REDTAIL Simulation Documentation

Parking Lot

Hardware: FPGA
Language: English

Table of Contents

1. Defining Remote and Simulated Laboratories

What is a Remote Laboratory?

Remote laboratories are systems that combine hardware and software to allow students to perform real educational experiments over the Internet. In other words, students can control physical lab equipment and collect real data remotely, without being physically present in the laboratory.

These labs remove space and time barriers, allowing students to practice anytime, increasing the experience essential for STEM learning. They also encourage active and inquiry-based learning, promoting autonomy since students can work independently of instructors and lab hours.

What is a Simulated Laboratory?

Remote laboratories provide students with, over the Internet, access to real hardware hosted in a server room, while simulated laboratories virtualize the inputs, outputs, and internal behavior of hardware and entire systems. This allows students to work with digital twins of common industry standard hardware as well as unique and engaging systems that remote laboratories could not support, such as entire industrial dams, arcade machines, communication networks, and other expensive or niche hardware.

REDTAIL Hardware + Simulation Context

REDTAIL is a Remote Laboratory Simulation framework developed by the Remote Hub Lab at the University of Washington and LabsLand.

In REDTAIL, students write or upload code through a LabsLand web IDE. Their code is compiled and deployed to real physical devices (e.g., FPGAs and microcontrollers) that communicate bidirectionally with other devices running an instructor-selected simulation. These simulation devices communicate with LabsLand servers, allowing students to interact with and control the simulation via a web interface. This enables them to test their code in real time, as they would with a physical device or environment.

Each simulation can support many different lab assignments. We provide example specifications for each simulation, but instructors are welcome to create their own.

The remaining sections of this document describe one REDTAIL simulation in detail.

2. Simulation Overview

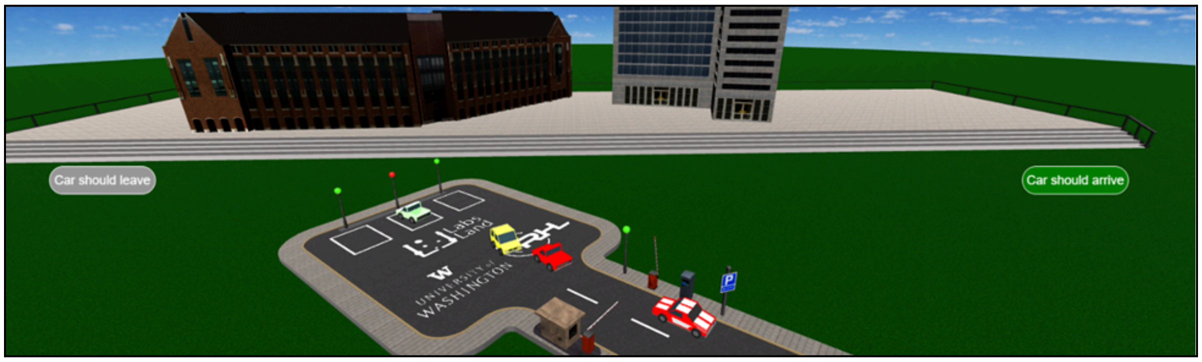


Figure 1: Screenshot of the Parking Lot Simulation

The 3D Parking Lot Simulation, shown in *Figure 1,* models a parking lot with **one entry and one exit gate**, and **three parking spots**. Only one car may enter or exit at a time. The **user** triggers car arrivals and departures via virtual buttons, while cars autonomously park or exit once inside. Gate operation is controlled via **GPIO outputs**, and car presence is detected through **GPIO inputs**.

Students can view the device under test via a live camera feed and interact with the simulation via buttons for car arrival and departure on the visualization iframe shown in Figure 1 and Figure 2. Using these web elements, students will test their design as they view the state of the parking lot change in real time.

