Proceedings of the Third International Conference on Computing Methodologies and Communication (ICCMC 2019)
IEEE Xplore Part Number: CFP19K25-ART; ISBN: 978-1-5386-7808-4

High Performance Instruction List Processor on FPGA Platform

Shilpa K. Rudrawar
Department of Electronics Engg.
School of Electrical Engg.
MIT Academy of Engg, Alandi (D)
Pune, India
skrudrawar@etx.maepune.ac.in

Dr. Dipti Sakhare
Department of Electronics Engg.
School of Electrical Engg.
MIT Academy of Engg, Alandi (D)
Pune, India
dysakhare@etx.maepune.ac.in

***Abstract*— Programmable Logic Controllers (PLC) are used in industry for automation. Now-a-days it is a standard practice to include safety interlocks along with logic functions in the PLC programming. The use of smart sensors also gives very fast changing inputs to the PLC. This makes the response time of PLC an important issue. The scan time of PLC depends upon two parameters viz. the length of PLC program and operating frequency of CPU. Thus, in order to improve PLC response time, it is necessary that scan time of PLC should be less. This paper suggests an IEC 61131-3 standards compatible PLC application dedicated Instruction List (IL) processor with a three-stage instruction pipeline on FPGA platform for better response time.**

Keywords—Dedicated Processor, Fast response PLC, Instruction Pipeline, Single Cycle Execution

I. INTRODUCTION

The Programmable Logic Controllers (PLC) is used widely in many industries for the process and design automation purpose. The ability to handle process parameters and control the processes makes PLC the best choice for their applications. PLC has input and output modules and a processor at its core. The performance of PLC greatly depends on the processor used. Many commercial PLCs use the general purpose processors, which are capable of handling byte and word operations. However, over 70% of PLC operations are Boolean (bitwise) operations [1]. The use of HDLs (Hardware description languages) allows modeling, synthesis, reconfigurability and rapid prototyping of application specific design using high density FPGA platform and can function for real time requirements which is also not expensive as compare to hardware prototyping [2].

High performance PLC dedicated processor implemented with combining general processor and the PLC application specific instruction set processor (ASIP). Instruction formats and instruction sets are designed for same. The architecture is designed to accelerate the instructions execution. The high performance of PLC dedicated processor is explained with the use of a ladder compiler which compile ladder program to binary code for PIC 16 or AVR [3]. 16-bit RISC processor has been designed using VHDL. Hierarchical approach has been used and modeled design using behavioral and structural programming. Pipelining is used to improve the overall CPI (Clock Cycles per Instruction. [4].

Moreover, operations in general purpose processor are implemented as micro-codes which take many steps for execution of any instruction. Thus, general purpose processor proves to be inefficient. If the processor used in PLC is designed for dedicated PLC operations, the performance of the PLC can be greatly enhanced. Further, implementing the dedicated processor as an instruction pipelined processor speeds-up the operation of PLC [5][6].

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. The result of the design is significant increase in the speed of bit-byte central unit. Most of the operations are bit-operations and thus, large gains are achieved through separate bit-processing, which is done independent of byte-processing [7].

An approach to the design and construction of central processing units for programmable logic controllers implemented in a FPGA development platform. Presented units are optimized for minimum response and throughput time. 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. The paper presents specific timers/counters hardware construction solution. The architecture has its own instruction list with dedicated assembler. The design is a fully custom design, optimized for the greater performance [8].

II. METHODOLOGY

The proposed dedicated IEC 61131-3 compatible high speed PLC processor is designed with following specifications to reduce PLC scan time:

- ALU: 8-bit operands
- Maximum number of inputs: 64
- Maximum number of outputs: 64
- Number of timers: 8
- Number of counters: 8
- Memory : RAM: 64-bit RAM, 64- byte RAM, ROM: 256 x 15-bit ROM

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 23 instructions are finalized for the proposed processor. The instruction set of proposed IL processor is shown in Table 1.

TABLE I. INSTRUCTION SET

Type of instruction	Instructions
Program branch	END, JMP
Data transfer	LD, ST
Immediate data transfer	LDi
ALU	ADD, SUB, AND, OR, XOR, GT, GE, EQ, LE, LT
Timer and Counter	TON, TOF, RTO, CUP, CDN, SRF, RST, STdn

After defining the set of instructions, the instruction format is defined. Instruction format consists of two parts viz. Instruction Op-Code and Instruction Field. Instruction Op-Code is a 5-bit binary code which represents each instruction uniquely. Instruction Field is 10-bit code which consists of parameters which needs to be specified for a particular instruction to execute, e.g. address of the source for a load operation. Thus, length of any instruction is total 15-bit. Instruction format varies for various types of instructions. Ultimately, there are six different formats for 23 instructions. Fig.1. shows various instruction formats used for the IL processor.

