I.J. Engineering and Manufacturing, **2018, 3, 12-22**
Published Online May 2018 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijem.2018.03.02

Available online at http://www.mecs-press.net/ijem

Design and Implementation of control Unit-ALU of 32 Bit Asynchronous Microprocessor based on FPGA

Archana Rani, Dr. Naresh Grover[a,b*]

[a,b] *Faculty of Engineering and Technology, Manav Rachna International University, Faridabad, India*

Received: 29 January 2018; Accepted: 15 March 2018; Published: 08 May 2018

Abstract

In today's fast growing world, the digital design domain has two dominant role factors i.e. the efficiency and speed. The design of asynchronous processor is used to reduce the various challenges faced in synchronous architectures. There are numerous advantages of asynchronous processors, especially in SOC (System on the chip), reducing the crosstalk between analog and digital circuits, easiness in multi-rate circuit integration, reusability of ease of component and at last the less power consumption. The objective of this research paper is to design and simulate control unit of the asynchronous processor by using Xilinx ISE tool in VHDL. A robust control unit has been designed using FPGA. This control unit is responsible for accumulating the whole processor functioning control at a single unit. This paper further presents the optimization techniques for reducing area power and delay constraints related to digital circuits using FPGA.

Index Terms: FPGA, Digital circuits, design optimization.

1. Introduction

As per the prediction by Moore's Law, the semiconductor industry has exhibit the great exponential growth in terms of device performance and complexity. For the verification and implementation of large digital systems, the Very Large Scale Integration (VLSI) is being used as key technology. In just over a decade a signal chip is supporting the designs from tens of thousands of transistors to the millions while maintaining the advantages in performance, reliability and cost reduction of large integrating systems. The complexity of such designs has been accomplished by partitioning them into the sequence of well-defined steps by Computer Aided software tools so that time and quality of final product can improve. These improvements are especially in most complex, lengthy and error –prone phases of the project. Every design process always begins with a

* Corresponding author.
E-mail address: Archana.bhatia.pec@gmail.com, dean.academics@mriu.edu.in

functional description by HDLs (Hardware Description Language) such as VHDL, Verilog HDL etc. Field Programmable gate arrays (FPGAs) are the only programmable logic devices (PLDs) that have shown the exponential growth in the last two decades, indeed, at an even rapid pace while compared to the rest of the semiconductor industry, especially, Application Specific Integrated Circuits (ASICs). That's why the field programmable gate-array (FPGA) technology is a major scope of interest in the algorithmic study and tool development for FPGA specific design automation problems. Thus FPGAs has brought new challenges to logic synthesis, optimization and hence resulting in many new techniques developed in recent years.

2. Design Flow: An Overview

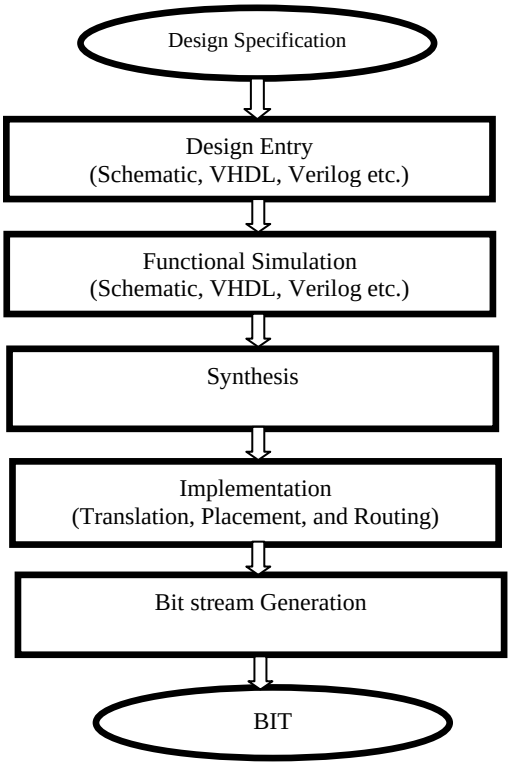

Fig.1. FPGA Implementation Design Cycle

The design process begins with a design specification i.e. functional description of the desired functionality by high-level description language. Several languages have been developed for this purpose (VHDL, Verilog HDL, and Hardware C).

