به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

شبیه سازی شی گرای سیستم های الکترونیکی

CA5

علیرضا جابری راد

۸۱۰۱۹۶۴۳۸

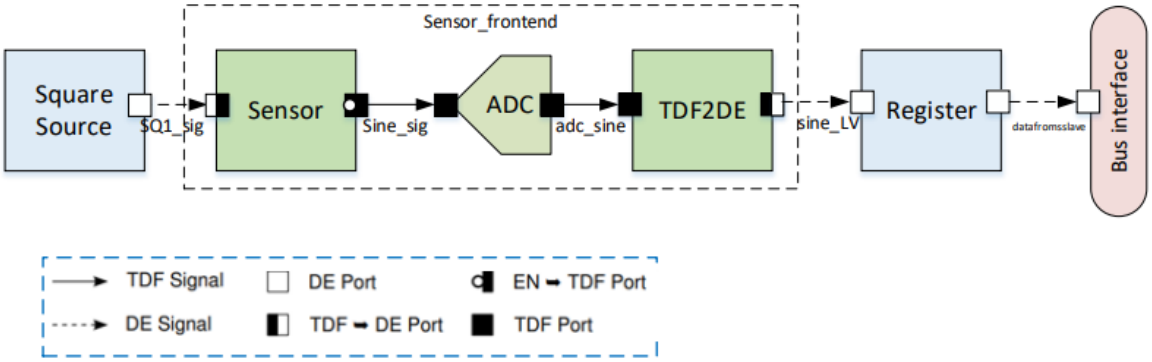
تابستان ۱۴۰۱

بخش اول

ابتدا به پیاده سازی کد بخش سنسور میپردازیم و در ادامه به سراغ پیاده سازی تایمر میرویم.

سنسور

سنسور را بر اساس نوع ورودی ها و خروجی و نحوه ی اتصال آنها که در تصویر زیر آمده، طراحی و پیاده سازی میکنیم.



به منظور پیاده سازی ماژول های مختلف، کد ماژول ها با ترتیب از چپ به راست در تصاویر زیر مشاهده میکنید.

```
#include<systemc.h>
#include<systemc-ams.h>

SC_MODULE(squareWave){
    sc_out <double> out;

    SC_CTOR(squareWave){
        SC_THREAD(wave);
    }
    void wave(){
        while(true){
            out->write(-5.0);
            wait(5, SC_US);
            out->write(5.0);
            wait(5, SC_US);
        }
    }
};
```

```
SC_MODULE(lp_filter){
private:
    sca_eln::sca_node a;
    sca_eln::sca_node b;
    sca_eln::sca_node_ref gnd;
public:
    sc_in<double> in;
    sca_tdf::sca_out<double> out;

    sca_eln::sca_r r1;
    sca_eln::sca_c c1;
    sca_eln::sca_de_vsource vin;
    sca_eln::sca_tdf::sca_vsink vout;

    SC_HAS_PROCESS(sc_module_name);
    lp_filter(sc_module_name, double r1_value, double c1_value):
        r1("r1", r1_value), c1("c1", c1_value), vin("vin",5.0), vout("vout",5.0){
            vin.p(a);
            vin.n(gnd);
            vin.inp(in);
            vin.set_timestep(1, SC_US);

            r1.n(a);
            r1.p(b);

            c1.n(b);
            c1.p(gnd);

            vout.p(b);
            vout.n(gnd);
            vout.outp(out);
        }
};
```

```
#define NBits 16
#define Vs 1000

SCA_TDF_MODULE(ADC)
{
public:
    sca_tdf::sca_in<double> in;
    sca_tdf::sca_out<sc_dt::sc_int<NBits> > out;
    ADC(sc_core::sc_module_name nm, double v_max_)
    : in("in"), out("out"), v_max(v_max_){};

    void processing()
    {
        using namespace std;

        double v_in = in.read();

        if (v_in < -v_max) {
            out.write(-Vs);
        } else if (v_in > v_max) {
            out.write(Vs);
        } else {
            sc_dt::sc_int<NBits> q_v_in;
            q_v_in = lround((v_in / v_max) * Vs);
            out.write(q_v_in);
        }
    }

private:
    const double v_max;
};
```

```
#define NBits 16

SCA_TDF_MODULE(tdf2de){
    sca_tdf::sca_in<sc_dt::sc_int<NBits> > in;
    sca_tdf::sca_out<sc_lv<NBits> > out;

    SC_CTOR(tdf2de){}
    void processing(){
        out.write(in.read());
    }
};
```

```
#define NBits 16

SC_MODULE(reg){
    sc_in<sc_logic> outEnable;
    sc_in<sc_lv<NBits> > in;
    sc_out_rv<NBits> out;

    sc_lv<NBits> reg_val;
    sc_logic clk;

    SC_CTOR(reg){
        SC_THREAD(clocking);
        SC_THREAD(ld_reg);
        SC_THREAD(tristate);
        sensitive<<outEnable;
    }
    void clocking(){
        while(true){
            clk=SC_LOGIC_0;
            wait(500, SC_NS);
            clk=SC_LOGIC_1;
            wait(500, SC_NS);
        }
    }
    void ld_reg(){
        while(true){
            if(clk==SC_LOGIC_1)
                reg_val = in.read();
        }
    }
    void tristate(){
        while(true){
            if(outEnable==SC_LOGIC_1)
                out = reg_val;
            else
                out.write("ZZZZZZZZZZZZZZZZ");
        }
    }
};
```

