FPGA IMPLEMENTATION OF A FUNCTIONAL MICROCONTROLLER

Pranoy T.M[1], Nitya Mary Kurian[2], Rizwana Parveen K.A[3], Radhika V Nambiar[4], Neethu George[5]

George M Jacob[6]

[1, 2, 3, 4]B.Tech Student, [5]Assistant Professor, [6]Lab Instructor, Electronics & Communication Engineering, TocH Institute of Science and Technology, Kerala, India

Abstract

In any control and controller system applications, microcontroller is an important module, which provides the control, timing and status signals. A microprocessor is usually defined as "a single chip that contains control logic and data processing logic, so that it can execute instructions listed in a program to operate on some data". Microcontrollers are nothing but microprocessors with on-chip memory. This paper primarily deals with the implementation of an 8 bit microcontroller to perform arithmetic and logical operations. The modules are programmed using Very High Speed IC Hardware Description Language (VHDL). Using top-down approach, the elements of the microcontroller are identified as basic registers, instruction register, program counter, ALU, ROM, Control and timing Unit. All these modules are connected together to form a top module which is controlled by a Finite State Machine (FSM). The Finite State Machine enters each of the predefined states based on the inputs. FSM implementation also aids in changing the functionalities of the system without completely redesigning the system. Finally, in the top level module, these blocks are connected to form a functional microcontroller. In this paper, we have simulated and synthesized the various parameters of microcontroller by using VHDL on Xilinx ISE 8.2i and SPARTAN 3 FPGA board.

Keywords: *FPGA, Microcontroller, Microprocessor and VHDL*

----------***----------

1. INTRODUCTION

A microcontroller is a small and low-cost computer built for the purpose of dealing with specific tasks, such as displaying information in a microwave LED or receiving information from a television's remote control. Microcontrollers are mainly used in products that require a degree of control to be exerted by the user.

Whether using ASIC, FPGA or CPLD based realizations, it is essential to incorporate the microcontroller module, as an integral part of the system. Functional microcontroller has been developed using VHDL coding using structural design of logic blocks which generates control and timing signals used for the data processing operation.

Present day VLSI technology has lead to the design and development of millions of gates on a chip. Hardware designers create several VLSI modules for their research and development purposes. It is often important to re-use these modules to reduce product development time, thereby minimizing the time to market. Therefore, it is important to design hardware in a modular fashion, so that these modules can be included in the development of a complex system. The design and development of such a modular design microcontroller helps other designers to incorporate this module with minimal or no modifications to the hardware module [1].

2. LITERATURE SURVEY

Although many innovative methodologies have been devised in the past, to handle more complex control problems and to achieve better performances, the great majority are still controlled by means of simple microcontrollers. When compared to von Neumann processor architectures, the Harvard architecture improves the bus bandwidth as in von Neumann architectures both program and data memory are being accessed through a shared bus.

Majority of previous works done have used VHDL to describe all the modules in the design which is a very useful tool with its degree of concurrency to cope with the parallelism of digital hardware. The VHDL software reduces the complexity and also provides a graphic presentation of the system. The key advantage of VHDL when used for systems design is that it allows the behaviour of the required system to be described (modelled) and verified (simulated) before synthesis tools translate the design into real hardware (gates and wires). This software not only compiles the given VHDL code but also produces waveform results. [2]

In designing a CPU, first its instruction set needs to be defined, and how the instructions are encoded and executed. Also the number of instructions, what the instructions are, what operation code (opcode) need to be assigned to each of the instructions, how many bits need to be used to encode an instruction etc have to be addressed. Once the instruction set is

decided upon, designing a datapath that can execute all the instructions in the instruction set is done. In this step, a custom datapath is created, so what the functional units are, how many registers need to be used, whether single register file or separate registers, how are the different units connected together etc are decided. [6]

The microcontroller consists of control and data processing parts. The control part is composed of program counter, program memory and instruction register. The Arithmetical-Logic unit (ALU) is the core of data processing part. Further parts are input multiplexer, Accumulator register and register file. [5]

The system development starts with top-down planning approach and the blocks are designed using bottom-up implementation. The programs are written and simulated using Electronic Data Automation (EDA) tool like ModelSim. [2]The main advantage of using a state machine in embedded design consists in its flexibility to add, delete or change the flow of the program without impacting the overall system code structure.

