

# Homework 6 – SystemC AMS Modeling - ELN

Taghizadeh,  
810198373

## I. INTRODUCTION

In this homework, we are about to make a Band-Pass Filter using *SystemC-AMS* ELN elements. This is done by combining a low-pass filter and a high-pass filter.

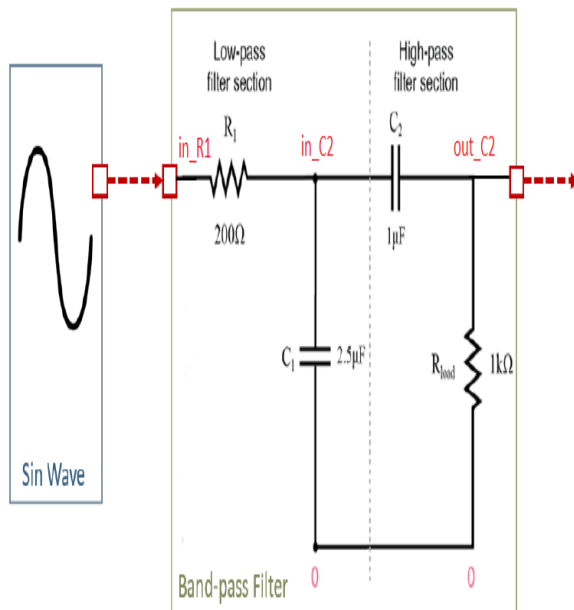


Figure 1 – An overall view of the system

```
1 #include "systemc.h"
2 #include "systemc-ams.h"
3 #include <stdio.h>
4
5 using namespace std;
6
7 SC_MODULE(band_pass_filter)
8 {
9     sc_core::sc_in<double> input;
10    sc_core::sc_out<double> output;
11
12    sca_eln::sca_r R1;
13    sca_eln::sca_r R_Load;
14    sca_eln::sca_c c1;
15    sca_eln::sca_c c2;
16    sca_eln::sca_de::sca_vsource DE_to_ELN_converter;
17    sca_eln::sca_de::sca_vsink ELN_to_DE_converter;
18
19    band_pass_filter(sc_core::sc_module_name, double R1_value, double
20                    R_Load_value, double c1_value, double c2_value)
21        : input("input"), output("output"),
22          R1("R1", R1_value), R_Load("R_Load", R_Load_value), c1("c1",
23            c1_value), c2("c2", c2_value), DE_to_ELN_converter("DE_to_ELN_converter"),
24            ELN_to_DE_converter("ELN_to_DE_converter")
25    {
```

Red arrows point to lines 9, 13, and 17 in the code.

Fig. 2 Band-Pass Filter SystemC-AMS

## II. BAND PASS FILTER IMPLEMENTATION

### A. Band-Pass Filter in SystemC-AMS

To design the *Band-Pass Filter*, first  $R_1$ ,  $R_{load}$ ,  $C_1$  and  $C_2$  are defined. Input and output of this filter are DE double signals, so a DE to ELN source converter and an ELN to DE sink converter are declared to convert the DE input signal to ELN and convert the ELN filter output to DE double signal.

Then simply we connect n and p (negative and positive) nodes of these elements to form the *Band-Pass Filter*. The DE to ELN converter must be connected to the input of the filter and the ELN to DE converter must be connected to the output of the filter. Gnd reference node and internal nodes (input node of R1, input node of C2, output node of C2) are defined as private ELN nodes.

```

24 DE_to_ELN_converter.inp(input);
25 DE_to_ELN_converter.p(in_R1);
26 DE_to_ELN_converter.n(gnd);
27 DE_to_ELN_converter.set_timestep(1, sc_core::SC_US);
28
29 R1.p(in_R1);
30 R1.n(in_C2);
31
32 R_Load.p(out_C2);
33 R_Load.n(gnd);
34
35 c1.p(in_C2);
36 c1.n(gnd);
37
38 c2.p(in_C2);
39 c2.n(out_C2);
40
41 ELN_to_DE_converter.p(out_C2);
42 ELN_to_DE_converter.n(gnd);
43 ELN_to_DE_converter.outp(output);
44 }
45
46 private:
47 sca_eln::sca_node in_R1, in_C2, out_C2;
48 sca_eln::sca_node_ref gnd;
49 };

