ELEC402 Introduction to VLSI Systems

Electric Washer & Dryer 2 in 1 Machine

Assignment 1 Report

Ruyi Zhou

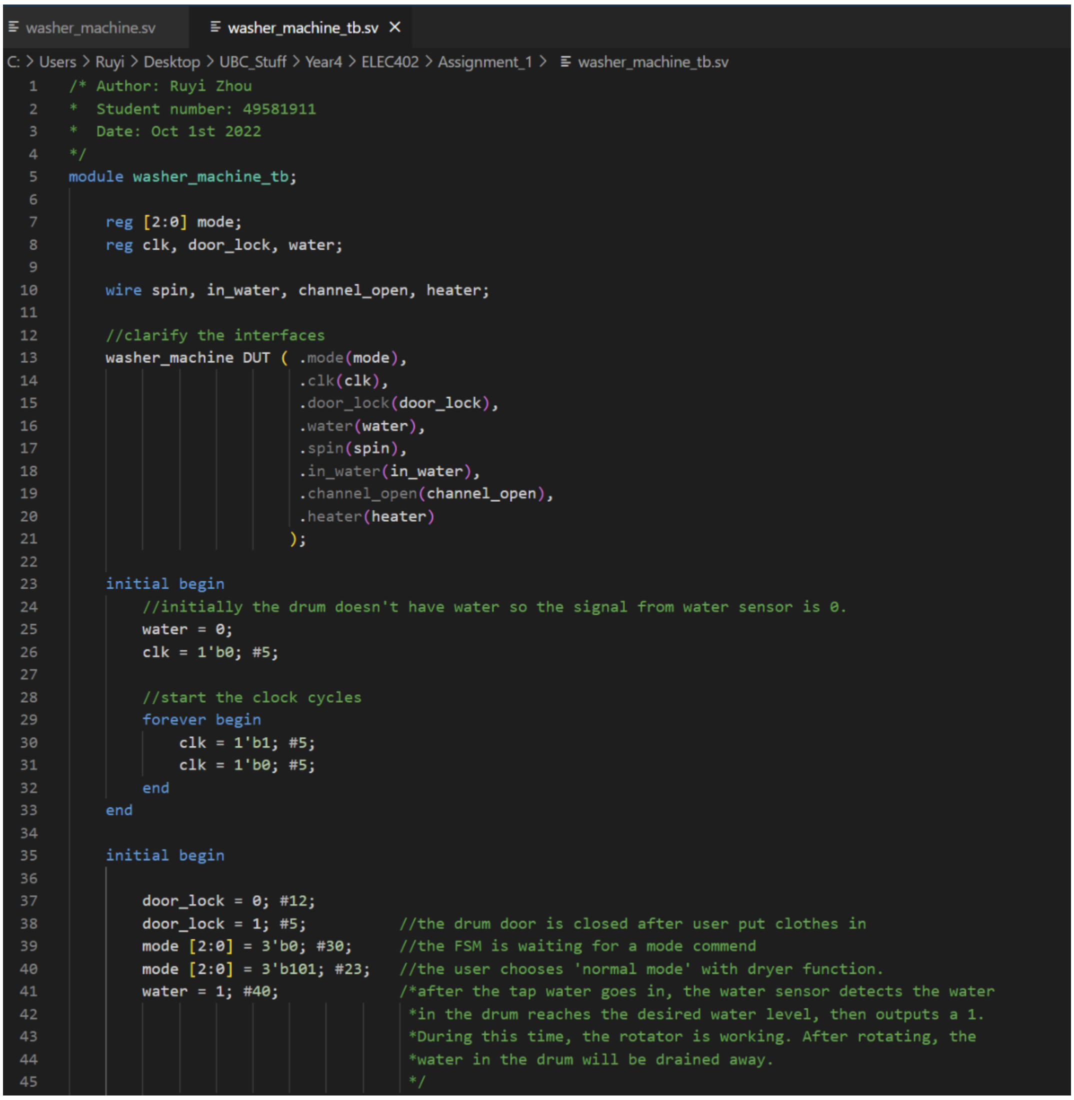
49581911

General Description

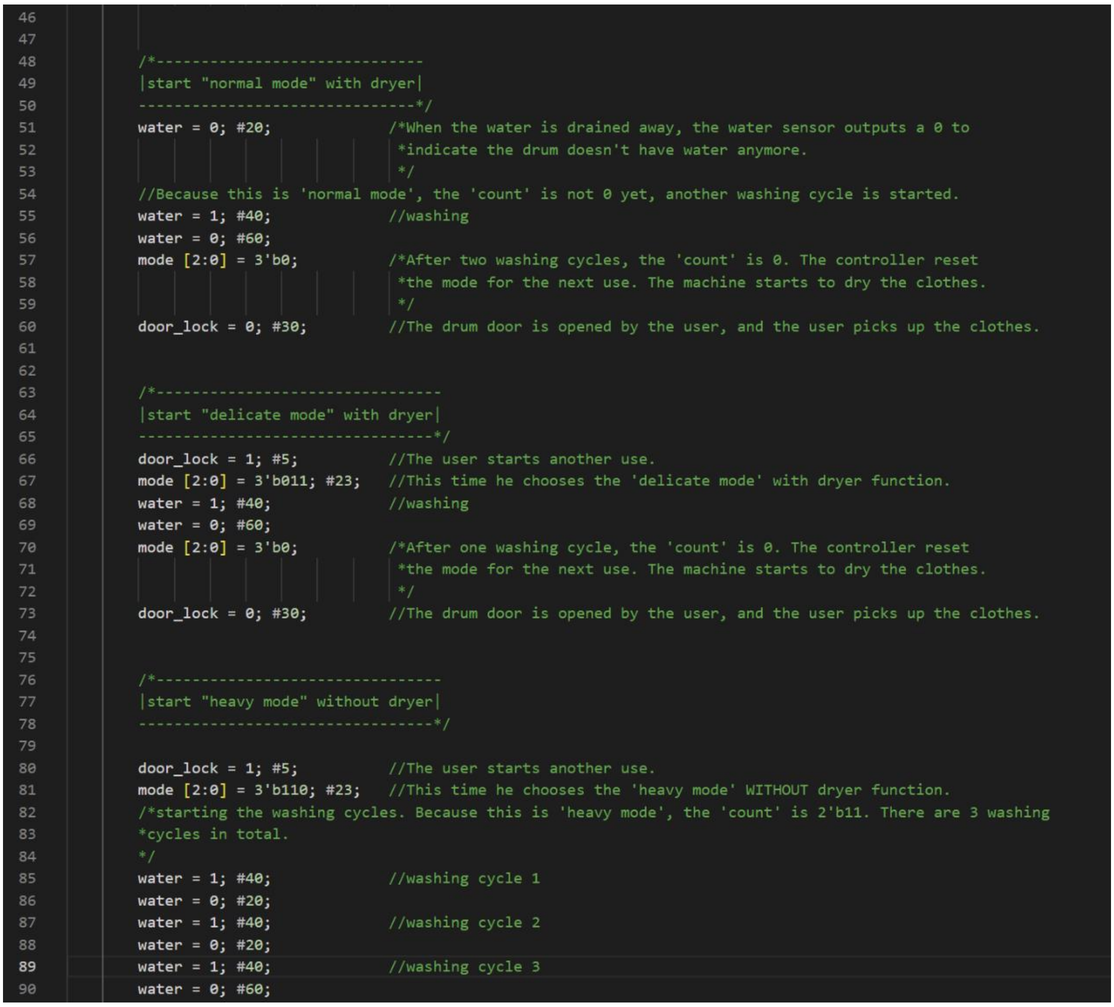
The finite state machine in this assignment is designed to be used in a laundry machine which has the functions of washing and drying. According to the state diagram (Figure 1) below, the machine is initially at the 'idle' state, and the door sensor starts to check whether the door of drum is closed. After the user puts the clothes in and closes the door, the FSM prompts the user to choose a mode. Then, the FSM outputs a '*in_water*' signal to the controller to add water into the drum. The FSM starts to check whether the water reaches the desired level. If the water is not enough, the FSM goes back to the 'Water_in' state and adds more water. When the water in the drum reaches the target water level, the water level sensor inputs a '1' to the FSM and the FSM goes to the 'Rotate' state to make the drum spin by outputting a '*spin*' signal to the controller. After rotating, the state goes to 'Water_Out'. At this state, the FSM outputs a '*channel_open*' signal to the controller to open the valve of the drum to drain away the water. Meanwhile, the water level sensor detects the water level in the drum. If it is empty, the state goes to 'Wait_for_dry'. This is one complete washing cycle. In this design, Normal mode has two washing cycles, Heavy mode has three washing cycles, and Delicate mode has one washing cycle. At 'Wait_for_dry' state, the FSM checks if the *count* number is 0. The *count* number depends on different modes (Normal is 2, heavy is 3, delicate is 1). When the *count* is not 0, the state goes back to 'Water_in' and starts a new washing cycle. If the *count* is 0, the FSM goes to 'Dryer' state to check whether the dryer will be used. If the user wants to dry (mode[0] = 1), the state goes to 'Heater_On' and the FSM outputs a *heat* signal to the controller to turn on the heater. After heating, the machine finished all the laundry cycles and goes to 'Finished' state. While at the 'dryer' state, if the user doesn't use the dryer function (mode[0] = 0), the states goes to 'Finished' state directly. At 'Finished' state, the machine is waiting for the user to take clothes. When the user open the drum door, the door sensor inputs a *!door_lock* signal to the FSM. Then the FSM goes to 'idle' state and is waiting for the next use.

