ECE 5723 Homework 3

By Bruce Huynh

10/5/2021

Introduction:

In this assignment, we work more with hardware simulation using C++. This time we are tasked with making a circuit that keeps taking the average of its bus input. It will output the average to us once there is an end pulse.

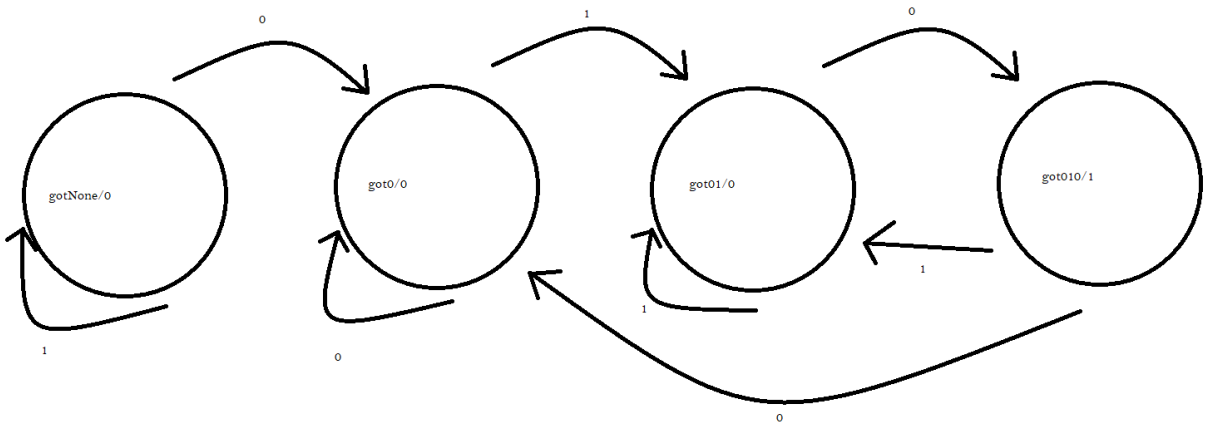
A.) Operator Overloading:

We were given a premade bus package and we were tasked to overload an operator that takes the average of two busses. Here is how I implemented this:

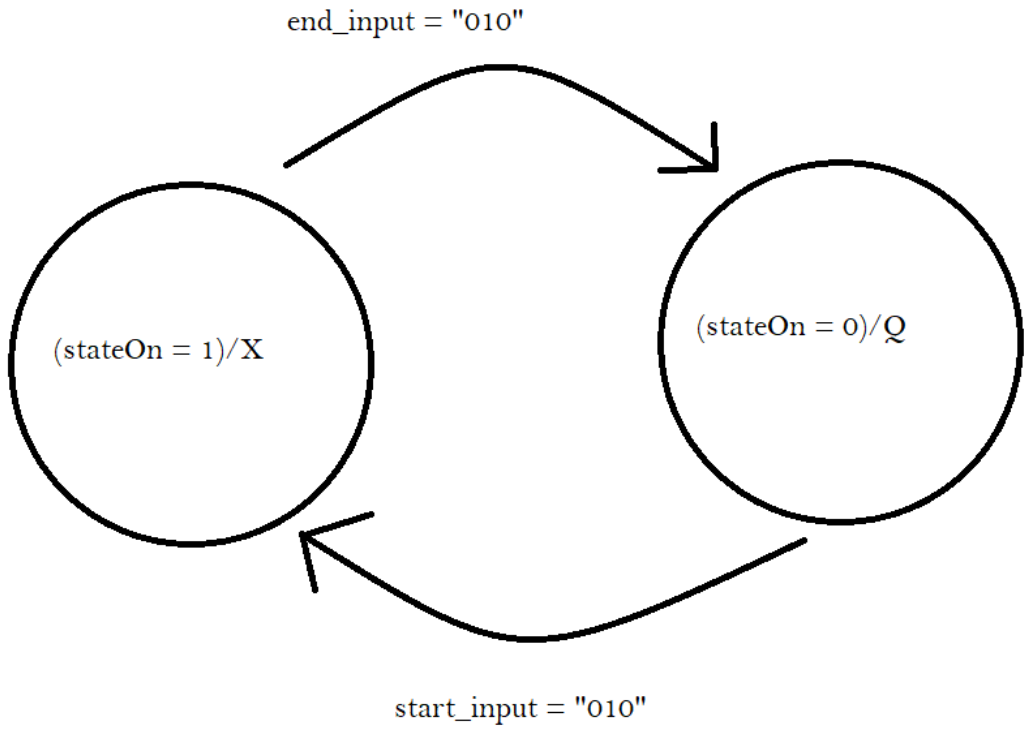
```
      friend bus operator/ (bus a, bus *b) {
            int aSize = a.v.length();
            int bSize = b->v.length();
            int rSize;
            int bit_num;
            string rS;
            int max = MAX(aSize, bSize);
            if (bSize == 1) rSize = aSize; else rSize = max;

            int aInt = stoi(a.v, 0, 2); // integer conversion
            int bInt = stoi(b->v, 0, 2); // integer conversion
            int ABaverage = (aInt + bInt) / 2; // averaging
            while (ABaverage != 0) {
                  rS = (ABaverage % 2 == 0 ? "0" : "1") + rS;
ABaverage /= 2; // binary conversion
            }
            bus r(rSize, 'X');
            r.v = rS;
            return r;
      }
```

