Efficient Hardware Implementation of 1024 Point Radix-4 FFT

Senthilkumar Ranganathan, Ravikumar Krishnan, and H.S. Sriharsha

Abstract. Since FFT algorithm is extremely demanding task and has several applications in the areas of signal processing and communication systems, it must be precisely designed to induce an efficient implementation of the parameters involving area and performance. To fulfill this requirement an optimized architecture is demonstrated in this paper for computing 1024-point, Radix-4 FFT using FPGA and is majorly compared with Xilinx LogiCore™ FFT IP and found that proposed design is more efficient and effective in terms of area and performance. A novel architecture referred to as 2-D Vector Rotation and Complex Math Processor has been proposed in this paper. This single structure rotation helps in effectively carrying out the complex multiplications. The algorithm implements multiplexor hardware for computing the complex multipliers, thus consuming the minimal hardware resources. The entire RTL design is described using Verilog HDL and simulated using Xilinx ISim[TM]. This experimental result is tested on Spartan-6 XC6SLX150T. The result shows 557 LUT's, 837 Flip Flops, 3 DSP Slices, Maximum Frequency of 215 MHz. This is about 52% improvement in resource usage and 5% upgrade in the performance.

Keywords: FPGA, FFT, 2-D Vector Rotation, Complex Math Processor.

Senthilkumar Ranganathan
Institution of Engineers (India)
e-mail: r.senthilkumar.in@ieee.org

Ravikumar Krishnan
KCG college of technology, Chennai
e-mail: r.ravikumar.be@gmail.com

H.S. Sriharsha
Asmaitha Wireless Technologies Pvt. Ltd
e-mail: sriharsha.suresh@gmail.com

S.M. Thampi, A. Gelbukh, and J. Mukhopadhyay (eds.), *Advances in Signal Processing and Intelligent Recognition Systems*, Advances in Intelligent Systems and Computing 264,
DOI: 10.1007/978-3-319-04960-1_18,

1 Introduction

DFT is a very complicated task as it involves huge complex hardware [1]. Fast Fourier Transform is a tool to travel between the time domain and frequency domain that has wide application in the area of Communication, Astronomy, Earth science and in Optics [2]. So, there's a need for low cost digital signal processing device which can compute FFT without negotiating on performance and flexibility ratio. Of the many interests in the Universal research, the prime is efficient utilization of area without compromising performance. There is an endless demand for FPGA as a result of its simplicity in implementation and low cost compared to ASIC. FFT can be effectively enforced in FPGA because it is a low volume application and cost efficient solution. In recent years, Fast Fourier transform of interest are Split Radix algorithms [3], Twisted Radix algorithms [4], Parallel FFT [5], Mixed-Radix algorithms [6], Vector-Radix [7]. This paper proposes a design of 1024 sampling point, DIF Radix-4 FFT [12, 13] considering area and power as the main factors. This design does not imply to any FFT architecture based on pipeline design [13, 14, 15, 16, 17], instead a unique architecture for computing the complex multiplications and additions has been administered using Complex Math processor and Rotational structures that is mentioned in the further sections. The design is compared with the prevailing Xilinx LogiCORE^{TM} Fast Fourier Transform IP and found that prescribed design is more effectual in terms of resource utilization and performance for the Spartan-6 for a speed grade of -2 thus making more appropriate in terms of arithmetic value and power. The rest of the work is organized as follows, section 2 describes the general architecture of Radix-4 FFT algorithm, section 3 explains the customised architecture and related optimization techniques. 4, describes the implementation and verification with FPGA board and elaborated comparison of results achieved with the Xilinx FFT IP core. The conclusion is given in section 5.

