Implementation of Parallel and Pipeline Scheme in the Standard Floating Point Adder to Improve the Speed

R. Prakash Rao

***Abstract:** In real time Signal Processing applications, the analogue signal is over sampled as per the Nyquist criterion in order to avoid the aliasing effect. Floating Point (FP) adder is used in the floating point Multiplier Accumulator Content (MAC) for real time Digital Signal Processing(DSP) applications. The heart of any real time DSP processor is floating point MAC. Floating Point MAC is constructed by Finite Impulse Response (FIR) or Infinite Impulse Response (IIR) filters. FIR filters are stable than IIR filters because the impulse response is finite in FIR. Hence, for stable applications FIR filters are preferred. These FIR filters are intern constituted by FP adder, FP multiplier and shifter. In conventional floating point adder the two floating point numbers are added in series. Series means one after the other so the computation speed is less. In series fashion adding the floating point numbers means definitely it furnishes more delay[1] because in the addition of floating point numbers, along with the addition of mantissas; computation is required for both signs and exponents also. Hence, the processing speed is slow for computing the floating point numbers compared with fixed point numbers. Therefore, in order to increase the speed of operation for floating point addition in real time application i.e., to add 16-samples at a time which are in floating notation; a parallel and pipe line technique is going to be incorporated to the two bit floating point architecture. Before developing such novel architecture, a novel algorithm is developed and after, the novel architecture is developed. The total work is simulated by Modelsim 10.3c tool and synthesized by Xilinx 13.6 tool.*

***Keywords :** Over sample, Nyquist criterion, Floating point adder, floating point MAC, parallel and pipe line technique, novel floating point architecture.*

I. INTRODUCTION

The enlarge form of MAC is multiplier accumulator content. It is the main core of all the DSP processors. FIR or IIR filters are used to design the MAC. Depending on the user application FIR or IIR filters are used. To design DSP processor, the sub-systems of the MAC are multiplier, adder and shifter. DSP processors are two types i.e., fixed point DSP processors and floating point DSP processors. If the speed is major concern fixed point DSP processors are used where as accuracy is major concern then floating point DSP processors are used. The signal which is to be processed by the DSP processor is to be sampled first as per the Nyquist criterion which says that in order to reconstruct the signal at the output, the input signal is to be oversampled[2] which means that the sampling rate fs should be greater than the twice to that of the fundamental frequency fm. In this work the input signal is sampled for 16 samples. The sampled signals are in floating point. A 16 bit floating point number can be represented by 1 bit of mantissa, 4 bits of exponent and 11 bits of mantissa or fraction. In the general operation of MAC, after receiving the first sample, it will be multiplied by the filter co-efficient and then the result is sent to the adder to add it to the previous value. After addition the output appears as y(n). To get the next sample, a shifting operation is used for the x(n). Like this to add two floating point numbers in the serial fashion an architecture already has been developed in the traditional work. But to add 16 floating point samples at a time, a novel algorithm and its corresponding architecture is certainly required for the DSP applications to increase the speed of operation of floating point processors. Figure1 shows the block diagram of general floating point MAC, in which h(n) indicates filter coefficients.

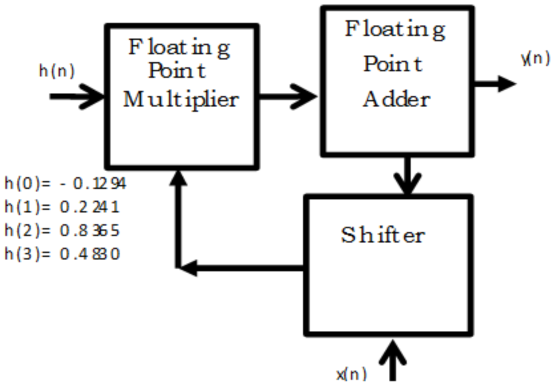


Figure1: floating point MAC

II. TRADITIONAL FLOATING POINT ADDITION

Figure2 shows the traditional or standard floating point adder. Traditional floating point adder operates in serial fashion[3]. After adding the first two numbers, the result is added to the third sample or third floating point value in the serial manner. Similarly this result is to be added to the fourth sample in the same serial fashion. Like this to add all 16 samples sixteen clock cycles are definitely required by the traditional floating point adder. The traditional floating point adder working principle is well known. It shows the Sx, Sy; Ex, Ey; and Fx, Fy.

