ASIAN JOURNAL OF PHARMACEUTICAL AND CLINICAL RESEARCH

INNOVARE ACADEMIC SCIENCES
Knowledge to Innovation

Special Issue (April)

Online - 2455-3891
Print - 0974-2441

Full Proceeding Paper

# DESIGN OF HIGH PERFORMANCE MULTIPLIERLESS LINEAR PHASE FINITE IMPULSE RESPONSE FILTERS

**APARNA A, VIGNESWARAN T**

**Department of Electronics and Communication, VIT University, Chennai Campus, Chennai, Tamil Nadu, India. Email: asha.s@vit.ac.in**

*Received: 23 January 2017, Revised and Accepted: 03 March 2017*

**ABSTRACT**

This research work proposes the finite impulse response (FIR) filters design using distributed arithmetic architecture optimized for field programmable gate array. To implement computationally efficient, low power, high-speed FIR filter a two-dimensional fully pipelined structure is used. The FIR filter is dynamically reconfigured to realize low pass and high pass filter by changing the filter coefficients. The FIR filter is most fundamental components in digital signal processing for high-speed application. The aim of this research work is to design multiplier-less FIR filter for the requirements of low power and high speed various embedded applications.

**Keywords:** Finite impulse response filter, Distributed arithmetic, Digital signal processing, Multiplier-less.

 DOI: http://dx.doi.org/10.22159/ajpcr.2017.v10s1.19564

## INTRODUCTION

Adaptive finite impulse response (FIR) filter plays an important role in digital signal processing. To improve the speed, FIR filter is used. In general, adaptive filter is a time variant filter [1]. Finite impulse response is updated. In this algorithm, tapped delay finite impulse weights are updated. In past years, the multiplier-less distributed arithmetic (DA) gained a high throughput that shows better cost-efficient and area time efficient for computing methods. In improved design, only one look-up table (LUT) is used for filtering and weight update.

By using DA, the output range is increased by LUT. Then, the improvement of the throughput is based on the FIR filtering and weight updating. In this paper, adder based process is used instead of carry save process. The shift accumulation process is complex compared to carry saving process. It reduces the area complex and power consumption. This reduction process is also same for shift accumulation. In distributed algorithm and least mean squares (LMS) algorithm is used. In general, in LMS algorithm for every clock cycle, it computes the output and error value. The desired response of the filter and the output response difference is equal to the filter output and error value. In every training cycle, the computed error is used to update the filter weight.

## REVIEW OF DISTRIBUTED ALGORITHM

To realize the multiply-accumulate function distributed algorithm is one of the ways. The multiply accumulate function is formed by adding the corresponding part of the input data and then it started to accumulate each part to get the final results. However, the general method is to wait for the first output and then produce the next output. However, in distributed algorithm, it reduces the hardware circuit and improves the speed. The coefficient of the FIR filter is taken as h(n) which we can compute in MATLAB also. By using the LUT and distributed algorithm, the convolution computation is realized. The symmetry of the linear phase of the FIR filter, n=16. The distributed algorithm appeared an efficient solution for Xilinx architecture. By using the pipeline, we can reduce the delay and improve the speed. The function of the LUT is to give the input module corresponding to the output value. The LUT consists of 4 LUT and additional module.

In Fig. 1, four input selected line is taken and the input is just passes to the shift accumulation. In the shift accumulation, full adder, D flip-flop, shift operator is used. For full adder the sign bit, shifting bit and the first block diagram output is taken as input. Two D flip-flop is used for sum and carry, and then it is shifted to shift operator. This operation continues up to 16 bit. Instead of shift accumulation, we can use the carry save accumulation. It well reduces the complexity. By using this method, we can able to analyze the power and delay.

## LMS ADAPTIVE ALGORITHMS

The LMS algorithm [2] is most widely algorithm in adaptive filter. The main feature of the LMS algorithm is low complexity. LMS algorithm changes the filter tap weight so that the minimum square error is minimized.

