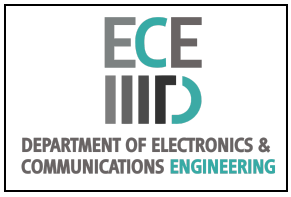


Advanced Embedded Logic Design (ECE - 573)
Project End Sem Report

OFDM Receiver with Synchronization

Mentor: Jai Mangal, PhD (ECE), IIITD
Faculty Advisor: Dr. Sumit J Darak, Associate Professor (ECE), IIITD

Group - 6
Member Details :

Harsh Nangia	Kunal Sharma	Bharti Bansal	Ayana
2022199	2022262	2022136	2022123

End Sem Deliverables :

The end-semester project deliverables encompass the design and analysis of IEEE 802.11a Architecture on FPGA, including a Root Raised Cosine (RRC) filter, packet detection, coarse and fine carrier frequency offset (CFO) estimations using phase angle calculations, channel estimation, one tap equalizer and a 64-point FFT IP. The IPs are optimized through pipelining, loop unrolling, and array partitioning, with efficient wordlength optimizations to minimize fixed-point bit-widths. Memory-mapped IPs with AXI protocol are designed. Further additionally separate design is made where interrupt is also used when using DMA. Performance metrics include Error Vector Magnitude (EVM) and Bit Error Rate (BER) calculations across various SNRs, Root Mean Square Error (RMSE) comparison between Programmable Logic (PL) and Processing System (PS) results, and acceleration factor evaluation of FPGA versus software. Finally, FPGA resource utilization (LUTs, FFs, DSPs, BRAMs) is analyzed, providing synthesis reports, performance plots, and optimization insights.

1. Introduction

The project focuses on designing and developing an IEEE 802.11a architecture on the Zedboard platform. The primary goal is implementing an Orthogonal Frequency Division Multiplexing (OFDM) receiver with synchronisation, a critical component in modern wireless communication systems. The report outlines the work done as part of the project in the Advanced Embedded Logic Design course in Winter25 semester at Indraprastha Institute of Information Technology Delhi.

2. Project Overview: IEEE 802.11a Transceiver on Zedboard

This project focuses on designing and implementing an IEEE 802.11a transceiver on the Zedboard FPGA platform. The transceiver is based on Orthogonal Frequency Division Multiplexing (OFDM) and includes key signal processing components such as IFFT/FFT, cyclic prefix insertion/removal, synchronisation, CFO estimation, channel estimation, and equalisation. The primary goal is to optimize the processing time by refining algorithms and leveraging FPGA-based acceleration. Key performance metrics such as Error Vector Magnitude (EVM), Bit Error Rate (BER), and execution time on PS and PL are evaluated under different SNR conditions.

Project Scope & Objectives:

1. **Software Implementation:** Develop a processor-based transceiver using optimised algorithms (mid sem goal).
2. **Performance Analysis:** Measure BER, EVM, and execution time under various channel conditions.
3. **FPGA Acceleration:** Implement key modules as hardware IPs for efficient execution. (end sem goal)
4. **Optimisation Strategies:** Reduce computational overhead using loop unrolling, memory reuse, and function merging to enhance real-time performance.

This project aims to improve the efficiency of IEEE 802.11a-based wireless communication systems by optimizing execution time while maintaining accuracy and robustness.

3. Methodology

3.1 OFDM Block Design

The OFDM block design consists of the following steps:

- Modulation
- Serial-to-Parallel (S/P) conversion.
- Inverse Fast Fourier Transform (IFFT).

- Insertion of Cyclic Prefix (CP) and preamble.
- Parallel-to-Serial (P/S) conversion.
- Digital-to-Analog Conversion (DAC).
- Demodulation:
- Packet detection.
- Fast Fourier Transform (FFT).
- Removal of CP.
- Channel estimation and equalisation.
- Coarse and fine CFO estimation.
- Demapping

3.2 Signal Parameters

The signal parameters used in the project are as follows:

- **Center Frequency: 5 GHz**
- **Bandwidth: 20 MHz**
- **Sample Time: 50 ns**
- **FFT Size: 64**
- **Threshold: 0.65**

3.3 Frame Design

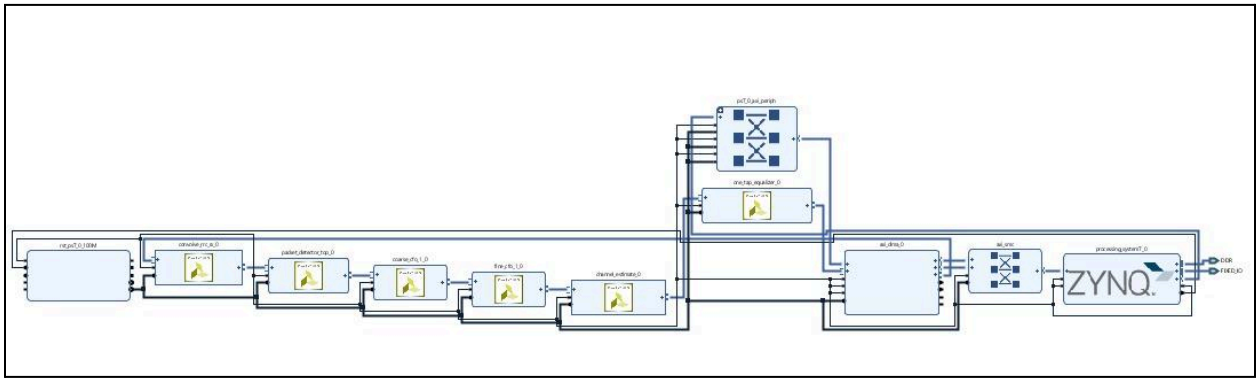
The transmit signal is designed with the following components:

- **Short Preamble:** 160 samples.
- **Long Preamble:** 160 samples.
- **Payload:** 80 samples per frame.
- **Oversampling:** The signal is oversampled by a factor of 2, resulting in 960 samples.
- **Root Raised Cosine (RRC) Filter:** Applied to the transmit signal to reduce inter-symbol interference (ISI).

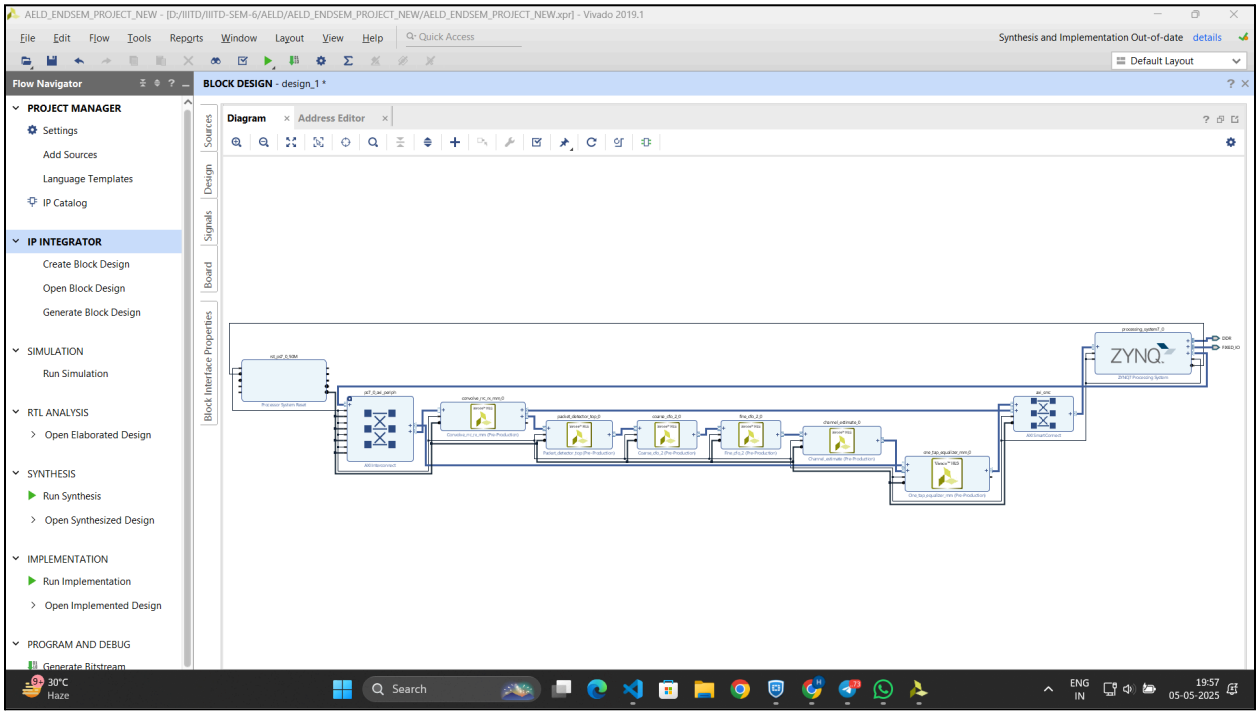
4. Implementation and Results

4.1 Vivado Block Design

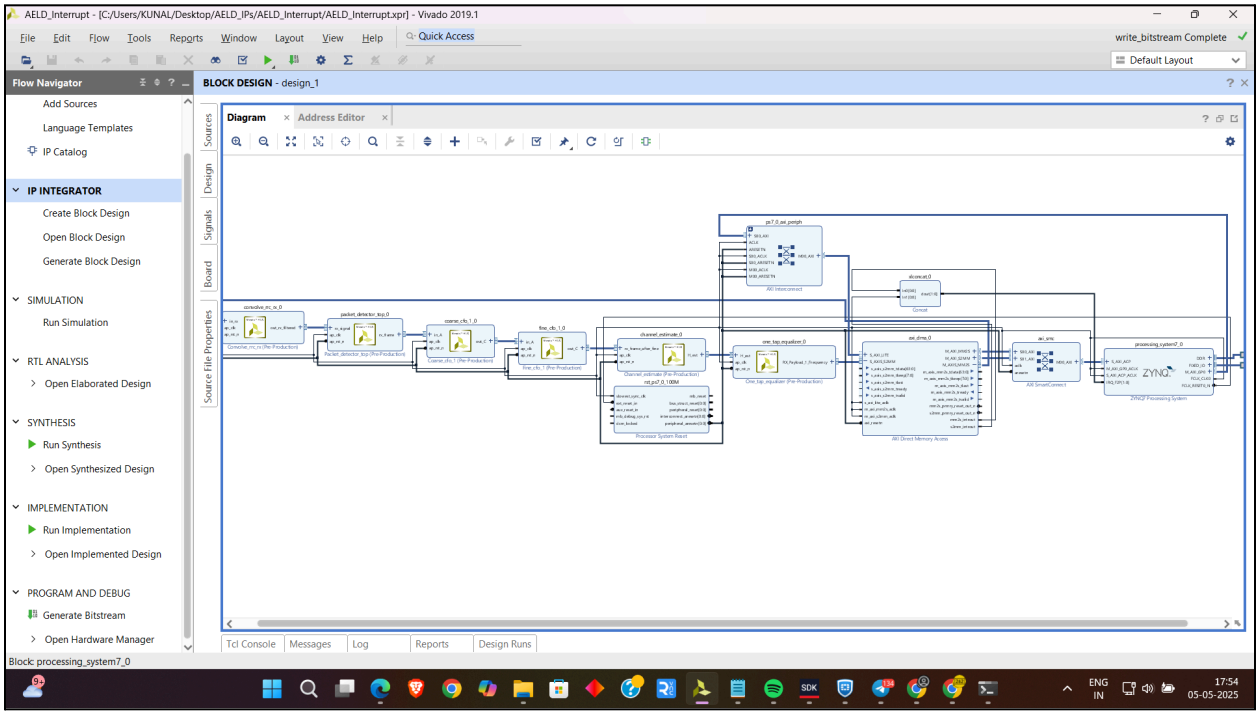
4.1.1 with DMA

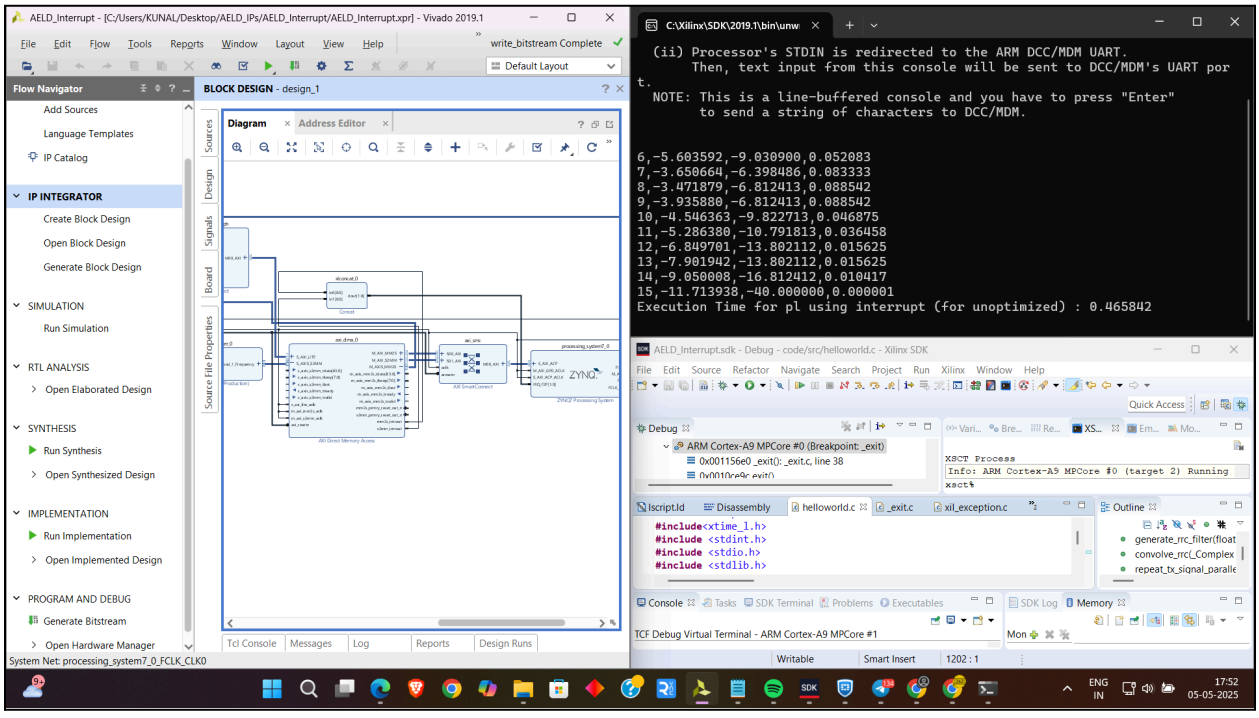


4.1.2 without DMA



4.1.3 using Interrupt



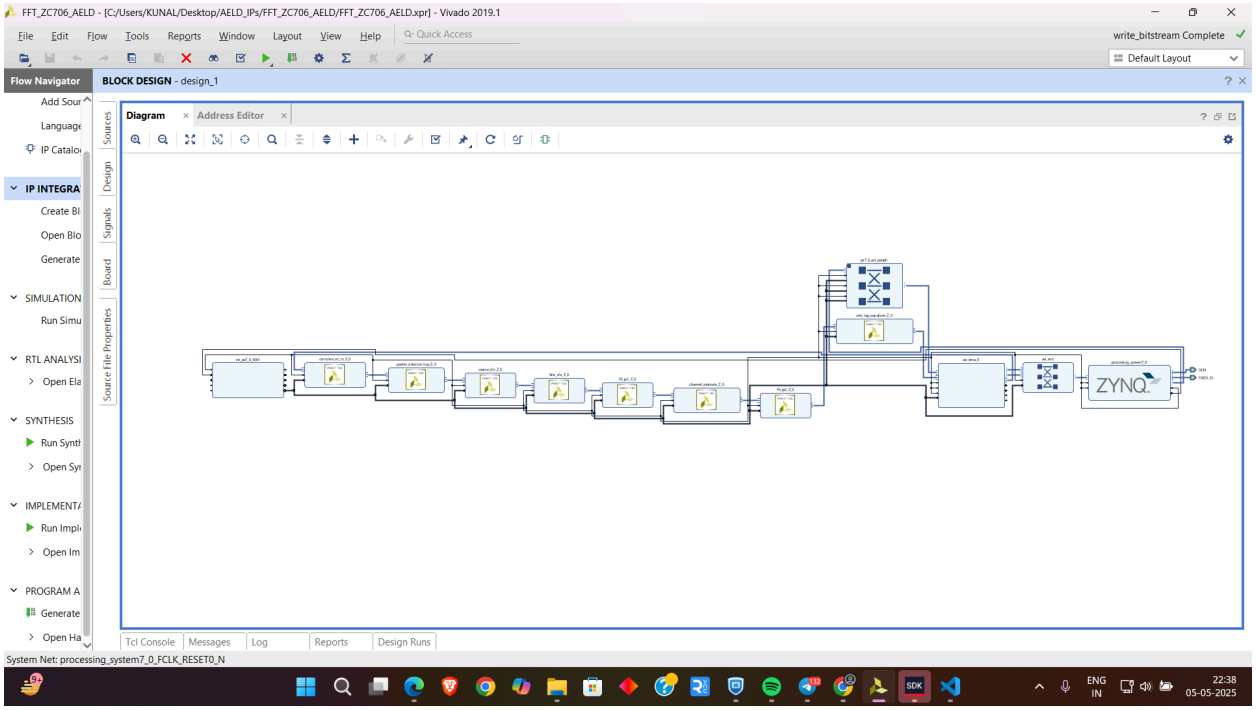


To improve system performance, DMA transfers were configured using interrupts instead of polling. This change led to a noticeable reduction in overall execution time, as the processor no longer remained busy-waiting during data transfers.

- DMA with Polling: 0.52 seconds

- DMA with Interrupt: 0.46 seconds
- Improvement: ~11.5% reduction in execution time
- Benefit: Frees CPU cycles for other tasks during data transfer
- Conclusion: Interrupt-based DMA is more efficient for our use-case

4.1.4 using FFT Ip

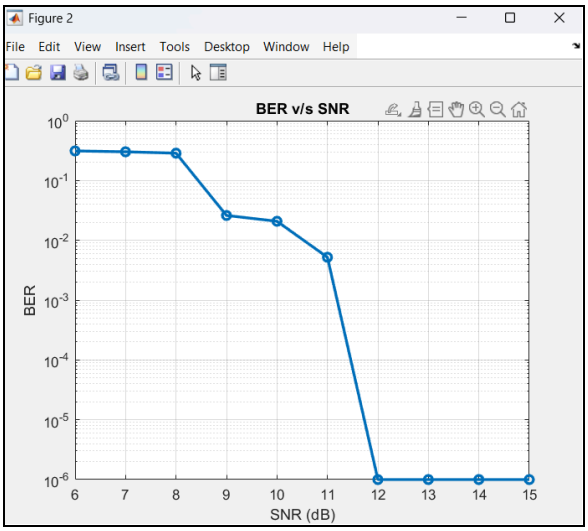


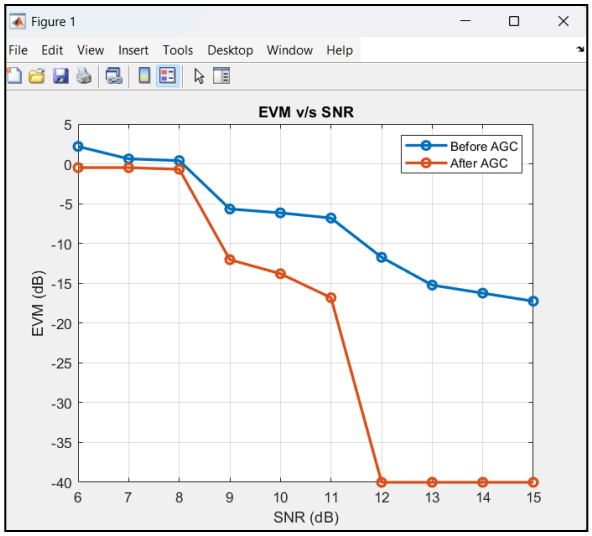
4.2 Functionality Check

The following graphs were obtained by plotting the SNR, BER, and EVM values on the graph using the
Matplot Library in Python.

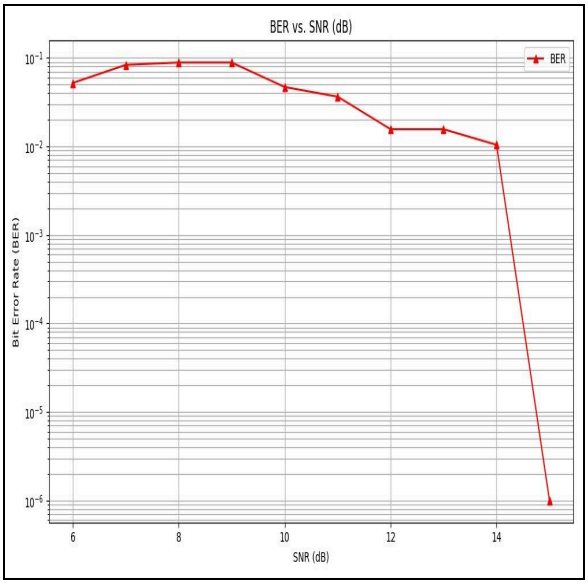
For SNR 6 to 15

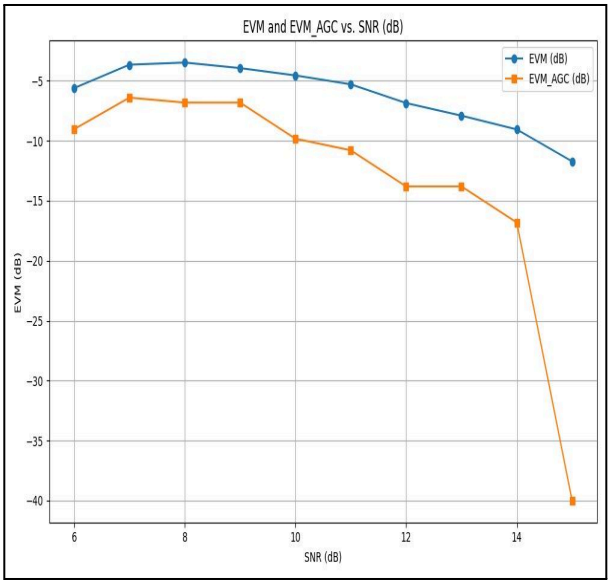
Expected Graphs [Source: Matlab Code provided my Mentor]

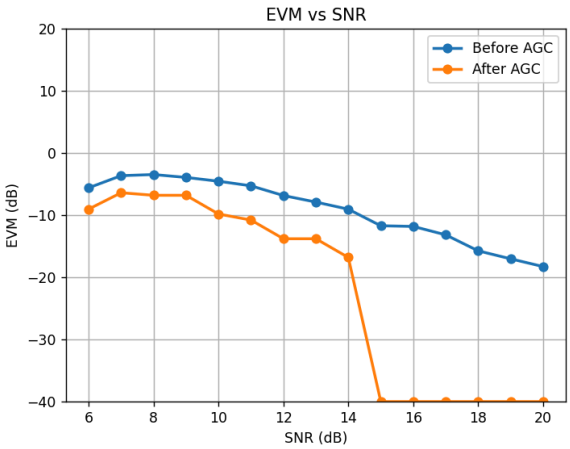


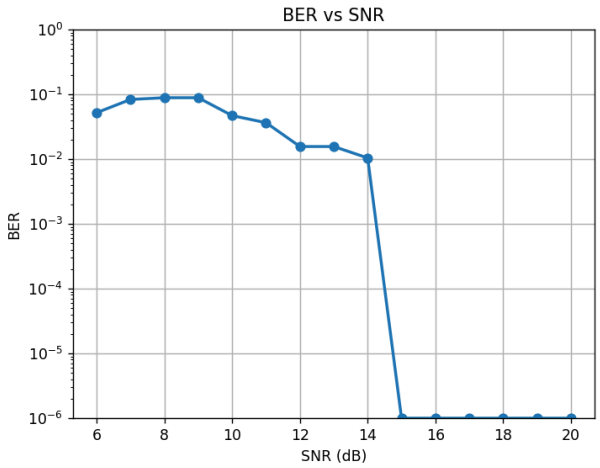


Our Observations :









Before AGC: EVM gradually improves with increasing SNR but stays higher overall.
After AGC: Significant improvement in EVM, especially after 14 dB SNR, where it drops sharply, indicating better signal quality.

- The BER remains relatively high until about 14 dB, then drops steeply to nearly 10−6, suggesting a threshold SNR where the system starts performing optimally.

Observation Values

SNR	EVM	EVM after AGC	BER
6	-5.603592	-9.030900	0.052083
7	-3.650664	-6.398486	0.083333
8	-3.471879	-6.812413	0.088542
9	-3.935880	-6.812413	0.088542
10	-4.546363	-9.822713	0.046875
11	-5.286380	-10.791813	0.036458
12	-6.849701	-13.802112	0.015625
13	-7.901942	-13.802112	0.015625
14	-9.050008	-16.812412	0.010417

15	-11.713938	-40.000000	0.000001

The sharp drop in BER corresponds with the steep EVM improvement after AGC at higher SNRs.
AGC effectively improves signal quality, especially noticeable beyond the 14 dB SNR mark.

Error Vector Magnitude (EVM)
The EVM was calculated for different SNRs to evaluate the receiver's performance.
The results are summarised in the following table:

The EVM decreases as the SNR increases, indicating better signal quality at higher SNRs.

Bit Error Rate (BER)
The BER was calculated for different SNRs to assess the system's error performance.
The results are shown below:

The BER decreases with increasing SNR, demonstrating improved error performance at higher SNRs.

4.4 Execution Time Analysis

The execution time for the Processing System (PS) was measured and compared.
The results are as follows:
Screenshots of the timings are attached at the end of this report.

4.4.1 PS Implementation

Component	**Timings (Optimised) (in s)**	**Timings (Normal) (in s)**
Transmitter	0.002416	0.002843
Receiver	0.498651	0.563215
FFT	0.000058	-
IFFT	0.000063	-
Total	0.499810	0.583123

SNR Values	**(Optimised) in seconds**	**Timings (Normal) in seconds**
6 to 15	0.499810	0.583123

4.4.2 PL Implementation

Component	Timings PL (optimised) (in us)	Timings PL (Normal) (in us)	Timings PS (in us)	Acceleration Factor(PS/PL) (Normal)	Acceleration Factor(PS/PL) (Optimised)
Convolve Rx	**368.204285**	10303.962891	4686.885254	0.45	12.72
Packet Detection	**470.380615**	31466.708984	39199.226562	1.24	83.33
Coarse CFO	**37.492924**	188.202469	289.037231	1.53577	7.70
Fine CFO	**39.593231**	195.845551	290.437561	1.48	7.33
Channel Estimation	**46.034531**	141.779297	180.122833	1.27	3.91
One Tap Equalizer	**34.205105**	135.181976	135.114105	1	3.95

4.4.2.1 Analysis on whole system

Component	Timings (in s)
Without Optimisations	**0.5043**
Optimised without memory Mapped	**0.071346**
Without optimisations with Memory Mapped	**0.491793**
With Optimisations with Memory Mapped	**0.0823**

The Optimised is faster than the Normal implementation, highlighting the benefits of pipelining, loop unrolling and array partitioning.

4.5 Memory Usage Analysis

Stack Size = 223,488 bytes ≈ 218.25 KB (0x36900)

Significance:

The stack stores local variables, function parameters, and return addresses. Proper stack size ensures smooth execution of functions, especially those with deep recursion or large local variables. If the stack is too small, it can lead to stack overflow.

Heap Size = 262,144 bytes = 256 KB (0x40000)

Significance:
The heap is used for dynamic memory allocation (like with malloc or calloc). A larger heap allows more dynamic allocations during runtime. If the heap is too small, dynamic memory allocation may fail, leading to errors like memory corruption or segmentation faults.

4.6 RMSE Values

Since we didn't apply Word Length optimisations, the values were almost the same with a difference of around 0.00001 in every optimisations.

The values below are RMSE values relative to PS and PL, since we used *double* datatype in mid sem submission, but now we have used float.

EVM: 0.00064
EVM after AGC: 6.32×10^{-7}
BER: 0.00000

Formula for RMSE Used:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}$$

5. Optimisations

We have used Pipelining, Array partitioning and Loop unrolling for optimisation. We have implemented memory mapped functionality for convolve_rrc_rx and one tap equalizer ip.

5.1 RRC Convolve IP:

RRC Convolve IP Unoptimised:

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	7.256	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
1032843	1032843	1032843	1032843	none

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	3039	3039	1	-	-	3040	no
- Loop 2	20	20	1	-	-	20	no
- Loop 3	6000	6000	2	-	-	3000	no
- Loop 4	1023780	1023780	339	-	-	3020	no
+ Loop 4.1	336	336	16	-	-	21	no

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	220	-
FIFO	-	-	-	-	-
Instance	-	16	982	2064	-
Memory	16	-	32	11	0
Multiplexer	-	-	-	488	-
Register	-	-	722	-	-
Total	16	16	1736	2783	0
Available	280	220	106400	53200	0
Utilization (%)	5	7	1	5	0

RRC IP Optimization (pipeline and array Partition) 1:

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.345	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
39413	39413	39413	39413	none

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	3039	3039	1	-	-	3040	no
- Loop 2	40	40	2	2	1	20	yes
- Loop 3	3001	3001	3	1	1	3000	yes
- Loop 4	33326	33326	118	11	1	3020	yes

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	22	-	-	-
Expression	-	-	0	645	-
FIFO	-	-	-	-	-
Instance	-	40	3362	7044	-
Memory	40	-	0	0	0
Multiplexer	-	-	-	3646	-
Register	0	-	6825	1152	-
Total	40	62	10187	12487	0
Available	280	220	106400	53200	0
Utilization (%)	14	28	9	23	0

RRC Fully Optimized (loop unrolled and Array Partitioning)(Zc706):

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.588	1.25

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.588	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
3114	3114	3114	3114	none

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- loop	3112	3112	94	1	1	3020	yes

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	3039	3039	1	-	-	3040	no
- Loop 2	20	20	1	1	1	20	yes
- Loop 3	3001	3001	3	1	1	3000	yes
- loop4	3112	3112	94	1	1	3020	yes

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	125	-
FIFO	-	-	-	-	-
Instance	-	420	35889	61024	-
Memory	32	-	0	0	0
Multiplexer	-	-	-	349	-
Register	0	-	9707	3428	-
Total	32	420	45596	64926	0
Available	1090	900	437200	218600	0
Utilization (%)	2	46	10	29	0

The optimization was carried out in three stages: a baseline non-optimized version, a pipelined and array-partitioned version, and a fully optimized version with loop unrolling. Each stage demonstrated clear trade-offs between latency, initiation interval (II), and hardware resource usage.

1. Non-Optimized Design

The initial design had no HLS optimizations applied. As a result, it performed poorly with extremely high latency and fully sequential loop execution.

- **Latency:** ~1,032,843 clock cycles
- **Loop Bottleneck:** Loop 4 dominated execution time
- **Pipelining:** None of the loops were pipelined
- **Key Issue:** Loop-carried dependencies blocked parallelism

---

2. Optimized with Pipelining and Array Partitioning

In this version, loop pipelining and array partitioning were introduced to improve parallelism. This led to a major reduction in latency.

- **Latency:** ~39,413 clock cycles (26× improvement)
- **Pipelining:** Loops 2, 3, and 4 partially pipelined
- **II (Loop 4):** Reduced to 11 but still not ideal
- **Improvement:** Significant speedup with manageable resource increase

---

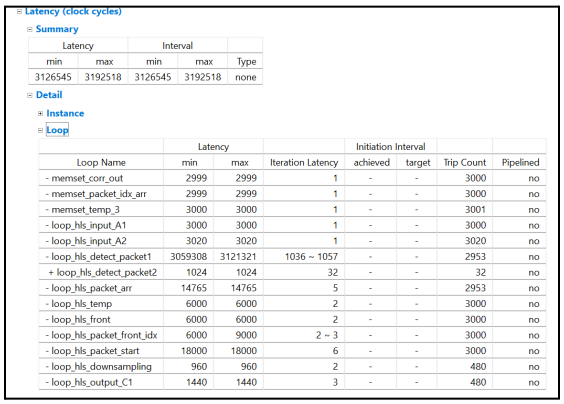
3. Fully Optimized Design (Loop Unrolling + Array Partitioning)

Final optimization stage removed loop-carried dependencies via loop unrolling. This allowed full pipelining and minimal initiation interval.

- **Latency:** ~3,114 clock cycles (330× faster than base)
- **Pipelining:** Loop 4 achieved II = 1
- **Resource Utilization:**
- **Result:** Maximum speedup, best suited for high-performance real-time systems

<u>This ip Worked 12 times faster than implementation on PS</u>

5.2 Packet Detection Unoptimised IP:



Latency (clock cycles)

Summary

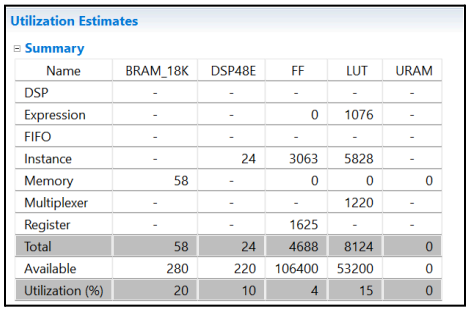
Latency		Interval		
min	max	min	max	Type
3126545	3192518	3126545	3192518	none

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- memset_corr_out	2999	2999	1	-	-	3000	no
- memset_packet_idx_arr	2999	2999	1	-	-	3000	no
- memset_temp_3	3000	3000	1	-	-	3001	no
- loop_hls_input_A1	3000	3000	1	-	-	3000	no
- loop_hls_input_A2	3020	3020	1	-	-	3020	no
- loop_hls_detect_packet1	3059308	3121321	1036 ~ 1057	-	-	2953	no
+ loop_hls_detect_packet2	1024	1024	32	-	-	32	no
- loop_hls_packet_arr	14765	14765	5	-	-	2953	no
- loop_hls_temp	6000	6000	2	-	-	3000	no
- loop_hls_front	6000	6000	2	-	-	3000	no
- loop_hls_packet_front_idx	6000	9000	2 ~ 3	-	-	3000	no
- loop_hls_packet_start	18000	18000	6	-	-	3000	no
- loop_hls_downsampling	960	960	2	-	-	480	no
- loop_hls_output_C1	1440	1440	3	-	-	480	no



Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	1076	-
FIFO	-	-	-	-	-
Instance	-	24	3063	5828	-
Memory	58	-	0	0	0
Multiplexer	-	-	-	1220	-
Register	-	-	1625	-	-
Total	58	24	4688	8124	0
Available	280	220	106400	53200	0
Utilization (%)	20	10	4	15	0

Packet Detection Optimisation 1 IP: (Pipelining)

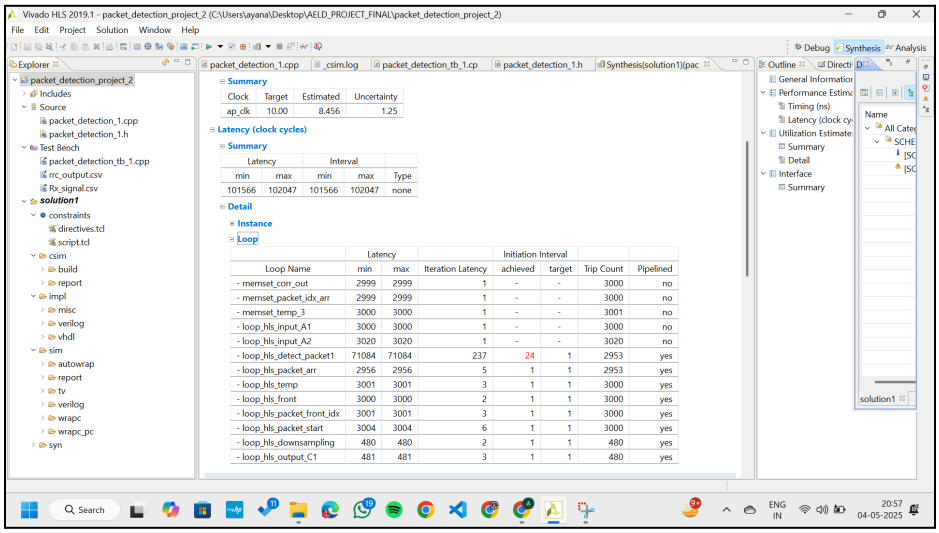
When Loop_hls_detect_packet1 is pipelined, loops in the hierarchy below are automatically unrolled (i.e loop_hls_detect_packet_2) reducing the latency significantly.

```
    float power;
    float corr_mag;
    complex_t sum_corr;
    loop_hls_detect_packet1: for (int n = 0; n < len_corr; n++) {
        #pragma HLS pipeline
        sum_corr = complex_t(0.0, 0.0);
        sum_peak = 0.0;

        loop_hls_detect_packet2: for (int k = 0; k < window_length; k++) {
//          #pragma HLS UNROLL factor=4
            corr_val = Rx_signal[n + k] * Rx_signal[n + k + delay_param];
            sum_corr += corr_val;

            power = sqrt(
                Rx_signal[n + k + delay_param].real() * Rx_signal[n + k + delay_param].real() +
                Rx_signal[n + k + delay_param].imag() * Rx_signal[n + k + delay_param].imag()
            );

            sum_peak += power * power;
        }

        corr_mag = sum_corr.real() * sum_corr.real() + sum_corr.imag() * sum_corr.imag();
        corr_out[n] = (sum_peak > 0) ? (corr_mag / (sum_peak * sum_peak)) : 0.0;
    }
```

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	1599	-
FIFO	-	-	-	-	-
Instance	-	61	7137	11882	-
Memory	58	-	0	0	0
Multiplexer	-	-	-	3959	-
Register	0	-	15089	2336	-
Total	58	61	22226	19776	0
Available	280	220	106400	53200	0
Utilization (%)	20	27	20	37	0



Packet Detection Optimisation 2 IP:(Array Partitioning)

```
complex_t Rx_signal[NP_PACKETS_CAPTURE];
#pragma HLS array_partition variable=Rx_signal cyclic factor=2
complex_t Rx_filter_signal[NP_PACKETS_CAPTURE + FILTER_SIZE - 1];
complex_t Rx_frame[FRAME_LENGTH] = {0};
```

This partitions Rx_signal cyclically with factor 2 meaning:

- Rx_signal[0], Rx_signal[2], Rx_signal[4] go to one memory bank
- Rx_signal[1], Rx_signal[3], Rx_signal[5] go to another

Two different elements of Rx_signal can be accessed in parallel without a memory conflict.

This enables parallel fetching of the required elements like Rx_signal[n + k] and Rx_signal[n + k + delay_param] within the same cycle, eliminating stalls. As a result, the pipeline operates more efficiently with reduced latency and a better Initiation Interval.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.456	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
66144	66625	66144	66625	none

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	1903	-
FIFO	-	-	-	-	-
Instance	-	99	10635	17831	-
Memory	58	-	0	0	0
Multiplexer	-	-	-	4880	-
Register	0	-	19308	2912	-
Total	58	99	29943	27526	0
Available	280	220	106400	53200	0
Utilization (%)	20	45	28	51	0

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- memset_corr_out	2999	2999	1	-	-	3000	no
- memset_packet_idx_arr	2999	2999	1	-	-	3000	no
- memset_temp_3	3000	3000	1	-	-	3001	no
- loop_hls_input_A1	3000	3000	1	-	-	3000	no
- loop_hls_input_A2	3020	3020	1	-	-	3020	no
- loop_hls_detect_packet1	35662	35662	239	12	1	2953	yes
- loop_hls_packet_arr	2956	2956	5	1	1	2953	yes
- loop_hls_temp	3001	3001	3	1	1	3000	yes
- loop_hls_front	3000	3000	2	1	1	3000	yes
- loop_hls_packet_front_idx	3001	3001	3	1	1	3000	yes
- loop_hls_packet_start	3004	3004	6	1	1	3000	yes
- loop_hls_downsampling	480	480	2	1	1	480	yes
- loop_hls_output_C1	481	481	3	1	1	480	yes

Packet Detection IP:(Zc706 Board) -

Since the upper bounds for loop_hls_temp, loop_hls_front, loop_packet_front_idx, loop_hls_packet_start, loop_hls_downsampling are not constant. So to calculate the latency, I have set the upper limit to MAX_PACKETS (3000). To minimise the computation break the loop

when the limit exceeds the original bound. That is why min latency has decreased for these loops.

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.239	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
54103	66580	54103	66580	none

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	1565	-
FIFO	-	-	-	-	-
Instance	-	99	10344	17976	-
Memory	58	-	0	0	0
Multiplexer	-	-	-	602	-
Register	0	-	17959	2816	-
Total	58	99	28303	22959	0
Available	1090	900	437200	218600	0
Utilization (%)	5	11	6	10	0

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- memset_corr_out	2999	2999	1	-	-	3000	no
- memset_packet_idx_arr	2999	2999	1	-	-	3000	no
- memset_temp_3	3000	3000	1	-	-	3001	no
- loop_hls_input_A1	3000	3000	1	-	-	3000	no
- loop_hls_input_A2	3020	3020	1	-	-	3020	no
- loop_hls_detect_packet1	35619	35619	196	12	1	2953	yes
- loop_hls_packet_arr	2955	2955	4	1	1	2953	yes
- loop_hls_temp	2	3001	3	1	1	1 ~ 3000	yes
- loop_hls_front	1	3000	2	1	1	1 ~ 3000	yes
- loop_hls_packet_front_idx	2	3001	3	1	1	1 ~ 3000	yes
- loop_hls_packet_start	4	3003	5	1	1	1 ~ 3000	yes
- loop_hls_downsampling	1	480	2	1	1	1 ~ 480	yes
- loop_hls_output_C1	481	481	3	1	1	480	yes

5.3 Coarse CFO

Normal

Latencies (loop wise)

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.665	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
19470	21535	19470	21535	none

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	479	479	1	-	-	480	no
- Loop 2	479	479	1	-	-	480	no
- loop_input_A1	480	480	1	-	-	480	no
- loop_ComplexConj	256	256	16	-	-	16	no
- loop_Coarse_Freq_Offset	16320	18240	34 ~ 38	-	-	480	no
- loop_output_C1	1440	1440	3	-	-	480	no

Resource Utilization

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	267	-
FIFO	-	-	-	-	-
Instance	4	34	5255	10279	-
Memory	6	-	0	0	0
Multiplexer	-	-	-	761	-
Register	-	-	913	-	-
Total	10	34	6168	11307	0
Available	280	220	106400	53200	0
Utilization (%)	3	15	5	21	0

Optimised

1. Latencies (loop wise)

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
3952	4051	3952	4051	none

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	479	479	1	-	-	480	no
- Loop 2	479	479	1	-	-	480	no
- loop_input_A1	480	480	1	1	1	480	yes
- loop_ComplexConj	76	76	17	4	1	16	yes
- loop_Coarse_Freq_Offset	1939	1939	24	4	1	480	yes
- loop_output_C1	481	481	3	1	1	480	yes

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.151	1.25

2. Resource Utilisation

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	339	-
FIFO	-	-	-	-	-
Instance	4	23	4418	7729	-
Memory	10	-	0	0	0
Multiplexer	-	-	-	155	-
Register	0	-	1328	128	-
Total	14	23	5746	8351	0
Available	1090	900	437200	218600	0
Utilization (%)	1	2	1	3	0

Description of Optimisations:

1. **Pipelining:**

Used #pragma HLS PIPELINE, to implement pipelining in all the for loops.
This decreased the Initiation Interval (II) to 1, wherever possible.
Pipelining helps in significantly improving throughput by overlapping operations across different loop iterations, leading to faster execution.

2. **Array Partitioning:**

Used #pragma HLS ARRAY_PARTITION variable=input_A cyclic factor=2 dim=1
This partitioned the array to two arrays which means, if A = [1.2.3.4.5.6]. The array is now split into [1,3,5] and [2,4,5] using BRAMs, and now it can help access the array in successive iterations so that when one iteration is in progress it uses first array and second iteration uses the second array since every successive iteration is taking odd and even index respectively.

5.4 Fine CFO

Normal

1. **Latencies (loopwise)**

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.665	1.25

Summary

Latency		Interval		
min	max	min	max	Type
20238	22303	20238	22303	none

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	479	479	1	-	-	480	no
- Loop 2	479	479	1	-	-	480	no
- loop_input_A1	480	480	1	-	-	480	no
- loop_ComplexConj	1024	1024	16	-	-	64	no
- loop_fine_Freq_Offset	16320	18240	34 ~ 38	-	-	480	no
- loop_output_C1	1440	1440	3	-	-	480	no

2. **Resource Utilization**

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	260	-
FIFO	-	-	-	-	-
Instance	4	34	5255	10279	-
Memory	6	-	0	0	0
Multiplexer	-	-	-	752	-
Register	-	-	885	-	-
Total	10	34	6140	11291	0
Available	280	220	106400	53200	0
Utilization (%)	3	15	5	21	0

Optimised

1. Latencies (loop wise)

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.151	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
4144	4243	4144	4243	none

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	479	479	1	-	-	480	no
- Loop 2	479	479	1	-	-	480	no
- loop_input_A1	480	480	1	1	1	480	yes
- loop_ComplexConj	268	268	17	4	1	64	yes
- loop_fine_Freq_Offset	1939	1939	24	4	1	480	yes
- loop_output_C1	481	481	3	1	1	480	yes

2. **Resource Utilisation**

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	340	-
FIFO	-	-	-	-	-
Instance	4	23	4418	7729	-
Memory	10	-	0	0	0
Multiplexer	-	-	-	149	-
Register	0	-	1364	128	-
Total	14	23	5782	8346	0
Available	1090	900	437200	218600	0
Utilization (%)	1	2	1	3	0

Optimisations Description:

1. **Pipelining:**

Used #pragma HLS PIPELINE, to implement pipelining in all the for loops.
This decreased the Initiation Interval (II) to 1, wherever possible.
Pipelining helps in significantly improving throughput by overlapping operations across different loop iterations, leading to faster execution.

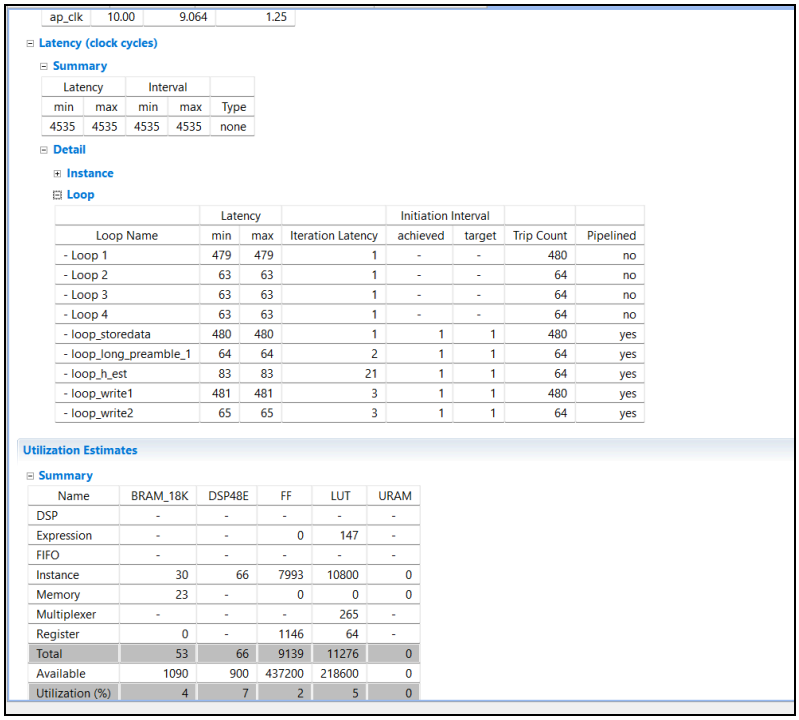
2. **Array Partitioning:**

Used #pragma HLS ARRAY_PARTITION variable=input_A cyclic factor=2 dim=1
This partitioned the array to two arrays which means, if A = [1.2.3.4.5.6]. The array is now split into [1,3,5] and [2,4,5] using BRAMs, and now it can help access the array in successive iterations so that when one iteration is in progress it uses first array and second iteration uses the second array since every successive iteration is taking odd and even index respectively.

5.5 Channel Estimation

Optimised

1. **Latencies(loop wise) and resource utilization**



ap_clk	10.00	9.064	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
4535	4535	4535	4535	none

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	479	479	1	-	-	480	no
- Loop 2	63	63	1	-	-	64	no
- Loop 3	63	63	1	-	-	64	no
- Loop 4	63	63	1	-	-	64	no
- loop_storedata	480	480	1	1	1	480	yes
- loop_long_preamble_1	64	64	2	1	1	64	yes
- loop_h_est	83	83	21	1	1	64	yes
- loop_write1	481	481	3	1	1	480	yes
- loop_write2	65	65	3	1	1	64	yes

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	147	-
FIFO	-	-	-	-	-
Instance	30	66	7993	10800	0
Memory	23	-	0	0	0
Multiplexer	-	-	-	265	-
Register	0	-	1146	64	-
Total	53	66	9139	11276	0
Available	1090	900	437200	218600	0
Utilization (%)	4	7	2	5	0

In the process of optimizing the channel estimation IP, pipelining was applied to both the FFT function and the top-level function. This alone was sufficient to achieve a latency of 1, while also resulting in minimal resource utilization. Additional attempts to optimize using array partitioning led to only marginal improvements in latency, typically a reduction of just 10–15 cycles. However, this came at the cost of a significant increase in resource utilization, making it less favorable in terms of overall design efficiency. Hence, pipelining proved to be the most effective optimization for balancing both performance and resource usage in the implementation.

Normal

1. **Latencies(loop wise)**

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	479	479	1	-	-	480	no
- Loop 2	63	63	1	-	-	64	no
- Loop 3	63	63	1	-	-	64	no
- Loop 4	63	63	1	-	-	64	no
- Loop 5	480	480	1	-	-	480	no
- Loop 6	128	128	2	-	-	64	no
- Loop 7	1664	1664	26	-	-	64	no
- Loop 8	1440	1440	3	-	-	480	no
- Loop 9	192	192	3	-	-	64	no

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
14323	14323	14323	14323	none

2. **Resource utilization**

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	301	-
FIFO	-	-	-	-	-
Instance	22	36	3647	7551	0
Memory	23	-	0	0	0
Multiplexer	-	-	-	1416	-
Register	-	-	775	-	-
Total	45	36	4422	9268	0
Available	280	220	106400	53200	0
Utilization (%)	16	16	4	17	0

5.6 One Tap Equaliser Optimised

1. **Latencies(Loop wise)**

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.790	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
3726	3726	3726	3726	none

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	79	79	1	-	-	80	no
- Loop 2	79	79	1	-	-	80	no
- Loop 3	63	63	1	-	-	64	no
- Loop 4	63	63	1	-	-	64	no
- Loop 5	479	479	1	-	-	480	no
- Loop 6	63	63	1	-	-	64	no
- Loop 7	63	63	1	-	-	64	no
- Loop 8	63	63	1	-	-	64	no
- loop_firstINput	480	480	1	1	1	480	yes
- loop_secondInput	64	64	1	1	1	64	yes
- loop_RX_PayLoad1_time	80	80	2	1	1	80	yes
- loop_RX_PayLoad1_no_CP	64	64	2	1	1	64	yes
- floop_Rx_1_eq	84	84	22	1	1	64	yes
- loop_output2	65	65	3	1	1	64	yes
- loop_output3	65	65	3	1	1	64	yes
- loop_output4	65	65	3	1	1	64	yes

2. Resource utilization

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	245	-
FIFO	-	-	-	-	-
Instance	30	86	15996	17557	0
Memory	30	-	0	0	0
Multiplexer	-	-	-	349	-
Register	0	-	1409	64	-
Total	60	86	17405	18215	0
Available	1090	900	437200	218600	0
Utilization (%)	5	9	3	8	0

In the one-tap equalizer IP, pipelining was applied to both the supporting functions and the top-level function. This optimization alone was sufficient to achieve a latency of 1 while maintaining minimal resource utilization.

Normal

1. Latencies(Loop wise)

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
13910	13910	13910	13910	none

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	79	79	1	-	-	80	no
- Loop 2	79	79	1	-	-	80	no
- Loop 3	63	63	1	-	-	64	no
- Loop 4	63	63	1	-	-	64	no
- Loop 5	479	479	1	-	-	480	no
- Loop 6	63	63	1	-	-	64	no
- Loop 7	63	63	1	-	-	64	no
- Loop 8	63	63	1	-	-	64	no
- loop_first	480	480	1	-	-	480	no
- loop_second	64	64	1	-	-	64	no
- Loop 11	160	160	2	-	-	80	no
- Loop 12	128	128	2	-	-	64	no
- Loop 13	1792	1792	28	-	-	64	no
- loop_output2	192	192	3	-	-	64	no
- loop_output3	192	192	3	-	-	64	no
- loop_output4	192	192	3	-	-	64	no

Utilization Estimates

2. Resource utilization

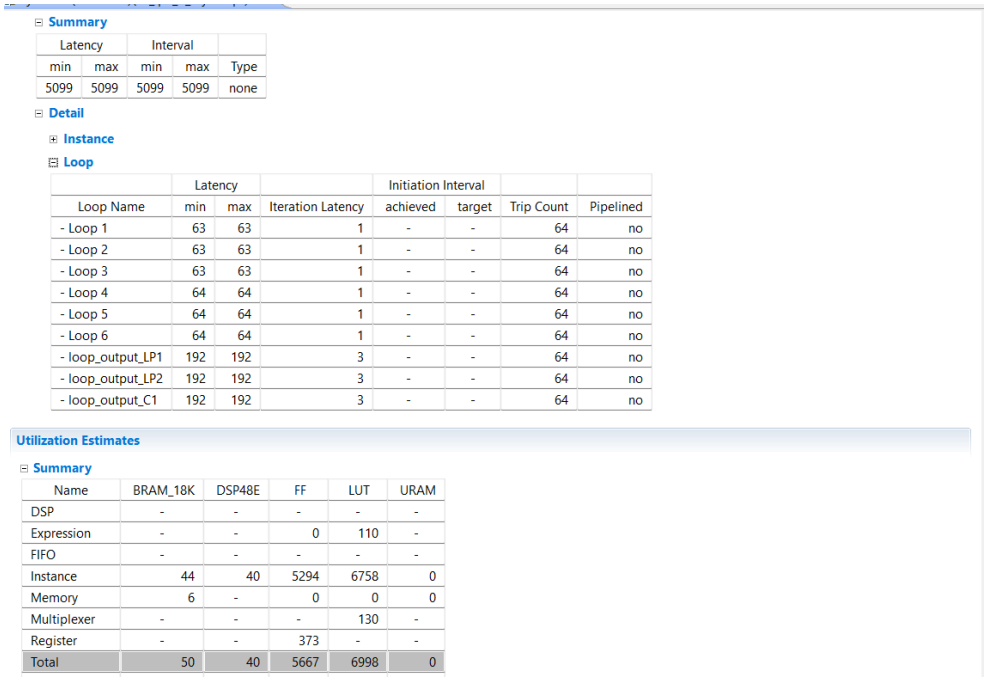
Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	520	-
FIFO	-	-	-	-	-
Instance	22	60	8164	14623	0
Memory	30	-	0	0	0
Multiplexer	-	-	-	1686	-
Register	-	-	1292	-	-
Total	52	60	9456	16829	0
Available	280	220	106400	53200	0
Utilization (%)	18	27	8	31	0

FFT-

Hardware utilisation:(unoptimised)



Summary

Latency		Interval		
min	max	min	max	Type
5099	5099	5099	5099	none

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	63	63	1	-	-	64	no
- Loop 2	63	63	1	-	-	64	no
- Loop 3	63	63	1	-	-	64	no
- Loop 4	64	64	1	-	-	64	no
- Loop 5	64	64	1	-	-	64	no
- Loop 6	64	64	1	-	-	64	no
- loop_output_LP1	192	192	3	-	-	64	no
- loop_output_LP2	192	192	3	-	-	64	no
- loop_output_C1	192	192	3	-	-	64	no

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	110	-
FIFO	-	-	-	-	-
Instance	44	40	5294	6758	0
Memory	6	-	0	0	0
Multiplexer	-	-	-	130	-
Register	-	-	373	-	-
Total	50	40	5667	6998	0

Pipeline Optimisation

Summary

Latency		Interval		
min	max	min	max	Type
1464	1464	1464	1464	none

Detail

Instance

Loop

	Latency			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- Loop 1	63	63	1	-	-	64	no
- Loop 2	63	63	1	-	-	64	no
- Loop 3	63	63	1	-	-	64	no
- Loop 4	64	64	1	1	1	64	yes
- Loop 5	64	64	1	1	1	64	yes
- Loop 6	64	64	1	1	1	64	yes
- loop_output_LP1	65	65	3	1	1	64	yes
- loop_output_LP2	65	65	3	1	1	64	yes
- loop_output_C1	65	65	3	1	1	64	yes

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	131	-
FIFO	-	-	-	-	-
Instance	60	92	22610	16688	0
Memory	6	-	0	0	0
Multiplexer	-	-	-	148	-
Register	-	-	367	-	-
Total	66	92	22977	16967	0

FFT IP Note

This is not a standard FFT IP core. It has been specially designed and customized to meet the specific requirements of this project. Unlike traditional FFT modules that handle one input at a time, this IP performs two FFT computations on two different input data sets in a single execution cycle.

- Custom Design: Tailored specifically for the project's signal flow
- Dual Operation: Executes FFT twice per run
- Input Handling: Processes two independent input arrays
- Efficiency: Reduces overhead by combining operations in one IP block
- Integration: Designed to fit directly into the project's data pipeline

6. Conclusion

The end semester deliverables have been successfully completed, and the OFDM receiver with synchronisation has been implemented on the Zedboard platform. The results demonstrate the system's effectiveness in EVM, BER, and execution time. Future work will focus on optimising and implementing additional IP blocks for QPSK modulation, packet detection, and CFO estimation.

7. References

- Matlab Code by Jai Mangal, PhD (ECE), IIITD
- AELD Lab PDFs

8. Individual Contributions:

Ayana:

1. Packet Detection (Delay & Correlate) (Perform Packet Detection, Packet Selection, Identify Packet Start, Downsampling)

Bharti Bansal:

1. Channel Estimation
2. One Tap Equalizer

Harsh Nangia:

1. Coarse CFO
2. Fine CFO
3. Memory Mapped IPs (converted convolve rrx and one tap equalizer ip to memory mapped)
4. Vivado Design and making the previous SDK code compatible with the new design.

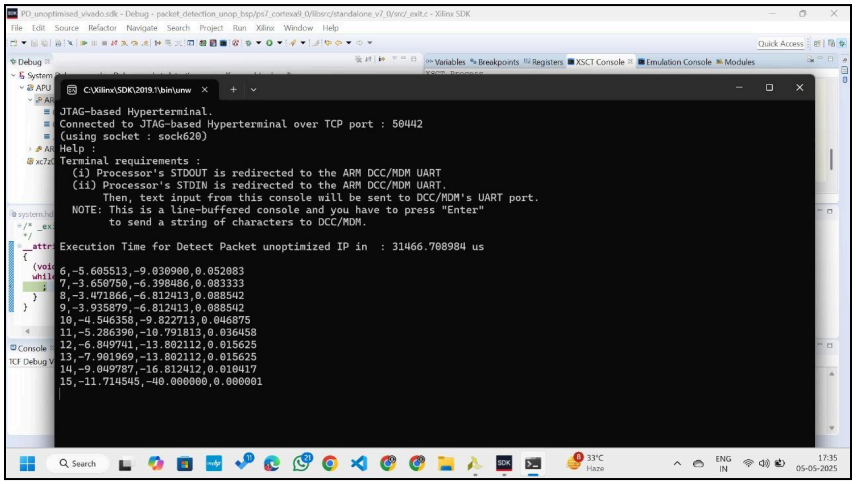
Kunal Sharma:

1. Convolve RRX IP.
2. Interrupts
3. Reduced hardware utilization of channel estimation and one tap equalizer for zedboard
4. FFT IP.

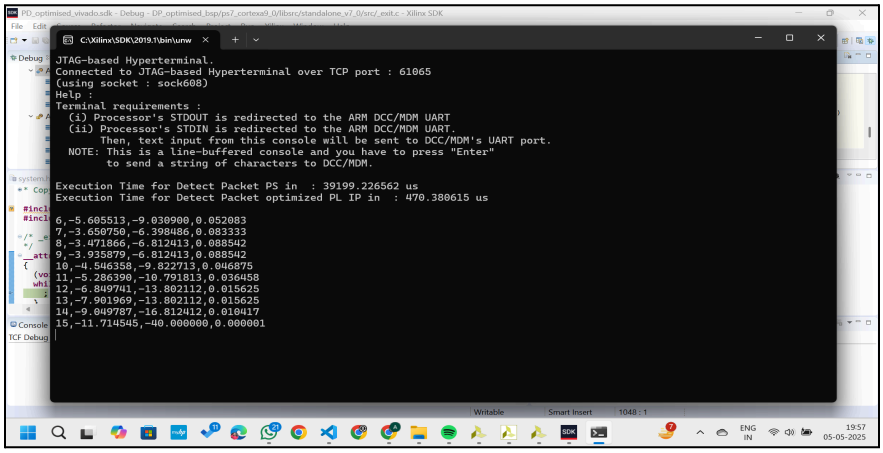
(changes done in 4 Ips to achieve this)

9. Screenshots

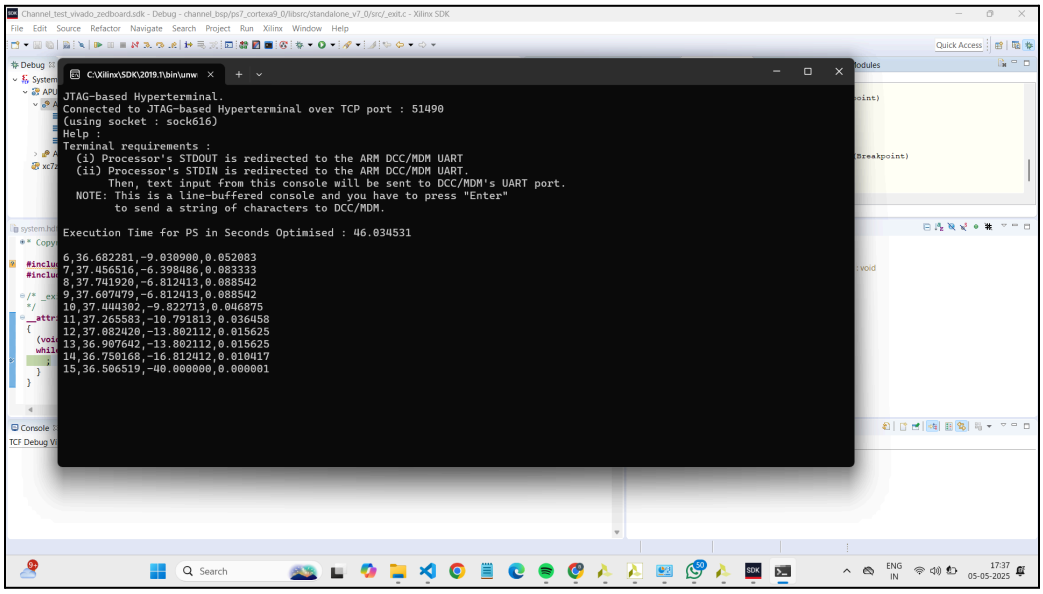
For packet detection



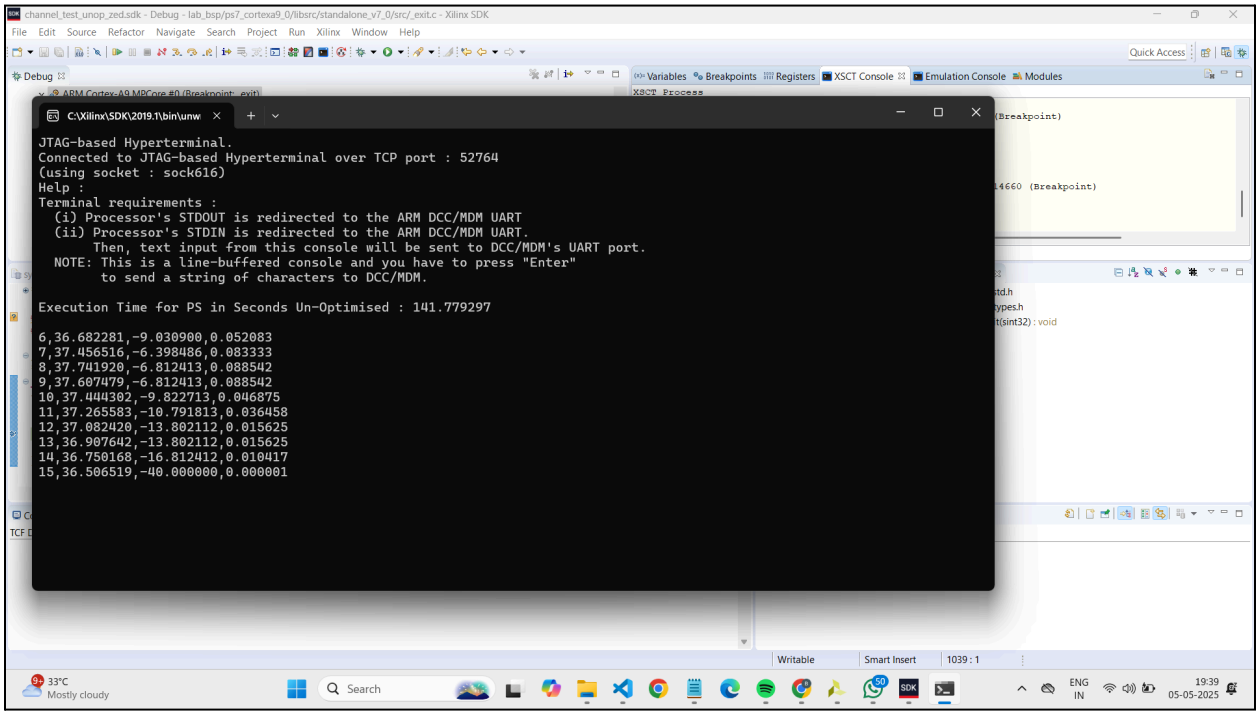
Packet Detection after array partitioning



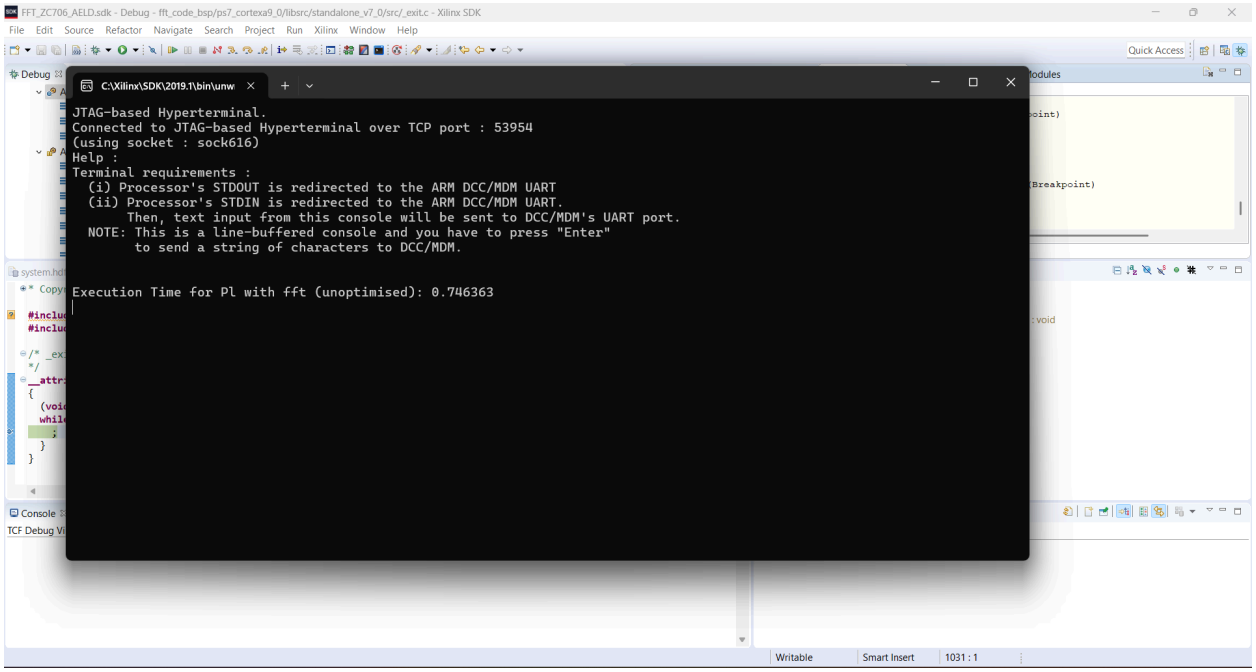
Channel Estimation PL (optimised)



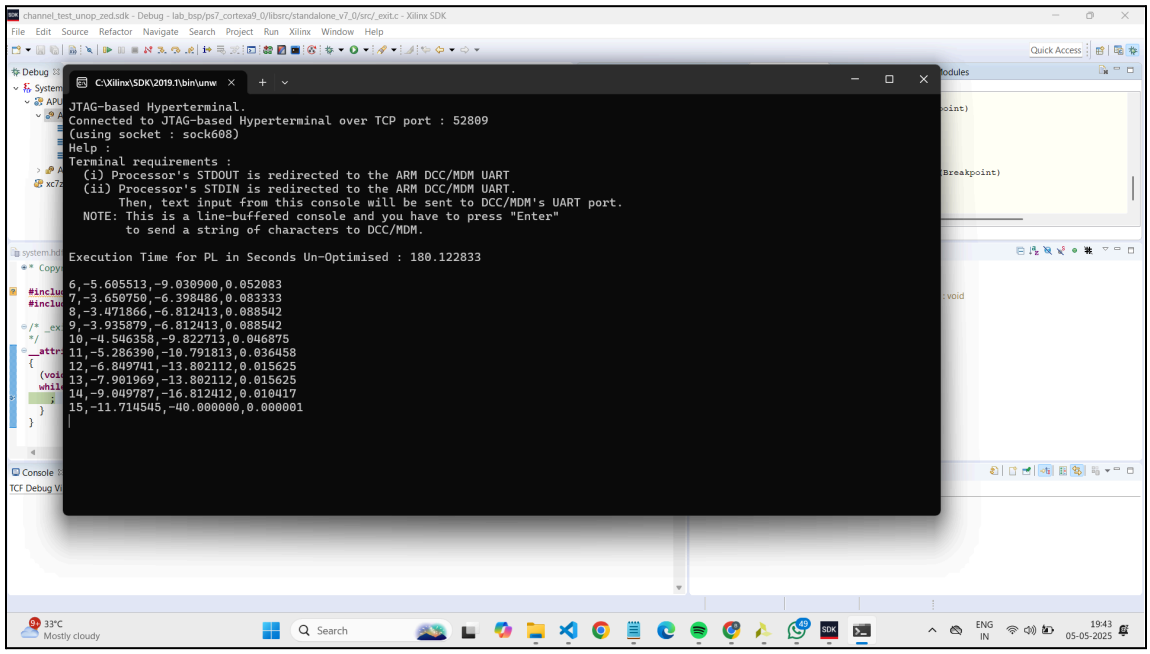
Channel Estimation PL (Un-optimised)



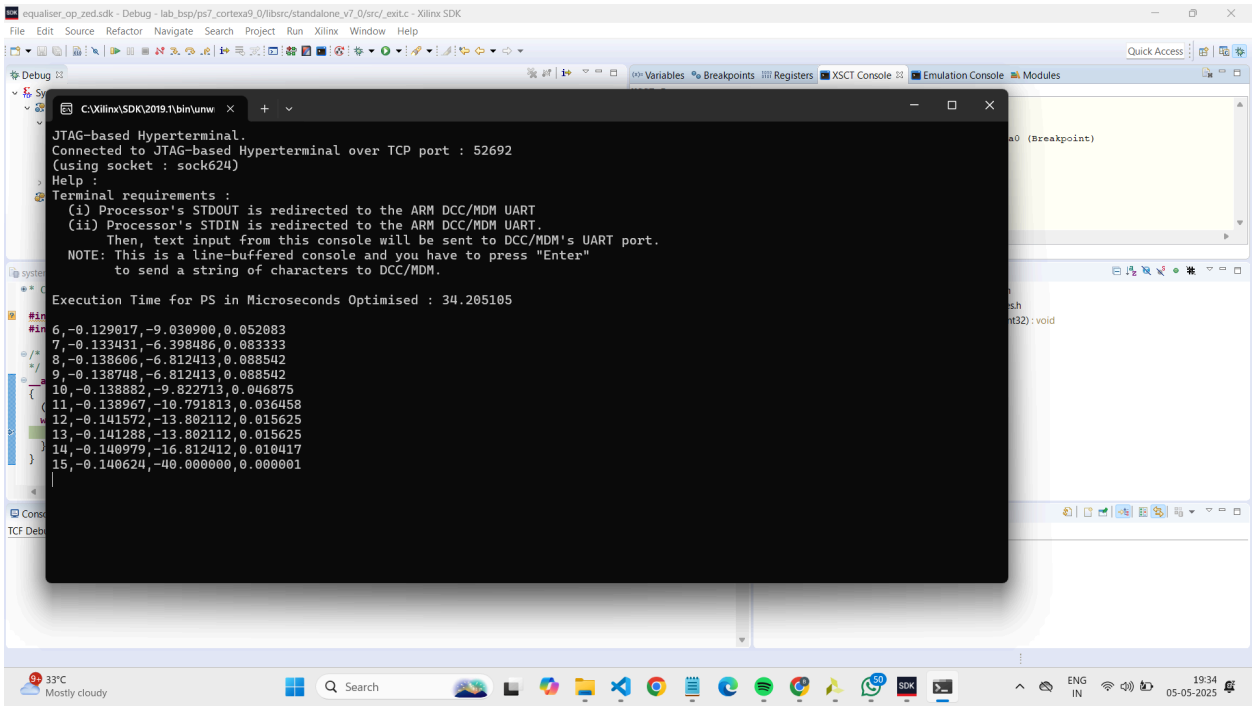
With fft ip



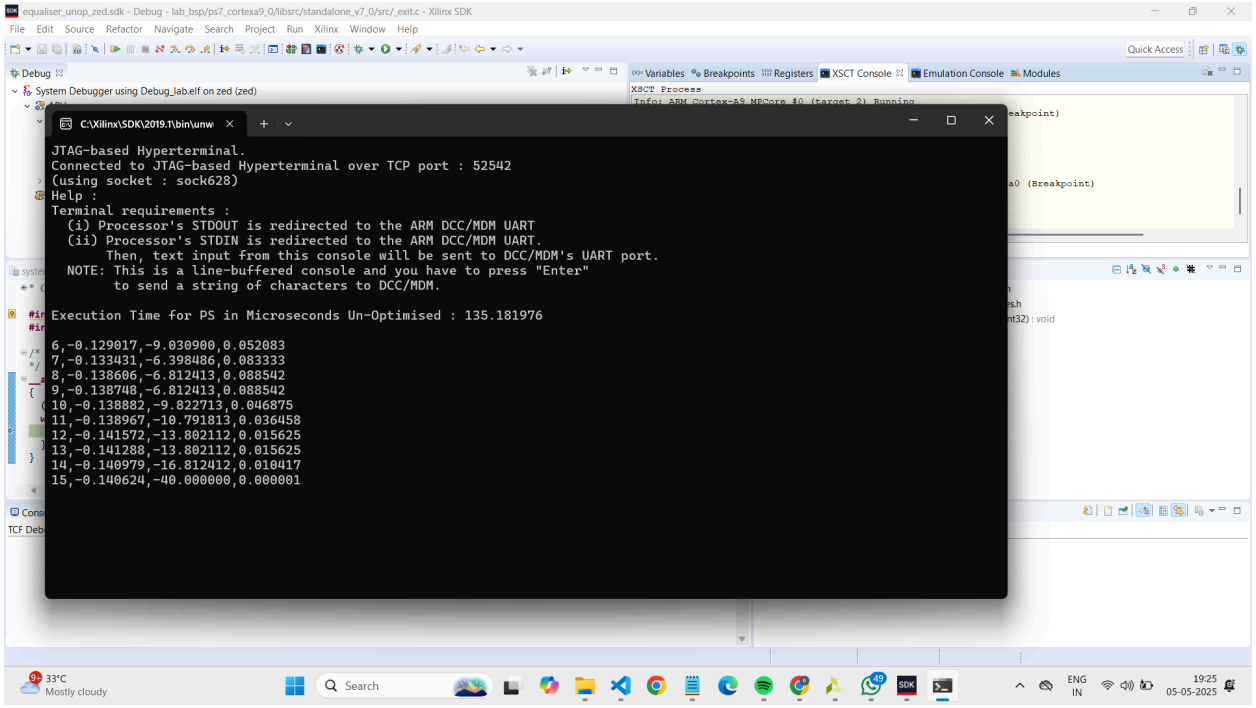
Channel Estimation PS:

1. 

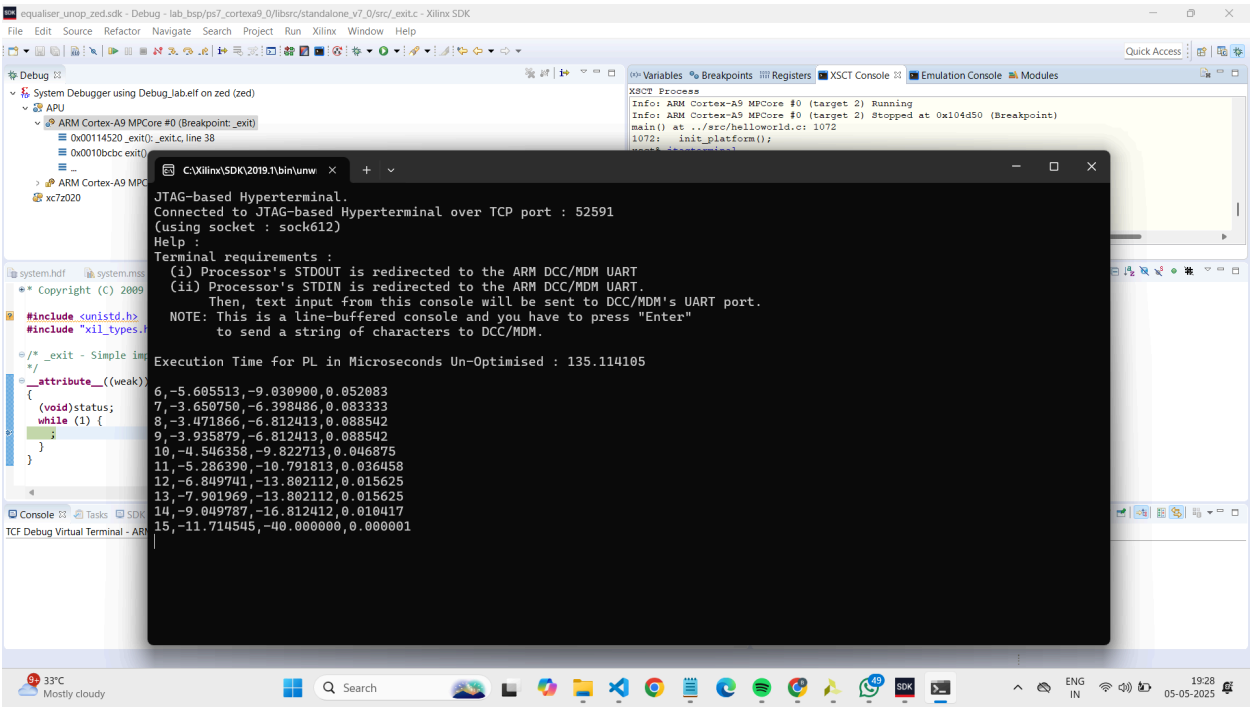
One Tap PL Optimised:



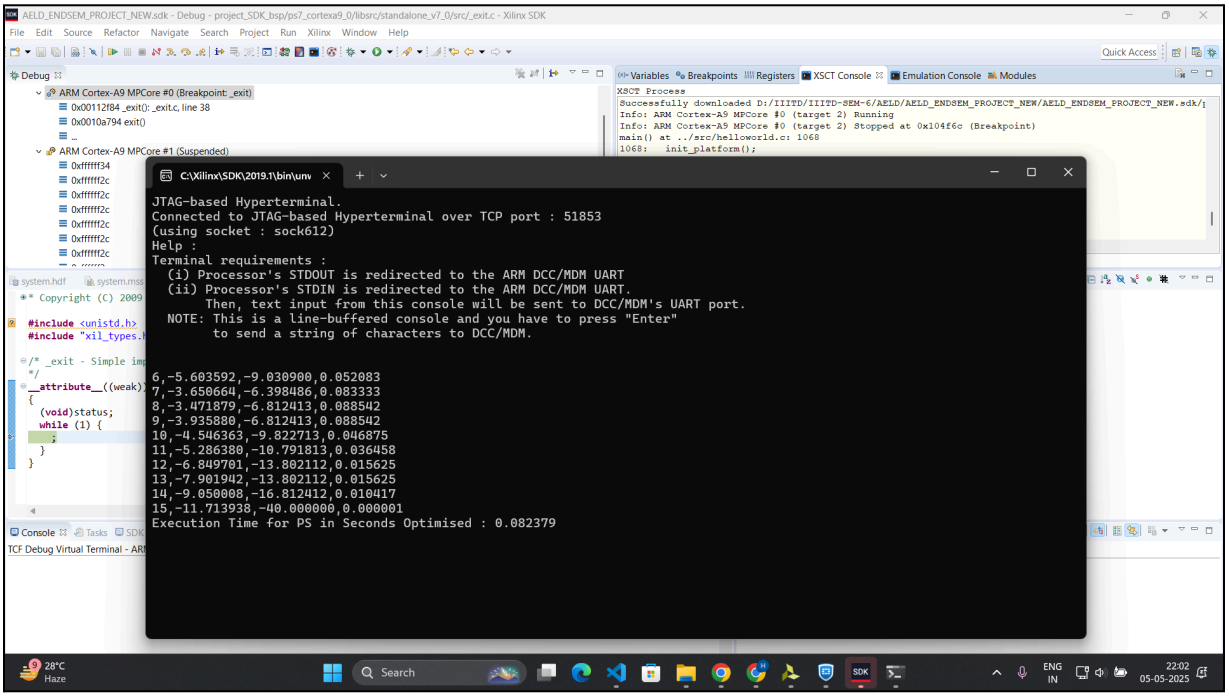
One Tap Equalizer PL Unoptimised:



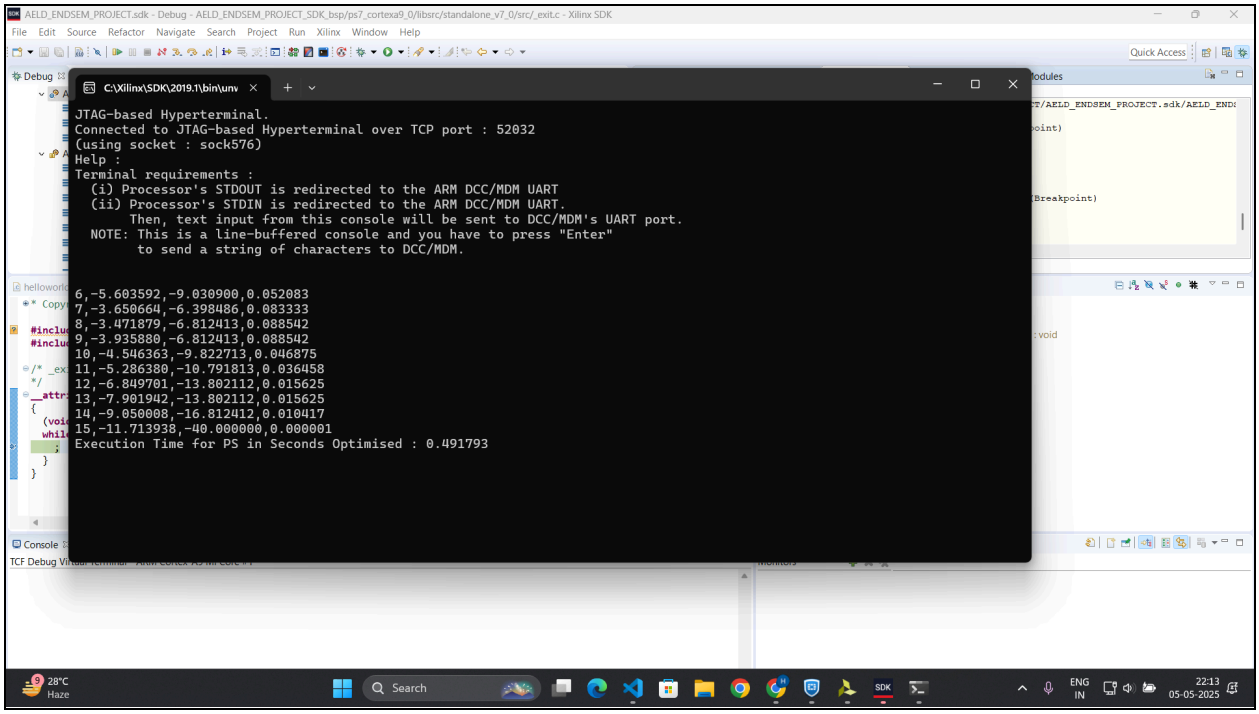
One Tap Equalizer PS:



Using memory mapped IO on whole system



Using memory mapped on whole system


C:\Xilinx\SDK\2019.1\bin\unv
JTAG-based Hyperterminal.
Connected to JTAG-based Hyperterminal over TCP port : 52032
(using socket : sock576)
Help :
Terminal requirements :
(i) Processor's STDOUT is redirected to the ARM DCC/MDM UART
(ii) Processor's STDIN is redirected to the ARM DCC/MDM UART.
Then, text input from this console will be sent to DCC/MDM's UART port.
NOTE: This is a line-buffered console and you have to press "Enter"
to send a string of characters to DCC/MDM.
6,-5.603592,-9.030900,0.052083
7,-3.650664,-6.398486,0.083333
8,-3.471879,-6.812413,0.088542
9,-3.935880,-6.812413,0.088542
10,-4.546363,-9.822713,0.046875
11,-5.286380,-10.791813,0.036458
12,-6.849701,-13.802112,0.015625
13,-7.901942,-13.802112,0.015625
14,-9.050008,-16.812412,0.010417
15,-11.713938,-40.000000,0.000001
Execution Time for PS in Seconds Optimised : 0.491793