Design and Implementation of Modular FPGA-based PID Controllers

Yuen Fong Chan, M. Moallem, and Wei Wang

September 26, 2005

Abstract

In this paper, modular design of embedded feedback controllers using the Field Programmable Gate Array (FPGA) technology is studied. To this end, a novel Distributed Arithmetic (DA)-based Proportional Integral Derivative (PID) controller algorithm is proposed and integrated into a digital feedback control system. The DA-based PID controller demonstrates 80% savings in hardware utilization and 40% savings in power consumption compared to the multiplier-based scheme. It also offers good closed-loop performance while using less resources, resulting in cost reduction, high speed, and low power consumption, which is desirable in embedded control applications. The complete digital control system is built using commercial FPGAs to demonstrate the efficiency. The design uses a modular approach so that some modules can be reused in other applications. These reusable modules can be ported into Matlab/Simulink as Simulink blocks for hardware/software co-simulation, or can be integrated into a larger design in the Matlab/Simulink environment to allow for rapid prototyping applications.

Keywords: Embedded controllers, FPGA design, distributed arithmetic, power and speed optimization, PID controller.

1 Introduction

The Proportional Integral Derivative (PID) controller is one of the most common types of feedback controllers used in dynamic systems [1]. This controller has been widely used in many different areas such as aerospace, process control, manufacturing, robotics, automation,

*This work was supported in part by discovery grants 227612 and 261527 from the Natural Sciences and Engineering Research Council (NSERC) of Canada and the Canada Foundation For Innovation under the New Opportunities Program.

†M. Moallem (corresponding author) is with the Department of Electrical and Computer Engineering, University of Western Ontario, London, Ontario, Canada, N6G 1H1 (Email: mmoallem@engga.uwo.ca, Fax: (519) 850-2436), W. Wang is with the Department of Electrical and Computer Engineering, Indiana University-Purdue University, Indianapolis, USA.

and transportation systems. Implementation of PID controllers has gone through several stages of evolution, from the early mechanical and pneumatic designs to the microprocessor-based systems. Recently, Field Programmable Gate Arrays (FPGA) have become an alternative solution for the realization of digital control systems, previously dominated by the general-purpose microprocessor systems [1], [2]. The FPGA-based controllers offer advantages such as high speed, complex functionality, and low power consumption. These are attractive features from the embedded systems design point of view [3]. Previous work has reported the use of FPGAs in digital feedback control systems such as magnetic bearings [4], PWM inverters [5], induction motors [6], AC/DC converters [7], variable-speed drives [8], and anti-windup compensation of controllers [9]. Another advantage of FPGA-based platforms is their capability to execute concurrent operations, allowing parallel architectural design of digital controllers [7], [9].

Conventional implementation of FPGA based controllers have not focused on an optimal use of hardware resources. These designs usually require a large number of multipliers and adders and do not efficiently utilize the memory-rich characteristics of FPGAs. In [10], the multiplier-based PID controller block takes up as much as 740 logic cells or 64% utilization of the chip. Each logic cell contains a four-input Look Up Table (LUT), a programmable Flip-Flop (FF) with a synchronous enable, a carry chain, and a cascade chain [11].

An FPGA chip consists of a lot of memory blocks, referred to as Look-Up Tables (LUT), which can be utilized to improve performance of certain operations such as multiplication while the tradeoff for speed can be tolerated. In this paper, we study the design of an efficient PID controller using the distributed arithmetic (DA) scheme. Based on the LUT scheme, the proposed PID controller reduces the cost of the FPGA design by enabling the chip to accommodate more logic and arithmetic functions while requiring less power consumption. Also, due to the flexibility of using look-up tables in FPGAs, the design method can be used implement other algorithms such as antiwindup compensation or adaptive control schemes. For the next generation of FPGAs, in which A/D and D/A converters are built into the chip, the proposed structure is more efficient in terms of hardware resources, power consumption and control performance when compared with the standard micro-controller IP cores. This is due to the fact that custom-made logic can generally outperform the general purpose micro-controllers [3].

In this paper, a case study is presented in which a modular FPGA-based design approach is applied to design a temperature control system. The same approach can be extended to design other embedded controllers using FPGA. The complete system is implemented by dividing system functions into reconfigurable modules. In general, embedded control designers need to go through three phases in the the design of digital control systems: (1) Software modeling/simulation in an environment such as Matlab/Simulink, (2) Hardware implementation, and (3) Co-simulation of the whole system including both hardware and software [12]. Using reusable and reconfigurable modules, the designer's task in developing control applications can be greatly simplified by porting the design into a familiar environment such as Matlab/Simulink. As a result, the development time for designing efficient embedded software is greatly reduced.

The organization of this paper is as follows. In section 2, an overview of the components of a general purpose PID-based feedback control system is presented followed by an approach for

designing the control system using FPGA technology. In this regard, a novel DA-based PID controller suitable for FPGA implementation is discussed. In Section 3, the implementation results for a temperature control system using the Xilinx [13] and Altera [11] FPGA chips are discussed. Section 4 describes how the FPGA-based controller modules can be used in other applications. Conclusions are discussed in Section 5.

2 Design of FPGA-based Modules in the Feedback Control Loop

The block diagram of a general purpose PID-based feedback control system is shown in Figure 1, where u_c is the command signal, y is the feedback signal, e is the error signal, and u is the control input.

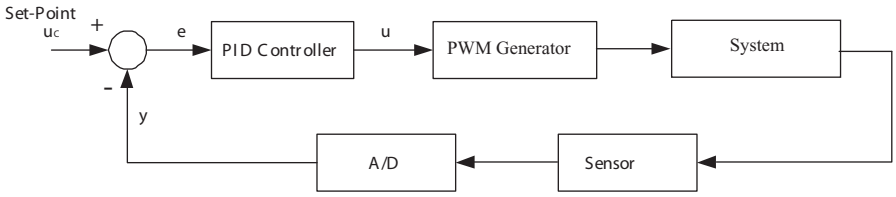

Figure 1: Block diagram of a general PID-based feedback control system.

