Washing Machine Controller

Group 3

Abdelrahman Mostafa Sallam

Mohamed Ashraf Mohamed

Marwan Ahmed Elsahy

Andrew Rami Bassily

Habeba Adel Sallam

Saif El Din Wael

—

22P0150
22P0210
22P0201
22P0187
22P0259
22P0191

—

Electronic Design Automation

- # Design Specifications:

Our design is used to simulate a washing machine's operation, covering phases such as washing, rinsing, spinning, and drying, with a focus on user-defined settings and safety features. This design functions as a state-based control system, progressing through a sequence of operational states, each representing a different phase of the washing process.

The washing machine's state is represented using a 6-bit full_state signal, combining a 2-bit errorpause register and a 4-bit state register. The errorpause register encodes high-level superstates: RUN (00), PAUSE (01), and ERROR (10), while the state register specifies the machine's current substate within these superstates, such as IDLE (0000), WASH (0110), or SPIN (1010). Together, these 6 bits allow the system to capture both the overall condition and the precise operational phase of the washing machine, enabling detailed state management and easy debugging. For example, a full_state of 6'b000110 represents the machine running (RUN) in the washing phase (WASH).

The system's design is organized around 14 distinct states. It starts in an IDLE state, waiting for user input. Once the user initiates the process by pressing start, the system moves to CHOOSE_SETTINGS, where it either applies preset settings or, if in custom mode, transitions to ADJUST_SETTINGS to apply user-defined configurations.

Next, CHECK_DOOR ensures the door is closed before operation. After that, the machine enters the operational states, starting with FILL_WATER which manages the process of filling the washing drum with water. During this state, the machine monitors the water_level_reading input to check the current water level in the drum and compares it to the desired water_level, which is determined by either the preset program or custom user settings. The machine continues to fill the drum until the current water level reaches the specified target. Once the water level is sufficient, the machine transitions to the next state (WASH). If there is an issue with water flow or if the waterflow signal is not active, the machine enters an ERROR state to ensure safe operation. This state is crucial for preparing the washing process, ensuring that the clothes are properly submerged before proceeding to the washing or rinsing phases.

Moving on to WASH state, the machine ensures that the water level remains at or above the desired level, checking the water_level_reading to verify this. If the water level drops below the target, the machine may revert to the FILL_WATER state to add more water. If the water level is correct, the

machine performs the washing process once and then the number of predefined cycles is deducted by 1. After the washing action is completed, the machine moves to the CHECK_CYCLES state. In this state, the machine evaluates the number of cycles remaining, based on the cycles value, which was either set by the preset program or defined by the user with custom settings. If there are more wash cycles to complete, the machine transitions back to the WASH state to repeat the washing process. If there are no cycles left, the machine moves on to the next state RINSE.

In the RINSE state, the washing machine checks if the water level is sufficient by comparing the water_level_reading to the desired water_level. If the water level is too low, the machine stays in the FILL_WATER state until the water level is reached. Once the water level is adequate, the rinse process begins. After each rinse cycle, the machine drains the water and then checks the rinse condition again. The number of remaining rinse cycles is tracked using the rinse_cycles counter. If there are more rinse cycles left, the system returns to the RINSE state to perform the next rinse cycle. Once all rinse cycles are completed, the rinse_complete signal is set to 1, and the machine transitions to the DRAIN state to remove any remaining water before proceeding to the next phase of the washing process.

Followed by the SPIN state in which clothes are spun at a specified speed to remove excess water. During this state, the machine starts the spin cycle, and the spin speed is controlled by the speed register, which is either set by the preset program or custom user input. The spinning process is executed by increasing the spin timer (spin_timer), which is incremented based on the selected spin speed. The machine continues spinning until the spin_timer reaches a threshold. This threshold is typically defined by the speed setting, which influences how quickly the timer increments and how long the spin cycle lasts. The machine will remain in the SPIN state as long as the spin_timer is below the defined limit (in this case, it checks if the spin_timer is less than or equal to 64, which represents the total duration of the spin phase). Once the spin cycle is complete (i.e., when the spin_timer exceeds the set limit), the machine checks if there is a dry_timer configured. If a drying phase is required, it transitions to the DRY state. If no drying is required, it moves to the WASH_COMPLETE state, marking the end of the washing process.

Transiting to DRY state, in this state, the machine uses a dry_timer, which was either set by the user or specified by a preset program, to control the duration of the drying process. The drying phase begins when the machine transitions into the DRY state, and the dry_counter is used to keep track of the elapsed time. During the DRY state, the dry_counter increments as long as it is less than the

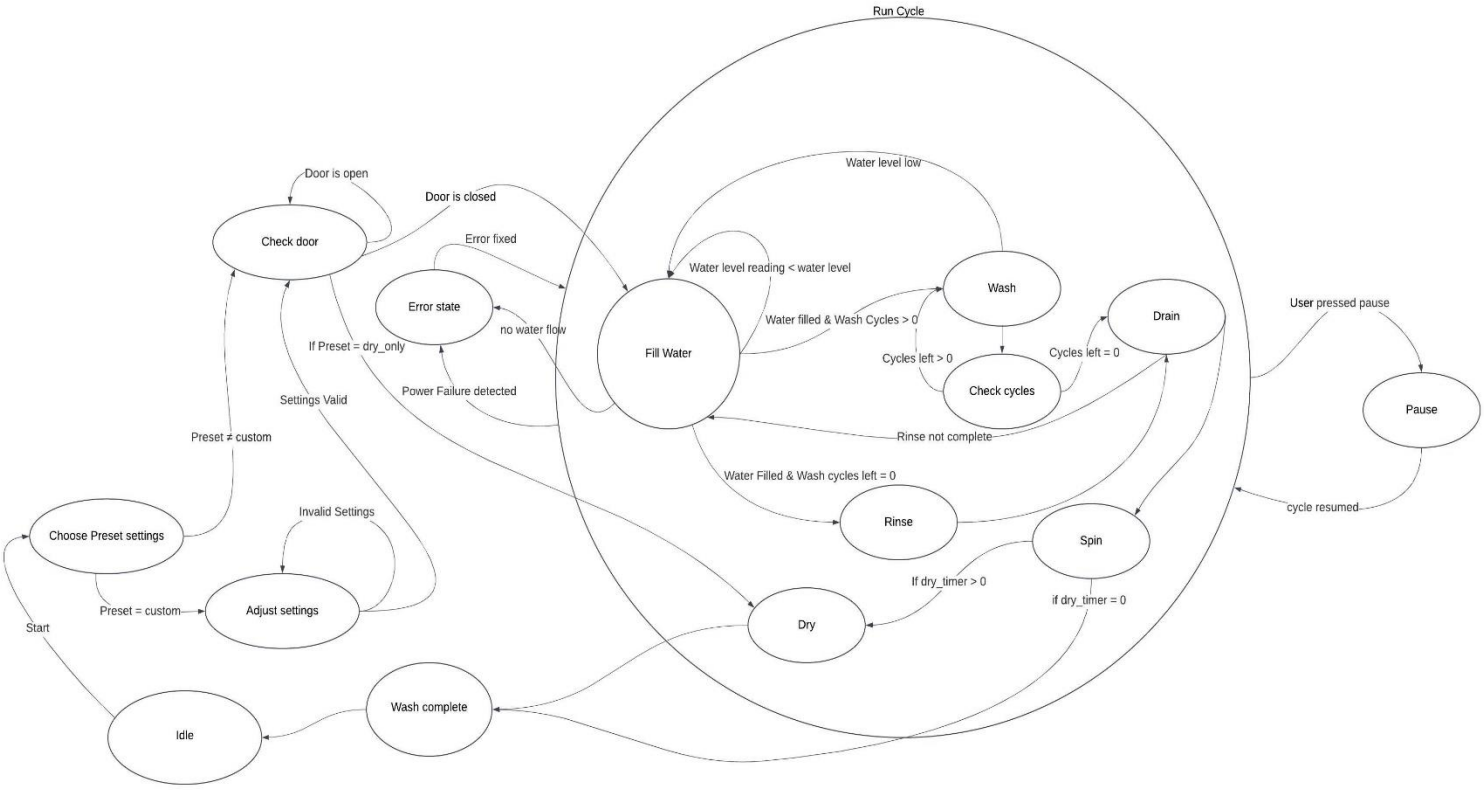
dry_timer value. The machine continues to dry the clothes, typically by applying heat or using a mechanical drying method, depending on the washing machine's design. Once the dry_counter reaches the specified dry_timer value, the machine completes the drying phase. After the drying is complete, the machine transitions to the WASH_COMPLETE state, indicating that the entire wash and dry process has been finished. At the end of the wash cycle, an alarm is heard.

If the reset signal is triggered (via the positive edge of the reset input), the machine's internal registers are reset to their default values. This includes resetting the washing parameters such as speed, temp, cycles, dry_timer, water_level, and other states to zero or predefined values.

If the user press pause, the machine enters the pause state, it temporarily halts all ongoing operations, preserving the current substate (state) so the process can resume seamlessly when unpaused. The pause state is triggered by the pause input signal and can only transition back to the RUN state if the start input is activated and there are no safety issues, such as the door being open. While paused, the errorpause[0] bit is set to 1, and the machine does not proceed to the next substate (e.g., WASH, SPIN). If conditions are met to resume, the machine clears the pause flag (errorpause[0]) and reverts to the corresponding substate within the RUN superstate, resuming operations exactly where they were interrupted.

Moving to the error state indicates that an operational issue has occurred, preventing the machine from continuing its cycle safely. The error state is triggered by specific conditions, such as a water flow failure during the FILL_WATER substate or a power cut, as detected by the input signals. When the machine enters the error state, errorpause[1] is set to 1, and all ongoing operations are halted to protect the machine and its surroundings. In this state, the machine does not transition to other substates until the error condition is resolved (e.g., water flow is restored or power is stabilized). Additionally, the machine may activate the alarm output to alert the user of the issue. Once the problem is addressed, and appropriate recovery actions (like resetting the machine) are taken, the errorpause flag is cleared, and the machine can return to its normal operation (RUN) or be restarted from a safe initial state.

- Finite State Machine Diagram:



- Table of inputs, outputs and parameters with description

➢ Inputs:

Signal Name	Type	Description
clk	1-bit	Clock signal that drives the state machine.
reset	1-bit	Active high reset signal that resets the washing machine's internal registers and state to their default values.
start	1-bit	Input signal to start the washing machine process.
pause	1-bit	Input signal to pause the washing machine process.
preset	3-bit	Preset setting input that defines the type of wash cycle selected (e.g., wool, cotton, quick, etc.).
custom_temp	3-bit	Custom temperature setting input (range 0–5).
custom_speed	3-bit	Custom spin speed setting input (range 1–5).
custom_cycles	4-bit	Custom number of wash cycles input (range 0–10).
custom_dry_timer	6-bit	Custom drying timer setting (range 0–60 minutes).
custom_water_level	3-bit	Custom water level setting input (range 0–5).
door_open_signal	1-bit	Signal indicating whether the washing machine door is open or closed. If the door is open, the machine pauses.
waterflow	1-bit	Signal indicating whether water is flowing into the washing machine. Used to detect errors in water supply.
water_level_reading	3-bit	Sensor input providing the current water level in the drum.
power_cut	1-bit	Signal indicating if a power cut or disruption occurred. Used to trigger an error state.

➢ Outputs:

Signal Name	Type	Description
speed	3-bit	Output signal that controls the washing machine's spin speed.
temp	3-bit	Output signal controlling the water temperature for washing.
cycles	4-bit	Output signal defining the number of washing cycles remaining.
dry_timer	6-bit	Output signal that sets the drying time (in minutes).
water_level	3-bit	Output signal that defines the water level inside the washing machine's drum.
rinse_complete	1-bit	Signal indicating whether the rinse cycle has been completed.
full_state	6-bit	A 6-bit register representing the current state of the washing machine, including both the superstate (e.g., RUN, PAUSE, ERROR) and the substate.
state	4-bit	A 4-bit register representing only the substates of RUN.
errorpause	2-bit	A 2-bit register representing only the higher-level states PAUSE and ERROR, and the superstate RUN.
alarm	1-bit	Alarm output signal that indicates errors, such as power cuts or issues during the washing process.
finishedAlarm	1-bit	Alarm output signal that indicates when the washing machine cycle has finished.

➢ Parameters:

Signal Name	Type	Description
RUN	2-bit	Superstate indicating that the washing machine is running (performing washing, rinsing, etc.).
ERROR	2-bit	Superstate indicating that an error has occurred, such as a power cut or waterflow issue.
PAUSE	2-bit	Superstate indicating that the washing machine is paused, waiting for user input or resolution of an error.
IDLE	4-bit	Substate indicating that the machine is idle, waiting for the user to start the cycle.
CHOOSE_SETTINGS	4-bit	Substate where the user selects the cycle settings (preset or custom).
ADJUST_SETTINGS	4-bit	Substate where the user adjusts custom settings like speed, temperature, cycles, and water level.
CHECK_DOOR	4-bit	Substate where the machine checks if the door is open or closed.
FILL_WATER	4-bit	Substate where the machine fills the drum with water to the desired level.
WASH	4-bit	Substate where the machine agitates the clothes in water for washing.
CHECK_CYCLES	4-bit	Substate where the machine checks the remaining cycles in the wash program.
DRAIN	4-bit	Substate where the machine drains the water after washing or rinsing.
RINSE	4-bit	Substate where the machine rinses the clothes to remove detergent.
SPIN	4-bit	Substate where the machine spins the clothes to remove excess water.
DRY	4-bit	Substate where the machine dries the clothes, based on the dry timer setting.
WASH_COMPLETE	4-bit	Substate where the washing cycle is complete and the machine transitions to idle.

➢ Internal Registers:

Signal Name	Type	Description
rinse_cycles	5-bit	Keeps track of the number of rinse cycles remaining during the rinse process. Initially set to 10.
spin_timer	6-bit	Timer for controlling the duration of the spin cycle, incremented based on speed.
dry_counter	6-bit	Counter that tracks the elapsed time for the drying cycle, used to determine when the drying process is complete.

