Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/110427

Delay Optimized Novel Architecture of FIR Filter using Clustered-Retimed MAC unit Cell for DSP Applications

S. Subathradevi[1,*] *and C. Vennila*[2]

[1] Department of ECE, Anna University BIT Campus, Tiruchirappalli, India
[2] Department of ECE, Saranathan College of Engineering, Tiruchirappalli, India

Received: 27 May 2017, Revised: 20 Jun. 2017, Accepted: 25 Jun. 2017
Published online: 1 Jul. 2017

Abstract: In VLSI based system design; optimum design is preferred in terms of area, delay and power. In the era of miniaturization the speed is the most important parameter in VLSI design. To achieve greater speed the designer must concentrate on reduction in delay. Generally the total delay consists of logic delay and path delay. For achieving delay optimization either path delay or logic delay can be modified. By re-arranging the architecture such that to modify interconnects, the delay can be reduced in the design of the system and speed can be comparatively increased. In this work, the authors propose a novel methodology for the construction of FIR filter design as a delay optimized one. The delay reduction is achieved using the reduction in the path delay,. The retimed MAC unit cell is used as a sub-component in the construction of FIR filter to obtain the required reduction. This construction was implemented using Xilinx software tool with the FPGA Spartan-3E and Virtex device.

Keywords: VLSI System design, Architecture, FIR filter, delay, Retiming, MAC Unit, Clustering.

1 Introduction

Finite Impulse Response (FIR) digital filter is a common component in many digital signals processing (DSP) system. In signal processing, a finite impulse response filter plays a vital role in design and analysis. With the rapid development in very large scale integration (VLSI) technology, the real time realization of FIR filter with less hardware requirement has reduced delay and less latency has become more important.

Generally the speed of the design is contributed by the critical path i.e. the longest path. The critical path length is directly related with the interconnect path which exist between sub-modules. This interconnected configuration decides the longest path of propagation. In any system design the arithmetic units are as much important as to make it a successful design. The delay constraint is being met by the number of adders and multipliers used in the FIR data flow graph architecture and the configuration in which how they are interlinked decides the performance of FIR filter.

Adders and multipliers play a major role in the design of FIR filter, since the total delay depends on the delay taken by number of adders and multipliers present in the architecture based on the N value in an N-tap filter. The speed of the design can be improved by minimizing the delay in the architecture of adders and multipliers that may lead to better performance. Several algorithms have been proposed for the sub-modules used in the literature for having effective architectures and implemented using ASIC and FPGA. In this work, the authors propose a delay optimized novel FIR filter with the retimed MAC unit cell as a sub-component for achieving greater speed.

The flow of the paper is organized as, Section 1 proceeds the Introduction. Section 2 gives out the details of FIR filter, Section 3 deals with Literature Survey related to the proposed work, Section 4 briefs the proposed work and its contribution to achieve the objective of this work, Section 5 narrates experimental results with discussion and Section 6 arrives at the conclusion.

2 FIR Filter

In signal processing, a finite impulse response (FIR) filter is a filter whose impulse response (or response to any finite length input) is of finite duration and it settles to

* Corresponding author e-mail: saminathan.sruthi@yahoo.com

zero within the finite time. This is in contrast to infinite impulse response (IIR) filters, which may have internal feedback and may continue to respond indefinitely. The impulse response of an N-th order discrete-time FIR filter lasts exactly $N+1$ samples before it settles to zero [1].

Finite impulse response (FIR) filter is a fundamental building block in many digital signal processing (DSP) systems. Efficient hardware implementation of FIR filters is thus important for high-speed and low-power realization of DSP systems. For hardware implementation of FIR filters, the transposed direct form (TDF) structure is preferred to the direct form structure due to its inherent pipelined structure and the shortest critical path delay. The input variable is concurrently multiplied by a set of filter coefficients in a TDF FIR filter. This operation is referred to as multiple constant multiplications (MCM). The product words generated by the MCM block are delayed and accumulated by the product–accumulation section to produce the filter output. Typically, the MCM block of an FIR filter is realized by a network of shift-add operations. The efficient implementation of the MCM blocks has been, widely studied in the present decade and several algorithms have been proposed to design high-speed and low-complexity MCM blocks [2].

