Ain shams University

Engineering College

Computer Engineering and Software Systems

ELECTRONIC DESIGN AUTOMATION PROJECT

PHASE 1

WASHING MACHINE CONTROLLER

Team 14

Name:	**ID:**
Martin Magued	22P0193
Mohamed Bashir	22P0223
Nouran Mokhtar	22P0254
Roaa Sherif Refaat	22P0188
Roger Sherif Selim	22P0112
Seif Aly Fahmy	22P0182

Presented to:

Dr. Eman Mohamed El Mandouh Hussein

TA. Abdelrahman Sherif Fayez

Table of contents:

Introduction:

This project focuses on designing and implementing a controller for a washing machine using a finite state machine (FSM) approach. The idea is to simulate how a real washing machine operates by managing different states, such as Idle, selecting options, configuring settings, and going through the washing cycle stages like Wash, Rinse, Drain, and Dry. The controller is built in Verilog, a popular hardware description language, and handles user inputs, settings validation, and even errors, like when the door is left open or there's not enough water.

To make sure the design works as expected, we used Property Specification Language (PSL) assertions to check the system's behavior under different conditions. These assertions help verify that the controller transitions correctly between states and handles errors reliably, making the system robust and efficient.

This project combines digital design concepts with real-world problem-solving, demonstrating how FSMs can control complex systems. In this report, we'll walk through the design, how it works, and the steps we took to test and verify it. The result is a functional washing machine controller that's both practical and fault tolerant.

Design Phase: -

Design Specifications: -

Purpose:

The purpose of this module is to implement a state machine for a washing machine controller, allowing it to handle various modes, errors, and configurations. It simulates the behavior of a real-world washing machine by transitioning through states like washing, rinsing, draining, and drying while addressing power control, user input, and error handling.

Functional Description:

The washing machine controller is a finite state machine (FSM) designed to:

- Power on/off the washing machine.
- Configure operation modes and manual timer settings.
- Transition through washing states: Wash, Rinse, Drain, Dry, and Completion.
- Handle errors such as door not being closed or insufficient water.
- Support a "Pause" feature that maintains the state when interrupted.
- Output the current state (cs) and indicate cycle completion (cycleComplete).

Inputs and outputs:

Inputs:

- clk: Clock signal to synchronize state transitions.
- powerButton: Signal to turn the machine on/off.
- configu: Signal to enter the configuration mode.
- run: Signal to start the washing process.
- mode: 3-bit input specifying the wash mode:
- 000: QuickWash
- 001: Sports
- 010: GentleCare
- 011: Denim
- 100: Wool
- 101: Synthetics
- manualTimer: 32-bit input for custom time settings (minimum: 120 seconds).
- door_error: Signal indicating if the machine door is open.
- water_error: Signal indicating insufficient water level.

Outputs:

- cs: 4-bit output representing the current state.
- cycleComplete: Indicates whether the washing cycle is complete.

State Machine Description

The FSM consists of the following states:

- **Idle** (0000): Default state when the machine is off or awaiting input.
- **Options** (0001): Select washing mode or enter configuration.
- **Configurations** (0010): Set manual timer (custom duration).
- **Ready** (0011): Prepares for the washing process.
- **Wash** (0100): Washing state; timer decrements until rinse threshold.
- **Rinse** (0101): Rinse state; timer decrements until drain threshold.
- **Drain** (0110): Drains water, handles intermittent transitions.
- **Dry** (0111): Drying state; final state before completion.
- **Completion** (1000): Indicates successful completion of the cycle.
- **Pause** (1010): Pauses the machine due to user interruption or error.
- **CheckForError** (1111): Validates door and water conditions before proceeding.
- **Bamla_Mayya** (1001): An internal state for filling water inside the machine.

Modes:

- The module supports six operational modes:
- QuickWash (120-time units): For lightly soiled clothes.
- Sports (190-time units): For sportswear.
- GentleCare (140-time units): For delicate fabrics.
- Denim (160-time units): For denim clothing.
- Wool (150-time units): For wool garments.
- Synthetics (230-time units): For synthetic fabrics.

Error handling:

- Door Error: If the door is not closed, the machine transitions to the Pause state.
- Water Error: If water level is insufficient, the machine pauses and retries checks until the error resolves.
- Configuration Timer Expiry: If no manual timer is set within a defined limit (20 seconds), the machine powers off.

Timer and Manuel settings:

- Default Timer: Based on the selected mode.
- Manual Timer: Must be greater than or equal to 120 seconds; otherwise, an error is displayed, and the machine remains in the Configurations state.

Power Management:

- **Internal Power State**: Tracks whether the machine is powered on or off.
- **Power Button**: Toggles the machine on/off. If the machine is running and power is interrupted, it transitions to Pause.

Simulation outputs:

During simulation, the FSM displays:

- Mode selection confirmations.
- Error notifications (e.g., "Door not closed!" or "Water level low!").
- State transitions.
- Cycle completion message.

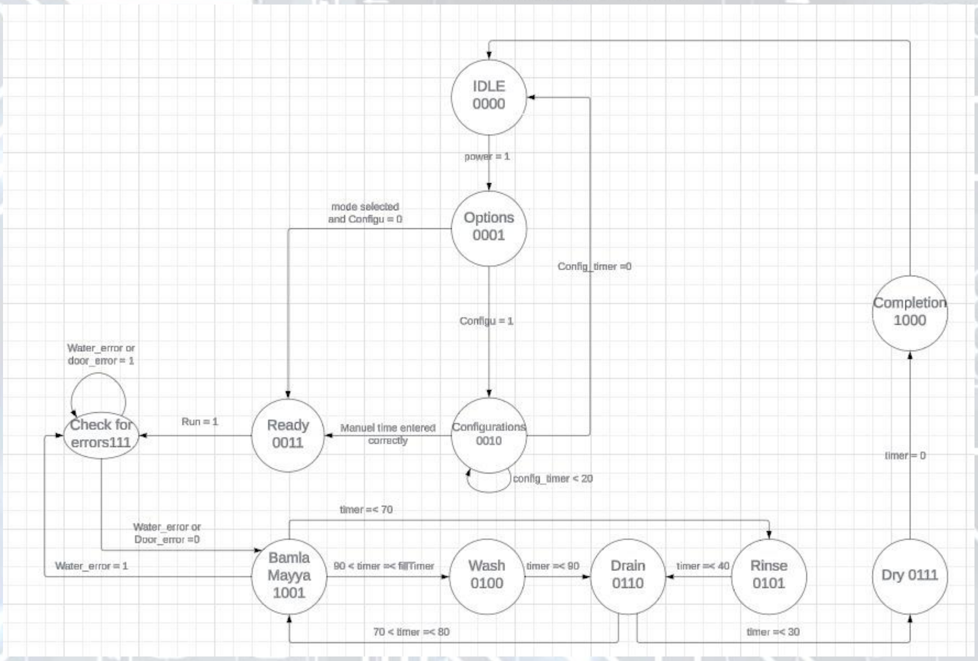
Advantages of design:

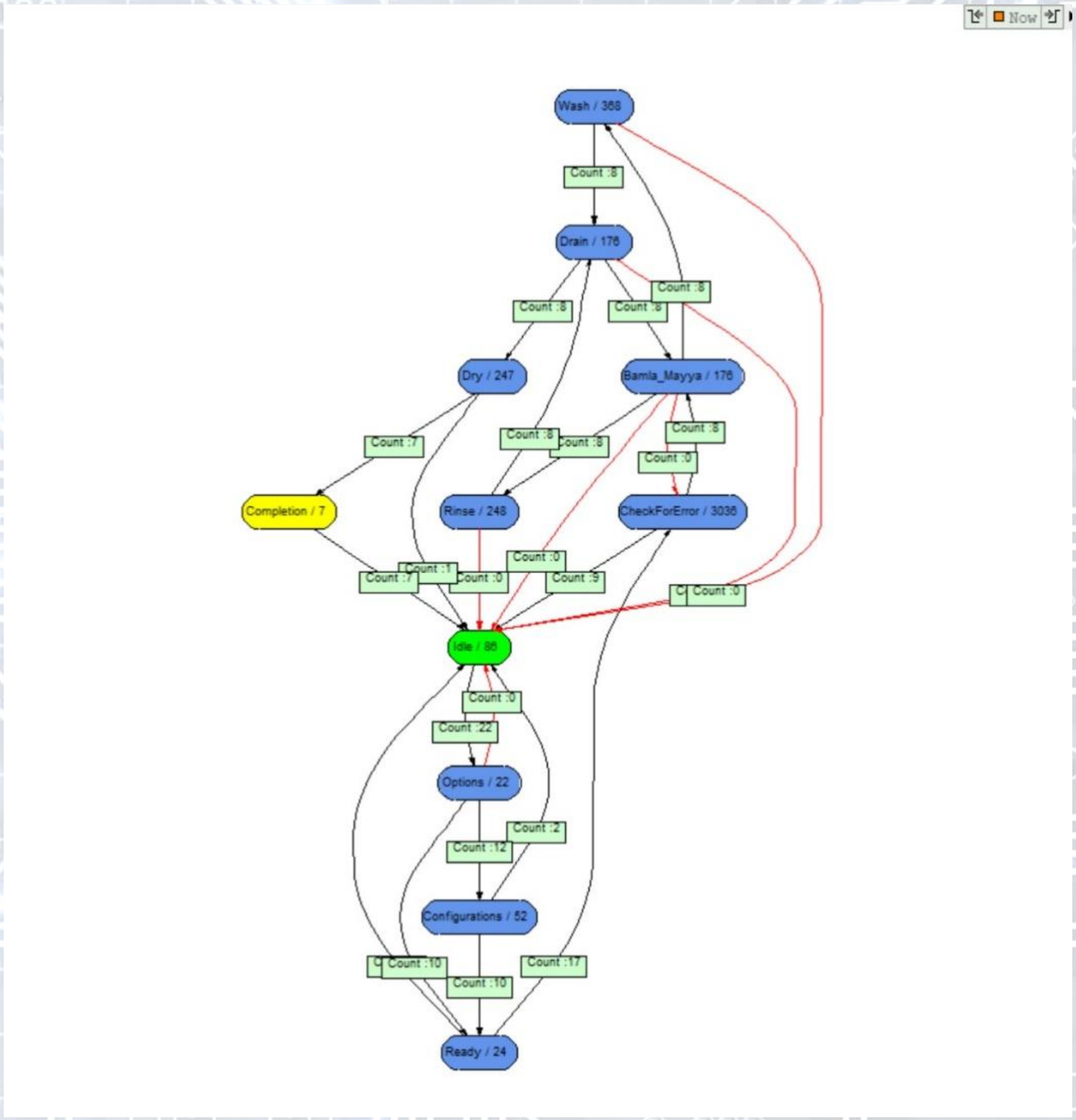
- Scalability: Additional states and modes can be added easily.
- Error Resilience: Ensures safe operation by pausing during errors.
- Flexibility: Supports manual configurations and multiple preset modes.
- Energy Efficiency: Automatically powers off after completion or timeout.

Design considerations:

- All state transitions are synchronized with the clock.
- Errors are prioritized over normal transitions.
- Manual timer inputs are validated to avoid invalid settings.
- Default mode (QuickWash) is used as a fallback when no mode is selected.

Finite state machine diagram: -

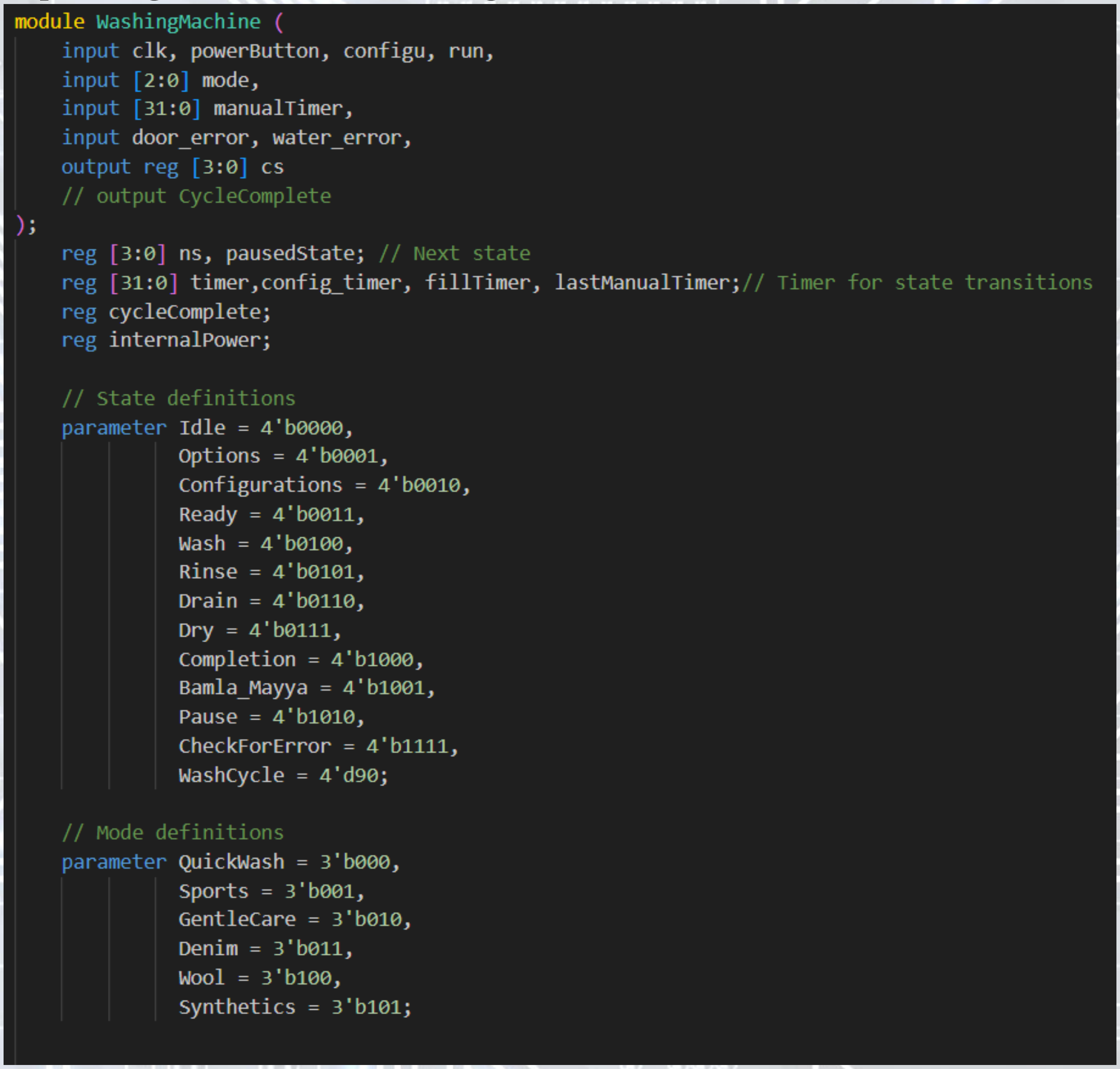




This is the FSM created by Questasim after detecting it during the run time.

