SINGLE CYCLE 64-BIT RISC-V PROCESSOR AND IT'S FPGA PROTOTYPE

PALLAVI S
Student, M.Tech, Dept of E&C
Dr. Ambedkar Institute of Technology
Bengaluru-560056

SWAMY T N
Assistant Professor, Dept of E&C
Dr. Ambedkar Institute of Technology
Bengaluru-560056

Abstract— In the early years this computer was stack architecture, later replaced by RISC architecture. Now the intent is to replace the hypothetical, emulated computer by a real one. This idea was made realistic by the advent of programmable hardware components called field programmable gate arrays (FPGA). The RISC-V Processor have 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. In this light, the choice of a RISC (Reduced Instruction Set Computer) is obvious. The use of an FPGA provides a substantial amount of freedom for design. Yet, the hardware designer must be much more aware of availability of resources and of limitations than the software developer. Also, timing is a concern that usually does not occur in software, but pops up unavoidably in circuit design.

In this paper, development of a fully synthesizable 64-bit processor based on the open-source RISC-V (RV64I) ISA is presented. This processor is designed for targeting low cost embedded devices. The resulting processor is a single core, in-order, non-bus based, and RISC-V processor with low hardware complexity. The proposed processor is implemented in Verilog HDL and further prototyped on FPGA" Spartan 6" board. It is found that the maximum operating frequency is 140.398MHz. The power is estimated to be 0.037 W using Xilinx Power Analyzer.

Index Terms — RISC-V,5-Stage Pipelining, FPGA, Verilog HDL.

I. INTRODUCTION

In the area of Computer Architecture, it was originally designed to support research and education, for academic and industrial applications RISC-V instruction set architecture (ISA) is now set to become a standard free and open architecture. For the success and adoption of RISC-V, 32-bit, 64-bit and 128-bit address spaces support by RISC-V. A minimal set of instructions adequate to provide a reasonable target for assemblers, linkers, compilers and operating systems, the ISA is separated into a small base integer ISA. The set of compatible tool chains which includes the above Suits, provided by RISC-V foundation.

Using both low level applications and mobile systems by the beginning of the 21st century, RISC based architectures have been used. By the RISC based ARM architectures, the low power and low cost embedded market are dominated. The ARM architecture is used by android based devices, Apple iPhone an iPad and most hand-held devices. Games like PlayStation Portable game consoles, Nintendo 64 and personal residential gateways like Linksys WRT54G series contains MIPS line. By Hitachi, SuperH (SH) is another 32-bit RISC ISA was developed. Since, many of the patents are expiring for SuperH, open source hardware under the name J2, SuperH2 is being reimplemented.

II. HISTORY

In [1] paper the increasing popularity of systems on a chip, where processors are just a small fraction of the design, calls into question why one of the most important interfaces is proprietary. We argue that: There is no good technical reason not to have free, open instruction sets just as we have free, open networking standards and free, open operating systems. The best architectural style for a free, open instruction set is RISC. Given the time it takes to design an instruction set, it makes more sense to adopt an existing RISC free, open instruction set than to design a new one from scratch. Among the existing RISC free, open instruction sets, RISC-V is the best and safest choice.

In [2] development of RISC-V to support our own needs in research and education, where our group is particularly interested in actual hardware implementations of research ideas , and in providing real implementations for students to explore in classes. In our current research, we are especially interested in the move towards specialized and heterogeneous accelerators, driven by the power constraints imposed by the end of conventional transistor scaling. We wanted a highly flexible and extensible base ISA around which to build our research effort.

In [3] Floating point operations are important and essential part of many scientific and engineering applications. This paper details an architectural exploration for floating point coprocessor enabled with RISC-V floating point instructions. The Floating unit that has been designed with RISC-V floating point instructions is fully compatible with IEEE 754-2008 standard as well. The floating point coprocessor is capable of handling both single and double precision floating point data operands in out-of-order for execution and in-order commit. The front end of the floating point processor accepts three data operands, rounding mode and associated. The coprocessor for floating point integrates with integer pipeline. The proposed architecture for floating point coprocessor with out-of-order execution, in-order commit and completion/retire has been synthesized, tested and verified on Xilinx Virtex 6 xc6vlx550t-2ff1759 FPGA. A system frequency of 240MHz for single precision floating point and 180 MHz double precision floating point operations has been observed on FPGA.

In [4] a 5-satge pipelined 32-bit instruction set processor compatible with RISC-V ISA has been micro-architected and analyzed the design issues / options for pipeline stages in terms Of grouping of instruction, decoder logic complexity, register file access and control flow instructions. Processor includes a sophisticated branch prediction and advanced data hazard Detection with data forwarding unit and IEEE 754-2008 compliant out-of-order execute and in-order-commit FPU. Both integer and floating point pipeline are verified at test bench level, core is implemented on Xilinx Virtex 6xc6vlx550t- 2ff1759 FPGA and ASIC (UMC 65 and 130 nm library).

III. OBJECTIVE

Through this approach a lot many new amount of Processor's are also into the market. Out of all these processor's a few of them were designed using processor cores i.e. Hardware Description Languages like Verilog-HDL and VHDL (Very High Speed Integrated Circuit Hardware Description Language) , is used for writing a particular version of processor. This helps the designer to use them in any of the embedded applications. These can be used in the processor just by embedding a particular application in the processor. RISC (Reduced Instruction Set Computer) is an efficient Computer Architecture which can be used for the Low power and high speed applications of the processor RISC Processors are important in application of pipelining. The curb of the processor is the Instruction Set Architecture used for developing it. The total worthiness of the processor depends on utilizing the Instruction Set Architecture. Instruction Set Architecture is a metaphysical interface between Low level system of the machine and the hardware, that contain all the information about the machine , required to However a lot of research is being carried out in the field of processor's to satisfy the performance issues.

