IMPLEMENTATION AND VERIFICATION OF RISC PROCESSOR ON FPGA USING CHIPSCOPE PRO TOOL

[1]Prof Ruckmani Divakaran, [2]Srinivas Babu .N, [3]Shashi Kiran .S, [4]Byrareddy .H.C
[1] Head of Department, Department of Electronics and Communication, Dr. TTIT, KGF, hod.ece@drttit.edu.in
[2] Assistant Professor, Department of Electronics and Communication, Dr. TTIT, KGF, srinivas.b@drttit.edu.in
[3] Assistant Professor, Department of Electronics and Communication, Dr. TTIT, KGF, shashikiran@drttit.edu.in
[4] PG Scholar, M.Tech in Digital Communication and Networking, Dr. TTIT, KGF, byra22@gmail.com

Abstract

The advanced microprocessors are widely used for most of the complex systems. A silicon chip of fingernail-size may exhibit entire high performance guaranteed processor, higher cache memory and logic needed for interfacing with external devices. Reduced Instruction Set Computing (RISC) is a CPU (Central Processing Unit) design mechanism based on the vision in which exhibits basic instruction set and yields better performance after comparison with microprocessor architecture and it has the capacity to perform the instructions through microprocessor cycles per instruction. In this paper, the Cost-effective and efficient RISC Processor is designed. The RISC Processor design includes Fetching, decoding, Data and instruction memory, and Execution units. The Execution unit contains ALU (Arthematic and Logical Unit) Operations. The RISC Processor design is synthesized and implemented using Xilinx ISE Tool and simulated using Modelsim6.5f. The implementation is done by Artix-7 FPGA device and the physically debugging of the RISC Processor, and ALU Units are verified using Chipscope pro tool. The performance results are analyzed in terms of the Area (Slices, LUT's), Timing period, and Maximum operating frequency. The comparison of the RISC Processor is made concerning previous similar architecture with improvements.

Key words: RISC Processor, ALU Unit, Execution, Fetching, decoding, Verification, FPGA

I. INTRODUCTION

The earlier days of the processor design has witnessed a quest for higher performance in computer models and architectures. To achieve significant performance, technology advantages, better architecture and optimization in the compiler technology. As per this technology, the machine performance can be increased in proportion with the technology enhancement which can be available for everyone.

The design of the processor is manufactures using semiconductor devices, the printed circuit board (PCB), etc. The operation of any processor depends on the instructions used in it. These instructions include the computation/manipulation of the data values by using the registers, changing/retrieve the values of the read/write memory, performing the relational test among the data values and to have the control over the program flow.

The design of a processor considers the areas like: (i) data paths like 'Arithmetic Logic-Unit (ALU) and pipelines', (ii) a control unit which helps in controlling the data paths, (iii) considers the register files (memory components), (iv) clock circuits, (v) library of

logic gates for logic implementation and (vii) includes transceiver circuits is represented in figure 1.

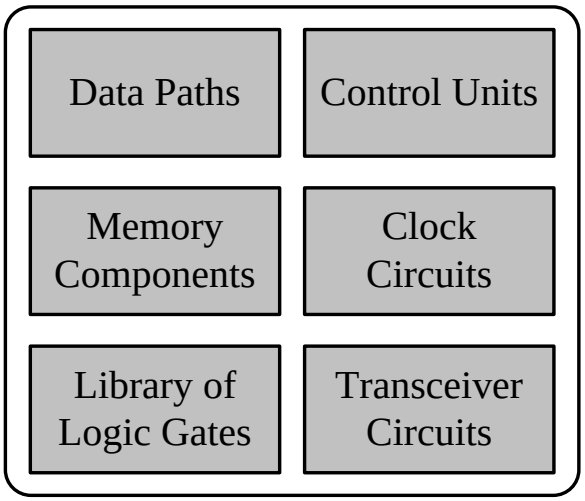

Figure.1 Design areas of processor

The processor's designs for high performance demanding applications require the custom designs in above-stated items to get power dissipation, frequency, etc., while for low performance demanding applications uses the less number of items.