Verilog Code:

In the following section we will be discussing and explaining the code for the design submitted.



```verilog
module WashingMachine (
    input clk, powerButton, configu, run,
    input [2:0] mode,
    input [31:0] manualTimer,
    input door_error, water_error,
    output reg [3:0] cs
    // output CycleComplete
);
    reg [3:0] ns, pausedState; // Next state
    reg [31:0] timer,config_timer, fillTimer, lastManualTimer;// Timer for state transitions
    reg cycleComplete;
    reg internalPower;

    // State definitions
    parameter Idle = 4'b0000,
              Options = 4'b0001,
              Configurations = 4'b0010,
              Ready = 4'b0011,
              Wash = 4'b0100,
              Rinse = 4'b0101,
              Drain = 4'b0110,
              Dry = 4'b0111,
              Completion = 4'b1000,
              Bamla_Mayya = 4'b1001,
              Pause = 4'b1010,
              CheckForError = 4'b1111,
              WashCycle = 4'd90;

    // Mode definitions
    parameter QuickWash = 3'b000,
              Sports = 3'b001,
              GentleCare = 3'b010,
              Denim = 3'b011,
              Wool = 3'b100,
              Synthetics = 3'b101;
```

Inputs, signals and parameters: -

Inputs:

Power button: which the user will press to turn on the washing machine or pause the ongoing cycle.

Configu: a one-bit input responsible for determining whether the user will enter a custom time or choose from the available modes.

Run: a one-bit input acting like a start button where the user orders the machine to start the cycle after choosing the preferred modes.

Mode: Contains 6 types of modes offered by the machine which are:

- Quick wash
- Sports
- Gentle care
- Denim
- Wool
- Synthetics

Manuel timer: the custom time that the user desires the cycle to last entered as in decimal form

Door_error: one bit input which would be connected to a sensor that detects whether the machine's door is open or not.

Water_error: one bit input which would be connected to a sensor that detects whether there is enough water flowing in the machine.

CS: to store the current state that the machine is in.

Signals:

NS: responsible for transitioning the machine to the next state.

PausedState: in case of pause state being invoked we will need to save the current state in order to go back to it when the cycle is resumed.

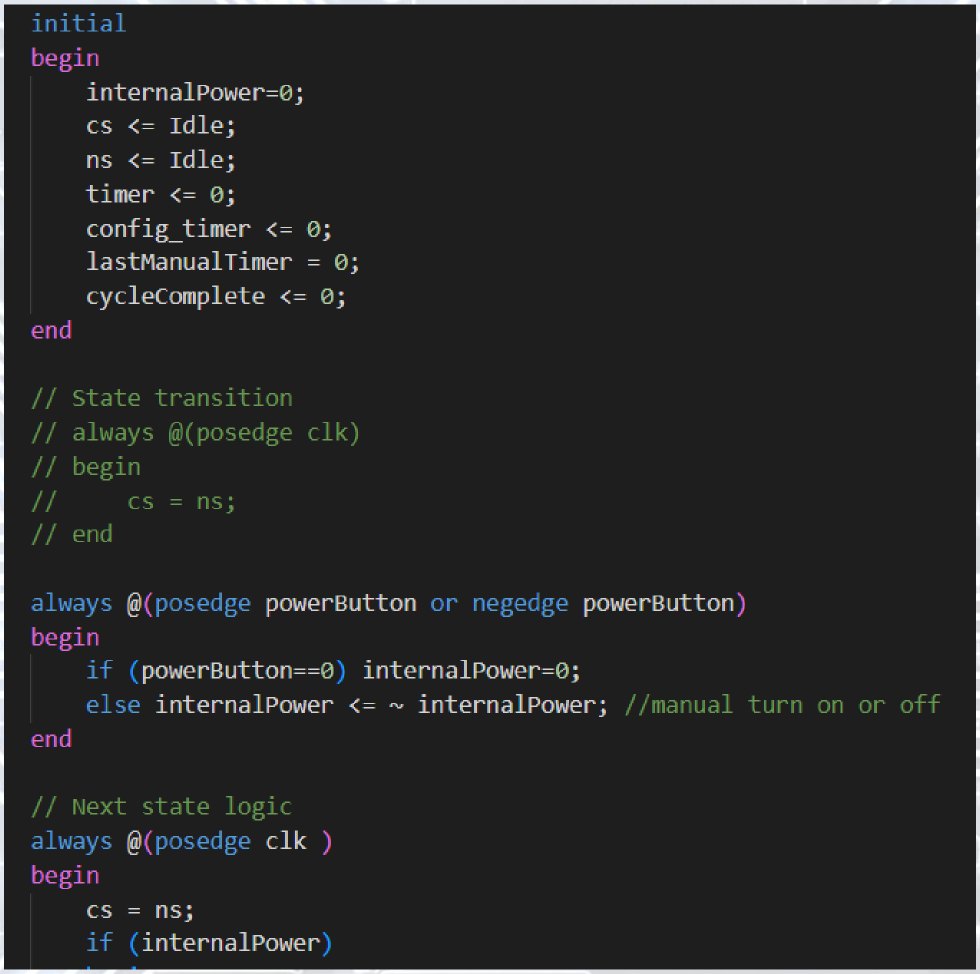
Timer: the register responsible for decrementing and changing the time.

Config_timer: responsible for counting down a time of 20-time units and resetting the washing machine.

Cyclecomplete: an output of one bit indicating the end of our cycle.

Internalpower: signal responsible for turning on and off our machine without interfering with the input power button.

Clock and power:

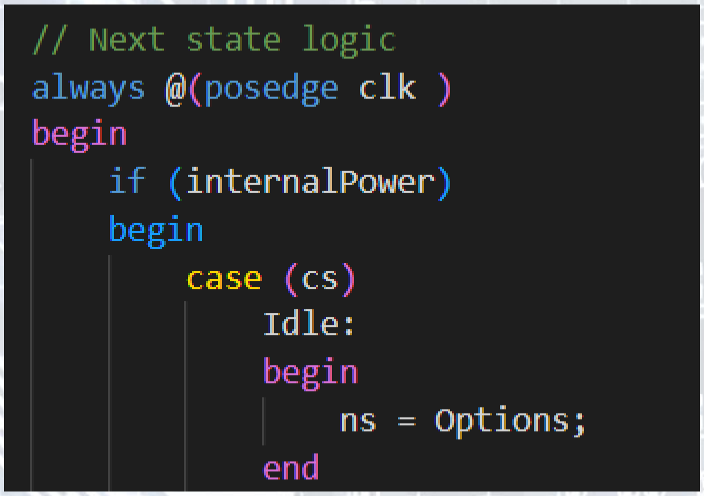


```
initial
begin
    internalPower=0;
    cs <= Idle;
    ns <= Idle;
    timer <= 0;
    config_timer <= 0;
    lastManualTimer = 0;
    cycleComplete <= 0;
end

// State transition
// always @(posedge clk)
// begin
//     cs = ns;
// end

always @(posedge powerButton or negedge powerButton)
begin
    if (powerButton==0) internalPower=0;
    else internalPower <= ~ internalPower; //manual turn on or off
end

// Next state logic
always @(posedge clk )
begin
    cs = ns;
    if (internalPower)
```

The powerButton is used as a trigger for internalPower, which is the main driver of the washing machine, when the powerButton's value changes it reflects the change on internalPower.

Next state logic:

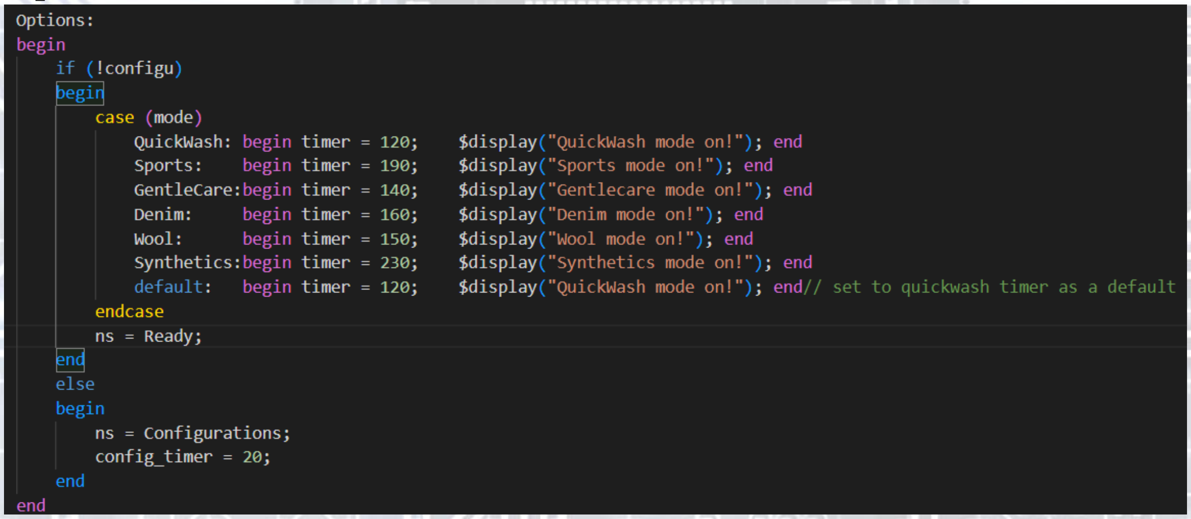
Idle



```
// Next state logic
always @(posedge clk )
begin
    if (internalPower)
    begin
        case (cs)
            Idle:
            begin
                ns = Options;
            end
```

The washing machine starts by checking whether the internal power register is one or zero if it's one then it starts by going to the IDLE state which goes to the options state.

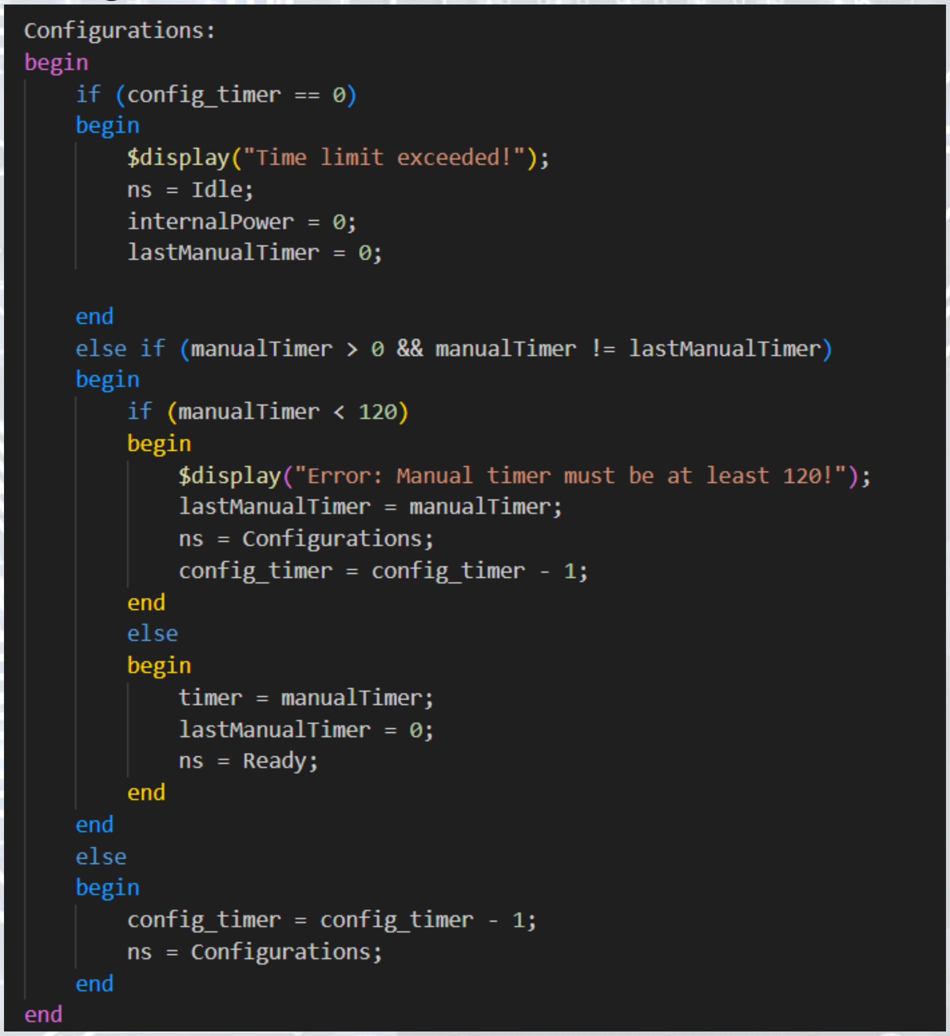
Options



```
Options:
begin
    if (!configu)
    begin
        case (mode)
            QuickWash: begin timer = 120;    $display("QuickWash mode on!"); end
            Sports:    begin timer = 190;    $display("Sports mode on!"); end
            GentleCare:begin timer = 140;    $display("Gentlecare mode on!"); end
            Denim:     begin timer = 160;    $display("Denim mode on!"); end
            Wool:      begin timer = 150;    $display("Wool mode on!"); end
            Synthetics:begin timer = 230;    $display("Synthetics mode on!"); end
            default:   begin timer = 120;    $display("QuickWash mode on!"); end// set to quickwash timer as a default
        endcase
        ns = Ready;
    end
    else
    begin
        ns = Configurations;
        config_timer = 20;
    end
end
```