1) To develop and implement RISC-V processor algorithm, with high efficiency and less power consumption.
2) To maximize the performance and operations of a processors for many application.
3) Software scheduling and optimizing compilers of RISC-V architectures.
4) Develop with a suitable technique and dump it on FPGA board.
5) To carry out study the RISCV processors for many further application

IV. EXISTING SYSTEM

OpenRISC is an existing system present which is aimed at developing an open source ISA based on RISC principles. OpenRISC implements architecture with 16 or 64 general purpose registers (64) and a fixed 64-bit instruction length. Two mainline processor core implementations for OpenRISC are OR1200 and mor1kx. Although not actively developed, OR1200 is the first original widely used implementation of the processor in Verilog HDL. Mor1kx is a novel implementation which is more refined and has diversities with respect to tightly coupled memory, presence of a delay slot or the number of pipeline stages. A number of system-on-chips (SoC) are available for OpenRISC that are used to perform RTL simulations, SystemC simulations or an FPGA synthesis of OpenRISC-powered entire system.

V. PROPOSED SYSTEM

Our proposed system presents hardware design architecture to realize RV64I base integer instruction set for 64-bit address space. The implementation is single core, in-order, non-bus based, single cycle architecture with full support for RV64I base integer instruction set. The application domains include acoustic signal processing, real-time embedded systems, sensor technology and myriad other domains. As required by target applications related to Internet of Things (IoT) and other embedded low-cost devices, the focus has been to optimize for price, power and design complexity at the cost of stringent timing constraints. This designed processor is prototyped on an FPGA board. To facilitate our required works in RISCV based tiny processor, a suite of tools and test framework around RISC-V were created targeted at 64-bit architectures. The work includes an outline of the developed framework and its use on the preliminary design.

RISC-V (RV64I) INSTRUCTION SET

For an increasing number of application domains, 2^{32} bytes of addressable memory are insufficient. Large servers in 2015 have as much as 64 TB of DRAM, requiring 46 bits to fully address. Even some wireless phones have exceeded 4 GB of DRAM. Hence, while RV32I is appropriate for most small systems, its limited address space renders it unusable for many others. The RV64I base ISA addresses the lack of addresses. As Figure 1 depicts, RV64I's user-visible state is very similar to RV32I's it differs only in the widths of the integer registers and the program counter, which have all doubled in width to 64 bits. In the same spirit, the RV32I instructions perform the same function as in RV64I, except that they operate on the full 64-bit register. There are 12 new instructions in RV64I.

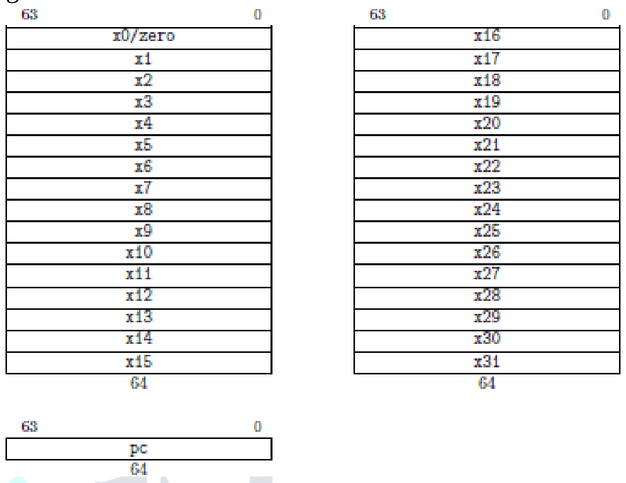


Figure 1: RV64I user-visible architectural state

While integer arithmetic often operates on the full width of a register, especially for the purpose of manipulating addresses, computations on sub-word quantities are also quite common. This effect is amplified in 64-bit architectures, since popular data types like int in Java and C remain 32 bits width. To maintain reasonable performance on 32-bit code, RV64I adds several computational instructions that operate on the lower 32 bits of the integer registers and produce 32-bit results, which are sign-extended to the full width of the register. Keeping 32-bit data sign-extended in the registers keeps casts between the C type's int and unsigned int, and between int and long, costless operations. The existing branch instructions also automatically work on both signed and unsigned 32-bit types. There are also three new memory access instructions: LWU loads a 32-bit word and zero-extends the result to 64 bits. LD and SD load and store 64-bit double-words.

