IMPLEMENTATION OF IEC 61131-3 STANDARD COMPATIBLE INSTRUCTION LIST PROCESSOR ON FPGA PLATFORM

Sagar D. Shedge[1], Prof. S.L. Tade[2]

[1] ME (VLSI and Embedded System), E&TC Department, PCET's PCCOE, Pune, Maharashtra, India
[2] Asst. Professor, E&TC Department, PCET's PCCOE, Pune, Maharashtra, India

ABSTRACT

This paper presents the design and implementation of an IEC 61131-3 standard compatible Instruction List processor used for PLC applications on SPARTAN 6 FPGA platform. At present as technology is increasing rapidly in plc processors, have become an important part of digital world. General purpose processors were used recently for plc applications, but due to timing constrains the general purpose processor was not efficient for the plc processes. Thus IEC 61131-3 standard gives the instruction list to be implemented for the plc dedicated processor. The IEC 61131-3 standard compatible processor with 5 stage instruction pipelined architecture is implemented in Verilog HDL.

Keyword: - *IEC, PLC, SPARTAN 6, FPGA, Verilog HDL.*

1. INTRODUCTION

Programmable Logic Controllers (PLC) are considered as most important part of an industrial automation applications for many years. Now-a-days the execution speed of the PLC is considered as a critical parameter while designing a system. The processors used in the PLC core are general purpose processors. These processors were unable to provide the high execution speed required by the industrial applications. The processor was needed to be dedicated for the PLC tasks rather than being general purpose. The instruction pipelining is also required to get high execution speed of the PLC program. After achieving this, the PLC can respond to the fast changing inputs. To achieve the fast response from the PLC, the design of Instruction List (IL) processor is proposed which is dedicated to the PLC applications.

PLC uses a relay logic or if this then that line execution. But this is not always able to model complex steps hence any complex execution and multiple step latency is more. But by single cycle instruction list processor instructions like AND, OR, ST, LD, MOV, SHIFT etc. which can execute logic in single cycle. If we need complex processing the PLC uses multistep so by using single cycle instruction list, we are doing complex processing single cycle. The Instruction List processor will be compatible with IEC 61131-3 standards.

Earlier, the RISC microprocessor used for Programmable Logic Controller was developed by (Gab SeonRho, 1995). It was the high performance processor implemented in VHDL on FPGA platform. High data bandwidth, conditional executive and reduced instruction set to obtain the high performance.

Later in 2000, the design of simple FPGA 16-bit RISC processor core and system-on-chip in synthesizable verilog was proposed by (Gray, 2000). The RISC instruction set architecture and the process for implementation of every part of the processor along with the design and implementation of on-chip RAM, peripheral bus and all peripherals are described by the author.

A hardware solution for supporting effective operation of CPU of PLC was proposed by (Chmiel, 2008). The response time of PLC was reduced by virtue of Dual Core bit-byte CPU. This architecture completely separates the bit-operations and byte-operations of the CPU as well as of I/O devices, so as to reduce the input-output operation time and hence the response time of PLC.

An approach to the design and construction of central processing units for programmable logic controllers implemented in the FPGA development platform by (M. Chmiel, 2011). The CPU structure is based on bit-word architecture and two types of control data exchange methods: with handshaking– control data are passed through the two flip-flop units with acknowledgement; without handshaking – control data are passed through the dual port RAM. Third unit – simple one processor – built to compare with the above two.

1.1 IEC 61131-3 Standard:

IEC 61131-3 is the third part (of 10) of the open international standard IEC 61131 for PLCs, published by International Electrotechnical Commission. This part of IEC 61131 deals with basic software architecture and programming languages of the control program within PLC. It defines five programming language standards.

a) **Ladder Diagram (LD):** It is a graphical programming language standard. This language was originally a written method to document the design and construction of relay racks as used in manufacturing and process control. This programming language represents a program by a graphical diagram based on the circuit diagrams of relay logic hardware.

b) **Functional Block Diagram (FBD):** It describes the function between input variables and output variables. A function is described as a set of elementary blocks. Input and output variables are connected to blocks by connection lines. It is another graphical type of programming language defined under IEC 61131-3 standard.

c) **Structured Text (ST):** It is a high level language that is block structured and syntactically resembles Pascal (a Programming language), on which it is based. The structured text language consists of complex statements and nested instructions such as Iteration loops (REPEAT-UNTIL; WHILE-DO), Conditional execution (IF-THEN-ELSE; CASE) and Functions (SQRT(); SIN()).

d) **Sequential Function Chart (SFC):** The sequential function chart standard is defined as, preparation of function charts for control systems, and was based on GRAFCET. SFC has elements to organize programs for sequential and parallel control processing.

e) **Instruction List (IL):** It is a low level language and resembles assembly. In instruction list processor, the variables and function call are defined by the common elements so different languages can be used in the same program. Program control is achieved by jump instructions and function calls.