- Design RTL Code with PSL assertions

```
module WashingMachine(
    input clk,
    input reset,
    input start,
    input pause,
    input [2:0] preset,
    input [2:0] custom_temp,
    input [2:0] custom_speed,
    input [3:0] custom_cycles,
    input [5:0] custom_dry_timer,
    input [2:0] custom_water_level,
    input door_open_signal,
    input waterflow,
    input [2:0] water_level_reading,
    input power_cut,
    output reg [2:0] speed,
    output reg [2:0] temp,
    output reg [3:0] cycles,
    output reg [5:0] dry_timer,
    output reg [2:0] water_level,
    output reg rinse_complete,
    output wire [5:0] full_state,
    output reg [3:0] state,
    output reg [1:0] errorpause,
    output reg alarm, //toot toot
    output reg finishedAlarm
);

// Define state parameters
    //superstates:
parameter RUN = 2'b00;
parameter ERROR = 2'b10;
parameter PAUSE = 2'b01;

parameter IDLE = 4'b0000;
parameter CHOOSE_SETTINGS = 4'b0001;
parameter ADJUST_SETTINGS = 4'b0010;
parameter CHECK_DOOR = 4'b0011;

    //substates of normal run
parameter FILL_WATER = 4'b0101;
parameter WASH = 4'b0110;
parameter CHECK_CYCLES = 4'b0111;
parameter DRAIN = 4'b1000;
parameter RINSE = 4'b1001;
parameter SPIN = 4'b1010;
parameter DRY = 4'b1011;
parameter WASH_COMPLETE = 4'b1100;

// Internal registers
reg [4:0] rinse_cycles;
reg [6:0] spin_timer;
reg [5:0] dry_counter;
//reg[1:0] errorpause;
//reg [3:0] state;
assign full_state = {errorpause, state};

```

```
58
59  initial begin
60      rinse_cycles <= 10;
61      speed <= 0;
62      temp <= 0;
63      cycles <= 0;
64      dry_timer <= 0;
65      dry_counter <=0;
66      water_level <= 0;
67      rinse_complete <= 0;
68      state <= 4'b0;
69      spin_timer <= 0;
70      errorpause <= 0;
71  end
72
73
74  //pause state always
75  always @(posedge clk) begin
76      if(pause)
77          errorpause[0] <= 1;
78      else if(start)
79          if(!door_open_signal)
80          begin
81              errorpause[0] <= 0;
82              alarm <= 0;
83          end
84          else
85              alarm <= 1;
86      else alarm <= 0;
87  end
88
89  //error state
90  //EDIT
91   always @(posedge clk) begin
92          if((state == FILL_WATER && !waterflow) || power_cut)
93                  errorpause[1] <= 1;
94          else errorpause[1] <= 0;
95  end
96
97
98
99  //Next State Logic
100 always @(posedge clk or posedge reset) begin
101
102     if(state == WASH_COMPLETE)
103         finishedAlarm <= 1;
104     else finishedAlarm <= 0;
105
106     if (reset) begin
107     speed <= 0;
108     temp <= 0;
109     cycles <= 0;
110     dry_timer <= 0;
111     water_level <= 0;
112     rinse_complete <= 0;
113     state <= 4'b0;
114     spin_timer <= 0;
```

```
115     dry_counter <= 0;
116     rinse_cycles <= 10;
117     errorpause <= 0;
118 
119     end else begin
120 
121         case (state) //synopsys full_case
122             IDLE:
123             begin
124                 if (start)
125                     state <= CHOOSE_SETTINGS;
126             end
127 
128             CHOOSE_SETTINGS:
129             begin
130                 if (preset == 7)            //custom settings
131                     state <= ADJUST_SETTINGS;
132                 else begin
133                     if (preset==0)
134                         begin               //wool
135                             speed <= 2;
136                             temp <= 2;
137                             cycles <= 4;
138                             dry_timer <= 0;
139                             water_level <= 5;
140                         end
141                     else if (preset==1)
142                         begin               //cotton
143                             speed <= 3;
144                             temp <= 5;
145                             cycles <= 6;
146                             dry_timer <= 0;
147                             water_level <= 7;
148                         end
149                     else if (preset==2)
150                         begin               //quick
151                             speed <= 4;
152                             temp <= 3;
153                             cycles <= 2;
154                             dry_timer <= 0;
155                             water_level <= 4;
156 
157                         end
158                     else if (preset==3)
159                         begin               //wash and dry
160                             speed <= 3;
161                             temp <= 3;
162                             cycles <= 8;
163                             dry_timer <= 30;
164                             water_level <= 6;
165 
166                         end
```

```
                else if (preset==4)
                    begin               //dry only
                        speed <= 0;
                        temp <= 0;
                        cycles <= 0;
                        dry_timer <= 60;
                        water_level <= 0;

                    end
                else if (preset==5)
                    begin               //sports
                        speed <= 3;
                        temp <= 2;
                        cycles <= 7;
                        dry_timer <= 0;
                        water_level <= 7;

                    end
                else if (preset==6)
                    begin                //silk
                        speed <= 1;
                        temp <= 20;
                        cycles <= 3;
                        dry_timer <= 0;
                        water_level <= 6;
                    end
                state <= CHECK_DOOR;
            end
        end

        ADJUST_SETTINGS: begin
            if (custom_speed >= 1 && custom_speed <= 5 &&
                custom_temp >= 0 && custom_temp <= 5 &&
                custom_cycles >= 0 && custom_cycles <= 10 &&
                custom_dry_timer >= 0 && custom_dry_timer <= 60 &&
                custom_water_level>=1 && custom_water_level<= 7)
                begin
                    speed <= custom_speed;
                    temp <= custom_temp;
                    cycles <= custom_cycles;
                    dry_timer <= custom_dry_timer;
                    water_level<=custom_water_level;
                    state <= CHECK_DOOR;
                end
            else
                state <= ADJUST_SETTINGS;
        end

        CHECK_DOOR: begin
            if (door_open_signal) begin
                state <= CHECK_DOOR;
            end
            else
                if (cycles == 0)
                    state <= DRY;
                else
                    state <= FILL_WATER;
        end

        default: begin
            if(!errorpause)
                case (state) //synopsys full_case
                    FILL_WATER: begin
                        if (water_level_reading < water_level) begin
                            state <= FILL_WATER;
                        end
                        else if(cycles==0) begin
                            state <= RINSE;
                        end
                        else begin
                            state <= WASH;
                        end
                    end
```

```
241                     WASH: begin
242                         if (water_level_reading < water_level)
243                         begin
244                               state <= FILL_WATER;
245                         end
246                         else
247                         begin
248                                cycles <= cycles - 1;
249                                state <= CHECK_CYCLES;
250                         end
251                     end
252
253                     CHECK_CYCLES: begin
254                       if (cycles == 0) begin
255                          state <= DRAIN;
256                       end else begin
257                          state<= WASH;
258                       end
259                     end
260
261                     RINSE: begin
262                         if (water_level_reading < water_level)
263                             begin
264                                 state <= FILL_WATER;
265                             end
266                         else
267                             begin
268                             if(rinse_cycles > 0)
269                                 rinse_cycles <= rinse_cycles - 1;
270                             else
271                                 rinse_complete <= 1;
272                             state <= DRAIN;
273                             end
274                     end
275
276                     DRAIN: begin
277                        if (!rinse_complete)
278                             begin
279                                 state <= FILL_WATER;
280                             end
281                         else
282                             begin
283                                 state <= SPIN;
284                             end
285                     end
286
287                     SPIN: begin
288                         spin_timer <= spin_timer + speed;
289                         if (spin_timer<=64) begin
290                             state <= SPIN;
291                         end else if (dry_timer) begin
292                             state <= DRY;
293                         end else begin
294                             state <= WASH_COMPLETE;
295                         end
296                     end
297
298
299                     DRY:
300                     begin
301                         if (dry_counter<dry_timer)
302                             begin
303                                 dry_counter <= dry_counter + 1;
304                                 state <= DRY;
305                             end
306                         else
307                             state <= WASH_COMPLETE;
308                     end
309
310                     WASH_COMPLETE: begin
311                             state <= IDLE;
312                     end
```

```
                        default: begin
                            state <= state;
                        end
                    endcase
                end
            endcase
        end
    end
    // Ensure reset moves FSM to IDLE
    /*
    psl default clock = rose(clk);
    psl property Reset_To_IDLE = always (reset -> next(state == IDLE));
    psl assert Reset_To_IDLE;
    */

    // Ensure that in IDLE, start initiates transition to CHOOSE_SETTINGS
    /*
    psl property Idle_To_ChooseSettings = always ((state == IDLE && start) -> next(state == CHOOSE_SETTINGS));
    psl assert Idle_To_ChooseSettings;
    */

    // Ensure preset 7 (custom settings) transitions to ADJUST_SETTINGS
    /*
    psl property Custom_To_AdjustSettings = always ((state == CHOOSE_SETTINGS && preset == 7) -> next(state == ADJUST_SETTINGS));
    psl assert Custom_To_AdjustSettings;
    */

    // Ensure other presets transition to CHECK_DOOR
    /*
    psl property Preset_To_CheckDoor = always ((state == CHOOSE_SETTINGS && preset != 7) -> next(state == CHECK_DOOR));
    psl assert Preset_To_CheckDoor;
    */

    // Ensure pause sets FSM to PAUSE state
    /*
    psl property Pause_State = always (pause -> next(errorpause[0] == 1));
    psl assert Pause_State;
    */

    // Ensure start with the door closed clears PAUSE state and alarm
    /*
    psl property Start_Without_Alarm = always ((errorpause == PAUSE && start && !door_open_signal) -> next(errorpause[0] == 0 && alarm == 0));
    psl assert Start_Without_Alarm;
    */

    // Ensure start with the door open triggers alarm
    /*
    psl property Start_With_Alarm = always ((errorpause == PAUSE && start && door_open_signal) -> next(alarm == 1));
    psl assert Start_With_Alarm;
    */

    // Ensure ERROR is triggered on power_cut
```

```
365  /*
366  psl property Error_On_PowerCut = always (power_cut -> next(errorpause[1] == 1));
367  psl assert Error_On_PowerCut;
368  */
369
370  // Ensure ERROR is triggered on missing waterflow in FILL_WATER
371  /*
372  psl property Error_On_NoWaterflow = always ((state == FILL_WATER && !waterflow) -> next(errorpause[1] == 1));
373  psl assert Error_On_NoWaterflow;
374  */
375
376  // Ensure correct transition from WASH to FILL_WATER based on water_level_reading
377  /*
378  psl property Wash_To_FillWater = always ((errorpause == 0 && state == WASH && water_level_reading < water_level) -> next(state == FILL_WATE
379  psl assert Wash_To_FillWater;
380  */
381
382  // Ensure CHECK_CYCLES transitions to appropriate states
383  /*
384  psl property CheckCycles_Transition = always ((state == CHECK_CYCLES && errorpause == 0) -> next(state == WASH || state == DRAIN));
385  psl assert CheckCycles_Transition;
386  */
387
388  // Ensure alarm is activated in WASH_COMPLETE
389  /*
390  psl property Alarm_On_WashComplete = always ((state == WASH_COMPLETE) -> next(finishedAlarm == 1));
391  psl assert Alarm_On_WashComplete;
392  */
393  endmodule
```

- # Verification Plan

1. Objective

The goal of this verification plan is to ensure the functional correctness, error handling, and edge-case robustness of the washing machine controller designed for this project. The verification will cover all primary features, normal operations, edge cases, and error recovery mechanisms.

2. Verification Methodology

- ***Directed Test Scenarios***:
  Test cases are explicitly defined based on the state machine design and expected behaviors. These include:
  - Single Feature scenarios: Verifying individual state transitions and functionality.
  - Normal Operation scenarios: Validating end-to-end operations under standard conditions.
  - Edge Case scenarios: Testing the system's behavior with extreme or boundary inputs.
  - Error Recovery scenarios: Simulating interruptions in power and waterflow to assess system resilience.

- ***Test Bench Simulation****:*
  The state machine will be implemented and tested in a Verilog simulation environment using a comprehensive test bench.
  - Input stimuli for each scenario will be applied sequentially.
  - Outputs will be monitored and compared against expected results for correctness.

- ***Assertion-Based Verification****:*
  Assertions will be embedded in the testbench to ensure key conditions and state transitions are met.

- ***Coverage Metrics**:*
    - Functional coverage will ensure all states and transitions in the FSM are exercised.
    - Code coverage will confirm that all lines and branches of Verilog code are tested.

3. Tools and Environment

- ***Tools Used**:*
    - Simulation software used was QuestaSim for running test scenarios.
    - Waveform views employed for analyzing state transitions and signal behaviors.
    - Visual Studio Code was used to write the Verilog design code, as well as the testbench code for the directed stimulus test scenarios.

- ***Test Environment**:*
    - Each scenario will be simulated under controlled conditions, with logs and waveforms recorded for analysis.

4. Test Scenarios

The verification plan includes directed test scenarios categorized as follows:

- ***Single Feature Scenarios**:*

  Test specific state transitions and input conditions, such as:
    - WASH → FILL_WATER → WASH transitions based on water level readings.
    - Custom preset validation for upper and lower input bounds.

- ***Normal Operation Scenarios**:*

  Simulate complete washing cycles for different presets to validate end-to-end functionality.

- ***Edge Case Scenarios:***

  Test boundary inputs and conditions, including extreme values for custom presets.

- ***Error Recovery Scenarios:***

  Validate the controller's response to waterflow and power interruptions.

5. Success Criteria

Verification will be considered successful if:

1. All directed test scenarios produce the expected state transitions and outputs.
2. Functional and code coverage metrics meet project standards.
3. No unhandled errors or unexpected behavior occur during testing.

➢ Directed Stimulus Test Scenarios:

A. <u>Single Feature scenario testing:</u>

1. WASH to FILL_WATER to WASH

Description:

Test transition from WASH to FILL_WATER and then back to WASH.

Initial Setup:

- Reset system.
- Select custom preset with the following:
  - Water Level: 3
  - Cycles: 10
- Ensure door_open_signal = 0 (door closed).
- Press Start to transition to the WASH state.
- Set water_level_reading = 4.

Inputs:

1. Change water_level_reading to 2.
2. Change water_level_reading back to 3.

Expected State Transitions:

WASH → FILL_WATER → WASH

2. DRAIN to FILL_WATER to RINSE to DRAIN

Description:

Test sequence of transitions involving DRAIN, FILL_WATER, RINSE, and back to DRAIN.

Initial Setup:

- Reset system.
- Set water_level = 3 and ensure cycles = 0 to transition to the DRAIN state.

Inputs:

1. DRAIN completes, transitioning to FILL_WATER.
2. Set water_level_reading sequentially:
    - First to 1,
    - Then to 2,
    - And finally, to 3 to reach RINSE.
3. After one rinse cycle, state transitions back to DRAIN.

Expected State Transitions:

DRAIN → FILL_WATER → RINSE → DRAIN

3. Closed/Open Door Start

Description:

Test behavior when attempting to start with the door open and then with the door closed.

Initial Setup:

- Reset system.

- Choose any preset (except custom).

Inputs:

1. Set door_open_signal = 1 (door open) and press start.
2. Set door_open_signal = 0 (door closed) and press start.

Expected Output:

- No state change when door_open_signal = 1.
- State transitions to RUN superstate when door_open_signal = 0.

4. Invalid Custom Input Behavior

Description:

Test the system's response to invalid custom input values and verify behavior for valid input values.

Initial Setup*:*

- Reset system.
- Choose Custom Preset (7).
- Provide custom settings as follows:
    - custom_cycles = 11 (invalid, exceeds max of 10).
    - custom_speed = 6 (invalid, exceeds max of 5).
    - The rest of the settings (custom_temp, custom_dry_timer, custom_water_level) are set to 0.

Inputs*:*

1. Press start with the above settings.
2. Update inputs:
    - custom_cycles = 10 (valid).
    - custom_speed = 5 (valid).
3. Press start again.

***Expected Behavior**:*

- First Input: No state change (remains in ADJUST_SETTINGS) due to invalid values.
- Second Input: State transitions from ADJUST_SETTINGS to RUN superstate with valid custom settings applied.

B. Normal Operation scenarios:

1. Wool Preset Normal Operation

Description

Test the washing machine's behavior with the Wool preset (Preset 0) under normal operating conditions.

