Lab Report
CSE4010 Computer Architecture

Name: Zachary A. Hampton **Score:** /10
Student ID: 008339494 **Due:** 02-16-2025
Lab: Lab 2 - An Understanding of Gates

Report

- What are logic gates? What are universal gates and why are they important?
  - Logic gates are the basic building blocks of digital circuits, performing fundamental Boolean operations such as AND, OR, and NOT.
  - Universal gates, NAND and NOR, are crucial because any digital circuit can be constructed using only these gates, demonstrating their fundamental role in digital logic design.
- Written code on how to make all 7 gates using NAND and NOR.
  - The following Verilog implementations demonstrate how each of the seven basic logic gates can be created using only NAND gates and only NOR gates.

Source Code

NORusingNAND.v

Listing 1: NORusingNAND.v

```
// NOR gate using only NAND gates
// Create module NANDgate with inputs A, B and output Q.
module NANDgate (A, B, Q);

    // Inputs
    input A, B;

    // Output
    output Q;

    // Assign Q = !(A & B)
    assign Q = !(A & B);

endmodule

// Create module NORusingNAND with inputs A, B and output Q.
module NORusingNAND (A, B, Q);

    // Inputs
    input A, B;

    // Output
    output Q;

    // Wires
    wire C, D, E, F;

    // Instantiate NANDgates
    NANDgate u1(A, A, C);
    NANDgate u2(B, B, D);
    NANDgate u3(C, D, E);
    NANDgate u4(E, E, F);

    // Assign Q = F
    assign Q = F;

endmodule
```

NORusingNAND_tb.v

Listing 2: NORusingNAND$_tb.v$

```
// Testbench for NORusingNAND
// Set the timescale for the simulation
`timescale 1ns/1ns

// Include the NORusingNAND module
`include "NORusingNAND.v"

// Create the testbench module
module NORusingNAND_tb;

    // Inputs
    reg A;
    reg B;
    wire Q;

    // Instantiate the NORusingNAND module
    NORusingNAND uut(A, B, Q);

    // Initial begin block
    initial begin

        // Set the dump file and dump variables
        $dumpfile("NORusingNAND_tb.vcd");
        $dumpvars(0, NORusingNAND_tb);

        // Test the NORusingNAND module
        A = 0; B = 0; #20;
        A = 0; B = 1; #20;
        A = 1; B = 0; #20;
        A = 1; B = 1; #20;

        // Display "Complete!" when the simulation is done
        $display("Complete!");
    end

endmodule
```

NANDusingNOR.v

Listing 3: NANDusingNOR.v

```
// NAND gate using only NOR gates
// Create module NORgate with inputs A, B and output Q.
module NORgate (A, B, Q);

    // Inputs
    input A, B;

    // Output
    output Q;

    // NOR operation: Q = !(A | B)
    assign Q = !(A | B);

endmodule

// Create module NANDusingNOR with inputs A, B and output Q.
// This module implements a NAND gate using four NOR gates.
module NANDusingNOR (A, B, Q);

    // Inputs
    input A, B;

    // Output
    output Q;

    // Wires for intermediate signals
    wire C, D, E, F;

    // Instantiate NOR gates to build a NAND gate:
    // u1: Invert A using NOR (A, A)
    NORgate u1(A, A, C);
    // u2: Invert B using NOR (B, B)
    NORgate u2(B, B, D);
    // u3: NOR the inverted signals; note that NOR of C and D
       gives !(C|D) = !( !A or !B) = A and B.
    NORgate u3(C, D, E);
    // u4: Invert E to get NAND; NORing E with itself gives !(E|E
       ) = !E = !(A and B) = A NAND B.
    NORgate u4(E, E, F);

    // Assign output Q
    assign Q = F;

endmodule
```

NANDusingNOR_tb.v

Listing 4: NANDusingNOR$_t b.v$

```
// Testbench for NANDusingNOR
`timescale 1ns/1ns
`include "NANDusingNOR.v"

module NANDusingNOR_tb;

    // Inputs
    reg A;
    reg B;

    // Output
    wire Q;

    // Instantiate the NANDusingNOR module
    NANDusingNOR uut (
        .A(A),
        .B(B),
        .Q(Q)
    );

    // Test stimulus
    initial begin
        // Dump waveform data
        $dumpfile("NANDusingNOR_tb.vcd");
        $dumpvars(0, NANDusingNOR_tb);

        // Test all input combinations with 20ns intervals
        A = 0; B = 0; #20;
        A = 0; B = 1; #20;
        A = 1; B = 0; #20;
        A = 1; B = 1; #20;

        $display("NANDusingNOR␣Test␣Complete!");
        $finish;
    end

endmodule
```

