IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p-ISSN: 2278-8735
PP 31-40
www.iosrjournals.org

A Reconfigurable Cordic Based Fft

Jismi K[1]
[1](*Department of ECE ,Muslim Association College of Engineering, Venjaramoodu)*

Abstract: *CORDIC (COrdinate Rotation for Digital Computers) is an ideal candidate for the implementation of low power FFT processor, because it uses set of shift-add algorithms which requires less complex hardware than the conventional method which is very well suited for VLSI implementation. This work implements FFT on a reconfigurable CORDIC only processor array. The paper compares the different CORDIC architectures with respect to their area, speed, and power analysis especially in two different major styles iterative and parallel structures. Also implemented the CORDIC based 8-point FFT processor. The results show that 2-point purely CORDIC based FFT guarantees reduced power consumption and area than 2-point butterfly structure, when compared with previous works from the literature. All the designs were designed in VHDL, simulated using ModelSim simulator and Implemented using Synopsis ASIC synthesis tools.*

Keywords: *Vector rotation, Iterative CORDIC, Parallel CORDIC, Fast Fourier Transform(FFT),FPGA.*

I. Introduction

CORDIC is an iterative arithmetic computing algorithm capable of evaluating various elementary functions using a unified shift-and-add approach. It is a hardware efficient algorithm used for high speed computations mostly in digital signal processing applications, which are dominated by microprocessors with single cycle multiply-accumulate instructions and special addressing modes. For a wide variety of DSP algorithms, CORDIC based VLSI architectures are very appealing alternatives to the architectures based on conventional multiply-and-add hardware. The CORDIC algorithm is found in numerous applications, such as pocket calculators,and in mainstream DSP objects, such as adaptive filters, FFTs, DCTs, demodulators, and neural networks. They can also be used in high speed satellite communication.

Jack E. Volder in 1959, derived CORDIC algorithm for the calculation of trigonometric functions [1], from the general equations for vector rotation and later generalized by Walther to solve a broader range of equations, including the hyperbolic equations, multiplication, division and conversion between binary and mixed radix number systems [2] of DSP applications, such as Fourier Transform. The CORDIC algorithm has become a widely used approach to elementary function evaluation when the silicon area is a primary constraint. The implementation of CORDIC algorithm requires fewer complex hardware than the conventional method and is particularly well-suited for applications in which cost (chip gate count has to be minimized) is much more important than speed.

The main characteristic of a signal processing system is having a high complexity and real-time operation.FFT acts as the basic conversion operation of frequency domain and time domain. On many occasions, the FFT operation is required to be real-time and fast. Therefore, improving the performance of the FFT processor, reducing its size and improving the speed become the key point of the design. With the increase of FFT size, the performance of FPGA will be reduced [3] because of the storage consumption and resource utilization. Butterfly unit is the basic module in FFT structure. It's important to reduce its area, as well as to improve throughput. CORDIC FFT has been shown to be an alternative to implement butterfly operation because it can finish multiplication by adders and shifters instead of multipliers and reduce the storage consumption, however, large amount of iterations in iterative CORDIC leads to high hardware cost which can be overcome by parallel CORDIC structure. With the widely use of FFT, configurable FFT is needed to meet different systems.

II. Cordic Algorithm

There are two ways in CORDIC algorithm for calculation of trigonometric and other related functions they are vector rotation mode and vector translation mode. Both methods initialize the angle accumulator with the desired angle value. In vector translation mode the coordinates (x_0,y_0) are rotated until y_0 converges to zero. This work discuss about the vector rotation mode of CORDIC. Initial vector (x_0,y_0) starts aligned with the x axis and is rotated by a specific angle during every cycle, so that after n iterations, gets the desired angle. The main idea consists of 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 sin Θ_n and

cos Θ_n in its second and first components respectively [4]. If a vector *V* with coordinates (*x*, *y*) is rotated through an angle α then a new vector V' can be obtained with coordinates (*x*', *y*') where *x*' and *y*' can be obtained using *x*, *y* and α by the following method.

$$\begin{bmatrix} x^{(i+1)} \\ y^{(i+1)} \end{bmatrix} = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i \\ \sin\alpha_i & \cos\alpha_i \end{bmatrix} \begin{bmatrix} x^{(i)} \\ y^{(i)} \end{bmatrix} \quad (2.1)$$

