IETE 46th Mid Term Symposium "Impact of Technology on Skill Development" MTS- 2015
Special Issue of International Journal of Electronics, Communication & Soft Computing Science and Engineering, ISSN: 2277-9477

Design and Implementation of FFT

Ms. Sneha Kherde

Abstract— **Fast Fourier Transforms are algorithms for quick calculation of discrete fourier transform of data vector. In this paper, we had designed an 8-point FFT and its computation time. Here we had simulated and implemented Radix-2 8-point FFT in hardware using hardware language (VHDL) here time constraint is measured with the help of ModelSim SE.**

Index Terms— **DFT, FFT, FPGA, butterfly, radix.**

I. Introduction

The fourier transform is the method of changing time representation to frequency representation. The discrete Fourier transform (DFT) is a one of the Fourier transform, used in Fourier analysis. It transforms one function that is time into another that is frequency, so as to get discrete signals, hence called the DFT, of the original function. The DFT of a given sequence x[n] can be computed using the formula

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad 0 \le k \le N-1$$
$$W_N = e^{-j2\pi/N}$$
…1

Where, W_N is twiddle factor. Twiddle factors referred to as the root-of-unity complex multiplicative constants in the butterfly operations of the FFT algorithm, used to recursively combine smaller discrete Fourier transforms. Practically, at the input there is time domain so only real values should be present But for our convenience we apply input data sequence *x*(*n*) having both real and imaginary term
We observe that for each value of *k*, direct computation of *X*(*k*) involves *N* complex multiplications (4*N* real multiplications) and *N*-1 complex additions (4*N*-2 real additions). Consequently, to compute all *N* values of the DFT requires N^2 complex multiplications and N^2-*N* complex additions [4][7].

The FFT uses a standard three-loop structure for the main FFT computation. A fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) and it's inverse. There are many distinct FFT algorithms involving a wide range of mathematics, from simple complex-number arithmetic to group theory and number theory. The Fast Fourier Transform is an optimized computational algorithm to implement the Discrete Fourier Transform to an array of 2^N samples where, N is the length of samples. It allows determining the frequency of a discrete signal, representing the signal in the frequency domain, convolution, etc. This algorithm has a complexity of O(N*log2(N)).The ordering minimizes the number of fetches or computations of the twiddle-factor values. Since the bit-reverse of a bit-reversed index is the original index, bit-reversal can be performed fairly simply by swapping pairs of data.

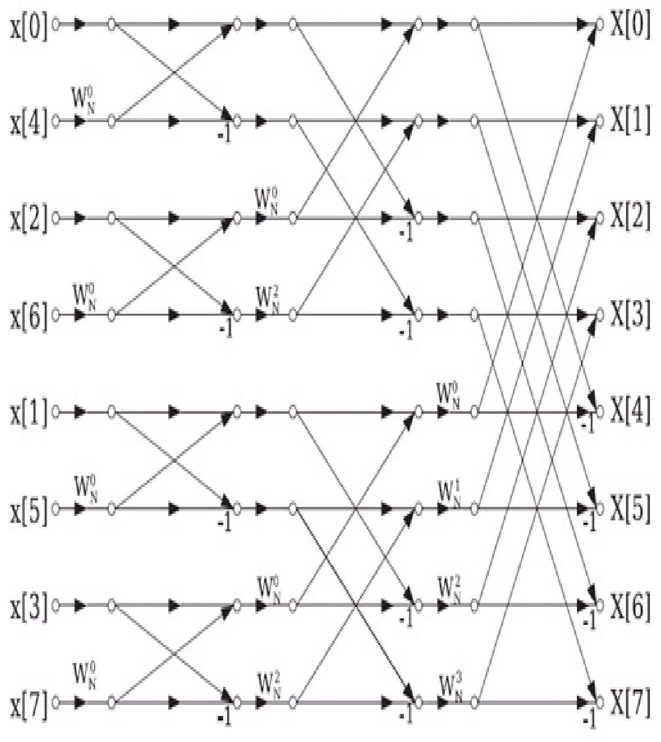

Fig1: Radix-2 decimation in time FFT algorithm for length 8-signal

"DFT" refers to a mathematical transformation or function, regardless of how it is computed, whereas "FFT" refers to a specific family of algorithms for computing DFT's. Fig 1 shows radix-2 decimation in time FFT algorithm for length 8-signal [2].This is built up of three stages whose first stage can be discussed further.

