ELEX 7660 : Digital System Design
2017 Winter Term

Assignment 2

Due Monday, March 20. Submit your assignment using the appropriate dropbox on the course web site. Assignments submitted after the solutions are made available will be given a mark of zero.

Question 1

BigSemiCo Ltd. has hired you to help with verification this summer. Your manager has given you a file, `cordic.v`[1], with some old-style Verilog code. This file defines a module, `cordic`, that can be used to compute trigonometric functions.

Your manager is thinking of using the module in an important project where it will be used to compute sines of angles. She wants to know if the module works properly so she's asked you to write a testbench to see how accurate the results are.

The module `cordic` is defined as follows:

```
`define XY_BITS    16
`define THETA_BITS 16
`define CORDIC_1 17'd19896        // CORDIC inverse

module cordic (
  input wire clk,
  input wire rst,
  input wire signed [`XY_BITS:0]    x_i,
  input wire signed [`XY_BITS:0]    y_i,
  input wire signed [`THETA_BITS:0] theta_i,

  output wire signed [`XY_BITS:0]    x_o,
  output wire signed [`XY_BITS:0]    y_o,
  output wire signed [`THETA_BITS:0] theta_o
);
```

By now you can probably figure out the purpose of `clk` and `rst`. As described in the file's header comments, the module can be used to compute sines by setting:

```
y_i = 0;
x_i = `CORDIC_1
theta_i = the input angle
```

and then after 16 clock cycles (the pipeline delay) the output will be:

```
y_o = sin(theta_i)
```

[1]This is in `asg2.zip` on the course web site and actually comes from github (via opencores.org and the author's site). comes with documentation and a testbench. But for this assignment you must write your own.

The input angle, `theta_i`, is specified in radians and valid values are between 0 and $\pi/2$. The angle and the input x and y values are in the following fixed-point format (from the documentation):

> *... the format U(1,15) where bit 16 is the sign bit, bit 15 is the whole number part and bits [14:0] are the fractional parts.*

This means you can convert to/from positive real values to this "U(1,15)" format by dividing/multiplying by 32768.

Unfortunately, Verilog does not have a built-in sin function so you must generate the test vectors – the input angle and the corresponding sine value – using your favorite programming language and then read the test vectors into your testbench (*don't* include the test vectors in your testbench code as the original author did – that would be too easy).

For example, if your favorite programming language were Matlab (or Octave) you might write:

```
a=0:pi/2/20:pi/2;
fprintf(fopen('asg2.tv','w'),'%05x %05x\n',...
        [a*32768;sin(a)*32768]);
```

to generate 20 test vectors between 0 and $\pi/2$.

Write a self-checking testbench that reads values from your test vector file and applies them to the module you're testing. Your testbench should compare the `cordic` module's sin output to the expected value and `$display()` the angle, the expected sin value, the actual computed sin value and the difference. Keep track of the largest difference found and `$display()` it at the end.

To read the test vectors you can use the `$readmemh()` system tasks used in the last assignment to read all the values at once to a single array or `$fopen` and `$fread()` to read one test vector at a time. Here are some examples of the latter, assuming `fd` and `n` are `int`s:

```
...
fd = $fopen("asg2.tv","r") ;
...
for ( n=0 ; $fscanf(fd,"%x %x",angle[n],sine[n]) == 2 ; n++ ) ;
```

Note that due to the “pipelined” design there is a delay of 16 clock cycles between changes to the input and the corresponding value appearing on the output. You can start your “checker” process 16 clock cycles after the start of the input or you can process one test vector at a time by leaving a delay equal to at least the pipeline delay between changing the input and checking the output.

Submit your testbench code and the *complete* simulation transcript showing everything from the simulator start to end messages as well as your `$display()` output to the dropbox on the course web site.