همچنین ماژولهایی که در شکل اول درون خط چین بودند نیز در یک ماژول کلی تر قرار گرفتند که تصویر کد آنرا در شکل زیر مشاهده میفرمایید:

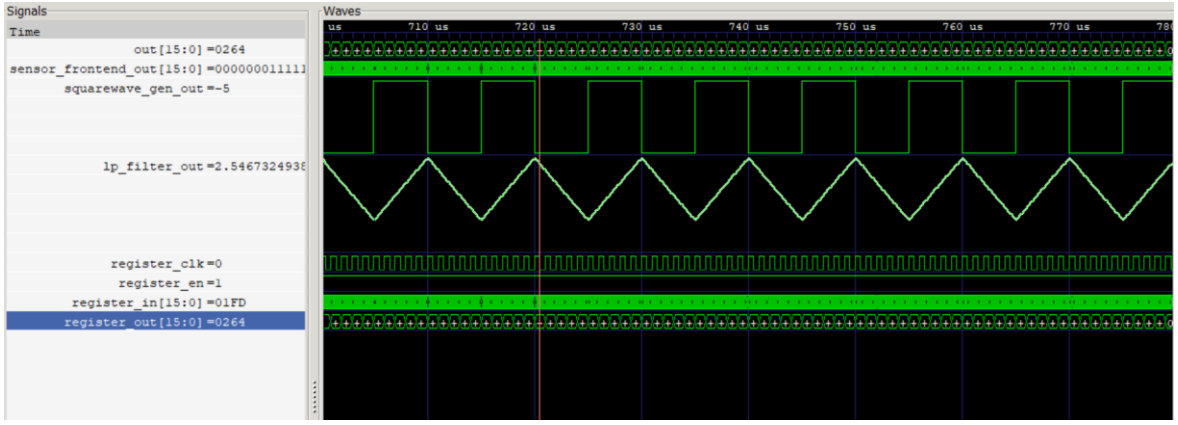
```cpp
#include "lp_filter.h"
#include "ADC.h"
#include "tdf2de.h"

#define NBits 16

SC_MODULE(sensor_frontend){
    sc_in<double> in;
    sc_out<sc_lv<NBits> > out;

    sca_tdf::sca_signal <double> sine_sig;
    sca_tdf::sca_signal <sc_dt::sc_int<NBits> > adc_sine;
    sca_tdf::sca_signal <sc_lv<NBits> > sine_lv;

    lp_filter* filter;
    ADC* adc;
    tdf2de* t2d;
    SC_CTOR(sensor_frontend){
        filter = new lp_filter("LowPassFilter",1000,10e-6);
        filter->in(in);
        filter->out(sine_sig);

        adc = new ADC("ADC", 5.0);
        adc->in(sine_sig);
        adc->out(adc_sine);

        t2d = new tdf2de("TDF2DE");
        t2d->in(adc_sine);
        t2d->out(sine_lv);
    }
};
```

