IMPACT: International Journal of Research in Engineering & Technology (IMPACT: IJRET)
ISSN(E): 2321-8843; ISSN(P): 2347-4599
Vol. 2, Issue 4, Apr 2014, 199-206

DESIGN OF PARALLEL PIPELINED FEED FORWARD ARCHITECTURE FOR ZERO FREQUENCY & MINIMUM COMPUTATION (ZMC) ALGORITHM OF FFT

K. JAYARAM[1], C. ARUN[2] & B. SAKTHIVEL[3]

[1,3]Assistant Professor, Department of ECE, Madurai Institute of Engineering & Technology, Tamil Nadu, India

[2]Professor, Department of ECE, R.M.K College of Engineering & Technology, Tamil Nadu, India

ABSTRACT

A new parallel pipelined feed forward architecture for real-time signal is proposed. A hardware oriented radix-2 algorithm is derived by integrating a twiddle factor decomposition technique in the divide and conquer approach. The butterfly structure of radix-2 algorithm is modified in accordance with the flow of signal. The new butterfly structures are designed to handle the hybrid data path which consists of real & complex data paths. The proposed approach which can be extended to all radix-2^k based DITFFT & DIFFFT algorithms. The zero frequency is computed without processing the zero input data. The symmetry property is applicable to minimize the stage computation in half of the actual stage. The proposed radix-2^k feed forward architectures need to use fewer hardware resources in hardware architecture. The proposed radix 2^k architectures lead to low hardware complexity with respect to adders and delays. The N-point 4-parallel radix-2^2 architecture requires $4(\log_{16}N-1)$ complex multipliers, $\log_2N$ real adders and N-1 complex delay elements.

KEYWORDS: DITFFT & DIFFFT Algorithms, Fast Fourier Transform (FFT)

INTRODUCTION

The Fast Fourier Transform (FFT) is an essential algorithm in digital signal processing. It is employed in various applications such as radar, wireless communication, medical imaging, spectral analysis, and acoustics. Fast Fourier Transforms have been implemented on different platforms, ranging from general purpose processors to specially designed computer chips. There are two main types of pipelined architectures: feedback and feed forward. On the one hand, feedback architectures are characterized by their feedback loops, i.e., some outputs of the butterflies are fed back to the memories at the same stage.

Feedback architectures can be divided into two main types of pipelined architectures: feedback and feed forward. On the one hand, feedback architectures are characterized by their feedback loops, i.e., some outputs of the butterflies are fed back to the memories at the same stage. Feedback architectures can be divided into single-path delay feedback which process a continuous flow of one sample per clock cycle, and multi-path delay feedback or parallel feedback which process several samples in parallel. There has been an increasing interest in the computation of FFT for real valued signals, since virtually most of the physical signals are real. The real valued signals which are of prime importance in real-time signal processing exhibit conjugate symmetry giving rise to redundancies. This property could be exploited to reduce both arithmetic and memory complexities. The RFFT plays an important role in different fields such as communication systems, biomedical applications, sensors and radar signal processing.

A low complexity implementation of RFFT can reduce the power consumption in implantable or portable devices. Even though specific algorithms for the computation of the RFFT have been proposed in the past, these algorithms lack regular geometries to design pipelined architectures. A novel approach to design efficient pipelined architectures for RFFT was presented for the first time in. This architecture is obtained by modifying the radix-2 flow graph to achieve real data paths and processes 4 samples in parallel.

This approach is specific to radix-2 algorithm and was limited to a 4-parallelarchitecture. A general approach which can be extended to other radix-2^n algorithms is needed to take advantage of less number of multiplication operations in higher radix algorithms.

RADIX-2^2 DIF FFT ALGORITHM

By the observations made in last section the most desirable hardware oriented algorithm will be that it has the same number of non-trivial multiplications at the same positions in the SFG as of radix-4 algorithms, but has the same butterfly structure as that of radix-2 algorithms. The clear derivation of the algorithm in DIF form with perception of reducing the hardware requirement in the context pipeline FFT processor is, however, yet to be developed.

