We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

11

Design of PSO-Based Optimal/Tunable PID Fuzzy Logic Controller Using FPGA

Zeyad Assi Obaid[1], Saad Abd Almageed Salman[1], Hazem I. Ali[2], Nasri Sulaiman[3], M. H. Marhaban[3] and M. N. Hamidon[3]
[1]*College of Engineering University of Diyala*
[2]*Control Engineering Department University of Technology*
[3]*Faculty of EngineeringUniversity Putra Malaysia*
[1,2]*Iraq*
[3]*Malaysia*

1. Introduction

Fuzzy Logic controllers have been successfully applied in a large number of control applications, with the most commonly used controller being the PID controller. Fuzzy logic controllers provide an alternative to PID controllers, as they are a good tool for the control of systems that are difficult to model. The control action in fuzzy logic controllers can be expressed with simple "if-then" rules (Poorani et al., 2005). Fuzzy controllers are more sufficient than classical controllers as they can cover a much wider range of operating conditions than classical Controllers. In addition, fuzzy controllers operate with noise and disturbances of a different nature. The common method for designing a fuzzy controller is to realize it as a computer program. Higher density programmable logic devices, such as the FPGA, can be used to integrate large amounts of logic in a single IC. The FPGA provides greater flexibility than ASIC, and can be used with tighter time-to-market schedules. The term programmable highlights the customization of the IC by the user. Many researchers have discussed the design of the hardware implementation of fuzzy logic controllers. A number was specialized for control applications, and aimed to get better control responses. These researches have concern using new techniques in fuzzy control, in order to get higher processing speed versus low utilization of chip resource (Jain et al., 2009 and Islam et al., 2007)

The hardware implementation of fuzzy logic controllers has many requirements, such as high-speed performance, low complexity and high flexibility (Leonid, 1997). With this type of application, and to provide these requirements, it is necessary to avoid various limitations and challenges. In deriving a practical PIDFC structure, it is desirable to reduce the number of inputs. In addition, it is difficult to formulate the fuzzy rules with the variable sum of error ($\sum e$), as its steady-state value is unknown for most control problems (Mann et al., 1999), (Hassan et al., 2007), (Obaid et al., 2009). Hence, it is necessary to design the FPGA-based PIDFC with fewer inputs and rules to get higher processing speed versus low utilization of chip resources. In addition, the design of digital fuzzy controllers has limitations concerning the structure. These include the restriction or limitation of the shapes

of fuzzy sets implemented in the fuzzifier block, such as those in the controllers proposed in (Hassan et al., 2007) and (Seng et al., 1999). It is desirable to simplify the structure of the PIDFC controller to offer higher flexibility versus low-chip resources. The majority of PID fuzzy controller applications belong to the direct action (DA) type; here the fuzzy inference is used to compute the PID controller actions, not to tune the PID parameters (Mann et al., 1999). Therefore, it is necessary to design a tuning method inside the digital fuzzy chip, especially with the DA type, in order to scale the universe of discourse for the inputs/outputs variable in the PIDFC, and to obtain the best tuning case in the operational range. Changing the scaling gains at the input and output of PIDFC has a significant impact on the performance of the resulting fuzzy control system, as well as the controller's stability and performance (Leonid et al., 2000). Therefore, it is necessary to use an optimization method to calculate the optimal values of these gains. In recent years, several researchers have designed fuzzy controllers with different ranges of accuracy. Most of these controllers have 6-8 bits of accuracy (Poorani et al., 2005), (Tipsuwanpornet al., 2004), (Hassan et al., 2007), (Solano et al., 1997), (Gabrielli et al., 2009). This accuracy has a trade off with the speed of the process and it may affect the process behavior inside the digital fuzzy chip (Jantzen, 1998), (Ibrahim, 2004). Furthermore, increasing the accuracy will also increase the hardware complexity versus low-speed performance and vice versa. However, none have evaluated the same controller with different ranges of accuracy. Hence, it is necessary to find the best accuracy inside the digital chip that offers low hardware complexity versus high-speed performance. The aim of this chapter is to design a PIDFC that can efficiently replace other fuzzy controllers realized as a computer program, that have the ability to serve a wide range of systems with real-time operations. To achieve this aim, the following points are addressed:

Ref No.	Controller type	Optimization and tuning method	System Type
(Li and Hu , 1996)	PID with FIS	Tuned by FIS	Process control
(Tipsuwanpornet al., 2004)	Gain scheduling PID FC	Gain scheduling Tuning method	Level and temperature
(Poorani et al., 2005)	Specific FC 6-inputs, 1-output	No method	Speed control of electric vehicle
(Alvarez et al., 2006)	Optimal FC 2-inputs, 1-output	No method	Multi phase converter
(Gonzalez-Vazquez et al, 2006)	PD FC	No method	No system
(Khatr and Rattan, 2006)	Multi-layered PDFC	No method	Autonomous mobile robot
(Hassan et al., 2007)	PID FC	No method	Linear system
(Jain et al., 2009)	Optimal PID controller	Bacterial Foraging Optimization method	Inverted pendulum

Table 1. Summary of the Related FPGA-Based Controller in the Literature

1. To design a PIDFC with improved fuzzy algorithm that offers low implementation size versus high processing speed.
2. To aid the PIDFC with a tuning gains block inside the FPGA chip that makes the design able to accept PSO-based optimal scaling gains.
3. To design two versions of the proposed PIDFC. The first one is an 8-bits PIDFC, while the second one is a 6-bits PIDFC.
4. To test the proposed design with different plants models with a unity feedback control system.

Many researchers have discussed different approaches to the present fuzzy algorithms that offer high-processing speed and small chip size. Many of the fuzzy controllers implemented in the literature using FPGA, have many limitations and challenges with the structure, such as those in the shape of fuzzy sets implemented in the Fuzzifier block. This thesis will deal with some of these limitations. Table I lists the related FPGA-based controllers in the literature, highlighting the tuning and optimization methods used as well as the type of application and the type of controller.

2. PID fuzzy logic controller

A PID fuzzy controller is a controller that takes error, summation of error and rate of change of error (rate for short) as inputs. Fuzzy controller with three inputs is difficult and not easy to implement, because it needs a large number of rules and memory (Leonid, 1997). In the proposed design, if each input is described with eight linguistic values, then 8x8x8=512 rules will be needed. The PIDFC can be constructed as a parallel structure of a PDFC and a PIFC, and the output of the PIDFC is formed by algebraically adding the outputs of the two fuzzy control blocks (Leonid, 1997). In deriving a practical PIDFC structure, the following remarks are made (Hassan et al., 2007) (Mann et al., 1999), (Leonid, 1997), (Obaid et al., 2009): *Remark 1:* For any PIDFC, the error (e) is considered as the necessary input for deriving any PID structure. *Remark 2:* It is difficult to formulate control rules with the input variable sum-of-error $\sum e$, as its steady-state value is unknown for most control problems. To overcome the problem stated in remark 2, a PDFC may be employed to serve as a PIFC in incremental form, where a PD fuzzy logic controller, with summation at its output, is used instead of the PIFC (Hassan et al., 2007), (Obaid et al., 1999).

3. Particle swarm optimization

PSO is an evolutionary computation technique-based developed by Eberhart and Kennedy in 1995 (Wang et al., 2006). It was inspired by the social behavior of birds flocking or fish schooling, more specifically, the collective behaviors of simple individuals interacting with their environment and each other (Wang et al., 2006). PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas where evolutionary computation can be applied. Similar to evolutionary computation (EC), PSO is a population-based optimization tool (Wang et al., 2006), (Allaoua et al., 2009). The system is initialized with a population of random solutions and searches for optima by updating generations. All of the particles have fitness values that are evaluated by the fitness function to be optimized, and have velocities that direct the flying of the particles (Wang et al., 2006). In a PSO system, particles change their positions by flying around in a multidimensional search space until computational limitations are exceeded.

The concept of the modification of a searching point by PSO is shown in Fig. 1 (Allaoua et al., 2009).

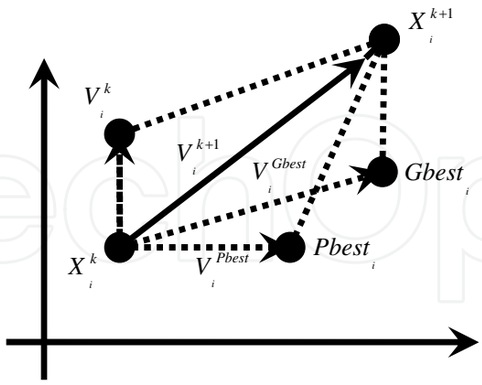


Fig. 1. Concept of the Modification of a Searching Point by PSO (Allaoua et al., 2009).

In the PSO algorithm, instead of using evolutionary operators such as mutation and crossover, to manipulate algorithms, for a d-variable optimization problem, a flock of particles are put into the d-dimensional search space with randomly chosen velocities and positions knowing their best values so far (Pbest) and the position in the d-dimensional space. The velocity of each particle, adjusted according to its own flying experience and the other particle's flying experience. For example, the *i*-th particle is represented as *xi* = (*xi,* 1, xi, 2… *xi,* d) in the d-dimensional space. The best previous position of the *i*-th particle is recorded and represented as (Allaoua et al., 2009):

$$\text{Pbest}i = (\text{Pbest}i, 1, \text{Pbest}i, 2 \ldots \text{Pbest}\ i, \text{d}) \tag{1}$$

The index of best particle among all of the particles in the group is gbest*d*. The velocity for particle *i* is represented as *vi* = (*vi,*1 ,*vi,*2 ,…, *vi,*d). The modified velocity and position of each particle can be calculated using the current velocity and the distance from Pbest*i,* d to gbestd as shown in the following formulas (Allaoua et al., 2009):

$$V_{i,m}^{(t+1)} = W.V_{i,m}^{(t)} + c_1 * rand\,() * (Pbest_{i,m} - X_{i,m}^{(t)}) + c_2 * rand\,() * (gbest_m - X_{i,m}^{(t)}) \tag{2}$$

$$X_{i,m}^{(t+1)} = X_{i,m}^{(t)} + V_{i,m}^{(t+1)} \;\; ; \quad i = 1, 2 \ldots n; \quad m = 1, 2 \ldots d \tag{3}$$

Where:
N: Number of particles in the group,
D: dimension,
t: Pointer of iterations (generations),
$V_{i,m}^{(t)}$: Velocity of particle I at iteration t, $V_d^{\min} \le V_{i,m}^{(t)} \le V_d^{\max}$
W: Inertia weight factor,
*c*1, *c2*: Acceleration constant,
rand ( ): Random number between 0 and 1
$X_{i,m}^{(t)}$: Current position of particle *i* at iterations,
Pbesti: Best previous position of i-th particle,
Gbest: Best particle among all the particles in the population.

4. The proposed PSO algorithm

The main aim of the PSO algorithm is to tune the controller parameters [Kp, Kd, Ki, Ko], by minimizing the cost function for minimum values in order to get the optimal gains value for these parameters. The target cost function is the integral square error (ISE), this is simple function and can easy represented in the fuzzy algorithm. The cost function (equation 4) is calculated by swapping the searching results in the local position with the minimum value of the function until reaching the best global search. In this case, the proposed PSO algorithm is 6-dimension in the population size for PIDFC (also can do it by 4-dimenssion), 3-dimension in the case of the PIFC and PDFC. This dimension belongs to the controller parameters, which represent the particle (X) inside the population space. These particles are explained in equations (5, 6 and 7) with i*th* iteration path. Note that during the search process the resulting gains were constrained by the interval [Xmin Xmax] to search with these limits in order to cover the range of the operational range (universe of discourse). Fig. 2 shows the flow chart of the proposed PSO algorithm.

