

UML and MDA for Transactional Level Modeling

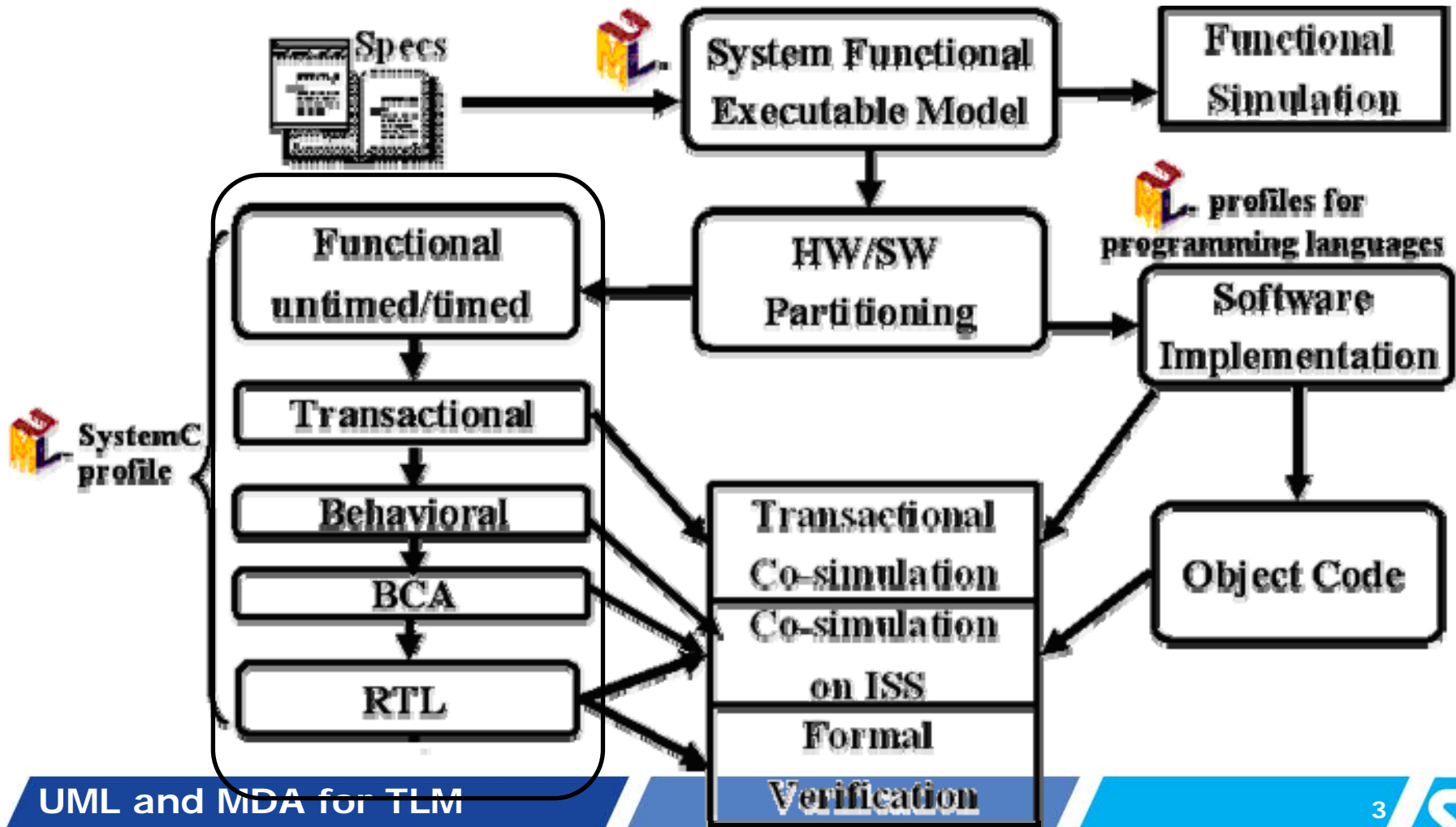


S. Bocchio, A. Rosti
STMicroelectronics
E. Riccobene,
P. Scandurra
University of Milan

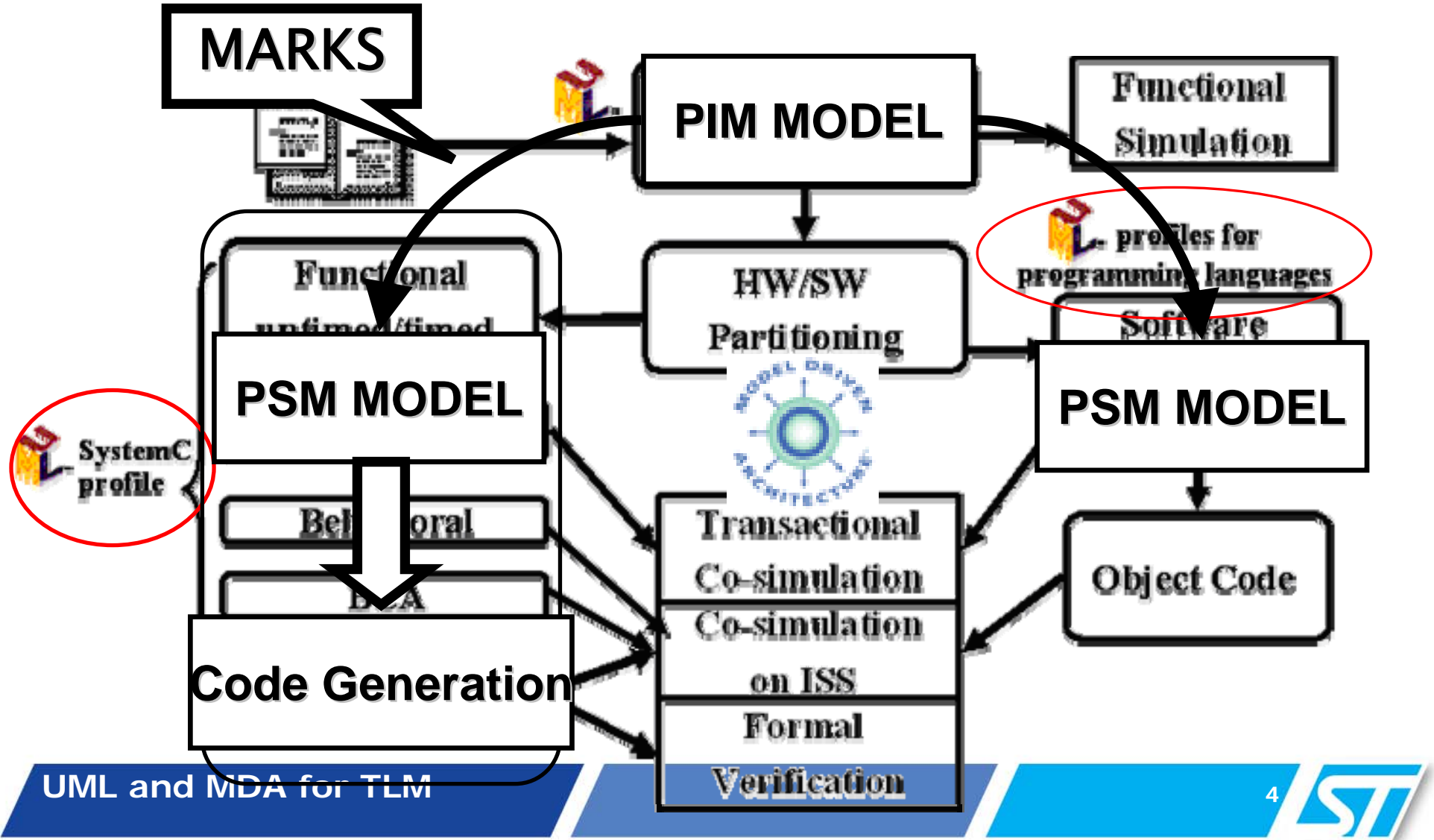
Outline

- ▣ Introduction: motivation and objectives
- ▣ Background: the SystemC UML profile and the tool for the UML/SystemC profile
- ▣ Update to SystemC2.1 and TLM
- ▣ Examples:
 - ▣ Simple bus
 - ▣ TLMinfra library and platform example

MDA-SoC design flow



MDA-SoC design flow



Programming vs Modelling

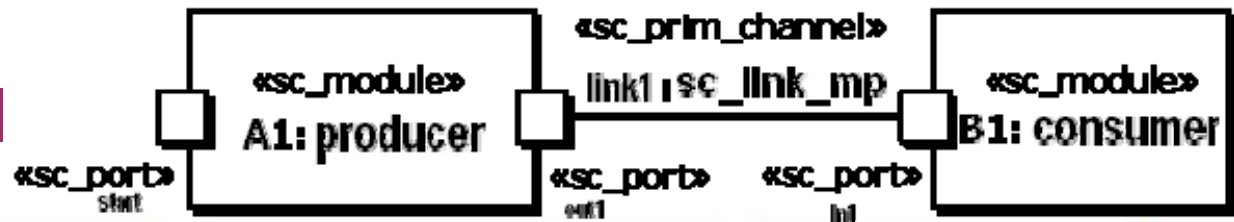
```
SC_MODULE(producer){
  sc_outmaster<int> out1;
  sc_in<bool> start;
  void generate_data(){
    for(int i=0;i<10;i++)
      out1 = i; // to invoke slave
  }
  SC_CTOR(producer){
  SC_METHOD(generate_data);
  sensitive<< start;
  }
};
```

```
SC_MODULE(consumer){
  sc_inslave<int> in1;
  int sum; //state variable
  void accumulate(){
    sum +=in1;
    cout<<"Sum ="<< sum <<endl;
  }
  SC_CTOR(consumer){
  sum =0; //initialize
  SC_SLAVE(accumulate, in1);
  }
};
```

