DESIGN AND IMPLEMENTATION OF 64-BIT VEDIC MULTIPLIER FOR DIGITAL SIGNAL PROCESSING (DSP) APPLICATION

Arunkumar P Chavan, A Prathibha , Divya H

Abstract

Multiplication is a very important operation in many DSP applications. This paper states the implementation of a high-speed 64 x 64 bit Vedic multiplier using the Urdhva-Tiryagbhyam sutra. The basic building block of multiplier is Adder, most of the DSP applications demand faster adders for arithmetic computations. Multipliers are designed using different adders such as Carry Select adders, Carry Skip adders and Carry Look Ahead adders. This paper compares the performance of a 64 x 64 bit Vedic multiplier using these three adders and the Karatsuba multiplication algorithm. The proposed algorithm is developed using verilog HDL. Implementation has been done using Xilinx14.3, and synthesized using a cadence tool.

Keywords

64x64 Vedic multiplier, Urdhva-Tiryagbhyam sutra, Carry skip adder, Carry select adder, Carry look ahead adder.

Introduction

Multipliers are the most commonly used computational blocks in DSP applications. There are several ways in which the multiplier circuit can be implemented. The variations chiefly depend on the type of adders which are being used. Depending on the type of adder used we get variations in power, delay and performance. Our chief objective is to demonstrate the differences in speed and power by using different adders.

Performance of the multiplier is compared using a Carry select adder, a Carry look ahead adder, and a Carry skip adder. A structural modelling approach is used to build the 64x64 multiplier by using 2x2 multipliers and corresponding adders of the above mentioned types. From the simulations it is observed that the multiplier implemented using Carry skip adder is faster compared to other adders.

Karatsuba is an algorithm developed for performing multiplication efficiently. It is used to multiply two n-digit numbers in n^{log3} time. But it involves a lot of computational steps which will increase the power consumption as can be seen from the inference we have drawn in this paper.

This paper has been organised as follows, Section II contains an introduction about various adders used in this paper. Section III contains the explanation for the 64x64 multiplier using Urdhva-the Vertically and crosswise (Urdhva-Tiryagbhyam) sutra. Section IV discusses the results obtained and section V is the conclusion.

II Adders

Adders are the combinational circuits used to perform the addition operation in computing systems. These circuits add bits of data and the result obtained can be further processed to make decisions or can be saved for future steps. They form the basic component of a multiplier circuit. Multiplication can be implemented using repetitive addition and shift operations.

(a) Carry Skip Adder

In carry skip adder, full adders are used to generate the sum bits. The XOR logic is used to generate the "propagation bits", which are fed to an AND logic to produce the "generate bits". The generate bit acts as a select input for the multiplexer which is used to select from amongst the carry inputs to finally produce the out carry bit. Multiples of these stages can be cascaded in parallel to build the respective higher order adders, while the carry from the previous stage is fed as an input carry to the next stage [1].

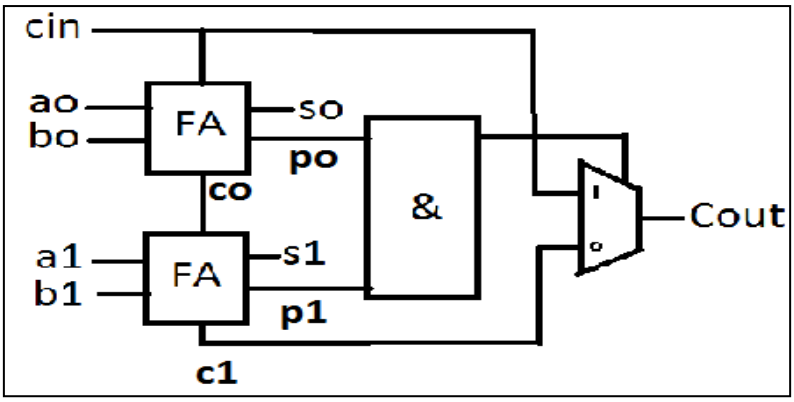

Fig.1. 2- Bit Carry Skip Adder

(b) Carry-Look ahead Adder

Carry look ahead adders are used to overcome the delay incurred by the ripple carry adder wherein the carry has to propagate through all the successive stages and the corresponding stages have to wait for carry from the previous stages before they can compute the sum. Carry look ahead adders overcome this by calculating the carry well in advance using a logic circuit. The successive stages will no longer have to wait for the carry to propagate and can carry on with the calculation of the sum bits [1].

