Digital Logic Laboratory

Report On Lab #2 – Pin Shift Register

Ana Maghradze

23rd October 2021

Contents

1 Description of The Assignment

For this assignment, we have to implement shift register. We should have 4-bit `data_in`, 1-bit `enter_in` as inputs and 4-bit `data_out` as output. User enters random numbers - indicates 4-bit number by `SWITCH` and then by pressing the `KEY[0]` enables to enter the number. If user enters such sequence of numbers that is the same as the last 3 digits of my redID (346), then first four `LED`s are ON.

2 Description of The Solution

I decided to have states and use FSM logic. As my red id ends with 346, I implemented the following:

At first, we are in 0th state until we enter 3. if we enter 3, we go to the 1st state and if we continue to enter 3s, we should stay in the 1st state, in case of any other input, we go back to 0st state. Then, if we have already entered 3 and now enter 4, we go to the 2nd state. Here, if we enter again 4, we go to 0th state or if we enter 3, we go to the 1st state (as we have 3 in first state). Finally, If we are in the 2nd state and enter 6, we go to the 3rd state where LEDs are on. Then we return to the initial state.

I have the code for this logic in a separate `shift_register_checker` module (as it was required) and instance of this module it in the `de0_nano_soc_baseline`.

3 ModelSim Simulation

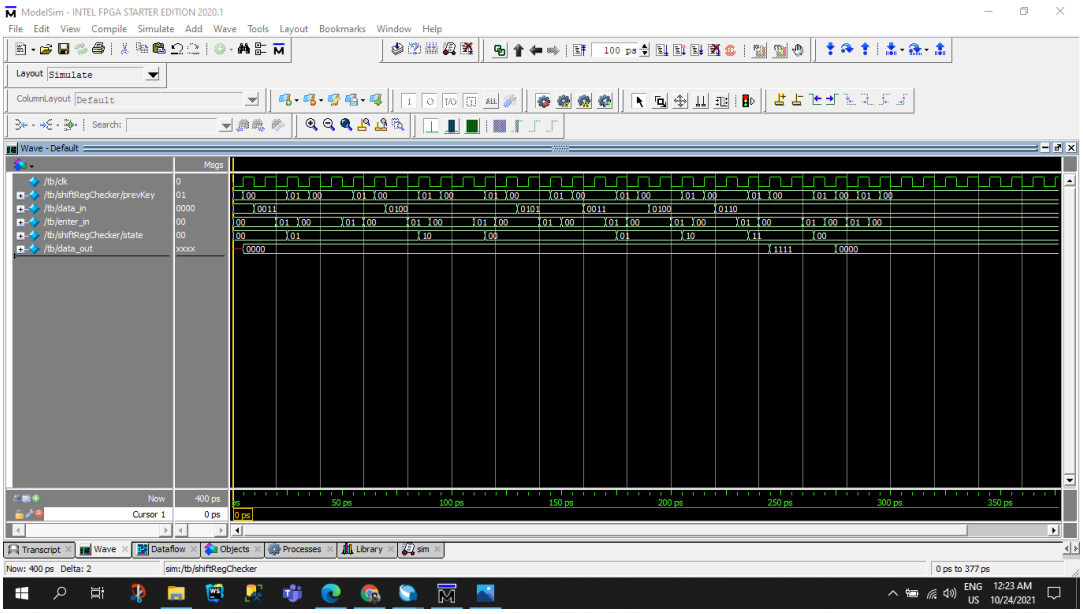

Figure 1: Simulation Result

4 Appendix

```
 module shift_register_checker(
  input clk,
  input [3:0] data_in,
  input [1:0] enter_in,
  output reg [3:0] data_out
);

// redID: 346
reg [1:0] state = 0;
reg [1:0] prevKey = 1;

always @(posedge clk)
begin
  if(enter_in[0] == 1 && prevKey[0] == 0)
  begin
    case(state)
      0: state <= data_in == 3 ? 1 : 0;
      1:  state <= data_in == 4 ? 2 : data_in == 3 ? 1 : 0;
      2: state <= data_in == 6 ? 3 : data_in == 3 ? 1 : 0;
      3:  state <= 0;
    endcase
  end

  data_out[3:0] <= state == 3 ? 4'b1111 : 0;

  prevKey[0] <= enter_in[0];
end

endmodule
```

```
module de0_nano_soc_baseline(
  input CLOCK_50,
  input [3:0] SW,
  input [1:0] KEY,
  output [7:0] LED
);

shift_register_checker shiftRegCheck(CLOCK_50, SW[3:0], KEY[1:0], LED[3:0]);

endmodule
```

```
module tb;
reg clk = 0;
reg [3:0] data_in = 0;
reg [1:0] enter_in = 0;
wire [3:0] data_out;

shift_register_checker shiftRegChecker(clk, data_in, enter_in, data_out);

always #5 clk <= ~clk;

initial
begin
  // 3 _ _
```

```
  #10 data_in = 3;
  #10 enter_in = 1;
  #10 enter_in = 0;

  // 3 _ _
  #10 data_in = 3;
  #10 enter_in = 1;
  #10 enter_in = 0;

  // 3 4 _
  #10 data_in = 4;
  #10 enter_in = 1;
  #10 enter_in = 0;

  //  3 4 _
  #10 data_in = 4;
  #10 enter_in = 1;
  #10 enter_in = 0;

  //  3 _ _
  #10 data_in = 5;
  #10 enter_in = 1;
  #10 enter_in = 0;

  // 3 _ _
  #10 data_in = 3;
  #10 enter_in = 1;
  #10 enter_in = 0;

  //  3 4 _
  #10 data_in = 4;
  #10 enter_in = 1;
  #10 enter_in = 0;

  // 3 4 6
  #10 data_in = 6;
  #10 enter_in = 1;
  #10 enter_in = 0;

  // 3 4 6
  #10 data_in = 6;
  #10 enter_in = 1;
  #10 enter_in = 0;

  #10 enter_in = 1;
  #10 enter_in = 0;

end

endmodule
```