The DFT of size N is defined by

$$X_p = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi}{N}np}, \quad p \in \{0,1,\ldots,N-1\} \tag{1}$$

$W = e^{-j\frac{2\pi}{N}}$ then

$$X_p = \sum_{n=0}^{N-1} x_n * W^{np}, \quad p \in \{0,1,\ldots,N-1\} \tag{2}$$

Where W_N denotes the N th primitive root of unity, with its exponent evaluated modulo N. To make the derivation of the new algorithm clearer, consider the first 2 steps of decomposition in the radix-2 DIF FFT together.

$$X_p = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-j\frac{2\pi}{N}(2n)p} + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-j\frac{2\pi}{N}(2n+1)p} \tag{3}$$

The key idea of the new algorithm is to proceed the second step decomposition to the remaining DFT coefficients, including the "twiddle factor" $W_{kn}^{\left(\frac{N}{4}n_2+n_3\right)(k_1)}$ to exploit the exceptional values in multiplication before the next butterfly is constructed. Decomposing the composite twiddle factor and observe that

$$A_p = \sum_{n=0}^{\frac{N}{4}-1} x_{2n} e^{-j\left[\frac{2\pi}{N/2}\right]np}, \quad B_p = \sum_{n=0}^{\frac{N}{4}-1} x_{2n+1} e^{-j\left[\frac{2\pi}{N/2}\right]np} \tag{4}$$

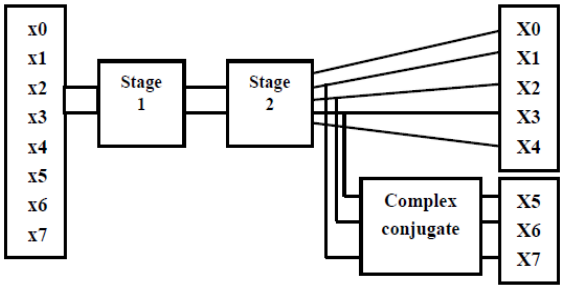

Figure 1: Different Stages of Computation

Radix-2^2 algorithm has the feature that it has the same multiplicative complexity as radix-4 algorithms, but still retains the radix-2 butterfly structures. The multiplicative operations are in a such an arrangement that only every other stage has non-trivial multiplications. This is a great structural advantage over other algorithms when pipeline/cascade FFT architecture is under consideration.

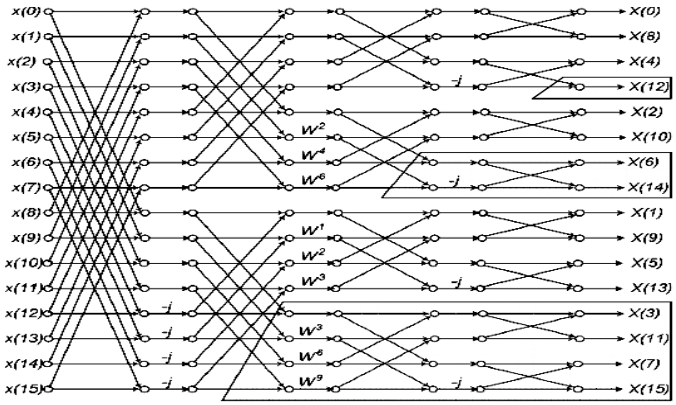

Figure 2: Feedforward Structure Radix-2^2 DIF FFT

In the first step, the FFT flow graph is modified by removing the redundant samples. Most of the approaches in literature compute the frequencies with indexes k=[0, N/2]. Figure 2 shows the flow graph of 16-point FFT decimated in frequency. The boxed regions show the redundant samples which can be removed from the flow graph.

These samples and the corresponding computations can be removed from the flow graph. The flow graph after removing the redundant samples will be irregular. This flow graph is suitable to implement efficient in-place architectures, but not for efficient pipelined architectures.

Figure 3 shows the modified flow graph which is obtained by rearranging the imaginary operations in place of redundant operations. All the edges in Figure 3 are real, i.e., the data have been separated into real and imaginary parts.

