International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 01, January-2023

Design of A Soft-Core Processor in FPGA

Bijoy Kumar Upadhyaya
Department of Electronics & Communication Engineering,
Tripura Institute of Technology, Narsingarh,
Tripura (W), Pin 799015

***Abstract*:-Soft-core processors are a form of microprocessors whose architecture and working can be fully described using a hardware description language such as VHDL or Verilog. These processors are customizable for a given application and can be synthesized for implementation in reconfigurable hardware platforms like Field Programmable Gate Array. On successful implementation of the said design, Application Specific Integrated Circuit (ASIC) can be developed for mass production.**

In this paper, we have designed, simulated and tested an 8-bit VHDL based soft-core processor. The proposed processor is targeted to be used for small to medium complexity control applications. The designed 8-bit soft processor can be used to develop ASIC. The simulation result obtained using ModelSim software is presented in the form of the timing diagram. Implementation results of the design on the Xilinx Spartan-3 FPGA kit in the laboratory are also presented.

Keywords: Soft Processor, IP, VHDL, FPGA.

1. INTRODUCTION:

Soft-core processors are treated as an Intellectual Property (IP) core and can be implemented on Field Programmable Gate Array (FPGA) platform [1]. These processors offer key advantages like configurability to trade between price and performance, faster time to market, easy integration with the FPGA fabric, and avoiding obsolescence. Present day FPGA embeds hard-core and soft-core processors within the chip, facilitating a great amount of configurability for the application developers [2]. Soft-core processor design on FPGA platform using dataflow based design approach for image processing applications has been reported by Amiri *et al.* [3]. A survey on soft-core processor design for embedded applications was carried out by Tong et al. [4]. This work presented an overview and comparison between various solutions available from commercial vendors and open source communities. Step by step implementation experience of a soft-core processor using Xilinx ISE is reported in [5].

In this paper, the design of an 8-bit soft-core processor for control applications with simple and moderate complexity has been presented. The processor is completely coded in VHDL using Xilinx Integrated Software Environment (ISE) tool and simulated using ModelSim software. Simulation results obtained are presented and analyzed. Digital logic requirement and resource utilization summary of the target FPGA IC has been presented which endorses the efficient implementation of the proposed soft-core processor for control applications.

The organization of the rest of the paper as follows: Section 2 presents the system consideration used in this design process. Brief description about register array including its simulation is presented in Section 3. Section 4 provides detailed view of ALU functionality and its modelsim implementation. Comparator and Shifter design approaches with their timing simulations have been presented in Section 5 and 6 respectively. Section 7 illustrates the control unit design process with simulation diagram to endorse it functionality. Section 8 presents the FPGA implementation results and Section 9 provides the concluding remarks.

2. SYSTEM CONSIDERATION:

Fig. 1 presents the hierarchy of the designed microprocessor. The top level design includes two components: external memory and CPU. External memory is required to hold program and data. Any off-the-shelf memory chip whose speed of operation matches with the designed FPGA based CPU can be used. Register array, ALU, comparator, shifter and control unit are the internal components of the CPU [6]. The control unit is further subdivided into three components namely Instruction Register (IR), controller and Program Counter (PC).

