SRI HARI A S
sriharias2204@gmail.com

Hardware Implementation of a 16-Point Radix-2 FFT Using CORDIC-Based Twiddle Multiplication

1. Introduction

Fast Fourier Transform (FFT) is one of the most important algorithms in digital signal processing (DSP), used in applications such as wireless communication, radar, biomedical signal processing, image processing, and spectral analysis. FFT efficiently computes the Discrete Fourier Transform (DFT) by reducing the computational complexity from $O(N^2)$to $O(N\log_2 N)$.

In hardware systems such as FPGA and ASIC designs, implementing FFT using floating-point arithmetic and multipliers leads to high area, power, and delay. Therefore, this project focuses on a **fixed-point FFT architecture** that avoids hardware multipliers by using **CORDIC (Coordinate Rotation Digital Computer)** for twiddle factor multiplication.

This project implements a **16-point radix-2 decimation-in-time (DIT) FFT** using:

- Multi-stage butterfly architecture
- CORDIC-based complex rotation
- Fixed-point arithmetic
- Gain-compensated twiddle multiplication

2. FFT Algorithm

The DFT of a sequence $x[n]$is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}$$

For $N = 16$, radix-2 FFT requires:

$$\log_2(16) = 4$$

4 stages of butterfly computations.

The FFT decomposes the computation into **butterflies** that combine pairs of inputs using twiddle factors:

$$W_N^k = e^{-j\frac{2\pi k}{N}}$$

3. Overall Architecture

The fft1 module implements a **four-stage FFT pipeline**.

Input

- 16 real inputs: xin1 … xin16
- 16 imaginary inputs: yin1 … yin16
- All inputs are **16-bit signed fixed-point numbers**

Output

- 16 real FFT outputs: xout1 … xout16
- 16 imaginary FFT outputs: yout1 … yout16
- All outputs are **17-bit signed**

Stage	**Function**
Stage 1	Bit-reversal and W_0 butterflies
Stage 2	W_0 and W_4 butterflies
Stage 3	General twiddle multiplication using CORDIC
Stage 4	Final FFT stage using all twiddle factors

4. Butterfly Units

Three butterfly blocks are used for hardware optimization.

4.1 Butterfly (Stage-1)

Used when the twiddle factor is $\mathbf{W_0 = 1}$.

No rotation is required:

$$X = a + bY = a - b$$

This stage uses only **adders and subtractors**, saving hardware.

4.2 Butterfly1 (Stage-2)

Used for twiddle factors:

- **$W_0 = 1$**
- **$W_4 = -j$**

Rotation by −j is computed as:

$$(a + jb)(-j) = b - ja$$

This is implemented using **sign swaps and negation**, without using CORDIC or multipliers.

This reduces both area and latency.

4.3 Butterfly2 (Stage-3 and Stage-4)

This block implements the full complex butterfly:

$$X = a + b \cdot W_k$$
$$Y = a - b \cdot W_k$$

Here, the multiplication by W_k is implemented using the **CORDIC engine**.

Before feeding the value to CORDIC, **gain compensation** is applied:

xin2 = (x2>>>1) + (x2>>>4) + (x2>>>5) + (x2>>>6);

This approximates:

$$\frac{1}{1.647}$$

which compensates the inherent CORDIC gain.

5. CORDIC Engine
The CORDIC module performs **vector rotation**:

$$(x', y') = (x, y) \cdot e^{-j\theta}$$

It uses:

- 16 iterations
- Shift-add arithmetic

- No multipliers
- Angle LUT (atan table)
- Quadrant correction

This allows efficient computation of:

$$b \cdot W_k$$

in fixed-point arithmetic.

6. Bit Width Growth

In FFT, each butterfly adds two values. To avoid overflow, the bit-width must increase.

Stage	**Bit Width**
Input	16 bits
Stage-1	17 bits
Stage-2	17 bits
Stage-3	17 bits

This follows:

$$\log_2(16) = 4$$

levels of addition.