This can be rewritten as:

$$\begin{bmatrix} x^{(i+1)} \\ y^{(i+1)} \end{bmatrix} = \cos\alpha_i \begin{bmatrix} 1 & -\tan\alpha_i \\ \tan\alpha_i & 1 \end{bmatrix} \begin{bmatrix} x^{(i)} \\ y^{(i)} \end{bmatrix} \quad (2.2)$$

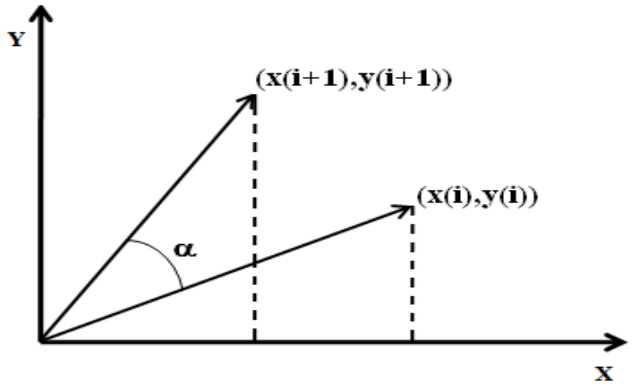

Fig 1: Vector rotation.

Where tan α_i can be restricted to $d_i\, 2^{-i}$, so the multiplication can be converted into an arithmetic right shift [5] , with $d_i = \pm 1$. The first factor $\cos\alpha_i = 1/\sqrt{(1+2^{-2i})}$ .For rotation over an arbitrary angle α, $-\pi/2 \le \alpha \le \pi/2$, it can be decomposed as :

$$\alpha = \sum_{i=0}^{n} d_i \arctan 2^{-i} \quad (2.3)$$

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:

$x^{(i+1)} = k^i [x^i - y^i d_i 2^{-i}]$ (2.4)

$y^{(i+1)} = k^i [y^i - x^i d_i 2^{-i}]$ (2.5)

where, *i* denotes the number of rotation required to reach the required angle of the required vector, $k^i = \cos(\arctan(2^{-i}))$ and $d_i = \pm 1$.

The product of the k^i's represents the scale factor [6] K :

$K = \Pi^{n}_{i=0} \cos\alpha i = \Pi^{n}_{i=0} 1/\sqrt{(1+2^{-2i})}$ (2.6)

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

$z^{(i+1)} = z^{(i)} - d_i I_{(i)}$. (2.7)

where $I_{(i)}$ is the LUT entries.

The sum of the rotating angles gives the desired angle

$$\theta_n = \sum d_i \arctan 2^{-i} \quad (2.8)$$

III. Different Cordic Archetecture

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/substractor units and a look up table to store

the values of $\alpha_i=\tan^{-1}2^{-i}$. The $d_i$factor (-1 and 1) selects the shift operand or its complement. The initial inputs to the architectures are $X_0=1$, $Y_0=0$.

The structure uses 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. These two blocks are inevitable for any application as the input range cannot be predicted always.The CORDIC core can converge angles only at the interval $[-\pi/2, \pi/2]$ or -90° to +90º. The pre-processing unit takes in angles of any range and converges it to the interval $[-\pi/2, \pi/2]$. It keeps record of the quadrant of the input angle of any range. The preprocessing unit passes this quadrant information to the post processing unit. The post-processing unit uses this quadrant information to fix the sign of outputs. Hence we can generate Sine and Cosine waves, if we give a continuous range of angles as input to the CORDIC processor.

1.1 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, is shown in Fig 3.1, which 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.To obtain sine and cosine values of a given angle z_0, iterative structure takes the value of (x_0,y_0) 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 [7]. Hence, it requires a minimum of N clock cycles for required output.

