Fast Fourier Transformation Realization with Distributed Arithmetic

Renu Bala
Department of Electronics and Communication Engineering
Indira Gandhi Technical University Delhi, India

Shamim Aktar
Department of Electronics and Communication Engineering
JIIT, Noida, India

ABSTRACT
Fast Fourier transform (FFT) is a sound tool for computation of Discrete Fourier transform (DFT). It is widely used for analysis of digital spectrums, FIR filters, and autocorrelation and pattern recognition applications [1]. FFT is based on violation the input sample sequence into smaller sample sequences and mingling them to get the total output order or transform. FFT reduces the computation time required to compute a DFT and thus improves speed of computation. In this paper FFT computations will be done using a different method known as distributed arithmetic algorithm. Method is designed in VHDL. Simulation of the code is done in ModelSim 6.4.

General Terms
DFT, FFT, Distributed arithmetic, Model Sim.

Keywords
Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT), Distributed arithmetic (DA), Model Sim.

1. INTRODUCTION
Prevailing FFT processors use different transformation algorithms to analyze the signals. Signals are made of various fundamental signals with different frequencies, amplitudes and phases [1]. Frequency transformation is used for transformation of a signal from time domain to frequency domain. When a Fourier Transform is computed for a sample of discrete frequency sequence, it is called as a DFT. DFT involves large number of computations. These computations can further be reduced by FFT, another transforming algorithm.

FFT is used for computing the DFT of a finite series. It also reduces the number of computations required for evaluation of DFT.

The DFT of a sequence is written as,

$$X(k)=\sum_n x(n)e^{-j2\pi n/N}, \quad 0\le k\le N-1 \qquad (1)$$

Or,

$$X(k) = \sum_n x(n)(WN)^{nk} \qquad (2)$$

Where X (k) = Fourier transform of discrete time domain signal x(n),

K= 0, 1, 2 ……N-1, and

$(WN)^{nk}$ = twiddle factor

This expression is known as N-point DFT as this involves summation running for N points. Discrete time domain sequence can be computed from its frequency domain sequence using inverse of this method known as IDFT. In direct evaluation of DFT, to evaluate one value of X(k), it requires N multiplications and (N-1) complex addition. Thus, to evaluate N values of X(k), it require NxN multiplications and N(N-1) complex additions. Direct evaluation of DFT is inefficient in utilizing symmetry and periodicity properties of the twiddle factor. On the other hand FFT exploits the two basic properties of the twiddle factor which are as follows [1]:

$W_N^{k+N/2} = -W_N^k$ Symmetry property

$W_N^{k+N} = W_N^k$ Periodicity property

DFT can be represented in the matrix form as;

$$X_N = [W_N] * x_N \qquad (3)$$

$$\text{Where } X_N = \begin{vmatrix} X(0) \\ X(1) \\ \\ X(N-1)\ N*1 \end{vmatrix} \qquad (4)$$

$$\text{and, } x_N = \begin{vmatrix} X(0) \\ X(1) \\ \\ X(N-1)\ N*1 \end{vmatrix} \qquad (5)$$

$$\text{and } W_N^{kn} = \begin{array}{c} \\ k=0 \\ k=1 \\ \\ k=N-1 \end{array} \begin{array}{c} n=0 \quad n=1 \quad\quad n=N-1 \\ \begin{vmatrix} W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & \cdots & W^{N-1} \\ \vdots & \vdots & & \vdots \\ W^0 & W^{N-1} & \cdots & W_N^{(N-1)2} \end{vmatrix} \end{array} \qquad (6)$$

W_N^{kn} is represented as NxN matrix where each value of W_N^{kn} is calculated by multiplication of 'k' and 'n'. twiddle factor contains an exponent term which can be resolved by Euler's Identity.

Present paper worked on 8-point FFT. The structure obtained due to such divisions is similar to a butterfly and thus known as butterfly structure. The butterfly structure for an 8-point FFT is depicted in fig.1.

The block diagram of a 8-point FFT shows eight inputs. Each input has a real part and an imaginary part. Real and imaginary parts are used separately for resolving the butterfly structure. Thus FFT calculations were done for sixteen points (eight real and eight imaginary).