An example of a PID-based feedback control system is the microcontroller-based temperature control system in [14] that will be used in this paper as a case study design problem. The microcontroller is replaced by an FPGA chip in this paper as shown in Figures 4 and 6. The system consists of a DC fan for cooling the tube, a lamp for heating the tube, and a 10-k thermistor for measuring temperature.

Since an FPGA-based controller can offer advantages in terms of speed, power consumption, and cost over the microprocessor-based approach, we concentrate on the FPGA design of modules required in this case study. In this regard, modules such as the Pulse-Width Modulation (PWM) Generator, PID Controller, and the Analog to Digital Converter (ADC) module can be reused in other applications.

Figure 2 shows the block diagram of the temperature control system using the Unified Modeling Language (UML) [3], consisting of the following modules: PID Controller, PWM Generator, User Interface, and ADC interface. The Temperature Controller module (class) is the main module that communicates with and controls PID Controller, PWM Generator, ADC Interface, and User Interface classes. The former three classes interact with hardware objects to acquire data from sensors, interact with the user, and send PWM signals to the motor or lamp.

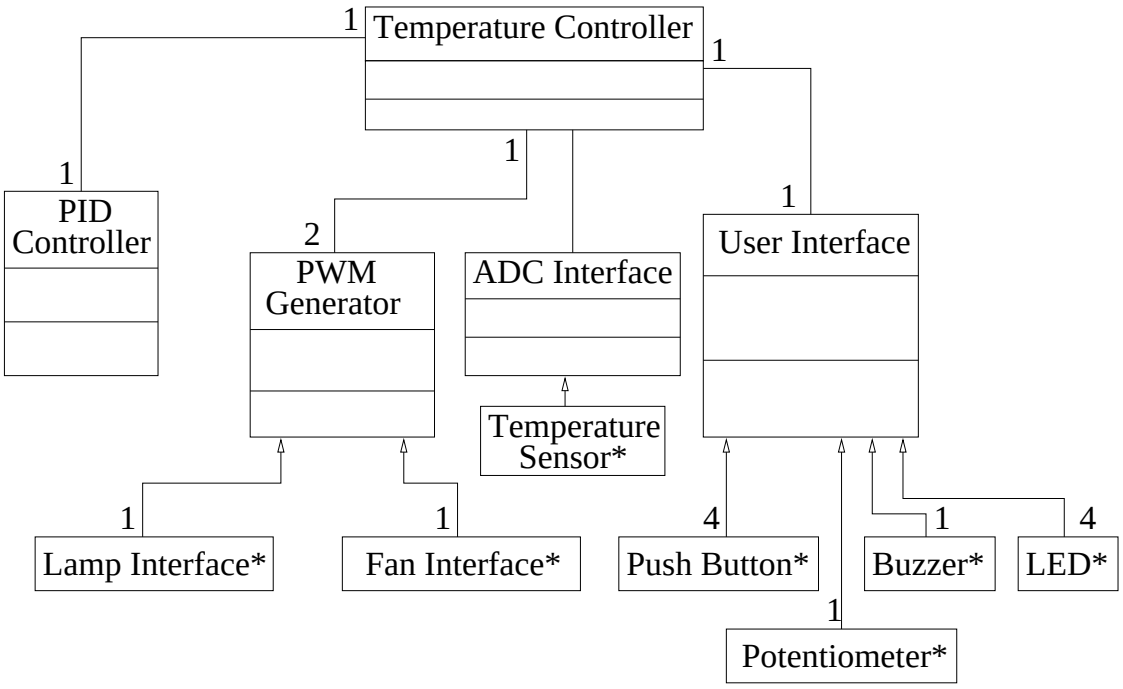

Figure 2: A UML class diagram of a general PID-based feedback control system ($*$ denotes a physical device).

2.1 PWM Generator, User Interface, ADC Interface, and FSM Modules

The PWM Generator is designed using a comparator and a counter to convert the control input to a periodic square wave of variable duty cycle. The counter is incremented every clock cycle and its value is compared with the reference control input u obtained from the PID controller. When the counter value is less than u, the PWM pulse is toggled high, otherwise, it is toggled low. The counter is incremented until the the PWM period is reached, after which it is reset to zero.

An external ADC is connected to the general user I/O connectors on the FPGA board. The ADC interface sends polling signals to the ADC in a periodic fashion to obtain the sensor voltage. The sampled sensor voltage is then sent to the FPGA for calculation of the control signal.

The UML class diagram is helpful in the conceptual design of the controller but lacks behavioral information. Since the controller has to respond to events and work under different modes, a Finite State Machine (FSM) was designed. In particular, the FSM controls operation of the system in response to user's requests such as PID tuning, startup, standby, and shut down which can be built using FPGA technology.

The User Interface module in Figure 2 decodes the user's inputs such as the set point temperature, tunes the PID gains, and starts and stops the system. The user interface contains two sub-modules including a keyboard interface and an LCD interface. The keyboard interface decodes user inputs from the keyboard and the LCD interface processes the keyboard

inputs for display on the LCD.

2.2 PID Controller Implementation using FPGA

In the FPGA implementation of the PID controller, a major effort is placed on the hardware optimization of the controller. In this regard, an area-efficient DA-based algorithm for the PID controller is proposed in this section. An area efficient controller means that it can fit in a smaller FPGA chip, resulting in cost reduction of the controller hardware.

