Implementation of 32-Point FFT Processor for OFDM System

G. Soundarya, V. Jagan Naveen and D. Tirumala Rao

Abstract Due to the advanced VLSI technology, Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) has been applied to wide field in wireless Communication applications. For modern communication systems, the FFT/IFFT is very important in the OFDM. Because of FFT/IFFT is the key computational block to execute the baseband multicarrier demodulation and modulation in OFDM system. An implementation of the 32-point FFT processor with radix-2 algorithm in R2MDC architecture is the processing element. This butterfly- Processing Element (PE) used in the 32-FFT processor reduces the multiplicative complexity by using real constant multiplication in one method and eliminates the multiplicative complexity by using add and shift operation in the proposed method.

Keywords R2MDC · FFT · OFDM · Verilog

1 Introduction

Now a days, FFT processors used in wireless communication system should have faster execution and low power consumption [1]. These are the most important constraints of FFT processor. In FFT/IFFT block the most arithmetic operation is complex multiplication. This is the main issue in processor which consume more time, a large area and power. When large point FFT is to be design it increases the complexity. There are two methods to reduce the multiplication complexity; one method is to perform real and constant multiplication in place of complex

G. Soundarya (✉) · V.J. Naveen · D.T. Rao
Department of ECE, GMR Institute of Technology, Rajam, A.P, India
e-mail: soundarya6020@gmail.com

V.J. Naveen
e-mail: Jagannaveen.n@gmrit.org

D.T. Rao
e-mail: tirumalarao.d@gmrit.org

multiplication. The second method is to eliminate the non-trivial multiplication by the twiddle factors and processing with no complex multiplication. Many complicated design of communication system became feasible. There is a rapidly growing demand in communications for high quality video and voice etc. Orthogonal Frequency Division Multiplexing technology is an effective modulation scheme to meet the demand.

According to the literature survey various designs and implementation of FFT/IFFT have been done for OFDM systems. A novel 8-point FFT processor based on pipeline architecture is discussed [1]. High speed data transmission in ultra wide band spectrum by dividing the spectrum band into multiple bands and provides high efficiency is described [2]. A 16-point FFT butterfly PE reduces the multiplication complexity by using real and constant multiplication [3]. The circuit complexity is reduced by means of pipeline FFT/IFFT architecture and provides better performance in terms of area and speed at low frequency [4]. The present works focus on implementation of 32-point FFT, which is used as processing element in R2MDC architecture and non-trivial multiplications are eliminated by using add and shift method for high throughput.

2 OFDM

The Orthogonal Frequency Division Multiplexing is a wideband wireless digital communication technique that is based on block modulation. OFDM is a subset of frequency division multiplexing in which a single channel utilizes multiple sub-carriers on adjacent frequencies. In addition the sub-carriers in an OFDM system are overlapping to maximize the spectral efficiency. Sub-carriers in OFDM system are orthogonal to one another, thus they are able to overlap without interfering. It can support high-speed video communication along with audio with elimination of ISI and ICI.

3 FFT Algorithm

The FFT algorithms are based on fundamental principle of decomposing the computation of the discrete Fourier transform of a sequence of length N into successively smaller DFT's.

$$\begin{gathered} X[k] = \sum_{n=0}^{N-1} x[n] \cdot W_N^{nk} \\ \text{where } W_N^{nk} = e^{-j\frac{2\pi nk}{N}} \qquad 0 \le \mathrm{k} \le \mathrm{N}-1 \end{gathered} \tag{1}$$

Direct DFT calculations requires a computational complexity of $O(N^2)$.

3.1 Cooley-Tukey Algorithm

By using most popular Cooley-Tukey FFT algorithm, the complexity can be reduced to O(N.log$_r$N) [4, 5]. It is most universal of all FFT algorithms. Cooley-Tukey FFTs are those were the transform length is power of a basis r, i.e., N = r^S S —stages. These algorithms are referred to as radix-r algorithms. The most commonly used are base r = 2 and r = 4. Decomposition is important role in FFT. There are two decomposed types of FFT. One is decimation-in-time and other is decimation-in-frequency. In addition there is no difference in computational complexity between these two types of FFT. Since the low computational complexity of FFT algorithms is desired for high speed consideration in VLSI implementation, here we discuss the computational complexity of different algorithms (Fig. 1).

