IMPLEMENTATION OF PID AUTO-TUNING CONTROLLER USING FPGA AND NIOS II PROCESSOR

RAPHAEL C. GOMEZ, EDSON A. BATISTA, LUIS HENRIQUE G. CORBELINO, CRISTIANO Q. ANDREA, ALEXANDRE C. R. DA SILVA, MARCO H. NAKA. ALEXSANDRO M. CARNEIRO.

Laboratory of Control and Automation, Department of Engineering, Dom Bosco Catholic University
Av. Tamandaré 6000, 79117-900 Campo Grande, MS, Brazil
E-mails: raphaelceni@gmail.com, edson.ucdb@gmail.com, luis.corbelino@gmail.com, cristiano@batlab.ufms.br, acrsilva@gmail.com, marco.h.naka@gmail.com, alexsandro.ucdb@gmail.com

Abstract— FPGAs devices are becoming faster and cheaper every day. Its fast implementation has been increasing its popularity, and modern control studies can be beneficed with its advantages. In this paper is presented a project of an auto tuning PID implemented in FPGA with a NIOS II processor, using Relay Feedback and Ziegler-Nichols methods.

Keywords— PID, FPGA, Ziegler-Nichols, Relay Feedback, Auto Tuning.

1 Introduction

Developing more complex and higher precise control systems, reducing time in manufacturing and the improvement of quality have always been the main focus in technological researches

From this point of view, the analysis of closed-loop control systems is essential, because it is possible to design controllers based on the plant's behavior in order to obtain a desired output. The PID controller is used to solve that problem. It is well accepted because of its easy implementation and for providing control of different kinds of processes.

PID works with a mechanism of control with a feedback, extensively used in industrial applications in order to minimize the error or eliminate it in second order systems. Another advantage is the fact that it increases the system's performance. Nowadays, more than 90% of feedback systems use PID or PI controllers, which can be found at any area where controllers are used (Aström and Hägglund, 2001). Some examples of PID's applications are: control of robotic arms in automotive industries, temperature control in petrochemical industries, or even pressure control in a production line.

As a result, a great variety of tuning methods has been developed for these controllers. The classic approach of Ziegler-Nichols method with the Relay-Feedback method makes possible to implement an auto-tuning PID controller. Other methods for tuning a PID controller are: Fuzzy, Root Locus, CHR (Chien, Hrones and Resqick, 1952), Cohen-Coon (Cohen and Coon, 1953), Visioli (Visioli, 2001), Ya-Gang (Wang and Cai, 2002) and also the IMC (Internal Model Control) method.

The auto-tuning is expected to identify the output of the system and automatically find the parameters of the controller. The Ziegler-Nichols method is practical and direct while the Relay-Feedback method is utilized to obtain the data that will be used for the Ziegler-Nichols tuning. It is expected that the results will be more accurate with the combination of both methods.

About the equipments, the PLCs (Programmable Logic Controller) are the most common hardware used as a controller in industry. These devices were developed around 1960 with the aim to replace relays in logic control. The PLCs have a closed architecture and the programming is carried out using the language ladder or relay diagrams. Another attractive option to satisfy the needs of industrial control and automation are the programmable logic devices.

The programmable logic devices were developed more than a decade ago and, nowadays, are used in several areas such as, telecommunication, instrumentation and in control. In comparison to an analogical controller, it has some advantages that are essential to obtain the desired result, as better noise immunity owing to its discrete signal (Ruschel 1996), it is easily integrated with other digital systems and it makes possible the implementation of more sophisticated control methods.

The FPGA (Field Programmable Gate Arrays) is a reconfigurable logic device that provides mainly practicality and portability, with low consumption of energy, high speedy of operation and large capacity of data storage.

The FPGA, contrary to TTL technology integrated circuits that have a fixed logic, allows the designer to make future modifications. The reconfigurable logic devices are synthesized by means of hardware description. This description can be developed with a standard language, for example, VHDL (VHSIC – Hardware Description Language).

In this paper, it is described the development of the Relay-Feedback and Ziegler-Nichols methods for the PID auto-tuning synthesized in a FPGA from Cyclone family of (Altera) with NIOS II processor.

