RESEARCH ARTICLE OPEN ACCESS

FPGA Based Implementation of Pipelined 32-bit RISC Processor with Floating Point Unit

Jinde Vijay Kumar[1], Chintakunta Swapna[2], Boya Nagaraju[3] , Thogata Ramanjappa[4]
1,2Research Scholar, Department of Physics, S.K.University, Anantapur, A.P.
3Assistant Professor, Department of Physics, INTELL Engg. College, Anantapur, A.P.
[4] Professor, Dean, Faculty of Physical Sciences, Department of Physics, S.K.University, Anantapur, A.P.

ABSTRACT
This paper presents 32-bit RISC processor with floating point unit to be designed using pipelined architecture; through this we can improve the speed of the operation as well as overall performance. This processor is developed especially for Arithmetic operations of both fixed and floating point numbers, branch and logical functions. The proposed architecture is able to prevent pipelining from flushing when branch instruction occurs and able to provide halt support. Floating point operations are widely used these days for many applications ranging from graphics application to medical imaging. Thus, the processor can be used for diversified application area. The necessary code is written in the hardware description language Verilog HDL. Quartus II 10.1 suite is used for software development; Modelsim is used for simulations and then implementation on Altera DE 2 FPGA board.
Keywords **-** FPGA, RISC and Altera DE2 board

I. INTRODUCTION

Now-a-days, computer and mobile phones are indispensable tools for most of everyday activities. This places an increasing burden on the embedded microprocessor to provide high performance while retaining low power consumption and small die size, which increases the complexity of the device. However, as products grow in complexity, more processing power is required while the expectation on battery life also increases.

The trend in the recent past shows the RISC processors clearly outsmarting the earlier CISC processor architecture. RISC is a type of microprocessor that has a relatively limited number of instructions. Though it may seem less effective for a computational task to be executed with many simple instructions rather than a few complex instructions, the simple instructions take fairly the same amount of time to be performed, making them ideal for pipelining. It is designed to perform a smaller number of types of computer instructions so that it can operate at a higher speed (perform more million instructions per second, or millions of instructions per second). Earlier, computers used only 20% of the instructions, making the other 80% unnecessary. One advantage of reduced instruction set computers is that they can execute their instructions very fast because the instructions are so simple.

This paper presents a very simple 32-bit data width general purpose 4 stage pipelined processor with floating point unit on FPGA. It has a complete instruction set, program and data memories, general purpose registers and a simple Arithmetic & Logical Unit (ALU) with single precision floating point arithmetic operations. In this design most instructions are of uniform length and similar structure, arithmetic operations are performed and the resultant value is stored in the memory/registers and retrieved back from memory when required. In this paper an efficient FPGA implementation of 32-bit single precision floating point unit which performs addition, subtraction, multiplication and division.

II. ARCHITECTURE OF 32-bit RISC PROCESSOR WITH FPU

Fig. 1 shows the proposed processor of 32-bit pipelined RISC processor with Floating Point Unit. The processor design is based on the RISC instruction set which is characterised by 32-bit architecture having four 32-bit registers. The RISC is designed using the Hardware Description Language Verilog HDL. Machine instructions were implemented directly in hardware.

