RESEARCH ARTICLE OPEN ACCESS

Novel Architecture of 17 Bit Address RISC CPU with Pipelining Technique Using Xilinx in VLSI Technology

Rakesh M. R*, Ajeya B**, Mohan A.R***
*M. Tech student, Dept of ECE, Canara Engineering College, Mangalore, Karnataka, India
**Assistant professor, Dept. of ECE, Canara Engineering College, Mangalore, Karnataka, India
***Assistant professor, Dept. of ECE, Canara Engineering College, Mangalore, Karnataka, India

ABSTRACT
This paper describes the design of a 17 bit 4 stage pipelined Reduced Instruction Set Computer (RISC) processor using Verilog HDL in Xilinx. The processor implements the Harvard memory architecture, so the instruction and data memory spaces are both physically and logically separate. There is 5 bit opcode with totally 23 set of instructions. The CPU designed by using the pipelining and it will increase speed of processor with CPI=1. The pipeline stages such as fetch, decode, execute and store are used. The RISC processor architecture presented in this paper is designed by using Registers, arithmetic and logical unit, Memory with pipeline techniques. Memory access is done only by using Load and Store instructions.
Keywords: RISC features, Pipelining, ALU, Memory, Load and Store

I. INTRODUCTION

The Processor is a programmable device that takes in numbers, performs on them arithmetic or logical operations according to the program stored in memory and then produces other numbers as a Result. Processor is also an electronic circuit that functions as the central processing unit (CPU) of a computer, providing computational control.

Computers are common and important tools for everyday activities. With the simple design technology and the decreasing cost of the integrated circuit, RISC processor is increasing widely used in every field. The simple design provides higher performance, cost-effective, compatible systems. Some typical applications include: data processing, scientific and engineering applications and real-time control. In the present work, the design of a 4-bit data width Reduced Instruction Set Computer (RISC) processor is presented. It has a complete instruction set, program and data memories, general purpose registers and a simple Arithmetical Logical Unit (ALU) for basic operations.

At earlier days the RISC processor design is over using pipeline concept in embedded system field. But the FPGA having less average complexity and cost compared to the ASIC. Therefore in recent days research on the RISC processor design in VLSI is done.

Verilog HDL has evolved as a standard hardware description language. A hardware descriptive language is a language used to describe a digital system. It means that by using HDL one can describe any hardware at any level. HDL's allows the design to be simulated earlier in the design cycle in order to correct errors or experiment with different architectures. Designs described in HDL are technology-independent, easy to design and debug, and are usually more readable than schematics, particularly for large circuits. Verilog is capable of describing simple behaviour. Machine cycle instructions allow the processor to handle several instructions at the same time. The processor can work at a high clock frequency and thus yields higher speed. This paper is about design of a simple RISC processor and synthesizing it. The RISC architecture follows single-cycle instruction execution.

II. RISC PROCESSOR ARCHITECTURE

RISC architecture has been developed as a result of the 801 project which started in 1975 at the IBM T.J.Watson Research Center and was completed by the early 1980s. This project was not widely known to the world outside of IBM and two other projects with similar objectives started in the early 1980s at the University of California Berkeley and Stanford University. The term RISC used for the Berkeley research project.

1. RISC features

The RISC Processor works on (features are) reduced number of Instructions, Simple operations, only load and store operations access memory and Rest of the operations on a register-to-register basis, Pipelined, Simple uniform instructions, fixed instruction length, No microcode, Many identical general purpose registers, load-store architecture and simplified addressing modes which makes individual instructions execute faster, achieve a net gain in

performance and an overall simpler design. RISC architecture starts with a small set of most frequently used instructions which determine the pipeline structure of the machine enabling fast execution of those instructions in one cycle. Pipelining, a standard feature, is an implementation technique used to improve both CPI (Cycle per Instruction) and overall system performance. Basically the instructions are handled in parts: Instruction fetch: get instruction from memory and increment PC. Instruction decode: translate opcode into control signals and read registers. Instruction execute: perform ALU operation Store result: update register file. The block diagram of a RISC CPU is shown in Figure1 which have four stage of pipelining, memory, Register.

