DESIGN AND IMPLEMENTATION OF 16 BIT LOW POWER FPGA BASED RISC EMBEDDED PROCESSOR USING VHDL

TARUN KUMAR SHARMA
M.Tech Student, Department of EE,
NITTTR, Kolkata
tarunkusharma@yahoo.co.in

Dr. SOUMITRA KUMAR MANDAL
Associate Professor, Department of EE,
NITTTR, Kolkata
mandal_soumitra@yahoo.com

Abstract

In this paper, low-power consumption is the main concern which has been taken into account and to maintain this issue, we used 16-Bit RISC processor coded into VHDL with Clock Gating technique. Here RISC processor incorporated with 5 pipeline stages which are instruction fetch, instruction decode, instruction execute, data memory and write back. For fast computing each instruction use one clock cycle (CPI), so that upon initiation of each clock pulse new instruction will be fetched and then go on with remaining pipeline stages unless and until hazard is detected.

This processor used several features for high-speed architectural design purpose like 16 bit data bus, 16 bit address bus, 16 general purpose register (GPR) of 2 byte each, barrel shifter to speed up the multiplication and division operation of the processor etc. Here we used only three instruction types for predefined instruction set architecture, which are register type, immediate type and branch type to reduce stalling and more heat dissipation of logic gates rather than for complex instruction in CISC.

Implementation of the processor has been done on Altera Stratix FPGAs board which operate faster than other logic design board for processor. Simulation of the each module for verification has been done on Altera Quartus II with individual test benches. Frequency range for processor is 69.27MHz to 77.94MHz.

Keywords *: Low-power, RISC, VHDL, Clock Gating, Pipeline, CPI, Hazard, High-speed Architecture, GPR, FPGAs, Altera.*

1. INTRODUCTION

As the high capacity, low-cost FPGA devices train continues its revolutionary journey through the electronics design Landscape, an ever-increasing number of design landscape, an ever-increasing number of designers are jumping on board trading their traditional hardware-based systems for the attractiveness of the FPGAs “soft programmability”. Soft processors are processors that are defined as part of the FPGA design that is programmed into the FPGA device, by using VHDL coding environment[1].

The efficiency of RISC(Reduced Instruction Set Computer) architecture has been proved in a wide range of applications such as embedded controllers and power efficient portable information devices like PDA(Personal Digital Assistant) as well as high performance main frame server and workstations. Especially, down-suing of electronic devices and cost effective ASIC design methodology lead the bulky FCs and workstations to be the portable ones. The designers of microprocessor, therefore, has focused their attention on not only integrating a large number of different functional blocks in a small chip size but also designing an application specific architecture at the cost of very high performance. The primary goal of this RISC microprocessor is a portable workstation that requires low cost, low power, small size, and high performance[2].

CISC processors have gained the common marketplace over the years but their design is complicated and instruction are variable in length. Study over the years proved that simple instruction are used 80% of the time and many complex instruction can be replaced by group of simple instruction[3]. To satisfy the above low cost, low power, small size, and high performance characteristics we used 16-Bit RISC processor due to its less complexity and high performance result than 32 and 64 bit RISC. RISC processor operates on very few data types and does the simple operations. It supports very few addressing modes and are mostly register based. Most of the instructions operate on data present in internal registers. Only LOAD and STORE instructions access data in external memory. Also the instruction length is fixed and hence decoding is easier. Parallel execution of instructions through the pipelined stages of processor improves the overall throughput of the processor but introduces some hazards in its working [4]. Data hazards are due to sharing of destination and source resources in succeeding instructions and they can be resolved by the method of forwarding. Structural hazards are due to common program and data memory. We used Harvard architecture to remove Structural hazards which has separate program and data memory.

Proposed architecture consist of 5 pipeline stages with 16 bit data and address bus, input and output ports connected to program and data memory, and few control signals shown in the figure 1.

