Abstract—This electronic document is a “live” template and already defines The overview of this paper is to design a micro rotation selection block for TW based FFT. This design is mainly based on Adaptive CORDIC algorithm. CORDIC stands for Coordinate Rotation Digital Computer. Two ways to implement the CORDIC algorithm are 1. Vector mode and 2. Rotation mode. The main difference between Vector and rotation are, vector mode is used to compute an angles at given point where as Rotation mode is used to compute the sine and cosine terms at given point. This paper mainly deals with vector method by using micro rotation for complex multiplications in FFT architectures. Oi- selection (Oi- sel) block for CORDIC based Twiddle Factor (TW) architecture is a computation approach which does an angle selection. CORDIC algorithm is easy to implement trigonometric, hyperbolic and exponential functions based on micro rotation for VLSI Signal processing. The simulation results are examined using Xilinx ISE 14.5 Tool.

Keywords—Fast Fourier Transform (FFT), Adaptive CORDIC Algorithm, Oi- selection (Oi- sel).

I. INTRODUCTION

In digital signal processing the Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) are playing an important role. The fast computation method of Discrete Fourier Transform (DFT) algorithm is Fast Fourier Transform (FFT).

In 1965 [1], Cooley and Tukey was first introduced the FFT signal flow graph. So that, FFT became most utilized circuits in many of signal processing and communication applications such as CDMA [5], OFDM [6], WiMAX [2], MIMO [4], WLAN [7], and 3GPP-LTE [3]. In addition to this FFT requires a computational technique in other applications like the image processing application of Fourier-Domain Optical Coherence Tomography (FD-OCT) [9], Synthetic Aperture Radar (SAR) [10], and the multimedia application of Digital Video Broadcasting-Terrestrial or Digital Video Broadcasting-Handheld (DVB-T/DVB-H) [8].

FFT architectures are of two types. They are Fixed-point and Floating-point architecture. The problem with fixed point architecture is Fixed-point Arithmetic Error (FAE) [11]. In systems fixed point or floating point FFT implementations are used for high speed with low resource cost or High Precision with wide data range.

In Twiddle Factor (TF) implementation, there are two categories. Namely, Look-Up Table (LUT) [14]-[17] and Direct computation based on Coordinate Rotation Digital Computer (CORDIC) algorithm [12],[13],[18]. Where the LUT ‘s uses a floating point multiplication and it stores the trigonometric constants where as the CORDIC is used to make a direct computation.

The advantages of LUT based design is high precision, low latency, and reasonable cost. But the problem with LUT based technique suites for only a limited number of FFT-point than a large number of FFT-points. Here by increasing the number of FFT-points the number of stored elements in LUT also increases. Therefore, there is a memory requirement problem. This problem can be overcome by many other techniques such as binary tree decomposition algorithm [21], memory access optimization [22], memory minimization method [19], and Read-After-Write latency optimization [20]. In these techniques also there is a trade-off between memory reduction, performance of accuracy, and the throughput rate.

For large-point FFT systems and Application Specific Integrated Chip (ASIC) implementation, a direct computation method is selected i.e., CORDIC. CORDIC algorithm provides a memory free solution for Twiddle Factor (TF). But the main disadvantage of CORDIC algorithm is high latency because of more number of iterations. To overcome this disadvantage we can move on with Adaptive CORDIC (ACor) method and it was introduced by Y. H. Hu et al. in 1993 [23]. The improvements in the method was introduced by Hong Thu Nguyen et al. in 2015 [24]. The main idea of the ACor method is to reduce the number of iterations and giving the equivalent or better accuracy performance. By reducing the number of iterations, the ACor method gives low-resources, low-latency, and high-precision.

II. BASIC CORDIC TECHNIQUES

Let us consider, the two vectors [X,Y] of the same magnitude. And the second vector is obtained by rotating the first vector with an angle of (\( \Phi \)).

Now we can calculate \( V_n = [X_n,Y_n] \) based on the input vector and the rotational angle \( (\Phi) \). The required equations are given below:

\[
X_n = X_0 \cos(\Phi) - Y_0 \sin(\Phi)
\]

\[
Y_n = X_0 \cos(\Phi) + X_0 \sin(\Phi)
\]

(1) Where \( \Phi \) is a rotation angle.
which indicates the direction of next rotation is used to reduce the angular error. The difference equation is based on logic as

\[ Z_{i+1} = Z_i - d_i \cdot \tan(\emptyset_i) \] (6)

where \( Z_i \) (i = 0, 1, ………, N-1). \( Z_i \) indicates the remaining rotation before performing the rotation by \( \emptyset_i \). And \( d_i \) is the direction of angle of rotation, which indicates the direction of rotation is

\[
\begin{align*}
&d_i = \begin{cases} 
-1, \quad i \leq 0 \\
+1, \quad i > 0
\end{cases}
\end{align*}
\] (7)

Fig. 1: Rotation of vector \([X,Y]\) by \( \emptyset \) degrees.

To concatenate angles \( \emptyset_i \), (i=0,1,2,……..., N-1) and to evaluate the angle \( \emptyset = \sum_{i=0}^{n} \emptyset_i \). The input and output to the Yi+1 = cos (\( \emptyset_i \)) (Yi + Xi . 2-i)

Xi+1 = Xi cos (\( \emptyset_i \)) – Yi sin (\( \emptyset_i \))

Yi+1 = Yi cos (\( \emptyset_i \)) + Xisin(\( \emptyset_i \))

(2)

In the above equation, common term cos (\( \emptyset_i \)), and the equations can be rewritten as,

Xi+1 = cos (\( \emptyset_i \)) (Xi – Yi tan(\( \emptyset_i \)))

Yi+1 = cos (\( \emptyset_i \)) (Yi + Xi tan(\( \emptyset_i \)))

(4)

where tan(\( \emptyset_i \)) = ±2-i, (i = 0, 1, 2, ………., N-1).

