ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

# Fast Fourier Transform

ADI Summer 25' Digital IC Design Internship Final Project

# Project Phases

- About FFT Algorithm
- System Modeling
- RTL Design
- RTL Verification
- ASIC Implementation
- FPGA Implementation

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

# About FFT Algorithm

## Phase 0

# Radix-2 Fast Fourier Transform Algorithm

- The straightforward implementation of Fourier series equation requires $\boldsymbol{N^2}$ complex multiplication operations.
- James Cooley and John Tukey found a way to reduce the required number of complex multiplications from $\boldsymbol{N^2}$ to be proportional to $\mathbf{N log_2}(\boldsymbol{N})$.
- Uses **divide-and-conquer** algorithm.
- There are two types of radix-2 FFT algorithms: Decimation-In-Time **(DIT)** and Decimation-In-Frequency **(DIF)**.

# Derivation of the mathematical formula

- We start by Fourier Series equation, as follows:

$$\hat{X} = \sum_{n=0}^{N-1} X[n] e^{\left(\frac{-j2\pi kn}{N}\right)}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} X[n] e^{\left(\frac{-j2\pi kn}{N}\right)} + \sum_{n=\frac{N}{2}}^{N-1} X[n] e^{\left(\frac{-j2\pi kn}{N}\right)}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} X[n] e^{\left(\frac{-j2\pi kn}{N}\right)} + \sum_{\hat{n}=0}^{N-1} X\left[\hat{n} + \frac{N}{2}\right] e^{\left(\frac{-j2\pi k\hat{n}}{N}\right)} e^{(j\pi k)}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} (X[n] + (-1)^k \left[\hat{n} + \frac{N}{2}\right]) e^{\left(\frac{-j2\pi kn}{N}\right)}$$

# Derivation of the mathematical formula

- We can divide frequency index k into even and odd groups, as follows:
- $\mathrm{X}[2\hat{k}] = \sum_{n=0}^{\frac{N}{2}-1}(X[n] + (-1)^{k}\left[\hat{n} + \frac{N}{2}\right])\, e^{(\frac{-j2\pi kn}{N})}$
- Even group $(k = 2\hat{k})$:
  - $\mathrm{X}[2\hat{k}] = \sum_{n=0}^{\frac{N}{2}-1}\left(X[n] + (-1)^{\hat{k}}\left[n + \frac{N}{2}\right]\right) e^{\left(\frac{-j2\pi\hat{k}n}{N}\right)}$
  - $\mathrm{X}[2\hat{k}] = \sum_{n=0}^{\frac{N}{2}-1}\underbrace{\left(X[n] + \left[n + \frac{N}{2}\right]\right)}_{\hat{\mathrm{X}}_1[n]} e^{\left(\frac{-j2\pi\hat{k}n}{\frac{N}{2}}\right)} = DFS_{\frac{N}{2}}(\widehat{\mathrm{X}_1}[n])$
- Odd group $(k = 2\hat{k} + 1)$:
  - $\mathrm{X}[2\hat{k}] = \sum_{n=0}^{\frac{N}{2}-1}\left(X[n] + (-1)^{\hat{k}}\left[n + \frac{N}{2}\right]\right) e^{\left(\frac{-j2\pi\hat{k}n}{N}\right)}$
  - $\mathrm{X}[2\hat{k}] = \sum_{n=0}^{\frac{N}{2}-1}\underbrace{\left(X[n] - \left[n + \frac{N}{2}\right]\right)}_{\hat{\mathrm{X}}_2[n]} e^{\left(\frac{-j2\pi\hat{k}n}{N}\right)} e^{\left(\frac{-j2\pi\hat{k}n}{\frac{N}{2}}\right)} = DFS_{\frac{N}{2}}(\hat{\mathrm{X}}_2[n])$

# Radix-2 Fast Fourier Transform Algorithm

- So, FFT is a fast efficient algorithm to calculate the N-point discrete Fourier transform (DFT), with N being a power-of-2 size.

$$\boldsymbol{X[k] = \sum_{n=0}^{N-1} x[n]\, W_N^{nk}, k = 0, 1, \ldots, N-1}$$

- $X[k]$ is the output at frequency k
- $W_N^{nk} = e^{-j\frac{2\pi}{N}nk}$ called the twiddle factor.

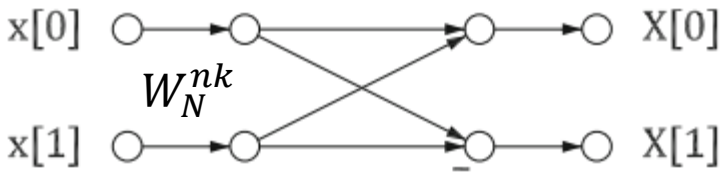
In implementation we used 8 points Radix-2 FFT, So the twiddle factor:

$$\boldsymbol{W_8^k = e^{-j\frac{2\pi}{8}nk}}$$

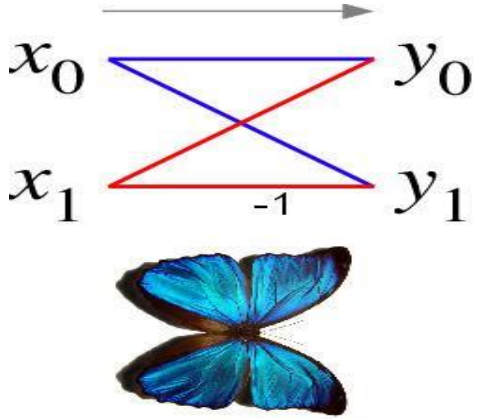
$$W_8^0 = 1,\ W_8^1 = \frac{1-j}{\sqrt{2}},\ W_8^2 = -j,\ W_8^3 = \frac{-(1+j)}{\sqrt{2}}$$

# Decomposition & Butterfly

- There are two types of decomposition, **DIT** (What I used) and **DIF**
- The building block in FFT Algorithm is called butterfly




DIT radix-2 FFT Butterfly



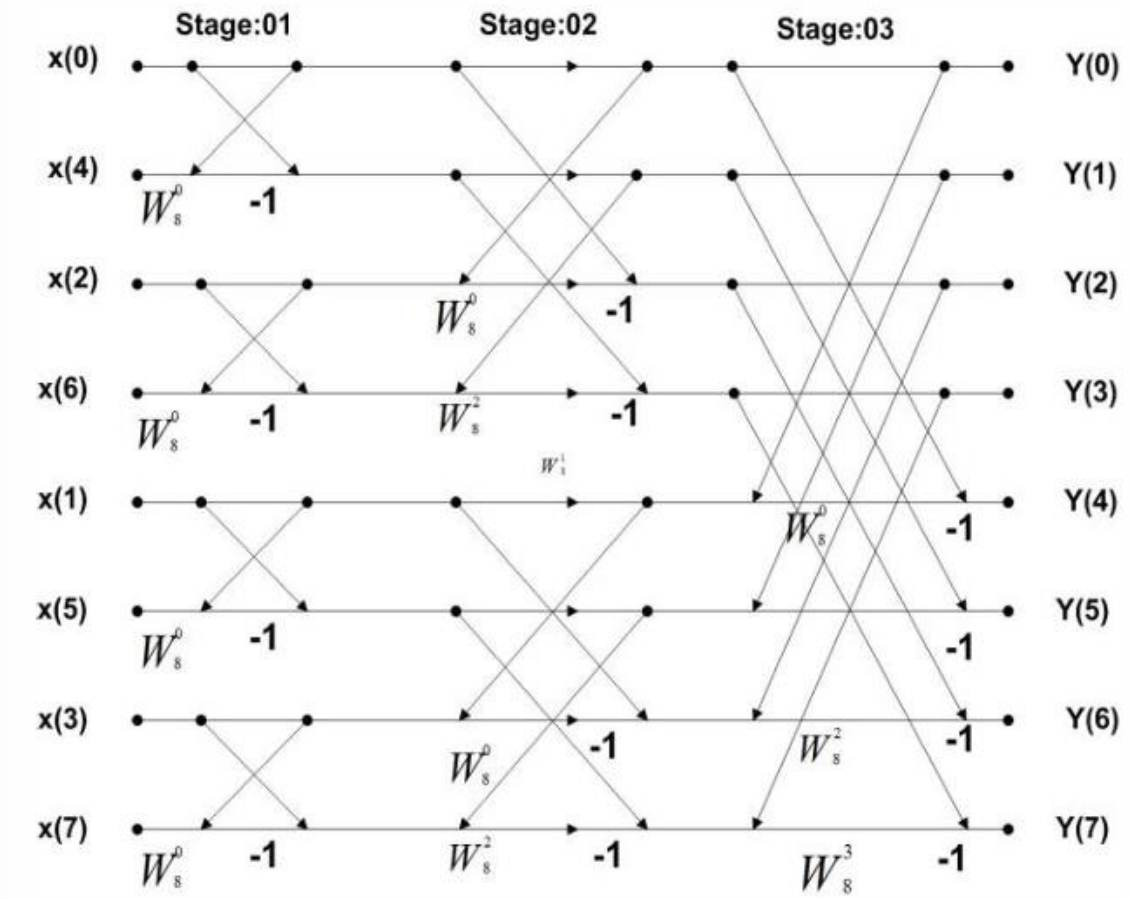

Butterfly Operation

# Radix-2 FFT and 8 Points Implementation

- **Number of stages** = $\log_2(8) = 3$
- **Complexity** = $O(8log(8))$
- **DIT** Decomposition

