# AREA EFFICIENT RECONFIGURABLE COEFFICIENT BASED FIR FILTER

J.Chaitanya[#], Ch.Venkateswarlu*, P.Sunitha[*]

*ECE Department Pragati Engineering*

*College, Surampalem, India*

## Abstract

***The implementation of FIR filters on FPGA based on traditional method cost considerable hardware resources, which goes against the decrease of circuit scale and the increase of system speed. A new design and implementation of FIR filters using Distributed Arithmetic and Dynamic Distributed Arithmetic is provided in this paper to solve this problem. Distributed Arithmetic structure is used to increase the resource usage while pipeline structure is also used to increase the system speed. The Proposed work in my paper is Dynamic Distributed Arithmetic algorithm is suitable for designing digital filter with varying co-efficient as compared to the conventional Distributed arithmetic algorithm. The extensive use of a Dynamic Distributed Arithmetic algorithm eliminates the multiplier unit which requires more number of adders. Xilinx Spartan III devices are used for optimization.***

**Keywords:** Distributed Arithmetic; DDA; FIR; pipeline; LUT; FPGA

## I. Introduction

Digital filters are the essential units for digital signal processing systems. Traditionally, digital filters are achieved in Digital Signal Processor (DSP), but DSP-based solution cannot meet the high speed requirements in some applications for its sequential structure. Nowadays, Field Programmable Gate Array (FPGA) technology is widely used in digital signal processing area because FPGA-based solution can achieve high speed due to its parallel structure and configurable logic, which provides great flexibility and high reliability in the course of design and later maintenance. In general, Digital filters are divided into two categories, including Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). And FIR filters are widely applied to a variety of digital signal processing areas for the virtues of providing linear phase and system stability. The FPGA-based FIR filters using traditional direct arithmetic costs considerable multiply-and-accumulate (MAC) blocks with the augment of the filter order. However, according to Distributed Arithmetic, we can make a Look-Up-Table (LUT) to conserve the MAC values and callout the values according to the input data if necessary. Therefore, LUT can be created to take the place of MAC units so as to save the hardware resources. This paper provide the principles of Distributed Arithmetic, and introduce it into the FIR filters design, and then presents a 40-order FIR low-pass filter using Distributed Arithmetic, which save considerable MAC blocks to decrease the circuit scale, meanwhile, divided LUT method is used to decrease the required memory units and pipeline structure is also used to increase the system speed. The extensive use of a dynamic distributed algorithm eliminates the multiplier unit which requires more number of adders**.**

### II. DISTRIBUTED ARITHMETIC

Distributed Arithmetic was first brought up by Crosier [1], and was extended to cover the signed data system by Liu, and then was introduced into FPGA design to save MAC blocks with the development of FPGA

The N-length FIR filter can be described as:

$$Y = \langle h, x \rangle = \sum_{n=0}^{N-1} h[n]\, x[n] \qquad (1)$$

Where h[n] is the filter coefficient and x[n] is the input sequence to be processed. The FIR structure consists of a series of multiplication and addition units, and consumes N MAC blocks of FPGA, which are expensive in high speed system. Compared with traditional direct arithmetic, Distributed Arithmetic can save considerable hardware resources through using LUT to take the place of MAC units [2]. Another virtue of this method is that it can avoid system speed decrease with the increase of the input data bit width or the filter coefficient bit width, which can occur in traditional direct method and consume considerable hardware resources [3].

Distributed Arithmetic is introduced into the design of FIR filters as follows.

In the two's complement system, x[n] can be described as:

$$X[n] = -2^B x_B[n] + \sum_{b=0}^{B-1} 2^b x_b[n] \qquad (2)$$

Substituting equ (2) into equ (1) Yields:

$$Y= -2^B x_B[n]h[n] + \sum_{b=0}^{B-1} h[n] \sum_{n=0}^{N-1} 2^b x_b[n] \qquad (3)$$

The second part of the equ (3) Can be changed into another form:

$$\sum_{b=0}^{B-1} h[n] \sum_{n=0}^{N-1} 2^b x_b[n] = \sum_{b=0}^{B-1} 2^b \sum_{n=0}^{N-1} h[n]\overline{x}_b[n]$$

(4)

