Volume 3, Issue 5, May 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Fast Multiplication Algorithm for PID Controller

Mayank Neema* *RKDF Bhopal India.*
Murli manohar hinnwar *RKDF Bhopal India.*

Abstract— : In this paper implementation of digital PID controller FPGA using different multiplication algorithm is presented. Now a day's embedded control applications requires low power, occupy less area and fast acting PID controllers with a closed loop performance using less resources, resulting in cost reduction. In digital PID controller error signal is generated by using comparator which is analog in nature. By using ADC it is converted in to digital. Digital output of FPGA is converted in to analog signal to drive any system using DAC, but potential output ripple in the baseline system due to quantization and other related errors. The controller algorithm is synthesized, simulated using Xilinx Spartan3e xc7a100t-3-csg324 board with Xilinx ISE 14.1 as a simulator.

Keywords— FPGA, PID, Verilog, Multiplayer

I. INTRODUCTION

MANY machines and processes require some form of control to improve their basic dynamic performance. Typically, this is achieved by some controlling device, which provides an input to the controlled system as a function of the error between the desired and actual system output. . Electronic circuits are used for implementing controllers owing to their low cost, high reliability, small size, and large flexibility. There are four different approaches for the implementation of controllers.

1) The controller functionality can be implemented exclusively in the analog domain using one or more operational amplifiers and a number of passive components (e.g., resistor or capacitors).
2) The design of the controller can be based on a microprocessor [1], a microcontroller [2], or a digital signal processor (DSP) [3], in which the controller functionality is implemented by means of software. This approach has become very popular in control applications. The prime reason is the functionality that can be implemented with the use of software, so that advanced and complex control algorithms can be executed. The software approach is generally more flexible by offering the possibility to modify the functionality in a cost-effective and easy way.
3) The controller can be implemented with an application- specific integrated circuit (ASIC) and is known as monolithic controller. The functionality of these integrated circuits (ICs) is specified by the user. Once the ASIC is hardwired for some specific controller architecture, it cannot be changed (only the controller parameters can be programmed). Controllers implemented this way are described in this paper. This approach offers advantages over the conventional software approach, which include increased processing speed, reduced size and reduced power consumption. Any of these advantages may justify adopting this approach (e.g., the size reduction achieved by using ASIC devices was the driver for the use of ASICs in the avionics suite of the Boeing 777 [4]).
4) The controller can be implemented with a field-programmable gate array (FPGA), which also allows modern complex control laws to be implemented in hardware. The FPGA allows us to change the controller architecture and to program the controller parameters. The drawback of this approach is that the power consumption of the FPGA is higher than the power consumption of the ASIC.

In general, a microsensor system comprises sensitive materials(e.g., metal oxide and polymer), transducers (e.g., cantilever and silicon oxide/nitride membrane), actuators (e.g., thermal bimorph and resistive heater), readout electronics (e.g., amplifiers, filters, and analog-to-digital converters), controller architectures (e.g., temperature controllers, deflection controllers, and position controllers), and an interface unit to handle the communication with the external world. The general microsystem requirements include low cost and high reliability. A CMOS monolithic microsystem meets these requirements since CMOS technology is cheap and reliable. Furthermore, monolithic solutions avoid the use of external components thereby reducing size and cost. Monolithically integrated inter-face circuits additionally reduce packaging efforts, the number of input/output pins, and provide more reliable electrical signals than a multichip solution with transducers and electronics on different chips. Therefore, the third design approach, i.e., the use of ASICs for controller implementation, meets the requirements for most microsystems, so that this approach was selected for the implementation of monolithic controllers that were used in smart sensors comprising micromembranes and microcantilevers. Monolithically integrated controllers can be implemented with analog or digital components. The advantages and disadvantages of both implementations will be discussed in Section II. There is a wealth of literature on how to design controllers for various purposes. These control techniques can be roughly classified as follows.