Testbench

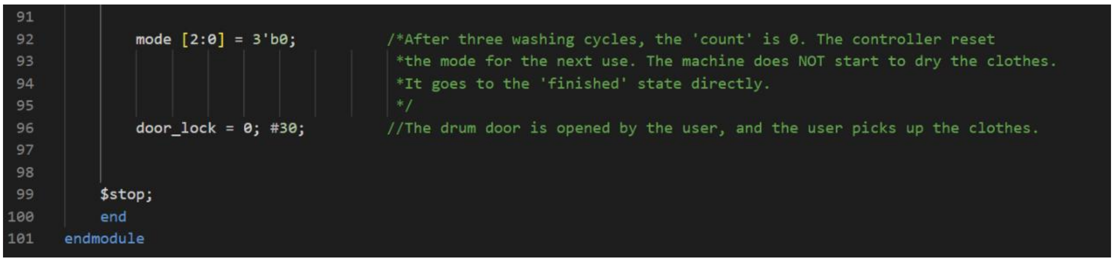
The comments are in the screenshot.


washer_machine.sv
washer_machine_tb.sv
C: > Users > Ruyi > Desktop > UBC_Stuff > Year4 > ELEC402 > Assignment_1 > washer_machine_tb.sv

```
/* Author: Ruyi Zhou
*  Student number: 49581911
*  Date: Oct 1st 2022
*/
module washer_machine_tb;

    reg [2:0] mode;
    reg clk, door_lock, water;

    wire spin, in_water, channel_open, heater;

    //clarify the interfaces
    washer_machine DUT ( .mode(mode),
                         .clk(clk),
                         .door_lock(door_lock),
                         .water(water),
                         .spin(spin),
                         .in_water(in_water),
                         .channel_open(channel_open),
                         .heater(heater)
                        );

    initial begin
        //initially the drum doesn't have water so the signal from water sensor is 0.
        water = 0;
        clk = 1'b0; #5;

        //start the clock cycles
        forever begin
            clk = 1'b1; #5;
            clk = 1'b0; #5;
        end
    end

    initial begin

        door_lock = 0; #12;
        door_lock = 1; #5;          //the drum door is closed after user put clothes in
        mode [2:0] = 3'b0; #30;     //the FSM is waiting for a mode commend
        mode [2:0] = 3'b101; #23;   //the user chooses 'normal mode' with dryer function.
        water = 1; #40;             /*after the tap water goes in, the water sensor detects the water
                                     *in the drum reaches the desired water level, then outputs a 1.
                                     *During this time, the rotator is working. After rotating, the
                                     *water in the drum will be drained away.
                                     */
```



```


        /*------------------------------
        |start "normal mode" with dryer|
        ------------------------------*/
        water = 0; #20;             /*When the water is drained away, the water sensor outputs a 0 to
                                     *indicate the drum doesn't have water anymore.
                                     */
        //Because this is 'normal mode', the 'count' is not 0 yet, another washing cycle is started.
        water = 1; #40;             //washing
        water = 0; #60;
        mode [2:0] = 3'b0;          /*After two washing cycles, the 'count' is 0. The controller reset
                                     *the mode for the next use. The machine starts to dry the clothes.
                                     */
        door_lock = 0; #30;         //The drum door is opened by the user, and the user picks up the clothes.


        /*--------------------------------
        |start "delicate mode" with dryer|
        --------------------------------*/
        door_lock = 1; #5;          //The user starts another use.
        mode [2:0] = 3'b011; #23;   //This time he chooses the 'delicate mode' with dryer function.
        water = 1; #40;             //washing
        water = 0; #60;
        mode [2:0] = 3'b0;          /*After one washing cycle, the 'count' is 0. The controller reset
                                     *the mode for the next use. The machine starts to dry the clothes.
                                     */
        door_lock = 0; #30;         //The drum door is opened by the user, and the user picks up the clothes.


        /*--------------------------------
        |start "heavy mode" without dryer|
        --------------------------------*/

        door_lock = 1; #5;          //The user starts another use.
        mode [2:0] = 3'b110; #23;   //This time he chooses the 'heavy mode' WITHOUT dryer function.
        /*starting the washing cycles. Because this is 'heavy mode', the 'count' is 2'b11. There are 3 washing
        *cycles in total.
        */
        water = 1; #40;             //washing cycle 1
        water = 0; #20;
        water = 1; #40;             //washing cycle 2
        water = 0; #20;
        water = 1; #40;             //washing cycle 3
        water = 0; #60;
```

