Design of High Performance IIR Filter Using Vedic Multiplication Method

Y. A. Durrani
Electronics Engineering Department, UET, Taxila, Pakistan
[1]yaseer.durrani@uettaxila.edu.pk

***Abstract*-**Digital filters can be found in most of the electronic circuits. One major application of these filters involve an input signals that are sampled and converted to the real-time signal to the modern computers. This paper describes the high performance infinite impulse response (IIR) filter using Vedic multiplication method due its low round-ff noise and low quantization errors. It reduces the computational delay and hardware to minimize the number of iterations and enhances the performance of the DSP processor. Vedic method enables the parallel processing stages of intermediate products and removes the unnecessary multiplication steps with zeros. The filter is designed on Xilinx software and different frequency responses are evaluated in Matlab at different filter orders.

***Keywords*-**Digital Filter, Vedic Multiplication, Frequency Responses, Cascaded Structure

I. INTRODUCTION

The design of digital filter is entirely different from analog filters. Digital filters process signals in time domain and if a specific frequency domain response is needed, it is highly desirable to convert it into the equivalent time domain response. Normally, the digital signals are in parallel form and these are more binary-coded version of analog signals. These filters perform mathematical operations on a sampled, discrete-time signal to enhance or reduce signal. It is simply a discrete-time, discrete-amplitude convolver [i].

A large percentage of digital filters are implemented in finite and infinite impulse response (FIR) and (IIR) filters. The FIR filters are used over a wide range of sample rates in electronic design automation (EDA) tools, and DSP-based systems. The IIR filters normally have low-order than FIR filters with same performance due to low sample rates (less than 200KHz), sharpness of cutoff, and pass-band ripples etc. Due to low-order, IIR filters tend to require efficient digital multipliers with less computation and less delay elements. The IIR filter, known as a recursive filter, uses feedback to compute outputs. These are very effective in several applications such as low-power communication transceivers, routers, etc. Furthermore, they require less number of parameters in designing, lower memory requirements and lower computational cost. IIR filter do not provide linear phase within its pass-band and more feasible to designers [ii].

High performance is most critical issue in very-large-scale integrated (VLSI) technology. For many DSP applications the hardware involves in actively multi-tasking computations and it increases large amount of delay. On the other hand, the sophisticated and portable wireless devices are very power conscious. We can attenuate these dependences by tuning complexity according to the given criteria and altering the behavior of the signal [iii]. A higher order sequential IIR filter can be realized by cascading the multiple stages of 2nd order filter as they known as biquads. Each biquad section can be turned off when they are not needed in order to improve the performance. Every biquad is designed so that it can satisfy the overall required specifications with stability, and amplitude response [iv].

The bi-linear transformation technique is the earlier technique that has been developed [v]. This technique adopts the digital filter with the transformation of the corresponding analog filter, and then known filter design methods, such as Chebyshev type I and II, or Butterworth are used to develop the design of related filter. The accomplished analog filter is again inter-transformed back to digital filter by using the bi-linear transformation. However, this method requires lot of related information and in many cases demonstrates the poor results [vi]. Recently more accurate techniques have been introduced with higher performance with lesser knowledge of the filter [vii]. Apart of this area of research, several other more advanced techniques have been proposed such as impulse invariance approach, matched Z-transformation, fixed-point representation, and evolutionary algorithm [viii], [ix].

Multiplication can be performed in DSP system through different types of operations such as convolution, fast fourier transform or the filter design. The processor performance can be dependent on the speed of the multipliers. Hence, the speed of the multiplication is crucial component in DSP system. The commonly used application in any DSP system is the design of the digital filter. The IIR filter is also known as convolution filter due to the sequence of time domain multiplication is same as the convolution is

implemented in frequency domain [x]. The focus of this paper is the design of high speed IIR filter by using Vedic multiplier method. These types of the multipliers increase the performance of the processors by reducing the number of iterations. Recently few methods [xi], [xii] have been proposed using Vedic multiplication structure. Their work with the reduction of number of bits in multiplication process gives reasonable performance but more average execution time as compared with Matlab built-in function.

