International Journal on Emerging Trends in Electronics & Communication Engineering
Vol. 4, Issue 2 – 2020
ISSN: 2581-558X

Design of 16-bit Pipelined RISC Processor

Ropali Singh, Vertika Mishra

Abstract

This paper presents the design of efficient and high throughput 16 bit pipelined RISC Processor. The design is carried out considering the pipeline 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 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 of Mentor Graphics. The presented design has throughput ~167 MIPS at 500 MHz, which shows better performance at a particular frequency.

Keywords: Processor, RISC, Pipelined and HDL.

Introduction

The increasing popularity of handheld items such as portable multimedia player, PDA and low power device application where processor is used, not the new concept for 21st century. The advancements of VLSI design technology together with demand of higher performance processor is the challenge to meet the balance between speed and power, pipeline 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].

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 Fig.1.

© Eureka Journals 2020. All Rights Reserved.

International Journal on Emerging Trends in Electronics & Communication Engineering
Vol. 4, Issue 2 – 2020
ISSN: 2581-558X

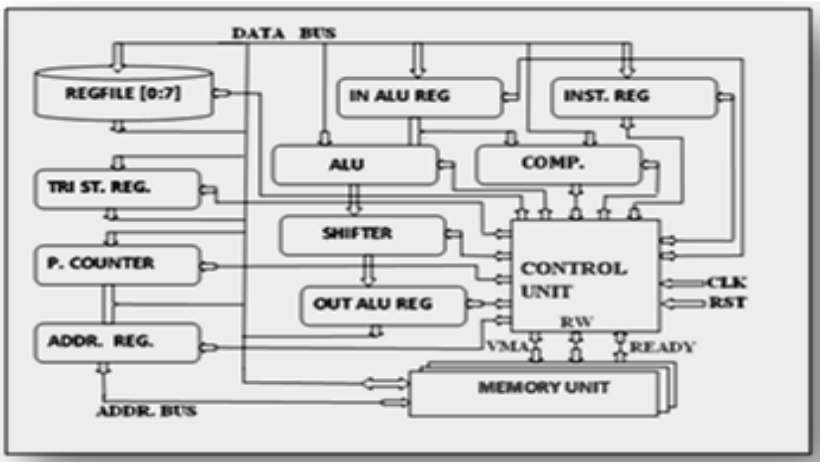

Figure 1.Processor block diagram

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 or branch instruction functions in different ways depending on what type of instruction the processor is executing. The fourth step only takes place to store word for storing computation result or write back. The entire steps include the load and store word instructions to access the memory and arithmetic-logical instructions to compute results.

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 calls all modules and its interfaces. The top modules simulation result verified through analysis. The coding is different from traditional design. There is an effort to avoid wait statement because it captures the extra silicon area which is the major issue for small devices. There are total 28 instructions, the data or address 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 on using this technique. When these blocks are required then an enable signal is used from the control unit to activate it. The instruction format is shown in fig.2. The left remaining bits are not used.

© Eureka Journals 2020. All Rights Reserved.

International Journal on Emerging Trends in Electronics & Communication Engineering
Vol. 4, Issue 2 – 2020
ISSN: 2581-558X

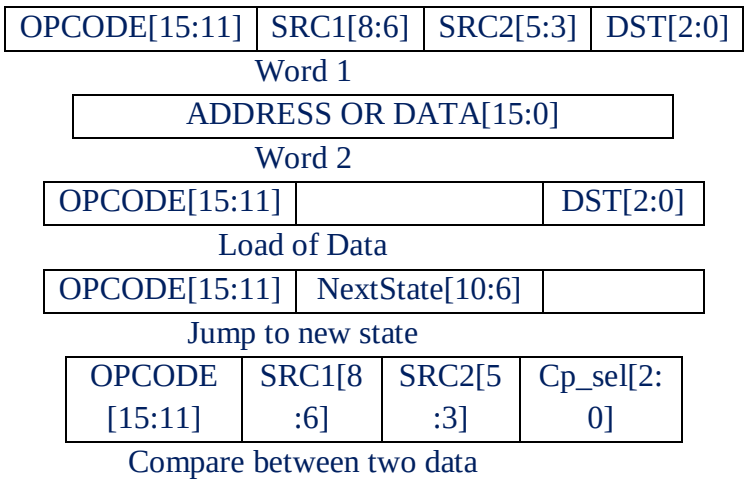

Figure 2.Instruction format

SRC: Source registers address

DST: Destination registers address

Here the design is based on pipelined architecture to speed up the processorbut branches are significant problem in the pipeline[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. Here for reducing branch penalties is 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, say

If (true) statement1; conditional instruction

If (false) statement2; conditional instruction

This is a sequence of 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, this implementation has several key advantages 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 steps also allows to significantly increase the clock cycle because no longer have to base the clock on the instruction that takes the longest to execute. The functionality is verified through writhing test bench to ensure the fully logical correctness of the design.

