Scientific Research Publishing

Circuits and Systems, 2016, 7, 3723-3733
http://www.scirp.org/journal/cs
ISSN Online: 2153-1293
ISSN Print: 2153-1285

High-Performance FIR Filter Implementation Using Anurupye Vedic Multiplier

S. Jayakumar[1], Dr. A. Sumathi[2]

[1]Department of EIE, Adhiyamaan College of Engineering, Hosur, India
[2]Department of ECE, Adhiyamaan College of Engineering, Hosur, India
Email: jayakmr1982@gmail.com, sumathi_2005@rediffmail.com

How to cite this paper: Jayakumar, S. and Sumathi, Dr.A. (2016) High-Performance FIR Filter Implementation Using Anurupye Vedic Multiplier. *Circuits and Systems*, **7**, 3723-3733.
http://dx.doi.org/10.4236/cs.2016.711312

Received: May 8, 2016
Accepted: May 19, 2016
Published: September 16, 2016

Abstract

In this, today's world immeasurable analysis goes within the field of communication and signal processing applications. The FIR filter is mostly employed in filtering applications to enhance the quality of the signal. In any processor, the performance of the system is based on the speed of the multiplier unit involved in its operation. Since multiplier forms the indispensable building blocks of the FIR filter system. Its performance has contributed in determining the execution of the FIR filter system. Also, due to the tremendous development in the technology, many approaches such as an array, Vedic methods are made to speed up the multiplier computations. The problem in speed-up operation and resource utilization of hardware with all the conventional methods due to the critical path found in partial products has to be optimized using proposed method. This paper presents the implementation and execution of a FIR Filter design using Anurupye multiplier. Here the FIR filter is examined by using various multiplier algorithms such as Anurupye, Urdhava Tiryagbhyam, and array multipliers. The FIR filter is simulated for analyzing delay; area and power are meted out and lessened by utilizing proposed Anurupye multiplier. The FIR filter design utilizing proposed multiplier offers delay around 18.99 and only 4% of LUT slice utilization compared to existing methods. This architecture is coded in VHDL, simulated using the ModelSim and synthesized with Xilinx.

Keywords

Finite Impulse Response (FIR) Filter, Urdhava Triyagbhyam, Anurupye Vedic Multiplier, Very High-Speed Hardware Description Language (VHDL)

1. Introduction

In overall DSP applications [1], FIR filtering process uses convolution and correlation

for performing its operation. Filtering is the process of adopting appropriate information from a signal. This is obtained by performing weighted summations in the given input signals used for filtering, digital video, and audio signals during transmission in the filter. The signals having useful information were stored consisting of only their required information after eliminating the unwanted signal termed as noise [2]. In order to condition the signal, a number of mathematical processes are carried out on a sampled discrete-time signal for any modifications. Considering the convolution strategy is used to compute the response of the filter for the given signal $k(n)$. Some methods for multiplication use adders for its operations. In [3] this work two architectures Sequential and parallel micro program FIR filters using Wallace tree and Vedic multipliers are used, for evaluating their performance based on the results obtained on ASIC implementation to analyze delay and critical path. This work shows Wallace tree multiplier delivers better performance for FIR architectures.

[4] has proposed the high-performance and low-power implementation for FIR filter for allocating FIR filter with programmable coefficients. These virtualized resources are based on the Computation sharing. This work targets the reduction of redundant computations to achieve high-performance Filtering operations. The computation sharing approach is effective to reduce surplus component involved for filtering by recognizing the familiar computations. [5] proposed FIR filter implementation with resource allocation algorithm for the higher performance and comparable power consumption with pre-emptive tasks with FIR filters based on the carry-save-array multiplier (CSAM) and Wallace tree multiplier (WTM), also used in DSP to speed up the operations and adaptive filter applications. [6] has developed a FIR filter using Urdhava-Tiryagbhyam algorithm to improve the performance of the system. He formulated the problem to reduce computation time better than the inbuilt MATLAB function with improved speed than the other methods. Among the various algorithms, linear convolution is the basic methods used in the FIR system.

In this paper, we present a high-speed efficient multiplier for the FIR Filter on ancient Vedic mathematic formulae. So we process with Anurupye Vedic multiplier methods for computing discrete linear convolution. The proposed Anyrupye Vedic algorithm is analyzed with other conventional algorithms in terms of power, delay, and the area is found to be much efficient. Anurupye denotes "proportionality" consisting of both working and theoretical index (base) very suitable for values far away from the index which was not possible in other techniques. These calculated outcomes are considered for power, delay, and area using a Vedic algorithm such as Anurupye and Urdhava Triyakbhayam along with array multiplier. The Vedic algorithm Anurupye is found to be fast and effective. This method is implemented in FIR design methodology for enhancing filtering functions in eliminating unwanted noise. Thereby the system efficiency with fewer resources and lesser computation is improved.

