

Pre Laboratory 2:

FINITE STATE MACHINES

OBJECTIVES

- Getting to know how to describe finite state machine (FSM) using variety styles of System Verilog code (logic expressions/ behavioral expressions/ shift registers).
- Design and implement digital circuits using FSM.
- Download the circuit into the FPGA chip and test its functionality.

PREPARATION FOR LAB 2

- Students have to simulate all the exercises in Pre Lab 2 at home. All results (codes, waveform, RTL viewer, ...) have to be captured and submitted to instructors prior to the lab session.
If not, students will not participate in the lab and be considered absent this session.

REFERENCE

1. Intel FPGA training



Pre Laboratory 2: FINITE STATE MACHINES

EXERCISE 1:

Objective: Create state diagram and build the circuit for a finite state machine.

Requirement: Consider following state diagram and one-hot state assignment

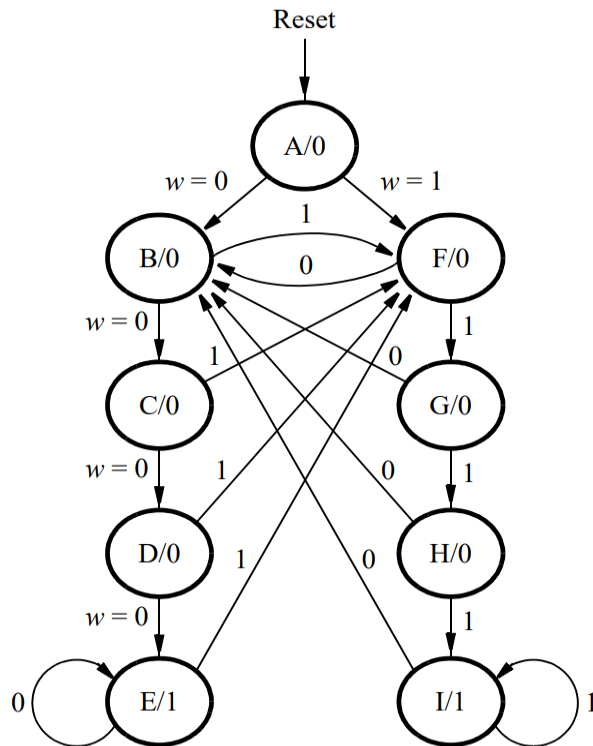


Figure 1: A state diagram for the FSM

Name	State Code
	$y_8y_7y_6y_5y_4y_3y_2y_1y_0$
A	000000001
B	000000010
C	000000100
D	000001000
E	000010000
F	000100000
G	001000000
H	010000000
I	100000000

Table 1: One-hot codes for the FSM

Instruction:

1. Write System Verilog assignment statements for all flip-flops of the FSM
2. Based on above statements, implement the circuit using flip-flops and logic gates.

Check: Your report has to show two results:

- Nine assignment statements for 9 flip-flops.
- The circuit diagram.



Pre Laboratory 2:

FINITE STATE MACHINES

EXERCISE 2:

Objective: Describe FSM using System Verilog behavioral expressions.

Requirement: The state table of a FSM is also described by using a System Verilog CASE statement in a ALWAYS block and use another ALWAYS block to instantiate the state flip-flops. The output z can be specified by using a third ALWAYS block or simple assignment statements.

Instruction:

Below is a suggested skeleton of the VHDL code. Write VHDL code which describe the FSM in exercise 1, which has state code shown in table 2. always_ff

Name	State Code <i>y₃y₂y₁y₀</i>
A	0000
B	0001
C	0010
D	0011
E	0100
F	0101
G	0110
H	0111
I	1000

Table 2: Binary codes for the FSM

```
module ...( ...
/* input and output declaration*/
);
//state declaration and assignment
typedef enum { A,B,C,D,E,F,G,H,I } statecode_;
// or enum bit [3:0] {A=4'b0000, B=4'b0001, ...} statecode;

statecode__ state , next ;
always_ff @ ( posedge clk , posedge rst ) /* FF description*/
if ( rst ) state <= A ;
```



Pre Laboratory 2:

FINITE STATE MACHINES

```
else state <= next ;
always_comb      /*next state transition*/
begin
next = state;    // default state : the same

case ( state )
A : if ( w ) next = F ;
    else next = B;
B :
...;
.
.
.
endcase
end
... assignments for output z and the LEDs
endmodule
```

Check: Your report has to show VHDL description for the FSM.