Before computing the actual sum bits it is essential to calculate the propagate and generate bits which depend only on the input bits. Since the input bits are readily available all the time there is no delay incurred. The propagate bits are computed as the EXOR of the input bits and the generate bits are the AND of the input bits. Depending on the inputs and the previous carry the circuit will determine if the carry needs to be propagated or if it has to be generated or killed.

The logic can be represented using the equation

C(i+1)=G(i)+P(i)C(i) ..... (1)

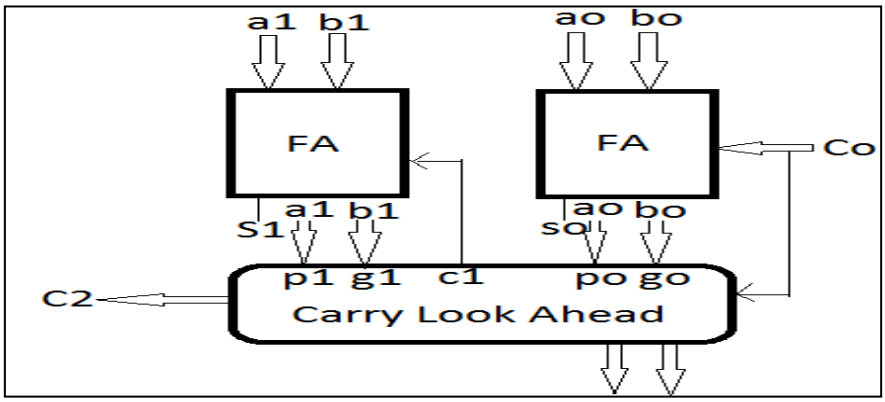

Fig.2. 2-bit Carry - look ahead Adder

(c) Carry Select Adder

The carry select adder provides the best performance in terms of speed and delay. There are two sets of full adders which are given complementary carry inputs. The same inputs are given to both sets of adders. The final sum bits are multiplexed using multiplexers whose inputs are the outputs from the two stages of the full adders. The select line for the multiplexers is the carry in from the input bit. The final carry out is also generated using a multiplexer whose select line is the carry in signal. The successive stages can be cascaded to get adders of higher order [1]. The carry select adder can be used to reduce the delay to a great extent when compared to the carry skip adders and carry look ahead adders.

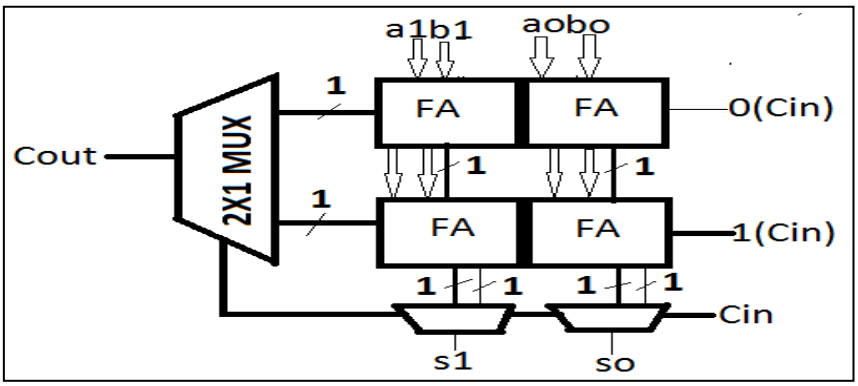

Fig.3. 2-bit Carry Select Adder

III Urdhva-Tiryagbhyam Multiplier

The Urdhva Tiryagbhyam multiplier is implemented using the Vedic Algorithm Urdhva Tiryagbhyam sutra. It is used to carry out multiplication operations quickly. It reduces the time required for n-bit multiplication [5]. The block diagram explains the various blocks and the corresponding operations it performs.