The paper is organized as follows. In Section II, the detailed background of digital IIR filter is discussed. The proposed architecture of IIR filter is described in Section III. The experimental results are demonstrated IV. Finally, the Section V summarizes the work.

II. IIR Filter Background

The IIR filter is a linear-time-invariant discrete time system characterized by difference equation in (1):

$$y[n] = \sum_{k=0}^{\infty} h(k)x(n-k) = \sum_{k=0}^{N} b[k]\,x(n-k) - \sum_{k=1}^{M} a[k]y(n-k) \quad (1)$$

where $h[k]$ is the impulse response of the filter with infinite in duration, $a[k]$ and $b[k]$ are coefficients of the filter, $x(n)$ and $y(n)$ are the input/output (I/O) of the filter respectively. The transfer function of IIR filter can be expressed in z-domain of (1) is given in difference equation in (2):

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + \cdots + a_M z^{-M}}$$

$$H(z) = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 + \sum_{k=1}^{M} a_k z^{-k}} \quad (2)$$

Using equation (2), the IIR digital filter realization needs feedback as shown in Fig. 1. The n_{th} order filter transfer function of II filter can be characterized as $2n + 1$ coefficients. Generally it requires $2n$ two adders and $2n + 1$ multipliers for the hardware implementation.

Different type of realization structures can be obtained by difference equation of the filter named as direct-form realizations. The most common realizations are: direct form-I, direct form-II and transposed direct form-II. These all forms are functionally equivalent but there is a difference in their architecture, such as [vii]:

- *Direct form-I* has $n + m + 1$ multipliers, $n + m$ adders and $n + m$ memory locations (or delay components). The coefficients of the multiplier are accurately the coefficient of the transfer function.
- *Direct form-II* architectures uses $n + m$ adders, $n + m + 1$ multipliers, and maximum of $\{n, m\}$ locations of memory. Such architectures use minimum locations of memory than the direct form-I architecture, it known as canonic.
- *Transposed direct form-II* realization architecture also uses same amount of adders, multipliers, and locations of the memory as direct form-II structure. It uses less number of delay elements in its design compared to direct form-I.

III. Proposed Architecture of IIR Filter

In this section, a *cascaded form* of digital IIR filter design methodology is proposed. The recursive system of filter architecture exhibits high performance, low computational complexity and minimum memory requirements. As the filter order increases, the complexity of the hardware implementation is increased and the range of the precision decreases. In this paper, for better attenuation the 4th order IIR filter is implemented by cascading two stages of 2nd order IIR filter or biquads (two poles and two zeros). The 2nd order architecture is not very sensitive to the quantization of coefficient and several bits are not required to identify the values of the coefficients. The IIR filter uses more than one cascaded biquad architecture. For higher-order filters, many biquads are cascaded such as:

$$H(z) = \prod_{i=1}^{n} \left(\frac{b_{i0} + b_{i1} Z^{-1} + b_{i2} Z^{-2}}{1 + a_{i0} Z^{-1} + a_{i1} Z^{-2}} \right) \quad (3)$$

where n is the number of biquads. The cascaded of biquads can be re-ordered to minimize the round off noise. However, the n biquads gives n possible options.

Analyzing fig. 1, each biquad structure is comprised of two feedback paths (a_1 and a_2) and two feed-forward paths (b_1 and b_2). The operation of the filter is summarized by the set of recursive equations in (4) and (5):