Revised Manuscript Received on December 30, 2019.
* Correspondence Author

Dr. R. Prkash Rao*, Assoc. Professor, Department of ECE, Matrusri Engineering, College, Saidabad, Hyderabad, India.
Email: prakashiits@gmail.com

S indicates the Sign, E indicates the Exponent and F indicates the Fraction or Mantissa. Similarly the suffix x indicates first floating point number and the suffix y indicates second floating point number. Sz is the resultant sign bit, Ez is the resultant exponent and Fz is the resultant fraction or mantissa[4].

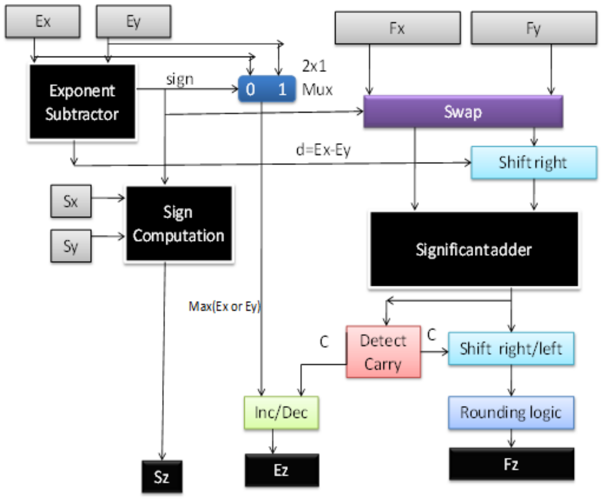
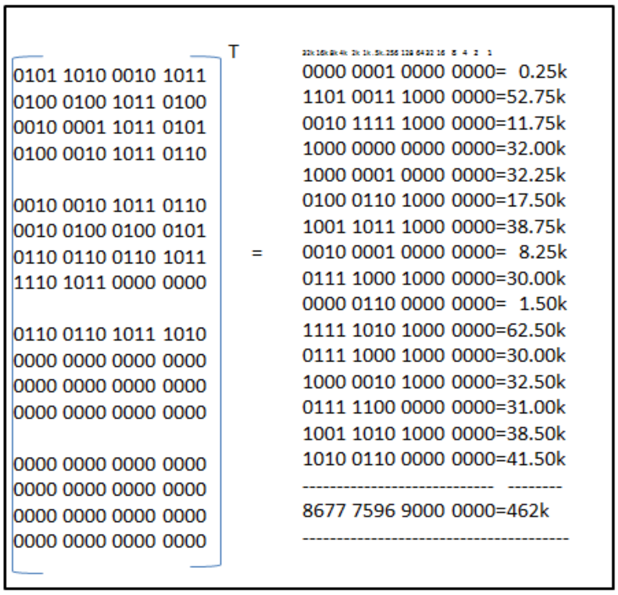


Figure2: traditional or standard floating point adder

III. NOVEL ALGORITHM

(Parallel and Pipeline processing of 16-bt floating point Numbers):

