ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 5, Issue 6, June 2016

Implementation of Phase Shifter using CORDIC on FPGA for RADAR Application

Triveni.C[1], Sudhakara Reddy.P[2], *Member IEEE*

***Abstract*— The phase shift is the core element in DSP design. The technique is used to implement phase shift is the complex multiply. This operation requires four multipliers and two adders. By using multipliers the consumption of power also increased. Coordinate Rotation Digital Computer (CORDIC) is an efficient and versatile algorithm that can implement a phase shift without using multipliers. In this work implementing the complex multiply for phase shift calculation using CORDIC algorithm. It reduces the number of computations while using shift-and-add operations during processing. The main applications of Complex multiply using CORDIC are Typical digital receiver front end, FMCW Radar front end airport ground control application, Industry surveillance and security systems**

***Index Terms*— CORDIC, PHASE SHIFT, FPGA, RADAR etc...**

I. INTRODUCTION

The RADAR means Radio Detection and Ranging which is used as a communication medium to find objects in space. The detected objects are at distances which are not observed visually in space. The RADAR provides a capability of detecting objects over extreme ranges through air at night and in unsuitable weather. The block diagram of typical digital receiver front end receiver is shown in figure 1. The Typical digital receiver is one that is completely digital with a wide dynamic range A/D converter that operates directly on the signal received at the antenna terminals. The Typical digital receiver is designed around 1990.

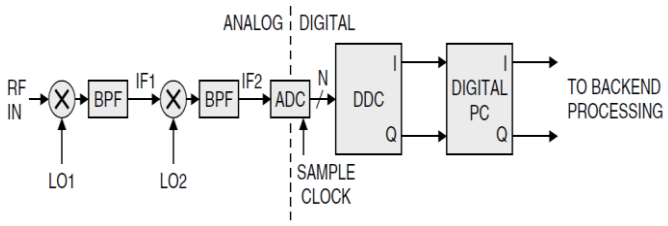

Fig 1.Typical radar digital receiver front end

This system incorporated analog pulse compression (PC). It also included several stages of analog down conversion, in order to generate baseband in-phase (I) and quadrature (Q) signals with a small enough bandwidth that the ADCs of the day could sample them. The digitized signals were then fed into digital Doppler/MTI and detection processors.

The RF input usually passes through one or two stages of analog down conversion to generate an Intermediate Frequency (IF) signal that is sampled directly by the ADC. A digital down converter (DDC) converts the digitized signal samples to complex form at a lower rate for passing through a digital pulse compressor to backend processing.

In this Section I describes Introduction of typical digital receiver front end application for complex phase shifter using CORDIC, Section II describes phase shift using standard complex multiply. Section III describes CORDIC based standard complex multiply. Section IV explains implementation results. Finally concluded in Section V.

Manuscript received May, 2016.

***Triveni.C**, M.Tech, DECS, Srikalahasteeswara Institute of technology, JNTU Anantapur., Srikalahasti, India,*

***Dr. Sudhakara Reddy Penubolu**, Member IEEE, Department of ECE, Srikalahasteeswara Institute of technology JNTU Anantapur, Srikalahasti.*

II. PHASE SHIFT USING STANDARD COMPLEX MULTIPLY

A. Phase Shift

The phase shift is a core element in DSP design, and there are several techniques available to implement one. Phase shift is any change that occurs in the phase of one quantity, or in the phase difference between two or more quantities. φ is sometimes referred to as a *phase shift* or *phase offset*, because it represents a "shift" from zero phase. Phase difference is the difference, expressed in degrees or time, between two waves having the same frequency and referenced to the same point in time. Two oscillators that have the same frequency and no phase difference are said to be **in phase**. Two oscillators that have the same frequency and different phases have a phase difference, and the oscillators are said to be **out of phase** with each other. The amount by which such oscillators are out of phase with each other can be expressed in degrees from 0° to 360°, or in radians from 0 to 2π. If the phase difference is 180 degrees (π radians), then the two oscillators are said to be in **anti phase.**

B. Standard Complex Multiply

The straightforward approach is to perform a complex multiply, as shown in Figure 2. For example, the complex input sample is denoted as A + jB, which is multiplied by the complex coefficient C + jD to produce (AC – BD) + j(AD + BC) in order to effect the phase shift. This operation requires four multipliers and two adders shows in equations (1) and (2).