تست سنسور

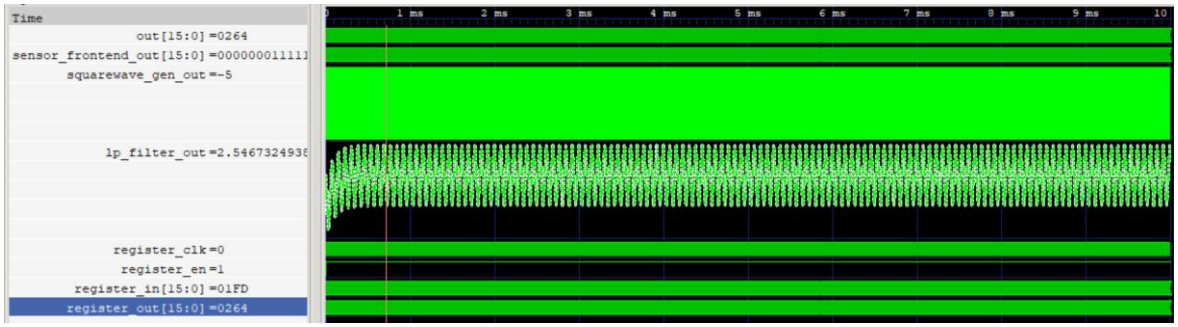
به منظور تست سنسور، آنرا در بدنه sc_main استفاده کرده و مدار را تست کردم:

```cpp
#include"sensor.h"

int sc_main(int argc, char** argv){
    sc_set_time_resolution(10, SC_NS);

    sc_signal_rv<NBits> out;
    sc_signal<sc_logic> en;
    en.write(SC_LOGIC_1);

    sensor*CUT = new sensor("sens");
    CUT->outEnable(en);
    CUT->datafromslave(out);

    sca_util::sca_trace_file* FILE;
    FILE = sca_util::sca_create_vcd_trace_file("sensor_test");
    sca_util::sca_trace(FILE, CUT->datafromslave, "out");
    sca_util::sca_trace(FILE, CUT->SF->out, "sensor_frontend_out");
    sca_util::sca_trace(FILE, CUT->SF->filter->out, "lp_filter_out");
    sca_util::sca_trace(FILE, CUT->SW->out, "squarewave_gen_out");
    sca_util::sca_trace(FILE, CUT->R->in, "register_in");
    sca_util::sca_trace(FILE, CUT->R->out, "register_out");
    sca_util::sca_trace(FILE, CUT->R->clk, "register_clk");
    sca_util::sca_trace(FILE, CUT->R->outEnable, "register_en");

    sc_start(10, SC_MS);
    sca_util::sca_close_vcd_trace_file(FILE);
    return 0;
}
```

با اجرای کد کامپایل شده در این بخش، خروجی شکل موج ها و خروجی کلی سنسور و برخی سیگنال های دیگر به ترتیب زیر خواهد بود:



نمایی دورتر از تست انجام شده:



تایمر

کد تایمر مطابق تصویر زیر پیاده سازی شده است.

```
SC_MODULE(timer){
    sc_in<sc_logic> clk, outEnable, startTimer;
    sc_out_rv<16> timeOut;
    sc_signal<sc_logic> flag;
    SC_CTOR(timer){
        SC_THREAD(evl);
        SC_THREAD(tristate);
        sensitive << outEnable;
    }

    void evl(){
        while(true){
            wait(startTimer.posedge_event());
            flag=SC_LOGIC_0;
            wait(1, SC_MS);
            flag=SC_LOGIC_1;
        }
    }

    void tristate(){
        while(true){
            if(outEnable == SC_LOGIC_1){
                if(flag==SC_LOGIC_0)
                    timeOut.write("0000000000000000");
                else if(flag==SC_LOGIC_1)
                    timeOut.write("0000000000000001");
            }
            else
                timeOut.write("ZZZZZZZZZZZZZZZZ");
        }
    }
};
```

بخش دوم

باس

کد پیاده سازی بخش باس مطابق تصویر زیر است:

```
SC_MODULE(bus_if){
    sc_in<sc_logic> readIO, writeIO;
    sc_in<sc_lv<16> > addrBus;

    sc_out<sc_logic> timerEnable, sensorEnable, startTimer;
    sc_lv<16> mask, mask_;
    SC_CTOR(bus_if){
        SC_THREAD(timer);
        sensitive<< readIO<< writeIO<< addrBus;
        SC_THREAD(sensor);
        sensitive<< readIO<< writeIO<< addrBus;
    }

    void timer(){
        while(true){
            timerEnable = SC_LOGIC_0;
            startTimer = SC_LOGIC_0;
            mask= addrBus.read().to_uint() & 0x00FF;
            if(mask == 0x0010 || mask == 0x0011 || mask == 0x0012 || mask == 0x0013 ){
                if(readIO == SC_LOGIC_1)
                    timerEnable = SC_LOGIC_1;
                else if(writeIO == SC_LOGIC_1)
                    startTimer = SC_LOGIC_1;
            }
            wait();
        }
    }

    void sensor(){
        while(true){
            mask_= addrBus.read().to_uint() & 0x00FF;
            if(mask_ == 0x0018 || mask_ == 0x0019 || mask_ == 0x001A || mask_ == 0x001B ){
                if(readIO == SC_LOGIC_1){
                    sensorEnable = SC_LOGIC_1;
                }
            }
            wait();
        }
    }
};
```

پردازنده

کد پیاده سازی بخش پردازنده را در تصویر زیر مشاهده میفرمایید:

```
using namespace std;

SC_MODULE(processor){
    sc_in <sc_logic> clk;

    sc_out<sc_logic> readIO, writeIO;
    sc_out<sc_lv<16> > addrBus;

    sc_inout_rv <16> dataBus;

    SC_CTOR(processor){
        SC_THREAD(bracketing);
        sensitive<<clk.pos();
    }

    void bracketing(){
        int arr[100];
        while(true){
            for(int i=0; i<100; i++){
                writeIO = SC_LOGIC_1;
                readIO = SC_LOGIC_0;
                addrBus = 0x0010;
                wait();

                writeIO = SC_LOGIC_1;
                readIO = SC_LOGIC_0;
                wait();

                while(dataBus.read().to_uint() != 1)
                    wait();

                writeIO = SC_LOGIC_0;
                readIO = SC_LOGIC_1;
                addrBus = 0x0018;
                wait();
                arr[i]=dataBus.read().to_uint();
            }
            double avg = 0;
            for(int i=0; i<100; i++)
                avg+=arr[i];
            avg=avg/100;

            cout<<"average is: "<<avg<<endl;
        }
    }
};
```