Increasing performance and gate capacity of recent FPGA devices permits complex logic systems to be implemented on a single programmable device. Such a growing complexity demands design approaches, which can cope with designs containing hundreds of thousands of logic gates, memories, high-speed interfaces, and other high-performance components.[4]

3. SYSTEM ARCHITECTURE

When compared to Von Neumann processor architectures, the Harvard architecture improves the bus bandwidth as in Von Neumann architectures both program and data memory is being accessed through a shared bus. Thus the architecture implemented is Harvard architecture.

3.1 Block Diagram

The RISC processor core provides an 8-bit ALU. The ALU receives its input from two eight bit registers namely the Accumulator (Register A) and Register B. If there is only one operand then that operand will be the Accumulator. The ALU supports simple arithmetic operations like addition, subtraction, increment and decrement; boolean logic operations like AND and OR; data transfer instructions and branching instructions.

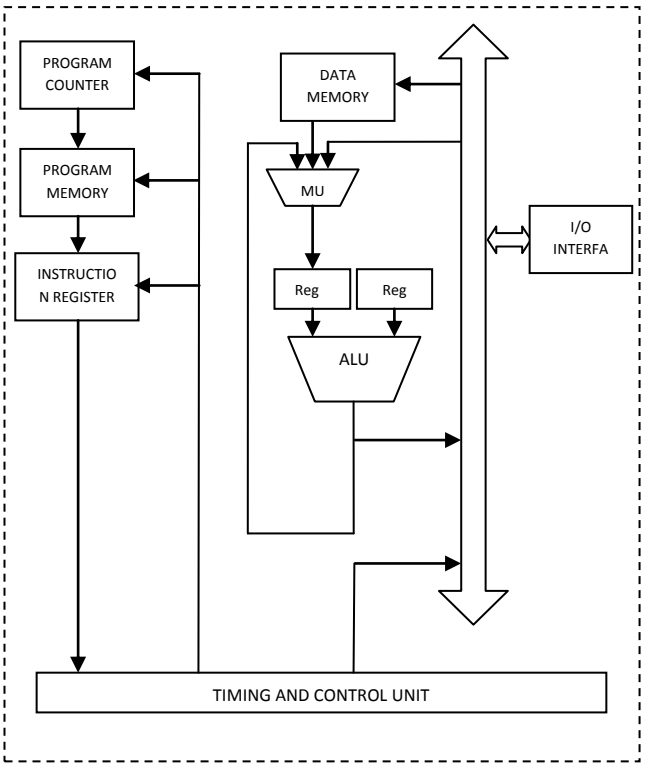

Fig -1: General block Diagram

4. MODULE DEVELOPMENT

4.1 Arithmetic Logic Unit

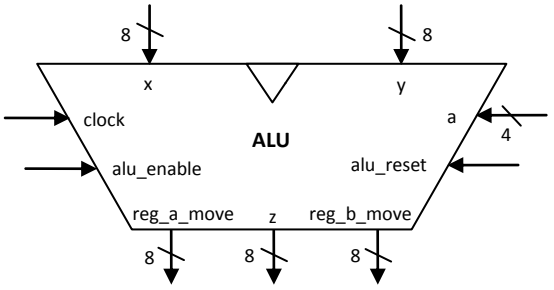

Fig -2: ALU

It is a multi operational combinational logic circuit which performs arithmetic and logical operations like ANDing, ORing, ADDITION, SUBTRACTION, etc. The word length of ALU depends upon internal data bus. It is 8 bit and is always controlled by timing and control circuits.

The inputs to the ALU are ‘x’ and ‘y’, which are 8 bit data and a 4 bit control input ‘a’. Apart from these there are clock, alu_reset and alu_enable inputs. ALU is positive edge triggered. ‘z’ is the 8 bit output from ALU. reg_a_move and reg_b_move are two outputs used for move instructions and

also for storing data to accumulator after an instruction is executed.

