ijecs open access www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 3 March, 2014 Page No. 5132-5137

Design and Performances Analysis of 16-bit RISC Processor using Xilinx tool

Prof. D.B. Rane, Pokharna Pritesh G, Pawade Swapnil B, Chopade Yogesh S
Department of Electronics Engineering
Pravara Rural Engineering College, Loni.

***Abstract*: In this project we have described the design of a 16-bit non-pipelined RISC processor for applications in real-time embedded systems. The processor executes most of the instructions in single machine cycle making it ideal for use in high speed systems. The processor has been designed to be implemented on an FPGA using VHDL such that one can reconfigure it according to specific requirements of the target applications. the main objective of this project is to design and implement an 16-bit Reduced Instruction Set (RISC) processor using XILINX Spartan 3E tool. It involves writing a VHDL / verilog behavioral model, developing test-bench and simulating the behavior. The important components of this processor include the Arithmetic Logic Unit, Shifter, Rotator and Control unit. The instruction code is received at the beginning of each cycle, all operations are executed during the clock period, and results are stored at the end of it.**

***Keyword*: RISC, load store architecture, pipeline, processor, VHDL, Xilinx, FPGA**

I. INTRODUCTION

The reduced instruction set computer, or RISC, microprocessor CPU design philosophy that favors a smaller and simpler set of instructions that all take about the same amount of time to execute. The most common RISC microprocessors are ARM, DEC Alpha, PA-RISC, SPARC, MIPS, and IBM's PowerPC. The idea was inspired by the discovery that many of the features that were included in traditional CPU designs to facilitate coding were being ignored by the programs that were running on them. Also these more complex features took several processor cycles to be performed. Additionally, the performance gap between the processor and main memory was increasing. This led to a number of techniques to streamline processing within the CPU, while at the same time attempting to reduce the total number of memory accesses.. There we provide more than one execution unit. The time when one unit is busy with the current execution task, the fetch unit can probably fetch he next instruction which would be executed with the help of some other execution unit present in system.

The features which are generally found in RISC designs are uniform instruction encoding which allows faster decoding, a homogeneous register set allowing any register to be used in any context and simplifying compiler design, simple addressing modes, It was also becoming cost-effective to employ small amounts of higher-speed cache memory to reduce memory latency. the instruction set can be hardwired to speed instruction execution. No microcode is needed for single cycle execution [1].
The objective of this project is like that writing a VHDL/ verilog behavioral model. Developing test bench and simulating the behavior. Implementing of same on FPGA platform to validate the functionality. Synthesizing it using Xilinx ISE. In our project 16-bit RISC processor is used. the processor consist of control register ,different memory register, universal shift register, barrel shift register, arithmetic logic unit, accumulator and flag register. The instruction can be executed by using processor when different commands are gives.
Concept of RISC is like that, Overall design procedure is composed of pipeline, stage analysis, instruction execution analysis, RT-level (Register Transfer Level) functional unit composition, and control signal generation. A conventional computer executes one instruction at a time with a Program Counter pointing to the instruction currently being executed. Pipelining is analogous to an oil pipeline where the last product may have gone in before the first result comes out [2].

In this project we used Spartan-3 family of FPGA. The Spartan®-3E family of Field-Programmable Gate Arrays (FPGAs) is specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications. The five-member family offers densities ranging from 100,000 to 1.6 million system gates.Spartan-3E family builds on the success of the earlier Spartan-3 family by increasing the amount of logic per I/O, significantly reducing the cost per logic cell. New features improve system performance and reduce the cost of configuration. These Spartan-3E FPGA enhancements, combined with advanced 90 nm process technology, deliver more functionality and bandwidth per dollar than was previously possible, setting new standards in the programmable logic industry. Because of their exceptionally low cost, Spartan-3E FPGAs are ideally suited to a wide range of consumer electronics applications including broadband access, home

networking, display/projection, and digital television equipment.

The Spartan-3E family is a superior alternative to mask programmed ASICs. FPGAs avoid the high initial cost, the lengthy development cycles, and the inherent inflexibility of conventional ASICs. Also, FPGA programmability permits design upgrades in the field with no hardware replacement necessary, an impossibility with ASICs.