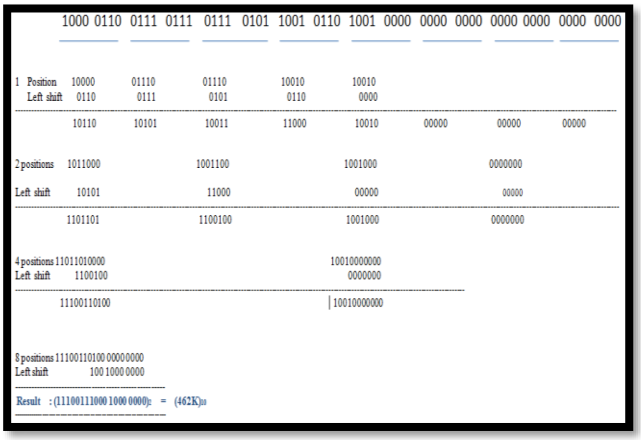
Figure3 shows transpose and computation of 16 floating point values (each floating point value is having 16-bit length) for parallel and pipe line processing. By transposing the same sample one's can be added. The 16 floating point values are stored in the 16 x16 matrix. Within the matrix, each floating point value indicates the value of one sample. In order to count the number of 1's within the same sample, the floating point value is to be transposed. Like this, to count the number of 1's within the respective sample of all the floating point values, the whole 16x16 matrix has to be transposed. For the example shown in Figure3, after transpose, the obtained decimal value is $(462K)_{10}$.



```
                          T    32k 16k 8k 4k 2k 1k .5k 256 128 64 32 16 8 4 2 1
0101 1010 0010 1011            0000 0001 0000 0000=  0.25k
0100 0100 1011 0100            1101 0011 1000 0000=52.75k
0010 0001 1011 0101            0010 1111 1000 0000=11.75k
0100 0010 1011 0110            1000 0000 0000 0000=32.00k
                               1000 0001 0000 0000=32.25k
0010 0010 1011 0110            0100 0110 1000 0000=17.50k
0010 0100 0100 0101            1001 1011 1000 0000=38.75k
0110 0110 0110 1011       =    0010 0001 0000 0000=  8.25k
1110 1011 0000 0000            0111 1000 1000 0000=30.00k
                               0000 0110 0000 0000=  1.50k
0110 0110 1011 1010            1111 1010 1000 0000=62.50k
0000 0000 0000 0000            0111 1000 1000 0000=30.00k
0000 0000 0000 0000            1000 0010 1000 0000=32.50k
0000 0000 0000 0000            0111 1100 0000 0000=31.00k
                               1001 1010 1000 0000=38.50k
0000 0000 0000 0000            1010 0110 0000 0000=41.50k
0000 0000 0000 0000            --------------------------  --------
0000 0000 0000 0000            8677 7596 9000 0000=462k
0000 0000 0000 0000            ------------------------------------
```

Figure 3: transpose of all 16-Sample values

Figure 4 shows the parallel and pipeline processing through the counter-cum-adder module. After transpose, the number of 1's are counted by the counter module through the parallel operation. For example, in figure3 after transpose first column consists of 8 one's . second column consists of 6 one's third, fourth, fifth columns consists of 7 one's etc. Its total binary value is 462 k . The same sequence of 8677 7596 9000 0000 are taken in binary(1000 0110 0111 0111 0111 0101 1001 0110 1001 0000 0000 0000 0000 0000 0000 0000) and be processed in parallel and pipe line operation and we can observe that the same result of $(462K)_{10.}$



```
        1000 0110  0111  0111   0111  0101  1001  0110  1001  0000  0000  0000  0000 0000  0000  0000

1  Position   10000      01110      01110       10010      10010
   Left shift  0110       0111       0101        0110       0000
-----------------------------------------------------------------------------------------------
              10110      10101      10011       11000      10010      00000      00000      00000

2 positions  1011000               1001100                1001000                0000000

Left shift     10101                 11000                  00000                  00000
-----------------------------------------------------------------------------------------------
             1101101               1100100                1001000                0000000

4 positions 11011010000                                  10010000000
Left shift     1100100                                       0000000
---------------------------------------------------------------------------
            11100110100                                  |10010000000

8 positions 11100110100 0000 0000
Left shift           100 1000 0000
-----------------------------------
Result   : (11100111000 1000 0000)2  =  (462K)10
-----------------------------------
```