| Input | Address (binary) | Output (bit reversed) | Address (binary) |
|---|---|---|---|
| x(0) | 000 | x(0) | 000 |
| x(1) | 001 | x(4) | 100 |
| x(2) | 010 | x(2) | 010 |
| x(3) | 011 | x(6) | 110 |
| x(4) | 100 | x(1) | 001 |
| x(5) | 101 | x(5) | 101 |
| x(6) | 110 | x(3) | 011 |
| x(7) | 111 | x(7) | 111 |

Bit Reversal




Butterfly Representation

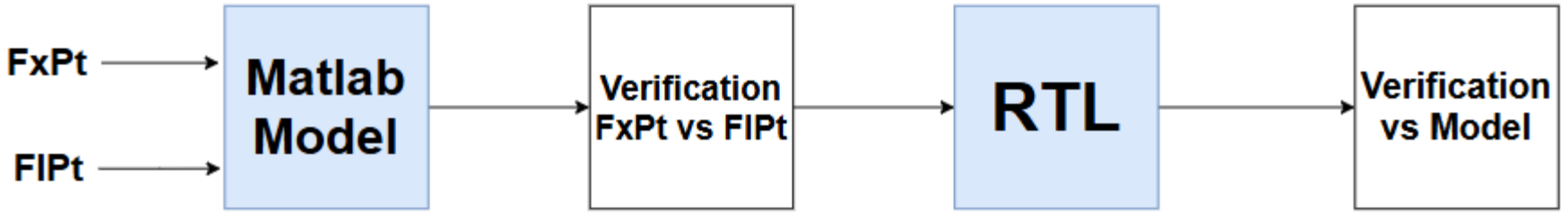
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

# System Modeling

## Phase 1

# Why System Modeling First

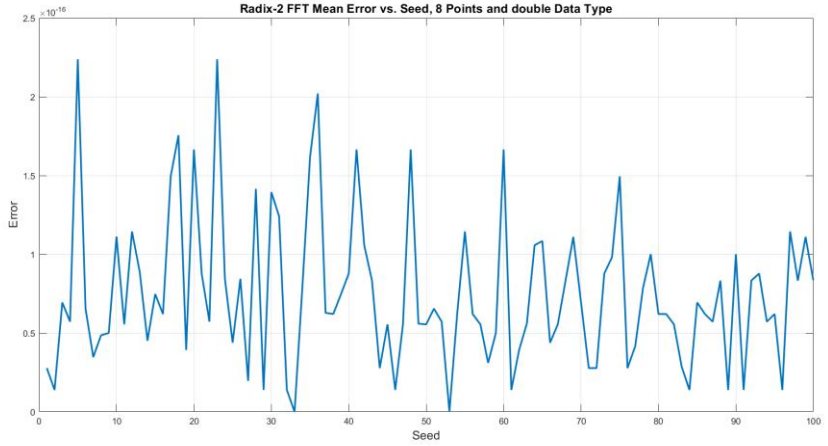
- **Algorithm Verification**
- **Bit-True Validation:**
  - Create a "golden reference model" that produces the *exact* expected outputs for a given set of inputs.
- **Performance Prediction (SQNR)**



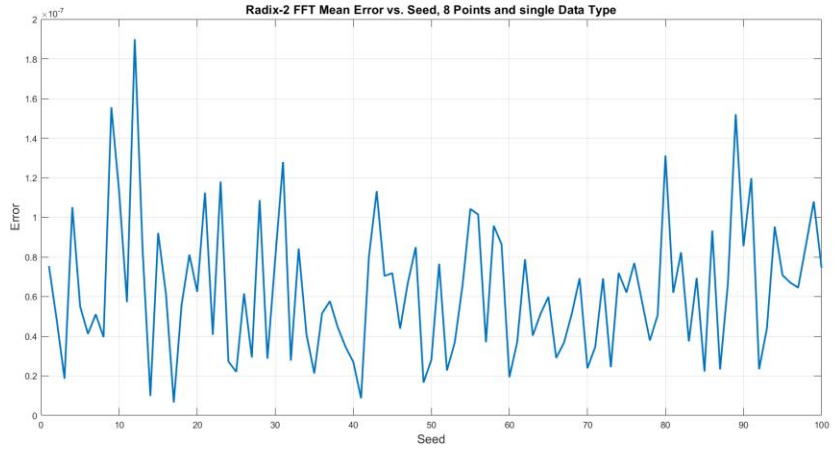

Modeling and Design Process

# Floating-Point Reference Model (Double/Single)

- **Purpose**:
  - Establishing the "**Golden Reference**" To verify the algorithm across different precisions
- **Implementation**:
  - Implemented **FFT.m** algorithm using Matlab and verified against **built-in fft** function
- **Error comparing against built-in fft :**



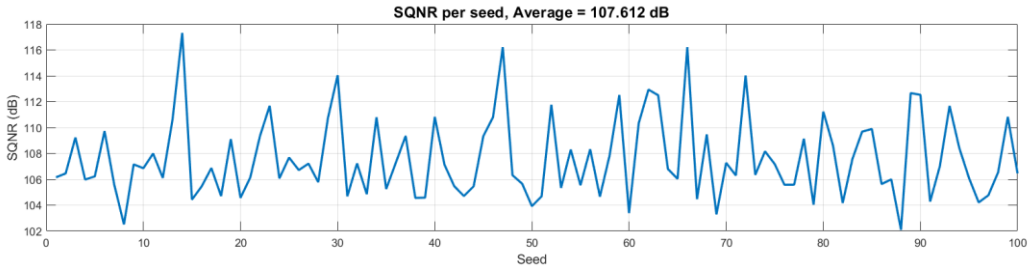

Error for Double data type



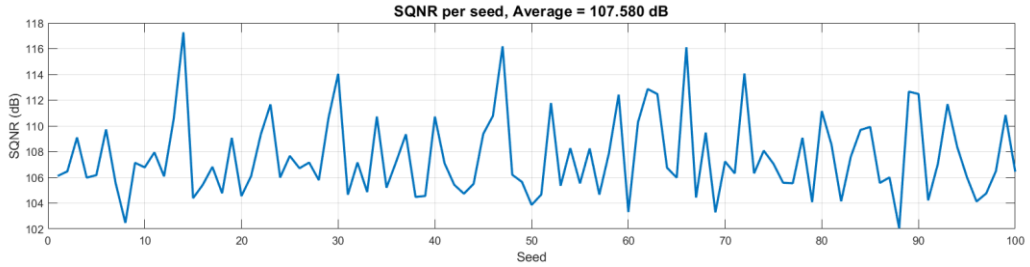

Error for Single data type

# Floating-Point Reference Model (Double/Single)

- SQNR:




SQNR for Double data type




SQNR for Single data type

| Metric | Double Precision | Single Precision | Winner & Analysis |
|---|---|---|---|
| **Average SQNR** | **~107.612 dB** | **~107.580 dB** | **Essentially Identical.** |
| **Numerical Resolution** | ~$10^{-16}$ | ~$10^{-7}$ | **Double is more precise.** |
| **Memory Footprint** | 16 Bytes per sample | 8 Bytes per sample | **Single wins.** Uses half the memory |
| **Computational Speed** | Slower | Faster | **Single Is faster.** |
| **Summary** | The "Golden Reference" | The "Efficient" | **Single precision provides an excellent balance of performance and accuracy** |

