Journal of Engineering and Applied Sciences 7 (2): 184-190, 2012
ISSN: 1816-949X

Design and FPGA Implementation of a Control Processor

Thakwan M. Saleem and Zeina Ali
Department of Computer, College of Engineering, University of Mosul, Iraq

Abstract: The speed and programmability are the main characteristics to determine the performance of a microprocessor. A processor has to fit some important characteristics like flexibility, high speed and low cost. The aim of this research is to design a control processor capable of implementing various control strategies like Fuzzy control PID control and non-linear control. By achieving this, the processor will provide a very simple and efficient method in designing control systems. The design of the processor was simulated using simulation program included in the ISE9.2. The processor was realized and implemented on the FPGA chip in a panel Spartan 3. The processor was tested to control a dc servo motor. The control methods mentioned were used simultaneously and the designed processor showed its efficiency.

Key words: FPGA, CPU, VHDL, RISC, processor, Iraq

INTRODUCTION

The Central Processing Unit (CPU) is the heart of the automatic computer and it contains a group of the logical circuits. It executes a group of the instructions and specific arithmetic operations. Then storing the execution results in a specific memory or transmitting them to other computer parts.

Many techniques have been developed and extensively used to improve or to replace conventional control techniques, since these techniques do not require a precise model.

Fuzzy logic is one of the intelligent techniques developed by Lotfi A. Zadeh as a new methodology for dealing with uncertainty; it is applied for controller design in many applications (Abdulhameed, 2010). Fuzzy logic has the advantage of modelling complex and non-linear problems linguistically rather than mathematically.

The mathematical model is too complicated to be of practical use and therefore a linguistic model becomes advantageous. The use of fuzzy logic requires, however the knowledge of a human expert to create an algorithm that agrees with human expertise and thinking (Abdulhameed, 2010).

Most of the industrial control requirements are still met by Proportional-Integral-Derivative (PID) type controllers. The PID Control will show the characteristics of the (P, I and D) controls and how to use them to obtain a desired response. It will consider the following unity feedback system as shown in Fig. 1. The digital version of PID algorithm is:

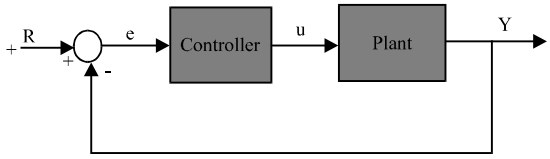

Fig. 1: Feedback control system

$$u(k) = Kp \times e(k) + Ki \times \sum^{k} e(k) + Kd \times (e(k) + e(k-1)) \quad (1)$$

$$Ki = \frac{Kp \times \Delta t}{Ti}, \qquad Kd = \frac{Kp \times Td}{\Delta t}$$

Where:

Δt = The sampling period
Kp = The proportional constant
Ti = The integral time constant
Td = The derivative time constant

Many different types of non-linearities may be found in physical control systems and they may be divided into two classes depending on whether they are inherent in the system or intentionally inserted into the systems. Inherent non-linearities are unavoidable in control systems (Burns, 2001).

The processor layout: The control processor presented in this study consists of three components

Corresponding Author: Thakwan M. Saleem, Department of Computer, College of Engineering, University of Mosul, Iraq