Following [15], let us consider an improved PID control algorithm given by

$$U(s) = K(bU_c(s) - Y(s) + \frac{1}{sT_i}(U_c(s) - Y(s)) - \frac{sT_d}{1 + sT_d/N}Y(s)) \quad (1)$$

where K, b, T_i, T_d, and N are controller parameters, and $U(s)$, $U_c(s)$, and $Y(s)$ denote the Laplace transforms of u, u_c, and y, respectively. In order to implement the control algorithm using digital technology, equation (1) has to be discretized. Denoting the sampling period as T, and using backward differences to discretize the derivative term and forward differences for the integral term, (1) can be written as

$$u(k) = P(k) + I(k) + D(k) \quad (2)$$

where k denotes the k-th sampling instant, and

$$\begin{aligned} P(k) &= K(bu_c(k) - y(k)) \\ I(k) &= I(k-1) + \frac{K}{T_i}(u_c(k-1) - y(k-1)) \\ D(k) &= \frac{T_d}{T_d + NT}D(k-1) - \frac{KT_dN}{T_d + NT}(y(k) - y(k-1)) \end{aligned} \quad (3)$$

in which $y(k)$ is the feedback signal at the current instant k, $y(k-1)$ is the feedback signal at the previous instant $k-1$, $u_c(k)$ is the command signal at the current instant, $u_c(k-1)$ is the command signal at the previous instant, $I(k-1)$ is the integral term at the previous instant, $D(k-1)$ is the derivative term at the previous instant, K, b, T_i, T_d, N are controller parameters, and T is the sampling period.

Having obtained the discretized control algorithm, the focus is then placed its efficient implementation. The direct implementation of the terms in (3) using FPGA requires a total of 5 multipliers, 7 adders/subtractors, and 4 delay blocks. The $P(k)$ term requires 2 multipliers and 1 adder/subtractor, $I(k)$ requires 1 multiplier and 2 adders/subtractors, $D(k)$ requires 2 multipliers and 2 adders/subtractors, and $u(k)$ requires 2 adders. Each of the $u_c(k-1)$, $y(k-1)$, $I(k-1)$ and $D(k-1)$ terms in (3) requires one delay block. Thus, a total of 4 delay blocks are also required.

The above multiplier-based design is not optimized as it uses many multipliers and adders. This is restrictive since an FPGA chip has a limited number of configurable logic blocks. In order to reduce the required multipliers and adders, the Distributed Arithmetic (DA) method [16] is applied in this paper to replace the multiplication operation by simple shifting and addition operations. The procedure is discussed in the following.

2.2.1 Distributed-Arithmetic (DA)

Distributed Arithmetic (DA) [16] is a bit-serial computation algorithm that performs multiplication using a Look-up Table (LUT)-based scheme. Consider a sum of product calculation written as

$$Y = \sum_{k=0}^{N-1} A_k x_k \tag{4}$$

where A_k's are constant coefficients, x_k's are input data of size N, and Y is the output. Representing x_k in a bit-wise format and assuming that it is a 2's complement fractional number, we have

$$x_k = -x_{k,0} + \sum_{j=1}^{M-1} x_{k,j} 2^{-j} \tag{5}$$

where $x_{k,j}$ is the j-th bit of x_k, $x_{k,0}$ is the sign bit, and M is the word size. Substituting (5) into (4), we have

$$\begin{aligned} Y &= \sum_{k=0}^{N-1} A_k(-x_{k,0} + \sum_{j=1}^{M-1} x_{k,j} 2^{-j}) && (6) \\ &= -\sum_{k=0}^{N-1} A_k x_{k,0} + \sum_{j=1}^{M-1} (\sum_{k=0}^{N-1} A_k x_{k,j}) 2^{-j}. && (7) \end{aligned}$$

Defining

$$Z_j = \sum_{k=0}^{N-1} A_k x_{k,j} \ (j \neq 0) \tag{8}$$

and

$$Z_0 = -\sum_{k=0}^{N-1} A_k x_{k,0} \tag{9}$$

the output Y can then be expressed as follows

$$Y = \sum_{j=0}^{M-1} Z_j 2^{-j}. \tag{10}$$

In (10), Z_j's are the sums of products of A_k's and x_k's. Since A_k's are constants and x_k's are either 1 or 0, there are 2^N possible values for Z_j. Therefore, they can be pre-computed and stored in an LUT. Each x_k, where $k = 0$ to $N-1$, forms the address of the LUT. The result Y can then be computed using shifting and addition operations for all Z_j's from j-th bit.

2.2.2 DA-based PID Controller

Let us consider the controller terms given in (3). Assuming that $u_c(k)$, $u_c(k-1)$, $y(k)$, $y(k-1)$, and $D(k-1)$ are m-bit fixed-point numbers and $[j]$ represents the j-th bit of each number, we have

$$P(k) = \sum_{j=0}^{m-1}(Kb \times u_c(k)[j] - K \times y(k)[j]) \times 2^j \quad (11a)$$

$$I(k) = I(k-1) + \sum_{j=0}^{m-1} \frac{KT}{T_i}(u_c(k-1)[j] - y(k-1)[j]) \times 2^j \quad (11b)$$

$$D(k) = \sum_{j=0}^{m-1}(\frac{T_d}{T_d+NT}D(k-1)[j] - \frac{KT_dN}{T_d+NT}(y(k)[j] - y(k-1)[j])) \times 2^j. \quad (11c)$$

The results of $(Kb \times u_c(k)[j] - K \times y(k)[j])$, $(\frac{KT}{T_i}(u_c(k-1)[j] - y(k-1)[j]))$, $(\frac{T_d}{T_d+NT}D(k-1)[j])$, and $(-\frac{KT_dN}{T_d+NT}(y(k)[j] - y(k-1)[j]))$ can be pre-computed and stored in four LUTs, referred to as LUT_P, LUT_I, LUT_{D1}, and LUT_{D2}. The contents of LUT_P is shown in Table 1, with other LUTs taking a similar form. Using the four LUTs and the corresponding shift-add accumulators (ACCs), the $P(k)$, $I(k)$, and $D(k)$ terms can be obtained in m clock cycles. The main advantage of the DA expression given by (11a), (11b), and (11c) lies in its capability to compute the PID function utilizing the LUT-rich FPGA.

Table 1: Contents of the LUT_P.

