Design of dexterous VLSI architecture of FFT using verilog

[1] P.Thirunavukku arasi,[2]P.S.Stellabai,
[1]Electronics and Communication Engineering,
[1]Holy cross Engineering College,Tuticorin,India.

***Abstract*— Fast Fourier Transform (FFT) having a considerable impact on the performance of communication system has been a hot topic of research for many years. Fixed point numbers are most widely involved in FFT computation, in which the accuracy is one of the most important backlogs. Thus the floating point operation was included in butterfly unit. However, FP butterfly architecture has the main disadvantage is slow as compared with the fixed point computation. In this paper, the floating point fused dot product unit is used for latency reduction. Efficient Radix8 butterfly unit with FDPA was proposed. This butterfly unit performs faster than the conventional butterfly. Finally, the performance analysis of various Radix architectures such as Radix2, Radix4 and Radix8 are shown.Radix8 requires less number of adders and multipliers when compared to the other lower order radices. Thus the Radix8 architecture was chosen, in which the FDPA algorithm was implemented. Hence the speed of FFT computation with FP multiplier can be enhanced.

***IndexTerms*—** Fused Dot Product Add (FDPA), Fast Fourier Transform (FFT), Floating Point multiplier (FP).

I. INTRODUCTION

Traditional Fast Fourier Transform (FFT) plays a great role in various applications such as OFDM, Filtering, spectral analyzer, Radar etc. The FFT computation is carried out by resolving the butterfly units. The major task is to include the floating point operation in FFT architecture. There are various FFT Radix architectures are available. The number of multipliers and adders required for the various Radix structures are varied. Thus the efficient Radix8 structure was chosen in which the floating point operation was included. But the latency rose tremendously when compared to the traditional Radix8. Hence, the enhanced multiplication technique called FDPA is used. By using XILINX and ModelSim, the dexterous VLSI architecture of FFT was implanted. In addition to that, the comparison between the various Radix architectures in terms of Timing constraints is shown. The performance analysis gives the complete information about the proposed system speed improvement.

II. BASIC RADIX2 ARCHITECTURE

In 1965, Fast Fourier Transform was developed by J.W.COOLEY and J.W.TUCKEY. The FFT is based on decomposition and breaking the samples into smaller units and combining them to obtain the frequency domain signal. First, here is the simplest butterfly. It's the basic unit, consisting of just two inputs and two outputs. If N is power of 2, it can divide DFT and more. Finally, it divide several DFT (butterfly computation) of N=2.

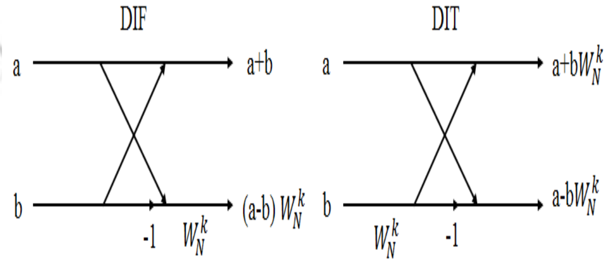


Figure.1 Radix2 Butterfly Diagram

Similarly, there are various butterfly units with different Radices are shown as below. In Radix4, four inputs and four outputs are available. Let us consider N be the number of samples, then it is splitted as N/4 in Radix4 and N/8 in Radix8 respectively.

