Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 49(63), Fascicola 1, 2004

The FPGA Implementation of a Digital Controller as a Digital Filter

Daniel Mic[1], Emil Micul[1], Ştefan Oniga[1], Ciprian Gavrincea[1]

Abstract - In this paper, the FPGA approach for implementation of digital controllers is selected because FPGA's can provide reconfigurable hardware designs, can process information faster than a general purpose DSP, can allow the controller architecture to be optimized for space or speed and bit widths for data registers can be selected based on application needs. Additionally, implementation in VHDL or Verilog allows the targeting of a variety of commercially available FPGA's. A digital filter very close to, if not exactly, the form of an Infinite Impulse Response (IIR) filter can represent most digital controllers. The software used for PID controller design is Matlab, specifically the tools Simulink and System Generator. The Simulink is used for determining the system response and for tuning the PID controller. With System Generator the controller is designed and the FPGA implementable VHDL code is generated. The controlled system chosen is a brushless DC motor (BLDC).

Keywords: IIR filter, PID controller, FPGA, brushless dc motor

I. INTRODUCTION

All industrial servo drives require some form of compensation often referred to as proportional, integral, and differential (PID). The process of applying this compensation is known as servo equalization or servo synthesis. In general, commercial industrial servo drives use proportional, and integral compensation (PI). It is the purpose of this paper to analyze and describe the procedure for implementing PID servo compensation using programmable logic circuits. Implementation of any complex digital controller must be done by means of some form of computer. Typical microcontrollers, while cheap, do not normally provide enough processing power to effectively perform all but the simplest calculations real-time. Digital signal processors, on the other hand, are designed to implement complex algorithms quickly. The major drawback of DSP's, however, is cost. This paper will attempt to find a median between these two extremes of performance and cost. The proposed solution is to design a special-purpose computer whose only purpose is to quickly execute the complex PID algorithm. This computer will be designed using the System Generator toolbox from Simulink and will be implemented on an FPGA (or Field Programmable Gate Array).

II. DESCRIPTION OF THE SYSTEM

The top-level system block diagram illustrates the Digital PID Controller and Brushless DC (BLDC) motor system in a closed loop configuration. The Motor Shaft Velocity will be fed back and confirmed with the Speed Command Signal to drive the system by means of an error signal. This is shown in figure 1:

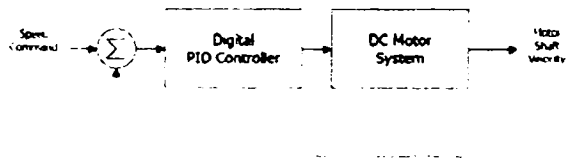

Fig. 1. Top-Level Block Diagram

The purpose of the proportional gain in the digital PID controller is to provide high loop gain in the system. This high loop gain is crucial to the operation of a closed loop system. Ideally, the system output should follow the system input. For this ideal case, the loop gain would be infinity. However, the proportional gain cannot be made arbitrarily large. What limits this is the FPGA (and the software).
The higher the proportional gain, the larger the working numbers in the software will become. At some point, the hardware will not be able to handle such large numbers, and software overflow will occur. This is the first issue, which has to be taken into account in the detailed design, that is, a reasonable balance between the desire for high loop gain and the importance of using smaller numbers. The purpose of the integrator in the digital PID controller will be to eliminate steady state error. The number of integrators actually implemented in software will depend largely upon how the system will need to track different changes in speed. This system is primarily concerned with tracking step up or step down inputs, so only one integrator will be required. The purpose of the differentiator in the digital PID controller will be to increase system speed by increasing the bandwidth.

[1] North University of Baia Mare, Electrotechnical Department
Dr. V. Babeş Str., Nr. 62/A, 430083 Baia Mare. e-mail danmic@ubm ro

The functionality of the PID controller on the system is shown in a root-locus sketch below in figure 2.

