

Memory Efficient Architecture For High Speed Fir Filter Using Distributed Arithmetic

N.V.Sai Rupa, M.Ram Mohan Reddy

Abstract— This paper presents the realization of memory efficient architecture using Distributed Arithmetic (DA) for implementation of Finite Impulse Response (FIR) filter. Here, the multipliers in FIR filter are replaced with multiplier less DA based technique. First, the theory of DA is described. In this technique, pre-computed values of inner product are stored in LUT, which are further added and shifted with number of iterations equal to the precision of input samples. But the exponential growth of LUT with the order of FIR filter, in its basic structure, makes it prohibitive for many applications. An improvement over the basic DA structure is presented in this paper, by the use of partitioning of LUT to the desired length. Architecture of 32 tap FIR filter is presented, with different length of partition of LUT. Design implementation and synthesis result shown the improvement in speed of operation as well as saving in memory, with more number of partitions. The proposed architecture provides an efficient memory-time-power implementation which involves significantly less latency and less area-delay complexity when compared with existing structures for FIR Filter.

Keywords— Distributed Arithmetic (DA), Field programmable gate arrays (FPGA), Finite Impulse response (FIR), Look Up Table (LUT), Partition.



1 INTRODUCTION

In the recent years, there has been a growing trend to implement DSP functions in Field Programmable Gate Arrays (FPGAs), which offer a balanced solution in comparison with traditional devices. Although application Specific Integrated Circuits (ASICs) and digital signal processors have been the traditional solution for high performance applications, now the technology and the market are imposing new rules. On one hand, high development costs and time-to-market factors associated with ASICs can be prohibitive for certain applications and, on the other hand, programmable DSP processors can be unable to reach a desired performance due to their sequential-execution architecture. In this context, FPGAs offer a very attractive solution that balance high flexibility, time-to-market, cost and performance. In that sense, the research community has put great effort in designing efficient architectures for DSP functions such as digital filters, which are extensively used in multiple applications in digital communications, speech processing, wireless/satellite communications, biomedical signal processing and many others[1], [2], [3].

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 FIR digital filter is represented as:

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

Where $y[n]$ is the FIR filter output, $x[n-k]$ is input data and $h[k]$ represents filter coefficients. Traditionally, direct implementation of a N-tap FIR filter requires N multiply-and-accumulate (MAC) blocks. Multiplication is strongest operation because it is repeated addition. It requires large portion of chip area. Power consumption is more. Memory-based structures are more regular compared with the MAC structures; and have many other advantages, e.g., greater potential for high throughput and reduced-latency implementation and are expected to have less dynamic power consumption due to less switching activities for memory-read operations compared to the conventional multipliers. Memory based structures are well-suited for many DSP algorithms, which involve multiplication with a fixed set of coefficients. For this Distributed Arithmetic architecture is used in FIR filter.

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

- **N.V.Sai Rupa**, PG Scholar, Department of Electronics and Communication Engineering, PBR Visvodaya Institute of Technology & Sciences, Kavali, Andhra Pradesh, India; sairupa.nv@gmail.com
- **M.Ram Mohan Reddy**, Professor, Department of Electronics and Communication Engineering, PBR Visvodaya Institute of Technology & Sciences, Kavali, Andhra Pradesh, India; ramu7772@yahoo.com

Arithmetic, and introduce it into the FIR filters design, and then presents a 31-order FIR low-pass filter using Distributed Arithmetic, which save considerable MAC blocks to decrease the circuit scale, meanwhile, partitioned LUT method is used to decrease the required memory units and pipeline structure is also used to increase the system speed

2 METHODOLOGY

Distributed Arithmetic was first brought up by Crosier [4], 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 technology.

The N-length FIR filter can be described as

$$y = \langle hx \rangle = \sum_{n=0}^{N-1} h[n]x[n] \tag{2}$$

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

$$x[n] = -2^B x_B[n] + \sum_{b=0}^{B-1} 2^b x_b[n] \tag{3}$$

Substituting eq.(3) into eq.(2) yields

$$x[n] = -2^B \sum_{n=0}^{N-1} x_B[n]h[n] + \sum_{n=0}^{N-1} h[n] \sum_{b=0}^{B-1} 2^b x_b[n] \tag{4}$$

The second part of the eq. (3) can be changed into another form:

$$\sum_{n=0}^{N-1} h[n] \sum_{b=0}^{B-1} 2^b x_b[n] = \sum_{b=0}^{B-1} 2^b \sum_{n=0}^{N-1} h[n]x_b[n] \tag{5}$$

Substituting eq.(4) into eq. (3) yields to the final form of Distributed Arithmetic

$$y = -2^B \sum_{n=0}^{N-1} x_B[n]h[n] + \sum_{b=0}^{B-1} 2^b \sum_{n=0}^{N-1} h[n]x_b[n] \tag{6}$$

Take a close look at the right part of eq. (6), considering the limited possibility of input data, we can conserve the values of $\sum_{n=0}^{N-1} h[n]x_b[n]$ 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 $\sum_{n=0}^{N-1} h[n]x_b[n]$ is calculated through shift registers, the result is $\sum_{b=0}^{B-1} 2^b \sum_{n=0}^{N-1} h[n]x_b[n]$. In signed system, the signed bit should be taken into consideration so $-2^B \sum_{n=0}^{N-1} x_B[n]h[n]$ is also added. As a result, the final form of Distributed Arithmetic is defined as Eq. (6) 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.

