Girish M, 2026, 14:1
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Design and Implementation of SIMD based RISC V Processor

Girish M, Gaganashree M, Inchara H M, Likhitha B, Moulya B N

[1]Professor Electronics and Communication Engineering ATME College Of Engineering Mysuru, India

[2]Electronics and Communication Engineering ATME College Of Engineering (VTU Affiliated) Mysuru, India

Abstract- High-performance computing is becoming more necessary, and this requires processors that can effectively utilize parallelism. Packed Single Instruction, Multiple Data (SIMD) instructions provide an attractive workaround by concurrently executing the same operation on several data pieces. To obtain notable performance benefits for data-intensive tasks, this article covers the design and implementation of a processor that takes advantage of the capabilities of packed SIMD instructions, with an emphasis on the P extension.

Keywords: Processor Architecture, RISC-V, P Extension, SIMD, Parallelism, Performance Optimization, Hardware Design, Instruction Set Architecture.

I. INTRODUCTION

Modern CPUs have moved toward techniques such as SIMD because of the demand for higher performance through processing. In SIMD, a single instruction operates on multiple data simultaneously. Workloads that could be parallelized thus significantly benefit in terms of performance boosts. In this project, we focus on designing a packed SIMD instruction set customized for the RISC-V architecture, making use of the P-extension to support fixed- point operations on integer registers.

The RISC-V P-extension is known for compact, low-power implementations; hence, it perfectly fits the use case in embedded and real-time systems where efficiency is at stake. Our design will seamlessly incorporate the P- extension-based SIMD instructions into the prevalent RISC-V framework, allowing parallel data processing with no additional hardware overhead.

As part of this work, we also investigate how data flows through the CPU data path to support these new SIMD operations; this includes looking at how the functional units process packed data as well as how the new operations introduced by the P-extension are executed.

Along with the data path, there is an important control path whose role is also carefully defined in managing instruction fetching, decoding, execution, and data flow. Extra attention has been given to how the control unit manages the extra functionality and complexity introduced by the P-extension instructions.

Finally, the design details the processor's timing requirements in terms of clock speed and cycle counts to specify how much time will be required by the execution of all standard RISC-V instructions as well as new packed SIMD operations.
These elements collectively serve to form a complete, efficient packed SIMD instruction set for the RISC-V architecture with P-extensions: ISA design, data path integration, control logic, and timing analysis.

Objectives

1. **Study and Survey:**
- Conduct an in-depth study and survey of existing SIMD instruction sets across different architectures.
- Analyze their strengths, weaknesses, and applicability to the RISC-V architecture.
- Identify common data types and operations used in data-intensive tasks across various application domains.

2. **Design:**
- Design a Packed SIMD Instruction set for the RISC-V architecture.
- Design efficient instruction formats and encoding schemes for high instruction density.
- Ensure the instruction set supports a wide range of data-intensive applications, including imaging processing, multimedia encoding/decoding, scientific computing, and machine learning algorithms.

3. **Analysis:**
- The performance and efficiency of the designed instruction set will be evaluated by simulation.
- Measure the benefit in terms of instruction density, power consumption, and execution throughput against existing solutions.
- Analyze its compatibility and integration with the existing RISC-V software ecosystem

The paper is organized as follows. Section II describes the working of the basic SIMD operation. Section III describes the methodology for implementing the single instruction multiple data-based RISC-V processor. Section IV discusses the simulation result. Conclusion is briefly explained in Section V.

II. WORKING

In this project we have implemented single instruction multiple data, which consists of two main components: the control unit and the processor array.

a. Program Counter (PC): This is a register within a processor that holds the address of the succeeding instruction that is to be executed. When an instruction is fetched from memory, it increments the PC so that it points to the succeeding instruction, which, in turn, assures orderly program execution.

b. Instruction Memory (IM): Stores all program instructions in ROM or flash memory. The PC sends an address to the IM, which gives output as a corresponding instruction to be executed by the processor.

c. Decoder: A decoder can be described as a device that is very significant in both digital electronics and computing. It selects one out of multiple outputs by using an input signal, which is determined by the configuration at the input.

d. Register File: It is a collection of registers within a computer processor that store data temporarily during program execution. These registers are small, high-speed storage units. It holds operands, intermediate results, and control information needed for executing instructions efficiently.

e. Arithmetic Logic Unit (ALU): It is the major building block of a computer's central processing unit that takes on most of the workload in executing arithmetic and logical operations on input data. It performs various operations like addition, subtraction, multiplication, division, and bitwise AND, OR, and XOR.