Figure 4 parallel and pipeline processing

It is clear that at the end of the parallel operation, the final count value will be passed to the pipe-line section. For the first clock pulse, one bit position of the first and the alternate numbers are left shifted. The immediate number will be added to the left shifted number. In the similar way during the second clock cycle, 2 bit positions; during the third clock cycle, 4 bit positions and in the fourth clock cycle, 8 bit positions are left shifted. In each level, after left shifted, the immediate number will be added. Figure4 shows the parallel and pipeline processing for the example of figure3. The answer through the parallel and pipeline processing is same as the example shown in Figure3 which is same as $(462K)_{10}$. Hence it is proven. The same algorithm is implemented as architecture in Figure 5.

IV. HARDWARE FOR NOVEL ALGORITHM:

The architecture of novel floating point adder requires insertion sort module, counter cum adder module and normalize module as shown in below figure 5.

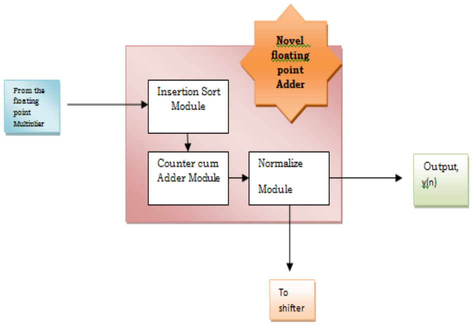


Figure 5: architecture of novel floating point adder

Retrieval Number: B4228129219/2019©BEIESP
DOI: 10.35940/ijeat.B4228.129219
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Insertion sort module:

While operating with floating point numbers, the operation is to be started with maximum number.
But, after sampling, all the 16-floating point numbers are not in the ascending or descending order. Hence, to get the order of all the 16 samples insertion sort module is used.

Counter cum Adder module:

After the multiplication operation, the resultant 16 floating point values are stored in the buffer. By applying insertion sort algorithm, the 16 ascending order samples are applied to the 16 counters. Each sample is having 16 bits hence 4-bit counter(2^4=16) is used for each sample to count the number of bits. Figure 6. operates as per the algorithm explained before.

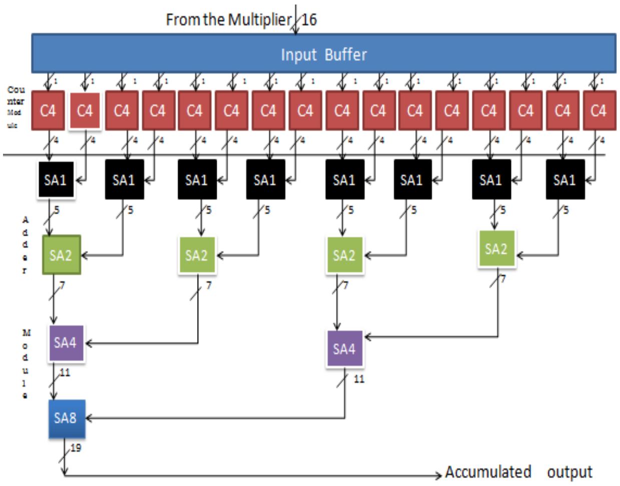


Figure6: counter cum adder module

Normalization module:

Figure7 explains the normalization operation. In floating point operations at the end to get the correct sign, exponent and mantissa normalization operation is required. After counter cum adder module operation sign bit(Sz) will be generated. The output of the counter cum adder module is applied to the shift register to find the exponent(Ez) and fraction(Fz).

