Ryerson University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	Systems-on-Chip Design
Course Number:	COE 838
Semester/Year (e.g.F2016)	W2022

Instructor:	Dr. Gul Khan

Assignment/Lab Number:	1
Assignment/Lab Title:	Introduction to SystemC

Submission Date:	Thursday, January 27th, 2022
Due Date:	Thursday, January 27th, 2022

Student LAST Name	**Student FIRST Name**	**Student Number**	**Section**	**Signature***
Pano	Toni	500822828	01	T. P.

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

Contents

1. Hand Calculations for Test Cases of ALU with Barrel Shifter

Note that bin(#### ####) shows the digits of a binary number, hex(##) shows the digits of a hex number, and dec(sign ###) shows the digits of a decimal number.

a) Hand Calculations for Barrel Shifter

Table 1. All possible inputs and outputs for the barrel shifter test cases.

Input B	Barrel Shift Operation	Result on Waveform
bin(1000 0001) = hex(81)	no shift	bin(1000 0001) = hex(81) = dec(-127)
bin(1000 0001) = hex(81)	shift left by 1 bit	bin(0000 0011) = hex(03) = dec(3)
bin(1000 0001) = hex(81)	Shift right by 1 bit	bin(1100 0000) = hex(C0) = dec(-64)

bin(1000 0001) = -[NOT(bin(1000 0001)) + bin(1)] = -[bin(0111 1110)) + bin(1)] = -bin(0111 1111)
bin(1000 0001) = -dec(127) = dec(-127)

bin(0000 0011) = +bin(0000 0011) = +dec(3) = dec(3)

bin(1100 0000) = -[NOT(bin(1100 0000)) + bin(1)] = -[bin(0011 1111) + bin(1)] = -bin(0100 0000)
bin(1100 0000) = -dec(64) = dec(-64)

b) Hand Calculations for ALU

Table 2. All possible hex codes for the ALU input and output test cases.

Input A	Input B	ALU Operation	Result
hex(01) = dec(1)	hex(81) = dec(-127)	Subtract (A - B)	dec(1) - dec(-127) = dec(128) = dec(-0) = hex(80)
hex(01) = dec(1)	hex(03) = dec(3)	Subtract (A - B)	dec(1) - dec(3) = dec(-2) = hex(FE)
hex(01) = dec(1)	hex(C0) = dec(-64)	Subtract (A - B)	dec(1) - dec(-64) = dec(65) = hex(41)
hex(01) = dec(1)	hex(81) = dec(-127)	Add (A + B)	dec(1) + dec(-127) = dec(-126) = hex(82)
hex(01) = dec(1)	hex(03) = dec(3)	Add (A + B)	dec(1) + dec(3) = dec(4) = hex(04)
hex(01) = dec(1)	hex(C0) = dec(-64)	Add (A + B)	dec(1) + dec(-64) = dec(-63) = hex(C1)

dec(1) = +dec(1) = +bin(0000 0001) = bin(0000 0001) = hex(01)

dec(128) = +dec(128) = +bin(1000 0000) = -[NOT(bin(1000 0000)) + bin(1)] = -[bin(0111 1111) + bin(1)]
dec(128) = -bin(1000 0000) = -dec(0). dec(128) maps onto the negative binary value for dec(0).
dec(128) = bin(1000 0000) = hex(80)

dec(-2) = -dec(2) = -bin(0000 0010) = [NOT(bin(0000 0010)) + bin(1)] = [bin(1111 1101) + bin(1)]
dec(-2) = bin(1111 1110) = hex(FE)

dec(65) = +dec(65) = +bin(0100 0001) = bin(0100 0001) = hex(41)

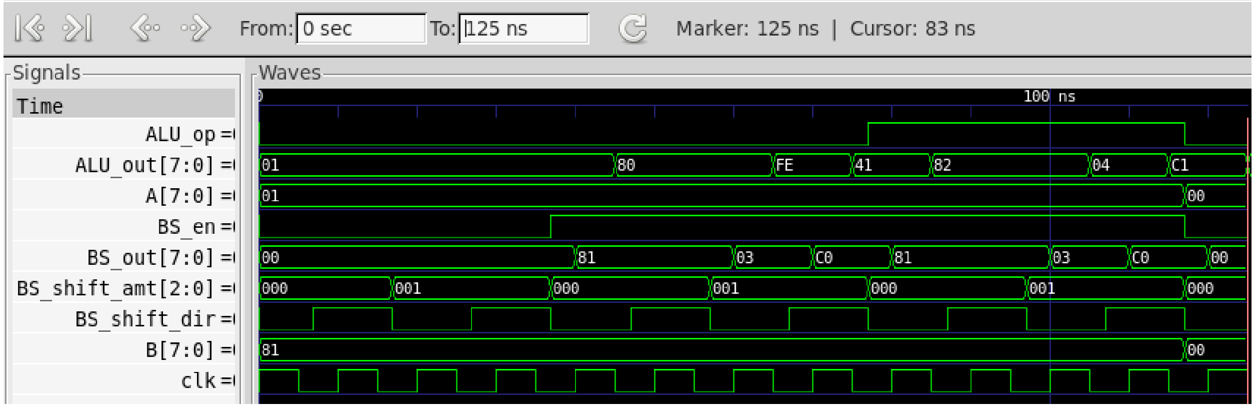
dec(-126) = -dec(126) = -bin(0111 1110) = [NOT(bin(0111 1110)) + bin(1)] = [bin(1000 0001) + bin(1)]
dec(-126) = bin(1000 0010) = hex(82)

