Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 30 (2012) 519 – 528

Procedia Engineering

www.elsevier.com/locate/procedia

International Conference on Communication Technology and System Design 2011

A Novel Method for Computing Exponential Function Using CORDIC Algorithm

J. Sudha[a], M. C Hanumantharaju[b], V. Venkateswarulu[a], Jayalaxmi H[c], a*

[a]*Department of PG Studies,VTU Extension Center,UTL Technologies,Bangalore, India - 560022*
[b]*Department of Information Science & Engineering,Dayananda Sagar College of Engineering, Kumaraswamy Layout, Bangalore, India – 560078*
[c]*Department of Electronics and Communication Engineering,Acharya Institute of Technology, Bangalore, India -560090*

Abstract

In this paper, design of Coordinate Rotation Digital Computer (CORDIC) based 2D Gaussian function and an efficient Very Large Scale Integration (VLSI) architecture suitable for Field Programmable Gate Array (FPGA) implementation is presented. The potential application of the proposed 2D Gaussian function includes image enhancement, smoothing, edge detection, filtering etc. The proposed algorithm uses expanded range of CORDIC algorithm since the conventional technique converges within the range of [-1.12, +1.12]. The architectures developed exploits high degrees of pipelining and parallel processing in order to achieve real time performance. The design has been realized using Register Transfer Language (RTL) compliant Verilog code and fits into a single chip with a gate count utilization of 75,151. The algorithm has been tested for Gaussian based image smoothening application and is implemented on XC2V1000-6bg575 Xilinx FPGA device. The architecture developed is capable of processing one pixel per clock cycle and provides results in real time. The experimental results shows that the proposed approach exhibits better performance when compared with the other researcher methods.

Keywords : Hyperbolic CORDIC Algorithm; 2D Gaussian Function; Exponential Function; FPGA;

1. Introduction

The CORDIC algorithm is a basic iterative algorithm which uses a fixed vector rotation method in order to evaluate the trigonometric functions. This algorithm simplifies hardware implementation process since the CORDIC uses only shift and add operations. The CORDIC algorithm has a past history of more than fifty years since then it has been used in diverse fields. The CORDIC algorithm was initially proposed by J. E Volder [1] for basic elementary mathematical functions such as multiplication, division, and trigonometric functions. CORDIC provides a unified approach [2] for most of the trigonometric functions

* J.Sudha. Tel.: +91-080-23426103;
E-mail address: sudhaj@acharya.ac.in

which have been used in various fields [3]. CORDIC has been widely used in applications such as signal processing, image processing [4], video processing, MIMO [5] and neural networks etc. With the advent of increase in real time Digital Signal Processing (DSP) related applications software implementation of algorithms has led to reduced speed, increase in chip area and high power consumption. Hu et al. [3] has proposed review of architecture for DSP applications such as Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT) and Discrete Hartley Transforms (DHT) etc. based on CORDIC algorithm.

The CORDIC algorithm provides hardware efficient solution since the multipliers are replaced by adders and shifters which in turn reduce the gate count, computational complexity, hardware cost and thereby improves speed. The conventional polynomial approximation scheme is more complex and less likely to be used in real time applications for evaluating trigonometric functions. The alternate Look Up Table (LUT) technique requires huge memory and not worthwhile to be used in this era of nano technology. However, polynomial and LUT based techniques are better in terms of performance as compared with software implementation approaches. This paper describes the basic CORDIC algorithm and its application for the implementation of expanded range of hyperbolic function. Further, we use hyperbolic function in order to evaluate exponential function and design of 2D-Gaussian function. The 2D-Gaussian function is used widely in the area of image and video processing applications. In addition, Gaussian function finds its application in low pass filtering, denoising, image deblurring, edge detection, image scaling and image smoothing.

The fixed point implementation is emphasised in this paper since the digital VLSI implementations of DSP algorithms rely on fixed-point approximations. In addition, fixed point implementation reduces the cost of hardware with increase in throughput rate. The floating point representation has limited number of bits used for mantissa and exponent which causes numerical errors either due to rounding off of least-significant-bit (LSB) of the mantissa or due to the saturation of the exponent [6, 7]. Further the floating point implementation consumes enormous amount of FPGA resources, hence not used in FPGA implementation. The expansion scheme of the hyperbolic CORDIC algorithm has been proposed by Hu et al. [8]. The expansion scheme assists to improve the limited range of convergence by increasing the number of iterations in the negative indices. The expansion scheme overcomes the limitation of conventional method whose range of convergence is [-1, +1]. The proposed approach extends the range of convergence based on the iterations.

