University of Stuttgart
Germany

Institute of Computer Architecture
and Computer Engineering

Dept. of Embedded Systems

Modeling, Simulation, and Specification

Prof. Dr. Martin Radetzki

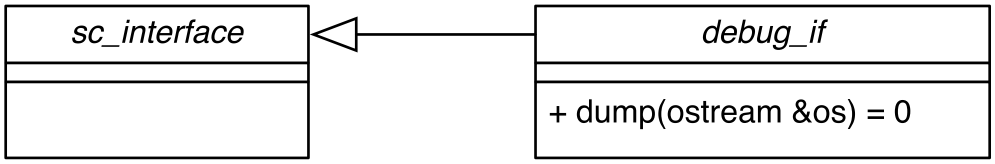
Exercise 4: *Architecture / Transaction Level Modelling PV, CX*

The source files for this exercise are available in the package exercise5.zip on the course webpage.

1. Declaration and implementation of interfaces

   In the directory **`bus_pv`**, you will find a programmers view model of a simple bus system, very similar to the system that has been presented in the lectures. You may want to compile and simulate the given model to see that it works properly. Then do the following modifications:

   a) Create a file **debug_if.h** and derive an abstract class `debug_if` as follows:



   b) Change the class **Ram** (files **ram.h**, **ram.cpp**) so that it implements the interface `debug_if`. This means to make Ram derived from `debug_if` and to implement all pure virtual methods of the interface. The dump method shall output the contents of the memory to the ostream that is passed as a parameter.

   c) Change the `sc_main` function so that, at the end of simulation, a `dump()` message is passed to each Ram object.

2. From the Programmers View to the Architecture View

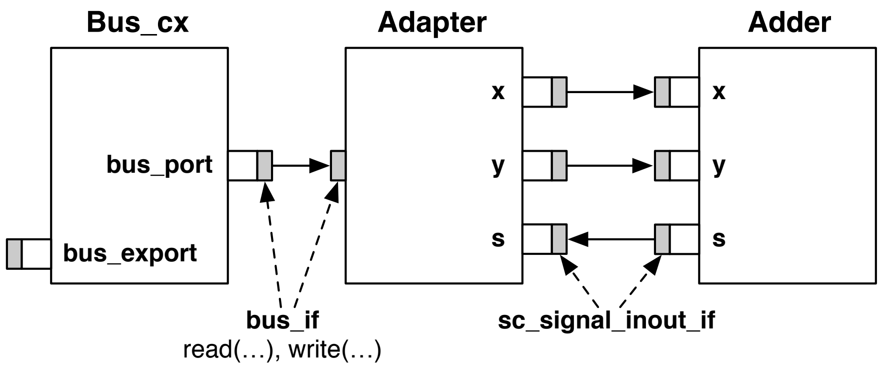
   In the directory **bus_cx**, you will find the same model that you have started with in problem 1, i.e., a programmers view model. In addition, the directory contains a cycle-approximate model of the bus (files **bus_cx.h**, **bus_cx.cpp**). Your task is to replace the programmers view bus **`Bus_pv`**, used in **`sc_main`**, with the cycle-approximate model, **`Bus_cx`**. Then start the simulation again and see the changes.

3. Use a memory-mapped slave and adapters

   Still in the directory bus_cx, there are two additional files, adder.h and adder.cpp, which contain an adder module very similar to the one you have developed in the previous exercise. Your job is to attach the adder to the system bus and test it in the following steps:

   a) The adder has an interface that consists of **`sc_in`** / **`sc_out`** ports. However, the interface of the system bus is a transaction interface. Write a transactor (adapter) for the two different interfaces. The transactor must implement read and write transaction and translate them into read and write operations on the adder ports. The transactor shall allow access to the adder as a memory-mapped bus slave. The adder shall be mapped to the addresses 1024 – 1026, where read and write transactions to these addresses shall have the following meaning:

   - `write(1024, data)` assigns the data to the first input of the adder,
   - `read(1024, data)` returns the value of the first input of the adder,
   - `write(1025, data)` assigns the data to the second input of the adder,
   - `read(1025, data)` returns the value of the second input of the adder,
   - `write(1026, data)` is forbidden and causes an error message because address 1026 is read-only,
   - `read(1026, data)` returns the value of the output of the adder.



   b) Add a new master to the bus accessing the adder. Copy the existing master (master.h, master.cpp) to new files and then modify it so that the new master:

   - waits for some time so that the existing masters can write their data into the RAMs first,
   - reads values from address 0 and 16 and writes them to addresses 1024 and 1025, then reads from 1026 and writes the result to address 64,
   - repeats the same for addresses 1 and 17 (result to address 65), 2 and 18, …
   - **Note**: you will have to create a third **`Ram`** object in **`sc_main`** and connect it to the bus so that it is mapped to addresses starting from 64 (to store the addition results). You could use the RAMs from problem 1 which can be dumped – this makes it easier to see if everything works correctly. Don't forget to add the new slaves to the memory map file **mem_map.txt**.