International Journal on Emerging Technologies **11**(2): 770-774(2020)

**ISSN No. (Print): 0975-8364**
**ISSN No. (Online): 2249-3255**

# Enhancement of Features of 8 Bit RISC Processor by Implementing 8 Bit Shift/Add Multiplier

***Tanaji M. Dudhane[1] and T. Ravi[2]***
*[1]Research Scholar, Department of Electronics and Communication Engineering, Sathyabama Institute of Science and Technology, Chennai, (Tamilnadu), India.*
*[2]Research Supervisor, Department of Electronics and Communication Engineering, Sathyabama Institute of Science and Technology, Chennai, (Tamilnadu), India.*

*(Corresponding author: Tanaji M. Dudhane)*
*(Received 23 December 2019, Revised 15 February 2020, Accepted 18 February 2020)*
*(Published by Research Trend, Website: www.researchtrend.net)*

**ABSTRACT: The 8 bit computational engines are superseded by 32 bit core. The 32 bit core is highly dense and available in smaller size as compared to earlier 8 bit cores thanks to improvement in process technology. Though market is populated with 32 bit core, the 8 bit cores are still very relevant and being used in many applications. The PIC16F84 is not capable for execution of 16 bit instructions as well as lack of multiplication operation. This paper integrates the 8 bit multiplier to the 8 bit RISC processor for enhancement of its multiplication capability. This work is an extension of previous work which is in regards with 8 bit core and it's feature enhancement to solidify it against current 32 bit core. The previous work added Co-operative ALU (CALU) in 8 bit core. Herein computational strength of 8 bit core is enhanced by implementing 8 bit multiplication using Shift/Add method. Using FPGA as reconfigurable hardware, the integration of multiplier is carried out with Xilinx 14.7 platform modifying instruction format having 15 bits. The design is carried out using Xillinx 14.7 software tool and Virtex xcv50 device platform as a hardware tool. The features enhancement enables 8 bit core to perform 16 bit addition/subtraction operations and 8 bit multiplication with 16 bit result.**

**Keywords:** CO-ALU, 8 bit core, shift/add multiplier FPGA.

## I. INTRODUCTION

The era of Internet of Thing has begun. It demands the sensor node processors with high capabilities as well as low power consumption. Also, the automation in the automotive field is increasing as well, which also demands low power sensor node processors with strong capabilities. Thus, we have selected the 8-bit PIC 16F84 processor for the functional enhancement, where we will enhance its capabilities with the fast multiplier unit based on shift/add multiplier design. This paper proposed complete multiplier design which is simulated and integrated with 16F84 microcontroller of 8 bit. The result is simulations of multiplication 8 bit shift/add multiplier design for improvement in area and speed.

Here are some researches carried out in the multiplier design. Ram *et al.*, (2016) has designed the 8-bit Area Efficient Modified Vedic multiplier. Urdhva- Tiryagbhyam sutra is the base of multiplier design which is ancient vedic. The multiplier is designed using the Verilog HDL and synthesized on the Spartan 3E platform. It has been designed using the BEC adder over the conventional ripple carry adder in order to improve the area utilization [1]. As the process of repeated additions is the multiplication, the adder is the main constituent in the multiplier. The speed of the adder performs the vital role in the performance of the multiplier algorithm. Gokhale and Gokhale (2015) has designed Vedic Multiplier using carry select adder for area and delay. The significant improvement in speed in the design shows than the Modified Vedic Multiplier and also utilizes lesser area [2]. Gupta *et al.*, (2014) has proposed technique of Vedic multiplication using the compressor. The multiplier is designed using the 4:2 and 5:7 compressors as replacement to the adders. The method shows the 6% of hardware reduction than the conventional Vedic multiplier [3].

