ELD PROJECT
FFT IP USING HLS

Aditya Gautam
2023043

Objectives

1. Introduction to the project
2. Block design and its explanation
3. Results along with the explanation
4. Comparison of the different results with each other
5. Conclusion, along with reference if any.

Introduction

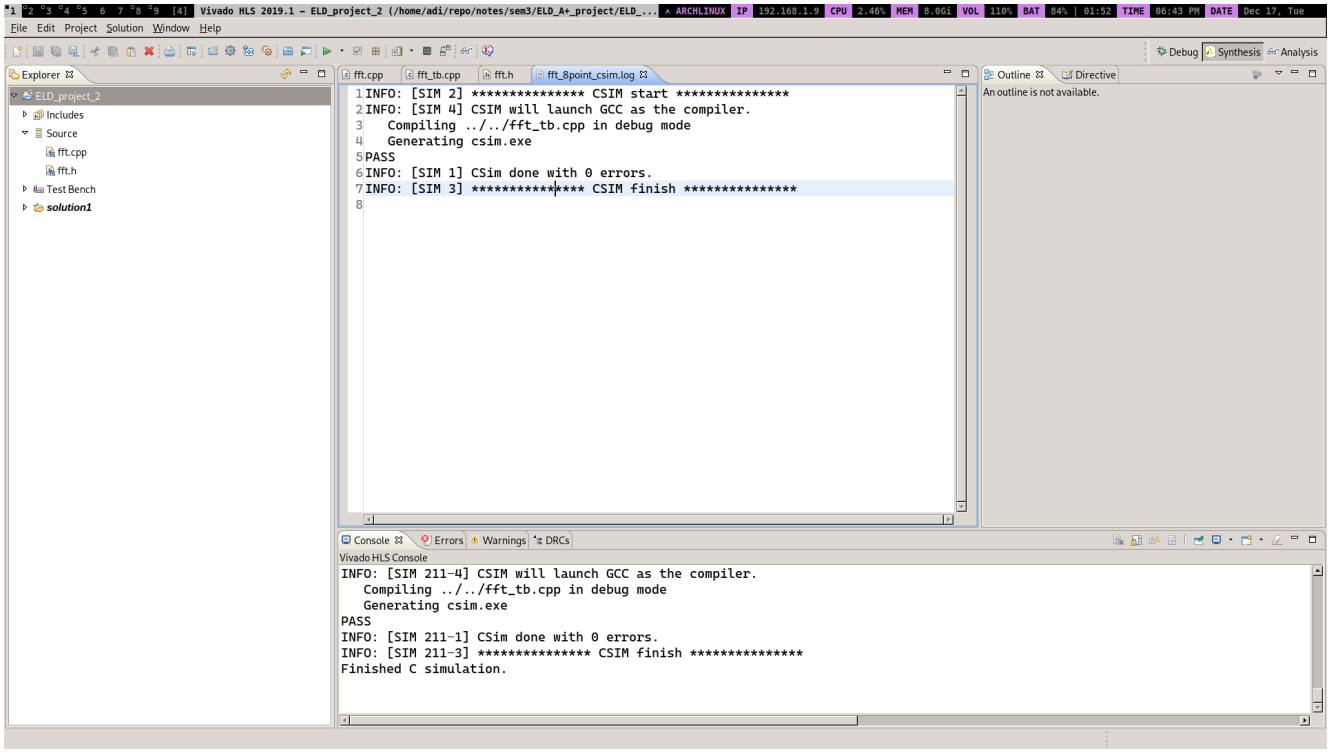
Understanding Vivado High Level Synthesis (HLS) Tool which converts C/C++ code into Verilog. Create hardware IP for Fast Fourier Transform Algorithm using HLS Tool.