***Initial Setup**:*

- Reset system.
- Choose Wool preset (Preset 0).
- Ensure door_open_signal = 0 (door is closed).
- User presses start.

***Inputs**:*

1. waterflow = 1 (water flows without interruption).
2. power_cut = 0 (power is never interrupted).
3. During the WASH state, water_level_reading fluctuates to simulate water usage.
4. During DRAIN and FILL_WATER states, water_level_reading updates normally to reflect transitions.

***Expected Behavior**:*

- State transitions follow the expected RUN superstate sequence: FILL_WATER → WASH → DRAIN → RINSE → SPIN → WASH_COMPLETE.
- Alarm remains inactive throughout the process until the WASH_COMPLETE state.

2. Quick Wash with Interruptions

Description

Test the Quick Wash preset (Preset 2) with interruptions such as Pause and an open door condition.

Initial Setup:

- Reset system.
- Choose Quick Wash preset (Preset 2).
- Ensure door_open_signal = 0 (door is closed).
- User presses start.

Inputs:

1. waterflow = 1 and power_cut = 0 (no interruptions during normal operation).
2. During the DRAIN state, pause = 1. After a while, pause = 0, and start = 1 with door_open_signal = 1 (door open).
3. Set door_open_signal = 0 (door closed), and start = 1.

Expected Behavior:

- Initial state transitions follow the Quick Wash sequence: IDLE → CHOOSE_SETTINGS → FILL_WATER → WASH → DRAIN.
- On pause during DRAIN, system transitions to PAUSE superstate.
- System remains paused when door_open_signal = 1, and alarm output = 1.
- Upon door_open_signal = 0 and start = 1, state resumes from DRAIN and continues the sequence: DRAIN → RINSE → SPIN → WASH_COMPLETE.

3. Dry Only Preset

Description

Test the Dry Only preset (Preset 4) under conditions where water is not needed.

***Initial Setup**:*

- Reset system.
- Choose Dry Only preset (Preset 4).
- Ensure door_open_signal = 0 (door is closed).
- User presses start.

***Inputs**:*

1. waterflow = 0 (no water needed for this preset).
2. power_cut = 0 (power is never interrupted).

***Expected Behavior**:*

- System transitions directly to DRY state and skips water-dependent states like FILL_WATER and WASH.
- System remains in DRY until the dry_timer completes.
- State transitions: IDLE → CHOOSE_SETTINGS → DRY → WASH_COMPLETE.

C. Edge-case Operation scenarios:

1. Upper Edge Case for Custom Inputs

Description

Test the upper boundary limits for custom input values and verify system behavior.

***Initial Setup**:*

- Reset system.
- Choose Custom preset (Preset 7).
- Set custom parameters to the upper allowed limits:
  - custom_cycles = 10 (maximum allowed cycles).
  - custom_speed = 5 (maximum allowed speed).
  - custom_temp = 5 (maximum allowed temperature).

    - custom_water_level = 7 (maximum water level input).
- Ensure door_open_signal = 0 (door is closed).
- User presses start.

***Inputs**:*

1. waterflow = 1 (normal water flow).
2. power_cut = 0 (no power interruptions).

***Expected Behavior**:*

- State transitions to RUN superstate.
- Washing sequence is executed according to custom settings:
    - WASH → DRAIN → RINSE → SPIN → WASH_COMPLETE.
- System completes the wash cycle without errors.
- Alarm remains inactive until the WASH_COMPLETE state.

2. Lower Edge Case for Custom Inputs

Description

Test the lower boundary limits for custom input values and verify system behavior.

***Initial Setup**:*

- Reset system.
- Choose Custom preset (Preset 7).
- Set custom parameters to the lower allowed limits:
    - custom_cycles = 1 (minimum allowed cycles).
    - custom_speed = 1 (minimum allowed speed).
    - custom_temp = 0 (minimum allowed temperature).
    - custom_water_level = 1 (minimum water level input).
- Ensure door_open_signal = 0 (door is closed).
- User presses start.

***Inputs**:*

1. waterflow = 1 (normal water flow).
2. power_cut = 0 (no power interruptions).

***Expected Behavior**:*

- State transitions to RUN superstate.
- Washing sequence is executed according to custom settings:
    - WASH → DRAIN → RINSE → SPIN → WASH_COMPLETE.
- System completes the wash cycle without errors.
- Alarm remains inactive until the WASH_COMPLETE state.

D. Error recovery scenarios:

1. Recovery from Waterflow and Power Interruptions

Description

Test system recovery when waterflow and power are interrupted at different stages of the cycle.

***Initial Setup**:*

- Reset system.
- Choose Wash and Dry (Preset 3).
- Ensure door_open_signal = 0 (door is closed).
- User presses start.
- Normal inputs:
    - waterflow = 1 (initially normal).
    - power_cut = 0 (initially no power cut).

***Error Events**:*

1. During WASH or FILL_WATER, waterflow = 0 (waterflow is cut).
2. waterflow = 1 (waterflow returns after some delay).
3. During RINSE, power_cut = 1 (power is cut).

4. power_cut = 0 (power returns after some delay).
5. Normal water level reading fluctuations during all states.

***Expected Behavior**:*

- On waterflow cut during WASH or FILL_WATER:
  - System transitions to an ERROR state and waits for waterflow to return.
  - Once waterflow = 1, system resumes from the paused state and continues the cycle.
- On power cut during RINSE:
  - System saves the current state to non-volatile memory.
  - On power return (power_cut = 0), system resumes from the last saved state.
- Final sequence after interruptions:
  - WASH → FILL_WATER → RINSE → SPIN → DRY → WASH_COMPLETE.

2. Recovery from Continuous Waterflow Interruptions

Description

Test system behavior when waterflow is repeatedly interrupted and restored multiple times during a single cycle.

***Initial Setup**:*

- Reset system.
- Choose Quick Wash (Preset 2).
- Ensure door_open_signal = 0 (door is closed).
- User presses start.
- Normal inputs:
  - waterflow = 1 (initially normal).
  - power_cut = 0 (no power interruptions).

***Error Events**:*

1. During WASH, waterflow = 0 (waterflow is cut).
2. After a short delay, waterflow = 1 (waterflow is restored).
3. This repeats for a total of 3 interruptions during the WASH and FILL_WATER states.

***Expected Behavior**:*

- On each waterflow cut:
  - System transitions to ERROR state and waits for waterflow = 1.
- On each waterflow restoration:
  - System resumes from the paused state without restarting the cycle.
- After 3 interruptions, the cycle progresses normally:
  - WASH → FILL_WATER → RINSE → SPIN → WASH_COMPLETE.
- Alarm remains inactive throughout the process unless a persistent error occurs.

- Testbench Code

```
module scenarios_TB;

    // Inputs
    reg clk;
    reg reset;
    reg start;
    reg pause;
    reg [2:0] preset;
    reg [2:0] custom_temp;
    reg [2:0] custom_speed;
    reg [3:0] custom_cycles;
    reg [5:0] custom_dry_timer;
    reg [2:0] custom_water_level;
    reg door_open_signal;
    reg waterflow;
    reg [2:0] water_level_reading;
    reg power_cut;

    // Outputs
    wire [2:0] speed;
    wire [2:0] temp;
    wire [3:0] cycles;
    wire [5:0] dry_timer;
    wire [2:0] water_level;
    wire rinse_complete;
    wire [5:0] full_state;
    wire [3:0] state;
    wire [1:0] errorpause;
    wire alarm;
    wire finishedAlarm;

    // Instantiate the WashingMachine module
    WashingMachine dut (
        .clk(clk),
        .reset(reset),
        .start(start),
        .pause(pause),
        .preset(preset),
        .custom_temp(custom_temp),
        .custom_speed(custom_speed),
        .custom_cycles(custom_cycles),
        .custom_dry_timer(custom_dry_timer),
        .custom_water_level(custom_water_level),
        .door_open_signal(door_open_signal),
        .waterflow(waterflow),
        .water_level_reading(water_level_reading),
        .power_cut(power_cut),
        .speed(speed),
        .temp(temp),
        .cycles(cycles),
        .dry_timer(dry_timer),
        .water_level(water_level),
        .rinse_complete(rinse_complete),
        .full_state(full_state),
        .state(state),
        .errorpause(errorpause),
        .alarm(alarm),
```

```
    .finishedAlarm(finishedAlarm)
);

// Clock generation
always #5 clk = ~clk;

initial begin
    // Initialize Inputs
    clk = 0;
    reset = 0;
    start = 0;
    pause = 0;
    preset = 0;
    custom_temp = 0;
    custom_speed = 0;
    custom_cycles = 0;
    custom_dry_timer = 0;
    custom_water_level = 0;
    door_open_signal = 0;
    waterflow = 1;
    water_level_reading = 0;
    power_cut = 0;

    // Reset the system
    reset = 1;
    #10 reset = 0;



    // Test Case A1: WASH to FILL_WATER to WASH
    preset = 7;                     // Custom preset
    custom_speed = 2; custom_temp = 3; custom_dry_timer = 50;

    custom_water_level = 3;
    custom_cycles = 10;
    start = 1;
    #30 start = 0;

    water_level_reading = 4;        // Initial state in WASH
    #30 water_level_reading = 2;    // Transition to FILL_WATER
    #30 water_level_reading = 3;    // Transition back to WASH



    // Test Case A2: DRAIN to FILL_WATER to RINSE to DRAIN
    reset = 1; #10 reset = 0; // Reset
    preset = 7; // Custom preset

    custom_speed = 2; custom_temp = 5; custom_dry_timer = 50;
    custom_water_level = 3;
    custom_cycles = 1; // Directly to DRAIN state
    start = 1;
```

```
        #20 start = 0;
        #20 water_level_reading = 1; // Transition to FILL_WATER
        #20 water_level_reading = 3; // Transition to RINSE
        #100; // Wait for one rinse cycle, back to DRAIN


        // Test Case A3: Closed/Open Door Start
        reset = 1; #10 reset = 0; // Reset
        preset = 1; // Cotton preset
        door_open_signal = 1; // Door open
        start = 1; #10 start = 0; #30; // No state change
        door_open_signal = 0; // Close door
        start = 1; #20 start = 0; // State transitions to RUN



        // Test Case A4: Invalid Custom Input Behavior
        reset = 1; #10 reset = 0; // Reset
        preset = 7; // Custom preset
        custom_temp = 0; custom_dry_timer = 0; custom_water_level = 0;

        custom_speed = 6;
        custom_cycles =11;
        start = 1;
        #30 start = 0;
        custom_speed = 5;
        custom_cycles =10;
        start = 1; #20 start = 0;



        // Test Case B1: Wool Preset Normal Operation
        reset = 1; #10 reset = 0; // Reset
        preset = 0; // Wool preset
        start = 1; #20 start = 0;
        water_level_reading = 6; waterflow = 1; power_cut = 0; pause=0; door_open_signal=0;
        #1000; // Simulate normal operation with fluctuating water_level_reading

        // Test Case B2: Quick Wash with Interruptions
        reset = 1; #10 reset = 0; // Reset
        preset = 2; // Quick Wash preset
        start = 1; #20 start = 0;
        water_level_reading = 4; waterflow = 1; power_cut = 0;
        #50 pause = 1; #5 pause = 0; door_open_signal = 1; #10 start = 1; #5; start=0;  // Pause and open door
        #20 door_open_signal = 0; start = 1; #5; start=0; // Resume from DRAIN
        #1000;


        // Test Case B3: Dry only
        reset = 1; #10 reset = 0; // Reset
        preset = 4; // Dry only preset
        start = 1; #20 start = 0;
        water_level_reading = 0; waterflow = 0; power_cut = 0;  door_open_signal = 0; pause=0;
        #300;
```

```
166
167
168
169
170         //Upper Edge Case
171         reset = 1; #10; reset = 0; // Reset the system
172         preset=7;
173         custom_cycles = 10;      // Maximum cycles = 10
174         custom_speed = 5;        // Maximum speed = 5
175         custom_temp = 5;         // Maximum temperature = 5
176         custom_water_level = 7;  // Maximum water level = 5
177         custom_dry_timer = 60;
178         door_open_signal = 0;         // Door is closed
179         start = 1; #10; start = 0;    // Start washing
180         waterflow = 1;                // Normal water flow
181         water_level_reading=7;
182         power_cut = 0;                // No power interruptions
183         #1000;
184
185
186         //lower Edge Case
187         reset = 1; #10; reset = 0; // Reset the system
188         custom_cycles = 1;      // Minimum cycles = 1
189         custom_speed = 1;        // Minimum speed = 1
190         custom_temp = 0;         // Minimum temperature = 0
191         custom_water_level = 1;  // Minimum water level = 1
192         custom_dry_timer = 0;
193         door_open_signal = 0;         // Door is closed
194         start = 1; #10; start = 0;    // Start washing
195         waterflow = 1;                // Normal water flow
196         water_level_reading=1;
197         power_cut = 0;                // No power interruptions
198         #1000;
199
200
201         // Test Case C: Recovery from Waterflow and Power Interruptions
202         reset = 1; #10; reset = 0; // Reset the system
203         preset = 3;                // Preset 3 (Wash and Dry)
204         door_open_signal = 0;      // Door is closed
205         start = 1; #10; start = 0; // Start washing
206         waterflow = 1;             // Normal waterflow
207         power_cut = 0;             // No power interruptions
208
209         water_level_reading=3; #20;
210         waterflow = 0; #50; // Waterflow cut
211
212         // Restore waterflow
213         waterflow = 1;
214         water_level_reading=6;#50;
215
216         // Simulate power cut during
217         power_cut = 1; #50; // Power cut
218         // Restore power
219         power_cut = 0; #50;
```

```
220
221
222
223         //Test Case D: Recovery from Continuous Waterflow Interruptions
224         reset = 1; #10; reset = 0; // Reset the system
225         preset = 3'b010;           // Preset 2 (Quick Wash)
226         door_open_signal = 0;      // Door is closed
227         start = 1; #10; start = 0; // Start washing
228         waterflow = 1;             // Normal waterflow
229         power_cut = 0;             // No power interruptions
230
231
232
233         water_level_reading=0;#50;
234         waterflow = 0; #20; // Waterflow cut
235
236         waterflow = 1; #20;
237         water_level_reading=2;#50;
238
239         waterflow = 0; #20; // Waterflow cut
240
241         waterflow = 1; #20;
242         water_level_reading=4;#50;
243
244         waterflow = 0; #20; // Waterflow cut
245
246         waterflow = 1; #20;
247       water_level_reading=6;#50;
248
249
250       $stop; // End simulation
251
252     end
253   endmodule
```

- Coverage Analysis

Objective

The primary goal of coverage analysis is to ensure that all functional and structural aspects of the washing machine controller design are thoroughly tested. By enabling coverage analysis, we verify that the test scenarios comprehensively exercise all critical parts of the Verilog code, including state transitions, input conditions, and recovery mechanisms.

Coverage Metrics

We utilized the following coverage metrics to evaluate the completeness of the testing process:

I. **Code Coverage**:
   - Statement Coverage: Confirms that every line of code in the design was executed during the simulation.
   - Branch Coverage: Ensures all possible branches (e.g., if-else and case statements) were exercised.
   - Expression Coverage: Verifies that all logical expressions (e.g., AND, OR, NOT) were evaluated for both true and false outcomes.

