DC Motor Speed Control using LabVIEW FPGA Modeling, Control Algorithm Simulation & Implementation

Dinesh O. Dange[1] Rajan Mevekari[2] Dipankar D. Khartad[3]
[1,2]PG Student [3]Assistant Professor
[1,2,3]Department of Electronics
[1,2,3]Walchand College of Engineering Sangli, Maharashtra, India

Abstract— This paper represents more reliable FPGA hardware implementation of dc motor speed control system. In this a control design & simulation module is used to simulate both DC motor model & its control system. Also it can simulate the FPGA clock timings. Novel Hardware & software co-design approach for design of PID control algorithm, stimulus (PWM) generator, feedback sensor (Quadrature encoder) interface is presented. Flexible FPGA module allows easily reuse of code used in development process during actual hardware implementation. Unlike conventional hardware designing which requires VHDL expertise, the developer need not to know in-between processes of ready to download final bit-file generation. Finally the bit file will be downloaded in CompactRIO-9076 (reconfigurable platform) assembled with C series analog & digital I/O modules which interact with signal from DC motor plant.

Key words: PID controller; PWM; Quadrature encoder interface; CompactRIO-9076; C series analog & digital I/O modules

I. INTRODUCTION

In recent years FPGA's are viewed as reliable solution to electric motor control systems. They are prevailing because they can provide more reliable hardware implementation directly on silicon chip. So there is need of real time simulation for developing an electric drive & modeling, evaluating the performance of an electric motor. It is really important to model electric motor that best represent its operation under different conditions, which could be used in simulation stage [1]. The conventional simulations carried are not running in real time. One has to work in different environment; as result there is great difference between actual & interpreted timing parameters. Also developing system takes lots of iteration hence time consuming. LabVIEW FPGA is unique graphical programming platform for both simulation and implementation. It is flexible since it allows reusability of code that we have used in development process. The target used is Xilinx FPGA [2].

User need not to know VHDL programming to configure it. The in between processes are carried out in background, we get the final downloadable bit-files. The use of one platform for real time simulation of electric machines allows predicting the good industrial performance under operational conditions and disturbances, saving money and time on industrial applications [2] [3].

In this paper dc motor speed control system is implemented. The common closed loop control system is shown in fig.1. In this desired reference (speed) is set by user, error is calculated by subtracting measured output (actual speed) from it. According to error, controller (PID controller) produces a necessary control action that acts as stimulus to the plant (DC motor) to minimize error. The response of plant is measured again and this continues until zero error is reached i.e. desired reference (speed) is achieved.

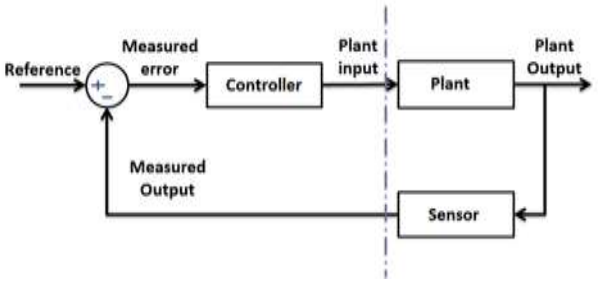

Fig. 1: DC motor speed control system

The dashed line separates controlling system from controlled system. The controlling system consist PID controller, PWM generator, H-bridge driver, Quadrature Encoder Interface (QEI) whereas controlled system is DC motor with quadrature encoder.

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]. We implemented discrete PID controller in LabVIEW FPGA. We have used the PWM signal generator IP (Intellectual Property), which receives duty cycle input from PID controller. PWM signal is given to bidirectional H-bridge motor driver, which efficiently drives motor with required voltage & current, reducing average power delivered to load. Optical Quadrature incremental encoder is used to read actual speed of motor. Optimized Quadrature Encoder Interface (QEI) is implemented in FPGA to interpret accurate position, speed & direction of DC motor. Based on feedback received, PID controller produce control action until desired speed is reached.

