Volume 2, Issue 2, February 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Performance Analysis of CORDIC Architectures Targeted for FPGA Devices.

Burhan Khurshid*
Dept. of ECE
National Institute of Tech.
Srinagar, J&K India
*khurshid_burhan@yahoo.com

Ghulam Mohd Rather
Dept. of ECE
National Institute of Tech.
Srinagar, J&K India

Hakim Najeeb-ud-din
Dept. of ECE
National Institute of Tech.
Srinagar, J&K India

***Abstract*— Digital Signal Processing domain has long been dominated by software systems; however, the state of art signal processing is now again switching back to hardware based solutions. This requires development of algorithms that can be efficiently implemented on different hardware platforms. CORDIC is one such hardware-efficient algorithm that is used in DSP systems for calculating trigonometric, hyperbolic, logarithmic and other transcendental functions. This paper attempts to explore the different implementations of CORDIC architectures, specific to FPGA devices. The algorithm is implemented in two different styles: folded and unfolded. Unfolded design is improved architecturally by pipelining it. Comparisons are then made between these architectures based on area, speed, throughput and power parameters and logical conclusions are drawn. All three designs have been coded in VHDL and implemented using Xilinx FPGA synthesis tool. To check the functionality of the algorithm each of the designs has been simulated for sine and cosine function evaluations. The simulations are carried out using Xilinx ISim tool and power metrics are obtained using Xilinx Xpower Analyzer tool.**

***Keywords*— CORDIC, FPGA, Rotation mode, Unfolded architecture, Folded architecture.**

I. INTRODUCTION

Digital Signal Processing has many applications such as digital audio broadcast, digital video, multimedia, digital cellular communications, image processing [1] etc. Traditionally dedicated architectures have been used for these DSP applications. These architectures are mostly based on general purpose microprocessors. Advancements such as single cycle multiply-accumulate instructions, special addressing modes, superscalar architectures and VLIW processors has led to the dominance of these general purpose microprocessors in the DSP landscape [2].

Today most of the DSP applications are based on real time multimedia processing. Digital representation of multimedia data can be handled in the same way as text; however the processing rate has to be much faster [1]. On account of this real time throughput constraint, conventional processors are not suitable for modern day DSP systems. Some hardware efficient algorithms are, therefore required for these high speed applications. These algorithms need to be implemented and optimized in hardware so as to enable them to handle real time data while maintaining an optimum trade-off between different performance parameters (area, speed and power). CORDIC is one such algorithm.

CORDIC (COordinate Rotation DIgital Computer) [3, 4] is a hardware efficient shift-and-add algorithm that can be used to
calculate various arithmetic functions. The algorithm has a very simple operation requiring only shift and add operations.

This simplicity in operation has made CORDIC a competitive alternative for evaluating various trigonometric and hyperbolic functions required in many DSP applications. The original algorithm, developed by Jack Volder [5] was limited to trigonometric calculations. John Walther [6] extended the CORDIC theory and made it possible to calculate a large variety of trigonometric and other linear and hyperbolic functions.
FPGAs are often used as co-processors to perform all the high speed tasks that cannot be achieved using conventional processors. Historically, FPGAs have been slower, less energy efficient and generally achieved less functionality than their fixed ASIC counterparts. Advantages include the ability to re-program in the field to

fix bugs, shorter time to market and lower non-recurring engineering costs.

This paper attempts to implement the CORDIC algorithm on FPGA platforms in different styles. The different architectures are compared for various performance parameters and based on these parameters an optimum hardware solution for FPGAs is presented.The rest of the paper is organized in the following manner. Section II discusses the CORDIC algorithm and its operating mode for sine and cosine function evaluation. Section III discusses the folded and unfolded CORDIC architectures. Section IV provides the implementation and simulation results and based on the comparison metrics, performance evaluation of folded and unfolded CORDIC architectures is carried out.

