

Efficient Modeling and Simulation of Data Communication Protocols in Communication-oriented Designs using the SystemC^{SV} Extension



Robert Siegmund, Dietmar Müller

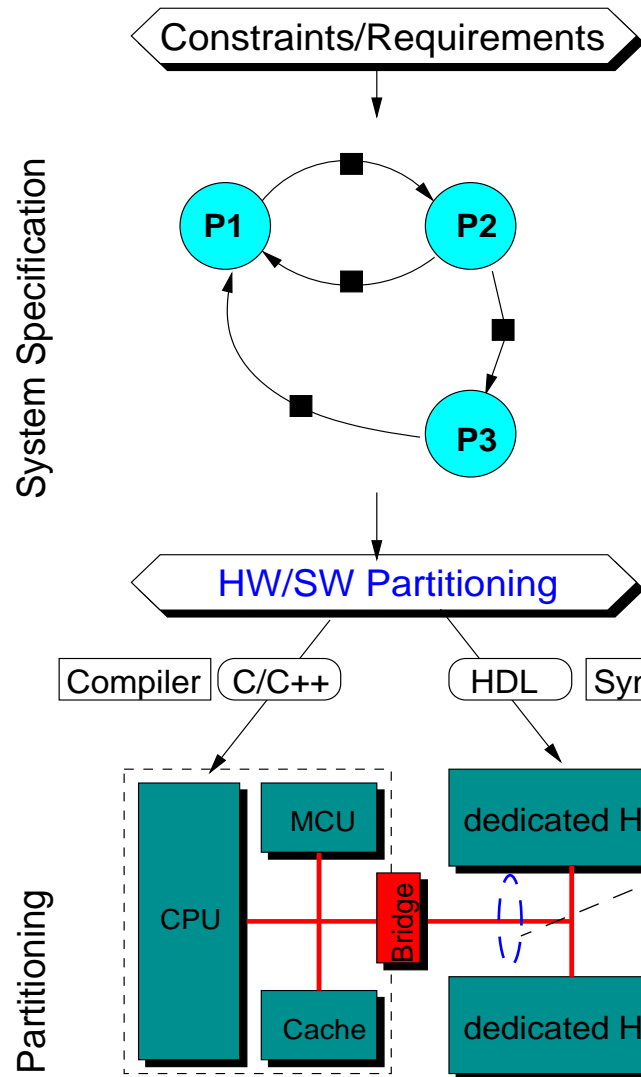
Chemnitz University of Technology
Professorship Circuit and System Design
D-09126 Chemnitz



Outline

- ❑ Motivation
- ❑ Basics of SystemC^{SV}
- ❑ General Structure of SystemC^{SV} Models
- ❑ Simulation Support
- ❑ Design Example: CAN Network
- ❑ Conclusions and Summary

Motivation: Challenges in SoC Design



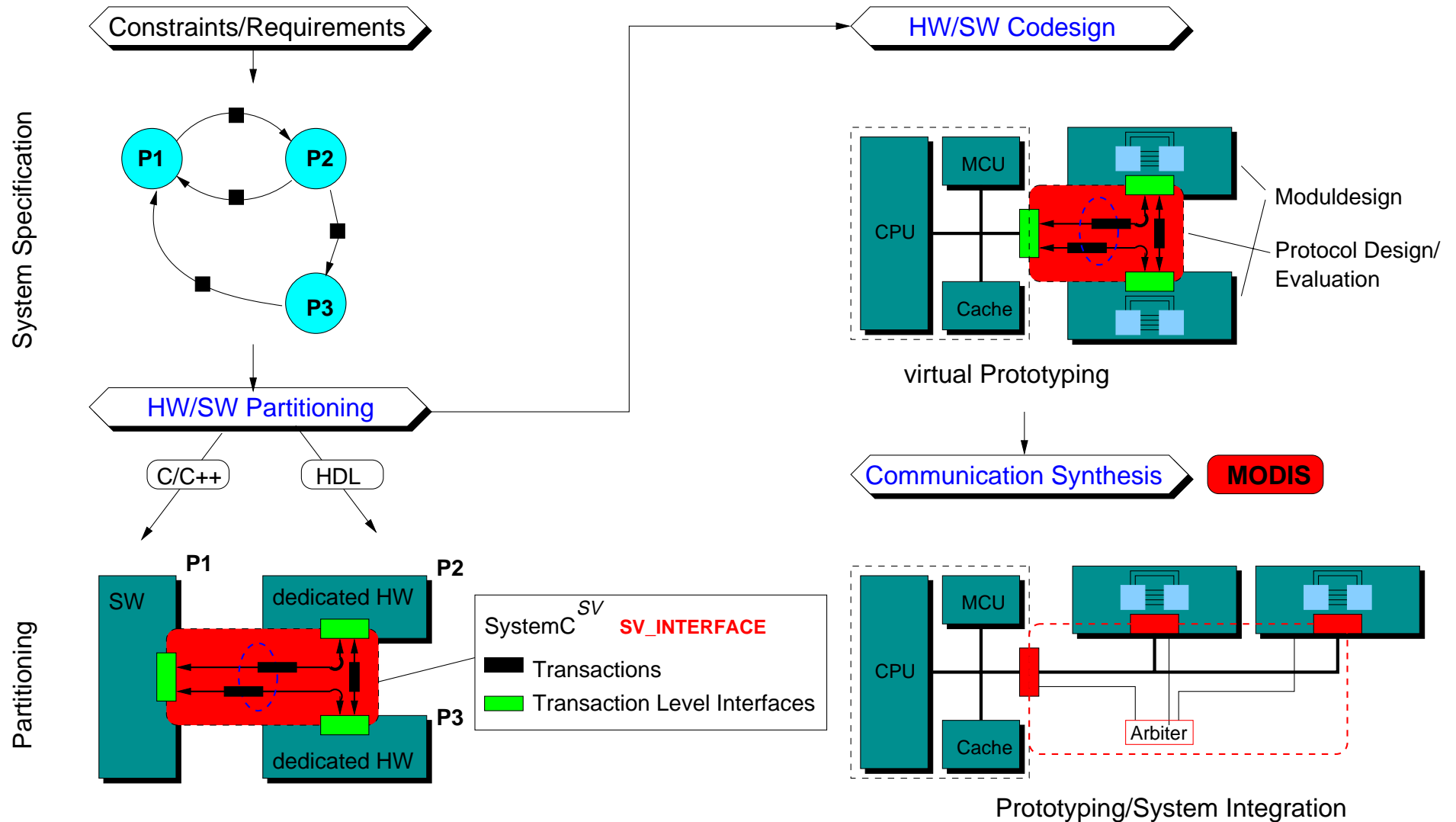
$$\text{System Performance} = f(\text{Module Performance, Performance of Communication})$$

High-Level System Model after HW/SW Partitioning:

- fixed Resources (Communication, Computation)
- static Communication Network, fixed Protocols

??? Design Space Exploration für System Communication ???