dec(4) = +dec(4) = +bin(0000 0100) = bin(0000 0100) = hex(04)

dec(-63) = -dec(63) = -bin(0011 1111) = [NOT(bin(0011 1111)) + bin(1)] = [bin(1100 0000) + bin(1)]
dec(-64) = bin(1100 0001) = hex(C1)

2. Waveforms of Signals in Test Cases

In all test cases, the barrel shifter is triggered on the rising edge of the clock signal, and the ALU is triggered on the falling edge of the clock signal.



From: 0 sec To: 125 ns Marker: 125 ns | Cursor: 83 ns

Figure 1. Waveforms of Signals in all Test Cases for ALU with Barrel Shifter.

Appendix A: Arithmetic Logic Unit (ALU) Code

alu.h

```
#ifndef ALU_H
#define ALU_H

#include <systemc.h>

SC_MODULE(alu){
        //inputs and outputs
        sc_in<bool> clk;                //clock
        sc_in<sc_int<8> > A;    //input A (8 bits)
        sc_in<sc_int<8> > B;    //input B (8 bits)
        sc_in<bool> op;                         //opcode (1 bit)
        sc_out<sc_int<8> > R;   //result (8 bits)

        //function for ALU behaviour, defined in .cpp file
        void behaviour();

        //constructor
        SC_CTOR(alu){
                SC_METHOD(behaviour);  //use SC_METHOD to synthesize ALU behaviour
                sensitive << clk.neg();       //behaviour called when clock changes to low
(negative clock edge)
        }
};
#endif
```

alu.cpp

```
#include <iostream>
#include "alu.h"

sc_int<8> result;

void alu :: behaviour(){
        cout << "\tALU op: " << op.read() << ", A: " << A.read() << ", B: " << B.read() << ", ";

        if(op.read() == 0){    //subtraction (A-B)
                result = A.read() - B.read();
                cout << "A - B = ";
        }
        else{ //addition (A+B)
                result = A.read() + B.read();
                cout << "A + B = ";
        }
        R.write(result);

        cout << result << endl;
}
```

Appendix B: Barrel Shifter Code

barrel_shifter.h

```
#ifndef BARREL_SHIFT_H
#define BARREL_SHIFT_H

#include <systemc.h>

SC_MODULE(barrel_shifter){
        //inputs and outputs
        sc_in<bool> clk;                             //clock
        sc_in<bool> en;                                     //enable
        sc_in<bool> l_r;                             //shift left or right
        sc_in<sc_uint<3> > shift_amt; //shift amount
        sc_in<sc_int<8> > d_in;                      //data in
        sc_out<sc_int<8> > d_out;            //data out

        //function for barrel shifter behaviour, defined in .cpp file
        void behaviour();

        //constructor
        SC_CTOR(barrel_shifter){
                SC_METHOD(behaviour);  //use SC_METHOD to synthesize barrel shifter behaviour
                sensitive << clk.pos();      //behaviour called when clock changes to low
(positive clock edge)
        }
};
#endif
```

barrel_shifter.cpp

```
#include <iostream>
#include "barrel_shifter.h"

sc_int<8> d_shifted;
int bit, shifted_bit;

void barrel_shifter :: behaviour(){
        if(en.read() == 1){    //see if enabled
                if(l_r.read() == 0){ //shift left (assuming MSB on left)
                        shifted_bit = shift_amt.read();
                }
                else{ //shift right (assuming MSB on left)
                        shifted_bit = 8 - shift_amt.read();
                }

                for(bit = 0; bit < 8; bit++){
                        if(shifted_bit == 8){  //mke shifted bit loop back to beginning of data
                                shifted_bit = 0;
                        }

                        d_shifted[shifted_bit] = d_in.read()[bit];
                        shifted_bit++;
                }

                for(bit = 0; bit < 8; bit++){
                        cout << "\t\td_in[" << bit << "]= " <<  d_in.read()[bit] << ", d_out[" <<
bit << "]= " << d_shifted[bit] << endl;
                }

                d_out.write(d_shifted); //write shifted result to output variable
        }
        else{   //see if disabled
                d_out.write(0);
        }

        cout << "\tBS en: " << en.read() << ", l_r: " << l_r.read() << ", shift_amt: " <<
shift_amt.read() << ", d_in: " << d_in.read() << ", d_out: " << d_shifted << endl;
}
```