After some manipulation, the following can be shown:

$$I = (AC - BD) = D(A - B) + A(C - D) \qquad (1)$$

$$Q = (AD + BD) = C(A + B) - A(C - D) \quad (2)$$

Noting that the final term is the same in both equations (1) and (2), we see that this complex multiplier can be implemented with only three real multipliers and five real adds. This can be important if real multipliers are at a premium. Figure 3 shows a block diagram of this architecture.

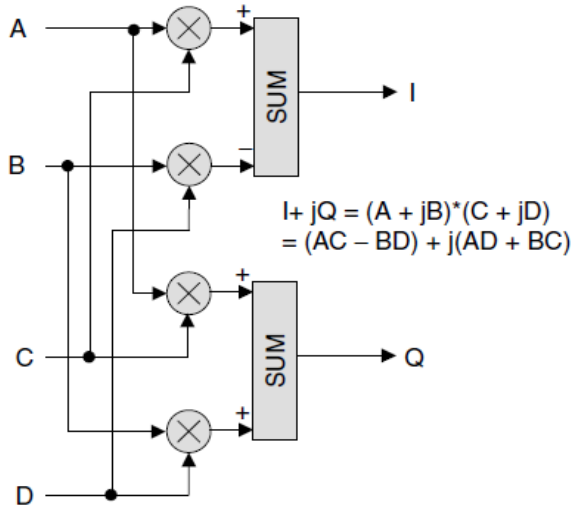

Fig 2 .Standard complex multiply

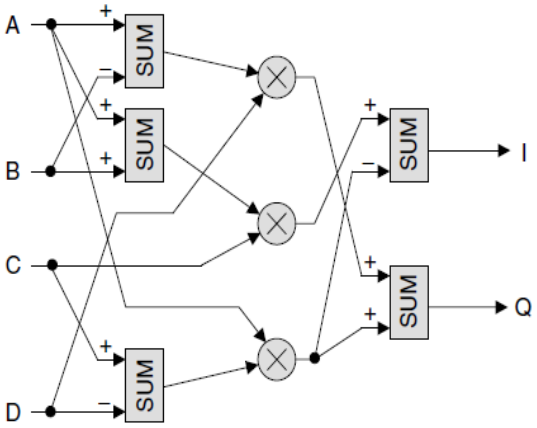

Fig 3 .Complex multiply with three real multipliers

III. CORDIC BASED STANDARD COMPLEX MULTIPLY

The equations that shift the phase of complex number $I_0 + jQ_0$ by an angle Θ to produce $I_1 + jQ_1$ are as follows:

$$I_1 = I_0(\cos(\theta)) - Q_0(\sin(\theta)) \quad (3)$$
$$Q_1 = I_0(\sin(\theta)) + Q_0(\cos(\theta)) \quad (4)$$

These equations can be manipulated to provide

$$I_1 = \cos(\theta)[I_0 - Q_0(\tan(\theta)) \quad (5)$$
$$Q_1 = \cos(\theta)[Q_0 + I_0(\tan(\theta)) \quad (6)$$

The CORDIC algorithm takes advantage of this relationship to approximate an arbitrary phase shift by implementing multiple stages of phase shifts, where the tangent of the phase shift in each successive stage is the next smaller fractional power of 2, and multiplication by this number can be implemented by shifting the input data bits an integer number of places. The first few stages are as follows:

$$I_1 = \cos(\theta_0)[I_0 - Q_0(\tan(\theta_0))]$$

$$= \cos(\theta_0)[I_0 - Q_0(1)] \quad (7)$$
$$Q_1 = \cos(\theta_0)[Q_0 + I_0(\tan(\theta_0))]$$

$$= \cos(\theta_0)[Q_0 + I_0(1)] \quad (8)$$
$$I_2 = \cos(\theta_1)[I_1 - Q_1(\tan(\theta_1))]$$

$$= \cos(\theta_1)[I_1 - Q_1(\tfrac{1}{2})] \quad (9)$$
$$Q_2 = \cos(\theta_1)[Q_1 + I_1(\tan(\theta_1))]$$
$$= \cos(\theta_1)\left[Q_1 + I_1\left(\frac{1}{2}\right)\right] \quad (10)$$

