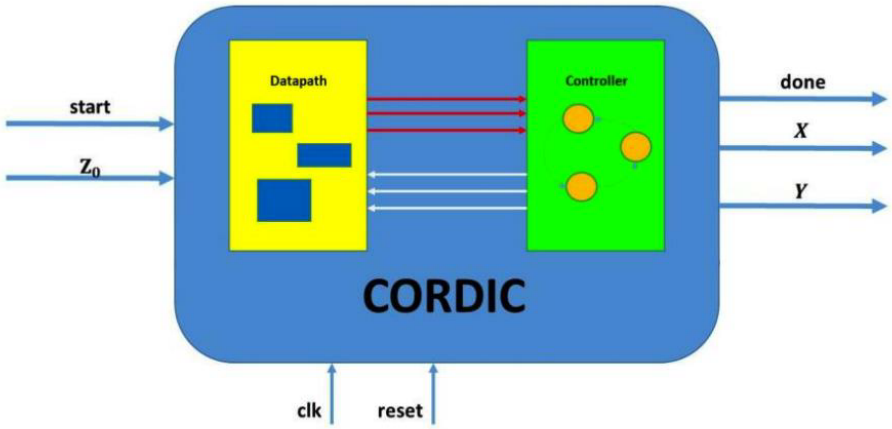
UIC COLLEGE OF ENGINEERING
UNIVERSITY OF ILLINOIS AT CHICAGO
Department of Electrical and Computer Engineering

Advanced Digital System Design
ECE 594-Online Offering-Spring 2018
Final Exam- Due date: May 11

1. In this Question you are to design a CORDIC coprocessor to calculate sine and cosine of a given angle at RT-level. The IO lines of the top-level module of your design must look like the figure bellow in which
Z0 (input angle), X (cosine) and Y(sine) are 8-bit lines.



Notice that:
1- You must follow RTL design flow (data path & controller).
3- The RTL elements you may use are: adder, shift-register, register, counter and comparator.
4- For θn from Eq (4) bellow, use a lookup table in which the values of θn are stored.
5- Handle “start”, “done” and “reset” signals properly.

You **Do NOT** need to Write the VHDL Code for this problem. Showing the RTL Design is enough.

The CORDIC Algorithm of Sine and Cosine:

CORDIC is an efficient algorithm of implementation for a variety of mathematical functions like trigonometric functions. This algorithm uses basic operations e.g. addition, subtraction, comparison, shifting and table lookup.

In the following we study how to use CORIDC in rotation mode to calculate sine and cosine of an angle, assuming that the angle is given in radian and is represented in a fixed point format. The matrix representation of this rotations is shown in Eq (1).

$$\begin{bmatrix} X_j \\ Y_j \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \qquad \text{Eq (1)}$$

The rotation can be performed in multiple steps. Each step can be shown as in Eq (2) or Eq (3):

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \qquad \text{Eq (2)}$$

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \cos\theta \begin{bmatrix} 1 & -\tan\theta \\ \tan\theta & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \qquad \text{Eq (3)}$$

If we use θ_n so that $\tan\theta$ equals a power of two, we can implement the multiplications of Eq (3) using bit shifting. Thus we use θ_n as:

$$\theta_n = \tan^{-1}\left(\frac{1}{2^n}\right) \qquad \text{Eq (4)}$$

In which the relation between the steps can be shown as:

$$\theta = \sum_n S_n \theta_n \quad S_n = +1, -1 \qquad \text{Eq (5)}$$

And thus we have:

$$\tan\theta = S_n 2^{-n} \qquad \text{Eq (6)}$$

Combining Eq (3) and Eq (6), we will have:

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \cos\theta \begin{bmatrix} 1 & -S_n 2^{-n} \\ S_n 2^{-n} & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \qquad \text{Eq (7)}$$

Since the total transfer function is derived from multiplication of all transfer functions, we can apply the coefficient of $\cos\theta$ at the final step only, as:

$$K = \frac{1}{P} = \prod_{n=0}^{\infty} \cos\left(\arctan\left(\frac{1}{2^n}\right)\right) \approx 0.607253 \qquad \text{Eq (8)}$$

Using the equations above, if we apply the cosθ coefficient at the final step, we can say that:

$$\begin{cases} X_{n+1} = X_n - S_n 2^{-n} Y_n \\ Y_{n+1} = Y_n - S_n 2^{-n} X_n \end{cases} \qquad \text{Eq (9)}$$

Now, we need an extra variable to represent the remaining portion of θ that has not been applied yet. We call this variable Z_n whose sign determines the value of S_n as:

$$Z_{n+1} = Z_n - S_n \theta_n \qquad \text{Eq (10)}$$

$$S_n = \begin{cases} -1 \;\; if \;\; Z_n < 0 \\ +1 \;\; if \;\; Z_n \geq 0 \end{cases} \qquad \text{Eq (11)}$$

We continue the steps of the algorithm until Z=0, then we have:

$$\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} P\left(x_{n-1}\cos\left(z_{n-1}\right) - y_{n-1}\sin\left(z_{n-1}\right)\right) \\ P\left(y_{n-1}\cos\left(z_{n-1}\right) - x_{n-1}\sin\left(z_{n-1}\right)\right) \\ 0 \end{bmatrix} \qquad \text{Eq (12)}$$

