Dr.D.Satyanarayana, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

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

Low power 16-bit RISC Processor using vedic mathematics

[1] Dr.D.Satyanarayana,[2] Dr.A.Ranganayakulu,[3] Maddasani Venkata Bala Krishnudu, [4] Surarapu.V.V.Chandramohan Reddy,[5] Challa Bhanu Prakash,[6] Kola Bharath

[1]Associate Professor, Department of ECE, Krishna Chaitanya Institute of Technology & Sciences, Markapur.
[2] Professor & HOD, Department of ECE, Krishna Chaitanya Institute of Technology & Sciences, Markapur.
[3,4,5,6] Student, Department of ECE, Krishna Chaitanya Institute of Technology & Sciences, Markapur.

Abstract- A 16-bit RISC processor with increased instruction execution capability via the use of Verilog and a Vedic multiplier design is the main focus of this study. For the purpose of design simulation, Vivado 2018.We utilize the 3D design suite. Applying the multiplier unit under the Arithmetic and logic unit (MAC) using the methods outlined in the Vedic Sutras is the primary objective of this study. Streamlining conventional computations in order to drastically reduce computing complexity is fundamental to Vedic mathematics. The data path allows information to move between different parts of the RISC processor, such as the memory, program counter, and register bank, and the control unit (which oversees the computer's operations) is another block that is part of the designed processor. The proposed RISC processor is quite simple and has the capability to carry out a grand total of fourteen commands. Compared to conventional ALU and MAC designs, this study successfully reduces power consumption in the Multiply-Accumulate (MAC) and arithmetic logic units (ALU) correspondingly. Both the Arithmetic Logic Unit (ALU) and the Multiply-Accumulate (MAC) have reduced delays as compared to their conventional counterparts. A 16-bit Vedic processor was born because of the gradual merging of the Vedic MAC and ALU with additional processing blocks. Compared to a regular CPU, this results in less delay and less power consumption. It follows that the most important characteristics of a well-designed CPU are a reduction in power consumption, an increase in operating speed, and smaller footprint. Verilog HDL, Vedic Mathematics, Von-Neumann architecture, Reduced Instruction Set Computer, and Sutras.

Keywords: Verilog HDL, Vedic Mathematics, Vedic Multiplier, 16-bit RISC Processor, Von Neumann architecture, Reduced Instruction Set Computer (RISC), Arithmetic Logic Unit (ALU), Multiply-Accumulate Unit (MAC), Vedic Sutras, Low Power Design, High-Speed Computing, Processor Architecture, FPGA Design, Vivado.

I. INTRODUCTION

As a research researcher, Bharati Krishna claimed that the sixteen sutras and thirteen sub-sutras essentially represent Vedic Mathematics, but he only dabbled in the subject. Mathematical equations for computations are made easier to understand. Shankaracharya of Goverdhan Peeth and Swami Bharati Krishna Tirthaji Maharaj were two pivotal figures who shed light on the ancient Vedic mathematical system. It is plain and reasonable, and it is also easy to understand. As a result, the subject is becoming increasingly well-known and, as a result, is widely practiced in India and across the globe thanks to Vedic mathematics. In order to carry out operations such as frequency domain filtering (finite impulse response), digital signal processing must contain sixteen distinct sutras that are divided according to the mathematical operations it does. An essential component of the hardware for these activities is multiplication. Therefore, the management of the multiplier is an essential factor in determining the display of the overall framework.

This is because the multiplier is the most powerful and sluggish component in the system. Furthermore, the developers of the framework have a formidable challenge in improving the speed and area of the multiplier. The use of antiquated mathematical methods may be used to successfully defeat this experiment.

Additionally, the digital circuit design has binary multipliers, which make them fast, reliable, and productive in carrying out any work. Various types of multipliers are available throughout the trial. As a result, the Vedic multiplier rests on the foundation of Vedic mathematics, which is robust due to its realism as well as its consistency and fluency. Prior to the mentioned certainty, its most crucial component is laid forth. Similar to the standard array multiplier, the Urdhvas Tiryakbhyam sutras provide a rationale.

