Dynamic Simulation of Direct Torque Control of Induction Motors with FPGA Based Accelerators

Hamed S. Kia, Mohammad A. Zare, Rejesh G. Kavasseri
Electrical and Computer Engineering
North Dakota State University
Fargo, ND 58102, USA
{hamed.sajjadikia, m.zare, rajesh.kavasseri}@ndsu.edu

Cristinel Ababei
Electrical and Computer Engineering
Marquette University
Milwaukee, WI 53233, USA
cristinel.ababei@marquette.edu

***Abstract*—We present an efficient FPGA based implementation of a dynamic simulation framework for Direct Torque Control (DTC) of induction motors. The merit of the proposed DTC emulation framework lies in that it is completely implemented within an FPGA, and therefore can be easily utilized for exploratory system optimizations. The high performance and low area footprint of the proposed framework implemented on a Virtex-5 FPGA is the result of several innovative design techniques based on hardware time-multiplexing and the utilization of built-in dedicated floating point IP cores. Our experimental results demonstrate that when FPGAs are utilized as computational cores, the DTC dynamic simulation speed can be improved by almost an order of magnitude compared to a software implementation run in Matlab on a personal computer.**

***Index Terms*—Direct torque control, FPGA, Dynamic simulation**

I. Introduction

Dynamic simulation is the method of choice to design and study the performance of electric systems. One of the main challenges of this approach however is that the simulation time increases with the increase in the system complexity. This problem can be addressed successfully by employing FPGA (field programmable gate array) based hardware accelerators as processing engines, which can dramatically reduce the simulation time. Significant speed-up of the simulation time can be achieved thanks to the increased parallelism in processing offered by FPGAs, which have become increasingly powerful and cost effective. In addition, due to their reconfigurability FPGAs are flexible in supporting design updates and exploratory optimizations. FPGAs are also a perfect choice for real-time hardware-in-the-loop (HIL) simulations. For example, the studies in [1]–[4] utilize FPGAs for real-time simulation and modeling of electric power components.

The idea of exploiting FPGAs to speed up simulation time has been studied in the past. An extensive review of the use of FPGAs in industrial control systems applications can be found in [9]. Here we briefly discuss some related work in which FPGAs are used to perform computational tasks. In a recent work [10], Runge-Kutta (RK4) numerical integration algorithm is used to solve the differential equations that model an induction motor. The authors simulate the free acceleration of the induction machine on an FPGA and show that the simulation time decreases considerably compared to the same system modeled on a PC using Simulink. The RK4 numerical integration algorithm cannot be utilized however for the simulation of closed loop drive schemes due to the dependence of RK4 algorithm on future inputs at any given point in time. In another work by the same authors [11], a Doubly-Fed Induction Generator (DFIG) wind turbine is dynamically simulated on FPGAs. They show that the FPGA based simulation is 40 times faster than PC based Simulink simulation. A real-time FPGA-based simulation of an induction motor in stationary, rotor, and synchronous reference frames and the torque-speed characteristics in the 3 reference frames are presented in [13]. A finite element analysis simulator implemented on FPGAs is reported in [12] for HIL testing of motor drive controllers.

FPGAs have been also utilized to emulate various electric power components with the objective of modeling for simulation purposes. For example, [14] studies a Permanent Magnet Synchronous Generator (PMSG) drive system with fault testing capability and simulates it with Xilinx System Generator (XSG). The simulation results show that in-the-loop latency can be improved when FPGAs are utilized. A brushless DC (BLDC) motor is partially simulated on an FPGA in [15]. It demonstrates how 3-phase windings can be simulated on an FPGA by using Forward-Euler integration method to solve the machine equations. Real-time simulation of BLDC and induction machines with state-space modeling and floating-point cores using the RT-XSG toolkit for Matlab/Simulink is presented in [16].

