Erfan Iravani

810197462

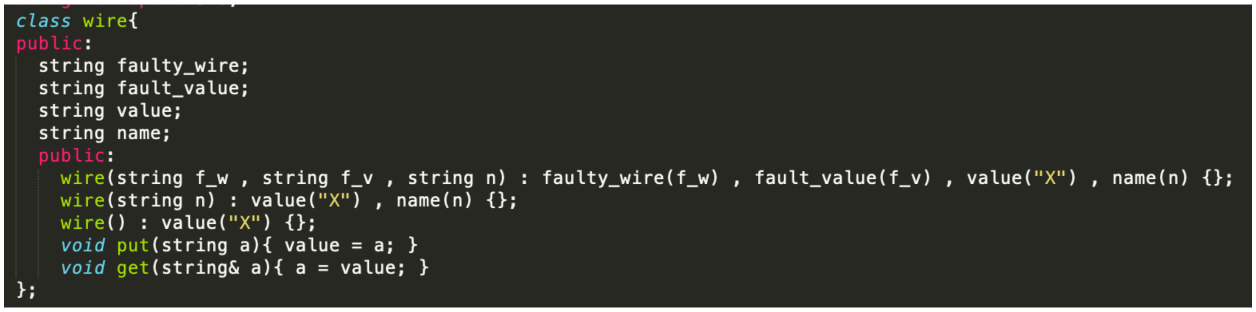
April 17, 2021

CA2 Report

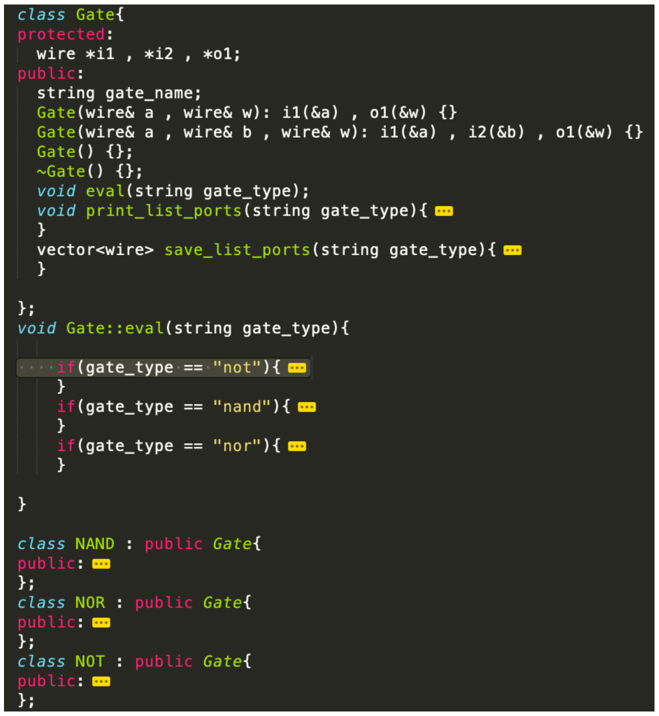
Fault simulation gate classes

1. Generate classes for wires and gates

As discussed in the homework , we generate a **wire** class and consider its name as its unique identifier and we also have a **gate** class that can make and function nand , nor and not gates.



```
class wire{
public:
    string faulty_wire;
    string fault_value;
    string value;
    string name;
    public:
        wire(string f_w , string f_v , string n) : faulty_wire(f_w) , fault_value(f_v) , value("X") , name(n) {};
        wire(string n) : value("X") , name(n) {};
        wire() : value("X") {};
        void put(string a){ value = a; }
        void get(string& a){ a = value; }
};
```



```
class Gate{
protected:
    wire *i1 , *i2 , *o1;
public:
    string gate_name;
    Gate(wire& a , wire& w): i1(&a) , o1(&w) {}
    Gate(wire& a , wire& b , wire& w): i1(&a) , i2(&b) , o1(&w) {}
    Gate() {};
    ~Gate() {};
    void eval(string gate_type);
    void print_list_ports(string gate_type){
    }
    vector<wire> save_list_ports(string gate_type){
    }

};
void Gate::eval(string gate_type){

        if(gate_type == "not"){
        }
        if(gate_type == "nand"){
        }
        if(gate_type == "nor"){
        }

}

class NAND : public Gate{
public:
};
class NOR : public Gate{
public:
};
class NOT : public Gate{
public:
};
```