4.2 Accumulator

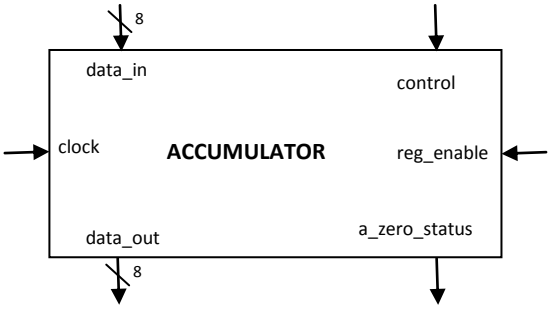

Fig -3: Accumulator

Accumulator is an 8 bit register. It is positive edge triggered. Inputs to the register are control, data_in and reg_enable. Control input is used to decide whether data is to be read from or written to the register. The output of register A 'data out' is given as input to the ALU. 'a_zero_status' is a flag that reflects the status of accumulator. If content of the accumulator is zero, the flag is set.

4.3 Program Counter

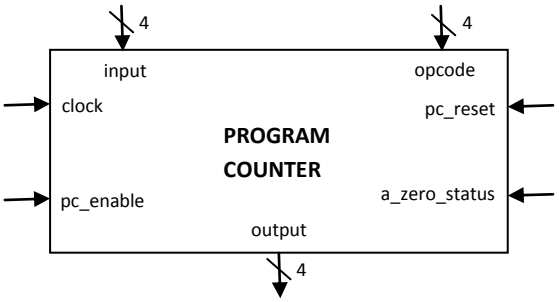

Fig -4: Program Counter

Program counter is a special purpose register which stores the address of the next instruction to be executed. Microcontroller increments the program counter whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction to be executed. The inputs to the program counter are input(4 bit), opcode(4 bit), pc_enable, pc_reset and a_zero_status. Output of program counter is given as the 4 bit address to ROM.

4.4 Program Memory

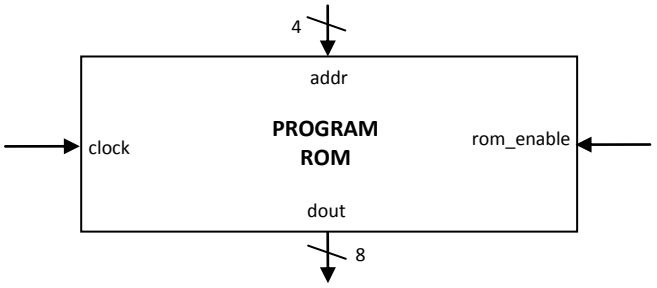

Fig -5: Program ROM

The program is stored inside a 16*8 ROM. It is positive edge triggered. The inputs to the ROM are addr and rom_enable. addr is a 4 bit address. Opcode is stored in the memory location given by this addr. When rom_enable is high, this opcode will be available at the output pin dout.

4.5 Instruction Register

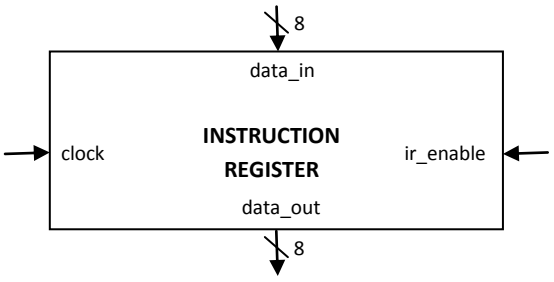

Fig -6: Instruction Register

Instruction register is an 8-bit register just like every other register of microcontroller. The instruction may be anything like adding two data, moving data etc. When such an instruction is fetched from memory, it is directed to instruction register. So the instruction registers are specifically to store the instructions that are fetched from memory.

4.6 Timing and Control Unit

Timing and control unit is a very important unit as it synchronizes the registers and flow of data through various registers and other units. This unit consists of an oscillator and controller sequencer which sends control signals needed for internal and external control of data and other units.

4.7 I/O Interface

An I/O interface is required whenever the I/O device is driven by the processor. The interface must have necessary logic to interpret the device address generated by the processor. If different data formats are being exchanged, the interface must be able to convert serial data to parallel form and vice-versa.

5. STATE DIAGRAM

The START state serves as the initial reset state. Program counter incrementing is performed in this state. All instructions are made to go back to the START state.

From the START state, the control unit goes to the FETCH state unconditionally. In the FETCH state, the program ROM is enabled and the opcode is fetched.

