COL215
Digital Logic & System Design

Assignment 1

Submitted by:
Dhairya Kuchhal (2024CS50396)
Siddhant Agrawal (2024CS50469)

August 6, 2025

1 Task

In this lab, we implemented three fundamental logic gates — AND, OR, and NOT — using Verilog and deployed them on the Basys 3 FPGA board. The task involved three major steps:

- Set up the AMD Vivado Design Suite on our allotted computer in the DHD Lab.
- Implement and simulate the three basic combinational gates: AND, OR, and NOT using Verilog.
- Synthesize our code and upload the generated bitstream to the Basys-3 FPGA board. Test its functionality.

2 Design Decisions

We kept a minimal design for correctness and clarity. After successfully deploying the AND gate, the logic for the OR and NOT gates was included in the same design file (‘AND_gate.v‘) and testbench file (‘AND_gate_tb.v‘). Input pins were chosen in the order NOT–OR–AND, adjacent to each other from left to right. Output pins followed the same pattern for clarity.

2.1 Base Module Structure

We implemented a single module defining the inputs and outputs for all three gates, as outlined below.

```
module AND_gate (
    input a,    // a and b are inputs for AND gate
    input b,
    input d,    // d and e are inputs for OR gate
    input e,
    output f,   // f is output for OR gate
    input g,    // g is input in NOT gate
    output c,   // c is output for AND gate
    output h    // h is output for NOT gate
);
    assign c = a & b;  // AND gate
    assign f = d | e;  // OR gate
    assign h = ~g;     // NOT gate
endmodule
```

2.2 Simulation using Testbench

We wrote a dedicated testbench module (AND_gate_tb.v) for simulation. Inputs were toggled at 10ns time intervals (Inertial delay). Changes reflected after the program delay time of 40ns. The goal was to:

- Verify logic correctness before synthesis.

- Observe transitions in a waveform simulator.

The code is as follows:

```
module AND_gate_tb();
// c,f, h are wires since their value is changing and unstored
    reg a,b,d,e,g;
    wire c,f,h;
// links the input variables used in design module with those in basys3.xdc file
    AND_gate UUT(
        .a (a),
        .b (b),
        .c (c),
        .d (d),
        .e (e),
        .f (f),
        .g (g),
        .h (h)
    );
// inputs were toggled at regular intervals to check reposnse from output variables.
    initial begin
    a=0;
    b=0;
    d=0;
    e=0;
    g=0;
    #10 a=1; d=1; g = 1;
    #10 b=0; a=0; g = 0; e = 1;
    #10 a=1; b=1; d=0;
    #10 $finish;
    end
endmodule
```

2.3 Mapping of Pins on Basys3

The constraint file (basys3.xdc) was edited to bind logical inputs and outputs to physical switches and LEDs. The following table summarizes the mapping between logical signals and FPGA pins:

Signal Type	Logical Name	FPGA Pin
Input	a	V17
Input	b	V16
Input	d	W16
Input	e	W17
Input	g	W15
Output	c	U16
Output	f	E19
Output	h	U19

Table 1: Mapping of logical inputs and outputs to Basys 3 FPGA pins

This mapping ensured that all gates could operate simultaneously without interference or discrepancies.

The following image shows position of mentioned pins on the board. Red represents the Not gate, Orange the OR gate, and Yellow the AND gate. The rectangles enclose the input pins, and the ovals cover the output pins

