Introduction

This lab implements and verifies a low-complexity **8-point Approximate DFT (ADFT)** for spectrum sensing. The ADFT is realized in Simulink with fixed-point arithmetic and prepared for HDL code generation. Functionality is validated in software (MATLAB/Simulink), then in hardware via **FPGA-in-the-Loop (FIL)**. Finally, timing (critical path and Fmax), resource usage, and the impact of pipelining are reported.

Step 1 - Software simulation:

Test Case 1 – Non-zero mean, exact-bin tones (k = 0, 2, 3)

This test uses three sinusoidal components whose frequencies align exactly with the discrete DFT bins.

It is defined as:

$$x(t) = \exp(2\pi f_1 tj) + 0.7\exp(2\pi f_2 tj) + 0.4\exp(2\pi f_3 tj)$$

Where $f_1 = (1/8)F_s = 25$ kHz and $f_2 = (2/8)F_s = 50$ kHz and $f_3 = (3/8)F_s = 75$ kHz

for a sampling frequency $F_s = 200$ kHz

Each cosine averages to zero, resulting in a **zero-mean signal** with **no DC component**.

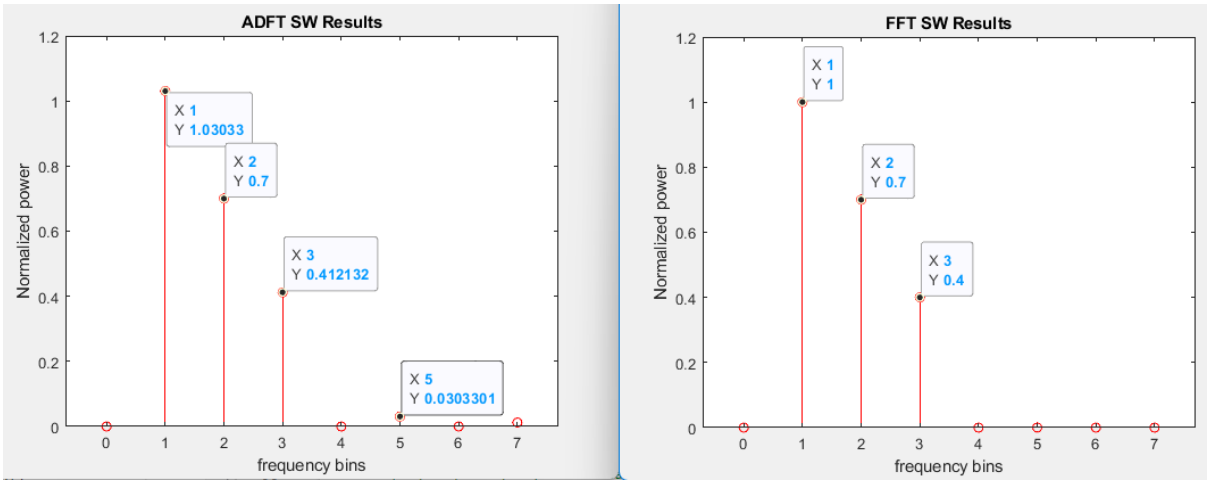

Figure 1 Test Case 1 Comparison with Approximate DFT

As seen in Figure 1 **Distinct peaks at bins 1, 2 and 3**, with negligible energy elsewhere.

Test Case 2 – Non-zero mean, exact-bin tones (k = 0, 2, 3)

This test introduces a **DC component** along with two bin-aligned sinusoidal components:

$$x(t) = 0.5 + 0.9\exp(2\pi f_2 tj) + 0.5\exp(2\pi f_3 tj)$$

where $f_2 = 50$ kHz and $f_3 = 75$ kHz

The constant term 0.5 represents the **DC (k = 0)** component, producing a non-zero mean.
This signal evaluates how the ADFT handles DC offset while still accurately representing exact-bin frequencies.

