Floating-Point Multiply and Accumulate Unit Core using Distributed Arithmetic for DSP Applications

Bharathi.M[1] , Dr. Yasha Jyothi M Shirur[2]

[1]Research Scholar in BNMIT ,VTU and Assistant Professor, Department of ECE, Center for VLSI & Embedded Systems, Sree Vidyanikethan Engineering College, Tirupati, Andhra Pradesh.
[2]Professor of ECE, BNMIT, Bangalore, VTU.
[1]bharathi891@gmail.com, [2]yashamallik@gmail.com

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 10 May 2021

Abstract: The main purpose of this article is to reduce the area and power of floating-point multiplication and accumulation units using distributed arithmetic Typically, Multiply and Accumulate unit, multiplier contains partial products. The speed of the multiplier is important in the MAC unit which decides critical path, as well as area, is additionally of amazing performance in the designing of MAC. The Multiply and Accumulate unit occupies more space and more power. So to overcome these shortcomings, a technique of Distributed Arithmetic is incorporated. Considering the DA method which replaces the MAC with pre-calculated results about to store in lookup tables. The foremost advantage of DA is it consumes low power and low area. So the area and power of the Multiply and Accumulate unit is reduced. The complete implementation of the Distributed Arithmetic-based Multiply and Accumulate unit is designed using Floating Point operator. In most cases, floating-point adders and multipliers are shown in single precision IEEE-754 format. Implementation of actual DA with different architectures use in a traditional lookup table (LUT) based DA architectures such as DA based implementation with single LUT, Reduction of LUT size using two-bank coefficient partitioning and four-bank coefficient partitioning,Combination of LUT & an adder, and Adder-based DA implementations for inner-product computation. The performance evaluation of different DA-based floating-point MAC can be further evaluated. Finally, Compared the overall performance of DA-based floating-point MAC techniques in terms of area and delay. The designs are modeled using Verilog HDL and are verified by using Xilinx ISE 14.5 and ISIM simulator.

Keywords: Distributed arithmetic, Multiply-accumulate unit, Lookup tables (LUT), Digital signal Processor, inner product computation and Floating point.

1. Introduction

Digital Signal Processing (DSP) is used in many applications such as RADAR, satellite, Image Processing, audio broadcasting, Video Processing, & Biomedical applications, etc., and any DSP will have a MAC unit. Multiply accumulate unit is also known as MAC. As the name itself, we can know that it is made up of the chain of multipliers, adders, and registers. At first, inputs are given to the multiplier and then the corresponding output will be obtained, and that output will be sent to the adder and that output will be given to the accumulated register for data feedback. Therefore, designing various MAC's based on different multipliers has been done. Among them, the MAC designed with modified booth multiplier consumes less area, even though it consumes less area, further the area can be minimized too. When speed is considered, to improve the speed of the MAC there are two ways.

1. Reduction of partial products and
2. Accumulator.

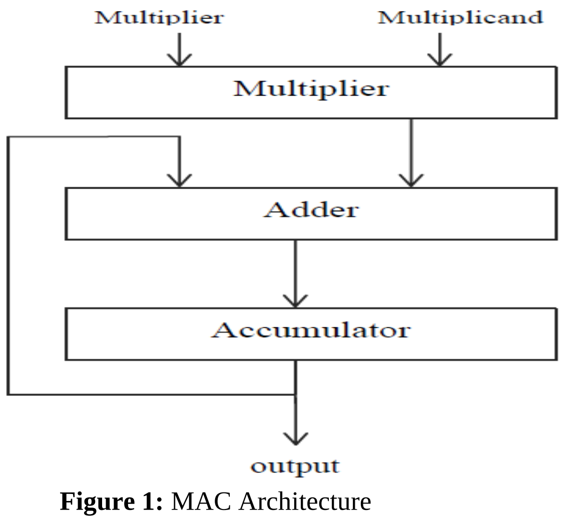


Figure 1: MAC Architecture

The above (figure1) describes the Structure of the general MAC, As discussed, MAC contains a multiplier, adder, and accumulator. Since it includes generation of partial products. Say for example NXM nmultiplier generates N partial partial products with M word length which tends to increase in area consumption and critical path delay. Therefore, designing various MAC's based on different multipliers has been carried for this research. Among them, the MAC designed with a modified booth multiplier consumes less area, even though it consumes less area we can decrease that too. So, to reduce the area consumption Distributed Arithmetic is preferred. In Distributed Arithmetic multiplication is done without a multiplier. It replaces LUT for the multiplication process. Therefore, there will be no need for partial products, in this area consumption can be decreased [1].

