Journal of VLSI Design Tools and Technology
Volume 2, Issue (1-3), Compilation 2012

STM JOURNALS

Design of 16-bit Pipelined RISC Processor

Kalyan Acharjya[1]*, N.B.Singh[2]
[1]Department of Electronics and Communication Engineering, Tezpur Central University
[2]Central Electronics Engineering Research Institute, Pilani

ABSTRACT

This paper presents the design of efficient and high throughput 16-bit pipelined RISC Processor. The design is challenge with the pipelined stall problem, speed with external memory, coding density and clock frequency for cost effectiveness and higher performance processor design. Paper describes about the architecture, programming model, synthesis results and analysis of the design. The presented design has throughput ~167 MIPS at 500 MHz, which shows better performance at a particular frequency. The coding is carried out in Verilog HDL and functionality is verified through simulation and test benches at different stages including behavioral and gate level RTL code using Modelsim (Mentor Graphics). The synthesis part is done using Leonardo Spectrum (Mentor Graphics) and implementation done in Xilinx ISE 9.1.

***Index**: Processor, RISC, Pipelined and HDL*

****Author for Correspondence:*** E-mail kalyan.phy@gmail.com

1. INTRODUCTION

The advancements of VLSI design technology together with demand of higher performance processor is the challenge to meet the balance between speed and power, pipelined the instruction as well as low power memories for popularity of processor application in daily life, it urges the needs for more development of RISC processor. The main goal of the work is to design the processor that is more cost effectiveness and performance driven than their predecessor [1].

2. DESIGN ARCHITECTURE

The design of RISC architecture that reduces chip complexity by using simpler instruction as well as design based on the instruction set computer architecture. The strategy on simple architecture on the insight can be providing higher performance, and this simplicity enables much faster execution of each instruction. The designed architecture [2] is shown in Figure 1.

The processor design is to complete the instruction in four stages i.e. fetch, decodes, execute and store or write back. During the instruction fetch step the processor fetches instructions from the memory and computes the address of the next instruction by incrementing the program counter (PC).

During the second step, the Instruction decodes and register fetch step and decode the instruction as on instruction decoder. The third step is the Execution, memory address computation depending on what type of instruction is executing. The fourth step only takes place to store word of computation result or write back. The entire steps only include the

load and store word instructions to access the memory and arithmetic-logical instructions to compute results, which is main the feature of RISC architecture.

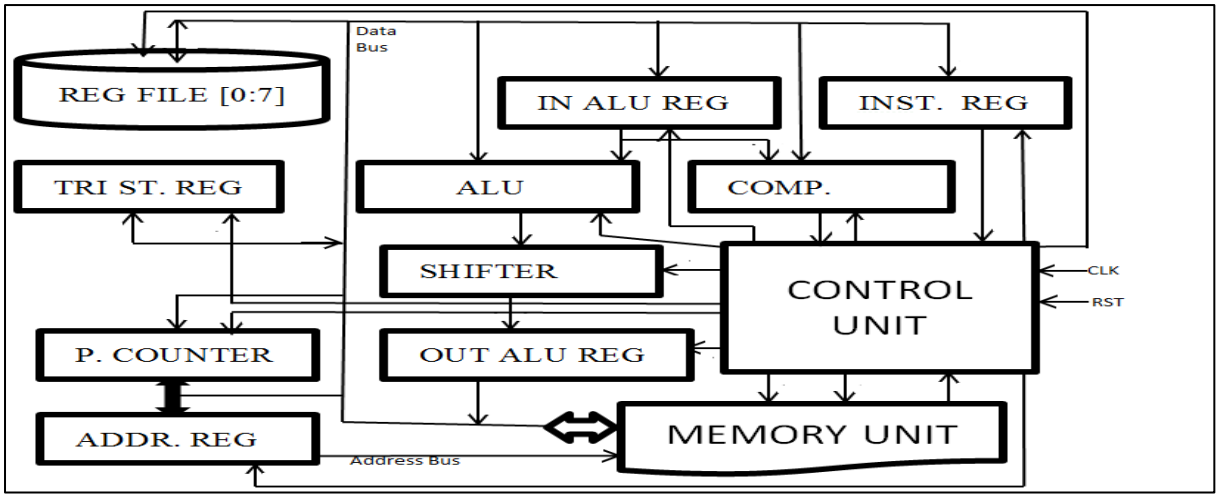

Fig. 1: *Design Processor Block Diagram.*

3. PROGRAMMING METHODOLOGY

The whole design is broken in thirteen modules including top level module. All modules are tested single and debugged the number of times for getting the correct result, after completion these small modules then top module used the call function to interfaces all modules. The top modules simulation result verified through analysis of its result of different instructions. The coding technique is different from traditional design, where completely removed the wait statement because it captures the extra silicon area which is the major issue for small devices. There are total 28 instructions. The word is 16-bit length. When some blocks are not used in particular instruction then that blocks are in dead mode during the execution, power can be saved by using this technique, when these blocks are required again then an enable signal is used from the control unit to activate it. The instruction format is shown in Figure 2. The left remaining bits are not used.