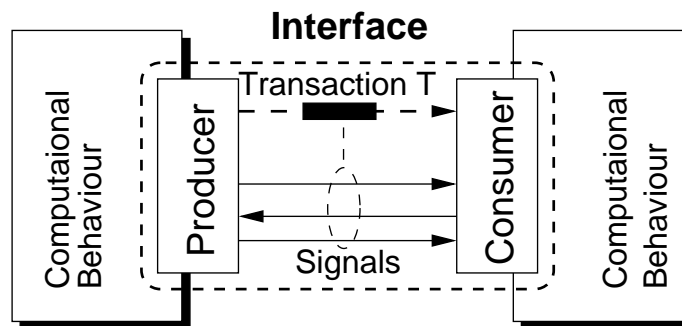
Design of Communication-oriented Systems



Basics of SystemC^{SV}

□ Purpose:

- ✗ Interface-based System Specification
- ✗ Mixed Multi Abstraction Level Communication Modeling
- ✗ Inclusion of Protocol Specifications in System Models

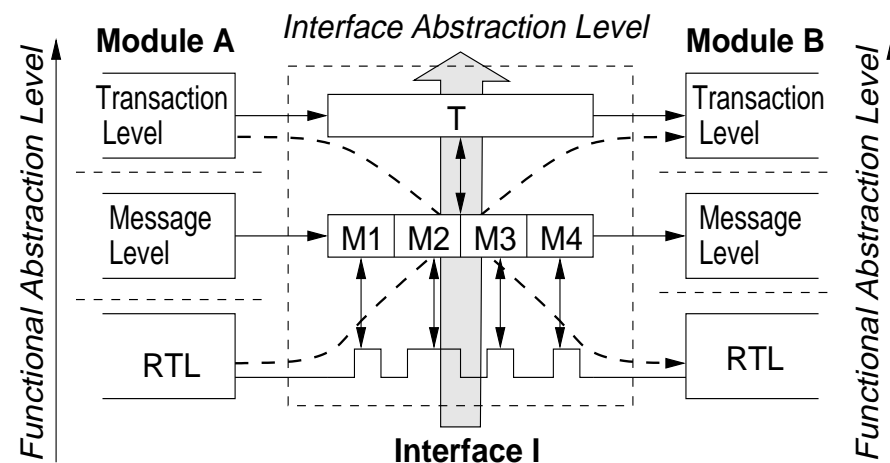


SystemC 2.0 / SpecC Interface Model:

- ✗ explicit specification of behaviours of transaction producer and consumer

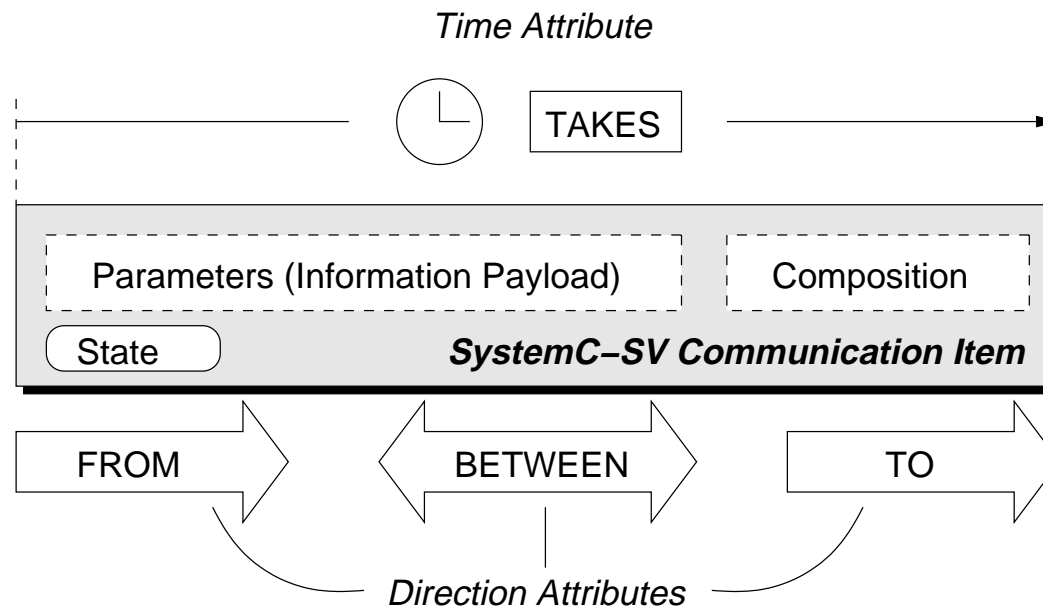
SystemC^{SV} Interface Model:

- ✗ Transactions and hierarically stacked Transaction Protocol Rules



SystemC^{SV} Communication Items

- ❑ System Communication specified with **Communication Items**
- ❑ **Communication Items**: static / dynamically created Resources used for Information Transfer between Modules



SystemC^{SV} Communication Items

```
SV_TRANSACTION(CAN_Data) {  
    ...  
};
```

Specification of a multi-directional Data Transfer

```
SV_MESSAGE(CAN_Arb) {  
    ...  
};
```

Specification of a directed Data Transfer

```
SV_PhyMap(CAN_XBit) {  
    ASSOCIATE(p1 - XD);  
};
```

Mapping of Transactions/Messages to RTL Signals

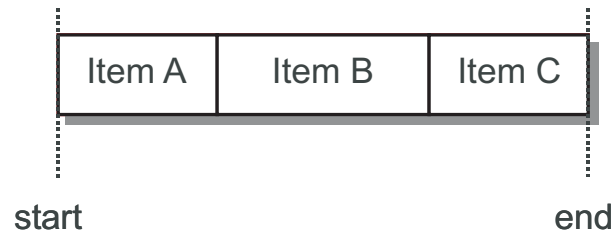
```
SV_SEQBEHAVIOUR(p_state) {  
    m_state = idle;  
};
```

