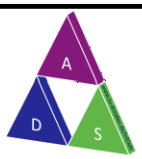


Prof. Dr.-Ing. Diana Göhringer

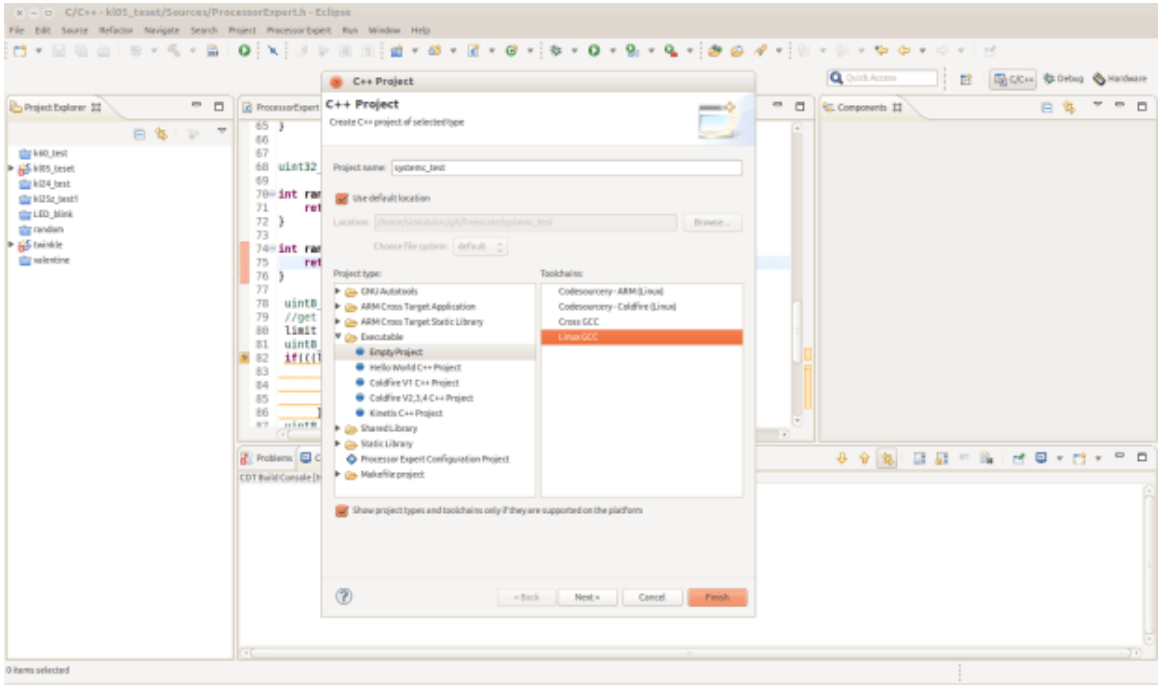
Hardware Modelling and Simulation

Lab 10

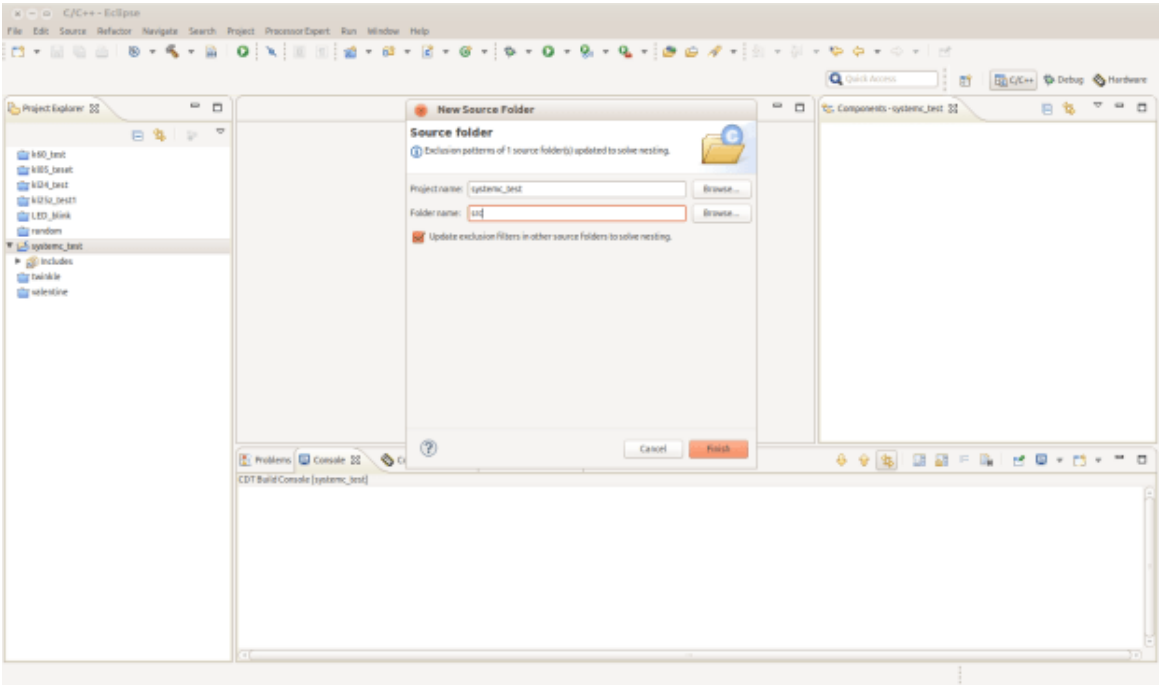
Chair of Adaptive Dynamic Systems

Introduction into SystemC design flow

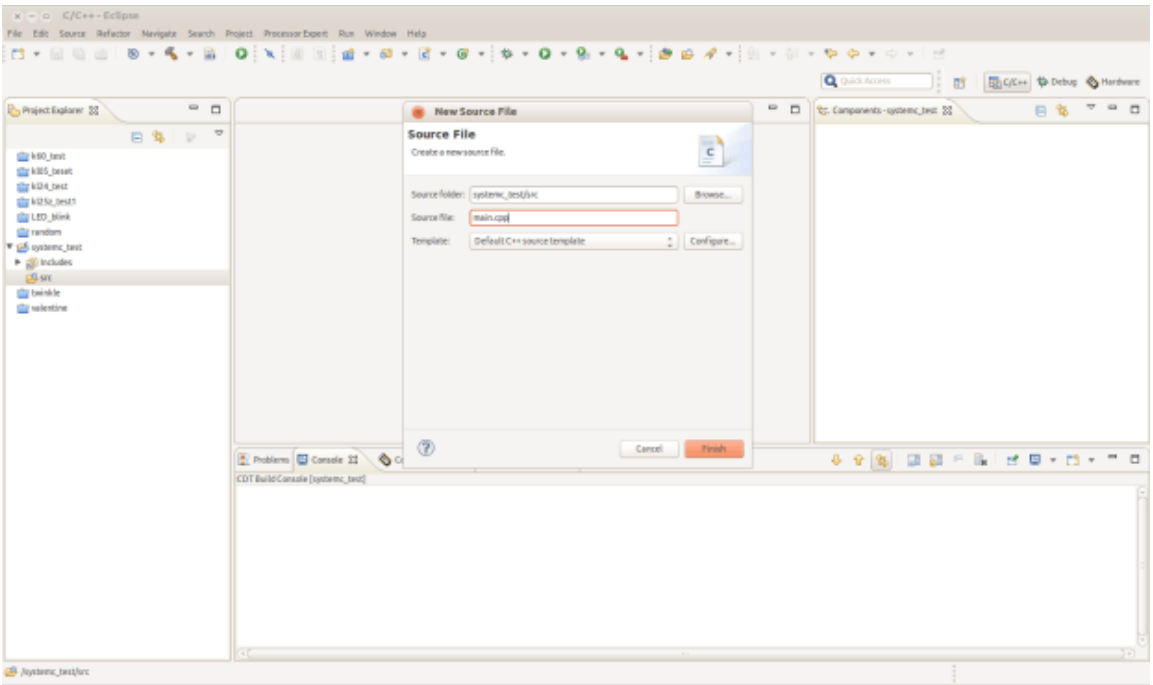
1. Open Eclipse and create a new C++ project, under ***New*** menu, Eclipse new C/C++ project. Then select Linux GCC beyond the Toolchain.

