REALIZATION OF FIR FILTER USING MODIFIED DISTRIBUTED ARITHMETIC ARCHITECTURE

Ramesh .R [1] , Nathiya .R [2]

[1]Department of Electronics and Communication Engineering,
Saveetha Engineering College, Chennai, Tamil Nadu, India.
raammesh1976@yahoo.co.in
[2]Department of Electronics and Communication Engineering,
SRM University, Kattankulathur, Chennai, Tamil Nadu, India

Abstract

This paper presents an efficient implementation of Finite Impulse Response Filter (FIR) using Distributed Arithmetic (DA) architecture. Here, the multipliers in FIR filter are replaced with multiplierless DA based technique. The DA based technique consists of Look Up Table (LUT), shift registers and scaling accumulator. Analysis on the performance of various filter orders with various partitions on different address length of partial tables are done using Xilinx 12.3 synthesis tool. The proposed 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.

Index terms*: *Distributed Arithmetic, Finite Impulse Response , Field Programmable Gate Array), Look Up Table

1. INTRODUCTION

Digital Signal Processing (DSP) has been increasing in popularity due to the declining cost of general purpose computers and Application Specific hardware. Since many telephony and data communications applications have been moving to digital, the need for digital filtering methods continues to grow[1][2]. Along with the advancement in Very Large Scale Integration (VLSI) technology and the DSP has become increasingly popular over the years, the high speed realization of FIR digital filters with less power consumption has become much more demanding. Since the complexity of implementation grows with the filter order and the precision of computation, real-time realization of these filters with desired level of accuracy is a challenging task. Several attempts have, therefore, been made to develop dedicated and reconfigurable architectures for realization of FIR filters in Application Specific Integrated Circuits (ASIC) and Field-Programmable Gate Arrays (FPGA) platforms.

Systolic design architectures represent an efficient hardware implementation for computation-intensive DSP applications because of its features like simplicity[3], regularity and modularity of structure. In addition, they also possess significant potential to yield high-throughput rate by exploiting high-level of concurrency using pipelining or parallel processing or both .To utilize the advantages of systolic processing, several algorithms and architectures have been suggested for systolization of FIR filters[4] However, the multipliers in these structures require a large portion of the chip-area, and consequently enforce limitation on the maximum possible number of Processing Elements (PE's) that can be accommodated and the highest order of the filter that can be realized.

As the scaling of silicon devices has progressed over the past four decades, semiconductor memory has become cheaper, faster and more efficient. According to the requirement of application environments, memory technology has been developed in a wide and diverse manner. To get the overall performance and to minimize the access-delay and power dissipation, either the processor has been move to memory or the memory has been move to processor in order to place the computing-logic and memory elements at closest proximity to each other .In addition to that, memory elements have also been used for a complete arithmetic circuit or a part of that in various applications. Memory-based structures are well-suited for many Digital Signal Processing (DSP) applications. Memory elements like RAM or ROM are used as a part or whole of an arithmetic unit. Memory-based structures are more regular compared with the multiply-accumulate structures; and have many other advantages, e.g., very greater potential for high-throughput and reduced-latency in implementation, (since the memory-access-time is much lesser than the usual multiplication-time) and are expected to have less dynamic power consumption due to less switching activities for memory-read operations when compared to the conventional multipliers [1]. The conventional Finite Impulse Response filters use multipliers, adders and delay elements to produce the required output. The multipliers which multiplies input with the fixed content significantly occupies more place to store their temporary values and also increases the power consumption .So, these multipliers are replaced with memory based structures to reduce area and also to reduce the system latency[5]. Several architectures have been reported for memory-based implementation of discrete sinusoidal transforms and digital filters for Digital Signal Processing Applications.

