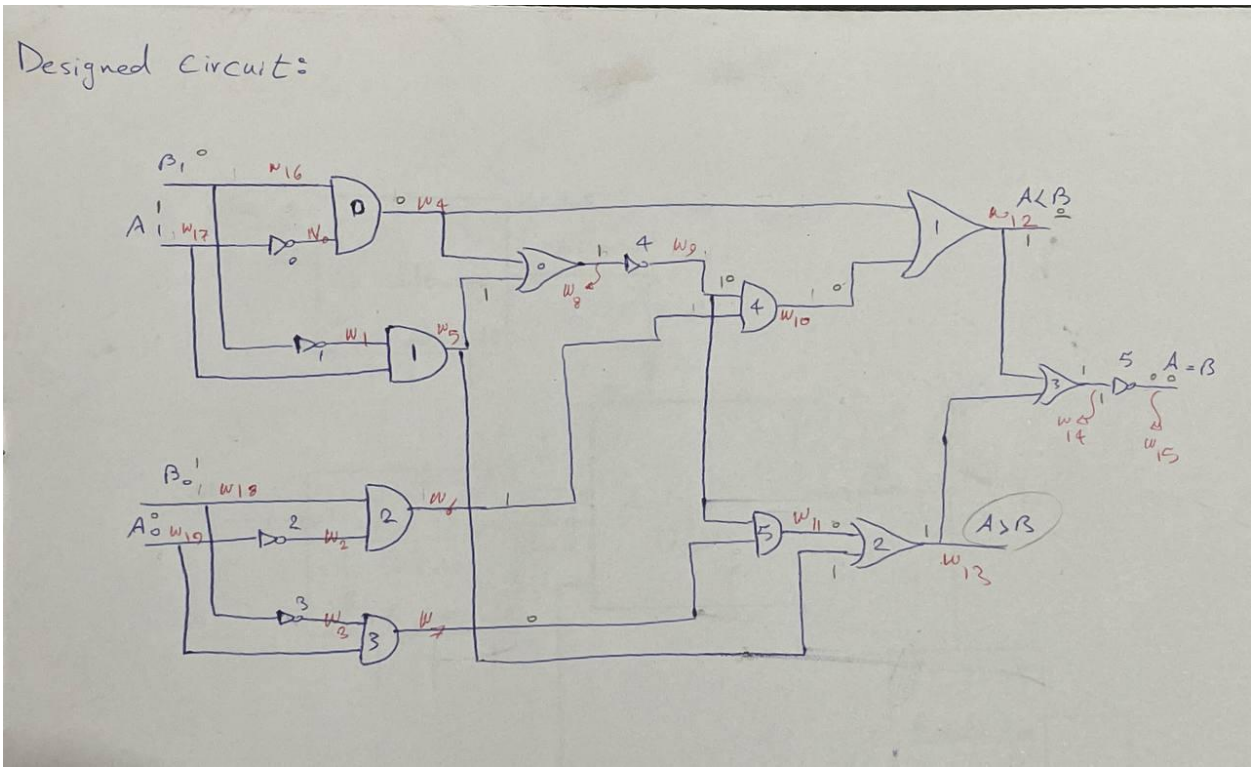
First Project Report for the Object-Oriented Design of Electronic Systems Course