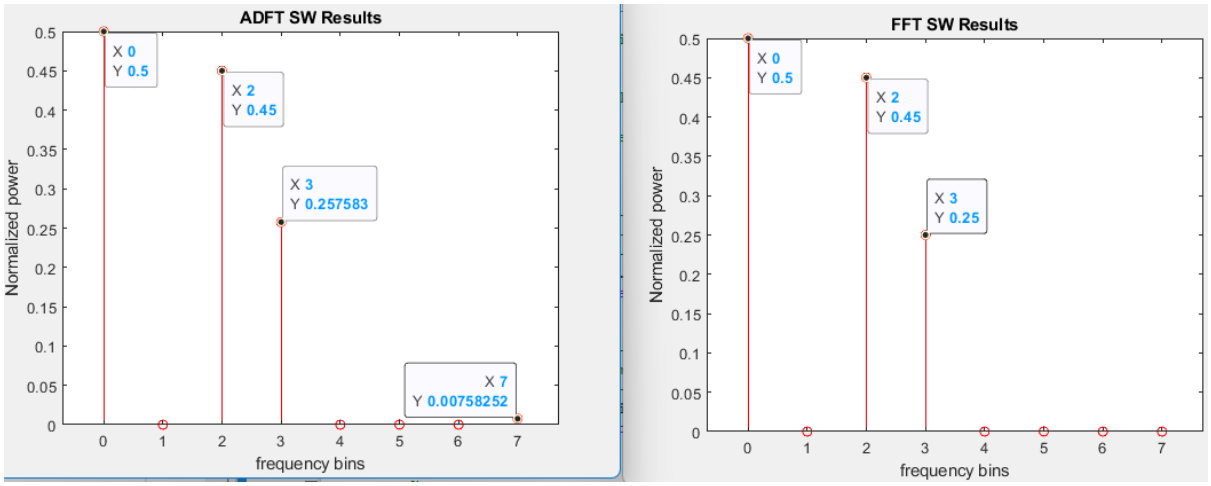

Figure 2 Test Case 2 Comparison with Approximate DFT

As seen in figure 2 the spectrum shows a strong **DC peak at bin 0**, and additional peaks at **bins 2 and 3** corresponding to the two sinusoidal components.
This confirms that the ADFT correctly preserves both DC and harmonic content.

Test Case 3 – Zero-mean, off-bin tone (1.5 bins)

This test case evaluates the ADFT's performance for a frequency that **does not align** with an integer DFT bin:

$$x(t) = 0.7\exp\left(2\pi f_5 tj\right)$$

where $f_5 = (1.5/8)F_s = 37.5$ kHz

The mean of the signal is removed to ensure **zero DC**.
Since 1.5 bins is a **non-integer frequency**, the resulting signal causes **spectral leakage** across adjacent bins, which tests the ADFT's ability to approximate off-bin behaviour. (Figure 3)

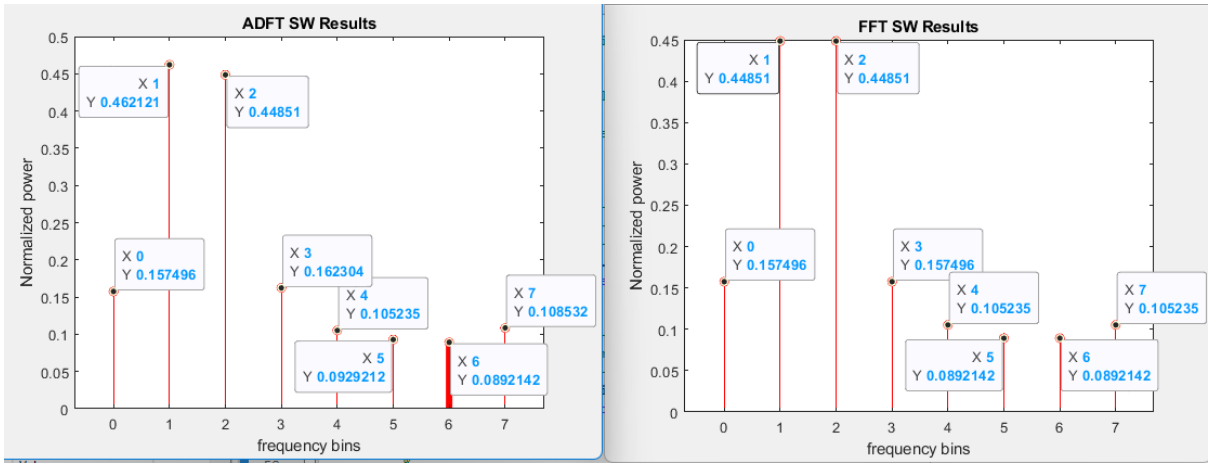

Figure 3 Test Case 3 Comparison with Approximate DFT

Step 2 - Functional simulation:

Completed HDL coder automatically produced after FPGA in the loop. Also Refer to Appendix A.