© Eureka Journals 2020. All Rights Reserved.

International Journal on Emerging Trends in Electronics & Communication Engineering
Vol. 4, Issue 2 – 2020
ISSN: 2581-558X

Simulation, synthesis and analysis the results

The instruction count in microprocessor performance measurement is the number of instructions executed during the run of a program, here process or CPI (Cycles per Instruction) are evaluated at different frequencies for various instructions. The instructions are tested in number of times and the results of few instructions are given in Table.1 for operating at frequency500MHz (applied time period is 2 ns).

Table 1.Verification result at 500 MHz

Instruction	Opcode	Reg1_s	Reg2_s	Reg_d	Execution time in ns	Complete cycle timens	Clock cycle	Remarks
Instruction Type: Arithmetic and Logical Instruction								
ADD	1101	10	1.1E+14	1.1E+14	1	6	3	ADD reg_d,reg1_s,reg2_s
INC	111	1.1E+13	-	1.1E+13	1	6	3	INCreg_d,reg1_s
DEC	1000	1.1E+13	-	1.1E+13	1	6	3	DECreg_d,reg1_s
XOR	1011	1E+12	1.1E+14	1.1E+14	1	6	3	XORreg_d,reg1_s,reg_s
AND	1001	1E+14	1.1E+14	1E+14	1	6	3	AND reg_d,reg1_s,reg_s
SUB	1110	1.1E+14	1.1E+14	100	1	6	3	SUB reg_d,reg1_s,reg_s
NOT	1100	1.1E+14		1E+15	1	6	3	NOT reg_d,reg1_s
OR	1010	1E+14	10	1E+14	1	6	3	OR reg_d,reg1_s
Instruction type: shifting operations								
SHL	11010	1E+14	-	1E+15	1	6	3	SHL Regd_s,reg1_s
SHR	11011	1.1E+15	-	1.1E+15	1	6	3	SHL Regd_s,reg1_s
ROL	11100	1.1E+15	-	1E+15	1	6	3	SHL Regd_s,reg1_s
ROR	11101	1E+14	-	1E+13	1	6	3	SHL Regd_s,reg1_s
Instruction type: Move instruction								
		Before move		After move				
		Reg1_s	Reg_d	Reg_d				
Move	11000	10	110	10	1	6	3	Move reg_d,reg1_s

- Complete cycle time : the time for opcode changes
- Reg1_s: operand 1 source register
- Reg2_s: operand 2 source register
- Reg_d : destination register
- Opcode: unique operation code
- CPI : Cycles Per Instruction

© Eureka Journals 2020. All Rights Reserved.

Processor Time=InstCount*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 x 10^{-9}) IPS

= 0.1666x10^{9} instruction/second

= 167 MIPS

Processor Performance = 500x10^{6}/(167x10^{6} x3)

= 500/501 = 0.99800

i.e. the performance of the processor is 99% , it may decrease after hardware realization.

Here the design considered not to operate in higher frequency rather than the specified, because then it has the chance for over clocking problem [5]. 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.1, 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 the instruction in coding. Secondly such design is used to overcome the pipelining stall problem of modern processor. The capability of execution of instruction has 167MIPS. The simulated waveform one part is shown in Fig.3.

The Table.2 is the CPI with respect to different frequencies. 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 from eqn.(1) and eqn.(2) [4].

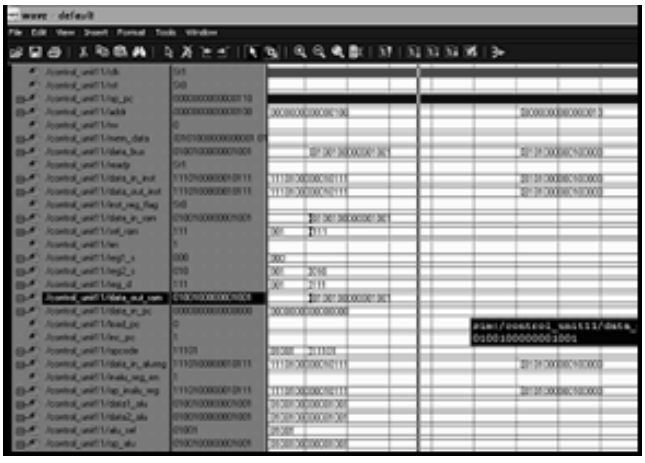

Figure3.Simulated waveform

© Eureka Journals 2020. All Rights Reserved.