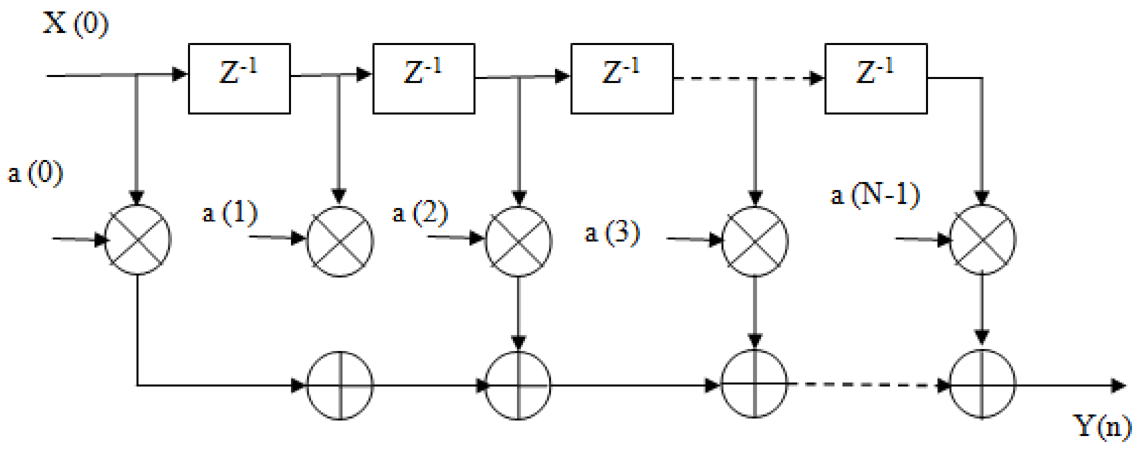

Figure 1: FIR filter using Multiplier

One of the memory based technique is Distributed Arithmetic (DA) [4]. In FIR filters, the multipliers are replaced with the multiplier less Distributed Arithmetic (DA)-based technique and it has gained popularity, for its high-throughput processing capability and increased regularity which results in cost-effective and area-time efficient computing structures. The DA based technique consists of Look Up Table, shift registers and scaling accumulator[6].

In this paper, section 2 describes the formulation of the proposed algorithm and section 3 derives the FIR structures for the proposed algorithm. Section 4 gives the simulation results. The conclusion and the scope of the future work are presented in section 5.

2. FORMULATION OF THE ALGORITHM

Distributed Arithmetic (DA) technique is bit-serial in nature [9]. It is actually a bit-level rearrangement of the multiply and accumulation operation. The basic DA is a computational algorithm that affords efficient implementation of the weighted sum of products, or dot product.

DA is a bit-serial operation used to compute the inner (dot) product of a constant coefficient vector and a variable input vector in a single direct step and is given by

$$y = \sum_{k=1}^{K} A_k x_k \tag{1}$$

where y - output response
A_k - constant filter coefficients
x_k - Input data

Let x_k be a N-bits and can be expressed in scaled two's complement number as

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

Substituting (2) in (1),

$$y = \sum_{k=1}^{K} A_k \left[-b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} \right]$$

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

Expanding the inner part,

$$\begin{aligned} y = & -[b_{10} \bullet A_1 + b_{20} \bullet A_2 + \cdots + b_{K0} \bullet A_K] \\ & + [(b_{11} \bullet A_1) 2^{-1} + (b_{12} \bullet A_1) 2^{-2} + \cdots + (b_{1(N-1)} \bullet A_1) 2^{-(N-1)}] \\ & + [(b_{21} \bullet A_2) 2^{-1} + (b_{22} \bullet A_2) 2^{-2} + \cdots + (b_{2(N-1)} \bullet A_2) 2^{-(N-1)}] \\ & \vdots \\ & + [(b_{K1} \bullet A_K) 2^{-1} + (b_{K2} \bullet A_K) 2^{-2} + \cdots + (b_{K(N-1)} \bullet A_K) 2^{-(N-1)}] \end{aligned}$$

Rearranging the summation based on power terms and then grouping the sum of the products,

$$y = -\sum_{k=1}^{K}(b_{k0}) \bullet A_k + \sum_{n=1}^{N-1}[b_{1n} \bullet A_k + b_{2n} \bullet A_2 + \cdots + b_{Kn} \bullet A_K]2^{-n}$$

The final formulation,

$$y = -\sum_{k=1}^{K} A_k \bullet (b_{k0}) + \sum_{n=1}^{N-1}\left[\sum_{k=1}^{K} A_k \bullet b_{kn}\right]2^{-n} \qquad (4)$$

3. FIR REALIZATION USING DA

The DA of FIR filter consists of Look Up Table (LUT), Shift registers and scaling accumulator.