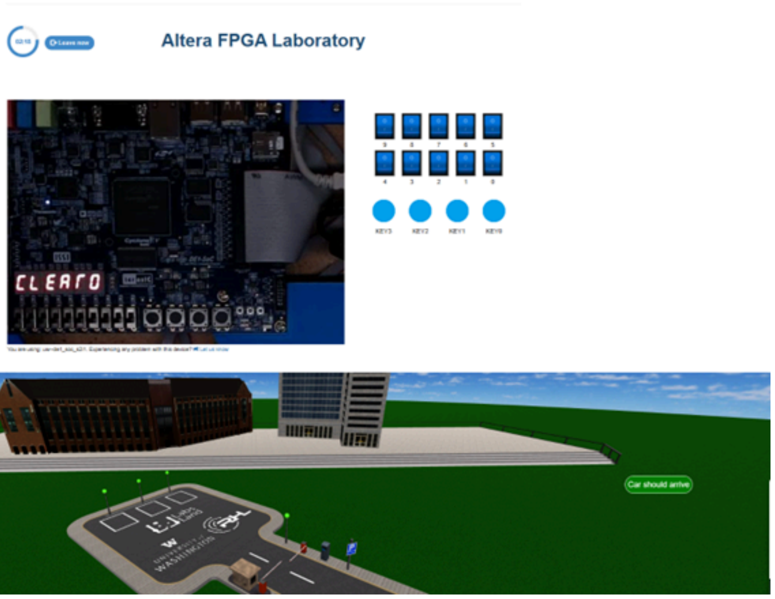


Figure 2: Simulation and Interaction Page

3. Simulation Components

3.1 Occupancy

The 3D parking lot can hold a maximum of three vehicles.

3.2 LEDs

There are four LEDs on the simulations: three of them to indicate the status of each parking space, and the fourth to indicate if the entire parking lot is full. These four LED colors can change between green and red. The color can be changed by asserting the appropriate V_GPIO connection to the FPGA (see the later page for GPIO mappings).

3.2 Car Arrival and Departure

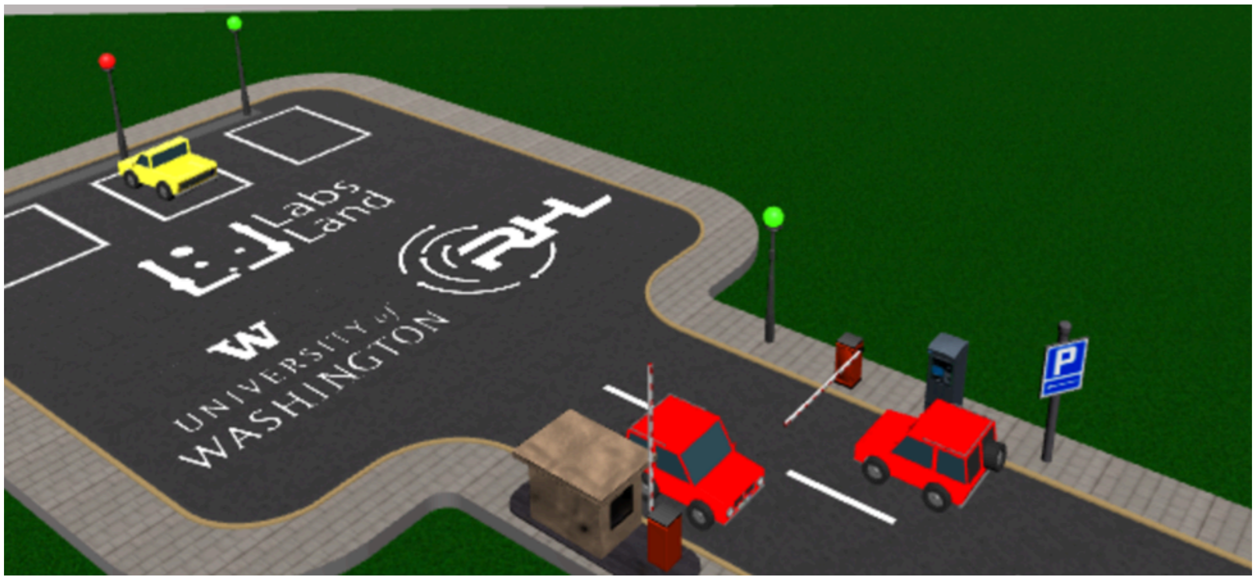


Figure 3: Close-up view of cars entering and exiting the parking lot.

Buttons

The 3D parking lot consists of two buttons: a “Car should arrive” and a “Car should leave”. See Figure 2 for a view of the buttons.

Entering

Pressing the “Car should arrive” button will automatically generate a car to wait at the parking lot barrier entrance.

Entrance Barrier

The barrier can be opened by asserting the appropriate V_GPIO connection to the device under test. When the barrier opens, the car waiting will automatically drive to the nearest available parking slot.

