ELEC6233 – SystemC Assignment

Karthik Sathyanarayanan
Ks6n19
MSc Embedded systems
tmak

ABSTRACT: *The following report aims to complete the objectives of the* SystemC assignment.

1. 1. Design 1

Requirement 1a:

The full adder uses a Adder.h header file to describe the hardware level functionality in the design. The AdderMain.cpp is the file that gets compiled in Questa Sim. This file declares the signals tied to the modules and instantates the testbench and the Adder.h header files. The Testbench.h gives in the input combinations 000, 001, 010, 011,100, 101, 110 and 111. The design is then simulated to verify the correctness of the design. Sccom -link was used in the command line of questa Sim to link the header modules with the main code.

Requirement 1b:

The modified testbench to validate the full adder design shows some erroneous behaviour. The waveform Figure 2b shows the original uncorrected design. We can see that the error signal is high between 40 to 60 ns and then it is high during 70-80 bs. The input combinations for the error prone cycles are:

A_s	*B_s*	*CIN_s*
0	*1*	*0*
0	*1*	*1*
1	*1*	*0*

:

This error happens due to the wrong code written in Adder.h header file that explains the circuit functionality of temporary signal E. The code shows the temporary signal E as Cin OR D. The Adder circuit functionality is shown below:

C = ‘A’ AND ‘B‘
D = ‘A’ XOR ‘B’
E = ‘Cin’ OR ‘D’
Sum = ‘Cin’ XOR ‘D’
Cout = ‘C’ OR ‘E’

This design is then corrected by using:

tempE = Cin.read() & tempD;

This corrected design is again simulated and the waveform is shown in figure 1b. We can see from the waveform that the error signal is zero throughout the operation.

1. 2. Design 2

Requirement 2a:

The counter uses a Counter.h header file to describe a high-level functionality of the counter. The circuit functionality is translated to hardware by systemC. The software level algorithm can be explained as:

```
 if enable is true the hardware is switched on
         if reset is read, count takes the value of zero
         else count gets incremented and counter_out readsthe incremented value
```

The counter_out is an 8 bit unsigned integer. This port reads out the counter incremented

value. The reset is a negative edge activated reset. The CounterMain.cpp file instantates the counter.h and the tesbench2.h files. The testbench is a simple testbench that simulates the counter output for various combinations of reset and enable with a 10ns gap between them. The waveform shows just the simulation for first 34ns. The counter does seem to work in the desired fashion and increments when asked to do so. Sccom -link was used in the command line of questa Sim to link the header modules with the main code.
<u>Requirement 2b:</u>
This part of the design aims to integrate design 2a with 1b. The design aims to keep the counter on during the error prone input combinations. In this design, the testbench from adder is modified to include the elements of counter testbench in it. The testbench is designed in a fashion that the counter remains disabled as long as the error signal is zero. The counter is enabled and reset every time an error signal has a true output. The counter increments during the error prone input combinations and resets. This design however does not fully complete the requirement 2b. In future, the modifications to the design such that a proper control flow chart so that the counter does not increment with the next clock edge and a better solution for the reset every time an error combination is there in the input would be appreciated.

Appendix

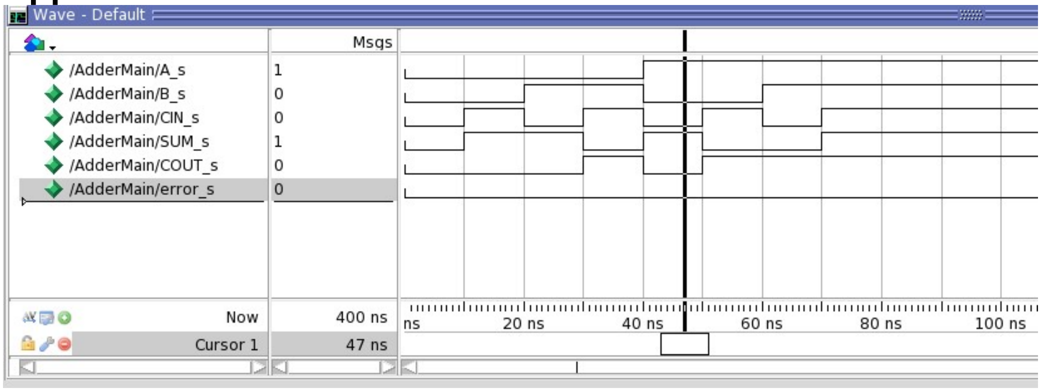

Figure 1b: Waveform of corrected design.