2. **FSM (Finite State Machine) Coverage**:
   - State Coverage: Checks whether all states in the FSM were reached during the simulation.
   - Transition Coverage: Verifies that all possible state transitions were triggered by the test scenarios.
   - Path Coverage: Confirms that all possible paths between states were exercised.

3. **Toggle Coverage**:
    - Tracks whether each bit of the design's internal signals toggled between 0 and 1.

4. **Functional Coverage**:
    - Monitors specific functional scenarios (e.g., handling of waterflow interruptions, power failures, and invalid inputs) to ensure all expected behaviors are tested.

Coverage Results

After running the .do file for coverage analysis, the results indicated the following:

1. Code Coverage: Achieved 88.70% branch coverage, with a 100% statement coverage, indicating that all parts of the design were exercised. It also contains 58.06% condition coverage.

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        62        55         7    88.70%
```

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                     216       216         0   100.00%
```

2. FSM Coverage:
    - State Coverage: 100%, confirming all FSM states (e.g., WASH, FILL_WATER, RINSE, etc.) were reached.
    - Transition Coverage: 31.50%, as most transitions were not covered in the directed tests, as they are illegal and invalid. Only the valid transitions were seen in the testing results.

```
FSM Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    FSM States                      12        12         0   100.00%
    FSM Transitions                 73        23        50    31.50%
```

3. Toggle Coverage: Achieved 91.66%, with a few infrequent signals not toggled during specific scenarios.

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        132       121        11    91.66%
```

4. Assertion Coverage: Achieved 75.00% and 77.22% by instance, indicating that the majority of the critical properties were validated during simulation. The remaining 25% of uncovered cases were rare edge cases unlikely to be visited.