Substituting equ (4) into equ (3) Yields to the final form of Distributed Arithmetic:

$$Y = -2^B x_B[n]h[n] + \sum_{b=0}^{B-1} 2^b \sum_{n=0}^{N-1} h[n]\overline{x}_b[n] \qquad (5)$$

Take a close look at the right part of equ. (5), considering the limited possibility of input data; we can conserve the values of $\sum_{n=0}^{N-1} h[n]\overline{x}[n]$ into a LUT unit and then callout the relevant value according to the input data to save MAC blocks [4]. And then the weighted sum of $\sum_{n=0}^{N-1} h[n]\overline{x}[n]$ is calculated through shift registers, the result is $\sum_{b=0}^{B-1} 2^b \sum_{n=0}^{N-1} h[n]\overline{x}_b[n]$ .In signed system, the signed bit should be taken into consideration so 2B [ ] [ ] B − x n h n is also added. As a result, the final form of Distributed Arithmetic is defined as equ. (5) And the implementation can be achieved on FPGA through LUT units. As the expatiation above, the basic Distributed Arithmetic structure can be described as Fig.1. The dotted rectangle is the register. Fig.

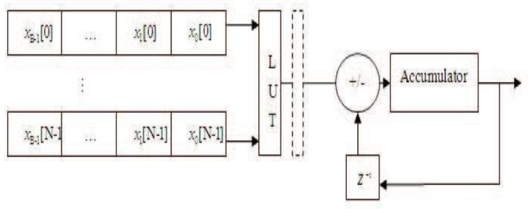
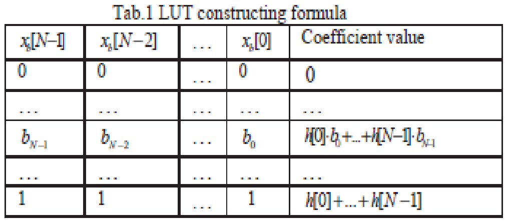


Fig.1 The basic Distributed Arithmetic structure

According to the input sequence, we can conserve the coefficient values in LUT unit, the LUT constructing formula is given in Tab.1.

Tab.1 LUT constructing formula

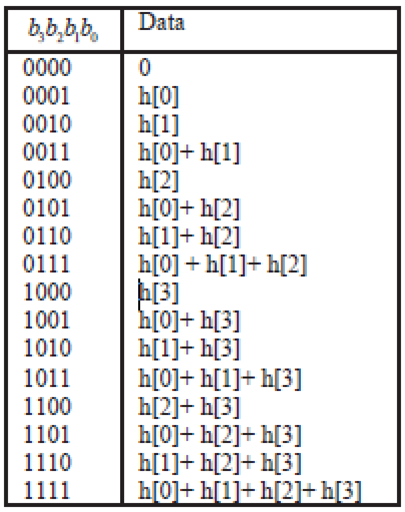


| $x_b[N-1]$ | $x_b[N-2]$ | ... | $x_b[0]$ | Coefficient value |
|---|---|---|---|---|
| 0 | 0 | ... | 0 | 0 |
| ... | ... | ... | ... | ... |
| $b_{N-1}$ | $b_{N-2}$ | ... | $b_0$ | $h[0]\cdot b_0+\ldots+h[N-1]\cdot b_{N-1}$ |
| ... | ... | ... | ... | ... |
| 1 | 1 | ... | 1 | $h[0]+\ldots+h[N-1]$ |

We can achieve the filter on FPGA according to Fig.1 and Tab.1. However, considering the even symmetry of the coefficients, we can use equ. (6) To simplify the system [6]. And the final value is definite as the input of shift registers.
y[i]=x[i]+ x[31-i] □ i=0,1...15 (6)

However, with the increase of filter order, the scale of LUT will increase dramatically [7], which will cost more time to look up the table and more memory to store the values. Therefore, we can divide the LUT unit into four small LUT units to solve this problem [8]. Then the values of the four divided LUT units are added as the final value. Coefficient values of small LUT is given in Tab.2.