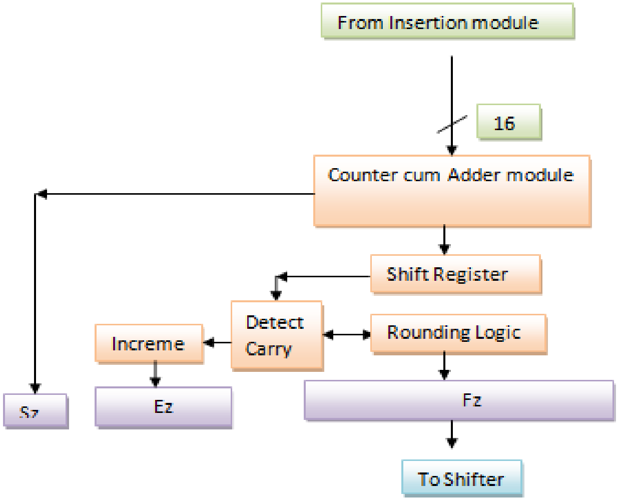
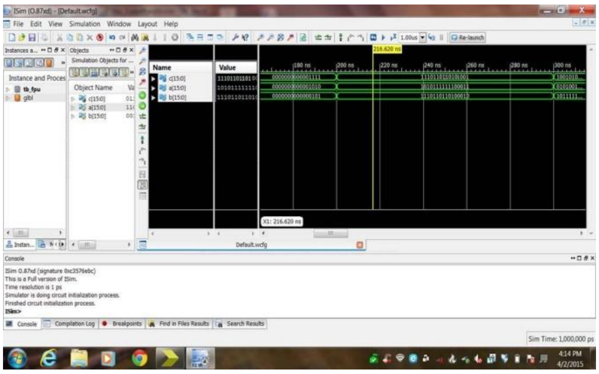


