



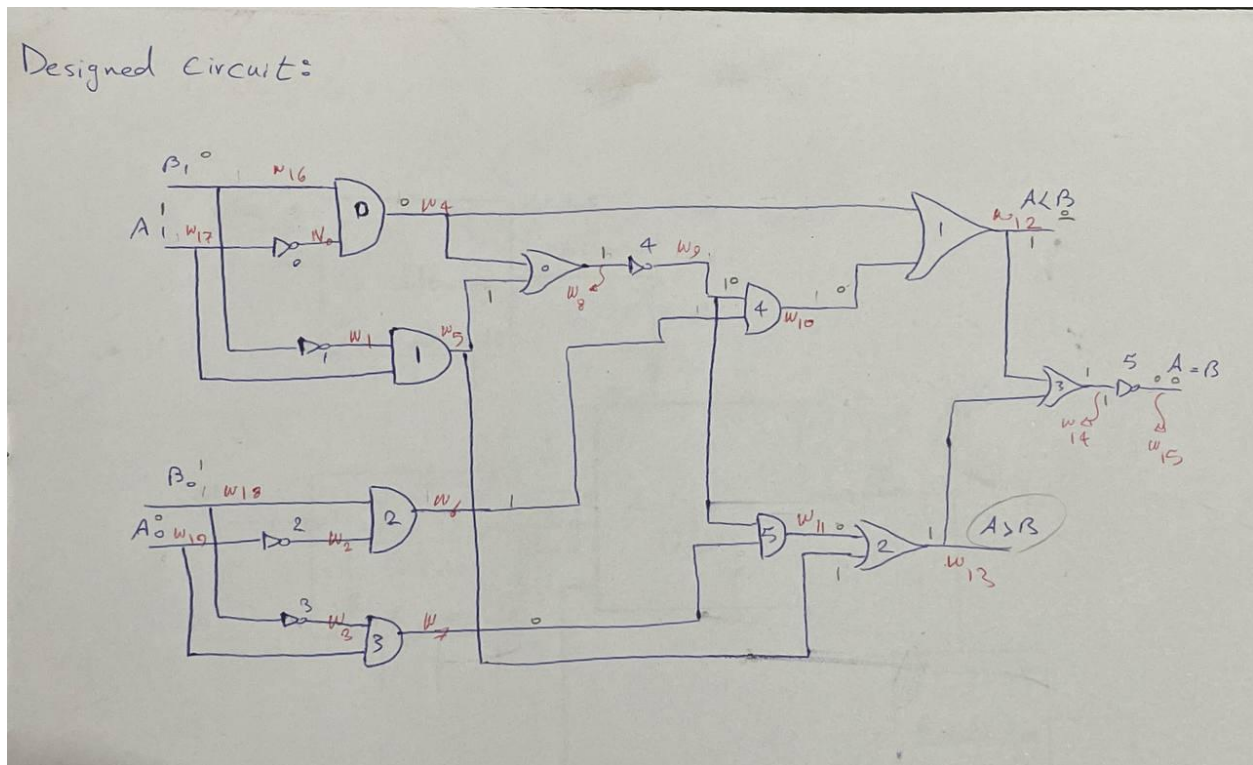
**First Project Report for the Object-Oriented Design of
Electronic Systems Course**

Dr. Navvabi

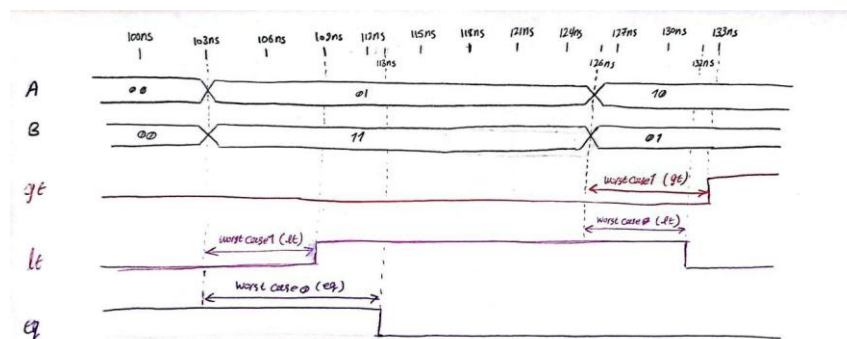
Kiana Rafiei 810101431

Paniz Alamolhoda 810101473

Designed Circuit:



Hand simulate and illustrations:



Verilog design code:

```
`timescale 1ns/1ns

module Comparator(input[1:0] A,B, output gt, lt, eq);
wire [0:15] w;

//and gates
and#3 and1(w[4],w[0],B[1]);
and#3 and2(w[5],w[1],A[1]);
and#3 and3(w[6],w[2],B[0]);
and#3 and4(w[7],w[3],A[0]);
and#3 and5(w[10],w[9],w[6]);
and#3 and6(w[11],w[7],w[9]);

//or gates
or#3 or1(w[8],w[4],w[5]);
or#3 or2(w[12],w[10],w[4]);
or#3 or3(w[13],w[11],w[5]);
or#3 or4(w[14],w[12],w[13]);

//not gates
not#1 not1(w[0],A[1]);
not#1 not2(w[1],B[1]);
not#1 not3(w[2],A[0]);
not#1 not4(w[3],B[0]);
not#1 not5(w[9],w[8]);
not#1 not6(w[15],w[14]);

assign eq = w[15];
assign gt = w[13];
assign lt = w[12];

endmodule
```

