LabVIEW FPGA Implementation Of a PID Controller For D.C. Motor Speed Control

Fakhrulddin H. Ali
Computer Engineering Dept.
College of Engineering
University of Mosul
Mosul-Iraq
fhali310@yahoo.com

Mohammed Mahmood Hussein
Computer Engineering Dept.
College of Engineering
University of Mosul
Mosul-Iraq
mohmah86@gmail.com

Sinan M.B. Ismael
Computer and Information Eng. Dept.
College of Electronics Engineering
University of Mosul
Mosul-Iraq
eng_sinan85@yahoo.com

***Abstract*— This Paper presents a novel hardware design methodology of digital control systems. For this, instead of synthesizing the control system using Very high speed integration circuit Hardware Description Language (VHDL), LabVIEW FPGA module from National Instrument (NI) is used to design the whole system that include analog capture circuit to take out the analog signals (set point and process variable) from the real world, PID controller module, and PWM signal generator module to drive the motor. The physical implementation of the digital system is based on Spartan-3E FPGA from Xilinx. Simulation studies of speed control of a D.C. motor are conducted and the effect of a sudden change in reference speed and load are also included.**

***Keywords*— PID Controller, LabVIEW FPGA, Speed Control, Spartan-3E, PWM.**

I. INTRODUCTION

The development of PID (Proportional-Integral-Derivative) control theories has already 60 years so far, PID control has been one of the control system design method of the longest history. However, this method is still extensively used now [1, 2]. PID-controller and its modifications are the most common controllers in the industry. It is robust and simple to design, its operation is well known, it has a good noise tolerance, it is inexpensive and it is commercially available [2].

The design and development of PID controller is habitually supported by simulation tools Matlab simulink and LabVIEW control design and simulation toolkit, for instance, once the controller is modeled its physical implementation can be directly carried out in software such as microprocessor, and microcontroller [3, 4] or hardware such as FPGA[5].

The FPGA is a superior alternative to mask programmed ASICs. FPGAs avoid the high initial cost, the lengthy development cycles, and the inherent inflexibility of conventional ASICs. Also, FPGA programmability permits design upgrades in the field with no hardware replacement necessary, an impossibility with ASICs[6].

The FPGA based system allows designers to create digital designs, test them, make modification very easily , and reduce development time greatly [7]. Digital designs can be modeled using hardware description languages like Very high speed integrated circuit Hardware Description Language (VHDL) and verified by simulation.

In this paper, simulation studies of a PID controller for D.C. motor speed control are conducted. Then, the hardware implementation of a PID module (The PID module is composed of several sub-modules including ADC module, a PWM generation module, and a PID calculation module) is designed using LabVIEW FPGA module and downloaded on the Xilinx Spartan-3E board.

The National Instruments LabVIEW FPGA Module uses LabVIEW Embedded technology to extend LabVIEW graphical development to target FPGAs on NI reconfigurable I/O (RIO) hardware or Xilinx Spartan-3E board. With the LabVIEW FPGA Module, users can: create custom hardware without VHDL coding or board design, Execute multiple tasks simultaneously and deterministically, and solve many applications, including unique timing and triggering routines, ultra high-speed control, interfacing to digital protocols, digital signal processing (DSP), and any other application requiring high-speed hardware reliability and tight determinism.

II. PID CONTROLLER

PID control, shown in fig. 1, is one of the earlier control strategies . Its early implementation was in pneumatic devices, followed by vacuum and solid state analog electronics, before arriving at today's digital implementation via microprocessors or FPGA. It has a simple control structure which was understood by plant operators which they found relatively

easy to tune. Since many control systems using PID control have proved its satisfactory performance, it still has a wide range of applications in industrial control [8] and it has been an active research topic for many years.