The CAD Tools have been developed for High-Level Synthesis as the first design step, which consists of functional description mapping along with a set of area & Performance constraints in terms of registers & combinational functional Units i.e. called Primitives e.g. ports, adders, comparators, multipliers, shifters. This stage is far differing from the format of data & Control Signals.

Register Transfer Level (RTL) is the output of High-Level Synthesis. This representation is typically divided into control and data path portions. The control unit is responsible for activation of the portions of data path as per the given schedule, so the desired computation can be achieved. The major problem with High-level synthesis is the selection of a scheduling in such a manner that a minimal number of resources of a device are being attracted. The accurate estimation of the figures of merit of a circuit cannot be achieved by the high-level

synthesis. Therefore, a direct mapping of an RTL design into a logic circuit rarely meets area, speed or power requirements. Thus optimization at the logical level is always a needful step.

The FPGA design process is much similar to that for usual technologies such as standard cell or gate array. The design process contains the system-level design (high-level synthesis), the logic-level design (logic synthesis), and the physical-level design (layout synthesis). In the logic-level design for FPGAs consists of combinational logic synthesis, which optimizes the logic gate networks and sequential logic synthesis, which deals with the assignment and/or arrangement of storage elements. On the other hand, system-level synthesis consists of the process that automatically generates high-level behavioral specifications into circuit implementations with the increasing design complexity of ICs.

2.1. FPGA Based design of Control Unit

For the explanation of above design flow, we are going to show the example of a design of a control unit circuit that might be used in our future work i.e. implementation of the asynchronous microprocessor using HDLs and FPGA. In this we are designing an FSM based control unit that will provide the control signals to various other blocks, i.e. to update the Program counter, read/write the data to and fro from register bank, generate the signals for the source or destination location(s) either for register bank(s) or immediate addressing instructions, etc.

The design starts from the specification of our processor i.e. 32-bit asynchronous processor,

Design entry starts with the information of primary input and output signals that will be further going to be used as a physical medium for the communication in the real world, or in general terms, we can say **'wires'.**

```
entity controlUnit is
   Port ( clk: in  STD_LOGIC;
          rst: in  STD_LOGIC;
           RegDst, RegWrite, ALUSrcA,
          MemRead, MemWrite,
          Mult_en, IorD, IRWrite,
          PCWrite, EqNq, ALUsw: out
          std_logic;
instr_31_26, immed_addr: in
std_logic_vector(5 downto 0);
ALUOp, ALUSrcB, PCSource,
ALUmux: OUT std_logic_vector(1
downto 0);
 ALUop_sw,RFmux : out
std_logic_vector(2 downto 0)
                                  );
end controlUnit;
```

Fig.2. Entity Declaration of Control Unit

In this, the name of an entity is the circuit that is going to be designed. The name of this circuit could be anything or totally depends on the user/designer or the nature of application environment. The primary inputs and outputs of our design are:

Table1. Input/Output Signals for Control Unit

Signal	Direction	Description
CLK	In	Clock for internal synchronization
Rst	In	Reset for initialization
Instr_31_26	In	6-bit instruction signal
Immed_addr	In	6-bit immediate address location
Regdst	Out	1-bit register destination
RegWrite	Out	1-bit Register write
ALUSrcA	Out	Control signal for ALU
MemRead/Memwrite	Out	For memory operations
Multen	Out	Multiplier enable/disable
IorD	Out	Immediate or from data memory
IRWrite	Out	Immediate register write
PCWrite	Out	Program counter register update
EqNq	Out	Equal/not equal for jump instruction
ALUSw	Out	ALU unit activation
ALUOp	Out	ALU operation Select
ALUSrcB	Out	B-input for ALU
PCSource	Out	Program Counter Source select
RFMux	Out	signal for register location