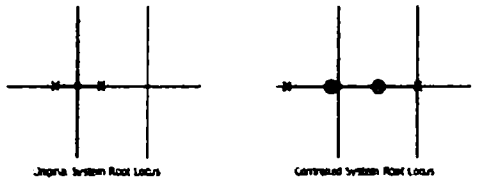

Fig. 2. Root locus sketch

III. ELECTRIC SERVO MOTOR EQUATIONS AND TIME CONSTANTS

In the analysis of electric servo drive motors, the equations for the motor indicate the presence of two time constants. One is a mechanical time constant and the other is an electrical time constant [4]. Commercial servo motor specifications usually list these two time constants. However, it should be cautioned that these two time constants as given in the specifications are for the motor alone with no load inertia connected to the motor shaft. Since these two time constants are part of the motor block diagram used in servo analysis, it is important to know the real value of the time constants under actual load conditions [4].

A derivation of the motor equations and the electrical and mechanical motor time constants will be discussed for the BLDC motor. The dc motor equivalent diagram is:

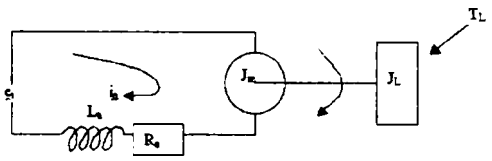

Fig. 3. DC motor equivalent diagram

Where:

e_i = Applied voltage (volts)
i_a = A[illegible] c[illegible]t (-----)
J_T = Total inertia of motor armature lus load(Kg m^2)
K_e = M[illegible]t[illegible] v[illegible]lt[illegible]ge constant (v/rad/sec)
K_T = Motor torque c nstant (Nm / A)
L_a = Motor w nd ng nductance (Henries)
R_a = A[illegible]t[illegible]i t nc (hms)
T_L = Load torque (N-m)
V_m = Motor velocity (rad/sec)
α = Acceleration (rad/sec^2)

The steady state (dc) equations are:

$$e_i = i_aR_a + K_eV_m \qquad (1)$$

$$T = \text{Torque} = i_aK_T = J\alpha \qquad (2)$$

For the general case, the differential equations are:

$$e_i = i_aR_a + L_a\frac{di_a}{dt} + K_eV_m \qquad (3)$$

$$\text{Laplace operator } S = \frac{d}{dt}$$

After some mathematical manipulations the closed loop equation is [4]:

$$\frac{V_m}{e_i} = \frac{\frac{1}{K_e}}{\left(\frac{R_aJ_T}{K_eK_T}\right)\left(\frac{L_a}{R_a}\right)S^2 + \left(\frac{R_aJ_T}{K_eK_T}\right)S + 1} \qquad (4)$$

The total inertia, J_T, is the sum of the reflected inertia to the motor shaft plus the motor inertia. The resistance, R_a, is the motor winding resistance plus the external circuit resistance. Thus the motor mechanical time constant is summarized as:

$$t_m = \frac{\sum R_aJ_T}{K_eK_T} \quad [\text{sec}] \qquad (5)$$

Also, the motor electrical time constant is:

$$t_e = \frac{L_a}{\sum R_a} \quad [\text{sec}] \qquad (6)$$

Therefore, the closed loop motor equation can be expressed as:

$$\frac{V_m}{e_i} = \frac{1/K_e}{t_mt_eS^2 + t_mS + 1} \qquad (7)$$

The mechanical and electrical time constants for a b[illegible]hl[illegible] d[illegible] [illegible]t[illegible] h[illegible] th[illegible] [illegible] b[illegible]i[illegible] [illegible]q[illegible]ti[illegible] with some variations. For a brushless dc motor with a wye connected motor, the electrical circuit is:

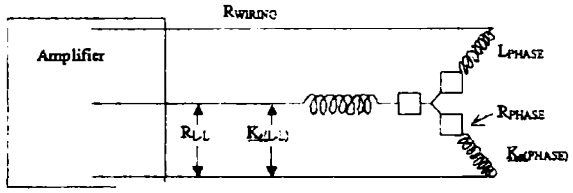

Fig.4 Wye connected BLDC

Most manufacturers give the electrical parameters in line-to-line values. Thus some of these values must be converted to the phase values as shown above. Summarizing, the mechanical time constant can be computed as:

$$t_m = \frac{\sum \frac{R_{L-L}}{2} J_{TOTAL}}{\frac{K_{e(L-L)}}{1.73} K_T} \text{ [sec]} =$$

$$0.86 \frac{R_{l-l} Jtotal\ at\ motor}{K_{e(l-l)} K_T} \quad (8)$$

The electrical time constant for the brushless dc motor is computed as:

$$t_e = \frac{L_{L-L}}{\sum R_{m(L-L)}} [\text{sec}] \quad (9)$$

IV. Controller Implementation

A. Digital Controllers

In general, digital controllers can be implemented as digital filters in the following form [3],

$$y(k) = \sum_{i=0}^{n} a_i x(k-i) - \sum_{i=1}^{n} b_i y(k-i) \quad (10)$$

Where:

- k is the current sample in time, for a given sample period T;
- y(k) is the output of the controller;
- x(k) is the input of the controller;
- a_i and b_i are coefficients, or gains, of the controller.

These gains must be selected to produce the desired controller response for a given dynamic system to be controlled. The process of determining these gains is called "tuning." The structure of this digital controller is illustrated in figure 5. In the figure, the Z^{-1} blocks represent delays of one sample period.

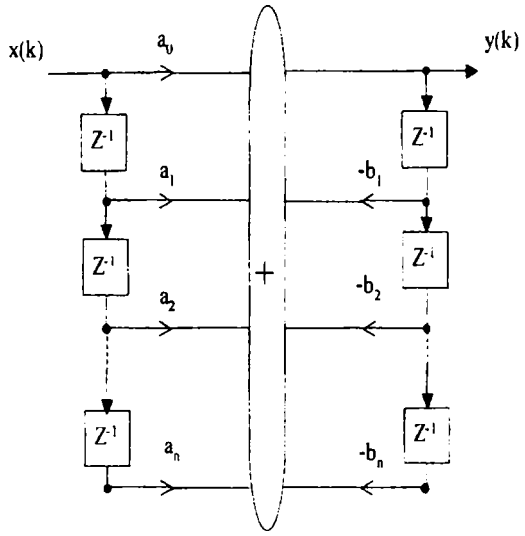

Fig. 5 Diagram of the digital controller

When n = 2, a second order filter is obtained which can be used to implement second order controllers or cascaded to create higher order controllers. This representation is shown in the sampled time domain below:

$$y(k) = a_0 x(k) + a_1 x(k-1) + a_2 x(k-2) - b_1 y(k-1) - b_2 x(k-2) \quad (11)$$

The z-transform of this gives the following transfer function:

$$D(z) = \frac{Y(z)}{X(z)} = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} \quad (12)$$

A. PID controller

A PID controller, as its name suggests, provides proportional, integral, and derivative compensation to an existing system. These three forms of compensation increase system performance in a variety of ways. Proportional control can both increase gain margin and stabilize a potentially unstable system. Integral control can minimize steady state error. Derivative control can increase system speed by increasing system bandwidth. One drawback of PID control is overall complexity. This results in very expensive means of implementing a digital version of a PID controller. Of the many possibilities, digital signal processors (or DSP's) are the most widely used to solve this problem, however other possibilities exist which may be more cost-effective.
The well-known PID controller can be implemented using a second order digital filter [3].
The continuous time representation of a PID controller is as follows;

$$u(t) = K_P e(t) + \frac{1}{(K_I)} \int e(t)dt + K_D \frac{de(t)}{dt} \quad (13)$$

Where:

- K_P is the proportional gain;
- K_I is the integral gain;
- K_D is the derivative gain
- e(t) is the error between the desired response and the actual system response.

In the sampled time domain, with sample period T, the PID controller is represented as [3]:

$$u(k) = K_P e(k) + \frac{T}{K_I} S(k) + K_D \frac{e(k) - e(k-1)}{T}$$
$$S(k) = S(k-1) + \frac{T}{2}[e(k) + e(k-1)] \quad (14)$$

where:

- *T* sample time period;
- *k* index of the k sample;
- *S* sum of the errors to k^{th} sample.

