ECE 5723 Homework 4

By Bruce Huynh

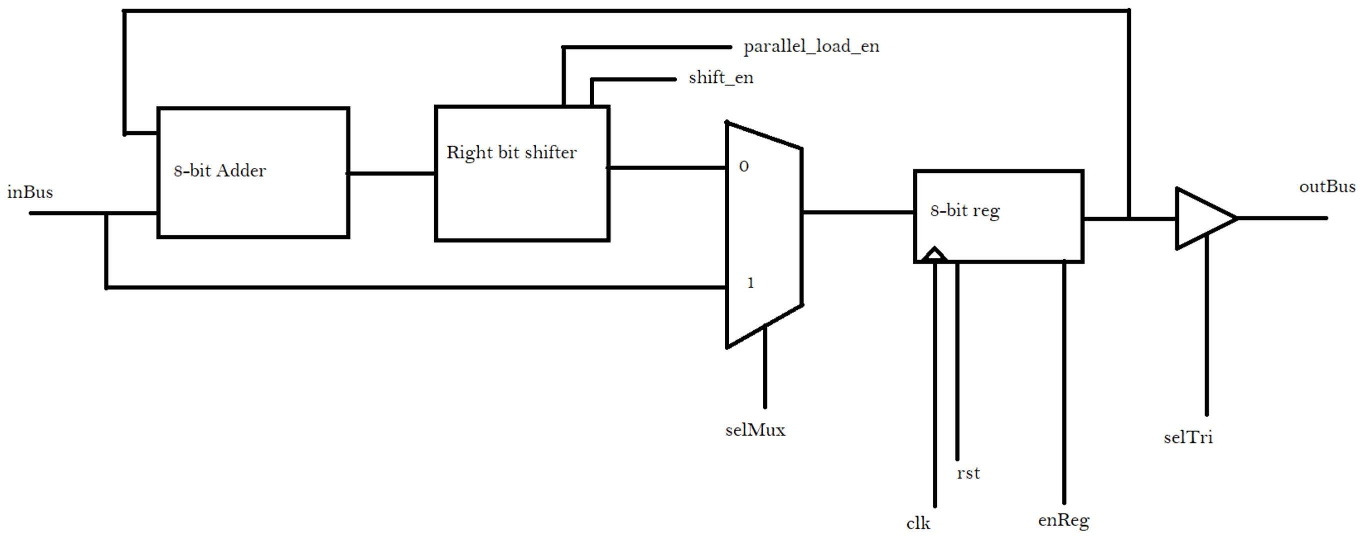
10/24/2021

Introduction:

In this assignment, we had to take our C++ based circuit and adapt it using the SystemC library. Specifically, we have to make a circuit that takes the average of the inBus and the previous averages.

A.) Datapath

The following is the datapath diagram for the circuit:



The following is the SystemC datapath implementation for the circuit:

```
#include <systemc.h>
#include "A:\SystemC Modeling Codes-1\SystemC Modeling
Codes\4_partsLibrary\partsLibrary.h"
```

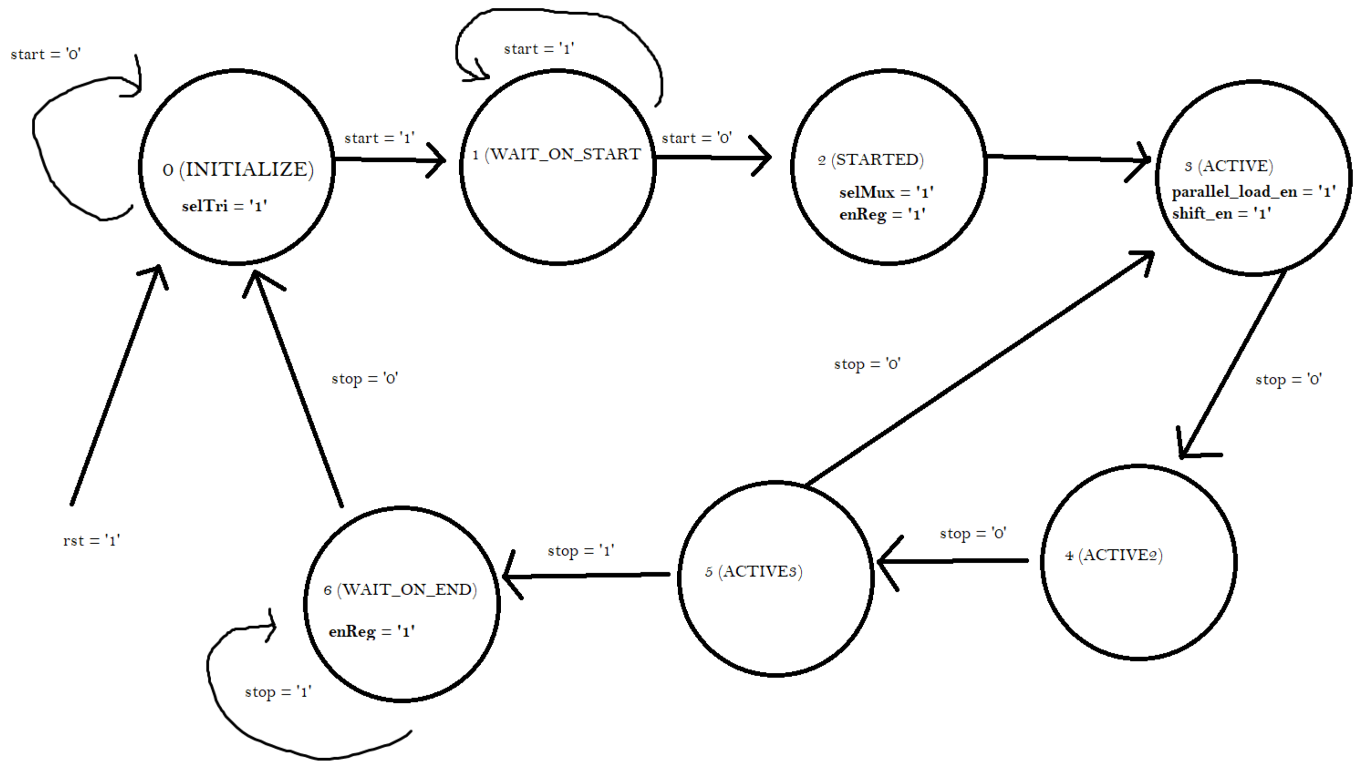
```
SC_MODULE(accAverage_DP) {

public:
	sc_in<sc_logic> rst, clk, enableReg, selMux, selTri,
parallel_load_en, shift_en;
	sc_in<sc_lv<8>> inBus;
	sc_out <sc_lv<8>> outBus;

	sc_signal<sc_lv<8>> outOpr, outMux, outReg, outAdd;
	sc_uint<8> inBus_uint, outReg_uint, outOpr_uint;
	sc_logic zero = SC_LOGIC_0;


	// instantiation
	octalMux2to1* myMux;
	dRegisterRaE* myReg;
	octalTriState* myTri;
	nBitAdder* myAdder;
	rShifterRaEL* myShifter;

	SC_CTOR(accAverage_DP) {
		outOpr = "00000000";
		outMux = "00000000";
		outReg = "00000000";



		myMux = new octalMux2to1("Multiplexer");
		myMux->sel(selMux);
		myMux->ain(outOpr);
		myMux->bin(inBus);
		myMux->yout(outMux);


		myReg = new dRegisterRaE("Register");
		myReg->clk(clk);
		myReg->rst(rst);
		myReg->cen(enableReg);
		myReg->regin(outMux);
```

```
            myReg->regout(outReg);



            myTri = new octalTriState("Tri-State");
            myTri->sel(selTri);
            myTri->ain(outReg);
            myTri->yout(outBus);

            myAdder = new nBitAdder("8-bit-adder");
            myAdder->ain(inBus);
            myAdder->bin(outReg);
            //myAdder->ci();
            myAdder->addout(outAdd);
            //myAdder->co();

            myShifter = new rShifterRaEL("Bit-shifter");
            myShifter->rst(rst);
            myShifter->clk(clk);
            myShifter->pld(parallel_load_en);
            myShifter->sen(shift_en);
            //myShifter->sin();
            myShifter->parin(outAdd);
            myShifter->shftout(outOpr);

      }
};
```

The averaging is done with the adder and the bit shifter. Shifting the added values to the right one bit is essentially dividing. The register and the tri-state are used in conjunction with the controller in order to store the average and to show the output when the stop signal is received. This will be elaborated further in the next part.