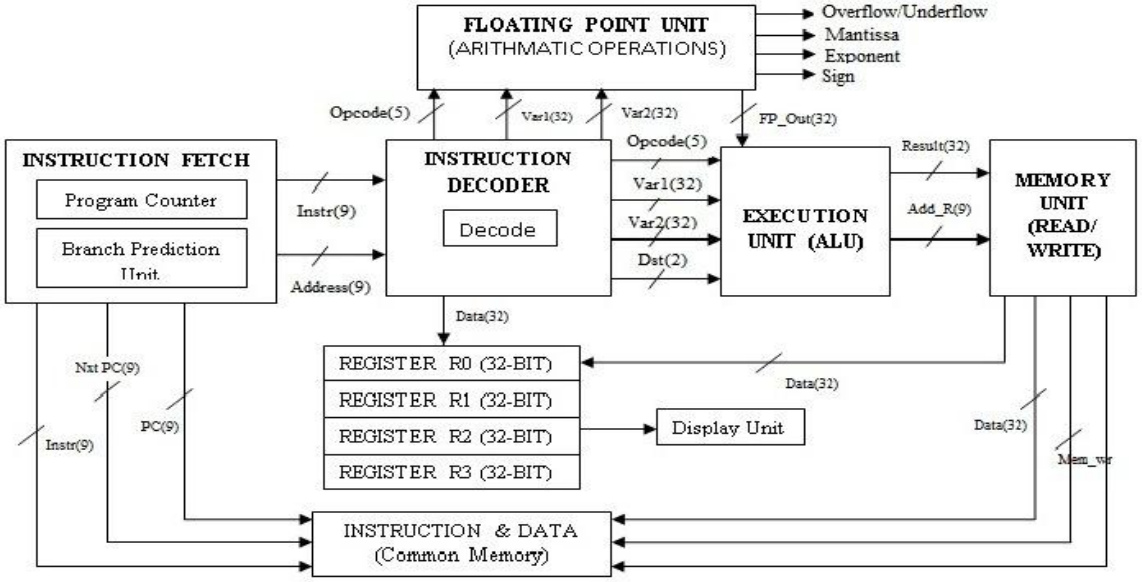

Fig.1 Architecture of 32-bit RISC Processor with FPU

The architecture of the pipelined 32-bit RISC processor consists of instruction fetch, branch prediction, instruction decode, execute, memory read/write back, instruction set and floating point unit. Pipelining technique allows for simulations execution of parts or stages of instructions more efficiently. With a RISC processor, one instruction is executed while the following instruction is being fetched. By overlapping these operations, the CPU executes one instruction per clock cycle, even though each instruction requires three cycles to be fetched, decoded, and executed.

The pipeline stages for different type of instructions are processed as follows, in the fetch stage; instructions are fetched at every cycle from the instruction memory whose address is pointed by the program counter (PC). During the decode stage, the registers are read from the register file and the opcode is passed to the control unit which asserts the required control signals. Sign extension is also done for the calculation of effective address. In the execute stage, for register type instruction, the ALU operation and also floating point arithmetic operations are performed according to the ALU operations control signals and for load and store instructions, effective address calculation is done. The load and store instructions write to and read from the data memory in the memory stage while the ALU results and the data read from the data memory are written in to the register file by the register type and load instructions respectively in the write-back stage.

There are basically three types of instructions namely Arithmetic & Logical Instructions (ALU) with floating point unit instructions, Load/Store instructions and Branch Prediction instructions.

1. ALU instructions with Floating point unit:

The ALU is responsible for all arithmetic and logic operations that take place within the processor. These operations can have one operand or two, with these values coming from either the register file or from the immediate value from the instruction directly. The operations supported by the ALU include add, subtract, compare, and, or, not, increment, decrement, nand, nor and xor. The output of the ALU goes either to the data memory or through a multiplexer back to the register file.

All the arithmetic operations are performed like addition, subtraction, multiplication and division instructions are implemented on Single Precision Floating Point Unit.

2. Load/Store Instructions:

Usually take a register as an operand and a 16-bit immediate value. If the instruction is a load, memory does a read using the effective address. If it is a store, then the memory writes the data from the second register read from the register file using the effective address. The purpose of store unit is store the result into corresponding register or memory.

3. Branch Prediction Instructions:

A branch prediction causes an immediate value to be added to the current program counter. Some branch

instructions are BZ (Branch Zero), BRZ (Branch Register Zero) and BRC (Branch Register Carry).

III. MODULES DESIGN OF RISC

This section presents the design of different modules like instruction fetch, instruction decode, register file, execute, floating point unit, memory read/write back and instruction set along with four general purpose registers namely Register0, Register1, Register2 and Register3.