To implement also, it adjusts the filter coefficient to minimize the cost function.

LMS algorithm performs the following operation [3]:

1. Calculate the output signal y(n) from the adaptive filter.
2. Calculate the error signals by using the equation: e(n)=d(n)-y(n).
3. Weight and update the filter coefficient.

In the LMS algorithm, first, we introduce the filter weight error vector e(n)=w(n)-w(0). Express the update vector as e(n).the start value for the filter coefficient is w(0).

In above figure contains 15 registers to store the precomputation sum of input samples. Since DA based long vector inner product requires a large LUT. Hence, generally, it can be spited into smaller and larger LMS adaptive filter.

## REVIEW OF INNER PRODUCT COMPUTATION

In the inner product structure, the input is given to the DA architecture. The input values are given to 0-15 and they are represented as x(n) so on…these bits are entered to the 16:1 multiplexer (mux) which select the one value. And then, the output of the mux is send to the carry save accumulation. In the carry save accumulation, the sign control is given as one of the input. In the sign control, two D flip-flop is attached, one for sum and another flip flop for carry. In the 16:1 mux, four selective products are chosen and then send into carry save accumulation. In the DA table consists of 15 input registers. It is termed as four-point inner product. These products are stored in the 16:1 mux. Then, the weight A=(w3l, w2l, w1l, w0l) are given to the mux. In the common scrambling

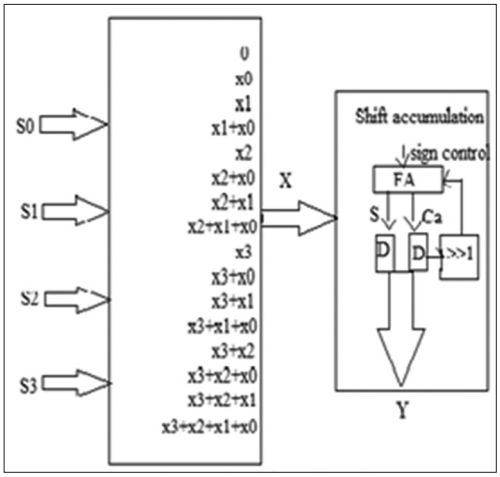


**Fig. 1: Distributed arithmetic based four point products**

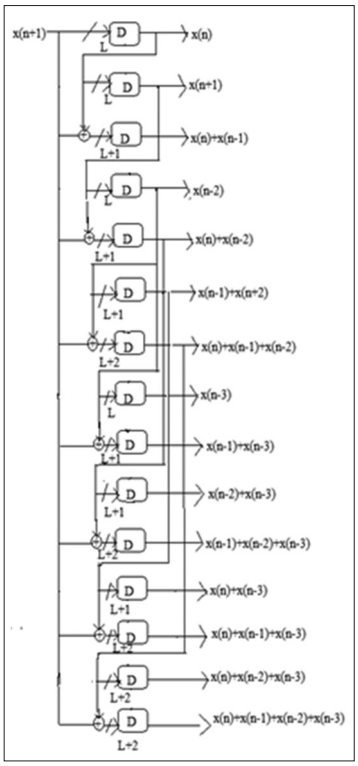


**Fig. 2: Generation of possible inputs**

algorithm, all the bits are generated. The bit slice is entered by one by one into the carry save accumulation. The array of 15 register is entered by the process of bit slice into the accumulation. The sign control bit is assumed as '0'. Two D flip flop is located for carry and sum process. The carry is comes from one D flip flop and then for sum one D flip flop. The clock bit is used for carry save accumulation. The sign bit is used to control the adder and subtraction in the carry save accumulation. Fig. 3 shows the inner product of the DA architecture. Then, the proposed design gives the one output per cycle. At the end of the process the throughput rate is produced as much as higher rate than the design. The duration of the clock is assumed as L cycle.