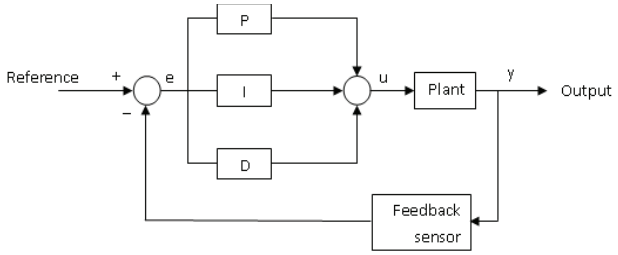

Figure 1. Closed loop PID Controller

The general transfer function of the PID controller looks like the following:

$$u = k_p e + k_i \int e\,dt + k_d \frac{de}{dt} \quad (1)$$

- K_p = Proportional gain
- K_i = Integral gain
- K_d = Derivative gain

The variable (e) represents the tracking error This error signal is sent to the PID controller, and the controller computes both the derivative and the integral of the error signal. This signal (u) is sent to the plant, and the new output (y) will be obtained. This new output (y) will be sent back to the sensor again to find the new error signal (e). The controller takes this new error signal and computes its derivative and integral again, this process goes on and on until the error signal (e) equals zero [9].

III. HARDWARE DESIGN AND IMPLEMENTATION

The hardware implementation of the controller has been done by LabVIEW FPGA and downloaded into XC3S500E FPGA from Xilinx. The Spartan®-3E family of Field-Programmable Gate Arrays (FPGAs) is specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications. The five-member family offers densities ranging from 100,000 to 1.6 million system gates [6]. The XC3S500E FPGA has 4,656 slices, almost 10,476 logic cells, twenty 18x18 hardware multipliers, as well as twenty 18Kbits modules of dedicated dual-port RAM. The block diagram of the whole system is illustrated in fig. 2.

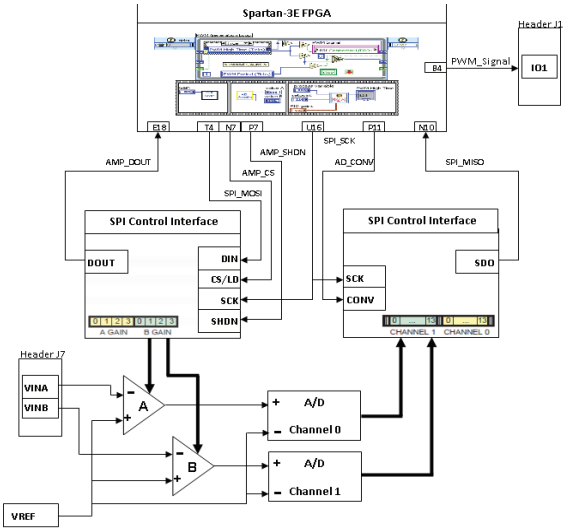

Figure 2. Block Diagram of the complete system implemented on Spartan-3E kit

A. Analog capture circuit

The connection of a digital controller to a real world requires ADC (Analogue to Digital Converter). The Spartan-3E board includes a two-channel analog capture circuit, consisting of a programmable scaling pre-amplifier and an ADC. Both are serially programmed or controlled by the FPGA [10]. The analog capture circuit converts the analog voltage on VINA or VINB to a 14-bit digital representation, D[13:0], as expressed by Eq. 2.

$$D[13to0] = GAIN \times \frac{(V_{IN} - 1.65V)}{1.25V} \times 8192 \quad (2)$$

The Programmable Pre-Amplifier provides two independent inverting amplifiers with a programmable gain. The purpose of the amplifier is to scale the incoming voltage on VINA or VINB so that it maximizes the conversion range of the DAC, namely 1.65 ± 1.25V. The GAIN is the current setting loaded into the programmable pre-amplifier. The various allowable settings for GAIN and allowable voltages applied to the VINA and VINB inputs appear in Table I.