Embedded Sequential Behaviour
Modification of State Variables
in Items

SystemC^{SV} Item Compositions

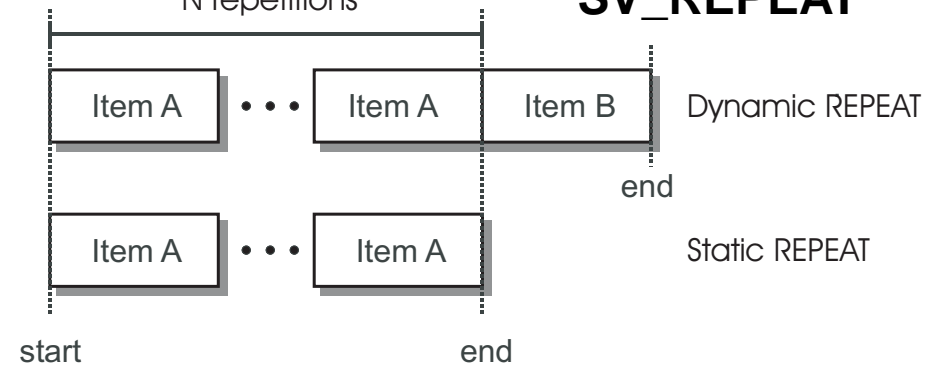
- define a Transmission Protocol for an Item, **are reversibly executable!!!**

