Simulation

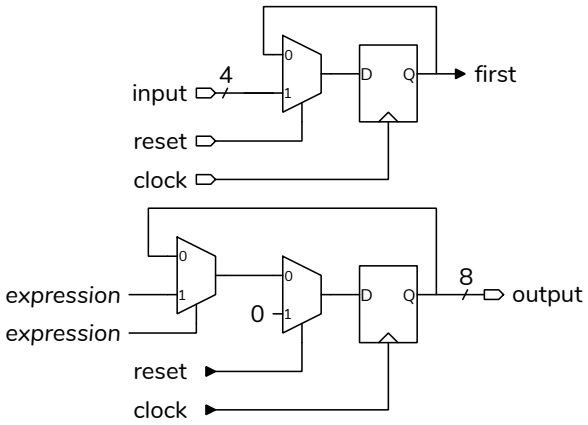
Revision 2: Corrected example simulation output.

Introduction

In this lab you will design and simulate a circuit that counts the number of digits that exceed the first digit.

Requirements

The incomplete block diagram below shows a circuit that loads the input into one register and clears a second register when reset is asserted. Otherwise it updates the second register with a count of the number of input values that have exceed the value loaded into the first register. This count is the circuit's output.



Your first test vector should assert reset and have an input equal to the average of the digits of your BCIT ID; rounded down. For example, if your ID were A00123456 then the average of the digits is $\lfloor (0+0+1+2+3+4+5+6)/8 \rfloor = 2$ then the input test vectors might start with:

```
1, 2, 0
```

For subsequent test vectors the input should be digits of your BCIT ID. For the above ID:

```
...
0, 0, 0
0, 0, 0
0, 1, 0
...
0, 5, 3
0, 6, 4
...
```

The last test vector should contain an error and so an error message will be printed. For example:

```
...
0, 0, 99
```

In the sample simulation code given in the lecture notes the inputs from each test vector are applied and the simulation waits for the next falling edge (i.e. after a rising edge) before testing the DUT output. Thus the expected output of each test vector corresponds to the expected output for that input.

Procedure

Write a module that implements the circuit described above and a testbench that applies test vectors read from a `.csv` file. You should be able to use the example testbench in the lecture notes with a few appropriate modifications. Remember to follow the course coding guidelines, including adding the appropriate comment at the beginning of each file.

Your testbench should print:

- Each test vector as it is read from the file.
- The output of your DUT
- An error message if there are mismatches between the expected and actual outputs. See the example testbench for an example.

You can create the `.csv` file containing the test vectors using a text editor such as Notepad. You can also use a spreadsheet and export the test vectors to a `.csv` file[1].

Follow the procedure in the *Software Installation and Use* document and the video on the course web site to create a simulation project, add the file(s)[2] with

[1] But don't use a UTF-8 (Unicode) output encoding.

[2] You may put the DUT and testbench modules in different files or in the same file.

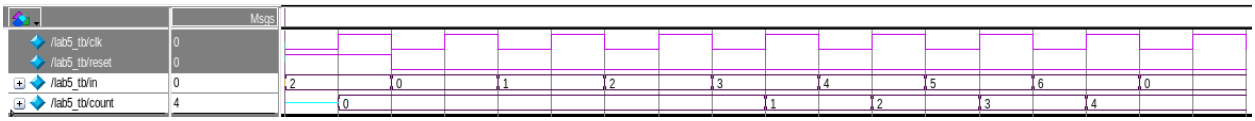


Figure 1: Simulation results.

your modules to the project and compile them. Copy the test vector file to the project folder. Add the clock, reset, input, and output signals to the Wave window. Run the simulation. The Transcript window should show any messages generated by the testbench and the Wave window should show the signal waveforms.

Report

Submit a PDF file to the appropriate Assignment folder that includes the following:

- Listing(s) of your DUT and testbench modules.
- A screen capture of the simulation waveforms similar to that shown above. Set the display format for the output waveform to unsigned decimal[3].
- A screen capture of the Transcript window showing the messages generated by running the simulation. For example:

```
# run -all
#    reset      in   count  count_
#        1       2       0       0
#        0       0       0       0
#        0       1       0       0
#        0       2       0       0
#        0       3       1       1
#        0       4       2       2
#        0       5       3       3
#        0       6       4       4
#        0       0      99       4 *** Error
# ** Note: $stop    : lab5_tb.sv(40)
```

[3]Right-click on the waveform name and select Radix / Unsigned