Implementation of 32-point FFT using Cooley-Tuckey Algorithm

Table of Contents

List of Figures

Abstract

The Fast Fourier Transform (FFT) plays a critical role in modern digital signal processing systems, especially in wireless communication protocols such as OFDM. This project focuses on the design and implementation of a 32-point FFT using the Radix-2 Decimation-in-Time (DIT) Cooley-Tukey algorithm using Verilog and synthesizing it using Vivado. The design aims to meet real-time processing constraints, including pipelining and serialization.

A set of MATLAB scripts were developed to support and verify the hardware design decisions. The results of these analyses guided the selection of numeric formats and the architecture for computational blocks. The system is designed to operate with 8-bit real input samples and aims to output 100 million FFT results per second (**Throughput = 100 Mbps**) using pipelined and parallelized data paths.

System Design

FFT and the Cooley-Tukey Algorithm

The 32-point FFT is implemented using the Radix-2 Decimation-in-Time (DIT) version of the Cooley-Tukey algorithm. The algorithm recursively divides a large DFT into smaller ones by splitting the input into even and odd indexed elements. A 32-point FFT requires $log_2(32) = 5$ stages, with each stage comprising 16 butterfly operations.

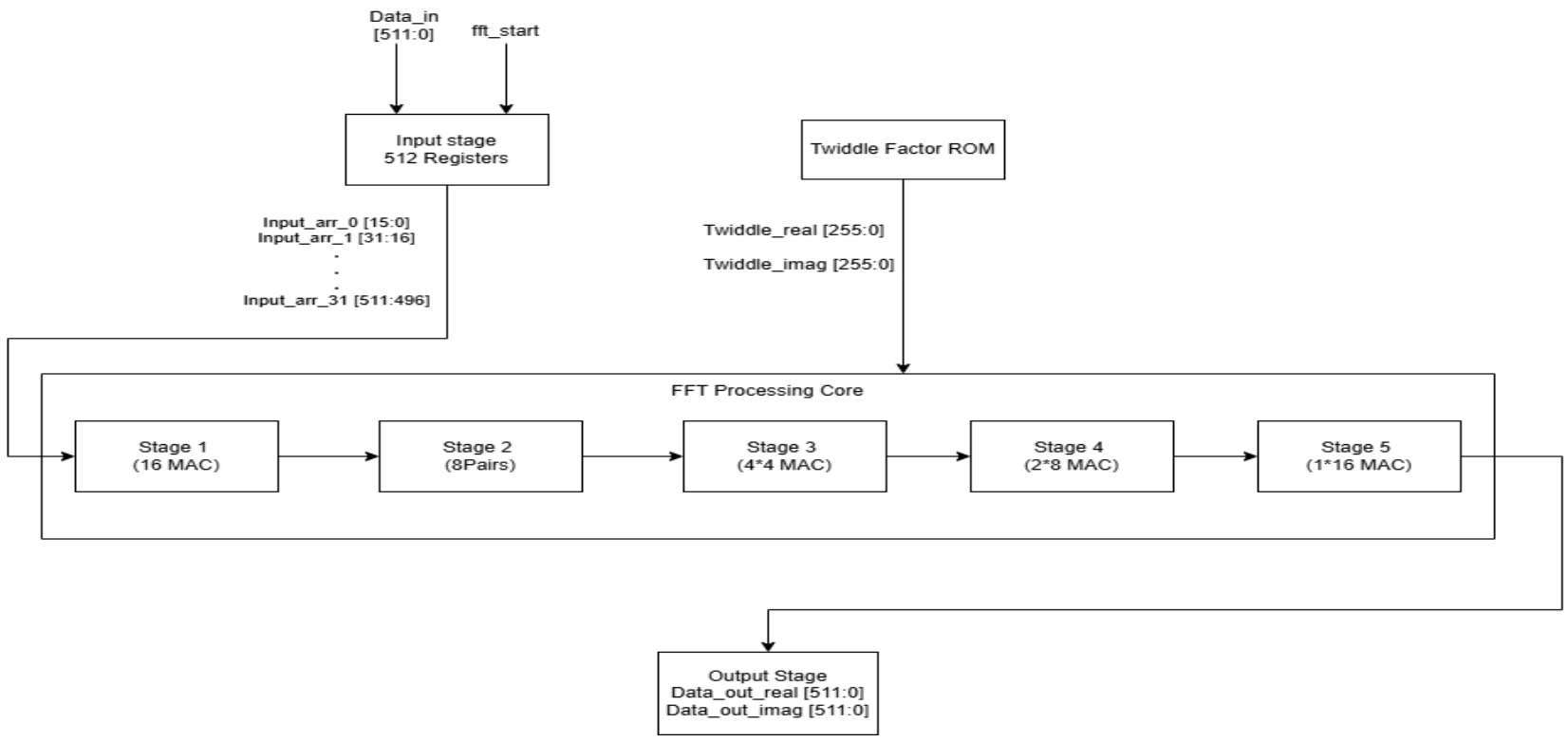

Figure (1): Top level Architecture

Butterfly Processing Element (MAC unit)

The butterfly unit is the fundamental building block of the FFT pipeline. It performs both addition and subtraction between two complex numbers and applies a twiddle factor multiplication to one of them. Each butterfly operates as follows:

Inputs: Two 16-bit fixed-point numbers with format **Q8.8** (8 bits assigned for integer part and 8 bits assigned for fractional part).

Operation:

$$X[k] = A + B * W_N^k \quad , \quad X\left[k + \frac{N}{2}\right] = A - B * W_N^k$$

Complex multiplication: $(a + jb) \times (c + jd) = (ac - bd) + j(ad + bc)$

Where: 1) A, B → Input Data, 2) W_N^k → twiddle factor $e^{-\frac{j2\pi k}{N}}$, 3) N →FFT size (32 in this case), 4) k → index of the butterfly in the current stage.

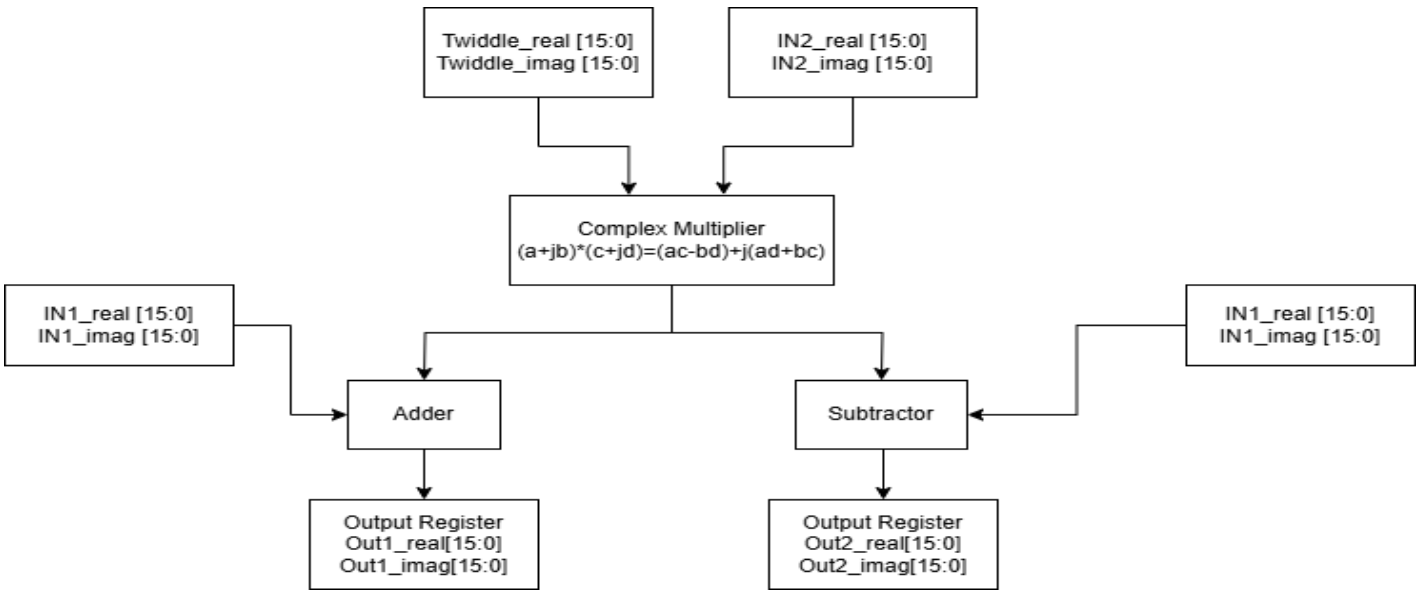

Figure (2): MAC unit

Twiddle Factors

Twiddle factors are precomputed in MATLAB and stored in a read-only memory (ROM). Each twiddle factor is represented as a pair of signed fixed-point numbers with format **Q1.15** determined from the fixed-point error analysis performed in MATLAB.