All Seven Gates Using NAND and NOR

allSevenGatesUsingNAND.v

Listing 5: allSevenGatesUsingNAND.v

```
// Implementing all seven gates using NAND only
module allSevenGatesUsingNAND(
    input wire a,
    input wire b,
    output wire and_out,
    output wire or_out,
    output wire nand_out,
    output wire nor_out,
    output wire xor_out,
    output wire xnor_out,
    output wire not_out
);
    // Internal wires
    wire nand_ab;
    wire not_a;
    wire not_b;
    wire nand1, nand2, nand3;

    // NAND gate (directly computed)
    assign nand_ab = ~(a & b);
    assign nand_out = nand_ab;

    // AND gate using NAND: a AND b = NOT(NAND(a,b))
    // Here we use a NAND as inverter: AND = NAND(nand_ab,
       nand_ab)
    assign and_out = ~(nand_ab & nand_ab);

    // NOT gate using NAND: NOT a = NAND(a, a)
    assign not_out = ~(a & a);
    assign not_a = not_out; // reuse not_a for OR computation

    // Generate NOT b signal using NAND: NOT b = NAND(b, b)
    assign not_b = ~(b & b);

    // OR gate using NAND: a OR b = NAND(NOT a, NOT b)
    assign or_out = ~(not_a & not_b);

    // NOR gate using NAND: NOR = NOT(OR) = NAND(OR, OR)
    assign nor_out = ~(or_out & or_out);

    // XOR gate using NAND (4-NAND implementation):
    // nand1 = NAND(a, b)
    // nand2 = NAND(a, nand1)
    // nand3 = NAND(b, nand1)
    // XOR = NAND(nand2, nand3)
    assign nand1 = ~(a & b);
```

```
46      assign nand2 = ~(a & nand1);
47      assign nand3 = ~(b & nand1);
48      assign xor_out = ~(nand2 & nand3);
49
50      // XNOR gate using NAND: XNOR = NOT(XOR) = NAND(xor_out,
            xor_out)
51      assign xnor_out = ~(xor_out & xor_out);
52
53  endmodule
```

allSevenGatesUsingNAND$_t b.v$

Listing 6: allSevenGatesUsingNAND$_t b.v$

```
// Testbench for allSevenGatesUsingNAND
`timescale 1ns/1ns
`include "allSevenGatesUsingNAND.v"

module allSevenGatesUsingNAND_tb;
    // Declare inputs as reg and outputs as wire
    reg a, b;
    wire and_out, or_out, nand_out, nor_out, xor_out, xnor_out,
       not_out;

    // Instantiate the DUT
    allSevenGatesUsingNAND uut (
        .a(a),
        .b(b),
        .and_out(and_out),
        .or_out(or_out),
        .nand_out(nand_out),
        .nor_out(nor_out),
        .xor_out(xor_out),
        .xnor_out(xnor_out),
        .not_out(not_out)
    );

    // Test sequence
    initial begin
        // Generate VCD file
        $dumpfile("allSevenGatesUsingNAND_tb.vcd");
        $dumpvars(0, allSevenGatesUsingNAND_tb);

        // Monitor outputs for each test vector
        $monitor("time=%0t: a=%b, b=%b | and=%b, or=%b, nand=%b,
           nor=%b, xor=%b, xnor=%b, not=%b",
                 $time, a, b, and_out, or_out, nand_out, nor_out,
                     xor_out, xnor_out, not_out);

        // Apply test vectors to all input combinations
        a = 0; b = 0; #10;
        a = 0; b = 1; #10;
        a = 1; b = 0; #10;
        a = 1; b = 1; #10;

        $finish;
    end
endmodule
```

