International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-2 Issue-5, April 2013

Design and Simulation of Cordic Co-Processor and Its Application in Avionics

Chowdi Ravitej, Elphesj Churchill, Kishore Sonti

Abstract: ***A technique is allocated going to discuss the application of cordic algorithm in avionics. Actually here the process is dealing with avionics so, the smart application of cordic algorithm is in ARM processor. In GNSS (global navigation satellite system) receiver make use of ARM processor floating point instruction (FPI) are there to calculate the FPI. It contains floating point unit (FPU). So, to make easy calculation in FPU have implemented cordic algorithm here cordic calculation means calculating trigonometric values. In this way FPU has implemented. Here trigonometric values means sin, cosine, tangent after getting tangent values have to see timing response of the binary output. So, in navigation system. Now, accurate signals have been sensed. Without any critical path delay then automatically speed will increase delay will reduce this is more advantage in avionics system. For floating point addition, exponent matching and shifting of 24 bit mantissa and sign logic are coded in behavioral style. Prototypes are implemented on Xilinx vertex-4 and 5. By designing pipelining in cordic and wave pipelining in cordic is implemented in cordic algorithm to reduce the timing response in the navigation system.***

Index terms – fpu, cordic algorithm, pipelining and wave pipelining in cordic, avionics.

I. INTRODUCTION

Cordic (co-ordinate rotation digital computer) algorithm is a hardware efficient algorithm.

It is an iterative in nature and is implemented in terms of rotation matrix. It can perform a rotation with the help of a series of incremental rotation angles each of which is performed by a shift and add/sub operation. Evaluation of trigonometric values sine, cosine, and tangent is generally a complex operations which requires a lot of memory has complex algorithm and requires a large number of clock cycles with expensive hardware organization. But the algorithm that is use here is absolutely simple with very low memory requirements faster calculation and commendable precision which use only bit shifting operation and add/sub operation to compute any function. CORDIC (Coordinate Rotation Digital Computer) is a simple and hardware-efficient algorithm for the implementation of various elementary, especially trigonometric, functions. Instead of using Calculus based methods such as polynomial or rational functional approximation, it uses simple shift, add, subtract and table look-up operations to achieve this objective.

Manuscript published on 30 April 2013.
***Correspondence Author(s)**
Chowdi Ravitej, , Chowdi Ravitej, Electronics and Communication Engineering, Sathyabama University,/Chennai/India.
Elphesj Churchill, Elphej Churchill, Electronics and Communication Engineering, Sathyabama University,/Chennai/India.
Kishore Sonti, vjk. kishore sonti, Electronics and communication Engineering, sathyabama University,/Chennai/India.

The implementation of cordic algorithm is as follows:

- Three registers for the values of xi,yi,zi.
- Shifter for implementation of 2^{-i}.
- Adder/subtraction blocks for arithmetic operations are there as per di (direction) value these values are got from alpha $i=\tan(2^{-i})$.

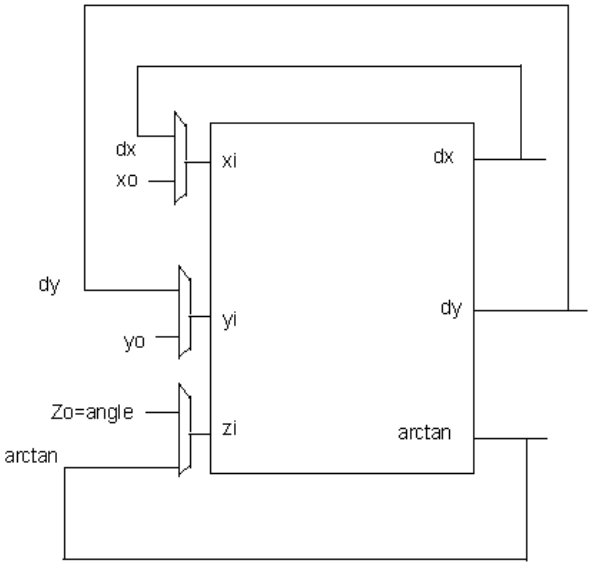


Fig: 1.1: Block diagram of cordic algorithm

The arc tan values are calculated by arc $\tan(1/2^n)$ for calculating number of iterations in cordic algorithm for one iteration cordic requires approximately two shifts and the three add/sub operations all required the functions of the cordic algorithm is carried out in the cordic core block to calculate each and every iteration the value of angle accumulator arc tan will be compared with given angle then the remaining angle of rotation will be calculated after comparison and the result is feedback to the input arc tan. The above is repeated to all specified iterations for outputs dx, dy is the feedback input to xi, yi after the n iterations the values.