The RISC architecture deals with more precisely their trade-offs and interaction [1]. The working function of instructions in RISC is simple. Hence, the execution time required for each installation can be minimized, and a number of cycles can be narrowed. The execution time for instruction can be divided into machine cycles, machine processing, and operation of instruction sets. The operation of the instruction can be performed in a pipelined manner [2, 3].

The pipelined process includes five different stages, i.e., instruction fetching (IF), Instruction decoding (ID), execution (EX), memory accessing (MA) and write back (WB). The overlapping of different instruction in a pipelined manner. RISC performs its inherent parallel execution which is responsible for performance enhancement than CISC. The RISC aims to achieve execution of single cycle per instruction, i.e., CPI = 1.0 with no interruption where pipelining take place. The selection of addressing modes and instruction in RISC can be performed and tailored on the basis of recently used instruction resulting in significant execution of RISC pipeline [4, 5].

The section 2 discuss about the existing works of RISC Processor and ALU Designs and Problems finding. The section 3 explains about the proposed RISC Processor with detailed descriptions. The results and analysis of the work is elaborated in section 4. Finally concludes the overall system with improvements with future work in section 5.

II. RELATED WORKS

The review of the existing work on RISC Processor architectures and ALU Units are described in below session.

The implementation of field programmable gate array (FPGA) is successively has been done in 8-bite Reduced instruction set Gal et al. discussed computer (RISC) [6]. The introduction of (FPGA) implementation and working experience of the 8-bit microcontroller is to provide in this given work. The uses of a microcontroller are on a big demand in an industrial application; the primary goal of the implementation in RISC is to decrease the number of instruction and produced a smaller and faster processor.

Nowadays technologies are entered in human beings dally life and demand for machine and computer technology is increased day by day. For the full fill of demand Khazaee et al. [7] have presented a Java programming language as Java virtual machine (JVM), but the given JVM is not enough capacity in speed or storage. The problem of low speed and memory overhead is carried out by Tomar et al. [8] in which discussed the working progress of RISC with the connecting of the digital signal processor (DSP) it can perform effectively in several operations.

Cernazanu et al. [9] have discussed the possibility of generating an energy profile of RISC by the implementation of FPGA. Energy is consumed by the following of group instruction like memory access instruction, arithmetic and logic instruction (ALU), compare and move instruction; these are the more energy consumed instruction. The work of power consumption is carried out in the next section by Murthy et al. [10] have focused on the power consumption of the 32-bit RISC core processor. For solving this more power consumption problem a DLX (32-bit

architecture). Power consumption and power management work are continues in the next segment, in which Kumar et al.[11] have focused on the capability of dynamic power management and to maintain the consumption proposed a low power embedded system. The development of 8-bit RISC processor is possible by the implementation of HDL on the board of FPGA. Jeemon et al. [12] have presented a pipeline RISC processor for the improvement of the performance of the 8-bit RISC processor. The process of a bit is continued in a change form of 32-bit, Singh et al. [13] have discussed on the design of 32-bit re-configurable RISC processor this processor is based on BETA (lesson-by-lesson) instruction. The main reason behind the using of a 32-bit processor is that the 32-bit processor is to provide flexibility and a great extent to the developer.

Work of the 32-bit processor is carried out by Dennis et al. [14] have presented an improvement of the fully synthesizable 32-bit processor, and it is based on RISC-V. The primary goal behind this proposed technique is to make a low cost embedded device and also completed the verification and assembling, testing all are at a limited cost. Rohit et al. [15] have presented a low power RISC processor with the using of stalling and forwarding process. A base of the architecture of the RISC processor is based on million instruction per second (MIPS) it is a non-interlocked pipeline method. The work of Venkategowada et al. [16] have concentrated on the plan of creating a synthesized shared memory and QUAD-Core processor and put it on a working path to solve the problem of processor verification, and multiprocessor interrupts management. The primary goal of this given technique is to improve a framework and a test QUAD-Core processor on an FPGA board.

