FPGA IMPLEMENTATION OF A LOW POWER 64-POINT FFT ARCHITECTURE USING MODIFIED RADIX-4 ALGORITHM

E. KONGUVEL, M. KANNAN, J. MADHUMITHA
Department of Electronics Engineering, Anna University - MIT Campus, Chennai - 600044, India.
konguart08@gmail.com, mkannan@annauniv.edu, sanjujsms@gmail.com

Abstract *– This paper presents an area efficient multiplier less parallel radix 4 Decimation in Frequency – Fast Fourier Transform (DIF-FFT) processor using Distributed Arithmetic Algorithm (DAA). Many numbers of complex computations involved in the radix-2 algorithm have driven to the usage of higher radix algorithms such as radix 4 and radix 8. DAA is an appropriate technique for eliminating the complex multiplications in FFT/IFFT processor by using look-up tables (LUTs) and shift-accumulators. Higher radix FFT algorithm along with DAA provides low area high-speed FFT/IFFT processor. The proposed multiplier less 64 point radix 4 FFT/IFFT processor using DAA algorithm is designed, implemented and simulated in Altera DE2 EP2C35F672C6 device. Hardware implementation of the proposed DAA based 64 point FFT processor has a total of 2.77% FPGA utilization which is 36.62% lesser than conventional FFT/IFFT processors and it operates at 46.30Gbps.*

Keywords – FFT, IFFT, Distributed Arithmetic, Multiplier-less, Radix 4.

1. Introduction

The Fast Fourier Transform (FFT) and its Inverse (IFFT) plays a vital role in signal processing applications. A number of FFT/IFFT algorithms have been proposed such as Cooley-Tukey algorithm, Prime factor algorithm, Bruun's algorithm, Rader's algorithm and Bluestein's algorithm. Cooley-Tukey algorithm is chosen in this research article for it's straightforward and efficient design and implementation of the basic butterfly unit [1]. FFT can be performed through radix-2, radix-4, radix-8 and other higher order radices too. For low speed applications, radix-2 processor can be used. For the high-speed processor, higher order radices are required [2]. Even though the number of computations in higher radix algorithms is reduced, the internal complexity is so higher that becomes difficult for implementation and debug. Therefore, radix-4 butterfly structure is utilized, since it provides a greater trade-off between the operational speed and design complexity [3].

FFT/IFFT computations can be performed using sequential, parallel or pipelined architecture. Sequential architecture involves round implementing the butterfly structures one after the other [4]. Therefore, sequential architecture increases the delay in processing the input data bits and thus results in the reduction of speed. In pipeline algorithms, each and every operation is processed simultaneously [5]. The FFT/IFFT computation process includes fetching complex input data points, and butterfly computations are performed after which Distributed Arithmetic based Look-Up-Table (DA-LUT) is used for complex multiplications, and then the complex output data is processed [6]. Pipeline architecture has two primary disadvantages. First, a non-pipelined or a parallel processing unit executes the instruction at an instance. This technique prevents branch delays and does not cause either any pipeline stalls. Consequently, the non-pipelined design is simpler and cheaper to manufacture. Second, the instructional latency in a parallel processor is lower than pipelined mechanism, since it requires additional flip flops for its intermediary data paths. Another type of architecture is parallel processors are in which more than one processors are running in parallel [7]. A dedicated hardware will be used for the specific tasks. Hence, parallel processors compensate both the speed and pipeline delays.

The primary shortcoming of the conventional FFT/IFFT algorithm is the requirement of much more hardware components to perform the complex multiplications involved. The Distributed Arithmetic Algorithm (DAA) is to carry out the same complex multiplications in a distributed approach through addition, subtraction and shift-accumulation process. DAA can be implemented by various techniques such as look-up table (LUT) based DA, offset binary DA, exploit the symmetric property and Hybrid DA [8] which practices carry save adder for computation. Parallel look-up table based distributed arithmetic (DA-LUT) architecture is proposed for its efficient managing of latency and area [9]. The binary scaling technique is incorporated along with DAA for floating point calculation of twiddle factors to maximize the performance of the FFT/IFFT processing element [10]. Implementation of

Maximum Power Point Tracking (MPPT) using fuzzy logic on Xilinx Virtex - II Field Programmable Gate Array (FPGA) for photovoltaic applications are discussed in [11]. A comparative study of efficient FFT/IFFT algorithms and architectures for Multiple Input Multiple Output - Orthogonal Frequency Division Multiplexing (MIMO-OFDM) based applications were presented in [12]. Prototype of radix-2 and radix-4 optimized butterflies are implemented in Altera Quartus - II FPGA device were presented in [13].

