International Journal of Electrical and Computer Engineering.
ISSN 0974-2190 Volume 2, Number 1 (2010), pp. 71--82

http://www.irphouse.com

Analysis and Implementation of Discrete Time PID Controllers using FPGA

Sreenivasappa B.V. and Dr. Udaykumar R.Y.

Research Scholar, E-mail: sreebms@gmail.com, sreebms@nitk.ac.in
Professor & Head, E-mail: udaykumarry@yahoo.com
Department of EEE, National Institute of Technology Karnataka
Surathkal, Mangalore-575025, India

Abstract

In this paper analysis and implementation of Proportional Integral Derivative (PID) controller using Field Programmable Gate Array (FPGA) is presented. Different types of discrete time PID controllers are analyzed with their frequency response using Matlab tool. The controller algorithm is synthesized, simulated and implemented using Xilinx Spartan3e XC3S100E board with XilinxISE 9.1i as a tool and synthesized and simulated using Altera Cyclone EP1C12Q240C8 with Quartus II 6.1 as a tool. The two results are compared in terms of their power consumption, speed, memory usage, Look Up Tables (LUTs) and number of Multipliers, Adders/Subtractors.

Keywords: Proportional Integral Derivative (PID) controller, Field Programmable Gate Array (FPGA), Backward Difference (BD), Look Up Tables (LUTs).

Introduction

Proportional Integral Derivative (PID) controllers have been widely used over the past five decades due to their simplicity, robustness, effectiveness, and applicability for a broad class of systems. Despite the numerous control design approaches that have appeared in the literature, it is estimated that nowadays PID controllers are still employed in more than 95% of industrial processes [1]. An important feature of this controller is that it does not require a precise analytical model of the system that is

being controlled. For this reason, PID controllers have been widely used in robotics, automation, process control, manufacturing, transportation, and interestingly in real time multi tasking applications [2].

Implementation of digital PID controller has gone through several stages of evolution, from the early mechanical and pneumatic designs to the microprocessor based systems but these systems have the drawback of demanding control requirements of modern power conditioning systems will overload most of the microprocessors and the computing speed limits the use of microprocessor in complex algorithms.

Microprocessors, Microcontrollers and Digital Signal Processors (DSPs) can no longer keep pace with the new generation of applications that requires more flexible and higher performance without increasing cost and resources. Further more the tasks are executed sequentially which takes longer processing time to accomplish the same task in Microcontrollers and DSPs.

Recently, Field Programmable Gate Arrays (FPGA) have becoming alternative solution for the realization of digital control systems. The FPGA based controllers offer advantages such as high speed computation, complex functionality, real time processing capabilities and low power consumption [3]. In this paper we consider discrete time PID controller and is implemented in a dedicated FPGA. Following the standard digital design practices, the controller functionality is described in Very High Speed Integrated Circuit Hardware Description Language (VHDL). Using synthesis tool, the design is then targeted to the FPGA board.

The organization of this paper is as follows: In section II different discrete time PID controllers are reviewed. Implementation of PID controller algorithm using FPGA is explained in section III. The simulation and FPGA implementation results are discussed in section IV.

Discretization of PID Controller

The general form of PID controller given in most of the text book is the standard form.

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \tag{1}$$

where K_p is the Proportional gain, T_i is the Integral time, T_d is the derivative time, $e(t)$ is the error signal and $u(t)$ is the output of the controller.

The ideal parallel form of the PID controller shown in Fig. 1 is represented by a mathematical equation as

$$u(t) = K_p e(t) + K_i\int_0^t e(\tau)d\tau + K_d\frac{de(t)}{dt} \tag{2}$$

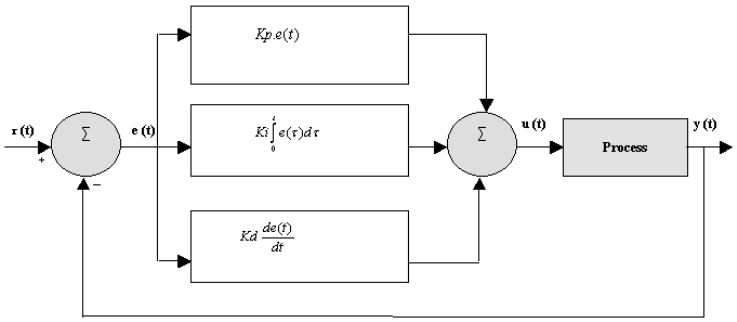


Figure 1 : Block diagram of PID controller

The gain parameters are related to the parameters of the standard form through $K_i = \frac{K_p}{T_i}$ and $K_d = K_p T_d$