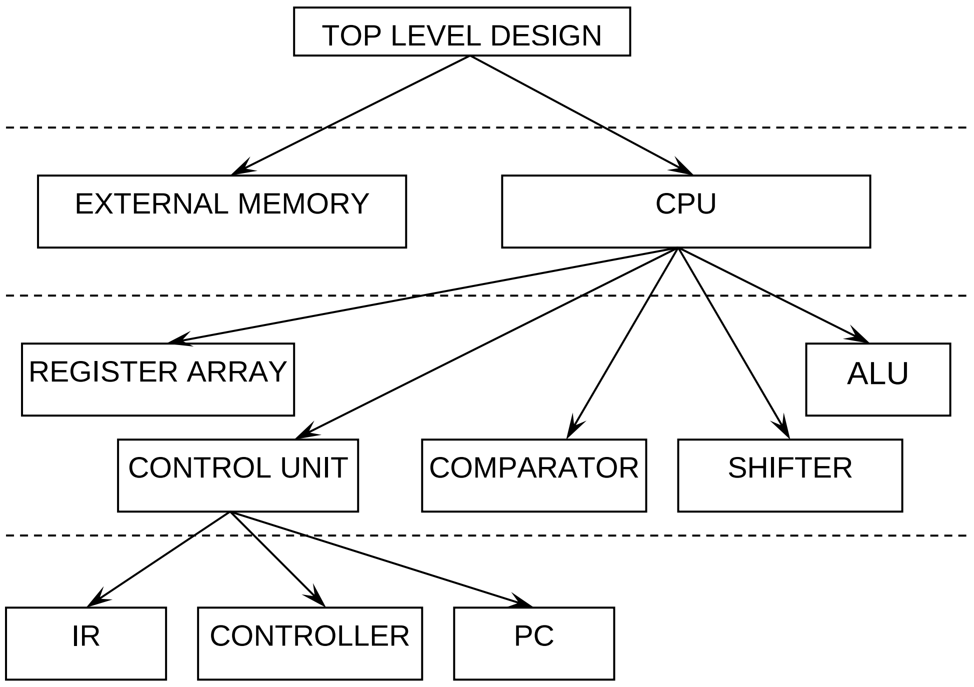

Fig. 1: Microprocessor design hierarchy

3. REGISTER ARRAY:

The register array in the design contains eight numbers of 8-bit registers, R(0) through R(7). These are used as temporary storage during the program execution. The VHDL model of the register array is described in the form of block diagram as shown in Fig. 2. One out of the eight registers is selected for read or write operation by a 3 : 8 decoder [7]. Logic values at SEL dictates which register is to be operated on. The selected register's content will be loaded into the data bus if READ = 1. Similarly, the content of data bus is written into the selected register if WRITE = 1.

The simulation result of register array in the form of timing diagram obtained using ModelSim XE-III software is shown in Fig. 3. In the first clk period register R(0) is selected for write as well as read operation (SEL = 000, WRITE = 1 and READ = 1). As seen in Fig. 3, input data, DATA_IN = 11111111 is written into R(0) and the same data is read. During second clk period, R(1) is selected for only read operation. As no data is written into R(1), undefined value is read. In the third clk period only write operation is permitted in R(0). But as READ = 0, output is at high impedance state (indicated by DATA_OUT = ZZZZZZZZ). Data stored in R(0) (= 11110000) during third clk period is read in 9th clock period which indicates that without assigning WRITE control signal it is not possible to change the stored data in a particular register.