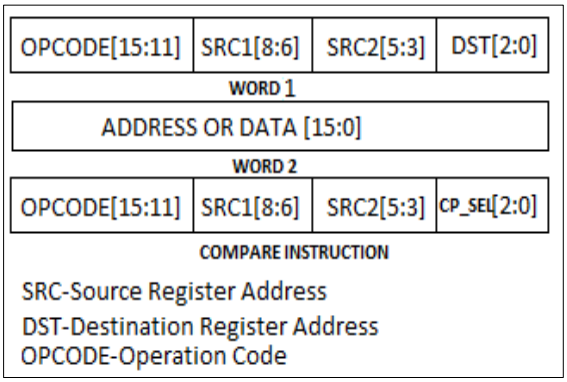

Fig. 2: *Instruction Format.*

Here the design is based on pipelined architecture to speed up the processor but branches are significant problem in the pipelined instructions [3]. One way to overcome this problem is using the branch delay slot, but it is not the sufficient solution. Another way of eliminating pipeline stalls was to predict the direction of the branch using a table stored in the decode unit. If the prediction was wrong, the instructions that

were in process had to be cancelled, resulting in wasted time and power [4–6]. Using the conditional execution the important thing is that to replace the test and branch sequences altogether. Using branches, this would require at least two branches to complete the instruction. Using conditional execution the sequence has three instructions with no branches. One of the two assignments executes, and the other acts as a nop (no operation). No branch prediction is needed, and the pipeline operates perfectly.

The design is multi cycle methodology which has several key advantages [1] over a single cycle implementation. First, it can share modules, allowing the use of fewer hardware components. Instead of multiple arithmetic logic units (ALU's), the multi cycle implementation uses only one. Only one memory is used for the data and the instructions also. Breaking complex instructions into multiple steps also allows to significantly increase the clock cycle because no longer has to base the clock on the instruction that takes the longest to execute. The functionality is verified through simulation and writhing test bench to ensure the fully logical correctness of the design.

4. SIMULATION AND SYNTHESIS

The first step to test the design through simulation of different type of instruction executes. The correctness also verified through writing test bench. The simulated waveform one part is shown in Figure 3.

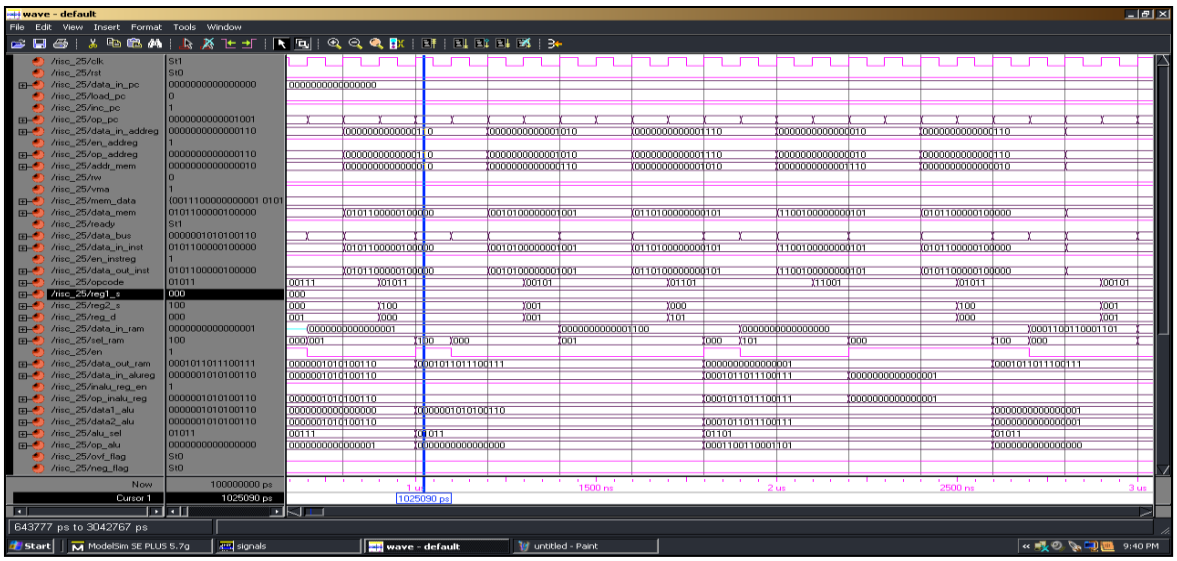

Fig. 3: *Simulated Waveform.*

The RTL code of the design is synthesized using FPGA Xilinx VertexII-Pro, device 2V40cs144 with speed grade-6. The whole architecture is mapped to generate the RTL schematic and gate level net list as shown in Figures 4 and 5.