Double vs Single Comparison

# Fixed-Point Model

## Instrumentation Results:

- We executed 100 exhaustive random test cases to get precisions of model's signals

| Name | Type | Size | Class | Proposed Signedness | Proposed WL | Proposed FL | Always Whole Number | Sim Min | Sim Max |
|---|---|---|---|---|---|---|---|---|---|
| Y | Output | 1 × 8 | complex double | Signed | 32 | 27 | No | -9.579709715114703 | 8.589724741655168 |
| ▸ T | Input | 1 × 1 | struct | - | - | - | No | - | - |
| X | Input | 1 × 8 | complex double | Signed | 32 | 29 | No | -3.075903420547428 3 | 3.1132750454258082 |
| i | Local | 1 × 1 | double | Signed | 32 | 0 | **Yes** | 1 | 8 |
| N | Local | 1 × 1 | double | Signed | 32 | 0 | **Yes** | 8 | 8 |
| s1 | Local | 1 × 8 | complex double | Signed | 32 | 28 | No | -4.133774590868551 | 4.798235113407159 |
| s2 | Local | 1 × 8 | complex double | Signed | 32 | 28 | No | -5.677757638708031 | 6.2331334899239526 |
| W | Local | 4 × 1 | complex double | Signed | 32 | 30 | No | -1 | 1 |

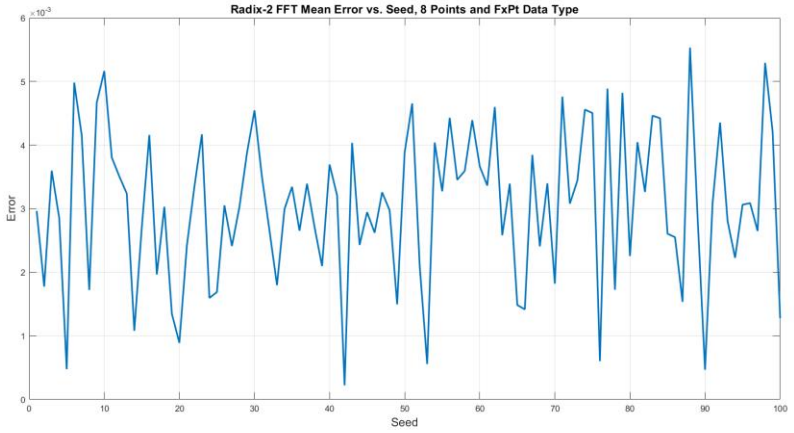
Instrumentation Results Table

```
case 'FxPt' % Word Length = 12 bits
    T.X   = fi(complex(0,0), 1, 3 + 9, 9);     % Input: 3 int, 9 frac
    T.W   = fi(complex(0,0), 1, 2 + 10, 10);   % Twiddle: 2 int, 10 frac
    T.a   = fi(complex(0,0), 1, 4 + 8, 8);     % Butterfly input: 4 int, 8 frac
    T.b   = fi(complex(0,0), 1, 4 + 8, 8);     % Butterfly input: 4 int, 8 frac
    T.s1  = fi(complex(0,0), 1, 4 + 8, 8);     % First Stage output: 4 int, 8 frac
    T.s2  = fi(complex(0,0), 1, 4 + 8, 8);     % Second Stage output: 4 int, 8 frac
    T.Y   = fi(complex(0,0), 1, 5 + 7, 7);     % Output: 5 int, 7 frac
```

Signals Precisions

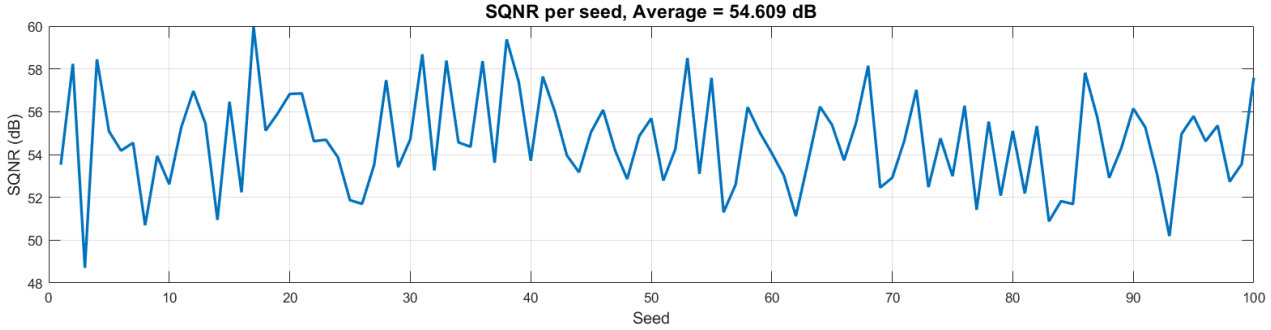
# Fixed-Point Model

- Error comparing against **built-in fft** :




Error for FxPt data type

- SQNR:




SQNR for FxPt data type

- **Interpretation:**
  - This SQNR value is excellent for many applications .
  - Error per seed is around $10^{-3}$ which is small and won't badly affect the accuracy.

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

# RTL Design

## Phase 2

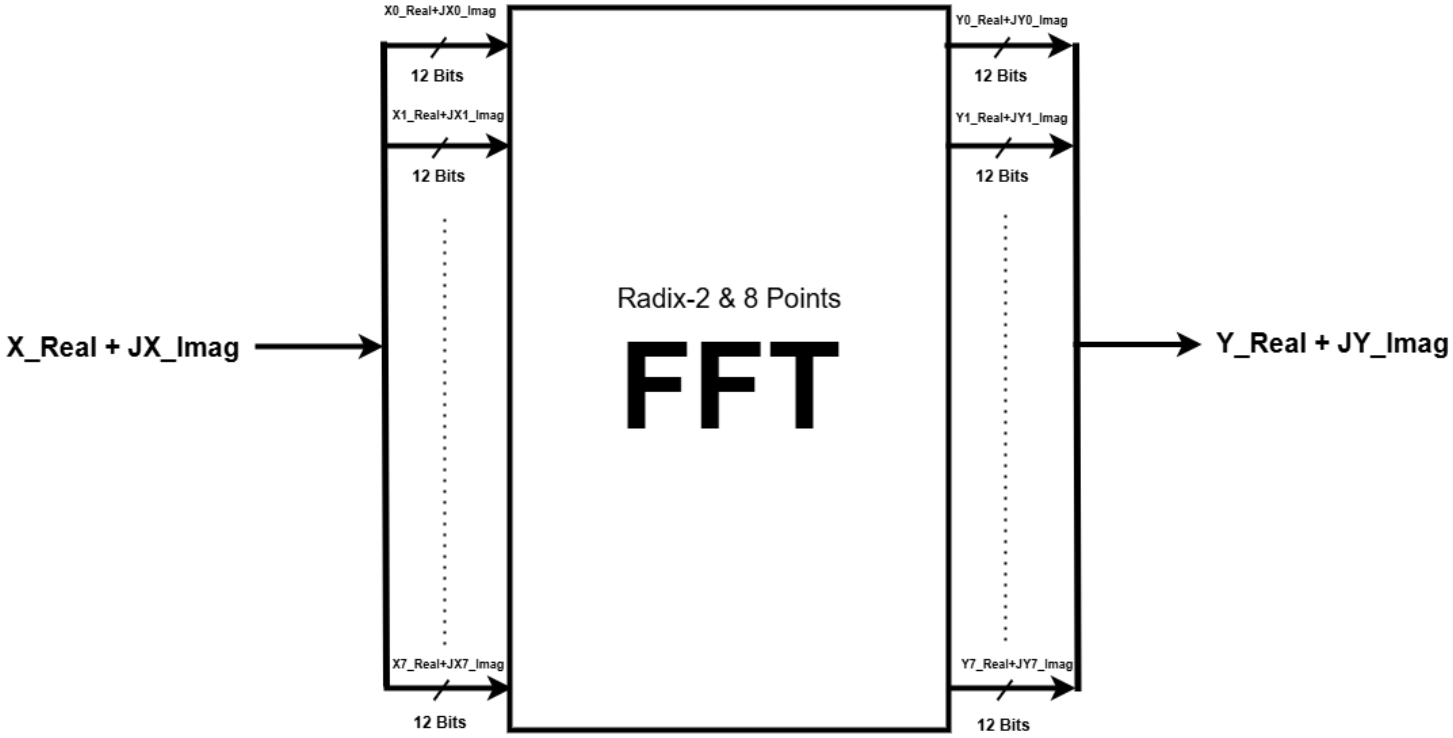
# High-Level Architecture & Features

## Algorithm:

- Radix-2 Decimation in Time (DIT)