The continuous edges compute the real parts and the broken edges represent the computations of the imaginary parts. The output samples of the flow graph include a letter (r or i) to indicate if the value corresponds to the real part or the imaginary part of the output.

Further, in higher point FFTs, a complex sample (real and imaginary components computed separately) may need to be multiplied by a complex twiddle factor, i.e., a full complex multiplier is required.

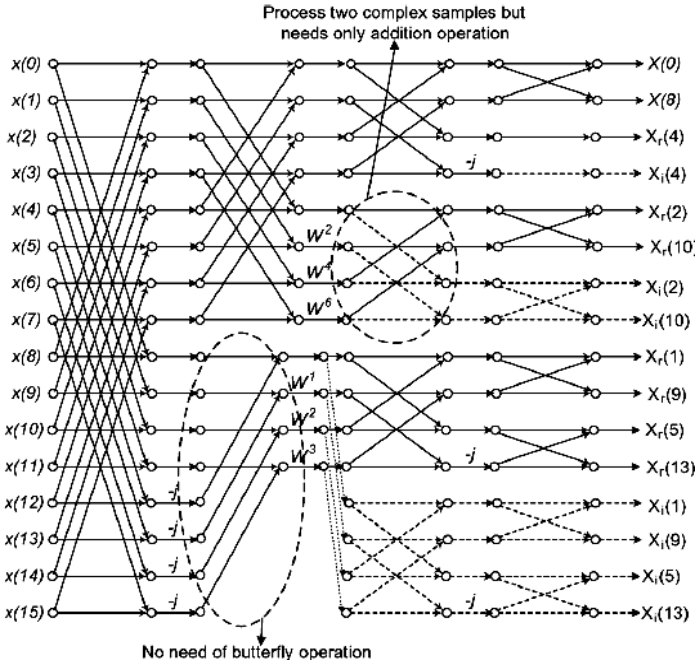

Figure 3: Adapted Flow Graph of a Radix-2^2 DIF FFT

In that situation, the proposed modifications will not lead to a regular flow graph. This problem can be solved by reordering the data before the multiplier to have corresponding samples at the input of the multiplier at the same time. At the architecture level, this problem can be solved by introducing an extra stage of reordering circuit before the multiplier. We modify the multiplier stage similar to that of the butterfly stage by adding a reordering circuit. This will increase the latency by a few cycles and the number of delay elements depending on the stage at which multipliers are required.

PROPOSED ARCHITECTURES

In this section, we present 2-parallel and 4-parallel architectures for real FFT computation based on radix-2^3 and radix-2^4 algorithms using the proposed methodology.

2-Parallel Architecture

Figure 4 shows a general N-point 2-parallel architecture based on the radix-2^3 algorithm, where N is a power of 8. The bottom part will be repeated $\log_8$ N-2 times. In general, for an N-point RFFT, with N power of 2, the 2-parallel architecture requires $2\log_2$ N real adders, $\log_8$ N-1 complex multipliers, 2 ($\log_8$ N-1) CSD(W^8_{64}) multipliers and 3N/2-2 real delay elements. Few extra delays are required for interchanging real and imaginary components before twiddle factor multiplications.

4-Parallel Architecture

Figure 5 shows a general N-point 4-parallel architecture based on radix-2^3 algorithm, where N is a power of 8. The bottom part will be repeated times. In a general case of N-point RFFT, the 4-parallel architecture requires $4\log_2$ N real adders, 2 ($\log_8$ N-1) complex multipliers, 4 $\log_8$ N-5 CSD W^8 multipliers and 7N/4-4 delay elements.