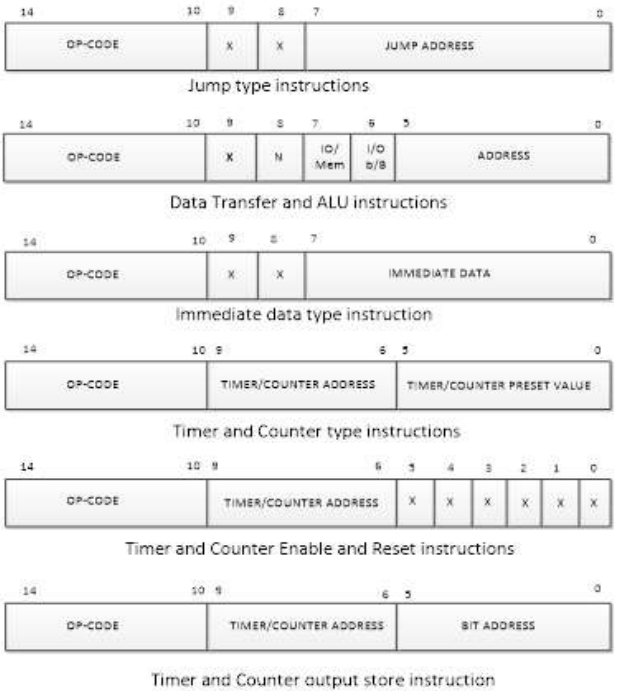

Fig. 1. *Instruction formats*

Next step is the design of various functional blocks of the processor. These are designed taking into consideration the instruction set. Once tentative blocks are designed, HDL code for each block is written using VHDL language. The Xilinx ISE v12.3 is used for this purpose. Every functional block is synthesized individually and tested using appropriate test-benches.

When all the functional blocks are functioning properly, they are integrated into a single design using structural modelling style of VHDL, by creating data and control paths. The final design is then tested using test-bench [8].

III. ARCHITECTURAL DESCRIPTION

The architecture of the proposed IL processor can be divided into three functional stages viz. Fetch Unit, Decode Unit & Execution Unit. These functional units are separated by the pipeline registers which play an important role in the instruction pipelining. The schematic is shown in Fig. 2.

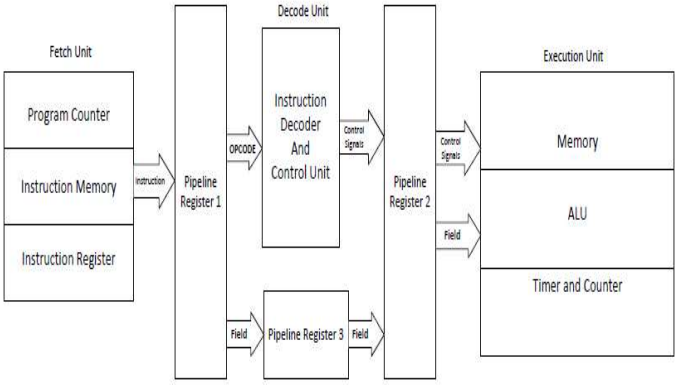

Fig. 2. *Architectural block diagram*

A. Instruction Fetch Unit

This unit contains the program memory, a program counter and an instruction register, as shown in the Fig. 3. The program memory is a read-only memory i.e. ROM with 15-bit wide 256 locations. Each of the locations can be filled with the 15-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.

The program memory is addressed by an 8-bit program counter. Program counter outputs the address of the instruction which is to be fetched, and increments the address at each clock cycle. Whenever 'jmpend' 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 15-bit register that holds the instruction fetched from the ROM and splits it into two parts viz. 5-bit instruction op-code and 10-bit instruction field.

The schematics of timer and counter are shown in Fig. 7. The timer is driven by a clock signal clk_t. When t_en signal is present, on-delay timer starts timing for the clock ticks from zero to the 6-bit 'preset' value. When finished, it generates an output in the form of DN bit. The off-delay timer starts timing when t_en signal is low. The retentive on-delay timer is same as on-delay timer except that it retains the accumulated timing value unless an explicit reset is given.

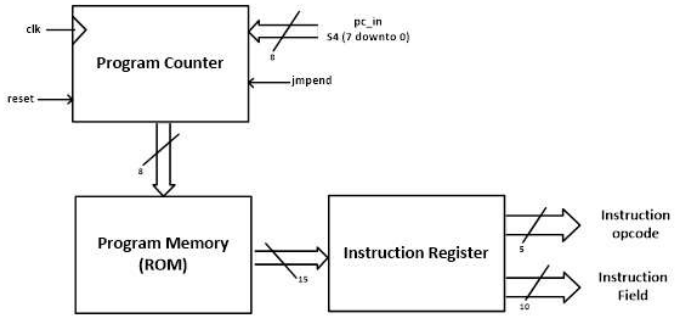

Fig. 3. *Instruction fetch unit.*

The counter on the other hand, is not driven by any clock, but counts the low to-high transitions on the c_en signal. It also produces a DN bit output when finished counting. The up-counter counts in ascending order while down-counter counts in descending order.