```

Fig. 3 Band-Pass Filter SystemC-AMS

increase) so we decided to have 1 micro second time step to have enough resolution for sine wave generation.

```

3 #include <stdio.h>
4
5 #define PI 3.1415926535897932384626433832795
6
7 using namespace std;
8
9 SC_MODULE(sine_wave_generator)
10 {
11     sc_out<double> out;
12     SC_CTOR(sine_wave_generator){
13
14         SC_THREAD(sine_wave_generation);
15     }
16     void sine_wave_generation();
17 };
18
19 void sine_wave_generator::sine_wave_generation()
20 {
21     double t = 0;
22     double freq = 1000.0;
23     while (1){
24         wait(1000, SC_NS);
25         out.write(sin( 2.0 * PI * freq * t ) );
26         t = t + 0.000001;
27     }
28 };

```

Fig. 4 Sine Wave Generator SystemC-AMS

### B. Sine Wave Generator in SystemC-AMS

A double variable 't' is defined to represent the time in the simulation. In a loop, time is advanced by *wait()*, then  $\sin(2\pi ft)$  is calculated and written in the output, in the end due to *wait()*, time of the simulation is advanced so 't' must be increased (1 micro second indicates the sine wave resolution and time step. It can be larger (so the resolution of the sine wave will decrease) or smaller (so the resolution of the sine wave will

### C. Band-Pass Filter Test Bench

To test the designed *Band-Pass Filter*, first time resolution (time step) is set to 10 nano second. Then *Vin* (input DE double signal of the *Band-Pass Filter*) and *Vout* (output DE double signal of the *Band-Pass Filter*) are defined, in the end an instance of *Sine\_Wave\_Generator* and *Band\_Pass\_Filter* are taken (200, 1000, 2.5e-6 and 1e-6 are the values of R1, R<sub>load</sub>, C1 and C2) and a VCD file is created to trace the input and output signals of the filter (*Vin* and *Vout*).

```
File Edit Selection Find View Goto Tools Project Preferences Help
multiplier_intermediate_component_fifo_interface_TB.cpp x Test_Bench.cpp x PE_controller.cpp x
1 #include "Band_Pass_Filter.h"
2 #include "Sine_Wave_Generator.h"
3
4
5 int sc_main(int argc, char *argv[]) {
6     sc_core::sc_set_time_resolution(10.0, sc_core::SC_NS);
7     sc_signal<double> Vin, Vout;
8
9     band_pass_filter Band_Pass_Filter("Band_Pass_Filter", 200, 1000, 2.5e-6, 1.0e-6);
10    sine_wave_generator Sine_Wave_Generator("Sine_Wave_Generator");
11
12    Sine_Wave_Generator.out(Vin);
13    Band_Pass_Filter.input(Vin);
14    Band_Pass_Filter.output(Vout);
15
16
17    sca_trace_file* trace_file = sca_create_vcd_trace_file("trace_file");
18
19    sca_trace(trace_file, Vin, "Vin");
20    sca_trace(trace_file, Vout, "Vout");
21
22
23    sc_start(400, SC_MS);
24
25    return 0;
26 }
```

Fig. 5 Band-Pass Filter Test Bench

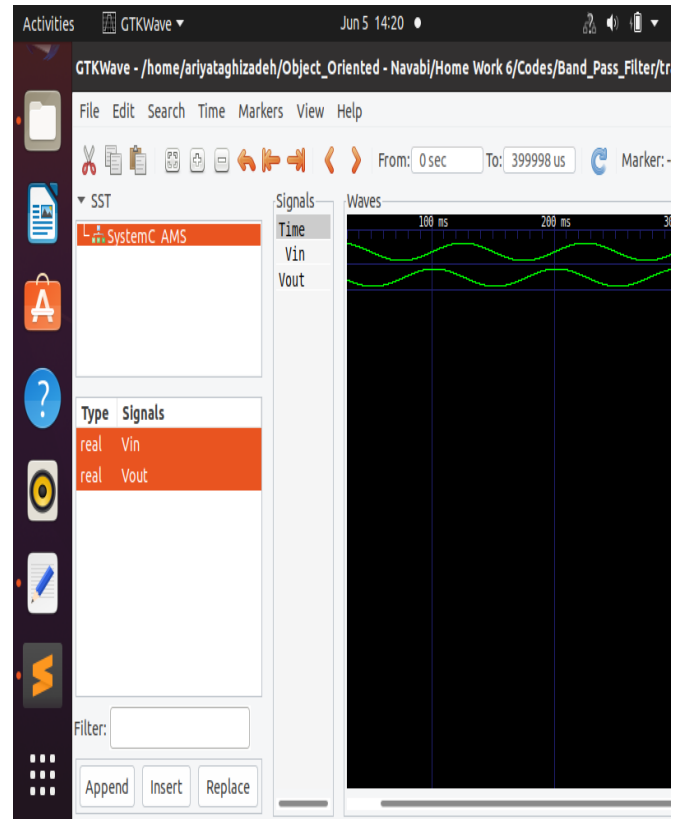


Fig. 6 Band-Pass Filter Test Bench Results (10 Hz sine wave)