For an n-bit multiplier, the circuit consist of four n/2 multipliers. The inputs are broken down into half the size of the original input data size and are fed to the multipliers in different orders so as to get all the combinations. This is in accordance with the algorithm which uses vertical and crosswise multiplication of the inputs successively to compute the final product.

The lower product from the lower n/2 multiplier is assigned to the lower order of the product directly. This corresponds to the first vertical multiplication that is done in the algorithm [2]. The product of the middle adders is given to an n-bit adder. The sum of this adder is given to another n-bit adder whose other inputs are the higher product of the first multiplier and zeros for padding. The lower product of this multiplier forms a part of the final product.

The product from the final multiplier is fed to another n-bit adder whose inputs are carry from the first adder, sum from the second adder and padding zeros. The sum of the third adder forms the remaining part of the final product.

This successive combination of the crosswise and vertical multiplication is exactly mimicking the operation performed by the Urdhva Tiryagbhyam algorithm [4]. Higher order multipliers can be built using the lower order multipliers and a combination of different adders. Vedic multipliers are very fast and can be used to reduce the delay in a wide variety of applications.

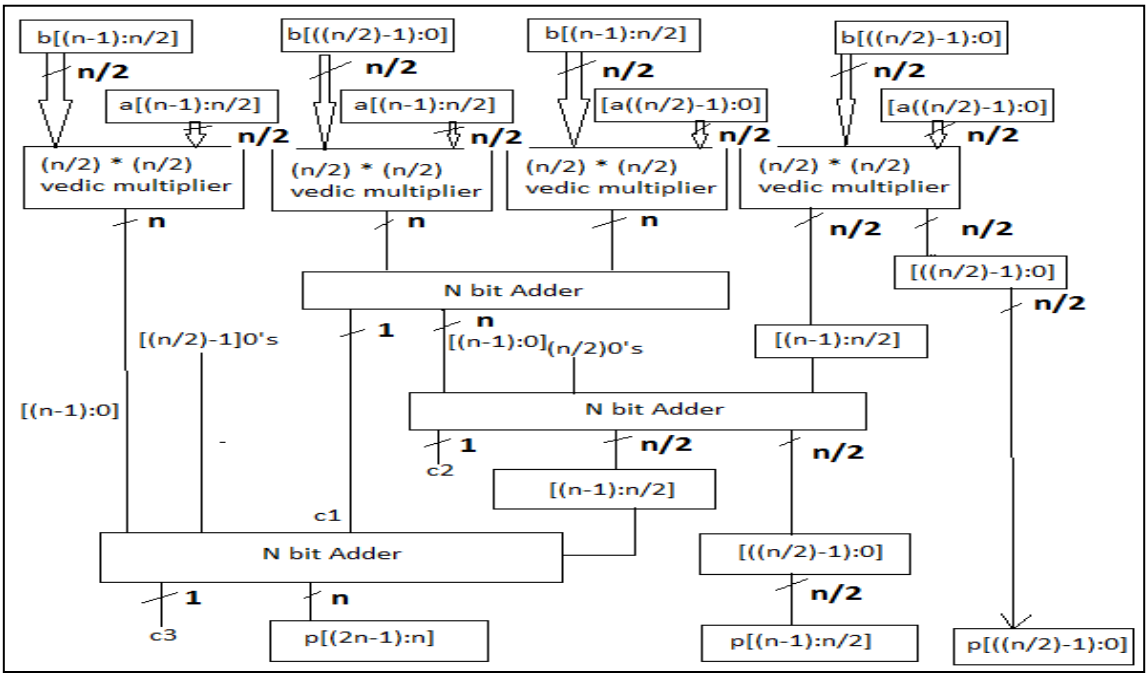

Fig.4. Block diagram of N x N-Bit Vedic Multiplier

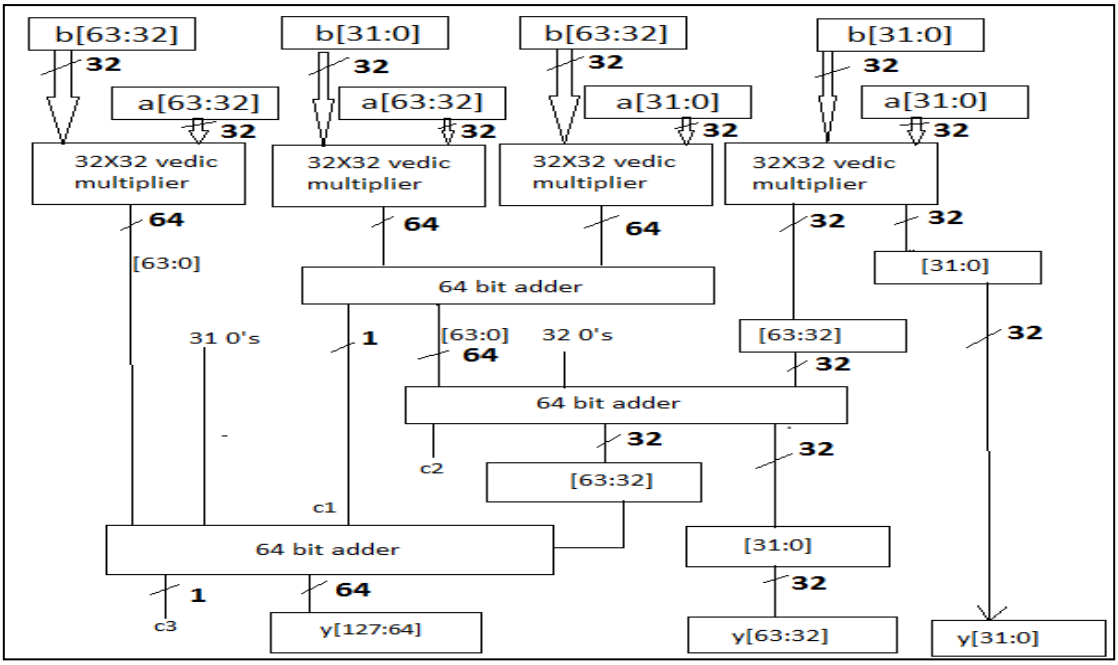

Fig.5. Block Diagram of 64 X 64 Vedic Multiplier.

IV Results

An RTL schematic view of 64-bit multiplier using Carry skip adder is shown in Fig 6. It is implemented using structural modeling. The 64-bit multiplier is implemented using the 32-bit multiplier and so on. Simulation results of 64-bit RTL Schematic view of 64-Bit Vedic Multiplier are shown in Fig.7.