Table.1 shows these parameters for an eight-stage CORDIC processor. Each row of the table represents successive iterations of the algorithm. The **$\tan(\Theta_i)$** column shows the factor by which the I and Q values are multiplied for each iteration. Note that these values are fractional powers of 2, so the multiplication can be realized by shifting the binary I and Q values right by i places. The **Θ_i** column shows the arctangent of this factor, which can also be thought of as the phase shift applied during each iteration.

TABLE.1 CORDIC Parameters for First Eight stages

i	$\tan(\theta_i)$	θ_i(deg)	$\cos(\theta_i)$	P$[\cos(\theta_i)]$
0	1	45.000	0.707107	0.707107
1	1/2	26.565	0.894427	0.632456
2	1/4	14.036	0.970143	0.613572
3	1/8	7.1250	0.992278	0.608834
4	1/16	3.5763	0.998053	0.607648
5	1/32	1.7899	0.999512	0.607352
6	1/64	0.8951	0.999878	0.607278
7	1/128	0.4476	0.999970	0.607259

The $\cos(\Theta_i)$ column shows the cosine of this angle, which should be multiplied by the result of each iteration, as shown in the equations above. In actual applications, however, this cosine multiplication is not performed at every iteration. At each stage, the implied factor that needs to be multiplied by the IQ outputs of the stage in order to provide the correct answer is the product of all of the cosines up to that point, as shown in the **P$[\cos(\Theta_i)]$** column. For a large number of iterations, this product of cosines converges to a value of 0.607253. In most cases, this scaling can be compensated for in later stages of processing. The inverse of this factor, 1.64676 for a large number of iterations, is the processing gain imposed on the IQ results of the CORDIC. If integer arithmetic is performed, an extra bit should be provided at the most significant end of the adders in order to accommodate this increased signal level.

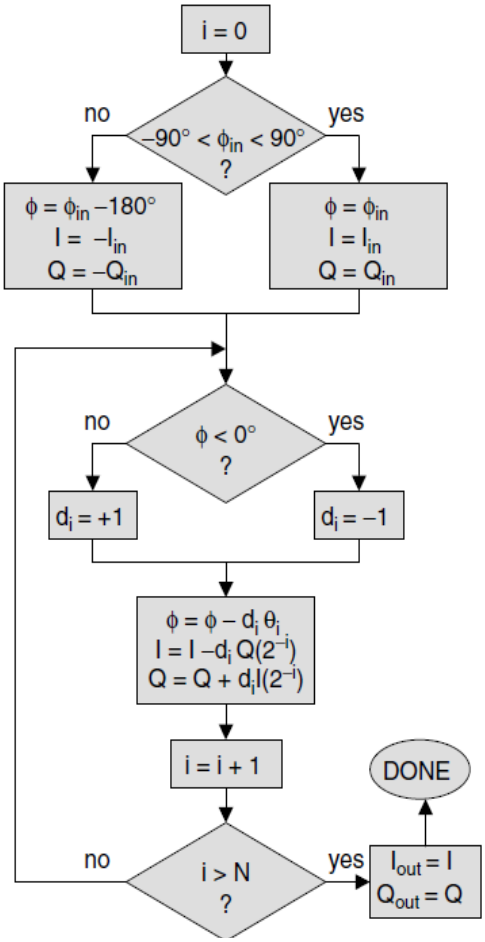

Fig 4. CORDIC algorithm flow chart

Figure 4 is a flow chart that represents the CORDIC algorithm to implement a phase shift. The inputs to the algorithm are the I_{in}, Q_{in}, and ϕ_{in} (the desired phase shift). The variable i will keep track of the processing stage being performed and is initialized to zero. The basic algorithm can perform a phase shift between ±90°. If the desired phase shift is outside of that range, the input I and Q values are first negated, imposing a 180° phase shift, and 180° is subtracted from the desired phase shift. The new phase shift is now within ±90°, and the algorithm proceeds normally.