This paper has the objective to make it easier and faster to develop automation systems using FPGA by developing a template that can be easily

Table 1. Ziegler-Nichols table for PID tuning.

Controller	Kc	Ti	Td
P	$0{,}5 \cdot Ku$	-	-
PI	$0{,}4 \cdot Ku$	$0{,}8 \cdot Tu$	-
PID	$0{,}6 \cdot Ku$	$0{,}5 \cdot Tu$	$0{,}12 \cdot Tu$

configured for future implementations. This template can be used in systems that already have FPGA devices or in future projects that need controllers, where the developer designs the application and uses the PÍD auto-tuning template for the controller.

The paper is organized by the following topics: In section 2 is presented the techniques of auto-tuning for PID, utilizing the Ziegler-Nichols and Relay-Feedback methods. Section 3 shows the control algorithm used in the NIOS II processor and the development of the hardware used to test the technique proposed. Finally, simulations of the controlled system in MATLAB and SIMULINK, and the results obtained are presented in section 4.

2 Auto-Tuning PID

The PID is the sum of the proportional, integral and derivative controllers. Considering second order systems, the Proportional (P) controller reduces the time of settlement, but does not eliminate the state error. The Integral (I) controller eliminates the state error, but may make the transitory output worse while the Derivative (D) controller increases the system's stability, reducing the transitory oscillations, the overshoot and improving the transitory output but it also amplifies high frequency noise (Nise, 2002).

2.1 PID Tuning

There are different techniques for tuning a PID controller. The Ziegler-Nichols method is well known for being the first to propose a simple and objective methodology for PID tuning.

The Ziegler-Nichols method proposes two techniques: in the first one, the proportional gain is increased until the closed-loop system's output has a constant period. That way, the ultimate gain Ku and the oscillation period Pu are determined. Any gain greater than Ku makes the systems unstable. The controller's tuning is obtained in Table 1. The second technique consists in applying an open-loop test. A step disturbance is generated at the controller's output, and by the system's answer to that disturbance, the delay or dead-time and the variation rate are calculated and used ahead for the controller's tuning.

In this paper, two methods are utilized. The Relay-Feedback proposed by (Aström and Hägglund, 1995) uses relays in a closed-loop system to provoke controlled oscillations in a determined process and then, to estimate the period Pu, which is approximately the same as the constant period. At last, the ultimate gain Ku is estimated, witch is related to the relay's output "h" and the plant's output "a" as seen in Equation (1).

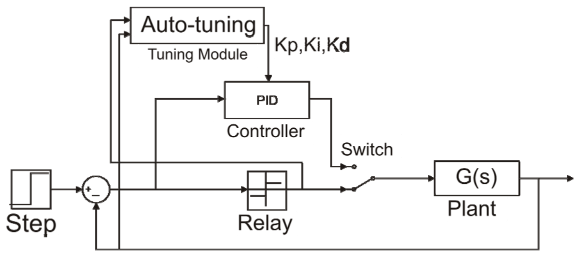


Figure 1. Scheme of an auto-tuned PID.

$$Ku \cong \frac{4 \times h}{a \times \pi} \tag{1}$$

In that way, it is possible to obtain a controller using the two methods together.

When using the FPGA, the period Pu is easily obtained by setting a counter on the simulated relay. This can be seen well in the next section.

The Equation (2) corresponds to a PID controller.

$$C(s) = K_p(1 + \frac{1}{T_i s} + T_d s) \tag{2}$$

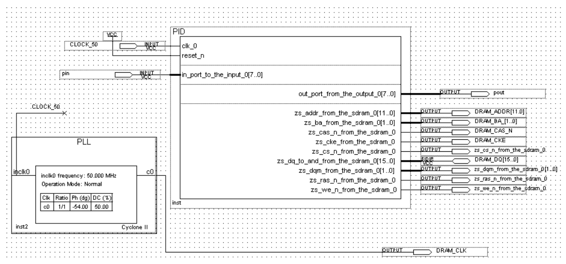
This controller is supposed to be used in an industrial environment. For this reason, some adjustments need to be done. The derivative term can cause some problems on a real system considering that it uses the derivate of the error. That way it can generate some excessive control actions that may compromise the system. Because of that, instead of using the derivate of the error, it is used the derivate of the system's output. This can be seen in the next section.

