FPGA Implementation of Low power Adaptive filter architecture

Sajan P. Philip, Dr. P. Sampath, P.V. Devakumar, S. Elango

Abstract--- We present a hardware architecture of adaptive filter for high throughput, low power and low area using distributed arithmetic (DA). Digital signal processing algorithms basically depends on a large number of multiplications and additions. Distributed arithmetic is an useful technique to perform MAC (Multiply And Accumulate), which is a very common operation in Digital Signal Processing Algorithms and also used to calculate inner product or simply MAC. This DSP multiplication is naturally both time and power consuming and also achieving high performance is one of the prime targets in DSP applications.

I. INTRODUCTION

Now days, Digital signal processor has been used in various electronic devices like hearing aids, PDAs, Cellular phones and so on. The main reason for the use of DSP is due to its high performance on multimedia input. The increased usage of DSP in electronic devices also increases its demand of implementation of the compact DSP algorithms, low design cost along with low power and low area consumption with better performance. The main and most widely used operation in DSP is Filtering. For example, in an N order FIR Filter, the linear convolution of the inputs x(n) with the weights *wn* will produce the output y(n). Here the Multiply – Accumulate Operation (MAC) has been performed to generate the corresponding output sample y(n). Basically, the dedicated multipliers which have expensive chip area need to be replaced with the multiplier free implementation methods frequently. But, in Distributed Arithmetic (DA), the values will be stored as the sum of scaled coefficients in Look Up Table (LUTs) in serial fashion. In LUTs, the stored value will be accessed with binary input used as address. The operations of the MAC units are replaced by the entities read from the LUTs and adding the values using DA. To access the LUT entities faster, decomposition of ROM and offset binary coding (OBC) has to be reduced along with the size of the LUTs, or else the filter length will exponentially increase with the size of the LUTs. ROM decomposition method is used to replaces the long address of the LUTs with the short ones. OBC is used to generate symmetric LUTs, by expressing the input values in different way. The size of the LUTs can be minimized using single bit logic controller.

Revised Manuscript Received on December 22, 2018.

Sajan P. Philip, Professor - Department of ECE, Bannari Amman Institute of Technology, Sathyamangalam, Tamill nadu, India. (E-mail-sajanpphilip@gmail.com)

Dr. P. Sampath, Professor - Department of ECE, Bannari Amman Institute of Technology, Sathyamangalam, Tamill nadu, India.

P.V. Devakumar, PG Student - Department of ECE, Bannari Amman Institute of Technology, Sathyamangalam, Tamill nadu, India.

S. Elango, Professor - Department of ECE, Bannari Amman Institute of Technology, Sathyamangalam, Tamill nadu, India.

II. LMS ADAPTIVE FILTER.

The adaptive FIR filter can be implemented using the LMS Filter block with five different algorithms. The figure depicts the basic block diagram of adaptive filter. x(n), which is a sample of digital signal is taken as input. Then, this x(n) is fed into the device called adaptive filter. Adaptive filter will compute the input signal with a sample of output signal y(n), at the time n. The error signal is given by,

$$e(n) = d(n) – y(n)$$

Where e(n) = Error signal or difference signal,
d(n) = Desired response signal and
y(n) = Output signal.

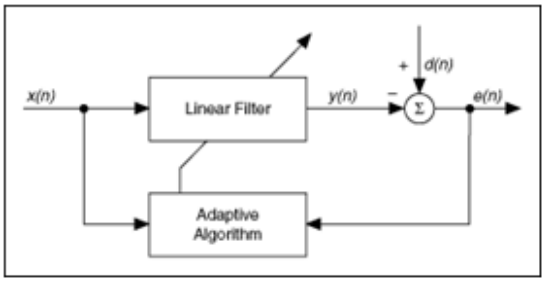

Figure 1: Block diagram of LMS Adaptive Filter.

The filter structure has adjustable parameters whose value will determine how y(n) is computed. Finally, the output signal is compared with another signal called desired response signal d(n). The difference signal or the error signal will be generated by subtracting the two signals, i.e., desired response signal from output signal at the time period n.

