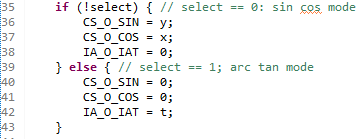
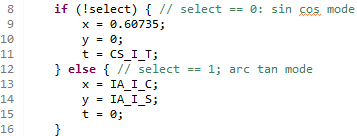
**ELEC 522 Assignment 4**

Student: Sixu Li (S01435077)

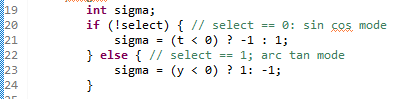
# 1. Write C/C++ code for the CORDIC circular mode module using VivadoHLS

The CORDIC core is designed with two modes. Hardware resources are reused as a reconfigurable design.

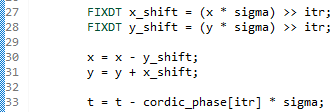
First is the reconfigurable design in I/O;



Apart from the reuse in I/O. The main reconfigurable part is the sigma, which will change in different modes.



Other computation units are all reused and will not change when in different modes.



In this way, the proposed module should consume less hardware resources compared to design two modules separately.

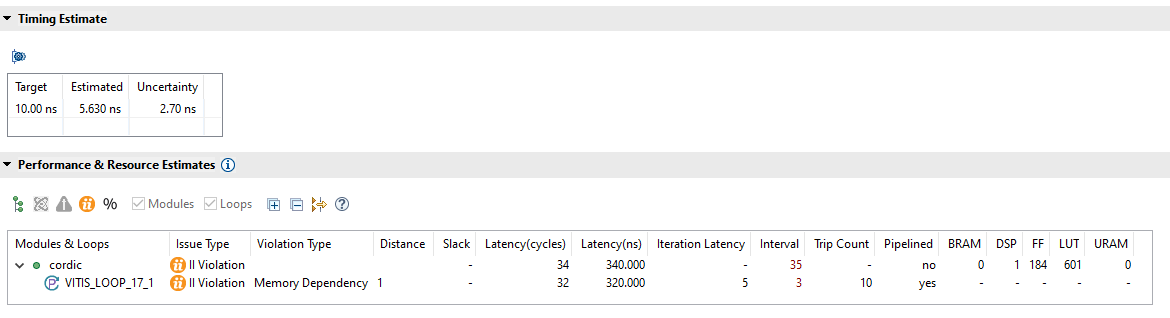
# 2. Write testbench code to verify the function of the C/C++ code code in VivadoHLS

In the C testbench, we choose to verify the cos and sin mode. We choose . The results matched but with some small errors.

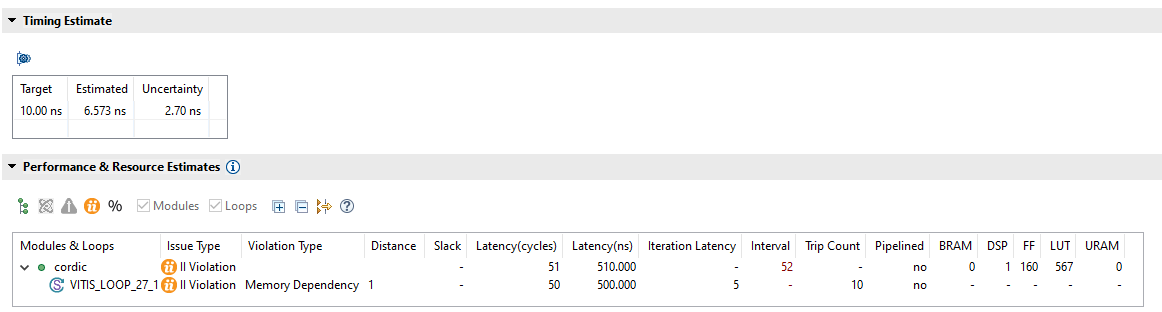


# 3. Architecture optimization, constraint configurations and advantages of final design.

Vivado HLS has automatically enabled pipeline for the FOR-LOOP. So the default design is optimal considering this FOR-LOOP is the only potential optimization oprtator.



But we can disable auto pipeline by insert #pragma HLS pipeline off. We can see that if we turn off pileine , we can save less than 10% area but get 60% more latency. So we choose the one with pipeline.



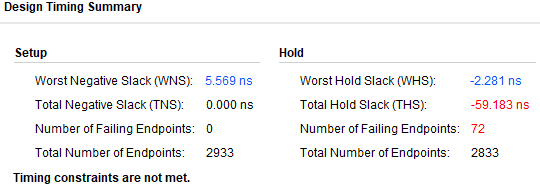
# 4. Model Composer model using the VivadoHLS block and testing results.

