IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

THE DESIGN AND VERIFICATION OF A 32-BIT RISC V PROCESSOR USING VEDIC MATHEMATICS

Girish H[1], Vajjala Sai Sree Hari [2], Ponde Lakshmi Narasimha[3], P N Shreya[4], and Dadigala Bhargavi[5]

[1] Professor, Department of Electronics and Communication Engineering, Cambridge Institute of Technology, Bengaluru, India

[2,3,4,5] Student, Department of Electronics and Communication Engineering, Cambridge Institute of Technology Bengaluru, India

ABSTRACT*:*

Vedic multiplier architecture is used in the construction of a 32-bit RISC V processor to improve speed and decrease computational complexity. Its Vedic Sutra-based ALU and MAC units, which are implemented in Verilog HDL and simulated using the Xilinx design suite, lower power consumption and latency as compared to traditional architectures. Comprising of conventional components such as Control Unit, Register Bank, Programme Counter and Memory. The processor can carry out up to 16 instructions. It is an effective option for a variety of computer jobs since it provides increased speed, decreased power consumption, and reduced area utilisation.

Keywords **- Reduced Instruction Set Computer, Von Neumann architecture, Verilog HDL, Vedic Mathematics, Urdhva-Tiryagbhyam Sutra.**

INTRODUCTION:

Simple instruction sets are given priority in the RISC V architecture, which promotes effective decoding. This design approach frequently yields better performance than CISC because it combines simple instructions with a microprocessor architecture that can execute instructions in a few cycles. RISC V offers fixed instruction sizes, additional registers, and better clock utilization, even if it can require more memory and code. Performance increases are usually two to four times higher than with CISC computers. Using Hardware Description Language to design a 32-bit RISC V processor, as described in includes internal parts such as registers, ALUs, memory units, and control units. It has been shown that such processors can be implemented using FPGA. Vedic mathematics uses techniques known as sutras to solve mathematical problems more quickly. It is an effective method inside mainstream mathematics, not a distinct field of study. In processors, multipliers are essential, particularly in fields like digital signal processing and communication (like FFT). Low power consumption is intended for logarithmic multipliers. There are drawbacks to some common binary multiplication techniques, such as array and Booth multiplication. Based on the Urdhva Triyagbhyam Sutra, Vedic multipliers provide effective substitutes. Research has shown the superiority of Vedic multipliers over traditional techniques by comparing several varieties in terms of area, power, and delays. Studies have suggested using Vedic

multipliers to create high-speed variants that perform better than conventional techniques and to enable quick calculation for DSP operations. Reversible circuits, as opposed to combinational logic circuits, can even further minimize power usage. RTL circuit verification is done using property-driven verification. In digital signal processing (DSP) applications, where multiplication often entails repetitive addition operations, the squaring task is facilitated by the Ekadhikena Purvena Sutra. Multipliers typically undergo a three-step process: partial product generation (PPG), partial product reduction (PPR), and final addition. For instance, AND gates can generate a partial product matrix (PPM) during the PPG stage for unsigned multiplication.

LITERATURE REVIEW:

S. Lad and V. S. Bendre, "Design and Comparison of Multiplier using Vedic Sutras," 2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA), Pune, India, 2019, pp. 1-5

In today's computational landscape, rapid processing units are imperative for numerous real-time applications. These units typically comprise Arithmetic Logic Units (ALUs) and Multiply-Accumulate (MAC) units as foundational components, essential for efficient and swift execution. Multipliers serve as a fundamental element in digital signal processors, playing a pivotal role in maintaining accuracy and boosting execution speed. To optimize the performance of ALUs and MAC units, modifications to multipliers, adders, and registers are essential.

Seung Pyo Jung, Jingzhe Xu, Donghoon Lee, Ju Sung Park, Kang-joo Kim, and Koon-shik Cho, "Design & verification of 16-bit RISC processor," 2008 International SoC Design Conference, Busan, 2008, pp. III-13-III-14

This paper introduces the design and verification process for a 16-bit RISC processor. The proposed processor features Harvard architecture, a 24-bit address, 5-stage pipeline instruction execution, and internal debug logic. Successful implementation of the ADPCM vocoder and SOLA algorithm on the FPGA-based processor is demonstrated. In the realm of portable multimedia players (PMPs) and personal digital assistants (PDAs), the demand for low-power and compact processors at the SOC level has surged. While the 8051 and ARM 7 processors have been widely used at the SOC level, they face limitations in terms of size and calculation time. To address this, the paper proposes a new 16-bit RISC processor designed specifically for SOC-level ASIC integration. The core execution has been tested, but on-chip debugging requires a GDB server program for application debugging. Figure 5 illustrates the connection between the GDB server program, the SW debugger, and the new core. Additionally, a compiler is essential for fast programming. The future objective of the proposed processor is to develop a compiler and GDB server program catering to high-level users.

