DESIGN AND REALIZATION OF THE RISC-V INSTRUCTION SET ARCHITECTURE

BOLLAM VAISHNAVI YADAV [1], Dr. M. RENU BABU[2],Dr. SK. UMAR FARUQ[3].

1 Ug Scholar, Department of Electronics and Communication Engineering, TEEGALA KRISHNA REDDY Engineering College, Hyderabad.

2 Associate Professor, Department of Electronics and Communication Engineering, TEEGALA KRISHNA REDDY Engineering College, Hyderabad.

3 Professor, Department of Electronics and Communication Engineering, TEEGALA KRISHNA REDDY Engineering College, Hyderabad.

ABSTRACT

The development of a fully synthesizable 32-bit processor using the open-source and free RISC-V (RV32I) ISA is described in this work. Low-Cost embedded devices were considered when designing this CPU. The framework for RISC-V development and validation, which includes putting together tools and automated test suites, is also provided in this document. The final processor will be a RISC-V processor with a single core and simple hardware. Verilog HDL is used to develop the recommended CPU, and a "Artix-7" FPGA board is used to further prototype it. This shows that the maximum operational frequency is 32MHz. With an initial focus on FPGA design and then SoC design later on, this article describes the design chores from high architectural level descriptions down to RTL and then proceeding through logic synthesis and physical design to get the layout ready for its final tape out in CMOS 90 nm technology.

Keywords – RISC(Reduced Instruction Set Computer),RISC-V, RV321I ISA, Micro Architecture, FPGA, Soc

INTRODUCTION

The RISC-V instruction set architecture (ISA), which was first developed to support research and education in the field of computer architecture, is now poised to become a popular free and open architecture for usage in both academic and industrial applications. To be successful and be widely accepted, RISC-V has been designed to support 32-bit, 64-bit, and 128-bit address spaces. The ISA is

broken down into a minuscule base integer ISA, a fundamental set of instructions appropriate for operating systems, linkers, compilers, and assemblers. The mentioned suits are a part of the tool chains that the RISC V foundation provides that are interoperable with one another.

Low level applications and mobile systems were both constructed using RISC architectures during the beginning of the twenty-first century. The market for affordable, low-power embedded systems is dominated by RISC-based ARM architectures. The Apple iPhone, iPad, and the vast majority of Android-based mobile devices all use ARM architecture.

Currently, gaming systems like the Nintendo 64, PlayStation Portable, and Linksys WRT54G series of home gateways employ the MIPS line. SuperH (SH), another 32-bit RISC ISA, was developed by Hitachi. SuperH2 is being implemented as open-source hardware under the name J2 due to the impending expiration of numerous SuperH patents.

Open RISC's objective is to provide an open-source ISA. Utilizing RISC-based principles. Open RISC supports architectures with 16 or 32 (32/64-bit) general-purpose registers of fixed capacity and 32-bit instruction length. OR1200 and mor1kx are two implementations of the mainline processor core for Open RSIC. Despite not being actively developed, the OR1200 is the first authentic, widely used Verilog HDL implementation of the CPU. A novel implementation known as Mor1kx is more complex and features variations in terms of memory proximity, the presence of a delay slot, and the number of pipeline stages. Several system-on-chip (SoC) devices for Open RISC can execute RTL simulations, System C simulations, or an FPGA synthesis of an entire system.

LITERATURE REVIEW

The RISC-V instruction set architecture (ISA), an open standard developed at the University of California, Berkeley, has significantly influenced processor design due to its modularity, extensibility, and open-source nature. Waterman et al laid the foundation with a comprehensive specification of the RISC-V user-level ISA, advocating for its openness and flexibility. Asanović and Patterson further argued for the freedom and transparency of instruction sets, establishing RISC-V as a platform for academic and industrial innovation.

Several studies have focused on extending RISC-V for high-performance computing. Zimmer et al introduced a vector processor built on RISC-V,

integrating switched-capacitor DC–DC converters for energy efficiency in 28 nm FDSOI technology. Lee et al demonstrated a 45 nm implementation of a RISC-V processor achieving 16.7 GFLOPS/W with vector accelerators, indicating its potential in scientific computing and embedded AI.

In terms of custom processor design, Qui et al implemented a dynamically scheduled Very Long Instruction Word (VLIW) processor using a 256-bit RISC-V-based architecture on FPGA, showcasing its reconfigurability and potential in parallel processing. Similarly, Patil et al developed an out-of-order floating-point co-processor for RISC-V, aiming at accelerating floating-point computations, a critical component in multimedia and AI applications.