The direct form is directly related to the difference equation. A chain of N register stages provides all delay input signals $x(n-k)$. $N+1$ multiplications are processed in parallel with coefficients $C_k = h(k)$. Products are added with one adder and consecutive additions as a process will form an adder chain in FPGA hardware. The longest signal path is built by the N adder chain and one multiplier. Hardware implementation of direct form-I under clock frequency constraints: the adder tree should be balanced by adding pairs of products. Further adder stages will combine pairs of added products. The number of adders will remain the same but a shorter signal delay path through the adders will result. The parallel structure of a balanced adder tree reduces the number of delay stages from N to $\sqrt{N}/2$. Separation of adder stages by register will allow a higher clock frequency. The products can be stored and a register is assigned to every adder stage. In case of a balanced adder tree pipelining will need less registers and less clock cycles. With $f_s \ll f_{clock}$ the products are constant during a sample interval so that no registers are needed for delay balancing [3].

In transpose form, it is derived from the direct form by several manipulations of the filter structure: (i) Interchange input and output, (ii) Reversal of signal flow graph in direction of arrow, (iii) Substitution of adders by a branch and vice versa. The main advantage of this structure is provided by the double use of delay stages because the registers directly decouple the adder stages. No additional pipelining stages for the adder tree are needed. The number of D-FFs increase because now products with $m = j+1$ bits are registered. The longest signal delay path is with a multiplier (coefficient C_0) and the last adder stage which contains the longest ripple carry chain with $(m+\log_2(N+1))$ bits [1].

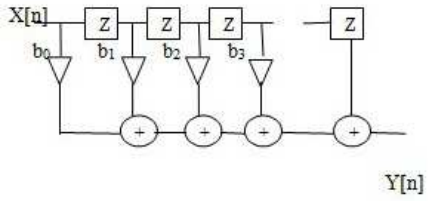

Fig. 1: N-tap FIR Filter (Data Flow Graph).

In pipelining form, it can increase the amount of concurrency in an algorithm. Pipelining is accomplished by placing latches at appropriate three different intermediate points in a dataflow graph that describes the algorithm. Each latch is also referred to a storage unit or buffer or register [3].

The latches can be placed at feed forward cut sets of the data flow graph. In synchronous hardware implementations, pipelining can increase the clock rate of the system and therefore the sample rate. The drawbacks associated with pipelining are the increase in system latency and the increase in the number of registers. While pipelining can be easily applied to all algorithms with no feedback loops by the appropriate placement of latches, it cannot be easily applied to algorithms with feedback loops. Pipelining can also be used to improve the performance in software programmable multiprocessor systems [3].

Parallel processing exploits concurrency by performing multiple larger tasks simultaneously in separate hardware units. In a 3-tap filter with two input samples, two output samples are generated in each clock cycle period within $4T_a$ times. Because each clock cycle processes two samples, hence the effective sample rate is $(1/2)(T_a)$. The parallel architecture leads to the speed increase with significant hardware overhead. The entire dataflow graph needs to be replicated with an increase in the amount of parallelism [4].

Different architectures of FIR filters are realized using verilog code, synthesized and simulated using ISim and then implemented on target device as Spartan 3E and Virtex FPGA. The architectures are: Direct form FIR filter, Transpose form FIR filter, Pipelined form of FIR filter and Novel proposed FIR architecture using retimed MAC unit cell [2].

Fig. 1 shows the data flow diagram for N-tap FIR filters. Triangle shape shows multiplier. The incoming $x[n]$ is multiplied with b_0, b_1, b_2, b_3, etc. which are the coefficients. Encircled plus sign symbol shows adders which are used to accumulate the convolution of an $x[n]$ where $n = 0,1,2,3$ etc.