Among various induction motor drive schemes, the Direct Torque Control (DTC) scheme has become an industry standard because of its simplicity and good dynamic response [5], [6]. Using pulse width modulated DTC methods, constant switching frequency can be achieved [7]. Increasing the sampling frequency can be utilized to reduce the torque ripple [8]. Therefore, the higher the sampling frequency is, the better the performance will be. In this high performance context, FPGAs represent a desirable approach to signal processing to achieve high speed simulations. Therefore, in this paper, we develop a dynamic simulation framework for DTC and implement it fully on an FPGA. Our goal is to achieve the fastest simulation times. The proposed DTC simulation framework deployed on

978-1-4799-2079-2/13/$31.00

a Virtex-5 FPGA can speed-up the simulation time by up to 92.06% compared to a software based implementation run in Matlab on a personal computer.

The remainder of this paper is organized as follows. In the next section II, we describe the induction motor model and DTC scheme. In section III, we describe the proposed FPGA implementation. Specifically, we discuss several innovative design techniques that help us to achieve fast clock frequency with minimum area utilization. We report experimental and simulation results in section IV.

II. Description of the Direct Torque Control Scheme

The block diagram of the Direct Torque Control (DTC) scheme is shown in Fig.1. It is comprised of the following main components: induction machine model, estimation block, switching table, torque and flux comparators, sector selector, and inverter. Each of these blocks is described in the following subsections.

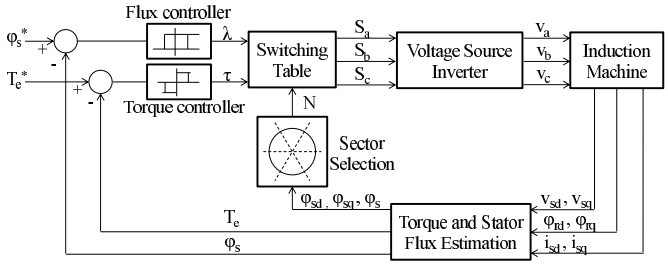


Fig. 1. Illustration of the block diagram the Direct Torque Control (DTC) scheme implemented on FPGAs.

A. Induction Machine

The behavior of the induction machine in the stationary reference frame can be modeled by the following equations (1)-(4) [17].

$$\frac{di_{sd}}{dt} = -\gamma i_{sd} + \frac{L_m}{\sigma L_s L_r T_r}\phi_{rd} + \frac{L_m}{\sigma L_s L_r}\frac{P}{2}\omega_m \phi_{rq} + \frac{1}{\sigma L_s} v_{sd} \tag{1}$$

$$\frac{di_{sq}}{dt} = -\gamma i_{sq} - \frac{L_m}{\sigma L_s L_r}\frac{P}{2}\omega_m \phi_{rd} + \frac{L_m}{\sigma L_s L_r T_r}\phi_{rq} + \frac{1}{\sigma L_s} v_{sq} \tag{2}$$

$$\frac{d\phi_{rd}}{dt} = \frac{L_m}{T_r} i_{sd} - \frac{1}{T_r}\phi_{rd} - \frac{P}{2}\omega_m \phi_{rq} \tag{3}$$

$$\frac{d\phi_{rq}}{dt} = \frac{L_m}{T_r} i_{sq} - \frac{1}{T_r}\phi_{rq} + \frac{P}{2}\omega_m \phi_{rd} \tag{4}$$

where we use the notations $\sigma = 1 - \frac{L_m^2}{L_s L_r}$ and $\gamma = \frac{R_s + \frac{L_m^2}{L_r}}{\sigma L_s}$; $T_r = \frac{L_r}{R_r}$ is the rotor constant; ω_m is the angular mechanical speed of the motor; R_s and R_r are stator and rotor resistances; L_s and L_r are the stator and rotor inductances, and L_m is the magnetizing inductance. The state variables in these differential equations are the dq axes stator current and rotor flux components (i_{sd}, i_{sq}, ϕ_{rd}, and ϕ_{rq}).

