Implementation and Evaluation of SIMD Instructions using RISC-V

Prof.Jaswanth V[1], Jeevan Kumar T M[2], K V Sanjay[3] , Manjunath B M[4,]

[1]*Department of Electronics and Communication Engineering, Vidyavardhaka College of Engineering, jaswanthv@gmail.com*
[2]*Department of Electronics and Communication Engineering, Vidyavardhaka College of Engineering, g1kumartm@gmail.com*
[3]*Department of Electronics and Communication Engineering, Vidyavardhaka College of Engineering, sanjaykv1711@gmail.com*
[4]*Department of Electronics and Communication Engineering, Vidyavardhaka College of Engineering, manjunahthbm@gmail.com*

Abstract **- This thesis introduces and explores the design and verification of a packed Single Instruction, Multiple Data (SIMD) instruction set for a RISC-V processor, known as the RISC-V P extension. This extension enhances the RISC-V instruction set architecture by providing packed SIMD support for 8-bit, 16-bit, and 32-bit integer data types. The significance of this architecture lies in its potential to empower developers in constructing more efficient and powerful data-parallel programs for RISC-V processors. This contribution enhances the overall capabilities of the RISC-V ecosystem, providing a valuable extension to the instruction set architecture for data-parallel applications.**

Keywords: Data-parallel Programs, instruction set architecture, RISC-V, P extension, Packed SIMD

1. INTRODUCTION

Single Instruction, Multiple Data (SIMD) is a type of parallel processing in which a single instruction is applied to multiple data items simultaneously. SIMD processors typically have multiple processing elements (PEs), each of which can perform the same operation on a single data item. SIMD is a powerful tool that can be used to accelerate a wide variety of applications, such as graphics processing, audio processing, and scientific computing.

There are two main types of SIMD parallelism:

1)Vector parallelism: Vector parallelism operates on vectors of data, which are contiguous sequences of data items of the same type. For example, a vector of 32-bit floating-point numbers could contain 32 floating-point numbers stored in contiguous memory locations.

2)Packed SIMD: Packed SIMD operates on packed data, which is a collection of data items of different types stored in a single register. For example, a 128-bit packed SIMD register could contain four 32-bit floating-point numbers. Packed SIMD is more flexible than vector parallelism because it allows different data types to be operated on in the same instruction. This can be useful for applications that need to operate on multiple data types simultaneously, such as image processing and multimedia applications. How many numbers it can process in parallel is limited by the length in bits of our general purpose registers or vector registers. On some CPUs it performs SIMD operations on your regular general purpose registers. On others it uses special registers for SIMD operations. But in RISC-V as an example because it offers a fairly simple instruction-set. It is going to use the ADD16 and ADD8 instructions in the RISC-V P Extension.

P-Extension in RISC-V:

The P-extension defines a new set of vector registers, which are wider than the general-purpose registers. Each vector register can hold multiple elements of the same data type.

Some example of P-Extension Instructions:

vadd : Add two vectors.

vsub : Subtract two vectors.

Vmul : Multiply two vectors.

vand : Bitwise AND operation on two vectors.

vor : Bitwise OR operation on two vectors.

vxor : Bitwise XOR operation on two vectors.

SIMD (Single Instruction, Multiple Data) is a technique designed to concurrently execute the same operation on multiple pieces of data. In traditional programming, particularly Single Instruction Single Data (SISD) scenarios, operations across large datasets are executed using loops. This approach, however, is often inefficient due to the need for numerous iterations. To address this

inefficiency, one optimization technique is loop unrolling. This involves replicating the single operation within the loop, reducing the overall number of iterations required. In SIMD, vectors play a crucial role. A SIMD vector can be utilized as an argument for a specific instruction, enabling the simultaneous execution of that instruction on all elements within the vector. The performance of SIMD is directly impacted by the number of values that can be loaded into the vector, influencing the speed at which a complete dataset can be processed. The vector size depends on both the data size in use and the specific SIMD implementation.