Dr. Navvabi

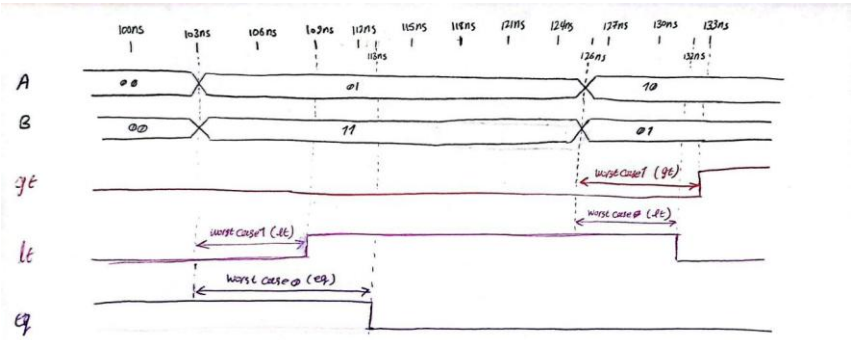
Kiana Rafiei 810101431

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
```

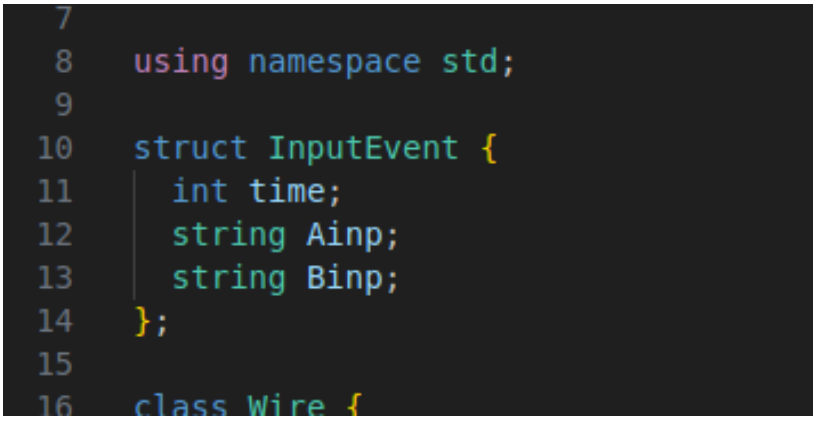
Verilog Test bench:

```
// 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;
```

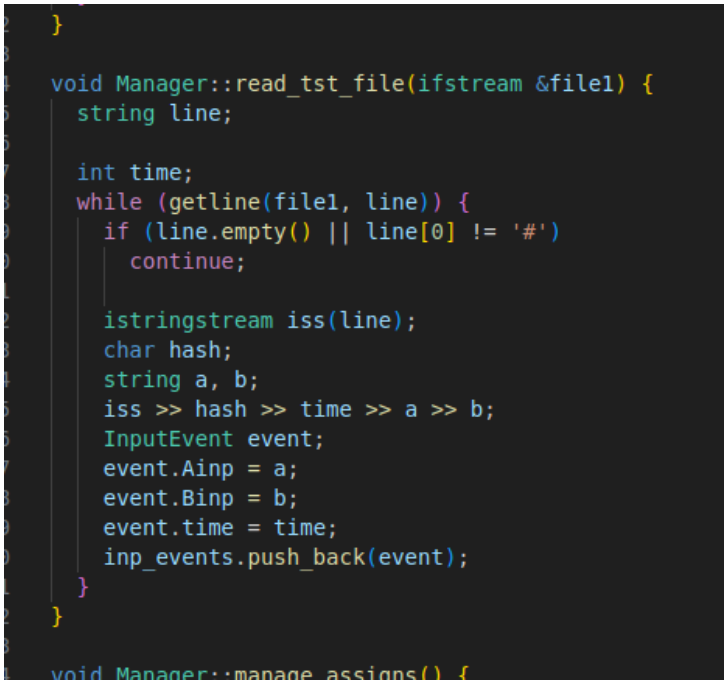
Verilog simulation:

C++ illustrations:

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



```
using namespace std;

struct InputEvent {
    int time;
    string Ainp;
    string Binp;
};

class Wire {
```



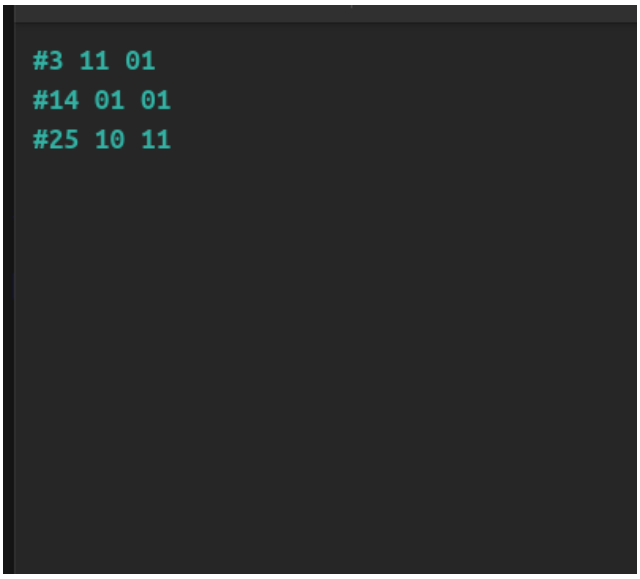
```
}

void Manager::read_tst_file(ifstream &file1) {
  string line;

