FPGA Based Memory-Less Phase Generation for Butterfly Operation of CORDIC FFT Processor

Saikat Kumar Shome, Santu Kumar Giri, and Uma Datta

***Abstract*—Efficient implementation of Fast Fourier Transform (FFT) in the hardware paradigm has been a major challenge for design engineers. Twiddle Factor generation and complex multiplication thereafter are the decisive steps of VLSI implementation of FFT. Conventional FFT analyzers call for a dedicated memory bank to store the twiddle factor angles in a predefined order. This storage results in a increased resource utilization which increases with N, the length of the Fourier Transform. This study presents a phase generation scheme that generates the necessary twiddle factor angles with simple hardware logic, depending on the present step and stage of FFT. This relinquishes the use of memory storage elements. Use of CORDIC to carry out complex multiplication further enhances system throughput. The present logic has been synthesized in Spartan 3E FPGA. The timing diagram results match the theoretical analysis and the synthesis report supports minimal hardware resource utilization.**

***Index Terms*—FFT, CORDIC, FPGA, memory-less.**

I. Introduction

Discrete Fourier Transform (DFT) plays a crucial role in Digital Signal Processing and is a powerful computational tool for evaluating Fourier Transform for sequences of finite length. DFT finds widespread usage in applications such as Synthetic Aperture Radar, medical image processing, Software Defined Radio [1] and Wireless communication protocols using OFDM. However, direct implementation of DFT is avoided due to its computational complexity. Fast Fourier Transform (FFT) is an efficient means of DFT computation, reducing the run time complexity from O(N^2) to O(Nlog2N) by exploiting the symmetric and periodic properties of Twiddle Factor [2].

In the recent past, there has been a rapid progress in FPGA technologies. This gain in renewed interest is mainly due to increased device size and emergence of fast hardware floating point library. FPGA co-processors have become an extremely cost effective means of off-loading computationally intensive algorithms, enhancing overall system performance. Reconfigurable computing architectures are sufficiently flexible so that new operations can be implemented on the existing hardware besides being high speed for real-time execution [3]. Moreover, the price/performance ratio of these systems makes them a broadly competitive alternative to ASICs. Besides reduced development time, decreased production cost while achieving higher throughput render FPGA nodes as a suitable hardware paradigm for low power, realtime applications.

The most computationally demanding stage of any FFT engine, especially in the hardware genre, is the butterfly operation. Conventionally, complex multipliers and adders constitute the butterfly unit and is the major speed impediment of FFT processors [4]. Though the VLSI multiplier structure can efficiently perform the complex multiplication, they are not much efficient in case of trigonometrical operations. To accomplish this, a common multiplier uses a Look UP Table (LUT). Although the process enhances throughput, the main setback is the requirement of a huge LUT in the form of ROM. The CORDIC algorithm is a hardware efficient alternating solution which eliminates the need of a dedicated multiplier hardware. Use of CORDIC instead of complex multiplier not only makes the hardware requirement very simple but also has the advantage of low switching activity, ideal for low power applications.

Of late, design of FFT architectures over VLSI platform has been a subject of thorough research [5]-[8]. Some FFT designs based on CORDIC have also been reported for various usance. Lin et al [9] proposed a modified CORDIC algorithm using mixed-scale rotation to reduce the total iterations albeit enhancing hardware complexity. Using non-iterative CORDIC micro-rotations, a non-recursive FFT has been proposed in [10]. Though the study reduces ROM size, it does not eliminate it completely. A memory less architecture with reduced memory requirement has been presented in [11] but calls for a complex implementation.

For butterfly operation, the CORDIC based FFT engine requires the storage of only the twiddle factor angles in ROM, instead of storing the actual twiddle factors. Prevalently, dedicated memory banks are needed for storing the twiddle factor angles for the rotation in such processors. This calls for a separate memory element thereby increasing the overall resource utilization of the system. In this study, we present an online phase generation logic for computation of twiddle factors angles. The designed rationale generates the necessary angles online successively through a shifter-comparator arrangement. With this approach, the memory requirements of the FFT processor is considerably reduced as the decisive angles are now generated rather than being stored in any dedicated memory element.

The paper is organized as follows. In Section II, a brief mathematical description of FFT and CORDIC is provided. The online phase generation logic and its hardware design is presented in Section III. Section IV contains the FPGA

Manuscript received January 18, 2013; revised May 31, 2013.

Saikat Kumar Shome and Santu Kumar Giri are with Scientist at Electronics & Instrumentation Division of CSIR-CMERI, Durgapur. Pin: 713209. West Bengal (e-mail : saikatkumarshome @gmail.com, saikatkshome@cmeri.res.in, sk_giri@cmeri.res.in)

Smt. Uma Datta is with the Department, Electronics & Instrumentation Division, CSIR-CMERI, Durgapur.(e-mail: uma_datta58@yahoo.in)

synthesis results and the resource utilization summary while Section V concludes the paper.

II. Fast Fourier Transform and CORDIC

A. Fast Fourier Transform-An Introduction

The FFT algorithm has become almost ubiquitous and most important in many high speed signal processing applications. An FFT produces exactly the same result as evaluating the DFT definition directly; the only difference is that an FFT is much faster [2]. The Discrete Fourier Transform (DFT) of an N-point discrete-time complex sequence $x(n)$, indexed by $n = 0, 1, \ldots, N\text{-}1$, is defined by,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad k = 0,1,\ldots,N\text{-}1 \tag{1}$$

where $W_N = e^{(-j2\pi/N)}$, commonly known as Twiddle Factor.

The excessively large amount of computations required to compute the DFT directly when N is large has prompted to work out alternate methods for computing the DFT efficiently. Fig. 1 shows a butterfly of radix-2 Decimation in Frequency (DIF) FFT algorithm. All such radix algorithms have similar structure, while differing only on computation of butterflies.

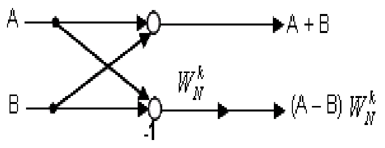

Fig. 1. Butterfly of a Radix-2 DIF FFT Algorithm.

B. Basics of CORDIC

In CORDIC algorithm, acronym for Coordinate Rotation Digital Computer, a vector (x, y) can be rotated through an arbitrary angle θ to obtain a new vector (x', y') [12]. The generalized equation governing CORDIC operation is given by

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2}$$

Since rotation is associative in nature, the order of operation is interchangeable ie.

$$\text{Rotation}\ (\alpha_1 \pm \alpha_2) = \text{Rotation}\ (\alpha_1)\ [\text{Rotation}\ (\alpha_2)] \tag{3}$$

Thus, the basic concept of the CORDIC computation is to decompose the desired rotation angle θ into the weighted sum of a set of predefined elementary rotation angles (α_i) satisfying the following condition

$$\theta = \alpha_0 \pm \alpha_1 \pm \alpha_2 \pm \ldots. \pm \alpha_i \pm \ldots.. \pm \alpha_{b-i}$$

where, b is the desired number of bits of precision and $i = 0$ to b-1.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\alpha_0 & \mp\sin\alpha_0 \\ \pm\sin\alpha_0 & \cos\alpha_0 \end{bmatrix} \begin{bmatrix} \cos\alpha_1 & \mp\sin\alpha_1 \\ \pm\sin\alpha_1 & \cos\alpha_1 \end{bmatrix} \cdots \begin{bmatrix} \cos\alpha_i & \mp\sin\alpha_i \\ \pm\sin\alpha_i & \cos\alpha_i \end{bmatrix} \cdots \begin{bmatrix} \cos\alpha_{b-1} & \mp\sin\alpha_{b-1} \\ \pm\sin\alpha_{b-1} & \cos\alpha_{b-1} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4}$$

In other words, the rotation angle θ can be expressed as $\theta = \sum_{i=0}^{b-1} d_i \alpha_i$, where $d_i \in \{-1,\ 1\}$, i.e. the signs of the elementary angles (α_i) are so chosen that the total angle will be equal to θ. This means that the sign of the difference between the angle θ and the partial sum of elementary angles $\theta - \sum_{j=0}^{i-1} d_i \alpha_i$ controls the sign d_i of the following elementary angle α_i with the angle approximation error δ as

$$\delta = \theta - \sum_{i=0}^{b-1} d_i \alpha_i \tag{5}$$

Factorizing cosine terms in Eq. (4) leads to

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \cos\alpha_0 \cos\alpha_1 \cdots \cos\alpha_i \cdots \cos\alpha_{b-1} \times \begin{bmatrix} 1 & \mp\tan\alpha_0 \\ \pm\tan\alpha_0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \mp\tan\alpha_1 \\ \pm\tan\alpha_1 & 1 \end{bmatrix} \cdots$$
$$\cdots \begin{bmatrix} 1 & \mp\tan\alpha_i \\ \pm\tan\alpha_i & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & \mp\tan\alpha_{b-1} \\ \pm\tan\alpha_{b-1} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{6}$$

To simplify the computation, the angles α_i are chosen such that (tan α_i) represents a series of powers of 2, i.e. tan $\alpha_i = 2^{-i}$ for $i = 0, 1, 2, \ldots., b$-1. So the multiplication by the tangent term is reduced to simple shift operation. Using this method, any angle between -90.0 and +90.0 can be represented as a sum or difference of the elementary angles. Hence the Eq. (6) can be written as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = K \begin{bmatrix} 1 & \mp 2^0 \\ \pm 2^0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \mp 2^1 \\ \pm 2^1 & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & \mp 2^i \\ \pm 2^i & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & \mp 2^{b-1} \\ \pm 2^{b-1} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

where, $K = \prod_{i=0}^{b-1} \cos\alpha_i = \prod_{i=0}^{b-1} \cos(\tan^{-1} 2^{-i}) = \prod_{i=0}^{b-1} \frac{1}{\sqrt{1+2^{-2i}}}$ (7)

III. Memory-Less Phase Generation Scheme for Butterfly Input

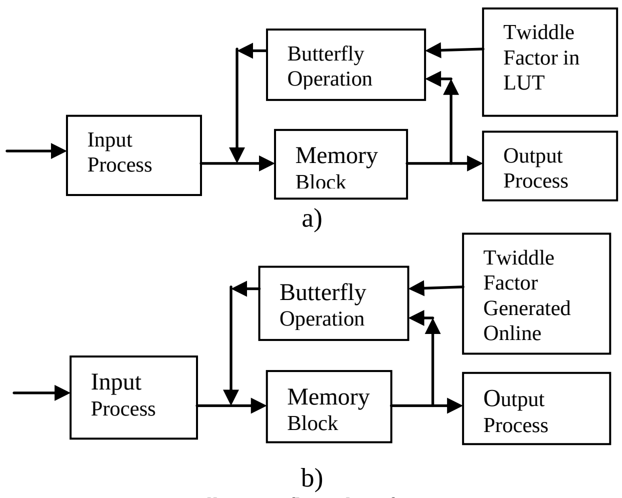

Fig. 2. Overall system floor plan of FFT processor.

A. Phase generation (kθ) for Butterfly Input

Fig. 2 (a) sketches the overall computation scheme of a traditional FFT computation paradigm. The mannerism is to save the Twiddle Factor values in a separate turf [13]. Our study employs CORDIC algorithm for ciphering butterfly operation and also eliminates the LUT based concept of storing Twiddle Factors, as in Fig. 2 (b). The details of the phase generation methodology and some customizations carried out for CORDIC usage are elaborated subsequently.

Since a CORDIC is expected to rotate a vector by any angle between 0 and 2π covering all quadrants, so instead of representing all the elementary angles in conventional radians, it is represented in a normalized 2's complement format with weights chosen from MSB side as –π, π/2, π/4, … , π/2^{b-1}, 'b' being the number of bits of precision This form of representing the angle offers the advantage of easy identification of the quadrant pertaining to the amount of rotation, simply by observing the first two bits from the MSB side. Also, for twiddle factor ($e^{-j\frac{2\pi nk}{N}}$) multiplication during butterfly operation in FFT, the rotation angle *kθ* (where $\theta = \frac{2\pi n}{N}$) for any butterfly stage can be represented in terms of multiplier *k* only, represented in Fig. 3.

As for ease of implementation, elementary rotation angles are represented as ($\tan^{-1} 2^{-i}$), however there is a constraint on the angle that a conventional CORDIC can compute, which ranges between *-π/2* to *π/2*. But as the CORDIC should be capable of computing for all angles between 0 to 2π, so the symmetry property of a sine wave is made use of. This process is carried out in two steps–(a) transform the given angle to either first or fourth quadrant by copying the (MSB-1) to MSB and computing the CORDIC operation within its operational range. First two MSB's of the normalized two's complemented representation of angle 00, 01, 10 and 11 represent the 1st, 2nd, 3rd and 4th quadrants respectively, where desired angle of rotation *z* lies. Thus 2nd and 3rd quadrant angles are transformed to 4th and 1st quadrant respectively by replacing first bit (MSB) with second bit (MSB-1), i.e. 01→11 and 10→00, where CORDIC can compute the vectors within its range. (b) Restore the correct result by reverting back to the original quadrant by changing the sign of the final output. For 512 point FFT, *N*=512, *n* = 9, $\theta = \pi/256$, and for 16-bit precision (wordlength, *b*=16), different values of *kθ* for first stage (*n*=1) of butterfly operation can be represented by Table I. For second stage of butterfly operation (i.e. *n*=2),

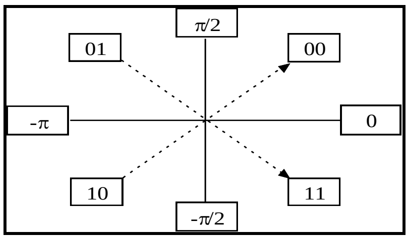

Fig. 3. Normalized angle representation for CORDIC.

The values of *kθ* are modified as in Table II and so on, for different stages. From the above tables it is observed that bits *b0* to *b6* and *b15* always remain 0, and *b7* to *b14* are corresponding to the outputs of and 8-bit up-counter with the varying output sequence depending on the number of stage of operation which is summarized in Table III.

TABLE I : TWIDDLE FACTOR VALUE FOR STAGE 0

Bit-b	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k	− π	π/2	π/4	π/8	π/16	π/32	π/64	π/128	π/256	π/512	π/1K	π/2K	π/4K	π/8K	π/16K	π/32K
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
…																…
255	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0

TABLE II: TWIDDLE FACTOR VALUE FOR STAGE 1

Bit-b	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k	− π	π/2	π/4	π/8	π/16	π/32	π/64	π/128	π/256	π/512	π/1K	π/2K	π/4K	π/8K	π/16K	π/32K
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
…																…
254	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
…									…							…
254	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

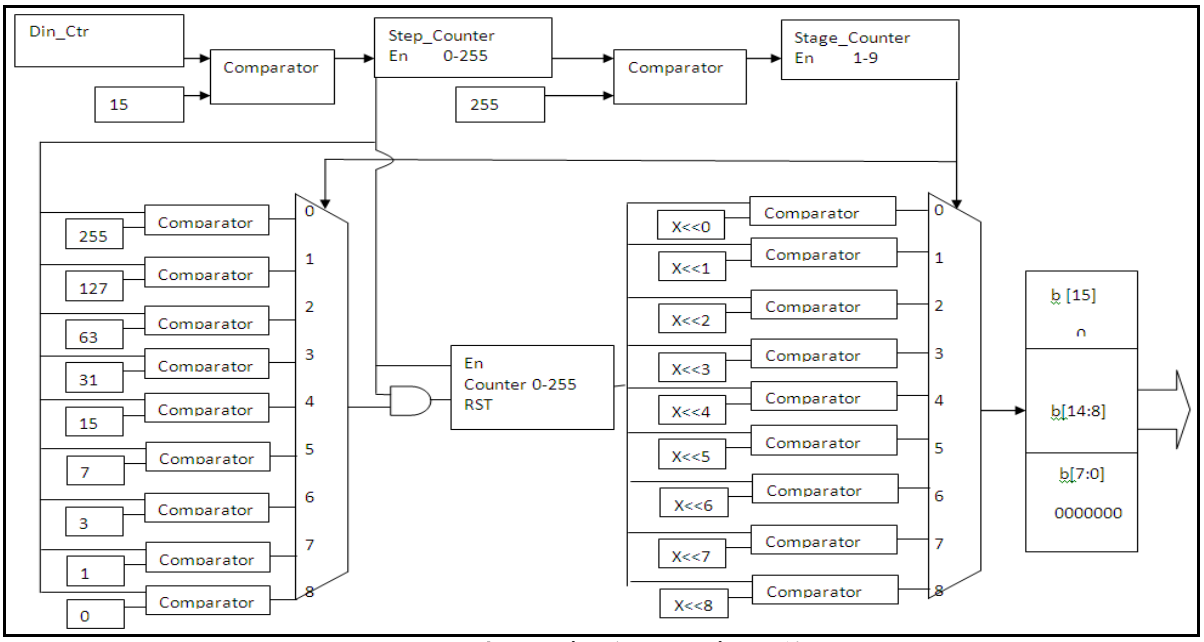

Fig. 4. Architecture for Kθ generator for N=512.

Stage no.	Value of $k\theta$ corresponding to bits b7 to b14	No. of times sequence repeats
1	0,1,2,3,4,5,6, ..., 255	1
2	0,2,4,6,8,10,12,...., 254	2
3	0,4,8,12,16,20, ... , 252	4
4	0,8,16,24,32, ... , 248	8
5	0,16,32,48,64, ... , 240	16
6	0,32,64,96,128, ..., 224	32
7	0, 64, 128, 192	64
8	1, 128	128
9	0	256

B. *Hardware Implementation*

Though the present archetype can be extended for any higher transform length (*N*), the case for *N*=512 has been considered for presentation in this paper.

Initially only 8-bits corresponding to b7 to *b14* are generated using a 8-bit up counter and then they are concatenated with other bit to represent the value of $k\theta$ in 16-bit two's complement format (Fig. 4). Since there are nine different sets of outputs corresponding to number of stages, two 9:1 desired output with the select input of MUX being controlled by a Stage counter, that counts form 0 to 8. As in each stage, there are (N/2) 256 butterfly the enable input of the Stage counter is controlled by a Step counter that counts from 0 to 255, shown in Fig 4. For stage-0, d0 of theoperations, so to ensure that the counter is incremented only after 256 outputs of a stage, MUXes are used to select the MUX1 will receive Step counter output for the full cycle from 0-255, which will be passed to the output without any shift, as the select signal of MUX, will remain at the same value continuously for 256 cycles.

TABLE III: STAGE-WISE PHASE GENERATION FOR N=512

During the next phase, when the Stage counter output is 1, the Step counter output fed to d1 of MUX1 will be reset after 128 counts (0 to 127). Further, this output will be multiplied by 2, by right shifting the data by one bit at the input of MUX2. So, the output of MUX2 will be available as 0, 2, 4, …., 254 and this process will be repeated twice. Similarly, the output for different stages will be generated. With 8-bits (d7-d14) already generated, this bits will be concatenated with other 8-bits (d0 to d6 and d15 being equal to zero) to provide $k\theta$ value as theta input to the CORDIC for twiddle factor multiplication. The angle generator generates the respective twiddle factor angles for each stage and step of the butterfly. This value is fed as input to the CORDIC block which multiplies (a-b) with this value.

IV. RESULTS AND DISCUSSION

The architecture has been mapped in Xilinx Spartan 3E FPGA XC2S200 device with speed grade -5. A technology independent schematic view which gives a basic logic representation of the circuit and the overall of the synthesis result of the block have been shown in Fig. 5 and Fig. 7 respectively. The ISim timing diagram for stage zero of a 512 point is depicted in Fig. 6; din counter stands for each bit of precision and counts from 0-15 for 16 bits of precision of each step. After its full count, the step counter is incremented by one for another 15 clocks and so on. Kq out contains the generated phase value for each step. Following the convention as mentioned in Table I, it is observed that for stage 0, the phase values for step 0, 1 and 2 are 0, π/256 and π/128 respectively which is in accordance with the theoretical results. The device utilization summary is shown in Table IV, suggests a minimal hardware resource consumption. It was found that the architecture can operate at a minimum time period of 13.212 ns with a maximum frequency of 76.22 MHz, using a total memory of 201MB.

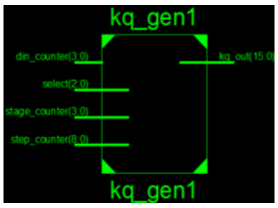

Fig. 5. RTL schematic view of angle generator.

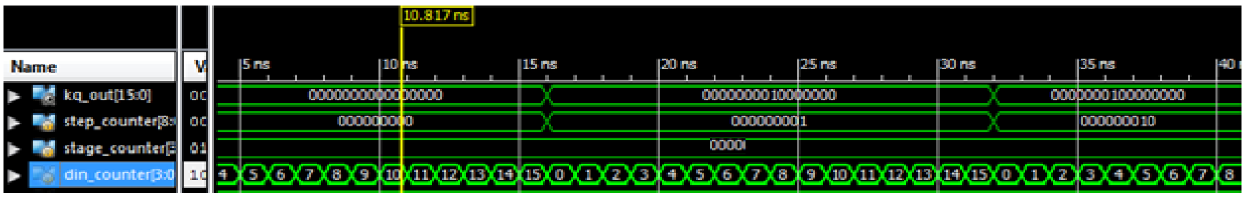

Fig. 6. RTL schematic view of angle generator.

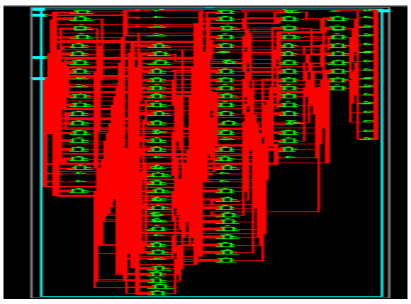

Fig. 7. Synthesis results of phase generator.

TABLE IV: DEVICE UTILIZATION SUMMARY

Resource Type	Used	Available	Percentage Utilization
Number of Slices	52	960	5.42
Number of 4 Input LUT	92	1920	4.79
Number of IOs	36		
Number of Bonded IOBs	32	66	48.48

V. CONCLUSION

In this work, leveraging the structured description of FFT algorithm, a FPGA based phase generation framework for FFT has been considered for. The phase generation technique eliminates the need of a dedicated memory bank as seen in conventional FFT engines. Although the design has been targeted for a Radix-2 Decimation in Frequency FFT processor of N=512, it can be extended for other N's with minor structural alterations. The adopted methodology of the present design is based on shifters and comparators which significantly reduce hardware as well as the latency introduced thereon. Such a reduced hardware logic when ported to a FPGA chip is ideal for low power, real time spectral analysis applications.

ACKNOWLEDGMENT

The authors acknowledge the continuous support of the Director, CSIR-CMERI, durgapur for this project.

REFERENCES

[1] S. Mittal, Z. A. Khan, and M. B. Srinivas, "Area Efficient High Speed Architecture of Bruun's FFT for Software Defined Radio," *IEEE Global Telecommunication Conference*, Nov. 2007.

[2] J. G. Proakis and D. G. Manolakis, "Digital signal processing: principles, algorithms and applications," *Pearson Prentice Hall*, 2007.

[3] S. Hauck. "The Roles of FPGAs in Reprogrammable Systems," in *Proc. of IEEE*, vol. 86, no. 4, pp. 615-638, 1998.

[4] J. H. Takala, T. S. Jarvinen, and H. T. Sorokin, "Conflict-Free Parallel Memory Access Scheme For FFT Processors," in *Proc. of the International Symposium on Circuits and Systems, ISCAS '03*, vol. 4, pp. 524-527, May 2003.

[5] Y. K. Xie and B. Fu, "Design And Implementation Of High Throughput FFT Processor," *Journal of Computer Research and Development*, vol. 41, no. 6, pp. 1022 1029, 2004.

[6] J. A. Hidalgo, J. Lopez, E Argiiello, and E. L. Zapata, "Area Efficient Architecture For Fast Fourier Transform," *IEEE Trans.Circuits Syst. 11*, vol. 46, no. 2, pp. 187-193, Feb. 1999.

[7] T. Pitkanen, T. Partanen, and J. Takala, "Low-Power Twiddle Factor Unit For FFT Computation," *SAMOS* 2007

[8] S. K. Shome, A. Ahesh, D. K. Gupta, and S. Vadali, "Low-Power Twiddle Factor Unit For FFT Computation Architectural design of a highly programmable Radix-2 FFT processor with efficient addressing logic," in *Proc. of IEEE International conference on Device, Circuits and Systems*, 2012.

[9] C. Lin and A. Wu, "Mixed-scaling-rotation CORDIC (MSR-CORDIC) algorithm and architecture for high-performance vector rotational DSP applications," *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 11, pp. 2385–2396, 2005.

[10] S. S. Abdullah, H. Nam, M. McDermot, and J. A. Abraham, "A high throughput FFT processor with no multipliers," in *Proc. IEEE International Conf. on Computer Design*, pp. 485–490, 2009.

[11] M. Garrido and J. Grajal, "Efficient memory-less CORDIC for FFT Computation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, pp. 113–116, 2007.

[12] J. Volder, "The CORDIC trigonometric computing technique," *IEEE Transactions on Electronic Computers*, vol. EC-8, no. 8, pp. 330-334, September 1959.

[13] J. O'Sullivan, S. Weiss, and G. Rice, "Automatic FFT code generation for FPGAs with high flexibility and human readability," *IEEE Signal, System and Computer Conference*, Nov. 2011

Saikat Kumar Shome received his Bachelor of Technology Degree in Computer Science & Engineering from Future Institute of Engineering and Management, Kolkata, India (2010) and his M.Tech. in Mechatronics from AcSir (CSIR), New Delhi, India (2012). Presently, he is working as a Scientist in Electronics & Instrumentation Group, CSIR-Central Mechanical Engineering Research Institute, Durgapur, (a Constituent Establishment of the Council of Scientific and Industrial Research), India and pursuing his Ph.D. His research areas include VLSI Signal Processing, Wireless Communication, Mechatronics, Bio-Medical Image Processing and Network Security.

Santu Kumar Giri received his B.E. (Hons.) and M. E. degree in Electronics and Telecommunication Engineering from Jadavpur University, Kolkata, India in 2004 and 2008 respectively and currently pursuing Ph. D there. Presently, he is working as Scientist in Electronics & Instrumentation Group, CSIR-Central Mechanical Engineering Research Institute, Durgapur, (a Constituent Establishment of the Council of Scientific and Industrial Research), India. His research interests include Microwave Communication, Digital Signal and Image Processing, Power Electronics and Motor Control.

Uma Datta received her B. Tech. degree in Electronics and Communication Engineering from Institute of Radio Physics and Electronics, Kolkata, India and M. Tech. degree in Industrial Electrical Systems from NIT, Durgapur, India respectively. She is presently working as a head and chief scientist in the department of Electronics and Instrumentation in Central Mechanical Engineering Research Institute, Durgapur. Her research interests include Control System for Machine Automation, Wireless Ad Hoc and sensor networks and Energy Harvesting System to extend Lifetime of Sensor Network. She is associated with a number of government and private sponsor projects and has published forty (40) research papers in various journals and conferences. She is a member of IEEE (Communication Society) and is a reviewer of several IEEE conference papers.