II. CORDIC ALGORITHM

The CORDIC algorithm provides an iterative method of performing vector rotations by arbitrary angles using only shift and add operations [3]. The algorithm, credited to Volder [5], is derived from the general (Givens) rotation transform:

$$x' = x \cos ø - y \sin ø \quad (1)$$

$$y' = x \sin ø + y \cos ø \quad (2)$$

This rotates a vector in a Cartesian plane by the angle *ø*. These can be rearranged so that:

$$x' = \cos ø \ [x - y \tan ø] \quad (3)$$

$$y' = \cos ø \ [y + x \tan ø] \quad (4)$$

The rotation angles are restricted so that, $\tan ø = \pm 2^{-i}$. This reduces the multiplication operation by the tangent term to simple shift operation. Any given target angle *ø* can be decomposed into a sequence of smaller micro rotations. Thus *ø* is decomposed as a sequence of elementary rotations:

$$ø = \Sigma \alpha_i \quad (5)$$

Using these basic ideas we have the basic iterative rotations as:

$$x_{i+1} = \cos \alpha_i \ [x_i - y_i \tan \alpha_i] \quad (6)$$

$$y_{i+1} = \cos \alpha_i \ [y_i + x_i \tan \alpha_i] \quad (7)$$

The rotation angles are restricted so that:

$$\tan \alpha_i = \pm 2^{-i}$$

This assures that the multiplication by the tangent term is reduced to simple shifting operation.

$$x_{i+1} = [x_i - y_i \tan \alpha_i] / (1 + \tan^2 \alpha_i)^{1/2}$$

$$y_{i+1} = [y_i + x_i \tan \alpha_i] / (1 + \tan^2 \alpha_i)^{1/2}$$

Rearranging:

$$x_{i+1} = [x_i - y_i (\pm 2^{-i})] / (1 + 2^{-2i})^{1/2}$$

$$y_{i+1} = [y_i + x_i (\pm 2^{-i})] / (1 + 2^{-2i})^{1/2}$$

Or

$$x_{i+1} = K_i [x_i - y_i . d_i . 2^{-i}] \quad (8)$$

$$y_{i+1} = K_i [y_i + x_i . d_i . 2^{-i}] \quad (9)$$

Where,

$K_i = 1/(1+2^{-2i})^{1/2}$; known as scale constant.

$d_i = \pm 1$; known as decision function.

Removing the scale constant from the iterative equations yields a shift-add algorithm for vector rotation. The product of the K_i's can be applied elsewhere in the system or treated as part of a system processing gain. That product approaches 0.6073 as the number of iterations goes to infinity. Therefore, the rotation algorithm has a gain, *An*, of approximately 1.647. The exact gain depends on the number of iterations, and obeys the relation:

$$A_n = \Pi [1+2^{-2i}]^{1/2}$$

The angle accumulator adds a third difference equation to the CORDIC algorithm:

$$z_{i+1} = z_i - \alpha_i$$

$$z_{i+1} = z_i - d_i \tan^{-1} (2^{-i}) \quad \{ \tan \alpha_i = \pm 2^{-i} \}$$

For a single CORDIC micro-rotation the resulting equations are:

$$x_{i+1} = x_i - y_i . d_i . 2^{-I} \quad (10)$$

$$y_{i+1} = y_i + x_i . d_i . 2^{-I} \quad (11)$$

$$z_{i+1} = z_i - d_i \tan^{-1} (2^{-i}) \quad (12)$$

The CORDIC rotator is normally operated in one of two modes. In rotation mode, the angle accumulator is initialized with the desired rotation angle. The rotation decision at each iteration is made to diminish the magnitude of the residual angle in the angle accumulator. The decision at each iteration is therefore based on the sign of the residual angle after each step. The CORDIC equations are:

$$x_{i+1} = x_i - y_i . d_i . 2^{-i}$$

$$y_{i+1} = y_i + x_i . d_i . 2^{-i}$$

$$z_{i+1} = z_i - d_i \tan^{-1} (2^{-i}),$$

Where,

$$d_i = -1 \text{ if } z_i < 0$$

$$+1, \quad \text{otherwise}$$

After n iterations we are provided with following results:

$$x_n = A_n [\, x_0 \cos z_0 - y_0 \sin z_0 \,] \tag{13}$$

$$y_n = A_n [\, y_0 \cos z_0 + x_0 \sin z_0 \,] \tag{14}$$

$$z_n = 0 \tag{15}$$

Setting the *y* component of the input vector to zero reduces the rotation mode result to:

$$x_n = A_n \,.\, x_0 \cos z_0 \tag{16}$$

$$y_n = A_n \,.\, x_0 \sin z_0 \tag{17}$$

By setting x_0 equal to $1/A_n$, the rotation produces the unscaled sine and cosine of the angle argument z_0.

III. CORDIC ARCHITECTURES

In general, CORDIC architectures can be broadly classified as folded and unfolded, based upon the hardware realization of the three iterative equations [7]. A direct duplication of equations 10, 11 and 12 into hardware results in folded architecture. Folded architectures have to be multiplexed in time domain so that all the iterations are carried out in a single functional unit. This provides a means for trading area for speed [8] in signal processing architectures. One of the widely used folded architectures is implementing the entire CORDIC core using a word serial design.

A. *Folded word serial design*

A folded word serial design [4, 9], also called iterative bit-parallel design is obtained simply by duplicating each of the three difference equations in hardware as shown in figure a.

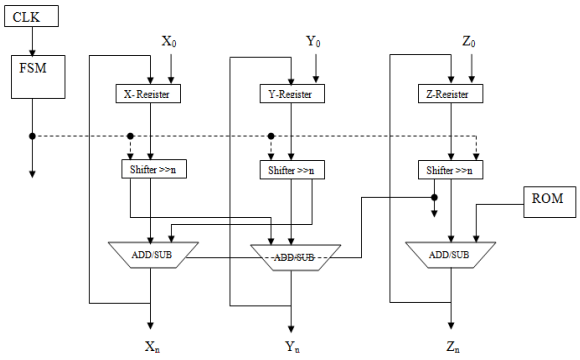


Fig. a folded word serial CORDIC

Being a shift- add algorithm, each individual unit consists of an adder/subtractor unit, a shifter and a register for holding the computed values after each iteration. To start with, the initial values are fed into each branch via a multiplexer. The value in the *z* branch determines the operation of the adder-subtractor unit. Signals in the *x* and *y* branch pass through the shifter units and are then added to (or subtracted from) the unshifted signal in the opposite path. The *z* branch arithmetically combines the register values with the values taken from a lookup table whose address is changed according to the number of iteration. The result of this operation determines the nature of operation for the next iteration. After *n* iterations the results are directly read from the adder/subtractor units. A finite state machine is used to keep a track of shifting distances and the ROM addresses. Since the adder/subtractor unit and the shifters in each path are shared on time basis this conventional approach of implementing the CORDIC algorithm is not suitable for high speed applications [4]. Another disadvantage is with respect to the shift operations. When implemented in hardware the shifters have to change the shift distance with the number of iteration. For large number of iterations these require a high fan in and reduce the maximum speed for the application [2, 4].These shifters do not map well into FPGA architectures and if implemented require several layers of logic. The result is a slow design that uses large number of logic cells. In addition the output rate is also limited by the fact that the operation is performed iteratively and therefore the maximum output rate equals *1/n* times the clock rate, where *n* is the number of iterations.