The Nikhilam sutra states that, with the exception of the last digit, all of the digits of the number should be subtracted from nine. Which, in order to make things easier to calculate, will be deducted from ten. It finds the closest base to the bigger integer and uses that to multiply it. Subtracting or adding one less than the preceding amount is recommended by Ekanyunena Purvena. When multiplying numbers with a multiplier of nine or a sequence of nines, this specific sutra comes into play. Because it requires repeated addition operations, particularly in applications that need numerous multiplications, square is useful in the Digital Signal Processing (DSP) implementation. In order to determine the square of a number, the Ekadhikena Purvena Sutra is used. Section One: Vedic Mathematics The following are some of the areas of mathematics that Vedic science considers, with the 16 Sutras serving as its foundation: (1) (Anurupye)Shunyamanyat, (2) Cha lana alanabyham The following are the names of the purvenes: iii. Ekadhikinaa, iv. Ekanyune na, v. Gunakasamuchyah, and vi. This is the order of the chapters: Gunitasamuchyah (vii), NikhilamaNavatashcaramam (viii), Paraavartya Yojayet (ix), Puranapuranabyham (x), Sankalana—vyavakalanabhyam (xi), and so on. Chapter xii of Shesanyankena Charamena. Chapters xiii, sopaantyadvayamantyam, xiv, and xv of the Bhagavad Gita are devoted to the Shunyam Saamyasamuccaye. No. xvi, Vyashtisamanstih. Problems in any scientific field may be quickly understood by consulting these sutras (yaavadunam). Vedic mathematics is well-suited for FPGA implementation due to its consistent structure. Data route operators designed using Vedic mathematics have the ability to increase efficiency by increasing throughput. Section B: RRISC Microprocessors The Harvard architecture is implemented by the 16-bit RISC processor, which makes use of separate memory access ports for data and programs. The internal section enroll maintains a 24-bit address for each memory [3]. Therefore, the center has sufficient RAM to run the logical program. In order to successfully get data, the processor makes use of the organization's pipeline structure. In addition, the two-piece repeater is prepared to reduce the looping program's superfluous organization. It has a 32-bit multiplier, a 16-bit barrel shifter, and a 16-bit reasoning unit for processing numbers. Only the standard enlistment of the 16-bit multiplier receives its output.

II. PROPOSED METHOD

The job of the processor, according to the machine language, is to efficiently carry out each set of instructions. An example of a combinational circuit is the Arithmetic Logical Unit, or ALU.

The Arithmetic Logic Unit (ALU) of a processor receives the instruction, a machine word that contains the operation code (opcode), and certain operands in order to process a variety of numbers using various instruction sets. Hence, the opcode specifies the action to be executed by the ALU, and then it uses these operands to carry out that operation. You can save a certain quantity of data in the register bank. The result of an operation is stored in the accumulator by the Arithmetic Logic Unit (ALU). The output is then saved in a register for further use. Additionally, the ALU verifies the operation's success or failure by analyzing the bits and providing an indicator. It will show a matching status if the execution fails. The Z-Flag on the flip side is another name for the status register. The data stored in a computer's memory

is a series of directives or instructions, and its job is to make program execution as efficient as possible. A central processing unit (CPU) has several registers, some of which are devoted to data, addresses, and instructions. Based on the registers, the central processing unit (CPU) executes memory operations such as fetching, decoding, and executing.

The Instruction Register's (IR) duties include identifying the instruction in the operating code, storing the operands in memory, retrieving them from memory, and modifying the CPU to execute the instruction. Importantly, the control unit directs time signals to supervise the same processing components that carry out instructions. After receiving the address from the program counter, the Memory Address Register (MAR), also known as the address buffer, uses it to access memory [1]. When the program counter reaches a new address, it stores the current instruction in the memory region that corresponds to that address. At 100% utilization, the Memory Address Register (MAR) stores binary codes that pinpoint the exact position of a word in the Random Access Memory (RAM). One name for this mathematical method is the "Urdhva Tiryakbhyam" (Vertical and Crosswise) sutra. All it takes is one step for the algorithm to do partial products and sums [24]. If you have n × n bits, the algorithm can handle them. Unlike previous multipliers, this one shows a progressive increase in gate latency and area as the number of bits grows, which is a major plus [18]. In light of its

