EE 3120: Digital Circuits Laboratory
LAB 3: SEQUENTIAL LOGIC ELEMENTS: FLIP-FLOPS & REGISTERS

Objectives:

The objectives of this laboratory assignment are:

- To understand and design flip-flops with synchronous and asynchronous inputs.
- To use D flip-flops to design and implement a universal shift register.
- To use address decoders and parallel-in, parallel-out registers to implement a register file suitable for a small microprocessor.

Laboratory Instructions

- Create the Verilog source file(s) for your designs before coming to the lab. You can use the Xilinx HDL editor or any text editor to create your files.
- Use the Xilinx software to create a new project in your directory on the C or H drive and then copy your Verilog (.v) to that new project directory. Remember to Add your Verilog file to this new project (see the EE 3120 Laboratory Tutorial for details).
- Perform functional simulation of your design and have it checked by the lab instructor or your TA before proceeding with the implementation.
- In case you modify your original Verilog source file, remember to update your copy too.
- If the circuit works as expected, implement it using the prototyping board assigned to you.
- Use switches and LEDs available on the board to test and demonstrate your circuit to the lab instructor or your TA.
- Before you leave the lab please remove any files or directories you created on your lab PC and leave your workplace at least as clean and tidy as you found it.

Pre-lab Work

- Complete your design and its Verilog implementation and bring your Verilog program on a floppy disk. You can use any text editor to create a Verilog file. If you have any problems with Verilog syntax and other pre-lab related issues, please resolve them with the instructor or your TA before coming to the lab. Your TA may not be able to help you with these issues during the lab session.
- At the end of the lab during the week starting on 03/21, submit a hardcopy of all your Verilog programs and design diagram (if any) to your TA or the lab instructor. This pre-lab work you submit will be graded as a part of this laboratory assignment. Incorrect or incomplete designs and Verilog programs will not receive full credit for pre-lab work.

Laboratory Report Instructions

Your report should be typed and prepared as per the guidelines given on the EE 3120 web

page.
For each design of this lab, submit the following:

- Documented listing of your Verilog source file(s) with appropriate pin assignments for the top level files in the same or separate file(s).
- For any hierarchical design, a block diagram showing all the modules used along with the interconnections.
- Simulation waveforms: Label the waveforms to indicate proper operation of your circuits.

DESIGN PROBLEMS

Using the Xilinx CAD tools, design, test and demonstrate logic circuit(s) which implement the following sequential logic elements. Your circuits should be as small as possible.

1. Use combinational logic elements to design a D flip-flop with active low synchronous clear input which will force the flip-flop state to 0 synchronous with the clock. Design and simulate this module. Implementation on a prototyping board is not strictly required.
2. Frequency Divider: Use the above D flip-flops and as few additional gates as possible to design a frequency divider circuit which will divide the input clock frequency by 1, 2, 4, 8. Besides the clock and clear inputs, the divider receives a 2-bit control input, a_1a_0 which specifies the divide-by amount ($a_1a_0 = 00$: divide by 1, $a_1a_0 = 01$: divide-by 2, $a_1a_0 = 10$: divide-by 4, $a_1a_0 = 11$: divide-by 8). Design, simulate and demonstrate the frequency divider operation using the prototyping board.
3. Register File: Design and implement a synchronous 8 x 3 Register File (i.e., eight, 3-bit registers) with dual 3-bit data buses ($Di=di_2di_1di_0$ for input and $Do=do_2do_1do_0$ for output) and a 3-bit address bus ($A=a_2a_1a_0$). Besides address and data signals, it also has one bit control input, RW. The register file performs the following operations:
    - READ (if RW=0): Contents of register Reg-A (where, A is the address on the address bus $A=a_2a_1a_0$) are put out (read) on the output data bus Do (i.e., read operation: Do <= Reg-A)
    - WRITE (if RW=1): Contents on the input data bus are stored (written) in register RA (i.e., write operation: Reg-A <= Di).