7/17/2021

CA6

Negin Safari 810197525

Dr. Navvabi

Negin Safari
UNIVERSITY OF TEHRAN

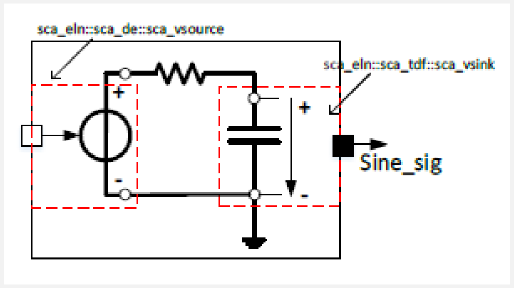
In this computer assignment we implemented an embedded system. We did it with both bracketing and ISS.

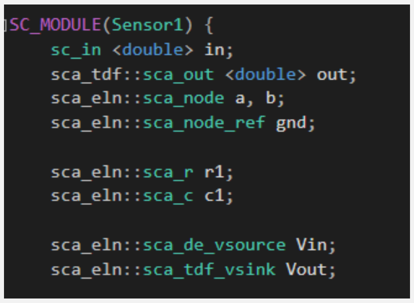
IO devices

This embedded system is made of 4 IO devices that are 2 sensors that one of them generates sine wave and the other one generates triangular wave, one timer and one display device. We are going to explain the implementation of each of them in the following paragraphs.

Sinusoidal IO

This module is made of a square source with frequency 100 KHz which is a simple function. The sensor of this part is an ELN module that is a simple RC circuit. We apply the square wave to its input and we get the voltage of the capacitor as an output. This output is the Voltage that changes by charging and discharging of the capacitor. The time step of this module is 100 ns which is so much higher than the frequency of input voltage.



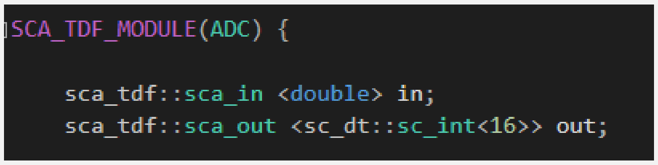

```
SC_MODULE(Sensor1) {
    sc_in <double> in;
    sca_tdf::sca_out <double> out;
    sca_eln::sca_node a, b;
    sca_eln::sca_node_ref gnd;

    sca_eln::sca_r r1;
    sca_eln::sca_c c1;

    sca_eln::sca_de_vsource Vin;
    sca_eln::sca_tdf_vsink Vout;
```

There is also a converter module that takes the TDF output of the circuit and by filtering the input due to the given Vmax it generates integer output.

The function that this ADC module does is like the function bellow:

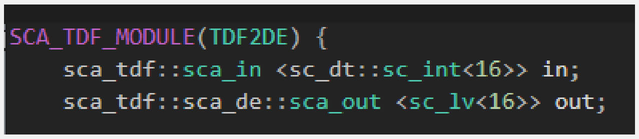
$$\begin{cases} Vmax & V_{in} > V_{max} \\ lround(\frac{V_{in}}{V_{max}} \times (2^{N-1} - 1)) & -V_{max} < V_{in} < V_{max} \\ -Vmax & V_{in} < -V_{max} \end{cases}$$

If the input value is higher than Vmax or less than –Vmax the output would be Vmax or –Vmax but if this input is inside this interval the output would follow the middle statement of the function. In this project Vmax is 0.6 and N is 16.



```
SCA_TDF_MODULE(ADC) {

    sca_tdf::sca_in <double> in;
    sca_tdf::sca_out <sc_dt::sc_int<16>> out;
```

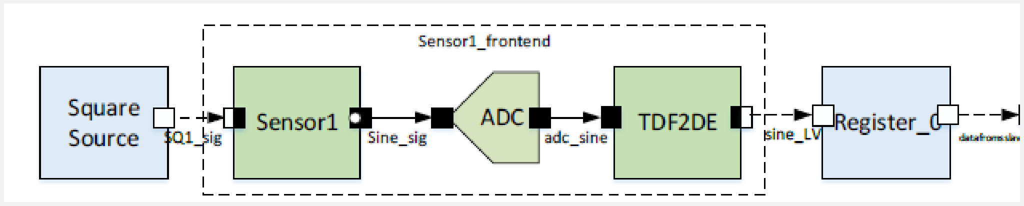
After this module there is a converter that converts that integer value in TDF to a discrete value (sc_lv<16>). The name of this module is TDF2DE.



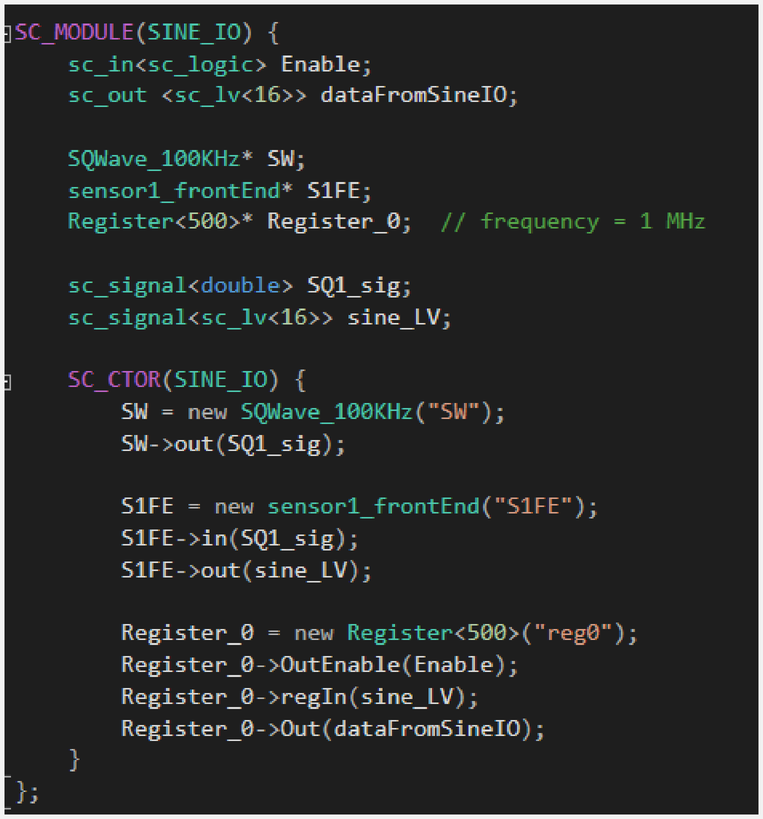
```
SCA_TDF_MODULE(TDF2DE) {
    sca_tdf::sca_in <sc_dt::sc_int<16>> in;
    sca_tdf::sca_de::sca_out <sc_lv<16>> out;
```

The last component in this sine generator is a regular register that has a tri state at its output. The enable of this tristate is controlled by the address that the processor generates. The frequency of this register is 1MHz.

