the 'Accelerator' mode of Simulink, which replaces normal interpreted code with compiled code. The simulation times on an Intel Core2 Duo 2.2 GHz computer were 2.7 s for ODE45, and 2.0 s for ODE23. The FPGA simulation time was 36.6 ms, which represents a speed-up of two orders of magnitude. The simulation time will be further decreased if the clock frequency can be set to a higher value, if the simulation time step h is increased, or if a lower order integration algorithm (e.g., the trapezoidal algorithm) is used.

V. Conclusion

This paper presented the FPGA implementation of an induction machine dynamic simulation, using the RK4 numerical integration algorithm. The entire machine model has been developed using VHDL, synthesized, and implemented on an FPGA board. An optimal VHDL design should be sought for the purpose of economizing FPGA hardware resources, especially when the model has high complexity. A comparison between the simulation results from FPGA and Simulink demonstrates the validity of this implementation. Simulation speed gains of two orders of magnitude demonstrate the performance advantage of FPGAs compared to PC-based simulations. FPGAs represent an interesting possibility for simulating more complex electrical power and power electronics-based systems because of their flexibility, high processing rates and possibility to parallelize numerical integration computations.

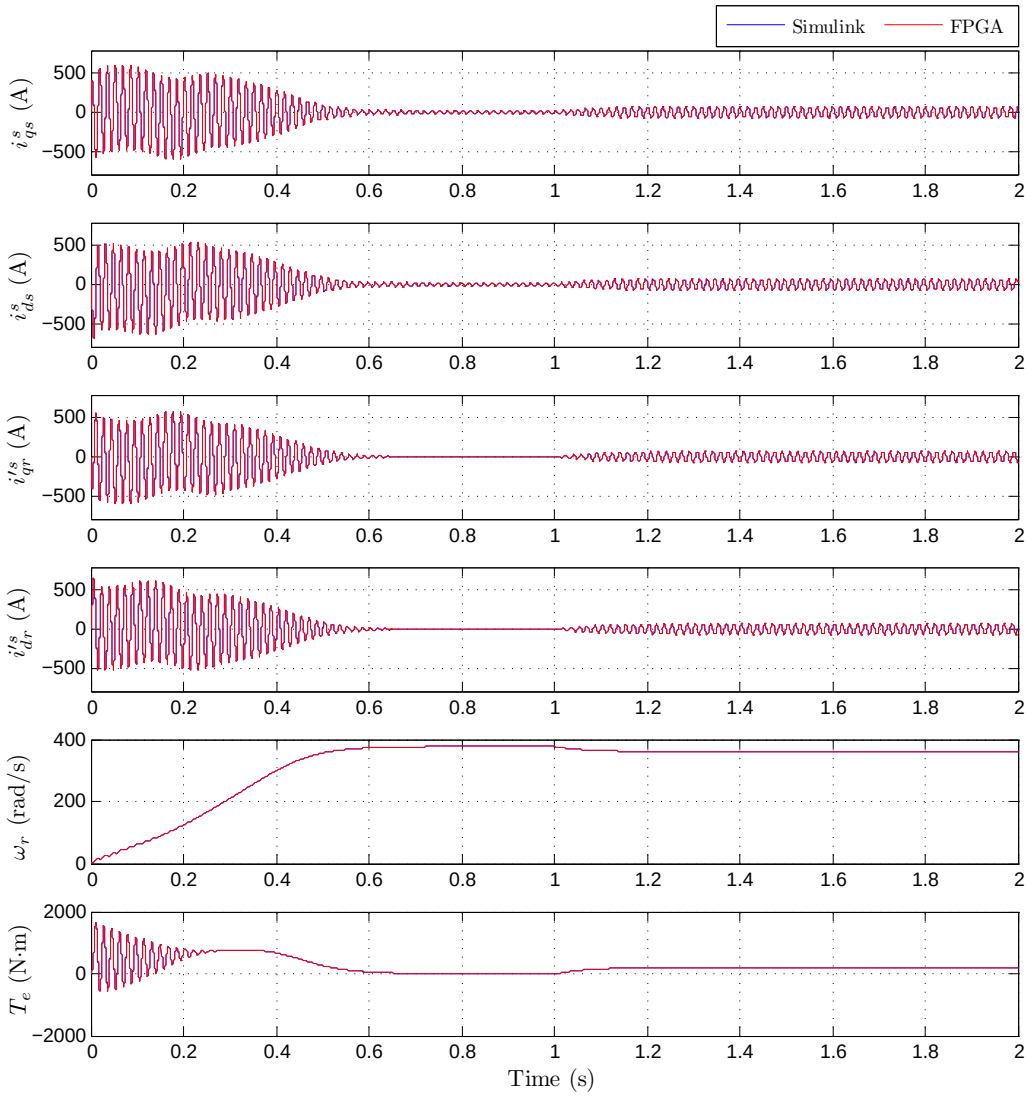

Fig. 9. Free acceleration characteristics of induction machine

REFERENCES

[1] M. Martar, M. Adbel-Rahman, and A.-M. Soliman, "FPGA-based real-time digital simulation," in *Int. Conf. Power Syst. Transients*, Montreal, Canada, Jun. 2005.

[2] P. Le-Huy, S. Guérette, L. A. Dessaint, and H. Le-Huy, "Real-time simulation of power electronics in power systems using an FPGA," in *Canadian Conf. Electr. Comp. Eng.*, May 2006, pp. 873–877.