SV_SERIAL

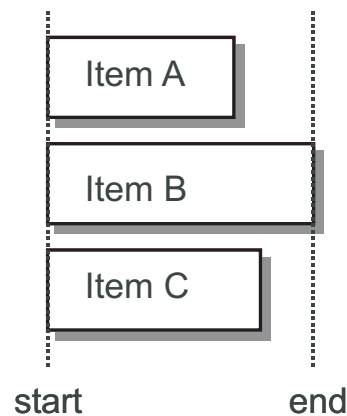


N repetitions

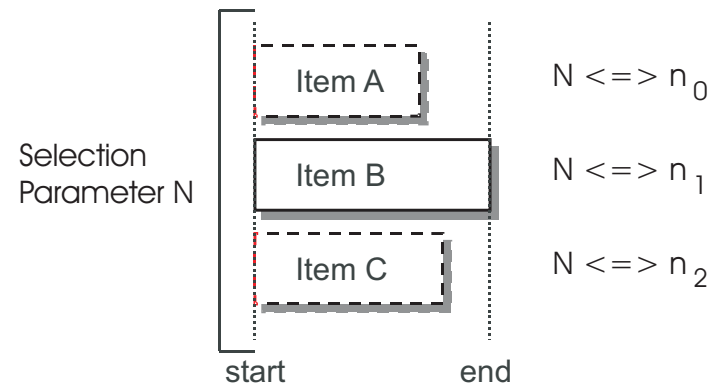
SV_REPEAT



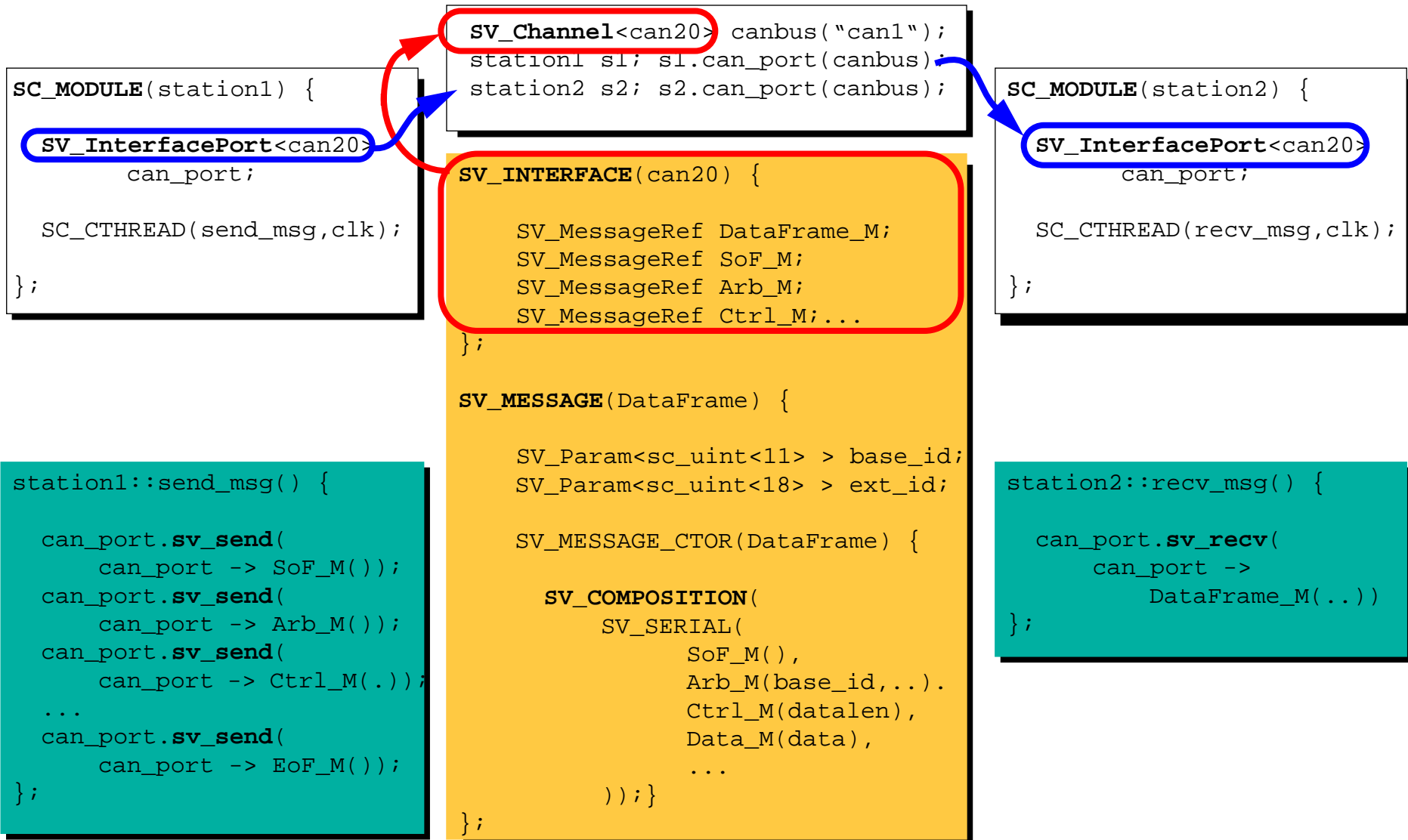
SV_PARALLEL



SV_SELECT



Basic Structure of a SystemC^{SV} Model



Example: Bit Message with Bit Stuffing

```

SV_MESSAGE(XBit) {

    SV_Param<bool> val; // message parameter
    SV_Param<sc_uint<8> > zero_count; // local state

    void incr_zero_count() { zero_count++; }
    void clear_zero_count() { zero_count = 0;}

    SV_MESSAGE_CTOR(XBit) {

        SV_COMPOSITION(
            SV_SERIAL(
                PhyBit(val), // map bit to wire

                // evaluate item parameter and modify
                // state variable accordingly
                SV_SELECT<bool>(SV_MATCH(val),
                    0 <= SV_SEQBEHAVIOUR(incr_zero_count),
                    1 <= SV_SEQBEHAVIOUR(clear_zero_count)),

                // insert '1' stuffing bit after 5 x '0'
                SV_SELECT<sc_uint<8> >(SV_MATCH(zero_count),
                    sc_uint<8>(5) <=
                        SV_SERIAL(
                            PhyBit('1'),
                            SV_SEQBEHAVIOUR(clear_zero_count)
                        ),
                    SV_OTHERS <= SV_NULL)
            );
    };
};

```

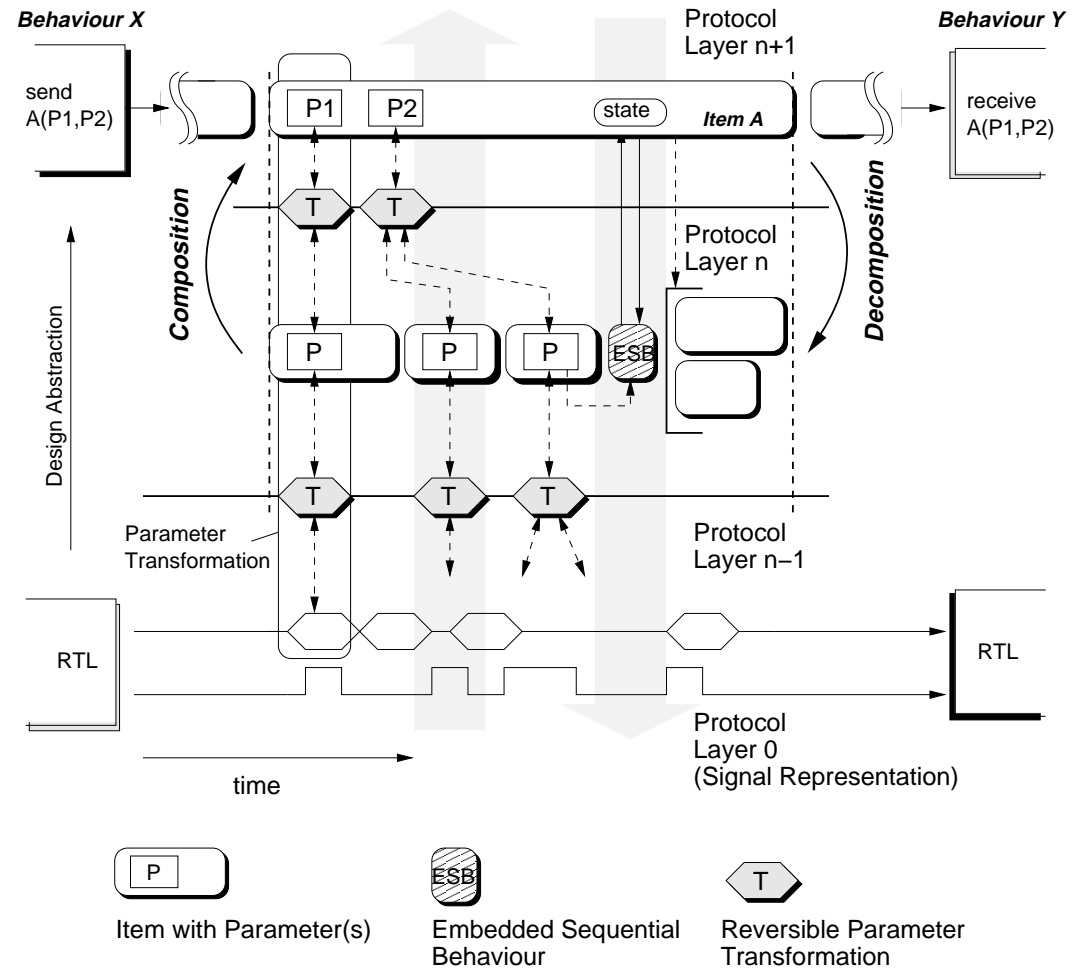
Simulation of SystemC^{SV} Models

□ Model of Computation: **Reversibly Executable Protocol Stack (REPS)**

□ Protocol Specifications of Items compiled into two distinct NDFA:

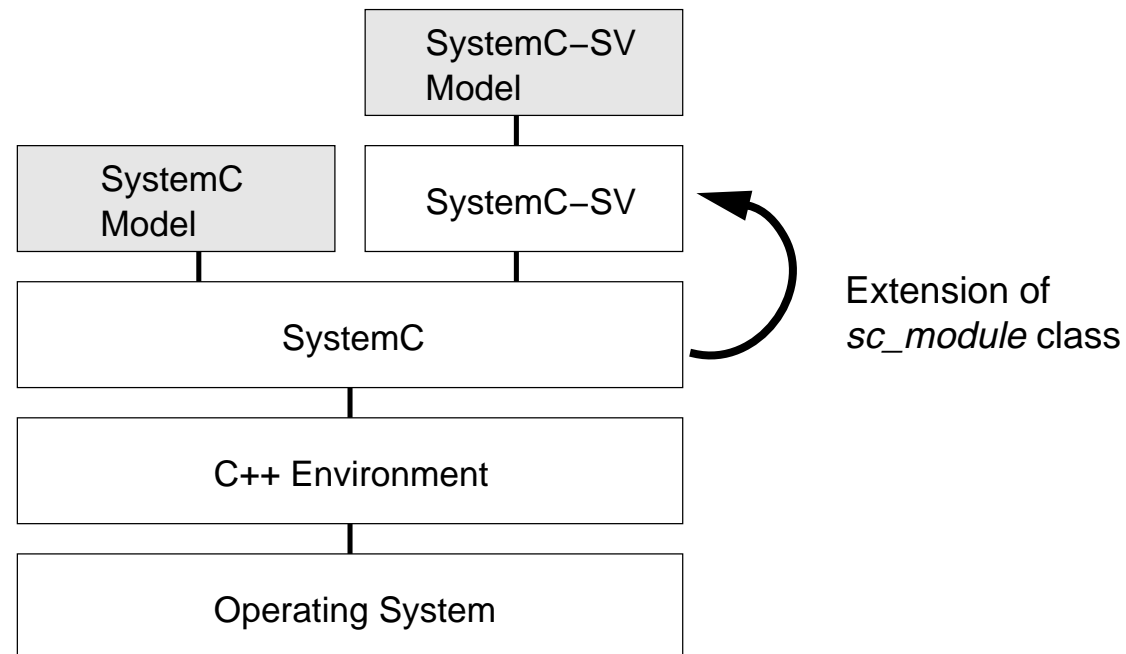
Decomposition NDFA
(Top Down Execution of the Protocol Stack)
= **Transaction Producer**

Composition NDFA
(Bottom Up Execution of the Protocol Stack)
= **Transaction Consumer**



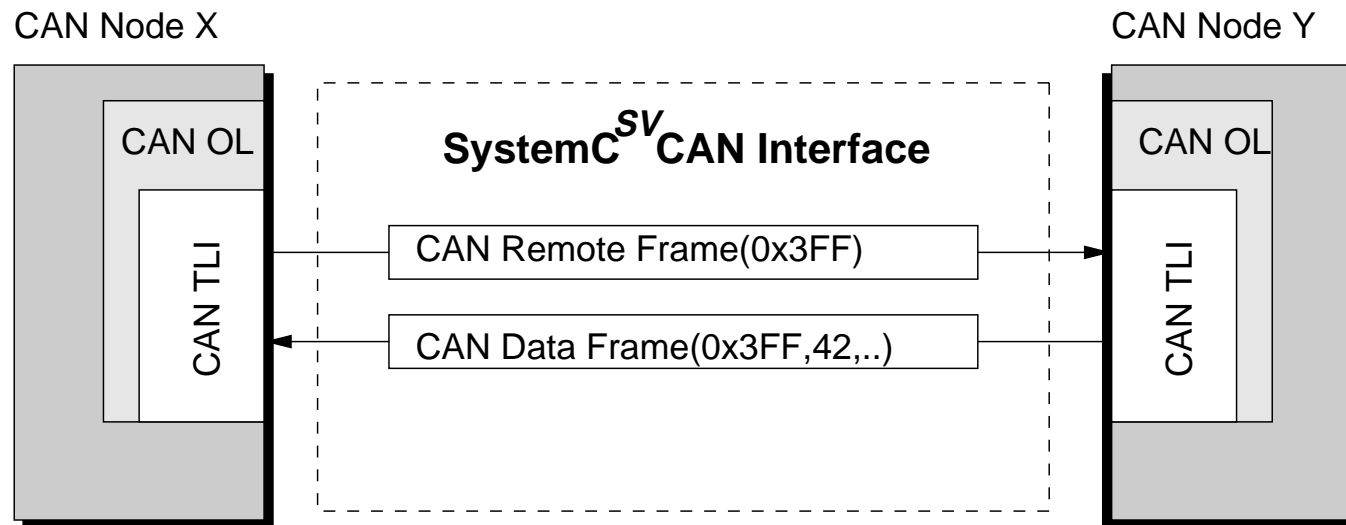
Implementation of the SystemC^{SV} Library

- ❑ All Communication Items derived from *sc_module* class
- ❑ Interoperability with future versions of SystemC ensured



SystemC^{SV} Example: CAN Bus

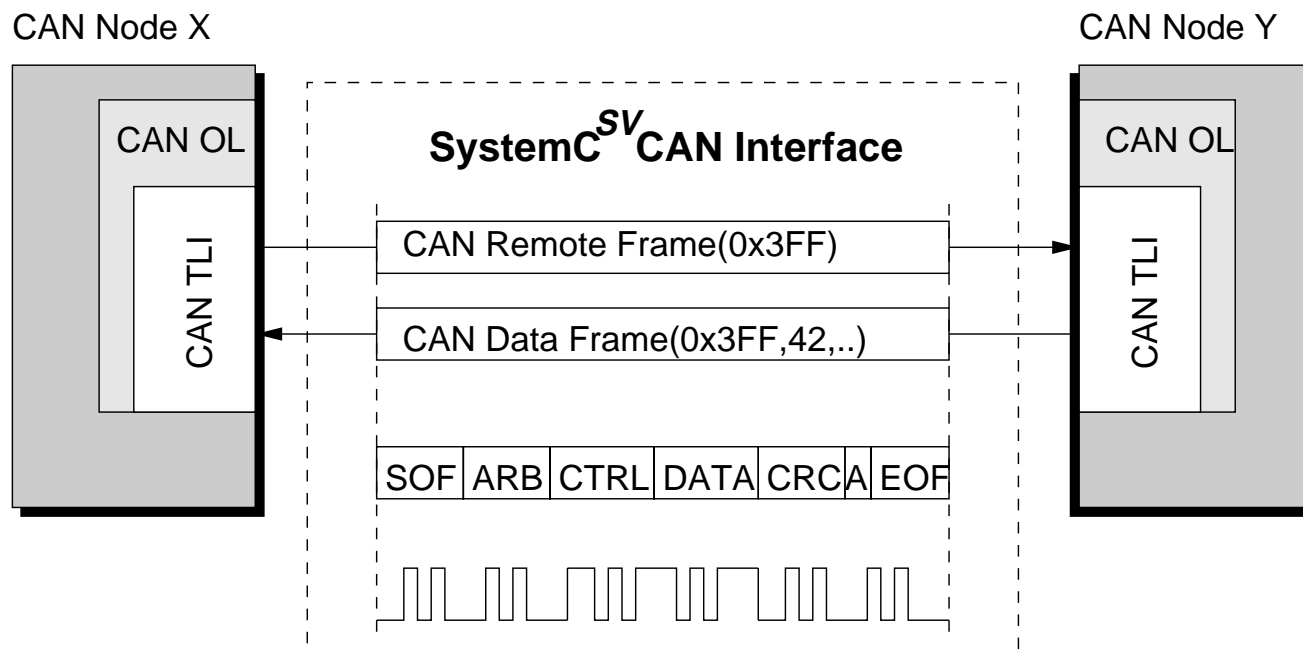
- ❑ Case Study: Network of CAN Nodes at different Levels of Design Abstraction
- ❑ 1. Abstract Behavioural Model with abstracted Communication