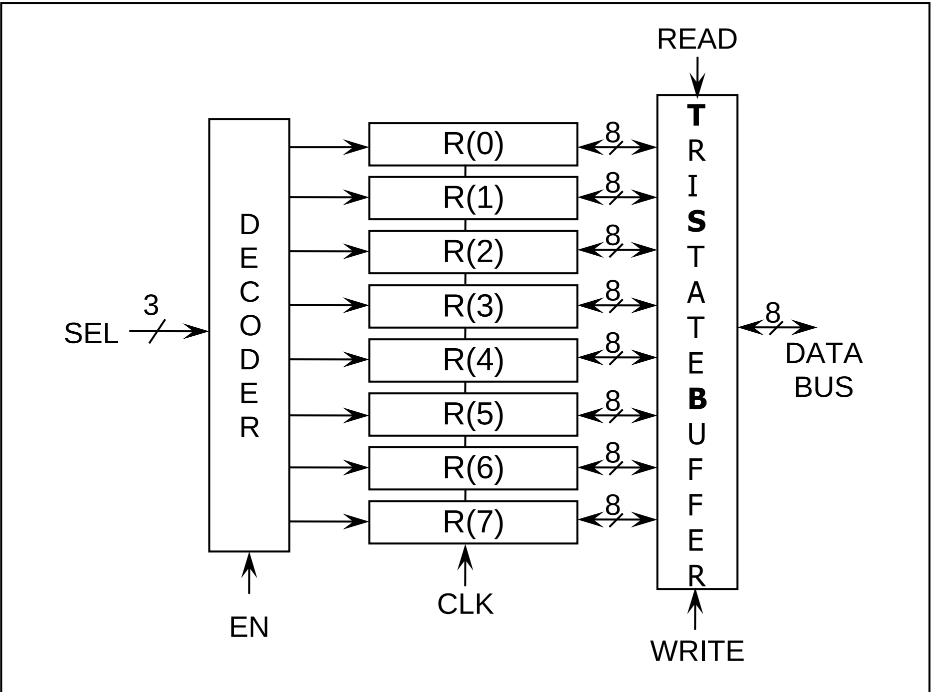

Fig. 2: VHDL model of Register Array

3.1 Simulation result of register array:

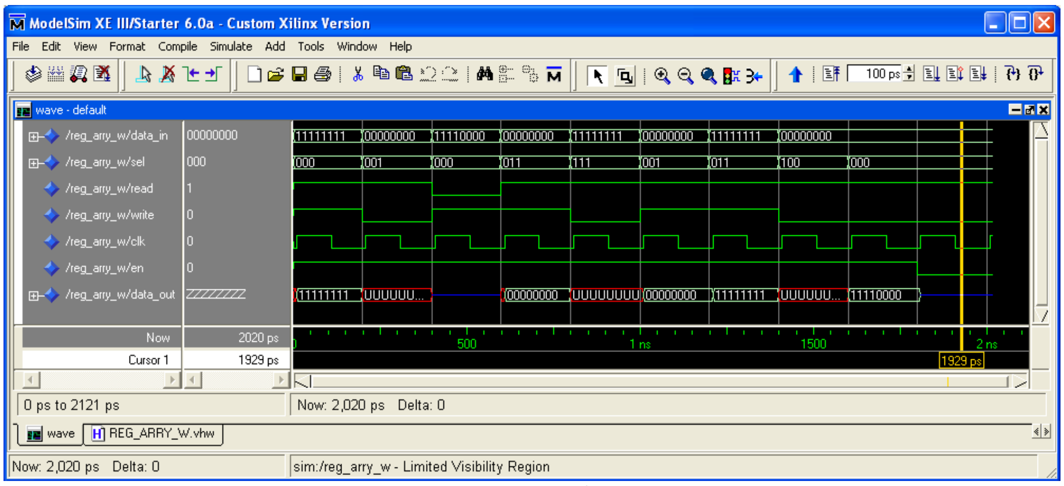

Fig. 3: Simulation result of register array

4. ARITHMETIC LOGIC UNIT (ALU):

The ALU of the soft processor is having the following operations:

1) Addition	2) Subtraction	3) Increment	4) Decrement
5) AND	6) OR	7) XOR	8) NOT

The first four operations of the ALU are arithmetic and rests four are logical. As 8 numbers of operations are chosen, three bits are assigned to signify them. Table 1 describes the ALU operations and their corresponding bit assignments:

Table 1: ALU operations and their bit assignments

Sl. No	ALU operations	Assignments		
		SEL(2)	SEL(1)	SEL(0)
1	Addition	0	0	0
2	Subtraction	0	0	1

3	Increment	0	1	0
4	Decrement	0	1	1
5	AND	1	0	0
6	OR	1	0	1
7	XOR	1	1	0
8	NOT	1	1	1

The ALU contains two 8-bit registers, A and B at its input. For the operations which require two operands, both registers are used whereas for single operand operations register A is only used. The result of an operation is transferred to another 8 bit register, C. The ALU also include three flags, Carry (CF), Zero (ZF) and Sign (SF) to reflect data condition of an ALU operation. The ALU remains in high impedance state when it is not selected for operation.

4.1 Simulation result of ALU: The simulation result of ALU in the form of timing diagram is shown in Fig. 4. All the 8 operations of the ALU are shown with typical data input along with corresponding output. Table 2 explains the simulation result.