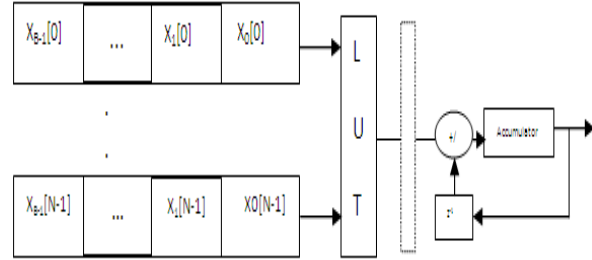


Fig.1 The basic Distributed Arithmetic structure

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

Table.1 LUT Constructing Formula

$x_b[N-1]$	$x_b[N-2]$...	$x_b[0]$	Coefficient Value
0	0	...	0	0
...
$bN-1$	$bN-2$...	$b0$	$h[0].b0 + \dots + h[N-1].bN-1$
...
1	1	...	1	$h[0] + \dots + h[N-1]$

With above structure and coefficient values of LUT, we can achieve a variety of filters to meet various requirements. Original LUT-based DA implementation of a 4 - tap (N=4) FIR filter is shown below in fig.2 the DA architecture includes three units: the shift register unit, the adder/shifter unit.

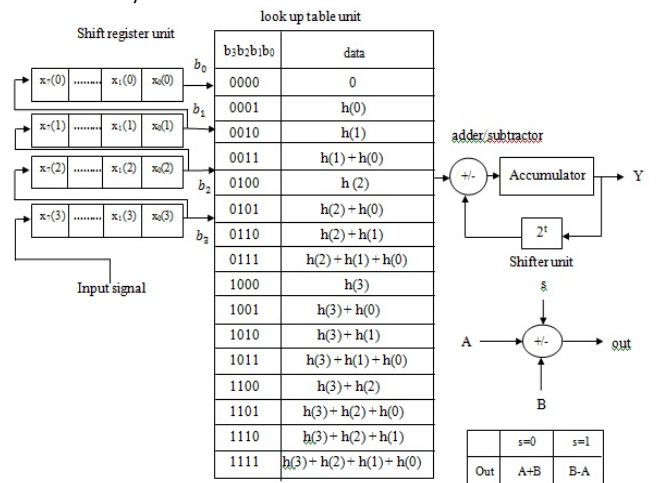


Fig.2 Original LUT-based DA Implementation of 4-tap Filter

3 PROPOSED ARCHITECTURE

3.1 Proposed DA-LUT unit:

In fig.2, we can see that the lower half of LUT (locations where $b_3 = 1$) is the same with the sum of the upper half of LUT (locations where $b_3 = 0$) and $h[3]$. Hence, LUT size can be reduced $\frac{1}{2}$ with an additional 2X1 multiplexer

and a full adder, as shown in fig.3.

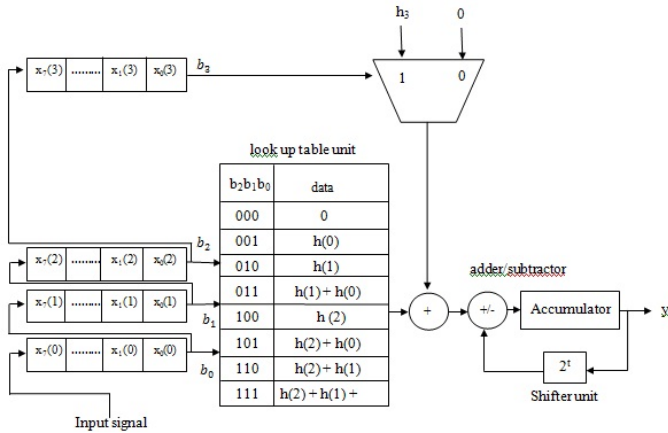


Fig.3 Proposed DA architecture for a 4-tap FIR filter (2³ - word LUT implementation)

3.2 Proposed DA architecture for high speed higher order FIR filters:

In the course of FIR filters design, Ringling’s can be generated at the edge of transition band for the reason that finite series Fourier transform cannot produce sharp edges [5]. So windows are often used to produce suitable transition band, and Kaiser Window is widely used for providing good performance. The parameter is an important coefficient of Kaiser Window which involves the windows types. We can get a variety of windows like Rectangular window, Hamming window, Hamming window, and Blackman window with the adjustment of. A 31-order FIR low-pass filter is designed using Kaiser Window, and the parameter is as follows: $\beta=3.39$, $w=0.18$. We can obtain the filter coefficients using Mat lab as follows.

