TELKOMNIKA, Vol.11, No.2, February 2013, pp. 747~753
ISSN: 2302-4046

A Model of FPGA-based Direct Torque Controller

Tole Sutikno*[1, 2], Nik Rumzi Nik Idris[2], Aiman Zakwan Jidin[3], Auzani Jidin[4]
[1]Department of Electrical Engineering, Universitas Ahmad Dahlan
[2]Department of Energy Conversion, Universiti Teknologi Malaysia
[3]IP Design Department, Altera Corporation (M) Sdn Bhd, Penang, Malaysia
[4]Department of Power Electronics & Drives, Universiti Teknikal Malaysia Melaka
*Corresponding author, e-mail: tole@ee.uad.ac.id

Abstract

This paper presents a generic model of a fully FPGA-based direct torque controller. This model is developed using two's-complement fixed-point format approaches, in register-transfer-level (RTL) VHDL abstraction for minimizing calculation errors and consuming hardware resource usage. Therefore, the model is universal and can be implemented for all FPGA types. The model is prepared for fast computation, without using of CORDIC algorithm, a soft-core CPU, a transformation from Cartesian-to-polar coordinates, and without the help of third-party applications. To get simpler implementation and fast computation, several methods were introduced: i) the backward-Euler approach to calculate the discrete-integration operation of stator flux, ii) the modified non-restoring method to calculate complicated square-root operation of stator flux, iii) a new sector analysis method. The design, which was coded in synthesizable VHDL in RTL abstraction for implementation on Altera DE2-board has produced very-precise calculations, with minimal error when being compared to MATLAB/Simulink double-precision calculation.

Keywords: *DTC, Fixed-point, FPGA, RTL, two's complement, VHDL*

1. Introduction

Direct Torque Control (DTC) which was proposed by Takahashi in 1986 [1], has gained popularity since it can provide fast instantaneous torque control with simple control structure. Ideally, the error or ripple of the torque (or flux) is restricted within the hysteresis band, so that the output torque (or flux) will satisfy its demand. Unfortunately, in practice, as the hysteresis controller performs in a discrete computation approach, this is impossible to achieve due to the delay between the instant the torque is sampled and the instant the corresponded switching status is passed to the inverter [2]. The ripple might exceed beyond the hysteresis bands, i.e. overshoot or undershoot condition, and hence tends to select the reverse voltage vector that causes rapid increase/decrease of the torque [3]. This, consequently, will produce larger torque ripples and slightly degrade the performance of DTC.

Several methods were proposed to minimize the output torque ripple. These include the use of space vector modulation (SVM) [4- 5- 6- 7], the injection of dithering signal [8], the use of constant carrier frequency [3] and recently, the hysteresis based DTC with predictive control [9- 10- 11]. All these methods require some modifications and knowledge of machine parameters that somehow will complicate the simple structure of the DTC and will increase the control sensitivity of the DTC. Moreover, the effectiveness in minimizing the output torque ripple using those methods can be achieved if only a high switching frequency is applied. Unfortunately, microcontrollers and DSPs (include DSPACE 1xxx or TMS C2xxx, etc) inadequate for the above requirement.

The best way to perform the DTC algorithm at highest sampling rate is the use of Field Programmable Gate Arrays (FPGA). Some works used a combination of DSP and FPGA, where the computational burden of the DSP is reduced by distributing some tasks of DTC algorithms to be executed by the FPGA. In this a way, the sampling period to execute the overall DTC algorithms can minimize the output torque ripple [12- 13- 14]. However, the combinations of controllers increase the cost and complexity of the interfacing circuit, and are not practical for commercialization purpose. Some attempts [15- 16] implemented entire DTC algorithms utilizing

Received September 17, 2012; Revised December 26, 2012; Accepted January 14, 2013

single FPGA. However, the VHDL or Verilog coding applied in [15- 16] was generated using third party simulation software, which is not fully optimized to achieve fast sampling frequency.