From the FETCH state, the control unit goes to the DECODE state unconditionally. In this state IR is enabled and the output of program ROM is stored in IR. The DECODE state tests the four most significant bits of the IR, IR7-4, and goes to the corresponding state as encoded by the four bit opcode for executing the instruction.

The ADD, SUB, OR & AND instructions respectively adds, subtracts, ORs and ANDs the content of register A with the content in register B, and stores the result back into Accumulator.

There are direct and register MOV operations. In direct addressing, the content of the specified address is moved to the accumulator. In register addressing, the contents of the source register is moved to the destination register (either A to B or vice versa).

The IN instruction inputs a value and stores it into A. The IN state waits for the Enter key signal before looping back to the START state. In doing so, several values can be read in correctly by having multiple input statements in the program. Notice that after the Enter signal is asserted, there is a zero state that waits for the Enter signal to be de-asserted, i.e. for the Enter key to be released.

The OUT instruction copies the content of the accumulator to the output port.

The JMP instruction loads the PC with the specified address of the IR. The JZ instruction loads the PC with the specified address if A is zero. Loading the PC with a new address simply causes the CPU to jump to this new memory location. The JNZ (Jump Not Zero) instruction tests to see if the value in A is equal to 0 or not. If A is equal to 0, then nothing is done. If A is not equal to 0, then the last four bits of the instruction, designated as addr in the encoding, is loaded into the PC. The four bits, addr, represent a memory address. When this value is loaded into the PC, we are essentially performing a jump to this new memory address, since the value stored in the PC is the location for the next fetch operation.

NOP instruction performs no operation.

The INC and DEC instructions increment and decrement the content of A by 1 respectively, and store the result back into Accumulator.

Once the FSM enters the HALT state, it unconditionally loops back to the HALT state, giving the impression that the CPU has halted.

Table -1: Instructions and Opcodes

Instruction	Opcode
ADD A,B	0000xxxx
SUB A,B	0001xxxx
MOV A,B	0010xxxx
AND A,B	0011xxxx
OR A,B	0100xxxx
MOV B,A	0101xxxx
IN A	0110xxxx
OUT A	0111xxxx
JMP addr	1000 addr
JNZ addr	1001 addr
JZ addr	1010 addr
NOP	1011xxxx
INC A	1100xxxx
DEC A	1101xxxx
MOV A, #addr	1110 addr
HALT	1111xxxx

6. SIMULATION RESULTS

6.1 Simulation Result of Addition in Datapath

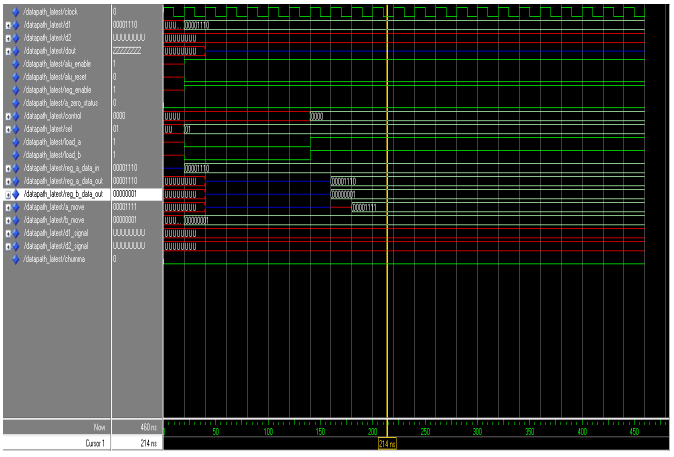

Fig -7: Simulation result of addition in Datapath

ADD A, B

Figure shows the simulation of ADD instruction in the datapath. When an add instruction is encountered in the program the datapath behaves as shown above.

Here two eight bit numbers at alu input pins reg_a_data_in (00001110) and reg_b_data_in (00000001) are added together and the result (00001111) appears in the output pin a_move.