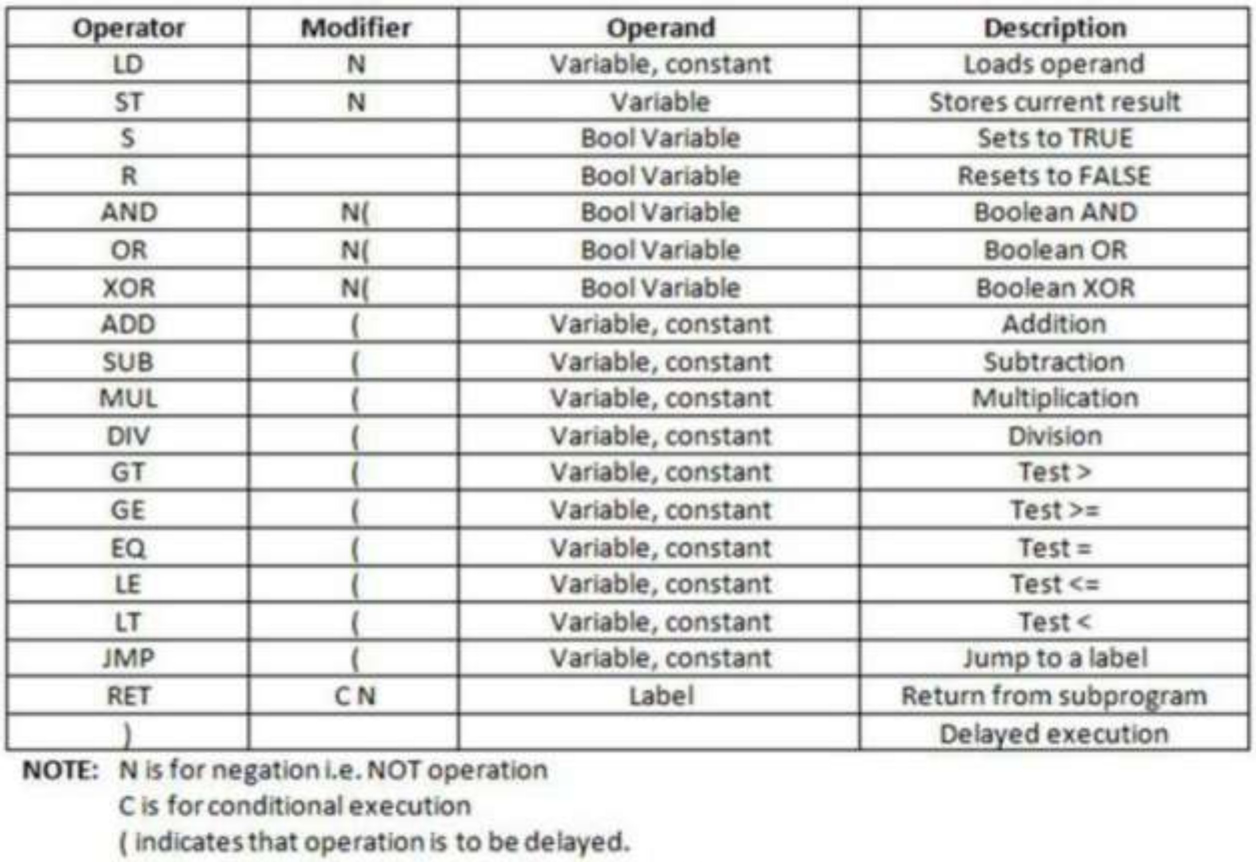
2. METHODOLOGY

The International Electro-technical Commission has defined standards for the programmable logic controllers, in the IEC 61131 suite. It has standardized five programming languages for PLC programming: Instruction List, Structured Text, Ladder Diagram, Function Blok Diagram, and Sequential Function Chart. One of these languages can be used to program a PLC.

The 'Instruction List' is a textual language which resembles the assembly language. It is used when compact size PLC programs are required, which takes less execution time. The IL is chosen as programming language for the proposed processor.