```
Total Coverage By Instance (filtered view): 77.22%
```

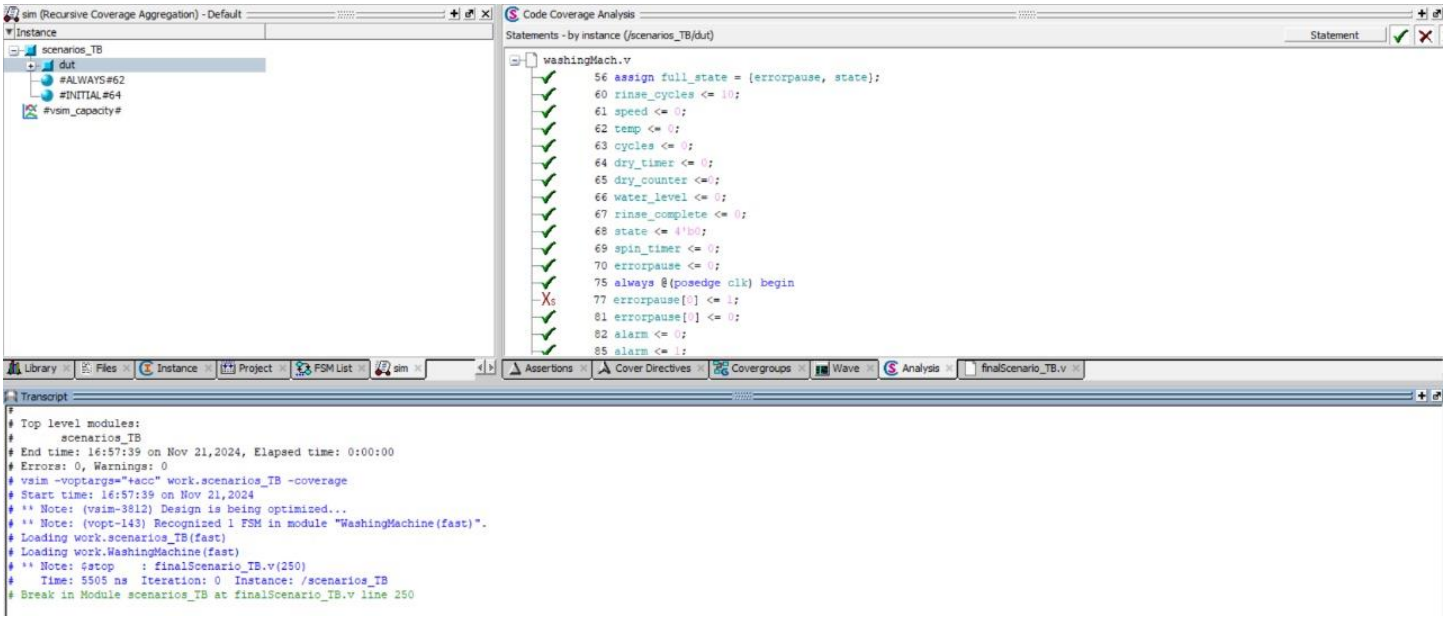
```
Assertion Coverage:
    Assertions                       12         9         3    75.00%
```

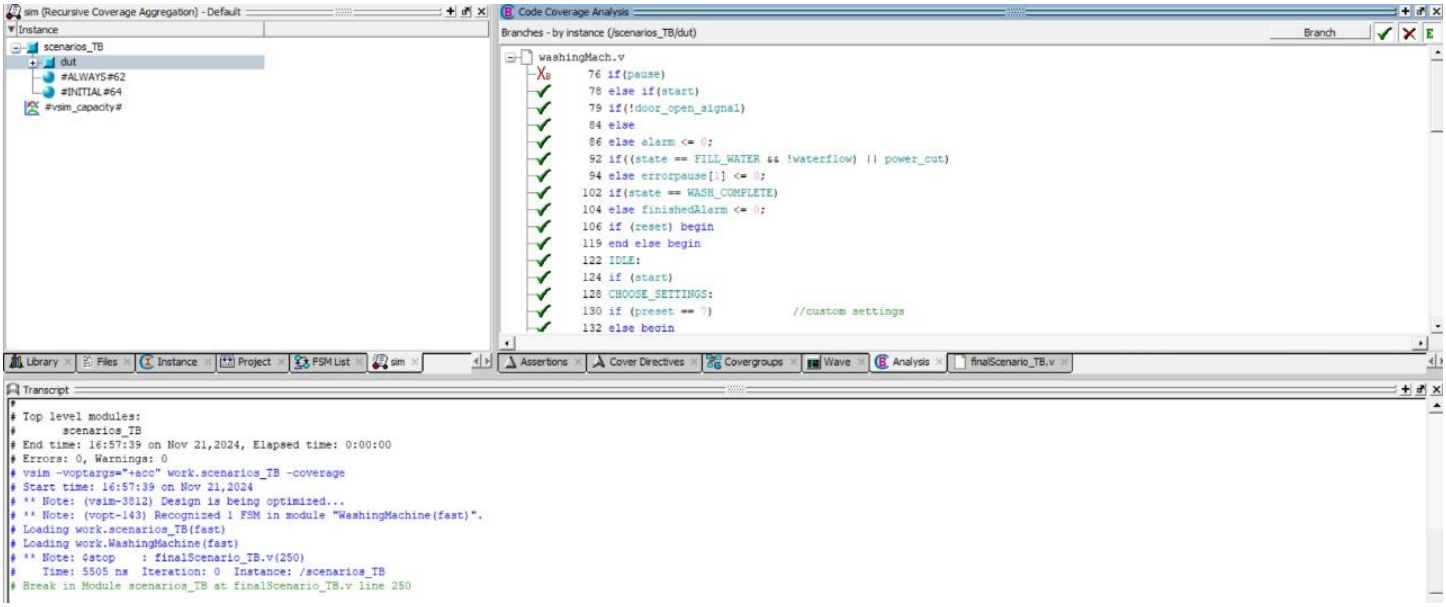
Analysis and Observations

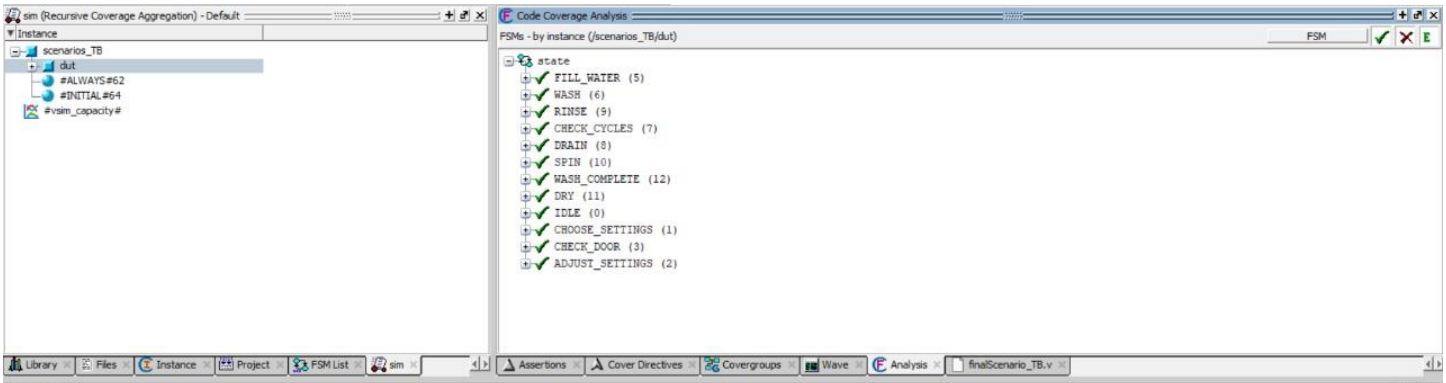
- The directed test scenarios effectively covered most design functionalities, especially critical error recovery mechanisms such as power and waterflow interruptions.
- Minor gaps in transition and toggle coverage indicate the need for additional tests targeting rare edge cases or extended runtime behavior.
- Functional coverage gaps were primarily due to the absence of randomized stress tests or prolonged use-case simulations.

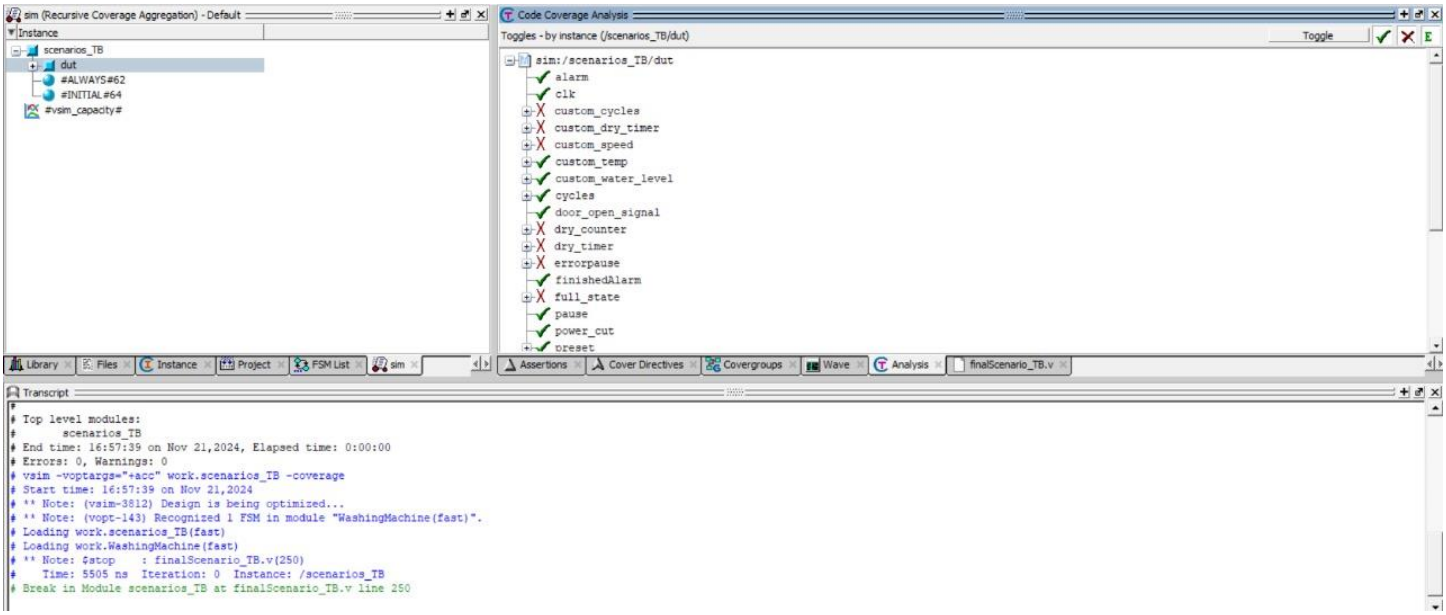
Conclusion

The coverage analysis demonstrates that the washing machine controller design has been rigorously tested with a high degree of confidence. The minor gaps identified can be addressed through additional targeted testing or randomized scenarios to achieve complete coverage. Overall, the design is well-verified against the specified functional and error recovery requirements.


sim (Recursive Coverage Aggregation) - Default
Instance
scenarios_TB
dut
#ALWAYS#62
#INITIAL#64
#vsim_capacity#
Code Coverage Analysis
Statements - by instance (/scenarios_TB/dut)
Statement
washingMach.v
56 assign full_state = {errorpause, state};
60 rinse_cycles <= 10;
61 speed <= 0;
62 temp <= 0;
63 cycles <= 0;
64 dry_timer <= 0;
65 dry_counter <=0;
66 water_level <= 0;
67 rinse_complete <= 0;
68 state <= 4'b0;
69 spin_timer <= 0;
70 errorpause <= 0;
75 always @(posedge clk) begin
77 errorpause[0] <= 1;
81 errorpause[0] <= 0;
82 alarm <= 0;
85 alarm <= 1;
Library
Files
Instance
Project
FSM List
sim
Assertions
Cover Directives
Covergroups
Wave
Analysis
finalScenario_TB.v
Transcript
Top level modules:
scenarios_TB
End time: 16:57:39 on Nov 21,2024, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
vsim -voptargs="+acc" work.scenarios_TB -coverage
Start time: 16:57:39 on Nov 21,2024
** Note: (vsim-3812) Design is being optimized...
** Note: (vopt-143) Recognized 1 FSM in module "WashingMachine(fast)".
Loading work.scenarios_TB(fast)
Loading work.WashingMachine(fast)