TABLE I. PROGRAMMABLE GAIN SETTINGS FOR PRE-AMPLIFIER

Gain	A3	A2	A1	A0	Input voltage range	
	B3	B2	B1	B0	Minimum	Maximum
0	*0*	*0*	*0*	*0*		
-1	*0*	*0*	*0*	*1*	*0.4*	*2.9*
-2	*0*	*0*	*1*	*0*	*1.025*	*2.275*
-5	*0*	*0*	*1*	*1*	*1.4*	*1.9*
-10	*0*	*1*	*0*	*0*	*1.525*	*1.775*
-20	*0*	*1*	*0*	*1*	*1.5875*	*1.7125*
-50	*0*	*1*	*1*	*0*	*1.625*	*1.675*
-100	*0*	*1*	*1*	*1*	*1.6375*	*1.6625*

The reference voltage for the amplifier and the ADC is 1.65V. Consequently, the maximum range of the ADC is ±1.25V, centered around the reference voltage 1.65V.

The FPGA uses a Serial Peripheral Interface (SPI) to communicate with devices on the board. The SPI bus is a full-duplex, synchronous, character-oriented channel employing a simple four-wire interface. Since the SPI bus signals are shared by other devices on the board, it is vital that other devices are disabled when the FPGA communicates with the amplifier or ADC to avoid bus contention. Table II provides the signals and logic values required to disable the devices.

TABLE II. DISABLED DEVICES ON THE SPI BUS

Signal	Disabled Device	Disable Value
SPI_SS_B	*SPI Serial Flash*	*1*
AMP_CS	*Programmable Pre-Amplifier*	*1*
DAC_CS	*Digital-to-Analog Converter (DAC)*	*1*
SF_CE0	*StrataFlash Parallel Flash PROM*	*1*
FPGA_INIT_B	*Platform Flash PROM*	*1*
AD_CONV	*Analog-to-Digital Converter (ADC)*	*0*

The analog capture circuit is design with a flat sequence structure. At the beginning all devices are disabled except Programmable Pre-Amplifier. Then, pre-amplifier is disabled, AD_CONV signal goes High, and ADC simultaneously samples both analog channels.

B. PID (FPGA) Module

The PID block implements a fixed-point PID algorithm for PID applications with high-speed control and/or high channel count on an FPGA target. This Express VI can be used with single-channel or multi-channel configurations. The PID algorithm features control action range and uses an integrator anti-windup calculation to limit the effect of the integral action during transients. The PID algorithm also features bumpless controller output for PID gain changes. Fig. 3 illustrates the block diagram of this module where:

- **Value A** represents the process variable which specifies the value of the wanted variable to be controlled. This value is represented by a fixed-point number with a maximum word length of 16bit.
- **Value B** represents the set point which specifies the value that the **process variable** to attain. This value is represented by a fixed-point number with a maximum word length of 16bit. This input is available only for single-channel configurations, that is, when **number of channels** is 1, The value wired to this input overrides the value of **initial set point** in the configuration dialog box.
- **PID gains** specifies the normalized PID gain parameters.
- **Output range** specifies the allowable range of the output.
- **PWM High Time** presents the control action (PWM signal high time) that the PID algorithm calculates.

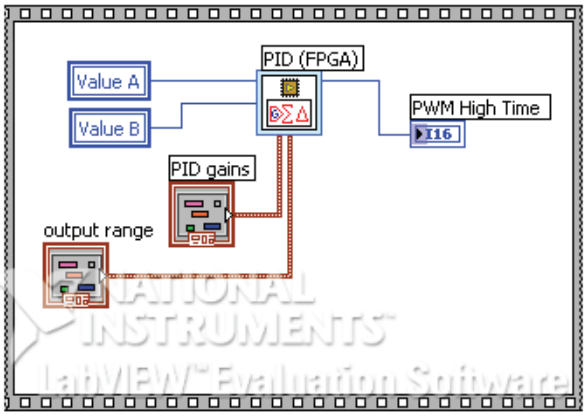

Figure 3. PID(FPGA) module block diagram

This Express VI represents the PID proportional, integral, and derivative gains as signed fixed-point numbers with word length 16bit and integer word length 8bit. Given the Proportional gain (K_c), Integral time (T_i), and Derivative time (T_d), respectively, this Express VI normalizes these gains according to the following formulas [11]:

$$K_p = K_c \tag{3}$$

$$K_i = \frac{K_c \times T_s}{T_i \times 60} \tag{4}$$

$$K_d = \frac{K_c \times T_d \times 60}{T_s} \tag{5}$$

where T_s is the sampling time T_s (s) at which the PID loop runs.