X Simulation Speed-Up of abstract System Model: 100x SystemC - RTL

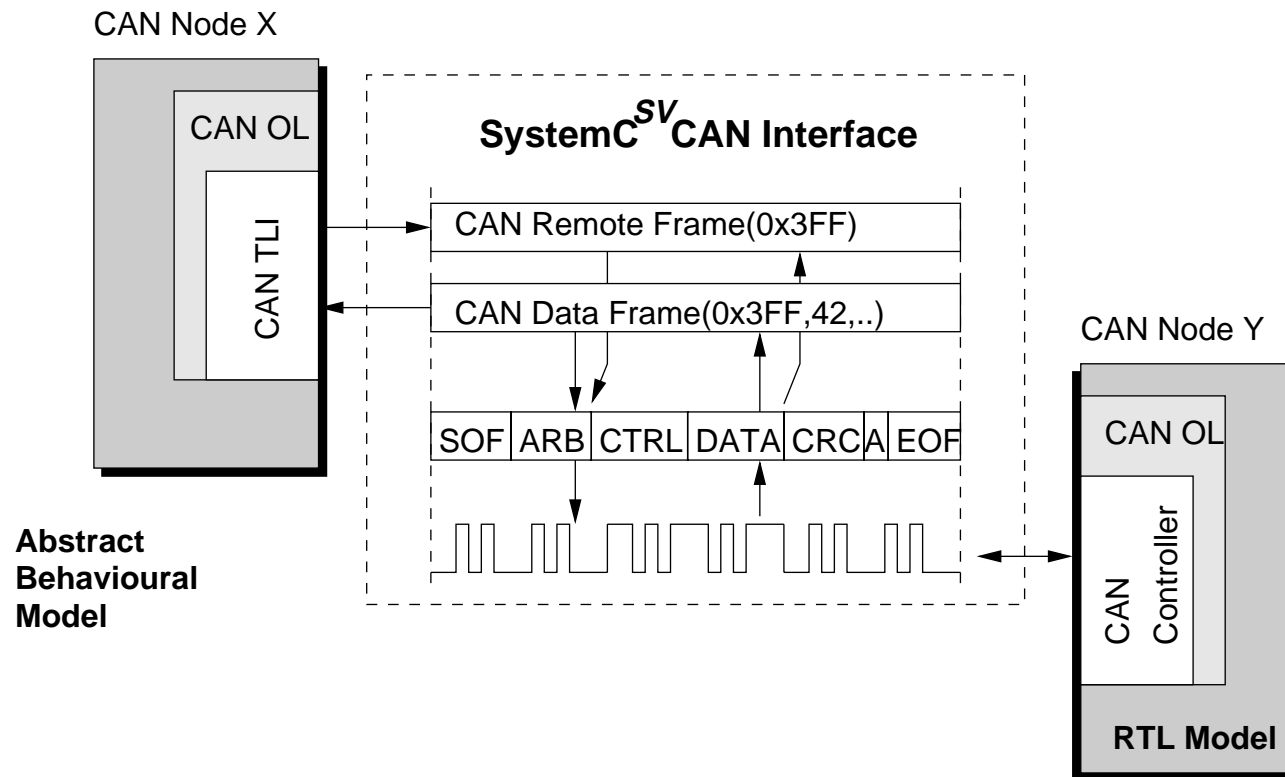
SystemC^{SV} Example: CAN Bus

- ❑ Interface Refinement down to RTL Signals (Protocol Specification added)
- ❑ Module Specifications of CAN Nodes not changed