II. BLOCK DIAGRAM OF SYSTEM

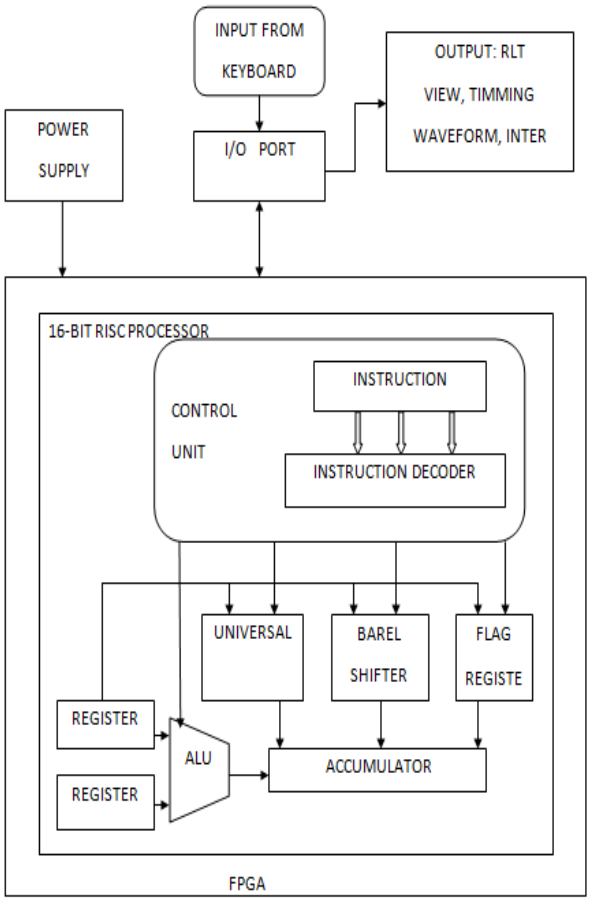


Fig.1 Block diagram of system

Design

In this project we look at some of the basic choices in the processor design space. We start our discussion with the number of addresses used in processor instructions. This is an important characteristic that influences instruction set design. We also look at the load/store architecture used by RISC processors.

Another important aspect that affects performance of the overall system is the flow control. Flow control deals with issues such as branching and procedure calls. We discuss the general principles used to efficiently implement branching and procedure invocation mechanisms. We wrap up the chapter with a discussion of some of the instruction set design issues.

POWER

Power is a critical part of our design, we have to make sure that we are not overloading or supplying too little power to any component. Either case would have been dangerous to the components. the controller rated at 5v and the processor need 2.5v. The power supply is used to convert the AC energy provided by the wall outlet to dc energy. In most electronic equipment, the power cord supplies the ac energy at 120 V to the power supply. The power supply then provides all the dc voltages needed to run the equipment.

FPGA

Today's electronic systems need to be brought to market more quickly, within budget, and with feature-sets that outperform competing products. Xilinx Spartan-3 FPGAs deliver the ideal solution, using 90nm process technology and staggered I/O pads to give you up to 5 million system gates and up to 784 I/Os with the lowest cost per gate and lowest cost per I/O of any FPGA. Spartan-3 Easy Path™ FPGAs further extend the benefits of the Spartan-3 FPGA family to volume production with a conversion-free, no-risk methodology that delivers up to 60% cost reduction. To address low-power challenges, Spartan-3L™ reduced-power devices lower quiescent power consumption by up to 98% and include an exclusive Hibernate mode. With all their cost and feature advantages, you can now use Spartan-3 FPGAs in higher volume than ever before. Plus, we make it easy to start your FPGA design immediately with the US $99 Spartan-3 Starter Kit.