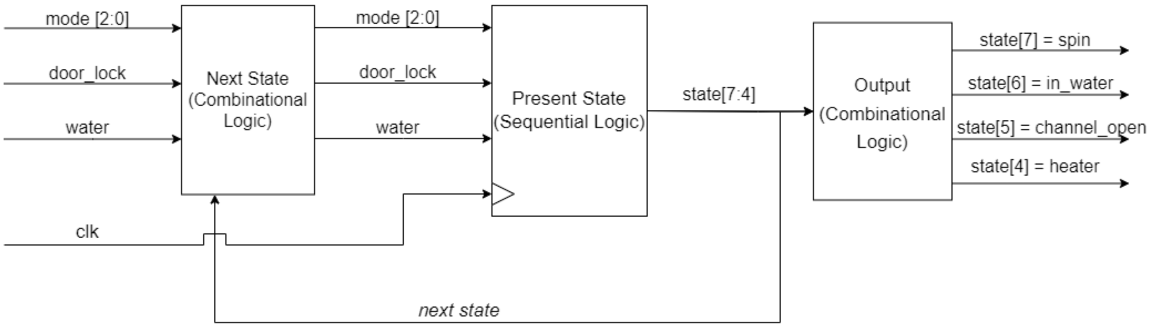


```

        mode [2:0] = 3'b0;          /*After three washing cycles, the 'count' is 0. The controller reset
                                     *the mode for the next use. The machine does NOT start to dry the clothes.
                                     *It goes to the 'finished' state directly.
                                     */
        door_lock = 0; #30;         //The drum door is opened by the user, and the user picks up the clothes.


    $stop;
    end
endmodule
```

Block Diagram of FSM

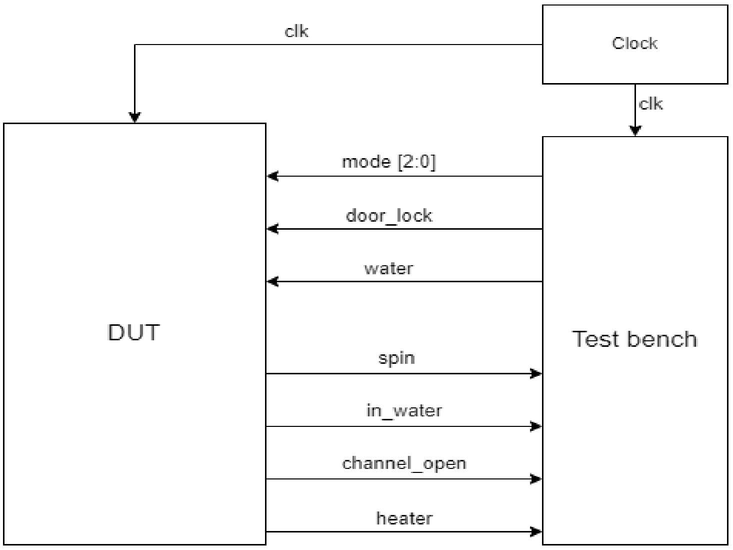
Diagram 1. the block diagram of the FSM module

Inputs: mode[2:0], door_lock, water, clk
Outputs: spin, in_water, channel_open, heater