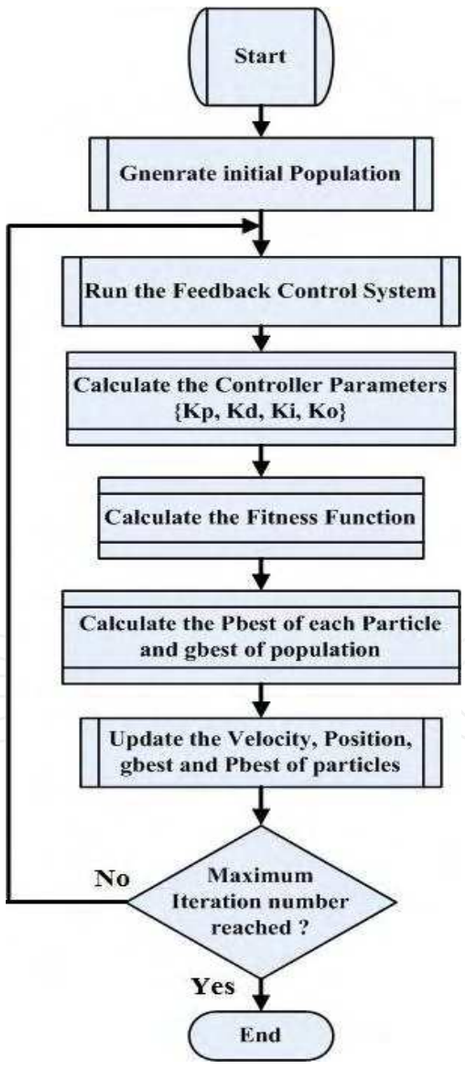


Fig. 2. Flowchart of the Proposed PSO Algorithm

$$ISE = \sum_{t=0}^{Maxiteration} (e(t))^2 \tag{4}$$

$$\begin{aligned} &X(PIDFC)= [x(1)\ \ x(2)\ \ x(3)\ \ x(4)\ x(5)\ \ x(6)]= \\ &[Kp\ \ Kd\ Ko + Kp\ \ Ki\ \ Ko]\quad , \text{Dimension} = 6 \end{aligned} \tag{5}$$

$$X(PIFC)= [x(1)\ \ x(2)\ \ x(3)]= [Kp\ \ Ki\ \ Ko] \quad , \text{Dimension} = 3 \tag{6}$$

$$X(PDFC)= [x(1)\ \ x(2)\ \ x(3)]= [Kp\ \ Kd\ \ Ko] \quad , \text{Dimension} = 3 \tag{7}$$

5. Block diagram of the PIDFC

The proposed controller accepts two signals, the first one is the plant output (y_p) and the second one is the desired output (y_d), both of them are digital signals, and deliver the control action signal as a digital output. It also accepts four 8-bit digital signals that represent the optimal gain parameters needed by the controller (K_p , K_d , K_i , and K_o). These parameters are used to aid the tuning block with optimal values of the scaling gains online with the digital FPGA chip. Other two (one-bit) signals have been used to select the type of the controller (PDFC, PIFC, or PIDFC) online with the chip. Fig. 3 shows the general block diagram of the controller chip in a unity feedback control system. In recent years, many of the digital fuzzy applications have different ranges of the accuracy. Most of them have 6-8 bits of accuracy (Poorani et al., 2005), (Tipsuwanpornet al., 2004), (Hassan et al., 2007), (Solano et al., 1997), (Gabrielli et al., 2009), (Obaid et al., 2009), (Obaid et al., 1999). This accuracy may affect the process behavior inside the digital fuzzy chip; also it has a trade off with the speed of the process (Leonid , 1997), (Jantzen, 1998), (Ibrahim, 2004). Therefore, it is necessary to find which range has better accuracy inside the digital chip. Two versions of the proposed PIDFC were designed, the first one is an 8-bit which uses 8 bits for each input/output variables. The second version is a 6-bit which uses 6 bits for each input/output variable. To make the discussion clear and general for the proposed controller in the following sections, symbol (q) will be used to represent the range of accuracy, (q=8) in the proposed 8-bits design, and (q=6) in the 6-bits version of the proposed design.

6. Structure of the PIDFC design

Generally, to represent PIDFC, it is required to design a fuzzy inference system with three inputs that represent the proportional, derivative, and integral components, and each one of them can have up to eight fuzzy sets. Therefore, the maximum number of the required fuzzy rules is 8^3=512 rules. To avoid this huge number of rules, the proposed controller was designed using two parallel PDFC to design the PIDFC as discussed earlier (Hassan et al., 2007), (Obaid et al., 2009), (Obaid et al., 1999). The second PDFC was converted to a PIFC by accumulating its output. Fig. 4 shows the structure of proposed PIDFC, where FIS refers to the fuzzy inference system with its three blocks, Fuzzifier, inference engine and defuzzifier. Both controllers, PDFC and PIFC, receive the same error signal. The structure of the single PDFC is discussed in the next sections. The main block in the PDFC is the fuzzy inference block which has two inputs ($e(n)$ and $\Delta e(n)$), one output (U(n)) fuzzy system of Mamdani

type that uses singleton membership functions for the output variable. Initially, the two input signals are multiplied by a gain coefficient (K_p and K_d or K_p and K_i) before entering the fuzzy inference block. Similarly, the output of the fuzzy inference block is multiplied by a gain coefficient (K_o) (Hassan et al., 2007), (Obaid et al., 2009), (Obaid et al., 1999). At the same time, the output of the fuzzy inference block in the second PDFC is multiplied by a gain coefficient and then accumulated to give the output of the PIFC. Subsequently, both outputs of the PDFC and PIFC are added together to give the PIDFC output (u_{PID}). The final design works as a PDFC, PIFC, or PIDFC, depending on the two selection lines sw_1 and sw_0, which provide a more flexible design to cover a wide range of systems. The PIDFC is designed using two blocks of PDFC, and the main block in the proposed design is the PDFC block. The main components inside the PDFC block are: *Tuning-gain* block, *Fuzzifier* block, *inference engine* block, and *Defuzzifier* block.

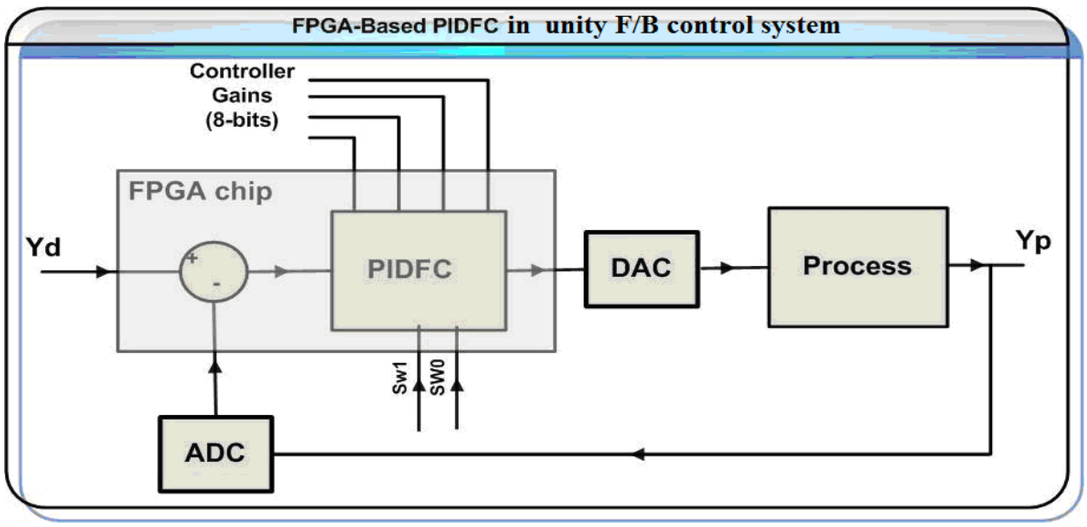


Fig. 3. Block Diagram of the PIDFC in a Unity Feedback Control System.

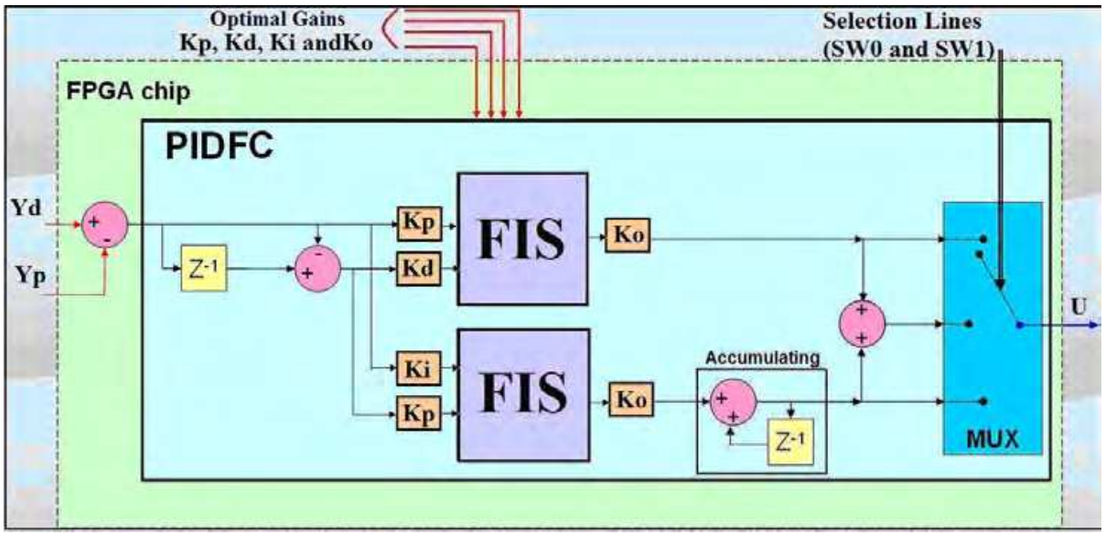


Fig. 4. Main structure of the Proposed Controller

Tuning-Gain Block

The tuning-gain block is used at each of the two inputs and also at the output of each PDFC block. This block receives and multiplies two inputs: the variable to be scaled (input or output) and its related gains, this implies the proposed tuning method via scaling the universe of discourse. An eight-bit latch was used at each Tuning-gain block to store the gain coefficient value received from one of the gain ports, depending on selection line values. The "*" operator was used in the VHDL files of the design to express a multiplication process just like a conventional language. This process has been designed at the *behavioral* level of abstraction in VHDL code, i.e. during the design synthesis process, if the library "*IEEE.std_logic_signed*" was included in the VHDL files (Hassan et al., 2007), (Obaid et al., 2009), (Obaid et al., 1999). Fig. 5 shows the Tuning-gain block with more details.

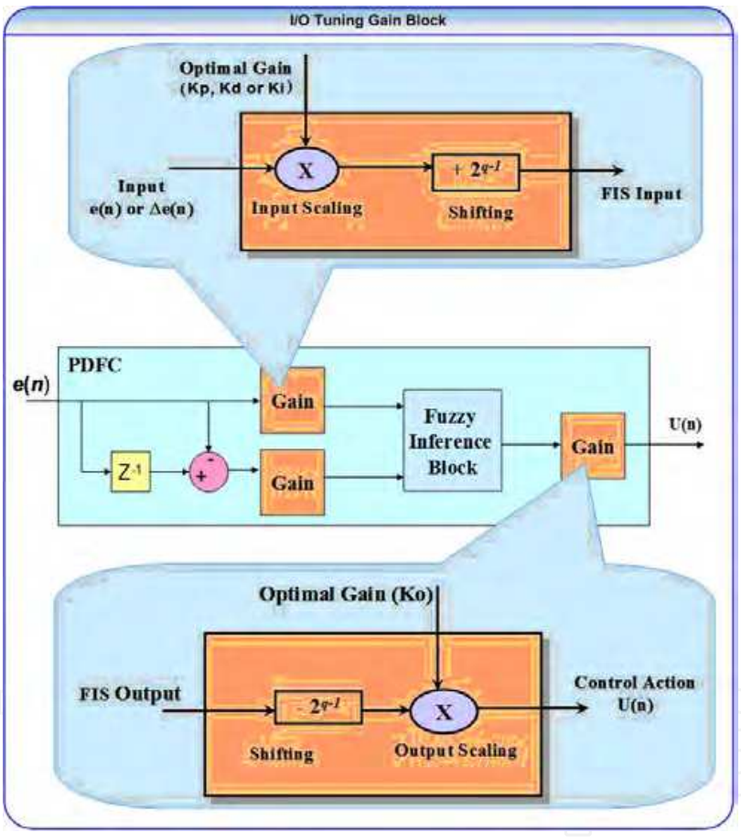


Fig. 5. Input/Output Tuning Block

Every "*" operator is synthesized to a *signed number multiplier* directly (Hassan et al., 2007), (Obaid et al., 2009), (Obaid et al., 1999). Fig. 6 shows the Tuning-gain block with more details. The fuzzy inference block in each PDFC can handle positive values only, and the error and its rate signals can have positive and negative values (Hassan et al., 2007), as the shifting process has been designed to convert the input variables range from $[-2^{q-1} \rightarrow 2^{q-1}-1\ ]$ to $[\,0 \rightarrow 2^{q}-1\,]$. This process implies adding the number (2^{q-1}) to the input variable. This addition has been designed by inverting the last bit (MSB) of the input variable (Hassan et al., 2007), (Obaid et al., 2009), (Obaid et al., 1999). The shift process at the

output has been designed using subtraction, instead of addition, to convert the range of the output variable from $[0 \rightarrow 2^{q}-1]$ to $[-2^{q-1} \rightarrow 2^{q-1}-1]$. This specification will increase the flexibility of the proposed design.

Fuzzifier Block

The overlapping degree (V) in the proposed design is two, which means that at each time instance there are two active, (have nonzero membership values), fuzzy sets for each input variable at maximum. The proposed fuzzification process has been designed using two fuzzifier blocks, one for each input variable. The fuzzifier block implies the fuzzification process by taking the input and producing four output values. These values represent the sequence numbers of the two active fuzzy sets (e1, e2 and de1, de2) and the membership degrees of the variable for each one of them (μe1, μe2 and μde1, μde2). The memory base was designed using ROM. The use of ROM is better than RAM when the programmability is directly achieved by the implementation techniques (as in the case of FPGA) (Barriga et al., 2006). The fuzzifier block was designed using memory based membership functions (MBMSF) (Solano et al., 1997), (Barriga et al., 2006). This method reduces the restrictions of the fuzzy set shapes, even it needs a smaller memory size than other method such as the arithmetic method. The memory model has been implemented with maximum possible membership values in the proposed design, where the maximum coded in p values is $(2^{p}-1)$, where p=4 bits in the 6-bits version of the PIDFC, and p=6 bits in the 8-bits version of the PIDFC. This dictates that the summation of membership values of two consecutive fuzzy set is always equal to $(2^{p}-1)$. Each word in the MBMSF is divided into two parts. The first part represents the sequence number of the active fuzzy set (3-bits, in both versions). Assigning 3 bits for the sequence number of the fuzzy sets, gives the controller flexibility to accept for each input variable for up-to 8 fuzzy sets. The second part of the memory word is p bits data word which represents the membership value of the input in the active fuzzy set. The total memory length for each input is equal to (2^{q}).