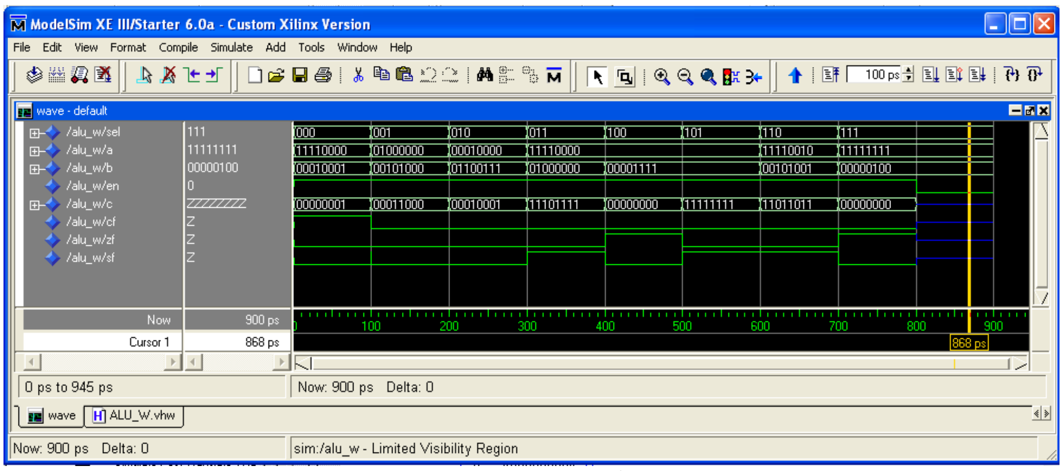

Fig. 4: Simulation result of ALU

Table 2: Interpretation of ALU simulation result

Sl. No	ALU operations	EN	SEL	Input		Output			
				A (Hex)	B (Hex)	C (Hex)	CF (2)	ZF (2)	SF (2)
1	Addition	1	000	F0	11	01	1	0	0
2	Subtraction	1	001	40	28	18	0	0	0
3	Increment	1	010	10	67	11	0	0	0
4	Decrement	1	011	F0	40	EF	0	0	1
5	AND	1	100	F0	0F	00	0	1	0
6	OR	1	101	F0	0F	FF	0	0	1
7	XOR	1	110	F2	29	DB	0	0	1
8	NOT	1	111	FF	04	00	0	1	0
9	---	0	x	xx	xx	xx	x	x	x

5. COMPARATOR:

The comparator circuit of the CPU compares the two given numbers, A and B. The result of comparison is reflected by logic state of CF and ZF as shown in Table 3. The source contents are not modified by a compare operation. The outputs remain in hi-impedance state when comparator is not selected.

Table 3: Flag status after a compare operation

Sl. No.	Operation	Flag Status	
		CF	ZF
1	A > B	0	0
2	A < B	1	0
3	A = B	0	1

5.1 Simulation result of Comparator: The simulation result of the comparator is presented in Fig. 5. The enable input (EN) is high for first three operation and low in the fourth. This causes the outputs (CF and ZF) to remain in high impedance state. We see CF = ZF = 0 when A = 10_H and B = 00_H. When A = B = FF_H, we find CF = 0 and ZF = 1. CF = 1 and ZF = 0, when A = 20_H and B = $2F_H$.

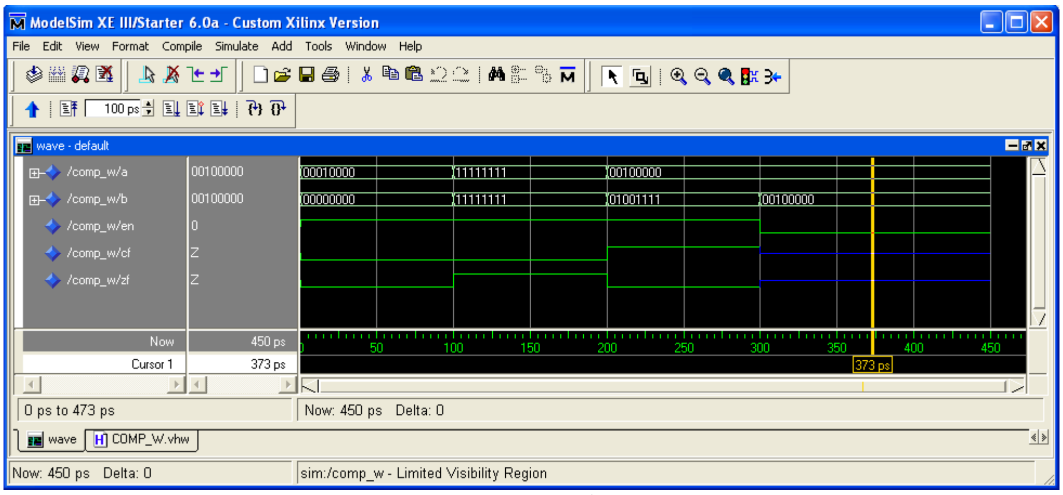

Fig. 5: Simulation result of Comparator

6. SHIFTER:

The shifter circuit in the CPU is responsible for shifting / rotating the contents of register A by one bit position towards left or right. The output is delivered to register C (8 bit). The shifter is assigned with four operations as listed in Table 4. Table 4 also explains the action of the shifter corresponding to each shift operation.