6.2 Simulation Result for Addition

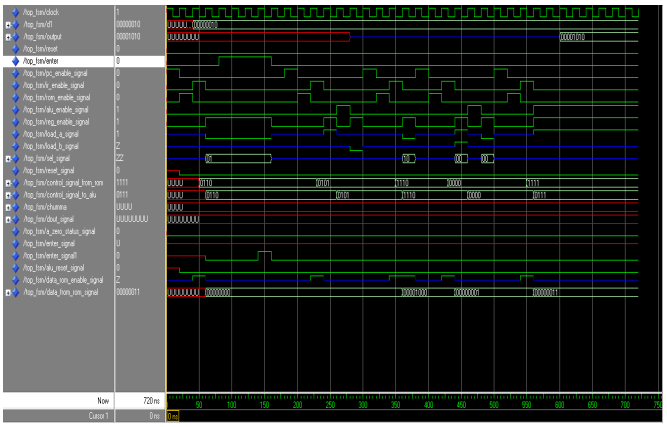

Fig -8: Simulation result for addition

IN A
MOV B,A
MOV A, #1000
ADD A,B
HALT

Figure shows the simulation of the program given above, where two eight bit numbers 00000010(external input) and 00001000(stored in data memory at address #1000) are added to get the output 00001010 at the output pin.

6.3 Simulation Result for Subtraction

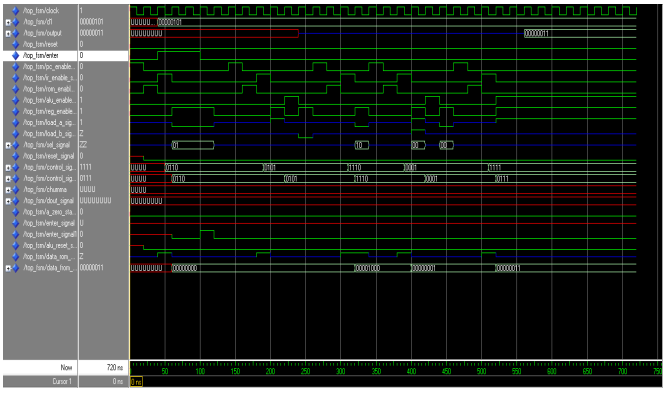

Fig -9: Simulation result for subtraction

IN A
MOV B,A
MOV A, #1000
SUB A,B
HALT

Figure shows the simulation of the program given above, where two eight bit numbers 00001000(stored in data memory at address #1000) and 00000101(external input) are subtracted to get the output 00000011 at the output pin.

6.4 Simulation Result for Decrementing until Zero

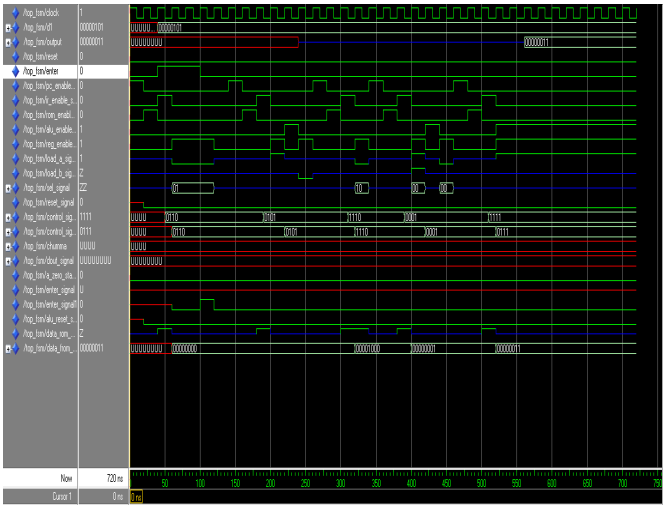

Fig -10: Simulation result for decrementing until zero

IN A
DEC A
OUT A
JNZ 0001
HALT

Figure shows the simulation of the program given above, where DEC instruction is followed by a JNZ instruction. Here an external eight bit number (00000011) is decremented until it becomes zero.