B.) Controller and State Diagram

This is the state diagram for the controller:



This controller has states for both awaiting the start pulse and awaiting the stop pulse. Following the start pulse, the output becomes high impedance due to the tri-state. The mux and register are then enabled. It goes to the active states which comprise three states. The reason for three active states is that it takes multiple clock cycles to add the logic vectors and to shift the bits over. Afterwards, if a stop pulse is received by the time of the 3rd active state, it will prepare to output the average or else it will return to the first active state.

Here is the code for the controller:

(.cpp file)

```cpp
#include "accAverage_CU.h"

void accAverage_CU::comb_S_function() {
	Nstate = INITIALIZE;

	switch (Pstate) {
	case INITIALIZE:
		if (start == '0') Nstate = INITIALIZE;
		else Nstate = WAIT_ON_START;
		break;

	case WAIT_ON_START:
		if (start == '1') Nstate = WAIT_ON_START;
		else Nstate = STARTED;
		break;

	case STARTED:
		Nstate = ACTIVE;
		break;

	case ACTIVE:
		if (stop == '0') Nstate = ACTIVE2;
		//else Nstate = WAIT_ON_END;
		break;

	case ACTIVE2:
		if (stop == '0') Nstate = ACTIVE3;
		//else Nstate = WAIT_ON_END;
		break;

	case ACTIVE3:
		if (stop == '0') Nstate = ACTIVE;
		else Nstate = WAIT_ON_END;
		break;
```

```
     case WAIT_ON_END:
          if (stop == '1') Nstate = WAIT_ON_END;
          else Nstate = INITIALIZE;
          break;

     default:
          Nstate = INITIALIZE;
          break;
     }
}

void accAverage_CU::comb_O_function() {
     enableReg = SC_LOGIC_0;
     selMux = SC_LOGIC_0;
     selTri = SC_LOGIC_0;

     switch (Pstate) {
     case INITIALIZE:
          selTri = SC_LOGIC_1;
          shift_en = SC_LOGIC_0;
          parallel_load_en = SC_LOGIC_0;
          break;

     case STARTED:
          selMux = SC_LOGIC_1;
          enableReg = SC_LOGIC_1;

          break;

     case ACTIVE:
          shift_en = SC_LOGIC_1;
          parallel_load_en = SC_LOGIC_1;

          break;

     case ACTIVE2:
          //shift_en = SC_LOGIC_1;
          //parallel_load_en = SC_LOGIC_0;
```

```
        break;

    case ACTIVE3:

    case WAIT_ON_END:
        enableReg = SC_LOGIC_1;
        break;

    default:
        enableReg = SC_LOGIC_0;
        selMux = SC_LOGIC_0;
        selTri = SC_LOGIC_0;
        break;
    }
}

void accAverage_CU::seq_function() {
    while (true) {
        if (rst == '1') {
            Pstate = INITIALIZE;
        }
        else if (clk->event() && (clk == '1')) {
            Pstate = Nstate;
            cout << "Pstate: " << Pstate << " *** Nstate: " <<
Nstate << endl;
            cout << " start: " << start << " *** stop:   " <<
stop << endl;


        }
        wait();
    }
}
```

(.h file)

```
#include <systemc.h>


SC_MODULE(accAverage_CU){
public:
	sc_in<sc_logic> start, stop, rst, clk;
	sc_out<sc_logic> enableReg, selMux, selTri, parallel_load_en,
shift_en;

	enum state_types {INITIALIZE, WAIT_ON_START, STARTED, ACTIVE,
ACTIVE2, ACTIVE3, WAIT_ON_END};

	sc_signal<state_types> Nstate, Pstate;

	SC_CTOR(accAverage_CU) {
		SC_THREAD(seq_function);
		sensitive << rst << clk;
		SC_METHOD(comb_S_function);
		sensitive << start << stop << Pstate;
		SC_METHOD(comb_O_function);
		sensitive << Pstate;
	}
	void seq_function();
	void comb_S_function();
	void comb_O_function();
};
```

