FPGA Implementation of Digital PID

Alireza rezaee
Amirkabir University of technology
IRAN

Abstract: This paper concerns the design of an ASIC with FPGA technology. The ASIC function is the PID controller .The design focus on optimization of a silicon area and speed operation.
This ASIC-PID has been implemented by means of three blocks: Adder, Multiplier and control unit. The speed performance of this system is compared to design with Digital PID on AVR (AT mega 128L).This hardware implementation can be done in three different architectures: Serial, Parallel or Mixed.
In all cases this kind of electronic implementation is very fast response than a software.

Keywords: Control, PID, FPGA.

1. Introduction

The modern digital control systems require more and more strong and fastest calculation components. This type of elements becomes yet indispensable with the utilization of some new control algorithms control like the fuzzy control, the adaptive control, and ….
Although the PID controllers are the oldest they represent the most used controllers in the industrial control systems but it is by far the most commonly used control algorithm.
Three types of PID controller are defined. The parallel controller where the action P,I and D are independent the equation of this type is

$$L(p) = E(p)(K_p + \frac{1}{T_i p} + T_d p)$$

The serial controller where the gain influence the integral and differential actions. This equation is

$$L(p) = E(p)K_p(1 + \frac{1}{T_i p})(1 + T_d p)$$

The mixed controller has the equations:

$$L(p) = E(p)K_p(1 + \frac{1}{T_i p} + T_d p)$$

This entire controller has a general equation.
The textbook version of PID controller can be described by the equation

$$u(t) = \alpha e(t) + \beta \int_0^t e(s)ds + \delta \frac{de(t)}{dt} \qquad (1)$$

Where u is the control variable and e is the control error .The controller parameters are: the gain, the integral time and the derivative time
The purpose of the integral action is to increase the low-frequency gain and thus reduce steady-state errors. The Derivation action adds phase lead, which improves stability and increases system bandwidth.

Implementation of PID controller using an FPGA will be discussed in this paper.
A lot of experience has been accumulated over many years of use of algorithm. The discrimination will be discussed in section 2.
The result is a linear digital algorithm that is suitable for implementation on general purpose digital computer. The algorithm can be implemented in a straightforward way in the FPGA hardware.
An overview of AVR (AT mega 128) code for PID controller is described in section 5.
Section 7 describes how the ASIC_PID can be validated and tested.

2. Discretization

The algorithm (1) has several drawbacks. Significant modifications of linear and non-linear behavior are necessary in order to obtain a useful algorithm. To obtain equations that can be implemented using computer, it is necessary to replace continuous time operation like derivation and integration by discrete time operations.
The proportional term $\alpha e(t)$ is implemented simply by replacing the continuous time Variables by their sampled equivalence. The proportional term then becomes

$$P(t_k) = \alpha\, e(t_k) \qquad (2)$$

With $e(t_k) = e_k$ and $u(t_k) = u_k$
Where denotes the sampling instants.
When a controller operates over a wide range of operating conditions, the control variable may reach actuator limits.
A pure derivative term can be written as

$$D(t_k) = \delta \frac{e(t_{k+1}) - e(t_k)}{h} \qquad (3)$$

Where h is the sampling period.

The feedback loop is then broken and the system effectively runs open loop. When this happens in a controller with integral action, the error will continue to be integrated and the integral term may become very large.

Using this method the integral term becomes

$$I(t) = \int_0^t e(s)ds \qquad (4)$$

To obtain an algorithm that can be implemented on a computer, three methods can be used for integral functions approximation:

$$1.\ I(t_k) - I(t_{k-1}) = he(t_{k-1})$$
$$and \quad I(t_k) = \sum_{j=0}^{k-1} he(t_k)$$

$$2.\ I(t_k) - I(t_{k-1}) = he(t_k)$$
$$and \quad I(t_k) = \sum_{j=0}^{k} he(t_k) \qquad (5)$$

$$3.\ I(t_k) - I(t_{k-1}) = he(t_k) \quad and$$
$$I(t_k) = \sum_{j=0}^{k} h \frac{(e(t_k) + e(t_{k-1}))}{2}$$

We use incremental method $\Delta u_k = u_k - u_{k-1}$ to compute the action.