This parallel form is shown in Fig 1, where the parameters are treated as simple gains, is the most general and flexible form. However, it is also the form where the parameters have the least physical interpretation and is generally reserved for theoretical treatment of the PID controller. The purpose of integral action is to increase the low frequency gain and thus steady state error reduces. The derivative action adds phase lead, which improves stability and increases system bandwidth. Fallowing [3], from a practical point of view, implementation of equation (2) has certain limitations. Firstly, actuator saturation can cause integrator windup, leading to sluggish transient response. Secondly, the pure differentiation term amplifies noise, leading to deterioration of the control command. Finally, the differentiation term acts on the error signal, taking the derivative of the command signal as well. This procedure can lead to spikes in the command signal when a user changes reference input abruptly. This paper proposes the two methods for control algorithm that overcomes the above problems is shown below.

A. Method # 1:

Differentiating both sides of equation (1) gives

$$\dot{u}(t) = K_p \dot{e}(t) + K_i e(t) + K_d \ddot{e}(t) \tag{3}$$

In order to implement the control algorithm using digital technology, equation (3) has to be discretized. The discretization can be performed in number of ways with the application of Laplace Transform to equation (1).

1. Backward Euler Method $\left(s = \frac{1 - z^{-1}}{T_s} \right)$
2. Forward Euler Method $\left(s = \frac{1 - z^{-1}}{T_s . z^{-1}} \right)$

3. Bilinear Transformation or Tustin Method $\left(s = \frac{2}{T_s} \frac{1 - z^{-1}}{1 + z^{-1}} \right)$

4. Backward Difference Method

We choose the Backward Difference Method [4] to discretize the controller. Applying Backward Difference (BD) Method to equation (3) gives

$$\frac{u(n) - u(n-1)}{T_s} = K_p \frac{e(n) - e(n-1)}{T_s} + K_i e(n)$$

$$+ K_d \frac{\dot{e}(n) - \ddot{e}(n-1)}{T_s} \tag{4}$$

Again applying BD Method on $\dot{e}(n)$ and $\ddot{e}(n-1)$ in equation (4) gives

$$\frac{u(n) - u(n-1)}{T_s} = K_p \frac{e(n) - e(n-1)}{T_s} + K_i e(n)$$

$$+ K_d \frac{\frac{e(n) - e(n-1)}{T_s} - \frac{e(n-1) - e(n-2)}{T_s}}{T_s} \tag{5}$$

Solving for $u(n)$ finally gives the discrete-time PID controller

$$u(n) = u(n-1) + K_p \{e(n) - e(n-1)\} + K_i T_s e(n)$$

$$\frac{K_d}{T_s} \{e(n) - 2e(n-1) + e(n-2)\} \tag{6}$$

In equation (6)
Ts = Sampling time of the Analog to Digital (A/D)

Converter shown Fig.2
u(n) = Discrete time PID controller output
e(n) = r(n)-y(n) = Error signal
r(n) = Reference signal
y(n) = Measured output
n = Discrete interval of time is an integer
Kp, Ki, Kd = Proportional, Integral, Derivative gain constants respectively.