B. *Unfolded parallel design*

The iterative nature of the CORDIC processor discussed above demands that the processor has to perform iterations at *n* times the data rate. The iteration process can be unfolded [9, 10] so that each of *n* processing elements always performs the same iteration. A direct application of the unfolding transformation is to design parallel processing architectures from serial processing architectures. At the word level, this means that word-parallel architectures can be designed from word-serial architectures [1]. An unfolded CORDIC processor is shown in figure b

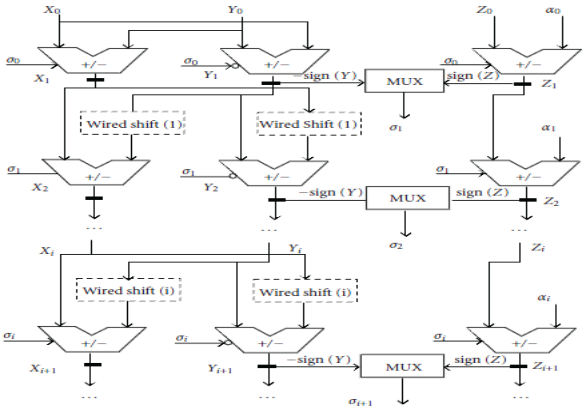


Fig. b Unfolded CORDIC design

Unfolding the CORDIC processor results in two significant changes. First, the shifter in each unit is of fixed shift i.e. it has to perform a constant shifting operation in each stage. Thus the shifter needs not to be updated as in the iterative structure. This makes their

implementation in FPGAs quite feasible. Second, the unfolding process eliminates the use of ROM from the processor which was required to hold the constant angle values during each iteration. Those constants can be hardwired instead of requiring storage space. The entire CORDIC processor is thus reduced to an array of interconnected adder- subtraction units. The need for registers is also eliminated, making the unrolled processor strictly combinatorial. Another advantage of the unrolled design is that the processor can be easily pipelined [11] by inserting registers between the adder-subtraction units. In the case of most FPGA architectures there are already registers present in each logic cell, so the addition of the pipeline registers has no additional hardware cost.

IV. IMPLEMENTATION AND RESULTS

A. Methodology

The CORDIC processor is implemented in seven stages and for a word length of 16 and 32 bits. The initial design entry is done using VHDL. The design translation is carried out in Xilinx ISE 12.4 [12]. The simulator database is then analyzed for different performance parameters and logical conclusions are drawn. The core is implemented with the following synthesis description:

Platform: FPGA
Family: Virtex5
Target device: XC5VLX30
Package: FF324

Figure c shows the generated RTL schematic of the folded CORDIC for one iteration. Figure d shows the RTL schematic for one stage of unfolded CORDIC.