$$p(n) = x(n) - a_1.p(n-1) - a_2.p(n-2) \quad (4)$$

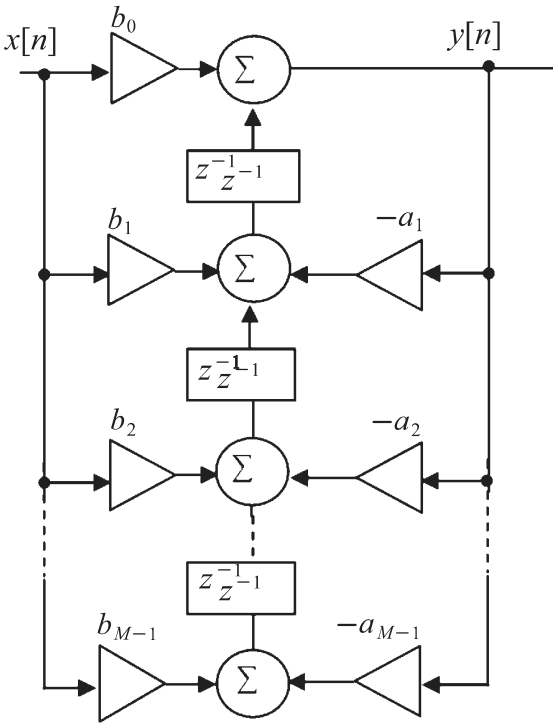


Fig. 1. Block diagram of transport-direct form II IIR filter

$$y(n) = b_0.p(n) + b_1.p(n-1) + b_2.p(n-2) \quad (5)$$

Analyzing (4) and (5), to implement a biquad filter, only four addition and five multiplication blocks are required. In the design of the IIR filter, the traditional array or Booth multiplier is not used, instead Vedic multiplier is implemented due to the fast switching ability and low-computational complexity. The transposed-direct form II structure is used for the reduction of round-off errors. This type of error is due to the coefficient quantization and truncation of bits. Every biquad section implements a 2nd order filter by using the equation of difference in (6) using (7) and (8):

$$y[n] = b_0 * x[n] + p_1 \quad (6)$$
$$p_1 = b_1 * x[n] + a_1 * y[n] + p_2 \quad (7)$$
$$p_2 = b_2 * x[n] + a_2 * y[n] \quad (8)$$

where p_1 and p_2 are the two state value *s*, b_0, b_1, and b_2. The coefficients multiply the input signal $x[n]$ as referred to the feed forward coefficients. The a_1 and a_2 coefficients multiply the output signal $y[n]$ is the feedback coefficients. The output not only depends on the present input but also on the previous values of outputs i.e., it has a recursive structure. The use of transpose-direct realization structure reduces the necessary number of adders and delay lines.

A. Vedic Multiplication

High-speed parallel multiplication is very significant in digital signal processing (DSP) system. Vedic multiplication method is used in many DSP applications such as digital IIR filter, FET, convolution and Fourier transform etc. The Vedic multiplication method [xiii] enables to reduce the traditional processing steps. It uses 16 formulae and 16 sub-formulae, which deals with the several basic and complex mathematical computational operations. It performs multiplication through crosswise and vertically steps. The numeric digits on the two different ends of the each line are multiplied and their result can be added with previous value of the carry. If there are more lines in one single step, all results can be added to the previous carry value. The least-significant digit is obtained and acts as one of the result that takes the carry for the next step. Initially the carry value is considered as zero.

The Vedic multiplication process is performed through the generations of all possible partial products with the concurrent addition of these partial products. Parallel partial products are calculated with their sum. Let the product of N x N-bit binary numbers P and Q where $P = p_1 p_2 p_3 ... p_n$ and $Q = q_1 q_2 q_3 ... q_n$. The multiplication process may be viewed by two steps: The evaluation of partial products and the accumulation of shifted partial products can be described in the following steps:

Step 1: Divide the multiplicand P and multiplier Q into two equal lines, each consists of $[1 + n \ to \ (n/2)]$ bits and $[n \ to \ 1]$ bits, where first part represents the most-significant-bit (MSB) and other indicates least-significant-bit (LSB).

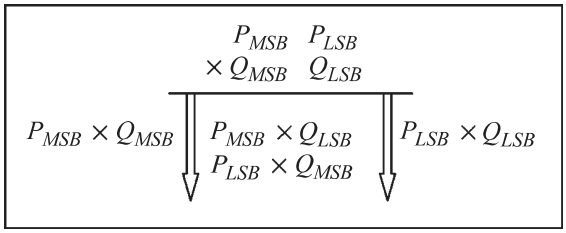
Step 2: Indicates the parts of P *as* P_{MSB} and P_{LSB} and parts of Q *as* Q_{MSB} and Q_{LSB}.