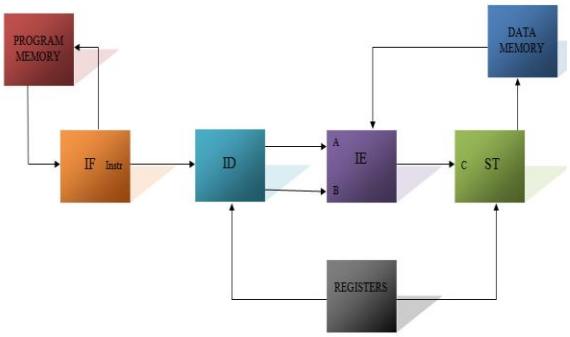

Fig1: Block diagram of Pipelined RISC processor

2. RISC & CISC comparison

Processors have traditionally been designed around two Philosophies: Complex Instruction Set Computer (CISC) and Reduced Instruction Set Computer (RISC).

The CISC concept is an approach to the Instruction Set Architecture (ISA) design that have wide variety of Addressing modes, the Instructions are of widely varying lengths and execution times thus demanding a very complex Control Unit, which tends to large chip area. On the other hand, the RISC Processor works on reduced number of Instructions, fixed Instruction length, more general-purpose registers, load-store architecture and simplified Addressing modes which makes individual instructions execute faster, achieve a net gain in performance and an overall simpler design with less silicon consumption as compared to CISC. The difference between these two RISC and CISC architecture is shown in Figure2.

CISC	RISC
Single register set	Multiple register set
Many instruction and modes	Few instruction and modes
Variable format instruction	Fixed format instruction
Microcode used	No microcode
Non pipelined	Highly pipelined
Any instruction may reference memory	Only load/store reference memory
Complex instruction taking multiple cycle	Simple instruction taking one cycle

Fig2: Difference between RISC and CISC architecture

3. Overall Operation

Instruction is fetched from physical memory, RAM. Program counter (PC) is used to fetch the address of the instruction. This register is visible to Control unit of CPU but not visible to programmer. The instruction is copied to Instruction Register from memory. If the instruction is ADD is considered, the addition operation is performed on corresponding registers. This is performed by the unit ALU. For RISC machine, memory access occurs for the load and store instructions. For the other instructions no operation is performed by ALU. Here data memory is accessed. The result of the operation being performed by ALU is written to appropriate register in the register file.

4. Pipelining

Pipelining, a standard feature, is an implementation technique used to improve both CPI (Cycle per Instruction) and overall system performance. Pipelining allows a processor to work on different steps of the instruction at the same time, thus more instruction can be executed in a shorter period of time. The sole purpose of many of those features is to support an efficient execution of RISC pipeline. It is clear that without pipelining the goal of CPI = 1 is not possible.

Clocks per instruction (CPI):

CPI is an effective average. It is averaged over all of the instruction executions in a program. CPI is affected by instruction-level parallelism and by instruction complexity. Without instruction-level parallelism, simple instructions usually take 4 or more cycles to execute. Instructions that execute loops take at least one clock per loop iteration. Pipelining (overlapping execution of instructions) can bring the average for simple instructions down to near 1 clock per instruction. This designed architecture will tend towards CPI=1.

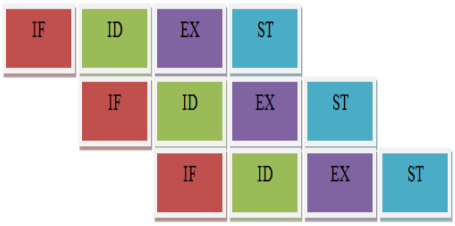

Fig3: instruction execution using Pipelining technique

The Figure3 shows how instructions are executed using the pipelining technique. In this paper four stage of pipeline techniques are used such as Fetch, Decode, Execute, and Store result.

Instruction Fetch Stage (IF):

In this stage, the content of the Program Counter is used to access memory and fetch the next instruction to be executed.

Instruction Decode Stage (ID):

During this stage, the instruction is decoded and the required operands are retrieved from the general purpose registers (GPRs).

Execute Stage (EX):

Any calculations are performed during this stage. This includes effective address calculation for Load or Store instructions. The next Program Counter value is also calculated during this stage of the pipeline so that branches, where applicable, can be executed.