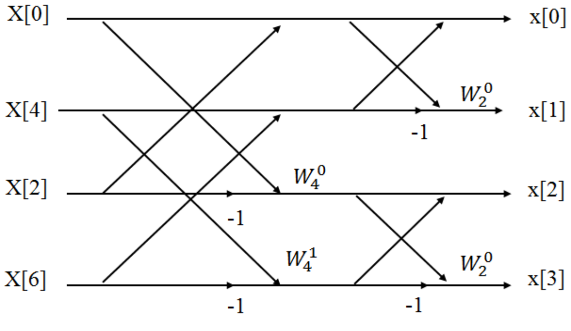


Figure.2 Radix4 Butterfly Diagram

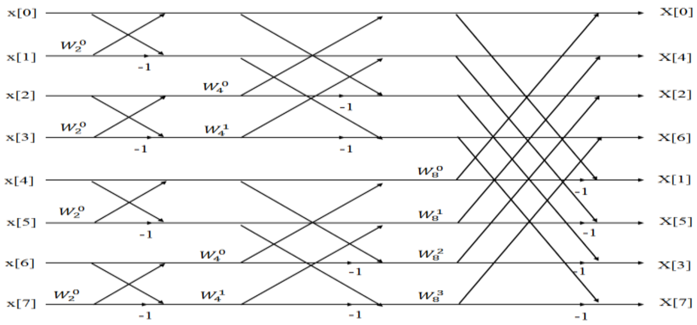


Figure.3 Radix8 Butterfly Diagram

III. FLOATING POINT OPERATION

Floating point number consists of exponent, mantissa and significant. IEEE 754 standards are used to represent the given input in 32 bit floating point numbers. In which single precision is carried out, thus the 32bit output can be obtained. Floating point number uses one bit for sign bit, 23 bits for mantissa and 8 bits for exponents.

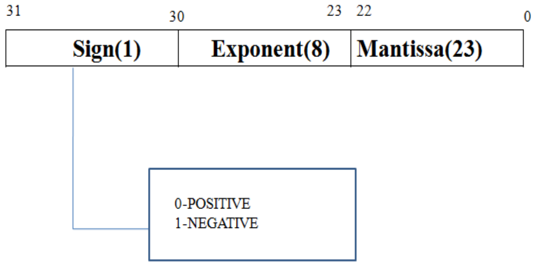


Figure.4 Floating Point Representation

The sign bit is used to indicate the nature of the number i.e, either positive or negative. If the sign bit is 0 then the number will be positive otherwise it will be negative.

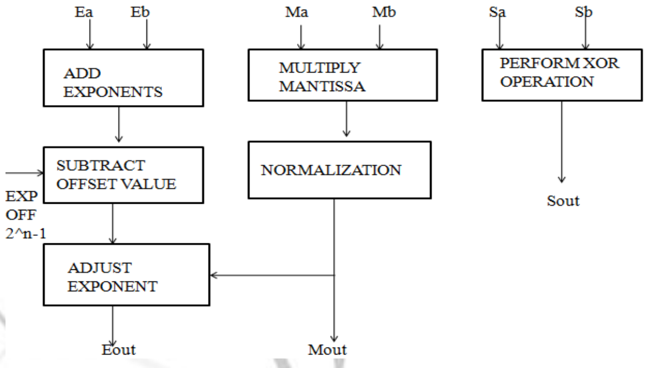


Figure. 5 Floating Point Multiplication

The mantissas are first multiplied together with an unsigned integer multiplier. The exponents are added, and the excess value (exponent_offset) 2(*n*–1) is subtracted from the result. The sign of the output (s_out) is the XOR of the signs of the inputs (sa and sb). After multiplication has taken place, the post-normalizer normalizes the result, if necessary, by adjusting the mantissa and exponent of the result to ensure that the MSB of the mantissa is 1.To normalize the mantissa, the mantissa is shifted left (i.e., the mantissa is multiplied by powers of 2). For each bit shifted left, the exponent must be reduced by 1.

IV. EXISTING SYSTEM

Fused dot product allows the difference of the product too along with addition that is useful in the implementation of complex multiplication. Fused floating point operation is the fused add-subtract operation. Computing the sum and difference of two floating point operands is used many times in DSP algorithms. Fused add-subtract unit performs the addition and subtraction on same pair of floating point operands in parallel fashion.

