International Journal of Scientific & Engineering Research, Volume 4, Issue 8, August 2013
ISSN 2229-5518

Comparison of FPGA Based Cordic Architectures

Jinu Elizabeth John,Asst.Prof.,Dept. of ECE, Saintgits College of Engineering

Abstract— The COordinate Rotational DIgital Computer (CORDIC) algorithm is an iterative method for performing basic arithmetic operations including certain linear, trigonometric, hyperbolic and logarithmic functions. [1]–[3]. The algorithm only uses elementary shift-and-add steps to performing two-dimensional (2-D) plane vector rotations facilitating easy hardware implementation. This is mainly used in signal processing architecture. This paper presents a comparison of the various CORDIC architectures, especially in three different major styles iterative, parallel and pipelined structures with respect to their speed, area, and data throughput performance All three designs were coded in VHDL, simulated using Modelsim 6.2 c and Implemented in Xilinx Sparten 3E and NET FPGA and Synopsis ASIC synthesis tools.

Index Terms—CORDIC ,Vector Rotation ,sine, hyperbolic

1 Introduction

IN 1959 , Jack E. Volder was first to describe the modern CORDIC algorithm at the aero electronics department of Convair so as to replace the analog resolver in the B-58 bomber's navigation computer. Even though CORDIC is similar to mathematical techniques published by Henry Briggs in 1624, it is optimized for low complexity finite state CPUs.The further generalization of the algorithm including the multiplications, divisions, and square roots, hyperbolic, exponential functions and logarithmic calculations was done by John Stephen Walther at Hewlett-Packard. The binary numeral system was first used to implement the CORDIC algorithm .CORDIC can be used in any architecture where speed of operation is not very important compared to cost. It is well-suited for handheld calculators where the chip gate count has to be minimized..

CORDIC seems to be faster than other approaches when a hardware multiplier like a micro controller is not available or when the number of gates required to implement the functions it supports should be minimized (e.g., in an FPGA).

On the other hand, when a hardware multiplier is available (e.g., in a DSP microprocessor), table-lookup methods and power series are generally faster than CORDIC. Nowadays, the CORDIC algorithm is used extensively for various FPGA implementations in biomedical field. The software implementations include integer-only CPUs . The most modern general-purpose CPUs have floating-point registers with common operations such as sin, cosine, square root, log10, natural log, addition, subtraction, multiplication and division .Adaptive signal processing algorithms, require the computation of Eigen values, the solution of systems of linear equations and conversions between polar and rectangular co-ordinates [3]. All these tasks can be efficiently implemented using processing elements performing vector rotations. The CORDIC has the capability to calculate all the desired functions in a rather simple way using addition, subtraction, bit shift operations and lookup tables, without using any hardware multiplier . Due to the simplicity of the involved operations the CORDIC algorithm is very well suited for VLSI implementation. The scaling of rotated vector is done for making necessary scale factor corrections.

2 Cordic Algorithm

The CORDIC algorithm exists in two different modes vector translation mode and vector rotation mode.

In vector translation mode the coordinates (x0, y0) are rotated until y0 converges to zero. This paper discuss about the vector rotation mode of CORDIC. Initial vector (x0, y0) starts aligned with the x axis and is rotated by a specific angle during every cycle, so that after n iterations, we get the desired angle. The main idea consists in taking a unit vector and applying successive rotations, called micro-rotations, until the desired angle is reached. The rotating vector is chosen to be unit vector, since after n iterations it will contain sinαn and cosαn in its second and first components respectively