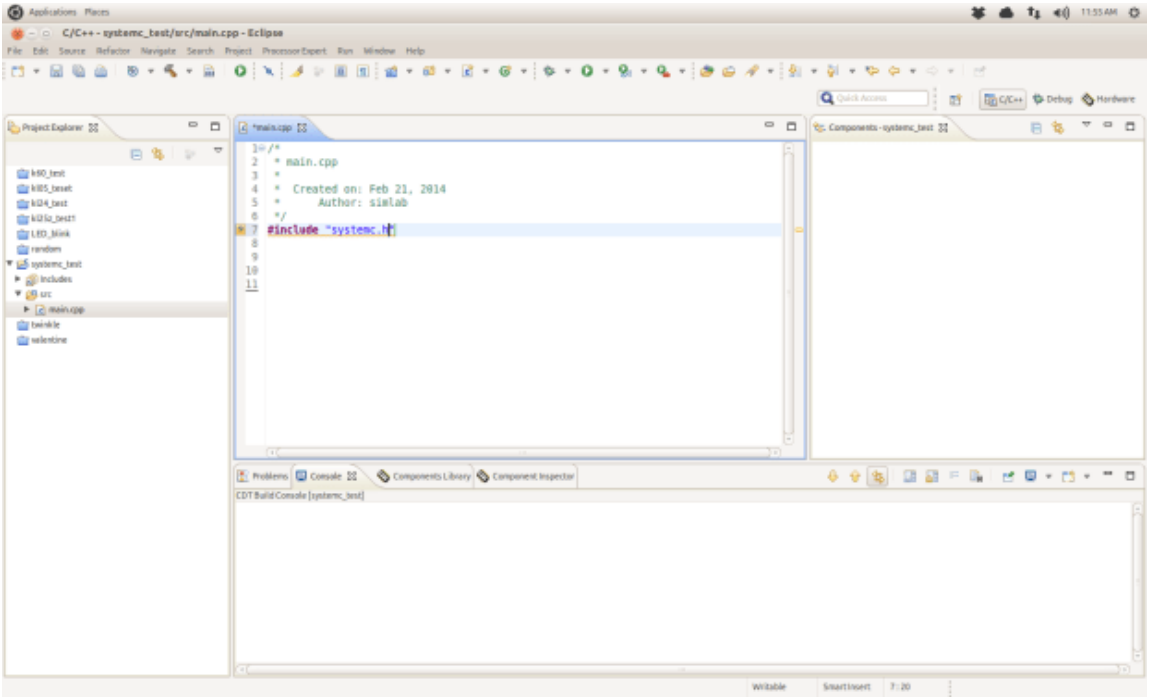


2. Create a new source folder ***src*** in your project



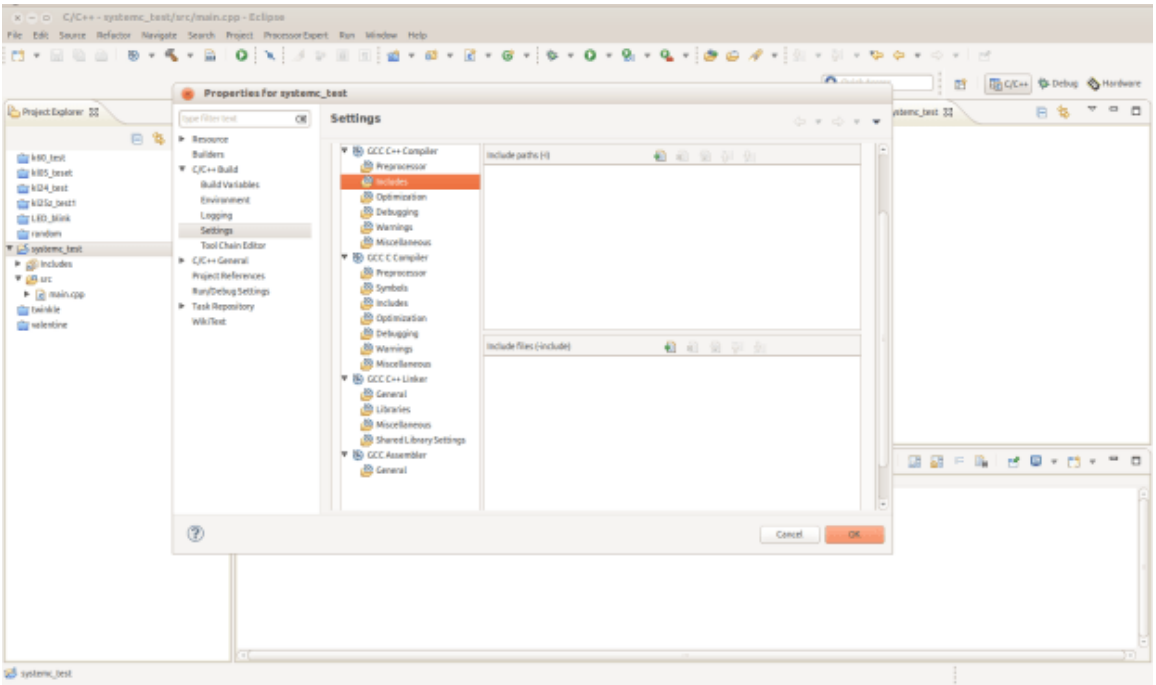
3. Create inside the new folder a new file main.cpp



