PROJECT STEP 1

SMARTY PANTS

Brian Lee, Chloe Gentry, Vinny Rose

CS.3339 Computer Architecture

Texas State University

September 25, 2024

Contents

1 Introduction

For step 1 we were assigned to design 4 computer circuits. A 1-bit Nand circuit, 1-bit Nor circuit, 1-bit Not circuit, and a 1x4 shifting circuit. We used Verilog to define the code for the circuits and GTKWave to simulate the waveforms for the circuits.

2 1x4 Shift Circuit

The 1x4 shift circuit takes a 4-bit input and shifts the bits to the left or right based on a control signal. This shifting operation moves each bit in the input to an adjacent position, with the vacant positions being filled by zeros.

2.1 Shift Circuit Verilog Code

```
`timescale 1ns / 1ns
`include "shift_gate.v"

module shift_tb;

reg [3:0] A;
wire [3:0] Y;

shift_gate uut(Y, A);

initial begin

    $dumpfile("shift_tb.vcd");//holds output waveform
    $dumpvars(0, shift_tb);

    A = 4'b0000; #10;
    A = 4'b0001; #10;
    A = 4'b0010; #10;
    A = 4'b1111; #10;

    $display("Testing shift");

end

endmodule
```

To test the Shift circuit we have created 2 registers, one for the input and one for the shifting then one wire for the output. If this circuit works properly the output should display the input shifted to the left and zeros filling the shifted positions.

```
`timescale 1ns / 1ns
`include "shift_gate.v"

module shift_tb;

reg [3:0] A;
wire [3:0] Y;

shift_gate uut(Y, A);

initial begin

    $dumpfile("shift_tb.vcd");//holds output waveform
    $dumpvars(0, shift_tb);

    A = 4'b0000; #10;
    A = 4'b0001; #10;
    A = 4'b0010; #10;
    A = 4'b1111; #10;

    $display("Testing shift");

end

endmodule
```

2.2 Shift Circuit Waveforms

At 0 ns, we can see that A is 0, so the output Y is 0.

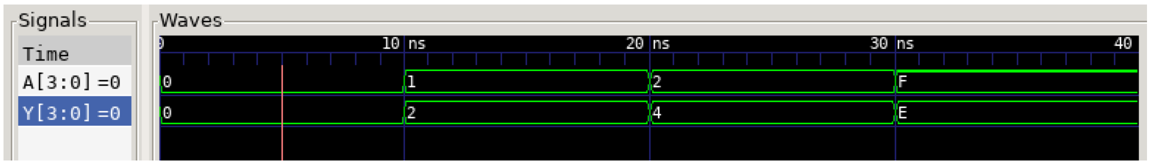

Figure 1: Shift Circuit with marker at 0ns

At 10 ns, A is 1 (0001), so Y becomes 2 (0010).

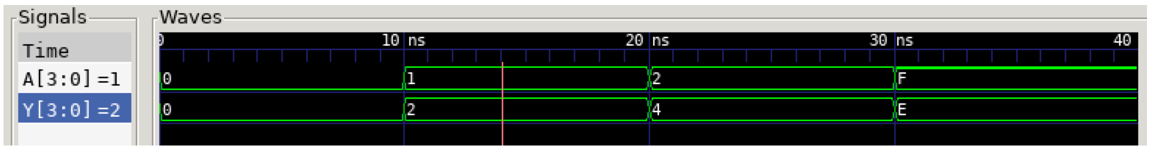

Figure 2: Shift Circuit with marker at 10ns

At 20 ns, A is 2 (0010), so Y becomes 4 (0100).

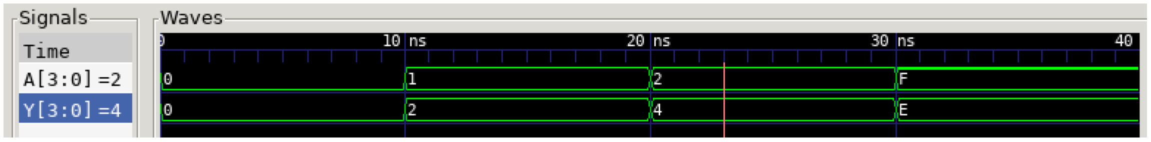

Figure 3: Shift Circuit with marker at 20ns

At 30 ns, A is F (1111) so Y becomes E (1110).