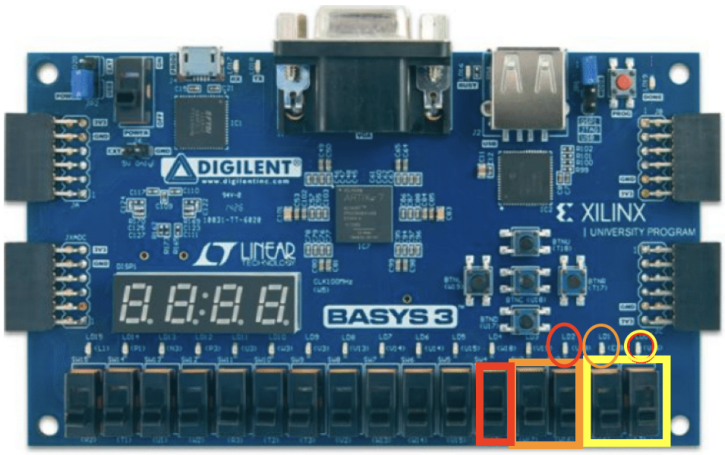

Figure 1: Labelled picture of Basys3 FPGA board

3 Lab Work

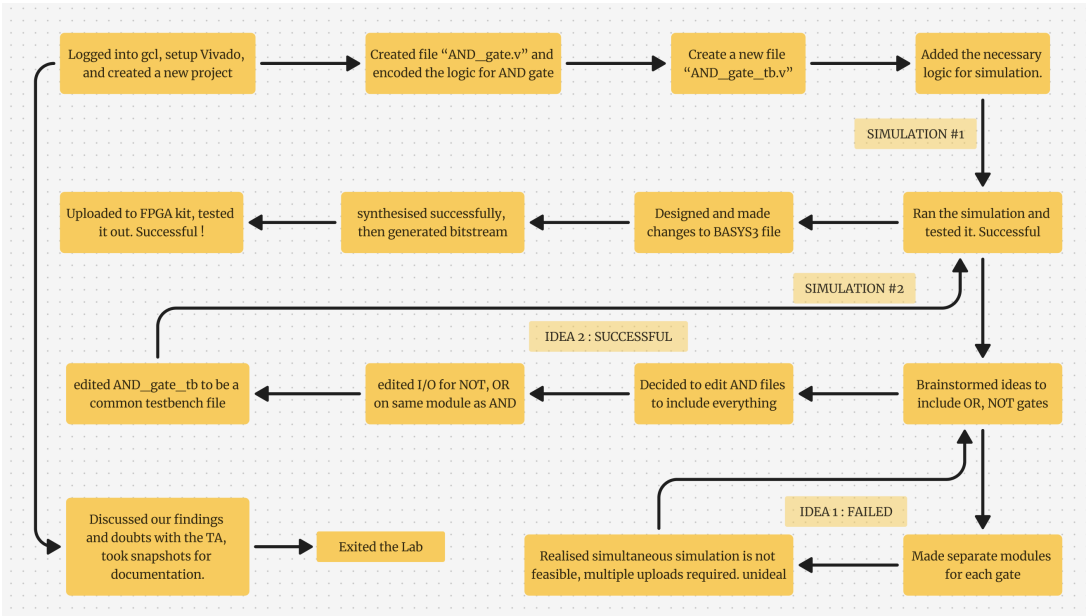

Figure 2: A flowchart showing the steps performed by us in lab in order of occurence.

4 Simulation Snapshots

Figure-1 shows our experiments with the "Force constant" option in simulation section of Vivado after coding all the gates. The top 3 bars represent a,b,c - Input and Output of the AND gate. The next 3 represent the input and output of OR gate, and the last are indicative of the NOT gate. We also captured the initial variation of output in the AND gate as per the testbench file above in the snapshot shown in Figure-2.