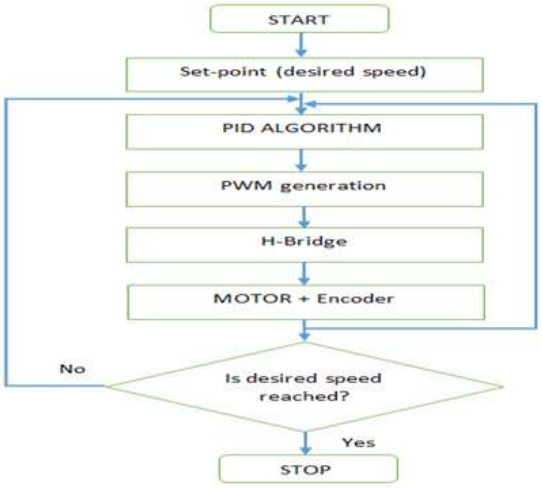

Fig. 2: Flow chart

All rights reserved by www.ijsrd.com

In stage of simulation DC motor can be modeled mathematically by using any of the Math-script node, Formula node or multisim node. The spice based model of DC motor is included using multisim node in control design & simulation loop. In stage of implementation, actual DC motor is interfaced.

II. BRUSHED DC MOTOR MODELLING

The electrical model for a brushed direct current (DC) motor is shown schematically in Fig.3. The motor terminal voltage is represented as $v_{m;}$ R and L represent the resistance and inductance of the motor armature circuit, while i represent the armature current. K_e represents the motor back-EMF constant, which determines the ratio between the shaft velocity, dθ/dt, and the back e.m.f voltage, v_e.

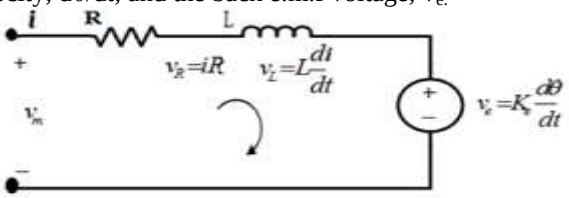

Fig. 3: Electrical model of DC motor

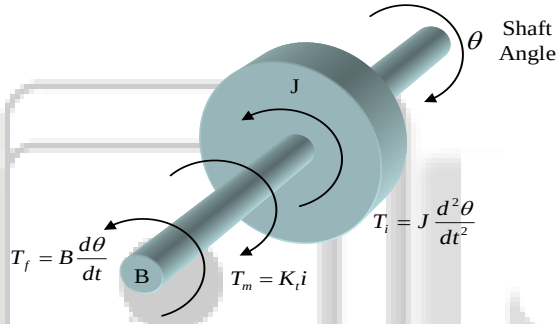

Fig. 4: Mechanical model of DC motor

Transfer Function between Motor Terminal Voltage and Angular Position. The time domain differential equations are summarized as follows.

$$v_m = Ri + L\frac{di}{dt} + K_e\frac{d\theta}{dt} \quad (1)$$

$$0 = K_t - J\frac{d^2\theta}{dt^2} + B\frac{d\theta}{dt} \quad (2)$$

We rearrange to move the highest order derivative term for the current and position variables to the left side of the equation as follows. This form of the equations provides a basis for simulation using the LabVIEW Simulation Module.

$$\frac{di}{dt} = -\frac{R}{L}i - \frac{K_e}{L}\frac{d\theta}{dt} + \frac{v_m}{L} \quad (3)$$

$$\frac{d^2\theta}{dt^2} = \frac{K_t}{J}i - \frac{B}{J}\frac{d\theta}{dt} \quad (4)$$

implified transfer function is given by:

$$\frac{\Omega_m}{V_m} = \frac{K_e}{J*R_i*s+K_e^2} \quad (5)$$

Specification sheet			
Symbol	**Description**	**Value**	**Unit**
R_m	Motor armature resistance	3.3	Ω
Kt	Motor torque constant	0.028	Nm
K_m	Motor back-emf constant(same as K_t in SI units)	0.028	V/(rad/s)
J_m	Moment of inertia of	9.60*10^{-6}	KgSqm
	motor rotor		
	Max continuous torque	0.10	Nm
	Max power rating	20	W
	Max continuous current	1	1
B	Friction Torque Constant	1.8e-6	N-m-s

a. DC motor Specification sheet is referred for obtaining parameters

Table 1: Sample Dc Motor Specification Sheet

III. CONTROL DESIGN & SIMULATION MODULE

Control & simulation loop executes the simulation diagram until it reaches the simulation final time or until the Halt Simulation function stops the execution programmatically. It includes a built-in ODE solver for handling integrals and derivative terms. The Summation, Gain, Integrator and Transfer Function blocks can also be found in the Control Design and Simulation palette under Simulation>>Signal Arithmetic and Simulation>>Continuous Linear Systems. Also we can simulate nonlinearities & disturbances prior to implementation [4]. Different ways to add motor models in loop:

- Using formula node: Equations 3 & 4 directly represent model

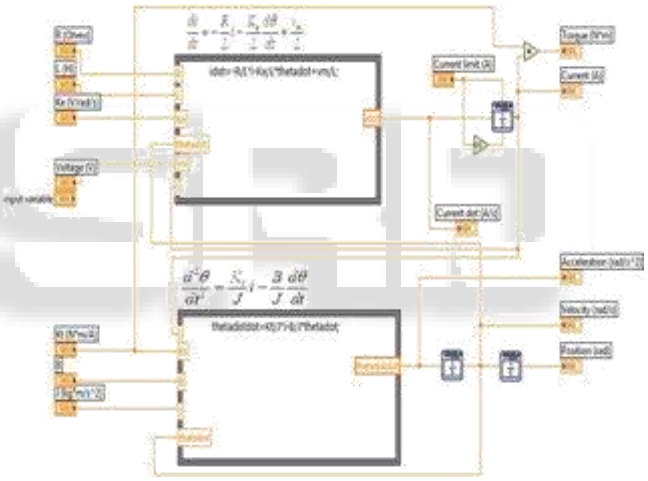

Fig. 5: Formula node

- Using Multisim node: Spice model are included which represent practical DC motor, H-bridge driver, Quadrature encoder feedback sensor shown in fig.9.

IV. CONTROL SYSTEM DESIGN

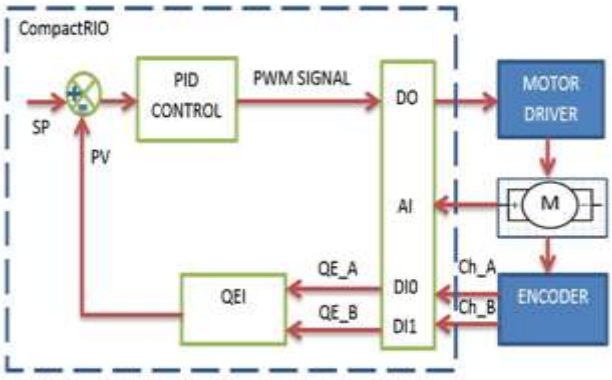

Fig. 6: Block diagram of control system

All rights reserved by www.ijsrd.com

- CompactRIO: cRIO-9076 chassis integrated Xilinx Spartan-6 FPGA, 40MHz.
- C series I/O module: NI-9401 digital I/O module, NI-9221 analog I/P module.
- Motor driver: MD10C (cytron) 24v 10A.
- Motor: 24v, 100w Brushed DC motor with Encoder.
- Encoder: Incremental (optical) encoder with 2000 PPR.
- Current Sensor: ACS-712, senses up-to 20A.(used just for monitoring purpose)

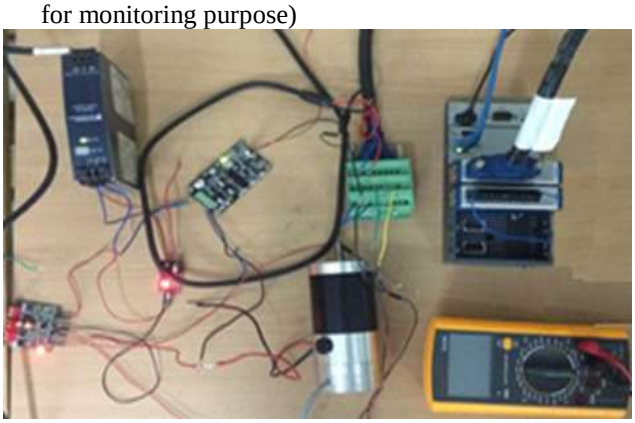

Fig. 7: Experimental setup

A. Discrete PID Controller

As the floating point algorithm consume unfeasibly large resources(CLB/Slices, Multipliers & DSP slices, embedded memory blocks) on limited Silicon FPGA chip .To implement the controller on an FPGA the continuous time model for the controller had to be discretized. Discretization used to move any floating point algorithm to a fixed point representation for execution on an FPGA. Fixed point (FXP) representation removes the dynamic shifting involved in floating point representation hence optimizes the resource utilization.