Tab.2 Coefficient values of LUT



| $b_3b_2b_1b_0$ | Data |
|---|---|
| 0000 | 0 |
| 0001 | h[0] |
| 0010 | h[1] |
| 0011 | h[0]+ h[1] |
| 0100 | h[2] |
| 0101 | h[0]+ h[2] |
| 0110 | h[1]+ h[2] |
| 0111 | h[0] + h[1]+ h[2] |
| 1000 | h[3] |
| 1001 | h[0]+ h[3] |
| 1010 | h[1]+ h[3] |
| 1011 | h[0]+ h[1]+ h[3] |
| 1100 | h[2]+ h[3] |
| 1101 | h[0]+ h[2]+ h[3] |
| 1110 | h[1]+ h[2]+ h[3] |
| 1111 | h[0]+ h[1]+ h[2]+ h[3] |

Pipeline structure is also used to increase the system speed. The pipelining technology is to divide combinational circuit into small parts, and then insert a register in the middle of the two parts to increase the system speed [9]. The filter designed in this paper contains 3 level registers. Although it will increase the time delay, but helps to increase the system speed [10]. Considering all the factors above, we achieve the new structure based on Distributed Arithmetic as Fig.2**.**

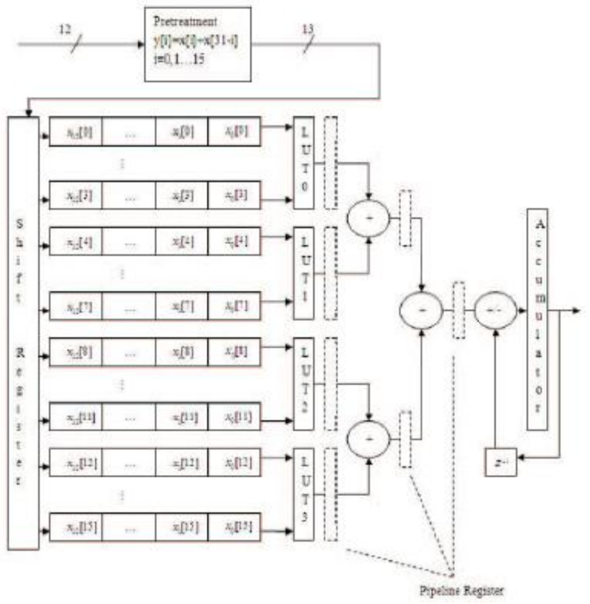


Fig.2 Structure of FIR filter based on Distributed Arithmetic

## III.DYNAMIC DISTRIBUTED ARITHMETIC:

The output of an N tap FIR filter, which is the convolution of the latest L input samples, is given in Equ 1.1 is the number of coefficients h(k) of the filter and x(n) represents the input time series.

Y[n] = ∑h[k] X [n-k] k = 0, 1… N-1 (6)

The conventional tapped delay line realization of this inner product is shown in figure 3.

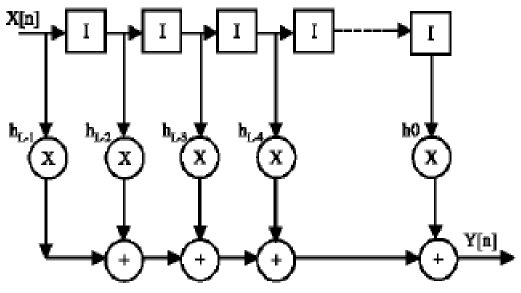


Fig 3: L- Tap FIR filter

The implementation translates to L multiplications and L-1 additions per sample to compute the result. This can be implemented using a single Multiply Accumulate (MAC) engine, but it would require L-MAC cycles, before the next input sample can be processed. A general purpose multiplier occupies a large area on FPGAs. Since all the multiplications are with constants, the full flexibility of a general purpose multiplier is not required and the area can be vastly reduced using techniques developed for constant multiplication like distributed algorithm. Though most of the current generation FPGAs such as Xilinx Spartan III have embedded multipliers to handle these multiplications, the number of these multipliers is typically limited. The ideal implementation would involve a sharing of the Combinational logic Blocks (CLBs) and these multipliers.

In this research, we present a Dynamic Distributed Arithmetic (DDA) technique that is better than conventional techniques for implementation on the CLBs. DDA which is well known method to save resources like slices, flip flops and LUTs. The DDA architecture make extensive use of look - up tables, which make them ideal for implementing digital signal processing functions on Xilinx FPGAs, whose architectures are based on look-up tables. Moreover, distributed architectures are suitable for low power portable applications, because they replace the costly multipliers with shifts and look-up tables.