B. Instruction Decode Unit

The instruction decoder and control unit, shown in Fig. 4, is the heart of the processor. It performs the most important tasks of decoding the fetched instruction by interpreting its 5-bit op-code and generating the control signals required to execute that instruction.

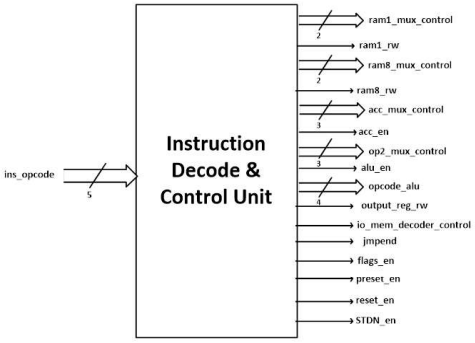

Fig. 4. *Instruction decode unit*

There are total 16 control signals that are generated by control unit of the proposed processor. These control signals are applied to the functional units in the execution unit and fetch unit.

C. Instruction Execution Unit

The instruction execution unit contains all the digital circuitry that is required to implement the defined instruction set of the processor. Following are the functional blocks of the execution unit.

1) Accumulator and ALU

Accumulator is an 8–bit register that holds one of the operands of ALU operations, and also the result of all the ALU operations. One of the possible sources of the accumulator input is selected through the multiplexer, as shown in Fig. 5

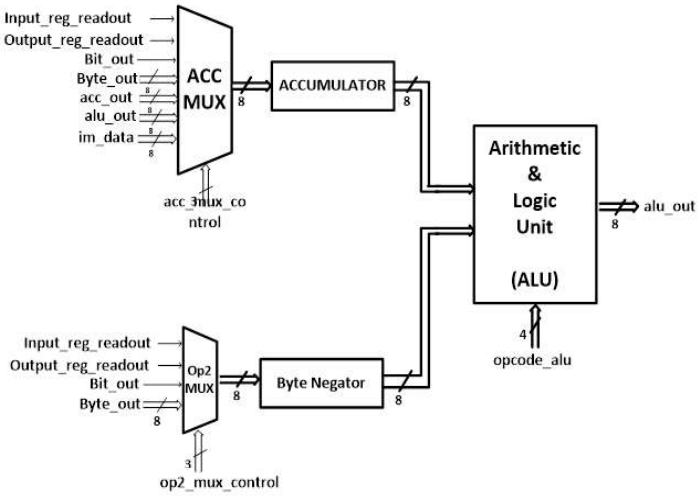

Fig. 5. *Accumulator and ALU*

The Arithmetic Logic Unit i.e. ALU 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 8-bit data i.e. it is 8-bit ALU. Boolean operations are also done as byte-operations and LSB is used as result. One of the inputs to the ALU is form the accumulator and other operand for ALU operations is selected through the op2 multiplexer.

2) Data Memories

Data for the ALU operations is stored in the data memories. Four data memories, viz. bit-RAM, byte-RAM, Input Register File and Output Register File, are used in the proposed IL processor.

Bit-RAM, shown in Fig. 6.a, is a 64 location bit-memory, used to store bit data. The data sources for bit-RAM are: ALU output for bit-operations and timer & counter DN bit output. The data to be stored is selected through a multiplexer. The bit-negator block is the implementation of bit-8 of instruction filed. It is affected by ST (store) instruction. Byte-RAM, shown in Fig. 6.b, is a 64 location byte memory, used to store byte data. Data source is ALU output only. The multiplexer is intentionally used so that high impedance can be presented at RAM input whenever required. The byte negator is affected by ST (store) instruction.