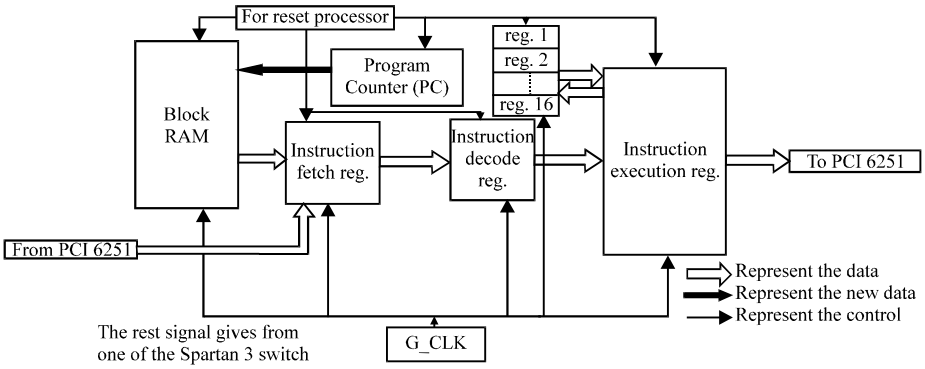

Fig. 2: Overall schematic of the designing processor

as shown in Fig. 2 which describes the main parts. The data sent from the data acquisition card PCI 6251 is inputted to the processor to calculate the controlling signal then sent to PCI 6251 and this operation is repeated.

The acquisition card is used to provide the interfacing between the processor and the plant to be controlled.

The main memory: The Spartan 3 board contains many single and dual memories. The main memory used in this work is single port.

Its port used is for read instructions through Program Counter register (PC), this register gives the address of the instruction stored in the specified memory location.

The fetch and decode registers are designed with 16 bit width. Therefore, the data length in main memory is 16 bit. Figure 3 shows connecting some processor parts with main memory.

The program counter: The address of the instruction stored in main memory is applied during the rising edge of the clock which is transferred from main memory to the bus connected with the instruction fetch register. Then the program counter is increased at each rising edge to fetch the next data from main memory.

Instruction fetch register: The fetch instruction register is triggered by a clock to read the instruction on the bus that connects it and the main memory. This register is built with 16 bit width.

Instruction decode register: The decode instruction register is triggered by a clock to read the instruction on

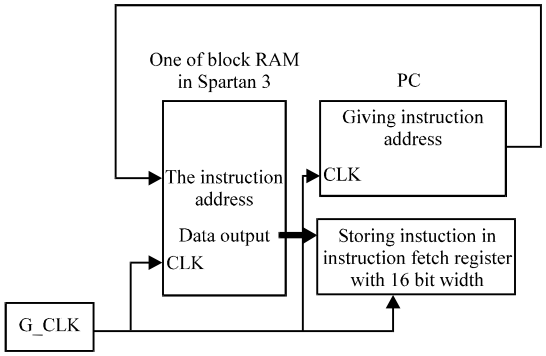

Fig. 3: Connecting some processor parts with main memory

the bus that connects it and the instruction fetch register. This register is built with 16 bit width. There are two methods for execution.

The first needs a single instruction stored in main memory or data stored in any of the processor registers such as In, Out, Not, shift instruction. The second needs dual data for execution the specific operation such as arithmetic operations.

Instruction format: The instruction bits can be specified according to design requirements. Some consist of 16 bit and another consist 32 bit (Mano, 1993).

The instruction obtained from the memory includes 16 bit. Figure 4 and Table 1 shows the important instruction format and instruction set of the processor.

PID control: The error value in LabView program ranges between [-1, +1] and to make it acceptable by FPGA kit it must be transformed to the range between [0, 255].

Table 1: Instruction set for control processor

Instruction	Meaning
Cont1	PID control
Cont1_kp_Ti_Td	Selection PID parameter
Cont2	Fuzzy control
Cont2_ge_gce_gu	Selection fuzzy parameter
Cont3	Nonlinear control
Cont3_1	On_Off non-linear control type
Cont3_2	Dead_Zone non-linear control type
Cont3_3	Saturation non-linear control type
PROP	Calculates proportion term
INTEG	Calculates integral term
DIFFER	Calculates differentiation term
JMP	Unconditional jump
MUL	Multiplication
SUB	Subtraction
SUM	Summation
ADD	Addition two values
DIV	Division
OUT	Output data
IN	Input data
CE	Calculates change of error value
QU	Calculates quantization parameter
DEG	Calculates degree values
MF	Calculates membership values
MIN	Calculates minimum values
CEN	Calculates center values
AND, OR, XOR, XNOR, NOT	Logical operations

