FPGA-Based PID Controller Implementation*

Mohamed Abdelati
The Islamic University of Gaza
Gaza, Palestine

ملخص: أجهزة التحكم التي تعمل بمبدأ التناسب و التكامل و التفاضل تستخدم بكثرة في الانظمة الالية. هذه الأجهزة يتم بناؤها علي شكل هاردوير يستخدم قطع الكترونية تماثلية أو يتم بناؤها برمجيا باستخدام نظم محوسبة. أيضا يتم بناء هذه الأجهزة احيانا باستخدام دوائر متكاملة ذات التطبيق الخاص. في هذا البحث يتم تصميم العديد من اللبنات الضرورية لبناء تلك الأجهزة باستخدام FPGAs حيث انها تؤدي الى تحسينات في السرعة و الدقة و الحجم و السعر و استهلاك الطاقة. و لقد تم عمليا بناء جهازين بالإعتماد على هذه اللبنات كتطبيق للتحكم بالسرعة و الوضع و ذلك بغرض توضيح هذه اللبنات و فحصها.

Abstract: Proportional-Integral-Derivative (PID) controllers are widely used in automation systems. They are usually implemented either in hardware using analog components or in software using computer-based systems. They may also be implemented using Application Specific Integrated Circuits (ASICs). This paper outlines several modules necessary for building PID controllers on Field Programmable Gate Arrays (FPGAs) which improve speed, accuracy, power, compactness, and cost effectiveness. Two PID controllers for speed and position utilizing these modules are implemented and used as experimental platforms to illustrate and test the designed modules.

1 Introduction

There are two approaches for implementing control systems using digital technology. The first approach is based on software which implies a memory-processor interaction. The memory holds the application program while the processor fetches, decodes, and executes the program instructions. Programmable Logic Controllers (PLCs), microcontrollers, microprocessors, Digital Signal Processors (DSPs), and general purpose computers are tools for software implementation.

*This research was supported by the Ministry of Higher Education in Palestine.

On the other hand, the second approach is based on hardware. Early hardware implementation is achieved by magnetic relays extensively used in old industry automation systems. It then became achievable by means of digital logic gates and Medium Scale Integration (MSI) components. When the system size and complexity increases, Application Specific Integrated Circuits (ASICs) are utilized. The ASIC must be fabricated on a manufacturing line, a process that takes several months, before it can be used or even tested [1]. FPGAs are configurable ICs and used to implement logic functions. Early generations of FPGAs were most often used as *glue logic* which is the logic needed to connect the major components of a system. They were often used in prototypes because they could be programmed and inserted into a board in a few minutes, but they did not always make it into the final product. Today's high-end FPGAs can hold several millions gates and have some significant advantages over ASICs. They ensure ease of design, lower development costs, more product revenue, and the opportunity to speed products to market. At the same time they are superior to software-based controllers as they are more compact, power-efficient, while adding high speed capabilities [2].

The target FPGA device used in this research is Spartan-3 manufactured recently by Xilinx [3]. Design development and debugging is carried on a low-cost, full featured kit provided by Digilent [4]. This board, which costs less than a 100$, provides all the tools required to quickly begin designing and verifying Spartan-3 platform designs. While the implemented modules are also suited to other high density FPGAs, designs are based on 50 MHz clock and should be updated if different frequency is used.

In control systems, the majority of actuating signals and sensor returns are analog signals. Therefore, analog to digital and digital to analog conversion plays an important role in digital controllers. These converters are located at the boundary of the digital controller. Usually there are some modules within the digital system that facilitate communication with these converters. In addition, digital controllers usually encompass input/output (I/O) modules to communicate with users. Push-buttons and seven segment displays are well suited to small size and compact controllers. Along with these four mentioned building blocks a pulse width modulation (PWM) device and an optical encoder interface adapter will be designed. They are used as building blocks in many control applications such as speed and position control. One more building block for digital filters will be addressed in this work. It is essential to implement transfer functions in PID controllers.

The rest of this paper is organized as follows. In Section 2 relevant work is addressed. In Section 3, the building blocks are constructed. In Section 4, experimental work is described. Finally in Section 5, conclusions and suggestions for future work are outlined.

2 Relevant work

Modern FPGAs and their distinguishable capabilities have been advertised extensively by FPGA vendors. Moreover, some refereed articles addressed the advantages of utilizing these powerful chips [2][5]. In the past two years, Spartan II and III FPGA families from Xilinx have been successfully utilized in a variety of applications which include inverters [6][7], communications [8][9], imbedded processors [10], and image processing [11].

The implementation of PID controllers using microprocessors and DSP chips is old and well known [12][13], whereas very little work can be found in the literature on how to implement PID controllers using FPGAs. The scheme proposed in [14] is based on a distributed arithmetic algorithm where a Look-Up-table (LUT) mechanism inside the FPGA is utilized. The contribution focused on power and area issues while FPGA interfacing is totally unaddressed. In our work we introduce a simple method for implementing PID controllers together with many related constructing modules. Some other contributions focused on proposing algorithms for tuning the coefficients of PID controllers using FPGAs while the controller itself is still implemented in software. These contributions are considered complementary to our work as they provide tools for building adaptive PID applications. In [15] [16] two different algorithms for fuzzy PID gain conditioner algorithm are proposed. Both are based on fuzzy control that tunes the PID controller on-line.