Designers of gate-centric solutions face a common problem —

increasing design functionality while also minimizing device costs. This has often meant sacrificing either features or cost-effectiveness. The Spartan-3E FPGA family offers the low cost and platform features you're looking for, making it ideal for gate-centric programmable logic designs. Sparatan-3E is the seventh family in the groundbreaking low-cost Spartan Series and the third Xilinx family manufactured with advanced 90nm process technology.Spartan-3E FPGAs deliver up to 1.6 million system gates, up to 376 I/Os, and a versatile platform FPG Architecture with the lowest cost per-logic in the industry. This combination of state-of-the art low-cost manufacturing and cost-efficient architecture provides unprecedented price points and value. The features and capabilities of the Spartan-3E family are optimized for high-volume and low-cost applications and the Xilinx supply chain is ready to fulfill your production requirements.

III. DESIGN OF 16-BIT RISC PROCESSOR

Always in design methods involve one or more than one PCBs (printed circuit board) that contain many chips together with other peripherals. After that the required integrated circuits chips are selected followed by the PCBs that house and connect the chips together are designed according to design concept and put different components on PCB. Since the complexity of circuits implemented on individual chips and on the circuit boards is usually very high, it is very much essential to make use of Xilinx software [3].

Flowchart of 16-bit processor is given below :

1) Design entry- First define logic which we want to executed then write code for logic. It consist of different logics. Because of different logics write code in section.
2) Synthesis- Written code executed for many instruction. in this mode overall code synthesis of the system.
3) Physical design- Physical design of processor is done in this part .the main focus of this part is that the materials used for processor and which gives better physical strength and low cost.
4) Timing simulation- When code synthesis that time we used timing simulation. In this part we can check out the execution is right or wrong.
5) Chip – when above all steps are right then code download in chip.

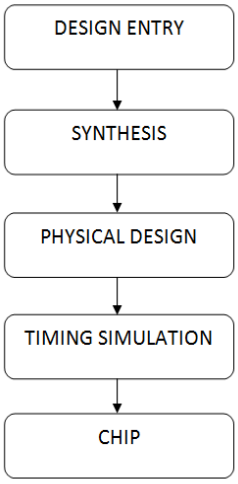


Fig.2 flowchart

Fig shows the design method of proposed 16 bit processor. FPGA devices are software configured and field programmable and easy to implementable. Hence, there is a significant cost saving in design and productions. Here FPGA kit use for only verification purpose. Design entry of the processor is carried out by using VHDL code. Functional simulation is done to verify the functionality of the circuit, based on inputs provided by the designer at the time of designing. Physical design determines how to implement the optimized circuit in a FPGA chip and consideration about area. Timing simulation determines the propagation delays that are expected in the implemented circuit. It also tells number of instruction executed in time.

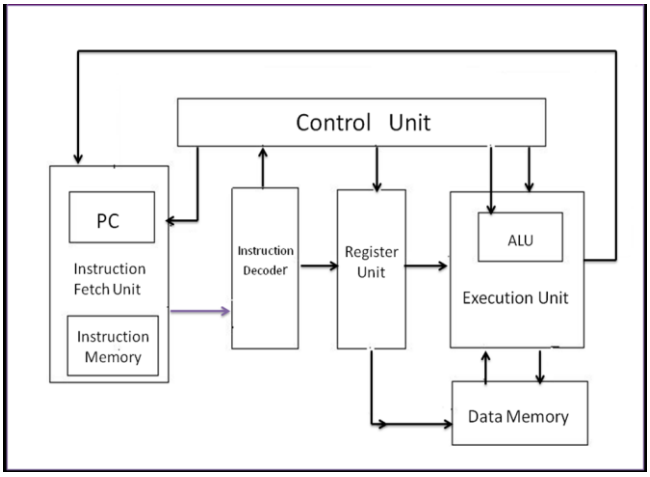


Fig.3 Block diagram of RISC Processor

The architecture of proposed 16-bit Processor is shown in Fig. which consists of processor block and memory block and data path. in control unit consist of controller, program counter, instruction register. ALU and resister file communicate with control unit through data path. The architecture of the 16-bit processor is designed based on three 16-bit instruction. The operation of the fetch stage starts when the program counter (PC) a 16-bit register is sent out to fetch the instruction from memory into the instruction register (IR) and the PC is incremented to address the next sequential instruction. The IR is used to hold the instruction needed on subsequent clock cycles.