** Note: $stop : finalScenario_TB.v(250)
Time: 5505 ns Iteration: 0 Instance: /scenarios_TB
Break in Module scenarios_TB at finalScenario_TB.v line 250

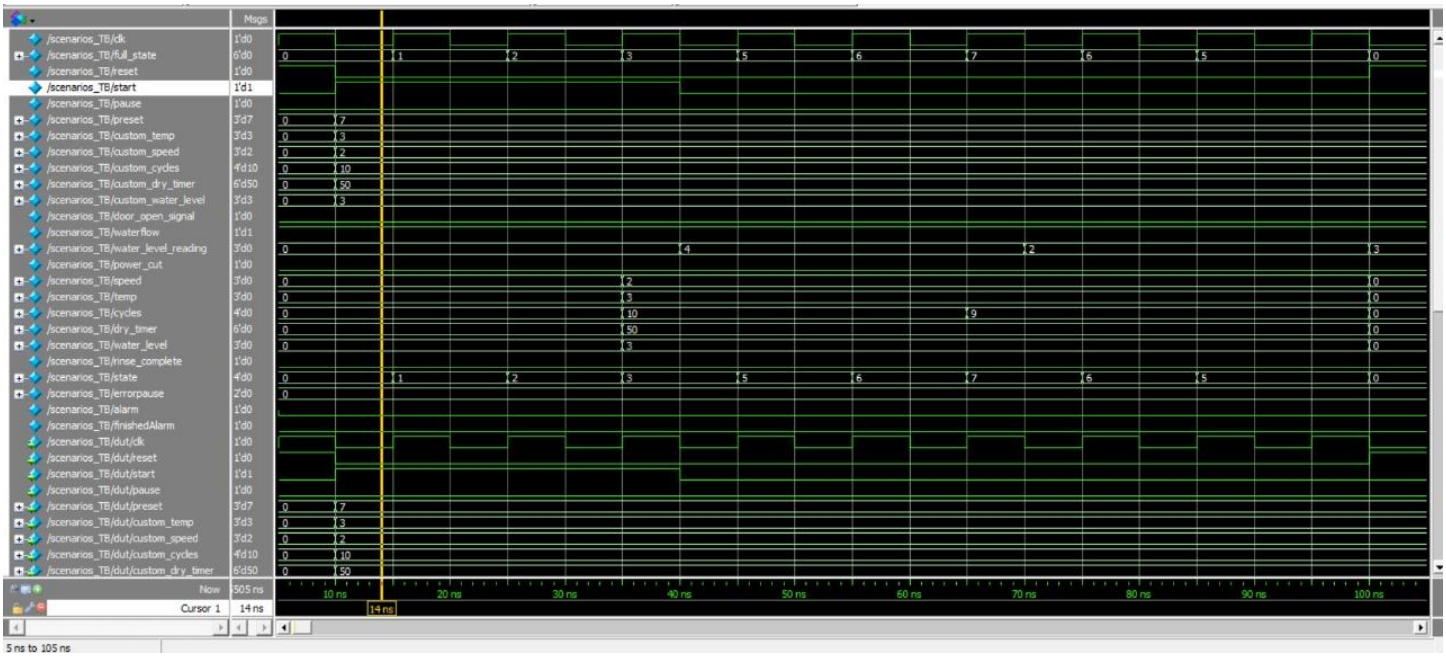

sim (Recursive Coverage Aggregation) - Default
Instance
scenarios_TB
dut
#ALWAYS#62
#INITIAL#64
#vsim_capacity#
Code Coverage Analysis
Branches - by instance (/scenarios_TB/dut)
Branch
washingMach.v
76 if(pause)
78 else if(start)
79 if(!door_open_signal)
84 else
86 else alarm <= 0;
92 if((state == FILL_WATER && !waterflow) || power_cut)
94 else errorpause[1] <= 0;
102 if(state == WASH_COMPLETE)
104 else finishedAlarm <= 0;
106 if (reset) begin
119 end else begin
122 IDLE:
124 if (start)
128 CHOOSE_SETTINGS:
130 if (preset == 7) //custom settings
132 else begin
Library
Files
Instance
Project
FSM List
sim
Assertions
Cover Directives
Covergroups
Wave
Analysis
finalScenario_TB.v
Transcript
Top level modules:
scenarios_TB
End time: 16:57:39 on Nov 21,2024, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
vsim -voptargs="+acc" work.scenarios_TB -coverage
Start time: 16:57:39 on Nov 21,2024
** Note: (vsim-3812) Design is being optimized...
** Note: (vopt-143) Recognized 1 FSM in module "WashingMachine(fast)".
Loading work.scenarios_TB(fast)
Loading work.WashingMachine(fast)
** Note: $stop : finalScenario_TB.v(250)
Time: 5505 ns Iteration: 0 Instance: /scenarios_TB
Break in Module scenarios_TB at finalScenario_TB.v line 250


sim (Recursive Coverage Aggregation) - Default
Instance
scenarios_TB
dut
#ALWAYS#62
#INITIAL#64
#vsim_capacity#
Code Coverage Analysis
FSMs - by instance (/scenarios_TB/dut)
FSM
state
FILL_WATER (5)
WASH (6)
RINSE (9)
CHECK_CYCLES (7)
DRAIN (8)
SPIN (10)
WASH_COMPLETE (12)
DRY (11)
IDLE (0)
CHOOSE_SETTINGS (1)
CHECK_DOOR (3)
ADJUST_SETTINGS (2)
Library
Files
Instance
Project
FSM List
sim
Assertions
Cover Directives
Covergroups
Wave
Analysis
finalScenario_TB.v
Transcript
Top level modules:
scenarios_TB
End time: 16:57:39 on Nov 21,2024, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
vsim -voptargs="+acc" work.scenarios_TB -coverage
Start time: 16:57:39 on Nov 21,2024
** Note: (vsim-3812) Design is being optimized...
** Note: (vopt-143) Recognized 1 FSM in module "WashingMachine(fast)".
Loading work.scenarios_TB(fast)