2. Vedic Multiplier

The former practices were recalled from Indian Sanskrit works named as the Vedas,

concerning 1911 in addition to 1918 by Sri Bharati Krishna Tirthaji and from (1884-1960) the Atharva Vedas. As indicated by his investigation, the greater part of the arithmetic is found with sixteen sutras, or word-formula [7] [8]. These formulae portray the ways in which mind sensibly works and are in this manner a remarkable assistance in directing the pupil to the correct tactics for results. In [9] the Vedic scheme, challenging complications or huge sums can regularly be solved instantly by this method. Many significant and striking techniques are just a part of an entire scheme of mathematics which is a future new regular than the modern method. Vedic Mathematics exhibits the vibrant and integrated collection of mathematics and these methods are complementary, direct and simple [7]. A wakefulness of Vedic mathematics can be meritoriously improved if it is included in engineering learning. In future, all the foremost academies may set-up suitable exploration centers to encourage research works in Vedic mathematics. For his research, all of the mathematics is created on sixteen Sutras, or word-formulas [10]. These formulae define the way the intellect absolutely works and are subsequently an unlimited help in guiding the student to the suitable method of solution. In this Vedic scheme, challenging problems or huge sums can often be resolved immediately by the Vedic method. These methods are prominent and more beautiful to be a part of a complete system of mathematics which is more simplified than existing methods. It endorses the clear and unified composition of computations and this attempt is paired, conservative and easy [11]. The cognizance of Vedic mathematics can be incorporated in engineering to explore and inspire research associations.

2.1. Urdhva Triyakbhyam

This method is a technique in Vedic mathematics to increase the speed and area parameters of a multiplier utilized. Hence this algorithm is used to produce a partial product with its summation assessed concurrently to produce a net result. This process is the greatest asset of this method. The major advantage of this multiplier with other multipliers is its uniformity of computation. This flexible nature leads to the layout design with a simple and realistic approach employed to several forms of multiplications. In a Sanskrit the term Urdhva implies vertical up-down, Tiryakbhyam implies left to right or the other way around [6]. This method is a general technique and can be applicable to any instances of multiplication steps. This sutra is successful and associated with the numbers which are closer to the index like 10, 100, 1000 *i.e.*, the numbers to the power of 10 [12] (e.g. 96×98 or 102×104). This depends on an alternate and the simple thought for generation of a wide range of partial results created simultaneously.

2.2. Anurupye Algorithm

Today we are discussing a genuine significant sutra [13]. Since this sutra permits us to multiply the numerical values that are adjacent to an index of 10 but not close to the index of 10. This makes more suitable in contrast to Nikhilam Multiplication as it permits us to elude adding/subtracting of only vast numbers closer to the index. The multiplication is a process accomplished by finding out the complement of a large number

of its closest index reducing the number of operations required. In this case, the sub-multiples are considered as the working index. Consequently, complication is reduced for larger single number multiplication. Anurupye Sutra accomplishes compliment by subtracting off a two numbers from its closest power index, *i.e.* 10, 20, 30, etc. Hence size, of the multipliers with size 8-bit each is reduced by finding the difference of the reference number from the nearest index. The Anurupye methods for multiplying decimal numbers 32 × 36 are explained [7] with effective, simplified steps to produce both LHS and RHS of the result as a shown in **Figure 1**.

Since both the numbers are far from 10 we consider 30 as our working index.

STEP 1: Register the compliments from the working index adjacent to the given number.

$$10 \times 3 = 30$$

$$32 \quad +2$$

$$36 \quad +6$$

STEP 2: Right Hand Side (RHS)

$$10 \times 3 = 30$$

$$32 \quad +2$$

$$36 \quad +6$$

$$38 \quad /+12$$

Cross adds diagonally to get 38. Multiply with its compliments (+6) × (+2) = 12 to get R.H.S.

Since we have 10 as our working index only one digit (2) is considered. Another digit (1) is propagated as carry to L.S.B.

STEP 3: Left Hand Side (LHS)