Problem statement:

Most of the Existing processor designs are software based and not compactable to real-time environments. In that, very few are hardware-based approaches and lacks hardware complexities and performance metrics in real time. Most of the Processor designs are ASIC Based approaches and limited to perform only a few parameters. Very few processor verifications are done on real-time environments and lack optimizations. In order to verify the hardware designs with real-time data, it is difficult without using the simulation testbench. The Unit under Test (UUT) is the design module is incorporated in the testbench to verify the designs. But it lacks with observability and controllability in real time.

Proposed Solution:

The proposed RISC Processor overcomes all the problems and improves processor performance. The processor Verification is done quickly by using two intellectual property (IP) cores. In that, the integrated logic Analyzer (ILA) improves the observability, and virtual input-output core (VIO) improves the controllability of the processor in real-time verification.

III. PROPOSED RISC PROCESSOR

This section describes about the proposed RISC Processor and its working functionality. The RISC Processor components have explained an overview in the below section:

- **Instruction Memory:** It is the part of a control unit that stores the instructions and will be executed or decoded.
- **Instruction Fetch:** It is loading of *instruction* or piece of data from memory into a CPU's register. All *instructions* must be *fetched* before they can be executed.
- **Instruction Decode:** what the instruction has to do, it has to be decoded. Part of the decoding method fetches the input-operands.
- **Data Register:** *Data register* is an array of processor-*registers* in a 'CPU.' Modern integrated-circuit based *Data register* is usually implemented by fast static 'RAMs' with multiple-ports.
- **Execute Logic (ALU Design):** It is the block where all the Arthematic and logical expressions are made based on the Mode.

ALU Design mainly includes Arthematic and logic operations. The Arthematic operation includes Addition, subtraction, Multiplication, division, Modulus, Increment, and decrement operations. The Logic Operations includes Logical AND, OR, XOR, NAND, NOR, XNOR, Right shift, left shift, negation and don't care operations.

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-6, ISSUE-6, 2019
DOI: 10.21276/ijcesr.2019.6.6.12

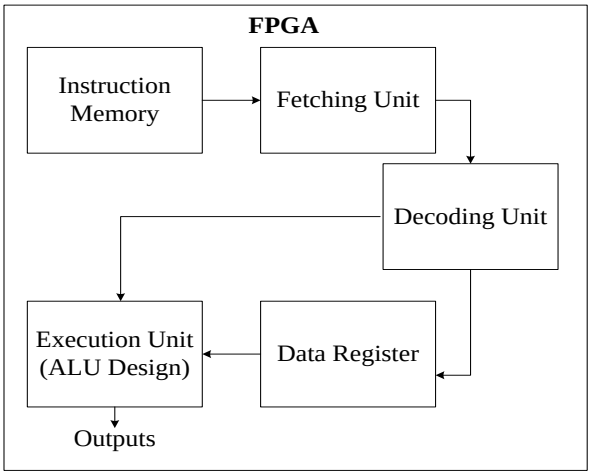

Figure 2: Block Diagram of ALU Design for RISC Processor

Instruction fetching (IF) unit is used to fetch the instruction from the instruction memory and pass to the decoder unit. The fetching unit mainly contains a program counter (PC) and instruction memory (IM) unit. The instruction memory receives PC Output as an input. The IM unit is having 16 memory locations (ROM), and each memory location is 16-bit. The PC Output acts as an address generator to the IM unit. The IM Unit contains mainly 16 different values which are stored in 16 memory locations.

The instruction decoding (ID) unit decodes the fetched instruction operands. The ID unit receives the fetching output as a 16-bit input and decodes into last 4-bit instruction [15:12] are acts as an operand. Based on the operands, generates the opcode for ALU unit (alu_in) and registers (Rdx and Rdy) for data memory. If the 4-bit instruction [15:12] is set to 4'b0000, the alu_in is set 4'b0000 and Rdx= 4'd0 and Rdy=4'b0 and If the 4-bit instruction [15:12] is set to 4'b0001, the alu_in is set 4'b0001 and Rdx= instruction [7:4] and Rdy= instruction [3:0] and similarly the decoding process continue till the last operand (i.e. 15th). The Data register or Data memory (DM) unit is used to read and store the user data according to the instruction operands. The DM Unit has 16 memory locations (a); each is memory location is 16-bit. The 4-bit Rdx and Rdy are acted as an address generator and inputs to memory (a) to read the user data. And store it in 16-bit registers (Rx and Ry) as an output.