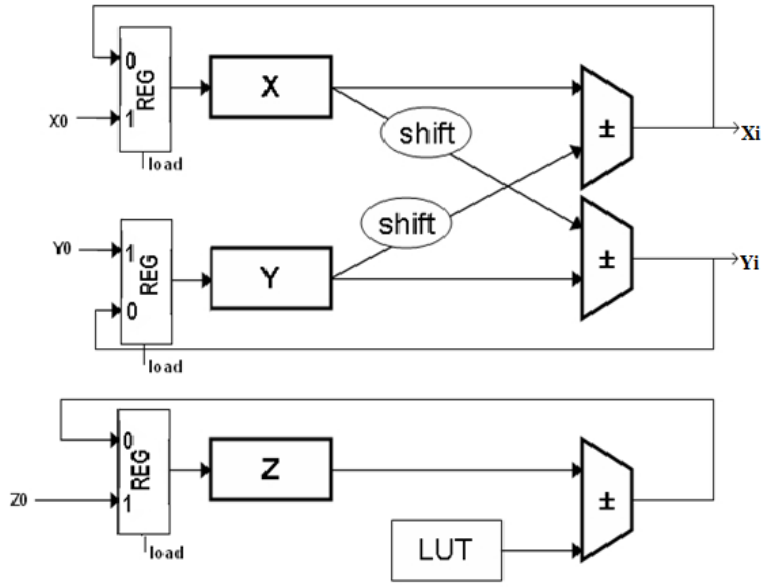

Fig 3.1: Iterative CORDIC.

1.2 Parallel/Cascaded CORDIC Architecture:

This architecture uses multiple instances of Iterative CORDIC structure. Fig 3.2 shows CORDIC core with parallel architectural configuration, which 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 parallelly 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.

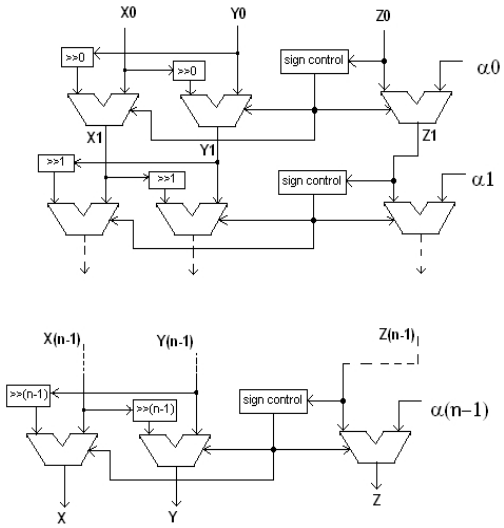

Fig: 3.2: Parallel CORDIC.

IV. Fast Fourier Transform

A Fast Fourier transform (FFT) is an efficient algorithm to compute the Discrete Fourier transform (DFT) and it's inverse. FFTs are of great importance to a wide variety of applications, from digital signal processing and solving partial differential equations to algorithms for quick multiplication of large integers. The DFT is defined by the formula

$X(k)= \sum_{n=0}^{N-1} x(n)e^{-(j\,2nk\,/N)}$ (4.1)

Evaluating these sums directly would take (N^2) arithmetical operations. An FFT is an algorithm to compute the same result in only (N log N) operations. The evaluation by CORDIC requires only log N complexity. Decimation is the process of breaking down something into its constituent parts. Decimation in time involves breaking down a signal in the time domain into smaller signals, each of which is easier to handle.

1.3 FFT using Butterfly structure:

The basic computation using FFT is called butterfly computation which s shown in Fig4.1:

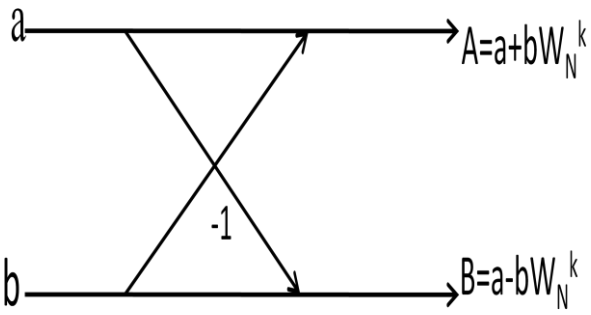

Fig: 4.1 Basic butterfly computations in the decimation-in-time.

If the input (time domain) signal, of N points, is x(n) then the frequency response X(k) can be calculated by using the DFT.

$X(k)=\sum_{n=0}^{N-1} x(n)W^{nk}{}_N \quad for\ k=0,1,2,3....N$ (4.2)

where $W^{nk}{}_N = e^{-j\,2\pi kn\,/N}$

By using the above butterfly computation technique an 8 point FFT can be represented asa structure consisting of three types of blocks. Four 2-point computing blocks, two 4-point computing blocks and one 8-point computing block shown in Fig:4.2 [11]:

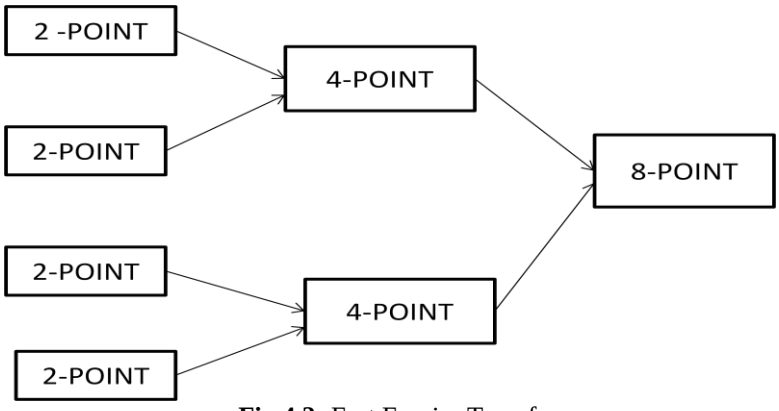

Fig 4.2: Fast Fourier Transform

4.2 Fft Using Cordic:

The Fast Fourier Transform is the most frequently used DSP computing algorithm in modern digital signal processing systems. For an N-point DFT calculation, each N-point DFT can be divided into two N/2 point sub problems. Recursively applying this procedure will lead to the popular radix-2 FFT algorithm. The basic operation for FFT algorithm is Butterfly operation. A basic Decimation In Time FFT operation is:

$A=a + b\ W_N^k$ (4.3)

$B=a - b\ W_N^k$ (4.4)

Here a, b, A and B are all complex numbers. So the equation 4.3 and 4.4 requires complex number multiplication and complex number addition. The complex number multiplication can be done with m iterations for an m bit bit complex number , which takes only one clock cycle if implement using parallel CORDIC processors. The complex addition will need just one more clock cycle. For small N, a FFT computation can be realized directly with a network of CORDIC processors.

If the input (time domain) signal, of N points, is x (n) then the frequency response *X(k)* can be calculated by using the FFT.

$X(k)= \sum_{n=0}^{N-1} x(n)e^{-(j2nk/N)}$ for k=0,1,2,3……N-1 (4.5)

For a real sample sequence f (n), where n is {0, 1,...., (N-1)} DFT can be defined as:

$F(k)=\sum_{n=0}^{N-1} f(N)\ [\cos\left(\frac{2\pi}{N}\right)kn - jsin\ \left(\frac{2\pi}{N}\right)kn]$ (4.6)

Where $\cos\left(\frac{2\pi}{N}\right)kn$ is the real part and $\sin\left(\frac{2\pi}{N}\right)kn$ is the imaginary part.

Thus equation 4.4 can be rewrite as:

$F(k)= F_r(k) + F_i(k)$ (4.7)

The real part in equation 4.7 is the cosine value and the imaginary part is the sine value output from the CORDIC processor.

First consider the basic CORDIC algorithm.All the input samples are given a vector rotation by the defined angle in each of the transforms. The CORDIC unit can iteratively rotate an input vector $\begin{bmatrix}Xi\\Yi\end{bmatrix}$by a target angle Θ through small steps of elementary angles Θ_i, to generate an output vector $\begin{bmatrix}Xi+1\\Yi+1\end{bmatrix}$. The operation can be represented mathematically as:

$\begin{bmatrix}Xi+1\\Yi+1\end{bmatrix}= \begin{bmatrix}cos\theta\ sin\theta\\-sin\theta cos\theta\end{bmatrix}\begin{bmatrix}Xi\\Yi\end{bmatrix}$ (4.8)

The rotation by a certain angle can be achieved by the summation of some elementary small rotations given by: $\theta = \sum_{i=0}^{15}\theta i$ for a 16 bit machine.

To derive CORDIC based FFT, stop the recursion at n=2 [10]. In this case

$\begin{bmatrix}Xi+1\\Yi+1\end{bmatrix}= \begin{bmatrix}1 & 1\\-1 & 1\end{bmatrix}\begin{bmatrix}Xi\\Yi\end{bmatrix}$

The first matrix can be decomposed into

$$\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2} & \\ & -\sqrt{2} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \quad (4.9)$$

This equals a CORDIC rotating the input values by $\frac{\pi}{4}$, followed by scaling of $\sqrt{2}/-\sqrt{2}$.