Cordic algorithms is only shift and add/sub computing algorithm it can perform various functions cordic based VLSI architectures provides a better alternative to the architectures based on multiply and add/sub hardware mainly cordic algorithm is used for reducing the complexity of the circuit it is main application in aircraft navigation process. By implementing the cordic algorithm complexity of the circuit will be reduced because of one bit shifting and add/sub operations and timing response will be reduced area will be decreased this all advantage by using cordic algorithm.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

The application of cordic algorithm in floating point unit reduced area and improved the speed and reduced delay. And it provides a better design architecture for further designs. Hence by reducing these major factors like area and delay. Is more advantage in real time process.

II. APPLICATION OF CORDIC IN VARIOUS AVIONICS

In avionics top-level functional requirements and structural constraints are two types:

1. Flow
2. Models

FLOW:

Avionics control system (ACS) continuously collects data is a large number of inputs including positional and environmental data from external sensors state controls and desired flight objectives and others control information it converts these in to a small number of outputs primarily displays altering pilot about Suggested control actions and direct manipulations of aircraft effectors to show the proper dimensions to the pilot.

MODEL:

To carry out it's functionally an ACS must maintain coherent, accurate models the top most model in an ACS is an action model representing desired effectors and displays settings estimated properties of the actual properties compared with ideal (or) desired properties that may be further transformed in to changes in effectors and/or display settings that reduce these difference thus error models representing differences between actual and desired state play a central role in ACS system because error models are used to effect particular changes in the aircraft must it has to be very simple form to support comparator algorithm other ACS models are sub divided in to two similar data mostly independent categories here navigation models that represents the actual state and objective models that represent desired states.

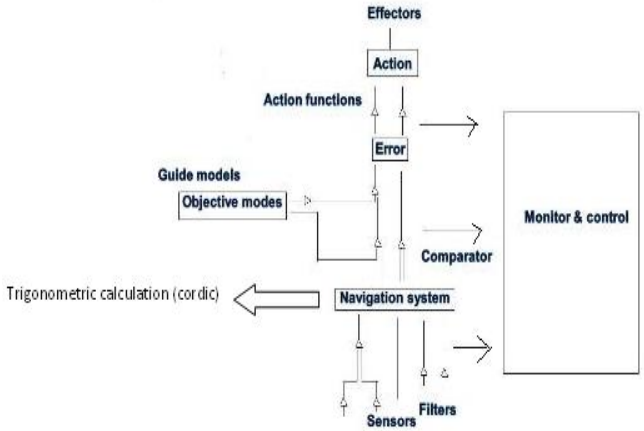


Fig: 2.2: Block diagram of navigation system

III. GUIDE MODEL

In this particular model for purpose of avionics control it has been found that the air navigation has been broken up in to a few standard categories with the vary according to particular top-down prosperities needed to support to the guidance and bottom-up for capabilities of sensor hardware here guidance model gives a proper estimator positions to the pilot how the environment is there and what angle should the pilot takes place in each and every models of comparator checks and gives a good support to the pilot this is more benefit to pilot and more over by this way pilot work will be fast guidance mode components that serve the same role as staged estimations in navigation.

IV. OBJECTIVE MODELS

Objective models are used to represent the desired state of the aircraft here the objective model state values are computed by any number of guidance mode by its components. These values are compared to estimated actual state represented in navigation models by guidance comparator component in order to effect flight this all Parameters are required to seen by the pilot.

