Review Paper on Finite Impulse Response Module and its FPGA Implementation

Shyam Ji[1], Prof. Suresh. S. Gawande[2], Prof. Satyarth Tiwari[3]

[1]M. Tech. Scholar, [2]Guide, [3]Co-guide

Department of Electronics and Communication, Bhabha Engineering Research Institute, Bhopal

Abstract— It offers the best reduction in delay and power. However to provide more configurations, Programmable Shift Method (PSM) based reconfiguration technique also be presented in the current research work. It reduces the hardware area utilization than direct form FIR filter. To further improve the performances of PSM based reconfiguration techniques, Pre-Computer based architectures have been developed and this technique gives the better performances than PSM based architectures in terms of less hardware slice utilization and power consumptions. The proposed architectures have been implemented in SPARTAN FPGA development board. Therefore, a shared-LUT design is proposed to realize the MDA computation. Instead of using separate registers to store the possible results of partial inner products for DA processing of different bit positions, registers are shared by the DA units for bit slices of different weightage. The all design implemented in Xilinx Virtex-5 FPGA device (XC5VSX95T-1FF1136).

***Keywords: - Finite Impulse Response (FIR), Look Up Table (LUT), Modified Distributive Arithmetic Technique*.**

I. INTRODUCTION

In signal processing and wireless communication-based technology, filters are widely used and considered to be the first hands on the tool in any electronic circuits. In fact, their extraordinary performance effect is one of the key reasons that DSP has become so popular. The primary function of the filter is to selectively allow the desired signal to pass through and suppress the undesired signal based on particular frequency. The process of designing a signal processing filter satisfies a set of requirements which are realization and optimization of the filter. The purpose is to find a realization of the filter that meets each of the requirements to a sufficient degree to make a useful. The filter design estimates the loss and degree of phase shift which occurs during insertion and rejection. The filters are broadly be classified into Analog filters and Digital filters. The main operations required for DA-based computation are a sequence of lookup-table (LUT) accesses followed by shift-accumulation operations of the LUT output. The conventional DA implementation used for the implementation of FIR filter assumes that impulse response coefficients are fixed and this behavior makes it possible to use ROM-based LUTs. The memory requirement for DA-based implementation of FIR filters, however, increases exponentially with the filter order. To get rid of the problem of such large memory requirement, systolic decomposition techniques are suggested by Meher et al. for DA-based implementation of long-length convolutions and FIR filter of large orders [7], [8]. For reconfigurable DA-based FIR filter whose filter coefficients change dynamically, we need to use rewritable RAM based LUT [9] instead of ROM-based LUT. Another approach is to store the coefficients in the analog domain by using serial digital-to-analog converters resulting in mixed-signal architecture [10]. We also find quite a few works on DA based implementation of adaptive filters [11], [12] where the coefficients change at every cycle. In this paper, we present efficient schemes for the optimized shared-LUT implementation of reconfigurable FIR filters using DA technique where LUTs are shared by the DA units for bit slices of different weightage. Also, the filter coefficients can be changed dynamically in runtime with very small reconfiguration latency. In the next section, we briefly discuss the mathematical background of DA-based implementation of FIR filter.

II. LITERATURE REVIEW

Usha Maddipati et al. [1], in this paper, concentrating on the real time demands of digital signal processing, a delay and power efficient 16-tap direct form low pass FIR filter is realized using FPGA. The filter coefficients are generated using Kaiser Window function of MATLAB FDA tool. For obtaining the high speed operation at reasonable power, various adder architectures are considered for the filter design along with vedic multiplier. The designs were implemented on Artix-7 xc7a100tcsg324-1 FPGA board and debugged using Virtual Input/output IP of Xilinx Vivado to validate the results. Experimental results show that efficiency in power-delay product can be obtained by using Carry Increment Adder for FIR filter design than that of various other multi-bit adder structures.

Ankit Upadhyay et al. [2], the execution of FIR channels on FPGA taking into account conventional technique costs significant equipment assets, which conflicts with the diminishing of circuit scale and increment of framework pace. FIR channels utilizing Arithmetic is utilized to build

the asset use while pipeline structure is additionally used to expand the framework speed. Moreover, the isolated LUT strategy is additionally used to diminish the required memory units.