Once the entity declaration part has been finished, the next part is the formulation of the flow of the input data to the desired output conditions. This formulation can also be said as the internal description of the circuit to design. The internal circuit can be described by different syntaxes and semantics as per the availability or environment of the EDA tools to be used. As the most EDA tool are common uses a VHDL or Verilog as a circuit design languages. The choosing of HDLs i.e. VHDL or Verilog depends on the interest of the designer and as per demanded by the customer. Although both languages have their own pros and cons, and also the difference between them is far beyond topic to discuss in this present paper. But we are using VHDL as a Hardware description language.

The internal description of the circuit to be designed can be broadly defined by three ways i.e. Dataflow Modeling Style, Behavioral Modeling style, structural modeling style or the mixing of any styles.

Figure 3 are showing the concept of writing the architectural description part of the Circuit to be design. Once the circuit designing has been completed the next step is to logically verify the output whether the circuit is working fine or not if it works fine then we will move go on to next step otherwise we have to recheck our design. For logical verification step, called functional simulation, every EDA tool equipped with their own simulators or third party simulation tools. The most popular simulation tools are ModelSim from Mentor Graphics, ISIM from Xilinx, Altera has simulation part with coordination with Mentor graphics.

```
architecture Dataflow of controlunit is
( Signal Declaration )
Begin
Signal Assignment statements1
:
:
End architecture;
```

```
architecture Behavioral of controlunit is
( Signal Declaration )
Begin
Process( Sensititvity list)
Variable declaration part
begin
Signal Assignment statements1
Variable assignment statement1
:
End process;
End architecture;
```

```
architecture Structural of controlunit is
( Signal Declaration )
Component Declaration part
Begin
Component instantiation 1
Component instantiation 2
:
:
:
End process;
End architecture;
```

Fig.3. Different Modeling style(s) for VHDL

The fig. 4 shows the output result for the Control Unit of 32-Bit Asynchronous Microprocessors. This is the main unit responsible for the whole processing operation starting from Fetching of the instruction to decoding then executing the operation given by the user/designer and then placing the result into the appropriate location such as internal RAM or temporary general purpose register area

These operations have been written in XILINX ISE tool using VHDL and then logically verified by ISIM tool. In figure 4 we can easily see that we had given CLK as a pulse input for 1us (micro second) and the reset

signal had been initiated by providing ‘0’ at the starting time for initializing all the signals into its original phase. After giving the input ‘1' at the reset our control unit will be activated and then be waiting for the instruction from the Program counter and then starts its working from a number of stages provided into the design.

The next step is to check the physical feasibility of the design that is called the synthesis. The synthesis can be done automatically by the EDA tool provider such as XST is the synthesis tool provided by the Xilinx. The major role of synthesizing the design is to generate the gate level netlist of the circuit to be designed as per the specified technology i.e FPGAs and CPLDs. It also tells the requirement or usage of the number of the gate(s) or logical resources of the specified device.