Step 3: For P x Q, the product can be obtained as:

P and Q are divided in P_a, P_b *and* Q_a, Q_b further their a and b part is divided into 4 bits. We can get the resultant product as shown in Fig. 2.

The Vedic multiplication can be performed as:

$$Q_b \ Q_a \text{ x } P_b \ P_a = R_3 R_2 R_1 R_0 \quad (9)$$



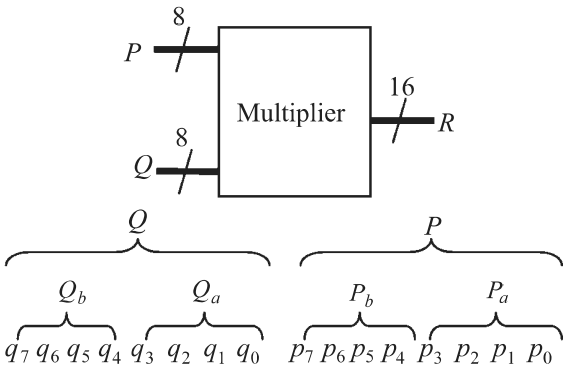


Fig. 2. 8 x 8-bit Vedic multiplication

where

$R_0 = P_a \times Q_a$, $R_1 = (P_b \times Q_a) + (P_a \times Q_b)$, $R_2 = P_b \times Q_b$ (10)

The line diagram of the Vedic product of mapping in binary system is shown in Fig. 3. Each bit is encircled in the Figure and the process of multiplication is demonstrated in [xiv].

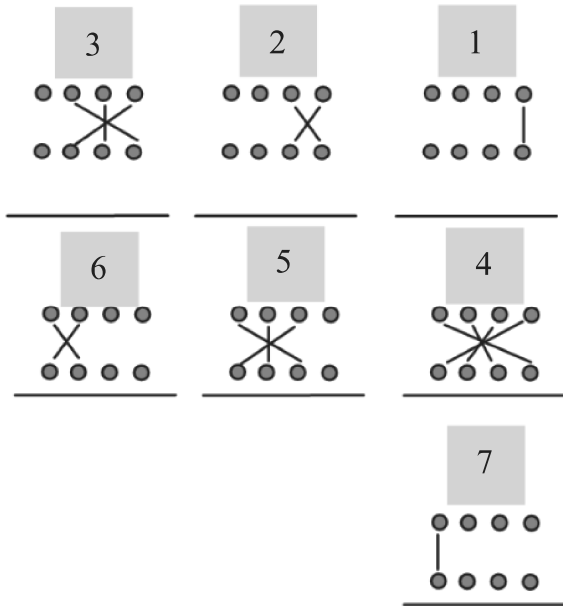


Fig. 3. Line Diagram for the multiplication of two 4 bit numbers [xiii].

B. Hardware Implementation

Binary multiplication consists of following basic operations:

0 x 0 = 0, 1 x 1 = 1, 0 x 1 = 0, 1 x 0 = 0

As hardware, two *n*-bit numbers *p* and *q*, can be expressed as in (11) and (12):

$$p = 2^x \pm \alpha_1 \quad (11)$$

$$q = 2^x \pm \alpha_2 \quad (12)$$

where x is the exponent, a_1 and a_2 of p and q respectively. Assuming the product Z of both numbers is:

$$Z = p \times q = (2^x \pm \alpha_1)(2^x \pm \alpha_2) \quad (13)$$

The bases of multiplier and multiplicand can be assumed same, thus (14) can be expressed as:

$$Z = pq = 2^x(p \pm \alpha_2) \pm \alpha_1\alpha_2 \quad (14)$$

The large multiplication can be decomposed into a small multiplication, shifting addition, and subtraction. Hence, the power, delay, and cost factors can be improved by the reduction of hardware. Similarly, if we multiply two bits *p* and *q*, then logical AND operation produces the same result as shown in Fig. 4.