II. Radix-2 Dit Fft Algorithm For Length 8 Signals

With the introduction of field programmable gate arrays (FPGAs), it is feasible to provide hardware for application specific computation design. The changes in designs in FPGA's can be accomplished within a few hours, and thus result in significant savings in cost and design cycle [10]. FPGAs offer speed comparable to dedicated and fixed hardware systems for parallel algorithm [8].

The radix-2 decimation in time is applied recursively to the two length N/2 DFT's to save computation time. The full radix-2 decimation-in-time of length 8-signals is illustrated in figure-1, using the simplified butterflies. It involves M = log2N stages, each with N/2 butterflies per stage. Each butterfly requires one complex multiplier [3] and two adder per butterfly. The total cost of the algorithm is thus computational cost of radix-2 DIT FFT
- N/2log2N complex multipliers
- Nlog2N complex adders

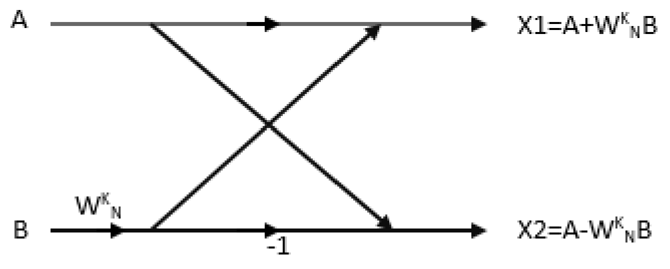

Fig 2 : Butterfly for radix-2 DIT FFT

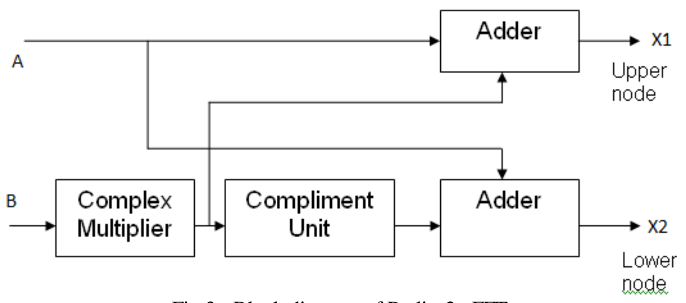

Fig 3 : Block diagram of Radix-2 FFT

Fig 2, shows butterfly diagram for radix-2 DIT FFT, whose block-diagram is as shown in fig-3. This is the first stage of FFT algorithm which is specified in fig 1. As we have seen earlier, there is the presence of complex number in twiddle factor so we need to use complex operation as in fig-3. Here again three blocks are used that is complex addition, complex subtraction, complex multiplication. The details of complex multiplier is shown in fig-4, where four simple multiplier, adder, subtractor has been used [9]. The implementation of a complex multiplier uses both the adders and subtractors needed for the FFT. The complex multiplier is designed, it in turn, contains the design for a complex adder and subtractor. The design of a FFT can be derived from the complex multiplier design [1].

For any complex number multiplier design, the most critical part is the multiplication process. In any multiplication operation, there are three major steps. For the first step, the partial products are generated. For the second step, the partial products are reduced to one row of final sums and carries. For the third step, the final sums and carries are added to generate the result.

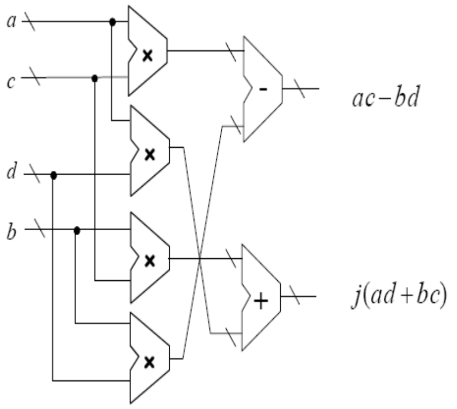

Fig 4: Schematic diagram of complex multiplier

Here the complex number multiplication can be divided into two main components known as real part and imaginary part as in [3]. This complex multiplier is shown by the following equation,

(a+jb)(c+jd) = (ac–bd) + j(bc + ad) …2

From equation 2, real part output is (ac – bd) and imaginary part output is (bc + ad). The real and imaginary parts of the output will be kept separate in design. Based on conventional methods and observation of equation 2, four separate multiplication are required to produce real part as well as imaginary part that is, this design requires four signed two's compliment multipliers, as well as a signed two's compliment adder and sub tractor, as shown in fig 4.