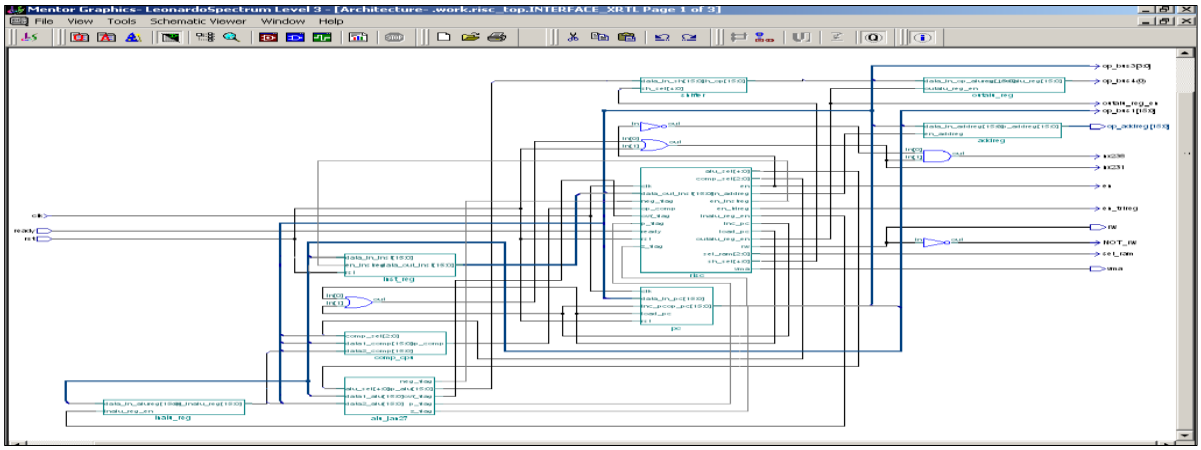

Fig. 4: *View of RTL Schematic.*

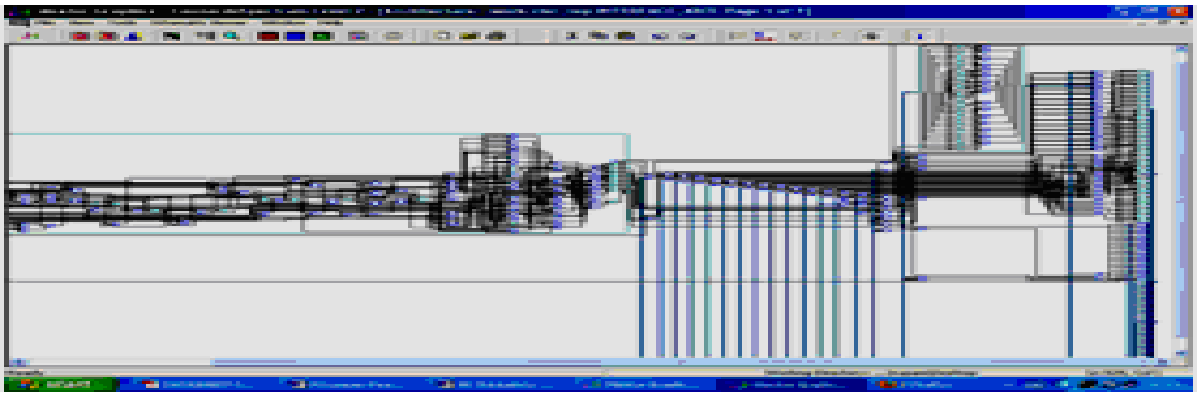

Fig. 5: *View of Gate Level Netlist.*

The Table I shows the critical path of different blocks of the processing unit, these are reported from resource file. This table also justifies the higher speed of the design .The critical path is low as compared to modern processor, so it has lower latency. After synthesis, the design was implemented using Xilinx ISE 9.1 (place and route). The device utilization is shown in Figure 6.

Table I: *Critical Path of Different Blocks of the Processor*.