FIR filter implemented using basic Arithmetic architecture is based on bit serial operation resulting in increase in delay with decrease in speed of operation. This is because the entire co-efficient are stored in single LUT. In Parallel DA architecture, instead of storing the co-efficient in single LUT as in traditional Arithmetic architecture, it is split into several ROM LUT's. All the LUT's are provided with different inputs at the same time, implying parallel mechanism. This increases the speed of operation.

Basant Kumar Mohanty et al. [3], transpose form finite-impulse response (FIR) filters are inherently pipelined and support multiple constant multiplications (MCM) technique that results in significant saving of computation. However, transpose form configuration does not directly support the block processing unlike direct form configuration. In this paper, we explore the possibility of realization of block FIR filter in transpose form configuration for area-delay efficient realization of large order FIR filters for both fixed and reconfigurable applications. Based on a detailed computational analysis of transpose form configuration of FIR filter, we have derived a flow graph for transpose form block FIR filter with optimized register complexity. A generalized block formulation is presented for transpose form FIR filter.

B. Madhu Latha et al. [4], this brief proposes a two-step optimization technique for designing a reconfigurable VLSI architecture of an interpolation filter for multi standard digital up converter (DUC) to reduce the power and area consumption. The proposed technique initially reduces the number of multiplications per input sample and additions per input sample by 83% in comparison with individual implementation of each standard's filter while designing a root-raised-cosine finite-impulse response filter for multi standard DUC for three different standards. In the next step, a 2-bit binary common subexpression (BCS)-based BCS elimination algorithm has been proposed to design an efficient constant multiplier, which is the basic element of any filter. This technique has succeeded in reducing the area and power usage by 41% and 38%, respectively, along with 36% improvement in operating frequency over a 3-bit BCS-based technique reported earlier, and can be considered more appropriate for designing the multi standard DUC. An advantage of this module is used to 2-bit binary common subexpression technique and reduces area and power of the system but disadvantage of this module is not used to higher bit.

Sang Yoon Park et al. [5], this paper presents efficient distributed arithmetic (DA)-based approaches for high-throughput reconfigurable implementation of finite impulse response (FIR) filters whose filter coefficients change during runtime. Conventionally, for reconfigurable DA-based implementation of FIR filter, the lookup tables (LUTs) are required to be implemented in RAM; and the RAM-based LUT is found to be costly for ASIC implementation. Therefore, a shared-LUT design is proposed to realize the DA computation. Instead of using separate registers to store the possible results of partial inner products for DA processing of different bit positions, registers are shared by the DA units for bit slices of different weightage. The proposed design has nearly 68% and 58% less area-delay product, and 78% and 59% less energy per sample than DA-based systolic structure and carry saved adder (CSA)-based structure, respectively for the ASIC implementation.

Basant K. Mohanty et al. [6], we have analyzed memory footprint and combinational complexity to arrive at a systematic design strategy to derive area-delay-power-efficient architectures for two-dimensional (2-D) finite impulse response (FIR) filters. We have presented novel block based structures for separable and non-separable filters with less memory footprint by memory sharing and memory-reuse along with appropriate scheduling of computations and design of storage architecture. The proposed structures involve times less storage per output (SPO), and nearly times less energy consumption per output (EPO) compared with the existing structures, where is the input block-size. They involve times more arithmetic resources than the best of the corresponding existing structures, and produce times more throughput with less memory band-width (MBW) than others. We have also proposed separate generic structures for separable and non-separable filter-banks, and a unified structure of filter-bank constituting symmetric and general filters.

Basant K. Mohanty et al. [7], in this paper, we present an efficient distributed arithmetic (DA) formulation for the implementation of block least mean square (BLMS) algorithm. The proposed DA-based design uses a novel look-up table (LUT)-sharing technique for the computation of filter outputs and weight-increment terms of BLMS algorithm. Besides, it offers significant saving of adders which constitute a major component of DA-based structures. Also, we have suggested a novel LUT-based weight updating scheme for BLMS algorithm, where only one set of LUTs out of sets need to be modified in every iteration, where , , and are, respectively, the filter length and input block-size. Based on the proposed DA formulation, we have derived a parallel architecture for the implementation of BLMS adaptive digital filter (ADF). Compared with the best of the existing DA-based LMS structures, proposed one involves nearly times adders and times LUT words, and offers nearly times throughput of the other.