International Journal on Emerging Trends in Electronics & Communication Engineering
Vol. 4, Issue 2 – 2020
ISSN: 2581-558X

Table 2.Comparison between frequency and speed

Applied Time	Clock	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 Seconds

The RTL code of the design is synthesized using FPGAXilinxVertexIIPro, device 2V40cs144 with speed grade -6. The whole architecture is mapped to generate the RTL and gate level netlist as shown in fig.4a and fig.4b.

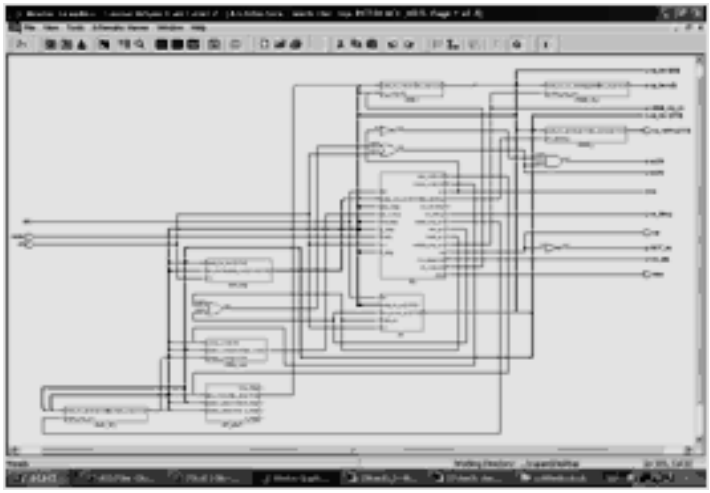

Figure 4a.View of RTL schematic

The Table.3 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 Fig.5.

© Eureka Journals 2020. All Rights Reserved.

International Journal on Emerging Trends in Electronics & Communication Engineering
Vol. 4, Issue 2 – 2020
ISSN: 2581-558X

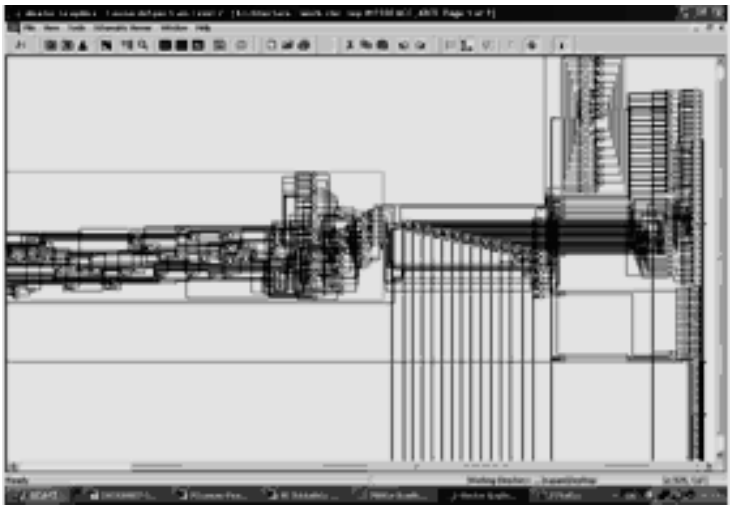

Figure 4b.View ofGate level netlist

Table 3.Critical Path of Different Blocks of the Processor

Processor Block	Critical Pathin ns
Top module	11.18
Control Unit	8.31
ALU	6.73
RAM	3.85
Programmed Counter	3.36
Shifter	5.82
InputALU 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
```

Figure 5.Device Utilization Report

© Eureka Journals 2020. All Rights Reserved.

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, 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 its speed in MIPS with low power and reduced instruction time as according to the type of instructions.

References

1. Paterson, David A and Ditzel, D. R. (1980) "The case for the reduced instruction set Computer," Computer Architecture News, 8(6) 25-33.
2. Perry, Douglas L. "VHDL Programming by Example" McGraw Hill.
3. Badeshi, Cyrusand Mesa-Martinez, Francisco J., Renau, Jose "μComplexity: Estimating Processor Design Effort" the 38th Annual IEEE/ACM International Symposium on Micro architecture (MICRO'05) 0-7695-2440-0/05 $20.00 © 2005.
4. Parhami, Behrooz "Computer Architecture, From microprocessor to Super computer". Oxford University Press, 2005, ISBN 0-19-515455-X.
5. Rahman, Rashid and Othman, "The PESONA16^{TM} RISC 16 bit microprocessor" ICSE2000 Proceedings, Nov2000.

© Eureka Journals 2020. All Rights Reserved.