Fig. 2 shows the data flow graph diagram for a 4-tap FIR filter. When the number of taps increases the longest path delay also increases. Also number of multipliers and adders required in the architecture also increases. The data flow graph shown in Fig. 2 is in direct form. Here the

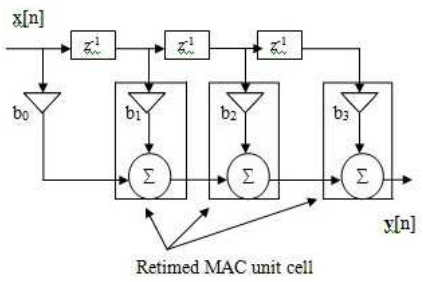

Fig. 2: 4-tap FIR Filter (Direct form).

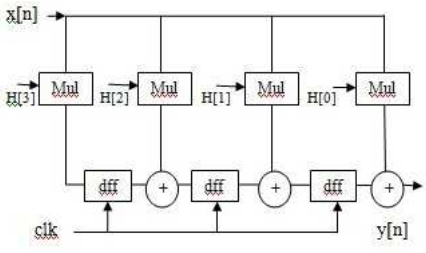

Fig. 3: FIR Filter Transpose form.

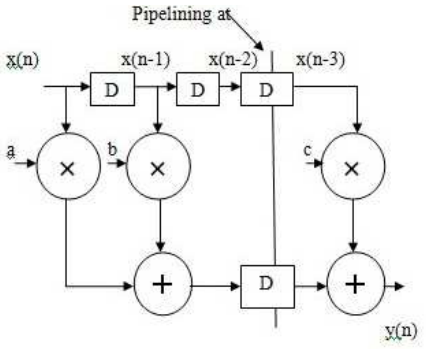

Fig. 4: FIR filter with 1-level pipelining.

longest path is decided by the time taken for the computation of one multiplier and four adders. i.e. $T_m + 4T_a$ [5].

Fig. 3 shows the data flow graph of a 4-tap FIR filter in transpose form. Here the longest path is decided by the time taken for the computation of one multiplier and one adder. i.e. $T_m + T_a$ [5].

Fig. 4 gives the data flow graph of FIR filter with 1-level pipelining done on the feed forward cut-set. In general pipelining results in reduced delay when compared with conventional architecture [4].

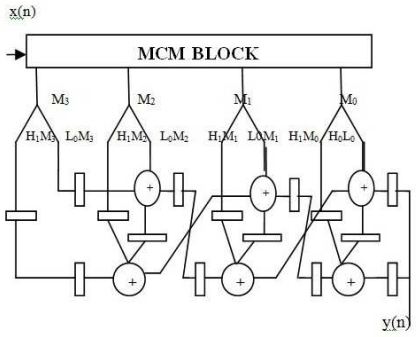

Fig. 5: Retiming in Accumulation block [1].

3 Literature Survey

Different architectures of FIR filter with less delay as target are surveyed compared and their contribution in terms of proposed work is viewed in connection with delay. In terms of delay optimization, many existing literatures are surveyed and are briefed related to the proposed work in this section.

In the existing literature [2] authors have used two sub-blocks. They are MCM block which holds multiplication and Process Accumulation block which holds addition. Authors have implemented retiming in the PA block alone by performing faster addition in PA block using an algorithm. Fig. 5 shows the graphical representation of it [2].

Whereas in paper [6] the authors explained about the efficient design of FIR filter in order to reduce the number of adders and multipliers and also the hardware complexity. For that, they have used algorithmic approach for performing addition using Hardware Description Language [6].

An efficient FIR filter design on FPGA was given in [7]. MATLAB tool was used to determine filter coefficients for a constant FIR filter. VHDL was used for implementation. The software ISE10.1 was used to simulate the performance of filter using two different techniques. Usage of this software has significantly reduced the design time. The authors of [7] have concluded that the windowing is better compared to frequency sampling.