7. Gain Compensation
CORDIC introduces a constant gain:

$$K \approx 1.647$$

Without compensation, every FFT stage using CORDIC would amplify the signal.

This is corrected by pre-scaling:

$$\frac{1}{1.647}$$

implemented using shifts instead of division.

8. Accuracy Considerations
Small numerical errors occur due to:

- Fixed-point arithmetic

- CORDIC approximation
- Shift-based gain compensation
- Finite number of iterations

These cause:

- Small imaginary leakage
- Slight amplitude errors

However, **FFT frequency bins and spectral peaks are correct**, which confirms the validity of the implementation.

9. Advantages

i) No hardware multipliers
ii) Fixed-point optimized
iii) CORDIC-based twiddle factors
iv) Hardware-friendly FFT
v) Suitable for FPGA and ASIC

This architecture is similar to **real FFT accelerators used in communication chips**.

10.Testcases

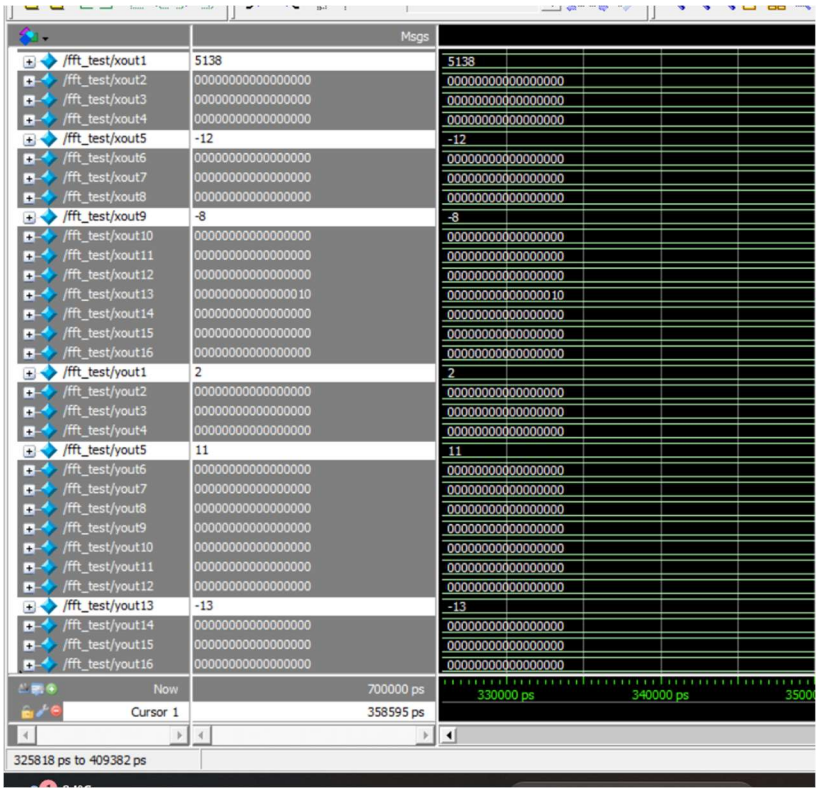
a)

k	**Input x+jy**	**Exact FFT X[k]**	**Cordic FFT Output**
0	320+0j	**5120 + 0j**	**5138 + 0j**
1	320+0j	0+0j	0 + 2j
2	320+0j	0+0j	0+0j
3	320+0j	0+0j	0+0j
4	320+0j	0+0j	-12+0j
5	320+0j	0+0j	0+0j
6	320+0j	0+0j	0+0j
7	320+0j	0+0j	0+0j
8	320+0j	0+0j	0+2j
9	320+0j	0+0j	0+0j
10	320+0j	0+0j	-8+0j
11	320+0j	0+0j	0+0j
12	320+0j	0+0j	0+11j
13	320+0j	0+0j	0+0j
14	320+0j	0+0j	0+0j
15	320+0j	0+0j	0+0j