This express VI calculates the output, u(k), according to the following equation:

$$u(k) = u_p(k) + u_i(k) + u_d(k) \tag{6}$$

where

$$u_p(k) = K_p\, e(k) \tag{7}$$

$$u_i(k) = K_p\, K_i \sum_{i=1}^{k} \left[\frac{e(i) + e(i-1)}{2}\right] \tag{8}$$

$$u_d(k) = -K_p\, K_d\, [PV(k) - PV(k-1)] \tag{9}$$

$$e(k) = setpoint(k) - PV(k) \tag{10}$$

PV(k) = value of process variable on the kth call after initialization.

C. PWM Generator Module

D.C. motor speed can be controlled by varying the power applied to the armature, PWM is very well known method do this task. In PWM technique, power is regulated by applying pulses of variable width by changing the pulse width of the power. With the pulse width small, the average voltage applied onto the motor is low, and the motor's speed is slow. If the width is wide, the average voltage is higher, and therefore motor speed is higher [12, 13].

Fig. 4 shows PWM signal with 50% duty cycle, where T is the signal period, V_H is the signal amplitude. The average output voltage in this case is 0.5 * V_H. In general, the output voltage will be:

$$V_{avg} = \frac{T_H}{T_H + T_L} \times V_H = D \times V_H \quad (11)$$

Where, V_{avg} is the average output voltage and D is a duty cycle. From this equation the average voltage is linearly related to the duty cycle. On the other hand, the period is fixed as illustrated in fig. 5.

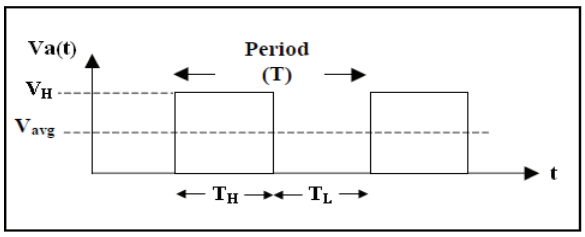

Figure 4. PWM signal with 50% duty cycle

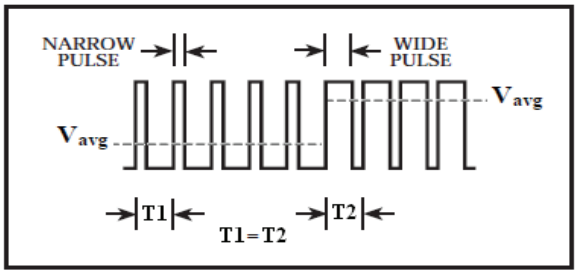

Figure 5. Pulse width determines average voltage with fixed period

Fig. 6 shows LabVIEW block diagram that generate PWM signal with different duty cycles dependent on PID control signal and eduction this signal through one of the FPGA I/O pins.

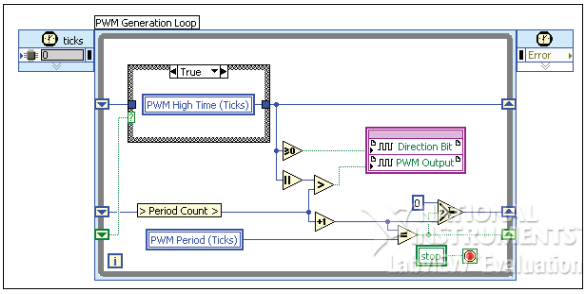

Figure 6. PWM signal generation block diagram

Table III shows FPGA logic resources used to implement the complete control unit represented by analog capture circuit, PID(FPGA) module, PWM generator module.