Authors of paper [3] have designed a reduced dynamic power consumption based digital filter which uses low power MUX based Shift/ Add multiplier without clock pulse. Glitching is also reduced. The proposed FIR filter has been synthesized and implemented using Xilinx ISE and Virtex. The proposed FIR filter has been synthesized and implemented using Xilinx ISE and implemented on target device XC4VLX200 Virtex FPGA. Power analysis was done using Xilinx XPower analyzer [3].

In paper [8] a smoothing FIR filter is designed for discrete time invariant state space polynomial models

which is commonly used to model signals over finite data. It finds applications in digital communication network, where it is used to find time interval errors of local slave clocks [8].

Distributed Arithmetic has been used in paper [9] to implement a bit serial scheme of a general asymmetric version of an FIR filter taking optimal advantage of three input LUT based structure of FPGA [9].

Design and implementation of low pass, high pass and band pass FIR filter were implemented in paper [1] using Spartan-6 FPGA device. The EDA tool was used to define filter order and coefficients, FIR filter was designed using Simulink simulation [1].

4 Proposed Architecture

In the existing architecture [2] the authors have tried to optimize the delay by concentrating on the accumulation unit alone. But this work proposes an efficient clustered and retimed MAC unit cell to reduce the longest path delay. The implemented conventional FIR filter consists of a retimed MAC unit cell. The sub-modules present in the conventional FIR filter architecture is binary array multiplier and ripple carry adder. But in the proposed architecture it consists of MAC unit cell which uses retiming. The sub-modules present in the retimed MAC unit cell are Vedic multiplier and Koggestone Adder. They are together used as a MAC unit. And that MAC unit is retimed using a retiming register in the feed forward path which exists between Vedic multiplier [10] and Koggestone adder [11].

The critical path can be reduced by reducing the path existing between any two sub-modules (inter delay) or by reducing the path within the sub-module (intra-delay). The proposed work is aimed at reducing path length within the sub-modules (intra-delay) called retimed cell.

The retimed MAC cell is a sub-module which consists of one Vedic multiplier and one Koggestone adder. In conventional FIR structure they are binary array multiplier and ripple carry adder. But in the proposed work they are Vedic multiplier and Koggestone adder.

The modification used in the architecture on the whole or in the sub-module leads to design filter in an efficient way.

The Vedic multiplier is the one of the fastest multiplier which is verified from literature and also by experimentation [10]. In the same way Koggestone adder is one of the fastest adders which are also verified by literature and by experiment [11]. Retiming is also an important technique used to achieve delay improvement. In the proposed architecture FIR filter sub-module MAC unit consists of all the above three which impact the reduced delay in the design.

The proposed novel MAC unit cell without retiming is shown in Fig. 6 and with retiming is shown in Fig. 7. A novel sub-module structure with Vedic multiplier and Koggestone adder with the retiming in between them is

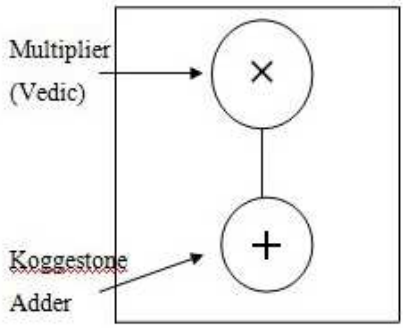

Fig. 6: MAC UNIT without Retiming.

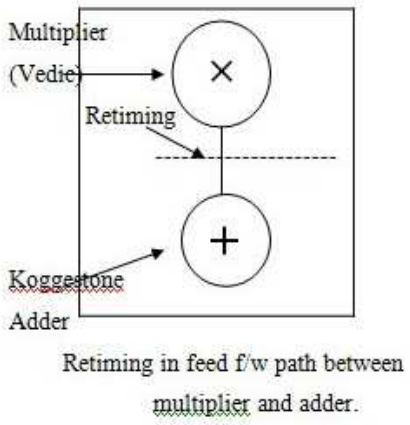

Fig. 7: Clustered MAC UNIT with Retiming.

clustered together as one sub module. This MAC unit is used as sub-module in different architectures to justify that it results in delay reduction in the design.