This paper presents an effective way to design, simulate and implement the hysteresis-based DTC utilizing FPGAs. The main contribution of this paper is the development of the DTC using VHDL code on the FPGA, i.e. from scratch, the code being optimized to achieve a sampling frequency of 200 kHz. The VHDL is chosen in order to shorten the design cycle and to manage such an increased complexity, as it offers support (test environment) at all levels of abstraction (behavioural, structural, physical) [17]. With the high sampling frequency, it is therefore, possible for the torque ripple to be restricted within its hysteresis band and hence minimize the ripple by reducing the band size.

2. The Proposed DTC Model

Figure 1 shows the proposed fully FPGA-based DTC model. All modules (in digital side) are designed by using VHDL. All calculations in the modules are conducted in two's complement fixed-point arithmetic. The explanation of the each module will be presented later.

2.1. Current Sensing Acquisition System

The current sensing acquisition system has two parts. First part is current sensing system which composed of current transducer LA 55-P, and second part is ADC system based on AD7862-10 simultaneous sampling dual 250 kSPS 12-bit ADC. The parts are analog side of the DTC.

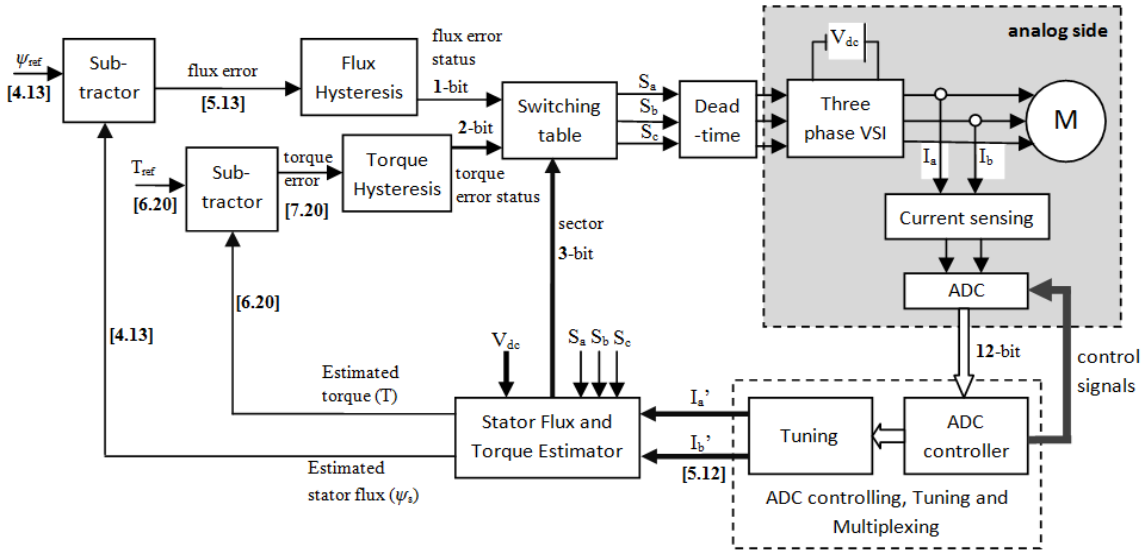


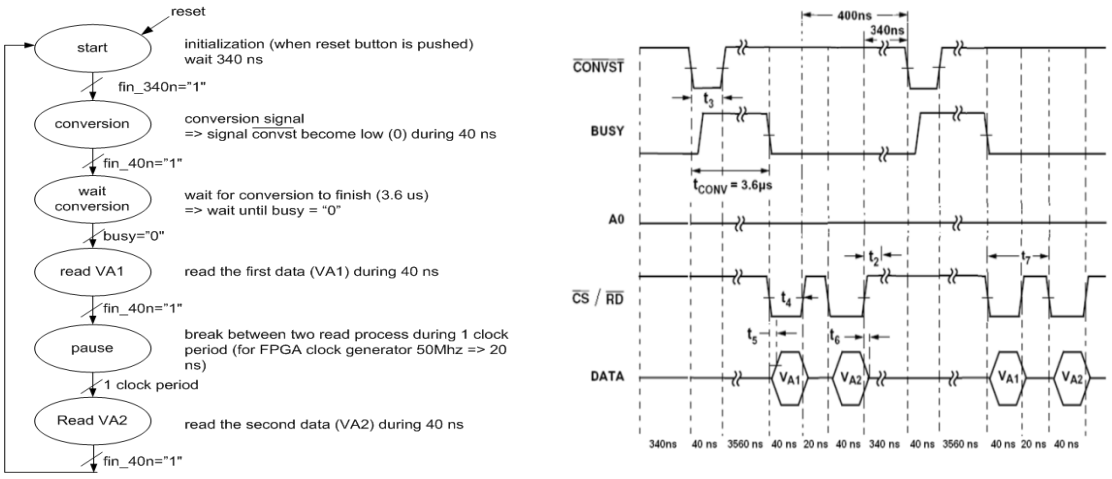
Figure 1 The proposed fully FPGA-based DTC model