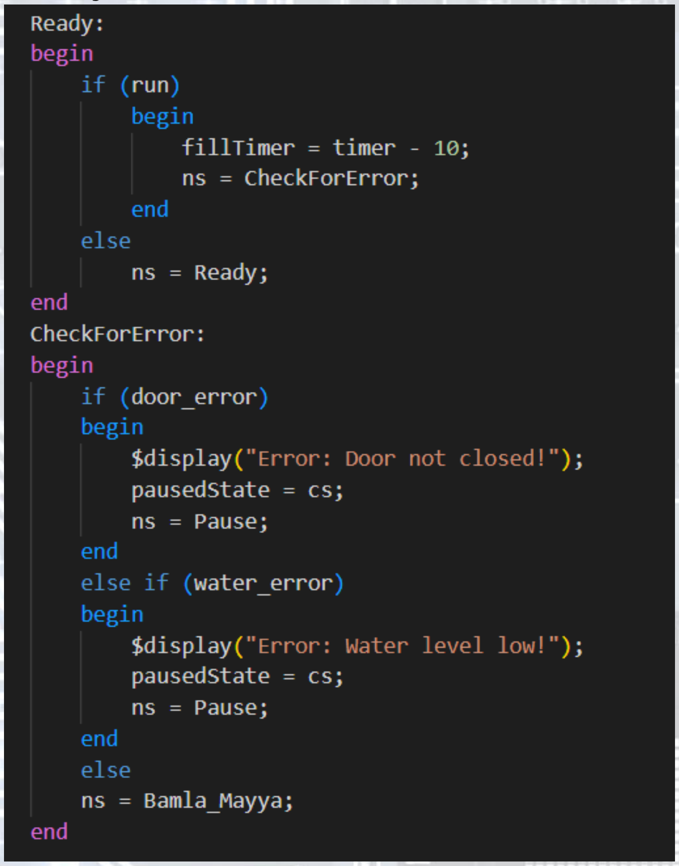
During the options state we start by checking if the configuration bit is equal to 0. If it is equal to 0 then the user will be selecting from the provided modes stated above, then it goes to the ready state. Otherwise, we initialize config_timer with 20 and go to next state configurations.

Configurations state:


```
Configurations:
begin
    if (config_timer == 0)
    begin
        $display("Time limit exceeded!");
        ns = Idle;
        internalPower = 0;
        lastManualTimer = 0;

    end
    else if (manualTimer > 0 && manualTimer != lastManualTimer)
    begin
        if (manualTimer < 120)
        begin
            $display("Error: Manual timer must be at least 120!");
            lastManualTimer = manualTimer;
            ns = Configurations;
            config_timer = config_timer - 1;
        end
        else
        begin
            timer = manualTimer;
            lastManualTimer = 0;
            ns = Ready;
        end
    end
    else
    begin
        config_timer = config_timer - 1;
        ns = Configurations;
    end
end
```

During the configuration state, we are waiting for an input timer from the user "manualTimer", we check the validity of the timer, where if the manualTimer is less than 120, an error message is displayed, otherwise the timer is set to manualTimer, to make sure we only display the error once, we use reg lastManualTimer to check if a new timer was set. After each clock cycle if the timer was not set properly, a countdown is activated and once it reaches 0, the machine goes back to idle state and turns internalPower off.

Ready state:

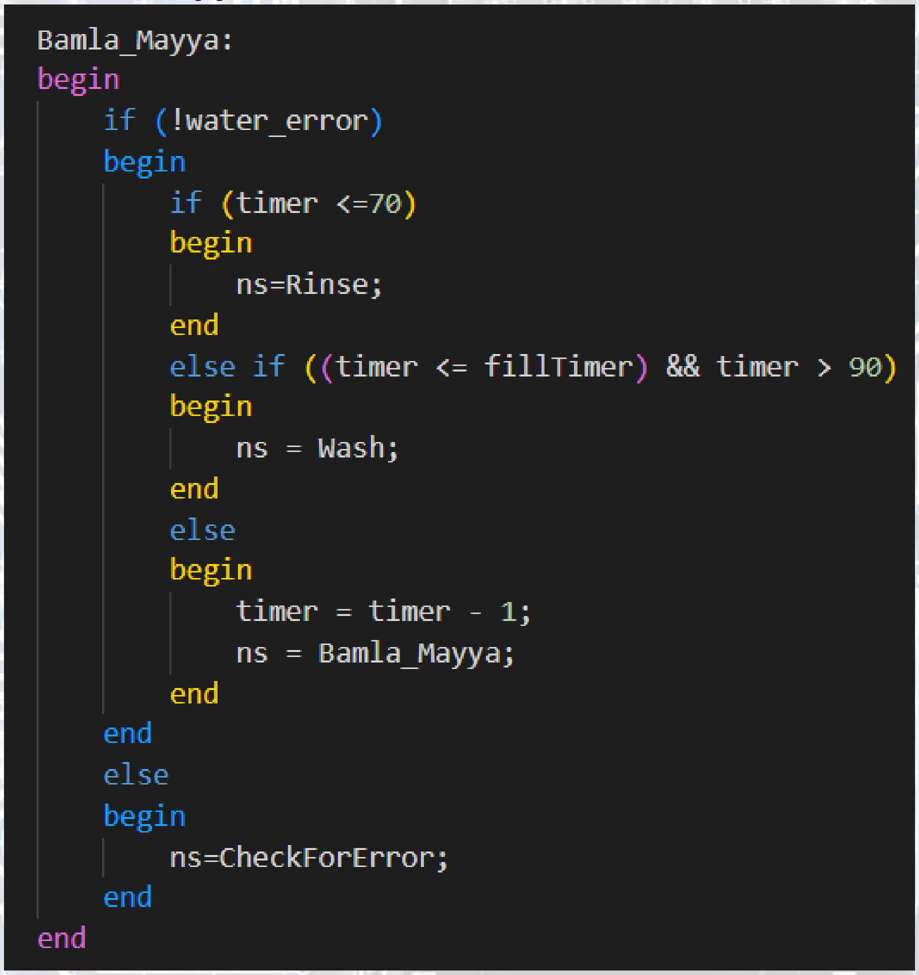


```
Ready:
begin
    if (run)
        begin
            fillTimer = timer - 10;
            ns = CheckForError;
        end
    else
        ns = Ready;
end
CheckForError:
begin
    if (door_error)
    begin
        $display("Error: Door not closed!");
        pausedState = cs;
        ns = Pause;
    end
    else if (water_error)
    begin
        $display("Error: Water level low!");
        pausedState = cs;
        ns = Pause;
    end
    else
    ns = Bamla_Mayya;
end
```

In the ready state, the washing machine is waiting for the input run to be 1 to start its cycle by moving to CheckForError state and introducing a new parameter fillTimer which equals the initial timer – 10. Otherwise, while the run input is still 0, the washing machine stays in the Ready state.

In check for errors state if either the door error or water error are equal to 1, then we display an error message. After that we move on to pause state and storing our current state in pausedState register to get back to it after the error is resolved. However, if no error is detected we move on to the Bamla mayya state.

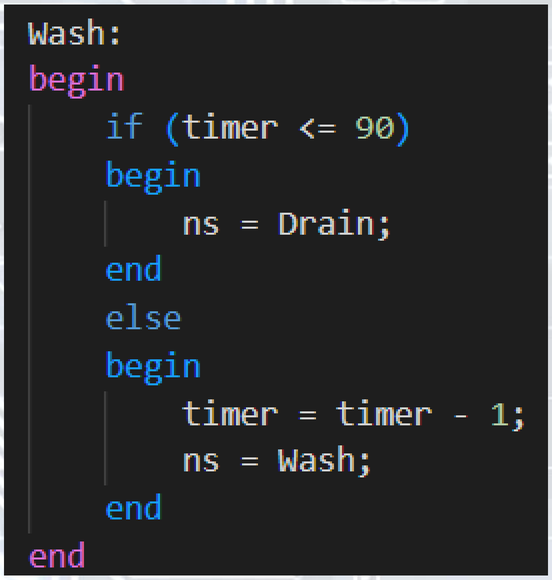
Bamla_mayya state:



```
Bamla_Mayya:
begin
    if (!water_error)
    begin
        if (timer <=70)
        begin
            ns=Rinse;
        end
        else if ((timer <= fillTimer) && timer > 90)
        begin
            ns = Wash;
        end
        else
        begin
            timer = timer - 1;
            ns = Bamla_Mayya;
        end
    end
    else
    begin
        ns=CheckForError;
    end
end
```

Since the Bamla_Mayya state is going to be visited twice we made two if conditions. The first one which is if the timer is more than 90 and less than fill timer which is the time taken to fill water in directs the code to the wash state after 10 seconds have passed. The second if condition is if the timer is less than or equal to 70 indicating that the first drain cycle is complete thus filling water again and redirecting to the rinse state. If neither condition has been invoked the Bamla_Mayya state will keep decrementing the timer.

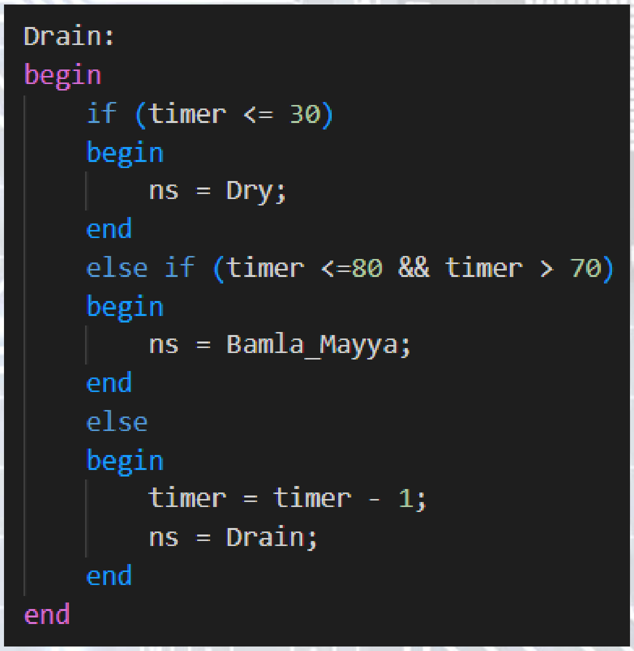
Wash state:



```
Wash:
begin
    if (timer <= 90)
    begin
        ns = Drain;
    end
    else
    begin
        timer = timer - 1;
        ns = Wash;
    end
end
```

In the Bamla_Mayya state we saw the if else condition that gets us to the Wash state. After the Wash state is entered, it keeps decrementing the wash time that is required until timer is less than or equal to 90. Then We will go to drain state, as the wash timer is finished.

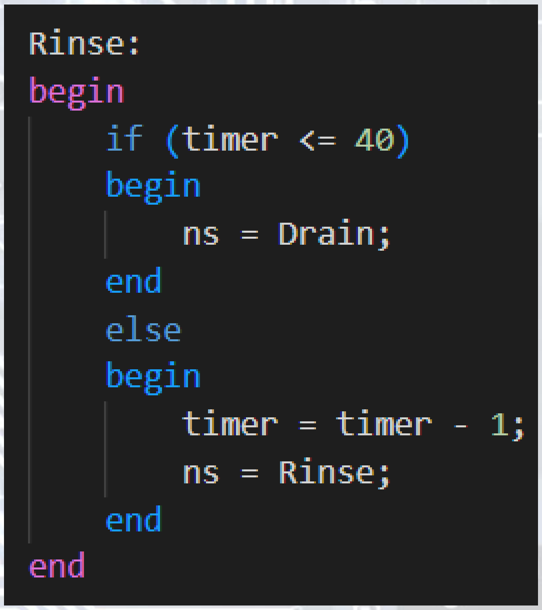
Drain state:



```
Drain:
begin
    if (timer <= 30)
    begin
        ns = Dry;
    end
    else if (timer <=80 && timer > 70)
    begin
        ns = Bamla_Mayya;
    end
    else
    begin
        timer = timer - 1;
        ns = Drain;
    end
end
```

Like the Bamla_Mayya state, the drain state is visited twice, once after the wash state and the other after the drain state to refill water for rinsing. The state checks whether the timer is between 80 and 70 and keeps decrementing the timer till the 70 mark is reached and moved on to the Bamla_Mayya state. The second time it checks if the timer is less than or equal to 30 and moves to the dry state.

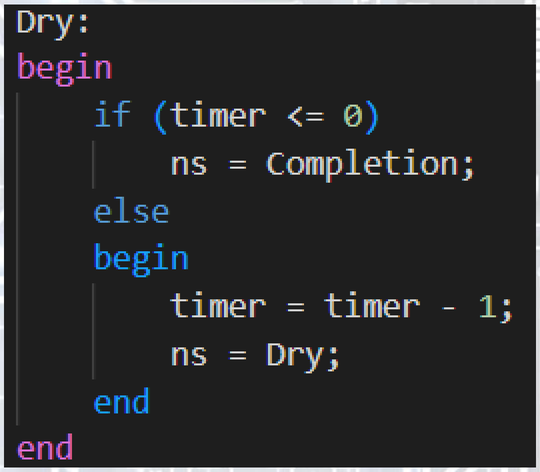
Rinse state:



```
Rinse:
begin
    if (timer <= 40)
    begin
        ns = Drain;
    end
    else
    begin
        timer = timer - 1;
        ns = Rinse;
    end
end
```

The rinse state is visited once after the Bamla_Mayya state and works for 30-time units and moves to the Drain state.

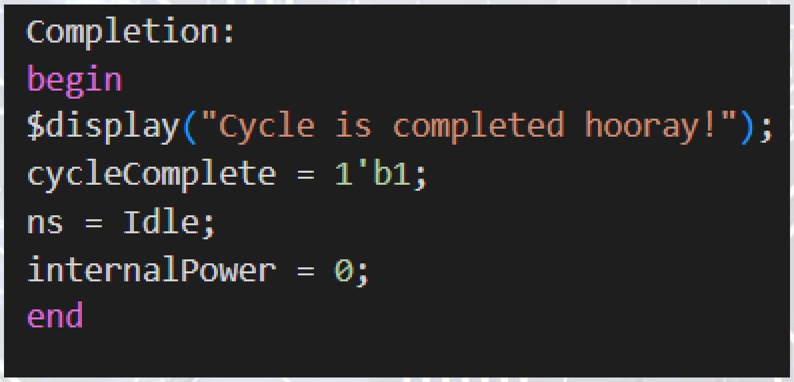
Dry state:



```
Dry:
begin
    if (timer <= 0)
        ns = Completion;
    else
    begin
        timer = timer - 1;
        ns = Dry;
    end
end
```

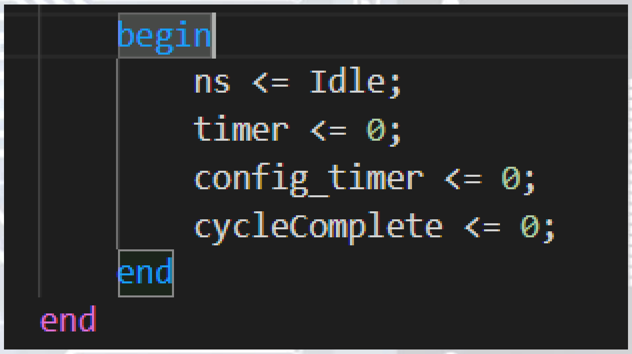
The dry state lasts for the remainder of the timer and when the timer is equal to 0 it moves to the final state completion.

Completion state**:**



```
Completion:
begin
$display("Cycle is completed hooray!");
cycleComplete = 1'b1;
ns = Idle;
internalPower = 0;
end
```

The final state completion is visited after the dry state and it displays “Cycle is completed hooray!” setting output cycle Complete to one, moving back to IDLE and turning off the internal power.



```
      begin
          ns <= Idle;
          timer <= 0;
          config_timer <= 0;
          cycleComplete <= 0;
      end
 end
```

This part is for resetting the important registers and moving the machine to IDLE state in case of powerButton turning off or the config_timer has reached zero indicating that 20 time units have passed without a manualtimer being entered turning interalpower to 0.

Verification plan: -

Our objective is the test case file is to verify the functionality and capability of the Washing Machine module under different operating conditions, including Directed, constrained and random cases.

1. **Directed Tests:**
   - Test basic operations like power-on, mode selection, and timer configuration.
   - Validate error handling (e.g., door error, water error).
   - Ensure FSM transitions work as expected.
2. **Random Tests:**
   - Generate random inputs to test for unexpected behaviors.
   - Verify stability of the system under diverse configurations.
3. **Constrained Random Tests:**
   - Constrain input ranges to ensure meaningful test cases (e.g., valid modes and timer ranges).
   - Validate FSM stability and correctness within these constraints.
4. **Error Scenarios:**
   - Simulate transient errors using door_error_trigger task:
     - Introduce door errors during operation.
     - Verify recovery and system stability after errors are resolved.

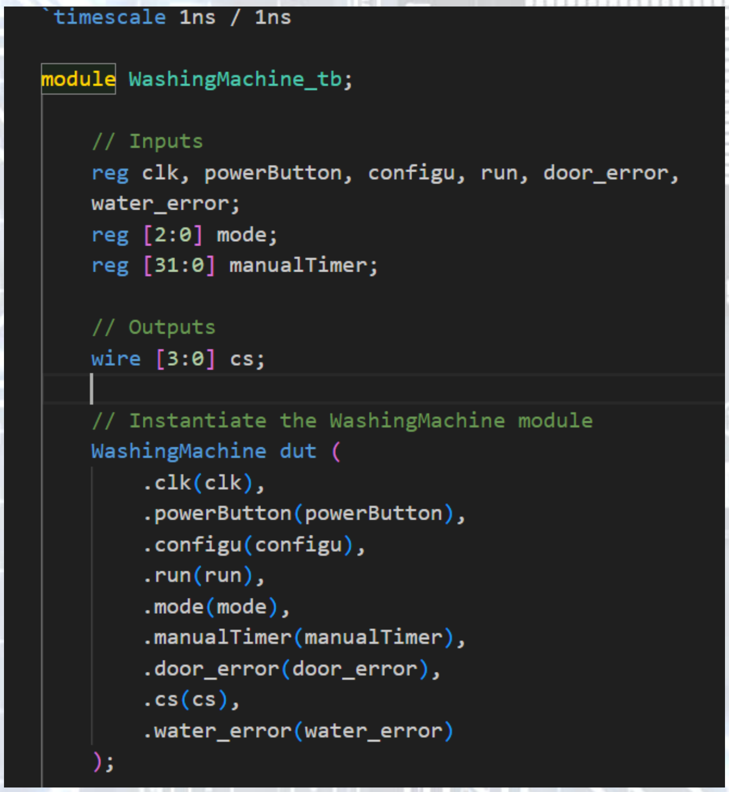
5. **Monitoring and Coverage:**

- Use $monitor to log critical signals like FSM states, timers, and error flags.
- Aim for complete state and transition coverage for the FSM.

Test bench: -

In the test bench. We covered multiple scenarios considering the logic of the code, (**Directed**, **Constrained random** and **random**). Using these types of test cases will cover all the cases where the code works normally and testing the code behavior in odd cases.

Module declaration and I/O:

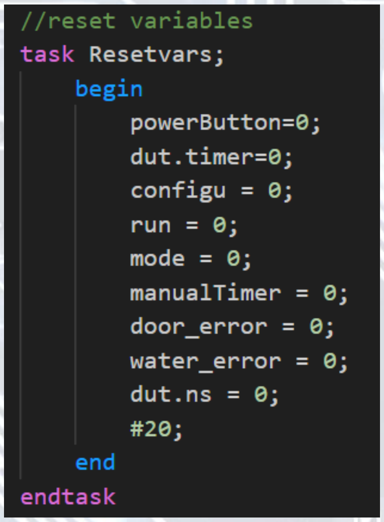


```
`timescale 1ns / 1ns

module WashingMachine_tb;

    // Inputs
    reg clk, powerButton, configu, run, door_error,
    water_error;
    reg [2:0] mode;
    reg [31:0] manualTimer;

    // Outputs
    wire [3:0] cs;

    // Instantiate the WashingMachine module
    WashingMachine dut (
        .clk(clk),
        .powerButton(powerButton),
        .configu(configu),
        .run(run),
        .mode(mode),
        .manualTimer(manualTimer),
        .door_error(door_error),
        .cs(cs),
        .water_error(water_error)
    );
```

We declared the inputs and outputs of the module of the washing machine in the test bench code under the DUT (design under test)

We then created some tasks to easily instantiate from it, for the code to be readable.

Reset Task:

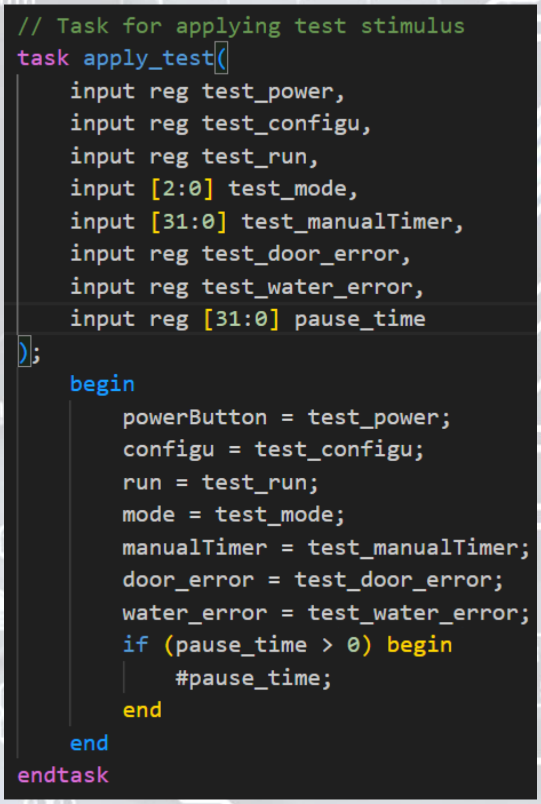


```
//reset variables
task Resetvars;
    begin
        powerButton=0;
        dut.timer=0;
        configu = 0;
        run = 0;
        mode = 0;
        manualTimer = 0;
        door_error = 0;
        water_error = 0;
        dut.ns = 0;
        #20;
    end
endtask
```

We created the reset task to use it before any test case to ensure a clean start and start from a known state for accurate results, so we set all the inputs to zero, and put a delay of 20 to ensure there is no delay and for the output to be clear.

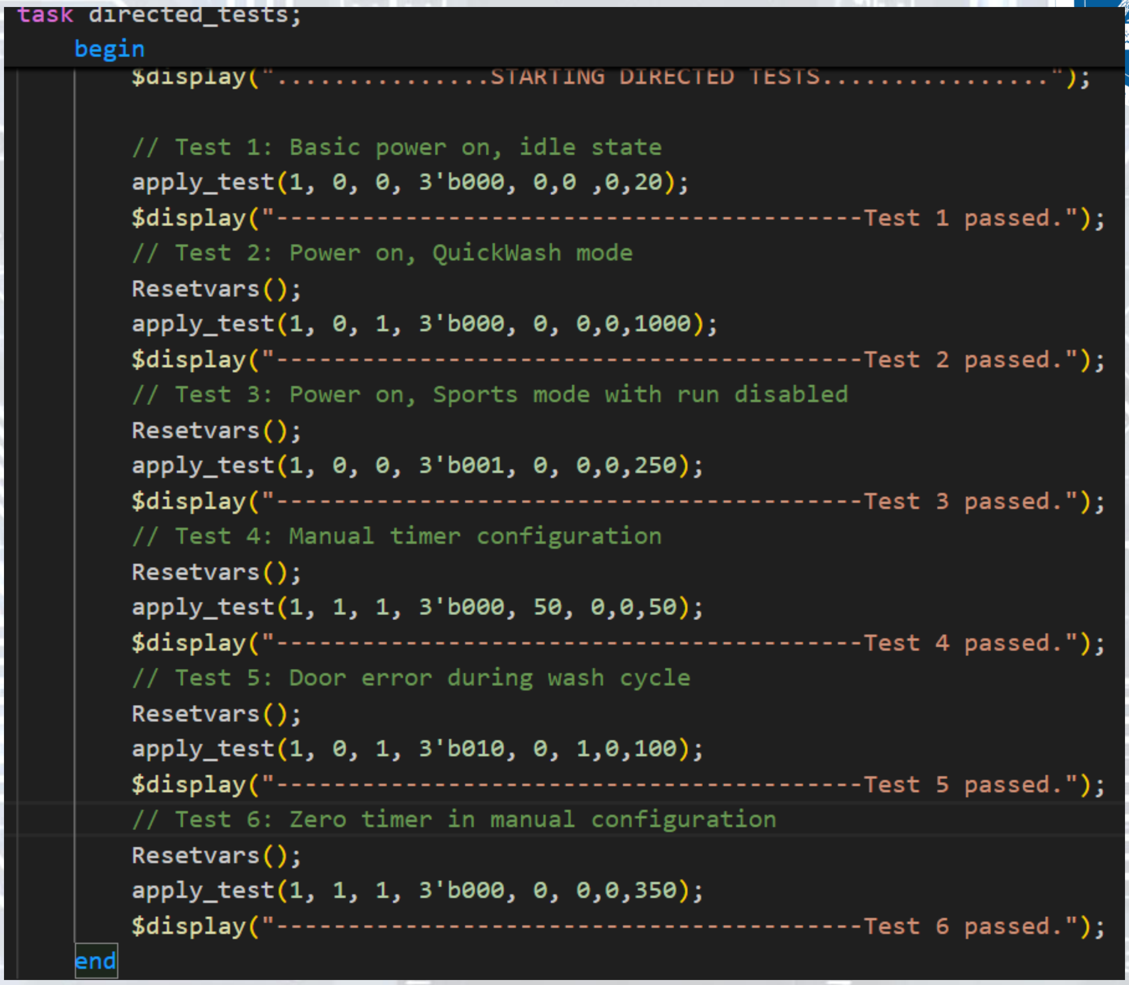
Application task:

This is the task responsible for all the test cases. We created new variables, the same as the variables declared in the module and in the begin, we assigned these variables to the variables in the washing machine module.



```
// Task for applying test stimulus
task apply_test(
    input reg test_power,
    input reg test_configu,
    input reg test_run,
    input [2:0] test_mode,
    input [31:0] test_manualTimer,
    input reg test_door_error,
    input reg test_water_error,
    input reg [31:0] pause_time
);
    begin
        powerButton = test_power;
        configu = test_configu;
        run = test_run;
        mode = test_mode;
        manualTimer = test_manualTimer;
        door_error = test_door_error;
        water_error = test_water_error;
        if (pause_time > 0) begin
            #pause_time;
        end
    end
endtask
```

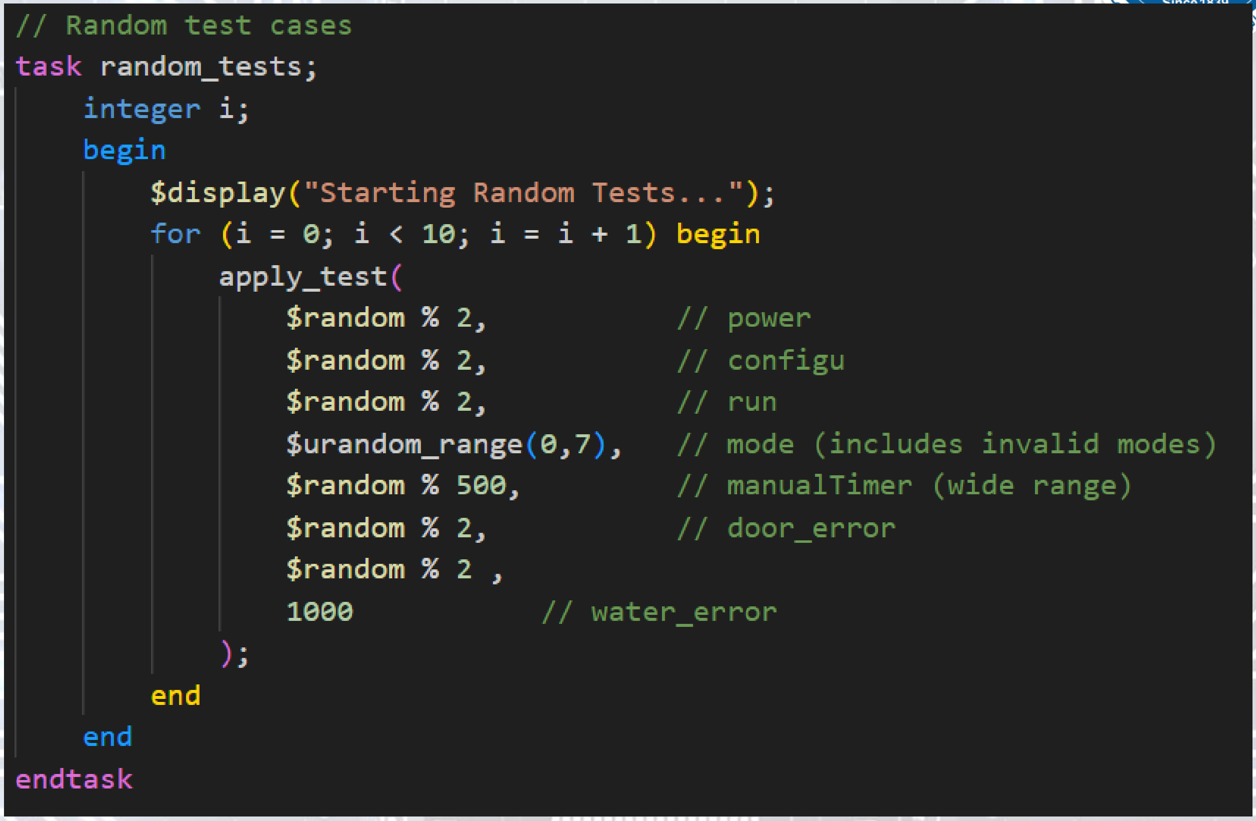
Directed test cases:



```
task directed_tests;
    begin
        $display("...............STARTING DIRECTED TESTS................");

        // Test 1: Basic power on, idle state
        apply_test(1, 0, 0, 3'b000, 0,0 ,0,20);
        $display("-----------------------------------------Test 1 passed.");
        // Test 2: Power on, QuickWash mode
        Resetvars();
        apply_test(1, 0, 1, 3'b000, 0, 0,0,1000);
        $display("-----------------------------------------Test 2 passed.");
        // Test 3: Power on, Sports mode with run disabled
        Resetvars();
        apply_test(1, 0, 0, 3'b001, 0, 0,0,250);
        $display("-----------------------------------------Test 3 passed.");
        // Test 4: Manual timer configuration
        Resetvars();
        apply_test(1, 1, 1, 3'b000, 50, 0,0,50);
        $display("-----------------------------------------Test 4 passed.");
        // Test 5: Door error during wash cycle
        Resetvars();
        apply_test(1, 0, 1, 3'b010, 0, 1,0,100);
        $display("-----------------------------------------Test 5 passed.");
        // Test 6: Zero timer in manual configuration
        Resetvars();
        apply_test(1, 1, 1, 3'b000, 0, 0,0,350);
        $display("-----------------------------------------Test 6 passed.");
    end
```

In the directed test case, we created multiple instances of the apply_test task and in the parameters, we created different scenarios and see how the code behaves. The directed test case is used in normal test cases like how the user will use the washing machine normally. Like turning on the washing machine, then choosing a mode, and so on. And as we said, we used the reset task before any test case.

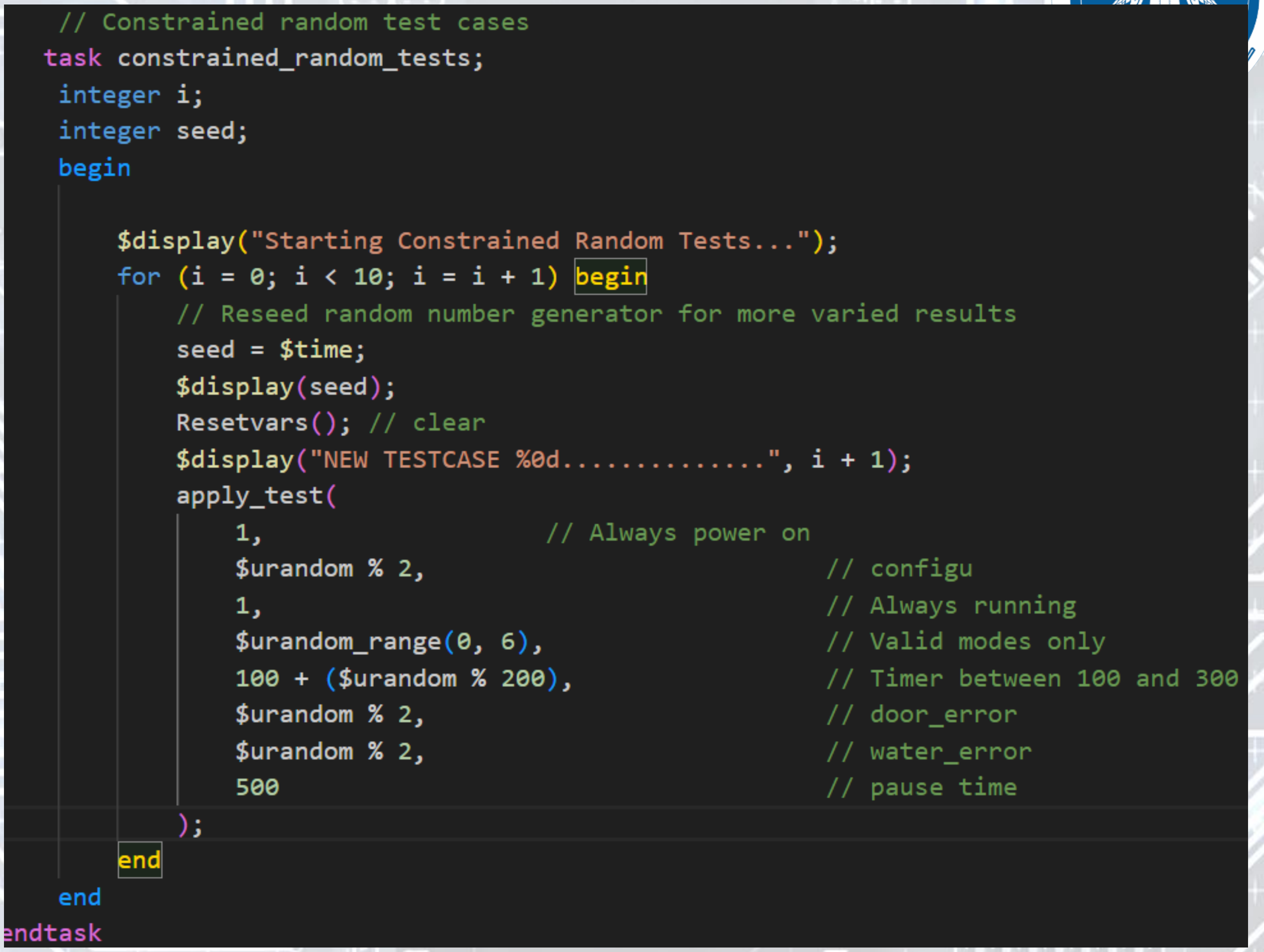
Random test cases:



```
// Random test cases
task random_tests;
    integer i;
    begin
        $display("Starting Random Tests...");
        for (i = 0; i < 10; i = i + 1) begin
            apply_test(
                $random % 2,            // power
                $random % 2,            // configu
                $random % 2,            // run
                $urandom_range(0,7),    // mode (includes invalid modes)
                $random % 500,          // manualTimer (wide range)
                $random % 2,            // door_error
                $random % 2 ,
                1000            // water_error
            );
        end
    end
endtask
```

In the random test cases, we defined integer i as the for-loop iterator, to apply multiples tests, then we called the apply_test() again but the parameters in the task are declared as random using **$random**, we don’t know the values until we run the code, this ensures that we can see how to program behaves in the odd cases and ensure a bug free program that can handle all error behavior from the user. The (random %2) for example ensures that the range of the numbers to generate from is 2 bits only, same for other inputs.

