International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Email:editorijoer@gmail.com http://www.ijoer.in

Vol.4., Issue.3., 2016
(May-June)

RESEARCH ARTICLE

ISSN: 2321-7758

FPGA IMPLEMENTATION OF FFT PROCESSOR USING DIFFERENT ALGORITHMS

YOJANA A. JADHAV[1], Prof. A. P. HATKAR[2]
[1]Student, Dept. of E & TC, SVIT COE, Nasik, India
[2]Prof, Dept. of E & TC, SVIT COE, Nasik, India

ABSTRACT

The Fast Fourier Transform (FFT) is an efficient algorithm for computing the Discrete Fourier Transform (DFT) and requires less number of computations than that of direct evaluation of DFT. It has several applications in signal processing. Because of the complexity of the processing algorithm of FFT, recently various FFT algorithms have been proposed to meet real-time processing requirements and to reduce hardware complexity over the last decades. This is in two directions. One related to the algorithmic point of view and the other based on ASIC architecture. The last one was pushed by VLSI technology evolution. In this work, we present three different architectures of FFT processor to perform 1024 point FFT analysis and also compared these three architectures with another radix-4 architecture of FFT processor that is using vedic mathematics. The designs have been simulated and its FPGA based implementation has been verified successfully using Xilinx ISE 11.1 tool using VHDL. There are also comparative studies among those architectures. The objective of this work was to get an area & time efficient architecture that could be used as a coprocessor with built in all resources necessary for an embedded DSP application.

Keywords- Fast Fourier Transform, FFT butterfly radix 2 & 4, CORDIC, Sine-Cosine lookup table, Vedic algorithm, DSP, Urdhava Tiryakbhyam, Xilinx Core.

I. INTRODUCTION

Audio and communications signal processing are well developed lines massively used now a days in many application lines and products. Since digital communications are quite active fields, the arithmetic complexity of the Discrete Fourier Transform (DFT) algorithm becomes a significant factor with impact in global computational costs. Cooley and Tukey [1] developed the well-known radix-2 Fast Fourier Transform (FFT) algorithm to reduce the computational load of the DFT. The Discrete Fourier Transform (DFT) X(k) of N points is given by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad \{0 \le k < N-1\},\ W_N^{nk} = \exp(-j2\pi nk/N) \tag{1}$$

Where the X(k) and x(n) are frequency-domain sequences and time-domain sequence. Instead of the direct implementation of the equation (1), the FFT algorithm factorizes a large point DFT recursively into many small point DFT in order to reduce the overall operations. There are two well-known types of decompositions called Decimation in Time (DIT) and Decimation In Frequency (DIF) FFT. The only difference between these two algorithms is that, DIT starts with bit reverse order input and generates normal order output. Nevertheless DIF starts with normal order input and generates bit reverse order output. Throughout this paper DIF algorithm is used. The conventional method of Fast Fourier Transform FFT calculation involves N2complex multiplications and N(N-1) complex additions. The radix-2 Cooley-Tukey algorithm performs the same computation involving (N/2)log2N complex multiplications and (N)log2N complex additions. But it is more efficient computationally to employ a radix-4 FFT algorithm other than radix -2 logarithms.

Note that the input to each N/4-pointDFT is a linear combination of four signal samples scaled by a twiddle factor. This procedure is repeated v times, where v = log4N.

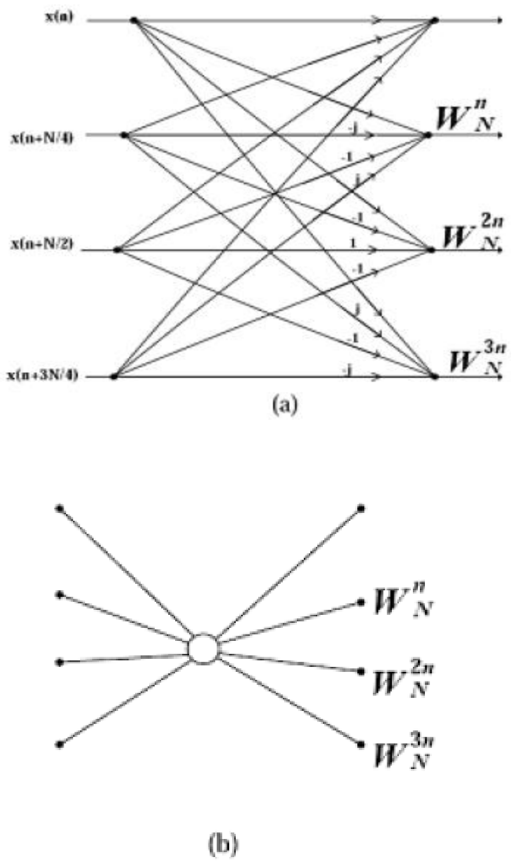


Figure 1: The basic butterfly for radix-4 DIF FFT algorithm

The complete butterfly operation for Radix-4 DIF is shown in figure 1 (a) and in a more compact form in figure 1(b) [2].