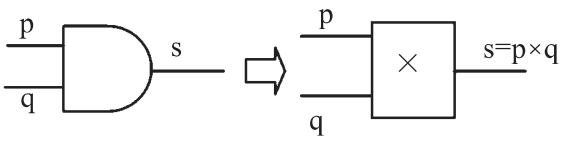


Fig. 4. Bit-level multiplier

The logic synthesis is performed at register transfer level (RTL). Initially 2x2-bit Vedic multiplication is performed with four input AND gates and half-adders (HA) as given in Fig. 5-(a). Similarly, the 4x4-bit Vedic multiplication uses same 2x2-bit multiplier as the basic building module and the 8x8-bit multiplication uses 4x4-bit module and similarly for *n* x *n*-bit multiplication. The 8-bit multiplicand *P* splits into two 4-bit sub-multiplicands P_a and P_b. Similarly for 8-bit multiplier *Q* also splits into two 4-bit sub-multiplier Q_a and Q_b. The results of this product would be more than 4-bits $R_3R_2R_1R_0$ are given in (10). To implement 8x8 multiplications, we need four 2x2-bit Vedic multipliers and three 4-bit full adders. Multiplication minimum hardware architecture and low-computational complexity is shown in figure 6. Here the partial product and addition is performed concurrently. The ripple carry adder (RCA) consist two parts:

- Sum and carry selector
- Sum and carry generator

The second part consumes the most of the logic sources of RCA and creates the critical path. The RCA design in the multiplier is with the minimum hardware. It effectively eliminates all redundant operations and the logical sequences are performed according to the data dependence.

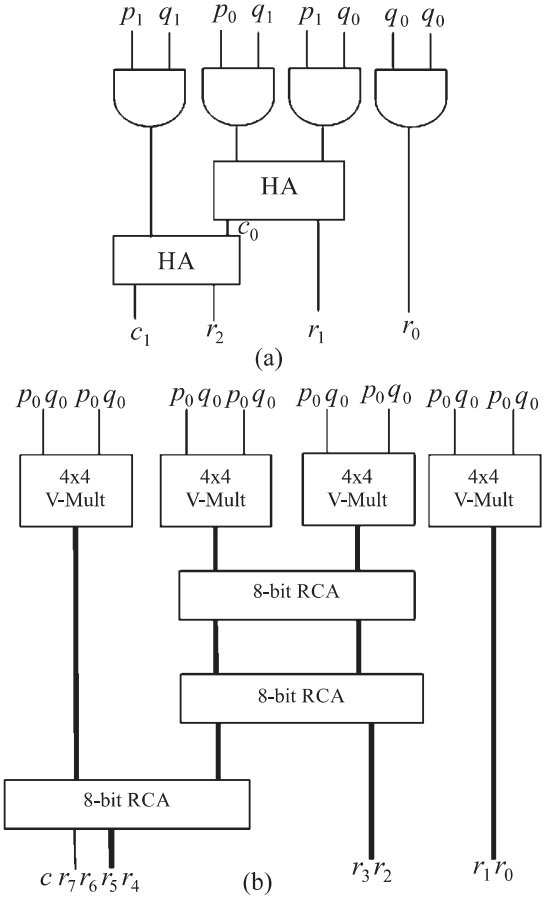


Fig. 5. Hardware implementation of (a) 2x2-bit Vedic multiplication (b) 8x8-bit Vedic multiplication

IV. Experimental Results

In this section, we evaluate the performance of 8-bit IIR filter through Vedic multiplication method. The logic synthesis and simulation is performed in Xilinx EDA (Electronic Design Automation) tool. The extensive simulations are performed for accurate results through various test benches. To verify the digital outputs of the filter, several waveforms are generated as shown in Fig. 6. The figure demonstrates accurate waveforms obtained from Xilinx EDA tool.

