2012 International Conference on Industrial and Intelligent Information (ICIII 2012)
IPCSIT vol.31 (2012) © (2012) IACSIT Press, Singapore

Area Efficient Pipelined Architecture For Realization of FIR Filter Using Distributed Arithmetic

V.Sudhakar[1+], N.S.Murthy [2], L.Anjaneyulu [3]

[1, 2, 3] Department of Electronics and Communication Engineering
National Institute of Technology, Warangal, Andhra Pradesh, 506004, India

Abstract. This paper presents the realization of area efficient architectures using Distributed Arithmetic (DA) for implementation of Finite Impulse Response (FIR) filter. The performance of the bit-serial and bit-parallel DA along with pipelining architecture with different quantized versions are analyzed for FIR filter design. Pipelined DA architecture has achieved double the maximum frequency of operation when compared to their non-pipelined implementations with an increase in hardware. Filters generated using 8 Bit fixed point implementation requires smaller area usage compared to 16 fixed point implementation at the cost of imprecision. The proposed implementations are synthesized with Xilinx ISE 13.2i

Keywords: Distributed Arithmetic (DA), Field programmable gate arrays (FPGA), Finite impulse response (FIR), look up table (LUT), Pipelined.

1. Introduction

Finite impulse response (FIR) digital filters are common components in many digital signal processing (DSP) systems [1]. Throughout the years, with the increasing development in very large scale integration (VLSI) technology, the real time realization of FIR filter with less hardware requirement and less latency has become more and more important. As the complexity of implementation increases with the length of filter, several algorithms have been mapped into effective architectures using ASIC's and FPGA's; one of them being distributed arithmetic (DA).

The main portion of DA-based computation is lookup table (LUT) that stores the precomputed values and can be read out easily, which makes DA-based computation well suited for FPGA realization, because the LUT is the basic component of FPGA. DA technique can be designed to meet various speed requirements, for example, it can be designed for high-speed implementation where all bits of one word are processed per clock, it can also be designed for medium-speed implementation where several bits of one word (not all bits) are processed per clock. Memory decomposition can reduce the memory significantly, but on the other side, it also leads to the increasing of latency and number of adders [2]. Presently high-speed and medium-speed realization of FIR filters with low latency and chip-area are in high demand. This paper addresses the issue of extending DA based FPGA implementation of FIR filter to an area-time optimized implementation.

This paper is organized as follows: Section 2 explores the main concept of DA technique and the basic DA solutions. The proposed DA architectures, FPGA implementation and simulation results are discussed in Section III. Conclusion is provided in Section IV.

2. Distributed Arithmetic (DA)

+ Corresponding author.
E-mail address: sudhakarvakiti@gmail.com.

DA is a bit-serial computational operation that forms an inner product of a pair of vectors in a single direct step. The advantage of DA is its efficiency of mechanization [3]. We consider the following sum of products:

$$Y = \sum_{k=1}^{K} A_k X_k \tag{1}$$

The A_k are fixed coefficients, and the X_k are the input data words. If each X_k is a 2's-complement binary number scaled (for convenience, not as necessity) such that mod (X_k) < 1, then we may express each X_k as

$$X_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn}\, 2^{-n} \tag{2}$$

Where the b_{kn} are the bits, 0 or 1, $b_{k0}$is the sign bit, and $b_{k,N-1}$ is the least significant bit (LSB).Combining Equations 1 and 2 in order to express y in terms of the bits of :

$$X_k = \sum_{k=1}^{K} \left[-b_{k0} + \sum_{n=1}^{N-1} b_{kn}\, 2^{-n} \right] \tag{3a}$$

Equation 3a is the conventional form of expressing the inner product. Direct mechanization of this equation defines a "lumped" arithmetic computation. Interchanging the order of the summations, we get

$$X_k = \sum_{n=1}^{N-1} \left[\sum_{k=1}^{K} A_k b_{kn} \right] 2^{-n} + \sum_{k=1}^{K} A_k (-b_{k0}) \tag{3b}$$

This is the crucial step: Equation 3b defines a distributed arithmetic computation. Consider the bracketed term in Equation 3b:

$$\left[\sum_{k=1}^{K} A_k b_{kn} \right] \tag{3c}$$

As each b_{kn} may take on values of 0 and 1 only, expression (3c) may have only 2k possible values. Rather than computing these values on line, they may be precomputed and stored in a ROM. The input data can be used to directly address the memory and the result, i.e., the $\sum_{k=1}^{K} A_k b_{kn}$ can be dropped into an accumulator. After N such cycles, the memory contains the result, Y as shown in Figure 1.