2.2. ADC Controller

The module responsible for acquisition current information (I_a and I_b) of the ADC system. Figure 2 shows state machine and timing operation diagram of the ADC controller. In this work, only first channel is used.

2.3. Scaling and Tuning

The tuning module in Figure 1 is used to get the current values (I_a and I_b) which close to the actual motor current values. As it's known that the input the module is not yet real motor current-representation values, but they are still in voltage-representation values which are output of the ADC controller. Output of the module is represented in [5.12], where 5 is signed two's complement integer and 12 is number of fractional bits.



(a). state machine of the ADC controller

(b). timing operation diagram of the ADC controller

Figure 2 State machine and timing operation diagram of the ADC controller based on AD7862

2.4. Stator Flux and Torque Estimators

The calculations of the flux and torque estimator are implemented in some modules, as shown in Figure 3.

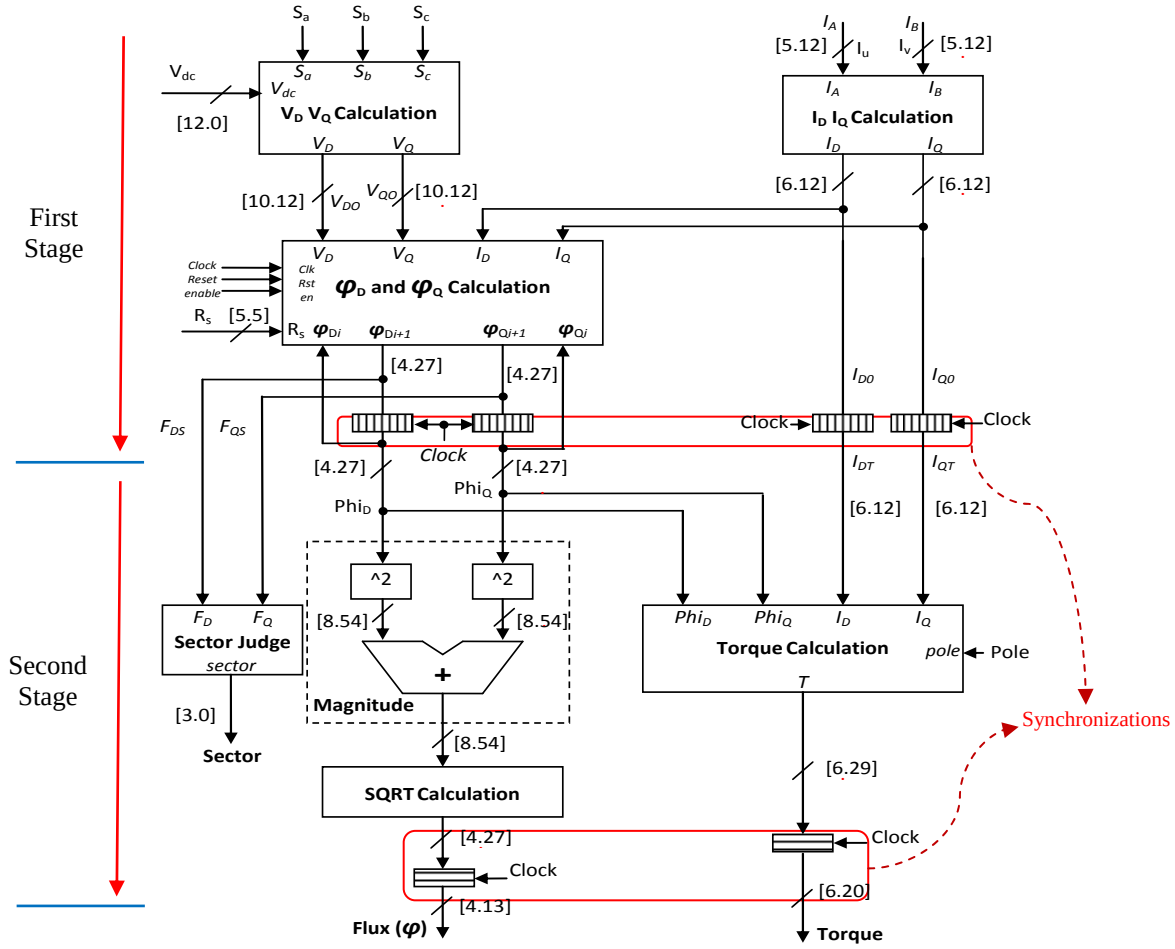
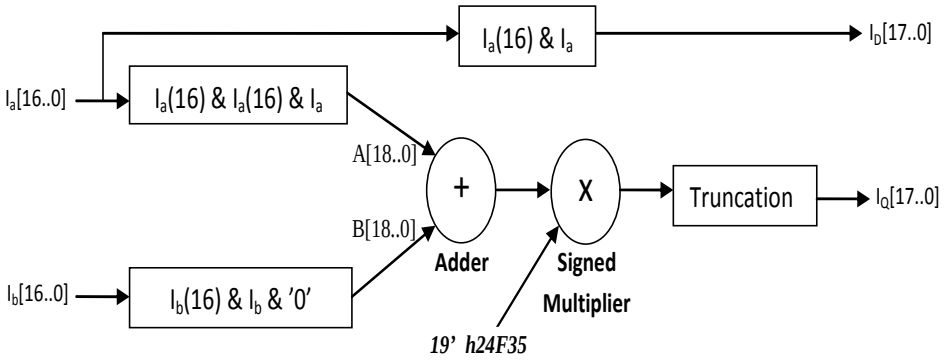


