Lab 1
Introduction to Digital Circuit Design
October 23 - October 30, 2024

Cameron Wilding
AK 317

Table of Contents

Abstract

The purpose of this laboratory is to write Verilog code to implement an alarm system using basic logic functions, such as And and Or gates.

Equipment

This experiment was carried out in Atwater Kent room 317 with the following equipment:

- ECE Lab Kit
- Verilog

Introduction

The alarm system we were supposed to design has 3 inputs;

- Door Open Sensor (C)
- Motion Sensor (B)
- Arm(Enable) Alarm (A)

And one output;

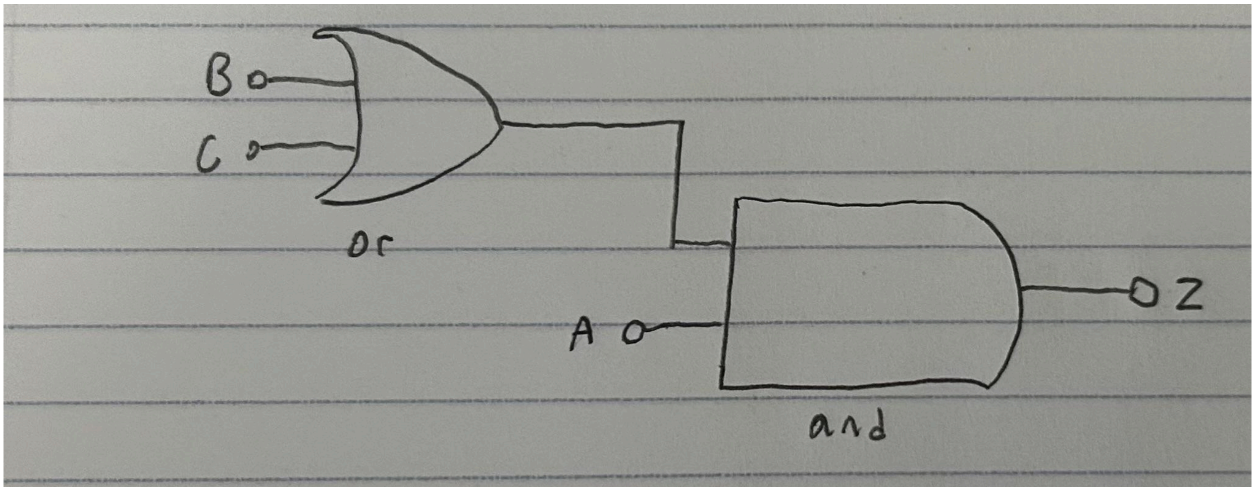
- Alarm (Z)

The boolean expression of the alarm system is: $Z = (B + C) * A$.

Theoretical Basis

Breaking down the Boolean expression into logic gates, we can see that first, C and B are combined together in an Or gate (as they are added in the Boolean expression), and then the output of said Or gate (written as wire X in the code) is put into an And gate with A, resulting in Z.

Circuit Description



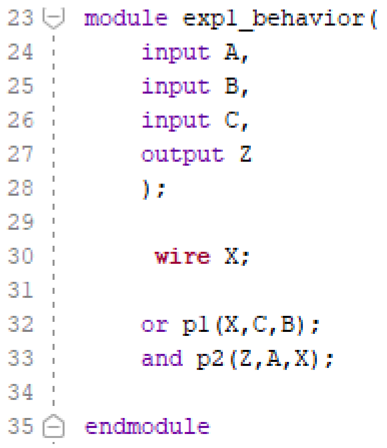
Logic Circuit Drawn

A	B	C	Z
0	X	X	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Truth Table Hypothesis.