Basic Radix-2 butterfly processor shown in Fig. 2 consists of adder and complex subtraction. Besides that, an additional complex multiplier for twiddle factor W_N is implemented. The complex multiplication with the twiddle factor requires four real multiplications and two add/subtract operations [1, 3].

3.2 Complex Multiplication

Since complex multiplication is an expensive operation, we tend to reduce the multiplicative complexity of the twiddle factor inside the butterfly processor by calculating only three real multiplications and three add/subtract operations as in (3) and (4).

The twiddle factor multiplication:

$$R + jI = (X\ + jY) \cdot (c + jS) \tag{2}$$

However complex multiplication can be simplified:

$$R = (C - S) \cdot Y + Z \tag{3}$$

$$I = (C + S) \cdot X - Z \tag{4}$$

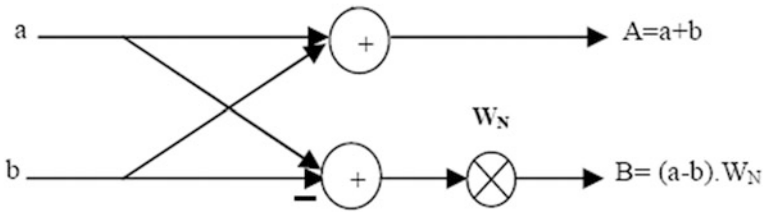


Fig. 1 Signal flow graph of DIF of FFT

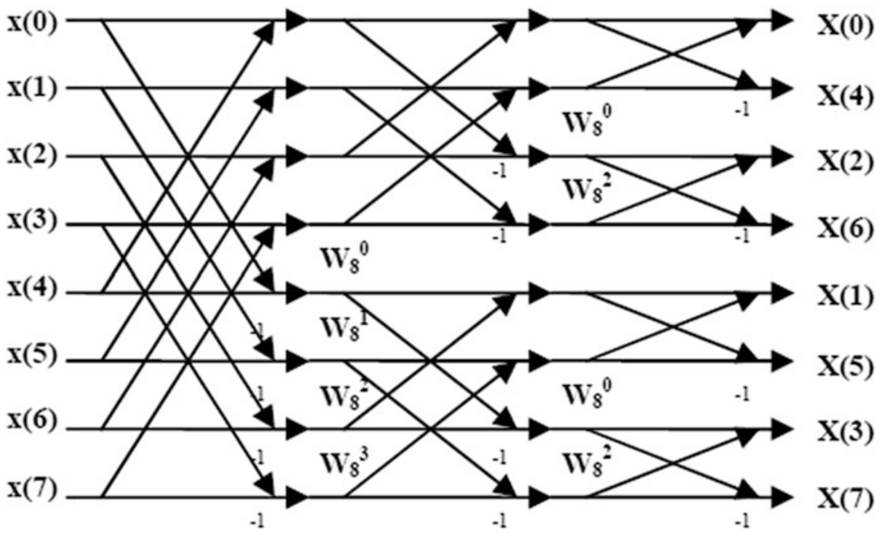


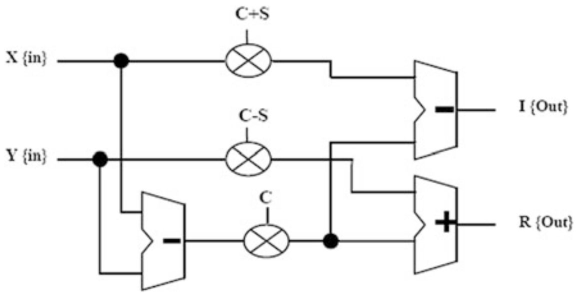
Fig. 2 Basic Butterfly computation

$$\text{with } Z = C \cdot (X - Y) \tag{5}$$

C and S are pre-computed and stored in a memory table. Therefore it is necessary to store the following three coefficients C, C + S and C − S. The implemented algorithm of complex multiplication used in this is three multiplications, one addition and two subtractions as show in Fig. 3.

