Area Efficient Reloadable FIR Filter based on NEw Distributed Arithmetic (NEDA)

Biplab Roy
Department of Electronics and Communication Engineering
Neotia Institute of Technology, Management and Science
South 24 pgns, West Bengal, India

***Abstract*— This paper describes the design and implementation of highly efficient circuit for the implementation of FIR filter in terms of power and area keeping the speed at per with the fully parallel DA. It is a multiplier-less FIR filter which is designed based on new distributed arithmetic algorithm (NEDA). This NEDA based technique consists of Multiplexers, shifters and accumulator. Moreover in this architecture the filter coefficients can be any time modified at once, which is certainly an advantage over simple DA where the LUT contents need to recomputed before the memory is rewritten. Analysis on the performance are done using Xilinx-Sysgen and Mathworks-Simulink. The proposed architecture provides an efficient implementation compared to other existing structures for FIR Filter.**

Keywords*— *DA, NEDA, FIR filter, XILINX-Sysgen.

I. Introduction

FIR filters are extensively used in various communication systems mainly where phase linearity is important and have been the primary signal processing unit. In fact any system which changes the frequency content of the incoming signal one way or other, generally is called the "Filter" and so its performance improvement in terms of speed, power and area have always been and will remain demanding forever. In signal processing, the implementation method of filters and many others like various transforms have been primarily based on multipliers [1], which is the main component of the MAC unit, the very important building block which is defined as:

$$y[k] = \sum_{k=1}^{K} A_k x_k[n] \quad (1)$$

Where y[n] = response of network at time n, $x_k[n]$ = k-th input variable at time n and A_k = weighting factor of k-th input variable that is constant for all n, and so it remains time-invariant.

Direct multiplier based implementation will be highly expensive in terms of area and power though due to the advent of low-power, low-cost FPGAs, which have embedded customized DSP blocks within it and increasing use of FPGAs in advanced systems, this problem have been partially addressed [2]. FPGAs are gaining popularity because of its highly parallel execution which increases the speed many fold.

So people have been searching for alternate implementations of MACs in FPGAs and subsequently Distributed Algorithm evolved as a very efficient solution for LUT-based FPGAs. Inspired by the potential of the Xilinx FPGA look-up table architecture, the DA algorithm was resurrected in the early 90’s and shown [3] to produce very efficient filter designs and it is based on an efficient partition of the function in partial terms using 2’s complement binary representation of data. Many DSP applications use functions like convolution, correlation and filtering where Inner product computations are important. This inner product computation is done using the Distributed Arithmetic principles. The partial terms can be pre-computed and stored in LUTs. Yoo et al. [4,5,6] observed that the requirement of memory/LUT capacity increases exponentially with the order of the filter, given that DA implementations need 2K – words, K being the number of taps of the filter. This approach has another problem in changing the filter coefficients for changing its functional characteristic as then the LUT contents need to be recomputed before the memory is rewritten. The problem of exponential increment of LUT size can be partially solved by breaking up the single LUT into many and partial addressing. But this approach has neither been very efficient nor it can solve the need of re-computation of LUT contents for changing the filter characteristic.

In this work, We have tried to address both the problems by utilizing the NEDA algorithm [7] and designing a reloadable architecture where filter coefficients can be any time changed without any extra effort. The results of the implementation experiment are analyzed in terms of parameters such as area and speed. The brief description of the distributed arithmetic is presented in Section 2. The implementation of the FIR filter using NEDA algorithm is discussed in Section 3. The Section 4 presents the implementation results. The last section concludes the work and presents the future work.

II. DISTRIBUTED ARITHMETIC

Distributed Arithmetic, along with Modulo Arithmetic, are computation algorithms that perform multiplication with look-up table based schemes. Indeed, DA specifically targets the sum of products (sometimes referred to as the vector dot product) computation that covers many of the important DSP filtering and frequency transforming functions. Ironically, many DSP designers have never heard of this algorithm. Inspired by the potential of the Xilinx FPGA look-up table architecture, the DA algorithm was resurrected in the early 90’s and shown to produce very efficient filter designs [1].