Values stored in SIMD vectors are processed in a set of special CPU registers dedicated to parallel processing. The size and number of these registers are determined by the chosen SIMD implementation. SIMD leverages multiple CPU functional units, independent units for arithmetic and Boolean operations that execute concurrently.

SIMD Instruction Set:

vaddu – unsigned addition of n-pairs of two registers,

vaddiu – unsigned addition of n-pairs of register and immediate value,

vand – logical and of n-pairs of two registers,

vandi – logical and of n-pairs of register and immediate value,

vor – logical or of n-pairs of two registers,

vori – logical or of n-pairs of register and immediate value,

vxor – logical xor of n-pairs of two registers,

vxori – logical xor of n-pairs of register and immediate value,

vmul – signed multiplication of n-pairs of two registers,

vlw – n load words,

vsw – n store words.

2. SIMD Architecture

There are two main types of SIMD architectures: True SIMD architecture and Pipelined SIMD architecture. SIMD architecture operates with two memory types: distributed memory and shared memory. True SIMD architecture can be implemented using both shared and distributed memory, featuring a single control unit. In distributed memory, each processing element has its own memory and collaborates with other processing elements. The control unit plays a crucial role in facilitating communication among processing elements within the same SIMD architecture.

The process begins with the control unit providing instructions to each processor, and simultaneously, each processor acts as an independent arithmetic unit. The control unit manages communication between processing elements, enabling specific elements to access information from others within the architecture. It oversees the transfer of information between processing elements, especially in architectures where processors can communicate with each other's memory through an interconnected network.

On the other hand, Pipelined SIMD architecture functions differently. The control unit sends instructions to each processing element, directing them to perform tasks using shared memory at various stages. The control unit is responsible for dispatching instructions in distinct streams to processing elements. The pipeline leverages different processing capabilities, operates on streams of instructions, and executes arithmetic operations.

In the pipeline, the instruction unit incorporates various arithmetic logic units linked to memory. Data undergoes evaluation and is stored in different memory units, ensuring that the pipeline is supplied with information as rapidly as possible. This approach allows for efficient parallel processing in SIMD architectures.

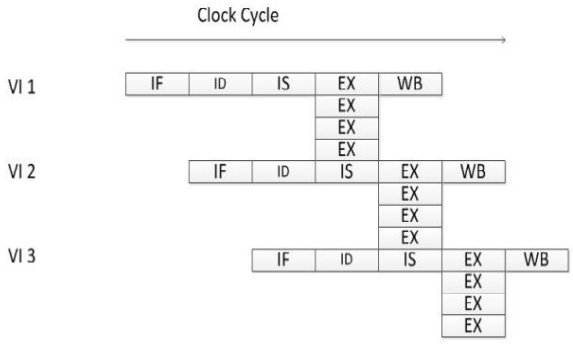

Fig.1 Pipelining in SIMD type Vector Processor

3. Literature Survey

Conduct a comprehensive survey of existing research and publications related to SIMD instruction sets, RISC-V architecture, and relevant compiler technologies. Understand the trade-offs involved in SIMD instruction set design.

4. Literature Review

Ahmed S. Al-Araji et.al [1]. The document discusses the architecture and advantages of Single Instruction Multi-Data (SIMD) in multimedia applications. It covers the basic principles of SIMD architecture, including data parallelism and the implementation of SIMD architecture in multimedia applications such as digital signal processing, image processing, and mobile applications. The advantages of SIMD architecture include improved execution time

performance, scalability of data size, cost savings, and energy efficiency. The paper also highlights the use of SIMD in accelerating processing speed and reducing energy consumption in multimedia applications. Additionally, it discusses the drawbacks and advantages of different SIMD processing architectures and provides an overview of the SIMD help framework for off loading multimedia applications.