Figure 3 The behavioural model of the proposed stator flux and torque estimators

The estimator is the heart of DTC. The successful implementation of DTC very much depends on the module. To get simpler implementation and faster computation of the estimator, several methods were introduced: i) the backward Euler approach to calculate the discrete integration operation of stator flux, ii) the modified non-restoring method to calculate complicated square root operation of stator flux, iii) a new sector analysis method.

2.4.1 I_D and I_Q Calculator

This module is to transform the motor stator currents into DQ refer to equation (1) and (2). In Figure 4, the value of $\frac{1}{3}\sqrt{3}\ x\ 2^{18}$ is represented as "19' h24F35" (the 19 is number of bits, and the h24F35 is value of 151349 in hexadecimal).




```
library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;

entity Id_Iq is
port(
    Ia, Ib : in std_logic_vector (16 downto 0);    --[5.12]
    Id, Iq : out std_logic_vector(17 downto 0));   --[6.12]
end entity;

architecture behavioral of Id_Iq is
-- sqrt3per3 (=sqrt(3)/3) = 0.5773506 in [1.18] => "01001001111001101 01"
constant sqrt3per3 : signed (18 downto 0) := "010010011110011010 1";
signal I1, I2, I3  : std_logic_vector (18 downto 0);
signal y1          : std_logic_vector (37 downto 0);
begin

I1 <= Ia(16)& Ia(16) & Ia;     -- [5.12] -> [7.12]
Id <= I1(17 downto 0);         -- Id in [6.12]
I2 <= Ib(16) & Ib & '0';       -- Ib*2; [5.12] -> [7.12]
I3 <= std_logic_vector (signed(I1) + signed(I2));  -- Ia + 2*Ib; [7.12]
y1 <= std_logic_vector (signed(I3) * sqrt3per3);   -- (sqrt(3)/3)*(Ia + 2Ib)       [8.30]
Iq <= y1(35 downto 18);                    -- truncated to [6.12]
end behavioral;
```