The CORDIC algorithm is based on two operating modes namely: rotation mode and vectoring mode. These modes are used to convert polar to rectangular coordinate system. In rotation mode the input vector is rotated by specific predetermined angles. The input vector is rotated, so that the summation of the angles rotated is equal to the desired angle to be computed. The angle accumulator holds the initial angle which slowly converges to 0. Finally the vector coordinates sinh and cosh functions are obtained. In vectoring mode the rectangular coordinates is converted to polar coordinates. The angle accumulator is initialised to zero and the summation of the angles rotated is equal to the desired angle to be computed. Further, the y component converges to zero at every iteration.

The most important part of our paper is the implementation of the expanded hyperbolic CORDIC architecture on FPGA. Further, we use the proposed approach in the design and implementation of 2D Gaussian function. Numerous researchers have proposed different CORDIC architectural designs namely bit serial, bit parallel, word serial or parallel, pipelined, unrolled [2, 10] etc. With the trade-off in area, speed, through put rates an optimum architecture that is a parallel and a pipelined architecture design has been proposed in this paper. The Parallel and a pipelined architecture technique adopted improves the speed and throughput rate [7, 9].

This paper is organised as follows. Section 2 describes the existing works of CORDIC algorithm. The section 3 describes the basic and expanded range CORDIC algorithm. The design of 2D Gaussian function is provided in section 4. In section 5 the architecture for hyperbolic function is presented. The experimental results and comparative study is provided in section 6. Finally, section 7 concludes this paper.

2. Literature Survey

Ray Andraka [10] has summarised the implementation of functions such as sine, cosine, tangent, circular, linear, hyperbolic functions, cartesian to polar transformations and inverse functions by using CORDIC algorithm. Further this paper also describes bit serial, iterative and online CORDIC architecture design for FPGA Implementation. Kaushik Bhattacharya et al. [12] have proposed radix 4 CORDIC algorithm to speed up the conventional algorithm. This scheme helps to reduce the number of iterations. The parallel and pipelined architecture design adopted in order to optimise the area and speed of the processor. Hu et al. [8] has proposed a unified CORDIC algorithm based on the parallel and pipelined CORDIC processor. This paper describes the implementation of CORDIC algorithm for various DSP algorithms such as Discrete Hartely Transforms (DHT), Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT), Kalman Filtering, Chirp Z Transform (CZT), ODF, Eigen Value and Singular value decomposition, and QR factorisation etc.

Lakshmi et al. [9] has discussed the comparison of various CORDIC techniques that are used to improve the computational speed and reduce the area. This paper emphasises the fact that the higher radix serves to improve the speed. A scale factor compensation technique is exploited in order to scale the final coordinates and reduce the overhead on CORDIC algorithm. Further this paper describes the technique to improve the convergence by using rounding error and angle approximation error. Llamocca et al. [7] paper discusses the fixed point implementation of hyperbolic CORDIC algorithm. This paper describes the expansion scheme of CORDIC algorithm. Further, the implementation of the algorithm on three architectural designs such as iterative, pipelined and bit serial has been described. Velmurugan [11] proposed a low power analog and mixed mode implementation of particle filter for target tracking. The particle filter algorithm used exploits addition, multiplication, Gaussian and arctan calculations based on CORDIC algorithm.

3. The Basic Cordic Algorithm

Volder [1] implemented the basic CORDIC algorithm for multiplication, division, conversion of binary to decimal and mixed radix number systems. Walther [2] generalised the techniques proposed by Volder in order to compute hyperbolic, exponential, logarithm and square root functions. Numerous researchers have been contributed for the CORDIC implementation with different applications. In this paper, FPGA implementation of expanded CORDIC hyperbolic and it application to design and implementation of 2D Gaussian function has been proposed. The CORDIC algorithm is an iterative technique, as it rotates the basic vector by small angles which are pre computed. The whole concepts revolve around using a simple shifter and adder for the implementation of the CORDIC algorithm in order to compute complex functions.

A vector V(x_0, y_0) is initially chosen by trial and error operation, the vector is extended by an angle theta which has to be computed. The vector is rotated by some small angles called ϕ, the summation of these small angles results in the angle theta that has to be pre computed. A new vector V' (x', y') is thus obtained at every iteration. In addition, the new vector gets scaled by a constant value. The CORDIC iterative equations is expressed as

$$\begin{aligned} x' &= x\cos\phi - y\sin\phi \\ y' &= y\cos\phi + x\sin\phi \end{aligned} \tag{1}$$

Further the above equations are simplified by factoring out cosϕ term

$$x' = \cos\phi(x - y\tan\phi)$$
$$y' = \cos\phi(y - x\tan\phi) \tag{2}$$

The angles are chosen in terms of 2^{-i} in order to avoid tangent term. The rearranged equations are expressed as follows

$$x_{i+1}=k_i(x_i - y_i d_i 2^{-i})$$
$$y_{i+1}=k_i(y_i + x_i d_i 2^{-i}) \tag{3}$$

where i is the number of iterations, k_i is a constant and d_i the direction of rotation. A new equation based on z_i is introduced in order to keep track of rotation angle

$$Z_{i+1}=Z_i-d_i \quad {}_i \tag{4}$$

where $_i$ =$\tan^{-1}2^{-i}$ is the pre computed angle and is stored in the look up table.