Mohammad Suaib et.al [2] proposed the design and architecture of a SIMD type vector processor, which aims to improve system-level model performance by exploiting data level parallelism (DLP) and instruction level parallelism (ILP). The processor operates on short vectors with 4 words, allowing for concurrent processing of elements in one clock cycle. This architecture reduces clock cycles per instruction (CPI) and enhances performance through parallelism. The paper also compares the proposed SIMD-Vector architecture with traditional SIMD extensions and vector architectures, highlighting its advantages in terms of hardware reduction, reduced instruction bandwidth, and improved throughput. Additionally, the document presents the motivations behind the design, the evaluation results, and future work for enhancing parallelism to support longer vectors.

Sudha B.S et.al [3]. proposed the design and physical synthesis of a 16-bit RISC processor using a 4-stage pipelining-based CPU. The processor is based on the Harvard architecture, with separate data and instruction buses to enhance speed and reliability. The RISC architecture is compared to CISC, highlighting its advantages in power efficiency and simplified design structure. The architecture of the processor is detailed, including the program counter, control unit, and arithmetic logic unit.The workflow of the project involves both frontend and backend design, encompassing HDL description, simulation, synthesis, and physical design. The simulation results demonstrate the functionality of the processor's ALU, while the physical design results showcase the gate-level netlist, placement, clock tree synthesis, and routing stages.The conclusion emphasizes the successful implementation of the Harvard structure RISC processor, with a focus on reducing delay and power consumption through the 4-stage pipelined architecture. The document includes references to related works in the field.In summary, the project presents a comprehensive approach to designing and synthesizing a 16-bit RISC processor, highlighting its architecture, simulation results, physical design process, and the achieved improvements in speed and power efficiency. The study contributes to the field of VLSI design and processor architecture.

Devaraconda Dinesh and R. Manoj Kumar [4], proposed the physical design implementation of a 16-bit Reduced Instruction Set Computing (RISC) processor using Verilog Hardware Description Language. The processor follows a 4-stage pipelined architecture and is designed to perform various operations with 16 instructions, each consisting of 16 bits. The control unit has four states representing the pipelining process: IDEL, FETCH, DECODE, and EXECUTE. The physical design process involves partitioning, floorplanning, placement, clock tree synthesis, and routing.The paper emphasizes the importance of clock tree synthesis, discussing factors affecting the clock path such as duty cycle, latency, clock tree power, signal integrity, and crosstalk. It also delves into the significance of routing, covering global routing and detailed routing. Additionally, the paper highlights the Engineering Change Order (ECO) process to address timing and design specification mismatches during the tape-out process.The document provides a detailed explanation of various aspects of physical design implementation, including clock tree synthesis, routing, and Engineering Change Order. It also discusses the impact of factors like duty cycle, latency, clock tree power, signal integrity, and crosstalk on the design process. The paper presents valuable insights into the physical design of the RISC processor, emphasizing the significance of clock tree synthesis and routing in achieving the required design specifications.

Otto Simol [5], proposed the master's thesis on the physical implementation of a RISC-V processor at Aalto University. It discusses the development of the "A-core" processor, focusing on the design and physical implementation. The thesis covers various aspects of the physical implementation process, including RISC-V platforms, Chisel, physical implementations of processors, and verification and validation. It emphasizes the importance of verification and validation in ensuring the correctness and manufacturability of the design. The thesis also discusses the steps involved in the physical implementation process, such as synthesis, floorplanning, power planning, clock tree synthesis, routing, and verification.

