CORDIC Based 16-Point FFT Processor

ALLU THANUJA
M.Tech, VLSI Design
SRM University
Chennai
thanuja.allu@gmail.com

SARADA V (Asst. Professor)
Department of ECE
SRM University
Chennai
saradasaran@gmail.com

Abstract—There is a high demand for the efficient implementation of complex arithmetic operations in many Digital Signal Processing (DSP) algorithms. The Coordinate Rotation Digital Computer (CORDIC) algorithm is suitable to be implemented in DSP algorithms because its calculation for complex arithmetic is simple. Besides, since it avoids using multiplications, adopting the CORDIC algorithm can reduce the complexity. CORDIC is implemented through repeated shift-add operations. A pipelined CORDIC based architecture for the Fast Fourier Transform Implementation (FFT) is designed and implemented in Modelsim Verilog.

Keywords: DSP, FFT, CORDIC.

I. INTRODUCTION

Fast Fourier Transform (FFT) is widely used transform in digital applications especially in communication systems. An FFT is an important processing block in these systems, which takes most of the hardware complexity. Due to demand in higher data rates in communication systems, large-point FFTs are required such as 1024/2048/4096 etc. An FFT is commonly implemented with complex multiplier which is equivalent to four real multipliers and two real adders, and a ROM to store the twiddle factors. The ROM in this type of implementation takes most of the chip area, consumes more power and degrades the speed because ROM size increases with large-point FFTs. Hence poor performance of the FFT in terms of power, speed, and area can be seen.

The Coordinate Rotation Digital Computer (CORDIC) is an arithmetic technique, which makes it possible to perform two dimensional rotations using simple hardware components. The algorithm can be used to evaluate elementary functions such as cosine, sine, arctangent, sinh, cosh, tanh, ln and exp. CORDIC algorithm appears in many applications because it uses only primitive operations like +, -, ×, ÷. CORDIC algorithm is suitable to be used in complex arithmetic operations in DSP algorithms because its calculation for complex arithmetic is simple. Besides, since it avoids using multiplications, adopting the CORDIC algorithm can reduce the complexity. CORDIC is implemented through repeated shift-add operations. A pipelined CORDIC based architecture for the Fast Fourier Transform Implementation (FFT) is designed and implemented in Modelsim Verilog.

Section 3 introduces the CORDIC algorithm in detail. Section 4 shows the results and power comparisons of both 16-point cooley-tukey and CORDIC FFT processor. Section 5 is conclusion.

II. THE COOLEY - TUKEY 16-POINT RADIX-2 FFT

The Cooley-Tukey algorithm was published in 1965. It has been the most widely used FFT algorithm. The basic idea of the algorithm is to divide the N-point DFT into M, N/M point DFTs, hence if M=2 then it is divided into two N/2 DFTs. These are called the radix-2. Similarly we have radix-4, 8, 16…etc. Cooley and Tukey demonstrated the simplicity and efficiency of the divide and conquer approach for DFT computation and made the FFT algorithms widely accepted. The Cooley–Tukey algorithm divides the DFT into smaller FFTs, so it can be combined arbitrarily with any other algorithm for the DFT.

The N-point Discrete Fourier transform (DFT) X (k) of an N-point sequence x(n) is defined as

\[ X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk} \, \quad 0 \leq k \leq N-1 \]

where the twiddle factor \( W_N^{nk} \) is given by,

\[ W_N^{nk} = e^{-j2\pi nk/N} \]

x(n) is the discrete time signal and X(k) is the signal in its frequency domain.

The Even and the Odd samples of the DFT of x[n] are calculated using the Decimation in Frequency radix-2 FFT Algorithm.

The Even samples of the DFT X[K] are given by the formula,X(2r) = DFT\(\frac{n}{2}\)\{x(n) + x\(\frac{n + N}{2}\)\} (2)
The Odd samples of the DFT X[K] are given by the formula,

\[ X(2r + 1) = \frac{1}{2} \left[ x(n) - x\left(\frac{n + N}{2}\right) \right] \]

(3)

Where, r=0, 1, …………………………(N/2)-1.

Thus figure. 1 represents the flow graph of 16-point FFT using the above radix-2 DIF algorithm.
### III. CORDIC ALGORITHM

The CO-ordinate Rotation Digital Computer (CORDIC) is a simple and efficient “shift-add” algorithm to calculate the wide range of functions including trigonometric, logarithmic, hyperbolic and linear. It is commonly used when there is no hardware multiplier is available. Vector rotations can also be used for the conversion of polar to rectangular and rectangular to polar coordinate conversions. It was developed by Jack Volder [3] in 1949 based on the observation that if we rotate a unit-length vector \((1, 0)\) by an angle \(\phi\) then its new end-point will be at \((\cos \phi, \sin \phi)\).

