International Journal of Computer Applications (0975 – 8887)
Volume 98– No.17, July 2014

An Efficient Multi-Path Delay Commutator Architecture

G. Narmadha
Asst Prof / Department of ECE
SACS MAVMM Engineering College
Madurai, India

S. Deivasigamani
Lecturer/FECT
AIMST University
Kedah Darul Aman, Malaysia

K. Balasubadra, Ph.D
Professor and HOD,
Department of ECE
RMD Engineering College
Chennai, India

ABSTRACT
The Appearance of radix-2^2 was a milestone in the design of pipelined FFT hardware architectures. Later, radix-2^2 was extended to radix-2^K. In the feed forward architectures radix-2^K can be used for any number of parallel samples which is a power of two. Indeed, it is shown that feed forward structures are more efficient than feedback ones when several samples in parallel must be processed. As a results shown that the proposed designs are efficient both in area and perform ace, being possible to obtain throughputs of the order of GSamples/s as well as very low latencies.

Keywords
Fast Fourier Transform (FFT), Radix-2^k, Multipath Delay Commutator (MDC), Pipelined Architecture, Very Large-Scale Integration (VLSI).

1. INTRODUCTION
The Fast Fourier Transforms (FFT) and its inverse (IFFT) is one of the fundamental operations in the field of digital signal processing. A Fast Fourier Transform (FFT) is an algorithm to compute the Discrete Fourier Transform (DFT) and its inverse. A Fourier Transforms converts time to frequency and vice versa; An FFT rapidly computes such transformations. As a result, FFT are widely used for many applications in engineering, science, and mathematics.

FFT architectures can be divided into three main classes: (1) Pipeline FFT utilizes concurrent processing of different stages to achieve high throughput. (2) Column FFT, each stage in the FFT is computed with a set of processing elements and the result is fed back to the same processing elements for the computation of the next stage and (3) Fully parallel FFT implementations are hardware intensive and are with current technology not practical for large FFT architectures. In this context, pipelined FFT hardware architecture is widely used, because they provide high throughputs and low latencies suitable for modern application.

There are two main types of pipelined architecture: feedback (FB) and feed forward (FF). 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 types. (1) Single-path Delay Feedback (SDF) which process a continuous flow of one sample per clock cycle, and (2) Multi-path Delay Feedback (MDF) or parallel feedback which process several samples in parallel. On the other hand, feed forward architectures also known as Multi-path Delay Commutator (MDC) do not have feedback loops and each stage passes the processed data to the next stage. These architectures can also process several samples in parallel.

Recently, the FFT/IFFT is used as one of the key component in OFDM-based wideband communication systems, like wireless mobile terminals. In this context two main challenges can be distinguished. The first one is to calculate the FFT of multiple independent data sequences because they provide all the FFT processors can share the memory in order to reduce the hardware implementations. The second challenge is to calculate the FFT when several samples of the same sequence are received in parallel. Because it is necessary to restore to FFT architectures that can manage several samples in parallel. However, radix-2^k had not been considered for feed forward architectures until the first radix-2^2 feed forward FFT architectures were introduced [1-4].

In this paper, the radix-2^k feed forward pipeline FFT hardware architectures are proposed designs include radix-2^2, radix-2^3 and radix-2^4 architectures. It shows that radix-2^k can be used for any number of parallel samples which is a power of two. Accordingly, radix-2^k FFT architectures for 2,4 and 8 parallel samples are presented. These architectures are shown to be more hardware-efficient than previous feed forward, folding and parallel feedback architectures designs.

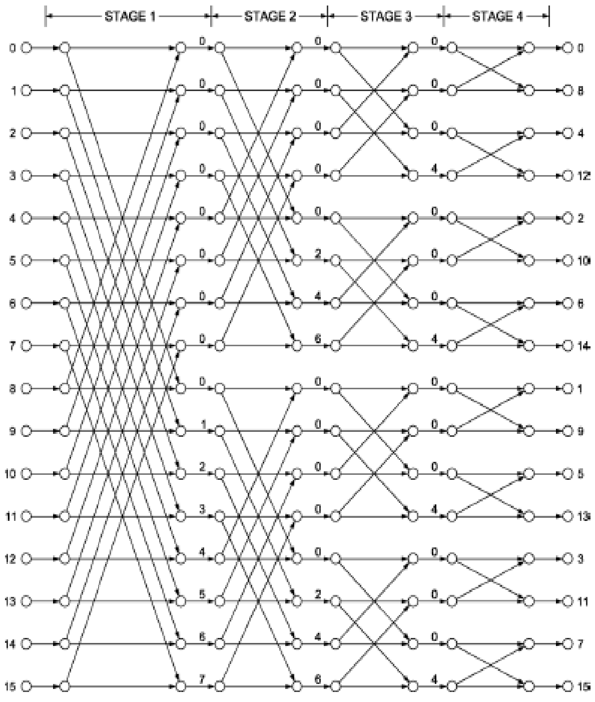

Fig 1: Flow graph of the 16-point DIF FFT

The paper is organized as follows. In Section 2 explains the radix-2^2 FFT algorithm. In section 3, the experimental results are provided and in section 4 performance analyses. Finally, the main contributions of this work are summarized in section 5.