The 2nd order IIR filter of direct-form II, with band-pass Butterworth response is implemented in Matlab to find the coefficients of the filter. The magnitude, phase and impulse responses are shown in Fig. 7. The filter response has wider range during the lower order with low-computational complexity, good linearity and insensitive to the parasitic capacitances. The filter has band-pass sampling and cut-off frequencies of 8400Hz and 1320Hz respectively. The experimental result shows that the responses of the complex filter are very similar with the commercial software plots and shows accurate results in terms of

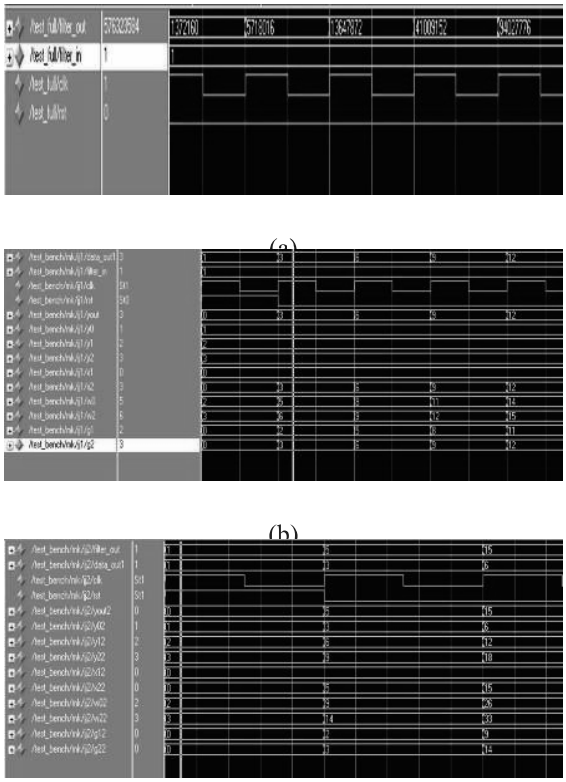


Fig. 6. (a) Filter output (b) Section-II simulation (c) Section-II simulation

The proposed Vedic multiplication method is analyzed for performance and well-know multiplication approaches. Table 1 demonstrates the comparison of the results in term of the basic design requirements. Table clearly shows the power consumption, memory usage and delay of the Booth, array and Vedic multipliers respectively. The Vedic multiplier has minimum hardware utilization and consumes almost equal power with Booth and Array multipliers as given in third and fourth column. In fifth and six columns, the Vedic multiplier occupies minimum memory usage and achieves highest performance with minimum gate delay.

TABLE I
Comparison of Different Multipliers

Multiplier	No. I/O	No. of Slices (Area)	Power (mW)	Memory Usage (KB)	Delay (ns)
Vedic	32	436	19.12	181233	18.12
Booth	32	770	19.06	230331	25.33
Array	32	856	21.88	250206	29.28

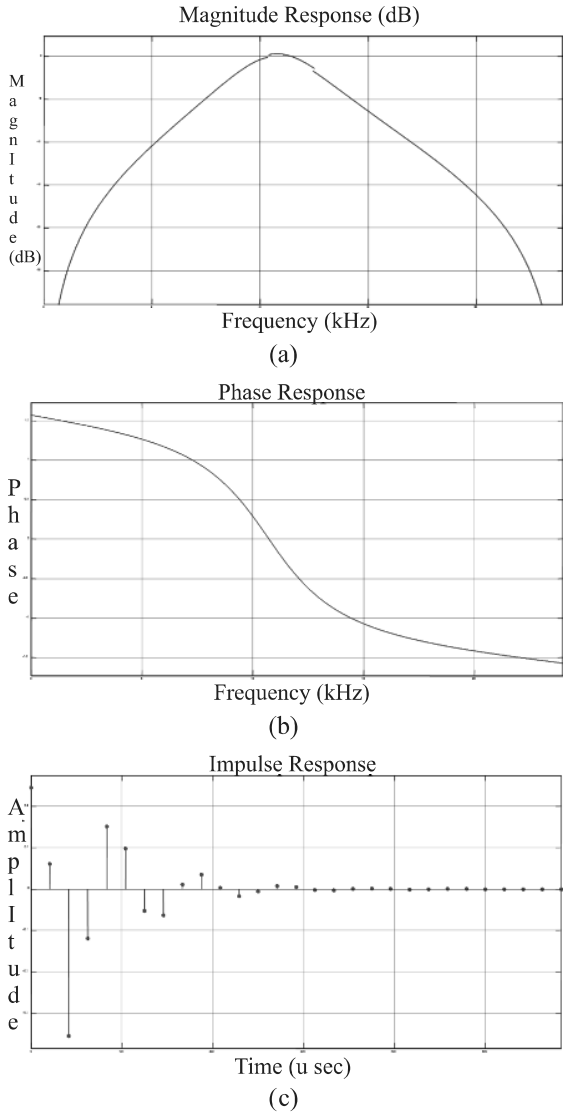


Fig. 7. Frequency responses of IIR band-pass filters with (a) Magnitude response (b) Phase response (c) Impulse response