Store result (ST):

During this stage, the results of the calculation from the Execute stage, or the memory load from the Memory Access stage, are updated into the general purpose registers.

5. Register File

The register file consists of general purpose registers of 8-bits capacity. These register files are utilized during the execution of arithmetic and data-centric instructions. It is fully visible to the programmer. The load instruction is used to load the values into the registers and store instruction is used to retrieve the values back to the memory to obtain the processed outputs back from the processor. The Link register is used to hold the addresses of the corresponding memory locations.

Program Counter*:*

The program counter (PC) contains the address of the instruction that will be fetched from the Instruction memory during the next clock cycle. 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 proposed design in this paper consists of 8 bit program counter which are incremented for every clock cycles.

The Status Register

The status register is an 8-bit register. Primarily it is updated, however, as a side effect of the execution of ALU operations. If, for example, an arithmetic/logic instruction produces a result of 0, then the zero flag (the second bit in the status register) is set. An arithmetic overflow in an ADD instruction causes the carry bit to be set, and so on. The proposed design consists of 8 bit status register which stores the carry bit which is generated by execution unit operations'.

6. Arithmetic Logic Unit

The ALU performs arithmetic and logical operations on data. The data is taken from two GPRs and is moved to the ALU. The result is stored in a GPR. The ALU has the 8-bit inputs A, B and the output C. The ALU takes two operands from the A and B registers. The result register C is used to hold the ALU output. The ALU has the capability to perform many operations. After every ALU instruction, the output register is updated.

7. Separate Data Memory and Instruction Memory access paths

Different stages of the pipeline perform simultaneous accesses to memory. This Harvard style of architecture can either be used with two completely different memory spaces.

Program Memory is implemented as a ROM. Its content can be preloaded from the file (containing instructions encoded in hexadecimal form). In this paper program memory consists of 23 set of instructions which is accessed by CPU during operation.

Data Memory is implemented as RAM. Memory model pre-loads initial data from the Reg file and dumps its content into file after the simulation is finished.

8. Design of the ROM

The CPU has a built in ROM which enables us to program simple code and execute it. The List of signals in the ROM is: 1. Address: address sent by the control unit. 2. Data out: the data that is contained the given address. 3. Clk: the main clock signal. 6. Reset: the initial reset signal.ROM consists of 23 set of instructions.

9. Instruction set and instruction formats

The various instructions performed by the processor such as: Arithmetic, Logical, data processing, data movement, branch, shift.

Branch instructions:

Branch instructions are used to break the sequence of instructions. A branch instruction requires two pieces of information that the processor must determine before executing the instruction, whether the branch is to be taken and the branch target address. The one basic types of braches in a this processor are conditional (such as decisions). In this designed CPU five types of branch instructions are taken for process.

Load/Store machine instructions:

If the instruction is a load, then read from memory using the effective address computed in the previous cycle. If it is a store, then the data from the second register read from the register file is written into memory using the effective address.

RISC Processor Instruction formats: Register.

Addressing modes are aspects of the instruction set architecture in most central processing unit (CPU) designs. The different addressing modes in an instruction set architecture define how machine language instructions in that architecture identify the operand (or operands) of each instruction. The addressing mode used in proposed method is: Register addressing mode. 17 bit address with opcode of 5bit and three operands each of 4 bits. The data width, status register, program counter are of 8 bit. 23 instructions are used in this processor and taken in ROM memory of this processor.

The instruction format is of register type and is shown in Figure4. Instruction of 17 bit with opcode 5bit and three operands of each 4 bit.

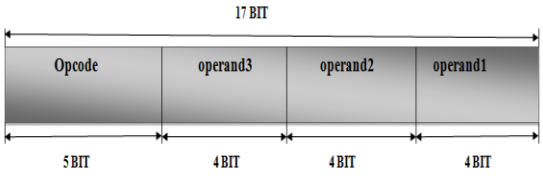

Fig4: instruction format for sinle RISC processor

The opcode and corresponding operation is shown in Figure5. The opcode of 5 bit starts from 00000 to 10110 is considered in this design.