Table 1 shows the IEC 61131-3 standards for IL:

Table -1: IEC Instruction List Standards



Operator	Modifier	Operand	Description
LD	N	Variable, constant	Loads operand
ST	N	Variable	Stores current result
S		Bool Variable	Sets to TRUE
R		Bool Variable	Resets to FALSE
AND	N(	Bool Variable	Boolean AND
OR	N(	Bool Variable	Boolean OR
XOR	N(	Bool Variable	Boolean XOR
ADD	(	Variable, constant	Addition
SUB	(	Variable, constant	Subtraction
MUL	(	Variable, constant	Multiplication
DIV	(	Variable, constant	Division
GT	(	Variable, constant	Test >
GE	(	Variable, constant	Test >=
EQ	(	Variable, constant	Test =
LE	(	Variable, constant	Test <=
LT	(	Variable, constant	Test <
JMP	(	Variable, constant	Jump to a label
RET	C N	Label	Return from subprogram
)			Delayed execution

NOTE: N is for negation i.e. NOT operation
C is for conditional execution
(indicates that operation is to be delayed.

The first step in designing a processor is defining the instruction set. An IEC 61131-3 standards compatible instruction set is initially defined for the proposed processor. However, the instruction set is not limited to instructions defined by IEC, but more instructions have been defined to provide more functionality. Total 15 instructions are finalized for the proposed processor.

The finalized instruction set of proposed IL processor is given in table 2:

Table -2: Instruction set for proposed IL processor

TYPES OF INSTRUCTION	**INSTRUCTIONS**
Program Branch	END, JMP
Data Transfer	LD, ST
Immediate Data Transfer	Ldi
ALU	ADD, SUB, AND, OR, XOR, GT, GE, EQ, LT, LE

2.1 Instruction Fetch:

The program memory is a read-only memory i.e. ROM with 32-bit wide 256 locations. The number 256 arises from the fact that ROM is addressed by 8-bit address. Each of the locations can be filled with the 32-bit PLC program instruction. The ROM is purposely not synchronized with any clock signal so that asynchronous response to the JMP or END instruction is achieved.

Program counter is used to address the ROM to fetch an instruction. It outputs the address of the instruction which is to be fetched, and increments the address at each machine clock cycle. Whenever 'JR' signal is encountered, program counter outputs the address present on pc_in i.e. the address at which the program should jump.

The instruction register is a 32-bit register that holds the instruction fetched from the ROM and splits it into two parts viz. 6-bit instruction op-code and 26-bit instruction field.

This unit contains the program memory, a program counter and an instruction register, as shown in Fig. 1.

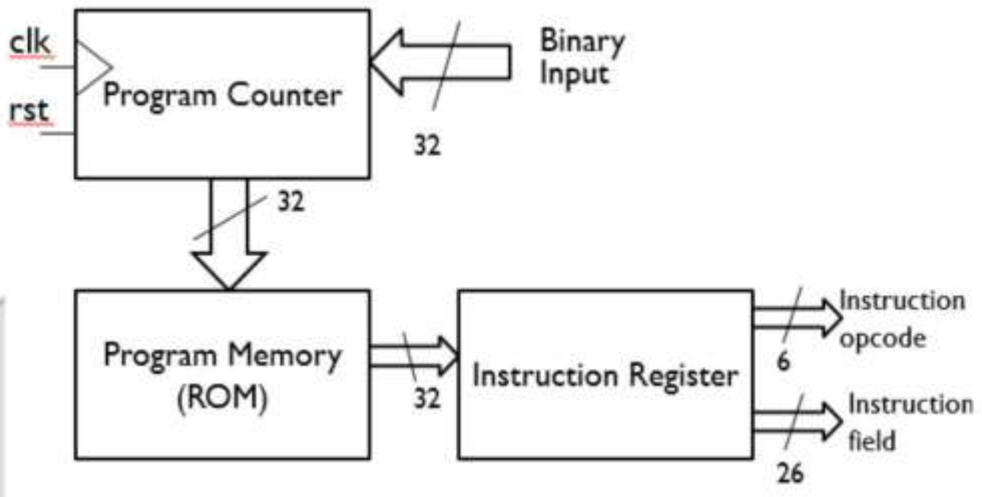


Fig -1 Instruction Fetch Unit

2.2 Instruction Decode Unit:

The instruction-decode and control unit, shown in Figure 2, decodes the fetched instruction by interpreting its 5-bit op-code and generates the control signals required to execute that instruction. The decode unit consist of a Finite State Machine (FSM). This FSM generates the control signals required for the Instruction Execution unit.