Balpande Vishwas V, Abhishek B. Pande, Meeta J. Walke, Bhavna D. Choudhari and Kiran R. Bagade. “Design and Implementation of 16 Bit Processor on FPGA.” (2015).

This project involves the design of a 16-bit RISC processor and the modeling of its components using Verilog HDL, all based on the Harvard architecture. The chosen instruction set is deliberately simple to ensure proper execution by the hardware. In addition to foundational building blocks like adders and registers, more intricate components such as ALU and memories were also designed and simulated. The ALU modeling in this project follows a fully structural approach, beginning with half adders and culminating in the development of a semi-custom layout. Memories, being complex blocks, were modeled using a behavioral approach, whereas simpler blocks like adders were approached structurally. The designed 16-bit RISC processor was simulated to verify its functionality, offering flexibility for potential extension to a 32 or 64-bit processor through straightforward code adjustments. The datapath can likewise be modified to incorporate new blocks, a capability not typically found in traditional processor units array, Booth, and Vedic operators.

F. Adamec and T. Fryza, "Design — Time configurable processor basic structure," 13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, Vienna, 2010, pp. 119-120

This project introduces the concept of designing a time-configurable processor, outlining the basic architecture of the processor core, particularly focusing on its minimal core configuration (referred to as the integer core). The conclusion provides a comparison with other similar designs, along with a summary of the current progress and proposed future work. A comparable solution in the field is the Xtensa processor from Tensilica, which offers two versions catering to control purposes and high computation demands. The instruction set for Xtensa processors can be extended if required, with instructions being a mix of 16/24-bit long and capable of execution in VLIW fashion with 3 to 16 instructions in one VLIW instruction packet. The architecture follows a RISC LOAD/STORE approach and can be expanded to include features like FPU, ConX D2 DSP for 16-bit optimized DSP engine, ConnX Vectra LX DSP Engine for SIM operations, ConnX Vectra VMB for baseband communications acceleration, and Advanced TIE for multi-cycle SIMD instruction execution in the pipeline. Additionally, components like Hifi 2 Audio can be added for multimedia codec acceleration.

METHODOLOGY:

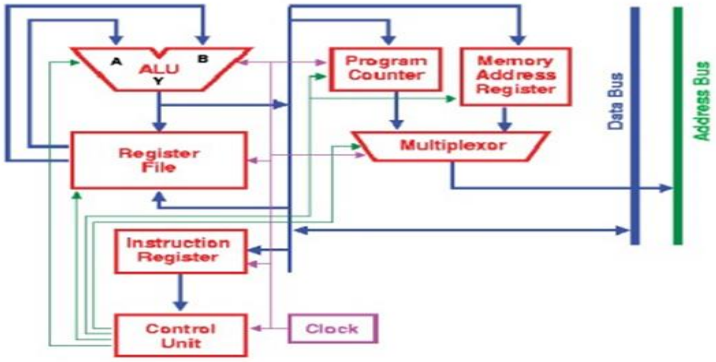

Fig 1: Block diagram of processor

Multiplexer : As was previously said, the most fundamental block that must be included in a processor is the multiplier unit. Techniques for designing at high speeds were created in [13][14]. Urdhva Tiryagbhyam was created since mental calculations should be made by multiplying each row. As a result, adding and multiplying vertically and crosswise were introduced, fixing one easy-to-remember digit from the unit of measurement for each operation.

Arithmetic Logic Unit: In a combinational circuit, the ALU stands for Arithmetic and Logical Unit. This unit's goal is to do various bitwise and arithmetic computations using a range of instruction sets. An instruction and a few operands make up an ALU input in a CPU. The operands are then used in the operation after the opcode tells the ALU which and what operation to perform.

Control Unit: To regulate the use of resources like registers, external memory, and tasks they must complete, the Control Unit creates control signals. In our system, to read or write data into blocks, the control unit sends control signals to the memory and register blocks. Additionally, it regulates the branching signal based on the execution of instructions.