Table 1: Execution (ALU) unit Operations

Sl.No.	ALU Input	Operation	Output Function
0	0000	Addition	Rx + Ry
1	0001	Subtraction	Rx - Ry
2	0010	Multiplication	Rx * Ry
3	0011	Division	Rx / Ry
4	0100	Modulus	Rx % Ry
5	0101	Increment	Rx + 1
6	0110	Decrement	Rx - 1
7	0111	Bitwise AND	Rx & Ry
8	1000	Bitwise OR	Rx | Ry
9	1001	Bitwise Negation	~Rx
10	1010	Bitwise XOR	Rx ^ Ry
11	1011	Bitwise XNOR	Rx ~^ Ry
12	1100	Bitwise NAND	~(Rx & Ry)
13	1101	Bitwise NOR	~(Rx | Ry)
14	1110	Logical Shift Right	Rx >> Ry
15	1111	Logical Shift Left	Rx << Ry

The central processing unit of the RISC Processor is Execution unit, i.e., ALU Unit. It performs the Arthematic and logic operation based on the instruction opcode and instruction registers. The Execution (ALU) contains clock (Clk), reset (rst), 16-bit registers Rx and Ry as an input and generates the 32-bit ALU outputs. Based on the ALU operations, the ALU output will be generated. The ALU Operations includes Addition, subtraction, multiplication, Division, Modulus, Increment, decrement, Bitwise AND, OR, Negation, XOR, XNOR, NAND, NOR, logical right and left shift operations are tabulated in table 4.3 with output functions. Based on ALU operations, the Rx and Ry inputs performs the function and generates the 32-bit ALU outputs is shown in the table 1.

IV. RISC PROCESSOR IMPLEMENTATION

The RISC Processor design is programmed on FPGA Chip. The physical verification will be checking using Chipscope pro tool. The Chipscope pro tool is hardware debugging method in real time environment with FPGA Design with the help of debugging verification

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-6, ISSUE-6, 2019
DOI: 10.21276/ijcesr.2019.6.6.12

IP Cores which includes ICON (Integrated controller), and Integrated Logic analyzer (ILA) is in figure 3.

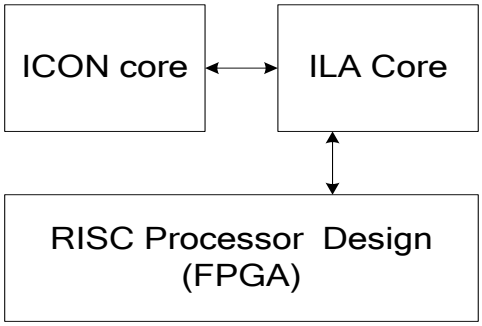

Figure 3: Physical verification of RISC Processor

The ICON is an IP core which provides the communication path between ILA and RISC Processor (FPGA) Design using JTAG Boundary scan port of the targeted Artix-7 FPGA. After dumping the RISC Processor module on FPGA, The program will be succeeded. Use the Chipscope pro tool to analyse the outputs on the monitor screen with hardware control.

V. RESULTS AND ANALYSIS

The proposed RISC processor results are described detail in the below section. The Complete RISC processor is designed using Verilog HDL over Xilinx ISE Platform and simulated on Modelsim simulator and Hardware prototyped on low cost Artix-7 FPGA.

The RISC Processor Simulation Results are a representation shown in figure 4. Once clock (Clk) is activated with low reset, The 8-bit pc_in is set to zero (00-Hex value). Outputs are 16-bit Register-X (Rx), and Register-Y (Ry) is generated from the data memory and also generates the 32-bit ALU outputs after one clock cycles, Based on the ALU operations.