Classical Control [5]: This technique deals primarily with linear, constant coefficient systems (also known as linear timeinvariant systems). Few real systems are exactly linear over their whole operating range, and few systems have parameter values that are precisely constant over longer time spans. However, many systems are approximately linear in a sufficiently narrow operating range. The treatment of linear, stationary systems is greatly simplified by the use of transform techniques and frequency- domain methods (e.g., Bode plots and Nyquist plots). However, transform techniques are mainly used for single-input/
single-output (SISO) systems since in multiple-input/multiple output (MIMO) systems the decomposition of the transfer functions may not be possible owing to their coupled nature. Continuous-time systems use the Laplace transform whereas discrete- time systems use the Z-transform. The most popular classical control techniques are proportional (P) control, proportional- integral (PI) control, proportional-derivative (PD) control, and proportional–integral–derivative (PID) control, which was developed in the 1940s and used for the control of industrial processes and servomechanisms (e.g., chemical plants and commercial airplanes)

2) Optimal Control [6]: This technique is used when specific performance or cost criteria (e.g., time or energy) must be minimized. The given criteria or cost function is used to derive an appropriate control law, which is then implemented with a controller. The state-space system representation is used in the calculation of the controller. Linear quadratic regulator
(LQR) control, linear quadratic Gaussian (LQG) control, and the most general H2 control are examples of the optimal control technique developed in the 1960s and promoted by the NASA Apollo project.

3) Robust Control [7]: The controller is designed with due consideration of both the nominal model of the system to be controlled and the characterization of model uncertainties and disturbances. The system to be controlled is usually denoted as plant. H control developed in the 1980s and the 1990s is a good example of the robust control technique.

4) Nonlinear Control [8]: The controller is designed to handle the nonlinearities of a system in a large operation range or to handle systems with discontinuous nonlinearities that do not allow for linear approximation.

5) Intelligent Control [9]: This technique is used in systems with insufficient information about the plant parameters, making it impossible to derive a plant model. It is also used in systems where plant parameters or plant models change over time. Knowledge-based control, adaptive control, and fuzzy control developed in the 1990s are good examples of this technique. In summary, a suitable control technique is selected depending on the plant dynamics, the control goals, and the controller implementation (i.e., monolithic implementation, FPGA, and DSP). The goal of this paper is to develop controllers for CMOS-based sensors, the system variables of which (e.g., temperature and deflection) need to be regulated on-chip. Micromembrane- and microcantilever-based systems are time-invariant SISO systems and allow for linear approximation. Therefore, classical control techniques like P control (see Section III) and PID control (see Section IV) were chosen to regulate these systems because they are suitable for effective control and can be realized as monolithic implementations.

II. ANALOG VERSUS DIGITAL IMPLEMENTATION

Monolithic classical controllers can be implemented in an analog or digital way. Here, some important implementation issues in analog and digital controllers will be discussed.

A. Implementation Issues in Analog Controllers

Analog controllers provide continuous processing of the signal, and they can be used for large-bandwidth systems. They also provide almost infinite resolution, thus ensuring precise
control. Analog controllers have been around for a long time, and there is a great deal of literature, practical experience, and design methods available. Analog controllers are implemented using one or more operational amplifiers and a number of components (e.g., resistors or capacitors). If they are integrated on a single chip (without using external components), their performance is limited by the CMOS component specifications, i.e., resistors of some hundreds of kilo-Ohms and capacitors of some tens of pico-Farads. Furthermore, analog controllers suffer from component aging, component process variations,
and temperature drifts. Therefore, a perfectly designed controller will start to exhibit undesired characteristics if such effects are not taken into account during the design. Analog
controllers are also limited to simple control algorithms from classical control theory like P controllers, PID controllers, or lead-lag compensators. The number of components (such
as capacitors, resistors, or operational amplifiers) increases with the order of the analog controller. Consequently, both its reliability and accuracy decrease.

B. Implementation Issues in Digital Controllers

Digital controllers sample the signal at discrete time intervals, which limits the bandwidth that can be handled by the controllers. Digital controllers can be programmed, which renders them flexible and easy to upgrade. The implementation of a low order digital controller is as reliable as the implementation of a high-order digital controller, but

they both require analog-to digital and digital-to-analog converters plus anti-aliasing and smoothing filters, the reliability of which is critical in any digital implementation. A number of issues have to be considered.when implementing a digital controller in hardware [10]. Digitalcontrollers can be implemented using both classical and modern control methods. All linear time-invariant digital controllers can be thought of as just linear time-invariant digital filters. The problem of implementing a digital controller is equivalent to the problem of implementing digital filters, which is a well-studied issue in signal processing. Hardware considerations include. sample rate, number system, word length, arithmetic operator behavior, and controller structure.

1) Sample Rate [11]: The minimum sample rate is the Nyquist rate, which is twice the highest frequency of the analog input signal. Undersampling introduces the phenomenon of signal aliasing where a high-frequency component appears as a low-frequency component. In practice, oversampling above the Nyquist rate (approximately ten times the Nyquist rate), is recommended because it relaxes the specifications of the low-pass anti-aliasing filter that precedes the analog-to-digital converter (ADC) at the input of the digital controller.