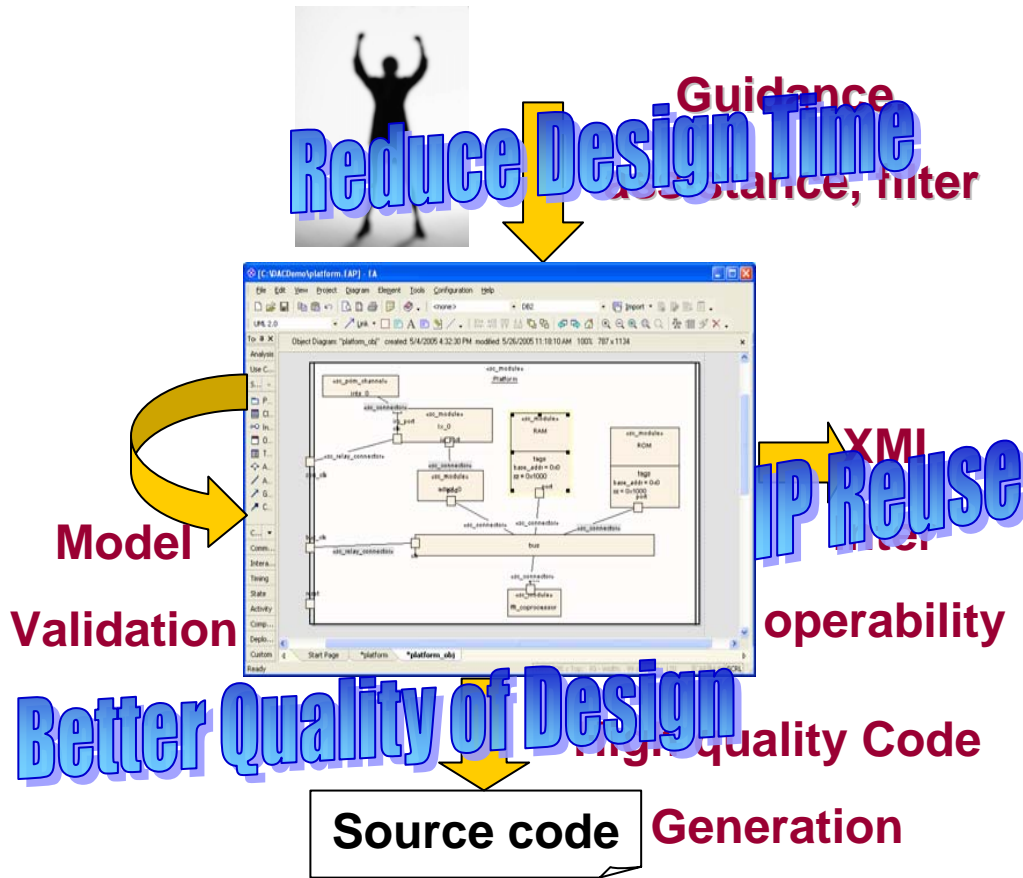
```
SC_MODULE(top){
  sc_link_mp<int> link1;
  producer* A1;
  consumer* B1;
  SC_CTOR(top){
  A1 = new producer("A1");
  A1.out1(link1);
  B1 = new consumer("B1");
  B1.in1(link1);
  }
};
```

A bit of modern SW

... and its model



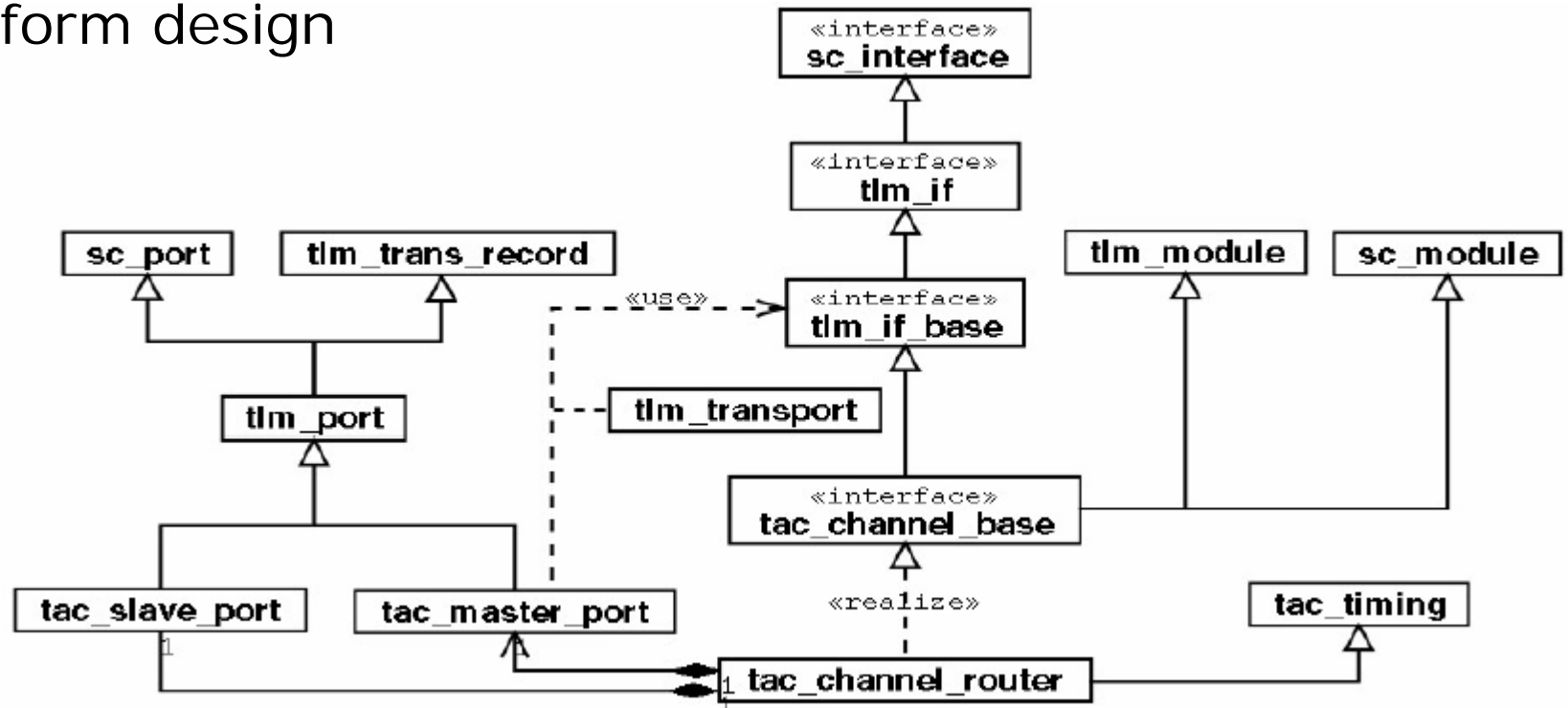
MDA for SystemC-TLM



- ▣ Definition of a proper model that allows
 - ▣ Better quality of design
 - ▣ Faster design
 - ▣ Higher integration levels
 - ▣ Rapid development of derivative designs
- ▣ Definition of metamodels communication semantic
- ▣ Tool support to make effective the metamodel based methodology

Does UML fit hw design?