3.1 Instruction Fetch (IF):

The instruction pointed to by the PC is fetched from memory into the instruction register of the CPU, and the PC is incremented to point to the next instruction in the memory. Normally, the PC is incremented by one, during each clock cycle unless a branch instruction is executed. When a branch instruction is encountered, the PC is incremented by the amount indicated by the branch offset. The block diagram of Instruction Fetch Unit is shown in Fig.2

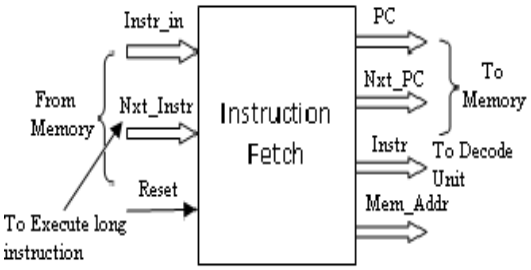

Fig.2 Instruction Fetch Unit

3.2 Branch Prediction:

The architecture uses dynamic branch prediction as it reduces branch penalties under hardware control. The prediction is made in Instruction Fetch of the pipeline. Thus branch prediction buffer is indexed by the lower order bits of the branch address in Instruction Fetch. In this paper, the architecture doesn't need any control hazards, as auto branch prediction is happening in the Fetch stage. Without branch prediction, the processor has to wait until the conditional jump has passed the execute cycle before the next instruction can enter the fetch stage in instruction pipeline. The branch predictor attempts to avoid the waste of time whether the conditional jump is most likely to be taken or not taken. The branch prediction part to be the most likely is then fetched and speculatively executed. This will increase flow in instruction pipeline and achieve high effective performance.

3.2 Instruction Decode (ID):

The control unit generates all the control signals needed to control the coordination among the entire component of the processor. This unit generates signals that control all the read and write operation of the register file, and the Data memory. It is also responsible for generating signals that decide when to use the multiplier and when to use the ALU, and it also generates appropriate branch flags that are used by the Branch Decide unit. The instruction decode unit is shown in Fig 3.

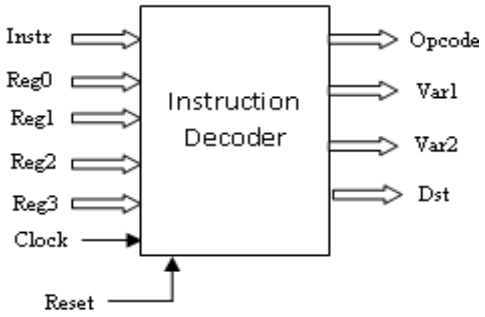

Fig.3 Instruction Decoder Unit

3.3 Register File:

This is a two port register file which can perform two simultaneous read and one write operation. It contains four 32-bit general purpose registers. The registers are named R0 through R4. R0 is a special register, which always contains the value zero and any write request to this register is always ignored. When the Reg_Write signal is high, a write operation is performed to the register.

3.4 Execution Unit:

This unit is responsible for providing signals to the ALU that indicates the operation it will perform. The input to this unit is the 5-bit opcode and the 2-bit function field of the instruction word. It uses these bits to decide the correct ALU operation for the current instruction cycle. This unit also provides another set of output that is used to gate the signals to the parts of the ALU that it will not be using for the current operation.

This stage consists of some control circuitry that forwards the appropriate data, generated by the

ALU or read from the Data Memory, to the register files to be written into the designated register. The block diagram of Execution Unit is shown in Fig. 4.