2 Radix-4 FFT Algorithm

The discrete Fourier Transform of a complex time series is given by,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \quad \text{where } k = 0,1,2...,N-1 \tag{1}$$

According to the Divide and Conquer rule [8], the complex series will be decimated in time and frequency. In Radix-4 implementation, the problem is decimated by four time units with period of (N/4) after the twiddle factor

generation [9]. The figure 1 shows the architecture of fundamental Radix-4 Butterfly unit. The four recurrent results of every butterfly unit is expressed as,

$$y(0) = (x(0) + x(1) + x(2) + x(3)) \tag{2}$$

$$y(1) = (x(0) - jx(1) - x(3) + \mathrm{j}\,x(4)) * \mathrm{W}_N^k \tag{3}$$

$$y(2) = (x(0) - x(1) + x(2) - x(3)) * \mathrm{W}_N^{2k} \tag{4}$$

$$y(3) = (x(0) + \mathrm{j}\,x(1) - x(3) - \mathrm{j}\,x(4)) * \mathrm{W}_N^{3k} \tag{5}$$

The detailed description of Radix-4 FFT is described in the paper [10].

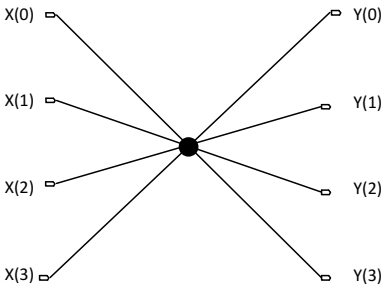

Fig. 1 Radix 4 Butterfly Diagram with 4 samples

3 Proposed Radix 4 Butterfly Architecture

The block diagram of four input sequence, Radix-4 Butterfly unit is shown in figure 2. This architecture contains the description of all the computations concerned in single Butterfly unit. Two complex Math Processors (CMPs) are used to perform the real and complex arithmetic's. Depending on the selector inputs, the output of CMP is obtained. The same selection lines are used for decisive addition or subtraction in the add/sub unit. The twiddle factors are multiplied with the output of complex adder by using the standard complex multiplier. The table describing the selection techniques is as shown below,

Table 1 Selection Technique performed by CMP

Sel[1:0]	CMP1	CMP2	ADD / SUB
00	x(0) + x(1)	x(2) + x(3)	c1 + c2
01	x(0) –jx(1)	x(2) –jx(3)	c1 – c2
10	x(0)-x(1)	x(2)-x(3)	c1 + c2
11	x(0)+jx(1)	x(2)+jx(3)	c1 - c2

Whenever the select line is 00, twiddle factor is considered as unity. Depending on the least significant bit of select inputs addition or subtraction is performed.

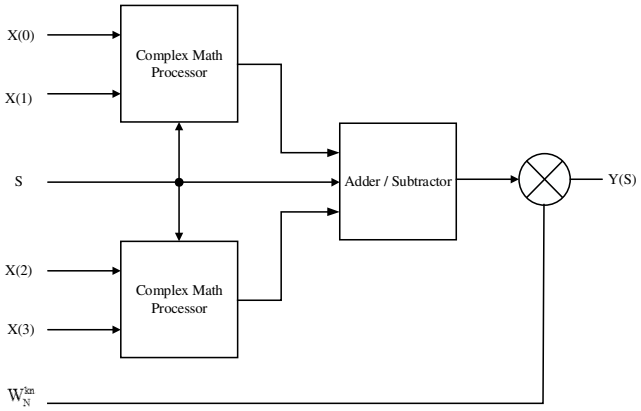

Fig. 2 Proposed Butterfly Architecture

3.1 Complex Math Processor and 2 – D Vector Rotation

This unit describes the methodology to calculate one single output of Radix-4 complex butterfly unit. A common architecture is outlined to compute all Radix-4 butterfly outputs considering just one twiddle factor input at a time. The processor simply uses the multiplexor, adder and subtractor hardware resources as shown in figure 3. The multiplication of imaginary term 'j' is achieved by the 2-D Vector Rotation technique. The vector rotation is performed for equation (3) and (5). The rotation is achieved using the multiplexor. The 2D quadrant is swapped with respect to (+j) and (-j) corresponding to +90 degree and -90 degree rotation. This sign change technique is incorporated using adder / subtractor circuit.