Dey and Chattopadhyay (2017) has proposed the 8-bit High performance binary multiplier using the Vedic sutras with the 16nm technology. The design is based on the sutra of Urdhva-Tiryagbhyam. It takes about 99.50% less time delay compared to the 65nm technology, which considerably reduces the power consumption [4]. Kumar and Charlie (2014) has presented the image processing using 2d-DWT Multilevel. The multiplier and accumulator unit in the FIR filter is redesigned using the Novel Vedic multiplier instead of conventional Vedic multiplier. There is reduction in area and power requirements using the Novel Vedic multiplier [5]. Kahar and Mehta (2017) has demonstrated the implementation of 4, 8, 16, 32 and 64-bit Vedic multiplier which consumes 11.477 ns, 21.550 ns, 28.086 ns, 30.956 ns and 44.377 ns respectively. The area and delay requirements are compared with the other author's proposed designs. Significant improvement has been recorded in speed [6]. Kunchigi *et al.*, (2012) has proposed vedic multiplier with high speed. The base of the design is the Urdhva Tiryagbhyam sutra. The implementation is performed on the Spartan 3E platform. It has been recorded that the method takes 4.585ns of time to perform the multiplication which is faster than the Array and Booth multipliers [7]. Pohokar *et al.*, (2015) has implemented

16 bit multiplier which multiplies 16 bit number using Urdhva-Tiryagbhyam sutra. Carry save adders are used as the part of the multiplier for addition of partial and final products. The 16-bit Vedic Multiplier using carry save adder takes 28.779 ns. The multiplier is designed using the VHDL language in Xilinx ISE 10.1. The results are compared with the array multiplier design [8].

Katreepalli and Haniotakis (2017) hierarchal approachdesign of efficient power-delay of Vedic multiplier using MCC which is Manchester Carry Chain adders. Lower power-delay product requirement using MCCis the proposed work of multiplier design than architectures of Vedic multipliers which are in existence [9]. Goswami *et al.*, (2014) has expressed in the paper about energy efficient and digitally controlled impedance, low voltage multiplier. The standards of the low voltage digitally controlled impedance (LVDCI) are available on reconfigurable hardware. By using the LVDCI block on the 28nm technology, author has demonstrated performance of the Vedic multiplier in the various temperatures [10]. Lakshmi *et al.*, (2016) has proposed high speed vedic BCD multiplier using Vinculum method. Due to the use of vinculum method there are only 1 to 5 numbers, this reduces the carry generation and the propagation there by improving the performance in terms of the delay. The BCD multiplier with vinculum takes 12.74 ns which is far faster than others [11].

Rao *et al.*, (2016) has proposed the complex multiplier using minimum delay real multiplier architecture. Proposed design will play noticeable performance improvement in the high performance wireless communication systems. The proposed designed takes the lesser computation time than the complex array multiplier and the complex booths multiplier [12]. Huddar *et al.*, (2013) has presented the speed, area efficient ALU design using ancient Vedic mathematics and implemented it on the reconfigurable hardware. The implementation shows that the proposed method is 2 times faster and consumes 3% lesser area than conventional one. The implementation is done on Spartan 3E platform [13]. Gupta *et al.,* (2012) has presented design of speed multiplier using reversible logic gate. Speed efficiency as well as energy efficiency for the arithmetic and logic unit which affects on efficiency of power, provided by reversible logic gate. Because of this the design presented requires less area for gates and hence making ALU more powerful. The designed unit of ALU is efficient in microcontroller/microprocessor and the performance of central processing unit depends on performance of ALU. Optimization is achieved by developing MAC unit of digital signal processing. Such design of ALU is used for optimization of different algorithms which are based on IIR, FIR, FFT, etc. for DSP [14]. For the formation of complex digital circuits, FPGAs are conceptually array of Configurable Logic Blocks (CLBs), connected together thorough a vast connecting matrix. The paper Implements logical, add, sub, inc, dec. instructions but not multiplication instruction also having instruction length of 20 bits, requiring more memory space [15].