The 3-phase voltages of the stator, transformed to dq components are obtained from the positive sequence dq transformation matrix:

$$\begin{bmatrix} v_{sd} \\ v_{sq} \\ v_{s_0} \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -0.5 & -0.5 \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} v_{sa} \\ v_{sb} \\ v_{sc} \end{bmatrix} \tag{5}$$

The state space representation of the model is:

$$\frac{dx(t)}{dt} = Ax(t) + Bv(t) \tag{6}$$

$$\frac{d\omega_m}{dt} = \frac{1}{J}\left(\frac{3}{2}\frac{P}{2}\frac{L_m}{L_r}(\phi_{rd} i_{sq} - \phi_{rq} i_{sd}) - T_L\right) \tag{7}$$

where

$$x(t) = \begin{bmatrix} i_{sd} \\ i_{sq} \\ \phi_{rd} \\ \phi_{rq} \end{bmatrix},\ v(t) = \begin{bmatrix} v_{sd} \\ v_{sq} \end{bmatrix}$$

$$A = \begin{bmatrix} -\gamma & 0 & \frac{L_m}{\sigma L_s L_r T_r} & \frac{L_m}{\sigma L_s L_r}\omega \\ 0 & -\gamma & -\frac{L_m}{\sigma L_s L_r}\omega & \frac{L_m}{\sigma L_s L_r T_r} \\ \frac{L_m}{T_r} & 0 & -\frac{1}{T_r} & -\omega \\ 0 & \frac{L_m}{T_r} & \omega & -\frac{1}{T_r} \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{\sigma L_s} & 0 \\ 0 & \frac{1}{\sigma L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

where T_L stands for the load torque and $\omega = \frac{P}{2}\omega_m$ is the rotor angular frequency.

To solve numerically the aforementioned differential equation set, we propose to use a discretization method based on forward shift approximation [18] due to its low computational complexity and good stability. The shift approximation can be calculated by:

$$p = \frac{(z-1)}{T} \tag{8}$$

where T is the integration time step and z is discrete mode forward shift operator.

Applying (8) to (6) yields:

$$x(k+1) = A_z x(k) + B_z v(k) \tag{9}$$

where $x(k) = \begin{bmatrix} i_{sd}(k) & i_{sq}(k) & \phi_{rd}(k) & \phi_{rq}(k) \end{bmatrix}^T$, $A_z = (I + AT)$, and $B_z = BT$ [19].

B. Estimation Block

The torque and stator flux estimation block (see Fig.1) estimates the motor electromechanical torque T_e and stator flux linkage ϕ_s described by the following equations.

$$T_e = \frac{3}{2}\frac{P}{2}\frac{L_m}{L_r}(\phi_{rd} i_{sq} - \phi_{rq} i_{sd}) \tag{10}$$

$$\phi_{sd} = \int (v_{sd} - R_s i_{sd}) dx \tag{11}$$

$$\phi_{sq} = \int (v_{sq} - R_s i_{sq}) dx \tag{12}$$

$$\phi_s = \sqrt{\phi_{sd}^2 + \phi_{sq}^2} \tag{13}$$

The evaluation of ϕ_{sd} and ϕ_{sq} from equations (11), (12) is done according to the following numerical estimation:

$$\phi_{sd} = \phi_{sd_{old}} + T_s(v_{sd} - R_s i_{sd}) \tag{14}$$

$$\phi_{sq} = \phi_{sq_{old}} + T_s(v_{sq} - R_s i_{sq}) \tag{15}$$

C. Torque and Flux Comparators

The values estimated by the estimation block are compared with the reference torque and stator flux values using 3 and 2-level hysteresis torque and flux comparators, respectively as shown in Fig.1. The result of this comparison is utilized to read the corresponding appropriate inverter switching signals from the switching table also shown in the block diagram from Fig.1. The contents of the switching table is presented in Table I.

TABLE I
CONTENTS OF THE SWITCHING TABLE BLOCK FROM THE DTC DIAGRAM SHOWN IN FIG.1.