The use of three types PID controller and three possible approximations of integral functions involves nine possible equations.
We chose to implement the mixed controller with trapezoidal approximations for three integral since it has the minimum to summarize, we find

that a practical version of the PID algorithm can be described by the equations (2), (4) and (5).
Using these equations, the PID output is given by equation

$$L_k = L_{k-1} + e_k(K_p + K_D + K_I/2) + e_{k-1}(K_i/2 - K_p - 2K_D) + e_{K-2}K_D)$$

3. Implementation of PID

Implementation of PID-controller using an FPGA is now discussed.
It is common practice to estimate computation times by a simple operation count. To improve the speed and minimize the cost while offering clearly good performances, one mixed architecture is used that includes essentially: one combinational logic multiplier, two adders, and accumulator. One finite state machine is in case necessary to manage the whole exchange and data transfer operations. The Fig. 1 gives the adopted architecture

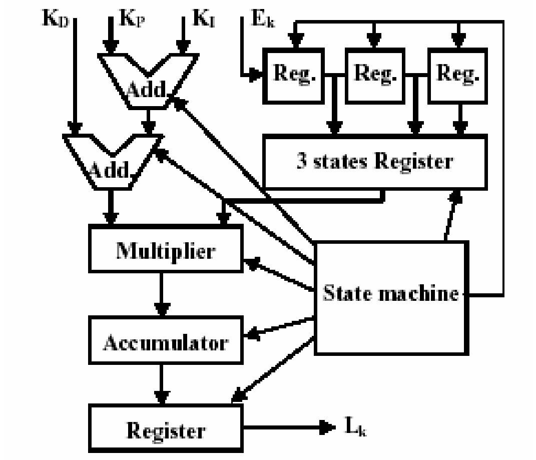

figure 1.Mixed PID Architecture

The routing and the placement of PID is done for circuits:

- XC4013PQ240-4, which has 576 CLBs and 192 inputs/outputs for the parallel architecture.
- XC4005PC84-4, which has 196 CLBs and 192 input/outputs for the serial architecture.
- XC4006PC84-4, which has 256 CLBs and 61 input/outputs for the mixed architecture.

The choice of those part types is the best to have optimal implementation. In fact, for a largest part type the occupation rate is very weak and the number of occupied CLBs is very high. In this case a big CLBs number is partially used.

- For parallel architecture: 512 CLBs for about 88% occupation rates.
- For serial architecture: 176 CLBs for about 89.8%.
- For mixed architecture: 225 CLBs for about 87% occupation rate.
- Those rate sill high results of the sustained attention agreed to logic FPGA blocks connection in the placement .Indeed, the disposition of these CLBs facilitates the distribution of registers content, which follows the privileged data circulation direction proper to the XC4000 family. Some constraints were introduced on the location of inputs and control lines to optimize the implementation.

4. The PID Simulation

The performances expected for the PID module has normally a direct relationship with the imposed time constrains.

The realization of PID circuit for serial and mixed architecture being synchronous to a basic clock, the execution time to directly a multiple of this clock cycle.

This clock cycle is dependent of the:

- Execution time or operators propagation
- The finite state machine cycle
- Some delays added in routed circuit