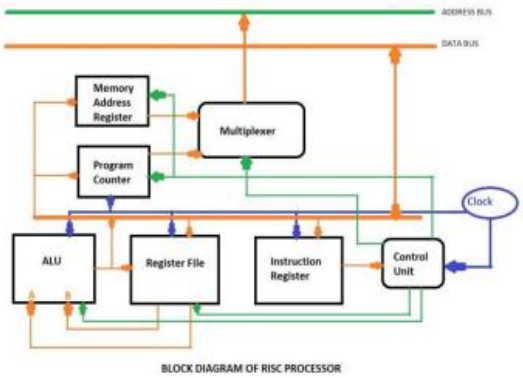

Fig. 1. Block diagram of RISC processor

uniform setup, To increase the multiplier's processing capacity, widen the input and output data buses. After multiplying the numbers at the beginning and end of the line, add this product to the carry from before. All of these results are integrated with the prior carry in a single step. Aside from serving as a carry for the subsequent step, the most extreme digit acquired becomes one of the resultant digits. The "Urdhva Tiryakbhya" method, which incorporates both vertical and transverse operations, is the basis of the hardware constructions of the 2x2-bit, 4x4-bit, and 8x8-bit multipliers found in Vedic mathematics. This method of multiplication from Vedic times allows for efficient multiplication by producing and adding fractional products at the same time. Reducing delays was a major motivating factor for this study, and this method's merit is that it can be easily adapted to parallel processing.Module A. Vedic Multiplier (2×2)

In the beginning, we will represent A as a1a0 and B as b1b0; the following is an overview of the procedure for these two 2-bit values. As a concluding step, the plan takes into account multiplying the least significant bits, which produces the least significant bit of the product (vertically). Then, the immediate higher digit of the multiplier is used to multiply the least significant bit of the multiplicand [28]. To get around this, we add the total to the product of the multiplier's least significant bit and the next higher bit, which is computed crosswise. Then we

add this product to the total. The second bit of the final result is obtained by adding the bits with the highest values, and the carry is added to the partial product that is earned by multiplying the bits with the highest values. This creates the more advanced sum and carry terms.

The third and fourth bits in the resultant product are generated by adding the sum and carry bits [29]. B. A 4-bit module with a Vedic multiplier The 4×4 bit Vedic multiplication unit is made larger by combining four sets of 2×2 multipliers that are comparable to each other [29]. B. Eight-bit Vedic Multiplier Module In order to build 8×8 Vedic multiplier modules, four 4×4-multiplier modules are used. Think about these two 8-bit binary integers, which are shown below: A 16-bit number, S15-S0, is the output of the multiplication. In addition, the values 'a' and 'b' are divided into two equal portions, with aH-aL and bH-bL being the respective sets of four bits. Multiplying the two 8-bit values yields a 16-bit value, which can be expressed as P = a × b = (aH-aL) × (bH-bL) = aH × bH + (aH × bL + aL × bH) + aL × bL. The operation of multiplication is completed thereafter. All the while making use of the four bits that are input to the four-bit multiplier. Two eight-bit values make up the final output, which is obtained by merging the derived product further.

D. N×N bit module with a Vedic multiplier Similar to previous Vedic multipliers we've already covered, the N×N bit Vedic multiplier module is executed in the same way [30]. In comparison to conventional multipliers, the UT Technique-based Multiplier provides a better ordered hardware architecture for integer multiplication. The fact that it can do multiplications quickly is one of its main selling points. You can tell what next instruction is going to be executed by looking at the Program Counter. During each iteration of an instruction, the CPU reads the instruction from the address in memory that the program counter indicates and stores it in the instruction register. An essential component of any system, the Control Unit coordinates the timing and control signals needed by the Central Processing Unit (CPU) to carry out activities.

When this is happening, the ALU is in charge of signal transmission between the main memory, the processor, and the memory, as well as the other busses. The function of the multiplexer block is to choose inputs. Wires that serve as selection lines are within its capabilities [1]. It's a configuration that takes in several signals and outputs only one.