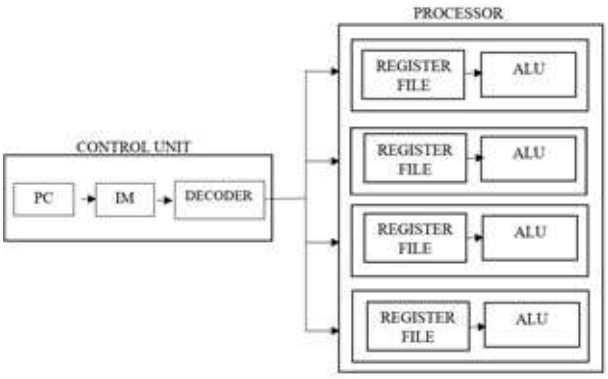

Fig 1. Microarchitecture of SIMD

The detailed description of the working of SIMD based RISC-V processor is shown below stepwise:

In this micro-architecture of the SIMD, the control unit and the array of processors can cooperate to perform one instruction on different data simultaneously. The first action that the Program Counter (PC) performs is to save the memory address of the next instruction that is going to be run and pass it through Instruction Memory (IM). This is then given as an IM instruction and the corresponding instruction is given to the Decoder. The Decoder recognizes the direction and creates the control necessary signals to the processor array. Every processor is made of his or her Register File and Arithmetic Logic Unit (ALU).

The input data and intermediate values together with final result are stored in the Register File whereas the arithmetic and logical operations such as addition, subtraction, multiplication, bitwise operations among others are done in the ALU. As it

is a SIMD system, the same control signals are presented to all the ALUs so that they are performing the same operation simultaneously but on the different data which is stored in their respective registers. Through this, SIMD enhances processing speed and efficiency by allowing a number of data operations at once to be executed on a single instruction.

III. METHODOLOGY

The ASIC (Application-Specific Integrated Circuit) design flow shown in fig 3.0 which involves several stages from concept to fabrication, encompassing both front-end and back-end processes. Here, we will focus on the front- end design flow, which primarily deals with the conceptualization and logical design of the chip.

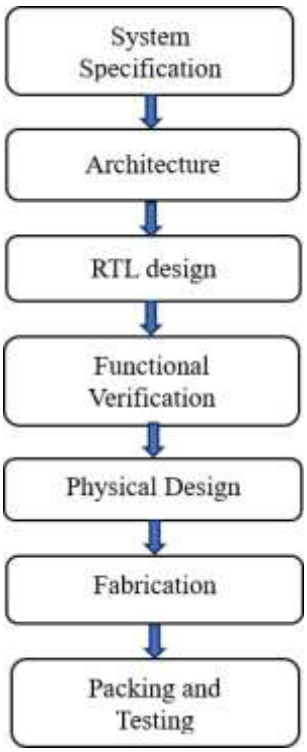

Fig 2. ASIC Design Flow

The designed processor is designed upon the RISC-V specification, specifically RV32I base ISA, and includes support of the P-extension instructions of packed-SIMD operations. The base PCI/ISA has arithmetic, logical, memory access, control transfer, and shift instructions that assign fixed 32-bit opcodes and defined operand fields to each including rd, rs1, rs2, immediate values and funct bits. The P-extension adds functionality to the architecture by having instructions operating on sub-word managed data explosion packed into a 32-bit register packed add, subtract, multiply, and saturating arithmetic are tested with efficiency to perform computations on a DSP or multimedia system. These instructions also use RISC-V encoding rules but they need to have extra functional units within the ALU to allow parallel operations on several pieces of data within a single instruction.