III. DISTRIBUTED ARITHMETIC

Distributed Arithmetic is a bridging factor of DSP systems with FPGA devices (Xilinx Family). Distributed arithmetic is a useful technique to perform MAC (Multiply And Accumulate), which is a very common operation in Digital Signal Processing Algorithms and also used to calculate inner product or simply MAC. Basically, Distributed Arithmetic is a computation algorithms which can perform multiplication with LUT (Look Up Table) based scenarios, if it is merged with modulo arithmetic. Specifically, Distributed Arithmetic concentrates on Vector dot products computation (also referred as sum of products), which will covers many frequency transformation functions and filtering functions in DSP. Even though Distributed Arithmetic has very simple derivation, it has wide range of applications.

The MAC operation of Distributed Arithmetic is illustrated below:

$$\mathbf{Y = A_1 X x_1 + A_2 X x_2 + ... A_K X x_K.}$$

$$\mathbf{i.e.,}\ y = \sum_{k=1}^{K} \mathbf{AK\ xK.}$$

IV. CONVENTIONAL DA-BASED APPROACH FOR INNER PRODUCT

Address (Weights)	Input values
0001	X(n)
0010	X(n-1)
0011	X(n)+ X(n-1)
0100	X(n-2)
0101	X(n)+ X(n-2)
0110	X(n-1)+ X(n-2)
0111	X(n)+X(n-1)+ X(n-2)
1000	X(n-3)
1001	X(n-1)+X(n-3)
1010	X(n-2)+X(n-3)
1011	X(n-1)+X(n-2)+X(n-3)
1100	X(n)+ X(n-3)
1101	X(n)+ X(n-1)+X(n-3)
1110	X(n)+ X(n-2)+X(n-3)
1111	X(n)+X(n-1)+ X(n-2))+X(n-3)

MUX → Shift accumulation → output

Figure 2: Block diagram of LMS Adaptive Filter.

The vectors **A** will be defined as A = [A_1, A_2, ...,A_n] and **B** will be defined as B = [B_1, B_2, ..., B_n] is defined as:

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{n} A_i B_i = A_1 B_1 + A_2 B_2 + \cdots + A_n B_n$$

DA is a bit serial operation which forms an inner product (dot product) of two vectors in a single step. The advantage of DA is efficiency of its operation. The increase in speed may be achieved by entering techniques such as bit pairing or partitioning the input words. This is done by separating the input into the most and least significant half. It will introduce parallelism during the computation of the most significant half of the least significant half.

Digital implementations of DSP systems generally require implementation of N-MAC operations.

A DSP processor with single processing unit will complete such operations in N-clock cycles given a single instruction for each MAC along with address generation, data fetch, and loop control. The signal is sampled 5N times faster than the rate at which the clock will operate N times faster than the sampled signal. The number of signals sampled per second will be limited, if the maximum clock speed is limited by power consumptions and other constrains. If the file size is more than N, then the limitations may be severe. The implementation of the multiple processing unit will improves the throughput along with the on chip area, logic complexity and power consumption. The employment of this unit may provide some unattractive implementation. Look up Table (LUTs) can be used to perform MAC operations in a filter. Thus, the implementation of the DA filter provides high throughput and lower logic complexity but provides high memory usage. Continued advance in memory design technology has resulted in the reduction of memory sizes, thus rendering the DA realization of digital filters an attractive alternative.

V. CARRY SAVE ADDERS

Carry save adder adds three input bits and provide a pair of output bits while other adders add two inputs. When traditional carry look ahead or ripple carry adders are used, we get the sum of all the three input bits. Analysis of literature towards this direction shows that for adding three or more inputs, a sequence of carry save adders ending with a carry look ahead adder provides a better propagation delat performance when compared to a sequence of carry look ahead adders. It is also well known that the propagation delay of the carry save adder is not affected by the input bit width of the numbers being added.

VI. PROPOSED SYSTEM

In proposed system we are using the DA table with D Flip Flop. It generates the possible sum of inputs with the given single input. These inputs are selected with filter weights it is the select lines of the 16:1 multiplexer. In proposed system the conventional adder-based shift accumulation for DA-based computation of inner-product is replaced by conditional signed carry-save accumulation to reduce the sampling period and complexity in area and power.