2. THE RADIX-2^2 FFT ALGORITHM

The best-known FFT algorithms depend upon the factorization of N, but there are FFTs with O (N log N) complexity for all N, even for prime N. Many FFT algorithms only depend on the fact that $e^{\frac{-2\pi i}{N}}$ is an N-th primitive root of unity, and thus can be applied to analogous transforms over any finite field, such as number-theoretic transforms [5]. Since the inverse DFT is the same as the DFT, but with the opposite sign in the exponent and a 1/N factor, any FFT algorithm can easily be adapted.

An FFT computes the DFT and produces exactly the same result as evaluating the DFT definition directly; the only difference is that an FFT is much faster.

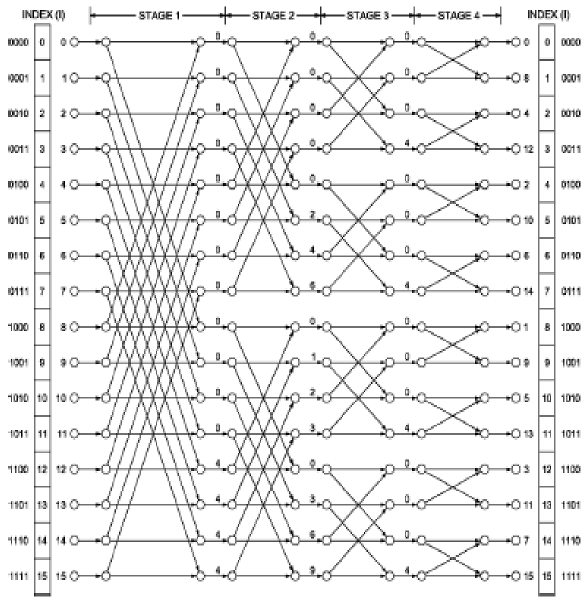

Fig 2: Flow graph of the 16-point radix-2^2 DIF FFT

Let $x_0 \ldots x_{N-1}$ be complex numbers. The DFT is defined by the formula,

$$X_K = \sum_{n=0}^{N-1} x_n e^{-i2\pi k\frac{n}{N}} \quad K=0, 1\ldots\ldots N-1$$

To illustrate the savings of an FFT, consider the count of complex multiplications and additions [6]. Evaluating the DFT's sums directly involves N^2 complex multiplications and N (N-1) complex additions. The well-known radix-2 Cooley-Tukey algorithm, for N a power of 2, can compute the same result with only (N/2) $\log_2$ (N) complex multiplications and $N\log_2$ (N) complex additions. In practice, actual performance on modern computers in usually dominated by factors other than the speed of arithmetic operations. But the overall improvement from O (N^2) to O (N log N) remains [7].

In many applications, the input data for the DFT are purely real, in which case the outputs satisfy the symmetry

$$X_{N-k} = X_k^*$$

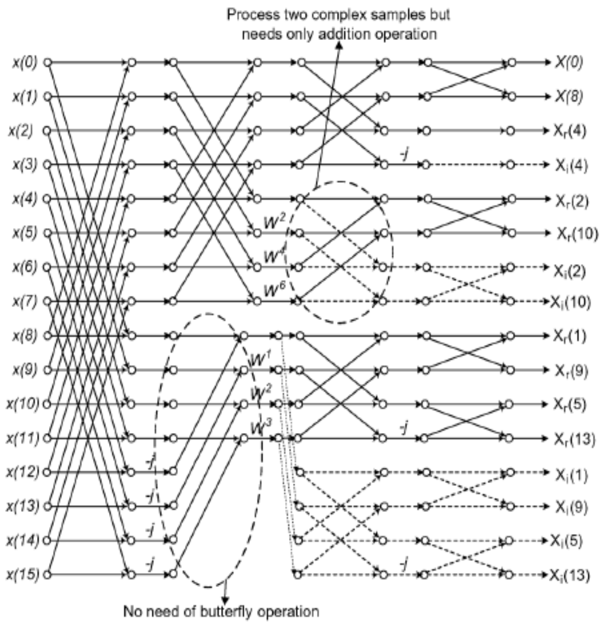
Process two complex samples but needs only addition operation
No need of butterfly operation

Fig 3: Flow graph of the 16-point radix-2^2 DIF FFT using redundant operation

and efficient FFT algorithms have been designed for this situation. One approach consists of taking an ordinary algorithm (e.g. Cooley-Tukey) and removing the redundant parts of the computation, saving roughly a factor of two in time and memory.

3. EXPERIMENTAL RESULTS

The radix-2^k feed forward FFT architectures has been simulated and synthesized by using Xilinx ISE12.1i and the design has been implemented by using spartan3 FPGA processor. Comparisons of the existing and proposed method are given in the Table 1.

