International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 10 – No.9, May 2016 – www.ijais.org

# Design of 32-bit 3-Stage Pipelined Processor based on MIPS in Verilog HDL and Implementation on FPGA Virtex7

Husainali S. Bhimani
Dept. of Electronics and Communication
Charusat University, Changa

Hitesh N. Patel
Faculty of Electronics and Communication
Charusat University, Changa

Abhishek A. Davda
Dept. of VLSI & Embedded Systems
DAIICT, Gandhinagar

## ABSTRACT
Reduced Instruction Set Compiler (RISC) is a microprocessor that had been designed to perform a small set of instructions, with the aim of increasing the overall speed of the processor. This paper presents 32 bit 3 stage architecture inspired by MIPS. The Idea of this paper is to implement custom architecture like MIPS 32 bit architecture in VERILOG HDL. The last step is to implement MIPS on FPGA (Field programmable gate array). MIPS (Microprocessor without Interlocked pipeline stages) processors are one of the first successful classical RISC architecture.

## Keywords
MIPS, RISC, FPGA, VERILOG HDL.

## 1. INTRODUCTION
RISC stands for reduced instruction set computer. RISC processors are widely used processor nowadays. There are many advantages of RISC over CISC (complex instruction set computer) architecture. Those are high performance, fixed length instruction, single cycle execution, smaller size, and short development time. RISC design resulted in computers that execute instructions faster than other computers built of the same technology [5].Complexity is at software level rather than hardware level. One of the main advantages is pipelining which is absent in CISC processor. Pipelining improves overall performance of RISC architecture [2].

This paper presents architecture which is inspired by one of the very popular RISC architecture i.e., MIPS. MIPS stand for microprocessor without interlocked pipeline stages. It is an open source soft processor & one of the first successful RISC architecture. MIPS are widely used in embedded systems. MIPS processor design is based on the RISC design principle that concentrates on load/store architecture [3]. MIPS has already been pronoun of MIPS instruction set and MIPS instruction set architecture [4].

This paper presents 3 stage RISC single cycle 32 bit pipelined processor inspired by MIPS having large set of registers. This architecture covers all basic instructions for general computation including load/store instructions.

## 2. INSTRUCTION FORMAT
Proposed architecture supports Two Types of instructions.

These are R-type instructions and J-type instructions. It considers two source registers, one destination register, shift value and opcodes for instructions.

### 2.1 Type: 1 General instruction format

| 32 Bit Instruction | | | | |
|---|---|---|---|---|
| 31-26 | 25-19 | 18-12 | 11-05 | 04-00 |
| Opcode | RS | RT | RD | SH |

**Fig 1: R-Type Instructions Format**

Figure 1 shows R-Type Instructions Format.

5 bits: for shifting or rotating operation (2^5=32 bits).

6 bits: opcode (2^6=64) to support instructions up to 64.

7 bits: To access registers up to 127$^{th}$. (R0-R127)

Example:

1. ADD r0, r1, r2;

r0=r1+r2

2. AND r5, r6, r7;

r5=r6 & r7

### 2.2 Type: 2 Jump instruction format

| JMP Instruction Format | | |
|---|---|---|
| 31-26 | 25-07 | 06-00 |
| Opcode | 00000000000000000000 | Target address |

**Fig 2: Jump type Instruction Format**

Figure 2 shows Jump type Instruction Format

Example: JMP LABEL

Suppose LABEL is located at 125$^{th}$ instruction then PC will immediately jumps to 125$^{th}$ instruction.

## 3. LIST OF INSTRUCTIONS AND OPCODE FOR REGISTERS

### 3.1 Instructions and their opcodes
Table 1 shows list of instructions included in purposed architecture with respective opcodes.

**Table 1. Opcode for Instructions**

| OPCODE FOR INSTRUCTIONS | |
|---|---|
| OPCODE | INSTRUCTION |
| 000000 | ADD |
| 000001 | ADC |
| 000010 | SUB |
| 000011 | SBC |
| 000100 | LSL |
| 000101 | LSR |
| 000110 | CMP |
| 000111 | NOP |
| 001000 | MOV |
| 001001 | AND |
| 001010 | OR |
| 001011 | NOT |
| 001100 | XOR |
| 001101 | JMP |
| 001110 | ROTL |
| 001111 | ROTR |
| 010000 | INC |
| 010001 | DEC |
| 010010 | ZERO |
| 010011 | LOAD |
| 010100 | STORE |

## 3.2 Registers and their Opcodes

Table 2 shows how register is accessed with their opcode.

Registers can be accessed like R0 as 0000000, R1 as 0000001, R2 as 0000010, R3 as 0000011, R4 as 0000100, R5 as 0000101, R6 as 0000110, R7 as 0000111, R8 as 0001000, and so on.

**Table 2. Opcode for Registers**

| Rs/Rt/Rd | Accessed Register |
|---|---|
| 0000000 | R0 |
| 0000001 | R1 |
| 0000010 | R2 |
| 0000011 | R3 |
| . | . |
| . | . |
| . | . |
| . | . |
| 1111111 | R127 |

# 4. INSTRUCTION MEMORY, REGISTER FILE AND DATA MEMORY

Here Instruction memory, register file and data memory are illustrated as blocks.

## 4.1 Instruction memory

Size: 128 bits
Figure 3 shows Instruction memory block.

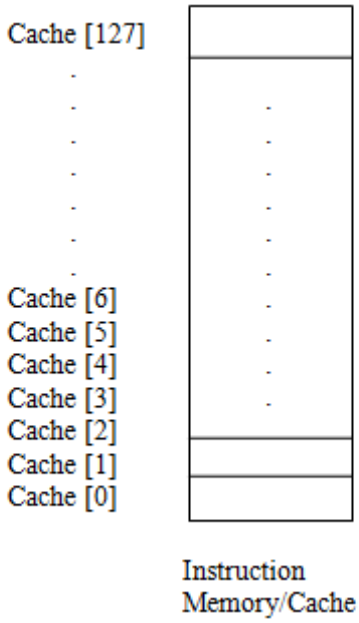


**Fig 3: Instruction memory**

## 4.2 Register File

Figure 4 shows Register File block.

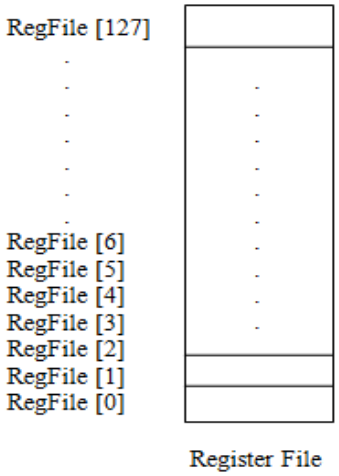


**Fig 4: Register File**

# 5. FLAG STRUCTURE

Figure 6 shows Flag structure.