- SoC design = component based design
 - Our view of an SoC design is defined by extensive use of reusable IP blocks, and mixed HW/SW design issues
- TLM methodology drives massive OO concepts usage in platform design



UML PROFILE

- ▣ A profile is a group of UML stereotypes, constraints, and tagged values that
 - ▣ add domain-specific information to the UML
 - ▣ possibly altering the notation (through special icons)
- ▣ A **stereotype** defines **how a UML construct** - a *class* in the UML *metamodel* - **is extended** for a specific target domain
 - ▣ with ***tags*** to state additional properties
 - ▣ and ***constraints*** in the **Object Constraint Language (OCL)** to add some restrictions
- ▣ It can be intended as a way of creating a new dialect of the UML for a particular platform or domain

Profile structure - updated

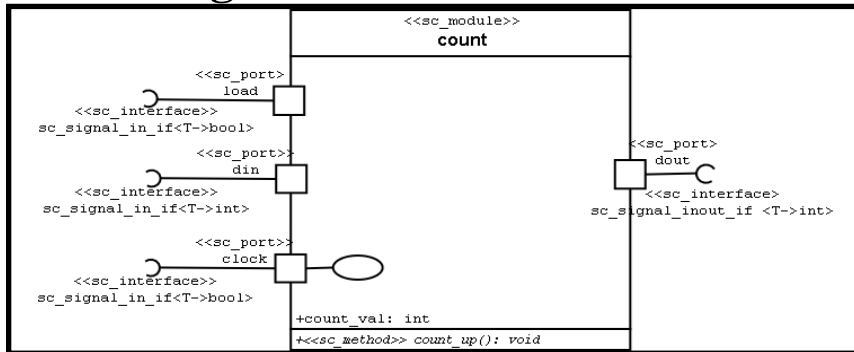
SystemC 2.1 profile structure

1. The SystemC core layer - structure and communication (modules, interfaces, ports and channels)
2. The SystemC core layer - behavior and synchronization (method state machines)
3. The SystemC core layer - data types - defines a UML class library to represent the set of SystemC data types.
4. The SystemC layer of predefined channels, interfaces and ports
5. The OSCI TLM 1.0 library of predefined channels and interfaces.

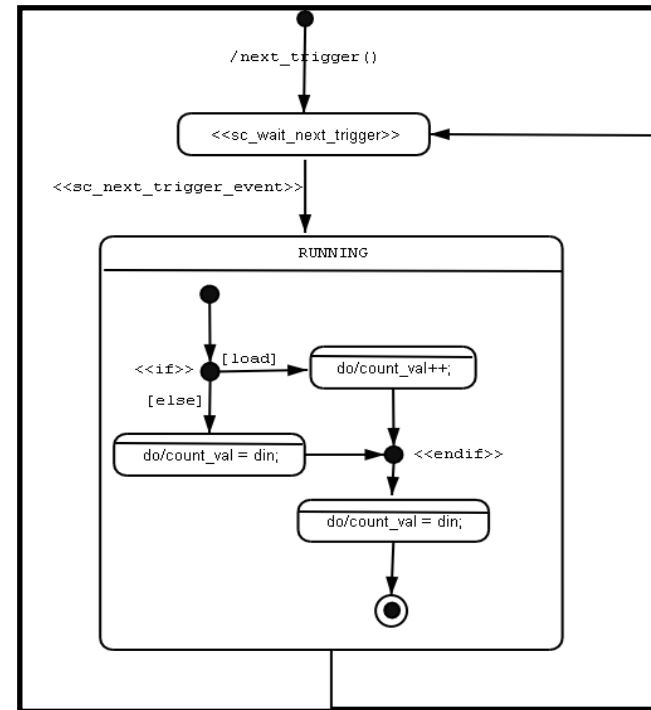
UML Profile for SystemC

provides a **graphical entry** to SystemC
stereotyped class, structured class and state machine diagrams

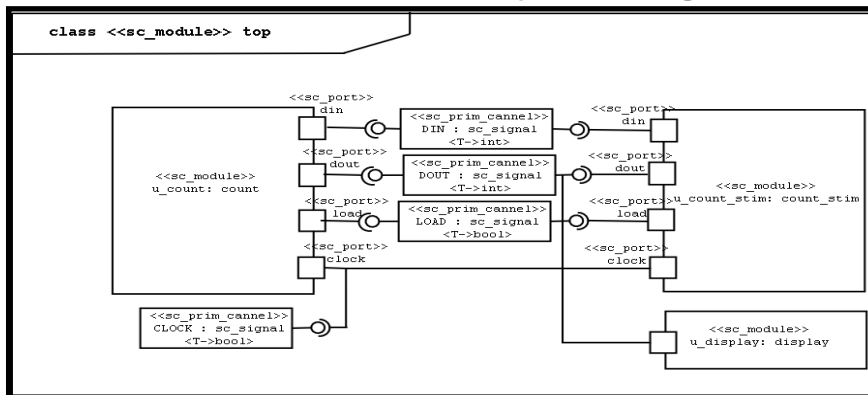
class diagram



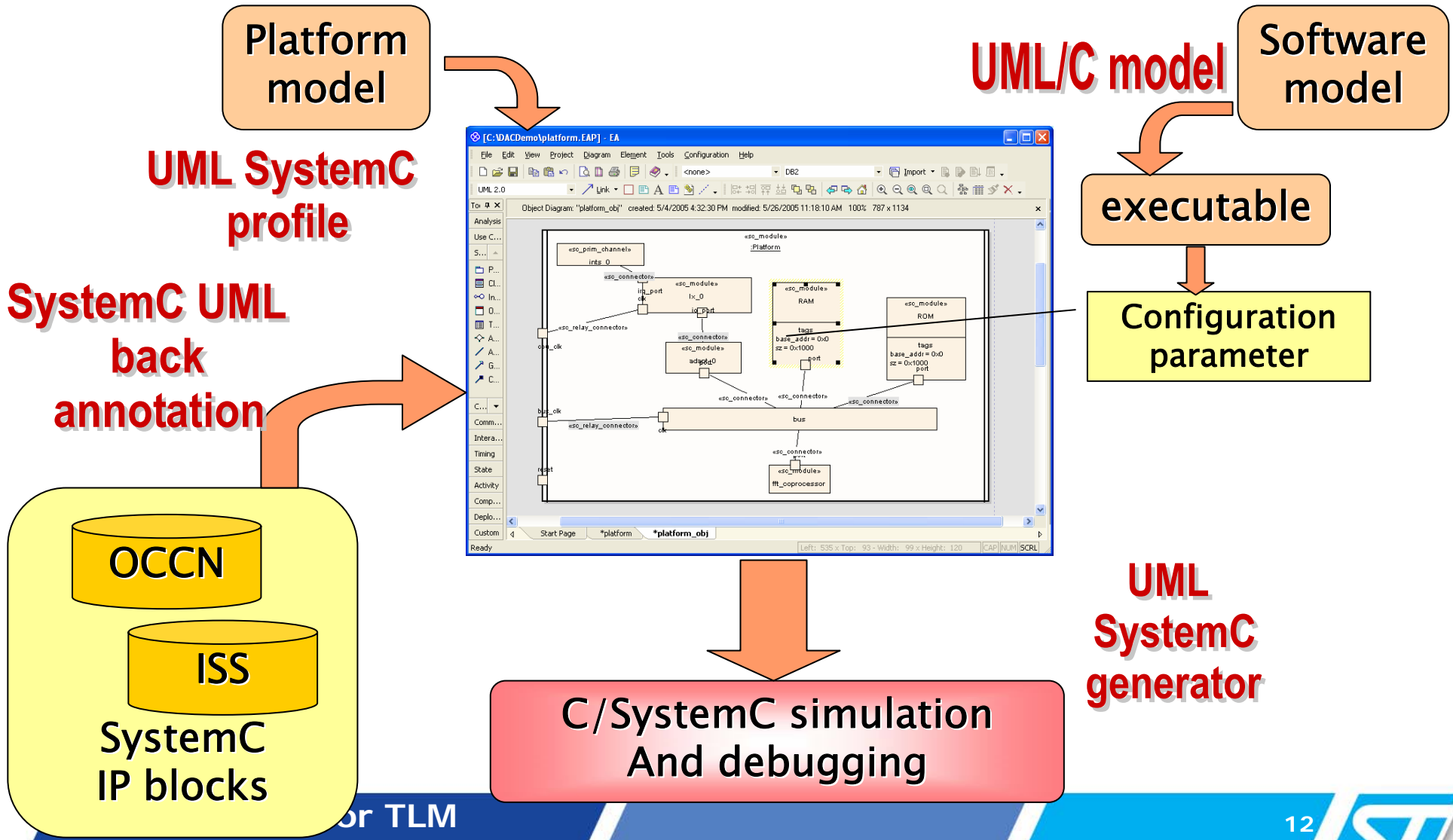
behavioral state machine



structured class and object diagram



EA Based framework for SystemC



SystemC code generator

The screenshot displays the UML 2.0 software interface. The main window shows a Statechart Diagram titled "exec". The diagram starts with an "Initial" state, leading to a "Running" state. The "Running" state contains the action: `+ Do Action / read_act
o = a.read() ^ b.read();`. Below the "Running" state is a state labeled "«wait_next_trigger»
Waiting for next trigger". A transition arrow points from the "Waiting for next trigger" state back to the "Running" state.

The "Tools" menu is open, showing options such as "Spell Check Model...", "Data Management", "Export Reference Data...", "SystemC" (highlighted), and "Options...".