Register Block: The register block has 32 registers that, in response to control signals, can write and store 32-bit values from the memory block or the ALU. A unique address, or the register's location inside the memory block, can be used to access any register in the register.

Memory Block: The 32-bit addressable words are stored in the memory block, which acts as an external memory. The ALU can read from and write to the memory block in response to control signals.

Instruction block, Program counter: The set of instructions to be carried out is stored in instruction blocks. The address of the instruction that will be executed is stored in the program counter. The program counter in instruction block G will be increased by one for each clock cycle, pointing to the subsequent

address. Decoder of instructions Different kinds of instructions are stored in processors' instruction libraries. Additionally, it is the responsibility of the instruction decoder to recognize the kind of instruction and assign the relevant registers to each component of the instruction. Depending on the opcode, this block decodes the instruction into various locations and function values.

RESULT:

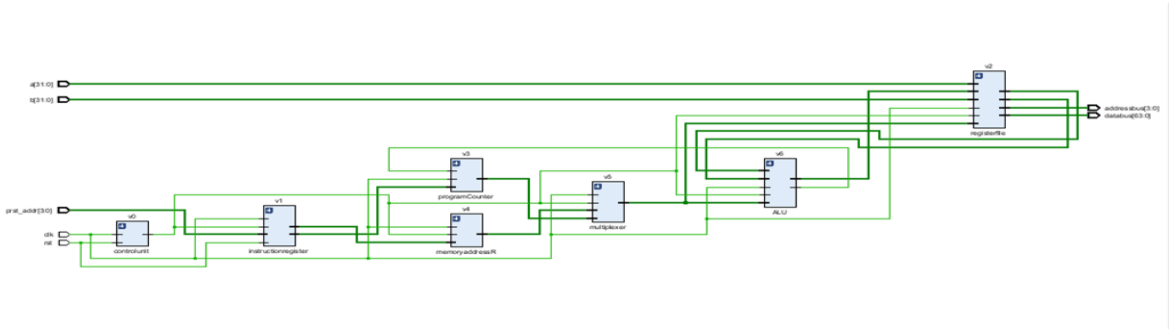

Fig 2: Creating an RTL schematic for a 32-bit RISC-V processor utilizing a Vedic multiplier would be quite complex and detailed. While I can't provide the entire schematic here, I can outline the general components and steps involved.

RISC-V Processor Core: Start with the basic components of a RISC-V processor core, including the instruction fetch, decode, execute, memory access, and write back stages.

Vedic Multiplier Integration: Integrate the Vedic multiplier into the execution stage of the processor pipeline. This involves adding components for multiplying two operands efficiently using Vedic multiplication techniques.

Pipeline Registers: Include pipeline registers between stages to facilitate the smooth flow of instructions and data through the processor pipeline.

Control Logic: Develop control logic for managing the flow of instructions and data, including handling hazards, stall conditions, and branching.

Memory Interface: Implement interfaces for accessing instruction and data memory, including instruction cache, data cache, and main memory.

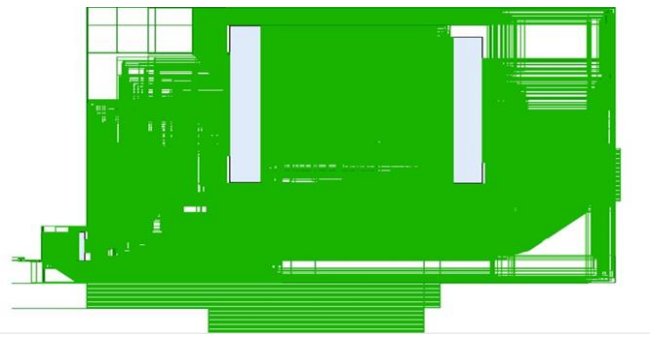

Fig 3: RTL Schematic

This schematic integrates the RISC-V processor's core components with the Vedic multi plier allowing for faster and more precise arithmetic calculations.The 32-bit RISC-V processor featuring a

Vedic multiplier utilizes a simplified instruction set architecture (RISC) to perform computations efficiently. The Vedic multiplier, based on ancient Indian mathematics, enhances multiplication operations by breaking them down into smaller, more manageable steps. This schematic integrates the RISC-V processor's core components with the Vedic multiplier.