Size of flag is 32 bits.

Flag [0]: Zero flag. It is updated on zero output of ALU.

Flag [1]: Carry flag. It is updated if there is carry/borrow generated by addition/subtraction operations.

Flag [2]: Parity flag. It is set for odd 1's in result of ALU. It will be cleared for even 1's in result of ALU.

Flag [3]: Overflow flag. It is updated as set if there is any overflow in result.

Flag [4] - Flag [31]: Future use.

| FLAG | | | | |
|---|---|---|---|---|
| FLAG[31]-FLAG[4] | FLAG[3] | FLAG[2] | FLAG[1] | FLAG[0] |
| Future Use | Over Flow Flag | Parity Flag | Carry Flag | Zero Flag |

**Fig 6: Flag structure**

# 6. ARCHITECTURE

There are two architectures covered here. First one is original MIPS and second one is modified custom architecture inspired by MIPS.

## 6.1 MIPS Architecture [1]

Figure 7 shows classical MIPS architecture which is single cycle RISC architecture. It was designed by John L. Hennessy at Stanford University. It is first classical and successful RISC processor which is of 5 stages i.e. instruction fetch, instruction decode, instruction execute, memory access and write back. Instruction Fetch module fetches instruction. Next module decodes it. Execute module executes the instruction. Memory access is required for load and store instruction. Write back module writes data from ALU to register file or loads data from data memory. The width of data bus of MIPS is 32 bits.

## 6.2 Proposed Architecture

Figure 8 shows Fetch stage of proposed architecture.

Instruction fetch: This is the first step in which instructions are fetched from instruction memory according to the address provided by PC. After each clock PC will be incremented according to the previous instruction fetched. Pipe1 is of 64 bits size. It saves 32 bits for instruction and 32 bits as next PC.

### *6.2.2 Decode Stage*

Figure 9 shows decode stage of proposed architecture.

Instruction decode: This stage decodes instructions, shift value, registers to be accessed & getting their content.

#### *6.2.2.1 Control Signals*

Table 3 shows control signals part-1 for processor.

Table 4 shows control signals part-2 for processor.

Pipe2 is of 124 bits size. It saves 14 bits of CSTSIG which is primarily used to control the whole processor.

Pipe2 [13:00]: CSTSIG [13:00].

Pipe2 [19:14]: Future use.

Pipe2 [52:20]: 32 bits Address into file register (generally). For store instruction it gives data.

Pipe2 [83:52]: Rtout.

Pipe2 [115:84]: Rsout.

Pipe2 [120:116]: shift/rotate use.

Pipe2 [115:84]: future use.

Note that this architecture is with all control signals. Control signals can be considered as the heart of processor. All the critical decisions are taken by control logic of a processor.

### *6.2.3 Execute/Data access/Write back*

Figure 10 shows Execute/Data Access/Write back stage of purposed architecture.

Instructions are executed in this stage. Data memory is accessed for load/store instructions. If it's not load/store instruction then result of ALU output is stored in register and flags are updated. Figure shows different required CSTSIG value for data memory and multiplexers.

Pipe2 is of 124 bits size. It saves 14 bits of CSTSIG which is primarily used to control whole processor.

Pipe2 [13:00]: CSTSIG [13:00].

Pipe2 [19:14]: Future use.

Pipe2 [52:20]: 32 bits Address into file register (generally). For store instruction it gives data.

Pipe2 [83:52]: Rtout.

Pipe2 [115:84]: Rsout.

Pipe2 [120:116]: shift/rotate use.

Pipe2 [115:84]: future use.