C.) Top-level Module and Testbench

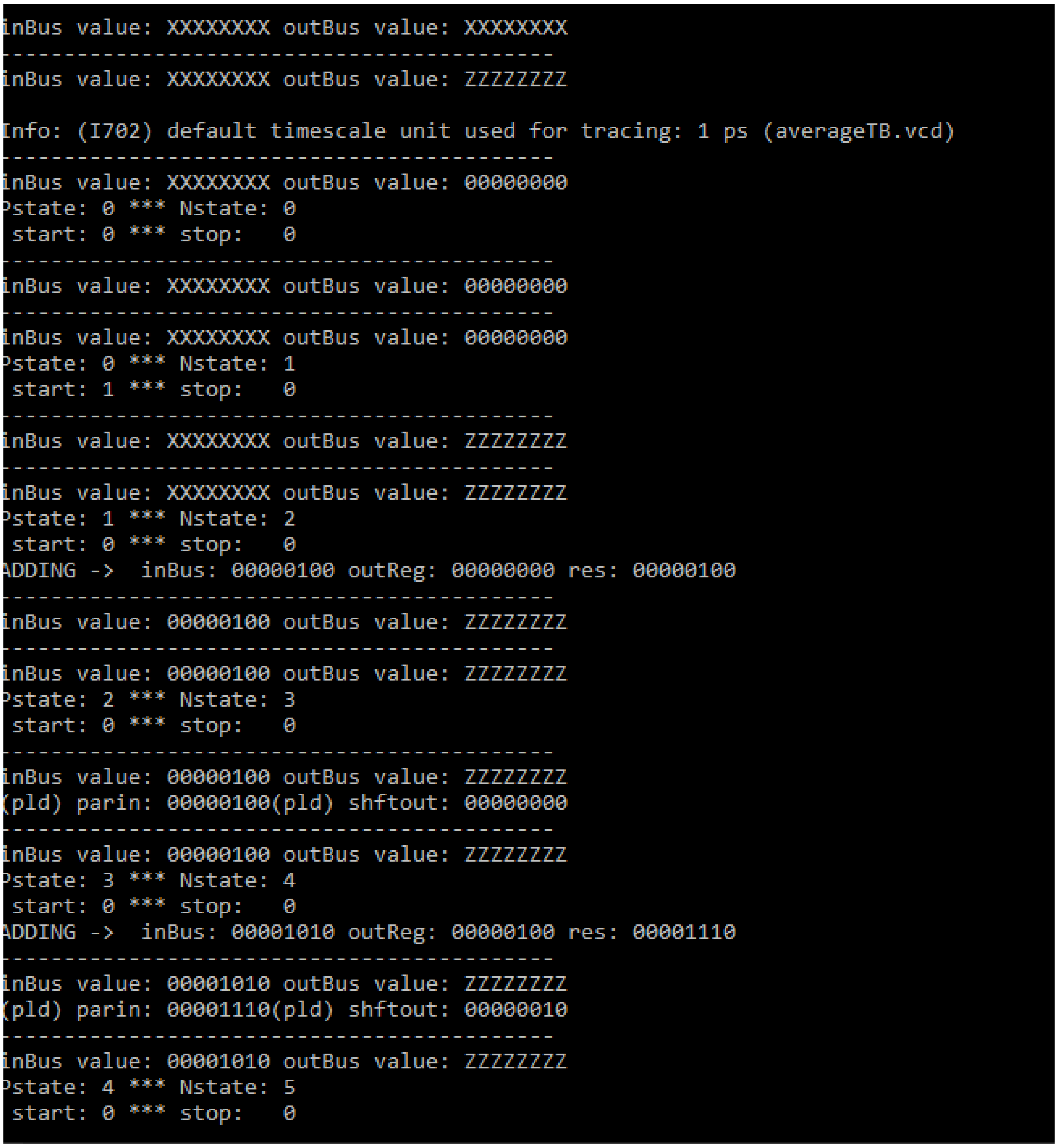
Top-Level:

```
#include "accAverage_CU.h"
#include "accAverage_DP.h"

SC_MODULE(accAverage_Top) {
      sc_in<sc_logic> rst, clk, start, stop;
      sc_in<sc_lv<8>> inBus;
      sc_out <sc_lv<8>> outBus;

      sc_signal<sc_logic> enableReg, selMux, selTri,
parallel_load_en, shift_en;

      accAverage_DP* myDatapath;
      accAverage_CU* myController;



      SC_CTOR(accAverage_Top) {
            myController = new accAverage_CU("controller");
            myController->clk(clk);
            myController->rst(rst);
            myController->start(start);
            myController->stop(stop);
            myController->enableReg(enableReg);
            myController->selMux(selMux);
            myController->selTri(selTri);
            myController->parallel_load_en(parallel_load_en);
            myController->shift_en(shift_en);


            myDatapath = new accAverage_DP("datapath");
            myDatapath->rst(rst);
            myDatapath->clk(clk);
            myDatapath->inBus(inBus);
            myDatapath->outBus(outBus);
            myDatapath->enableReg(enableReg);
            myDatapath->selMux(selMux);
            myDatapath->selTri(selTri);
            myDatapath->parallel_load_en(parallel_load_en);
            myDatapath->shift_en(shift_en);

      }
};
```

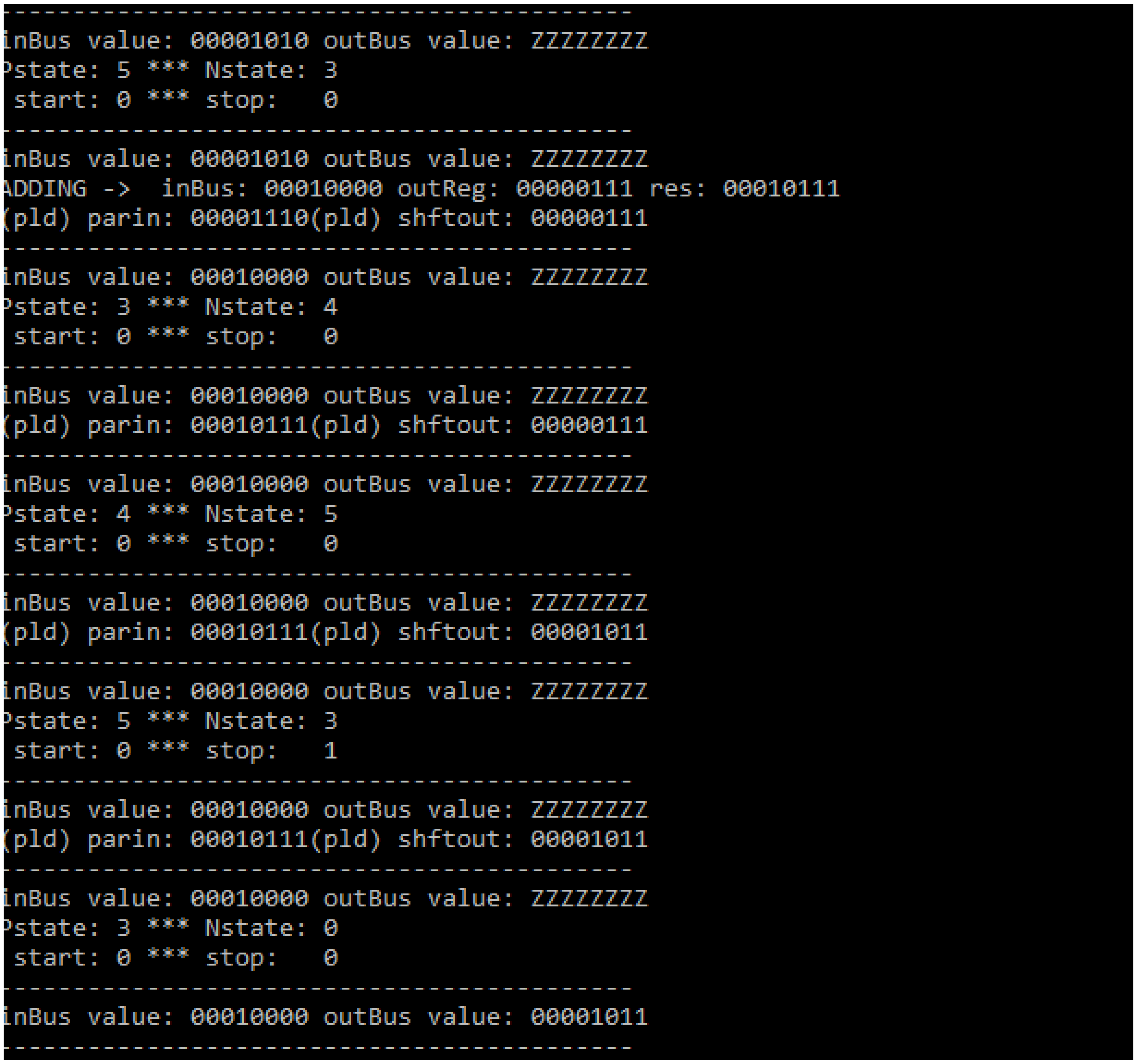
(code for testbench can be seen in the file)