TABLE I
INSTRUCTION TYPE AND OPCODE

Instruction Category (Type)	RV32I Instruction	Example
Integer Register-Register Instructions (R-Type)	add, sub, sll, slt, sltu, xor, srl, sra, or, and	*add* rd,rs1,rs2 The content of register rs1 is added with that of rs2 and the result is stored in register rd.
Integer Register-Immediate Instructions (I-Type)	addi, slti, sltiu, ori, xori, andi, slli, srli, srai	*xori* rd,rs1,imm XOR operation is performed on the content of register rs1 and immediate value and the result is stored in register rd.
Integer Computational Instructions (U-Type)	lui, auipc	*lui* rd,imm The immediate value is placed in the top 20 bits of the register rd, filling in the lowest 12 bits with zeros.
Control Transfer Instructions -Unconditional Jumps (UJ, I-Type)	jal, jalr	*jal* rd,imm The immediate offset is sign -extended and added to PC to form the jump target address. The address of the instruction following the jump (PC+4) is stored into register rd.
Control Transfer Instructions -Conditional Branches (SB-Type)	beq, bne, blt, bltu, bge, bgeu	*beq* rs1,rs2,imm The 12-bit immediate value is added to the current PC to give the target address. Branch is taken if the contents of register rs1 and rs2 are equal.
Load Instructions (I-Type)	lw, lh, lb, lhu, lbu	*lw* rd,rs1,imm A 32-bit value is copied from memory to register rd. The effective byte address is obtained by adding register rs1 to the sign-extended 12-bit offset.
Store Instructions (S-Type)	sw, sh, sb	*sw* rs1,rs2,imm The 32-bit value in register rs2 is copied to memory. The effective byte address is obtained by adding register rs1 to the sign-extended 12-bit offset.

TABLE II
INSTRUCTION ENCODING

31 27 26 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]		rs1	funct3	rd	opcode	I-type
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12/10:5]	rs2	rs1	funct3	imm[4:1/11]	opcode	SB-type
imm[31:12]				rd	opcode	U-type
imm[20/10:1/11/19:12]				rd	opcode	UJ-type

RV64I Base Instruction Set (in addition to RV32I)

A dedicated adder is also included for incrementing the PC in each cycle. The control unit is purely combinatorial with all the control signals generated in the same cycle. Dedicated signals have been provided for both internal and external exceptions. The register bank and ALU logic module comprises of the 63 general purpose registers, the register addressing, read and write logic as well as arithmetic logic unit (ALU).

ALU in this module deals only with arithmetic register-register and register-immediate operations. For register-immediate operations, the sign extension, shift and shuffle logic block provides the properly sign extended and reordered 64-bit immediate operand for ALU and the second operand is drawn from the register bank. ALU is not involved in branch and jump target address calculation and dedicated adders have been included for the same.

For branch instructions, ALU calculates the condition and provides the results to the control unit which then decides whether a branch is to be taken. ALU is kept close to the register bank to decrease critical path delay.

There are five different types of immediate formats, named as I, S, B, U and J-type immediate. The encoding of immediate in instruction often has shuffled bits as can be seen in Table II. The decoding involves the proper reordering, left shift or sign extension of the immediate values. Such an encoding is chosen to minimize implementation complexity across ISA extensions.

This type of immediate decoding is handled by the sign extension, shift and shuffle logic module. The memory control logic module is included to accommodate the various store/load operations which are allowed to have word aligned as well as misaligned addresses of the data memory.

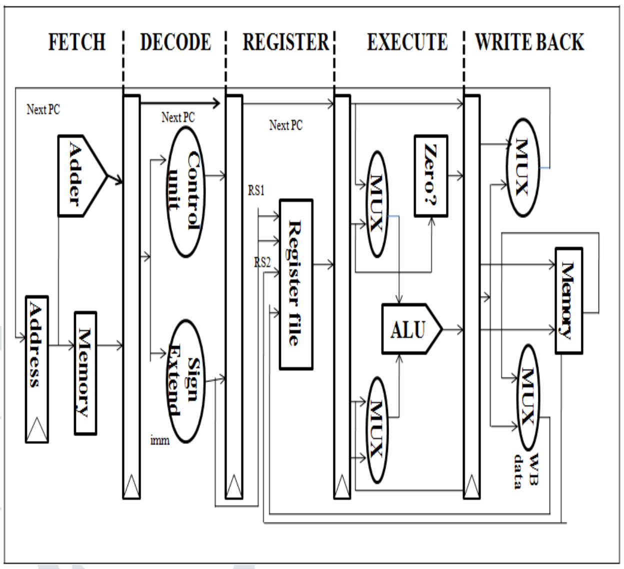


Figure 2: Block Diagram Of Processor

64-bit processor is used for most embedded applications where a tiny processor is required. A single cycle design was chosen for initial baseline performance analysis. The

Subsequent subsections describe about top level and micro-architecture of this design in detail.

At the top level, the processor core can be divided into four logical modules as shown in Figure 2. The fetch, decode and control logic block is responsible for fetching the instruction from the instruction memory, decoding the instruction and generating the control signals. It is also responsible for resolving jump and branch target addresses. It hosts the program counter, the target address selection logic, the instruction memory controller, the instruction decoder and a control unit. A dedicated adder is also included for incrementing the PC in each cycle. The control unit is purely combinatorial with all the control signals generated in the same cycle. Dedicated signals have been provided for both internal and external exceptions. The register bank and ALU logic module comprises of the 63 general purpose registers, the register addressing, read and write logic as well as arithmetic logic unit (ALU). ALU in this module deals only with arithmetic register-register and register-immediate operations. For register-immediate operations, the sign extension, shift and shuffle logic block provides the properly sign extended and reordered 64-bit immediate operand for ALU and the second operand is drawn from the register bank. ALU is not involved in branch and jump target address calculation and dedicated adders have been included for the same. For branch instructions, ALU calculates the condition and provides the results to the control unit which then decides whether a branch is to be taken. ALU is kept close to the register bank to decrease critical path delay. There are five different types of immediate formats, named as I, S, B, U and J-type immediate. The encoding of immediate in instruction often has shuffled bits as can be seen in Table II. The decoding involves the proper reordering, left shift or sign extension of the immediate values. Such an encoding is chosen to minimize implementation complexity across ISA extensions. This type of immediate decoding is handled by the sign extension, shift and shuffle logic module. The memory control logic module is included to accommodate the various store/load operations which are allowed to have word aligned as well as misaligned addresses of the data memory.