(S_a, S_b, S_c)							
ϕ, τ, N		N = 1	N = 2	N = 3	N = 4	N = 5	N = 6
ϕ = 1	τ = 1	(1,1,0)	(0,1,0)	(0,1,1)	(0,0,1)	(1,0,1)	(1,0,0)
	τ = 0	(1,1,1)	(0,0,0)	(1,1,1)	(0,0,0)	(1,1,1)	(0,0,0)
	τ = -1	(1,0,1)	(1,0,0)	(1,1,0)	(0,1,0)	(0,1,1)	(0,0,1)
ϕ = 0	τ = 1	(0,1,0)	(0,1,1)	(0,0,1)	(1,0,1)	(1,0,0)	(1,1,0)
	τ = 0	(0,0,0)	(1,1,1)	(0,0,0)	(1,1,1)	(0,0,0)	(1,1,1)
	τ = -1	(0,0,1)	(1,0,1)	(1,0,0)	(1,1,0)	(0,1,0)	(0,1,1)

D. Sector Selector

The dq coordinate plane is divided into 6 sectors. The inverter switching signals can be determined simply by the stator flux vector position and the outputs of the hysteresis comparators.

E. Inverter

A three phase inverter is utilized to provide the needed voltage for the stator of the induction machine. The inverter input DC voltage is considered to be 170 V, which practically can be provided by a three phase rectifier. The switching signals applied to the inverter come from the hysteresis controllers based on the reading from the switching table.

III. DESCRIPTION OF THE FPGA IMPLEMENTATION

In this section, we describe the FPGA based hardware implementation of the DTC scheme. More specifically, we discuss several innovative design techniques based on hardware time-multiplexing and the utilization of built-in dedicated floating point IP cores. These techniques help us achieve the best performance with minimal FPGA resource utilization.

One of the main challenges in the FPGA based hardware design of the proposed DTC scheme is the implementation of the complex functions, such as the square root function necessary in equation (13). This difficulty is exacerbated by the fact that we must work with floating point number representation instead of simple integers. Consequently, we utilize the IEEE-754 single precision standard for floating point computations. Therefore, all DTC variables in our implementation shown in Fig.1 are 32-bit long. This in turn requires us to work with floating point operators, whose hardware implementation occupy larger FPGA areas compared to those of integer operators.

To address this issue we develop our hardware implementation using a unified approach for the induction machine block (characterized by equations (1), (2), (3), (4), and (7)) and the part of the estimation block (characterized by equations (11) and (12)) from Fig.1. The architecture of this combination is shown in Fig.2.

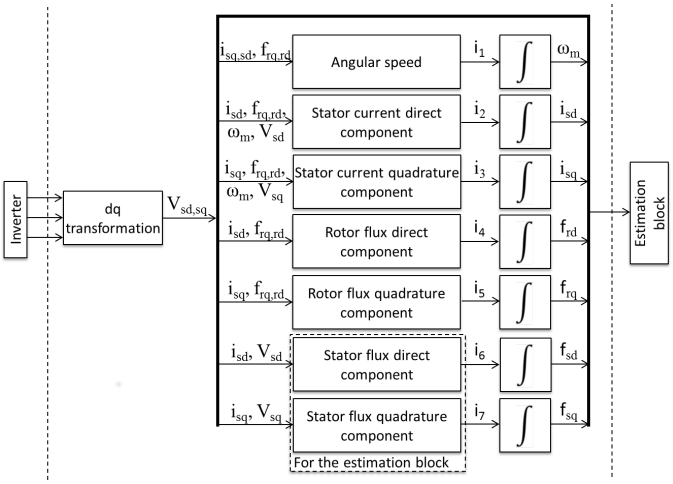


Fig. 2. Illustration of the induction motor architecture. The stator flux direct and quadrature components from the estimation block are implemented alongside the induction motor variables to save area.

The key component of this architecture is the integrator, which is implemented with a floating point multiplier and a floating point adder. As seen in Fig.2, we need a total of 7 integrators: 5 integrators to model the induction motor plus 2 integrators to implement the estimation block. In order to minimize the area occupied by the integrators, we propose to utilize a single integrator, which is time shared to process all seven required integrations by utilizing the scheme with multiplexers and demultiplexers shown in Fig.3.

