Lab 6 – Arithmetic Unit

CS 2052 Computer Architecture

Dept. of Computer Science and Engineering, University of Moratuwa

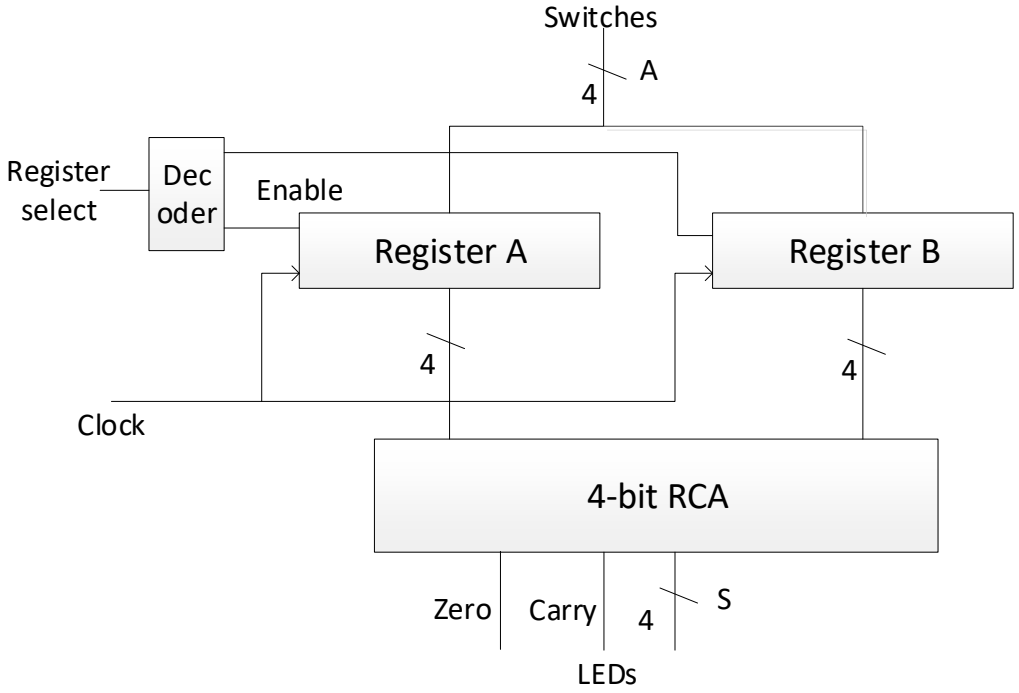
Learning Outcomes

In this lab, we will design a 4-bit arithmetic unit that can add 2 numbers stored in registers. After completing the lab, you will be able to:

- design and develop a 4-bit register
- design and develop a 4-bit arithmetic unit
- verify their functionality via simulation and on the development board

Introduction

Registers are used to store a set of bits inside a microprocessor. In this lab, we will design a 4-bit Arithmetic Unit (AU) that can add numbers stored in 2 different registers. High-level diagram of the AU is given below.



We will build a 4-bit register using D Flip Flops. Register also has an Enable input and a Clock input. 2 registers are then connected to our 4-bit Ripple Carry Adder (RCA) developed in Lab 3. This forms a simple AU.

Registers are loaded with a 4-bit binary value specified via 4 switches. Which register to load is determined by the input to the Enable pin of a register which is controlled via Register select. The output of the RCA is connected to a set of LEDs. Clock input is used to synchronize the behavior of the circuit.

Building the Circuits

Step 1: Building 4-bit Register.

Create a VHDL file and name is as **Reg**. Label the inputs as **D**, **En**, and **Clk**. Label the output as **Q**. Both **D** and **Q** should be 4-bit busses.

While we could build a D Flip Flop with En and Clk and then connect 4 of them to build the 4-bit register, we could use the following VHDL code to build a 4-bit register in one step:

```
process (Clk) begin
    if (rising_edge(Clk)) then      -- respond when clock rises
        if En = '1' then            -- Enable should be set
            Q <= D;
        end if;
    end if;
end process;
```

Step 2: Importing RCA.

Import HA, FA, and RCA VHDL files from Lab 3. For this, click on the **+** sign to add a new Design Source. Then select **Add File**. Locate the files from Lab 3 (file should be inside Lab3.srcs → sources_1 → new). Make sure you will be copying these files to your new project. Click on **Finish** button.

Step 3: Building 4-bit Arithmetic Unit.

Create a new VHDL file and name it as **AU**. Inputs should be labelled as **A** (4-bits), **RegSel**, and **Clk**, while the outputs should be labelled as **S** (4-bits), **Zero**, and **Carry**.

Build the circuit given in the above diagram. You need to derive the Boolean expressions for 1-to-2 decoder and Zero flag. You may also need some internal signals (e.g., directly connecting output from RCA to S may not give you an opportunity to calculate Zero flag. If you do so, you may see an error in VHDL syntax).

You will also need the **Slow_Clk** developed in Lab 5 to slow down the clock. Otherwise, while you change the **RegSel** from 0 to 1 or vice versa, both registers may ended up getting the same value or intermediate value may read as input while you set input A. Therefore, further slow-down the clock to 0.5 Hz (in Lab 5 we used 1 Hz).

Verify the functionality of the AU using the simulator. Use your index number for some of the input combinations.

Step 4: Connecting inputs and outputs.

Connect switches **SW0**-**SW3** as the **D** inputs and **SW15** as the **RegSel** input. Connect outputs **S** to LEDs **LD0**-**LD3**, **Carry** to **LD14, Zero** to **LD15**. Use BASYS 3's internal clock for **Clk**.

Step 5: Test on BASYS 3.

Generate the programming file (i.e., bitstream) and load it to the BASYS 3 board.

Change the switches on the board and verify the functionality of your circuit. Try different number combinations including ones that produce carry and zero.

Demonstrate the circuit to the instructor and get the Lab Completion Log singed.

No lab report is due for this lab.

Prepared By

- Dilum Bandara, PhD – Mar 20, 2014.
- Updated on Oct 17, 2018.