International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Design of an FFT Processor using Mixed-Radix Algorithm for OFDM system

Jaishri Katekhaye[1], Amit Lamba[2]

[1]RTMN University, Electronics and Communication Department, GHRAET, Nagpur, India

[2] Professor, RTMN University, Electronics and Communication Department, GHRAET, Nagpur, India

Abstract: ***A parallel Fast Fourier Transform (FFT) processor for the use in Orthogonal Frequency Division Multiplexing (OFDM) is proposed here. The proposed processor is 64-point which is based on mixed radix (4-2) algorithm and execute 16-bit fixed point data format. The clock cycles required for the FFT processor are 92 which are less in number due to the use of parallel processing. The delay required for simulation of 64-point FFT is 11.924ns. For simulation we used XILINX 14.2 ISE software and for coding we employed Very High Speed Integrated Circuit Hardware Description Language (VHDL).***

Keywords: FFT, Mixed Radix, OFDM, VHDL**.**

1. Introduction

Fast Fourier transform (FFT) are widely used in different areas of applications such as communications, radars, imaging, etc. One of the major concerns for researchers is to meet real-time processing requirements and to reduce hardware complexity mainly with respect to area and power and to improve processing speed. Fast Fourier Transform (FFT) is an efficient algorithm to evaluate DFT. The DFT is obtained by decomposing a sequence of values into components of different frequencies. This operation is useful in many fields but computing it directly from the definition is often too slow to be practical. An FFT is a way to compute the same result more quickly. Discrete Fourier Transform (DFT) is defined as

$$X[K] = \sum_{n=0}^{N-1} x[n].\ W_N^{nk} \qquad (1)$$

$$W_N^{nk} = e^{-j2\pi nk/N}\ 0 \le k \le N\text{-}1$$

Where X[k] and x[n] are frequency domain and time domain sequences. To compute all N values DFT requires N^2 complex multiplications and N(N-1) complex additions. Since the amount of computation and thus the computation time, is approximately proportional to N^2, it will cost a long computation time for large values of N. For this reason, it is very important to reduce the number of multiplications and additions. This algorithm is an efficient algorithm to compute the DFT, which is called Fast Fourier Transform (FFT) algorithm.

2. Literature Review

In paper [1] radix-2 algorithm is used for FFT implementation. Also they proposed pipeline architecture. Pipeline architecture is also known as cascaded FFT architecture in which each stage has its own processing element. The proposed architecture operates at fixed and variable length FFT processor and gives good speed, flexibility and scalability. It supports any N-point FFT. Author proposed from 16-point to 2048-point. It meets the requirement of various wireless standards. ROM is used for the storage of twiddle factors. It is found that clock cycles required for 64-point FFT processor are 254 which are large in number. The time required for the computation of processor is 1.27μs. It is because of radix-2 algorithm.

In paper [6] comparison of 32-point and 64-point FFT is done by using radix-2 algorithm. Here, author has employed Decimation-In-Time (DIT) algorithm for radix-2. Here the separation of FFT points is done in such a way that at first points are divided into 2, then into 4, then into 8 after that 16 and so on. Due to this division of FFT points, the no of stages required for the calculation of butterflies are more. Hence delay required is more. Therefore delay required in case of 32-point FFT is 31.522ns and the delay required in case of 64-point FFT is 30.412ns. It is found that 32-point require 5 stages and the 64-point requires 6-stages. Hence delay required in case of 64-point is more [6].

3. Methodology

In our project, we have used mixed-radix algorithm for the design of FFT processor. Mixed-radix having the combination of radix-4 and radix-2 is employed. Hence before switching towards mixed-radix algorithm let we have the little knowledge of radix2 and radix-4.

3.1 Radix-2

There are two types of algorithms Decimation-in-time (DIT) and Decimation-in-frequency (DIF). The radix-2 is the simplest one among all algorithms.. The decimation-in-frequency (DIF) radix-2 FFT divides the DFT equation into even-indexed and odd-indexed outputs, which can each be computed by small-length DFTs of various combinations of input samples.

$$X[K] = \sum_{n=0}^{N-1} x[n].\ W_N^{nk}$$

Above formula is split into two summations-

$$X[K] = \sum_{n=0}^{\frac{N}{2}-1} x(n)\, W_N^{nk} + \sum_{N-1}^{\frac{N}{2}-1} x(n)\, W_N^{nk}$$

$$X[K] = \sum_{n=0}^{\frac{N}{2}-1} x(n)\, W_N^{nk} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right) W_N^{\left(n+\frac{N}{2}\right)k}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n)\, W_N^{nk} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right) W_N^{nk}.\, W_N^{\frac{N}{2}k}$$