2.2 Auto Tuning

The Auto-tuning PID utilized in this paper is basically a controller which the tuning is done by means of Ziegler-Nichols and Relay-Feedback methods. The hardware used as controller is responsible for analyzing the plant's output, simulating the Relay-Feedback's method, then calculate the parameters necessaries as Ku and Pu, then tune the PID controller using Ziegler-Nichols. It is relevant to consider that this is an auto-tuning controller, and not an adaptive, which means that if there is any change on the plant, it will not do a new tuning of the controller, unless it has been reset. A PID auto-tuning system with relay feedback is seen in Figure 1.

For implementing the PID into the FPGA device and inserting the logic into the NIOS II processor, it is necessary to obtain an equation that represents a digital PID.

The digital structure of a PID is represented by Equation (3), u(t) is the control signal, e(t) is the



PID
VCC
CLOCK_50
INPUT
clk_0
reset_n
pin
in_port_to_the_input_0[7..0]
out_port_from_the_output_0[7..0]
OUTPUT
pout
CLOCK_50
zs_addr_from_the_sdram_0[11..0]
zs_ba_from_the_sdram_0[1..0]
zs_cas_n_from_the_sdram_0
zs_cke_from_the_sdram_0
zs_cs_n_from_the_sdram_0
zs_dq_to_and_from_the_sdram_0[15..0]
zs_dqm_from_the_sdram_0[1..0]
zs_ras_n_from_the_sdram_0
zs_we_n_from_the_sdram_0
DRAM_ADDR[11..0]
DRAM_BA_[1..0]
DRAM_CAS_N
DRAM_CKE
DRAM_DQ[15..0]
BIDIR
PLL
inclk0
c0
inclk0 frequency: 50.000 MHz
Operation Mode: Normal
Clk | Ratio | Ph (dg) | DC (%)
c0 | 1/1 | -54.00 | 50.00
inst2
Cyclone II
inst
DRAM_CLK

Figure 2. Configuration of the embedded module on Quartus II

system's error, Ts is the system's time sample and q0, q1 and q2 are related to Kc, Ti and Td according to Equations (4), (5) and (6).

$$u(t) = u(t-1) + q0 \times e(t) + q1 \times e(t-1) + q2 \times e(t-2) \quad (3)$$

$$q0 = kc \cdot \left(1 + \frac{Ts}{2 \cdot Ti} + \frac{Td}{Ts}\right) \quad (4)$$

$$q1 = -kc \cdot \left(1 - \frac{Ts}{2 \cdot Ti} + \frac{2 \cdot Td}{Ts}\right) \quad (5)$$

$$q2 = kc\left(\frac{Td}{Ts}\right) \quad (6)$$

The equations for q0, q1 and q2 are found substituting the derivate by the first order difference and using the trapezoidal approximation for an integral.

Instead of using the error for the derivative term, it is used the system's output to prevent excessive control action. Equation 7 represents the proportional and integral error, and Equation 8 represents the derivative error.

$$e(t) = y_{ref}(t) - y(t) \quad (7)$$

$$e_d(t) = -y(t) \quad (8)$$

Equation 9 represents the new PID equation, or PI+D.

$$u(t) = u(t-1) + Kc \cdot \{-y(t) + y(t-1) + \quad (9)$$
$$+ \frac{Ts}{Ti} \cdot e(t-1) + \frac{Td}{Ts} \cdot [-y(t) + 2y(t-1) - y(t-2)]\}$$

As the prototyping of the proposed system was synthesized using FPGA, in the next section the hardware and the programming is described.

3 FPGA-NIOS II

FPGA is a reconfigurable logic device which provides a fast prototyping. One advantage in hardware development synthesized in FPGA is the utilization of the NIOS II processor. The applicability of the NIOS II processor favors the development of embedded systems to work in the industrial automation sector. In this paper, a hardware was developed in order to allow the testing of an execution of a PID controller. The system developed in this paper can be used, for example, to control the speed or position of a DC motor.