Figure 4 The RTL model of the I_D and I_Q calculations, and its VHDL source code

The output of the signed multiplier is represented on 38-bits, as [8.30]. However, the I_Q is only represented on 18-bits as [6.12] to minimize hardware resource, so the 38-bit [8.30] is truncated to become 18-bit [6.12]. Based on the evaluation result, the 18-bit has been considered suitable to represent I_Q precisely.

$$I_D = I_a \tag{1}$$

$$I_Q = \frac{\sqrt{3}}{3}(I_a + 2I_b) \tag{2}$$

2.4.2 V_D and V_Q Calculator

The function of this module is to calculate the stator voltages into DQ components refer to equation (3) and (4). The same concept (in I_D and I_Q Calculator) is used for implementing this module.

$$V_D = \frac{V_{dc}}{3}(2S_a - S_b - S_c) \quad (3)$$

$$V_Q = \frac{\sqrt{3}}{3}V_{dc}(S_b - S_c) \quad (4)$$

2.4.3 Stator Flux Calculator

After the calculation of the DQ components of current and voltage, the DQ stator flux (plus a filter) is calculated in this module refer to equation (5) and (6). The equations are the backward Euler approach for calculating the discrete integration operation of stator flux. Next, the calculation of the magnitude of the stator flux refers to equation (7). The modified non-restoring method is used to calculate the square root operation.

$$\varphi_D = (\varphi_{D_{old}} + (V_D - R_S I_D)T_s)(1 - \omega_c * T_s) \quad (5)$$

$$\varphi_Q = (\varphi_{Q_{old}} + (V_Q - R_S I_Q)T_s)(1 - \omega_c * T_s) \quad (6)$$

$$\varphi_s = \sqrt{\varphi_D^2 + \varphi_Q^2} \quad (7)$$

2.4.4 Sector Calculator

The work has used a simpler method to judge the sector of the voltage vectors, which is modified from [18]. By using Karnaugh map simplification, it only involves two comparisons.

2.4.5 Torque Calculator

The function of this module is to calculate torque - refer to equation (8).

$$T = \frac{3}{4}P(I_Q\varphi_D - I_D\varphi_Q) \quad (8)$$

2.5. Subtractor

In this work, there two subtractor modules: for comparison actual stator flux and torque values with their references. Both the substractors are conducted in two's complement fixed-point arithmetic. The stator flux comparison module is subtractor 17-bit [4.13], with its output is 18-bit [5.13]. The torque comparison module is subtractor 26-bit [6.20], with its output is 27-bit [7.20].

2.6. Hysteresis Controller

Both the stator flux and torque controllers are responsible to generate the appropriate error status of the stator flux and torque. Both controllers are the 2-level and 3-level hysteresis controllers respectively. So, the output of the both controller are represented in 1- and 2-bit respectively.

2.7. Switching Table

The module is used to select the voltage vectors depending on stator flux error, torque error and the sector. The module is implemented by using the look-up table refer to [1].

2.8. Dead-time

A dead-time is required to avoid short circuit. In this work, the dead-time 2μs is constructed by using the pair of 16-bits counter and comparator.

3. Results and Discussion

The experiments were conducted on Altera DE2 Board, and consume 2093 logic elements for the implementation. The tests have taken place only during certain periods of motor's steady state and the results were observed on the oscilloscope. Thereafter, the results were compared to the validated MATLAB/Simulink simulation results.