Loading work.WashingMachine(fast)
** Note: $stop : finalScenario_TB.v(250)
Time: 5505 ns Iteration: 0 Instance: /scenarios_TB
Break in Module scenarios_TB at finalScenario_TB.v line 250

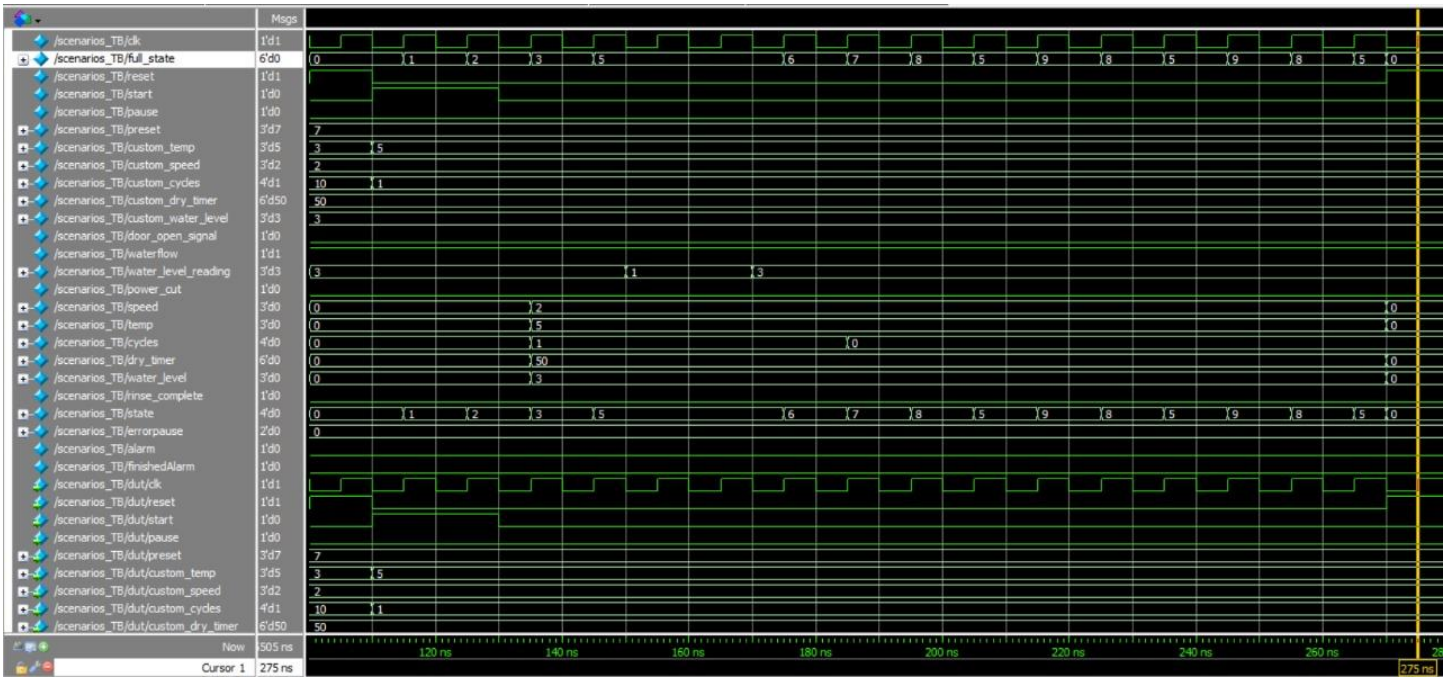

sim (Recursive Coverage Aggregation) - Default
Instance
scenarios_TB
dut
#ALWAYS#62
#INITIAL#64
#vsim_capacity#
Code Coverage Analysis
Toggles - by instance (/scenarios_TB/dut)
Toggle
sim:/scenarios_TB/dut
alarm
clk
custom_cycles
custom_dry_timer
custom_speed
custom_temp
custom_water_level
cycles
door_open_signal
dry_counter
dry_timer
errorpause
finishedAlarm
full_state
pause
power_cut
preset
Library
Files
Instance
Project
FSM List
sim
Assertions
Cover Directives
Covergroups
Wave
Analysis
finalScenario_TB.v
Transcript
Top level modules:
scenarios_TB
End time: 16:57:39 on Nov 21,2024, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
vsim -voptargs="+acc" work.scenarios_TB -coverage
Start time: 16:57:39 on Nov 21,2024
** Note: (vsim-3812) Design is being optimized...
** Note: (vopt-143) Recognized 1 FSM in module "WashingMachine(fast)".
Loading work.scenarios_TB(fast)
Loading work.WashingMachine(fast)
** Note: $stop : finalScenario_TB.v(250)
Time: 5505 ns Iteration: 0 Instance: /scenarios_TB
Break in Module scenarios_TB at finalScenario_TB.v line 250

- Simultaneous Waveform Snippets

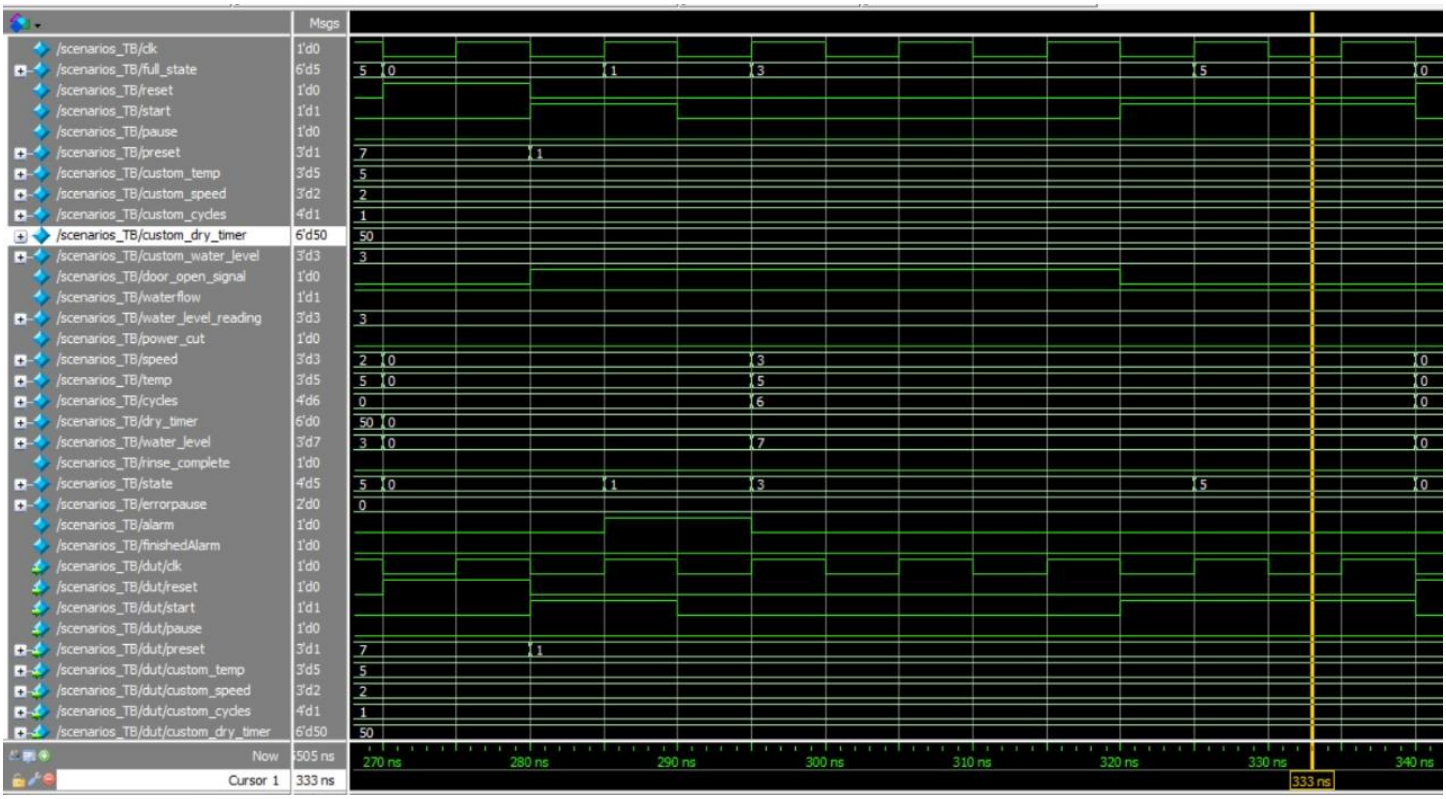
Test case 1: WASH to FILL_WATER to WASH



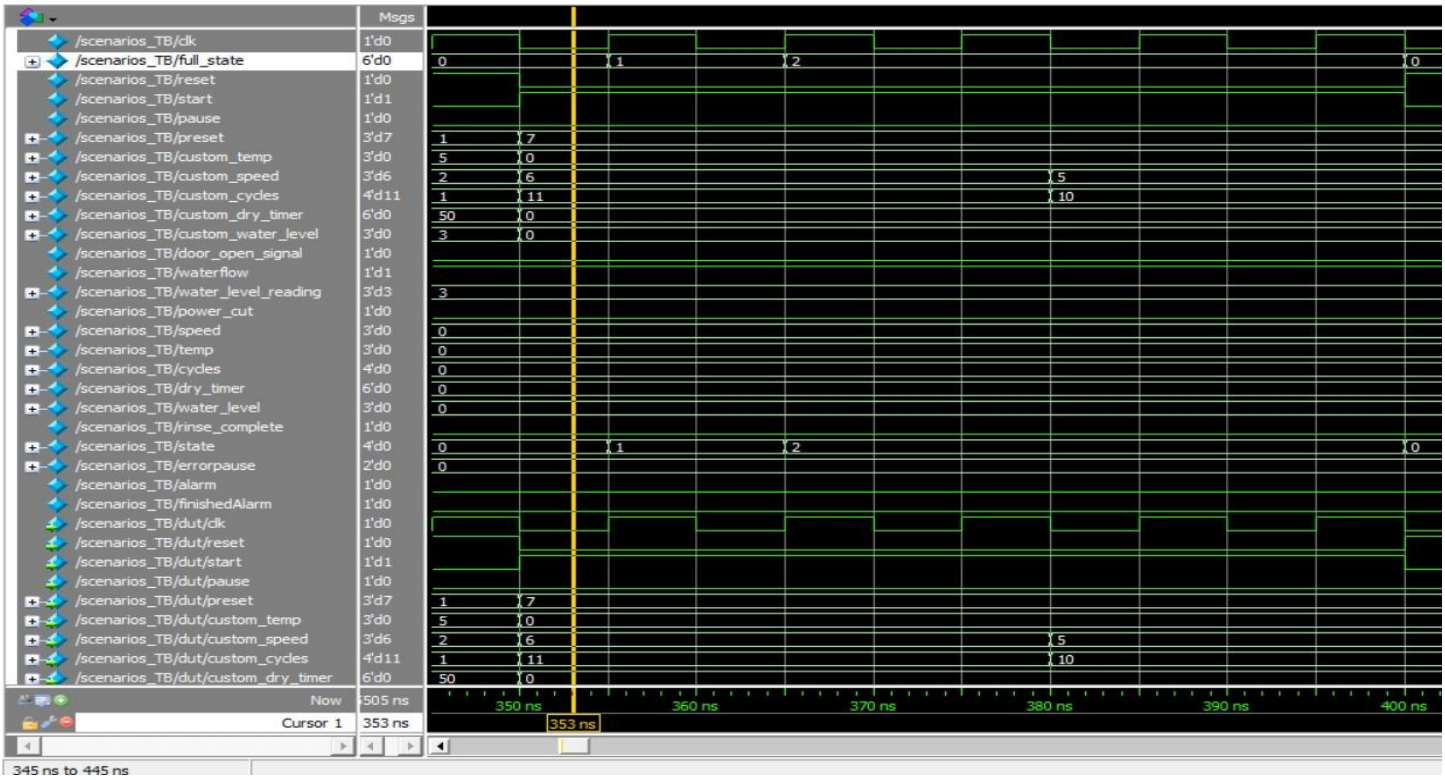
Test case 2: DRAIN to FILL_WATER to RINSE to DRAIN



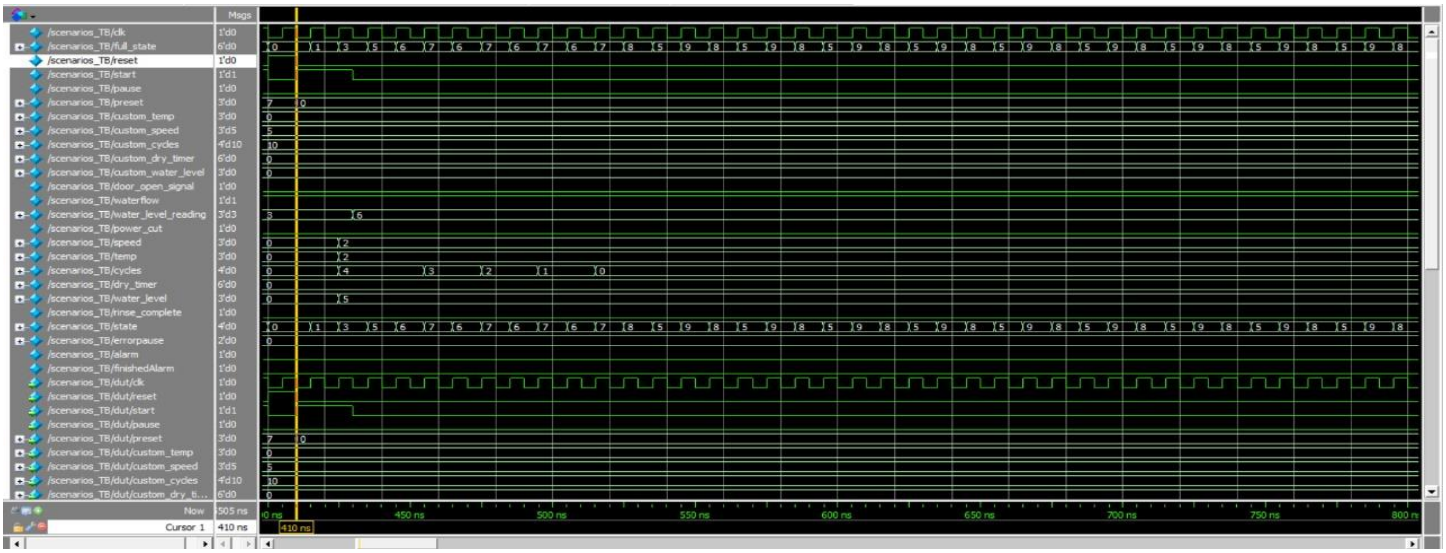
Test case 3: Closed/Open Door Start

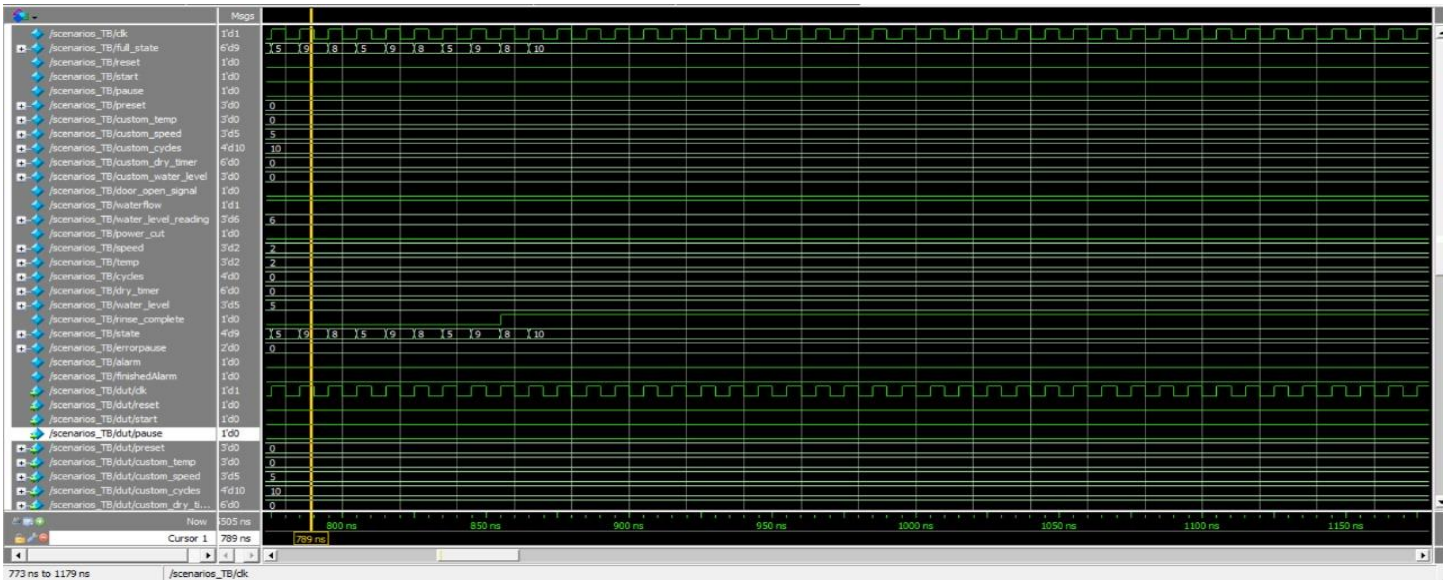


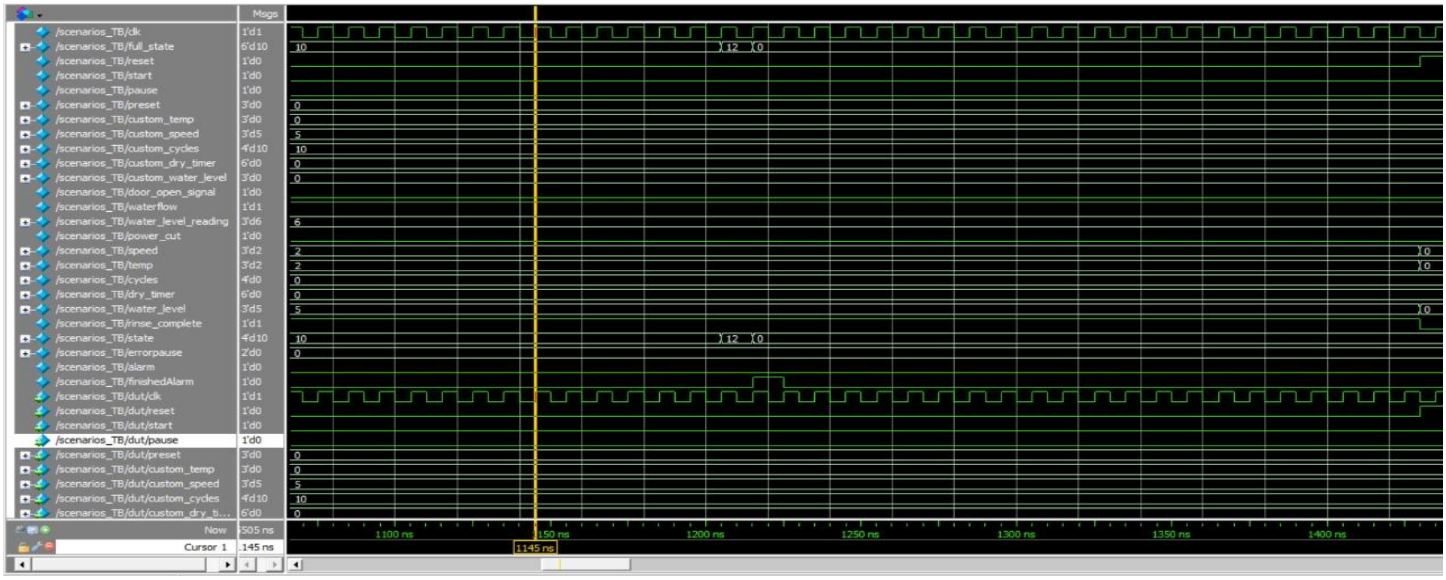
Test case 4: Invalid Custom Input Behavior



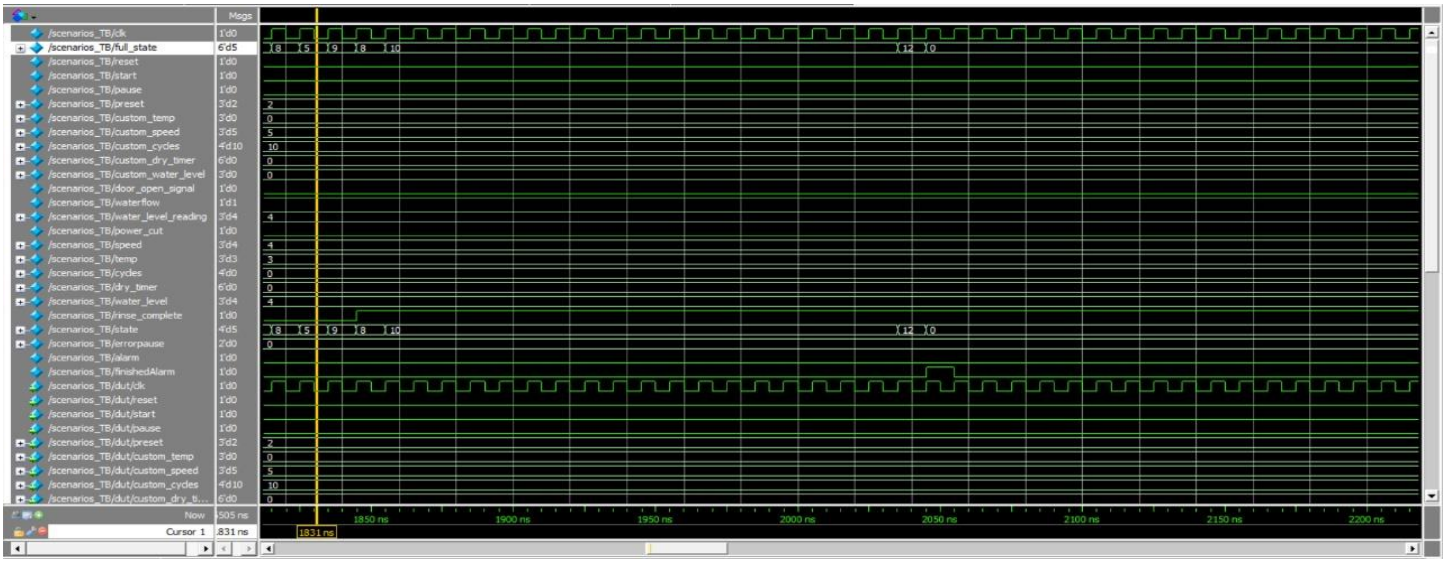
Test case 5: Wool Preset Normal Operation

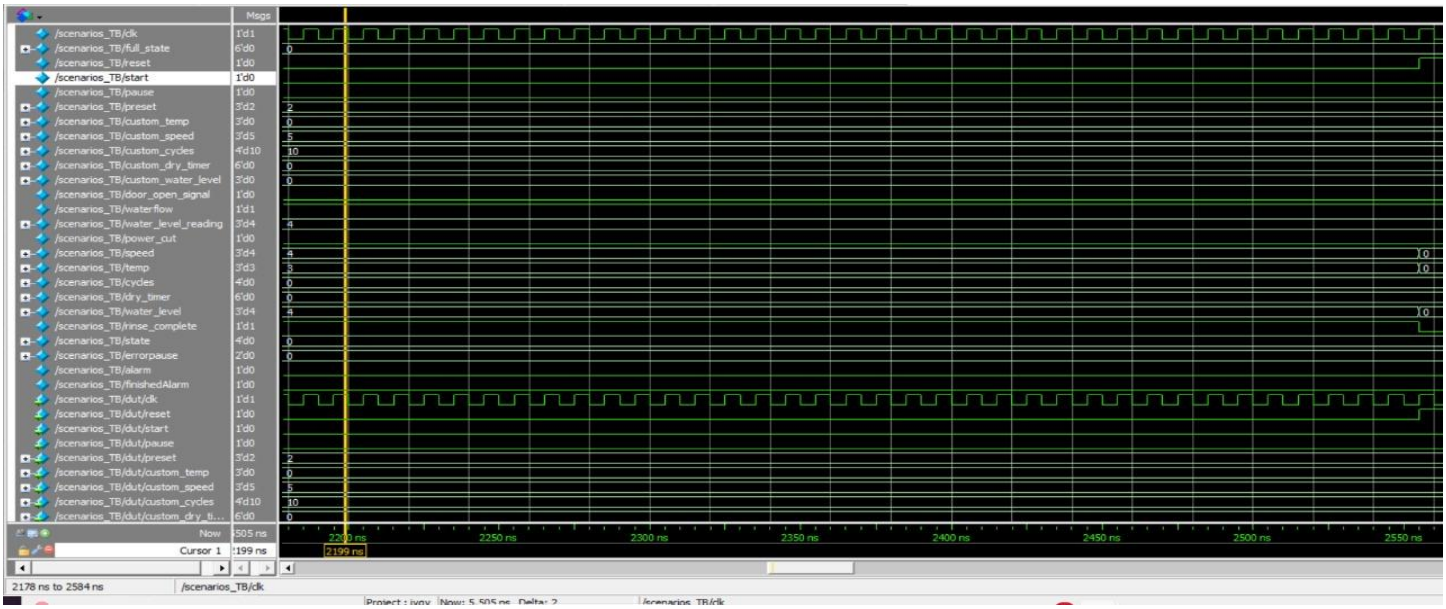




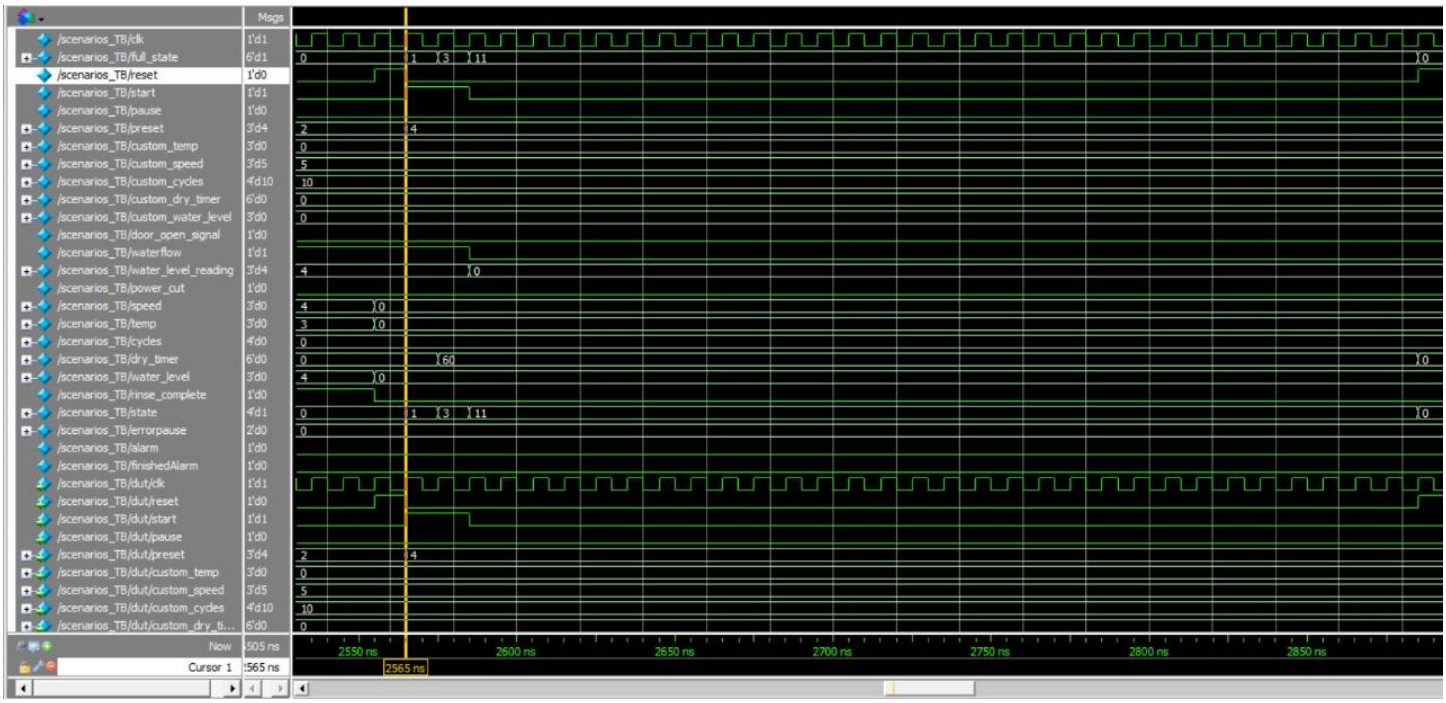


Test case 6: Quick Wash with Interruptions

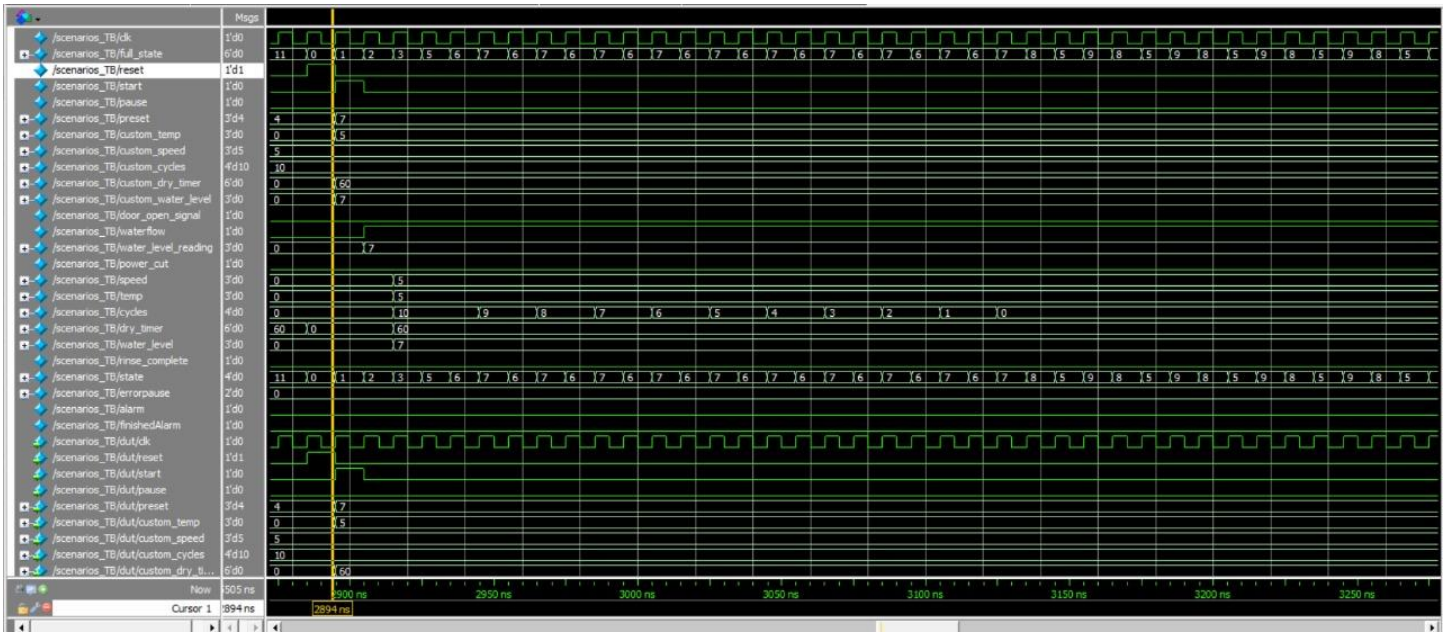


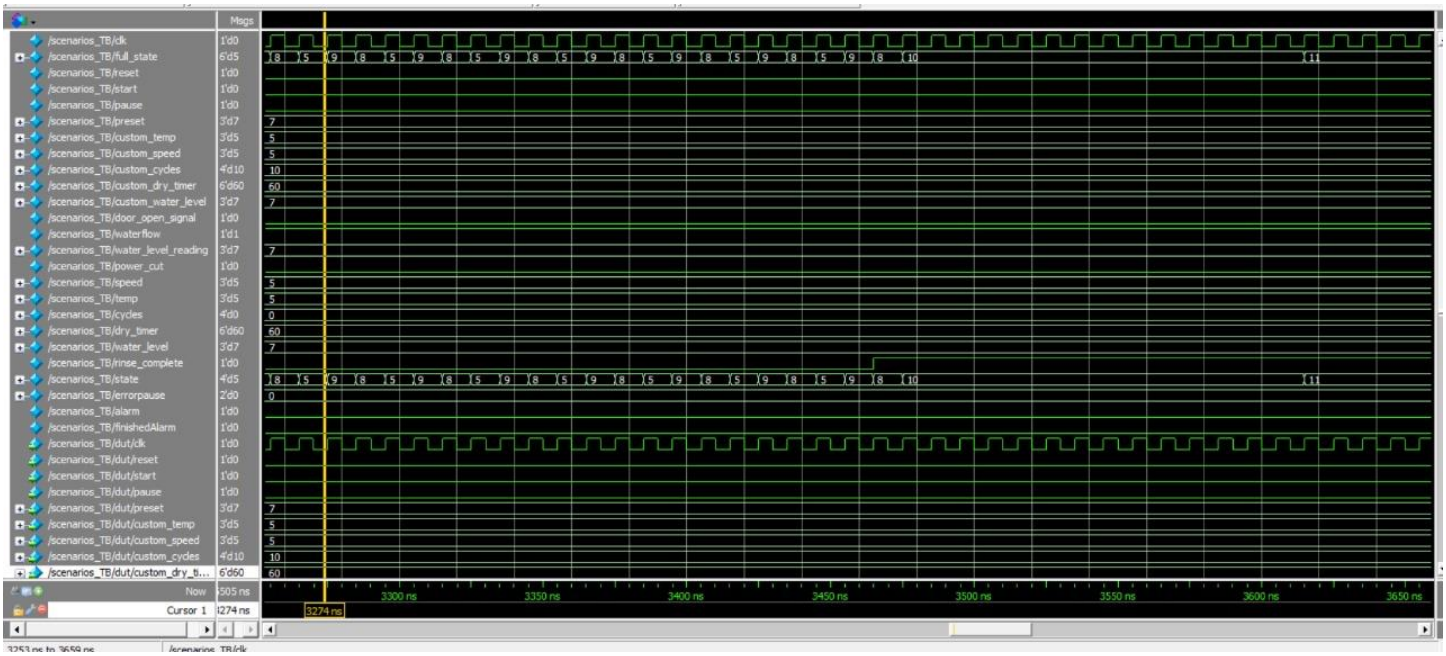


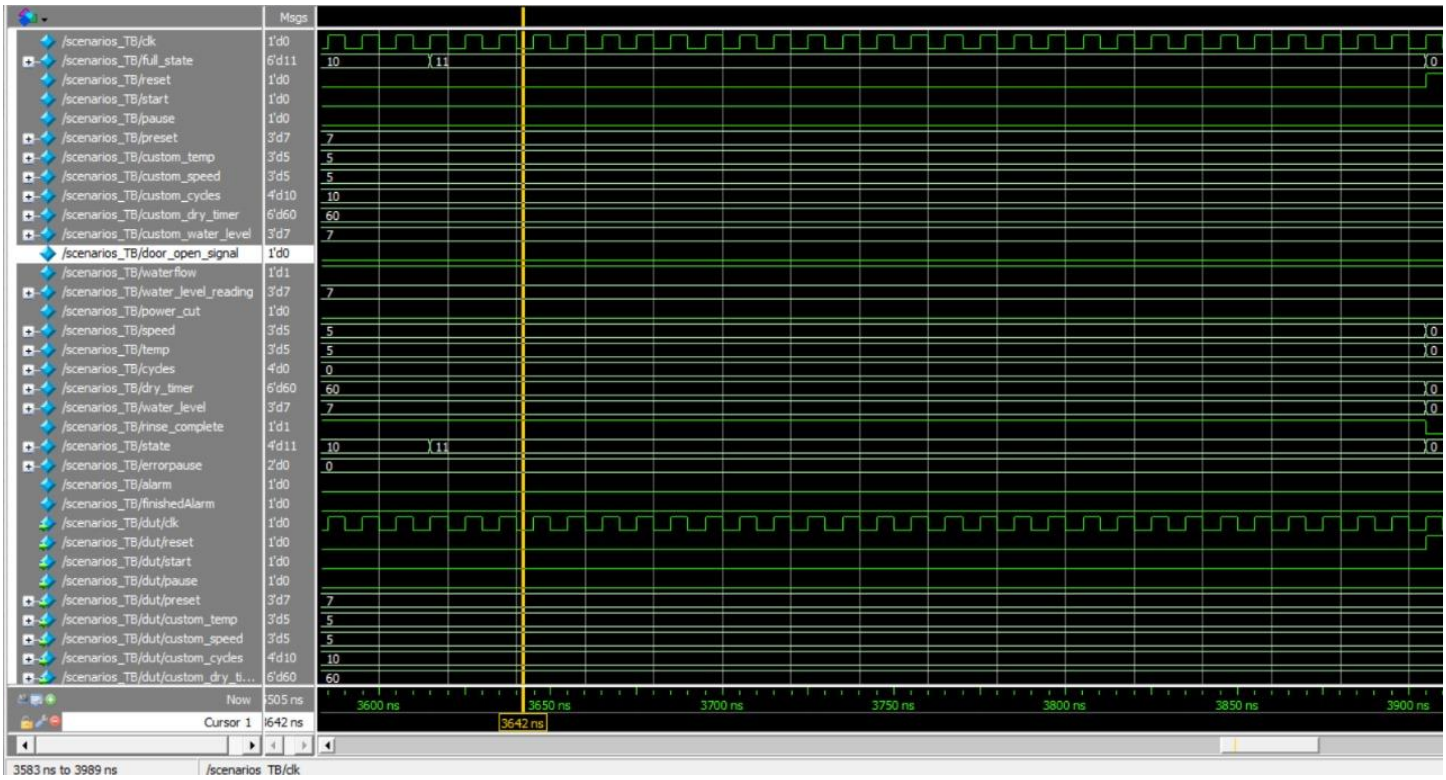
Test case 7: Dry Only Preset



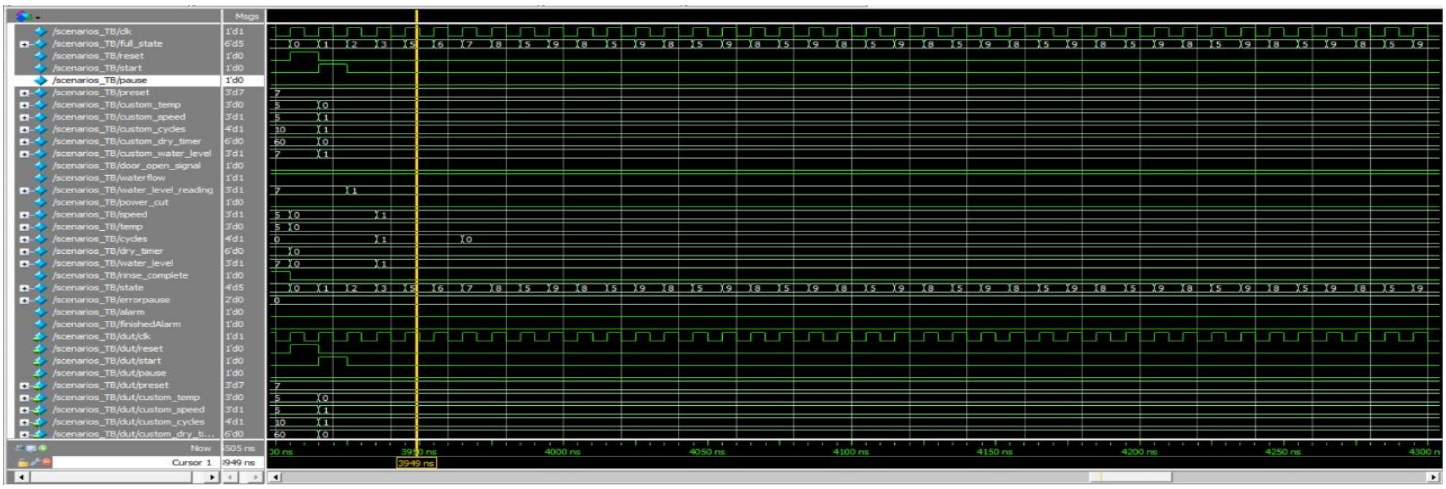
Test case 8: Upper Edge Case for Custom Inputs

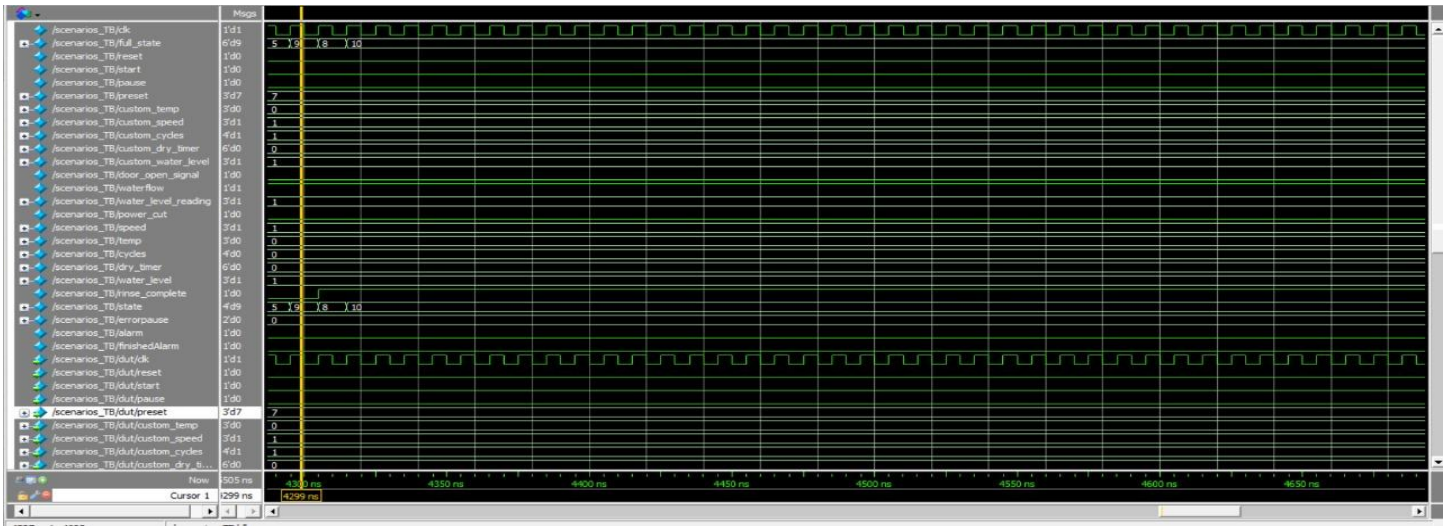