Basil Sh. Mahmood et.al[6]. The paper discusses the design and implementation of a SIMD Vector Processor on FPGA, consisting of 4 parallel lanes with independent processing elements, arithmetic units, vector register files, and local memories. A scalar processor is also included to handle instructions that cannot be processed by vector lanes. The architecture, memory system, fetch and decode units, vector register file, vector lane interior architecture, and instructions format are detailed. The processor was implemented on an FPGA chip and examples of vector and scalar instructions results are provided. The paper concludes by discussing the benefits of the designed processor.The paper discusses the design and implementation of a SIMD Vector Processor on FPGA, consisting of 4 parallel lanes with independent processing elements, arithmetic units, vector register files, and local memories. A scalar processor is also included to handle instructions that cannot be processed by vector lanes. The architecture, memory system, fetch and decode units, vector register file, vector lane interior architecture, and instructions format are detailed. The

processor was implemented on an FPGA chip and examples of vector and scalar instructions results are provided. The paper concludes by discussing the benefits of the designed processor.

D.A.M. Koene [7], proposed the "Implementation and Evaluation of Packed-SIMD Instructions for a RISC-V Processor" by D.A.M. Koene. The thesis explores the addition of packed-SIMD instructions to a RISC-V processor and evaluates its impact on performance, energy usage, and area usage. The P-extension, a draft instruction set extension for RISC-V, introduces SIMD instructions for 8-bit, 16-bit, and 32-bit integer data types. The thesis divides the 332 instructions of the P-extension into subsets based on application requirements and hardware implementation. The impact of the newly added SIMD instructions is tested in scenarios of matrix multiplication and a real-world machine learning application. The results show significant speedups in performance, but additional hardware usage comes at a cost. The P-extension is compared to the V-extension, which supports vector-SIMD instructions and floating-point operations. The P-extension reuses general-purpose registers, while the V-extension uses dedicated vector registers. The P-extension focuses on packed-SIMD instructions, while the V-extension supports vector-SIMD instructions.

Benjamin D et.al [8], proposed the implementation of a SIMD pipelined processor on an FPGA. Mar for his Master's thesis in Computer Engineering at the University of New Mexico. The processor is designed using VHDL and is intended for educational purposes and as a foundation for digital signal processing applications. It is based on a subset of MIPS instructions and features a five-stage pipeline with hazard control. The SIMD architecture extends the basic architecture by adding multiple basic units and control logic for different datapaths. The processor was verified in both simulation and synthesis, operating at a maximum speed of 100 MHz.

Andrew Waterman et.al [9], proposed a detailed technical document describing the RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 2.2. It includes information about the editors, contributors, licensing, preface, and various standard extensions of the RISC-V ISA. The document also covers the RISC-V ISA overview, instruction length encoding, exceptions, traps, and interrupts, as well as the programmers' model for the base integer subset and base instruction formats. The document provides a comprehensive overview of the RISC-V ISA, including its design principles, user-visible state, instruction formats, and various standard extensions. It also discusses the rationale behind the design decisions and the considerations for different application areas.

LIU Peng et.al [10], proposed a rapid prototyping approach for designing multiprocessor system-on-a-chips (MPSoCs). The approach seeks to alleviate the difficulties back-end designers encounter as a result of growing complexity and dwindling process technology. The floorplan, clock network, and power plan design processes are the main topics of this article. A unique diagonal floorplan is suggested in the floorplan design to improve data sharing between the MPSoC's many cores. This technique lowers route delay and permits inter-core data transfer. To ensure balanced synchronisation, a unique clock network with manually adjusted buffers and changeable routing rules is built for the clock network architecture. Clock buffers and inverters receive special consideration in order to enhance timing performance.The goal of the power network architecture is to preserve power integrity while reducing electromigration and IR drop.Power pads are positioned for uniform distribution, and the width of power cables is carefully considered. To reduce the impacts of noise coupling, clock signals are routed according to specific principles. The study indicates that the diagonal floorplan, interactive clock tree synthesis, power network design, and the suggested quick prototyping technique enable the management of deep sub-micron design issues and reduce the design cycle.