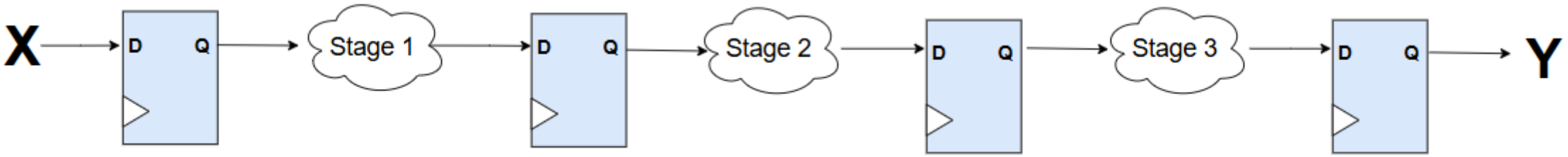
## Key Features:

- **Fully Pipelined:** 4-stage pipeline for high throughput.
- **Fixed-Point Arithmetic:** Optimized for area and power.
- **Format:** Input (Q3.9), Internal (Q4.8), Output (Q5.7).




Block Interface

# Pipelined Structure




High Level Block Diagram

| Stage | Description | Latency (Cycles) |
|---|---|---|
| **Input Reg** | Samples input X before processing | 1 |
| **Stage 1** | 4x Radix-2 Butterflies | 1 |
| **Stage 2** | 4x Radix-2 Butterflies + Twiddles ($W^2$) | 1 |
| **Stage 3 and Output Registering** | 4x Radix-2 Butterflies + Full Twiddles and stabilize outputs | 1 |
| **Total** | **Throughput: 1 result/cycle** | **4 cycles** |

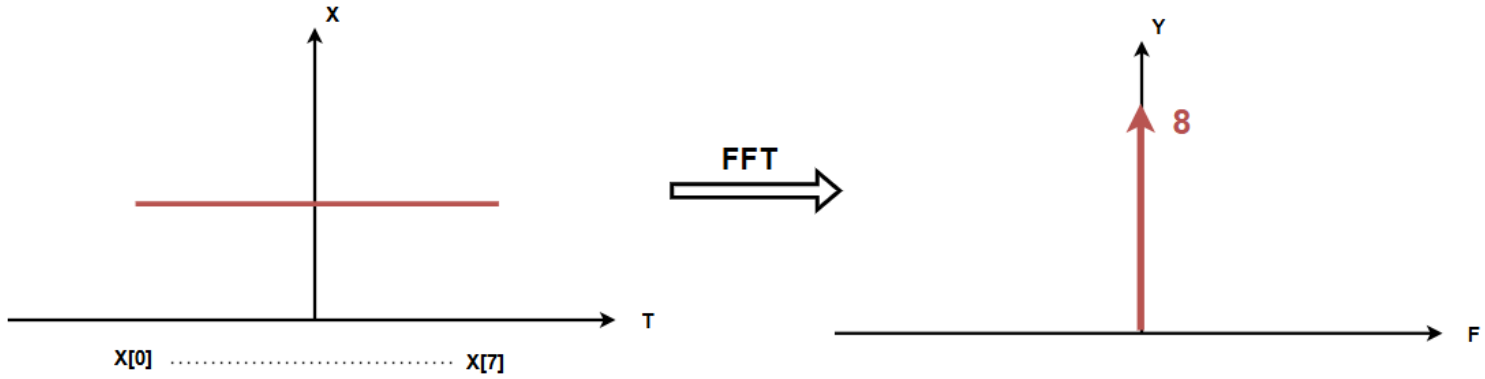
Latency and Throughput details

# Core Computational Engine: The Butterfly MAC

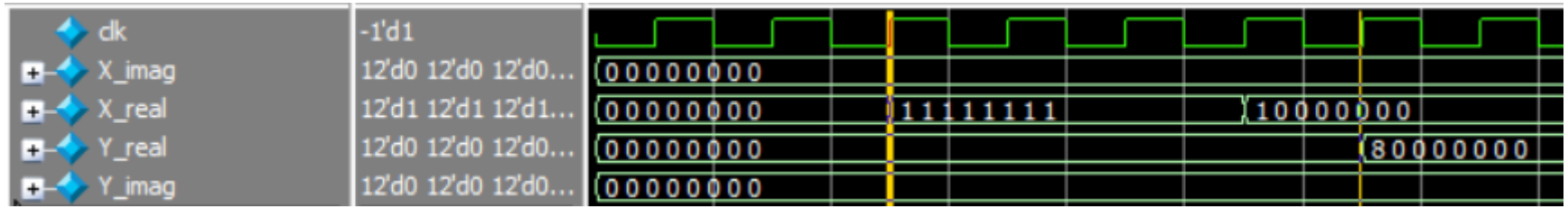
- A **single**, **parameterized** function handles all stages and optimizations
- **Unified Processing**:
- One function performs all butterfly operations (**A + W*B, A - W*B**) for all three stages.
- **Intelligent Twiddle Handling:**
  - **Stage 1:**
    - No multiplication. Uses W0 = 1, simply passing data through.
  - **Stage 2:**
    - Optimized rotation. Uses W2 = -j, implemented via hardwired logic (swap and negate)
  - **Stage 3:**
    - Full multiplication. Only stages needing W1 and W3 use actual multipliers.

# Verifying Design (Simple Test)

- I Have applied a **direct input** to see if the design is functioning well or not




FFT DC Test




Test Output Waveform

- As we see here:
  - Output is generated after 4 cycles from feeding the design with inputs
  - DC Test output is correct as expected after 4 clock cycles

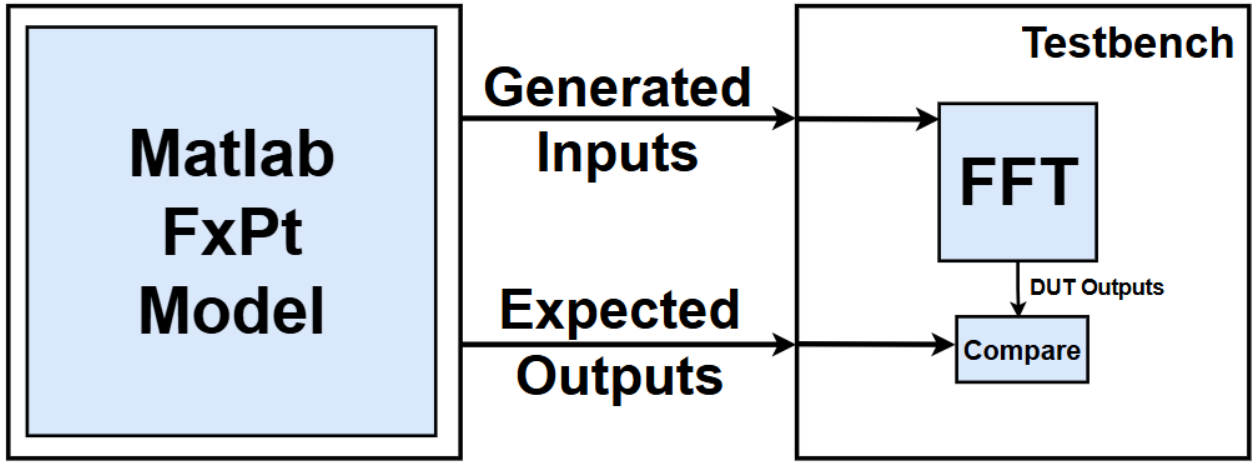
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

# RTL Verification

## Phase 3

# MATLAB Reference Model

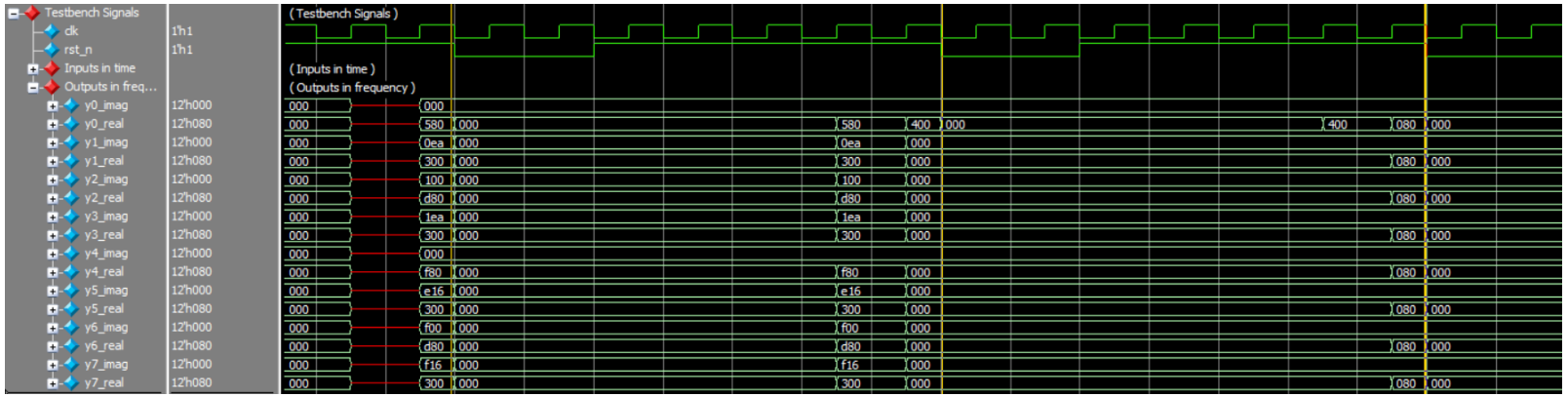
- **Methodology:**
  - Generated 100 random complex input vectors using MATLAB's Model then fed them into design and verified DUT outputs against expected outputs from Matlab model
- **Output Files:**
  - FFT_inputs_real.txt,
  - FFT_inputs_imag.txt
  - FFT_outputs_real.txt
  - FFT_outputs_imag.txt




Verification Environment

# Verification Results

- I ran simulation using **QuestaSim:**
  - Verifying reset functionality




Test Waveform

```
# ====================================
#            SUMMARY REPORT
# ====================================
#  Total Points Checked          : 800
#  Total Passes                  : 800
#  Total Fails                   : 0
#  Average Absolute Error (Real) : 0.000000
#  Average Absolute Error (Imag) : 0.000000
# ====================================
# Simulation finished after 100 tests.
```

Results Summary Report

```
# PASS: k=0 RTL Output = -2 + j-4 | MATLAB Model Output= -2 + j-4
# PASS: k=1 RTL Output = 6 + j-2 | MATLAB Model Output= 6 + j-2
# PASS: k=2 RTL Output = -1 + j1 | MATLAB Model Output= -1 + j1
# PASS: k=3 RTL Output = -1 + j0 | MATLAB Model Output= -1 + j0
# PASS: k=4 RTL Output = 0 + j2 | MATLAB Model Output= 0 + j2
# PASS: k=5 RTL Output = 0 + j0 | MATLAB Model Output= 0 + j0
# PASS: k=6 RTL Output = 0 + j2 | MATLAB Model Output= 0 + j2
# PASS: k=7 RTL Output = -6 + j2 | MATLAB Model Output= -6 + j2
# All outputs MATCH for test number = 88!
```

Test Sample

# Code Coverage Analysis

```
Condition Coverage:
    Enabled Coverage              Bins    Covered     Misses  Coverage
    ----------------              ----       ----     ------  --------
    Conditions                       3          3          0   100.00%
```

Condition Coverage

```
Branch Coverage:
    Enabled Coverage              Bins       Hits     Misses  Coverage
    ----------------              ----       ----     ------  --------
    Branches                        17         17          0   100.00%
```

Branch Coverage

```
Statement Coverage:
    Enabled Coverage              Bins       Hits     Misses  Coverage
    ----------------              ----       ----     ------  --------
    Statements                     135        135          0   100.00%
```

Statement Coverage

```
Total Node Count     =        962
Toggled Node Count   =        962
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (1924 of 1924 bins)
```

Toggle Coverage

```
Total Coverage By Instance (filtered view): 100.00%
```

Total Coverage

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

# ASIC Implementation

## Phase 4

# ASIC Physical Design

## Key Constraints:

```
# ========== Clock Period ==========
create_clock -name clk -period 6.667 [get_ports clk]

# ========== Clock Uncertainties ==========
set_clock_uncertainty -setup 0.1 [get_clocks clk]

set_clock_uncertainty -hold 0.2 [get_clocks clk]
```

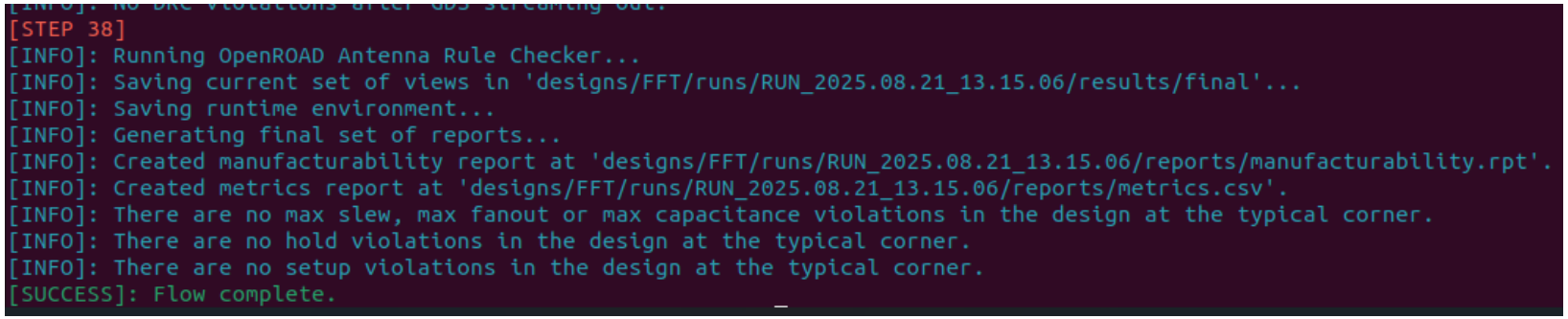
Constraints.sdc

## Key Configurations:

```
# ========== Design Info ==========
set ::env(DESIGN_NAME) "FFT"

# RTL sources
set ::env(VERILOG_INCLUDE_DIRS) "$::env(DESIGN_DIR)/src"
set ::env(VERILOG_FILES) "$::env(DESIGN_DIR)/src/FFT.v"

# Use Verilog mode in Yosys
set ::env(SYNTH_READ_VERILOG) "v"

# ========== Clock ==========
set ::env(CLOCK_PORT) "clk"

# ========== Floorplan ==========
set ::env(DIE_AREA) "0 0 500 500"
```

Configurations.tcl

- I used **OpenLane** tool to run the ASIC flow:




OpenLane flow complete happy message

# Implementation Metrics

- **Technology:** SkyWater 130nm (sky130_fd_sc_hd - High Density library)
- **Metrics**:

| Metric | 4 Stages | 3 Stages | Winner & Analysis |
|---|---|---|---|
| **Max WNS** (Worst Negative Slack) for Setup | **0.05 ns** | **0.01 ns** | 4 Stages |
| **Min WNS** (Worst Negative Slack) for Hold | **0.28 ns** | **0.09 ns** | 4 Stages |
| **TNS** (Total Negative Slack) & **WNS** (Worst Negative Slack) | **0 ns** | **0 ns** | 4 Stages |
| **Max Frequency** | **150 MHz** | **148 MHz** | 4 Stages |
| **Area and Post Rout Utilization** | **224587 $\mu m^2$** , with **97% utilization** | **225053 $\mu m^2$**, with **97% utilization** | For Area: 4 Stages<br>Utilization: Draw |

4 Stages vs 3 Stages metrics table

# Implementation Metrics

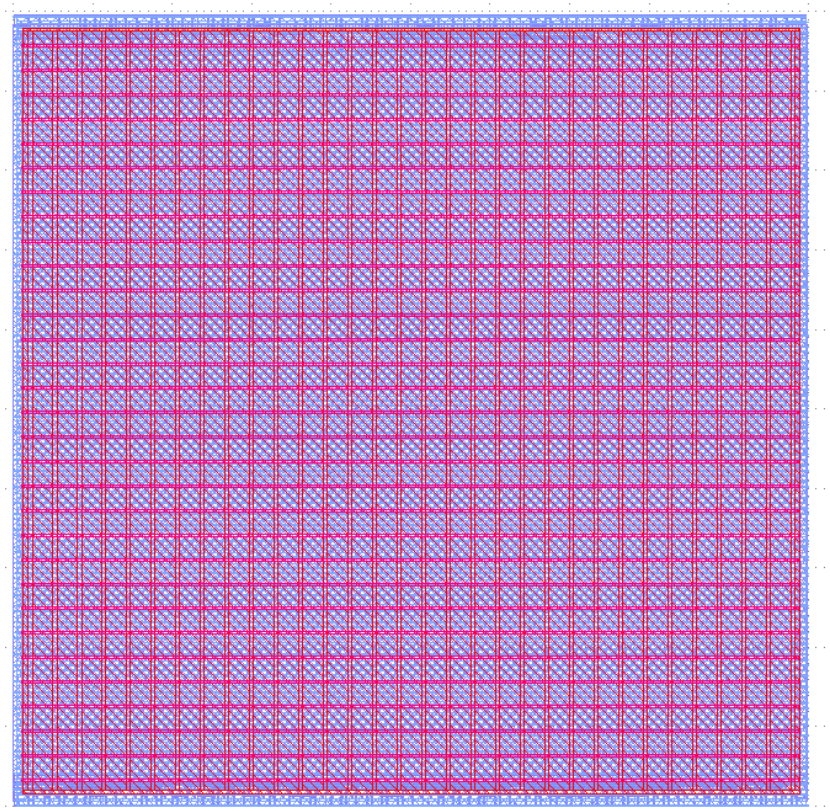
- Standard Cells:

| Metric | 4 Stages | 3 Stages | Difference | Analysis |
|---|---|---|---|---|
| **Total Cells** | **5,414** | **5,241** | **-173 cells (-3.2%)** | The 3-stage design is smaller. |
| **Flip-Flops (dfrtp_2)** | **768** | **568** | **-200 FFs (-25%)** | **Proof** of a removed pipeline stage. This is the primary area saving. |
| **Total Area ($\mu m^2$)** | **64,194** | **59,114** | **-5,080 $\mu m^2$ (-7.9%)** | Confirms the area reduction from fewer FFs. |
| **XOR/XNOR Gates** | **1,027** | **1,020** | **-7 gates** | Nearly identical. The core arithmetic logic is unchanged. |
| **Combinational Cells** | **4,654** | **4,673** | **+19 cells** | The 3-stage design has **slightly more** logic. |

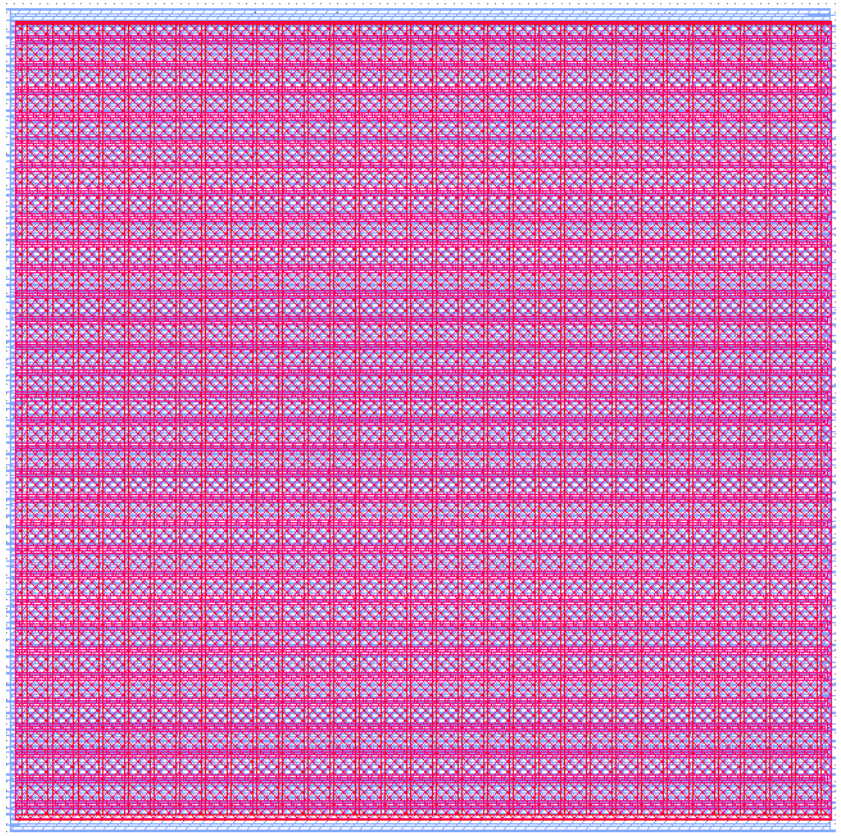
4 Stages vs 3 Stages Standard Cells table

# Implementation Metrics

- **Floorplan**:



4 Stages floorplan



3 Stages floorplan

- **Dimensions**:
  - 488.96 x 478.24

# Implementation Metrics

- **Power Analysis:**

| Metric | Internal Power | Switching Power | Leakage Power | Total Power |
|---|---|---|---|---|
| **3 Stages** | $2.480 * 10^{-2}$ W | $2.30 * 10^{-2}$W | $5.70 * 10^{-8}$W | $4.78 * 10^{-2}$W |
| **4 Stages** | $4.510 * 10^{-2}$ W | $5.10 * 10^{-2}$W | $4.830 * 10^{-8}$W | $9.61 * 10^{-2}$W |

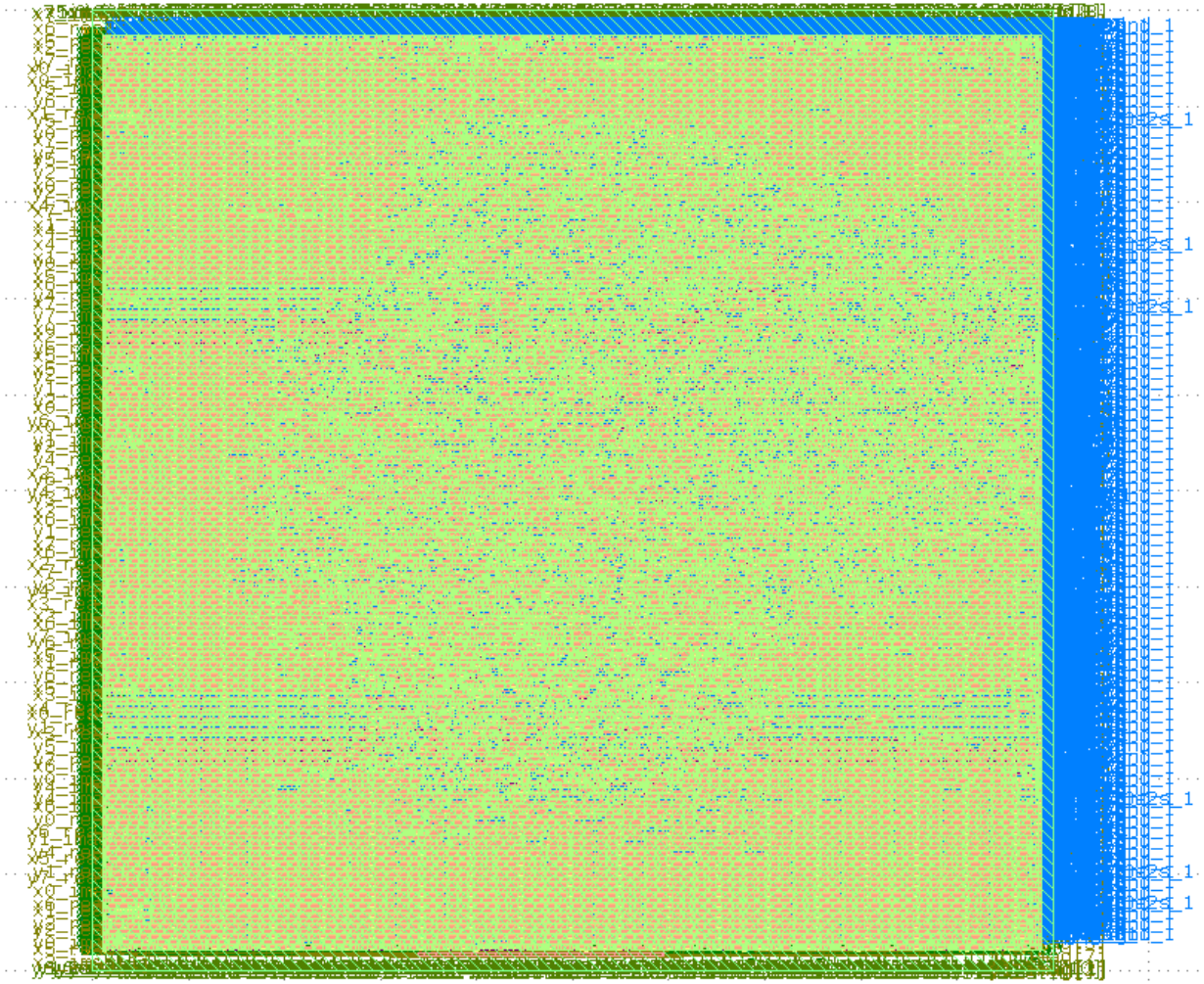
4 Stages vs 3 Stages Power Analysis table

- **So Which is better,**

**3 or 4 Stages ?**

# ASIC Final Product

- **GDS II Screenshot:**



Final GDSII Screenshot

# FPGA Implementation

## Phase 5

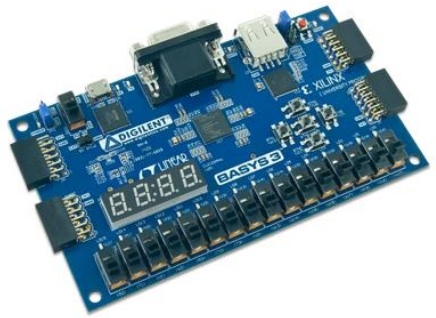
# FPGA Implementation

## Key Constraints:

```
## Clock
set_property -dict { PACKAGE_PIN W5 IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 6.7 -waveform {0 5} [get_ports clk]

## Reset
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports rst_n]

## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
```

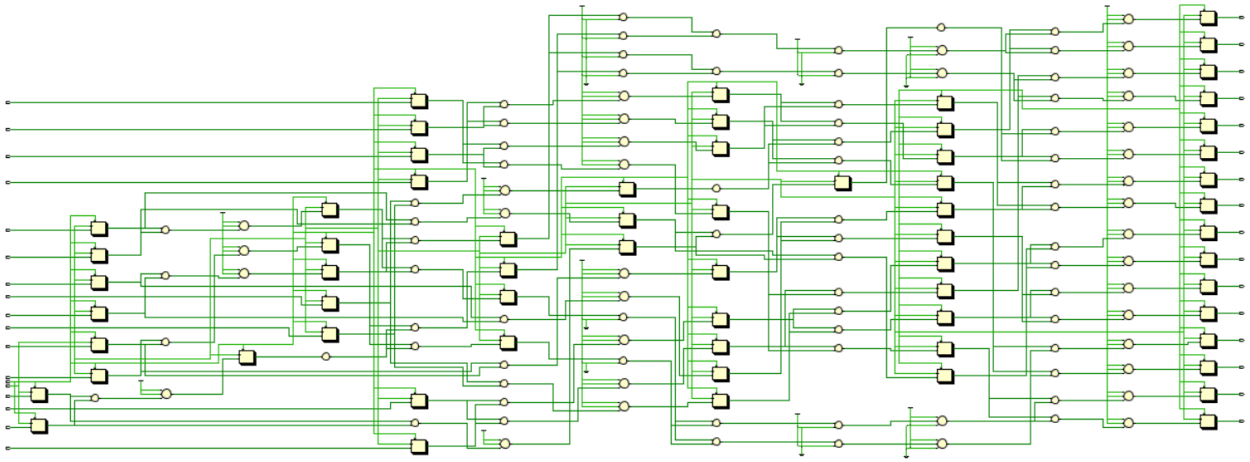
Constraints.xdc

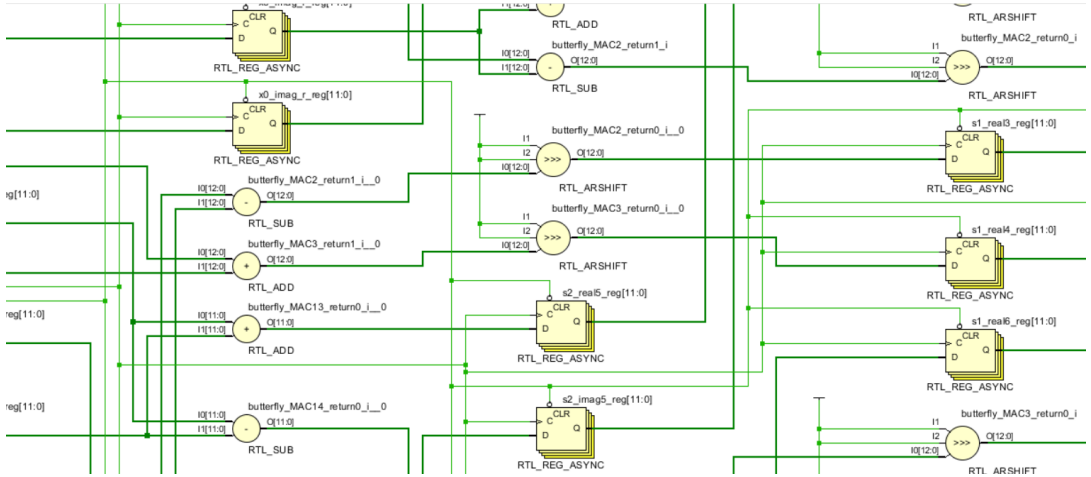
## Board Used:




Artix-7 FPGA

## RTL Elaboration:

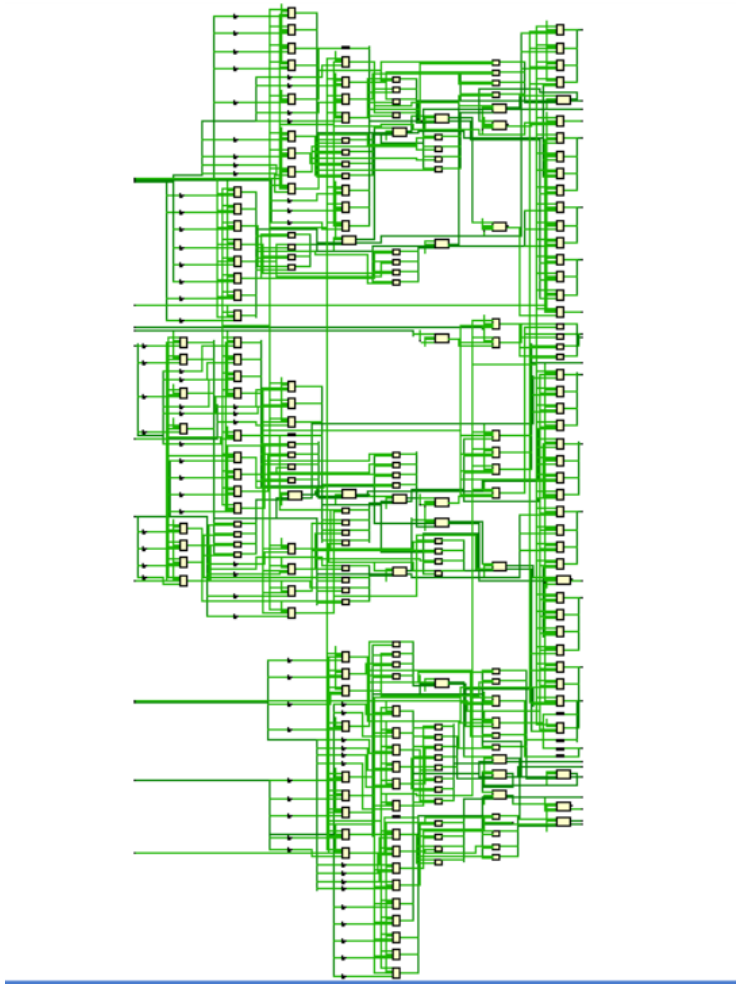


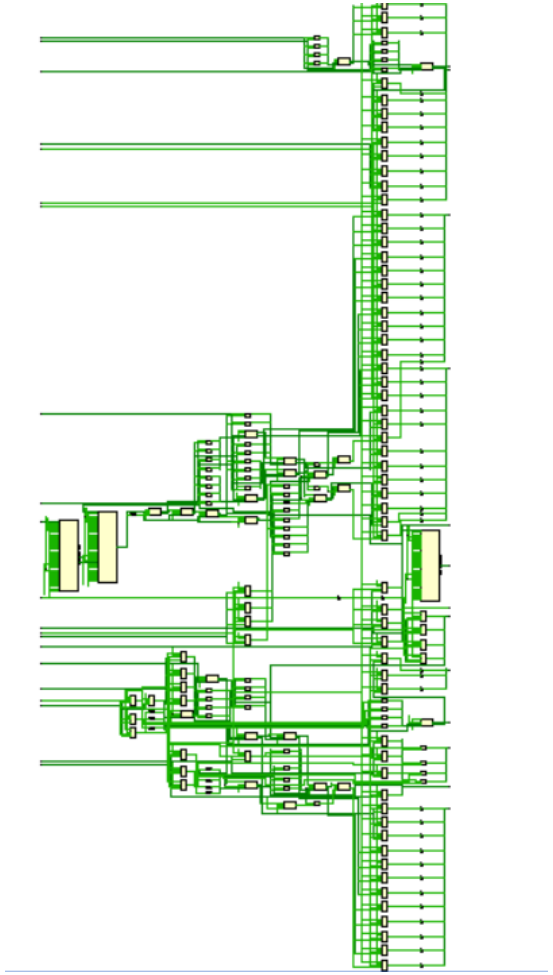


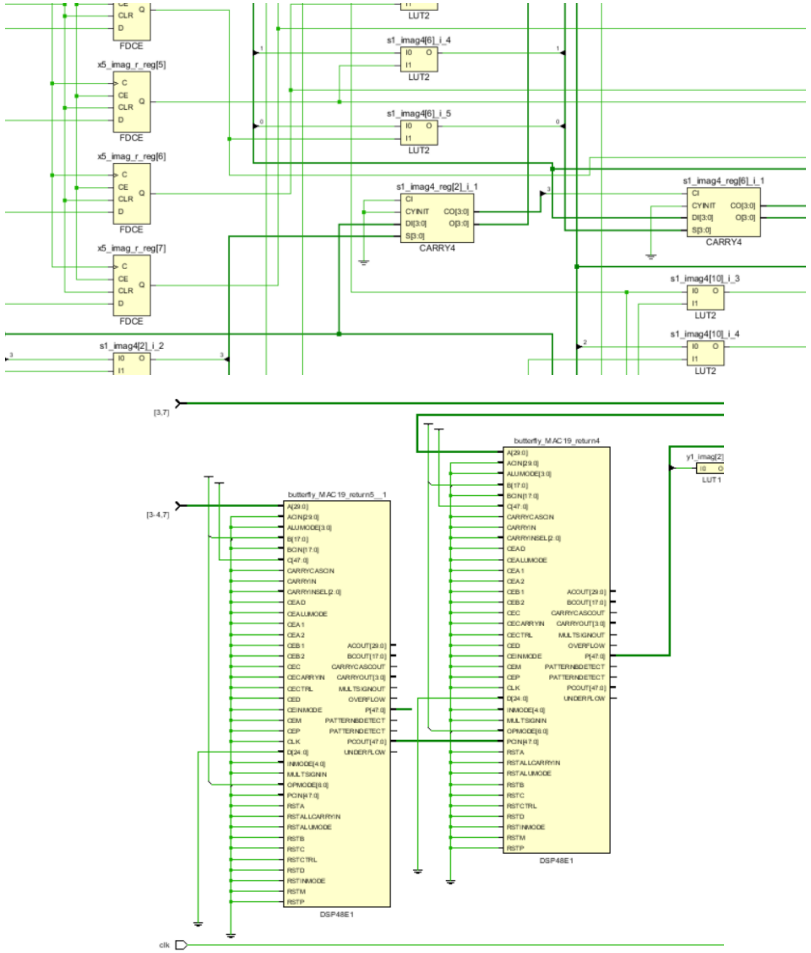

Elaboration Schematic

# FPGA Implementation

## Synthesis:



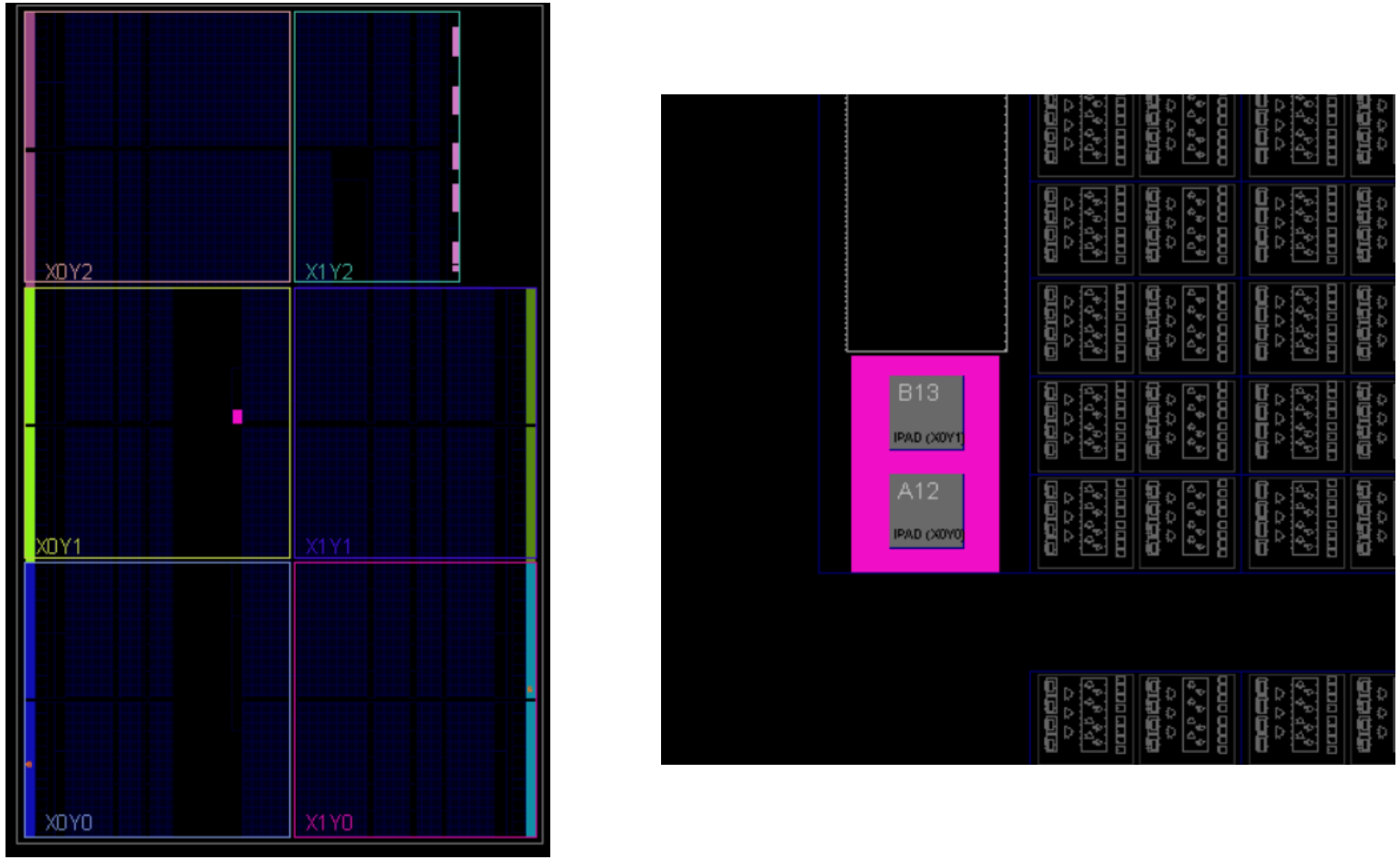




Synthesis Schematic

# FPGA Implementation

- **Device:**




Implemented Device Schematic

# FPGA Implementation Results

| Metric | Slice LUTs | Slice Register | DSPs | WNS | WNS (Hold) | WPWS |
|---|---|---|---|---|---|---|
| Value | 691 LUTs | 768 register | 9 | 0.278 ns | 0.188 ns | 4.5 ns |

FPGA results table

- **So Which is Preferred, ASIC or FPGA ?**

# Future Works

- Upgrade FFT to 16/32/.. points
- Verify RTL against Matlab Model using DPI
- Verify using UVM environment
- Develop SV Assertions to verify important features
- Develop Functional Coverage Analysis

# References

- (PDF) FAST FOURIER TRANSFORM BY VERILOG
- Signals Lecture #9
- FFT Implementation on Mathworks

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

# Thanks !