Scenario 1: ((4+10)/2 + 16)/2

Result:



```
inBus value: XXXXXXXX outBus value: XXXXXXXX
-------------------------------------------
inBus value: XXXXXXXX outBus value: ZZZZZZZZ

Info: (I702) default timescale unit used for tracing: 1 ps (averageTB.vcd)
-------------------------------------------
inBus value: XXXXXXXX outBus value: 00000000
Pstate: 0 *** Nstate: 0
 start: 0 *** stop:   0
-------------------------------------------
inBus value: XXXXXXXX outBus value: 00000000
-------------------------------------------
inBus value: XXXXXXXX outBus value: 00000000
Pstate: 0 *** Nstate: 1
 start: 1 *** stop:   0
-------------------------------------------
inBus value: XXXXXXXX outBus value: ZZZZZZZZ
-------------------------------------------
inBus value: XXXXXXXX outBus value: ZZZZZZZZ
Pstate: 1 *** Nstate: 2
 start: 0 *** stop:   0
ADDING ->  inBus: 00000100 outReg: 00000000 res: 00000100
-------------------------------------------
inBus value: 00000100 outBus value: ZZZZZZZZ
-------------------------------------------
inBus value: 00000100 outBus value: ZZZZZZZZ
Pstate: 2 *** Nstate: 3
 start: 0 *** stop:   0
-------------------------------------------
inBus value: 00000100 outBus value: ZZZZZZZZ
(pld) parin: 00000100(pld) shftout: 00000000
-------------------------------------------
inBus value: 00000100 outBus value: ZZZZZZZZ
Pstate: 3 *** Nstate: 4
 start: 0 *** stop:   0
ADDING ->  inBus: 00001010 outReg: 00000100 res: 00001110
-------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
(pld) parin: 00001110(pld) shftout: 00000010
-------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
Pstate: 4 *** Nstate: 5
 start: 0 *** stop:   0
```



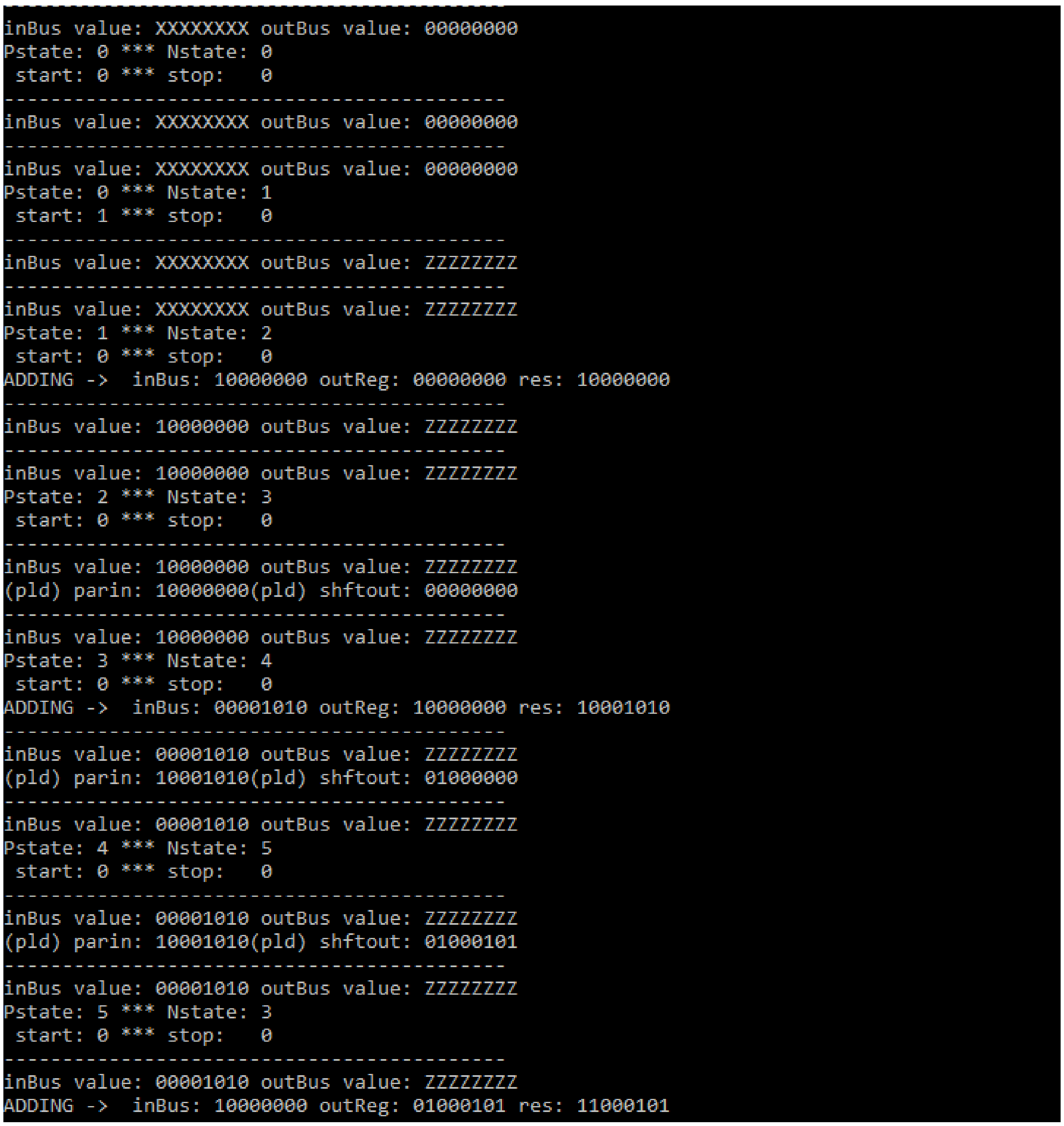
```
------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
Pstate: 5 *** Nstate: 3
 start: 0 *** stop:   0
------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
ADDING ->  inBus: 00010000 outReg: 00000111 res: 00010111
(pld) parin: 00001110(pld) shftout: 00000111
------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
Pstate: 3 *** Nstate: 4
 start: 0 *** stop:   0
------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
(pld) parin: 00010111(pld) shftout: 00000111
------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
Pstate: 4 *** Nstate: 5
 start: 0 *** stop:   0
------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
(pld) parin: 00010111(pld) shftout: 00001011
------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
Pstate: 5 *** Nstate: 3
 start: 0 *** stop:   1
------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
(pld) parin: 00010111(pld) shftout: 00001011
------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
Pstate: 3 *** Nstate: 0
 start: 0 *** stop:   0
------------------------------------------
inBus value: 00010000 outBus value: 00001011
------------------------------------------
```

Expected value: 11 (1011)