In general the whole sine generator IO device has the following structure.

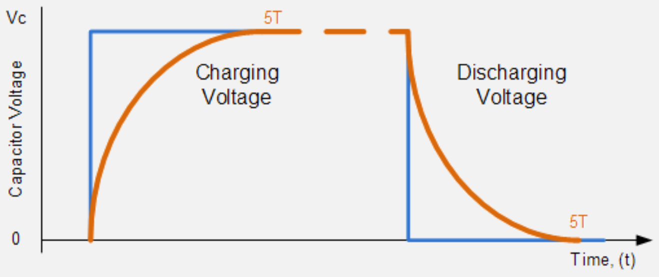


In systemC code the main function is in the following picture:



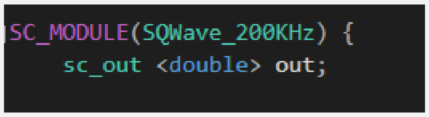
```
SC_MODULE(SINE_IO) {
    sc_in<sc_logic> Enable;
    sc_out <sc_lv<16>> dataFromSineIO;

    SQWave_100KHz* SW;
    sensor1_frontEnd* S1FE;
    Register<500>* Register_0;   // frequency = 1 MHz

    sc_signal<double> SQ1_sig;
    sc_signal<sc_lv<16>> sine_LV;

    SC_CTOR(SINE_IO) {
        SW = new SQWave_100KHz("SW");
        SW->out(SQ1_sig);

        S1FE = new sensor1_frontEnd("S1FE");
        S1FE->in(SQ1_sig);
        S1FE->out(sine_LV);

        Register_0 = new Register<500>("reg0");
        Register_0->OutEnable(Enable);
        Register_0->regIn(sine_LV);
        Register_0->Out(dataFromSineIO);
    }
};
```

In this picture you can see the schematic of square wave and the output of the ELN:



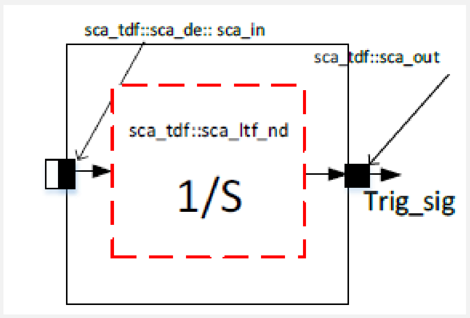
Triangular IO

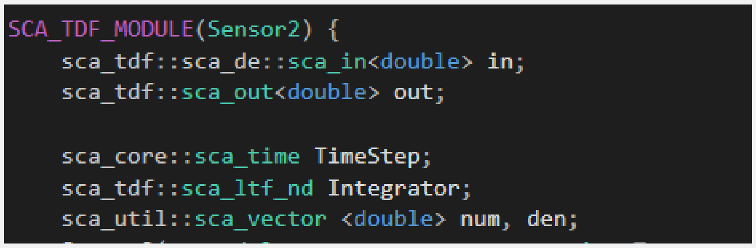
This IO device is our second IO device that is really similar to the previous one in general. The frequency of the input square wave of it is 200 KHz. As we want to generate triangular pulses in the output, we should have a symmetrical domain and for having a high value in order to pass the filters and avoiding zero outputs the domain of this input is 300000. The register works with a clock with 2 MHz frequency.



```
SC_MODULE(SQWave_200KHz) {
    sc_out <double> out;
```

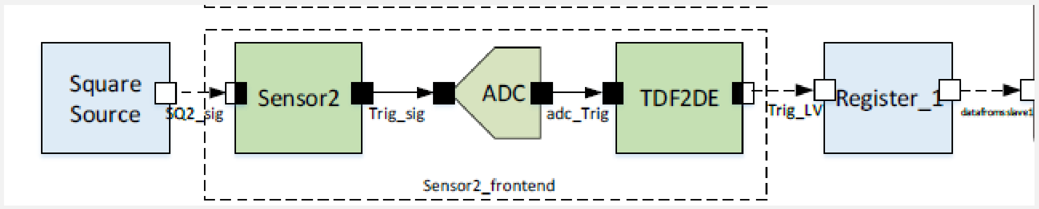
The sensor in this part is an integrator (1/s) so the output of it would be triangular pulses.