Table 4: Shifter operations and their bit assignments

Sl. No	Shifter operations	Assignments		Action of Shifter
		SEL(1)	SEL(0)	
1	Shift left	0	0	A(7:1) ← A'(6:0) ; A(0) ← 0 [#]
2	Shift right	0	1	A'(7:1)→A(6:0) ; 0 → A(7)
3	Rotate left	1	0	A(7:1) ← A'(6:0) ; A(0) ← A'(7) ; CF←A'(7)
4	Rotate right	1	1	A'(7:1)→A(6:0) ; A'(0) → A(7) ; A'(0) → CF

[#] A' and A represent contents of register A before and after operation is performed respectively.

6.1 Simulation result of shifter: Table 5 interprets the simulation result of the shifter shown Fig. 6. All the four operations are tested with some typical data and their corresponding output is also presented. The shifter goes to hi-impedance state when it is not selected (EN = 0).

Table 5: Interpretation of Shifter simulation result

Sl. No	Shifter operations	SEL	Input	Output	
			A	C	CF
1	Shift left	00	10101010	01010100	-
2	Shift right	01	11110000	01111000	-
3	Rotate left	10	01010101	10101010	0
4	Rotate right	11	01111111	10111111	1

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 01, January-2023

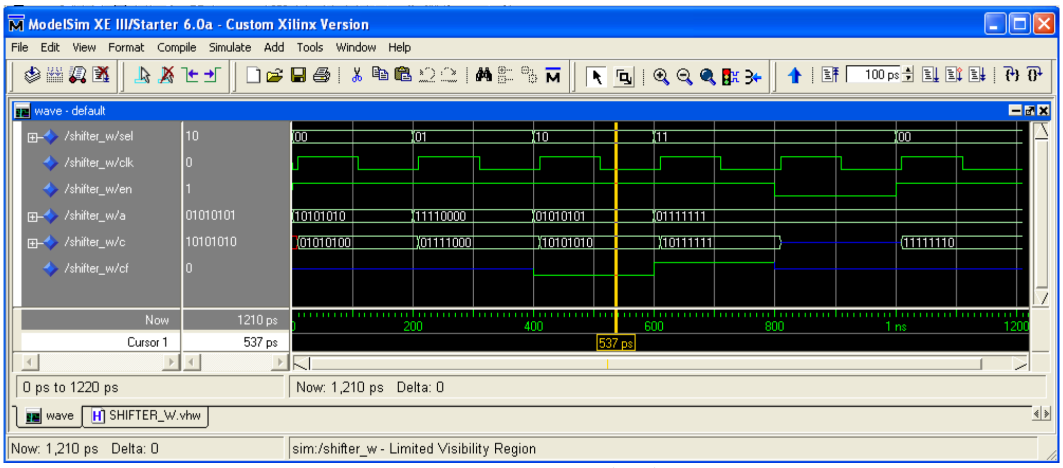

Fig. 6: Simulation result of Shifter

7. CONTROL UNIT (CU):

This is the most important block of the CPU. It is responsible for providing necessary timing and control signals to all the operations of the CPU [8]. The timing and control signals are to be issued in proper sequence without which the required operation will not be accomplished. As shown in Fig. 1, the CU is further divided into three circuits called IR, controller and PC.

7.1 Instruction register (IR): IR in a CPU holds the instruction code to be decoded. The CPU reads an instruction code from external memory using PC and loads it into the IR. It is an 8-bit special purpose register. D_7 and D_6 of IR indicate that out of four circuits (register array, ALU, comparator or shifter) where the current instruction is to be executed. If D_7D_6 = 00, register array is selected, if 01 ALU is selected, with 10 comparator is selected whereas for 11 shifter is selected. Table 6 below describes the interpretation of the different instruction code.