Block Diagram of How FSM and Testbench connected

Diagram 2. the connection between DUT and test bench



A State Diagram with data flow

Figure 1. State Diagram of the FSM in the Washer & Dryer 2 in 1 Machine

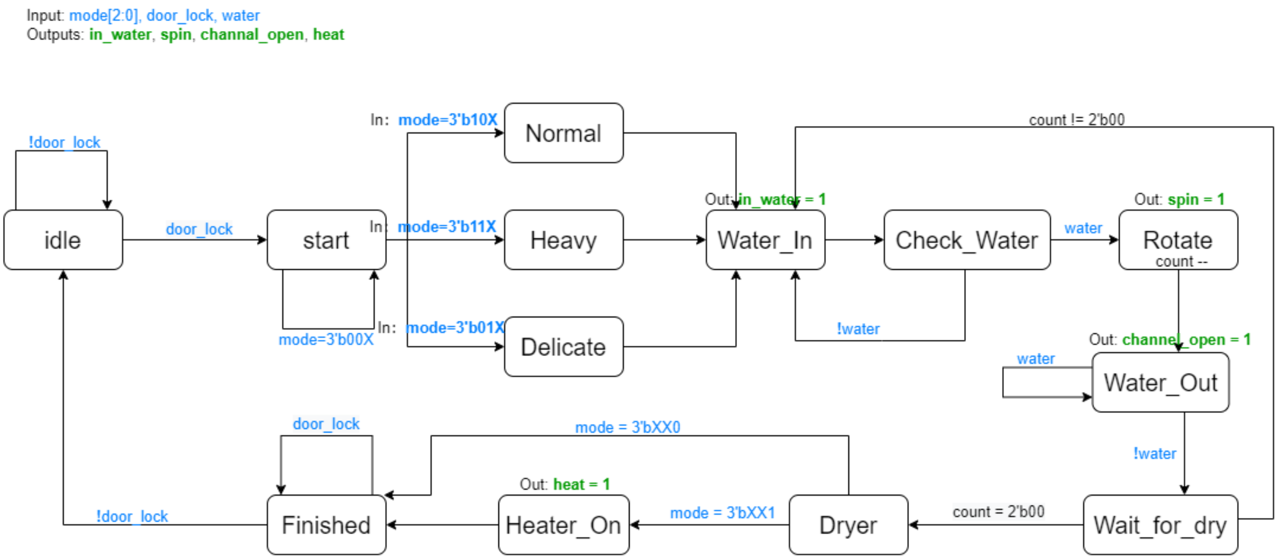


Figure 1 shows the state diagram of the FSM with data flow. The inputs and outputs signals are shown in blue and green color respectively. The inputs signals *door_lock* and *water* are from the sensor to detect if the door is closed and if the water in the drum reaches the desired water level. The *mode* signal is an input from users. The last bit of '*mode*' represents the use of dryer (mode[0] = 0 means the dryer will not be used). The Normal mode sets the *count* signal to 2'b10; Heavy mode sets the *count* signal to 2'b11; Delicate mode sets the *count* signal to 2'b01. The green output signals only indicate the output value at the current state, otherwise is 0. The black signals represent the wires.

Copy of code (separate file, size font 8)

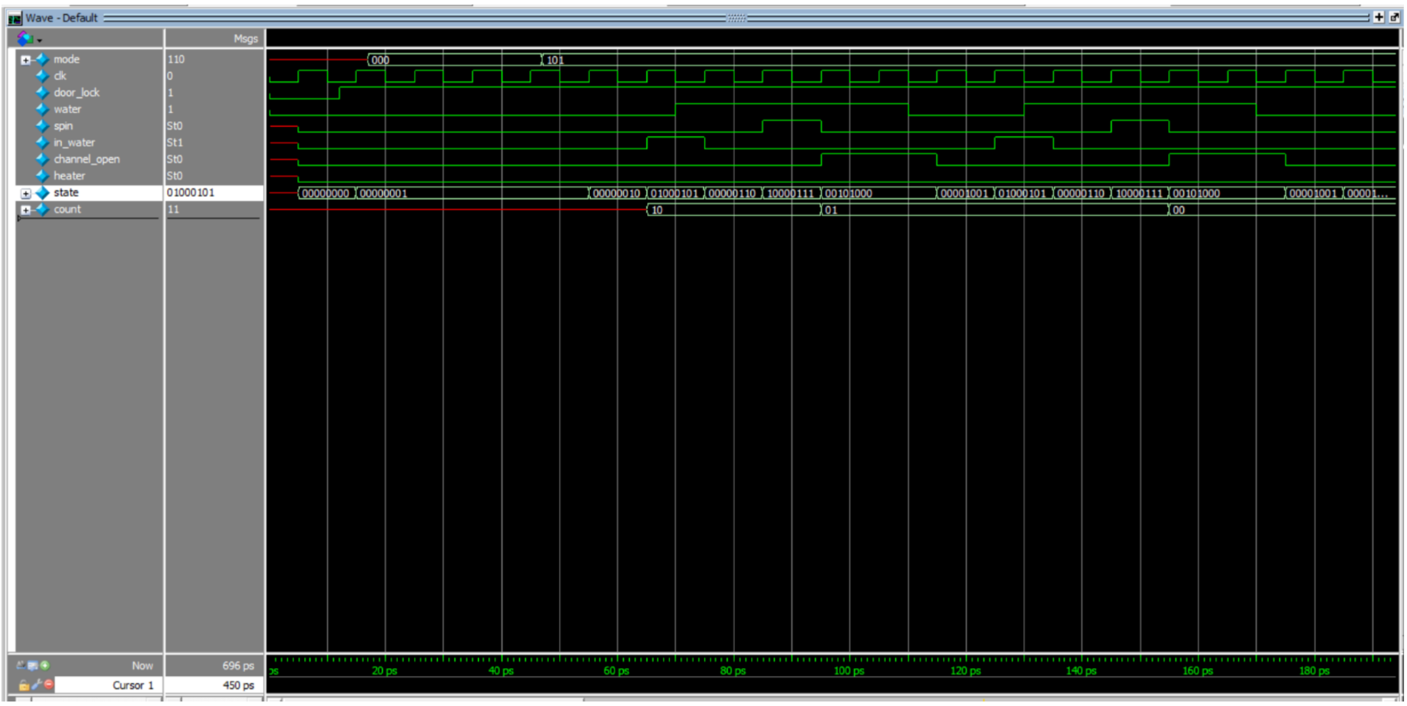
The code is included in the **Appendix**.