Those architectures are direct form FIR filter, Transpose form FIR filter, FIR filter with 1-level pipelining. This new sub-module retimed MAC unit is used in a 4-tap FIR filter as shown in Fig. 2. Obviously it shows that multiplier and adder were replaced by this new MAC cell.

So, in the three architectures of FIR filter as a first case they are implemented using the sub-modules of binary array multiplier and carry save adder. As a second case three architectures are implemented using the proposed sub-module retimed MAC unit cell. Compared with first case second case results in reduced delay.

5 Experimental Results with Discussion

In this proposed architecture of FIR filter data flow graph is designed for delay optimized system design. And it is discussed based on the various delay constrained architectures which already exists. Different architectures

Table 1: Experimental results for conventional FIR filter with binary array multiplier and ripple carry adder.

Type of the Architecture/ Parameter	Area		Delay
	Slices (4656)	LUTs (9312)	(ns)
Direct form	74	130	15.389
Transpose form	72	126	14.091
Pipeline in 1-Level	82	147	13.886

Table 2: Experimental results for the FIR filter with Vedic multiplier and Koggestone adder.

Type of the Architecture/ Parameter	Area		Delay
	Slices	LUTs	(ns)
Direct form	74	130	15.389
Transpose form	72	126	14.091
Pipeline in 1-Level	82	147	13.886

Table 3: Experimental results for the proposed FIR filter architecture with clustered and retimed MAC UNIT cell.

Type of the Architecture/ Parameter	Area		Delay
	Slices	LUTs	(ns)
Direct form	78	143	12.836
Transpose form	76	129	12.765
Pipeline in 1-Level	79	144	12.682

of FIR filter for 4-tap, 8-tap, 12-tap and 16-tap with the data of 4-bits, 8-bits, 12-bits and 16-bits are designed and implemented on the target FPGA Spartan 3E device and on Virtex FPGA using the tool Xilinx ISE 14.5. Those architectures are synthesized using the XST-tool in the Xilinx software 14.5 version for checking the functionality justification and physical verification. Verilog code is used for making a prototype of the proposed architecture which leads to the optimized delay in trade-off with an average of 5% increase in area.

Table 1 shows the experimental result data for the conventional FIR filter architecture of 4-bits with binary array multiplier and ripple carry adder as the sub-module; In Table 2, the results with Vedic multiplier and Koggestone adder as the sub-module without retiming is given. Table 3 shows the experimental results for the proposed FIR filter architecture with clustered and retimed MAC UNIT cell. In all the 3 tables, area in terms of slices and LUTs, and delay in terms of ns are listed.

The pictographs in Figs. 8–11, they provide the comparison of different FIR data flow graph architectures in terms of Slices, LUTs and Delay. It inferred that the delay of the retimed cell is lower in its value compared to the conventional architecture [2]. This is true for all architectures.

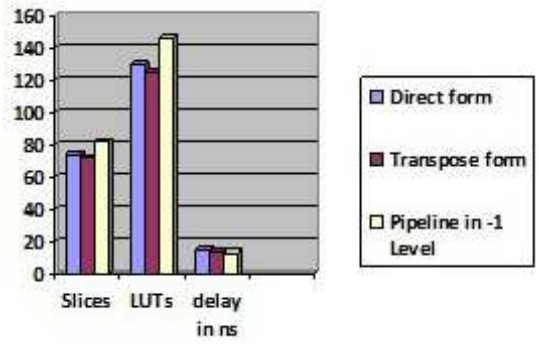

Fig. 8: Pictograph representation of the results present in Table 1.

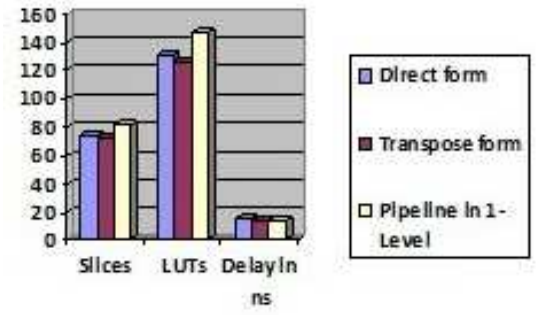

Fig. 9: Pictograph representation of the results present in Table 2.