Appendix C: Top Level Test File Code

sc_main.cpp

```
#include <systemc.h>

//modules
#include "barrel_shifter.h"
#include "alu.h"


int sc_main(int argc, char* argv[]){
        //signals
        sc_clock clk("clk", 10, SC_NS, 0.5);  //main clock signal (10 ns period, 50% duty cycle, 0 
offset, initial value is 0)
        sc_signal<bool> en, l_r, op;
        sc_signal<sc_uint<3> > shift_amt;
        sc_signal<sc_int<8> > A, B, B_shifted, output;

        //barrel shifter module initialization and signal routing
        barrel_shifter barrel("barrel-shifter");
        barrel.clk(clk);
        barrel.en(en);
        barrel.l_r(l_r);
        barrel.shift_amt(shift_amt);
        barrel.d_in(B);
        barrel.d_out(B_shifted);

        //arithmetic logic unit module initialization and signal routing
        alu alu("arithmetic-logic-unit");
        alu.clk(clk);
        alu.A(A);
        alu.B(B_shifted);
        alu.op(op);
        alu.R(output);

        //setup waveform trace file
        sc_trace_file *tf;
        tf = sc_create_vcd_trace_file("trace_file"); //create trace file named "trace_file.vcd"
        tf -> set_time_unit(1, SC_NS);        //set unit time of trace file to 1 ns
        sc_trace(tf, clk, "clk");
        sc_trace(tf, A, "A");
        sc_trace(tf, B, "B");
        sc_trace(tf, en, "BS_en");
        sc_trace(tf, l_r, "BS_shift_dir");
        sc_trace(tf, shift_amt, "BS_shift_amt");
        sc_trace(tf, B_shifted, "BS_out");
        sc_trace(tf, op, "ALU_op");
        sc_trace(tf, output, "ALU_out");

        //simulation test sequence
        cout << "test enable off with all barrel shifter settings" << endl;
        A.write(1);                   //test enable off with all barrel shifter settings
        B.write(-127);
        en.write(0);
        op.write(0);
        l_r.write(0);
        shift_amt.write(0);
        sc_start(7, SC_NS);

        l_r.write(1);
        sc_start(10, SC_NS);

        l_r.write(0);
        shift_amt.write(1);
        sc_start(10, SC_NS);

        l_r.write(1);
        sc_start(10, SC_NS);

        cout << "test enable on, barrel shift both dirs, shift by 1 bit, alu subtract" << endl;
        en.write(1);          //test enable on, barrel shift both dirs, shift by 1 bit, alu
subtract
        l_r.write(0);
        shift_amt.write(0);
        sc_start(10, SC_NS);

        l_r.write(1);
        sc_start(10, SC_NS);

        l_r.write(0);
        shift_amt.write(1);
        sc_start(10, SC_NS);

        l_r.write(1);
        sc_start(10, SC_NS);

        cout << "test enable on, barrel shift both dirs, shift by 1 bit, alu add" << endl;
        en.write(1);          //test enable on, barrel shift both dirs, shift by 1 bit, alu add
        l_r.write(0);
        shift_amt.write(0);
        op.write(1);
        sc_start(10, SC_NS);

        l_r.write(1);
        sc_start(10, SC_NS);

        l_r.write(0);
        shift_amt.write(1);
        sc_start(10, SC_NS);

        l_r.write(1);
        sc_start(10, SC_NS);

        A.write(0);                   //test enable off with all barrel shifter settings
        B.write(0);
        en.write(0);
        op.write(0);
        l_r.write(0);
        shift_amt.write(0);
        sc_start(13, SC_NS);

        //stop simulation
        sc_close_vcd_trace_file(tf);

        return 0;
}
```

Appendix D: Makefile Code

Makefile

```
#
CC=/usr/bin/g++
ARCH := $(shell arch)
SYSTEMC_HOME=/usr/local/SystemC-2.3.0

# 64bit or 32bit libaries to link to
LINUXLIB := $(shell if [ ${ARCH} = "i686" ]; \
                    then \
                         echo lib-linux; \
                    else \
                         echo lib-linux64; \
                    fi)

INCLUDES = -I$(SYSTEMC_HOME)/include -I.

LIBRARIES = -L. -L$(SYSTEMC_HOME)/$(LINUXLIB) -lsystemc -lm

RPATH = -Wl,-rpath=$(SYSTEMC_HOME)/$(LINUXLIB)

PROGRAM = barrel_alu.x
SRCS    =  barrel_shifter.h barrel_shifter.cpp alu.h alu.cpp sc_main.cpp
OBJS    =  barrel_shifter.o alu.o sc_main.o

all : $(PROGRAM)

$(OBJS) : $(SRCS)
        $(CC) $(INCLUDES) -c $(SRCS)

$(PROGRAM) : $(OBJS)
        $(CC) $(INCLUDES) $(LIBRARIES) $(RPATH) -o $(PROGRAM) $(OBJS)

clean:
        @rm -f $(OBJS) $(PROGRAM) *.cpp~ *.h~
```