OPCODE	OPERATION	
00000	ADD	Addition of two register
00001	SUB	Subtraction of two register
00010	MUL	Multiplication of two register
00011	DIV	Division of two register
00100	INC	Increment of two register
00101	DEC	Decrement of two register
00110	SR	Register value Shift right
00111	SL	Register value shift left
01000	SRC	Register value Shift right with carry
01001	SLC	Register value Shift left with carry
01010	OR	or operation on two register
01011	AND	and operation on two register
01100	XOR	xor operation on two register
01101	NOT	not operation register
01110	MVI	Move immediate
01111	MOV	Register moving
10000	LD	Load from memory to register
10001	ST	store to memory from register
10010	BZ	Branch occurs if value zero
10011	BNZ	Branch occurs if value not zero
10100	BC	Branch occurs if it has carry
10101	BNC	Branch occurs if it has no carry
10110	GOTO	Go to specified address

Fig5: Instruction set of Processor

The power analysis for the proposed CPU architecture design is done by using XPower analyzer. The clock frequency is taken as 100MHZ. The Figure6 shows the some specification for power analysis.

Device family	Vertex 6
Ambient temperature	50.0 C
Airflow	250 LFM
Clock frequency	100MHZ

Fig6: specification for power analyzer

III. SIMULATION RESULTS

The single RISC processor or CPU is simulated by using pipeline concept and Xilinx ISE and Modelsim simulator is used to get results. The Verilog HDL is used for designing the CPU. The technology schematic for single RISC processor obtained in Xilinx is shown in Figure7. The RTL schematic for single RISC processor obtained in Xilinx is shown in Figure8.

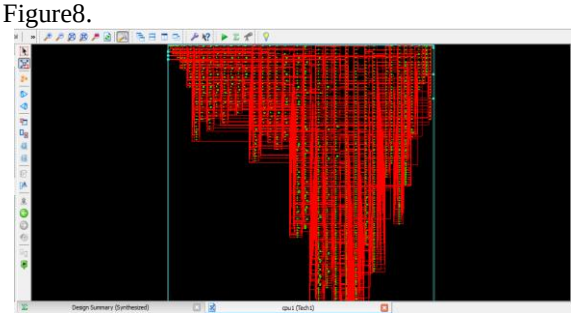

Fig7: Technological schematic of single RISC processor

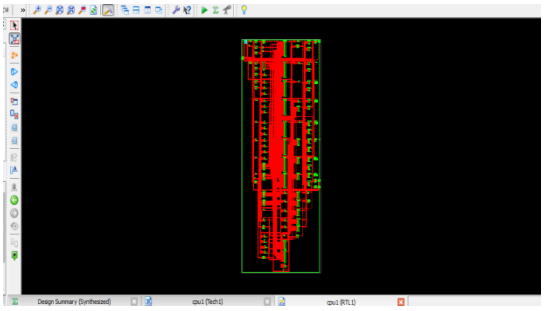

Fig8: RTL schematic of single RISC processor

The simulation waveform for Single CPU is obtained by using Modelsim is shown in Figure9. Here there is a five input such as clock (1bit), reset (1bit), input1 (8 bit), input2 (8 bit), ram data (8 bit). The data read from ram by using load instruction. The output are op (8 bit), ram write (8 bit).

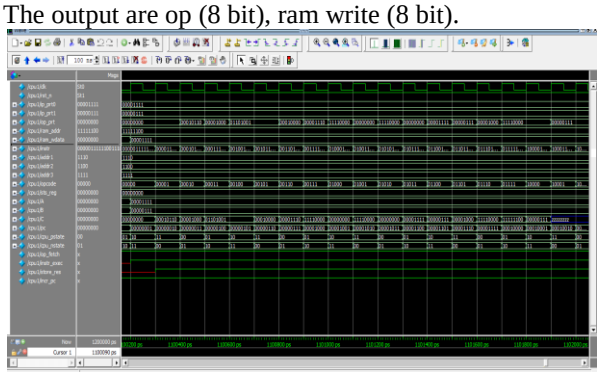

Fig9: Simulation waveform for single RISC processor

The design summary shows how much used from total amount and it shown in terms of numbers. Utilization of different parts in terms of percentage is shown in Figure10.