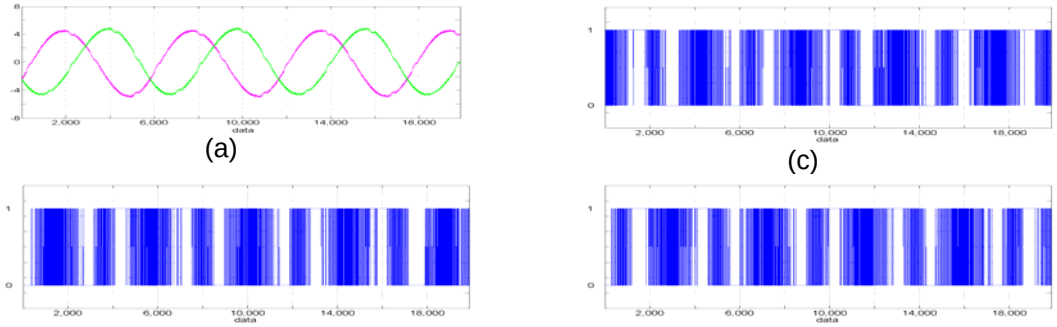


Figure 5. The inputs test. (a) the stator currents I_a & I_b; (b) S_a; (c) S_b; (d) S_c.

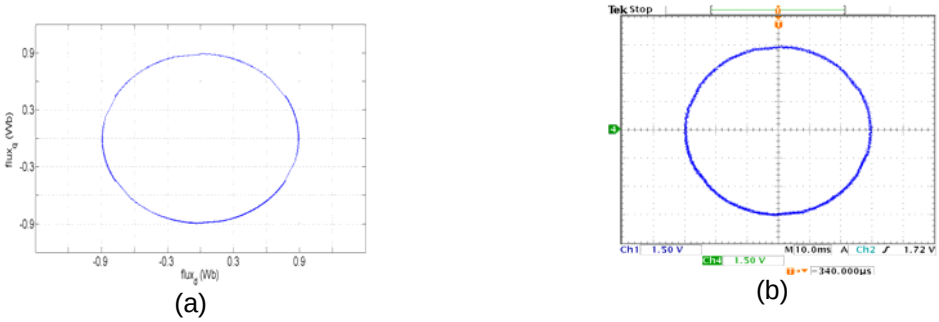


Figure 6. The comparison between MATLAB/Simulink simulation and the experimental result for flux locus. (a) MATLAB/Simulink simulation, (b) FPGA based experimental result

Figure 5 presents the input test, while Figure 6 shows one of the results of the comparisons between MATLAB/Simulink simulations and the experimental. The results shown were matched well with the simulation in MATLAB, which was carried out in double-precision computation.

All units in the system have been designed in fully generic VHDL code, which is independent of the target implementation technology; it does not need third parties and special FPGAs. It is also important to be known that the experimental outputs are displayed through a 12-bit DAC. So, all outputs are truncated within 12-bits. Therefore, it is fair for accepting that the outputs are not perfectly the same as the simulation outputs. The results have proved that the proposed FPGA implementation of the DTC was suitable. Because most of the DTC research solutions have limitation on the performance of the implementation of the torque and flux estimator, obviously this contribution has been eagerly awaited by researchers to support, enable and take forward their improvement on the DTC.

4. Conclusion

This paper has explained how to implement the DTC based on FPGA. The proposed model has conducted by using two's-complement fixed-point arithmetic approaches, in register-transfer-level (RTL) VHDL abstraction for minimizing calculation errors. The choice of word sizes, the binary format and the sampling time used are very important in order to achieve a good implementation of the DTC. This way is simplest and most effective technique, and therefore consumes minimum hardware resource usage. The work has proved that FPGA is suitable for the realization of a high-performance DTC implementation due to its high processing speed, which cannot be obtained by any DSP application. This contribution is a big leap forward

for improvements on the DTC, and it has been eagerly awaited by researchers to support, enable and take forward their DTC improvements.

Acknowledgment

The authors would like to thank Universiti Teknologi Malaysia for providing the funding for the research.