Important Note

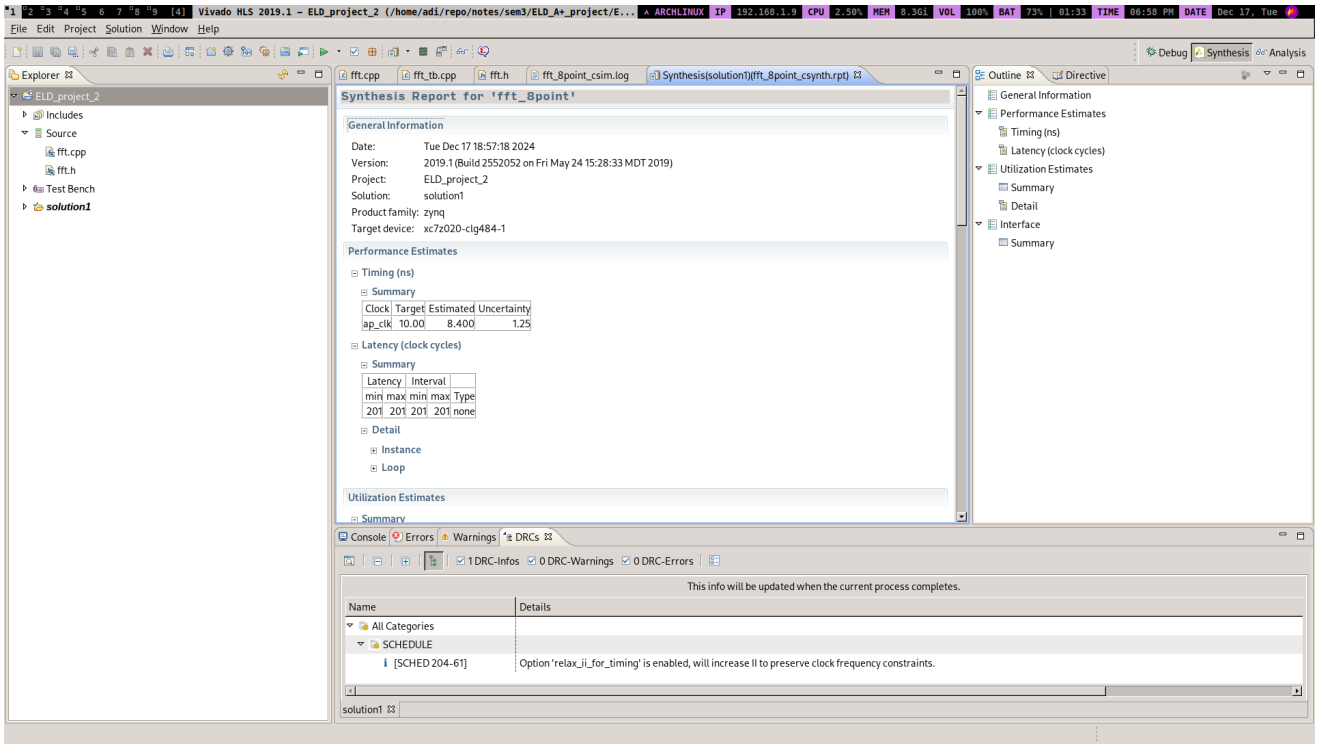
I am running Vivado version 2019.1 on Arch Linux and after trying to run C simulation in Vivado HLS, I was getting weird linking errors. After immense research, I was able to reach to the conclusion that the `ld` binary that the Vivado HLS was using to link the object files was outdated due to the older version of Vivado and hence I had to copy the latest binary of `ld` from the Arch Repositories to the directory where the Vivado HLS's `ld` binary was placed. Conclusion: If you are in a linux environment and facing a similar issue, just copy the latest `ld` binary from your clang packages and replace it with the older `ld` binary.

Development Log

- After fixing the issue with the `ld` binary, I was able to make the C simulation and test bench work successfully.