Philippos Papaphilippou et.al [11], proposed a demonstration of Simodense, an open-source softcore designed specifically for analysing custom SIMD (custom single-instruction, multiple-data) instructions. By using a Verilog template to create a custom SIMD instruction for sorting four integers 32 bits wide, the sample illustrates the programmability feature of Simodense. Custom SIMD instructions in CPUs can be made programmable via RTL, enabling highly-integrated accelerators that take advantage of the CPU logic already in place, including caches and high memory throughput to main memory. Because of the adaptability of the RISC-V instruction set and its support for custom instructions, the proposed RISC-V-based softcore is appropriate for investigating reconfigurable SIMD instructions. There isn't much research on reconfigurable SIMD instructions, but the suggested method may help prevent instruction sets from growing unnecessarily large with overly specialised instructions.

Hongjiang Yu et.al [12], suggested an improved implementation of activation instructions based on the RISC-V architecture, drawing from the paper "An Optimized. Implementation of Activation Instruction Based on RISC-V" by Yu, H. et al. Neural networks depends on activation, however array activation cannot be handled well by the RISC-V standard instructions. The authors created a custom instruction for the Hummingbird E203 RISC-V processor to provide activation functions in order to solve this problem. The complete activation process, including data loading, data arithmetic, and data write-back, is completed by this one instruction. The authors also suggest an additional reading and writing technique that satisfies the activation operation requirements for lengthy arrays by utilising a tiny hardware storage unit. Furthermore, the authors introduced a new parameter, the array's length, to tell their developed hardware to accommodate arrays of any length. To increase the effectiveness of execution, they also enhanced the scheduling of instructions throughout the activation process. The suggested design showed a notable speed boost over typical RISC-V instructions while using a small fraction of the energy, according to the authors' studies.

Rongxue Sun et.al [13], proposed a An optimised multiplier circuit construction based on the RISC-V instruction set architecture is presented in the paper "Design and Implementation of RISC-V Based Pipelined Multiplier". A processor's total processing power is greatly influenced by the multiplier's performance. The authors suggest a pipelined multiplier to increase the effectiveness of multiplication instructions. The suggested multiplier design creates partial products using the Radix-4 Booth technique, which is followed by a Wallace tree structure compression. To optimise timing, the resulting partial product is calculated using a parallel prefix adder. To further improve the performance of the multiplication calculation, a pipeline structure is implemented. According to the simulation findings, the suggested pipelined multiplier outperforms the conventional CLA multiplier and the SK multiplier without a pipeline in terms of runtime. In gate-level simulation, the pipelined multiplier's ultimate frequency exceeds 1 GHz, demonstrating its high throughput rate. In comparison to conventional methods, this study presents an optimised design for a pipelined multiplier based on RISC-V, showcasing increased performance and economy.

Paul H. J. Kelly et.al [14], proposed a new method for investigating sophisticated reconfigurable Single Instruction, Multiple Data (SIMD) instructions by extending the RISC-V Instruction Set Architecture (ISA). A non-standard set of vector instruction types optimised for unique SIMD instructions in a softcore is introduced by the authors. Additionally, they suggest an open-source RISC-V softcore that has been tuned for streaming efficiency and unique SIMD instructions. Enhancing memory-intensive Field Programmable Gate Array (FPGA) applications is the main goal. The idea is to make it possible to test out sophisticated SIMD instructions that may be incorporated into CPUs in the future that have changeable sections for unique instructions. The custom SIMD instruction types are described in the document along with Verilog templates that can be used to implement these instructions in hardware. For complex activities, fewer instructions are needed thanks to the suggested instruction types, which provide access to a large number of operands. The document offers information on the difficulties and potential benefits of upcoming micro-architectures, and the cache hierarchy of the softcore is optimised for bandwidth. The evaluation shows that the suggested custom SIMD instructions for prefix sum and sorting are effective and achieve notable speedups over serial implementations.All things considered, the paper offers a thorough framework for investigating unique SIMD instructions on FPGAs and offers perspectives on the possibility of incorporating these instructions into CPUs down the road. The suggested method makes it possible to effectively experiment with sophisticated SIMD instructions and has the potential to greatly improve the performance of memory-intensive applications. The research and development of custom SIMD instructions and their integration into future CPU architectures will have a strong basis thanks to the document's conclusions, which are beneficial for practitioners and academics in the fields of computer architecture, FPGA-based computing, and embedded systems.