To make the PID controller design reusable (IP-intellectual property), the controller was designed to operate at a rate of 40 MHz (the default clock speed for most FPGAs). Any code executing at that rate must compute its outputs within a single clock period of the FPGA. Such a fast controller was not necessary for the brushed DC motor application presented in the paper (especially when the brushed DC motor was loaded by a large inertial disk); however, a high speed controller with fast PID loop rate offers improved disturbance rejection, and The continuous time PID controller equation was discretized by using the backward Euler approximation:

$$u(t) = u_p(t) + u_i(t) + u_d(t) \quad (6)$$

Where,

$$u_p(t) = K_p e(t)dt$$

$$u_i(t) = K_i \int_0^t e(t)dt$$

$$u_d(t) = K_d \frac{d}{dt} e(t)$$

Discretizing:

$$u(t_n) = u_i(t_{n-1}) + K_p e(t_n) + K_i T_s e(t_n) + K_d \left(\frac{e(t_n) - e(t_{n-1})}{T_s} \right) \quad (7)$$

As word-length & integer-lengths of fixed point math can be configured the discrete time PID controller logic was further improved to include anti-windup. By enabling saturation in the fixed point configuration of the high throughput math functions, the integral term and the controller output, were limited to a range of -1024 to 1024. This range was chosen since it is a multiple of 2, and the PWM generator block which is fed by the controller expected a value in the range 0 to 2000. It is observed that derivative action is hardly needed in our system.

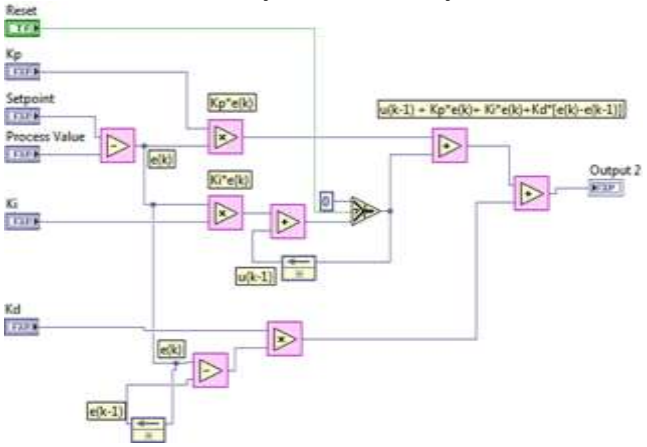

Fig. 8: Discrete PID implementation

To reduce the propagation delay of the high throughput math functions:

- K_p was limited to 6 bit word-length & integer-length, and could be changed in steps of 0.125.
- For a 40 MHz clock K_i could be changed from 0 in steps of approximately 2.384.

B. Importing Spice Dc Motor Model

Amongst different methods of dc motor modeling (mentioned in section III) multisim spice based models are more accurate to represent DC motor. The spice based model includes:

- The analog drive circuitry of dc motor. It includes common H bridge topology for bidirectional current flow control and hence both Clock-Wise (CW) & Counter Clock-Wise (CCW) rotation control of motor.
- Quadrature encoder as feed-back sensor.

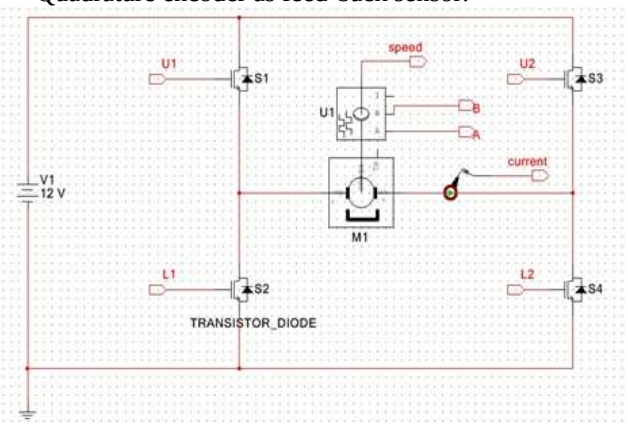

Fig. 9: DC motor, H bridge, Quadrature Encoder

Through hierarchical connector shown in Fig.8, multisim I/O are accessed in LabVIEW.