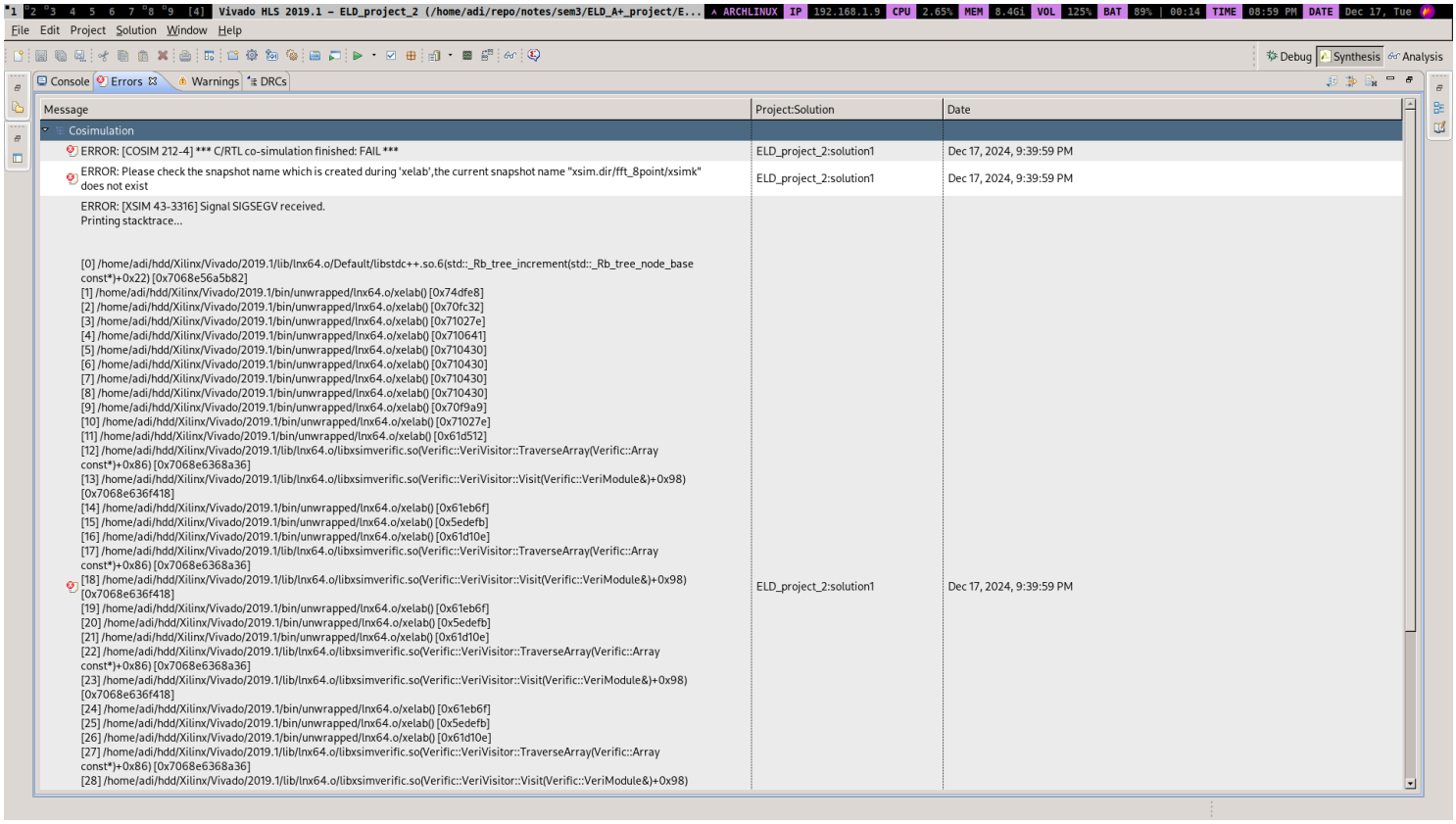


- C Synthesis worked successfully as well



- I am not sure if I can hard code input in the main c++ file or if I am supposed to add AXI Stream interface to take input and pass the output. But before thinking about that, I am not able to make the C/RTL Cosimulation pass (the step where the Verilog code is verified)

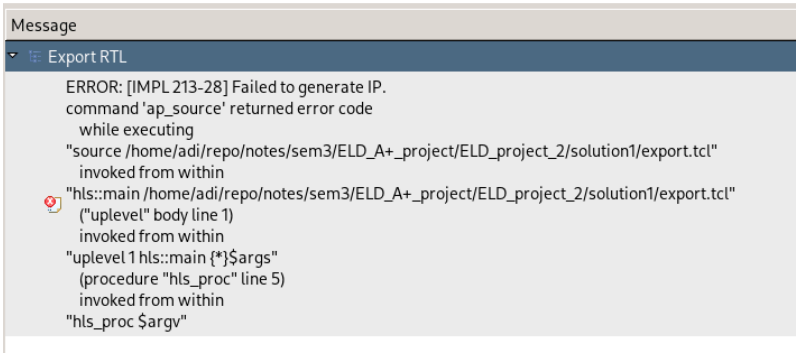
- I am getting this error >



- I have tried every possible solution from the forums besides upgrading Vivado from version 2019.1 with no working outcome.
- Since the project objective does not explicitly state that we have to make the FFT IP compatible with AXI Stream Interface, I will just leave the IP as is and conclude that I have gained crucial knowledge through this project and that I have understood to create my very own IP even with AXI Stream compatibility even if I was unable to make one in this attempt.

• Important Note V2

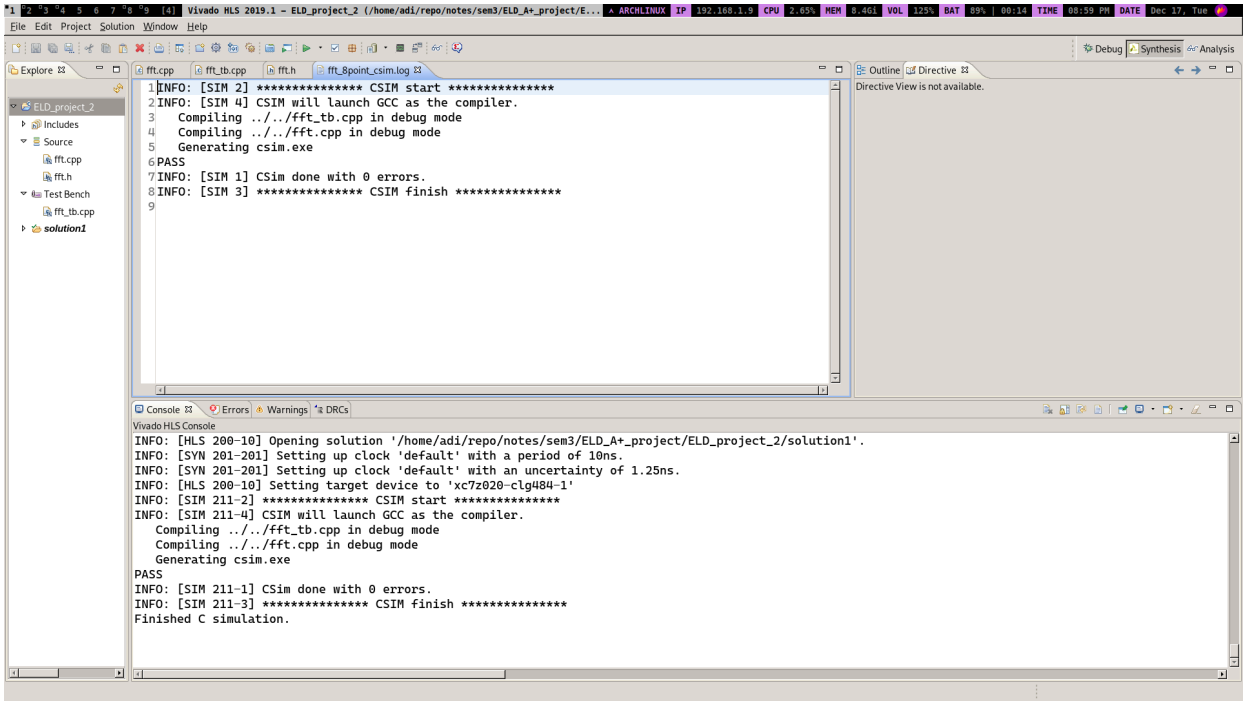
- When running export RTL to IP, it was showing a very weird error ->