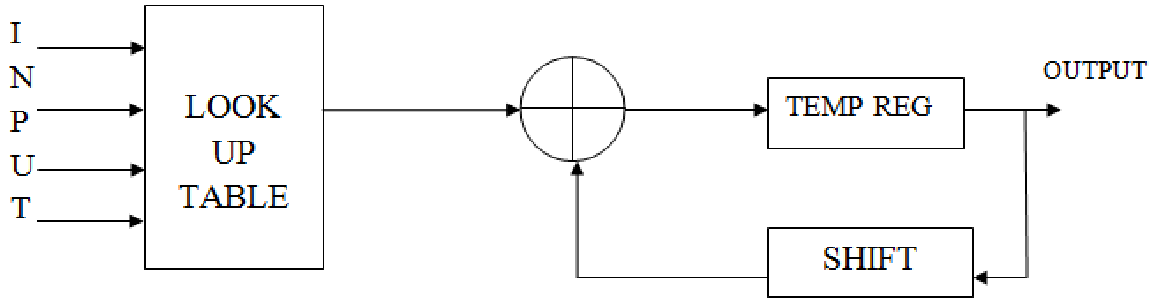

Figure 2: FIR filter using Look Up Table

By eq.4, each term inside the brackets indicates a binary AND operation involving a bit of the input variable and all the bits of the constant. The plus signs denote arithmetic addition operations. The exponential factors denote the scaled parts of the bracketed pairs to the total sum [19]. Using this, a look-up table can be constructed that can be addressed by the same type of scaled bit of all the input variables and can access the sum of the terms within each pair of brackets.

From eq.4, we can deduce that $\left[\sum_{k=1}^{K} A_k b_{kn}\right]$ has only 2^K possible values and it can be pre-calculated for all possible values of b_{1n} b_{2n} ...$b_{Kn...}$ We can store these in a look-up table of 2^K words addressed by *K-bits*. For e.g., if the number of inputs is 4, then the LUT will have 2^4 = 16 memory words[7].

Table I: Look Up Table for 4 Tap Filter

ADDRESS	DATA
0000	0
0001	H_3
0010	H_2
0011	H_2+H_3
0100	H_1
0101	H_1+H_3
0110	H_1+H_2
0111	$H_1+H_2+H_3$
1000	H_0
1001	H_0+H_3
1010	H_0+H_2
1011	$H_0+H_2+H_3$
1100	H_0+H_1
1101	$H_0+H_1+H_3$
1110	$H_0+H_1+H_2$
1111	$H_0 + H_1+H_2+H_3$

Each product term consists of a variable (signal) and a constant (coefficient) both in fixed point binary format but not necessarily of the same word length; Rather than compute the product on a term by term basis, the partial products of all terms are computed simultaneously, and in the time it would take to compute a single partial product on bit by bit basis. These partial products are generally the filter coefficients. These partial product filter coefficients of all terms are cumulated on bit by bit basis .Finally all the cumulative partial products of each bit are added and the result is produced.

In DA, all the cumulative partial product outcomes are recomputed and stored in a look up table which is addressed by the multiplier bits. All input variables are sequenced simultaneously, bit serial first to address the LUT; its outcome is added to the accumulated partial products [8].

The complete dot product computation takes M clocks where M is the number of input variable bits, and is independent of the number of input variables[11].

During the first iteration, the Least-Significant Bits $x_0(n)$, $x_0(n-1)$,..., of the K input samples form an K-bit address to the Look Up Table for f(x,0), and that table's output becomes the initial value of the accumulator. During the second iteration, the next-to-least significant bits $x_1(n)$, $x_1(n-1)$,..., $x_1(n-K+1)$ of the K input samples form another K-bit address to the lookup table for f(x,1), and the adder sums the Look up Table output to the contents of the accumulator shifted by one bit. This process continues until the last iteration, where the most-significant bits $x_{N-1}(n)$, $x_{N-1}(n-1)$,..., $x_{N-1}(n-K+1)$ of the K input samples form an K-bit address to the Lookup Table for f(x, N-1) and the adder sums the Look up Table output to the contents of the accumulator after shifting it to the corresponding position[9][17][18].
Let the filter inputs be X0 = 0(0000); X_1 =10(1010); X_2 =15(1111); X_4 =5(0101)
And the filter coefficients be H_0=1(0001); H_1 =2(0010); H2 =6(0110); H3 =4(0100);
Number of filter inputs=4