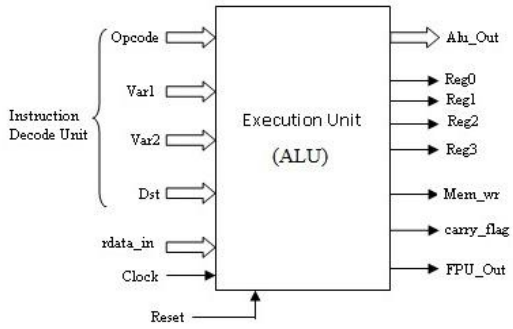

Fig.4 Execution Unit

3.5 Floating Point Unit:

Most of today's computers are equipped with specialized hardware that performs floating-point arithmetic with no special programming required. Floating point computational logic has long been a mandatory component of high performance computer systems as well as embedded systems and mobile applications. The advantage of floating point representation over fixed-point and integer representation is that it can support a much wider range of values. In the present work 32-bit FPU is incorporated, which supports Single Precision IEEE-754 format. The IEEE-754 standard defines a Single as 1 bit for sign, 8 bits for exponent and 23 bits for mantissa.

The FPGA implementation of 32-bit Single Precision floating point unit provides to addition, subtraction, multiply and division operations for any two operands of the same format. The destination format shall be at least as wide as the operands format. The block diagram of floating point unit is shown in Fig.5.

3.5.1Floating Point Addition/Subtraction Algorithm:

Step I: Align exponents (if necessary)

Temporarily De-normalize one with smaller exponent.

Add 2 to exponent! Shift significand right by 2.

Step II: add significands.

Remember overflow, it isn't treated like integer overflow.

Step III: Normalize result.

Shift significand right by 1 add 1 to exponent.

3.5.2 Floating Point Multiplication Algorithm:

Step I: Normalization

Step II: Addition of exponents in biased notation (must subtract bias)

Step III: When multiplying two normalized significands.

3.5.3 Floating Point Division Algorithm:

Step I: Normalization

Step II: Subtraction of exponents in biased notation (must add bias).

Step III: Binary point placement.

Step IV: Compare the significands.

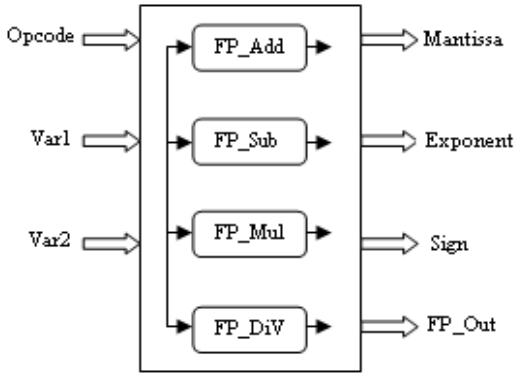

Fig.5 Floating Point

3.6 Memory Read/Write:

The architecture used is Modified Harvard architecture .This module supports 512 depth of 32-bit data words. The Load and Store instructions are used to access this module. Finally, the Memory Access stage is where, if necessary, system memory is accessed for data. Also, if a write to data memory is required by the instruction, it is done in this stage. In order to avoid additional complications it is assumed that a single read or write is accomplished within a single CPU clock cycle.

3.7 Instruction Set:

The instruction set used in this architecture consists of arithmetic, logical, floating point, memory and branch instructions. It will have short (8-bit) and long (16-bit) instructions. For all Arithmetic and Logical operations 8-bit instructions are used, and for all memory transactions and jump instructions 16-bit instructions are used. It will also have special instructions to access external ports.

The architecture will also have internal 64-bit general purpose registers that can be used in all operations. For all the jump instruction, the processor architecture, will automatically flushes the data in the pipeline, so as to avoid any misbehavior.