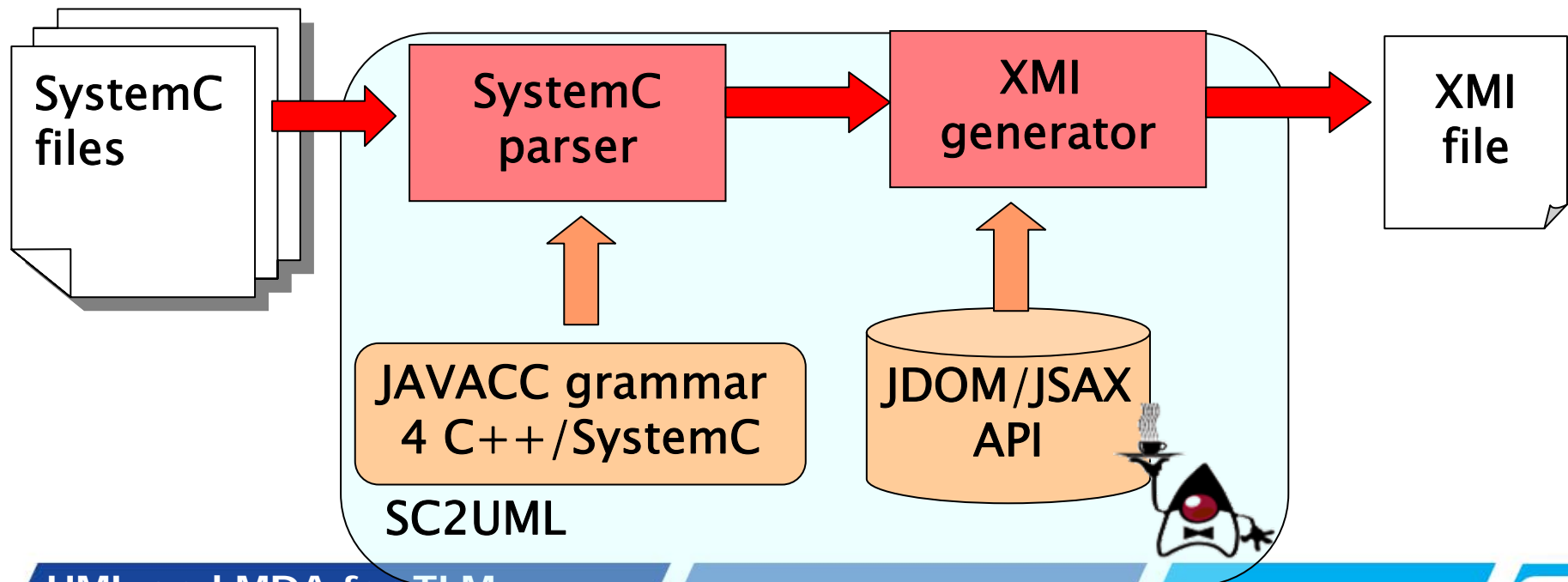
The "Project View" on the right shows a hierarchical structure of SystemC components:

- SystemC_Layer1
 - SystemCDataTypes
 - SystemCPredefinedChannelsandPorts
 - SystemCModel
 - adder
 - adder4
 - and2
 - fulladder
 - halfadder
 - or2
 - xor2
 - adder4
 - and2
 - fulladder
 - halfadder
 - or2
 - xor2
 - adder4
 - and2
 - fulladder
 - halfadder
 - or2
 - xor2
 - exec

The "Tagged Values" panel at the bottom right shows "SystemCModel (Package)".

SystemC back annotation

- import existing SystemC models into UML
- SystemC back annotation = SystemC parser + XMI generator
- EA selecting Project | Import/Export | Import package from XMI file..

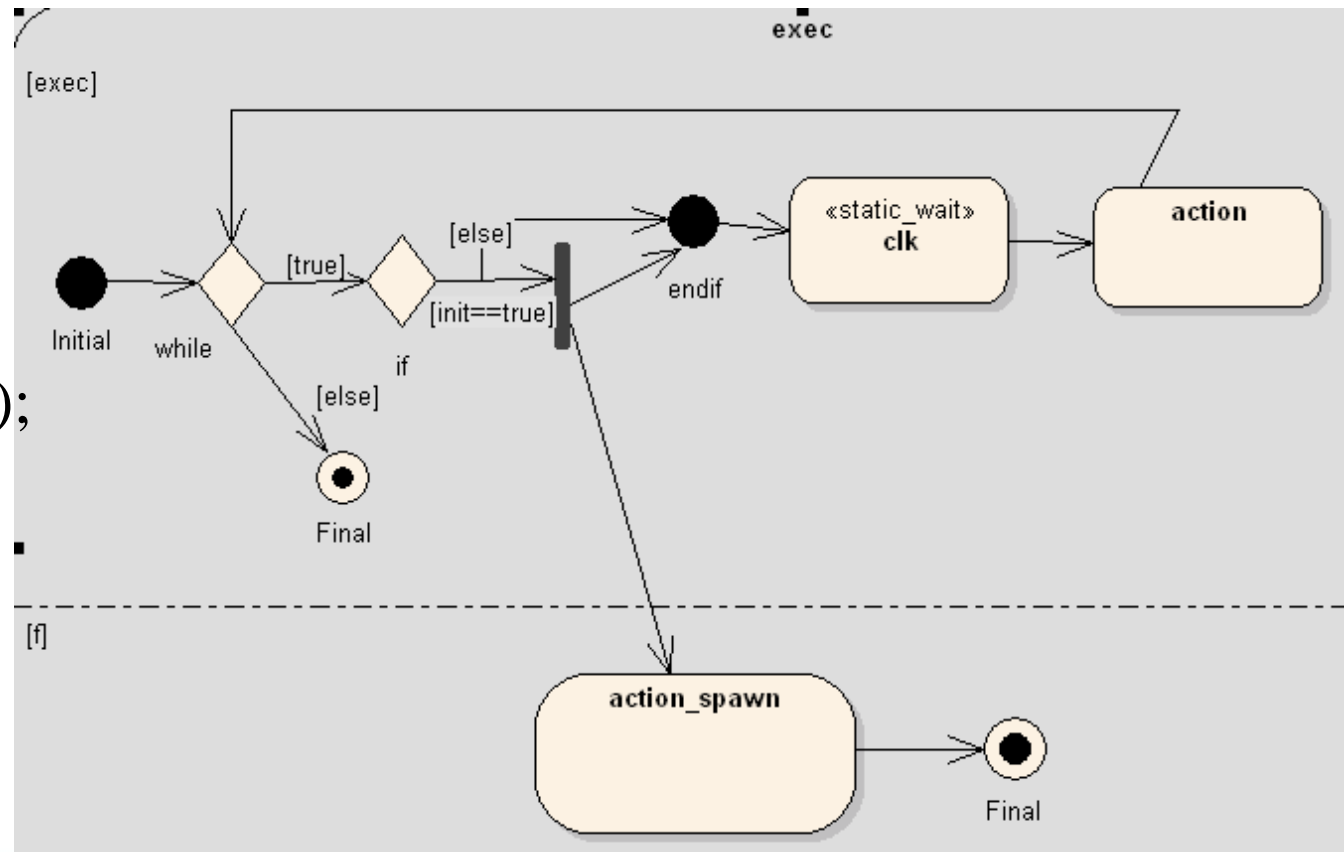


SystemC 2.1: new features

Sc_export

Dynamic thread

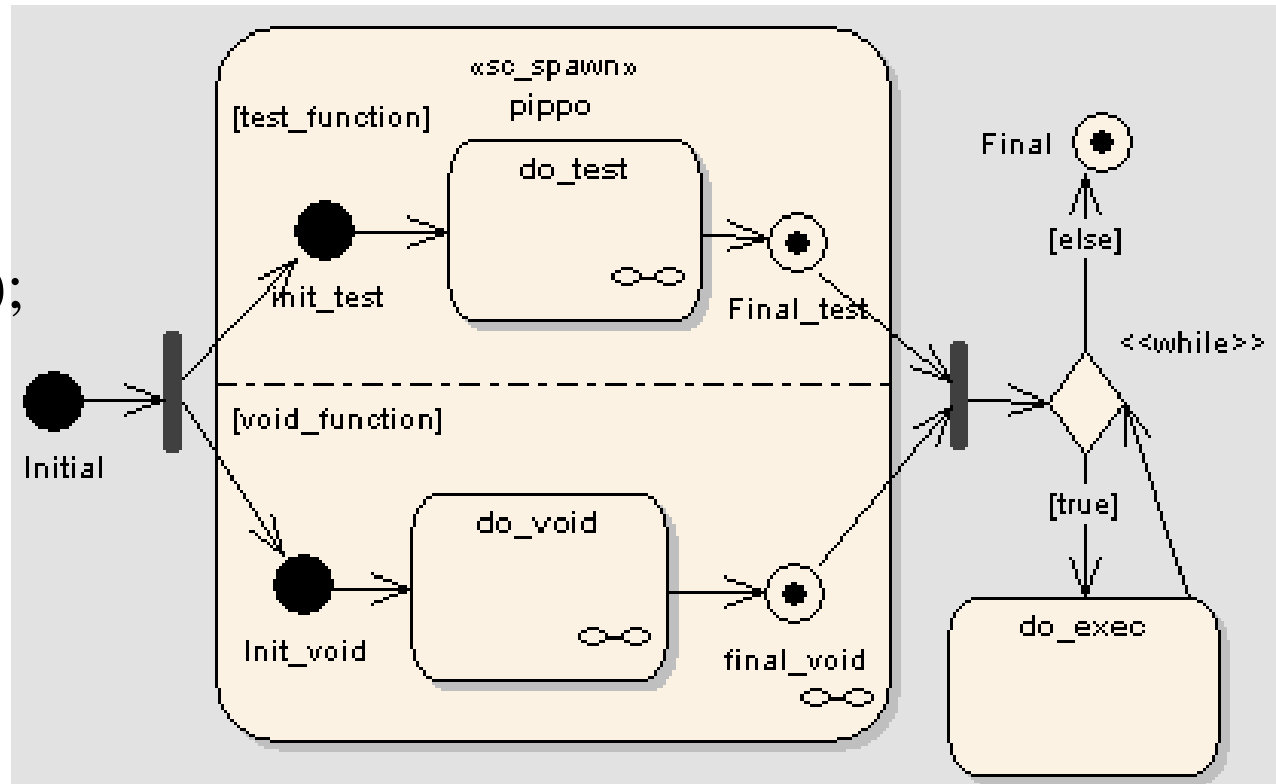
```
void exec(){  
  while(true){  
    if(init==true)  
      sc_spawn(&f,..);  
    wait();  
    action();  
  }  
}
```



