Laboratory 2

(Due date: May 27th)

OBJECTIVES

- ✓ Design an AXI4-Lite Interface for a custom VHDL peripheral.
- ✓ Integrate the custom VHDL peripheral in an embedded system in Vivado.
- ✓ Create a software application in SDK that can communicate with the custom peripheral.

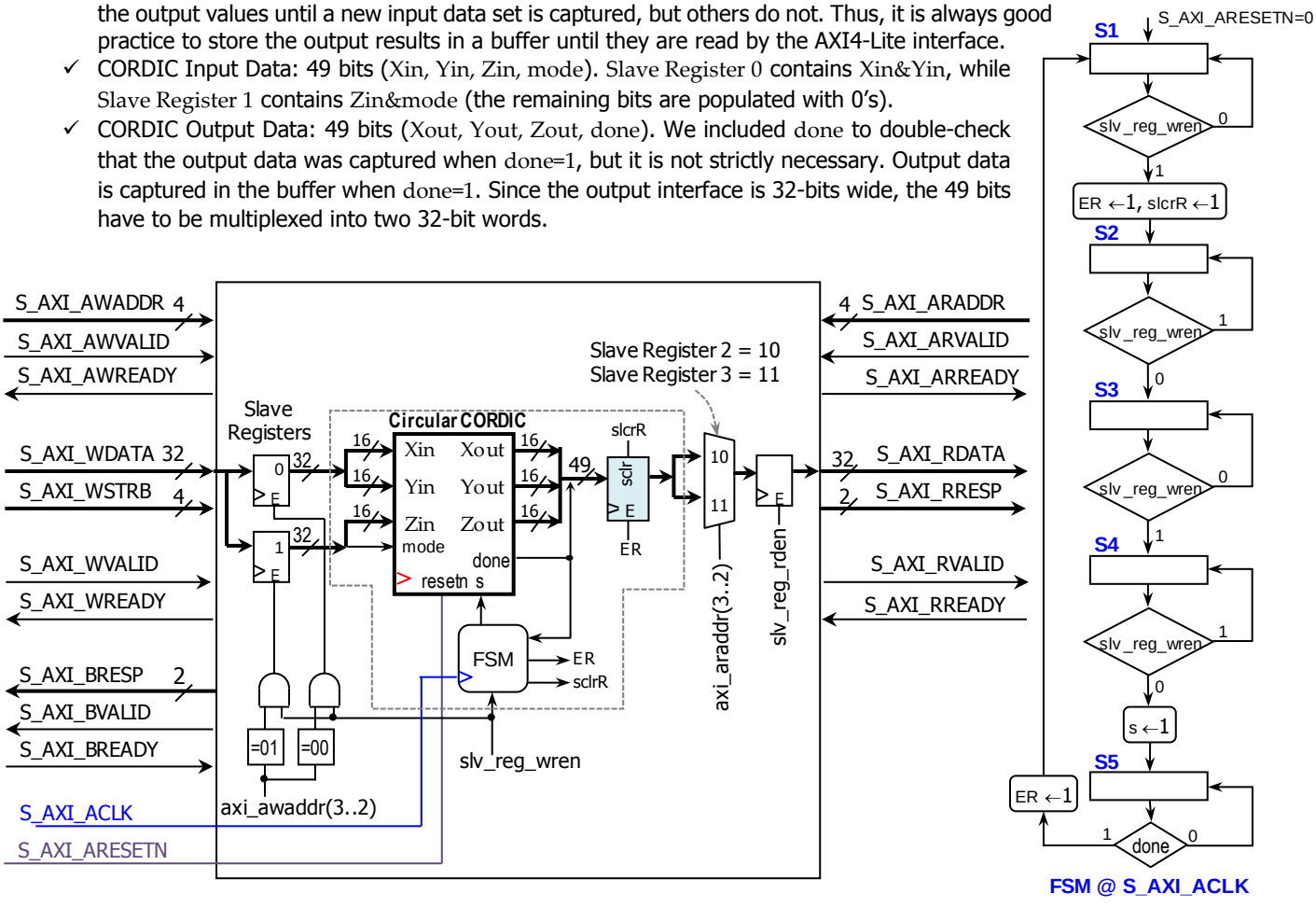
VHDL CODING

- ✓ Refer to the Tutorial: VHDL for FPGAs for a tutorial and a list of examples.
- ✓ Refer to the Tutorial: Embedded System Design for Zynq PSoC for information on how to create AXI interfaces for custom peripherals as well as embedded system integration in Vivado.

FIRST ACTIVITY (100/100)

- Custom Hardware Peripheral: Circular CORDIC Architecture (see Notes – Unit 3):
  - ✓ Operation: The circuit reads input data (16-bit Xin, 16-bit Yin, 16-bit Zin, and mode) when the s signal (usually a one-cycle pulse) is asserted. When the result (16-bit Xout, 16-bit Yout, 16-bit Zout) is ready, the signal done is asserted. Only after this, we can feed a new input data set (with s = 1). This is an iterative circuit.
  - ✓ VHDL design (provided in `mycordic.zip`): It uses the signed FX format [16 14] to represent the inputs and outputs.
    - Design sources: `mycordic.vhd` (top file). This file uses all the other `.vhd` design files. These files make up the design for which you need to create an AXI4-Lite interface around.
    - Simulation sources (supplemental): `tb_mycordic.vhd` (testbench); `in_benchR.txt, in_benchV.txt` (test data).

- Vivado: Create an AXI4-Lite Interface for the Circular CORDIC architecture.
- AXI4-Lite Interface: A suggested architecture is depicted. It uses 4 Slave Registers, a Register, and an FSM.
  - ✓ Extra Register: Buffer that stores the output results (when done=1). Most digital systems with an iterative behavior keep the output values until a new input data set is captured, but others do not. Thus, it is always good practice to store the output results in a buffer until they are read by the AXI4-Lite interface.
  - ✓ CORDIC Input Data: 49 bits (Xin, Yin, Zin, mode). Slave Register 0 contains Xin&Yin, while Slave Register 1 contains Zin&mode (the remaining bits are populated with 0's).
  - ✓ CORDIC Output Data: 49 bits (Xout, Yout, Zout, done). We included done to double-check that the output data was captured when done=1, but it is not strictly necessary. Output data is captured in the buffer when done=1. Since the output interface is 32-bits wide, the 49 bits have to be multiplexed into two 32-bit words.



- Once your custom AXI4-Lite Peripheral is ready, integrate it into an embedded system using the Block-Based Design approach in Vivado. Synthesize, Implement, and generate the bitstream.

SDK SOFTWARE APPLICATION

- To compute one input set at a time, you can write two 32-bit words and then read two 32-bit words. The output word includes the signal done. If done=1, the software routine can start a new computation.
  - ✓ You can poll for done=1 before you read output data. But if you want to simplify your code, you can read right after the last write. Since this circuit has a relatively long processing latency (~16 clock cycles), it is recommended to insert a small delay between the last write and the read.

- Write a software application that test the circuit for the cases shown in the table. Write input data on the peripheral and retrieve output data form the peripheral. Print out the results as hexadecimal characters on the SDK terminal (via UART). Verify the results match the expected output results (approximately).
  - ✓ Note that the numbers are represented in signed FX format [16 14]. To convert from FX to decimal (and vice versa), use any online tool or these MATLAB/Octave scripts.

$A_n = 1.6468$	**Input Data**			**Expected Output Results**		
	x_0	y_0	z_0	x_N	y_N	z_N
Rotation Mode (mode = 0)	0	$1/A_n$	$\pi/6$	$-\sin(\pi/6)$	$\cos(\pi/6)$	0
	0	$1/A_n$	$-\pi/3$	$-\sin(-\pi/3)$	$\cos(-\pi/3)$	0
Vectoring Mode (mode = 1)	0.8	0.8	0	$A_n\sqrt{0.8^2+0.8^2}$	0	$\tan^{-1}(1)$
	0.5	1	0	$A_n\sqrt{0.5^2+1^2}$	0	$\tan^{-1}(2)$

- Download the hardware bitstream on the ZYNQ PSoC.
- Launch your software application on the Zynq PS. The program should display the output results (as hexadecimal characters) on the Terminal. **Demonstrate this to your instructor**.

- Submit (as a .zip file) the generated files: VHDL code (the .vhd files that constitute the AXI4-Lite Peripheral, see Embedded System Design for Zynq PSoC Tutorial → Unit 3 for examples), .c files to Moodle (an assignment will be created). DO NOT submit the whole Vivado Project.

Instructor signature: _____ **Date:** _____