Next, the algorithm loops through N iterations with the goal of driving the residual phase error, ϕ, to zero. In each iteration, a new ϕ is calculated by subtracting or adding the phase shift for that stage ($\boldsymbol{\Theta}_i$ from the table) to the previous value of ϕ. If ϕ < 0, $\boldsymbol{\Theta}_i$ is added to ϕ. Otherwise, $\boldsymbol{\Theta}_i$ is subtracted from ϕ. In each stage, the Q (or I) input is divided by a factor of 2^i by shifting the number to the right by i bits, and then added to or subtracted from the I (or Q) input, depending on the sign of ϕ. The variable i is incremented and the process repeats until i > N, at which point the phase-shifted results are available.

Figure 5 is a block diagram of an eight-stage CORDIC processor that implements a phase shift, where each stage represents an iteration in the flow chart. An N-stage processor provides a phase shift that is accurate to within $\pm\boldsymbol{\Theta}_N$ degrees (from the table), so the more stages in the processor, the more accurate the answer. The input I and Q values change on the rising edge of an assumed sample clock. In the first stage, the I value is either added to or subtracted from the Q value in the ADD/SUB block. The control block on the bottom of the figure determines whether additions or subtractions are performed at each stage, based on the algorithm described previously. If the ADD/SUB block in the Q channel performs an addition, the same block in the I channel will perform a subtraction, and vice-versa. The result of the ADD/SUB blocks is stored in a register (REG) on the next clock edge and passed to the next stage of processing. In this implementation the last block labelled (PASS/INV) performs the required inversion of I and Q if the desired phase shift is beyond the $\pm 90^0$ range of the algorithm. The final multiplication by a constant is optional, as described earlier.

The architecture shown in Figure 5 is a good example of a pipelined processor, in which a portion of the computation is performed and the result is stored in a register on each rising edge of the sample clock and passed to the next stage of processing. The processor would still function if the registers were removed. However, in that case, when the input I and Q values changed, the final output would not be useable until the results of the new input values rippled through all of the stages of processing, which would generally be an unacceptably long period of time. In a pipelined processor, a small portion of the total calculations is performed at a time, and the result is stored in a register and passed to the next processing stage. This architecture provides a higher throughput than the non pipelined version, which means that the final result can be produced at a much higher sample rate, which is inversely proportional to the delay of a single stage. The latency of a pipelined processor refers to the delay experienced between the time a new data sample is entered into the processor and the time that the result based on that input is available on the output. The eight-stage, pipelined CORDIC processor shown in the figure 5, would have a latency equivalent to eight clock periods and a throughput equivalent to the clock rate.

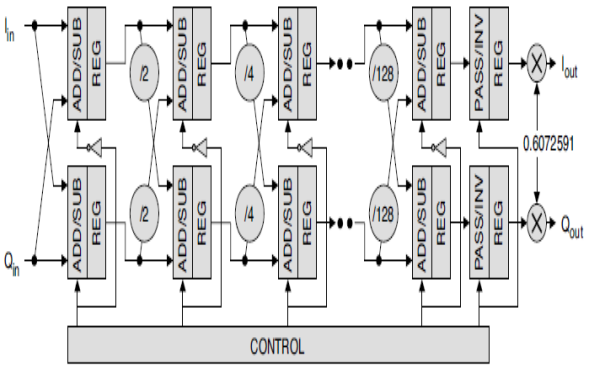

Fig 5. Eight stage CORDIC processor

IV. SIMULATION RESULTS

This includes the simulation and synthesis of sine cosine generator implemented on the target device XILINX SPARTAN-3E using XC3S1600E. The sine cosine generator uses simple pipelined architecture based on CORDIC algorithm. Area and delay reports are also included for the

corresponding target device. The CORDIC employed uses circular co-ordinate system and is operated in rotating mode. Hence, only phase is given as input and x, y values are given internally in the program or hardware.