The Multiply and Accumulate unit occupies more space and more power. So, to overcome these drawbacks used in Distributed Arithmetic based Multiply and Accumulate unit. On considering the DA method which replaces the MAC with pre-computed results about to store in lookup tables. DA-based MAC area will be reduced to half when compared to normal multiply and accumulate unit.

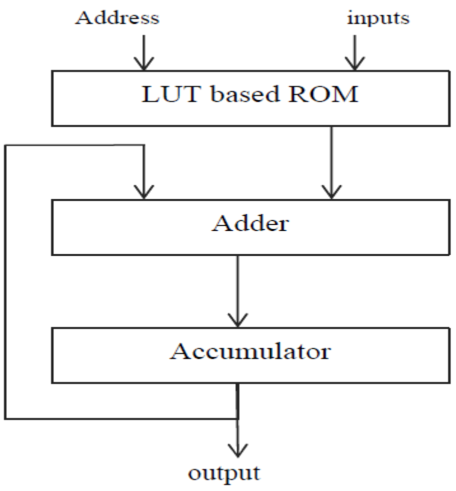


Figure 2: DA based MAC Architecture

The above (figure2) architecture is described in conventional Distributed Arithmetic based MAC, The Distributed arithmetic (DA) is a computational algorithm. This is done in multiplication using a lookup table (LUTs). It is well suited to implementation on the same field in programmable gate arrays because of its high utilization of the available Look-Up Tables(LUTs). DA targets the dot products operation and many digital signal processing (DSP) operations such as filter implementation can be reduced to one or more sum of product Computational algorithms [3]. Behind DA it is based on the rearrangement of vector dot products around the binary representation of the vector dot products around the binary format. It refers to the vector elements. Assume that X is the vector of input samples & A is a constant vector of FIR(finite impulse response) filter coefficient [2]. The dot product Y of A &X can be transcribed as

$$y = \sum_{n=1}^{N-1} \left[\sum_{k=1}^{k} A_k\, b \right] 2^{-n} + \sum_{k=1}^{k} A_k\,(-b_{k0})$$

- $\left[\sum_{k=1}^{k} A_k\, b_{kn} \right]$ $\quad$ *has only* 2^k *possible values*

With the sign bit as an input, can store in a ROM of size=2*2^k

2. Method

In general, floating point number system is an attractive method which can support dynamic range than that of fixed number system for a same word length.The main theme of this research paper is to design a Floating-point MAC core using Distributed Arithmetic. Floating point MAC is an essential block in every DSP processor. Every processor consists of floating point core which performs mathematical computations.Floating point arithmetic number system is redefined as IEEE 754 standard[8]. This floating point MAC core includes three blocks composed of Floating point multipliers & adders and accumulators.Like MAC core, Figure 3 performs multiplication and accumulation on floating point number system.

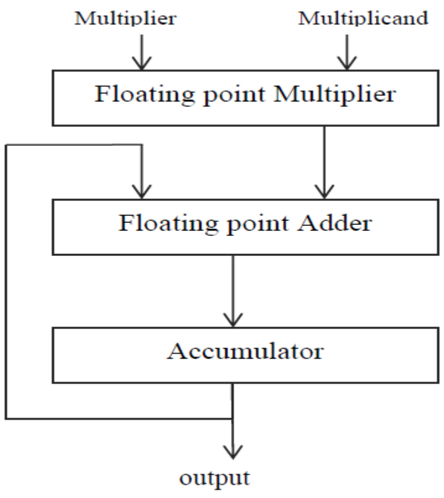


Figure 3: Floating point MAC Architecture

The below (Figure 4) describes the structure of the Distributed Arithmetic based Floating point MAC Architecture, in Distributed Arithmetic, the multiplication is completely carried out entirely using LUT. The output that we receive through LUT is Floating point operated and that is given as one of the inputs to adder and addition is performed with other inputs later the output what we receive through adder is Floating point operated and then it is followed by the accumulator. In most cases, floating-point adders and multipliers are provided in IEEE-754 format. This is a single most precise format.