2) Number System: The number system is another choice that is often overlooked. Floating-point number systems offer a large dynamic range due to their exponential representation, which is useful if data values become very large or small in magnitude. On the other hand, floating-point operations require large amounts of hardware. In FPGA or on-chip-based filter architectures, it will be very expensive to implement floating-point operators (large silicon area). Fixed-point number systems require comparably less resources for operations like addition and multiplication. The disadvantage is the limited dynamic range, but if data values are bounded and the design takes into account this limitation, then this is no longer a problem.

3) Word Length: The selection of the word length has significant effects on the silicon area. In digital hardware, the adder area scales linearly with the word length while the multiplier area scales quadratically with the word length. Large word lengths allow a number representation to approach a continuous-value system, so that numerical quantization effects
are reduced, and the closed-loop performance more closely matches that of the designed controller. The goal is to find the minimum word length, which provides acceptable closed-loop performance. This can be done through closed-loop system simulations with finite word lengths. The minimum word length as found in the simulations is then used in the hardware
implementation.

4) Arithmetic Operator Behavior: Given a number system and finite word lengths, the behavior of arithmetic operators is another issue. Addition operations and especially multiplication operations produce results that require a larger word length for accurate representation. Normally, each result is rounded in some manner at the least significant bit and truncated or limited at the most significant bit. Arithmetic rounding can be modelled as a noise source after each operation and can often be tolerated. Truncation, on the other hand, results in overflowconditions and can have drastic effects on the closed-loop stability. Overflow conditions should be avoided whenever possible.

5) Controller Structure: In SISO systems, there is a variety of well-studied filter architectures that minimize quantization effects [11]. These architectures take advantage of cascading sections of the filter transfer function thereby reducing sensitivity to coefficient perturbation and overflow probability. Decomposition of the transfer functions may not be possible due to the coupling nature of MIMO systems. Instead, a direct implementation of the state-space form must be chosen with some variation in the matrix elements to alter the implemented structure. In fact, the cascaded SISO forms may be expressed in sparse state-space forms thus rendering the state-space representation the most generic.

C. Implementation of Digital PID control :
Proportional-Integral-Derivative (PID) control is still widely used in industries because of its simplicity. No need for a plant model. No design to be performed. The user just installs a controller and adjusts 3 gains to get the best achievable performance. Most PID controllers nowadays are digital. In this document we discuss digital PID implementation on an embedded system. We assume the reader has some basic understanding of linear controllers as described in our other document.

Different forms of PID
A standard equation of PID controller is

$$U(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \qquad (2.1$$

where the error e(t), the difference between command and plant output, is the controller input, and the control variable u(t) is the controller output. The 3 parameters are K (the proportional gain), Ti (integral time), and Td (derivative time).

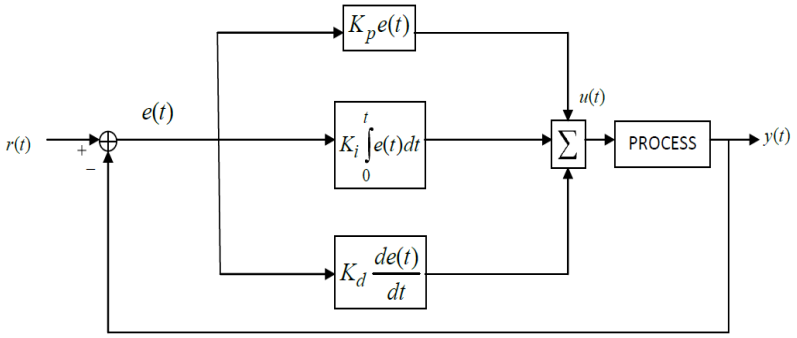

Fig.1: Implementation of Continuous time PID control

Performing Laplace transform on equation 2.1, we get

$$G(s) = K\left(1 + \frac{1}{ST_i} + ST_d\right) \tag{2.2}$$

Another form of PID that will be discussed further in this document is sometimes called a parallel form.

$$U(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{2.3}$$

With its Laplace transform

$$G(s) = K_p + \frac{K_i}{S} + SK_d \tag{2.4}$$

We can easily convert the parameters from one form to another by noting that

$$K_p = K$$
$$K_i = \frac{K}{T_i}$$
$$K_d = KT_d$$

Discrete-time PID Algorithm

For digital implementation, we are more interested in a Z-transform of equation 2.4,

$$U(z) = \left[K_p + \frac{K}{1 - Z^{-1}} + K_d(1 - Z^{-1})\right] E(z) \tag{2.5}$$