K. Vishnuvardhan Rao et.al [15], proposed The creation of a 16-bit Reduced Instruction Set Computing (RISC) processor utilizing a bespoke design technique is covered in this work. It describes the processor's architecture, including the Control Unit, Registers, Arithmetic and Logical Unit (ALU), and Instruction

Execution Unit. The design of useful blocks like the Wallace Tree Multiplier, Carry Look Ahead Adder, and Barrel Shifter is also covered in this article. The processor's performance analysis is shown, together with the amount of power used and the time it takes to execute particular instructions. The possibility for low power operation and the use of suitable logic styles for the processor.

XIAO Zhi-bin et.al [16], The pipeline optimization for a 32-bit embedded RISC3200 processor with multimedia extension instructions is covered in this article. The CPU runs at 200 MHz with an average CPI of 1.16 and is intended for low-cost consumer multimedia products. The pipeline consists of a simple RISC core and an extension section that has memory management, video/audio interface logic, and a multimedia extension unit.The article suggests several optimization strategies, such as implementing a scalable super-pipeline methodology, utilizing a data flow graph (DFG) with delay information to analyze the pipeline stage's important path, implementing a distributed data bypass unit, and implementing a centralized pipeline control scheme. The document describes the adjustments and improvements made to the pipeline organization, as well as how the addition of multimedia instructions complicated the architecture of the pipeline.

Joon-Seo Yim, Seong-Ok Bae et.al [17], proposed a floor plan-based planning methodology for power and clock distribution in ASICs. It highlights how crucial early planning is in resolving issues with clock skew, IR-drop, and power usage. Using floorplan data and predicted power usage, the methodology optimizes clock and power networks. Power distribution determines the size, current sources, and trunk width of the power network through an iterative process of simulation and resizing. The power network is modeled at the block and chip levels using a hierarchical method, and block size and transition activity are used to determine the current source.A hierarchical structure with multiple levels of clock drivers is used for clock distribution. The connecting settings and wire are used to optimize the top-level clock tree.

Andrew Waterma et.al [18], proposed The document "The RISC-V Instruction Set Manual Volume II: Privileged Architecture" provides an overview of the privileged architecture of the RISC-V processor. It includes the allocation of control and status registers (CSRs) for different privilege levels, such as unprivileged, supervisor, hypervisor, and machine levels. The document also outlines the conventions for CSR address mapping and the accessibility of CSRs based on privilege levels. Additionally, it lists the currently allocated CSRs for unprivileged floating-point, counter/timers, supervisor trap setup, configuration, handling, protection and translation, as well as debug/trace registers, hypervisor counter/timer virtualization, and virtual supervisor registers.

David M. et.al [19], proposed Dahle The Kestrel high-performance programmable parallel co-processor is designed and its applications are covered in "Design of an 8 bit SIMD parallel processor." Each of the eight-bit processing components in the Kestrel processor's linear array has a multiplier, local memory, and an arithmetic logic unit (ALU). Applications like picture compression, neural networks, sequence analysis, and floating-point arithmetic are also covered in the study. Kestrel design is a closely integrated process that looks at VLSI implementation, system and PE architecture, and algorithms. The report also presents a performance comparison of Kestrel with other SIMD chips. All things considered, the Kestrel CPU is built for speed and versatility in parallel processing.