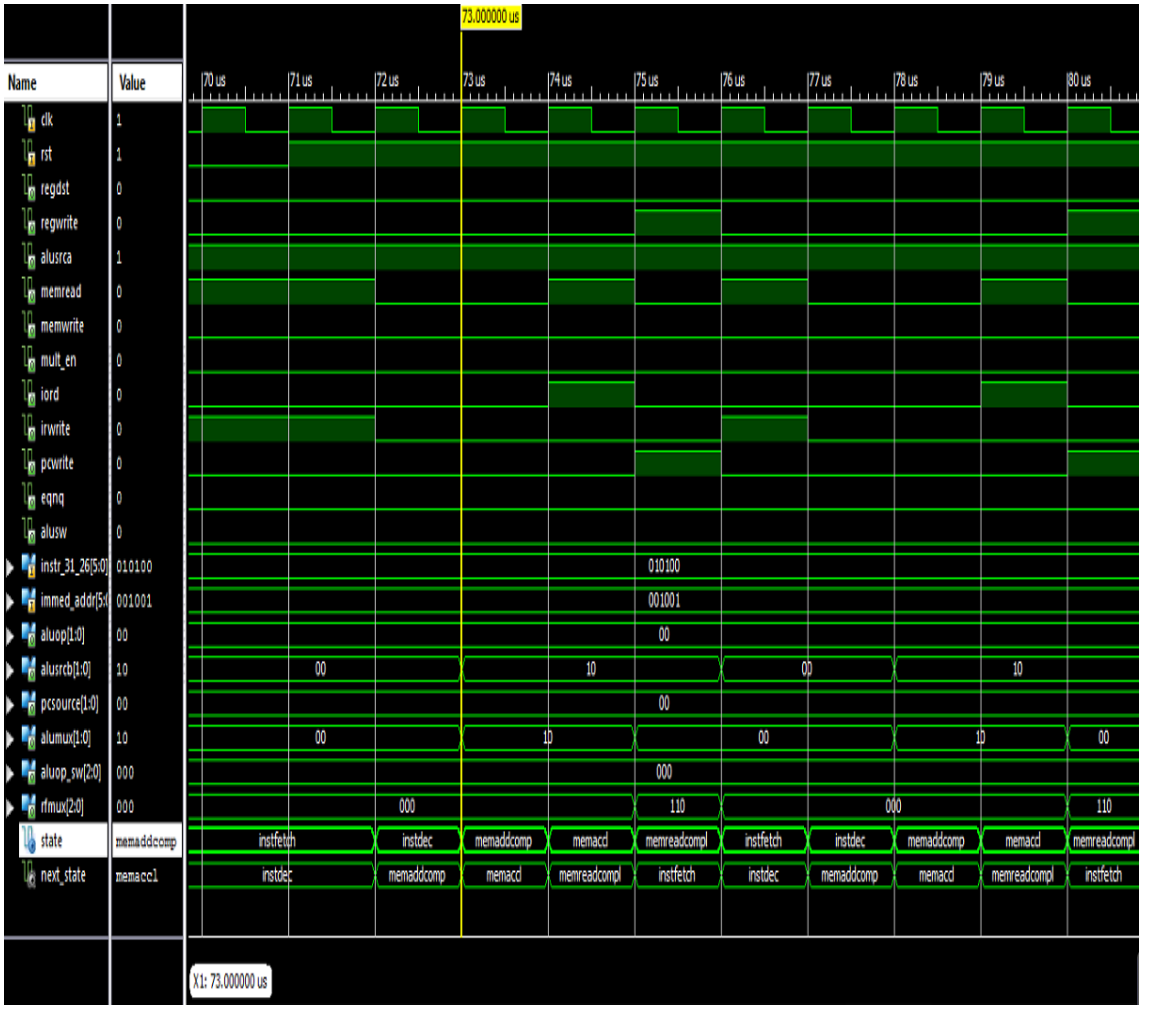

Fig.4. ISIM for Functional Simulation

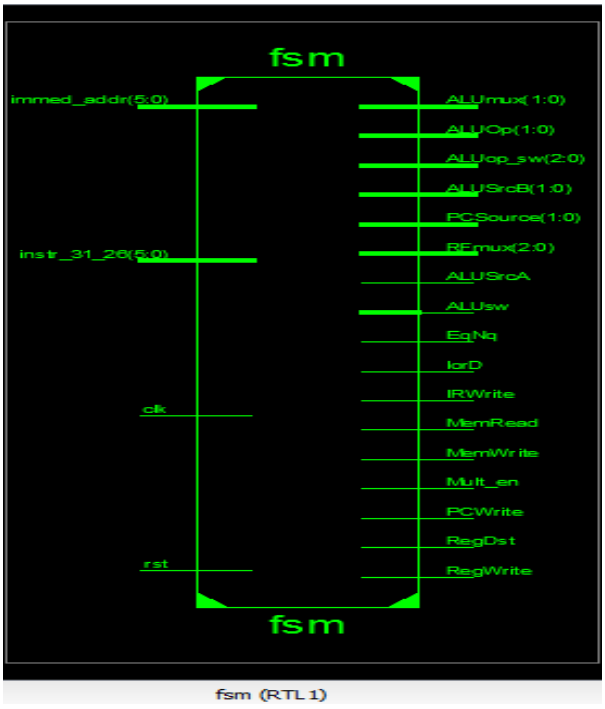

Fig.5. RTL view after Synthesis

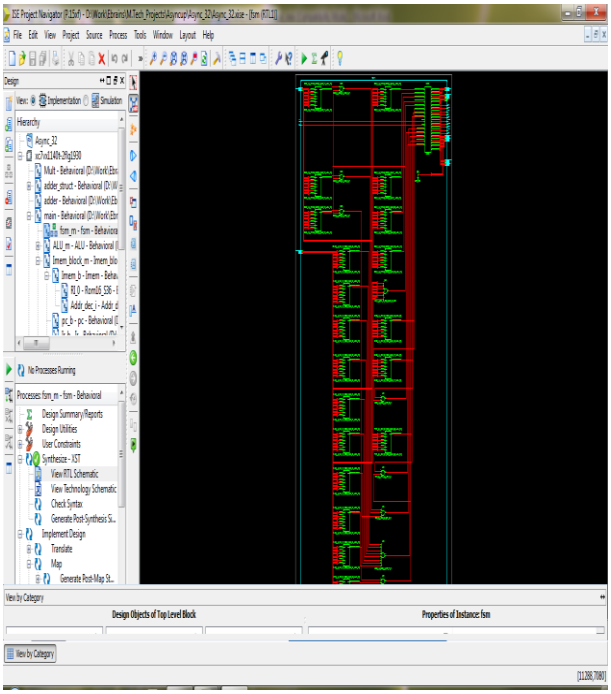

Fig.6. Internal structure of the Control Unit

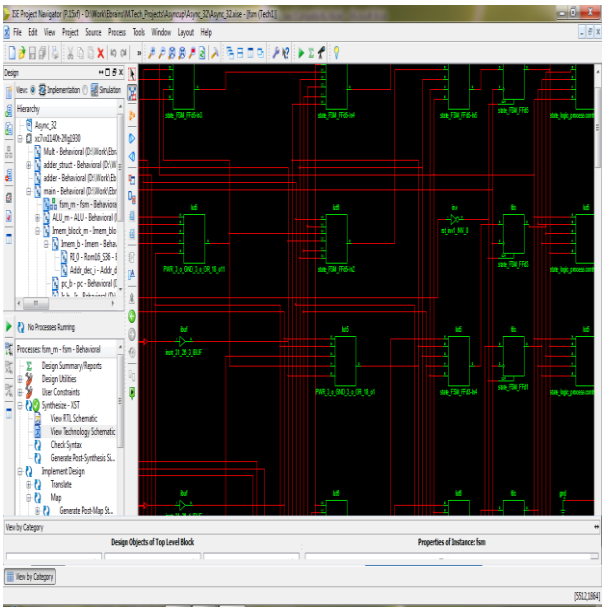

Fig.7. Technological View (i.e. In terms of LUTs) of the Control Unit(Internal structural

Synthesis results are shown in figure 5, 6 and 7 respectively. The figure 5 shows the only front/top view of the circuit to be designed. While the figure 6 and 7 are showing the internal of the circuit in terms of general logic gates and FFs and in terms of specified technology device i.e in terms of LUTs in the case of an FPGA and in terms of AND or OR gate arrays for CPLDs.