In a single FPGA, one or more microprocessor/microcontrollers can be embedded. Though the work talks about the design of microcontrollers and microprocessors completely implemented in FPGA, none of the work is related with the implementation of multiplication instruction. The instruction format is 13 bits as compared to our work having 15 bit instruction format [16, 17]. The research work by JER Prieto and FM Santa, said about the optimization in fast systems using 14 bit instruction length [18], in contrast, our work using 15 bit format, in which MSB bit, separates the original instructions of 8 bit and additional instructions of 16 bit. Vedic multiplier is based on vertical and crosswise structure of vedic mathematics [19, 20, 21 24]. We have proposed the design of multiplier using Shift/Add method and its integration to RISC processor for enhancement of its functionality and finally developing VHDL code for the multiplier, using FPGA, for Improvement in speeds and area. The power dissipation is 600 mW [22] but by implementing 11 more instructions having size of 16 bits, improving power dissipation of 15 mW [17]. Power dissipation, propagation delay, chip area, these parameters are discussed and calculated from simulation results [23].Enhancement 8 bit processor to execute 16 bit instructions with addition of more number of 16 bit instructions, is the lack of research work of the previous papers.

The previous study of many research paper was the lack of implementation of 16 bit instructions with the original 8 bit core as well as lack of multiplication operation. Original processor PIC16F84 implements instruction like arithmetic, logical etc. category with 8 bits. We have selected the 8 bit processor for the functional enhancement. The 8 × 8 multiplier is integrated to the RISC processor, implementing multiplication operation where the processer requires number of operation repeatedly. The proposed system using reconfigurable hardware platform like Field Programmable Gate Array that is FPGA , implements the 8 bit multiplication operation for the PIC 16F84, RISC processor for improvement in speed and area.

The aim of this paper is to design and implement 8 bit shift/add multiplier. The paper consists as follows. Section Second describes the proposed design and requirement of multiplier, Section III describes methodology, Section IV with Results and conclusion in the Section V.

## II. PROPOSED DESIGN

As shown in Fig.1, 8 bit multiplier is comprised of the following blocks. First block is 8 bit register for multiplicand, second block is 8 bit register for multiplier, third block is 16 bit register for adder and finally forth block is shift and add control logic. Shifter is for right shifting and checking LSB. If LSB is 0, add 0 as it is otherwise add multiplicand.

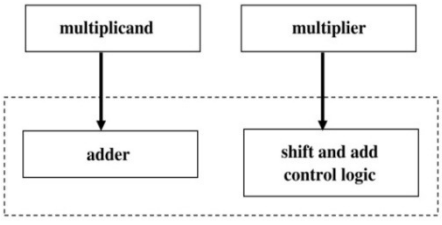


**Fig. 1.** 8 bit shift/add multiplier block diagram.

After addition, result is shifted towards left by one bit. Multiplication produces 16 bit result. The design is coded in VHDL and simulation is carried out for proper timing and functionality.

## III. METHODOLOGY

*A. Multiplier Design*

Proposed multiplier design is based on shift and add method.

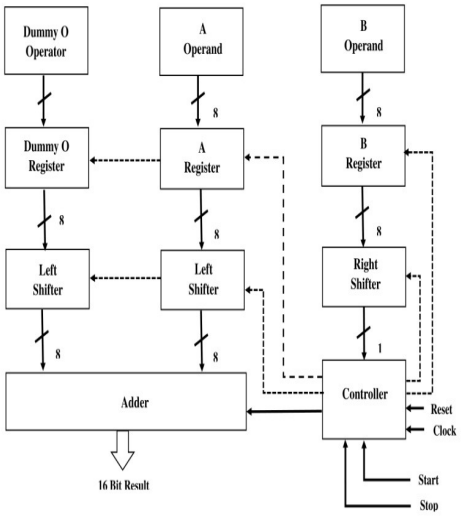


**Fig. 2.** Block diagram of Multiplier design.

Fig. 2 demonstrates the design flow for 8 bit multiplication. It is comprised of registers to hold multiplier, multiplicand and dummy register, whose value is 0 and being used only once in the beginning. Execution of multiplication operation starts with the reception of 'start' trigger pulse and ends with the generation of 'stop' pulse. Operand registers are 8 bits and result register is of 16 bits. The controller along with adder and shift registers participates in multiplication operation. The multiplier is shifted towards right by one bit position. Result register is shifted towards left by one bit position for each round of multiplication operation. For example for 8 bit operation multiplier gets shifted towards right by one bit position 8 times as the multiplication operation is being conducted for 8 bit operands.