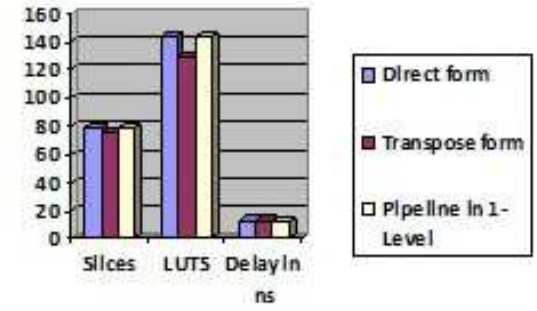

Fig. 10: Pictograph representation of the results present in Table 3.

6 Conclusion

The construction of FIR filter and its design as a delay optimized one has been discussed. It has demonstrated by an experiment that the delay reduction is achieved using the reduction in the path delay. Different architectures have been implemented with 3-tap, 4-tap, 5-tap and 6-tap with the input bit size as 4-bit, 8-bit, 12-bit and 16-bit. A

Table 4: Area, delay comparison table for all the four architectures direct form, transpose form and 1-level pipelining of 3-tap for 8-bit $x[n]$.

Method Used in MAC cell	Direct Form Delay (ns)	Transpose form Delay (ns)	Pipeline Form Delay (ns)
Array Multiplier and ripple adder	19.593	19.851	18.012
Vedic multiplier & Koggestone	15.389	14.091	13.866
Vedic & Kogge with retiming [Proposed]	12.836	12.765	12.682
Percentage of improvement (with Vedic)	21.45%	29.01%	23.01%
Percentage of increase in area (with Vedic and retiming)	0%,0%	0%,0%	0%,0%
Percentage of improvement (with Vedic and retiming)	34.48%	35.69%	29.59%
Percentage of increase in area (with Vedic and retiming)	5% 10%	5%, 2%	13%, 1%

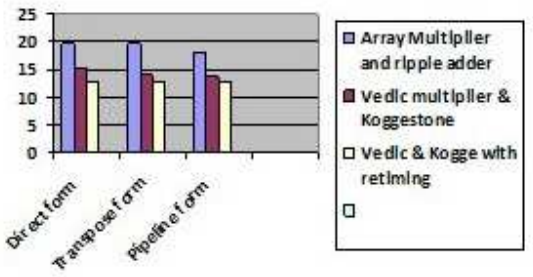

Fig. 11: Pictograph representation of the results present in Table 4.

novel sub-module structure with Vedic multiplier and Kogge stone adder with the retiming in between them is clustered together as one sub module. So, it results in an optimized delay. When compared with conventional architecture, this proposed work results on an average of 25% improvement in delay against the trade off with increase in 7% area. This construction is implemented using Xilinx software tool on FPGA Spartan-3E and Virtex device. In future, the introduction of this clustered retimed MAC unit cell would be a better solution to get fast high level synthesis VLSI designs with reduced delay performance. This architecture will find applications in high speed system design like System on-Chip, Network on Chip and in Signal processing.

References

[1] Emmanuel S. Kolawole Warsame H. Ali, Penrose Cfie, John Fuller, C. Tolliver, Pamela Obiomon, "Design and Implementation of Low-pass, High-pass and Band- pass Finite Impulse Response (FIR) Filters using FPGA", Circuits and Systems 2015, **6**, 30–48, Published online February 2015 in SciRes.

[2] Ya Jun Yu and Pramod Kumar Meher, "Analysis and Optimization of Product- Accumulation Section for Efficient Implementation of FIR Filters", IEEE Transactions on Circuits and Systems Digital Object Identifier 10.1109/TCSI. 2587105, 1549-8328, 2016.