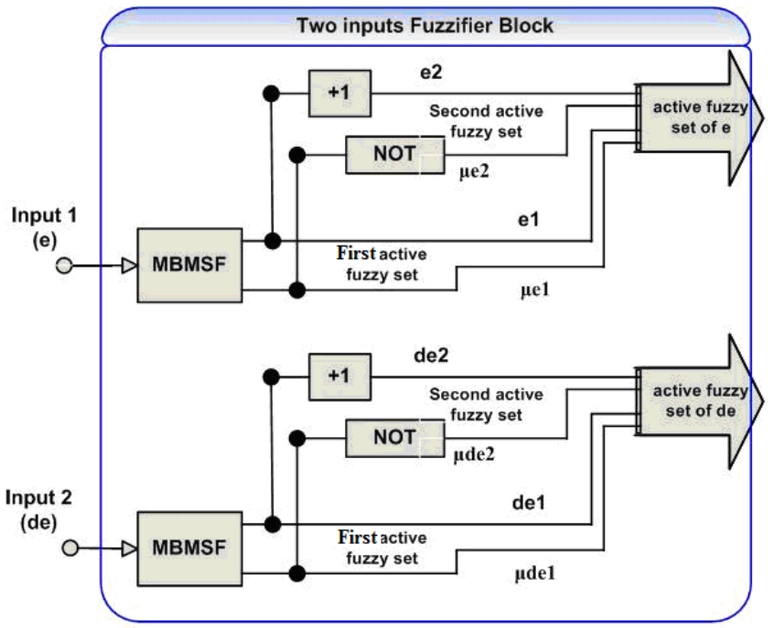


Fig. 6. Two Inputs Fuzzifier Block.

Inference Engine Block

The inference engine consists of three blocks: rule selector, rule memory, and minimum circuit as shown in Fig. 7. Different mechanisms have been used to minimize both the calculation time and the area of the fuzzy inference system; among the most interesting methods is the active rules selector concept (Hassan et al., 2007), (Solano et al., 1997), (Obaid et al., 2009), (Barriga et al., 2006), (Huang and Lai, 2005), (Obaid et al., 1999). This block uses the information from fuzzifier, which belongs to the active fuzzy sets to launch only active rules. This reduces the number of processed rules. Furthermore, by using an active rule selector, the number of rules to be processed will be reduced according to this equation (Hassan et al., 2007):

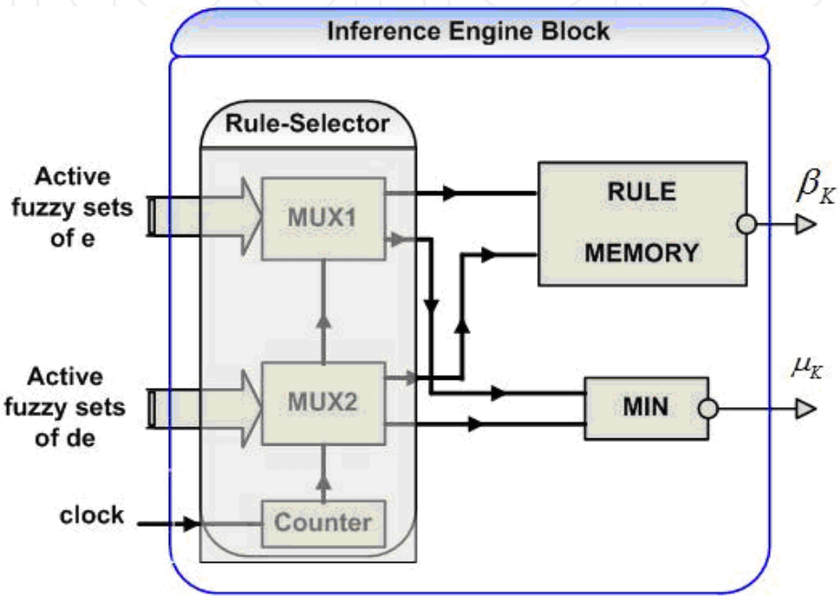


Fig. 7. Inference Engine Block

$$\text{Number of active rules} = V^m \tag{8}$$

Where m is the number of inputs, and V is the maximum number of overlapped fuzzy sets. In the proposed design, it is assumed that m = 2 and V = 2. Hence, the number of active rules at each time is: $V^m = 2^2 = 4$ rules. In each counter cycle, the membership degrees delivered from the two multiplexers are combined through the minimum circuit to calculate the applicability degree of the rule (μ_k), while the active fuzzy set sequence numbers are combined directly to address a rule memory location that contains the corresponding rule consequent ($ß_k$). The rule memory is a ($2^{2\times3} \times q$) bits ROM, and each word in it represents the position of the output singleton membership functions of one rule.

Defuzzifier Block

The defuzzification process in the defuzzifier block has been designed using the *Centroid* method. The four main components that represent the proposed defuzzifier are: two accumulators, one multiplier, and one divider. The defuzzifier block accepts the information from the inference engine (four (μ_k), and four ($ß_k$) each time), and produces an output (crisp set) to the output –tuning gain block, as shown in Fig. 8.

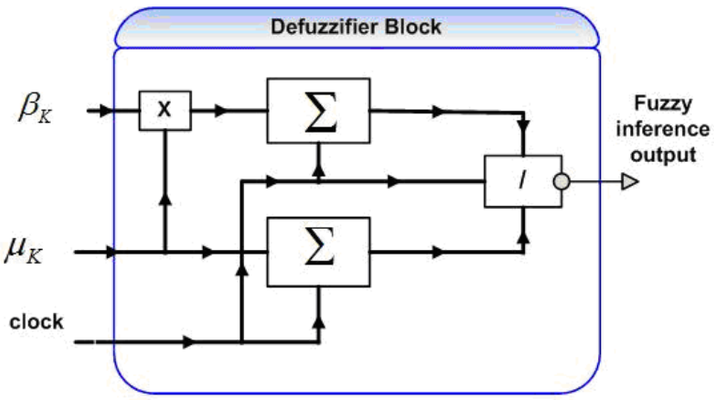


Fig. 8. Defuzzifier Block.

		e							
		NB	NM	NS	NZ	PZ	PS	PM	PB
de	NB	-1	-1	-1	-0.7	-0.7	-0.4	-0.1	0.1
	NM	-1	-1	-0.7	-0.7	-0.4	-0.1	0.1	0.1
	NS	-1	-0.7	-0.7	-0.4	-0.1	0.1	0.1	0.4
	NZ	-0.7	-0.7	-0.4	-0.1	0.1	0.1	0.4	0.7
	PZ	-0.7	-0.4	-0.1	-0.1	0.1	0.4	0.7	0.7
	PS	-0.4	-0.1	-0.1	0.1	0.4	0.7	0.7	1
	PM	-0.1	-0.1	0.1	0.4	0.7	0.7	1	1
	PB	-0.1	0.1	0.4	0.7	0.7	1	1	1

Table 2. Fuzzy Rules.

The defuzzifier block was designed with two stages to reduce the memory size of the target device. Both accumulators are reset every four clock cycles to receive the next four active rules of the next input. Note that μ_k and $ß_k$ are delivered from the inference engine in series during four consecutive clock cycles, instead of being produced in parallel in one clock cycle. This will reduce the used area of the FPGA device, at the expense of increasing time interval between input latching and output producing (Hassan et al., 2007). However, during the design, whenever a trade off between area and speed is found, it is directed to optimize (reduce) area at the expense of speed reduction, since the maximum time delay caused by controller is still much less than the minimum sampling time in many control systems. Even less than other controllers proposed in the literature (Poorani et al., 2005), (Tipsuwanpornet al., 2004), (Hassan et al., 2007), (Obaid et al., 2009), (Alvarez et al., 2006), (Lund et al., 2006), (Obaid et al., 1999). Here, the multiplication process was designed at the *behavioral* level (the method used in the tuning-gain block). The only difference is that another library called "*IEEE.std_logic_unsigned*" must be used instead of the library "*IEEE.std_logic_signed*", to ensure that the produced multiplier after the synthesis process is an unsigned number multiplier (because the proposed fuzzy inference block can only handle positive numbers only) (Hassan et al., 2007), (Obaid et al., 2009). The group involves eight triangular membership functions for each input variable, eight singleton membership functions for output variable, and the rule table of 64 rules has been used in the proposed

PIDFC, as shown in Fig. 9 and Table II. The use of a singleton membership function is to increase the computation speed versus low complexity (Leonid , 1997). And also for the majority of applications, using singleton fuzzy sets is more sufficient (Ying, 2000).

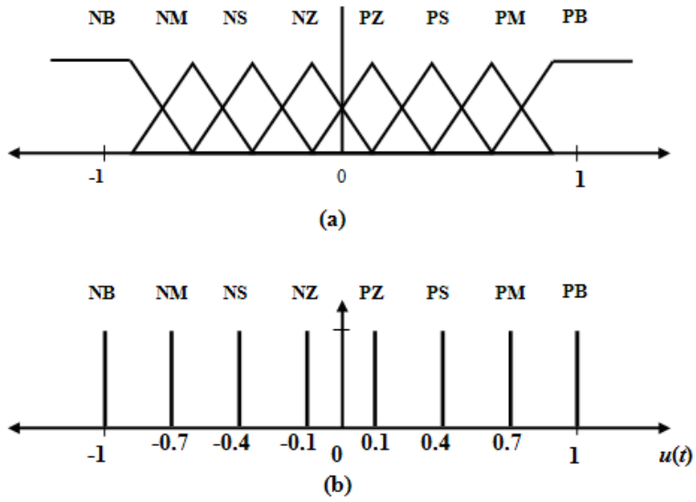


Fig. 9. (a) Inputs Membership Functions, (b) Output Membership Functions.

7. FPGA design considerations

The proposed device for the hardware implementation is the Virtex FPGAs family from Xilinx Company. Vertex FPGAs family is a useful device to the proposed design, it has internal RAM block. Virtex FPGAs consist of several large block memories. These complement the Look Up Table (LUT). This performance is very useful because the fuzzy system always needs large memory to store fuzzy sets information and rules table (Xilinx Company, 2009). The final design of the PIDFC has (3*q + 36) pins, four 8-bit input ports, two q-bits input ports and one q-bit output ports as well as 4 control signal pins, Table (III) lists the port names, sizes, and types.

Port name	Port size (bit)	Port type
Desired output	q	**Input data**
Plant output	q	**Input data**
Control action	q	**Output data**
K_p	8	**Input data**
K_d	8	**Input data**
K_i	8	**Input data**
K_o	8	**Input data**
sw1	1	**Control signal**
sw0	1	**Control signal**
Reset	1	**Control signal**
Clock	1	**Control signal**

Table 3. Port Names, Sizes, and Types Which Used In the Proposed Controller.

Note that the Reset signal does not change the contents of ROM or gain coefficients latches. However, the contents of ROM, as stated before can not be altered during the operation of the controller. The clock speed has a maximum frequency of 40 MH. This is necessary to cover a wide range of systems with a high sampling time.

8. Simulation environments

The Altera Quartus II 9.0 program was used to get the compilation and timing test results as well as the synthesized design. The ModelSim simulation program was also used for the purpose of simulation for all tests with the proposed design. The same design was designed in Matlab environments in order to make comparisons. The ModelSim stores the simulation data in text files, these files are used in Matlab and convert it to decimal vectors, which are used to plot the analog responses. Fig. 10 shows the Coding and Simulation environments used with the proposed design.

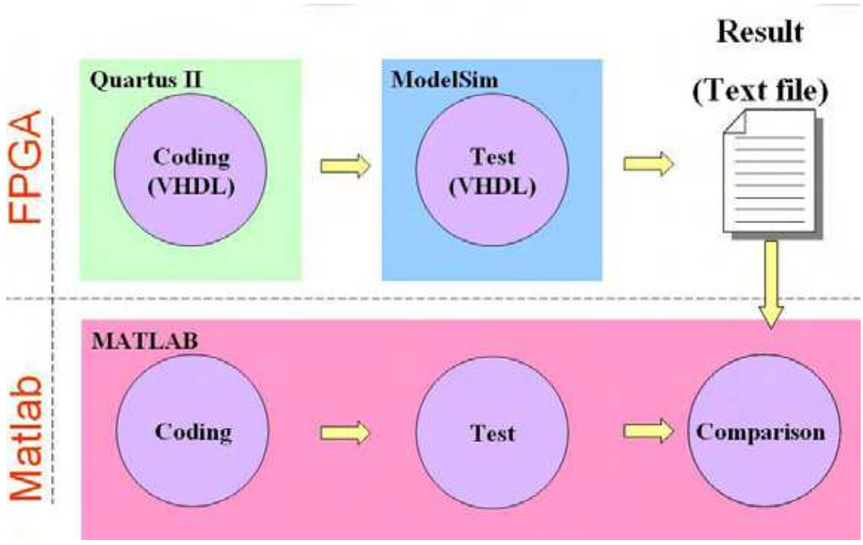


Fig. 10. Coding and Simulation Environments.

9. Timing analysis

The overall structure of the proposed PIDFC needs 16 clock cycles to complete one process. The input tuning-gain block needs one clock cycle. The fuzzification process needs one clock cycle. The inference engine needs four clock cycles to generate the four active rules consequent, $\beta_1, \beta_2, \beta_3, \beta_4$ and their corresponding applicability degrees $\mu_1, \mu_2, \mu_3, \mu_4$. Another four clock cycles are needed to calculate the terms $\sum_{k=1}^{4} \mu_k$ and $\sum_{k=1}^{4} \mu_k \times \beta_k$. Three of these four clocks are parallel to the four clocks of the inference engine, because the accumulation process starts after delivering the first rule consequent β_1 and its applicability degree μ_1. Subsequently, the division process starts and it takes eight clock cycles, which are split into two four-clock stages. The last clock cycle is needed to perform the output tuning-gain block.