In the 8-point FFT with radix 2 algorithm, the multiplication with $W_8^2 = -j$ and W_8^0 factors is trivial, the multiplication with W_8^2 simply can be done by swapping from real to imaginary part and vice versa, followed by changing the sign [6, 7]. The number of complex multiplication in this scheme of FFT is two: W_8^1, W_8^3, these are non-trivial complex multiplication was implemented with two multiplications for W_8^1 and three multiplications for W_8^3. Therefore, the number of real multiplications is 5. However this solution was not suitable in practice, because the first

Fig. 3 Implementation of complex multiplication



stage to process four different twiddle factors (trivial and Non-trivial multiplications) in pipeline architecture with one complex multiplier. Therefore it will require more elements in the structure to implement the two complex multiplications in addition to the two trivial multiplications in one block. This can be done by first method i.e., R2MDC. Another architecture solution was proposed in order to eliminate the complex multiplication inside the butterfly processor completely.

4 Pipeline FFT/IFFT Processor Architecture

4.1 R2MDC Architecture

Simplicity, modularity and high throughput are required for FFT/IFFT processors in communication systems. The pipeline architecture is suitable for those ends [5]. The sequential input stream in pipeline architecture unfortunately doesn't match the FFT/IFFT algorithm since the bloc FFT/IFFT requires temporal separation of data. In this case data memory is required in the pipeline processor to be rearranged according to FFT/IFFT algorithm as shown in Fig. 1. One of the straightforward approaches for pipeline implementation of radix-2 FFT algorithm is Radix-2 Multi-path Delay Commutator architecture which is shown in Fig. 4. It is a simplest way to rearrange data for the FFT/IFFT algorithm. The input data sequence are broken into two parallel data elements flowing forward, with correct distance between data elements entering the butterfly scheduled by proper delays. At each stage half of the data flow is delayed via the memory (Registers) and processed with the second half data stream.

4.2 ADD and Shift Method

Another method proposed eliminates the non-trivial complex multiplication with the twiddle factor (W_8^1, W_8^3) and implements the processor without complex multiplication. The proposed butterfly processor performs the multiplication with the