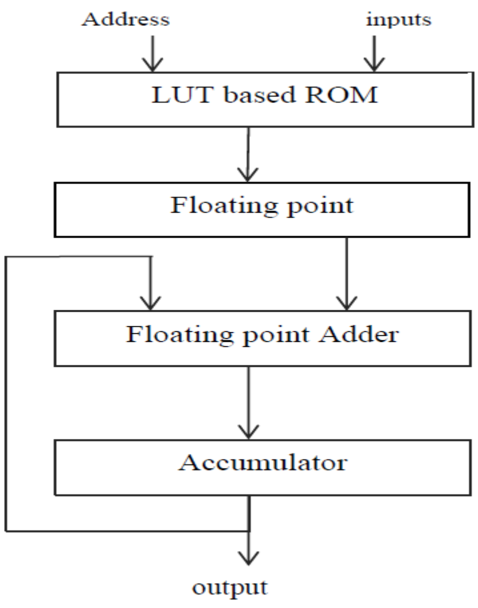


Figure 4: Distributed Arithmetic based Floating point MAC Architecture

In figure 5, includes 3 sections of blocks :Data block,LUT block & Accumulation block. Data block accepts the inputs from external world. The data block holds the incoming data and based on the address valueof LUT block it provides the output. The accumulator eithers adds or subtracts the output from that of LUT block of its previous value. The sum of products of weighted sum are aviable at the accumulator itself [2].

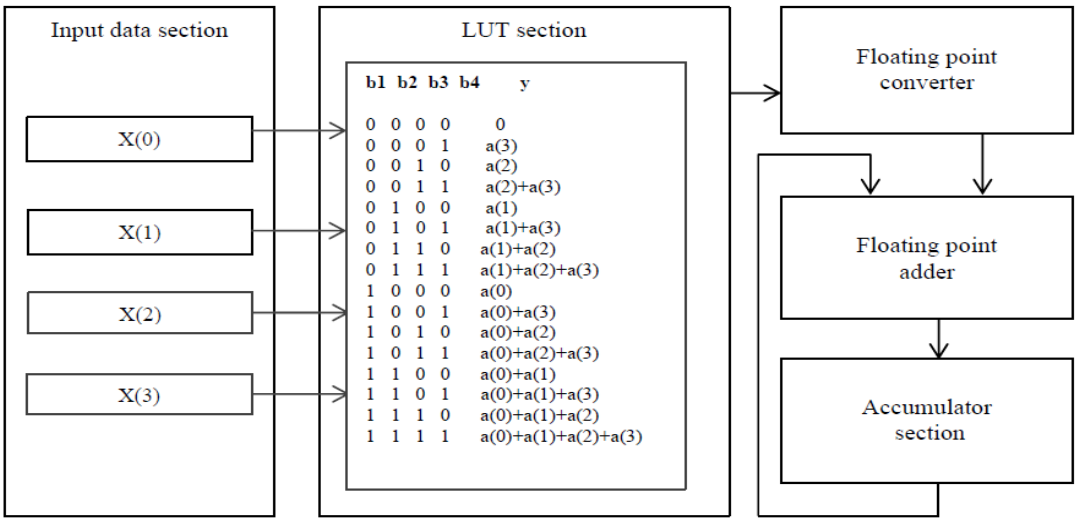


Figure 5: DA based Floating point MAC architecture using single LUT.

Let us consider the each word length of inner product is of ‘W bits’ and also assume that the LUT is divided into 2 subblocks as shown in figure. For a LUT into two sub blocks, it requires two subblocks of (W/2)rows. As in this case it requires only 4 rows of a sub block of 2- LUTs whereas for a single LUT has 16 rows. Due to partioning of a LUT into two halves , an additional adder is required and thus it in turn affects the performance parameters [2].

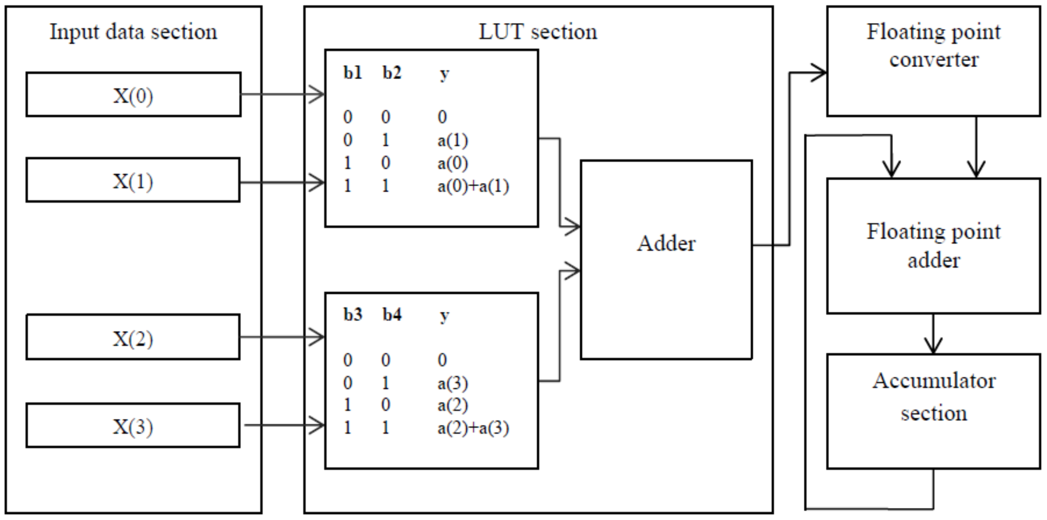


Figure 6: DA based Floating point MAC architecture using Two-LUTs.