10. Control surfaces test (comparison case study)

This test is performed to make sure that the fuzzy inference system used inside the FPGA-based controller (FBC) is working properly. This test involves generating the control surface using fuzzy sets and the rule shown in Fig. 9 and Table 2. This test has been used to make a comparison between both types of FBC with MSBC in order to evaluate the accuracy of the digital design implemented on FPGA with respect to the Matlab design. The control surfaces generated by MSBPD, 6FBC, MSBC and 8FBC are shown in Fig. 11 and Fig. 12. This surface reveals the effect of rounding and approximation processes (inside the FPGA design) on the result and also shows the accuracy of each version of the controller with respect to the Matlab-Based design. Generally, these statistics show that the surfaces generated by the fuzzy inference system of 8FBC are smoother than the surfaces generated by the fuzzy inference system of 6FBC with respect to MSBC, since the 8FBC has better accuracy and is adequate for this design.

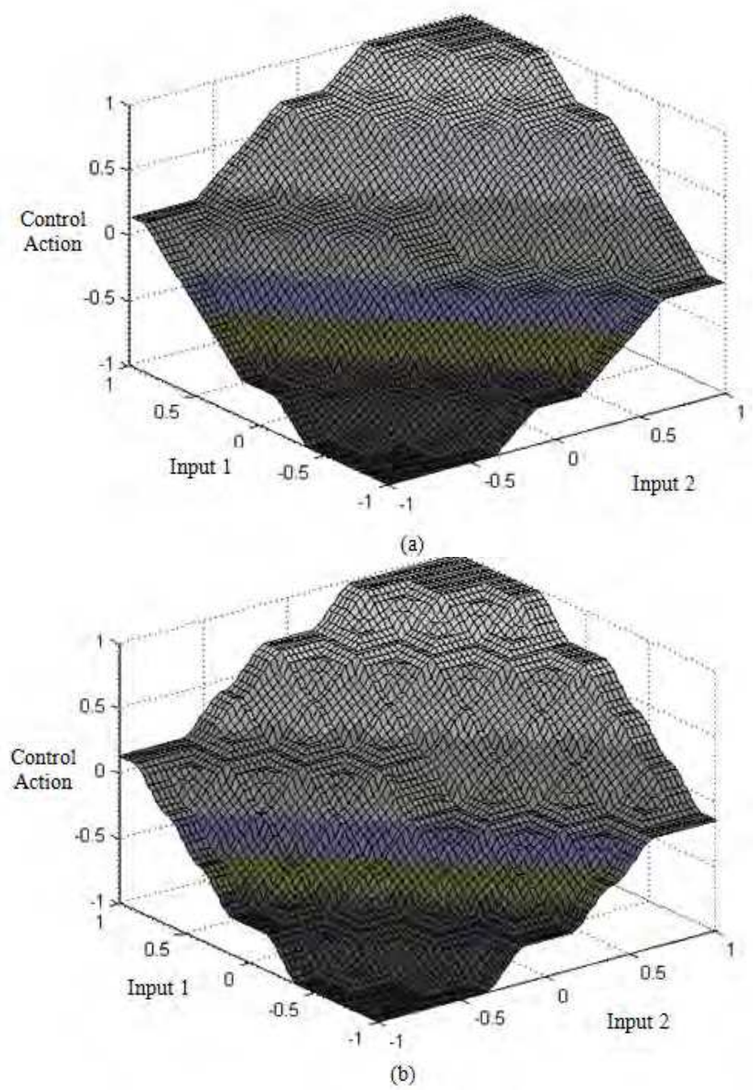


Fig. 11. (a) Control surface of MSBC (b) Control surface of 6FBC

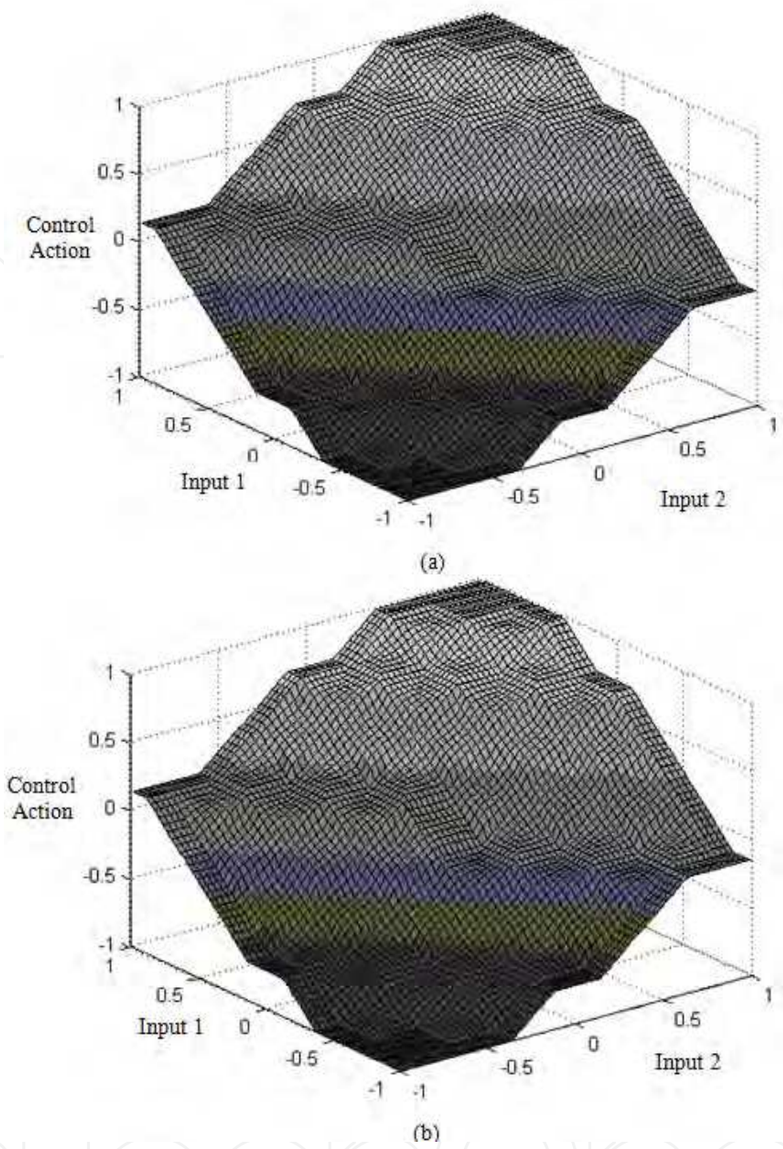


Fig. 12. (a) Control surface of MSBC (b) Control surface of 8FBC

11. The proposed controller with unity feedback control system

As mentioned before, the simplest and most usual way to implement a fuzzy controller is to realize it as a computer program on general purpose computers. Therefore, a comparison has been made between the simulation results of the two FPGA-based controller versions. The 6 bits **F**PGA-**B**ased **C**ontroller (6FBC) and the 8 bits **F**PGA-**B**ased **C**ontroller (8FBC), and the simulation results of the **S**oftware **B**ase **C**ontroller designed using **M**atlab (MSBC**)**. These comparisons are necessary to show that how FPGA-based design is close to Matlab-based design. The first level of this comparison was made using ModelSim as the test bench simulation before generating the results in a Text File. Subsequently, as explained before, these files were taken to the Matlab environment to do the comparison. The controllers

(6FBC, 8FBC, and MSBC) have been used in unity feedback control systems, and subjected to 0.5 step input. Mathematical models of five different plants have been used for this test. These consist of four case studies with linear systems and one case study with a nonlinear system. Each one of these plants has been designed in MATLAB software (for simulation in MATLAB), and also in non-synthesizable VHDL code (for simulation in ModelSim). Since each controller could serve as PDFC, PIFC, or PIDFC, a test was made for each one of these types. PSO was used to obtain the optimal values of the controller parameters that represent the tuning gains. Where the information of the proposed PSO algorithm is listed as follows: Population size: 100, W= [0.4 to 0.9], C1, C2=2, Iteration reached with every case is=1000 iteration path, and the particle searching range depends on the trial and is different in every case. All X-axes represent the time.

11.1 First order plant (first case study)

Many industrial processes such as level process can be represented by a first order model (Hu et al., 1999). Equation (9) shows the mathematical plant model (in s-plane). A discrete transfer function of this model has been obtained using the ZOH method, and the selected sampling period (T) is 0.1. Equation (10) shows the discrete transfer functions, (in z-plane). The searching range of the particle for this case ranges from [Xmin Xmax], and by using trial to reach the operational range with the universe of discourse. The optimal values of Kp, Kd, Ki, and Ko used in this test were selected using PSO; and listed in Table 4.

$$CS_1(s) = \frac{1}{s+1} \tag{9}$$

$$CS_1(z) = \frac{0.09516}{z-0.9048},\ \mathrm{T} = 0.1 \tag{10}$$

Range of Particle X	Controller type	Gain type	Value
$0.0001 \le X \le 5$	PIDFC	K_p	4.5111
		K_d	0.8751
		K_i	4.6875
		K_o	0.5625
$0.0001 \le X \le 3.5$	PIFC	K_p	0.6875
		K_i	2.4375
		K_o	1.0212
$0.0001 \le X \le 15$	PDFC	K_p	13.7501
		K_d	0.6251
		K_o	0.5011

Table 4. Optimal Gains Values Used With Cs1.

Fig. 13 shows the test bench simulation results using ModelSim. This test is generated using non-synthesizable VHDL code, and the controller gives action at 0.3 μs. The 6FBC has the same procedure except the real data which has different values. ModelSim stores the results as digital data in a text file; this file is manipulated in Matlab environments to change the data to decimal before using it as a comparison. The closed loop responses with 0.5 step input are shown in Fig. 14. In Fig. 14-a, it seems that the response has a large study state error. This is because the controller is PDFC, this controller affects the transient response (rise time, overshoot), but has no effect on the steady state error (at most). When the PIFC is applied on the first order system, the error disappears, and the system is first order, since there is no overshoot in this system (Fig. 14-b). When the PIDFC is applied for this system, as shown in Fig. 14-c, the response has a fast rising time with zero overshoot and error. However, although the 6FBC can sometimes give a response close to the MSBC response, at most, the 8FBC has smoother responses to MSBC than the 6FBC. The response performance of the proposed controllers is listed in Table 5.

Controller type	Error	Over shoot	Rising time	Settling time
PDFC	0.025	0.0	0.89	0.1
PIFC	0.0	0.0	0.3	0.4
PIDFC	0.0	0.0	0.15	0.198

Table 5. Responses Performance of the Proposed 8fbc with Cs1.

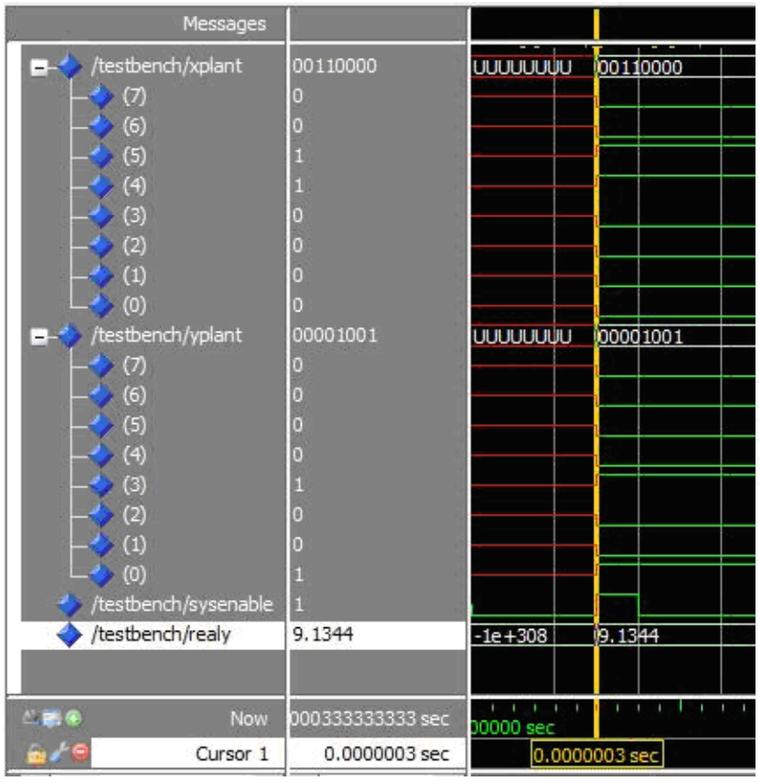
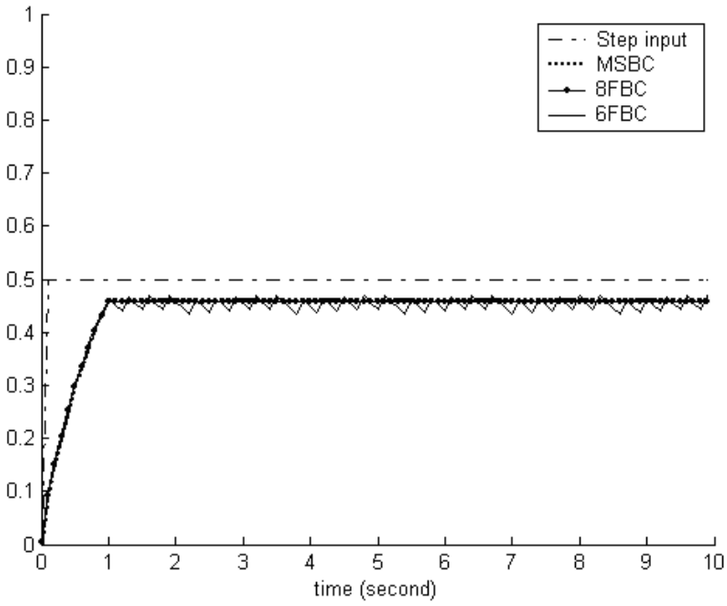
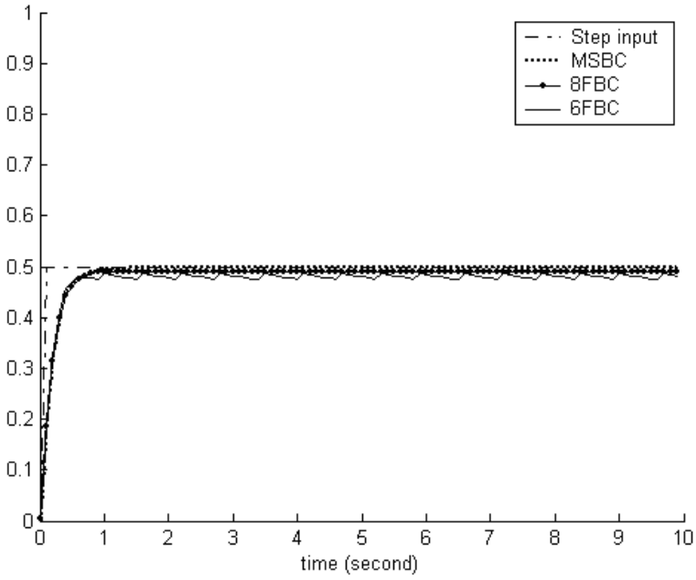