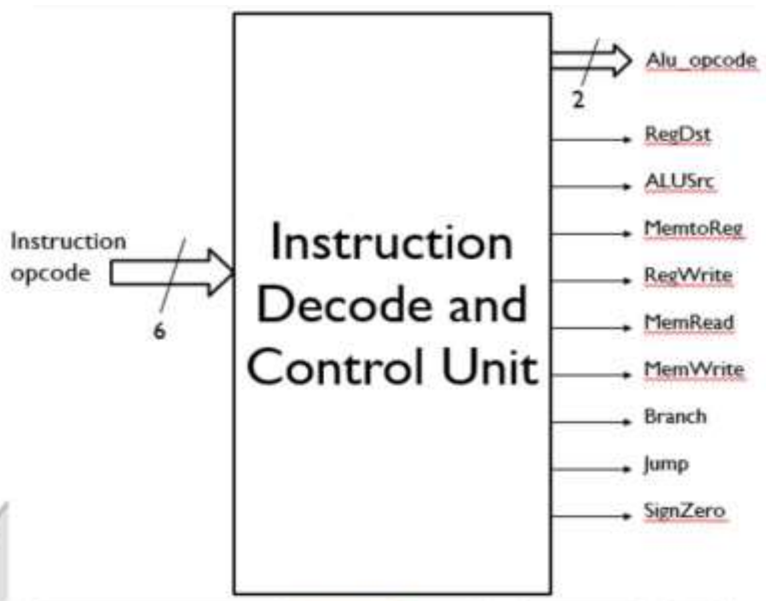


Fig 2: Instruction Decode and Control Unit

2.3 Instruction Execution Unit:

It performs all the arithmetic and logical operations according to the instructions. The ALU is designed to perform arithmetic operations: Addition, subtraction; logical operations: AND, OR, XOR and comparison operations.

The ALU of proposed processor is capable of performing operations on 16-bit data i.e. it is 32-bit ALU. Boolean operations are also done as byte-operations and LSB is used as result. One of the possible sources for ALU operations is selected through the multiplexer, as shown in Figure 3.

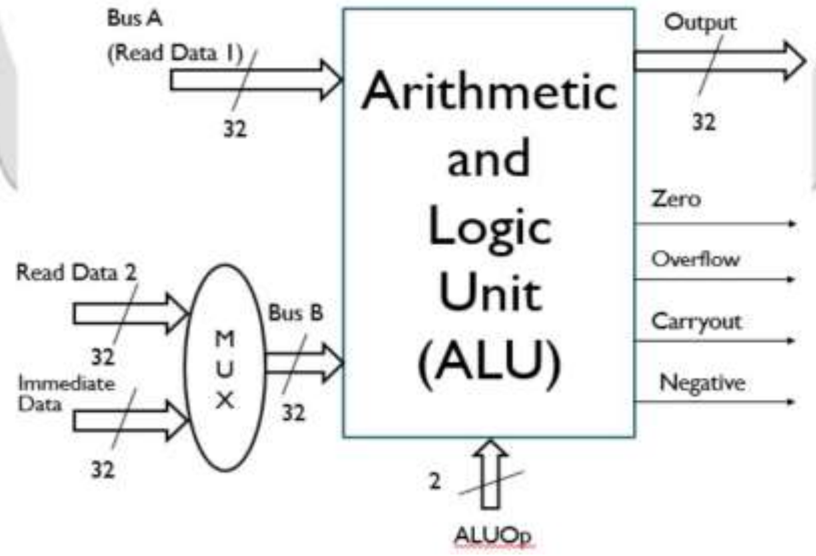


Fig -3 ALU in Execution Unit

2.4 Memory Access Stage:

Memory Access stage is used for accessing the data from the data memory of the processor. The results from the execution unit are given as the input for the data memory block. The data from this memory can be used for the further processing or for the storage purpose. The multiplexers are used for selecting the data from the different signals obtained from the execution unit. The data memory block and signals used to execute the block are shown in fig. 3.7.

The 32-bit Address is obtained from the output of the ALU.