The vectors are rotated in two modes namely, vector mode and rotation mode. In vector mode, the vector is rotated so that the y value approaches to zero and finally, we arrive at our desired angle. However, in the rotation mode the vector is rotated by small angles. The summation of all angles will arrive at our desired angle. The final coordinates are obtained if z_i equals zero. The CORDIC hyperbolic equations is

$$x_{i+1}=k_i(x_i - \mu_i y_i d_i 2^{-i})$$
$$y_{i+1}=k_i(y_i + x_i d_i 2^{-i}) \tag{5}$$
$$z_{i+1} = z_i - d_i\phi_i$$

Where $_i$ =$\tanh^{-1} 2^{-i}$, μ = -1 for hyperbolic equations i represent the iterations (i = 1, 2, 3...N). For hyperbolic equations the iterations (4, 13, 40......k, 3k+1) has to be repeated. After n iterations the generalised hyperbolic equation can expressed as below where $A_n = \prod_{i=1}^{n}\sqrt{1-2^{-2i}}$

$$x_n = A_n(x\cosh z + y\sinh z)$$
$$y_n = A_n(y\cosh z + x\sinh z) \tag{6}$$
$$z_n = 0$$

The initial values of x and y are chosen by trial and error process in order to reduce error. Therefore, it is possible to evaluate sinh, cosh, tanh, tanh-1, exponential and Gaussian function. In this paper expanded range of convergence [7, 8] is adopted i. e. the range used in the proposed method is (-12, +12). The basic algorithm is modified for extending the range for both positive and negative values of i as shown below.

for $i \le 0$:

$$\phi_i = \tanh^{-1}\left(1-2^{i-2}\right)$$
$$x_{i+1} = x_i + d_i(1-2^{i-2})y_i$$
$$y_{i+1} = y_i + d_i(1-2^{i-2})\, x_i$$
$$z_{i+1} = z_i - d_i\tanh^{-1}\left(1-2^{i-2}\right) \tag{7}$$

for $i > 0$:

$$\begin{aligned} &\phi_i = \tanh^{-1} 2^{-i} \\ &x_{i+1} = k_i (x_i + y_i d_i 2^{-i}) \\ &y_{i+1} = k_i (y_i + x_i d_i 2^{-i}) \\ &z_{i+1} = z_i - d_i \phi_i \end{aligned} \tag{8}$$

where $A_n = \prod_{i=-M}^{0} \sqrt{1-(1-2^{(i-2)^2}} \prod_{i=1}^{N} \sqrt{1-2^{-2i}}$

We thus are able to calculate sinh, cosh, tanh, $\tanh^{-1}$, exp and thereby Gaussian function.

$$\begin{aligned} &x_n = A_n \cosh z \\ &y_n = A_n \sinh z \\ &e^x = \cosh z + \sinh z \end{aligned} \tag{9}$$

4. The 2D Gaussian Function

The Gaussian 2D distribution function is used in image smoothing application since Gaussian function is a point spread function. The Gaussian function is used improve the fine details of an image. It filters out high frequency noise in the image. The Gaussian based image enhancement method is found to be one of the best image enhancement methods among various image enhancement techniques. The 2D Gaussian distribution function can be expressed as

$$F_n(x, y) = K_n \times e^{\frac{-(x^2+y^2)}{2\times\sigma_n^2}} \tag{10}$$

where x, y are spatial coordinates and K_n is expressed as

$K_n = \dfrac{1}{\sum_{i=1}^{p} \sum_{j=1}^{q} e^{\frac{-(x^2+y^2)}{2\times\sigma_n^2}}}$, where n = 1, 2, 3; p and q represents the image size

Here σ is the standard deviation of the Gaussian distribution. The Gaussian function uses a kernel to

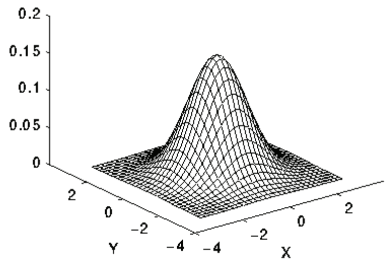


Fig. 1. Two Dimensional Gaussian Distribution Function

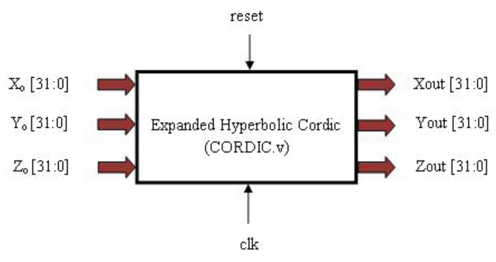


Fig. 2. Top Level Signal Diagram of Expanded CORDIC Hyperbolic Function Module

represents a bell shaped curve. The closeness of the curve to the centre of the bell is determined by σ. The curve maximises the pixel values at the centre and minimises it at the tails. Thereby helps to improve the features required in the image. The image pixels are convolved with the Gaussian kernel. Each pixels new value is the weighted average of neighbouring pixels. Thus helps in preserving edges and improving the features in the image. The Gaussian function is also used in down sampling an image. The 2D Gaussian curve is shown in Fig. 1.