پیاده سازی کل مدار و تست آن

در تصویر زیر، در بدنه‌ی sc_main همه ی ماژول ها به هم وصل شده اند و مدار کلی به مدت ۳ ثانیه شبیه سازی میشود:

```
#include "processor.h"
#include "sensor.h"
#include "timer.h"
#include "clk_gen.h"
#include "bus_if.h"

int sc_main(int argc, char** argv){
    sc_set_time_resolution(10, SC_NS);
    sca_util::sca_trace_file* File = sca_util::sca_create_vcd_trace_file("completeCircuitTB");

    sc_signal<sc_logic> sensorEnable;
    sc_signal_rv<16> dataBus;
    sc_signal<sc_logic> clk, startTimer, timerEnable;
    sc_signal<sc_logic> readIO, writeIO;
    sc_signal<sc_lv<16> > addrBus;

    sensor S("sensor");
    timer T("timer");
    processor P("processor");
    clk_gen C("clock");
    bus_if B("Bus");

    C.out(clk);

    S.outEnable(sensorEnable);
    S.datafromslave(dataBus);

    T.clk(clk);
    T.startTimer(startTimer);
    T.outEnable(timerEnable);
    T.timeOut(dataBus);

    B.readIO(readIO);
    B.writeIO(writeIO);
    B.addrBus(addrBus);
    B.timerEnable(timerEnable);
    B.sensorEnable(sensorEnable);
    B.startTimer(startTimer);

    P.clk(clk);
    P.dataBus(dataBus);
    P.readIO(readIO);
    P.writeIO(writeIO);
    P.addrBus(addrBus);

    sca_util::sca_trace(File, S.SF->out,"sensorOut");

    sc_start(3000, SC_MS);
    sca_util::sca_close_vcd_trace_file(File);
}
```

متاسفانه این بخش هم مانند بخش اول، کد کامپایل میشود اما پس از اجرای فایل اجرایی آن، متوقف نمیشود. تصویر آنرا در ادامه مشاهده میکنید:

```
alireza@ubuntu:~/OO/CA5$ g++ -std=c++11 -I. -I$SYSTEMC_PATH/include -I$SYSTEMC_AMS_PATH/include -L. -L$SYSTEMC_PATH/lib-linux64 -L$SYSTEMC_AMS_PATH/lib-linux64 -Wl,-rpath=$SYSTEMC_PATH/lib
-linux64 -o ca5 squareWave.h ADC.h lp_filter.h tdf2de.h sensor_frontend.h reg.h sensor.h timer.h processor.h clk_gen.h bus_if.h main.cpp -lsystemc -lsystemc-ams -lm
main.cpp: In function ‘int sc_main(int, char**)’:
main.cpp:50:1: warning: no return statement in function returning non-void [-Wreturn-type]
   50 | }
      | ^
alireza@ubuntu:~/OO/CA5$ ./ca5

        SystemC 2.3.1-Accellera --- Jun 10 2022 05:04:53
        Copyright (c) 1996-2014 by all Contributors,
        ALL RIGHTS RESERVED

Warning: (W516) default time unit changed to time resolution
In file: ../../../../src/sysc/kernel/sc_time.cpp:329


        SystemC AMS extensions 2.3.0-COSEDA Release date: 20200312 2138
        Copyright (c) 2010-2014  by Fraunhofer-Gesellschaft IIS/EAS
        Copyright (c) 2015-2020  by COSEDA Technologies GmbH
        Licensed under the Apache License, Version 2.0


Info: SystemC-AMS:
        6 SystemC-AMS modules instantiated
        2 SystemC-AMS views created
        3 SystemC-AMS synchronization objects/solvers instantiated


Info: SystemC-AMS:
        1 dataflow clusters instantiated
          cluster 0:
                3 dataflow modules/solver, contains e.g. module: sca_linear_solver_0 containing modules:
                sensor.InternalSensor.LowPassFilter.r1
                sensor.InternalSensor.LowPassFilter.c1
                sensor.InternalSensor.LowPassFilter.vout
                sensor.InternalSensor.LowPassFilter.vin

                3 elements in schedule list,
                1 us cluster period,
                ratio to lowest:  1             e.g. module: sca_linear_solver_0
                ratio to highest: 1 sample time  e.g. module: sca_linear_solver_0
                0 connections to SystemC de, 1 connections from SystemC de
```