Simulation waveform result

The waveforms below are continuous in time. The state parameters are clarified below:

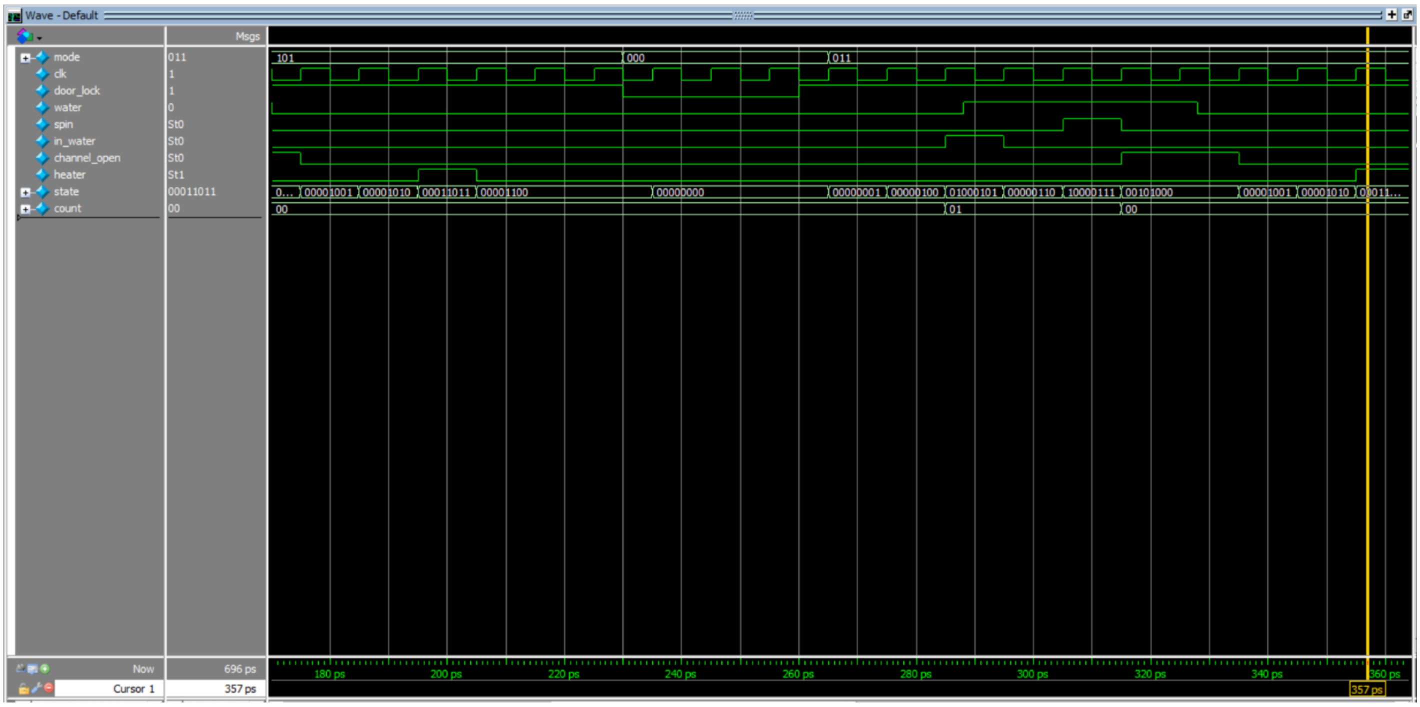
Inputs are in blue; Outputs are in green; Wires are in black.

0000_0000	idle
0000_0001	start
0000_0010	normal
0000_0011	heavy
0000_0100	delicate
0100_0101	water_in
0000_0110	check_water
1000_0111	clean_spin
0010_1000	water_out
0000_1001	wait_for_dry
0000_1010	dryer
0001_1011	heater_on
0000_1100	finish