Exiting

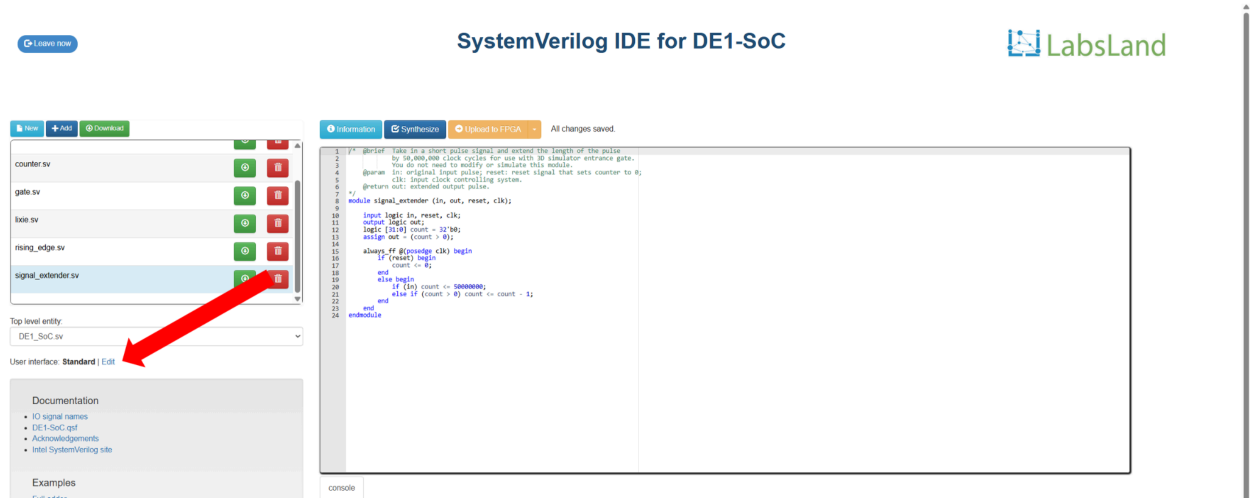
When pressing the “Car should leave” button on the 3D parking lot simulator, one car parked in the parking lot (if any) will attempt to leave by driving to and waiting at the exit gate.

Exit Barrier

The exit barrier can be opened by asserting the appropriate V_GPIO connection to the device under test. When the exit barrier opens, the waiting car will automatically leave the parking lot.

Digital Twin Interface

To select the Parking simulation as our user interface, go to the FPGA remote lab and click Edit next to the User interface, as indicated by the red arrow in Figure 4.

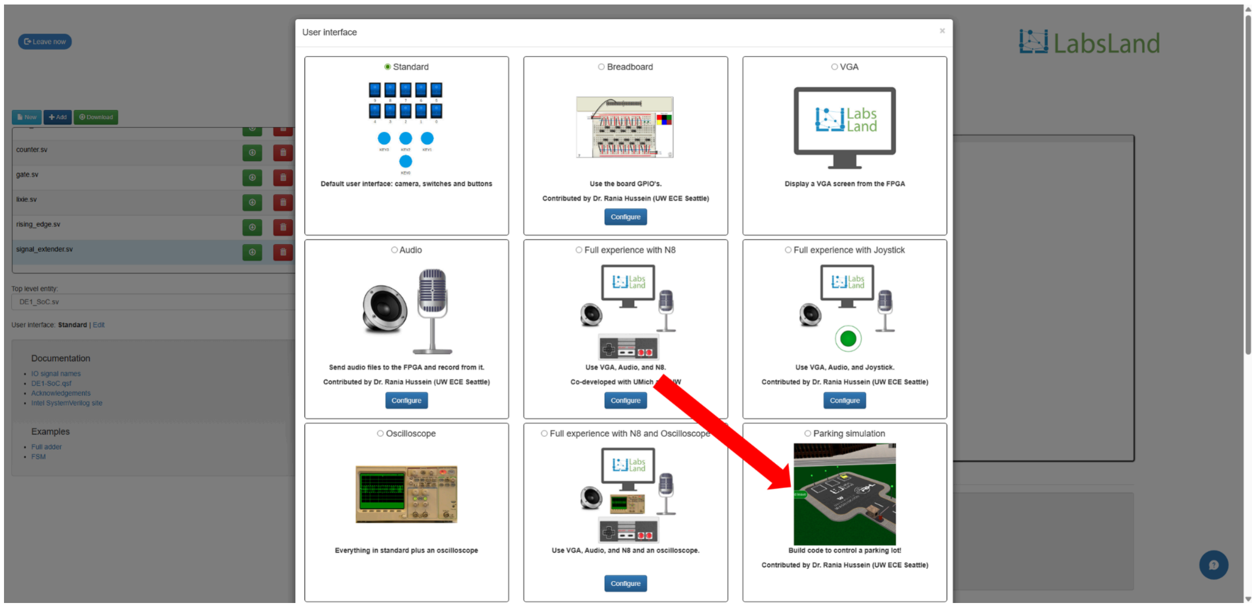


```
/*  @brief  Take in a short pulse signal and extend the length of the pulse
            by 50,000,000 clock cycles for use with 3D simulator entrance gate.
            You do not need to modify or simulate this module.
    @param  in: original input pulse; reset: reset signal that sets counter to 0;
            clk: input clock controlling system.
    @return out: extended output pulse.
*/
module signal_extender (in, out, reset, clk);

    input logic in, reset, clk;
    output logic out;
    logic [31:0] count = 32'b0;
    assign out = (count > 0);

    always_ff @(posedge clk) begin
        if (reset) begin
            count <= 0;
        end
        else begin
            if (in) count <= 50000000;
            else if (count > 0) count <= count - 1;
        end
    end
endmodule
```