A PWM generator is introduced in [17]. However, only simulation results are presented and the proposed algorithm results in greater consumption of FPGA resources compared to our algorithm which is tested experimentally. However, despite its complexity, the algorithm in [17] is superior in terms of harmonic content and is more suited to inverter applications.

In [18] the authors describe the architecture of a data acquisition system for a gamma-ray imaging camera. The system has been designed by using Xilinx Spartan II devices and 12-bit parallel A/D converters. In our work, data acquisition for

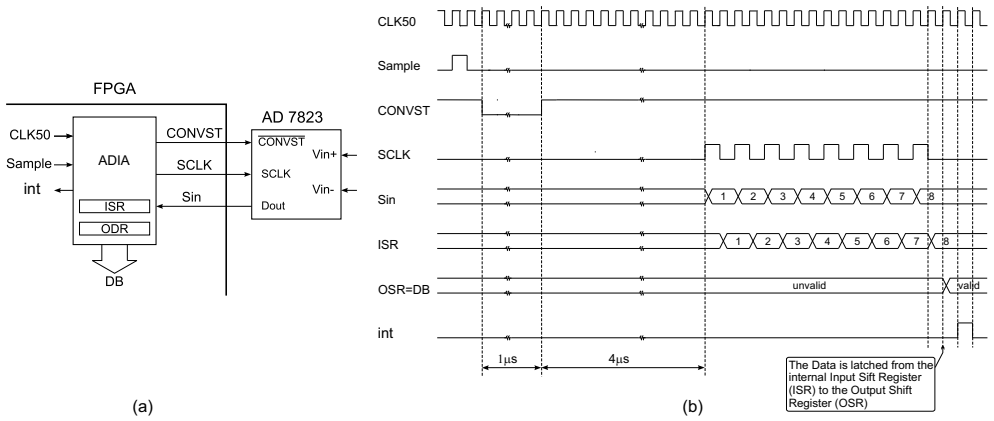

Figure 1. A/D converter interface and timing diagram.

the PID controller, which is implemented using Xilinx Spartan III, is based on 8-bit serial A/D converters. Similar converters are utilized in [19] to implement an adaptable strain gage conditioner using FPGAs. While being a smart data acquisition approach, it is costly as it is based on a soft intellectual property (IP) processor.

3 PID building blocks

In this section, implementation of analog input interface, analog output interface, pulse width modulation, optical encoder interface, user interface, and digital filters are introduced. These building blocks are the major blocks that are essential for implementing most PID controllers on FPGAs.

3.1 Analog input interface

FPGAs are well suited for serial Analog to Digital (A/D) converters. This is mainly because serial interface consumes less communication lines while the FPGA is fast enough to accommodate the high speed serial data. The AD7823 is a high speed, low power, 8-bit A/D converter [20]. The part contains a 4 μs typical successive approximation A/D converter and a high speed serial interface that interfaces easily to FPGAs as illustrated in Figure 1a.

The A/D interface adapter (ADIA) is implemented within the FPGA. Inside the FPGA, this adapter facilitates parallel data acquisition. Sampling is initiated at the

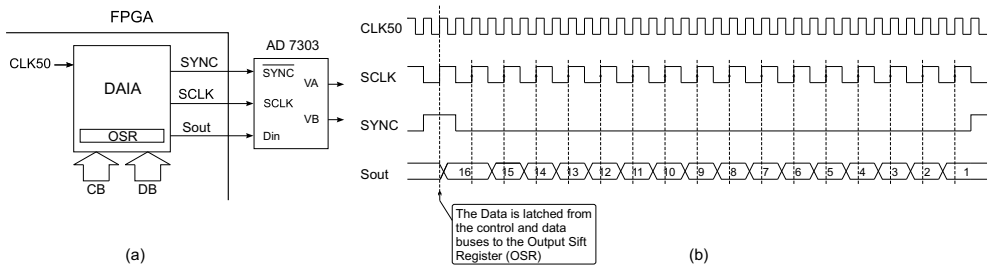

Figure 2. D/A converter interface and timing diagram.

rising edge of a clock applied at the line *sample*. Once conversion and transmission is completed, a pulse is generated at the interrupt line (*int*) and the parallel data will be available at the Data Bus (*DB*). The timing diagram of the communication protocol is illustrated in Figure 1b. The whole conversion and acquisition period is 5.4μs allowing sampling up to a rate of 185 Kilo Sample per second. This rate is more than sufficient for most PID control applications.

3.2 Analog output interface

The AD7303 is a dual, 8-bit voltage out Digital to Analog (D/A) converter [21]. This devive uses a verstile 3-wire serial interface that operates at a clock up to 30 MHz. The serial input register is 16 bits wide; 8 bits act as data bits for the D/A converter, and the remaining 8 bits make up a control register. It is interfaced to an FPGA as illustrated in Figure 2a.

The D/A interface adapter (DAIA), which is implemented within the FPGA, facilitates parallel data input for the dual D/A converters. A logic zero on the synchronization signal (*SYNC*) enables the shift register at the D/A chip to receive data from the DAIA's serial data output (*Sout*). Its serial clock (*SCLK*) frequency is 25 MHz which is half of the master clock (*CLK50*) frequency. Data is clocked into the shift register on the rising edge of the serial clock and it is sent most significant bit (MSB) first. Each transfer must consist of a 16-bit packet which is described in Table 1 while the timing diagram of the communication protocol is illustrated in Figure 2b. The transmission period of a sample is 680 ns allowing D/A conversion at an excellent rate of 1.47 MHz.