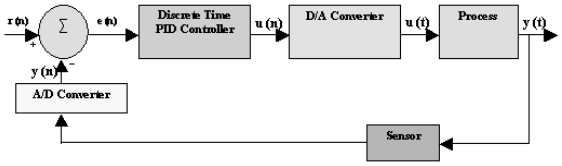


Figure 2 : Block diagram of a Discrete Time PID controller

To study the frequency responses of the PID controller take z-transform to equation (6) with zero initial conditions

$$U(z) = z^{-1}U(z) + K_p\left\{E(z) - z^{-1}E(z)\right\} + K_i T_s E(z)$$

$$+ \frac{K_d}{T_s}\left\{E(z) - 2z^{-1}E(z) + z^{-2}E(z)\right\} \quad (7)$$

On simplification of the above equation, we get

$$G_{M1}(z) = \frac{\left(K_p + K_i T_s + \frac{K_d}{T_s}\right)z^2 - \left(K_p + \frac{2K_d}{T_s}\right)z + \frac{K_d}{T_s}}{z^2 - z} \quad (8)$$

where $G_{M1}(z) = \dfrac{U(z)}{E(z)}$ = Transfer Function of Method #1 in z-domain = G_z

The frequency response (Bode diagram) of the equation (8) is shown in Fig. 3. At 100 kHz frequency the gain of the discrete time PID controller is 60 db and phase is 0.00073 degree.

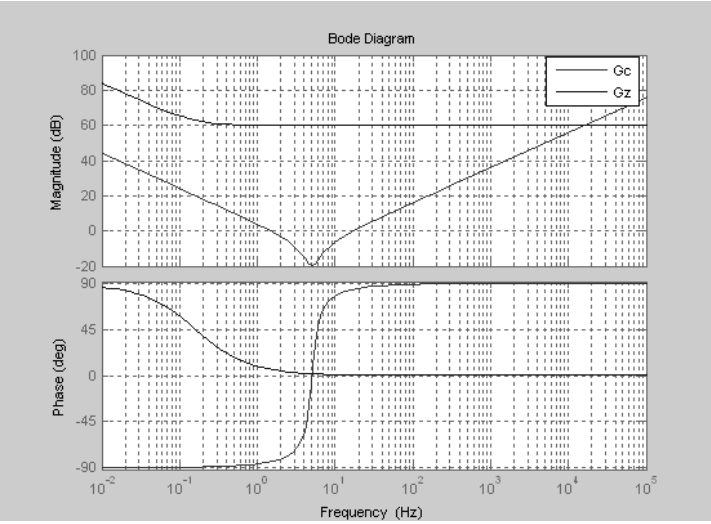


Figure 3 : Frequency response for Method #1 (Continuous-Gc & Discrete-Gz) with Kp=0.1, Ki=10, Kd=0.01, Ts=10μs

Method # 2:

An alternate equation for a discrete time PID controller can be represented by the following expression [5].

$$u(k) = u(k-1) + (K_p + K_i + K_d)e(k)$$

$$+(K_i - K_p - 2K_d)e(k-1) + K_d e(k-2) \quad (9)$$

To study the frequency responses, take z-transform on both sides of equation (9) with zero initial conditions.

$$U(z) = z^{-1}U(z) + (K_p + K_i + K_d)E(z)$$
$$+(K_i - K_p - 2K_d)z^{-1}E(z) + K_d z^{-2}E(z) \quad (10)$$

On simplification of the above equation, we get

$$G_{M2}(z) = \frac{(K_p + K_i + K_d)z^2 + (K_i - K_p - 2K_d)z + K_d}{z^2 - z} \quad (11)$$

Where $G_{M2}(z) = \frac{U(z)}{E(z)}$ = Transfer Function of Method #2 in z-domain = G_z

Fig. 4 shows the frequency response (Bode diagram) for the equation (11). At 100 kHz frequency the gain of the discrete time PID controller is 20.1 db and phase is 0 degree.

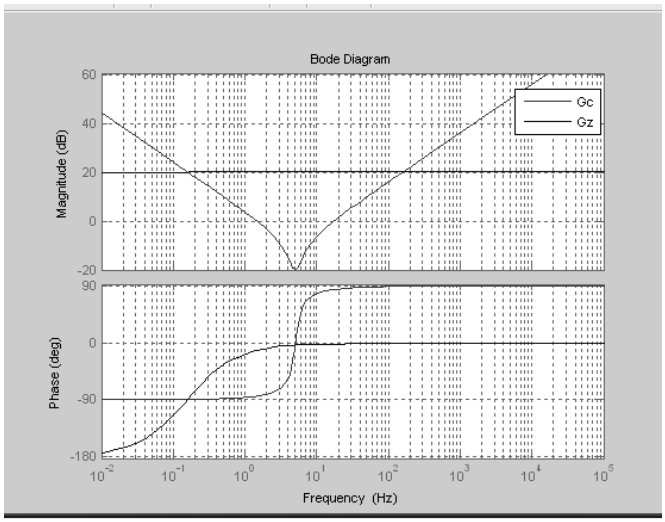


Figure 4 : Frequency response for Method #2 (Continuous-Gc & Discrete-Gz) with Kp=0.1, Ki=10, Kd=0.01, Ts=10µs