MAE = 3.31

MRED = 0.352%

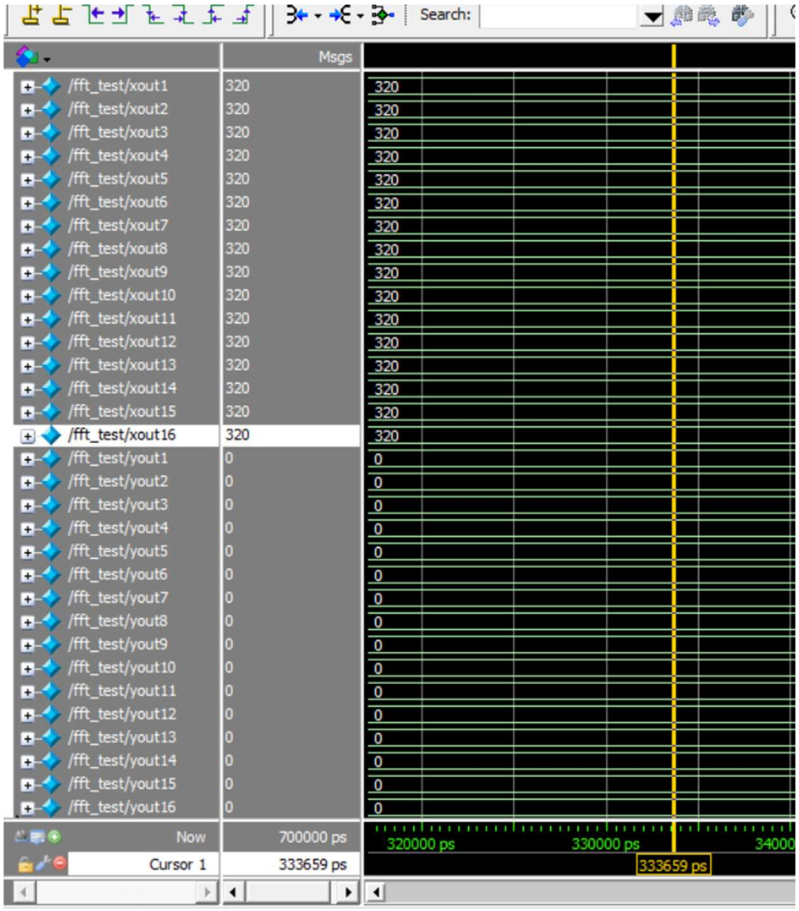
NRMSE = 0.126%



b)

k	Input	Exact FFT	Cordic FFT
0	320+0j	320+0j	320+0j
1	0+0j	320+0j	320+0j
2	0+0j	320+0j	320+0j
3	0+0j	320+0j	320+0j
4	0+0j	320+0j	320+0j
5	0+0j	320+0j	320+0j
6	0+0j	320+0j	320+0j
7	0+0j	320+0j	320+0j
8	0+0j	320+0j	320+0j
9	0+0j	320+0j	320+0j
10	0+0j	320+0j	320+0j
11	0+0j	320+0j	320+0j
12	0+0j	320+0j	320+0j
13	0+0j	320+0j	320+0j
14	0+0j	320+0j	320+0j
15	0+0j	320+0j	320+0j