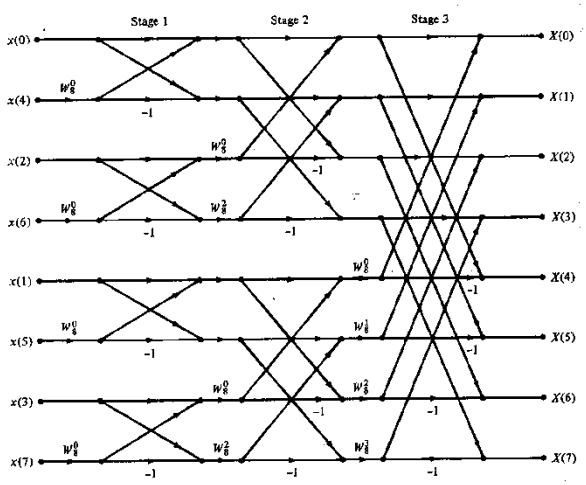

Figure 1. Eight-point decimation-in-time FFT algorithm

2. PERIODICITY PROPERTY OF TWIDDLE FACTOR

Twiddle factor (W_N^{kn}) repeats its value after some period of time as shown in table below. This shows values of twiddle factor at a time period of '8' repeats itself. This property is known as periodicity and twiddle factor is said to have periodic property [1].

Table1. Twiddle Factor Values

Kn	W_N^{kn}	VALUE
0	W_8^0	1 - j0
1	W_8^1	0.707 – j0.707
2	W_8^2	0 – j
3	W_8^3	-0.707 – j0.707
4	W_8^4	-1

And symmetric when it satisfies the following condition;

$$W_N^{k+N/2} = - W_N^k \qquad (7)$$

Based on these properties and values of twiddle factor, some of the calculations are repeated. Therefore the entire FFT requires $(N/2)\log_2 N$ complex multiplications and $N\log_2 N$ complex additions. The FFT is calculated by decomposing the given sequence of N-length into smaller sequences and then combining them to get the total transform. There are two algorithms for FFT, Decimation In Time (DIT) and Decimation In Frequency (DIF). In this paper DIT is considered.

3. BIT-REVERSAL

The input sequence has to be bit retreated so that output sequence is obtained in normal order. To do this the input sequence is separated into even and odd parts on the basis of index. Then even part is further separated into again even and odd parts [2]. Similarly same is done to the odd part. This process is continued until only two points are left at the end. In this paper FFT realization was done using DIF. The input samples are divided into group of even and odd sequence based on sample index position. Here the sequence is separated in time domain.

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)W_N^{2kn} + \sum_{n=0}^{N/2-1} x(2n+1)W_N^{k(2n+1)}$$

Even Odd

$$X(k) = \sum_{n=0}^{N/2-1} x(2n)W_{N/2}^{kn} + W_N^k \sum_{n=0}^{N/2-1} x(2n+1)W_{N/2}^{kn}$$

Separating the input sequence into even and odd sub-sequences continues until 2 point sequence is reached [2].This work is done on 8-point FFT. The structure obtained due to such divisions is similar to a butterfly and thus known as butterfly structure. The butterfly structure for an 8-point FFT is depicted in fig.1. The block diagram of a 8-point FFT shows eight inputs. Each input has a real part and an imaginary part. Real and imaginary parts are used separately for resolving the butterfly structure. Thus FFT calculations for sixteen points (eight real and eight imaginary) is needed.

4. FIXED-POINT NUMBERS

Many arithmetic computations deal with real numbers with fractional parts. When working with real numbers in VHDL, it does not result in a synthesizable design. So a representation for fractional numbers to design some synthesizable code is required. There are two options, fixed points and floating points. Fixed point number representation is used in arithmetic calculations for numbers having fixed number of digits [3]. These are used for representing the fractional values. Use of fixed point representation in arithmetic design improves performance and accuracy.

For getting the fixed point representation of a decimal number, say for example 0.707_{10}, The key point here is to keep a track of the decimal point. Decimal point is also called the radix point. The procedure of converting the 0.707_{10} into binary is to multiply it by 2 and take the magnitude as the first bit after the radix point. Next, further multiply the new fractional part by 2 and assigning the next position after the first bit. Repeat this until the same pattern appears or as per the bit format chosen.

So if 8-bits for magnitude and 8-bits for representing the fractional part are chosen, then the decimal number 0.707_{10} can be represented as;

0.707_{10} = 00000000. 10110100

This representation is known fixed<8, 8> that means '8' magnitude and 8 bits after the radix. Fixed point values for different values of twiddle factor are displayed in the table 2.

Table2. fixed point representation for twiddle factor

Twiddle factor	**Twiddle factor value**	**Fixed-point representation**
rW_8^0	1	00000010 00000000
iW_8^0	0	00000000 00000000
rW_8^1	0.707	00000000 10110100
iW_8^1	-j0.707	10000000 10110100
rW_8^2	0	00000000 00000000
iW_8^2	-j	10000001 00000000
rW_8^3	-0.707	10000000 10110100
iW_8^3	-j0.707	10000000 10110100

5. DISTRIBUTED ARITHMETIC

Distributed arithmetic is a method that uses lookup table schemes and modulo arithmetic [4]. It is based on a bit level movement process. It replaces the complex multiplications. The distributed arithmetic is can be applied in DSP and image processing applications, frequency analysis and convolution computation. The distributed arithmetic specifically purposes its application on sum of product designs.

In this paper DA was applied on a single butterfly structure [5]. Say for example a single butterfly structure is composed of following elements which can be complex.

Inputs of single BF

$$a = a\_real + a\_img \quad (8)$$

$$b = b\_real + b\_img \quad (9)$$

outputs of single BF

$$A = (A\_real + B\_real) + (A\_img + B\_img) \quad (10)$$

$$B=[(A\_real + B\_real)+(A\_img + B\_img)](W\_real + jW\_img) \quad (11)$$

As twiddle factor is a known value, we can perform the above calculations with fixed point representation Q8.8 and look up tables.

Say

$$(B\_real + B\_img)*W\_real = K1 \quad (12)$$

$$(B\_real + B\_img)*W\_img = K2 \quad (13)$$

Each bit of K1 and K2 is evaluated and value from the Table 3 is used for the respective combinations. These way real and imaginary values are calculated.

Table.3 LookUp Table (LUT) for solving SOP

K1	K2	LUT VALUE
0	0	0
0	1	W_IMG
1	0	W_REAL
1	1	W_IMG-W_REAL

6. INPUT/OUTPUT MATCHING

This FFT processor was run for input sequence,

x(n)={-1,0,2,0,-4,0,2,0}

Output was recorded as,

X(N)={-1,3,-9,3,-1,3,-9,3}

Figure 3 displays the input sequence. Figure 4 shows the output sequence. Figure 2 displays results with Matlab program which can be used for result matching. Table 4 shows the output values in the result.