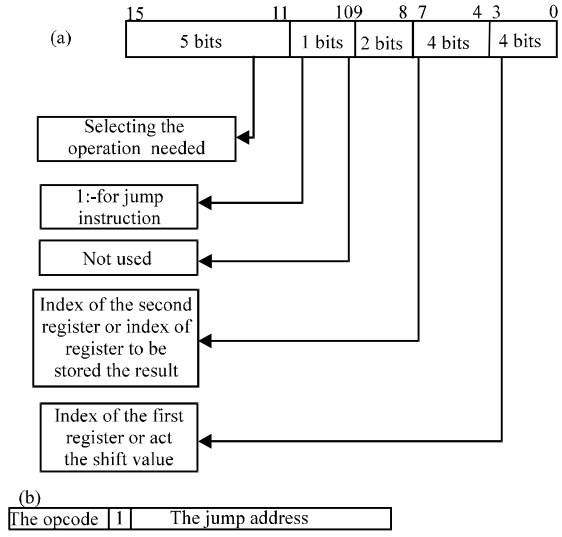

Fig. 4: General instruction formula; a) Instruction formula; b) Jump instruction formula

After obtaining the error value (0-255), this value includes the positive and the negative values of the error. When Er (7) used to signed and other bits used for value. The PID parameter is taken for Kp = 2, Ti = 0.03 and Td = 0.001.

Table 2 shows the results of calculates these terms when Mfs bit (15) of each term used for indicated the signed number and the timing diagram of the summation for these terms using ISE 9.2 simulator can be shown in Fig. 5.

Table 2: PID terms

Error in LabView	Error in VHDL code	Proportion term	Integration term	Differentiation term	PID controlling signal
0.99	7E	00FC	0054	0	0150
0.98	7C	00F8	00A6	0	019E
0.95	77	00EE	00F5	8001	01E2
0.93	76	00EC	0143	0	022F
0.92	75	00EA	0191	0	027B

Table 3: Selection fuzzy controller type

Switch3	Fuzzy controller type
0	Classical fuzzy controller (BFC)
1	Adaptive fuzzy controller (SFC with BFC)

Fuzzy control: The error value in LabView program ranges between [-1,+1] and to make it acceptable by FPGA kit it must be transformed to the range between [0, 252]. After inserted error signal from DAQ to FPGA done finding change of error value the multiplying them with quantization parameters (Ge and Gce), respectively then inserting these values to fuzzy controller parts, finally the controller signal multiplied with scaling parameter. The type of fuzzy controller (BFC or BFC and SFC in both) can be selecting through switch 3 and Table 3.

Finding change of error value: The proposed fuzzy system in FPGA have one input (error) and the designed fuzzy has two inputs (e and ce). Therefore must be found the second value form the error value, the ce value may take many possible values because the error and previous error may take positive or negative values and which of them is larger or smaller.

Figure 6 show the flowchart for finding ce value when the error value is taken 0.4 in LabVIEW program and Ge = Gce = Gu = 1.

Representation of the Fuzzy data in memory: The input value of the error to Spartan 3 kit ranging between 0-255 for the positive and negative values. Each value of error or change of error has two fuzzy values (two degree), each input needs 756 locations (8 bits for each location) this includes 8 bit for error or change of error and 16 bit for storing the two fuzzy value.

This means for two input to 1512 locations are needed. This data is large and needs a large area memory size storing. One of the block RAM in Spartan 3 is used for storing the fuzzy data, the block RAM dual match (1 k×16) is used for this purpose Fig. 7 shows the