V. NAVIGATION MODELS

The purpose of the avionics control the air of navigation may broken up in to a few standard categories with respect to details of vary according to particular to details of vary according to particular top-down properties needed to support guidance algorithm and bottom up capabilities of sensor. Navigation model properties may be needed at a number of points in an avionics control system it every time includes the computation of error initialization (or) correction of data for the desired state of computation. It has look over the entire atmosphere in the sky so, all these types of terms should be observed by pilot. The atmosphere should be seen by earth directions so, by these way pilot can come to know that what is exact position is there and is an environmental position it all these conceptually will be get information to pilot by the earth actually these position will be gathered by very higher frequency Omni- directional range (VOR) to transmit radials (degree) to allow pilot to determine position to form the station.

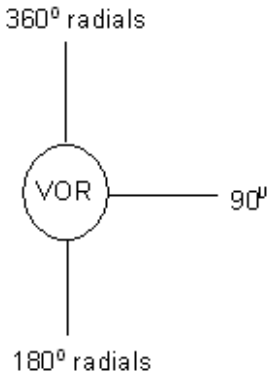


Fig: 2.3: Example Diagram Of very high frequency Omni-directional range

VI. APPLICATION OF CORDIC IN FLOATING POINT UNIT

Floating-point units (FPU) mainly are a math coprocessor which is designed specially to carry out operations on floating point numbers. Typically FPUs can handle operations like addition, subtraction, shifting. FPUs can also do various transcendental functions such as exponential or trigonometric calculations, though these are done with software library routines in most modern processors. Used FPU is basically a single precision IEEE754 compliant integrated unit.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Sign bit is the sign of the number where 0 denotes a positive number and 1 denotes a negative number. It is the sign of the mantissa as well. Exponent is an 8 bit signed integer from −128 to 127 (2's Complement) or can be an 8 bit unsigned integer from 0 to 255 which is the accepted biased form in IEEE 754 single precision definition. In this case an exponent with value 127 represents actual zero. The true mantissa includes 23 fraction bits to the right of the binary point and an implicit leading bit (to the left of the binary point) with value 1 unless the exponent is stored with all zeros. Thus only 23 fraction bits of the mantissa appear in the memory format but the total precision is 24 bits.

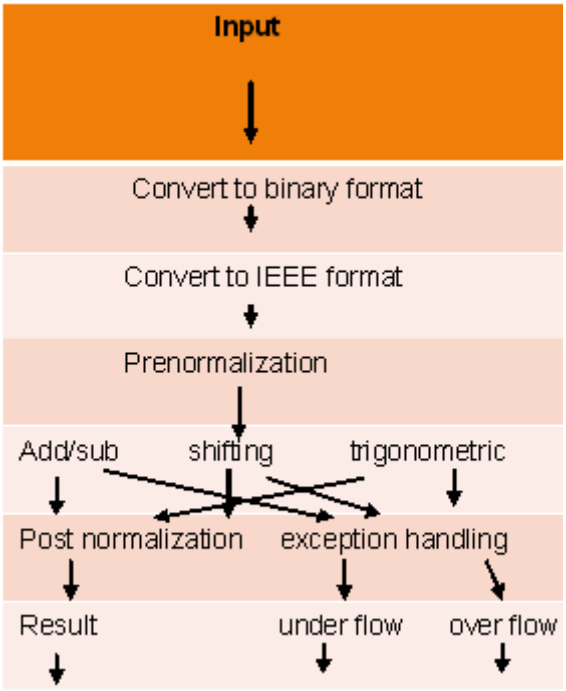
Step 1: The sign bit (31^{st} bit) of the input integer part becomes the sign bit of the effective operand.

Step 2: Then the position of 1^{st} significant 1 is searched the input integer part from rhs this position is stored.

Step 3: All the bits from this position to the end of the input integer part (i.e till the 0^{th} bit) is taken and inserted into the effective operand from its 30^{th} bit on word. (this step stores the actual useful bits of the integer part as not all the 32 bits are use to accommodate the integer part).

Step 4: If there are still positions in the effective operand that are not filled, then it is filled with the bits from the input fractional part from its msb down to the number of bits equal to place left to be filled.

VII. ALGORITHM FLOW OF CORDIC IN FPU



- Step 1: Two inputs have given.
- Step 2: Convert to binary format. Convert to 32 bit integral and 32 bit fraction part in to single novel integral representation.
- Step 3: Convert to IEEE format. Convert 32 bit binary to its equivalent IEEE 754 format.
- Step 4: Pre-normalization adjusts the input by performing the necessary shifts before an add or subtraction operation.
- Step 5: Add/sub performs the shifting of the inputs to the specified bit in specified direction.
- Step 6: Trigonometric performs the trigonometric values (cordic).
- Step 7: Post normalization normalizes the results of add/sub operation to its IEEE 754 format.
- Step 8: Shifting performs the shifting of the inputs to specified bit in specified direction.
- Step 9: Except ion handling which includes any exceptional conditions like under flow , over flow, occurred during the operation.
- Step 10: Under flow occurs when an operation results very small i.e. outside the normal range and in exact (de-normalized value) by default.
- Step 11: Over flow occurs when an operation results a very large number that can't be represented correctly i.e. which returns ±infinity by default.

VIII. PIPELINING AND WAVE PIPELINING IN CORDIC

The pipelined cordic algorithm is proven to be more advantageous for continuous input values when compared with other N-stage of obtained after N-clock cycles and the rest of the outputs are generated continuously for every clock cycle.

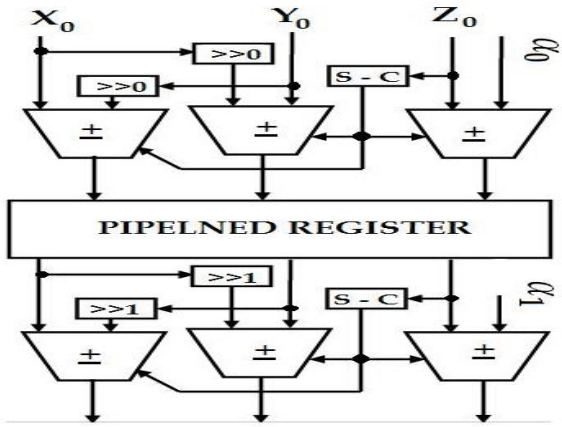


Fig: 4.4: Diagram Of pipelined in cordic

By using this pipeline converts iterations in to pipeline phases in this way an output is obtained at each and every clock cycle after pipeline stages will generate each pipeline stages exactly one clock cycle to complex an iterations one of the most problems for a cordic implementation of over flow. The risk is present in binary when angles $\pi/2$ (or) $-\pi/2$ this is due to the fact that the difference in binary represents between these two angles in one bit when approximation is being made an angle could cross from a positive right angle to a negative one here over flow is solved by adding an overflow control it checks for the signs of the operation involved in additions (or) subtraction and the result of the operation overflow is produced then the result keeps of its last sign by this process it will not affect on the final result for each stages the value of arc tan $(1/2^n)$ is taken from memory in the overflow control the sign of zi is used to determine di so, for each stage is applied as input to adder/subtraction to decide if a sum (or) a subtraction is performed. The wave pipelined in cordic algorithm is proven to be more advantageous then pipelining process for continuous input values when compared with other N-stage of obtained after N-clock cycles and the rest of the outputs are generated continuously for every maximum and minimum clock cycle.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

From this process the timing response is reduced and number of logic gates are reduced speed will increase if clock min is one then some output will come and if clock is max is one then total output will come if clock is min and max then output is zero.

IX. ANGULAR SEQUENCER

Cordic module sine and cosine these are the angles depends on the tangent it is an angle so, it is necessary to give it sequence of angle in the range of cordic convergence angle this sequence is not a simple of calculating angles but it preferably able to change the frequency of the signal which is implemented by cordic this module is implemented on new angles to accomplish this function requires an angle sequencer.

Cordic only convergence with in the first and fourth quadrant this type is needed a sequencer to generate angles are from 0 to $\pi/2$ to generate angles are from $\pi/2$ to $-\pi/2$ and $\pi/2$ to 0. To solve overcome this problem requires a example saw tooth wave generator is needed and it is designed states at zero and increases its output value until 2^n where n is the angle of depth in bits as the reduced the signal area will goes from the top positive values to the minimum negative values so, it increase again and repeats the cycle.

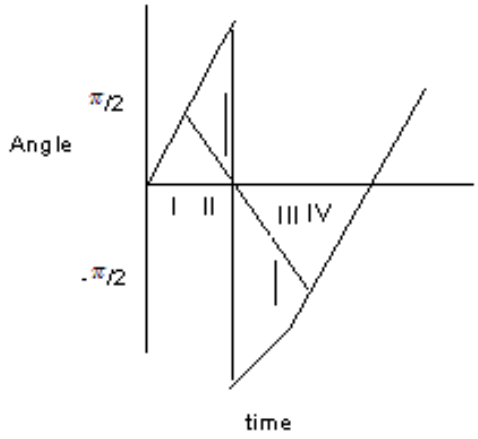


Fig: 5.5: Diagram Of angle sequencer

This saw tooth wave is not use full without triangle wave so, it needs the triangle wave to feed the cordic module this triangle is for represents the first and fourth quadrants now saw tooth wave is performing in to the triangle wave. The two most significant bits indicates the current quadrant here when others two most significant bits are 01 (or) 10 then the saw tooth signal must be performed from first quadrant to fourth quadrant. This process is done by inverting every single bit without changing the sign bit.

VI.CORDIC ITERATIONS COMPARISON

Iterations	Delay time	Angle	Directions
0	8192ps	28.9^0	Di +1
1	5216ps	30.7^0	Di +1
2	2607ps	29.8^0	Di -1
3	1303ps	30.3^0	Di +1
4	651ps	30.0^0	Di -1
5	326ps	29.9^0	Di -1

Cordic iterations are compared here time delay can be observed reduced timing response can be considered from the iterations and here angle of the each iteration is mentioned with the help of positive and negative directions. Here in the output delay time can be observed. Here the delay time is mentioned it is observed by output wave form in decimal form with respect to time in picoseconds (PS).

VII. SIMULATIONS AND RESULTS

7.1 SIMULATION RESULT OF CORDIC IN FPU

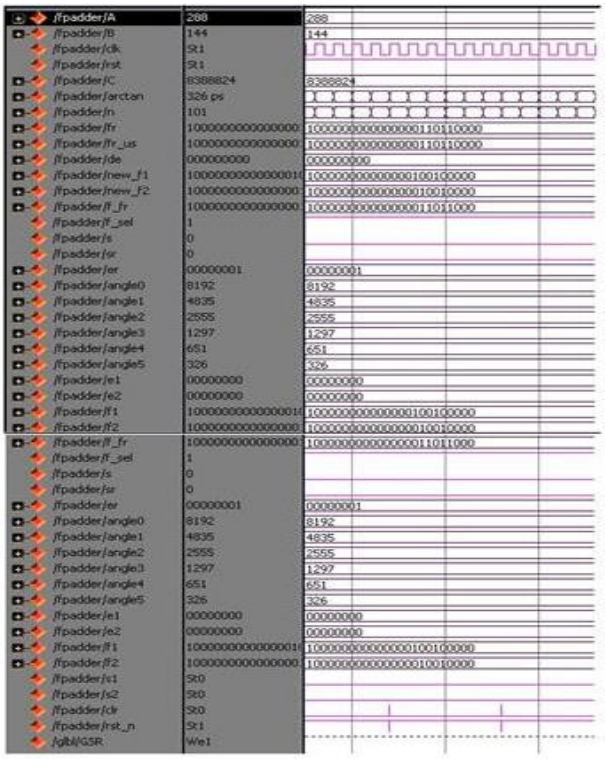


Fig: 7.6: Output Waveform of cordic in FPU

In Fig:7.6 the output wave is explained as follows:

1. A, B: input 32-bit single precision numbers.
2. C: output 32-bit single precision number.
3. s1, s2, s3, e1, e2, e3 & f1, f2, f3: sign exponent and fraction parts of inputs.
4. new_f1, new_f2: aligned mantissas.
5. de: difference between exponents.
6. fr: 25-bit result of addition of mantissas.
7. fr_us: unsigned 25-bit result.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

8. f_fr: normalized 24-bit fraction result.
9. er, sr: exponent and sign of result.
10. arc tan for angles purpose.
11. n is taken for shifting the angles.

This Table Shows the Combinational Path Delay and Number of LUT's occupied

Floating-point unit in cordic Xilinx IPs	LUTs	Combinational path Delay (ns)
Fpu in cordic	1325	34.826ns
Complete architecture	1325	34.826ns

The proposed CIFM architecture is implemented in Verilog HDL and logic simulation is done in Verilog Simulator the synthesis and FPGA implementation is done using Xilinx 9.1. The design is optimized for speed and area using Xilinx, Device Family is Virtex-4, Device XC4VLS15, Package SF363, Speed grade: -12. The device is made up of multiplexers and LUTs. It has been found that for Xilinx Virtex4 family, the delay of the proposed architecture is 34.826 ns with the area (cell usage) of 691.

7.2 SIMULATION RESULT OF PIPELINING IN CORDIC

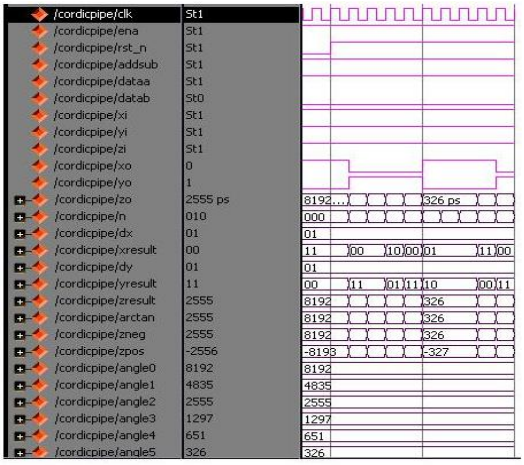


Fig: 7.7: output waveform of pipe lining in cordic

In **Fig: 4.2** output is pipelining in cordic with the help of pipeline process the delay is reduced some area will increase but it is good process from this timing response will be reduced processing speed will increase. Here pos and neg values are observed in the output.

7.3 SIMULATION RESULT OF WAVE PIPELINING IN CORDIC

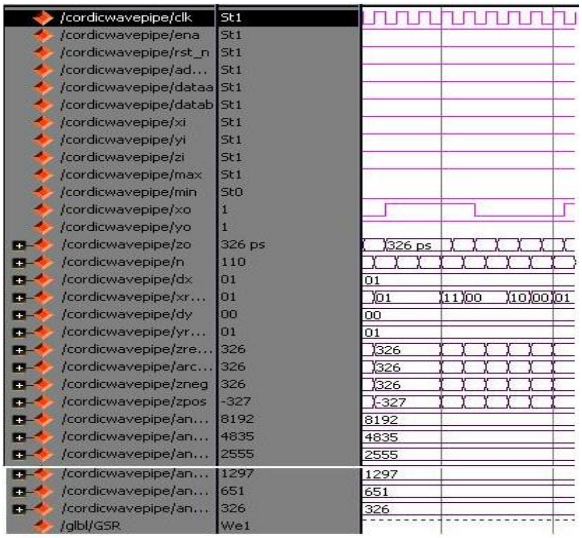


Fig: 7.8: output waveform of wave pipelining in cordic

In the **Fig: 7.8** output is wave pipelining in cordic with the help of wave pipeline process the delay is reduced some area will increase but it is good process from this timing response will be reduced processing speed will increase.

This Table Shows the Combinational Path Delay and Number of LUT's occupied

Floating-point unit in cordic Xilinx IPs	LUTs	Combinational path Delay (ns)
Wave pipelining in cordic	38	3.649ns
Complete architecture	38	3.649ns

X. VERIFICATION OF DESIGN SUMMERY

In this study, the proposed CIFM architecture is implemented in Verilog HDL and logic simulation is done in Verilog Simulator the synthesis and FPGA implementation is done using Xilinx 9.1. The design is optimized for speed and area using Xilinx, Device Family is Virtex-5, Devicexc5vlx30 , Package ff324, Speed grade: -3. The device is made up of multiplexers and LUTs. It has been found that for Xilinx Virtex4 family, the delay of the proposed architecture is 3.649 ns with the area (cell usage) of 38

XI. CONCLUSION

Floating point unit add/sub and shifting process has implemented by using cordic algorithm a single bit add/sub and shifting process have done here mainly the concept is perform application of cordic angles in avionics in navigation system so, by applying cordic in floating point unit the synthesis have done this is main concept of this project. Cordic angles calculation has done by using formula z(n)=arc tan (2^n) initial formula is 1 (radian) = $2^{20}/2\pi$i by this way real time iteration angles are obtained this is implemented in pipelining in cordic method also simulated from this timing response is reduced and in wave pipe lining in cordic method also implemented here combinational path delay is reduced and timing response is reduced when compared to cordic in fpu and pipelining in cordic so, wave pipelining in cordic in reduced technique.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Hence this cordic algorithm is implemented to design avionics aircraft navigation system to reduce critical path delay and reduce complexity of the circuit then area will be reduced and speed will increase and power consumption is reduced and timing response will reduced these are major techniques are observed by implementing cordic algorithm in navigation system.

REFERENCES

1. Alexander, C., S. Ishikawa, & M. Silverstein, 1977 *A Pattern Language*, Oxford University Press.
2. Brooch, G., 1993 *Object-Oriented Analysis and Design*, Benjamin Cummings.
3. Claude-Pierre Jeannerod, Herv´e nochel, Christophe Monat, Member, IEEE, and Guillaume Revy, "Faster floating-point square root for integer processors", Laboratoire LIP (CNRS, ENSL, INRIA, UCBL) .
4. Coglianese, L., W. Tracz, D. Batory, M. Goodwin, S. Shafer, R. Smith, R. Szymanski, & P. Young *Collected Papers of the Domain-Specific Software Architectures (DSSA) Avionics Domain Application Generation Environment (ADAGE)*, Document ADAGE-IBM-93-09, IBM Federal Sector Company,1994.
5. David Goldberg, "What Every Computer Scientist Should Know About Floating-Point Arithmetic", *ACM Computing Surveys, Vol 23, No 1, March 1991, Xerox Palo Alto Research Center*, 3333 Coyote Hill Road, Palo Alto, California 94304 .
6. De Champeaux, D, D. Lea, & P. Faure. 1993 *Object Oriented System Development.* Addison Wesley.
7. G. Kappen, T.G. Noll, 2006 "Mapping of multioperable GNSS receiver algorithms to a heterogeneous ASIP based platform", *Proceedings of the International Global Navigation Satellite Systems Society (IGNSS)* Symposium 2006, Surfers Paradise, Australia.
8. J. Duprat and J. M. Muller, 1993 "The CORDIC Algorithm: New Results for fast VLSI Implementation", *IEEE Transactions on Computers*.
9. K. Keutzer, S. Malik, A. R. Newton, 2002 "From ASIC to ASIP: The Next Design Discontinuity", *ICCD Proceedings*.
10. Prof. Kris Gaj, Gaurav, Doshi, Hiren Shah, "Sine/Cosine using CORDIC Algorithm".
11. S. Fischer, P. Rastetter, M. Mittnacht, F. Griesauer, P. Silvestrin, "AGGA-3 in an Avionic System", *ESA Workshop on Spacecraft Data Systems and Software*.
12. Samuel Ginsberg, "Compact and Efficient Generation of Trigonometric Functions using a CORDIC algorithm", Cape Town, South Africa.
13. T. G. Noll, 2004 "Application Domain Specific Embedded FPGAs for SoC Platforms", *Invited Survey Lecture, IrishSignals and Systems Conference* 2004 (ISSC'04), Jun.
14. Taek-Jun Kwon, Jeff Sondeen, Jeff Draper, "Design Trade-Offs in Floating-Point Unit, Implementation for Embedded and Processing-In-Memory Systems", *USC Information Sciences Institute, 4676 Admiralty Way Marina del Rey*, CA 90292 U.S.A.
15. Yamin Li and Wanming Chu, 1996 "A New Non-Restoring Square Root Algorithm and Its VLSI Implementations", International Conference on Computer Design (ICCD‟ 96), October, Austin, Texas, USA.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)