allSevenGatesUsingNOR.v

Listing 7: allSevenGatesUsingNOR.v

```
// Implementing all seven gates using NOR only
// allSevenGatesUsingNOR.v
// Implementing AND, OR, NAND, NOR, XOR, XNOR, and NOT using only NOR gates

module allSevenGatesUsingNOR(
    input  wire a,
    input  wire b,
    output wire and_out,
    output wire or_out,
    output wire nand_out,
    output wire nor_out,
    output wire xor_out,
    output wire xnor_out,
    output wire not_out
);

    // Internal wires
    wire not_a,
    wire not_b;
    wire nor_ab;
    wire term1, term2, xor_temp;

    // NOT gate: NOT a = a NOR a
    assign not_a  = ~(a | a);
    assign not_out = not_a;  // Use not_a for the output NOT gate

    // NOT b: used in other constructions
    assign not_b = ~(b | b);

    // OR gate: a OR b = NOT(a NOR b)
    // First, compute a NOR b
    assign nor_ab = ~(a | b);
    // Then, invert nor_ab (using a NOR as inverter) to get OR
    assign or_out = ~(nor_ab | nor_ab);

    // AND gate: a AND b = ~(~a OR ~b)
    // Since not_a = ~a and not_b = ~b, then:
    assign and_out = ~(not_a | not_b);

    // NAND gate: NAND = NOT(AND) = and_out NOR and_out
    assign nand_out = ~(and_out | and_out);

    // NOR gate: already computed as nor_ab
    assign nor_out = nor_ab;

    // XOR gate using only NOR gates
    // XOR = (a AND NOT b) OR (NOT a AND b)
```

```
48      // Compute a AND NOT b as: ~(~a OR b)  [using NOR as inverter
            ]
49      assign term1 = ~(not_a | b);  // equals a AND ~b
50      // Compute NOT a AND b as: ~(a OR ~b)
51      assign term2 = ~(a | not_b);  // equals ~a AND b
52      // Now, term1 OR term2 = NOT( (term1 NOR term2) )
53      assign xor_temp = ~(term1 | term2);
54      assign xor_out  = ~(xor_temp | xor_temp);  // invert to get (
            term1 OR term2)
55
56      // XNOR gate: simply the inversion of XOR
57      assign xnor_out = ~(xor_out | xor_out);
58
59  endmodule
```

allSevenGatesUsingNOR$_t b.v$

Listing 8: allSevenGatesUsingNOR$_t b.v$

```
// Testbench for allSevenGatesUsingNOR
`timescale 1ns/1ns
`include "allSevenGatesUsingNOR.v"

module allSevenGatesUsingNOR_tb;
    // Declare inputs as reg and outputs as wire
    reg a, b;
    wire and_out, or_out, nand_out, nor_out, xor_out, xnor_out,
       not_out;

    // Instantiate the DUT
    allSevenGatesUsingNOR uut (
        .a(a),
        .b(b),
        .and_out(and_out),
        .or_out(or_out),
        .nand_out(nand_out),
        .nor_out(nor_out),
        .xor_out(xor_out),
        .xnor_out(xnor_out),
        .not_out(not_out)
    );

    // Test sequence
    initial begin
        // Generate VCD file for waveform viewing
        $dumpfile("allSevenGatesUsingNOR_tb.vcd");
        $dumpvars(0, allSevenGatesUsingNOR_tb);

        // Monitor the signals
        $monitor("time=%0t: a=%b, b=%b | and=%b, or=%b, nand=%b,
           nor=%b, xor=%b, xnor=%b, not=%b",
                 $time, a, b, and_out, or_out, nand_out, nor_out
                    , xor_out, xnor_out, not_out);

        // Test all input combinations
        a = 0; b = 0; #10;
        a = 0; b = 1; #10;
        a = 1; b = 0; #10;
        a = 1; b = 1; #10;

        $finish;
    end
endmodule
```