III. DISTRIBUTIVE ARITHMETIC TECHNIQUE

Distributed Arithmetic (DA) is a widely-used technique for implementing sum-of-products computations without the use of multipliers. Designers frequently use DA to build efficient Multiply-Accumulate Circuitry (MAC) for filters and other DSP applications. The main advantage of DA is its high computational efficiency. DA distributes multiply and accumulates operations across shifters; lookup tables (LUTs) and adders in such a way that conventional multipliers are not required.

Distributed arithmetic is an important algorithm for DSP applications. It is based on a bit level rearrangement of the multiply and accumulate operation to replace it with set of addition and shifting operations. The basic operations required are a sequence of table lookups, additions, subtractions and shifts of the input data sequence. The Look Up Table (LUT) stores all possible partial products over the filter coefficient space.

Assuming coefficients c[n] is known constants, and then y[n] can be rewritten as follows:

$$y[n] = \Sigma\, c[n] \cdot x[n] \quad n = 0, 1, \ldots, N\text{-}1 \tag{1}$$

Variable x[n] can be represented by:

$$x[n] = \Sigma\, x_b\,[n] \cdot 2^b \qquad b=0, 1, \ldots, B\text{-}1$$
$$x_b[n] \in [0, 1] \tag{2}$$

Where $x_b\,[n]$ is the b^{th} bit of x[n] and B is the input width. Finally, the inner product can be rewritten as follows:

$$y = \Sigma\, c[n] \Sigma\, x_b\,[k] \cdot 2^b \tag{3}$$

$$= c[0]\,(x_{B-1}\,[0]2^{B-1} + x_{B-2}\,[0]\,2^{B-2} + \ldots + x_0\,[0]\,2^0)$$
$$+ c[1]\,(x_{B-1}\,[1]\,2^{B-1} + x\,x_{B-2}\,[1]\,2^{B-2} + \ldots + x_0\,[1]\,2^0) + \ldots$$
$$+ c[N\text{-}1]\,(x_{B-1}[N\text{-}1]\,2^{B-1} + x_{B-2}\,[0]\,2^{B-2} + \ldots + x_0\,[N\text{-}1]\,2^0) \tag{4}$$

$$= (c[0]\,x_{B-1}\,[0] + c[1]\,x_{B-1} + \ldots + c[N\text{-}1]\,x_{B-1}[N\text{-}1])\,2^{B-1} + (c[0]\,x_{B-2}[0]$$
$$+ c[1]\,x_{B-2}[1] + \ldots + c[N\text{-}1]\,x_{B-2}[N\text{-}1])\,2^{B-2} + \ldots + (c[0]\,x_0[0] +$$
$$c[1]\,x_0\,[1] + \ldots + c[N\text{-}1]\,x_0[N\text{-}1])\,2^0 \tag{5}$$

$$= \Sigma\; 2^b\, \Sigma\, c[n] \cdot x_b\,[k]$$

Where n=0, 1… N-1 and b=0, 1… B-1

The coefficients in most of DSP applications for the multiply accumulate operation are constants.

IV. FPGA

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Although one-time programmable (OTP) FPGAs are available, the dominant types are SRAM based which can be reprogrammed as the design evolves.

ASIC and FPGAs have different value propositions, and they must be carefully evaluated before choosing any one over the other. Information abounds that compares the two technologies. While FPGAs used to be selected for lower speed/complexity/volume designs in the past, today's FPGAs easily push the 500 MHz performance barrier. With unprecedented logic density increases and a host of other features, such as embedded processors, DSP blocks, clocking, and high-speed serial at ever lower price points, FPGAs are a compelling proposition for almost any type of design.

V. PROPOSED METHODOLOGY

The above technique holds good only when we go for lower order filters. For higher order filters, the size of the LUT also increases exponentially with the order of the filter. For a filter with N coefficients, the LUT have 2N values. This in turn reduces the performance. it to block transpose form type-I configuration of block FIR filter. The DFG-3 can be retimed to obtain the DFG-4 of Figure 1, which is referred to block transpose form type-II configuration.