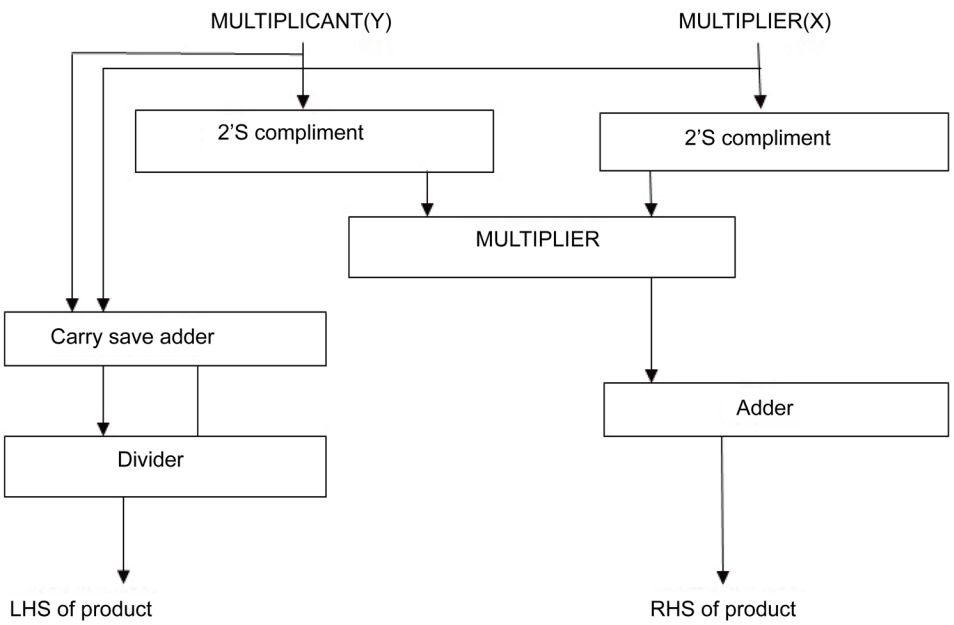

Figure 1. Proposed Anurupye algorithm.

$$\begin{array}{ll} 10 \times 3 = 30 & \\ 32 & +2 \\ 36 & +6 \\ 115 & /12 \end{array}$$

The L.H.S is obtained by taking a product with its corresponding working index. $38 \times 3 = 114$. Then taking the sum of the L.H.S with carry 114 + 1 (carry) = 115. Thus the answer is obtained from the above calculations as $32 \times 36 =$ **1152**.

3. FIR Implementation

A FIR filter is also known as the non-recursive digital filter. In this filter, no feedback exists and the output depends only on the present and the past input values of the signal [14]. The FIR filter impulse response consists of finite no of zeros of for the duration 0 to N − 1. Thus, for N-order impulse response of a FIR filter exists for N + 1 samples and then decays to zero.

The output equation for the FIR filter [6] is obtained from convolution of two sequences expressed as

$$g(n) = j(n) * k(n) \tag{1}$$

$$g(n) = \sum_{n=0}^{N-1} j(n) k(n-k) \tag{2}$$

where, $k(n)$ is the input, $g(n)$ is the output of the given signal to the FIR filter respectively, $j(n)$ is the impulse response (*i.e.*) the filter coefficients. Linear phase FIR filters are used in the application such as audio and video where exact linear phase response is required [15].

4. Proposed Method

In this approach, we design a FIR filter by implementing a proposed Anurupye multiplier algorithm. The conventional direct form structure with 8-Tap FIR filter is shown in the **Figure 2** use 8 numbers of multipliers, 7 numbers of additions for a single sample. The conventional FIR design uses delay elements, which adds delay to the signal by certain value by multiplying the values of the previous steps with the corresponding coefficient. The same filter can be reduced by using Anurupye multiplier for calculating product between inputs with the coefficient by replacing normal multiplication as shown in **Figure 3**. The main advantage of this proposed method is the usage of fewer resources utilized to accomplish the same task by reducing area, delay, and power effectively.

4.1. Simulation Results

The design of the Anurupye Vedic multiplier has been analyzed and this algorithm is utilized in FIR Filter for prevailing low power and high performance. This proposed method is coded in Hardware description language such as VHDL, simulated using Model Sim-Altera 6.3 g and the net synthesize is obtained in the Xilinx ISE14.4 Software.

4.2. Simulation Results of Multiplier Topologies

The Design of 8 × 8 multiplier topologies for various multipliers such as Array multiplier, Vedic multipliers such as Urdhava Triyagbhyam and Anurupye algorithm are shown in Table 1. All the FPGA devices for targeting are same for all cases of multipliers to have uniformity in constraints for easier synthesis.

The analysis is examined for different multiplier unit. From the comparison, the memory usage of Anurupye is 191 mb small compared to other multiplier architecture. Anurupye Vedic multiplier entertains the greatest improvements compared to other