Screenshots for Lab 2

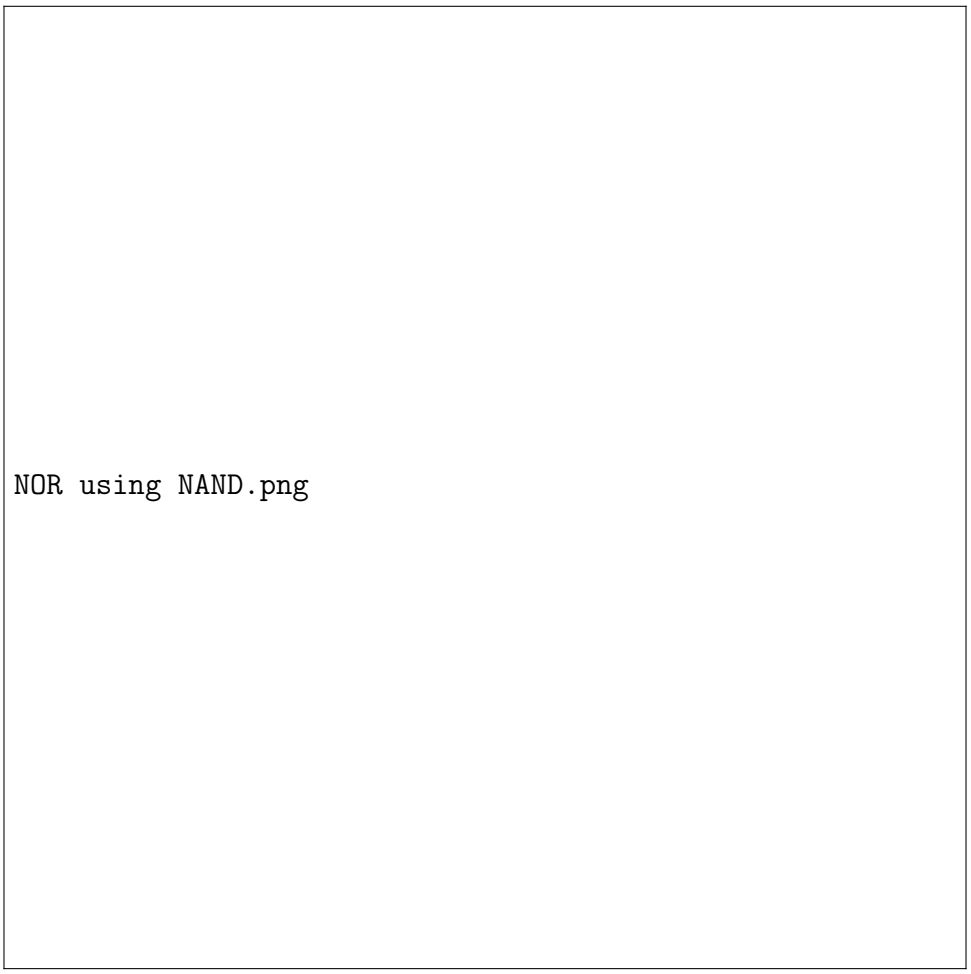

Figure 1: Screenshot for Part A: NOR using NAND

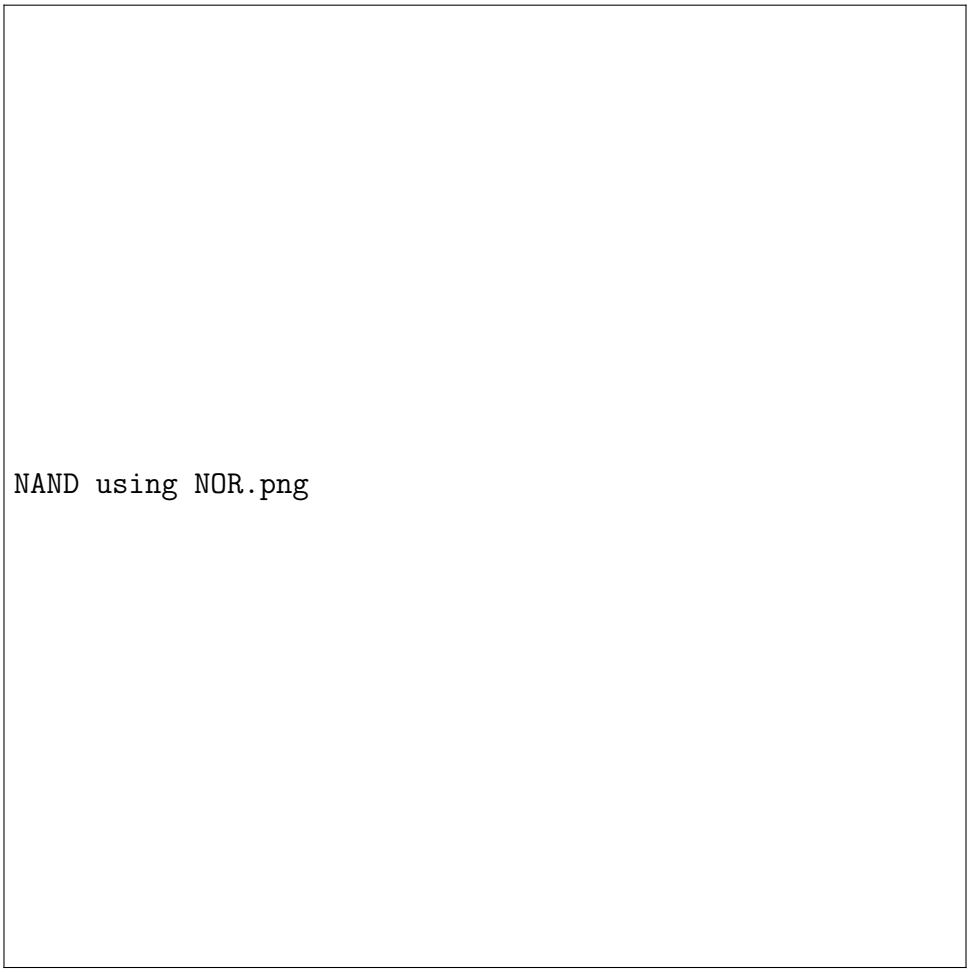

Figure 2: Screenshot for Part B: NAND using NOR

