REDTAIL LABORATORY EXERCISE

# Rain-Responsive Wiper Controller

DE1-SoC with SystemVerilog

LEVEL **Basic** | TIME **60-90 minutes** | PLATFORM **Altera DE1-SoC**

Design and test a one-bit controller for a simulated vehicle windshield wiper. Your SystemVerilog module decides whether the blade should move. The simulation owns the blade angle and reverses direction automatically at the endpoints.

The Wiper 3D simulation and original remote-laboratory exercise were created by Giovanna Lani and WebLab-Deusto at the University of Deusto in Spain. This lesson adapts their rain-driven challenge to the current REDTAIL SystemVerilog workflow. The completed implementation is available separately to verified instructors.


*Figure 1. Wiper 3D simulation by Giovanna Lani and WebLab-Deusto, University of Deusto (Spain).*

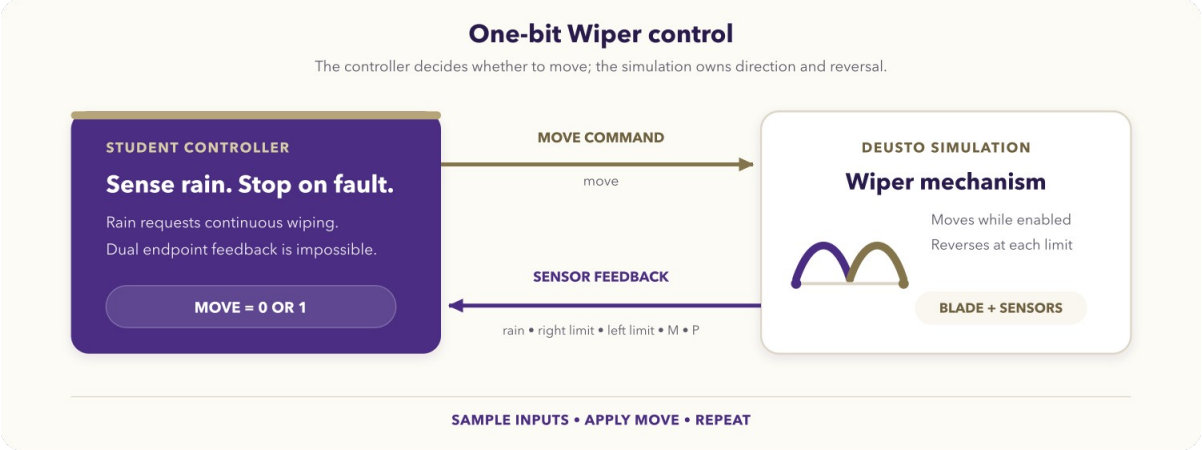


*Figure 2. One-bit Wiper control loop.*

### Learning objectives

After completing the activity, you should be able to:

- translate physical behavior into a Boolean control expression;
- implement and review combinational SystemVerilog logic;
- map logical signals to the DE1-SoC virtual GPIO interface;
- separate controller-owned behavior from simulation-owned animation;
- stop safely when endpoint feedback is physically contradictory;
- verify normal movement, pause, resume, reversal, and fault cases.

### Prerequisites

You should know SystemVerilog modules, continuous assignments or `always_comb`, and scalar/vector signals. You also need the DE1-SoC SystemVerilog Code-IDE and access to the Wiper simulation.

Open the REDTAIL Wiper simulation and the current DE1-SoC mapping before starting.

### System behavior

| Input | Meaning in the current simulation |
|---|---|
| `rainSensor` | Rain is active and automatic wiping is requested. |
| `rightSensor` | The blade is within the right endpoint range. |
| `leftSensor` | The blade is within the left endpoint range. |
| `mButton` | Optional M button; not required by the core assignment. |
| `pButton` | Optional P button; not required by the core assignment. |

| Output | Meaning |
|---|---|
| `move` | `1` allows movement; `0` pauses the blade at its current angle. |

The core module must assert `move` during rain, deassert it while dry, leave direction and reversal to the simulation, stop if both endpoint sensors are high, and ignore M and P.

### DE1-SoC signal mapping

| Runtime signal | SystemVerilog access | Direction |
|---|---|---|
| M button | `V_GPIO[23]` | Simulation to FPGA |
| P button | `V_GPIO[24]` | Simulation to FPGA |
| Move | `V_GPIO[26]` | FPGA to simulation |
| Rain sensor | `V_GPIO[28]` | Simulation to FPGA |
| Right endpoint | `V_GPIO[29]` | Simulation to FPGA |
| Left endpoint | `V_GPIO[30]` | Simulation to FPGA |

The indices identify LabsLand virtual GPIO lanes. The current REDTAIL mapping is authoritative over older development tables.

## Assignment

### 1. Create a truth table

Cover dry and rainy conditions at neither endpoint, each endpoint separately, and both endpoints together. Explain why the dual-endpoint row must stop.

### 2. Create the module

Create `wiper_deusto.sv`:

```
module wiper_deusto (
    input  logic       CLOCK_50,
    inout  wire [35:23] V_GPIO,
    output logic [9:0]  G_LEDR
);
    // Read the five mapped inputs.
    // Derive one move command without storing direction.
    // Drive only V_GPIO[26] toward the simulation.
endmodule
```

Complete the logic yourself. Assign every LED bit deterministically if you use `G_LEDR`; do not leave diagnostic outputs unknown.

### 3. Review combinational completeness

Confirm that every input combination produces a defined `move` value, that no latch is inferred, and that M/P cannot affect the required behavior.

### 4. Compile and program

1. Select the DE1-SoC SystemVerilog environment and Wiper simulation.
2. Add `wiper_deusto.sv` and select `wiper_deusto` as the top level.
3. Synthesize, resolve relevant warnings, and program the FPGA.
4. Keep the simulation visible while checking the mapped inputs and output.

### Required test sequence

| Test | Action | Expected behavior |
|---|---|---|
| Dry start | Start with rain disabled. | `move = 0`; the blade remains still. |
| Rain start | Enable rain. | `move = 1`; the blade starts moving. |
| Automatic reversal | Keep rain enabled through both endpoints. | The simulation reverses the blade while `move` remains high. |
| Pause and resume | Disable rain between endpoints, then re-enable it. | The blade pauses and later resumes. |
| Button isolation | Press M and P independently while dry. | The core solution remains stopped. |
| Sensor fault | Trigger both endpoint sensors together. | `move = 0` until the fault clears. |
| Recovery | Clear the fault while rain remains enabled. | Normal movement resumes. |

### Deliverables

Submit the truth table, `wiper_deusto.sv`, required test evidence, and a short explanation of automatic reversal ownership and the fault response.

### Optional extensions

- Define an instructor-approved manual behavior for M or P.
- Latch and display the dual-endpoint fault.
- Add SystemVerilog assertions for the required truth table.
- Compare a continuous assignment with an `always_comb` implementation.

### Completion checklist

- The source uses the current REDTAIL mapping.
- Every combinational input case has a defined output.
- Rain starts wiping and dry conditions stop it.
- The simulation owns normal reversal.
- Dual endpoints force a stop and M/P remain optional.
- All required evidence is ready to submit.