Beyond processors, significant efforts have been made in developing the tool chains and interconnects supporting RISC-V. The *TileLink* specification and the *Open SoC Debug* framework offer scalable and open-source support for debugging and system-level communication. Crossley et alcontributed with the BAG framework—an automated analog and mixed-signal design generator, compatible with RISC-V integration for analog-intensive SoCs.

Educational and documentation resources such as Farquhar and Bunce's MIPS Programmer’s Handbook provide foundational knowledge for comparing traditional ISAs like MIPS with RISC-V, while modern efforts such as the Red RISC-V initiative continue to expand the ecosystem with collaborative community-driven development.

Emerging trends in low-power and intelligent systems also show adoption of RISC-V in diverse areas. Arul et al proposed intelligent power control models for IoT wearable devices using RISC-V-based models in Body Area Networks (BAN), while Krishna Kumrai et al explored deep learning methods such as CNN-Inception and ResNet-50 for automatic facial expression recognition, potentially benefiting from custom RISC-V accelerators for edge AI.

Finally, innovations in device-level design such as Dual Gate CNTFETs illustrate hardware advancements that could synergize with RISC-V-based platforms for future nanoelectronics. Furthermore, green semiconductor processing methods explored by Furgal and Lenora stress the need for environmentally conscious manufacturing practices in tandem with hardware design.

The RISC-V Instruction Set Architecture (ISA) has emerged as a groundbreaking open-source alternative to proprietary ISAs, enabling innovation in processor design and implementation. Developed at the University of California, Berkeley, the foundational work by Waterman et al laid out the specifications of the base user-level ISA, promoting modularity and extensibility. Further reinforcing this philosophy, Asanovic and Patterson emphasized that instruction sets should be freely available to foster academic and industrial research.

PHYSICAL MODEL DESCRIPTION

The core of the system is a 32-bit single-cycle RISC-V processor that adheres to the RV32I base instruction set. The term single-cycle means that each instruction is fetched, decoded, executed, and written back within one clock cycle. This design strategy reduces hardware complexity and increases power efficiency, making it especially beneficial for low-power embedded devices.

The processor is developed using Verilog HDL (Hardware Description Language) and tested on an FPGA (Field Programmable Gate Array). This prototyping approach allows developers to validate the design before transitioning to an ASIC (Application-Specific Integrated Circuit), which would be used for mass production. An important advantage of this project is its open-source and modular design, which not only enables customization but also aligns with the global RISC-V movement promoting innovation without licensing constraints.

The block diagram of the proposed RISC-V processor architecture plays a critical role in visualizing its modular structure and operational flow. At the top level, the processor is divided into four main components: the Fetch-Decode-Control Unit, the Register Bank with the ALU, the Sign Extension and Shuffle Logic, and the Memory Control Logic. The Fetch-Decode-Control Unit is responsible for retrieving instructions from memory, decoding them, generating control signals, and calculating target addresses for jumps and branches. It consists of key subcomponents like the control unit, instruction decoder, program counter, and instruction memory controller.

The Register Bank and ALU module houses 31 general-purpose registers and performs arithmetic and logical operations. It takes operands from the register bank and uses the ALU to execute register-register or register-immediate instructions. The Sign

Extension and Shuffle Logic module decodes immediate values based on instruction type, supporting five distinct formats and ensuring compatibility with the RISC-V ISA. The Memory Control Logic enables both aligned and unaligned memory access by generating appropriate read/write signals and adjusting data based on offset positions. Due to its single-cycle design, the architecture duplicates some logic to balance speed and component reuse. Overall, the block diagram reflects a compact yet powerful architecture suitable for edge IoT and embedded applications.

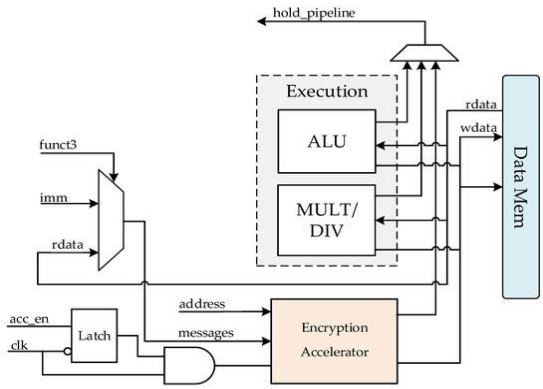

Fig: Block Diagram

Block Diagram Description

The block diagram provides a clear visual representation of the processor's architecture. It includes several crucial functional blocks:

MUX(Multiplexer):

This unit selects between an immediate value and register data based on the instruction type. The selected input is then forwarded to the execution stage. It plays a critical role in instruction flexibility and operational control.