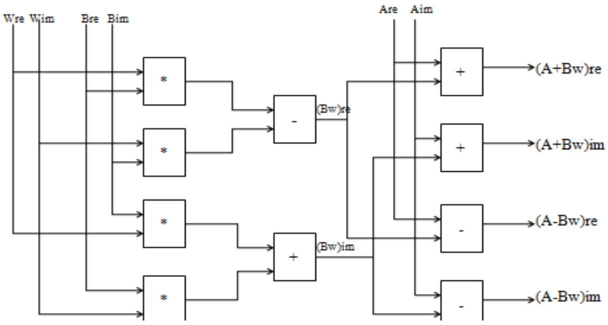


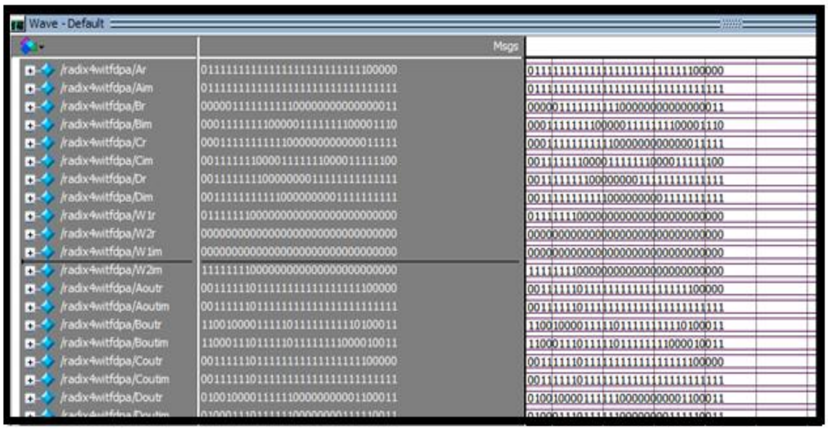
Figure.6 FFT Butterfly unit with complex number

In the existing system,the Fused Dot Product Add unit is implemented in the basic FFT Radix2 architecture.Floating point operation was also included in the Radix architecture.In which the latency is reduced to some extent.

V. PROPOSED SYSTEM

FDPA unit in Rdix4

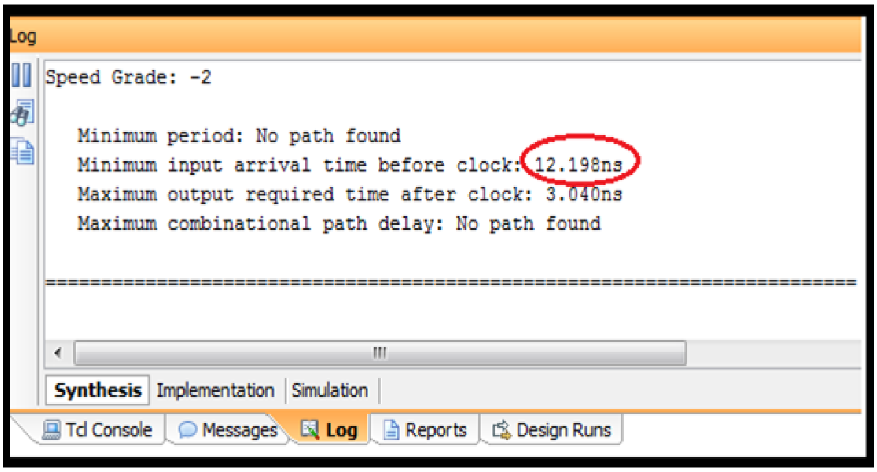
Radix4 is an efficient FFT architecture, in which the number of adders and multipliers required are reduced. Four inputs and four outputs are available.



```
/radix4witfdpa/Ar      01111111111111111111111111100000
/radix4witfdpa/Aim     01111111111111111111111111111111
/radix4witfdpa/Br      00000111111111100000000000000011
/radix4witfdpa/Bim     00011111111000001111111100001110
/radix4witfdpa/Cr      00011111111111000000000000011111
/radix4witfdpa/Cim     00111111100001111111000011111100
/radix4witfdpa/Dr      00111111111000000000111111111111
/radix4witfdpa/Dim     00111111111110000000001111111111
/radix4witfdpa/W1r     01111111000000000000000000000000
/radix4witfdpa/W2r     00000000000000000000000000000000
/radix4witfdpa/W1im    00000000000000000000000000000000
/radix4witfdpa/W2im    11111111000000000000000000000000
/radix4witfdpa/Aoutr   00111111011111111111111111100000
/radix4witfdpa/Aoutim  00111111011111111111111111111111
/radix4witfdpa/Boutr   11001000011111011111111110100011
/radix4witfdpa/Boutim  11000111011111011111111100010011
/radix4witfdpa/Coutr   00111111011111111111111111100000
/radix4witfdpa/Coutim  00111111011111111111111111111111
/radix4witfdpa/Doutr   01001000011111100000000001100011
/radix4witfdpa/Doutim  01000111011111110000000011110011
```

