



Lab 1

Introduction to Vivado Design Suite and Basic Gate-Level Modeling in SystemVerilog

Module ID : CX-203

Digital System Design

Instructor : Dr. Abid Rafique

Version 1.2

*Information contained within this document is for the sole readership of the recipient,
without authorization of distribution to individuals and / or corporations without prior
notification and approval.*

Document History

The changes and versions of the document are outlined below:

| Version | State / Changes | Date | Author |
|---------|-----------------------------|--------------|----------------------------------|
| 1.0 | Initial Draft | Dec, 2023 | Qamar Moavia |
| 1.1 | Modified with new exercises | August, 2024 | Dr. Abid Rafique Qamar Moavia |
| | | | |
| | | | |
| | | | |

Table of Contents

| | |
|---|----|
| Objectives | 4 |
| Deliverables | 4 |
| 1. Intro to System Verilog | 5 |
| 2. Creating Project in Vivado | 6 |
| Step 1: Open Vivado and Start a New Project | 6 |
| Step 2: Specify Project Type | 6 |
| Step 3: Select the Board (Arty A7-100T) | 6 |
| Step 4: Confirm and Finish | 6 |
| Step 5: Add Sources (Can be used in Later Tasks) | 7 |
| Step 6: Optional Settings (Will be used in Later Labs) | 7 |
| Step 7: Generate Bitstream and Program (For Hardware Deployment) | 7 |
| 3. Creating and Simulating a 2-bit andgate in Vivado | 8 |
| Step 1: Add a New Source File: | 8 |
| Step 2: Write the Code for Basic Gates: | 8 |
| Step 3: Creating a Testbench for Simulation | 8 |
| Step 4: Running the Simulation | 11 |
| 4. Creating and Simulating a 4-input and gate using 2-input and gates | 13 |
| TASK 1 : 4-input andgate Simulation | 14 |
| TASK 2 : Half Adder | 14 |
| TASK 3 : Full Adder | 15 |

Objectives

By the end of this lab, students will:

- Learn to set up a project in Vivado, add SystemVerilog source files, and run simulations.
- Understand and model basic logic gates (AND, OR, NOT) using SystemVerilog.
- Understand what a testbench is? How do testbenches test designs?
- Develop a testbench to verify the functionality of design.

Deliverables

A single PDF document containing the following:

- Source code (code or screenshots)
- Simulation results (waveforms) screenshots
- Link to git repository

1. Intro to System Verilog

In this section, you'll learn how to model basic digital logic circuits using SystemVerilog.

SystemVerilog is a hardware description and verification language (HDVL) that is widely used in the design and verification of digital systems, such as processors, FPGAs, and ASICs. It extends Verilog, a traditional hardware description language, by adding features to support both design and verification.

In this lab, we will focus on **gate-level modeling**, which is one of the key features of SystemVerilog used to describe digital circuits at the logic gate level. This allows us to model circuits using basic logic gates like AND, OR, and NOT.

Here is the basic SystemVerilog code to model AND gate using gate-level modeling.

```
module andgate (  
    input logic a,  
    input logic b,  
    output logic f  
);  
  
    and u_and(f, a, b); // AND gate  
endmodule
```

Similarly we can model OR, NOT and other gates as well using gate-level modeling.

2. Creating Project in Vivado

Here's a complete step-by-step guide to create a project in Vivado for the Arty A7-100T board:

Step 1: Open Vivado and Start a New Project

- Launch Vivado: Open the Vivado Design Suite.
- Create New Project: Click on "Create Project" on the start page or go to File > Project > New.
- Project Name:
 - Enter a name for your project (e.g., Lab1_Project).
 - Choose a project location where your project files will be stored.
 - Ensure "Create project subdirectory" is checked.
 - Click Next.

Step 2: Specify Project Type

- Project Type:
 - Select RTL Project.
 - Check "Do not specify sources at this time" (you can add them later).
 - Click Next.

Step 3: Select the Board (Arty A7-100T)

- Board Selection:
 - In the "Default Part" step, switch to the "Boards" tab.
 - Search for Board: In the search bar, type "**Arty A7-100**".
 - Select this part **Arty A7-100**.
 - Click Next.

Step 4: Confirm and Finish

- Review Settings: The final page will display a summary of your project settings.
- Click Finish: Complete the project setup.

Step 5: Add Sources (Can be used in Later Tasks)

- Add Design Files: Once the project is created, you can add your Verilog/SystemVerilog files or IP blocks by clicking on "Add Sources" in the Flow Navigator.
- Synthesize/Simulate: Use the Flow Navigator to run synthesis, simulation, and implementation once your design is complete.