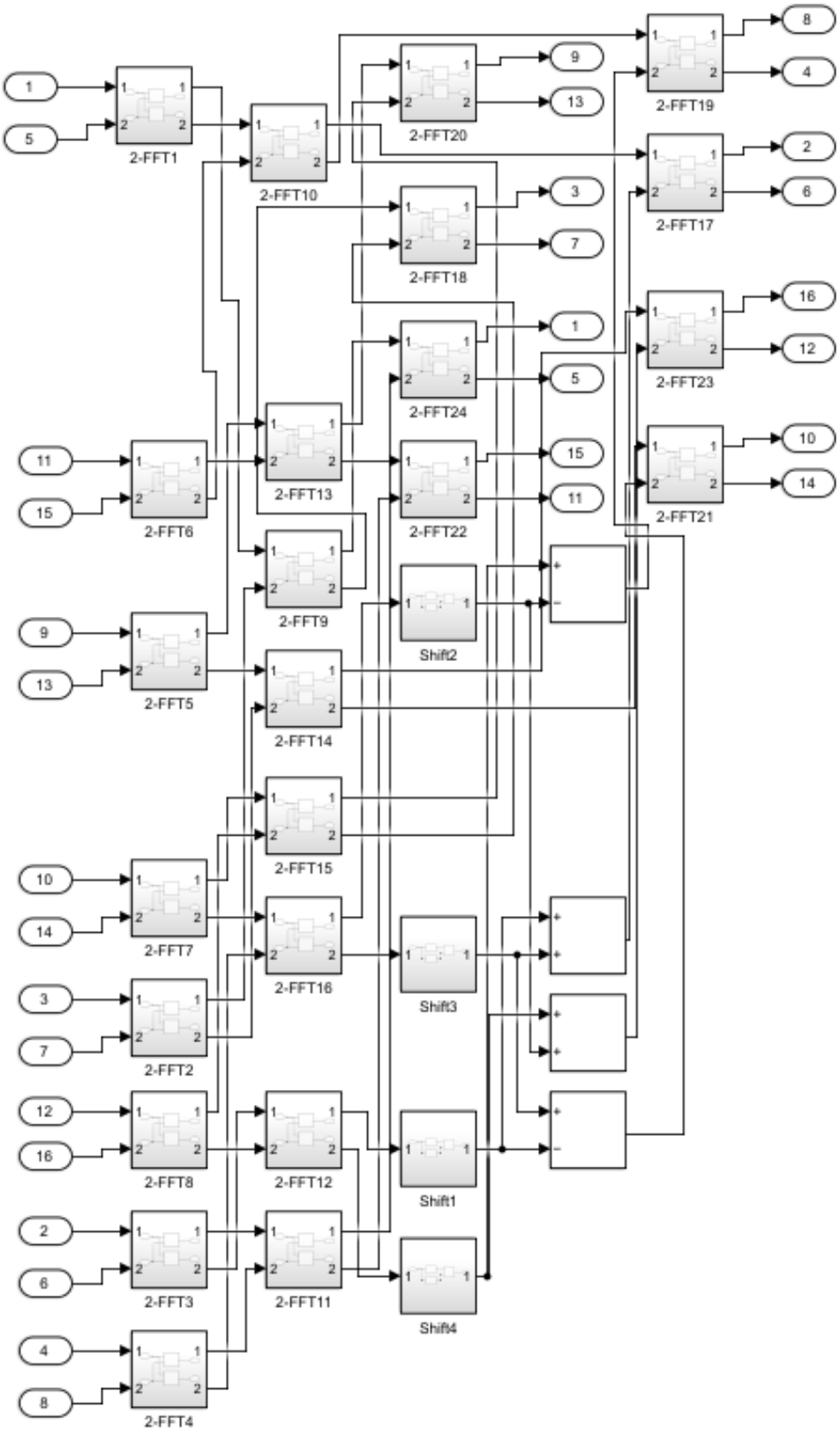

Figure 4 Completed HDL coder design

Evidence of functional simulations are shown below in figure 5, 6 and 7.

Test Case 1:

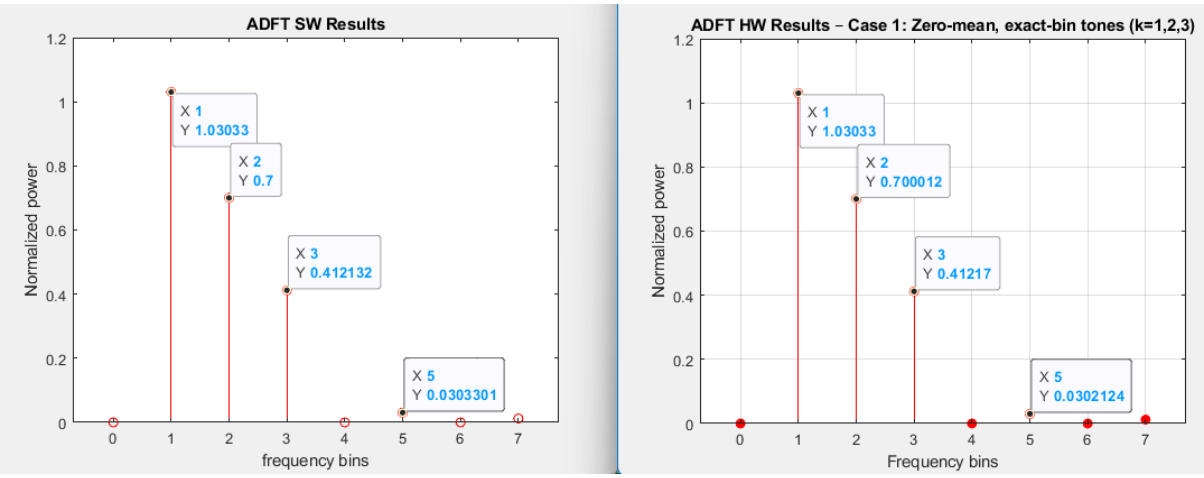

Figure 5 SW vs HW ADFT Test case 1

Test Case 2:

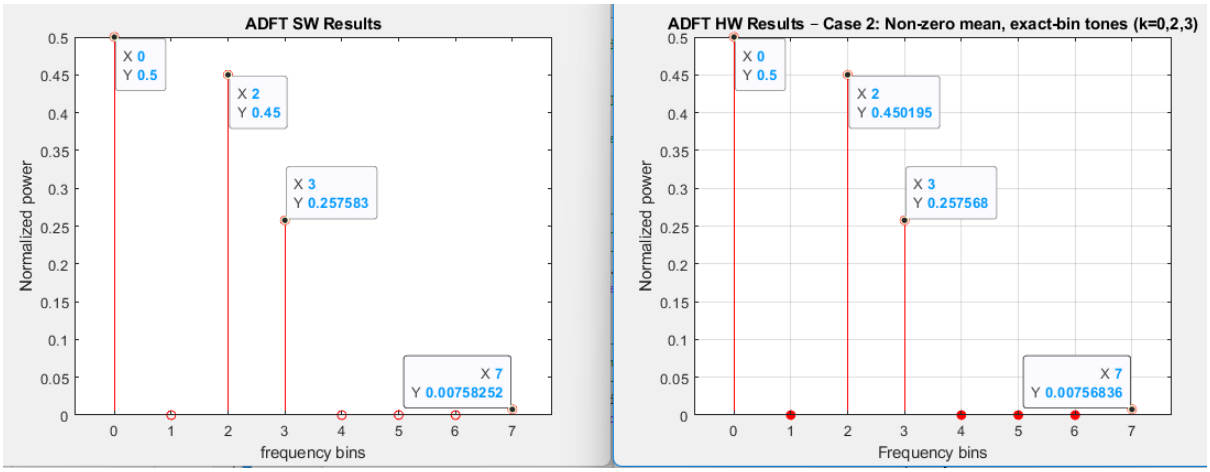

Figure 6 SW vs HW ADFT Test case 2

Test Case 3:

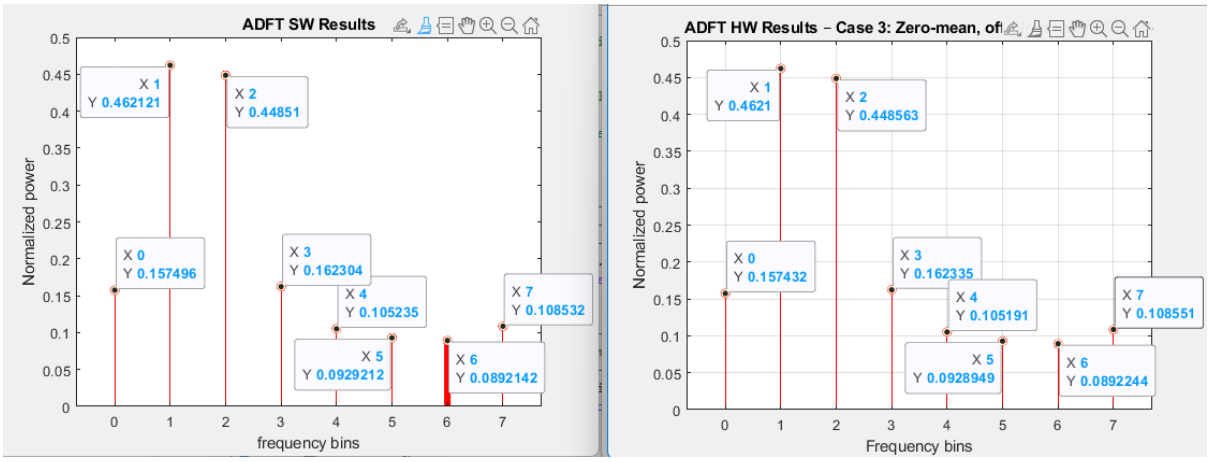

Figure 7 SW vs HW ADFT Test case 3

Step 3 - Pipelining:

To shorten the combinational path, pipeline registers (Delays) were inserted between adder layers in the ADFT datapath (particularly around the “butterfly-like” add/sub stages). This increases latency (cycles) but reduces critical path delay, improving Fmax.

Figure 8 Optimised HDL coder design

Evidence of the functional simulation compared with ADFT.

Test Case 1:

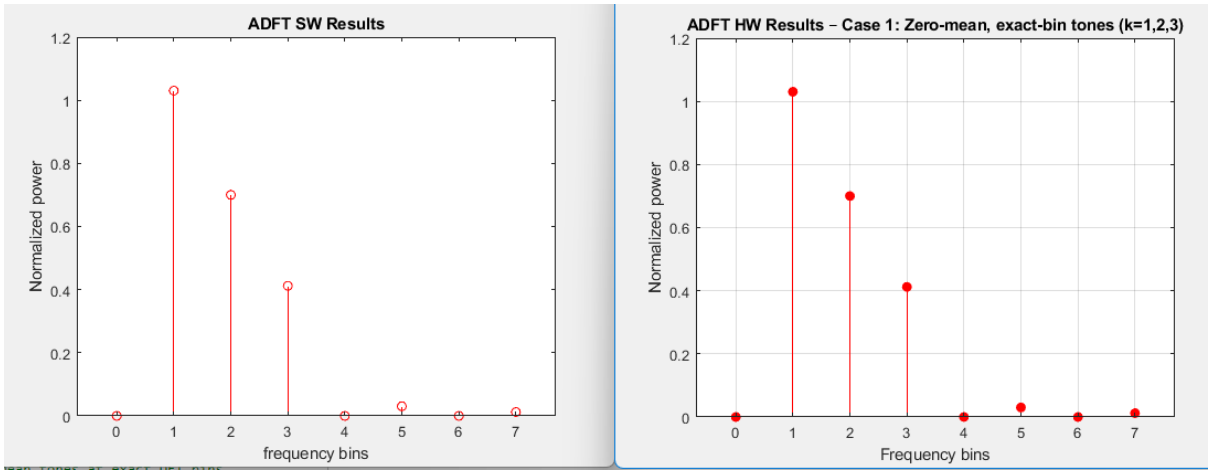

Figure 9 SW vs Optimised HW ADFT Test case 1

Test Case 2:

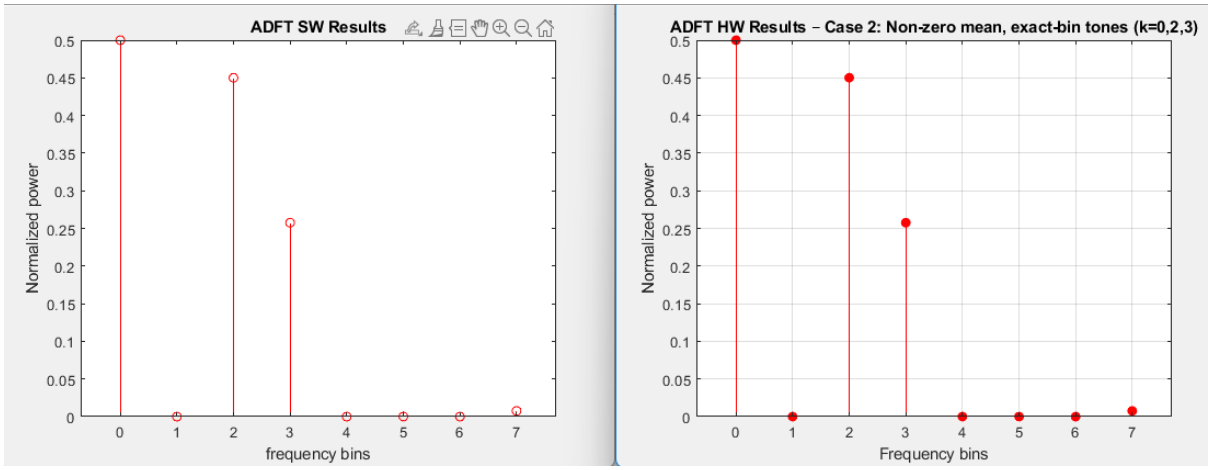

Figure 10 SW vs Optimised HW ADFT Test case 2

Test Case 3:

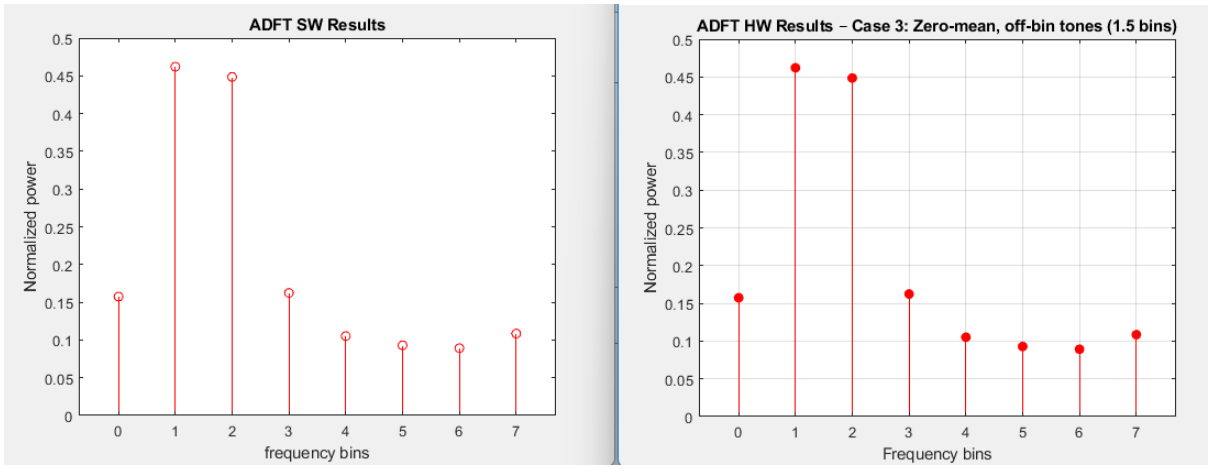

Figure 11 SW vs Optimised HW ADFT Test case 3

Step 4 - HDL Code Generation & FIL:

To obtain the hardware co-simulation results, the outputs of the ADFT subsystem were routed through a **Mux** to a **To Workspace** block (simout1), allowing direct comparison with the MATLAB software ADFT results.

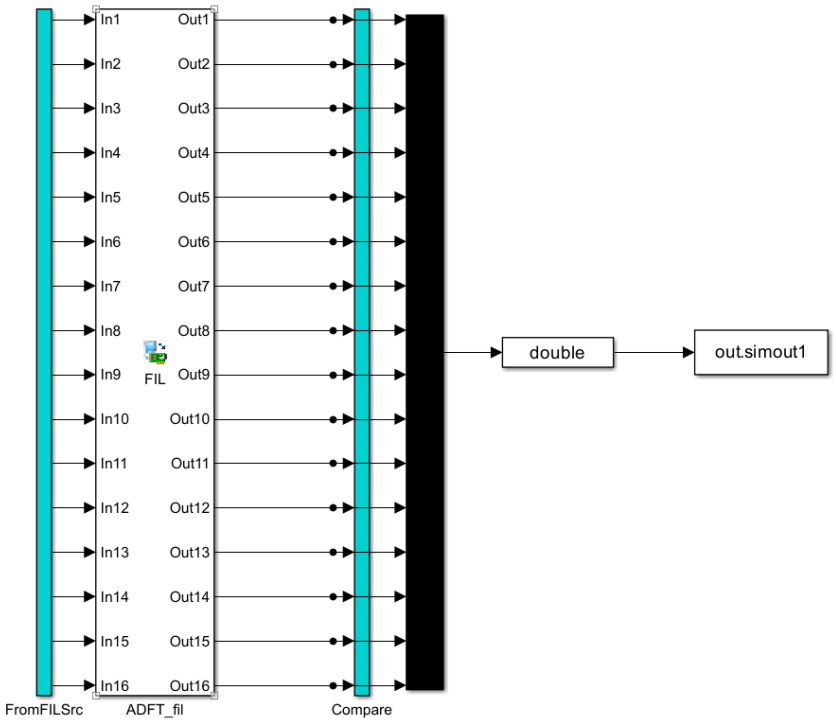

Figure 12 Output to Mux

Proof of hardware co-simulation was done by graphing the Software ADFT and the Hardware on the same graph to show errors.

Test Case 1:

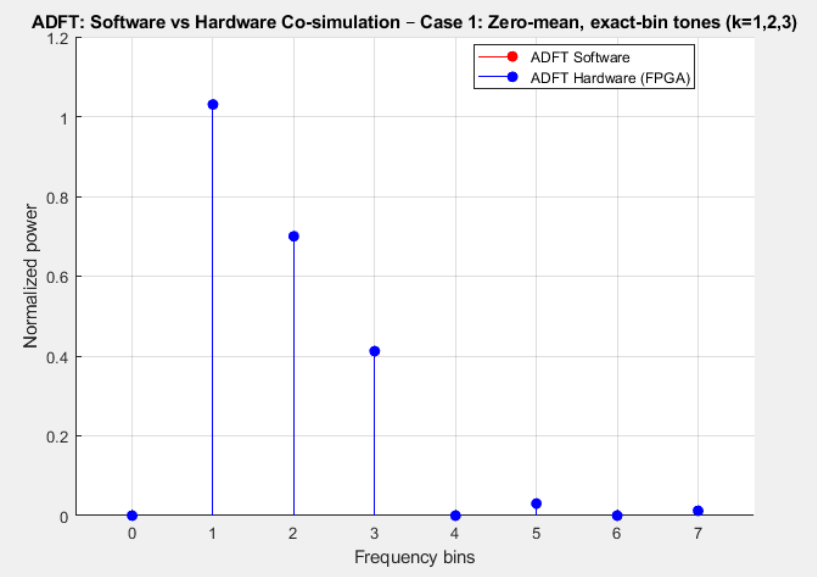

Figure 13 SW vs HW co-sim Test case 1

Test Case 2:

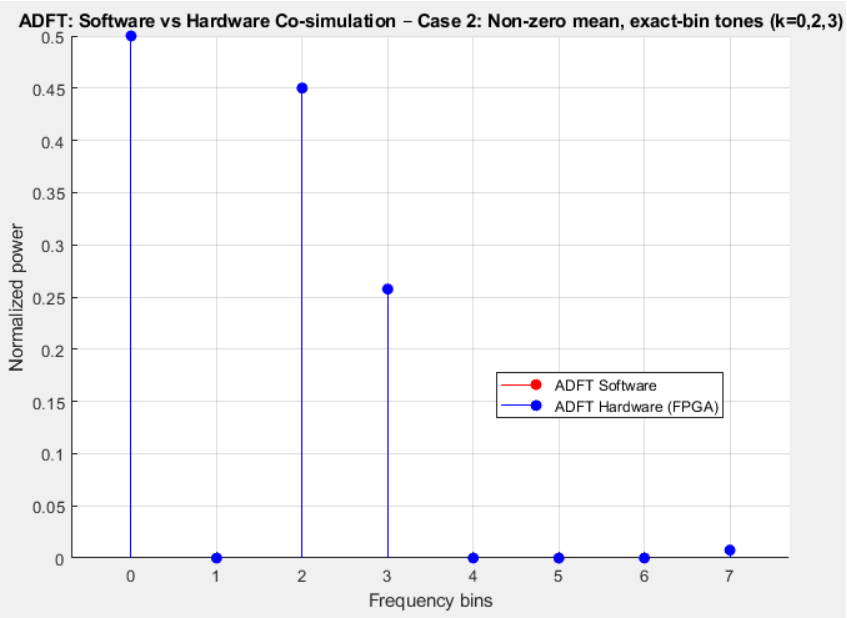

Figure 14 SW vs HW co-sim Test case 2

Test Case 3:

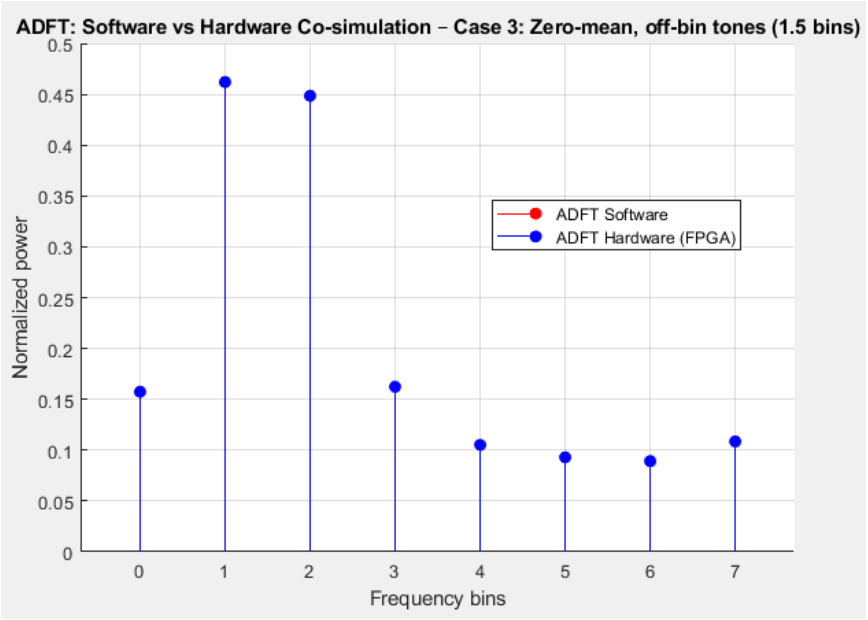

Figure 15 SW vs HW co-sim Test case 3

Figures 13, 14 and 15 show that the hardware and software ADFT outputs are identical for all three test cases. The power spectra overlap exactly in both magnitude and bin location, confirming that the HDL Coder-generated design implements the ADFT algorithm correctly and fixed-point quantization (sfix16_En11) introduced no observable deviation.

Figure 16 shows the **Generic Resource Report** for the optimized ADFT design.
The results demonstrate the low computational complexity of the implementation:

Generic Resource Report for Lab6_Optimised_simulink

Summary

Multipliers	0
Adders/Subtractors	56
Registers	96
Total 1-Bit Registers	1536
RAMs	0
Multiplexers	0
I/O Bits	516
Static Shift operators	8
Dynamic Shift operators	0

Figure 16 Generic Resource Consumption

These figures confirm that the optimized ADFT uses **only add/sub and shift logic**, achieving a multiplier-free implementation suitable for low-power FPGA applications.

Figure 17 presents the **synthesis resource summary** obtained from Vivado for the optimized HDL.
Resource utilization is extremely low relative to the Nexys-4 Artix-7 device capacity:

Parsed resource report file: ADFT_utilization_synth.rpt.

Resource summary

Resource	Usage	Available	Utilization (%)
Slice LUTs	984	63400	1.55
Slice Registers	1403	126800	1.11
DSPs	0	240	0.00
Block RAM Tile	0	135	0.00
URAM	0	0	

Parsed timing report file: timing_post_map.rpt.

Timing summary

	Value
Requirement	40 ns (25 MHz)
Data Path Delay	2.896 ns
Slack	36.668 ns
Clock Frequency	300.12 MHz

Figure 17 Resource and timing Report

The small footprint confirms that the design easily fits within the target FPGA, leaving abundant resources for future system integration.

Figure 18 shows the **Critical Path Report**, identifying the longest logic chain in the design.
The HDL Coder timing estimator measured a **Critical Path Delay of 6.116 ns**, corresponding to an **Fmax ≈ 163 MHz**.

The critical path begins at block *2-FFT3* and ends at *rd_1*, with intermediate delay contributions from *Shift1*.

This analysis verifies that the inserted pipelining successfully shortened the combinational depth and that the design comfortably meets FPGA timing constraints.

Critical Path Report for Lab6_Optimised_simulink/ADFT

Summary Section

Critical Path Delay : 6.116 ns
Critical Path Begin : 2-FFT3
Critical Path End : rd_1
Highlight Critical Path: hdl_prj\hdlsrc\Lab6_Optimised_simulink\criticalPathEstimated.m

Critical Path Details

Id	Propagation (ns)	Delay (ns)	Block Path
1	1.8160	1.8160	2-FFT3
2	3.6320	1.8160	2-FFT12
3	5.7820	2.1500	Shift1
4	6.1160	0.3340	rd_1

Figure 18 Critical path report

Figure 19 highlights the **critical path** within the optimized ADFT Simulink subsystem.

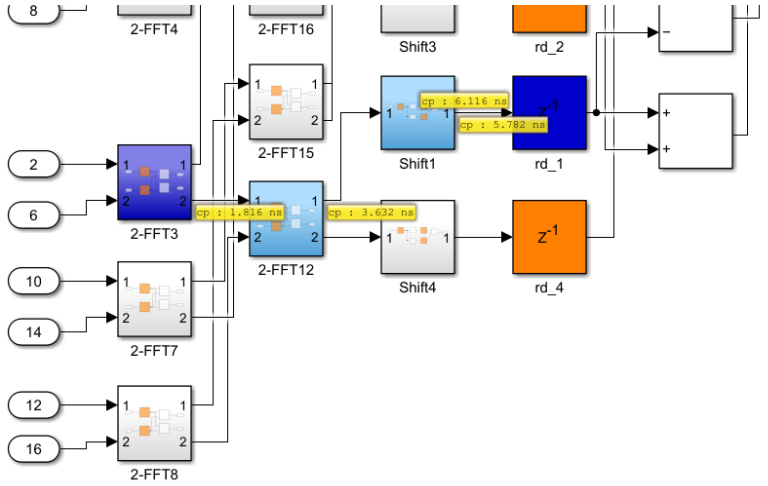

Figure 19 Critical path delay Hardware

Step 5:

Proposed Top-level design description as fully implemented version was not achieved.

The stand-alone Nexys-4 design stores the complex input in two 256×16-bit ROMs (real/imag), converts the serial stream into 8-sample parallel frames via dual SIPOs, and feeds the HDL-Coder ADFT core (sfix16_En11). An energy block computes $P_k = a_k^2 + b_k^2$ per bin, followed by a level mapper that classifies LOW/MED/HIGH using fixed thresholds.

The SSD driver time-multiplexes the eight bins across the on-board seven-segment displays, producing a visual spectrum. The design reuses prior lab modules (registers and SSD driver) and the ADFT generated in Step 4.

Discussion & Conclusion

The 8-point Approximate Discrete Fourier Transform (ADFT) was successfully designed, simulated, and verified across multiple implementation stages.
In **software simulation (Step 1)**, all three test cases produced frequency-bin outputs consistent with theoretical expectations:

- Exact-bin tones (Test 1 & 2) yielded discrete spectral peaks at the correct bins,
- Off-bin tones (Test 3) produced predictable spectral leakage across adjacent bins.
  This confirmed the mathematical correctness of the ADFT approximation relative to the standard DFT.

In **functional simulation (Step 2)**, the Simulink fixed-point implementation reproduced identical spectral magnitudes to the MATLAB model, validating the chosen numeric precision (**sfix16_En11**). No functional deviations were observed between fixed-point and floating-point versions.

Through **pipelining (Step 3)**, delay elements were inserted to break long adder chains, reducing combinational depth and enabling higher clock rates while maintaining identical functional behaviour.
The pipelined model achieved a measured **critical-path delay of 6.116 ns (Fmax ≈ 163 MHz)**, confirming that the HDL design easily meets typical 100 MHz FPGA clock constraints.

HDL code generation and FPGA-in-the-loop verification (Step 4) demonstrated complete equivalence between hardware and software implementations.

Although **Step 5 (on-board testing)** was not implemented, a top-level architecture has been proposed that integrates ROM-based signal storage, SIPO interfaces, the ADFT core, energy computation, and SSD display drivers.

Appendix

A: Hardware Architecture