Table 6: Interpretation of IR contents

Sl. No.	D_7D_6 of IR	Selected CPU block	Interpretation of other bits in IR	
1	00	Register array	D_5-D_3 : destination register	D_2-D_0 : source register
2	01	ALU	D_5-D_3 : ALU operation	---
3	10	Comparator	---	
4	11	Shifter	D_5-D_4 : Shifter operation	---

7.2 Program counter (PC): It is a 16 bit special purpose register used to hold the memory address of the next instruction to be executed. The PC is incremented during the execution of an instruction so that it points to the address of the next instruction in the program.

7.3 Controller: It is the most important block of CU and provides the necessary timing and control signals for the different circuits of CPU. It causes the contents of PC to be placed in Address bus. It places operation codes from data bus into the IR. The controller activates the enable input of one out of four circuits based on the value of D_7D_6 of IR as narrated in Table 6. A 4 line to 1 line MUX is used to select the outputs of the four circuits and send them to reg. C as output.

7.4 Simulation result of Control unit: The simulation result of the control unit is presented in Fig. 7 below. For clarity of understanding the simulation is performed with typical data, A = 10011001 and B = 01000010. As seen in the timing diagram, the first instruction code loaded into IR is 00000001 (INS_CODE). As D_7D_6 = 00, this is an instruction related to register array where the source register is $D_2D_1D_0$ = 001 (= R(1)) and destination register is $D_5D_4D_3$ = 000 (=R(0)). Accordingly contents of R(1) = 10011001 is written into R(0) and displayed it as C1 in the diagram. As only one circuit is enabled at a time, C2 and C3 outputs are in hi-impedance state during the first clock period. The multiplexer action makes C = C1 for this instruction code.

The next INS_CODE loaded into IR is 01000010 where D_7D_6 = 01. So ALU is selected. With $D_5D_4D_3$ = 000, A + B operation is chosen. The result of the addition operation with the preloaded data in A and B is found to be C1 = C = 11011011, CF = 0, ZF = 0 and SF = 1.

The third INS_CODE loaded into the IR is 10000100. With D_7D_6 = 10 the comparator circuit is selected by the CU. With assumed data for A and B we find A > B, and the result is reflected by CF = 0 and ZF = 0.

The last INS_CODE loaded into the IR is 11110111. The CU selects the shifter circuit for operation as D_7D_6 = 11. With D_5D_4 = 11, Rotate right operation is chosen. The results are C3 = C = 11001100 and CF = 1.

PC is one of the outputs of CU. As shown in Fig. 7 it is incrementing itself during the execution of current instruction so that it can bring the next instruction from memory immediately thereafter.