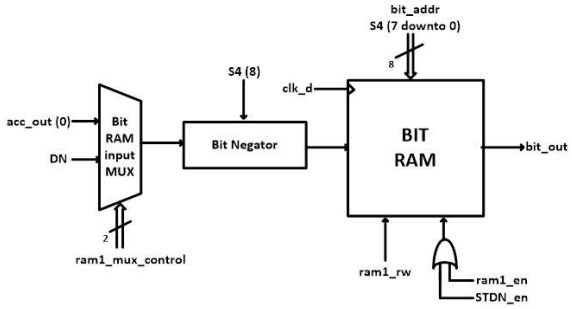

Fig. 6. *a. Bit RAM*

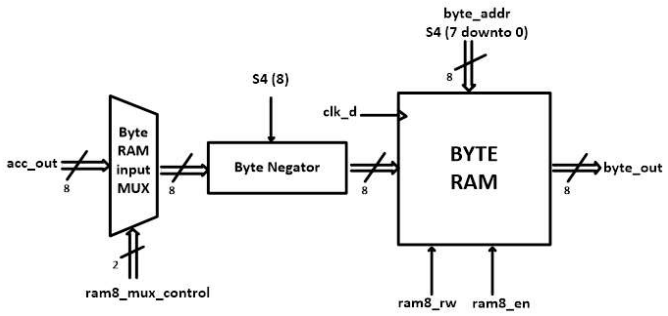

Fig. 6. *b. Byte RAM*

Input Register File, shown in Fig. 6.c, is a 64 location bit memory. This memory is always written by external inputs and is readable. It is assumed that PLC inputs are always present in this memory and processor uses the current values of inputs. Output Register File, shown in Fig. 6.d, is also a 64 location bit memory which contains PLC output image. This memory is writable as well as readable. It is assumed that PLC outputs are updated from this memory, upon completion of every execution cycle.

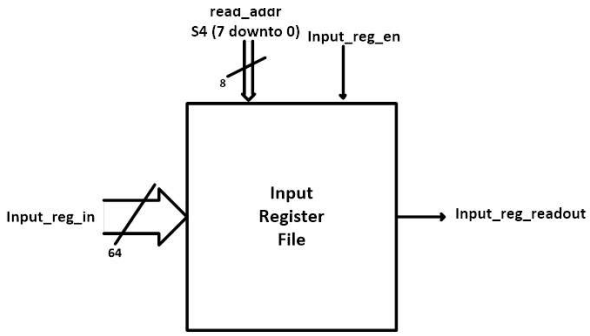

Fig. 6. *c. Byte RAM*

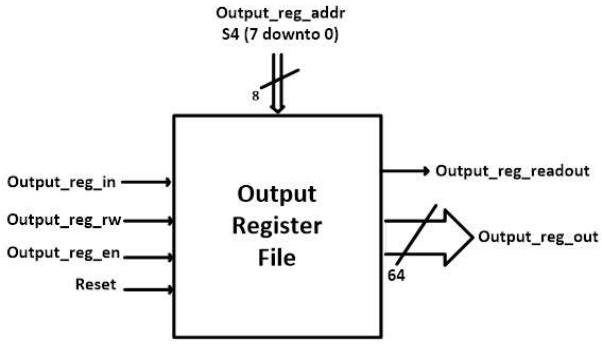

Fig. 6. *d. Byte RAM*

3) Input-Output & Memory Decoder

The data read and write operation on the above mentioned memories is performed in multiplexed way. The memory to perform these operations in selected using two bits in the instruction field. These two bits are decoded, using a 2:4 decoder to generate an enable signal. The control unit has the control over this decoded so that enable signal is generated only when desired.

4) Timer and Counter

Timers and Counters are very important entities for a PLC. Most of the industrial applications are based on timers and counters. In general words, timers and counters, both are kind of delay generators. Timer is time-based delay generator while counter is external event based delay generator.

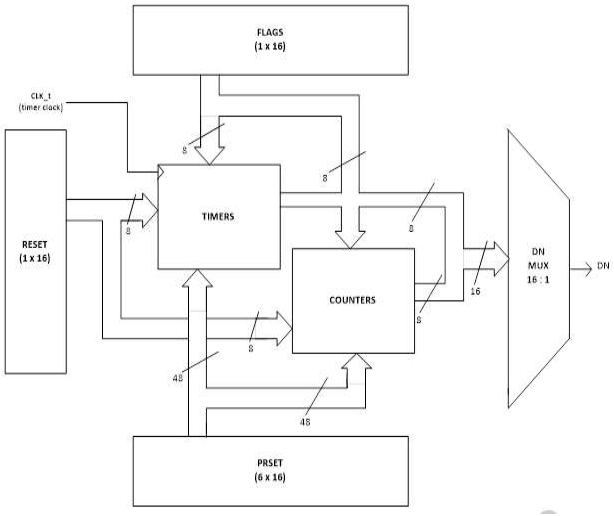

Fig. 7. *Timer and couter*

TABLE II. TYPES OF TIMER & COUNTER

Sr. No.	Timer / Counter	Number of entities
1	On-delay Timer	3
2	Off-delay Timer	3
3	Retentive On-delay Timer	2
4	Up Counter	4
5	Down Counter	4

Table II shows number of different timer and counters provided in proposed IL processor.

The complete timer and counter module consists of all 8-timers, 8-counters, a FLAGS register to indicate enable or disable condition for respective timer/counter, PRESET register that hold preset values of all timers & counters, RESET register that generate local reset signal for each individual timer/counter, The 6-bit 'preset' value can be set in counter as well as timer or counting can be done up to 6 bit combination. After counting or timing is over a multiplexer passes one of the DN output bits from respective timers & counters to the bit-RAM for further processing. The complete timer-counter module is shown in the Fig. 7.