Execution Block:

ALU (Arithmetic Logic Unit): Responsible for executing basic arithmetic and logic operations such as addition, subtraction, AND, OR, etc.

MULT/DIV Unit: This separate unit handles multiplication and division operations. Including this block ensures the processor can support a broader range of computations.

DataMemory:

This block handles load and store instructions. It communicates with both the execution unit and the encryption accelerator to allow secure reading and writing of data. The interaction with memory is vital for managing operational data and temporary results.

EncryptionAccelerator:

An essential feature in modern IoT systems is data security. This module performs on-the-fly encryption and decryption of data being accessed or transmitted. By integrating encryption within the processor pipeline, secure communication and data integrity are ensured, especially important for edge devices interacting with cloud systems.

Latch:

A control unit that holds the acc_en signal synchronized with the clock. It ensures the timing and synchronization of control signals, enabling smooth operation and correct sequencing of processor tasks.

PipelineControl:

Though single-cycle in nature, the processor includes a pipeline hold control to manage scenarios where an operation may need to stall temporarily due to data dependencies or control flow changes.

METHODOLOGY

RISCV Methodology

RISC-V (Reduced Instruction Set Computer - Five) is an open-standard instruction set architecture (ISA) based on established RISC principles. The diagram represents the five primary stages of a RISC-V processor pipeline: Instruction Fetch, Instruction Decode, Register File Access, Execute, and Write Back, along with two essential support units: Control Unit and Data/Instruction Memory.

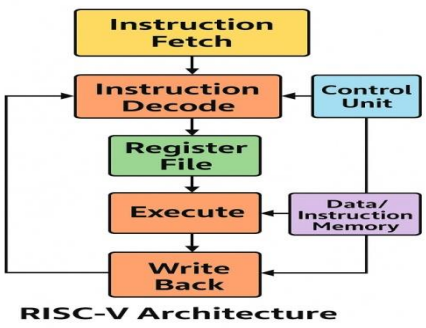

Fig: RISC-V Architecture.

Instruction Fetch (IF)

This is the first stage in the processor pipeline where the next instruction is fetched from memory. The Program Counter (PC) holds the address of the instruction to be executed. This address is sent to memory to retrieve the instruction, which is then passed on to the decode stage. The PC is then updated to point to the next instruction. This step is crucial because it starts the execution process.

Instruction Decode (ID)

In this stage, the fetched instruction is analyzed and interpreted. The opcode (operation code) part of the instruction is examined to determine the type of operation to be performed (e.g., arithmetic, memory access, branching). Based on this decoding, the Control Unit generates the necessary control signals to direct the subsequent stages.

Register File

Once the instruction is decoded, the operands required for execution are fetched from the Register File. The Register File is a collection of CPU registers. Typically, RISC-V uses 32 general-purpose registers (x0 to x31), where x0 is hardwired to zero.

Execute (EX)

This is the core computational stage where arithmetic or logic operations are performed using the ALU (Arithmetic Logic Unit). The operation performed depends on the decoded instruction and the control signals generated earlier. This stage may also include branch calculations or effective address generation for memory instructions.

Write Back (WB)

The final stage in the pipeline involves writing the result of the execution (or loaded data from memory) back to the Register File. This ensures that the outcome of the instruction is stored and can be used by subsequent instructions.

Control Unit

The Control Unit is essential for orchestrating how each part of the processor functions. It interprets the decoded instruction and sends out signals to activate specific operations in each stage, such as reading from memory, selecting ALU operations, or writing back to registers.

Data/Instruction Memory

This component represents the unified or separate memory units for storing both data and instructions. In RISC-V, memory access is typically handled in the execute stage for efficiency.

RESULT

The image below depicts a detailed gate-level schematic of a digital processor system, likely representing the RTL (Register Transfer Level) implementation of the previously discussed 32-bit RISC-V processor. This schematic shows the intricate network of logic gates, multiplexers, registers, control units, and data paths that collectively perform instruction processing. Each block and interconnection reflects how the Verilog code is synthesized into hardware logic, intended for simulation or FPGA deployment.

