HW6 : SystemC AMS Modeling - ELN

Student Name: Amirhosein Yavarikhoo

Student ID: 810199514

***Abstract*— SystemC AMS offers us a solution to connect digital circuits to analog circuits. In this experiment, we use a systemC module to create a DE sin signal, convert it to analog, filter it using a bandpass filter and at last convert it again to digital.**

***Keywords*— System C modelling, analog circuits, digital to analog converters, bandpass filter**

I. Introduction

In this assignment, we design a bandpass filter and connect it to a simple sin wave generator that is digital and see how the filter works.

II. Bandpass Filter

Bandpass filter with two converters for digital to analog signals is shown below:

```cpp
#include <systemc.h>
#include <systemc-ams.h>
SC_MODULE(MyFilter) {
public:
    sc_in<double> in;
    sc_out <double> out;
    sca_eln::sca_de_vsource Vin;
    sca_eln::sca_de_vsink Vout;
    sca_eln::sca_r *R1;
    sca_eln::sca_r *Rload;
    sca_eln::sca_c *C1;
    sca_eln::sca_c *C2;
    sca_eln::sca_node a;
    sca_eln::sca_node b;
    sca_eln::sca_node c;
    sca_eln::sca_node_ref gnd;
    MyFilter(sc_module_name): Vin("Vin"),Vout("Vout") {
        R1 = new sca_eln::sca_r("r1", 200);
        R1->n(a);
        R1->p(b);
        R1->set_timestep(1, SC_MS);
        Rload = new sca_eln::sca_r("rload", 1000);
        Rload->n(c);
        Rload->p(gnd);
        C1 = new sca_eln::sca_c("c1", 2.5e-6);
        C1->n(b);
        C1->p(gnd);
        C2 = new sca_eln::sca_c("c2", 1e-6);
        C2->n(b);
        C2->p(c);
        Vin.p(a);
        Vin.n(gnd);
        Vin.inp(in);
        Vin.set_timestep(1, SC_MS);
        Vout.p(c);
        Vout.n(gnd);
        Vout.outp(out);
    }

};
```

Fig. 1 bandpass filter code

III. Sin wave generator

To create a sin wave, we use a simple SC_THREAD and a for loop.

```cpp
#include <systemc.h>
#include <systemc-ams.h>
SC_MODULE(Sinus) {
    sc_out<double> out;
    SC_CTOR(Sinus) {
        SC_THREAD(generate);
    }
    void generate() {
        for (int i = 0; i < 2000; i++) {
            out->write(sin(2 * 3.14 * 100 * i));
            wait(0.1, SC_MS);
        }
    }
};
```

Fig. 2 Sin wave generator

IV. Testbench

Code for testbench is shown below:

```cpp
#include "MyFilter.cpp"
#include "Sin.cpp"
SC_MODULE(TB) {
    sc_signal<double> in, out;
    MyFilter* UUT;
    Sinus* input_wave;
    SC_CTOR(TB) {
        input_wave = new Sinus("wave_instance");
        input_wave->out(in);
        UUT = new MyFilter("filter_instance");
        UUT->in(in);
        UUT->out(out);
    }
};
```

Fig. 3 Testbench code

Output of the sin wave generator connects to the input of the bandpass filter.
Main function is shown below:

```cpp
#include "TB.cpp"

int sc_main(int argc, char* argv[]) {
    sc_set_time_resolution(0.01, SC_MS);

    TB* testbench = new TB("testbench_instance");
    sc_trace_file* HW6 = sc_create_vcd_trace_file("HW6");
    sc_trace(HW6, testbench->in, "vin");
    sc_trace(HW6, testbench->out, "vout");

    sc_start(500, SC_MS);
    return 0;
}
```

Fig. 4 main function

Result of the trace file is shown below:

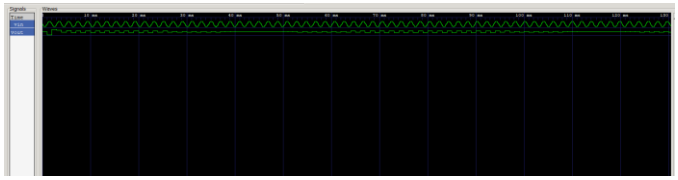

Fig. 5 bandpass result