$u_c(k)[j]$	$y(k)[j]$	LUT_P
0	0	0
0	1	$-K$
1	0	Kb
1	1	$Kb-K$

Based on equations (11a)-(11c), the direct DA implementation of the PID controller is shown in Figure 3. It consists of four delay blocks, four LUTs, four ACCs, and four adders. The delay blocks 1 and 2 are used to obtain $u_c(k-1)$ and $y(k-1)$, while the delay blocks 3 and 4 are used to generate the terms $I(k-1)$ and $D(k-1)$, respectively. Four LUTs and four ACCs are used to provide the terms $P(k)$, $I'(k)$, $D1(k)$, and $D2(k)$, respectively. The ACC consists of a shift register and an adder/subtractor. Finally, one adder computes the sum of $I(k-1)$ and $I'(k)$, resulting in $I(k)$, one adder calculates the sum of $D1(k)$ and $D2(k)$, resulting in $D(k)$, and two adders calculate the sum of $P(k)$, $I(k)$, and $D(k)$, resulting in $u(k)$. The throughput (speed) of this PID implementation is $(m+1)$ clock cycles, i.e., m clock cycles to generate the result, and one more clock cycle to update the $u_c(k-1)$, $y(k-1)$, $I(k-1)$, and $D(k-1)$ terms. For the multiplier-based controller the throughput is one clock cycle. The latency is also $(m+1)$ clock cycles, whereas for the multiplier-based method it is one clock cycle. However, it terms of complexity, the DA-based scheme requires 4 LUTs, 4 ACCs, 4 delay blocks, and 4 adders, whereas the multiplier-based method requires 5 multipliers, 4 delay blocks, and 7 adders which is less area efficient than the DA-based scheme.

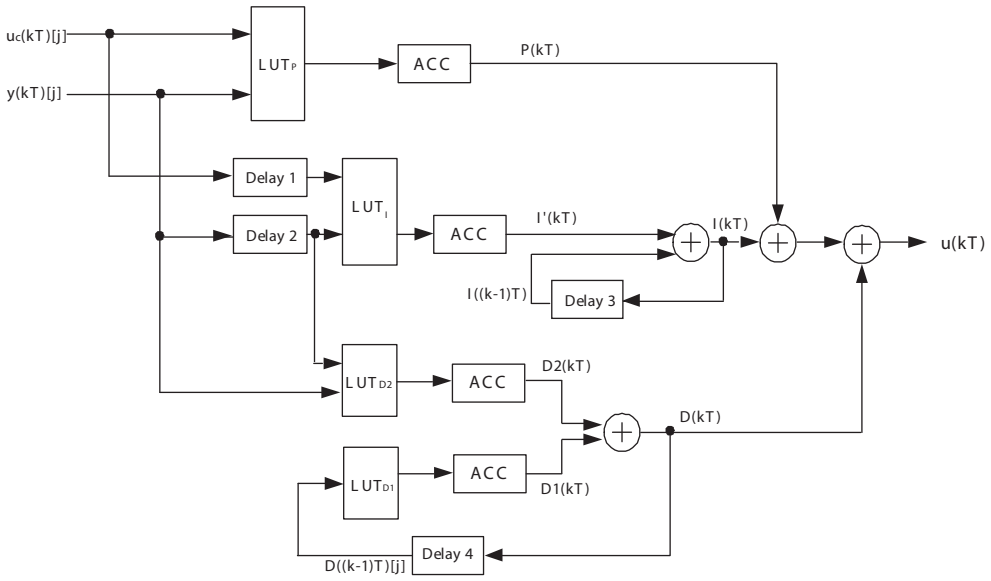

Figure 3: Architecture of the proposed DA-based PID controller.

2.3 Application to a Temperature Control System

In this section, we use the modules described in sections 2.2.1 and 2.2.2 to design a PID feedback controller for a temperature control system. The FPGA-based temperature control system shown in Figure 4 is comprised of the following components: (1) A tube with a fan, a light bulb, and a thermistor, (2) An I/O panel that includes a Digilent DIO2 board with an on-board CPLD [17], push-button keys with an LCD display, and a PS2 keyboard which is connected to the DIO2 board, (3) An ADC chip mounted on a separate circuit board, and (4) A Digilent D2E FPGA development board consisting a Xilinx Spartan-2E FPGA [13]. Figure 5 illustrates the block diagram of the FPGA-based temperature control system with the actual setup shown in Figure 6.

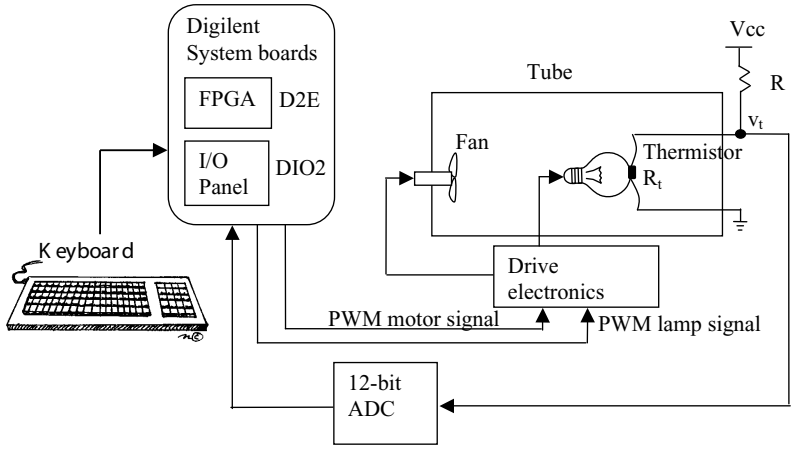

Figure 4: Components of the FPGA-based temperature control system.