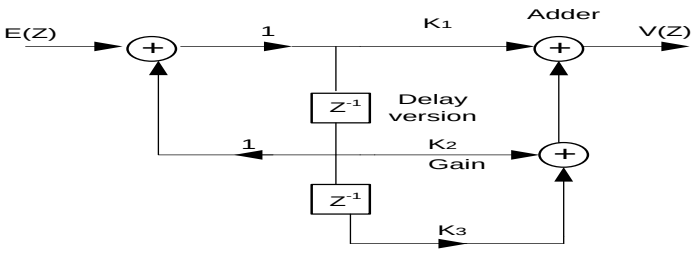

Fig.2 Implementation of Discrete time PID control

$$U(z) = \left[\frac{(K_p + K_i + K_d) + (-K_p - 2K_d)Z^{-1} + K_d Z^{-2}}{1 - Z^{-1}}\right] E(z) \tag{2.6}$$

Assume that

$$K_1 = K_p + K_i + K_d$$
$$K_2 = K_p - 2K_i$$
$$K_3 = K_i$$

Equation 2.6 can then be rewritten as

$$U(z) - Z^{-1}U(z) = [K_1 + K_2Z^{-1} + K_3Z^{-2}]E(z) \quad (2.7)$$

which then converted back to difference equation as

$$U(k) = U(K-1) + K_1e[k] + K_2e[K-1] + K_3e[K-2] \quad (2.8)$$

a form suitable for implementation. We assume that the plant output is returned from a function read ADC, and the control variable u is outputted using write DAC.

III. Proposed Work

Generally for digital signal multiplication on FPGA digital signal processor used which is based on conventional multiplier occupy more area on FPGA and produce undesired delay which affect PID time response. To solve above problem we apply here booth multiplication method to reduce hardware and delay in the system.

43.1 Different Multiplication method

Multiplication Method	Logic	Delay(ns)	Normalized Delay	Area (LUT)	Normalized Area
Conventional Method	Combinational	6.483ns	1.10	550 LUT	1.612
Radix-4 or Booth 2	Combinational	5.865ns	1	341 LUT	1
Radix-8 or Booth 3	Combinational	5.905ns	1.0068	575 LUT	1.686
Radix-4 or Booth 2	Sequential	7.118ns	1.213	432 LUT	1.26

3.2 PID based on Different Multiplication Algorithm

Multiplication Method	Logic	Normalized Delay	Area	Normalized Area
Conventional Method	Combinational	1.10	1681 LUT	1.594
Radix-4 or Booth 2	Combinational	1	1054 LUT	1
Radix-8 or Booth 3	Combinational	1.0068	1556 LUT	1.476
Radix-4 or Booth 2	Sequential	1.213	1327 LUT	1.259

IV. CONCLUSION

The most popular approaches for the implementation of Digital PID controllers were discussed and four multiplication algorithms were reviewed in this paper. it is observed that conventional multiplication method occupy10% larger area on chip and produce 60% high delay large delay as compare to radix-4 approach. i.e. radix-4 approach is much than conventional method. when we apply this approach on PID then get better improvement on rise time, peak overshoot time and setting time.

References

[1] K. Warwick and D. Reeds, *Industrial Digital Control Systems*. London, U.K.: Peter Peregrinus, 1988.
[2] D.Wilson, "16-bit DSP servo control with MC68HC16Z1," *Microprocessors Microsyst.*, vol. 18, no. 2, pp. 109–117, 1994.
[3] K. Astrom and H. Steingrimsson, "Implementation of a PID controller on a DSP," *Texas Instruments Inc. Digital Signal Process. Products*, 1990.
[4] G. Norris, "Inside the 777," *Flight Int.*, pp. 23–26, Apr. 1994.
[5] G. Franklin, J. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Reading, MA: Addison-Wesley, 1994.
[6] J. B. M. Anderson, *Optimal Control: Linear Quadratic Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1990.
[7] R. Sanchez-Peña and M. Sznaier, *Robust Systems Theory and Applications*. New York: Wiley,1998.
[8] J.-J. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Clifs, NJ: Prentice-Hall, 1991.
[9] D. A. White and D. A. Sofge, *Handbook of Intelligent Control Neural, Fuzzy, and Adaptive Approaches*. NewYork: Van Nostrand Reinhold, 1992.
[10] H. Hanselmann, "Implementation of digital controllers—A survey," *Automatica*, vol. 23, no. 1, pp. 7–32, 1987.
[11] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ:Prentice-Hall, 1989.