Pre Laboratory 2: FINITE STATE MACHINES

EXERCISE 3

Objective: Implement a shift register.

Requirement: Write a VHDL program which describes following shift register:

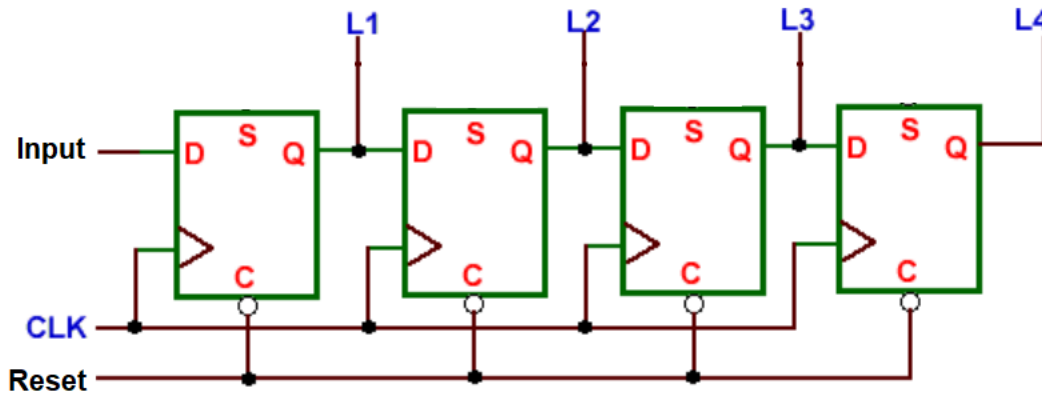


Figure 3: Shift register.

The output $L_4L_3L_2L_1$ equals “0000” when Reset = 0. Otherwise, input value will be shifted from L_1 to L_4 .

Instruction:

1. Create a new Quartus project for your circuit.
2. Write a System Verilog file that instantiates the four flip-flops in the circuit.
3. Compile the code. Use the Quartus RTL Viewer tool to examine the gate-level circuit produced from the code.
4. Simulate the behavior of your System Verilog code. Test the functionality of your design by inputting various data values and observing the generated outputs.

Check: Your report has to show two results:

- The waveform to prove the circuit works correctly.
- The result of RTL viewer.



Pre Laboratory 2:

FINITE STATE MACHINES

EXERCISE 4

Objective: Implement a periodic signal.

Requirement: Write a System Verilog program which create an enable signal that is asserted once every second.

Instruction:

1. Create a new Quartus project for your circuit.
2. Write a System Verilog file that create an enable signal that is asserted every second.
3. Compile and simulate the behavior of your. Test the functionality of your design.

Check: Your report has to show two results:

- The waveform to prove the circuit works correctly.



Pre Laboratory 2:

FINITE STATE MACHINES

EXERCISE 5

Objective: Know how to implement a digital circuit using an FSM.

Requirement: The Morse code uses patterns of short and long pulses to represent a message. Each letter is represented as a sequence of dots (a short pulse), and dashes (a long pulse). For example, the first eight letters of the alphabet have the following representation:

A • —
B — • • •
C — • — •
D — • •
E •
F • • — •
G — — •
H • • • •

Design and implement a Morse-code encoder circuit using an FSM.

Instruction:

1. Assign dot as '0' and dash as '1', we have:

Letter	SW ₂₋₀	Morse code	Morse Length
A	000	0010	010
B	001	0001	100
C	010	0101	100
D	011	0001	011
E	100	0000	001
F	101	0100	100
G	110	0011	011
H	111	0000	100

2. In order to construct the FSM, follow these notes:

- When KEY1 is pressed, the system begins to store the Morse code to be sent in a shift register (data = Morse_code), and its length in a counter (size = Morse_length). Otherwise, the system is idle. (notes: Morse_code and Morse_length depend on SW₂₋₀ value.)
- After loading data and size, the system starts to send LSB bit of data:



Pre Laboratory 2:

FINITE STATE MACHINES

If $\text{data}(0) = 0$, a LEDR will be on for half of a second. After a second, the data is right shifted and its size decreased by one.

If $\text{data}(0) = 1$, a LEDG will be on for half of a second. After a second, the data is right shifted and its size decreased by one.

- The process continues to display data on LED using new data and size value until data size equal 1.

Check: Your report has to show the FSM diagram.