Experimental Results

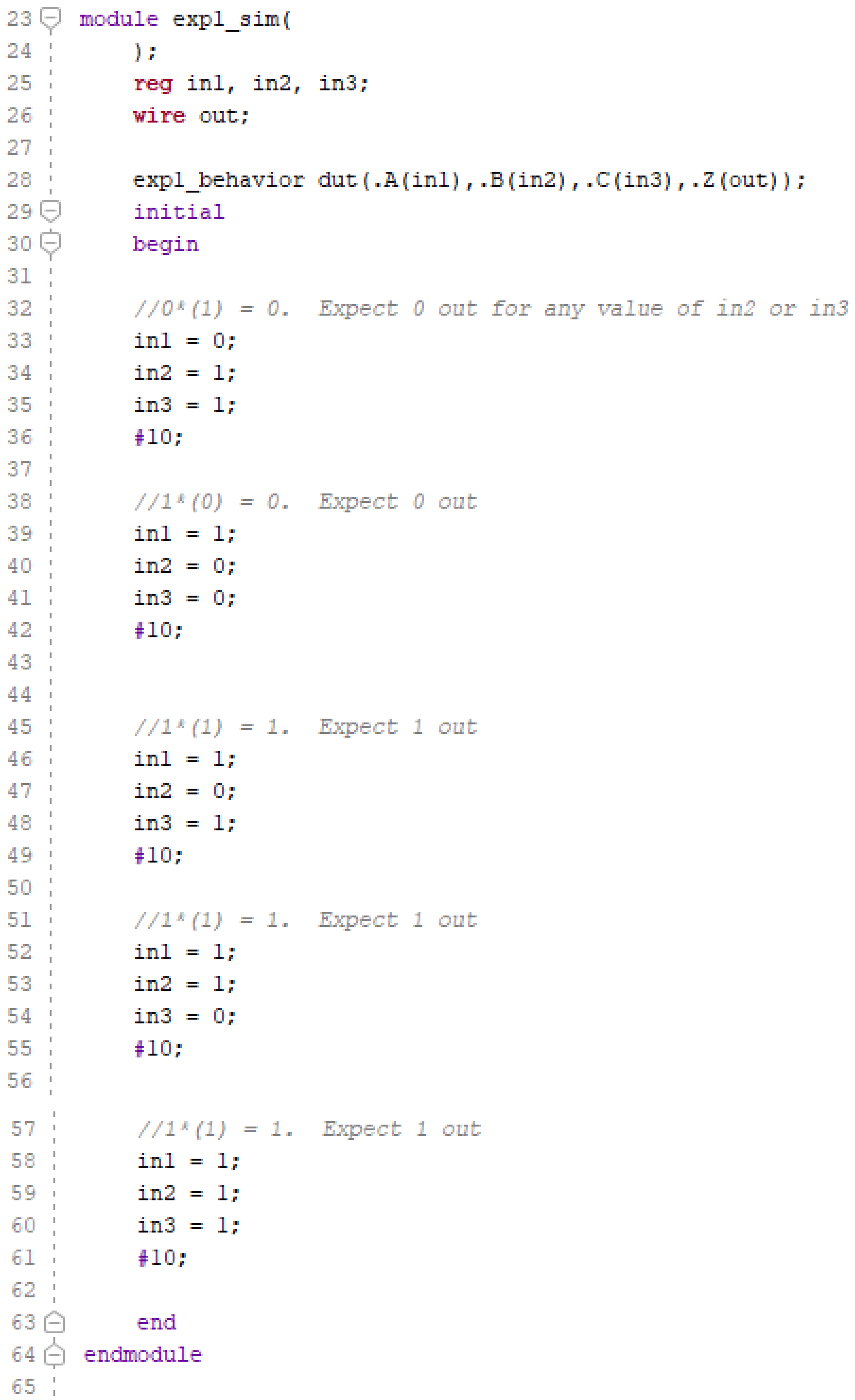
To implement this logic circuit into Verilog, I first had to write the structural code of the logic circuit. This code, as structured in the lab requirements, intakes A, B, and C. Then it puts C and B through an Or gate, with the result being stored as wire X. Then, X and A are run through an And gate, resulting in the final output Z.



```
module expl_behavior(
    input A,
    input B,
    input C,
    output Z
    );

     wire X;

    or p1(X,C,B);
    and p2(Z,A,X);

endmodule
```