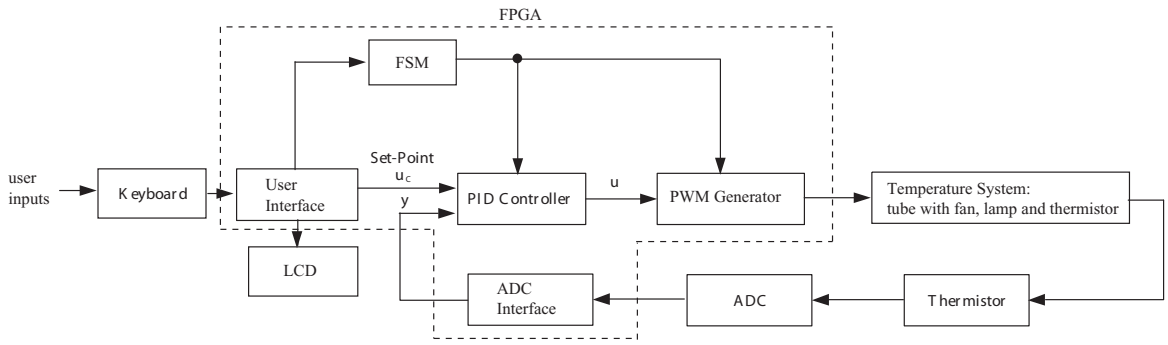

Figure 5: Block diagram of the FPGA-based temperature control system.

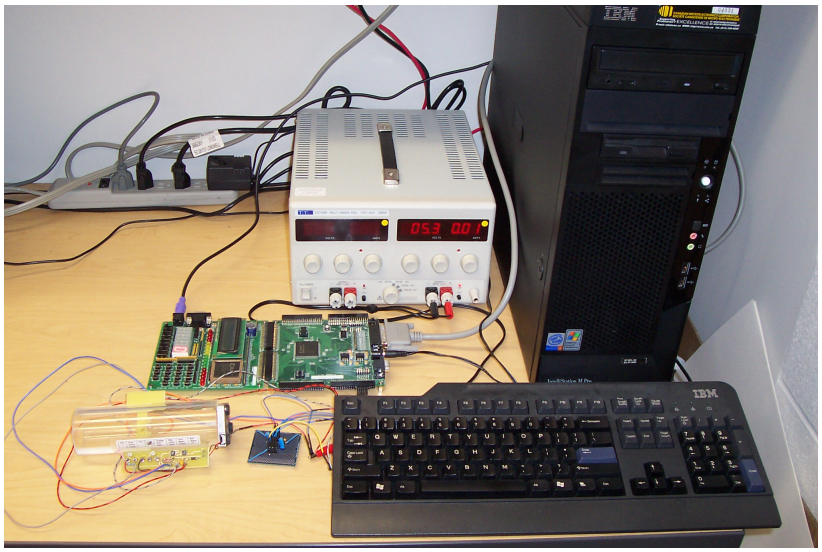

Figure 6: Setup of the FPGA-based temperature control system.

The thermistor in series with a resistor in Figure 4 is used for temperature measurement inside the tube. The voltage v_t across R_t is used to calculate the temperature which is sampled by the ADC to be used in the control law.

Two PWM generators are implemented to control the DC motor and the lamp. The motor and lamp are turned either on or off depending on the PID controller output u. If u is a negative number, the PWM generator for the fan will generate a PWM waveform with appropriate duty cycle, while the PWM generator for the lamp will generate an off pulse. If u is positive, the opposite will be true. In our implementation, u is a 34-bit fixed-point two's complement number. It is converted to a sign and magnitude representation and then scaled accordingly in order to generate the correct PWM output. Both PWM generators run at a clock frequency of $50MHz$ with a PWM period of $20ms$.

The ADC used in this system is a Maxim MAX189 12-bit ADC with an internal reference voltage of 4.096V. The ADC interface polls the ADC every $20ms$ to receive the sampled data. The received 12-bit ADC data is converted into a 16-bit fixed-point format in order to represent the feedback signal y. The 16-bit fixed-point representation contains 4 bits for the integer portion and 12 bits for the fraction portion.

A PS2 keyboard, which is connected to the I/O board, is used for user input. An LCD on the I/O board displays the user's requests entered from the keyboard. The user interface decodes the user's requests and sends appropriate commands to other modules in the system.

For the purpose of comparison, both multiplier-based and DA-based PID controllers are implemented. Each PID controller is integrated with the rest of the system. Considering that general-purpose microprocessors can only accommodate fixed bit-width data representation, FPGA offers an attractive feature of customizable bit-widths [4]. Because of that, binary points have to be tracked accordingly in each computation. For both controllers, the inputs, u_c and y, are both 16-bit fixed-point two's complement numbers with 4 bits for the integer portion and 12 bits for the fraction portion. Due to the bit-serial nature of the DA-based PID controller, parallel-in-serial-out (PISO) shift registers are used to serialize the inputs.

The LUTs of the DA-based PID controller are implemented using the existing 4-input LUTs inside the FPGA slices. The choice of word length in fixed-point arithmetic is crucial in maintaining system stability. Even though a longer word length will reduce quantization effects, it will result in an increase in hardware resources; however, shorter word lengths will affect the controller precision, causing an increase in control error, or destabilization of the system [6]. Therefore, a compromise has to be made between the amount of hardware resources required and the controller precision desired. In this paper, the word length for the fractional part of the bit-width is 12 which was experimentally found to be sufficient for the application being studied. However, the performance of the controller in face of discretization may be further studied using analysis [18] or simulation tools such as the DSP Builder, a development tool that interfaces between the FPGA development software and the MathWorks MATLAB/Simulink tools [11].

2.4 Comparison of DA-based and Multiplier-based Designs

In the following, a comparison is provided between DA-based and multiplier-based design of the PID controller in terms of resource utilization. To this end, both the multiplier-based and DA-based designs were implemented for the 16-bit input case. The DA-based design uses 437 slices and 406 slice FFs, while the multiplier-based design (Spartan-IIE) uses 1142 slices and 327 slice FFs. The equivalent gate count for DA-based design is 16728 and for multiplier-based design is 83796. Thus the DA-based design offers area improvement over the multiplier-based design in the sense that it uses only about 20% of the total equivalent gates as required by the design using multipliers (Spartan-IIE multiplier-based design). The synthesized DA-based PID allows a maximum clock frequency of 47 MHz with 456 mW power consumption and that of the synthesized multiplier-based PID is 15 MHz with 765 mW power consumption. It should be noted that the maximum clock frequency does not represent the computation speed of the control output. Due to the serial nature of the DA method, the DA-based PID controller needs 17 clock cycles, or 361 ns computation time, while the design using multipliers needs 1 clock cycle, or 67 ns. Thus there is a tradeoff between speed and hardware resources. Nevertheless, the speed of the DA-based design (47 MHz) is adequate for typical control loops in industry applications.