To control the synchronization among different blocks of the system, we utilize a global counter, which basically rep-

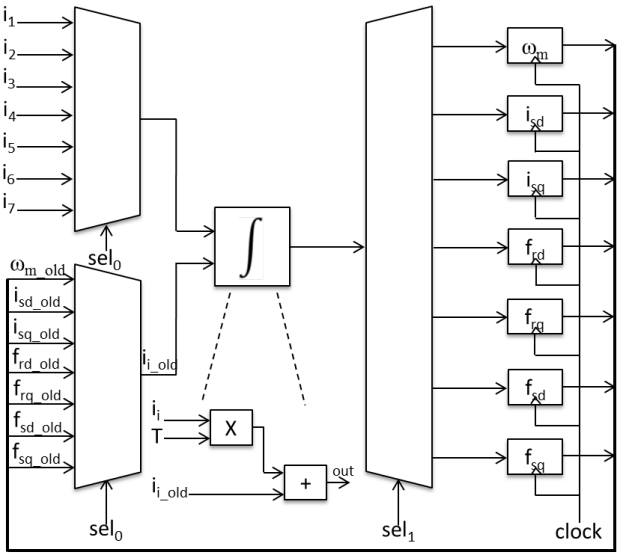


Fig. 3. Using multiplexers and demultiplexers to eliminate the need for multiple integrators is one of the design optimizations utilized to save area.

resents the primary element of the control path of our DTC architecture. For example, signals sel_0 and sel_1 from Fig.3 are generated based on the value of this counter. To further save area, we use a similar area minimization technique to reduce the number of floating point multipliers and adders in implementing different blocks shown in Fig.2.

As mentioned earlier, one design challenge is the implementation of the square root function, which is required as part of the estimation block, described by the equations (10), (11), (12), and (13). We address this by employing the floating point square root IP core provided by Xilinx. The estimation block calculates also the electromechanical torque, estimated using equation (13), where ϕ_{rd}, ϕ_{sq}, i_{sd}, and i_{sq} are already available as computed by the induction motor block.

The torque and flux comparators compare the estimated torque and flux values of the DTC scheme with the user set reference torque and flux values using 2 and 3-level hysteresis operators. Implementing hysteresis functions requires the *Greater than or equal* and *Less than or equal* operators. However, because we work with floating point 32-bit variables (i.e., IEEE-754 standard), we can not utilize simply the *if-statement* based programming to implement these operators. We addressed this issue by again employing Xilinx floating point comparison IP cores to implement these functions.

To implement the sector selection block from Fig.1, we utilize a simple look-up-table (LUT). The output of the look-up-table is determined based on the comparison between ϕ_{sd}, ϕ_{sq}, $\frac{\sqrt{3}}{2}\phi_s$, and 0. Similarly to [8], this approach is very efficient and simpler compared to previous methods that utilize complex functions such as *arctan* or the CORDIC algorithm [21]. We used 6 look-up-tables to implement the switching table block. A look-up-table is associated with each of the 6 sectors in the sector selection block. Therefore, the output of the switching table block is determined by the output of the sector selection block and the 3 bits provided by the torque (2 bits) and flux (1 bit) comparators. Finally, the last block in the DTC scheme is the inverter. The inverter has a 3-phase voltage source structure. The inverter output is determined by the switching table output.

The DTC scheme depicted in Fig.1 is redrawn in Fig.4 to illustrate the number of clock cycles that each block needs to complete its operation. Each run or iteration of the DTC scheme simulation on the FPGA takes 88 clock cycles. This number is primarily determined by the use of multiplexers and demultiplexers in the induction motor model to reduce area overhead. As already discussed, we use a global counter to count the clock cycles necessary for each block and to synchronize the operation of the whole system. Once the counter reaches a certain number signifying that the operation of a block is completed, the outputs of the block are sampled and applied as inputs to the next block. The select signals for all the multiplexers and demultiplexers shown in Fig.3 are also generated based on the global counter values.

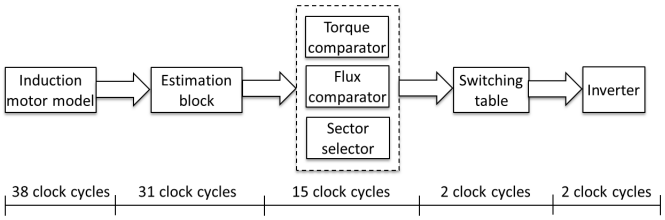


Fig. 4. Minimum number of clock cycles required by the building blocks of the proposed DTC scheme implementation. It takes 88 clock cycles to run the DTC loop once.