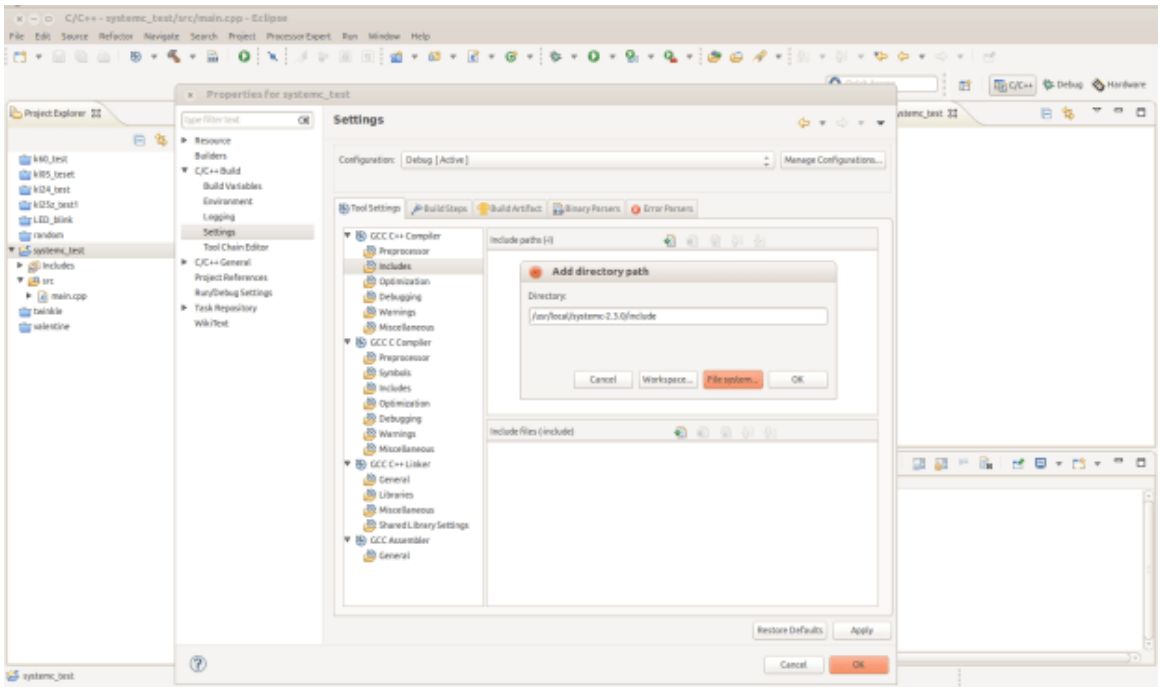


4. Configure systemc library Path for the project

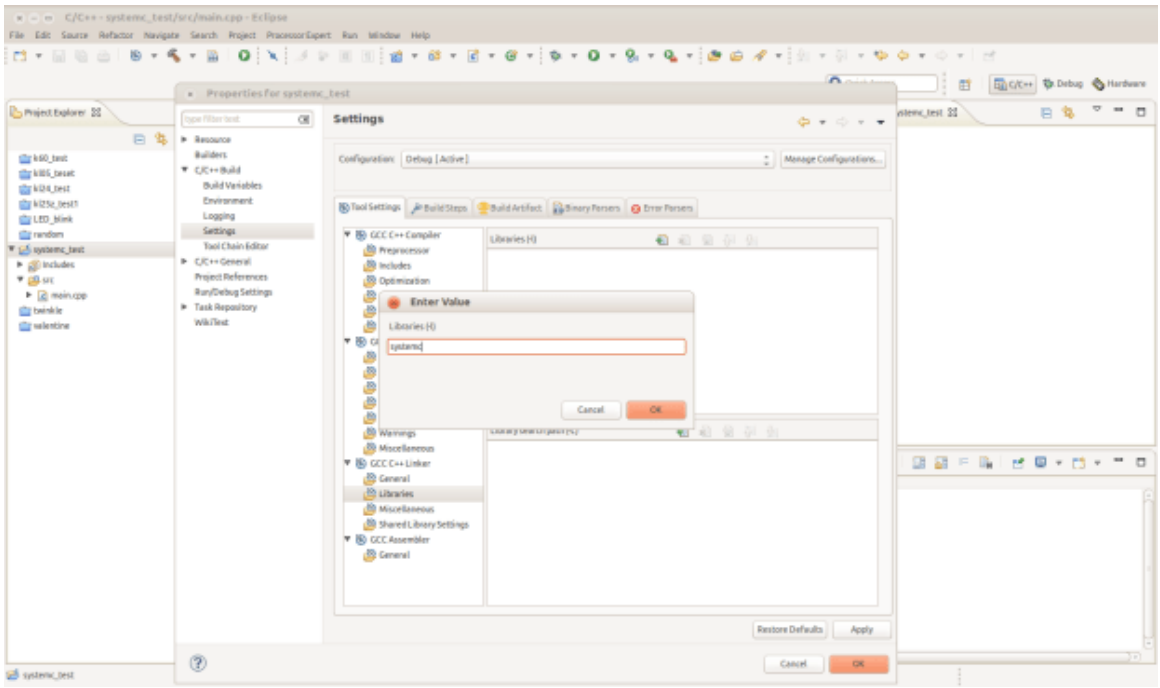
Right Click on the project root folder in the project explorer view and click on ***properties***, alternatively, go to the ***Project*** menu and click ***properties***.