So, 4 bits are used to address the LUT where each bit from one filter input

Table II: LUT for 4 tap FIR filter for the given input

ADDRESS	DATA	DATA VALUE
0000	0	0(0000)
0001	H_3	4(0100)
0010	H_2	6(0110)
0011	H_2+H_3	10(1010)
0100	H_1	2(0010)
0101	H_1+H_3	6(0110)
0110	H_1+H_2	8(1000)
0111	H_1+H_2+H_3	12(1100)
1000	H_0	1(0001)
1001	H_0+H_3	5(0101)
1010	H_0+H_2	7(0111)
1011	H_0+H_2+H_3	11(1011)
1100	H_0+H_1	3(0011)
1101	H_0+H_1+H_3	7(0111)
1110	H_0+H_1+H_2	9(1001)
1111	H_0 + H_1+H_2+H_3	13(1101)

PARTITIONING

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 2^N 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 subdivided into a number of LUTs, called *LUT partitions*. Each LUT partition operates on a different set of filter taps. The results obtained from the partitions are summed[10][12][13][16].

Suppose the length LK inner product, then Eq.1 becomes

$$y = \sum_{k=1}^{LK} A_k x_k \qquad (5)$$

Then the sum can be partitioned into L independent Kth parallel DALUTs resulting in

$$y = \sum_{l=0}^{L-1} \left[\sum_{n=0}^{N-1} X_{Ll} + {}_{n}A_{Ll+n} \right] \qquad (6)$$

This is shown in figure for the realization of 16 tap filter using Partition 4.

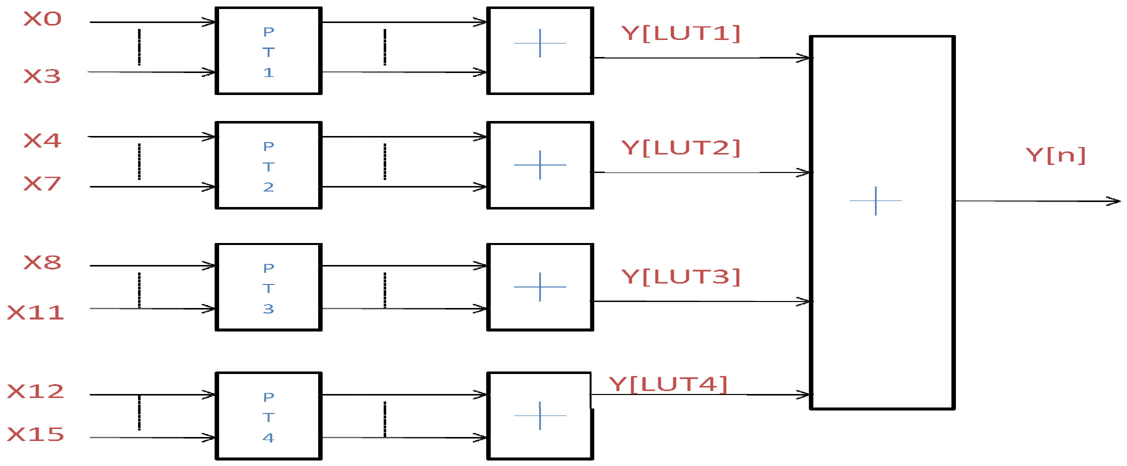

Figure 3: Realization of 16 tap Fir filter using partial tables

The realization of the above filter with various partitions

No Partition
Memory locations = 2^16 = 65,536

Partition 8
Partial Tables = 2; each with 8 inputs

Memory locations = 2* (2^8) = 5

Partition 4

Partial Tables = 4; each with 4 inputs

Memory locations = 4 * (2^ 4) = 64

Partition 2

Partial Tables = 8; each with 2 inputs

Memory locations = 8 * (2^2) = 32

4. PERFORMANCE ANALYSIS

The results of realization of the FIR filter in terms of area and delay with respect to the filter of various taps of various address length is presented below