IV. SIMULATION RESULTS:

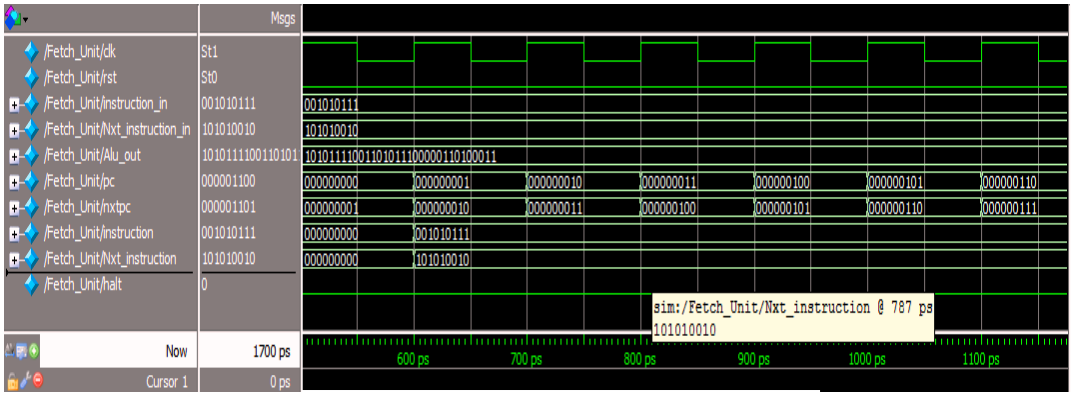

Fig.6 Simulation Waveform of Instruction Fetch Unit

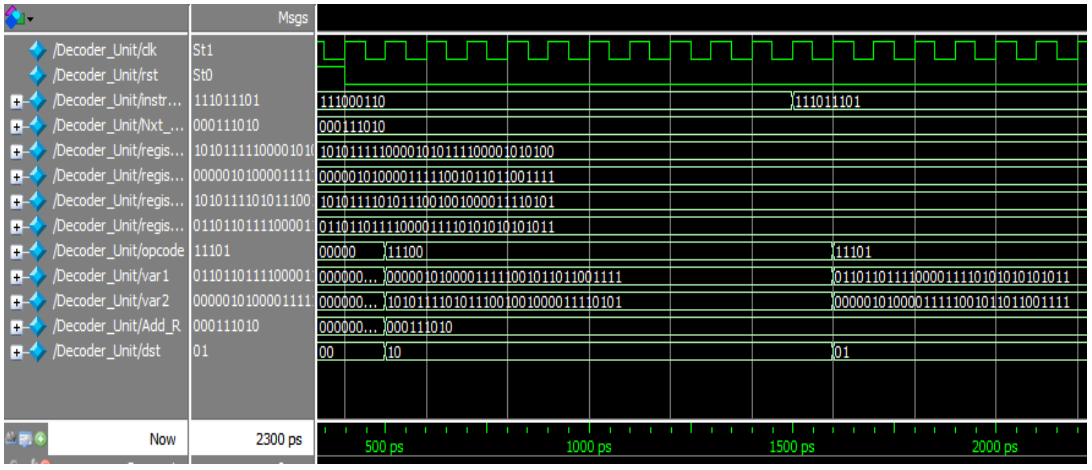

Fig.7 Simulation Waveform of Instruction Decoder Unit

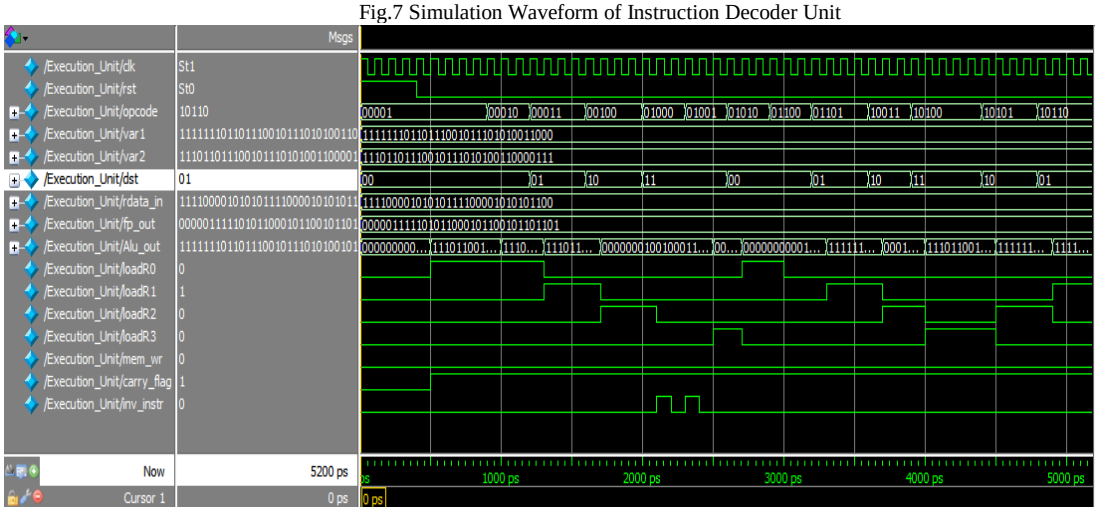

Fig. 8 Simulation Waveform of Execution Unit

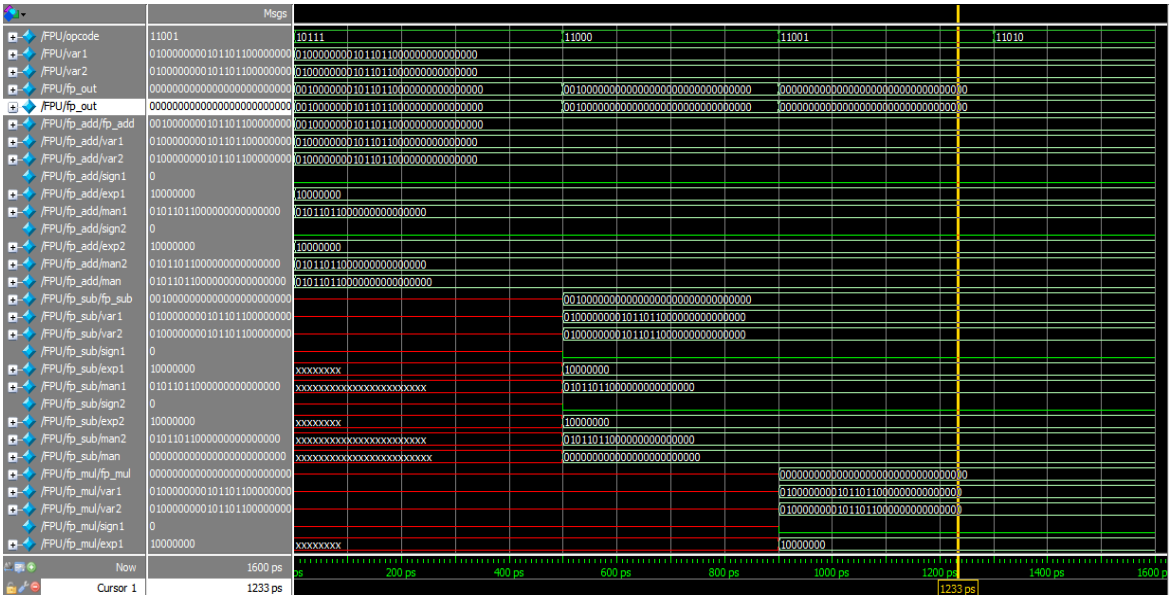

Fig.9 Simulation Waveform of Single Precision Floating Point Unit

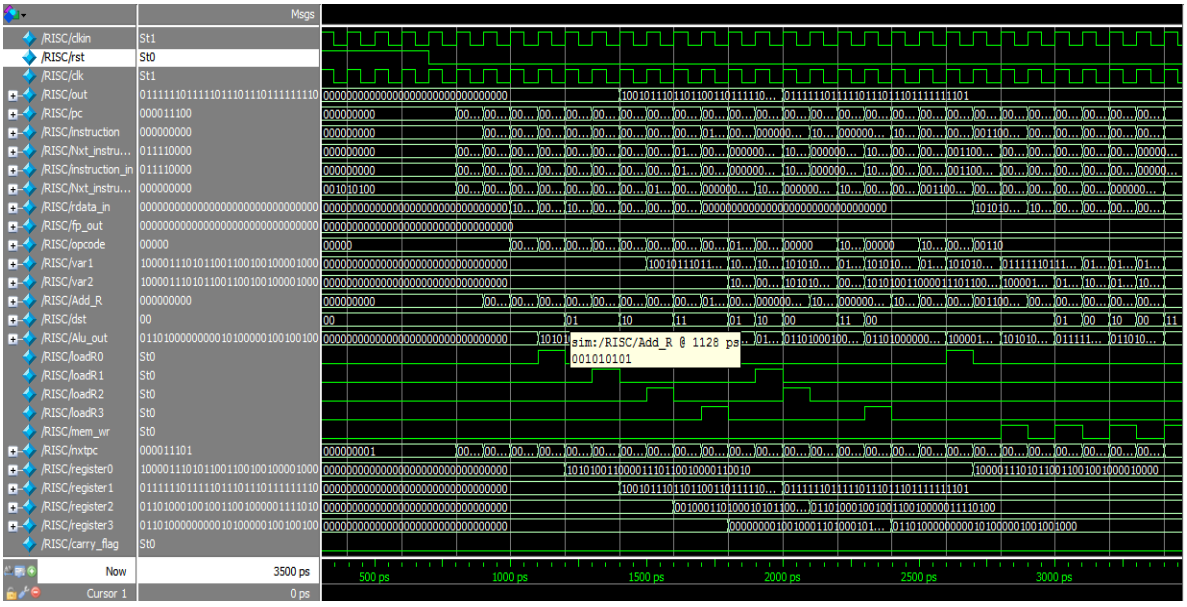

Fig.10 Simulation Waveform of 32-bit RISC Processor

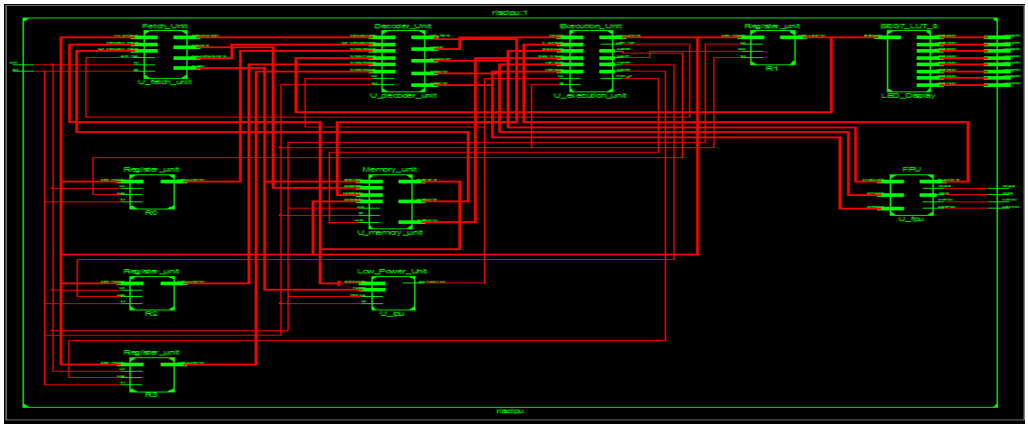

Fig.11 RTL Schematic view of Proposed Processor

V. FLOW CHART OF PROPOSED PROCESSOR

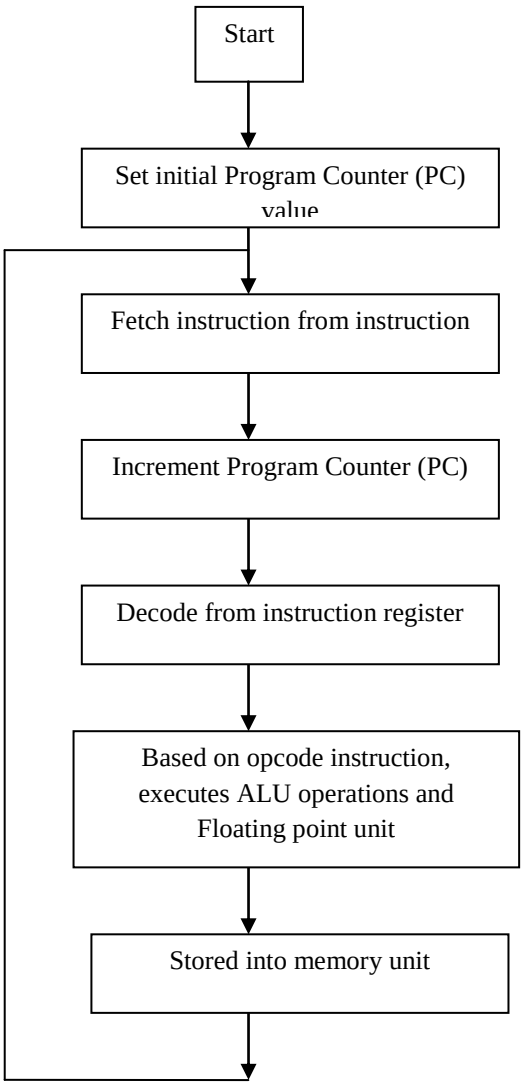

Fig.13 Flow Chart of Proposed Processor

V. CONCLUSION