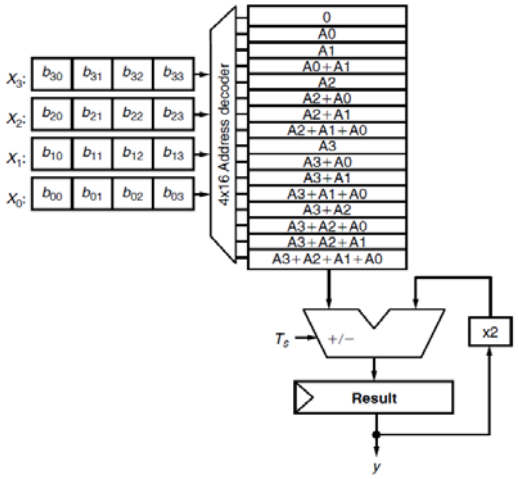

Figure1: Simple implementation of Distributed Arithmetic

Two approaches can be taken to improve DA performance on an FPGA platform. First employing parallelism and the other is pipelining as shown in Figures 2 and 3 respectively [5].

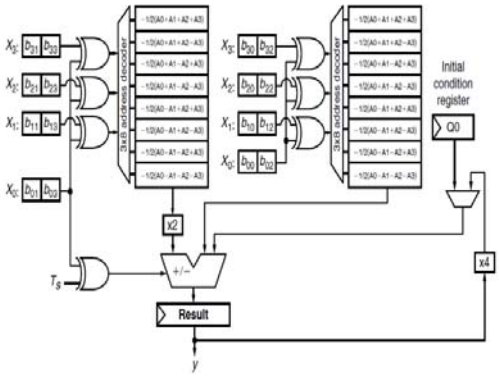

Figure2: Two-bit-at-a-time (2 BAAT) DA implementation. (Parallel Implementation)

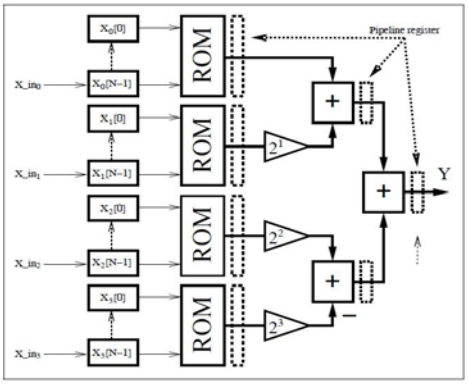

Figure3: Pipelined implementation of a distributed arithmetic FIR filter

3. Implementation

An FIR Low Pass filter is designed as per the specifications given (Table 1). Its performance characteristics are compared with different architectures.

Table 1: Filter Specifications

Parameter	Value
Sampling Frequency	48 KHz
Pass band Frequency	9.6 KHz
Stop band Frequency	11.5 KHz
Pass band Attenuation	1 dB
Stop band Attenuation	80 dB

A special class of FIR filter that is particularly effective in meeting such specifications is called the equiripple FIR filter [4]. An equiripple design protocol minimizes the maximal deviations (ripple error) from the ideal transfer function. The filer designed for the mentioned specifications using equiripple design method is of order 64.

Figure 4 shows the response the reference filter. Using the FDA Tool, the reference filter is quantized to Q16.14 and Q8.7 fixed point numeric representation format [4]. Figures 5 and 6 show the response of the filter after quantizing the coefficients to Q16.14 and Q8.7 formats respectively.