The z-transform of this controller gives the following transfer function:

$$D(z) = K_P + \frac{K_I T}{2}\left(\frac{z+1}{z-1}\right) + \frac{K_D}{T}\left(\frac{z-1}{z}\right) \quad (15)$$

With some manipulation this transfer function can be represented as a second order filter:

$$D(z)=\frac{(K_P+K_IT/2+K_D/T)}{1-z^{-1}}+\frac{(-K_P+K_IT/2-2K_D/T)z^{-1}+K_D/Tz^{-2}}{1-z^{-1}} \quad (16)$$

$$a_0=K_P+\frac{K_IT}{2}+\frac{K_D}{T}$$

$$a_1=-K_P+\frac{K_IT}{2}-\frac{2K_D}{T}$$

$$a_2=\frac{K_D}{T} \quad (17)$$

$$b_1=-1$$

$$b_2=0$$

V. SIMILATION RESULTS

The experimental system chosen is a brushless DC motor. A diagram of the system is shown in figure1. The controller must produce a motor current (torque) command, which is supplied to the motor driver. The motor driver controls the motor current to produce motor torque proportional to the current command.
The first step will be the system simulation, as an open loop system. This will be done in Matlab – Simulink. The second step will be the system simulation as a close loop system, including the PID controller. These simulations will be crucial to developing appropriate PID constants for desired system specifications. Although this project will not result in a practical system for direct implementation in industry, a few specifications have been laid down for "academic" purposes. These specifications are listed below:

- Settling time less than 2 seconds;
- Overshoot less than 5%;
- Steady-state error less than 1%.

Since the most basic requirement of a motor is that it should rotate at the desired speed, the steady-state error of the motor speed should be less than 1%. The other performance requirement is that the motor must accelerate to its steady-state speed as soon as it turns on. In this case, we want it to have a settling time of 2 seconds. Since a speed faster than the reference may damage the equipment, we want to have an overshoot of less than 5%.
Simulink will be used to show uncompensated system plots, compensated system plots and to determine digital PID controller coefficients.

A. Motor characteristics

The brushless dc motor, choused for this experiment, is a three phase synchronous dc motor having a position transducer inside the motor to transmit motor shaft position to the drive amplifier for the purpose of controlling current commutation in the three phases of the motor windings. Motor characteristics are shown in table 1 [8].

Table 1

Parameter	Value	Unit
E – reference voltage	19.1	V
V- no load speed	7400	rpm
K_T - Torque constant	2.43E-02	Nm / A
K_E - Back-EMF constant	2.43E-02	V/rad/s
R_T - Resistance	2.67	Ω
L - Inductance	0.29	MH
J_M - Rotor inertia	1.4E-06	Kg m^2
τ_E - Electrical Time Constant	0.12	ms
τ_M - Mechanical Time Constant	6.2	ms

B. Open-loop response of the motor

Figure 6 shows the Simulink model for BLDC motor. This model was written based on mathematical equations (7,8,9). Figure 7 shows the system response at a unit step signal applied to the input. It can be noticed an uncontrolled rising of speed and a large period in time response, over 20s.

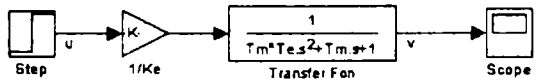

Fig. 6. Simulink Model for Open Loop Response

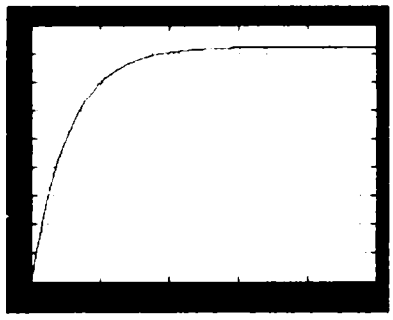

Fig. 7. Signal Step Response for Open Loop BLDC

C. Closed loop response

Figure 8 shows the BLDC closed loop system. In order to make a comparison, the PID controller was first modeled in Simulink based on equation (13), in its classic form. The output response at a unit step signal it is shown in figure 10. In can be noticed that the settling time, overshot and steady state error specifications are accomplished.

