Area Optimized and Frequency Efficient 1024 Point Radix-2 FFT Processor on FPGA

Md. Ali Ghazi Islam[1], Kazi Nikhat Parvin[2], Md. Zakir Hussain[3*]
[1]M. E. Student, ECED, Muffakham Jah College of Engineering and Technology, Hyderabad, Telangana, India.
Email: mdalighazi123@gmail.com
[2]Assistant Professor, ECED, Bhoj Reddy Engineering College for Women, Hyderabad, Telangana, India.
Email: iamnikhatparvin@gmail.com
[3]Assistant Professor, ECED, Muffakham Jah College of Engineering and Technology, Hyderabad, Telangana, India.
Email: zakirhussainsm@gmail.com
*Corresponding Author

Abstract: This paper presents optimized area and frequency efficient Fast Fourier Transform (FFT) processor using radix-2 Decimation in Time (DIT) algorithm. The proposed FFT processor is a complex FFT processor where a time-multiplexed approach to the butterfly of 1024 point, fixed, 32-bit, based on Field Programmable Gate Array (FPGA) is designed. The architecture is based on burst I/O and the pipelined-streaming I/O structure in the butterfly module and the ping-pong operation which is clocking at 480 MHz on Xilinx vertex 6 xc6vlx550t-2ff1759.

Keywords: Fast fourier transform, Field programmable gate array, Ping-pong operation, Pipelined-streaming I/O, Time-multiplexed butterfly.

I. Introduction

The Fast Fourier Transform (FFT) is one of the most important algorithms in signal processing. Many hardware FFT architectures have been proposed with the aims of speeding up the calculation of the FFT and reducing the amount of hardware resources. Computing the Discrete Fourier Transform (DFT) [1] of 'N' points in the naive way takes O(N2) arithmetical operations. To calculate DFT with reduced number of arithmetical operations from O(N2) to N/2(log2N) FFT algorithm [2] is performed, by taking full advantage of Symmetry and Periodicity of Twiddle Factors. Cooley and Tukey [3] proposed the Fast Fourier Transform (FFT) algorithm, their work was first published in 1965.

In the realization of FFT, the most generally hardware realization methods are included of DSP (Digital Signal Processor), FFT dedicated chip and FPGA [4]. General purpose DSP programmable chips are used for the implementation of DSP algorithms, for high performance applications, special purpose fixed function DSP chipsets or FPGAs are used.

The FPGA is suitable for the high-speed signal processing system owing to its parallel signal processing architecture. FPGA is suitable for the FFT algorithm and has superiorities in performance, costing and power consumption. It is realized by the related EDA (Electronic Design Automation) software and hardware description language. In modern signal processing, the requirements of high speed and reliability are becoming a hot research point. Furthermore, many researchers are researching in combining the real-time requirement of FFT with the flexibility design by the FPGA, realizing the optimal configuration of the parallel algorithm and hardware structure, and improving the FFT processing speed.

According to the requirements, this paper presents a method which realizes the FFT operation based on the FPGA [5]. The FFT design coded in Verilog HDL [6], The FFT design is synthesized on the Xilinx ISE 14.7.

II. Cooley-Tukey FFT Algorithm

The Cooley-Tukey FFT algorithm [3] provides a systematic solution with a moderate computational complexity. It is based on the observation that multiple operations can be shared when calculating the output frequencies of the FFT. This is done by decomposing the equation of the DFT. The most common decompositions are decimation in time (DIT) and decimation in frequency (DIF).

The DIT decomposition separates the sequence x[n] into its even and odd samples, whereas the DIF decomposition is

applied on the output sequence X[k].

$$X[k] = \sum_{n=0}^{N-1} x[n]\, W_N^k \text{ , k= 0,1,…..,N-1}$$

Where $W_N^k = e^{-j\frac{2\pi}{N}nk}$

The odd part and even part of x(n) using radix-2 DIT in (l), obtain the following equations can be obtained.

Even terms Odd terms

$$X(k) = \sum_{n=0}^{N/2-1} (x[2n]\, W_N^{2nk}) + \sum_{n=0}^{N/2-1} (x[2n+1]\, W_N^{(2n+1)k}) \forall N \in \mathbb{Z}$$

Where in both decompositions, the N-point DFT is transformed into two N/2-point DFTs. By applying the procedure iteratively, each step halves the number of points of the DFTs, which finally leads to 2-point DFTs. One large computation is reduced to several sequential smaller computations which lead to the radix-2 butterfly as shown.

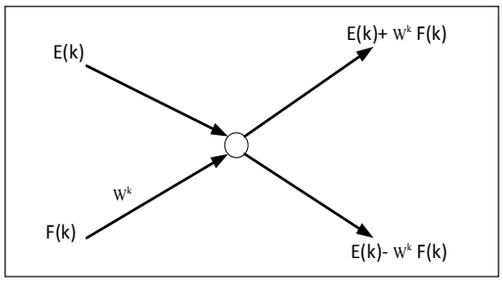

Fig. 1: Basic Butterfly Element

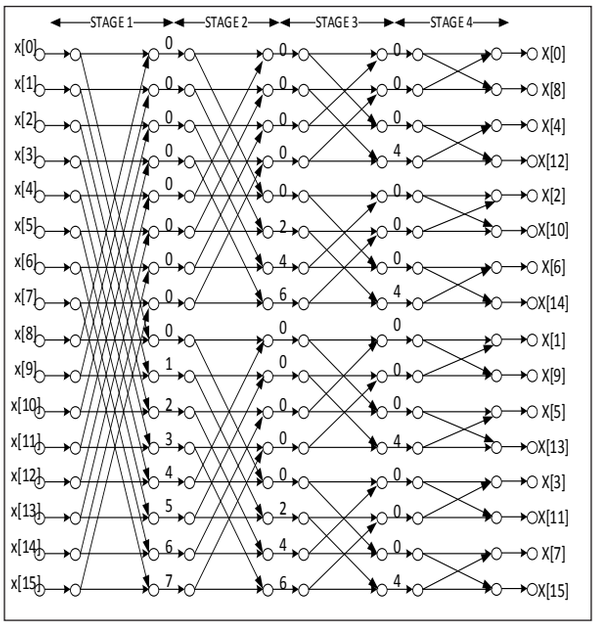

Fig. 2: 16-Point Radix-2 DIFFFT Flow Graph

III. FFT Implementation

In any FFT architecture, for good performance efficient addressing plays the key role. We efficiently did in place computation in this DIT FFT architecture using dual port RAM's which best suits for radix-2 FFT. The FFT system diagram [1] is shown below.

Fig. 3 system diagram shows dual RAM ping pong FFT architecture, where the data reading, processing and writing is done from one RAM to another at each stage and vice versa. To increase the throughput, we are using an extra block RAM named 'Block RAM 3' as shown in Fig. 3. Here user no need to wait until the first set data is read and user can read once the output is available while writing the second set of data. In this architecture, we are using pre-computed twiddle factor values.

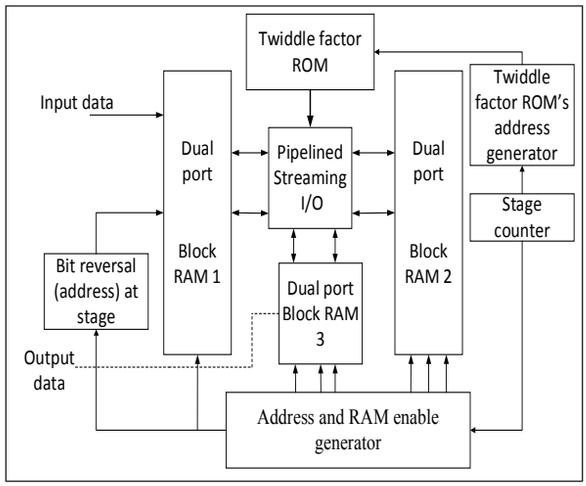

Fig. 3: FFT System Diagram

We designed a FFT architecture where we are using pipelined, Streaming I/O which allows continuous data processing. Radix-2 butterfly block as show in Fig. 4 is 9 stages pipelined as it involves complex multiplications resulting bottle neck for maximum frequency of operation. Where 6 clock cycles for multiplication, 2 clock cycles for addition and 1 clock cycle for signed shift operation.

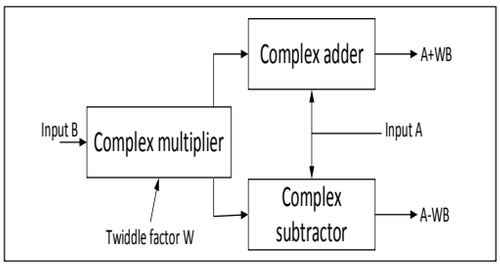

Fig. 4: Radix-2 Butterfly Operation

For one complete FFT operation 512 radix-2 butterflies operation requires total of 10 such stages takes in which 9 clock

cycles for pipelining, 8 clock cycles for internal read and write delays. For 1024 radix-2 butterflies operation to load the input data which is nearly 6320 clock cycles for one complete FFT operation.

TABLE I: THREE BITS EXAMPLE OF BIT REVERSAL OPERATION

Index	Address
000	000
100	001
010	010
110	011
001	100
101	101
011	110
111	111

As told in OFT summarizing steps, input should be stored in the reversal address of a one-bit incremental counter. Bit reversal block takes care of reversing input addresses.

A. Pipelined, Streaming I/O

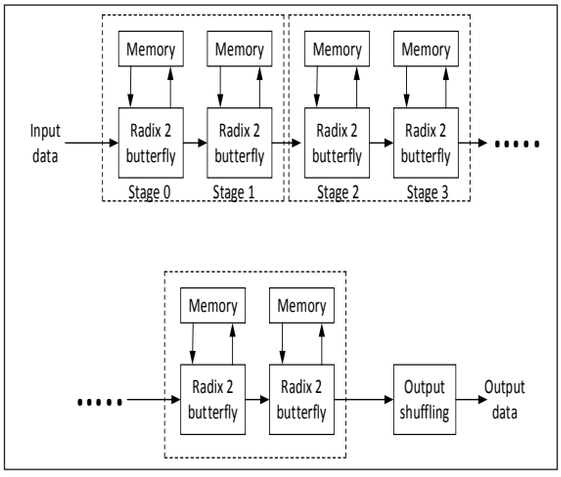

Fig. 5: Pipelined Streaming I/O

The Pipelined, Streaming I/O allows continuous data processing. This solution pipelines several Radix-2 butterfly processing engines to offer continuous data processing. Each processing engine has its own memory banks to store the input and intermediate data. The core has the ability to simultaneously perform transform calculations on the current frame of data, load input data for the next frame of data, and unload the results of the previous frame of data. The user can continuously stream in data and, after the calculation latency, can continuously unload the results.

This architecture covers point sizes from 8 to 65536. The user has flexibility to select the number of stages to use block RAM for data and phase factor storage. The remaining stages use distributed memory as shown in Fig. 5.

B. Radix-2, Burst I/O

The Radix-2, Burst I/O architecture uses the same iterative approach as Radix-4, but the butterfly is smaller. This means it is smaller in size than the Radix-4 solution, but the transform time is longer. It uses one Radix-2 butterfly processing engine (Fig. 6). After a frame of data is loaded, the input data stream must halt until the transform calculation is completed.

Then, the data can be unloaded. As with the Radix-4, Burst I/O architecture, data can be simultaneously loaded and unloaded when the output samples are in bit reversed order.

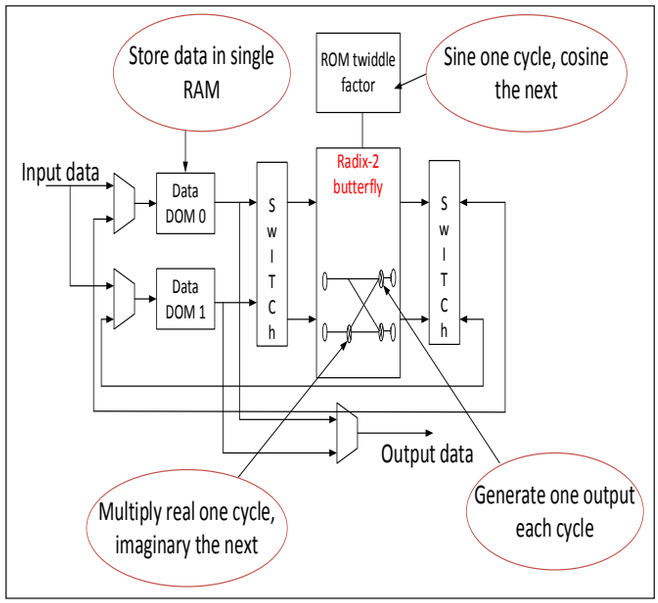

Fig. 6: Radix-2, Burst I/O

This solution sup- ports point sizes from 8 to 65536. Both the data memories and phase factor memories can be in either block RAM or distributed RAM (the latter for point sizes less than or equal to 1024).

C. Address Generation for Twiddle Factor ROM

As mentioned, for high throughput and performance of this design efficient addressing is important. To generate addresses to both Twiddle factor ROM and 'In place' computation Block RAM's we used only simple shift operations. The logic to generate twiddle factors addresses to ROM addr_twid, as shown in Fig. 6 is as follows in Table II:

Table II: Logic to Generate Twiddle Factors Addresses to ROM

Addresses at	Stage 1	Stage 2	Stage 3	Stage 4
n=2	0	0,1	-	-
n=3	0	0,2	0,1,2,3	-
n=4	0	0,4	0,2,4,6	0, 1, 2, 3, 4, 5, 6, 7

As shown in Fig. 7 initial shifter value is loaded as N/2 (Number of twiddle factors for given 'N') with left shifted by one. The one bit counter 'K' is designed such that the logic is as follows:

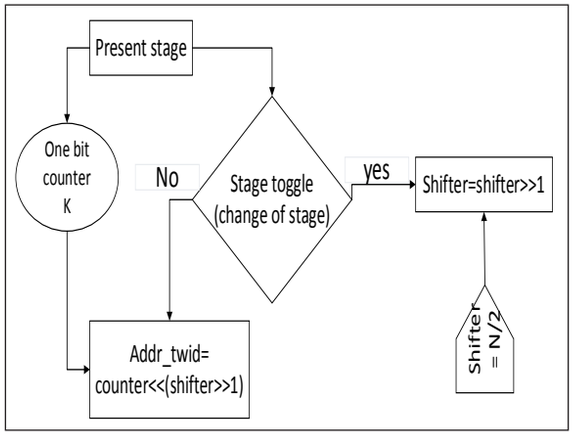

Fig. 7: Flow Chart for Address Generation to the Twiddle Factor ROM

Table III: Logic of One Bit Counter 'K'

Counter value at	Stage 1 K=	Stage 2 K=	Stage 3 K=	Stage 4 K=
n=2	0,0,0..	0,1,0,1	-	-
n=3	0,0,0	0,1,0,1	0,1,2,3 0,1,2,3	-
n=4	0,0,0	0,1,0,1	0,1,2,3 0,1,2,3	0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7

At each stage toggle (i.e. is change of stage) shifter shifts its value left by l (or divided by 2). The shifter values for N = 16 at stage 1, stage 2, stage 3, stage 4 are 8, 4, 2 and 1 respectively.

D. Address Generation for Block RAM's

In radix-2 FFT as shown in Fig. 4, if the first input A address (Address A) to the Block RAM 1 is 'M' then the input B address (Address B) can be calculated as 'M + 2^n/2' where 'n' is present stage. The logic to calculate input addresses in block RAM is shown in Table IV.

Table IV: Logic to Calculate Input Addresses in Block RAM

	M=	M + 2"/2 =
n=1	0,2,4,6"	1,3,5,7"
n=2	0,1,4,5....	2,3,6,7"
n=3	0,1,2,3"	4,5,6,7"

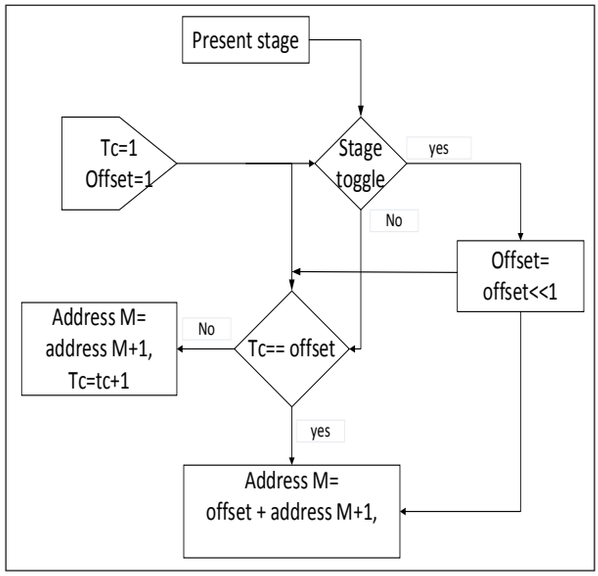

Fig. 8: Flow Chart for Address Generation to the Block RAM's

Fig. 8 the logic flow to generate value of 'M', where 'tc' in Fig. 8 is the terminal counter which resets to '1' if its value equals to the value of offset. It adds offset address value to the present address and resumes the normal counter operation for address and terminal counter.

IV. Comparison and Analysis

We implemented this FFT architecture in Xilinx xc6vlx550t-2ff1759 vertex 6 FPGA for length 1024 and the comparison results are shown in Table V, this design is operated on 480 MHz frequency. The proposed work utilizes less area with less number of slice LUT's and flip flop's.

Table V: Comparison with Previous Work

	[7]	Proposed work
Algorithm	Radix-2	Radix-2
Chip type	Vertex-6, xc6vlx550t	Vertex-6, xc6vlx550t
Number of points	1024	1024
Operating frequency	385 MHz	480 MHz
Number of slices	2633	2059

TABLE VI: COMPARISON OF DEVICE UTILIZATION SUMMARY

Logic utilization	Available	Used in [7]	Used in proposed work
Number of slice flip flops	687,360	2663	2059
Number of slice LUT's	343680	1883	1662
Number of occupied slices	85920	722	689
Number of RAMB36E1	632	8	3
Number of DSP48E1'S	864	17	6

V. CONCLUSION

In this paper, we have proposed a FFT architecture which has less utilization of number of LUTs and Flip Flops compared with other architecture presented. By efficient addressing using only simple shift operations, we reduced number of clocks for reading and processing of address logic, achieved area and frequency optimized FFT core with minimum number of clock latency and maximum clock frequency (480 MHz).

REFERENCES

1. A. V. Oppenheim, R. W. Schaefer, and J. R. Buck, "Discrete-time signal processing," in 2nd ed. Prentice-Hall, Englewood Cliffs, NJ, 1998.
2. A. E. Zonst, "Understanding the FFT: A tutorial on the algorithm and software for laymen," Citrus press, 1995.
3. J. Cooley, and J. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation,* vol. 19, no. 90, pp. 297-301, 1965.
4. H. Guangshu, "Digital signal processing-theoretical algorithm and realization," Peking: Tsinghua University Press, 2005.
5. U. Meyer-Baese, "Digital signal processing with field programmable gate arrays," Springer International Publication, 2007.
6. W. F. Lee, "Verilog coding for logic synthesis," A John Wiley & Sons, Inc., Publications, 2005.
7. V. Kumar, D. Selvakumar, and P. M. Sobha, "Area and frequency optimized 1024 point radix-2 FFT processor on FPGA," *International Conference in (VLSI-SATA),* 2015.
8. Xilinx, "Fast fourier transform logic IP core v7.1 product specifications DS260," 15 March 2011.