Multiplication has been done in such a way that "a" and "b" are always kept in multiplier and "c","d" are kept in multiplicand. By implementing this technique, several logic gates can be reduced. Complex multiplier finds extensive applications in areas like digital signal processing, communication etc. It is one of basic building block of digital system. The multiplier, adder and subtractor are all built using half adders and full adders. Half adders find the sum of two binary numbers and provide a sum and a carry. A full adder is similar to the half adder however, it has an input for a carry bit. Full and half adders can be constructed using logic gates; however, this is not the most efficient solution. This completes our first butterfly diagram refered in fig 2. The combination of these blocks can form complete FFT.

III. Simulation Result

The VHDL codes has been successfully simulated in ModelSim SE (version 6.2c) and can be synthesized using Xilinx ISE(version 6.2c). This paper has discussed about the DIT FFT algorithm. The output simulation result of X1 (Upper node) of radix-2 (fig-3) is as shown in fig-5. Here "ir1" and "im1"is a first input with real and imaginary part respectively. The output term is also a complex value denoted by "oreal" and "oimag".

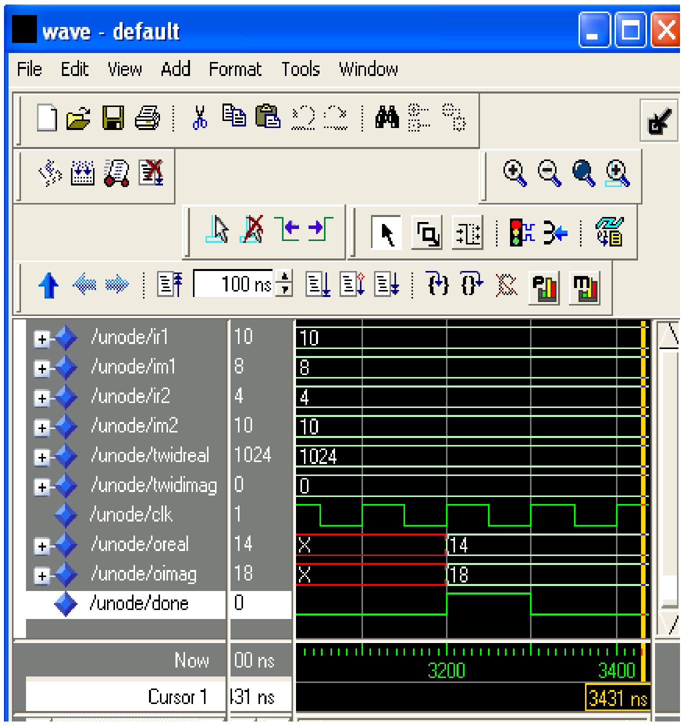
```
wave - default
/unode/ir1        10
/unode/im1        8
/unode/ir2        4
/unode/im2        10
/unode/twidreal   1024
/unode/twidimag   0
/unode/clk        1
/unode/oreal      14
/unode/oimag      18
/unode/done       0
Now               00 ns     3200   3400
Cursor 1          131 ns    3431 ns
```

Fig 5: Simulation result of X1 as shown in fig 3

Similarly, output simulation result of X2 (lower node) of radix-2 (fig-3) is shown in fig 6. These are the result of 1st stage FFT which is as shown in Fig 1.

The simulation result of complete FFT that is the combination of 1st stage, 2nd stage and 3rd stage is as shown in figure below. This is the combination of 16-bit real number and 16-bit imaginary number. Simulation of all these stages together gives reduction of time.

Fig 7 specifies the output of FFT obtained at 238 nanosecond.