Step 6: Optional Settings (Will be used in Later Labs)

- Assign Constraints: If you're working with I/O pins (e.g., switches, LEDs, buttons), you'll need to add the constraints file (XDC file) specific to the Arty A7-100T board.
 - You can download the constraints file from Digilent's website or Vivado's board repository.

Step 7: Generate Bitstream and Program (For Hardware Deployment, will be used in Later Labs)

Once your design is synthesized and implemented, you can generate the bitstream and program your board using the **"Program and Debug"** section.

3. Creating and Simulating a 2-bit andgate in Vivado

Step 1: Add a New Source File:

- In the **Flow Navigator** on the **left**, click on Add Sources under Project Manager.
- Select Add or Create Design Sources and click Next.
- Click on Create File.
- Choose File type as **SystemVerilog**.
- Name the file **andgate** and click OK.
- In the **Define Module** window click OK.

Step 2: Write the Code for Basic Gates:

Open `andgate.sv` and copy the following code into that file:

```
module andgate (  
    input logic a,  
    input logic b,  
    output logic f  
);  
  
    and u_and(f, a, b); // AND gate  
endmodule
```

You can use or, xor, not etc as well in the gate level modeling.

Step 3: Creating a Testbench for Simulation

Now that you've created the **andgate.sv** file, it's time to simulate the design to see if it functions as expected. To do this, you'll need to create a testbench.

- **Add a New Simulation Source:**
 - In the Flow Navigator, click on Add Sources under Project Manager.
 - Select Add or Create Simulation Sources and click Next.
 - Click on Create File.
 - Choose File type as **SystemVerilog**.
 - Name the file **tb_andgate** and click OK.
 - In the **Define Module** window click OK.

- **Write the Testbench Code:**

Before writing testbench code, let's discuss what a testbench actually is. A testbench is a SystemVerilog module used to test or verify a design. For example, a testbench for an AND gate will verify whether the design functions as expected.

Next, the question arises: how do we verify the design? The answer is simple—ensure the design behaves correctly for different inputs. For a 2-input AND gate, we need to confirm its behavior for all four possible input combinations. These combinations and the expected outputs are shown in the table below:

| a | b | f |
|----------|----------|----------|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

You will drive these input combinations of a and b to the AND gate and check the output for each case.

So, now let's get back to the vivado, open the `tb_andgate.sv` and copy the following testbench code into that file:

```
module tb_andgate;
    // Declare testbench signals
    logic a, b;
    logic f;
    // Instantiate the andgate module
    andgate dut (
        .a(a),
        .b(b),
        .f(f)
    );

    // Testbench logic
    initial begin
        // Display header
        $display("Time\t a\t b\t f");
        $display("-----");

        // Apply test vectors with $display statements
        a = 0; b = 0; #10;
        // Display the value of a, b and f along with time
        $display("%0t\t %b\t %b\t %b", $time, a, b, f);

        a = 0; b = 1; #10;
        $display("%0t\t %b\t %b\t %b", $time, a, b, f);

        a = 1; b = 0; #10;
        $display("%0t\t %b\t %b\t %b", $time, a, b, f);

        a = 1; b = 1; #10;
        $display("%0t\t %b\t %b\t %b", $time, a, b, f);

        // End simulation
        $finish;
    end
endmodule
```


This testbench will apply (drive) different combinations of inputs a and b to the andgate module. After driving any combination test bench display's the response of the design under test (andgate) for that particular combination.

Step 4: Running the Simulation

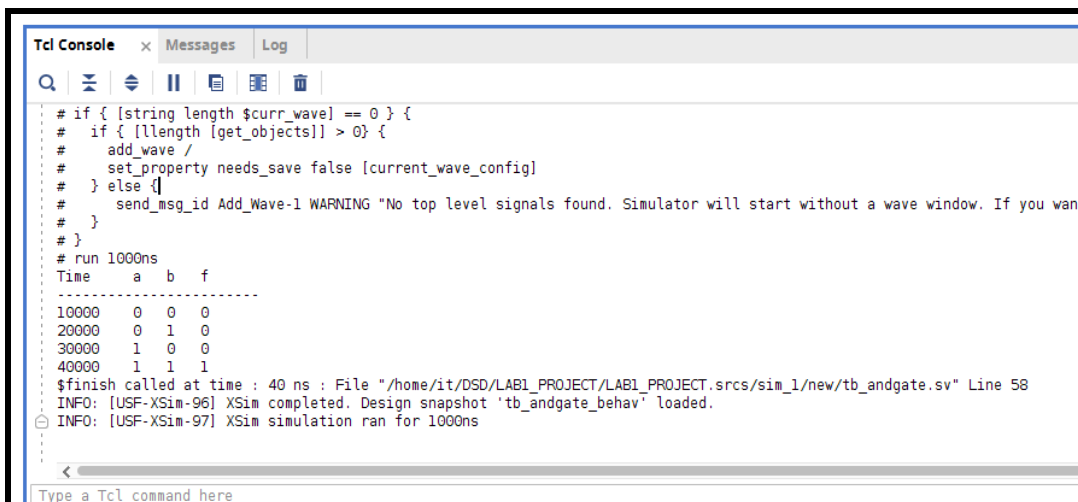
Once you've created the testbench, you can proceed to simulate the design and view the waveforms.