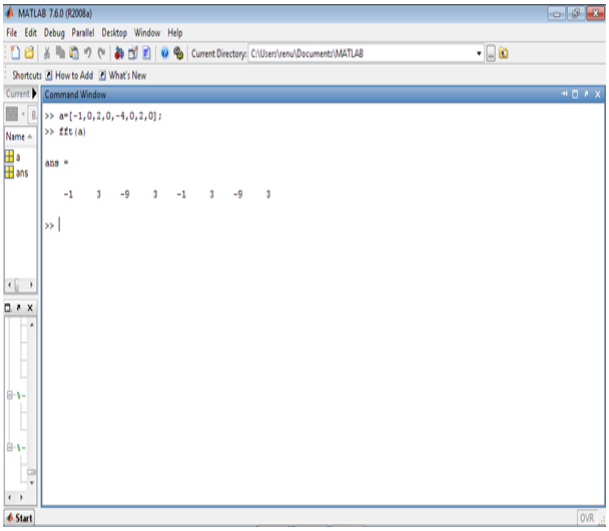

Figure2. Output matching with Matlab software

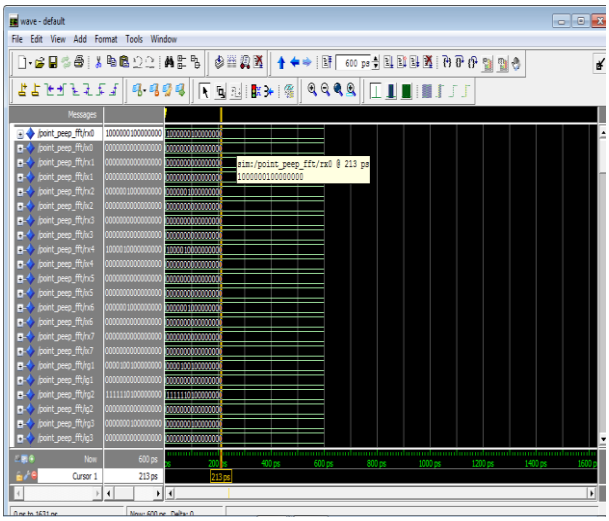

Figure3. Waveforms of input sequence

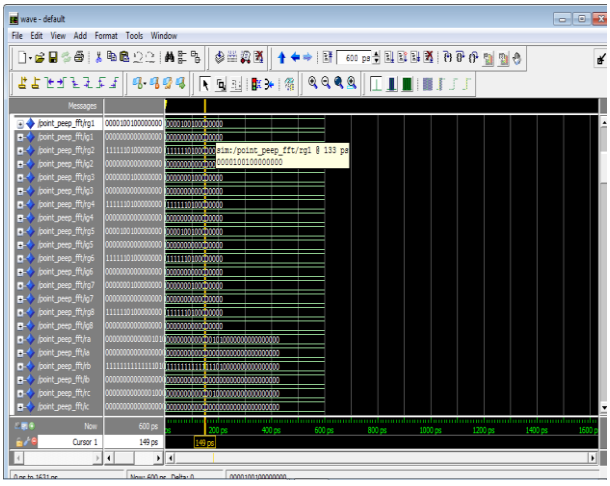

Figure4. Waveform of output sequences

Table4. Resulted output values

Outputs	Binary value	Integer value
Rgo	01111111	-1
Rg1	00000011	3
Rg2	01110111	-9
Rg3	00000011	3
Rg4	01111111	-1
Rg5	00000011	3
Rg6	01110111	-9
Rg7	00000011	3

7. CONCLUSION

In this paper, an easier way is shown to realise Fast Fourier Transformation with distributed arithmetic algorithm. As in plain FFT, six adders and four multipliers are required in a single butterfly; DA algorithm replaces multipliers with adders, thus making the design simple and more efficient. With the use of look up tables (ROM) the overall speed has enhanced. This design uses eight point samples each sixteen bits long; however, it can be used for higher sample rates.

8. REFERENCES

[1] R. A Barapate, "Digital Signal processing," Edition 2010, Tech-Max Publications, Pune.

[2] P. Ramesh Babu, "Digital signal processing," fourth edition, Scitech publications(India).

[3] L. Erick Oberstar, "Fixed-point representation and fractional math," Oberstar consulting, revision August, 30, 2007.

[4] P.Augusta Sophi, R.Srinivasan, J. Raja, "Distributed arithmetic based butterfly element for FFT processor in 45 nm technology," ARPN Journal of Engineering and Applied Sciences, vol.8, no.1, January 2013.

[5] M. Suhasini, K. Prabhu Kumar and P. Srinivas, "Multiplier design and performance estimation with distributed arithmetic algorithm," IJCCT, Vol. 3, Issue-4, 2012.

[6] S.A. White, "A simple FFT butterfly arithmetic unit", IEEE Trans. Circuits system, vol.28 no.4, pp.352-355, April 1981.

[7] S. A. White " Applications of distributed arithmetic to digital signal processing," Analog and Digital Signal processing, 1989, Vol. 12, no. 23, pp. 4-19.

[8] J. Cooley and J. Tuckey, "An algorithm for machine calculation of the complex Fourier series," Math. Comput.,vol.19, 297-301, April. 1965.

[9] T. Thong and B. Liu, "Fixed-point fast fourier transform error analysis, " IEEE Trans. Acoust., Speech, Signal Processing, vol.24, no.6, pp. 563-573, dec.1976.

[10] A.Wenzler and E.Luder, "New structure for complex multipliers and their noise analysis," in proceeding. IEEE ISCAS, Seatle, WA, April.30-May3 1995, vol.2, pp.1432-1435.

[11] J. Takala and K. Punkka, "Butterfly unit supporting radix-4 and radix-2 FFT," Tampere University of Technology, P . O . Box 553, FIN-33101 Tampere, FINLAND.

[12] S.G Smith and P.B Denyer, "Efficient bit-serial complex multiplication and sum of product computation using distributed arithmetic," in proc. IEEE int. conf. Acoustics, Speech and Signal processing, 1986, pp. 271-276.

[13] A. Berkman, V. Owall, and Mats Torkelson, "A low logic depth complex multiplier using distributed arithmetic," IEEE Journal of Solid State Circuits, vol. 35, no.4, April 2000.

[14] T. Ranjith Kumar, "design and implementation of DDA architecture for FIR Filters," IJETT vol. 4, Issue 9, September 2013.