7. SYNTHESIS RESULTS

7.1 Design Summary

Table -2: Device utilization summary

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	125	1920	6%

Number of Slice Flip Flops	161	3840	4%
Number of 4 input LUTs	234	3840	6%
Number of bonded IOBs	19	173	10%
Number of GCLKs	2	8	25%

7.2 RTL Schematic Diagrams

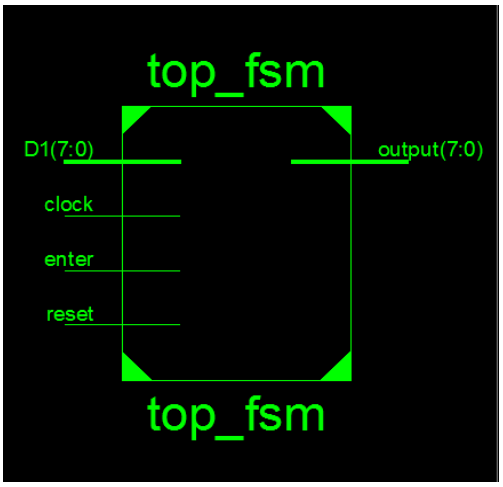

Fig -11: RTL Schematic of top module

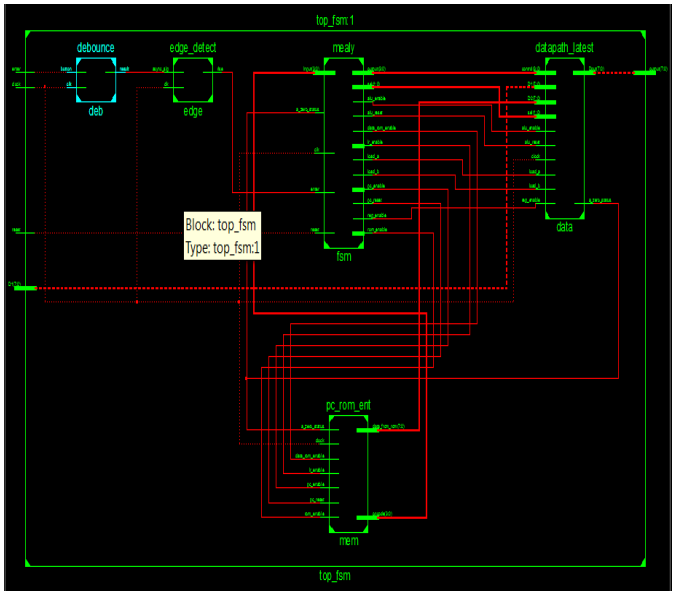

Fig -12: Expanded RTL Schematic of top module

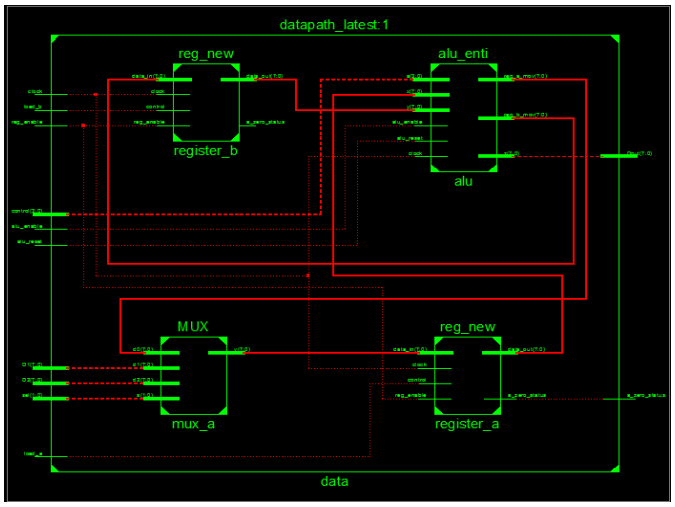

Fig -13: RTL Schematic of datapath

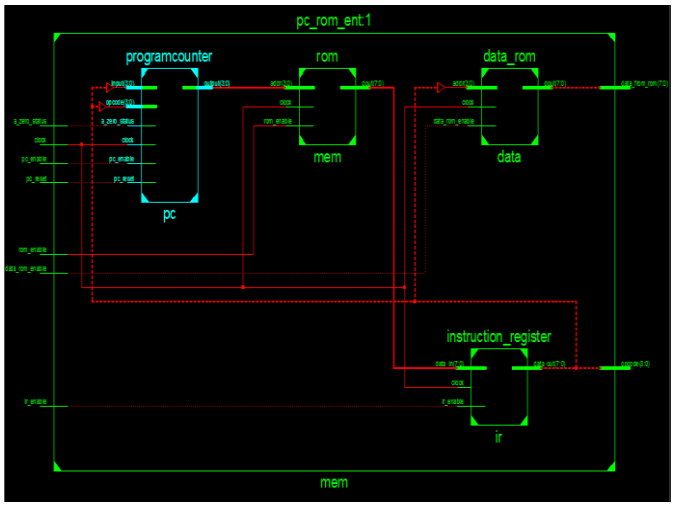

Fig -14: RTL Schematic of program path

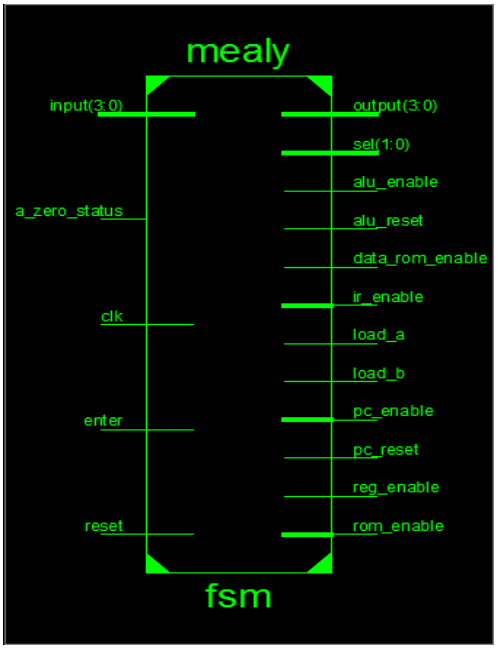

Fig -15: RTL Schematic of control unit

8. CONCLUSIONS

In this paper, the design and the development of a basic 8-bit microcontroller has been discussed. The developed microcontroller module functions with simple control signals. The developed module is functionally built in VHDL. Due to the modular design and bottom-up implementation of this microcontroller, the VHDL code can easily be expanded to develop higher order microcontroller without making extensive changes. The design was implemented by using Xilinx Synthesis tool choosing Spartan 3 as the FPGA target device.

REFERENCES

[1] Vivekananda Jayaram, Subbarao Wunnava , "Functional Microcontroller Design and Implementation," *LACCET'2006* , Mayagüez, Puerto Rico ,June 2006

[2] Suchita Kamble,N. N. Mhala, "VHDL Implementation of 8-Bit ALU," *IOSR Journal of Electronics and Communication Engineering (IOSRJECE) ISSN : 2278-2834* Vol 1, Issue 1 (May-June 2012)

[3] DulÍk M. Vasilko Durakov P. Fuchs," Design of a RISC Microcontroller Core in 48 Hours", 1Microelectronics Systems Research Group School of Design, Engineering & Computing, Bournemouth University Fern Barrow, Poole, Dorset BH12 5BB United Kingdom