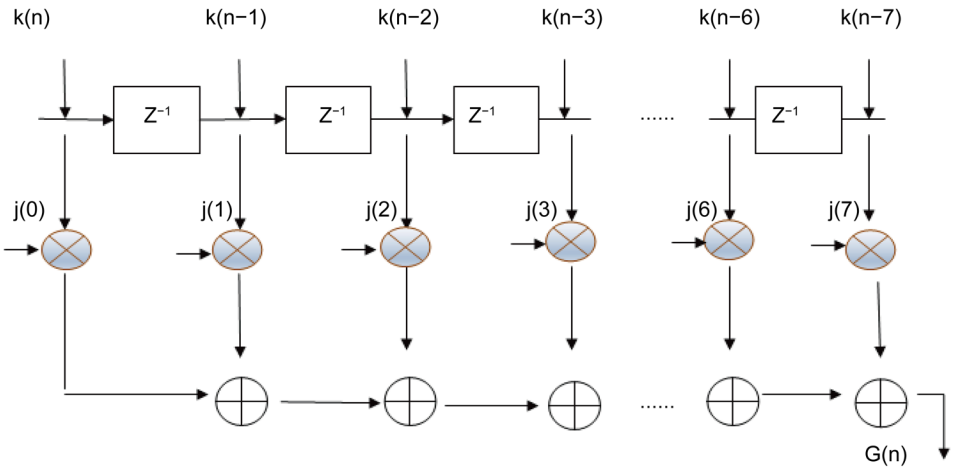

Figure 2. Conventional direct form structure of FIR filter.

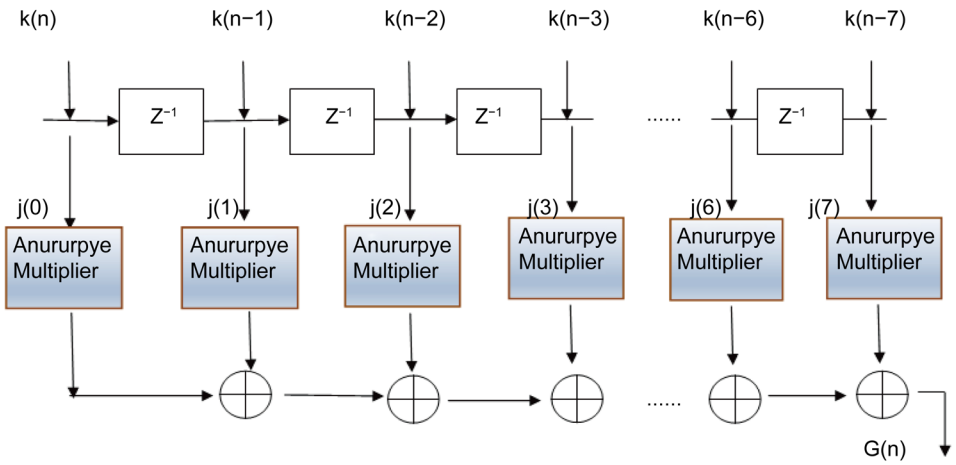

Figure 3. Proposed direct form structure of FIR Filter.

Table 1. Comparison of multiplier topologies.

Multipliers		Memory	Power	Delay
Vedic	Array	196 mb	56 mW	22.05 ns
	Urdhava Triyagbhyam	194 mb	52 mW	23 ns
	Anurupye	191 mb	34 mW	20.54 ns

multipliers over combinational path delay and the orderliness of arrangements. The Delay in proposed multiplier for 8 × 8-bit number is 20.54 ns whereas the delays in Urdhava Triyagbhyam and Array are and 22.0 ns and 23 ns respectively. Thus, this multiplier shows the maximum speed among conventional multipliers. This benefits than others provide the choice to prefer the best multiplier. It has less number of gates required for given 8 × 8 bit multiplier, so its power dissipation is 34 mW very lesser and it has less switching activity as associated to array multiplier. Figures 4-6 show the graph for the multiplier comparison based on memory consumption, delay and power consumption.

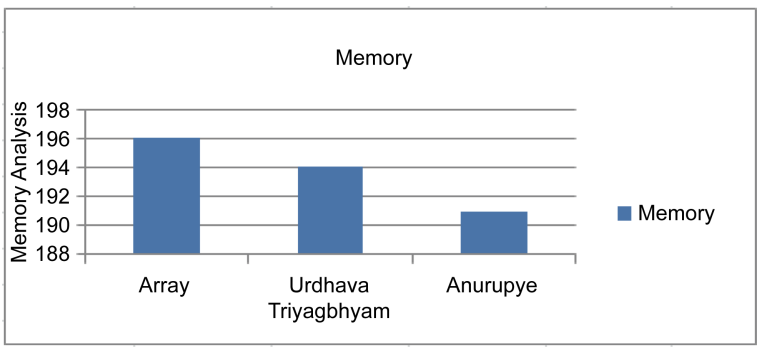

Figure 4. Area comparison.