Table 1. Description of the D/A data packet.

Location	Mnemonic	Description
15	$\overline{INT}$/EXT	Select between internal and external reference.
14	X	Uncommitted bit.
13	LDAC	Load DAC bit for synchronous update of DAC outputs.
12	PDB	Power-down converter B.
11	PDA	Power-down converter A.
10	$\overline{A}$/B	Address bit to select either converter A or converter B.
9	CR1	Control Bit 1 used to implement the various data loading functions.
8	CR0	Control Bit 0 used to implement the various data loading functions.
7-0	DATA	8-bit data word where DB7 is the MSB.

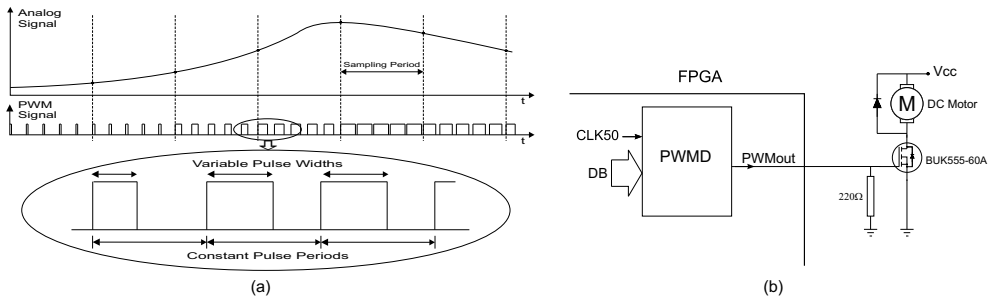

Figure 3. Pulse width modulation waveforms and hardware.

3.3 Pulse width modulation

Switching power converters are used in most DC motor drives to deliver the required energy to the motor. The energy that a switching power converter delivers to a DC motor is controlled by Pulse Width Modulated (PWM) signal applied to the gate of a power transistor. PWM signals are pulse trains with fixed frequency and magnitude and variable pulse width. There is one pulse of fixed magnitude in every Pulse Width Modulation (PWM) period. However, the width of the pulses (duty cycle) changes from pulse to pulse according to a modulating signal as illustrated in Figure 3a.

A pulse width modulation device (PWMD) implemented within an FPGA and used to control a DC motor through a Mosfet transistor driver is illustrated in Figure 3b. The modulating signal is supplied in 8-bit digital format. With 50 MHz synchronization clock, a minimum pulse width of 20 ns for the synchronization clock period is obtained. Therefore, $20 \times 256 = 5120$ ns period is necessary to represent the 256 levels of the modulating signal. This allows a maximum PWM frequency of about 195 KHz. However, such a high frequency is unnecessary and may cause undesir-

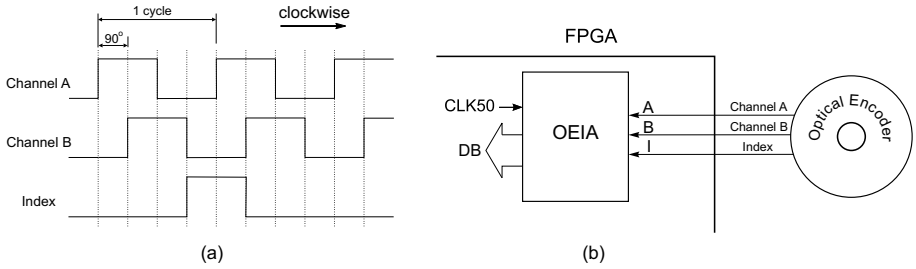

Figure 4. Optical encoder output waveforms and FPGA interface.

able effects. It contains more high frequency harmonics and its very short pulse widths may be unable to trigger the transistor driver. In most control applications, the maximum frequency content of the modulating signal is very low and a PWM frequency of about 50 Hz is suitable. A 20-bit counter runs at 50MHz completes cycles at a rate $50\text{M}/2^{20} \simeq 47\text{Hz}$. In this case, each level in an 8-bit modulating signal corresponds to $2^{20}/256 = 2^{12}$ clock pulses. Therefore, in order to generate a PWM signal at a frequency of about 47 Hz for an 8-bit modulating signal, the modulating signal is multiplied by 2^{12} and a 20-bit counter with 50 MHz clock is fired. Then, this counter is compared with the scaled modulating signal. If the scaled modulating signal is larger, the PWM output is set to one, otherwise, it is set to zero.

3.4 Optical encoder interface

An incremental optical encoder is basically an instrumented mechanical light chopper. It produces a certain number of square pulses for each shaft revolution. Single channel type encoder has one output only while phase-quadrature type has two channels whose pulse trains are $90°$ out of phase as illustrated in Figure 4a. Unlike single-channel encoders, the phase-quadrature encoders are able to detect the direction of rotation. An external electronic circuit determines which channel is leading the other and hence ascertain the direction of rotation. In order to provide a reference for the angular position information, most encoders incorporate an *index* output that goes high once for each complete revolution of the shaft [22].