TABLE III. DEVICE UTILIZATION SUMMARY FOR A CONTROL UNIT

Device Type	Percentage area occupied
Number of Slices	*1473 out of 4656 31%*
Number of Slice Flip Flops	*1683 out of 9312 18%*
Number of 4 input LUTs	*2234 out of 9312 23%*
Number of bonded IOBs	*106 out of 232 45%*
Number of MULT18X18SIOs	*3 out of 20 15%*
Number of GCLKs	*2 out of 24 8%*
Maximum Frequency	*51.509MHz*

IV. SIMULATION RESULTS

Simulation of a PID control action is applied upon the armature of the D.C. motor. The equivalent circuit is illustrated in fig. 7.

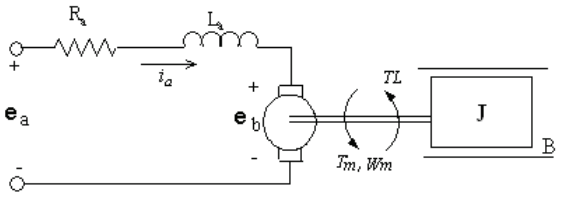

Figure 7. Armature controlled D.C. motor equivalent circuit

Equations 12 through 15 consider that the applied voltage $e_a(t)$ is the cause, Eq. 12 considers that $di_a(t)/dt$ is the immediate effect due to $e_a(t)$, in Eq. 13, $i_a(t)$ causes the torque $T_m(t)$, Eq. 14 defines the back emf (electromotive force); and, finally, in Eq. 15, the torque $T_m(t)$ produces the angular velocity $\Omega_m(t)$. Fig. 8 shows the block diagram representing the D.C. motor system [14].

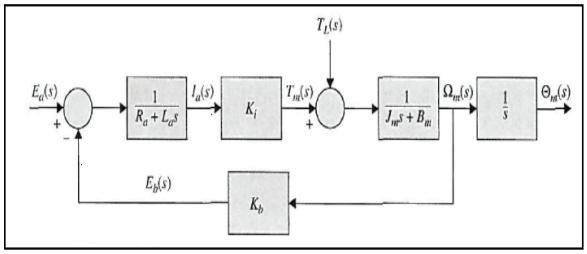

Figure 8. diagram of a D.C. motor system

$$\frac{di_a(t)}{dt} = \frac{1}{L_a} e_a(t) - \frac{R_a}{L_a} i_a(t) - \frac{1}{L_a} e_b(t) \quad (12)$$

$$T_m(t) = K_i \varphi i_a(t) \quad (13)$$

$$e_b(t) = K_b \Omega_m(t) \quad (14)$$

$$\frac{d^2\theta_m(t)}{dt^2} = \frac{1}{J_m} T_L(t) - \frac{B_m}{J_m} \frac{d\theta_m(t)}{dt} \quad (15)$$

The motor parameter values used in this paper are listed in table IV. The PID controller parameters are K_p=7, K_i= 65, and K_d=0.08.

TABLE IV. D.C. MOTOR PARAMETERS

Parameters	descriptions	Measuring units	Value
K_i	*torque constant*	*Nm/A*	*0.5*
R_a	*armature resistance*	*Ω*	*5*
L_a	*armature inductance*	*H*	*0.1*
J_m	*Armature moment of inertia*	*kg-m2*	*0.001*
B	*viscous-friction coefficient*	*N-m/rad/sec*	*Negligible*
K_b	*Speed constant*	*V/rad/sec*	*1.356Ki*

To demonstrate the system performance, a sudden change in reference speed is introduced. The response is illustrated in fig. 9. Initially, the reference speed is 700 rpm, then increased to 1400 rpm, and finally set back to 700 rpm. The control performance parameters for the step response and sudden change of speed reference are summarized in table V.