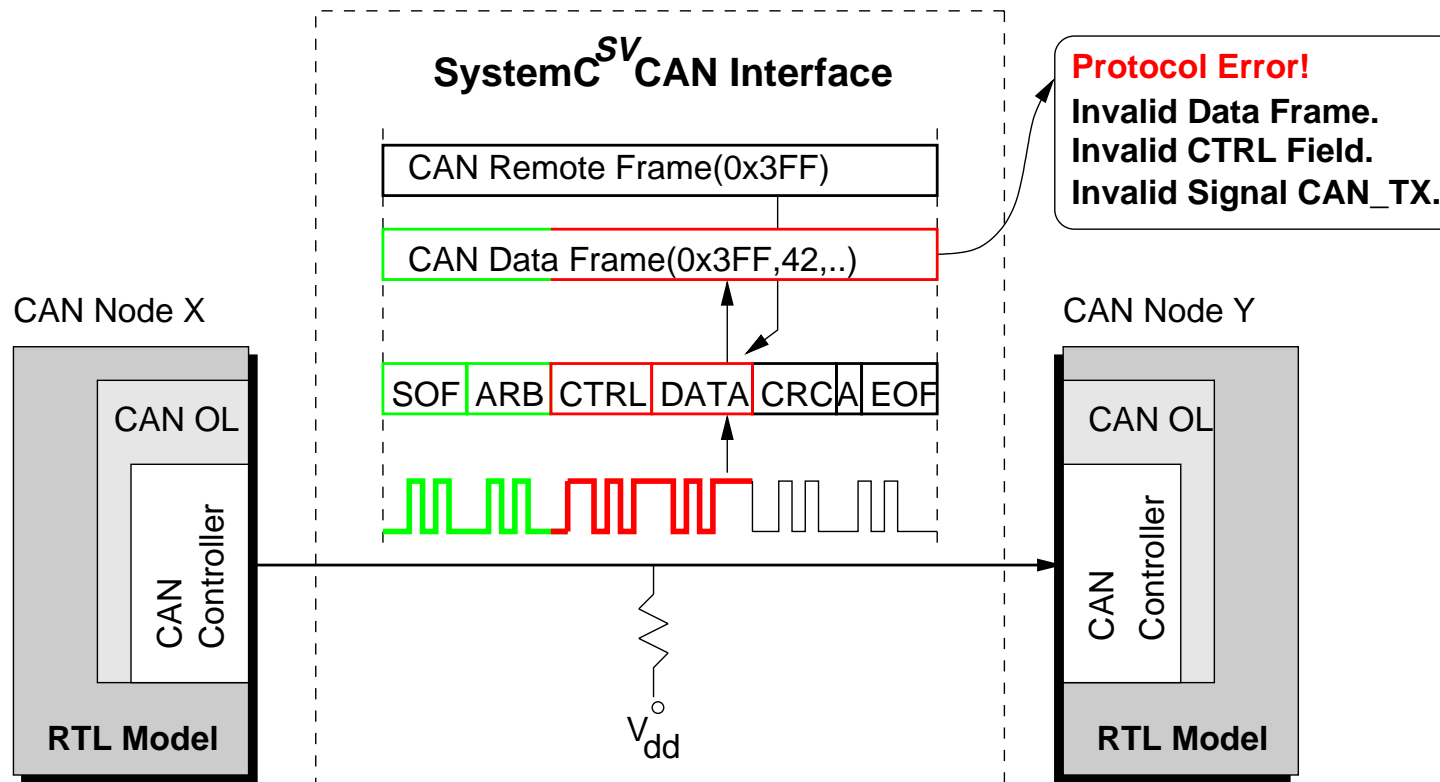
SystemC^{SV} Example: CAN Bus

- ❑ Mixed Abstraction Level Simulation of CAN Nodes
- ❑ Translation between Levels of Communication Abstraction **automatically** performed by SystemC^{SV} CAN Interface

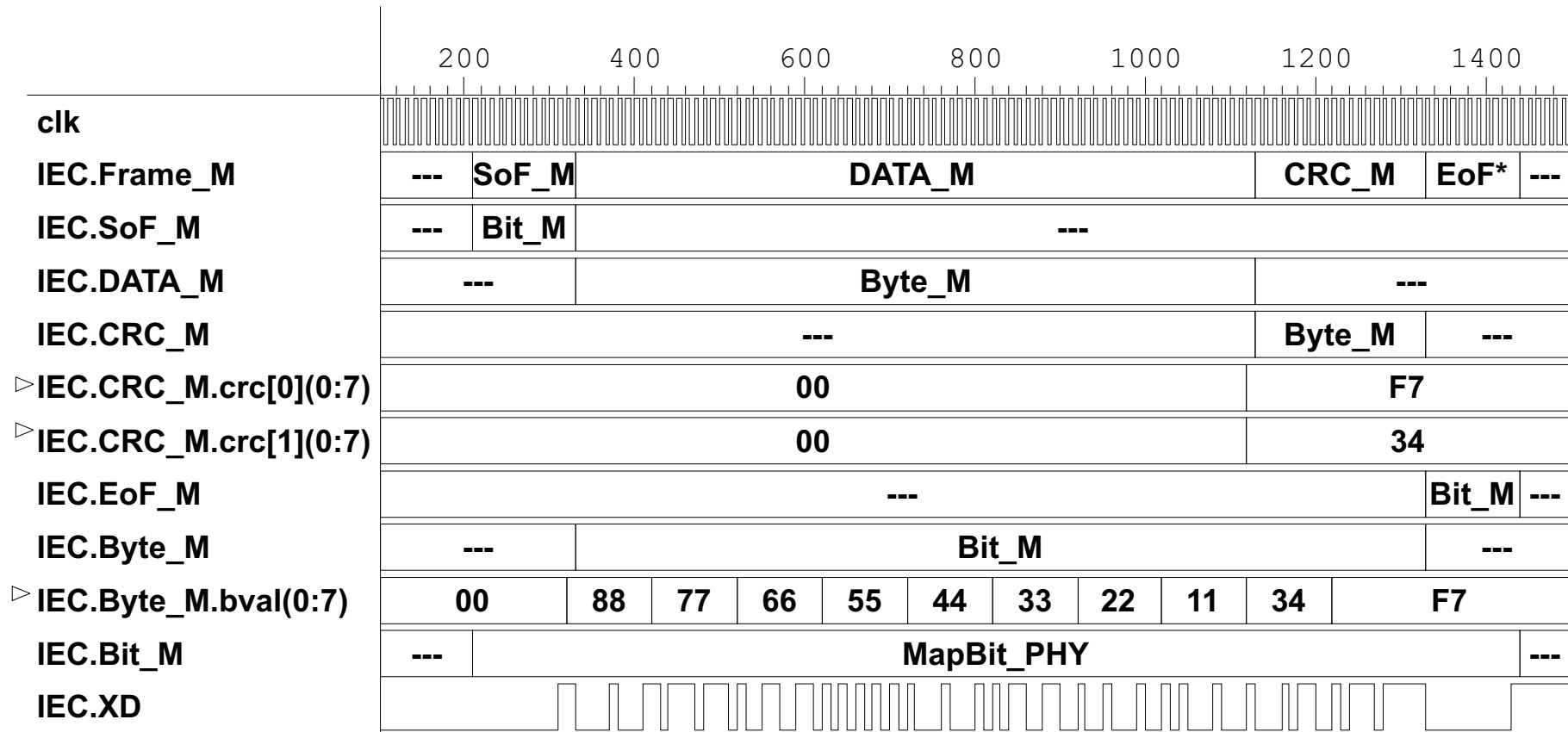


SystemC^{SV} Example: CAN Bus

- ❑ **Implicit Protocol Checking** of RT Signal Level Communication during Bottom-Up Translation of CAN Frames in **SV_INTERFACE**
- ❑ **No extra protocol checker modules required!**



SystemC^{SV} Capabilities-Interface Tracing



❑ SystemC^{SV} Interface trace visualizes Protocol Stack

Conclusions and Future Work

- ❑ **Advantages of the SystemC^{SV} Methodology:**
 - ✗ fast and easy Capturing and Simulation of **Communication Protocols** in System Specification
 - ✗ Design Space Exploration for System Communication Independently from System Functionality
 - ✗ **Automatic up/down Translation** of Communication between all Abstraction Levels (e.g. abstract Model <-> RTL Model)
 - ✗ **Implicit Verification** of Protocol Rules in RTL waveforms during bottom-up execution of Transaction Protocol Stack
 - ✗ **SystemC^{SV} Interfaces fully synthesizable into Hardware!** (Tool is being prepared)
- ❑ **SystemC^{SV} and example models freely available!**