2. Find possible faults and generate a .txt file showing every fault

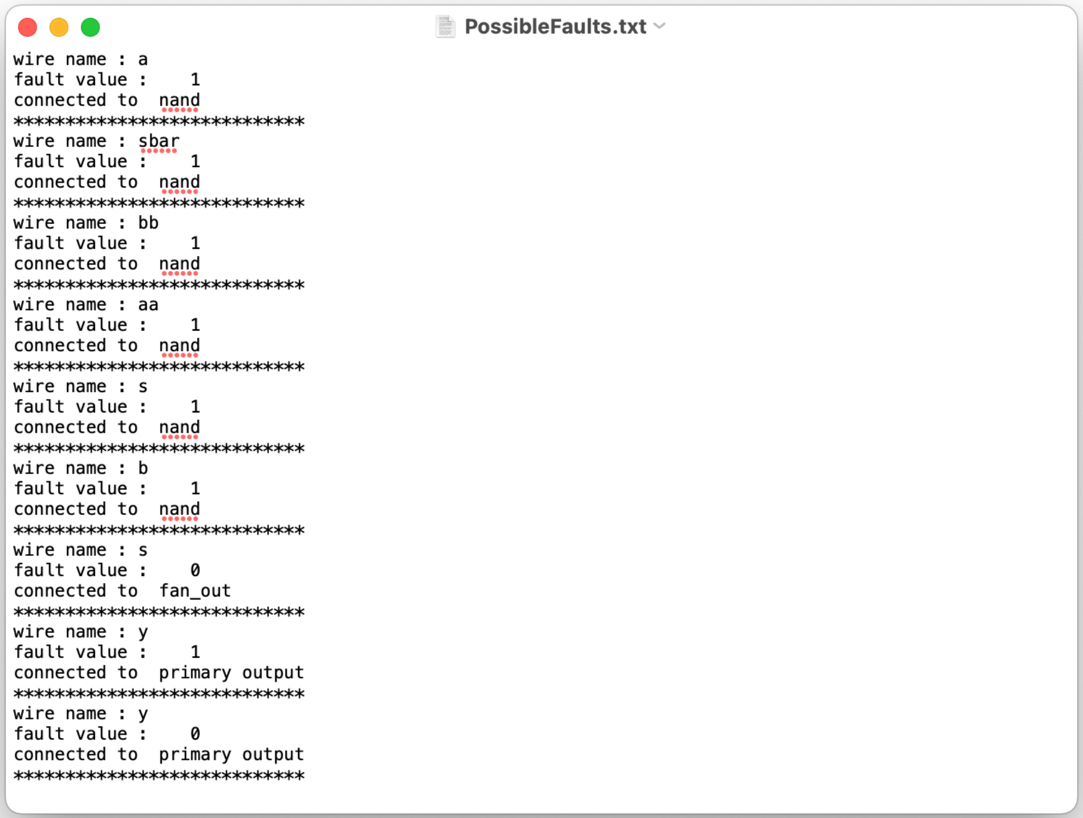
After assigning wires to gates according to verilog file that comes in as an input file , according to the names of gates nand and nor faults can easily be detected.

Primary output faults can also be easily found as we know which wire is an output wire .

For finding Fanout faults we generate a 2D string vector containing the names of wire connected to each gate. Now we can check if a wire is used more than once as an input or not and if it was used more than once , then it is a Fanout .

And finally according to the given table the proper fault value for each wire is set and can be written into a .txt file.

Picture below shows the result of what we did in this part for the example circuit shown in homework.