ISBN: 978-81-923777-9-7

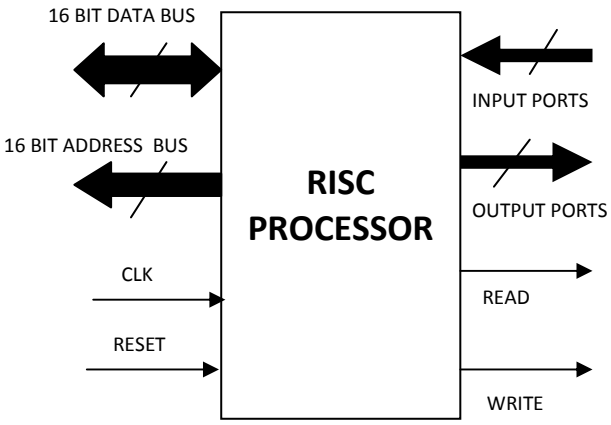


Fig.1. RISC Processor

2. LOW-POWER CONSUMPTION

The increasing prominence of portable electronics and consumer-oriented devices has become a fundamental driving factor in the design of embedded processor. As the focus shifts away from tethered desktop computing to the mobile appliance, a rethinking of design optimizations traditionally targeting ever-increasing performance goals and high clock rates at almost any cost are required in order to optimize battery life and extend the utility of these devices. Now the technology shift towards flexible mobiles which will require low-power design of the processor for miniaturization of the every component of the device[5].

For lowering power consumption we used generally three techniques which are

Supply Voltage Reduction Method

Reducing power dissipation has become an important objective in the design of digital circuits. One common technique for reducing power is to reduce the supply voltage. Power consumption (P) of a CMOS based processor is related to the supply voltage (V), switching frequency (f), and CMOS gate capacitance (C).

$$P=C*f*V*V$$

Full Custom Design

Full custom design will help to reduce the number logic gates in the implementation of the necessary functionality of the processor. Since each logic gate requires power to operate, a lower gate count means fewer switching and thus power need.

Clock Gating

Clock gating is a method where the clock signal is prevented from reaching the various modules of the processor. The absence of the clock signal prevents any register and/or flip-flop from changing their value. As a result of this, the input to any combinational logic circuit remains unchanged, and thus no switching activity takes place in those circuits. Since, in CMOS circuits, most of the power dissipation results from switching activity, clock gating greatly reduces the overall power consumption. In this design, we mostly concentrated on the Multiplier, ALU, ALU Control Unit, and the Branch Adder for clock gating. The reason for this is that these are the biggest modules in our architecture in terms of number of logic gates,. Therefore, a significant improvement in power use can be achieved by gating these components.

3. DETAILED ARCHITECTURE

The proposed architecture consist of 5 pipeline stages, separate instruction as well as data memory as per Harvard architecture for faster execution of the processor, controller to generate control signal for each block, 16 GPR of 2 bytes with 6 status registers to store the processor state, ALU for arithmetic calculation with two external features 16*8 multiplier and barrel shifter which reduces the work load of ALU to speed up the processing of multiplication and division instructions thus gives better performance result than the previous 16-Bit RISC. Two extra circuitry hazard detection and forwarding unit used here which got activated by controller when -ever hazard is detected and stall the next instruction.