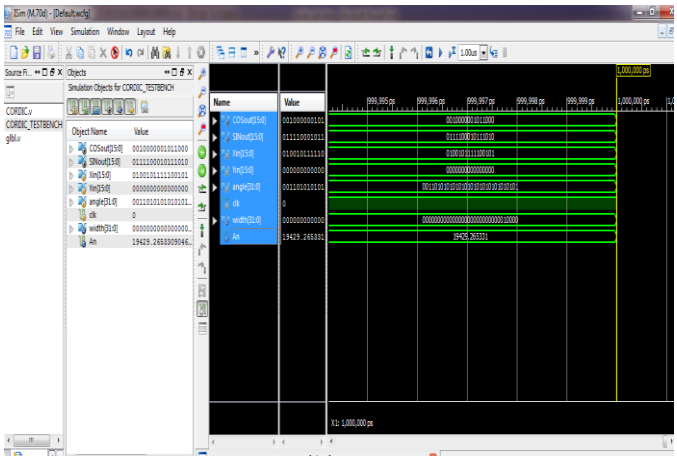

Fig.6.simulation results

The code written in verilog is simulated using Xilinx ISE. After simulation of main block it has to be functionally verified. So, a test program with arbitrary inputs is written, shows the compilation of test bench program by giving various test inputs to the master program. As it is a pipelined architecture inputs can be given at every clock pulse and the cosine and sine values for corresponding inputs will output after eight clock cycles as it. The two outputs for every angle input are sine function, cosine function .

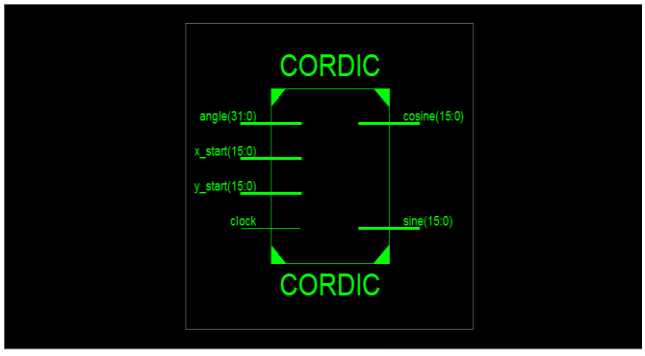

Fig 7. RTL schematic

Figure.7 XILINX window showing synthesized RTL schematic. After the simulation of code in verilog the code is synthesized using XILINX ISE design suit. The table 2. gives the device utilization summary of SPARTAN-3E when sine cosine generator using CORDIC algorithm is implemented on it.

TABLE 2: DEVICE UTILISATION SUMMARY

Logic Utilization	Used	Available	Utilization
Number of Slices	978	14752	6%
Number of Slice Flip Flops	949	29504	3%
Number of 4 input LUTs	1843	29504	6%
Number of bonded IOBs	97	250	38%
Number of GCLKs	1	24	4%

The timing report includes the total time delay for the output to appear after giving input. Table 3 shows the simulation of time .

TABLE 3: TIMING REPORT

1.	MAXIMUM FREQUENCY	175.952MHz
2.	MINIMUM PERIOD REQUIRED	5.683NS
3.	MINIMUM INPUT ARRIVAL TIME BEFORE CLOCK	5.726NS
4.	MAXIMUM OUTPUT TIME AFTER CLOCK	4.040NS

V. CONCLUSION

In this work, I proposed and implemented a standard complex multiplier using a CORDIC Algorithm. The complex phase shifter is targeted for SPARTAN 3E and it requires 6% of its total number of slices with a time delay of 5.683 ns. As the implemented design is a pipelined one, it is more efficient than bit serial approach and is more accurate and advantageous than bit serial architecture. CORDIC is not only used in computation of elementary functions but also used for tracking of moving targets in space, distance between the multiple targets, reconfigurable systems.

