Lab 1: Finite State Machines + Datapaths

EEL 6935 – Reconfigurable Computing 2

Objective:

The objective of this lab is to use a finite state machine integrated with a datapath to calculate the number of asserted bits in a given input using several different SystemVerilog models.

Required tools and parts:

Quartus2 software package, ModelSim-Altera Starter Edition

Lab requirements:

1. Study the following pseudo-code to make sure you understand the basic algorithm for efficiently calculating the number of asserted bits in a given input.

```
// inputs: go, in (WIDTH bits)
// outputs: out (clog2(WIDTH+1) bits), done

// reset values (add any others that you might need)
out = 0;
done = 0;

while(1) {

   // wait for go to start circuit
   while (go == 0);
   done = 0;
   count = 0;

   // store input in register
   n_r = in;

   // main algorithm
   while(n_r != 0) {
         n_r = n_r & (n_r - 1);
         count ++;
   }

   // assign output and assert done
   output = count;
   done = 1;
}
```

1-process FSMD

2. Using the provided file asserted_bit_count.sv, fill in the asserted_bit_count_fsmd_1p module with a design that uses the 1-process FSMD model. **Done should remain asserted until the application is started again, which is represented by go being asserted while done is asserted signal followed by a 1. Done should be cleared on the cycle after go is asserted.**

   Use the provided testbench to test your module. Note that the testbench only tests a single module. Therefore, make sure the following line:

   ```
   asserted_bit_count_fsmd_1p #(.WIDTH(WIDTH)) DUT (.*);
   ```

   is uncommented, and that all other modules are commented out. For the other parts of the lab, you will specify a different module.

2-process FSMD

3. Repeat the same process as the previous step, but using a 2-process FSMD in the asserted_bit_count_fsmd_2p module**.**

   Use the provided testbench to test your module. Note that the testbench only tests a single module. Therefore, make sure the following line:

   ```
   asserted_bit_count_fsmd_2p #(.WIDTH(WIDTH)) DUT (.*);
   ```

   is uncommented, and that all other modules are commented out.

Datapath

4. In this step, you will create your own datapath to implement the algorithm. Create your own datapath module in a new file (datapath.sv). Implement it however you like (behaviorally, structurally, with an number of resources, etc.). Draw a schematic of the datapath and include it in the report described below.

FSM

5. Create a 2-process finite-state machine in a new file (fsm.sv) that works with your custom datapath to implement the algorithm.

FSM+D

6. Connect your FSM and datapath together in the asserted_bit_count_fsm_plus_d module of asserted_bit_count.sv.

   Use the provided testbench to test your module. Note that the testbench only tests a single module. Therefore, make sure the following line:

   ```
   asserted_bit_count_fsm_plus_d #(.WIDTH(WIDTH)) DUT (.*);
   ```

   is uncommented, and that all other modules are commented out.

Report

7. Create a report that illustrates
   a. A simulation screenshot of each of the modules with the provided testbench. Make sure the screenshot shows the number of passed and failed tests.
   b. A screenshot of each module synthesized in Quartus proving there are no warnings (you can modify the module at the bottom of the asserted_bit_count.sv file to easily change the top level).
   c. Your datapath schematic.

Turn in instructions:

To submit your lab please create a folder that is named lab1. Inside that folder include:

- Your modified asserted_bit_count.sv file.
- Your datapath and fsm files (datapath.sv and fsm.sv).
- Your report (report.pdf)
- README.txt with your group member names. If any problems occurred that I should be aware of for grading, include them here.