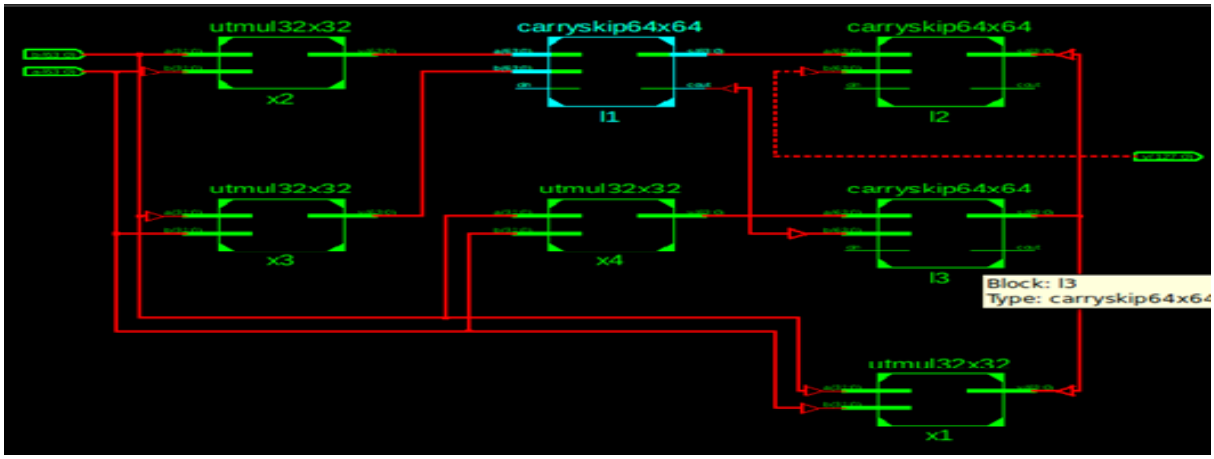

Fig.6. RTL Schematic view of 64-Bit Carry skip adder.

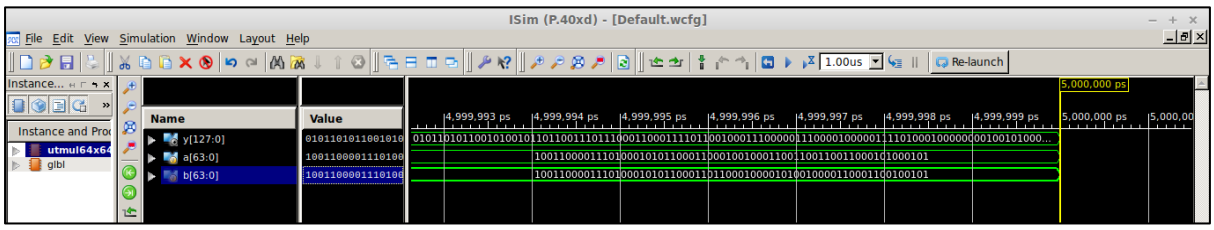

Fig.7. Simulation results for 64-Bit Vedic Multiplier using Carry Skip Adder

An RTL schematic view of 64-bit multiplier using 6carry look-ahead adder is shown in Fig 8. See Fig.9 for simulation results for 64-Bit Vedic Multiplier using Carry look ahead adder.

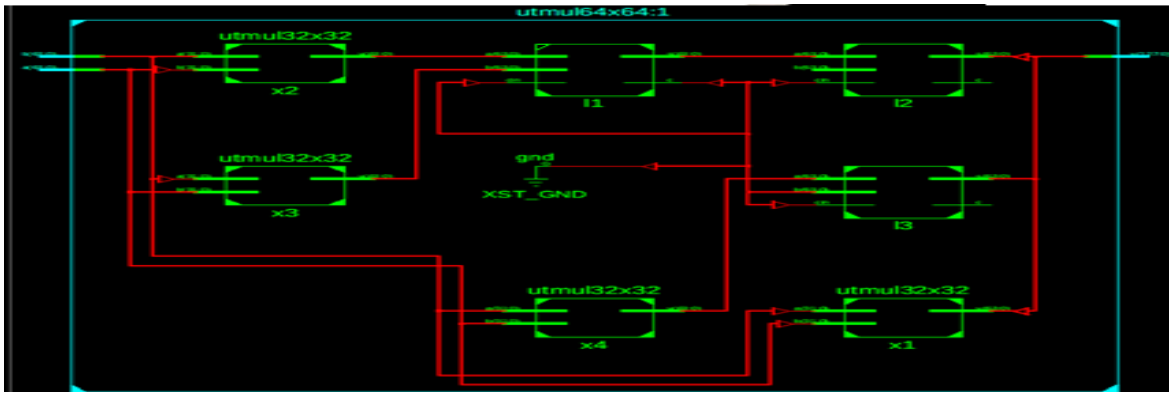

Fig.8. RTL Schematic view of 64-Bit Vedic Multiplier using Carry-Look ahead Adder