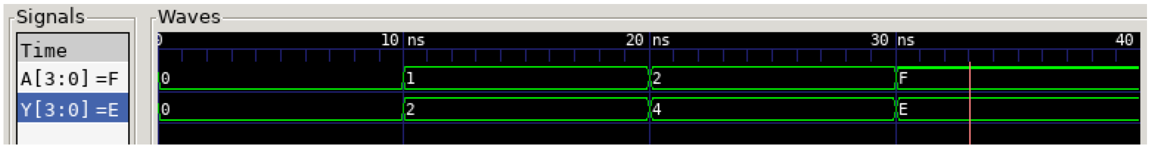

Figure 4: Shift Circuit with marker at 30ns

3 Not Circuit

The Not circuit takes one input A, with an output Y. This will output a 0 if A is a 1, and will output 1 if A is a 0.

‘‘‘

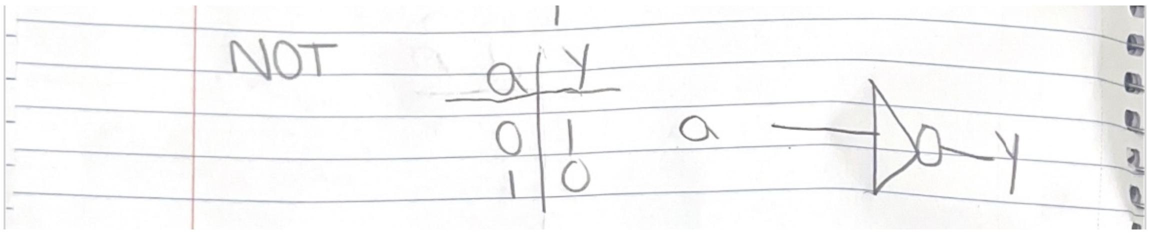

Figure 5: Not truth table and gate

3.1 Not Circuit Verilog Code

```verilog
`ifndef NOT_GATE_V
`define NOT_GATE_V

module not_gate (Y, A);
    output Y;
    input A;
    assign Y = !A;
endmodule

`endif
```

To test the Not circuit, we have created one register A, as well as a wire Y. This way we are able to take each input for A and test output for A=0 and A=1. We'll know if this is working correctly if Y returns 1 for the test where A is 0 and Y returns 0 when A is 1.

```verilog
`timescale 1ns / 1ns
`include "not_gate.v"

module not_tb;

reg A;
wire Y;

not_gate uut(Y, A);

initial begin

    $dumpfile("not_tb.vcd");//holds output waveform
    $dumpvars(0, not_tb);

    A = 0;
    #10;

    A = 1;
    #10;

    $display("Testing not");

end

endmodule
```

3.2 Not Circuit Waveform

At 2 ns, we can see that A is 0, so the output Y is 1.

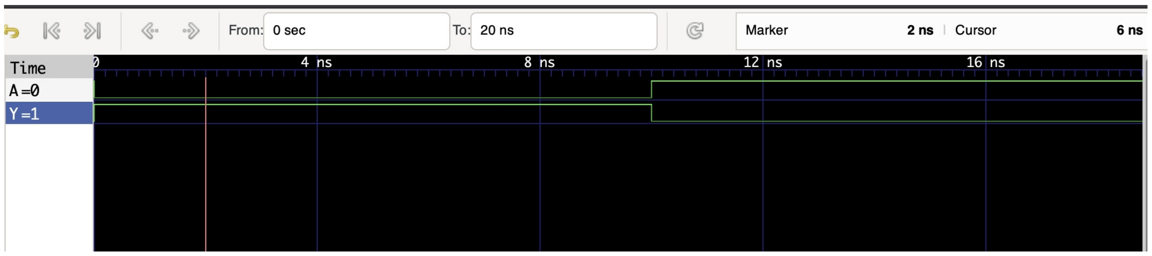

Figure 6: Not Circuit with marker at 2ns

At 10 ns, A becomes 1, so Y becomes 0.

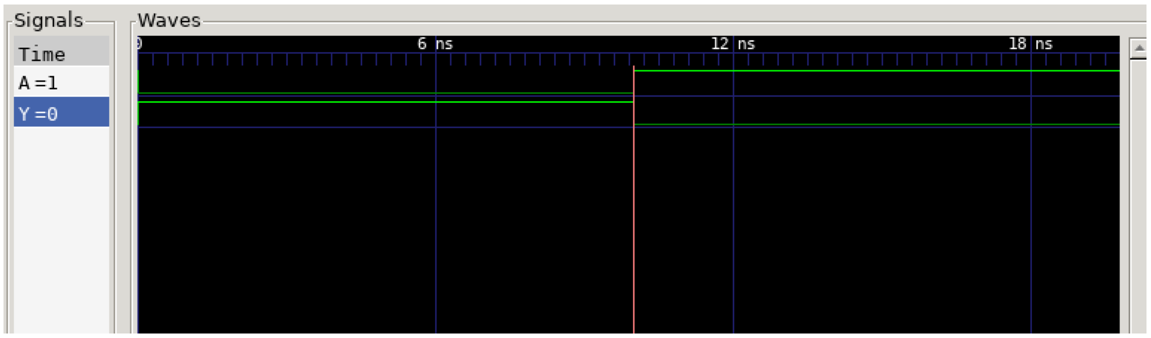

Figure 7: Not Circuit with marker at 10ns

4 Nand Circuit

The Nand circuit takes two inputs, A and B, with an output Y. Nand will only give us back a 0 when A and B are both 1.

‘‘‘

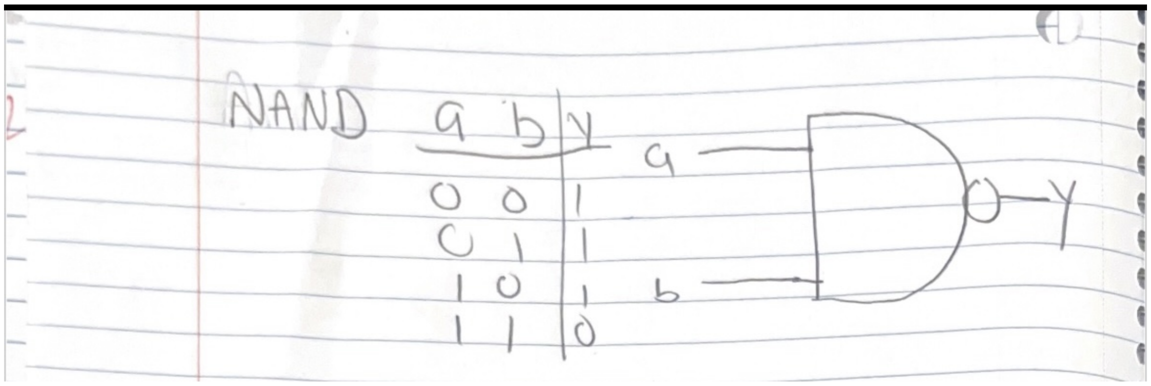

Figure 8: Nand truth table and gate

4.1 Nand Circuit Verilog Code

At 2 ns, we can see that A is 0, so the output Y is 1.

```verilog
`ifndef NAND_GATE_V
`define NAND_GATE_V

module nand_gate (Y, A, B);
    output Y;
    input A, B;
    assign Y = !(A && B);
endmodule

`endif
```

To test the Nand circuit, we have created two registers, A and B, as well as a wire Y. This allows us to test both inputs and one output. We’ll know its correct if Y is 1 for all cases except where A and B are both 1.

```verilog
`timescale 1ns / 1ns
`include "nand_gate.v"

module nand_tb;

reg A, B;
wire Y;

nand_gate uut(Y, A, B);

initial begin
```

```

    $dumpfile("nand_tb.vcd");//holds output waveform
    $dumpvars(0, nand_tb);

    A = 0;
    B = 0;
    #10;

    A = 0;
    B = 1;
    #10;

    A = 1;
    B = 0;
    #10;

    A = 1;
    B = 1;
    #10;

    $display("Testing nand");

end

endmodule
```

4.2 Nand Circuit Waveform

At 0 ns, A and B are 0, so Y is 1.

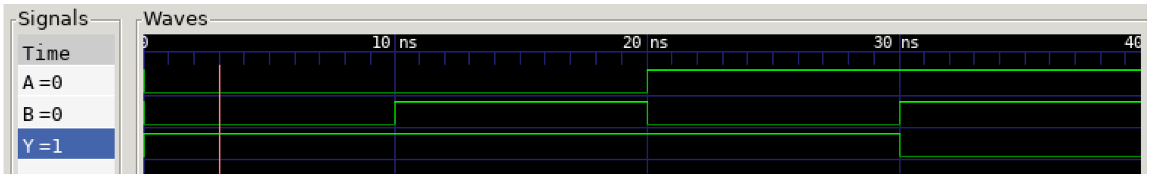

Figure 9: Nand Circuit with marker at 0ns

At 10 ns, A is 0 and B is 1, so Y stays 1.