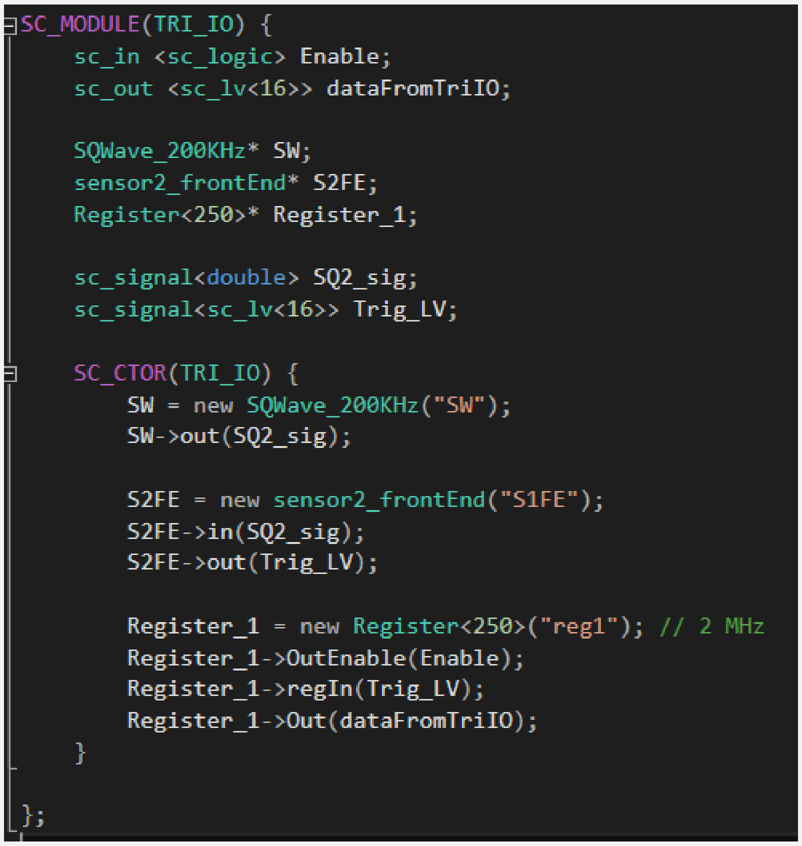


```
SCA_TDF_MODULE(Sensor2) {
    sca_tdf::sca_de::sca_in<double> in;
    sca_tdf::sca_out<double> out;

    sca_core::sca_time TimeStep;
    sca_tdf::sca_ltf_nd Integrator;
    sca_util::sca_vector <double> num, den;
```

In general the whole Tri generator IO device has the following structure.

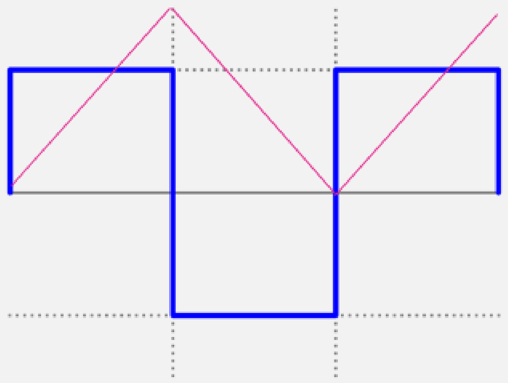


SystemC code of the main function is in the following picture:



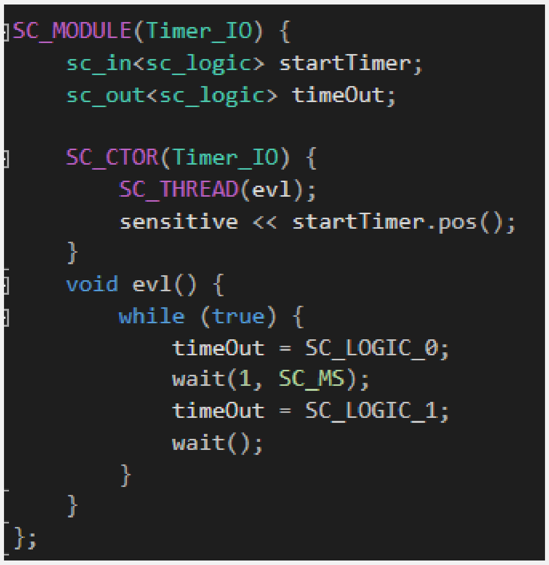
```
SC_MODULE(TRI_IO) {
    sc_in <sc_logic> Enable;
    sc_out <sc_lv<16>> dataFromTriIO;

    SQWave_200KHz* SW;
    sensor2_frontEnd* S2FE;
    Register<250>* Register_1;

    sc_signal<double> SQ2_sig;
    sc_signal<sc_lv<16>> Trig_LV;

    SC_CTOR(TRI_IO) {
        SW = new SQWave_200KHz("SW");
        SW->out(SQ2_sig);

        S2FE = new sensor2_frontEnd("S1FE");
        S2FE->in(SQ2_sig);
        S2FE->out(Trig_LV);

        Register_1 = new Register<250>("reg1"); // 2 MHz
        Register_1->OutEnable(Enable);
        Register_1->regIn(Trig_LV);
        Register_1->Out(dataFromTriIO);
    }

};
```

In this picture you can see the schematic of symmetrical square wave and the output of the TDF integrator function:



Timer IO

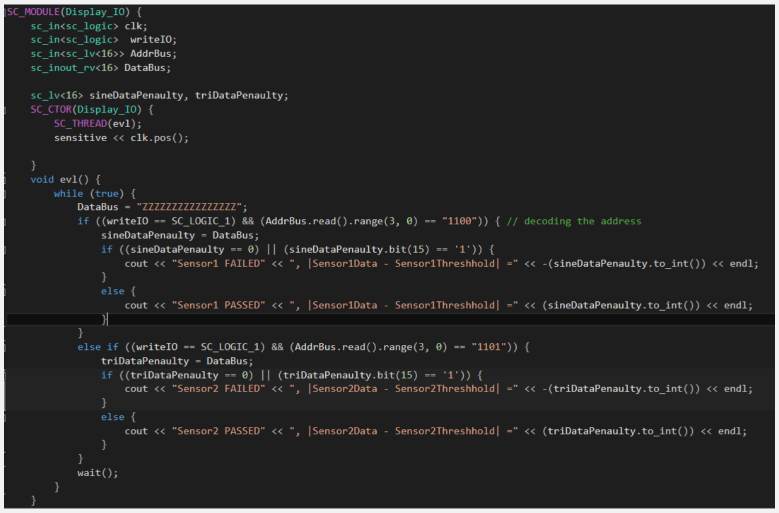
This IO device is a timer that has a start timer input and whenever a positive edge detected on this port the timer starts waiting for 1 ms and after that time interval the time out output will be issued.



```
SC_MODULE(Timer_IO) {
    sc_in<sc_logic> startTimer;
    sc_out<sc_logic> timeOut;

    SC_CTOR(Timer_IO) {
        SC_THREAD(evl);
        sensitive << startTimer.pos();
    }
    void evl() {
        while (true) {
            timeOut = SC_LOGIC_0;
            wait(1, SC_MS);
            timeOut = SC_LOGIC_1;
            wait();
        }
    }
};
```