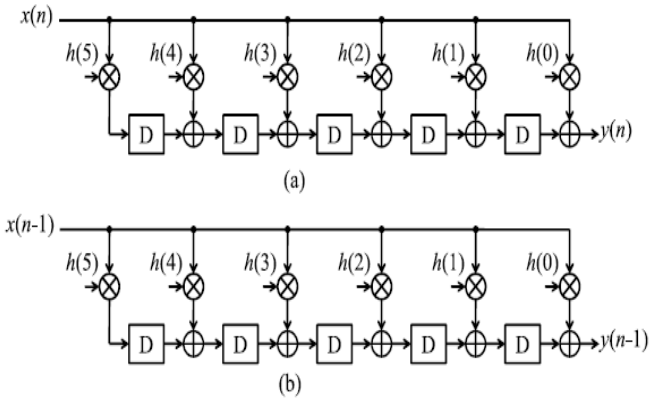

Figure 1: DFG of transpose form structure for $N = 6$. (a) DFG-1 for output $y(n)$. (b) DFG-2 for output $y(n-1)$.

Note that both type-I and type-II configurations involve the same number of multipliers and adders, but type-II configuration involves nearly L times less delay elements than those of type-I configuration. We have, therefore,

used block transpose form type-II configuration to derive the proposed structure. In Section II-C, we present mathematical formulation of block transpose form type-II FIR filter for a generalized formulation of the concept of block-based computation of transpose form FIR filers.

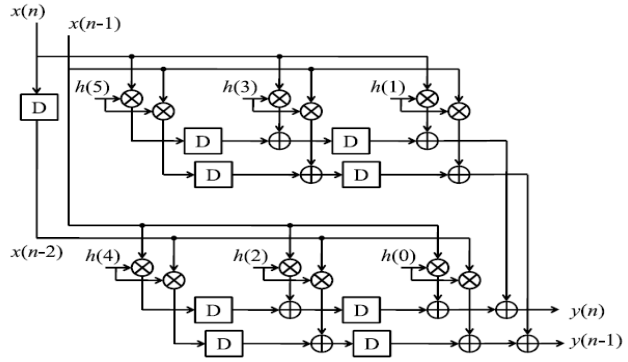

Figure 2: Merged DFG (DFG-3: transpose form type-I configuration for block FIR structure)

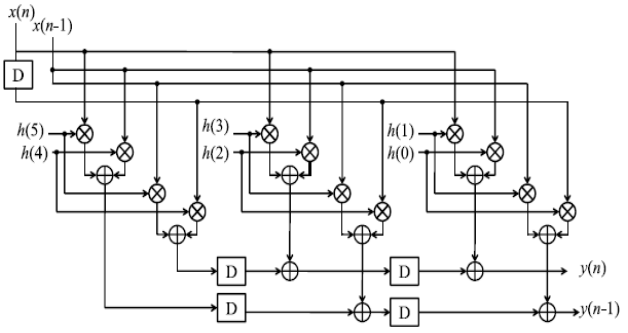

Figure 3: DFG-4 (retimed DFG-3) transpose form type-II configuration for block FIR structure

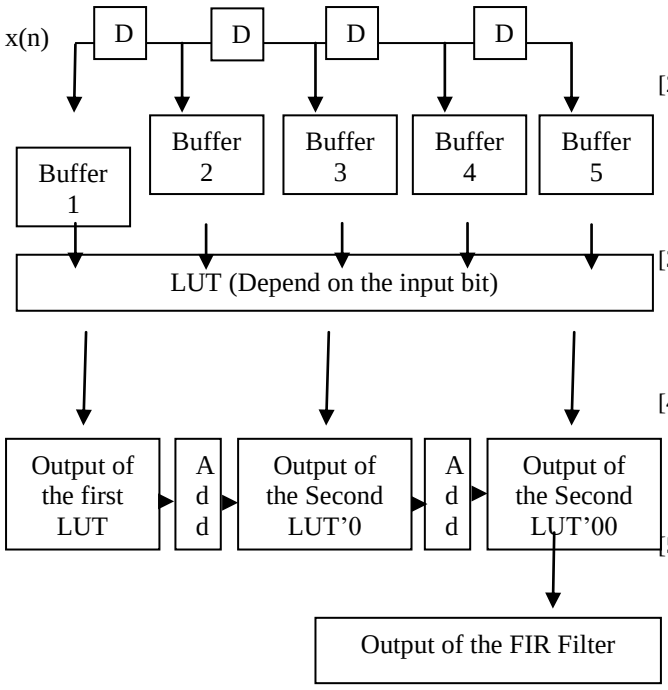

Figure 4: Block Diagram of Proposed Methodology

VI. CONCLUSION

In this paper, we have explored the possibility of realization of block FIR filters in transpose form configuration for area delay efficient realization of both fixed and reconfigurable applications. A generalized block formulation is presented for transpose form block FIR filter, and based on that we have derived transpose form block filter for reconfigurable applications.