Similarly, result register is shifted towards left by one bit position 8 times. After shifting of multiplier towards right by one bit position, the LSB bit comes out of it, decides whether 0 or multiplicand is to be added with previous result. For example if bit is 0, '0' is added otherwise multiplicand gets added in previous result. Controller continues operation till it completes $8^{th}$ round.

*B. Controller Design*

The Fig.3 shows the FSM diagram controller. The multiplier goes through various states such as initial state, load operand, store operand and computation state. Initially system is in initial state. It remains in initial state until it receives start trigger pulse. There in, operands such as multiplicand and multiplier loaded into respective holding register.

Once operands are loaded, system enters into computation state and carries out computation. For initial state of computation, count value is 0 and it gets incremented after completion of each round of operation. System remains in computation state as long as count value is less than 8. It leaves computation stage and enters into store operand stage when count value reaches at 8. In this state, end result is stored.

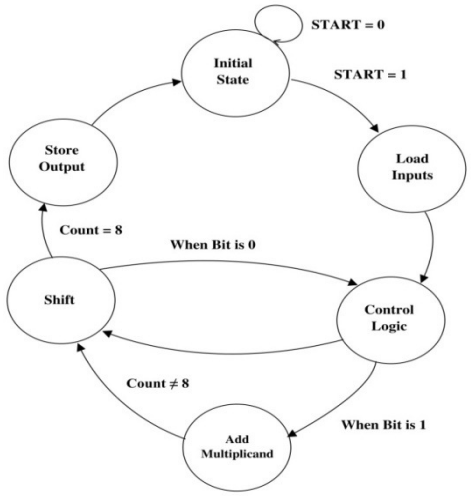


**Fig. 3.** FSM diagram of the controller.

*C. Multiplicand Design*

The basic design for the Multiplicand block is that of an 8-bit register.

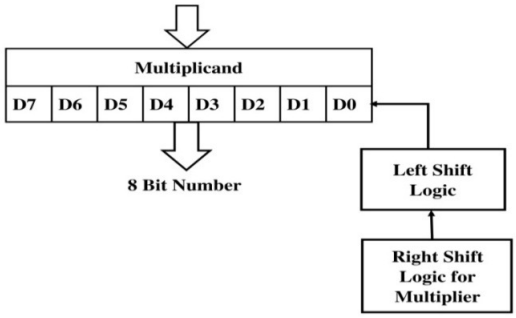


**Fig. 4.** Diagram of Multiplicand.

The Fig. 4 shows the multiplicand diagram having 8 bits. It is 8 bit register having bits D0 – D7. Left shift logic is carried out for 8 bit multiplicand and right shift logic is achieved for 8 bit multiplier.

*D. Adder Design*

Fig. 5 is the Data flow of the 8 Bit multiplication.

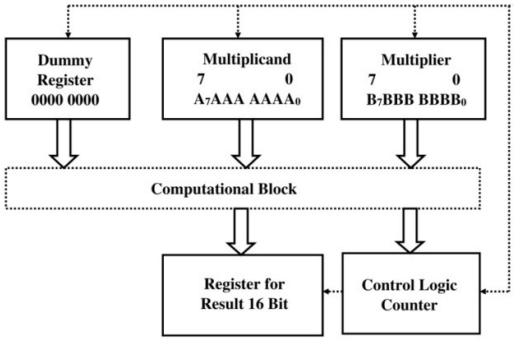


**Fig. 5.** 8 Bit multiplication data flow diagram.

Algorithm:
– Start
– Initialize count = 0
– Shift result register by count value.
– Shift multiplier right and check LSB bit.
– If bit is 0, add dummy register value with result register.
– If bit is 1, add multiplicand value with result register.
– Increment counter.
– If less than 8, go to 3, otherwise stop.

