**EECS 395: CORDIC ALGORITHM IN VHDL**

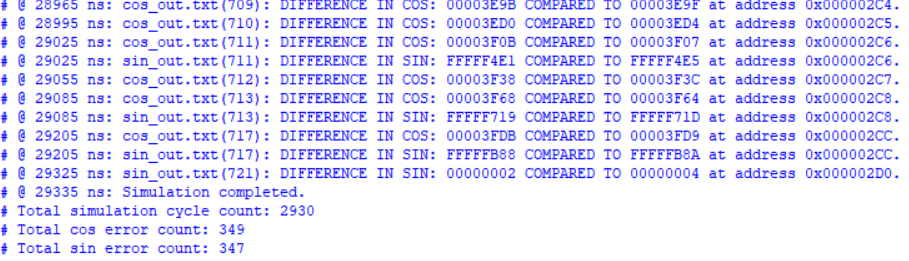
Jae Woo Ok

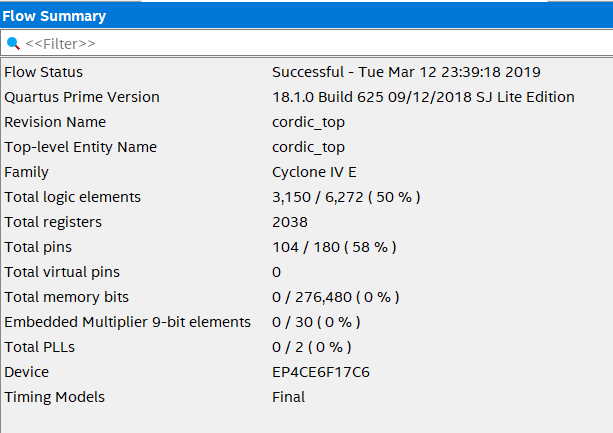
Compared to my first submission, I have made a lot of changes and fixes which I have listed as below.

1. I still kept the FSM (although we didn’t really need it, but I thought it’d be easier to just expand on it) but I have revised it so that it now creates the pipelined stages through a for-generate by mapping to a separate Cordic step, instead of keeping it under one entity from my first submission.
2. After running some tests, I realized that my original FSM was only able to traverse from -90 to 90 degrees so I made some changes so that it would be able to take in all angles. The basic idea of the clocked process, pipelined process, and fsm process is still the same.
3. Changed my testbench so that now it is able to show the differences between the hardware and software sin cos values.

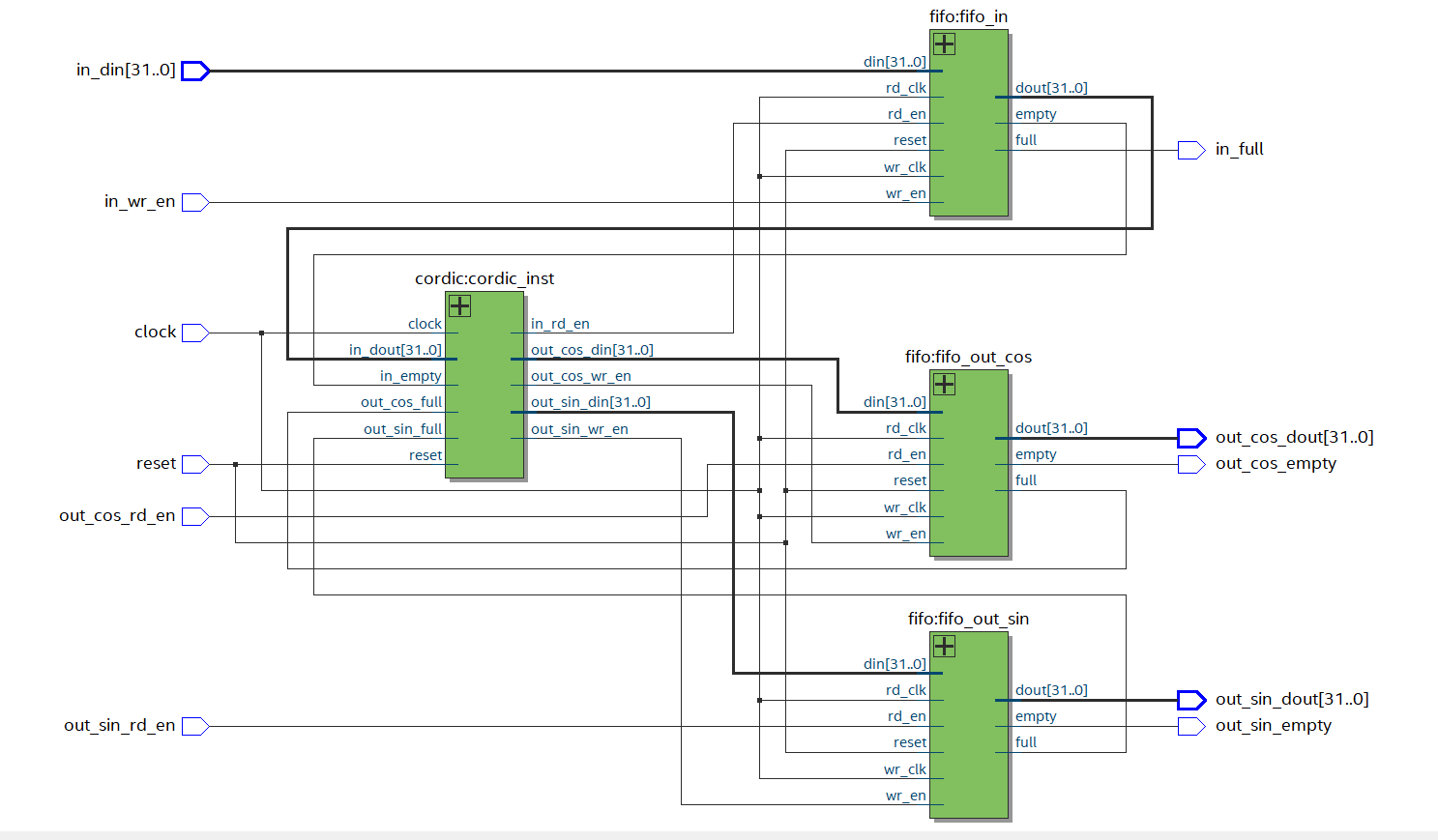
After running the corrected code, these are the new results.

**Cycle Time and Flow Summary:**





**The RTL of the Design**



Throughput: 721 samples (-360 to 360) / (2930 cycles \* 10 ns/cycles) = 2.46 \* 10^7 samples/s

**Cordic Core Description:**

Because the Cordic Core gets the inputs from a FIFO and outputs into a FIFO, I had just assumed that it used an FSM like previous projects. Thus, the Core gets the inputs through the s0 state, and the s1 state uses the input angle to get the output. Then in the s2 state, the Core sends the output of the cordic to the input of the next cordic to get the pipelined stages. The cordic step gets the x, y and phase vectors as the input and outputs the new x, y, and phase vectors. By using the for-generate function on this step, we are able to get the desired pipelined design.