[3] Bahram Rashidi, Farshad Mirzaei, Bahman Rashidi and Majid Pourormazd, "Low power FPGA Implementation of Digital FIR Filter based on Low Power Multiplexer Based Shift/Add Multiplier", International Journal of Computer Theory and Engineering, **5(2)**, April 2013.

[4] T. Arslan, et al. "Low power implementation of high throughput FIR filters", IEEE International Symposium on Circuits and Systems, **4**, 373–376, 2002.

[5] Miss Pooja D Kocher, Mr. U. A. Patil, "Implementation and Comparison of Low pass FIR Filter on FPGA using different Techniques", International Journal of Innovative Research in Science, Engineering and Technology, **2(9)**, September 2013.

[6] Keshab K. Parhi, "VLSI Digital Signal Processing Systems– Design and Implementation", Wiley Publications©, Delhi, 1999.

[7] Kai-Yuan Jheng, Shyh-Jje Jou and An-Yeu Wu, "A design flow for multiplier less linear-phase FIR filter: from system specification to verilog code", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, **4(1)**, January 2004.

[8] Ritu Sharma, Ravi Mohan Verma, "FPGA Implementation of Fast Fourier Transform (FFT) Based Finite Impulse Response (FIR) filter using VHDL", International Journal of Engineering and Computer Science, ISSN: 2319-7242, **3(3)**, March 2014.

[9] Ibarra-Manzano Oscar, Shmaliy Yuiy. S and Morales-Mendoza Luis, "Smoothing FIR Filtering of discrete state-space polynomial signal models", Conference Paper in Acoustics, Speech and Signal Processing, 1988.

[10] P. Sravanthi, C. H. Srinivasa Rao, "A novel approach of area-efficient FIR filter design using Distributed Arithmetic with Decomposed LUT", IOSR Journal of Electronics and communication Engineering, e-ISSN: 2278–2834, p-ISSN:2278–8735, **7(2)**, July–Aug, 2013.

[11] Harpreet Singh Dhillon and Abhijit Mitra, "A Digital Multiplier Architecture using Urdhva Tiryakbhyam Sutra of Vedic Mathematics", Department of Electronics and Communication Engineering, Indian Institute of Technology, Guwahati, e-material.

[12] V. Krishnakumari, Y. Sri Chakrapani, "Designing and Charecrerisation of Koggestone, Spanning tree and Brent Kungg adder", International Journal of Modern Engineering

