EEL 4712C: Digital Design
Spring 2024

LAB 02: Basic RTL Components & Adders
(50 Points)

Lab Objectives

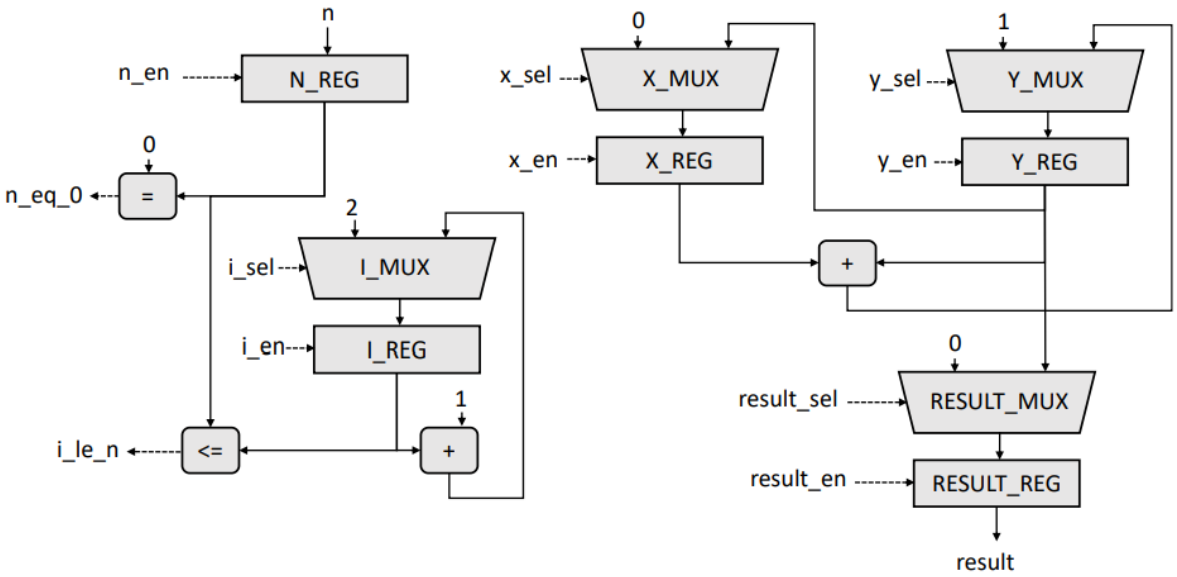
The objective of this lab is to design a datapath capable of calculating the Nth Fibonacci number along with a testbench to verify its correctness. The algorithm for computing the Nth Fibonacci number is shown below; make sure you read and understand this code before continuing on to writing the datapath or testbench.

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

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

while (1) {
        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 the Nth Fibonacci number
        if (reg_n == 0)
                output = reg_a;
        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;
}
```

Lab Requirements

Part 1: Datapath

To create the Fibonacci calculator datapath, you will need to implement the schematic shown below. While it is not required, it is strongly recommended to implement this datapath using a structural architecture to make debugging easier. This datapath and the algorithm it implements each assume that the first Fibonacci number is 0, the second is 1, the third is 1, etc. An entity for this datapath, including all necessary inputs and outputs, has been included at the bottom of this document.



Part 2: Testbench

With the datapath completed, it can now be tested before being used on the FPGA. Using testbench constructs learned in class such as signal assignment and 'wait for' statements, write a rudimentary testbench to model the flow of data in the datapath. Note that N should be fairly small (N <= 35) to prevent data width from exceeding 24 bits (maximum that can be displayed on the LEDs).

Part 3: Demo

Using the provided top_level.vhd file, you will run the created datapath on your FPGA. In order to properly control the datapath, an obfuscated controller has been provided in fsm.vhd. Include this file in your project so that it synthesizes with your design.

Demo Procedure

1. Using the provided top_level.vhd, the 7-segment display entity you created in Lab 0, and the provided fsm.vhdp synthesize your circuit.
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. Show your TA the RTL viewer showing the properly synthesized datapath.
4. Once the everything is working, show a TA the correct output for Fib(11) = 89.

Deliverables & Submission Guidelines

- Summarize your work in a report. 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

```
Your_UFID/
├── report.pdf
├── P1/
│   ├── Datapath.vhd
│   ├── fsm.vhd
│   └── <Names_of_additional_files>.vhd
├── P2/
│   ├── Datapath_tb.vhd
│   └── datapath_sim.jpg
```

Provided Entities

Datapath

```
entity datapath is
    port (
        clk    : in  std_logic;
        rst    : in  std_logic;
        n      : in  std_logic_vector(5 downto 0);
        result : out std_logic_vector(23 downto 0);

        n_en       :  in std_logic;
        result_en  :  in std_logic;
        result_sel :  in std_logic;
        x_en       : in std_logic;
        x_sel      : in std_logic;
        y_en       : in std_logic;
        y_sel      : in std_logic;
        i_en       : in std_logic;
        i_sel      : in std_logic;

        n_eq_0 : out std_logic;
```

```
        i_le_n : out std_logic
      );
end datapath;
```