REFERENCES

[1] J. E. Volder, “The CORDIC trigonometric computing technique,” IRE Transactions on Electronic Computers, vol. 8, no. 3, pp. 330–334, 1959.
[2] J. S. Walther, “A unified algorithm for elementary functions," in Proceedings of the AFIPS Spring Joint Computer Conference, pp. 379–385, May 1971.
[3] R. Andraka, “A survey of CORDIC algorithms for FPGA based computers,” in Proceedings of the 6th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA ’98), pp. 191–200, February 1998.
[4] B.Lakshmi and A.S. Dhar "CORDIC Architectures: A Survey" in Hindawi Publishing Corporation, VLSI Design, Volume 2010, Article ID 794891, 19 pages.
[5] Ron Warner, Cort Lansenderfer "Low cost, Low power FPGAs Provide Ideal Platform for CORDIC Applications" in lattice Semiconductor Corporation.
[6] Jan Raby and Konstantin’s sarrigeorgidis "Ultra low power CORDIC processor for wireless communication algorithms" Journal of signal processing 38,115-130, 2004 @ kluwer academic publishers. Manufactured in the Netherlands.

[7] Jeongseon Euh, Jeevan Chittamuru, Wayne Burleson CORDIC VECTOR INTERPOLATOR FOR POWER-AWARE 3D COMPUTER GRAPHICS [8] Y. H.Hu, "CORDIC-based VLSI architectures for digital signal processing," IEEE S
[8] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing,"IEEE Signal Processing Magazine, vol. 9, no. 3, pp. 16–35,July 1992.
[9]James J. Alter and Jeffrey O.Coleman, Radar digital Signal Processing,25th chapter, radar handbook,2nd Ed,1990.
[10]J. G. Proakis and D. G. Manolakis, Digital Signal Processing: Principles, Algorithms and Applications. Upper Saddle River, NJ: Prentice-Hall, 1996.
[11] Pramod K. Meher and K. Sridharan, 50 Years of CORDIC Algorithms, Architectures and Applications, IEEE Member,2009
[12] J. S. Walther, "A unified algorithm for elementary functions," in Proceedings of the 38th Spring Joint Computer Conference, Atlantic City, NJ, 1971, pp.379–385.
[13] D. S. Cochran, "Algorithms and accuracy in the HP-35," Hewlett-Packard Journal, pp. 1–11, June 1972.
[14] S. Wang, V. Piuri, and J. E. E. Swartzlander, "Hybrid CORDIC algorithms,"IEEE Transactions on Computers, volume 46, no. 11, pp. 1202–1207, November1997.
[15]S. Wang and E. E. Swartzlander, "Merged CORDIC algorithm," in IEEE International Symposium on Circuits Systems (ISCAS'95),1995, volume 3, pp.1988–1991.
[16] B.Gisuthan and T. Srikanthan, "Pipelining flat CORDIC based trigonometric function generators," Microelectronics Journal, volume 33, Pp.77–89, 2002.

C.TRIVENI,
Completed B.Tech in Srikalahasteeswara Institute of Technology, Srikalahasti.
Pursuing M.Tech in Srikalahasteeswara Institute of Technology, Srikalahasti.

Dr P.SUDHAKARA REDDY,
Associate Professor in Dept. of ECE and coordinator for Research and Development centre and Principal of Srikalahasteeswara Institute of Technology, Srikalahasti, 517640.He has 20 years of experience .He is doing research work on VLSI Architecture in DSP. He had memberships in various bodies such as IEEE, IACSIT, ISTE, IETE, SSI etc. He is interested in other Research fields in VLSI in wireless and Biomedical Engineering in Medicine and Digital Arithmetic. He published nearly 25 research publications in various International journals and conferences.