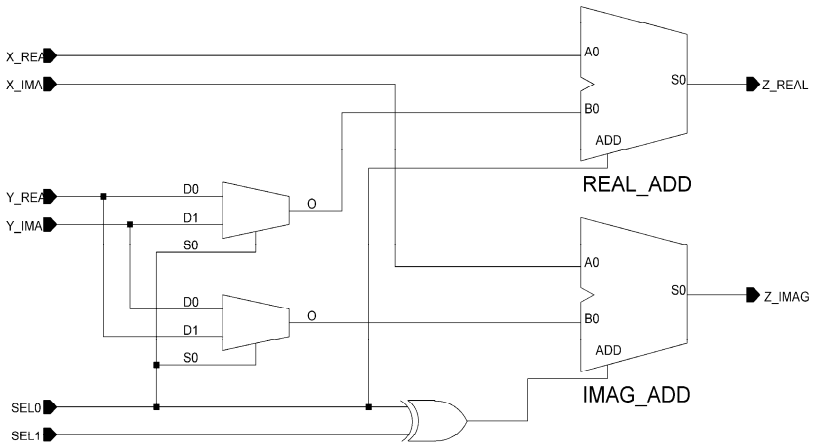

Fig. 3 Complex Math Processor Architecture

3.2 1024 Point DIF Radix-4 FFT Processor

The 1024 Radix-4 point FFT uses two Block RAMs for accessing input and output data. These two Block RAMs acts as a Ping-Pong Buffers. A multiplexor at the input side selects the source of data and stores it in Block RAM 1. After processing the sequence, Block RAM 2 holds the output data. Address Generator unit (ADG) is responsible for generating the address in every stage based on indexing. The twiddle factors are stored serially in the ROM and accordingly select lines are chosen. The outputs are generated sequentially from the Block RAM 2. The figure 4 shows the complete FFT Processor design.

Initially, at the input stage of the computation, Block RAM 1 is used for reading and Block RAM 2 will be idle. After computing the first stage of Butterfly unit, Block RAM 2 is used for writing the output sequences. The cycle of operation of two Block RAMs switches in every FFT stage as showed in the below table 2.

Table 2 Block RAM usage in every stage of computation

Stages	Block RAM 1	Block RAM 2
Input	Write	-
Stage 0	Read	Write
Stage 1	Write	Read
Stage 2	Read	Write
Stage 3	Write	Read
Stage 4	Read	Write
Output	-	Read

4 Implementation Results

The submitted novel design involving the Complex Math Processor and 2-D Vector Rotation is described in Verilog HDL and has been functionally verified. The timing simulations are carried out in Xilinx ISim™ as shown in figure 6. The entire RTL has been successfully synthesized using Spartan -6 XC6SLX150T FPGA.

The presented design is compared with Xilinx LogiCORE™ IP Fast Fourier Transform (v8.0) [11] and with [12] which is based on pointer FIFO embedded with gray code counters and found a major difference in performance and resource usage. The following table presents the architectural implementations when compared to the Xilinx LogiCore™ FFT IP synthesized for specific Spartan-6 architecture with speed grade of -2. The slice count, block RAM count, and XtremeDSP slice count are listed. The maximum clock frequency is also listed with the transform latency.

Implementation resources, as reported by Xilinx® ISE 14.5 synthesis tool, are summarized in table 3 for the proposed FFT Processor. The performance of the FFT core can be estimated by the clock cycle number, required to compute a full 1024-point input signal. There are 5 Butterfly stages containing 256 individual complex Butterflies. 1039 Clock cycle are required for the computation of each stage and then be schematized as shown in Figure 5. FFT processor runs at 215 MHz with the latency of 24.16μS as indicated in below table.