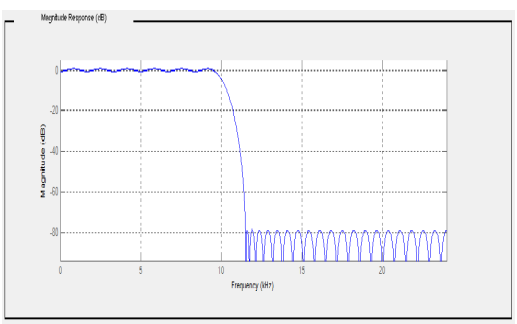

Figure4: Response of the reference filter

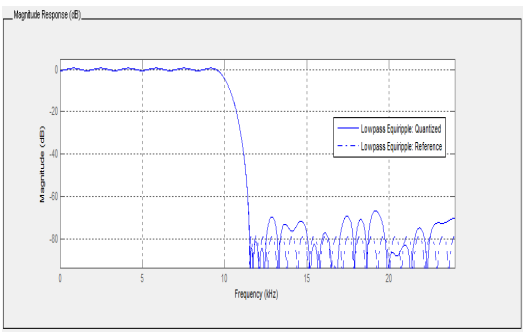

Figure5: FIR filter response for Q16.14 format

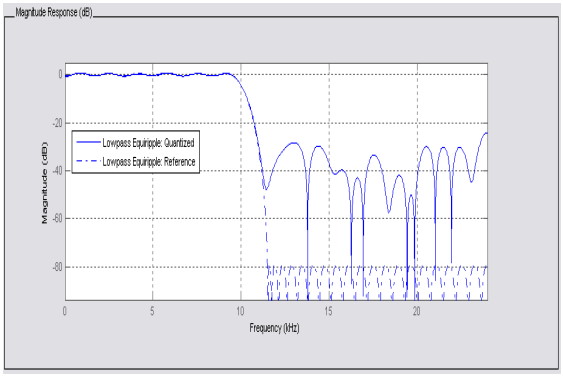

Figure6: FIR filter response for Q8.7 format

Ingesting data serially, One bit at a time (BAAT), results in a slow computation. If the input words are N bits in length, N clock cycles or periods are required in which to form the dot product. The number of clock cycles elapsed before an output is obtained is equal to the number of bits in the input data. This potentially limits the throughput of DA. To improve the throughput of DA, DA can be configured to process multiple bits by replicating LUTs and operate on multiple bits in parallel.

Pipelining transformation leads to a reduction in the critical path, which increases the speed of operation. Although there is a slight increase in system resources and latency this technique can be employed to increase the overall performance of the Filter. Comparison of the various performance parameters for 1-BAAT (Bit-at-a-time), 2-BAAT and their corresponding pipelined versions for Q16.14 fixed point implementation and Q8.7 fixed point implementation are shown in Figures 7 and 8 respectively.

When Compared to straight forward implementation based on MAC, DA based implementations have achieved higher speed and are shown in Figure 9 and 10. Simulation results for impulse input are shown in Figure 11 and 12.

4. Conclusion

Designs of pipelined architectures are derived for high speed implementation of FIR filter. The pipelined architectures are implemented on Xilinx Spartan3E FPGA. The pipelined architecture has achieved maximum frequency of operation of 357 MHz whereas for conventional MAC based design has achieved only 11.47 MHz for Q16.14 fixed point representation based implementation. The maximum frequency of operation for Q8.7 fixed point representation based implementation in the case of pipelined architecture is 129 MHz whereas conventional MAC based design it is 10 MHz. Filters implemented with Q8.7 representation results in nearly 53% lesser slice LUT's when compared to Q16.14 representation.