- **Run Simulation:**
 - In the Flow Navigator, click on Run Simulation under Simulation.
 - Select Run Behavioral Simulation.
 - Vivado will now compile your design and launch the simulator.
- **View the output :**

Once the simulation starts, the **Tcl Console** will display the **\$display** outputs in real time. By default, Vivado runs the simulation for a short period and pauses; click the **Run** icon to continue the simulation.

The **Run** icon is located in the **Simulation** toolbar at the top of the **Waveform** window in Vivado. It looks like a blue play button .

Verify that the **f** signal displayed on the Tcl Console is exactly an **and** operation between **a** and **b**.



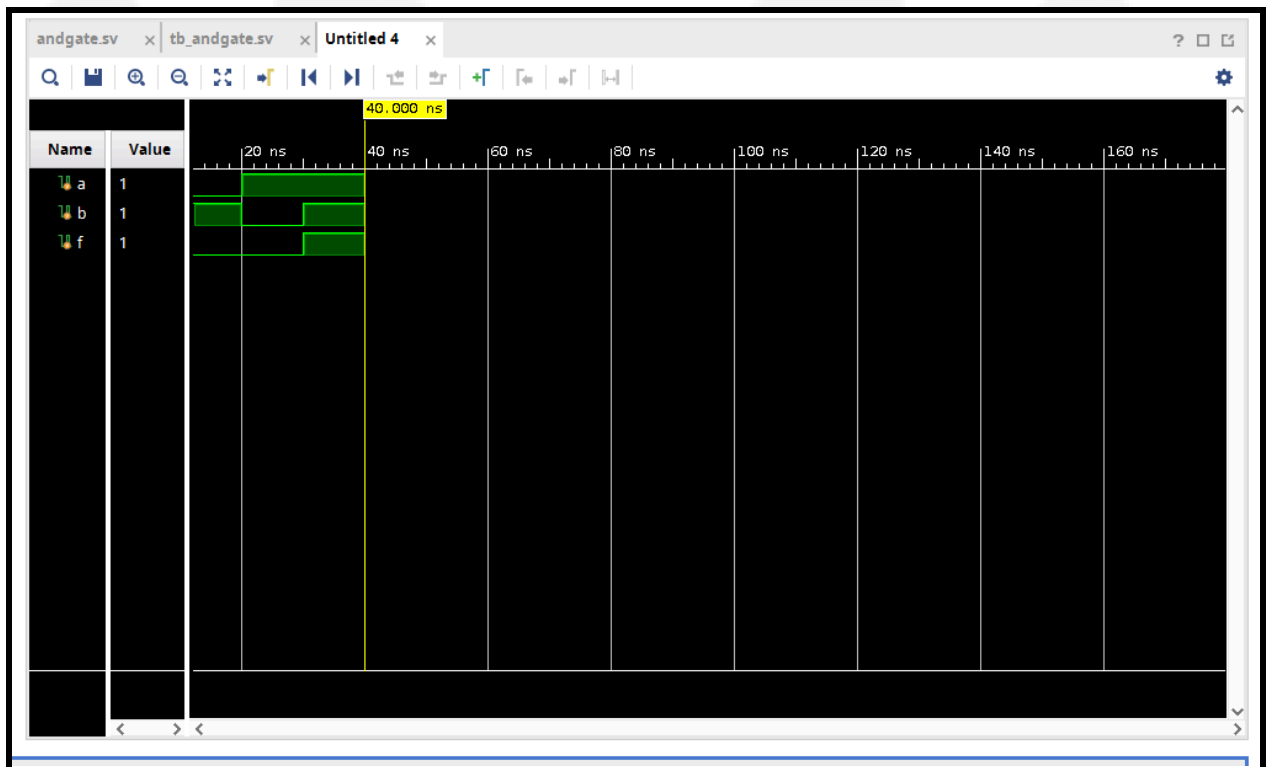
```

Tcl Console x Messages Log
[Icons]
# if { [string length $curr_wave] == 0 } {
#   if { [length [get_objects]] > 0 } {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#     send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want
#   }
# }
# run 1000ns
Time      a      b      f
-----
10000     0      0      0
20000     0      1      0
30000     1      0      0
40000     1      1      1
$finish called at time : 40 ns : File "/home/it/DSD/LAB1_PROJECT/LAB1_PROJECT.srscs/sim_1/new/tb_andgate.sv" Line 58
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_andgate_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
Type a Tcl command here
  