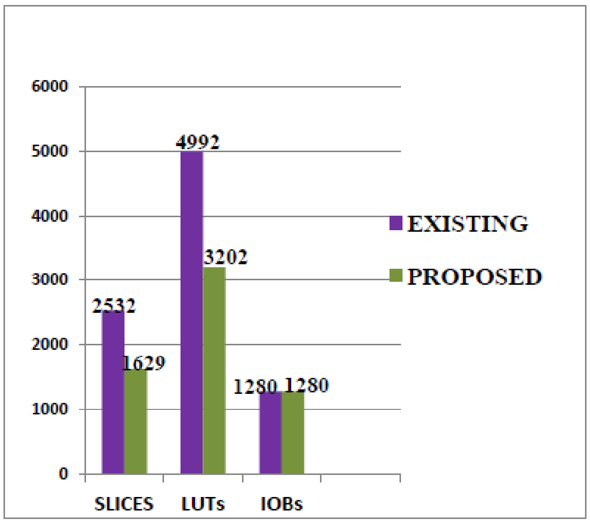

Fig 4: Performance analysis of existing and proposed method

Table 1. Comparison of Slices, LUTs, IOBs and Time

S.NO	PARAMETER	EXISTING METHOD	PROPOSED METHOD
1.	Number of Slices	2532	1629
2.	Number of 4 input LUTs	4992	3202
3.	Number of Bonded IOBs	1280	1280
4.	Time(ns)	28.162	12.198

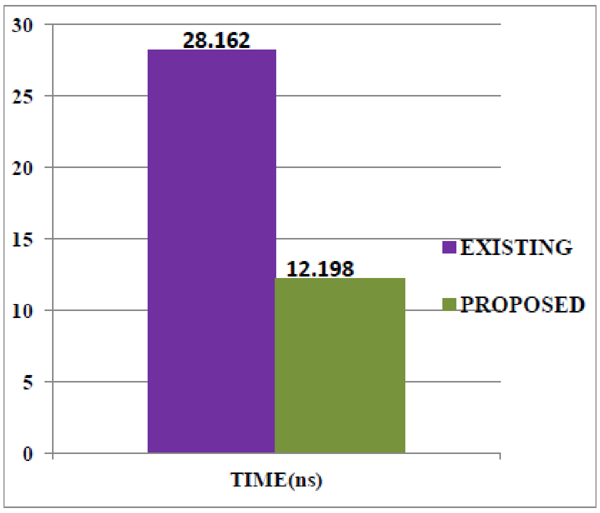

Fig 5: Comparison of Time (ns)

4. PERFORMANCE ANALYSIS

The feed forward FFT architectures have been programmed for the use in FPGAs. The target FPGA is a Spartan3 FPGA, XC3S50 -5 PQ208. This FPGA includes DSP48E blocks that can be used to carry out mathematical operations. In the proposed designs these blocks have been used to implement complex multiplication of the FFT.

Figure.4 compares the area of the proposed design to other equivalent higher-throughput radix-2^k pipelined FFT hardware architectures for the synthesis condition.

Figure.5 compares the time of the proposed designs to other 4-parallel and 8-parallel pipelined FFTs. As can be observed, the proposed designs achieve the highest throughput and low latency for 4-parallel and 8-parallel architectures. In the higher throughput can be achieved by restoring to 16-point radix-2^2 feed forward architectures.

5. CONCLUSIONS

This paper has presented a generalized approach to design efficient architectures for the computation of radix-2^2 pipelined FFT hardware architectures. The proposed architectures lead to low hardware complexity compared to the existing designs. Further higher parallel architectures can be developed using the proposed approach.

Finally, experimental results show that the designs are efficient in both area and performance, being possible as well as very low latencies.

6. REFERENCES

[1] L. Yang, K.Zhang, H. Liu, J. Huang, and S. Huang, “An efficient locally pipelined FFT processor”, IEEE Trans. Circuits Syst. II, Exp. Briefs, Vol.53, no.7, pp.585-589, Jul.2006.

[2] Y. W. Lin and C.Y. Lee, “Design of an FFT/IFFT processor for MIMO OFDM systems”, IEEE Trans. Circuits Syst. I, Reg. papers, Vol.54, no.4, pp.807-815, Apr.2007.

[3] M. Ayinala, M.J. Brown, and Keshab K.Parhi, “Pipelined Parallel FFT Architectures via Folding Transformations”, IEEE Trans.VLSI systems, Vol.20, No.6, June2012.

[4] Bevan M. Baas, “A low-power High-performance, 1024-point FFT Processor”, IEEE journal of solid state circuits, Vol.34, No.3, March1999.

[5] 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, 2634-2643, Dec.2009.

[6] Y. N. Chang, “An efficient VLSI architecture for normal I/O order pipeline FFT design”, IEEE Trans. Circuits Syst. II, Briefs, Vol.555, No.12, pp.1234-1238, Dec.2008.

[7] M. Garrido, O. Gustafsson, and J. Grajal, “Accurate rotations based on co-efficient scaling”, IEEE Trans. Circuits Syst. II, Exp. Briefs, Vol.58, N0.10, pp.662-666, Oct.2011.