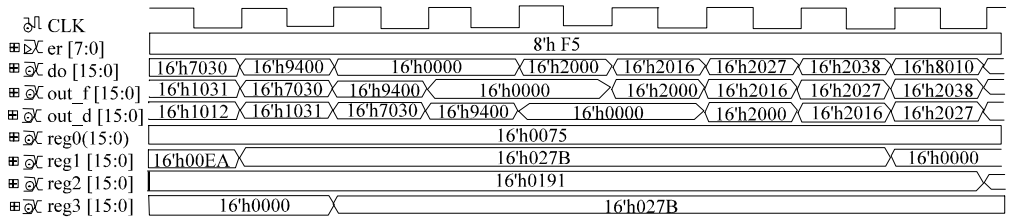

Fig. 5: Timing diagram of the summation terms

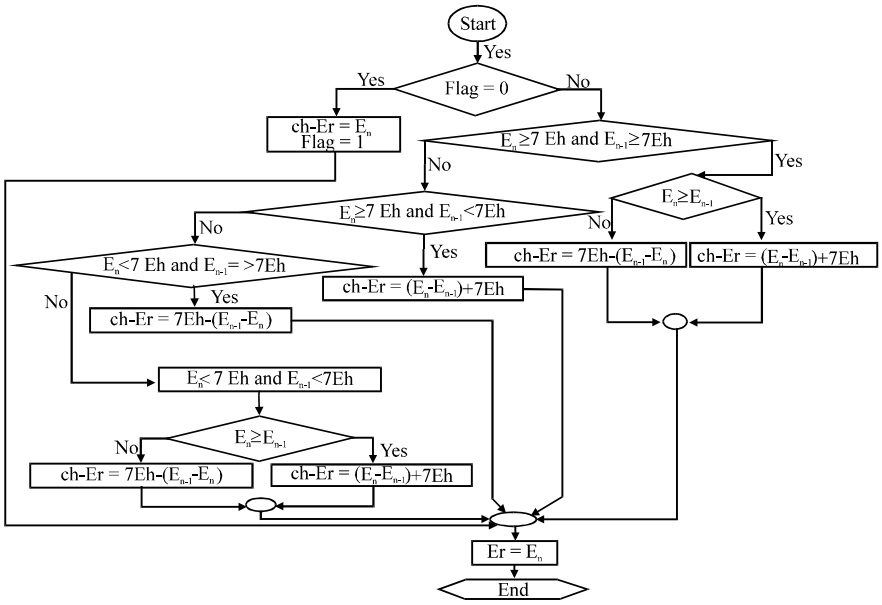

Fig. 6: Flowchart for finding the ce value

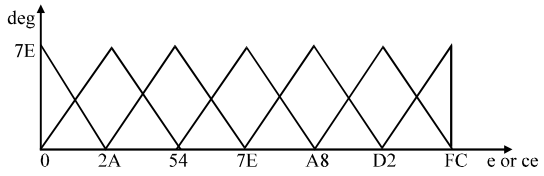

Fig. 7: Representation of the MFs in FPGA

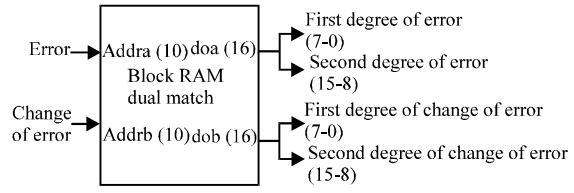

Fig. 8: Organization fuzzy data in block RAM

representation of the MFs in FPGA and Fig. 8 shows the memory organization. The degrees of e and ce can be calculated manually using Eq. 2 where p5 is the slope and P1 is the first point of the MF. Figure 9 shows the timing diagram for finding degrees of the e and ce.

Degree 1 = ((e or ce)-p1)×p5, Degree 2 = not (degree 1) (2)

Finding MFS for each input: Each input of the fuzzy take two MFs because the overlapping between MFs is 50% and the second MF is the same first MF plus one because these MFs are neighboring. Figure 10 shows the flow chart to obtain the MF. Table 4 shows the theoretical result to calculate the degrees and MFs of inputs.

The Min-Max operations: In this instruction the selection two degrees and two MF and finding the minimum between them then saving the result in special register depending on FSM (Finite State Machine) principle is done.

This FSM consists of four states executed in each clock and one by one in each state comparing of two degrees one for e and another for the ce is done and finding the minimum between them then storing the crisp value of the minimum and integrate the MF in one register. Table 5 shows the theoretical result for min-max operation.