The Chipscope-pro-tool analyzed results for RISC Processor–when reset=1 is represented in figure 5. The ILA display only the output signals of the RISC processor. The Chipscope-pro-tool analyzed results for RISC Processor–when reset=0 and for addition is represented in the figure 6. When the alu_in=0001, Rx=000F, Ry=000F, The ALU output is 32-bit 0000_001E.

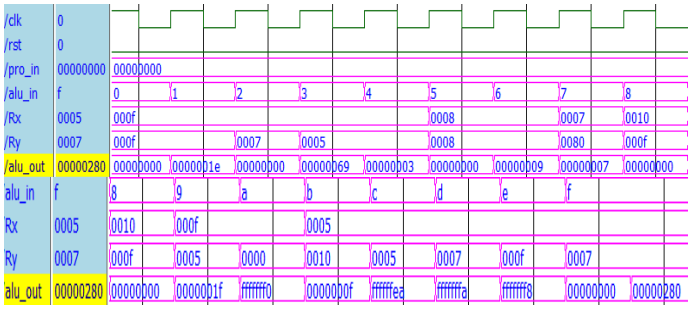

Figure 4: RISC Processor Simulation Results

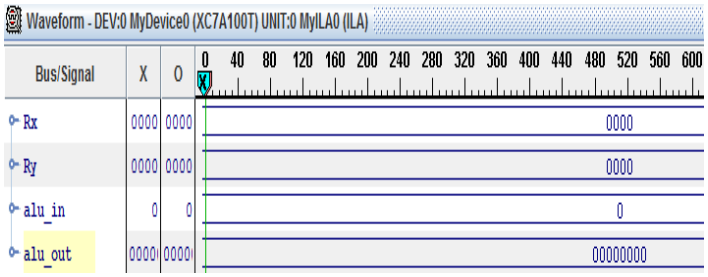

Figure 5: FPGA –Chipscope-pro-tool RISC Processor-Results – when reset=1

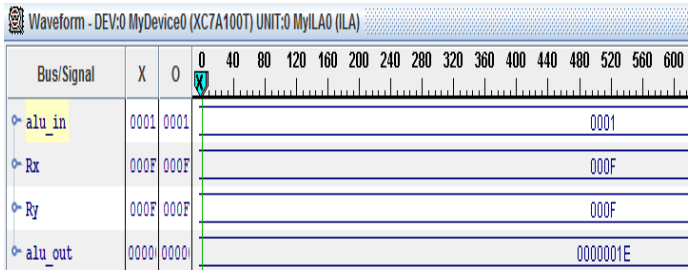

Figure 6: FPGA –Chipscope-pro-tool RISC Processor-Results for Addition

The proposed RISC Processor is compared with similar previous RISC Processor unit [11] concerning resource utilization on the same selected Virtex-6 FPGA. The improvements in slice registers around 82.03%, slice LUT's around 4.55%, LUT-FF pairs are 66.66%, and Input of Input-Outputs is around 55.93% is tabulated in table 2.

Table 2: Comparison of RISC Processor with previous [11]

Resource Utilization	Our RISC Processor	Previous [11]	Overhead
Number of Slice Registers	90	501	82.03%
Number of Slice LUTs	984	1031	4.55%
Number of fully used LUT-FF pairs	64	425	84.95%
Number of bonded IOBs	78	177	55.93%
Number of BUFG/BUFGCTRLs	1	3	66.66%

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-6, ISSUE-6, 2019
DOI: 10.21276/ijcesr.2019.6.6.12

Overall the RISC Processor and ALU Unit is efficient and consumes less resource utilization than previous architectures.

