# ECE 5723 Homework 2

By Bruce Huynh

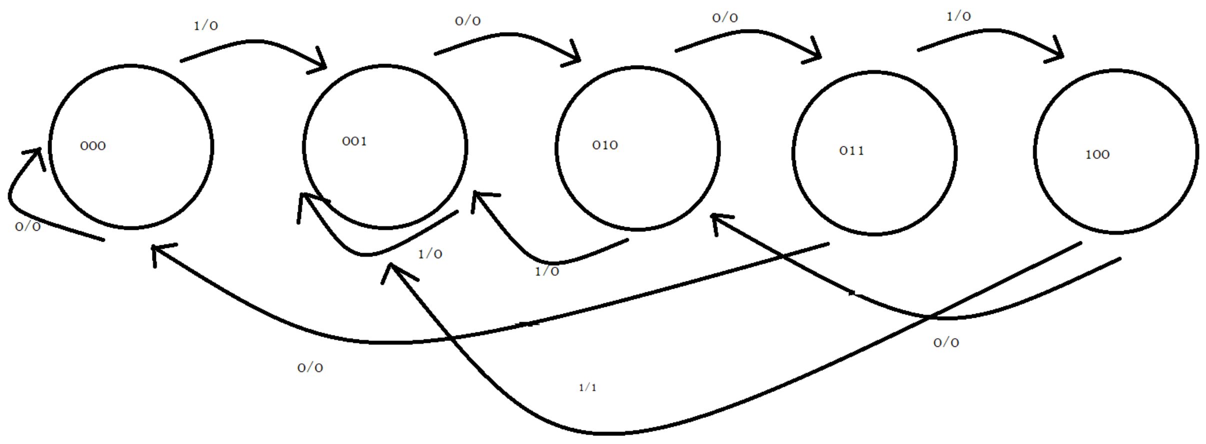
9/26/2021

**Introduction:**

In this assignment, we begin to take steps into modeling hardware using procedural programming languages such as C++. For our first assignment in C++, we had to make a simple sequence detector using a Mealy model.

**A.) State Diagram:**

The following diagram is the Mealy model state diagram for a sequence detector detecting “10011”:



Compared to the Moore model, the Mealy model usually has one less state and its output is based on current input and current state. That is why the outputs happen on the transitions as compared to in the states.

**B.) State Machine Code:**

Class declaration in ConsoleApplication1.h:

```
#include "D:\Logic Modeling Cpp Codes v21\Logic Modeling Cpp Codes
v21\18_Logic Class
Inheritance_Wire\inheritedLogicClassesPrimitives.h"



class Statemachine {
      wire *in, *clk, *rst, *out;

      int Nstate, Pstate;
public:
      Statemachine(wire& in, wire& rst, wire& clk, wire& out);
      ~Statemachine();
      void evl();

};
```

In this class, we defined Statemachine as a class with three wire inputs and one wire output. Aside from those, we also instantiated our two states: Nstate and Pstate.

State machine in ConsoleApplication1.cpp:

```
#include "ConsoleApplication1.h"


Statemachine::Statemachine(wire& in, wire& clk, wire& rst, wire& out)
{
    this->in = &in; this->clk = &clk; this->rst = &rst; this->out =
&out;
    Nstate = 0;
    Pstate = 0;
}

void Statemachine::evl() {
    out->value = '0';
```

```c
switch (Pstate) {
case 0:
    if (in->value == '1') {
        Nstate = 1;
        out->value = '0';
    }
    else {
        Nstate = 0;
        out->value = '0';
    }
    break;

case 1:
    if (in->value == '1') {
        Nstate = 1;
        out->value = '0';
    }
    else {
        Nstate = 2;
        out->value = '0';
    }
    break;

case 2:
    if (in->value == '1') {
        Nstate = 1;
        out->value = '0';
    }
    else {
        Nstate = 3;
        out->value = '0';
    }
    break;

case 3:
    if (in->value == '1') {
        Nstate = 4;
        out->value = '0';
        }
        else {
            Nstate = 0;
            out->value = '0';
        }
        break;

    case 4:
        if (in->value == '1') {
            Nstate = 1;
            out->value = '1';
        }
        else {
            Nstate = 2;
            out->value = '0';
        }
        break;


    }

    if (rst->value == '1')
        Pstate = 0;
    else if (clk->value == 'P')
        Pstate = Nstate;


}
```