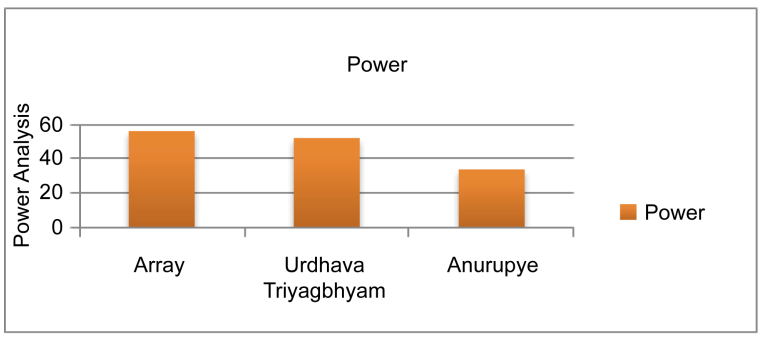

Figure 5. Power comparison.

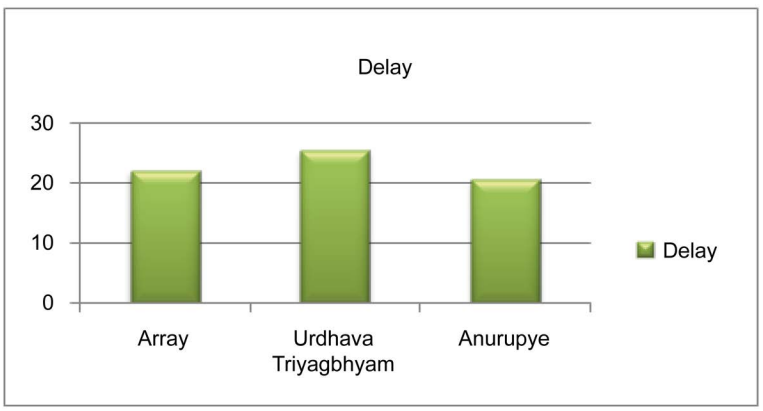

Figure 6. Delay comparison.

4.3. Simulation Results of Proposed FIR Filter Design

Table 2 shows in order to achieve FIR implementation the proposed Anurupye algorithm have been used. This method operates much faster than a conventional multiplier with a delay of 18.99 ns and less power consumption of 14 mW. The delay time is obtained from synthesis as shown in Figure 7. Anurupyena is an approach that permits for a quicker and compact circuit, by substituting the equivalent lowest numbers. Figure 8 shows the memory usage result obtained from synthesis report. Hence the stages such as partial product generation and final accumulation involved in other methods are simplified to get reduced output.

The logical simulation of recommended multiplier architecture is simulated and the result is shown in Figure 9. In Figure 9 shows the simulation result of 8 × 8 bit Anurupye multiplier. Where the signal final x, signal final y are the eight-bit input of the multiplier and the signal final output_1 is the output with sixteen-bits are obtained as output. The system encompasses of 2's complement unit, Divider block, Adder block and multiplier block for obtaining LHS and RHS of the product. In Figure 10, shows the simulation of the Output Response for 8 × 8 bit FIR Filter using Anurupye filter is

Table 2. Comparison of Anurpye multiplier and FIR implementation.

Multiplier	Power (mW)	Delay (ns)	No of slices	Memory (Mb)
Anurupye Multiplier	34	20.54	132	191
FIR Implementation	14	18.99	120	109

```
    MUXCY:S->O              1   0.404   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             1   0.051   0.000  sub_1/Madd_value_adds
    MUXCY:CI->O             0   0.051   0.000  sub_1/Madd_value_adds
    XORCY:CI->O             1   0.699   0.509  sub_1/Madd_value_adds
    LUT2:I0->O              1   0.612   0.357  sub_1/value<15>1 (rhs
    OBUF:I->O                   3.169          rhs_15_OBUF (rhs<15>)
   ----------------------------------------
   Total                      20.546ns (13.443ns logic, 7.103ns rou
                                       (65.4% logic, 34.6% route)

=====================================================================
```

Figure 7. Delay time of proposed Anrupye multiplier.

Peak memory Usage: 191 MB
Total REAL time to MAP completion: 6 secs
Total CPU time to MAP completion: 3 secs

Figure 8. Memory usage of proposed Anrupye multiplier.