Category	Op code	Instructions
Load and Store Instructions	0000011	LWU
	0000011	LD
	0100011	SD
Integer Register-Immediate Instructions	0010011	SLY
	0010011	SRLI
	0010011	SRAI
	0011011	ADDIW
	0011011	SLLIW
	0011011	SRLIW
	0011011	SRAIW
Integer Register-Register Operations	0111011	ADDW
	0111011	SUBW
	0111011	SLLW
	0111011	SRLW
	0111011	SRAW

VI. PIPELINE ORGANIZATION

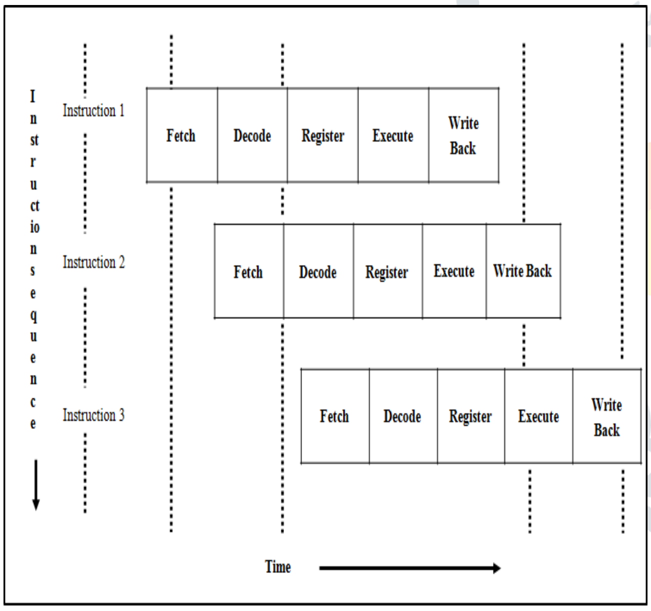


Figure 3: 5 Stage Pipelining of Processor

Proposed architecture comprises of 5 pipeline stages viz. Fetch, Decode, Register Select (RS), Execute (EXE) and Write Back (WB). In fetch stage, instruction is fetched from instruction memory from location indicated by Program Counter (PC). Instruction memory sends instruction to decode stage for instruction information extraction. The extracted information is forwarded to register select stage. From register select stage, the pipeline is split into three concurrent pipeline units, as integer instructions pass through integer execution pipeline, memory instructions pass through the memory pipeline and floating point instructions pass through the FP execution pipeline unit. Write back stage receives the responses from all of these three concurrent execution stages and based on the scheduling of the instruction, WB stage allows commit the instructions in-order.

A. Fetch Stage

It computes the current PC and predicts the next PC value. Fetch unit has a 64 bit register to hold the current PC value, a 64-bit adder to calculate the next PC by adding 4 to the current PC. Next PC is predicted either as PC+4 or by using a sophisticated branch prediction scheme. Branch prediction scheme consists of Branch Target Buffer (BTB) and Branch Predication (BP) unit. BTB unit outputs a Boolean valid signal along with target PC address. BP unit provides Boolean information which contains whether the PC value is present or not. A multiplexer selects either PC+4 or predicted PC as next PC value based on the valid and prediction signals. The PC and next PC are given to the decoder stage for further processing.

B. Decode Stage

Instruction decoder receives the instructions from program memory with corresponding PC and predicted PC from fetch stage. Based on the lower 7-bit (Op-code) of instruction, Subgroup information is decoded from 3-bit field (function field). Address of sources and destination registers are of 5-bit field each and their position is fixed irrespective of type of instruction. If an immediate type instruction has occurred, the immediate data is sign-extended to 64-bit for further processing. Op-code is given to main group classifier, which gives main group class as output. Main group classifier has seven parallel comparators (7-bit comparator) to decode main group class. Instruction op-code is compared against seven categories of op-codes concurrently. If any of the comparators' output goes high, corresponding main group signal is generated otherwise the instruction being decoded is considered as ILLEGAL. Further, the subgroup classifier determines the exact operation to be performed by the instruction. Subgroup is determined by using both 3-bit function field and main class. If 3-bit function field of instruction matches with 3-bit fields of the decoded main class, sub group classifier generates corresponding subgroup category under the decoded main class, otherwise the instruction falls under ILLEGAL. Decoder generates control signals for further processing of the instruction. Control signals include, Register Access, Register update, and pipeline information (the pipeline on which the instruction is to be scheduled). Register Access can be integer register file access (INT-ACCESS) or floating point register access (FP-ACCESS) or NO-ACCESS. Register update gives which register file needs to be written back after executing the instruction. Instruction processing is organized through two independent pipelines for integer ALU / Memory Address computation, and FPU. EXPIPE signal indicates the pipeline through which the instruction has to be processed. Decoded information is forwarded to register select stage in the form of packet along with PC and predicted PC.