Fig. 6 shows the Enhanced CALU architecture. While enhancing the ALU architecture of the original 8 bit RISC processor, 8 bit multiplier is integrated with 8 bit processor and 16 bit co-operative ALU is integrated.

For 16 bit Co-ALU, the size of instruction register is changed from 14 bits to 15 bits adding one bit in MSB. This bit differentiates the 8 bit and 16 bit instructions. If MSB is 0 in instruction register, then 8 bit instructions are executed and when MSB is 1, 16 bit instructions are executed.

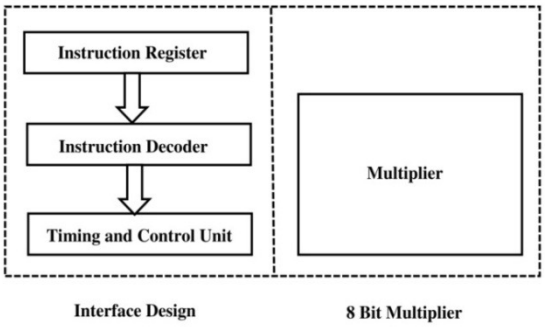


**Fig. 6.** Enhanced CALU architecture.

## IV. RESULTS

The results are captured for small, moderate and big 8 bit numbers.The original PIC 16F84 RISC processor, is 8 bit having lack of multiplication operation. For the enhancement of its functional ability to make it compatible with 16 or 32 bit processors, 8 by 8 bit multiplier is integrated, simulated. VHDL code is developed for small, big and moderate 8 bit numbers. Simulation of multiplication result are shown, enhancing its functionality and same processor is developed as an IP core.

Fig. 7 shows the simulation of the multiplication result.


**Fig. 7.** Simulation diagram of multiplication of two small numbers.

Fig. 7 represents a simulation diagram for two small numbers i.e. A=02 and B=05. The overall time required is shown in terms of number of cycles.

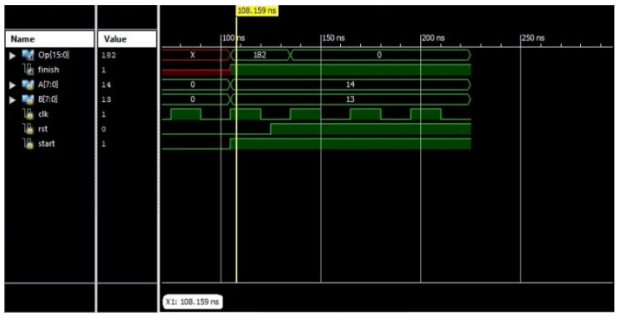


**Fig. 8.** Simulation diagram of multiplication of two moderate numbers.

Fig. 8 shows a simulation diagram for two moderate numbers i.e. A=13 and B=14. The overall time required is shown in terms of number of cycles.

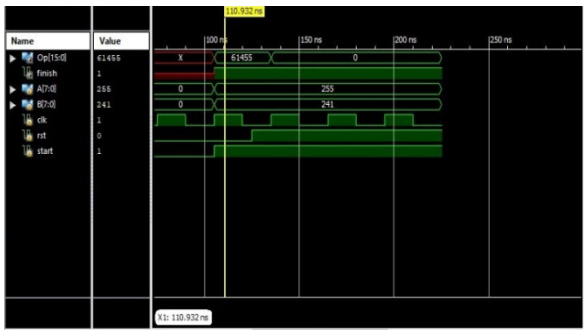


**Fig. 9.** Simulation diagram of multiplication of two big numbers.

Fig. 9 represents a simulation diagram for two big 8 bit numbers i.e. a=241 and b=255. The overall time required for the multiplication of two 8 bit large value numbers is shown in terms of number of cycles.

## V. CONCLUSION

The successful implementation of 8 bit multiplier is achieved. The VHDL code is successfully built using Xilinx ISE tool and simulation results are verified. The final implementation is done using Virtexxcv50 hardware platform. The results are found for different 8 bit numbers.