5. Architecture of Expanded Range CORDIC Hyperbolic Function

The signal diagram for expanded range CORDIC hyperbolic function is shown in Fig. 2. The signal diagram performs the actual CORDIC algorithm operations. The architecture is designed for a parallel and a pipelined structure in order to improve the speed of operation. The parallel and pipeline structured can perform a CORDIC transformation in every clock cycle, producing a new output at a new cycle. The scaled data input is given to the block, the data is scaled by a factor to obtain the result up to 10 decimal places. The CORDIC unit is reset by a "reset_n" signal.

The 32 bit scaled data input X_0, Y_0 and Z_0 is supplied to the block after the pre determined iterations. The output obtained is a 32 bit scaled value. The outputs X_{out}, Y_{out} and Z_{out} are obtained after 450 ns with a clock frequency of 344.94 MHz. A pipelined and a parallel architecture design are proposed as it offers a high throughput rate and a good latency [7, 13, and 14]. Further the proposed implementation accepts direct word as input, hardwired constant angles fixed shifters and registers. The design helps to overcome looping involved in iterative architectures. The pipelined architecture proposed for the implementation of 32 bit word length expanded range CORDIC hyperbolic function is shown in Fig. 3. The proposed scheme implements the unified algorithm based on equations (7) to (8). In the proposed approach the hardware requirement for the architecture design is minimal. At each iteration X, Y and Z data values enters the registers, Look-Up Tables (LUT) values are hardwired across the architecture. A multiplexer is used to determine the addition or subtract operation. In addition, the output of one stage forms input to another stage.

6. Experimental Results and Discussions

The expanded hyperbolic CORDIC algorithm and its application as 2D Gaussian function design is initially implemented in Matlab (R2008a) in order to verify the concept. The hardware realization of the algorithm is based on Register Transfer Logic (RTL) compliant Verilog code. The algorithm is