C. Register Select Stage

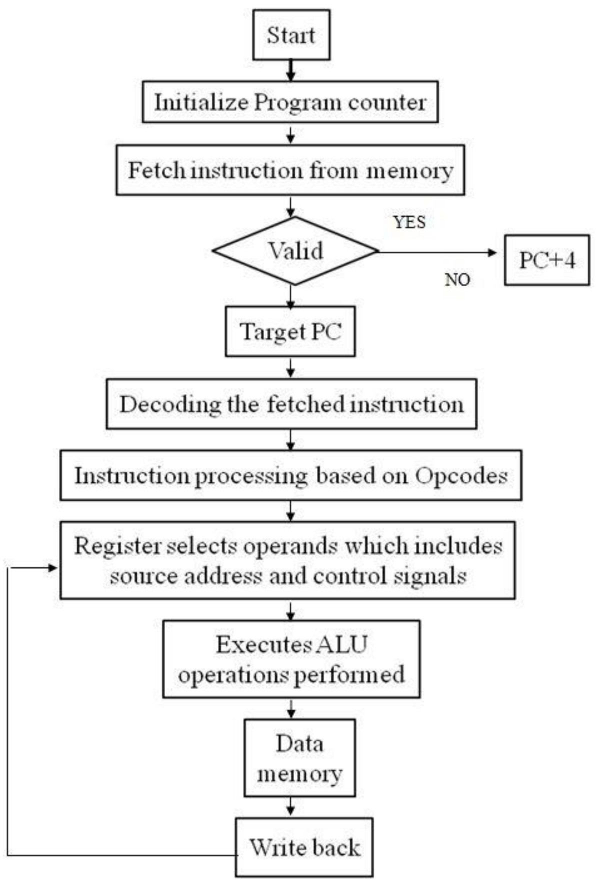
RS stage accepts decoded information of instructions from decoder stage and selects the operand from either integer or floating point register file. RS stage contains an integer and floating point register file. Register file is implemented as a Random Access Memory (RAM), which has a latency of one clock cycle with three read ports and one write port. Register file unit accepts three source addresses (Rd_Addr_1, Rd_Addr_2, Rd_Addr_3) and a control signal (Reg_Access) which specifies the access of Register files (Integer Access or Floating point Access). Register file output is 64-bitwhich holds the data from either integer or floating point register file and has one 64-bit write port for write back. Based on the pipeline information from decode stage, operands after register select are passed either to integer ALU /memory address computation pipeline or FP execution pipeline. An instruction scheduler FIFO which is part of integer / memory / FPU execute stage is used to schedule the instructions in the various pipelines to commit the instructions in-order.

D. Execute Stage

Execute stage consists of three concurrent units for Integer arithmetic and logic operations etc., operand memory address computation and Floating point computation. Integer execution performs arithmetic (Addition, subtraction, multiplication and division) and logical (AND, OR, XOR and shift) operations. Also, Integer arithmetic unit calculates the target address for unconditional or conditional jump and branch instructions. Integer execution unit executes the system related instruction such as SCALL, SBREAK instructions for supervisor level access of the instruction. An efficient data forwarding scheme is used to forward output of execution units to the input of the execution unit. Unit for Memory related instructions calculate the target data memory address for load and store operations. RISC-V ISA supports load or store operations on a byte, half-word and word data to and from data memory.

E. Write Back Stage

Write Back (WB) stage commits the instruction from the pipeline and updates the register file with the results from execute units. WB reads the instruction from top of the scheduled instruction FIFO and based on the pipeline information, it reads either from integer or memory or floating point concurrent units.


Start
Initialize Program counter
Fetch instruction from memory
Valid
YES
NO
PC+4
Target PC
Decoding the fetched instruction
Instruction processing based on Opcodes
Register selects operands which includes source address and control signals
Executes ALU operations performed
Data memory
Write back

Flow chart of Processor

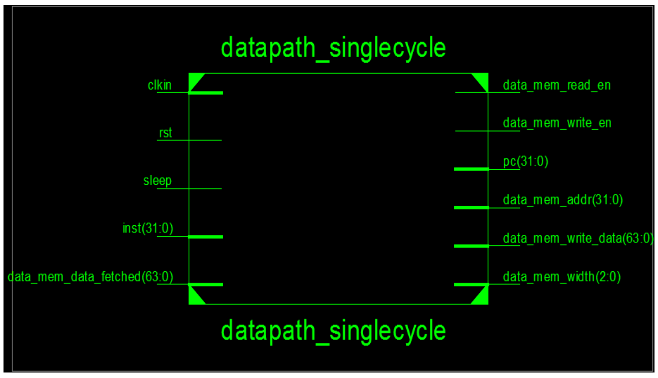
VII. ADVANTAGES

- Faster computation
- Low cost

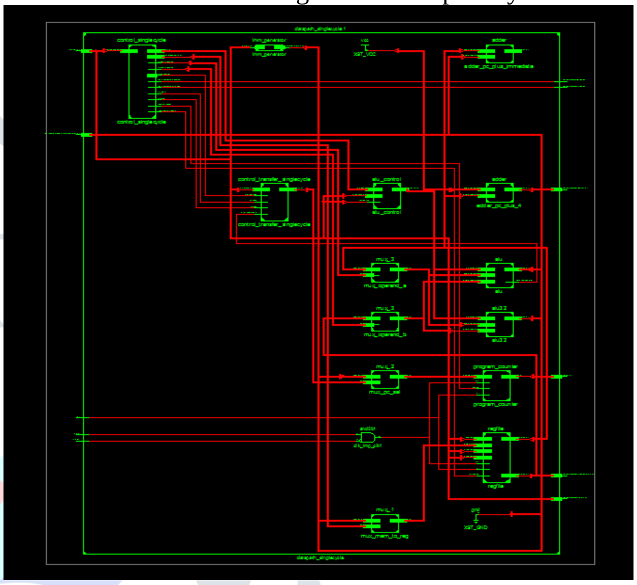
VIII. APPLICATIONS

- VLSI system applications: RISC architectures allow effective use of on-chip transistors in functional units that provide fast access to frequently used operands and instructions.
- Acoustic signal processing: Digital signal processors are paired with microcontrollers in many applications.
- Real-time embedded systems: many systems on chips.
- Sensor technology : Audio and Video processing
- other embedded low-cost devices