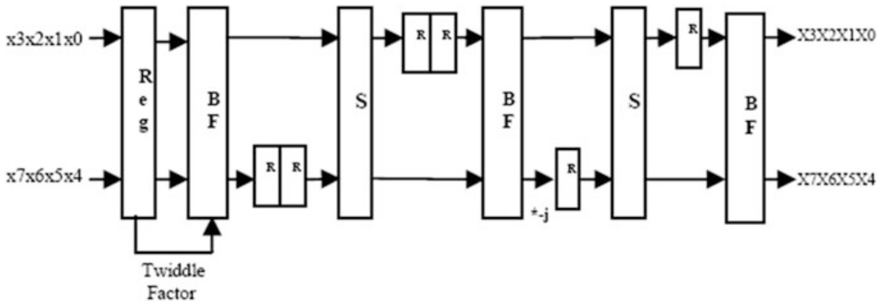


Fig. 4 R2MDC Architecture

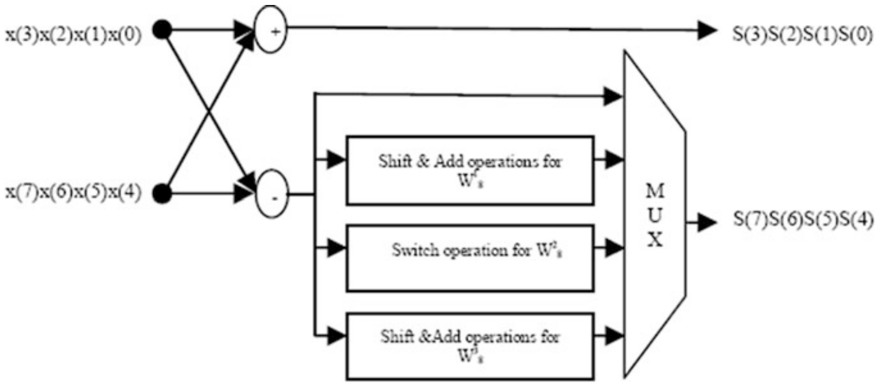


Fig. 5 Butterfly processor with no complex multiplication

trivial factors $W_8^2 = -j$ by switching from real to imaginary and imaginary real, with the factor W_8^0 by a simple cable. With the nom-trivial factors W_8^1, W_8^3 the processor realize the multiplication by factor $1/\sqrt{2}$ using hard wired shift and add operation as shown in Fig. 5.

5 Experimental Results

This simulation shows Figs. 6 and 7 the radix-2 operations. Two inputs are selected with the help of de-multiplexer which is stored inside the buffer block. After addition and subtraction of two inputs with a twiddle factor multiplication final FFT will come.

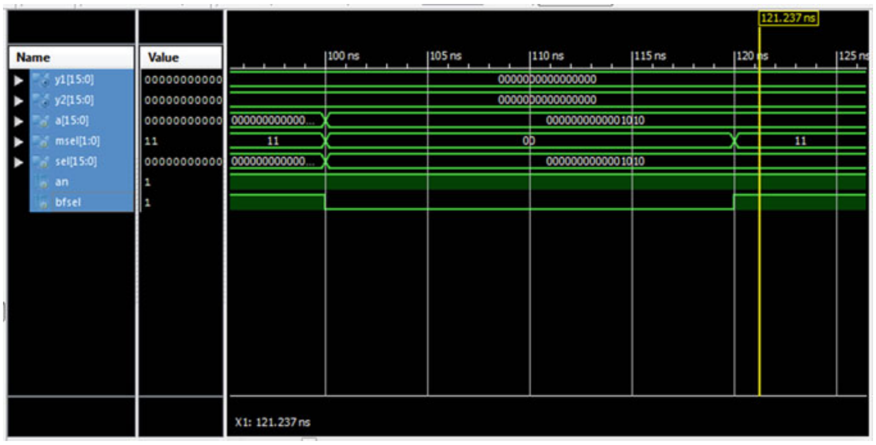


Fig. 6 Simulation result of R2MDC 32-point FFT

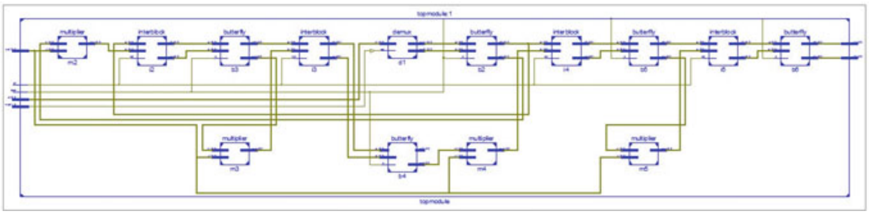


Fig. 7 RTL of R2MDC 32-point FFT

The verilog program is compiled in Xilinx 13.4 to generate a butterfly processor operation with the help of R2MDC architecture, Add and shift method.

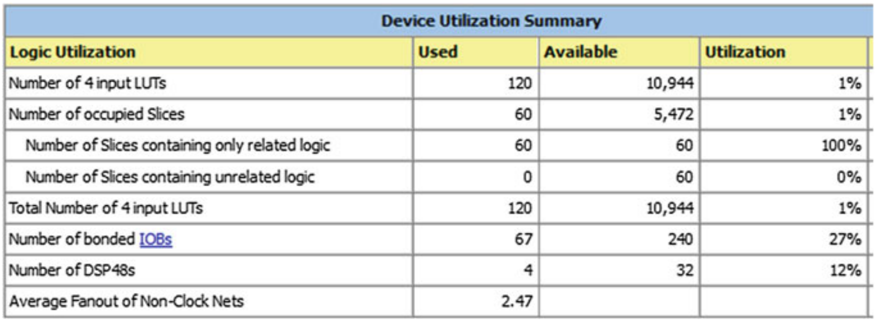
The add and shift method simulation shows, that the input after addition and subtraction of radix-2 process multiplied a real number of complex variable, and produce a final FFT.