III. SIMULATION RESULTS

A. Urdhva-Triyagbhyam Sutra used in MAC

The Urdhva-based answer to the Vedic multiplier As you can see in the 16-bit multiplier simulation result, Triyagbhyam sutra is there. The two inputs are 252 and 846, and the output is 213192, which is the result of applying MAC methods. When the number of bits increases, the latency and gate area very slightly increase, unlike other multipliers [1]. The speed, power consumption, and processing time of the CPU are all improved by extending the data bus.

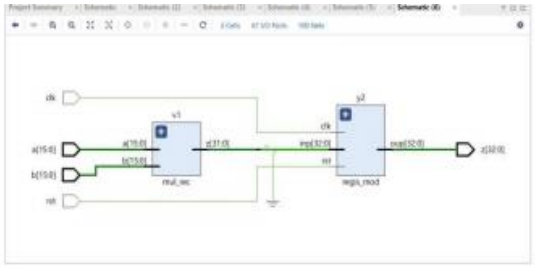

Fig. 2. Using the Urdhva Triyakbhyam Sutra, the RTL schematic of MAC

B. ALU using Urdhva Triyagbhyam

An integral part of the 16-bit ALU design prior to modification is a reversible full adder constructed of gates. In an ALU design, two adders are required.

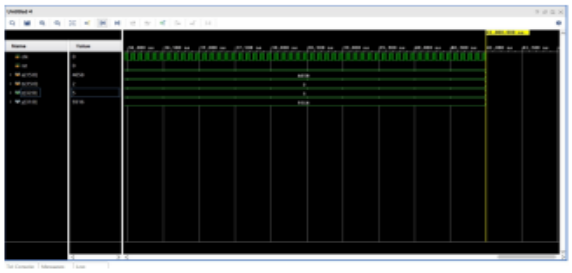

Fig. 3. Simulation of RTL Schematic of MAC

together with four 8-bit multipliers also. Based on the Arithmetic Logic Unit, this simulation is based on the UrdhvaTriyagbhyam sutra, which mimics its fundamental functions. This method requires fewer computer processes to build and produce the Multiplier's findings.

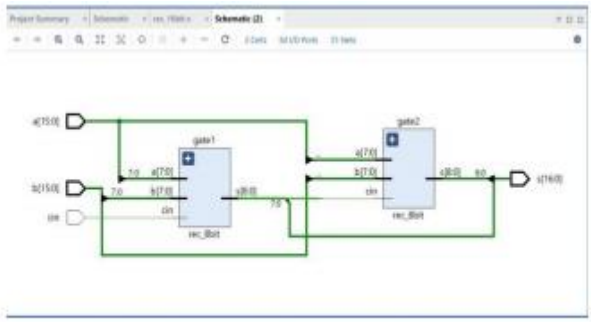

Fig. 4. RTL schematic of ALU before modification

We utilize an adder circuit for additions after modifications. In which we employed full adders and half adders. We have lowered an area and power by changing the adder structure. The only change is to the ALU unit; otherwise, the rest of the CPU remains unchanged as before. When working with 16 bits, we use 2-8 bit adders; when working with 8 bits, we employ 2- 4 bit adders; and while working with 4-bit adders, we employ regular adders. We utilized sixteen complete adders before making the change to the 16-bit adder. A full adder uses AND gates and XOR gates. Since XOR gates are space-hogs, we've been using rectifiers—specifically, a 4-bit adder mirrored in an 8-bit adder—to reduce the adder's area and power consumption. This has allowed us to go from designing a single 4-bit adder to a single 8-bit adder, and then to a single 16-bit adder—all while reducing the adder's area and power consumption. Everything is the same for multiplication as well, with the exception that we represent an 8-bit Vedic multiplier as a 16-bit one [31]. Specifically, we are expressing a 4-bit Vedic multiplier, which is equivalent to a 2-bit Vedic multiplier [32]. Reduced space and power consumption are the results of using a half adder.