<u>Code for Design 1</u>

```
/////////////////////////////////////////////////////////////////////////////////
// Description: Header file for Adder uncorrected
// Author: ks6n19
// Version: 1.0
// Date: 20/May/2020
/////////////////////////////////////////////////////////////////////////////////
#include "systemc.h"
SC_MODULE(Adder){
        sc_out<sc_logic> sum, Cout;
        sc_in<sc_logic> A, B, Cin;

        void add(){
                sc_logic tempC, tempD, tempE;
                tempC = A.read() & B.read();
                tempD = A.read() ^ B.read();
                tempE = Cin.read() | tempD; // AND gate instead of OR gate for corrected
waveform
                sum.write(tempD ^ Cin.read());
                Cout.write(tempC | tempE);
        }
```

```
    SC_CTOR(Adder){
        SC_METHOD(add);
        sensitive << A << B << Cin;
    }
};
////////////////////////////////////////////////////////////////////////////////
// Description: Main for Adder
// Author: ks6n19
// Version: 1.0
// Date: 20/May/2020
////////////////////////////////////////////////////////////////////////////////
#include "Adder.h"
#include "Testbench.h"

SC_MODULE(AdderMain){
    //Declare signals to be tied to the modules
    sc_signal<sc_logic> A_s,B_s,CIN_s;
    sc_signal<sc_logic> SUM_s,COUT_s;

    Adder adder1;
    Testbench test1;

     sc_signal<sc_logic>error_s;

    //Instantiate Adder and Bind Ports
    SC_CTOR(AdderMain): adder1("Adder"), test1("TestBench"){
      adder1.A(A_s);
      adder1.B(B_s);
      adder1.Cin(CIN_s);
      adder1.sum(SUM_s);
      adder1.Cout(COUT_s);

      //Instantiate Testbench And Bind Ports
      test1.TA(A_s);
      test1.TB(B_s);
      test1.TCin(CIN_s);
       test1.TSum(SUM_s);
       test1.TCout(COUT_s);
       test1.error(error_s);
    }
};

SC_MODULE_EXPORT(AdderMain);
////////////////////////////////////////////////////////////////////////////////
// Description: Testbench for adder
// Author: ks6n19
// Version: 1.0
// Date: 20/May/2020
////////////////////////////////////////////////////////////////////////////////
#include "systemc.h" //Testbench.h
SC_MODULE(Testbench){
    sc_out<sc_logic> TA, TB, TCin; //changes made here
     sc_out<sc_logic>error;
     sc_out<sc_logic>TSum,TCout;   //changes made here
```

```
void testprocess(){
            error.write(SC_LOGIC_0);

            //combination 000
          TA.write(SC_LOGIC_0);
          TB.write(SC_LOGIC_0);
          TCin.write(SC_LOGIC_0);
          wait(10, SC_NS);
            if(TSum.read()==SC_LOGIC_0&&TCout.read()==SC_LOGIC_0)
            error.write(SC_LOGIC_0);
            else
               error.write(SC_LOGIC_1);

             //combination 001
          TA.write(SC_LOGIC_0);
          TB.write(SC_LOGIC_0);
          TCin.write(SC_LOGIC_1);
          wait(10, SC_NS);
            if(TSum.read()==SC_LOGIC_1&&TCout.read()==SC_LOGIC_0)
            error.write(SC_LOGIC_0);
            else
               error.write(SC_LOGIC_1);

            //combination 010
          TA.write(SC_LOGIC_0);
          TB.write(SC_LOGIC_1);
          TCin.write(SC_LOGIC_0);
          wait(10, SC_NS);
            if(TSum.read()==SC_LOGIC_1&&TCout.read()==SC_LOGIC_0)
            error.write(SC_LOGIC_0);
            else
               error.write(SC_LOGIC_1);

            //combination 011
          TA.write(SC_LOGIC_0);
          TB.write(SC_LOGIC_1);
          TCin.write(SC_LOGIC_1);
          wait(10, SC_NS);
            if(TSum.read()==SC_LOGIC_0&&TCout.read()==SC_LOGIC_1)
            error.write(SC_LOGIC_0);
            else
               error.write(SC_LOGIC_1);

            //combination 100
          TA.write(SC_LOGIC_1);
          TB.write(SC_LOGIC_0);
          TCin.write(SC_LOGIC_0);
          wait(10, SC_NS);
            if(TSum.read()==SC_LOGIC_1&&TCout.read()==SC_LOGIC_0)
            error.write(SC_LOGIC_0);
            else
               error.write(SC_LOGIC_1);
```

```
                  //combination 101
                 TA.write(SC_LOGIC_1);
                 TB.write(SC_LOGIC_0);
                 TCin.write(SC_LOGIC_1);
                 wait(10, SC_NS);
                  if(TSum.read()==SC_LOGIC_0&&TCout.read()==SC_LOGIC_1)
                  error.write(SC_LOGIC_0);
                  else
                     error.write(SC_LOGIC_1);

                  //combination 110
                 TA.write(SC_LOGIC_1);
                 TB.write(SC_LOGIC_1);
                 TCin.write(SC_LOGIC_0);
                 wait(10, SC_NS);
                  if(TSum.read()==SC_LOGIC_0&&TCout.read()==SC_LOGIC_1)
                  error.write(SC_LOGIC_0);
                  else
                     error.write(SC_LOGIC_1);


                 //combination 111
                 TA.write(SC_LOGIC_1);
                 TB.write(SC_LOGIC_1);
                 TCin.write(SC_LOGIC_1);
                 wait(10, SC_NS);
                  if(TSum.read()==SC_LOGIC_1&&TCout.read()==SC_LOGIC_1)
                  error.write(SC_LOGIC_0);
                  else
                     error.write(SC_LOGIC_1);
        }
        SC_CTOR(Testbench){
                 SC_THREAD(testprocess);
        }
};
```