OUTPUT:

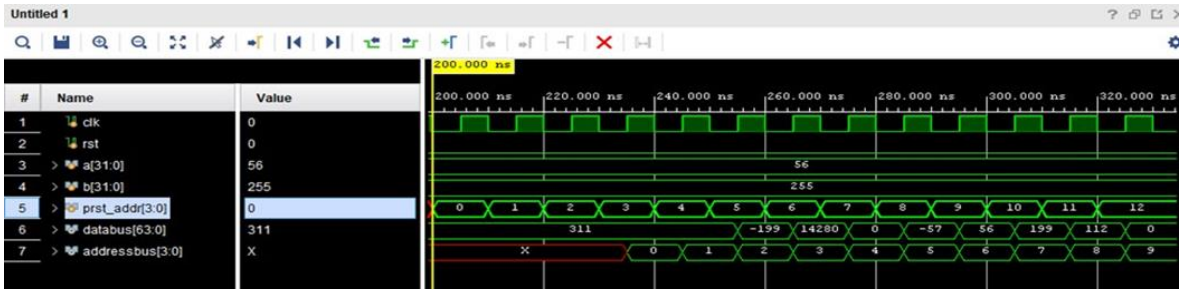

Fig 4: Simulation Results

The waveform output for a 32-bit RISC-V processor utilizing the Vedic multiplier would depict various stages of the processor's operation over time. This includes signals representing instruction fetch, decode, execution, memory access, and write-back stages. Additionally, specific signals related to the Vedic multiplier, such as partial products generation, addition, and accumulation, would be visible. The waveform would showcase the efficient parallel processing capabilities of the Vedic multiplier, resulting in faster multiplication operations compared to traditional methods.

	Area (LUT's)	Delay (ns)
processor	3795	64.549

Fig 5: Evaluation of Area and delay report

CONCLUSION:

This paper describes the implementation of a RISC V processor in ALU based on Vedic sutras. Vedic multipliers are used to increase multiplication's speed while lowering the multipliers' size and power requirements. In this work, a Vedic processor is constructed with standard processor blocks in addition to Vedic MAC and Vedic ALU. The architecture of the 14-instruction instruction set. The current ALU and MAC results are compared with the simulated results of the Vedic ALU and MAC design. In comparison to traditional processors, the 32-bit Vedic processor minimizes latency and conserves energy. Therefore, the most important characteristics of the RISC CPU design are its increased operating speed, decreased power consumption, and reduced area usage.

REFERENCES:

[1] S. Lad and V. S. Bendre, "Design and Comparison of Multiplier using Vedic Sutras," 2019 5th International Conference

on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 2019, pp. 1-5.

[2]Balpande Vishwas V, Abhishek B. Pande, Meeta J. Walke, Bhavna D. Choudhari and Kiran R. Bagade. “Design and

Implementation of 16 Bit Processor on FPGA.” (2015)

[3]Seung Pyo Jung, Jingzhi Xu, Dong-Hoon Lee, Ju Sung Park, Kang-Joo Kim, and Koon-Shik Cho,

"Design & verification

of 16-bit RISC processor," 2008 International SoC Design Conference, Busan, 2008, pp. III-13-III-14, Doi:

10.1109/SOCDC.2008.4815726.

[4]F. Adamec and T. Feyza, "Design — Time configurable processor basic structure," 13th IEEE Symposium on Design and

Diagnostics of Electronic Circuits and Systems, Vienna, 2010, pp. 119-120, Doi: 10.1109/DDECS.2010.5491804.

[5]A. Bisoyi, M. Baral and M. K. Senapati, "Comparison of a 32-bit Vedic multiplier with a conventional binary multiplier,"

2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies,

Padmanabhapuram, 2014, pp. 1757- 1760, Doi: 10.1109/ICACCCT.2014.7019410.

[6]Mr. Nishant G. Deshpande, Prof. Rashmi Mahajan, “Ancient Indian Vedic Mathematics based Multiplier Design for High

Speed and Low Power Processor”, IJAREEIE, Pune, 2014

[7]Priyanka Jain, Dr. G. S. Virdi, Multiplier-Accumulator (MAC) Unit: International Journal of Digital Application &

Contemporary Research, Volume 5, Issue 3, October 2016.

[8]P. S. Mane, I. Gupta, and M. K. Vasantha, "Implementation of RISC Processor on FPGA," 2006 IEEE International

Conference on Industrial Technology, Mumbai, 2006, pp. 2096-2100, doi: 10.1109/ICIT.2006.372448.

[9] Ram, G. &

Lakshmana, Y. & Rani, D. & Kandula, Bala. (2016). Area efficient modified vedic multiplier. 1-5.