Laboratory 5

(Due date: **002**: March 28th, **003**: March 29th)

OBJECTIVES

- ✓ Design a dedicated circuitry for trigonometric circuit (CORDIC) in Fixed-point Arithmetic: FSM + Datapath
- ✓ Test the CORDIC circuit using real data represented in Fixed-Point Arithmetic.
- ✓ Learn how to read input and output text files for Simulation.

VHDL CODING

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

FIRST ACTIVITY: FX CIRCULAR CORDIC (100/100)

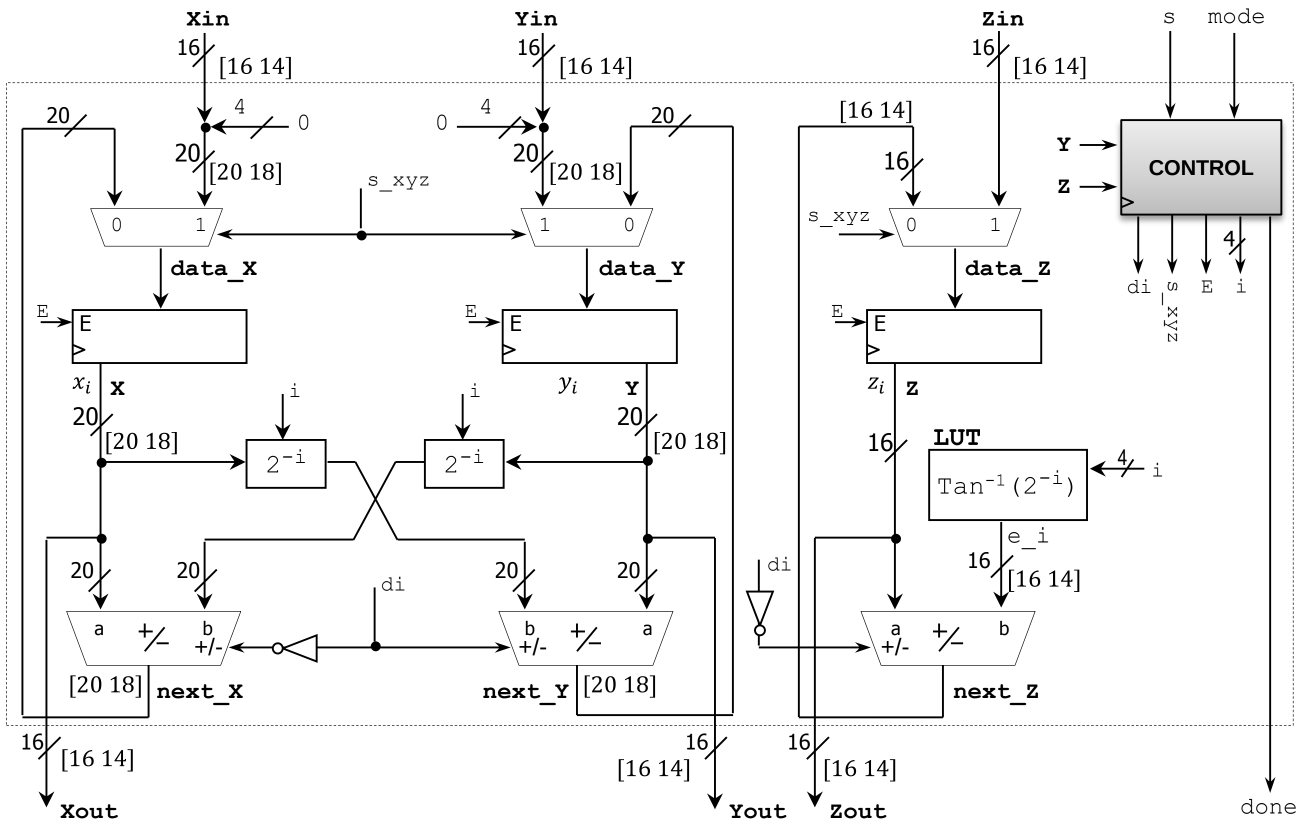
- Design the iterative Circular CORDIC FX architecture with N=16 iterations. $i = 0,1,2,3,\dots 15$. x_0, y_0, z_0: initial conditions. $mode$ = '0' → Rotation Mode. $mode$ = '1' → Vectoring Mode.

- **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 $x_{out}, y_{out}, z_{out}$.

