Lab 1: Is Perfect Square Calculator

EEL 5721/4720 – Reconfigurable Computing

Objective:

The objective of this lab is to create an FSMD and FSM+D that determines if an integer is a perfect square. It is also your responsibility to build your own testbench.

Required tools and parts:

Vivado, ModelSim-Altera Starter Edition (or equivalent simulator)

Lab requirements:

1. For this lab, the entity will determine if a provided number (n) is a perfect square. First, study the following pseudo-code to make sure you understand the basic algorithm.

```
// Inputs: go, n (WIDTH bits)
// Outputs: output, done

// Reset values (add any others that you might need)
output = 0; done = 0;

while(1){

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

   // Register the n input (should happen in the cycle go is asserted)
   n_r = n;

   // Initialization
   k_r = 1;
   square_r = 0;

   while (square_r < n_r) {
     square_r = k_r * k_r;
     k_r++;
   }

   if (square_r == n_r)
     output = 1;
   else
     output = 0;

   // should remained asserted until circuit is started again
   done = 1;
}
```

2. IMPORTANT: The design has the following timing requirements:
   - Done should be cleared 1 cycle after the assertion of go.
   - Upon completion, done should remain asserted indefinitely until go is asserted again.
   - k_r should be WIDTH bits
   - square_r should be 2*WIDTH bits

FSMD

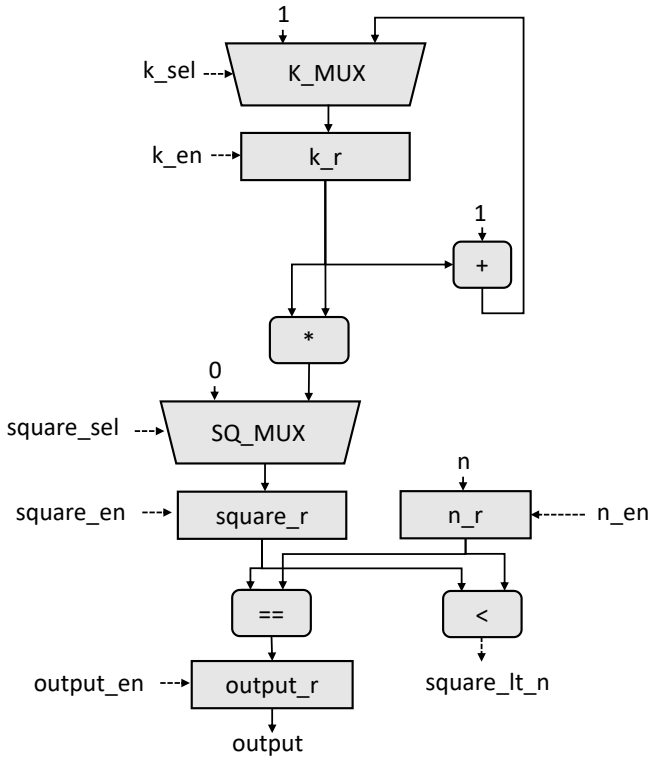
3. Create a 1-process FSMD that implements the code shown above. Add the code to the FSMD architecture within is_perfect_square.vhd.

4. Feel free to use the provided is_perfect_square_crv_tb.sv to test your design. You'll be writing your own testbench in a later step.

5. Synthesize your FSMD in Vivado (for any FPGA) and take a screenshot showing no synthesis warnings. ***Make sure that the default_arch architecture of is_perfect_square.vhd is using the FSMD architecture. Save the screenshot to synthesis_fsmd.jpg***.

FSM+D

6. Create the following datapath by filling in the datapath.vhd file. You can create the datapath structurally or behaviorally, but it must synthesize to this same structure. ***If you create a structural architecture, make sure to add your extra entities to datapath.vhd to ensure that all submissions have the same files.***



7. Create a 2-process FSM controller that controls the datapath from the previous step to implement the required functionality from the pseudo-code in part 1. Your controller should be implemented in the fsm.vhd file.

8. Fill in the FSM_D architecture in is_perfect_square.vhd to instantiate and connect the datapath and controller.

9. Synthesize your FSM_D architecture in Vivado (for any FPGA) and take a screenshot showing no synthesis warnings. ***Make sure that the default_arch architecture of is_perfect_sqaure.vhd is using the FSM_D architecture. Save the screenshot to synthesis_fsm_d.jpg***.

10. Important: you will likely get these warnings (or something similar) for the FSM+D. These warnings are fine, but only if they occur at time 0. They are caused by a value that isn't '0' or '1' reaching an

arithmetic operation. This is often unavoidable at time 0 because certain signals haven't been initialized yet depending on how the simulation orders the execution of processes.

*# ** Warning: NUMERIC_STD."<=": metavalue detected, returning FALSE*
 # Time: 0 ns Iteration: 0 Instance: /is_perfect_square_crv_tb/DUT/TOP/datapath_inst
*# ** Warning: NUMERIC_STD."=": metavalue detected, returning FALSE*
 # Time: 0 ns Iteration: 0 Instance /is_perfect_square_crv_tb/DUT/TOP/datapath_inst

Testbench

11. Create a testbench in is_perfect_square_tb.vhd that tests at least 3 different values of *n*. I would highly suggest testing a large number of input values to ensure that your design is working. I will be testing it with a very thorough testbench, which will likely catch errors your testbench might miss.

12. Your testbench should be used to test both the FSMD and FSM_D architectures. You can do this either by manually changing the architecture that is used in the testbench, or by changing the default_arch architecture in is_perfect_square.vhd. For this lab, do *not* create multiple testbench files.

13. Take a screenshot of your testbench running the FSMD and save it to a file testbench_fsmd.jpg.

14. Take a screenshot of your testbench running the FSM+D and save it to a file testbench_fsm_d.jpg.

15. An additional testbench is provided in is_perfect_square_crv_tb.vhd that uses a strategy called constrained-random verification. It performs numerous random tests, while also verifying the timing of the done signal. I would highly recommend using it in addition to your own testbench because the graders will use something similar.

Turn in instructions:

Submit the following to Canvas as a single zip file where the file name is your UFID. e.g., for 12345678 your submission should be a single zip called 12345678.zip. Do not include any extra folders, do not change the name of any files, do not use another compression format, etc. This specific structure is intended to help automate grading.

- is_perfect_square.vhd
- fsm.vhd
- datapath.vhd
- is_perfect_square_tb.vhd
- synthesis_fsmd.jpg
- synthesis_fsm_d.jpg
- testbench_fsmd.jpg
- testbench_fsm_d.jpg
- README.txt - If any problems occurred that I should be aware of for grading, include them here.