Assume that the each word length of inner product is of ‘W bits’ also the LUT is divided into 4 subblocks as shown in figure 7. For a LUT into four sub blocks, it requires two subblocks of (W/4)rows. As in this case it requires only 2 rows of a sub block of 4- LUTs whereas for a single LUT has 16 rows. Due to further partioning of a LUT into four halves , an additional 3 adders are required and thus it in turn varies the performance.

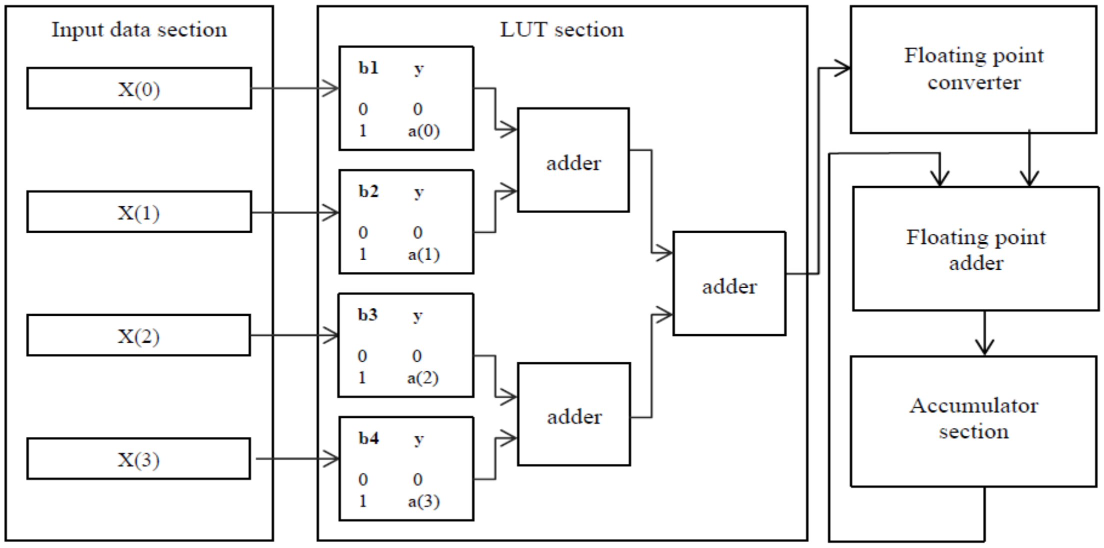


Figure 7: DA based Floating point MAC architecture using Four- LUTs.

See that the Contents in LUT of lower half is replicates in upper half. Means that still area saving is possible by reducing it to half. The resultant one is Reduction of LUT size using an adder with an inputA[0]is as shown in (Figure 8) [2]