The schematics of timer and counter are shown in Fig. 8. The timer is driven by a clock signal clk_t. When t_en signal is present, on-delay timer starts timing for the clock ticks from zero to the 6-bit 'preset' value. When finished, it generates an output in the form of DN bit. The off-delay timer starts timing when t_en signal is low. The retentive on-delay timer is same

as on-delay timer except that it retains the accumulated timing value unless an explicit reset is given.

The counter on the other hand, is not driven by any clock, but counts the low-to-high transitions on the c_en signal. It also produces a DN bit output when finished counting. The up-counter counts in ascending order while down-counter counts in descending order.

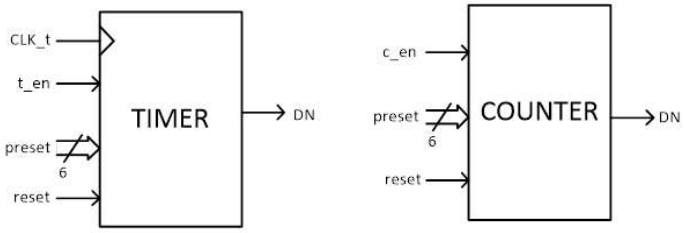

Fig. 8. *Timer-counter module*

5) Pipeline Implementation

The proposed IL processor suggests that instruction pipeline be used for speeding-up the execution. The meaning of instruction pipeline is that the processor has multiple functional blocks which are working simultaneously to execute any instruction just like an assembly line in a manufacturing plant. The instruction travels through these functional blocks under governance of a clock signal.

The very popular ARM processor design uses the instruction pipeline [10]. The ARM design states that 'key element in a pipelined processor is the pipeline register without which the pipelining is impossible to implement. The pipeline register is as good as a D- Flip Flop. It simply separates the functional units and passes the inputs to the output on the rising clock signal. Thus, the instruction propagates through the so formed pipeline on every rising clock signal.

The timing diagram of an ideal three-stage pipelined processor is shown in Fig.9a.

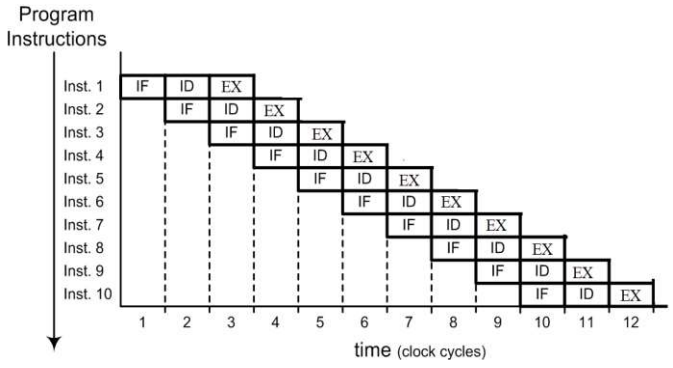

Fig. 9. *a. Three stage pipelined IL processor*

The pipeline registers as shown in Fig. 2 are introduced in between the three functional units of the processor so as to implement an instruction pipeline. The pipeline registers simply act as a D flip-flop, driven by same clock. The instruction passes through each functional unit and consequently, all functional units are operating in parallel. When first instruction is being executed, the second is in decoding stage while the first is being fetched. Finally, from the third machine clock, every clock cycle gives an output.

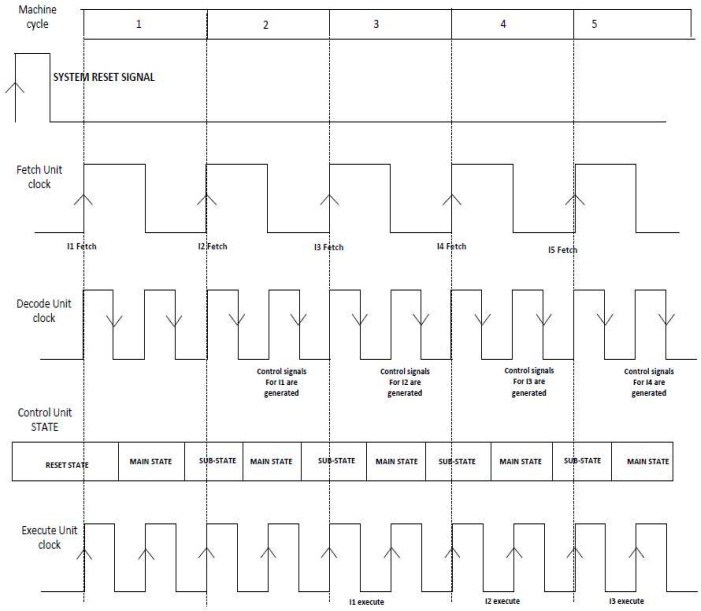

Fig. 10. *b.Timing diagram of the IL processor*

The pipeline hazards that may arise are avoided by some design considerations as following:

- ROM is not driven on clock so that program jump is achieved as soon as instruction is decoded.
- The control unit and execute unit are properly synchronized by introducing Pipeline Register 3.
- Note that decode unit clock frequency is double the clock frequency of fetch unit. The state of the control unit which is FSM is also shown with respect to the clock signal.