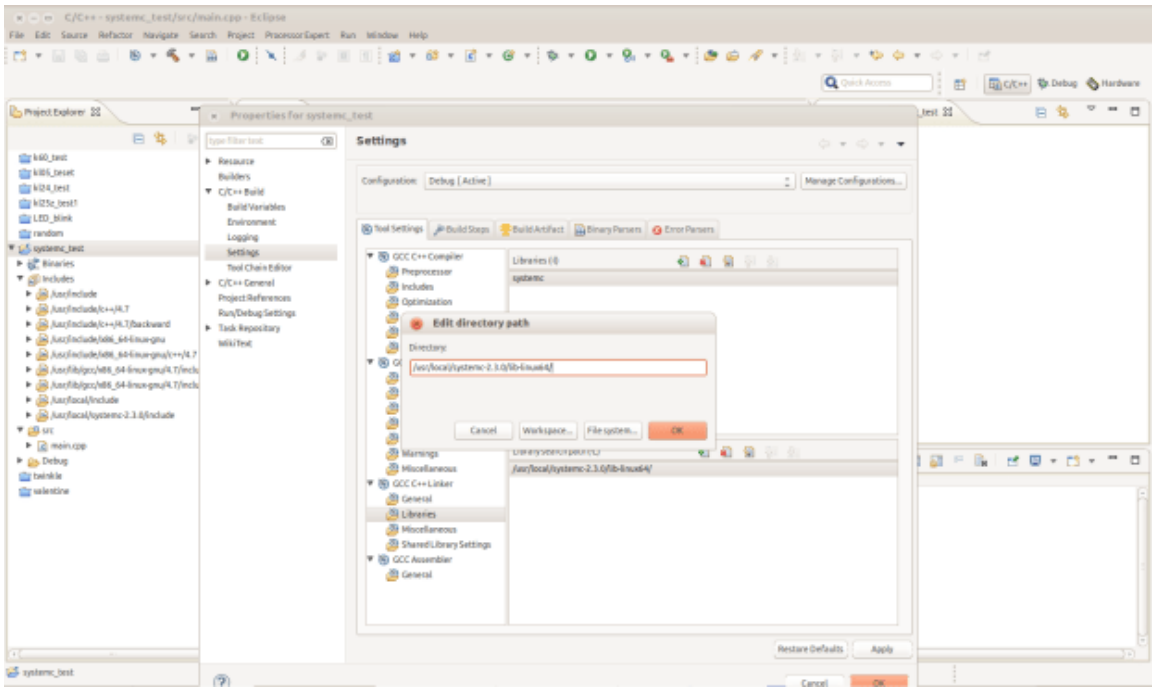
Expand the ***C/C++ Build*** entry, and click on ***settings***. Under ***tool settings*** tab, expand ***GCC C++ Compiler*** and click on ***includes***. On the right under ***include paths(-i)***, click the + icon and add a path to ***/usr/local/systemc-2.3.2/include/*** by browsing ***File System.***

