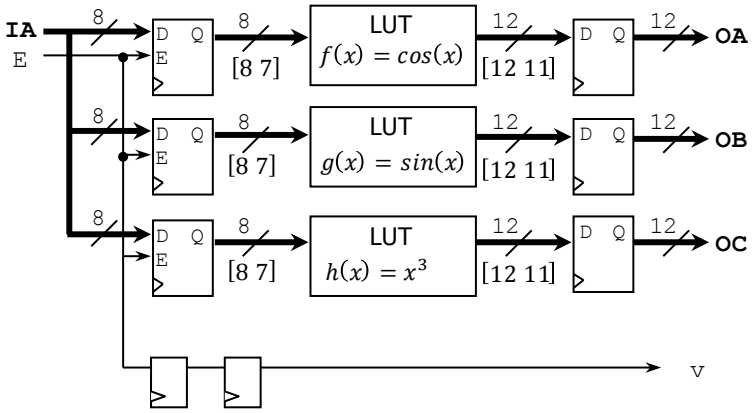
Homework 3

(Due date: March 21st @ 7:30 pm)
Presentation and clarity are very important! Show your procedure!

PROBLEM 1 (15 PTS)

- LUT approach: We want to implement the following system.
- The purpose of this exercise is to explore how to rapidly fill up LUT values and verify the correct operation. Here, you are asked not to write VHDL code, but rather to setup the parameters for it, synthesize, and simulate.

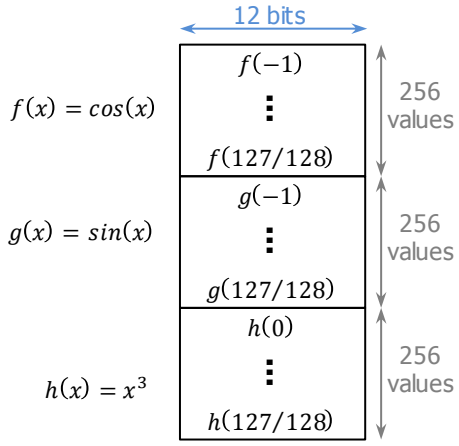
- The figure shows three 3 LUT 8-to-12, where each LUT holds the pre-computed results of 3 functions:
  - ✓ Input format: [8 7] (signed). Input data range: $[-1,1)$.
  - ✓ Output format: [12 11] (signed). Output data range: $[-1,1)$



- The VHDL code and testbench for this system can be found here.

```
test.vhd
   LUT_group.vhd → File that includes all the LUTs.
      LUT_NItoNO.vhd → File that implement one LUT.
   dffe.vhd

atb_test_sim.vhd → Testbench
```