Preetam Bhosle bin et.al [20], proposed The design and implementation of a low power, 32-bit, high performance RISC core are presented in this study. For low power consumption, it makes use of the clock gating approach and Harvard architecture. The architecture is divided into four phases: memory read/write back, fetch, decode, and execute. Verilog is used for design, Modelsim is used for simulation, and Altera Quartus II and Altera FPGA board are used for implementation. The suggested architecture offers halt support and can stop pipelining from flushing in the event of a branch command. With a speed of about 110.92 MHz, the Altera FPGA board is used to simulate and validate the design.

Sharda P. Katke et.al [21], proposed The design and development of a 32-bit RISC processor with a five-stage pipelined architecture using VHDL are covered in the text. The CPU can be designed with a variety of applications in mind, with the goal of increasing speed and performance. The utilization of VHDL programming and the role of VLSI in processor design are highlighted in this study. The system architecture with pipeline, comprising the memory access unit, write back unit, execution unit, fetch unit, and decoding unit, is described in depth. Future efforts to improve the processor's performance are discussed in the paper's conclusion. The references contain a range of publications about VHDL-based development and processor design.

5. CONCLUSIONS

The Implementation and Evaluation of the RISC-V P extension represent a packed Single Instruction, Multiple Data (SIMD) instruction set tailored for RISC-V processors. This extension introduces support for 8-bit, 16-bit, and 32-bit integer data types, thereby augmenting the RISC-V instruction set architecture. This enhancement empowers developers to craft more efficient and potent data-parallel programs. The significance of the P extension becomes evident in its potential to expedite a diverse array of applications, spanning graphics and audio processing to scientific computing.

The evaluation of the P extension's impact extends to considerations such as performance, energy usage, and area utilization. Comparative analyses with vector-SIMD instructions offer valuable insights into the intricate trade-offs inherent in the design of SIMD instruction sets. In summary, this research makes a substantial contribution to the ongoing evolution of RISC-V processors, expanding their capabilities and fostering the development of more resource-efficient data-parallel applications.

6. REFERENCES

[1] Ahmed S. Al-Araji et.al Architecture and Advantages of SIMD in Multimedia ApplicationsJOURNAL OF XI AN UNIVERSITY OF ARCHITECTURE & TECHNOLOGY 12(6):1452-1459DOI:10.37896/JXAT12.06/2045

[2] Suaib, M., Palaty, A., & Sambhav Pandey, K. (2011). Architecture of SIMD type vector processor. *International Journal of Computer Applications*, *20*(4), 42–45. https://doi.org/10.5120/2418-3233

[3] Design and physical synthesis of a 16 bit RISC processor. (2023). *International Research Journal of Modernization in Engineering Technology and Science*. https://doi.org/10.56726/irjmets42723

[4] Dinesh, D., & Manoj Kumar, R. (2016). Physical design implementation of 16 bit RISC processor. *Indian Journal of Science and Technology*, *9*(36). https://doi.org/10.17485/ijst/2016/v9i36/102911

[5] Otto Simol. Physical implementation of a RISC-V processor – Aalto. https://aaltodoc.aalto.fi/bitstream/handle/123456789/120255/master_Simola_Otto_2023.pdf?sequence=1

[6] Mahmood, B. Sh., & Jbaar, M. A. (2011). Design and implementation of SIMD vector processor on FPGA. *International Symposium on Innovations in Information and Communications Technology*. https://doi.org/10.1109/isiict.2011.6149607

[7] 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

[8] Benjamin Mar SIMD pipelined processor implemented on a FPGA Benjamin Mar Publication Date 9-9-2007 Abstract The goal of this thesis was to create a processor using VHDL that could be used for educational purposes as well as a stepping stone in creating a reconfigurable system for digital signal processing or image processing applications. https://digitalrepository.unm.edu/ece_etds/170

[9] A. Waterman1 and K. Asanovi, “The risc-v instruction set manual volume i: Unprivileged isa,” EECS Department, University of California, Berkeley, Tech. Rep. 0191213, December 2019. [Online]. Available: https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMAFDQC/riscv-spec-20191213.pdf

[10] Physical design method of MPSoC Liu Peng, Xia Bing-jie & Teng Zhao-wei Journal of Zhejiang University-SCIENCE A 8 , 631–637 (2007) Cite this article 42 Accesses 1 Citations 9 Altmetric Metrics Abstract Floorplan, clock network and power plan are crucial steps in deep sub-micron system-on-chip design. https://link.springer.com/article/10.1631/jzus.2007.A0631

[11] P. 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.