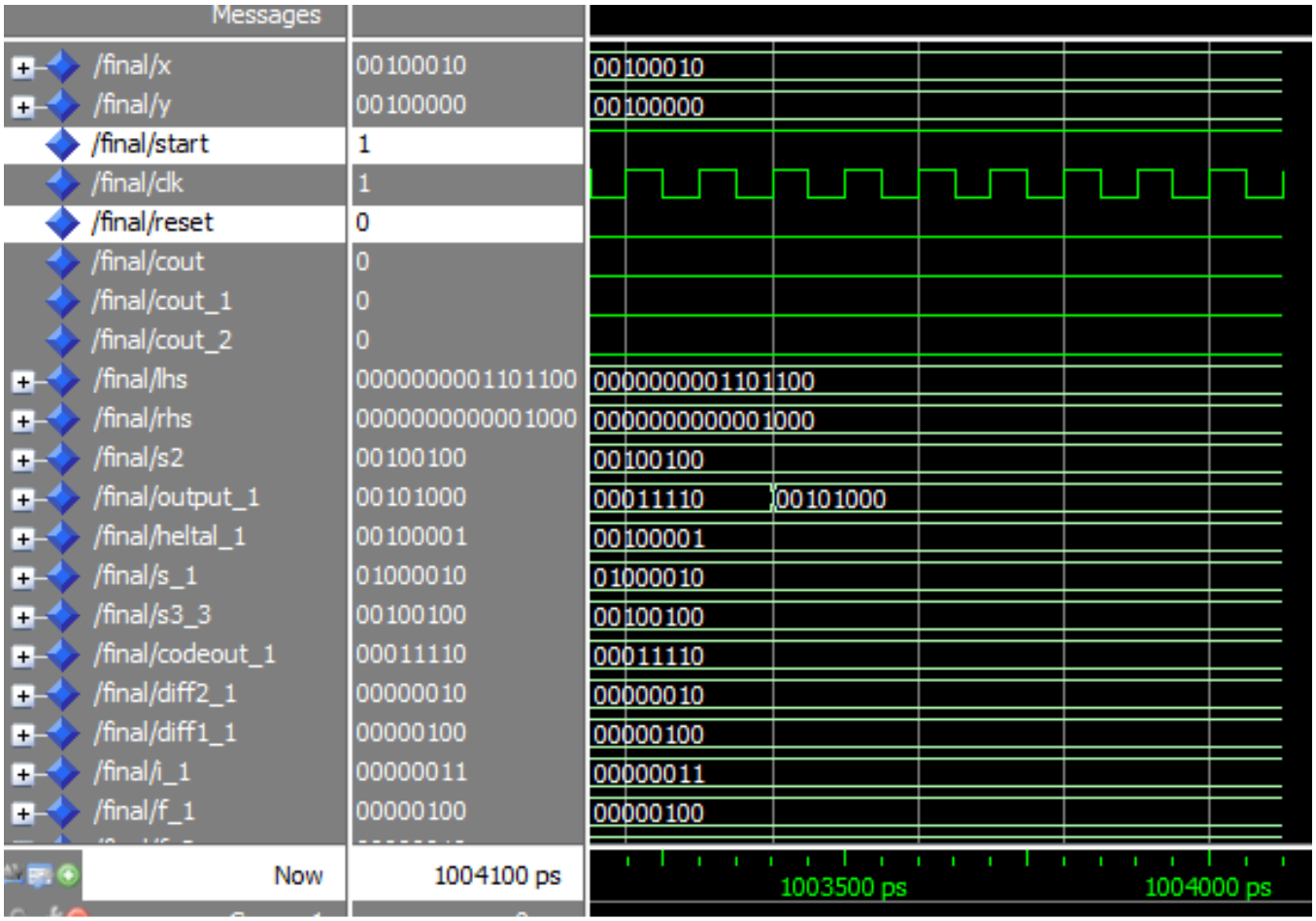

Figure 9. Output response for 8 × 8 bit Anurupye multiplier.