Under this condition, the above equation becomes cos(\( \emptyset_i \)) = cos (arctan (2-i)) =

because, we know that

\[
\cos(x) = \pm \frac{1}{1 + \tan^2(x)} 1 + 2^{-2i}
\]

\[
1
\]
\[ X_{i+1} = \cos(\theta_i) \cdot (X_i - Y_i \cdot 2^{-i}) \]
\[ Y_{i+1} = \cos(\theta_i) \cdot (Y_i + X_i \cdot 2^{-i}) \]

(5) Now, to divide equation (8) by \( \cos(\theta_i) = 1 + 2^{-2i} \)

\[ X_{i+1} \cdot a_i = X_i - d_i \cdot Y_i \cdot 2^{-i} \]
\[ Y_{i+1} \cdot a_i = Y_i + d_i \cdot X_i \cdot 2^{-i} \]

Where \( a_i = 1 + 2^{-2i} \). At the right side the computation is performed as shift-and-addition parts. So the amplified gain \( a_i \) is obtained \( X_{i+1} \) and \( Y_{i+1} \).

The gain is compensate by multiply a constant \( A_i \) on both sides. Let \( A_{i+1} = a_i \cdot A_i \). The recursive equations is obtained as

Fig. 2: Organizing computations of the transform. For some specific angle \( \theta_i \), multiplication by \( \tan(\theta_i) \) can be replaced by an arithmetic shift and some sign multiplication.

A. Determining Rotation Directions

Define For the composite rotation of an angle \( \theta \) is uniquely defined by the elementary rotation direction is sequence \( (d_0, d_1, \ldots, d_{N-1}) \).

An angle accumulator is used to determine the sequence. The accumulator accumulates the elementary rotation angles, \( X_{i+1} \cdot A_{i+1} = X_i \cdot A_i - d_i \cdot Y_i \cdot A_i \cdot 2^{-i} \). \( A_{i+1} = Y_i \cdot A_i - d_i \cdot X_i \cdot 2^{-i} \).

By equating, the above steps in \( N \) iterations. The Gain can be calculated by using the equation given below.

\[ A_N = \prod_{\theta=0}^{N-1} \sqrt{1 + 2^{-2i}} \]

In this method, the scale factors are considered for computations during iterations. So that, the process is complicate and occupies more memory. The whole process will be stopped, when the \( Z \) value is larger than the threshold value.
The input angle has range of [00 to 450] for the applied ACor algorithm is noted. If the range is extended then it should be normalized. From the figure, explains about the 8-segment trigonometric circle and by using simple trigonometric transformations, the change of angle from another segment to zeroth-segment. Table I. The first 16 iteration values of tan(ø), and ø.

### Table I. The first 16 iteration values of tan(ø), and ø.

<table>
<thead>
<tr>
<th>Segment</th>
<th>ø Norm</th>
<th>X-Y swap</th>
<th>reversed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ø0</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>1</td>
<td>π/2-ø0</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>π0-π/2</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>π-π0</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>π0-π</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>3π/2-ø0</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>ø0-3π/2</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>2π-π0</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

As the ACor range is [0-45], other than the ranges values must be normalized as shown in Table 2. Z0 is used to modify the input values X0 and Y0. If necessary the input and output values can be swapped and sine can be reversed.

#### A. øi–Sel module Block

The òi – sel modules are the Z-Normalized, Arithmetic Logic Unit (ALU), Absolute Value Modifier (ABS), Set rotation (SetRot), òi look-up table (òi-LUT), and the input angle Z0 can be normalized by using the controller. The results of ALU can be positive toby Add/sub the current Zt with the òi value.
If necessary the input angle of $\theta_0$ is normalized by $Z_{\text{Normalizer}}$ to the zeroth segment from the table-II. The normalized angle is the $\text{NormZ}$ signal and the $\text{oRec_info}$ signal will carry recovery information.

The next $\theta_i$ value is evaluated by adding/subtracting the current value $\theta_i$ at the ALU and the register with $\Theta_i$-LUT forms appropriate $\Theta_i$ value. By using abs module the ALU output can be changed to positive value and at the outside $\text{oSignZ}$ signal is transferred as signed outside.

IV. RESULTS

The simulation is carried out in Xilinx ISIM Tool and the synthesis is performed in Xilinx ISE 14.5 Tool for Spartan3E FPGA Family with Device XC3S500E, Package of FG320 and Speed Grade of -5. The design utilizes 81 slices out of 4656 slices, 0 LUTs out of 9312 LUTs and has a combinational path delay of 12.047ns as shown in Figure 6. The corresponding RTL and Technology Schematics are shown in figures 7 and 8 respectively. The simulation result is shown in figure 9, where the input given is 3F49, for which as per the design, the output obtained is 3F49.
V. CONCLUSION

The design of theta I sel block was implemented in this paper. Adaptive CORDIC (ACor) algorithm is a decision making algorithm so that, it reduces the number of iterations which in turns reduces the latency compare to the conventional CORDIC algorithm.

VI. REFERENCES


[25] Hong-Thu Nguyen, Xuan-Thuan Nguyen, Trong-Thuc Hoang, DucHung Le, and Cong-Kha Pham, “Low-resource Low-latency Hybrid Adaptive CORDIC with Floating-point Precision,” IEICE Electronics Express (ELEX), vol. 12, no. 9, pp. 20 150 258–20 150 258, 2015.