Display IO

For this device in the ISS part I have a module that has interface with bus and it has to display the data that is send to it when the write IO and the address has the special values.



```
SC_MODULE(Display_IO) {
    sc_in<sc_logic> clk;
    sc_in<sc_logic>  writeIO;
    sc_in<sc_lv<16>> AddrBus;
    sc_inout_rv<16> DataBus;

    sc_lv<16> sineDataPenaulty, triDataPenaulty;
    SC_CTOR(Display_IO) {
        SC_THREAD(evl);
        sensitive << clk.pos();

    }
    void evl() {
        while (true) {
            DataBus = "ZZZZZZZZZZZZZZZZ";
            if ((writeIO == SC_LOGIC_1) && (AddrBus.read().range(3, 0) == "1100")) { // decoding the address
                sineDataPenaulty = DataBus;
                if ((sineDataPenaulty == 0) || (sineDataPenaulty.bit(15) == '1')) {
                    cout << "Sensor1 FAILED" << ", |Sensor1Data - Sensor1Threshhold| =" << -(sineDataPenaulty.to_int()) << endl;
                }
                else {
                    cout << "Sensor1 PASSED" << ", |Sensor1Data - Sensor1Threshhold| =" << (sineDataPenaulty.to_int()) << endl;
                }
            }
            else if ((writeIO == SC_LOGIC_1) && (AddrBus.read().range(3, 0) == "1101")) {
                triDataPenaulty = DataBus;
                if ((triDataPenaulty == 0) || (triDataPenaulty.bit(15) == '1')) {
                    cout << "Sensor2 FAILED" << ", |Sensor2Data - Sensor2Threshhold| =" << -(triDataPenaulty.to_int()) << endl;
                }
                else {
                    cout << "Sensor2 PASSED" << ", |Sensor2Data - Sensor2Threshhold| =" << (triDataPenaulty.to_int()) << endl;
                }
            }
            wait();
        }
    }
```

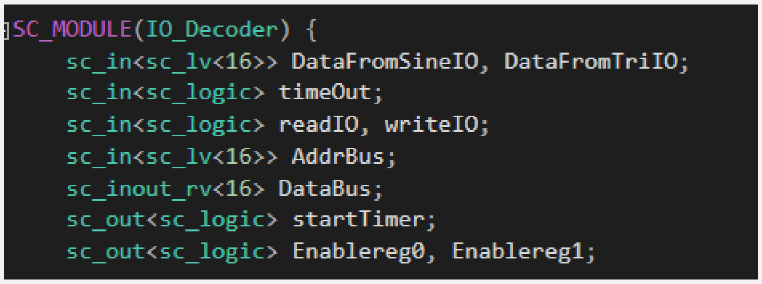
This device has 2 places in memory address for its inputs and by noticing the difference between these two addresses it recognizes the IO device that sent the data to the processor.

In the systemC bracketing there is no separated module for Display and it is only some print statements in the processing function.

Bus interface

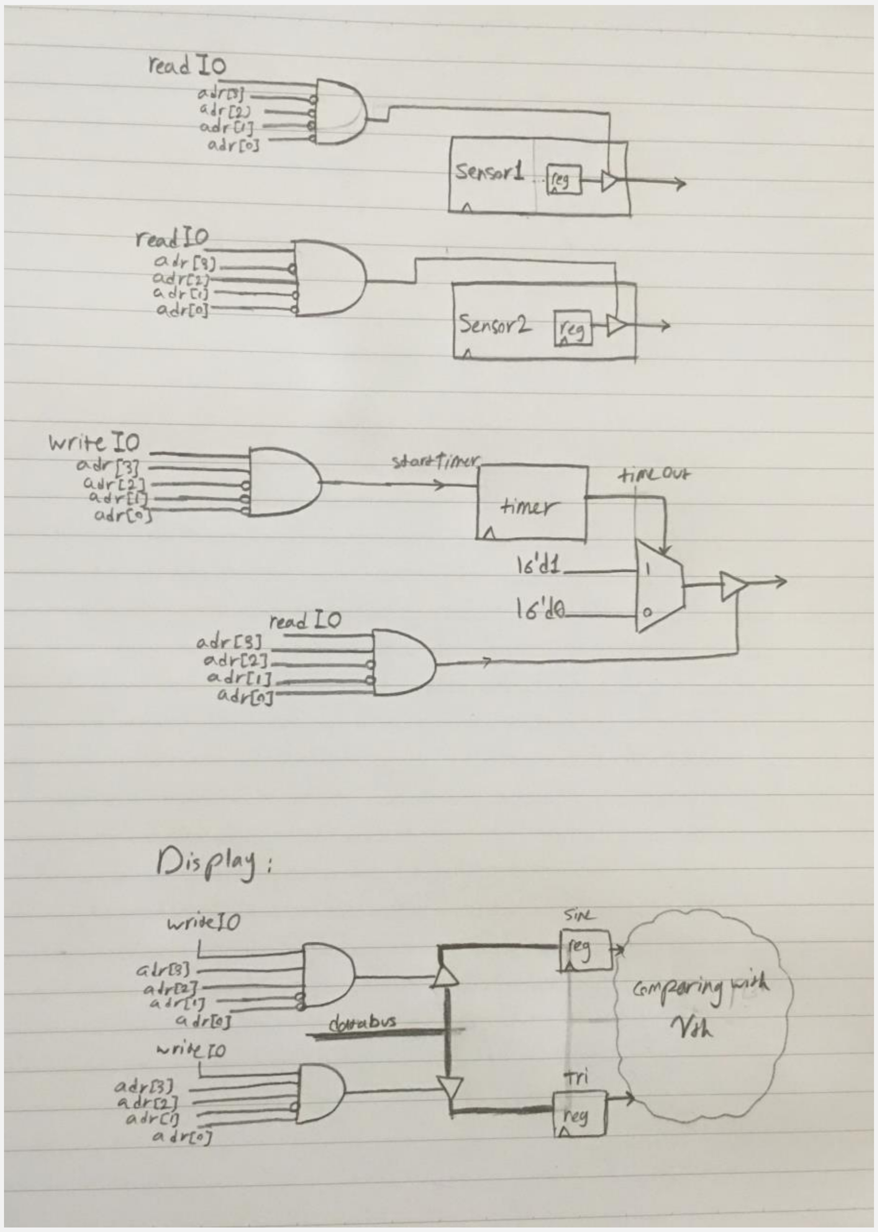
There is a module that generates the bus interfaces. For sensor 1 I used address FX10 for the data of the sensor1's output. For sensor 2 I used address FX14 for the data of the sensor2's output. For the timer I used Fx18 for startTimer and Fx19 for achieving the value of timeOut port.