This research article is organized as follows. After this introduction, section 2 deals about the proposed Distributed Arithmetic Algorithm (DAA) based radix-4 64-point FFT/IFFT processor. Section 3 deals with the FPGA implementation and simulation results of the proposed radix-4 64-point parallel FFT computations as well as the performance analysis and comparison with conventional and modified booth multiplier based FFT/IFFT methodologies. Quite a few concluding observations are acknowledged in section 4.

2. Distributed Arithmetic based FFT architecture

2.1 Modified Distributed Arithmetic Algorithm (DAA)

The proposed method presents a parallel FFT/IFFT processor using DAA look-up table implementation. The architectural flow of the proposed FFT/IFFT processor is shown in fig 1. Distributed arithmetic algorithm (DAA) comprises of three essential components, namely shift register, accumulator and look-up table.

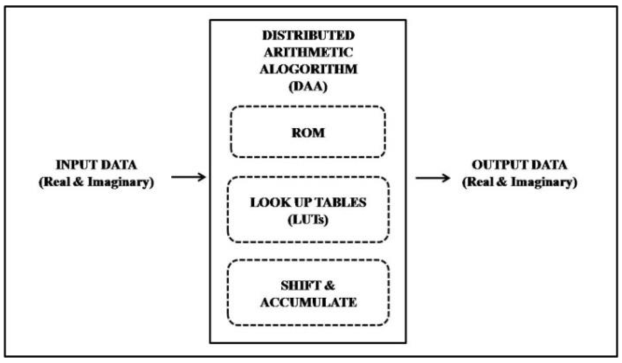
DISTRIBUTED ARITHMETIC ALOGORITHM (DAA)
ROM
INPUT DATA (Real & Imaginary)
LOOK UP TABLES (LUTs)
OUTPUT DATA (Real & Imaginary)
SHIFT & ACCUMULATE

Fig. 1. Multipliers replaced by LUTs & Shift Accumulator

The complex input data is given to the processing element and multiplied with twiddle factor (retrieved from the look-up table), which carries the pre-computed partial products. The twiddle factor values are stored in a Read Only Memory (ROM). The obtained complex output is then applied to a complex parallel adder, and the same is stored in an accumulator-register. The scaled value of accumulator output is the second complex input to the adder. The course of multiplication is being replaced by the successive of retrieval data from the look-up table, addition, and shift accumulation.

The Distributed Arithmetic Algorithm (DAA) can be formulated by assuming multiply and accumulate operation that can be represented mathematically as,

$$y = \sum_{k=1}^{K} A_k x_k \tag{1}$$

Where A_k is a constant and x_k is N-bit scaled two's complement number,

$$x_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} \tag{2}$$

Where b_{k0} is the sign bit.
On substituting equation (2) in equation (1),

$$y = -\sum_{k=1}^{K} (b_{k0} \bullet A_k) + \sum_{k=1}^{K}\sum_{n=1}^{N-1} (A_k \bullet b_{kn}) 2^{-n} \tag{3}$$

By re-arranging equation (3),

$$y = \sum_{k=1}^{K} A_k \bullet (-b_{k0}) + \sum_{n=1}^{N-1}\left[\sum_{k=1}^{K} A_k \bullet b_{kn}\right] 2^{-n} \tag{4}$$

From equation (4), it is evident that each of the components $\sum_{k=1}^{K} A_k(-b_{k0})$ and $\left[\sum_{k=1}^{K} A_k b_{kn}\right]$ are to be stored in the look-up table to replace the multiplication with successive addition and shift accumulation. It is also apparent that each term requires 2^K vales. Therefore, 2 * 2^K look up tables sizes are needed to accumulate the data from two multiplication components in equation (4).