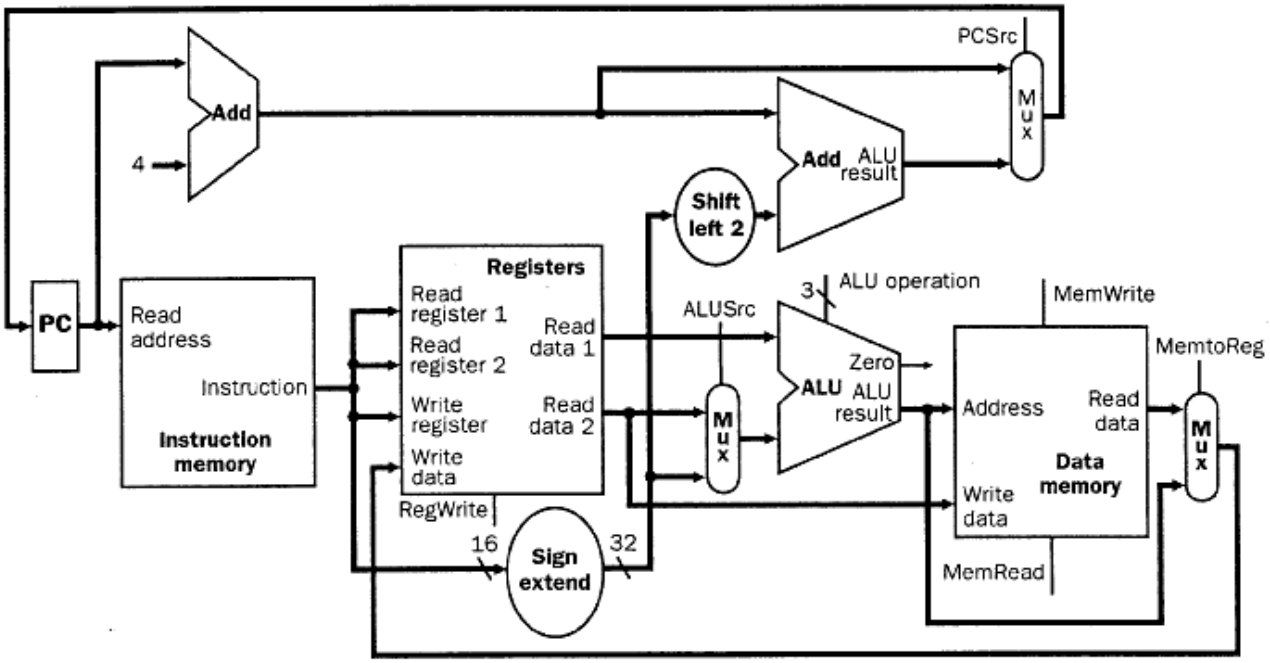


**Fig 7: MIPS Architecture [1]**

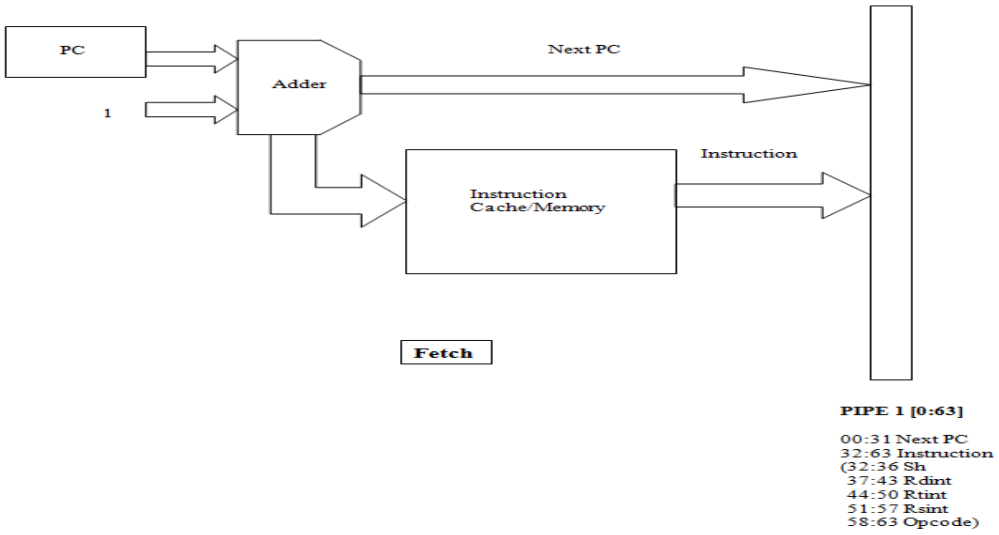


**Fig 8: Fetch Stage**

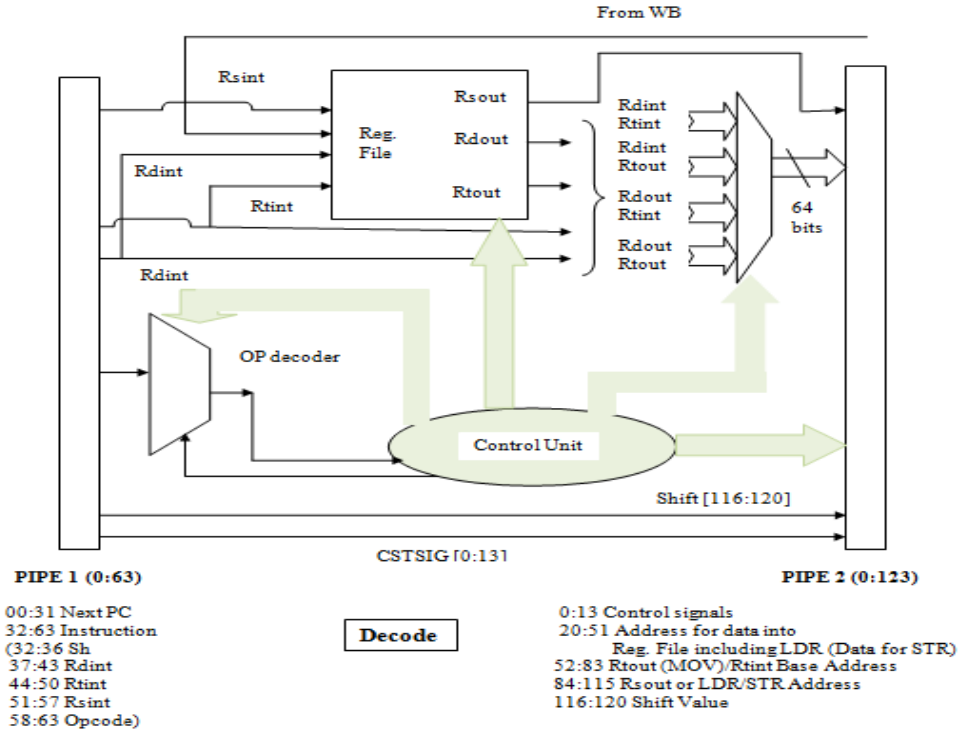


**Fig 9: Decode Stage**

**Table 3. Control signals part-1**

CSTSIG[0:13]

| CSTSIG[n] | STATUS | DESCRIPTION |
|---|---|---|
| CSTSIG[1] | 0 | Addition enabled including LDR,STR |
| | 1 | Subtraction enabled including CMP |
| CSTSIG[8:7] | 00 | AND |
| | 01 | OR |
| | 10 | XOR |
| | 11 | NOT |
| CSTSIG[0] | 0 | - |
| | 1 | Always asserted to update flags |
| CSTSIG[10:09] | 00 | To enable INC |
| | 01 | To enable DEC |
| | 10 | To enable ZERO |
| | 11 | -- |

**Table 4: Control signals part 2**

CSTSIG[0:13]

| CSTSIG[n] | STATUS | DESCRIPTION |
|---|---|---|
| CSTSIG[4] | 0 | RdFinal address Disabled |
| | 1 | RdFinal address enabled |
| CSTSIG[3] | 00 | LDR **data** from memory to register file |
| | 01 | ALU o/p or Rt (**data** to be moved) |
| CSTSIG[2] | 0 | ALU o/p |
| | 1 | Rt (data to be moved) |
| CSTSIG[12] | 0 | MemWrite Disabled |
| | 1 | MemWrite Enabled |
| CSTSIG[11] | 0 | MemRead Disabled |
| | 1 | MemRead Enabled |
| CSTSIG[1:0] | 00 | LSL |
| | 01 | LSR |
| | 10 | ROTL |
| | 11 | ROTR |
| CSTSIG[13] | 0 | To enable ADC |
| | 1 | To enable SBC |

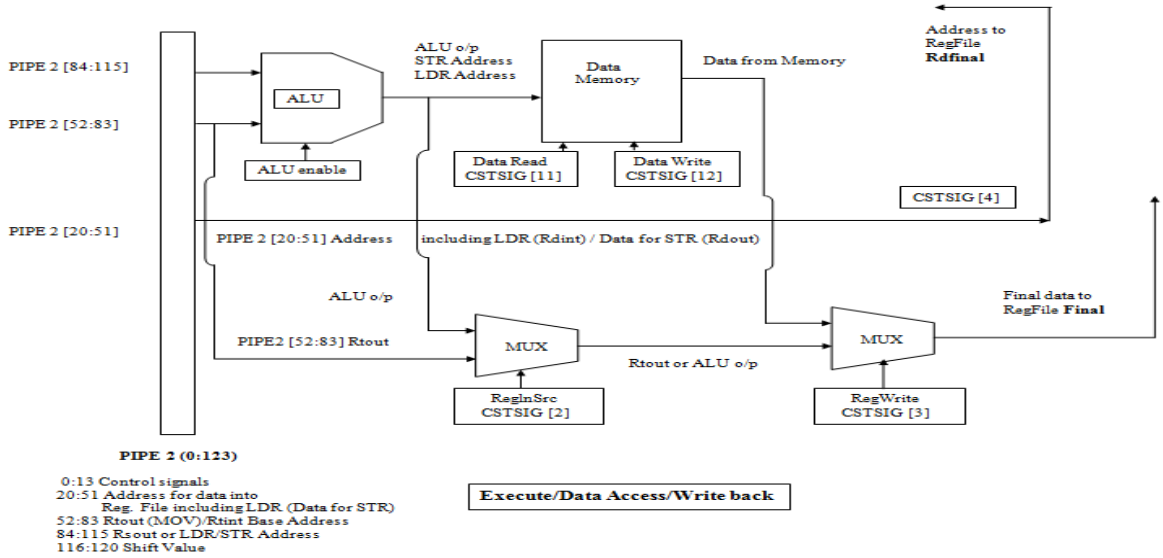


**Fig 10: Execute/Data Access/Write back stage**

## 7. SIMULATION RESULTS

ISIM is used for simulation and results have been verified. The complete project has been developed on Package: FFG1157, FPGA VIRTEX 7 and device is XC7VX330T. Different output patterns have been generated with RISC Instructions. Coding of processor is done in VERILOG HDL.