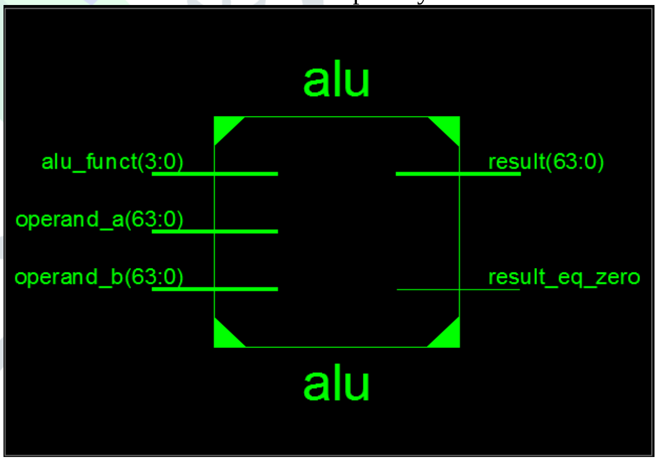
IX. RESULTS



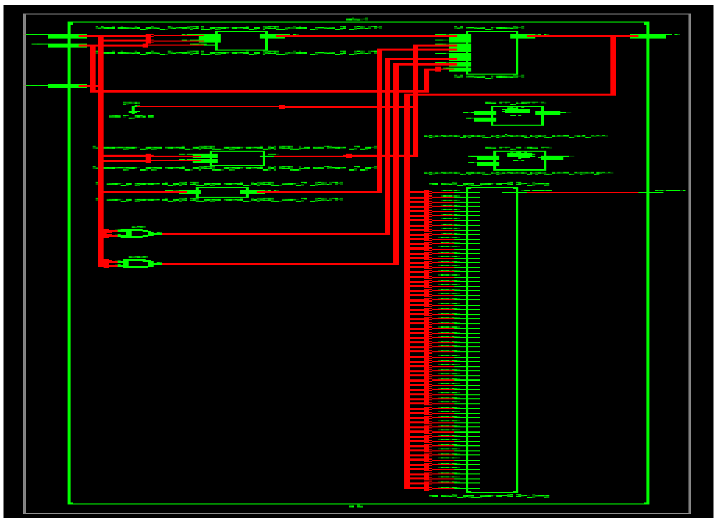
Block diagram of Data path cycle

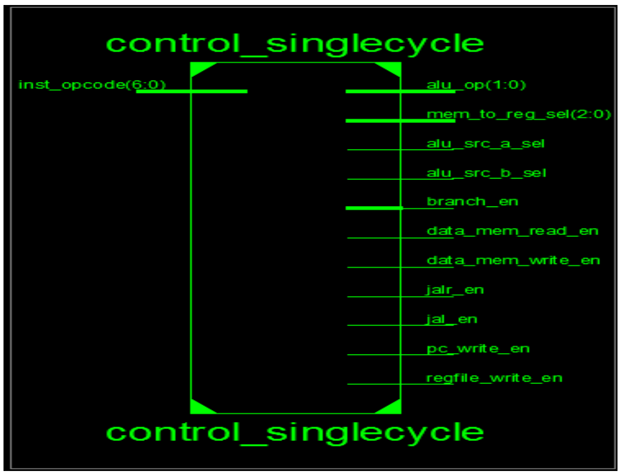


RTL schematic of Data path cycle

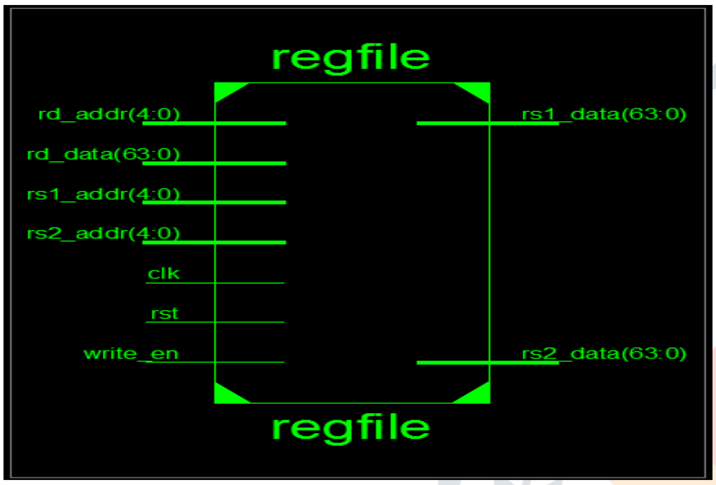


Block diagram of ALU

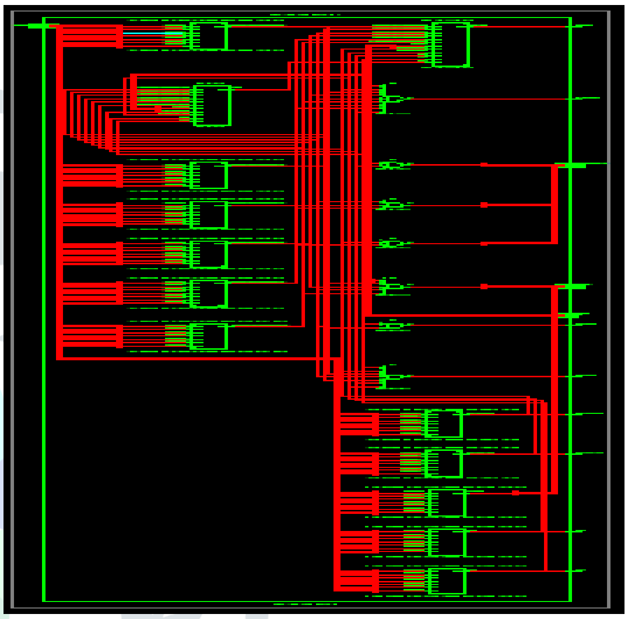

RTL schematic of ALU

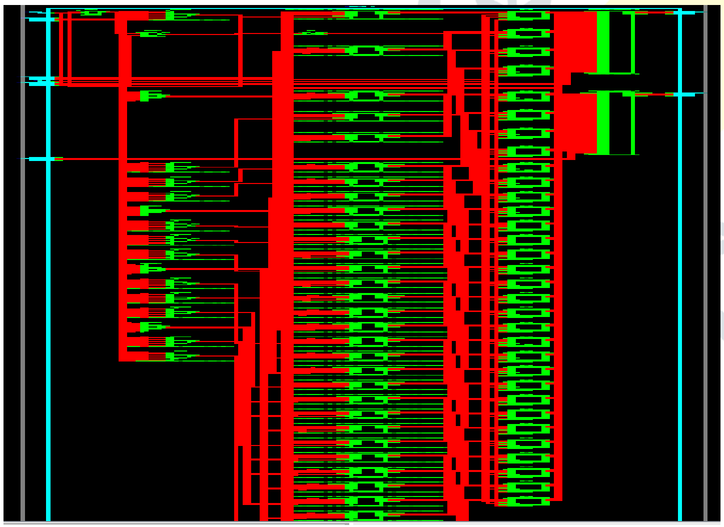


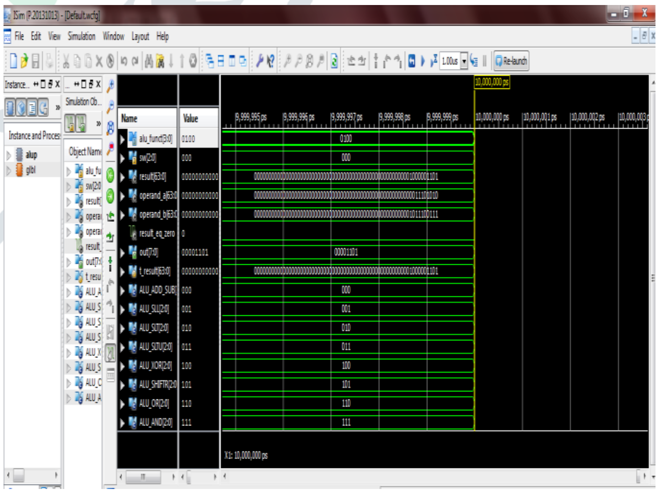
Block diagram of Control unit



Block diagram of Register file


RTL schematic of Control unit


RTL schematic of Register file

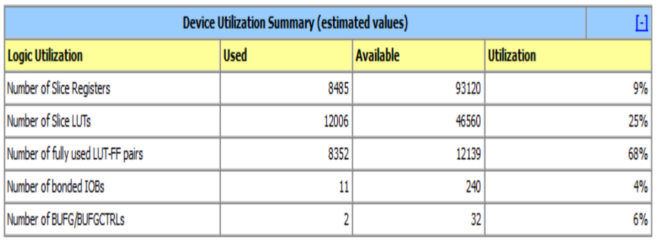