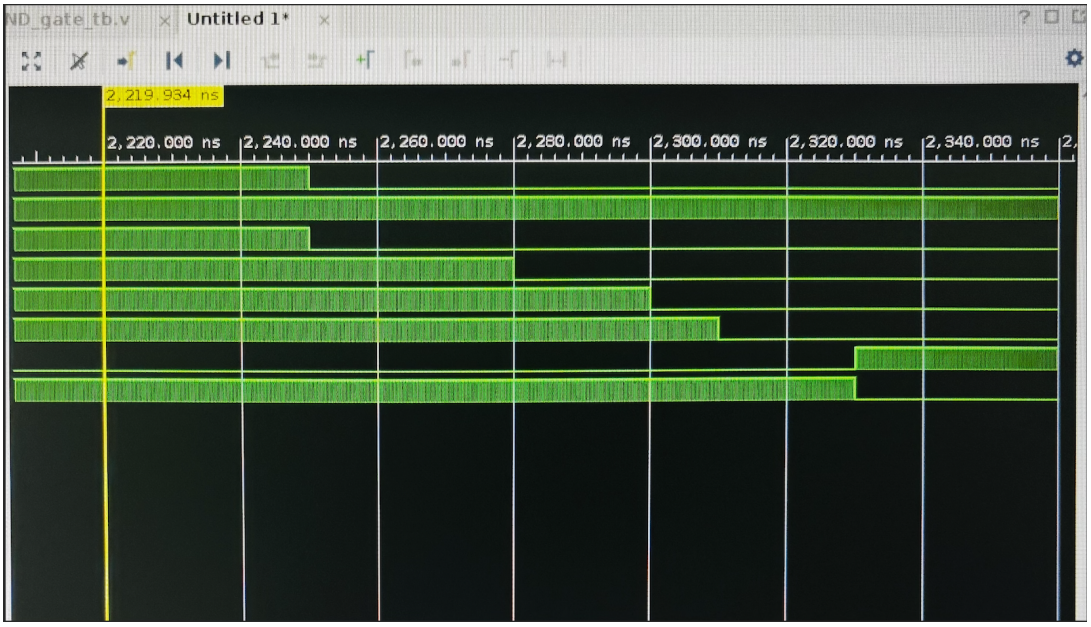

Figure 3: Simulation waveform showing the transitions of AND, OR, and NOT gate outputs on varying inputs with Force Constant option

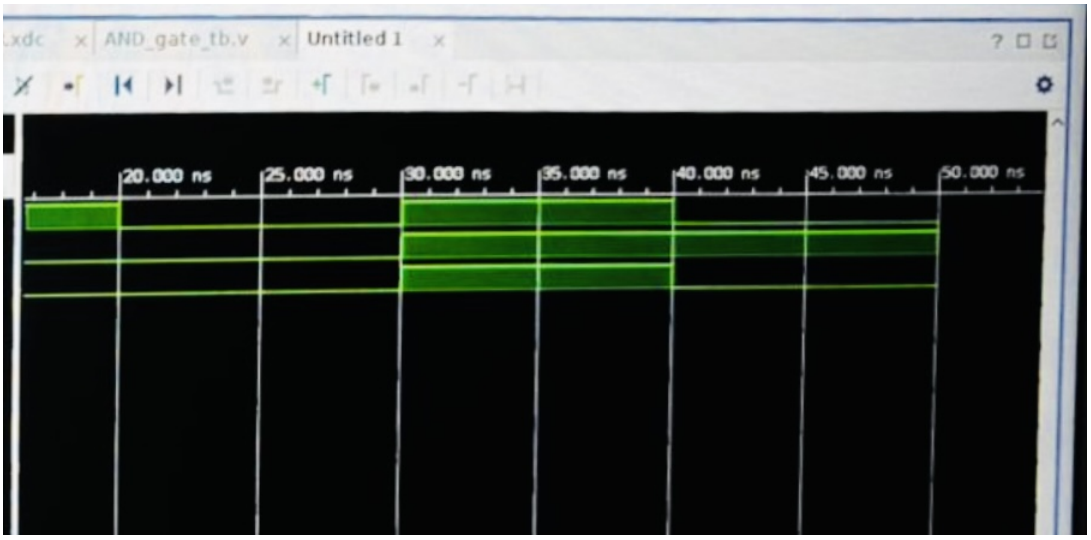

Figure 4: Simulation waveform showing the transitions of AND gate from testbench file

5 Utilization Report and Generated Schematics

Figure - 3 shows the resource utilization during synthesis and the amount of Look Up Tables used by our code. In the ZIP file with all the other files we also attach the Utilization for further inspection if needed.