Since the system target Xilinx Spartan-IIE FPGA chip does not have hardware multipliers,

the multiplier-based PID controller is compiled towards a Xilinx Spartan-3 FPGA target chip to make a further comparison of hardware utilization. Xilinx Spartan-3 FPGA chip contains embedded 18×18 multipliers which are dedicated multiplier blocks on the FPGA chip. Note that the multipliers used in Spartan-IIE are constant coefficient multipliers (KCMs) while multipliers used in Spartan 3 are dedicated multiplier blocks provided by the chip. It can be seen that the DA-based PID controller offers better area efficiency over multiplier-based design although the speed is lower than the Spartan-IIE and Spartan-3 multiplier-based designs.

3 Complete FPGA-based System Performance

3.1 System Implementation Using Xilinx [13] Platform

The temperature control system is implemented using the FPGA XC2S200E from Xilinx, Inc. The XC2S200E chip contains 4704 Logic Cells (LCs), 1176 Configurable Logic Blocks (CLBs), 56Kbits Block RAM (BRAM), and 146 user input/output blocks (IOBs). The Xilinx ISE Foundation CAD tool is used for the design and development of the FPGA. The FPGA design flow for the system is as follows. First, the system is implemented using the Xilinx ISE foundation tools and simulated at the Register Transfer Level (RTL) to verify the correctness of the design. By using the Xilinx ISE Foundation tools, the logic synthesis is carried out to optimize the design, and the placement and routing are carried out automatically to generate the FPGA implementation file. Finally, the generated implementation file is downloaded to the FPGA development board for testing.

The D2E FPGA development board has a 50 MHz oscillator. A clock generator is used to generate the various clocks for the internal modules. The PID controller, FSM, PWM generators, ADC interface, and the user interface are integrated into a top-level module, compiled, synthesized, and generated into a *bitgen* file, and then downloaded to the Xilinx FPGA. The functional block diagram of the complete system inside the FPGA is illustrated in Figure 7. The testing is conducted on the FPGA board to validate the functions of the whole system as shown in Figure 6.

A further analysis of resource utilization is carried out for both DA-based and multiplier-based PID controllers by including additional components such as FSM, User Interface, PWM Generator, ADC Interface, and other logic modules for comparison purposes. This analysis reveals that while the multiplier-based version utilizes 97% of the FPGA resources, the DA-based utilizes only 70% of the resources, with the savings coming mainly from the DA-based PID controller.

Figure 8 shows the measured waveforms as a result of applying a square wave reference input (changing from 1.75 V to 2.18 V) to the DA-based system. As similar result was obtained for the multiplier-based design. The results indicate that both systems demonstrate good closed-loop performance with the DA-based system using less hardware resources. Figure 9 shows the measured waveform of the thermistor voltage as a result of temporarily heating the tube using a soldering iron in the DA-based system. Both systems demonstrate stable

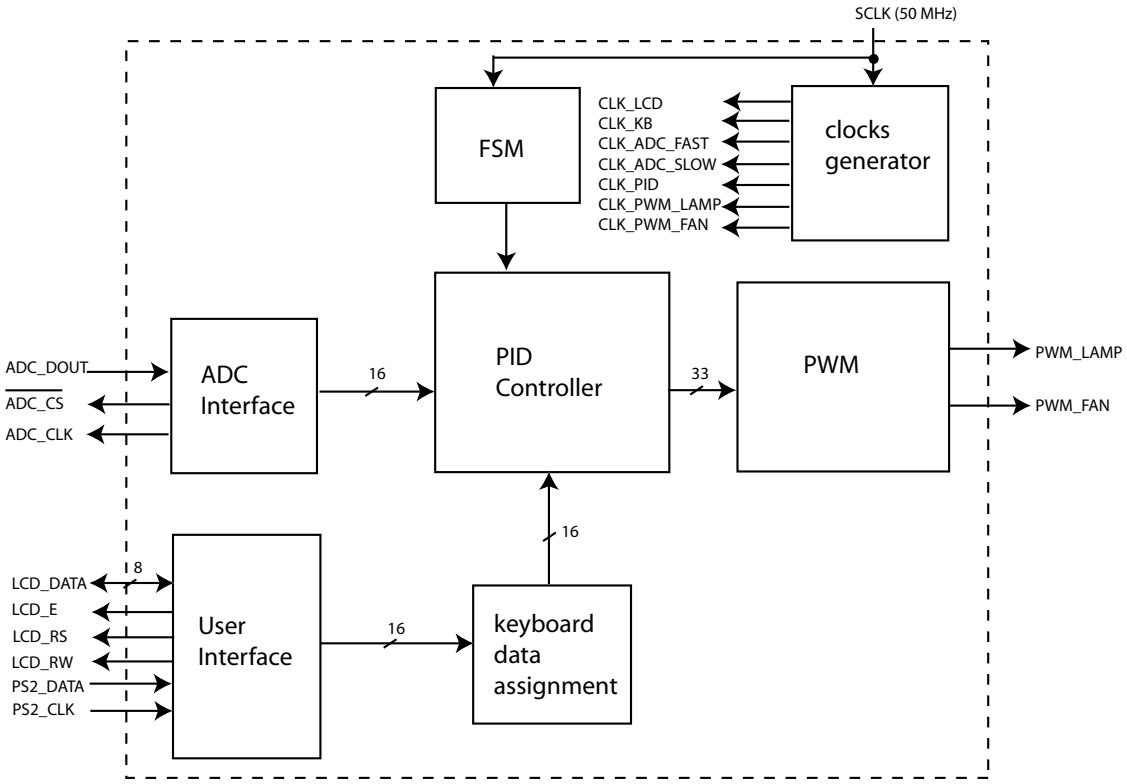

Figure 7: Structure of the temperature control system inside the FPGA with the I/O pin assignments.

regulation of the set point temperature when disturbances are introduced in the system.