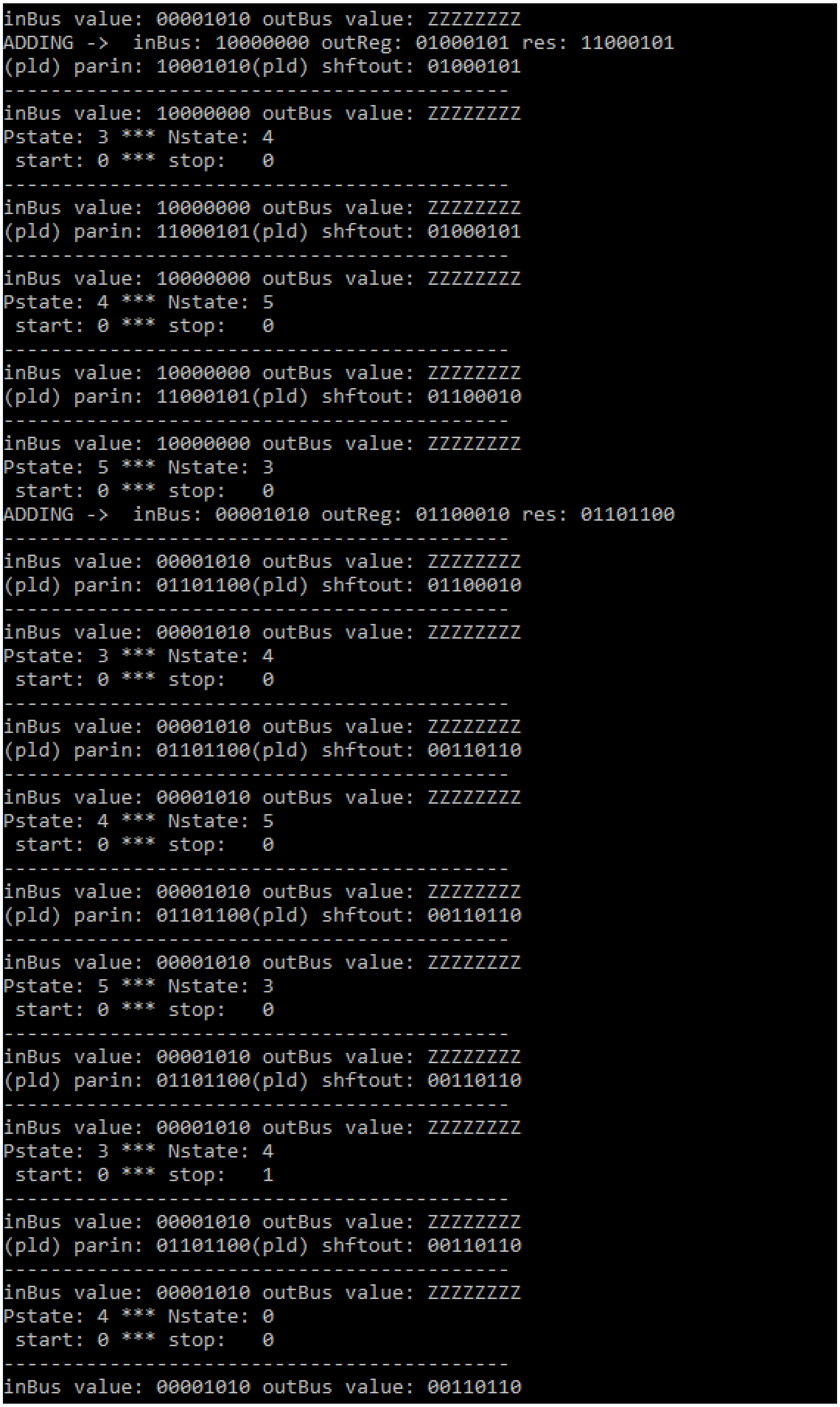
Actual value: 11 (1011)

Scenario 2: (((128 + 10)/2 + 128)/2 + 10)/2

Result:



```
inBus value: XXXXXXXX outBus value: 00000000
Pstate: 0 *** Nstate: 0
 start: 0 *** stop:   0
-------------------------------------------
inBus value: XXXXXXXX outBus value: 00000000
-------------------------------------------
inBus value: XXXXXXXX outBus value: 00000000
Pstate: 0 *** Nstate: 1
 start: 1 *** stop:   0
-------------------------------------------
inBus value: XXXXXXXX outBus value: ZZZZZZZZ
-------------------------------------------
inBus value: XXXXXXXX outBus value: ZZZZZZZZ
Pstate: 1 *** Nstate: 2
 start: 0 *** stop:   0
ADDING ->  inBus: 10000000 outReg: 00000000 res: 10000000
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
Pstate: 2 *** Nstate: 3
 start: 0 *** stop:   0
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
(pld) parin: 10000000(pld) shftout: 00000000
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
Pstate: 3 *** Nstate: 4
 start: 0 *** stop:   0
ADDING ->  inBus: 00001010 outReg: 10000000 res: 10001010
-------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
(pld) parin: 10001010(pld) shftout: 01000000
-------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
Pstate: 4 *** Nstate: 5
 start: 0 *** stop:   0
-------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
(pld) parin: 10001010(pld) shftout: 01000101
-------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
Pstate: 5 *** Nstate: 3
 start: 0 *** stop:   0
-------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
ADDING ->  inBus: 10000000 outReg: 01000101 res: 11000101
```



```
inBus value: 00001010 outBus value: ZZZZZZZZ
ADDING ->  inBus: 10000000 outReg: 01000101 res: 11000101
(pld) parin: 10001010(pld) shftout: 01000101
--------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
Pstate: 3 *** Nstate: 4
 start: 0 *** stop:   0
--------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
(pld) parin: 11000101(pld) shftout: 01000101
--------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
Pstate: 4 *** Nstate: 5
 start: 0 *** stop:   0
--------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
(pld) parin: 11000101(pld) shftout: 01100010
--------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
Pstate: 5 *** Nstate: 3
 start: 0 *** stop:   0
ADDING ->  inBus: 00001010 outReg: 01100010 res: 01101100
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
(pld) parin: 01101100(pld) shftout: 01100010
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
Pstate: 3 *** Nstate: 4
 start: 0 *** stop:   0
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
(pld) parin: 01101100(pld) shftout: 00110110
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
Pstate: 4 *** Nstate: 5
 start: 0 *** stop:   0
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
(pld) parin: 01101100(pld) shftout: 00110110
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
Pstate: 5 *** Nstate: 3
 start: 0 *** stop:   0
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
(pld) parin: 01101100(pld) shftout: 00110110
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
Pstate: 3 *** Nstate: 4
 start: 0 *** stop:   1
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
(pld) parin: 01101100(pld) shftout: 00110110
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
Pstate: 4 *** Nstate: 0
 start: 0 *** stop:   0
--------------------------------------------
inBus value: 00001010 outBus value: 00110110
```

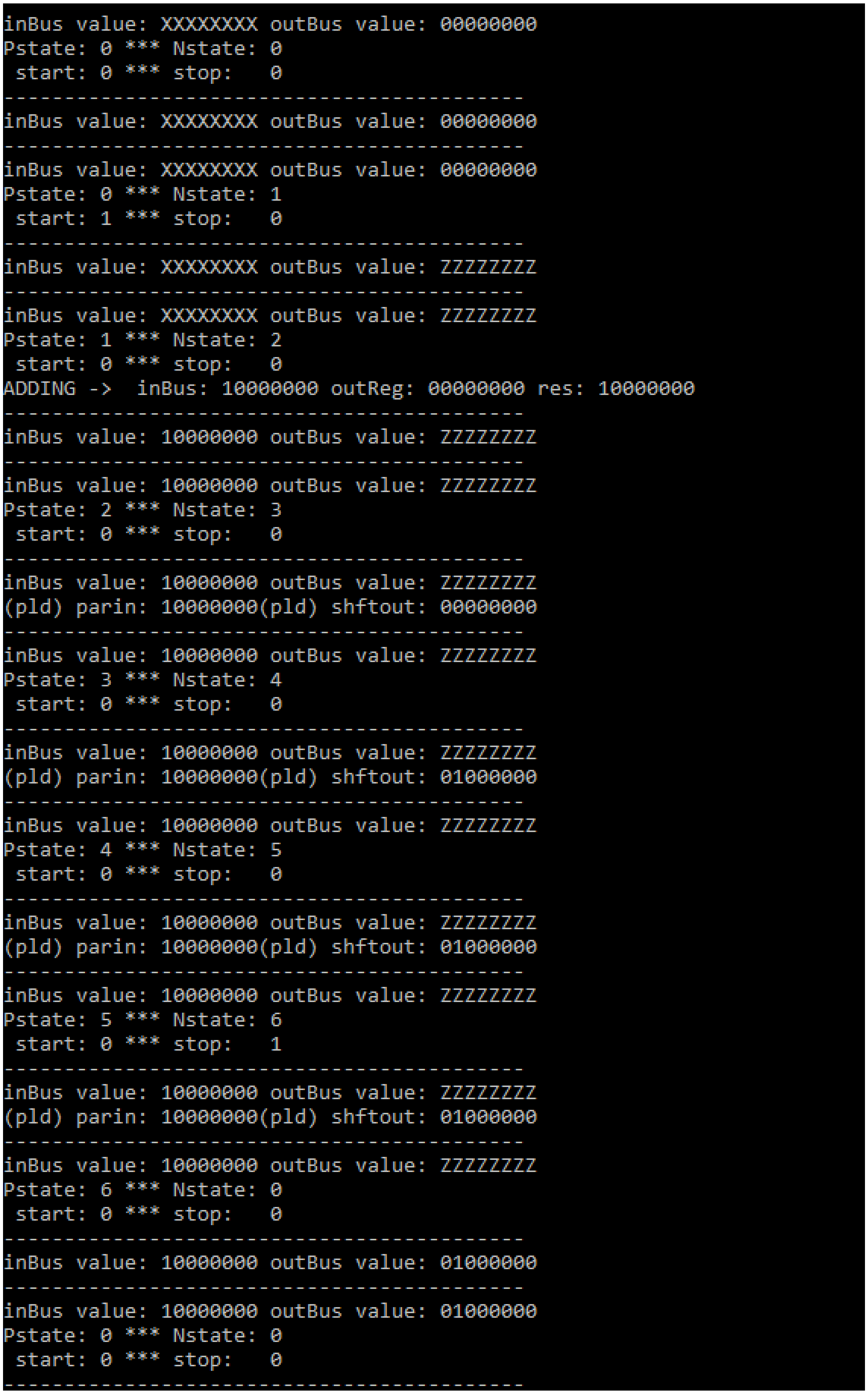
Expected value: 54 (110110)