Implementation of PID Controller using FPGA

Having obtained the discrete time equation, now our focus is on the implementation of the equation. In this work we have implemented the equation (9) of Method #2 using two FPGAs (simulated, synthesized and implemented using Xilinx FPGA..

And only simulated and synthesized using Altera FPGA). The input for the control algorithm is taken from the output of PCF8591, 8-bit, 4 channel, InterIC (I2C) based Analog to Digital Converter (ADC) and a constant variable dc voltage of 0 to

4V is applied at the input of ADC to test the controller. To achieve a fast dynamic response the ADC must sample the voltage at the rate at least equal to the switching frequency. In addition, the ADC resolution must be high enough to meet voltage regulation specifications. A discrete time controller computes the digital duty-cycle command. To convert this digital duty cycle command into analog, the Digital Pulse Width Modulator (DPWM) which serves the purpose of Digital to Analog Converter (DAC) is implemented using a fast clocked counter and a digital comparator [7]. This approach is commonly used in motor drive applications.

Implementation using Xilinx FPGA

In this FPGA, First the controller was implemented using VHDL language with Xilinx ISE9.1i as a foundation tool [8] and simulated at the Register Transfer Level (RTL) to verify the correctness of the design using Modelsim 6.2C simulator tool from Mentor Graphics. By using the Xilinx ISE Foundation tools, the logic synthesis was carried out to optimize the design and the placement and routing were carried out automatically to generate the FPGA implementation file i.e. Bit file. The file is targeted to a Spartan3E XC3S100E-TQ144 with a speed grade of 5 to obtain the Pulse Width Modulated (PWM) pulses.The technology schematic of the controller is shown in Fig. 5.

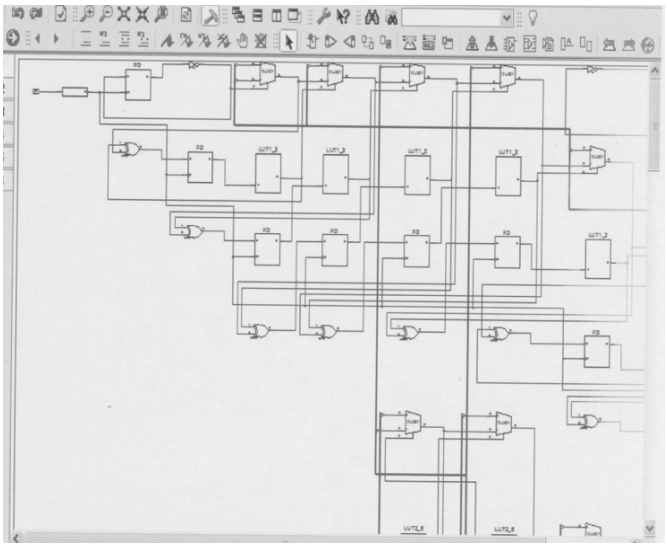


Figure 5 : Technology Schematic of the equation (9).

Implementation using Altera FPGA

In this FPGA, the control algorithm is first implemented and synthesized using Altera Quartus II 6.1 as a foundation tool [9] and simulated with the internal simulator tool. The equation (9) is represented in Fig. 6 in graphical language. The register error block stores values of e(k), using D-FF(Flip-Flop) shift operation is done to obtain e(k-1)and e(k-2). These error signals are multiplied with control gain parameters Kp,

Ki, Kd. Finally, counter based DPWM is implemented to obtain the PWM pulses.