The Display device uses the address Fx1C and Fx1D for saving its input data. In this module I didn't implement the Display interface and I did it in the display module in order to use bus interface module in both ISS and Bracketing.

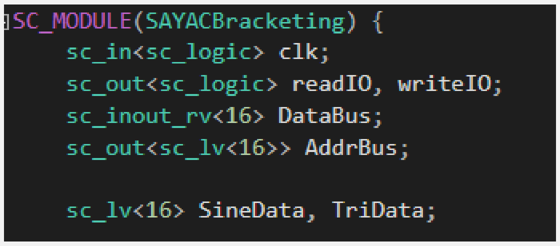


```
SC_MODULE(IO_Decoder) {
    sc_in<sc_lv<16>> DataFromSineIO, DataFromTriIO;
    sc_in<sc_logic> timeOut;
    sc_in<sc_logic> readIO, writeIO;
    sc_in<sc_lv<16>> AddrBus;
    sc_inout_rv<16> DataBus;
    sc_out<sc_logic> startTimer;
    sc_out<sc_logic> Enablereg0, Enablereg1;
```

In this picture you can see the schematics of the bus interface:

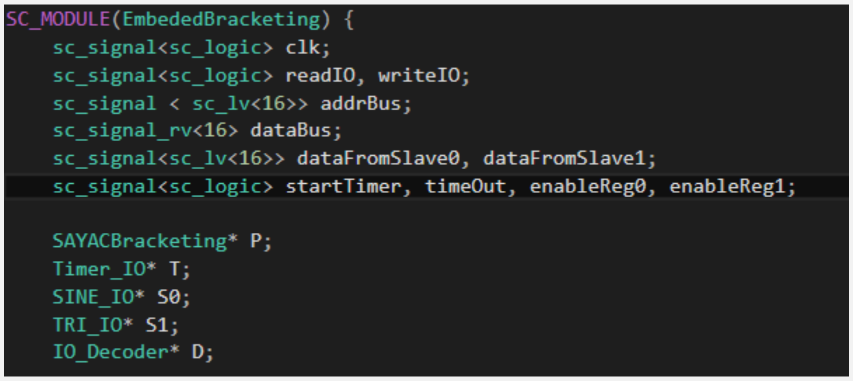


For this part I have a module named SAYACBracketing that plays the role of a processor that has its program with writeIO, readIO, DataBus and AddressBus ports.



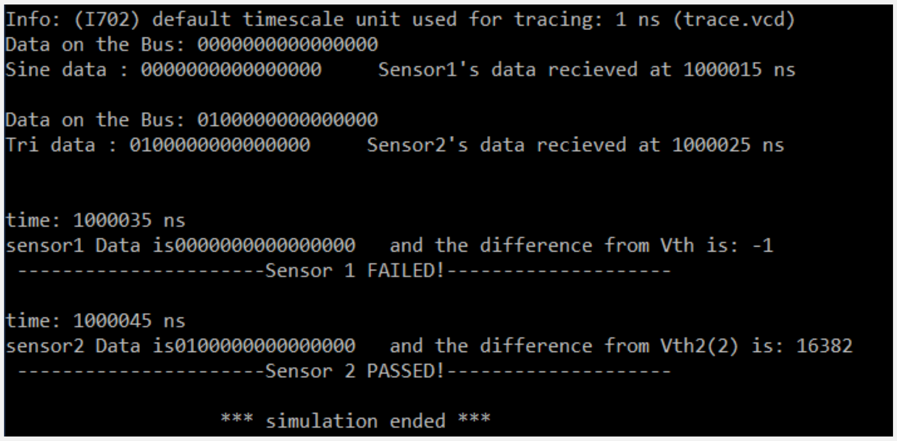
```
SC_MODULE(SAYACBracketing) {
    sc_in<sc_logic> clk;
    sc_out<sc_logic> readIO, writeIO;
    sc_inout_rv<16> DataBus;
    sc_out<sc_lv<16>> AddrBus;

    sc_lv<16> SineData, TriData;
```

In the main module named EmbededBracketing I used the SAYACBracketing and decoder and IO devices to run the program.



```
SC_MODULE(EmbededBracketing) {
    sc_signal<sc_logic> clk;
    sc_signal<sc_logic> readIO, writeIO;
    sc_signal < sc_lv<16>> addrBus;
    sc_signal_rv<16> dataBus;
    sc_signal<sc_lv<16>> dataFromSlave0, dataFromSlave1;
    sc_signal<sc_logic> startTimer, timeOut, enableReg0, enableReg1;

    SAYACBracketing* P;
    Timer_IO* T;
    SINE_IO* S0;
    TRI_IO* S1;
    IO_Decoder* D;
```