Actual value: 54 (110110)

This test expands on the previous one by adding another bus input.

Scenario 3: (128 + 0)/2 - only one bus input

Results:



```
inBus value: XXXXXXXX outBus value: 00000000
Pstate: 0 *** Nstate: 0
 start: 0 *** stop:   0
-------------------------------------------
inBus value: XXXXXXXX outBus value: 00000000
-------------------------------------------
inBus value: XXXXXXXX outBus value: 00000000
Pstate: 0 *** Nstate: 1
 start: 1 *** stop:   0
-------------------------------------------
inBus value: XXXXXXXX outBus value: ZZZZZZZZ
-------------------------------------------
inBus value: XXXXXXXX outBus value: ZZZZZZZZ
Pstate: 1 *** Nstate: 2
 start: 0 *** stop:   0
ADDING ->  inBus: 10000000 outReg: 00000000 res: 10000000
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
Pstate: 2 *** Nstate: 3
 start: 0 *** stop:   0
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
(pld) parin: 10000000(pld) shftout: 00000000
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
Pstate: 3 *** Nstate: 4
 start: 0 *** stop:   0
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
(pld) parin: 10000000(pld) shftout: 01000000
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
Pstate: 4 *** Nstate: 5
 start: 0 *** stop:   0
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
(pld) parin: 10000000(pld) shftout: 01000000
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
Pstate: 5 *** Nstate: 6
 start: 0 *** stop:   1
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
(pld) parin: 10000000(pld) shftout: 01000000
-------------------------------------------
inBus value: 10000000 outBus value: ZZZZZZZZ
Pstate: 6 *** Nstate: 0
 start: 0 *** stop:   0
-------------------------------------------
inBus value: 10000000 outBus value: 01000000
-------------------------------------------
inBus value: 10000000 outBus value: 01000000
Pstate: 0 *** Nstate: 0
 start: 0 *** stop:   0
-------------------------------------------
```

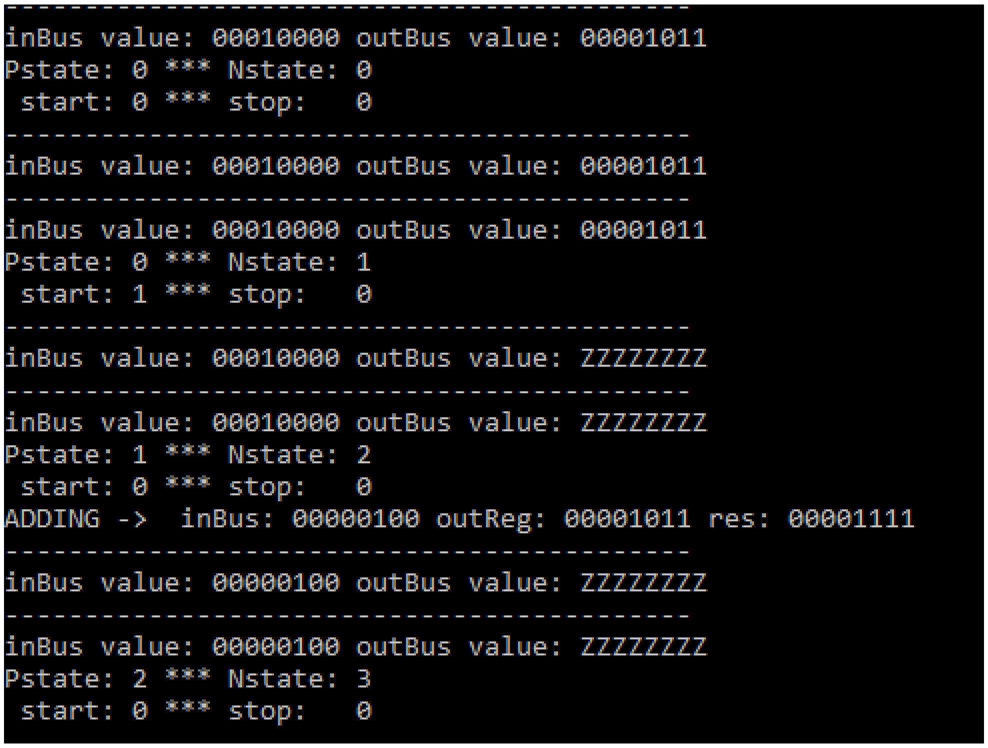
Expected value: 64 (01000000)

Actual value: 64 (01000000)

This test shows that the circuit can handle doing an average even if there is only one bus value.

