University of Stuttgart
Germany

Institute of Computer Architecture
and Computer Engineering

Dept. of Embedded Systems

Modeling, Simulation, and Specification

Prof. Dr. Martin Radetzki

Exercise 3: *Basic SystemC Modeling / Simulating KPN with SystemC*

Download the source files for this exercise (exercise3.zip) from the course webpage. If you wish to prepare the exercise at home, you need to download and build (compile) the SystemC library from www.systemc.org, using one of the supported platforms and tools. In the computer pool (see file exercise_preparations.pdf), the SystemC library is pre-compiled and you can use it with the GNU tools under Linux.

1. Basic SystemC Modeling

   In directory directory `ex3_1` you will find a model of a simple adder, a simple test bench and the associated main program file.

   a) Build the given example using the provided Makefile and simulate the system.

   b) Change the adder module so that it uses SC_THREAD instead of SC_METHOD.

   c) Modify the adder component so that it can be connected to `sc_fifo` channels.

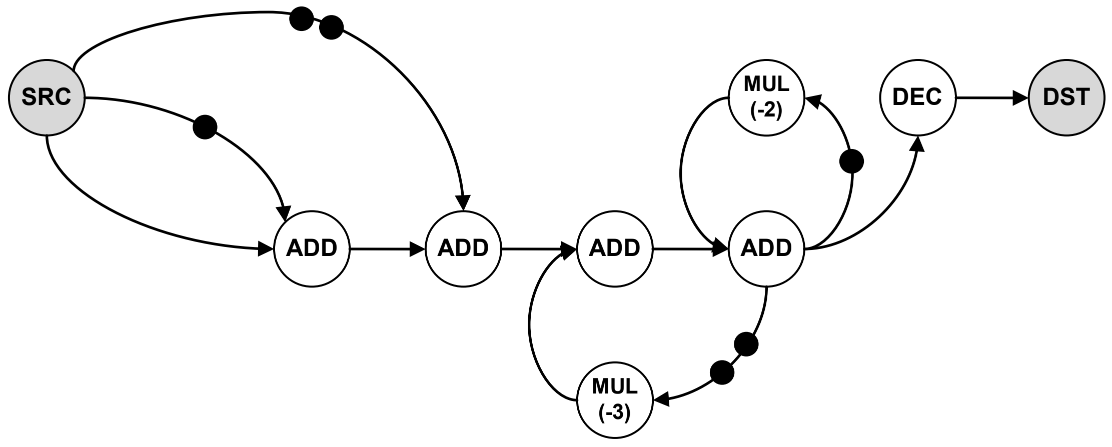
   d) Similar to the adder, develop two other components: a multiplier and a decimator.

      - The multiplier multiplies its input values by a constant *K,* which is a parameter to its constructor.
      - The decimator outputs every tenth input value. If *x* is the input and *y* the output then *y[n] = x[10n]*.

   Complete the test bench and test the new components before going to the next problem.

2. Simulating KPN with SystemC

   The following diagram shows a Kahn Process Network (KPN) model of a simple signal processing system:



SRC is the data source and *DST* the destination. Each data token corresponds to an integer value. *ADD* nodes are adders and *MUL(k)* nodes multiply the input by a constant *k*. The *DEC* node is a decimator, as described in the previous exercise.

a) In directory `ex3_2` you will find the code for *SRC* and *DST* components. *DST* simply prints the received value to the console. Using your adder, multiplier and decimator components from the previous problem, construct and simulate a model of the above system. The value of all initial tokens is zero. What is the output printed by *DST*.

b) How can you trace the number of tokens in the FIFOs in the above model over time? Propose a mechanism to do this.

3. SystemC events

In directory `ex3_3` you will find a counter implementation, based on which you are required to implement a SystemC module called *Watch*. The *Watch* module shall include the following:

- A clock input port,
- Three counters, one for seconds, one for minutes, and one for hours
- A process that triggers on positive clock edges and increments the seconds counter on each activation,
- Two more processes, one to increment the minute and the other to increment the hour,
- Communication between the three processes via the event objects `next_minute` and `next_hour`, using event notification and waiting,
- Another process, triggered by the negative clock edge that writes the hours, minutes, and seconds to the standard output.

The seconds process must inform the minutes process when the seconds counter goes from 59 to 0. Similar communication from the minutes process to the hours process is required.

Write a `sc_main()` function that instantiates a Watch and a SystemC clock, connect them, and run the SystemC simulation for a little bit more than a minute of simulation time, later for a little bit more than an hour, and finally for a little bit more than a day. The time to be simulated can be specified when calling `sc_start()` with a time parameter, e.g. `sc_start(62, SC_SEC)`. Make sure that the watch works correctly in each step.