Logic utilization	Used	Available	Utilization
Number of slice registers	56	93120	0%
Number of slice LUT's	203	46560	0%
Number of fully use LUT-FF pairs	43	216	19%
Number of bonded IOBs	51	240	21%
Number of block RAM/FIFO	1	156	0%
Number of BUFG/BUFGCTRLs	2	32	6%
Number of DSP48E1s	1	288	0%

Fig10: Design summary of CPU

The Figure11 shows the power analysis report for proposed architecture of CPU and it can be getting by using XPower analyzer in Xilinx. It shows how much power each part in our design will be going to use.

On chip	Power	used	Available	Utilization
Clock	0.009	7	-	-
Logic	0.000	191	46560	0
Signal	0.001	289	-	-
BRAMs	0.001	*	*	*
DSPs	0.000	1	288	0
IOs	0.011	51	240	21
Leakage	1.296			
Total	1.318			

Fig11: Results obtained during power analysis power analysis

IV. ADVANTAGES & DISADVANTAGES

Advantages of the processor are following: It has been designed to perform a small set Instructions, with the aim of increasing the overall speed of the processor. The design is an attempt to reduce the instruction time by simplifying the instruction set of the processor. Disadvantages of the processor are following: It is used were simple operations are required for processing data; complex operations like floating point addition etc. are not performed.

V. CONCLUSION

17 bit address RISC Processor core has been design and simulated in Xilinx ise13.1. The design has been achieved using Verilog and simulated with Modelsim simulator. Most of the goals were achieved and simulation shows that the processor is working perfectly, Future work will be added by increasing the number of instructions and make a pipelined design with less clock cycles per instruction and more improvement can be added in the future work.

REFERENCES

[1] R. Uma, "Design and Performance Analysis of 8-bit RISC Processor using Xilinx Tool", *International Journal of Engineering Research and Applications (IJERA)*, Vol. 2, Issue 2, Mar-Apr 2012, pp.053-058

[2] 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)*, Vol. 5, Issue. 1, July-2013.

[3] Galani Tina R.D.Daruwala, February 2013, " Performance Improvement of MIPS Architecture by Adding New Features" Tina et al., *International Journal of Advanced Research in Computer Science and Software Engineering 3(2)*, February - 2013, pp. 1-6

[4] Samiappa Sakthikumaran,S.Salivahanan and V.S.Kaanchana Bhaaskaran , "16-Bit RISC Processor Design For Convolution Application", *IEEE International Conference on Recent Trends In Information Technology*, pp.394-397, June 2011.

[5] Rohit Sharma, Vivek Kumar Sehgal, Nitin Nitin1, Pranav Bhasker, Ishita Verma , 2009, "Design And Implementation Of 64-Bit RISC Processor Using VHDL", *UKSim : 11th International Conference on Computer Modeling And Simulation*, pp. 568 – 573.

[6] Rupali S. Balpande and Rashmi S. Keote.2011, "Design of FPGA based Instruction Fetch & Decode Module of 32-bit RISC (MIPS) Processor", *International Conference on Communication Systems and Network Technologies*, pp. 409 – 413

[7] Xiao Li, Longwei Ji, Bo Shen, Wenhong Li, Qianling Zhang, "VLSI implementation of a High-performance 32-bit RISC Microprocessor", *IEEE 2002 International Conference on Communications, Circuits*

and Systems and West Sino Expositions, Volume 2, 2002 ,pp.1458 – 1461.

BIOGRAPHIES

Mr. Rakesh M.R received his B.E degree in Electronics and Communication from KVG College of Engineering Sullia in 2012. Currently he is pursuing M.Tech degree in Electronics at Canara Engineering College, Mangalore. His areas of interest are VLSI and Image Processing.

Mr. Ajeya B. Obtained his M.Tech degree in Digital Electronics and Communication from NMAM Institute of technology, Nitte in 2010 and B.E. degree from UBDT College of Engineering, Davangere in 2007. Currently he is working as an Asst. Professor in the Department of E&C, Canara Engineering College, Mangalore. His areas of interest include Wireless Sensor Networks, Cryptography and Digital Communication.

Mr. Mohan A.R received his M.Tech degree in VLSI Design and Embedded Systems from SJB Institute of technology, Bangalore in 2011 and B.E. degree from Coorg Institute of Technology, Ponnampet in 2007. Currently he is working as an Asst. Professor in the Department of E&C, Canara Engineering College, Mangalore. His area of interest is Digital VLSI Design.