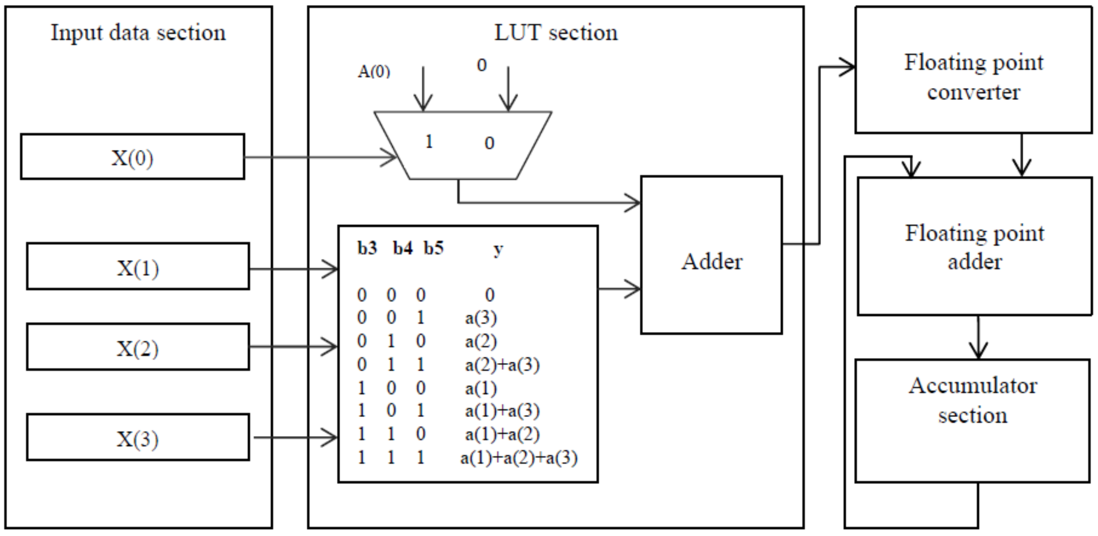


Figure 8: DA based Floating point MAC architecture using single LUT & Adder.

In this case the LUT section is recinstructed using MUX's, but many adders are required based on the data section. Since the ouput is made upof adders also called Adder Based DA[2].

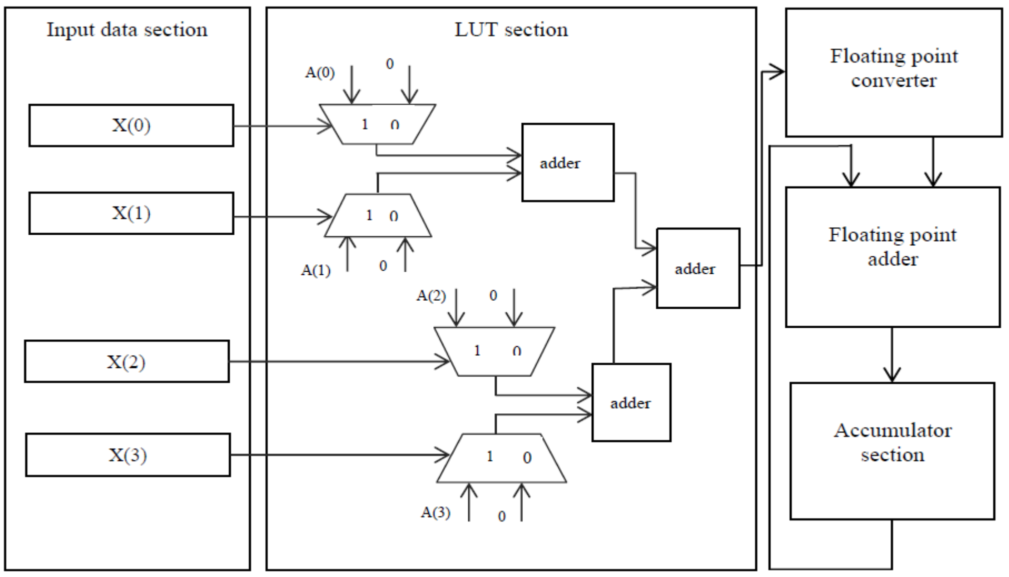
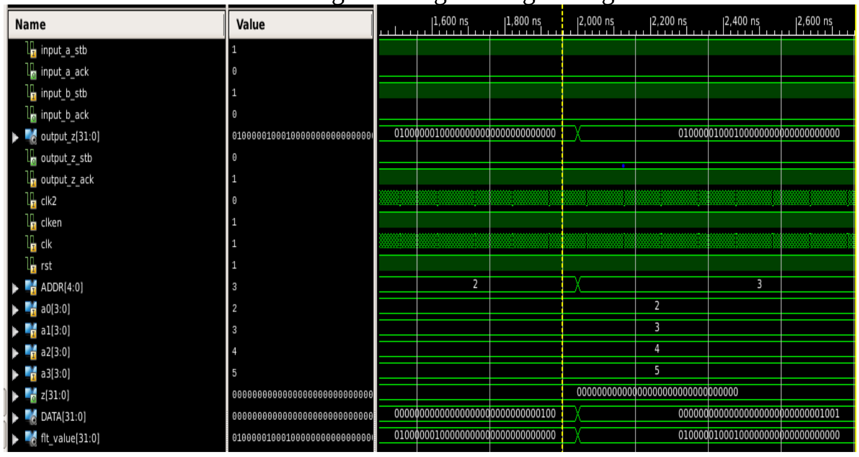


Figure 9: DA based Floating point MAC architecture using only adders.