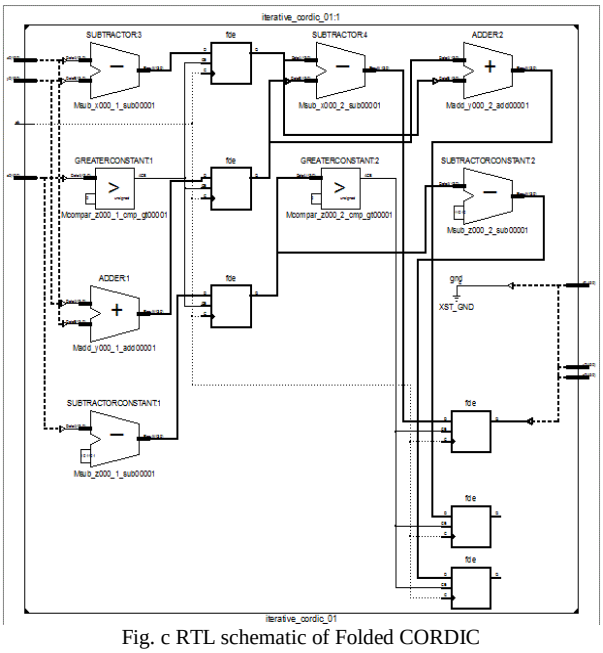


Fig. c RTL schematic of Folded CORDIC

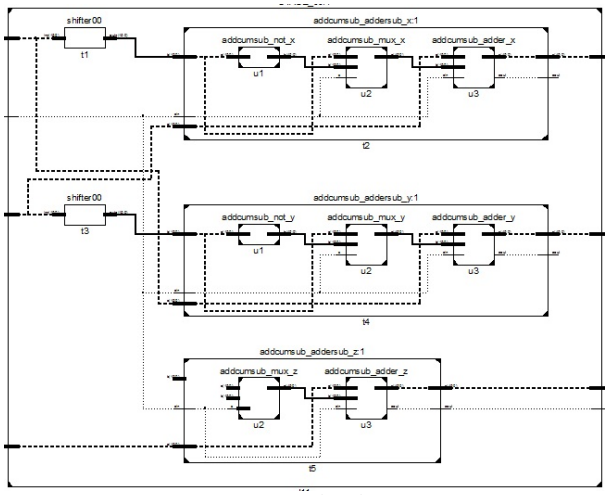


Fig. d RTL schematic of Unfolded CORDIC

B. Simulations

The generated core has been simulated for sine and cosine functions by operating it in the rotation mode. Figure e shows the simulated sine and cosine values of certain angles calculated using 16-bit iterative CORDIC. Figure f and figure g shows the simulated sine and cosine values calculated using parallel and pipelined designs respectively.

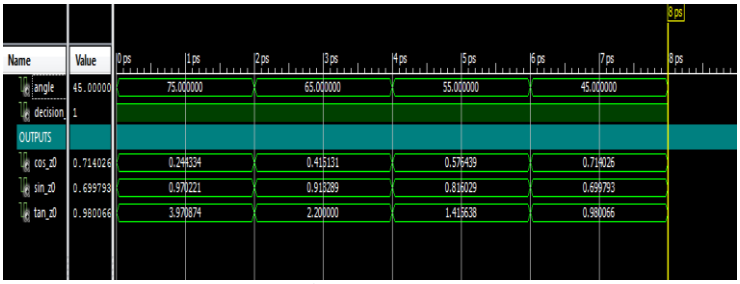


Fig. e Simulation result for 16-bit Folded CORDIC

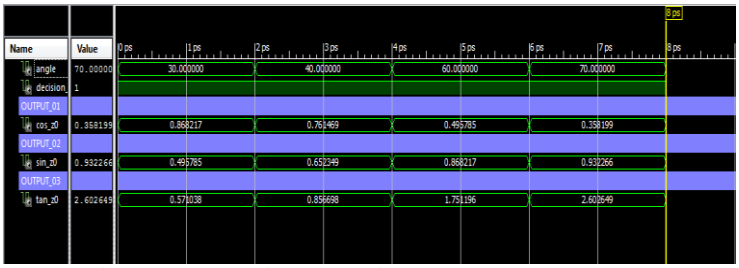


Fig. f Simulation result for 16-bit Unfolded parallel CORDIC

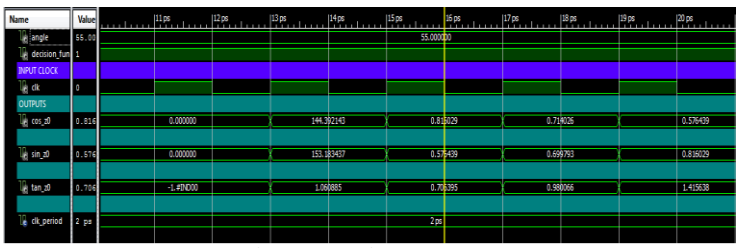


Fig. g Simulation result for 16-bit Unfolded pipelined CORDIC

C. Analysis and Results

The folded and unfolded structures are analyzed for different performance parameters. Table 1 provides latency

comparison for the two structures. All the structures are implemented with the same synthesis description.

TABLE 1 LATENCY COMPARISON FOR16 AND 32-BIT CORDIC

Parameter	CORDIC architectures.			
	Folded		Unfolded	
	16 Bit	32 Bit	16 Bit	32 Bit
Logic delay.	5.594ns	6.959ns	5.804 ns	9.18 ns
Route delay.	25.023ns	33.071ns	18.69 ns	25.0 ns
Max. Combinational delay.	33.078ns	42.414ns	24.472ns	34.2 ns

Table 2 gives the maximum operating frequency comparison of the folded, unfolded and pipelined structures for word lengths of 16 and 32 bits.