- $h(0)=h(31)=0.0019; h(1)=h(30)=0.0043;$
- $h(2)=h(29)=0.0062; h(3)=h(28)=0.0061;$
- $h(4)=h(27)=0.0025; h(5)=h(26)=0.0050;$
- $h(6)=h(25)=0.0148; h(7)=h(24)=0.0236;$
- $h(8)=h(23)=0.0266; h(9)=h(22)=0.0192;$
- $h(10)=h(21)=0.0015; h(11)=h(20)=0.0351;$
- $h(12)=h(19)=0.0774; h(13)=h(18)=0.1208;$
- $h(14)=h(17)=0.1566; h(15)=h(16)=0.1768;$

In Mat lab data is described in the floating-point form while described in the fixed-point form in this FPGA system. After quantizing the filter coefficients using 12-bit-width signed binary [6], we can obtain the final coefficients as follows:

- $h(0)=h(31)=4; h(1)=h(30)=9; h(2)=h(29)=13;$
- $h(3)=h(28)=12; h(4)=h(27)=5; h(5)=h(26)=-10;$
- $h(6)=h(25)=30; h(7)=h(24)=-48; h(8)=h(23)=-55;$
- $h(9)=h(22)=-39; h(10)=h(21)=3; h(11)=h(20)=72;$
- $h(12)=h(19)=158; h(13)=h(18)=247; h(14)=h(17)=321;$
- $h(15)=h(16)=362;$

With above coefficients in Xilinx, the frequency amplitude characteristics for the filter are described in Fig.4.

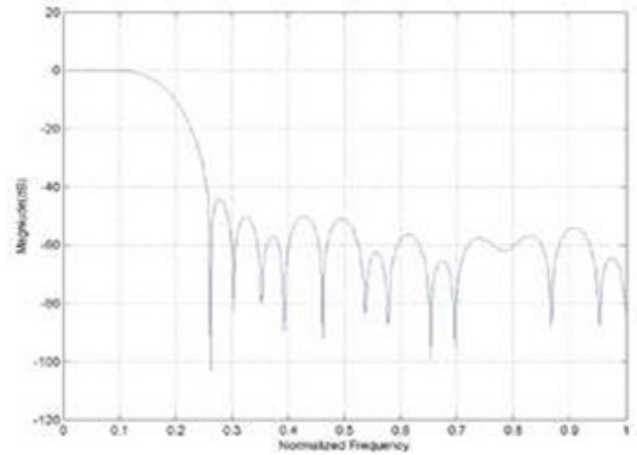


FIG:4 CHARACTERISTICS OF FIR FILTER

From Fig.4 we can observe that the low-pass filter has a good performance for low-pass filtering, and the cut-off frequency is 2.88 MHz if the sampling frequency is defined as 32 MHz’s 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 eq. (7) to simplify the system and the final values are defined as the input of shift registers.

$$P[i]=x[i]+x[31-i], \text{ for } i=0,1...15. \tag{7}$$

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 k coefficients, the LUT have 2^k values. This in turn reduces the performance. Therefore, for higher order filters, LUT size to be reduced to reasonable levels. To reduce the size, the LUT can be sub divided into a number of LUTs, LUT partitions. Each LUT partition operates on different set of filter taps. The results obtained from the partitions are summed. For 32-tap FIR low-pass filter, LUT is divided into four small LUT units. Coefficient values of small LUT is given in Table 2.

Table.2 Coefficient Values of LUT

b ₃ b ₂ b ₁ b ₀	Data
000	0
001	h[0]
010	h[1]
011	h[0]+h[1]
100	h[2]
101	h[0]+h[2]
110	h[1]+ h[2]
111	h[0]+ h[1]+ h[2]

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 [7]. The

filter designed in this paper contains 3 level registers. Although it will increase the time delay, but helps to increase the system speed[8].

Considering all the factors above, we achieve the new structure based on Distributed Arithmetic as Fig 5.