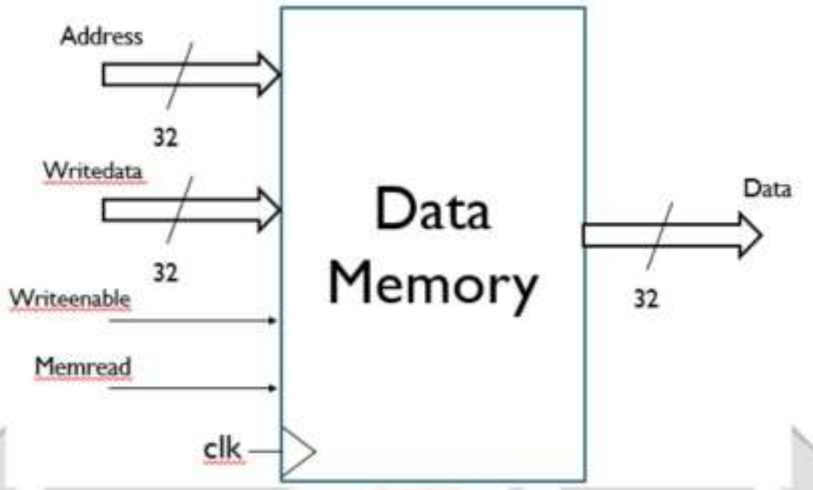


Fig -4: Data Memory Block

2.5 Write Back Stage:

In the Write back stage of the processor, the data is written back to the register file. The Multiplexer is used to select the data stream to be written back to the register memory, among ALU result or data stored in Data Memory. The write back operation is shown in the fig. 5

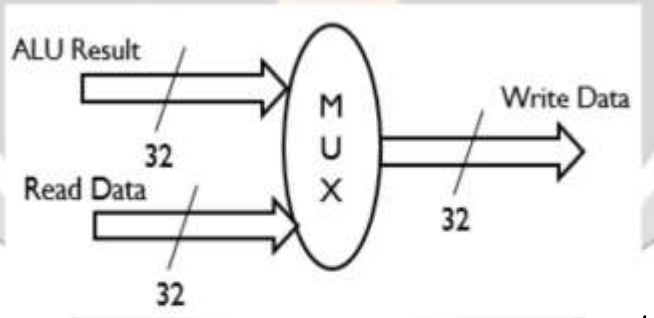
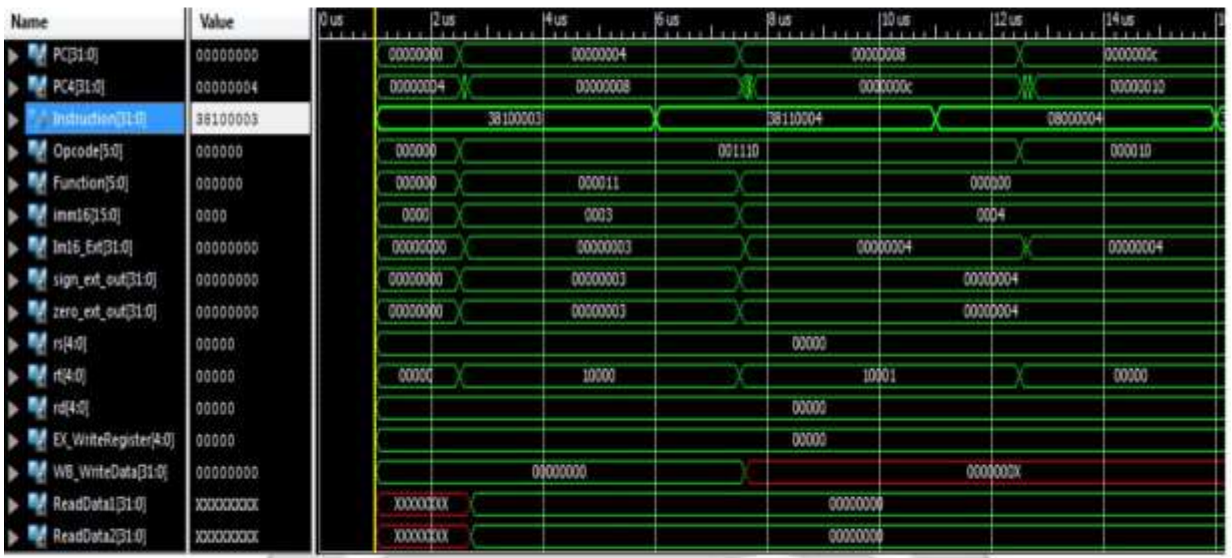
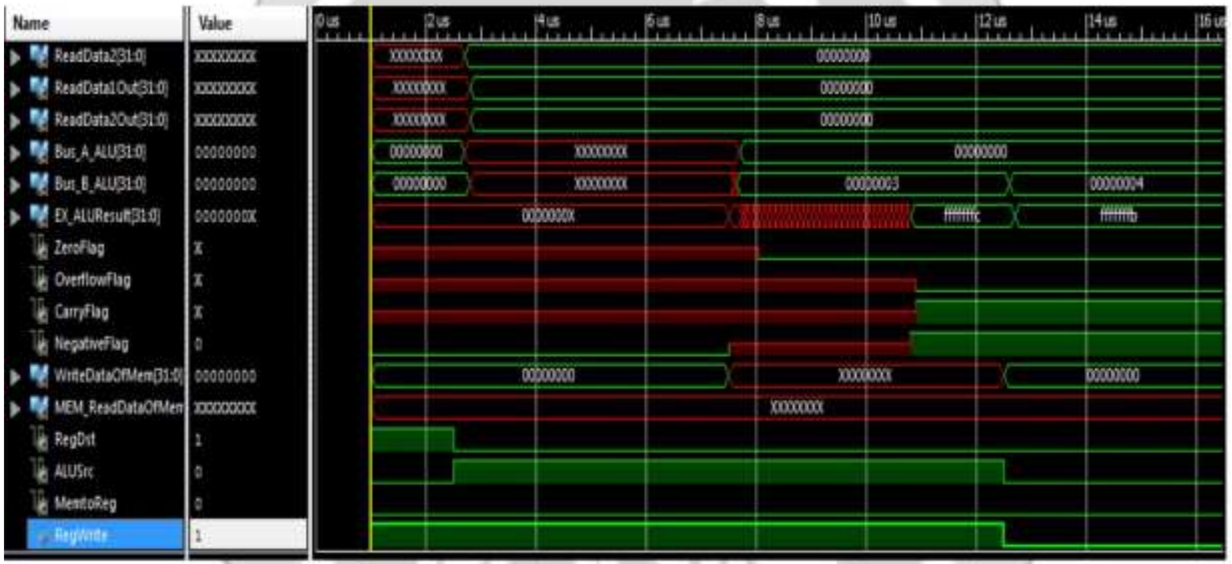