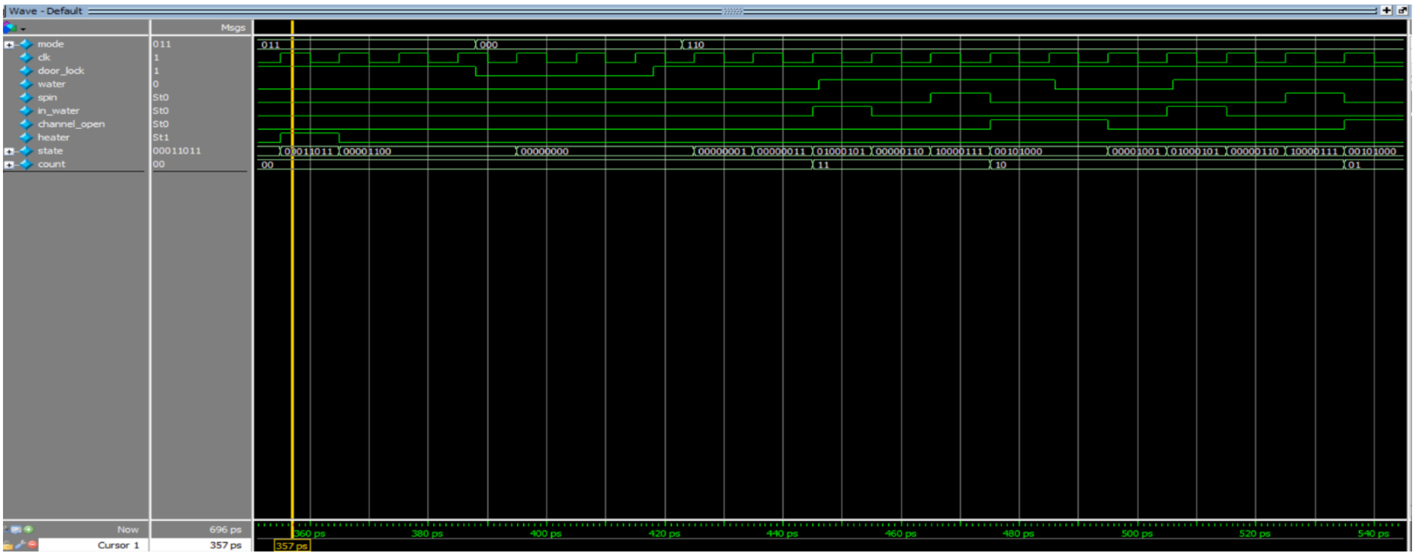
Initially, the state is 'idle (0000_0000)' to wait for a user. The user puts the clothes in and closes the door so the *door_lock* signal is 1. Then the state goes to 'start (0000_0001)'. The user chooses 'normal mode (101)' with dryer so the state goes to 'normal (0000_001)'. Then the state goes to 'water_in (0100_0101)' to let tap water goes into the drum by outputting a *in_water* signal. Then the water level sensor detects the water in the drum (at check_water 0000_0110) reaches the desired level, thus the sensor inputs a *water* signal to the FSM. The state goes to 'clean_spin (1000_0111)' which outputs a *spin* signal to start rotation. Meanwhile, the *count* is decreased by 1.

Then the state goes to ‘water_out (0010_1000)’ and outputs a *channel_open* signal to drain the water away. At this time, the *water* signal is low, indicating the drum does not have any water. The state then goes to ‘wait_for_dry (0000_1001)’. However, the *count* is not 0, so the state goes back to ‘water_in (0100_0101)’ to start a new washing cycle.

