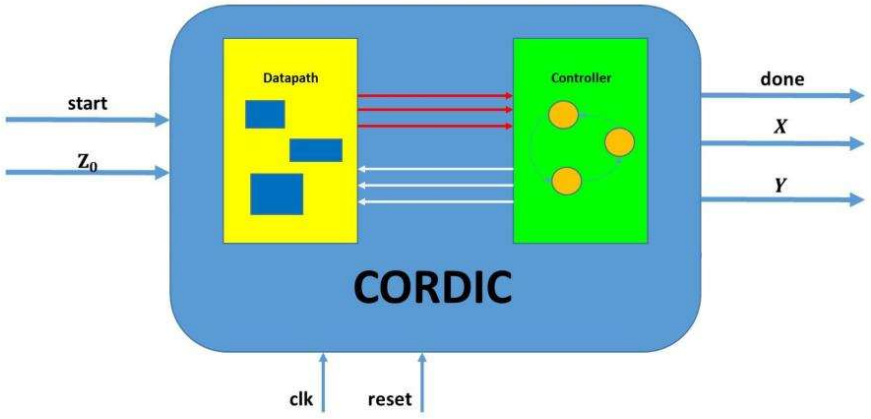
Homework #2 CORDIC

The Problem:

In this assignment you are to implement 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 Z_0 (input angle), X (cosine) and Y(sine) are 8-bit lines.



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 tanθ 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 cosθ 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):

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

Hints:

A) Input/output range and accuracy.

In your design, you get an input angle and calculate the corresponding Sine and Cosine values of it. To simplify your task, you may use the following format for the 8-bit input and output lines.

1- *Input Angle:*An 8-bit unsigned vale that varies between 0 and 255. This range of inputs represents 0-2π radian, or 0-360 degrees. You can calculate the accuracy of your module as:

$$Accuracy_{angle} = \frac{2\pi - 0}{255} \cong 0.025 \ \ rad$$

$$Accuracy_{angle} = \frac{360 - 0}{255} \cong 1.41 \ \ \deg$$

This means that increasing the input value by 1, translates to increasing the input angle by 0.025 rad, or 1.41 degrees. If higher word-sizes were used, the accuracy would have increased.

2- Output Sine and Cosine Values: An 8-bit line with a signed-digit representation. This way, you have the MSB to represent the sign and 7 bits to represent the value. Similar to the accuracy of the angle, in this case we have:

$$Accuracy_{\sin/\cos} = \frac{1-0}{127} \cong 0.008$$

Thus, in an 8-bit representation, you can present sine and cosine values from -1 to +1 with an accuracy of 0.008 units per bit.

B) Preparation of the data

First you need to determine the trigonometric quarter corresponding to the input angle. This helps you to determine the sign of Sine and Cosine values.

After this step, calculate the smallest angle (between 0-π/2) that has the same absolute values of Sine and Cosine, as the original angle. For example, if the original angle is 127 degrees, you are looking for 127-90=37 degrees, or if the original angle is 225 degrees, you are looking for 225-180=45 degrees. A simple combinational hardware can do this for you.

After finding the corresponding angle between 0 and π/2, you may start the CORDIC algorithm. When the output values are calculated, you just need to set the signs of outputs according to the trigonometric quarter of the original angle.

Also:

1- You don't have to, but it is better to design a generic module. In a generic module, you do not put constraints on the size of input and output lines. Use the GENERIC keyword to determine the size of input and output lines. In this case, your system must work correctly for various values of the generic parameter. Larger generic values propose higher rates of accuracy in the results.
2- Provide a proper way of evaluation in your testbench. For instance, in the simplest way, you may externally calculate the desired output values so that you could compare them with the generated outputs.
3- You may use the input angle in degrees instead of radian. But make sure to fill your lookup table with proper values.

Notice that:

1- You must follow RTL design flow (data path & controller).
2- Your design must be synthesizable.
3- For θ_n from Eq (4), use a lookup table in which the values of θ_n are stored.
4- Handle "start", "done" and "reset" signals properly.
5- In your testbench, consider various scenarios and a number of system-cycles for various inputs.

The exact delivery time will be announced later, but there is definitely time until the end of the semester exam.

good luck

Behrooz Abdoli abdoli.behrooz66@gmail.com