systemC 2.1 new features

SC_FORK, SC_JOIN

```
void exec(){  
    SC_FORK  
    sc_spawn(&test,..);  
    sc_spawn(&void,..);  
    SC_JOIN  
    while(true){  
        do_exec();  
    }  
}
```



What is TLM?

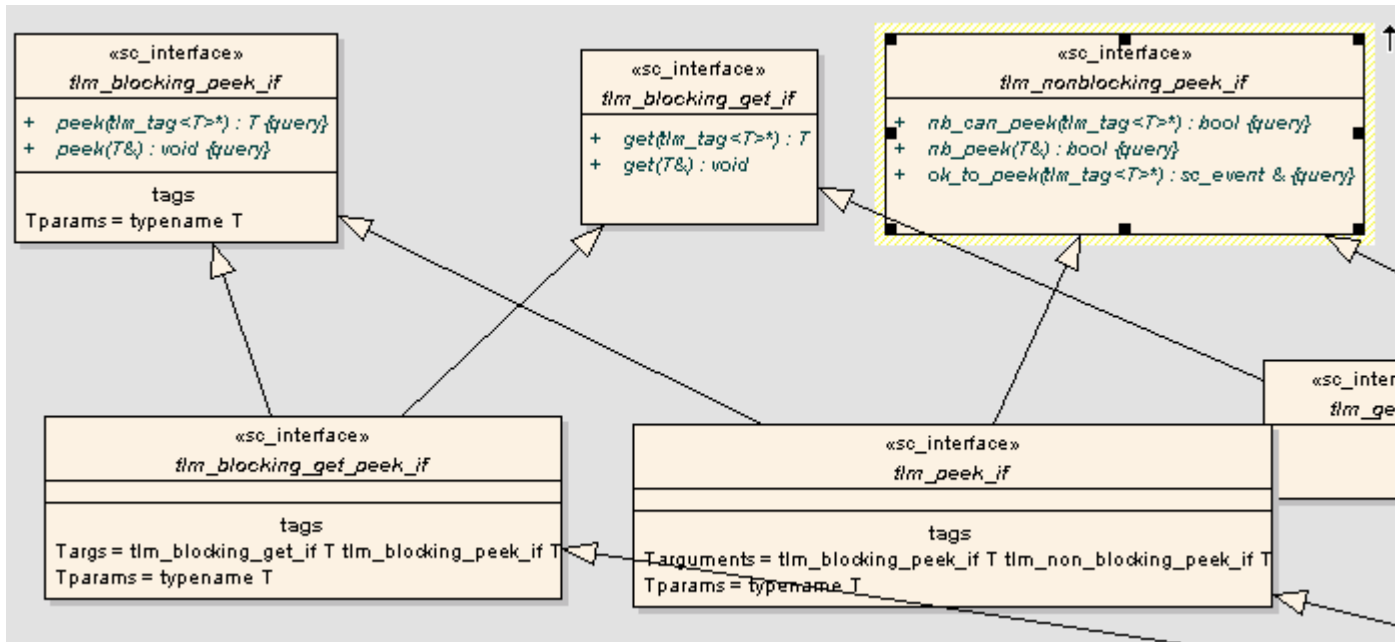
- ▣ Modeling communication through function calls
 - ▣ Based on the concept of **interfaces**
 - ▣ But can be accurate from the Timing perspective
 - ▣ Implemented by channels exposing interfaces.
 - ▣ Also gain simulation speed because communication is not pin accurate

OSCI TLM 1.0 Standard

- ❏ TML was possible since SystemC 2.0
 - ❏ Lack of standard library and methodologies can lead to incompatibilities
- ❏ OSCI TLM standard set of API for communication
 - ❏ Unidirectional blocking and non blocking
 - ❏ `put()`, `get()`, `peek()`
 - ❏ Implemented by `tlm_fifo<T>`
 - ❏ Bidirectional blocking interface
 - ❏ `transport()`
 - ❏ Implemented by `tlm_transport_channel<REQ,RSP>`

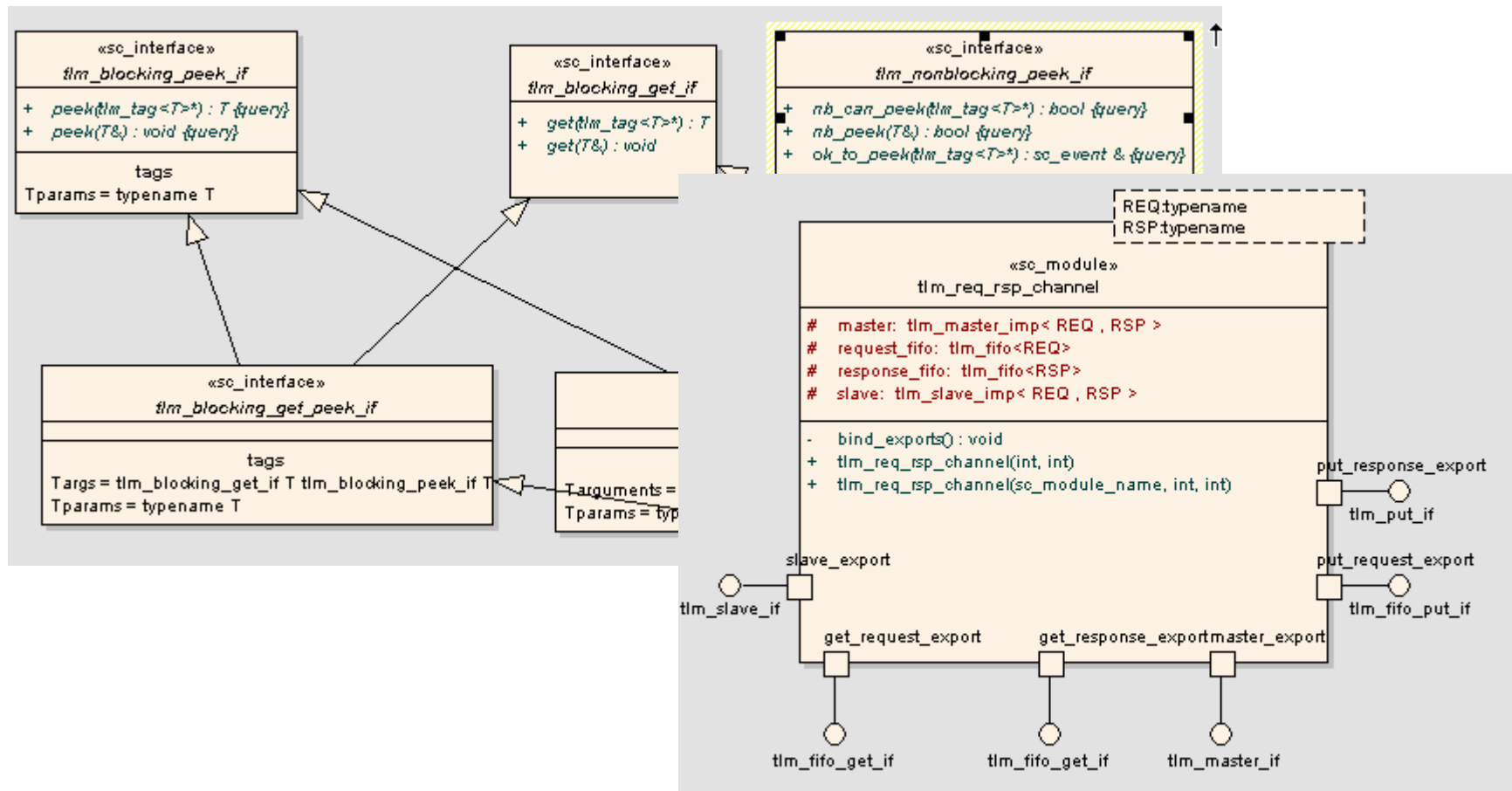
OSCI TLM library in UML

A set of model..