Figure 4b illustrates interfacing an optical encoder to an FPGA. The optical encoder interface adapter (OEIA) helps decoding the encoder signals and provides an absolute value for the angular displacement. At the rising edge of *Channel A* signal, the value of *Channel B* is monitored. If it is found logic low, then clockwise rotation

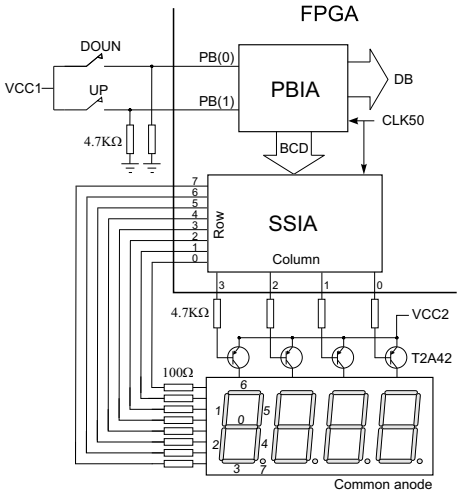

Figure 5. Push-buttons and multiplexed 7-segment displays interfacing.

is concluded and the value of *DB* is incremented by one. On the other hand, if *Channel B* is found logic high, counterclockwise rotation is inferred, and hence, the value of *DB* is decremented by one. The *Index* signal is asynchronous as it is evaluated before *Channel A* rising edge action. Once this signal is detected, the value of *DB* is reset. This helps initial condition setup and also run-time correction for any possible drift caused by noise.

3.5 User interface

An input module that implements a simple technique widely used in digital equipments is implemented. This module utilizes two push-buttons to input an 8-bit value. Pressing on the UP push-button increments the input register while pressing on the DOWN push-button decrements it. The module handles the bounce effect of the push-buttons and adjusts scanning cycle of the push-buttons according to the period of time a push-button is pressed. On the other hand, a module that facilitates displaying Binary Coded Decimal (BCD) digits on 4 multiplexed 7-segment display units is designed. This general purpose module is utilized here to display the value of the input register of the input module.

Figure 5 illustrates interfacing the input push-buttons and the multiplexed 7-segment display units. The push-buttons interface adapter (PBIA) overcomes the bounce effect, calculates the input value, supplies it to the data bus (*DB*), and generates its BCD code for possible display on 7-segment display units. The seven segments

interface adapter (SSIA) accepts a 16-bit BCD number and generates the scan patterns necessary to display the coded decimal number on the 7-segment display units which are multiplexed at a rate of 1KHz.

The bounce effect is eliminated by scanning the push-buttons at a rate with time period larger than the bounce time of the push-buttons. A bounce might last for few tens of milliseconds. Therefore, the default scanning period used in the PBIA, which is about 167.77 ms, cancels the bounce effect and yet allows fast sensitivity for the input. This scan period is made adaptive according to the last 5 input samples of the push-buttons. This history information about the input samples is used to adjust the scan rate.

3.6 Digital filters

While integer addition, subtraction, and multiplication operators are inferred by the synthesis tool, division as well as floating point arithmetic are not supported. When the divisor is an order of 2, division may be carried easily by shift operation. However, when the divisor is an arbitrary number, division is not straight forward and considered to be a difficult and time consuming operation. Little is available in the literature for VHDL source code of floating point arithmetic modules. Although there are some limited contributions available on the Internet [23], they are not officially refereed and tested by experts. The majority of contributions are not open source and sold commercially as Intellectual Property (IP) cores [24]. While they are not free, IP cores consume a substantial amount of the FPGA resources. One may avoid division in filter implementation by some normalization tricks. To illustrate these methods and compare their performance, consider the transfer function of a first order filter given by

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}} \tag{1}$$

The time domain relationship between the input u and the output y at time k is given by

$$y(k) = b_0 u(k) + b_1 u(k-1) - a_1 y(k-1) \tag{2}$$

In order to avoid floating point computation, the designer should restrict all data

representation to be integers. The communication with all interface devices is already specified to have 8-bit data width. Therefore, in order to calculate the right hand side of Equation 2 using 2's complement arithmetic, the coefficients must be rounded to integers. Unfortunately, this will not work as most of these coefficients are usually less than one or they spread a very small part of the available data domain. This can best be illustrated by examining the coefficients of the following filter which is used in the position controller example that is given in next section.

$$y(k) = 9.639u(k) - 9.543u(k-1) + 0.865y(k-1) \tag{3}$$

One way to overcome this trouble is to scale the equation by a suitable large number (N) which is an order of 2 so that the scaled coefficients in the right side are fairly spread and rounded. Then the number of bits required to represent these scaled coefficients are determined. Denoting the integer rounding operator by $\langle\ \rangle$, then Equation 3 is approximated by

$$y(k) = \langle \frac{\langle Nb_0 \rangle u(k) + \langle Nb_1 \rangle u(k-1) + \langle -Na_1 \rangle y(k-1)}{N} \rangle \tag{4}$$

Since N is chosen to be a power of 2, division is simply handled by shifting operation. This approach while being easy will not provide satisfactory results whatever the value of N. This is because N helps only to increase the resolution of the filter coefficients. But there is another source of error which is rounding the value of y each time, the fact that it influences the upcoming values in this recursive formula. This crude approximation method is used in the filter example specified in Equation 3 with $N = 16$. The resulting unit step response is shown in Figure 6 and labeled "Crude approximation."

