

Design of Digital Circuits

Lab 4 Supplement: Finite-State Machines

Prof. Onur Mutlu

ETH Zurich

Spring 2020

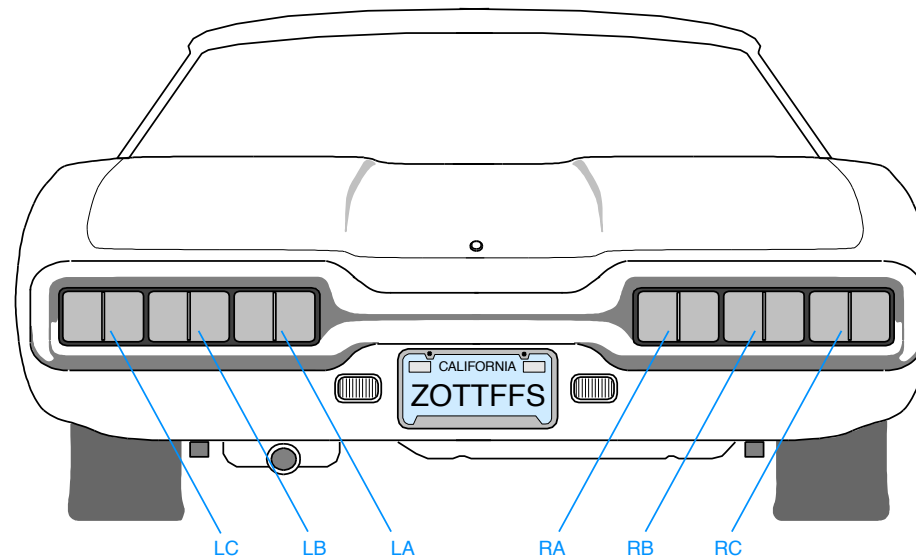
20 March 2020

What Will We Learn?

- In Lab 4, you will **implement a finite-state machine using Verilog.**
- Design and implement a simple circuit that emulates the blinking lights of a Ford Thunderbird.
- Understand how the **clock signal is derived in the FPGA board.**
- **Write an FSM** that implements the Ford Thunderbird blinking sequence.

Tail Lights of a 1965 Ford Thunderbird

- In this lab, you will design a finite-state machine to control the tail lights of a 1965 Ford Thunderbird.

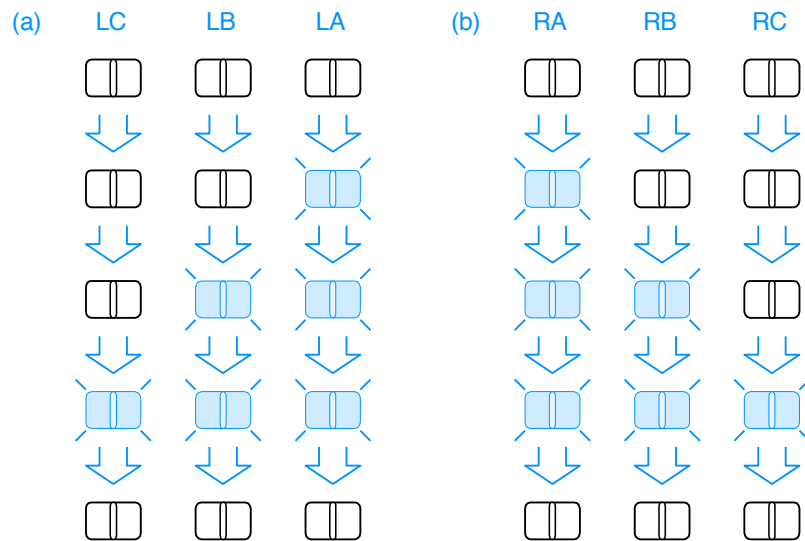


Copyright © 2000 by Prentice Hall, Inc.
Digital Design Principles and Practices, 3/e

Tail Lights of a 1965 Ford Thunderbird

- There are three lights on each side that operate in sequence to indicate the direction of a turn.