### 7.1 RTL Schematic

Figure 11 shows RTL Schematic of three stage processor.

### 7.2 Technology Schematic

Figure 12 shows Technology Schematic of three stage processor.

It shows different output values of Rdfinal (address into register file) and Final (data to be written into register file)

### 7.3 Simulation Waveforms

#### *7.3.1 General simulation waveform*

Figure 13 shows General simulation waveform.

### 7.4 Instruction Cache Simulation Waveform

Figure 14 shows General simulation waveform.

### 7.5 Register File Simulation Waveform

Figure 15 shows Register file simulation waveform.

### 7.6 Data Memory Simulation Waveform

Figure 16 shows Data memory simulation waveform.

## 7.7 Flag Simulation Waveform

### *7.7.1 Flag[0]- Flag[3]*

Figure 17 shows Flag[0]-Flag[3] simulation waveform.

### *7.7.2 Flag[0]- Flag[31]*

Figure 18 shows Flag[0]-Flag[31] simulation waveform.

## 7.8 XPower Analysis

Figure 19 shows XPower Analysis of purposed architecture.

## 8. DEVICE UTILIZATION SUMMARY

Selected Device : 7vx330tffg1157-3

**Table 5. Logic utilization**

| | | |
|---|---|---|
| Number of Slice Registers: | 32 out of 408000 | 0% |
| Number of Slice LUTs: | 40 out of 204000 | 0% |
| Number used as Logic: | 40 out of 204000 | 0% |

**Table 6. Logic distribution**

| | | |
|---|---|---|
| Number of LUT Flip Flop pairs used: | 62 | NA |
| Number with an unused Flip Flop: | 30 out of 62 | 48% |
| Number with an unused LUT: | 22 out of 62 | 35% |
| Number of fully used LUT-FF pairs: | 10 out of 62 | 16% |
| Number of unique control sets: | 4 | NA |

**Table 7. IO utilization**

| | | |
|---|---|---|
| Number of IOs: | 4 | NA |
| Number of bonded IOBs: | 3 out of 600 | 0% |

**Table 8. Specific feature utilization**

| | | |
|---|---|---|
| Number of Block RAM/FIFO: | 1 out of 750 | 0% |
| Number using Block RAM only: | 1 | NA |
| Number of BUFG/BUFGCTRL/BUFHCEs: | 1 out of 200 | 0% |

