Object Oriented Electronic Modeling - CA#2 Report

Amir Abbas Moumeni Zadeh - Mahdis Mirzaei

In this Projects we were supposed to Implimennt a Hardware Circuit in Gate Level and Simulate in C++ envirement.

First we need to develop a *Wire* class which is going to be used as inputs and outputs to our gates. After that we need to develop a top *Gate* class and inherit all other gates such as *AND, OR, XOR* and etc from that.

```
7   class gates
8   {
9   protected:
10      Wire *i1, *i2, *i3, *i4, *o1;
11      int gateDelay;
12      char lastOutputValue;
13      bool flag;
14
15  public:
16      gates(Wire &a, Wire &b, Wire &c , Wire &d, Wire &w, int delay) :  i1(&a), i2(&b), i3(&c), i4(&d), o1(&w), gateDelay(delay) {};
17      gates(Wire &a, Wire &b, Wire &c , Wire &w, int delay) :  i1(&a), i2(&b), i3(&c), o1(&w), gateDelay(delay) {};
18      gates(Wire &a, Wire &b, Wire &w, int delay) : i1(&a), i2(&b), o1(&w), gateDelay(delay) {};
19      gates(Wire &a, Wire &w , int delay) : i1(&a), o1(&w), gateDelay(delay) {};
20      gates() {};
21      virtual ~gates() {};
22
23      void evl() {};
24      char out() {return o1->value(); }
25  };
26
```

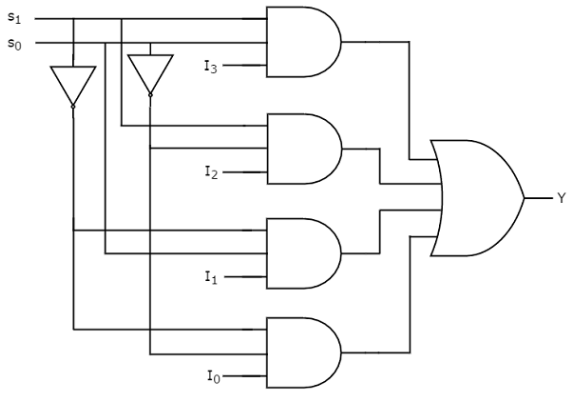
Class Gate

Class *Gate* contains : three type of constructors for 3, 2, 1 input gates , a distructor, and two attributes one for evaluation and one for reading gate output.

```
59  class And_3 : public gates
60  {
61      public:
62      And_3(Wire &a, Wire &b, Wire &c, Wire &w, int delay) : gates(a, b, c, w, delay) {};
63      ~And_3() {};
64      void evl();
65  };
66
67  class Or_3 : public gates
68  {
69      public:
70      Or_3(Wire &a, Wire &b, Wire &c, Wire &w, int delay) : gates(a, b, c, w, delay) {};
71      ~Or_3() {};
72      void evl();
73  };
74
```

```
8   class Mux {
9   protected:
10      Wire *i0, *i1;
11      Wire *not_i0, *not_i1;
12      Wire *d0, *d1, *d2, *d3;
13      Wire *o1;
14      int moduleDelay;
15      char lastOutputValue;
16      bool flag;
17  public:
18      Mux(Wire &a, Wire &b, Wire &c, Wire &d, Wire &e, Wire &f, Wire &w, int delay);
19      Mux() {};
20      ~Mux();
21
22      void evl();
23      char out() {return o1->value(); }
24  };
25
26  #endif
```

Then the *Mux* class developed which is our final module to be implimented with designed gates.



We need four 3-input *AND gates, a 3-input OR* gate and two *Not* gates. Also we need 9 wires for interconnections between gates.

After developing *Mux* circuit, we need to test it in the main function.

```
int main() {

    Wire *a = new Wire();
    Wire *b = new Wire();
    Wire *c = new Wire();
    Wire *d = new Wire();
    Wire *s1 = new Wire();
    Wire *s2 = new Wire();
    Wire *out = new Wire();

    a->setValue(ZERO);
    cout << "00 : " << a->value() <<endl;
    b->setValue(ONE);
    cout << "01 : " << b->value() <<endl;
    c->setValue(ONE);
    cout << "10 : " << c->value() <<endl;
    d->setValue(ZERO);
    cout << "11 : " << d->value() <<endl;
    s1->setValue(ZERO);
    s2->setValue(ZERO);
    cout << "Sel : " << s1->value() << s2->value() <<endl;
    Mux* my_mux = new Mux(*s1, *s2, *a, *b, *c, *d, *out, 0);
    my_mux->evl();

    cout << "output : " << out->value() << endl;

    return 0;
}
```

** its also possible to get inputs from terminal but we fixed them on our testbench.