Figure.7 Simulated output of Radix4 with FDPA

In the above figure, W1r,W1im,W2r,W2im are the real and imaginary twiddle factors. The twiddle factors are nothing but the trignometric coefficients in FFT computation.

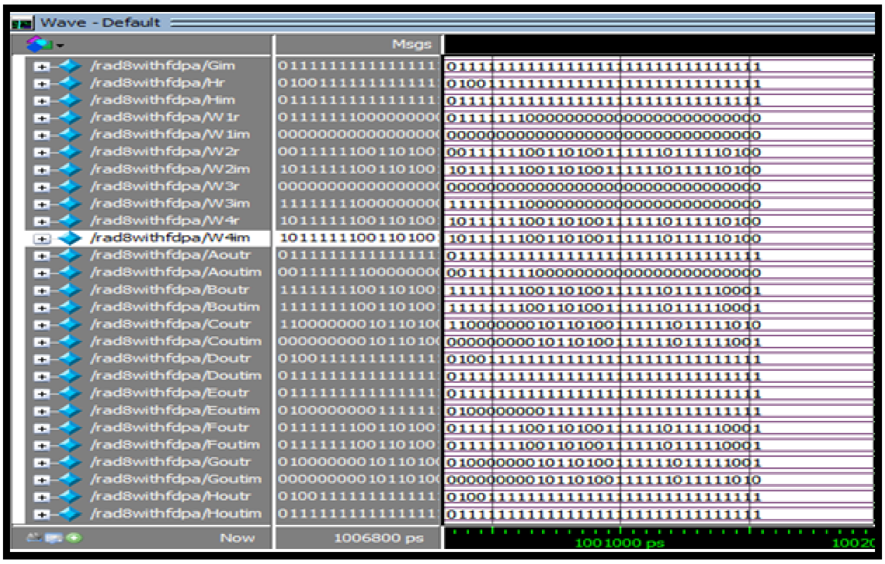


```
Speed Grade: -2

   Minimum period: No path found
   Minimum input arrival time before clock: 12.198ns
   Maximum output required time after clock: 3.040ns
   Maximum combinational path delay: No path found

==============================================================================
```