PROCEDURE

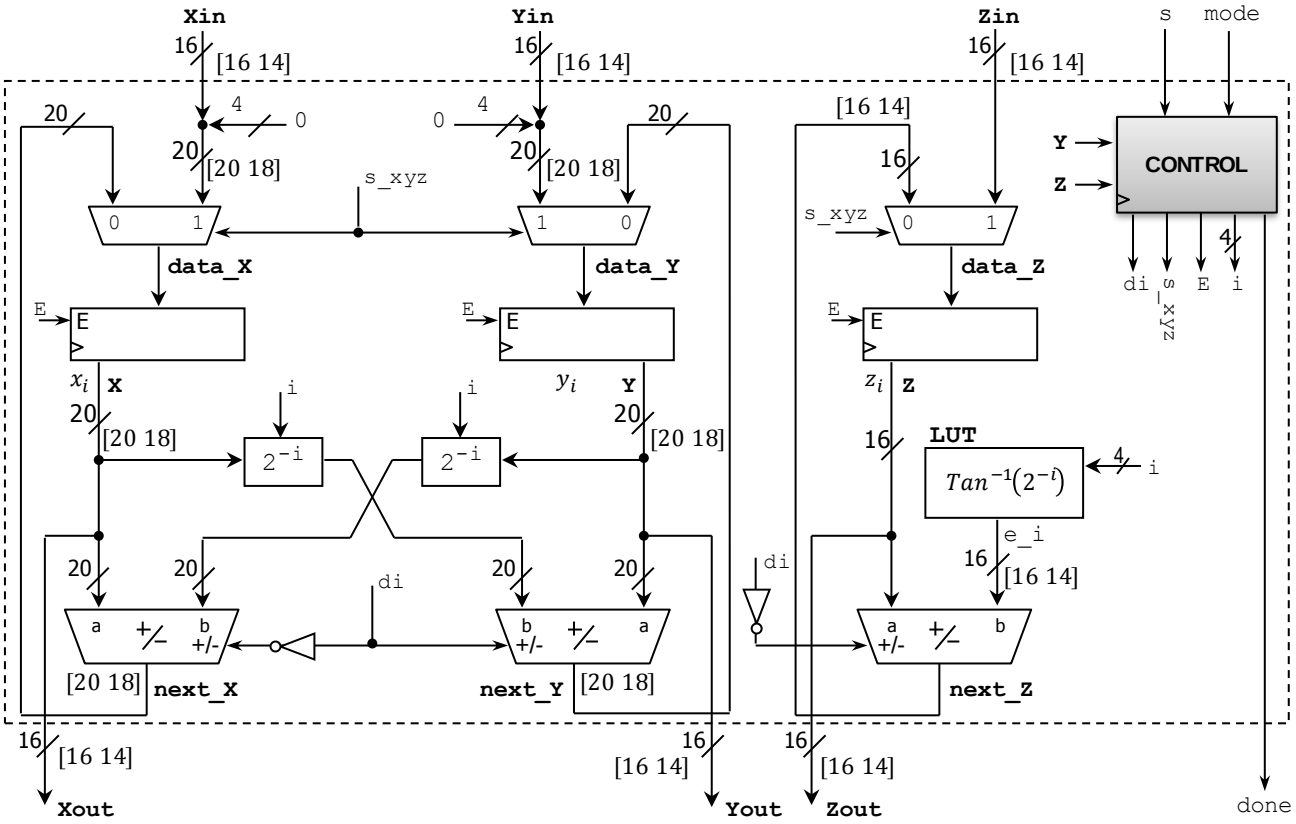
- Select the proper parameters to implement the circuit shown. These need to be indicated in test.vhd and in atb_test_sim.vhd.
- Generate the text file (LUT_values8to12.txt) that contains the pre-computed values. The filename is a parameter of LUT_group.vhd. The text file is divided as follows: the first 256 entries for the first function, the second 256 entries for the second function, and the third 256 entries for the third function. An 'L' separator is included between groups of 256 entries.
  You can use the provided MATLAB script (LUTvalGen8to12.m) to generate this file. This script requires the FX converter.

- Create a Vivado project and synthesize your circuit.
- Perform Functional Simulation:
  - ✓ The provided testbench atb_test_sim.vhd will generate all possible input cases (from 00000000 to 11111111) and write the output results in a text file. The testbench writes three 12-bit words per line (256 lines), where each 12-bit word represents the output of a different function.
  - ✓ Simulate the circuit until all the 256 input cases are processed. To verify the correct operation of your circuit, compare the text file generated by the Simulation with the input text file you created for Synthesis.

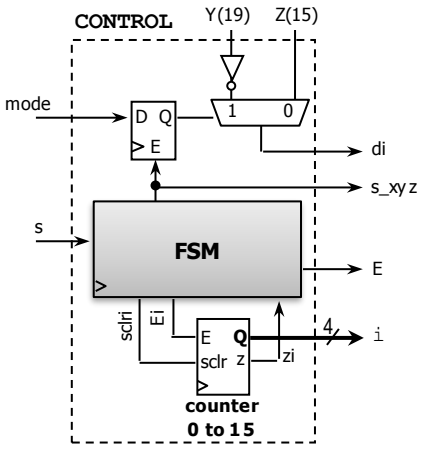
- Upload (as a .zip file) the following files to Moodle (an assignment will be created). DO NOT submit the whole Vivado project.
  - ✓ VHDL code, VHDL testbench: These files will be modified with the proper VHDL parameters.
  - ✓ Input text file (pre-computed function values) and output text file (results).

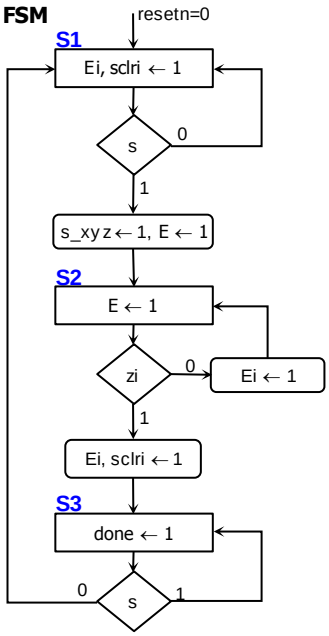
PROBLEM 2 (65 PTS)

- Design (write the VHDL code) for the iterative Circular CORDIC FX architecture with 16 iterations. $i = 0, 1, 2, 3, \ldots 15$. x_0, y_0, z_0: initial conditions. $mode$ = '0' → Rotation Mode. $mode$ = '1' → Vectoring Mode. (35 pts)
- **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 file `mybarrelshift_gen.vhd` with `SHIFTTYPE="ARITHMETIC"` (signed data), `N=20`, `SW=4`, `dir='1'`.



