ENGG 125.03 Laboratory Activity 6: Finite State Machine (Simple Washing Machine)

Activity Sheet; Submitted: 17 May 2021

Mallari, Kevin Luis T.
BS Computer Engineering
Electronics, Computer, and Communications Engineering
Ateneo de Manila University
Philippines

***Abstract*—The laboratory activity is about simulating a simple washing machine using a finite state machine. In order to do this, a finite state machine was first created in quartus using the state machine wizard. From there, the Verilog HDL code was outputted. The test bench was created such that different outputs were outputted.**

Keywords - Finite State Machine, ModelSim, Quartus, Verilog

I. Computations and Solutions

The finite state machine is set to simulate a simple washing machine. Figure 1 shows the different functions of the virtual buttons and LEDs

Button	Function
S0	Coin
S1	Lid
S2	Double Wash

LEDs	Function
L0	Idle
L1	Soak
L2	Wash 1
L3	Rinse 2
L4	Wash 2
L5	Rinse 1
L6	Spin
L7	Stop

Fig. 1.Virtual Buttons and LEDs Functions

II. HDL Code

This section shows and explains the code made for the activity.

```
`timescale 1ns/1ns

module washing_machine (
    clock,reset,coin,lid,double_wash,timer,
    state[2:0]);

    input clock;
    input reset;
    input coin;
    input lid;
    input double_wash;
    input timer;
    tri0 reset;
    tri0 coin;
    tri0 lid;
    tri0 double_wash;
    tri0 timer;
    output [2:0] state;
    reg [2:0] state;
    reg [2:0] reg_state;
    reg [7:0] fstate;
    reg [7:0] reg_fstate;
    parameter
idle=0,soak=1,wash1=2,rinse2=3,wash2=4,rinse1=
5,spin=6,stop=7;

    initial
    begin
        reg_state <= 3'b000;
    end

    always @(posedge clock or posedge reset)
    begin
        if (reset) begin
            fstate <= idle;
            state <= 3'b000;
        end
        else begin
            fstate <= reg_fstate;
            state <= reg_state;
        end
    end
```

```
    always @(fstate or coin or lid or
double_wash or timer)
    begin
        reg_state <= 3'b000;
        case (fstate)
            idle: begin
                if (coin)
                    reg_fstate <= soak;
                else
                    reg_fstate <= idle;

                reg_state <= 3'b000;
            end
            soak: begin
                if (timer)
                    reg_fstate <= wash1;
                else
                    reg_fstate <= soak;

                reg_state <= 3'b001;
            end
            wash1: begin
                if (timer)
                    reg_fstate <= rinse2;
                else
                    reg_fstate <= wash1;

                reg_state <= 3'b010;
            end
            rinse2: begin
                if ((timer & double_wash))
                    reg_fstate <= wash2;
                else if ((timer &
~(double_wash)))
                    reg_fstate <= spin;
                else
                    reg_fstate <= rinse2;

                reg_state <= 3'b011;
            end
            wash2: begin
                if (timer)
                    reg_fstate <= rinse1;
                else
                    reg_fstate <= wash2;

                reg_state <= 3'b100;
            end
            rinse1: begin
                if ((timer & double_wash))
                    reg_fstate <= spin;
                else
                    reg_fstate <= rinse1;

                reg_state <= 3'b101;
            end
            spin: begin
                if (lid)
                    reg_fstate <= stop;
                else if ((timer & ~(lid)))
                    reg_fstate <= idle;
                else
                    reg_fstate <= spin;

                reg_state <= 3'b110;
            end
            stop: begin
                if (~(lid))
                    reg_fstate <= spin;
                else
                    reg_fstate <= stop;

                reg_state <= 3'b111;
            end
            default: begin
                reg_state <= 3'bxxx;
                $display ("Reach undefined
state");
            end
        endcase
    end
endmodule // SM1

// -------------------------

module tb_washingmachine();

reg clock, reset, coin, lid, double_wash,
timer;
wire [2:0] state;

washing_machine dut(
    clock,reset,coin,lid,double_wash,timer,
    state[2:0]
);

always #10 clock = ~clock;

initial begin
    clock = 0; #0 reset = 1; coin = 0; lid =
0; double_wash = 0; timer = 1;
    #100 reset = 0;
```

```
    #100 coin = 1;
    #1000 lid = 1;
    #1000 double_wash = 1;
    #1010 lid = 0;
    #1100 coin = 0;

    #2000 $stop;
end

endmodule
```

Fig. 2. Code for the FSM

The code is based on the finite state machine made using the finite state machine wizard. The different inputs for the state machine were instantiated. These are the clock, reset, lid, double_wash, coin, and timer variables. The variable state, which is a 3 bit number, determines the state of the state machine. The initial value of state is 000, which is the idle state while the maximum value is 111, which is the stop state.