5.1 Synthesis Report of R2MDC and ADD/Shift Method

The area analysis can be done by using Xilinx power analyzer and synthesis tool in Xilinx 13.4. The 32-point FFT computation with radix-2 in R2MDC was coded in verilog using Xilinx tool simulated and synthesized on vertex4. These results show the better performance when compared to 8-point FFT and speed will be more.

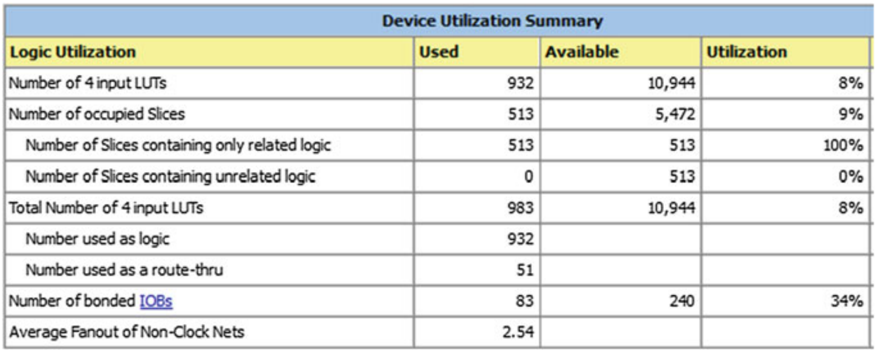
Tables 1 and 2: Shows the device utilization summary of the 32-point FFT for vertex-4 family and number of slice registers, number of slice LUTs, number of and bonded IOBs used. The RTL of R2MDC and Add and shift method are shown in Figs. 8 and 9.

Table 1 R2MDC 32-point FFT



Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	120	10,944	1%
Number of occupied Slices	60	5,472	1%
Number of Slices containing only related logic	60	60	100%
Number of Slices containing unrelated logic	0	60	0%
Total Number of 4 input LUTs	120	10,944	1%
Number of bonded IOBs	67	240	27%
Number of DSP48s	4	32	12%
Average Fanout of Non-Clock Nets	2.47		

Table 2 Add and shift 32-point



Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	932	10,944	8%
Number of occupied Slices	513	5,472	9%
Number of Slices containing only related logic	513	513	100%
Number of Slices containing unrelated logic	0	513	0%
Total Number of 4 input LUTs	983	10,944	8%
Number used as logic	932		
Number used as a route-thru	51		
Number of bonded IOBs	83	240	34%
Average Fanout of Non-Clock Nets	2.54		