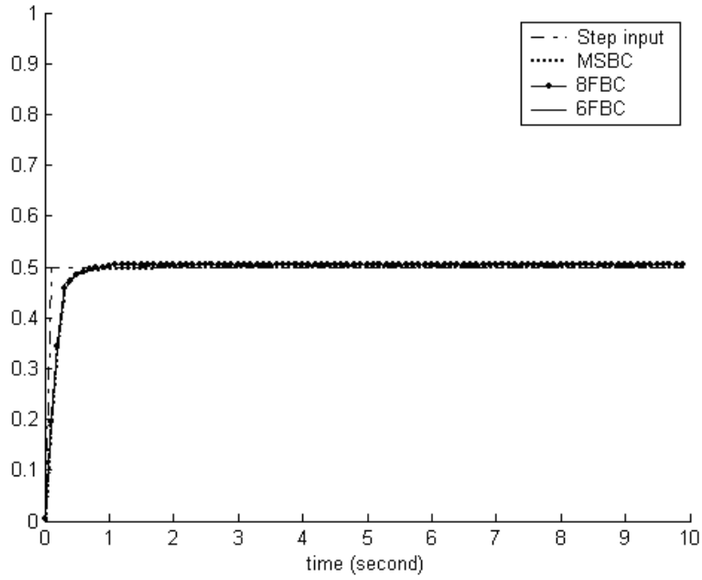
Fig. 13. Test Bench Results using ModelSim of the Proposed Controller (8FBC) with CS1 in unity Feedback Control system.


Step input
MSBC
8FBC
6FBC
time (second)

(a)


Step input
MSBC
8FBC
6FBC
time (second)

(b)



(c)

Fig. 14. First Order Linear Plant Controlled by (a) PDFC, (b) PIFC and (c) PIDFC.

11.2 Delayed first order plant (second case study)

The time delay occurs when a sensor or an actuator is used with a physical separation (Hu et al., 1999). Equation (11) shows the mathematical plant model (in *s-plane*). The discrete transfer functions of this model were obtained using the ZOH method, and the selected sampling period (T) is 0.1. Equation (12) shows the discrete transfer functions, (in *z-plane*). The searching range of the particle for this case is [Xmin Xmax], and by using trial to reach the operational range with the universe of discourse. The optimal values of K_p, K_d, K_i, *and* K_o used in this test were selected using PSO; and listed in Table 6.

Range of Particle X	Controller type	Gain type	Value
$0.0001 \le X \le 1.9$	PIDFC	K_p	1.4372
		K_d	1.687
		K_i	0.5625
		K_o	0.437
$0.0001 \le X \le 1.9$	PIFC	K_p	0.501
		K_i	1.5
		K_o	0.51
$0.0001 \le X \le 6$	PDFC	K_p	5
		K_d	0.125
		K_o	0.375

Table 6. Optimal Gains Values Used With CS2.

$$CS_2(z) = z^{-2} \times CS_1(z) \tag{11}$$

$$CS_2(z) = z^{-2} \times \left(\frac{0.09516}{z - 0.9048} \right) \quad , \text{T} = 0.1 \tag{12}$$

Fig. 15 shows the test bench simulation results using ModelSim. This test is generated in the same procedure as explained before. The controller gives action at 0.3 μs (Fig. 15), the delay with the systems affects the beginning of the real data (the response). The closed loop responses with 0.5 step input are shown in Fig. 16. In Fig. 16-a, again the response has a large study state error with the PDFC. When the PIFC is applied on the first order system (see Fig. 16-b), the error disappears with 8FBC. The 6FBC has a large steady state error, as the responses of the systems that use 8FBC are closer to the MSBC responses. When the PIDFC is applied for this system, as shown in Fig. 16-c, the response is close to the responses when using the PIFC. In all this, 8FBC has smoother responses to the MSBC than the 6FBC. The responses performance of the 8FBC are listed in Table 7.

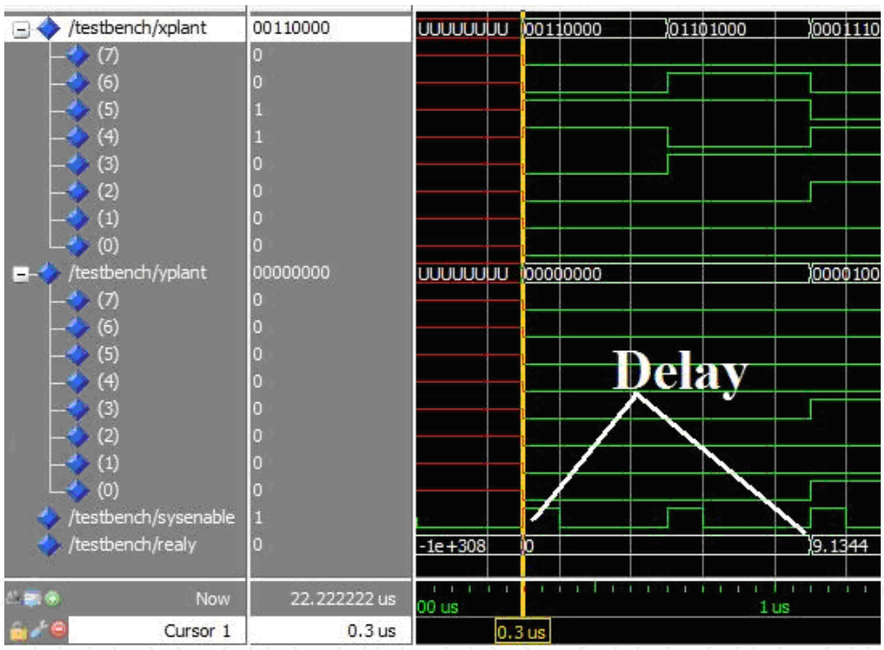
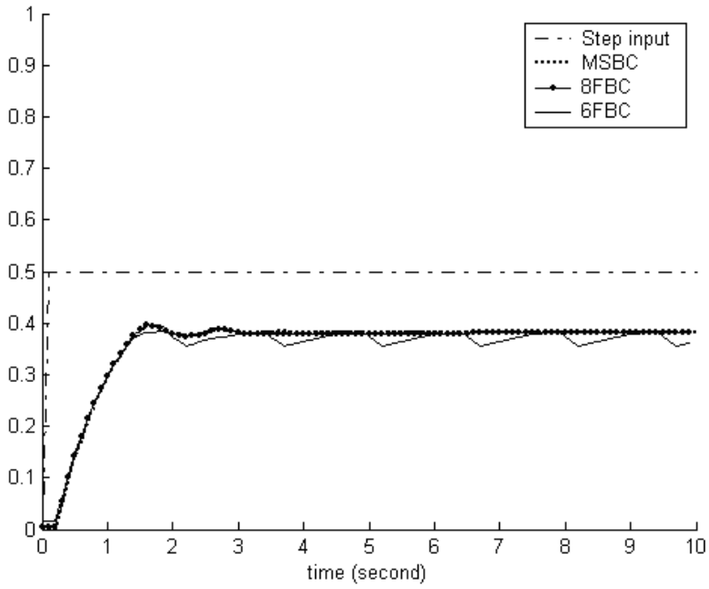


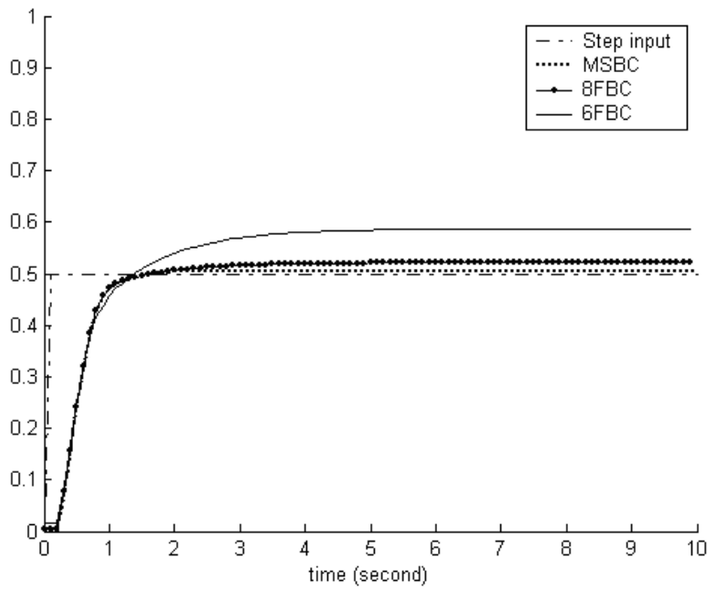
Fig. 15. Timing Diagram using ModelSim of the Controller (8FBC) with CS2 in unity Feedback Control system.

Controller type	Error	Over shoot	Rising time	Settling time
PDFC	0.11	0.01	1.12	1.2
PIFC	0.01	0.0	0.8	0.9
PIDFC	0.02	0.0	0.48	0.49

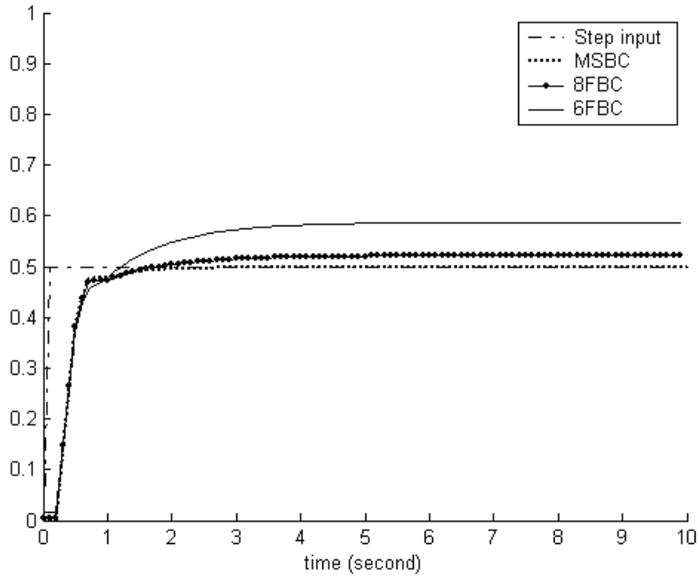
Table 7. Responses Performance of the Proposed 8fbc with CS2.


Step input
MSBC
8FBC
6FBC
time (second)

(a)


Step input
MSBC
8FBC
6FBC
time (second)

(b)



(c)

Fig. 16. Delayed first order linear plant controlled by(a) PDFC, (b) PIFC and (c) PIDFC.

11.3 Second order plant (third case study)

The position control of an AC motor process or temperature control can be represented by a second order model (Hu et al., 1999). Equation (13) shows the mathematical plant model (in *s-plane*). Discrete transfer functions of this model were obtained using the ZOH method, and the selected sampling period (T) is 0.2. Equation (14) shows the discrete transfer functions, (in *z-plane*). The searching range of the particle for this case is ranging as [Xmin Xmax], and by using trial to reach the operational range with the universe of discourse. The optimal values of K_p, K_d, K_i, *and* K_o used in this test were selected using PSO; and listed in Table 8.

$$CS_3(s) = \frac{1}{s^2 + 4s + 3} \tag{13}$$

$$CS_3(z) = \frac{0.01544\ z + 0.01183}{z^2 - 1.368\ z + 0.4493},\ \mathrm{T} = 0.2 \tag{14}$$

Fig. 17 shows the test bench simulation results using ModelSim for 8FBC; this test is generated using the same procedure as explained. The controller gives action at 0.3 μs (Fig. 17). This means the same action with CS1 and CS2, which represent the linear models. The closed loop responses with 0.5 step input are shown in Fig. 17. CS3 is a second order plant, and has a steady state error with non-controlled response. In Fig. 18-a, when PDFC is applied the overshoot is limited by the action of this controller, but the response still has a steady state error. When the PIFC is applied to this system (see Fig. 18-b), the error is disappears with 8FBC, and the system still has overshoot. The 6FBC has a rough and non-smooth response as can be seen in the control action figures where sharp spikes appear along the steady state part, while the responses of the systems that use 8FBC are closer to the MSBC responses. When the PIDFC is applied for this system, as shown in Fig. 18-c, the

8FBC response is close to the responses using MSBC, with zero error and little overshoot. The Responses Performance of the proposed 8FBC with CS3 is listed in Table 9.