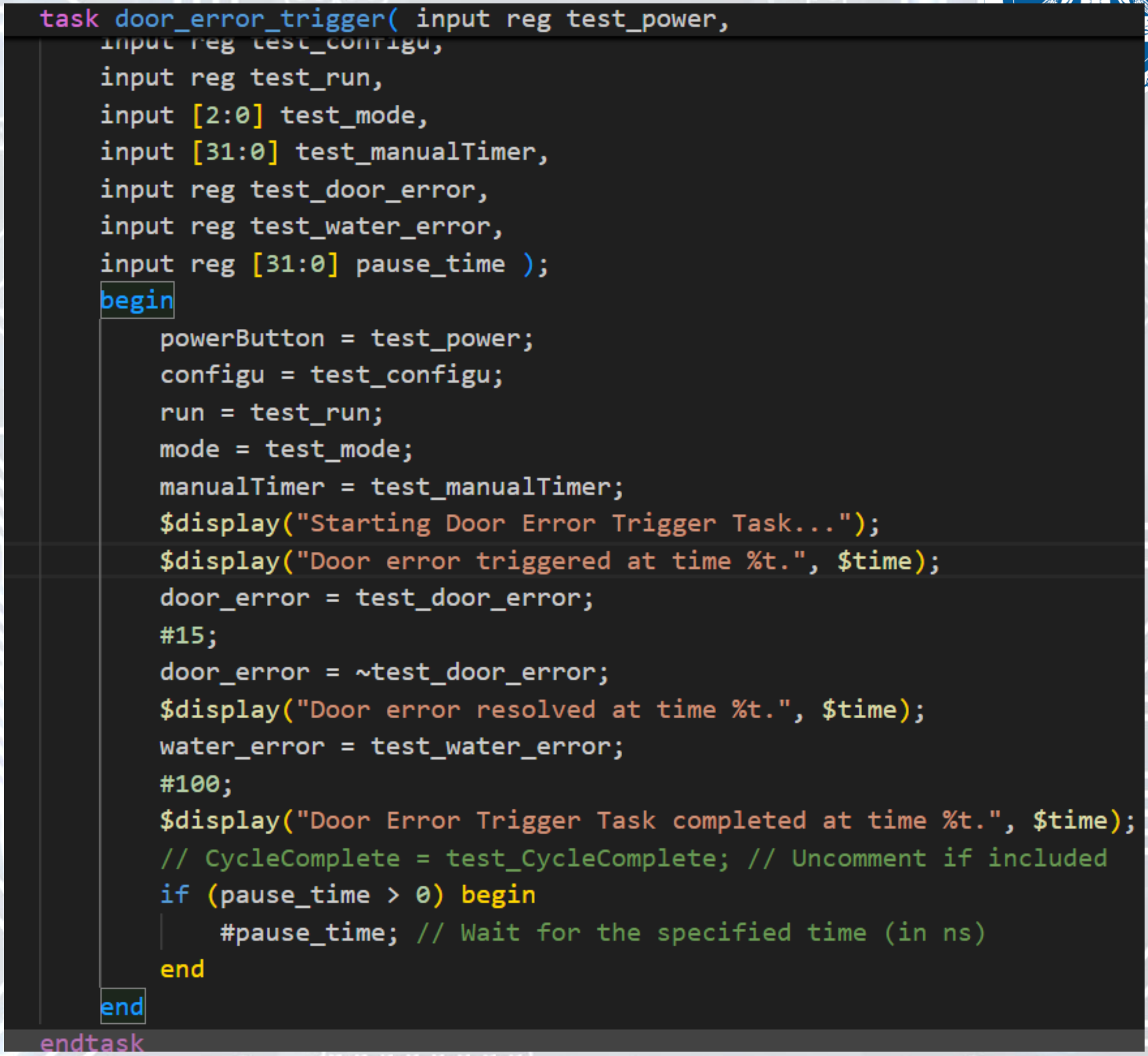
Constrained random test cases:



```
  // Constrained random test cases
 task constrained_random_tests;
  integer i;
  integer seed;
  begin

      $display("Starting Constrained Random Tests...");
      for (i = 0; i < 10; i = i + 1) begin
          // Reseed random number generator for more varied results
          seed = $time;
          $display(seed);
          Resetvars(); // clear
          $display("NEW TESTCASE %0d.............", i + 1);
          apply_test(
              1,                   // Always power on
              $urandom % 2,                              // configu
              1,                                         // Always running
              $urandom_range(0, 6),                      // Valid modes only
              100 + ($urandom % 200),                    // Timer between 100 and 300
              $urandom % 2,                              // door_error
              $urandom % 2,                              // water_error
              500                                        // pause time
          );
      end
  end
endtask
```

The constrained random test cases are like the random test cases, but there are some inputs that are constant during the test cases and the other values are set to random. This is used to check for different cases while fixing some inputs. For example, we set the values of power button to (1). It's always 1 in these test cases, same for the running, other inputs are random, so we make sure that the washing machine is always on and it's running, but other options are random.

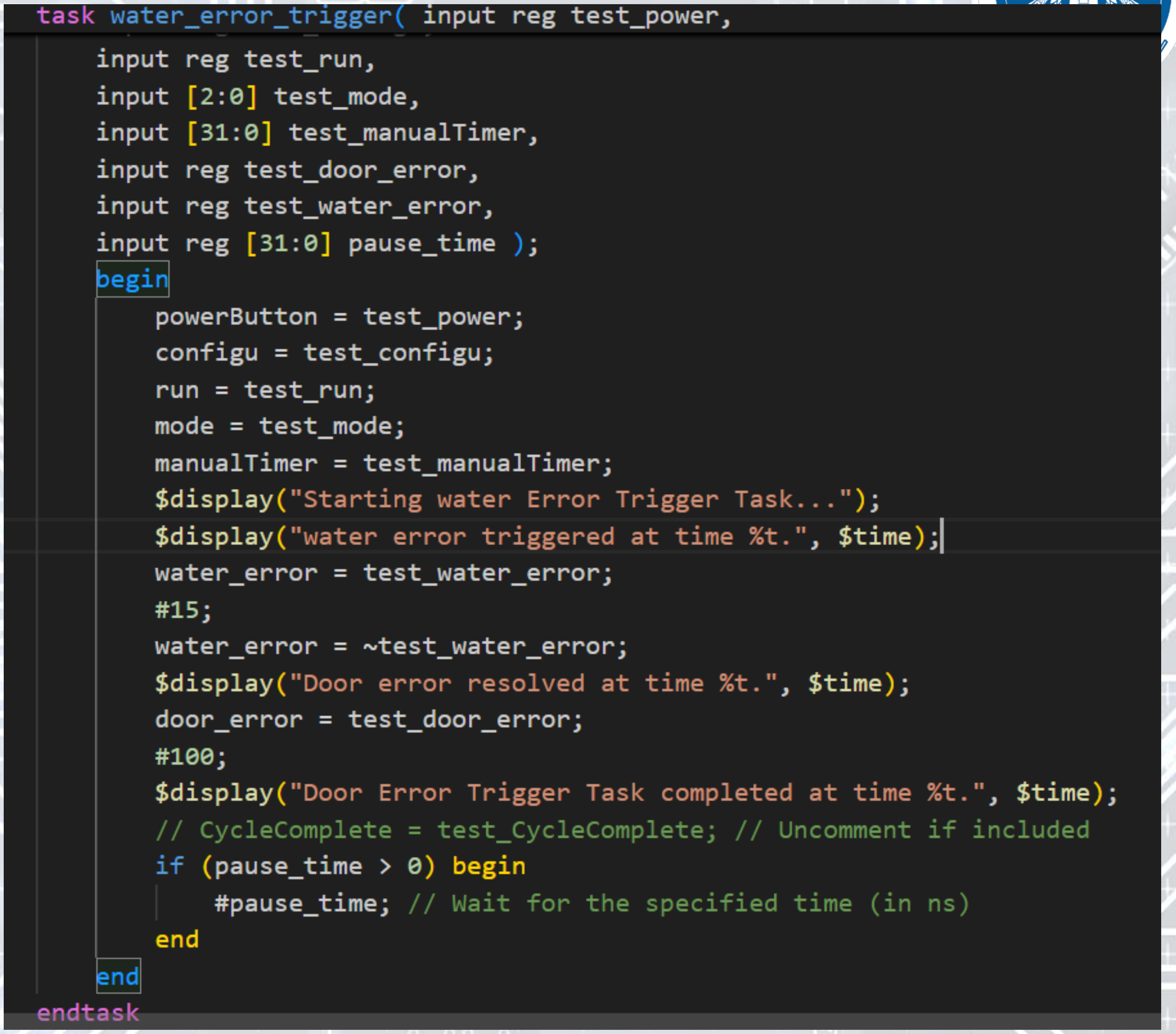
Door error Trigger task:



```
task door_error_trigger( input reg test_power,
    input reg test_configu,
    input reg test_run,
    input [2:0] test_mode,
    input [31:0] test_manualTimer,
    input reg test_door_error,
    input reg test_water_error,
    input reg [31:0] pause_time );
    begin
        powerButton = test_power;
        configu = test_configu;
        run = test_run;
        mode = test_mode;
        manualTimer = test_manualTimer;
        $display("Starting Door Error Trigger Task...");
        $display("Door error triggered at time %t.", $time);
        door_error = test_door_error;
        #15;
        door_error = ~test_door_error;
        $display("Door error resolved at time %t.", $time);
        water_error = test_water_error;
        #100;
        $display("Door Error Trigger Task completed at time %t.", $time);
        // CycleComplete = test_CycleComplete; // Uncomment if included
        if (pause_time > 0) begin
            #pause_time; // Wait for the specified time (in ns)
        end
    end
endtask
```

In this task, we tested the alternating door error to see how the code behaves, we defined parameters, and then alternated the door error and put a 15-time delay between the alternative triggers.

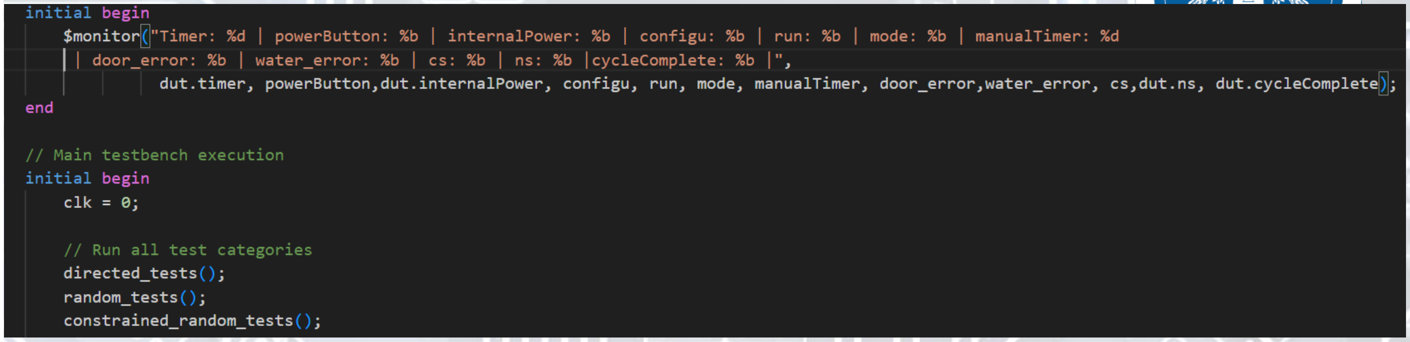
Water error task trigger:



```
task water_error_trigger( input reg test_power,
    input reg test_run,
    input [2:0] test_mode,
    input [31:0] test_manualTimer,
    input reg test_door_error,
    input reg test_water_error,
    input reg [31:0] pause_time );
    begin
        powerButton = test_power;
        configu = test_configu;
        run = test_run;
        mode = test_mode;
        manualTimer = test_manualTimer;
        $display("Starting water Error Trigger Task...");
        $display("water error triggered at time %t.", $time);
        water_error = test_water_error;
        #15;
        water_error = ~test_water_error;
        $display("Door error resolved at time %t.", $time);
        door_error = test_door_error;
        #100;
        $display("Door Error Trigger Task completed at time %t.", $time);
        // CycleComplete = test_CycleComplete; // Uncomment if included
        if (pause_time > 0) begin
            #pause_time; // Wait for the specified time (in ns)
        end
    end
endtask
```

Same for the water error trigger, we toggle the water error flag each duration to see how the code behaves.

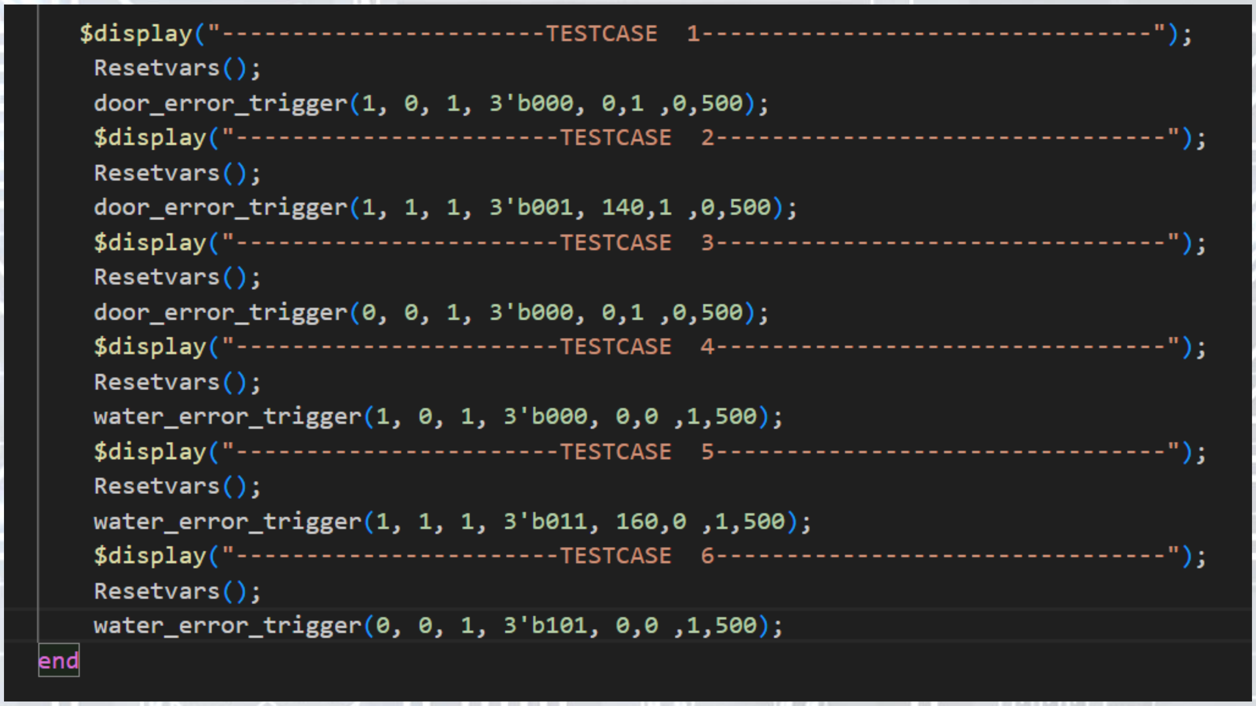
Monitoring outputs:



```
initial begin
    $monitor("Timer: %d | powerButton: %b | internalPower: %b | configu: %b | run: %b | mode: %b | manualTimer: %d
    | door_error: %b | water_error: %b | cs: %b | ns: %b |cycleComplete: %b |",
            dut.timer, powerButton,dut.internalPower, configu, run, mode, manualTimer, door_error,water_error, cs,dut.ns, dut.cycleComplete);
end

// Main testbench execution
initial begin
    clk = 0;

    // Run all test categories
    directed_tests();
    random_tests();
    constrained_random_tests();
```

We printed the outputs of everything in the monitor to trace the cs and ns of every test case to ensure that the code works correctly.