In order to achieve fine approximation, the amount of information lost after each recursive iteration should be reduced. This can be done by keeping the quantity $N^2y(k)$ instead of $y(k)$ and utilizing it in the upcoming iteration. This approach leads to

$$N^2y(k) = \langle N^2b_0 \rangle u(k) + \langle N^2b_1 \rangle u(k-1) - \langle Na_1 \rangle \times \langle \frac{N^2y(k-1)}{N} \rangle \tag{5}$$

The block diagram implementation is illustrated in Figure 7. Using this method with $N = 16$ for the example filter used above, the filter's unit step response will

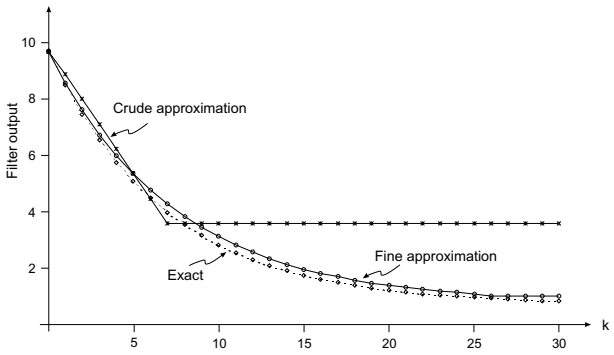

Figure 6. Unit step response of the filter.

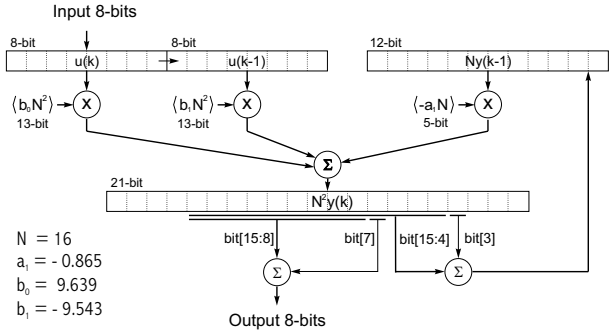

Figure 7. Implementing Equation 5.

be as shown in Figure 6. It is labeled “Fine approximation” in the figure.

4 Experimental work

This section aims to integrate the modules designed earlier in some practical examples so that they are tested and illustrated. The first example is a standard DC-motor speed control while the other one addresses demonstration of position control of an unmanned electrical dual rotor helicopter.

4.1 DC-motor speed controller

Figure 8 represents a block diagrams of a feedback motor speed control system. The formula for this basic proportional closed-loop system is

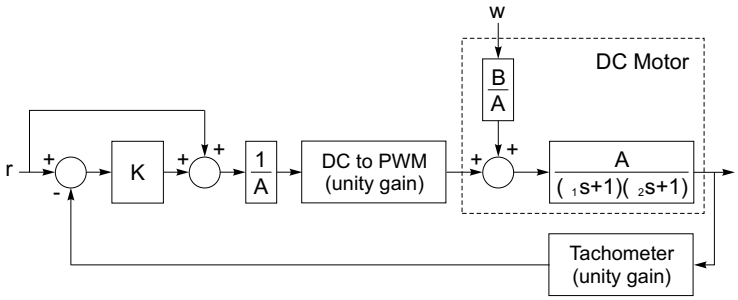

Figure 8. Proportional feedback control of a DC-Motor.

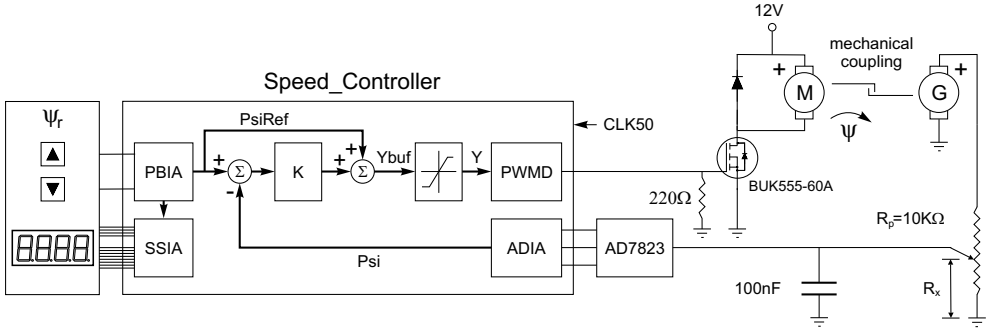

Figure 9. DC-motor speed control using FPGA.

$$\Psi(s) = \frac{K+1}{(\tau_1 s+1)(\tau_2 s+1)+K}\Psi_r(s) + \frac{B}{(\tau_1 s+1)(\tau_2 s+1)+K}W(s) \qquad (6)$$

where K is the gain of the proportional controller, τ_1 and τ_2, as well as the constants B and A (appears in the figure) are expressed in terms of the DC motor variables [25]. $\Psi_r(s)$ and $W(s)$ are the Laplace transforms of the reference speed $(\psi_r(t))$ and the load torque $(w(t))$ respectively. If these quantities are constants, that is $\psi_r(t) = \psi_r$ and $w(t) = w$, then the steady-state speed is given by

$$\psi_{ss} = \psi_r + \frac{B}{(1+K)}w \qquad (7)$$

The DC motor used in this experimental example has a rated voltage 12V, a rated speed 2500 rpm, and equipped with a tachometer which generates 18V at rated speed. A value of K=64 is found suitable for a good dynamic response of the system. Using FPGA technology, the system may be implemented as illustrated in Figure 9. The input module (PBIA) is supposed to handle entering the reference speed which will be any value from 0 to 2500 or as a percentage from 0% to 100%.