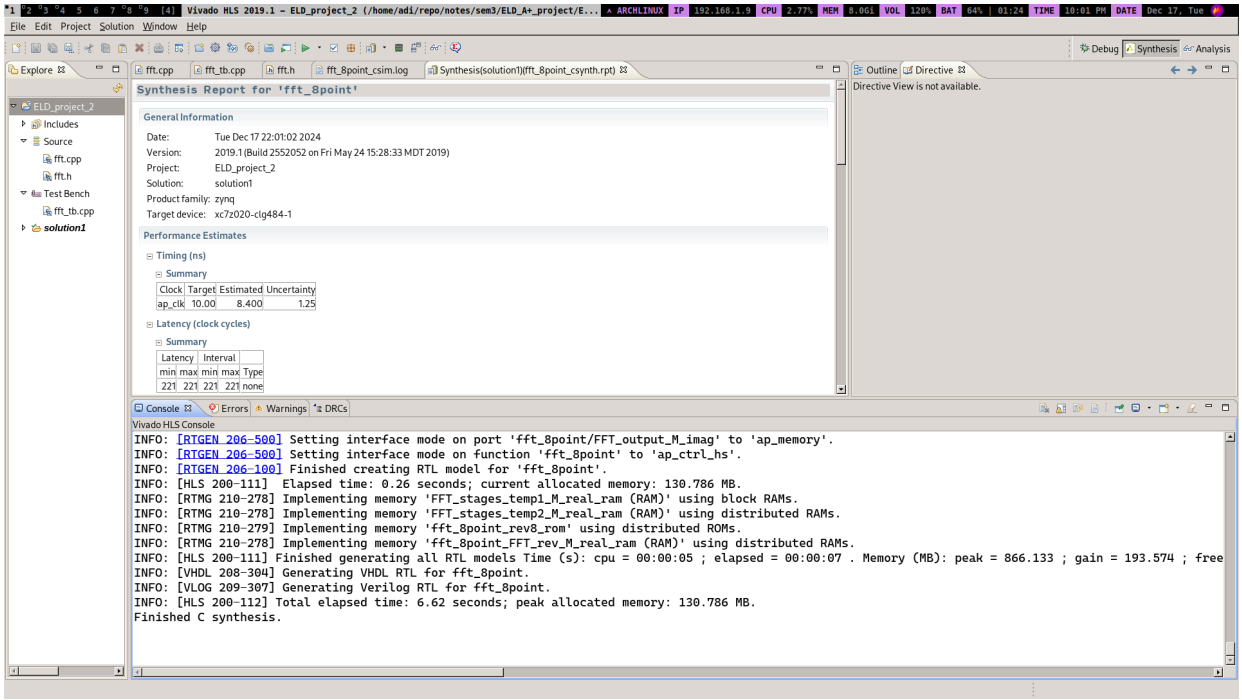
- After searching for this error for hours online, I found a very old AMD forum comment regarding switching back the date to something in 2019 and I gave it a shot and it finally worked and I was able to export the IP.
- Through this note, I would like to bring to your attention that I am trying my best to make this project work but I am not able to fully achieve the objective because of reasons I cannot resolve such as the Vivado version I am using. Hence, I would like you to consider the efforts and research put in the project and not just the final outcome of the project.