- **Control**: This circuit controls the iteration index i, as well as the internal signals:





SIMULATION (Functional)

- A Circular CORDIC MATLAB/Octave model is available. Make sure to use the 'Basic CORDIC'. This model can be useful to verify the hardware output data as well as to generate input data and LUT angles represented in Fixed-Point arithmetic. The .zip file contains the following files:
  - ✓ run_examples_cordic.m (this file only works in MATLAB), cordic_circular.m, get_scalefactor.m.
  - ✓ my_dec2fx.m, my_fx2dec.m, my_bitcmp.m: This Fixed-Point to decimal converter is helpful to convert the LUT angles and input data to their Fixed-Point representation (binary).

- **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. (10 pts)
  - ✓ 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. (20 pts). Your testbench must:

  - ✓ Read input values (x_0, y_0, z_0) from two input text files (provided):

    - `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 $x_{16} \rightarrow A_n\sqrt{x_0^2 + y_0^2}, z_{16} \rightarrow atan(y_0/x_0)$.

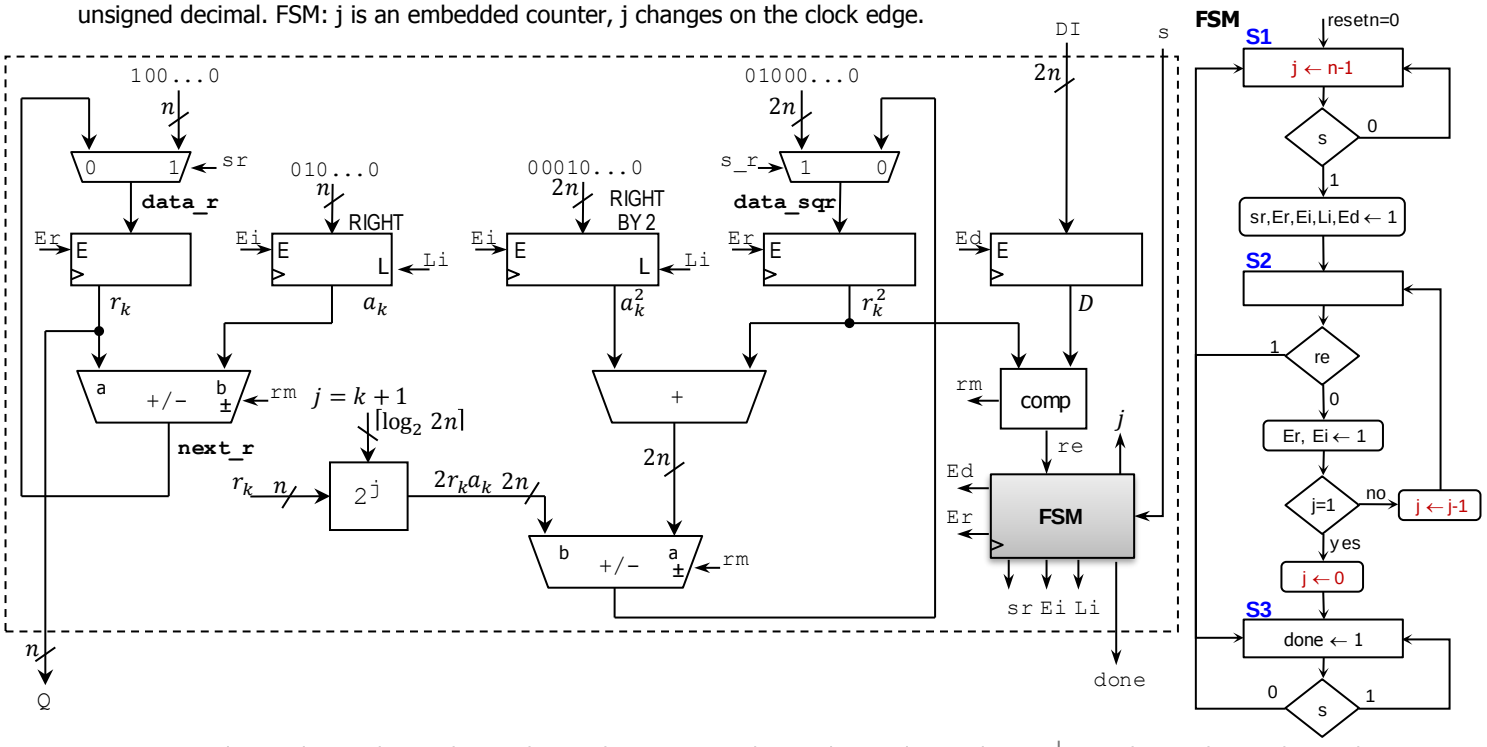
  - ✓ Write output results (x_{16}, y_{16}, z_{16}) on `out_bench.txt`. Data format: [16 14], each line per data point written as hexadecimals: `|x16|y16|z16|`. The output text file should have 40 data points (20 from the rotation mode and 20 from the vectoring mode).