Code for Design 2

```
///////////////////////////////////////////////////////////////////////////////
// Description: Header file for Counter
// Author: ks6n19
// Version: 1.0
// Date: 20/May/2020
///////////////////////////////////////////////////////////////////////////////

#include "systemc.h"
SC_MODULE (Counter8bit) {

sc_in_clk clock; //clock
sc_in<bool> reset;
sc_in<bool> enable;
sc_out<sc_uint<8> >counter_out; //8bit

sc_uint<8> count; //8bit
```

```
void incr_count(){
while (true){
        if (enable.read()) {
           if (reset.read()){
              count = 0;
              counter_out.write(count);
              } else {
                       count= count+1;
                       counter_out.write(count);
                       }
           }
           wait();
        }
};


SC_CTOR(Counter8bit) {
         SC_THREAD(incr_count);
         sensitive << clock.pos();
         count = 0;
   } // End of Constructor


};

//////////////////////////////////////////////////////////////////////////////
// Description: Main file for Counter
// Author: ks6n19
// Version: 1.0
// Date: 20/May/2020
//////////////////////////////////////////////////////////////////////////////
#include "systemc.h"
#include "Counter8bit.h"
#include "TestBenchDesign2.h"

SC_MODULE(CounterMain){
         sc_clock clock;
         //sc_signal<bool> clock;
         sc_signal<bool> reset;
         sc_signal<bool> enable;
         sc_signal<sc_uint<8> >counter_out;

Counter8bit counter;
TestBenchDesign2 test1;

SC_CTOR(CounterMain): clock("SystemClock", 2, 0.5, true),
counter("Counter1"),test1("Test1"){
              counter.reset(reset);
              counter.enable(enable);
              counter.clock(clock);
              counter.counter_out(counter_out);

              test1.reset(reset);
              test1.enable(enable);
}

};
SC_MODULE_EXPORT(CounterMain);

///////////////////////////////////////////////////////////////////////////////
// Description: Testbench for counter
// Author: ks6n19
// Version: 1.0
// Date: 20/May/2020
///////////////////////////////////////////////////////////////////////////////
#include "systemc.h"

SC_MODULE(TestBenchDesign2){
            sc_out<bool>reset;
            sc_out<bool>enable;
            //sc_in<bool>clock;

void testprocess(){
      enable.write(false);
      reset.write(false);
      wait(10,SC_NS);

      enable.write(true);
      reset.write(true);
      wait(10,SC_NS);

      enable.write(true);
      reset.write(false);
      wait(10,SC_NS);

      enable.write(false);
      reset.write(true);
      wait(10,SC_NS);

      enable.write(false);
      reset.write(false);
      wait(10,SC_NS);
      }

SC_CTOR(TestBenchDesign2){
         SC_THREAD(testprocess);
         //sensitive << clock.pos();
         }
};
```