In Figure 2, it is shown the hardware architecture, which includes a 50MHz CLOCK that provides the system with a capacity of making all the operations needed without compromising the controller. An 8bits input and output, which may be reconfigured accordantly to the AD and DA available. An 8Mbytes SD-RAM memory, used to store the code responsible and the variables to be computed and a PLL which adjust the 50 MHz CLOCK for the SD-RAM.

Table 2. Resources utilized to generate prototyped hardware.

Total logic elements	Dedicated Logic Register	Memory Bit	Total Pins
9%	5%	6%	12%

3.1 Hardware Configuration

The prototyping was done using the DE2 board from Altera. This board has an EP2C35F672C6, from the CYCLONE II family.

The hardware that will be used is configured using QUARTUS II software.

The hardware developed in this project is composed by:

- CPU - 50 MHz NIOS II/s processor.
- SDRAM - 8 Mbytes SDRAM memory.
- SYS_CLK_TIMER - 1 μs internal timer module.
- SYSID - peripheral identification system module.
- JTAG_UART - USB communication port.
- INPUT - 8 bits input.
- OUTPUT - 8 bits output.

The main components are selected and configured at SOPC Builder and then attached to the peripherals in QUARTUS II. With the hardware generated at SOPC Builder and the compilation in Quartus II, the embedded module is ready to receive the operation logic. The operation logic is programmed using C/C++ language in NIOS II IDE environment. The configuration of the embedded module can be seen with the respective I/O in Figure 2. The resources used to generate the embedded module are presented in Table 2.

3.2 NIOS II processor configuration

The configuration of the embedded NIOS II processor, present in the FPGA is done using the NIOS II IDE software. This software allows designers to program using C/C++ language. In this paper, a library with functions that allows the automatic generation

```
/*
   Closed-Loop-Relay - method used to turn the output
   into a constant frequency output.
---------------------------------------------------------
    pin = input of the embbebed module
    step = Amplitude of the step input
    pout = output value
    lr = lower range value
    hr = higher range value
---------------------------------------------------------*/

    do
    {   pin=IORD(INPUT_0_BASE,0);

        if (step-pin< 0)
        {
           pout=lr;
           Pu-counter();
           restart_counter();    // Function that reset the
                                 // counter and restart it.
                    }
        else if (step-pin > 0)
        {
            pout=hr;
                    }
                              }
```

Figure 3. C/C++ code which simulates the relay feedback.

```
/*
   PID Auto Tuning
---------------------------------------------------------
    ad0 = input of the embbebed module, 8bits
    ad  = VREF amplitude from the AD converter
    u   = controller output
    u1  = u(t-1), previous controller output
    ts  = sample time
---------------------------------------------------------*/
    q0=kc*(1+(ts/2*ti)+(td/ts));
    q1=-kc*(1-(ts/2*ti)+(2*td/ts));
    q2=kc*(td/ts);

/*---------------------------------------------------------
*/

      void autotuning(void)
   {
    ad0=IORD(INPUT_0_BASE,0);  // Getting DATA from AD
    ad1= (signed int) ad0;
    ad2= (ad1*(ad/256.0); //Converting from 8bits to
decimal
                           //'ad' is the AD conversor range
    error=(1.0-ad2); //Calculating error
    en2=en1;         //e(t-2)
    en1=en;          //e(t-1)
    en=error;        //e(t)
    u=(u1+(q0*en)+(q1*en1)+(q2*en2));// controller

    u1=u;              //adjusting u(t-1)
    ad1= (u/(ad/256.0)); //Converting from decimal to 8bits
    IOWR(OUTPUT_0_BASE, 0x0, ad1);

                                  }
```

Figure 4. C/C++ code responsible for tuning the controller.

of a PID controller was developed. This library was perfectly inserted into the Select Project Template and might serve as base to PID controller's projects.

The activities programmed into the processor makes the embedded module receive the signal from the 8 bits input, this signal corresponds to the plant's output. The input can be changed to 16 bits if necessary, although for controlling a motor's speed 8 bits are enough. After receiving the 8 bits signal, this signal is converted to a decimal base. The step is generated in the FPGA device and the error is also calculated internally.

