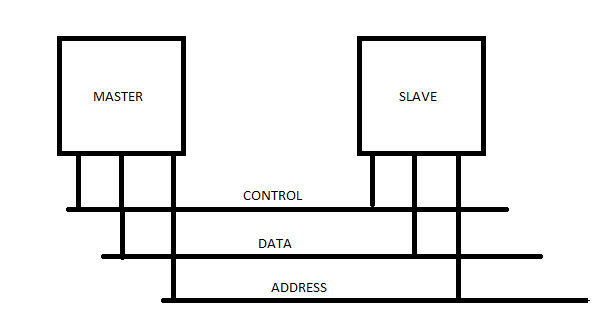
Student: Kinman Tai

ECE 338 –Assignment 2

CORDIC System Report

* INTRODUCTION

We are asked to design a CORDIC system that has a bus structure and it provides the data and accepts results. The data provides a single number, which is a value represented as a 32 bit fixed point value, scaled to be a number between +/- 1. The answer (it will be available in 3 registers) is including the terminal Z value, the final X value and the final Y value. The CORDIC system contains a bus transaction which can be understood as having communication within two machines. One is the master, another is the slave. Students are only responsible to design the slave part of the CORDIC system, and the master part is provided as the testbench in VHDL code. The communication between the master and slave is similar to what we call as “handshaking”. They respond to each other by sending the information of addresses, data, read, acknowledge, request etc. The following graph illustrates the bus transaction concept of the CORDIC system:



Figure

* DESIGN APPROCH

The slave machine handles the algorithm of the CORDIC system. The X value is the approximate sine value of initial Z, and the Y value is the approximate cosine value of initial Z. Here are the three equations we have to implement.

CORDIC EQUATION

……………………………………. (1)

……………………………… (2)

………………………………. (3)

I scaled for i from 0 to 35, to be the 36 bits fix point value and convert them to hexadecimal. The result is the following:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i=0 | 6487ED511 | i=8 | 007FFFD55 | i=16 | 00007FFFF | i=24 | 0000007FF | i=32 | 000000008 |
| 1 | 3B58CE0AC | 9 | 003FFFFAA | 17 | 00003FFFF | 25 | 0000003FF | 33 | 000000004 |
| 2 | 1F5B75F92 | 10 | 001FFFFF5 | 18 | 00001FFFF | 26 | 0000001FF | 34 | 000000002 |
| 3 | 0FEADD4D5 | 11 | 000FFFFFE | 19 | 00000FFFF | 27 | 000000100 | 35 | 000000001 |
| 4 | 07FD56EDC | 12 | 0007FFFFF | 20 | 000007FFF | 28 | 000000080 |  |  |
| 5 | 03FFAAB77 | 13 | 0003FFFFF | 21 | 000003FFF | 29 | 000000040 |  |  |
| 6 | 01FFF555B | 14 | 0001FFFFF | 22 | 000001FFF | 30 | 000000020 |  |  |
| 7 | 00FFFEAAA | 15 | 0000FFFFF | 23 | 000000FFF | 31 | 000000010 |  |  |

The three equations of CORDIC system can be represented as the block diagram:

Here is the state diagram of the slave CORDIC system:

Here is the state diagram of the master CORDIC system:

* EXPLANATION of DESIGN of the SLAVE STATE DIAGRAM

At the initial state (state 0), the slave will wait for signals from the master. Although there are 3 different addresses that the master can assert, I distinguish them as two different paths to go, which is either write or read. Therefore, if the master sends the Read/Write address of Z and READ low and REQUEST high, then state 0 will go to state1. If the master sends the read address of X or read address of Y, then state 0 will go to state 5.

Let’s focus on the writing path first. It is because a heavy part of the algorithm of CORDIC equations will be done by this path. State 1 will always go to State 2, and the initial values of X, Y, Z and alpha (0) are loaded at state 1. Also, the algorithm needs a counter being as an reference or a direction guide to determine the shifting amount of the values, the correct X, Y, Z, alpha to use, and most important to indicate the iteration. The count of iteration would determine the state going path. Thus, counter is assign to be zero in state 1.

At state 2, the counter is incremented, and the three CORDIC equations will be implemented. Then state 2 will go to state 3. In state 3, it will check the counter number if the counter is not equal or greater than 35 then state 3 will go back to state 2. The purpose of doing backward is because we want the iteration goes from 0 to 35. This is similar to the concept of *for* loop.

At state 4, the final result of X, Y and Z are loaded here. And so, it forces the slave to send the acknowledge signal to let the master knows that the Z result is written and it is ready to read the X, Y and Z value. If you tie state 5 to state 4 up, the read activity will be processed. State 4 will go back to state 0.

Of course, the signals REQUEST, RESET, will change the direction of the state going to the other state.

* RESULT of CordicData