The processor's 16-bit registers are stored in 'register file' that contains the register state of the machine. For R-format instructions, there are three register operands. Two data words are read from the register file and one data word is written into the register file for each instruction. The register number inputs are 4 bits wide to specify one of the 16 registers, whereas the data input and two data output buses are each 16 bits wide. When the instruction is fetched from the fetch stage, the instruction's operation code is sent to the control unit. The instruction's register address fields are used to address the two port register file. The two-port register performs two independent reads and one write in one clock cycle .In the decode stage, the instructions are decoded and the register file is accessed to read the registers. The outputs of the General purpose registers are read into two registers, register 1 and register 2. After the Instructions decode stage, the execution stage performs calculations [3].

Top Level Block Description

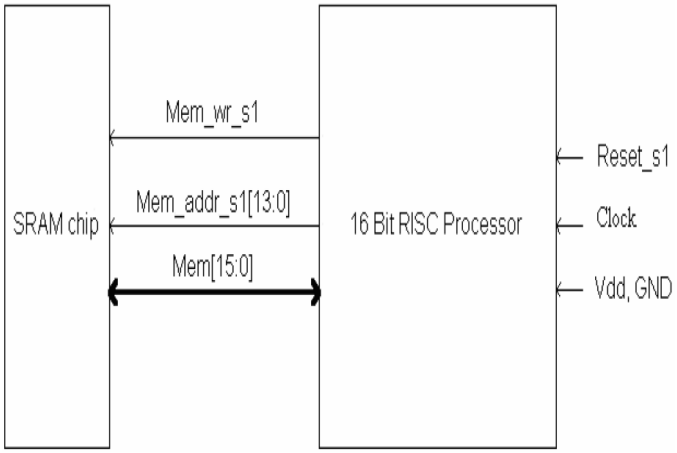


Fig.4 Block Description

The micro-architecture refers to a view of the machine that exposes the registers, buses and all other important functional units such as ALUs and counters. The principle subsystems of a processor are the CPU, main memory and the input/output. The data path and the control unit interact to do the actual processing task. The control unit receives signals from the data path and sends control signals to the data oath. These signal s control the data flow within the CPU and between the CPU and the main memory and input/output.

program counter

The program counter output is used as address of the instruction memory. The program counter is a 16 bit counter. The program counter value depends on input condition. The program counter value is either incremented or loaded with branch address or loaded from stack registers

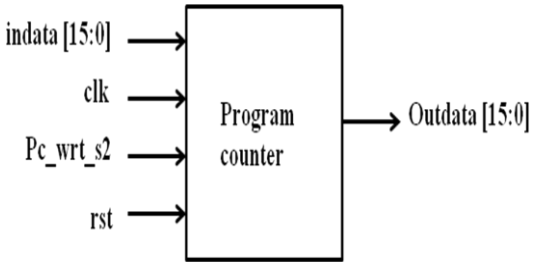


Fig.5 Program counter

instruction register and register file

The instruction register stores the instruction to be executed. The instruction register is loaded from an external instruction ROM. The program counter value is used as address to the instruction ROM. The format of the 16 bits instruction register is shown here. The first 4 bits are op-code, which are input to the control unit. The control unit decides the operation sequence of the data path. The next 4 bits are used as address of the source register 1. The next 4 bits are used as address of the source register 2. The next 4 bits are used as address of the target register. The function of the two source registers, target register are explained in instruction set description.

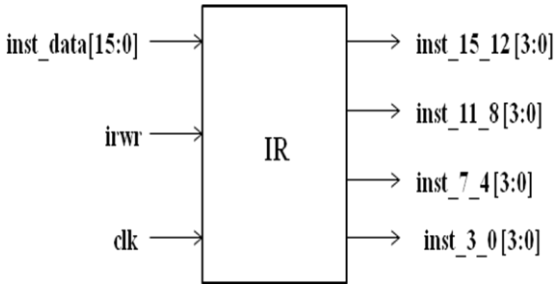


Fig.6 Instruction register

Control Unit Design

The Control FSM has only three distinct states that determine the operation of the processor: IDLE, FETCH and EXECUTE. Here fetch and Execute is further divided into two states, Fetch instruction state and Fetch operands state. Similarly Execute state also divided into two parts. When the reset signal (reset_s1) goes high from any state, the FSM will be placed in the IDLE state. While in the IDLE state the control unit will send the PC write enable signal (pc_wrt_s2 =1) and select zero (pc_sel_s2=0) as the current program counter.