For which, at each step we calculate x_n, y_n and z_n as:

$$\begin{cases} X_{n+1} = X_n - S_n 2^{-n} Y_n \\ Y_{n+1} = Y_n - S_n 2^{-n} X_n \\ Z_{n+1} = Z_n - S_n \theta_n \end{cases} \qquad \text{Eq (13)}$$

With the initial values of:

$$\begin{cases} X_0 = K = 0.60725 \\ Y_0 = 0 \\ Z_0 = \theta \end{cases} \qquad \text{Eq (14)}$$

Using these initial values, the Eq (12) is simplified to Eq (15):

$$\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \end{bmatrix} \qquad \text{Eq (15)}$$

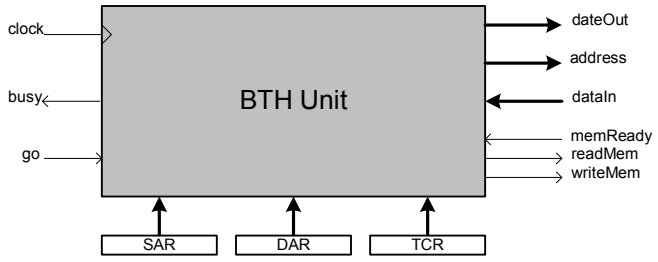
2. An NoC (Network on Chip) switch is responsible for routing data between several processing elements that are connected together through the switch. Inside the switch, blocks of data are transferred from buffers belonging to one port to another buffer for a different port. Buffers can be implemented by contiguous memories and each port uses a section of such a memory.

A block transfer hardware (BTH) that copies a section of a switch buffer memory to another section is part of a switch hardware that is being designed here. When this hardware begins its operation, it assumes its source address register (SAR) its destination address register (DAR) and transfer count register (TCR) have been properly loaded and are available to use. The contents of these registers must remain unchanged when the BTH hardware has completed its operation.

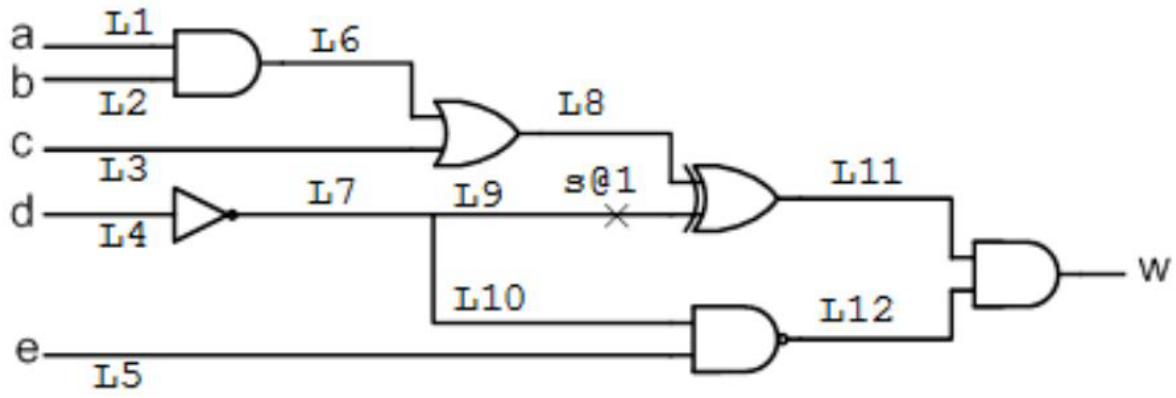
When a clocked synchronous *go* pulse appears on the input of BTH, the BTH starts transferring the specified block of memory from one address (i.e., SAR) to another (i.e., DAR), and it continues this byte-by-byte transfer TCR number of times. During this operation, the transfer hardware (BTH) asserts its *busy* signal preventing another *go* signal to be issued. Upon completion, the *busy* signal is deasserted, and BTH waits for another block transfer.

The buffer memory is a single port slow memory. It can only be written or read from by one process. In addition, it requires handshaking for read and write operations. This memory has 8-bit *dataIn*, *dataOut*, and *address* busses. In addition it has *readMem*, *writeMem*, outputs and a *memReady* output. To read or write the memory, *readMem* or *writeMem* are asserted and stay asserted until the completion of the operation. During this time, proper data and or address values must be provided for the memory until. When the memory completes its requested operation (read or write), it issues *memReady* and keeps it asserted until *readMem* or *witeMem* are removed by the BTH unit. While *memReady* is 1, in case of a read operation, memory holds valid data on its *dataOut* output and keeps *memReady* asserted.

You are to design the BTH unit with proper interfaces from the part of an NoC switch that provides information about the blocks, and a proper memory interface. A) Show the complete datapath of this circuit. Include initial registers, your own registers, and busses. B) List all necessary control signals. And specify their functionalities. You may use signal names to indicate their functions. C) Show the complete state diagram of the controller. D) Write VHDL code for the design. E) Write a comprehensive testbench and verify the circuit of Part D.



3. Find a test for L9-sa1. Show all steps of Fault Activation and Fault Propagation.



Deliverables:

1. All VHDL codes
2. A complete report illustrating all your observations, simulation results, and required waveforms.