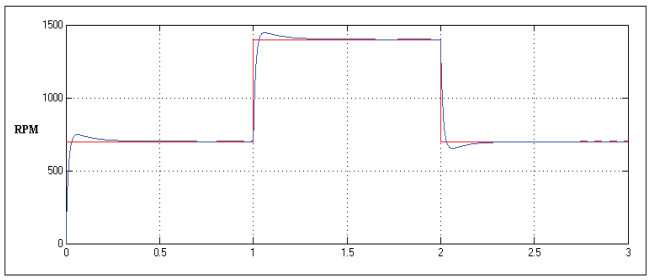

Figure 9. System response to the sudden change of speed reference

TABLE V. RESPONSE PARAMETERS FOR THE SUDDEN CHANGE OF SPEED REFERENCE

Reference (rpm)	OS (%)	tr (sec)	US (%)	tf (sec)	ts (sec)
700 To 1400	*6.56*	*0.02*	——	——	*0.26*
1400 To 700	——	——	*6.59*	*0.02*	*0.26*

Fig 10 and 11 shows the system response when a load is applied and released during a constant motor speed of 700 rpm and 1400 rpm respectively. The load applied is 5 N.m. Table VI and VII summarize the overshoot, drop speed, and rise speed when load is applied and released respectively.

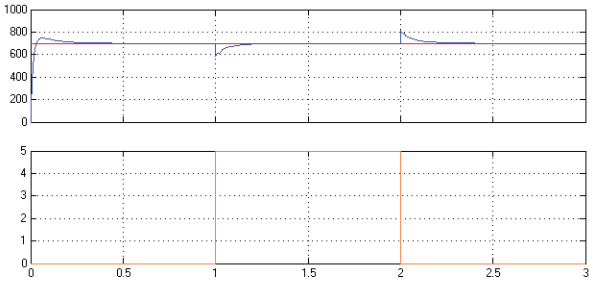

Figure 10. System response to sudden change of load at 700 rpm constant speed

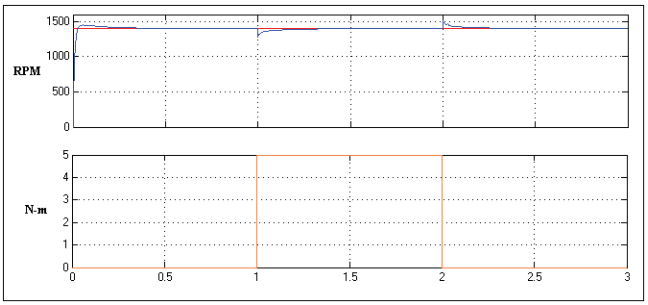

Figure 11. System response to sudden change of load at 1400 rpm constant speed

TABLE VI. RESPONSE PARAMETERS WHEN 5 N-M LOAD APPLIED

Reference (rpm)	Drop (rpm)	tdrop (sec)	OS (%)	tos (sec)	ts (sec)
700	*117.46*	*0.006*	*6.59*	*0.062*	*0.21*
1400	*114.79*	*0.007*	*3.55*	*0.052*	*0.17*

TABLE VII. RESPONSE PARAMETERS WHEN 5 N-M LOAD RELEASED

Reference (rpm)	Rise (rpm)	trise (sec)	ts (sec)
700	*118.26*	*0.006*	*0.21*
1400	*114.81*	*0.007*	*0.16*

Finally, it is assumed that the reference speed and the motor load are changing at the same time. The system response is depicted in fig. 12 and its parameters are summarized in table VIII.

TABLE VIII. RESPONSE PARAMETERS FOR THE SUDDEN CHANGE OF SPEED REFERENCE AND LOAD

Reference (rpm)	Load (N.m)	OS (%)	Drop or Rise (rpm)	tdrop or trise (sec)	ts (sec)
700 To 1400	Zero To 50	*1.77*	*88.06*	*0.0004*	*0.16*
1400 To 700	50 To Zero	*1.84*	*88.11*	*0.0004*	*0.16*