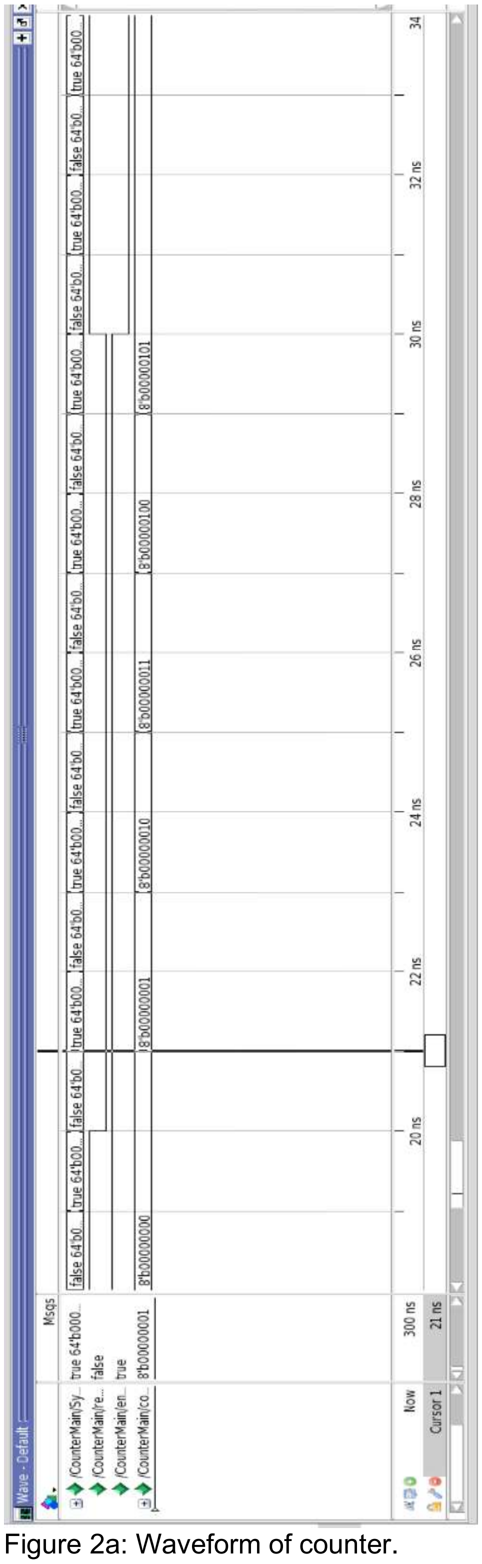
Wave - Default
Msgs
/CounterMain/Sy... true 64'b000...
/CounterMain/re... false
/CounterMain/en... true
/CounterMain/co... 8'b00000001
Now 300 ns
Cursor 1 21 ns
false 64'b0... true 64'b00... false 64'b0... true 64'b00... false 64'b0... true 64'b00... false 64'b0... true 64'b00... false 64'b0... true 64'b00... false 64'b0... true 64'b00... false 64'b0... true 64'b00... false 64'b0... true 64'b00...
8'b00000000
8'b00000001
8'b00000010
8'b00000011
8'b00000100
8'b00000101
20 ns
22 ns
24 ns
26 ns
28 ns
30 ns
32 ns
34

