Laboratory 2

(Due date: October 11h)

OBJECTIVES

- ✓ Design of a dedicated circuitry for trigonometric functions (CORDIC) in fixed-point arithmetic: FSM + Datapath
- ✓ Test of the CORDIC circuit using fixed-point inputs/outputs.

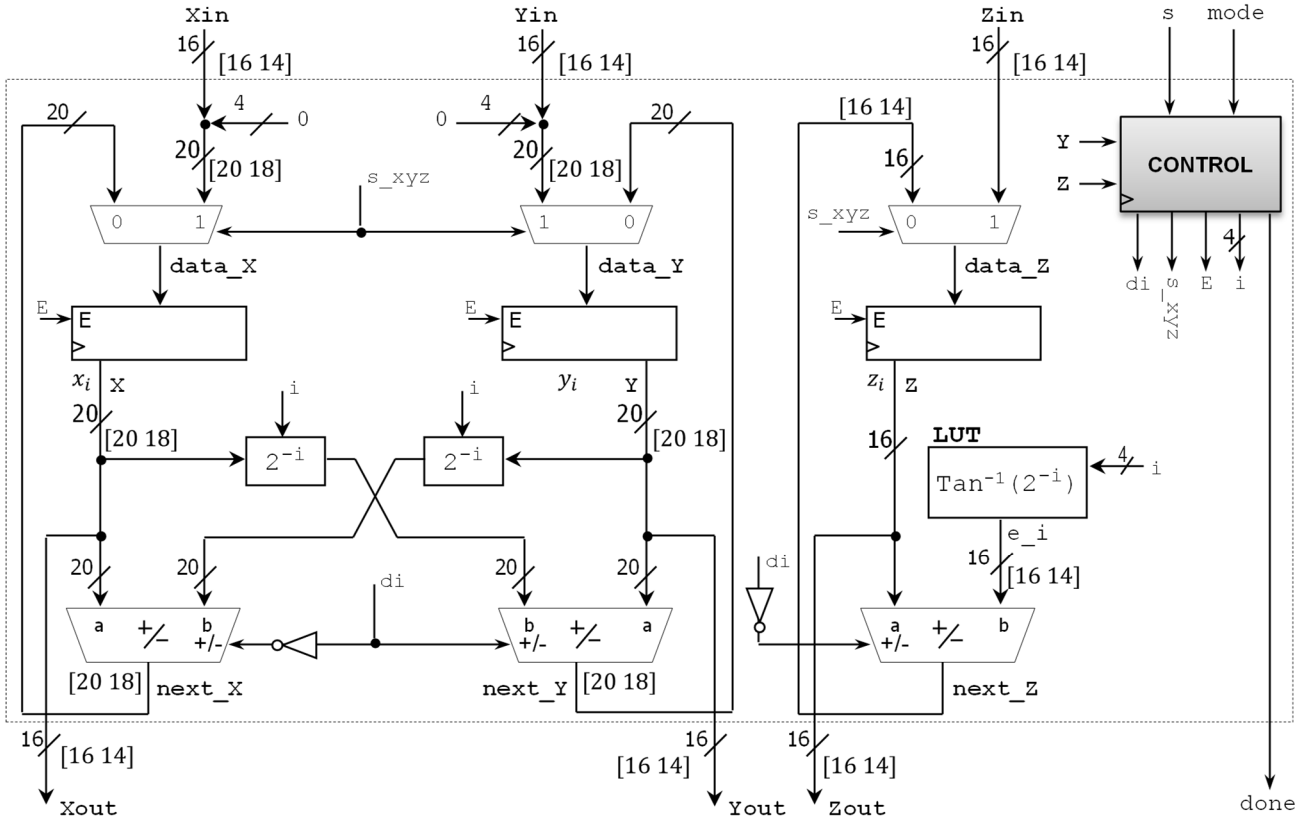
VHDL CODING

- ✓ Refer to the Tutorial: VHDL for FPGAs for a tutorial and a list of examples.