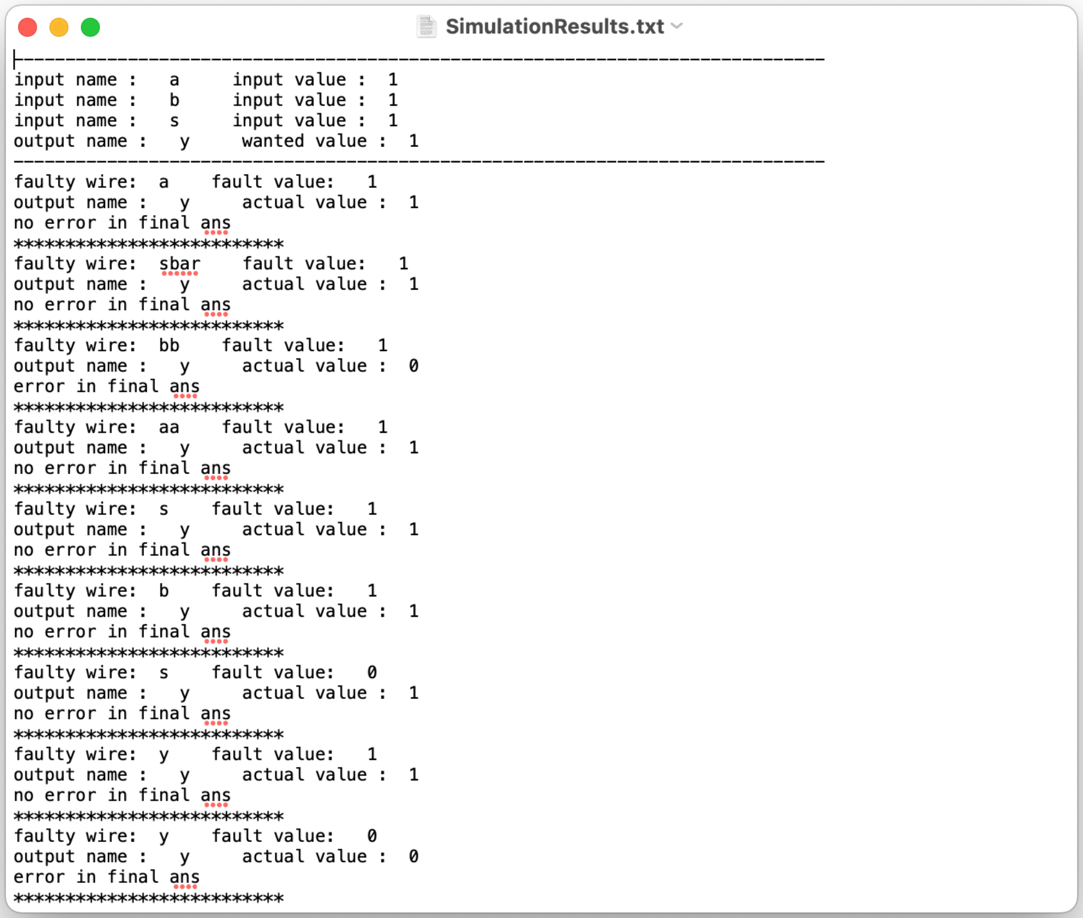
```
wire name : a
fault value :    1
connected to  nand
****************************
wire name : sbar
fault value :    1
connected to  nand
****************************
wire name : bb
fault value :    1
connected to  nand
****************************
wire name : aa
fault value :    1
connected to  nand
****************************
wire name : s
fault value :    1
connected to  nand
****************************
wire name : b
fault value :    1
connected to  nand
****************************
wire name : s
fault value :    0
connected to  fan_out
****************************
wire name : y
fault value :    1
connected to  primary output
****************************
wire name : y
fault value :    0
connected to  primary output
****************************
```

3. Simulate circuit

In this part before testing circuit with faulty wires , we simulate the actual circuit with correct wire values and save the output values in a vector ; so that we can use it to check if our next simulations with faulty wires generate the right output or not.

Using a for loop and the file we generated in previous part , we can make all wires faulty with its proper fault (each loop one wire) and then simulate the circuit and compare the results . and finally we report the results in a .txt file.

The picture underneath demonstrates the results of this part for the example circuit given in homework



```
|------------------------------------------------------------------------------
input name :   a     input value :  1
input name :   b     input value :  1
input name :   s     input value :  1
output name :   y     wanted value :  1
-------------------------------------------------------------------------------
faulty wire:  a    fault value:   1
output name :   y     actual value :  1
no error in final ans
**************************
faulty wire:  sbar    fault value:   1
output name :   y     actual value :  1
no error in final ans
**************************
faulty wire:  bb    fault value:   1
output name :   y     actual value :  0
error in final ans
**************************
faulty wire:  aa    fault value:   1
output name :   y     actual value :  1
no error in final ans
**************************
faulty wire:  s    fault value:   1
output name :   y     actual value :  1
no error in final ans
**************************
faulty wire:  b    fault value:   1
output name :   y     actual value :  1
no error in final ans
**************************
faulty wire:  s    fault value:   0
output name :   y     actual value :  1
no error in final ans
**************************
faulty wire:  y    fault value:   1
output name :   y     actual value :  1
no error in final ans
**************************
faulty wire:  y    fault value:   0
output name :   y     actual value :  0
error in final ans
**************************
```