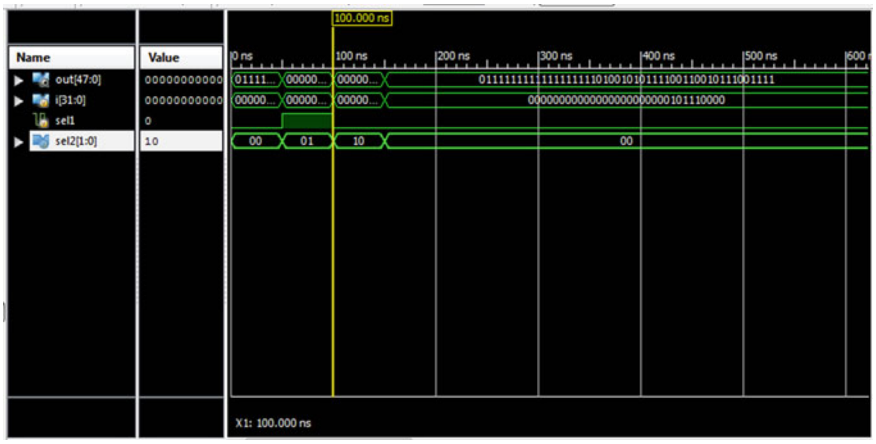


Fig. 8 Simulation result of add and shift method 32-point FFT

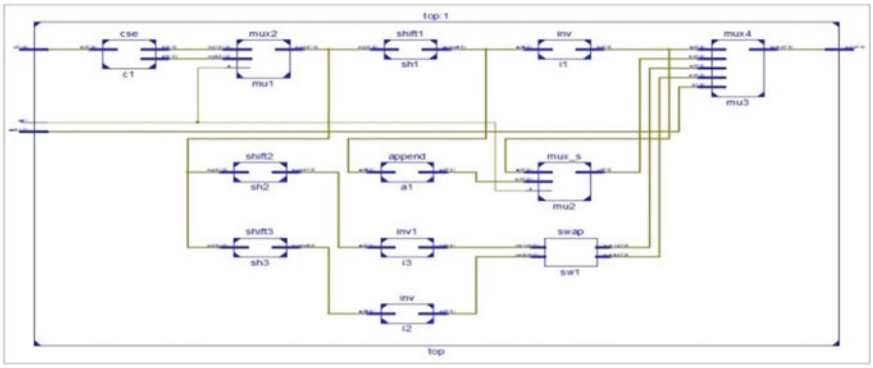


Fig. 9 RTL of ADD and shift method 32-point FFT

6 Conclusion

In this paper, two pipeline-based FFT Architectures are proposed. Both methods are applied on 8-point FFT and 32-point FFT. The 8-point FFT can perform limited operations where as 32-point FFT can perform large complex operations. To reduce computational complexity and increase hardware utility, we adopt different radix FFT algorithms and multiple-path delay commutates FFT architecture in our processors. The multi-path delay commutator FFT architecture requires fewer delay elements and different radix FFT algorithms require fewer complex multiplication. The proposed FFT processor architectures are suitable for various MIMO OFDM-based communication systems, such as IEEE802.11n and IEEE802.16 WiMAX, etc.

References

1. Verma, P., Kaur, H., Singh, M.: VHDL implementation of FFT/IFFT blocks for OFDM. In: Proceedings of International, Conference on Advances in Recent in Communication and Computing, PI 978-1-2244-51-4-3, Kerala, pp. 186–188 (2009)
2. Jinsing, X., Xiaochun, L., Haitao, W., Yujing, B., Decai, Z., Xiaolong, Z., Chaogang, W.: Implementation of MB-OFDM transmitter baseband based on FPGA. In: 4th IEEE (ICCSC 2008), 26–28 May 2008
3. Li, W., Wanhammar, L.: Complex multiplication reduction in FFT processor. In: SSoCC'02 Falkenberg, Sweden, Mar 2002
4. Preeti, G.B., Uma Reddy N.V.: Implementation of area efficient OFDM transceiver on FPGA. Int. J. Soft Comput. Eng. **3**(3), 144–147 (2013). ISSN 2231-2307
5. Prajapati, K., Sharma, A., Thakor, H.: Implementation of an optimized 8 point FFT in pipeline architecture for FPGA's system, vol. 1, Issue 4, (2014)
6. U.M. Baese: Digital Signal Processing with FPGA, 3rd edn. Springer, Berlin (2007)
7. Petrov, M., Glenser, M.: Optimal FFT architecture selection for OFDM receiver on FPGA. In: Proceedings of 2005 IEEE International Conference on Field Programmable technology, PI. 0-7803-9407-0, Singapore, pp. 313–314 (2005)
8. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex fourier series. Math. Comput. **19**, 297–301 (1965)
9. Li, W., Wanhammar, L.: An FFT processor based on 16 point module. In: Proceedings of Norchip Conference, Stckholm, Sweden, pp. 125–130 (2001)
10. Wang, B., Zang, Q., Ao, T., Huang, M.: Design of pipelined FFT processor based on FPGA. In: Proceedings of 2nd International Conference on (ICCMS'10), pp. 432–435 (2010)