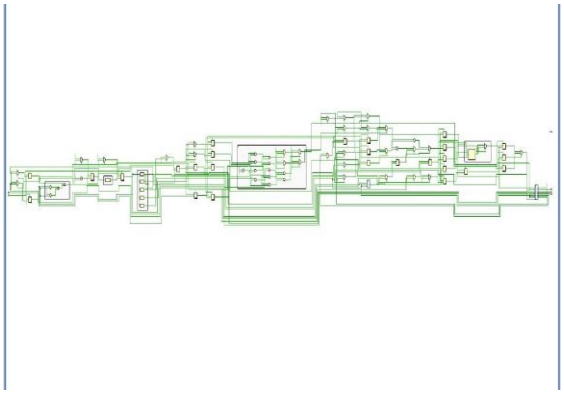

Fig: RTL schematic diagram of RISC-V

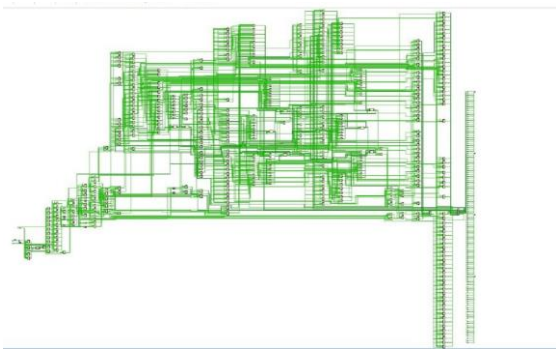

Fig: Layout View of RISC-V

Fig: Waveforms of RISC-V

CONCLUSION

A unique single-cycle small embedded processor design that is modular and greatly expandable has been implemented by the authors of the current research. For data memory and instruction memory, on-chip block RAMs of 16KB and 64KB are used. A 32MHz maximum clock speed is achieved by the CPU using a Xilinx Vivado FPGA chip with a total power consumption of 7.9 mW. The current processor design paves the way for future implementations for specific and general IoT and embedded applications, considering the advantages of the expanding RISC V community as well as the existing tool-chain and software around this new instruction set and based upon this design paradigm.

FUTURE SCOPE

The implementation of a RISC-V based processor for edge IoT platforms opens numerous avenues for future development. With its modular and extensible architecture, this design can be adapted and scaled to meet evolving demands in embedded systems. One significant scope lies in enhancing the processor's functionality by integrating I/O bus interfaces, interrupt handling mechanisms, and support for floating-point operations through a co-processor. These additions would make the processor suitable for a broader range of real-time and data-intensive applications. Furthermore, incorporating multi-level cache systems would significantly improve execution efficiency, especially in complex IoT environments.

REFERENCES

[1] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, “The RISCV instruction set manual, volume I: User-level ISA, version 2.0,” EECS Dept., Univ. California at Berkeley, Berkeley, CA, USA, Rep. UCB/EECS-2014-54, May 2014. [Online]. Available: http://www2.eecs. berkeley.edu/Pubs/TechRpts/2014/EECS-2014-54.html

[2] B. Zimmer et al., “A RISC-V vector processor with simultaneousswitching switched-capacitor DC–DC converters in 28 nm FDSOI,” IEEE J. Solid-State

Circuits, vol. 51, no. 4, pp. 930–942, Apr. 2016.

[3] Y. Lee et al., “A 45 nm 1.3 GHz 16.7 double-precision GFLOPS/W RISC-V processor with vector accelerators,” in Proc. 40th Eur. Solid-State Circuits Conf. (ESSCIRC), Sep. 2014, pp. 199–202. [Online]. Available: http://ieeexplore.ieee.org/document/6942056/I.S. 1963, pp. 271-350.

[4] J. C. Furgal and C. U. Lenora, ``Green routes to silicon-basd materials and their environmental implications," Phys. Sci. Rev., vol. 5, no. 1, p. 24,

[5] N. M. Qui, C. H. Lin, and P. Chen, ``Design and implementation of a 256 bit RISC-V-based dynamically scheduled very long instruction word on FPGA," IEEE Access, vol. 8, pp. 172996 173007, 2020.

[6] 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.

[7] E. Farquhar and P. Bunce, The Mips Programmer’s Handbook, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. S. Williams, “vvp(1) - linux man page,” 2001, [Online; accessed 8 January-2017]. [Online]. Available: https://linux.die.net/man/1/vvp.

[8] 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.