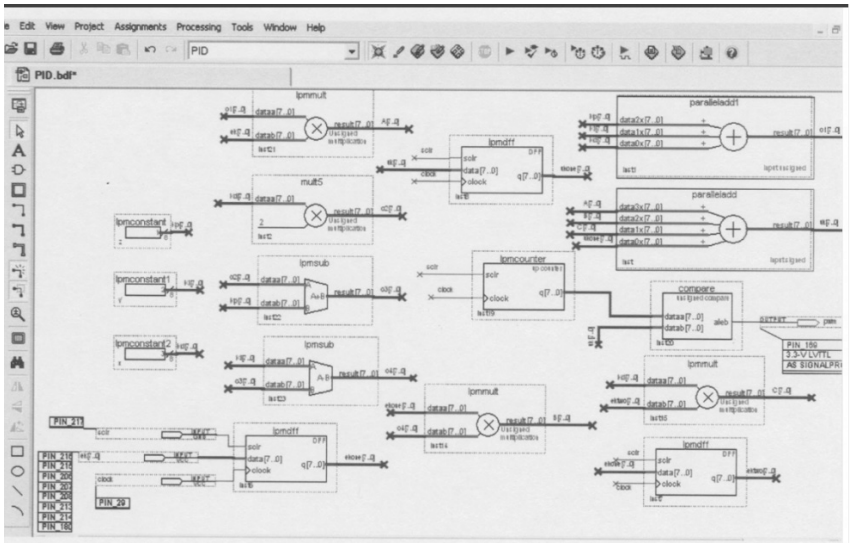


Figure 6 : Equation (9) graphical language implementation and the PID Controller simulation Interface**.**

Simulation and FPGA Implementation Results

Using Xilinx FPGA and Modelsim Simulator:

The RTL level implementation of PID controller and DPWM was described using VHDL language. To verify the behavior (function) of the controller, controller was simulated with Modelsim simulator. The simulated results are shown in Fig. 7.

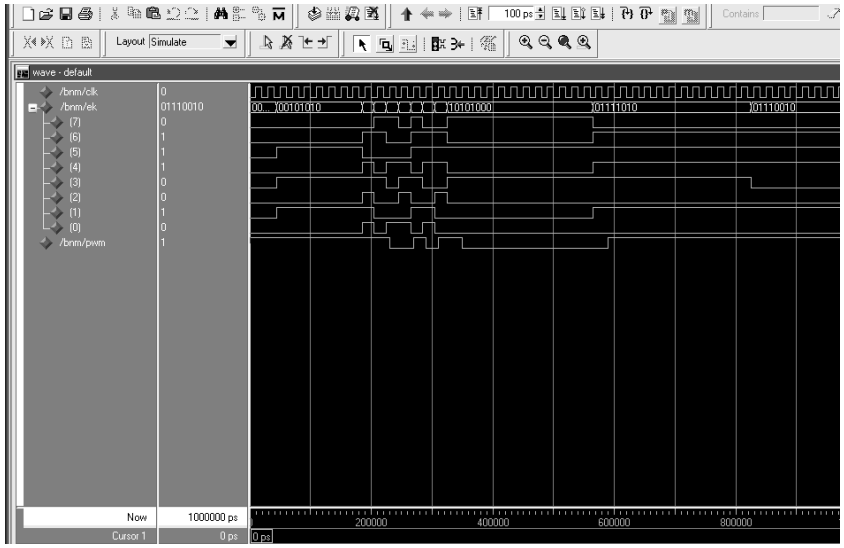
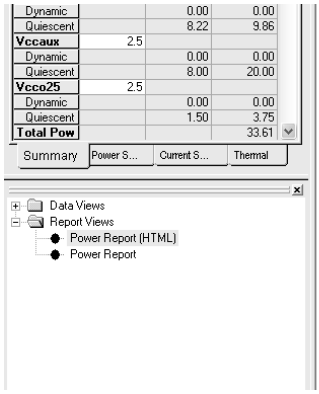


Figure 7 : Simulation of PID Controller Using Xilinx Modelsim Simulator.

Table I : Device Utilization Summary Of Xilinx Fpga

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	64	960	6%
Number of Slice Flip Flops	59	1920	3%
Number of 4 input LUTs	122	1920	6%
Number of bonded IOBs	10	108	9%
Number of MULT18X18SIOs	2	4	50%
Number of GCLKs	2	24	8%

Table I shows how much logic resource of FPGA is used to implement the whole system, and as shown in the table almost every item is below 40%. It means one can select a smaller and cheaper FPGA to further reduce the cost, or one can also build up a microcontroller Intellectual Properties (IP) into FPGA to implement more sophisticated control algorithm.



Power summary:	I(mA)	P(mW)
Total estimated power consumption:		34
Vccint 1.20V:	8	10
Vccaux 2.50V:	8	20
Vcco25 2.50V:	2	4
Clocks:	0	0
Inputs:	0	0
Logic:	0	0
Outputs:		
Vcco25	0	0
Signals:	0	0
Quiescent Vccint 1.20V:	8	10
Quiescent Vccaux 2.50V:	8	20
Quiescent Vcco25 2.50V:	2	4

Figure 8 : Estimated Power Summary of PID Controller Using Xilinx FPGA.

Timing Summary Report:
Minimum period: 13.152ns
Minimum input arrival time before clock: 15.271ns
Maximum output required time after clock: 20.391ns
Maximum combinational path delay: 22.494ns

Total Memory usage: 110000kbytes

Using Altera FPGA and Quartus II Simulator:

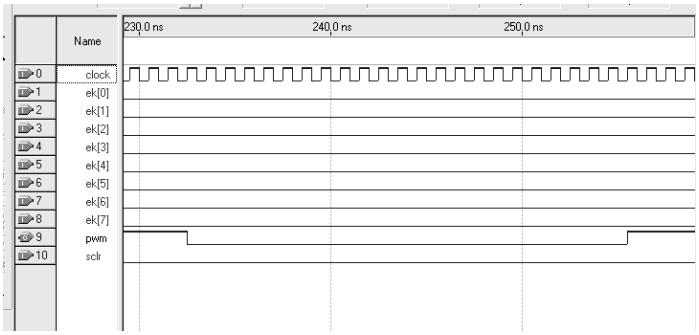


Figure 9 : Simulation of PID Controller Using Altera Quartus II Simulator.

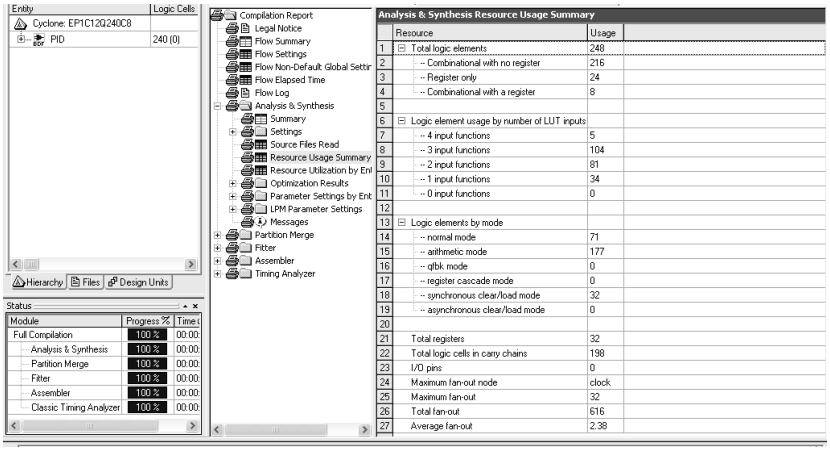


Figure 10 : Analysis and Synthesis Summary Using Altera FPGA

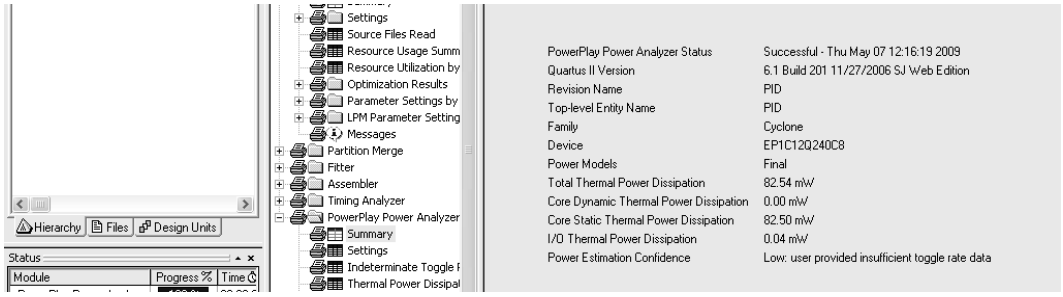


Figure 11 : Estimated Power Summary of PID Controller Using Altera FPGA.