Figure 2a: Waveform of counter.

Code for Design 2b

```
///////////////////////////////////////////////////////////////////////////////////
// Description: Testbench for integrated adder and counter
// Author: ks6n19
// Version: 1.0
// Date: 20/May/2020
///////////////////////////////////////////////////////////////////////////////////
#include "systemc.h" //Testbench.h
SC_MODULE(Testbench){
        sc_out<sc_logic> TA, TB, TCin; //changes made here
         sc_out<sc_logic>error;
         sc_out<sc_logic>TSum,TCout;   //changes made here
                 sc_out<bool>reset;   // make changes in CounterMain.cpp to instantate this
instead of testbench 2
          sc_out<bool>enable;

        void testprocess(){
                 error.write(SC_LOGIC_0);
         enable.write(false);
     reset.write(false);
     wait(10,SC_NS);

     enable.write(true);
     reset.write(true);
     wait(10,SC_NS);

                 //combination 000
                TA.write(SC_LOGIC_0);
                TB.write(SC_LOGIC_0);
                TCin.write(SC_LOGIC_0);
                wait(10, SC_NS);
                 if(TSum.read()==SC_LOGIC_0&&TCout.read()==SC_LOGIC_0)
                 error.write(SC_LOGIC_0);
                            enable.write(false);          // counter does not run if error is 0
                            reset.write(true);
                 wait(10,SC_NS);
                 else
                    error.write(SC_LOGIC_1);
                              enable.write(true);   // counter is reset   if error is 1
                                  reset.write(false);

                  //combination 001
                TA.write(SC_LOGIC_0);
                TB.write(SC_LOGIC_0);
                TCin.write(SC_LOGIC_1);
                wait(10, SC_NS);
                 if(TSum.read()==SC_LOGIC_1&&TCout.read()==SC_LOGIC_0)
                 error.write(SC_LOGIC_0);
                            enable.write(false);          // counter does not run if error is 0
                            reset.write(true);
                 else
                    error.write(SC_LOGIC_1);
                              enable.write(true);   // counter is reset if error is 1
                    reset.write(false);

 //combination 010
TA.write(SC_LOGIC_0);
TB.write(SC_LOGIC_1);
TCin.write(SC_LOGIC_0);
wait(10, SC_NS);
 if(TSum.read()==SC_LOGIC_1&&TCout.read()==SC_LOGIC_0)
 error.write(SC_LOGIC_0);
              enable.write(false);          // counter does not run if error is 0
              reset.write(true);
 else
   error.write(SC_LOGIC_1);
                enable.write(true);   // counter is reset if error is 1
                    reset.write(false);

 //combination 011
TA.write(SC_LOGIC_0);
TB.write(SC_LOGIC_1);
TCin.write(SC_LOGIC_1);
wait(10, SC_NS);
 if(TSum.read()==SC_LOGIC_0&&TCout.read()==SC_LOGIC_1)
 error.write(SC_LOGIC_0);
              enable.write(false);          // counter does not   run if error is 0
              reset.write(true);

 else
   error.write(SC_LOGIC_1);
                enable.write(true);   // counter is reset if error is 1
                    reset.write(false);

 //combination 100
TA.write(SC_LOGIC_1);
TB.write(SC_LOGIC_0);
TCin.write(SC_LOGIC_0);
wait(10, SC_NS);
 if(TSum.read()==SC_LOGIC_1&&TCout.read()==SC_LOGIC_0)
 error.write(SC_LOGIC_0);
              enable.write(false);          // counter does not runs if error is 0
              reset.write(true);

else
   error.write(SC_LOGIC_1);
                enable.write(true);   // counter is reset if error is 1
                    reset.write(false);


 //combination 101
TA.write(SC_LOGIC_1);
TB.write(SC_LOGIC_0);
TCin.write(SC_LOGIC_1);
wait(10, SC_NS);
 if(TSum.read()==SC_LOGIC_0&&TCout.read()==SC_LOGIC_1)
 error.write(SC_LOGIC_0);
              enable.write(false);          // counter does not run if error is 0
              reset.write(true);

         else
          error.write(SC_LOGIC_1);
               enable.write(true);   // counter is reset if error is 1
                 reset.write(false);

         //combination 110
        TA.write(SC_LOGIC_1);
        TB.write(SC_LOGIC_1);
        TCin.write(SC_LOGIC_0);
        wait(10, SC_NS);
         if(TSum.read()==SC_LOGIC_0&&TCout.read()==SC_LOGIC_1)
         error.write(SC_LOGIC_0);
              enable.write(false);          // counter does not run if error is 0
              reset.write(true);

         else
          error.write(SC_LOGIC_1);
               enable.write(true);   // counter is reset if error is 1
                 reset.write(false);


        //combination 111
        TA.write(SC_LOGIC_1);
        TB.write(SC_LOGIC_1);
        TCin.write(SC_LOGIC_1);
        wait(10, SC_NS);
         if(TSum.read()==SC_LOGIC_1&&TCout.read()==SC_LOGIC_1)
         error.write(SC_LOGIC_0);
              enable.write(false);          // counter does not run if error is 0
              reset.write(true);

         else
          error.write(SC_LOGIC_1);
               enable.write(true);   // counter is reset if error is 1
                 reset.write(false);

    }
    SC_CTOR(Testbench){
        SC_THREAD(testprocess);
    }
};
```