TABLE 2 THROUGHOUT COMPARISON FOR16 AND 32-BIT CORDIC

Parameter	CORDIC architectures.					
	Folded		Unfolded (parallel)		Unfolded (pipelined)	
	16 Bit	32 Bit	16 Bit	32 Bit	16 Bit	32 Bit
Max. operating frequency	216.57MHz	125.27 MHz	44.29 MHz	31.67 MHz	232.6 MHz	163.43 MHz

It is observed that when timing response of the CORDIC structures is concerned, the unfolded architecture has less worst-case delay compared to the folded structure. This is due to the unfolding process which eliminates the use of storage registers and thus the corresponding set-up and hold times. The overall latency is thus reduced by a factor proportional to these set-up and hold times. Note, however that the maximum operating frequency and thus the throughput of the unfolded CORDIC is determined by the worst case delay of the structure. This is because the structure is purely combinatorial. Contrast to this, the folded structure can be clocked at high frequencies resulting in large operating frequencies. However, pipelining the unfolded CORDIC makes it possible to process multiple inputs simultaneously, thereby increasing the maximum operating frequency of the unfolded structures. For an N stage CORDIC core, N stage pipeline can give maximum result. The first output of an N-stage pipelined CORDIC core is obtained after N clock cycles. Thereafter, outputs will be generated after every clock cycle. Further analysis of CORDIC is carried out by comparing the power consumption for 32 bit word length. Table 3 gives the power consumption for the three structures.

TABLE 3 POWER COMPARISON FOR 32-BIT CORDIC

Instance (resource)	CORDIC architectures.		
	Folded	Unfolded (parallel)	Folded (pipelined)
power (clock)	21.32 mW	--	17.75 mW
power (logic)	2.15 mW	13.87 mW	9.20 mW
power (signals)	15.71 mW	11.01 mW	12.33 mW
power (IOs)	93.60 mW	196.07 mW	196.64 mW
power (leakage)/ quiescent	380.99 mW	382.30 mW	382.21 mW
dynamic power	132.78 mW	220.95mW	235.92 mW
total power dissipation	513.77 mW	603.25 mW	618.13 mW

Folded structures have less power dissipation compared to the parallel and pipelined structures. The power consumed by logical components in case of folded structures is quite low. This is due to the fact that the folded structure uses the same components repetitively. Similarly due to the multiple input/output instantiations in unfolded structures the power consumed by the input and output resources is quite high resulting in high dynamic power dissipation in the parallel and pipelined designs. Finally the three designs are analyzed for area consumption in terms of resource utilization and the results are tabulated in table 4 below

TABLE 4 AREA COMPARISON FOR 32-BIT CORDIC

parameter	CORDIC architectures.		
	Folded	Unfolded (parallel)	Folded (pipelined)
No. of Registers	768	--	678
No. of LUTs	287	1093	1006
No. of logic blocks used	285	1093	1006
No. of occupied Slices	121	589	336
No. of LUT Flip Flop pairs used	768	1093	1013
No. of bonded IOBs	193	193	194

As expected, the folded structure is an efficient user of logic since the same logical units are used over every iteration. But since the results need to be fed back after every iteration a large number of registers are used in the folded word serial implementation.

V. CONCLUSION

This paper carried out the performance analysis of the folded and unfolded CORDIC architectures. The implementation was targeted for FPGA devices. The resulting structures showed differences in the way of using resources available in the target FPGA device. The unfolded and fully pipelined design uses the resources extensively but shows the best latency per sample and thus maximum throughput rate. The folded word serial design uses less on-chip resources but has a large latency per sample. Thus these are not suitable for high speed DSP applications. To sum up, a judicious trade-off between area, power and throughput parameters, and the intended application will determine the correct approach for implementing the CORDIC algorithm. Moreover the selected approach will have no effect on the precision of the results, as the precision is a function of number of iterations (in case of folded design) or number of stages (in case of unfolded design) in the CORDIC core and not the approach used to implement the core.