The timing diagram of the proposed IL processor is shown in Fig. 9b.

IV. HARDWARE IMPLEMENTATION

The IL processor design is implemented on Field Programmable Gate Array i.e. FPGA chip. The hardware platform chosen is Xilinx Spartan 3E FPGA XC3S500E. This FPGA has 500K number of gates on-chip to implement the digital logic. The development board used is "Papilio One 500k" manufactured by the Gadget Factory. The inputs to the IL processor implemented on-board are given through push-buttons and toggle-switches. The outputs are observed on the LEDs. The board has an oscillator of 32MHz which is used as clock signal for the design.

V. RESULTS

The design is simulated in software using the Xilinx ISim simulator v12.3. A ladder diagram program is converted into IL program as shown in Fig. 12 and it runs in simulation as

well as on the hardware [11]. The results are found to be correct on both software and hardware platforms. While the software simulation Fig. 13. gives total timing analysis and hardware execution Fig. 11. shows the correctness of the design.

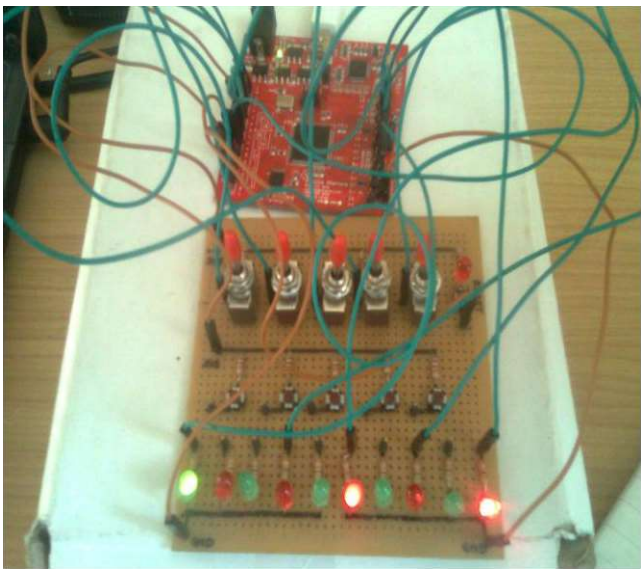

Fig. 11. *Hardware execution of the design*

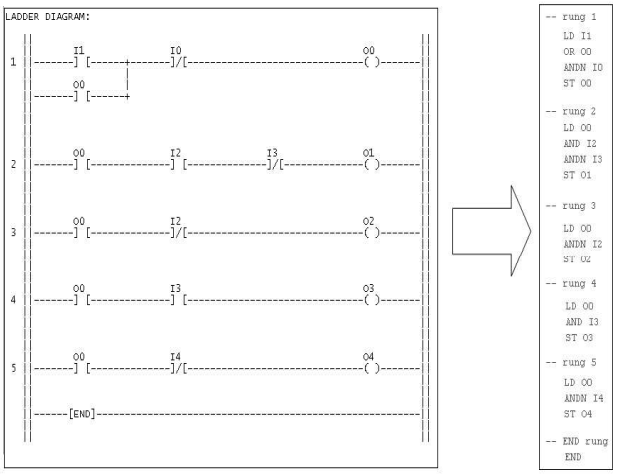

Fig. 12. *Conversion of Ladder-program into IL-program*

To further enhance the speed of design, a Xilinx technology named Digital Clock Manager (DCM) [12] is used. The use of DCM permits the input clock frequency to be converted into any desirable frequency.