When the reset signal goes low, the FSM's next state will be the FETCH state and the instruction from Memory address 0 will be loaded into the Instruction Register (IR) to begin program execution. The control looks at the next state = FETCH and generates the IR write (ir_wrt_s1), Operand a Select (opA_sel_s1), Operand B Select (opB_sel_s1 = 0010) and the ALU add operation (alu_op_s1 = 00000001) to load the IR with the next instruction and increment the PC by 1. These events all occur on the first clock of the FETCH state. One-hot signals are used for alu_op_s1, opB_sel_s1, and data_sel_s2 to make for easier decoding in the datapath units. The operation at the next phase of FETCH will be determined by the opcode (opcode_s2) from the IR, except for the incremented PC that is written in from the ALU output latch in all cases. The ALU Operations will load in Operands A and B from the Register File. The Load word will only need Operand A, while the Store word will need both operands (one for the address and one for the data word). The Branch instructions will use the offset in its instruction word and PC + 1 count as operands into the ALU. The JAL stores the incremented PC in the Register File, while the JR loads the return address into Operand A.

After phase two of the FETCH state, the FSM enters the EXECUTE state. During the first phase for an ALU operation, the appropriate alu_op_s1 control signals are sent to the ALU as decoded from the opcode. The operand mux (opA_sel_s1 & opB_sel_s1) control signals are also generated to select the latch outputs. For the other operations (except LI), an add operation is required from the ALU. The operands chosen for the add operation are determined by the operation specified. The Load and Store words will access Memory on this first phase as well. The second phase of EXECUTE writes data into the register file or writes a new address into the PC. For the branch instruction, the control will look at the check zero signals from operand A to determine if the branch should be taken and the new PC should be written. The control returns the next state to FETCH to repeat the process for the next instruction. Instruction operation opcode is given below.

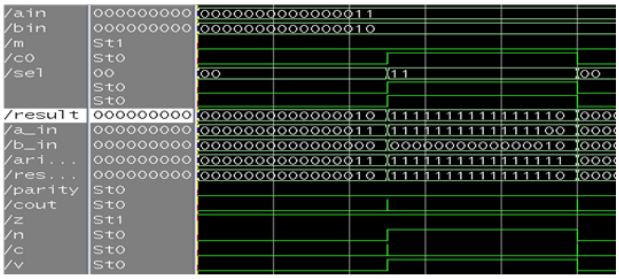
Operation	Opcode				Destination Reg (Rd)				Source Reg (Rs)				Target Reg (Rt)			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD	0	0	0	0	Rd				Rs				Rt			
SUB	0	0	0	1	Rd				Rs				Rt			
AND	0	0	1	0	Rd				Rs				Rt			
OR	0	0	1	1	Rd				Rs				Rt			
XOR	0	1	0	0	Rd				Rs				Rt			
NOT	0	1	0	1	Rd				Rs							
SLA	0	1	1	0	Rd				Rs							
SRA	0	1	1	1	Rd				Rs							
LI	1	0	0	0	Rd				Immediate							
LW	1	0	0	1	Rd				Rs							
SW	1	0	1	0					Rs				Rt			
BIZ	1	0	1	1					Rs				offset			
BNZ	1	1	0	0					Rs				offset			
JAL	1	1	0	1	Rd								offset			
JMP	1	1	1	0									offset			
JR	1	1	1	1					Rs							