• Milestones

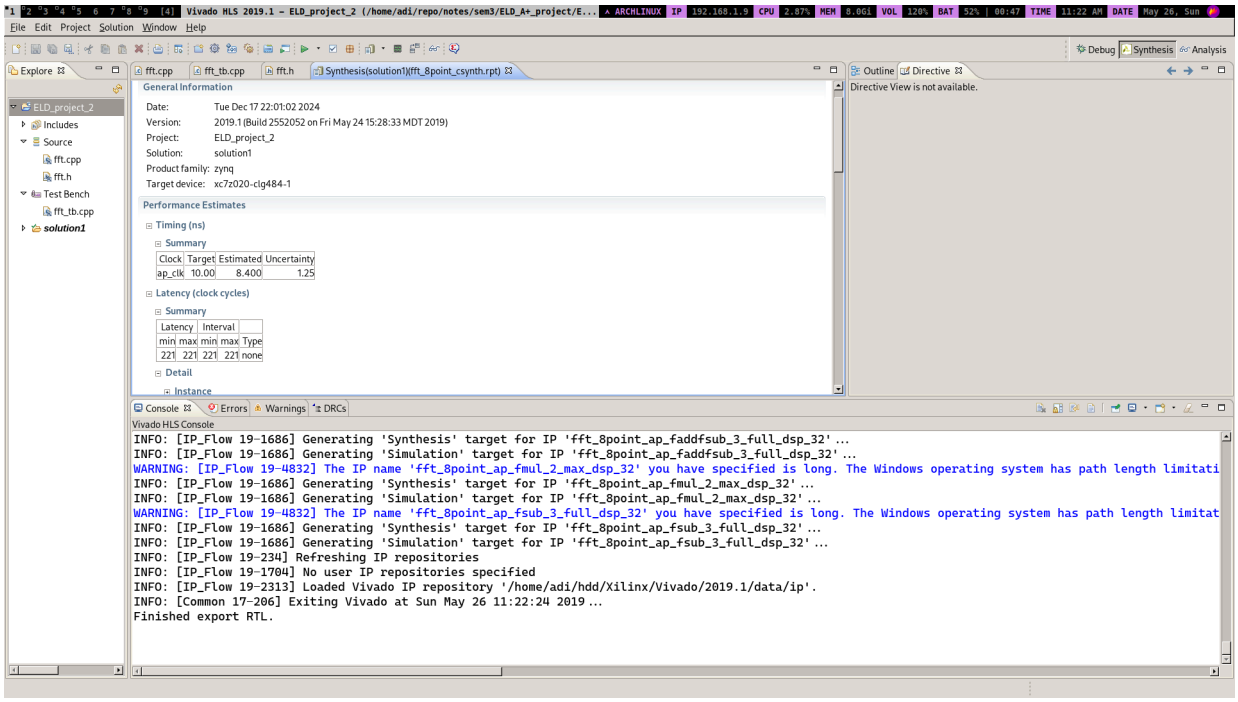
- IP Simulation Result



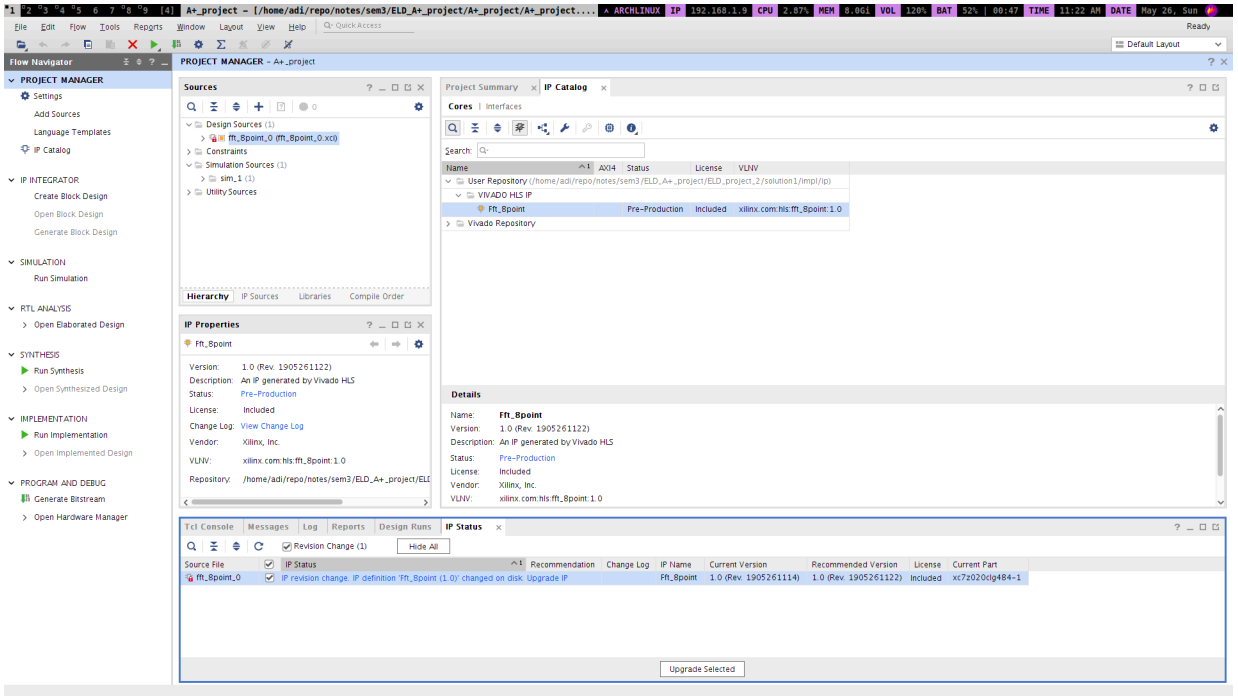
- IP Synthesis Result



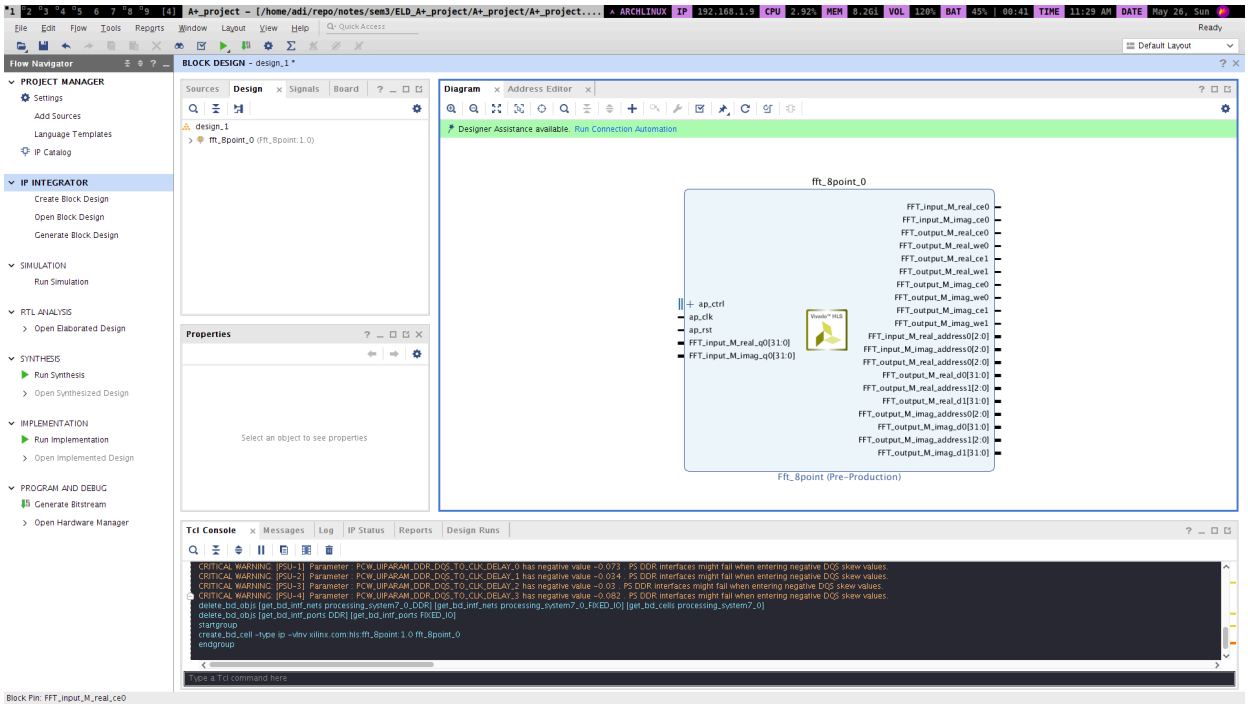
- IP Export Result



- Importing IP in Vivado



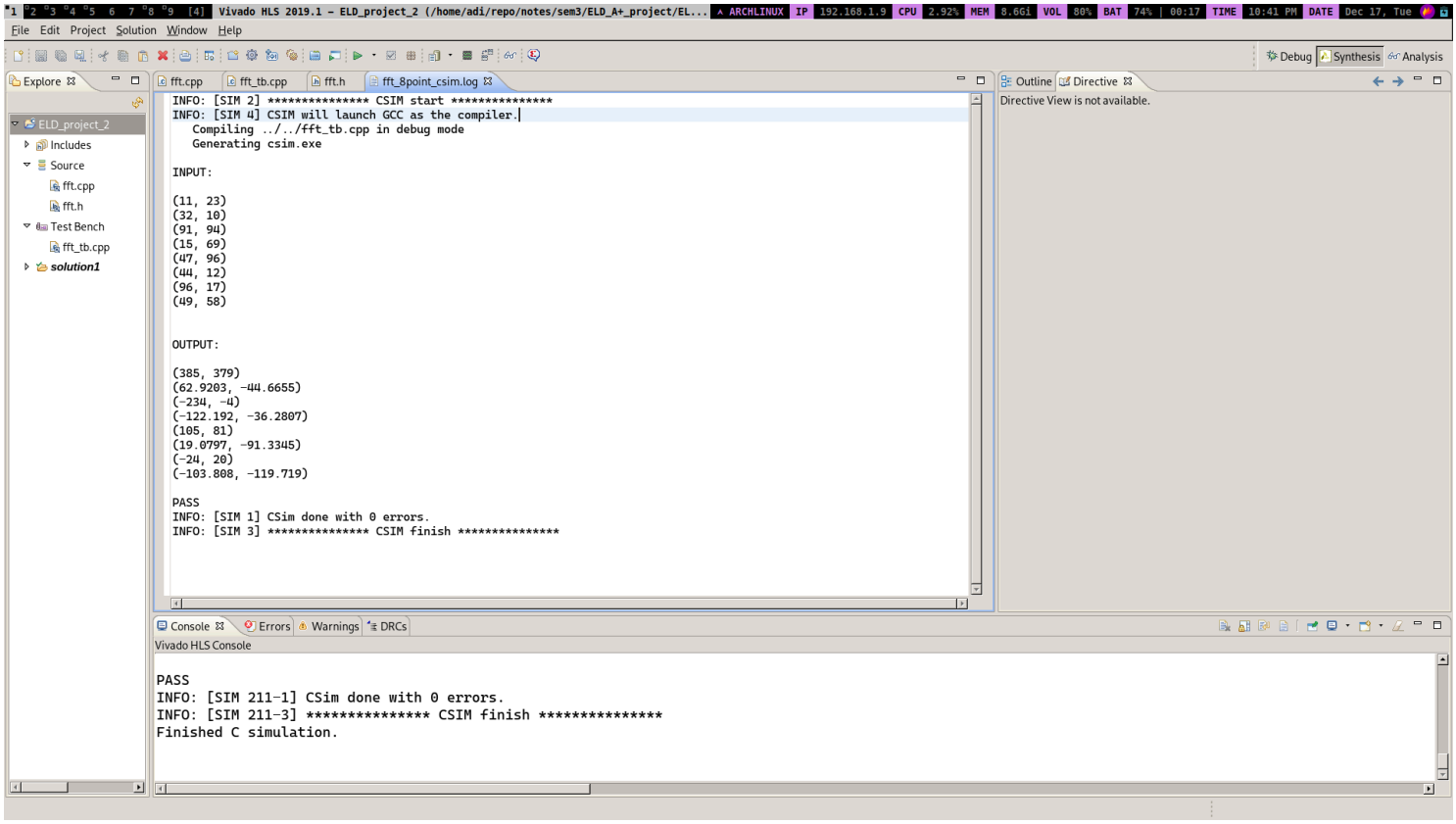
- Block Diagram

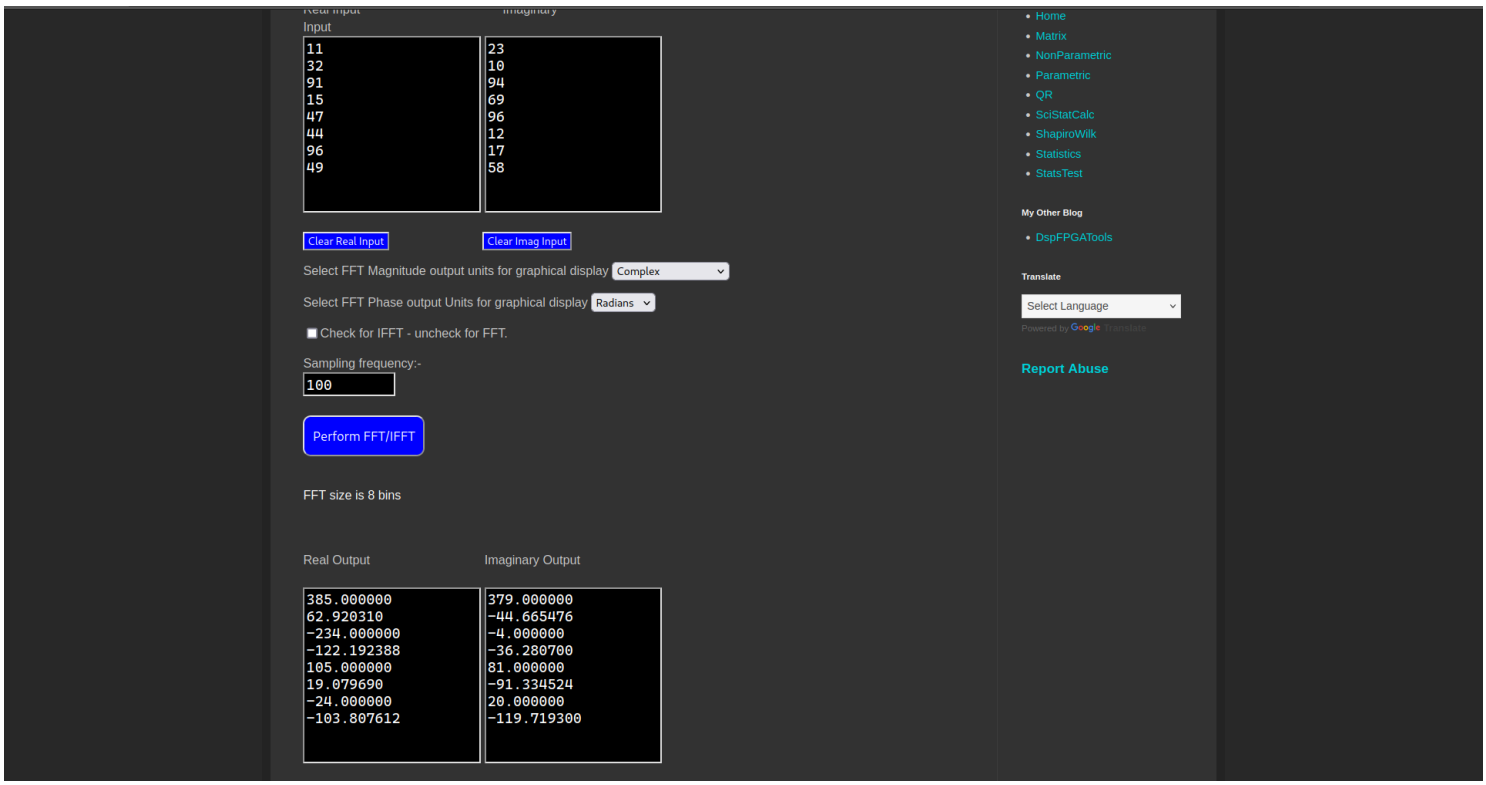


Since I was not able to make the AXI Stream Interface work in my custom 8 point FFT IP, I will not be able to properly connect the IP with ZynQ IP and hence will not be able to show the proper block diagram. Also, the date and time before this screenshot will look messed up because I had set it to 2019 because of the problem mentioned above.

• Results

- Since I was not able to make the proper block diagram, I could not load up Vivado SDK and run the IP and observe the results, so I will use the test bench in Vivado HLS to test the logic of my code.


Vivado HLS 2019.1 – ELD_project_2 (/home/adi/repo/notes/sem3/ELD_A+_project/EL...
File Edit Project Solution Window Help
Explore
ELD_project_2
Includes
Source
fft.cpp
fft.h
Test Bench
fft_tb.cpp
solution1
fft.cpp
fft_tb.cpp
fft.h
fft_8point_csim.log
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
Compiling ../../fft_tb.cpp in debug mode
Generating csim.exe
INPUT:
(11, 23)
(32, 10)
(91, 94)
(15, 69)
(47, 96)
(44, 12)
(96, 17)
(49, 58)
OUTPUT:
(385, 379)
(62.9203, -44.6655)
(-234, -4)
(-122.192, -36.2807)
(105, 81)
(19.0797, -91.3345)
(-24, 20)
(-103.808, -119.719)
PASS
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
Outline
Directive
Directive View is not available.
Console
Errors
Warnings
DRCs
Vivado HLS Console
PASS
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] *************** CSIM finish ***************
Finished C simulation.