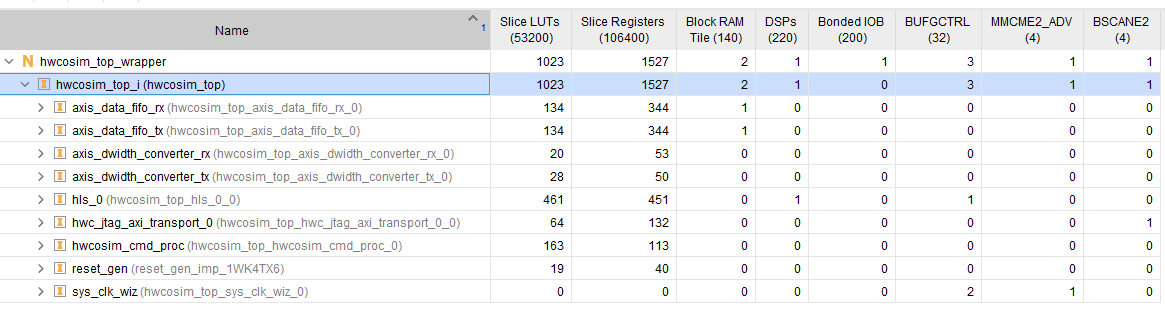
See screenshots in Question 6.

# 5. Synthesis and place and route implementation report.

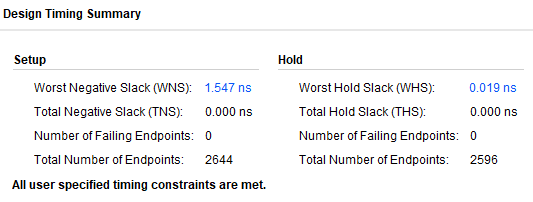
The following screenshots are the timing & utilization reports post-synthesis & post-PR.

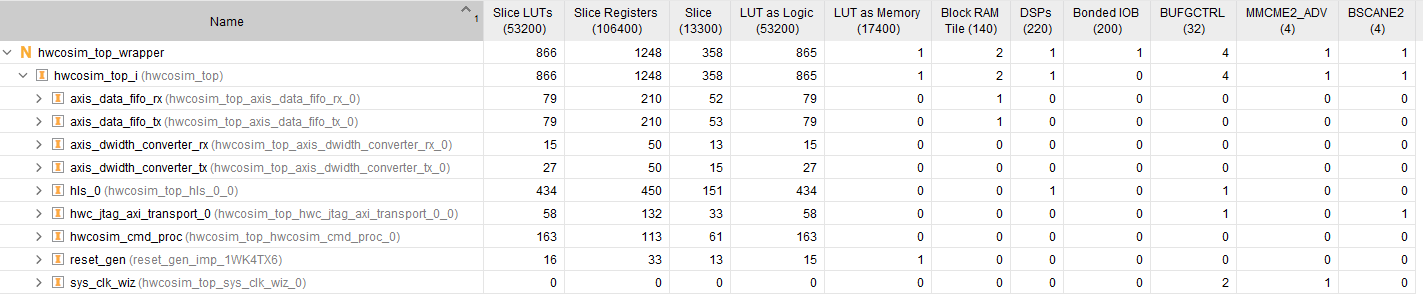
post-synthesis





post-PR

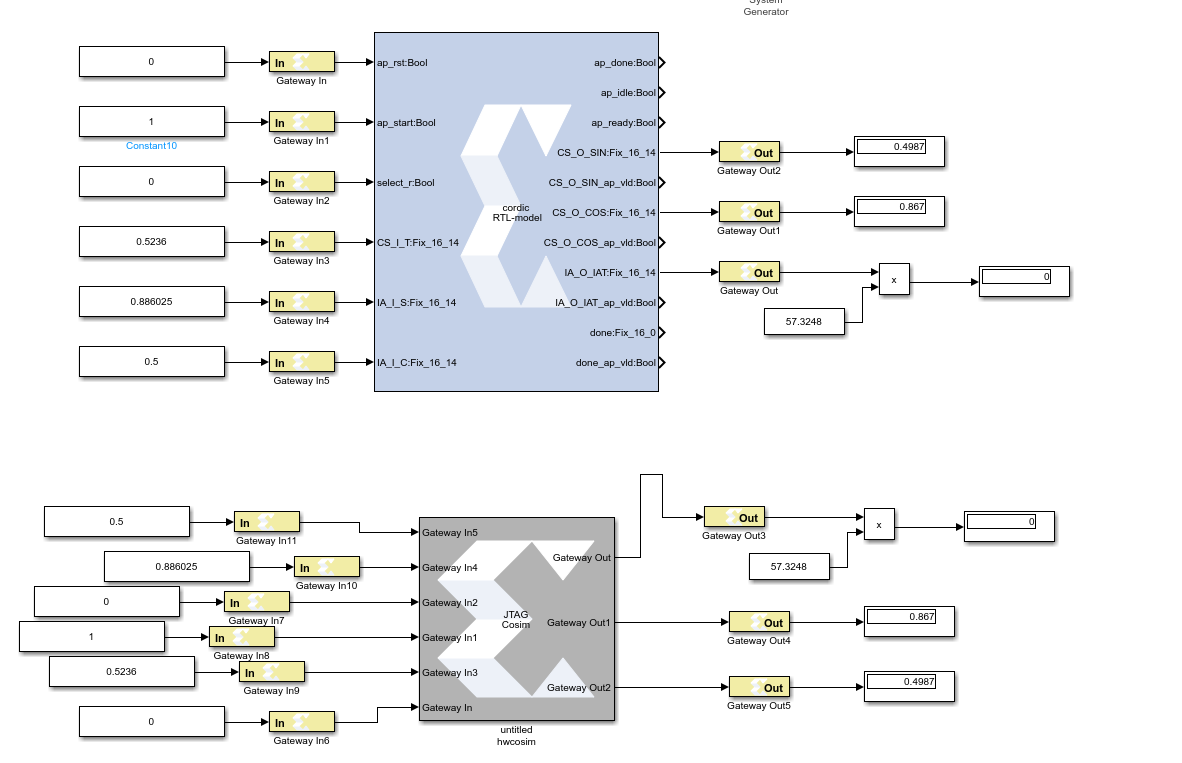


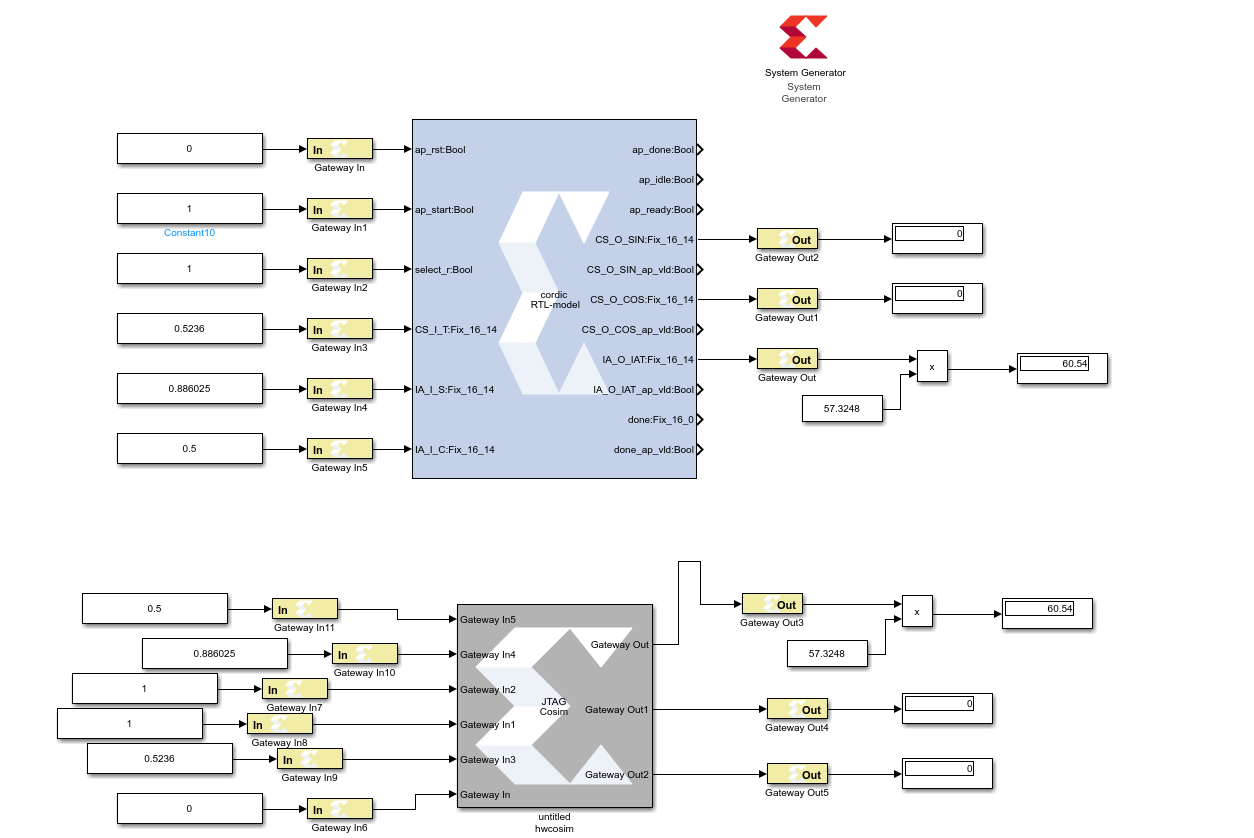
4

# 6. Hardware in the loop Co-Simulation to verify performance of VivadoHLS code on the ZedBoard in the lab.

The following figure are the test results for mode 0 (sin&cos) and mode 1 (arctan).

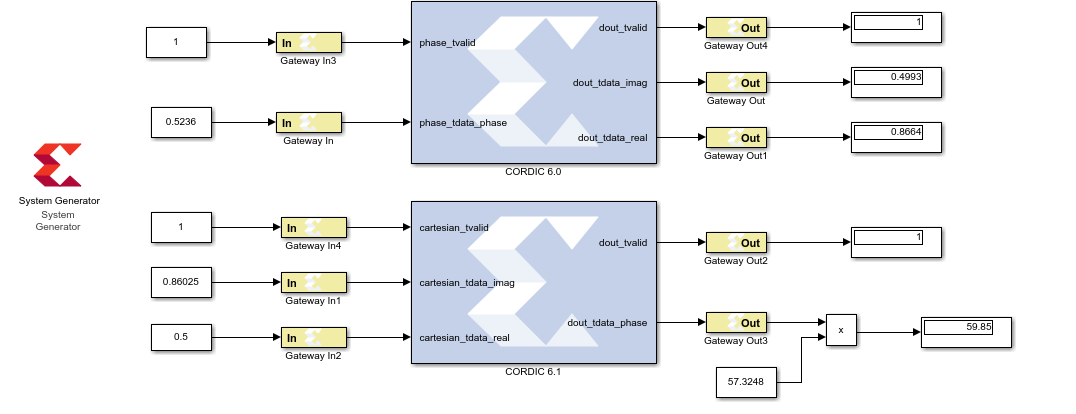
We can see that the Simulink result have some errors compared to HLS C simulation, but within 0.05% which can be ignored.



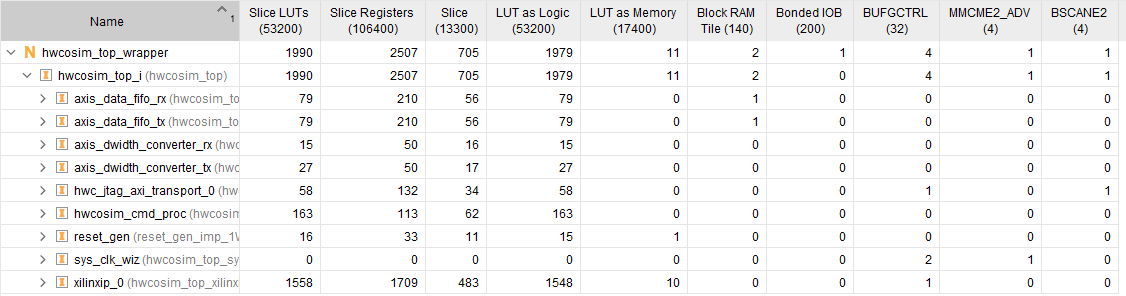


# 7. Results comparison with the built-in Xilinx CORDIC modules in SysGen in terms of numerical results and also FPGA area usage.

The iteration is configured to 10 which is the same as the HLS verision.



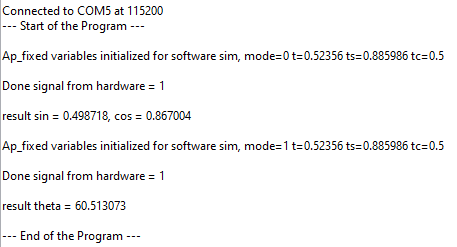
The numerical results in the built in CORDIC module is slightly different compared to my HLS module, but the relative error is smaller than 2%.



We can see that the Xilinx CORDIC consume two times of hardware resources. This makes sense considering we use two CORDIC unit for SINCOS and ARCTAN which will cost more area compared to reconfigurable design.

# 8. IP block export to SDK and Zynq ARM program control

The following figure shows the UART output of vitis arm baremetal program, we can see that the screenshot matched the C testbench.



# 9. Turning in files

All the files are attached in this zip file.