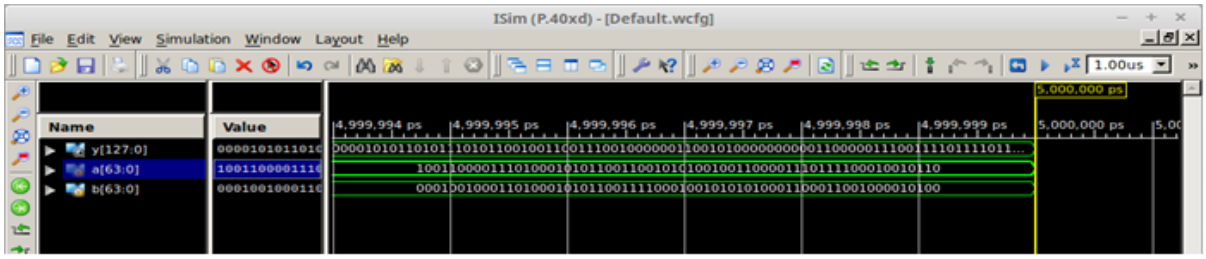

Fig.9. Simulation results for 64-Bit Vedic Multiplier using Carry-Look ahead Adder

In Fig.10 below is the RTL schematic view of the 64-bit multiplier using the carry select adder. Simulation results of multiplier using carry select adder are shown in Fig.11.

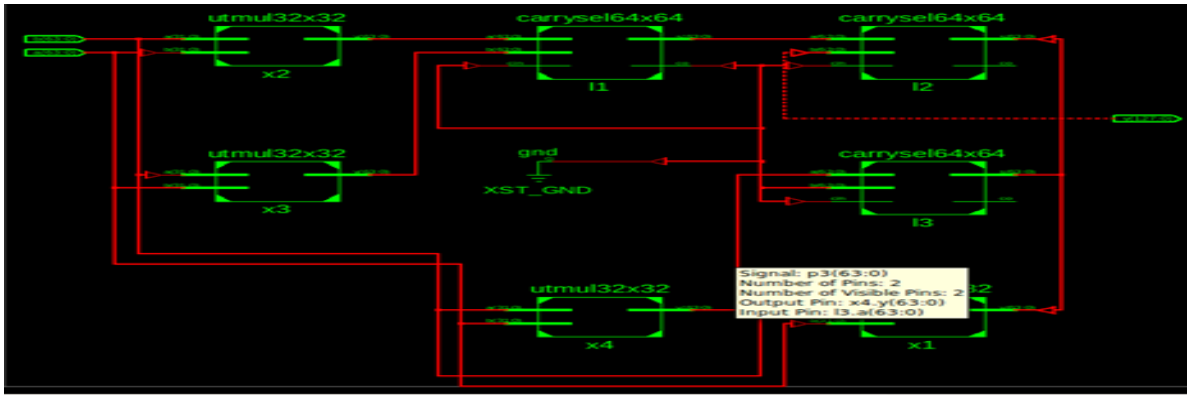

Fig.10. RTL Schematic view of 64-Bit Vedic Multiplier using Carry Select Adder

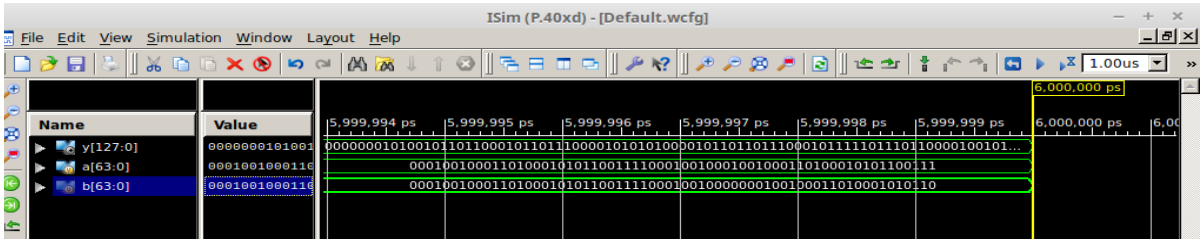

Fig.11. 64- Simulation results for 64-Bit Vedic Multiplier using Carry Select Adder

A comparison of different multipliers using different adders in terms of cells, LUT, Power and speed is shown in Table1.