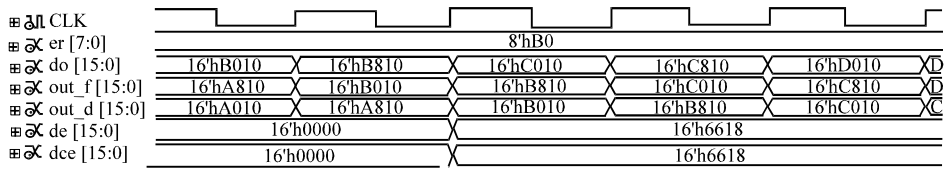

Fig. 9: Timing diagram for degrees values

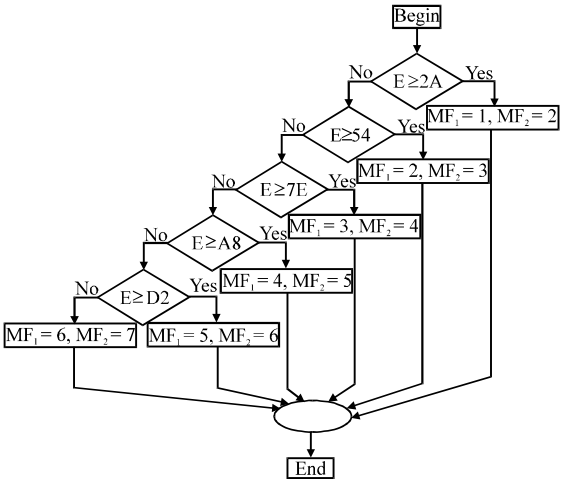

Fig. 10: Flow chart for finding the MFs of e or ce

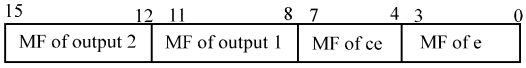

Fig. 11: Representation of the MFs in array

Table 4 theoretical result to calculate the degrees and MFs of inputs

Result (H)	e (B0)	ce (B0)
de 1	(B0-A8)* 3 = 18	(B0-A8)* 3 = 18
de 2	7E-18 = 66	7E-18 = 66
mr 1	5	5
mce 2	6	6

Table 5: Theoretical result to calculate the FSM result

	State 1			State 2			State 3			State 4		
Result	d	c	MF	d	c	MF	d	c	MF	d	c	MF
	66	B0	5	66	B0	5	18	B0	6	18	B0	6
	66	B0	5	18	B0	6	66	B0	5	18	B0	6
Final	66	B0	55	18	B0	65	18	B0	56	18	B0	66

Representation of fuzzy rules: The two inputs (e and ce) and each input is uses seven MFs, therefore 49 fuzzy rules are used to control the limits the output states according to inputs states.

An array is used to represent the fuzzy rules, this array is built with 49 elements (0-48) and each element have 16 bits width, the first 4 bits are used to represent the MF of e and the second represent the MF of the ce and third 4 bits represent the MF of the BFLC (output 1) and the last 4 bits act as the MF of the SFCL (output 2). Figure 11 shows the MF of two inputs and two outputs. Figure 12 shows the timing diagram for finding the MFs of the outputs. There are four MFs of the output.

Each final degree cuts MF of the fuzzy output in two point (center 1 and 2). Equation 3 used for obtaining the center values:

$$Cen11 = (min.\ value/p5)+p1 \quad (3)$$

The final result can be obtained using Eq. 4 this equation shows the need for eight multipliers to calculate the output. The divider used for dividing num (18 bit) on den (10 bit) for obtaining the final result and Fig. 13 shows the timing diagram for dividing operation:

$$\text{Final output} = \frac{\sum \min_i c_i}{\sum \min_i} \quad (4)$$

Adaptive fuzzy controller: The self tuning adaptive fuzzy controller was implemented to adapt the work of the BFC to respond with changes in the control loop (Abdalrazaaq, 2008).

The SFC changes the parameters of BFC in real time depending on error and change of error values. Figure 14 shows the general schematic of STFLC (Self Tuning Fuzzy Logic Control) executed in this thesis:

$$U(k) = u(k).\alpha.Gu \quad (5)$$

Nonlinear control: The type of non-linear control (On-Off, Dead-zone and Saturation) and constraints can be inserted through Spartan 3 switches and push button and result is displayed on LEDs.