As the CORDIC elements are real valued but the input values are complex valued, the complex CORDIC operation has to be separated into real valued operations. If we assume two complex numbers *a*, *b* ∈C, the result of the complex rotation will be: (t=1/$\sqrt{2}$)

$$T \cdot \begin{bmatrix} a_r \\ b_r \end{bmatrix} + jT \cdot \begin{bmatrix} a_i \\ b_i \end{bmatrix} \quad (4.10)$$

Thus the complex butterfly can be calculated by using two real valued CORDIC, by applying the CORDIC operation to the real and imaginary part of the inputs independently.

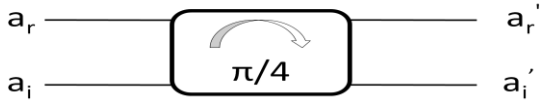

Fig 4.3: one real valued cordic

Which results in a complex CORDIC, called type I, by rotating two complex valued vectors by $\frac{\pi}{4}$, as shown in Fig 4.4.

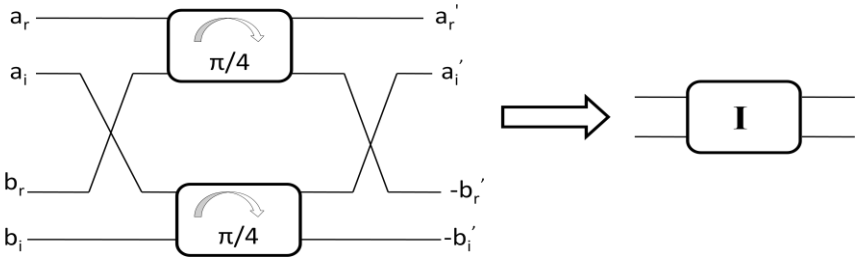

Fig 4.4 : Internal structure of complex CORDIC type I

The second scaling factor has got a negative sign. So in each stage of the FFT the sign reversed results are just combined with other sign reversed results. Therefore a -3$\frac{\pi}{4}$rotation is applied to the results in these cases to project the result from the third quadrant back into the first one. This equals a multiplication of the complex input value by –T. The complex CORDIC performing this operation is called type II.

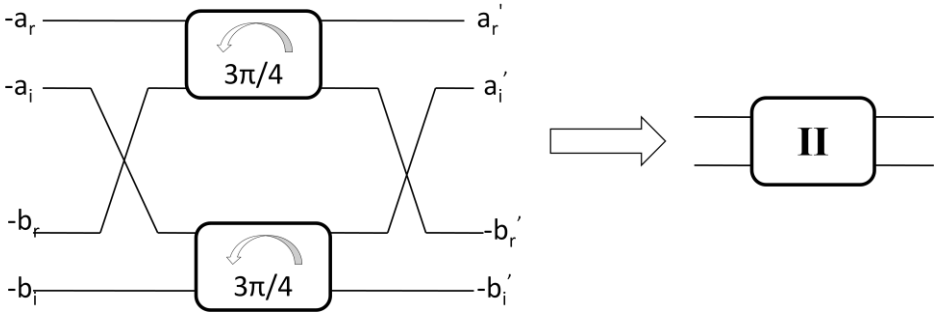

Fig 4.5 : Internal structure of complex CORDIC type II

The above two structures represent 2 point FFT with CORDIC modules.
A complete 8-FFT based on CORDIC operations is shown in Figure 4.6.

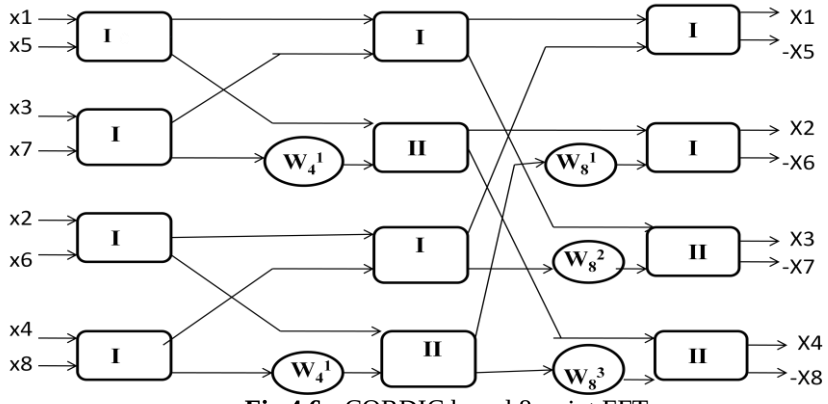

Fig 4.6 : CORDIC based 8-point FFT.