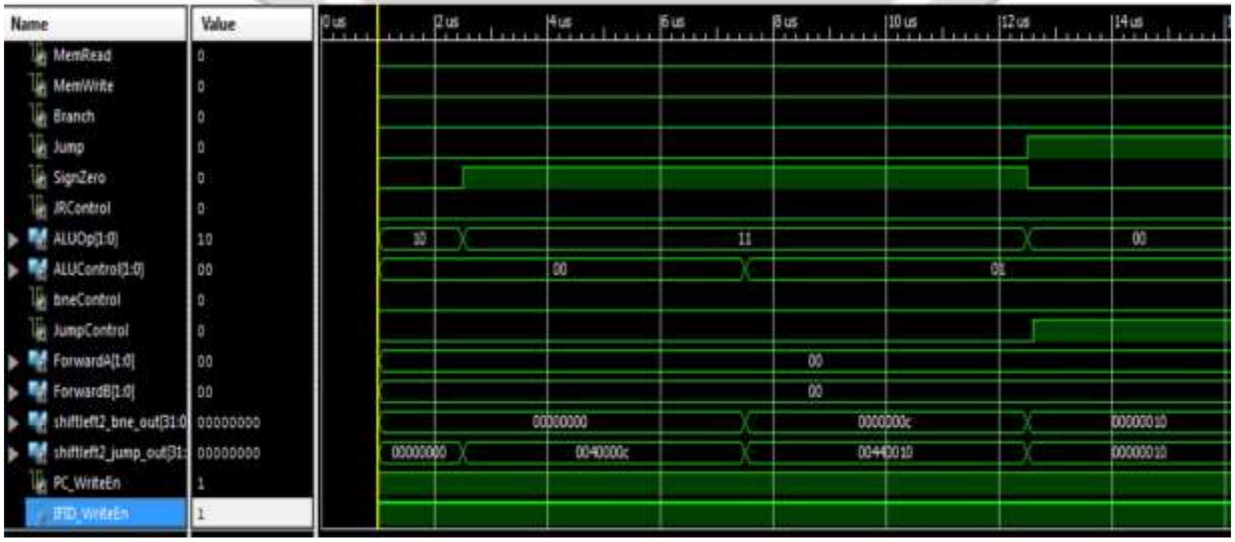
Fig -5: Memory write-back Block

4. SIMULATION RESULTS

The Simulation Results obtained after implementing the designed processor on Spartan 6 FPGA platform, in Xilinx 14.04. The simulation is performed in Isim simulator. The set of instructions are simulated in Isim Simulator. Some of the instructions simulations are shown below