Flowchart in Fig. 15 shows the designed steps of the non-linear control. Saturation non-linear type is used with switching point = 10_h, slope = 2 and assuming error value = 7_h. Figure 16 show the timing diagrams of non-linear output.

Resource estimation: When implementing the PID control processor on FPGA Spartan 3 kit, the percentage occupied area is 29%. When implementing the BFC only on FPGA Spartan 3 kit, the percentage occupied area is

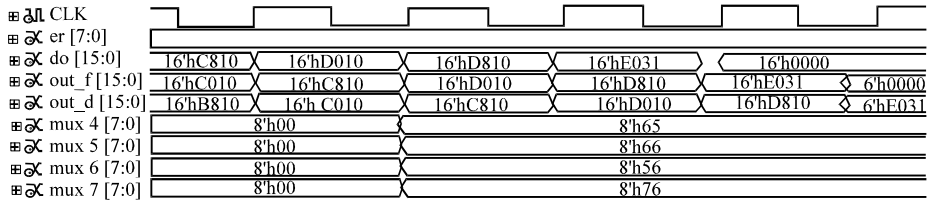

Fig. 12: Timing diagram of the MFs outputs

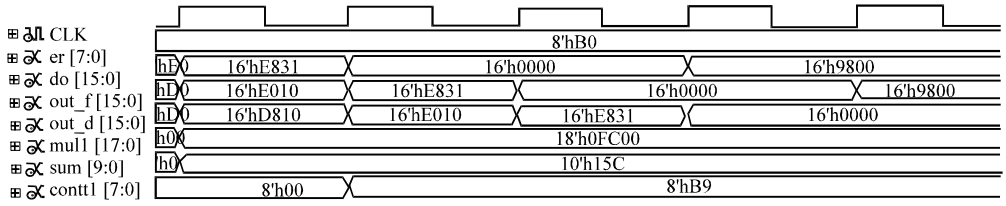

Fig. 13: Tiiming diagram of the crisp fuzzy output

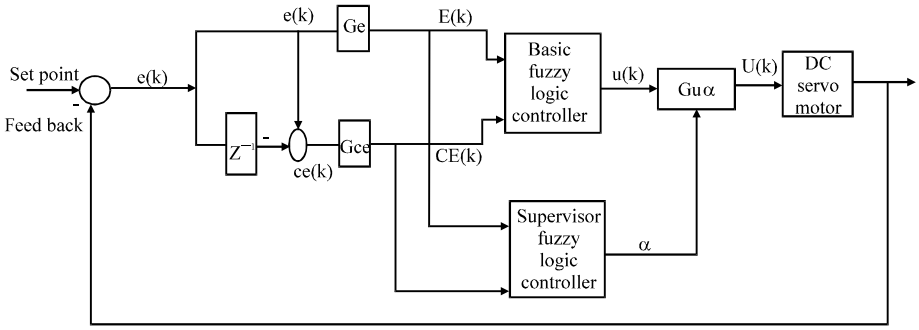

Fig. 14: General schematic of STFLC

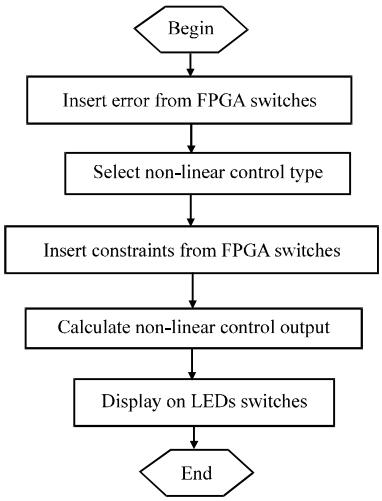

Fig. 15: Flow chart of non-linear control

39% and when adding SFC of the designed processor the percentage of the occupied area became 63%. When implementing the non-linear control on FPGA Spartan 3 kit, the percentage occupied area is 11%. Table 6 shows the hardware resources for overall controller processor (PID, fuzzy controller (BFC+SFC) and non-linear).