Input
11
32
91
15
47
44
96
49
23
10
94
69
96
12
17
58
Clear Real Input
Clear Imag Input
Select FFT Magnitude output units for graphical display
Complex
Select FFT Phase output Units for graphical display
Radians
Check for IFFT - uncheck for FFT.
Sampling frequency:-
100
Perform FFT/IFFT
FFT size is 8 bins
Real Output
Imaginary Output
385.000000
62.920310
-234.000000
-122.192388
105.000000
19.079690
-24.000000
-103.807612
379.000000
-44.665476
-4.000000
-36.280700
81.000000
-91.334524
20.000000
-119.719300
Home
Matrix
NonParametric
Parametric
QR
SciStatCalc
ShapiroWilk
Statistics
StatsTest
My Other Blog
DspFPGATools
Translate
Select Language
Powered by Google Translate
Report Abuse

Input

Real Input	Imaginary Input
1	0
2	0
3	0
4	0
5	1
6	2
7	3
8	4

Clear Real Input Clear Imag Input

Select FFT Magnitude output units for graphical display: Complex

Select FFT Phase output Units for graphical display: Radians

Check for IFFT - uncheck for FFT.

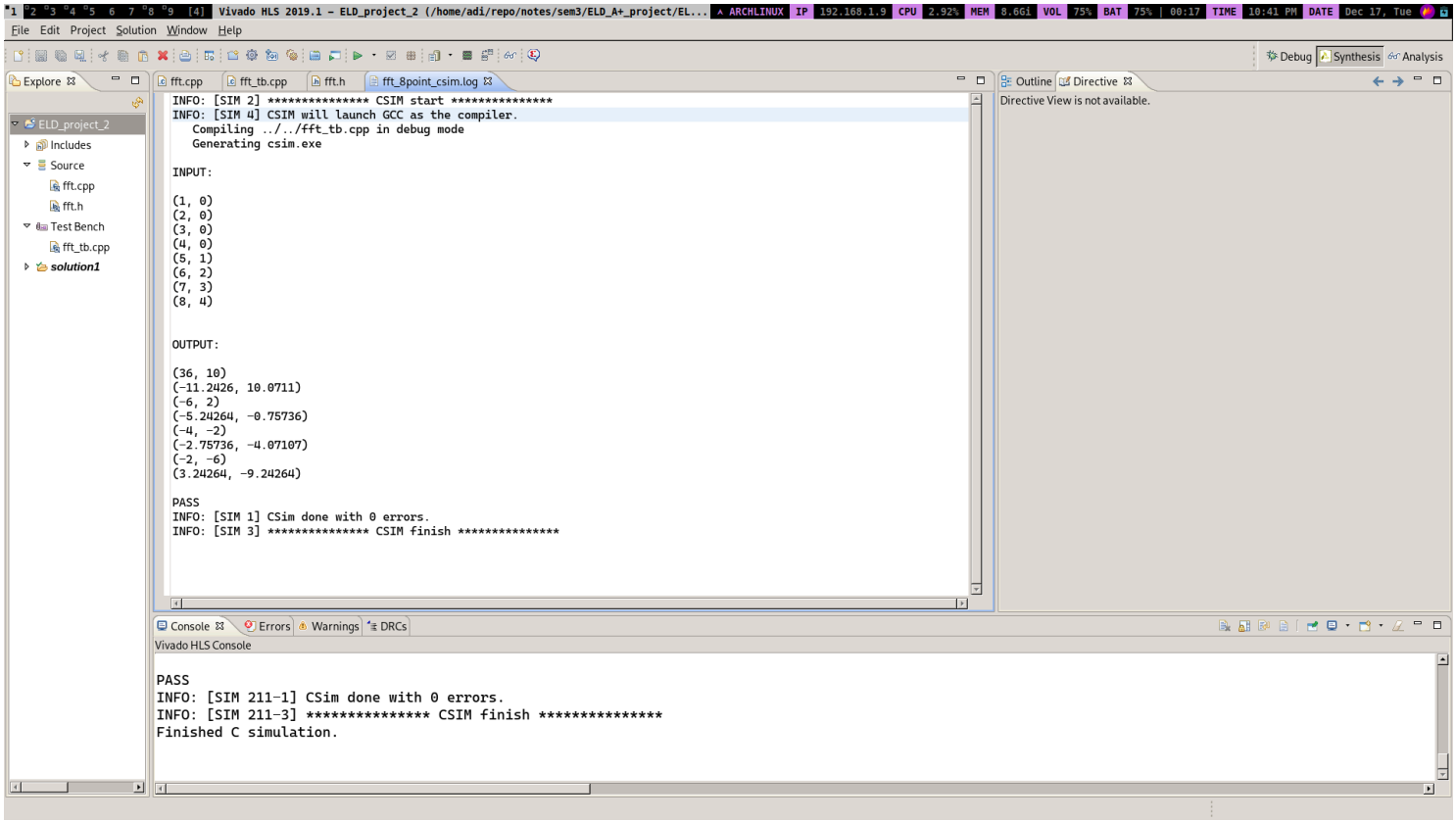
Sampling frequency:-

100

Perform FFT/IFFT

FFT size is 8 bins

Real Output	Imaginary Output
36.000000	10.000000
-11.242641	10.071068
-6.000000	2.000000
-5.242641	-0.757359
-4.000000	-2.000000
-2.757359	-4.071068
-2.000000	-6.000000
3.242641	-9.242641



- ## Conclusion

  Through this project, I was able to expand my knowledge in the Embedded Systems Domain and was able to create my own custom IP based on 8 Point Fast Fourier Transform, utilizing the Vivado HLS (high level synthesis) tool which converts C/C++ code to Verilog code which can be used to program Field Programmable Gate Arrays (FPGAs).