Number of Stages and Layers

The design includes 5 pipeline stages, with each stage responsible for a full level of butterfly computations. Register stages are placed between these layers to meet the pipelining requirement and ensure throughput at 100 Mbps.
Each stage includes **N/2 = 16 butterflies (MAC unit)**, **Complex multipliers**, **Adders/Subtractors**.

MATLAB Fixed-Point Analysis

To ensure numerical stability and minimal quantization error in the hardware implementation, multiple MATLAB scripts were developed:

Twiddle Factor Generation (Twiddle factor calculation file)

A script computes all required twiddle factors for a 32-point FFT: $\boldsymbol{W_N^k = e^{-j2\pi k/N}}$, $\boldsymbol{k = 0 \ldots N/2 - 1}$

These values are exported in fixed-point formats to be included in the RTL design.

Fixed-Point Representation Comparison (Fixed point format file)

Three fixed-point formats were evaluated:

- **Q1.15:** 1 integer bit, 15 fraction bits
- **Q1.7:** 1 integer bit, 7 fraction bits
- **Q8.8:** 8 integer bits, 8 fraction bits

Result Summary:

Format	Mean Relative Error	Comment
Q1.15	Real = -0.1736, Imaginary = 0.0146	Best for precision
Q1.7	Real = -0.2318, Imaginary = 0.0531	Moderate precision
Q8.8	Real = -0.9743, Imaginary = 0.1608	Poor precision but has a bigger range

Therefore, we choose Q1.15 format for twiddle factors due to its high accuracy and low hardware overhead when using proper rounding and saturation.

Algorithm Verification Script (Butterfly FFT Model file)

This script verifies the correctness of the custom Cooley-Tukey Algorithm against MATLAB built-in fast Fourier transform function using:

- Real inputs (matching hardware specs)
- Stage-by-stage comparisons
- Error calculation for each output point

We exclude from this script text files for the input vectors in time domain and the expected output vectors (real, imaginary) that we take it as a golden reference to our RTL results.

Pre-Synthesis Simulation

- Validating the functionality of the FFT design at the RTL level before synthesis. A testbench-driven simulation was used to verify numerical correctness, input handling, and timing behavior.

Test Vector Generation

- MATLAB scripts were used to generate and validate the input/output behavior of the FFT system under different conditions as we extract from the MATLAB model text files for the input stimulus and expected output to compare it with the Design output.

Testbench Operation

The testbench performs the following:

1. Reads time-domain input vectors from Input_vector.txt.
2. Feeds them into the FFT module on the negative edge of the clock.
3. Triggers FFT execution by asserting the fft_start signal.
4. Waits for the pipelining delay before capturing the output.
5. Unpacks $data\_out\_real$ and $data\_out\_imag$ to compare with reference results.

Waveform and Result Verification

Figure (3): Pre-Synthesis Waveform

Note: The output is generated after **6 clock-cycles** as expected as we register the inputs (1 clock cycle) then take $\log_2(32) = 5$ clock cycles for the 5 stages.

FPGA Flow using Vivado

Project Summary

Project Summary

Settings Edit

Project name:	project_1
Project location:	D:/OneDrive - Cairo University - Students/Desktop/FFT Project/RTL_Implementation/project_1
Product family:	Virtex UltraScale
Project part:	xcvu440-flga2892-3-e
Top module name:	FFT_32
Target language:	Verilog
Simulator language:	Mixed

Synthesis		Implementation	
Status:	Not started	Status:	Not started
Messages:	No errors or warnings	Messages:	No errors or warnings
Part:	xcvu440-flga2892-3-e	Part:	xcvu440-flga2892-3-e

Figure (4): Project Summary from Vivado

Utilization Area

SYNTHESIZED DESIGN * - xcvu440-flga2892-3-e (active)

Tcl Console | Messages | Log | Reports | Design Runs | **Utilization** | Timing | Debug

Hierarchy

- Hierarchy
- Summary
- CLB Logic
  - CARRY8 (1%)
  - CLB LUTs (1%)
    - LUT as Logic (1%)
  - CLB Registers (<1%)
    - Register as Flip Flop (<1%)
- BLOCKRAM
- ARITHMETIC
  - DSPs (11%)
    - DSP48E2 only
- I/O
  - Bonded IOB (>100%)
- CLOCK
  - GLOBAL CLOCK BUFFERs (<1%)
    - BUFGCE (<1%)
- ADVANCED
- CONFIGURATION
- Primitives
- Black Boxes
- Instantiated Netlists