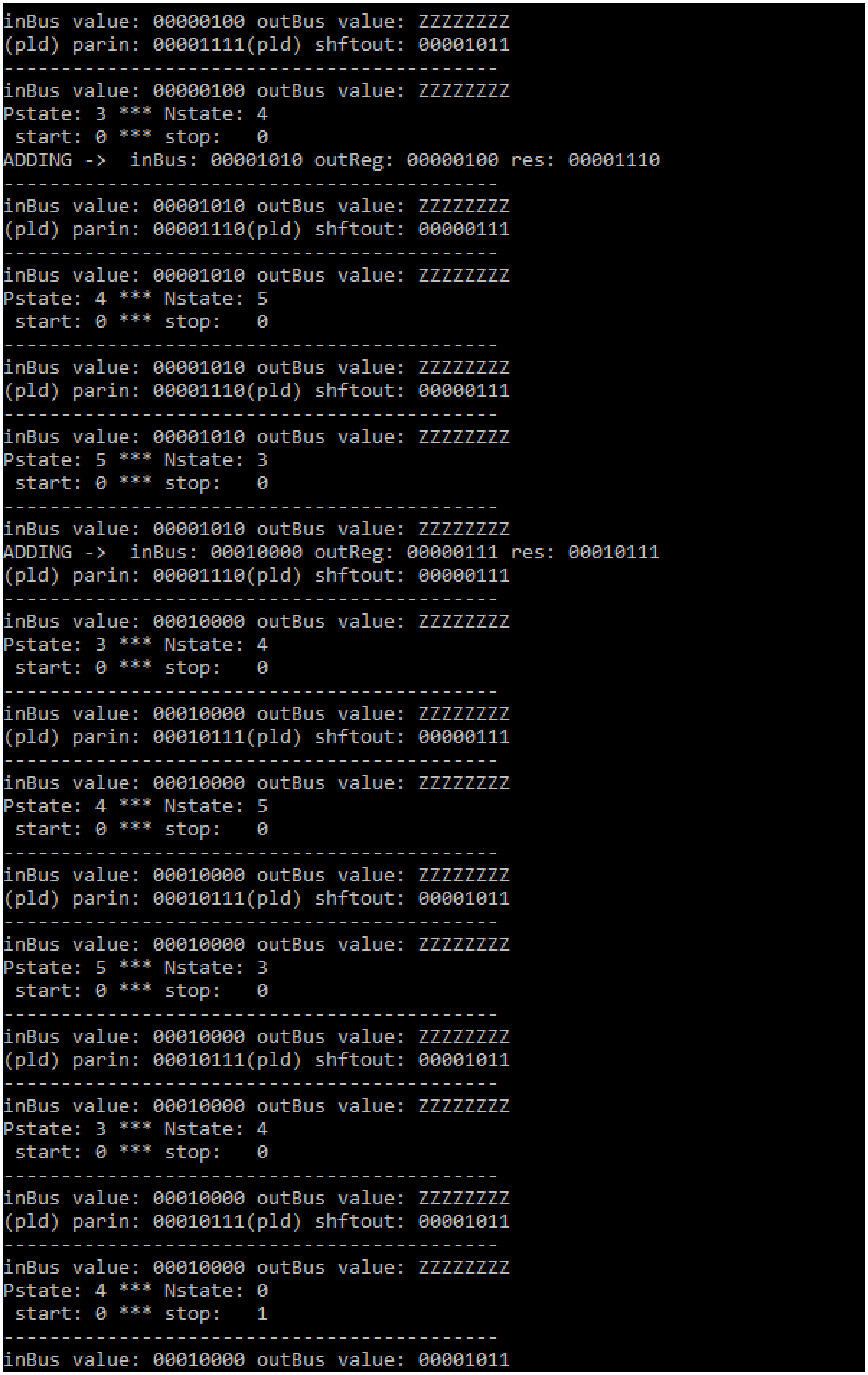
Scenario 4: ((4+10)/2 + 16)/2 after stopping

Results:



```
-------------------------------------------
inBus value: 00010000 outBus value: 00001011
Pstate: 0 *** Nstate: 0
 start: 0 *** stop:   0
-------------------------------------------
inBus value: 00010000 outBus value: 00001011
-------------------------------------------
inBus value: 00010000 outBus value: 00001011
Pstate: 0 *** Nstate: 1
 start: 1 *** stop:   0
-------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
-------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
Pstate: 1 *** Nstate: 2
 start: 0 *** stop:   0
ADDING ->  inBus: 00000100 outReg: 00001011 res: 00001111
-------------------------------------------
inBus value: 00000100 outBus value: ZZZZZZZZ
-------------------------------------------
inBus value: 00000100 outBus value: ZZZZZZZZ
Pstate: 2 *** Nstate: 3
 start: 0 *** stop:   0
```

Start pulse turns off the tri-state again, making the output unknown.



```
inBus value: 00000100 outBus value: ZZZZZZZZ
(pld) parin: 00001111(pld) shftout: 00001011
--------------------------------------------
inBus value: 00000100 outBus value: ZZZZZZZZ
Pstate: 3 *** Nstate: 4
 start: 0 *** stop:   0
ADDING ->  inBus: 00001010 outReg: 00000100 res: 00001110
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
(pld) parin: 00001110(pld) shftout: 00000111
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
Pstate: 4 *** Nstate: 5
 start: 0 *** stop:   0
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
(pld) parin: 00001110(pld) shftout: 00000111
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
Pstate: 5 *** Nstate: 3
 start: 0 *** stop:   0
--------------------------------------------
inBus value: 00001010 outBus value: ZZZZZZZZ
ADDING ->  inBus: 00010000 outReg: 00000111 res: 00010111
(pld) parin: 00001110(pld) shftout: 00000111
--------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
Pstate: 3 *** Nstate: 4
 start: 0 *** stop:   0
--------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
(pld) parin: 00010111(pld) shftout: 00000111
--------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
Pstate: 4 *** Nstate: 5
 start: 0 *** stop:   0
--------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
(pld) parin: 00010111(pld) shftout: 00001011
--------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
Pstate: 5 *** Nstate: 3
 start: 0 *** stop:   0
--------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
(pld) parin: 00010111(pld) shftout: 00001011
--------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
Pstate: 3 *** Nstate: 4
 start: 0 *** stop:   0
--------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
(pld) parin: 00010111(pld) shftout: 00001011
--------------------------------------------
inBus value: 00010000 outBus value: ZZZZZZZZ
Pstate: 4 *** Nstate: 0
 start: 0 *** stop:   1
--------------------------------------------
inBus value: 00010000 outBus value: 00001011
```

Expected value: 11 (1011)

Actual value: 11 (1011)

After the stop pulse, another start and stop pulse returns the same values for the output.

D.) Bus Functional Model and Testbench

BFM Code:

```
#include <systemc.h>
#include <iostream>


template <int NumClk>
SC_MODULE(averageBFM) {
      sc_in<sc_logic> rst, clk, start, stop;
      sc_in<sc_lv<8>> inBus;
      sc_out <sc_lv<8>> outBus;
      sc_lv<8> currHold;
      enum state_types {INITIALIZE, WAIT_ON_START, STARTED, ACTIVE,
ACTIVE2, ACTIVE3, WAIT_ON_END};
      sc_signal<state_types> Pstate, Nstate;
      sc_lv<8> currInBus;

      SC_CTOR(averageBFM) {
            SC_THREAD(operation);
            sensitive << clk << rst;
            SC_THREAD(averaging);
            sensitive << inBus;
            SC_THREAD(stateReset);
            sensitive << Nstate << Pstate;
            SC_THREAD(stateChanges);
            sensitive << start << stop << Pstate;
      }
      void operation();
      void averaging();
```

```
    void stateReset();
    void stateChanges();
};

template <int NumClk>
void averageBFM<NumClk>::stateChanges() {
    while(true){
        switch (Pstate) {
        case INITIALIZE:
            if (start == '0') Nstate = INITIALIZE;
            else Nstate = WAIT_ON_START;
            break;

        case WAIT_ON_START:
            if (start == '1') Nstate = WAIT_ON_START;
            else Nstate = STARTED;
            break;

        case STARTED:
            Nstate = ACTIVE;
            break;

        case ACTIVE:
            if (stop == '0') Nstate = ACTIVE2;
            else Nstate = WAIT_ON_END;
            break;

        case ACTIVE2:
            if (stop == '0') Nstate = ACTIVE3;
            else Nstate = WAIT_ON_END;
            break;

        case ACTIVE3:
            if (stop == '0') Nstate = ACTIVE;
            else Nstate = WAIT_ON_END;
            break;

        case WAIT_ON_END:
            if (stop == '1') Nstate = WAIT_ON_END;
```