Table III: Performance Analysis - LUT

FILTER TAP	ADDRESS SIZE	NO PARTITION	PARTITION 8	PARTITION 4	PARTITION 2
4	2	15	-	-	12
	4	33	-	-	29
	8	70	-	-	63
	16	144	-	-	119
8	2	121	-	36	31
	4	247	-	72	68
	8	426	-	151	139
	16	953	-	305	297
1 6	2	248	255	92	69
	4	513	513	186	148
	8	875	864	340	290
	16	2705	1925	627	610

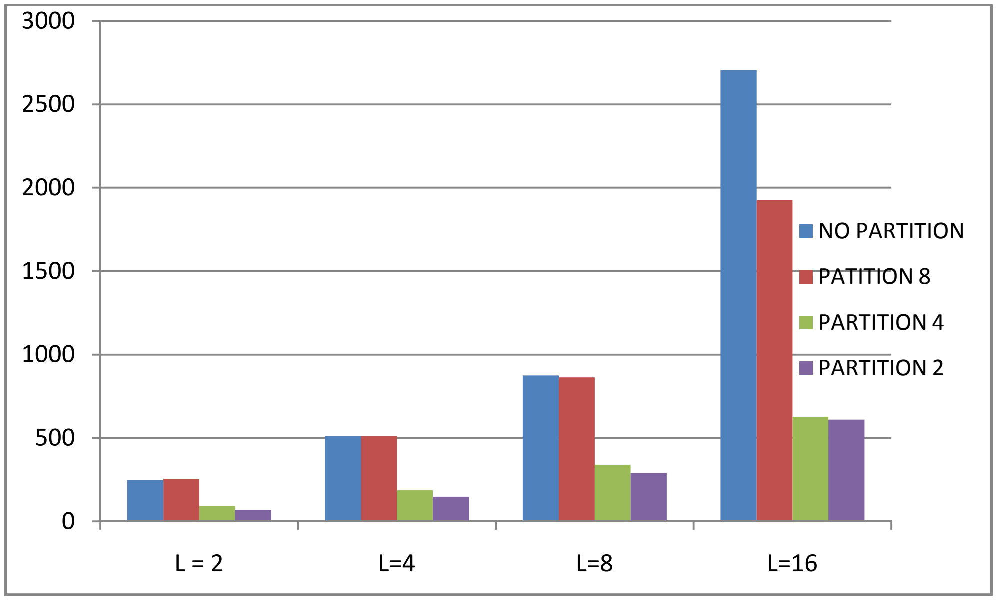

Figure 4. No of LUT's Vs Address size for 16 tap FIR Filter

From the above analysis, it is predicted that there is a reduction in number of LUT's by 75% from that of the conventional method[14][15].

Table IV: Performance Analysis – Area (Slices)

FILTER TAP	ADDRESS SIZE	NO PARTITION	PARTITION 8	PARTITION 4	PARTITION 2
4	2	8	-	-	6
	4	18	-	-	16
	8	38	-	-	35
	16	83	-	-	66
8	2	61	-	19	18
	4	127	-	39	39
	8	218	-	82	76
	16	487	-	165	163
16	2	147	129	43	41
	4	257	213	88	84
	8	487	444	165	158
	16	923	1925	627	610