Copyright © 2000 by Prentice Hall, Inc.
Digital Design Principles and Practices, 3/e



Copyright from ClassicLEDs.com



Part 1: FSM Design

- An FSM must do three things:
 - **Next State Logic:** Determine the next state from **the present state and the inputs.**
 - **Output Logic:** **Determine the output signals** based on the present state and input signals.
 - **State Register:** keeps track of the present state; must be **updated at every clock cycle.**

The manual contains the details of this FSM specifications.

Part 1: FSM Design

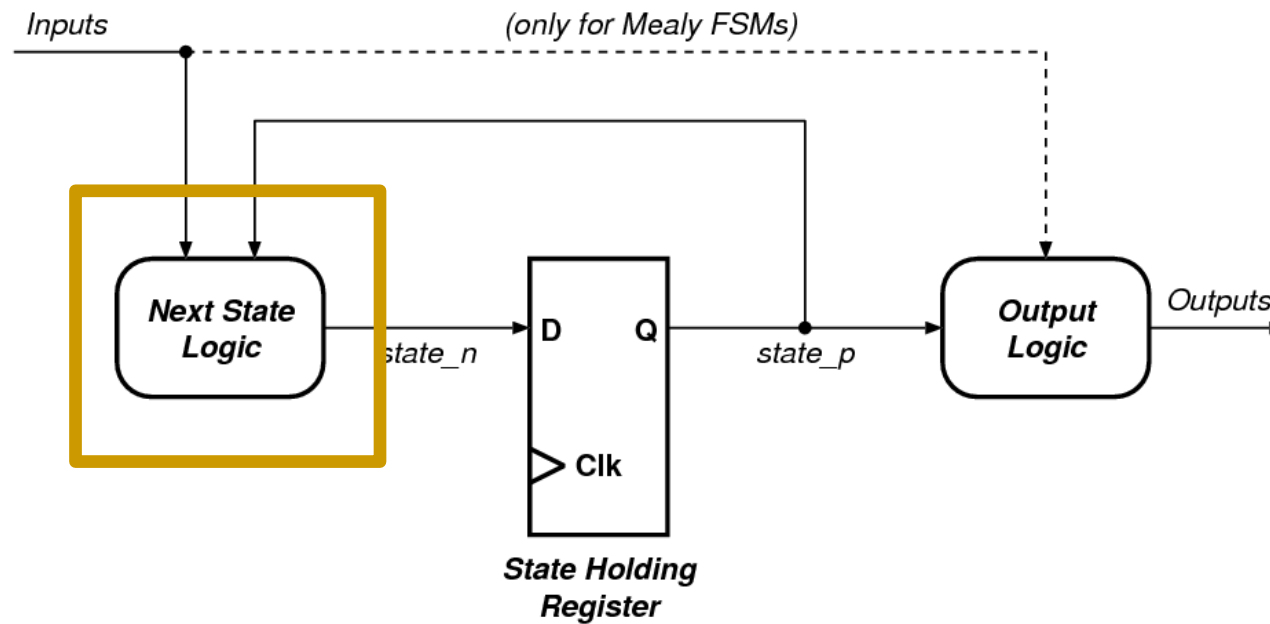
- An FSM must do three things:
 - **Next State Logic:** Determine the next state from **the present state and the inputs.**
 - **Output Logic:** **Determine the output signals** based on the present state and input signals.
 - **State Register:** keeps track of the present state; must be **updated at every clock cycle.**

For more details please refer to [lecture 6](#):

- Slide 62: Finite-State Machines.
- Slide 127: Moore vs. Mealy FSMs.