Figure7: Normalization module

V. RESULTS AND DISCUSSION

Figure8 explains simulation results of traditional or Standard Floating Point Adder. The simulation results for the traditional floating point adder has been performed by the Modelsim 10.3c tool. As the Modelsim tool does not contain floating point packages, we have developed floating point library packages and added with the existed library of the Modelsim tool 10.3c. Hence, by sending the 16-bit floating point values as input, the output is attained.



```
ISim 0.87xd (signature 0xc3576ebc)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Figure8: simulation results of traditional or Standard Floating Point Adder

Figure9 gives the simulation results of the Novel floating point Adder.

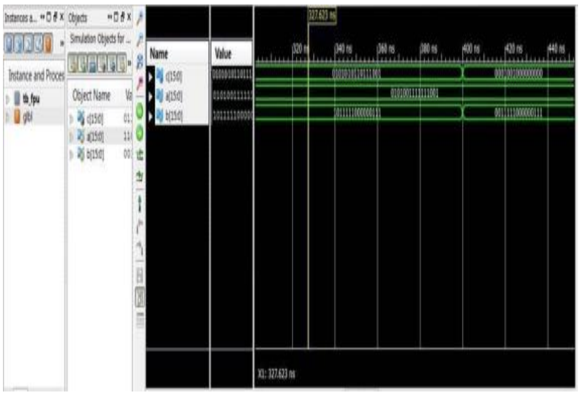


Figure9: simulation result of Novel floating point Adder

The synthesis has been done by XST tool. The chosen hardware device is Xc2s50e-ft256-6 with speed grade of -6. Table.1 and graph.1 show the results for various parameters of standard floating point adder i.e., addition with serial fashion and novel floating point adder i.e., addition with parallel and pipeline fashion. The input-output pins taken by standard floating point adder and novel floating point adder are 17% . The number of basic elements taken by standard floating point adder are 3.4% and novel floating point adder are 4.16%. The delay of the traditional floating point adder is 0.983ns and the delay of the novel floating point adder is 0.729ns. The speed attained by novel floating point adder is 34.85% more compared with the standard floating point adder which is due to the incorporation of parallel and pipe line architecture. The power consumption for traditional adder is 9.726mW and for novel adder is 11.726mW.

Retrieval Number: B4228129219/2019©BEIESP
DOI: 10.35940/ijeat.B4228.129219
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Table1: synthesis report

Hardware Resources/ Parameters	Standard floating point Adder(addition is in the serial fashion)	Novel floating point Adder(addition is in the parallel and pipe line fashion)
Number of IOs	31 out of 182(17%)	31outof182 (17%)
Number of BELs	59 out of 1728(3.4%)	72 out of 1728(4.16%)
Minimum period	0.983 ns	0.729 ns (25.83%)
Max. Frequency	1017.2 MHz	1371.7MHz (34.85%)
Power consumption	9.726mw	11.726mw (20.56%)

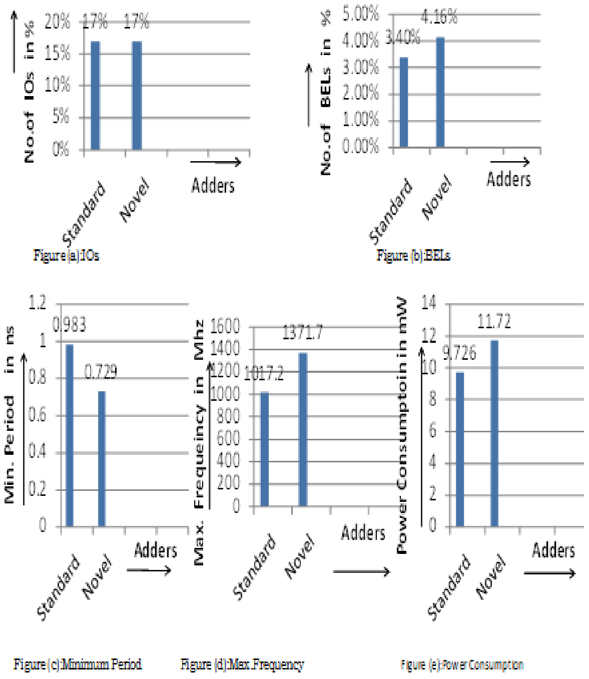


Figure (a):IOs

Figure (b):BELs

Figure (c):Minimum Period

Figure (d):Max.Frequency

Figure (e):Power Consumption

Graph1: synthesis results

VI. CONCLUSION

In this work both the standard floating point adder and novel floating point adder are synthesized with the Xilinx IST 13.6 software tool. Simulation is done with Modelsim 10.3c software. Modelsim 10.3c tool is not having floating point packages. Hence, a specially developed IEEE 754 standard floating point packages are compiled with the tool and then the synthesis has been performed. The speed of the novel floating point adder is 34.85% more compared with the traditional floating point adder but with the scarifying of some power consumption which is in the order of 20.56%.

REFERENCES:

1. Pramod Kumar Meher, Senior Member, IEEE, “New Approach to Scalable arallel and Pipelined Realization of Repetitive Multiple-Accumulations”, Submitted To Ieee Transactions On Circuits And Systems-Ii: Express Briefs
2. Israel Koren, Computer Arithmetic Algorithms, A K Peters, second edition, 2002.
3. J. Hennessy and D. A. Peterson, Computer Architecture a Quantitative Approach, Morgan Kauffman Publishers, second edition, 1996.
4. M. Karthik kumar, D.Manoranjitham, K.Praveen kumar, “Implementation of Efficient 16-Bit MAC Using Modified Booth Algorithm and Different Adders”, International Journal Scientific and Research Publications, Volume 4, Issue 3, March 2014, ISSN 2250-3153.

AUTHOR’S PROFILE

Dr. R. Prakash Rao, received Ph.D from JNTUH, Hyderabad; M.Tech from College of Engineering Andhra University,Vizag and B.Tech from Nagarjuna University, Guntur, India. He has 14 years of teaching experience. He worked as Professor & HOD in the department of ECE, St.Peter‟s Engineering College, Hyderabad. Presently he is working as Associate Professor in Matrusri Engineering College, Saidabad, Hyderabad. He published 18 papers in various National and International Journals and Conferences. He got best faculty award for the A.Y 2017-2018 by the Rotaract club, Secundrabad. He has guided 14 M.Tech projects and about 30 B.Tech projects in various fields. His research interest includes VLSI, VLSI with signal processing, Microwave Engineering and Embedded Systems.