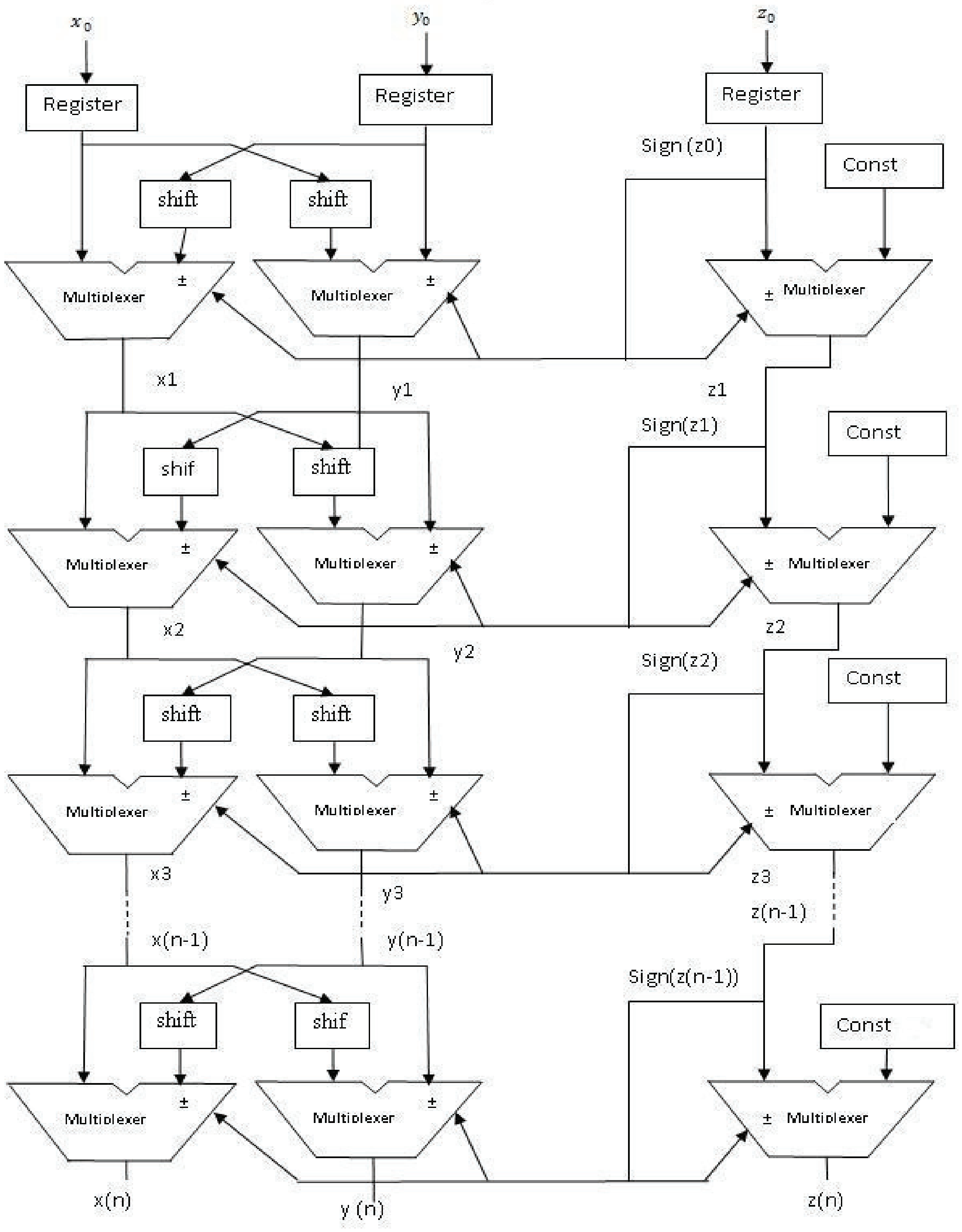
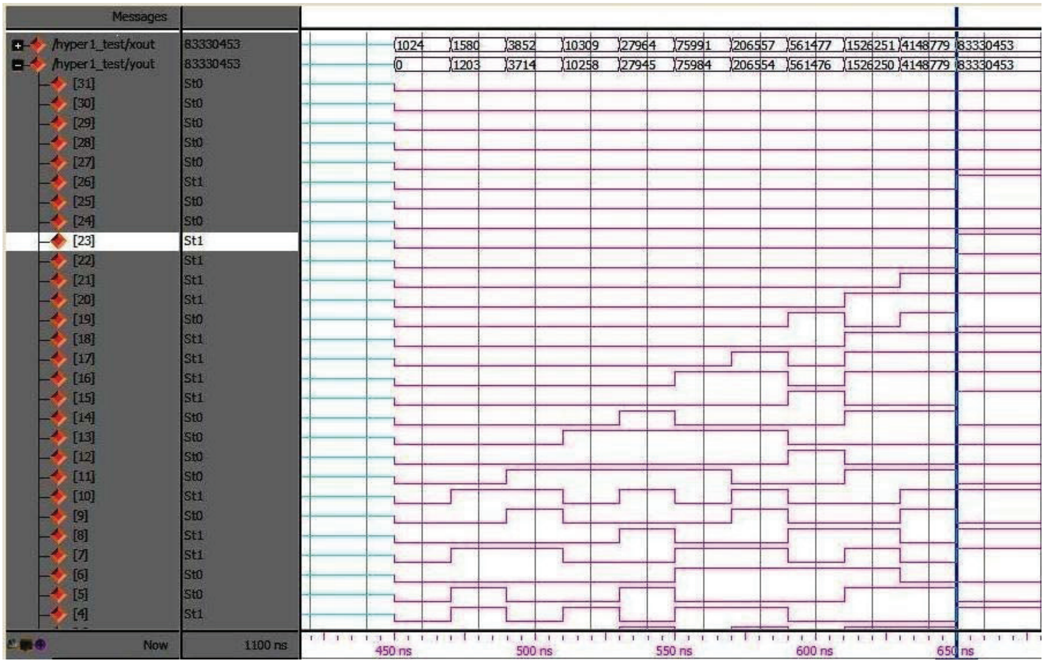
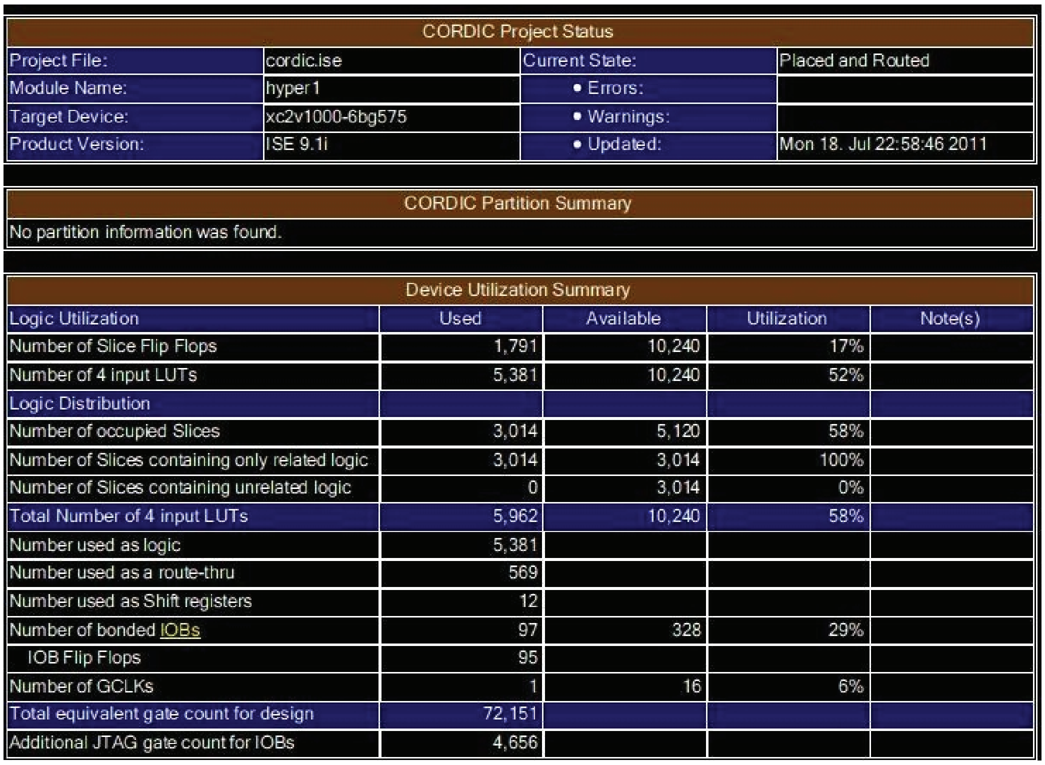