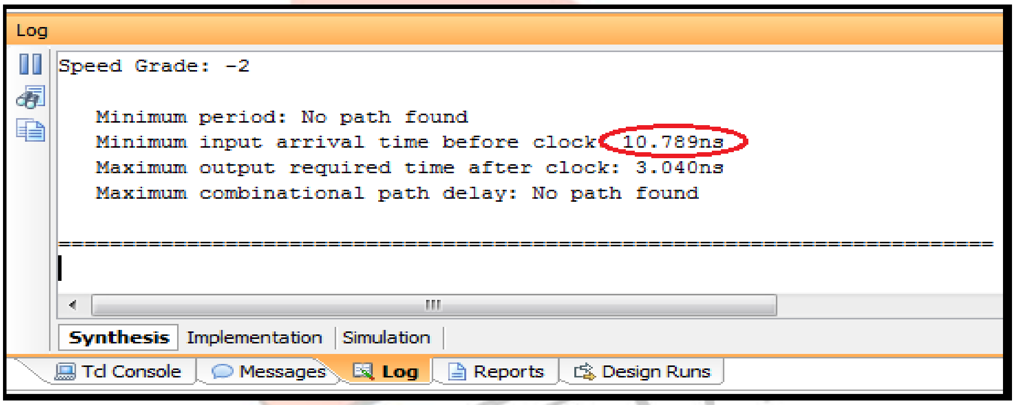
Figure.8 Timing constraints in Radix4 with FDPA



```
/rad8withfdpa/Gim      0111111111111111
/rad8withfdpa/Hr       0100111111111111
/rad8withfdpa/Him      0111111111111111
/rad8withfdpa/W1r      0111111100000000
/rad8withfdpa/W1im     0000000000000000
/rad8withfdpa/W2r      0011111100110100
/rad8withfdpa/W2im     1011111100110100
/rad8withfdpa/W3r      0000000000000000
/rad8withfdpa/W3im     1111111100000000
/rad8withfdpa/W4r      1011111100110100
/rad8withfdpa/W4im     1011111100110100
/rad8withfdpa/Aoutr    0111111111111111
/rad8withfdpa/Aoutim   0011111110000000
/rad8withfdpa/Boutr    1111111100110100
/rad8withfdpa/Boutim   1111111100110100
/rad8withfdpa/Coutr    1100000000101101
/rad8withfdpa/Coutim   0000000000101101
/rad8withfdpa/Doutr    0100111111111111
/rad8withfdpa/Doutim   0111111111111111
/rad8withfdpa/Eoutr    0111111111111111
/rad8withfdpa/Eoutim   0100000000111111
/rad8withfdpa/Foutr    0111111100110100
/rad8withfdpa/Foutim   0111111100110100
/rad8withfdpa/Goutr    0100000000101101
/rad8withfdpa/Goutim   0000000000101101
/rad8withfdpa/Houtr    0100111111111111
/rad8withfdpa/Houtim   0111111111111111
Now                    1006800 ps
```

Figure.9 Simulated outut of Radix8 with FDPA

The above Figure.9 represents the simulated output of Radix8 with FDPA. In which 'Are', 'Bre', 'Cre', 'Dre', 'Ere', 'Fre', 'Gre', 'Hre', 'Aim', 'Bim', 'Cim', 'Dim', 'Eim', 'Fim', 'Gim', 'Him' are inputs. W1re, W2re, W3re, W4re, W1im, W2im, W3im, W4im are twiddle factor which is also one of the inputs. 'Aoutr', 'Boutr', 'Doutr', 'Eoutr', 'Foutr', 'Goutr', 'Houtr', 'Aoutim', 'Boutim', 'Doutim', 'Eoutim', 'Foutim', 'Goutim' and 'Houtim' represents the outputs.



```
Speed Grade: -2

   Minimum period: No path found
   Minimum input arrival time before clock: 10.789ns
   Maximum output required time after clock: 3.040ns
   Maximum combinational path delay: No path found

============================================================================
```

Figure.10 Timing constraints in Radix8 with FDPA

VI. ACKNOWLEDGMENT

The authors would like to acknowledge the valuable suggestions from our dedicated supervisor Miss.D.Sindhuja.,M.E and our beloved HOD sir Mr.A.Jeyamurugan.,M.E.

REFERENCES

[1] Kornerup P, "Correcting the normalization shift of redundant binary representation," IEEE Trans, comput., vol.58, no,10,pp,1435-1439,2001

[2] Cooley J.W and Tukey J.W, "An algorithm for the machine calculation of Complex Fourier Series ," Math. Comput., vol.19,no.90,pp.297-301,1965

[3] Min J.H, Kim S.W and Swartzlander E.E, Jr, "A Floating point Fused FFT butterfly arithmetic unit with merged multiple constant multipliers," In Proc. 45th Asilomar Conf. Singnals, Syst,Comput,2011

[4] Parhami B, "Computer arithmetic: Algorithm and hardware," 2nd edition," New York ,NY,USA: Oxford Univ.Press 2010.

[5] Sohn J and swartlander E.E, Jr.(2013), "Improved architecture for a floating pointfused dot product unit," in proc .IEEE 21st Symp.Comput. Arithmatic. pp.41-48.

.