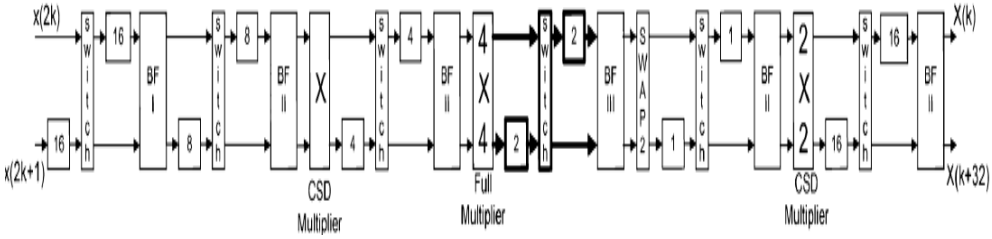

Figure 4: Proposed 2-Parallel Architecture

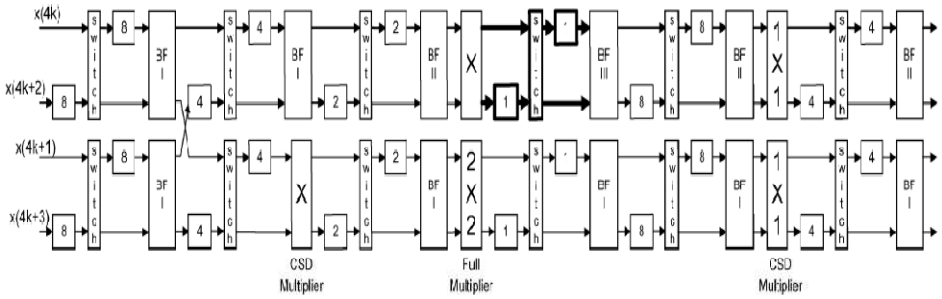

Figure 5: Proposed 4-Parallel Architecture

SYNTHESIS RESULTS

The architectures depends on the required number of multipliers, adders, delay elements. The performance is represented by throughput. The proposed designs are the only specific approaches for the computation of RFFT. The symmetric property in proposed method is used to reduce the hardware requirement.

The proposed architectures are feed-forward. The pipelining stages are added as necessary to increase the frequency of operation. Much parallel pipelined architecture to compute FFT with complex inputs have been proposed in the literature based on radix- 2^n algorithms.

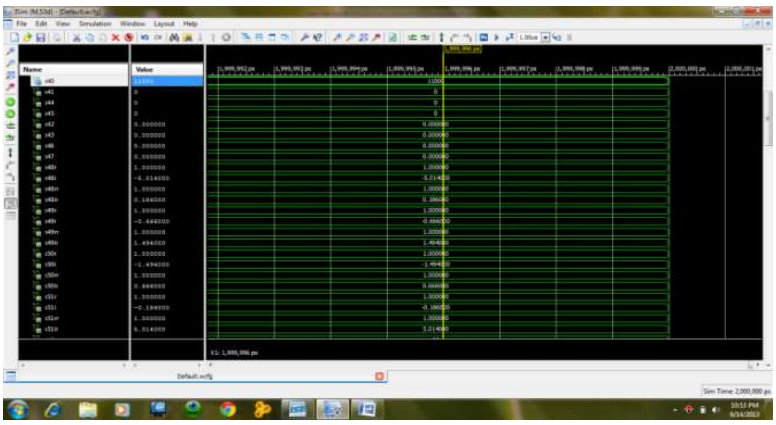

Figure 6

Table 1

Algorithm	Radix-2^2
FFT Size	16
Frequency	200MHZ
Area	2132µm^2
Power	5.2mW

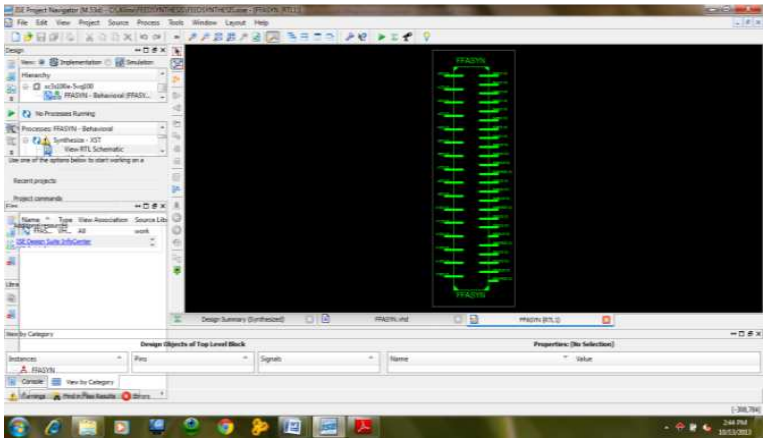

Figure 7

COMPARISON OF FFT

The comparison of Fast Fourier Transform & Real Value Signal Fast Fourier Transform can be done for knowing the computation time. And also we can improvise the concept by changing some parameters.