Type I and Type II CORDIC structures are used instead of Butterfly computation. The twiddle factors shown are the same as used for the standard FFT.
The twiddle factor (w_y^x)is derived as:

$$W_y^x = e^{-\frac{j2\pi x}{y}}. \qquad (4.10)$$

Here $W_4^1 = e^{-90}$.which can be implemented by simple CORDIC rotation with an angle of 90°.This rotation will take an extra one clock cycle. This circuit can be extended up to N point structure. Similarly W_8^1 , W_8^2 , W_8^3 can be implemented by simple CORDIC rotations with angle 45° , 90°, 135° respectively.

V. Implementation And Result

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

1.4 Sine and Cosine Waveform generation using complete CORDIC core

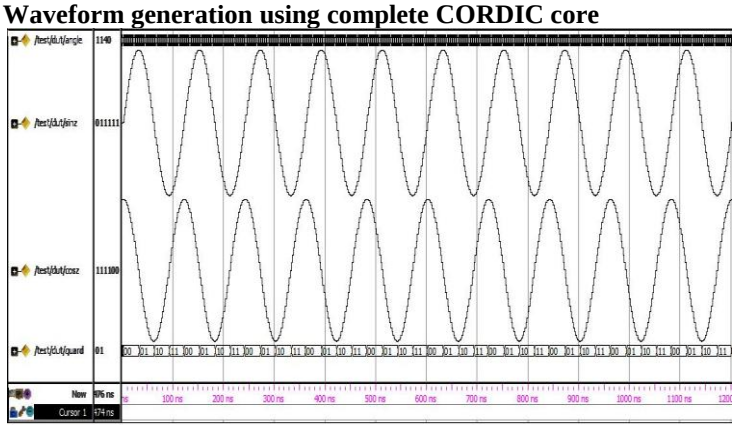

Fig 5.1: Sine, Cosine wave generated using CORDIC complete core.

1.5 Cordic 2-Point Fft

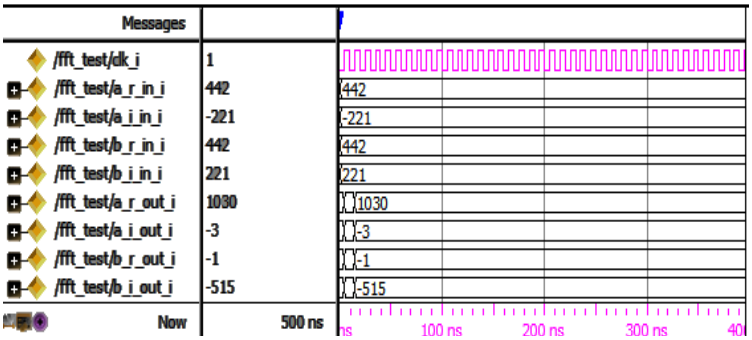

Fig 5.2: Cordic Fft 2point Output

1.6 Comparison of 8 point FFT with Matlab output

Table 1: Comparison Results Of 8 Point Fft Matlab And Vhdl

Instants	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X8**
Input	1	0	1	0	1	0	1	0
Matlab output	4	0	0	0	4	0	0	0
VHDL output	3.85	0	0	0	3.85	0	0	0

5.4 Area and Power analysis of different CORDIC structures

Area, Power and Timing parameters of different architectures are analyzed by using Synopsis ASIC synthesis tool. The work done based on 13 micron technology.

5.4.1 Area Analysis

Table II: comparison of area of different cordic structures.

Parameter	Number of ports	Number of net	Number of cells	Number of references	Combinational area	Non-combinational area	Total area (nm^2)
Iterative CORDIC	50	224	70	24	1144	500	1644
Parallel CORDIC	49	1587	1587	111	14925	336	15261

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.

5.4.2. Power Analysis

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

Table ii: comparison of power of different cordic structures.

Parameter	**Iterative CORDIC**	**Parallel CORDIC**
Cell Internal Power	0.00 nW	0.00 nW
Net Switching Power	104.6365 µW	2.1496 mW
Total Dynamic Power	104.6365 µW	2.1496 mW

5.4.3 Timing Analysis

Table Iii: Comparison Of Timing Of Different Cordic Structures

Point	Iterative CORDIC		Parallel CORDIC	
	Incr	Path	Incr	Path
load_reg/CP (FDS2L)	0.00	0.00r	0.00	0.00r
load_reg/Q (FDS2L)	23.46	23.46 r	23.46	23.46 r
done_out (out)	0.00	23.46 r	0.00	23.46 r
data arrival time	---	23.46	---	23.46

From the tables it is evident that parallel CORDIC is much faster than the iterative CORDIC. Though it consumes a higher area, Parallel CORDIC structures will be preferable for high speed applications. Hence preferred the Parallel CORDIC Structure for the FFT implementation in this work.