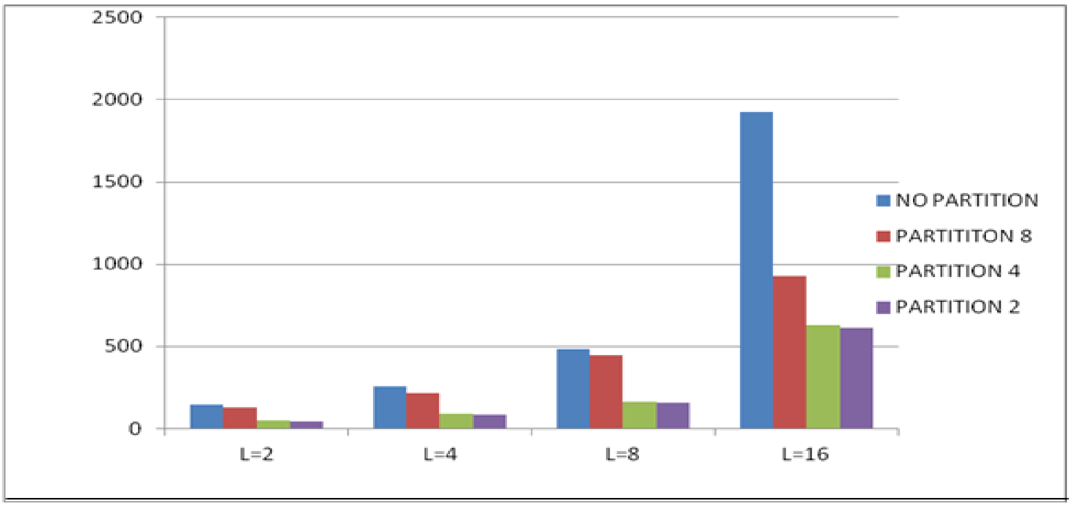

Figure 5. No of slices Vs Address size for 16 tap FIR FilterFrom the above analysis, it is predicted that there is a reduction in area in terms of slices by 34% from that of the conventional method.

Table V: Performance Analysis – Delay (ns)

FILTER TAP	ADDRESS SIZE	NO PARTITION	PARTITION 8	PARTITION 4	PARTITION 2
4	2	6.579	-	-	6.536
	4	8.354	-	-	7.899
	8	10.091	-	-	9.159
	16	11.131	-	-	10.631
8	2	8.855	-	7.937	7.847
	4	10.314	-	9.669	9.548
	8	11.976	-	10.987	10.781
	16	13.268	-	12.404	12.26
16	2	11.679	10.123	9.114	9.065
	4	12.356	11.685	10.89	10.87
	8	13.267	12.919	12.699	12.103
	16	15.879	14.515	13.637	13.547

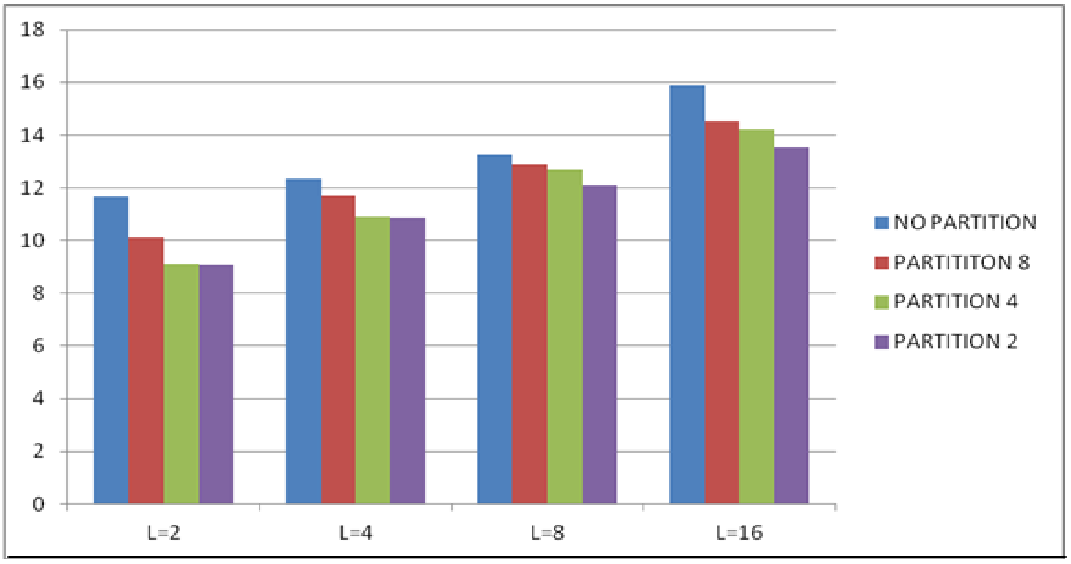

Figure 6. Delay Vs Address size for 16 tap FIR Filter

From the above analysis, it is predicted that there is a reduction in delay by 15% from that of conventional method.