3.2 System Implementation Using Altera [11] Platform

An attractive feature of the proposed FPGA design is its programmability. Both versions of the PID controllers as well as the PWM generators can be reused for various applications by simply reconfiguring them with the desired parameters and instantiating the modules into different target systems. In order to show the efficiency of the proposed method, the system is also implemented using Altera Stratix EP1S40 devices [11]. The software used is Quartus II. The results of the Altera implementation are similar to the results for Xilinx implementation in terms of gate usage, speed, and power requirements.

Next, let us compare the proposed DA-based FPGA controller with the NIOS IP core using Altera Quartus II and SOPC builder software [11]. In the next generation of FPGAs, the A/D and D/A converters are built inside the chip. Also, the microcontroller IP core can be used to establish a customized microcontroller inside FPGA as well as a C/C++ design environment. The IP-based system consists of processor, memory and controller, peripherals, and custom logic. In this design, after the SOPC builder configures these components inside the FPGA,

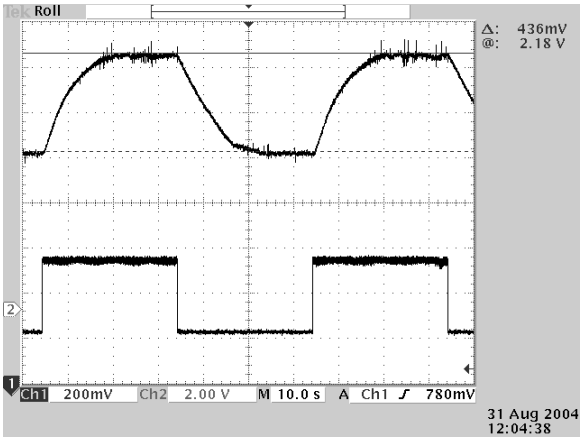

Figure 8: Closed-loop step response of the DA-based Controller System.

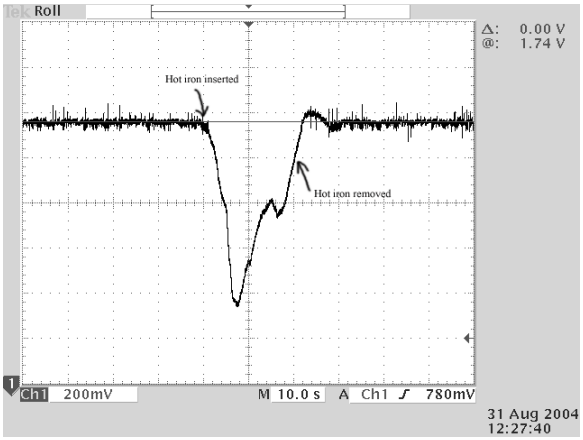

Figure 9: Closed-loop response of the DA-based system due to an external disturbance.

C code will be running on this FPGA's microcontroller to implement the temperature control system. Its implementation results are shown in Table 2. For the purpose of comparison, the Altera FPGA implementation results of the proposed DA-based design are also included in the table. It is seen from this table that the DA-based method is more efficient in terms of hardware resources, speed, and power consumption when compared to the IP-based design. The reason is that the microcontroller IP core consists of many standard components that might not be useful for the PID FPGA system.

Table 2: Comparison of DA-based and IP-based system implementations in Altera.

	Components	Clock frequency	Speed	Power
DA-based	3354 LEs	47MHz	380 ns	498 mW
IP-based	54,033 LEs	50MHz	7650 ns	1235 mW

4 System Reconfigurability and Testing

The modules developed in the previous section can be ported into Matlab/Simulink for hardware/software co-simulation, or they can be integrated into a larger design for rapid prototyping of various control applications. A total of 5 modules were ported into Matlab/Simulink as Simulink blocks, including the DA-based PID controller, PWM generator, FSM, ADC interface, and keyboard interface. The DA-based PID controller and the PWM generator modules are reconfigurable blocks in Matlab/Simulink for different precision and control parameters. This is particularly useful since the user can easily reconfigure these blocks according to the desired controller performance and integrate them into various designs. The other modules including the FSM, ADC interface, and keyboard interface are specific to the implementation of the temperature control system described in the previous section. Therefore, they can be ported into Matlab/Simulink for hardware/software co-simulation or can be used without reconfiguration in other designs.

The DA-based PID controller module has reconfigurable word widths for inputs u_c and y, and output u, as shown in Figures 10 and 11. Since the DA-based PID controller utilizes Xilinx Core Generator modules including the accumulator (ACC) and Random Access Memory (RAM), reconfigurable RAM and ACC blocks are defined in Matlab/Simulink as with their configuration menus not shown for brevity. There are a total of four ACCs and four RAMs in the DA-based PID controller. The four RAMs are used as LUTs and store the contents of Tables such as 1. The accumulator's input word length, input word binary point, and output word length are reconfigurable. The RAM depth, address width, contents, word size, and word binary point can be specified accordingly for the four RAMs. Once the reconfiguration of the Xilinx CORE Generator ACC and RAM modules completes, they are compiled into the *NGC NETLIST* format so that the DA-based PID controller Simulink block can include these NGC files in the final compilation of the controller.

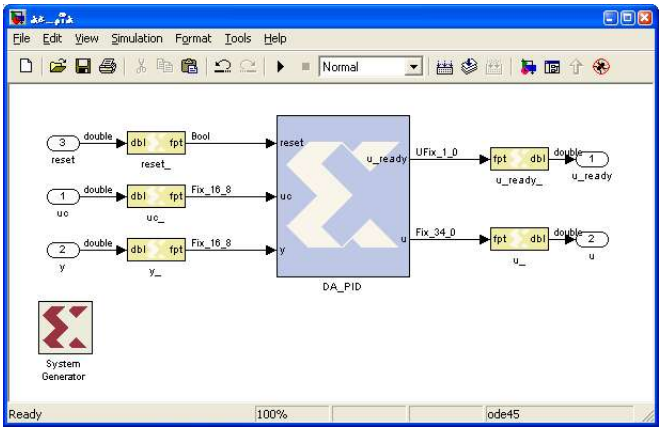

Figure 10: DA-based PID Simulink block.