```
Name                Value
alu_funct[3:0]      0100
sw[2:0]             000
result[63:0]        00000000000
operand_a[63:0]     00000000000
operand_b[63:0]     00000000000
result_eq_zero      0
out[7:0]            00001101
t_result[63:0]      00000000000
ALU_ADD_SUB[2:0]    000
ALU_SLL[2:0]        001
ALU_SLT[2:0]        010
ALU_SLTU[2:0]       011
ALU_XOR[2:0]        100
ALU_SHIFTR[2:0]     101
ALU_OR[2:0]         110
ALU_AND[2:0]        111

X1: 10,000,000 ps
```

Simulation results

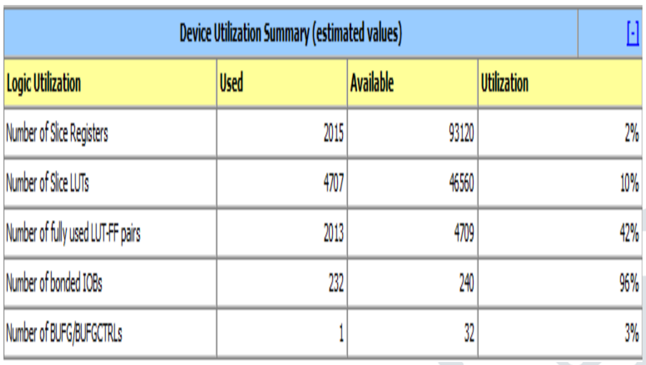
RISC Instruction Set (Maximum Frequency: 303.324MHz)