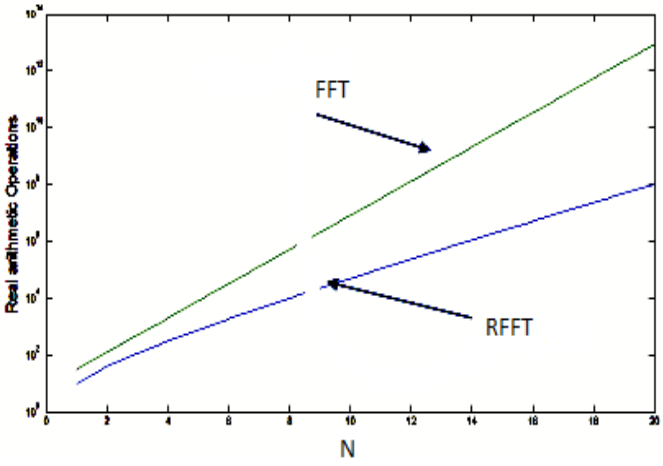

Figure 8

CONCLUSIONS

This parallel pipelined feed forward architecture paper has presented to design efficient architectures for the computation of RFFT. The computation can be completed by limiting the signal flow graph diagram into real & imaginary paths. This approach is mainly used to effective utilization of memory and to avoid reusability. The approach can be implemented for higher radix algorithms. Based on the modified flow graph and hybrid data path design using radix-2^3 and radix-2^4 algorithms, novel 2-parallel and 4-parallel pipelined architectures are developed. This approach can also be designed for decimation-in-time (DIT) algorithms. The same approach is applicable for complex FFT, the DIT architectures have need of less delay elements than DIF architectures.

REFERENCES

1. 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.

2. *Chandan.M, S.L.Pinjare, Chandra Mohan Umapthy Chandan M, S.L Pinjare (2012). Optimized FFT Design using Constant Co-efficient Multiplier. International Journal of Emerging Technology and Advanced Engineering, Vol. 2, Issue 6, June 2012.*
3. *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.
4. 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.
5. 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.
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. Cooley, James W., and John W. Tukey (1965) *An algorithm for the machine calculation of complex Fourier series, Math. Computer.* Vol. 19, 1965.
8. C. Rader(1965), *Discrete Fourier Transform when the Number of Data Samples is Prime, Proceedings of the IEEE* 56, 1968, Vol. 19, No. 90 (April 1965).
9. Brenner, N.; Rader, C. (1976). *A New Principle for Fast Fourier Transformation*. IEEE Acoustics, Speech & Signal Processing, Vol. 23, Issue 3, Jun 1976.
10. 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.
11. H. Sorensen, D. Jones, M. Heideman, and C. Burrus, "Real-valued fast fourier transform algorithms," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 35, no. 6, pp. 849–863, Jun. 1987.
12. J. Chen and H. Sorensen, "An efficient FFT algorithm for real-symmetric data," in *Proc. IEEE ICASSP*, Mar. 1992, vol. 5, pp. 17–20.
13. B. R. Sekhar and K. M. M. Prabhu, "Radix-2 decimation in frequency algorithm for the computation of the real-valued FFT," *IEEE Trans, Signal Process.*, vol. 47, no. 4, pp. 1181–1184, Apr. 1999.
14. B. M. Baas, "A low-power, high-performance, 1024-point FFT processor," *IEEE J. Solid-State Circuits*, vol. 34, no. 3, pp. 380–387, Mar.1999.
15. M. Garrido, K. K. Parhi, and J. Grajal, "A pipelined FFT architecture for real-valued signals," *IEEE Trans. Circuits Syst. I, Reg Papers*, vol.56, no. 12, pp. 2634–2643, Dec. 2009.
16. M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 6, pp. 1068–1081, Jun. 2012.