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	3	0	0	20800	0.01
LUT as Logic	3	0	0	20800	0.01
LUT as Memory	0	0	0	9600	0.00
Slice Registers	0	0	0	41600	0.00
Register as Flip Flop	0	0	0	41600	0.00
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt_design after synthesis, if not already completed, for a more real count.

Figure 5: Resource Utilization during Synthesis

The schematic shown below is the result of synthesizing the AND_gate design using Vivado 2022.1. The visual representation maps the logic design onto the physical FPGA fabric. Each colored block corresponds to a configurable logic block (CLB) on the Basys 3 board. In the bottom panel, the resource utilization report indicates that the design uses only 3 Slice LUTs (Look-Up Tables) along with no Flip Flops, BRAMs or DSPs, reflecting minimal resource usage — as expected from 3 simple logic gates as we see in the utilization report.

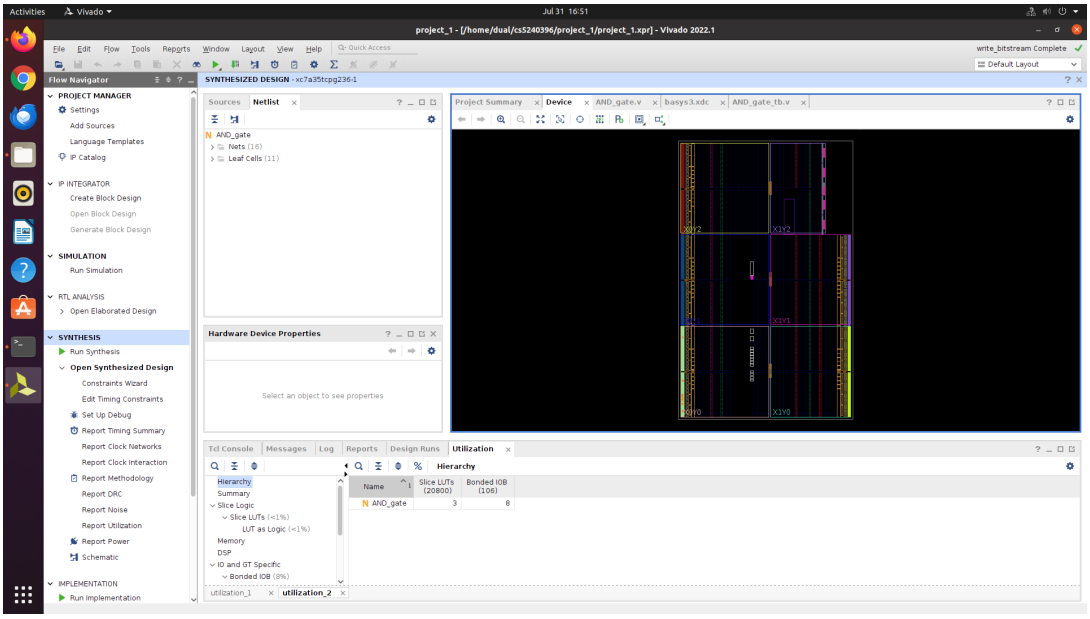

Figure 6: Generated Schematics of the board

6 Conclusion

This lab introduced us to the complete FPGA design workflow using the Vivado Design Suite and Basys 3 development board. By implementing basic logic gates - AND, OR, and NOT we gained experience with Verilog coding, simulation via testbenches, pin mapping using constraint files, and synthesis to hardware deployment, giving us a strong foundation for future assignments

All three gates functioned as expected in simulation and on hardware, demonstrating correct logical behavior and minimal resource usage. The use of a modular design and centralized mapping helped in achieving clarity and scalability in the setup.

By completing this lab, we now have a solid understanding of:

- The Vivado Design Suite environment
- The Verilog based design process.
- Functional simulation using waveform viewers and testbenches.
- The practical use of switches and LEDs for on-board testing.

This foundational lab prepares us well for future experiments involving more complex combinational circuits.