However, as 8-bit A/D converter will be used to read the actual speed, it makes no sense to have very fine reference speed specification. As the actual speed will be quantized to a maximum of 256 levels, the reference speed is better specified in 256 or less levels. Taking the speed step to be 10, a total of 250 steps will be used to cover the whole speed range.

The feedback signal is a low pass signal. Thus, the voltage divider and the capacitor at the entrance of the A/D converter are implemented to act as a low pass filter that suppresses any potential high frequency noise. Moreover, the voltage divider acts as a signal conditioner device that maps the sensor signal to the convertible input range of the A/D converter. In this experiment a 3.3V reference voltage is used for the converter which implies that in order to utilize the whole range of the converter the range of the feedback signal should be treated to be from 0 to 3.3V. As the rated speed will be represented by the number 250, its corresponding input voltage will be $3.3 \times \frac{250}{255} = 3.2353V$. Therefore, the voltage divider ratio is

$$\frac{R_x}{R_p} = \frac{3.2353}{18} = 0.1797$$

A $10\mathrm{K}\Omega$ sensitive potentiometer is used to implement this voltage divider. It should be noted that these precise calculations are just to clarify design steps and methodology. However, realization is easier; once the system is built, a certain reference speed is set and the actual speed is measured using a tachometer. Then the potentiometer is adjusted so that the measured speed is matched to the input reference speed.

The top-level module of the motor controller will utilize the PBIA, SSIA, ADIA, and PWMD modules developed earlier. Therefore, these components are declared and instanced properly. The module basically consists of 3 processes: The first one generates the sampling clock from CLK50. The maximum permissible sampling clock frequency is 185 KHz which is determined by the A/D converter as mentioned in Section 3.1. For this simple application, sampling clock frequency in the order of few hundreds is sufficient. This clock is connected to the "*Sample*" port of ADIA. Consequently, this adapter generates interrupt pulses at its port "*int*" after each sample acquisition as illustrated earlier in Figure 1. These pulses are used to synchronize the other two processes. One process works at the rising edge of the "*int*" signal to calculate the unsaturated output while the other process works at the falling edge of the "*int*" signal to register this output after handling saturation conditions.

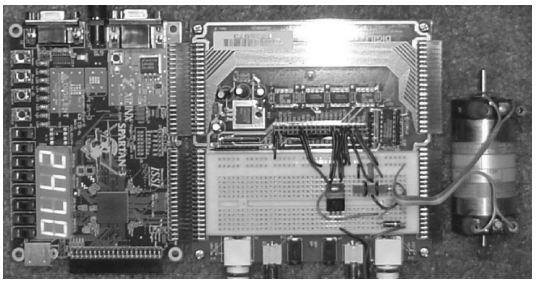

Figure 10. The FPGA-based DC-motor speed controller implementation.

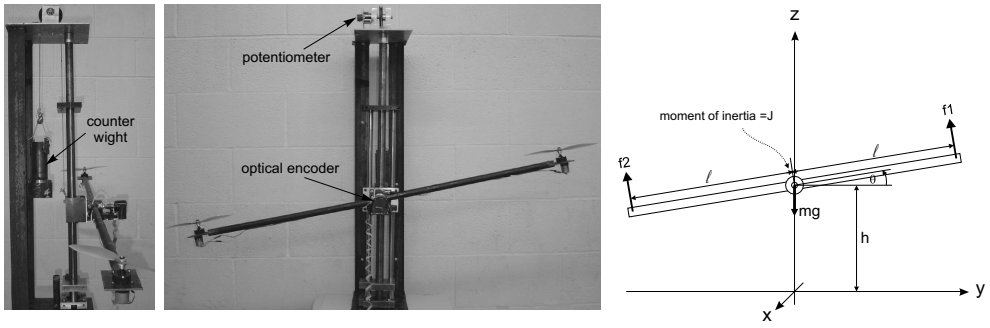

Figure 11. An unmanned helicopter model and its free-body diagram.

The pinouts of the target FPGA are specified and the standard design procedure is followed to generate and download the configuration file. The implemented system is shown in Figure 10. It worked properly as expected.

4.2 Position controller

The second experiment demonstrates position control of an unmanned electric dual rotor helicopter using the model shown in Figure 11. Each rotor in this model is powered by a 12V rated DC motor. While generating force in a direction perpendicular to the body, rotors rotate in opposite directions so that they almost cancel the moment in that direction. The control variables of the system are the elevation (h) and the slope angle (θ). These two physical quantities are measured in our model by means of a potentiometer and an optical encoder respectively.

