DESIGN, SIMULATION AND COMPARISON OF 256-BITS 64-POINTS RADIX-4 AND RADIX-2 ALGORITHMS

[1]G SUDHA KIRAN, [2]P BRUNDAVANI

[1]Post Graduate in VLSI, Annamacharya Institute of Technology and Sciences, Rajampet [2]Assistant Professor, Department of ECE, Annamacharya Institute of Technology and Sciences, Rajampet.
[1]sudhakiran.495@gmail.com, [2]brundavenky@yahoo.com

Abstract- Technical device uses and users are rapidly increasing, besides this the device accuracy and fastness is more important requirement. In the view of this, present project came into existence and its focus on the development of the Fast Fourier Transform (FFT) algorithm, based on Decimation-In- Time (DIT) domain, Radix-4 algorithm, which uses VHDL as a design entity and their Synthesis by Xilinx Synthesis Tool of 8.1 version. The synthesis results show that the computation for calculating the 256 bits 64-point FFT is efficient in terms of speed.

Key words: Butterfly, DFT, DIFFFT, DITFFT, FFT, Radix.

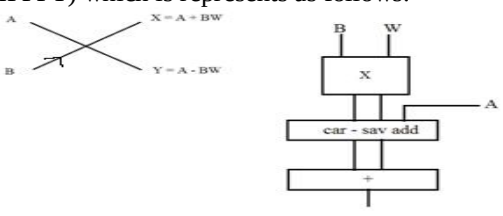
I. INTRODUCTION

The fastness of the system depends on their intra and inter peripherals, the intra peripherals depends on the designers choice and the inter peripherals depends on the users choice. Designers choice includes components, algorithms etc, users choice includes inputs, external devices, signals etc.

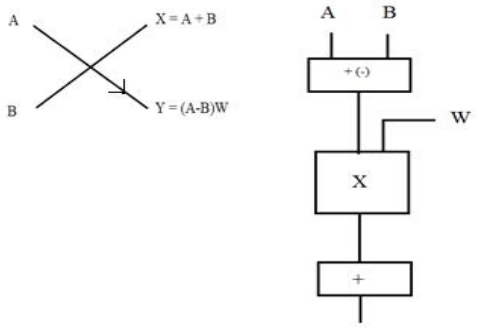
The previous project had focused on radix-2 algorithm which has more delay. To overcome this problem there is a need to change in the algorithm majorly, the present project deals with change in the algorithm called radix-4 algorithm and focus on the design and implementation of 256-bits 64-points Fast Fourier Transform (FFT) for a Field Programmable Gate Array (FPGA) kit. The coding is done in VHDL, simulation and synthesis can be done by using ModelSIM ISE and Xilinx ISE Design Suite respectively.

II. RADIX-2 ALGORITHM

To speed up the DFT computations, FFT came into existence. There are many ways to decompose FFT among them the easiest and simplest way is Radix-2. The so called radix-2 is due to its base is equals to 2 and the representation is 2^M where M represents the index/stage and its value is a positive integer. The computation of radix-2 made up of butterflies called Radix-2 butterflies. Depending on the type of decimation in the different domains it is of two types, they are Decimation in Time FFT (DITFFT) and Decimation in Frequency FFT (DIFFFT) which is represents as follows:



(a) DITFFT butterfly



(b) DIFFFT butterfly

Fig. 1: Radix-2 FFT butterfly diagrams

For radix-2 there are two inputs and two outputs and the inputs are arranged in bit reversal order this is because of saving the memory locations and outputs are in a normal order its only for DITFFT and vice versa for DIFFFT. The given whole number decides the number of stages as $\log_2 N$, and each stage consists of blocks it can be given as $N/2^{stage}$ and each block contains butterflies it can be given as $2^{stage-1}$, each stage includes the twiddle factors. Clearly analyze that each and every stage having complex computations or simply complex multiplications and complex additions. Generally radix-2 having $N/2\log_2 N$ complex multiplications and $N\log_2 N$ complex additions.

Mainly this algorithm depends on these computations, if number of stages increases proportionally computations are increases. For example: for 16-point FFT the complex multiplications and additions are 32 and 64 respectively and for 32 point FFT these are 90 and 180 respectively and so on. If number of inputs is more it became impossible to calculate by using R2.

Now examine for 64-points how many computations it requires, how much delay for obtaining the output by processing the applied input and how many slices it requires for computations etc as follows:

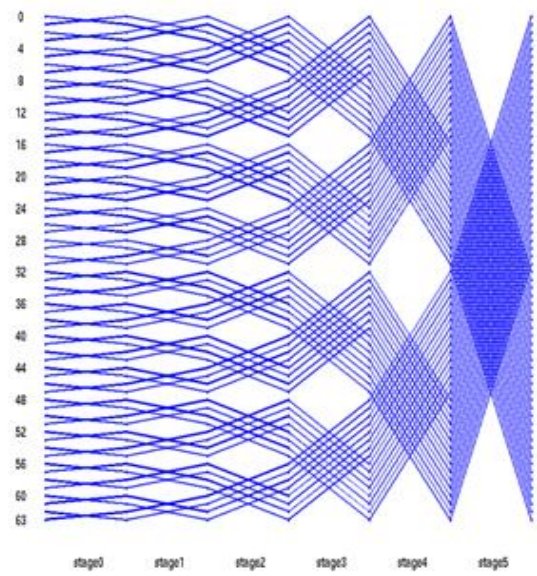


Fig. 2: 64-point radix-2 DITFFT butterfly diagram

The above structure having six stages, each stage include complex computations and code has to be develop for each stage and it can be simulated by using modelsim simulator. The synthesis report of 64-point FFT is depicted as below which gives information about timing analysis, slices used etc.

I Synthesis report of radix-2 64-point DITFFT

LOGIC UTILIZATION	USED	AVAILABLE	UTILIZATION
Number of Slices	2388	4656	51%
Number of 4 input LUTs	4282	9312	45%
Number of bonded IOBs	135	232	58%
Number of slices FFs	235	9312	2%
Number of GCLKs	3	24	12%
Minimum Delay	**75.050ns** (44.212ns logic, 30.838ns route)(58.9% logic, 41.1% route)		
Total memory usage	**206720** kilobytes		

III. RADIX-4 ALGORITHM

Radix-4 is another FFT algorithm which was surveyed to improve the speed of functioning by reducing the computation; this can be obtained by change the base to 4. For a same number if base increases the power/index will reduce. For the above project the number of stages are reduced to 50% since $N=4^3$ ($N=4^M$) i.e. only 3 stages. Radix-4 having four inputs and four outputs and its follows in-place algorithm. The following will explains the functioning of radix-4 and how the computational complexity is reduced.

A. FUNCTIONING OF RADIX-4 ALGORITHM

The DIT radix-4 FFT recursively partitions a DFT into four quarter-length DFTs of groups of every fourth time sample. The outputs of these shorter FFTs are reused to compute many outputs, thus greatly reducing the total computational cost. The radix-4 decimation-in-frequency FFT groups every fourth output sample into shorter-length DFTs to save computations. The radix-4 FFTs require only 75% as many complex multiplications as the radix-2 FFTs.

Let us begin by describing a radix-4 DITFFT algorithm briefly. The radix-4 decimation-in-time and decimation-in-time fast Fourier transforms (FFTs) gain their speed by reusing the results of smaller, intermediate computations to compute multiple DFT frequency outputs. The radix-4 decimation-in-time algorithm rearranges the DFT equation into four parts: sums over all groups of every fourth discrete-time index n = [0, 4, 8 ... N - 4], n = [1, 5, 9 ... N - 3], n = [2, 6, 10 ... N - 2] and n = [3, 7, 11 ... N - 1], (This works out only when the FFT length is a multiple of four.) Just as in the radix-2 DITFFT, further mathematical manipulation shows that the length-N DFT can be computed as the sum of the outputs of four length-N/4 DFTs, of the even-indexed and odd-indexed discrete-time samples, respectively, where three of them are multiplied by so-called twiddle factors $W^k{}_N = e^{-(i\,2\pi k/N)}$, $W^{2k}{}_N$, and $W^{3k}{}_N$.

We split or decimate the *N*-point input sequence into four subsequences, $x(4n)$, $x(4n+1)$, $x(4n+2)$, $x(4n+3)$, $n = 0, 1, \ldots, N/4-1$.

$$X(k) = \sum_{n=0}^{N-1} x(n)exp(-j2\pi nk/N) = \sum_{n=0}^{\frac{N}{4}-1} x(4n)exp(-j2\pi(4n)k/N) + \sum_{n=0}^{\frac{N}{4}-1} x(4n+1)exp(-j2\pi(4n+1)k/N) + \sum_{n=0}^{\frac{N}{4}-1} x(4n+2)exp(-j2\pi(4n+2)k/N) + \sum_{n=0}^{\frac{N}{4}-1} x(4n+3)exp(-j2\pi(4n+3)k/N) \quad (1)$$

$$= DFT_{N/4}[x(4n)] + W^k{}_N\, DFT_{N/4}[x(4n+1)] + W^{2k}{}_N\, DFT_{N/4}[x(4n+2)] + W^{3k}{}_N\, DFT_{N/4}[x(4n+3)] \quad (2)$$

This is called decimation in time because the time samples are rearranged in alternating groups and a radix-4 algorithm because there are four groups refer to "(1), (2)". The basic operation of R4 butterfly is shown in the following figure.

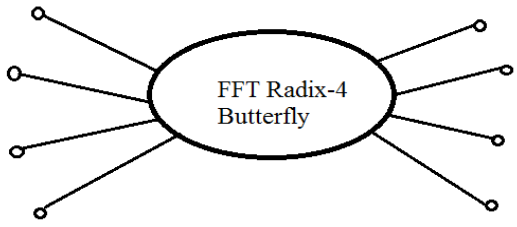


Fig. 3: Basic structure of R4 FFT

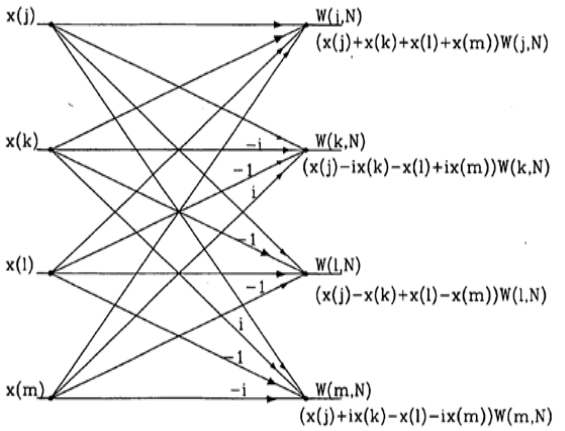


Fig. 4: Basic radix-4 butterfly operation

Fig. 5 depicts a 64-point radix-4 FFT using the butterfly symbol shown in Fig. 3 and Fig. 4 to represent the mathematical operations.

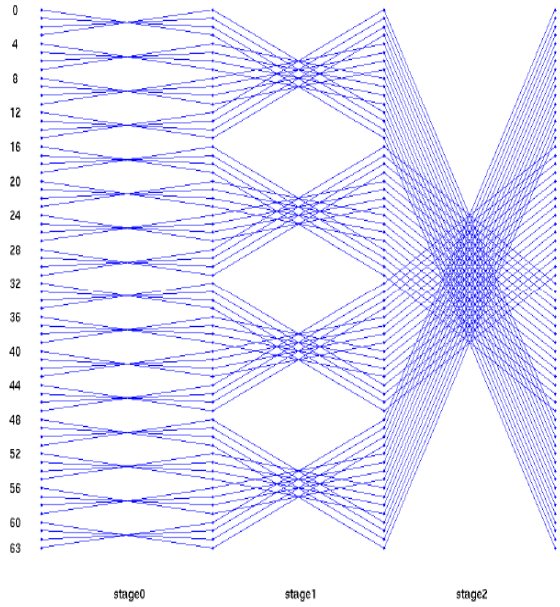


Fig. 5: 64-point radix-4 DITFFT butterfly diagram

The above diagram represents 64-point radix-4 DITFFT butterfly diagram in which we have only 3 stages, the internal RTL views and internal components are as shown below:

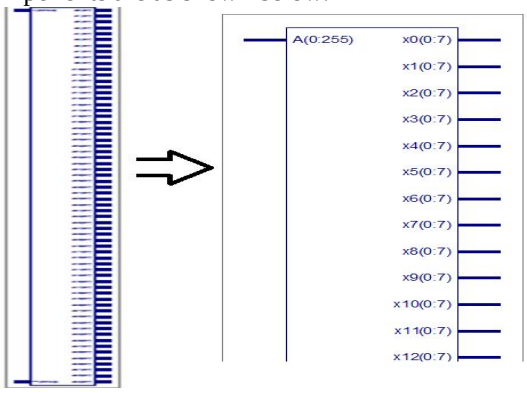


Fig. 6: RTL view of 256-bits 64-point radix-4 DITFFT

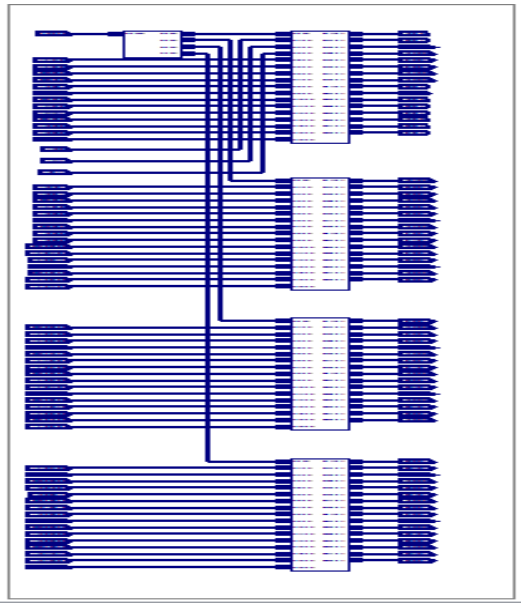


Fig. 7: Internal diagram of RTL view

B. SYNTHESIS AND SIMULATION RESULTS OF RADIX-4 ALGORITHM:

The next are shown the simulation results of the 256-bits 64-points radix-4 DITFFT butterfly block. The input A of 64-points each point of 4-bits totally 256-bits, input twiddle factor (W) of 256 points. The X is an output of 64-points each point of 8-bits since even and odd part results, totally 512 points all are in binary formats.

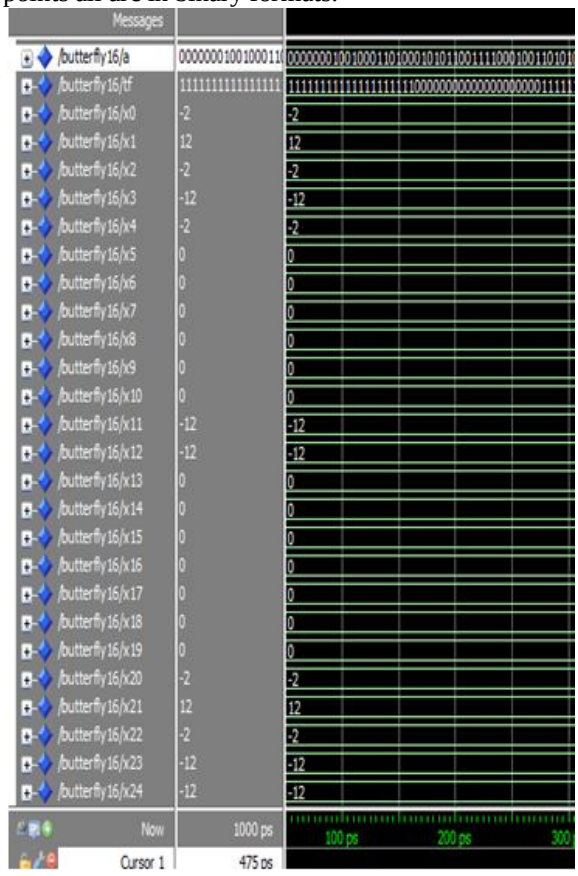


Fig. 8 (Cont…)

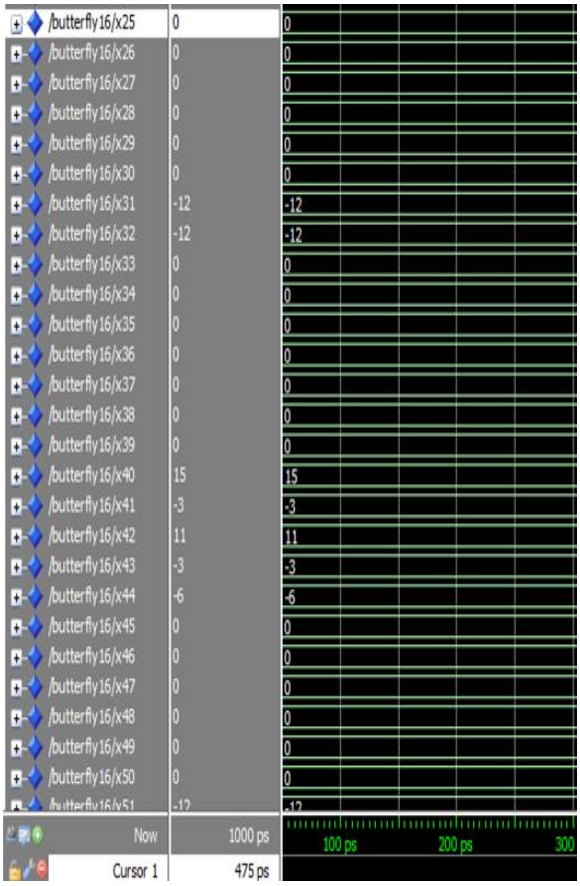


Fig. 8: (Cont…)

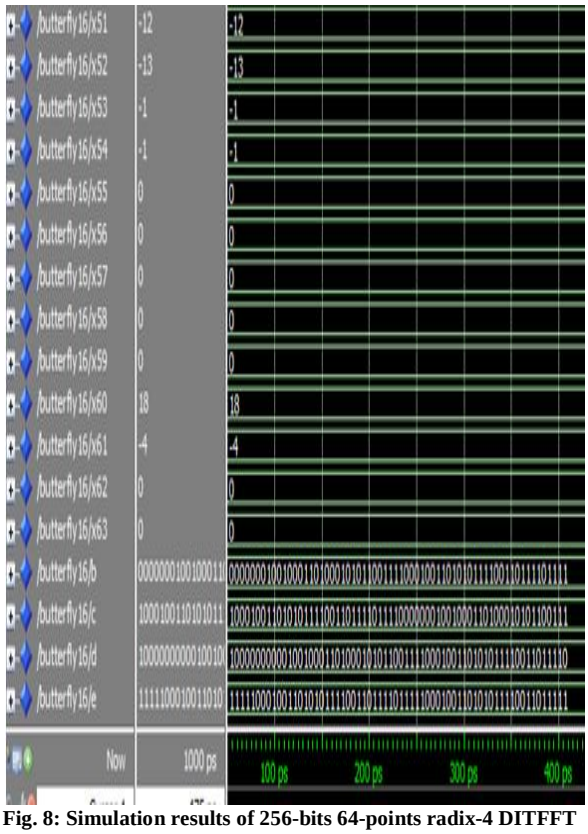


Fig. 8: Simulation results of 256-bits 64-points radix-4 DITFFT

II Device utilization summary and time delay of 256-bits 64-points radix-4 DITFFT

LOGIC UTILIZATION	USED	AVAILABLE	UTILIZATION	RESULTS
Number of Slices	3332	4656	71%	
Number of 4 input LUTs	5936	9312	63%	
Number of bonded IOBs	1024	232	441%	Resource overused
Number of MULT18X18SIOs	16	20	80%	
Minimum delay	18.963ns (11.549ns logic, 7.414ns route) (60.9% logic, 39.1% route)			
Total memory usage	228928 kilobytes			

By comparing the table I and II it is proved that radix-4 saves approximately 75% computations when compared with the radix-2 algorithm.

CONCLUSION

This project presents the new high speed FFT architecture based on radix-4 algorithm. The pipelined 256-bits 64-points radix-4 DITFFT can be implemented easily by using both FPGA and standard cell technologies such portability is offered by this algorithm. From the above synthesis and simulation results of radix-2 64-points and radix-4 64-points it is understandable radix-4 having less delay in processing the input when compared with radix-2. By considering the time delay table it is analyzable around 75% of time is saved. The delay time is reduced then the fastness of the system increased.

REFERENCES

[1] Asmitha Haveliya, Amity University Lucknow, India "Design And Simulation Of 32-Point FFT Using Radix-2 Algorithm For FPGA Implementation" 2012 Second International Conference on Advanced Computing & Communication Technologies.

[2] Douglas L. Jones "Radix-4 FFT Algorithms" Connexions module: m12027

[3] "Discrete Fourier Transform", Dmitriy Shutin, dshutin@tugraz.at Klaus Witrisal, witrisal@tugraz.at, ErhardRank, rank@tugraz.at Mari´an K´epesi, kepesi@tugraz.at PaulMeissner, paul.meissner@tugraz.at. Signal Processing and Speech Communication Laboratory, http://www.spsc.tugraz.at/ Inffeldgasse 16c/EG

[4] J. Rabaey,A. Chandrakasan, and B. Nikolic, Digital Integrated Circuits, A Design Perspective. Prentice-Hall, 2003.

[5] K. Parhi, VLSI Digital Signal Processing Systems. New York, NY, USA: John Wiley & Sons, 1999.

[6] Transmeta Corporation, "Transmeta Long Run Power Management," http://www.transmeta.com.

[7] J. G. Proakis and D. G. Manolakis, Digital Signal Processing. Prentice-Hall, 1996.

[8] J. W. Cooley and J. W. Tukey, "An Algorithm for Machine Calculation of Complex Fourier Series," Math. Comput., vol. 19, pp. 297–301, Apr. 1965.

[9] W. Li and L. Wanhammar, "A Pipelined FFT Processor," in IEEE Workshop on Signal Processing Systems, 1999, pp. 654–662.

[10] S. Johansson, S. He, and P. Nilsson, "Wordlength Optimization of a Pipelined FFT Processor," in Proc. of 42nd Midwest Symposium on Circuits and Systems, Las Cruces, NM, USA, Aug. 8-11 1999.