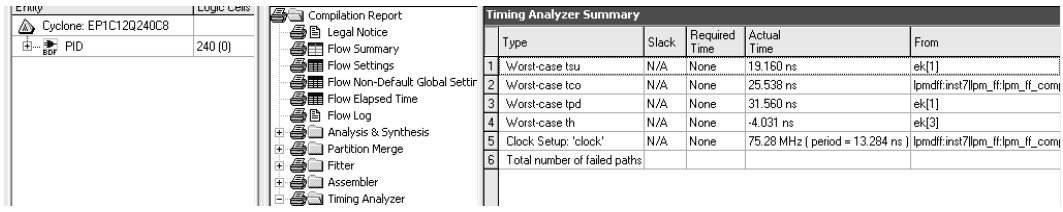


Figure 12 : Timing Analyzer Summary Using Altera FPGA.

Figure 13 : FPGA Based PID Controller Setup.

Table II : Comparision Between Xilinx and Altera Fpga

Parameters	**Xilinx FPGA**	**Altera FPGA**
Multipliers	2	3
Adders/Subtractors	3	4
Counters	2	1
Comparators	1	1
4 Input LUTs	122	79
Flip-Flops	3	3
Clock Period	13.152ns	13.284ns
Power Consumption	34mw	82.54mw
Memory Usage	110MB	128kB

Conclusion

In this paper, two FPGA platforms have been proposed for the implementation of PID controller. The results of both FPGA are compared for many parameters. The Xilinx FPGA gave a promising result for Multipliers, Adders/Subtractors, power consumption and speed compared to Altera FPGA. But for memory usage, Counters, and LUTs, Altera FPGA gave a good result. Future work will involve the implementation and integration of PID controller into a complete control system consisting of analog and digital input-output with high frequency DC-DC converter. Also we plan to investigate the quantization and limit cycle oscillations effect of the controller on DC-DC converter.

References

[1] K.J. Åström and T. H. Hagglund, "New Tuning Methods for PID Controllers." *Proc. of 3rd European Conference*, pp. 2456-2462, 1995.

[2] K.J. Åström and B. Wittenmark, "Computer Controlled Systems." Printice Hall, New Jersery, USA, 1997.

[3] B. Wittenmark, K.J. Astrom and K.E. Arzen, Computer Control: An Overview, Technical Report, Departement of Automatic Control, Lund Institute of Technology, Sweden (www.control.lth.se/kursdr/ifac.pdf), April 2003.

[4] Mukhopadya.S., A.Patra and G.P. Rao,"New Class of Discrete Time Models for Continous Time Systems," International Journal of Control, 55, pp.1161-1187.

[5] Samet, L. Masmoudi; N. Kharrat, M.W. Kamoun, L. "A digital PID Controller for Real Time and Multi Loop Control: a Comparative Study„. IEEE International Conference on Electronics, Circuits and Systems, pp. 291-296, vol.1, 1998.

[6] Klotchkov, I. V.; Pedersen, S. " A Codesign Case Study: Implementing Arithmetic Functions in FPGAçs„. IEEE 1996.

[7] A.M. Wu, J. Xiao, D. Markovic, S.R. Sanders, "Digital PWM Control: Application in Voltage Regulator Models," IEEE PESE, 1999, pp. 77-83.

[8] Xilinx ISE9.1i Software Manuals, Xilinx Inc., California, USA (www.xilinx.com), 2007.

[9] Altera Quartus II 6.1 Software Manuals, Altera Inc., San Jose, CA (www.altera.com), 2006.