In the weight increment block, D flip flop, adder/subtraction and shift register is used. The D flip flop output is given to the adder/subtraction. Here, adder block is taken as full adder. For sum and carry two D flip flop is used. The output of the D flip flop is fed into the word parallel bit serial converter. The sign bit and t bit is use in the parallel bit

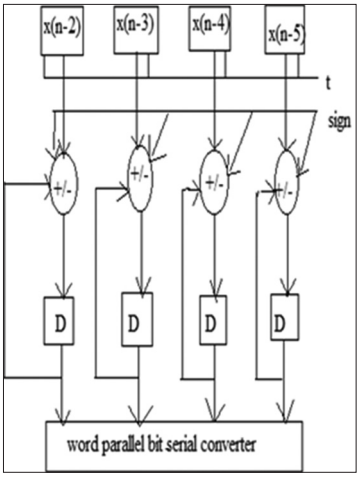


**Fig. 3: Weight increment block**

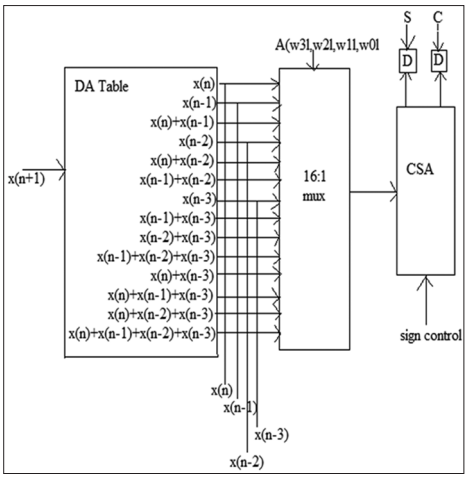


**Fig. 4: Four point inner product**

**Table 1: Comparison of DA table**

| Logic utilization | Used | Utilization (%) | Used | Utilization (%) |
|---|---|---|---|---|
| Number of slices | 344 | 9 | 0 | 0 |
| Number of 4 input LUTs | 569 | 7 | 569 | 7 |
| Number of bonded IOBs | 55 | 39 | 55 | 1 |

DA: Distributed arithmetic, LUT: Look up table, IOB: Input output blocks

serial adder. This process is used for parallel bit and then the weight increment bit should be increased. By using instead of subtractor full adder is too easy to compute the bits. The sign bit is given to all the D flip flop as a input. The output of the D flip flop is again gives as a input to the full adder. For full adder, one input bit is sign control and another bit is given input value and another bit is D flip flop output.

In the four-point inner product, the size of the bit slice is given as 0<l<l-1. The process runs by in the order of bit slice. The number of bit is mentioned in the distributive arithmetic table. Before it can be mentioned in the LUT, the input value is mentioned in the 15 array of registers. By increasing the process length, the bit slice also increased. The four products are chosen as in the DA table. These a values are fed into the 16:1 mux and then the output is passed to the carry save accumulation. By using the carry save accumulation, the output delay should be reduced. The next

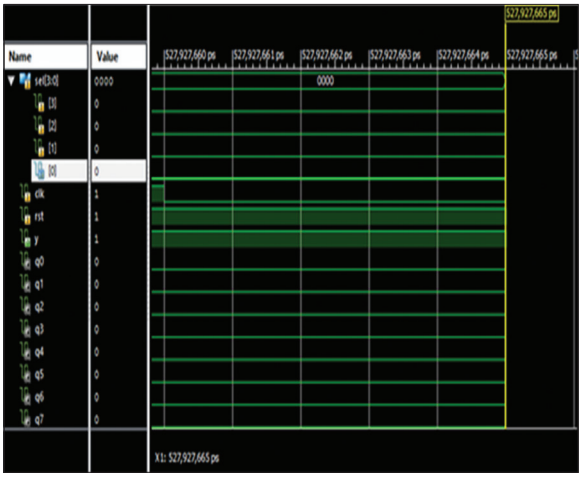


**Fig. 5: Simulation of inner product**

process should be continuous in the bit slices. The order of the bit slice should not be varied. It process through the number of input bits. In total, the array of 15 register is passed to the accumulation process. The output process is based on the selection of the input samples. It can be spited into smaller and larger LMS adaptive filter.