Once the synthesis has been done, our next goal is to take our design to the physical working model. To accomplish this task we have to go through the Implementation process. This process is divided into Translate, Map, Place & Route steps. For implementation part also most EDA tools are facilitating the automatic design flow. The translate takes all the netlist and NGC files till now generated with internal core generator & synthesis and then turns it into one big design file. While mapping takes all the generic logic gates and flip flops described in the design files and turns them into the resources available in the targeted FPGA (i.e. slices and IOB's). The place and route tool are responsible for the placing of the resources inside the die and the connection wire to bind them together in order to meet the design criteria. At last, the bit file describes the final configuration and connection of the FPGAs through the system and then downloading the bit file onto FPGA.

3. Optimal FPGA Mapping

FPGA designs are progressively used in distinct areas. These areas can be instantly modelling of new products which may include fast ASICs implementation, or to logic testing and to fabricating a small number of a device. And at last to facilitate that if a device should be re-configurable while in use. The Programmable Logic Blocks (PLBs)/ Combination Logic Block (CLBs), the routing switch matrix and IOBs (Input-Output Blocks) are three programmable elements in an FPGA. The PLBs are collection of combinational components such as simple gates (e.g., OR & AND), multiplexers (MUXs), programmable lookup tables (LUTs), and sequential components such as flip-flops. Whereas, the routing resources include sections of interconnects and switching blocks. These sectioned interconnects are responsible for the connection in between the inputs and outputs blocks where the switching blocks are responsible for the link of these segments to form long routing tracks to implement routing topology. The I/O blocks are the last programmable element of FPGAs, which can be programmed to create the primary inputs or primary outputs of the circuits. The FPGAs available today are

based on Multiple to single output lookup tables. There are distinct technologies mapping algorithms available that produce a mapping solution for k-LUTs, and then pack these LUTs into Programmable Logic Blocks. Whereas the placement and routing are robustly connected with the defined architecture internal to the chip and majorly governed by the commercial FPGA software or EDA tools, thus the optimization and mapping can be more affected by the user/designer.

3.1. Logic Optimization

There are various standards of levels to differentiate the basic design techniques used for FPGA based designs. The first approach could be recognized as a logic optimization, which deals with the transformation of the gate level network into more optimized gate-level network for next stages in terms of fitting area. Another approach could be defined as a technology mapping. In this the design has to re-target into a network of cells, as per the specified technology by overlaying the previous network. Thus in Logic design optimization these techniques can be used as an alternative to the low-level design optimization approaches.

The first approach i.e. the logic optimization is technology independent This operates on the knowledge and functionality of gates and works only for the combinational circuit by only Boolean optimization. Whereas the second approach, following only structural information of the network as per the FPGA technology and always uses combinational optimization technologies, known as technology mapping.

On the other hand for the sequential circuit in digital design the optimization could be done by circuit partitioning, retiming for computation of new equivalent states analogous to initial states. Thus it is very essential to mention that both approaches are having difference in their techniques and functional properties. Although there are many optimization algorithms and systems tools available that uses both approaches in a combination.

3.2. Optimization Targets

There are various optimization goals for FPGAs while implementing a digital circuit. Some of most used are as follows:

To optimize the delay from primary inputs to primary outputs could be known as the delay optimization. The delay of LUTs is depended on the longest path length between the basic input and basic output. The total length of a given path is calculated by the addition of delays of nodes and edges in the provided path. The minimum chip area utilization by the implemented circuit, known as the area optimization. The area of a LUT network depends on the number of LUTs or any actual logic elements used in a specific/targeted FPGA internal architecture. To reduce the power consumption of the digital circuit implementation considered as power minimization objectives. The calculation of Power consumption can be based on the transition frequency of LUT and its basic inputs and output load variance. While the load variance changes aggressively along with the mapping process.The exact calculation of these objectives can only be obtained after layout synthesis. Hence a proper balance is required among these three objectives.