Licensed Under Creative Commons Attribution CC BY

And, $W_N^{\frac{N}{2}.k} = (-1)^k$

$$X[K] = \sum_{n=0}^{\frac{N}{2}-1}\{x(n) + (-1)^k . x(n+\tfrac{N}{2})\}. W_N^{nk} \quad (2)$$

Radix-2 decimation-in-frequency algorithm rearranges the Discrete Fourier Transform (DFT) equation into two parts: Computation of even-numbered discrete Fourier indices X(2K) for k= 0, 2, 4,….N-2 as in (3) and computation of odd-numbered indices X(2K+1) for K=1, 3, 5,….N-1 as in (4).

$$X[2K] = \sum_{n=0}^{\frac{N}{2}-1}(x(n) + x(n+\tfrac{N}{2}))\, W_N^{2nk} = \sum_{n=0}^{\frac{N}{2}-1}(x(n) + x(n+\tfrac{N}{2}))\, W_{\frac{N}{2}}^{nk} \quad (3)$$

$$X[2K+1] = \sum_{n=0}^{\frac{N}{2}-1}(x(n) - x(n+\tfrac{N}{2}))\, W_N^{2nk} = \sum_{n=0}^{\frac{N}{2}-1}(x(n) - x(n+\tfrac{N}{2}))\, W_{\frac{N}{2}}^{nk} \quad (4)$$

Mathematical simplification in (3) and (4) shows that both even-indexed and odd-indexed can each be computed by a length N/2 DFT. The input to these DFTs are sums or differences of the first and second halves of the input signal, where the input to the short DFT producing the odd-indexed frequencies is multiplied by a so called twiddle factor term $W_N^k = e^{-\frac{i2\pi k}{N}}$. This is called decimation-in-frequency.

3.2 Radix-4

The butterfly of a radix-4 algorithm consists of four inputs and four outputs Fig 1. The FFT length is 4M, where M is the number of stages. A stage is half of radix-2. The radix-4 DIF FFT divides an N-point discrete Fourier transform (DFT) into four N/4 -point DFTs, then into 16 N/16 -point DFTs, and so on. In the radix-2 DIF FFT, the DFT equation is expressed as the sum of two calculations. One calculation sum for the first half and one calculation sum for the second half of the input sequence. Similarly, the radix-4 DIF fast Fourier transform (FFT) expresses the DFT equation as four summations, then divides it into four equations, each of which computes every fourth output sample. The following equations illustrate radix-4 decimation in frequency. There are two types of algorithms Decimation-in-time (DIT) and Decimation-in-frequency (DIF).

Equation (1) is split into four summations-

$$X[K] = \sum_{n=0}^{\frac{N}{4}-1} x(n) W_N^{nk} + \sum_{\frac{N}{4}}^{\frac{2N}{4}-1} x(n) W_N^{nk} + \sum_{\frac{2N}{4}}^{\frac{3N}{4}-1} x(n) W_N^{nk} + \sum_{\frac{3N}{4}}^{N-1} x(n) W_N^{nk}$$

$$X[K] = \sum_{n=0}^{\frac{N}{4}-1} x(n) W_N^{nk} + \sum_{n=0}^{\frac{N}{4}-1} x\left(n+\tfrac{N}{4}\right) W_N^{\left(n+\frac{N}{4}\right)k} + \sum_{n=0}^{\frac{N}{4}-1} x\left(n+\tfrac{N}{2}\right) W_N^{\left(n+\frac{N}{2}\right)k} + \sum_{n=0}^{\frac{N}{4}-1} x\left(n+\tfrac{3N}{4}\right) W_N^{\left(n+\frac{3N}{4}\right)k}$$

$$= \sum_{n=0}^{\frac{N}{4}-1}\left[x(n) + x\left(n+\tfrac{N}{4}\right) W_N^{\left(n+\frac{N}{4}\right)k} + x\left(n+\tfrac{N}{2}\right) W_N^{\left(n+\frac{N}{2}\right)k} + x\left(n+\tfrac{3N}{4}\right) W_N^{\left(n+\frac{3N}{4}\right)k}\right] . W_N^{nk} \quad (5)$$

The three twiddle factor coefficients can be expressed as follows:

$$W_N^{(N/4)k} = \left[\cos(\pi/2) - j\sin(\pi/2)\right]^k = (-j)^k$$

$$W_N^{(N/2)k} = \left[\cos(\pi) - j\sin(\pi)\right]^k = (-1)^k$$

$$W_N^{(3N/4)k} = \left[\cos(3\pi/2) - j\sin(3\pi/2)\right]^k = j^k$$

Equation (5) can thus be expressed as:

$$X(k) = \sum_{n=0}^{N/4-1}\left[\begin{matrix} x(n) + (-j)^k x\left(n+N/4\right) + (-1)^k x\left(n+N/2\right) \\ +(j)^k x\left(n+3N/4\right) \end{matrix}\right] W_N^{nk} \quad (6)$$

To arrive at four point DFT decomposition, let $W_N^4 = W_N$ Then equation (6) can be written as four N/4 point DFTs.

$$X(4k) = \sum_{n=0}^{N/4-1}\left[\begin{matrix} x(n) + x\left(n+N/4\right) + x\left(n+N/2\right) \\ +x\left(n+3N/4\right) \end{matrix}\right] W_{N/4}^{nk}$$

$$X(4k+1) = \sum_{n=0}^{N/4-1}\left[\begin{matrix} x(n) - jx\left(n+N/4\right) - x\left(n+N/2\right) \\ +jx\left(n+3N/4\right) \end{matrix}\right] W_N^n W_{N/4}^{nk}$$

$$X(4k+2) = \sum_{n=0}^{N/4-1}\left[\begin{matrix} x(n) - x\left(n+N/4\right) - x\left(n+N/2\right) \\ -x\left(n+3N/4\right) \end{matrix}\right] W_N^{2n} W_{N/4}^{nk}$$

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

for k=0 to N/4

X(4k), X(4k+1), X(4k+2) and X(4k+3) are N/4 point DFTs. Each of their N/4 points is a sum of four input samples x(n), x(n+N/4), x(n+N/2) and x(n+3N/ 4), each multiplied by either +1, -1, j, or -j. The sum is multiplied by a twiddle factor (W_N^0 W_N^n W_N^{2n} W_N^{3n}). The four N/4 point DFTs together make up an N point DFT. Each of these N/4 point DFTs is divided into four N/16 point DFTs. Each N/16 DFT is further divided into four N/64 point DFTs, and so on, until the final decimation produces four point DFTs. The four point DFT equation makes up the butterfly calculation of the radix-4 FFT. A radix-4 butterfly is shown graphically in fig 4.

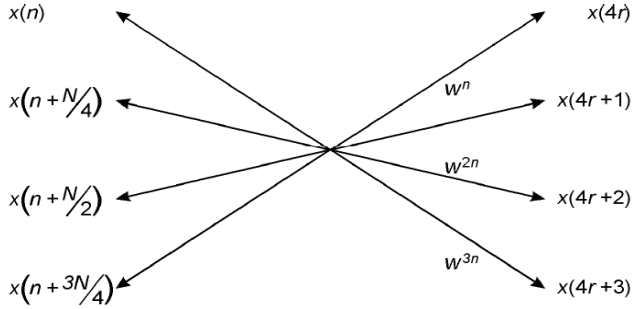

Figure1: Radix-4 DIF FFT butterfly

3.3 Mixed-Radix

A mixed radix algorithm is a combination of different radix-r algorithms. That is, different stages in the FFT computation

Licensed Under Creative Commons Attribution CC BY

have different radices. For instance, a 64-point long FFT can be computed in two stages using one stage with radix-4 processing elements, followed by a stage of radix-2 processing elements. This adds a bit of complexity to the algorithm compared to radix-r, however it gives more options in choosing the transform length. The Mixed-radix FFT algorithm is based on sub-transform modules with highly optimized small length FFT, which are combined to create large FFT. However, this algorithm does not offer simple bit reversing for ordering the output sequences. Following figure [2] shows the basic mixed radix algorithm employed in our design which consists of two butterfly units of radix-4 and four units of radix-2. But the actual working in the mixed-radix butterfly is done as shown in figure [3].

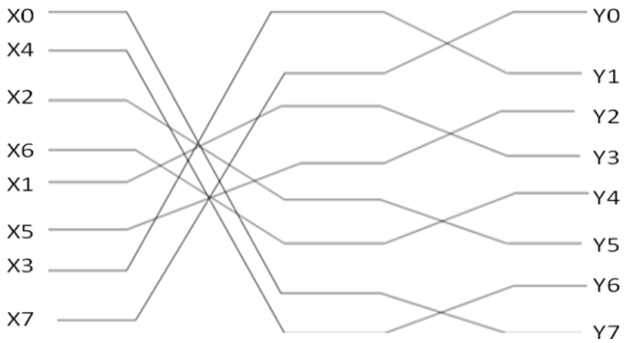

Figure2: Basic mixed-radix operation

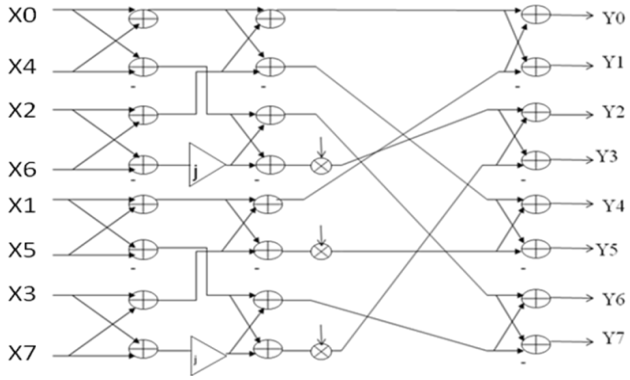

Figure3: Actual operation of mixed-radix algorithm

4. Architecture

Following figure shows the architecture of proposed processor:

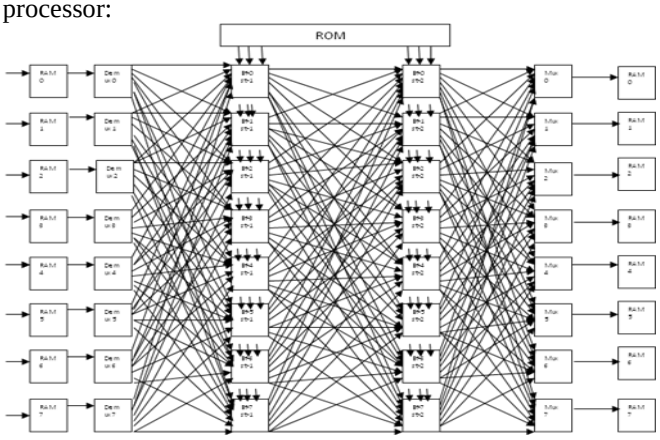

Figure4: Architecture of FFT processor

The proposed architecture consists of two sets of RAM memory, two stages of butterfly units, ROM memory in order to store the twiddle factors and also there is one address generation unit which generates the addresses of ram memory for storage of inputs and outputs, ROM memory for addresses of twiddle factors and one control unit for synchronisation. The address generation unit and control unit are not shown in figure. Also it contains de-multiplexers, multiplexers. The data provided to the ram memory is in text format which is given one input per clock cycle. Hence, in order to save 64-points 64 clock cycles are required. But the data we want for butterfly calculations is parallel so that it will calculate all the 64-points at a time. Therefore we used de-multiplexers between ram memory and first stage of butterfly units. The input data considered here is real having imaginary part zero which is in 16-bit fixed point format. But the data we are getting at the output side is complex having real as well as imaginary part due to the multiplication of twiddle factors. Here, the data coming out from second stage of butterfly unit is given to the multiplexer which is the complex data. After that it is stored in another set of RAM memory.

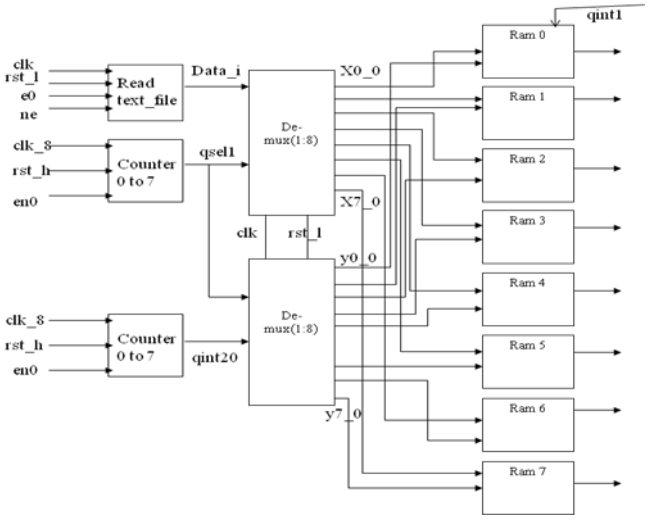

Figure5: How data enter into RAM

The above figure shows how data enter into the RAM. Here we have taken a text file where all 64-points are stored. But for writing those 64-points into the RAM, we require 64 clock pulses. Also for giving data in RAM, we require four counters. One for counting 64-points, one for providing write address, one for providing read address and the one for counting data 8 times. Here, two de-multiplexers are used. First one is for providing data to RAM and the second one for proving address to the respective data. The first four clock signals are given for the initialization process. Hence after the fourth clock pulse actual working will start. Hence at the 68^{th} clock pulse all the 64-points will be in RAM.

Licensed Under Creative Commons Attribution CC BY

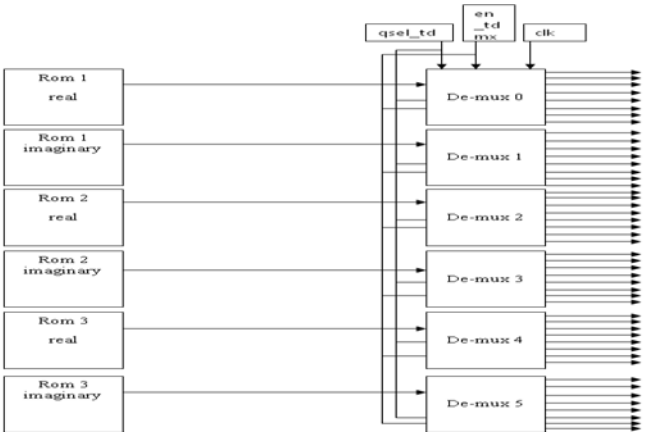

Figure6: Extraction of twiddle factors from ROM

ROM memory is used for the storage of twiddle factors. The first twiddle factor of all the radix-4 butterfly is 1. So, it is not shown in diagram. Here six ROM memories are used for the storage of real and imaginary part of twiddle factors separately for radix-4 calculation. The twiddle factor for all radix-2 is 1. Two counters are used one for selection of data from ROM and other for the de-multiplexer. At the 69th clock pulse ROM is enabled and at the 77th clock pulse all the twiddle factors are in de-multiplexers and whole data is in respective de-multiplexers as shown in figure [3] for the further butterfly calculations. At the 92 clock pulse the real and imaginary part of output is available in RAM as shown in figure [4].

5. Simulation Results

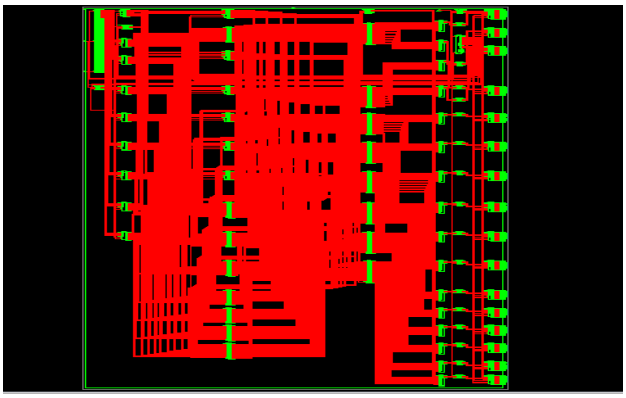

Figure7: RTL view

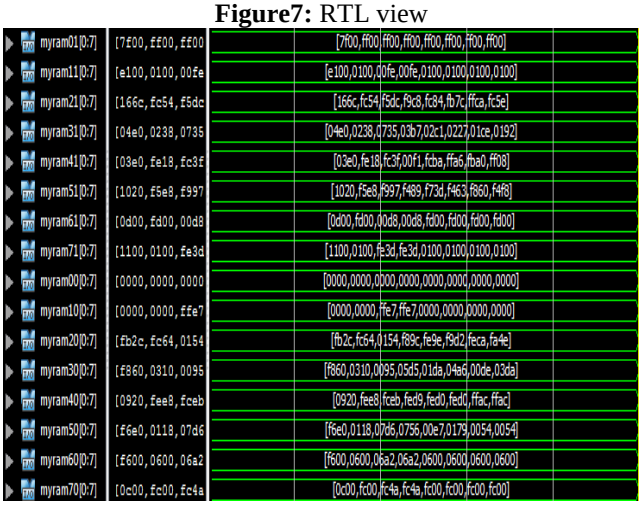

Figure8: output of 64-point FFT

Table1: The device utilization summary

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	7350	607200	1%
Number of Slice LUTs	24779	303600	8%
Number of fully used LUT-FF pairs	2985	29144	10%
Number of bonded IOBs	2306	700	329%
Number of BUFG/BUFGCTRLs	2	32	6%
Number of DSP48E1s	192	2800	6%

Table 2: Comparison table

	Delay	Clock cycles
Obtained results	11.924ns	92
Deepak Revanna, Omer Anjum, Manuele Cucchi, Roberto Airoldi, Jari Nurmi, "A Scalable FFT Processor Architecture for OFDM Based Communication Systems".	1.27us	254
M.Vijaya kumar, M.Vidya and G.Sriramulu, "Design and VLSI Implementation of A Radix-4 64 - Point FFT Processor".	29.688ns	-
K. Sowjanya, B. Leele Kumari, "Design and Performance Analysis of 32 and 64 Point FFT using RADIX-2 Algorithm".	30.412ns	-

6. Conclusion

In this paper, we conclude the overall process of FFT processor using Mixed-Radix algorithm. And it is found that this architecture of the FFT processor is efficient for the transmission of data at high speed. Hence it is very much useful for the OFDM system. The device used is vertex-7 using Xilinx ISE 14.2 software. The delay found in our project is 11.924ns and number of clock cycles required are 92. In [1], number of clock cycles used are 254 which are very high in number due to the use of radix-2 algorithm. In [4], [6], the delay required is more as compared to the proposed processor. Hence the proposed FFT processor is efficient for the fast communication of data.

References

[1] Deepak Revanna, Omer Anjum, Manuele Cucchi, Roberto Airoldi, Jari Nurmi, "A Scalable FFT Processor Architecture for OFDM Based Communication Systems", 2013 IEEE.

[2] Mounir Arioua, Said Belkouch, Mohamed Agdad, Moha M'rabet Hassani, "VHDL Implementation of an Optimized 8-point FFT/IFFT processor in Pipeline Architecture for OFDM systems",2010 IEEE.

[3] N Kirubanandasarathy , Dr.K.Karthikeyan & K.T hirunadanasikamani, "VLSI Design of Mixed radix FFT Processor for MIMO OFDM in wireless Communications", 201 0 IEEE.

[4] M.Vijaya kumar, M.Vidya and G.Sriramulu, "Design and VLSI Implementation of A Radix-4 64 - Point FFT Processor". International Journal of Research in Computer and Communication technology, IJRCCT, ISSN 2278-5841, Vol 1, Issue 7, December 2012.

Licensed Under Creative Commons Attribution CC BY

[5] Kala S, Nalesh S, S K Nandy, Ranjani Narayan "Design of a Low Power 64 Point FFT Architecture for WLAN Applications", 2013 IEEE.
[6] K. Sowjanya, B. Leele Kumari, "Design and Performance Analysis of 32 and 64 Point FFT using RADIX-2 Algorithm", Proceedings of AECE-IRAJ International Conference, 14th July 2013, Tirupati, India, ISBN: 978-81-927147-9-0.
[7] Vasantha Sudheer N, Venu Gopal B , "FPGA implementation of 64-Point FFT Processor", International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-1, Issue-4, September 2012 .
[8] Kandhi Srikanth , " Design Radix-4 64-Point Pipeline FFT/IFFT Processor for Wireless Application ". International Journal of Research in Computer and Communication technology, International Journal of Engineering Inventions e-ISSN: 2278-7461, p-ISSN: 2319-6491 Volume 3, Issue 2 (September 2013) PP: 67-70 .
[9] Sheng Zhou, Xiaochun Wang, Jianjun Ji, Yanqun Wang, "Design and Implementation of a 1024-point High-speed FFT Processor Based on the FPGA", 2013 6th International Congress on Image and Signal Processing (CISP 2013).

Author Profile

Jaishri Katekhaye is a PG scholar (M.Tech. VLSI) from GHRAET, from electronics and communication department, RTMN University. She completed the B.E. degree in Electronics and communication Engineering from Kavikulguru Institute of Technology and Science (KITS), Ramtek (Nagpur) from RTMN University, Maharashtra, India.

Prof. Amit Lamba is an assistance professor at GHRAET, electronics and communication department, RTMN University, Maharashtra, India.

Licensed Under Creative Commons Attribution CC BY