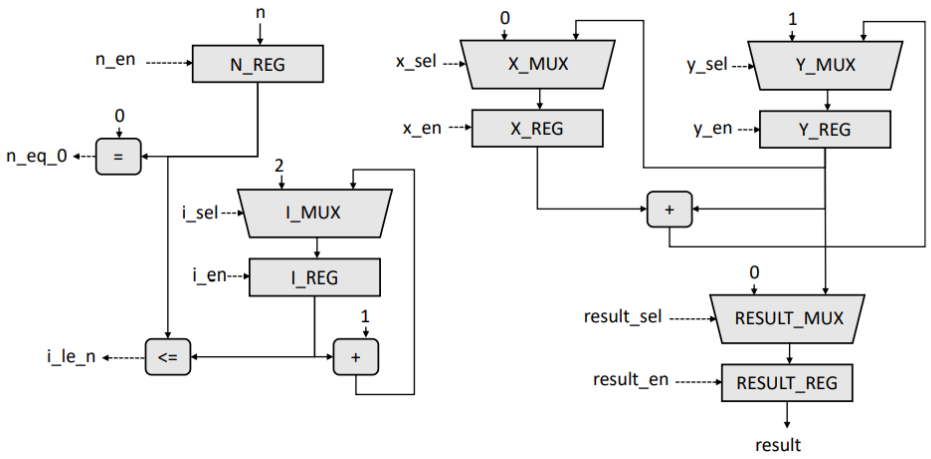
EEL 4712C: Digital Design
Spring 2024

LAB 03: Finite State Machines
Updates: 1 2/9/24
(50 Points)

Lab Objectives

This lab will add on two different types of controllers to the previous labs datapath. In lab 2, you should have created a datapath that synthesized to the schematic shown below. For this lab, you will write a one process and a two process finite state machine (FSM) to highlight the differences between how they perform at runtime.

This lab REQUIRES your Lab 2 to be fully functional. If your datapath does not work properly, this lab will not work either.



Lab Requirements

Part 1: One Process FSM

Knowing the signals needed to control the Fibonacci datapath (i.e. n_eq_0, n_en, x_sel, result_en, etc.), design a one process finite state machine that can correctly control the datapath. The one process model should, as the name implies, only contain one process with a sensitivity list containing a clock and a reset.

This controller should split up the execution of the Fibonacci code, provided again below for reference, into multiple states where computation is performed. In each state, the controller should

enable to datapath components necessary to perform the computation desired. These signals are listed in the bottom of the document and the signal names need to match exactly.

It is advised that you create a state transition diagram on paper before trying to implement anything in code. This may be checked by the PIs before you continue to the rest of the lab. Both parts of the lab should use the same state transition diagram. There are virtually hundreds of ways to create this diagram with different numbers of states. Include states that handle the done signals and allow the system to restart if certain inputs are loaded in.

```
//Inputs: go, N
//Outputs: output, done

//Reset any values needed before execution
output = 0;
done = 0;

while (1) {

        //Wait for go to be asserted
        while(go == 0);
        done = 0;

        //Register the input value when go is asserted
        reg_n = N;

        //Set Fibonacci initial conditions
        reg_a = 0;
        reg_b = 1;

        //Compute Nth Fibonacci number
        if (n == 0)
                output = x_r;
        else {
                for (int i = 2; i <= reg_n; i++) {
                        int temp = reg_a + reg_b;
                        reg_a = reg_b;
                        reg_b = temp;
                }
                output = reg_b;
        }

        //Keep done asserted until execution restarts
        done = 1;
}
```

Part 2: Two Process FSM

Unlike the one process finite state machine that relies on a single clocked process, the two process implementation should rely on a sequential (clocked) process and a combinatorial process. In one process the next state should be defined and in the other the current state should be updated. This finite state machine will require more fine tuning than the one process model, but is capable of updating the control signals on the state change rather than a clock cycle after the state has changed.

Part 3: Demo

Using the provided top_level.vhd and the datapath created in the previous lab, you will run the entire design on your FPGA first with the single process finite state machine and then with the two process finite state machine.

Demo Procedure

1. Using the provided top_level.vhd, the 7-segment display entity you created in Lab 0, and the datapath from the previous lab, test each of your finite state machines and understand the differences between the outputs.
2. In Quartus, assign pins to the inputs (switches & buttons) and outputs (LEDs) such that the correct outputs are displayed on the 7-segment display.
3. Once the everything is working, show a TA the correct output for Fib(11) = 89 using the single process finite state machine. Then reprogram the FPGA and show the result again with the two process finite state machine

Deliverables & Submission Guidelines

- Summarize your work in a report and highlight the key differences between a one and two process finite state machine. Template for lab report is provide on Canvas
- Zip your lab solutions and report in a file name <Your_UFID>.zip with a folder structure as shown below.
    - Make sure to use the file names shown below in your submission
    - If your lab report is not yet finished, you can submit it up to a week after the lab demo due date, but the source code must still be submitted on the demo due date

```
Your_UFID/
├── report.pdf
├── P1/
│   ├── Datapath.vhd
│   ├── fsm1.vhd
│   ├── simulation_fsm1.vhd
│   └── <Names_of_additional_files>.vhd
├── P2/
│   ├── simulation_fsm2.vhd
│   └── fsm2.vhd
```

Provided Entities

Datapath

```
entity fsm<1 or 2> is
    port (
        clk  : in  std_logic;
        rst  : in  std_logic;
        go   : in  std_logic;
        done : out std_logic;

        n_en       : out std_logic;
        result_en  : out std_logic;
        result_sel : out std_logic;
        x_en       : out std_logic;
        x_sel      : out std_logic;
        y_en       : out std_logic;
```

```
        y_sel      : out std_logic;
        i_en       : out std_logic;
        i_sel      : out std_logic;

        n_eq_0 : in std_logic;
        i_le_n : in std_logic
        );
end fsm;
```