3. Result

In this proposed work DA based implementation circuits for inner product Floating point, MAC architectures have been designed using Verilog coding and simulation has been done using Xilinx ISE 14.7.

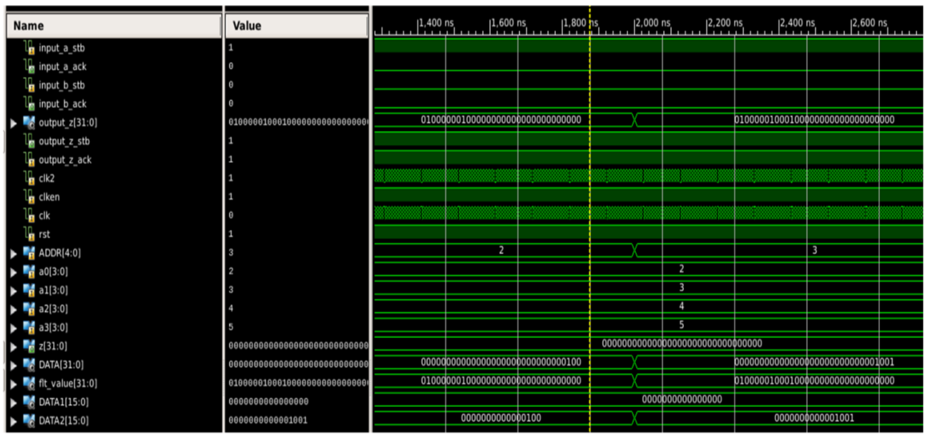


```
Name              Value
input_a_stb       1
input_a_ack       0
input_b_stb       1
input_b_ack       0
output_z[31:0]    0100000100010000000000000000000
output_z_stb      0
output_z_ack      1
clk2              0
clken             1
clk               1
rst               1
ADDR[4:0]         3
a0[3:0]           2
a1[3:0]           3
a2[3:0]           4
a3[3:0]           5
z[31:0]           00000000000000000000000000000
DATA[31:0]        00000000000000000000000000000
flt_value[31:0]   0100000100010000000000000000000
```

```
|1,600 ns  |1,800 ns  |2,000 ns  |2,200 ns  |2,400 ns  |2,600 ns
output_z[31:0]   01000000100000000000000000000000   01000001000100000000000000000000
ADDR[4:0]        2   3
a0[3:0]          2
a1[3:0]          3
a2[3:0]          4
a3[3:0]          5
z[31:0]          00000000000000000000000000000000
DATA[31:0]       00000000000000000000000000000100   00000000000000000000000000001001
flt_value[31:0]  01000000100000000000000000000000   01000001000100000000000000000000
```

Simulation 1:DA based Floating point MAC architecture using single LUT.

a0,a1,a2,a3 = inputs = 2,3,4,5
ADDR =0011= a2 +a3 = 4+5 = 9 = DATA
Sum = DATA + cin = 9+1=10
Floating point = DATA= 9 = 01000001000100000000000000000000
Clk, clken =1 then z =0 else accumulation can be done

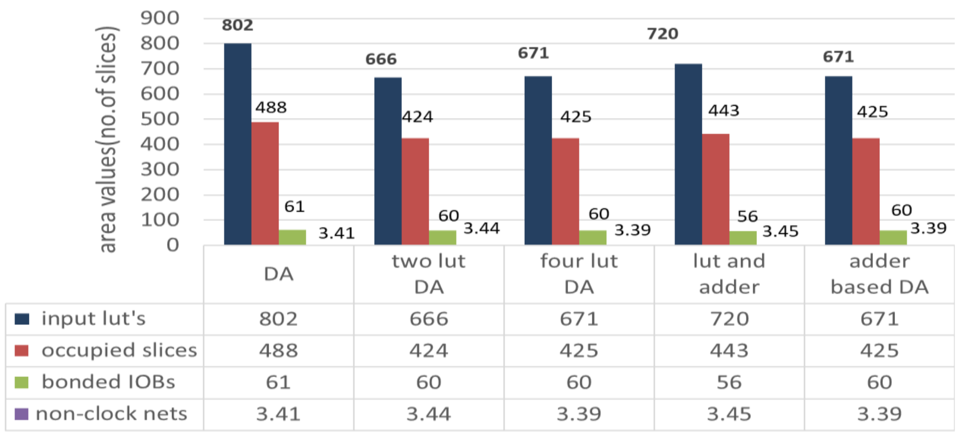


Name	Value
input_a_stb	1
input_a_ack	0
input_b_stb	0
input_b_ack	0
output_z[31:0]	0100000100010000000000000000000
output_z_stb	1
output_z_ack	1
clk2	1
clken	1
clk	0
rst	1
ADDR[4:0]	3
a0[3:0]	2
a1[3:0]	3
a2[3:0]	4
a3[3:0]	5
z[31:0]	00000000000000000000000000000
DATA[31:0]	00000000000000000000000000000
flt_value[31:0]	0100000100010000000000000000000
DATA1[15:0]	0000000000000000
DATA2[15:0]	0000000000001001