4. Conclusion

This paper presents control unit of 32-bit asynchronous processor design, its implementation by using ISE simulation tool. Next working of the control unit is shown by simulation window: following by processor instruction set-fetch, decode, execution, memory access, Register type, I-type, load/store type instruction flow. Further, this paper deals with control unit design of processor that shows the controlling of signals for different units in processor design, at the end, results have been shown using implementation windows. Thus whole paper presents the largest and complex part of our research topic and techniques available for circuit optimization.

References

[1] I. Kuon and J. Rose, ―Measuring the Gap Between FPGAs and ASICs,‖ ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 21-30, 2006.

[2] Shackleford, B., Okushi, E., Yasuda, M., Koizumi, H., Seo, K.,Iwamoto, T., and Yasuura, H., "High-performance hardware design and implementation of genetic algorithms", in Teodorescu et al, *Hardware implementation of Intelligent Systems*, pp. 53 - 87, 2001.

[3] Dick R.P., Jha N.K., *MOGAC: A Multiobjective Genetic Algorithm for Hardware-Software Cosynthesis of Distributed Embedded Systems*, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 17, no. 10, pp. 920-935, Oct. 1998.

[4] Jamro E., Wiatr K., *Genetic Programming in FPGA Implementation of Addition as a Part of the Convolution*, Proc. Of the IEEE Int. Conf. Digital System Design, Warszawa, Poland, IEEE Computer Society Press, 4-6 Sep. 2001.

[5] Hassan, R., and Crossley, W., "Variable Population-Based Sampling for Probabilistic Design Optimization with a Genetic Algorithm," AIAA-2004-0452, 42nd Aerospace Sciences Meeting, Reno, NV, January 2004.

[6] Hassan, R., Genetic Algorithm Approaches for Conceptual Design of Spacecraft Systems Including Multi Objective Optimization and Design under Uncertainty, doctoral thesis, Purdue University, May 2004.

[7] G. Chen and J. Cong, "Simultaneous logic decomposition with technology mapping in FPGA designs", in the Proceedings of the 2001 ACM/SIGDA ninth International Symposium on Field Programmable Gate Arrays, 2001, pp. 48-55.

[8] J. Cong and Y. Ding, "Flow Map: An Optimal Technology mapping algorithm for delay optimization in lookup-table based FPGA designs", in IEEE Transactions on Computer-Aided Design and Integrated Circuits and Systems, **Vol. 13**, No. 1, January 1994, pp. 1-12.

[9] J. Cong and C. Wu, "An Efficient Algorithm for Performance-Optimal FPGA Technology Mapping with Retiming", in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **Vol. 17**, No. 9, September 1998, pp. 738-748.

[10] J. Zhang, Z. Zhang, S. Zhou, M. Tan, X. Liu, X. Cheng, and J.Cong, "Bit-level optimization for high-level synthesis and FPGA-based acceleration," in *Proc. FPGA*, Feb. 2010, pp. 59–68.

Authors' Profiles

Ms. Archana Rani Bhatia is a Ph.D. Scholar from Manav Rachna International University, Faridabad. Had completed Post Graduation in Electronics Product Design and Technology from Punjab Engineering College Chandigarh in 2008 and did Graduation in Electronics and Communication Engineering, with sound working experience over 6.6 years in various electronic works, related to Teaching, Research & Industry side.

Prof. (Dr.) Naresh Grover did his B.Sc (Engg.) in 1984 and M.Tech in Electronics and Communication Engineering in 1998 from REC Kurukshetra (Now NIT Kurukshetra). He has a rich experience of 33 years in academics. He has authored two books on Microprocessors and is a co-author of a book on Electronic Components and Materials. His core area of interest is Microprocessors and Digital System Design. Presently he is Dean-Academics at Manav Rachna International University, Faridabad.

How to cite this paper: Archana Rani, Naresh Grover,"Design and Implementation of control Unit-ALU of 32 Bit Asynchronous Microprocessor based on FPGA", International Journal of Engineering and Manufacturing(IJEM), Vol.8, No.3, pp.12-22, 2018.DOI: 10.5815/ijem.2018.03.02