MAE, MRED, NRMSE = 0



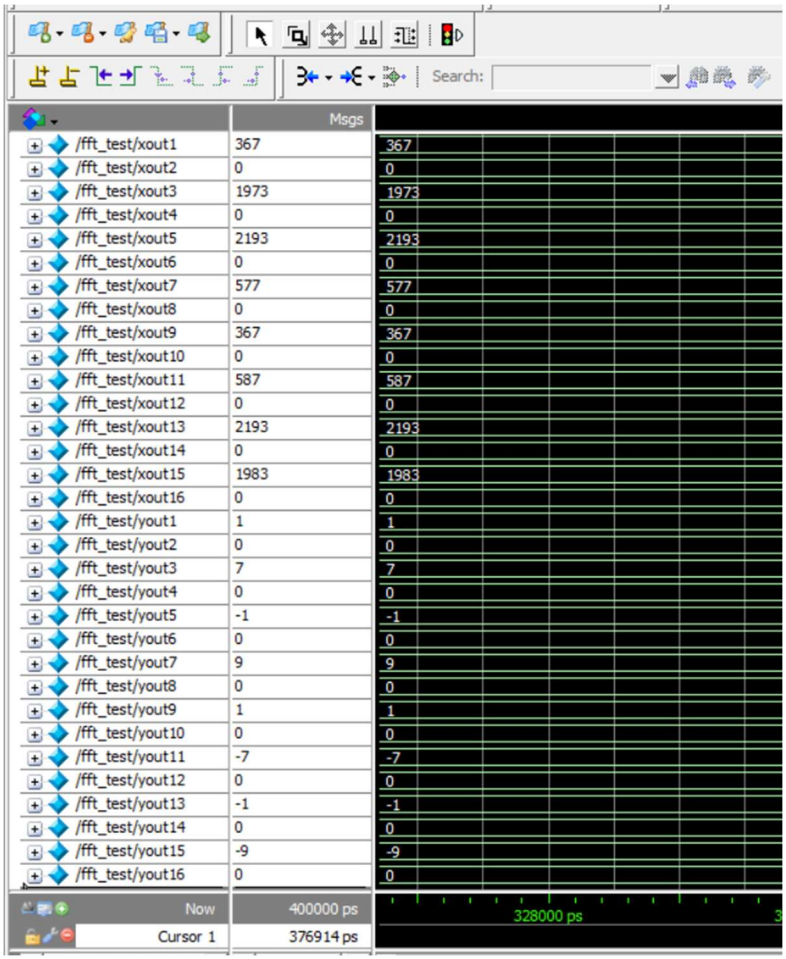
c)

Index (n/k)	Input x[n]	Exact Output (Golden)	Given Output (Approx)
0	640.00 + 0.00j	376.00 + 0.00j	367.00 + 1.00j
1	122.00 + 0.00j	0.00 + 0.00j	0.00 + 0.00j
2	-226.00 + 0.00j	1970.14 + 0.00j	1973.00 + 7.00j
3	-122.00 + 0.00j	0.00 + 0.00j	0.00 + 0.00j
4	0.00 + 0.00j	2184.00 + 0.00j	2193.00 - 1.00j
5	-122.00 + 0.00j	0.00 + 0.00j	0.00 + 0.00j
6	-226.00 + 0.00j	589.86 + 0.00j	577.00 + 9.00j
7	122.00 + 0.00j	0.00 + 0.00j	0.00 + 0.00j
8	640.00 + 0.00j	376.00 + 0.00j	367.00 + 1.00j
9	122.00 + 0.00j	0.00 + 0.00j	0.00 + 0.00j
10	-226.00 + 0.00j	589.86 + 0.00j	587.00 - 7.00j
11	-122.00 + 0.00j	0.00 + 0.00j	0.00 + 0.00j
12	0.00 + 0.00j	2184.00 + 0.00j	2193.00 - 1.00j
13	-122.00 + 0.00j	0.00 + 0.00j	0.00 + 0.00j
14	-226.00 + 0.00j	1970.14 + 0.00j	1983.00 - 9.00j
15	122.00 + 0.00j	0.00 + 0.00j	0.00 + 0.00j

MAE = 5.1717

MRED = 0.8081%

NRMSE = 0.31%



d)