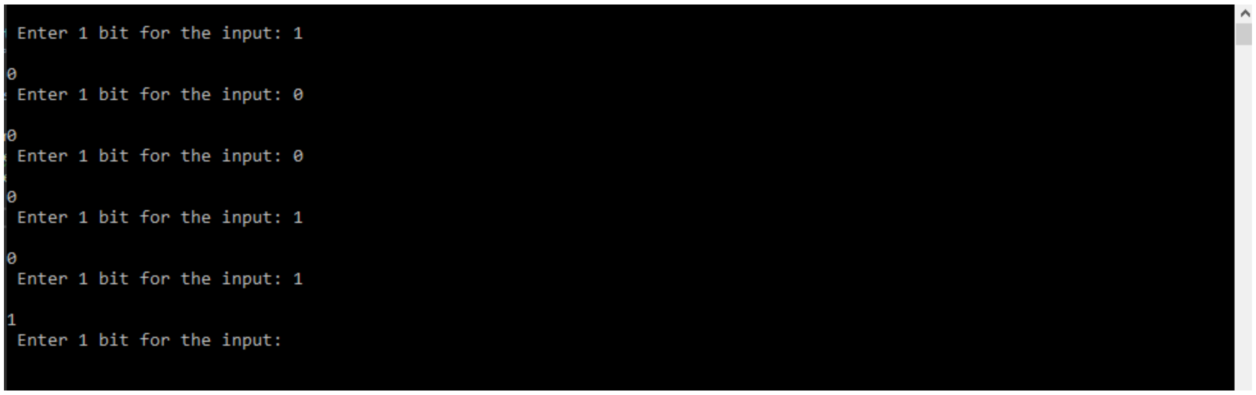
Our state machine works primarily off of switch cases. Note that instead of the output depending on the current state, the output is declared in the transition to the next state.

**C.) Testbench**

With C++, we are able to open the console and enter the serial information ourselves. Therefore, the testbench will comprise mainly of user inputs. The following is the code used to accomplish this:

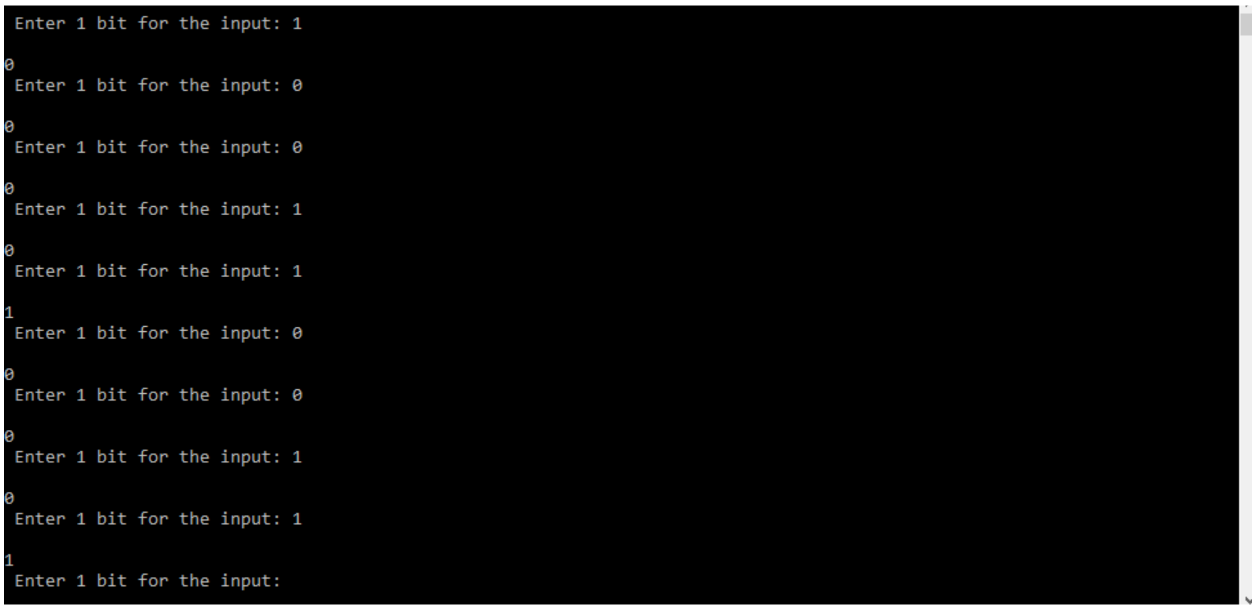
```
#include "ConsoleApplication1.h"


int main()
{
    int ij;
    wire in;
    wire clk;
    wire rst;
    wire out;


    Statemachine* Statemachine1 = new Statemachine(in, clk, rst,
out);

    rst.value = '1';
    Statemachine1->evl();
    rst.value = '0';

    do {
        for (int i = 0; i < 20; i++) {
            cout << "\n Enter 1 bit for the input: "; cin >>
in.value;
            clk.value = 'P';
            Statemachine1->evl();
            cout << "\n" << out.value;
        }

        cout << "\n" << "Continue (0 or 1)?"; cin >> ij;

    } while (ij > 0);
}
```