V. Conclusions

The IIR filter is designed with Vedic multiplier method is proposed in this paper. The Vedic multiplier architecture is used due its low round-ff noise and low quantization errors. The results demonstrate that it reduces the computational delay and hardware to minimize the number of iterations and enhances the performance of the DSP processor. Vedic method enables the parallel processing stages of intermediate products and removes the unnecessary multiplication steps with zeros. The filter coefficients and frequency responses are found in Matlab. The magnitude and phase responses of the filter are very similar to the Multisim software.

References

[i] P. E. Danielson, "Serial-parallel convolvers", *IEEE Transaction on Computers*, vol. 33, no. 7, pp. 652-667, 1984.

[ii] R. Dutta, "Power efficient VLSI architecture for IIR filter using modified-Booth algorithm", *International Journal of Advanced Research in Technology*, vol. 2, no. 1, pp. 27-34, 2012.

[iii] K. Yeo, and Kaushik Roy, "Low-voltage, low-power VLSI subsystems", *McGraw-Hill companies publisher*, USA, 2005.

[iv] E. Singhal, "Performance analysis of FIR filter design using various window methods", *International Journal of Scientific Research Engineering & Technology*, vol.1, no. 5 , pp. 018-021, 2012.

[v] A. Gupta, U. Malviya, and V. Kapse, "Design high speed arithmetic logic unit based on ancient Vedic multiplication technique", *International Journal of Modern Engineering Research*, vol. 2, no.4, pp. 2695-2698, 2012.

[vi] S. Gupta, and A. Panghal "Performance analysis of FIR filter design by using rectangular, hanning and hamming windows methods", *International Journal of Advanced Research in Computer Science & Software Engineering*, vol. 2, no.6, pp. 334-347, 2012.

[vii] S. K. Mitra, "Digital Signal Processing: A computer based approach", *Mc-Graw Hill Higher Education Publisher*, 3rd Edition, 2006.

[viii] Y. Wang, and B. Li, "Two-stage based ensemble optimization for large-scale global optimization" *Proceeding of the IEEE Conference on Evolutionary Computation*, Spain, pp. 4488-4495, 2010.

[ix] Y. Wang, B. Li, and Y. B. Chen, "Digital IIR filter design using multi-objective optimization evolutionary algorithm" *Journal of Applied Soft Computing*, vol. 23, pp. 45-53, 2010.

[x] Y. Yu, and Y. Xinjie, "Cooperative co-evolutionary genetic algorithm for digital IIR filter design" *IEEE Transactions on Industrial Electronics*, vol. 54, pp. 1811-1819, 2007.

[xi] Y. Wang, B. Li, and W. Thomas, "Two-stage ensemble memetic algorithm: Function optimization and digital IIR filter design", *Journal of Information Sciences*, vol. 220, pp. 408-424. 2013.

[xii] H. Choo, K. Muhammad, and K. Roy, "Complexity reduction of digital filters using shift inclusive differential coefficients", *IEEE Transactions on Signal Processing,* vol. 52, pp. 1760-1772, 2014.

[xiii] P. Puri, and U. Patil, "High Speed Vedic Multiplier in FIR Filter on FPGA", *Journal of VLSI & Signal Processing*, vol. 4, no.3, pp. 48-53, 2014.

[xiv] K. Padma, Z. Sameena, S. Ankita, "16-order IIR filter design using Vedic mathematics technique", *International Journal of Engineering Innovation and Research*, vol. 3, no. 2, pp. 138-145, 2014.