VI. CONCLUSION

The cost-effective and straightforward RSIC Processor is designed using Verilog-HDL and implemented on the Artix-7 FPGA Platform. The RSIC mainly contains fetching unit, data, and instruction memory units, decoding units and execution unit which includes ALU operations. The Hardware architecture of RISC Processor using ICON IP Core and ALU Unit using VIO IP cores are designed. The simulation results of the RISC Processor is seen using Modelsim 6.5f. The RISC Processor using ICON IP Core and ALU Unit using VIO IP cores implemented on Artix-7 FPGA with physically debugging using Chipscope pro tool which improves the observability and controllability respectively. The RISC Processor is synthesized and implemented using Xilinx ISE Tool. The RISC Processor is compared with previous similar architecture on the same FPGA devices with an Improvement in area overhead of slice registers 82.03%, slice LUT's 4.55%, LUT-FF pairs 66.66% than previous RISC Processor. In Future, for the same RISC Processor, add some complex instructions to perform the Digital signal processing operations.

REFERENCES

[1] G.M.Amdahl, G.A. Blaauw, F.P. Brooks, "Architecture of the IBM System/360, IBM Journal of Research and Development, Vol.8, No.2, p.87-101, April 1964.

[2] G.A. Blaauw, F.P. Brooks, "The Structure of System/360", IBM Systems Journal, Vol.3, No.2, p.119- 135, 1964.

[3] R.P.Case, A.Padegs, "Architecture of the IBM System/370", Communications of ACM, Vol.21, No.1, p. 73-96, January 1978.

[4] G. Radin, "The 801 Minicomputer", IBM T.J.Watson Research Center, Report RC 9125, November 11, 1981, also in SIGARCH Computer Architecture News 10, No.2, p.39-47, March 1982.

[5] D.A. Patterson, C.H.Sequin, "A VLSI RISC", IEEE Computer Magazine, September 1982.

[6] Gal, Ryszard, et al. "FPGA implementation of 8-bit RISC microcontroller for embedded systems." Proceedings of the 18th International Conference Mixed Design of Integrated Circuits and Systems-MIXDES 2011. IEEE, 2011.

[7] Khazaee, Mohammad Irfan, and Shima Hoseinzadeh. "Using Java optimized processor as an intellectual property core beside a RISC processor in FPGA." Proceedings of IEEE East-West Design & Test Symposium (EWDTS 2014). IEEE, 2014.

[8] Tomar, Amit Kumar Singh, and Rita Jain. "20-Bit RISC and DSP System Design in an FPGA." Computing in Science & Engineering 16.2 (2014): 16-20.

[9] Cernazanu-Glavan, Cosmin, et al. "Direct FPGA-based power profiling for a RISC processor." 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings. IEEE, 2015.

[10] Murthy, Soumya, and Usha Verma. "FPGA Based Implementation of Power Optimization of 32 Bit RISC Core Using DLX Architecture." 2015 International Conference on Computing Communication Control and Automation. IEEE, 2015.

[11] Kumar, Narender, and Munish Rattan. "Implementation of embedded RISC processor with dynamic power management for low-power embedded system on SOC." 2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS). IEEE, 2015.

[12] Jeemon, Jikku. "Pipelined 8-bit RISC processor design using Verilog HDL on FPGA." 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). IEEE, 2016.

[13] Singh, Raj Prakash, Ankit K. Vashishtha, and R. Krishna. "32 Bit re-configurable RISC processor design and implementation for BETA ISA with inbuilt matrix multiplier." 2016 Sixth International Symposium on Embedded Computing and System Design (ISED). IEEE, 2016.

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-6, ISSUE-6, 2019
DOI: 10.21276/ijcesr.2019.6.6.12

[14] Dennis, Don Kurian, et al. "Single-cycle RISC-V micro-architecture processor and its FPGA prototype." 2017 7th International Symposium on Embedded Computing and System Design (ISED). IEEE, 2017.

[15] Rohit, J., and M. Raghavendra. "Implementation of 32-bit RISC processors without interlocked Pipelining on Artix-7 FPGA board." 2017 International Conference on Circuits, Controls, and Communications (CCUBE). IEEE, 2017.

[16] Venkategowada, N., et al. "Memory Architecture Quad Core Risc Processor on Altera FPGA De Nano Board." 2015 Fifth International Conference on Communication Systems and Network Technologies. IEEE, 2015.

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-6, ISSUE-6, 2019
DOI: 10.21276/ijcesr.2019.6.6.12