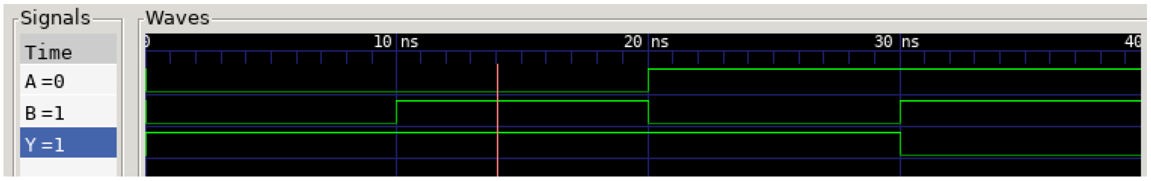

Figure 10: Nand Circuit with marker at 10ns

At 20 ns, A is 1 and B is 0, so Y is still 1.

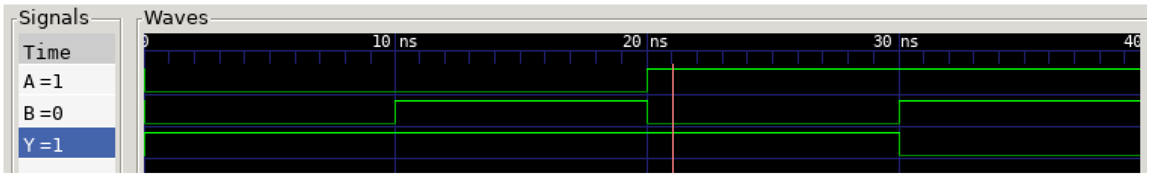

Figure 11: Nand Circuit with marker at 20ns

At 30 ns, both A and B are 1, so Y finally flips to 0.

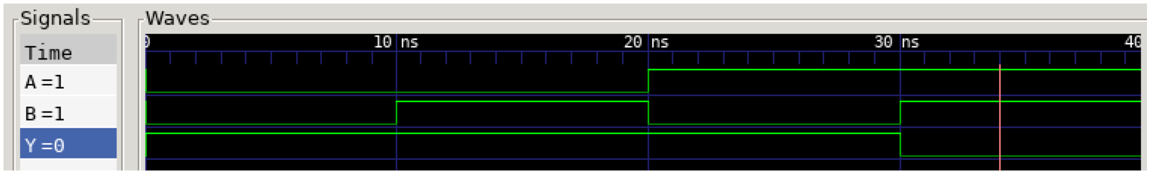

Figure 12: Nand Circuit with marker at 30ns

5 Nor Circuit

The Nor circuit takes two inputs, A and B, with an output Y. This will output a 0 if either A or B is a 1, and will only output 1 if A and B are both 0.

‘‘‘

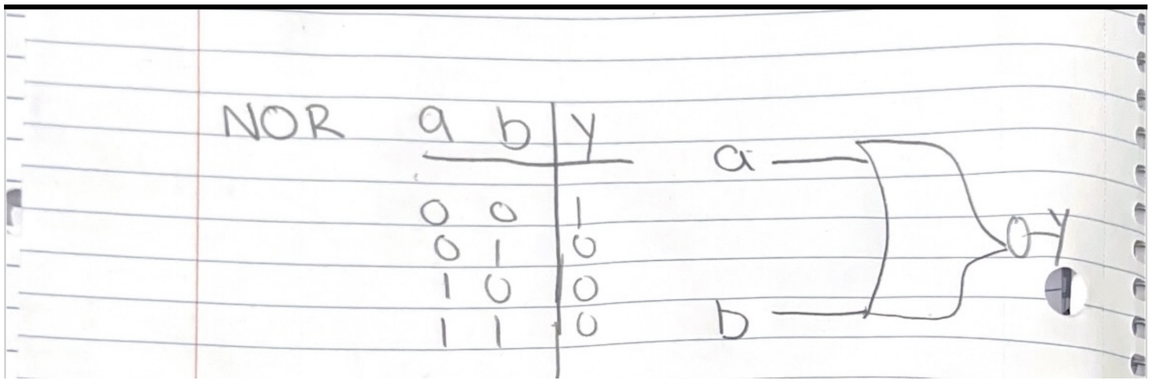

Figure 13: Nor truth table and gate

5.1 Nor Circuit Verilog Code

At 2 ns, we can see that A is 0, so the output Y is 1.

```
`ifndef NOR_GATE_V
`define NOR_GATE_V

module nor_gate (Y, A, B);
    output Y;
    input A, B;
    assign Y = !(A | B);
endmodule

`endif
```

To test the Nor circuit, we have created two registers, A and B, as well as a wire Y. This way we are able to take two inputs at a time and test each possible input for the circuit. We'll know if this is working correctly if Y returns 1 for the test where A and B are 0 and Y should return 0 for the rest.

```
`timescale 1ns / 1ns
`include "nor_gate.v"

module nor_tb;

reg A, B;
wire Y;

nor_gate uut(Y, A, B);

```

```
initial begin

    $dumpfile("nor_tb.vcd");//holds output waveform
    $dumpvars(0, nor_tb);

    A = 0;
    B = 0;
    #10;

    A = 0;
    B = 1;
    #10;

    A = 1;
    B = 0;
    #10;

    A = 1;
    B = 1;
    #10;

    $display("Testing nor");

end

endmodule
```

5.2 Nor Circuit Waveform

At 0 ns, A and B are 0, so Y is 1.

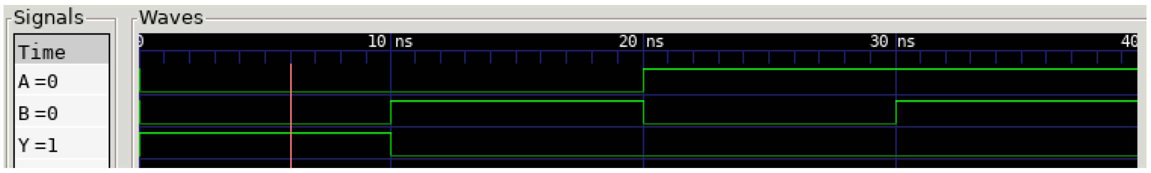

Figure 14: Nor Circuit with marker at 0ns

At 10 ns, A is 0 and B is 1, so Y is now 0.

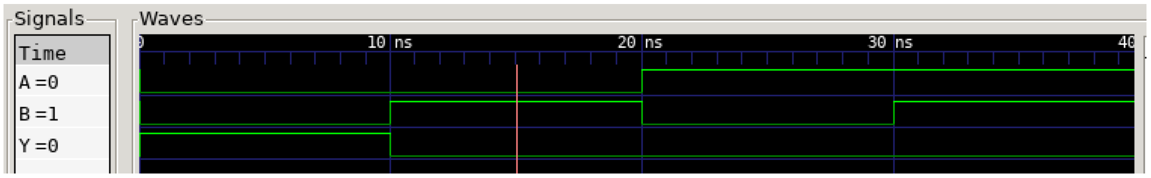

Figure 15: Nor Circuit with marker at 10ns

At 20 ns, A is 1 and B is 0, so Y is still 0.

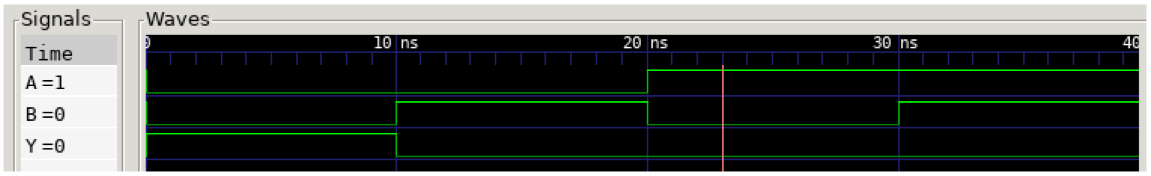

Figure 16: Nor Circuit with marker at 20ns

At 30 ns, both A and B are 1, so Y again stays 0.

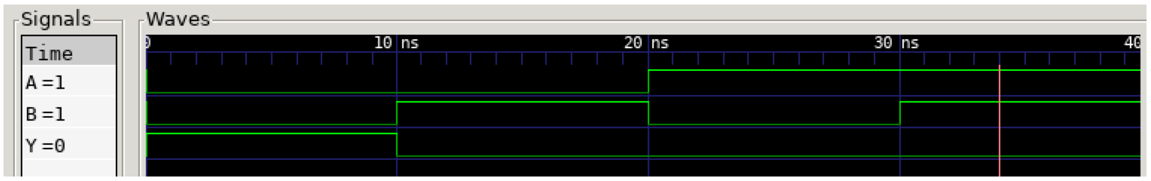

Figure 17: Nor Circuit with marker at 30ns

6 Conclusion

In Conclusion, we created 4 different circuits in Verilog and simulated them in GTKWave. The logic was quite trivial but the biggest trouble we ran into was getting all of the software set up on our respective systems. We worked well together as a group and learned quite a bit about the basics of verilog and GTKWave.