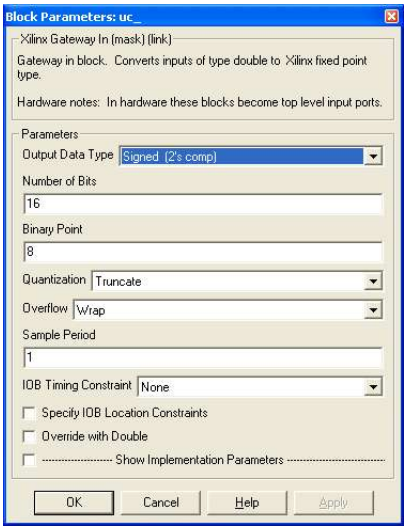

Figure 11: Configuring input u_c in Figure 10.

There are two PWM generator modules i.e., PWM fan and PWM lamp, which are the dual of each other. While the PWM lamp module handles the positive-valued control inputs, the PWM fan module handles the negative-valued inputs. The word length of the reference input is reconfigurable in each PWM generator module.

The FSM, ADC interface, and keyboard interface modules can also be ported into Matlab/Simulink as Simulink blocks. However, they are not reconfigurable modules since they are specific to the application. The modular approach for design allows for testing individual components before implementing and testing the final system on the practical setup. The modules can thus be tested individually and then integrated into the closed-loop system.

5 Conclusions

In this paper, an FPGA-based digital feedback control system using a novel DA-based PID controller is presented. Based on the LUT scheme, the proposed PID controller reduces

the cost of the FPGA design. Also, due to the flexibility of the LUT in the FPGA, this FPGA-based PID controller can be easily extended to incorporate other algorithms such as antiwindup protection or adaptive schemes. This design approach would specifically be suitable for the next generation of FPGA chips, in which A/D and D/A converters are built inside the chip. In such a case, the proposed structure would be more efficient in terms of hardware resources and control performance compared with the standard micro-controller IP cores.

The complete system was designed using a modular approach and integrated and downloaded into both Xilinx and Altera FPGA chips. The system was tested by simulations and experiments, demonstrating good closed-loop stability and performance. The controller modules are reusable and reconfigurable, which can be ported into Matlab/Simulink as Simulink blocks for hardware/software co-simulation and can be utilized in other embedded control applications.

References

[1] K.J. Astrom and B. Wittenmark, *Computer Controlled Systems*, Prentice Hall, New Jersery, USA, 1997.

[2] L. Samet, N. Masmoudi, M. W. Kharrat, and L. Kamoun, "A digital *PID* Controller for Real-time and Multi-loop Control: A Comparative Study," *IEEE Int. Conf. on Electronics, Circuits and Systems,* Vol. 1, pp. 291-296, Sept. 1998.

[3] W. Wolf, *Computers as Components: Principles of Embedded Computing System Design*, San Francisco, Morgan Kaufman, 2001.

[4] F. Krach, B. Frackelton, J. Carletta and R. Veillette,"FPGA-Based Implementation of Digital Control for a Magnetic Bearing," *American Control Conference,* Vol.2, pp. 1080-1085, June 2003.

[5] S.L. Jung, M.Y. Chang, J.Y. Jyang, L.C. Yeh, Y.Y. Tzou, "Design and Implementation of an FPGA-Based Control IC for AC-Voltage Regulation," *IEEE Trans. on Power Electronics,* Vol. 14, pp. 522-532, May 1999.

[6] S. Ferreira, F. Haffner, L.F Pereira, F. Moraes, "Design and Prototyping of Direct Torque Control of Induction Motors in FPGAs," *IEEE Symposium on Integrated Circuits and Systems Design,* pp. 105-110, Sept. 2003.

[7] P. Zumel, A. de Castro, O. Garcia, T. Riesgo, J. Uceda, "Concurrent and Simple Digital Controller of an AC/DC Converter with Power Factor Correction," *IEEE Applied Power Electronics Conference and Exposition*, vol. 1, pp. 469-475, 2002.

[8] F. Ricci, H. Le-Huy, "An FPGA-Based Rapid Prototyping Platform For Variable-Speed Drives", *IEEE Industrial Electronics Society Annual Conference*, vol.2, pp. 1156-1161, Nov. 2002.

[9] L. Charaabi, E. Monmasson, I. Slama-Belkhodja, "Presentation of an Efficient Design Methodology for FPGA Implementation of Control Systems: Application to the Design of an Antiwindup PI Controller," *IEEE Industrial Electronics Society Annual Conference,"* Vol. 3 , pp. 1942 - 1947, Nov. 2002.

[10] R. Chen, L. Chen and L. Chen, "System Design Consideration for Digital Wheelchair Controller," *IEEE Trans. On Industrial Electronics,* vol. 47, pp. 898-907, Aug. 2000.

[11] Altera Flex10K Embedded Programmable Logic Family Data Sheet, Altera Corp., 2003: http://altera.com.

[12] R. Ruelland, G. Gateau, T. A. Meynard and J-C Hapiot, "Design of FPGA-based Emulator for Series Multicell Converters using Co-simulation Tools", *IEEE Transactions on Power Electronics*, Vol. 18, No. 1, pp. 455-463, Jan. 2003.

[13] Xilinx Inc., Spartan IIE Data sheet: http://www.xilinx.com.

[14] Moallem M., "A Laboratory Testbed for Embedded Computer Control", *IEEE Transactions on Education*, Vol. 47, No. 3, pp. 340-347, 2004.

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

[16] A. White, "Application of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review," *IEEE Accoustic, Speech, and Signal Processing Magzine,* vol. 6, pp. 4-19, 1989.

[17] Digilent Inc.: http://www.digilentinc.com.

[18] G. F. Franklin, J. D. Powell and M. L. Workman, *Digital Control of Dynamic Systems,* Addison-Wesley, New-York, 1990.