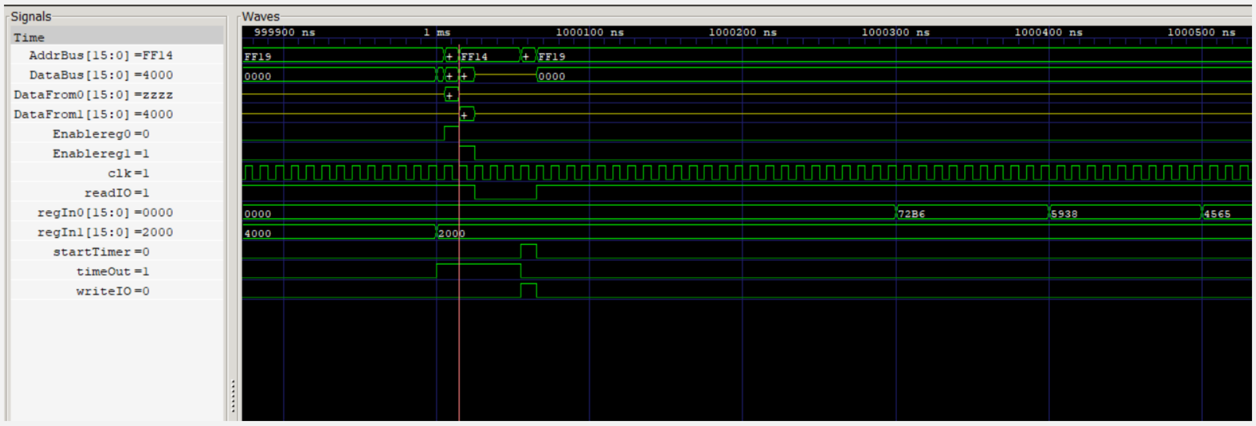
The result of this program is shown in the following picture:



```
Info: (I702) default timescale unit used for tracing: 1 ns (trace.vcd)
Data on the Bus: 0000000000000000
Sine data : 0000000000000000     Sensor1's data recieved at 1000015 ns

Data on the Bus: 0100000000000000
Tri data : 0100000000000000     Sensor2's data recieved at 1000025 ns


time: 1000035 ns
sensor1 Data is0000000000000000   and the difference from Vth is: -1
 ----------------------Sensor 1 FAILED!--------------------

time: 1000045 ns
sensor2 Data is0100000000000000   and the difference from Vth2(2) is: 16382
 ----------------------Sensor 2 PASSED!--------------------

                   *** simulation ended ***
```

As you can see the result of sensor 1 failed and the result of sensor 2 passed.

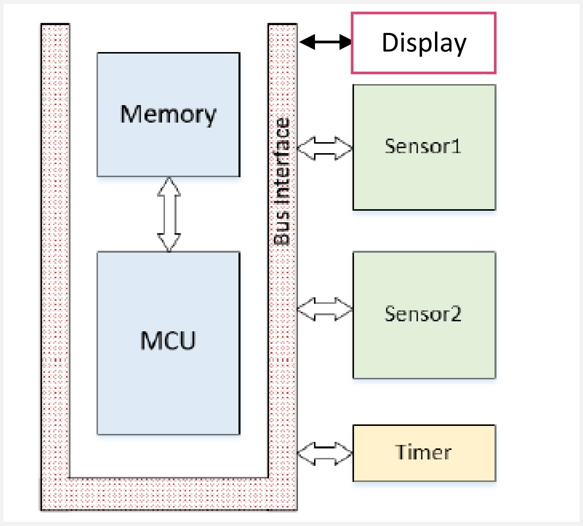
This is the vcd file that shows the signals and timings:



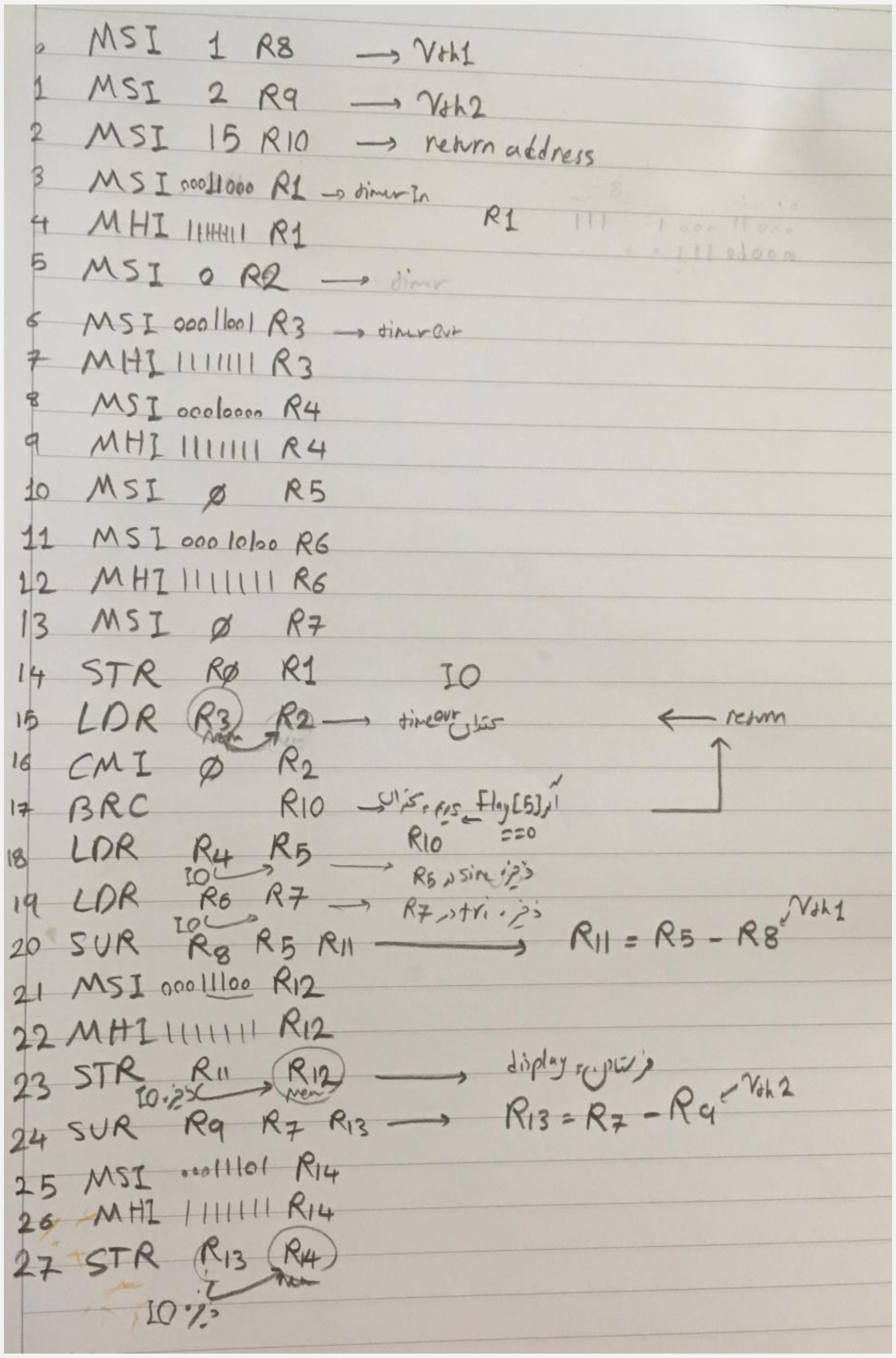
```
Time
AddrBus[15:0] =FF14
DataBus[15:0] =4000
DataFrom0[15:0] =zzzz
DataFrom1[15:0] =4000
Enablereg0 =0
Enablereg1 =1
clk =1
readIO =1
regIn0[15:0] =0000
regIn1[15:0] =2000
startTimer =0
timeOut =1
writeIO =0
```