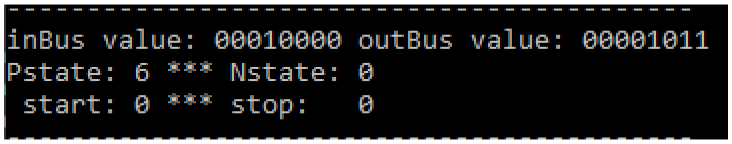
```cpp
            else Nstate = INITIALIZE;
            break;

        default:
            Nstate = INITIALIZE;
            break;
        }
        wait();
    }
}

template <int NumClk>
void averageBFM<NumClk>::stateReset() {
    while (true) {
        if (Pstate == INITIALIZE) {
            outBus = currHold;
        }
        else {
            outBus = "XXXXXXXX";
        }
        if (Pstate == WAIT_ON_START) {
            currHold = "00000000";
        }
        wait();
    }
}

template <int NumClk>
void averageBFM<NumClk>::averaging() {
    while (true){
        if (Pstate != INITIALIZE && Pstate != WAIT_ON_START /*&&
Pstate != STARTED*/) {
            cout << "currHold: " << currHold << "inBus read: "
<< inBus << endl;
            if (currHold != "00000000") {
                currHold = (inBus->read().to_uint() +
currHold.to_uint()) / 2;
                cout << "average: " << currHold << endl;
            }
            else {
                currHold = (inBus->read().to_uint());

            }
        }
        wait();
    }

}

template <int NumClk>
void averageBFM<NumClk>::operation() {

    while (true) {
        if (rst == '1') {
            currHold = "00000000";
        }
        else if (clk->event() && clk == '1') {
            for (int i = 0; i < NumClk; i++) {
                wait(clk->posedge_event());

                Pstate = Nstate;

                cout << "Pstate: " << Pstate << " *** Nstate: "
<< Nstate << endl;
                cout << " start: " << start << " *** stop:    "
<< stop << endl;
                cout << "average: " << currHold << endl;
            }
        }
        wait();
    }
}
```

Since we could not have the adder be sensitive to changes in the inBus, we made a separate thread for the operation. Same thing goes with enabling the register which caused us to create the stateReset thread. Overall, our BFM strongly resembles the controller unit.

The following are the results of the same tests being performed on the BFM model:

Scenario 1: ((4+10)/2 + 16)/2

Results:



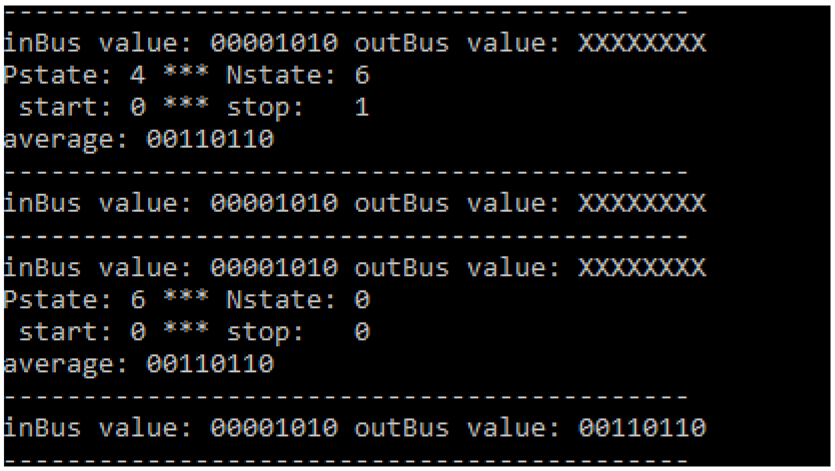
```
------------------------------------------
inBus value: 00010000 outBus value: 00001011
Pstate: 6 *** Nstate: 0
 start: 0 *** stop:   0
------------------------------------------
```

Expected value: 11 (1011)

Actual value: 11 (1011)

Scenario 2: (((128 + 10)/2 + 128)/2 + 10)/2

Results:



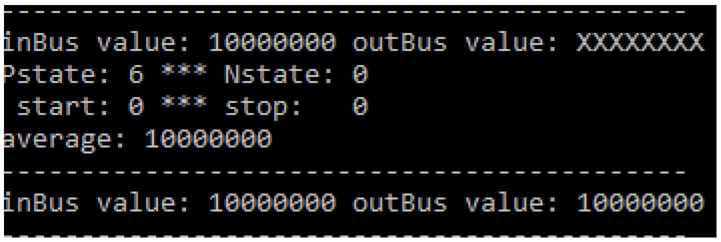
```
------------------------------------------
inBus value: 00001010 outBus value: XXXXXXXX
Pstate: 4 *** Nstate: 6
 start: 0 *** stop:   1
average: 00110110
------------------------------------------
inBus value: 00001010 outBus value: XXXXXXXX
------------------------------------------
inBus value: 00001010 outBus value: XXXXXXXX
Pstate: 6 *** Nstate: 0
 start: 0 *** stop:   0
average: 00110110
------------------------------------------
inBus value: 00001010 outBus value: 00110110
------------------------------------------
```

Expected value: 54 (110110)

Actual value: 54 (110110)

Scenario 3: (128 + 0)/2 - only one bus input

Results:



```
-------------------------------------------
inBus value: 10000000 outBus value: XXXXXXXX
Pstate: 6 *** Nstate: 0
 start: 0 *** stop:   0
average: 10000000
-------------------------------------------
inBus value: 10000000 outBus value: 10000000
-------------------------------------------
```

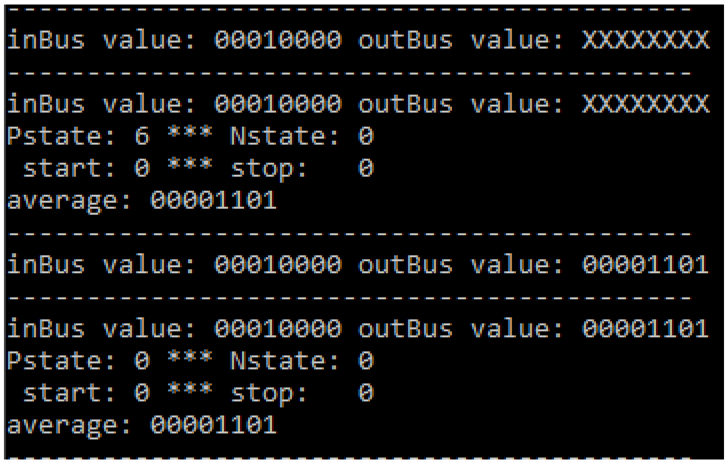
Expected value: 64 (01000000)

Actual value: 128 (10000000)

The BFM model does not average by itself if it does not have another value in the held in the average variable.

Scenario 4: ((4+10)/2 + 16)/2 after stopping

Results:



```
-------------------------------------------
inBus value: 00010000 outBus value: XXXXXXXX
-------------------------------------------
inBus value: 00010000 outBus value: XXXXXXXX
Pstate: 6 *** Nstate: 0
 start: 0 *** stop:   0
average: 00001101
-------------------------------------------
inBus value: 00010000 outBus value: 00001101
-------------------------------------------
inBus value: 00010000 outBus value: 00001101
Pstate: 0 *** Nstate: 0
 start: 0 *** stop:   0
average: 00001101
-------------------------------------------
```

Expected value: 11 (00001011)

Actual value: 13 (00001101)

The BFM model does not automatically reset the held average by itself. It will only do so if it gets a reset pulse.