The force ($f(t)$) generated by each rotor is adjusted by changing its speed. Therefore, this force, which is proportional to the speed of a DC motor, is related to its voltage ($v(t)$) in the Laplace domain as follows [25]:

$$F_1(s) = \frac{q_1 A_1 \; V_1(s)}{(\tau_{11}s + 1)(\tau_{12}s + 1)} \tag{8}$$

$$F_2(s) = \frac{q_2 A_2 \; V_2(s)}{(\tau_{21}s + 1)(\tau_{22}s + 1)} \tag{9}$$

where q_1, A_1, τ_{11}, τ_{12}, q_2, A_2, τ_{21}, and τ_{22} are constants. The angular acceleration along the x-axis equals the net torque in this direction divided by the model's moment of inertia (J) at the axis of rotation. Therefore, the angular displacement (θ) is given by

$$\Theta(s) = (F_1(s) - F_2(s))\frac{l}{Js^2} \tag{10}$$

where l is the force-arm length. The net vertical force is the summation of the two forces multiplied by cos(θ) minus the effective weight of the model[1]. However, since θ is small, the cosine factor may be safely dropped. The resultant vertical acceleration equals the net vertical force divided by the model's effective mass (m). Therefore, the vertical displacement (h) is given by

$$H(s) = (F_1(s) + F_2(s) - mg)\frac{1}{ms^2} \tag{11}$$

where g is the acceleration of gravity. This described feedback control system may be modeled as shown in Figure 12. In this system, h_r and θ_r represent the reference points of the elevation and slope respectively while h and θ are the actual measured values. The error signal in h influences both forces in the same direction whereas the error signal in θ influences them in opposite directions. The elevation and slope controllers are decoupled to simplify design and implementation.

This model is digitized and Matlab is used for simulation and estimation of the PID filters' parameters. It is found that

$$G(z) = \frac{9.639 - 9.543z^{-1}}{1 - 0.865z^{-1}} \tag{12}$$

[1]Similar to elevators, a counter weight is used to minimize the force necessary to move the body. Therefore, the effective mass (m) equals the mass of the main body minus the mass used in the counter weight. Equivalently, the effective weight equals the weight of the main body minus the counter weight.

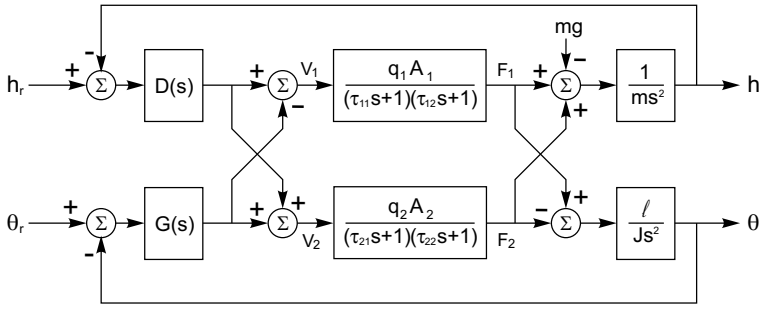

Figure 12. Block diagram of the position control system.

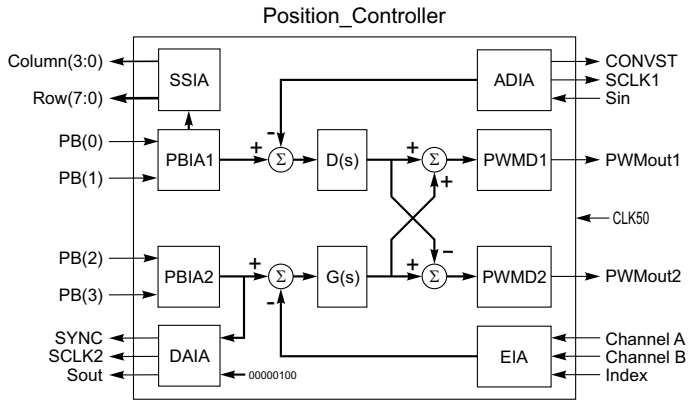

Figure 13. The top-level module of the position controller.

and

$$D(z) = \frac{6.482 - 6.394z^{-1}}{1 - 0.913z^{-1}} \tag{13}$$

insures good dynamic response behavior. The top-level module of the position controller is illustrated in Figure 13. Push-buttons PB(0) and PB(1) are used to enter the elevation set point which is echoed on seven segment displays. On the other hand, push-buttons PB(2) and PB(3) are used to enter the slope set point which is displayed on an analog voltmeter. The rest of work is similar to what is done in the first experiment. Upon running the system, it is noticed that the angle control works perfectly while the elevation control has some lazy behavior. This unexpected behavior is reasoned by considerable friction encountered in the vertical mounting beam. However, as an overall performance, results are good and fulfil the aim of the experiment.

5 Conclusions

Today's high-speed and high-density FPGAs provide viable design alternatives to ASIC and microprocessor-based implementations. Several building modules for implementing PID controllers on these FPGAs are constructed in this work. These modules are tested successfully through two experimental platforms.

Algorithms and implementations are described in sufficient details from a practical point of view for readers to digest the addressed subject and replicate the work. The VHDL code of all presented modules and examples may be obtained directly from the author.

Implementing PID controllers on FPGAs features speed, accuracy, power, compactness, and cost improvement over other digital implementation techniques. In a future fork we plan to investigate implementation of fuzzy logic controllers on FPGAs. Also we plan to explore embedded soft processors, such as *MicroBlaze*, and study some applications in which design partitioning between software and hardware provides better implementations.

Acknowledgment: The author is grateful to his Fulbright host at Texas A&M university, Dr. Reza Langari, for his kind company and helpful research environment.

References

[1] G. Martin and H. Chang, "System-on-Chip design", Proceedings of the 4th international conference on ASIC, Oct. 2001, pp. 12-17.

[2] Anthony Cataldo, "Low-priced FPGA options set to expand" Electronic Engineering Times Journal, N 1361, PP 38-45, USA 2005.