Simulation 2: DA based Floating point MAC architecture using Two-LUTs
a0,a1,a2,a3 = inputs = 2,3,4,5
ADDR =0011
DATA1 = a0+a1 = 0
DATA2 = a3+a2 = 9
DATA = DATA1+DATA2 = 0+9 = 9
Sum = DATA + cin = 9+1=10
Floating point = DATA= 9 = 01000001000100000000000000000000
Clk, clken =1 then z =0 else accumulation can be done

Here with shown simulation results for Single and two LUTs. Like wise peerformed simulation and synthesis for 4 LUTs, LUT &adder,Adder based DA structures.The below figures are the performance evaluations for area, delay-Energy & Power.

Performance evalution of various DA based floating point MAC in terms of Area :

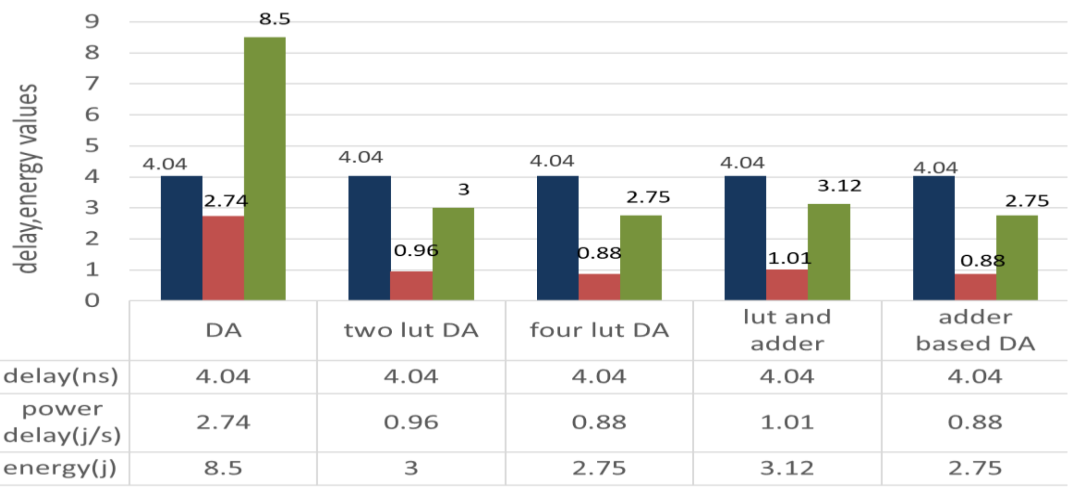


	DA	two lut DA	four lut DA	lut and adder	adder based DA
🟦 input lut's	802	666	671	720	671
🟥 occupied slices	488	424	425	443	425
🟩 bonded IOBs	61	60	60	56	60
🟪 non-clock nets	3.41	3.44	3.39	3.45	3.39

Comparison of Different Techniques of DA

Here in DA-based floating-point MAC architectures, the comparison is done with all types of DA-based floating-point MAC architectures.Among them Using LUT size reduction using two-bank coefficient partitioning consumes less area among all types of DA-based floating-point MAC architectures. From the above chart, DA consumes an area of 802 slices out of 17344 slices, and Using Using LUT size reduction using two-bank coefficient partitioning consumes an area of 666 slices, so that area consumption is decreased.

Performance evaluation of various DA based floating-point MAC in terms of delay and energy :

