Laboratory 3

(Due date: June 3rd)

OBJECTIVES

- ✓ Design an AXI4-Full 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.

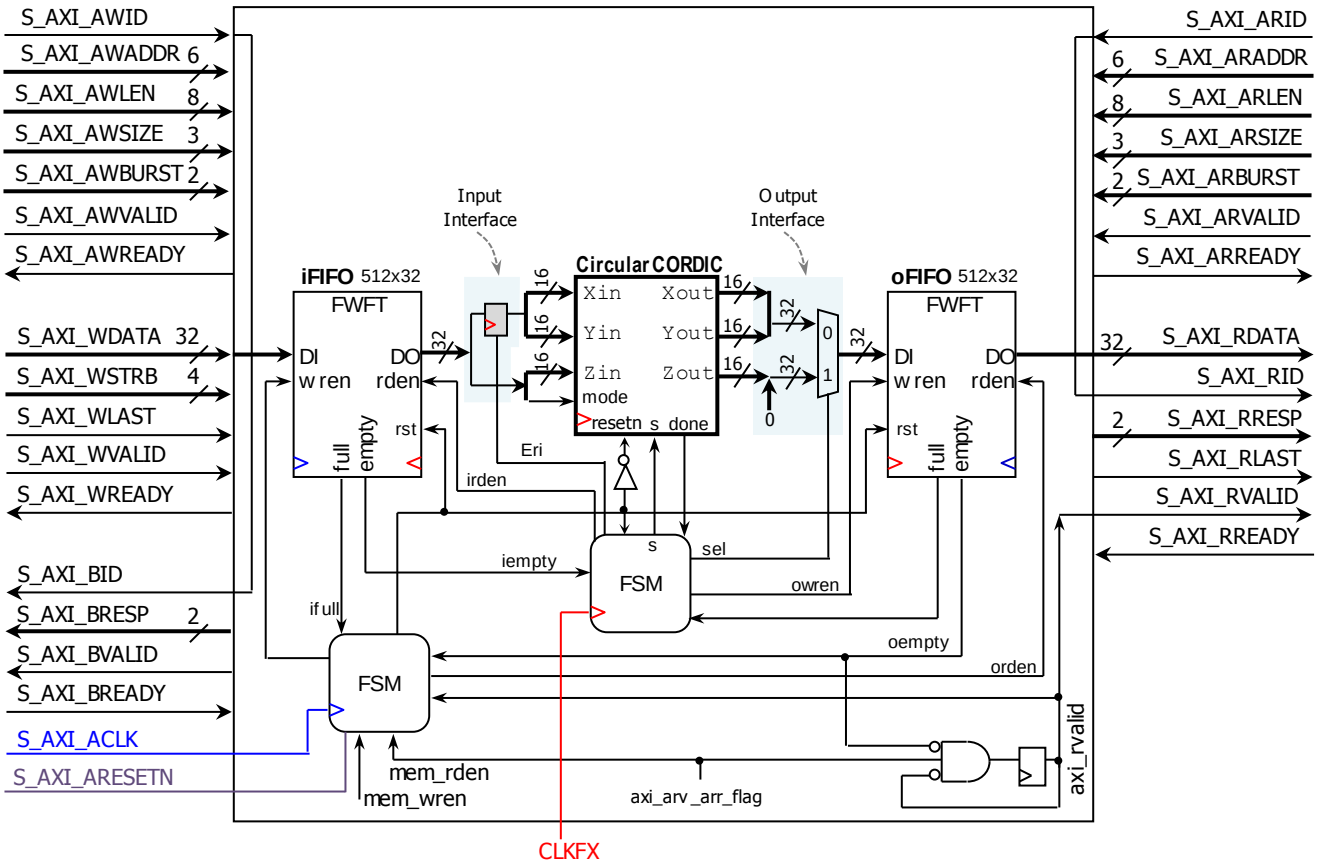
VHDL CODING

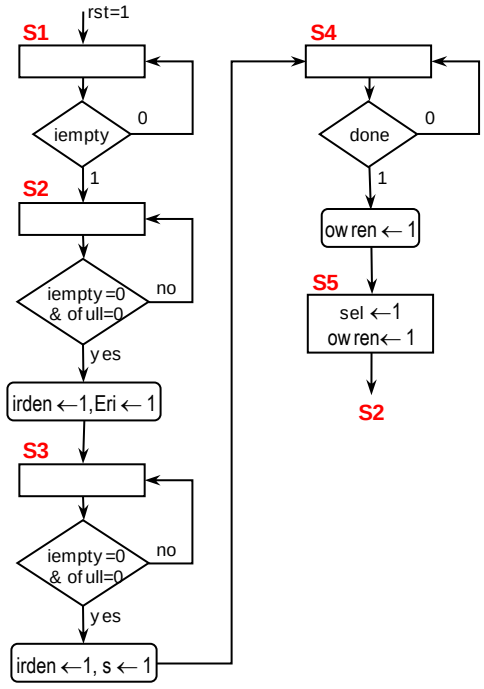
- ✓ Refer to the Tutorial: VHDL for FPGAs for a tutorial and a list of examples.
- ✓ Refer to the Tutorial: Embedded System Design for Zynq PSoC for information on how to create AXI interfaces for custom peripherals as well as embedded system integration in Vivado.

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 that keeps the output values until a new input data set is captured (with s=1). This is an advantageous feature.
  - ✓ VHDL design (provided in `mycordic.zip`): It uses the signed FX format [16 14] to represent the inputs and outputs.

- Vivado: Create an AXI4-Full Interface for the iterative Circular CORDIC.
- AXI4-Full Interface: A suggested architecture is depicted.
  - ✓ We use the same iFIFO/oFIFO approach used for the Pixel Processor (See Notes – Unit 5).
  - ✓ Note that the *reset* signal generated by FSM @ S_AXI_ACLK is active-high.
  - ✓ FSM @ S_AXI_ACLK: It is the same as the one used for the Pixel Processor.
  - ✓ FSM @ CLK_FX: This FSM controls the Input and Output interfaces to the FIFOs as well as FIFOs' signals.
  - ✓ Input and Output Interfaces: Since AXI bus size is 32 bits wide, we need to properly route data in and out of the CORDIC circuit (49-bit input, 48-bit output). The 48-bit output is stored onto oFIFO as two 32-bit words via a MUX (no need for a buffer as the CORDIC result is kept until s is asserted again). This runs @ CLK_FX. We connect CLK_FX to S_AXI_ACLK.





FSM @ CLKFX

- If you prefer to use a different approach, provide a detailed schematic of your AXI4-Full interface and the FSMs.

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

SDK SOFTWARE APPLICATION

- Since we have a FIFO-based system, we can write a chunk of input sets (two 32-bit words) and then retrieve the corresponding results. This is the optimal way to use this AXI4-Full Interface. You can also do it one input set at a time (write two 32-bit words, then read two 32-bit words).
- 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 ($\text{mode} = 0$)	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 ($\text{mode} = 1$)	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 on the Terminal. **Demonstrate this to your instructor**.

- Submit (as a .zip file) the generated files: VHDL code, .c files to Moodle (an assignment will be created). DO NOT submit the whole Vivado Project.

Instructor signature: _____ **Date:** _____