Structural Code

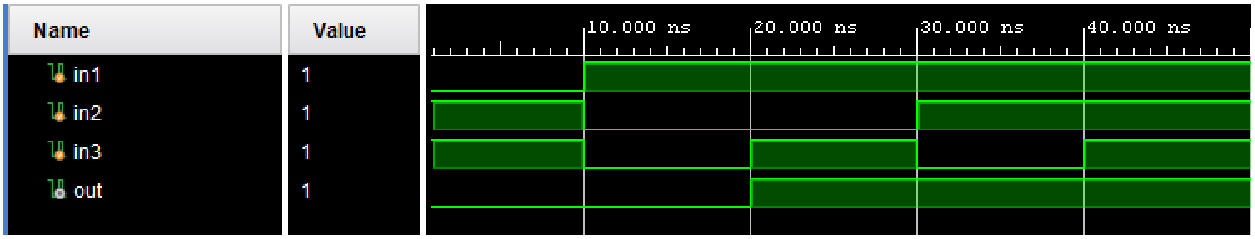
Next, the code that controlled the experimental behavior of this circuit for testing purposes was written. I first defined A, B, and C as in1, in2, and in3 respectively, then assigned A to 0, and B+C to 1, and had the computer hold there for 10 milliseconds. I then did the same for every other possible value of A, B, and C, as listed in the truth table in the previous section, so I could test every iteration of the circuit.



```
module exp1_sim(
    );
    reg in1, in2, in3;
    wire out;

    exp1_behavior dut(.A(in1),.B(in2),.C(in3),.Z(out));
    initial
    begin

    //0*(1) = 0.  Expect 0 out for any value of in2 or in3
    in1 = 0;
    in2 = 1;
    in3 = 1;
    #10;

    //1*(0) = 0.  Expect 0 out
    in1 = 1;
    in2 = 0;
    in3 = 0;
    #10;


    //1*(1) = 1.  Expect 1 out
    in1 = 1;
    in2 = 0;
    in3 = 1;
    #10;

    //1*(1) = 1.  Expect 1 out
    in1 = 1;
    in2 = 1;
    in3 = 0;
    #10;

    //1*(1) = 1.  Expect 1 out
    in1 = 1;
    in2 = 1;
    in3 = 1;
    #10;

    end
endmodule

```

Behavioral Code

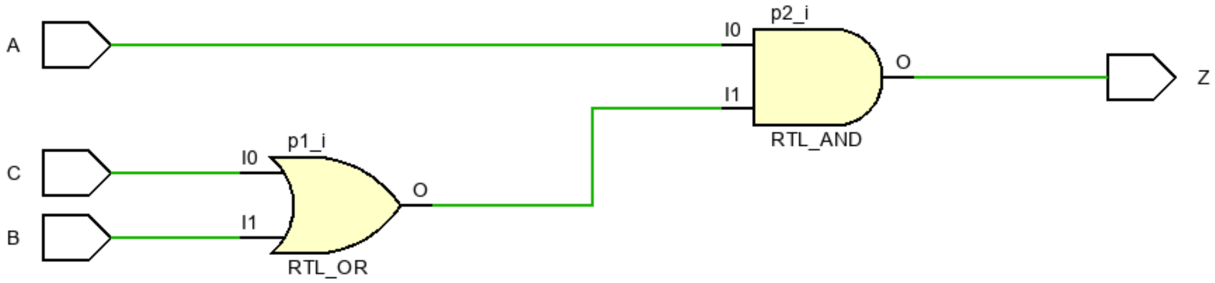
Finally, I ran all of this code through the Verilog simulator, and ended up with this graph as a result, where again, in1 = A, in2 = B, in3 = C, and out = Z.



Name	Value
in1	1
in2	1
in3	1
out	1

Sim Result

Verilog also output a RTL design of the equivalent circuit, seen below



RTL Design

Comparison to Theory & Simulation

Looking at the sim result, we can see that the truth table was exactly correct. As expected, when A = 0, then Z = 0. When both B and C equal 0, then Z =0. Finally, whenever A = 1, and at least 1 of B and C equal 1, then Z = 1. This shows that the truth table is entirely correct. The RTL design, as expected, is exactly the same as the logic circuit I drew at the beginning, ensuring I wrote my code correctly.

Observations and Conclusions

The results of this lab show that for the specified alarm system to go off, the alarm needs to be on, and at least one of the motion sensor or the door open sensor needs to be tripped. There were no discrepancies between the theory and the experimental result.

Overall Summary

In this lab, I took a boolean expression of an alarm system, wrote a truth table and drew a logic circuit for said alarm system, then integrated them into Verilog. Through Verilog, I simulated the circuit, proving my truth table to be correct.