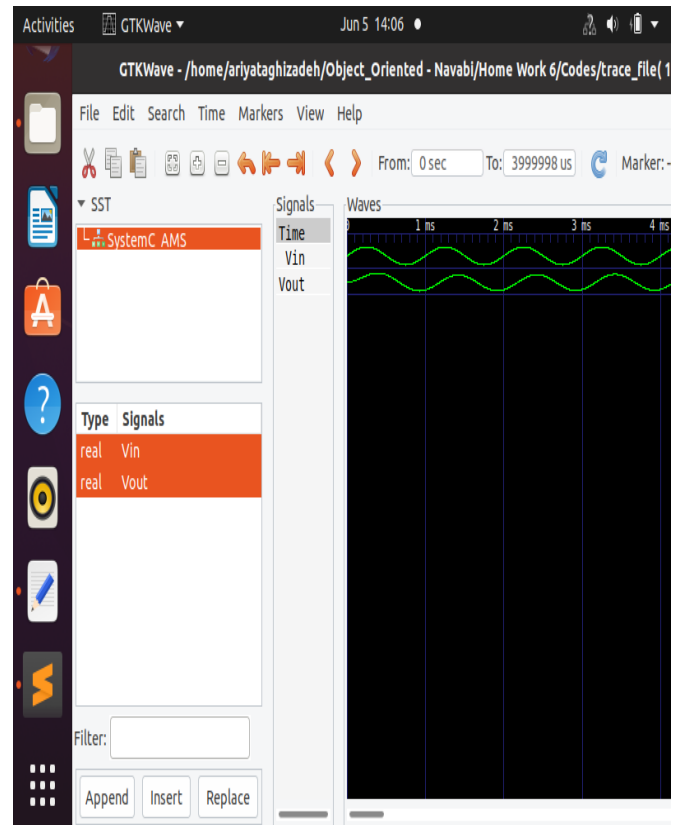


Fig. 7 Band-Pass Filter Test Bench Results (1 KHz sine wave)

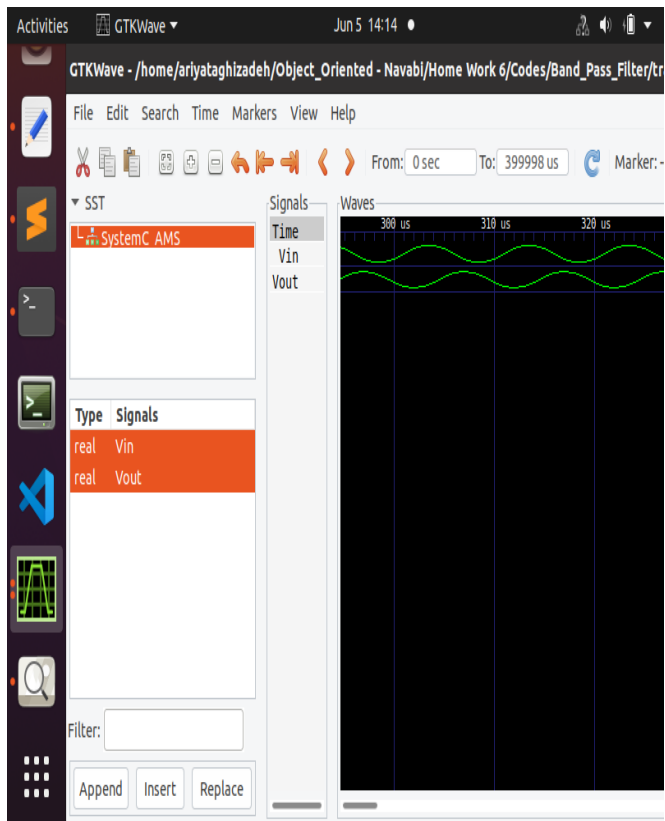


Fig. 8 Band-Pass Filter Test Bench Results (100 KHz sine wave)

According to the above results, the output signal ( $V_{out}$ ) is shifted version of the input signal ( $V_{in}$ ) which has lower amplitude (about 0.1 of the input sine wave's amplitude), for 10 Hz sine wave, this shift is negative, which makes the output comes before the input signal and have lag. For 1 KHz sine wave, this shift is positive, which makes the output comes after the input signal. For 100 KHz sine wave, this shift is more positive. This is what a Band-Pass filter does to a single ton sine wave in time domain.