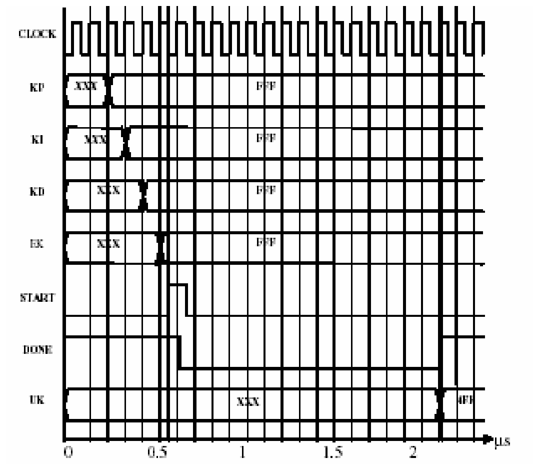

Figure2. simulation of PID controller

5. AVR microcontroller code

The main PID controller routine was designed to be fairly general purpose and hence modular. Whilst here it is used to control a DC motor, it could be re-deployed to other situations where some parameter has to be controlled to a set value under varying conditions. The actual control software is located in a single function and its major inputs and output are held in a structure. Although it was designed originally for a specific job it is really only intended as an example of the basic techniques involved and to allow those with no control system knowledge to experiment with a simple PID system.

The routines that gather the inputs and process the output are kept in separate functions in another module. The code vision compiler was used .

For simplicity and to allow the easy modification of the important PID gain parameters, the AT mega 10-bit analog to digital converter was used to derive 10-bit resolution inputs from simple trimmer potentiometers. The PWM used to drive the motor was chosen as 10-bits so that motor speed can be defined to approximately 0.1%, sufficient for most practical application. The ATmega128 high resolution dedicated PWM unit could have been used to increase this to a supersonic 19.5kHz but to allow easy porting to other AT mega variants this route was not taken. Fortunately the use of a 10 bit resolution on the inputs and output makes some of the arithmetic easier!

The Atmega128 IO pins are allocated as per:

P2.0 optical chopper encoder input

P2.1 channel 2 - PWM drive output

P5.0 analog channel 0 - set point input

P5.1 analog channel 1 - derivative gain input
P5.2 analog channel1 proportional gain input

P5.3 analog channel 1 - integral gain

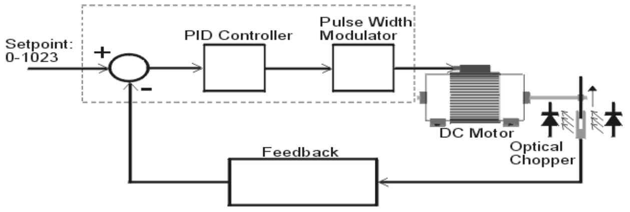

figure3. Implementation OF PID on AVRAtmega28

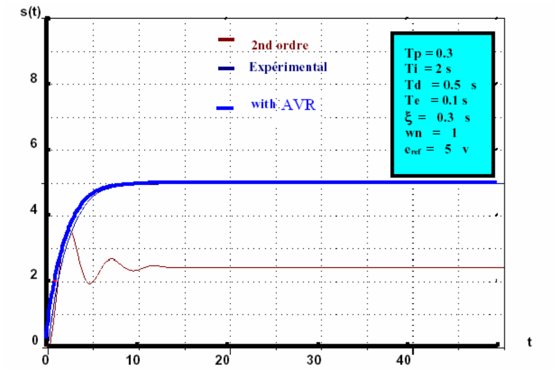

Figure 4.compare of PID on FPGA and AVR

6. Conclusion

This paper has given an FPGA implementation for PID controller. It has also shown how the ASIC-PID can be implemented on an FPGA. Such implementation necessarily requires some operators architectures(Adder,Mutiplier).
The same PID algorithm is implemented on a AVR(ATmega128).

REFRENCES

[1]Fargeon C. Commande numerique des systemes .444 p,1986,masson.
[2]BUHLER H,Reglagess echantillones. 382 p,1986 ,Presses polytechniques romande.
[3]The Programmable Gate Array Data Book,Xilinx 1991-94.
[4]KHARRAT M.W.,MASMOUDI N.,SAMET L.,KAMOUN L. Integration des additionneurs numeriques en technologic FPGA JTEA 8-9 Novembre 1996
[5]CLAVEZ J.P Specification et conception des ASICs,Paris 1993,Edition MASSON.