Fig.7 instruction operation

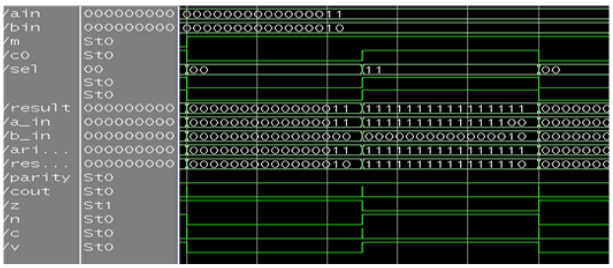
IV. SIMULATION AND DISCUUSSION

We have simulated the VHDL code of the proposed processor using Xilinx Software tool (Version 10.1). The simulation results of processor are presented for justification. The proposed 16 bit RISC processor is coded with VHDL (very high speed integrated circuit hardware description language). When there is no error, the code is synthesized and simulated using Xilinx software. Before the start of simulation, the code written in memory that is in register. The completed processor with memory is tested for arithmetic, logical and shifting program. When the VHDL code is 100% synthesized and check, then the code is downloaded to the FPGA device. After downloading the code, the functionality of 16 bit RISC processor is done according as a input instruction.

Logical operation



Arithmetic operation



The simulation result of the proposed 16 bit RISC processor is shown in Fig. the simulation window is shown for different operation. Control path is used for synthesis and simulation.

Analysis of16-bit RISC processor-

1) 16-bit RISC processor is synthesis on 50MHz frequency and time required for that is 7ns. For non-pipeline processor one pass is selected for best. There is different default because of wire used.

2) We can see area report after synthesizing. For this processor one ground is used and one VCC is used. Total number of IOs is 47 and total number of LCs is 28 in processor. 25.41 % area used for IOs that is 47 out of 185.

3) Because of reset load is 0.52 on processor. Data required time for processor is 7ns and data arrivals time for processor is 6.26ns. Therefore the total slag is 0.74.

V. CONCLUSION

We can conclude that from overall system. We checked out our processor architecture by different instruction. We perform arithmetic, logical and shifting programs which created using Xilinx Software. We have compared the simulated output results with the expected results which we considered at the timing of design.. From synthesis report, the minimum clock period that can be achieved in this proposed architecture is 50 ns (50 MHz). Similarly, the functionality of execution of instruction, area used time delay and flow of data path is tested and found correct. Finally, we eliminated errors which produce infinite result.

REFERENCES

[1] R. Uma, Apr 2012, “Design and Performances Analysis of 8-bit RISC Processor using Xilinx tool”, international journal of engineering research and application pp.053-058.

[2] Swati Joshi and Puran Gour, Sept.2012,”Area Optimized 32-bit pipeline RISC processor in VHDL”, International journal of Digital Application and Contemporary research.

[3] Manoranjan Pradhan, April-2011,”Simulation and verification of self test 16-bit processor”, International Journal of Computer Application.

[4] Galani Tina G., Riya Saini and R.D. Daruwala, July-2013,”Design And Implementation of 32-bit RISC Processor Using Xilinx”, International Journal of Emerging Trends in Electrical and Electronics.

[5] V. N. Sireesha and D. Hari Hara Santosh, june-2012,”FPGA Implementation of A MIPS RISC Processor”, International Journal of Computer Technology and Application.

[6] Amit kumar Sing Tomar and Rita jain, Sept-2012,”Imlemenation of RISC System in FPGA”, International Journal of Emerging Technology and Advance Engineering.

[7] V.R.Gaikwad,April2013,”Desingn,Implementation and Testing of 16-bit RISC Processor”, IOSR journal of VLSI and Signal processing , pp-01-04.

AUTHOR BIOGRAPHY

Prof. D.B.Rane, Department of Electronics Engineering, P.R.E.C, LONI, MAHARASHTRA, INDIA

Mr. Pokharna Pritesh G. B.E. student, Department of Electronics Engineering, P.R.E.C, LONI, MAHARASHTRA, INDIA.

Mr. Pawade swapnil B. B.E. student, Department of Electronics Engineering, P.R.E.C, LONI, MAHARASHTRA, INDIA.

Mr. Chopade Yogesh S B.E. student, Department of Electronics Engineering, P.R.E.C, LONI, MAHARASHTRA, INDIA.