C. PWM Signal Generation

It is well known fact that PWM signal are used to save the average power delivered to motor. It is achieved by switching four switches of the H-bridge turned on and off in

All rights reserved by www.ijsrd.com

every cycle, with the diagonal switches pairs of the H-bridge driven together. Most common 20 kHz high speed PWM signal is generated by PWM LOOP (IP) implemented in FPGA. Corresponding calculations:

$$\text{PWM}_{\text{period(ticks)}} = \frac{\text{FPGA clock rate(Hz)}}{\text{DESIRED PWM rate(Hz)}} \quad (8)$$

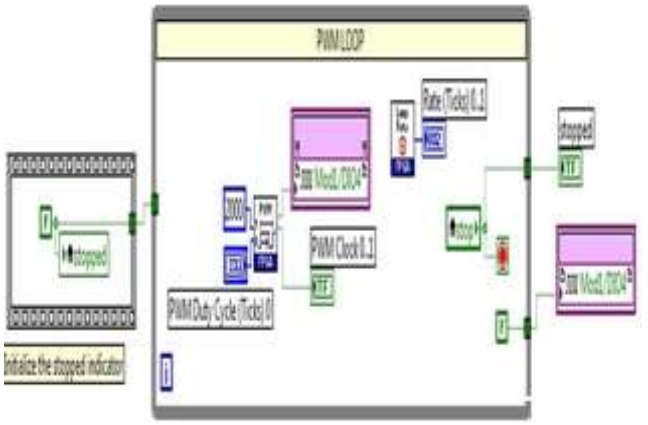

Fig. 10: PWM signal generation

Prior to whole system implementation, PWM signal used to test the motor driver (MD10C-24v, 20A) we have brought. The responses observed with motor connected to driver are shown in fig.11 & 12.

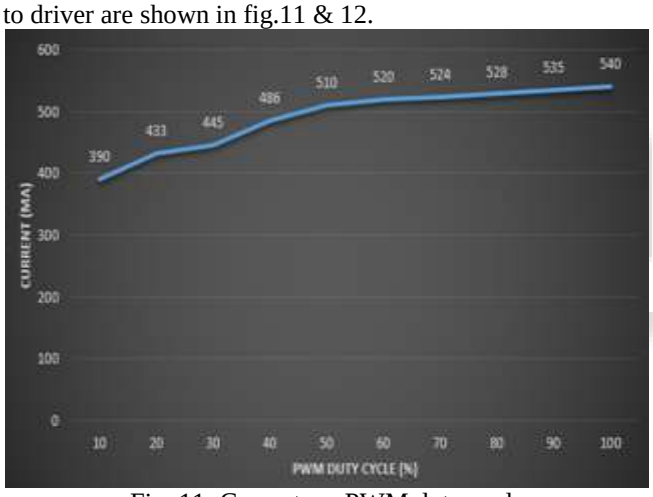

Fig. 11: Current vs. PWM duty cycle

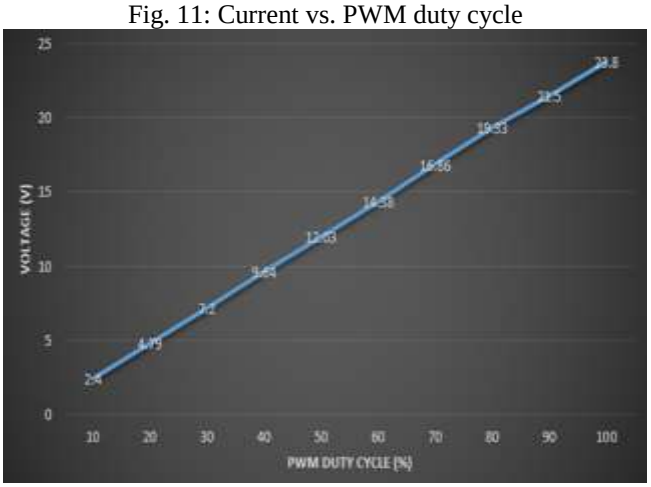

Fig. 12: Voltage vs. PWM duty cycle

As expected the motor draws almost constant current (ăpprox.600mA) without load and speed of motor increases as voltage supplied to motor increases with duty cycle of PWM signal.

D. Quadrature Encoder Interface

The purpose of Quadrature Encoder Interface (QEI) is to allow user to connect the encoder feedback sensor to the FPGA. It consists of Filter, Quadrature Decoder & UP/DOWN counter [5].

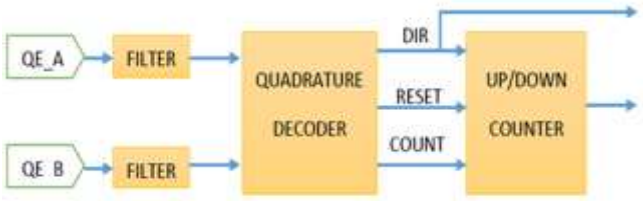

Fig. 13: Quadrature Encoder Interface (QEI)

1) Filter

The signal coming through Ch_A & Ch_B are square wave with 90 degree phase shift between them. In practical scenario these signals may get infected by noise, which leads to wrong interpretation of position & speed of system under control. These spurious signals contain noise components which are more like a spike. That justifies the requirement of a digital filter to avoid abrupt change in signal levels due to noise. The corresponding LabVIEW FPGA implementation of digital filter is also shown in fig.14. The logic here is that if the logic levels on channels QE_A & QE_B sustained for predefined 'Filter Period (Ticks)' then only they are considered valid for counting operation, otherwise previous logic levels are considered.

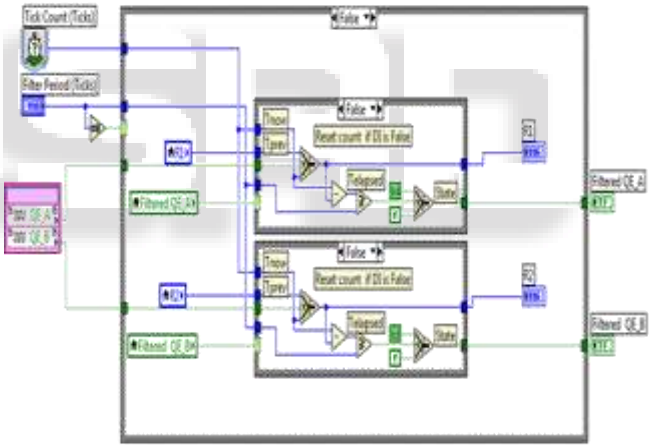

Fig. 14: Filter logic in LabVIEW FPGA

2) Quadrature Decoder & UP/DOWN Counter

The filtered QE_A & QE_B signals available to Quadrature Decode loop. Based on how many of transition points are used for the measurement, there are three different types of evaluation methods. The single evaluation method (×1) uses only one channels rising or falling edges, the double evaluation method(×2) using only one channel, but both its rising and falling edges, and the quadruple evaluation method (×4) uses every edges of the two channels. The logic here implements the quadruple evaluation method. With every rising edge & falling edge of two channels position counter is incremented. Also direction is decided based on whether QE_A is following QE_B (CW) or QE_B is following QE_A (CCW). For velocity estimation proper fixed 'velocity interval (us)' is chosen, and position counts per fixed velocity interval are evaluated. It is used to calculate velocity in real time.

All rights reserved by www.ijsrd.com

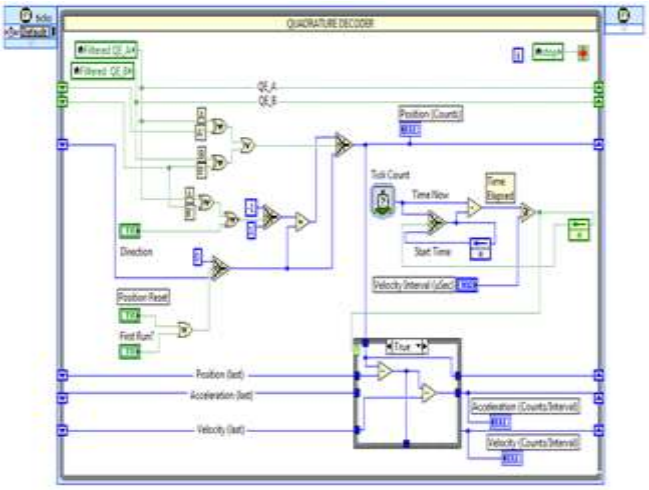

Fig. 15: Quadrature decoder & UP/DOWN counter

3) Velocity Estimation

Choosing the correct measurement time i.e. fixed velocity interval is essential. By allowing more time between updates you obtain more averaging on the velocity and acceleration values and therefore smoother results. A larger time interval results in less noise on the velocity and acceleration calculations. Also, using an encoder with more counts per revolution results in better accuracy and smoother results on the velocity and acceleration calculations Basically, the longer the measurement time, the better the results are. In exchange for the longer measurement, the speed controller will be slower and this will decrease the quality of the drive. So there is trade-off between speed & accuracy [6].

$$\mathrm{RPM} = \left(\frac{\text{counts}}{\text{interval}}\right).\mathrm{RPM}_{\mathrm{CONSTANT}} \qquad (9)$$

Where,

$$\mathrm{RPM}_{\mathrm{CONSTANT}} = \frac{1}{\mathrm{PPR}}.\frac{10^6}{\text{fixed velocity interval(us)}}.\frac{60\mathrm{s}}{1\mathrm{min}}$$

We have used encoder with 2000ppr, and fixed velocity interval is selected 1000 µs, so $\mathrm{RPM}_{\mathrm{CONSTANT}}$ is equal to 30. Above calculations are carried out in host VI which includes reference of VI running in FPGA. (Default clock rate of on board FPGA is 40MHz).

V. CONCLUSION

Brushed DC motor mathematical models facilitate the simulation of control system. With simultaneous simulation in LabVIEW & Multisim entire system is analyzed & optimized. Additionally performance of quadrature encoder interface (QEI) improved by implementing digital filter in FPGA. Flexible LabVIEW FPGA platform offers reusability of Intellectual Property (IP) during both development & implementation. Hence electric motor & its drive performance are increased.

REFERENCES

[1] Real Time Simulation for DC and AC Motors Based on LabVIEW FPGAs,Pedro Ponce, Luis Ibarra, Arturo Molina,Brian MacCleery.
[2] http://www.ni.com/white-paper/14556/en/
[3] Ali, F.H.; Mahmood, H.M.; Ismael, S.M.B., "LabVIEW FPGA implementation of a PID controller for D.C. motor speed control," Energy, Power and Control (EPC-IQ), 2010 1st International Conference on , vol., no., pp.139,144, Nov. 30 2010-Dec. 2 2010.
[4] http://www.ni.com/labview/cd-sim/
[5] http://www.eetimes.com/document.asp?doc_id=1274595
[6] http://www.ni.com/tutorial/3921/en/
[7] Rajesh Nema, Rajeev Thakur, Ruchi Gupta., "Design & Implementation of FPGA Based On PID Controller," International Journal of Inventive Engineering and Sciences (IJIES), ISSN: 2319–9598,Volume-1, Issue-2, January 2013
[8] Vikas Gupta1, Kavita Khare2 & R. P. Singh2., "Efficient Design and Fpga Implementation of Digital Controller Using Xilinx SysGen®"International Journal of Electronics Engineering, 2(1),2010, pp. 99-102
[9] Seung-Min Baek; Tae-Yong Kuc, "An adaptive PID learning control of DC motors," Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on , vol.3, no., pp.2877,2882 vol.3, 12-15 Oct 1997 doi: 10.1109/ICSMC.1997.635431.
[10] Michael A. Johnson and Mohammad H.Moradi, "PID Control: New Identification and Design Methods", Springer-Verlag London, ISBN-10: 1-85233-702-8, 2005.
[11] 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.
[12] 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.
[13] National Instruments Co., "LabVIEW: PID and Fuzzy Logic Toolkit User Manual", Part Number 372192D-01, June 2009.
[14] Baldor Electric Company "Servo Control Facts" a handbook explaining the basics of motion 2000.

All rights reserved by *www.ijsrd.com*