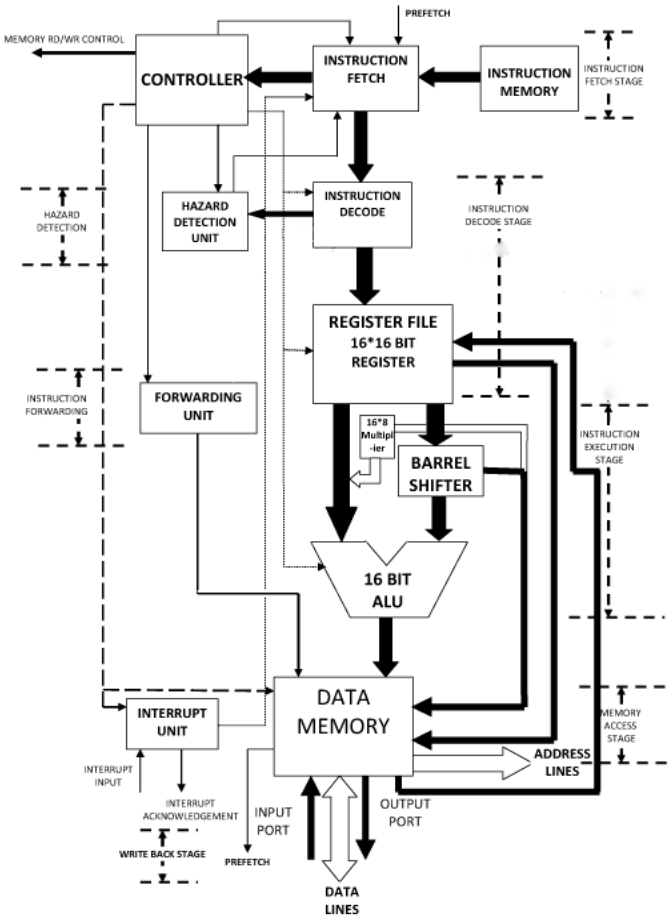


Fig.2. Detailed Architecture

ISBN: 978-81-923777-9-7

Pipeline Stages **:** The proposed architecture consist of five pipeline stages viz. *instruction fetch stage, instruction decode stage, instruction execution stage, memory read write stage* and *memory/IO write back stage*. Each stage is connected to the next stage through output register of each stage to implement just-in-time principle. *Instruction fetch stage* fetches instruction from instruction memory and prefetch signal consist of program counter and selector to choose correct PC value based on current instructions in execution stage. *Instruction decode stage* separates the opcode, function and the source and destination codes for control unit. Register file is a part of this unit, which consist of 16 general purpose register. If the instruction has immediate data, it is signed extended to 16-bit by this stage. This stage also responsible for differentiating between absolute branching and relative branching. *Instruction execution stage* does all the arithmetic and logical operation including shift and rotate. Arithmetical and logical operations are carried out by basic ALU, shift and rotate operation are carried out by barrel shifter unit. Multiplier unit helps to reduce the complexity and time consuming of multiplication and division instructions in ALU. *Memory access stage* act as a output register for all the remaining pipeline stages except instruction decode stage which already has register file. *Memory/IO write back stage* is responsible for accessing external memory for data and for writing the results back into destination register. In case of IN instruction, data on input port is directly sent to destination register by this unit. It also place data on output port for OUT instruction.

Control Unit **:** It gives actuating signal to each block for their actions. Control unit performs vital role during decoding stage upon which, it generates necessary signals for specific operations to be performed by ALU, register file, barrel shifter and multiplier. It also generates necessary signal for hazard detection and forwarding unit.

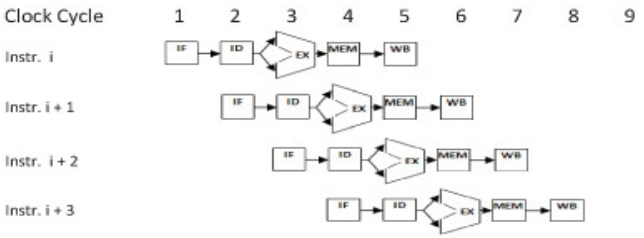


Fig.3. Pipeline Processing without Hazard

Hazard Detection Unit **:** This unit primarily detects the resource conflicts upon getting signal from instruction decode stage and generates the necessary signal for execution forwarding unit through controller and for instruction fetch stage. The source of current instruction may be the destination of previous instruction and in such case the data supplied to execution stage is not correct until result of previous instruction is written back to destination and there is stall in the processor.

Forwarding Unit **:** When data hazards occurs this unit directly forwards the result of execution unit or results from write-back stage as a source to the instruction following the current instruction in execution stage.

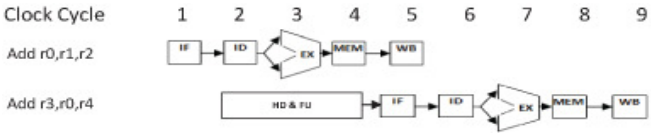


Fig.4. Pipeline Processing with Hazard

In fig.4, result of r0 is required in the next instruction, so there is a stall of 3 clock pulse by hazard detection and forwarding unit.