The multiply-intensive nature of equation 1 can be appreciated by observing that a single output response requires the accumulation of K product terms. In DA the task of summing product terms is replaced by table look-up procedures that are easily implemented in the Xilinx configurable logic block (CLB) look-up table architecture. We start by defining the number format of the variable to be 2's complement, fractional - a standard practice for fixed-point microprocessors in order to bound number growth under multiplication. The constant factors, Ak, need not be so restricted, nor are they required to match the data word length, as is the case for the microprocessor. The constants may have a mixed integer and fractional format; they need not be defined at this time. The variable, xk, may be written in the fractional format as shown in equation 2:

$$x_k = -x_{k0} + \sum_{b=1}^{B-1} x_{kb} 2^{-b} \tag{2}$$

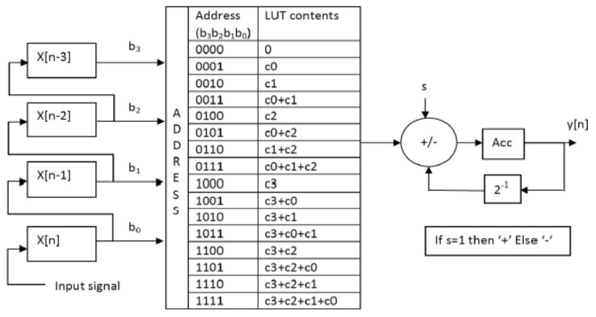
where x_{kb} is a binary variable and can assume only values of '0' and '1'. A sign bit of value -1 is indicated by x_{k0}. The final result is obtained by first substituting equation 2 into equation 1 and is:

$$y = \sum_{k=1}^{K} x_{k0} A_k + \sum_{k=1}^{K} \sum_{b=1}^{B-1} x_{kb} A_k 2^{-b} \tag{3}$$

and then explicitly expressing all the product terms under the summation symbols with

$$\begin{aligned} y = &-[x_{10}A_1 + x_{20}A_2 + \ \dots + x_{k0}A_k]2^0 \\ &+[x_{11}A_1 + x_{21}A_2 + \ \dots + x_{k1}A_k]2^{-1} \\ &\quad \dots \\ &\quad \dots \\ &+[x_{1(B-1)}A_1 + x_{2(B-1)}A_2 + \ \dots + x_{k(B-1)}A_k]2^{-(B-1)} \end{aligned} \tag{4}$$

Each term within the brackets of equation 4 denotes a binary AND operation involving a bit of the input variable and all the bits of the constant. The plus signs denote arithmetic sum operations. The exponential factors denote the scaled contributions of the bracketed pairs to the total sum. One can now construct a look-up table that can be addressed by the same scaled bit of all the input variables and can access the sum of the terms within each pair of brackets. One such implementation is shown in the fig 1.



Address ($b_3b_2b_1b_0$)	LUT contents
0000	0
0001	c0
0010	c1
0011	c0+c1
0100	c2
0101	c0+c2
0110	c1+c2
0111	c0+c1+c2
1000	c3
1001	c3+c0
1010	c3+c1
1011	c3+c0+c1
1100	c3+c2
1101	c3+c2+c0
1110	c3+c2+c1
1111	c3+c2+c1+c0

Fig. 1. Example DA Architecture

III. NEW DISTRIBUTED ARITHMETIC

DA computes the inner product of two multi-dimensional vectors. Thus, increase in the number of dimensions increases the memory requirement to store all the obtained products. This is due to the reason that, increase in number of dimensions increases the number of obtained partial products. The elimination of increased memory requirement is possible only if one or both of the inputs has a fixed set of coefficients. This method is commonly known as NEw Distributed Arithmetic (NEDA) [7]. Thus, using NEDA, distribution of arithmetic is done on the coefficient values instead of doing on the inputs. This results in memory-less DA architecture of the implemented systems. Conventional NEDA based architectures are bit-serial in nature. Depending on the application and requirement, they can be designed as digit-serial or bit-parallel architectures. Thus, NEDA is classified under the family of shift-add algorithms. This can be simply explained considering the following MAC operation:

$$z = \sum_{i=0}^{k} c_i x_i \tag{5}$$

This can be written as:

$$z = (c_1 c_2 \dots c_k) \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_k \end{pmatrix} \tag{6}$$

Considering both c_i and x_i in two's complement form as:

$$c_i = c_i^M 2^M + \sum_{k=N}^{M-1} c_i^k 2^k \tag{7}$$

where c_i=0 or 1, k=N, N+1,…M-1, C_i^M is the sign bit and C_i^N is the least significant bit. Substituting equation 7 into equation 6 one can get the following matrix:

$$z = [-2^0 \ 2^{-1} \dots \ 2^{-7}] \begin{pmatrix} c_1^0 \dots c_k^0 \\ \ddots \\ c_1^7 \dots c_k^7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} \tag{8}$$

where it is considered that the coefficients are of 8-bit in fix_8_7 format and there are k no of coefficients. Equation 8 can be implemented as some initial additions of xk-s as the terms of the second bracket are all either '1' or '0'. And finally these temporary addition results can be passed through a shifter-adder to get the final result.

IV. FIR FILTERS DESIGNED

We have designed FIR filters based on both DA and NEDA. Both are fully parallel implementation to gain the maximum speed. Fig 2 shows our circuit for DA based 7-tap FIR LPF. Here we have fragmented addressing of the LUTs to minimize the memory buildup. First the inputs are registered, then its similar bits of the input samples are sliced out parallely by the bitbasher circuit, LUTs are addressed to get the partial sums parallel and finally shift- accumulated to get the final result again parallel.