## VI. FUTURE SCOPE

The 8 bit multiplier is implemented and integrated with 8 bit processor. This is additional enhancement which is implemented with the processor which was lack of this in hardware. Future scope is that, such microcontroller may be used where multiple operations required multiple iterations for speed improvement. Also, using Soft cores, implementing on FPGA, gives a solution for development of such processors with high speed and with minimum area for industrial applications.

**Conflict of Interest.** Co-operative ALU (CALU) and Arithmetic Logic Unit (ALU). **C-ALU:** This co-operative arithmetic logic unit (CALU), is integrated to the 8 bit PIC 16F84 PIC processor, which executes 16 bit

instructions, making compatible this 8 bit processor with 16 bit or even 32 bit processors. Without changing the original functionality of the 8 bit processer, this 16 bit CALU, performing 16 bit instructions. So, 11 more arithmetic and logic instructions, having width 16 bits, executed with the addition of CALU. **ALU:** For 8 bit RISC processor 16F84, having 8 bit arithmetic and logic unit, performing 8 bit operations. This ALU is performing and executing 8 bit Instructions This ALU is responsible for executing 8 bit arithmetic and logical instructions of the origin core. One MSB bit which is newly added in instruction format, making 15 bits, differentiates between 8 bit and 16 bit instructions.

## REFERENCES


[1]. Ram, G. C., Lakshmanna, Y. R., Rani, D. S., & Sindhuri, K. B. (2016). Area efficient modified vedic multiplier. *International Conference on Circuit, Power and Computing Technologies in India*, 1-5.

[2]. Gokhale, G. R. & Gokhale, S. R. (2015). Design of area and delay efficient Vedic multiplier using Carry Select Adder. *International Conference on Information Processing in India*, 295-300.

[3]. Gupta, R., Dhar, R., Baishnab, K. L., & Mehedi, J. (2014). Design of high performance 8 bit Vedic Multiplier using compressor. *International Conference on Advances in Engineering and Technology in India*, 1-05.

[4]. Dey, K., & Chattopadhyay, S. (2017). Design of high performance 8 bit binary multiplier using vedic multiplication algorithm with 16 nm technology. *First International Conference on Electronics, Materials Engineering and Nano-Technology in India*, 1-05.

[5]. Kumar, J. V., & Charlie, P. C. K. (2014). Design of modified vedic multiplier and FPGA implementation in multilevel 2d-DWT for image processing applications. *Second International Conference on Current Trends In Engineering and Technology in India*, 508-511.

[6]. Kahar, D. K. & Mehta, H. (2017). High speed vedic multiplier used vedic mathematics. *International Conference on Intelligent Computing and Control Systems in India*, 356-359.

[7]. Kunchigi, V. Kulkarni, L., & Kulkarni, S. (2012). High speed and area efficient vedic multiplier. *International Conference on Devices, Circuits and Systems in India,* 360-364.

[8]. Pohokar, S. P., Sisal, R. S., Gaikwad, K. M., Patil, M. M., & Borse, R. (2015). Design and implementation of 16 × 16 multiplier using Vedic mathematics. *International Conference on Industrial Instrumentation and Control in India*, 1174-1177.

[9]. Katreepalli, R., & Haniotakis, T. (2017). Power-delay-area efficient design of vedic multiplier using adaptable manchester carry chain adder. *International Conference on Communication and Signal Processing in India,* 1418-1422.

[10]. Goswami, K., Pandey, B., Jain, A., & Singh, D. (2014). Low Voltage Digitally Controlled Impedance Based Energy Efficient Vedic Multiplier Design on 28nm FPGA. *International Conference on Computational Intelligence and Communication* Networks *in India*, 951-955, doi:10.1109/CICN.2014.201.

[11]. Lakshmi, G. S., Fatima, K., & Madhavi, B. K. (2016). Implementation of high speed Vedic BCD Multiplier using Vinculum method. *IEEE Region 10 Conference (TENCON)* in Singapore. 147-151, doi:10.1109/TENCON.2016.7847978.

