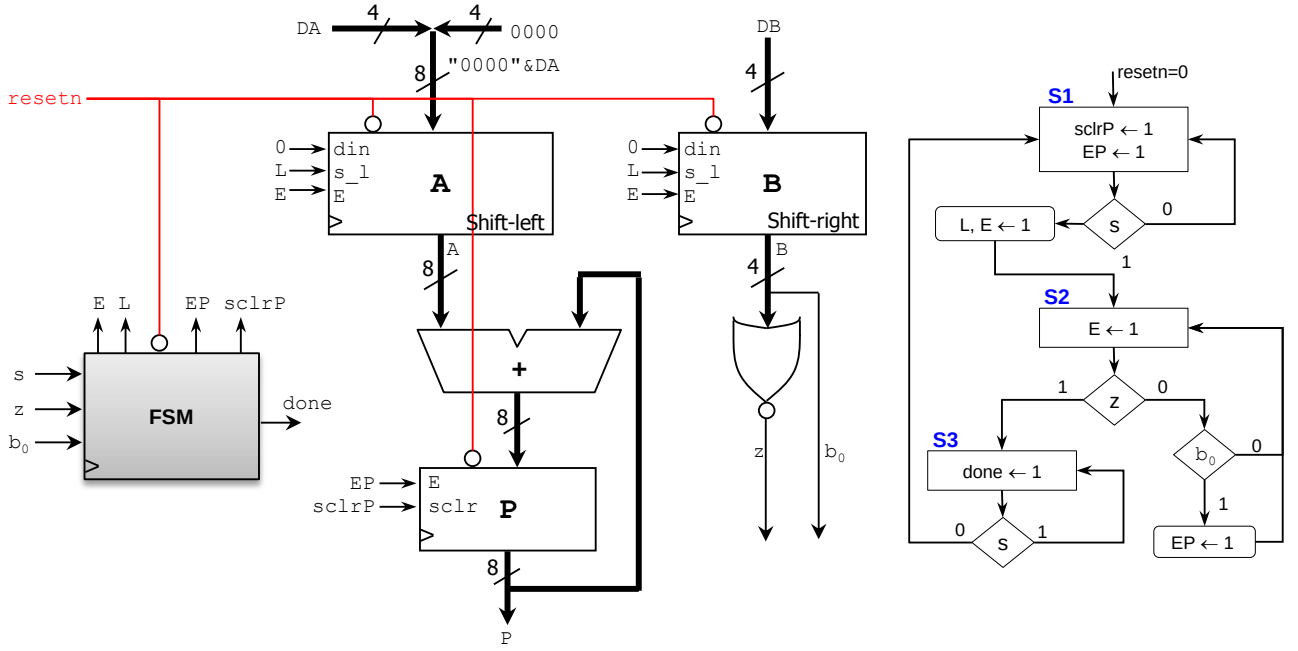
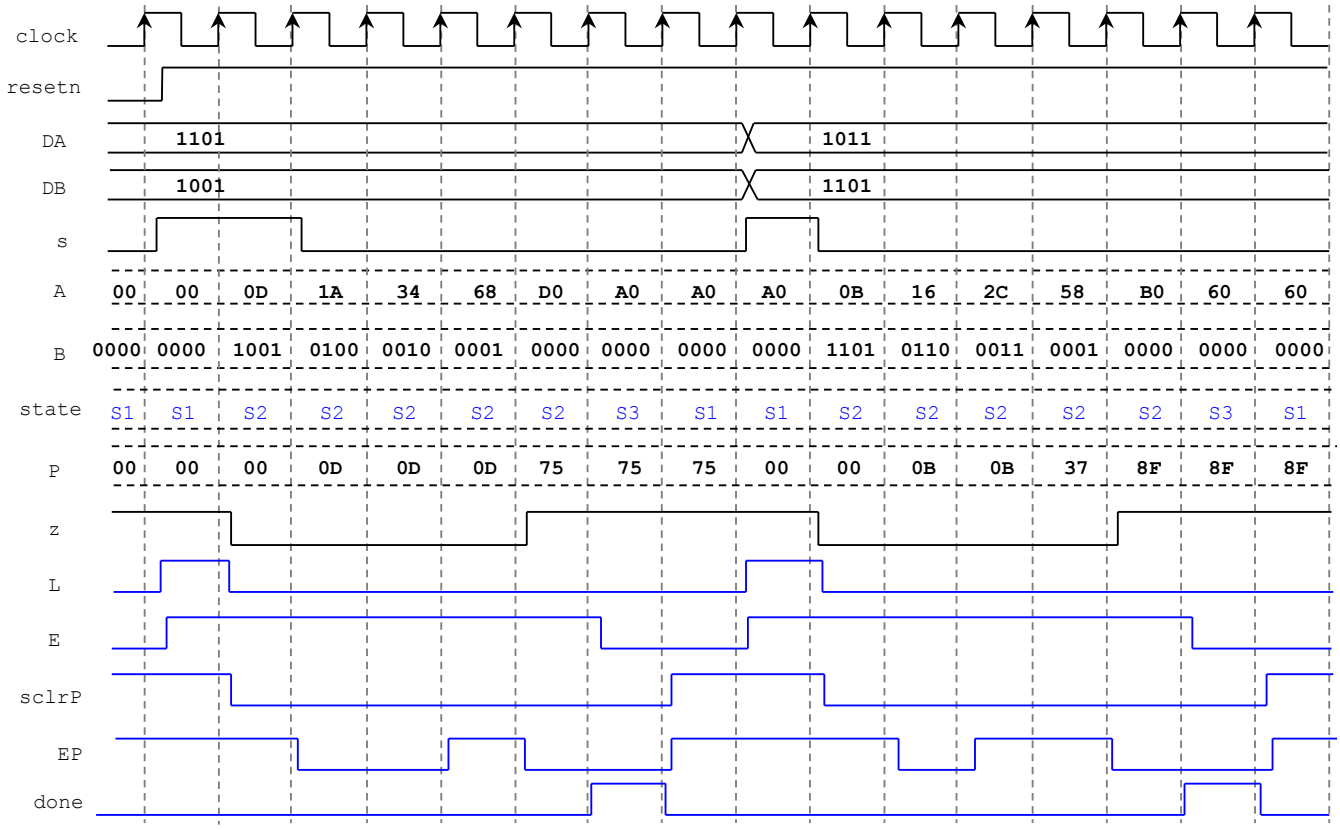
Solutions - Homework 4

(Due date: April 6th @ 5:30 pm)
Presentation and clarity are very important! Show your procedure!

PROBLEM 1 (15 PTS)

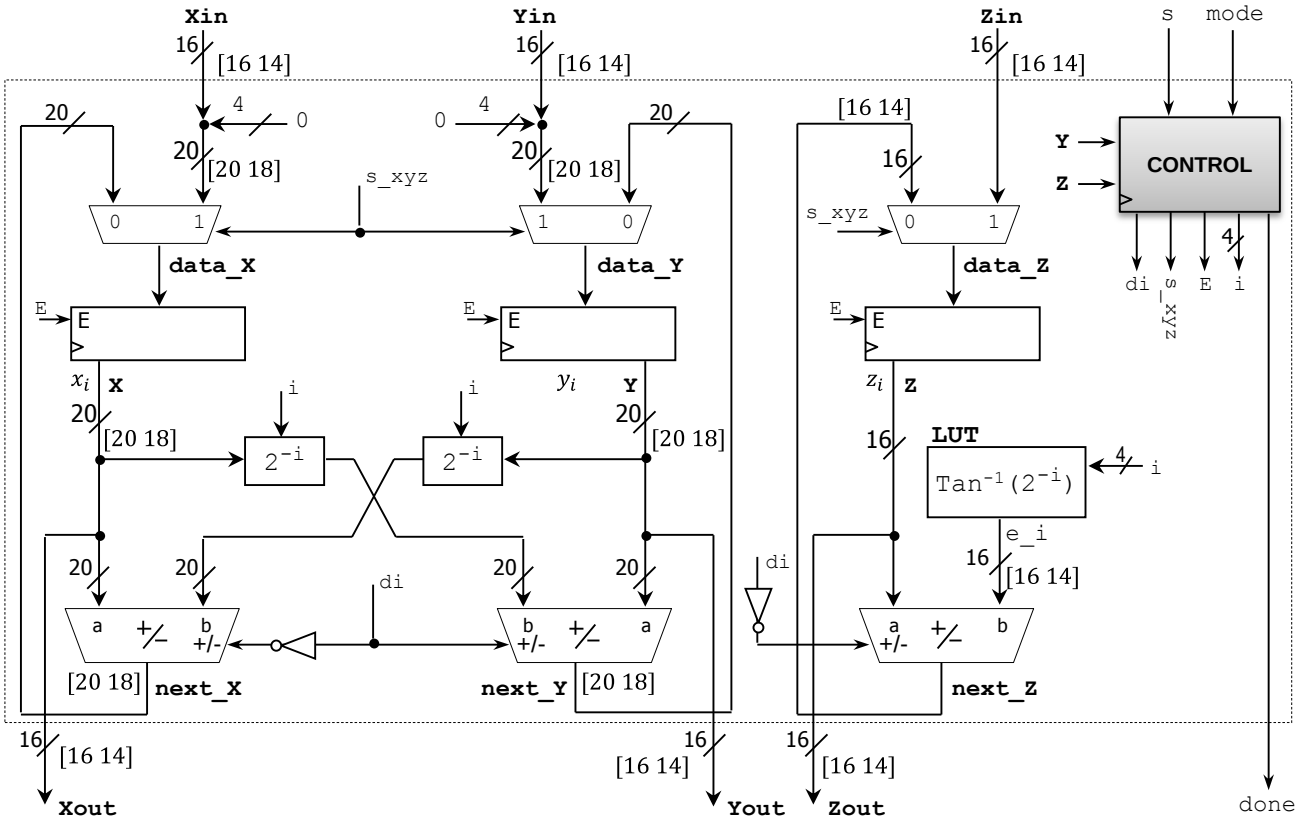
- Complete the following timing diagram (A and P are specified as hexadecimals) of the following Iterative unsigned multiplier. The circuit includes an FSM (in ASM form) and a datapath circuit.
  Register (for P): $sclr$: synchronous clear. Here, if $sclr = E = 1$, the register contents are initialized to 0.
  Parallel access shift registers (for A and B): If $E = 1$: $s\_l = 1 \rightarrow$ Load, $s\_l = 0 \rightarrow$ Shift