Also we called the directed, constrained and random tasks to run the tests .



```
    $display("----------------------TESTCASE  1------------------------------");
     Resetvars();
     door_error_trigger(1, 0, 1, 3'b000, 0,1 ,0,500);
     $display("----------------------TESTCASE  2------------------------------");
     Resetvars();
     door_error_trigger(1, 1, 1, 3'b001, 140,1 ,0,500);
     $display("----------------------TESTCASE  3------------------------------");
     Resetvars();
     door_error_trigger(0, 0, 1, 3'b000, 0,1 ,0,500);
     $display("----------------------TESTCASE  4------------------------------");
     Resetvars();
     water_error_trigger(1, 0, 1, 3'b000, 0,0 ,1,500);
     $display("----------------------TESTCASE  5------------------------------");
     Resetvars();
     water_error_trigger(1, 1, 1, 3'b011, 160,0 ,1,500);
     $display("----------------------TESTCASE  6------------------------------");
     Resetvars();
     water_error_trigger(0, 0, 1, 3'b101, 0,0 ,1,500);
  end
```

And here we called the door and water error triggers tasks and sent inputs to the parameters.

Code Coverage: -

Coverage Analysis Overview:

The coverage analysis provides insight into the testing thoroughness and the system's verification status. Here's a breakdown of the results across different coverage metrics:

Assertion Coverage:

A total of 21 assertions were defined, and all were successfully covered, achieving perfect 100% coverage. This indicates that the system behaves as expected under all the defined scenarios.

Branch Coverage:

Out of 49 branches, 45 were hit during testing, with 4 misses, resulting in 91.83% coverage. This high coverage demonstrates robust testing of the decision points in the design but highlights a few untested branches that may need further exploration.

Condition Coverage:

All 13 conditions were covered, achieving 100% coverage. This ensures that all logical expressions within the design were thoroughly tested, providing confidence in the correctness of conditional logic.

FSM Coverage:

The finite state machine coverage includes both states and transitions:

FSM States: All 11 states were reached during testing, resulting in 100% coverage, confirming that every state in the design is reachable.

FSM Transitions: Out of 24 transitions, 18 were hit, achieving 75% coverage. This indicates that while all states are covered, not all transitions were exercised, suggesting potential areas for further testing.

Statement Coverage:

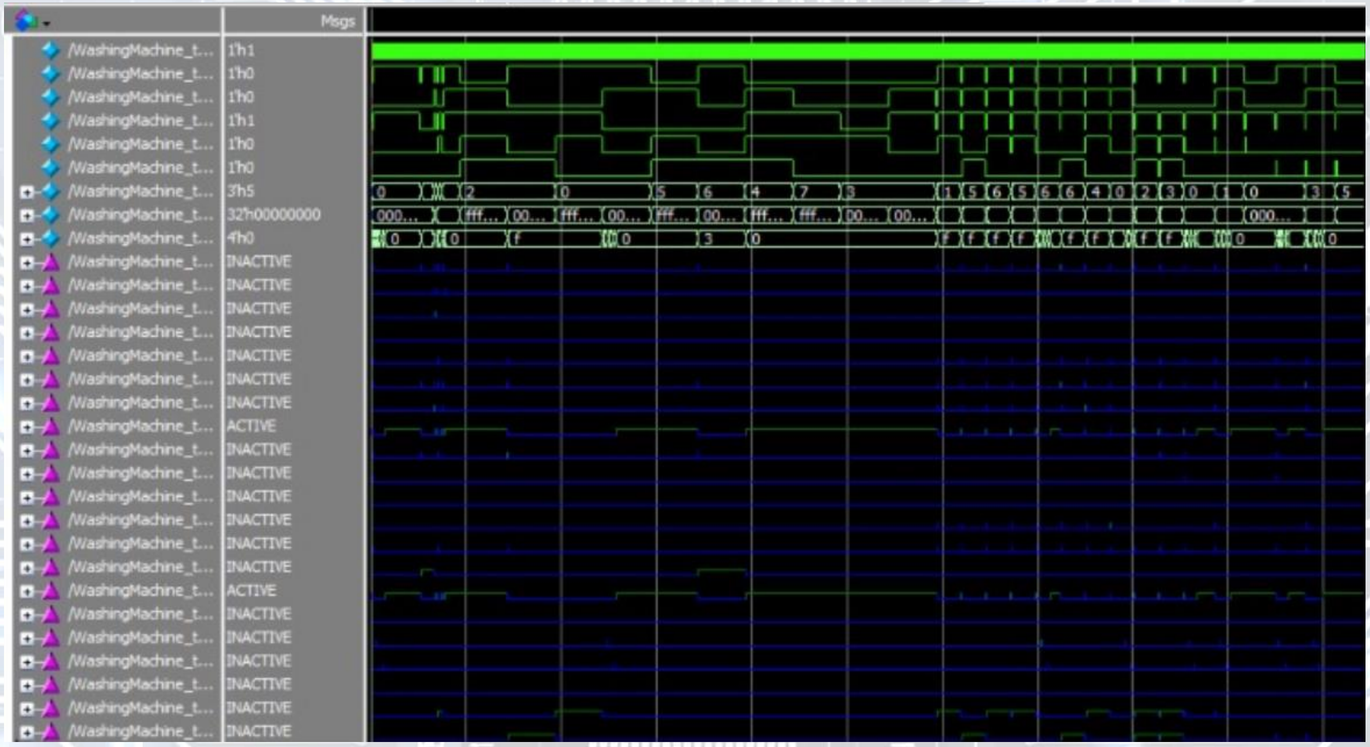
Out of 78 statements, 72 were executed during testing, yielding a 92.30% coverage. This high coverage ensures most of the design's functionality has been validated, with minor gaps to address.

Toggle Coverage:

Out of 182 toggles, 150 were hit, resulting in 82.41% coverage. This relatively high coverage highlights the successful testing of toggling of inputs and registers, which indicates numerous considerations of multiple test cases.

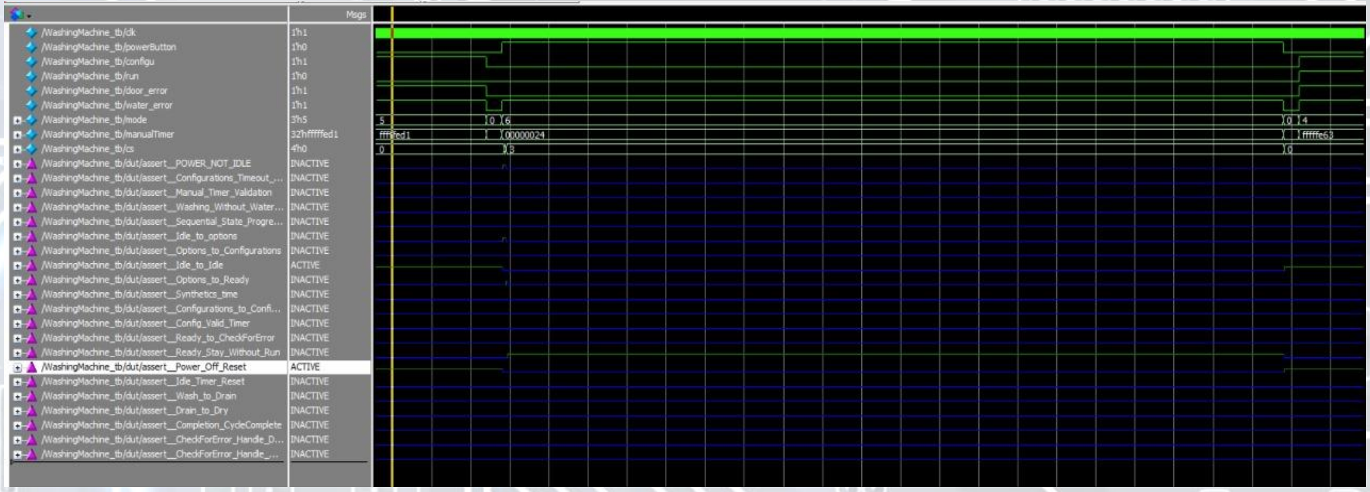
Assertions:

Wave form and outputs:

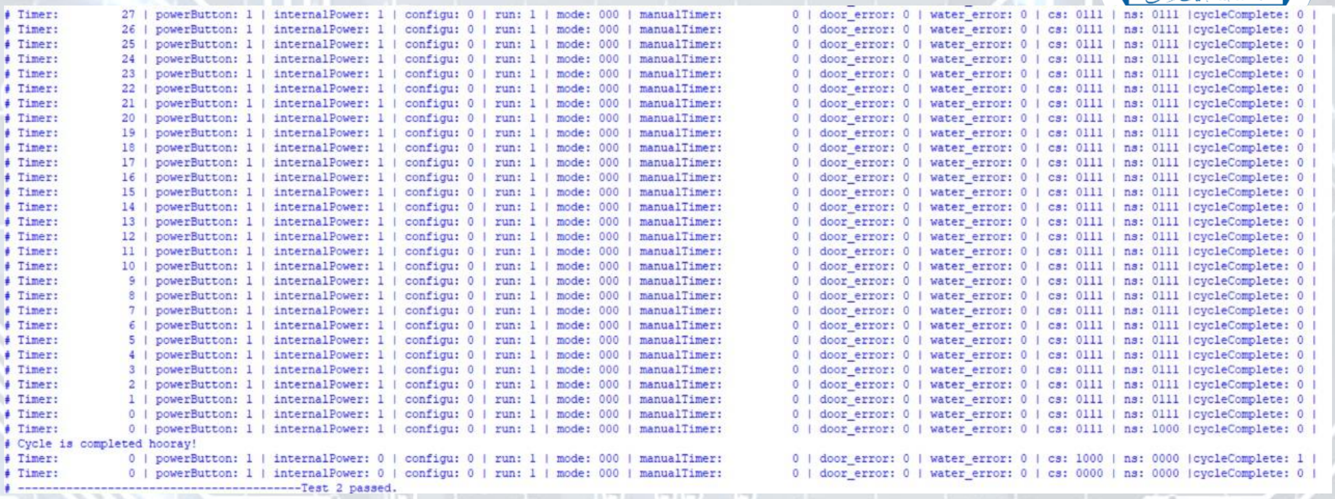


```
Msgs
/WashingMachine_t...   1'h1
/WashingMachine_t...   1'h0
/WashingMachine_t...   1'h0
/WashingMachine_t...   1'h1
/WashingMachine_t...   1'h0
/WashingMachine_t...   1'h0
/WashingMachine_t...   3'h5
/WashingMachine_t...   32'h00000000
/WashingMachine_t...   4'h0
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   ACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   ACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
/WashingMachine_t...   INACTIVE
```

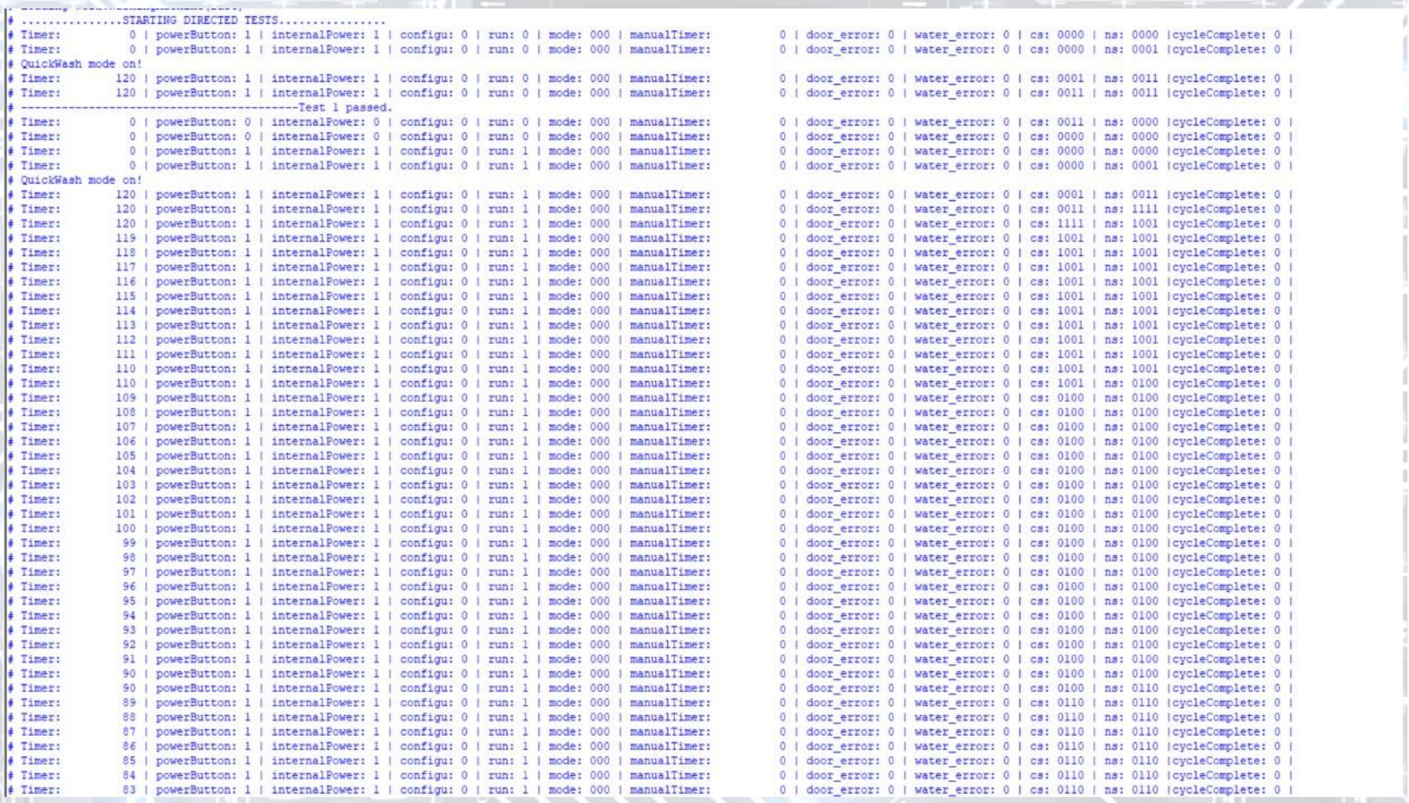
This is the wave form, with all the assertions.