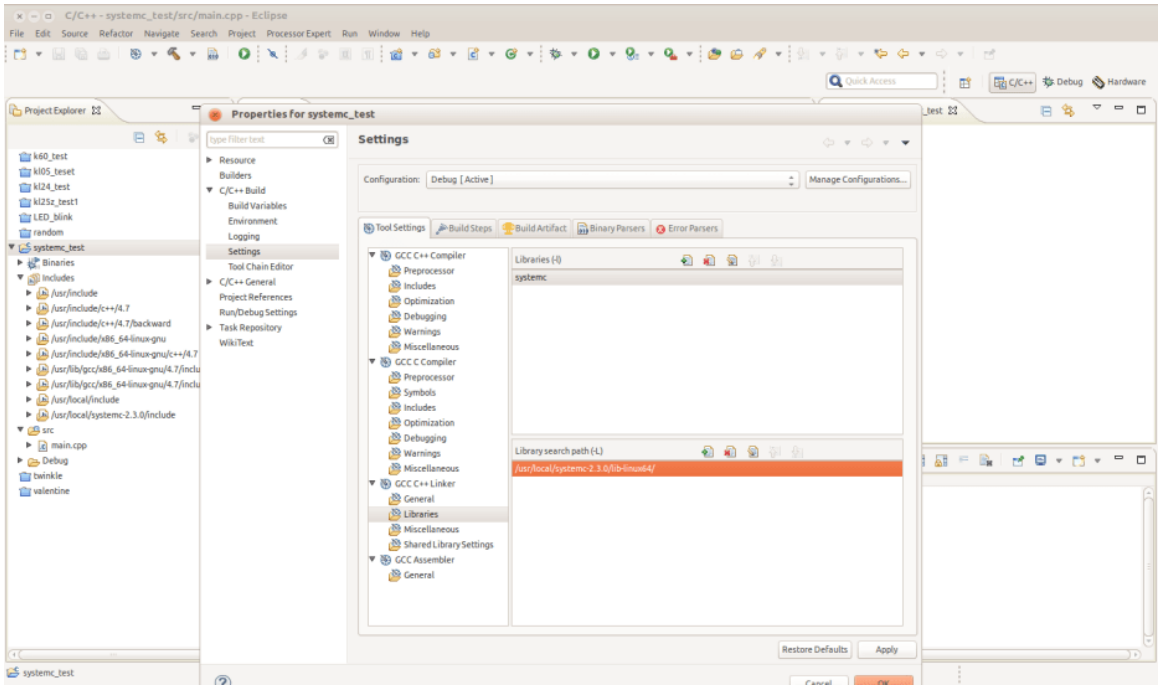


Now expand ***GCC C++ Linker*** and under ***Libraries***, at the top ***Libraries(-l)*** entry, click on the + icon and add the library name “systemc”



then at the bottom ***Library search path***, click on the + icon and on the pop-up, add the path to the installation directory ***/usr/local/systemc-2.3.2/lib-linux64*** for 64-bit.