[3] Xilinx, "Spartan-3 FPGA Family: Complete Data Sheet" 2004. http://www.xilinx.com/bvdocs/publications/ds099.pdf

[4] Digilent, Inc., "Digilent Spartan-3 System Board", June, 2004. http://www.digilentinc.com/Data/Products/S3BOARD/S3BOARD-brochure.pdf

[5] Gordon Hands, "Optimised FPGAs vs dedicated DSPs", Electronic Product Design Journal, V 25, N 12, UK December 2004.

[6] R. Jastrzebski, A. Napieralski,O. Pyrhonen, H. Saren, “Implementation and simulation of fast inverter control algorithms with the use of FPGA circuit”, 2003 Nanotechnology Conference and Trade Show, pp 238-241, Nanotech 2003.

[7] Lin, F.S.; Chen, J.F.; Liang, T.J.; Lin, R.L.; Kuo, Y.C. “Design and implementation of FPGA-based single stage photovoltaic energy conversion system”, Proceedings of IEEE Asia-Pacific Conference on Circuits and Systems, pp 745-748, Taiwan, Dec. 2004.

[8] Bouzid Aliane and Aladin Sabanovic, “Design and implementation of digital bandpass FIR filter in FPGA”, Computers in Education Journal, v14, p 76-81, 2004.

[9] M. Canet, F. Vicedo,V. Almenar, J. Valls, “FPGA implementation of an IF transceiver for OFDM-based WLAN”, IEEE Workshop on Signal Processing Systems, SiPS: Design and Implementation, PP 227-232, USA 2004.

[10] Xizhi Li, Tiecai Li, “ECOMIPS: An economic MIPS CPU design on FPGA”, Proceedings - 4th IEEE International Workshop on System-on-Chip for Real-Time Applications, PP 291-294, Canada 2004.

[11] R. Gao, D. Xu,J. P. Bentley, “Reconfigurable hardware implementation of an improved parallel architecture for MPEG-4 motion estimation in mobile applications”, IEEE Transactions on Consumer Electronics, V49, N4, November 2003.

[12] H. D. Maheshappa, R. D. Samuel, A. Prakashan, “Digital PID controller for speed control of DC motors”, IETE Technical Review Journal, V6, N3, PP171-176, India 1989.

[13] J. Tang, “PID controller using the TMS320C31 DSK with on-line parameter adjustment for real-time DC motor speed and position control”, IEEE International Symposium on Industrial Electronics, V2, PP 786-791, Pusan 2001.

[14] Y. F. Chan, M. Moallem, W. Wang, “Efficient implementation of PID control algorithm using FPGA technology”, Proceedings of the 43ed IEEE Conference on Decision and Control, V5, PP. 4885-4890, Bahamas 2004.

[15] Bao-Sheng Hu, Jing Li, “Fuzzy PID gain conditioner: Algorithm, architecture and FPGA implementation”, Journal of Engineering and Applied Science, PP 621-624, US 1996.

[16] Wei Jiang, Jianbo Luo, “Realization of fuzzy controller with parameters PID self-tuning by combination of software and hardware”, Proceedings of the Second International Symposium on Instrumentation Science and Technology, V1, PP 407-410, China 2002.

[17] D. Deng, S. Chen,G. Joos, “FPGA implementation of PWM pattern generators”, Canadian Conference on Electrical and Computer Engineering, V1, PP 225-230 May 2001. and Electronics Engineers Inc.

[18] K. Nurdan, T. Conka-Nurdana, H. J. Beschc, B. Freislebenb, N. A. Pavelc, A. H. Walentac, “FPGA-based data acquisition system for a Compton camera”, Proceedings of the 2nd International Symposium on Applications of Particle Detectors in Medicine, Biology and Astrophysics, V510, N1, PP. 122-125, Sep 2003.

[19] S. Poussier, H. Rabah, S. Weber, “Smart Adaptable Strain Gage Conditioner: Hardware/Software Implementation”, IEEE Sensors Journal, V4, N2, April 2004.

[20] Analog Devices Inc., “2.7 V to 5.5 V, 5 μs, 8-Bit ADC in 8-Lead microSOIC/DIP”, 2000.
http://www.analog.com/UploadedFiles/Data_Sheets/216007123AD7823_c.pdf

[21] Analog Devices Inc., “+2.7 V to +5.5 V, Serial Input, Dual Voltage Output 8-Bit DAC”, 1997.
http://www.analog.com/UploadedFiles/Data_Sheets/48853020AD7303_0.pdf

[22] S. Holle, “Incremental Encoder Basics”, Sensors, April 1990, pp.22-30.

[23] J. Detrey, “A VHDL Library of Parametrisable Floating-Point and LNS Operators for FPGA”, 2003.
http://perso.ens-lyon.fr/jeremie.detrey/FPLibrary/FPLibrary-0.9.tgz

[24] Transtech DSP, “Quixilica Floating Point FPGA Cores”, 2002.
http://www.transtech-dsp.com/datasheets/qx-dsp001-fp_hm_v1.pdf

[25] G. Franklin, J. Powell, and A. Emami-Naeini, “Feedback control of dynamic systems”, Addison-Wesley, 4th ed., 2002.

[26] R. Coughlin and F. Driscoll, “Operational Amplifiers and Linear Integrated Circuits”, Prentice Hall, 6th ed., 2000.