The DDA is suitable for designing digital FIR filter with varying co-efficient as compared to the conventional Distributed Arithmetic Algorithm based Fir filter design where the filter coefficient are constant. Whenever there is a change in filter co-efficient buffer updated dynamically and then performs the defined operations. In this research, the input sequence is fed into the input buffer register at the input sample rate. The co-efficient are also fed to the corresponding buffer. The LUTs are updated whenever there is a change in C (n) values through buffer. The serial output is presented to the RAM based shift registers. The RAM based shift registers stores the data in a particular address. The outputs of registered LUTs are added and loaded to the scaling accumulator from LSB to MSB and the result which is the filter output will be accumulated on to the output register over the time. For an n bit input, n+1 clock cycles are needed for a symmetric filter to generate the output, if there is any change in h[n], it will be updated and the resultant content is stored in the LUTs as shown in fig 4.

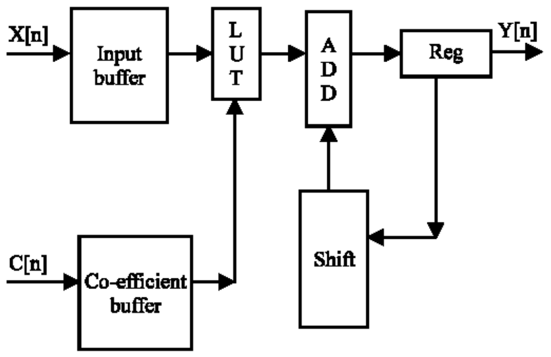


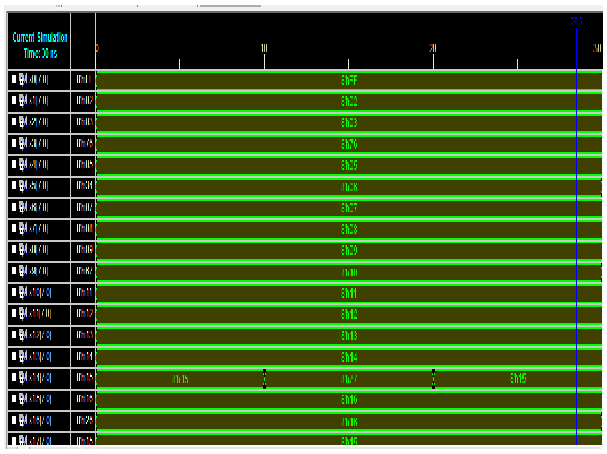
Fig 4: Block Diagram of Dynamic Distributed Algorithm based FIR filter.

Xilinx Spartan III are programmed using Verilog HDL; a popular hardware description Language. The language has capabilities to describe the behavioural nature of design, the data flow of design, a design’s structural composition, delays and a waveform generation mechanism. Models written in this language can be verified using a Verilog simulator. As a programming and development environment, Xilinx ISE

Foundation Series tools have been used to produce a physical implementation for the Xilinx Spartan III.

## IV.RESULTS:

Xilinx ISE is used as simulation platform. After file synthesis with the Xilinx XST tool and the simulation analysis shown below. These results reflect the hardware resources consumption and timing performance of the FIR Filter.



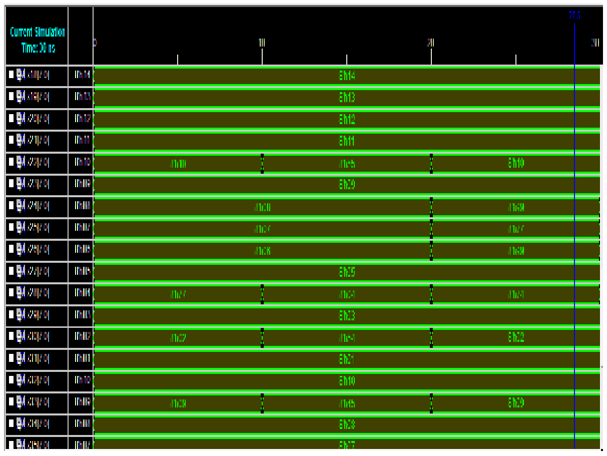
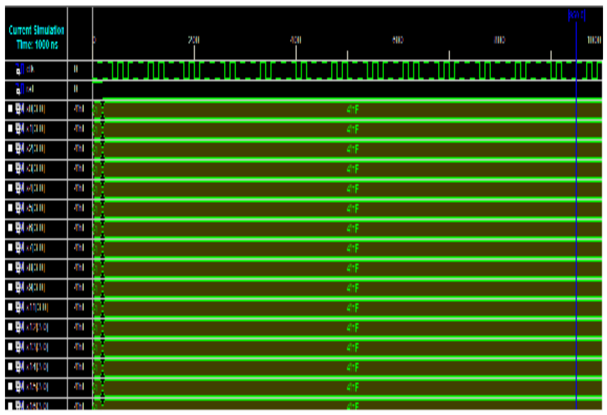