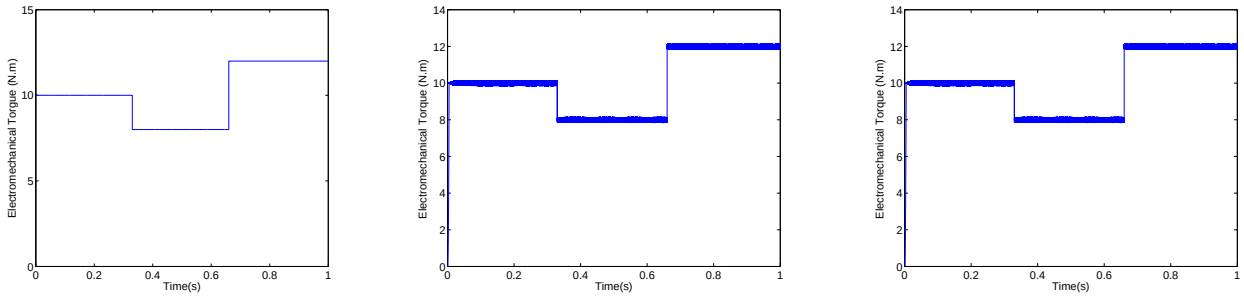
IV. Experimental Results

The values of the induction machine parameters utilized in our experiments are shown in Table II. We utilize a time step of $1\mu s$. We implemented the proposed DTC scheme from Fig.1 by coding it in Verilog, synthezing it with Xilinx ISE Webpack tools, and deploying it on a Xilinx Virtex-5 FPGA chip [22]. Table III reports the implementation results. The maximum clock frequency is 25 MHz while the entire design occupies 91% of the available slice LUTs and 12% of the available slice registers on the Virtex-5 chip.

TABLE II
Parameters of the modeled induction motor

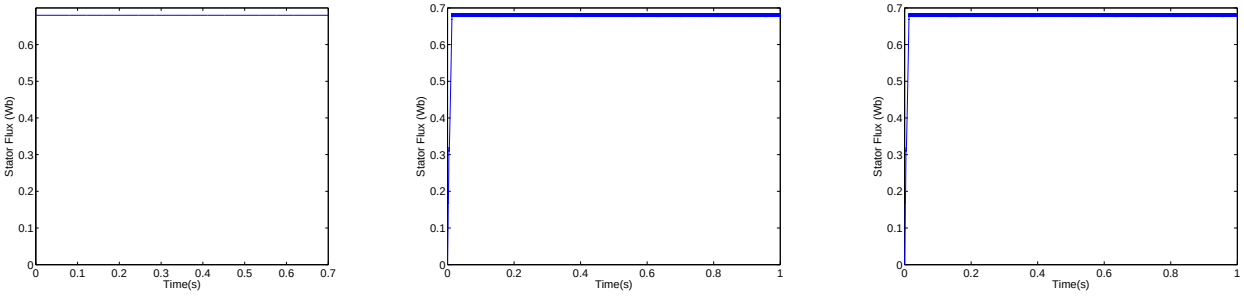
R_s	0.18 Ω
R_r	0.50 Ω
L_s	0.0553 H
L_r	0.0560 H
L_m	0.0538 H
J	1.0033 kg.m^2
P	4

For comparison purposes, we implemented the same DTC scheme in Matlab [23]. We verified the results from the FPGA based implementation by comparing them with the simulation results obtained from Matlab. For example, when a stepwise



(a) (b) (c)

Fig. 5. (a) Reference torque. (b) Matlab simulation result for electromechanical torque. (c) FPGA simulation result for electromechanical torque.



(a) (b) (c)

Fig. 6. (a) Reference flux. (b) Matlab simulation result for stator flux. (c) FPGA simulation result for stator flux.