VII. DA TABLE GENERATION USING DFF

Figure 3 depicts the DA table generation using D Flip-flops. Here, x (n+1) is given as input. With the help of Carry Save Adder (CSA) and D Flip-flops, the 15 output values will be predicted automatically. The obtained predicted values are the summation values of the existing two outputs with the input.

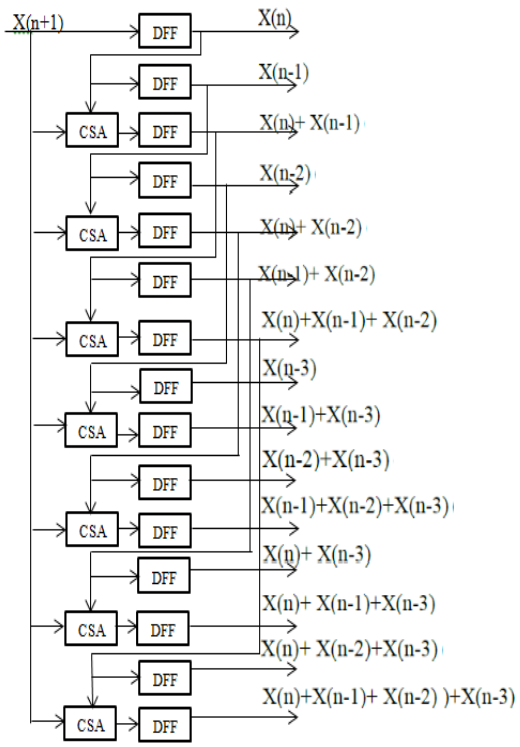

Figure 3: Block diagram of DA table generation using DFF.

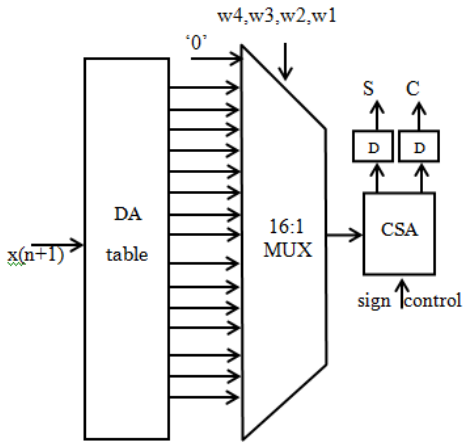

Figure 4: Four point inner product block.

Above figure shows the structure of four point inner product block. This block has the DA table which includes the 15 array registers. These 15 registers stores the partial inner products y_n, for which $0 < n < 15$. A 16:1 multiplexor (MUX) is used to select the content from one of the registers.

The selected content from the MUX will be further given to the Carry Save Adder (CSA). The CSA is used to predict the sum and carry of the selected content. This predicted sum and carry value is the output of Distributive Arithmetic.

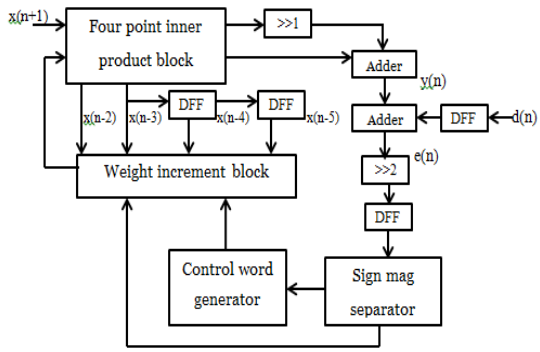

Figure 5: DA based LMS Adaptive Filter.

The above Figure 5 shows the structure of DA based LMS adaptive filter. In this block, we uses two blocks, weight increment block and control word generator. Also, we uses two inputs, one input is x and another input is d(n) which is obtained from desired signal.

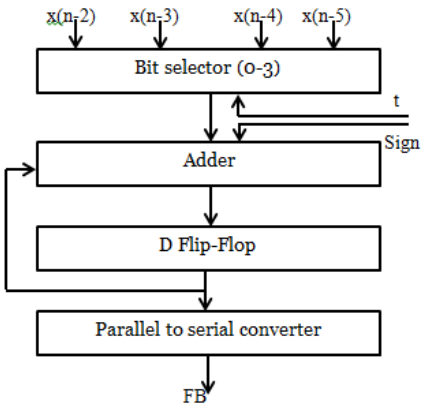

Figure 6: Weight Increment Block.