Fig: Simulation Results for Distributed Arithmetic



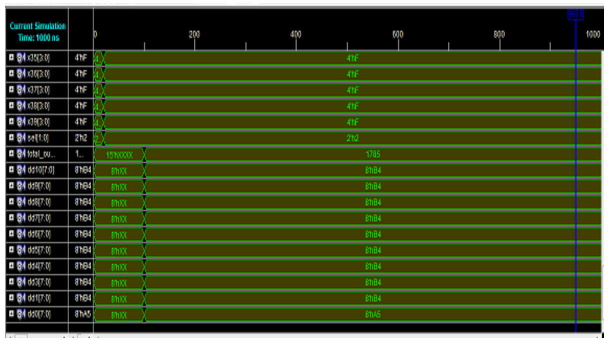


Fig: Simulation Results For Dynamic Distributed Arithmetic.

## V. CONCLUSION:

This paper presents the design and implementation based on Distributed Arithmetic and Dynamic Distributed Arithmetic is used to realize a 40-order FIR low-pass filter. Distributed Arithmetic structure is used to increase the resource usage while pipeline structure is used to increase the system speed. The test results indicate that the designed filter using Distributed Arithmetic can work stable with high speed and can save almost 50 per cent hardware resources. Meanwhile, it is very easy to transplant the filter to other applications through modifying the order parameter or bit width and other parameters, and therefore have great practical applications in digit signal processing. The extensive use of a dynamic distributed algorithm eliminates the multiplier unit which requires more number of adders.

## ACKNOWLEDGMENT

Thanks to all the friends engaging in the discus of the relevant subjects.

## REFERENCES

[1] Uwe Meyer-Baese.Digital signal processing with FPGA[M]. Beijing:Tsinghua University Press,2006:50~51
[2] Tsao Y C and Choi K. Area-Efficient Parallel FIR Digital Filter Structures for Symmetric Convolutions Based on Fast FIR Algorithm [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2010,PP(99):1~5.

[3] Chao Cheng and Keshab K Parhi. Low-Cost Parallel FIR Filter Structures With 2-Stage Parallelism[J].IEEE Transactions on Circuits and Systems I: Regular ,2007,54(2):280~290.
[4] Tearney G J and Bouma B E. Real-Time FPGA Processing for High-Speed Optical Frequency Domain Imaging [J]. IEEE Transactions on Medical Imaging,2009 ,28(9):1468~1472.
[5] Hu Guang-shu. Digital signal processing-theory,algorithm and realizes[M]. 2nd ed.Beijing: Tsinghua University Press,2003:296~307.
[6] Chun Hok Ho,Chi Wai Yu and Leong P. Floating-Point FPGA: Architecture and Modeling [J]. IEEE Transactions on Very Large Scale Integration Systems, 2008,17(12): 1709~1718.
[7] Evans J B. Efficient FIR filter architectures suitable for FPGA implementation [J].IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 2002,41(7):490~493.
[8] Meher P K , Chandrasekaran S and Amira A. FPGA Realization of FIR Filters by Efficient and Flexible Systolization Using Distributed Arithmetic [J]. IEEE Transactions on Signal Processing, 2008,56(7): 3009~3017.
[9] Xia Yu-wen. Digital system design with Verilog[M]. 2nd ed.Beijing:Higher Education Press,2008:102~103.
[10] Sungwook Yu and Swartziander E E. DCT implementation with distributed arithmetic[J]. IEEE Transactions on Computers, 2001,50(9):985~991.