Finite Impulse Response filter plays an important role in many Digital Signal Processing applications. In this method, the multiplier less FIR filter is implemented using Distributed Arithmetic which consists of Look Up Table and then partitioning is involved. This architecture provides an efficient area-time power implementation which involves significantly less latency and less area-delay complexity when compared with existing structures for FIR Filter.

VII. REFERENCES

[1] Usha Maddipati, Shaik Ahemedali, Maddipati Sri Sai Ramya, M D Praneeth Reddy and K N J Priya, "Comparative analysis of 16-tap FIR filter design using different adders", ICCCNT, IEEE 2020.

[2] Ankit Upadhyay and Prof. Uday Panwar, "High Performance VLSI Architecture for Transpose Form FIR Filter using Integrated Module", International Conference on Computer Communication and Informatics (ICCCI), IEEE 2018.

[1] Basant Kumar Mohanty, and Pramod Kumar Meher, High-Performance FIR Filter Architecture for Fixed and Reconfigurable Applications", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 78, No.06, April 2016.

[2] Indranil Hatai, Indrajit Chakrabarti, and Swapna Banerjee, "An Efficient VLSI Architecture of a Reconfigurable Pulse-Shaping FIR Interpolation Filter for Multi-standard DUC", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 23, No. 6, June 2015.

[3] Sang Yoon Park and Pramod Kumar Meher, "Efficient FPGA and ASIC Realizations of DA-Based Reconfigurable FIR Digital Filter", IEEE Transactions on Circuits And Systems-Ii: Express Briefs, 2014.

[4] B. K. Mohanty, P. K. Meher, S. Al-Maadeed, and A. Amira, "Memory footprint reduction for power-efficient realization of 2-D finite impulse response filters," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 1, pp. 120–133, Jan. 2014.

[5] B. K. Mohanty and P. K. Meher, "A high-performance energy-efficient architecture for FIR adaptive filter based on new distributed arithmetic formulation of block LMS algorithm," *IEEE Trans. Signal Process.*, vol. 61, no. 4, pp. 921–932, Feb. 2013.

[6] G. Gokhale and P. D. Bahirgonde, "Design of Vedic Multiplier using Area-Efficient Carry Select Adder", 4th IEEE International Conference on Advances in Computing,

Communications and Informatics (ICACCI-2015), Kochi, August 10-13, 2015, India.

[7] G. Gokhale and Mr. S. R. Gokhale, "Design of Area and Delay Efficient Vedic Multiplier Using Carry Select Adder", 4th IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI-2015), Kochi, August 10-13, 2015, India.

[8] Pavan Kumar, Saiprasad Goud A, and A Radhika had published their research with the title "FPGA Implementation of high speed 8-bit Vedic multiplier using barrel shifter", 978-1-4673-6150-7/13 IEEE.

[9] B. Madhu Latha1, B. Nageswar Rao, published their research with title "Design and Implementation of High Speed 8-Bit Vedic Multiplier on FPGA" International Journal of Advanced Research in Electrical ,Electronics and Instrumentation Engineering, Vol. 3, Issue 8, August 2014.

[10] A Murali, G Vijaya Padma, T Saritha, published their research with title "An Optimized Implementation of Vedic Multiplier Using Barrel Shifter in FPGA Technology", Journal of Innovative Engineering 2014, 2(2).

[11] Sweta Khatri, Ghanshyam Jangid, "FPGA Implementation of 64-bit fast multiplier using barrel shifter" Vol. 2 Issue VII, July 2014 ISSN: 2321-9653.

[12] Toni J. Billore, D. R. Rotake, "FPGA implementation of high speed 8 bit Vedic Multiplier using Fast adders" Journal of VLSI and Signal Processing, Volume 4, Issue 3, Ver. II (May-Jun. 2014), PP 54-59 e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197.

[13] S. S. Kerur, Prakash Narchi, Jayashree C N, Harish M Kittur and Girish V A, "Implementation of Vedic Multiplier for Digital Signal processing" International Conference on VLSI, Communication & Instrumentation (ICVCI) 2011.