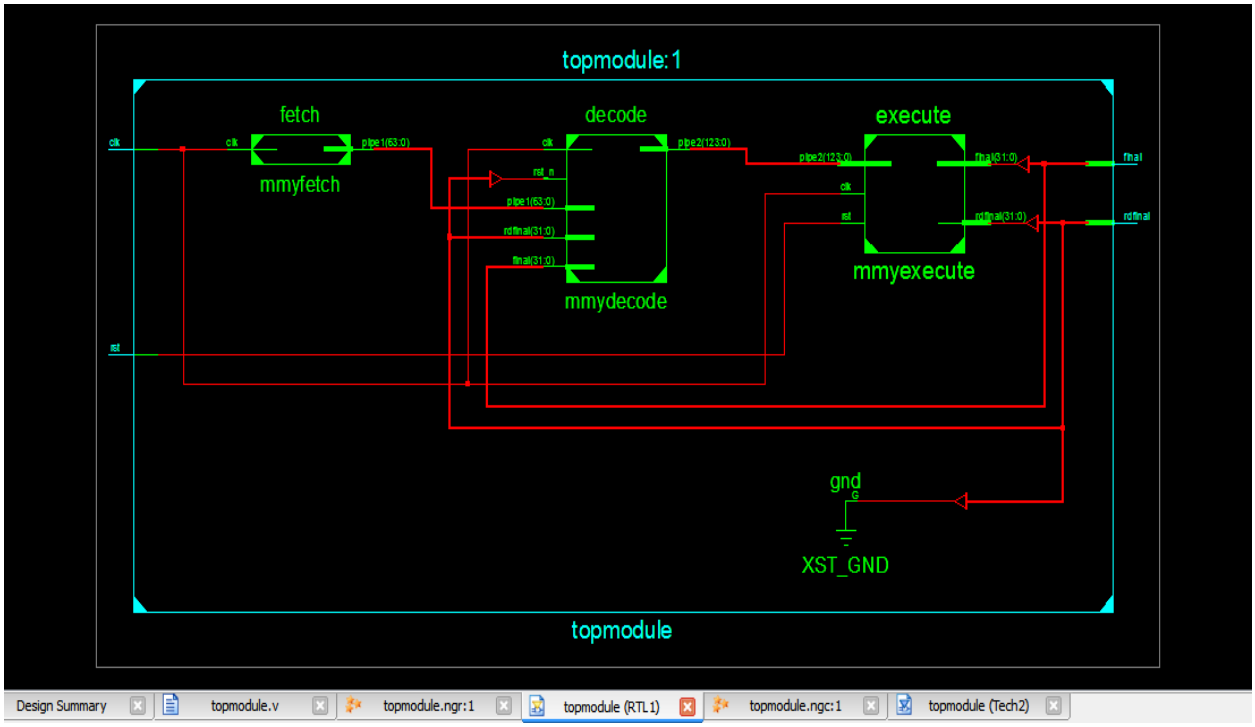


**Fig 11: RTL Schematic**

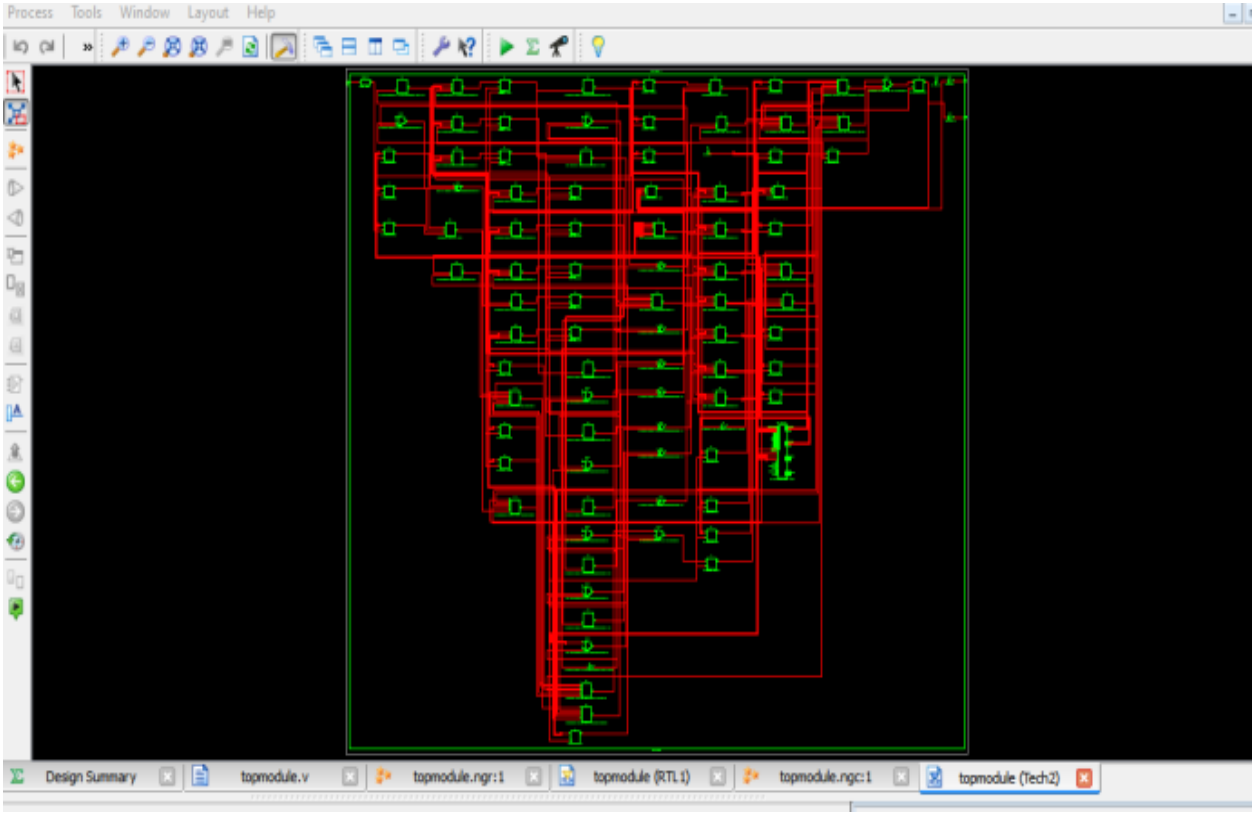


**Fig 12: Technology Schematic**

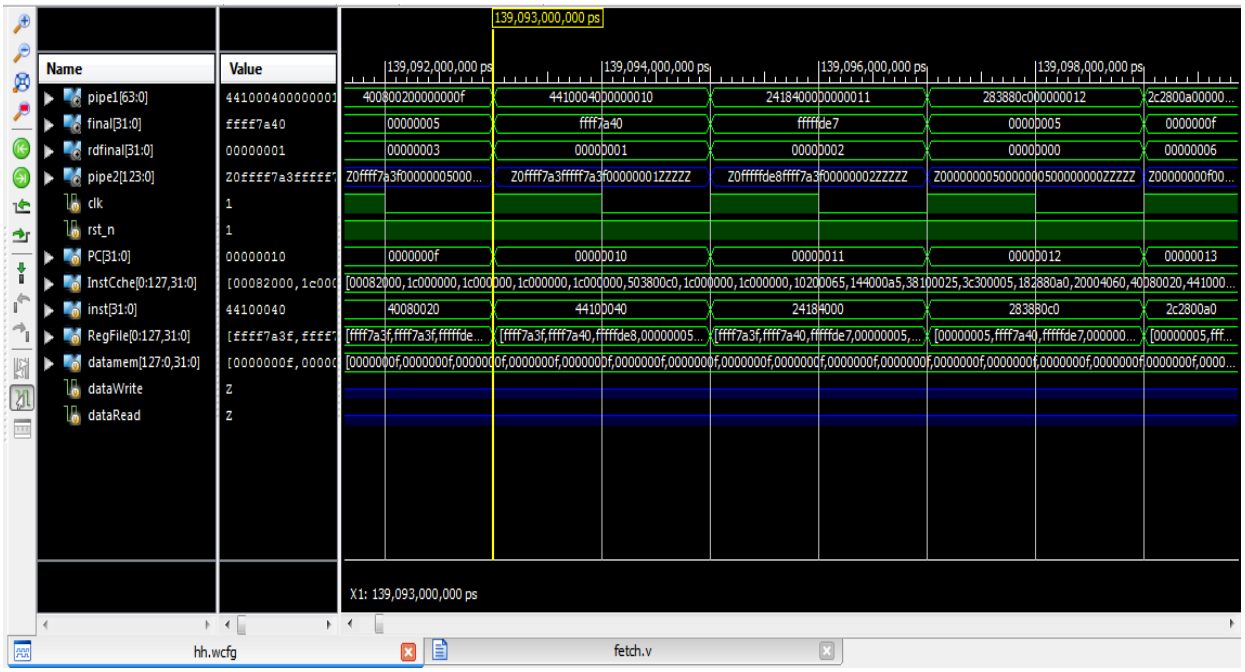


**Fig 13: General simulation waveform**

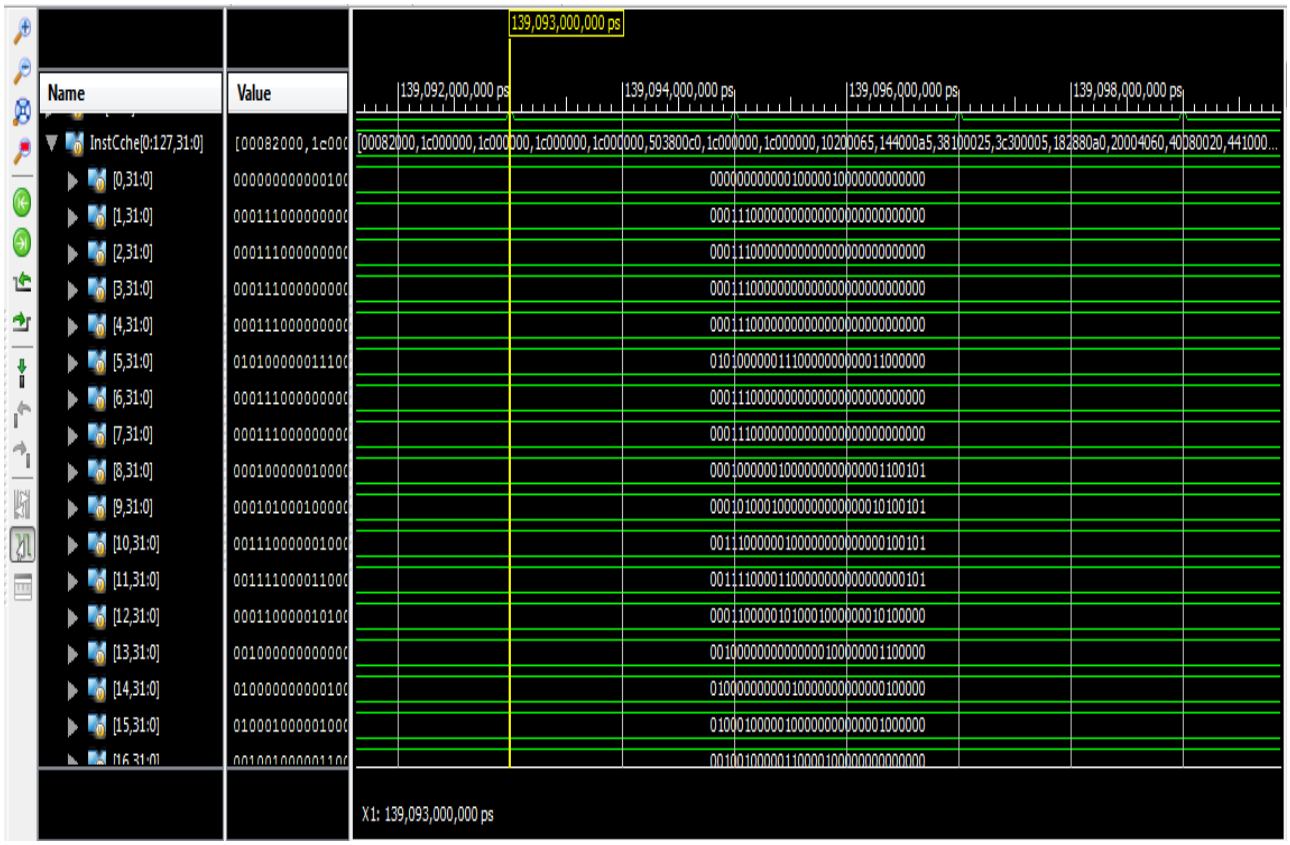


**Fig 14: Instruction Cache Simulation Waveform**

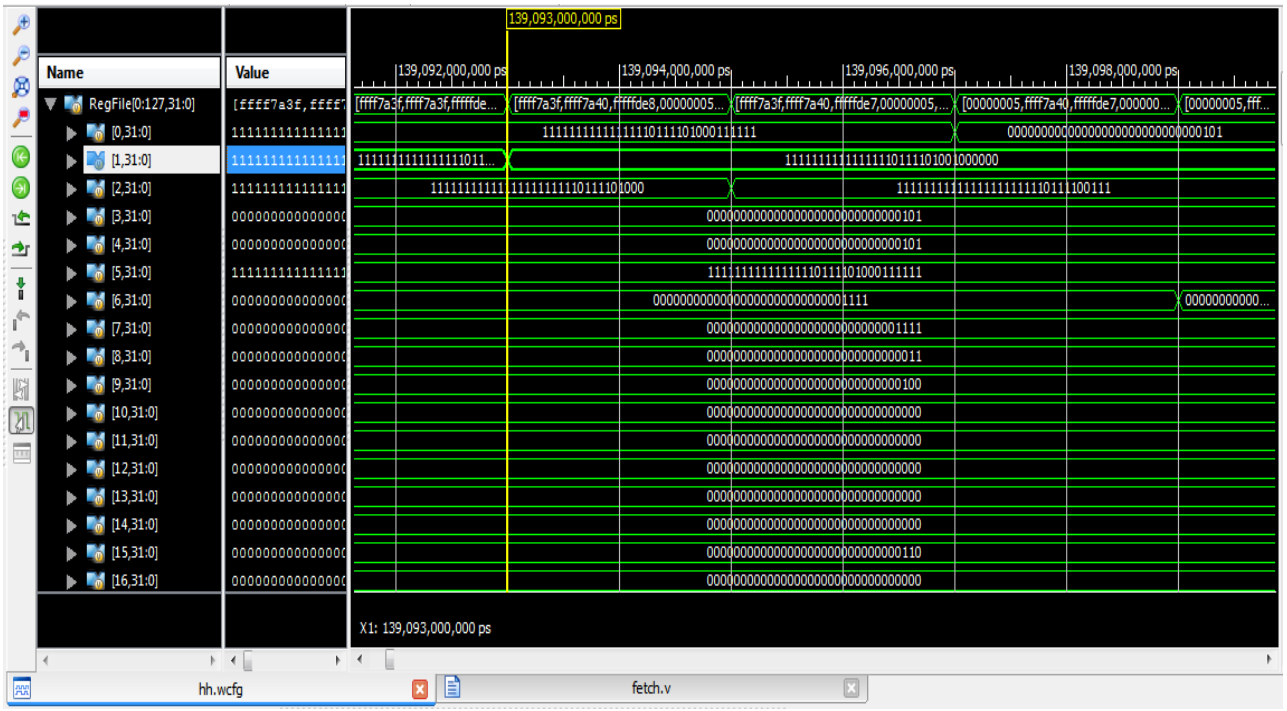


**Fig 15: Register file simulation waveform**

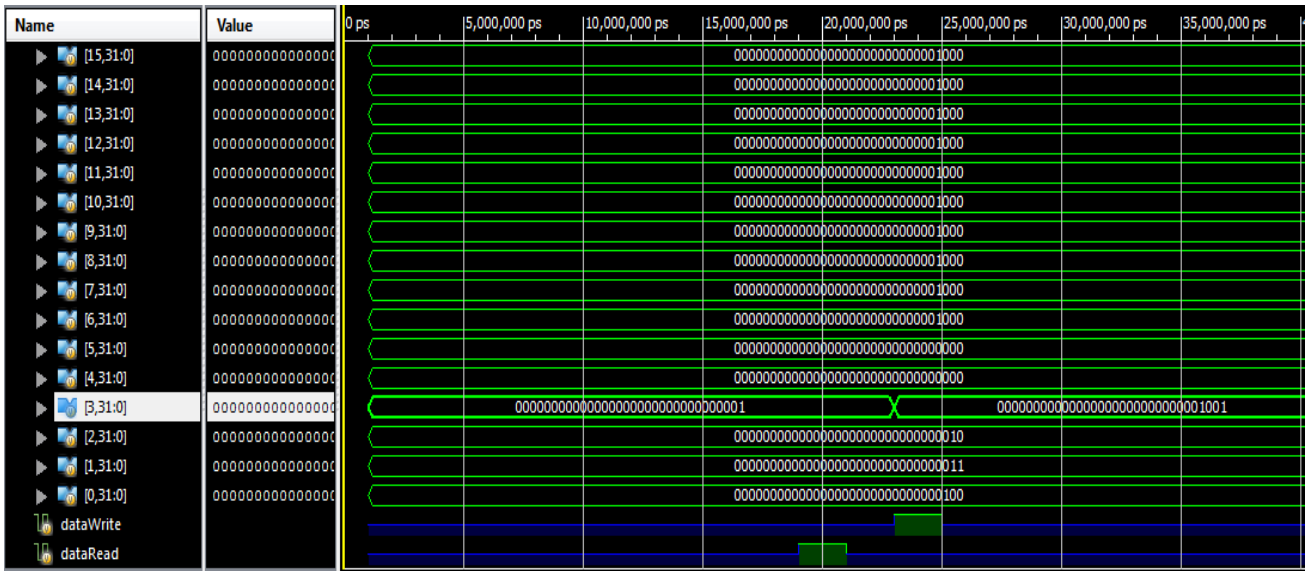


**Fig 16: Data memory simulation waveform**

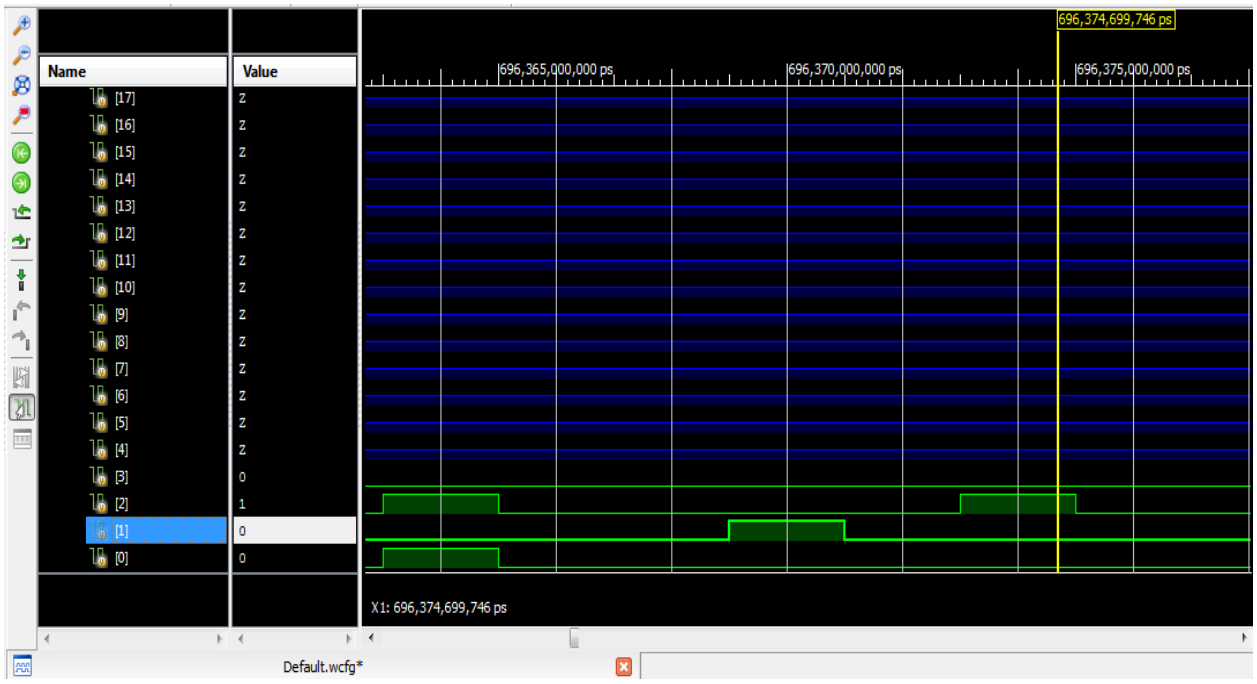


**Fig 17: Flag simulation waveform**

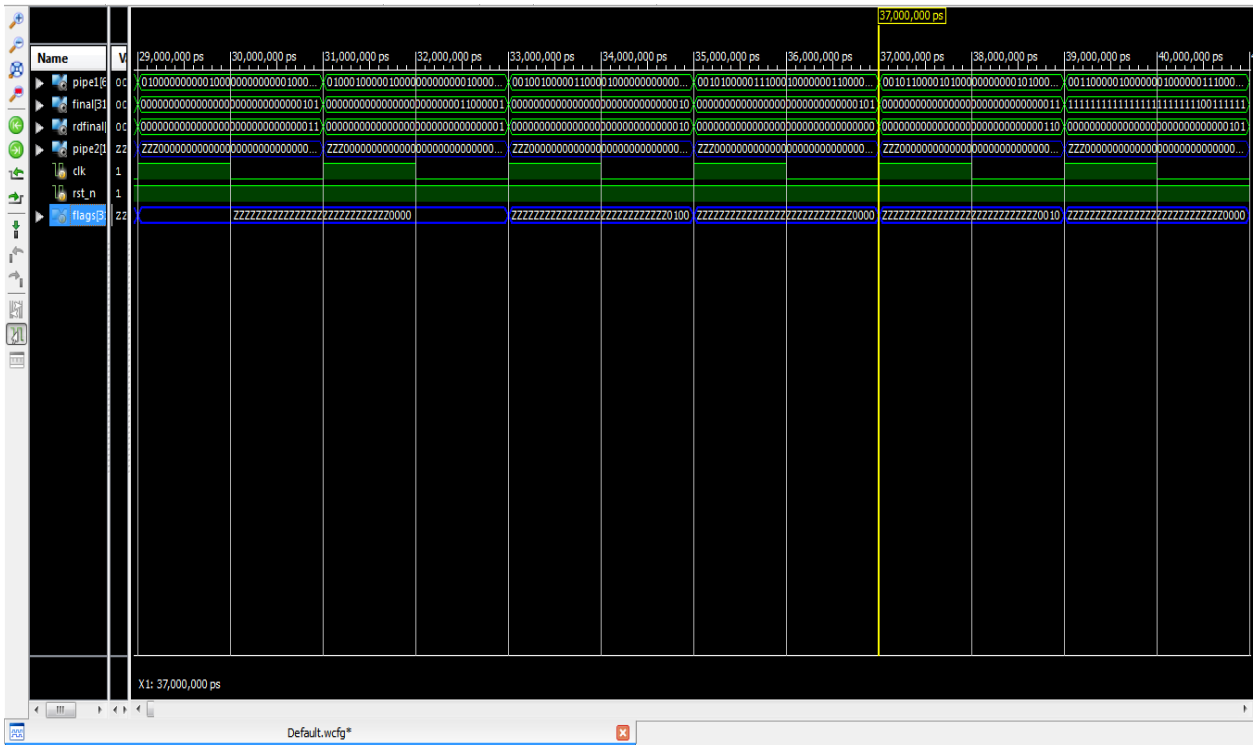


**Fig 18: Flag simulation waveform**

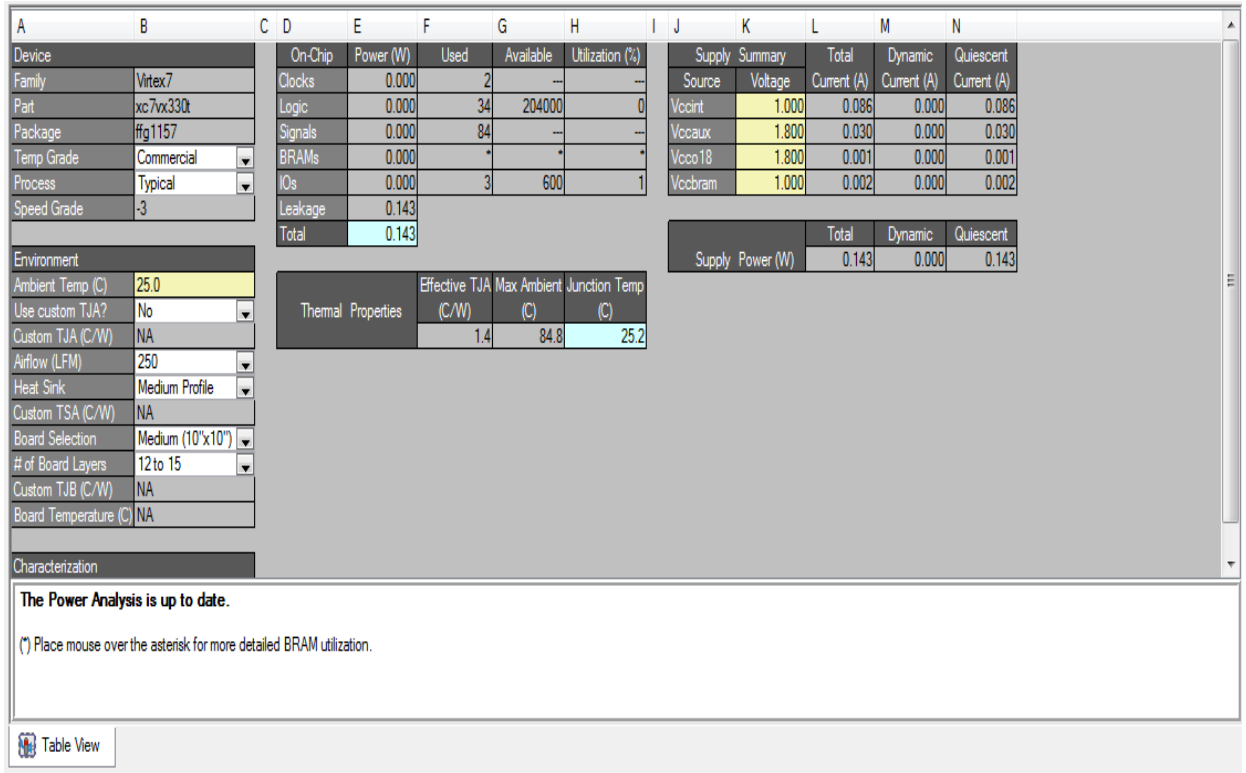


**Fig 19: XPower Analysis**

## 9. DESIGN SUMMARY

Top Level Output File Name: topmodule.ngc

Primitive and Black Box Usage:

# BELS : 59
# GND : 1
# INV : 2
# LUT1 : 7
# LUT2 : 3

| | | |