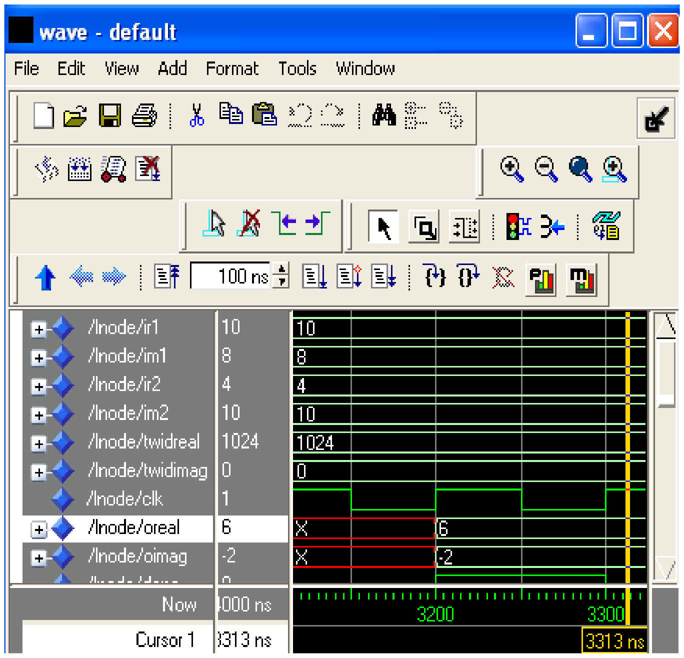
```
wave - default
/lnode/ir1        10
/lnode/im1        8
/lnode/ir2        4
/lnode/im2        10
/lnode/twidreal   1024
/lnode/twidimag   0
/lnode/clk        1
/lnode/oreal      6
/lnode/oimag      -2
Now               000 ns    3200   3300
Cursor 1          3313 ns   3313 ns
```

Fig 6 : Simulation result of X2 as shown in fig 3

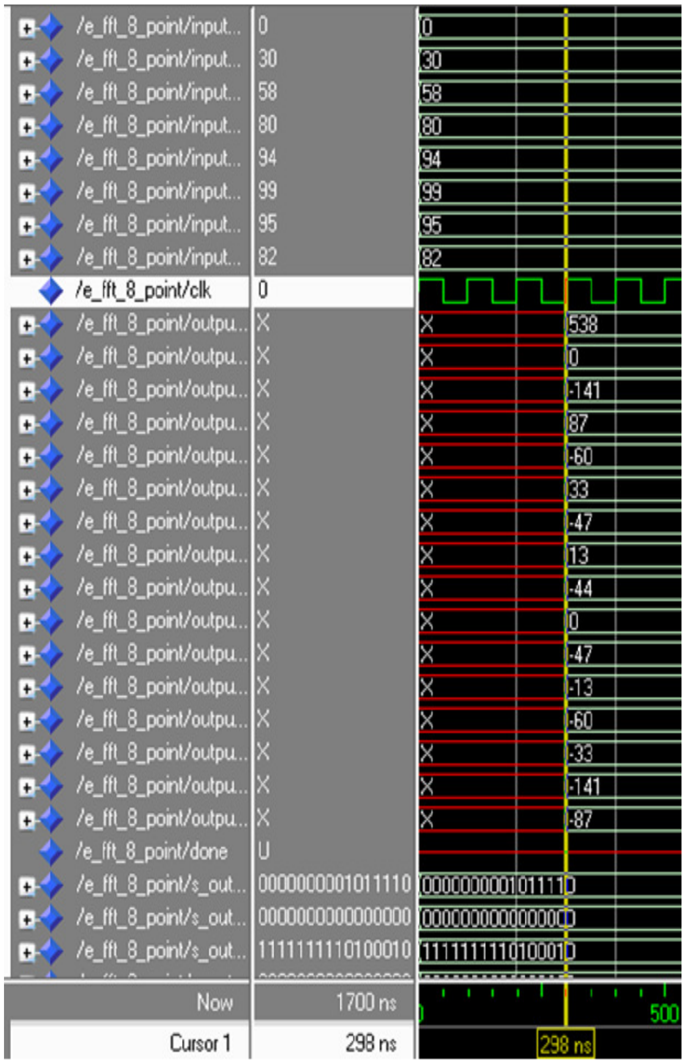
```
/e_fft_8_point/input...   0
/e_fft_8_point/input...   30
/e_fft_8_point/input...   58
/e_fft_8_point/input...   80
/e_fft_8_point/input...   94
/e_fft_8_point/input...   99
/e_fft_8_point/input...   95
/e_fft_8_point/input...   82
/e_fft_8_point/clk        0
/e_fft_8_point/outpu...   X    538
/e_fft_8_point/outpu...   X    0
/e_fft_8_point/outpu...   X    -141
/e_fft_8_point/outpu...   X    87
/e_fft_8_point/outpu...   X    -60
/e_fft_8_point/outpu...   X    33
/e_fft_8_point/outpu...   X    -47
/e_fft_8_point/outpu...   X    13
/e_fft_8_point/outpu...   X    -44
/e_fft_8_point/outpu...   X    0
/e_fft_8_point/outpu...   X    -47
/e_fft_8_point/outpu...   X    -13
/e_fft_8_point/outpu...   X    -60
/e_fft_8_point/outpu...   X    -33
/e_fft_8_point/outpu...   X    -141
/e_fft_8_point/outpu...   X    -87
/e_fft_8_point/done       U
/e_fft_8_point/s_out...   0000000001011110
/e_fft_8_point/s_out...   0000000000000000
/e_fft_8_point/s_out...   1111111110100010
Now                       1700 ns    500
Cursor 1                  298 ns     298 ns
```