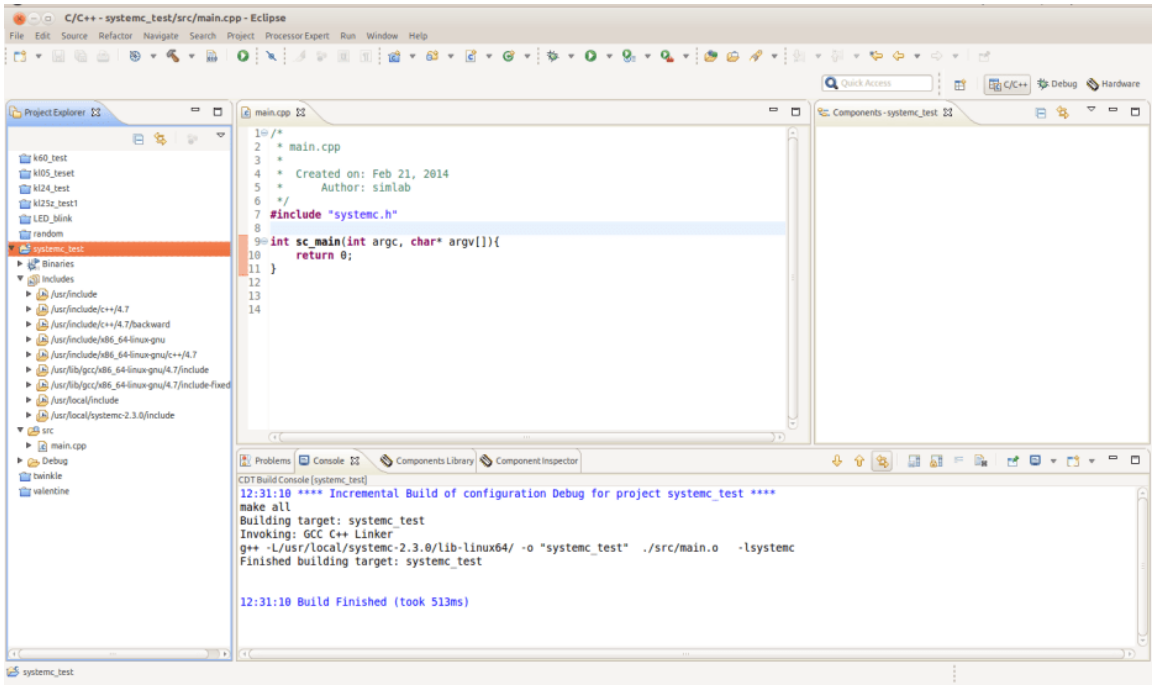

Properties for systemc_test
Settings
Edit directory path
Directory:
Cancel
Workspace...
File system...
OK
Restore Defaults
Apply


Properties for systemc_test
Settings
Configuration: Debug [Active]
Manage Configurations...
Tool Settings
Build Steps
Build Artifact
Binary Parsers
Error Parsers
GCC C++ Compiler
GCC C Compiler
GCC C++ Linker
GCC Assembler
Libraries (-l)
systemc
Library search path (-L)
/usr/local/systemc-2.3.0/lib-linux64/
Restore Defaults
Apply

5. Insert the following code example into your main.cpp file.

```
#include “systemc.h”
int sc_main(int argc, char* argv[])
{
cout << “Hello World.” << endl;
return 0;
}
```

6. Click on the Build symbol (hummer icon) and if everything is ok, the terminal will show “finished building target”. Then, click on the Run symbol (play icon) to run your code.



Traffic light modelling in SystemC

The traffic light is an important application that is used on daily basis. The task is to control the traffic lights of an intersection shown in Figure 1.

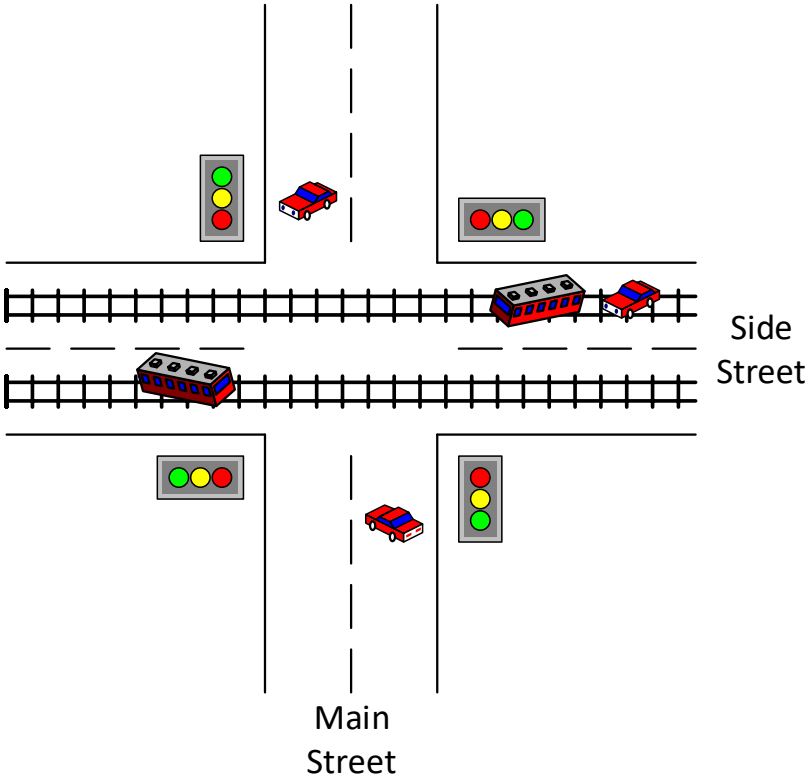


Figure 1 Road intersection

Vehicle Density: The number of vehicles in the surveillance area.

The following block diagram shows the traffic light model (trafficlight).