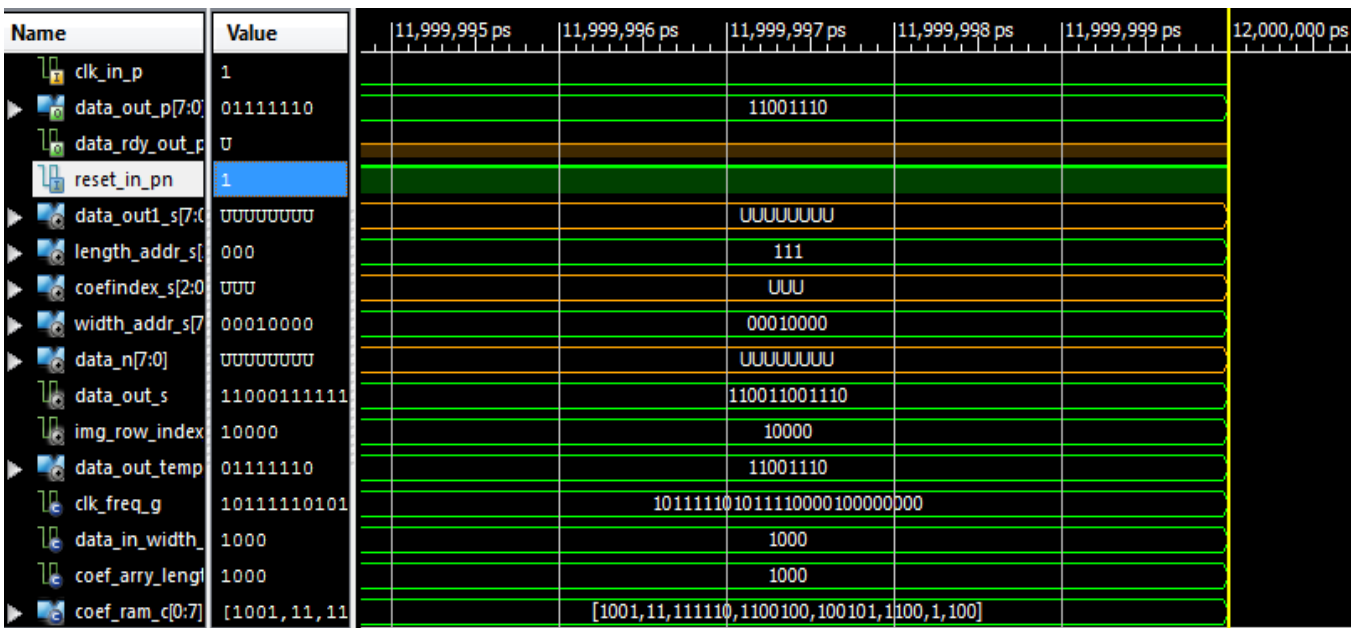

Figure 10. Output response for 8 × 8 bit FIR filter.

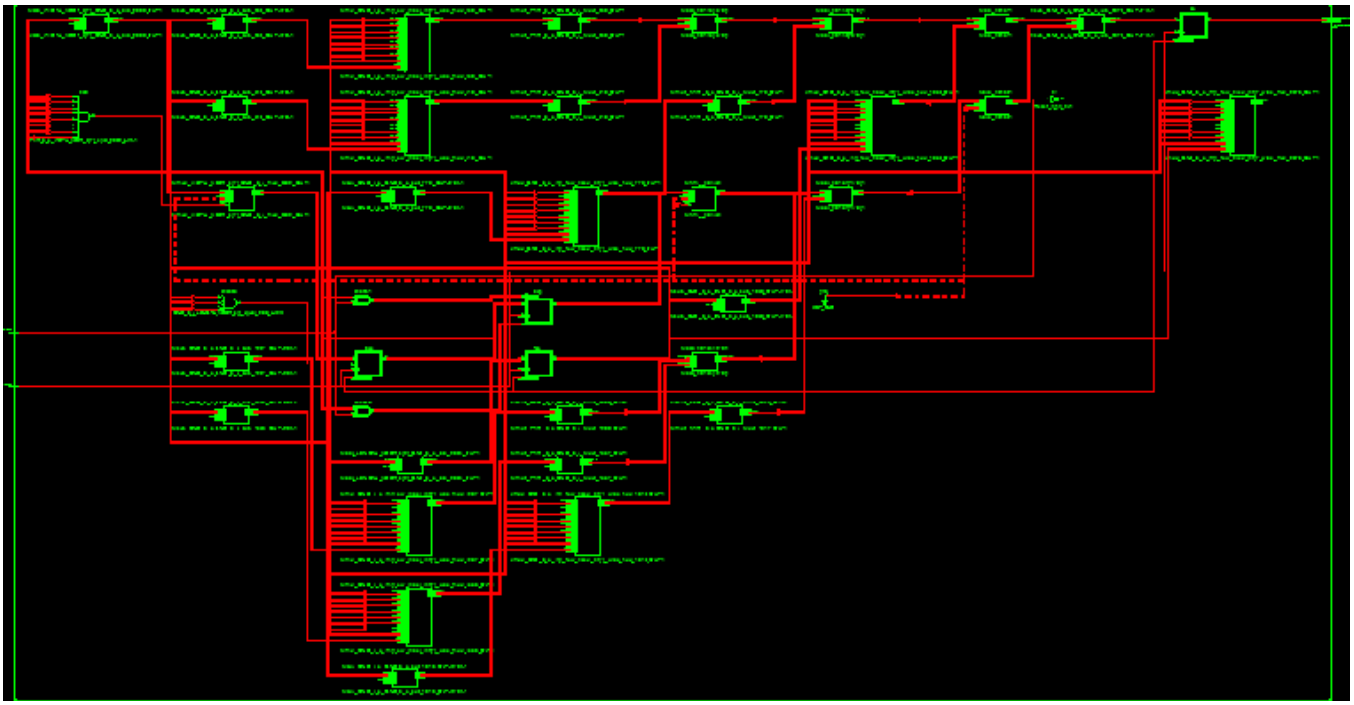

Figure 11. RTL schematic symbol for 8 × 8 bit FIR filter.

implemented with lesser no of slices, memory, power, and delay. The RTL view of the FIR filter is shown in Figure 11. Containing many individual blocks is optimized.