Fig 7: Input and Output of FFT

IV. Conclusion

An FFT computes DFT and produces exactly same results as DFT. In this paper the functionality of Fast Fourier Transform was done in VHDL by ModelSim SE 6.3f version. The multipliers, adders, and compliment unit were implemented. We have seen single butterfly suited for FPGA implementation. Thus we had simulated the result of complete FFT where time required is 238ns.

In comparision of the arithmetic operation counts, the 8-point FFT requires more time than above implemented FFT. Hence it reduces time of computation and improves speed.

REFERENCES

[1] Ahmed Saeed, M. Elbably, G. Abdelfadeel, and M. I. Eladawy,"Efficient FPGA implementation of FFT\IFFT Processor", International Journal Of Circuits, Systems And Signal Processing.

[2] Saad Bouguezel, M. Omair Ahmad,"IMPROVED RADIX-4 AND RADIX-8 FFT ALGORITHMS" IEEE. Department of Electrical and Computer Engineering Concordia University 1455 de Maisonneuve Blvd. West Montreal, P.Q., Canada.

[3]Rizalafande Che Ismail and Razaidi Hussin "High Performance Complex Number Multiplier Using Booth-Wallace Algorithm" School of Microelectronic Engineering Kolej University Kejuruteraan Utara Malaysia.

[4] J.G.Proakis and D.G.Manolakis, "Digital Signal Processing, Principles,algorithms and applications" Prentice Hall India Publication.

[5] J.A.Hidalgo,V.Moreno-Vergara,O.Oballa, "A Radix-8 Multiplier Unit Design For Specfic Purpose",Dept of the Electronica , E.T.S.I.Industriales.

[6] C. S. Burrus and T. W. Parks,"DFT/FFT and Convolution Algorithms", New York,NY : John Wiley,1985.

[7] A. Saidi , "Generalized FFT algorithm",Proc. ICC,pp.227-231,May 1993.

[8] Robert Polge and Brooks Lawence, "Comparion of New Multiple Radix Fast fourier Number Theoetic Transform with FFT Algorithm in Terms of Performance and Hardware Cost", ECC Department, University of Alabama Huntsville, AL35899,IEEE : 744-749.

[9] Keiichi Satoh and Jubee Taba, "Complex Multiplier Suited for FPGA Structure", in Proc.ITC-CSCC, 2008, P.341-344.

[10] Xilinx Co."Xcell journal vol 58-59",2007 spring.

[11] Y.N.Chang and K.K.Parhi,"An efficient pipelined FFT architecture,"IEEE Trans. Circuits Systems II, vol. 50, pp.322-352, June 2003.