<http://www.tu-chemnitz.de/~rsie/systemc-sv>

Future Developments of SystemC^{SV}

Controlled Item Concurrency

□ currently:

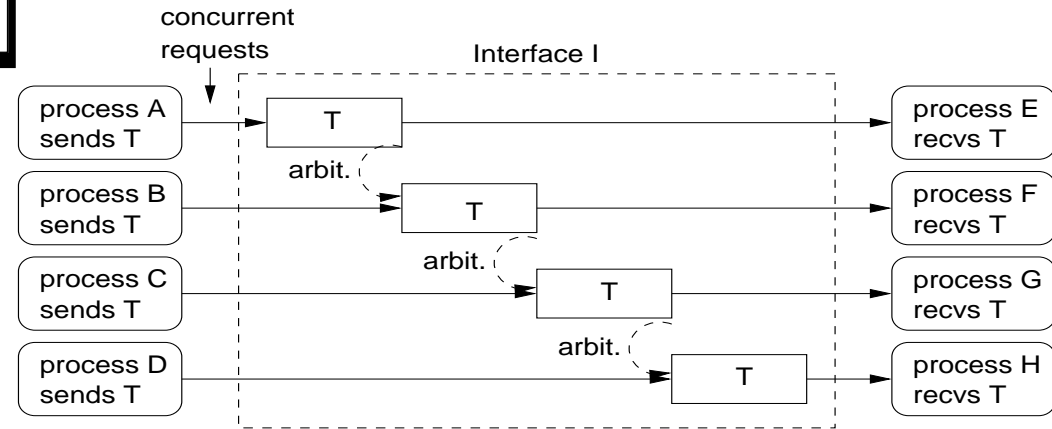
✗ only one Instance per Item

✗ concurrent Requests for same Item not supported

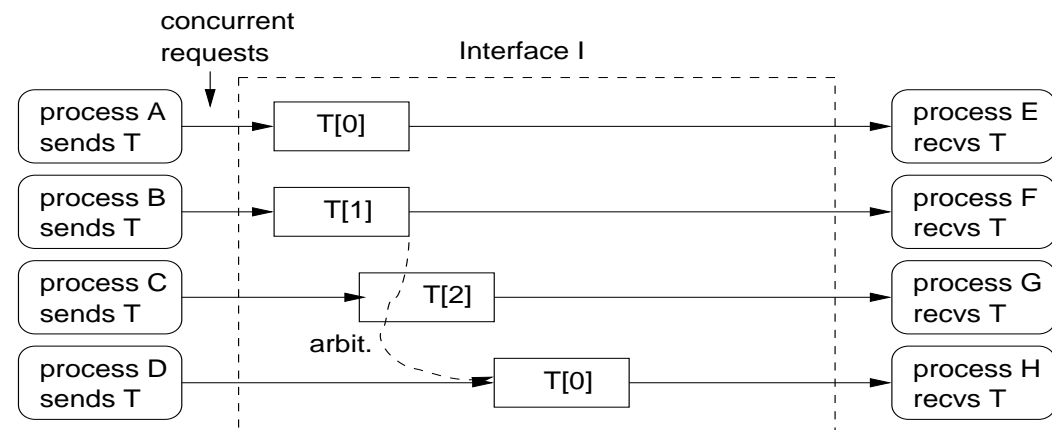
□ Future:

✗ any number of Instances per Item (controllable)

✗ Arbitration of concurrent Requests



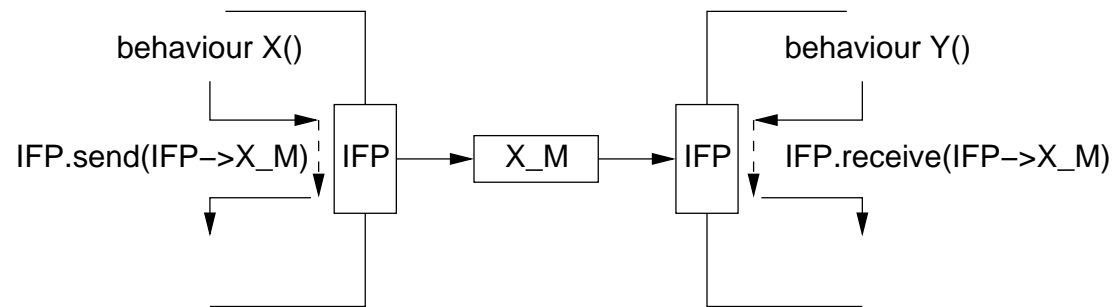
a) Transaction T limited to one instance



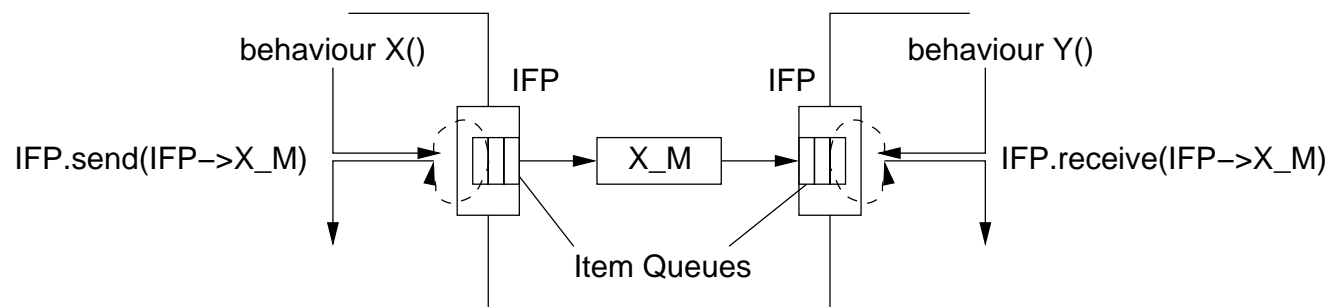
b) Transaction T limited to 3 instances

Future Developments of SystemC^{SV}

□ Pipelined Transactions and Item Queues



a) current implementation of SystemC^{SV} interface ports



b) future implementation of SystemC^{SV} interface ports