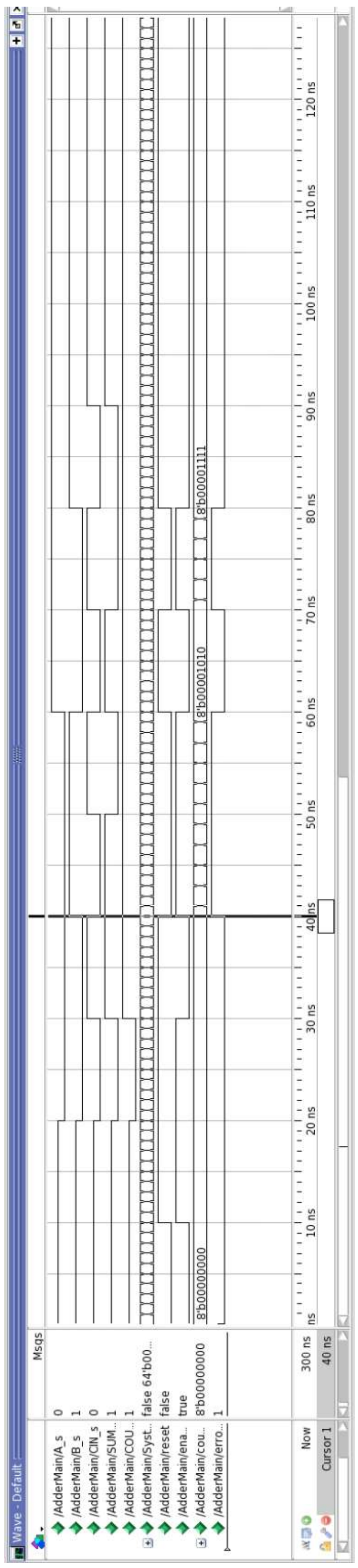

Figure 2b: Waveform of counter integrated with adder.