Processor ISS

For this part I used the SAYAC processor ISS codes as the processor and I wired the IO devices with it such as the following picture:

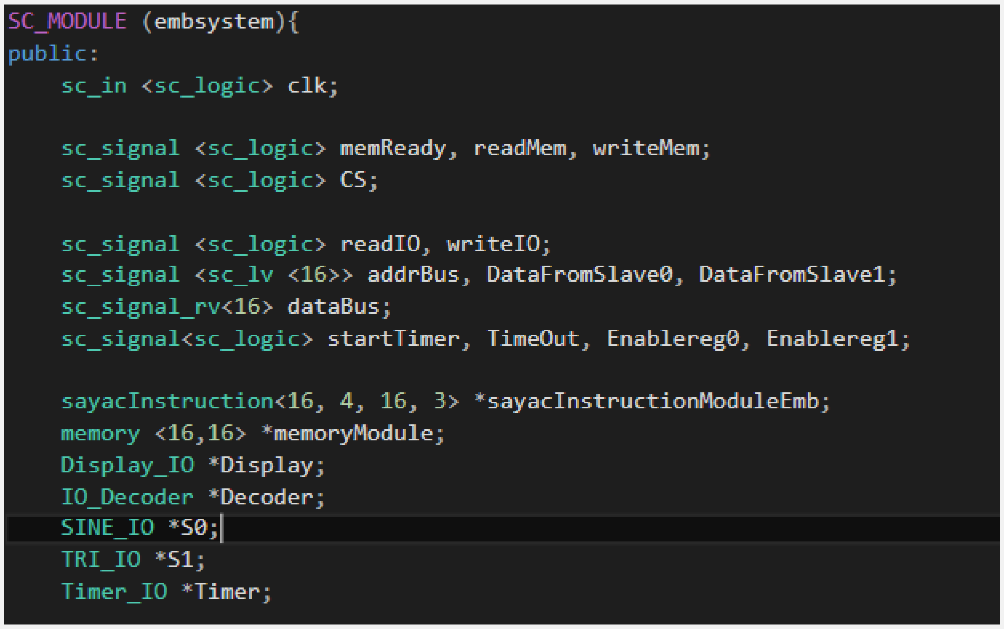


The program that the project wants us to run on the processor is in the following picture and its functionality is the same as the previous part, so we expect to have the same outputs:



```
0  MSI 1 R8              → Vth1
1  MSI 2 R9              → Vth2
2  MSI 15 R10            → return address
3  MSI 00011000 R1       → timer In
4  MHI 1111111 R1                      R1
5  MSI 0 R2              → timer
6  MSI 00011001 R3       → timer Out
7  MHI 1111111 R3
8  MSI 00010000 R4
9  MHI 1111111 R4
10 MSI Ø R5
11 MSI 00010100 R6
12 MHI 1111111 R6
13 MSI Ø R7
14 STR RØ R1                  IO
15 LDR R3 R2             → timeout کنترل            ← return
       mem                                               ↑
16 CMI Ø R2                                              |
17 BRC   R10             → اگر Flag[5] ==0 برو به R10 ———┘
18 LDR R4 R5             → ذخیرهٔ sine در R5
       IO
19 LDR R6 R7             → ذخیرهٔ tri در R7
       IO
20 SUR R8 R5 R11         ————→  R11 = R5 - R8  ← Vth1
21 MSI 00011100 R12
22 MHI 1111111 R12
23 STR R11 R12           ——→  فرستادن به display
       IO به    mem
24 SUR R9 R7 R13         ——→  R13 = R7 - R9  ← Vth2
25 MSI 00011101 R14
26 MHI 1111111 R14
27 STR R13 R14
           mem
   IO به
```

In the following module the whole circuit is wired together and in the test bench the clock is generated.



```
SC_MODULE (embsystem){
public:
    sc_in <sc_logic> clk;

    sc_signal <sc_logic> memReady, readMem, writeMem;
    sc_signal <sc_logic> CS;

    sc_signal <sc_logic> readIO, writeIO;
    sc_signal <sc_lv <16>> addrBus, DataFromSlave0, DataFromSlave1;
    sc_signal_rv<16> dataBus;
    sc_signal<sc_logic> startTimer, TimeOut, Enablereg0, Enablereg1;

    sayacInstruction<16, 4, 16, 3> *sayacInstructionModuleEmb;
    memory <16,16> *memoryModule;
    Display_IO *Display;
    IO_Decoder *Decoder;
    SINE_IO *S0;
    TRI_IO *S1;
    Timer_IO *Timer;
```