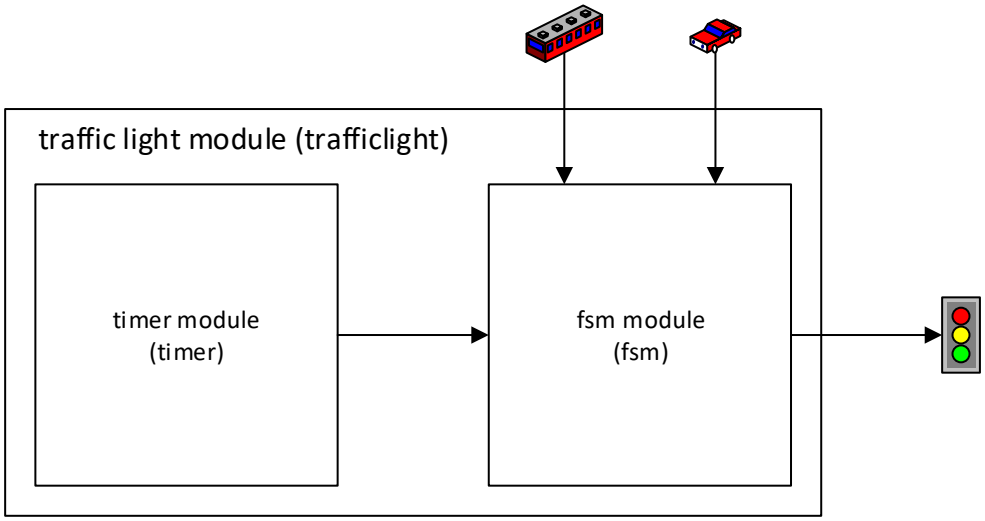


Figure 2 Design of the traffic light module

The timer generates signals for the FSM. It will control the traffic lights based on the given input signals from the surveillance area and the timer. The designer will be able to simulate the behavior of the traffic light model with SystemC.

The application is divided into three portions (timer, FSM, trafficlight) for the implementation.

Task 1 Implementation of the timer module

At the beginning, the timer needs to be implemented. The timer consists of different blocks in SystemC.

The following block diagram shows the internal blocks of the timer and its connections. In this case, the timer module (timer) and timer testbench (tb_timer) are parts of the top module design (tb_timer_top) and they get connected through appropriate interfaces.

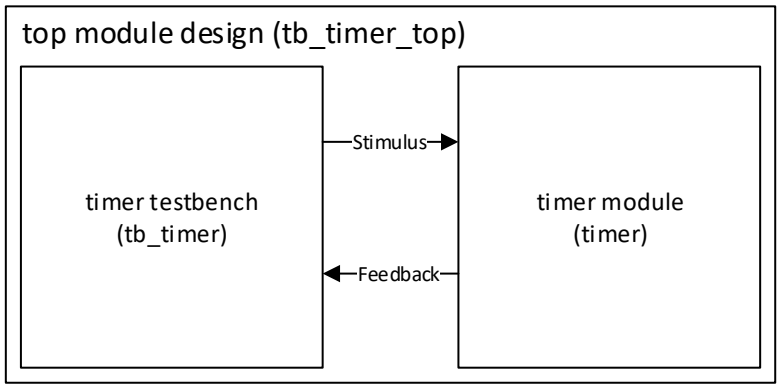


Figure 3 Design of the whole timer module

1. First of all, the timer module needs to be implemented. It has the following interfaces:

```
sc_in_clk          clk;          // clock counts every second
sc_in <bool>       reset;        // reset the time module
sc_out<bool> timer_30sec;        // signals every 30 seconds
sc_out<bool> timer_60sec;        // signals every 60 seconds
sc_out<bool> timer_120sec;       // signals every 120 seconds
```

Furthermore, the following hints should help you in your implementation:

SC_MODULE(timer), SC_THREAD, SC_CTOR and sensitivity list.

Please add the function dont_initialize() to the end of the constructor.

To test the functionalities of your developed code in SystemC, you need a testbench. This testbench generates signals that stimulate the developed modules.

2. You need to implement a suitable timer testbench (tb_timer) to evaluate the functionalities of your timer module (timer). Your implementation for the testbench should contain the following interfaces:

```
sc_in_clk          clk;
sc_in<bool> timer_30sec;
sc_in<bool> timer_60sec;
sc_in<bool> timer_120sec;
sc_out<bool> reset;
```

3. Use the top module design (tb_timer_top) to simulate the timer testbench (tb_timer) with your implemented timer code (timer). You need to instantiate the top module design (tb_timer_top) in the main function (sc_main). Set up the time resolution of the simulation to 100 ms and simulate the behavior of the timer for 125 s. You can find the top module timer testbench in the given source files.

   Besides, you should test your module with meaningful stimulus signals and compare the feedback process (tb_timer).

Additional information: During the simulation you will get some feedback signals from the timer module which needs to be proven by the feedback process in testbench or checking manually the waveforms through the trace function called sc_trace().

Task 2 Implementation of the FSM module

A finite state-machine (FSM) is required to control the traffic lights. There are two states:

1. Main Street: green and Side Street: red
2. Main Street: red and Side Street: green

Furthermore, it has the following three conditions for state transition:

1. The main street traffic light turns from red to green when one of the following conditions is fulfilled:
    a. the vehicle density of main street is greater than to the vehicle density of side street.
    b. the side street traffic light is green for 30s and no tram is on the side street.
    c. the vehicle density of side street is zero.
2. The side street traffic light turns from red to green when one of the following conditions is fulfilled:
    a. the vehicle density of the side street is greater than the vehicle density of main street and the main street traffic light is green for longer than one minute.
    b. the vehicle density of the side street is greater than zero and the main street traffic light is green for longer than two minutes.
    c. there are trams in the surveillance area.
3. All traffic lights should stay red for 5s and change then its signals from red to green if a transition of the traffic light is required.