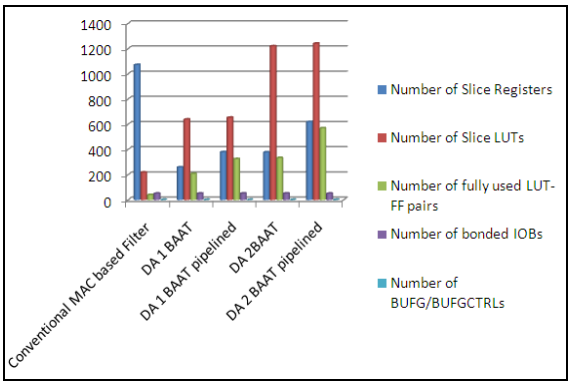

Figure7: Resource utilization for Q16.14 Implementation

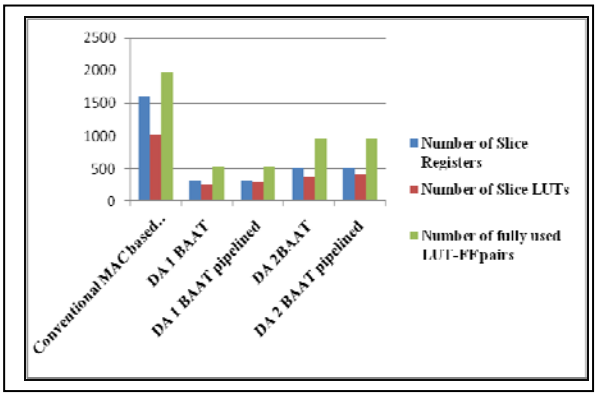

Figure8: Resource utilization for Q8.7 Implementation

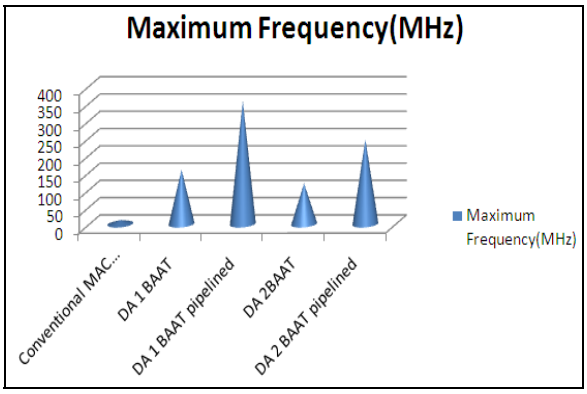

Figure9: Maximum Frequency of operation of different architectures for Q16.14 implementation

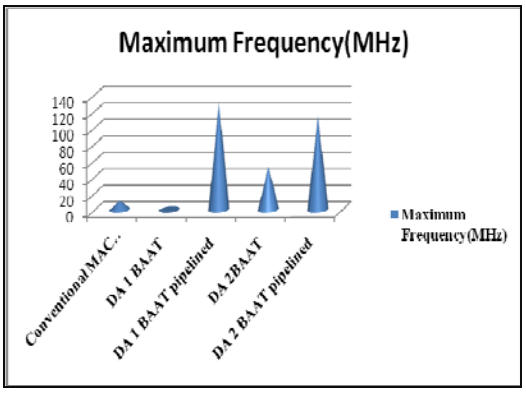

Figure 10: Maximum Frequency of operation of different architectures for Q8.7 implementation

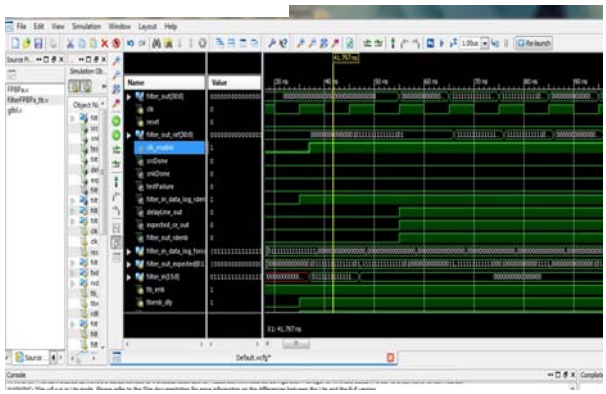

Figure11: Simulation results for Q16.14 FIR filter

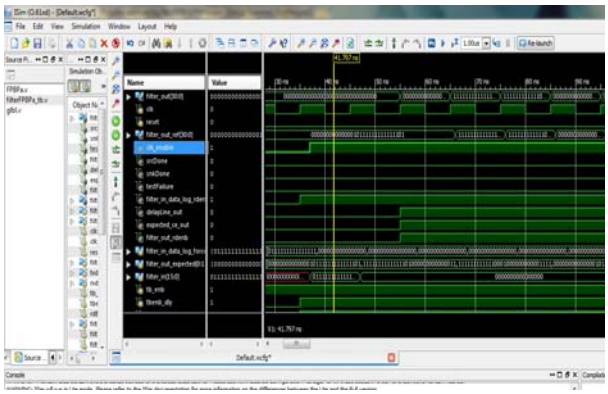

Figure12: Simulation results for Q8.7 FIR filter

5. References

[1] A. Antonion, Digital Filters: Analysis, Design, and Applications, McGraw-Hill, New York, 1993.

[2] P.K. Meher, S. Chandrasekaran, A. Amira, FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic, IEEE Transactions on Signal Processing 56 (7) (2008) 3009–3017.

[3] U. Meyer-Baese, Digital Signal Processing with Field Programmable Gate Arrays, Springer

[4] Shanthala S, and S. Y. Kulkarni, "High Speed and Low Power FPGA Implementation of FIR Filter for DSP Applications" European Journal of Scientific Research, 2009

[5] Stanley A.White "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review" IEEE Acoustic speech signal processing Magazine, July 1989