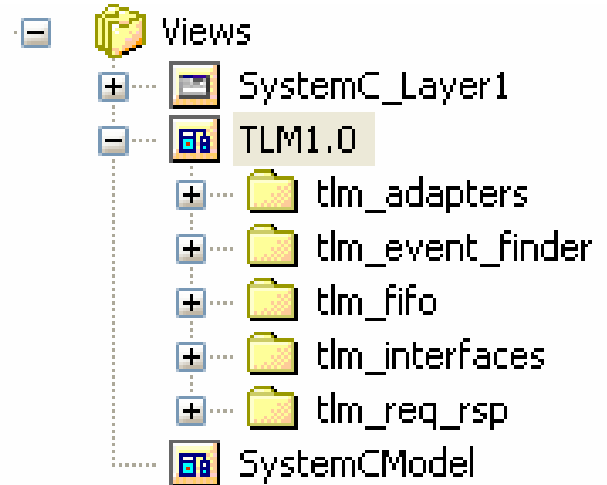
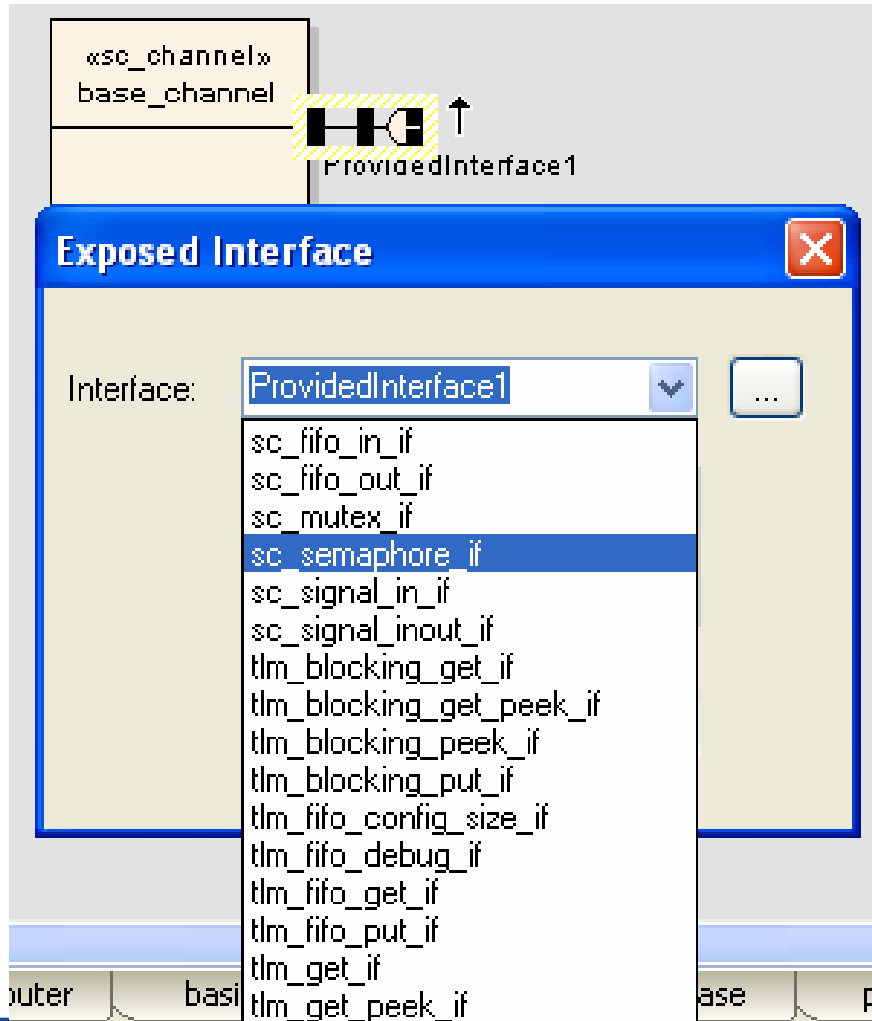


OSCI TLM library in UML

A set of model..

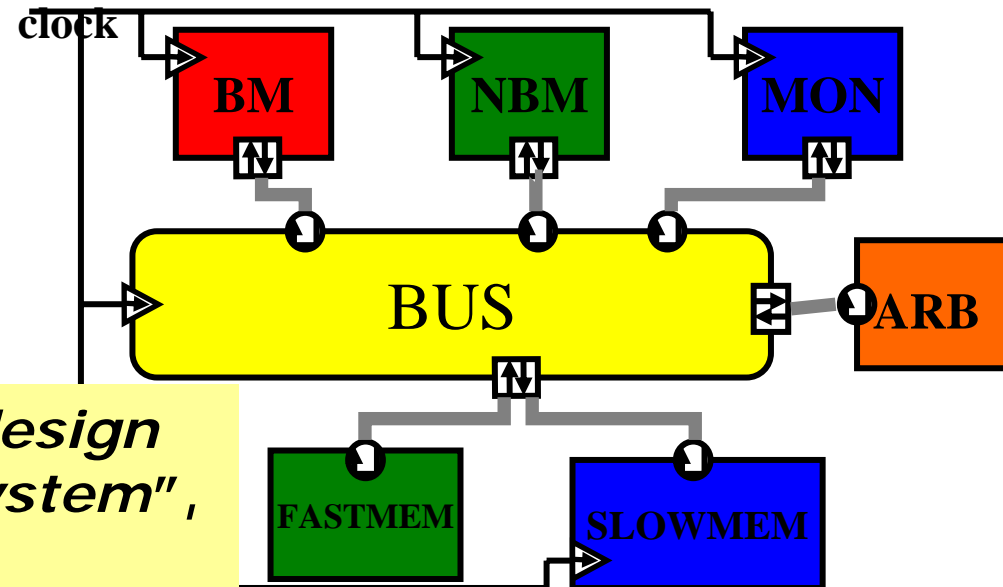


OSCI TLM 1.0 library



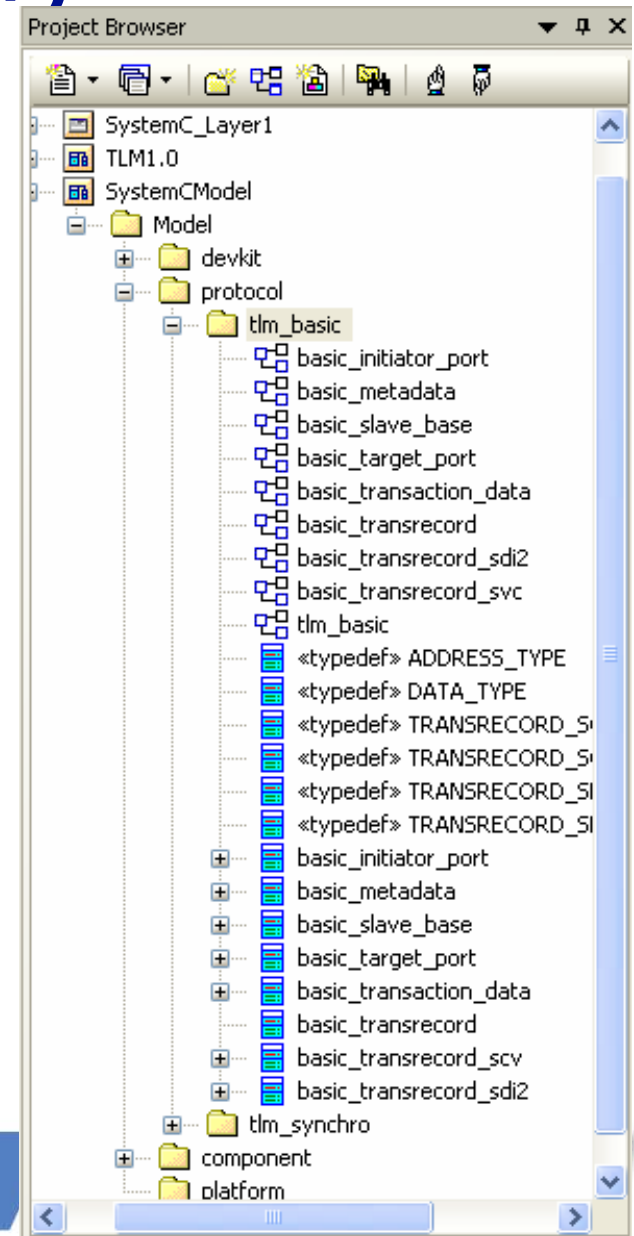
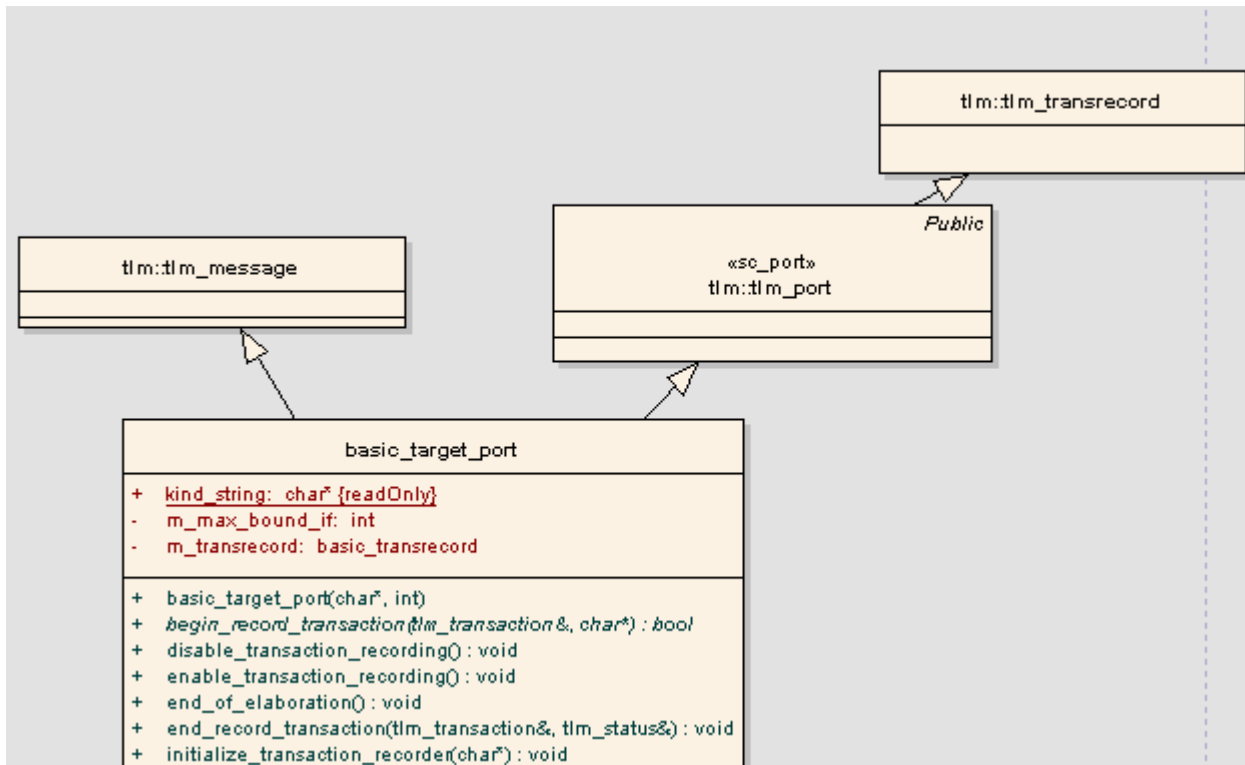
The Simple Bus (transactional level)