Range of Particle X	Controller type	Gain type	Value
$0.0001 \le X \le 8.5$	PIDFC	K_p	5.0191
		K_d	7.101
		K_i	1.6875
		K_o	0.937
$0.0001 \le X \le 2$	PIFC	K_p	1.02
		K_i	1.812
		K_o	1.75
$0.0001 \le X \le 15$	PDFC	K_p	14.625
		K_d	6.021
		K_o	1.062

Table 8. Optimal Gains Values Used With CS3.

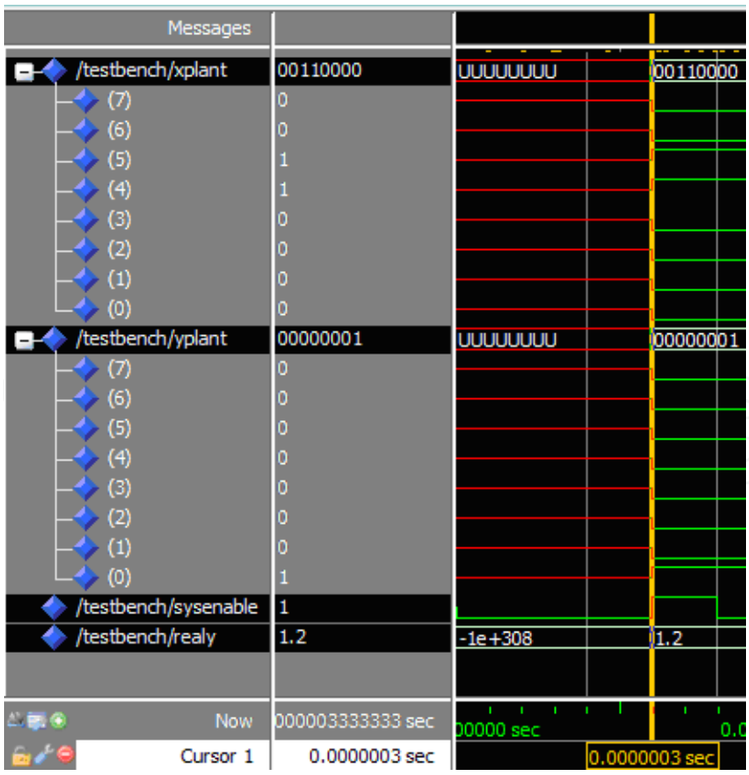
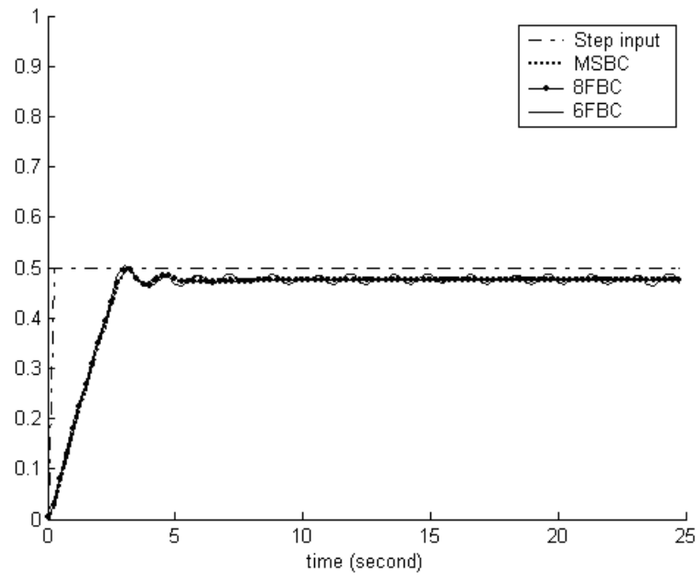
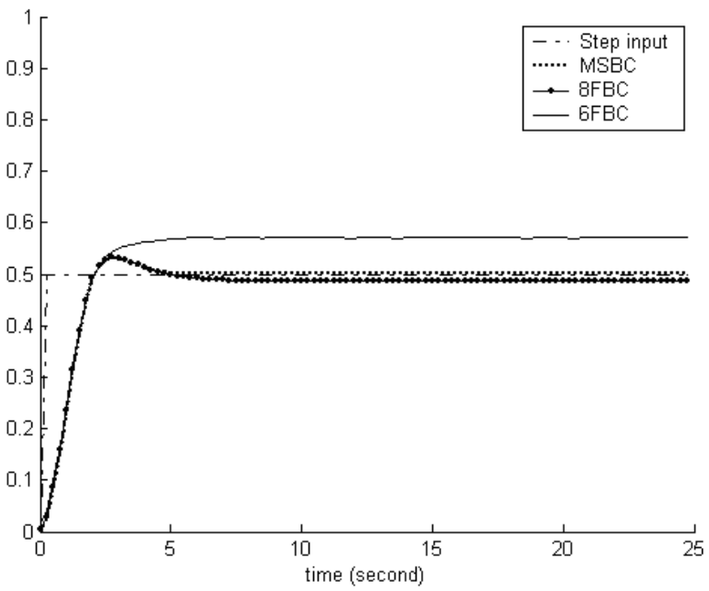


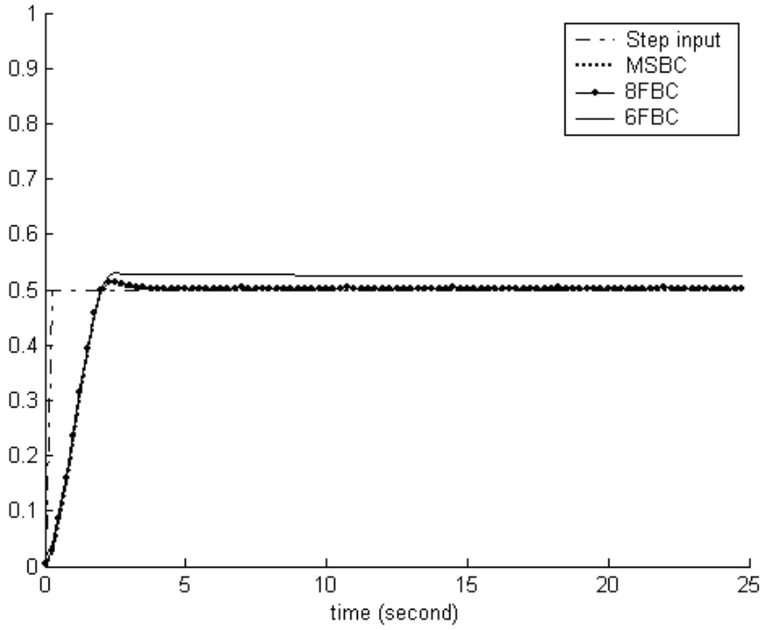
Fig. 17. Timing Diagram using ModelSim of the Controller (8FBC) with CS3 in unity Feedback Control system.


Step input
MSBC
8FBC
6FBC
time (second)

(a)


Step input
MSBC
8FBC
6FBC
time (second)

(b)



(c)

Fig. 18. Second order linear plant controlled by(a) PDFC, (b) PIFC and (c) PIDFC.

Controller type	Error	Over shoot	Rising time	Settling time
PDFC	0.02	0.02	2.1	2.3
PIFC	0.005	0.03	1.89	2
PIDFC	0.0	0.005	1.6	1.8

Table 9. Responses Performance of the Proposed 8fbc with CS3.

11.4 Delayed second order plant (fourth case study)

The time delay occurs when a sensor or an actuator are used with a physical separation (Hu et al., 1999). Equation (15) shows the mathematical plant model (in *s-plane*). Discrete transfer function of this model was obtained using the ZOH method, and the selected sampling period (T) is 0.2. Equation (16) shows the discrete transfer functions, (in *z-plane*). The searching range of the particle for this case is ranging as [Xmin Xmax], and by using trial to reach the operational range with the universe of discourse. The optimal values of K_p, K_d, K_i, *and* K_o used in this test were selected using PSO; and listed in Table 10.

$$CS_4(z) = z^{-2} \times CS_3 \tag{15}$$

$$CS_4(z) = z^{-2} \times \left(\frac{0.01544\ \ z + 0.01183}{z^2 - \ 1.368\ \ z + 0.4493} \right) \text{; T = 0.2} \tag{16}$$

Range of Particle X	Controller type	Gain type	Value
$0.0001 \le X \le 8.5$	PIDFC	K_p	2.11
		K_d	1.687
		K_i	0.5012
		K_o	0.375
$0.0001 \le X \le 2$	PIFC	K_p	0.253
		K_i	1.185
		K_o	1.251
$0.0001 \le X \le 15$	PDFC	K_p	7.18
		K_d	8.754
		K_o	0.503

Table 10. Optimal Gains Values Used with CS4.

Fig. 19 shows the test bench simulation results using ModelSim for 8FBC. This test is generated using the same procedure as explained before. The delay with the system only affects the value of the real data (response). The controller gives action at 0.3 µs. The closed loop responses with 0.5 step input are shown in Fig. 20. CS4 is the same model as CS3 but with delay. In Fig. 20-a, when the PDFC is applied, the overshoot is limited by the action of this controller, but the response has a large steady state error. In this case the 8FBC is very close to the MSBC while the 6FBC has a non-smooth response. When the PIFC is applied to

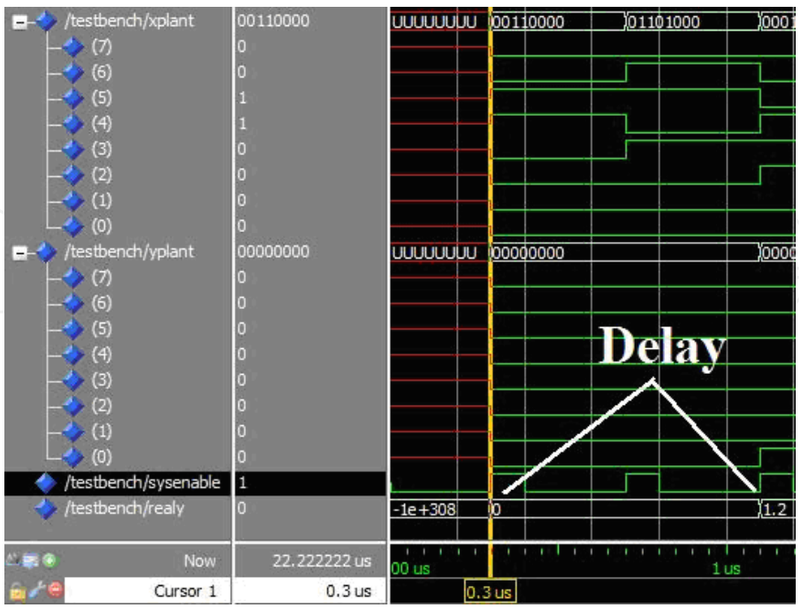
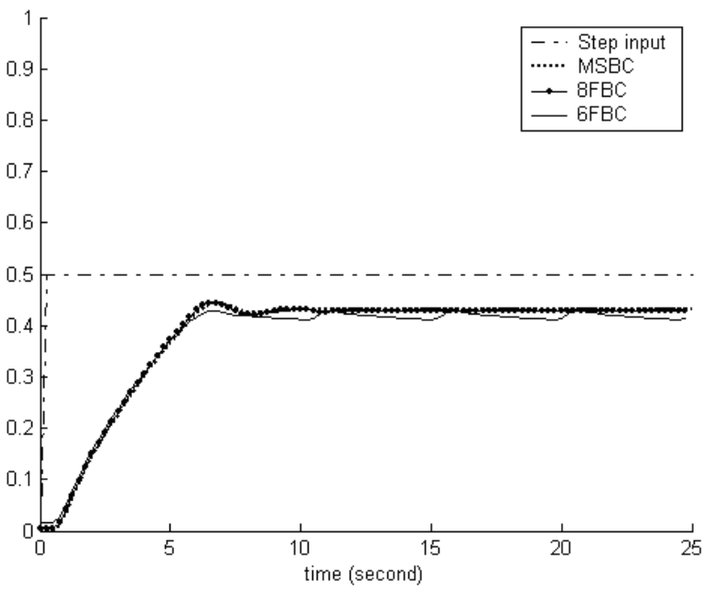
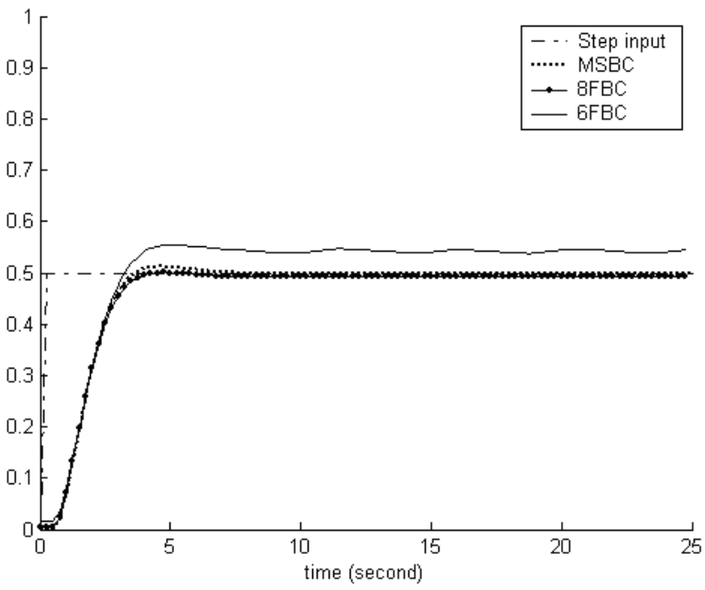


Fig. 19. Timing Diagram using ModelSim of the Controller (8FBC) with CS4 in unity Feedback Control system.

