Digital System Design with Verilog HDL

LAB1: Introduction to Vivado

Objectives

- Understand the basic concepts of digital modeling and functional verification with Verilog.
- Explore the simulation and verification environment (**Xilinx Vivado**).
- Create simple digital model and verify its functionality using a test-bench.

Procedure

- **Run Vivado** by clicking on its icon on the desktop.
- **Create a project,** make sure to create a new folder for your projects in the **Students** drive.
- **Name your project**, give a good descriptive name for the project.
- **Select RTL project,** RTL stands for Register Transfer Level, which is the name given for this type of modeling digital logic based on the idea that it describes the transfer of data between registers.
- **Create Verilog files,** create the file buzzer and mux2, make sure that the Target language is Verilog and the Simulator language is either mixed or Veirlog.
- **Select a device,** even though we will not be implementing on a particular device in this lab, however, due to some license limitations in some devices please: **XC7a35tcpg236-1**.
- **Click finish**.
- **Insert input and output ports,** Vivado automatically creates the module name and ports for you, all you need to do is insert the name and width of each port.
- **Edit the module buzzer.v**, write the description of the logic of the circuit seen below:

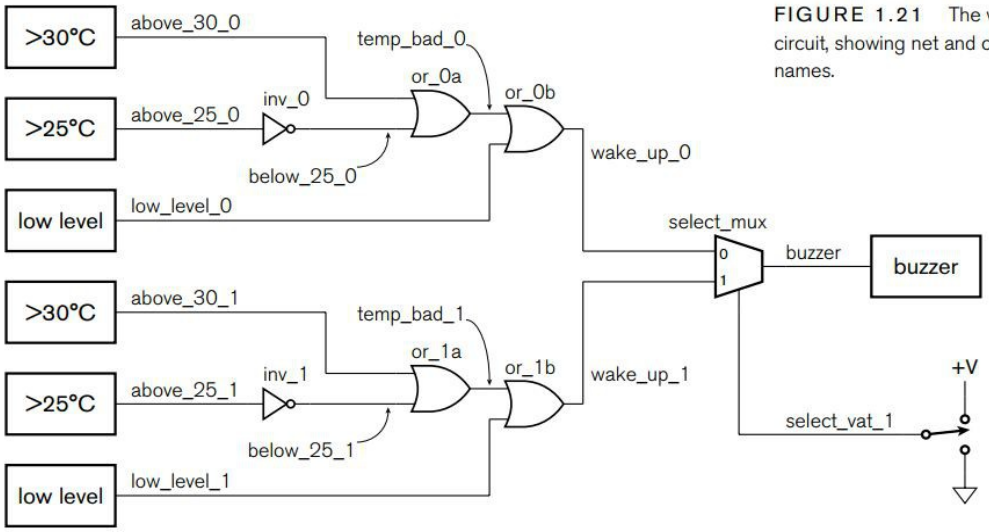


FIGURE 1.21 The vat buzzer circuit, showing net and component names.

The complete code for the above logic is as shown below:

```
module buzzer_controller
  ( output buzzer,
    input above_25_0, above_30_0, low_level_0,
    input above_25_1, above_30_1, low_level_1,
    input select_vat_1 );

  wire below_25_0, temp_bad_0, wake_up_0;
  wire below_25_1, temp_bad_1, wake_up_1;

  // components for vat 0
  not inv_0 (below_25_0, above_25_0);
  or  or_0a (temp_bad_0, above_30_0, below_25_0);
  or  or_0b (wake_up_0, temp_bad_0, low_level_0);

  // components for vat 1
  not inv_1 (below_25_1, above_25_1);
  or  or_1a (temp_bad_1, above_30_1, below_25_1);
  or  or_1b (wake_up_1, temp_bad_1, low_level_1);

  mux2 select_mux (buzzer, wake_up_0, wake_up_1, select_vat_1);

endmodule
```

- Edit the module mux2.v, in the case of the multiplexer we will be describing its **behavior** instead of its **structure**, write the following code for the multiplexer:

```
module mux2 (output y, input i0, i1, sel1);
  assign y = sel1 ? i1 : i0;
endmodule
```

- Now it is time to test the functionality of our system using a test-bench, create a test-bench by:
  - Right clicking on the sources area.
  - Select Add sources.