[12]. Rao, K. D., Gangadhar, C., & Korrai, P. K., (2016). FPGA implementation of complex multiplier using minimum delay Vedic real multiplier architecture. IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering in India, 580-584, doi: 10.1109/UPCON.2016.7894719.

[13]. Huddar, S. R., Rupanagudi, S. R., Janardhan, V., Mohan, S., & Sandya, S. (2013). Area and Speed Efficient Arithmetic Logic Unit Design Using Ancient Vedic Mathematics on FPGA. *International conference on Advances in Computing, Communication, and Control in Berlin,* 475-483.

[14]. Gupta, A., Malviya, U., & Kapse, V. (2012). Design of speed, energy and power efficient reversible logic based vedic ALU for digital processors. Nirma University international conference in India, 1-06, doi:10.1109/NUICONE.2012.6493259.

[15]. Kumar, D., & Anusudha (2013). RISC System Design in Xilinx. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, *2*(4), 1406-1411.

[16]. Pardo, D. F. G. (2006). Embedded microcontrollers and FPGAs Soft-cores. *Electronica-UNMSM. N*(18), 3-14.

[17]. Dudhane, T. M., & Ravi, T. (2019). Design and Implementation of Extended 16 Bit Co-operative Arithmetic and Logic Unit (CALU) for 16 Bit Instructions. *Journal Low Power Electronics, 15*(3), 309-314.

[18]. Prieto, J. E. R., Gomez, E. J., & Santa, F. M. (2017). Implementation of an 8 Bit Soft-core microcontroller on Xilinx Spartan FPGA family. *Indian Journal of Science and Technology, 10*(22). 1-7.

[19]. Pushpalata, V. , & Mehta, K. K. (2012). Implementation of an Efficient Multiplier based on Vedic Mathematics Using EDA Tool. *International Journal of Engineering and Advanced Technology*, *1*(5), 75-79.

[20]. Kumar, G. G., & Charishma, V. (2012). Design of high speed vedic multiplier using vedic mathematics techniques. *International Journal of Scientific and Research Publications*, *2*(3), 2250-2253

[21]. Poornima, M., Patil, S. K., Shivukumar, Shridhar, K. P., & Sanjay, H. (2013). Implementation of Multiplier using Vedic Algorithm. *International Journal of Innovative Technology and Exploring Engineering, 2*(6), 219-223.

[22]. Nanda, A., & Shreetam, B. (2014). Design and Implementation of Urdhva-Tiryagbhyam based Fast 8 × 8 Vedic Binary Multiplier. *International Journal of Engineering Research and Technology, 3*(3), 1856-1859.

[23]. Senthilpari, C., Singh, A. K., & Diwakar, K. (2008). Design of a low-power, high performance, 8 × 8 bit multiplier using a Shannon-based adder cell. *Microelectronics Journal, 39*(5), 812-821.

[24]. Asmita, H. (2011). A Novel Design for High Speed Multiplier for Digital Signal Processing Applications (Ancient Indian Vedic mathematics approach). *International Journal of Technology and Engineering System, 2*(1), 27-31.

[25]. Lee, J. D., Yong, J. Y., Lee, K. H., & Park, B. G. (2001).Application of Dynamic Pass-Transistor Logic to an 8-Bit Multiplier. *Journal of the Korean Physical Society*, *38*(3), 220-223.

[26]. Lee, J. Y., Garvin, H. L., & Slayman, C. W. (1987 ). A high-speed high-density silicon 8/spl times/8-bit parallel multiplier. *IEEE Journal of Solid-State Circuits, 22*(1), 35-40.



**How to cite this article:** Dudhane, T. M. and Ravi, T. (2020). Enhancement of Features of 8 Bit RISC Processor by Implementing 8 Bit Shift/Add Multiplier. *International Journal on Emerging Technologies*, *11*(2): 770–774.