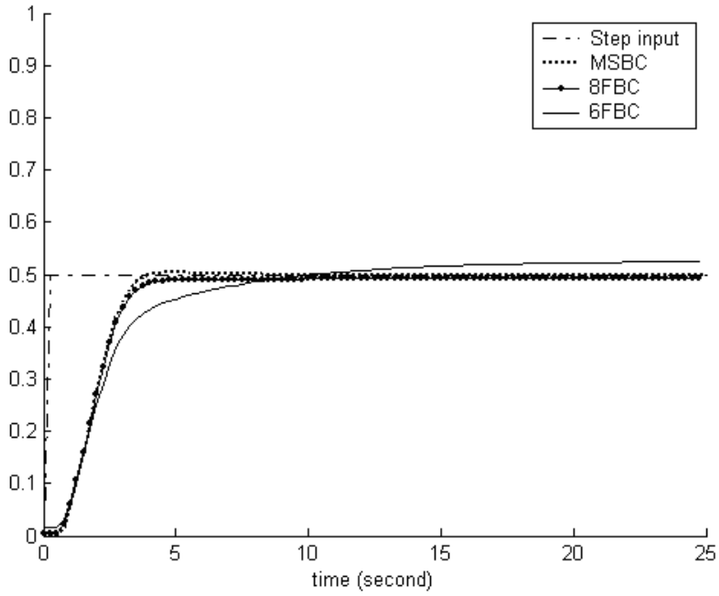
this system (see Fig. 20-b), the error disappears with the 8FBC with little overshoot. The 6FBC has a large steady state error while the responses of the systems that use 8FBC are closer to the MSBC responses. When the PIDFC is applied for this system, as shown in Fig. 20-c, the 8FBC response has better overshoot to the responses using MSBC, and is very close to the MSBC in the steady state response. The 6FBC has a long rising time with steady state error. The Responses Performance of the proposed 8FBC with CS4 is listed in Table 11.



(a)



(b)



(c)

Fig. 20. Delayed second order linear plant controlled by (a) PDFC, (b) PIFC and (c) PIDFC.

Controller type	Error	Over shoot	Rising time	Settling time
PDFC	0.07	0.02	4.9	5.3
PIFC	0.0	0.0	2.3	2.4
PIDFC	0.0	0.0	2.4	2.5

Table 11. Responses Performance of the Proposed 8FBC with CS4.

11.5 Non-linear plant model (fifth case study)

A mathematical model of nonlinear plant (inverted pendulum) has been used to test the PIDFC with unity feedback control system; this model is characterized by Equation (17) and Equation (18) (Passino and Yurkovich, 1998).

$$CS_5 = \ddot{y} = \frac{9.8\sin(y) + \cos(y)\left[\dfrac{-\bar{u} - 0.25\dot{y}^2\sin(y)}{1.5}\right]}{0.5\left[\dfrac{4}{3} - \dfrac{1}{3}\cos^2(y)\right]} \tag{17}$$

$$\dot{\bar{u}} = -100\,\bar{u} + 100\,u \tag{18}$$

The first order filter on u to produce $\bar{u}$ represents an actuator. Assuming the initial conditions $y(0) = 0.1$ radians (= 5.73 deg.), $\dot{y}(0) = 0$, and the initial condition for the actuator state is zero. For simulation of the fourth-order, the Runge-Kutta method was used with an integration step size of 0.01 (Passino and Yurkovich, 1998), (Obaid et al., 1999). Again, this plant has been designed using MATLAB software (for simulation in MATLAB), and in

VHDL code (for simulation in ModelSim). A special package was designed in VHDL code to represent the trigonometric functions and fourth-order Runge-Kutta method, which are not available in Quartus II (or in ISE) standard libraries (Obaid et al., 1999). The searching range of the particle for this case is [Xmin Xmax], and by using trial to reach the proposed algorithm, the values of K_p, K_d, K_i, *and* K_o used in this test were selected using PSO. These values are listed in Table XII.

Range of Particle X	Controller type	Gain type	Value
$0.0001 \le X \le 11.5$	**PIDFC**	K_p	1.1012
		K_d	10.1103
		K_i	1.5013
		K_o	5.0032

Table 12. Optimal Gains Values Used With CS5.

Fig. 21 shows the test bench simulation results using ModelSim for 8FBC and the controller gives an output at 0.7 μs after the input latching (Fig. 21). The 6FBC has the same procedure in ModelSim and produces an output at 0.62 μs. The Responses Performance of the proposed controller with CS5 is listed in Table 13. Where the bound of the settling time of the pendulum to reach its initial position with the force applied to the cart is -0.02 and +0.02 with both versions. The first time of the pendulum reach s the initial position is listed as the rising time. When using a nonlinear system for testing, both versions (6FBC and 8FBC) provide generally good responses although there is some oscillation. One must not be deceived by the steady state error that appears in Fig. 22, as it represents less than 1% of the output range in the case of 6FBC and less than 0.5% of the output range, in the case of 8FBC. The absolute mean difference between the nonlinear plant response, using MSBC, and the

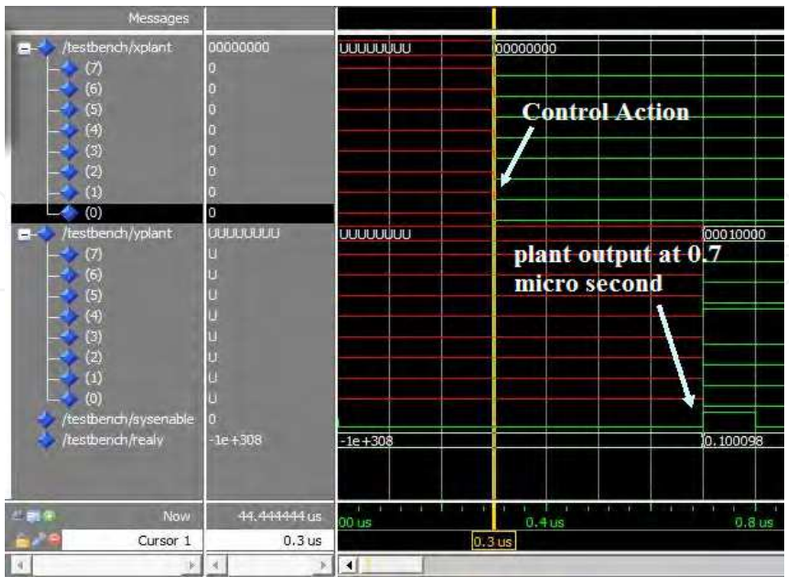


Fig. 21. Timing Diagram using ModelSim of the Controller (8FBC) with CS5 in unity Feedback Control system.

nonlinear plant response, using 6FBC is less than 0.017. The absolute mean difference between the nonlinear plant response, using MSBC, and the nonlinear plant response, using 8FBC is less than 0.006 as shown in Fig. 22. The experimental result is carried out by using the nonlinear inverted pendulum (the last case study CS5). The experimental data recorded to the inverted pendulum has been used in this test in unity feedback control system. This data has been recorded by Sultan (Sultan, 2006) to analyze, design & develop a control loop for the given inverted pendulum (with servomechanism). The pendulum reaches the initial position zero at 0.058 second with overshoot equal to 0.025 and undershoot equal to 0.02. Fig. 23 shows the experimental data simulation of the inverted pendulum with the PIDFC in 8-bit version. The controller (8FBC) provides a good control performance with respect to the simulation results of the same case as shown in Table 13.

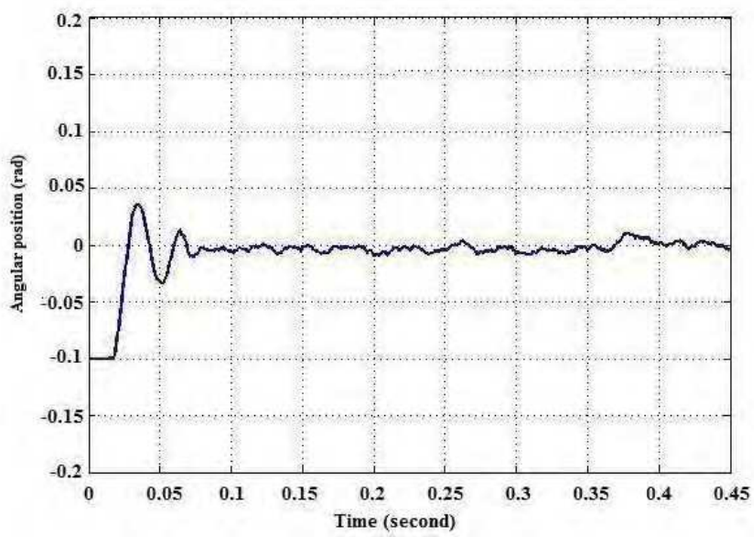


Fig. 23. Experimental data simulation of the PIDFC with the inverted pendulum in unity feedback control system.

Controller type	Error	Over Peak angle (rad)	Under Peak angle (rad)	Rising time	Settling time
6FBC	0.017	0.0	-0.01	0.13	0.1
8FBC	0.006	0.0	-0.01	0.13	0.1
6MSBC	0.0	0.018	-0.041	0.13	0.1
8MSBC	0.0	0.018	-0.041	0.13	0.1
Experimental case	0.0	0.025	0.024	0.029	0.058

Table 13. Responses Performance of the Proposed 8fbc with CS5.

12. Results comparison and discussion

The proposed design has been coded in Matlab environments as explained before. The aim of this test is to find to what extent the 6FBC and 8FBC responses are close to the MSBC responses with respect to the accuracy. In contrast to the 6FBC, the responses of the systems that use 8FBC are smooth (as MSBC responses). When the 6FBC or 8FBC is used as PIFC or

PIDFC, the system responses will settle at a value close to the response settling value of systems that use MSBC. The reason being the rounding error in the PI component, the proposed PIDFC consists of a PDFC and PIFC, the PIFC is a PDFC with a summation block at its output. This error could be positive or negative. Sometimes, during the summation process, the rounding error cancels itself. Commonly, in CS1 and CS2, it is clear that the responses have a steady state error with PDFC, because these systems do not have an overshoot, only a steady state error (no need to use PDFC), and this test is used to evaluate the multi structure in the proposed PIDFC. The absolute mean of differences between the MSBC and 6FBC was less than 0.07 with the linear systems and less than 0.017 with the nonlinear system, while the absolute mean of differences between MSBC and 8FBC was less than 0.017 with the linear systems and less than 0.006 with the nonlinear system. The proposed controller has good responses performance with respect to the classical controller proposed in the literature, and also better responses as compared it with other type of fuzzy PID controller.

CS1, CS2 and CS3 have been used in the work proposed in (Hu et al., 1999). In this work, Genetic algorithm (GA) based optimal fuzzy PID controller was used with new methodology (Matlab-based). Table 14 lists the comparison of the responses performance of this work with respect to the proposed PIDFC (8FBC) with CS1, CS2 and CS3. This comparison consists of rising time (Tr), settling time (Ts) and overshoot (OV). The proposed PSO-PIDFC with the linear cases CS1 and

CS2 have no overshoot and a short rising and settling time with respect to other types of controller designed in the literature with the same models. For the CS3, there is a 0.005 overshoot and short rising and settling time with respect to other types of controller designed in the literature with the same model. The proposed PSO-PIDFC with the nonlinear cases (CS5) have zero overshoot and 0.13, 0.1 rising and settling time respectively, while other types of controller with the same nonlinear model have an overshoot and longer rising and settling time. In CS5, a mathematical model of nonlinear plant (inverted pendulum) was used to test the controller with a unity feedback control system. This case has been used in (Jain et al., 2009), (Masmoudi et al., 1999) and (Jain et al., 1999) (Matlab-based) with different types of controllers. In (Jain et al., 2009), Bacterial Foraging (BF) algorithm was used for tuning the parameters of the PID controller for optimal performance, while (Jain et al., 1999) used a comparison between Evolutionary Algorithms namely Gas (Genetic Algorithms), and Swarm Intelligence i.e. PSO and BG. In (Jain et al., 1999) as there was no need to know the value of rising time as they used a reference input equal to zero. It is not considered by this author either. Therefore, a comparison is made with the proposed 8FB (PIDFC) with those presented in (Jain et al., 2009), (Masmoudi et al., 1999) and (Jain et al., 1999). This comparison is listed in Table 15. In the case of (Jain et al., 1999), PSO had the best responses with respect to the other methods proposed by this author, hence, we will compare with PSO and ISE only. Other comparison has been made to the proposed design with respect to FPGA chip resources. This comparison involves the utilization of the chip resources in the proposed FPGA-based PIDFC with respect to the other FPGA-based controllers proposed in the literature. It also involves a comparison with respect to the time required per one action with the maximum frequency. This comparison was made after compiling the design using the ISE program provided by Xilinx Company, because this tool provides a clear Report for the chip resources, even it was used by the authors in the literature. This comparison is listed in Table 16.