5. CONCLUSION

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 technique reduces the delay by 15%, area by 34% and LUT by 75%.The performance can be further improved by pipelining all the partial tables. 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.

6. REFERENCES

[1] Jiafeng Xie,Jianjun He, Guanzheng Tan, “FPGA Realization of FIR filters for high-speed and medium-speed by using modified distributed arithmetic architectures”, Microelectronics journal 41(2010) 365-370.

[2] Antonion, “Digital Filters: Analysis, Design, and Applications”, McGraw-Hill, New York, 1993.

[3] H.T.Kung, “Why systolic architecture?” IEEE Computer 15 (1) (1982)37–45.

[4] S.Yu,E.E.Swartzlander, “DCT implementation with distributed arithmetic”, IEEE Transactions on Computers 50(9)(2001)985–991.

[5] HanhoLee, GeraldE. Sobelman, “FPGA-based digit-serial CSD FIR filter for image signal format conversion”,MicroelectronicsJournal33(5–6)(2002) 501–508.

[6] Valeria Garofalo, “Fixed-width multipliers for the implementation of efficient digital FIR filters”, Microelectronics Journal 39(12)(2008)1491–1498.

[7] LeiZhang,Tadeusz Kwasniewski, “FIR filter optimization using bit-edge equalization in high-speed back plane data transmission”, Microelectronics Journal 40(10)(2009)1449–1457.

[8] M.A.M.Eshtawie and M.Othman, “On-line DA-LUT architecture for high- Speed high-order digital FIR filters”,in : Proceedings of the IEEE International Conference on Communication Systems(ICCS),Singapore,November.2006, Pp.5.

[9] J.P.Choi, S.C.Shin and J.G.Chung, “Efficient ROM size reduction for distributed arithmetic”,In: Proceedings of the IEEE International Symposium Circuits Systems(ISCAS),May2000,pp.61–64.

[10] Sanjay, AttriB.S., Sohi and Y.C.Chopra, “Efficient design of Application Specific DSP cores using FPGAs “,in:International Conference on ASIC Proceedings, 2001,pp.462-466.

[11] Kim Kyung-Saeng, KwyroLee, “Low-power and area efficient FIR filter implementation suitable for multiple tape”, IEEE Transactions onVLSI Systemss 11 (1)(February2003).

[12] Uwe Meyer-Base, “Digital Signal Processing with Field Programmable Gate Arrays”, SecondEdition, 2004, Chapter2, Chapter3, pp.60–66,112–113.

[13] J.G.Peoakis, D.G.Manolakia, "Digital Signal Processing Principles, Algorithms and Application" ,Prentice-Hall, Upper Saddle River, NJ, 1996.

[14] A.Croisier, D.J.Esteban, M.E.Lecilion and V.Rizo, "Digital filter for PCM encoded signals", U.S.Patent 3777130, December4, 1973.

[15] A.Peled,B.Li "A new hardware realization of digital filters" ,IEEE Transactions on Acoustics Speech and Signal Processing(6)(1974)456–462.

[16] K.Yiu,"On sign-bit assignment for a vector multiplier", Processing of IEEE 64 (1976) 372–373.

[17] K.K.Parhi,in: "VLSI Digital Signal Processing Systems : Design and Implementation", Wiley, NewYork, 1999.

[18] P.K.Meher, "Hardware efficient systolization of DA-based calculation of finite digital convolution of finite digital convolution", IEEE Transactions on Circuit and Systems II: Express Briefs 53(8) (Aug.2006)707–711.

[19] 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.

Authors

R.Ramesh was born in Kanyakumari, India, 1976. He received the B.E.Degree in Electronics and Communication Engineering in 1998 and the M.E. degree in Communication Systems in 2000, both from Madurai Kamaraj University, India. He has been awarded Doctoral degree in the SRM University in the year 2009 for his research work on Testing the Stability of two dimensional recursive filters. He is currently working as a Professor in the Department of Electronics and Communication Engineering at Saveetha Engineering College, Chennai. India. His current research interests concern digital signal processing particularly to find the stability of two dimensional recursive digital filters.