Research, **3(4)**, 2266–2270, ISSN 2249-6645, July–August 2013.
[13] Pooja D Kocher, U.A. Patil, "Implementation and Comparison of Low pass FIR Filter on FPGA using different Techniques", International Journal of Innovative Research in Science, Engineering and Technology, **2(9)**, September 2013.
[14] Yinan Wang, Hakan Johansson and Jietao Diao, "Minimax Design and Order Estimation of FIR Filters for Extending the Bandwidth of ADCs", 4 March 2016.
[15] Jeffrey T. Ludwig, Hamid Nawab, Ananta P. Chandrasekaran, "Low Power digital filtering using approximate processing", IEEE journals of solid- state circuits, **31(3)**, March 1996.
[16] Dusan M. Kodek, "Design of optimal finite word length FIR digital filters using Integer programming techniques", IEEE transactions on acoustics speech and signal processing, **28(3)**, June 1980.
[17] Ludwig J. T., et al. "Low power digital filtering using approximate processing", IEEE Journal of Solid-State Circuits, **31(3)**, 395–399, 1996.
[18] Manish Bhardwaj, et al. "Quantifying and Enhancing Power Awareness of VLSI Systems", IEEE Transactions on VLSI Systems, **9(6)**, 757–772, 2001.
[19] H. J. G. Chung and K. K. Parhi, "Frequency spectrum based low-area low- power parallel fir filter design", EURASIP Journal on Applied Signal Processing, **31**, 944–953, 2002.
[20] A.F. Shalash and K.K. Parhi, "Power efficient folding of pipelined LMS adaptive filters with applications", Journal of VLSI Signal Processing, pp. 199213, 2000.
[21] K. Tarumi, A. Hyodo, M. Muroyama, and H. Yasuura, "A design method for a low power digital FIR Filter in digital wireless communication systems", 2004.
[22] A. Senthilkumar, A. M. Natarajan, and S. Subha, "Design and implementation of low power digital FIR filters relying on data transition power diminution technique", DSP Journal, **8**, 21–29, 2008.
[23] Chao-Tsung Huang, Po-Chih Tseng, and Liang-Gee Chen, "Analysis and VLSI Architecture for 1-D and 2-D Discrete Wavelet Transform", IEEE Transactions on signal processing, **53(4)**, April 2005.
[24] Chih-Chi Cheng, Chao-Tsung Huang, Ching-Yeh Chen, Chung-JrLian, and Liang- Gee Chen, "On-Chip Memory Optimization Scheme for VLSI Implementation of Line-Based Two-Dimentional Discrete Wavelet Transform", IEEE Transactions on circuits and systems for video technology, **17(7)**, July 2007.
[25] XinTian, Lin Wu, Yi-Hua Tan, and Jin-Wen Tian, "Efficient Multi-Input/Multi- Output VLSI Architecture for Two-Dimensional Lifting-Based Discrete Wavelet Transform", IEEE transactions on computers, **60(8)**, August 2011.
[26] Sze-Wei Lee, Soon-Chieh Lim, "VLSI Design of a Wavelet Processing Core", IEEE transactions on circuits and systems for video technology, **16(11)**, November 2006.
[27] Chao Cheng, Keshab K. Parhi, "High-Speed VLSI Implementation of 2-D Discrete Wavelet Transform", IEEE transactions on signal processing, **56(1)**, January 2008.
[28] Amit Acharyya, Koushik Maharatna, Bashir M. Al-Hashimi, Steve R. Gunn, "Memory Reduction Methodology for Distributed-Arithmetic-Based DWT/IDWT Exploiting Data Symmetry", IEEE transactions on circuits and systems, **56(4)**, April 2009.
[29] Yamini, S. Bute, R.W. Jasutkar, "Implementation of Discrete Wavelet Transform Processor For Image Compression", International Journal of Computer Science and Network (IJCSN) Volume 1, Issue 3, June 2012 www.ijcsn.org ISSN 2277–5420.
[30] Swami Bharati Krsna Tirtha, Vedic Mathematics. Delhi: Motilal Banarsidass Publishers, 1965.
[31] D. Goldberg, "Computer Arithmetic in Computer Architecture: A Quantitative Approach", J. L. Hennessy and D. A. Patterson ed., pp. A1A66, San Mateo, CA: Morgan Kaufmann, 1990.
[32] A.D. Booth, "A Signed Binary Multiplication Technique", Qrt. J. Mech. App. Math., vol. 4, pp. 236–240, 1951.
[33] P.D. Chidgupkar and M.T. Karad, "The Implementation of Vedic Algorithms in Digital Signal Processing", Global J. of Engg. Edu., **8(2)**, 2004.
[34] H. Thapliyal and M.B. Srinivas, "High Speed Efficient $N \times N$ Bit Parallel Hierarchical Overlay Multiplier Architecture Based on Ancient Indian Vedic Mathematics", Enformatika Trans., **2**, 225–228, Dec. 2004.
[35] Rashmi Patil, M.T. Kolte, "VLSI Implementation of FIR Filter for Discrete Wavelet Transform", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, **4(1)**, January 2015.

S. Subathradevi, is an Assistant Professor in the Department of Electronics and Communication Engineering, Anna University- BIT Campus, Trichy. Her area of research interest is towards VLSI design, Digital System design and CAD VLSI.

C. Vennila, is a Professor in the Department of Electronics and Communication Engineering, Saranathan College of Engineering, Trichy. Her area of research interest includes reconfigurable architecture, Algorithms and techniques for cognitive radio and VLSI design.