Name
Value
PC[31:0] 00000000
PC4[31:0] 00000004
Instruction[31:0] 38100003
Opcode[5:0] 000000
Function[5:0] 000000
imm16[15:0] 0000
Im16_Ext[31:0] 00000000
sign_ext_out[31:0] 00000000
zero_ext_out[31:0] 00000000
rs[4:0] 00000
rt[4:0] 00000
rd[4:0] 00000
EX_WriteRegister[4:0] 00000
WB_WriteData[31:0] 00000000
ReadData1[31:0] XXXXXXXX
ReadData2[31:0] XXXXXXXX
0 us
2 us
4 us
6 us
8 us
10 us
12 us
14 us


Name
Value
ReadData2[31:0] XXXXXXXX
ReadData1Out[31:0] XXXXXXXX
ReadData2Out[31:0] XXXXXXXX
Bus_A_ALU[31:0] 00000000
Bus_B_ALU[31:0] 00000000
EX_ALUResult[31:0] 0000000X
ZeroFlag X
OverflowFlag X
CarryFlag X
NegativeFlag 0
WriteDataOfMem[31:0] 00000000
MEM_ReadDataOfMem XXXXXXXX
RegDst 1
ALUSrc 0
MemtoReg 0
RegWrite 1
0 us
2 us
4 us
6 us
8 us
10 us
12 us
14 us
16 us


Name
Value
MemRead 0
MemWrite 0
Branch 0
Jump 0
SignZero 0
JRControl 0
ALUOp[1:0] 10
ALUControl[1:0] 00
bneControl 0
JumpControl 0
ForwardA[1:0] 00
ForwardB[1:0] 00
shiftleft2_bne_out[31:0] 00000000
shiftleft2_jump_out[31:0] 00000000
PC_WriteEn 1
IFID_WriteEn 1
0 us
2 us
4 us
6 us
8 us
10 us
12 us
14 us

5. CONCLUSION

The FPGA platform is selected for the implementation of processor due to high performance advantage over the CPLD. Instruction list processor is selected because of the ease of use over the other 4 types of the PLC processor design. The five-stage pipeline is proposed for the boosting of speed of the operation. The implemented design will be simulated using Xilinx Isim simulator.

6. REFERENCES

[1] Gab SeonRho, Kyeonog-hoon Koo, Naehyuc Chang, Jaehyun Park,Yeong-gi Kim and Wook Hyun Kwon, "*Implementation of a RISC microprocessor forprogrammable logic controllers*", Elsevier Science B.V, Microprocessors and MicrosystemsVolume 19 .Number 10, December. 1995.

[2] Jan Gray, "*Designing a Simple FPGA-Optimized RISC CPU and System-on-a-Chip*", Gray Research LLC, Copyright 2000.

[3] M. Chmiel, "*On reducing PLC response time*", Bulletin of Polish Academy ofSciences, Technical Sciences, Vol. 56, No. 3, 2008.

[4] M. Chmiel, J. Mocha, E. Hrynkiewicz, A. Milik, "*Central Processing UnitsforPLC implementation in Virtex-4 FPGA*", Preprints of the 18th IFAC WorldCongress Milano (Italy) August 28 - September 2, 2011,pages 7860-7865.

[5] Piotr Chodorowski, Miroslaw Chmiel, *"IEC 61131-3 Compliant PLC Structure based on FPGA multi-core solution"*, International Conference on Signals and Electronic Systems (ICSES) September 5-7, 2016, Krakow, Poland.

[6] Miroslaw Chmiel, Wojciech Kloska, Dariusz Polok, Jan Mocha, *"FPGA-based Two-processor CPU for PLC"*, International Conference on Signals and Electronic Systems (ICSES) September 5-7, 2016, Krakow, Poland.

[7] http://www.calc1.kss.ia.posl.pl/content/dydaktyka