1.7 Area and Power analysis of CORDIC FFT

1.7.1 Device Utilizations Summary

As the parallel CORDIC modules are instantiated many times in the 8 point FFT structure, the number of slices and LUTs increases, and it is about 20 times as that of parallel CORDIC structure, whereas 14 times greater that of 2 point FFT.

Tableiv: Comparison Of Timing Of Different Cordic Structures

Parameter	No. of Slices	No. of Slice Flip Flops	No. of 4 input LUTs	No. of IOs	No. of bonded IOBs	No. of GCLKs
2 point FFT	512	-----	1024	129	129	1
8 point FFT	7569	928	14848	513	513	1

1.7.2 Power Analysis

Table V: Comparison Of Power Of Different Cordic Structures

Parameter	Cell Internal Power	Net Switching Power	Total Dynamic Power
2 point FFT	0.00 nW	16.0453 mW	16.0453 mW
8point FFT	0.00 nW	437.6234 mW	437.6234 mW

1.7.3 Area Analysis

Table Vi: Comparison Of Area Of Different Cordic Structures

Parameter	Number of ports	Number of net	Number of cells	Number of references	Combinational area	Non-combinational area	Total area(nm^2)
2 point FFT	129	131	2	2	28676	672	29348
8point FFT	513	1187	17	17	795615	9744	805359

Since The CORDIC processor get instantiated in the 8 point FFT structure many times the area is nearly 50 times greater than CORDIC structure.

1.7.4 Comparison with Butterfly structures

Table Vii: Results Of Butterfly And Cordic Structures

Instance	Area	Total Dynamic power
Butterfly structure[3]	3345µm	583.4µW
CORDIC structure	29348nm	16.0453 mW

Results shows that the FFT using purely CORDIC structure gives less power and reduced area system, when compared with Butterfly structure.

VI. Conclusion

A tradeoff area/speed will determine the right structural approach to CORDIC FPGA implementation for an application. An iterative CORDIC uses lesser hardware than parallel 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 CORDIC is much higher compared to that of Iterative CORDIC. This difference in hardware units has caused an increased power usage by Parallel structures but it is having a gain of high speed.

This work also discussed about CORDIC based FFT algorithm. This algorithm is best suited for the implementation in reconfigurable CORDIC processor fields. The work compares the different CORDIC architectures with respect to their area, speed, and data throughput performance especially in three different major styles iterative, parallel and pipelined structures. Then implemented the CORDIC based 2-point FFT and 8-point FFT processor. The work also compares the performance of 2 point FFT with optimized butterfly structure and CORDIC structure. From the obtained results it was possible to note that, the proposed CORDIC radix-2 butterfly has been proved to be power efficient and area efficient, when compared with the original one.

The future work we intend to implement IFFT, DHT, DCT and CZT calculations using reconfigurable CORDIC only modules

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]. Renato Neuenfeld, 2Mateus Fonseca, 1Eduardo Costa"Design of Optimized Radix-2 and Radix-4 Butterflies

[4]. from FFT with Decimation in Time",VII Latin American Symposium on Circuits and Systems (LASCAS) 2016 IEEE

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

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

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

[8]. Ramesh Bhakthavatchal, Sinith.M.S, Parvathi Nair[2], Jismi.K[3]" A Comparison of Pipelined Parallel and IterativeCORDIC Design on FPGA" , 2010 5th International Conference on Industrial and Information Systems, ICIIS 2010, Jul 29 – Aug 01, 2010 IEEE

[9]. Esteban O. Garcia, Rene Cumplido, Miguel Arias, "Pipelined CORDIC Design on FPGA for a Digital Sine and Cosine Waves Generator". Proceedings of 3rd international conference on Electrical and Electronics Engineering, pp 1-4, Sept 2006.

[10]. Proakis J. G.,Manolakis D. G., .Digital signal processing principles, algorithms and applications., Prentice Hall, Delhi, 2008.

[11]. "Implementation of a Cordic Based FFT on a Reconfigurable Hardware Accelerator",Benjamin Heyne_, Jurgen Gotze University of Dortmund, Information Processing Lab