ACKNOWLEDGEMENTS

This work has been carried out in SMDP-II VLSI laboratory of the Electronics and Communication Engineering Department, of National Institute of Technology Srinagar, India. This SMDP – II VLSI project is funded by Ministry of Communication and Information Technology, Government of India. Authors are grateful to the Ministry for the facilities provided under this project.

REFERENCES

[1] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, Wiley, 1999.

[2] R.Andraka, "A survey of CORDIC algorithms for FPGA based computers," FPGA '98, in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp 191-200, 1998.

[3] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electronic Computing*, volume EC-8, pp 330 – 334, 1959.

[4] J.S. Walther, "A unified algorithm for elementary functions," *Proc. Spring. Joint Comp. Conf.*, vol. 38, pp. 379-385, 1971.

[5] E. Deprettere, P. Dewilde, and R. Udo, "Pipelined CORDIC Architecture for Fast VLSI Filtering and Array Processing," *Proc. ICASSP'84*, 1984, pp. 41.A.6.1- 41.A.6.4.

[6] H.M. Ahmed, J.M. Delosme and M. Morf, "Highly concurrent computing structures for matrix arithmetic and signal processing," *IEEE Computer magazine*, vol. 15, pp. 65–82, 1982.

[7] J. E. Meggitt, "Pseudo division and pseudo multiplication processes," *IBM Journal*, vol. 6, no. 2, pp. 210–226, 1962.

[8] C.H.Lin and A.Y. Wu, "Algorithm and Architecture for High-Performance Vector Rotational DSP Applications," *Regular IEEE Transactions: Circuits and Systems I,* Volume 52, pp 2385- 2398, November 2005.

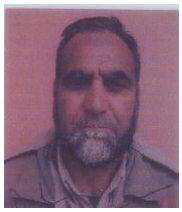
[9] M.D. Erecegovac and T. Lang, *Digital Arithmetic*, Elsevier, Amsterdam, the Netherlands, 2004.

[10] Y.H. Hu, "Pipelined CORDIC architecture for the implementation of rotational based algorithm," in *Proceedings of the International Symposium on VLSI Technology, Systems and Applications*, p. 259, May 1985.

[11] A.A. De Lange, A.J. Van der Hoeven, E.F. Deprettere, and J. Bu, " An optimal floating-point pipeline CMOS CORDIC Processor," *IEEE ISCAS'88*, pp. 2043-47, 1988.

[12] ISE Simulator, Xilinx incorporation San Jose U.S.A, 2011.

Burhan Khurshid received the B.E. degree in Electronics and Communications Engineering from the Kashmir University, India, in 2008, the M.Tech degree in Communications and IT from National Institute of Technology, Srinagar, India in 2011. His research interests are in the field of Reconfigurable and DSP Design (system level) using VLSI.



G.M. Rather received the B.E. degree in Electronics and Communications Engineering from the Kashmir University, India, in 1981, the M. S. degree (1988) in Computer Communications and the Ph.D. degree (1997) from the Indian Institute of Sciences Bangalore India. He is currently Professor in the Department of Electronics and Communication Engineering, National Institute of Technology, Srinagar, India. His research interests are in the field of Communications and DSP Design using VLSI. He is a member of IETE India.

Najeeb-ud-din received the B.E. degree in Electronics and Communications Engineering from the Kashmir University, India, in 1985, the M. Eng. degree in Solid-state Electronics from the University of Roorkee, India, and the Ph.D. degree from the Indian Institute of Technology (IIT), Bombay, India, in 2003. He is currently Associate Professor in the Department of Electronics and Communication Engineering, National Institute of Technology, Srinagar, India. His research interests are in the field of SOI, CMOS Devices, Design, and Technology; in mixed-signal applications. He is a Senior Member of IEEE.