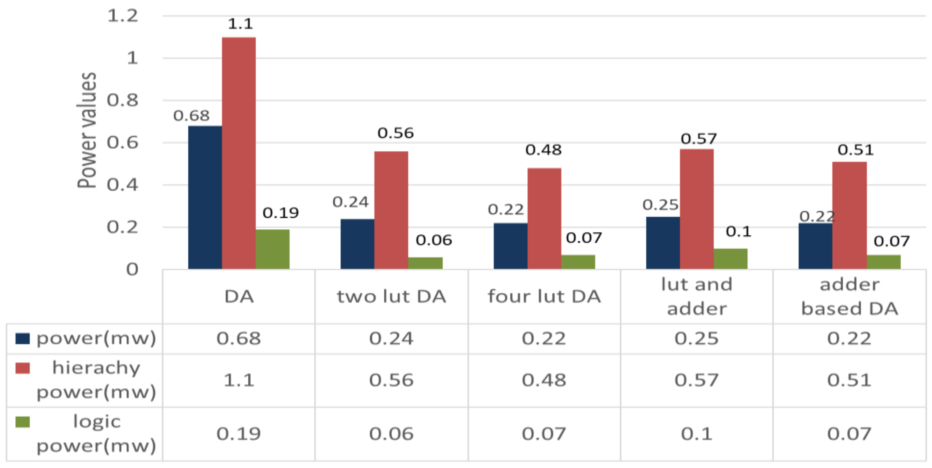


	DA	two lut DA	four lut DA	lut and adder	adder based DA
🟦 delay(ns)	4.04	4.04	4.04	4.04	4.04
🟥 power delay(j/s)	2.74	0.96	0.88	1.01	0.88
🟩 energy(j)	8.5	3	2.75	3.12	2.75

Comparison of Different Techniques of DA

From the above chart, DA consumes an energy of 8.5j and LUT size reduction using four-bank coefficient partitioning consumes energy of 2.75j, so that delay consumption is decreased.

Performance evaluation of various DA based floating-point MAC in terms of Power :



	DA	two lut DA	four lut DA	lut and adder	adder based DA
🟦 power(mw)	0.68	0.24	0.22	0.25	0.22
🟥 hierachy power(mw)	1.1	0.56	0.48	0.57	0.51
🟩 logic power(mw)	0.19	0.06	0.07	0.1	0.07

Comparison of Different Techniques of DA

From the above chart, DA consumes a power of 0.68 MW and LUT size reduction using four-bank coefficient partitioning consumes a power of 0.22mw, so that power consumption is decreased.

4. Conclusion

In this paper implementation of Distributed Arithmetic based floating point MAC Core is implemented using Xilinx ISE 14.5 and ISIM simulator. Comparison of different techniques of DA-based floating-point MAC have done. The main weakness of distributed arithmetic is as input section size increase exponentionally for each added input line. So, area consumption increases little. To overcome this drawback Offset Binary Coding is selected, so that area consumption is completely reduced to half when compared with DA based MAC. On considering, LUT size reduction using four-bank coefficient partitioning power consumption is decreased to 32% and energy is decreased to 32%, From these overall comparisons of different techniques of DA based floating-point MAC unit consumes power and area.

5. Acknowledgment

Foremost,I would like to express my sincere thank to Dr. Yasha Jyothi M Shirur for giving valuable suggestions and comments on this work.

References

1. Priyanka Nain and Dr.G.S.Virdi, Multiplier Accumulator Unit, October 2016.
2. Mahesh Mehandale, Mohit Sharma, and Pramod Kumar Meher, DA-Based Circuits for Inner-Product

Computation, pp:77-103.
3. Rajeevan Amirtharajah, Distributed Arithmetic, pp-503-512
4. Jiafeng Xie, jianjun He and guanzheng Tan, FPGA realization of FIR filters for high speed and medium speed by using modified distributed arithmetic architectures, June 2010, pp: 365-370.
5. Amita Nandal, T.Vigeswarn, Ashwani K. Rana and Arvind Dhaka, An Efficient 256-Tap Parallel FIR Digital Filter Implementation Using Distributed Arithmetic Architecture, 2015, pp: 605-611.
6. Ramesh. R, Nathiya. R, Realization of FIR filter using Modified Distributed Arithmetic Architecture, 2012.
7. Wayne P.Burleson, Louis L.Scharf, A VLSI design methodology for distributed arithmetic, June 1991
8. Sonali Mehta, Balwinder singh and Dilip Kumar, performance Analysis of Floating Point MAC Unit, September 2013.
9. R.Prakash Rao, N.Dhanunjaya Rao , K.Naveen and P.Ramya, Implementation of the Standard Floating point MAC Using IEEE 754 Floating point adder, Feb 2018.
10. Yadagiri Karri, Rajesh Misra, Implementation of 32 Bit Floating point MAC Unit to Feed Weighted Inputs to Neural Networks, April 2015.
11. Mohamed Asan Basiri M, Noor Mahammad Sk, An Efficient Hardware-Based Higher Radix Floating Point MAC Design, November 2014.
12. Dhananjaya A, Deepali Koppad, Design Of High Speed Floating Point MAC Using Vedic Multiplier And Parallel Prefix Adder, June 2013.