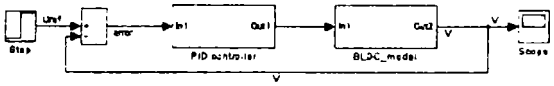

Fig. 8. Simulink Model for Closed Loop Response

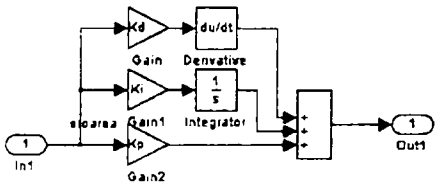

Fig. 9. Simulink Model of Classic PID controller

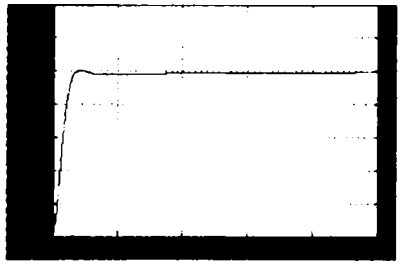

Fig. 10. Response of Classic PID controller

Figure 11 shows the proposed PID controller modeled as an IIR filter. This model was build based on equations (16, 17). It was implemented using System Generator that is a toolbox of Simulink. System Generator is a powerful high level-modeling environment for DSP systems, and consequently is widely used for algorithm development and verification. System Generator for DSP maintains an abstraction level very much in keeping with the traditional Simulink blocksets, but at the same time automatically translates designs into hardware implementations that are faithful, synthesizable, and efficient [7]. Figure 12 shows the output response of the closed loop system based on the new IIR form of PID controller. In can be noticed that the settling time, overshot and steady state error specifications are accomplished.

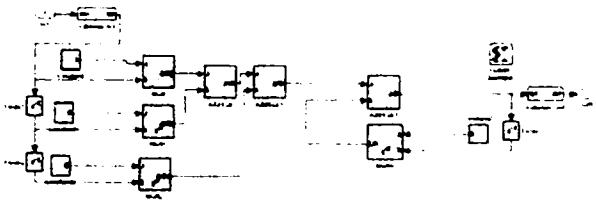

Fig. 11. System Generator Model of PID controller

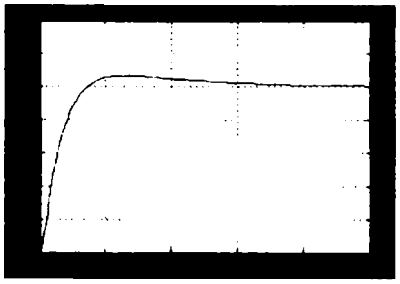

Fig. 12. Response of of PID controller build with System Generator

V. CONCLUSIONS

Implementation of controllers as digital filters in FPGA's is shown to be feasible, and the implementation of the added components for an adaptive controller is expected to also be successful. The relevant literature contains many examples of successful controller implementations in FPGA's (1,2,3,6). High order digital filters, with greater complexity than most digital controllers, have also been implemented on FPGA's, implying that controllers of greater complexity can be successfully implemented. The capability of FPGA's to operate at clock rates in the range of 10's to 100's of MHz provides plenty of processing capability. Controllers generally operate at sampling frequencies of much less than 10 kHz, for most practical applications, so throughput should not be a problem. Implementation of the identifier and controller design calculations will be challenging, but is considered feasible. The identifier and design equations have multiple variable terms and will require careful scaling.

REFERENCES

[1] Cirstea, Marcian, Khor, Jean and McCormick, Malcolm, "FPGA Fuzzy Logic Controller for Variable Speed Generators", *Proceedings of the IEEE International Conference on Control Applications*, pp 301-304, September 2001

[2] Daniel. Mic, Implementation of Stepper Motor Controller in a XILINX FPGA, Buletin Stintific seria C volumul IX, Fascicola Electronica, Electrotehnica, Automatizări, Simpozionul Stintific National, Baia Mare, 15-16 Mai 2001, pp 83-88.

[3] David A. Gwaltney, Kenneth D. King and Keary J. Smith "Implementation of Adaptive Digital Controllers on Programmable Logic Devices", NASA Marshall Space Flight Center, Huntsville www.nasa.org

[4] George W. Younkin, P.E., Life FELLOW – IEEE, „Electric Servo Motor Equations And Time Constants" Industrial Controls Consulting, Inc, Fond du Lac, Wisconsin

[5] Mathlab the language of technical computing, Simulink for model-based and system level design (www.mathworks.com)

[6] Stefan Oniga, D. Mic, Application possibilities of a development board with FPGA in signal processing, International Computer Science Conference microCAD 2001, Miskolc, Hungary, 1-2 March 2001, Section G.

[7] Xilinx System Generator for DSP (www.xilinx.com)

[8] PittmanExpres, "Brushless dc servomotor – data sheet", 3442S003-R3.pdf.