Multiplier **:** The multiplier employs a modified Booth's algorithm and an early termination algorithm that multiplies multiplicands in 8-bit stages. Each 8-bit multiply stage requires one instruction cycle to execute. Thus, a multiply instruction can take 2 - 5 cycles. The multiplier employs carry-save adder to avoid the carry-propagate delays. Intermediate results are held as partial sum and partial carries where the true binary result is obtained by adding these two together in main ALU. You could see in fig.2, that one bus is directly connected and other through barrel shifter to the ALU.

Barrel Shifter **:** The barrel shifter allows one of the ALU input operands to be shifted or rotated by any number of bits prior to ALU operations. It helps in the multiplication and division instruction to be executed quick, and saves the stalling time due to hazard and forwarding unit.

Interrupt Unit **:** RESET has the highest priority and then clocking of modules using clock gating technique in the processor. These are the hardware interrupt which were used here. It also interrupted instruction fetch stage during hazard which is a software interrupt.

Prefetch Unit **:** This unit makes use of idle bus time to prefetch instructions from memory and stores them in prefetch queue. Prefetch queue has the capacity of storing four instructions at a time along with tag for each of them. The tag specifies from which memory location the corresponding instruction has been fetched. The prefetch unit works like small look-ahead cache. It does not work on FIFO principle, This unit supplies instruction to instruction fetch stage.

4. INSTRUCTION SET OVERVIEW

The architecture of the advanced 16-bit RISC processor is designed based on three 16-bit instruction formats R-format, I-format and J-format illustrated in Tab.1. The design of this project consists of 16-bit instructions and 16-bit datapath.

ISBN: 978-81-923777-9-7

Tab.1. 16-bit Instruction Format

Field Strength	4-bits	4-bits	4-bits	4-bits
R-Format	Opcode	Rs1/Rd	Rs2	Function
I-Format	Opcode	Rs1/Rd	Immediate value	
J-Format	Opcode	Branch Target		

Tab.2. Instruction Set of 16-bit RISC

Instruction	Description	Instruction Type
Add	Addition	Register
Addu	Unsigned addition	Register
Addi	Addition(Immediate)	Immediate
Sub	Subtraction	Register
Subu	Unsigned Subtraction	Register
Subi	Subtraction (immediate)	Immediate
Mul	Multiplication	Register
Muli	Multiplication(immediate)	Immediate
Cmp	Compare	Register
And	AND	Register
Andi	AND(immediate)	Immediate
Or	OR	Register
Ori	OR(immediate)	Immediate
Not	NOT	Register
Xor	XOR	Register
Sll	Logical shift left	Register
Srl	Logical shift right	Register
Sla	Arithmetic shift left	Register
Sra	Arithmetic shift right	Register
Lw	Load word	Register
Sw	Store word	Register
Mov	Move data between register	Register
Movi	Move data(immediate)	Immediate
Beq	Branch if equal to 0	Branch
Bne	Branch if not equal to 0	Branch
Ba	Branch always	Branch
Movy	Move data from Y register	Register
Nop	No operation	N/A

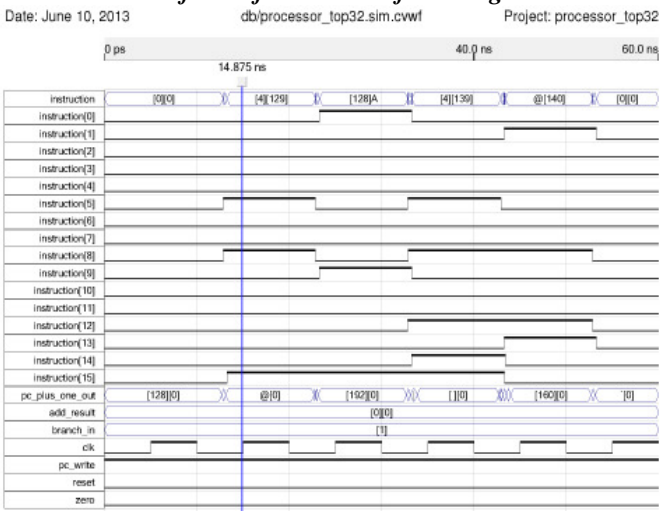
Advanced 16-bit RISC design technique typically uses a large number (commonly 16) of registers. Most instructions operate on these registers, with access to memory made using a very limited set of Load and Store instructions. This limits the need for continuous access to slow memory for loading and storing data. The implementation of RISC performs fetch, decode, and execute in clock cycle per instruction (CPI) technique which makes it faster than the previous 16-bit RISC. The single clock cycle RISC architecture is separated into five pipeline stages as already discussed.