## PROPOSED STRUCTURE OF ADAPTIVE FILTER

The structure which has been proposed contains of filter length of n=14. This structure contains of four-point inner product, weight increment block, adder, control word, sign magnitude comparator for the calculation of error value e(n) and control word t. In the structure of the proposed work contains the four-point inner product that also includes the DA table. This section introduces the synthesis of coefficient, optimization object function and cutoff scheme. In the proposed method, carry save implementation of shift accumulation is used instead of shift accumulation. While using hardware implementation carry save adder produce simplicity. In the case of [4], all the error bits are ignored except the most significant bit. The calculating error is decoding for the control word t. The barrel shifter is used in this logic. Word parallel bit serial converter is also used in the proposed structure. In the $L^{th}$ bit cycles, the carry process occurs shift accumulates in the inner product and generates a sum and carry of word size (L+2) bit each. And then, the sum and carry bit shifted and then added with an input carry "1" for the filter output. However, one can take as a input bit as x(n-2), it is shifted to barrel shifter and then passes into adder. The output of the adder is passes into D flip flop and again the D flip flop output is passes into adder. At last the D flip flop output is taken to the word parallel bit serial converter. The input samples is taken as [5] x(n-2), x(n-3), x(n-4), x(n-5). The adder function is taken as full adder and one more input bit is taken as sign bit.

## COMPLEXITY AND SYNTHESIS RESULTS

It is estimated the inner product calculation of the DA algorithm. The clock bit is used in shift accumulation to carry the input bit. The 16:1 mux is used to select any one of the input to shift accumulation and then the bit generates. After getting the final output, we want to calculate the power delay and area [6]. Since the design of the proposed work gives the output per cycle. Then, the output is found to be much higher than that of the design. The proposed design n=14 includes the 14 adder/subtraction, four logarithmic barrel shifter and 31 registers. We have coded the existing design in Verilog code and it be synthesized [7]. The proposed work is to design compiler using CMOS library for further calculation of the area, time, power complexities. The length of the input samples and weights are taken to be 8.

## CONCLUSION

The suggested part for efficient architecture for the low power and area of DA depend on adaptive filter. The throughput rate is significantly increased by parallel LUT. This paper proposed a shift accumulation of signed partial inner product for computation of filter output. Binary coding is used to reduce the LUT size to for area efficient.

## REFERENCES

1. Haykin S, Widrow B. Least-mean Square Adaptive, Filters. Hobokon NJ, USA: Wiley; 2003.
2. Guo R, DeBrunner LS. Two high-performance adaptive filter implementation schemes using distributed arithmetic. IEEE Trans Circuits Syst II Express Briefs 2011;58(9):600-4.
3. White SA. Application of the distributed arithmetic to digital signal processing: Tutorial review. IEEE ASSP Mag 1989;6(3):4-19.
4. Allred DJ, Yoo H, Krishnan V, Huang W, Anderson DV. LMS adaptive using distributed arithmetic for high throughput. IEEE Trans Circuits Pap 2005;52(7):1327-37.
5. Guo R, DeBrunner LS. Two high performance adaptive filter implementation scheme using distributed arithmetic. In: Proceeding Asilomar Conference Signals, Systems, and Computers, November, 2011. p. 160-4.
6. Meyer PK, Agrawal P. A modular pipelined implementation of a delayed LMS transversal adaptive filter. In: Proceeding IEEE International Symposium on Circuits and Systems, May, 1990. p. 428-33.
7. Meyerand MD, Agrawal P. Amodular pipelined implementation of a delayed LMS transversal adaptive filter. In: IEEE International Symposium on Circuits and Systems, May, 1990. p. 1943-6.