[3] ——, "Dual-step real-time simulation of power electronic converters using an FPGA," in *IEEE Int. Symp. Ind. Electron.*, Montreal, Canada, Jul. 2006, pp. 1548–1553.

[4] J. C. G. Pimentel and H. Le-Huy, "Hardware emulation for real-time power system simulation," in *IEEE Int. Symp. Ind. Electron.*, Montreal, Canada, Jul. 2006, pp. 1560–1565.

[5] J. C. G. Pimentel, "Implementation of simulation algorithms in FPGA for real time simulation of electrical networks with power electronics devices," in *IEEE Int. Conf. Reconfig. Comp. & FPGA's*, Sep. 2006, pp. 1–8.

[6] 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. 1235–1246, Apr. 2007.

[7] T. Maguire and J. Giesbrecht, "Small time-step ($< 2\mu$Sec) VSC model for the real time digital simulator," in *Int. Conf. Power Syst. Transients*, Montreal, Canada, Jun. 2005.

[8] C. Dufour, J. Bélanger, S. Abourida, and V. Lapointe, "FPGA-based real-time simulation of finite-element analysis permanent magnet synchronous machine drives," in *IEEE Power Electron. Spec. Conf.*, Jun. 2007, pp. 909–915.

[9] P. C. Krause, O. Wasynczuk, and S. D. Sudhoff, *Analysis of Electric Machinery and Drive Systems*, 2nd ed. IEEE Press, 2002.

[10] W. Gautschi, *Numerical Analysis: An Introduction*. Boston: Birkhäuser, 1997.

[11] *ModelSim User's Manual Software Version 6.4a*, Mentor Graphics Corporation, Wilsonville, Oregon.

[12] P. P. Chu, *RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability*. Hoboken, New Jersey: Wiley-Interscience, 2006.

[13] J. Cavanagh, *Verilog HDL: Digital Design and Modeling*. Boca Raton, Florida: CRC Press, 2007.

[14] V. A. Pedroni, *Circuit Design with VHDL*. Cambridge, Massachusetts: MIT Press, 2004.

[15] Xilinx Virtex-5 FPGA Data Sheet. [Online]. Available: http://www.xilinx.com/onlinestore/silicon/online_store_v5.htm

[16] XUPV5-LX110T User Manual. [Online]. Available: http://www.xilinx.com/univ/xupv5-lx110T-manual.htm