Table 1. Comparison of Urdhva-Tiryagbhyam and Karatsuba Multiplier

Logic Utilization	Urdhva-Tiryagbhyam			Karatsuba [8]
	Carry Skip Adders	Carry Lookahead Adders	Carry Select Adders	
Cells	15618	19198	15798	-
LUT's	13112	9838	8651	2928
Bonded IOB's (Available)	97	102	102	600
Leakage Power (nW)	3208.7	3715.5	2245.7	-
Dynamic Power (nW)	6568802.9	7357498.5	5999772.3	-
Total Power (mW)	6.572011	7.361214	6.00026	-
Delay (ns)	6.575	6.968	12.84	28.2

IV Conclusion

This paper illustrates the implementation of the Vedic multiplier using carry skip, carry look ahead and carry select adders. It can be concluded that the Vedic multiplier with the carry skip adders is the most efficient in terms of delay and power consumption as can be seen from the results. The multiplier using the carry select adder uses a total power of 6.002 mW while the same multiplier using the carry skip and carry look ahead adders consume a power of about 6.57 mW and 7.36 mW respectively. Carry look ahead adders provides the worst performance in terms of power amongst the three adders. It can also be seen that the Vedic multiplier is better than implementation of the Karatsuba multiplier algorithm.

References

[1] M. Akila, C. Gowribala, and S. Maflin Shaby, "Implementation of High Speed Vedic Multiplier using Modified Adders" in International Conference on Communication and Signal Processing, April 6-8, 2016, India.

[2] Paras Gulati, Harsh Yadav and Manoj Kumar Talejaean, "Implementation of an efficient multiplier using the vedic algorithm" in International Conference on Computing, Communication and Automation, ICCCA2016.

[3] Bhawna Kalra and J.B.Sharma, "Vedic Multiplication based efficient OFDM FFT processor" in Journal of Intelligent and Fuzzy Systems 32(2017) 3121-3128.

[4] Richa Sharma, Manjit Kaur and Gurmohan Singh, "Design and FPGA Implementation of Optimised 32 bit Vedic Multiplier and Square Architectures," in International Conference on Industrial Instrumentation and Control.

[5] Sheetal N. Gadakh and Amitkumar Khade, "Design and optimisation of 16 x 16 bit multiplier using Vedic mathematics," in International Conference on Automatic Control and Dynamic Optimisation Techniques,2016.

[6] Gowthami. P and R.V.S. Satyanarayana, "Design of an efficient multiplier using Vedic mathematics and reversible logic," in 2016 IEEE International Conference on Computational Inteligence and Computing Research.

[7] Amish Jais and Prasanna Palsodkar, "Design and Implementation of 64 bit vedic multiplier using Vedic algorithm," in International Conference on Communication and Signal Processing , April 6-8,2016.

[8] Ravi Kishore Kodali, Lakshmi Boppana and Sai Sourabh Yenamachintala, "FPGA Implementation of Vedic Floating Point Multiplier", in Signal Processing Informatics Communication and Energy Systems (SPICES), 2015 IEEE International Conference on, 19-21 Feb 2015, pp. 1–4.

[9] Honey Durga Tiwari, Ganzorig Gankhuyag, Chan Mo Kim and Yong Beom Cho, "Multiplier design based on ancient Indian Vedic Mathematics" IEEE International SOC Design Conference, 2008, vol. 2,pp. 65-68.

[10] Jagadguru Swami Sri Bharati Krsna Tirthaji Maharaja, Vedic Mathematics. Delhi: Motilal Banarsidass Publishers, 1965.