Fig. 3. Pipelined Architecture for Expanded Range CORDIC Hyperbolic Function


```
Messages
/hyper1_test/xout   83330453   1024    1580  3852  10309  27964  75991  206557  561477  1526251  4148779  83330453
/hyper1_test/yout   83330453   0       1203  3714  10258  27945  75984  206554  561476  1526250  4148779  83330453
  [31]  St0
  [30]  St0
  [29]  St0
  [28]  St0
  [27]  St0
  [26]  St1
  [25]  St0
  [24]  St0
  [23]  St1
  [22]  St1
  [21]  St1
  [20]  St1
  [19]  St0
  [18]  St1
  [17]  St1
  [16]  St1
  [15]  St1
  [14]  St0
  [13]  St0
  [12]  St0
  [11]  St0
  [10]  St1
  [9]   St0
  [8]   St1
  [7]   St1
  [6]   St0
  [5]   St0
  [4]   St1
Now  1100 ns        450 ns   500 ns   550 ns   600 ns   650 ns
```

(a) Waveforms for Sinh and Cosh Function: Start of Computation



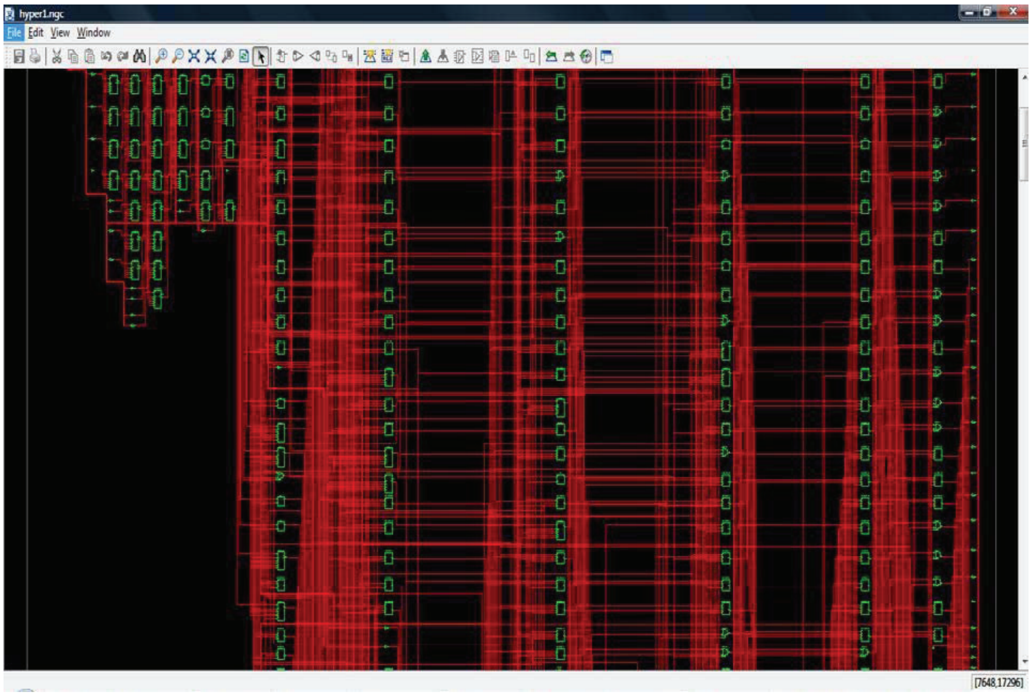
CORDIC Project Status			
Project File:	cordic.ise	Current State:	Placed and Routed
Module Name:	hyper1	• Errors:	
Target Device:	xc2v1000-6bg575	• Warnings:	
Product Version:	ISE 9.1i	• Updated:	Mon 18. Jul 22:58:46 2011

CORDIC Partition Summary
No partition information was found.