Reference

[1]. I. Takahashi and T. Noguchi. A New Quick-Response and High-Efficiency Control Strategy of an Induction Motor. *IEEE Transactions on Industry Applications*. 1986; IA-22(5): 820-827.
[2]. A. Jidin, N.R.N. Idris, A.H.M. Yatim, A.Z. Jidin, and T. Sutikno. *Torque Ripple Minimization in Dtc Induction Motor Drive Using Constant Frequency Torque Controller*. 2010 International Conference on Electrical Machines and Systems (ICEMS). 2010: 919-924.
[3]. N.R.N. Idris and A.H.M. Yatim. Direct Torque Control of Induction Machines with Constant Switching Frequency and Reduced Torque Ripple. *IEEE Transactions on Industrial Electronics*. 2004; 51(4): 758-767.
[4]. A. Tripathi, A.M. Khambadkone, and S.K. Panda. Torque Ripple Analysis and Dynamic Performance of a Space Vector Modulation Based Control Method for Ac-Drives. *IEEE Transactions on Power Electronics*. 2005; 20(2): 485-492.
[5]. D. Casadei, G. Serra, and K. Tani. Implementation of a Direct Control Algorithm for Induction Motors Based on Discrete Space Vector Modulation. *IEEE Transactions on Power Electronics*. 2000; 15(4): 769-777.
[6]. C. Lascu, I. Boldea, and F. Blaabjerg. A Modified Direct Torque Control for Induction Motor Sensorless Drive. *IEEE Transactions on Industry Applications*. 2000; 36(1): 122-130.
[7]. T.G. Habetler, F. Profumo, M. Pastorelli, and L.M. Tolbert. Direct Torque Control of Induction Machines Using Space Vector Modulation. IEEE Transactions on Industry Applications. 1992; 28(5): 1045-1053.
[8]. T. Noguchi, M. Yamamoto, S. Kondo, and I. Takahashi. Enlarging Switching Frequency in Direct Torque-Controlled Inverter by Means of Dithering. IEEE Transactions on Industry Applications. 1999; 35(6): 1358-1366.
[9]. J. Beerten, J. Verveckken, and J. Driesen. Predictive Direct Torque Control for Flux and Torque Ripple Reduction. IEEE Transactions on Industrial Electronics. 2010; 57(1): 404-412.
[10]. T. Geyer. Computationally Efficient Model Predictive Direct Torque Control. *IEEE Transactions on Power Electronics*. 2011; 26(10): 2804-2816.
[11]. G. Papafotiou, J. Kley, K.G. Papadopoulos, P. Bohren, and M. Morari. Model Predictive Direct Torque Control-Part II: Implementation and Experimental Evaluation. *IEEE Transactions on Industrial Electronics*. 2009; 56(6): 1906-1915.
[12]. A. Jidin, N.R.N. Idris, A.H. Yatim, T. Sutikno, and M. Elbuluk. An Optimized Switching Strategy for Quick Dynamic Torque Control in DTC Hysteresis-Based Induction Machines. IEEE Transactions on Industrial Electronics. 2011; 58(8): 3391 - 3400.
[13]. N.R.N. Idris, T. Chuen Ling, and M.E. Elbuluk. A New Torque and Flux Controller for Direct Torque Control of Induction Machines. IEEE Transactions on Industry Applications. 2006; 42(6): 1358-1366.
[14]. A. Jidin, N.R.N. Idris, A.H.M. Yatim, T. Sutikno, and M.E. Elbuluk. Simple Dynamic Overmodulation Strategy for Fast Torque Control in Dtc of Induction Machines with Constant-Switching-Frequency Controller. IEEE Transactions on Industry Applications. 2011; 47(5): 2283-2291.
[15]. E. Monmasson and M.N. Cirstea. FPGA Design Methodology for Industrial Control Systems: A Review. *IEEE Transactions on Industrial Electronics*. 2007; 54(4): 1824-1842.
[16]. S. Ferreira, F. Haffner, L.F. Pereira, and F. Moraes. *Design and Prototyping of Direct Torque Control of Induction Motors in FPGAs, in Integrated Circuits and Systems Design*. Proceedings. 16th Symposium on SBCCI. 2003: 105-110.
[17]. M. Cirstea, A. Dinu, M. McCormick, and J.G. Khor. Neural and Fuzzy Logic Control of Drives and Power Systems. Neural and Fuzzy Logic Control of Drives and Power Systems, ed., Newnes, Oxford. 2002: 294-304.
[18]. T. Sutikno, A. Jidin, and N.R.N. Idris. New Approach FPGA-Based Implementation of Discontinuous SVPWM. *Turk J Elec Eng & Comp Sci*. 2010; 18(4): 499-514.