At a first moment, the module works as a relay feedback, providing a step input in the closed-loop system. Secondly, it compares the error, and change the input's amplitude with a higher or lower value depending on its Setpoint. This effect generates an oscillatory signal with a constant period as seen in

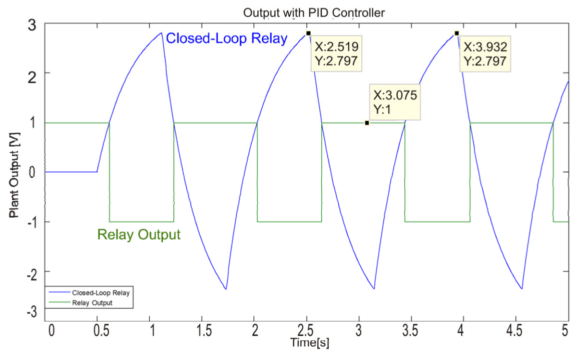


Figure 5. Plant's output with relay feedback and relay output.

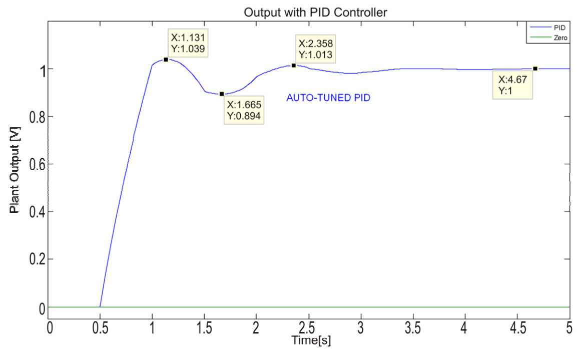


Figure 6. Plant's output with auto-tuned PID.

Figure 5. Part of the code responsible for that analysis and for setting the counter that determines the period Pu is presented in Figure 3.

The period found is used to find the parameters of the Ziegler-Nichols controller, and tune the controller. After tuning the controller, the signal is changed from the comparator to the actual controller. The tuning of the controller using C/C++ with the 8 bits input coming from the AD converter is seen in Figure 4. It can be observed that the processor calculates the parameters q0, q1 and q2, which are the controller's constants.

4 Simulation and Implementation

The system was modeled and simulated in MATLAB using the tool Simulink, as seen in Figure 1. The plant utilized for simulation is a plant present in (Nascimento Jr. and Yoneyama, 2000) whose answer is known, so that the results can be compared to the expected values. This plant is represented in the Equation (10).

$$G(S) = \frac{10 \cdot e^{(-0,5 \cdot S)}}{S+3} \tag{10}$$

Figure 5 corresponds to the plant's output using relay feedback, which is analyzed for finding the controller's parameters. The system's output using auto-tuning can be seen in Figure 6.

For the implementation of the system, it is necessary to use two converters, an AD (analogical/digital) and a DA (digital/analogical) converter.

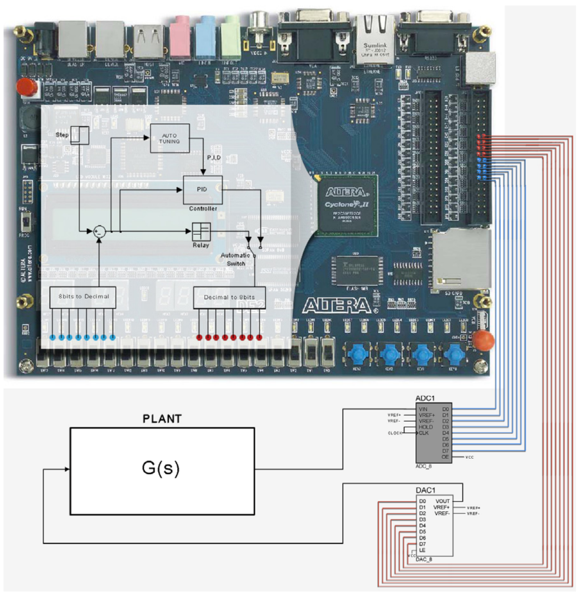


Figure 7. Representative scheme of implemented system in FPGA from DE2 Altera.

The system implemented is presented in Figure 7, where it is possible to see the complete system, and each component's function, with the FPGA responsible for the major part.

The simulation carried out with the known plant had the objective to calculate the digital PID constants q0, q1 and q2. With these parameters calculated, it is necessary to know the previous inputs as seen in Equation (3). The simulation was performed in MATLAB and the parameters calculated in the DE2 board. In that way, it was not necessary to transform the plant into a discrete one. The method presented in this article does not need an equation that corresponds to a plant, it tunes the controller automatically from an unknown plant.

5 Conclusion

The development of the PID auto-tuning controller using FPGA and NIOS II processor helps developers to increase speed in designing industrial systems. That way the developer can concentrate on other things and later use the template of the PID Auto-Tuning, saving time and hardware, because the FPGA can be used for something else at the same time. An example is a robotic arm that uses camera's images to change an object's position. The FPGA can do the image processing and the control of the arm position using the PID auto-tuning, without the need of another hardware. The applicability of the results presented in this paper can be utilized as parameters to practical implementations in control and automation industry. The results obtained with the embedded module are approximately the same as the ones obtained in simulation using MATLAB. To make it easier to future applications of the system proposed in this paper, a specific library was developed with the necessary functions to generate the PID auto-tuning. This library can be made available by the manufacturer in applications that use the NIOS II processor. For the next steps, it is hoped to do some real tests without the knowledge of the plant, where this prototype will be linked to a power circuit, signal converters and to the plant, in order to make the parameters' control. The tuning will also be done using different methods instead of Ziegler-Nichol to compare the results.

References

Altera Corporation, Nios II Software Developer's Handbook. Página web <www.altera.com> acessada em 10 de março de 2009, San Jose: Altera Corporation, 2007. pp. 620.

Anthony Cataldo (2005), Low-priced FPGA options set to expand. Electronic Engineering Times Journal, N 1361, PP 38-45, USA.

Åström, K. and Hägglund, T. (1995), PID Controllers: Theory, Design and Tuning, Ed. ISA.

Åström, K. and Hägglund, T. (2001), The Future of PID Control, *Control Engineering Practice*, Vol.9, No. 107, pp. 1163-1175.

Chien et al. (1952), Chien, Hrones, and Reswick. On the Automatic Control of Generalized Passive Systems. Transactions of the ASME, V. 74, pp. 175-185.

Cohen, G.H. and Coon, G.A. (1953), Theoretical Considerations of Retarded Control. Transactions of the ASME, pp. 827-834.

Deng D., Chen S. and Joos G. (2001), FPGA implementation of PWM pattern generators, Canadian Conference on Electrical and Computer Engineering, V1, pp. 225-230.

Franklin G., Powell J., and Emami-Naeini A. (2002), Feedback control of dynamic systems, Addison-Wesley, 4th ed.

Gordon Hands (2004), Optimised FPGAs vs dedicated DSPs, Electronic Product Design Journal, V 25, N 12, UK.

Nise, N. S. (2002). Engenharia de Sistemas de Controle, 3 Ed, Rio de Janeiro: LTC.

Ogata, K. (1998). Engenharia de Controle Moderno. 3 Ed. Rio de Janeiro: Prentice Hall do Brasil.

Rivera, Morari and Skogestad (1986). Internal Model Control, 4. PID Controler Design, Industrial and Enginering Chemistry Process Design and Development, V. 25, pp. 252-265.

Ruschel, O. T. (1996). Princípios da comunicação digital, EDIPUCRS, Porto Alegre, pp. 12.

Visioli, A. (2001), Optimal Tuning of PID Controllers for Integral and Unstable Processes. IEE Proc.Control Theory Appl., No. 148, pp. 180-184.

Wang, Y.-G, and Cai, W.-J. (2002), Advanced Proportional-Integral-Derivative Tuning for Integrating and Unstable Processes with Gain and Phase Margin Specifications. Ind. Eng. Chem. Res., Vol. 41, pp. 2910-2914.

Chan Y. F., Moallem M. and Wang W. (2004), Efficient Implementation of PID control algorithm using FPGA technology, Proceedings of the 43ed EE Conference on Decision and Control, V5, pp. 4885-4890.

Ziegler, J. and Nichols, N. (1942). Optimum Settings for Automatic Controllers, Transactions of the ASME, No. 1, pp. 759-768.