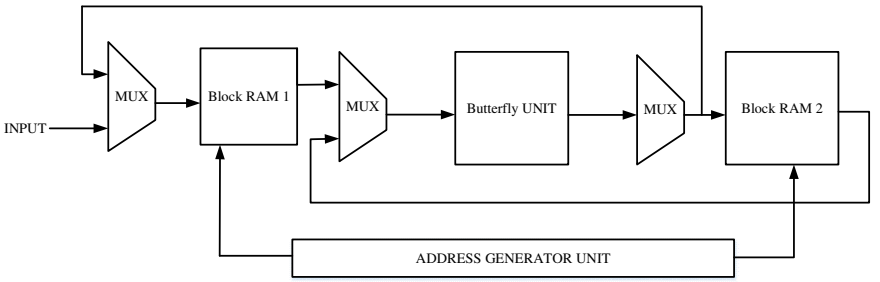

Fig. 4 Radix -4 FFT Processor

Table 3 Implementation Results of Xilinx LogiCORE IP and proposed design

Parameter	Xilinx LogiCORE FFT IP	Our Design
Channel	1	1
Point Size	1k	1k
Implementation	R4	R4
Configurable Point Size	NO	1024
Input Data Width	16	16
Phase factor Width	16	16
Scaling type	S	S
Rounding Mode	C	C
Output Ordering	N	N
Memory Type	B	B
Xilinx Part	XC6SLX150T	XC6SLX150T
LUT/FF Paris	1750	837
LUTs	1438	557
FFs	1608	837
9k Block RAMs	10	6
XtremeDSP slices	9	3
Max clock frequency	206	215
Latency (clock cycles)	3453	5195
Latency(Micro Sec)	18.27	24.16

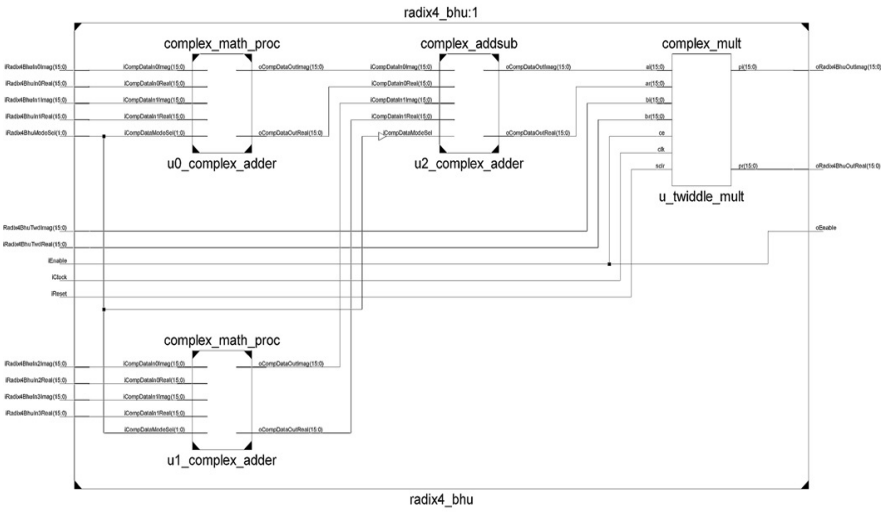

Fig. 5 RTL Design of 1024 Point DIF FFT Processor

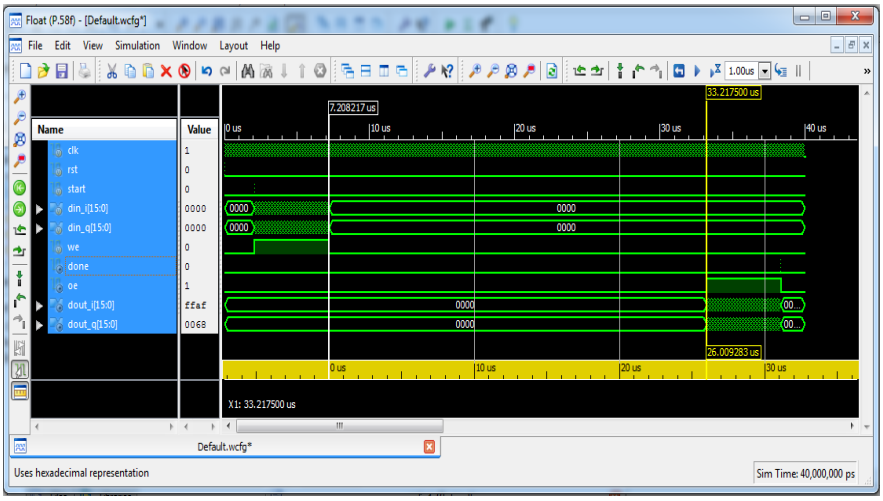

Fig. 6 Xilinx ISim[TM] Simulation Result of FFT Processor

5 Conclusion

The cumulative work of this paper is to formulate an FFT algorithm that would consume less area and achieve the greater frequency of operation than the Xilinx LogiCore™ FFT IP. This paper illustrates the custom designed Complex Math Processor and 2 – D Vector Rotation approach for multiplying the imaginary

terms. This architecture uses very minimal resource for complex manipulations thus creating the low power and optimized design of 1024 point Radix-4 FFT. Finally the Place and Route (PAR) report were compared as indicated in the Table 3, where a significant difference in engaged resource and performance is noted. The comparison study shows that around 52% resource minimization and 5% increase in throughput with a little change rate in latency clock cycle is achieved.

Acknowledgements. The authors would like to thank Prof. Chandrasekaran Subramaniam, Jaraline Kirubavathy, Subhashini Vaidyanathan for their valuable technical support and making it success.

References

1. Burrus, C.S., Perks, T.W.: DFUFFT and Convolution Algorithms. Wiley Interscience, New York (1985)
2. Cooley, J.W., Lewis, P.A.W., Welch, P.D.: Historical notes on the fast Fourier transform. Proc. IEEE 55, 1675–1677 (1967)
3. Shu, C.J., Peng, X.: Dept. of Electron. & Eng. Tsinghua Univ., Beijing
4. Mateer, T.: Ph.D. dissertation, Dept. Mathematical Science., Clemson Uni., Clemson., SC
5. Silvia, M., Giancarlo, R., Gaetano, S.: A parallel fast Fourier transform. Mod. Phy. C 10, 781–805 (1999)
6. Grioryan, A.M., Agaian, S.S.: Split mangeable efficient algorithm for Fourier and Hadamard transform. IEEE Trans. Signal Processs. 48(1), 172–183 (2000)
7. Chan, I.C., Ho, K.L.: Split vector-radix fast Fouriet transform. IEEE Trasn. Signal Process. 40, 2029–2040 (1992)
8. Cooley, J.W., Tukey, W.: An Algorithm for the Machine Calculation of Complex Fourier Series. Math. of Computations 19, 297–301 (1965)
9. Knight, W.R., Kaiser, R.: A Simple Fixed-Point Error Bound for the Fast Fourier Transform. IEEE Trans. Acoustics, Speech and Signal Proc. 27(6), 615–620 (1979)
10. Saidi, A.: Decimation-in-Time-Frequency FFT Algorithm. In: Proc. IEEE International Conf. on Acoustics, Speech, and Signal Processing, vol. 3, pp. 453–456 (1994)
11. Xilinx® Logic core™, Fast Fourier Transform V8.0, Xilinx® (2012)
12. Zhong, G., Zheng, H., Jin, Z., Chen, D., Pang, Z.: 1024-Point Pipeline FFT Processor with Pointer FIFOs based on FPGA. In: 2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip (2011)
13. He, H., Guo, H.: The Realization of FFT Algorithm based on FPGA Co-processor. In: Second International Symposium on Intelligent Information Technology Application, vol. 3, pp. 239–243 (December 2008)
14. Oh, J.-Y., Lim, M.-S.: Area and power efficient pipeline FFT algorithm. In: IEEE Workshop on Signal Processing Systems Design and Implementation, November 2-4, pp. 520–525 (2005)

15. Wang, H.-Y., Wu, J.-J., Chiu, C.-W., Lai, Y.-H.: A Modified Pipeline FFT Architecture. In: 2010 International Conference on Electrical and Control Engineering (ICECE), pp. 4611–4614 (June 2010)
16. Sukhsawas, S., Benkrid, K.: A high-level implementation of a high performance pipeline FFT on Virtex-E FPGAs. In: Proceedings of the IEEE Computer society Annual Symposium on VLSI, February 19-20, pp. 229–232 (2004)
17. He, S., Torkelson, M.: Design and implementation of a 1024-point pipeline FFT processor. In: Proceedings of the IEEE 1998 Custom Integrated Circuits Conference, May 11-14, pp. 131–134 (1998)