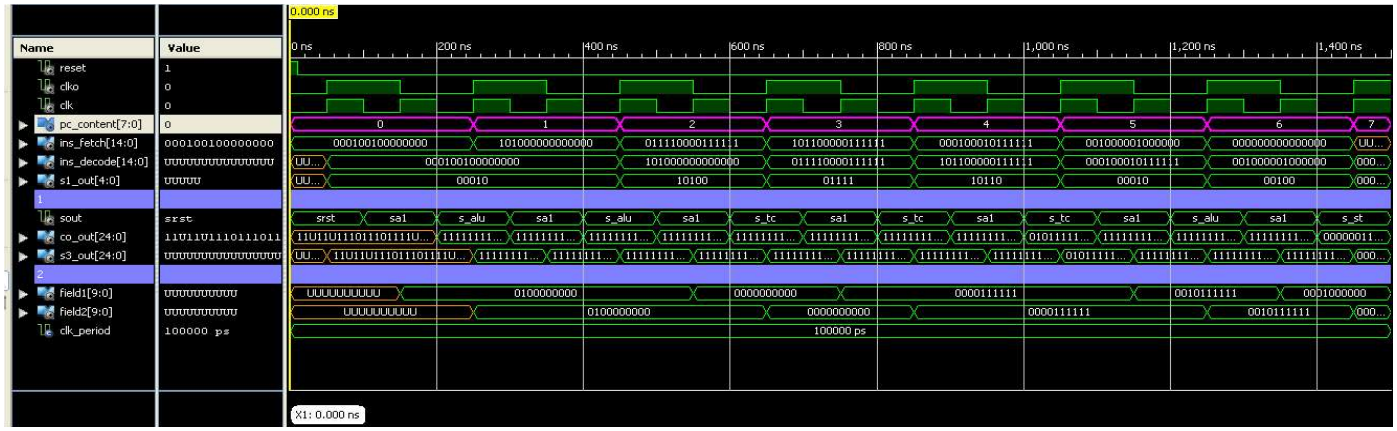

Fig. 13. *Software simulation of IL processor design*

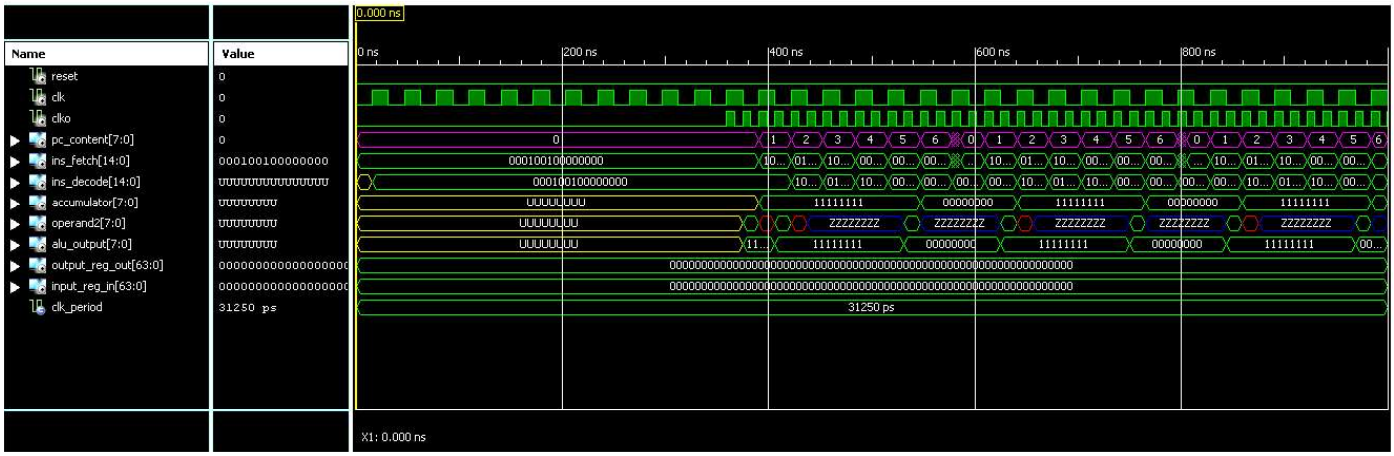

Fig. 14. *Software simulation of IL processor design using DCM*

The FPGA development board provides an oscillator with 32MHz clock frequency. This frequency is doubled to 64MHz using DCM. The 32MHz clock is used to drive fetch unit while 64MHz clock is used to drive decode and execute unit which requires double the frequency than fetching. Thus, each instruction now executes in a single cycle of the on-board clock oscillator. Fig. 14. shows the simulation of the design using DCM. Comparison of required number of clock cycles per instruction

INSTRUCTIONS	CLOCK CYCLES REQUIRED	
	Embedded Microprocessor by Snaider Carrillo L., Agenor Polo Z., Mario Esmeral P	Proposed IL Processor
LD	19	1
LDi	NA	1
ST	15	1
R	15	NA
S	15	NA
AND	5	1
OR	5	1
XOR	5	1
ADD	5	1
SUB	5	1
GT	5	1
GE	5	1
EQ	5	1
LE	5	1
LT	5	1
JMP	5	1
RET	1	1
TON	NA	1
TOF	NA	1
RTO	NA	1
CUP	NA	1
CDN	NA	1
SRF	NA	1
STdn	NA	1
END	NA	1

The number of clock cycles required to execute the instructions is compared with a previous implementation of IL processor designed by Snaider Carrillo L., Agenor Polo Z., Mario Esmeral P. [9], in the Table 3. From this table, it can be seen that, one or two instructions of this previous implementation are executed in single clock cycle while most of them require 5 clock cycles. On the other hand, every instruction of proposed IL processor is executed in a single clock cycle.