Processor Block	Critical Path in
Top module	11.18
Control Unit	8.31
ALU	6.73
RAM	3.85
Programmed Counter	3.36
Shifter	5.82
Input ALU Register	5.88
Output ALU Register	4.20
Comparator	6.02
Address Register	5.88
Instruction Register	5.88
Tri-state Register	5.88

```
*********************************************
Device Utilization for 2V80cs144
*********************************************
Resource                Used    Avail    Utilization
---------------------------------------------
IOs                     49      92        53.26%
Global Buffers          1       16         6.25%
Function Generators     660     1024      64.45%
CLB Slices              330     512       64.45%
Dffs or Latches         372     1300      28.62%
Block RAMs              0       8          0.00%
Block Multipliers       0       8          0.00%

---------------------------------------------
Using wire table: xcv2-80-6_wc
```

Fig.6: *Device Utilization Report.*

5. ANALYSIS THE RESULTS

The instruction count for measurement the performance of the processor is the number of instructions execute during the run of a program, here processor CPI (Cycles per Instruction) are evaluated at different frequencies for various instructions which is shown in Table II. The performance of the processor depends on the processor time as well as the frequency in some factor. The processor time and performance is measured for operating at frequency 500 MHz (applied time period is 2 ns) from Eqn. (1) and Eqn. (2) [6].

***Table II**: Comparison between Frequency and Speed.*

Applied Time Period	Clock frequency	Execution time	CCT	CPI
200 ns	5 MHz	100 ns	100 ns	2
100 ns	10 MHz	50 ns	200 ns	2
20 ns	50 MHz	9 ns	40 ns	2
10 ns	100 MHz	4 ns	20 ns	2
5 ns	200 MHz	2 ns	10 ns	2
3 ns	333 MHz	2 ns	9 ns	3
2 ns	500 MHz	1 ns	6 ns	3
1.9 ns	526 MHz	0.995 ns	7.5	4
1.5 ns	666 MHz	0.5 ns	6 ns	4

CCT: Complete Cycles Time; CPI: Cycles per Instruction

Processor Time=Inst Count*CPI/Clk frequency..(1)

Performance =Clk frequency/(Inst*CPI)… (2)

The throughput and performance of the processor at 500 MHz is estimated here.

Throughput = $1 / (6 \times 10^{-9})$ IPS

= 0.1666×10^{9} instruction/second

= 167 MIPS

Processor Performance=$500 \times 10^{6}/(167 \times 10^{6} \times 3)$

= 500/501

= 0.99800

Its shows the performance of the processor is 99.8 percent, although it may decrease after hardware realization.

Here the design not considered for operating in higher frequency rather than the specified, because then it has the chance for over clocking problem [5,7]. So, lots of power can be lost as heat dissipation and overall device performance decreases for increasing processor time. The design can improve overall processor performance (i.e., reduce processor time) in a

way that increases the instruction count, by using instructions in that inner loop that may do less work per instruction. From the Table I, each instruction is finished in same time; this is the crucial factor for increasing the performance which is based on coding density and way of assignments of instruction in coding. Secondly such design is used to overcome the pipelining stall problem of modern processor. The capability of execution of instruction has 167 MIPS.

6. CONCLUSION AND FUTURE PROSPECTS

The actual performance of the RISC processor is determined after its hardware realization, although the paper presents the design for high throughput of 167 MIPS at 500 MHz. The processor operates in medium frequency range which overcomes the over clocking and pipelining stall problem. The design gives the higher performance as well as lead on trade-off between speed and power. The important point is that it's speed in MIPS with low power and reduced instruction time as according to the type of instructions.

The future works to be modified the design especially for space [9] and mobile applications [8].

REFERENCES

1. Paterson, David A. and Ditzel, D. R. *Computer Architecture News.* 1980. 8(6). 25–33p.
2. Perry, Douglas L. *VHDL Programming by Example.* McGraw Hill.
3. Badeshi, C., Mesa-Martinez, Francisco J. et al. *38th Annual IEEE/ACM International Symposium on Micro-architecture (MICRO'05)* 0-7695-2440-0/05 $20.00 © 2005.
4. Parhami, B. *Computer Architecture, from Microprocessor to Supercomputer*. Oxford University Press. 2005, ISBN 0-19-515455-X.
5. Rahman R., Othman. *ICSE2000 Proceedings*. Nov 2000.
6. Pani Saha and Chowdhury, *"Performance Enhancement in the Associative Processing of Floating Point Numbers Using Multi-core Superscalar Architecture" IEEE*. 2007
7. Habibullah Jamal, Shoab Khan, Fahed Hameed, et al. *An RTL Verilog Processor-IEEE 2003*.
8. Sohn, Woo, Yoo, et al. *Design and Test of Fixed-point Multimedia Co-processor for Mobile Applications* EDAA. 2006
9. Kimbrough, Collela, Denton, Clark. *IEEE Transactions on Nuclear Science*. 1994. 41. 6.