```

- **View the Waveform:**

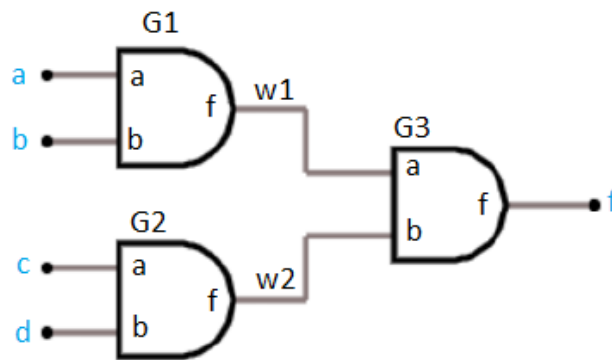
We sometimes use the testbench to drive different combinations of input to the design and use waveform to check that the design's output is exactly as expected.

- Once the simulation is running, you should see the **Waveform** window open automatically.
- If you don't see the waveform window, go to **Window > Waveform**.
- The waveform will display the signals **a**, **b**, and **f** over time.
- Zoom in and out using the toolbar buttons to inspect specific portions of the waveform.
- Verify that the **f** signal on the waveform is exactly an **and** operation between **a** and **b**.



4. Creating and Simulating a 4-input and gate using 2-input and gates

In this part, you will learn how to construct a 4-input AND gate using 2-input AND gates. In the working example of AND gate you saw that AND gate can be constructed by using its gate primitive. To construct a 4-input and gate we will use three instances of 2-input AND gates as shown in the figure 41 below.



Here, we are simply connecting the output port of one gate to the input port of another gate. In other words we are driving one module's output as another module's input. The complete code for 4-input and gate using 2-input and gates is shown below.

```

module and4gate(
    input logic a,
    input logic b,
    input logic c,
    input logic d,
    output logic f
);

    logic w1, w2;

    andgate G1( .f(w1) , .a(a) , .b(b) );
    andgate G2( .f(w2) , .a(c) , .b(d) );
    andgate G3( .f(f) , .a(w1) , .b(w2) );

endmodule
  
```

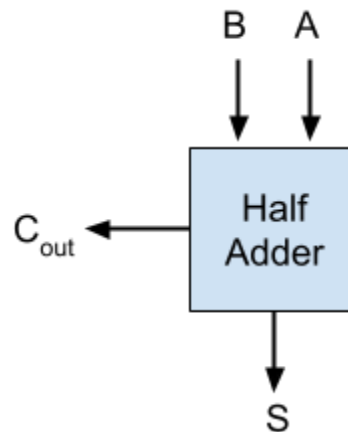
TASK 1: 4-input andgate Simulation

Write a testbench to verify the 4-input AND gate discussed above. Ensure all possible input combinations are applied to the design under test (DUT). Confirm that the DUT (4-input AND gate) behaves as expected, performing the AND operation across the four input bits.

TASK 2: Half Adder

Create a Half Adder design. Write SystemVerilog code to describe half adder using gate level modeling. Verify its operation by writing a testbench and checking the outputs through waveform.

| A | B | S | C _{out} |
|---|---|---|------------------|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |



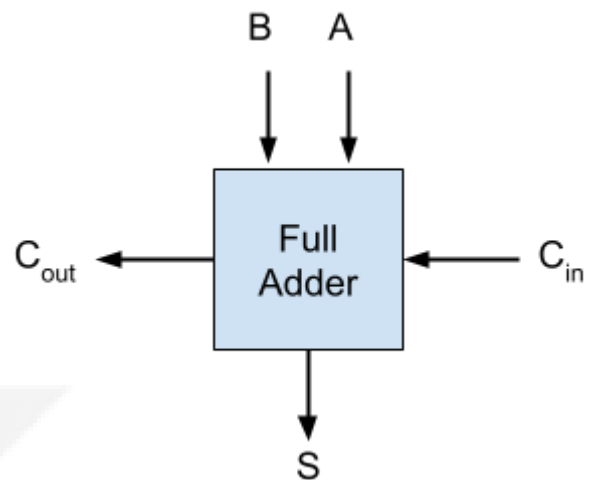
TASK 3 : Full Adder

Design and simulate a **Full Adder using half adders** from TASK 2.

Note: It's mandatory to use half adders to create Full Adder. Don't just directly create a full adder.

You can use this truth table as a reference in the simulation to verify your Full Adder.

| A | B | C_{in} | S | C_{out} |
|---|---|----------|---|-----------|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |



Good Luck 😊