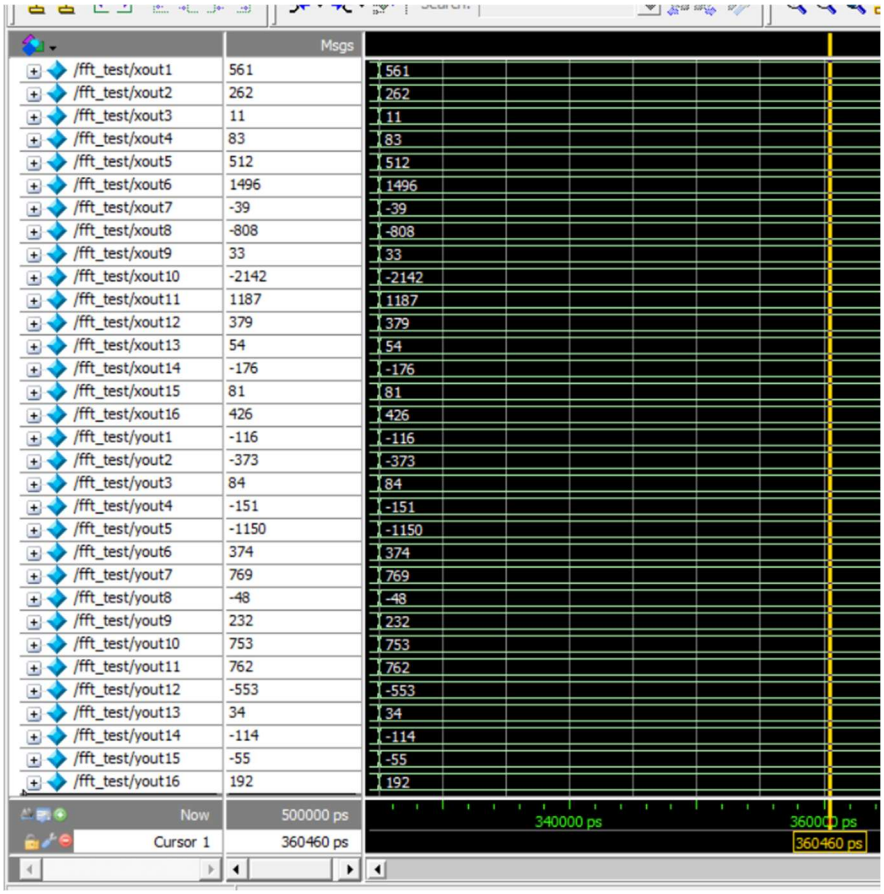
Index (k)	Input x[n]	Exact Output (Golden)	Given Output (Approx)
0	120 + 40j	570.00 - 110.00j	561.00 - 116.00j
1	200 - 60j	268.61 - 371.46j	262.00 - 373.00j
2	-150 + 90j	20.17 + 80.59j	11.00 + 84.00j
3	300 - 20j	85.47 - 159.32j	83.00 - 151.00j
4	-80 - 200j	510.00 - 1150.00j	512.00 - 1150.00j
5	60 + 180j	1486.01 + 374.39j	1496.00 + 374.00j
6	220 - 140j	-36.57 + 770.12j	-39.00 + 769.00j
7	-260 + 100j	-804.90 - 53.66j	-808.00 - 48.00j
8	180 + 30j	30.00 + 230.00j	33.00 + 232.00j
9	-90 - 170j	-2132.53 + 748.02j	-2142.00 + 753.00j
10	140 + 210j	1179.83 + 759.41j	1187.00 + 762.00j
11	-300 - 60j	378.80 - 544.95j	379.00 - 553.00j

12	70 - 120j	50.00 + 30.00j	54.00 + 34.00j
13	260 + 80j	-182.09 - 110.96j	-176.00 - 114.00j
14	-200 + 150j	76.57 - 50.12j	81.00 - 55.00j
15	100 - 220j	420.64 + 197.92j	426.00 + 192.00j

MAE = 7.1385

MRED = 2.97%

NRMSE = 0.34%



11. Conclusion

This project successfully implements a **16-point radix-2 FFT** using a **CORDIC-based hardware-optimized architecture**. By eliminating multipliers and using fixed-point arithmetic with gain compensation, the design achieves an efficient and scalable FFT suitable for digital signal processing hardware.