In Tab.1. we could easily verified the simplicity of instruction format due to equal field strength of each column in register format. This reduces the VHDL coding complexity than 32-bit RISC processor.
In Tab.2. there are 28 instructions which are tested completely by simulation in Altera Quartus II.

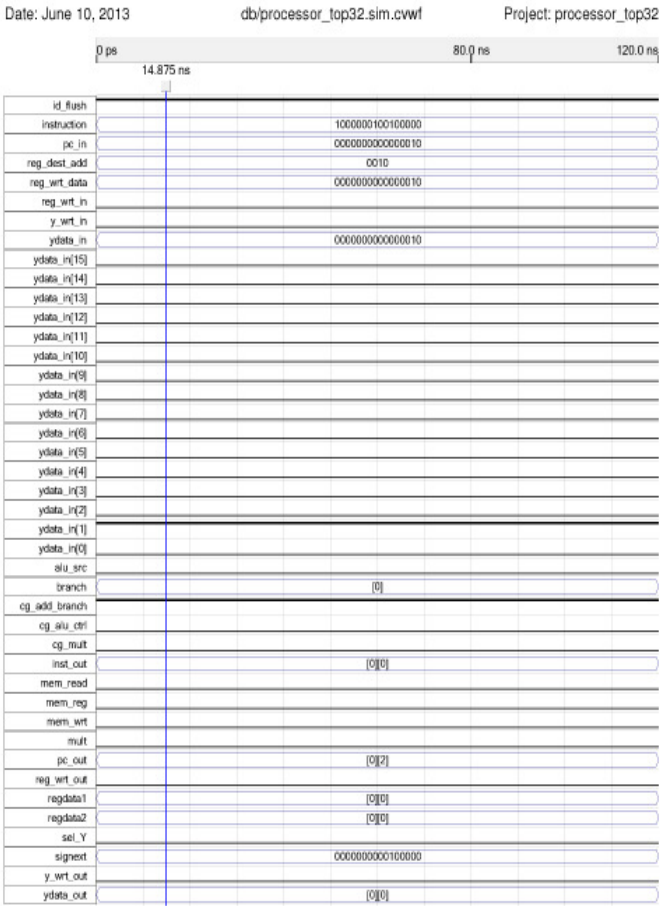
5. SIMULATION AND DISCUSSION

For successful implementation of this project the compilation and simulation of the model is run using ALTERA Quartus II Web Edition version 8.1. Simulation of design is done in a bottom-up fashion to verify the correctness of design. Small modules are simulated in separate test benches before they are integrated. There is one test bench for each small module and a test bench for the complete device. Simulated waveforms for each stages are shown. The simulated results of the top-level module are tabulated in Tab.3. that shows the correctness of the model.

Simulation waveform of instruction fetch stage:

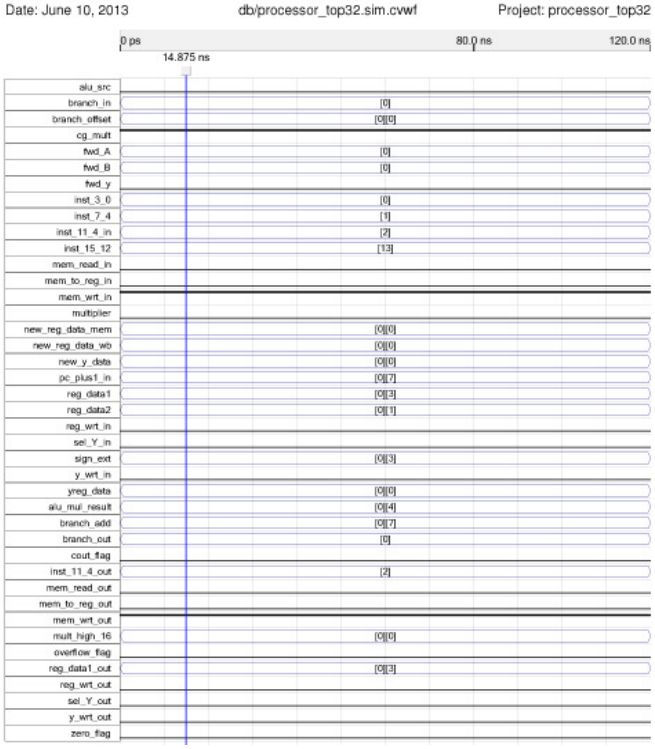


***Simulation waveform of instruction decode stage*:**

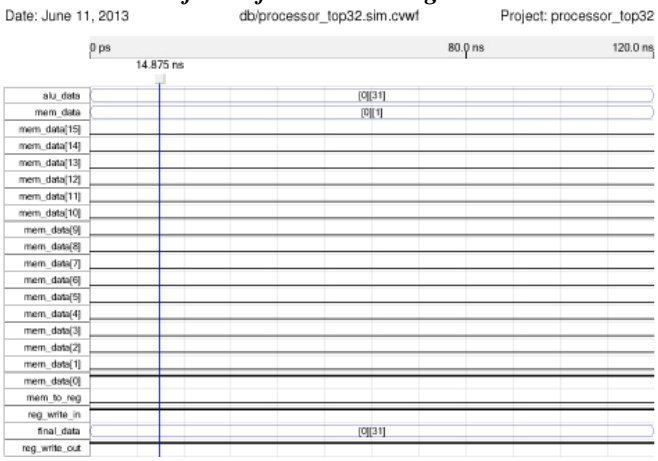


ISBN: 978-81-923777-9-7

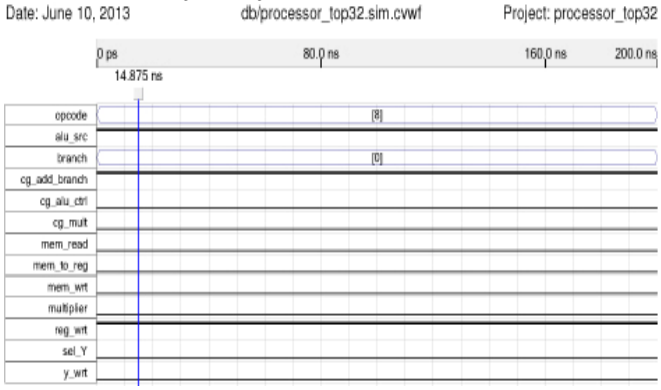
Simulation waveform of instruction execte stage:



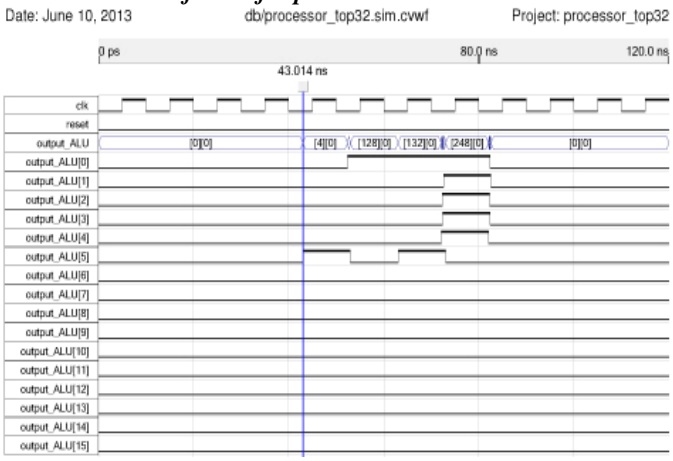
Simulation waveform of write back stage :



Simulation waveform of control unit :



Simulation waveform of top-level module :



Tab.3. Result of the top-level module simulation

Instructions	Clock	Reset	Register1	Register2	ALU
NOP	0	0	0	0	0
NOP	0	1	0	0	0
MVI R1,32	1	0	32	0	32
MVI R2,1	1	0	32	01	01
ADD R1,R2	1	0	33	01	33
SUBI R1,2	1	0	31	01	31
NOP	0	0	31	01	0

6. CONCLUSION

By simulating with various test vectors a single clock cycle advanced 16-bit RISC processor using VHDL is successfully designed, implemented and tested. We finally downloaded the VHDL code into ALTERA Stratix EP1S10F484C5 FPGA device. The RISC concept proved that 20% of instruction did 80% of the work[3]. We can use this processor in digital camera for its low cost technology.