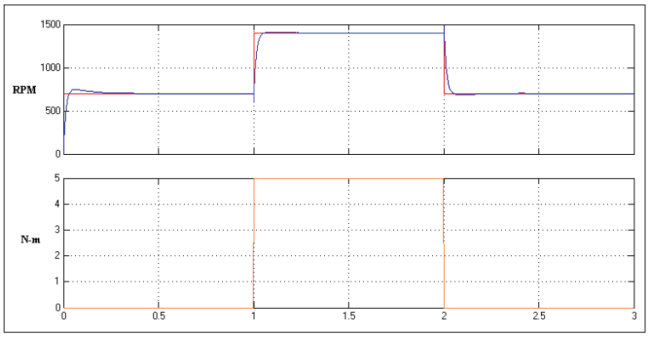

Figure 12. System response to the sudden change of speed reference and load

V. CONCLUSION

In this paper, a hardware digital controller is designed using LabVIEW program. LabVIEW is a graphical programming language used not only for data acquisition and measurement but also for designing digital systems on FPGA targets without prior knowledge of hardware description language. The block diagram of a LabVIEW FPGA VI can represent the parallelism and timing of embedded systems much better than text-based languages. At present, the system maximum operating frequency is (51.509MHz) this could take advantage of the high speed achievable using hardware. The design of PID controller based on FPGA allows their future utilization to use this controller for other application. Simulation results shows that the controller is provide stability and minimum overshoot in response to sudden change in reference speed and load.

REFERENCES

[1] Seung-Min Baek Tae-Yong Kuc , " An adaptive PID learning control of DC motors ", International Conference on Systems, Man, and Cybernetics, Orlando, FL, page(s): 2877 - 2882 vol.3, Oct 1997.

[2] Michael A. Johnson and Mohammad H.Moradi, "PID Control: New Identification and Design Methods", Springer-Verlag London , ISBN-10: 1-85233-702-8, 2005.

[3] Guoshing Huang Shuocheng Lee, "PC-based PID speed control in DC motor", International Conference on Audio, Language and Image Processing, Shanghai, page(s): 400 – 407, July 2008.

[4] S. Kamalasadan, A. Hande, "A PID Controller for Real-Time DC Motor Speed Control using the C505C Microcontroller", In Proceedings of the 17th International Conference of Computer Applications in Industry and Engineering (CAINE'04),pp 34-39, November 2004.

[5] A. Trimeche, A. Sakly, A. Mtibaa, M. Benrejeb, "PID control implementation using FPGA technology", 3rd International Design and Test Workshop IDT, page(s): 341 – 344, Dec. 2008, IEEE.

[6] "Spartan-3E FPGA Family: Complete Data Sheet", DS312 April 18, 2008, Xilinx.

[7] S. Singh, S. K. Rattan, "Implementation of a Fuzzy Logic Controller on an FPGA using VHDL", 22nd International Conference of the North American Fuzzy Information Processing Society NAFIPS, Page(s): 110 - 115, July 2003 IEEE.

[8] Dingyu Xue, YangQuan Chen, and Derek P. Atherton, "Linear Feedback Control: Analysis and Design with MATLAB", the Society for Industrial and Applied Mathematics, Philadelphia, ISBN 978-0-898716-38-2, 2007.

[9] Antonio Visioli, "Practical PID Control", Springer-Verlag London Limited, ISBN-10: 1-84628-585-2, 2006.

[10] "Spartan-3E Starter Kit Board User Guide", UG230 (v1.0) 9 March 2006, Xilinx.

[11] National Instruments Co., "LabVIEW: PID and Fuzzy Logic Toolkit User Manual", Part Number 372192D-01, June 2009.

[12] Baldor Electric Company “Servo Control Facts” a handbook explaining the basics of motion 2000.

[13] P. C. Sen, "Principles of Electric Machines and Power Electronics", Second Edition. John Wiley & Sons Inc., 1997.

[14] Benjamin C. Kuo and Farid Golnaraghi, "Automatic Control Systems", John Wiley and Sons Inc. U.S.A. , ISBN 978-0-471-13476-3, 8th edition , Sep. 2002.