CIU datapath of the CPU is altered to continue with the base functions and at the same time perform parallel SIMD type computations. It is made up of program counter, instruction memory, register file, ALU, P-extension functional unit, data memory and internal buses that enable the free flow of data between blocks. When a P-extension instruction is found encobbling is spilled onto a special section of the ALU which process packed arithmetic and saturation logic. The output is then back written into the destination register in a packed form. To guarantee the right path and destination channels are chosen dynamically, a multiplexing structure is used so that based on the type of instruction, the appropriate path to the source and destination is chosen.

The control path consists of a control unit which receives the instructions, decodes them, executes them, accesses and manipulates the memory, and provides a write-back. Instruction decoder also determines the region of the base ISA as operating or P-extension group by checking the fields of opcode and funct. According to such a classification, the control unit produces the corresponding signals to either turn on the normal ALU or the P-functional unit. Hazard detection and pipeline control is also provided to ensure the right execution flow and prevent erroneous use of data. Branch instructions and jump instructions American and change the program counter without choosing the packed-operation data-path.

The timing of the CPU is configured to allow each base RISC-V aspire to be filled in a small number of clock cycles based on the pipeline stage implementation. When an RV32I pipeline has a 5-stage pipeline structure, most of its instructions take a single cycle to complete, and one instruction takes a single clock to complete in perfect circumstances. Nevertheless, the extra internal cycles might be incurred by P-extension instructions due to parallel arithmetic checks as well as saturation checks, as comprising 1-3 extra cycles, depending on the difficulty of the packed operation. The clock frequency may be selected over a realistic range like 200 to 300 MHz of implementation on FPGA, and so the additional SIMD hardware will not cause timing violations.

IV. RESULTS

The results section highlights the functional verification and performance evaluation of the designed SIMD-based RISC-V processor. The analysis demonstrates that the implemented architecture performs reliably and meets the intended design objectives. The fig below shows t the simulation of the top module, respectively.

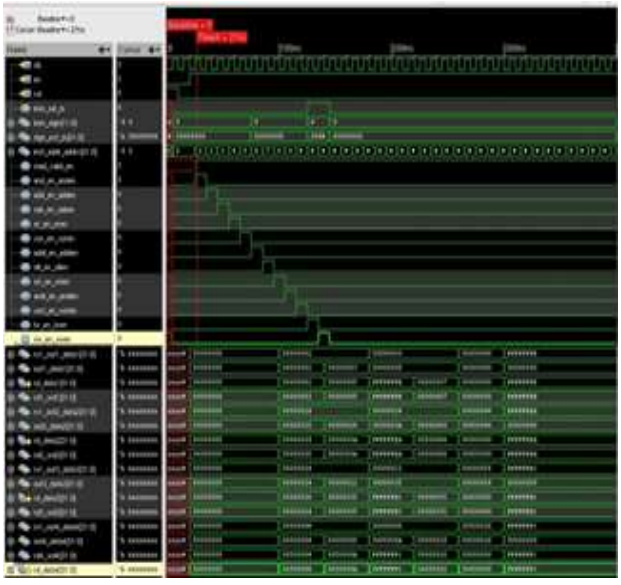

Fig 3. Simulation result of top module

The area and time synthesis reports for top module as shown in the below figures fig9, fig10.