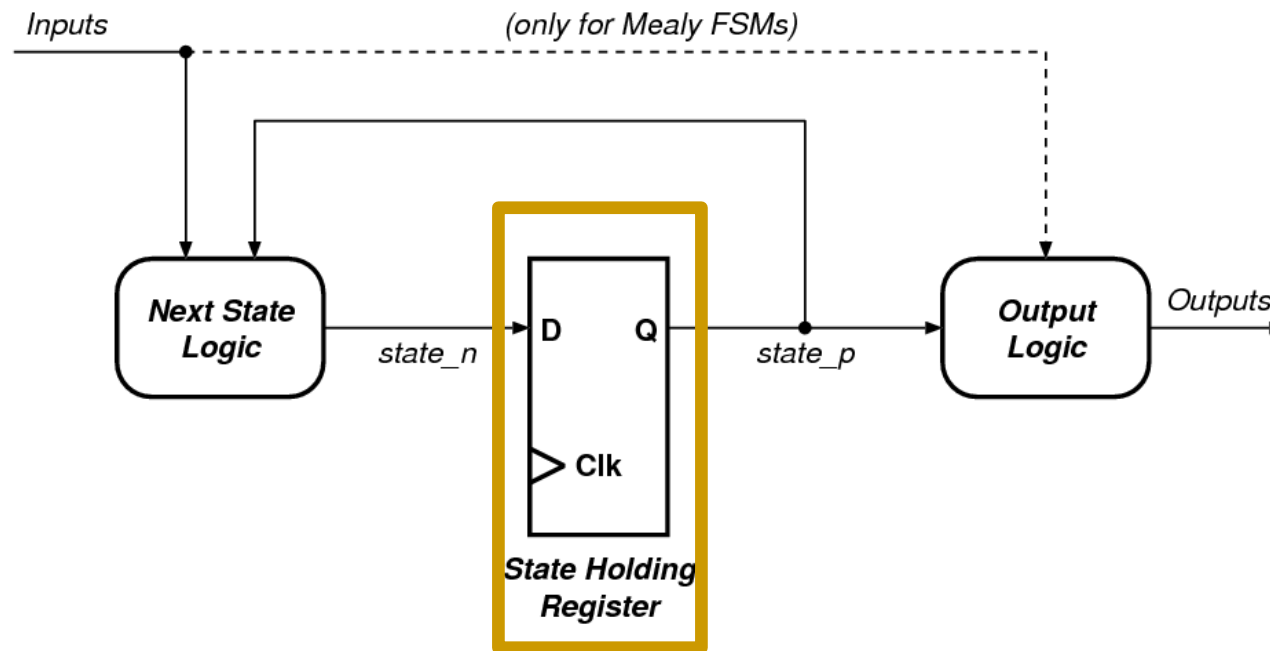
Part 2: Verilog Implementation

- Separate three parts of the code:



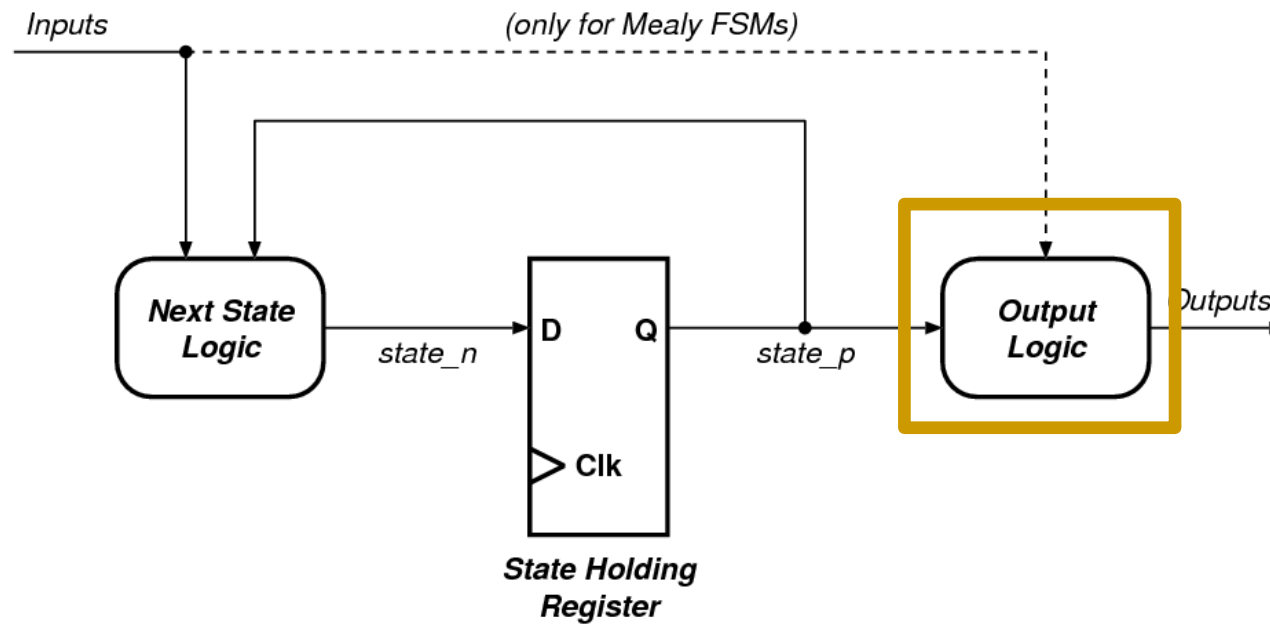
Part 2: Verilog Implementation

- Separate three parts of the code:



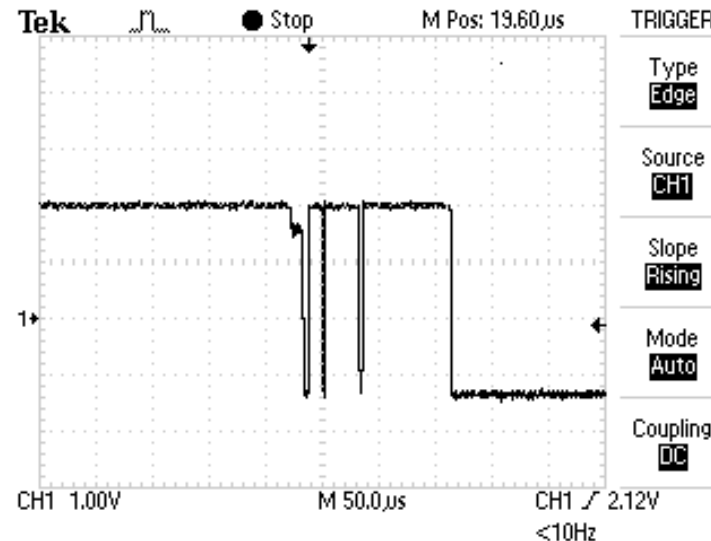
Part 2: Verilog Implementation

- Separate three parts of the code:



Part 3: Implementing the Clock (I)

- The problem of using push-buttons as clock:
 - Compared to the speed of the FPGA the change in a push button is *very* slow (~ 1 million times slower)
 - During the slow transition, the FPGA will see many fast occurring transitions and would interpret each of them as a clock edge. (**Bouncing**)



Part 3: Implementing the Clock (II)

- **CLK100Mhz (W5):** Your board contains a 100Mhz crystal oscillator circuit.
- **Problem:** The clock is too fast.
- **Solution:** A clock divider:

```
module clk_div(input clk, input rst, output clk_en);  
    reg [24:0] clk_count;  
    always @ (posedge clk)  
        //posedge defines a rising edge (transition from 0 to 1)  
        begin  
            if (rst)  
                clk_count <= 0;  
            else  
                clk_count <= clk_count + 1;  
            end  
            assign clk_en = &clk_count;  
        endmodule
```

Part 4: Defining the Constraints

- We must specify constraints for:
 - Buttons for control
 - LEDs for output lights
 - Connections for clock

The manual contains more information
about the constraints.

Last Words

- In Lab 4, you will **implement a finite-state machine using Verilog.**
- Design and implement a simple circuit that emulates the blinking lights of a Ford Thunderbird.
- Understand how the **clock signal is derived in the FPGA board.**
- **Write an FSM** that implements the Ford Thunderbird blinking sequence.
- In the report you will implement a **dimming function**, so that the lights are not only on and off, but can have intermediate levels

Report Deadline

23:59, 17 April 2020

Design of Digital Circuits

Lab 4 Supplement: Finite-State Machines

Prof. Onur Mutlu

ETH Zurich

Spring 2020

20 March 2020