The algorithm can be derived from the rotation transform:

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

On rearrangement of the terms, this can be given as:

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

The implementation of these equations is still complex due to the presence of the trigonometric functions. However, if the rotation angles are restricted to values such that \(\tan \phi = \pm 2^{-i}\), the multiplication by the tangent can be greatly simplified as it can be implemented using simple shift operations. Thus, arbitrary angles can be obtained by performing a series of rotations iteratively. At each rotation, the direction of rotation is chosen by obtaining the difference between the actual angle and the angle obtained by rotation. Mathematically, it can be given as shown below.

\[
\begin{align*}
x_{i+1} &= K_i [x_i - y_i d_i 2^{-i}] \\
y_{i+1} &= K_i [y_i + x_i d_i 2^{-i}]
\end{align*}
\]

where,

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

\(d_i = \pm 1\)

The value of \(d_i\) is +1 if the angle to be obtained is greater than the current iterative angle and is -1 if the current iterative angle is greater. The value of \(K_i\) can be taken to be a constant with a value of about 0.6073 when the number of iterations is taken large.

The CORDIC rotation and vectoring algorithms as stated are limited to rotation angles between \(-\pi/2\) and \(\pi/2\). This limitation is due to the use of \(2^0\) for the tangent in the first iteration. For composite rotation angles larger than \(\pi/2\), an additional rotation is required. Volder [2] describes an initial rotation \(\pm \pi/2\). This gives the correction iteration:

\[
\begin{align*}
x' &= -d \cdot y \\
y' &= d \cdot x \\
z' &= z + d \cdot \pi/2
\end{align*}
\]

where \(d = +1\) if \(y < 0\)

\(-1\) otherwise.

There is no growth for this initial rotation. Alternatively, an initial rotation of either \(\pi\) or 0 can be made, avoiding the reassignment of the \(x\) and \(y\) components to the rotator elements. Again, there is no growth due to the initial rotation:

\[
\begin{align*}
x' &= -d \cdot x \\
y' &= d \cdot y \\
z' &= z \text{ if } d=1, \text{ or } z - \pi \text{ if } d = -1
\end{align*}
\]

where \(d = -1\) if \(x < 0\)
\[ z' = (\cos \theta, \sin \theta) \]

Figure 2: Vector rotation method of sine and cosine calculation.

While computing we start with an initial value of \( x \)-coordinate at 1 and an initial value of \( y \)-coordinate as 0. In the first iteration, the vector rotates by an angle of 45°, which gives us the first iteration result. If this angle is greater than the angle \( \beta \), the next rotation takes place in the reverse direction, else in the same direction. Finally, after the specified number of rotations, the value of \( \cos (\beta) \) is given by the \( x \)-coordinate while the value of \( \sin (\beta) \) is given by the \( y \)-coordinate.

The CORDIC is normally operated in one of two modes. The first, called rotation by Volder [2] rotates the input vector by a specified angle (given as an argument). The second mode, called vectoring, rotates the input vector to the \( x \)-axis while recording the angle required to make that rotation.

The twiddle factor angles that are required for 16-point FFT design in each stage namely stage0, stage1, stage2, and stage3 is as shown in the table 1.

<table>
<thead>
<tr>
<th>Stage 0</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twiddle factor angles</td>
<td>Twiddle factor angles</td>
<td>Twiddle factor angles</td>
<td>Twiddle factor angles</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \pi/8 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3( \pi/8 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4( \pi/8 )</td>
<td>4( \pi/8 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5( \pi/8 )</td>
<td>4( \pi/8 )</td>
<td>4( \pi/8 )</td>
<td>0</td>
</tr>
<tr>
<td>6( \pi/8 )</td>
<td>6( \pi/8 )</td>
<td>4( \pi/8 )</td>
<td>0</td>
</tr>
<tr>
<td>7( \pi/8 )</td>
<td>6( \pi/8 )</td>
<td>4( \pi/8 )</td>
<td>0</td>
</tr>
</tbody>
</table>

IV. RESULTS

The below are the simulation results for 16-point radix-2 cooley-tukey FFT and 16-point CORDIC based FFT. The inputs given are:

\[ x(n) = \{0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7\} \]

Figure 3: Cordic pipeline unit

Figure 4: Simulation results for 16-point radix-2 FFT
**V. CONCLUSION**

In this paper, the 16-point radix-2 CORDIC based FFT processor and cooley-tukey 16-point FFT processor is implemented using Verilog HDL and simulated. Also, the analysis of power is done for both the processors and the results are compared. The result shows that the CORDIC based FFT processor consumes less power than the cooley-tukey FFT processor.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>General 16-point Radix-2 FFT</th>
<th>CORDIC based 16-point FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power (mW)</td>
<td>143</td>
<td>109</td>
</tr>
</tbody>
</table>

Table.2: Comparison of power and area

**ACKNOWLEDGMENT**

The authors are thankful to the journal IJRIT for the support to develop this manuscript.

**REFERENCES**