|---|---|---|
| # | LUT3 | : 11 |
| # | LUT4 | : 7 |
| # | LUT5 | : 4 |
| # | LUT6 | : 6 |
| # | MUXCY | : 7 |
| # | MUXF7 | : 2 |
| # | VCC | : 1 |
| # | XORCY | : 8 |
| # | Flops/Latches | : 32 |
| # | FD | : 22 |
| # | FDR | : 8 |
| # | LDE | : 2 |
| # | RAMS | : 1 |
| # | RAMB18E1 | : 1 |
| # | Clock Buffers | : 1 |
| # | BUFGP | : 1 |
| # | IO Buffers | : 2 |
| # | OBUF | : 2 |

Coding: Verilog

Implementation:

Family: Virtex7

Device: XC7VX330T

Package: FFG1157

Synthesis Tool: XST (VHDL/Verilog)

Simulator: ISim (VHDL/Verilog**)**

## 10. FEATURES

Covers all basic instructions for general computation

3 stage processor

32 bit processor

Single cycle Processor

Pipelining Implemented for faster operation

Based on MIPS

Frequency: 310.878 MHz

Clock Period: 3.217 ns

Data Memory: 128 bits

Instruction Memory Size: 128 bits

Register Files: 128 registers

Large Register File (R0-R127)