  int time;
  while (getline(file1, line)) {
    if (line.empty() || line[0] != '#')
      continue;

    istringstream iss(line);
    char hash;
    string a, b;
    iss >> hash >> time >> a >> b;
    InputEvent event;
    event.Ainp = a;
    event.Binp = b;
    event.time = time;
    inp_events.push_back(event);
  }
}

void Manager::manage_assigns() {
```

Concurrent simulator considering timing:

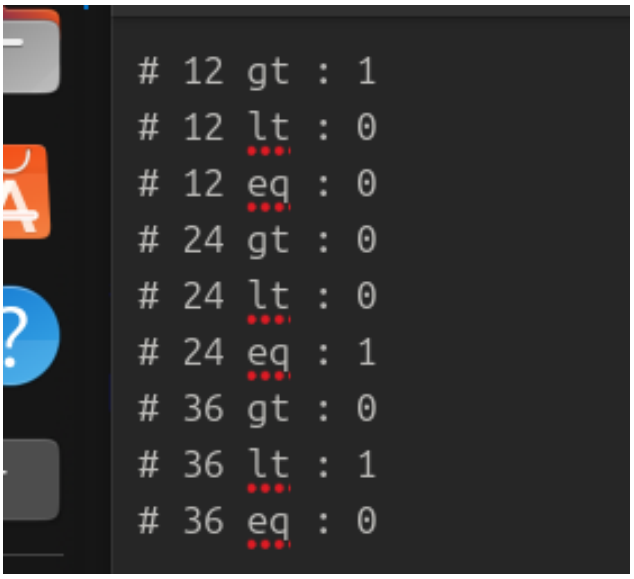
```
void Manager::set_delta_time() {

  for (int i = 0; i < 4; i++) {
    and_gates[i]->set_delta(inverter_gates[0]->get_delay());
    inverter_gates[i]->set_delta(0);
  }
  or_gates[0]->set_delta(inverter_gates[0]->get_delay() +
                         and_gates[0]->get_delay());
  inverter_gates[4]->set_delta(or_gates[0]->get_deta_time() +
                               and_gates[0]->get_delay());
  and_gates[4]->set_delta(inverter_gates[4]->get_deta_time() +
                          inverter_gates[0]->get_delay());
  and_gates[5]->set_delta(and_gates[4]->get_deta_time());
  or_gates[1]->set_delta(and_gates[4]->get_deta_time() +
                         and_gates[4]->get_delay());
  or_gates[2]->set_delta(or_gates[1]->get_deta_time());
  or_gates[3]->set_delta(or_gates[2]->get_deta_time() +
                         or_gates[2]->get_delay());
  inverter_gates[5]->set_delta(or_gates[3]->get_deta_time() +
                               or_gates[2]->get_delay());
}
void Manager::run() {
  ofstream myFile("result.txt");
  int current_time = 0;
  set_delta_time();

  for (int k = 0; k < inp_events.size(); k++) {

    int worst_case_delay =
        inverter_gates[0]->get_delay() + and_gates[0]->get_delay() +
        or_gates[0]->get_delay() + inverter_gates[4]->get_delay() +
        and_gates[4]->get_delay() + or_gates[1]->get_delay() +
        or_gates[3]->get_delay() + inverter_gates[5]->get_delay() +
        inp_events[0].time;

    for (int i = 0; i < worst_case_delay + 1; i++) {
      if (current_time == inp_events[k].time) {

        A_input = inp_events[k].Ainp;
        B_input = inp_events[k].Binp;
        manage_assigns();

        for (auto gate : gates) {
          gate->reset();
        }
      }

      for (int j = 0; j < gates.size(); j++) {
        if (current_time - inp_events[k].time == gates[j]->get_deta_time()) {
          gates[j]->evl();
        }
      }
      current_time++;
    }
    write_output(current_time, myFile);
  }
  myFile.close();
}

void Manager::write_output(int current_time, ofstream &myFile) {

```

C++ tests:

Inputs:



Outputs:



The end.