Name	CLB LUTs (253296)	CLB Registers (5065920)	CARRY8 (31660)	DSPs (28...)	Bonded IOB (1456)	GLOBAL CLOCK BUFFERs (1440)
FFT_32	10963	3328	1872	320	1539	1
Stage_1[0].MAC_inst_...	314	64	43	4	0	0
Stage_1[2].MAC_inst_...	54	0	7	4	0	0
Stage_1[4].MAC_inst_...	314	64	43	4	0	0
Stage_1[6].MAC_inst_...	54	0	7	4	0	0
Stage_1[8].MAC_inst_...	314	64	43	4	0	0
Stage_1[10].MAC_inst...	54	0	7	4	0	0
Stage_1[12].MAC_inst...	314	64	43	4	0	0
Stage_1[14].MAC_inst...	54	0	7	4	0	0
Stage_1[16].MAC_inst...	314	64	43	4	0	0
Stage_1[18].MAC_inst...	54	0	7	4	0	0
Stage_1[20].MAC_inst...	314	64	43	4	0	0
Stage_1[22].MAC_inst...	54	0	7	4	0	0
Stage_1[24].MAC_inst...	264	64	29	4	0	0
Stage_1[26].MAC_inst...	124	0	21	4	0	0
Stage_1[28].MAC_inst...	264	64	29	4	0	0
Stage_1[30].MAC_inst...	124	0	21	4	0	0
Stage_2[0].MAC_inst_...	267	64	42	4	0	0
Stage_2[0].MAC_inst_...	250	64	34	4	0	0
Stage_2[2].MAC_inst_...	63	0	10	4	0	0
Stage_2[2].MAC_inst_...	72	0	10	4	0	0
Stage_2[4].MAC_inst_...	267	64	42	4	0	0
Stage_2[4].MAC_inst_...	276	64	42	4	0	0

utilization_1

Figure (5): Utilization report from Vivado

Note that: we target the largest FPGA with respect to input/output bins (IOB) and still we exceed the limit as we have **1539 in/out bins,** and our target FPGA has 1456 bins **only.**