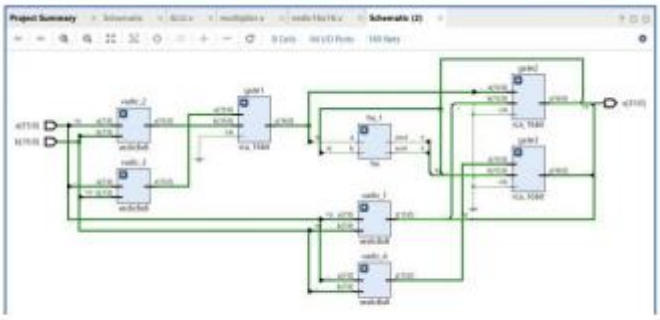

Fig. 5. RTL schematic of ALU after modification

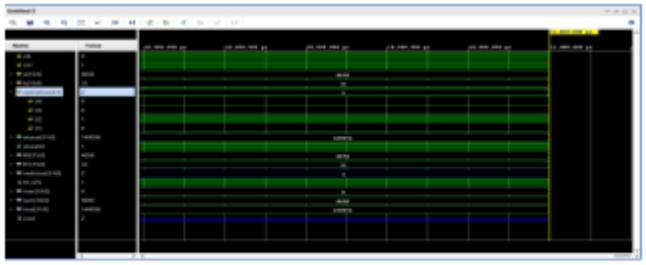

Fig. 6. ALU using Urdhva-TriyagbhyamaSutra for Multiplication

TABLE I
VALUES OBTAINED FROM URDHVAATRIYAGBHYAM SUTRA

Parameters	conventional	vedic	Modified vedic
No. Of slice LUT's for ALU	1086	311	287
No. Of slice LUT's for MAC	56	24	24

C. Simulation Results of Various Arithmetic Operations

Mathematical processes like adding, subtracting, multiplying, and dividing are shown in the following simulation on a dataset of 16 bits [1]. The Urdhva Triyagb hyam technique is used to calculate the simulations for multiplication operations. The four-bit input select lines are collected by the multiplexer for primary arithmetic operations.

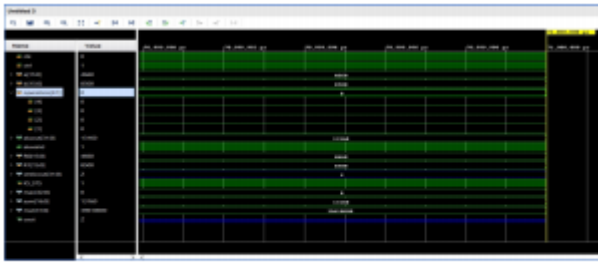

Fig. 7. ALU Addition Operation

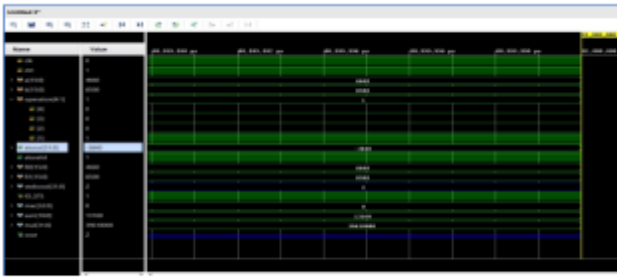

Fig. 8. ALU Subtraction operation

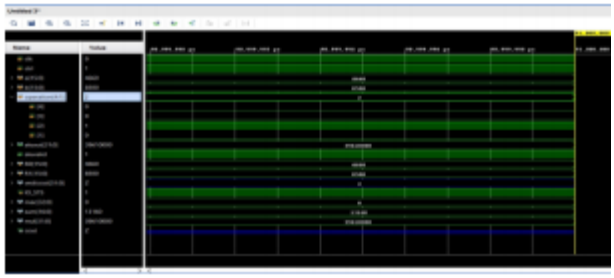

Fig. 9. ALU Multiplication operation

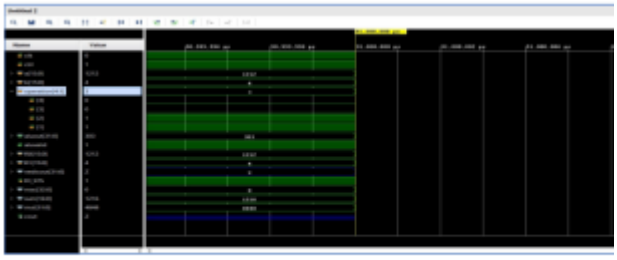

Fig. 10. ALU division operation

D. Processing Unit

The processor design consists of the fundamental components needed to install hardware on a processor. Storage Regis Files, Instruction Registers, Control Signals, and Multiplexer Select Lines are all part of the processing unit [1]. One way to overcome the processor's power consumption, latency, speed, and other restrictions is to include a Vedic multiplier into its design.