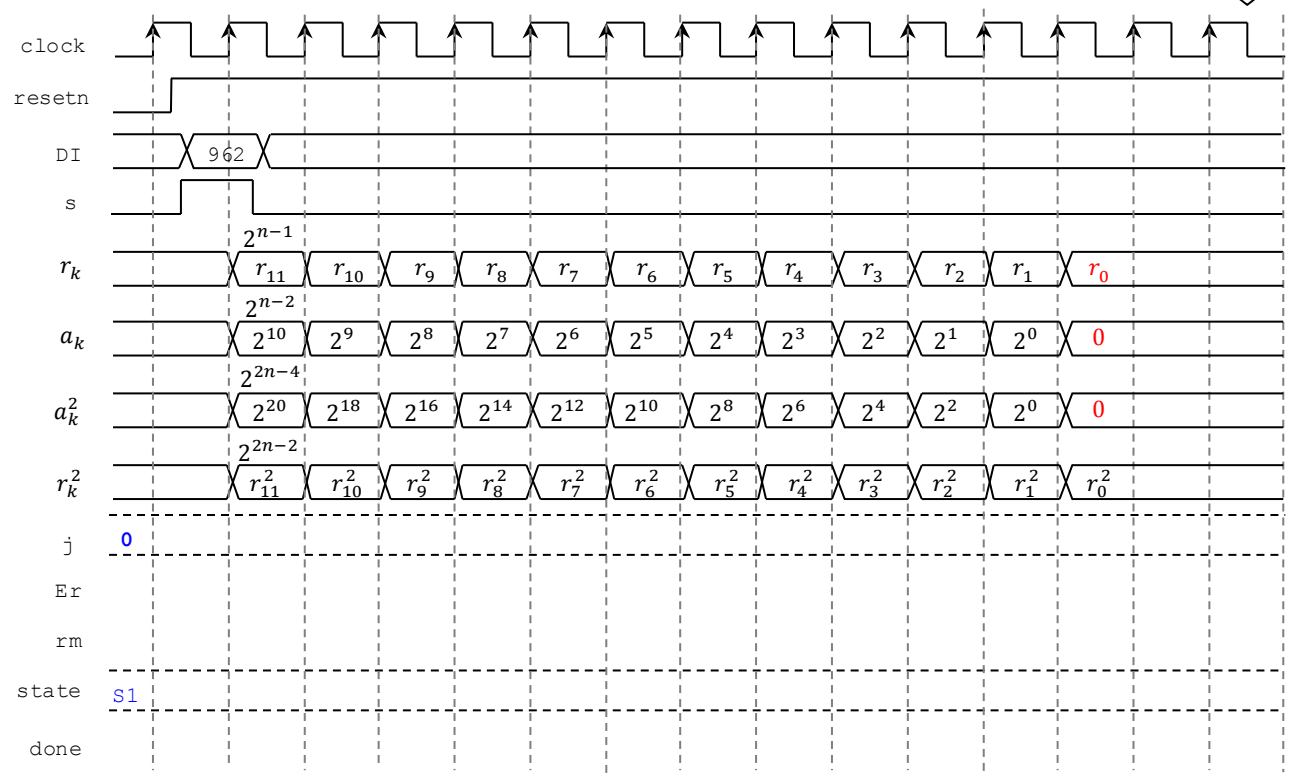
  - ✓ Vivado tips:
    - Make sure that the input text files are loaded as simulation sources.
    - The output text file should appear in `sim/sim_1/behav`.
    - To verify that the output results are correct, you need to represent data as fixed-point numbers. Use `Radix` → `Real Settings` in the Vivado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

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

PROBLEM 3 (10 PTS)

- Complete the timing diagram of the following circuit, which computes integer square root using a binary search approach. $n = 12$. Note that $rm = 1\ if\ r_k^2 > D, else\ 0$, $re = 1\ if\ r_k^2 = D, else\ 0$. Shift registers: serial input is '0'. The value of D is an unsigned decimal. FSM: j is an embedded counter, j changes on the clock edge.





r_{11}	r_{10}	r_9	r_8	r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0

PROBLEM 4 (10 PTS)

- Attach a printout of your Initial Project Report (no more than a page). This report should contain the project title, a brief project description, and the current status of the project, including a block diagram of your system. Use the provided template (`Final Project – Report Template.docx`).