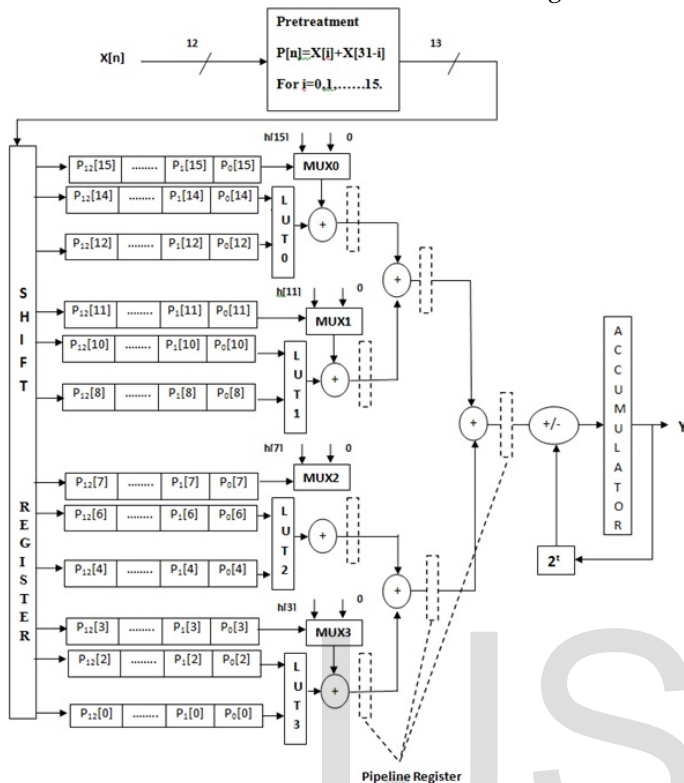


Fig: 5 Proposed Architecture Of 32 – Tap Fir Filter Based On Distributed Arithmetic

4 RESULTS

Xilinx is used as the simulation platform. We can analysis the changes between the input wave and the output wave to observe the permanence of the designed filter through Xilinx, while observing the real-time implementation performance of FPGA through Xilinx. In Fig.6, the waveforms are in the Xilinx Model. By using the basic formula we are assigning the values. As we have said MAC i.e.by multiplying and adding the corresponding coefficient values the final resultant value of fir filter is shown. We can conclude that the filter coefficients are suitable for the test.

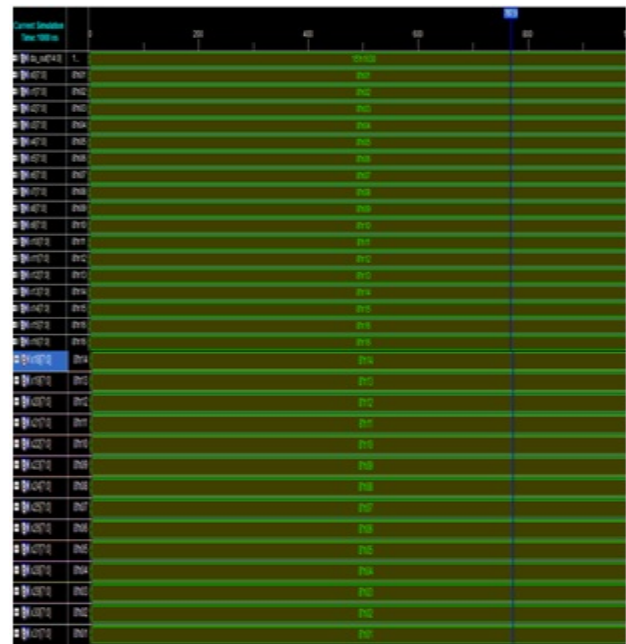


Fig: 6 simulation waveforms for FIR filter

5 CONCLUSION

This paper presents the design and implementation based on Distributed Arithmetic, which is used to realize a 31-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 percent 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 digital signal processing. Further this FIR filter has many applications in audio, video signal processing, Digital Signal Processing.

REFERENCES

- [1] J. B. Evans, "Efficient FIR Filter Architectures Suitable for FPGA Implementation," IEEE International Symposium on Circuits and Systems (ISCAS) '93, pp.152-156.
- [2] K. Kwang-Il Yeon, Han-Cheol Jo, Jong-Wha Chong, K. Kim, "Multiple 1:N Interpolation FIR Filter Design Based on a Single Architecture" IEEE International Symposium on Circuits and Systems(ISCAS) '98, pp. 316- 319
- [3] A.T. Erdogan and T. Arslan, "High Throughput Fir Filter Design For Low Power Soc Applications," IEEE International ASIC / SoC Conference2000, pp. 374-378
- [4] Uwe Meyer-Baese. Digital signal processing with FPGA [M]. Beijing: Tsinghua University Press, 2006:50-51
- [5] Hu Guang-shu. Digital signal processing-theory, algorithm and real-izes [M]. 2nd ed. Beijing: Tsinghua University Press, 2003:296-307.
- [6] Chun Hok Ho, Chi Wail 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] Xia Yu-wen. Digital system design with Verilog [M]. 2nd ed. Beijing: Higher Education Press, 2008:102-103.

- [8] Sungwook Yu and Swartzlander EE. DCT implementation with distributed s. *IEEE Transactions on Computers*, 2001, 50(9):985-991.

IJSER