In future we can work on advanced version of 16-bit MIPS processor for its low cost and great deal with low power consumption. As today's evolution towards flexible portable instrument, so this will help to fulfill the criteria of the device[11].

For example take a look on register type instruction, which partially describe our concept on 16-bit MIPS for future advancement.

Tab.4. R-Format Instruction of 16-bit MIPS

OPCODE	Rs1	Rs2	Rd	Reserved
5-bit	3-bit	3-bit	3-bit	2-bit

By using this technique, we can process maximum of 64 instructions as opcode utilizes 5 bit and 2 bit for reseve state for instructions which have same opcodes. If we can extend 3-bit of register to 8-bit of register indexing, then there will be 128 GPR, which will prove the statement led by John Hennessy, Stanford "At the time that the CISC machines

ISBN: 978-81-923777-9-7

were able to do 32-bit microprocessors, the RISC machines were able to build pipelined 32-bit microprocessors. At the time you could do a basic pipelining in CISC machine, in a RISC machine you could do superscalar designs, like the RS/6000, or superpipelined designs like the R4000. I think that will continue. At the time you can do multiple instruction issue with reasonable efficiency on an x86, I believe you will be able to put second-level caches, or perhaps even two processors on the same piece of silicon, with a RISC machine"[12].

7. REFERENCES

[1] M.Kishore Kumar, MD.Shabeena Begum, " FPGA BASED IMPLEMENTATION OF 32 BIT RISC PROCESSOR", IJERA, ISSN: 2248-9622, Vol. 1, Issue 3, pp.1148-1151.

[2] Seung Ho Lee, Beoyng Yoon Choi, Moon Key Lee, "ASIC IMPLEMENTATION OF A RISC MICROPROCESSOR FOR PORTABLE WORKSTATION",IEEE Publication,1995

[3] Pravin S. Mane, Indra Gupta, M. K. Vasantha, "Implementation of RISC processor on FPGA", International conference on Computing & Processing, IEEE 2006 , Page(s): 2096 – 2100.

[4] John L. Hennessy and David A. Patterson, "Computer Architecture A Quantitative Approach", 3rd Edition, Morgan Kaufmann Publishers, 2003.

[5] Sagar Bhavsar , Akhil Rao , Abhishek Sen , Rohan Joshi, "A 16-bit MIPS Based Instruction Set Architecture for RISC Processor", International Journal of Scientific and Research Publications, Volume 3, Issue 4, April 2013

[6] BILL MOYER, "Low-Power Design for Embedded Processors", IEEE Journal,2001

[7] Geun-young Jeong, Ju-sung Park, "DESIGN OF 32-BIT RISC PROCESSOR AND EFFICIENT VERIFICATION", 7th Korea-Russia Infernational Symposium. KORUS 2003

[8] Yasuhiro Takahashi, Daijiro Tsuzuki, Toshikazu Sekine, and Michio Yokoyama, "Design of a 16-bit RISC CPU Core in a Two Phase Drive Adiabatic Dynamic CMOS Logic",IEEE,2007

[9] Samiappa Sakthikumaran, S. Salivahanan, V. S. Kanchana Bhaaskaran, "16-Bit RISC Processor Design for Convolution Application", IEEE-International Conference,2011

[10] Mamun Bin Ibne Reaz, Md. Shabiul Islam, Mohd. S. Sulaiman, "A Single Clock Cycle MIPS RISC Processor Design using VHDL",IEEE Conference,2002

[11] Sagar Bhavsar , Akhil Rao , Abhishek Sen , Rohan Joshi, "A 16-bit MIPS Based Instruction Set Architecture for RISC Processor", IJSRP, Volume 3, Issue 4, April 2013.

[12] Ciji Isen, Lizy John, and Eugene John, "A Tale of Two Processors: Revisiting the RISC-CISC Debate", Springer-Verlag Berlin Heidelberg, 2009

ISBN: 978-81-923777-9-7