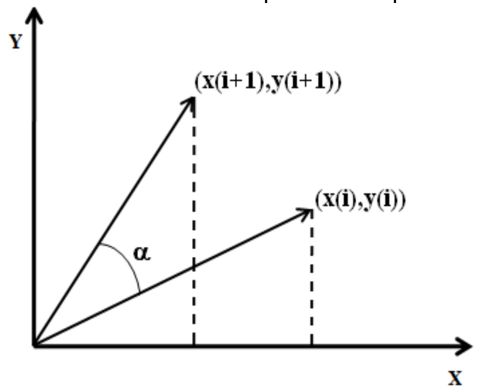

Fig 1: Illustration of Vector Rotation

Consider the iterative rotation of a vector (x(i),y(i)) ,by an angle αi to obtain an (x(i+1), y(i+1)) [1],

$$\begin{bmatrix} x^{(i+1)} \\ y^{(i+1)} \end{bmatrix} = \begin{bmatrix} \cos \propto_i & - & \sin \propto_i \\ \sin \propto_i & - & \cos \propto_i \end{bmatrix} \begin{bmatrix} x^{(i)} \\ y^{(i)} \end{bmatrix}$$

This can be rewritten as

$$\begin{bmatrix} x^{(i+1)} \\ y^{(i+1)} \end{bmatrix} = \cos\propto_i \left[\begin{matrix} 1 & -\tan\propto_i \\ \tan\propto_i & 1 \end{matrix}\right]\begin{bmatrix} x^{(i)} \\ y^{(i)} \end{bmatrix}$$

Where tan αi can be restricted to di 2-i , so the multiplication can be converted into an arithmetic right shift [5] , with di =± 1. The first factor

$$\cos\propto_i = {}^{1}\!/\!\sqrt{(1+2^{-2i})}$$

For rotation over an arbitrary angle α, -π/2 ≤ α ≤ π/2 , it can be decomposed as :

$$\propto = \sum_{i=0}^{n} d_i \, arctan 2^{-i}$$

The cosine term could also be simplified and since cos(α) is a constant for a fixed number of iterations. This iterative rotation can now be expressed as:

$$x^{(i+1)} = k^i[x^i - y^i d_i 2^{-i}]$$

$$y^{(i+1)} = k^i[y^i - x^i d_i 2^{-i}]$$

where, i denotes the number of rotation required to reach the required angle of the required vector, ki=cos(arctan(2-i)) and

$$d_i = \pm 1$$

.The product of the ki 's represents the scale factor [6] K :

$$K = \prod_{i=0}^{n} \cos\propto_i = \prod_{i=0}^{n} {}^{1}\!/\!\sqrt{(1+2^{-2i})}$$

On each iteration it is necessary to decide whether

$$d_i = 1 \, or d_i = -1$$

In order to make that decision, the difference between the desired angle and the current angle is used. So a new variable known as accumulator is defined as :

$$z^{(i+1)} = z^{(i)} - d_i \, I_{(i)}$$

where I(i) is the LUT entries.
The sum of the rotating angles gives the desired angle

$$\theta_n = \sum d_i \arctan \quad 2^{-i}$$

3 Proposed Architecture

This section deals with different hardware used for computation of sine and cosine using CORDIC [7]. Here iterative rotations of a point around the origin on the x-y plane are considered. In each rotation, the coordinates of the rotated point and the remaining angle to be rotated are calculated. Since each rotation is a rotation extension the number of rotations for each angle should be a constant independent of operands .So the gain factor K becomes a constant. Hardware implementation for CORDIC arithmetic requires three registers for x, y and z, two shifter to supply the terms 2-i x and 2-i y to the adder/subs tractor units and a look up table to store the values of αi=tan -12-i . The di factor (-1 and 1) selects the shift operand or its complement. The initial inputs to the architectures are X0=1, Y0=0. The structure requires a preprocessing unit to converge the input angles to the desired range and a post processing unit to fix the sign of outputs depending on the initial angle quadrants.
The pre-processing unit takes in angles of any range and converges it to the interval [-п/2, п/2]. It keeps record of the quadrant of the input angle which may be used in the post-processing unit to fix the sign of outputs. These two blocks are inevitable for any application as the input range cannot be predicted always.