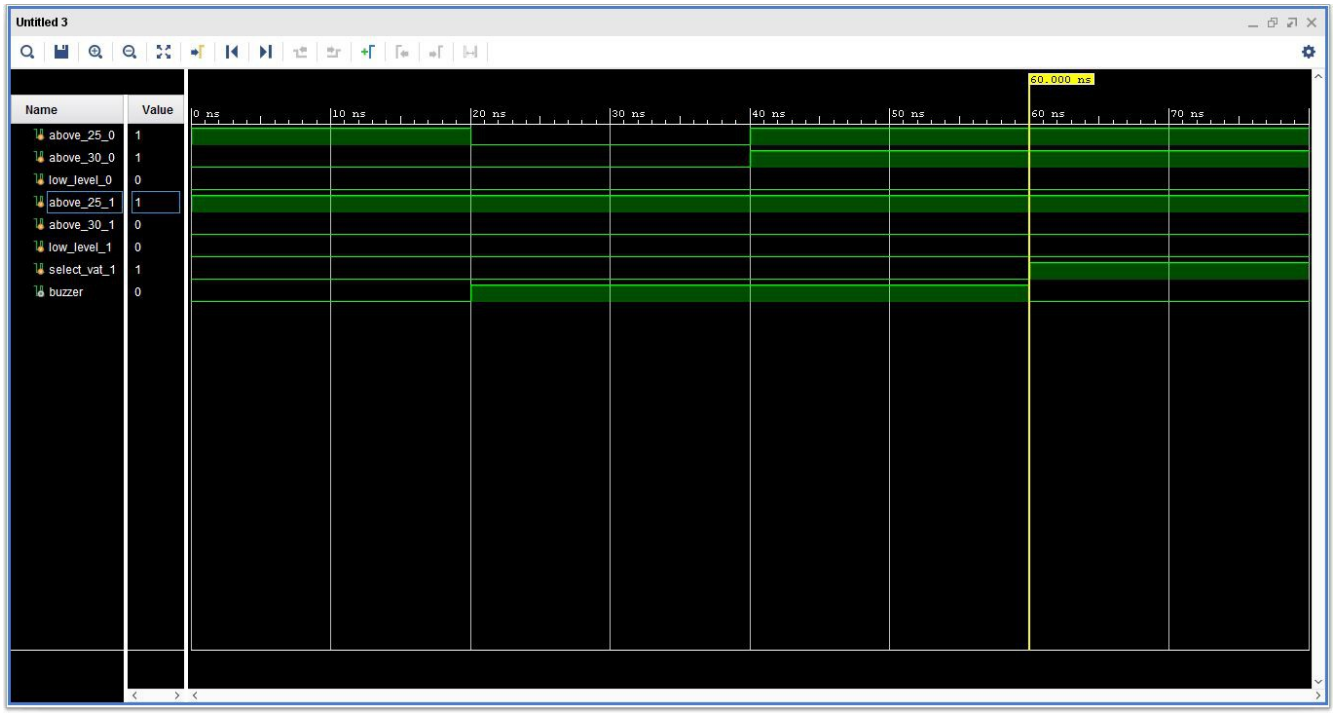
- Select Add or create simulation sources.
- Create file.
- Name it test_bench.
- Click Finish.
- Do not add input or output ports to the test-bench.
- Write the following code for the test-bench.

```
module test_bench(

  );
  reg above_25_0, above_30_0, low_level_0;
  reg above_25_1, above_30_1, low_level_1;
  reg select_vat_1;
  wire buzzer;

  buzzer_controller buzzer_1(buzzer,
    above_25_0, above_30_0, low_level_0,
    above_25_1, above_30_1, low_level_1,
    select_vat_1);

    initial
      begin
      above_25_0 = 1'b1;
      above_30_0 = 1'b0;
      low_level_0 = 1'b0;
      above_25_1 = 1'b1;
      above_30_1 = 1'b0;
      low_level_1 = 1'b0;
      select_vat_1 = 1'b0;

      #20 above_25_0 = 1'b0;

      #20 above_30_0 = 1'b1;
        above_25_0 = 1'b0;
```

```
        #20 select_vat_1 = 1'b1;
        #20 $finish;
        end

endmodule
```

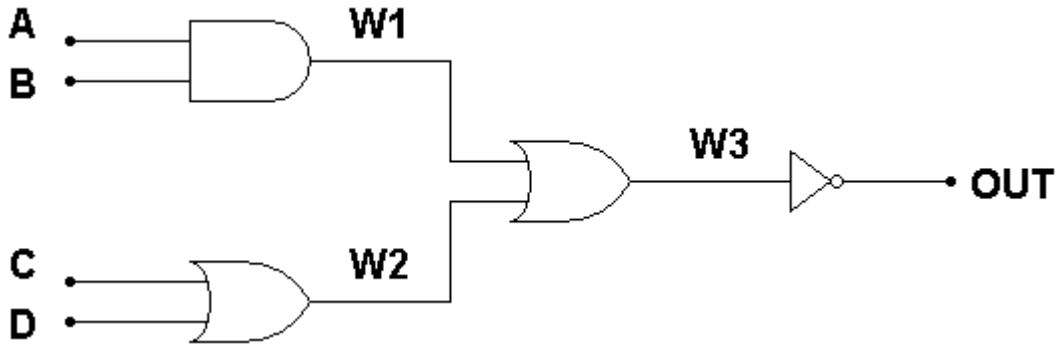
The above test-bench describes a scenario to verify the logic functionality, this scenario is the following:

- At the beginning, all sensors were having normal behavior.
- After 20ns the input **above_25_0** changed its value to 0, which lead to the buzzer be activated.
- After another 20ns **above_25_0** changed its value to 1 again but **above_30_0** changed to 1, this made the buzzer continue to be activated.
- After another 20ns **select_vat_1** was changed to 1, this lead to the de-activation of the buzzer due to the selection of the other vat.
- After 20ns the simulation finished.



Assignment

1. Write the Verilog module that describes this circuit.



2. Devise a circuit for a simple burglar alarm that activates a siren if either a motion sensor detects motion or a sensor on a window detects that the window is open, Then write a Verilog module for it and verify it functionality with a test-bench and simulate it with Vivado.
3. Re-write mux2.v using a structural Verilog module (i.e. using gates) instead of the behavioral module used above.