[4] M. Kovac," Asynchronous Microcontroller Simulation Model in VHDL," World Academy of Science, Engineering and Technology 21 2008

[5] Enoch O. Hwang, "General-Purpose Microprocessors," in Digital Logic and Microprocessor Design With VHDL of His Published Book, 1st Indian reprint ed. Thomson Learning

BIOGRAPHIES

Pranoy T.M is presently a student of TocH Institute of Science and Technology, Arakkunnam, Kochi. He is pursuing his BTech degree in Electronics and Communication Engineering from CUSAT.

Nitya Mary Kurian is presently a student of TocH Institute of Science and Technology, Arakkunnam, Kochi. She is pursuing her BTech degree in Electronics and Communication Engineering from CUSAT.

Rizwana Parveen K.A is presently a student of TocH Institute of Science and Technology, Arakkunnam, Kochi. She is pursuing her BTech degree in Electronics and Communication Engineering from CUSAT.

Radhika V Nambiar is presently a student of TocH Institute of Science and Technology, Arakkunnam, Kochi. She is pursuing her BTech degree in Electronics and Communication Engineering from CUSAT.

Neethu George is presently working as Assistant Professor in the department of Electronics and Communication Engineering, TocH Institute of Science and Technology, Arakkunnam, Kochi .She received her MTech degree in VLSI & Embedded Systems from CUSAT.

George M Jacob is presently working as Lab Instructor in the department of Electronics and Communication Engineering, TocH Institute of Science and Technology, Arakkunnam, Kochi. He received his engineering diploma in Computer Science from the Board of Technical Education.