**Testing “10011”:**


```
 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input: 1

1
 Enter 1 bit for the input:
```

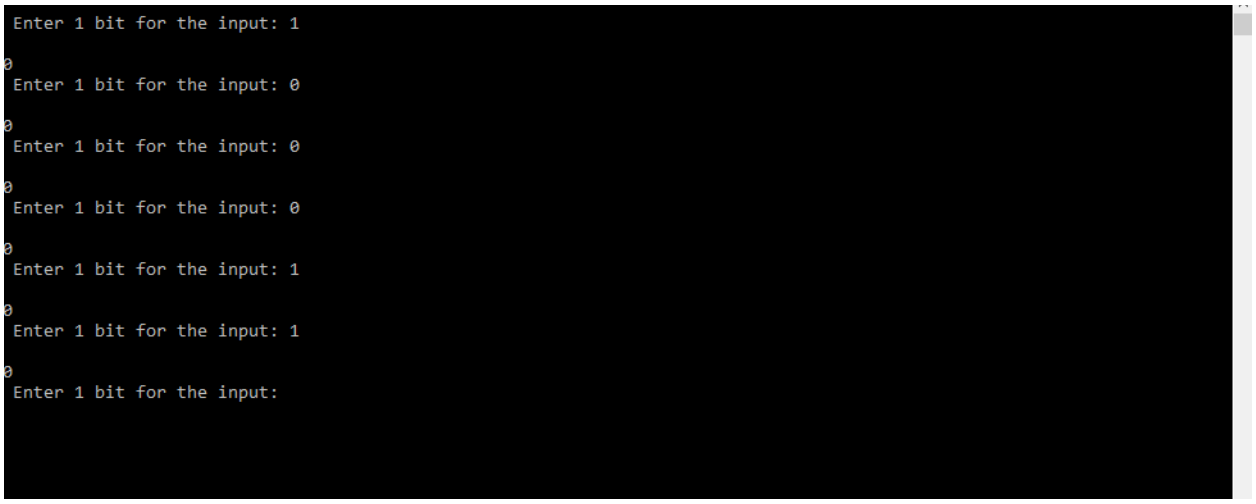
Our state machine detects the sequence correctly and outputs ‘1’ when successfully detected.

**Testing “100110011”:**


```
 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input: 1

1
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input: 1

1
 Enter 1 bit for the input:
```

Our state machine detects the sequence twice and outputs ‘1’ each time the sequence is detected.

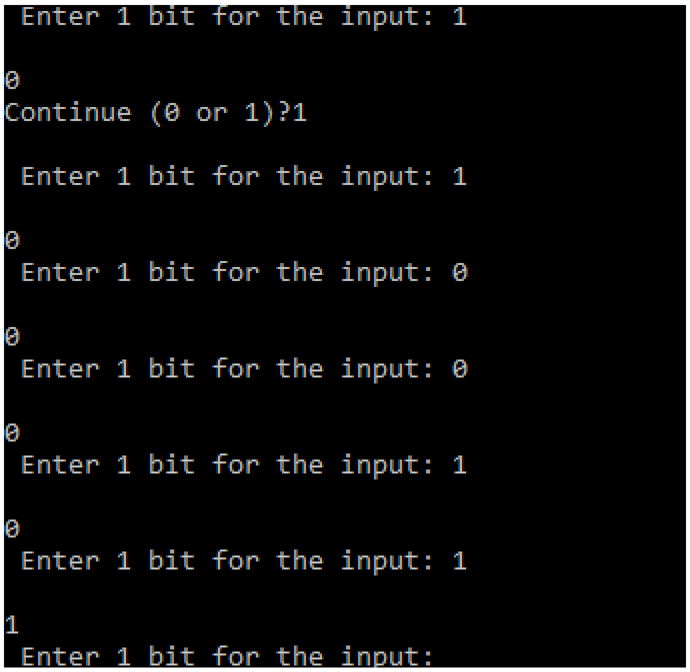
**Testing “100011”:**



```
 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input:
```

Our state machine never detects the right sequence so it never outputs ‘1’.