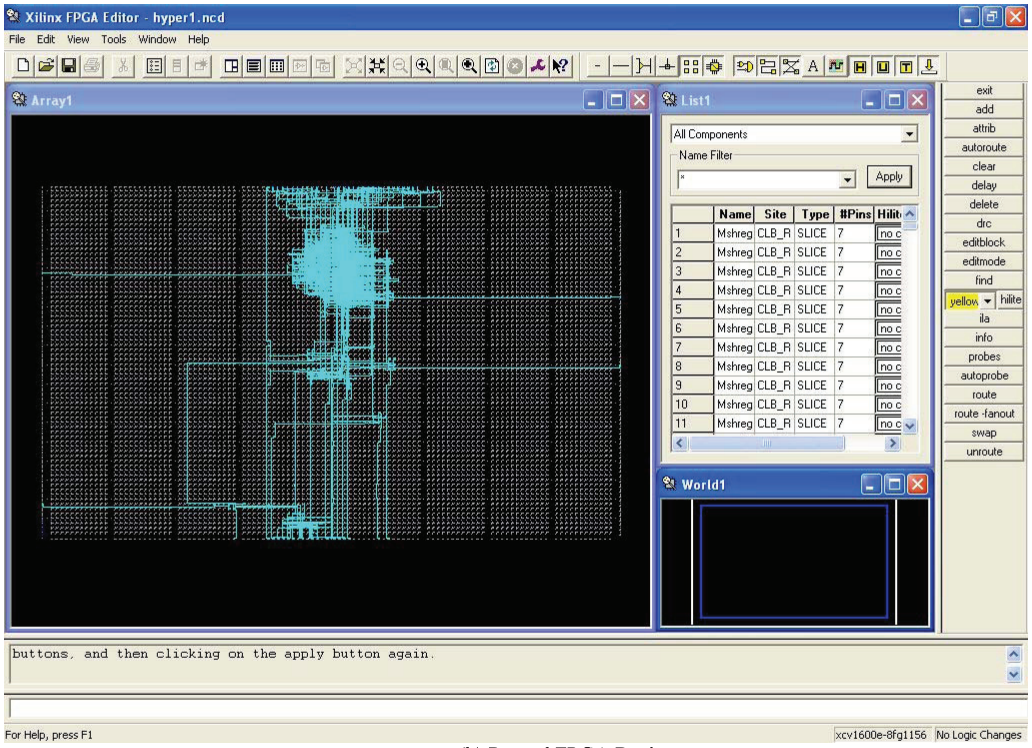
Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	1,791	10,240	17%	
Number of 4 input LUTs	5,381	10,240	52%	
Logic Distribution				
Number of occupied Slices	3,014	5,120	58%	
Number of Slices containing only related logic	3,014	3,014	100%	
Number of Slices containing unrelated logic	0	3,014	0%	
Total Number of 4 input LUTs	5,962	10,240	58%	
Number used as logic	5,381			
Number used as a route-thru	569			
Number used as Shift registers	12			
Number of bonded IOBs	97	328	29%	
IOB Flip Flops	95			
Number of GCLKs	1	16	6%	
Total equivalent gate count for design	72,151			
Additional JTAG gate count for IOBs	4,656			

(b) FPGA Device Utilization Summary of Sinh and Cosh Function

Fig. 4. CORDIC Simulation Waveforms and Synthesis Results



(a) Top View Zoomed CORDIC Hyperbolic Function



	Name	Site	Type	#Pins	Hilit
1	Mshreg	CLB_R	SLICE	7	no c
2	Mshreg	CLB_R	SLICE	7	no c
3	Mshreg	CLB_R	SLICE	7	no c
4	Mshreg	CLB_R	SLICE	7	no c
5	Mshreg	CLB_R	SLICE	7	no c
6	Mshreg	CLB_R	SLICE	7	no c
7	Mshreg	CLB_R	SLICE	7	no c
8	Mshreg	CLB_R	SLICE	7	no c
9	Mshreg	CLB_R	SLICE	7	no c
10	Mshreg	CLB_R	SLICE	7	no c
11	Mshreg	CLB_R	SLICE	7	no c

```
buttons, and then clicking on the apply button again.
```

(b) Routed FPGA Design

Fig. 5. CORDIC Algorithm Zoomed Top View and Routed FPGA Design

verified by simulating it using ModelSim (Ver. SE 6.4) based on test bench. The input stimulus covering all possible test cases is provided from the test bench. The ModelSim output waveforms are observed and are verified with Matlab results. After an initial delay of 450 ns, the output is obtained at every clock cycle. For the given input angle, sinh and cosh values are directly obtained. The exponential value is determined by adding sinh and cosh functions. Further, we use the exponential function for the design and implementation of 2D Gaussian function. The Gaussian function implementation is based on equation (10). The Timing diagram obtained from ModelSim tool is shown in Fig. 4a. The proposed method uses Xilinx ISE 9.1i for logic synthesis, placement, routing and FPGA implementation. The design is realized using XC2V1000-6bg575 Xilinx FPGA device. The device utilization and performance summary are shown in Fig. 4b. The zoomed top view modules of "CORDIC" and routed FPGA design are presented in Fig. 5.

7. Conclusion

The CORDIC algorithm is a widely used tool for digital signal processing and image processing applications. The hardware implementation of hyperbolic sine and cosine using CORDIC algorithm on FPGA is the main aim as the FPGAs can give enhanced speed at low cost with a lot of flexibility. Moreover it replaces multipliers with shifters and adders. In this paper the hyperbolic sine cosine CORDIC based generator is simulated using ModelSim. Further, the implementation of hyperbolic sine and cosine CORDIC based generators is done on Xilinx ISE 9.1i.

References

[1] Volder J. E, "The CORDIC Trigonometric Computing Technique", *IRE Transactions Electronic Computing*, Volume EC-8, pp. 330-334, 1959.

[2] Walther J. S, "A Unified Algorithm for Elementary Functions", *Springer Joint Computer Conference*, pp. 379-385, Atlantic city, 1971.

[3] Hu.Y. H, "CORDIC-based VLSI Architectures for Digital Signal Processing," *IEEE Signal Processing Magazine*, pp. 16-35, July 1992.

[4] Bonato V, Marques E, Constantinides G, "A Parallel Hardware Architecture for Image Feature Detection", *Reconfigurable Computing: Architectures, Tools and Application,* pp. 137-148, 2008.

[5] Wang H, Leray P, and Palicot J, "Reconfigurable Architecture for MIMO Systems based on CORDIC operators", *Comptes Rendus Physique*, 2006, Vol. 7, No. 7, pp. 735-750.

[6] K. Turkowski, "Fixed-Point Trigonometry with CORDIC Iterations," *Apple Computer White Paper*, January 17, 1990.

[7] Daniel R, Llamocca-Obregón, Carla P, and Agurto-Ríos, "A Fixed-Point Implementation of the Expanded Hyperbolic CORDIC Algorithm," *Latin American Appl. res*., 2007, Vol. 37, No. 1 .

[8] X. Hu, R. Huber, S. Bass, "Expanding the Range of Convergence of the CORDIC Algorithm", *IEEE Transactions on Computers*, Jan., 1991, Vol. 40, No. 1, pp. 13-21.

[9] Lakshmi B and Dhar A. S, "CORDIC Architectures: A Survey," Journal of *VLSI Design*, 2010, pp. 2-2.

[10] Andraka R. A, "A Survey of CORDIC Algorithms for FPGA based Computers," *Proceedings of the 1998 ACM/SIGDA sixth International Symposium on FPGAs*, , Monterey, California, 22-24 Feb., 1998, pp. 191-200.

[11] Velmurugan R, Cevher V, and McClellan J. H, "Implementation of batch-based Particle Filters for Multisensory Tracking" , IEEE *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Dec., 2007, U. S. Virgin Islands.

[12] Bhattacharyya K, Biswas R, Dhar A. S, "Architectural Design and FPGA Implementation of Radix-4 CORDIC Processor", *Microprocessors and Microsystems (MICPRO, Elsevier) Embedded Hardware Design*, 2010,Vol. 34, pp. 96-10.

[13] Giacomantone J. O, "Tradeoffs in Arithmetic Architectures for CORDIC Algorithm Design", pp. 1-9, CeTAD, De Ingenieria, 2001.

[14] Trio A, Purba R. S, "Scalable Pipelined CORDIC Architecture Design and Implementation in FPGA", *International Conference on Electrical Engineering and Informatics,* pp. 5-7 August 2009.

[15] http://www.dspguru.com/info/faqs/CORDIC.htm.

[16] Proakis J. G. Manolakis D. G, *Digital Signal Processing Principles*, Algorithms and Applications, Prentice Hall Upper Saddle River, NJ.

[17] www.51protel. com/tech /Introduction

[18] www.xilinx.com/partinfo/#4000.pdf