- ❏ M1 uses the **blocking master interface** (a high level software)
- ❏ M2 uses the **non blocking master interface** (a processor executing on every clock edge even if its bus transactions are not completed)
- ❏ M3 uses the **direct master interface** to print debug information about memories
- ❏ S1, S2 provide the same **interface**
 - ❏ S1 is a fast memory supporting single-cycle read/write operations
 - ❏ S2 is a slow memory that takes n cycles per each read/write operation

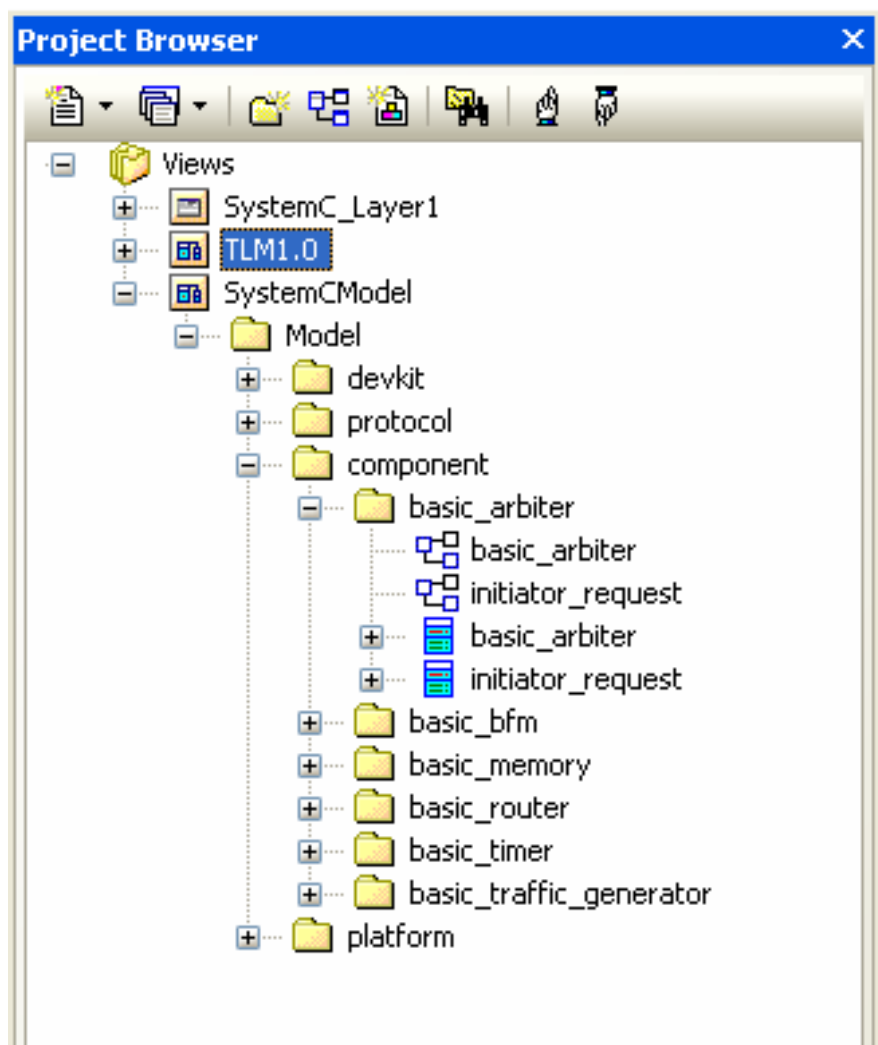