2.2 Proposed Radix 4 64 point FFT architecture

In radix-4 decomposition, the four complex input data points are processed at a time. For single radix FFTs, the transform size must be a power of the radix. The FFT length is 4^N, where N is the number of stages and 4 represents the appropriate radix decomposition. The DAA based radix-4 butterfly algorithm is shown in figure 2.
The output of the each term in radix 4 algorithm can be represented mathematically as,

$$X(4k) = \sum_{n=0}^{N/4-1}\left[x(n) + x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) + x\left(n+\frac{3N}{4}\right)\right] W_N^0 W_{N/4}^{kn} \tag{5}$$

$$X(4k+1) = \sum_{n=0}^{N/4-1}\left[x(n) - jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) + jx\left(n+\frac{3N}{4}\right)\right] W_N^n W_{N/4}^{kn} \tag{6}$$

$$X(4k+2)=\sum_{n=0}^{N/4-1}\left[x(n)-x\left(n+\frac{N}{4}\right)+x\left(n+\frac{N}{2}\right)-x\left(n+\frac{3N}{4}\right)\right]W_N^{2n}W_{N/4}^{kn} \quad (7)$$

$$X(4k+3)=\sum_{n=0}^{N/4-1}\left[x(n)+jx\left(n+\frac{N}{4}\right)-x\left(n+\frac{N}{2}\right)-jx\left(n+\frac{3N}{4}\right)\right]W_N^{3n}W_{N/4}^{kn} \quad (8)$$

Where $W_N^0 W_{N/4}^{kn}$ to $W_N^{3n} W_{N/4}^{kn}$ in equation (5) to (8) are the twiddle factors of the FFT computation whose values are known. These twiddle factors are to be multiplied with the respective complex input data points to form the partial products.

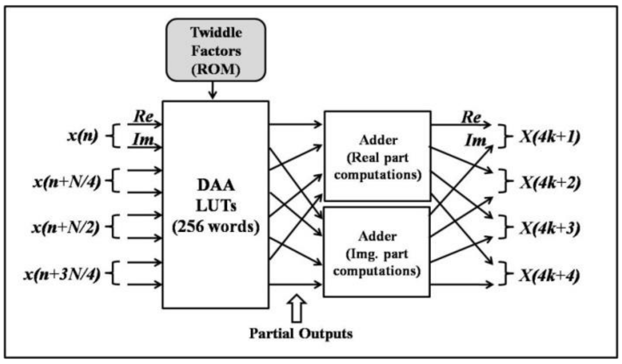

Fig. 2. Proposed DAA based radix-4 butterfly.

Since the first multiplicand values (twiddle factor) are known, all the products for possible second multiplicand values (both real and imaginary input data points) are stored in the LUTs as pre-computed values. When the complex input is fetched from the input buffer, the real and imaginary values are mapped with the LUTs; respective pre-computed values are given as inputs for the complex adders.

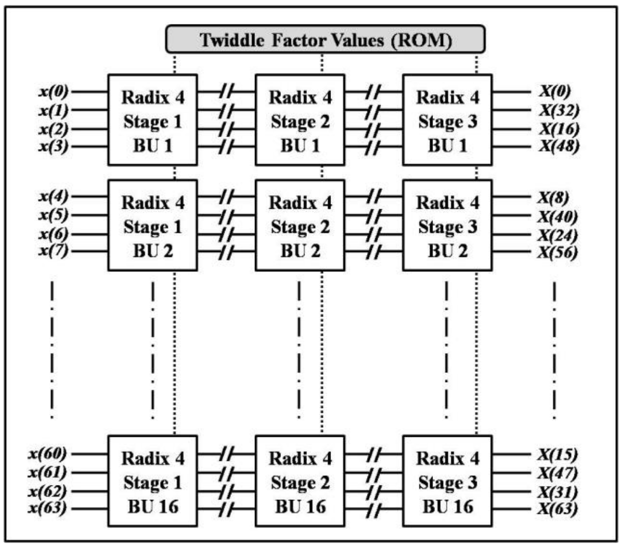

Fig. 3. Radix 4 64 point FFT architecture

By using Distributed Arithmetic Algorithm, pre-calculated partial products are accumulated in the LUTs, which eliminate the complex multipliers. Since no multipliers are used, that reduces the area occupied the multiplying units as well as it reduces the power usage. This look-up table (LUT) based multiplier less radix-4 butterfly is used in the DIF based flow diagram for 64 points FFT architecture which is shown in figure 3. Radix 4 64 point FFT flow requires three stages with 16 butterfly units each which are represented as BU1 to BU 16 for all the stages. The twiddle factor values are given to the respective LUTs in the butterflies employing a ROM.

3. Experimental Results

The proposed distributed arithmetic algorithm based radix-4 64 point FFT processor is designed, implemented and simulated in Altera Quartus II DE2 EP2C35F672C6 device. The experimental results that are obtained from the simulations are discussed in detail below.

3.1 Synthesis Report

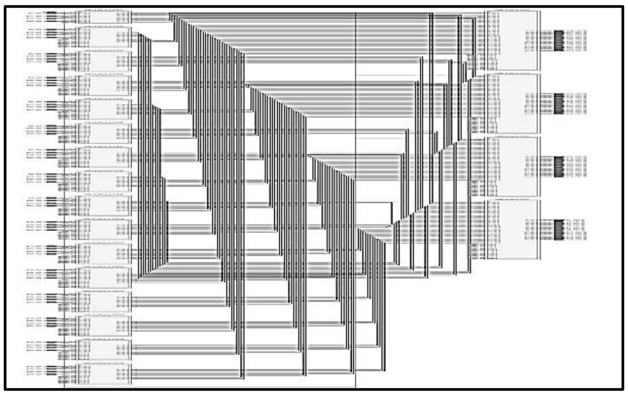

Fig. 4. RTL schematic of proposed FFT processor

The RTL of the proposed distributed arithmetic algorithm based radix-4 64 point FFT processor is shown in figure 4. The proposed system achieves low utilization of about 2.77% which includes 770 registers and 2778 combinational slices.

3.2 Timing and Power Analysis

The proposed distributed arithmetic based 64 point FFT algorithm achieves a maximum clock set up time of 5.479nS, worst case propagation delay of 10.516nS, combinational delay of 8.802nS at a clock speed of 182.52MHz. The proposed system has a core static power dissipation of 80.10mW and IO thermal power dissipation of 83.16mW under the final models of EP2C35F672C6 FPGA device for 64 points FFT. The proposed system has a core static power dissipation of 70.24mW and IO thermal power dissipation of 74.36mW under the final models of EP2C35F672C6 FPGA device for 16 points FFT.

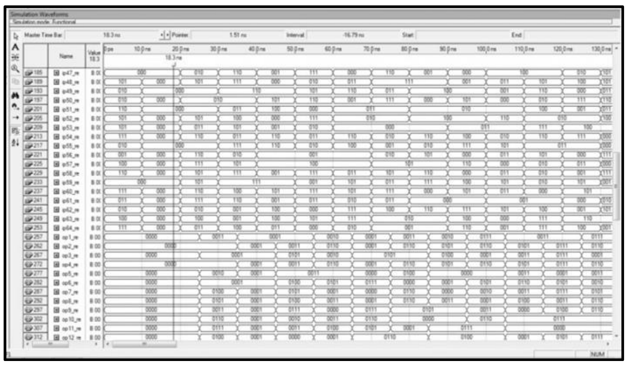
Fig. 5. Simulation Output

3.3 Simulation Output

The input samples are specified for a period cycle, and the outputs are generated at the second positive triggered edge**.** The clock is given to the design as input through the pin *clk*, the complex input data points are given through *ip1_re* to *ip64_re* for the real part and *ip1_im* to *ip64_im* for the imaginary part. The generated complex output data points can be viewed from *op1_re* to *op64_re* and *op1_im* to *op64_im*. The simulation waveforms for the proposed system are shown in figure 5. The chip planner view is shown in figure 6 which shows the utilization area, fan-ins, and fan-outs of the proposed system.

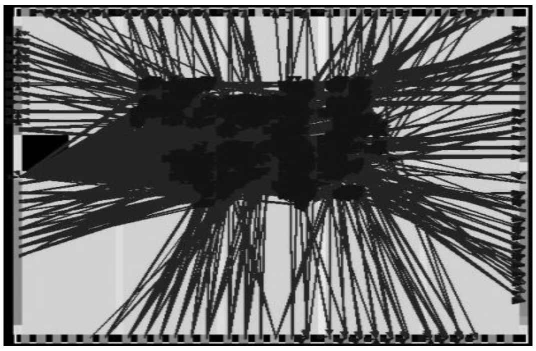
Fig. 6. Chip planner view

3.4 Performance analysis and comparison

The performance comparisons of conventional FFT, modified Booth multiplier based FFT and proposed DAA based FFT are presented in Table 1. In the conventional method, both radix-2, radix-4 based FFT implementation results and in modified Booth multiplier based FFT and proposed DAA based FFT, radix 4 16 and 64 points are presented for the comparative analysis.

The proposed FFT processor utilizes 1.52% of registers and 3.98% of combinational slices which is lesser than the conventional and modified Booth multiplier based methods. The setup time for the proposed system is lesser than the other two methods since its area is reduced which is shown in figure 7 for radix 4 16 point FFT architecture. Similarly, the operating frequency of the proposed system is 222.60MHz for 16-point FFT is higher than the operating frequencies of the other two given systems. From figure 8, it is clear that throughput of the proposed system is improved by 4.18Gbps than the conventional FFT and 3.06Gbps than modified Booth multiplier based FFT method. It is acknowledged in figure 8; the power consumption of multiplier less radix 4 64 point FFT architecture is lesser than the conventional and modified Booth multiplier based FFT architectures.

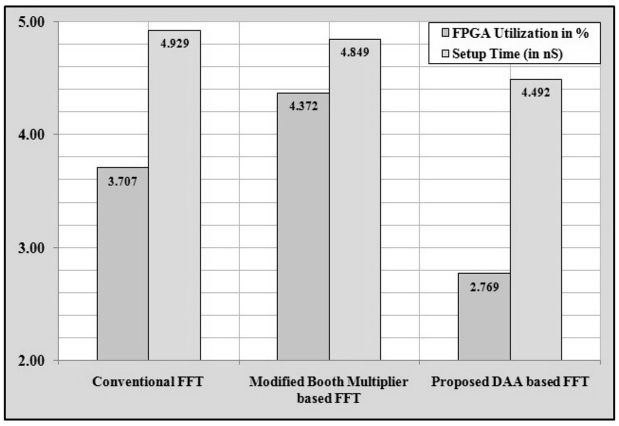

Fig.7: FPGA utilization and setup time comparison

4. Conclusion

In this paper, we have presented FPGA implementation of multiplier less radix-4 64-point FFT/IFFT processor which achieves a maximum throughput of 68Gbps is presented. The multiplier less design is attained by the usage of modified Distributed Arithmetic Algorithm (DAA). The circuit is designed, implemented and simulated in Altera Quartus II DE2 EP2C35F672C6 FPGA device. Performance comparisons show that the proposed method is balanced in parameters such as area, power, and throughput. The future work includes the application of modified Distributed Arithmetic Algorithm (DAA) for higher data point FFT/IFFT algorithms which make the system more suitable for MIMO-OFDM broadband based applications such as WiMax or LTE.

Table 1: Performance comparisons of different FFT processors.

Parameter	**Conventional FFT**		**Modified Booth based FFT**	**Proposed DAA based FFT**	
Radix/FFT Point	2/16	4/16	4/16	4/16	4/64
Total Pins (475)	449 (94.52%)	449 (94.52%)	449 (94.52%)	449 (94.52%)	463 (97.47%)
Flip-flops (33216)	1040 (3.13%)	553 (1.66%)	559 (1.68%)	507 (1.52%)	770 (2.31%)
Combinational Elements (33216)	2044 (6.15%)	1910 (5.75%)	2346 (7.06%)	1333 (3.98%)	2778 (8.36%)
Frequency (MHz)	139.68	202.88	206.23	222.60	182.52
Clock Setup Time (nS)	7.159	4.929	4.849	4.492	5.478
Power (mW)	155.79	151.55	154.36	144.60	163.26
Throughput (Gbps)	44.50	64.26	65.38	68.44	46.30

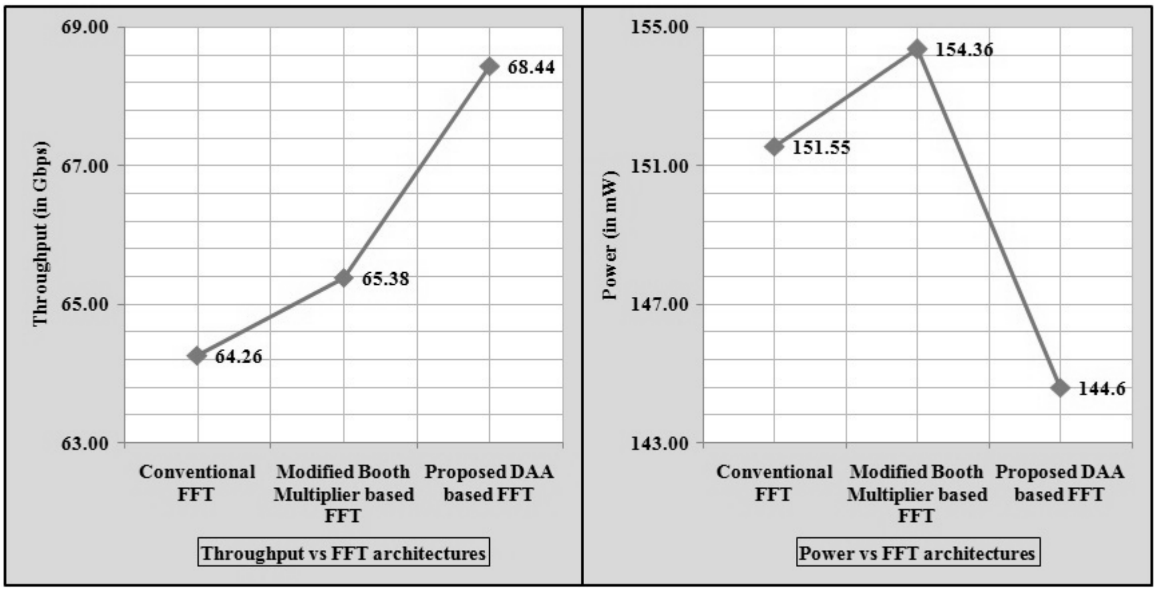

Fig. 8: Throughput and Power vs. different FFT processors

5. References

[1]. J. W. Cooley and J. Tukey, "An algorithm for machine calculation of complex Fourier series," Math. Comput., vol. 19, pp. 297–301, Apr. 1965.

[2]. H. S. Kang, S. H. Chang, I. K. Hwang and J. K. Lee. (2016). A design and implementation of 32-paths parallel 256-point FFT/IFFT for optical OFDM systems. *18th International Conference on Advanced Communication Technology*, South Korea, 1-1.

[3]. R. Neuenfeld, M. Fonseca and E. Costa. (2016). Design of optimized radix-2 and radix-4 butterflies from FFT with decimation in time. *IEEE 7th Latin American Symposium on Circuits & Systems*, Florianopolis, 171-174.

[4]. P. A. Sophy, R. Srinivasan, J. Raja and M. Avinash. (2015). Analysis and design of low power radix-4 FFT processor using pipelined architecture. *International Conference on Computing and Communications Technologies,* Chennai, 227-232.

[5]. S. Singh & J. Kedia. (2015). Pipelined FFT architectures: A review. *International Conference on Electrical, Electronics, Signals, Communication and Optimization*, Visakhapatnam, 1-5.

[6]. Sunil P. Joshi & Roy Paily. (2014). Distributed Arithmetic based Split-Radix FFT, *Journal of Signal Processing Systems,* 2014(75), 85-92.

[7]. Eleanor Chu & Alan George (2000). Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms. Florida, CRC press LCC.

[8]. Nagakishore Bhavanam. S, Keerthi. M, Vasujadevi Midasala & JeevanReddy. K. (2012). FPGA Implementation of Distributed Arithmetic For FIR Filter. *International Journal of Engineering Research & Technology*, 1(9), 1-8.

[9]. Nisha Laguri & N. Anusudha. (2014). VLSI implementation of efficient split radix FFT based on distributed arithmetic, *International Conference on Green Computing, Communication and Electrical Engineering,* Coimbatore, 1-5.

[10]. Augusta Sophy, R.Srinivasan & J.Raja, (2014) Low power reconfigurable FP-FFT core with an array of folded DA butterflies. *Eurasip Journal of Advances in Signal Processing*, 2014(144).

[11]. F.Chekired, C.Larbes, D.Rekioua & F. Haddad (2011) Implementation of a MPPT fuzzy controller for photovoltaic systems on FPGA circuit, *Energy Procedia*, 6, 541 – 549.

[12]. E. Konguvel & M. Kannan, (2018), A Survey on FFT/IFFT Processors for Next Generation Telecommunication Systems, *Journal of Circuits, Systems and Computers,* 27 (3), 1830001.

[13]. B. R. Manuel, E. Konguvel and M. Kannan, 2017, An area efficient high speed optimized FFT algorithm, *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, Chennai, pp. 1-5.