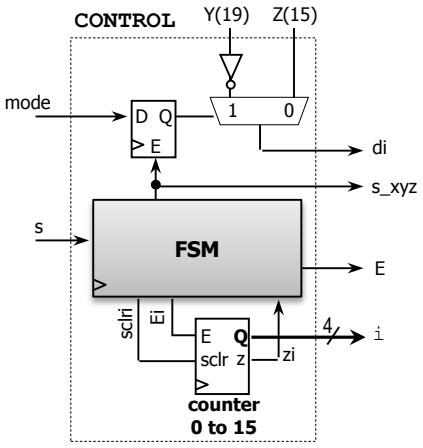


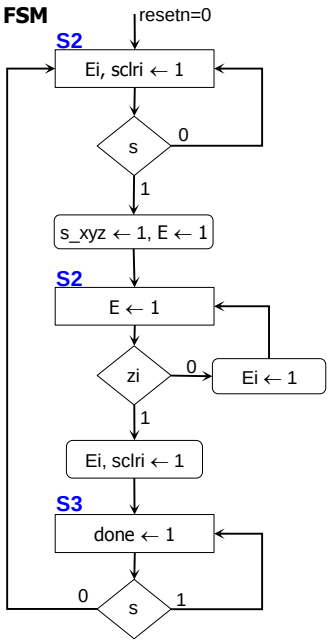
PROBLEM 2 (70 PTS)

- Design 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. (40 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.
- **Barrel shifters**: Use the VHDL code `mybarrelshifter.vhd` with `mode="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)

- **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:

    - `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.

- **For this Problem 2, you can work in teams of up to two (2) students. Only one Moodle submission per team, make sure to indicate who you worked with in your Homework 4 assignment.**

Output text file (`out_bench.txt`): They were obtained a very precise A_n value. To verify your values, just make sure that they approximate the real values of the functions that the CORDIC algorithm approximates to.

```
3FFFFFFCFFFD | 6964FFFF6483
3F1F0A8B0000 | 696E000062D7
3C8814C60001 | 698A00006129
38471E780000 | 69B800005F75
3281274E0001 | 69F9FFFF5DD1
2B5A2F130002 | 6A4DFFFF5C2B
22FF3594FFFF | 6AB200005A7F
19B23A9C0000 | 6B2A000058DB
0FB53E0A0003 | 6BB40000573B
05463FC7FFFE | 6C4EFFFF55A1
FAB93FC70002 | 6CFB0000540F
F04A3E0AFFFD | 6DB80000527D
E64A3A9B0000 | 6E86FFFF50F7
DCFF3594FFFF | 6F6300004F6D
D4A52F13FFFE | 7051FFFF4DED
CD7E274EFFFF | 714E00004C7B
C7B71E750000 | 725AFFFF4B03
C37714C6FFFF | 73750000499B
C0DF0A880000 | 749E0000482F
C000FFFAFFFD | 75D5FFFF46DB
```