Hardware performance evaluation: To evaluate the performance of implementing the control method in hardware over software the speed up ratio is used. The speed up of a hardware processing time over an equivalent software processing time is defined by the ratio (Zaki, 2009):

$$\text{Speed up} = \frac{\text{Software time}}{\text{Hardware time}}$$

So, the speeds for design are compared with that when softwarely (Lab View) running on computer (Pentium 4, 2.00 GHz, 1 GB RAM), the overall time required for implementing control method on Xilinx Spartan 3 FPGA (XC3S200) 50 MHz is equal to (16.3227 μ sec) while the same model needs (31250 μ sec) when implemented softwarely, producing a speedup of 1914.51 and the operating frequency is 45.314 MHz.

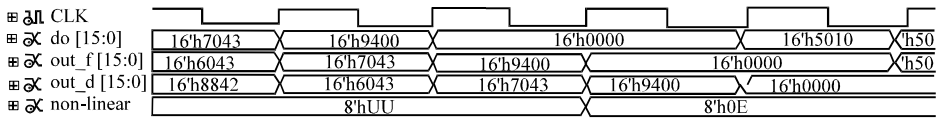

Fig. 16: Timing diagram of the saturation non-linear output

Table 6: Hardware resources required for overall control processor

	Device utilization summary		
Logic utilization	Used	Available	Utilization (%)
Total no. of slice registers	656	3,840	17
No. of flip flops	624	-	-
No. used as latches	32	-	-
No. of 4 input LUTs	3,392	3,840	88
Logic distribution			
No. of occupied slices	1,887	1,920	98
No.of slices containing only related logic	1,887	1,887	100
No. of slices containing only unrelated logic	0	1,887	0
Total no. of 4 input LUTs	3,451	3,840	89
No. used as logic	3,392	-	-
No. used as route-thru	43	-	-
No. used as shift registers	16	-	-
No.of bonded IQBs	45	173	26
IQB flip flops	16	-	-
No. of block RAMs	2	12	16
No. of MULT18×18s	12	12	100
No. of GCLKs	1	8	12

CONCLUSION

The proposed controller methods used one input (error) instead of using two inputs (error and change of error) this method reduce the hardware resources. The BFLC utilizes roundly 40% of the area while the STFLC controller utilizes roundly 65% of the FPGA area. Choosing the controller methods are done on line on the designed processor.

The implementation of the control system on hardware gives a high speed up over the software calculation time of the system. The obtained results showed a very good with small error responses compared to the practical results of the control system. The designed processor combined three different controller methods (PID, fuzzy and non-linear control elements) in single chip and selection between them can be done using switches. Spartan 3 FPGA is suitable for implementing the control processor because the best utilization of the resources. The goal of the SFCL is to adjust the output scaling value of the basic fuzzy in real time to obtain the required response in each state also to reduce the effort of finding this value in trial and error method. Future research can be added by designing Fuzzy-neural processors through generation MF and fuzzy values depending on neural network manner and use fuzzy control to analyze the fuzzy rules and finding the final result, design a controller processor using VHDL language of optimal control type and other types specialized for control using the implemented non-linear elements. Design Ethernet-bus and adding it to the processor on the same chip to obtains complete system on chip. The design of an optimal method for auto tuning of the PID parameters.

REFERENCES

Abdalrazaaq, S.N., 2008. Design and implementation of a fuzzy controller with self tuning factors using labview. M.Sc. Thesis, Mosul University.

Abdulhameed, M.A., 2010. Design and implementation of a fuzzy logic based photovoltaic peak power tracking controller. M.Sc. Thesis, Mosul University.

Burns, R.S., 2001. Advanced Control Engineering. University of Plymouth, UK.

Mano, M.M., 1993. Computer System Architecture. 3rd Edn., Prentice Hall, USA., ISBN: 0-13-175738-5, Pages: 525.

Zaki, K., 2009. Implementation of neuro-crypto system using FPGA. Master's Thesis, Technical Computer Engineering, Technical College of Mosul.