This Figure 6 provides the Weight increment block. In this block, a bit selector component is used to select any one of the 15 registers which is obtained from the DA table. These registers will be selected based on the input value *t* (which is a 3 bit input) given to the bit selector. The registers will be selected based on the change in the values of input *t*.

VIII. RESUTS

The synthesised results are generated based on the Xilinx software. In this, The RTL schematic diagram of DA table and DA based LMS filter structure. This simulation results are obtained based on different weight values. For example, the weight values of Figure 8 are 0000 and the weight values of Figure 9 are 0010 and the weights values of Figure 10 are 0111. But the output values and the error values of these simulations will vary based upon the weight. The devices usage in the FPGA implementation of the proposed architecture is shown in Table 1.

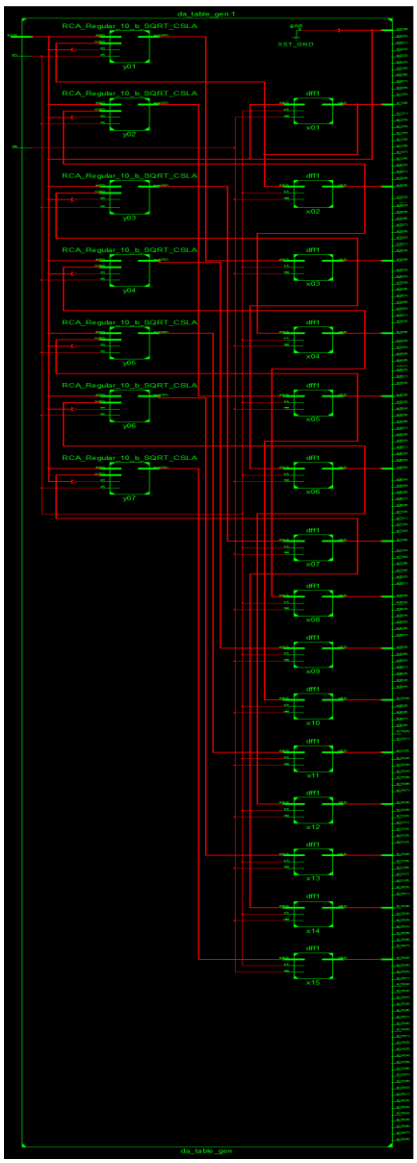

Figure 7: RTL Schematic Diagram for DA Table.

A) SIMULATION RESULTS

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

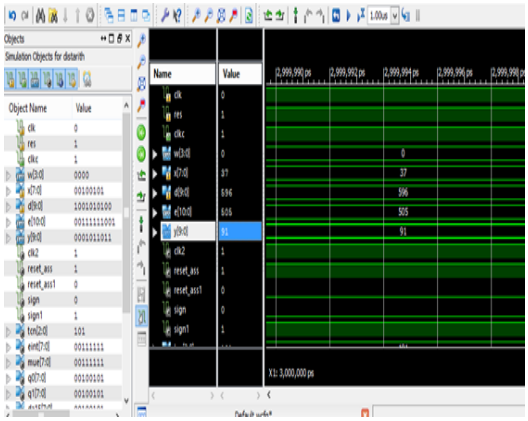

Figure 8: Simulation with weight 0000.

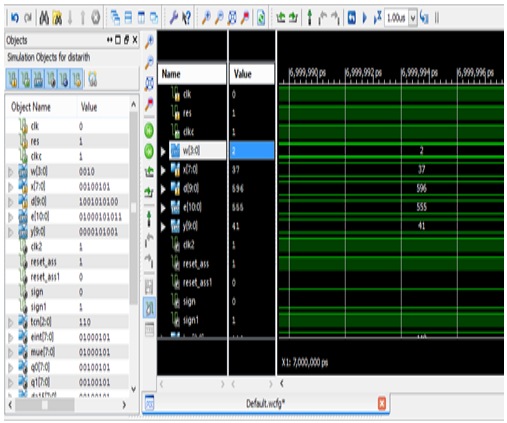

Figure 9: Simulation with weight 0010.