The following block diagram shows the connections between the FSM module (fsm) and the FSM testbench (tb_fsm) which are parts of the top module design (tb_fsm_top).

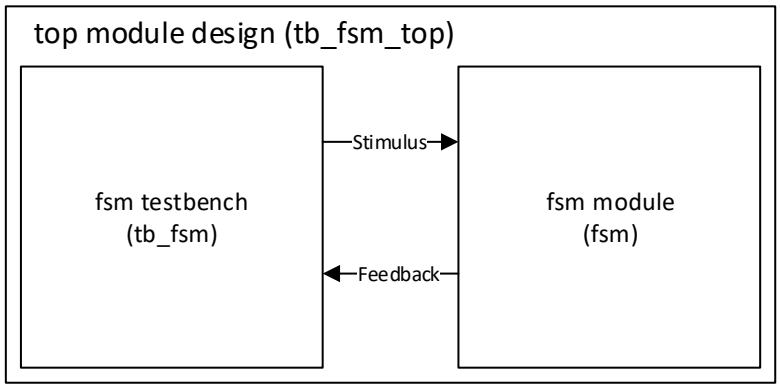


Figure 4 Design of the whole FSM module

1. The FSM module needs to be implemented with the following interfaces:

sc_in_clk	clk;
sc_in<bool>	timer_30sec;
sc_in<bool>	timer_60sec;
sc_in<bool>	timer_120sec;
sc_in<int>	counter_tram;
sc_in<int>	counter_cars_mainstreet;
sc_in<int>	counter_cars_sidestreet;
sc_out<bool>	trafficlight_mainstreet;
sc_out<bool>	trafficlight_sidestreet;
sc_out<bool>	timer_reset;

Furthermore, the following hints should help you in your implementation:

SC_MODULE(fsm), SC_THREAD, SC_CTOR and sensitivity list.

Please add the function dont_initialize() to the end of the constructor.

2. To test the functionalities of your FSM module, you need to implement a testbench. Your implementation for the FSM testbench (tb_fsm) should contain the following interfaces:

sc_in_clk	clk;
sc_out<bool>	timer_30sec;
sc_out<bool>	timer_60sec;
sc_out<bool>	timer_120sec;
sc_out<int>	counter_tram;
sc_out<int>	counter_cars_mainstreet;
sc_out<int>	counter_cars_sidestreet;
sc_in<bool>	trafficlight_mainstreet;
sc_in<bool>	trafficlight_sidestreet;
sc_in<bool>	timer_reset;

3. Now, the top module of the FSM testbench (tb_fsm_top) needs to be implemented so that the functionalities of the FSM module (fsm) can be tested through the FSM testbench (tb_fsm).

Besides, you should test your module with meaningful stimulus signals to ensure that all transitions are correctly implemented. You need to instantiate the top module design (tb_fsm_top) in the main function (sc_main).

Task 3 Implementation of the traffic light module

1. Integrate the timer module (timer) and FSM module (fsm) in the traffic light module (trafficlight) as shown in Figure 2. In doing so, connect the signals with the both modules through the following interfaces:

sc_in_clk	clk;
sc_in<int>	counter_tram;
sc_in<int>	counter_cars_mainstreet;
sc_in<int>	counter_cars_sidestreet;
sc_out<bool>	trafficlight_mainstreet;
sc_out<bool>	trafficlight_sidestreet;

2. Write a testbench to evaluate the behavior of the whole system. In doing so, you should define suitable interfaces for the simulation. The following block diagram shows the connections between the traffic light module (trafficlight) and the traffic light testbench (tb_trafficlight) which are parts of the top module design (tb_trafficlight_top).

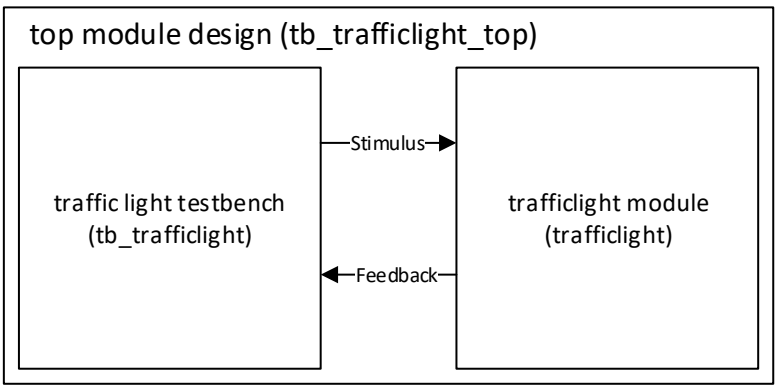


Figure 5 Design of the whole traffic light module

3. Now, you should implement the top module design of the traffic light (tb_trafficlight_top) and insert the traffic light testbench (tb_trafficlight) to evaluate the functionalities of your traffic light module (trafficlight). Add all implemented testbenches in the main.cpp and simulate it with meaningful signals.