5. Conclusion

In this paper, we have bestowed a unique approach to improve the performance of the FIR Filter considerably by using an Anurupye Vedic Multiplier. This proposed multiplier provides improved performance parameters with less number of gates used for a given 8 × 8 bit multiplier. Also, from the results achieved it can be obviously apparent that the delay of this multiplier is relatively reduced compared to the other common designs of multipliers. It's therefore, decided that the Anurupye Vedic Multiplier based FIR filter design would be a good choice for high-speed DSP applications in the future. Further research can be performed with the other algorithms of Vedic mathematics and to obtain efficient design to be utilized in cryptography network for providing secure data transfer.

References

[1] Itawadiya, A.K., Mahle, R., Patel, V. and Kumar, D. (2013) Design a DSP Operation Using Vedic Mathematics. 2013 *International Conference on Communications and Signal Processing* (*ICCSP*), Melmaruvathur, 3-5 April 2013, 897-902. http://dx.doi.org/10.1109/iccsp.2013.6577186

[2] Proakis, J.G. and Manolakis, D.K. (1996) Digital Signal Processing. Prentice Hall Inc., Upper Saddle River, New Jersey, 82.

[3] Abhilash, R., Dubey, S. and Chinnaaiah, M.C. (2015) High Performance and Area Efficient

Signed Baugh-Wooley Multiplier with Wallace Tree Using Compressors. *International Conference on Electrical, Electronics, Signals, Communication and Optimization* (*EESCO*), Visakhapatnam, 24-25 January 2015, 1-4. http://dx.doi.org/10.1109/eesco.2015.7253847

[4] Park, J., Jeong, W., Choo, H., Mahmoodi-Meimand, H., Wang, Y. and Roy, K. (2002) High Performance and Low Power FIR Filter Design Based on Sharing Multiplication. *Proceedings of the* 2002 *International Symposium on Low Power Electronics and Design*, 14 August 2002, 295-300. http://dx.doi.org/10.1145/566408.566485

[5] Park, J., Muhammad, K. and Roy, K. (2003) High-Performance FIR Filter Design Based on Sharing Multiplier. *IEEE Transactions on Very Large Scale Integration VLSI System*, **11**, 244-253. http://dx.doi.org/10.1109/TVLSI.2002.800529

[6] Chaturvedi, S. and Roy, V. (2014) FIR Filter Design Using Vedic Urdhava-Tiryagbhyam Technique. *International Journal of Engineering Trends and Technology*, **8**, 229-304. http://dx.doi.org/10.14445/22315381/IJETT-V8P255

[7] Singhal, V. (2014) Vedic Mathematics for All Ages: A Beginners Guide 16 Sutras for Mental Calculations Easily Explained Formulae with Practice Exercises. Second Reprint Edition, Motilal Banarsidass Publishers, New Delhi.

[8] Dharmannavar, K.H. and Dharmambal, Mrs. (2015) The Application of Vedic Mathematics for High Speed Multiplier in Fir Filter Design. *International Journal of Engineering Research and General Science*, **3**, 364-379.

[9] Koushighan, S., Hariharan, K. and Vaithiyanathan, V. (2014) Design of an Optimized High Speed Multiplier Using Vedic Mathematics. *Contemporary Engineering Sciences*, **7**, 443-448.

[10] Vaithiyanathan, G., Venkatesan, K., Sivaramakrishnan, S., Siva, S. and Jayakumar, S. (2013) Simulation and Implementation of Vedic Multiplier Using VHDL Coding. *International Journal of Scientific Engineering Research*, **4**, 1-5.

[11] Bhongade, R.K., Mungale, S.G. and Bogawar, K. (2014) VHDL Implementation and Comparison of Complex .Multiplier Using Booth's and Vedic Algorithm. *Compusoft*, **3**, 599-603.

[12] Saokar, S.S., Banakar, R.M. and Siddamal, S. (2012) High Speed Signed Multiplier for Digital Signal Processing Applications. *IEEE International Conference on* Signal Processing, *Computing and Control* (*ISPCC*), Waknaghat Solan, 15-17 March 2012, 1-6.

[13] Sujatha, S. and Krishnammal, V.P. (2016) Performance Analysis of Anurupye Vedic Multiplier in FFT Processor. *Australian Journal of Basic and Applied Sciences,* **10**, 579-585.

[14] Mohanty, B.K. and Meher, P.K. (2016) A High-Performance FIR Filter Architecture for Fixed and Reconfigurable Applications. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, **24**, 444-452. http://dx.doi.org/10.1109/TVLSI.2015.2412556

[15] Savadi, A., Yanamshetti, R. and Biradar, S. (2016) Design and Implementation of 64 Bit IIR Filters Using Vedic Multipliers. *International Conference on Computational Modeling and Security*, **85**, 790-797. http://dx.doi.org/10.1016/j.procs.2016.05.267