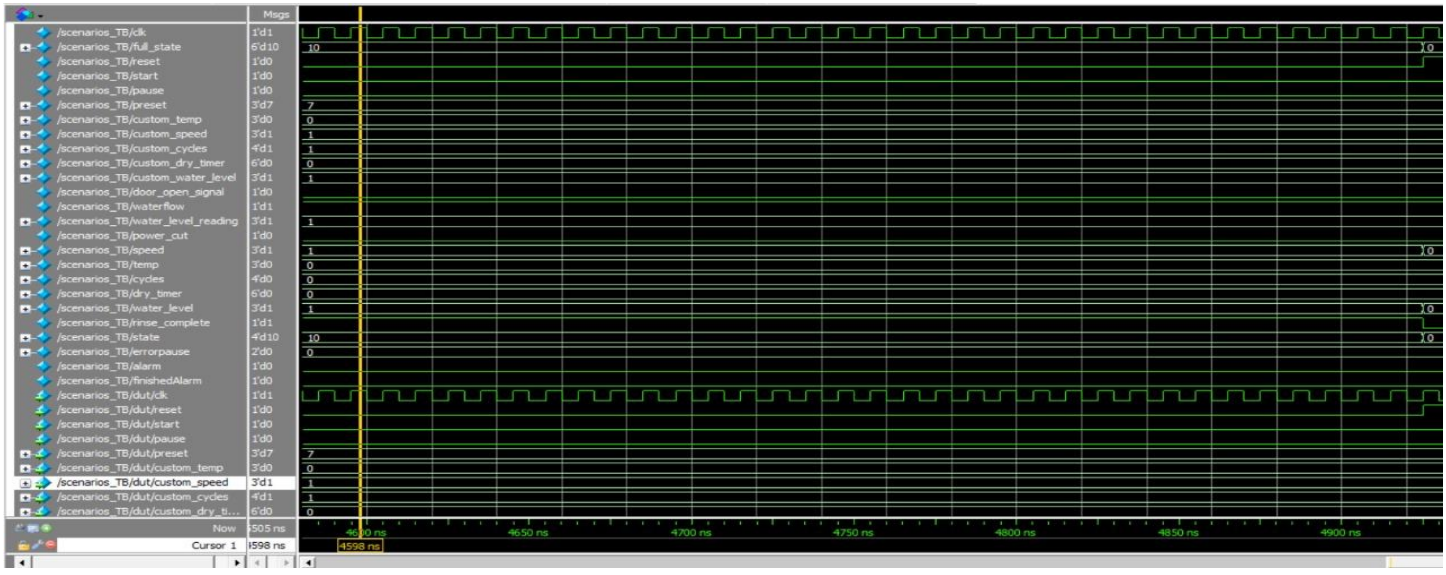

Msgs
/scenarios_TB/clk
/scenarios_TB/full_state
/scenarios_TB/reset
/scenarios_TB/start
/scenarios_TB/pause
/scenarios_TB/preset
/scenarios_TB/custom_temp
/scenarios_TB/custom_speed
/scenarios_TB/custom_cycles
/scenarios_TB/custom_dry_timer
/scenarios_TB/custom_water_level
/scenarios_TB/door_open_signal
/scenarios_TB/waterflow
/scenarios_TB/water_level_reading
/scenarios_TB/power_cut
/scenarios_TB/speed
/scenarios_TB/temp
/scenarios_TB/cycles
/scenarios_TB/dry_timer
/scenarios_TB/water_level
/scenarios_TB/rinse_complete
/scenarios_TB/state
/scenarios_TB/errorpause
/scenarios_TB/alarm
/scenarios_TB/finishedAlarm
/scenarios_TB/dut/clk
/scenarios_TB/dut/reset
/scenarios_TB/dut/start
/scenarios_TB/dut/pause
/scenarios_TB/dut/preset
/scenarios_TB/dut/custom_temp
/scenarios_TB/dut/custom_speed
/scenarios_TB/dut/custom_cycles
/scenarios_TB/dut/custom_dry_t...
Now
Cursor 1
3274 ns
3300 ns
3350 ns
3400 ns
3450 ns
3500 ns
3550 ns
3600 ns
3650 ns


Msgs
/scenarios_TB/clk
/scenarios_TB/full_state
/scenarios_TB/reset
/scenarios_TB/start
/scenarios_TB/pause
/scenarios_TB/preset
/scenarios_TB/custom_temp
/scenarios_TB/custom_speed
/scenarios_TB/custom_cycles
/scenarios_TB/custom_dry_timer
/scenarios_TB/custom_water_level
/scenarios_TB/door_open_signal
/scenarios_TB/waterflow
/scenarios_TB/water_level_reading
/scenarios_TB/power_cut
/scenarios_TB/speed
/scenarios_TB/temp
/scenarios_TB/cycles
/scenarios_TB/dry_timer
/scenarios_TB/water_level
/scenarios_TB/rinse_complete
/scenarios_TB/state
/scenarios_TB/errorpause
/scenarios_TB/alarm
/scenarios_TB/finishedAlarm
/scenarios_TB/dut/clk
/scenarios_TB/dut/reset
/scenarios_TB/dut/start
/scenarios_TB/dut/pause
/scenarios_TB/dut/preset
/scenarios_TB/dut/custom_temp
/scenarios_TB/dut/custom_speed
/scenarios_TB/dut/custom_cycles
/scenarios_TB/dut/custom_dry_ti...
Now
Cursor 1
3642 ns
3600 ns
3650 ns
3700 ns
3750 ns
3800 ns
3850 ns
3900 ns
3583 ns to 3989 ns
/scenarios_TB/clk

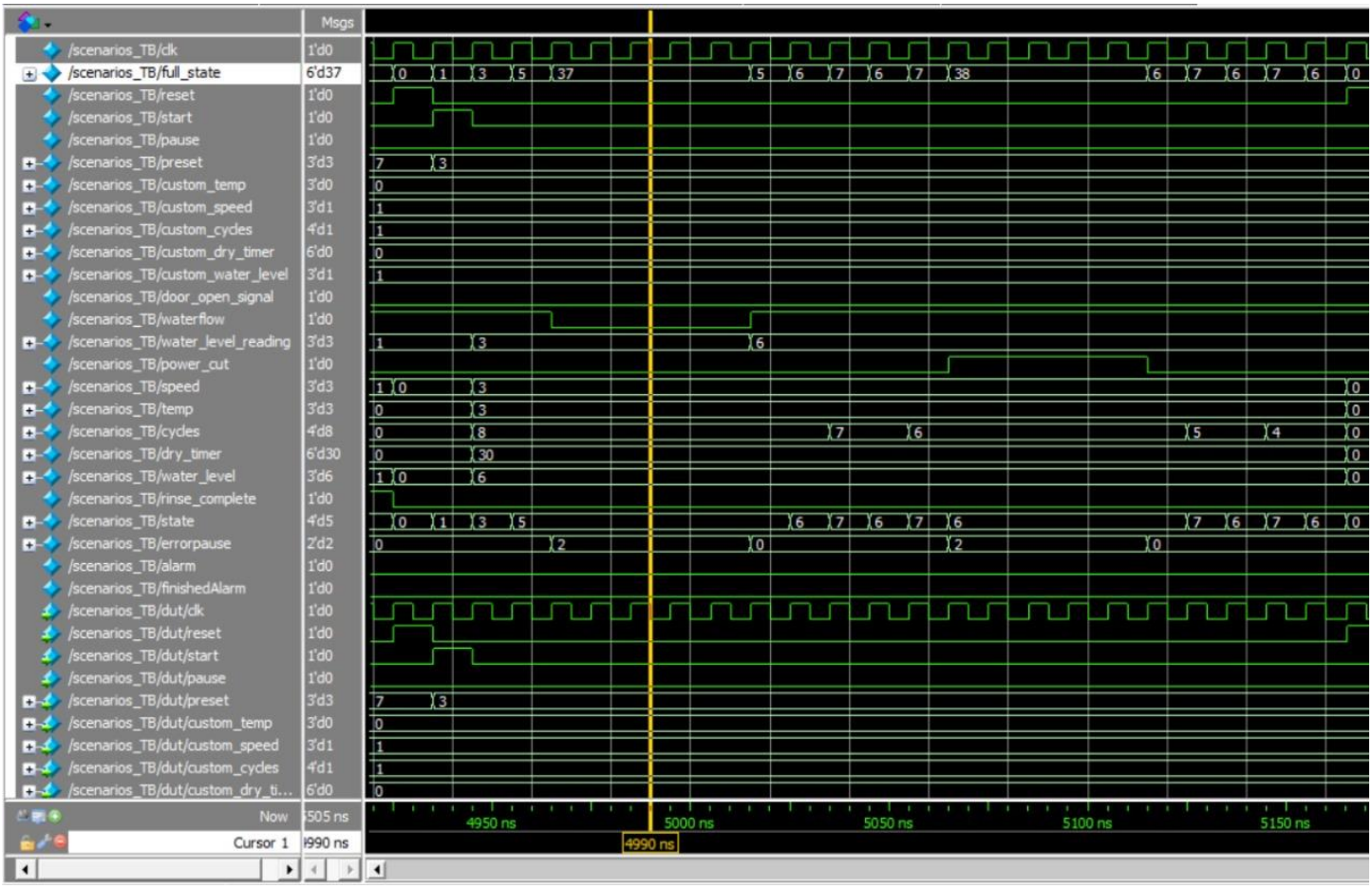
Test case 9: Lower Edge Case for Custom Inputs



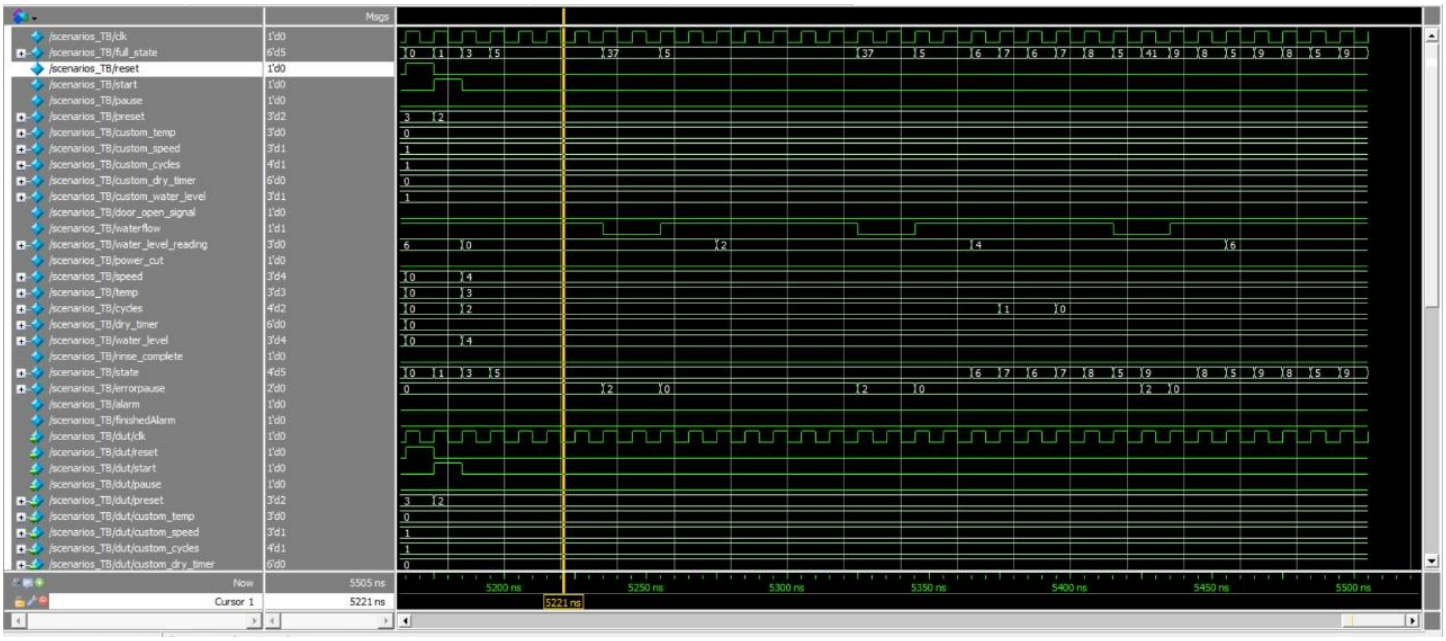




Test case 10: Recovery from Waterflow and Power Interruptions



Test case 11: Recovery from Continuous Waterflow Interruptions



- Waveform diagram with PSL assertions