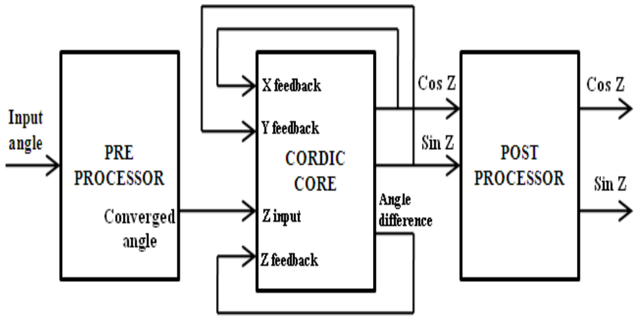

Fig 2: CORDIC Block Diagram

3.1 Sequential/Iterative Architecture

The CORDIC algorithm requires approximately one shift-add/sub operation for each bit of accuracy. A CORDIC core implemented with sequential architectural configuration, implements these shift-add/sub operations serially, using a single shift-add/sub stage and feeding back the output. An iterative CORDIC core with N bit width has a minimum latency of N cycles. It takes at least N cycles to produce new output. The implementation size is directly proportional to the internal precision. This architecture finds major application in pocket calculators, since even a delay of thousands of clock cycles constitute a small fraction of a second for a human user. To obtain sine and cosine values of a given angle z0, iterative structure takes the value of (x0,y0) as (1,0) in the first clock cycle. From the next clock cycle onwards it takes the feedback values and the operation continues till the required output is obtained. The control signal for the input registers is provided by a state-machine designed for the purpose. To get an N bit precise output, the structure requires iterating at least N times [4]. Hence, it requires a minimum of N clock cycles for required output.

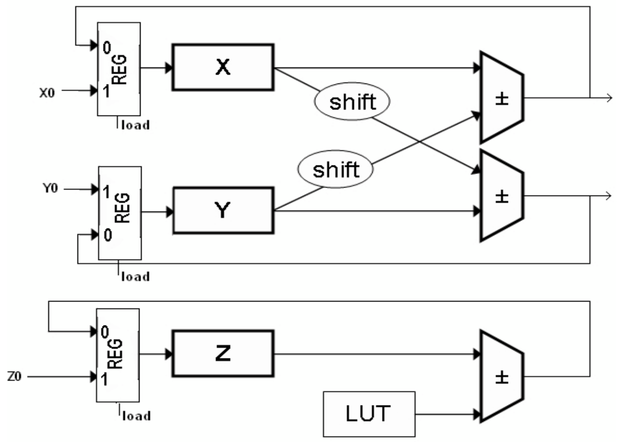

Fig 3: Sequential CORDIC

3.2 Cascaded CORDIC Architecture

This architecture uses multiple instances of Iterative CORDIC structure. A CORDIC core with parallel architectural configuration implements the shift-add/sub operations in parallel using an array of shift-add/sub stages [8]. A parallel CORDIC core with N bit output has a latency of one clock cycle.

The implementation size of a parallel CORDIC core is directly proportional to the internal precision times the number of iterations. Instantiation of blocks must be done N times for an N bit precise output. Unlike in iterative CORDIC, all iterations are done in parallel and hence need not wait for N clock cycles. But, the latency of each block has an inevitable role in fixing the clock frequency. The frequency of operation for Parallel CORDIC core will be lesser than the frequency of operation of iterative CORDIC. But this is the case with a single iteration. While dealing with a chain of inputs, the parallel structure proves to be more efficient one since the throughput of parallel structure is much greater than that of iterative. The shifters used in this structure are constant shifters, which can be implemented in the wiring, so that the hardware can be reduced.

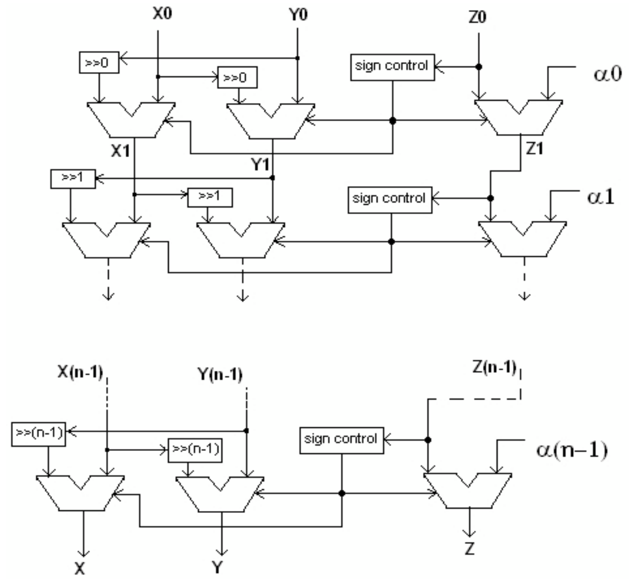

Fig 4: Cascaded CORDIC

3.3 Pipelined CORDIC Architecture

Pipelined architecture uses a structure similar to that of a Parallel CORDIC. It uses pipeline registers in between each iteration phase as shown in Fig. 5.

Pipelined CORDIC proves to be advantageous with continuous input values. For an N bit data 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 during every clock cycle. The advantage of pipelined CORDIC core over parallel and iterative CORDIC cores is its frequency of operation which is much higher when compared to the latter two structures. Pipeline realizes same throughput as that of parallel core with improved frequency of operation. This feature of pipelined structure makes it the best possible option for high frequency satellite communication and other communication systems. A drawback of pipelined structure is the increase in area introduced by the registers. Hence, there is a trade-off between parallel and pipelined cores based on frequency and area.

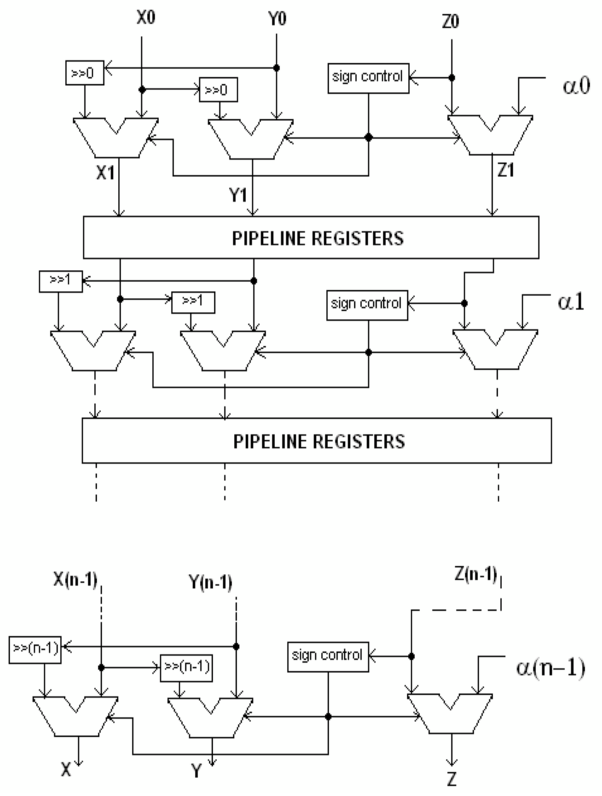

Fig 5: Pipelined CORDIC

4 Implementations and Results

CORDIC design implementation was first done in Matlab. Iterative, parallel and pipelined CORDIC were implemented in VHDL as well.
Matlab does not take into account any structural differences. The implementation consists of modeling the basic CORDIC equations. Input angles were given in degrees. The scaling factor K was fixed to 0.6073 [5].
In order to generate the sine and cosine values, it was inevitable to design a preprocessing as well as post-processing modules. The preprocessing module restricts the angles beyond π/2 and - π/2 to the [-π/2, π/2] range. Similarly, the post-processing module fixes the sign of generated sine and cosine values based on the quadrant of the initial input angle. Sine and cosine waves were generated in Matlab and compared with the waves generated by inbuilt functions. Waveforms generated by the designed CORDIC function are shown below.

VHDL coding for iterative, parallel and pipelined CORDIC cores were done and simulated in Modelsim. Synthesis was done in Xilinx and results were obtained as given in the tables below.
Selected Device: Xilinx Spartan 3

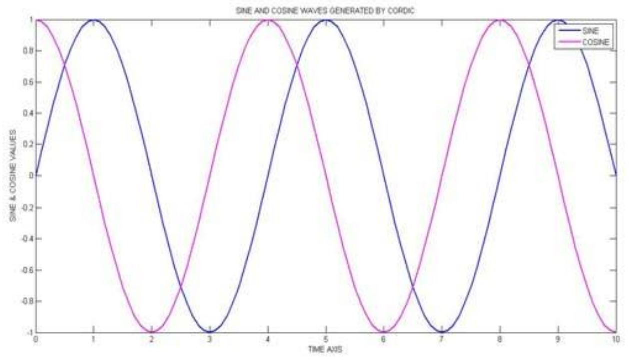

Fig 5: MATLAB Waveforms

Table 1. Device Utilization Summary

Parameter	Iterative CORDIC		Parallel CORDIC		Pipelined CORDIC	
No. of Slices	128	13%	373	38%	373	38%
No.of Slice FlipFlops	59	3%	50	2%	723	37%
No. of 4 input LUTs	241	12%	721	37%	721	37%
No. of IOs	52	----	49	----	49	----
No. of bonded IOBs	52	78%	49	74%	49	74%
No. of GCLKs	2	8%	1	4%	1	4%

From the table given above, iterative, parallel and pipelined CORDIC has a commendable difference in terms of the blocks synthesized. As evident from the block diagram shown above, parallel CORDIC has multiple instantiation of hardware blocks which has been instantiated only once in iterative structure. Hence, the number of slices and 4 input LUTs are more for parallel CORDIC whereas number of slice flip-flops and number of IO blocks are more for iterative structure. The pipelined structure uses N instantiation of three register sets which results in an increase in the number of slice flip-flops.
Another major observation is on the frequency of operation compared to parallel and pipelined has the highest frequency. The data output time corresponding to the frequency of iterative core is the time required to get output after single iteration. In order to get the required output at least 16 similar iterations are needed, which will take ~154.752 nS. The data output time shown for parallel and pipelined structures is the time required to get the final output as the instantiations are done simultaneously. Time taken for parallel is almost less than half of the iterative structure and that of pipelined is less than almost thirteen times that of parallel. Hence, pipelined structure can be used for high speed applications like in satellite communication.

Further analysis was done in Design Vision. All three of the circuits were analyzed for area, timing and power parameters. The work done is based on 13 micron technology.

Table 2. Total Area Analysis

Parameter	Iterative CORDIC	Parallel CORDIC	Pipelined CORDIC
Number of ports	52	49	49
Number of net	255	1587	2323
Number of cells	103	158	174
Number of references	26	111	159
Combinational area	1146	14925	15148
Non-combinational area	660	336	5488
Total area (nm^2)	1806	15261	20636

Parallel CORDIC structure is a simple yet bigger structure than iterative CORDIC structure. Due to multiple instantiations of consisting blocks, parallel has a higher area (almost eight times) than the iterative structure. Pipelined structure has an added area of the registers apart from that of parallel. From the table above, iterative structure provides the best design in terms of area.

Table 3. Iterative CORDIC _Timing Analysis

Point	Incr	Path
load_reg/CP	0.00	0.00r
(FDS2L)	23.46	23.46 r
load_reg/Q	0.00	23.46 r
(FDS2L)	---	23.46
done_out (out)		

Table 4. Parallel CORDIC _Timing Analysis

Point	Incr	Path
load_reg/CP	0.00	0.00r
(FDS2L)	23.46	23.46 r
load_reg/Q	0.00	23.46 r
(FDS2L)	---	23.46

Table 5. Pipelined CORDIC _Timing Analysis

Point	Incr	Path
reg_y_15/outp_reg[15]/CP	0.00	0.00r
(FD1)	1.41	1.41 f
reg_y_15/outp_reg[15]/Q	0.00	1.41f
(FD1) reg_y_15/outp[15]	0.00	1.41 f
(pipe_reg_1) sin_z[15] (out)	---	1.41
data arrival time		

From the tables it is evident that parallel and pipelined CORDIC is much faster than the iterative CORDIC. Though it consumes a higher area, Parallel and pipelined CORDIC structures will be preferable for high speed applications.

Table 6. Synopsis_Power Analysis

Parameter	Iterative CORDIC	Parallel CORDIC	Pipelined CORDIC
Cell Internal Power	0.00 nW	0.00 nW	0.00 nW
Net Switching Power	197.6024 uW	2.1496 mW	2.3158 mW
Total Dynamic Power	197.6024 uW	2.1496 mW	2.3158 mW

The table above gives the comparison between the power consumption of parallel, pipelined and iterative CORDIC. While iterative consumes power in uW range, due to its higher hardware complexity, parallel and pipelined consumes power in mW range.

5 Conclusion

A tradeoff area/speed will determine the right structural application. An iterative CORDIC uses lesser hardware than parallel or pipelined CORDIC, but with the number of iterations the shift distance changes, which requires a high fan in and reduce the maximum speed of application. Area used by Parallel and pipelined CORDIC is much higher compared to that of Iterative CORDIC. This difference in hardware units has caused an increased power usage by Parallel and pipelined structures. The time advantage of parallel and pipelined over iterative is that it gives the output in just one clock cycle whereas iterative takes at least N clock cycles, where N is the number of bits used. So these two structures can be used for high speed applications, like satellite data processing system. Current work implemented a completely programmable and re-configurable cordic block which consist of iterative, parallel and pipelined structures, with variable word size and can be configured by the user based on the application

References

[1] J. Volder, "The CORDIC computing technique,"IRE Trans. Electronic Computers, vol. EC-8, pp. 330-334, Sept. 1959.

[2] J. Walther, "A unified algorithm for elementary functions," Proc. AFIPS Spring Joint Computing Conf., vol. 38, pp. 379-385, 1971.

[3] F. Angarita, A. Perez-Pascual, T. Sansaloni, and J. Vails, "Ef cient FPGA Implementation of CORDIC Algorithm for Circular and Linear Coordinates," International Conference on Field Programmable Logic and Applications, pp. 535–538, Aug 2005.

[4] Uwe Meyer-Baese. "Digital Signal Processing with Field Programmable Gate Arrays". Springer-Verlag, New York, Inc., Secaucus, NJ, USA, pp. 70-75, 200.

[5] Milos D.Ercegovac, Tomas Lang "Digital Arithmetic", pp 631-664, Morgan Kaufmann Publishers,San Francisco,USA

[6] M.Garrido and J.Grajal,"Memoryless CORDIC for FFT computation" ICASSP 2007.

[7] Ray Andraka. "A survey of CORDIC algorithms for FPGA based computers". Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays , pp 192-200, New York, NY, USA, 1998.

ACM Press.

[8] Esteban O. Garcia, Rene Cumplido, Miguel Arias, "Pipelined CORDIC Design on FPGA for a Digital Sine and Cosine Waves

[9] Wu, An-Yeu Andy ," A unified view for vector rotational CORDIC algorithms and architectures based on angle quantization approach",2002

[10] Wu, Cheng-Shing," A high-performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes"2003

[11] Antelo,Elisardo Fac. de Fisica, Santiago de Compostela Univ., Spain "High performance rotation architectures based on the radix-4 CORDIC algorithm"

[12] Pongyupinpanich, S. ; Samman, Faizal Arya ;Glesner, Mandfred ; Singhaniyom, S. "Design and evaluation of a floating-point division operator based on CORDIC algorithm",2012