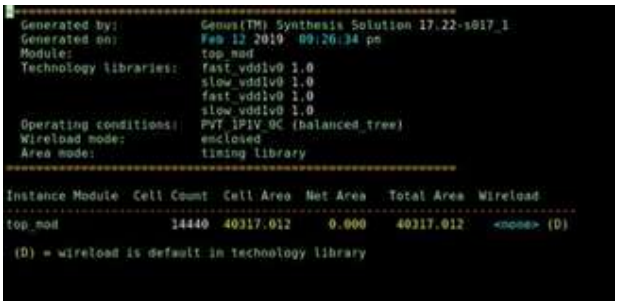

Fig 9. Area synthesis report for the top module

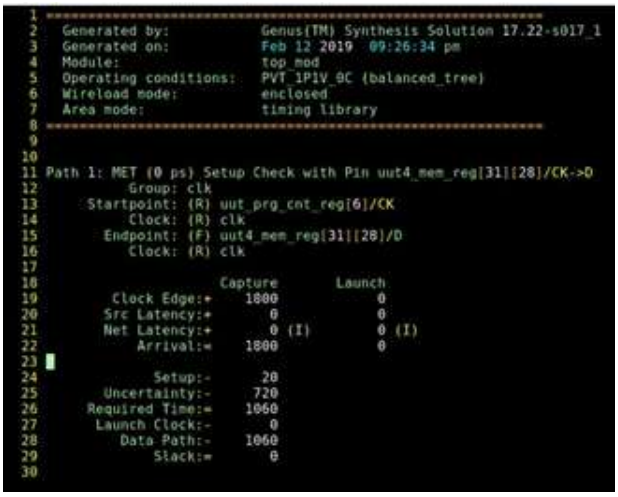

Fig 10. Time synthesis report for the top module

V. CONCLUSION

There are some obvious benefits to the use of a SIMD processor on the RISC-V architecture in the current applications of parallel processing. Owing to its open-source and extensibility, the RISC-V can be extended and hardened to generate targeted SIMD instructions to suit the needs of a particular workload. This flexibility therefore allows an individual operation to be efficiently run in parallel on large numbers of data, and performance improvements are observable in classes of tasks like multimedia processing, scientific computation, and machine learning. Being designed to have a V in RISC-V, the modular architecture of RISC-V and the natural support of operations in a vector- like style, RISC-V is naturally compatible with SIMD extensions; in other words, it can easily be extended to support additions.

Combined, the combination of SIMD methods using the RISC-V platform is a powerful and scalable

solution to run data-intensive tasks and enjoy the limits of the parallel processing performance.

REFERENCES

1. Suaib, Mohammad & Palaty, Abel & Pandey, Kumar. (2011). Architecture of SIMD Type Vector Processor. International Journal of Computer Applications. 20. 10.5120/2418-3233.
2. D.A.M. Koene, "Implementation and Evaluation of Packed-SIMD Instructions for a RISC-V Processor". https://repository.tudelft.nl/islandora/object/uuid:c4162ff8 9419- 4434-852d c1c3297df808/datastream/OBJ/download
3. Ashworth, Mike, Ian J. Bush and Martyn F. Guest. "Vector vs. Scalar Processors: A Performance Comparison Using a Set of Computational Science Benchmarks." (2005).
4. Papaphilippou, P. H. J. Kelly and W. Luk, "Demonstrating custom SIMD instruction development for a RISC-V softcore," 2021 31st International Conference on Field-Programmable Logic and Applications (FPL), Dresden, Germany, 2021, pp. 139- 139, doi: 10.1109/FPL53798.2021.00030.
5. M. Cavalcante, F. Schuiki, F. Zaruba, M. Schaffner and L. Benini, "Ara: A 1-GHz+ Scalable and Energy-Efficient RISC-V Vector Processor with Multi-precision Floating Point Support in 22-nm FD- SOI" in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 28, no. 02, pp. 530-543, 2020. doi: 10.1109/TVLSI.2019.2950087
6. M. Perotti, M. Cavalcante, A. Ottaviano, J. Liu and L. Benini, "Yun: An Open-Source, 64-Bit RISC-V-Based Vector Processor with Multi- Precision Integer and Floating Point Support in 65-nm CMOS," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 70, no. 10, pp. 3732-3736, Oct. 2023, doi: 10.1109/TCSII.2023.3292579.
7. K. Li, W. Yin and Q. Liu, "A Portable DSP Coprocessor Design Using RISC-V Packed-SIMD Instructions," 2023 IEEE International Symposium on Circuits and Systems (ISCAS), Monterey, CA, USA, 2023, pp. 1-5, doi: 10.1109/ISCAS46773.2023.10181681.