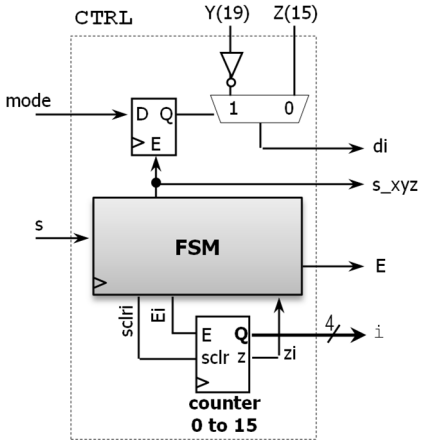
FIRST ACTIVITY (100/100)

- Design the iterative Expanded Circular CORDIC FX architecture for $i = 0,0,0,1,2,3,\ldots 15$. x_0, y_0, z_0: initial conditions. $mode$ = '0' → Rotation Mode. $mode$ = '1' → Vectoring Mode. Iteration $i = 0$ is repeated two extra times in order to achieve convergence over the entire domain of $sin(x)$ and $cos(x)$. This affects the value of A_n.
  MATLAB scripts (`cordic_circular_esp.m`, `testcordicesp.m`): They implement the expanded circular CORDIC algorithm.
- **Operation**: When $s = 1$, x_{in}, y_{in}, z_{in}, and $mode$ are captured. Data will then be processed iteratively. When data is ready ($done$ = '1'), output results appear in $x_{out}, y_{out}, z_{out}$.
- **Input/Intermediate/Output FX Format (signed)**:
  - ✓ Input values: x_{in}, y_{in}, z_{in}: [16 14]. Output values: $x_{out}, y_{out}, z_{out}$: [16 14]
  - ✓ Intermediate values: z_i: [16 14]. x_i, y_i: [20 18]. Here, we use 4 extra bits (add four 0's to the LSB) for extra precision.
  - ✓ We restrict the inputs $x_0 = x_{in}, y_0 = y_{in}$ to $[-1,1)$. Then, CORDIC operations need up to 2 integer bits (determined via MATLAB simulation). For consistency, we use 2 integer bits for all input/intermediate/output data.
- **Angles**: They are represented in the format [16 14]. Unit: radians.
- **Barrel shifters**: Use the VHDL code `mybarrelshifter.vhd` with `mode="ARITHMETIC"` (signed data), `N=20,SW=4,dir='1'`.

- Simulate it for the following cases. For each case verify that x_{16}, y_{16}, z_{16} reach the proper values.
  - ✓ Rotation Mode: $x_0 = 0, y_0 = 1/A_n, z_0 = -\pi/2$.
  - ✓ Rotation Mode: $x_0 = 0, y_0 = 1/A_n, z_0 = -5\pi/8$.
  - ✓ Rotation Mode: $x_0 = 0, y_0 = 1/A_n, z_0 = -\pi/6$.
  - ✓ Vectoring Mode: $x_0 = y_0 = 0.3, z_0 = 0$
  - ✓ Vectoring Mode: $x_0 = 0.1, y_0 = -0.3, z_0 = 0$
  - ✓ Vectoring Mode: $x_0 = 0.3, y_0 = -0.1, z_0 = 0$



- **Control**: This circuit controls the iteration index i, as well as the internal signals. The figure below is a suggested implementation.
  FSM: You need to design and implement a State Machine that controls the iteration index i, as well as the internal signals.



XILINX ZYNQ SOC DESIGN FLOW:

- ✓ Create a new Vivado Project. Select the **ZYNQ XC7Z010-1CLG400C** device.
- ✓ Using the structural coding approach in VHDL: Instantiate the barrel shifter, multiplexors, LUT, FSM, and adder/subtractors into a top file. Synthesize your circuit (Run Synthesis).
- ✓ Write the VHDL testbench to properly test the circuit.
- ✓ Perform Functional Simulation (Run Simulation → Run Behavioral Simulation). **Demonstrate this to your instructor.**

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

Instructor signature: _____ **Date:** _____