In this paper, we present radix-4 FFT processor using different architectures that allows any size points to transform, fixed point arithmetic, pipeline structure and parameterized data format. To improve the performance of such complex computation Vedic algorithm is adopted. The benefits of Vedic algorithm for efficient implementation of complex multiplier have been largely unexplored. In recent years, several designs of FFT using Vedic algorithm [3] have been proposed, but with the limited set of parameters.

VEDIC mathematics [5] is the ancient Indian system of mathematics which mainly deals with Vedic mathematical formulae and their application to various branches of mathematics. The word 'Vedic' is derived from the word 'Veda' which means the store-house of all knowledge. Vedic mathematics was reconstructed from the ancient Indian scriptures (Vedas) by Sri BharatiKrisnaTirtha (1884-1960) after his eight years of research on Vedas [5].This paper presents a simple digital multiplier architecture [5] based on the ancient Vedic mathematics Sutra (formula) called Urdhva-Tiryakbhyam(Vertically and Cross wise) Sutra which was traditionally used for decimal system in ancient India. In [4,6], this Sutra is shown to be a much more efficient multiplication algorithm as compared to the conventional counterparts. Urdhva-Tiryakbhyam Sutra [7] is first applied to the binary number system and is used to develop digital multiplier architecture. This Sutra also shows the effectiveness of reducing the N×N multiplier [7] structure into an efficient 4×4 multiplier structures. This work presents a systematic design methodology for fast and area efficient digital multiplier based on Vedic mathematics [7].

II. RELATED WORK

FFT processors are designed using four different architectures. In one of the architecture Twiddle factors are generated using CORDIC (Coordinate Rotation Digital Compute) algorithm, in another one through Sine/Cosine Look up table it is generated. Xilinx Logicore FFT processor is also used as useful architecture. And a simple digital multiplier architecture [4] based on the ancient Vedic mathematics Sutra (formula) called Urdhva Tiryakbhyam(Vertically and Cross wise) Sutra which was traditionally used for decimal system in ancient India. All are designed in FPGA through VHDL.

2.1 Design of FFT Processor using CORDIC algorithm

The design flow of FFT Processor using CORDIC is shown in figure 2.The selector block is nothing but a memory path buffer which compute respective memory of input samples. When Active signal is asserted and there are some input data, the address generator block assigns a memory position for each input sample. Now when Dual port Ram gets write Address signal from address generator block, it saves both memory path along with respective input samples. The 4 point FFT block has butterfly unit within it [2].

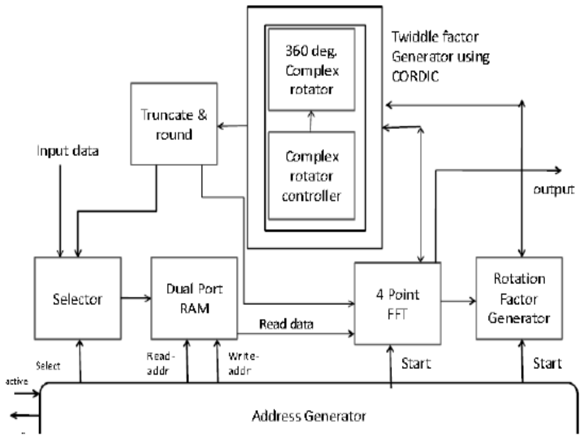


Figure 2: Architecture of FFT processor using Cordic

When a start signal is asserted, at the same time, both to 4 Point FFT and Rotation factor generator block, the FFT block sends a signal to CORDIC block for computing necessary twiddle factors consisting of sine-cosine terms. This block is controlled by Rotation factor generator block. In truncate & round block, remapping of memory path and twiddle factors are held and fed back to FFT block. Now when address generator block sends read address signal to DRAM, it sends stored input data samples along with memory path in FFT block. Finally this twiddle factors are applied to the output of the butterflies, and a bit reverse scramble is done. in the implementation of FFT, it is noticed that remapping of the memory is necessary. In the implementation of DIF the remapping is made from the exit of FFT. However that remapping can be made in a simple way. For instance, for the FFT radix-4 DIF, the entrance has to be written in the addresses of memory 0, 1, 2, 3, 4, 5, 6 and 7. After having processed a scrambling phases, it has to write in 0, 4, 2, 6, 1, 5, 3 and 7. That scrambling follows a much defined order. As 1024 point FFT processor is designed, the whole module of architecture is used for 5 times. The formula behind this is

Stage = log4 (computing point) (3)

2.2 Design of FFT Processor using Sine-Cosine lookup table algorithm

The design flow of FFT Processor using Sine-Cosine look up table is shown in figure 3. The design flow is quite similar with the processor designed in figure 3.Only the difference is that sine-cosine look up table is here used to compute the value of twiddle factors, which are previously stored in a RAM [2].

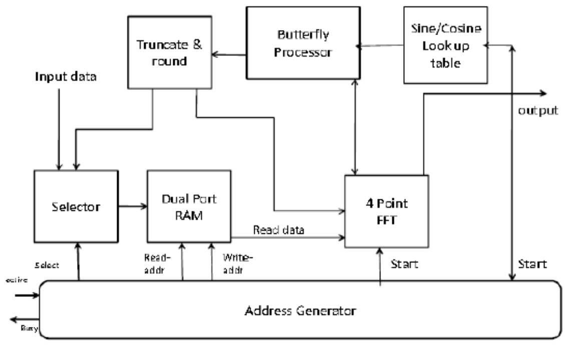


Figure 3: Architecture of FFT processor using Sine-Cosine look up table.

2.3 Design of FFT Processor using Xilinx Core

In this section Xilinx FFT core is used during to compute 1024 point FFT transform. The Xilinx FFT core offers a number of different architectures and also supports several arithmetic computations. The architecture of Core FFT is shown in figure 4.During the implementation of FFT Core many initialization was made. Such as Radix-4, Burst I/O architecture is used because in this solution the FFT core uses one radix-4 butterfly processing engine and has two processes. One process is loading and/or unloading the data. The second process is calculating the transform. Data I/O and processing are not simultaneous. When the FFT is started, the data is loaded in. After a full frame has been loaded, the core will compute the FFT. When the computation has finished, the data can now be unloaded. During the calculation process, data loading and unloading cannot take place. The data loading and unloading processes can be overlapped if the data is unloaded in digit reversed order [8].

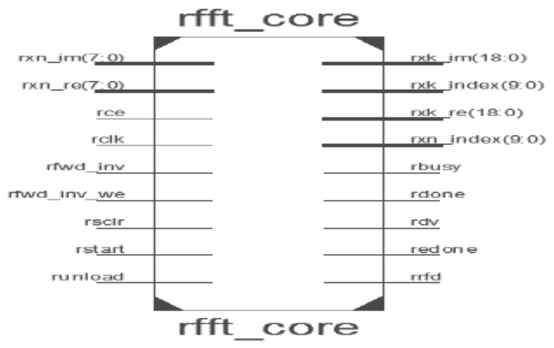


Figure 4: Architecture of Xilinx FFT core

The inputs are provided as 8-bits fixed-point data type. Coefficients are internally saved in the core and are also represented as 8-bit fixed-point data. We apply full-precision unscaled arithmetic, which takes into account the number of bit growth at each stage.In order to determine the necessary bits for correct representing the outputs, the core applied the formula [8]:

Output data width = input data width + log2 (transform length)+ 1 (4)

This approach will make sure that almost no data will be lost during the computation.

2.4 Design of FFT Processor using Vedic algorithm.

To perform 64-point FFT a single 4-point FFT unit is recursively used. This 4-point FFT is designed using high speed radix-4 algorithm which is shown in figure 5[4]. Moreover, the performance of FFT is limited by arithmetic operation such as complex multiplication. Complex multiplication of two numbers requires 4 multipliers, 2 adders and 1 subtractor or 3 multiplier and 5 adders. This large number of multipliers degrades the performance of FFT [3].

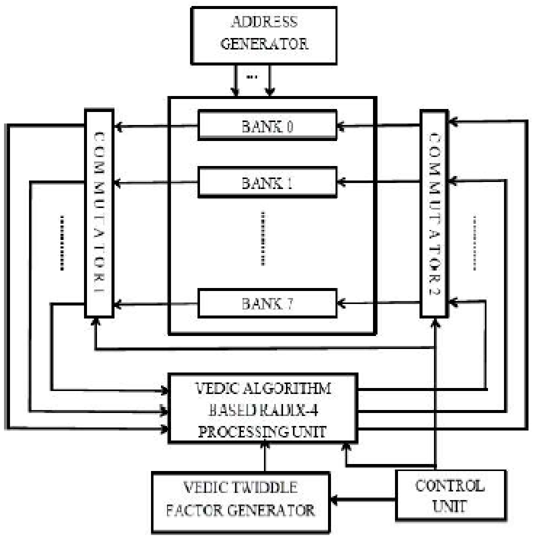

ADDRESS GENERATOR
BANK 0
BANK 1
BANK 7
COMMUTATOR 1
COMMUTATOR 2
VEDIC ALGORITHM BASED RADIX-4 PROCESSING UNIT
VEDIC TWIDDLE FACTOR GENERATOR
CONTROL UNIT

Figure 5: Architecture of 64-point FFT using Vedic algorithm.

Vedic algorithm is an ancient and well known technique for arithmetic operation. The method which is used for multiplications is 'Urdhvatiryagbhyam' which means vertical and crosswise. In this way, this algorithm performs multiplication of two given numbers in vertical and crosswise manner until left with only MSB bits. The proposed FFT utilizes Urdhvatiryagbhyam method of Vedic algorithm to perform complex twiddle factor multiplications.

III. RESULT AND DISCUSSION

We have simulated the four mentioned FFT processor architecture blocks using Xilinx Isim 11.1.In these concerned designs we have used fixed point format to truncate & round of the values.

3.1 Design Summary of 1024-point FFT processor

Design summary is a report which allows designer to view the information like targeted device, the number of errors and warning, device utilization & design goal. We have implemented our design in FPGA family Virtex4 (4vfx12ff668speed grade -12).Also the RTL schematic of the three mentioned FFT processor blocks are shown. These RTL schematics are basic logical representation of the circuit in terms of logic primitives which are generated when the design become correct in simulation and synthesis level. Figure 6 & figure 7 shows the RTL schematic &device utilization summary of 1024 point FFT processor using CORDIC algorithm [2]. Figure 8 shows the device utilization summary of 1024 point FFT processor using Sine-Cosine lookup table. Figure 9 shows the device utilization summary of 1024 point Xilinx Core FFT processor [2].

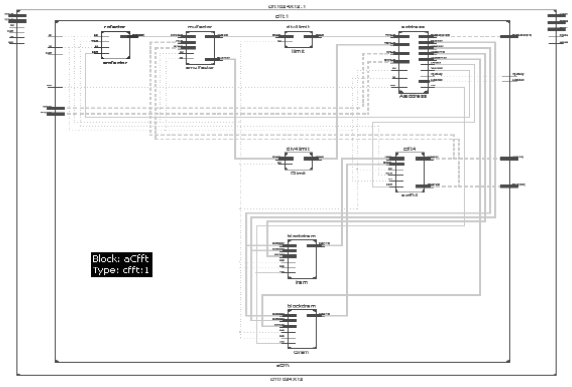
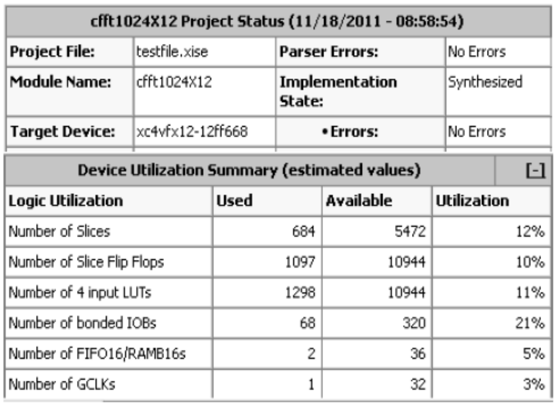


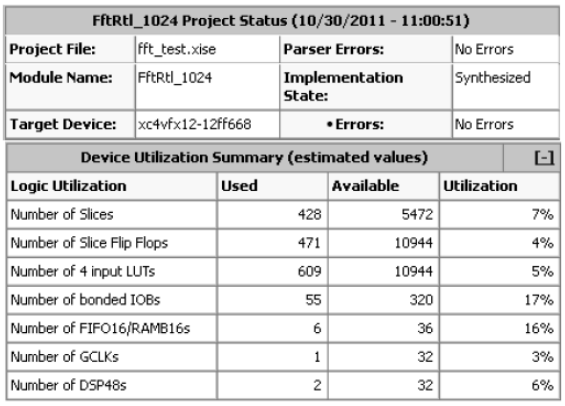
Figure 6: Rtl schematic of 1024 point fft in cordic.



cfft1024X12 Project Status (11/18/2011 - 08:58:54)			
Project File:	testfile.xise	**Parser Errors:**	No Errors
Module Name:	cfft1024X12	**Implementation State:**	Synthesized
Target Device:	xc4vfx12-12ff668	**• Errors:**	No Errors

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	684	5472	12%
Number of Slice Flip Flops	1097	10944	10%
Number of 4 input LUTs	1298	10944	11%
Number of bonded IOBs	68	320	21%
Number of FIFO16/RAMB16s	2	36	5%
Number of GCLKs	1	32	3%

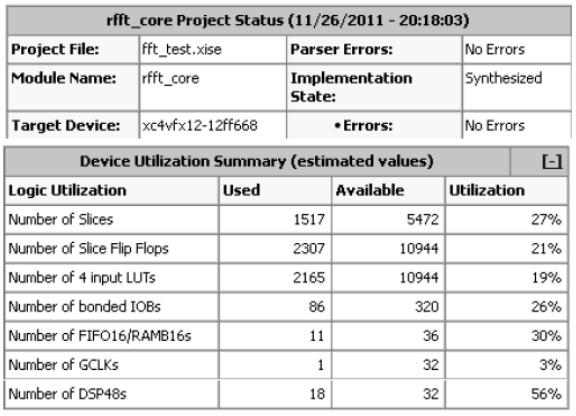
Figure 7: Device utilization summary of 1024 point fft processor using cordic



FftRtl_1024 Project Status (10/30/2011 - 11:00:51)			
Project File:	fft_test.xise	**Parser Errors:**	No Errors
Module Name:	FftRtl_1024	**Implementation State:**	Synthesized
Target Device:	xc4vfx12-12ff668	**• Errors:**	No Errors

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	428	5472	7%
Number of Slice Flip Flops	471	10944	4%
Number of 4 input LUTs	609	10944	5%
Number of bonded IOBs	55	320	17%
Number of FIFO16/RAMB16s	6	36	16%
Number of GCLKs	1	32	3%
Number of DSP48s	2	32	6%

Figure 8: Device utilization summary of 1024 point fft processor using Sine-Cosine Look up table



rfft_core Project Status (11/26/2011 - 20:18:03)			
Project File:	fft_test.xise	**Parser Errors:**	No Errors
Module Name:	rfft_core	**Implementation State:**	Synthesized
Target Device:	xc4vfx12-12ff668	**• Errors:**	No Errors

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	1517	5472	27%
Number of Slice Flip Flops	2307	10944	21%
Number of 4 input LUTs	2165	10944	19%
Number of bonded IOBs	86	320	26%
Number of FIFO16/RAMB16s	11	36	30%
Number of GCLKs	1	32	3%
Number of DSP48s	18	32	56%

Figure 9: Device utilization summary of 1024 point Xilinx Core FFT processor

3.2 Design Summary of 64-point FFT processor using vedic mathematics

The entire architecture was synthesized and implemented using Xilinx ISE v13.1. The device used for testing the design was Virtex-5 FPGA. The functionality was tested by creating test bench waveform and used in behavioral and post layout simulations. Fig. 10[4] depicts simulation result of the proposed design. A signal “inputbusy” is asserted high as soon as all inputs are stored into RAM. This signal is used to start the FFT calculation and to read RAMs, which contain 12 bits wide data input values. When signal “inputbusy” goes low the next stage of FFT is enabled. The “invert” signal specify FFT/IFFT mode. Here, we have used FFT mode of operation by setting “invert” signal to low logic. The output values are 14 bits wide with 3 bits representing the fractional part. A signal “outdataen” represents valid output. When it is asserted high, this indicates presence of valid output. A signal “reset” is active high and used to reset the system. Table 1[4] shows synthesis result for proposed design. Of the 7200 available slices on the Xilinx Virtex-5 FPGA, proposed architecture consumes around five percent occupying 373 slices. The number of lookup tables (LUTs) available is 28,800 and the FFT uses four percent available resources. Dynamic power consumption is only 10 mW. Table III shows performance comparison of proposed FFT design with other existing designs. The result shows that the number of LUTs and slices has been reduced significantly in the proposed design. The operating clock frequency is 380.51 MHz.

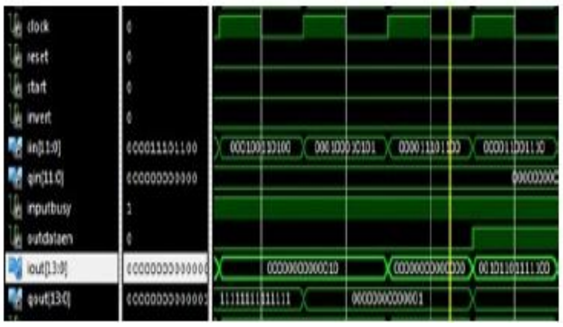


Figure 10: Simulation of proposed FFT using Vedic algorithm.

Table 1: Synthesis result for proposed design

Parameters	Proposed FFT Design
Word length	12 bits
Number of Slices	373
Occupied LUTs	1212
Operating Frequency	380.51 MHz
Dynamic Power (mW)	10

IV CONCLUSION

This paper presents 1024 point FFT processor using three different architectures which are portable among different EDA tools and technology independent. The whole designs are implemented in VHDL through Xilinx ISE11.1 The performance of the designs that is using CORDIC algorithm, and using Sine-Cosine look up table, have been compared with the commercial cores provided by Xilinx . This core was configured with the closet characteristics to our designs in order to make the results comparable. The performance of our designs present better results in terms of physical resources demanded but the throughput is poorer when compared with the IP commercial implementations. Along with these performance results come other considerations which need to be evaluated to select the best approach depending on system requirements like easy implementation, costs and performance. The generation of a design from an IP commercial core is as easy as to press a button but the design has not been controlled because they are provided as a black box. They offer a variety of features and functionalities to be configured and supposedly their implementations are optimized for a subset of their devices, giving the best performance for them but they lack portability. Our FFT designs have been integrated as part of a Speech Recognition System together with the other parts of the system such as end point detection, MFCC feature extraction. In this case the physical resources performance in order to have full implementation of the system in the same FPGA is more important than other criteria used. The designs are currently under final FPGA realization and will be reported in the future.

The 64-point FFT implemented using Vedic concepts and modified adder was found to have a good balance between performance and hardware requirements design had a maximum clock frequency of 95.2MHz. Also, area minimization is obtained by devising an efficient Vedic algorithm based butterfly processing structure, while the novel twiddle factor multiplier has low power consumption and hardware complexity. Synthesis results show that the proposed FFT processor can provide up to 380.51 MHz speed and slices count is 373. The proposed FFT architecture can also be altered to support other longer FFT sizes. The main applications of this proposed FFT are in Digital Signal Processing, OFDM Systems, Digital Image Processing and Communication systems.

IV. REFERENCES

[1]. António M. Grilo, Jaime Chen, Manuel Díaz, Daniel Garrido, and Augusto Casaca, "An Integrated WSAN and SCADA System for Monitoring a Critical Infrastructure", IEEE Transactions on Industrial Informatics, Vol. 10, No. 3, pp. 1755-1764, August 2014.

[2]. Debalina Ghosh, Depanwita Debnath, Dr. Amlan Chakrabarti "FPGA Based Implementation of FFT Processor Using Different Architectures", IJAITI VOLUME 1 NUMBER 1, PP 24-33, Jan/Feb 2012.

[3]. Nisha John, Prof. Sadanandan G.K, "FPGA Implementation of a Novel Efficient Vedic FFT/IFFT Processor For OFDM", International Journal of Advanced Research in Electrical, Electronics and

Instrumentation Engineering, Vol. 3, Issue 3, September 2014.

[4]. More T.V. ,Panat A.R. "FPGA implementation of FFT using vedic algorithm", Computational intelligene and Computing Research(ICCIC),pp-1-5 , 2013.

[5]. Harpreet Singh Dhillon, AbhijitMitra, "A Digital Multiplier Architecture using UrdhvaTiryakbhyam Sutra of Vedic Mathematics",IITG , pp-1-4 , 2010.

[6]. AsmitaHaveliya, "FPGA implementation of a Vedic convolution algorithm", International Journal of Engineering Research and Applications, Vol. 2, Issue 1, pp.678-684,Jan-Feb 2012.

[7]. A.Ronisha Prakash, S. Kirubaveni, "Performance Evaluation of FFT Processor Using Conventional and Vedic Algorithm", IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology, pp-1-6, 2013.

[8]. http://www.xilinx.com/support/documentation/ip_documentation/xfft/v9_0/pg109-xfft.pdf