Timing Summary

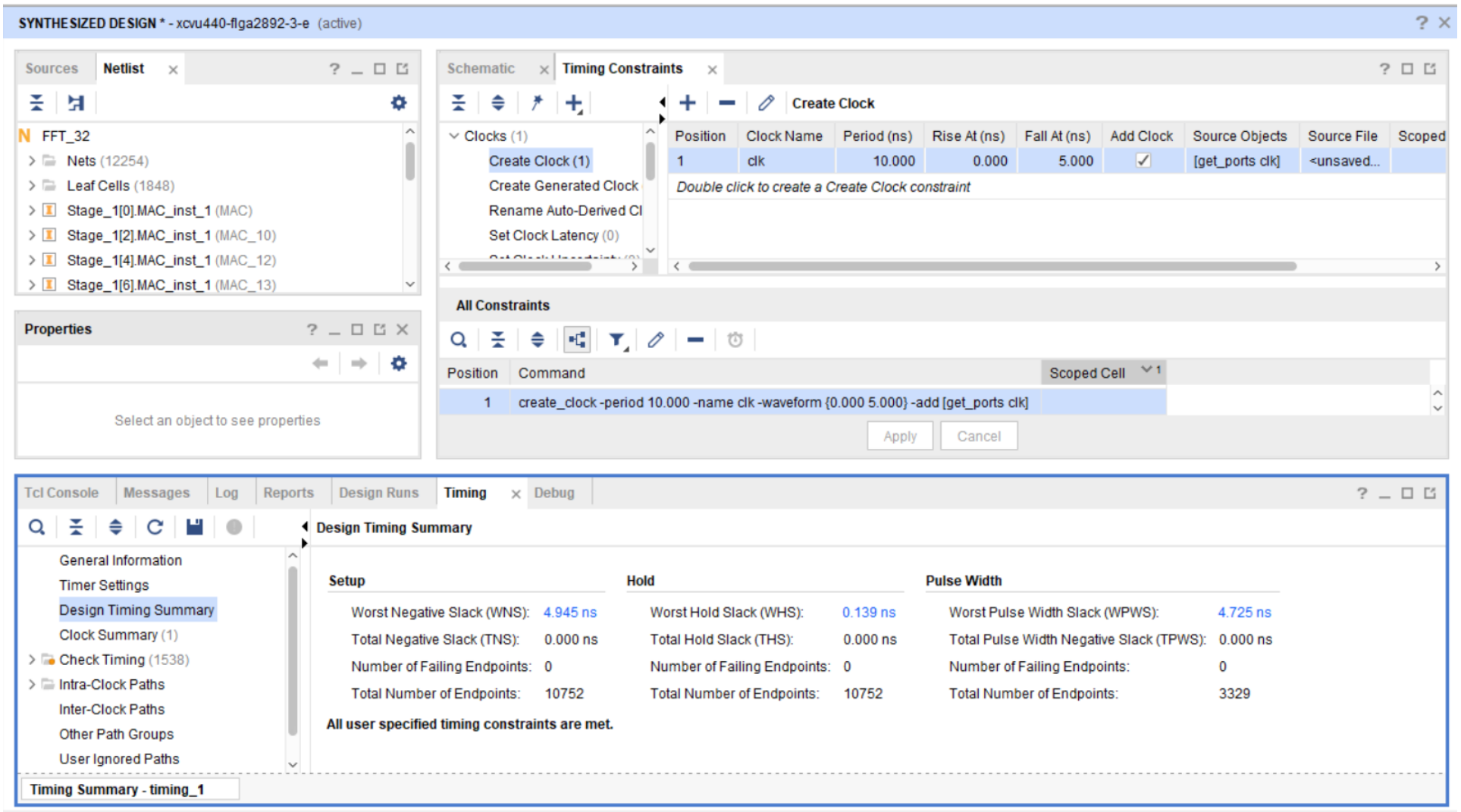

Figure (6): Timing report from Vivado

- From the above timing report, we observe that Setup Slack: **4.945 ns** and Hold Slack: **0.139 ns**. Therefore, out FFT design can operate at approximately double the frequency $f \approx 200\ MHz$ as we have approximated **5 ns** setup slack.

Post Synthesis Simulation

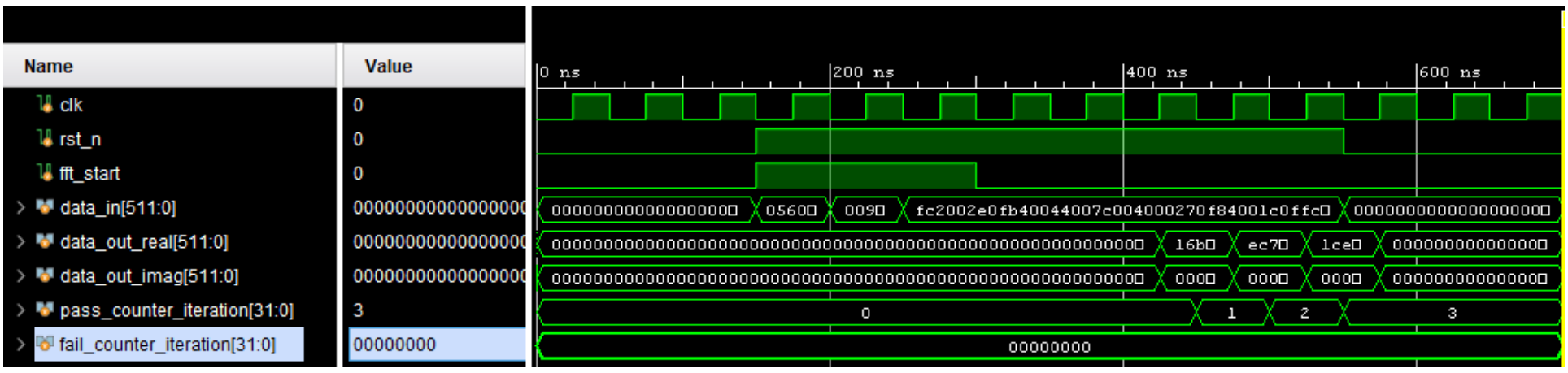

Figure (7): Post-Synthesis simulation Waveform

- Figure (7) captures the functional simulation results after synthesis. Where it is clear from the waveform that the outputs (real, imaginary) are produced after **6 clock cycle latency**.

  [1 clock cycle to register the input vector + 5 clock cycle for the 5 stages of 32-point FFT]