B.) Datapath and Controller:

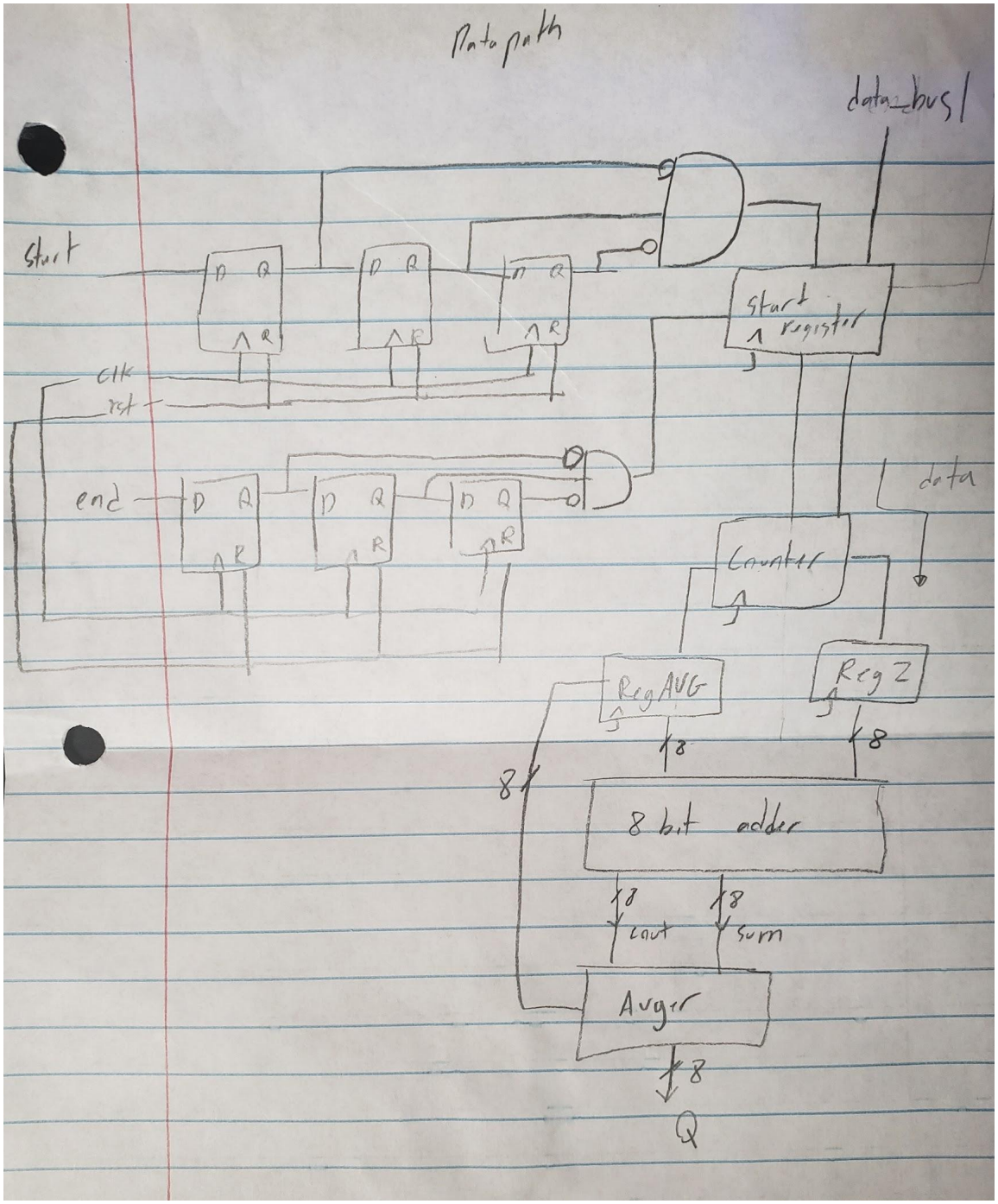
For both start and end pulses, here is the state diagram showing how both are controlled. The output ‘1’ tells the system to either start averaging or stop averaging.