FPGA based pipelined 32-bit RISC processor with Single Precision Floating Point Unit is designed and Verilog coding adopted. The design is implemented on Altera DE2 FPGA on which Arithmetic operations, Branch operations, Logical functions and Floating Point Arithmetic Operations are verified. Pipelining would not flush when branch instruction occurs as it is implemented using dynamic branch prediction. This will increase flow in instruction pipeline and high effective performance. This architecture has become indispensable and increasingly important in many applications like signal processing, graphics and medical.

REFERENCES

[1] R. Uma, "Design and Performance Analysis of 8-bit RISC processor using Xilinx Tool", International Journal of Engineering Research and Applications (IJERA), ISSN: 2248-9622, Vol-2, Mar-Apr-2012.

[2] J. Poornima, G.V.Ganesh, M. Jyothi, M. Shanti and A.Jhansi Rani,"Design and implementation of pipelined 32-bit Advanced RISC processor for various D.S.P Applications", Proceedings of International Journal of Computer Science and Information Technology, ISSN: 3208-3213, Vol-3(1), June-2012.

[3] Xiao Li, Longwei Ji, Bo Shen, Wenhong Li, Quianling Zhang, " VLSI implementation of a High-performance 32-bit RISC microprocessor", Communications, Circuits and Systems and West Sino Expositions, IEEE , International Conference, ISSN:1458-1461,Vol-2,2002.

[4] http://elearning.vtu.ac.in/12/enotes/Adv_Com_Arch/Pipeline/Unit2-KGM.pdf.

[5] Asmita Haveliya "Design and simulation of 32-point FFT using Radix-2 Algorithm for FPGA Implementation" IEEE, 167-171, 2012.

[6] The pipelined RISC-16 ENEE 446: Digital Computer Deisgn, Fall 2000 by Prof. Bruce Jacob.

[7] Preetam Bhosle, Hari Krishna Moorthy," FPGA Implementation of Low Power Pipelined 32-bit RISC Processor", Proceedings of International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Vol-1, Issue-3, August 2012.

[8] Charles H.Roth Jr, "Deisgn Systems Design using VHDL", Prentice Hall 2nd Edition, 2000.

[9] Samir Palnitkar,"Verilog HDL: A Guide to Digital Design and Synthesis", Prentice Hall, 2nd Edition, 2003.

[10] http://en.wikipedia.org/wiki/Singleprecision_floating-point_format.

[11] Galani Tina G,Riya Saini and R.D.Daruwala, "Design and Implementation of 32-bit RISC Processor using Xilinx", International Journal of Emerging Trends in Electrical and Electronics(IJETEE)-ISNN:2320-9569,Vol No.5,Issue 1,July-2013.