TABLE III
IMPLEMENTATION RESULTS OF THE DTC SCHEME ON XILINX VIRTEX 5

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	3698	28800	12%
Number of Slice LUTs	26423	28800	91%
Maximum Clock Frequency	25 MHz		

torque command is applied as the torque reference value of the studied DTC scheme, the responses from Matlab-based and FPGA-based simulations are shown in Fig.5. The simulation time considered is 1 s. First, we note that the resulting torque response has a good dynamic response and relatively low ripple. This is mostly because of the high sampling frequency of the simulation (1 MHz). Second, as it can be seen in Fig.5, the simulation results from both Matlab and FPGA implementations are identical as expected. Fig.6 shows the results from the Matlab and FPGA based simulations for a command in the stator flux. Again, we observe that as expected, the two waveforms in Fig.6.b and Fig.6.c coincide. The flux values are also identical for both Matlab and FPGA based simulations. This is shown in Fig.7, where the flux path (which has a circular trajectory, as expected) of both simulation methods is shown.

Finally, in order to compare the simulation speed of the proposed FPGA based implementation of the DTC scheme with that of the Matlab simulation, we ran the simulation with a step size of $1\mu s$. The Matlab simulation was done on a desktop computer running Windows 7 on an Intel Core i5 processor clocked at 2.5 GHz. The Matlab simulation took 44.38 s to simulate 1 s of DTC operation while it took the proposed FPGA based simulation only 3.52 s to perform the same computations. This means a 92.06% reduction in simulation time. Notice that if the simulation time step is reduced this improvement in simulation time would be even higher.

V. CONCLUSION

We presented a novel efficient and cost effective FPGA based implementation of a dynamic simulation framework for Direct Torque Control (DTC). Such a simulation framework can be very useful to designers who can benefit from the significantly improved simulation times and the flexibility offered by the reconfigurability of the overall system, which can be exploited for design optimization. Experimental results showed that the proposed FPGA based simulation framework achieves up to 92.06% reduction in simulation time compared to Matlab based approaches.

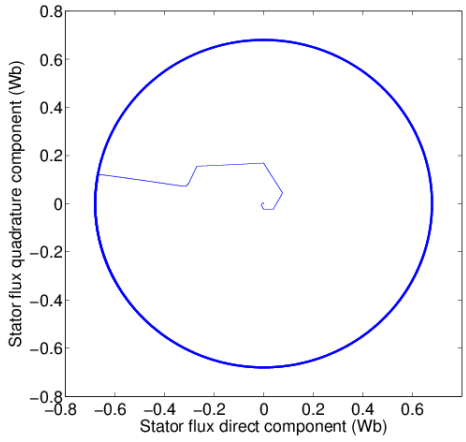
REFERENCES

[1] M. Martar, M. Adbel-Rahman, and A.-M. Soliman, "FPGA based realtime digital simulation," *Int. Conf. on Power Systems Transients*, 2005.

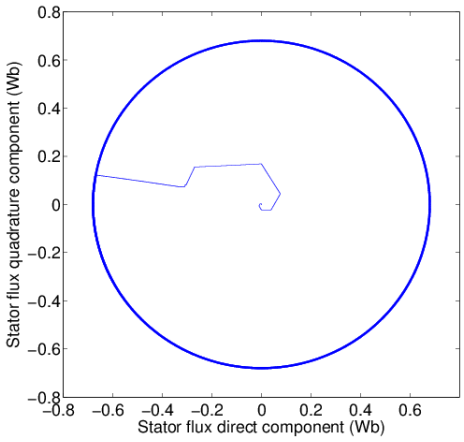
[2] G. Parma and V. Dinavahi,"Real-time digital hardware simulation of power electronics and drives," *IEEE Trans. on Power Delivery*, vol. 22, no. 2, pp. 1235-1246, Apr. 2007.

[3] J. Pimentel, "Implementation of simulation algorithms in FPGA for real time simulation of electrical networks with power electronics devices," *IEEE Int. Conf. on Reconfigurable Computing and FPGAs*, 2006.