Figure 3: Screenshot for Part C: All Seven Gates Using NAND

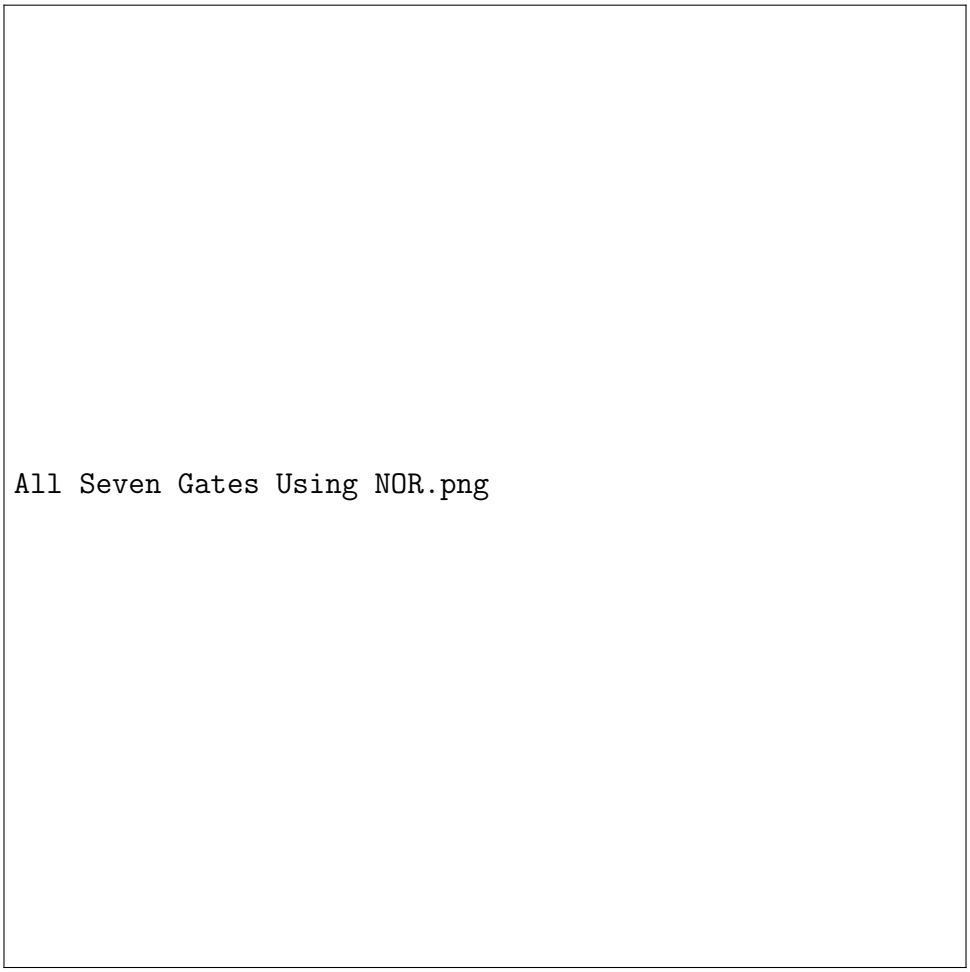

Figure 4: Screenshot for Part D: All Seven Gates Using NOR