In this part we have the results of some last instructions:

```
*************BRC Instruction*************
regFile p1:  0000000000001111
regFile p2:  0000000000001111
Case 1
memReady Ready is 1
MEM Ready before  is: 0
memReady Ready is 0
addrBus   0000000000010010
MEM Ready before IR is:1
Ir IS:    0010001001000101
*************Ldr Instruction*************
regFile p1:  1111111100010000
regFile p2:  1111111100010000
memReady Ready is 1
memReady Ready is 0
[adr]   5
RegFile Write Data Is:  0000000000000000Time :   1000110 ns
MEM Ready before  is: 0
memReady Ready is 0
addrBus   0000000000010011
MEM Ready before IR is:1
Ir IS:    0010001001100111
**************Ldr Instruction**************
regFile p1:  1111111100010100
regFile p2:  1111111100010100
memReady Ready is 1
memReady Ready is 0
[adr]   7
RegFile Write Data Is:  0100000000000000Time :   1000150 ns
MEM Ready before  is: 0
memReady Ready is 0
addrBus   0000000000010100
MEM Ready before IR is:1
Ir IS:    1010100001011011
**************SUR Instruction**************
regFile p1:  0000000000000000
regFile p2:  0000000000000001
[adr]   11
RegFile Write Data Is:  1111111111111111Time :   1000170 ns
memReady Ready is 1
MEM Ready before  is: 0
memReady Ready is 0
addrBus   0000000000010101
MEM Ready before IR is:1
```

```
Ir IS:    0101000111001100
*************MSI Instruction*************
signEximm is:  0000000000011100
[adr]   12
RegFile Write Data Is:  0000000000011100Time :   1000190 ns
memReady Ready is 1
MEM Ready before  is: 0
memReady Ready is 0
addrBus   0000000000010110
MEM Ready before IR is:1
Ir IS:    0110111111111100
*************MHI Instruction*************
regFile p1:  0000000000011100
regFile p2:  0000000000011100
[adr]   12
RegFile Write Data Is:  1111111100011100Time :   1000210 ns
memReady Ready is 1
MEM Ready before  is: 0
memReady Ready is 0
addrBus   0000000000010111
MEM Ready before IR is:1
Ir IS:    0010011010111100
*************STR Instruction*************
regFile p1:  1111111100011100
regFile p2:  1111111111111111
memReady Ready is 1
memReady Ready is 0
Sensor1 FAILED, |Sensor1Data - Sensor1Threshhold| =1
MEM Ready before  is: 0
memReady Ready is 0
```

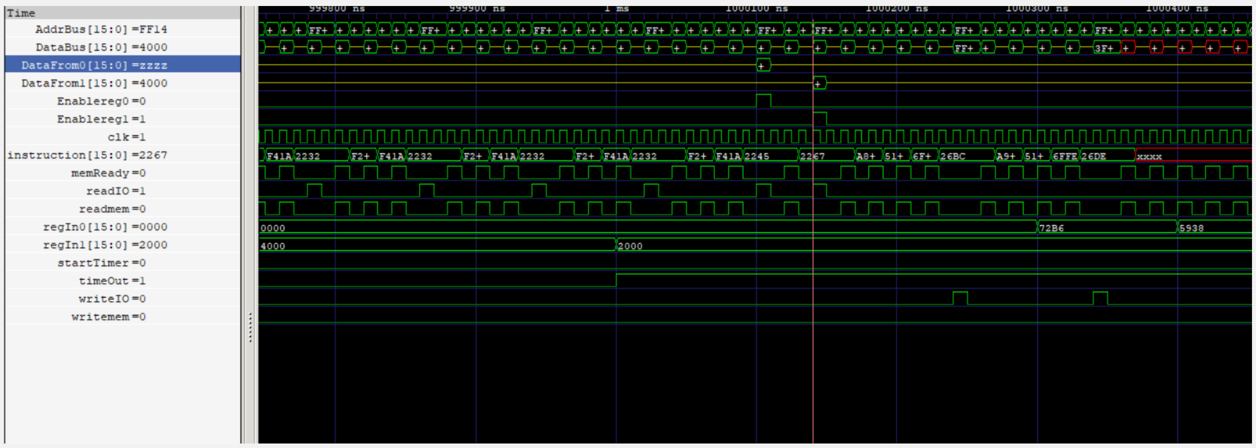
```
addrBus   0000000000011000
MEM Ready before IR is:1
Ir IS:    1010100101111101
**************SUR Instruction**************
regFile p1:  0100000000000000
regFile p2:  0000000000000010
[adr]   13
RegFile Write Data Is:  0011111111111110Time :   1000270 ns
memReady Ready is 1
MEM Ready before  is: 0
memReady Ready is 0
addrBus   0000000000011001
MEM Ready before IR is:1
Ir IS:    0101000111011110
**************MSI Instruction**************
signEximm is:  0000000000011101
[adr]   14
RegFile Write Data Is:  0000000000011101Time :   1000290 ns
memReady Ready is 1
MEM Ready before  is: 0
memReady Ready is 0
addrBus   0000000000011010
MEM Ready before IR is:1
Ir IS:    0110111111111110
**************MHI Instruction**************
regFile p1:  0000000000011101
regFile p2:  0000000000011101
[adr]   14
RegFile Write Data Is:  1111111100011101Time :   1000310 ns
memReady Ready is 1
```

```
MEM Ready before  is: 0
memReady Ready is 0
addrBus   0000000000011011
MEM Ready before IR is:1
Ir IS:    0010011011011110
*************STR Instruction*************
regFile p1:  1111111100011101
regFile p2:  0011111111111110
memReady Ready is 1
memReady Ready is 0
Sensor2 PASSED, |Sensor2Data - Sensor2Threshhold| =16382
MEM Ready before  is: 0
memReady Ready is 0
addrBus   0000000000011100
MEM Ready before IR is:1
Ir IS:    XXXXXXXXXXXXXXXX
```

The results that we can see on display are highlighted and as you can see these results are the same as systemC bracketing.

As you can see the result of sensor 1 failed and the result of sensor 2 passed.

This is the vcd file that shows the signals and timings:


Time
AddrBus[15:0] =FF14
DataBus[15:0] =4000
DataFrom0[15:0] =zzzz
DataFrom1[15:0] =4000
Enablereg0 =0
Enablereg1 =1
clk =1
instruction[15:0] =2267
memReady =0
readIO =1
readmem =0
regIn0[15:0] =0000
regIn1[15:0] =2000
startTimer =0
timeOut =1
writeIO =0
writemem =0