This next diagram helps show the relationship between the pulses and the output of the circuit:



The following is the datapath used when coming up with the circuit:



C.) Datapath and Controller code:

Datapath:

```
class averager{

public:
	bus* start1;
	bus* end1;
	bus* D1;
	bus* C1;
	bus* R1;
	bus* Q1;
	//int present_stateS;
	//int next_stateS;
	//int present_stateE;
	//int next_stateE;
	int stateOn;
	int currAvg;
	bus avgBus = "00000000";

	averager(bus &start_input, bus &END, bus& D, bus& C, bus& R,
bus& Q);
	~averager();
	void evl();
};
```

```
averager::averager(bus& start_input, bus& END, bus& D, bus& C, bus&
R, bus& Q) :

	start1(&start_input), end1(&END), D1(&D), C1(&C), R1(&R),
Q1(&Q)

{}
```

Controller:

```
if (start1->v == "010") {
                        stateOn = 1;
                        /*end1->v = "000";*/
                    }
                    if (end1->v == "010") {
                        stateOn = 0;
                        Q1->v = avgBus.v;
                        /*start1->v = "000";*/
                    }
```

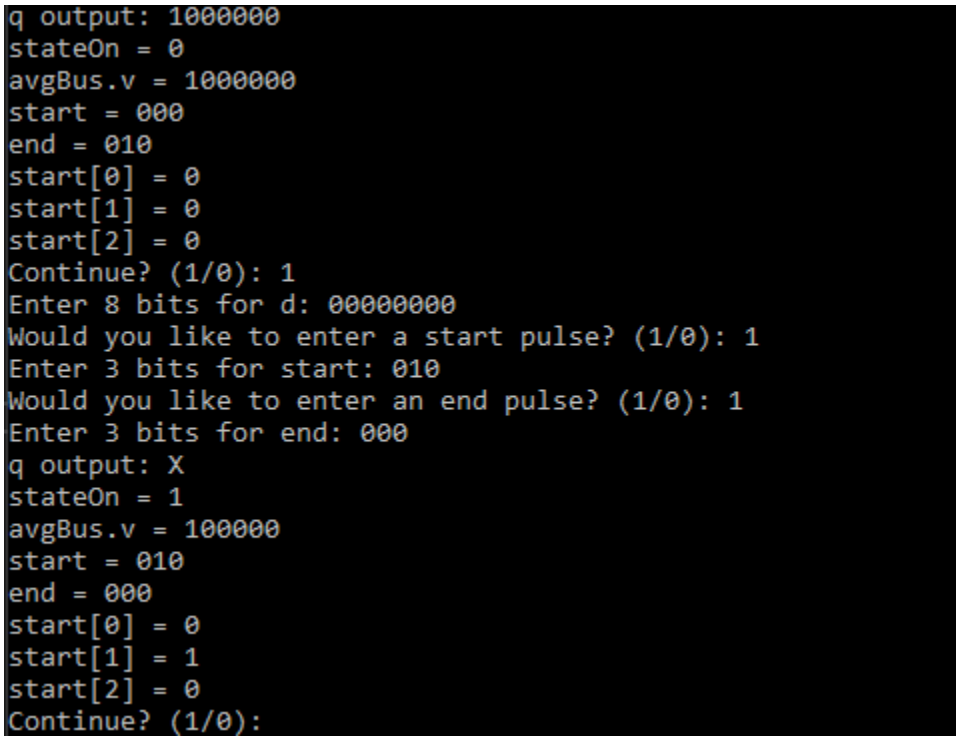
D.) Testbenches

Test 1: Initial start input and 8 bit bus (10000000), then end:

```
Enter N amount of bits: 8
Enter 8 bits for d: 10000000
Would you like to enter a start pulse? (1/0): 1
Enter 3 bits for start: 010
Would you like to enter an end pulse? (1/0): 0
q output: X
stateOn = 1
avgBus.v = 1000000
start = 010
end = XXX
start[0] = 0
start[1] = 1
start[2] = 0
Continue? (1/0): 1
Enter 8 bits for d: 00000000
Would you like to enter a start pulse? (1/0): 1
Enter 3 bits for start: 000
Would you like to enter an end pulse? (1/0): 1
Enter 3 bits for end: 010
q output: 1000000
stateOn = 0
avgBus.v = 1000000
start = 000
end = 010
start[0] = 0
start[1] = 0
start[2] = 0
Continue? (1/0): 1
```

This test shows that the averaging function does work and the averaging stops when the end input is detected. It also shows that the average is accumulated and remembered.

Test 2: Initial start input and 8 bit bus (10000000), then end and restart:



This test shows that when we end and then start again, the Q output goes back to unknown and the averaging continues again.

Test 3: Initial start input and 4 bit bus (1010), then another continue:

```
Enter N amount of bits: 4
Enter 4 bits for d: 1010
Would you like to enter a start pulse? (1/0): 1
Enter 3 bits for start: 010
Would you like to enter an end pulse? (1/0): 0
q output: X
stateOn = 1
avgBus.v = 101
start = 010
end = XXX
start[0] = 0
start[1] = 1
start[2] = 0
Continue? (1/0): 1
Enter 4 bits for d: 0000
Would you like to enter a start pulse? (1/0): 1
Enter 3 bits for start: 000
Would you like to enter an end pulse? (1/0): 0
q output: X
stateOn = 1
avgBus.v = 10
start = 000
end = XXX
start[0] = 0
start[1] = 0
start[2] = 0
Continue? (1/0):
```

This test shows that the circuit works with different amounts of user bits. It also shows that the circuit floors integer values that are not divisible by two (five goes down to two). Although the value cannot directly be seen on the Q output, it can be seen with avgBus.V.

Test 4: Initial start input with 6 bit bus, starting with value “100100” (36) and trying to get average value of “100111” (39):

```
Enter N amount of bits: 6
Enter 6 bits for d: 100100
Would you like to enter a start pulse? (1/0): 1
Enter 3 bits for start: 010
Would you like to enter an end pulse? (1/0): 0
q output: X
stateOn = 1
avgBus.v = 10010
start = 010
end = XXX
start[0] = 0
start[1] = 1
start[2] = 0
Continue? (1/0): 1
Enter 6 bits for d: 111100
Would you like to enter a start pulse? (1/0): 1
Enter 3 bits for start: 000
Would you like to enter an end pulse? (1/0): 0
q output: X
stateOn = 1
avgBus.v = 100111
start = 000
end = XXX
start[0] = 0
start[1] = 0
start[2] = 0
Continue? (1/0):
```

This test shows that averaging works on non-zero values that we enter for d. This test also shows that we can accurately use the circuit to find a specific average if we would like.

Conclusion:

In this assignment, we strengthened our knowledge of C++ and although we haven't directly gotten into SystemC, many concepts from this homework will still apply and aid us in the future.