[12] In this paper, we propose an optimized implementation of activation instruction based on RISC-V. Based on the opensource RISC-V processor Hummingbird E203, we designed a special instruction for the implementation of activation functions.

https://www.mdpi.com/2079-9292/12/9/1986

[13] Design and Implementation of RISC-V Based Pipelined Multiplier registers are added as a pipeline to achieve a high-efficiency multiplication calculation. With the operating voltage and temperature set to typical conditions, the integrated multiplier area is 50260.6 μm 2, and the power consumption is 20.41 mW. The final frequency of the multiplier is 1 GHz in gate-level simulation.

https://iopscience.iop.org/article/10.1088/1742-6596/2625/1/012006

[14] Extending the RISC-V ISA for exploring advanced reconfigurable SIMD instructions | Semantic Scholar Corpus ID: 235422540 Extending the RISC-V ISA for exploring advanced reconfigurable SIMD instructions Philippos Papaphilippou, P. Kelly, W. Luk Published in arXiv.org 14 June 2021 Computer Science, Engineering TLDR

https://www.semanticscholar.org/paper/Extending-the-RISC-V-ISA-for-exploring-advanced...

[15] This paper presents the design of a 16 bit Reduced Instruction Set Computing (RISC) processor using the custom design approach. Statistical Analysis: The type of processor...

https://www.researchgate.net/publication/283580452_Design_of_a_16_bit_RISC_processor

[16] In this paper, an optimized 16-bit RISC processor is proposed with the concept of pipelining and clock gating, which utilizes minimum on-chip power with maximum throughput achieved.

https://link.springer.com/chapter/10.1007/978-981-99-1410-4_9

[17] A floorplan-based planning methodology for power and clock distribution in ASICs [CMOS technology] Abstract: In deep submicron technology, IR-drop and clock skew issues become more crucial to the functionality of chip. This paper presents a floorplan-based power and clock distribution methodology for ASIC design.

https://ieeexplore.ieee.org/document/782120

[18] The RISC-V Instruction Set Manual Volume I: Unprivileged ISA Document Version 20191213 Editors: Andrew Waterman 1, Krste Asanovi´c,2 1SiFive Inc., 2CS Division, EECS Department, University of California, Berkeley andrew@sifive.com, krste@berkeley.edu December 13, 2019.

https://riscv.org/wp-content/uploads/2019/12/riscv-spec-20191213.pdf

[19] Kestrel: design of an 8-bit SIMD parallel processor Abstract:Kestrel is a high-performance programmable parallel co-processor. Its design is the result of examination and reexamination of algorithmic, architectural, packaging, and silicon design issues, and the interrelations between them.

https://ieeexplore.ieee.org/document/634852

[20] This paper presents the design and implementation of a low power pipelined 32-bit High performance RISC Core. The various blocks include the Fetch, Decode, Execute and Memory Read / Write Back to implement 4 stage pipelining. In this paper we are proposing low power design technique in front end process.

https://www.ijitee.org/wp-content/uploads/papers/v1i3/C0208071312.pdf

[21] she proposed work is the design of a 32 bit RISC (Reduced Instruction
Set Computer) processor, which has 5 stages of pipeline viz. instruction fetch,
instruction decode, instruction execute, memory access and write back all in
one clock cycle. Expand No Paper Link Available Save to Library Create
Alert Cite 7 Citations Citation Type More Filters
https://www.semanticscholar.org/paper/Design-and-Implementation-of-5-Stages-Pipelined...