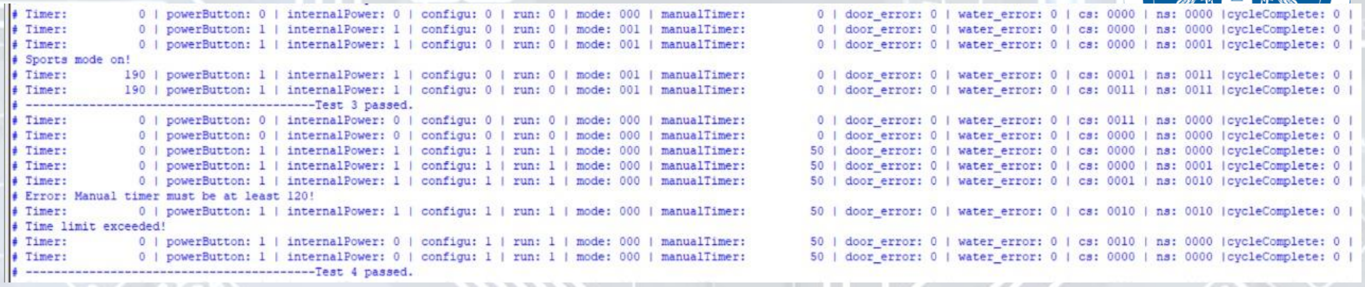


```
Msgs
/WashingMachine_tb/clk                                           1'h1
/WashingMachine_tb/powerButton                                   1'h0
/WashingMachine_tb/configu                                       1'h1
/WashingMachine_tb/run                                           1'h0
/WashingMachine_tb/door_error                                    1'h1
/WashingMachine_tb/water_error                                   1'h1
/WashingMachine_tb/mode                                          3'h5
/WashingMachine_tb/manualTimer                                   32'hffffffed1
/WashingMachine_tb/cs                                            4'h0
/WashingMachine_tb/dut/assert__POWER_NOT_IDLE                    INACTIVE
/WashingMachine_tb/dut/assert__Configurations_Timeout_...        INACTIVE
/WashingMachine_tb/dut/assert__Manual_Timer_Validation           INACTIVE
/WashingMachine_tb/dut/assert__Washing_Without_Water...          INACTIVE
/WashingMachine_tb/dut/assert__Sequential_State_Progre...        INACTIVE
/WashingMachine_tb/dut/assert__Idle_to_options                   INACTIVE
/WashingMachine_tb/dut/assert__Options_to_Configurations         INACTIVE
/WashingMachine_tb/dut/assert__Idle_to_Idle                      ACTIVE
/WashingMachine_tb/dut/assert__Options_to_Ready                  INACTIVE
/WashingMachine_tb/dut/assert__Synthetics_time                   INACTIVE
/WashingMachine_tb/dut/assert__Configurations_to_Confi...        INACTIVE
/WashingMachine_tb/dut/assert__Config_Valid_Timer                INACTIVE
/WashingMachine_tb/dut/assert__Ready_to_CheckForError            INACTIVE
/WashingMachine_tb/dut/assert__Ready_Stay_Without_Run            INACTIVE
/WashingMachine_tb/dut/assert__Power_Off_Reset                   ACTIVE
/WashingMachine_tb/dut/assert__Idle_Timer_Reset                  INACTIVE
/WashingMachine_tb/dut/assert__Wash_to_Drain                     INACTIVE
/WashingMachine_tb/dut/assert__Drain_to_Dry                      INACTIVE
/WashingMachine_tb/dut/assert__Completion_CycleComplete          INACTIVE
/WashingMachine_tb/dut/assert__CheckForError_Handle_D...         INACTIVE
/WashingMachine_tb/dut/assert__CheckForError_Handle_...          INACTIVE
```


```
# Timer:         27 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         26 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         25 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         24 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         23 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         22 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         21 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         20 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         19 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         18 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         17 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         16 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         15 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         14 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         13 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         12 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         11 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:         10 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:          9 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:          8 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:          7 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:          6 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:          5 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:          4 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:          3 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:          2 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:          1 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:          0 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 0111 |cycleComplete: 0 |
# Timer:          0 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0111 | ns: 1000 |cycleComplete: 0 |
# Cycle is completed hooray!
# Timer:          0 | powerButton: 1 | internalPower: 0 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1000 | ns: 0000 |cycleComplete: 1 |
# Timer:          0 | powerButton: 1 | internalPower: 0 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0000 |cycleComplete: 0 |
# ----------------------------------------Test 2 passed.
```

This is a sample of the output of the monitor for a test case in the directed mode.


```
# ...............STARTING DIRECTED TESTS................
# Timer:          0 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 0 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0000 |cycleComplete: 0 |
# Timer:          0 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 0 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0001 |cycleComplete: 0 |
# QuickWash mode on!
# Timer:        120 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 0 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0001 | ns: 0011 |cycleComplete: 0 |
# Timer:        120 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 0 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0011 | ns: 0011 |cycleComplete: 0 |
# ----------------------------------------Test 1 passed.
# Timer:          0 | powerButton: 0 | internalPower: 0 | configu: 0 | run: 0 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0011 | ns: 0000 |cycleComplete: 0 |
# Timer:          0 | powerButton: 0 | internalPower: 0 | configu: 0 | run: 0 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0000 |cycleComplete: 0 |
# Timer:          0 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0000 |cycleComplete: 0 |
# Timer:          0 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0001 |cycleComplete: 0 |
# QuickWash mode on!
# Timer:        120 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0001 | ns: 0011 |cycleComplete: 0 |
# Timer:        120 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0011 | ns: 1111 |cycleComplete: 0 |
# Timer:        120 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1111 | ns: 1001 |cycleComplete: 0 |
# Timer:        119 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1001 | ns: 1001 |cycleComplete: 0 |
# Timer:        118 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1001 | ns: 1001 |cycleComplete: 0 |
# Timer:        117 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1001 | ns: 1001 |cycleComplete: 0 |
# Timer:        116 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1001 | ns: 1001 |cycleComplete: 0 |
# Timer:        115 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1001 | ns: 1001 |cycleComplete: 0 |
# Timer:        114 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1001 | ns: 1001 |cycleComplete: 0 |
# Timer:        113 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1001 | ns: 1001 |cycleComplete: 0 |
# Timer:        112 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1001 | ns: 1001 |cycleComplete: 0 |
# Timer:        111 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1001 | ns: 1001 |cycleComplete: 0 |
# Timer:        110 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1001 | ns: 1001 |cycleComplete: 0 |
# Timer:        110 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 1001 | ns: 0100 |cycleComplete: 0 |
# Timer:        109 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:        108 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:        107 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:        106 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:        105 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:        104 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:        103 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:        102 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:        101 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:        100 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:         99 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:         98 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:         97 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:         96 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:         95 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:         94 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:         93 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:         92 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:         91 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:         90 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0100 |cycleComplete: 0 |
# Timer:         90 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0100 | ns: 0110 |cycleComplete: 0 |
# Timer:         89 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0110 | ns: 0110 |cycleComplete: 0 |
# Timer:         88 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0110 | ns: 0110 |cycleComplete: 0 |
# Timer:         87 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0110 | ns: 0110 |cycleComplete: 0 |
# Timer:         86 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0110 | ns: 0110 |cycleComplete: 0 |
# Timer:         85 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0110 | ns: 0110 |cycleComplete: 0 |
# Timer:         84 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0110 | ns: 0110 |cycleComplete: 0 |
# Timer:         83 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 1 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0110 | ns: 0110 |cycleComplete: 0 |
```



```
# Timer:           0 | powerButton: 0 | internalPower: 0 | configu: 0 | run: 0 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0000 |cycleComplete: 0 |
# Timer:           0 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 0 | mode: 001 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0000 |cycleComplete: 0 |
# Timer:           0 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 0 | mode: 001 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0001 |cycleComplete: 0 |
# Sports mode on!
# Timer:         190 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 0 | mode: 001 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0001 | ns: 0011 |cycleComplete: 0 |
# Timer:         190 | powerButton: 1 | internalPower: 1 | configu: 0 | run: 0 | mode: 001 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0011 | ns: 0011 |cycleComplete: 0 |
# ----------------------------------------Test 3 passed.
# Timer:           0 | powerButton: 0 | internalPower: 0 | configu: 0 | run: 0 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0011 | ns: 0000 |cycleComplete: 0 |
# Timer:           0 | powerButton: 0 | internalPower: 0 | configu: 0 | run: 0 | mode: 000 | manualTimer:          0 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0000 |cycleComplete: 0 |
# Timer:           0 | powerButton: 1 | internalPower: 1 | configu: 1 | run: 1 | mode: 000 | manualTimer:         50 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0000 |cycleComplete: 0 |
# Timer:           0 | powerButton: 1 | internalPower: 1 | configu: 1 | run: 1 | mode: 000 | manualTimer:         50 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0001 |cycleComplete: 0 |
# Timer:           0 | powerButton: 1 | internalPower: 1 | configu: 1 | run: 1 | mode: 000 | manualTimer:         50 | door_error: 0 | water_error: 0 | cs: 0001 | ns: 0010 |cycleComplete: 0 |
# Error: Manual timer must be at least 120!
# Timer:           0 | powerButton: 1 | internalPower: 1 | configu: 1 | run: 1 | mode: 000 | manualTimer:         50 | door_error: 0 | water_error: 0 | cs: 0010 | ns: 0010 |cycleComplete: 0 |
# Time limit exceeded!
# Timer:           0 | powerButton: 1 | internalPower: 0 | configu: 1 | run: 1 | mode: 000 | manualTimer:         50 | door_error: 0 | water_error: 0 | cs: 0010 | ns: 0000 |cycleComplete: 0 |
# Timer:           0 | powerButton: 1 | internalPower: 0 | configu: 1 | run: 1 | mode: 000 | manualTimer:         50 | door_error: 0 | water_error: 0 | cs: 0000 | ns: 0000 |cycleComplete: 0 |
# ----------------------------------------Test 4 passed.
```

Overview of PSL Assertions in the Washing Machine Controller: -

The provided PSL assertions are used to verify the behavior and correctness of a finite state machine (FSM) controlling a washing machine. These assertions ensure that the FSM adheres to its expected transitions, handles errors correctly, and validates specific conditions across different states. Here's a general explanation of the assertion groups, along with a few examples:

Error Handling Assertions:

Purpose: Ensure the FSM handles error conditions like water or door errors appropriately.

Example:

CheckForError_Handle_Door_Error: Ensures that the FSM remains in the error-checking state if a door error occurs.

Assertion: always (cs == CheckForError && door_error) -> next (cs == CheckForError);

CheckForError_Handle_Water_Error: Similar to the door error, this ensures that the FSM does not leave the error-checking state when a water error is present.

Timer and Configuration Validations:

Purpose: Validate that timers and configurations are correctly managed within specific states.

Example:

Manual_Timer_Validation: Ensures that the FSM stays in the configuration state if the provided manual timer is invalid (e.g., less than 120 seconds).

Assertion: always (cs == Configurations && manualTimer > 0 && manualTimer < 120) -> next (cs == Configurations);

Idle_Timer_Reset: Ensures that all timers are reset when the FSM is in the idle state.

Assertion: always (cs == Idle) -> (timer == 0 && config_timer == 0);

Conclusion: -

The provided FSM code for a washing machine effectively models its real-world operation by incorporating multiple washing modes, user-configurable timers, and robust error handling. It features well-defined states, including Idle, Options, Configurations, Ready, Wash, Rinse, Drain, Dry, Completion, and CheckForError, with logical transitions governed by user inputs and environmental feedback. The FSM supports power management via a manual toggle, ensuring a reset to Idle with variable initialization when powered off. Users can choose from pre-defined modes like QuickWash and GentleCare or set custom configurations, with validations ensuring proper timer values. The system handles errors, such as door and water issues, by pausing and retrying until conditions are resolved, ensuring resilience and safety. The washing cycle progresses sequentially through phases, culminating in the Completion state, where a cycle completion flag is set, and the machine powers down. While the FSM is robust and scalable, some areas could benefit from refinement, such as improved error handling timeouts, clearer variable naming, and additional timer validation. The inclusion of Property Specification Language (PSL) assertions strengthens its testability and correctness, making it a strong foundation for practical hardware implementation.