Verilog Test bench:

```
// Test case 1
A = 2'b00; B = 2'b00; #20;

// Test case 2
A = 2'b00; B = 2'b01; #20;

// Test case 3
A = 2'b00; B = 2'b10; #20;

// Test case 4
A = 2'b00; B = 2'b11; #20;

// Test case 5
A = 2'b01; B = 2'b00; #20;

// Test case 6
A = 2'b01; B = 2'b01; #20;

// Test case 7
A = 2'b01; B = 2'b10; #20;

// Test case 8
A = 2'b01; B = 2'b11; #20;

// Test case 9
A = 2'b10; B = 2'b00; #20;

// Test case 10
A = 2'b10; B = 2'b01; #20;

// Test case 11
A = 2'b10; B = 2'b10; #20;

// Test case 12
A = 2'b10; B = 2'b11; #20;

// Test case 13
A = 2'b11; B = 2'b00; #20;

// Test case 14
A = 2'b11; B = 2'b01; #20;

// Test case 15
A = 2'b11; B = 2'b10; #20;

// Test case 16
A = 2'b11; B = 2'b11; #20;
```

Verilog simulation:



Project - C:/Users/User/Desktop/Mds/Compute Artitecture/CA1_2/ca11

Name	Status	Type	Order	Modified
testbench.v	✓	Verilog	1	04/20/2025 08:26:51 ...
gatelevel.v	✓	Verilog	0	04/20/2025 08:26:34 ...

C++ illustrations:

Struct defining for input events and adding them to vector in order to make queue.

```
7
8  using namespace std;
9
10 struct InputEvent {
11     int time;
12     string Ainp;
13     string Binp;
14 };
15
16 class Wire {
```

```
1
2 }
3
4 void Manager::read_tst_file(ifstream &file1) {
5     string line;
6
7     int time;
8     while (getline(file1, line)) {
9         if (line.empty() || line[0] != '#')
10             continue;
11
12         istringstream iss(line);
13         char hash;
14         string a, b;
15         iss >> hash >> time >> a >> b;
16         InputEvent event;
17         event.Ainp = a;
18         event.Binp = b;
19         event.time = time;
20         inp_events.push_back(event);
21     }
22 }
23
24 void Manager::manage_assigns() {
```

Concurrent simulator considering timing:

```
8
9 void Manager::set_delta_time() {
10
11     for (int i = 0; i < 4; i++) {
12         and_gates[i]->set_delta(inverter_gates[0]->get_delay());
13         inverter_gates[i]->set_delta(0);
14     }
15     or_gates[0]->set_delta(inverter_gates[0]->get_delay() +
16                             and_gates[0]->get_delay());
17     inverter_gates[4]->set_delta(or_gates[0]->get_delta_time() +
18                                 and_gates[0]->get_delay());
19     and_gates[4]->set_delta(inverter_gates[4]->get_delta_time() +
20                             inverter_gates[0]->get_delay());
21     and_gates[5]->set_delta(and_gates[4]->get_delta_time());
22     or_gates[1]->set_delta(and_gates[4]->get_delta_time() +
23                             and_gates[4]->get_delay());
24     or_gates[2]->set_delta(or_gates[1]->get_delta_time());
25     or_gates[3]->set_delta(or_gates[2]->get_delta_time() +
26                             or_gates[2]->get_delay());
27     inverter_gates[5]->set_delta(or_gates[3]->get_delta_time() +
28                                 or_gates[2]->get_delay());
29 }
30 void Manager::run() {
31     ofstream myFile("result.txt");
32     int current_time = 0;
33     set_delta_time();
34
35     for (int k = 0; k < inp_events.size(); k++) {
36
37         int worst_case_delay =
38             inverter_gates[0]->get_delay() + and_gates[0]->get_delay() +
39             or_gates[0]->get_delay() + inverter_gates[4]->get_delay() +
40             and_gates[4]->get_delay() + or_gates[1]->get_delay() +
41             or_gates[3]->get_delay() + inverter_gates[5]->get_delay() +
42             inp_events[0].time;
43
44         for (int i = 0; i < worst_case_delay + 1; i++) {
45             if (current_time == inp_events[k].time) {
46
47                 A_input = inp_events[k].Ainp;
48                 B_input = inp_events[k].Binp;
49                 manage_assigns();
50
51                 for (auto gate : gates) {
52                     gate->reset();
53                 }
54             }
55
56             for (int j = 0; j < gates.size(); j++) {
57                 if (current_time - inp_events[k].time == gates[j]->get_delta_time()) {
58                     gates[j]->evl();
59                 }
60             }
61             current_time++;
62         }
63         write_output(current_time, myFile);
64     }
65     myFile.close();
66 }
67
68 void Manager::write_output(int current_time, ofstream &myFile) {
69
70     // Write the current time and the state of the circuit to the file
71     myFile << current_time << "\n";
72     for (int i = 0; i < gates.size(); i++) {
73         myFile << gates[i]->get_value() << " ";
74     }
75     myFile << "\n";
76 }
```

C++ tests:

Inputs:

```
#3 11 01  
#14 01 01  
#25 10 11
```

Outputs:

```
# 12 gt : 1  
# 12 lt : 0  
# 12 eq : 0  
# 24 gt : 0  
# 24 lt : 0  
# 24 eq : 1  
# 36 gt : 0  
# 36 lt : 1  
# 36 eq : 0
```

The end.