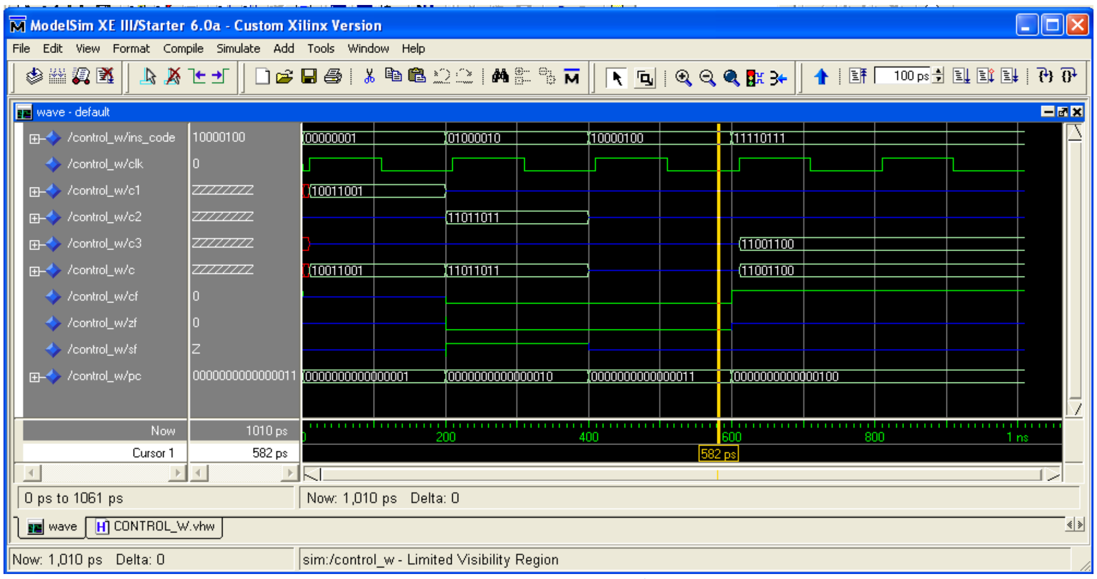

Fig. 7: Simulation result of CU

8. FPGA IMPLEMENTATION RESULTS:

The 8-bit soft processor is implemented and tested on Xilinx Spartan -3 FPGA in the laboratory environment. The logic resource requirement and utilization of FPGA resources is presented in Table 7 below.

Table 7: Logic requirement and FPGA resource utilization

HDL Synthesis Summary	Device Utilization Summary
Total Adders/Subtractors : 3 16-bit adder : 1 8-bit addsub : 1 9-bit addsub : 1	Number of Slices: 80 out of 3584 2.23%
Total Registers : 10 16-bit register : 1 8-bit register : 9	Number of Slice Flip Flops: 59 out of 7168 0.83%
Total Latches : 20 1-bit latch : 17 2-bit latch : 1 3-bit latch : 1 9-bit latch : 1	Number of 4 input LUTs: 126 out of 7168 1.76%
Total Comparators : 2 8-bit comparator less : 1 8-bit comparator greater : 1	Number of bonded IOBs: 58 out of 141 41.13%
Total Multiplexers : 24 8-bit 8-to-1 multiplexer : 1 1-bit 2-to-1 multiplexer : 22 9-bit 2-to-1 multiplexer : 1	Number of GCLKs: 2 out of 8 25%
Total Decoders : 1 1-of-4 decoder : 1	
Total Tristates : 17 8-bit tristate buffer : 3 1-bit tristate buffer : 14	
Total XORs : 1 8-bit xor2 : 1	

From Table 7 it is seen that the soft processor uses very small FPGA resources of Xilinx Spartan-3 FPGA (device XC3S400). So soft form of other peripheral chips can also be implemented in the same chip and the resulting FPGA can be called System-on-chip (SOC) in true sense.

9. CONCLUSION:

In this paper the design of an 8-bit soft-core microprocessor is discussed. The processor's internal components are described, and modelled in VHDL language using Xilinx ISE tool. The processor is divided into five internal sections and the control unit co-ordinates between the other four sections. Simulation result obtained using ModelSim software for each of the four blocks is individually presented first. Thereafter, simulation result of all the four said circuits under the co-ordination of CU is presented and described. Finally the complete design is downloaded into a FPGA chip in the laboratory and resource occupancy of such FPGA implementation is also presented.

REFERENCES:

[1] S. Brown & J. Rose, "FPGA and CPLD Architectures: A Tutorial", *IEEE Design & Test of Computers*, pp. 42-57, Summer 1996.
[2] V. Jayakrishnan and C. Parikh, "Embedded Processors on FPGA: Soft vs Hard", *ASEE North Central Section Conference,* pp.1-9, *2019*
[3] M. Amiri, F. M. Siddiqui, C. Kelly, R. Woods, K. Rafferty, B. Bardak, "FPGA-Based Soft-Core Processors for Image Processing Applications", Journal of Signal Processing System, vol. 87, pp: 139-156, 2017.
[4] J. G. Tong, I. D. L. Anderson and M. A. S. Khalid, "Soft-Core Processors for Embedded Systems", 18th International Confernece on Microelectronics (ICM) 2006, pp. 170-173, 2006.
[5] Ali Elkateeb, "A Processor Design Course Project: Creating Soft-Core MIPS Processor Using Step-by-Step Components' Integration Approach", International Journal of Information and Education Technology, Vol. 1, No. 5, pp. 432-440, 2011.
[6] M. Morris Mano , *Digital Logic and Computer Design*, Prentice Hall of India, New Delhi, 2002.
[7] D. P. Leach & A. P. Malvino, *Digital Principles and Applications*, 5/e, Tata McGraw-Hill, New Delhi, 2002.
[8] Carl Hamacher, Zvonko Vranesic & Safwat Zaky, *Computer Organization*, McGraw-Hill, New Delhi, 2002.