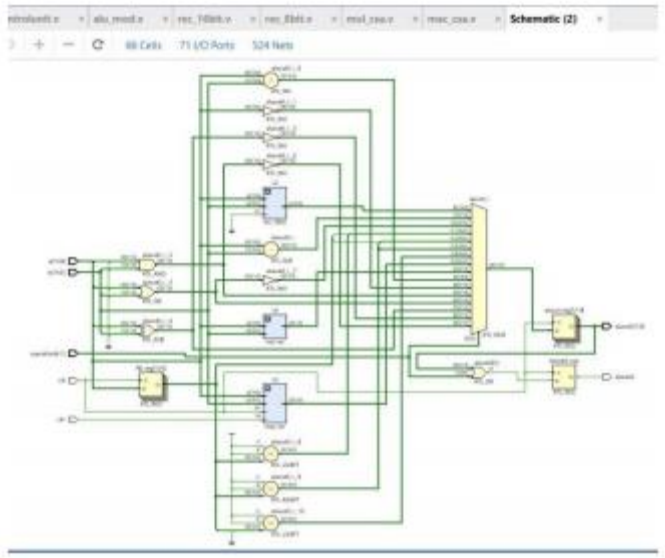

Fig. 11. RTL Schematic processor

A 16-bit CPU, as shown in the above figure, may do multiplier operations and all other common ALU-like operations by integrating a Vedic multiplier block into an ALU. To do the multiplication, the UrdhvaTriyagbhyam Sutra is used. In this way, we have achieved our principal objective of incorporating the Vedic ALU into the proposed CPU design framework. In addition to using variables like power and area to multiply, and

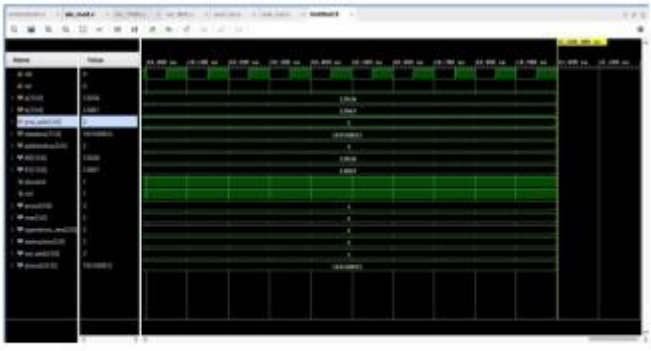

Fig. 12. Simulation of Vedic Multiplier in Processing Unit

comparisons between the original and modified designs are made.

IV. COMPARISON AND RESULTS

Prior to and subsequent to modifications to the addition and multiplication operations, the 16-bit CPU features a Vedic Arithmetic Logic Unit (ALU). Next, we compared them using area and power characteristics.

TABLE II
RESOURCES UTILIZATION BY VEDIC PROCESSOR AFTER MODIFICAT

Slice logic utilization	Used	Available	Utilization
No.of Slice Registers	170	269200	0.063%
No.of Slice LUT's	937	134600	0.69%
Bonded IOB's	74	400	18.5%

A. Before Modifications

Area:

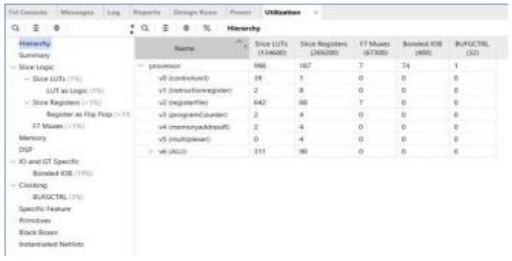

Fig. 13. Area utilization before modification

Power:

Fig. 14. Power utilization before modification

B. After Modifications

Area:

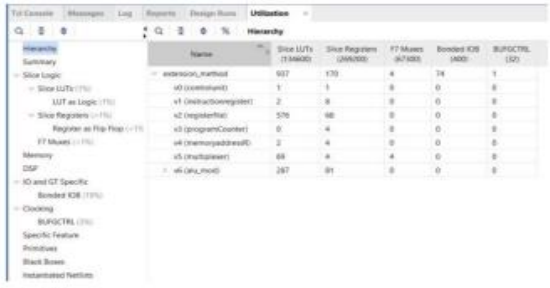

Fig. 15. Area utilization after modification

Power:

Fig. 16. Power utilization after modification

TABLE III
COMPARISON OF CONVENTIONAL PROCESSOR AND VEDIC PROCESSOR (BEFORE AND AFTER)

Parameters	Conventional processor	Vedic processor	Modified vedic processor	Efficiency increase after modification
No. Of Slice LUT's	2123	998	937	6.11%
Power(w)	87	37.388	20.927	44.02%

V. CONCLUSION

An ALU-based RISC processor grounded in the Vedic sutras is put into action in this research. When doing multiplication, Vedic multipliers work to reduce the multiplier's size and power consumption while increasing speed. Further reductions in both space and power consumption have been achieved by means of modification. A CPU is built in this project by combining conventional processor components with a Vedic Multiply-Accumulate unit and an Arithmetic Logic Unit. An instruction register and a single-bit Z flag register work together to monitor the status of arithmetic group instructions; the set of instructions is fourteen in number. The results of the Vedic ALU and MAC designs' simulations are compared to those of the present ALU and MAC designs. There is reduced power consumption and less latency with the 16-bit Vedic processor compared to

REFERENCES

[1] Ankita Yadav and Varsh Bendre. "Design and Verification of 16-bit RISC Processor Using Vedic Mathematics," International Conference on Emerging Smart Computing and Informatics, (2021).

[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." International Journal of Advanced Research in Computer Science and Software Engineering (2015).

[3] Seung Pyo Jung, Jingzhe Xu, Donghoon Lee, Ju Sung Park, Kangjoo 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. Fryza, "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, Ramanathapuram, 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 PoweraProcessor",IJAREEIE, Pune, 2014

[7] P.Nain, and G. S. Virdi. "Multiplier-accumulator (MAC) unit." International journal of digital application and Contemporary Research 5, no. 3 (2016).

[8] P. S. Mane, I. Gupta and M. K. Vasantha, "Implementation of RISC Processor on FPGA," IEEE International Conference on Industrial Technology, Mumbai, 2006, pp. 2096-2100, DOI: 10.1109/ICIT.2006.372448.

[9] Ram, G. Challa, Y. Rama Lakshmanna, D. Sudha Rani, and K. Bala Sindhuri, "Area efficient modified vedic multiplier." In 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), pp. 1-5. IEEE, 2016.

[10] Yogesh M. Motey, Tejaswini G. Panse, "Traditional and Truncation Schemes for Different Multipliers", International Journal of Electronics and Computer Science Engg., vol.2, no.2, pp.627-633, May 2013."

[11] Maroju SaiKumar, P.Samundiswary, "Design and Performance Analysis of Various Multipliers using Verilog HDL", CiiT International Journal of Programmable Device Circuits and Systems, vol.5, no.9, pp.391- 398,Sep2013.

[12] Xilinx13.4, "Synthesis and Simulation Design Guide", UG626 (v13.4) January19,2012

[13] Xilinx 13.1, "RTL and Technology Schematic Viewers Tutorial", UG685(v13.1),March1,2011.

[14] Xilinx, "7 Series FPGAs Configurable Logic Block", UG 474 (v 1.5), August6,2013.

[15] Xilinx 12.4, "ISim User Guide", UG660 (v 12.4), December 14, 2010.

[16] Jikku Jeemon, "Low power pipelined 8-bit RISC processor design and implementation on FPGA" ,ICCICCTa2015.

[17] Supraj Gaonkar and Anitha M, "Design of 16-bit RISC Processor", IJERT, Vol.2, Issue 7, July2013.

[18] D. J. Smith, "HDL Chip Design", International Edition, Doone Publications,2000

[19] J.F. Wakerly, "Digital Design: Principles and Practices", Third Edition, Prentice-Hall,2000.

[20] A. S. Tanenbaum, "Structured Computer Organization", Fourth Edition, Prentice-Hall,2000.