During the idle state (state = 000), the machine will only move on if the coin is inserted (coin = 1). When the machine is in the soak state (state = 001), it will only move on if the timer is done. The wash 1 (state = 010) state has the same behavior. The rinse 2 state (state = 011) will either go to wash 2 (state = 100) or spin (state = 110) depending on the value of double_wash. If double_wash is on, then it will wash and rinse for the second time. Otherwise, it will go to spin. Once at spin, after the timer is done and the lid is closed, it will go back to idle. If the lid is open, however, the state machine will go to the stop (state = 111) state.

For the testbench, different values of inputs were initialized so that the behavior of the state machine can be observed. The different inputs changed were the coin, lid, and double_wash.

III. Output Circuit

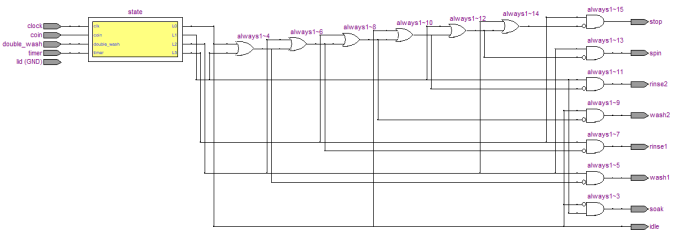

Fig. 3. Circuit for the T-Bird Pattern

The circuit in figure 3 is the output circuit of the entire code. Different gates were implemented into the circuit to produce the state machine.

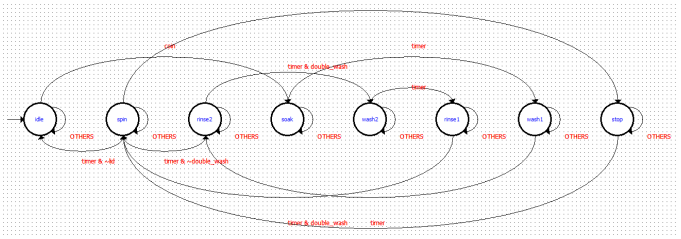

Fig. 4. Circuit for the T-Bird Pattern

Figure 4 shows the finite state machine diagram for the activity. It follows the needed transitions for the activity.

IV. Screenshot(s) of Simulations and VWF file

The screenshots of the simulations are presented below. The simulations of the different outputs produced by the system are shown. Multiple possible outputs are shown in all of the figures. It must be noted that the timer is expected to be set by the user, therefore, only 1 and 0 values were used in the testbench.

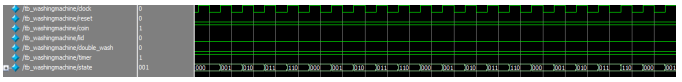

Fig. 4. Simulation for only coin is high.

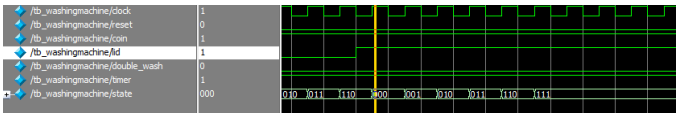

Fig. 5. Simulation for when the lid is open.

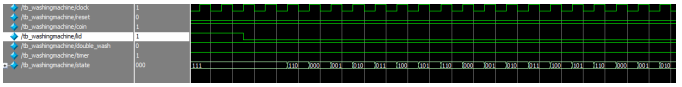

Fig. 6. Simulation for a when the lid is closed.

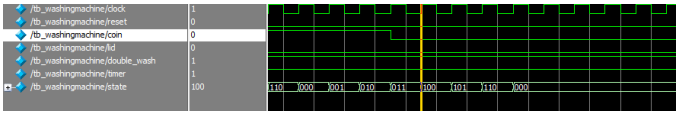

Fig. 7. Simulation for when the coin is not inserted..