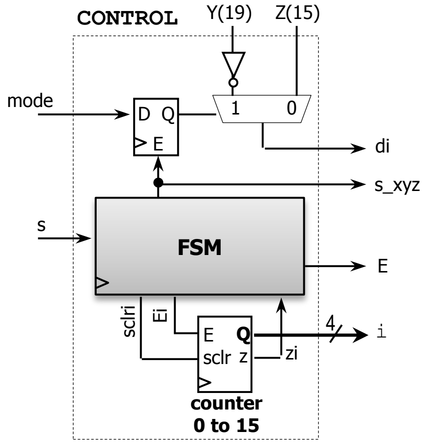
- **Input/Intermediate/Output FX Format**:
  - ✓ Input values: x_{in}, y_{in}, z_{in}: $[16\ 14]$. Output values: $x_{out}, y_{out}, z_{out}$: $[16\ 14]$
  - ✓ Intermediate values: z_i: $[16\ 14]$. x_i, y_i: $[20\ 18]$. Here, we use 4 extra bits (add four 0's to the LSB) for extra precision.
  - ✓ We 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.

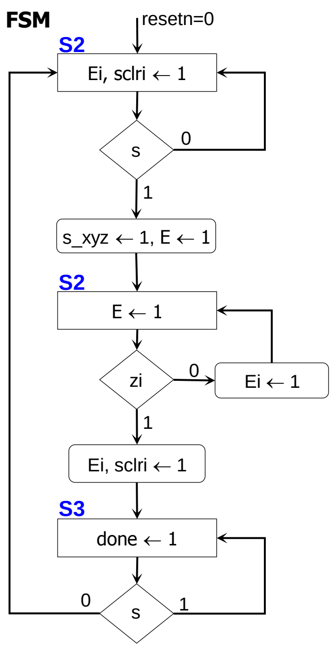
- **Angles**: They are represented in the format $[16\ 14]$. Units: radians. Pre-compute the values and store them in an LUT.

- **Barrel shifters**: Use the VHDL code `mybarrelshifter.vhd` with `mode="ARITHMETIC"` (signed data), `N=20`, `SW=4`, `dir='1'`.



- **Control**: This circuit includes a State Machine that controls the iteration index i, as well as the internal signals.





SIMULATION

- **First testbench**: Simulate the circuit for the following cases. You can use $A_n = 1.6468$. Convert the real numbers to the signed FX format $[16\ 14]$. For each case, verify that x_{16}, y_{16}, z_{16} reach the proper values.
  - ✓ Rotation Mode: $x_0 = 0, y_0 = 1/A_n, z_0 = \pi/6$.
  - ✓ Rotation Mode: $x_0 = 0, y_0 = 1/A_n, z_0 = -\pi/3$.
  - ✓ Vectoring Mode: $x_0 = y_0 = 0.8, z_0 = 0$
  - ✓ Vectoring Mode: $x_0 = 0.5, y_0 = 1, z_0 = 0$

- **Second Testbench**: Simulate the circuit reading input values (x_0, y_0, z_0) from input text files and writing output values (x_{16}, y_{16}, z_{16}) on an output text file. Your testbench must:
  - ✓ Read input values (x_0, y_0, z_0) from two input text files:
    - `in_benchR.txt`: Data for Rotation Mode testing.
      20 data points (x_0, y_0, z_0). Data format: $[16\ 14]$. Each line per data point written as hexadecimals: `|x0|y0|z0|`.
      Data set: $x_0 = 0, y_0 = 1/A_n, z_0 = -\pi/2\ to\ \pi/2$. z_0: 20 equally-spaced values between $-\pi/2\ to\ \pi/2$.
      With this data set in the rotation mode, note that $x_{16} \rightarrow -sin(z_0)$, $y_{16} \rightarrow cos(z_0)$.

    - `in_benchV.txt`: Data for Vectoring Mode testing.
      20 data points (x_0, y_0, z_0). Data format: $[16\ 14]$. Each line per data point written as hexadecimals: `|x0|y0|z0|`.
      Data set: $x_0 = 0.0\ to\ 0.5, y_0 = 1, z_0 = 0$. x_0: 20 equally-spaced values between $0.0\ to\ 0.5$.
      With this data set in the vectoring mode, note that $x_{16} \rightarrow A_n\sqrt{x_0^2 + y_0^2}, z_{16} \rightarrow atan(y_0/x_0)$.

  - ✓ Write output results (x_{16}, y_{16}, z_{16}) on `out_bench.txt`. 40 data points (20 from the rotation mode and 20 from the vectoring mode). Data format: $[16\ 14]$. Each line per data point written as hexadecimals: `|x16|y16|z16|`.

  - ✓ Vivado tips:
    - Make sure that the input text files are loaded as simulation sources.
    - The output text file should appear in `sim/sim_1/behav`.
    - To represent data as fixed-point numbers, use `Radix` → `Real Settings` in the Vivado simulator window.

- **VIVADO DESIGN FLOW FOR FPGAs:**
  - ✓ Create a new Vivado Project. Select the **XCA100T-1CSG324 Artix-7 FPGA** device.
  - ✓ Write the VHDL for the given circuit. Synthesize your circuit. (Run Synthesis).
  - ✓ Perform Functional Simulation (Run Simulation → Run Behavioral Simulation). **Demonstrate this to your TA.**

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

TA signature: ____________________________ **Date:** __________________________