[4] P. Le-Huy, S. Guaerette, L. A. Dessaint, and H. Le-Huy, "Real-simulation of power electronics in power systems using an FPGA," *Canadian Confe Electrical and Computer Engineering*, 2006.



(a)



(b)

Fig. 7. (a) Matlab simulation result for flux path. (b) FPGA based simulation result for flux path.

[5] I. Takahashi and T. Noguchi, "A new quick-response and high efficiency control strategy of an induction motor," *IEEE Trans. on Industry Application*, vol. 22, no. 5, pp. 820-827, Sept./Oct. 1986.

[6] M. Depenbrock, "Direct self control (DSC) of inverter fed induction machine," *IEEE Trans. on Power Electronics*, vol. 3, no.4, pp.420-429, Oct. 1988.

[7] D. Casadei, F. Profumo, G. Serra, and A. Tani, "FOC and DTC: two viable schemes for induction motors torque control," *IEEE Trans. on Power Electonics*, vol. 17, no. 5, pp. 779-787, Sept. 2002.

[8] T. Sutikno, N. Idris, A. Jidin, and M. Cirstea, "An improved FPGA implementation of direct torque control for induction machines," *IEEE Trans. on Industrial Informatics*, Oct. 2012.

[9] E. Monmasson and M. Cirstea, "FPGA design methodology for industrial control systems - a review," *IEEE Trans. on Industrial Electronics*, vol. 54, no. 4, pp. 1824-1842, Aug. 2007.

[10] H. Chen, S. Sun, D.C. Aliprantis, and J. Zambreno, "Dynamic simulation of electric machines on FPGA boards," *IEEE Int. Electric Machines and Drives Conf. (IEMDC)*, 2009.

[11] H. Chen, S. Sun, D. C. Aliprantis, and J. Zambreno, "Dynamic simulation of DFIG wind turbines on FPGA boards," *Power and Energy Conf.*, 2010.

[12] C. Dufour, J. Beilanger, and V. Lapointe, "FPGA-based ultra-low latency HIL fault testing of a permanent magnet motor drive using RT-LAB-XSG," *Int. Conf. on Power System Technology*, 2008.

[13] S. Tola and M. Sengupta, "Real-time simulation of an induction motor in different reference frames on an FPGA platform," *IEEE Int. Conf. on Power Electronics, Drives and Energy Systems*, 2012.

[14] C. Dufour, J. Belanger, S. Abourida, and V. Lapointe "FPGA-based real-time simulation of finite-element analysis permanent magnet synchronous machine drives," *Power Electronics Specialists Conf. (PESC 2007)*, 2007.

[15] T. Schulte and J. Bracker, "Real-time simulation of BLDC motors for hardware-in-the-loop applications incorporating sensorless control," *IEEE Int. Symp. on Industrial Electronics (ISIE)*, 2008.

[16] T.O. Bachir, J. David, C. Dufour, and J. Belanger, "Effective FPGA based electric motor modeling with floating-point cores," *IEEE Industrial Electronics Society (IECON)*, 2010.

[17] B.K. Bose. Modern Power Electronics and AC Drives, Englewood Cliffs, NJ: Prentice-Hall, 2001.

[18] O. Vainio, S.J. Ovaska, and J.J. Pasanen, "A digital signal processing approach to real-time AC motor modeling," *IEEE Trans. on Industrial Electronics*, vol. 39, no. 1, pp. 36-45, 1992.

[19] L. Charaabi,"FPGA-based fixed point implementation of a real-time induction motor emulator," *Advances in Power Electronics*, 2012.

[20] T. Sutikno, A. Jidin, and N. Idris, "New approach FPGA-based implementation of discontinuous SVPWM," *Turkish Journal of Electrical Engineering and Computer Science*, vol. 18, pp. 499-514, 2010.

[21] C. Kowalski, J. Lis, and T.O Kowalska, "FPGA implementation of DTC control method for the induction motor drive," *Int. Conf. on Computer as a Tool (EUROCON)*, 2007.

[22] http://www.xilinx.com

[23] http://www.mathworks.com