The device utilization summary of the Xilinx FPGA XC3S500E VQ100 for the IL processor design is shown in Fig. 15.

Device Utilization Summary				[-]
Logic Utilization	Used	Available	Utilization	Note(s)
Total Number Slice Registers	249	9,312	2%	
Number used as Flip Flops	176			
Number used as Latches	73			
Number of 4 input LUTs	482	9,312	5%	
Number of occupied Slices	340	4,656	7%	
Number of Slices containing only related logic	340	340	100%	
Number of Slices containing unrelated logic	0	340	0%	
Total Number of 4 input LUTs	506	9,312	5%	
Number used as logic	457			
Number used as a route-thru	24			
Number used for 32x1 RAMs	18			
Number used as Shift registers	7			
Number of bonded IOBs	19	66	28%	
Number of BUFGMUXs	4	24	16%	
Average Fanout of Non-Clock Nets	3.48			

Fig. 15. *Device utilization of XC3S500E VQ400*

The Xilinx synthesis report shows the timing summary, which is much important part of the design, as shown in Fig. 16. It can be seen that the design can be run at a maximum clock frequency of 180.897 MHz. Thus, the execution time of an instruction can be as small as 5.528 ns, if the design is run at maximum possible frequency.

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 5.528ns (Maximum Frequency: 180.897MHz)
   Minimum input arrival time before clock: 13.244ns
   Maximum output required time after clock: 4.931ns
   Maximum combinational path delay: No path found
```

Fig. 16. *Timing summary of the design*

VI. CONCLUSION

The work in the paper suggests a dedicated processor to be used as a PLC core. The three-stage instruction pipelining is implemented so that PLC response is boosted to large extent. The design of processor uses its own PLC application specific instruction set, whose implementation is optimized by Xilinx Synthesis Technology.

Every instruction supported by the processor is executed in the single clock cycle. This is achieved by making use of Xilinx DCM technology. Finally, the design is implemented on FPGA and the design is tested successfully on hardware platform.

REFERENCES

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

[2] J. Viñas, N. Díaz, y H. Campanella, “Modem Bandabase (Nivel Físico) Basado en el Estándar IEEE 802.11b”,Internacional Conference on Reconfigurable Computing and FPGA, México, September 20-21, 2004, pp. 1-10

[3] Shuting-zeng , Zhijia-yang, “High performance architecture design of PLC dedicated processor” Industrial informatics laboratory Shenyang institute of automation, Graduate School of the Chinese Academy of Sciences Shenyang, China 978-1-4244-6542-2/$26.00 © 2010 IEEE

[4] Pravin S. Mane, Indra Gupta, M. K. Vasantha, “Implementation of RISC Processor on FPGA” Computer Science & Engineering Department,

Mody Institute Of Technology & Science, Lakshmangarh-332311."Electrical Engineering Department, Indian Institute of Technology Roorkee, Roorkee-247667, 1-4244-0726-5/06/$20.00 '2006 IEEE 2006 IEEE

[5] Andreas Otto and Klas Hellmann, “IEC 61131 A general overview & emerging trends” Integration of Motion Control and Safety Functions December 2009 IEEE Industrial Electronic Magazine, 2009 n IEEE. Industrial Electronic Magazine 1932-4529/09/$26.00&2009IEEE

[6] Götz Kappen, Lothar Kurz, Tobias G. Noll “Comparison of ASIP and Standard Microprocessor based Navigation Processors “ Chair of Electrical Engineering and Computer Systems, Aachen University, Germany 2007

[7] M. Chmiel, “On reducing PLC response time”, Bulletin of Polish Academy of Sciences, Technical Sciences, Vol. 56, No. 3, 2008

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

[9] Snaider Carrillo L., Agenor Polo Z., Mario Esmeral P."Design and Implementation of an Embedded Microprocessor Compatible with IL Language in Accordance to the Norm IEC 61131-3", Proceedings of the 2005 International Conference on Reconfigurable Computing and FPGAs (ReConFig 2005) 0-7695-2456-7/05-$20.00 © 2005 IEEE

[10] XU Mei-hua, RAN Feng, CHEN Zhang-jin, KANG Shu-feng and L I Run-guang, " IP Core Design of PLC Microprocessor with Boolean Module", High Density Microsystem Design and Packaging and Component Failure Analysis, 2005 Conference on Digital Object Identifier: IO.l109IHDP.2005.251460 Publication Year: 2005 , Page(s): 1- 5

[11] K.H. John, y M. Tiegelkamp, IEC 61131-3: Programming Industrial Automation Systems, Springer-Verlag, Berlin, 2001.

[12] Xilinx reference manual for Digital Clock Manager http://www.xilinx.com/support/documentation/ip_documentation/dcm_module.pdf