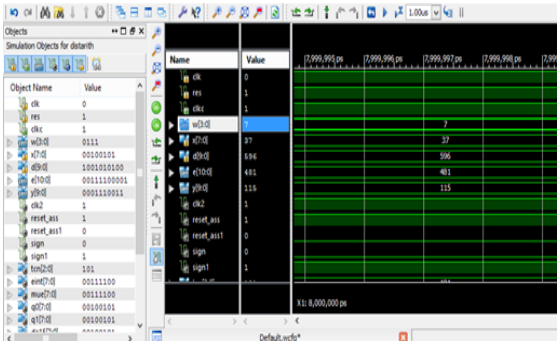

Figure 10: Simulation with weight 0111.

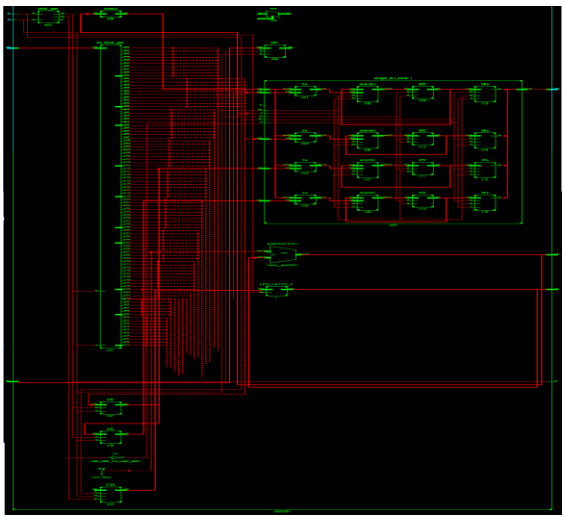

Figure 11: RTL Schematic Diagram for DA Based LMS Adaptive Filter.

B) Device usage

S.No	Device Utilization	Existing system	Proposed system
1	Total Number Slice Registers	568	548
2	Number used as Flip Flops	282	280
3	Number used as Latches	287	268
4	Average Fan out of Non-Clock Nets	2.92	2.86

X. CONCLUSION

In this paper, we have suggested an efficient pipelined architecture for DA-based adaptive filter to support very high sampling rate. This DA based scheme is used to perform vector – vector multiplication. In Our proposed system, we use the sums of delayed input samples which are generated using the D Flip-flop and carry save adders. The device usage of filter is reduced by carry save adders and proposed DA table. 3.5% hardware will be reduced to compare proposed and present system.

REFERENCES

1. C.H. Wei and J.J. Lou, “Multi-memory block structure for implementing a digital adaptive filter using DA,” IEE Proceedings, volume 133, pages. 19–26, Feb., 1986.
2. C.F.N. Cowan and J. Mavor, “New digital adaptive-filter implementation using DA techniques,” IEE Proceedings, volume 128, pages. 225–230, Feb., 1981.
3. D.J. Allred, H. Yoo, V. Krishnan, W. Huang, and D.V. Anderson,“LMS adaptive filters using distributed arithmetic for high throughput,” IEEE Transactions on Circuits and Systems, volume 52, pages 1327–1337, July, 2005.
4. D.J. Allred, H. Yoo, V. Krishnan, W. Huang, D.V. Anderson, “A novel high performance DA adaptive filter implementation on an FPGA,” IEEE International Conf. on Acoustics, Speech, and Signal Processing, volume 5, pages–161–164, Mar. 2004.
5. S.A. White, “Applications of DA to digital signal processing: A tutorial review,” IEEE ASSP Magazine, volume 6, pages 4–19, Jul. 1989.
6. A. Croisier, D. Esteban, M. Levilion, V. Rizo, “Digital filter for PCM encoded signals,” U.S. patent number 3,777, Apr., 1973.
7. D.J. Allred, H. Yoo, V. Krishnan, W. Huang, D.V. Anderson, “An FPGA implementation for a high throughput adaptive filter using DA,” 12th Annual IEEE Symposium on FPGA and Custom Computing Machines, pagesp. 324–325, Jan. 2004.
8. A. Peled and B. Liu, “A new hardware realization of digial filters,” IEEE Trans. on Acoustics, Speech and Signal Processing, volume 22, pages 456–462, Dec. 1974.