Experiment 6

Sequential Circuit Design

In this lab, we are going to design a sequential circuit which is going to display eight different characters on our FPGA at the same time. Nexys A7 board includes a 100MHz clock signal which is connected to pin `E3`. For this lab, we will use D flip-flops, with reset and write enable, as storage elements.

Tasks

You are required to build a circuit to display different characters (0 to F) on different seven-segment display. Fig. 6.1 shows the modular diagram for this task.

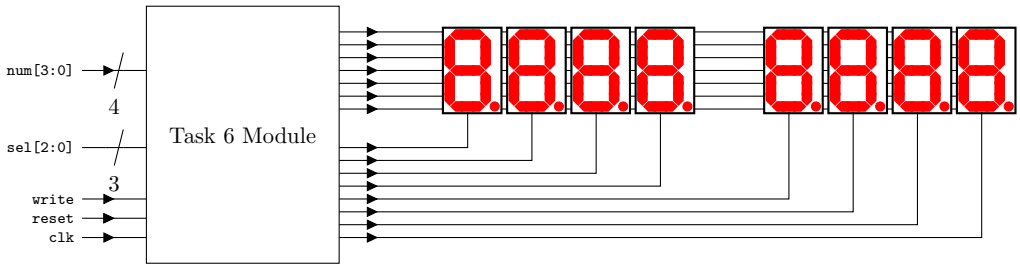

Fig. 6.1: Lab task modular circuit.

The user will first use the write operation (`write = 1`) to store the eight values corresponding to different seven-segment displays. The user stores the number to be displayed on the first seven-segment display by choosing `sel = 3'b000`, `num = desired number`, and turning the switch `write` ON. Similarly, the user stores the numbers to be displayed on other seven seven-segment displays by following the same method except choosing a different memory location with the help of `sel`.

The `reset` input will set all the saved values to zero.

After the write operation, we will read the stored numbers using `write = 0` signal. Fig. 6.3 and Fig. 6.2 show sample output for write and read operation of our module.

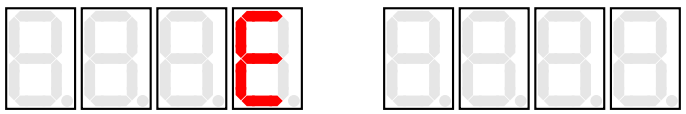

Fig. 6.2: Sample output for the Lab Task with `sel=3'b100`, `num=4'b1110` and `wr=1'b1`.

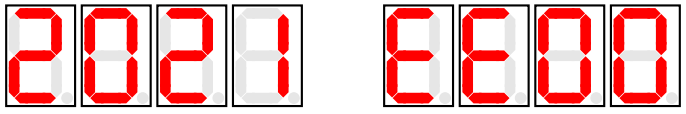

Fig. 6.3: Sample output for the Lab Task with `wr=1'b0`.

All the seven-segment displays share the same data wires. Therefore, at any point in time, we are limited to display only one number on a single display. However, if we enable different seven-segment displays fast

enough, it can create an illusion that all seven-segment displays are turned ON. Human eyes cannot perceive that a display is turning ON and OFF if this switching happens for more than 24 times a second. To be on the safe side, you can switch different displays ON and OFF at a rate of 100 times per second (100 Hz).

The lab requires that the circuit should have a time reference in it. The starter kit board provides an on board clock signal of 100 MHz. However, if you make a circuit that switches different seven-segment displays at a rate of 100 Mega times per second, the seven-segment display drivers may not be able to respond to such a high frequency change. Therefore, you will need to bring the clock down to approximately 100 Hz.

Note:
Use of latches are not allowed in the design of synchronous circuits for the lab. Synchronous circuits must be designed with flip-flops.

Hint:
To bring the clock down to 100 Hz from 100 MHz, you may be able to use a toggle flip-flop in a clever way. Look at the output of a toggle flip-flop carefully.

Deliverables

1. A report containing the following items:
   (a) The designed circuit for task module in the diagram above with hand-sketched design partition of the system. The design partition should include multiplexers, encoders, decoders and flip-flops, and should avoid gates as much as possible. The design should also show the labels of wires , along with the label for the width of the line for multi-bit wires.
   (b) You can use behavioral modeling **only** for the components you have studied in the class. For the rest, draw different K-maps used to generate those component circuits.
   (c) Synthesis maximum combinational delay: Read the synthesis report of your circuit and describe which path has the maximum combinational delay?
   (d) Implementation maximum combinational delay: Read the post implementation static timing report and identify the path with the maximum combinational delay? Is the path same as the one in the synthesis report?
   (e) Resource Utilization Report: Read the synthesis report and identify how many resources in the FPGA such as lookup tables (LUTs), input/output (IOs), etc., has been utilized.
2. Simulation of your designed module on QuestaSim. Show the simulation to the instructor.
3. FPGA Implementation: Synthesize the circuit for Nexys A7 FPGA available in the lab. Tie inputs to the switches and outputs to the seven-segment displays available on the board.

The collaboration between students is encouraged, but blind code sharing/copying is not allowed. If you are unable to explain anything in your code, it will be assumed you have copied it. So make sure you know every thing you have written in your code. I am least concerned about how you have learnt something as long as you have learnt it well.

RTL and Testbench Design for Sequential Circuits

Listing 1 and Listing 2 provide RTL and testbench codes for sequential circuits. For the simulation of sequential circuits, you are required to declare the `clk` signal as `reg` or `logic`. You generate the clock signal using the code given first initial block in Listing 2.

Acknowledgments

The manual has been written by Dr. Ubaid Ullah Fayyaz and Mr. Ali Imran.

```
module d_ff(output logic q_o,
input  logic clk_i, reset_i, enable_i, d_i
);

always_ff @ (posedge clk_i or posedge reset_i)
begin
if (reset_i)
q_o <= 1'b0;
else
begin
if (enable_i)
q_o <= d_i;
else
q_o <= q_o;
end
end
endmodule
```

Listing 1: RTL for D-Flip-Flop

```
module d_ff_tb();
logic q;
logic clk, reset, enable, d;

d_ff DUT (q,clk,reset,enable,d);

initial
begin
//initial value of clock
clk = 1'b0;
//generating clock signal
forever #5 clk = ~clk;
end

initial
begin
reset <= 0;
d     <= 1;
//change inputs at clock edges
@ (posedge clk);
reset <= 1;
@ (posedge clk);
reset <= 0;
d     <= 1;
@ (posedge clk);
d     <= 0;
// repeat(n) executes the given statement n number of times
repeat (3) @ (posedge clk);
$stop;
end
endmodule
```

Listing 2: Testbench for D-Flip-Flop