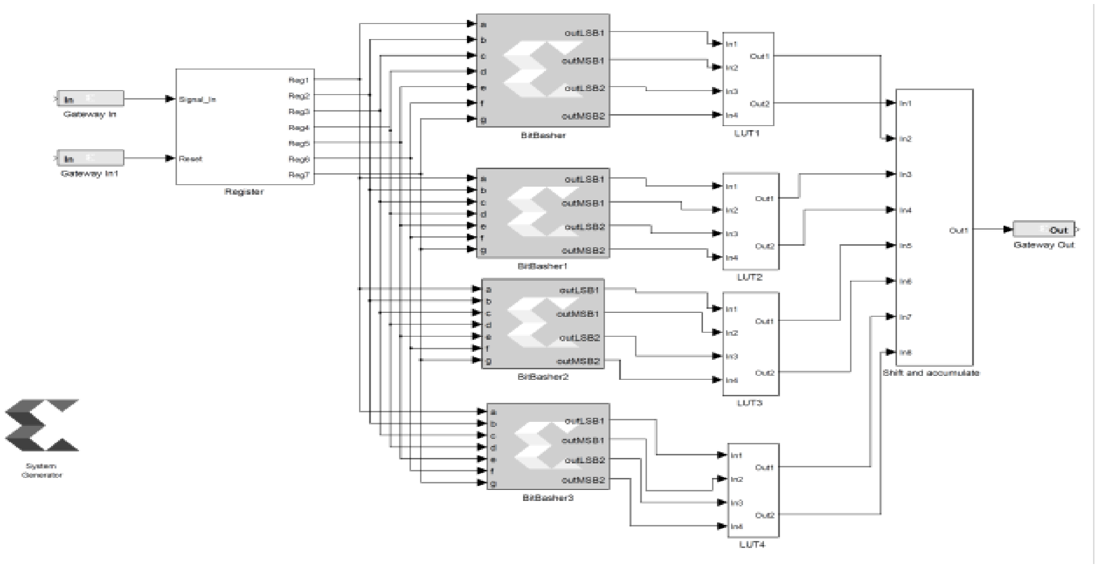


Fig. 2. 8-bit 7-tap FIR LPF –DA implementation

Fig 3 shows the same 7-tap FIR filter but based on NEDA. Here also first the inputs are registered, then muxed- out if the similar bits of the coefficients are logic ‘1’, otherwise digit ‘0’ is passed. The partial sums were calculated and shift-accumulated to get the final result again parallely. The filter coefficients were calculated from the FDAtool with 10kHz sampling frequency and 2kHz cut-off and quantized into sfix_8_7 format.

V. RESULTS

Both the circuits were tested by applying a mixture of two sinusoids of 100Hz and 4kHz to test its low-pass behavior. Both behaved exactly same and the output is shown in the fig 4. Following the same procedure we also designed 14-tap 8-bit FIR LPFs for both DA and NEDA for the purpose of comparing the resources taken when implemented in Spartan-3an FPGA which has been calculated by the Resource Estimator tool after they were mapped into the FPGA, as shown in the table 1.