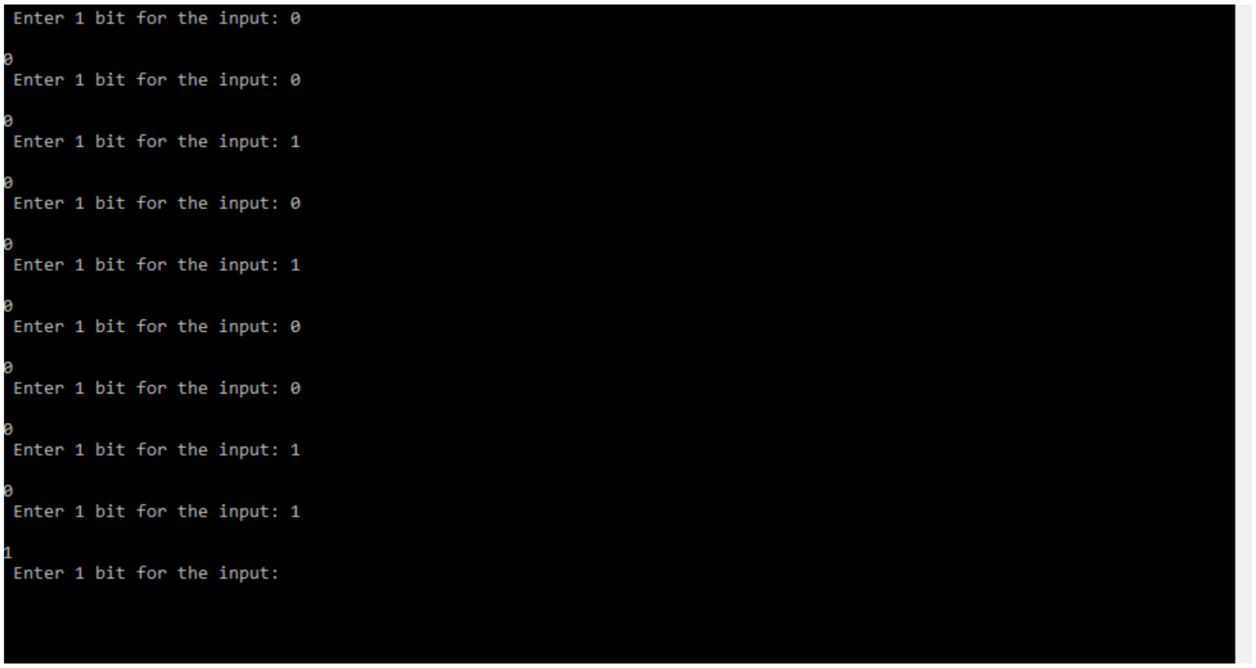
**Testing “10011” after Continue:**



```
 Enter 1 bit for the input: 1

0
Continue (0 or 1)?1

 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input: 1

1
 Enter 1 bit for the input:
```

After 20 inputs (A value I chose randomly to simulate more than five inputs), we press continue and try to enter the right sequence. Our state machine detects the right input after continue and outputs ‘1’;

**Testing “001010011”:**



```
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 0

0
 Enter 1 bit for the input: 1

0
 Enter 1 bit for the input: 1

1
 Enter 1 bit for the input:
```

Our state machine does not receive the right input for the first four inputs. Afterwards, the correct sequence is imputed and it outputs a ‘1’. This shows that our sequence detector is not index dependent. We do not try to look at the sequences five inputs at a time.

**Conclusion:**

This assignment allowed us to dip our feet into C++ and modeling hardware using it. As our first C++ assignment, it is noticeably easier than the last assignment as we have to get used to using a procedural programming language again. In the coming assignments, the projects will get more difficult but our C++ will get better to match this difficulty as well.