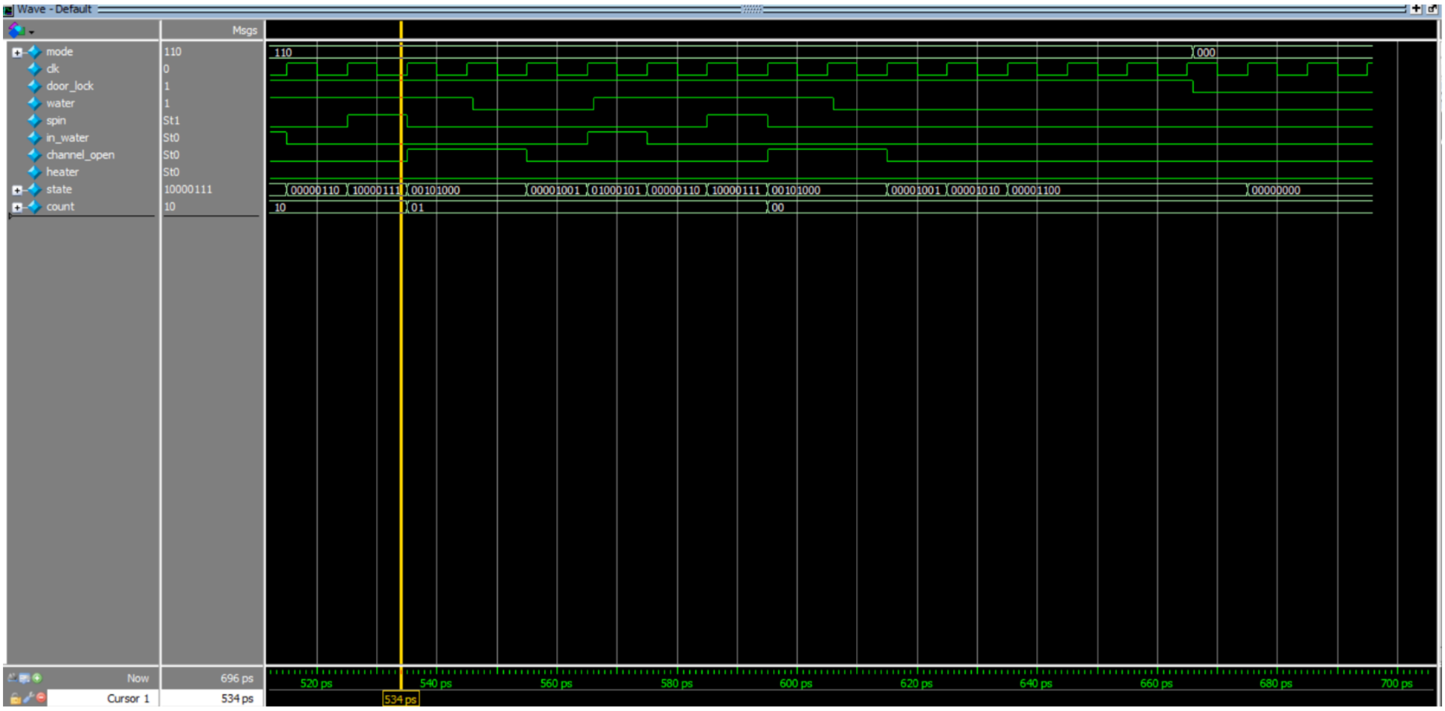


After this washing cycle (state is at ‘wait_for_dry(0000_1001)’), the *count* is 0, and the state goes to ‘dryer (0000_1010)’. Then the state changes to ‘heater_on (0001_1011)’ and outputs a *heater* signal to enable the heater. After heating, the state goes to ‘finish (0000_1100) and waiting for the user to pick up the clothes. At the moment of the *door_lock* signal is 0, the user opens the drum door and pick up the clothes. The state goes to ‘idle (0000_0000)’ and waiting for the next use.

The time after **260ps** is the simulation of ‘delicate mode’. The mechanism is the same as ‘normal mode’.



The time after **420ps** is the simulation of ‘heavy mode’. The mechanism is the same as ‘normal mode’ as well.



This simulation waveforms contains all three modes of this machine. The simulation results follow the expectation which means this design is good to work.

Appendix

```
/* Author: Ruyi Zhou
*   Student number: 49581911
*   Date: Oct 1st 2022
*/
module washer_machine (mode, clk, door_lock, water, spin, in_water, channel_open, heater);

    input logic [2:0] mode;
    input logic clk, door_lock, water;


    output logic spin, in_water, channel_open, heater;

    parameter [7:0] idle          = 8'b0000_0000,
                    start         = 8'b0000_0001,
                    normal        = 8'b0000_0010,
                    heavy         = 8'b0000_0011,
                    delicate      = 8'b0000_0100,
                    water_in      = 8'b0100_0101,
                    check_water   = 8'b0000_0110,
                    clean_spin    = 8'b1000_0111,
                    water_out     = 8'b0010_1000,
                    wait_for_dry  = 8'b0000_1001,
                    dryer         = 8'b0000_1010,
                    heater_on     = 8'b0001_1011,
                    finish        = 8'b0000_1100;


    logic [7:0] state;
    logic [1:0] count;


    always_ff@(posedge clk) begin

        case(state)

            idle:begin
                if(door_lock) state <= start;
                else state <= idle;
            end

            start:begin
                if(mode[2:1] == 2'b10) state <= normal;
                else if (mode[2:1] == 2'b11) state <= heavy;
                else if (mode[2:1] == 2'b01) state <= delicate;
                else state <= start;
            end

            normal:begin
                count <= 2'b10;
                state <= water_in;
            end

            heavy:begin
                count <= 2'b11;
                state <= water_in;
            end

            delicate:begin
                count <= 2'b01;
                state <= water_in;
            end

            water_in:begin
                state <= check_water;
            end

            check_water:begin
                if(water) state <= clean_spin;
                else state <= water_in;
            end

            clean_spin:begin
                count--;
                state <= water_out;
            end

            water_out:begin

                if(!water) begin
                    state <= wait_for_dry;
                end
                else begin state <= water_out; end
            end

            wait_for_dry:begin
                if(count == 0) state <= dryer;
```

```verilog
                else state <= water_in;
            end

            dryer:begin
                if(mode[0]) state <= heater_on;
                else state <= finish;
            end

            heater_on:begin
                state <= finish;
            end

            finish:begin
                if(!door_lock) state <= idle;
                else state <= finish;
            end
            default:begin
                state <= idle;
            end

        endcase

    end


    assign {spin, in_water, channel_open, heater} = state[7:4];


endmodule
```