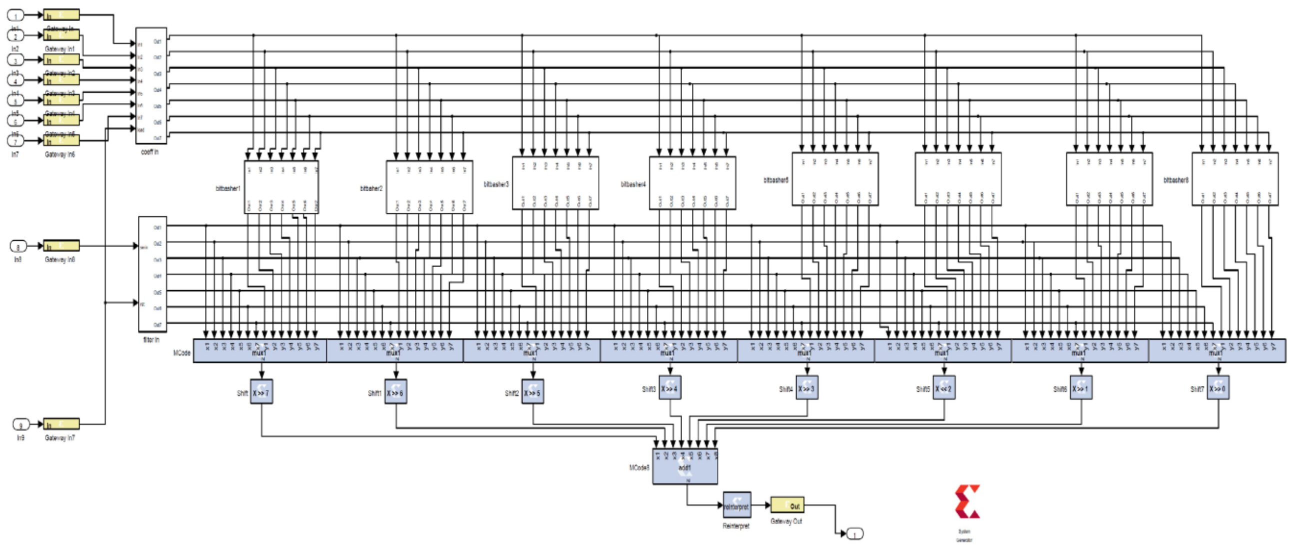


Fig. 3 8-bit 7-tap FIR LPF –NEDA implementation

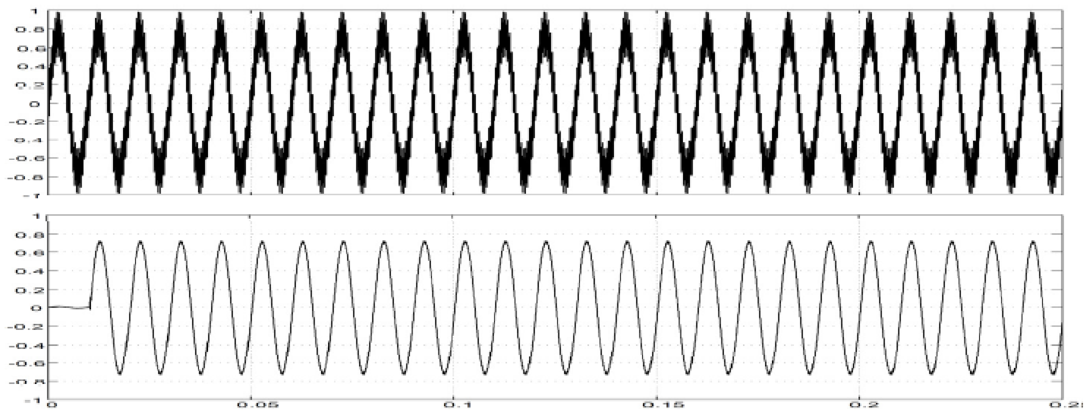


Fig. 4. Simulation results-(upper) mixture of two sinusoids 100Hz and 4kHz applied, (lower) the output which is of 100Hz showing low pass behavior.

TABLE 1. Resource utilisation summery for Spartan 700an FPGA.

Resources	8-bit DA FIR		8-bit NEDA FIR	
	7-tap	14-tap	7-tap	14-tap
Slices	324	3024	507	1020
Multipliers	0	0	0	0

VI. CONCLUSION

From the resource table (table 1) it seems that DA is area efficient for lower tap order but our NEDA based implementation becomes efficient as the number of taps are increased. Additionally our NEDA based implementation has got the advantage that the filter coefficients can be any time loaded into the circuit as can be seen from the figures.

ACKNOWLEDGMENT

Author acknowledges “Neotia Institute of Technology, Management and Sciences”, WB, India, for all of its financial support and encouragement.

REFERENCES

[1] Keshab K. Parhi, “VLSI Digital Signal Processing Systems: Design and Implementation”, Wiley, 1999.
[2] Uwe Meyer-Baese.Digital signal processing with FPGA[M]. Beijing: Tsinghua University Press, 2006
[3] Mintzer, L. “FIR filters with the Xilinx FPGA “ FPGA ’92 ACM/SIGDA Workshop on FPGAs pp. 129-134
[4] M.A. Soderstrand, L.G. Johnson, H. Arichanthiran, M. Hoque, and R.Elangovan, “Reducing Hardware Requirement in FIR Filter Design”,in Proceedings IEEE International Conference on Acoustics, Speech,and Signal Processing 2000, Vol. 6, pp. 3275 – 3278
[5] H. Yoo, and D. Anderson, “Hardware-Efficient Distributed Arithmetic Architecture for High-Order Digital Filters”, in Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005, Vol. 5, pp. 125 – 128
[6] T.Vigneswarn and P.Subbarami Reddy”Design of Digital FIR Filter Based on DDA algorithm” Journal of Applied Science ,2007
[7] Wendi Pan, Ahmed Shams, and Magdy A. Bayoumi, “NEDA: A NEw Distributed Arithmetic Architecture and its Application to One Dimensional Discrete Cosine Transform,” Proc. IEEE Workshop on Signal Processing Syst., Oct. 1999, pp. 159 – 168.G. O. Young, “Synthetic structure of industrial plastics (Book style with paper title and editor),” in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.