S. Bocchio... "A model driven design Environment for Embedded System", DAC '06

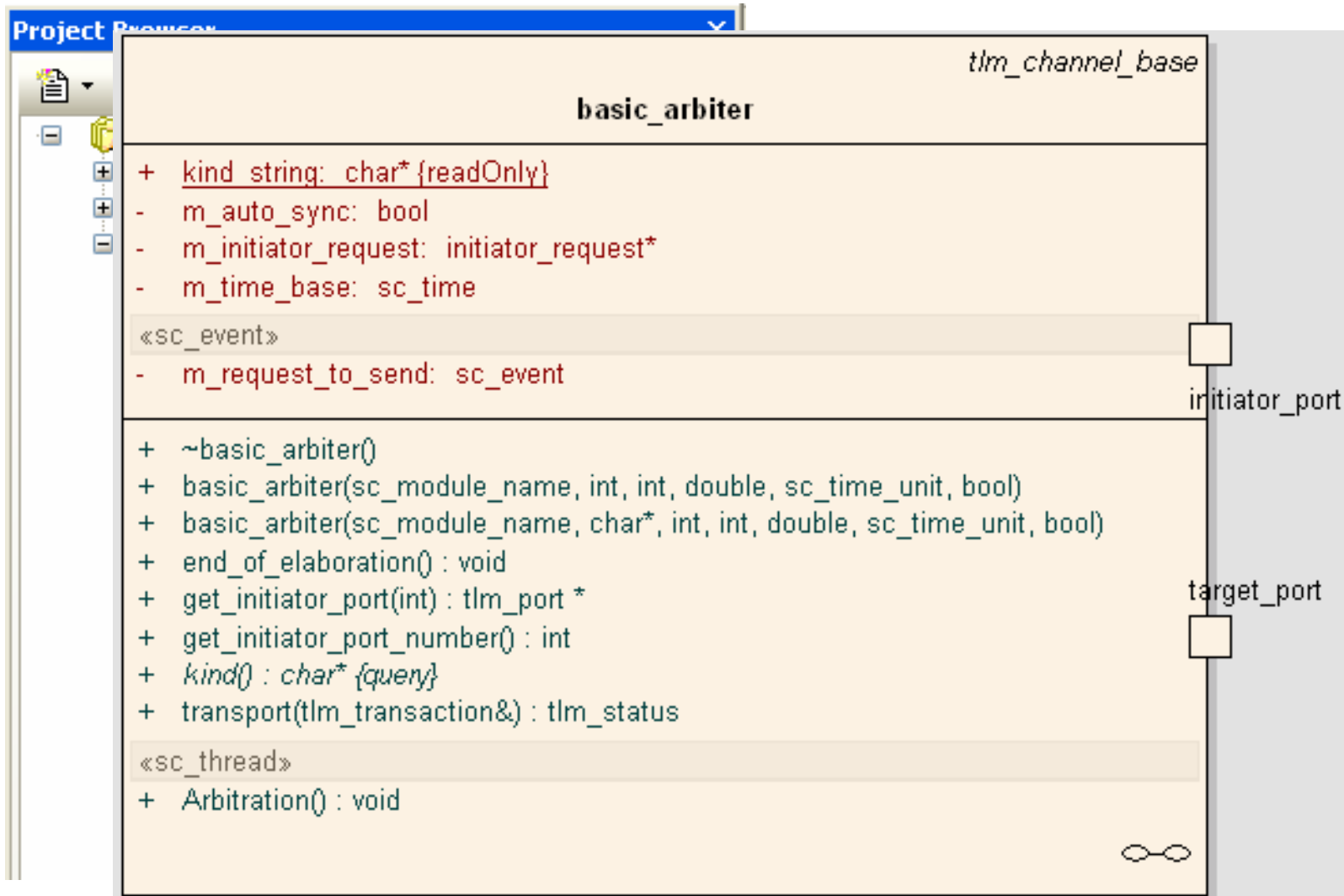
ST TLM_infra library



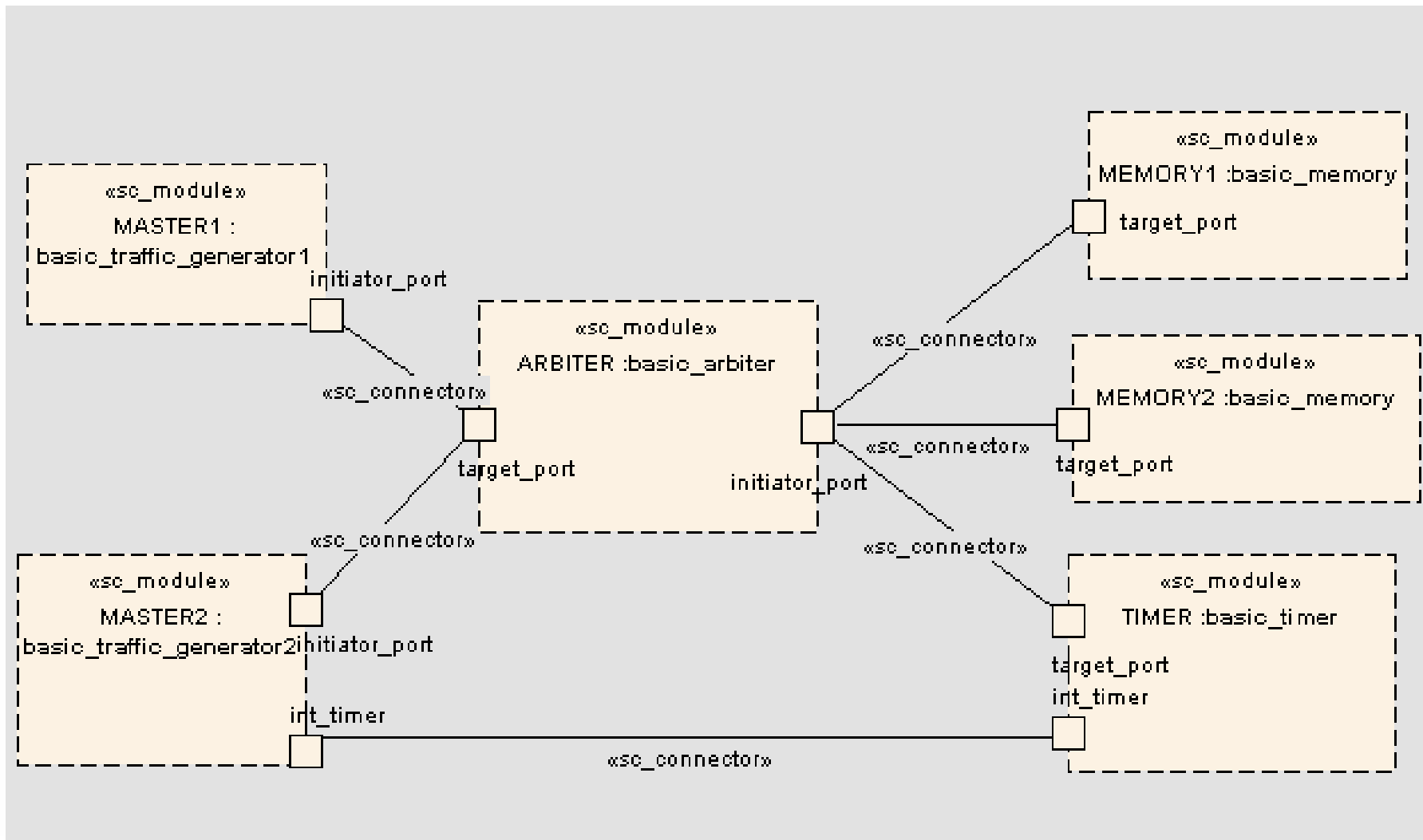
The arbiter



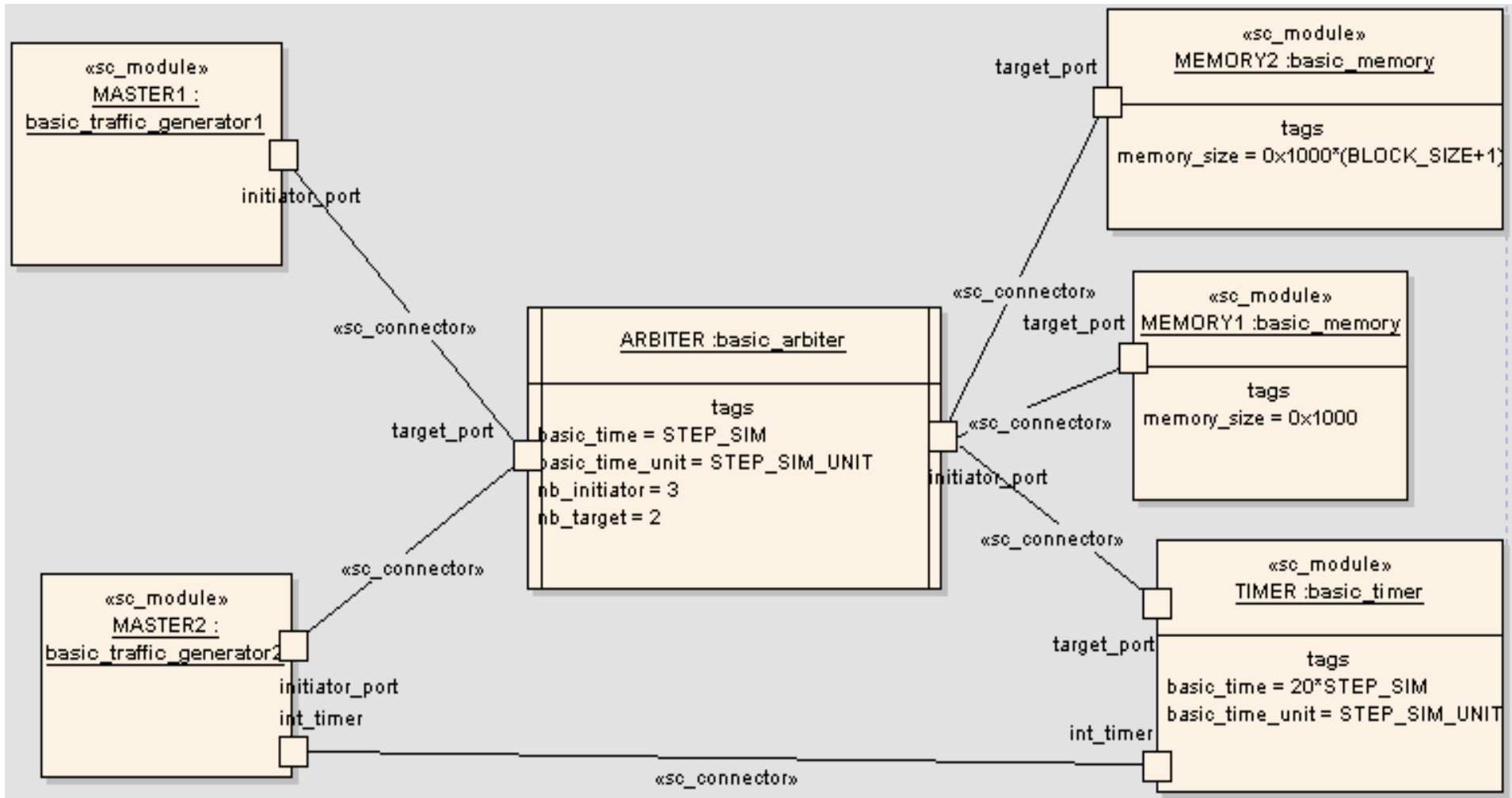
The arbiter



The platform (class diagram)



The platform instance (object diagram)



Conclusion

- ▣ UML profile make simpler platform building
- ▣ UML is NOT just a way to have a composition tool!
 - ▣ Code generation
 - ▣ System view
 - ▣ Model validation...