Case	Performance	Proposed FPGA-based PIDFC with PSO	GA - Fuzzy PID In (Hu et al., 1999)	GA -Optimal PID In (Hu et al., 1999)
CS1	Tr (s)	0.19	0.16	0.2
	Ts(s)	0.23	0.2	0.36
	OV	0.0	0.0	0.0039
CS2	Tr (s)	0.48	0.16	0.38
	Ts(s)	0.49	0.46	0.74
	OV	0.0	0.0051	0.0162
CS3	Tr (s)	1.6	0.74	0.88
	Ts(s)	1.8	2.34	1.34
	OV	0.005	0.0622	0.0107

Table 14. Performance Comparison of the PIDFC with the Work Proposed by Hu et al. In (Hu et al., 1999)

Performance	The proposed Controller In Experimental Case with CS5	The Proposed FPGA-based PIDFC With PSO	Optimal PID In (Jain et al., 2009) With BG	Fuzzy logic controller in (Masmoudi et al., 1999)	Optimal PD-PI In (Jain et al., 1999) With PSO
Ts (s)	0.058	0.1	0.4	0.21	2.4
Tr(s)	0.029	0.13	0.2	0.22	---
Peak angle (rad)	0.025	0.0	0.178	0.75	0.00127

Table 15. Performance Comparison of the PIDFC (8FBC) With Those Proposed In (Jain et al., 2009), (Masmoudi et al., 1999) and (Jain et al., 1999) By Using CS5.

References	Number of CLBs	Number of IOBs	Frequency	Time per action
PIDFC	494	68	40 MHz	0.3 µs
(Poorani et al., 2005)	757	39	8 KHz	41.1 ms
(Tipsuwanpornet al., 2004)	---	---	40.55 MHz	2.1 µs
(Hassan et al., 2007)	1394	61	40.245 MHz	0.421 µs
(Alvarez et al., 2006)	3492	51	100 MHz	1.4 µs
(Lund et al., 2006)	63	---	20 MHz	1 µs

Table 16. FPGA Chip Resources Comparison between the PIDFC and Other Type of Controllers Proposed In the Literature.

13. Conclusion

From the design and simulation results of the PIDFC, it can be concluded that: Higher execution speed versus small chip size is achieved by designing PIDFC with a simplified fuzzy algorithm as a parallel structure of PDFC and PIFC and also by designing PIFC by accumulating the output of the PDFC. These methods significantly reduce the number of rules needed. It also enables the controller to work as a PIFC, PDFC or PIDFC depending on two external signals to provide high-flexibilities with different applications. The controller needs 16 clock cycles to generate an output with a maximum clock frequency of 40 MHz. Therefore, the proposed controller will be able to control a wide range of systems with a high sampling rate. Higher flexibility versus good control performance is achieved by designing tuning-gains block at each input/output stage. This block involves a tuning by scaling the universe of discourse for the input/output variables (renormalization). This block makes the controller chip accept the PSO-based optimal scaling gains, and also enables the digital controller chip able to accept unsigned inputs. The PSO algorithm has better simulation results than other intelligent optimization methods proposed in the literature such as genetic algorithm. This block is very important and is useful for providing a best tuning case for the universe of discourse.
In addition, it makes the design applicable for different systems without requiring reprogramming the controller chip. Higher execution speed and small chip size versus acceptable accuracy is achieved by designing each one of the scaling gains as two parts: integer and fraction, and perform all mathematical operations using integer number algorithms, which are smaller in the implementation size than floating number algorithms, and even faster. Sufficient design accuracy can be achieved with 8FBC in Particular. 8FBC is superior to 6FBC since it presents higher accuracy versus moderately low target device utilizations. 8FBC was able to produce a control action in 0.3 μs after input latching (the computational time of the controller is 0.3 μs). The 8FBC produced responses approximately similar or better than the MSBC compared with the 6FBC or with the results in the literature. The absolute mean of differences between the responses of the 6FBC and the MSBC, was less than 4% of the output range, for the linear plants, and less than 0.5% of the output range for the nonlinear plant, while the absolute mean of differences between the responses of 8FBC and the MSBC, was less than 1% of the output range, for the linear plants, and less than 0.3% of the output range for the nonlinear plant. Both versions showed some error at the steady state part of the response when serving as PIFLC or PIDFLC because of the accumulation of the rounding error at the summation block. This error depends on the rounding error; therefore it becomes larger when using the 6FBC than when using 8FBC. As a result, the proposed controller could be used to control many industrial applications with high sampling time. Its small size versus high speed makes it a good choice for other applications, such as robots. It is hoped that some future work could settle down the feasibility of the suggestions: Increasing the number of the first part of the fuzzy set in the MBMSF inside the fuzzifier block more than (3-bits) could make the design accept more than 8 fuzzy sets at each input. Increasing the number of bits of the entire design may be useful in decreasing the error in the PIFC component (in some cases) of the controller at the expense of increasing design area and processing time.

14. Acknowledgment

The authors would firstly like to thank God, and all friends who gave us any help related to this work. Other Appreciation goes to the Assist chancellor amend to Diyala University-Iraq

(Prof. Dr Amer Mohammed Ibrahim) and the Dean of Collage of Engineering-Diyala University- Iraq (Assist. Prof. Dr. Adel Khaleel Mahmoud). Further appreciation goes to My Colleagues Mr. Waleed Fawwaz and Mrs. Areej Sadiq for their wide help and support. Finally, a heartfelt thank to our families and home countries.

15. References

Poorani, S., Priya, T.V.S.U., Kumar, K.U., and Renganarayanan, S., (2005) "*FPGA Based Fuzzy Logic Controller for Electric Vehicle*," Journal of the Institution of Engineers, Singapore, Vol. 45 Issue 5.

Jain, T., Patel, V., Nigam and M.J., "*Implementation of PID Controlled SIMO Process on FPGA Using Bacterial Foraging for Optimal Performance*," International Journal of Computer and Electrical Engineering, Vol. 1, No. 2, p: 1793-8198, June.

Tipsuwanporn, V., Intajag, S. and Krongratana, V., (2004) "*Fuzzy Logic PID Controller Based on FPGA for Process Control*," Proceedings of IEEE International Symposium on Industrial Electronics, Vol. 2, May 4-7, page(s):1495-1500.

Karasakal, O., Yesl, E., Guzelkaya, M. and Eksin, I., (2005) "*Implementation of a New Self-Tuning Fuzzy PID Controller on PLC*," Turk J Elec Engin, Vol.13, NO.2.

Hassan, M.Y., and Sharif, W.F., (2007) "*Design of FPGA based PID-like Fuzzy Controller for Industrial Applications*," IAENG International Journal of Computer Science, Vol. 34 Issue 2, November.

Kim, D., (2000) "*An Implementation of Fuzzy Logic Controller on the Reconfigurable FPGA System*," IEEE Transactions on Industrial Electronics, Vol. 47, No. 3, p: 703- 715, June.

Solano, S.S., Barriga, A., Jimenez, C.J and Huertas, J.L., (1997) "*Design and Application of Digital Fuzzy Controllers*", Proceedings of Sixth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'97), Vol. 2, July 1-5, page(s):869-874.

Islam, S., Amin, N., Bhuyan, M.S, Zaman, M., Madon, B. and Othman, M., (2007) "*FPGA Realization of Fuzzy Temperature Controller for Industrial Application*", WSEAS Transactions on Systems and Control, Volume 2, Issue 10, p: 484- 490, October.

Mann, G.K.I., Hu, B.G. and Gosine, R.G., (1999) "*Analysis of Direct Action Fuzzy PID Controller Structures*", IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, Vol. 29, No. 3, p: 371 – 388, June.

Seng, T.L., Khalid, M. and Yusuf, R., (1999) "*Tuning of a Neuro-Fuzzy Controller by Genetic Algorithm*", IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, Vol. 29, No. 2, P: 226- 236, April.

Leonid, R., Ghanayem, O. and Bourmistrov, A., (2000) "*PID plus fuzzy controller structures as a design base for industrial applications*", Engineering Applications of Artificial Intelligence, Elsevier Science Ltd, 13, p: 419-430.

Leonid, R. (1997). *Fuzzy Controllers*, first edition, Newnes.

Gabrielli, Gandolfi, E. and Masetti, M., (2009) "*Design of a Very High Speed Fuzzy Processor by VHDL Language*", Physics Department University of Bologna, URL:http://www.bo.infn.it/dacel/papers/96_03_parigi_EDTC.pdf. Accessed on 4 October 2009.

Jantzen, J., (1998) "*Design of Fuzzy Controllers*". Technical University of Denmark, and Department of Automation, reports no 98-H 869 (soc), 19 Aug.

Ibrahim, A.M., (2004) "*Fuzzy Logic for Embedded Systems Applications*", USA, Elsevier Science.

Obaid, Z.A., Sulaiman, N. and Hamidon, M.N., (2009) "*Developed Method of FPGA-based Fuzzy Logic Controller Design with the Aid of Conventional PID Algorithm*", Australian Journal of Basics and Applied Science, Vol. 3(3), P: 2724-2740.

Ying, H., (2000) "*Fuzzy Control and Modeling, Analytical Foundations and Applications*", Institute of Electrical and Electronic Engineers Inc., USA.

Wang, J., Zhang, Y. and Wang, W., (2006) "*Optimal design of PI/PD controller for non-minimum phase system*", Transactions of the Institute of Measurement and Control, Vol. 28 No. 1, p: 27-35.

Allaoua, B., Gasbaoui, B. and Mebarki, B., (2009) "Setting Up PID DC Motor Speed Control Alteration Parameters Using Particle Swarm Optimization Strategy", Leonardo Electronic Journal of Practices and Technologies, Issue 14, p: 19-32, January-June 2009.

Barriga, A., Sanchez-Solano, S., Brox, P., Cabrera, A., and Baturone, I., (2006) "*Modeling and implementation of Fuzzy Systems Based on VHDL*", International Journal of Approximate Reasoning, Elsevier Inc, Vol. 41, p: 164–178.

Huang, S.H. and Lai, J.Y., (2005) "*A High Speed Fuzzy Inference Processor with Dynamic Analysis and Scheduling Capabilities*", IEICE Transaction Information & System., Vol. E88-D, No.10 October.

Alvarez, J., Lago, A. and Nogueiras, A., (2006) "*FPGA Implementation of a Fuzzy Controller for Automobile DC-DC Converters*" Proceeding of IEEE International Conference on Field Programmable Technology, December, page(s): 237-240.

Lund, T., Aguirre, M. and Torralba, A., (2004) "Fuzzy Logic Control via an FPGA: A Design using techniques from Digital Signal Processing", Proceedings of IEEE International Symposium on Industrial Electronics, vol. 1, May 4-7, page(s): 555- 559.

Sultan, KH., "*Inverted Pendulum, Analysis, Design and Implementation*" IIEE Visionaries Document Version 1.0, Institute of Industrial Electronics Engineering, Karachi, Pakistan. [Available online at] Matlab Central File Exchange http://www.Mathworks.com.

Xilinx Company. 2009. Virtex 2.5 V Field Programmable Gate Arrays, Data Sheet DS003, URL www.xilinx.com. Accessed on 2009.

Hu, B.G., Mann, G.K. and Gosine, R.G, (1999) "*New Methodology for Analytical and Optimal Design of Fuzzy PID Controllers*", IEEE Transactions on Fuzzy Systems, Vol. 7, No. 5, pp. 521-539, October.

Passino, K.M., and Yurkovich, S., (1998) "*Fuzzy Control*, Addison-Weslwey Longman Inc., USA.

Masmoudi, N., Hachicha, M. and Kamoun, L., (1999) "*Hardware Design of Programmable Fuzzy Controller on FPGA*", Proceedings of IEEE International Fuzzy Systems Conference Proceedings, August 22-25, page(s):1675-1679.

Jain, T. and Nigam, M.J., (2008) "*Optimization of PD-PI Controller Using Swarm Intelligence*", Journal of Theoretical and Applied Information Technology, Page: 1013-1018, 2008.

Obaid, Z.A., Sulaiman, N., Marhaban, M.H. and Hamidon, M.N. (2010), "Implementation of Multistructure PID-like Fuzzy Logic Controller using Field Programmable Gate Array" IEICE Electronics Express, Vol. 7 No. 3, P: 132-137, 10 February.

Li J. and Hu B.S., (1996) "The Architecture of Fuzzy PID Gain Conditioner and its FPGA Prototype Implementation", Proceedings of the Second International Conference on ASIC, October 21-24, page(s):61-65.

Khatr, A. P., and Rattan, K.S., (2006) "Implementation of a Multi-Layered Fuzzy Controller on an FPGA", Annual meeting of the North America, by IEEE Fuzzy Information Processing Society, Page(s): 420- 425.

J.L. Gonzalez-vazquez, O. Castillo and L.T. Aguilar-bustos, "A Generic Approach to Fuzzy Logic Controller Synthesis on FPGA", Proceedings of IEEE international conference on fuzzy systems, 2006, page(s): 2317 – 2322.

Gonzalez-Vazquez, J.L., Castillo, O. and Aguilar-bustos, L.T., (2006) "A Generic Approach to Fuzzy Logic Controller Synthesis on FPGA", Proceedings of IEEE international conference on fuzzy systems, page(s): 2317 – 2322.

MATLAB - A Ubiquitous Tool for the Practical Engineer

Edited by Prof. Clara Ionescu

ISBN 978-953-307-907-3
Hard cover, 564 pages
Publisher InTech
Published online 13, October, 2011
Published in print edition October, 2011

A well-known statement says that the PID controller is the â€œbread and butterâ€ of the control engineer. This is indeed true, from a scientific standpoint. However, nowadays, in the era of computer science, when the paper and pencil have been replaced by the keyboard and the display of computers, one may equally say that MATLAB is the â€œbreadâ€ in the above statement. MATLAB has became a de facto tool for the modern system engineer. This book is written for both engineering students, as well as for practicing engineers. The wide range of applications in which MATLAB is the working framework, shows that it is a powerful, comprehensive and easy-to-use environment for performing technical computations. The book includes various excellent applications in which MATLAB is employed: from pure algebraic computations to data acquisition in real-life experiments, from control strategies to image processing algorithms, from graphical user interface design for educational purposes to Simulink embedded systems.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen