High throughput VLSI architecture for Blackman windowing in real time spectral analysis

K. C. Ray and A. S. Dhar
Department of E&ECE
Indian Institute of Technology
Kharagpur, India-721302
Email: {kcr, asd}@ece.iitkgp.ernet.in

***Abstract*— This paper presents a high throughput VLSI architecture for Blackman windowing. Since most of the implementation of windowing functions for real time applications, are based on either ROM or DSP processor. Here the proposed architecture is designed using major blocks like CORDIC(CO-ordinate Rotation DIgital Computer) and Han-Carlson adder. This architecture is flexible in terms of window length. So that a single chip can be used for those applications, where variable length is required. The synthesized result of 16-bit word size architecture with commercially available 0.18*μm* CMOS technology using Synopsys Design Analyzer, shows that the throughput of this architecture is 400Msamples/s with core area of 21*mm*^{2} .**

***Index Terms* — Blackman windowing, CORDIC, VLSI Architecture**

I. INTRODUCTION

In many digital signal processing applications, fast Fourier transform(FFT) is widely used for real time spectral analysis and filtering. For spectral analysis applications, well-liked windowing functions such as Hamming, Hanning and Blackman windowing methods have been used for preprocessing input signals before FFT to minimize the spectral leakage and picket fence effect [1]. In this paradigm, various algorithms and architectures for FFT [2-6] have been proposed for high speed implementations [3-4] with variable transform length [5]. So it requires a windowing computation with high throughput to meet the speed of the FFT processor. But to the knowledge of the authors, there is no new hardware architecture for windowing other than existing ROM based implementations which is having the constraints of speed and flexibility. To overcome these constraints we have proposed CORDIC based architectures [7, 8] for windowing to meet the specifications of recently developed FFT architecture in terms of high throughput and flexibility for real time applications. In this work, we have enhanced the throughput of the proposed architecture for Blackman windowing [7] using a fast adder known as Han-Carlson (HC) adder [9]. This work at the beginning presents expression for Blackman windowing functions, and then, mapping that expression to the architecture for parallelism and pipelining hardware implementation [10, 11]. Here CORDIC units in different modes such as circular and linear mode, are used primarily as basic block for trigonometric and linear multiplication functions. The critical path for this architecture is simply the adder/subtractor module in the pipelined CORDIC unit, so we have used a fast adder known as Han-Carlson adder to make it suitable for real time applications that require very high throughput rate.

Rest of this paper is organized as follows. Section II describes Blackman windowing function and CORDIC algorithm with its architectures. Our proposed Blackman windowing architecture is discussed in Section III with highlighting the Han-Carlson adder and its implementation subsequently. Section IV presents the synthesis results and discussions, and finally Section V concludes the paper.

II. BACKGROUND

A. Blackman Windowing

During spectral analysis, the input signals are to be truncated to fit a finite observation window according to the length of FFT processor. This direct truncation using conventional windowing, known as rectangular window function leads to undesirable effects known as spectral leakage and picket fence effect in frequency domain. To minimize these effects during spectral analysis, researchers have proposed different kinds of windowing functions [1] such as Hanning, Hamming and Blackman windowing functions. These windowing functions are widely adopted because of their good spectral characteristics like central peak width, 6-dB point, highest side lobe and rate of side lobe fall off and equivalent noise bandwidth (ENBW). Among these, Blackman windowing leads to better side lobe attenuation. It is needless to present all these characteristics in detail here, however readers may refer to [1] for the same. Here only Blackman windowing has been discussed for implementation. Though ROM based

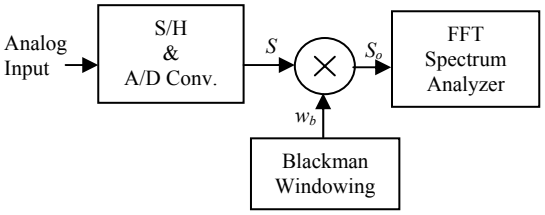


Fig.1. Spectral analysis system

implementation is already existing, which restricts flexible implementation and also restricts fitting with the advanced FFT processors in terms of variable length and speed. Basic idea of this work is to propose a flexible and fast architecture for Blackman windowing function to fit with the advanced FFT processor. Before presenting the proposed architecture in the next section, Blackman windowing function has been highlighted here briefly. A typical block diagram for real time FFT based spectral analysis system is shown in Fig.1, where input samples are preprocessed with real time Blackman windowing function before FFT as follows

$$s_o(r) = w_b(n) \cdot s_i(\text{N-1+ r} - n) \quad (1)$$

Here r indicates present time variable. n is discrete time index ranges from *0* to *N-1,* where N is the windowing or observation length. s_i and s_o are input and output signals for windowing function, where as $w_b(n)$ is the Blackman windowing function defined as follows.

$$w_b(n) = b_0 + b_1 \cos(\tfrac{2\pi n}{N}) + b_2 \cos(\tfrac{4\pi n}{N}) \quad (2)$$

where b_0 =0.43, b_1 =0.5 and b_2 =0.08 and n varies from $-\frac{N}{2}$ to $\frac{N}{2}$, here N is the window(observation) length.

The Blackman window, with the above approximation coefficients, provide attenuation of at least 60dB of side lobes[1] with only a modest increase in computation over that required by the Hanning and Hamming window due to another cosine term as in equation (2). This windowing function demands the attention for designing hardware efficient, flexible window length setting and high throughput VLSI architecture using CORDIC whose implementation is quite economic in terms of hardware. Now from equation (2), we shall have a parallel and pipelined architecture for aforesaid windowing function, where the selection of window length (N) is user defined as per requirement for the application. Since the equation (2) needs trigonometric computation, so the implementation using CORDIC algorithm is better choice in terms of computation and to change the value of N dynamically. But look up table or ROM method fails to achieve the same. In case of fixed N also, though existing implementation is based on look up table, it consumes more time to access the ROM and to compute multiplication and addition. Whereas CORDIC based proposed architecture gives same result with high throughput and lesser hardware compared to ROM based computation. Here multiplication and trigonometric computations are realized using linear and circular CORDIC algorithm respectively.

B. CORDIC algorithm and its pipelined Architecture

The CORDIC algorithm was first developed by Volder [12] and further modified for computing different elementary functions [13]. The CORDIC algorithm is an iterative method to compute rotation of two dimensional vectors and to evaluate the magnitude and angle of rotated vector. The generalized rotational equation for vectors is given as follows,

$$\begin{bmatrix} x_m \\ y_m \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad (3)$$

where (x_0, y_0) and (x_m, y_m) are input and output vectors respectively for rotation angle θ as input. The angle (θ) is now decomposed to elementary angles.

$$\theta = \sigma_0\beta_0 + \sigma_1\beta_1 + \sigma_2\beta_2 + ... + \sigma_i\beta_i + ... + \sigma_{m-1}\beta_{m-1} \quad (4)$$

Here $\beta_i = \arctan 2^{-i}$ and $\sigma_i \in \{-1,1\}$, where i varies from 0 to $m-1$ and m is an integer equal to bit precision. Expanding equation (3) using equation (4) and introducing new parameter(η) for different rotation, leads to the following equation:

$$\begin{bmatrix} x_m \\ y_m \end{bmatrix} = \sum_{i=0}^{m-1} \cos\beta_i \begin{bmatrix} 1 & -\sigma_0\eta 2^{-0} \\ \sigma_0 2^{-0} & 1 \end{bmatrix} ... \begin{bmatrix} 1 & -\sigma_i\eta 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \quad (5)$$
$$... \begin{bmatrix} 1 & -\sigma_{m-1}\eta 2^{-(m-1)} \\ \sigma_{m-1} 2^{-(m-1)} & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

where η is either 1 or 0 based on type of rotation i.e. circular or linear, and $k = \sum_{i=0}^{m-1} \cos\beta_i$ will lead to scale factor, here ignoring the scale factor k in equation (5) which is an amplification factor in spectral domain. Now the equation (5) computes each matrix i.e. each rotation stage (i) with respect to the residue angle given here.

$$z_{i+1} = z_i - \sigma_i\beta_i \quad (6)$$

The initial residue angle is equal to input angle., i.e. $z_0 = \theta$. Putting η = 1 or 0 in equation(5) leads to circular or linear CORDIC [14] respectively. Where $\beta_i = \arctan 2^{-i}$ and $\beta_i = 2^{-i}$ are defined for circular and linear CORDIC respectively. The sign of σ_i for rotation of vectors depends on the sign of the residual angle z_i. Equations (5) and (6) are suitable for pipelined implementation of CORDIC using adders and registers with hard wired shifters as shown in Fig.2. Each stage of pipelined CORDIC for circular and linear CORDIC have

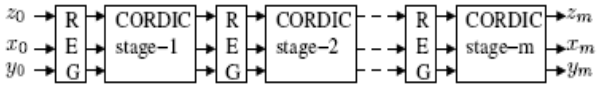
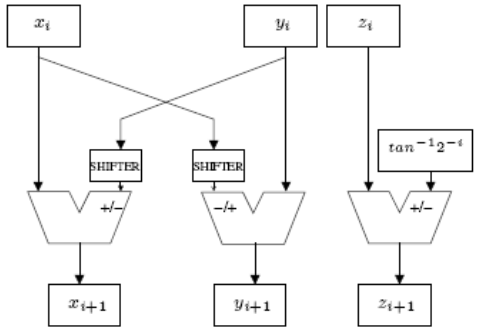
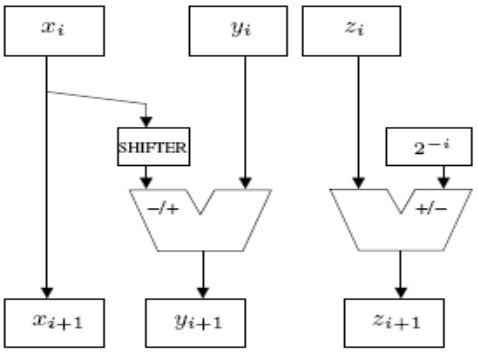


Fig.2. Pipelined CORDIC architecture



(a)



(b)

Fig.3. CORDIC architecture (a) circular (b) linear

been derived using equations (5) and (6) with putting the appropriate value of β_i's and η's. In case of η=1, the scale factor(k) leads to 1.647325 for 16bit precision in circular CORDIC. In case of linear mode, i.e for η =0 the scale factor is equal to unity, i.e. no scaling compensation is required in linear CORDIC. Thus the scale factor compensation is required only in circular CORDIC stage. But this overhead is not required in spectral analysis where the spectral components leads to amplified values equal to this scaled value.

III. PROPOSED ARCHITECTURE

Since the implementation of Blackman windowing function as presented in Section II needs trigonometric computations and multiplications with constant coefficients b_0, b_1 and b_2. So we adopted a CORDIC based parallel and pipelined version [10, 11] for cosine and multiplication functions using circular and linear CORDIC described in last section. The proposed architecture is shown in Fig.4 for VLSI implementation of Blackman windowing function. Major blocks of proposed architecture are described subsequently. Two

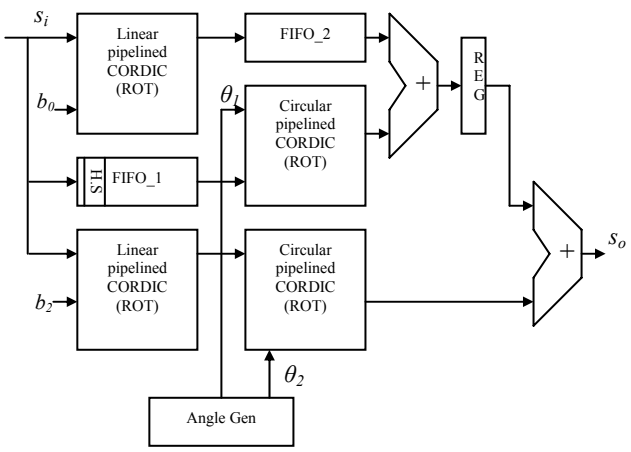


Fig.4. Proposed Blackman windowing architecture

linear CORDIC blocks are used for multiplying input samples with constant coefficients (b_0 and b_2), however the multiplication of constant coefficient (b_1=0.5) with input samples is done with only hard shifter (1-bit right) and passes through FIFO_1 for synchronization with other parallel paths and similarly FIFO_2 is also used for synchronization. Circular CORDIC has been used to compute cosine functions given in equation (2) and multiplication of intermediate values (i.e. values from lower linear CORDIC and FIFO_1 as shown in Fig.4). CORDIC blocks used in our proposed architecture are purely pipelined, where add/sub circuit is the critical path. Here length of FIFOs is equal to the number of stages of pipelined CORDIC minus one, e.g, for 16-bit precision CORDIC, the number of stages are sixteen and thus FIFO length to be fifteen.

A. Angle Generator Architecture

Now the angle generator block shown in Fig.4, generates arguments to cosine functions (i.e. angles $\theta_1 = \frac{2\pi n}{N}$ and $\theta_2 = \frac{4\pi n}{N}$) in equation (2). The sequences of angles θ_1 and θ_2 can be represented by following general expression (7) in discrete form.

$$i.e. \qquad \theta_{n+1} = \theta_n + \frac{2\pi}{N} \qquad (7)$$

with $\theta_0 = 0$ and $n = 0, 1, 2, \ldots, N-1$. This equation is mapped to an architecture which is shown in Fig.5. This architecture generates all the discrete angles ranging from 0 to 2π. Here *AND* gate blocks are used to reset the angle to zero after generating all discrete angles in a window. Consider window (observation) length is $N=2^b$ where b is an integer. The encoder, a combinatorial logic shown in Fig.5, is functionally used to convert N to b, and this b is used in log shift module, which is a logarithmic shifter designed using multiplexers for division of N and HS is a hard shifter for one bit left shift to get θ_2 from θ_1.

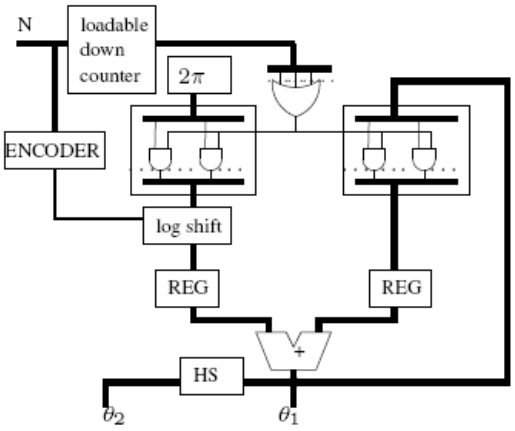


Fig.5. Angle generator architecture

Binary weight ⇨ Quadrant ⇩	-π	π/2	π/4	π/8	----	$\pi/2^{b-1}$
1st	0	0	x	x	----	x
2nd	0	1	x	x	----	x
3rd	1	0	x	x	----	x
4th	1	1	x	x	----	x

Fig.6. Angle representation in four quadrants

Since Blackman windowing function takes angle ranging from 0 to 2π as arguments to cosine function in equation(2), where the constraint in CORDIC algorithm is that it can only compute for angles $\left[\frac{-\pi}{2}, \frac{\pi}{2}\right]$, i.e. only 1st and 4th quadrant angles. Here the normalized angle format as shown in Fig.6, has been used to extend the range $\left[\frac{-\pi}{2}, \frac{\pi}{2}\right]$ to [*0*, *2π*]. This technique is having advantage that quadrant transformation can be achieved with few combinatorial logic considering two most significant bits (MSB) in this angle representation. Here the logic value of "x" presents either "1" or "0" according to value of the angle. Two MSBs "00", "01", "10" and "11" of input angle indicate the 1st, 2nd, 3rd and 4th quadrants respectively, where residue angle *z* lies. XORing two most significant bits flags the quadrant transformation and, angles in 2nd and 3rd quadrant are transformed to 4th and 1st quadrant respectively by replacing first bit(MSB) with second bit (next to MSB). *i.e*. 01→11 and 10→00 where CORDIC can compute the vectors within its range. Finally output vectors are transformed back to their original quadrant by assigning proper sign to the vectors.

B. Han-Carlson Adder Structure

By this time, readers appreciate that pipelined CORDIC modules are main circuits in our proposed architecture, where adder/subtractor circuit lead to critical delay. So we have used a fast adder called Han-Carlson

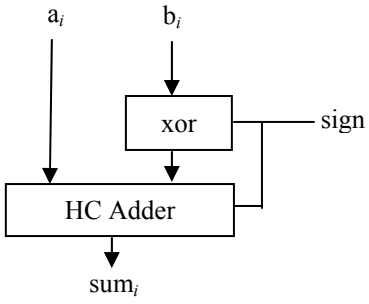


Fig.7. HC Adder/Subtractor

adder [9] for adder/subtractor as shown in Fig.7, to get reduced critical delay compared to conventional adders like RCA, CSA[15,16]. Hence it gives the high throughput result compared to [7] where RCA(ripple carry adder) has been implemented. The HC adder/subtractor for 16-bit, has been presented here for the sake of clarity.

The generalized adder expressions [15] for sum and carry are given here as in equations (8) and (9)

$$sum_i = a_i \wedge b_i \wedge cout_{i-1} \quad (8)$$

$$cout_i = a_i.b_i \mid cout_{i-1}.(a_i \wedge b_i) \quad (9)$$

where $i = 0$ to $m - 1$; m is bit precision of inputs to adder. To make the adder faster, many architectures such as carry select adder and look ahead carry adder [15] are presented. Although all of these cases ripple propagation time decreases compared to conventional ripple carry adder, but hardware increases almost linearly with increasing bit sizes. But for higher bit sizes the delay in pre-fix adders is much less than the other existing fast adders and hardware is also much less than aforesaid adders. Among these aforesaid three fast prefix adders, HC adder structure is optimized in area, timing and routing complexity. So rest of this section describes the HC adder/subtractor as shown in Fig.7.

The HC adder as shown in Fig.8, computes in three steps i.e. *Pre-computation, pre-fix tree network* and *post computation step*. Now the pre-computation step represented with white dots in pre-fix network shown in Fig.9(a), computes two variables g_i and p_i known as carry generate and propagate respectively. These variables are defined as $g_i = a_i.b_i$ and $p_i = a_i \wedge b_i$.

Hence carry out in equation (9) can be rewritten as
$cout_i = g_i \mid p_i .cout_{i-1}$
for example, carry outputs are expressed as
$cout_0 = g_0 \mid p_0 .c_{in}$
$cout_1 = g_1 \mid p_1.cout_0$
$= g_1 \mid p_1.(g_0 \mid p_0.c_{in})$
$= (g_1 \mid p_1 .g_0) \mid (p_1. p_0).c_{in}$
$= G_1 \mid P_1.c_{in}$
Where $G_1 = g_1 \mid p_1 .g_0$ and $P_1 = p_1. p_0$

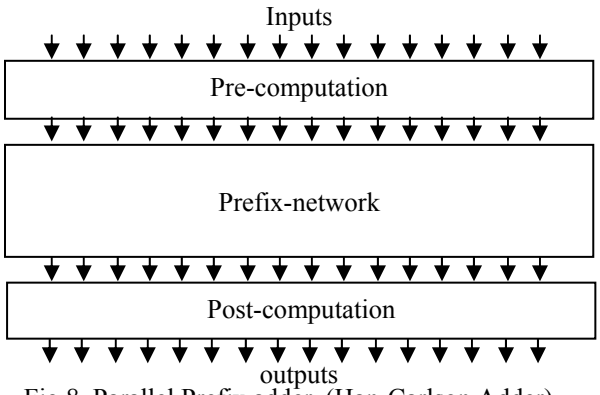


Fig.8. Parallel Prefix adder (Han-Carlson Adder)

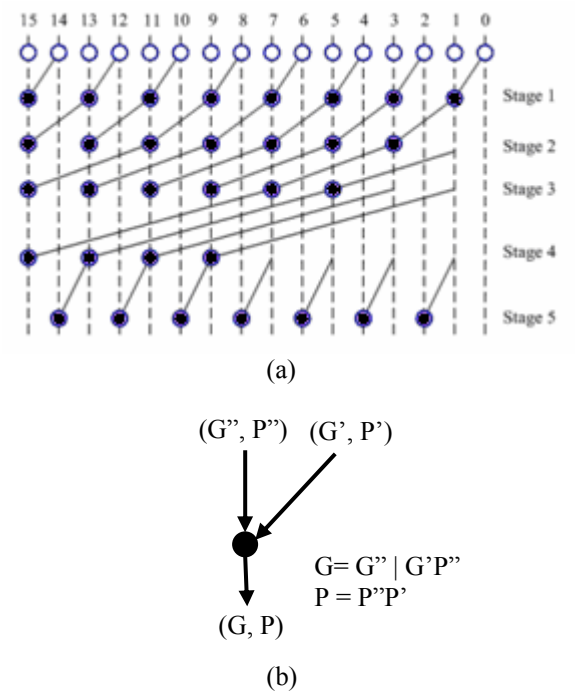


Fig.9. (a) Han-Carlson prefix adder network
(b) Black dot computation

Now prefix-network shown in Fig.9(a), is computed using g_i and p_i as input to it and computes $G_{i,s}$ and $P_{i,s}$ at each stage(s) using black dots (●), where black dots are computed using the Fig.9(b).

The black dot operators in prefix-network is defined as

$$(G,P) = (G'',P'')\bullet(G',P') = (G_i \mid P_i.\ G_{i-1},\ P_i.\ P_{i-1}) \qquad (8)$$

Post-computation stage computes final carry and sum from prefix-network given here.
$cout_i = G_{i,5} \mid P_{i,5}.cin$
$sum_i = a_i \hat{} b_i \hat{} cout_{i-1}$
Here $G_{i,5}$ and $P_{i,5}$ are last stage (in this case 5th stage) output of prefix network.

The advantage of this HC adder circuit is that the delay depends the number of stages in prefix tree, which is equal to log(m) +2, where m is bit precision.

This proposed architecture is having advantages for both fixed and variable window length. In case of fixed windowing, this architecture is better than the ROM based implementation for higher order windowing (observation) length in terms of hardware. ROM based implementation needs ROM as well as multiplier block, where multiplier leads to critical delay. But adder/subtractor block is the critical path in this proposed implementation which leads to higher frequency of operation compared to ROM based implementation. This architecture is an obvious choice for variable length FFT processor, where the length of windowing can be varied either on line or by user according to the specific applications.

IV. RESULTS AND DISCUSSION

The proposed architecture for 16 bit word size has been coded using verilogHDL at structural level and simulated using verilog-XL, and finally synthesized to gate level netlist using commercially available 0.18µm CMOS technology by Synopsys design compiler. It has been found that it can operate at a maximum frequency of 400 MHz. The synthesized result shows that, the chip occupies a core area of 21 mm^2 (approx.). Thus the implementation of this architecture using HC adder enhances the result [7] in terms of throughput. i.e. the throughput of this implementation is 400 Msamples/s. The comparison of synthesized results with [7] is shown in Table.I. The clock latency of this architecture is equal to 33 however it can be reduced to nearly half using alternative CORDIC architectures [17,18] with extra silicon cost.

The parametric comparison of this CORDIC based architecture for Blackman windowing with conventional ROM based implementation is shown in Table-II. Where t_{acc}, t_{mul} and t_{add} are ROM access time, multiplier and adder timing respectively. The windowing coefficients with 16-bit precision are generated for window length of 512 using our proposed architecture, and compared with the coefficients using MATLAB functions. It is found that the maximum absolute error for 16-bit precision and windowing length of 512, is approximately equal to 0.000085.

Table I.
Synthesis result comparision

Parameters	With RCA[7]	With HCA [this work]
Throughput(Msamples/s)	125	400
Area (mm^2)	9	21
Latency (clocks)	33	33

Table II.
Parametric comparison

	ROM based[1]	CORDIC based [Proposed]
Window Size	fixed	flexible
Critical delay	$t_{acc}+t_{mul}+t_{add}$	t_{add}

V. CONCLUSION

This proposed architecture is a parallel and pipelined structure for Blackman windowing, providing a high throughput, area efficient and having flexibility on changing windowing (observation) length for user specific applications, as well modifying the register *N* online with either software protocol or hardware control signal. This architecture can be embedded with advanced FFT processor architectures for real time spectral analysis. This architecture can be used in real time filtering applications, where Blackman windowing filter is required. The technology independent netlist of this proposed architecture can be implemented in advanced FPGA chips.

REFERENCES

[1] Richard J. Higgins, "Digital Signal Processing in VLSI", *Prentice Hall Englewood Cliffs*, 1990.

[2] A.Banerjee, A. S. Dhar and S. Banerjee, "FPGA realization of a CORDIC Based FFT Processor for Biomedical Signal Processing", *Microprocessors and Micro Systems*, vol 25/3, pp131-142, May 2001.

[3] Kai Zhong, Guangxi Zhu, and Hui He, "single-chip, ultra high speed FFT architecture", *Proceedings .5th ASIC, 2003, International Conference*, Vol 2,pp752-756 Oct.2003.

[4] A. Samad, A. Ragoub, M. Othman and Z.A.M.Shariff, "Implementation of high speed Fast Fourier Transform VLSI chip", *Microelectronics Journal*, Vol 29,Issue 11,pp881-887 Nov 1998.

[5] Chung-Ping Hung, Sau-Gee Chen,and Kun-Lung Chen, "Design of an efecient variable length FFT processor", Proc, *Circuits and Systems, ISCAS '04.* Vol 2, pp 833-836, 23-26 May 2004.

[6] Shousheng HE, Torkelson M., "Design and implementation of a 1024-point pipeline FFT processor", *Proc, IEEE-Custom Integrated Circuits Conference,* pp131-134, May 1998.

[7] K. C. Ray and A. S. Dhar, "ASIC Architecture for implementing Blackman windowing for real time spectral analysis", Proc, of *Intl, Conf, on Signal Processing, Communication and Networking*(*IEEE-ICSCN 07*), *pp.388-391*, 22-24th February-2007.

[8] K. C. Ray and A. S. Dhar, "Unified CORDIC based VLSI architecture for implementing windowing functions for real time spectral analysis", ***IEE Proc.-Circuits Devices Syst.***, Vol. 53, No. 6, *pp539-544*, December 2006.

[9] D. Harris, "A Taxonomy of Parallel prefix Networks", Proc, IEEE ACSSC-2003, Vol.2, pp.2213-2217, Nov.2003.

[10] S.Y.Kung, H.J. Whitehouse and T. Kailath, "VLSI and Modern Signal Processing", *Prentice Hall Englewood Cliffs*, 1985.

[11] S. K. Padala and K M M Prabhu, "Pipelined CORDIC Processor for generating Gaussian Random Numbers", *Signal Processing*, Vol 72, No.3, *pp* 177-181, Feb 1999.

[12] J. E. Volder, "The CORDIC Trigonometric Computing Technique", *IRE Computers*, Vol.8, pp 330-334 Sep 1959.

[13] J. S. Walther, "A unified algorithm for elementary functions", Proc, *Spring joint comput, conf*, pp 379-385.1971.

[14] G.L. Haviland and A.A. Tuszynski, " A CORDIC Arithmetic Processor Chip", IEEE Trans. On Comput, Vol. C-29, No.2, Feb1980.

[15] Behrooz Parhami, "Computer Arithmetic algorithms and hardware designs", *Oxford University Press,New York*, 2000.

[16] A. Gyyot, B Hochet and J. M. Muller, "A way to build efficient carry skip Adders", *IEEE Trans.Comput.*,Vol.C-36, no.10, pp1144-1151,Oct 1987.

[17] E. Antelo, J. Villalba, J. D. Bruguera, and E. Zapata, "High performance rotation Architectures based on the Radix-4 CORDIC Algorithm", *IEEE Trans Comput*, Vol.46,No.8, pp855-870, Aug 1997.

[18] J. Duprat and J.M. Muller, "The CORDIC algorithm: new results for fast VLSI implementation", *IEEE Trans. Comput.* Vol.42, No.2, pp.168-178, Feb 1993.

K.C. Ray is presently pursuing his PhD degree in Electronics and Electrical Communication Engineering department. In 2000, He received his M.Tech degree in Electrical Engineering with specialization of power system from Regional Institute of Technology (Presently NIT), Jamshedpur, India. In 1997, He received his Bachelor of Engineering degree in Electrical Engineering from Orissa Engineering College, Bhubaneswar, India.

His research interests include VLSI Signal Processing and CORDIC.

A.S. Dhar received his Bachelor degree in Electronics and Telecommunication Engineering from Bengal Engineering College, Howrah, India in 1987. In 1989, he received his M.Tech degree in Integrated circuits and systems Engineering from Indian Institute of Technology, Kharagpur, India. He received his PhD degree from the same institute in 1994, where he is presently serving as an Associate Professor in the Department of Electronics and Electrical Communication Engineering.

His research interests include VLSI design, CORDIC, DSP architectures and VLSI for Communication.