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	8485	93120	9%
Number of Slice LUTs	12006	46560	25%
Number of fully used LUT-FF pairs	8352	12139	68%
Number of bonded IOBs	11	240	4%
Number of BUFG/BUFGCTRLs	2	32	6%

Design summary of Existing System

RISC-V Instruction Set (Maximum Frequency: 140.398MHz)

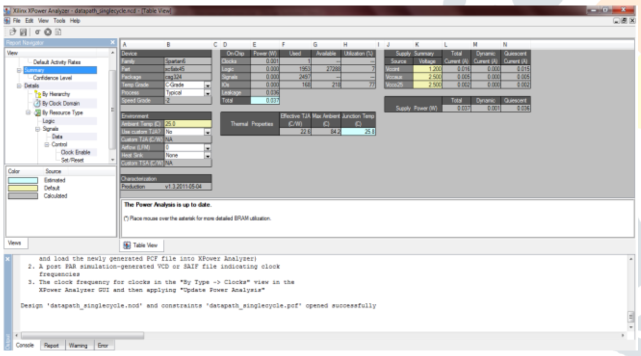


Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	2015	93120	2%
Number of Slice LUTs	4707	46560	10%
Number of fully used LUT-FF pairs	2013	4709	42%
Number of bonded IOBs	232	240	96%
Number of BUFG/BUFGCTRLs	1	32	3%

Design summary of Proposed System

X. POWER ANALYSIS

Whereas power consumption points of view this RISC V processor consumes 0.037 W within the frequency range of 140.398MHz.



Device	
Family	Spartan6
Part	xc6slx45
Package	csg324
Temp Grade	C-Grade
Process	Typical
Speed Grade	-2

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.001	1	---	---
Logic	0.000	1953	27288	7
Signals	0.000	2497	---	---
IOs	0.000	168	218	77
Leakage	0.036			
Total	0.037			

Supply Source	Summary Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.200	0.016	0.000	0.015
Vccaux	2.500	0.005	0.000	0.005
Vcco25	2.500	0.002	0.000	0.002

	Total	Dynamic	Quiescent
Supply Power (W)	0.037	0.001	0.036

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0
Heat Sink	None
Custom TSA (C/W)	NA

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	22.6	84.2	25.8

Characterization	
Production	v1.3.2011-05-04

The Power Analysis is up to date.

(*) Place mouse over the asterisk for more detailed BRAM utilization.

```
   and load the newly generated PCF file into XPower Analyzer)
 2. A post PAR simulation-generated VCD or SAIF file indicating clock
    frequencies
 3. The clock frequency for clocks in the "By Type -> Clocks" view in the
    XPower Analyzer GUI and then applying "Update Power Analysis"

Design 'datapath_singlecycle.ncd' and constraints 'datapath_singlecycle.pcf' opened successfully
```

Hierarchy based Power analysis

XI. CONCLUSION

In this paper, we proposed an FPGA based pipelined 64-bit RISC-V processor. Verilog is used to design modules; several modules are integrated in single module. To simulate the circuit, it needs to write a test bench for getting different output for different inputs at different interval. Xilinx ISE 14.7, ISIM 14.7, to synthesize and simulate the design. Translation, mapping and routing done using Xilinx ISE 14.2 and Plan Ahead 14.2. Finally bit file is created to program Spatran-6 FPGA. The enormous complexity of modern microprocessor designs poses significant challenges in the usage of these systems.

Branch predictions will increase flow in instruction pipeline and achieve high effective performance. The proposed architecture is able to prevent pipeline to multiple executions with a single instruction. The proposed design can access more data processing for data intensive applications like packet processing. This 64-bit RISC-V processor consumes only 1 instruction, whereas 32-bit RISC-V processor needs more than 1 instruction. As it is Single-cycle processor an instruction is fetched from memory, it is executed, and the results are stored all in a single clock cycle. Single-cycle processors tend to be the most simple in terms of hardware requirements, and they are easy to design. As per the design summary it is obvious that the proposed system uses minimum hardware resources and found that the maximum operating frequency is 140.398MHz. The power is estimated to be 0.037 W using Xilinx Power Analyzer. The proposed design can find its applications in high configured robotic work-stations such as, portable pong gaming kits, smart phones, ATMs.

REFERENCE

[1] K. Asanovi and D. A. Patterson, “Instruction sets should be free: The case for risc-v,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146, Aug 2014.
[2] A. Waterman, Y. Lee, D. A. Patterson, K. Asanovic, V. I. U. level Isa, A. Waterman, Y. Lee, and D. Patterson, “The risc-v instruction set manual,” 2014.
[3] V. Patil, A. Raveendran, P. M. Sobha, A. D. Selvakumar, and D. Vivian,“Out of order floating point coprocessor for risc v isa,” in 2015 19th International Symposium on VLSI Design and Test, June 2015, pp. 1–7.
[4] Aneesh Raveendran, Vinayak Baramu Patil, David Selvakumar, Vivian Desalphine, “A RISC-V Instruction Set Processor-Micro architecture Design and Analysis”, 2016 International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA)