Design and Implementation of Low Latency CORDIC Algorithm in Field Programmable Gate Array

Mr. K Murugesh1, Mr. Thippleswamy K H2 and Mrs. Shilpa V3
1&2 Department of ECE, Don Bosco Institute of Technology, Bangalore, India.
Email: muruvel@dbit.co.in1, thipeswamykh@dbit.co.in2
Department of ECE, Sapthagiri College of Engineering, Bangalore, India.
Email: shilpav@sapthagiri.edu.in3

Abstract— Coordinate Rotation Digital Computer is a general purpose digital computer for the real time computations. At present the need of Multipliers is increasing in various real time domains which have the core operations like trigonometric computations, Multiplication and Division. The three important parameters which affect the current trends of VLSI domains are power, speed and area. Usage of conventional multipliers in the design will result in more power and area consumption. CORDIC algorithm is an effective replacement of the Multipliers by using adders/ subtractor and shifters. The objective of this work is to design and implement a low power Iterative and high performance Pipelined architecture for trigonometric computations using CORDIC algorithm. The two different VLSI architectures were designed and functionally verified using Modelsim 10.1b Simulator. The designs were implemented in Field Programmable Gate Array (FPGA) using Altera Quatrus II Synthesis tools.

Index Terms— CORDIC, Multipliers, Iterative CORDIC, Pipelined CORDIC, VHDL, FPGA

I. INTRODUCTION

The acronym of COordinate Rotation Digital Computer is CORDIC. The concept of CORDIC algorithm is based on the principles of 2D geometry [1]. CORDIC offers the constructive condition of not requiring either multiplication or division blocks, instead it works only with adder-subtractor and shifter. The CORDIC algorithm was first described in 1959 by Jack E. Volder [1] for the computation of various functions which uses multiplication as core operations. The add-sub and the shifter operations involved in this algorithm are less complex and due to that the CORDIC algorithm is well-matched for the VLSI based implementation of any designs. This algorithm is used in various applications like digital signal processing, Real time systems, Communication systems, Image processing and so on. The Field Programmable Gate Array has its own advantage of reconfiguration and software flexibility with hardware performance. So the designs can be reconfigured and implemented in the hardware for any number of times. The real time signals will be in real or integer format. At present, computing these real values are difficult and the numbers in real format cannot be computed like other number formats.

The recent trend in the software developments leads to the flexibility of implementing the CORDIC algorithm using the VHDL libraries instead of using any other additional components for the number
representation. In this work the characteristics of CORDIC algorithm is used to design the Iterative CORIC core and the flexibility of Field Programmable Gate Arrays used to design the Pipelined architecture for the CORDIC core which eliminates the Iterative cycles of a same block. The designs were simulated using Mentor Graphics Modelsim simulator, a simulator which is well-organized for implementing the real time signal values and synthesized using Altera Quatrus II 12.1sp1 synthesis tools [8].

II. CORDIC ALGORITHM

The CORDIC algorithm can be derived by using the rotation mode and the vectoring mode. In vectoring mode, the coordinates \((X_0, Y_0)\) are rotated until \(Y_0\) converges to zero. In rotation mode, initial vector \((X_0, Y_0)\) starts aligned with the X axis and is rotated by an angle of \(\theta_i\) every cycle, so after \(n\) iterations, \(\theta_n\) is the obtained angle. In this work, rotation mode is used to approximate sine and cosine functions. The concept of CORDIC algorithm is to approximate the desired rotation angles through a series of deflected angle which is fixed and is relevant to do the base operation [1], [2]. The algorithm reduces the computation to simple addition, subtraction and bit shifts operations. The CORDIC algorithm operates by decaying the preferred angle into the subjective sum of a set of predefined elementary rotation angles such that it can be accomplished with simple shift and adds operations. The “Figure 1” shows the two dimensional geometry of vector rotation for calculating the desired angles.

As shown in the “Figure 1”, by rotating the vector \((x, y)\) to vector \((x', y')\) the general equation CORDIC algorithm can be formulated as \(\theta = \Phi - \theta_h\) and by using the rectangular co-ordinates, the cosine and sine values can be computed as

\[
\cos \theta = \frac{x}{r} \quad \text{and} \quad \sin \theta = \frac{y}{r}
\]

(1)

The “(1)” can be re-arranged as. By using this co-ordinates, the expression for \(Ox'\) is formulated as

\[
Ox' = x' \quad \text{and} \quad x' = r \cos \Phi
\]

(2)

We know that \(\Phi = \theta + \theta_h\), and by using this in “(2)” we get,

\[
x' = r \cos (\theta + \theta_h) = r \cos \theta \cos \theta_h - r \sin \theta \sin \theta_h
\]

(3)

Let’s Consider, that the vector \(v'\) is rotated in the anticlockwise direction and by using the relations \(x = r \cos \theta\) and \(y = r \sin \theta\) in “(3)”, a set of expressions for \(Ox'\) and \(Oy'\) is derived as

\[
\begin{align*}
Ox' &= x' = x \cos \theta_h - y \sin \theta_h \\
Oy' &= y' = y \cos \theta_h + x \sin \theta_h
\end{align*}
\]

(4)

For Anti-Clockwise rotation of the Vector \(v'\),

\[
\begin{align*}
Ox' &= x' = x \cos \theta_h - y \tan \theta_h \\
Oy' &= y' = y \cos \theta_h + x \tan \theta_h
\end{align*}
\]

(5)

In this work, the anti-clockwise vector rotation is considered for deriving the equations of CORDIC algorithm. The “(4)” can be re-arranged as,

\[
\begin{align*}
x' &= \cos \theta_h [x - y \tan \theta_h] \\
y' &= \cos \theta_h [y + x \tan \theta_h]
\end{align*}
\]

(6)

We know that \(\tan \theta_h = \sin \theta_h / \cos \theta_h\), by using the same in “(6)” we get,

\[
\begin{align*}
x' &= \cos \theta_h [x - y \sin \theta_h / \cos \theta_h] \\
y' &= \cos \theta_h [y + x \sin \theta_h / \cos \theta_h]
\end{align*}
\]

(7)
In the “(7)” the hardware utilization of \( \tan \theta_n \) can be reduced by estimating \( \tan \theta_n \) to \( 2^{-n} \) and the binary multiplication operation between the terms \( x \) and \( y \) with \( \tan \theta_n \) in the “(7)” is replaced by simple shift operation [3]. The next multiplication operation is between cosine terms with rest of the expression. In order to eliminate this operation from “(7)” the cosine is simplified and since \( \cos(\theta_n) = \cos(-\theta_n) \) it can be replaced by a constant \( K(n) \) for a fixed number of iterations even if \( \theta_n \) is negative. Another parameter can also be introduced for pre-defining the number of rotations required to get the desired angle. The value of \( K(n) \) can be derived from \( \cos(\arctan(2^{-n})) \). The term \( \theta_n \) is the angle of rotation for \( n \) number of rotations. The product of \( K(n) \) represents the K factor as

\[
K = \prod_{i=0}^{n-1} K(i) = \cos \theta'_{0} \cos \theta'_{1} \ldots \cos \theta'_{n-1}
\]  

Where, \( n \) denotes the present iteration value and \( (n+1) \) denotes the next iteration value. The Angle accumulator can be used to monitor the difference in the desired angle & current angle. The equation for the Iterative rotation of the vector \( v' \) and angle accumulator can be expressed as “(8)”

\[
\begin{align*}
x(n+1) &= K(n)[x(n) - y(n)d(n)2^{-n}] \\
y(n+1) &= K(n)[y(n) + x(n)d(n)2^{-n}] \\
z(n+1) &= z(n) - d(n)\arctan(2^{-n})
\end{align*}
\]  

Where \( \arctan(2^{-n}) \) is obtained from LUT. The value of \( d(n) = \pm 1 \), decided based on the \( z(n) \).

\[
d(n) = \begin{cases} 
+1, & \text{if } z(n) \geq 0 \\
-1, & \text{if } z(n) < 0
\end{cases}
\]  

III. CORDIC ARCHITECTURE

In this section, the implementation of CORDIC algorithm is discussed at the architectural level. The hardware utilization for computing of sine and cosine values using CORDIC algorithm [2] is discussed. For every rotation, coordinates of the rotated point and the angle to be rotated is calculated. The Inputs for the “(9)” are \( x = 0.607 \) (Scaling factor, obtained using “(10)”), \( y = 0 \) and for \( z = \) Input angle (the angle for which the trigonometric values is required). The hardware resources required for the implementation of these expressions are three registers for storing the values of \( x, y \) & \( z \). A Shifter for implementing \( (2^{-n}) \), a look up table for storing the values of \( \arctan(2^{-n}) \) and the Adder or Subtractor blocks for the arithmetic operations as per the value of \( d(n) \).

The Architectural view of CORDIC Algorithm is shown in “Figure 2”. For each iteration, the architecture of CORDIC algorithm requires approximately two shifts and three add/sub operations. In the above “Figure 2” the Selection lines for the muxes are used to choose the inputs either from the primary input or from the feedback of the CORDIC core’s output. The two different VLSI architectures of CORDIC core are discussed in the next sub sections.

A. Iterative CORDIC Architecture.

The hardware implementation of Iterative architecture is discussed in this section [4],[6].The “(9)” from CORDIC Algorithm section is implemented in the Iterative architecture. The block diagram for the Iterative architecture of CORDIC core is shown in “Figure 3”. The shifting operation is represented by the notation “\( \gg \) (n-1)” .In the “Figure 3”, muxes chooses the Inputs viz., \( x_0, y_0 \) and \( z_0 \) as primary inputs for the first
iteration. For rest of the iterations, the output is fed back to the input through a MUX which is controlled by a select signal $SL$. The adder / subtractor block is represented by the modules with “±” sign in “Figure 3”. The value of $d(n)$ determines the operations to be carried out either addition or subtraction. To obtain defined output, $N$-number of iterations is required for $N$-number of bits [3]. The Iterative architecture requires less area for the physical implementation and also it consumes less power for calculating fewer angle inputs. Thus the Iterative architecture of CORDIC core is more efficient in terms of area, power and throughput, only for the computing of few trigonometric values.

B. Pipelined CORDIC Architecture.

The Pipelined architecture for the general equations of CORDIC algorithm is discussed in this section [6]. While calculating multiple angles or a stream of angles, Iterative architecture becomes inefficient in terms of performance and power consumption. In order to calculate the stream of angles using the CORDIC algorithm the Pipelined CORDIC core is implemented using an array of shifters and adder/subtractor stages [6], which overcomes the demerits of Iterative architecture. The merits of Pipelined CORDIC core are high frequency of operation and high throughput.

The number of Pipelined stages for calculating a stream of angles using Pipelined architecture can be decided based on the number of vector rotation required to get the desired result. The Pipelined CORDIC architecture is shown in “Figure 4”. This architecture is designed by replicating the idea of Iterative CORDIC core; here Pipelined registers are added in between the CORDIC core stages to improve the throughput [6] and also the design can be reset at any time without losing the computed values of previous clock cycle. The Pipelined registers are introduced in the design to store the values for a specified period or clock cycles. The values can be checked at any clock period. In this, Pipelined CORDIC architecture for $N$ stages of Pipelined structures $N-1$ Pipelined registers are required. In a Very Large Scale Integration domain, always there is a trade of between area, speed and the power. By using the Pipelined architecture for the applications will lead to the increase in area with much improvement in throughput and performance.
IV. IMPLEMENTATION OF CORDIC ALGORITHM

The Iterative and Pipelined CORDIC architectures were designed using VHDL coding language. In this work a few specific libraries and packages from the updated version of vhdl (vhdl2008 from vhdl93) is used for representing the real numbers in the fixed point format. Due to these libraries the design of Iterative and Pipelined architectures are less complex and gives accurate results than any other style of coding. The inputs and outputs of these architectures are of 32 bit size.

The Functional verification of these designs were done using Mentor Graphics Modelsim 10.1 b Simulator shown in the “Figure 5”. The next stage after the functional verification is to synthesize the designed CORDIC architectures using Altera Quatrus II 12.1sp1 synthesizer [8]. Unlike other synthesizer this Altera Quatrus II 12.1sp1 synthesizer is updated for the usage of these vhdl2008 libraries and packages. At the same time this synthesizer supports all three vhdl version viz., vhdl87, vhdl93 and vhdl2008. These are the merits of the tools used in this work viz., Mentor Graphics Modelsim 10.1 b Simulator and Altera Quatrus II 12.1sp1 synthesizer.

V. RESULTS

The Simulation and Synthesis results obtained for the Iterative and Pipelined CORDIC architectures using Modelsim Simulator and Quatrus II 12.1sp1 Synthesizer is discussed in this section. The “Figure 7” shows the output waveform from the Modelsim 10.1b Simulator for the Iterative CORDIC architecture, in this output waveform decimal representation is used as the radix for representing the computed values, for an angle of 30°.

![Figure 7. Output waveform of Iterative CORDIC core](image)

![Figure 9. Output waveform of Pipelined CORDIC core](image)
The Simulation waveform for the Pipelined CORDIC architecture with the modules of CORDIC core and Pipelined registers is shown in the “Figure 9” in decimal radix format. From the “Figure 9” it is clear that after N-numbers of clock cycles the Pipelined CORDIC architecture produces computational results for every clock cycles. Hence the Pipelined CORDIC core is efficient for calculating a stream of angles than computing fewer angle values. As mentioned in the Implementation of CORDIC algorithm section, Altera Quatrus II synthesizer is used for implementing the Iterative and Pipelined CORDIC architectures in Field Programmable Gate Array. The “Figure 10” shows the Block symbol file generated for the Pipelined CORDIC core from the Quatrus II synthesizer. The libraries used in the designs should be synthesized before synthesizing the Iterative and Pipelined CORDIC design. The Inputs for the CORDIC core modules are Clock, reset, \( x (2:-30), y (2:-30), z (8:-24) \) and the outputs from the design are \( x \cos (2:-30), y \sin (2:-30), z \text{out} (8:-24) \). Where the output parameter \( y \sin \) and \( x \cos \) gives the sine and cosine value of any given angle after the \( N \) - numbers of clock cycles respectively. Also the Block symbol file is created for the Pipelined register; these Pipelined registers are inserted between the CORDIC cores to form the Pipelined architecture as shown in “Figure 10”.

The Net-list RTL view for the Pipelined CORDIC architecture is also generated using the Quatrus II synthesizer which gives and clear idea about the connections made in the Pipelined Structure using the CORDIC core and the Pipelined registers. In this work the Pipelined architecture is designed using seven stages of CORDIC cores as shown in the “Figure 11” Net list-RTL view of Pipelined CORDIC architecture. In the FPGA Implementation the resource utilization details are necessary to compare and analyze the resources utilized by different designs. From the results of the simulator and synthesizer, the analysis can be done for different parameters. The device EP4CGX150DF31I7AD which belongs to the Cyclone IV GX Altera FPGA family is used in this work.
The “Figure 12” shows the resource utilization summary for the Iterative CORDIC and Pipelined CORDIC Architectures. In the “Figure 12” it’s clearly shown that, in Pipelined architecture there is an increment of 68% in utilization of logic elements comparing to the Iterative architecture.

Thus when the area is considered as a prime constraint the Iterative architecture is efficient for computing less number of trigonometric angle values. In the Pipelined architecture more resources are used because of the replication of CORDIC core and Pipelined registers. The “Figure 13” shows the Power Dissipation Summary for the Iterative and Pipelined CORDIC architectures. The Performance analysis was done between the Iterative and CORDIC architecture on the basis of resource utilized, power and throughput. The Analysis results are tabulated as shown in the Table.1. When the power is considered as the prime factor, the Pipelined Architecture design consumes more than Iterative architecture for the computation of single angle value computation.

| TABLE I. PERFORMANCE ANALYSIS |
|------------------------------|------------------|------------------|
| SL.NO | ANALYSIS PARAMETER | ITERATIVE CORDIC | PIPELINED CORDIC |
| 1     | LOGIC ELEMENTS UTILIZED | 1983            | 3347             |
| 2     | REGISTERS UTILIZED       | 226             | 1287             |
| 3     | TOTAL POWER DISSIPATION  | 253.54 mW       | 506.69 mW        |
| 4     | THROUGHPUT – FOR COMPUTING SINGLE ANGLE VALUE OF 30° | 800 ns | 1400 ns |
| 5     | THROUGHPUT FOR COMPUTING STREAM OF ANGLE VALUES (-90° TO 90°) | 5700 ns | 2000 ns |
In real time application the computations [7] of trigonometric values is continuous and not limited to a fixed number of computations. As shown in the Table. 1 For computing trigonometric values from -90° to +90°, the Iterative CORDIC architecture takes 5700 ns which is greater than the Pipelined architecture. The Pipelined architecture is having better performance results and high frequency of operation than the Iterative CORDIC architecture.

VI. CONCLUSION

In this paper, we have presented a FPGA implementation of the CORDIC algorithm in Iterative and Pipelined VLSI architectures. On comparing the performance of these two architectures for real time applications [7], the throughput of the Pipelined CORDIC architecture for computing the trigonometric values are 1.85 times greater than that of Iterative CORDIC architecture. Hence the Pipelined Architecture is proved to be the best in terms of throughput and performance for real time applications. The increment in the power consumption can be further reduced by using the low power techniques for real time applications.

ACKNOWLEDGMENT

This is to place on record our deep indebtedness to our guide Mr. BABU C (Assistant Professor, ASE, Bangalore) who has been a source of inspiration all throughout the course of this work. Also we wish to express our gratitude to Dr. Mahesh P K Professor (ECE Department, Don Bosco Institute of Technology Bangalore, India) who has always extended a helping hand.

Above all, I bow my head for Parents and God, the almighty for all the blessings showered on me.

REFERENCES