International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 02, February-2015

FFT Implementation on FPGA using Butterfly Algorithm

Enis ÇERRI[1]
Aleksander Moisiu University
Faculty of Information Technology
Durres, Albania

Marsida IBRO[2]
Aleksander Moisiu University
Faculty of Information Technology
Durres, Albania

***Abstract* — Today digital signals processing operation requires several multiplication and for the same we need very fast multiplier for a wide range of requirements for hardware and speed. This paper presents a FFT implementation using FPGA for fast and area efficient digital multiplier based on Butterfly algorithm. FFT is an efficient tool in signal processing in the linear system analysis. Complex arithmetic modules like multiplier and powering units are now being extensively are used in design. The parallel pipelined technology is introduced to increase the throughput of the circuit at low frequency. Based on low power technology FFT power saving is achieved. For the purpose of this implementation, we have used the Altera EPF8282ALC84-4 element and SN-DSP54B device platform FPGA.**

Keywords* — *DSP, FFT algorithm, Butterfly algorithm, FPGA

INTRODUCTION

In the discrete Fourier transform (DFT), both the input and the output consist of sequences of numbers defined at uniformly spaced point in time and frequency, respectively. Accordingly, the DFT lends itself directly to numerical evaluation on digital computers. Moreover, the computation can be implemented most efficiently using a class of algorithms, called Fast Fourier Transform (FFT) algorithms. The FFT refers to a class of efficient algorithms for computing the DFT. The algorithms are efficient in that they use a greatly reduced number of arithmetic operations as compared with the brute force computation of the DFT.
Fourier transform transforms a time-domain function into the frequency domain. Inversely, the Inverse Fourier Transform converse a frequency-domain function into the time domain. For an aperiodic continuous signal, the continuous Fourier Transform is expressed by:

$$X(\Omega) = \int_{-\infty}^{\infty} x_c(t) e^{-j\Omega t} dt \qquad (1)$$

and the continuous inverse Fourier Transform is:

$$x_c(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_c(\Omega) e^{-j\Omega t} d\Omega \qquad (2)$$

Equations (1) and (2) are known as the continuous Fourier Transform pair for aperiodic signals which related the time and frequency domains. Similarly form an aperiodic digital sequence, the discrete-time Fourier Transform (DTFT) is given by:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n} \qquad (3)$$

and the discrete-time inverse Fourier Transform is:

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega \qquad (4)$$

Equations (3) and (4) are known as the discrete-time Fourier Transform (DTFT) pair for aperiodic digital sequences which related the time and frequency domains. Although, equation (3) gives the frequency spectrum of a signal, there are two implementation problems in practice. The first problem is associated with the limits of summation which extend from $-\infty$ to ∞, implying the length of the signal must be infinitely long. The second problem is associated with the frequency variable ω which is continuous implying there is an infinite number of frequency points to be computed [1, 9].
To overcome the first problem, the limit of the summation are reduced, thereby truncating an infinitely long signal to a finite length signal. This is known as windowing because only a portion of the actual discrete signal is available for the transform operation. To overcome the second problem, the number of frequency points to be computed is confined to a finite number and the selected frequency points should be spaced evenly over the range 0 to f_s. Taking these factors into account, equation (3) can be expressed in a new form to give the discrete Fourier Transform (DFT), that is:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N{}^{nk} \qquad (5)$$

where W_N is called the twiddle factor and $W_N = e^{-j2\pi/N}$.

The twiddle has two important properties: symmetry and periodicity.
The smallest transform used in 2-point DFT which is known as radix-2. It processes a group of two samples. Radix-2 is the fastest method for calculating FFT. These are amongst the one of large number of FFT algorithm being developed. Radix-2 algorithm are useful if N is a regular power of 2(N=2^p). The term FFT is actually slightly ambiguous

because there are several commonly used FFT algorithms. There are two different radix-2 algorithms, they are: decimation-in-time (DIT) and decimation-in-frequency (DIF) algorithms. A butterfly unit block consisting of N/2 butterflies. Each one containing two (N/2)*16-bits ROMs to store the sine and cosine of the twiddle factors, four 16*16 multipliers in 2's complement, six 32-bits accumulators and two special operators to adequate the data format [2, 4].

1. *DIT FFT Algorithm*

The decimation-in-time FFT (DIT FFT) is a process of dividing the N-point DFT into two (N/2)-point DFTs by splitting the input samples into even and odd indexed samples. The two (N/2)-point DFTs are then further divided in the same way into (N/2)-point DFTs and this decomposition process continues until 2-point DFTs are obtained [5]. This approach is demonstrated below. Since $W_N^2 = [e^{-j(2\pi/N)}]^2 = [e^{-j2\pi(N/2)}]^2 = W_{N/2}$, the following equation can be derived by rearranging equation (5) and then:

$$X(k) = \sum_{n=0}^{N/2-1} x(2n) W_N^{2nk} + \sum_{n=0}^{N/2-1} x(2n+1) W_N^{(2n+1)k}$$
$$= \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{nk} + \sum_{n=0}^{N/2-1} x(2n+1) W_{N/2}^{nk} W_N^k$$
$$= \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x(2n+1) W_{N/2}^{nk}$$
$$= Y(k) + W_N^k Z(k) \quad (6)$$

where $k = 0,1, \ldots N-1$, $Y(k)$ and $Z(k)$ are the DFTs of the even- and odd- indexed samples respectively. Although, equation (6) still needs to be valuated N times for k varying from 0 to N-1, each summation only needs to be computed N/2 times for k varying from 0 to $(N/2)-1$ because $Y(k)$ and $Z(k)$ are repetitive with an internal interval of N/2 (periodicity property). Consequently, the original DFT computation time is reduced by approximately 50%.

Consequently by restricting k to the range 0 to $(N/2)-1$, equation (6) can be rewritten in two parts; one part for the first half of the frequency points $(0 \le k \le N/2-1)$ and other part for the second half of the frequency points $(N/2 \le k \le N-1)$, that is $X(k) = Y(k) + W_N^k Z(k)$.

$$X(k+N/2) = \sum_{n=0}^{N/2-1} x(2n) W_{\frac{N}{2}}^{n\left(k+\frac{N}{2}\right)}$$
$$+ \sum_{n=0}^{N/2-1} x(2n+1) W_{N/2}^{n(k+N/2)} W_N^{(k+N/2)}$$
$$= \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{nk} + W_N^{(k+N/2)} \sum_{n=0}^{N/2-1} x(2n+1) W_{N/2}^{nk}$$
$$= Y(k) - W_N^k Z(k) \quad (7)$$

The implementation of equation (7) for an 8-point DFT can be shown as a butterfly diagram as in Figure 1. Each butterfly takes a pair of inputs and generates a pair of outputs.

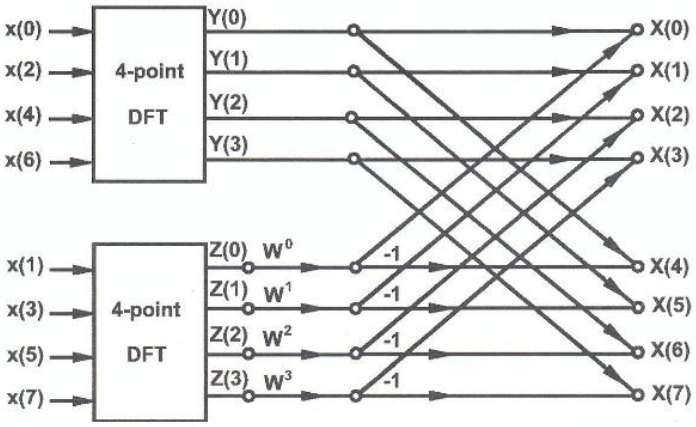

Figure 1. Butterfly diagram for 8-point DFT with one decimation stage

Applying the same decomposition technique again to divide two N/2-points DFTs into four N/4-point DFTs by splitting the even- and odd-indexed sequences into four subsequences, see the stage 2 in Figure 2. Consequently, the same technique is applied to divide four N/4-point DFT into eight N/8-point DFTs, see stage 3 in Figure 2.

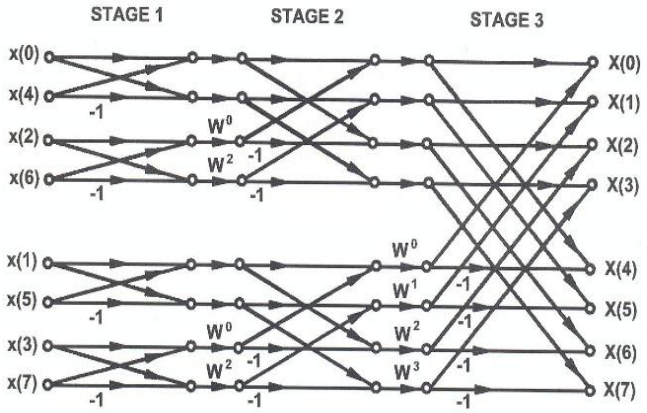

Figure 2. Butterfly diagram for a 8-point DIT FFT

Each decomposition stage doubles the number of separate DFTs, but halves the number of points in DFT. In computing an N-point DFT, this decimation process can be repeated $log_2 N$ times. The number of computation stages is seen to be 3 since $2^3 = 8$.

In the stage 1 are required N additions/substractions and the other two stages require N complex additions/substractions (or 2N additions/substractions) and N/2 complex multiplications (or 2N multiplications and N additions/substractions, since each complex multiplication requires 4 ordinary multiplications and 2 additions/substractions). Since there are $log_2 N$ stages for an N-point FFT, the total number of multiplications is $2N(log_2 N - 1)$ and the total number of additions is $3N(log_2 N - 1) + N$. This can be compared with the $2N^2$ multiplications and $2(N-1)N$ additions required for the direct implementation of DFT.

3. *DIF FFT Algorithm*

In contrast to the DIT FFT which decomposes the DFT by recursively splitting the input samples in the time domain into subsequences, the decimation-in-frequency FFT (DIF FFT) decomposes the DFT by recursively splitting the sequence elements in the frequency domain into smaller subsequences [5]. Dividing equation (5) into two N/2-point DFTs by splitting the input samples into halves yields:

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

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

$$= \sum_{n=0}^{N/2-1} x(n) W_N^{nk} + W_N^k \sum_{n=0}^{N/2-1} e^{-j\pi k} [x(n) + x(n+N/2)] W_N^{nk} \quad (8)$$

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

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

$$= \sum_{n=0}^{N/2-1} [x(n) + e^{-j\pi k} x(n+N/2))] W_{\frac{N}{2}}^{n(k)}$$

$$+ \sum_{n=0}^{N/2-1} [x(n) + e^{-j\pi k} x(n+N/2)) W_N^n] W_{N/2}^{n(k)}$$

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

$$\sum_{n=0}^{N/2-1} [x(n) - x(n+N/2) W_N^n] W_{N/2}^{n(k)} \quad (9)$$

The implementation of equation (9) for a 8-point DFT is shown as butterfly diagram in Figure 3.

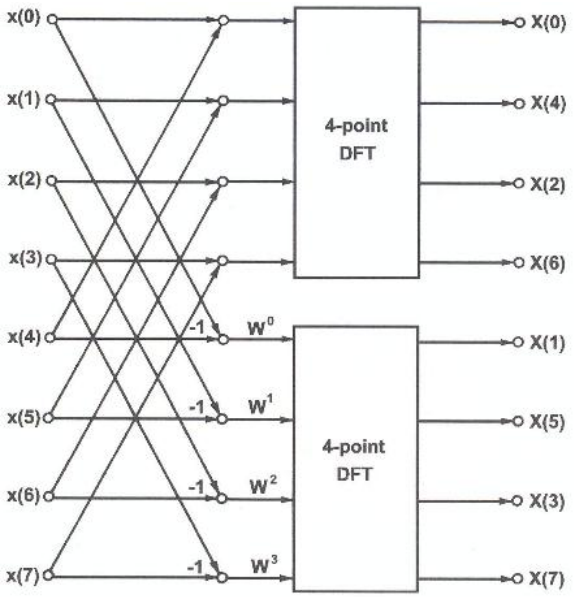

Figure 3. Butterfly diagram for 8-point DFT with one decimation stage

In contrast to Figure 2, Figure 4 shows that DIF FFT has its input data sequence in natural order and the output sequence in bit-reversed order. For a 512-point FFT, 512-points cosine and sine tables should be built to involve this computation.

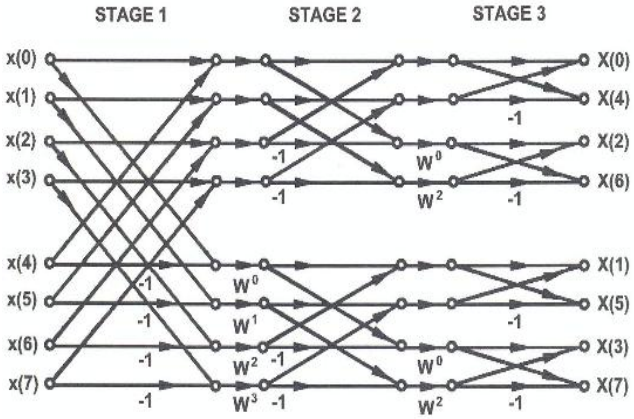

Figure 4. Butterfly diagram for 8-point DIF FFT

4. *Implementation*

To implement the computation of butterfly with C54x instructions we have considered equations (8) and (9). If letting $Y(k) = P$, $X(k) = P`$ and $X(k+N/2) = Q`$, these two equations can be rewritten as:

$$P` = P + W_N^K Q$$

$$Q` = P - W_N^K Q$$

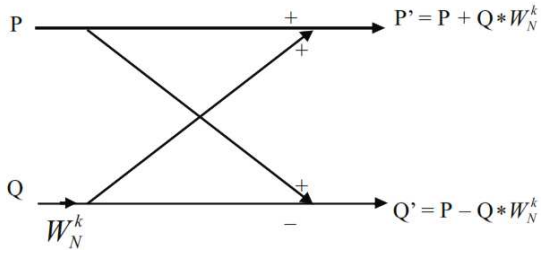

Figure 5. DIT Radix 2 Butterfly

Since sequences P and Q and twiddle coefficient W are complex, we can divide these butterfly computations into two parts (real and imaginary). In practical computation, data of each part is stored in memory buffer. This is called in-place operation.

At stage 1, there are only two values of twiddle factor (k=0 and k=n/2) required for DFT computations. If k=0, then $W_{0/n} = 1$ since $W_n = \cos(2\pi \cdot k/n) - \sin(2\pi \cdot k/n)$. If $k = n/2$, then $W_{1/2} = -1$.

At stage 2, there are four twiddle required for the butterfly computation $W_{0/n}$, $W_{1/2}$, $W_{1/n}$ and $W_{1/2}$. Complex computations are required for the butterfly computations after stage 3. For a 1024-point DIT FFT 10 stages are needed $(2^{10} = 1024)$ to achieve the butterfly computations.

In DSP system, imaginary part must be placed by zero for FFT computations. This process is called packing. The unpacking process is the reverse operation of packing. It is used to unpack the FFT outputs to N-point complex values because the FFT outputs are N/2-point complex values. After unpacking, the 1024 complex values are used to compute the power values of FFT outputs [7]. Each power values is computed by the formula $P = |R^2 + I^2|$, where R is the real part and I is the imaginary part [6, 8].

In this implementation we have used a deterministic 3 kHz signal with low-level white noise created by MATLAB. Once the FFT processor is executed, the deterministic data will be loaded and stored in input data buffer 1C00H-1FFFH.

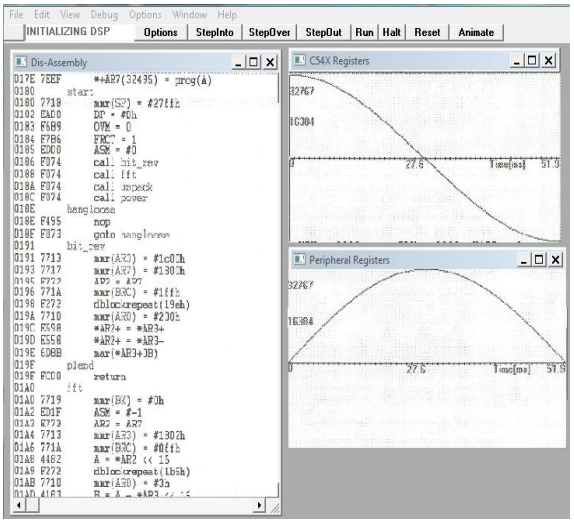

Figure 6. Sine and cosine waves

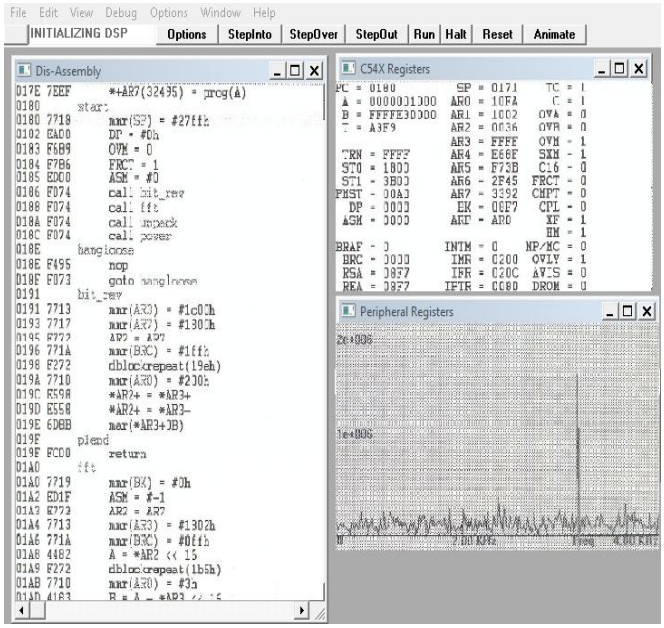

Figure 7. Spectrum of input signal

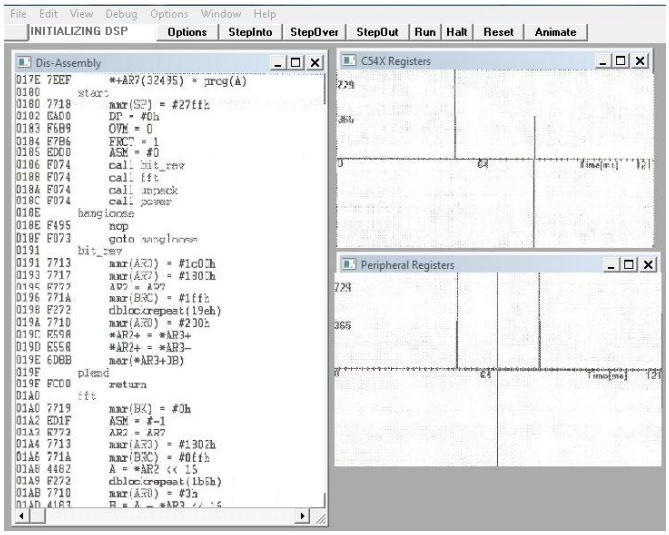

Figure 8. Power of FFT signal spectrum

5. CONCLUSIONS

In this paper we have described the basic algorithm for radix-2 FFTs. Modeling and hardware description of FFT approaches such as Butterfly algorithms by VHDL were introduced and the realization of them on Altera EPF8282ALC84-4 chip was proposed. The power impact parameter of the technique has been observed and compared with the conventional FFT blocks to analyze the performance. This paper presented a new, very high speed FFT architecture based on the radix-2 butterfly algorithm. A fully pipelined, processing core of a 1024-point FFT has been implemented in FPGA. The results shows a very high operating frequencies and low latencies of the implementation.

REFERENCES

[1] Arman Chahardahcherik, Yousef S. Kavian, Otto Strobel, and Ridha (2011). *Implementing FFT Algorithms on FPGA* (IJCSNS). International Journal of Computer Science and Network Security, Vol. 11, No.11, November 2011.

[2] Aniket Shukla, Mayuresh Deshmukh (2012), *Comparative Study of Various FFT Algorithm Implementation on FPGA,* International Journal of Emerging Trends in Signal Processing Vol.1, Issue 1, November 2012.

[3] Chandan.M,S.L.Pinjare, Chandra Mohan Umapthy Chandan M, S.L Pinjare (2012). *Optimized FFT Design using Constant Coefficient Multiplier.*International Journal of Emerging Technologyand Advanced Engineering, Vol.2, Issue 6, June 2012.

[4] Niladri Mandal, Souragni Ghosh (2012). *A Modified Fast FFT Algorithm for OFDM.* International Journal of Soft Computing and Engineering (IJSCE), Vol.1, Issue-6, January 2012.

[5] K.Sreekanth Yadav, V.Charishma, , Neelima Koppala (2013).*Design and simulation of 64 point FFT using Radix 4 algorithm for FPGA Implementation,* International Journal of Engineering Trends and Technology- Volume 4 Issue2- 2013

[6] M. Kannan and S.K. Srivatsa (2007). *Low Power Hardware Implementation of High Speed FFT Core.* Journal of Computer Science. Vol. 3, Issue 6, 2007.

[7] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline fft processor," in Proc. IEEE Custom Integrated Circuits Conf., Santa Clara, CA, May 11-14 1998, vol. 2, pp. 131–134

[8] Senthil Sivakumar M & Banupriya M & Arockia Jayadhas S (2012). *Design of Low Power High Performance 16-Point 2 Parallel Pipelined FFT Architecture.* International Journal of Electronics,`Communication& Instrumentation Engineering Research and Development (IJECIERD). Vol. 2, Issue 3 September 2012.

[9] Sneha N.Kherde, Meghana Hasamnis (2011). *Efficient Design and Implementation of FFT.* International Journal of Engineering Science & Technology, Vol. 3 Issue Sup, p10, February 2011.