Supports load store instructions.

Supports arithmetic, logical and comparison operation

Supports full rotation and shifting of 32 bits.

No structural hazard since separate data memory and instruction memory is used.

## 11. CONCLUSION

This work successfully proposes a custom design of 3 stage pipelined RISC processor architecture. Pipeline is implemented and every instruction is tested using test bench. Proposed architecture supports large set of registers (R0-R127). Design is simulated and synthesized using Xilinx ISE. Static timing analysis and power analysis of the design has been carried out. Feature such as data forwarding unit can also be added in future. The design can be synthesized using Cadence RTL complier in future. Also complete ASIC flow till RTL to GDS II can be done using Cadence SOC Encounter and it is possible to analyze the complete physical design flow.

## 12. ACKNOWLEDGEMENT

We wish to express our heartfelt appreciation to our friends, families and many who have rendered their support for the successful completion of the paper, both explicitly and implicitly.

## 13. REFERENCES

[1] David A. Patterson, John L. Hennessy 2005. Computer organization and design, 3rd Edition, Elsevier.

[2] Sharda P. Katke, G.P. Jain,"Design and Implementation of 5 Stages Pipelined Architecture in 32 Bit RISC Processor", IJETAE, Volume 2. Issue 4. April 2012, pp. 340-346.

[3] Preetam Bhosle, Hari Krishna Moorthy, "FPGA Implementation of low power pipelined 32-bit RISC Processor", International Journal of Innovative Technology and Exploring Engineering (IJITEE), August 2012.

[4] Bai-ZhongYing, Computer Organization, Science Press, 2000.11.

[5] Charles E. Gimarc, Veljko M. Mhtinovic, "RISC Principles, Architecture, and Design", Computer Science Press Inc., 1989.