Figure 4

License

Next, locate the Parking simulation and click on it, as indicated by the red arrow in Figure 5.



Default user interface: camera, switches and buttons

Use the board GPIO's.

Contributed by Dr. Rania Hussein (UW ECE Seattle)

Display a VGA screen from the FPGA

Send audio files to the FPGA and record from it.

Contributed by Dr. Rania Hussein (UW ECE Seattle)

Use VGA, Audio, and N8.

Co-developed with UMich and UW

Use VGA, Audio, and Joystick.

Contributed by Dr. Rania Hussein (UW ECE Seattle)

Everything in standard plus an oscilloscope

Use VGA, Audio, and N8 and an oscilloscope.

Build code to control a parking lot!

Contributed by Dr. Rania Hussein (UW ECE Seattle)

Figure 5

Exit the user interface selection window by clicking the X button on the upper right of the user interface selection window or anywhere outside of the user interface selection window.

Note: Because there is a lot of 3D data to load, it may take some time to load the parking simulation. The loading speed varies with network quality.

4. GPIO Mappings

4.1 Inputs

Signal Name	***Description***	***V_GPIO***
Presence Parking 1	Indicate if there is a car in parking spot 1.	28
Presence Parking 2	Indicate if there is a car in parking spot 2.	29
Presence Parking 3	Indicate if there is a car in parking spot 3.	30
Presence at Entrance Gate	Indicate if there is a car waiting at the entrance.	23
Presence at Exit Gate	Indicate if there is a car waiting at the exit.	24

4.2 Outputs

Signal Name	***Description***	***V_GPIO***
LED Parking 1	Change the LED color of spot 1 (0 = green, 1 = red).	26
LED Parking 2	Change the LED color of spot 2 (0 = green, 1 = red).	27
LED Parking 3	Change the LED color of spot 3 (0 = green, 1 = red).	32
LED Full	Change the LED color of the full indicator (0 = green, 1 = red).	34
Open Entrance Gate	Opens the entrance gate when set to 1. The gate will stay open until a car enters the lot.	31
Open Exit Gate	Opens the exit gate when set to 1. The gate will stay open until a car leaves the lot.	33

5. Possible Specifications

This section briefly outlines possible ways to use this simulation to develop a laboratory assignment. Example specifications for this simulation go into further detail.

5.1 Standard Parking Lot

Use V_GPIO signals to manage the parking lot and obey occupancy rules. Read inputs for waiting cars and assert outputs for managing gate controls and LEDs.

5.2 Parking Lot Metrics

Building off of standard parking lot implementation, calculate metrics throughout the operation of the parking lot, including total cars, average number of spots filled throughout duration, peak traffic times, etc.

---

6. Suggested Learning Objectives

This section outlines possible learning objectives/topics students could demonstrate using this simulation.

Broad Learning Objectives

- Design a hardware system to meet specific system requirements
- Implement and debug a synchronous digital control system
- Interface with an external device using a defined communication protocol

Possible Hardware and Specification Specific Objectives

- Counters
- Memory (RAM / ROM)
- Finite State Machines (FSM)
- Algorithmic State Machine and Datapath (ASMD)
- On-board switches and buttons